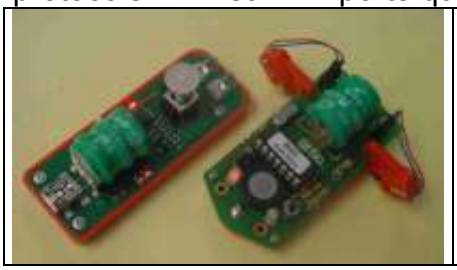
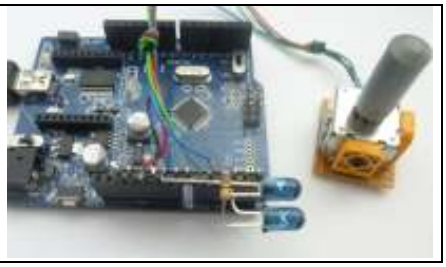
Commander le Rollover avec Arduino

.. ou le Bimo, la Tortue ou ce que vous inventerez

Le Rollover **www.didel.com/RolloverPub.pdf** a un récepteur infrarouge et un processeur local qui décode le protocole "Emir", particulièrement simple à transmettre, ce qu l'on motre ici, et à décoder pour piloter un robot, allumer des lampes, etc, ce qui est expliqué sous www.didel.com/Ir/Emir.pdf

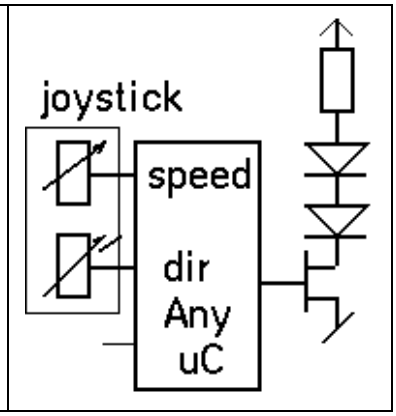
La télécommande du kit à souder Bimo www.didel.com/BimoPubE.pdf est compatible. Avec un joystick et une ou deux diodes infrarouge de 940 nm, on programme facilement le protocole Emir sur n'importe quel microcontrôleur.

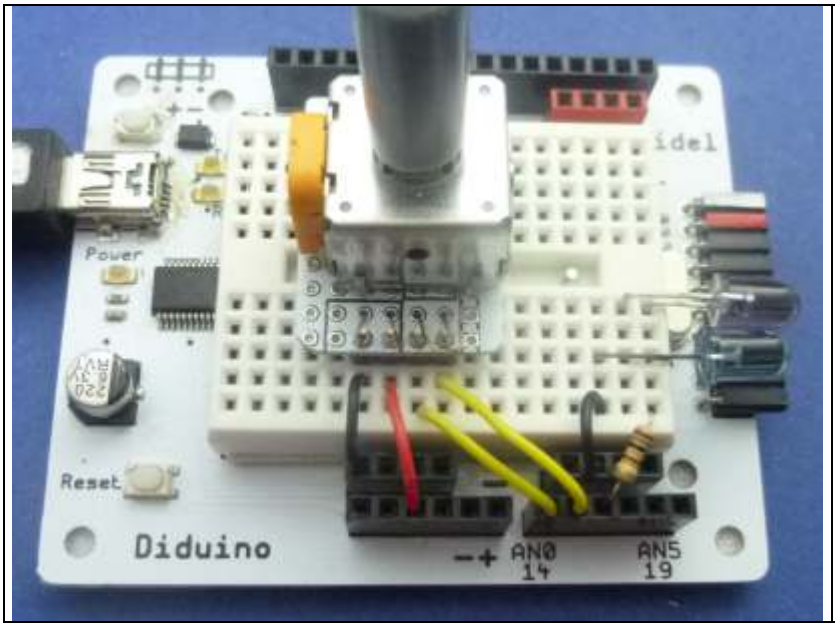




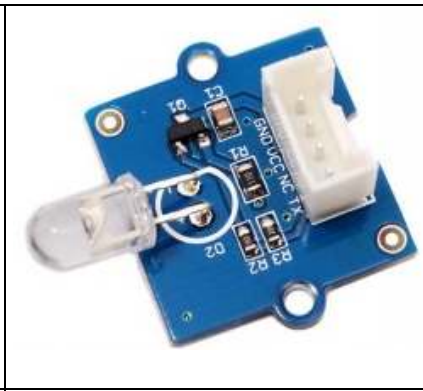
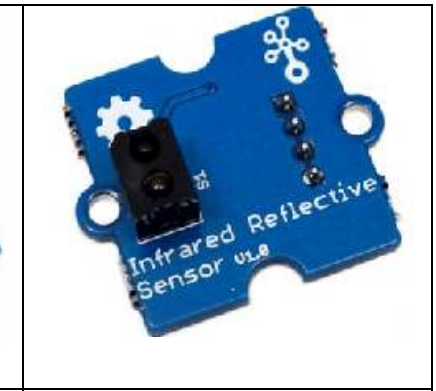
Partie électronique

