Notes d'applications modules Pmod™ et Arduino™

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

Modules Pmod™ boutons-poussoirs / interrupteurs

PMODBTN	PMODCDC1	PMODSWT	

Modules Pmod™ claviers / joystick / encodeurs

PMODKYPD	PMODJTK	PMODJSTK2	PMODENC

Modules Pmod™ leds/ afficheurs 7 segments à leds

PMODLED	PMOD8LD	PMODSSD	

Modules Pmod™ afficheurs LCD

PMODCLP	PMODCLS	PMODOLED	PMODOLEDRGB

Modules Pmod™ mémoire

PMODSF2	PMODSD		

Modules Pmod™ convertisseurs « Analogique / Numérique »

PMODAD1 PMODMIC3	PMODAD2	PMODAD5	PMODIA

Modules Pmod™ convertisseurs « Numérique / Analogique »

PMODDA1 PMODR2R	PMODDA2 PMODDPOT	PMODDA3 PMODI2S	PMODA4

Modules Pmod™ connecteurs

PMODCON3	PMODPS2		

Modules Pmod™ entrées / sorties

PMODOC1	PMODOD1	PMODIOXP	

Modules Pmod™ radio / GPS /bus

PMODBT2 PMODRS232	PMODRF2 PMODRS485	PMODWIF PMODNIC	PMODGPS PMODNIC100

Modules Pmod™ accéléromètres / gyroscopes / boussoles

PMODACL PMODNAV	PMODACL2	PMODGYRO	PMODCMPS

Modules Pmod™ capteurs divers

PMODTMP2 PMODTMP3 PMODTC1 PMODDPG1	PMODSONAR PMODMIC3 PMODALS	PMODISNS20	PMODLS1

Modules Pmod™ moteurs / servomoteurs

PMODHB3 PMODHB5 PMODDHB1	PMODSTEP		

Modules Pmod™ divers

PMODRTCC PMODUSBU PMODPMON1	PMODVLSHF	PMODAMP2	PMODAMP3

PMODBTN : Module 4 boutons-poussoirs

Programme Arduino :

```
/*****************************************************************************
* Description:  Pmod_BTN
* L'état du bouton poussoir est affiché dans le moniteur série.
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod BTN
*
*****************************************************************************/

boolean bp;
int index=0;
void setup()
{
Serial.begin(9600);                 // initialisation du moniteur série
for (int i=2; i<=5; i++)            // configuration des broches 2 à 5 en entrée
{
pinMode(i,INPUT);
 }
}

void loop()
{
for(int i=2; i<=5; i++)             // lecture de l'état des boutons
 {
bp=digitalRead(i);
delay(40);                          // anti-rebonds
if(bp==HIGH)
  {
Serial.print("Le bouton BTN");
index=i-2;                          // calcul de l'index du bouton
Serial.print(index);
Serial.println(" est actif.");
  }
 }
}
```

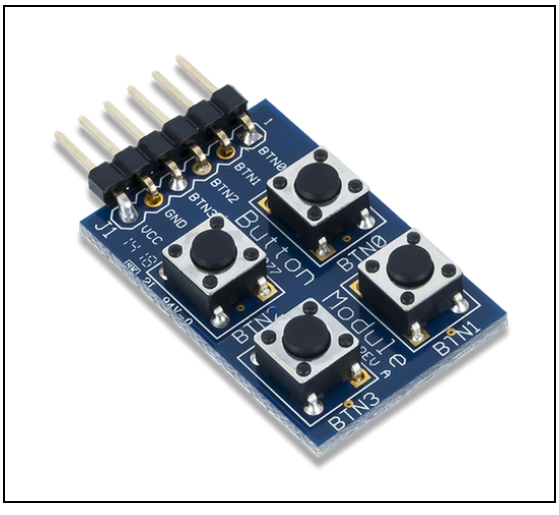
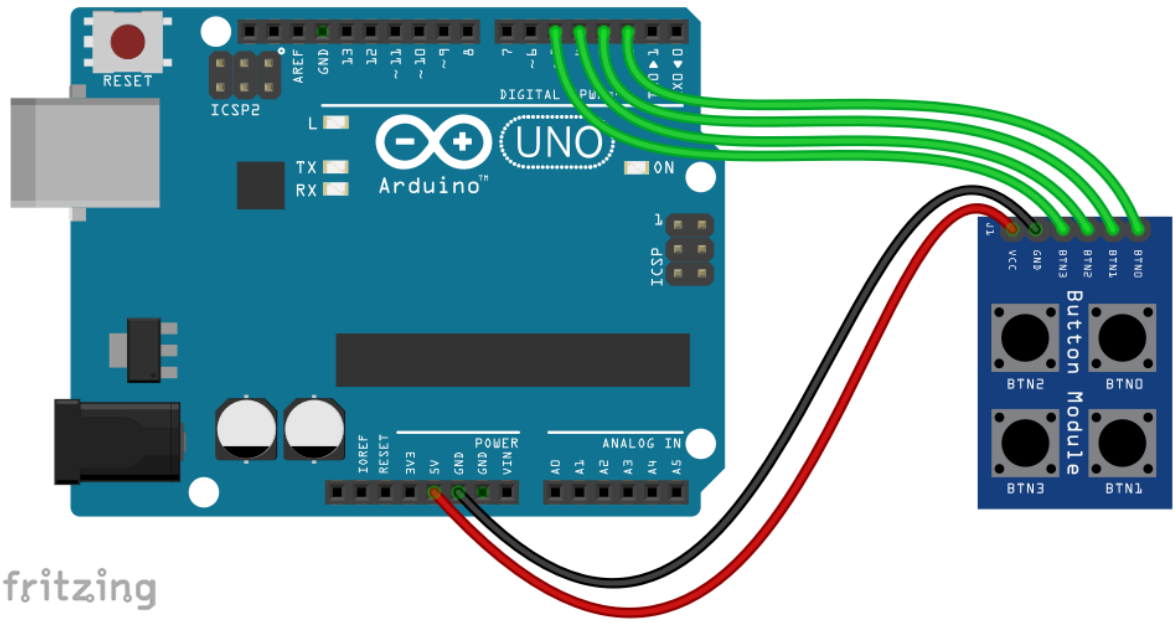


