Camille JOUHET
1ère générale

Lycée Paul REY
2019-2020

Club numérique lycée Paul Rey

Projet : recréer une télécommande infrarouge afin d'en comprendre le fonctionnement.

1- Comment fonctionne une télécommande en général ?

La télécommande est un objet que nous avons tous en au moins un exemplaire chez nous. Pour guider à distance nos télévisions, radios, chaînes hi-fi, rétroprojecteurs et autres, elles sont omniprésentes dans notre quotidien. Mais au fait, comment fonctionnent-elles ? L'utilisateur n'a qu' à appuyer sur la touche correspondante à l'action qu'il veut effectuer, et, cette action se produit sur la machine associée. C'est ce qui se passe du point de vue de l'utilisateur. C'est aussi simple que d'appuyer sur le bouton « démarrer » d'un ordinateur.

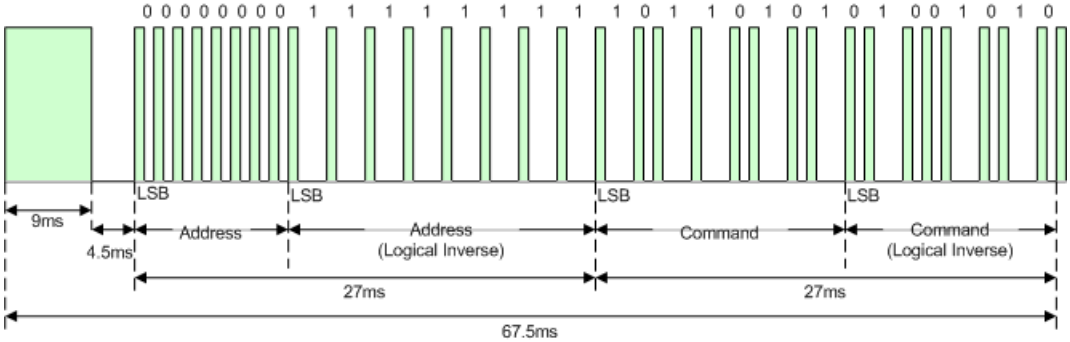
Si vous vous êtes déjà servi d'une télécommande vous avez sûrement observé tout cela… Mais vous êtes-vous déjà demandé ce qu'il se passe sous vos doigts ? C'est un peu plus complexe, mais le principe est compréhensible.

Dans un premier temps, l'appui sur la touche fait le contact entre deux fils pour laisser passer le courant. Ce passage d'électrons entraîne des calculs dans le microcontrôleur pour finalement transmettre des informations par infrarouge à l'objet visé (à la fréquence de 3 GHz pour une longueur d'onde de $\lambda = 10$ cm en général). Cette dernière information nous interroge sur comment transmettre des informations par des signaux électromagnétiques ?

Télécommande utilisée pour les vidéoprojecteurs du lycée

2- Les protocoles de communication → le protocole NEC

Pour transmettre des données par des signaux lumineux, l'être humain a trouvé plusieurs méthodes. Pour les télécommandes, nous utilisons des protocoles plus ou moins différents. Concentrons nous sur le protocole NEC car c'est celui-là qui a été choisi par les constructeurs des rétroprojecteurs du lycée. Ce protocole permet d'envoyer un nombre en base hexadécimale, codé sur 32 bits (= 8 chiffres hexadécimaux). Les 4 premiers chiffres sont dédiés à l'adresse de l'objet visé et les 4 derniers correspondent à la touche appuyée. Pour les physiciens :



Le nombre hexadécimal est envoyé en binaire, à l'aide de différentes durées d'émission :

- 0 logique :impulsion de 562,5 µs suivie d'un espace de 562,5 µs, avec un temps de transmission total de 1,125 ms.
- 1 logique : 562,5 µs suivie d'un espace de 1.6875 µs, avec un temps de transmission total de 2.25 ms).

La durée totale d'envoi est de 67,5ms, et cette durée est fixe. L'adresse et la commande sont doublées mais le 0 et le 1 sont inversés (voir schéma), ce qui garantit une durée d'envoi constante.

Voilà pour le protocole de transmission infrarouge NEC.