La manette de commande utilise un joystick et une led en série avec une résistance de limitation de courant et un transistor 100mA, bipolaire (3904) ou Mos (2N7000, BS170) Si le microcontrôleur est compatible Arduino, les signaux analogiques des potentionètres sont connectés sur A0 et A1, par exemple. La led, ou mieux deux leds en série avec une résistance de 33 Ohm puisque la chute de tension d'une diode IR est de 1.3 à 1.5V, est pilotée par un transistor. La distance est alors de 5 à 10 mètres. Sans transistor avec une résistance de 220 Ohm, on atteint 2 mètres, ce qui est suffisant pour un petit robot.

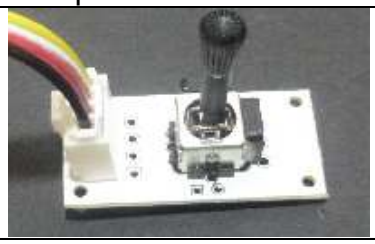
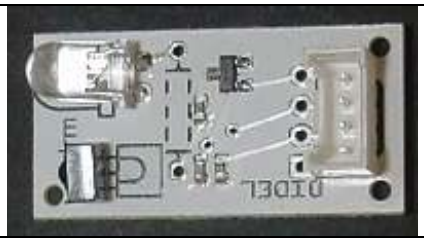




Si vous êtes un fan de grove, joystick et diode infrarouge existent. Vous pouvez même utiliser un capteur IR par réflexion, en ignorant le récepteur.

		
Joystick Grove COM90133P	IR Grove WLS12148P	IR reflective WLS07061P

Vous préférez les nouveaux DiGrove?

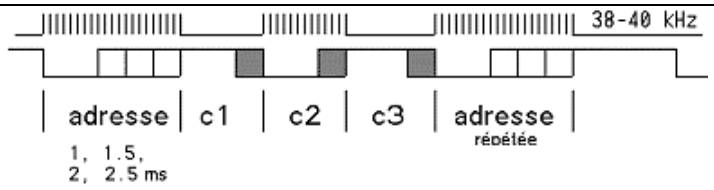
		
DgJoy	DgIrIrm	DgDistIr

Protocole de transmission Emir

Les télécommandes infrarouge utilisent en général le protocole RC5 qui a d'innombrables variantes et qui n'est pas si simple à programmer. Didel a défini le protocole Emir, utilisé depuis plusieurs années, en particulier pour le Bimo, avec l'avantage que plusieurs robots peuvent être pilotés simultanément, sans sélection de canal sur le robot. Le protocole est particulièrement simple à programmer avec Arduino, si on a déjà vu comment commander un servo et lire un capteur ultrason. Un document ancien trop détaillé est la base de ce qui suit www.didel.com/Ir/EmirSpecs.pdf

Il faut savoir pour commencer que la diode IR est modulée à 38 kHz. Le circuit récepteur IRM (InfraRed Module) est un circuit assez complexe qui crée l'enveloppe des impulsions reçues en filtrant les paraites. Un contrôle automatique de gain permet de couvrir une grande gamme d'intensité des signaux reçus.

Emission

Trois trains d'impulsions à 38-40 kHz forment la trame Emir. La durée entre impulsions compte, ce qui permet d'encoder 5 valeurs avec 3 impulsions. La trame est répétée toutes les 20 à 50 ms.	

Pour transmettre on envoie des trains d'impulsions à 38 kHz comme avec toutes les télécommandes. Ces trains d'impulsions sont filtrés par le récepteur infrarouge (IRM) qui génère un signal de même durée (à quelques % près).

Toutes les 20 à 100 ms, on envoie 3 impulsions de 1 à 2 ms, et l'astuce du protocole Emir est d'avoir une première et dernière impulsion qui est l'adresse du robot. Si l'adresse est la même, le message a de forte chance de ne pas avoir été perturbé par une autre transmission.

Entre ces deux impulsions, on a deux silences et une impulsion don't la durée de 1 à 2ms définit 3 canaux, que l'on peut facilement par programmation aiguiller vers des servos ou décoder avec les fonctions Arduino `pulseIn()` ou `micros()`.

Générer les signaux Emir est aussi simple que piloter un servo de télécommande.

