Contrôle de la présence de courrier ou de colis d'une boite au lettre distante de plus de 100 m

Ce projet permettra quand il sera terminé ... de voir un texte sur un afficheur LCD indiquant la présence d'un courrier ou d'un colis se trouvant dans une boite au lettre se trouvant au bout d'un chemin. Cette boite au lettre est distante de la maison d'environ 100 à 120 m. Pour cela je vais utiliser 2 Arduinos : 1 coté BAL et l'autre coté maison. Ces 2 Arduinos seront reliés par 1 émetteur 433mhz (Fréquence autorisée en France) coté BAL et 1 récepteur 433mhz coté maison.

Coté BAL :

- 1 BAL
- 1 arduino uno ou 1 attiny85(8k , 6 I/O , 6 int) + Circuit imprimé
- 1 batterie 3,7v
- 1 cellule solaire délivrant environ 3v pour charger la batterie.
- 1 émetteur 433mhz
- 1 contact à fermeture ou à ouverture, magnétique ou mécanique (je ne sais pas encore) pour le volet d'insertion du courrier
- 1 contact à fermeture pour l'ouverture de la porte (colis ou retrait du courrier)
- 1 capteur de pression pour vérifier si un courrier à été déposé ou non
- 1 boite étanche pour protéger le tout.

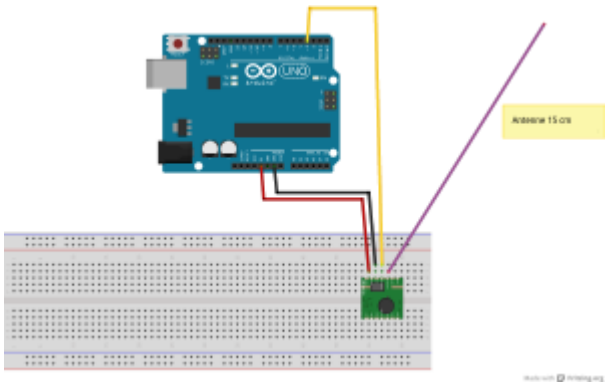
Coté Maison:

- 1 Arduino uno
- 1 afficheur LCD 16×2 I2C (pour afficher l'information de réception du courrier et plus tard, la température extérieure, l'heure et la date de dépose du courrier , la photo de la factrice (non je blague))
- 1 alimentation sur pile ou batterie rechargeable.
- 1 détecteur de présence pour afficher l'info que lorsque que quelqu'un se trouve à proximité.
- 1 récepteur 433mhz
- 1 boîtier non étanche (il se trouvera dans la maison)

Test communication 433Mhz

Pour l'instant je teste la liaison entre 2 Arduinos avec les modules 433mhz

Coté émetteur :

Schéma composants : 

Test_liaison_433mhz_Emetteur.ino

```
#include <VirtualWire.h> // Vous devez télécharger et installer la
librairie VirtualWire.h dans votre dossier "/libraries" !


void setup()
{
    Serial.begin(9600);
    vw_setup(2000);               // Bits par seconde (vous pouvez le
modifier mais cela modifiera la portée). Voir la documentation de la
librairie VirtualWire.
    vw_set_tx_pin(3);             // La broche 3 sera utilisée pour
transmettre la DATA, vous pouvez changez de broche si vous le désirez.
}

void loop()
{
   const char *msg="Tu as du courrier !";                          //
C'est le message à envoyer.
   vw_send((uint8_t *)msg, strlen(msg));
   vw_wait_tx();                                                 // On attend
que le message complet soit envoyé.
   delay(200);                                                   // Très
important sinon cela peut brouiller d'autres appareils !
}
```

Coté récepteur :

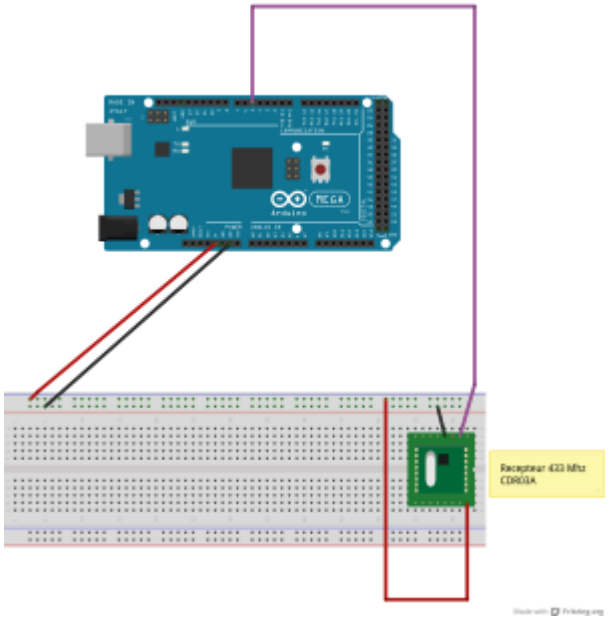


Schéma composants :

