DSBM - Laboratori 5: Incorporació de nous sensors

Victor Expósito Griñán, Francesc Ruiz Tuyà

4 de maig de 2021

1 Introducció

L'objectiu d'aquesta sessió de laboratori és incorporar diferents tipus de sensors al nostre sistema Arduino. En concret, disposàvem d'un acceleròmetre ADXL330 i un sensor d'ultrasons HC SR04. Nosaltres hem escollit utilitzar el segon, ja que ja tenim experiència amb acceleròmetres.

2 Codi Arduino - Calibració del sensor

Primer de tot, per veure si el sensor està ben ajustat, hem realitzat una sèrie de mesures (amb ajuda d'un DINA3 que conté distàncies marcades fins a 40 cm) i recol·lectat les dades captades pel sensor. El rang del dispositiu és de 3 cm fins a quasi 4 m (amb distàncies fora d'aquest rang el sensor no capta bé els senyals de resposta).

```
float time_to_cm(unsigned long t) {
    return (float(t)/1000000) * velocitatSo / 2;
}

float get_dist() {
  digitalWrite(triggerPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(triggerPin, LOW);
  tickStartTrigger = micros();
  while (digitalRead(echoPin) == LOW);
  tickStartEcho = micros();
  while (digitalRead(echoPin) == HIGH);
  tickEnd = micros();
  elapsedTimeTrigger = tickEnd - tickStartTrigger;
  elapsedTimeEcho = tickEnd - tickStartEcho;
  //Serial.print("trigger: ");
  //Serial.print(time_to_cm(elapsedTimeTrigger));
  //Serial.print(" echo: ");
  return time_to_cm(elapsedTimeEcho);
}
```

La funció *get_dist* és la més important del codi, ja que s'encarrega de la captació de dades utilitzant els pins de trigger i echo del sensor. La idea darrera aquest codi és implementar el format de comunicació mostrat a la Figura 1. Per acabar de representar les dades obtingudes en format distància, hem de fer la conversió a cm (funció *time_to_cm*), usant la velocitat del so (34320 cm/s aprox.).