On raisonne en nombre d'oscillations de 38kHz, 37 pour 1 milliseconde. La durée de l'adresse est de 1ms pour le premier robot. La durées des trains d'impulsions est de 1 à 2ms. Les impulsions (pulses) sont formés d'alternances à 38 kHz (période 26 us), calibrés au mieux à l'oscilloscope en tenant compte de la durée des autres instructions dans la

boucle. Les silences entre impulsions sont considérés comme faites d'alternances invisibles (nopulses).
On a dans l'ordre
Adresse 1ms 37 pulses. de 26 us
Silence 1-2 ms 37-75 nopulses of 26 us – canal vitesse 1.5ms manche au milieu - stop
Impulsion 1-2 ms 37-75 pulses of 26 us -- canal direction, 1.5ms ligne droite
Silence 1ms 37 pulses, 3e canal optionnel
Adresse 1ms 37 pulses of 26 us, répète l'adresse initiale
Les adresse utilisées sur le Bimo ont une durée de 37, 50, 63, 76 pulses.
On n'utilise que la durée 37 (1ms). A vous de rajouter un paramètre si vous voulet piloter 2-4 robots, grues, lampes d'intensité variable simultanément.

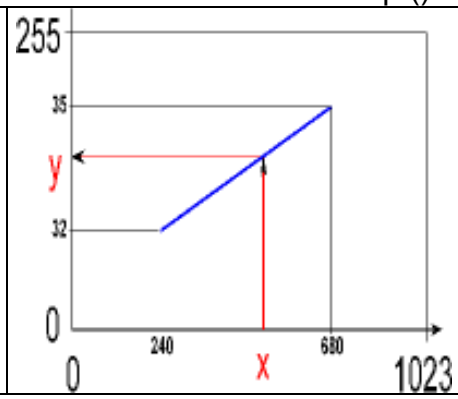
Procédure d'envoi avec un AVR 16MHz. (Arduino Uno, Diduino)
Fonctions pour la durée des inpulsions et silences:

```
void Pulse (byte pp)  {
    for (byte i ; i<pp ; i++) {
       IrOn;
       delayMicroseconds (10);
       IrOff;
       delayMicroseconds (10);
   }
}
```

```
void NoPulse (byte np)  {
   for (byte i ; i<np ; i++) {
      delayMicroseconds (28);
   }
}
```

La lecture des potentiomètres est convertie avec la fonction arduino map ().

La figure ci-contre rapelle ce que cette fonction fait.
Si les potentiomètres ne vont pas de 0 à 1000, on corrige dans le programme ci-dessous les valeurs 80,1000. Pour une impulsion de 1 à 2ms, on code les valeurs 37,78.




```
// EmirSoft
// Génère les signaux Emir à partir de 2 entrées ananlog..
#define IrOut 3
#define IrOn digitalWrite (IrOut,1)
#define IrOff digitalWrite (IrOut,0)

void setup() {
  pinMode (IrOut,1);
}
void Pulse (byte pp)  {
    for (byte i ; i<pp ; i++) {
       IrOn;  delayMicroseconds (10);
       IrOff; delayMicroseconds (10);
   }
}
void NoPulse (byte np)  {
   for (byte i ; i<np ; i++) {
      delayMicroseconds (28);
   }
}
```

```
int vit,tourne;

void loop() {
 avance = analogRead (A0);
 tourne = analogRead (A1);
 //   Serial.print(vit);
 //   Serial.print(" ");
 //   Serial.println(tourne);
Pulse (37);
NoPulse (map (avance, 80, 1000, 37, 78));
Pulse (map (tourne, 30, 1000, 37, 78));
NoPulse (37);
Pulse (37);
delay (20);  // between 10 and 100 ms
                    }
```

Le programme se trouve sous www.didel.com/Ir/RolloverCommandeArduino.zip

Avant de l'exécuter, il faut vérifier le câblage et les paramètres mis dans la fonction `map ()`

1) Lecture analogique
Connecter le joystick sur A0 A1 et vérifier les valeurs en extrémité et au centre.

```
//TestAna2.ino| Lit et affiche les 2 valeurs
void setup() {
  Serial.begin(9600);
  // la direction est en entrée par défaut, rien à initialiser
}
int var;
void loop() {
  Serial.print ("Av  ");
  Serial.print (analogRead (A0)) ;
  Serial.print ("Tou  ");
  Serial.println (analogRead (A1)) ;
  delay (1000);
}
```

