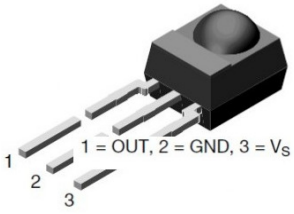
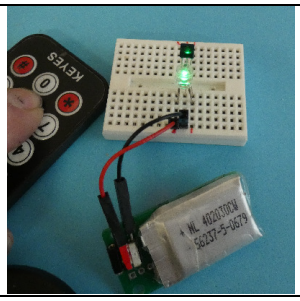
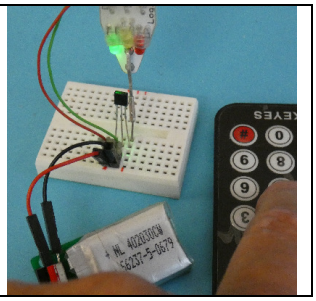
Télécommande IR simple

Une télécommande standard est facile à trouver, mais la séquence des impulsions qui permet de reconnaître chaque touche est compliquée, et toutes les télécommandes sont différentes. La solution simple est de ne s'intéresser qu'à l'enveloppe du signal. On presse une fois, rapidement ou lentement, deux fois, etc. C'est du Morse très simplifié, adapté à ce que l'on veut commander.

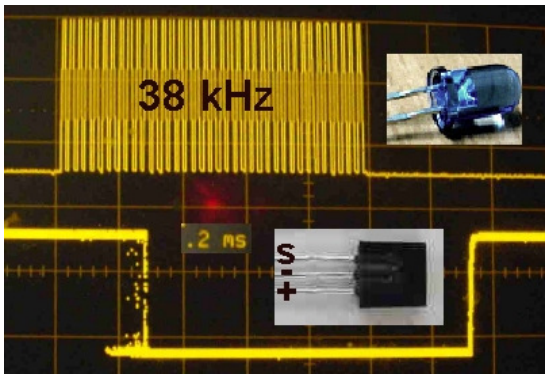
La plupart des télécommandes, quand on presse sur un bouton, envoient une séquence d'impulsions infrarouge qui dure 10 à 20 ms, et la séquence est répétée tant que l'on presse. Quelques télécommandes ne répètent pas, ce qui ne permet pas de détecter une impulsion longue.

Capteur IR (Infrared module IRM)

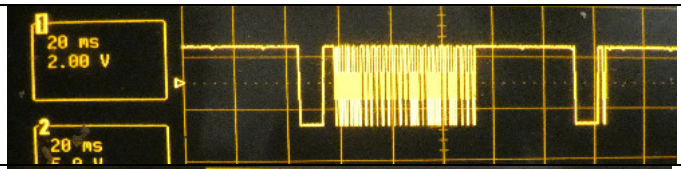
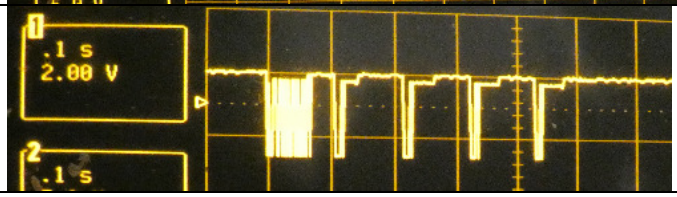
Tous les modèles de capteurs IR, de dimensions variables, certains avec un blindage, conviennent. Leur brochage semble toujours le même.

Comme premier test, vérifiez que vous avez un signal. Une Led peut être câblée entre Out et Vs, sans résistance, puisque c'est pulsé. Le crayon logique, c'est mieux.			

Ce test est indispensable. Il permet de voir qu'il y a des impulsions parasites, d'estimer la distance maximum de réception selon l'orientation du circuit et les réflexions dans la pièce.

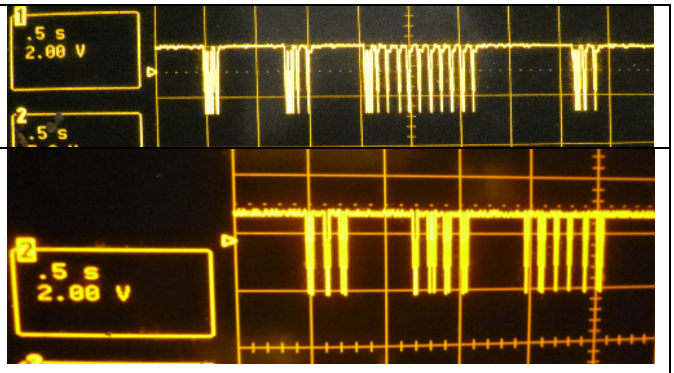
Le principe pour toutes les télécommandes et d'avoir une diode infrarouge (en général 940nm) qui envoie des trains d'impulsions à 38 kHz, d'une durée de 1-2ms. Le récepteur reçoit un signal d'intensité très variable. Un filtre de fréquence et ampli avec contrôleur automatique de gain rend l'enveloppe du signal. On peut travailler avec des impulsions de longueur variable, comme pour la télécommande EMIR du BIMO, et agir sur des moteurs ou servos. Voir didel.com/Emir.pdf	

Signal reçu

La télécommande de la photo envoie le code de la touche suivi par un code de répétition, toutes les 100 ms. Bon à savoir, mais on ne va pas en tenir compte.	
Ce qui nous intéresse c'est de savoir l'espace entre les répétitions. Ici, on a pressé 0.5s et on voir qu'il faut réarmer un délai de 0.1s à chaque lecture du signal. On va donc mesurer 0.6s	

Ici on a pressé 4 fois avec des durées de ~0.2, 0.3 1.2 et 0.3 secondes. Entre les pressions, il y a un délai de 0.5 à 1 seconde. Il faudra un délai de 1 à 2 secondes pour savoir que l'on a fini d'envoyer la séquence.