Schéma de câblage :



PMODCDC1 : Module 2 boutons capacitifs

Attention, ce module fonctionne sous 3,3 V

Programme Arduino :

```
/*************************************************************************
*
*   Test du module Pmod 2 boutons capacitifs
*
*************************************************************************
* Description:  Pmod_CDC1
* L'état du bouton est affiché dans le moniteur série.
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod CDC1
*
*************************************************************************/

// Affectation des broches
#define BTN_1 2
#define BTN_2 3

void setup()
{
 Serial.begin(9600);                // initialisation du moniteur série
 pinMode(BTN_1,INPUT);              // configuration des broches 2 en entrée
 pinMode(BTN_2,INPUT);              // configuration des broches 3 en entrée
}

void loop()
{
 if(digitalRead(BTN_1)==true)       // si le bouton 1 est actif
 {
  Serial.println("Bouton 1 actif");  // écriture dans le moniteur série
 }
 if(digitalRead(BTN_2)==true)       // si le bouton 2 est actif
 {
  Serial.println("Bouton 2 actif");  // écriture dans le moniteur
série
 }
 delay(100);
}
```

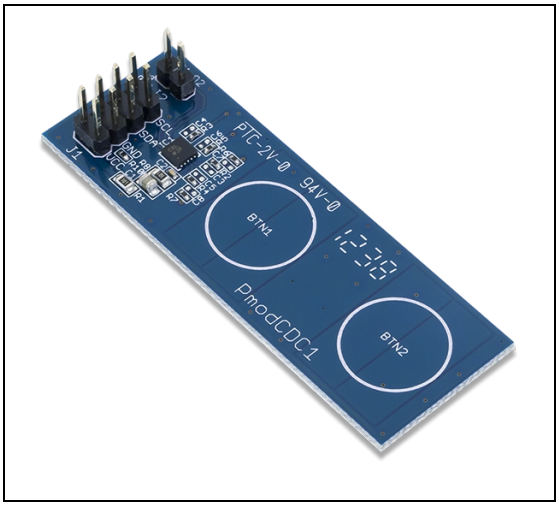
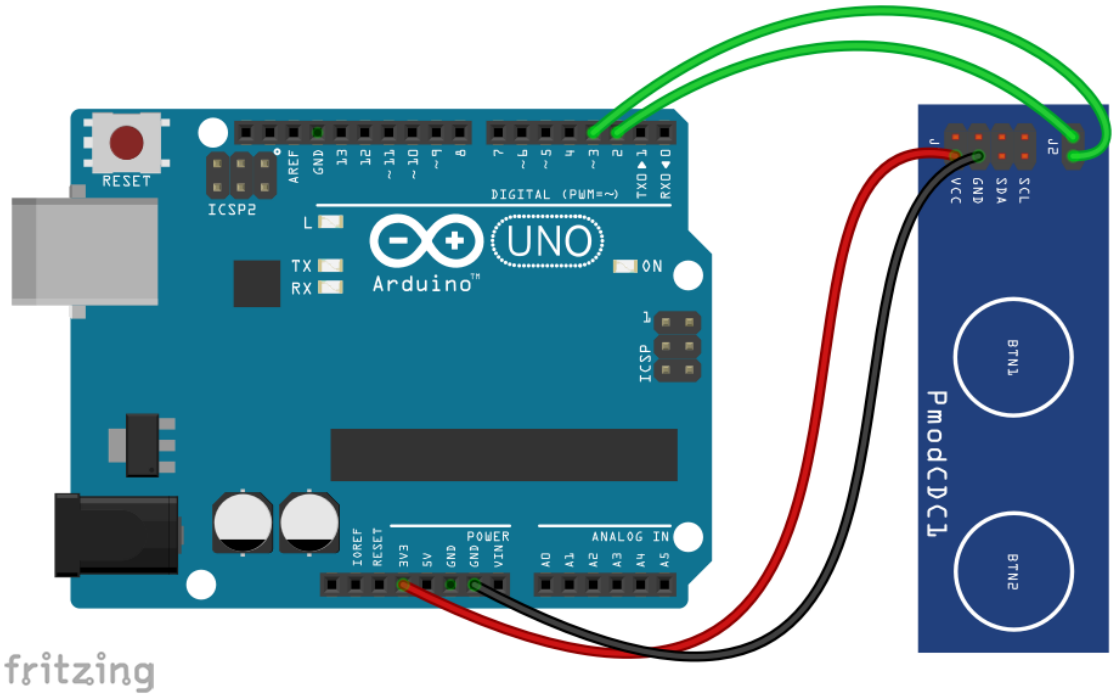
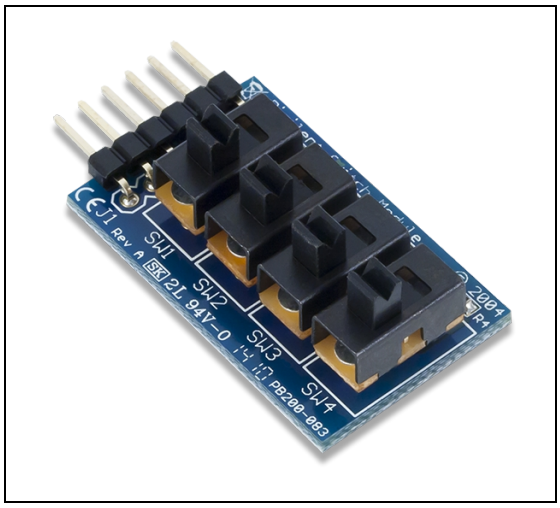


