Notes d'applications Pmod™ et Arduino™ (Tome 2)

Conditions d'utilisations et limite de responsabilité

Les notes d'applications de ce document ont été conçues avec la plus grande attention. Tous les efforts ont été mis en œuvre pour éviter les anomalies. Toutefois, nous ne pouvons garantir que ces notes d'applications soit à 100% exempt de toute erreur. Les informations présentes dans cette documentation sont données à titre indicatif. Il est important de toujours considérer les programmes sources présents dans ce document comme des programmes en version Béta. Lextronic ne pourra en aucun cas être tenu responsable de dommages quels qu'ils soient (intégrant, mais sans limitation, les dommages pour perte de bénéfice commercial, interruption d'exploitation commerciale, perte d'informations et de données à caractère commercial ou de toute autre perte financière) provenant de l'utilisation ou de l'incapacité à pouvoir utiliser les notes d'applications de ce document, même si Lextronic a été informé de la possibilité de tels dommages. Ces notes d'applications sont exclusivement conçues dans un but pédagogique (essentiellement pour l'apprentissage de la programmation). Nous ne pouvons donner aucune garantie de leur fonctionnement pour une utilisation au sein de vos propres applications, ni pour une utilisation de ces dernières au sein d'applications à caractère professionnel . De manière général, ces notes d'applications ne sont pas conçues, ni destinées, ni autorisées pour expérimenter, développer ou être intégrées au sein d'applications dans lesquelles une défaillance de celles-ci pourrait créer une situation dangereuse pouvant entraîner des pertes financières, des dégâts matériel, des blessures corporelles ou la mort de personnes ou d'animaux. Si vous utilisez ces notes d'applications volontairement ou involontairement pour de telles applications non autorisées, vous vous engagez à soustraire Lextronic de toute responsabilité et de toute demande de dédommagement.

Schémas de raccordement

Les schémas de raccordement de cette documentation ont été réalisés à l'aide du logiciel Fritzing.

http://fritzing.org/home/ Ces schémas sont distribués sous licence CC Attribution-ShareALike

Librairies additionnelles

Certains code sources font appel à des librairies externes (qu'il vous faudra télécharger). Une recherche sur Internet vous permettra de trouver aisément ces librairies. Certaines de ces librairies existent sous différentes versions. En cas de non fonctionnement d'un programme, pensez à tester à nouveau ce dernier avec une version de librairie différente. Ces librairies sont distribuées selon divers types de licence. Merci de prendre connaissance de ces licences avant leur utilisation.

Copyright et appellations commerciales

Toutes les marques, les procédés, les références et les appellations commerciales des produits cités dans cette documentation appartiennent à leur propriétaire et Fabricant respectif.

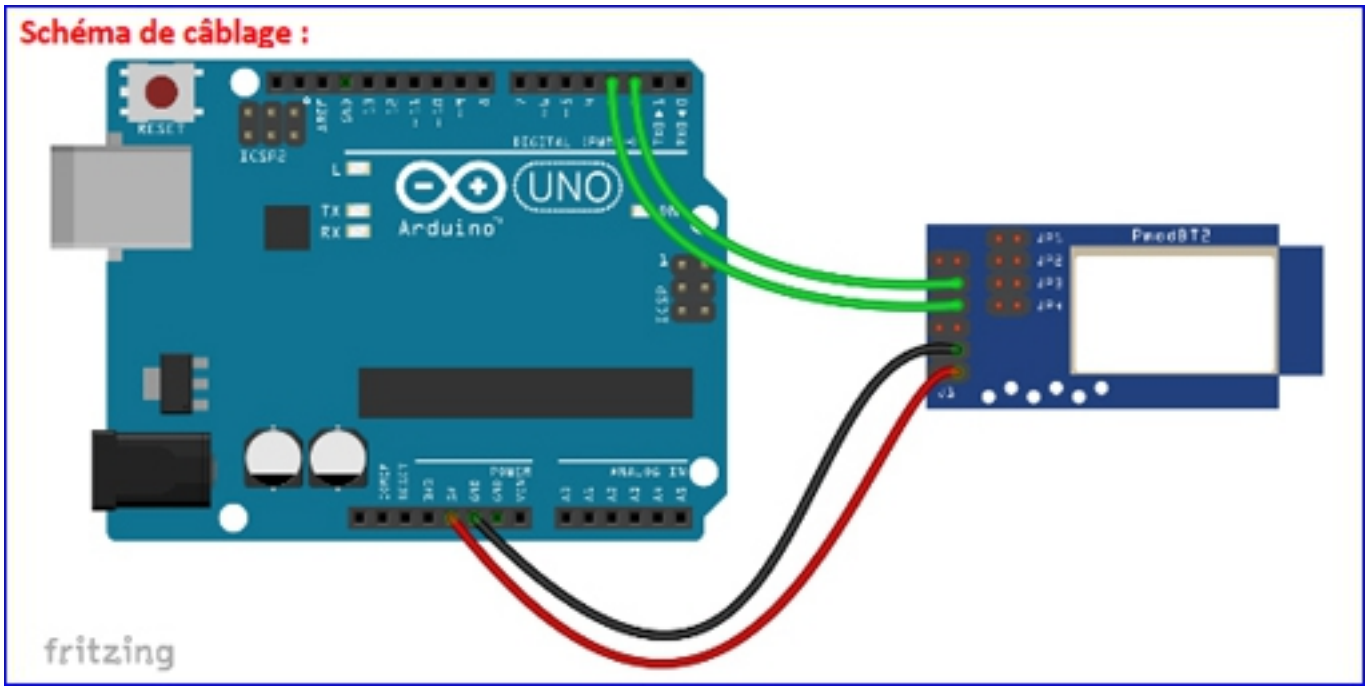
Les codes sources et les schémas de ce document sont téléchargeables ici :
https://www.lextronic.fr/~lextronic_doc/Pmod_APP.zip

