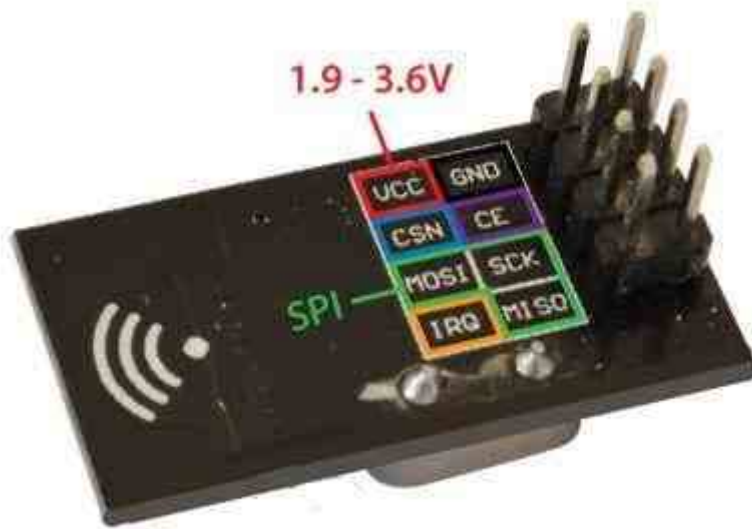


# NRF24L01



## Faire dialoguer un Raspberry et un Arduino via nRF24L01

[Arduino et Raspberry : le Dialogue](#)

### Datasheet NRF24L01

[nrf24l01\\_prelim\\_prod\\_spec\\_1\\_2.pdf](#)

[Explications Librairie NF24 EN](#)



*Chaque seconde, un nombre est émis par un module nRF24L01 branché à une carte ESP32 ou ESP8266.*

```

*****/

#include <SPI.h>
#include "nRF24L01.h"
#include "RF24.h"

int compteur = 0;

RF24 radio(4, 5);
const uint64_t adresse = 0x1111111111;
const int taille = 32;
char message[taille + 1];

void setup(void)
{
  Serial.begin(115200);
  Serial.println("Emetteur de donnees");
  radio.begin();
  radio.openWritingPipe(adresse);
}

void loop(void)
{
  compteur++;
  itoa(compteur, message, 10);
  Serial.print("J'envoie maintenant "); // pour débogage
  Serial.println(message);

  radio.write( message, taille ); // émission du message via nRF24L01

  delay(1000);
}

```

### Esp32\_NRF24L01\_Reception001.ino

```

/*****

Sketch permettant à un ESP32 ou un ESP8266 de recevoir des messages
en provenance d'un autre microcontrôleur par l'entremise d'un
module nRF24L01.
Les messages reçus sont affichés dans le moniteur série.

*****/

#include <SPI.h>
#include "nRF24L01.h"

```

```
#include "RF24.h"

RF24 radio(4, 5);

const uint64_t adresse = 0x1111111111;
const int taille = 32;
char message[taille + 1];

void setup(void)
{
  Serial.begin(115200);
  Serial.println("Recepteur RF24");
  radio.begin();
  radio.openReadingPipe(1, adresse);
  radio.startListening();
}

void loop(void)
{
  while ( radio.available() )
  {
    radio.read( message, taille );
    Serial.print("Message recu : ");
    Serial.println(message);
  }
}
```

## Maquette de test

[ESP32\\_BP\\_NRF24L01\\_Reception002\\_GL.ino](#)

```
#include <DFRobot_MCP23017.h>

#include <SPI.h>
#include "nRF24L01.h"
#include "RF24.h"

DFRobot_MCP23017 mcp(Wire, /*addr =*/0x27); //constructor, change the
Level of A2, A1, A0 via DIP switch to revise the I2C address within
0x20~0x27.
#define Led0 mcp.eGPB0
#define Led1 mcp.eGPB1
#define Bp0 mcp.eGPA0
#define Bp1 mcp.eGPA1
```

```

int MemLed0 =0;
int MemBp0 =0;
int tempo0 =50;
int Mmentier = 0;

int MemLed1 =0;
int MemBp1 =0;
int tempo1 =50;

int compteur = 0;

//definition tunnel de communication
#define tunnel1 "PIPE2"
#define tunnel2 "PIPE1"
#define tunnel3 "PIPE3"
#define tunnel4 "PIPE4"
#define tunnel5 "PIPE5"
#define tunnel6 "PIPE6"

RF24 radio(4, 5);
const byte addresses[][6] = {tunnel1, tunnel2, tunnel3, tunnel4,
tunnel5, tunnel6};
const int taille = 32;
char message[taille + 1];
char messageACK[taille +1];

int LectBp0(int ValMessage) {
// Bouton 0 -- Led0

//int valeurBp0 = mcp.digitalRead(mcp.eGPA0);
int valeurBp0 = ValMessage;

//Temps 0
if(valeurBp0== 0 && MemBp0 == 0 && MemLed0 == 0){
mcp.digitalWrite(mcp.eGPB0, LOW); MemBp0 = 0; MemLed0 =0;
//delay(tempo0);DebugVar(0,valeurBp0);
}
//Temps 1
if (valeurBp0 == 1 && MemBp0 ==0 && MemLed0 == 0) {
mcp.digitalWrite(mcp.eGPB0, HIGH);MemBp0 =1;MemLed0 = 1;
//delay(tempo0);DebugVar(1,valeurBp0);
}
//Temps 2
if (valeurBp0 == 0 && MemBp0 ==1 && MemLed0 == 1) {
mcp.digitalWrite(mcp.eGPB0, HIGH);MemBp0 =0;MemLed0 = 1;