En savoir plus :
https://techdocs.altium.com/display/FPGA/NEC+Infrared+Transmission+Protocol

3- Récupération des codes des touches de la télécommande

Capteur infrarouge utilisé dans le projet

Pour récupérer les codes, on reproduit la chaîne d'acquisition du rétroprojecteur. Le capteur infrarouge détecte les variations de la luminosité des infrarouges. Ensuite, on tire d'une bibliothèque (collection de fonctionnalités pré-codées) les fonctions qui permettent de traduire les signaux transmis par le capteur en valeurs hexadécimales.

On mémorise les codes (on y reviendra plus tard).

4- Programme

Au total nous avons écrit deux programmes principaux. Les deux marchent grâce à la bibliothèque IRremote `#include <IRremote.h>` . (→ cette première ligne de code sert à « appeler » la bibliothèque). Le premier code (page suivante) sert à la récupération des codes pour les utiliser dans le second (page 5).

```
#include <IRremote.h>

const int cptInf = 2;
// const int del = 3; déclarée d'avance par la bibliothèque
IRrecv recepteurIR(cptInf);
decode_results messageRecu;
IRsend irsend;

void setup()
{
  Serial.begin(9600);
  recepteurIR.enableIRIn();
  recepteurIR.blink13(true);
}

void loop()
{
  if (recepteurIR.decode(&messageRecu))
  {
    if (messageRecu.decode_type == NEC)
    {
      Serial.print("NEC: ");
    }
    else if (messageRecu.decode_type == SONY)
    {
      Serial.print("SONY: ");
    }
    else if (messageRecu.decode_type == RC5)
    {
      Serial.print("RC5: ");
    }
    else if (messageRecu.decode_type == RC6)
    {
      Serial.print("RC6: ");
    }
    else if (messageRecu.decode_type == UNKNOWN)
    {
      Serial.print("UNKNOWN: ");
    }
    else
    {
      Serial.print(messageRecu.decode_type);
    }
    Serial.println(messageRecu.value, HEX);
    recepteurIR.resume();
  }
}
```

Les numéros sont les numéros des lignes de code.

3 : on enregistre dans une constantes de type nombre entier le numéro du pin (sorties ou entrées numériques) du capteur infrarouge (la diode infrarouge est déclarée sur le pin 3 par la bibliothèque, utilisée dans ce cas-ci pour effectuer des tests).

5, 6 et 7 : on déclare les variables à utiliser dans certaines des fonctions de la bibliothèque.

Ensuite, dans la boucle setup() (ligne 9 à 14), on initialise la communication avec le port serial, l'équivalent de la console en python (ligne 12), et on active le capteur infrarouge (ligne 12).

La ligne 13 sert juste à faire clignoter une diode intégrée à la carte (sortie 13) quand des données sont reçues par le capteur. Cela sert de témoin visuel.

Pour la boucle void loop (lignes 16 à 47), on fait afficher le nom du type de protocole reçu dans le port serial (boucles if/else if/else) puis on écrit la valeur reçue en hexadécimale (ligne 44).

Pour limiter les actions faites par la carte, on applique toutes ces étapes si la condition à la ligne 18 est vérifiée. Cette condition est vraie si le capteur capte effectivement un « message ».

```
1  #include <IRremote.h>
2  #include <SoftwareSerial.h>
3
4  SoftwareSerial HC06(10, 11); //RX et TX
5  IRsend irsend;
6
7  const char mode_power = 'A', mode_source = 'B', mode_pointer = 'C', mode_freeze =
   'D', mode_help = 'E';
8  const char mode_zoomPlus = 'F', mode_zoomMoins = 'G';
9  char mode = mode_power;
10
11 void setup() {
12   HC06.begin(9600);
13 }
14
15 void loop() {
16   if (HC06.available()) {
17     char mode_char = HC06.read();
18
19     switch (mode_char) {
20       case mode_power:
21         {
22           mode = mode_power;
23           irsend.sendNEC(/*code hexadécimal*/, 32);
24           break;
25         }
26       case mode_source:
27         {
28           mode = mode_source;
29           irsend.sendNEC(/*code hexadécimal*/, 32);
30           break;
31         }
```

Page précédente, le deuxième programme, celui qui « exécute » l'envoi d'un signal infrarouge lorsqu'une touche est appuyée. N'ayant pas de clavier, j'utilise un smartphone comme clavier, et je fais communiquer l'arduino et le smartphone via bluetooth.

