Télécommande flash radio 433Mhz RC/RF 08A

(Allo papa tango charlie .....)

Table des matières

Coté studio 1

Conclusion 1

Emetteur RF-08 2

Récepteur RC-08 2

Coté cour 4

Emetteur 4

Récepteur 5

Caractéristiques Sortie 5

Impédance de charge 5

Temps de réaction 6

Interfaçage Arduino 6

Codage utilisé 6

Tableau d'exemples de trames suivant codage 7

Listing exemple pour test 7

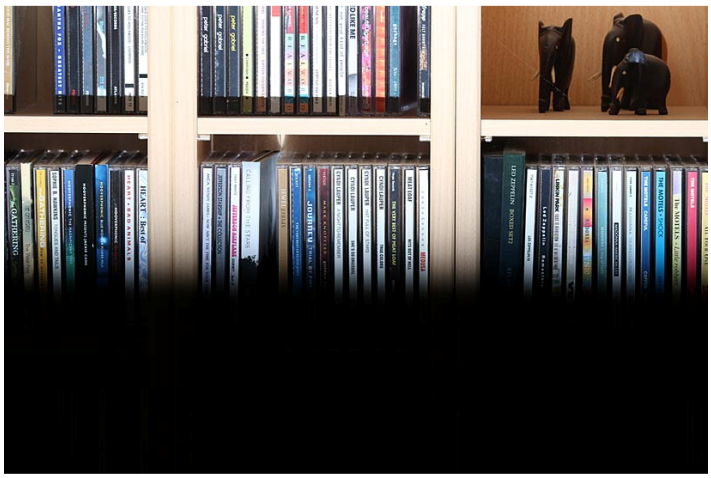
Révisions document 10

Coté studio

Cet ensemble à bas cout se trouve sous multiples formes, modèles et marques. Outre les différences d'ordre esthétique les modèles se différentient par leurs possibilités de codage avec un nombre d'identifiants allant de 4 à 16. Le modèle étudié autorise 8 codes différents grâce à 3 interrupteurs DIL mais le circuit est prévu pour en utiliser quatre soit 16 combinaisons. Il est probable que ce principe soit appliqué sur les modèles similaires.

La fréquence de fonctionnement de 433 Mhz utilise la plage réservée aux petits systèmes de commande à distance (Alarmes, déverrouillage portes voitures, sonnettes sans fil …). Le mode de transmission en modulation d'amplitude tout ou rien rend le système assez sensible aux perturbations externes. Une portée en milieu dégagé non perturbé de 75m est possible mais en habitation ou en studio il est préférable de compter une dizaine de mètres maximum pour garder un temps de latence correct.

Un temps de latence élevé provoquera l'effet suivant pour des temps de pose trop proches de la vitesse de synchro du boitier, ici photo au 1/200^{e} pour une vitesse sy/x du boitier de 1/250e :



Conclusion

Avantages :

- Cout très réduit.
- Compatibilité large entre modèles.
- Facilité d'interfaçage avec des systèmes type Arduino

Inconvénients :

- Fiabilité déclenchement et portée réduite.
- Temps de latence variable.
- Conception et faible résistance mécanique.
- Système de piles.

Au vu de la portée réduite et de la fiabilité de ce type de matériel des systèmes 2.4Ghz comme les YongNuo RF-603 sont nettement préférables pour un cout guère plus élevé, bref a eviter.

Emetteur RF-08

L'émetteur peut être actionné par un bouton de test sur sa face supérieure, le contact synchro flash sur un sabot standard doté d'une molette de verrouillage ou par l'intermédiaire d'une prise jack 2.5mm mono. L'envoi d'une commande est visualisé par une Led rouge située a proximité du bouton test. Pour pallier aux erreurs de transmission chaque commande emet une salve d'environ 200 codes par action sur le déclencheur de l'émetteur.

Les trois interrupteurs de codage (bloc rouge) permettent d'apparier l'émetteur et le récepteur, la réalisation de huit groupes de commande étant alors possible.

L'ensemble est alimenté par une pile 12v de type 23A, son remplacement nécessite l'ouverture du boiter par la dépose d'une petite vis Parker.

Récepteur RC-08

Lors de la réception d'un code valide le récepteur envoie sur sa sortie une impulsion de durée indépendante de celle de commande et pouvant varier avec l'impédance du matériel commandé. Le retard de réaction varie suivant la qualité de transmission et ne peut être inferieur à 0.7ms, chaque échec de transmission à un cout d'environ 1ms. Une Led à coté de l'interrupteur marche arrêt permet de visualiser la réception d'une commande.

