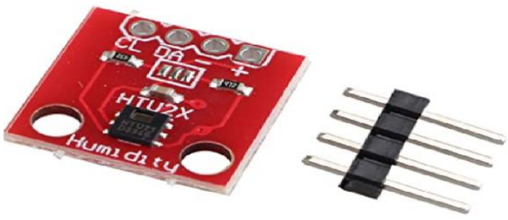
**TIME COSTANT EVALUATION OF A THERMAL SENSOR AND DISTANCE STATIC CALIBRATION OF A TIME OF FLIGHT SENSOR USING THE THERMAL SENSOR FOR A MORE PRECISE EVALUATION**

## 1. TEMPERATURE SENSOR

The sensor used for the thermal acquisition is called HTU21D. The data acquired by the sensor are internally digitalized and then transmitted to the Arduino board through the I2C protocol.
Information about its specs are reported below:



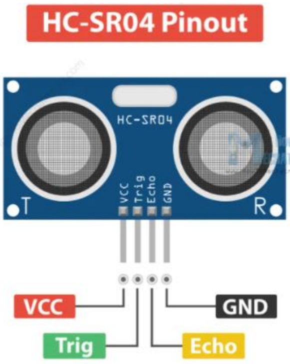

| Characteristics | | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| Resolution | 14 bit | | | 0.01 | | °C |
| | 12 bit | | | 0.04 | | °C |
| Temperature Operating Range | | T | -40 | | +125 | °C |
| Temperature Accuracy @25°C | typ | | | ±0.3 | | °C |
| | max | | See graph 2 | | | °C |
| Replacement | | | fully interchangeable | | | |
| Measuring time (1) | 14 bit | | | 44 | 50 | ms |
| | 13 bit | | | 22 | 25 | ms |
| | 12 bit | | | 11 | 13 | ms |
| | 11 bit | | | 6 | 7 | ms |

The measurement information is saved in 14 bits:

| 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | ACK | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | ACK |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Data (MSB) | | | | | | | | | Data (LSB) | | | | | | Status | | |
| 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | | | | | | | | | |

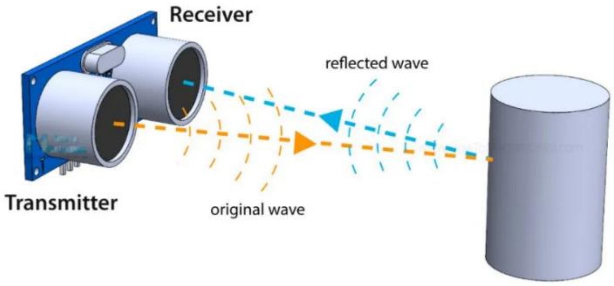
So in order to read correctly the digital value it’s required to link 2 byte and then cancel the last 2 digits

## 2. TIME OF FLIGHT SENSOR




The distance acquisition has been made through the HC-SR04 sensor, made of 2 ultrasonic transducers. Once triggered the start condition with an impulse of 10µs in the trigger pin, the transmitter will send out an 8-cycle ultrasonic burst at 40kHz, which will travel at the speed of sound. Right after sending the $8^{th}$ sonic burst, the Echo pin goes high, going down again once the reflection of the sound wave made by an obstacle has been observed. Once captured, the Echo pin goes down, and through the time passed from these 2 events, it's possible to evaluate the distance according to the following formula:

$$distance = speed * time/2$$




However, since the speed of sound is not constant but it's (also) a function of the temperature, it's possible to obtain better measurement results with the following formula:

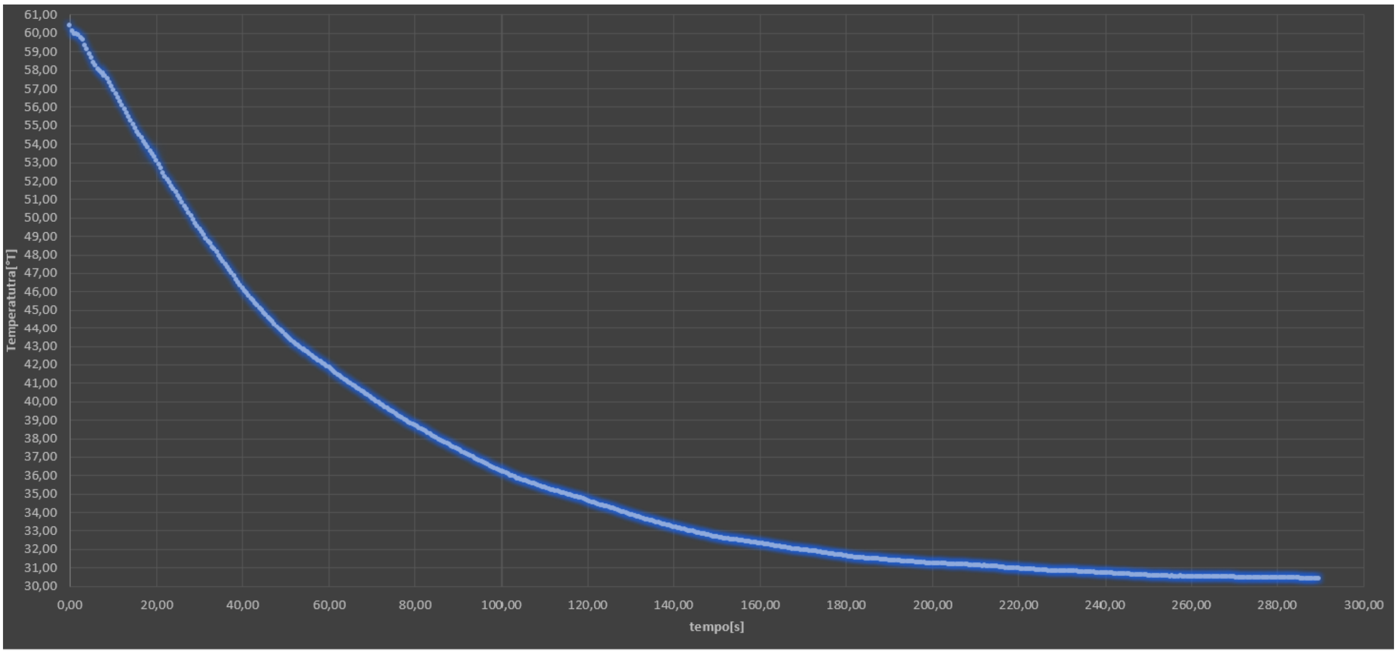
$d = \frac{t}{2} * \sqrt{401.8 * T}$ which is a linear approximation of the overall speed of sound function.

Information about the sensor specs are reported below:

| Electrical Parameters | Value |
|---|---|
| Operating Voltage | 3.3Vdc ~ 5Vdc |
| Quiescent Current | <2mA |
| Operating Current | 15mA |
| Operating Frequency | 40KHz |
| Operating Range & Accuracy | 2cm ~ 400cm ( 1in ~ 13ft) ± 3mm |
| Sensitivity | -65dB min |
| Sound Pressure | 112dB |
| Effective Angle | 15° |
| Connector | 4-pins header with 2.54mm pitch |
| Dimension | 45mm x 20mm x 15mm |
| Weight | 9g |

## 3. TIME CONSTANT EVALUATION

In order to evaluate the time constant I used a sort-of step response. I pre-heated the sensor up to a temperature of 60 degree, and then I it cool down until it reached the room temperature of almost 31 degree. The sensor acquired data every 500ms. Plotting the acquired data the result is the following:




As expected it's the time response of a first order system.

According to the solution of the differential equation for first order systems: $T(t) - T(0) = (T_\infty - T(0)) * (1 - e^{\frac{t}{\tau}})$

It's possible to evaluate its time constant through the following procedure:

$T(t = 0) = 60.44(15)°C$ $\quad$ $T_\infty = 30.44(15)°C$ $\quad$ ← data acquired from the sensor

1) Evaluating T(t) for t= $\tau$

   $$\mathrm{T}(\mathrm{t} = \tau) = \mathrm{T}(0) - 0.632 * (\mathrm{T}(0) - T_\infty) = 41.45(20)^{\circ}C$$

   Where the uncertainty of $\mathrm{T}(\mathrm{t} = \tau)$ has been evaluated as follows:

   $$u(T = \tau) = \sqrt{u(T_0)^2 + 0.632^2 u(T_0)^2 + 0.632^2 u(T_\infty)^2} = 0.20°C$$

   (below it's reported how $u(T_0)$ has been evaluated)

2) Finding the corresponding t:

   t such that $\mathrm{T}(\mathrm{t} = \tau) = 41.45^{\circ}C$ is a $t \cong 62\mathrm{s}$