Test_liaison_433mhz_Recepteur.ino

```
#include <VirtualWire.h> // Vous devez télécharger et installer la
librairie VirtualWire.h dans votre dossier "/libraries" !

void setup()
```

```
{
    Serial.begin(9600);
    vw_setup(2000);               // Bits par seconde (vous pouvez le
modifier mais cela modifiera la portée). Voir la documentation de la
librairie VirtualWire.
    vw_set_rx_pin(5);             // C'est sur cette broche que l'on
reliera les broches DATA du récepteur, vous pouvez changez de broche si
vous le désirez.
    vw_rx_start();                // On démarre le récepteur.
}

void loop()
{
    uint8_t buf[VW_MAX_MESSAGE_LEN];
    uint8_t buflen = VW_MAX_MESSAGE_LEN;
    if (vw_get_message(buf, &buflen)) // On test afin de savoir si un
message est reçu.
    {// Un message est reçu.
      int i;

        for (i = 0; i < buflen; i++)
        {
          Serial.write(buf[i]);  // On affiche le message lettre par
lettre. Par exemple buf[4] sera égale à la 5ème lettre du mot envoyé
(Snootlab donc "t") car on compte le zéro ici.
        }
      Serial.println(""); // On saute une ligne afin d'avoir plus de
clarté.
     }
}
```

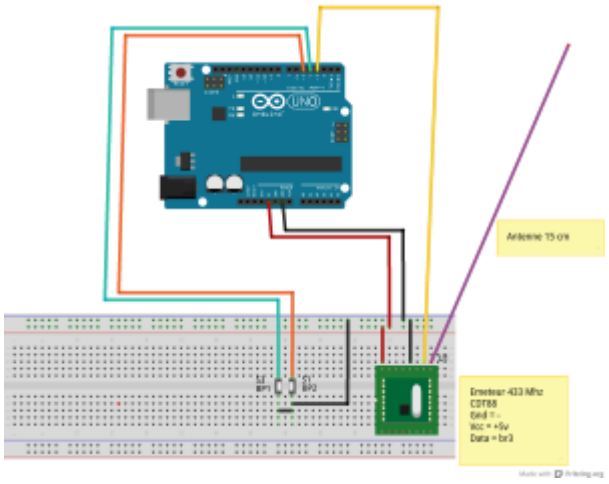
Cela fonctionne , il faudra que je teste la distance maximum entre les deux platines (A suivre ..)

J'ai essayé avec les modules 433mhz , en ligne droite sans obstacle , j'arrive à une distance de 95m , à 96 m cela ne passe plus. Il faut que je teste en câblant des antennes pour augmenter la distance . Sinon je vais essayer avec des modules XBEE pro qui dépasse les 100m.

Maintenant je teste coté BAL , les interrupteurs indiquant la présence de courrier (volet BAL ouvert) ou la présence d'un colis (porte ouverte) j'ajoute deux interrupteurs pour le test.

Coté Émetteur

Schéma composants :