Les sorties sont constitués d'une griffe standard porte flash dotée du seul contact synchro-flash actif et d'une prise au format mini USB (Un adaptateur mini USB - jack 3.5mm est fourni). Le modèle représenté ici a été modifié par l'ajout d'une prise jack femelle 3.5 mm pour permettre l'utilisation d'un câble standard Jack - N3 de commande déclenchement de boitier Canon.

L'élément de sortie étant un triac 600v 0.5A les anciens flashs de studio délivrant une haute tension ou négative peuvent être commandés. Même si ce circuit n'est pas adapté il permet aussi la commande d'un boitier, le plus gros problème résidant dans les possibilités de câblage.

Le sabot verrouillable n'est pas actif et permet juste le montage sur une griffe standard, il est aussi muni d'un taraudage au pas ¼'.

Le récepteur assez gourmand en énergie est muni d'un interrupteur marche arrêt, une trappe permet le remplacement des deux piles AAA 1v5.

Coté cour

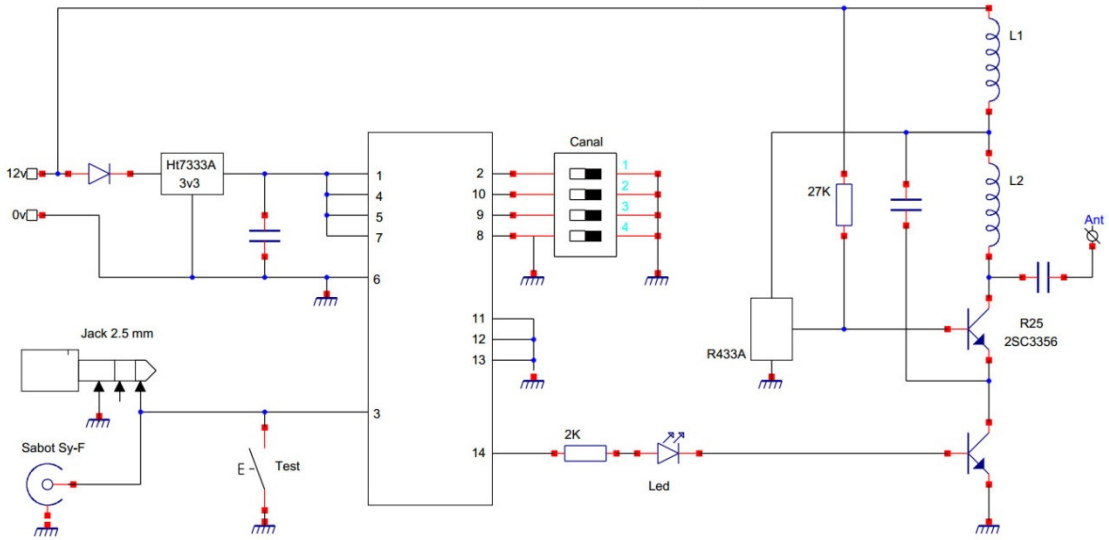
Emetteur

La partie codage est constituée d'un petit circuit intégré spécialisé en boitier SOIC-14 dont le marquage a été effacé. Un régulateur à faible perte protégé contre les inversions de polarité l'alimente en 3v3. La consommation au repos est négligeable, une pile ayant une durée de vie supérieure a un an sans utilisation.

La sortie 14 de ce circuit spécialisé met sous tension au rythme des niveaux hauts de la séquence de codage l'ensemble émetteur. Cette partie RF est calée en fréquence par un résonateur R433A en boitier métallique et utilise une structure classique pour ce composant.

La conception du circuit imprimé est prévue pour l'utilisation d'un bloc de quatre d'interrupteurs DIL, un bloc de trois étant utilisé le bit 4 est mis à la masse.

L'envoi d'une impulsion de commande consiste en l'émission d'une répétition de trames de codes successives pendant environ 230ms.

