Chapitre 3

Projet 1 - Décodeur de message infrarouge

1. Présentation

La télécommande infrarouge (IR) fait partie de notre quotidien depuis le milieu des années 1970. Une multitude d'appareils sont pilotables à distance grâce à cette technologie : téléviseur, porte de garage, voiture, box internet, casque audio, etc.

Après la réalisation de ce premier projet, nous serons capables de vérifier le bon fonctionnement d'une télécommande IR, de déchiffrer les signaux qu'elle envoie, et de les enregistrer afin de les utiliser dans le second projet et de pouvoir ainsi piloter n'importe quel appareil IR à partir d'une Arduino.

1.1 Principe de fonctionnement

Après avoir terminé le montage, il sera possible de pointer une télécommande dans sa direction. L'appui sur les boutons de celle-ci déclenchera le décodage et l'affichage des trames d'informations IR dans le **moniteur série** de l'IDE Arduino.

1.2 Notions abordées

Ce premier projet permettra d'évoquer l'utilisation du **moniteur série** de l'Arduino, de faire appel aux librairies, de différencier les entrées analogiques et numériques de l'Arduino, et de s'initier aux protocoles d'échanges de données en électronique.

2. Matériel nécessaire

Le montage est construit autour d'un composant principal, le récepteur IR VS1838B.

Récepteur IR VS1838B

Le VS1838B est d'un rapport qualité/prix assez intéressant puisqu'il est possible de le trouver pour moins d'un euro. Il capte les ondes IR au travers de sa photodiode, le signal est amplifié pour être ensuite filtré, et finalement traité de manière logique afin d'en extraire des trames d'impulsions. Ces impulsions sont alors disponibles sur sa broche de signal S.

Pour réaliser ce montage, nous utilisons également une carte Arduino Uno afin de réaliser toute la partie « intelligente » de décodage et d'affichage des signaux.

Carte Arduino Uno

Remarque

Comme pour tous les montages présentés dans ce livre, toute autre carte Arduino « classique » (Mega, Nano, Due...) peut également faire l'affaire moyennant quelques adaptations simples du montage et/ou du programme. C'est une bonne manière pour le lecteur de s'approprier le projet.

3. Schéma et montage

Comme dans tout protocole d'échange d'informations, il y a un émetteur (ici, la télécommande IR), et un récepteur (l'appareil à piloter).

Nous décrirons brièvement le fonctionnement de l'émetteur pour nous attarder ensuite sur la raison d'être de ce premier projet : le récepteur IR.

3.1 La télécommande IR

Il existe plusieurs technologies de télécommande (infrarouge, radio, ultrason, etc.). Celle qui nous intéresse ici permet d'envoyer des signaux vers un récepteur IR grâce à un composant électronique bien connu et largement utilisé : la LED (*Light Emitting Diode*, ou DEL en français, diode électroluminescente). Les LED peuvent émettre dans toutes les couleurs mais la première à avoir vu le jour, en 1962, fut la LED IR qui émet donc dans la lumière non visible.

Remarque

Il est possible, malgré tout, de voir la LED d'une télécommande IR fonctionner en l'observant à travers un appareil photo numérique comme celui d'un smartphone.

Comment reconnaît-on une télécommande IR ? S'il est nécessaire de pointer la télécommande vers l'appareil à piloter pour que cela fonctionne, c'est qu'elle est infrarouge. Autre indice : nous pouvons distinguer une petite coque (la LED) sur la partie supérieure de la télécommande.

Télécommande IR avec sa LED d'émission

Il existe aussi des modèles avec un cache de protection en plastique noir mais transparent afin de laisser passer les infrarouges.

Autres télécommandes IR

L'émission de messages IR sera vue plus en détail dans le projet 2 - Télécommande infrarouge.

3.2 Le récepteur IR

Toute émission de signal IR par un émetteur est destinée à être reçue par un récepteur IR. Bien sûr, le protocole d'échange doit être connu des deux appareils pour qu'ils puissent se comprendre. C'est pourquoi il y a autant de télécommandes que d'appareils télécommandés, il serait effectivement fâcheux que la porte du garage réagisse à la télécommande du téléviseur.

En revanche, le récepteur que nous allons construire ici sera capable de comprendre toutes les télécommandes IR, rien de dangereux à cela puisqu'il ne pilotera finalement rien.

Brochage du récepteur IR

Le récepteur IR VS1838B dispose de trois pattes (ou pin), de gauche à droite :

- la sortie de signal S, sur laquelle les impulsions IR peuvent être récupérées ;
- la pin G (pour GND ou *Ground*, comprenez 0 volt) ;
- l'alimentation Vin, capable de supporter entre 3 et 5 V.

3.3 Montage

Le montage est relativement rapide puisqu'il consiste à relier un unique composant à l'Arduino Uno. Câblez le récepteur VS1838B de la manière suivante :

- La sortie de signal S, à relier à l'entrée numérique 7 de l'Arduino.
- La pin G, à relier au GND de l'Arduino.
- L'alimentation Vin, à relier à la sortie 5 V de l'Arduino.