Programme Arduino :

```
/************************************************************************
*
*   Test du module Pmod BT2
*
************************************************************************
* Description:  Pmod_BT2
* L'envoi en Bluetooth depuis une tablette d'un '0' ou d'un '1' éteint ou
* allume la led reliée à la broche 13 de l'Arduino.
*
* Matériel
*      1. Arduino Uno
*      2. Module Pmod BT2
*
************************************************************************/
#define led 13                    // affectation des broches
#define RX 2
#define TX 3


word octet_recu;                  // mot qui reçoit les trames émises

// création d'une liaison série
#include <SoftwareSerial.h>
SoftwareSerial bluetooth(RX,TX);


void setup()
{
Serial.begin(115200);             // initialisation du moniteur série
bluetooth.begin(115200);          // initialisation connexion série Bluetooth
pinMode(led,OUTPUT) ;             // configuration de la broche en sortie
}


void loop() {
recevoir();                       // appel de la procédure recevoir
if (octet_recu==49)               // si octet reçu est 1 (49=code ASCII de 1)
   {
   Serial.println("Led allumee");  // écriture dans le moniteur série
   digitalWrite(led,HIGH);         // allumage de la led
   }
if (octet_recu==48)               // si octet reçu est 0 (48=code ASCII de 0)
```

```
   {
    Serial.println("Led eteinte");        // écriture dans le moniteur série
    digitalWrite(led,LOW);               // extinction de la led
   }
delay(200);
   }

//procédure qui lit les trames de la tablette
void recevoir()
{
   if (bluetooth.available())            // si un caractère est présent sur la liaison bluetooth
   {
   octet_recu=bluetooth.read();          // lecture et stockage dans la variable octet_recu
   }
}
```

Schéma de câblage :



049 : PMODNIC100 : Module contrôleur réseau

Attention, ce module fonctionne sous 3,3 V

Programme Arduino :

```
/************************************************************************
*
*    Test du module Pmod contrôleur réseau
*
************************************************************************
* Description:  Pmod_NIC100
* Un serveur HTTP est créé, permettant la communication entre un ordinateur
* et la carte Arduino.
* Sur votre ordinateur lancer l'invite de commandes et taper la commande suivante :
* ping 192.168.1.83.
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod NIC100
*       3. Module Adafruit TXB0108
*       4. Câble réseau croisé RJ45
*
************************************************************************/

#include <SPI.h>                                    // appel des bibliothèques
#include <Ethernet.h>

byte MAC[] = { 0x00, 0x18, 0xE3, 0x01, 0xD4, 0x06 };    // adresse MAC du module Pmod NIC100
// voir étiquette livrée avec le module

IPAddress IP(192,168,1, 83);                        // adresse IP donné module Pmod NIC100

EthernetServer serveurHTTP(80);                     // création de l'objet serveur sur le port 80 (port HTTP)

void setup()
 {
 Ethernet.begin(MAC, IP);                           // connexion Ethernet avec l'adresse MAC et l'adresse IP
 serveurHTTP.begin();
 }

void loop()
 {
 }
```

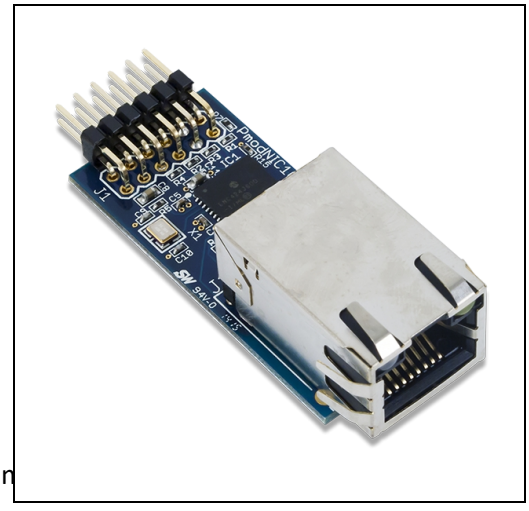
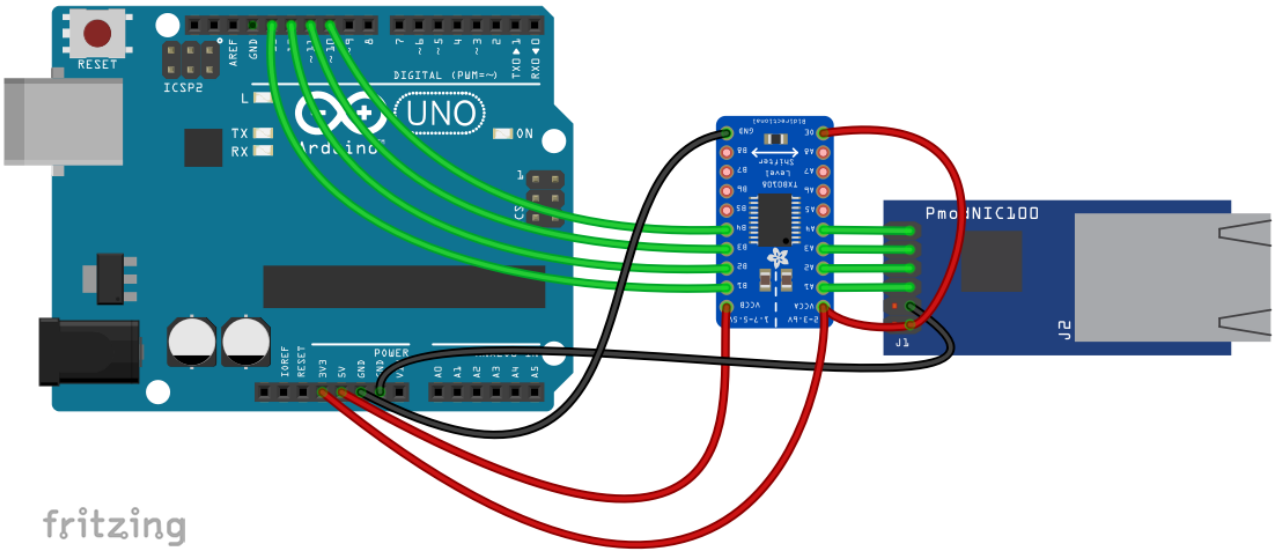