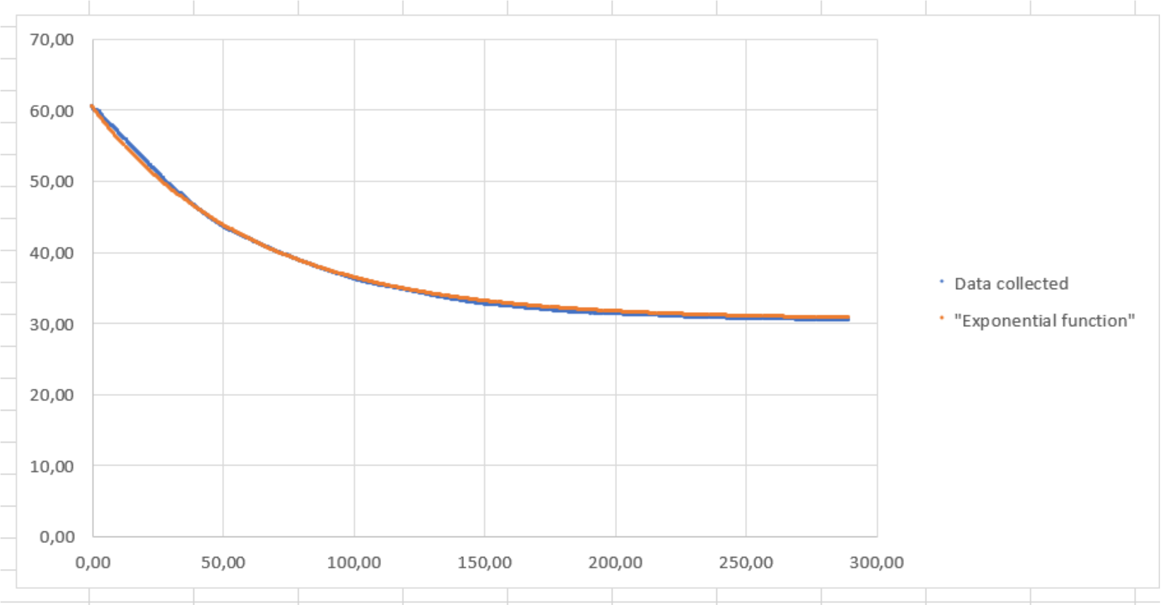
Note: for $t = 3\,\tau = 186\mathrm{s}$ the temperature reached is T = $31{,}50(15)^{\circ}C$, which is the 95% of the final temperature $T_\infty$

*Considerations about the evaluation of u(T):*

In the datasheet is given the resolution of the instrument: 0.03°C, so would be possible to evaluate it's uncertainty considering the standard deviation of a uniform PDF: $u(T) = \frac{res}{\sqrt{12}} = \frac{0.01}{\sqrt{12}} = 0.0029°C$ but it's really low, so I considered other bigger sources of uncertainty. In particular, I hypothesized that the accuracy reported in the datasheet is an extended uncertainty with factor k=2, so that the uncertainty is:

$u(T) = \frac{U}{2} = 0.15°C$ which I think is more reliable.

Plot of both the collected data and the ideal temperature exponential decay with $\tau = 62s$




Note: the Arduino code is not here reported since it's the same of the calibration procedure (that will be later explained) with small differences.

## 4. STATIC CALIBRATION PROCEDURE

### *4.1 Description and Arduino Code*

The ultrasonic sensor has been placed over the floor with a tape meter (millimeter accuracy) next to it. A box has been placed over the meter at 5cm, 10cm and so on up to 2m.

Each measurement consists in the evaluation of the temperature and 10 times of flight measurements. With the combination of these information is possible to compute the distance, according to the formula:

$$d = \frac{t}{2} * v_{sound} \cong \frac{t}{2} * \sqrt{401.8 * T}$$

Where t is the avg value of the 10 measurements.