On retrouve à la ligne 1 la bibliothèque précédente.

On ajoute une nouvelle bibliothèque pour communiquer avec un téléphone en bluetooth (voir montage 7).

On donne les pins du port virtuel serial à la ligne 4. HC06 est le nom du module bluetooth. Rx (pin 10) est celui qui reçoit (Recieved) des données. Tx (pin 11) est celui qui transmet (Transmetted) des données.

Ensuite, à la ligne 5 on donne la variable « irsend » à utiliser par la fonction « IRsend » de la première bibliothèque.

Pour les lignes 7-8-9, on enregistre les lettres envoyées par l'interface graphique du téléphone par bluetooth.

Dans le setup, on initialise la communication entre le téléphone et le module.

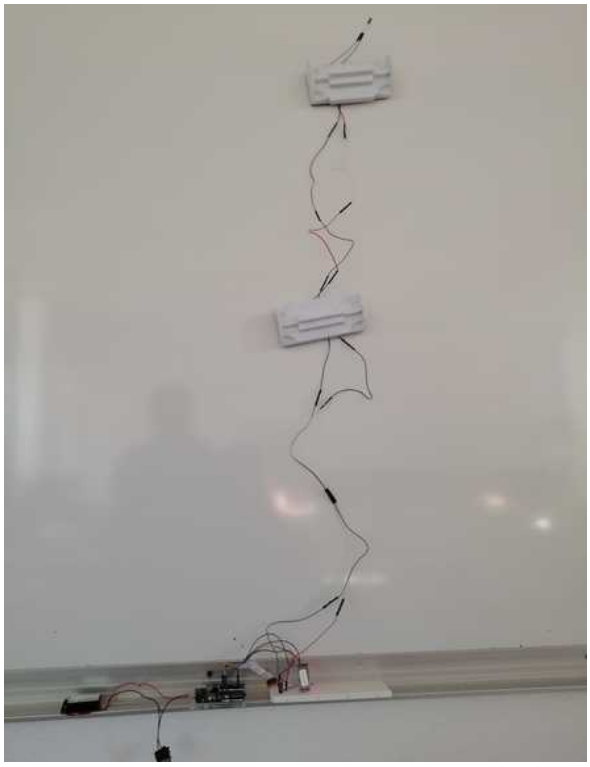
Pour le loop on limite le nombre d'actions comme précédemment avec la condition de la ligne 16. Ainsi, à la ligne 17, on récupère la lettre envoyée et on compare dans la boucle switch (ligne 19) (boucle imposée par la bibliothèque) chaque lettre à ce qu'elle est censée faire.

Dans les « case », on envoie, grâce aux fonctions irsend et sendNEC, la valeur voulue en hexadécimal (de la forme 0x (+code) où 0x désigne la base hexadécimale.) sur 32 bits par la diode infrarouge (voir protocole nec).

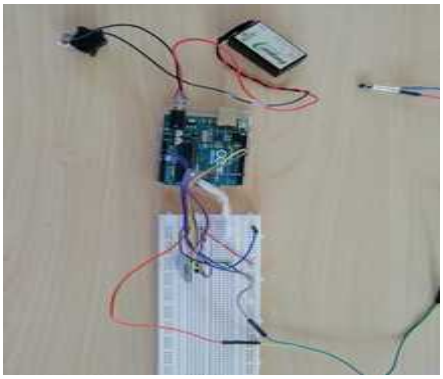
5a- Réalisation du montage électronique

Plusieurs problèmes se sont posés durant la réalisation du projet.

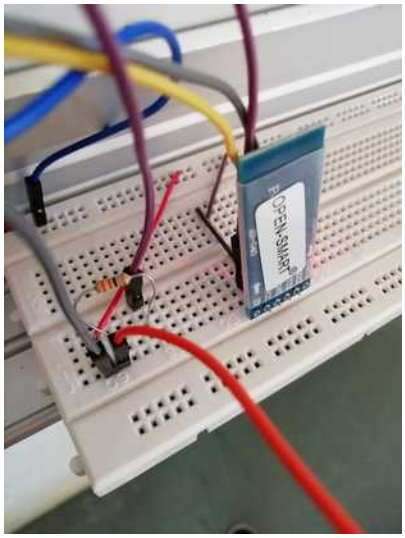
En premier lieu, la diode n'est pas assez lumineuse pour agir avec une distance de plus de 1 mètre. J'ai donc accroché comme j'ai pu la diode au tableau sous le vidéo projecteur (à courte focale), chose acceptable pour une expérience de prototype.