Schéma de câblage :



050 : PMODRF2 : Module transmetteur 2.4 GHz

Attention, ce module fonctionne sous 3,3 V

Programme Arduino :

```
/************************************************************************
*
*   Test du module Pmod transmetteur 2.4 GHz
*
************************************************************************
* Description:  Pmod_RF2
* Les données échangées entre deux modules RF2 sont affichés
* dans le moniteur série (basé sur l'exemple de la bibliothèque).
*
* Matériel
*      1. Arduino Uno
*      2. Module Pmod RF2 (télécharger la bibliothèque
*      https://github.com/karlp/Mrf24j40-arduino-library)
************************************************************************/

// Appel des bibliothèques
#include <SPI.h>
#include <mrf24j.h>

#define Reset 6                                   // affectation des broches
#define CS 10
#define Interruption 2


Mrf24j mrf(Reset, CS, Interruption);              // création de l'objet mrf

long last_time;
long tx_interval = 1000;
unsigned long current_time;

void setup(void)
{
  Serial.begin(9600);                             // initialisation de la liaison série
  mrf.reset();
  mrf.init();
  mrf.set_pan(0x0000);                            // adresse du canal
  mrf.address16_write(0x0001);                    // adresse du module N°1
  attachInterrupt(0, interrupt_routine, CHANGE);
  last_time = millis();
  interrupts();
}

void interrupt_routine()
{
```

```
   mrf.interrupt_handler();                    // création de l'objet mrf.interrupt
}

void loop()
{
 mrf.check_flags(&handle_rx, &handle_tx);
 current_time = millis();
 if (current_time - last_time > tx_interval)
   {
      last_time = current_time;
      Serial.println("Envoi de la chaine ");
      mrf.send16(0x0002, "Bonjour");           // envoi de la chaine au module N°2
   }
}

void handle_rx()
{
   Serial.print("Reception de la chaine ");
   Serial.print(mrf.get_rxinfo()->frame_length, DEC);
   Serial.println(" octets");

   if(mrf.get_bufferPHY())
   {
    Serial.println("Packet data (PHY Payload):");
    for (int i = 0; i < mrf.get_rxinfo()->frame_length; i++)
    {
       Serial.print(mrf.get_rxbuf()[i]);
    }
   }

   Serial.println("\r\nASCII data :");
   for (int i = 0; i < mrf.rx_datalength(); i++)
   {
      Serial.write(mrf.get_rxinfo()->rx_data[i]);
   }

   Serial.print("\r\nLQI/RSSI=");
   Serial.print(mrf.get_rxinfo()->lqi, DEC);
   Serial.print("/");
   Serial.println(mrf.get_rxinfo()->rssi, DEC);
}

void handle_tx()
{
   if (mrf.get_txinfo()->tx_ok)
   {
      Serial.println("Transmission OK");
   }
   else
   {
```

```
      Serial.print("Echec transmission ");
      Serial.print(mrf.get_txinfo()->retries);
      Serial.println(" essais\n");
    }
}
```

Programme Arduino : Programme N°2

```
/**************************************************************************
*
*    Test du module Pmod transmetteur 2.4 GHz
*
**************************************************************************
* Description:  Pmod_RF2
* Les données échangées entre deux modules RF2 sont affichés
* dans le moniteur série (basé sur l'exemple de la bibliothèque).
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod RF2 (télécharger la bibliothèque
*       https://github.com/karlp/Mrf24j40-arduino-library)
**************************************************************************/

// Appel des bibliothèques
#include <SPI.h>
#include <mrf24j.h>

#define Reset 6                                  // affectation des broches
#define CS 10
#define Interruption 2


Mrf24j mrf(Reset, CS, Interruption);             // création de l'objet mrf

long last_time;
long tx_interval = 1000;
unsigned long current_time;

void setup(void)
{
  Serial.begin(9600);                            // initialisation de la liaison série
  mrf.reset();
  mrf.init();
  mrf.set_pan(0x0000);                           // adresse du canal
  mrf.address16_write(0x0002);                   // adresse du module N°2
  attachInterrupt(0, interrupt_routine, CHANGE);
  last_time = millis();
  interrupts();
}
```

```
void interrupt_routine()
{
   mrf.interrupt_handler();                          // création de l'objet mrf.interrupt
}

void loop()
{
 mrf.check_flags(&handle_rx, &handle_tx);
 current_time = millis();
 if (current_time - last_time > tx_interval)
   {
      last_time = current_time;
      Serial.println("Envoi de la chaine ");
      mrf.send16(0x0001, "Lextronic");              // envoi de la chaine au module N°1
   }
}

void handle_rx()
{
   Serial.print("Reception de la chaine ");
   Serial.print(mrf.get_rxinfo()->frame_length, DEC);
   Serial.println(" octets");

   if(mrf.get_bufferPHY())
   {
    Serial.println("Packet data (PHY Payload):");
    for (int i = 0; i < mrf.get_rxinfo()->frame_length; i++)
    {
       Serial.print(mrf.get_rxbuf()[i]);
    }
   }

   Serial.println("\r\nASCII data :");
   for (int i = 0; i < mrf.rx_datalength(); i++)
   {
      Serial.write(mrf.get_rxinfo()->rx_data[i]);
   }

   Serial.print("\r\nLQI/RSSI=");
   Serial.print(mrf.get_rxinfo()->lqi, DEC);
   Serial.print("/");
   Serial.println(mrf.get_rxinfo()->rssi, DEC);
}

void handle_tx()
{
   if (mrf.get_txinfo()->tx_ok)
   {
      Serial.println("Transmission OK");
   }
```

```
    else
    {
      Serial.print("Echec transmission ");
      Serial.print(mrf.get_txinfo()->retries);
      Serial.println(" essais\n");
    }
}
```