Schéma de câblage :



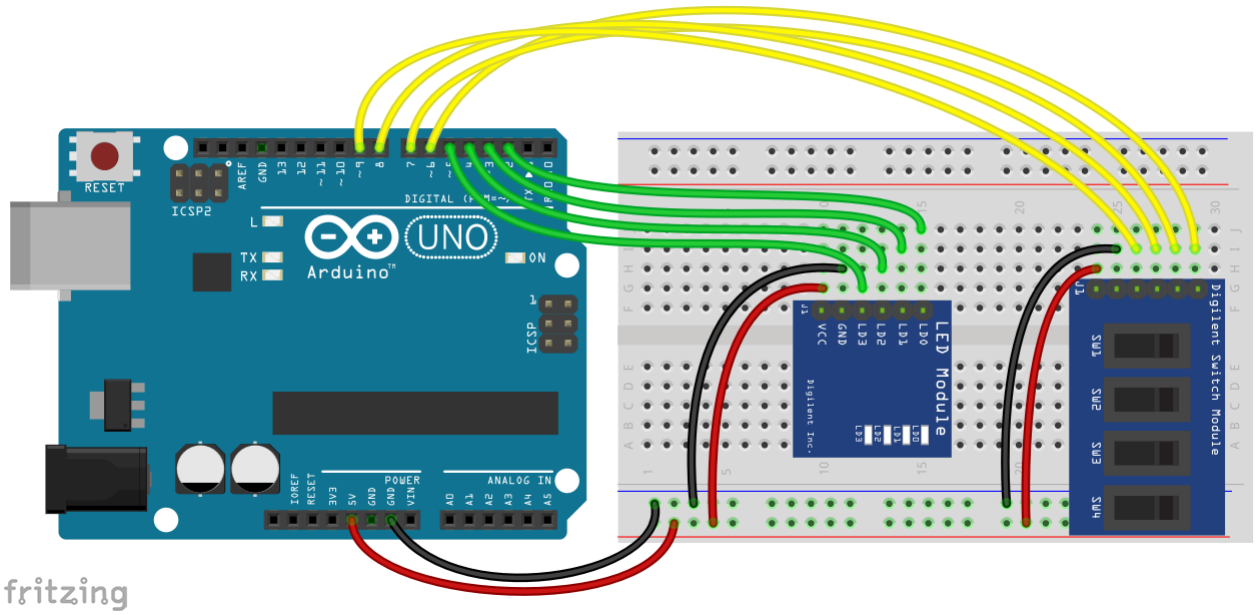
Programme Arduino :

```
/************************************************************************
*
*    Test du module Pmod 4 interrupteurs
*
************************************************************************
* Description:  Pmod_SWT
* L'état des interrupteurs est affiché sur un module LED
* et le nombre décimal équivalent dans le moniteur série
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod LED
*       3. Module Pmod SWT
*
************************************************************************/
// Affectation des broches
#define LED_0 2
#define LED_1 3
#define LED_2 4
#define LED_3 5
#define SWT_1 6
#define SWT_2 7
#define SWT_3 8
#define SWT_4 9

boolean inter_1;
boolean inter_2;
boolean inter_3;
boolean inter_4;
int nombre;

void setup()
{
 Serial.begin(9600);              // initialisation du moniteur série
 for (int i=2; i<=5; i++)         // configuration des broches 2 à 5 en sortie
 {
   pinMode(i,OUTPUT);
 }
 for (int i=6; i<=9; i++)         // configuration des broches 6 à 9 en entrée
 {
   pinMode(i,INPUT);
 }
```



```
}

void loop()
{
 inter_1=digitalRead(SWT_1);            // lecture de l'interrupteur SW1
 digitalWrite(LED_0,inter_1);           // commande de la led LED0 en fonction de l'état de l'interrupteur SW1
 inter_2=digitalRead(SWT_2);
 digitalWrite(LED_1,inter_2);
 inter_3=digitalRead(SWT_3);
 digitalWrite(LED_2,inter_3);
 inter_4=digitalRead(SWT_4);
 digitalWrite(LED_3,inter_4);
 nombre=inter_1+inter_2*2+inter_3*4+inter_4*8;   // conversion binaire-décimal
 Serial.print("Le nombre decimal est egal à ");          // affichage dans le moniteur série
 Serial.println(nombre);
}
```