Test_BP_433mhz_Emetteur.ino

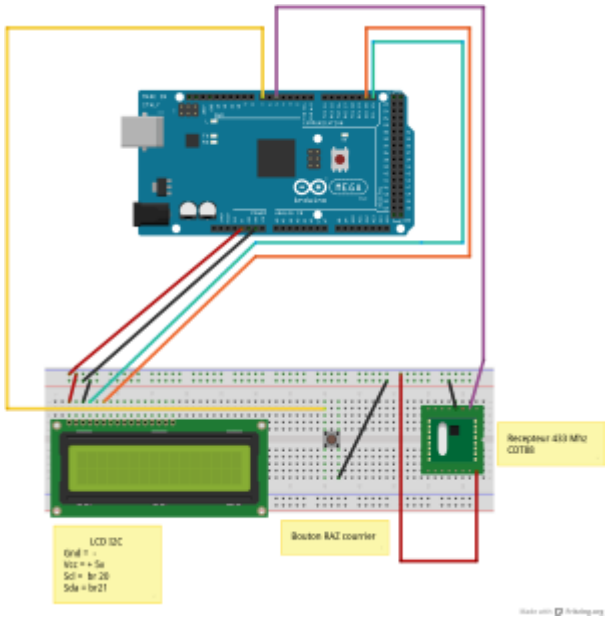
```
#include <VirtualWire.h> // Vous devez télécharger et installer la librairie VirtualWire.h dans votre dossier "/libraries" !

const int BP1=4; //Lettres
const int BP2=5; //Colis


// --- Déclaration des variables globales ---

int ETAT_BP1; // variable d'état du bouton poussoir 1
int ETAT_BP2;// variable d'état du bouton poussoir 2


void setup()
{
    pinMode(BP1, INPUT); //met la broche en entree
    pinMode(BP2, INPUT); //met la broche en entree
    digitalWrite(BP1, HIGH) ; // activation du pullup de la broche en entrée
    digitalWrite(BP2, HIGH) ; // activation du pullup de la broche en entrée
    Serial.begin(9600);
    vw_setup(2000);               // Bits par seconde (vous pouvez le modifier mais cela modifiera la portée). Voir la documentation de la librairie VirtualWire.
    vw_set_tx_pin(3);             // La broche 3 sera utilisée pour transmettre la DATA, vous pouvez changer de broche si vous le désirez.
}

void loop()
{
   envoimessage();
   delay(200);

}
```

```
void envoimessage()

{
    ETAT_BP1=digitalRead(BP1); // lit l'état du BP1 et met la valeur
0/1 dans la variable
    ETAT_BP2=digitalRead(BP2); // lit l'état du BP2 et met la valeur
0/1 dans la variable

      if (ETAT_BP1==0)
      {
              const char *msg="Du courrier !";
// C'est le message à envoyer.
              vw_send((uint8_t *)msg, strlen(msg));
              vw_wait_tx();
// On attend que le message complet soit envoyé.
              delay(200);                                          //
Très important sinon cela peut brouiller d'autres appareils !
       }
      if (ETAT_BP2==0)
      {
              const char *msg="Tu as un colis !";
// C'est le message à envoyer.
              vw_send((uint8_t *)msg, strlen(msg));
              vw_wait_tx();                                        //
On attend que le message complet soit envoyé.
              delay(200);
      }

}
```

Coté Récepteur :

Je teste l'afficheur LCD 16×2 I2C , pour l'affichage des informations coté maison. J'ajoute un bouton RAZ pour mise à zéro de l'afficheur marquant “ Pas de courrier” .