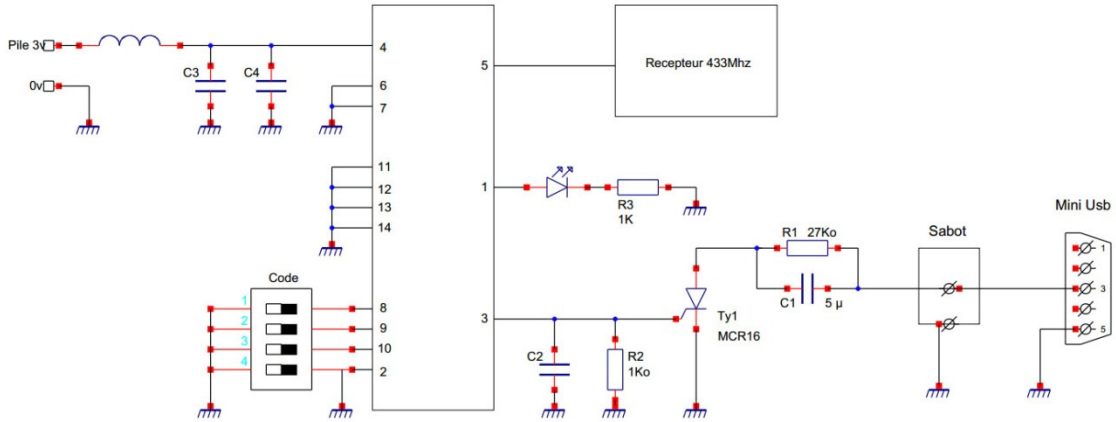


Récepteur

Le récepteur alimenté en 3v est constitué de deux platines, l'une comprenant l'ensemble radio 433 Mhz, l'autre la partie codage et interface de sortie.

Tout comme pour l'émetteur le circuit de décodage au format SOIC-14 a son marquage effacé. Le circuit imprimé est lui aussi prévu pour un codage par quatre interrupteurs, le bit 4 étant mis au 0v par un pont.

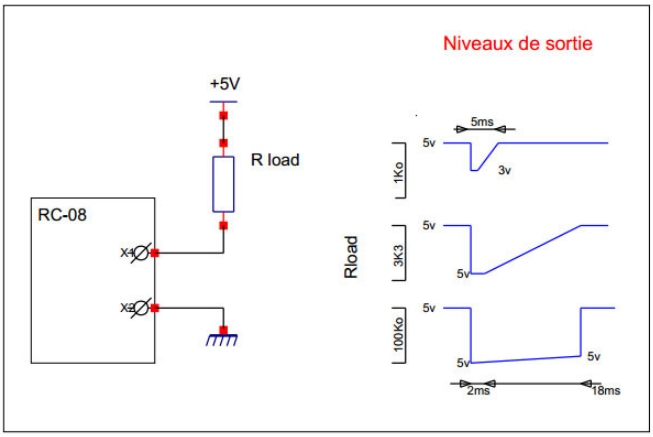
L'étage de commande flash est constitué par une cellule RC limitant la durée de l'impulsion et permettant le désamorçage du thyristor. Ce dernier a des capacités de commutation de 0.8A sous 600v.



Caractéristiques Sortie

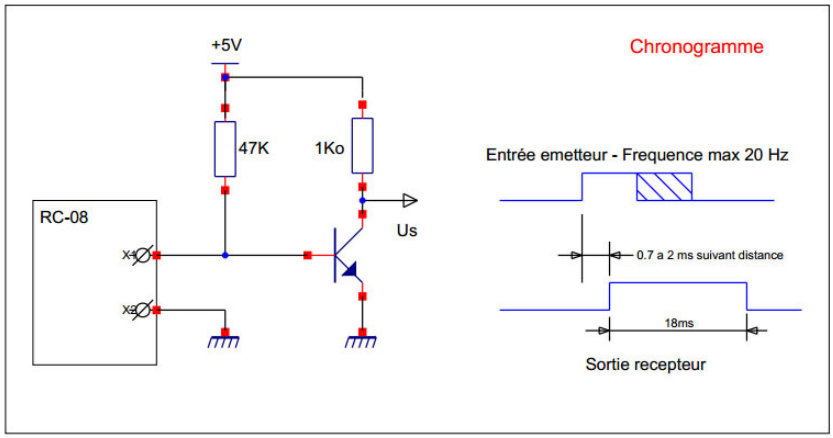
Impédance de charge

Du fait de la présence d'une cellule RC en sortie la durée de l'impulsion de sortie est dépendante de l'impédance du circuit de charge.



Temps de réaction

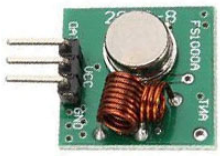
Structurellement au vu du système de codage utilisé le temps de latence est au moins égal au temps de réception du code le plus court soit environ 600µs pour le code 0 et 850µs pour le plus long. En outre une trame complète durant environ 1ms il faudra ajouter cette valeur a chaque erreur ou trame mal interprétée.



Interfaçage Arduino

Contrairement aux transmetteurs FSK 2.4 Ghz il est facile de reproduire les codes utilisés par ce type de transmetteurs. Des petits modules de qualités plus ou moins diverses permettant d'émettre une porteuse 4500 Khz existent pour un cout dérisoire. Il suffit de relier une sortie digitale de l'Arduino a l'entrée de commande modulation de l'émetteur radio.