Schéma de câblage :



PMODLED : Module 4 led

Programme Arduino :

```
/************************************************************************
*
*    Test du module Pmod 4 led
*
************************************************************************
* Description:  Pmod_LED
* Les led s'allument successivement les unes après les autres,
* puis s'éteignent.
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod LED
*
************************************************************************/

void setup()
{
for (inti=2; i<=5; i++)              // configuration des broches 2 à 5 en sortie
 {
pinMode(i,OUTPUT);
 }
}

void loop()
{
for(inti = 2; i<=5; i++)            // allumage des 4 led
 {
digitalWrite(i,HIGH);
delay(250);
 }
for(inti = 2; i<=5; i++)             // extinction des 4 led
 {
digitalWrite(i,LOW);
 }
delay(250);
}
```

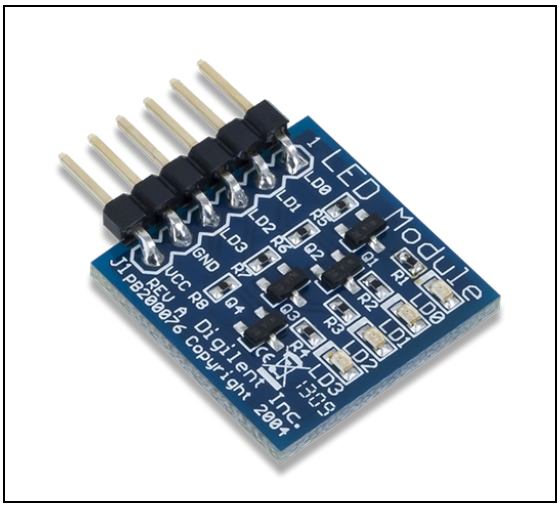
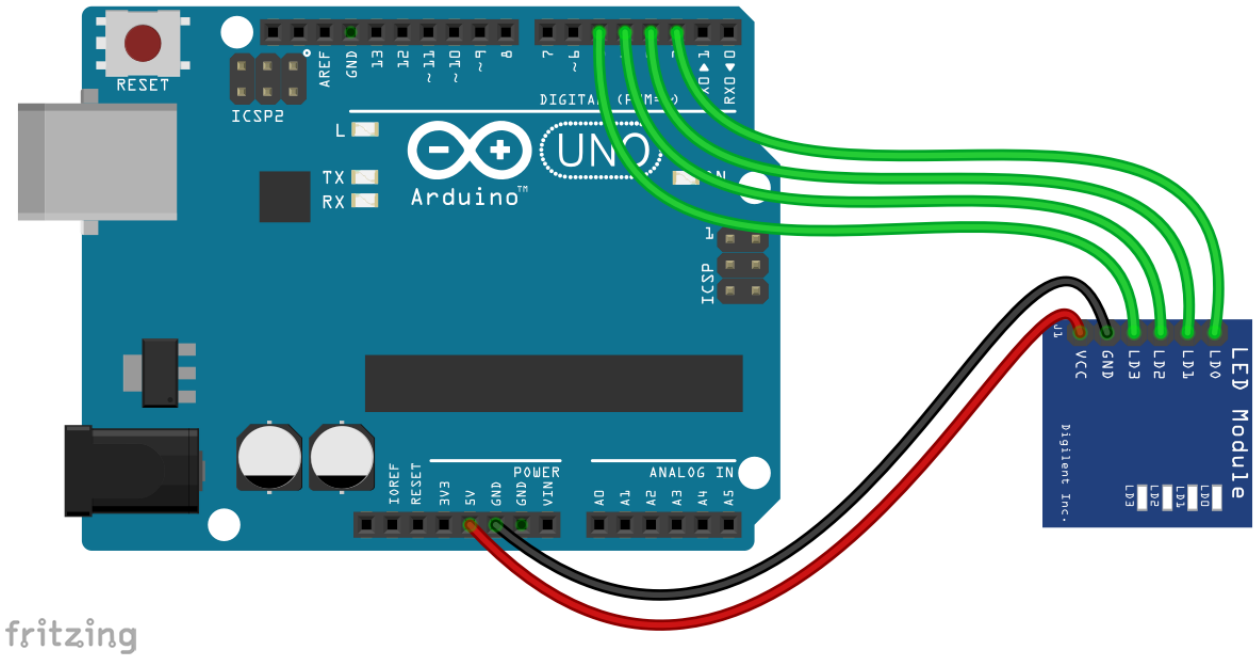