The Arduino code used for the measurements is here reported:

```
#include <Wire.h>
// distance sensor
const int trigPin = 9;
const int echoPin = 10;

//temperature sensor
const byte Temp_meas_trigger = 0xF3;
const byte Write = 0xE6; //command
const byte Read  = 0xE7; // command
const byte Soft_Reset = 0xFE; //command
const byte Taddress = 0x40;  //indirizzo fisico x lettura
const byte Taddress_write = 0x80; //x scrittura


void setup() {
  Serial.begin(9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  Wire.begin();  //initialize the library wire
  reset(Taddress_write,Soft_Reset);  //temperature sensor reset
}

void loop() {
 float dist = distanza();
 Serial.print("distanza[mm] =");
 Serial.println(dist);
 delay(100);
  exit(0);

}
```

```
void reset(int Taddress, int Soft_reset){
  Wire.beginTransmission(Taddress_write);
  Wire.write(Soft_Reset);
  Wire.endTransmission(true);
  delay(500);
}



float read_temp(int Taddress, int Temp_meas_trigger){  //temperature reading
  Wire.beginTransmission(Taddress);
  Wire.write(Temp_meas_trigger);
  Wire.endTransmission(false);
  delay(100);
    if (Wire.requestFrom(Taddress, 2)!= 2){
      Serial.print("byte non disponibili");
      return 0;
    }
  long V = Wire.read();
  long V2 = Wire.read();
  Wire.endTransmission(true);
  long V3 = estrai_ultimi_2bit(V2);
  V = (V<<8) | V3;
  float Temp = -46.85+175.72*V/pow(2,16);
  //Serial.print();
  return Temp;

}

long estrai_ultimi_2bit(long a){      //pone a 0 gli ultimi due bit
  long x = 0;
  int k=7;
  int vett[8];
  int i = 0;

  for(i=0; i<8; i++)  //inizializzo vettore a 0
    vett[i] = 0;

  for(int i=0; i<8; i++){ //vettore in binario dell'intero a, ma al contrario
    vett[i] = a%2;
    a = a/2;
  }

  for(i=0, k=7; i<4; i++,k--){     //cambio ordine vettore
    int aux = vett[i];
    vett[i] = vett[k];
    vett[k] = aux;
  }
  vett[6] = 0;
  vett[7] = 0;    //ultime 2 cifre a 0
  for(i=0; i<8; i++){
    x = x + vett[i] * potenza(2,7-i);   //valore intero con zero per le ultime 2
cifre binarie
  }

  return x;
}
int potenza(int base, int esponente){
   int i, result = 1;
    for (i = 0; i < esponente; i++){
        result *= base;
    }
        return result;
}
```

```
float distanza(void){
  float vel = vsuono();
  Serial.print("vsuono[m/s] = ");
  Serial.println(vel);
  delay(100);
  float duration = calcolo_tempo();
  float distance = float_duration/2*vel* pow(10,-3);
  return distance;
}

float calcolo_tempo(){  //returns avg time after 10 meas
  long duration[10];
  long somma = 0;
  Serial.print("misure tempo:\n");
  for(int i=0; i<10; i++){
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);
    duration[i] = pulseIn(echoPin, HIGH); //return time in microseconds
    Serial.println(duration[i]);
    delay(100);
  }
  for(int i=0; i<10; i++){
    somma = somma + duration[i];
  }
  float durata_media = (float)somma/10;
  Serial.print("durata media:\n");
  Serial.println(durata_media);
  delay(100);
  return durata_media;
}

float vsuono(){
  float temp = read_temp(Taddress, Temp_meas_trigger) + 273.15;
  float vel = sqrt(401.8*temp); //speed_of_sound function of T formula
  Serial.print("temperatura [K] = ");
  Serial.println(temp);
  delay(100);
  return vel;
}
```

*Some comments:*

●The reset command is used for rebooting the sensor switching the power on and off again. In the datasheet its usage is suggested before acquiring data.