Ce modèle de référence FS1000 trouvé couramment sur le net a posé quelques soucis de qualité de transmissions avec des erreurs de discrimination du bon code dues a une diminution des longueurs d'impulsion du signal codé a la réception.



Codage utilisé

Un décryptage des signaux de sortie du circuit de codage avec un analyseur logique a permis de déterminer les règles suivantes :

- Le circuit est configuré pour utiliser 4 bits de données, un bit sera considéré à l'état haut par mise au 0v de son entrée par fermeture de l'interrupteur lui étant associé.

- Une trame consiste en l'envoi des quatre bits de codage transmis en ordre inverse soit du poids fort au poids faible.

- Les trois bits de poids fort sont systématiquement émis. Un bit à l'état 1 consiste en l'envoi d'un signal long (Activation de la sortie pendant 160 a 170 µs suivi d'une séparation de 60 µs), un bit à l'état 0 consiste en l'envoi d'un signal court (Activation de la sortie pendant 60 µs suivi d'un séparateur de 110 µs).

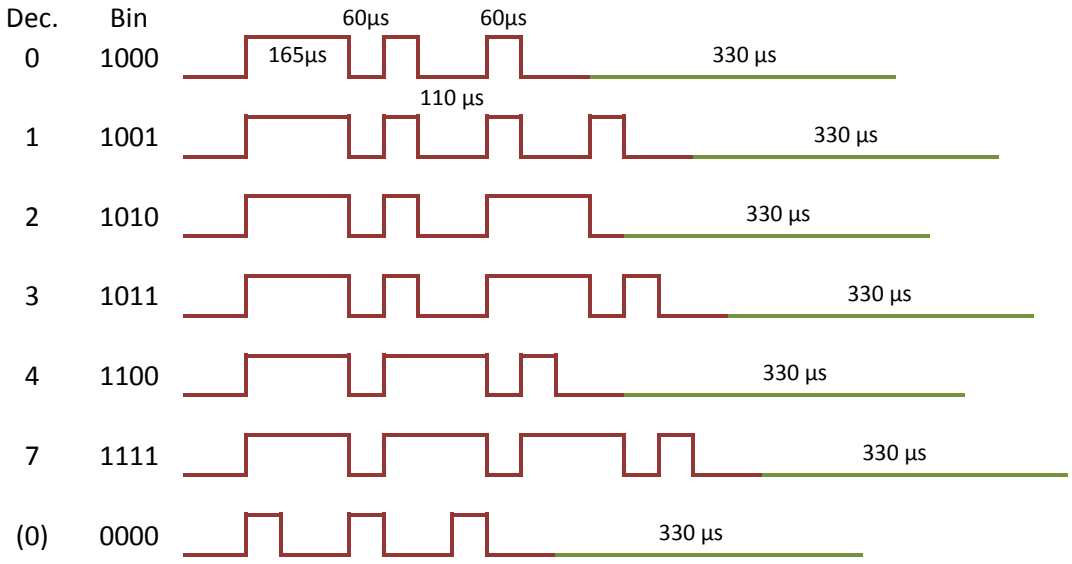
- Le bit 0 est soit ignoré à l'état 0, soit à l'état 1 constitué de l'envoi d'un signal court.

- Chaque trame est terminée par une pause d'environ 330 µs.

La qualité de transmission et de l'émetteur va jouer sur la conformité du signal de réception par rapport à ce qui est émis, en particulier sur la largeur des impulsions du signal démodulé, c'est pourquoi le codage a été analysé directement sur le signal source avant modulation RF et non pas sur un signal reçu.

Tableau d'exemples de trames suivant codage

La dernière ligne correspond à un modèle avec 4 interrupteurs montés correspondant a un vrai zéro décimal.



Listing exemple pour test

Dans cet exemple un Arduino Mega a été utilisé avec un shield LCD standard, la sortie de modulation est sur le port 53, une Led branchée sur le port 52 s'allume pendant la transmission.