Schéma de câblage :



PMOD8LD : Module 8 led

Programme Arduino :

```
/************************************************************************
*
*    Test du module Pmod 8 led
*
************************************************************************
* Description:  Pmod_8LD
* Les led s'allument et s'éteignent comme dans un célèbre série télévisée des années 2000
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod 8LD
*
************************************************************************/

void setup()
{
 for (int i=2; i<=9; i++)              // configuration des broches 2 à 9 en sortie
 {
   pinMode(i,OUTPUT);
 }
}

void loop()
{
 for (int i=1; i <=4; i++)            // les led s'allument de l'extérieur vers l'intérieur du module
  {
    digitalWrite(i+1,HIGH);
    digitalWrite(10-i,HIGH);
    delay(300);
  }
 delay(600);
 for (int i=1; i <=4; i++)            // les led s'éteignent de l'intérieur vers l'extérieur du module
  {
    digitalWrite(6-i,LOW);
    digitalWrite(5+i,LOW);
    delay(300);
  }
 delay(600);
}
```

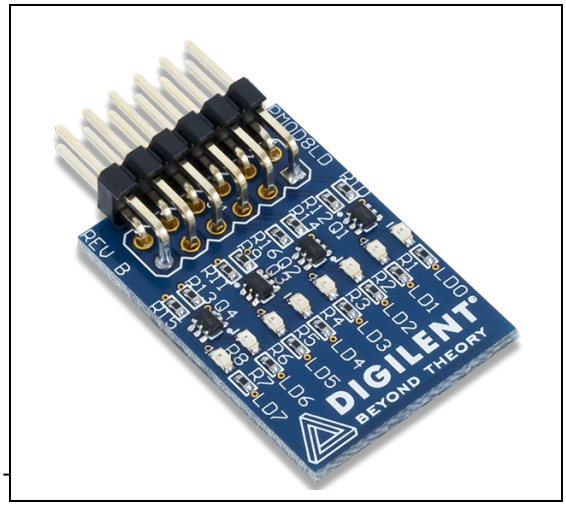
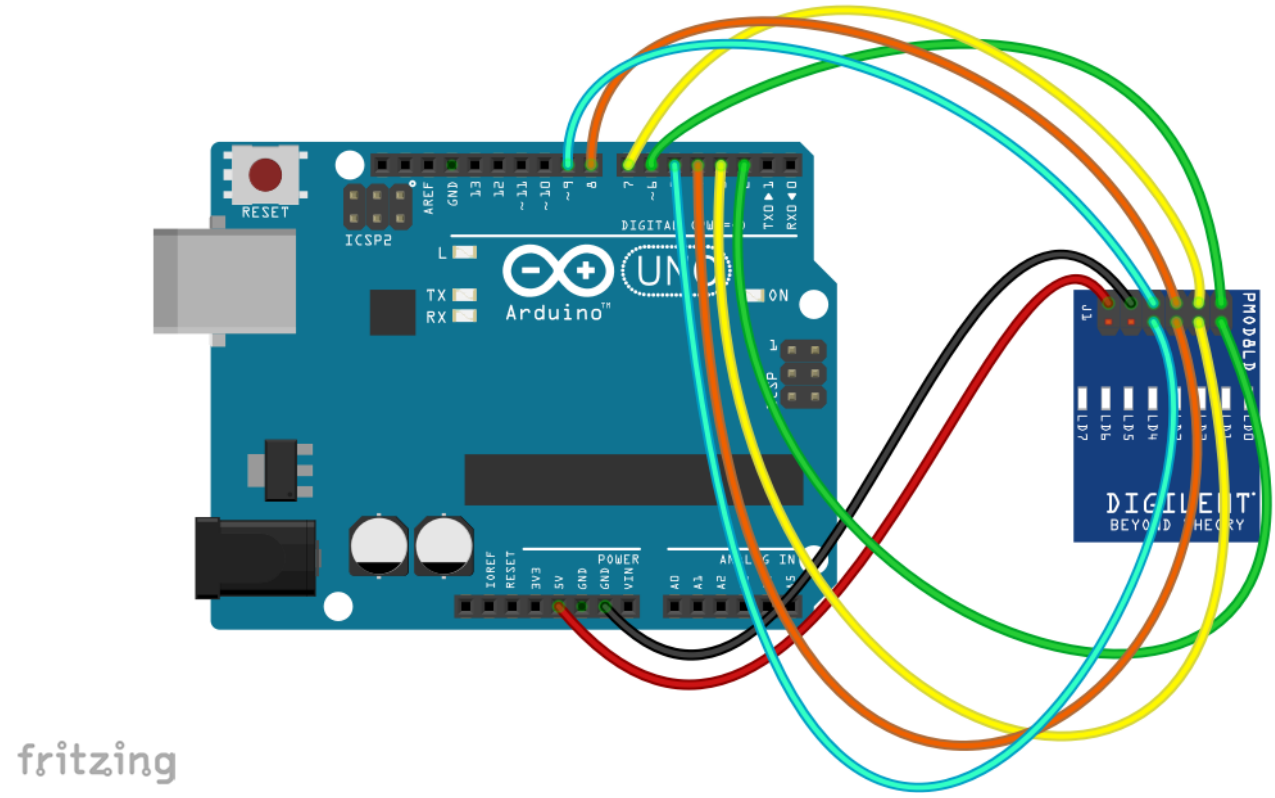
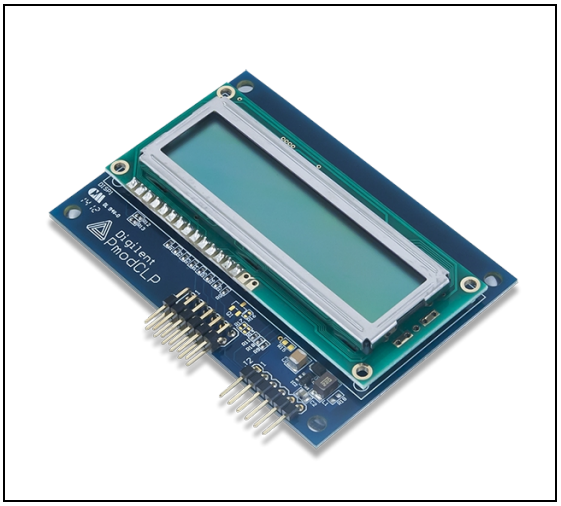