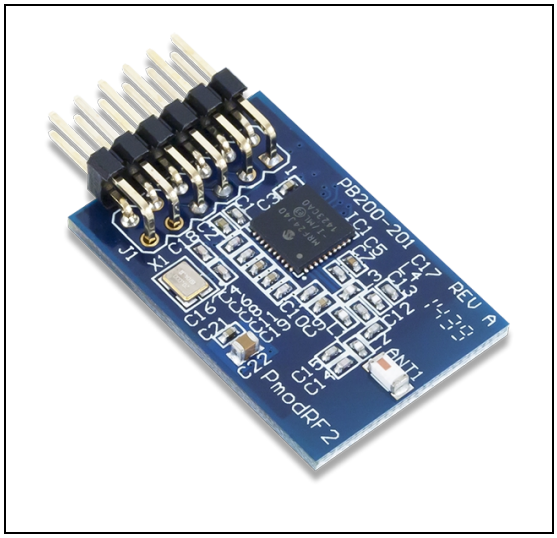
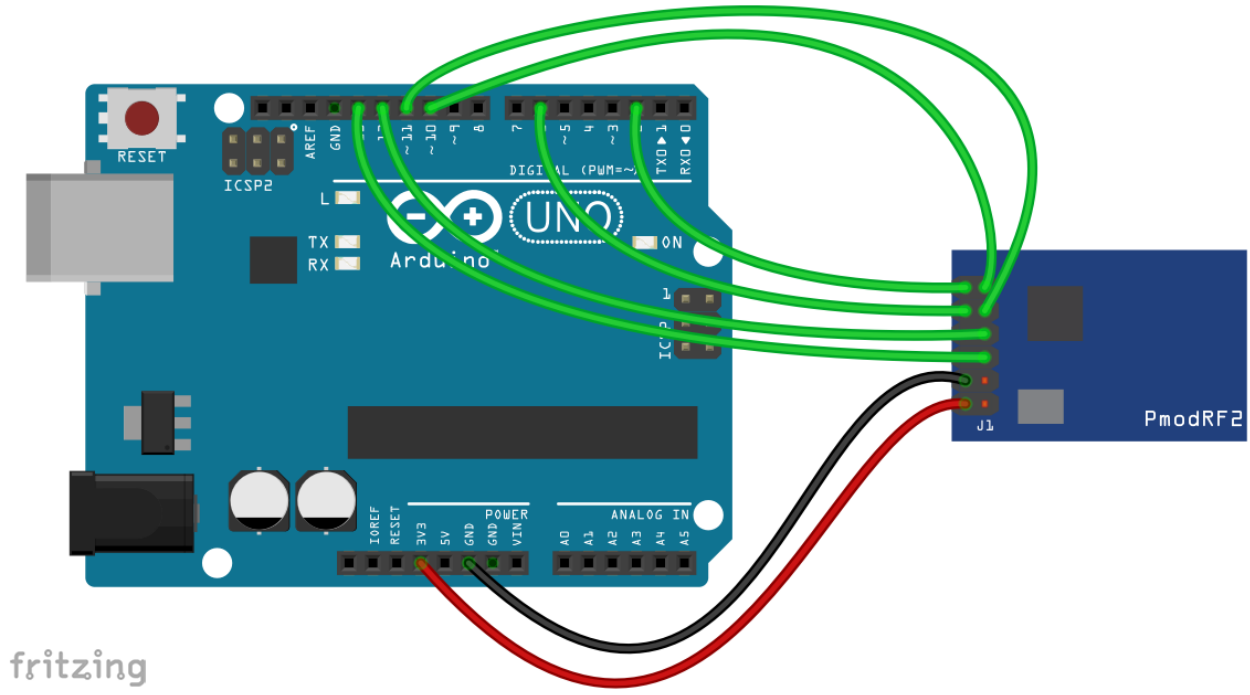


Schéma de câblage :



051 : PMODRS232 : Module RS232

Programme Arduino :

```
/************************************************************************
*
*   Test du module Pmod RS232
*
************************************************************************
* Description:  Pmod_RS232   Emetteur
* Une carte Arduino + module Pmod RS232 demande à une autre carte Arduino
* + module Pmod RS232 de lui envoyer une chaîne de caractères.
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod RS232 (Le cavalier JP1 n'est pas positionné,
*       le cavalier JP2 est positionné).
*
************************************************************************/
#define RX 9                                // affectation des broches
#define TX 8

#include <SoftwareSerial.h>                 // appel de la bibliothèque
SoftwareSerial RS232(RX, TX);               // création de l'objet RS232

int machaine[]={0,1,2,3,4,5,6,7,8,9};
int i;

void setup()
{
 RS232.begin(9600);                         // initialisation de la liaison série RS232
}

void loop()
{
if (RS232.available())                      // attente que le récepteur demande un envoi
  {
     if (RS232.read()=='0')                 // si un '0' a été envoyé
     {
     for (i=0;i<10;i=i+1)                   // envoi caractère par caractère
      {
      RS232.write(machaine[i]);
      delay(10);
      }
     }
  }
```

```
}
```