```

```
        //delay(tempo0);DebugVar(2,valeurBp0);
    }
    //Temps 3
    if (valeurBp0 == 1 && MemBp0 ==0 && MemLed0 == 1) {
        mcp.digitalWrite(mcp.eGPB0, LOW);MemBp0 =1;MemLed0 = 0;
        //delay(tempo0);DebugVar(3,valeurBp0);
    }
    //Temps 4
    if (valeurBp0 == 0 && MemBp0 ==1 && MemLed0 == 0) {
        mcp.digitalWrite(mcp.eGPB0, LOW);MemBp0 =0;MemLed0 = 0;
        //delay(tempo0);DebugVar(4,valeurBp0);
    }

    return(valeurBp0);
}

int LectBp1(){
    // Bouton 1 -- Led1

    uint8_t valeurBp1 = mcp.digitalRead(mcp.eGPA1);
    //uint8_t valeurBp1 = );

    //Temps 0
    if(valeurBp1== 0 && MemBp1 == 0 && MemLed1 == 0){
        mcp.digitalWrite(mcp.eGPB1, LOW); MemBp1 = 0; MemLed1 =0;
        //delay(tempo0);DebugVar(0,valeurBp0);
    }
    //Temps 1
    if (valeurBp1 == 1 && MemBp1 ==0 && MemLed1 == 0) {
        mcp.digitalWrite(mcp.eGPB1, HIGH);MemBp1 =1;MemLed1 = 1;
        //delay(tempo0);DebugVar(1,valeurBp0);
    }
    //Temps 2
    if (valeurBp1 == 0 && MemBp1 ==1 && MemLed1 == 1) {
        mcp.digitalWrite(mcp.eGPB1, HIGH);MemBp1 =0;MemLed1 = 1;
        //delay(tempo0);DebugVar(2,valeurBp0);
    }
    //Temps 3
    if (valeurBp1 == 1 && MemBp1 ==0 && MemLed1 == 1) {
        mcp.digitalWrite(mcp.eGPB1, LOW);MemBp1 =1;MemLed1 = 0;
        //delay(tempo0);DebugVar(3,valeurBp0);
    }
    //Temps 4
    if (valeurBp1 == 0 && MemBp1 ==1 && MemLed1 == 0) {
        mcp.digitalWrite(mcp.eGPB1, LOW);MemBp1 =0;MemLed1 = 0;
        //delay(tempo0);DebugVar(4,valeurBp0);
    }
    return(valeurBp1);
}
```

```
}

void envoiemessage1(){
  radio.stopListening();
  itoa(255,messageACK, 10);
  radio.write(messageACK, taille);
  Serial.print("messageACK1 = ");Serial.println(messageACK);
  delay(20);
  radio.startListening();
}

void envoiemessage2(){
  radio.stopListening();
  itoa(128,messageACK, 10);
  radio.write(messageACK, taille);
  Serial.print("messageACK2 = ");Serial.println(messageACK);
  delay(20);
  radio.startListening();
}

void setup() {
  Serial.begin(115200);

  Serial.println("Recepteur RF24");
  radio.begin();
  radio.setChannel(125);
  radio.setPALevel(RF24_PA_MIN);
  radio.setDataRate(RF24_250KBPS);
  radio.openWritingPipe(adresses[0]); // Ouverture tunnel1 en ECRITURE
  radio.openReadingPipe(1, adresses[1]); // Ouverture tunnel2 en
LECTURE
  radio.openReadingPipe(2, adresses[2]); // Ouverture tunnel3 en
LECTURE
  radio.openReadingPipe(3, adresses[3]); // Ouverture tunnel4 en
LECTURE
  radio.openReadingPipe(4, adresses[4]); // Ouverture tunnel5 en
LECTURE
  radio.openReadingPipe(5, adresses[5]); // Ouverture tunnel6 en
LECTURE

  while(mcp.begin() != 0){
    Serial.println("Initialization of the chip failed, please confirm
that the chip connection is correct!");
  }
}
```

```
    delay(1000);
}

mcp.pinMode(Bp0, INPUT);
mcp.pinMode(Bp1, INPUT);
mcp.pinMode(mcp.eGPB0, OUTPUT);
mcp.pinMode(Led0, OUTPUT);
mcp.digitalWrite(mcp.eGPB0, LOW);
mcp.digitalWrite(Led1, LOW);

}

void loop() {

//Reception Message
radio.startListening();
if (radio.available()){
while ( radio.available() )
{
    radio.read( message, taille );
    Serial.print("Message recu : ");
    Serial.println(message);
}
    delay(20);
}
delay(5);

int Mentier = atoi(message);

if ( Mentier == 0 && MMentier == 0 ) {

    message[0] = '\0'; Mentier = 0;
    mcp.digitalWrite(mcp.eGPB0, LOW);
    delay(500);
}
if ( Mentier == 1 && MMentier == 0 ) {

    message[0] = '\0'; Mentier = 0;
    mcp.digitalWrite(mcp.eGPB0, HIGH);
    MMentier = 1;
    envoiemessage1();
    delay(500);
}
if ( Mentier == 1 && MMentier == 1 ) {
    message[0] = '\0'; Mentier = 0;
```

```
mcp.digitalWrite(mcp.eGPB0, LOW);  
MMentier = 0;  
envoiemessage2();  
delay(500);  
}  
}
```

## L'achat de nrf24L01

[Pas presser sur Aliexpress](#)

[Plus rapide sur Gotronic ... mais plus cher](#)

From:

<https://www.magenealogie.chanterie37.fr/www/fablab37110/> - **Castel'Lab le Fablab MJC de Château-Renault**

Permanent link:

<https://www.magenealogie.chanterie37.fr/www/fablab37110/doku.php?id=start:arduino:nrf24l01&rev=1653232405>

Last update: **2023/01/27 16:08**