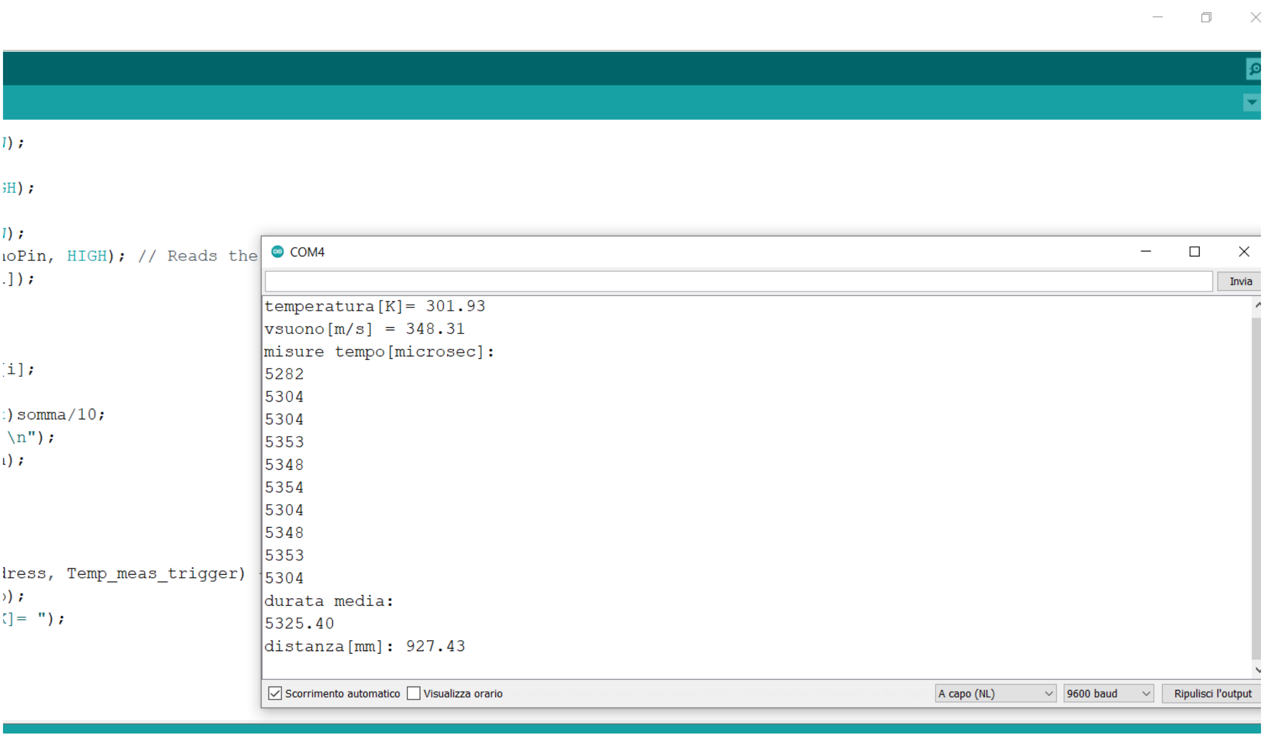
●The HTU21D sensor uses the I2c protocol and returns the temperature measurement in 14 bits, so it's required to link 2 bytes and then cancel the last 2 bits of the LSB. These operations are made by the "estrai_ultimi_2bit" function.

●According to the datasheet of the sensor, given the acquired data, in order to read a temperature the following formula have to be applied:

```
float Temp = -46.85+175.72*V/pow(2,16);
```

which returns a temperature in Celsius degrees.

An example of what is acquired in the serial monitor is here reported:



*4.2 Evaluation of the measurement uncertainty:*

The distance is given by the following formula: $d = \frac{t}{2} * v_{sound} \cong \frac{t}{2} * \sqrt{401.8 * T}$

So in order to compute the uncertainty, the Propagation law have to be applied:

$u_y = \sqrt{\sum_{i=1}^{n} (\frac{\partial f}{x_i} * u_{x_i})^2}$ which in our case consist in:

$u(d)_2 = \sqrt{\left(\frac{\partial d}{\partial t} * u(t)\right)^2 + \left(\frac{\partial d}{\partial T} * u(T)\right)^2}$ where $\frac{\partial d}{\partial t} = \frac{1}{2} * \sqrt{401.8 * T}$ and $\frac{\partial d}{\partial T} = \frac{t}{4\sqrt{T}} * \sqrt{401.8}$

For each distance I computed 10 time measurements, so the uncertainty (type A) is evaluated as follows:

$\bar{t} = \frac{1}{N} * \sum_{i=1}^{N} (t_i)$

$s^2(t) = \frac{1}{N-1} \sum_{i=1}^{N} (t_i - \bar{t})^2 \quad u(t) = \sqrt{s^2(t)/n}$

As for the thermal sensor, I also considered as source of uncertainty the accuracy reported in the datasheet, considered as extended uncertainty with factor k=2, so that $u_2(d) = U/2 = 1.5mm$

Finally I evaluated the combined uncertainty $u_c(d) = \sqrt{u(d)_1^2 + u(d)_2^2}$

*4.3 uncertainty results with respect to measured data*

Here are reported the main results of the acquisition procedure and the evaluation of the uncertanty.