Programme Arduino : Programme Recepteur

```
/*************************************************************************
*
*   Test du module Pmod RS232
*
*************************************************************************
* Description:  Pmod_RS232 Récepteur
* Une carte Arduino + module Pmod RS232 demande à une autre carte Arduino
* + module Pmod RS232 de lui envoyerune chaîne de caractères.
*
* Matériel
*      1. Arduino Uno
*      2. Module Pmod RS232 (Le cavalier JP1 n'est pas positionné,
*      le cavalier JP2 est positionné).
*
*************************************************************************/
#define RX 9                                  // affectation des broches
#define TX 8

#include <SoftwareSerial.h>                   // appel de la bibliothèque
SoftwareSerial RS232(RX, TX);                 // création de l'objet RS232

int recu[20];
int i;

void setup()
{
 Serial.begin(9600);                          // initialisation du moniteur série
 RS232.begin(9600);                           // initialisation de la liaison série RS232
}

void loop()
{
    i = 0 ;
    while ( RS232.available()>0 )             // tant que des caractères sont présents sur la liaison RS232
    {
        recu[i] = RS232.read();               // lecture caractère par caractère
        Serial.print(recu[i]);                // affichage dans le moniteur série
        Serial.print(",");
        i++;
    }
RS232.print('0');                            // demande de l'envoi de la chaîne depuis l'émetteur
}
```

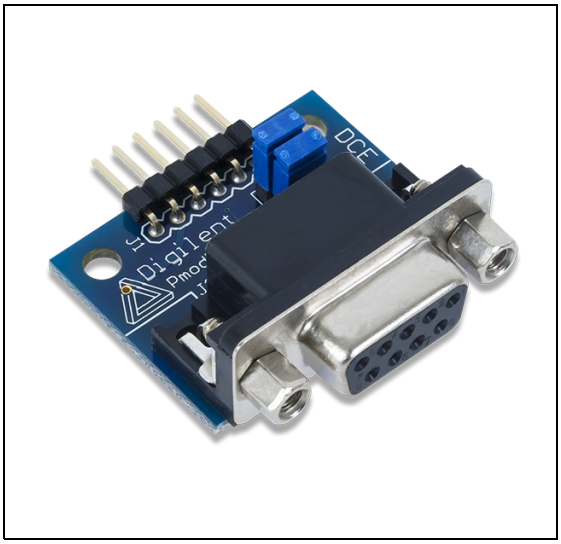
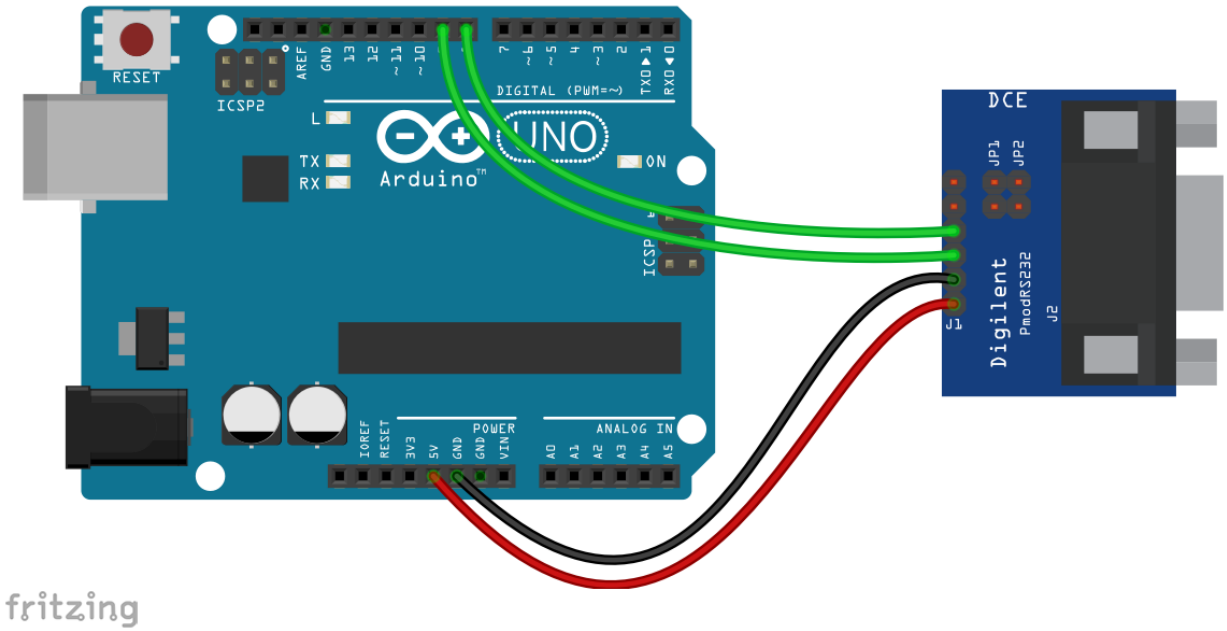


Schéma de câblage : Il faut deux schémas identiques (un pour l'émetteur et un pour le récepteur). Au niveau des connecteurs DB9 des modules Pmod, relier les masses et croiser les lignes Rx et Tx.