En deuxième lieu, je me suis demandé comment piloter tout ce système à distance. Le bluetooth, que j'avais déjà utilisé dans un autre projet, me paraissait être la meilleure solution pour faire communiquer l'arduino avec un smartphone qui me sert de clavier.

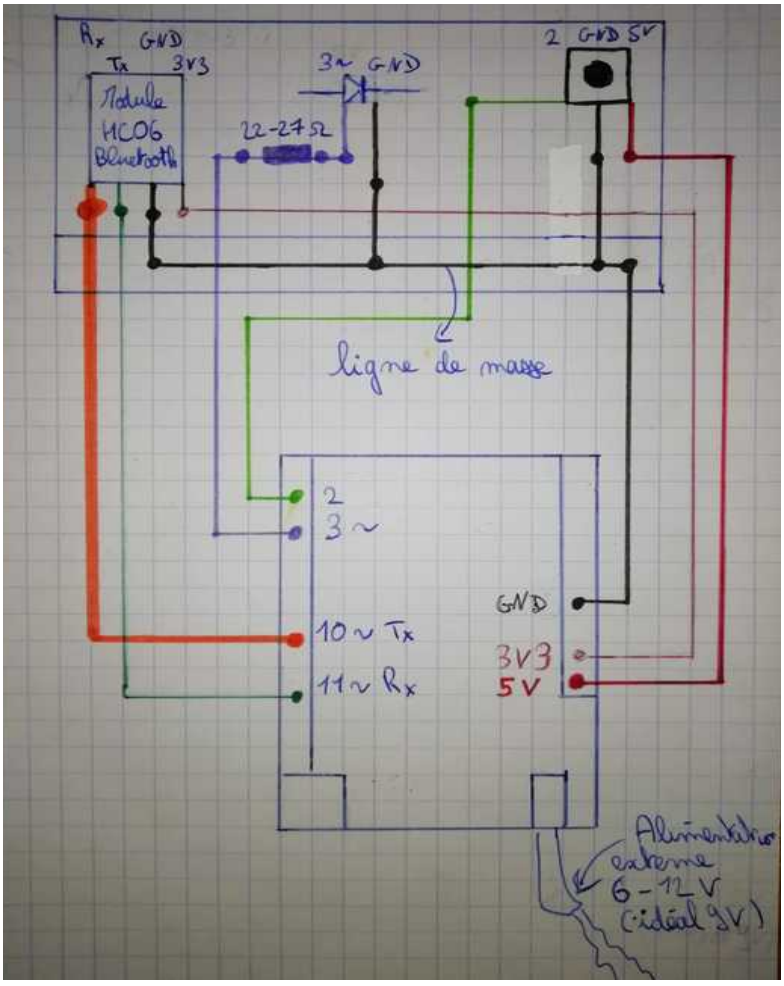


Module bluetooth sur support de prototypage, Breadboard.



5b-Schéma

Voici le schéma papier du montage :



Les pins avec ~ sont des PWM (sorties « analogiques » → *Pulse Width Modulation*). Pour le module Bluetooth, **NE PAS LE PLUGGER SUR LE 5 VOLTS**, en effet, il est prévu pour fonctionner sur du 3,3 volts, le 5 le grillerait. Ensuite, le Tx du module transmet des données sur la PWM 11 Rx qui reçoit les données et vice versa pour le pin 10 ~ . Le capteur infrarouge **ne doit pas recevoir le courant dans le mauvais sens,** la carte arduino s'éteindrait, le capteur surchaufferait et grillerait (oui, il y a du vécu …). Les composants sur le schéma sont dessinés de face. Personnellement, j'alimente l'arduino en externe avec une pile 9v pour permettre sa portabilité.

Références du matériel :

- Carte arduino → arduino uno

- Module dents bleues (de face) → HC06

- Diode infrarouge

- Capteur infrarouge (de face) → Tsop 4836

Merci à M. Bourumeau de m'avoir accompagné dans ce projet et aidé dans la rédaction de ce tutoriel