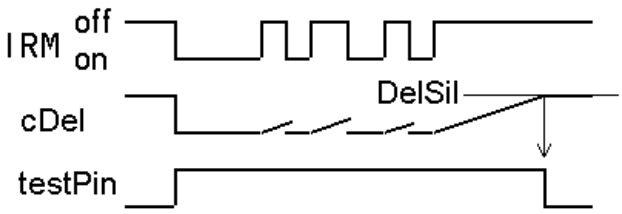
Ici, on a une autre télécommande qui répète tout le code. Il y a 3 pressions rapides.
Il faut 3 secondes pour transmettre cet ordre. Lent, mais simple.



Décodage

Pour mesurer les durées, on peut utiliser la fonction milli() d'Arduino. Quand le signal est actif, on lit le temps et on ajoute le délai de 0.2s ou 1s. Quand le signal est inactif, on lit le compteur de temps. Si le délai de 0.2s est dépassé, on ajoute 1 au nombre de pressions. Si le délai de 1s est dépassé, l'envoi de l'ordre est terminé. A peine plus compliqué s'il y a des impulsions courtes et longues !

Une solution plus efficace, en C portable et gérable par interruption si nécessaire, est d'échantillonner le signal toutes les 20ms et remettre un compteur à zéro si le signal est actif. On compte si le signal est inactif. Comme le montre la figure, si le compteur dépasse une valeur, on sait que l'envoi est terminé.



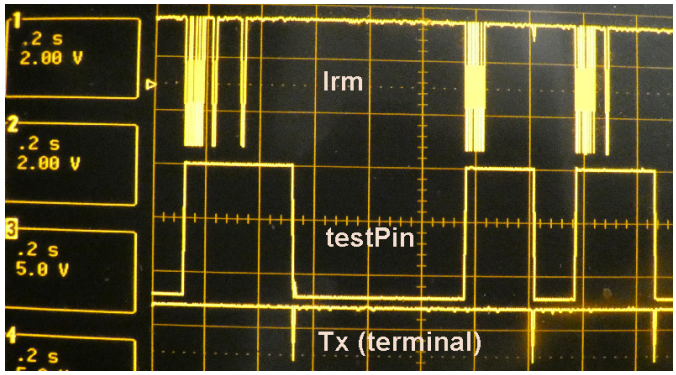

```
while (!IrOn) ;  // attend 1ere imp
// sensible aux impulsions parasites!
cDel=0;
while (cDel++<DelSil) {
   if (IrOn) { cDel=0;}
   delay (2);
}
```

Le programme de test doit déclarer IrOn (`#define IrOn !digitalRead(pinIRM`) en entrée et permettre de vérifier que le décodage est correct.
Avec un oscilloscope, c'est facile : on voit le signal et on peut activer un pin pour montrer l'effet. Voila notre programme de test qui active la pin Servo sur laquelle une sonde d'oscillo est branchée. Les paramètres ont été déterminés en remarquant que les impulsons sont assez espacées. En échantillonnant toutes les 20ms, on risque de ne pas remettre le compteur à zéro.

```
//TestTelecSimple.ino
#include "Dixi.h"
#include "Tell.h"
void setup() {
  SetupDixi();
  SetupTell();
  Serial.begin(9600);
}
// Irm.h
#define bIrm 4  // PORTB
#define IrOn !(PINB&(1<<bIrm))
#define IrOn !digitalRead(12)
#define DelSil 100

void Enveloppe () {  // on attend une imp,
puis un silence de durée
  while (!IrOn) ;  // attend 1ere imp
   // sensible aux impulsions parasites!
     ServoOn;
  int cDel=0;
  while (cDel++<DelSil) {
    if (IrOn) { cDel=0;}
    DelMs(2);
  }
    ServoOff;
}
```

```
byte cnt;
void loop() {
  Enveloppe();
 // if (IrOn) ServoOn; else ServoOff;
  Tell(cnt++);
  Serial.print(cnt);
}
```

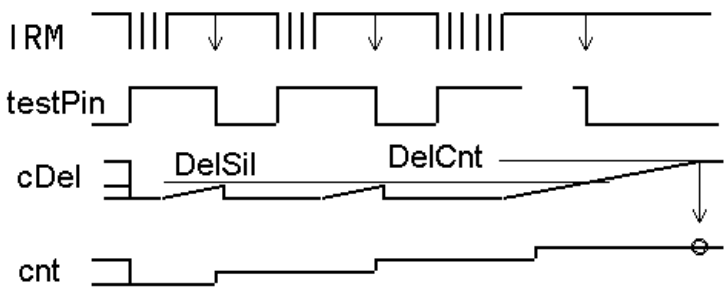


Sans oscillosope, il faut espionner avec le terminal. L'ordre `Serial.print (cnt);` prend 5ms. Dans le programme ci-dessus on compte les pressions. DelSil a été optimisé pour la télécommande utilisées, ok avec 2 autres. Si DelSil est trop cours, on va compter toutes les répétitions de la telecommande. Si DelSil est trop long, on ne va pas compter les pressions rapprochées.