Notez le débattement et remarquez l'erreur au centre

2) Envoi IR

Connectez les diodes IR et tester les routines données.
Une caméra permet de voir si la diode émet.
Observer a l'oscillo la tensions sur la diode ir et ce que recoit un capteur IRM
Le programme de test envoie une séquence fixe, sans lire le joystick

```
//TestIr.ino| 1ms à 38kHz
#define IrOut 3
#define IrOn digitalWrite (IrOut,1)
#define IrOff digitalWrite (IrOut,0)

void setup() {
  pinMode (IrOut,1);
}
void Pulse (byte pp)  {
    for (byte i ; i<pp ; i++) {
       IrOn;  delayMicroseconds (10);
       IrOff; delayMicroseconds (10);
   }
}
void NoPulse (byte np)  {
   for (byte i ; i<np ; i++) {
      delayMicroseconds (28);
   }
}
void loop () {
  Pulse (37);
  NoPulse (37);
  Pulse (15);
  NoPulse (37);
  Pulse (37);
  delay (20);  // between 10 and 100 ms
}
```

3) Ajuster la lecture du pot avec map ou autre

Ana = 0 → 1ms =37 Ana = 1000→ 2ms =74

```
//TestBimo.ino| Pilote un Bimo ou MiniRoll ou DDR2
#define IrOut 16  // PinArduino 16=A2
#define IrOn digitalWrite (IrOut,1)
#define IrOff digitalWrite (IrOut,0)

void setup() {
  pinMode (IrOut,1);
}
void Pulse (byte pp)  {
    for (byte i ; i<pp ; i++) {
       IrOn;  delayMicroseconds (10);
       IrOff; delayMicroseconds (10);
   }
}
void NoPulse (byte np)  {
   for (byte i ; i<np ; i++) {
      delayMicroseconds (28);
   }
}
// solution des avec des pots qui couvrent toutes les valeurs
byte avance, tourne;
```

```
void loop () {
  avance = 37 + (analogRead(A2)/37);
  tourne  = 37+ (analogRead(A3)/37);
  Pulse (37);
  NoPulse (avance);
  Pulse (tourne);
  NoPulse (37);
  Pulse (37);
  delay (20);  // between 10 and 100 ms
}
```

Décoder les signaux Emir pour piloter un robot, des Leds, etc

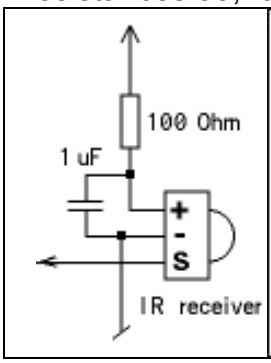
Si la vitesse des moteurs est commandée par interruption, la lecture du signal IR peut être bloquante. On attend une trame Emir avec un while, on mesure la première impulsion avec pulsein() et les suivantes avec la même fonction pulsein () et demandant d'attendre la fin du 1, puis la fin du zéro, etc. Les valeurs sont mises de côté et analysée à la fin de la trame. Les adresses de début et de fin doivent être proches, autrement on annule la trame (et on peu compter les erreurs et signalier qu'il y en a trop).
Le durées avance et tourne sont vérifiées et utilisées pour assigner la nouvelle valeur de vitesse des moteurs. Pour un robot à 2 roues motrices, la vitesse de chaque roue dépend de l'avance et de la rotation voulue. Les formules sont (à un facteur proportionnel près)

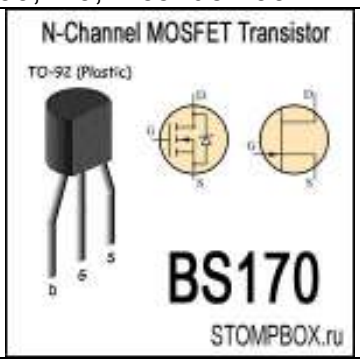
```
  vitMotG = avance + tourne;
  vitMotD = avance - tourne;
```

On peut avoir avantage à passer par une table, pour avoir des mouvement très lents comme avec le Bimo.

Si vous avez trop de peine à écrire ce programme, vous pouvez envoyer votre esquisse à info@didel.com.

Contenu du kit Emir
1 joystick 20mm
2 Leds IR
1 IRM
1 transistor BS170
Résistances 33,100,220, 1 condo 100nF





Attention, le + est sur la pin 1, ce qui n'est pas habituel.
Le filtre sur l'alimentation du IRM n'est pas nécessaire.
Une pull-up sur la sortie n'est pas nécessaire.
Ce capteur convient pour toutes les télécommandes.

Commande en tout ou rien
Une télécommande TV répète un signal tant que une touche est pressée. On peut donc très facilement voir que le IRM reçoit un signal, déterminer l'enveloppe de ce signal et décider d'actions qui dépendent de la durée de la pression ou d'un codage genre Morse.

jdn 150515