(avg) measured times and temperatures:

| dist(mm) | tmedio(µs) | dev standard | incertezza[µs] | meas T [K] | incertezza T[°C] |
|---|---|---|---|---|---|
| 50 | 297 | 1,897366596 | 1 | 302,32 | 0.15 |
| 100 | 613 | 3,265986324 | 1 | 302,33 | |
| 150 | 894 | 3,341656276 | 1 | 302,34 | |
| 200 | 1167 | 3,0713732 | 1 | 302,35 | |
| 250 | 1465 | 4,423422506 | 1 | 302,36 | |
| 300 | 1763 | 6,522610248 | 2 | 302,37 | |
| 350 | 2014 | 9,818350167 | 3 | 302,38 | |
| 400 | 2315 | 11,42171421 | 4 | 302,39 | |
| 450 | 2576 | 6,583649781 | 2 | 302,40 | |
| 500 | 2846 | 5,73585216 | 2 | 302,41 | |
| 550 | 3161 | 17,00196067 | 5 | 302,42 | |
| 600 | 3427 | 18,57597014 | 6 | 302,43 | |
| 650 | 3701 | 14,41449887 | 5 | 302,44 | |
| 700 | 4002 | 36,37703793 | 12 | 302,45 | |
| 750 | 4272 | 22,44128141 | 7 | 302,46 | |
| 800 | 4532 | 37,65766854 | 12 | 302,47 | |
| 850 | 4832 | 30,75060975 | 10 | 302,48 | |
| 900 | 5201 | 46,26541785 | 15 | 302,49 | |
| 950 | 5323 | 22,80570104 | 7 | 302,50 | |
| 1000 | 5727 | 48,16649366 | 15 | 302,51 | |
| 1050 | 5367 | 64,06498437 | 20 | 302,52 | |
| 1100 | 6338 | 50,69834536 | 16 | 302,53 | |
| 1150 | 6606 | 53,73598008 | 17 | 302,54 | |
| 1200 | 895 | 51,4224335 | 16 | 302,55 | |
| 1250 | 7096 | 24,67567403 | 8 | 302,56 | |
| 1300 | 7391 | 67,7266565 | 21 | 302,57 | |
| 1350 | 7671 | 51,90118817 | 16 | 302,58 | |
| 1400 | 7953 | 56,07148216 | 18 | 302,59 | |
| 1450 | 8223 | 50,68157675 | 16 | 302,60 | |
| 1500 | 8636 | 56,53749985 | 18 | 302,61 | |
| 1550 | 8800 | 62,17904256 | 20 | 302,62 | |
| 1600 | 9079 | 25,83042994 | 8 | 302,63 | |
| 1650 | 9387 | 72,3328725 | 23 | 302,64 | |
| 1700 | 9654 | 39,96442863 | 13 | 302,65 | |
| 1750 | 9980 | 17,38581798 | 5 | 302,66 | |
| 1800 | 10257 | 51,00119824 | 16 | 302,67 | |
| 1850 | 10534 | 47,10508347 | 15 | 302,68 | |
| 1900 | 10786 | 43,48230802 | 14 | 302,69 | |
| 1950 | 11110 | 34,25460229 | 11 | 302,70 | |
| 2000 | 11413 | 48,66712329 | 15 | 302,71 | |

$u_1(d)$ computation:

$$u_{d1} = \sqrt{(c1(T) * u(t))^2 + (c2(t,T) * u(T))^2}$$

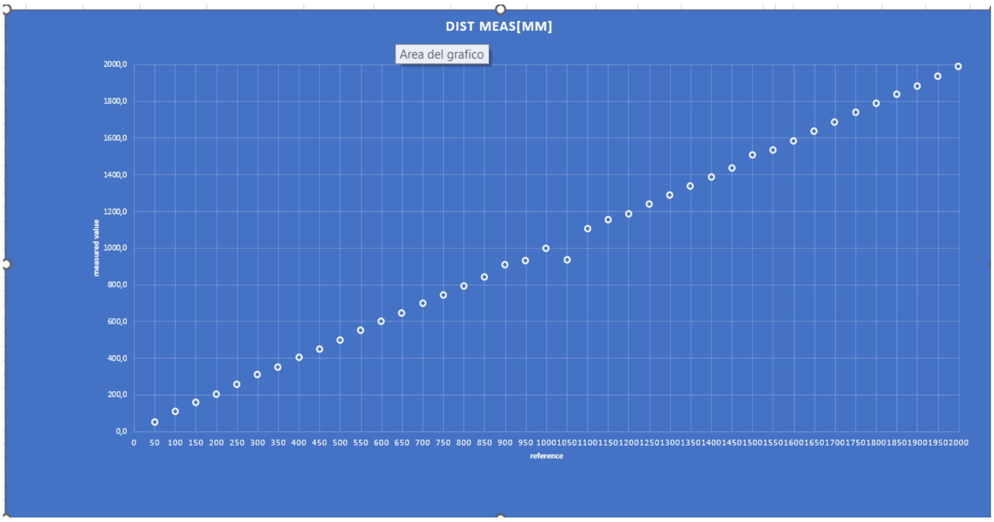
| c1(T) | c2(t,T) | u1(d)[m] |
|---|---|---|
| 174,26 | 8,6E-05 | 0,00011 |
| 174,27 | 0,00018 | 0,00018 |
| 174,27 | 0,00026 | 0,00019 |
| 174,27 | 0,00034 | 0,00018 |
| 174,28 | 0,00042 | 0,00025 |
| 174,28 | 0,00051 | 0,00037 |
| 174,28 | 0,00058 | 0,00055 |
| 174,28 | 0,00067 | 0,00064 |
| 174,29 | 0,00074 | 0,00038 |
| 174,29 | 0,00082 | 0,00035 |
| 174,29 | 0,00091 | 0,00095 |
| 174,30 | 0,00099 | 0,00104 |
| 174,30 | 0,00107 | 0,00081 |
| 174,30 | 0,00115 | 0,00201 |
| 174,30 | 0,00123 | 0,00125 |
| 174,31 | 0,00131 | 0,00209 |
| 174,31 | 0,00139 | 0,00171 |
| 174,31 | 0,0015 | 0,00256 |
| 174,32 | 0,00153 | 0,00128 |
| 174,32 | 0,00165 | 0,00267 |
| 174,32 | 0,00155 | 0,00354 |
| 174,32 | 0,00183 | 0,00281 |
| 174,33 | 0,0019 | 0,00298 |
| 174,33 | 0,00026 | 0,00284 |
| 174,33 | 0,00204 | 0,0014 |
| 174,34 | 0,00213 | 0,00375 |
| 174,34 | 0,00221 | 0,00289 |
| 174,34 | 0,00229 | 0,00312 |
| 174,34 | 0,00237 | 0,00282 |
| 174,35 | 0,00249 | 0,00315 |
| 174,35 | 0,00253 | 0,00346 |
| 174,35 | 0,00262 | 0,00149 |
| 174,36 | 0,0027 | 0,00401 |
| 174,36 | 0,00278 | 0,00225 |
| 174,36 | 0,00287 | 0,00108 |
| 174,37 | 0,00295 | 0,00286 |
| 174,37 | 0,00303 | 0,00265 |
| 174,37 | 0,00311 | 0,00246 |
| 174,37 | 0,0032 | 0,00197 |
| 174,38 | 0,00329 | 0,00274 |

Measured distance and combined uncertainty:

| J | K | L | M | N |
|---|---|---|---|---|
| dist(mm) | dist meas[mm] | u1[mm] | u2[mm] | combined u [mm] |
| 50 | 51,75 | 0,1 | 1,50 | 1,5 |
| 100 | 106,82 | 0,2 | | 1,5 |
| 150 | 155,62 | 0,2 | | 1,5 |
| 200 | 203,37 | 0,2 | | 1,5 |
| 250 | 255,29 | 0,3 | | 1,5 |
| 300 | 307,05 | 0,4 | | 1,5 |
| 350 | 350,97 | 0,5 | | 1,6 |
| 400 | 403,43 | 0,6 | | 1,6 |
| 450 | 448,75 | 0,4 | | 1,5 |
| 500 | 495,82 | 0,3 | | 1,5 |
| 550 | 550,70 | 0,9 | | 1,8 |
| 600 | 597,08 | 1,0 | | 1,8 |
| 650 | 645,02 | 0,8 | | 1,7 |
| 700 | 697,32 | 2,0 | | 2,5 |
| 750 | 744,36 | 1,3 | | 2,0 |
| 800 | 789,67 | 2,1 | | 2,6 |
| 850 | 842,15 | 1,7 | | 2,3 |
| 900 | 906,46 | 2,6 | | 3,0 |
| 950 | 927,72 | 1,3 | | 2,0 |
| 1000 | 998,15 | 2,7 | | 3,1 |
| 1050 | 935,23 | 3,5 | | 3,8 |
| 1100 | 1104,66 | 2,8 | | 3,2 |
| 1150 | 1151,39 | 3,0 | | 3,3 |
| 1200 | 1186,58 | 2,8 | | 3,2 |
| 1250 | 1236,75 | 1,4 | | 2,0 |
| 1300 | 1288,21 | 3,7 | | 4,0 |
| 1350 | 1336,87 | 2,9 | | 3,2 |
| 1400 | 1385,95 | 3,1 | | 3,5 |
| 1450 | 1433,18 | 2,8 | | 3,2 |
| 1500 | 1505,16 | 3,1 | | 3,5 |
| 1550 | 1533,62 | 3,4 | | 3,8 |
| 1600 | 1582,22 | 1,5 | | 2,1 |
| 1650 | 1635,90 | 4,0 | | 4,3 |
| 1700 | 1682,67 | 2,2 | | 2,7 |
| 1750 | 1739,35 | 1,1 | | 1,8 |
| 1800 | 1787,63 | 2,8 | | 3,2 |
| 1850 | 1836,09 | 2,6 | | 3,0 |
| 1900 | 1879,87 | 2,4 | | 2,9 |
| 1950 | 1936,56 | 1,9 | | 2,5 |
| 2000 | 1989,34 | 2,7 | | 3,1 |

