GUIDE SUR L'UTILISATION DES CAPTEURS

Tome 2

Réalisé par Thomas MARTINEZ

24 avril 2020

Ce deuxième tome a pour but de vous aider à utiliser des capteurs sur des cartes arduino. Dans ce tutoriel, nous nous concentrerons sur les deux des cinq types de capteurs vus dans l'article précédent.

Introduction

Dans le précédent tutoriel, nous avons vu comment choisir un capteur selon différentes caractéristriques : La précision, la stabilité, le coût...

Nous avions également travailler sur cinq types de capteur assez couremment utilisés en expliquant leur principe de fonctionne et en donnant quelques exemples d'utilisation. À présent, nous allons voir ensemble comment utiliser les capteurs à ultrasons à infrarouge pour le contrôle à distance.

Dans le prochain tome, nous verrons comment utiliser le capteur d'humidité ainsi que les capteurs laser et de contact (de type TOR).

Pour bien comprendre ce tutoriel, il est nécessaire d'avoir quelques bases sur les capteurs et sur le fonctionnement d'une carte arduino (et en langage C++). C'est pourquoi nous recommandons d'avoir déjà étudér les tutoriels traitant de ces sujets avant de commencer celui-ci.

Table des matières

⋆ ⋆ ⋆

A Les capteurs analalogiques/numériques

A.1 Le capteur à ultrasons

i Le branchement

Pour notre explication, nous utiliserons comme exemple une carte arduino UNO et un capteur ultrasons HC-SR04 car ces deux composants sont assez répandus et fonctionnent de manière standard.

FIGURE 1 – Capteur à ultrasons – Modèle HC-SR04

On peut voir que le capteur est muni de 4 broches :

- La broche "**VCC**" pour l'alimentation que l'on va relier au pin 5V de la carte arduino.
- La broche "**Trigg**" qui devra être pilotée par la carte arduino (pin de sortie donc), on peut la relier au pin D2 par exemple.
- La broche "**Echo**" sera elle à brancher à un pin paramètré en entrée. On peut par exemple la relier au pin D3
- La broche "**GND**" est à relier au ground de la carte arduino.

À Savoir

Ici les pins D2 et D3 ont été choisies arbitrairement, tout autre pin paramètrable de la carte est utilisable.
Une bonne connaissance des bases de l'électricité peuvent vous permettre de faire des montages plus compliqués (si il n'y a pas que ce capteur sur la carte arduino par exemple).

On se retrouve ainsi avec le montage suivant :

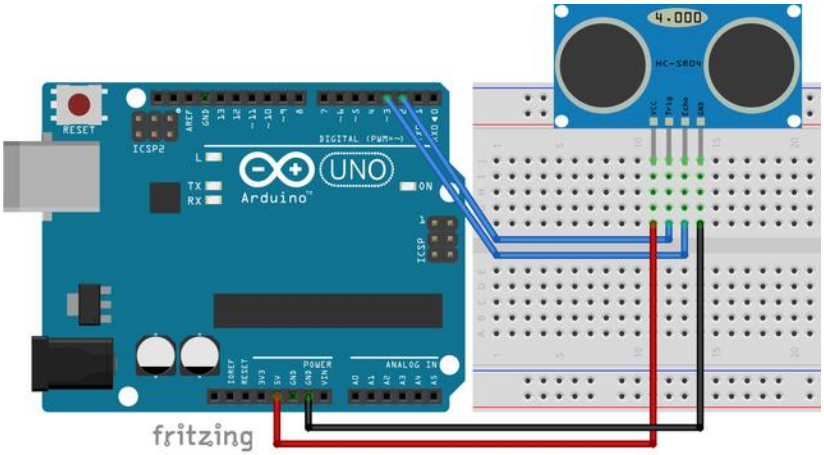

FIGURE 2 – Montage de notre exemple

ii Le code

Pour commencer, nous allons devoir déclarer quatres constantes :

- Deux constantes pour nos pins D2 et D3
- Une constante pour la vitesse du son
- Une constante pour mesurer la durée que l'onde à mise à être captée et renvoyée sur le pin D3

Codage

```
1  /* Constantes pour les broches */
2  const byte TRIGGER_PIN = 2; // Broche TRIGGER
3  const byte ECHO_PIN = 3; // Broche ECHO
4
5  /* Constantes pour le timeout */
6  const unsigned long MEASURE_TIMEOUT = 25000UL;
7
8  /* Vitesse du son dans l air en mm/us */
9  const float SOUND_SPEED = 340.0 / 1000;
```

Le Timeout correspond au temps d'écoute de l'arduino. 25ms correspond à peu près au temps que prendrait l'onde pour détecter un objet à 4 mètres.