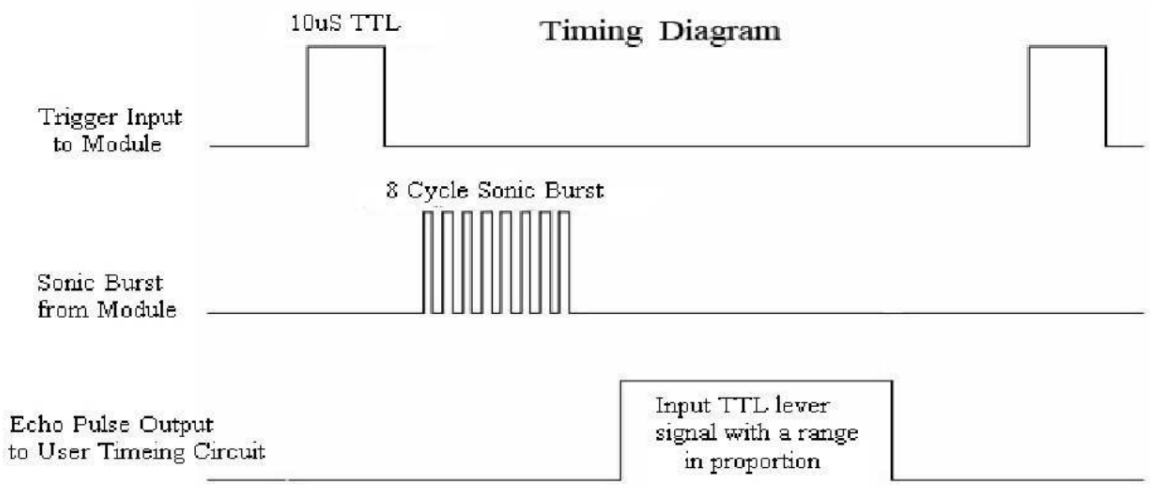

Figura 1: Diagrama de comunicació del sensor HC SR04

El motiu pel qual hi ha dues variables anomenades *elapsedTimeTrigger* i *elapsedTimeEcho* és a causa de la interpretació que es pot fer de la mesura temporal (i per tant de distància). Si s'interpreta que la mesura comença al flanc ascendent del senyal emès pel trigger (*tickStartTrigger*) el resultat és *elapsedTimeTrigger*, mentre que si s'interpreta que la mesura comença al flanc ascendent del senyal echo (*tickStartEcho*) el resultat és *elapsedTimeEcho*. Per sortir de dubtes, hem provat les dos variants, obtenint unes conclusions bastant fermes, i és que usant *elapsedTimeTrigger* la mesura de distància té un error de més de 6 cm. És per això que la millor alternativa és emprar *elapsedTimeEcho*.

```
float get_avg() {
  float avg = 0;
  float aux;
  for (int i = 0; i < num_samples; ++i) {
      aux = get_dist();
      if (aux > max_dist) --i;
      else avg += aux;
      Serial.println(aux);
  }
  return avg / num_samples;
}

void loop() {
  delay(5000);
  float inc = 5;
  for (int i = 0; i < 8; ++i){
    real_dist = 5 + inc * i;
    max_dist = real_dist + 20;
    float avg = get_avg();
    Serial.print(real_dist);
    Serial.print(" -> ");
    Serial.print(avg);
    Serial.print('\n');
    delay(5000);
  }
  while(true);
}
```

Mesura Real	Mesura Obtinguda pel Sensor
5 cm	5.06 cm
10 cm	10.69 cm
15 cm	15.3 cm
20 cm	20.19 cm
25 cm	25.47 cm
30 cm	30.15 cm
35 cm	35.09 cm
40 cm	40.42 cm

Taula 1: Taula de calibració del sensor

Una vegada fetes les comprovacions, hem elaborat la taula de calibració del sensor amb la funció *get_avg*, que realitza la mitjana de *num_samples* mesures (per obtenir una mesura més fiable) per cada distància.

(El codi de calibració del sensor es troba a l'arxiu *sensorUltrasonsDades*).

3 Codi Arduino - Mesura de Velocitat

Una de les aplicacions que es poden implementar i que hem desenvolupat amb el sensor d'ultrasons és la mesura de velocitat d'un objecte.

```
float get_dist() {
  digitalWrite(triggerPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(triggerPin, LOW);
  tickStartTrigger = micros();
  while (digitalRead(echoPin) == LOW);
  tickStartEcho = micros();
  while (digitalRead(echoPin) == HIGH);
  tickEnd = micros();
  elapsedTimeTrigger = tickEnd - tickStartTrigger;
  elapsedTimeEcho = tickEnd - tickStartEcho;
  t = ((tickEnd + tickStartEcho) / 2);
  return time_to_cm(elapsedTimeEcho);
}

void loop() {
  float x0 = get_dist();
  float t0 = t;
  while (true) {
    delay(1000);
    float x1 = get_dist();
    float t1 = t;
    float inc_dist = x1 - x0;
    float inc_t = (t1 - t0) / 1000000;
    Serial.println( inc_dist / inc_t );
    x0 = x1;
    t0 = t1;
  }
}
```

Com ja sabem que el sensor proporciona unes mesures raonables, només hem hagut de modificar el codi del loop, fent una petita modificació a *get_dist* per a que pugui guardar el temps mesurat anteriorment i fer així un increment de temps.

El càlcul de la velocitat consisteix en agafar amb una certa freqüència valors de distància amb *get_dist* i valors de temps amb l'increment mencionat i dividir-los per obtenir una velocitat. Fent proves amb diversos objectes, hem comprovat que el codi funciona bastant bé, encara que hi ha un factor determinant a la hora de calcular la velocitat de l'objecte, la freqüència de mostreig. Al nostre codi, aquest valor es pot saber mitjançant la línia *delay(1000)* dins el loop, el qual ens diu que es fa una mesura cada segon. Després de fer diverses proves amb diferents números, hem escollit aquest valor ja que considerem que és el que ofereix uns resultats més òptims.

(El codi per mesurar velocitats es troba a l'arxiu *sensorUltrasons*).