052 : PMODRTC : Module horloge RTC

Programme Arduino :

```
/************************************************************************
*
*    Test du module Pmod horloge
*
************************************************************************
* Description:  Pmod_RTC
* La date et l'heure sont affichées sur le moniteur série.
*
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod RTC
*
************************************************************************/

#include <Wire.h>                          // appel de la bibliothèque
#define MCP79410_Adresse 0x6F              // adresse I2C du module Pmod RTC

void setup()
{
 Serial.begin(9600);                       // initialisation de la liaison série
 Wire.begin();                             // initialisation de la liaison I2C
 Init_RTC();                               // initialisation du module RTC
}

void loop()
{
 DisplayRTCDay();                          // affichage jour de la semaine
 DisplayRTCData(4,6);                      // affichage date
 Serial.print(".");
 DisplayRTCData(5,5);
 Serial.print(".");
 Serial.print("20");
 DisplayRTCData(6,8);
 Serial.print("  ");
 DisplayRTCData(2,6);                      // affichage de l'heure
 Serial.print(":");
 DisplayRTCData(1,7);
 Serial.print(":");
 DisplayRTCData(0,7);
 Serial.println();
```

```
delay(1000);
}

// Initialisation du module Pmod RTC
void Init_RTC(void)
{
 WriteRTCByte(0,0x00);                  // configuration registre des secondes
 WriteRTCByte(1,0x55);                  // configuration registre des minutes
 WriteRTCByte(2,0x13);                  // configuration registre des heures
 WriteRTCByte(3,0x03);                  // configuration registre des jours de la semaine
 WriteRTCByte(4,0x09);                  // configuration registre du jour
 WriteRTCByte(5,0x08);                  // configuration registre du mois
 WriteRTCByte(6,0x17);                  // configuration registre de l'année
 WriteRTCByte(0,0x80);                  // démarrage de l'horloge
 delay(100);
}

void WriteRTCByte(const unsigned char adresse, const unsigned char donnee)
{
 Wire.beginTransmission(MCP79410_Adresse);
 Wire.write(adresse);
 Wire.write(donnee);
 Wire.endTransmission();
}

void DisplayRTCData(const unsigned char adresse, const unsigned char nb_bits)
{
 unsigned char data;
 data=ReadRTCByte(adresse);
 data=data & 0xFF>>(8-nb_bits);
 if (data<10)
 {
  Serial.print("0");
 }
 Serial.print(data,HEX);
}

void DisplayRTCDay()
{
 unsigned char data;
 data=ReadRTCByte(3);
 data=data & 0xFF>>(5);
 switch (data)
 {
 case 1:
  Serial.print("Lundi ");
  break;
 case 2:
```

```
  Serial.print("Mardi ");
  break;
 case 3:
  Serial.print("Mercredi ");
  break;
 case 4:
  Serial.print("Jeudi ");
  break;
 case 5:
  Serial.print("Vendredi ");
  break;
 case 6:
  Serial.print("Samedi ");
  break;
 case 7:
  Serial.print("Dimanche ");
  break;
 }
}
unsigned char ReadRTCByte(const unsigned char adresse)
{
 unsigned char data;

 Wire.beginTransmission(MCP79410_Adresse);
 Wire.write(adresse);
 Wire.endTransmission();

 Wire.requestFrom(MCP79410_Adresse,1);
 while (Wire.available())
 {
  data=Wire.read();
 }
 return data;
```

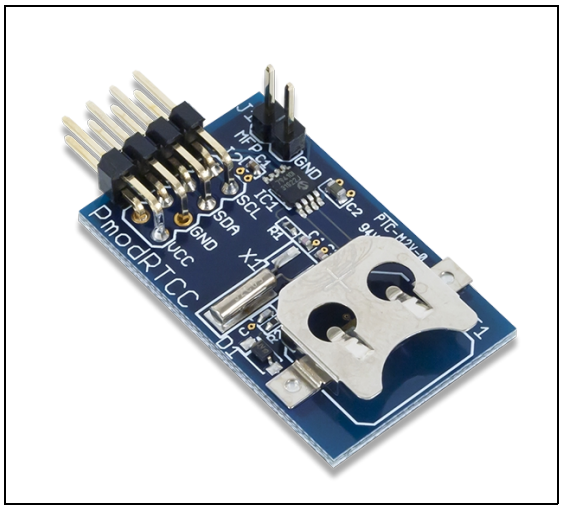
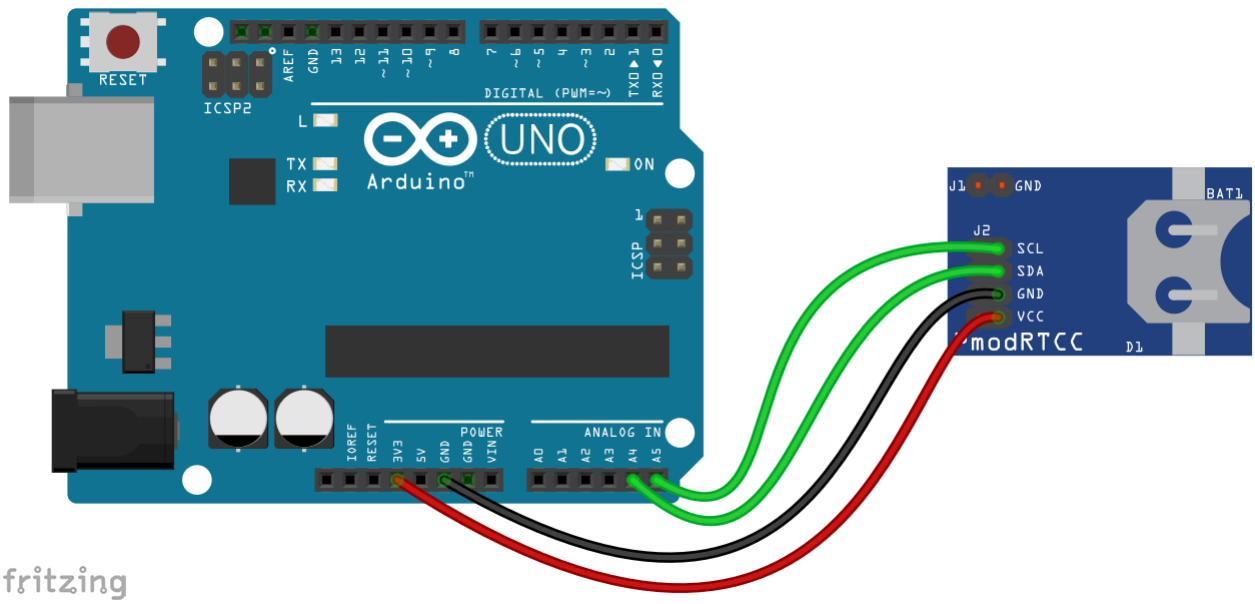