On initialise les pins et le mode de la carte UNO

Codage

```
1
2  void setup()
3
4    /* Initialise le port serie */
5    Serial.begin(115200);
6
7    /* Initialise les broches */
8    pinMode(TRIGGER_PIN, OUTPUT);
9    digitalWrite(TRIGGER_PIN, LOW); // La broche TRIGGER est a l etat LOW au
         repos
10   pinMode(ECHO_PIN, INPUT);
```

Le programme qui suit envoit une impulsion sur la broche TRIGG pour générer l'onde puis le programme observe le temps mis pour détecter l'écho. Par un calcul simple on trouve la distance séparant le capteur et l'obstacle. On peut ajouter une ligne pour ajouter l'évènement "il n'y a pas d'obstacle" et pouvoir le traiter comme on le souhaite ensuite notammant avec la fonction pulseIN().

Codage

```
1  void loop() {
2
3    /* Lance une mesure en envoyant une impulsion HIGH de 10us sur la broche
         TRIGGER */
4    digitalWrite(TRIGGER_PIN, HIGH);
5    delayMicroseconds(10);
6    digitalWrite(TRIGGER_PIN, LOW);
7
8    /* Mesure le temps entre l'envoi de l'impulsion ultrasonique et son echo (si il
         existe) */
9    long measure = pulseIn(ECHO_PIN, HIGH, MEASURE_TIMEOUT);
10
11   /* Calcul la distance a partir du temps mesure */
12   float distance_mm = measure / 2.0 * SOUND_SPEED;
13
14   /* Delai d'attente pour eviter d'afficher trop de resultats a la seconde */
15   delay(500);
```

Le programme complet en arduino

A.2 Le capteur infrarouge

i Le branchement

Nous allons voir comment utiliser un capteur infrarouge pour le pilotage à distance. Nous nous appuierons sur le modèle T38238 de capteur IR pour la suite.

FIGURE 3 – Récepteur IR modèle TSOP38238

Ici, nous observons que ce système possède trois branches :

- Le GND et l'alimentation 5V.
- La broche OUT à connecter sur un pin paramétré en tant qu'entrée qui est initialement à l'état HIGH.

Lorsque ce capteur reçoit un signal infrarouge, sa sortie passe à l'état LOW. Ce système acquiert les informations en tant que TOR puis les traite. On peut par exemple faire de la détection d'obstacle sur le chemin émetteur/récepteur avec le changement d'état de la sortie et ensuite utilsier cette information pour commander.

Cependant, nous pouvons utiliser ce système pour une multitude de commandes. En effet, en connaissant la fréquence d'émission de l'émetteur IR, on peut stocker les infromations envoyées et les décripter dans le programme. Cette méthode, plus complexe permet néanmoins de pouvoir effectuer plus de tâches avec un seul capteur. À savoir que les télécommandes du marché ne suivent pas les même normes et n'envoient donc pas les mêmes signaus pour une même donnée. Il faut alors se fixer sur une télécommande pour ne plus avoir à changer la partie programme. Le principe est le suivant :

- Nous intialisisons la carte arduino, nos constantes de travail, le pin de lecture et l'ensemble des codes que l'émetteur envoie et que nous utilisons

- On fait une boucle qui attend un signal
- Lorsqu'un signal est détecté, on stock les données lues sur le pin avec la même cadence qu'à l'émission (généralement, les télécommandes émettent à 38kHz)
- Puis on regarde dans notre tableau de codes recensés quelle est la commande désirée