Schéma de câblage :



Programme Arduino :

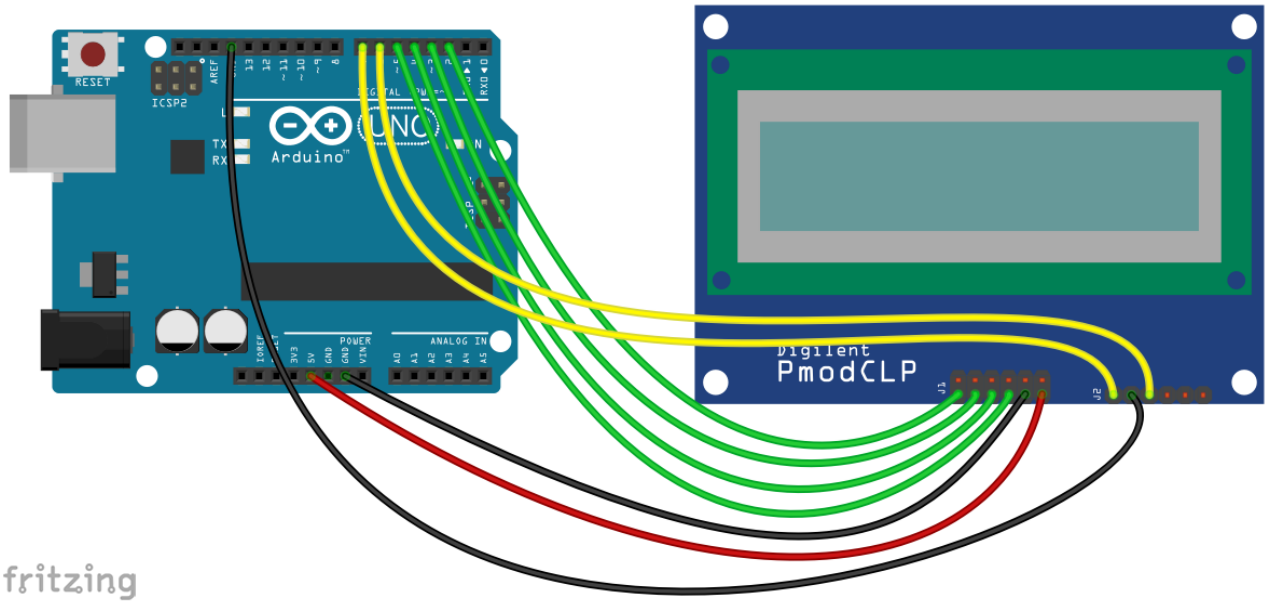
```
/************************************************************************
*
*    Test du module Pmod afficheur LCD interface parallèle
*
************************************************************************
* Description:  Pmod_CLP
* Le message "Test module Pmod Digilent partenaire de Lextronic" est affiché sur l'afficheur.
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod CLP
*
************************************************/
// Affectation des broches
#define en 7
#define rs 6
#define d7 5
#define d6 4
#define d5 3
#define d4 2
```



```
#include<LiquidCrystal.h>                  // appel de la bibliothèque LiquidCrystal
LiquidCrystallcd(rs, en, d4, d5, d6, d7);  // création de l'objet lcd

voidsetup()
{
lcd.begin(16,2);                           // initialisation de l'objet lcd
}

voidloop()
{
lcd.clear();                               // effacement de l'écran
lcd.print("Test module Pmod");
lcd.setCursor(4,1);                        // placement du curseur 2nde ligne 5ème colonne
lcd.print("Digilent");
delay(3000);
lcd.clear();
lcd.setCursor(1,0);                        // placement du curseur 1ère ligne 2nde colonne
lcd.print("partenaire de");
lcd.setCursor(3,1);                        // placement du curseur 2nde ligne 4ème colonne
lcd.print("Lextronic");
delay(3000);
```