Schéma de câblage :



053 : PMODUSBUSART : Module USB < > Série

Programme Arduino :

```
/*****************************************************************************
*
*    Test du module Pmod USBUART
*
*****************************************************************************
* Description:  Pmod_USBUART
* Une carte Arduino + module Pmod USBUART échange des chaînes de caractères
* avec le logiciel ComTools depuis un port USB et le moniteur série.
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod USBUART (Le cavalier JP1 est positionné sur LCL-VCC).
*       3. Logiciel ComTools (à télécharger https://www.sonelec-musique.com/logiciels_freewares_comtools.html)
*       ATTENTION, il faut brancher le module avant de lancer le logiciel
*****************************************************************************/
#define RX 9                                        // affectation des broches
#define TX 8

#include <SoftwareSerial.h>                         // appel de la bibliothèque
SoftwareSerial USB(RX, TX);                         // création de l'objet USB

void setup()
{
 Serial.begin(9600);                                // initialisation de la liaison série
 USB.begin(9600);                                   // initialisation de la liaison série USB
 Serial.println("Test du module Pmod USB-UART");    // écriture dans le moniteur série
 USB.println("Connexion etablie");                  // écriture sur le logiciel ComTools
}

void loop()
{
 if (USB.available())                               // si une donnée est présente sur le port USB
 {
   Serial.write(USB.read());                        // écriture de cette donnée dans le moniteur série
 }
 if (Serial.available())                            // si une donnée est présente sur moniteur série
 {
   USB.write(Serial.read());                        // écriture de cette donnée dans le logiciel
 }
}
```

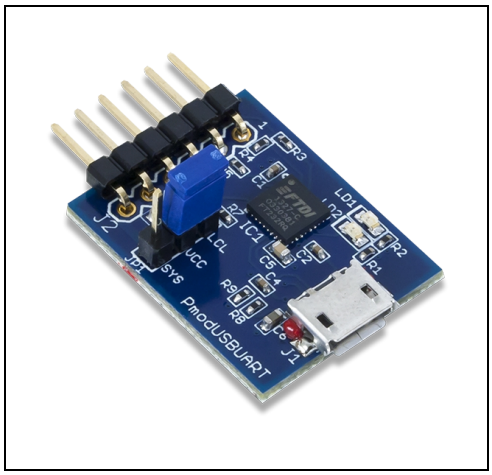
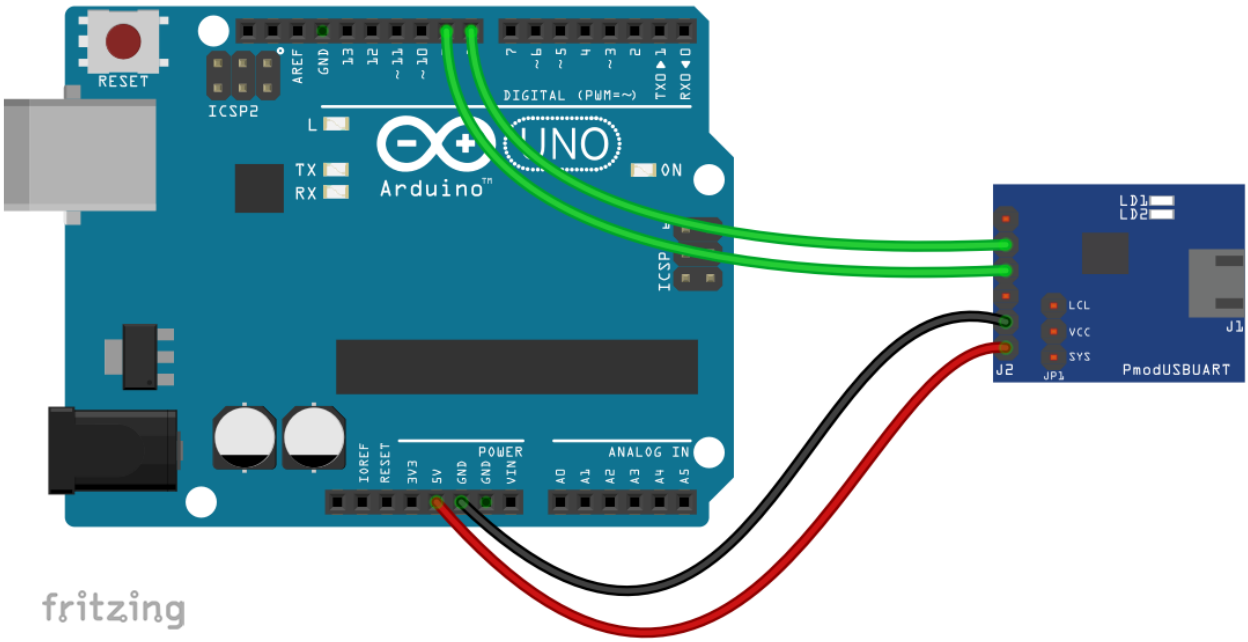
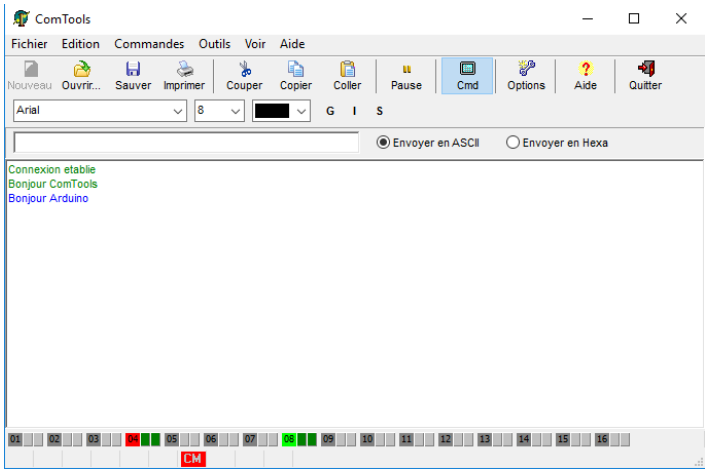