Schéma montage et code



Test_LCD_I2C_433mhz_Recepteur.ino

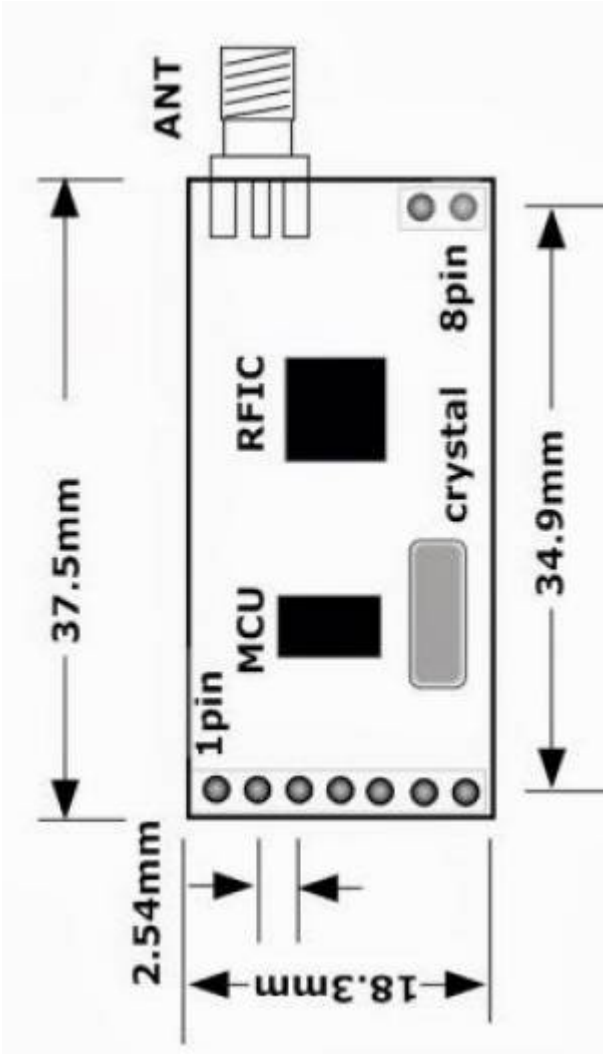
```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <VirtualWire.h>

LiquidCrystal_I2C lcd(0x27,16,2);  // set the LCD address to 0x20 for a
16 chars and 2 line display

String stringone = " ";
char stringtwo[20] = " "; // déclaration de la variable stringtwo à 20
caractères avec un espace vide à l'initialisation
const int BPRAZ=7; // la Broche 7 reçoit l'info de RAZ

int ETAT_BPRAZ; // Déclaration variable état bouton RAZ

void setup()
{
  lcd.init();                      // initialise l'afficheur
  pinMode(BPRAZ, INPUT);          // met la broche 7 en entrée
  digitalWrite(BPRAZ, HIGH);    // activation du pullup de la broche 7
en entrée


  // Print a message to the LCD.
  lcd.backlight();
  lcd.print("Bonjour ... "); // Test l'afficheur LCD , reste afficher
tant qu'aucun courrier ou colis de reçu à l'initialisation
  delay(1000);

}

void loop()
{
```

```
   reception_message();// on vérifie si on reçoit une information de la
BAL
   raz_message(); // on vérifie que le bouton RAZ n'est pas appuyer


}

void raz_message()
 {
   ETAT_BPRAZ=digitalRead(BPRAZ); // lit l'état du BPRAZ et met la
valeur 0/1 dans la variable
     if (ETAT_BPRAZ==0)
        {
         lcd.init();
         lcd.print("pas de courrier ... "); // quand on a relevé le
courrier ou colis on peut remettre à zéro l'afficheur
        }

 }


void reception_message()

{
    vw_setup(2000);               // Bits par seconde (vous pouvez le
modifier mais cela modifiera la portée). Voir la documentation de la
librairie VirtualWire.
    vw_set_rx_pin(5);             // C'est sur cette broche que l'on
reliera les broches DATA du récepteur, vous pouvez changez de broche si
vous le désirez.
    vw_rx_start();                // On démarre le récepteur.


     uint8_t buf[VW_MAX_MESSAGE_LEN];
     uint8_t buflen = VW_MAX_MESSAGE_LEN;
    if (vw_get_message(buf, &buflen)) // On test afin de savoir si un
message est reçu.
        {// Un message est reçu.

      int b;
      stringone = " ";
      for (b = 0; b < buflen; b++)
            {
               stringtwo[b] = (buf[b]);
           stringone =  stringone + stringtwo[b] ; // concatène les
caractères reçus en une seule chaîne
            }

        lcd.init();
        lcd.print(stringone); // affichage du texte " Du courrier" ou "
```

```
Tu as un colis"
        delay(2000);

          }

}
```

Test des APC220

Après les essais des modules 433mhz , j'ai décidé d'utiliser des modules APC220 qui permettent de créer une liaison série transparente entre deux Arduinos et surtout qui permettent de passer la barre des 95m de mes modules 433mhz.