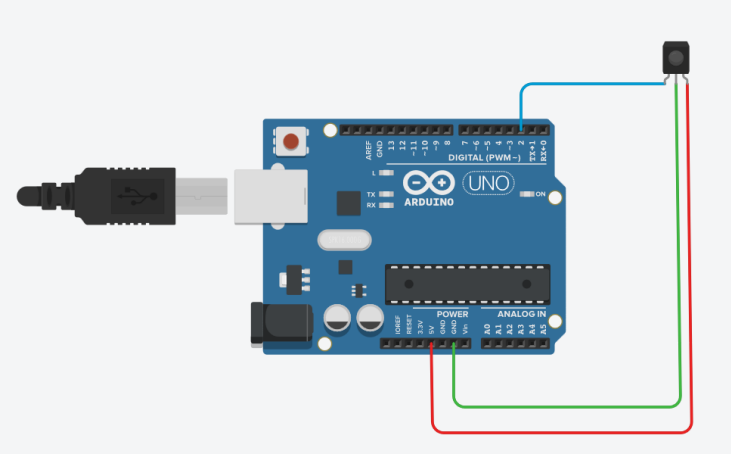

FIGURE 4 – Récepteur IR modèle TSOP38238

ii Le code

Pour utiliser notre récepteur, on suppose que les codes de l'émetteur sont connus. Il existe néanmoins des moyens de décoder une télécommande, mais ce n'est pas la partie traitée ici.

Voici un exemple de programme pour décoder une télécommande :

Codage

```
1  const int pin = 8; // pin de lecture du TSOP
2  const int nbMax = 64; // nombre de lectures
3  const int temporisation=25;
4  unsigned int tableau[nbMax][2]; // tableau pour stocker les valeurs hautes et
       basses
5
6  void setup() {
7    pinMode(pin, INPUT); // mode INPUT pour le pin de lecture
8    Serial.begin(9600); //initialisation communication serie.
9  }
10
11 void loop() {
12   if (digitalRead(pin)) { //si etat haut capte
13     lecture(); // appel de la fonction de lecture du code
14     affichage(); // appel de la fonction d' affichage du resultat
15   }
16 }
17
18 // fonction de lecture du code
19 void lecture() {
20   for (int t = 0; t < nbMax; t++) { //boucle pour le nombre de lectures
21     while (PINB & B00000001) { // tant que etat pulse
22       tableau[t][0]++; // on incremente la valeur du tableau
23       delayMicroseconds(temporisation);
24     }
25     while (!(PINB & B00000001)) { // puis tant que etat non pulse
26       tableau[t][1]++; // on incremente l'autre valeur du tableau
27       delayMicroseconds(temporisation);
28     }
29   }
30   delay(200);
31 }
32
33 // fonction d' affichage du resultat
34 void affichage() {
35   Serial.println("*");
36   for (int i = 0; i < 2; i++) { // boucle pour etat Haut, puis etat Bas
37     for (int t = 0; t < nbMax; t++) { //lecture des valeurs
38       Serial.print(tableau[t][i]*temporisation); //affichage en
             microsecondes
39       Serial.print ("\t"); //tabulation pour presentation
40       tableau[t][i]=0; //effacement de la valeur pour prochaine lecture
41     }
42     Serial.println(); //saut de ligne
43     } }
```

Une fois que l'on a le codage, il suffit d'enregistrer les données que l'on reçoit, vérifier qu'elles correspondent à ce que l'on est censé trouver (en testant avec les codes enregistrés dans un tableau par exemple) puis on effectue l'action désirée selon le code trouvé.

Voici un exemple de code pour contrôler un servomoteur :

Lien arduino

Astuce

Le plus simple semble de coder sois-même l'émetteur pour s'affranchir de l'étape de décodage et cela permet de mieux gérer son programme. Par exemple, la première donnée pourrait déclencher une interruption puis toutes celles qui suivent pourraient être enregistrées, cela permettrait de ne pas avoir une carte arduino constamment en écoute et permettrait alors de l'utiliser pour autre chose.

De plus, cela permettrait d'envoyer juste le nombres d'informations nécessaires. si on a 6 commandes différentes possibles, on pourra envoyer 4 impulsions (la première active le système d'écoute, les trois autres constituent le code, on a donc 2^3=8 codes possibles.

On gagne alors en efficacité et en compréhension du système.