Deux codes différents peuvent être émis simultanément, leur valeur étant modifiable par les touches de direction, la touche Sel provoque l'envoi d'une série de codes.

```
// ============================================================ Declarations

 #include <LiquidCrystal.h>
 LiquidCrystal Lcd216(8, 9, 4, 5, 6, 7);
 const int KeyInput = 0;
 int AnalogKey = 0;
 int ValKey = 0;
 int MemoKey =0;
 const int KeyHaut =0x5;
 const int KeyBas =0xA;
 const int KeyDroite =0x1;
 const int KeyGauche =0xF;
 const int KeySel =0x17;
 const int KeyNone =0x20;

 byte CanalA = 0;
 byte CanalB = 1;

 byte NbTrans = 255;
 byte CptTrans =0;

 const byte DataRF = 53;
 const byte LedE = 52;

 const byte DelayHigh = 160;
 const byte DelayLow  = 58;
 const byte DelayGapH = 55;
 const byte DelayGapL = 110;

 const char* MsgCanalA = "Canal 1 = ";
 const char* MsgCanalB = "Canal 2 = ";

// ============================================================ Intialisation
void setup() {

 Lcd216.begin(16, 2);
 Lcd216.clear();
 spAffCanalA();
 spAffCanalB();

 pinMode(DataRF, OUTPUT);
 digitalWrite(DataRF, LOW);

 pinMode(LedE, OUTPUT);
 digitalWrite(LedE, LOW);
}

// ============================================================ Boucle Principale
void loop() {

 AnalogKey = analogRead(KeyInput);        // ++++++++++++++++++++++++Gestion des Touches
 ValKey = (AnalogKey / 32) + 1;

 if (MemoKey != ValKey){
  MemoKey = ValKey;

 if (ValKey == KeyDroite) {
  if (CanalA & B1000)
   {CanalA =0; }
  else
   { CanalA ++; }
  spAffCanalA();  }

 if (ValKey == KeyGauche) {
  if (!CanalA)
   {CanalA = B1000; }
  else
   {CanalA --; }
  spAffCanalA();   }

 if (ValKey == KeyHaut) {
  if (CanalB & B1000)
   {CanalB =0; }
  else
   {CanalB ++; }
  spAffCanalB();   }

 if (ValKey == KeyBas) {
  if (!CanalB)
   {CanalB = B1000; }
  else
   {CanalB --; }
  spAffCanalB();  }

 if (ValKey == KeySel) spTransmit();


 delay(50);
 }
```

```
}

//========================================================================== Divers
SP
void spAffCanalA () {

 Lcd216.setCursor(1,0);
 Lcd216.print(MsgCanalA);
 if (CanalA & B1000) {
  Lcd216.print("Off");
  }
 else {
  Lcd216.print (CanalA);
  Lcd216.print ("  ");
  }
}


void spAffCanalB () {

 Lcd216.setCursor(1,1);
 Lcd216.print(MsgCanalB);
 if (CanalB & B1000) {
  Lcd216.print("Off");
  }
 else {
  Lcd216.print (CanalB);
  Lcd216.print ("  ");
  }
}


// ========================================================================== Emission
void spTransmit () {

 if (!(CanalA & CanalB & B1000)) {

  digitalWrite(LedE, HIGH);
  CptTrans = NbTrans;

  while (CptTrans--) {
   if (!(CanalA & B1000)) { EnvoiTrame(CanalA & B111); }
   if (!(CanalB & B1000)) { EnvoiTrame(CanalB & B111); }
  }
  digitalWrite(LedE, LOW);
 }


}


void EnvoiTrame (byte Data) {        // ++++++++++++++++++++++++++++++++++++++++   1 Envoi

  digitalWrite (DataRF, HIGH);       // -------- Bit4
  delayMicroseconds(DelayHigh);
  digitalWrite (DataRF, LOW);
  delayMicroseconds(DelayGapH);

  digitalWrite (DataRF, HIGH);       // ---------Bit3
  if ( Data & B100) {
   delayMicroseconds(DelayHigh);
   digitalWrite (DataRF, LOW);
   delayMicroseconds (DelayGapH); }
  else {
   delayMicroseconds(DelayLow);
   digitalWrite (DataRF, LOW);
   delayMicroseconds (DelayGapL); }

  digitalWrite (DataRF, HIGH);       // ---------Bit2
  if ( Data & B10) {
   delayMicroseconds(DelayHigh);
   digitalWrite (DataRF, LOW);
   delayMicroseconds (DelayGapH); }
  else {
   delayMicroseconds(DelayLow);
   digitalWrite (DataRF, LOW);
   delayMicroseconds (DelayGapL); }

  if ( Data & B1) {                  // ---------Bit1
   digitalWrite (DataRF, HIGH);
   delayMicroseconds(DelayLow);
   digitalWrite (DataRF, LOW);
   delayMicroseconds (DelayGapH);
  }

  delayMicroseconds (320);

}
```

Révisions document

v1.00	21/03/2014	Première diffusion.
v1.01	11/04/2014	Quelques corrections mineures.
v1.02	29/02/2015	Correction coquille, ajout photo exemple latence excessive.
V1.03	04/04/2015	Coquille vitesse chronogramme en ms au lieu de µs