Schéma de câblage :



ComTools :



054 : PMODRS485 : Module RS485

Programme Arduino :

```
/*************************************************************************
*
*   Test du module Pmod RS485
*
*************************************************************************
* Description:  Pmod_RS485
* Une led est commandée depuis une liaison série RS485.
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod RS485 (le cavalier JP1 est présent)
*
* Câblage
*       Module<----------> Arduino
*       J1 broche 6       5 V
*       J1 broche 5       GND
*       J1 broche 4       7
*       J1 broche 3       9
*       J1 broche 2       8
*       J1 broche 1       6
*
*************************************************************************/
#define RX 9                                    // affectation des broches
#define TX 8
#define DE 7
#define RE 6

#include <SoftwareSerial.h>                     // appel de la bibliothèque
SoftwareSerial RS485(RX, TX);                   // création de l'objet RS485

int recu;

void setup()
{
 Serial.begin(9600);                            // initialisation du moniteur série
 RS485.begin(9600);                             // initialisation de la liaison série RS485
 pinMode(DE,OUTPUT);                            // configuration de la broche DE en sortie
 pinMode(RE,OUTPUT);                            // configuration de la broche RE en sortie
 digitalWrite(DE, HIGH);                        // validation de l'émission
 digitalWrite(RE, HIGH);                        // blocage de la réception
 Serial.println("Commande led par liaison série:");
 Serial.println("Extinction:0 Allumage:1");
```

```
 delay(500);
}

void loop()
{
if (Serial.available())                    // si la liaison série RS232 (moniteur série) reçoit un caractère
 {
   recu = Serial.read();                   // lecture et stockage dans la variable recu
   RS485.write(recu);                      // envoi de la variable recu
 }
}
```

Programme Arduino : Programme Recepteur

```
/*************************************************************************
*
*   Test du module Pmod RS485
*
*************************************************************************
* Description:  Pmod_RS485
* Une led est commandée depuis une liaison série RS485.
*
* Matériel
*      1. Arduino Uno
*      2. Module Pmod RS485 (le cavalier JP1 est présent)
*
* Câblage
*      Module<----------> Arduino
*      J1 broche 6      5 V
*      J1 broche 5      GND
*      J1 broche 4      7
*      J1 broche 3      9
*      J1 broche 2      8
*      J1 broche 1      6
*************************************************************************/
#define RX 9                                  // affectation des broches
#define TX 8
#define DE 7
#define RE 6
#define led 13

#include <SoftwareSerial.h>                   // appel de la bibliothèque
SoftwareSerial RS485(RX, TX);                 // création de l'objet RS485

char recu;

void setup()
{
```

```
 Serial.begin(9600);                    // initialisation du moniteur série
 RS485.begin(9600);                     // initialisation de la liaison série RS485
 pinMode(DE,OUTPUT);                    // configuration de la broche DE en sortie
 pinMode(RE,OUTPUT);                    // configuration de la broche RE en sortie
 pinMode(led,OUTPUT);                   // configuration de la broche led en sortie
 digitalWrite(DE, LOW);                 // blocage de l'émission
 digitalWrite(RE, LOW);                 // validation de la réception
 digitalWrite(led,LOW);                 // extinction de la led
}

void loop()
{
if (RS485.available())                  // si la liaison série RS485 reçoit un caractère
 {
   recu = RS485.read();                 // lecture du caractère
   Serial.print("J'ai recu le caractère:");
   Serial.println(recu);
   if (recu=='0')
   {
    digitalWrite(led,LOW);              // extinction de la led
   }
   if (recu=='1')
   {
    digitalWrite(led,HIGH);             // allumage de la led
   }
 }
}
```

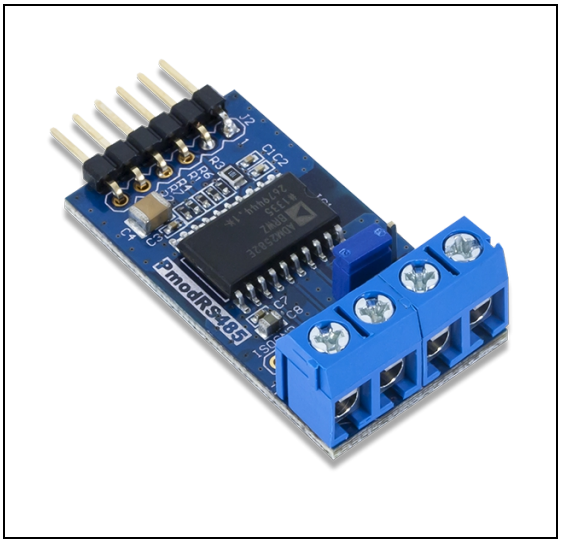


Schéma de câblage : Il faut câbler les deux modules RS485 de la façon suivante :

Module 1 ←----------→ Module 2

A	**Y**
B	**Z**
Y	**A**
Z	**B**