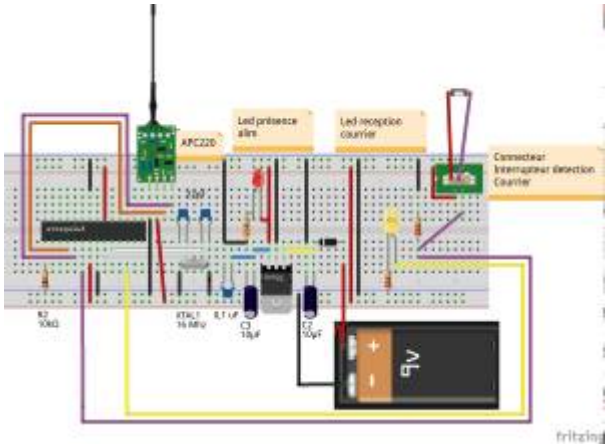


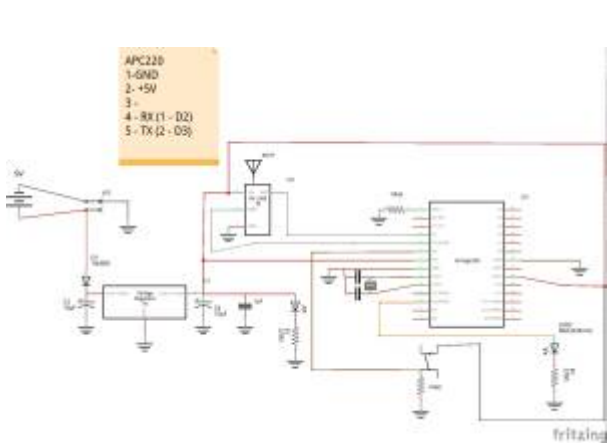
Pin	Definition	Detail
1	GND	0V Ground
2	VCC	3.3V-5.5V Power
3	EN	Enable the device when leave it disconnected or apply >1.6V Disable the device when apply <0.5V
4	RXD	UART RX
5	TXD	UART TX
6	AUX	UART Signal- Receive (low) Transmit (high)
7	SET	Set parameters (low)

Caractéristiques du module APC220

```
  Portée maximale : 1000m (en extérieur) à 2400 bps
  Puissance d'émission : de 1 à 20 mW (réglable, puissance à utiliser en
Europe : 10 mW)
  Vitesse de transmission RF : de 2400 à 9600 bps
  Vitesse UART : 1200 à 57600 bps
  Tampon (Buffer) de 256 octets
  Haute sensibilité (-112dbbm à 9600 bps)
  Modulation GFSK
  Interface UART / TTL
  Fréquence : de 418 Mhz à 455 Mhz (réglable, fréquence à utiliser en Europe
: 433 Mhz)
  Plus de 100 canaux (intervale de 200 Khz)
  Alimentation : 3,5V à 5,5V, 42 mA (maxi)
  Courant en réception : 28 mA, en veille : 5 uA
  Logiciel de configuration
  Le module interface série USB / TTL à base de CP2102 est compatible
Windows, Mac et Linux.
  Dimensions d'un module : 38 x 19 x 7 mm
  Dimensions d'une antenne : longueur : 58 mm, diamètre 7,7 mm
  Poids d'un module avec son antenne : 10 grammes
```

Schéma de l'émetteur coté BAL utilisant un ATméga328, avec son Quartz 16 Mhz, son régulateur 7805 5V, un interrupteur de détection de courrier, une LED de vérification, un APC220 pour la liaison série.





name="BAL_APC220_emetteur_BP_LED_UNO.ino

```
#include <SoftwareSerial.h> // inclus la librairie pour virtualiser un
port série sur les broches 2  (TX) et 3 (RX) de l'arduino Uno

SoftwareSerial myserial = SoftwareSerial(2,3); // TX, RX // Déclaration
de la liaison série sur les broches 2 et 3

const int APPUI=0; // constante état du BP - appui sur niveau bas
const int PAS_APPUI=1; // constante état du BP - relâché sur niveau
haut
```

```
// --- constantes des broches ---

const int BP=4; //declaration constante de broche
const int LED=6; //declaration constante de broche

// --- Déclaration des variables globales ---

int ETAT_BP; // variable d'état du bouton poussoir


void setup()   { // debut de la fonction setup()

// --- ici instructions à exécuter au démarrage ---
pinMode(2, INPUT); // déclaration de la broche 2 TX en Entrée
pinMode(3, OUTPUT);// déclaration de la broche 3 RX en Sortie
myserial.begin(9600); // déclaration de la vitesse de transmission,
maximum 9600bps pour la liaison série virtuelle
Serial.begin(9600);

pinMode(LED, OUTPUT); //met la broche en sortie

pinMode(BP, INPUT); //met la broche en entree

digitalWrite(BP, HIGH) ; // activation du pullup de la broche en entrée


}
void loop(){ // debut de la fonction loop()

// --- ici instructions à exécuter par le programme principal ---

ETAT_BP=digitalRead(BP); // lit l'état du BP et met la valeur 0/1 dans
la variable ETAT_BP

if (ETAT_BP==APPUI){ // si l'état du BP est appuyé (càd si variable
état BP = 0)
// Attention à bien utiliser == et non =

  digitalWrite(LED,1); // allume la LED
  myserial.println(ETAT_BP); //envoi l'état du Bouton la valeur "0" =
48 en ASCII , on envoi la valeur 48 sur le récepteur
  Serial.println(ETAT_BP);

  delay(100);

}

else { // sinon (càd si variable état bp=1)
```

```
  digitalWrite(LED,0); // éteint la LED
}

}
```