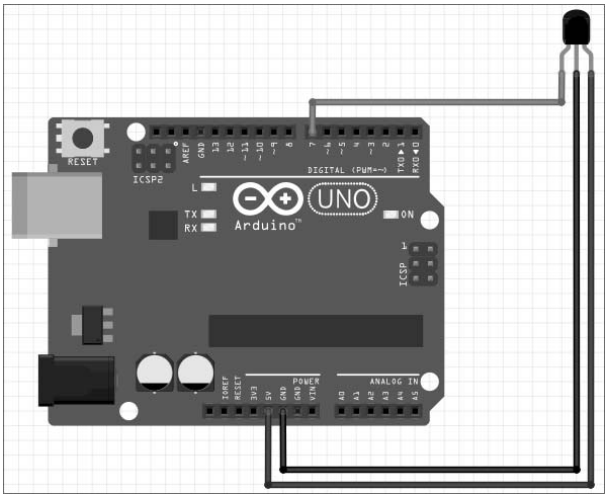

Schéma de montage du décodeur IR

Entrée analogique/numérique

Pourquoi relier la pin de signal du VS1838B à une entrée numérique de l'Arduino ? Il existe deux types d'entrées :

- Les entrées numériques ou digitales permettent de recevoir un signal de type HIGH ou LOW (1 ou 0 pour parler en binaire) selon que le voltage appliqué à la pin d'entrée soit respectivement de 5 V ou 0 V.

Remarque

En réalité, le seuil est plus bas, et tout voltage supérieur à environ 2,5 V est considéré comme étant un signal HIGH. Inversement, tout voltage inférieur à 2,5 V est considéré comme étant un signalement LOW. Cette explication est elle-même très simplifiée mais l'idée est là. À noter également que ce seuil peut varier légèrement selon les composants, ou les fabricants.

- Les entrées analogiques, qui ne se limitent pas à 0 ou 1 mais acceptent des valeurs proportionnelles à la tension appliquée. Elles peuvent aller de 0 à 255, 1023 voire plus, tout dépend de la capacité du convertisseur analogique/numérique (CAN, ou DAC (en anglais), *Digital Analog Converter*) implémenté dans le microcontrôleur. Si la tension appliquée à l'entrée est de 0 V, la valeur résultante sera 0. Si la tension est de 5 V, le résultat sera la valeur maximale autorisée par le DAC. Celui de l'Arduino est un DAC 10 bits, les valeurs des entrées analogiques peuvent donc varier entre 0 et 1023.

Dans le cas de ce montage, la sortie « signal » du VS1838B est soit à l'état HIGH, soit à l'état LOW selon les impulsions reçues par le capteur. Il n'y a donc pas de valeur intermédiaire possible. Dans ce cas d'usage, il est préférable de choisir une entrée numérique.

4. Programmation de l'Arduino

La structure d'un programme (ou *sketch*, nom fréquemment rencontré sur les sites web destinés à l'Arduino) est composée de plusieurs blocs :

- l'initialisation (ou *setup*) ;
- la boucle d'exécution (ou *loop*).

Mais avant cela, il est nécessaire de déclarer l'ensemble des librairies, des constantes et des variables globales utilisées dans nos deux fonctions `setup()` et `loop()`.

4.1 Les déclarations préliminaires

La majorité des programmes Arduino nécessitent l'utilisation d'une ou plusieurs librairies. Une librairie (ou bibliothèque) est un ensemble de déclarations et de fonctions destinées à un usage particulier. Le nombre de bibliothèques disponibles est impressionnant, et il existe pratiquement une bibliothèque pour chaque usage. L'étape de recherche de librairie fait donc partie de la démarche classique dans un projet Arduino. La multitude de sites web ou de chaînes YouTube traitant de l'Arduino est une source inépuisable d'informations, mais qu'il faut parfois traiter avec un peu de recul et un œil aguerri. Le plus simple reste sans doute d'ouvrir le gestionnaire de bibliothèque de l'IDE Arduino en cliquant sur **Outils - Gérer les bibliothèques** afin d'afficher la fenêtre suivante, et d'avoir un aperçu et une brève description de toutes les librairies disponibles :

Fenêtre de gestion des bibliothèques

N'hésitez pas à naviguer dans ce menu pour vous faire une idée de toutes les possibilités offertes.

Pour ce premier projet, nous incluons la librairie **IRremote** qui nous dispense de programmer toute la partie liée à la réception des trames d'impulsions sur la pin 7 de l'Arduino, leur décodage, la reconnaissance des protocoles, et bien d'autres choses :

```
#include <IRremote.h>

#define PIN_SIG 7
```

La seconde ligne définit la constante `PIN_SIG` à une valeur de 7, cette constante nous servira dans la suite du programme pour faire référence à la pin de l'Arduino sur laquelle est connectée la pin de signal du VS1838B.

Pourquoi utiliser une constante de type macro `#define` plutôt qu'une variable ou mieux une constante C++ ? Pour plusieurs raisons :

- Cette valeur ne va jamais changer au cours de l'exécution du programme, elle est constante.
- Pour optimiser le code : à la compilation du programme, l'IDE Arduino va remplacer toutes les références à `PIN_SIG` par sa valeur, ce qui se traduit par un gain de temps d'exécution (accès direct à la valeur, sans référencement), et de place (pas d'espace mémoire réservé pour la variable).
- Le langage utilisé sur Arduino est très souvent le C, parfois le C++. Il est donc plus prudent d'utiliser `#define` plutôt que `const`.

Sur des programmes volumineux, cela peut vraiment faire une différence surtout en termes d'espace mémoire. Prenez donc l'habitude d'utiliser les macros de préprocesseur (`#define`, `#ifdef`, etc.)

4.2 La fonction setup()

Le programme à télécharger dans l'Arduino doit effectuer plusieurs tâches :

- Démarrer le gestionnaire de communication série.
- Initialiser toutes les variables nécessaires.
- Configurer les pins de l'Arduino.

C'est la partie de « setup ». Tout programme Arduino dispose d'une fonction `setup()` permettant d'initialiser et de mettre en place tout ce qui sera utile par la suite.