}

Schéma de câblage :



Programme Arduino :

```
/************************************************************************
*
*    Test du module Pmod afficheur LCD interface série
*
************************************************************************
* Description:  Pmod_CLS
* Le message envoyé depuis le moniteur série est affiché sur l'afficheur.
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod CLS (position des cavaliers sur MOD0 et MOD2)
* voir liste des instructions https://reference.digilentinc.com/pmod/pmod/cls/user_guide
*
************************************************************************/

//Déclaration d'un port série
#include <SoftwareSerial.h>
SoftwareSerial lcd(2,3);            // RX, TX

char machaine[40];
int i=0;

void setup()
{
 Serial.begin(9600);                // initialisation de la liaison série du moniteur
 lcd.begin(9600);                   // initialisation de la liaison série de l'afficheur
 lcd.write("\x1b[j");               // effacement de l'afficheur
 lcd.write("\x1b[0h");              // configuration de l'afficheur en mode écriture du message sur deux lignes
 lcd.write("\x1b[0;5H");            // placement du curseur 1ère ligne 5ème colonne
 lcd.print("Entrer");
 lcd.write("\x1b[1;1H");            // placement du curseur 2nde ligne 1ère colonne
 lcd.print("votre message");
 delay(2000);
 lcd.write("\x1b[j");               // effacement de l'afficheur
 lcd.write("\x1b[0;1H");            // placement du curseur 1ère ligne 1ère colonne
 lcd.print("sur le moniteur");
 lcd.write("\x1b[1;5H");            // placement du curseur 2nde ligne 5ème colonne
 lcd.print("serie");
}

void loop()
{
```

```
  while (Serial.available())          // tant que des caractères sont présents sur la liaison série
      {
      machaine[i]=Serial.read();      // stockage des caractères dans le tableau machaine
      Serial.print(machaine[i]);      // écriture des caractères dans le moniteur série
      delay(10);
      if (i==0)                       // si premier caractère
        {
        lcd.write("\x1b[j");          // effacement de l'afficheur
        lcd.print(machaine[i]);       // envoi du caractère sur l'afficheur
        }
      else
        {
        lcd.print(machaine[i]);
        }
      i++;
      }
i=0;                  // réinitialisation pour le prochain message
}
```

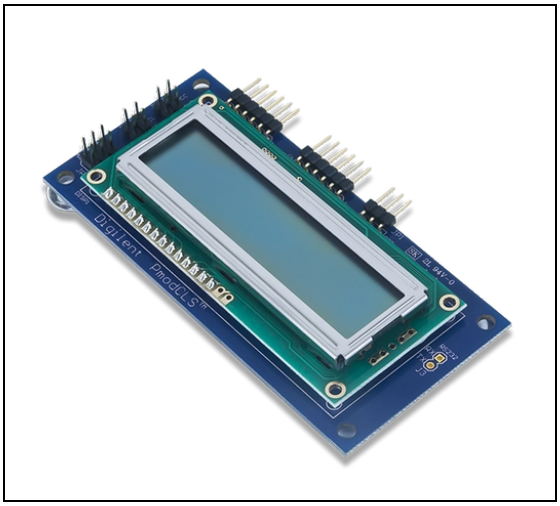
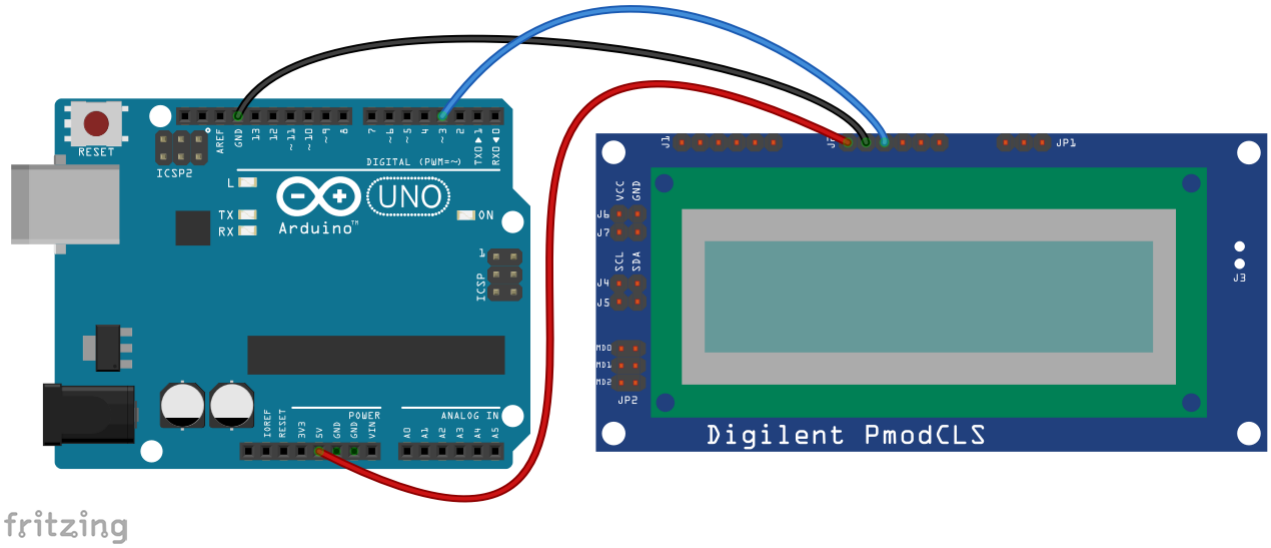