Pour compter les pressions, il suffit d'introduire un délai plus grand.

```
//TestTelecComptePressions.ino
#include "Dixi.h"
#include "Tell.h"
void setup() {
  SetupDixi();
  SetupTell();
  Serial.begin(9600);
}
// Irm.h
#define bIrm 4  // PORTB
#define IrOn !(PINB&(1<<bIrm))
#define IrOn !digitalRead(12)

//Pour compter les pressions, il suffit
d'introduire
//un délai plus grand.
#define DelSil 100
#define DelCnt 500
byte Compte () {
  byte cnt;
  while (!IrOn) ;  // attend 1ere imp
  int cDel=0; cnt=0;
  while (cDel++<DelCnt) {
   if (IrOn) { cDel=0;}
   if (cDel==DelSil) { cnt++; }
   delay (2);
  }
  return cnt;
}
```




```
byte nPress;
void loop() {
  nPress= Compte();
 // if (IrOn) ServoOn; else ServoOff;
  Tell(nPress++);
  Serial.print(nPress);
}
```

Prorgrammes sous didel.com/TelecommandeIrSimple.zip

Programmes de démo du Dixi sous didel.com/Dixi.zip

Piloter le robot

On a une fonction qui fournit un nombre, un robot que l'on sait faire avancer et tourner.
Les primitives dans Dixi sont AvD, AvG, RecD, RecG, StopD, StopG.
Décidons que
1 stoppe 2 avance 3 tourne à droite, 4 tourne à gauche.
Une machine d'état code proprement ces actions.

```
//TestTelecDixi4Ordres.ino
#include "Dixi.h"
#include "Tell.h"
void setup() {
  SetupDixi();
  SetupTell();
}
// Créer librairie Iim.h en fin de développement
#define bIrm 4  // PORTB
#define IrOn !(PINB&(1<<bIrm))
#define IrOn !digitalRead(12)
#define DelSil 100
#define DelCnt 500
byte Compte () {
  byte cnt;
  while (!IrOn) ;  // attend 1ere imp
  int cDel=0; cnt=0;
  while (cDel++<DelCnt) {
   if (IrOn) { cDel=0;}
   if (cDel==DelSil) { cnt++; }
   DelMs (2);
  }
  return (cnt);
}
```

```
byte nPress=0;
void loop () {  //4 ordres
  nPress= Compte(); Tell (nPress);
  switch (nPress) {
     case 1:   // stop
       FreeG; FreeD;
     break;
     case 2:   // avance
      AvG; AvD;
     break;
     case 3:   // tourneG
      FreeG; AvD;
     break;
     case 4:   // tourneG
      AvG; FreeD;
     break;
     default: // plus de 4
      for (byte i=0; i<2*nPress; i++) {LedToggle;
DelMs(300);}
     break;
  }  // end switch
}
```

Faire mieux

C'est évidemment une première étape On ne peut pas avoir beaucoup d'ordre. Une première amélioration est d'avoir un stop rapide. Dans la fonction compte, il faut tester un délai plus long, et décider de couper les moteurs à ce niveau. Cela va permettre des déplacements plus précis.

Une télécommande n'est de toute façon pas pratique avec ses touches. Il faut un joystick, et la télécommande du Bimo, ou un joystick lié à une carte Arduino permet d'utiliser les routines Emir (www.didel.com/Emir.pdf).

Visualisation sur Oled

Tout le monde n'a pas la chance d'avoir un bon oscilloscope pour aider à mettre au point des applications personnelles. Mais vous avez certainement un Oled SSD1306 qui ne coûte quasi rien.
Un premier programme affiche le signal IRM échantillonné toutes les 100us, mais les impulsions sont prolongées sur 20ms pour que l'écran .

```
 void loop() {
     while(tx<128) {
         tlp=0;
         for (byte j=0; j<20; j++) {  //
boucle de 100ms
           tp=0;
           for (byte i=0; i<20; i++) {
// boucle 20ms
             if (IrOn) {tp=8;  tlp=8; }
             DelMs(1);
           }     // 10ms
           Dot(tx,10-tp);
   // on affiche 20ms*128 = 2.5s sur largeur écran
           DDot(tx,20-tlp);
           tx++;
         }
         if (tlp) {tct++;}
        }  // fin balayage
        tx=0; Clear();
}
```

Il faut définir IrOn et appeler la librairie Oled qui connaôt la fonction Dot(x,y).

Ces instructions sont extraites du programme Demo du Dixi

jdn 161224/181004/181214