*4.4 Calibration procedure:*

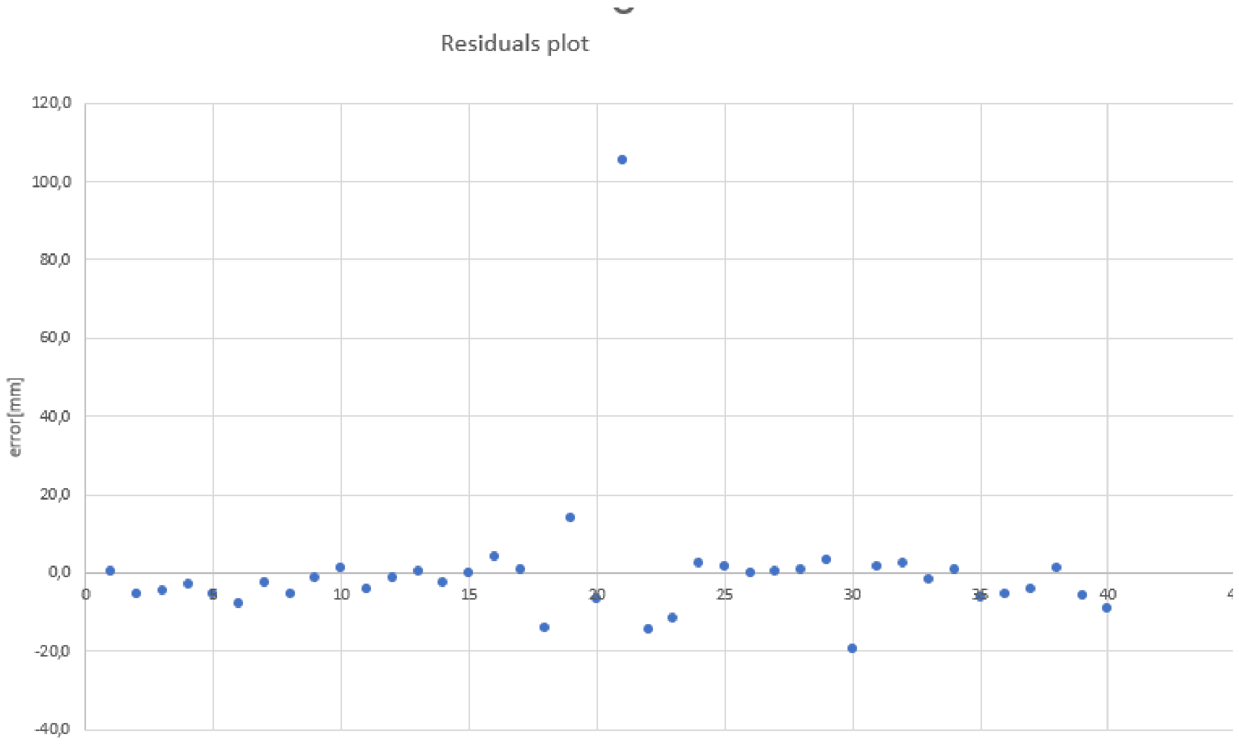
Plotting the measured distance on y axis and the reference distance on x axis the result is the following:




It's clearly a linear relationship so I used a linear regression model that minimizes the quadratic distance between the line and the points:


The index $R^2$ = 0.999 and the angular coefficient is 0.9888, so the line is a good approximation of the real behaviour.

Plotting the residuals:




Can be seen that they are randomly distributed, so again the regression is a “good” one.

Notice also that a point is very far from the others, probably due to an error during the data acquisition procedure. Could be possible to obtain even better results cancelling that point from the regression equation.