Schéma de câblage :



Programme Arduino :

```
/************************************************************************
*
*   Test du module Pmod joystick
*
************************************************************************
* Description:  Pmod_JSTK
* Les valeurs X et Y sont affichées dans le moniteur série sous forme de
* tableau et les led LD1 et LD2 du module s'allument lorsque les boutons
* BTN1 et BTN2 sont actifs.
*
*
* Matériel
*      1. Arduino Uno
*      2. Module Pmod JSTK
*
************************************************************************/

#define CS 10              // affectation de la broche CS
#include <SPI.h>           // appel de la bibliothèque
int i;
byte recu[6];              // stockage des données du module
int X;
int Y;
int led=128;
void setup()
{
 Serial.begin(9600);                          // initialisation de la liaison série
 SPI.begin();                                 // initialisation du port SPI
 SPI.setDataMode(SPI_MODE0);                  // configuration de la liaison SPI en mode 0
 SPI.setClockDivider(SPI_CLOCK_DIV16);        // configuration de l'horloge à 1MHz
 pinMode(10, OUTPUT);
}

void loop()
{
 digitalWrite(CS, LOW);             // activation de la ligne CS
 delayMicroseconds(15);             // voir doc: pause de 15us après l'activation de la ligne CS
 for (i=0;i<5;i=i+1)
    {
    recu[i] = SPI.transfer(led);    // envoi de 5 données pour récupérer les données du module, les led sont éteintes
    delayMicroseconds(10);          // voir doc: pause de 10us après chaque envoi
    }
```

```
 digitalWrite(CS, HIGH);            // désactivation de la ligne CS
 X = recu[0];                       // X a un format de 10 bit
 X |= (recu[1] << 8);
 Y = recu[2];                       // Y a un format de 10 bit
 Y |= (recu[3] << 8);
 for (i=0;i<5;i=i+1)                // écriture dans le moniteur série
     {
     Serial.print("i");
     Serial.print(i);
     Serial.print("=");
     Serial.print(recu[i]);
     Serial.print('\t');            // tabulation
     }
 Serial.print("X=");
 Serial.print(X);
 Serial.print('\t');                // tabulation
 Serial.print("Y=");
 Serial.println(Y);
 delay(10);
 switch (recu[4])
 {
 case 2:                            // BTN1 actif
   led=129;
   break;
 case 4:                            // BTN2 actif
   led=130;
   break;
 case 6:                            // BTN1 et BTN2 actifs
   led=131;
   break;
 default:
   led=128;
 break;
 }
}
```

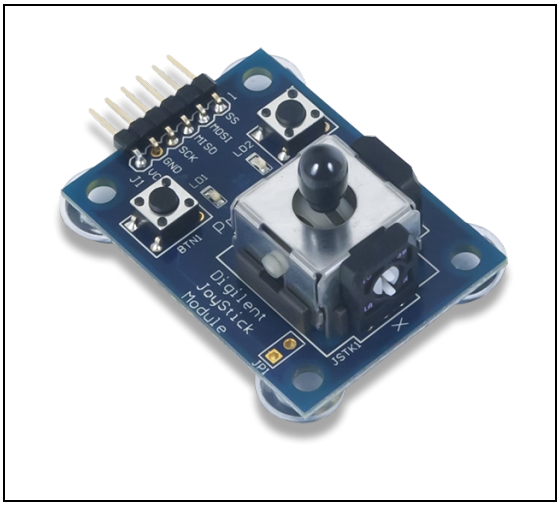
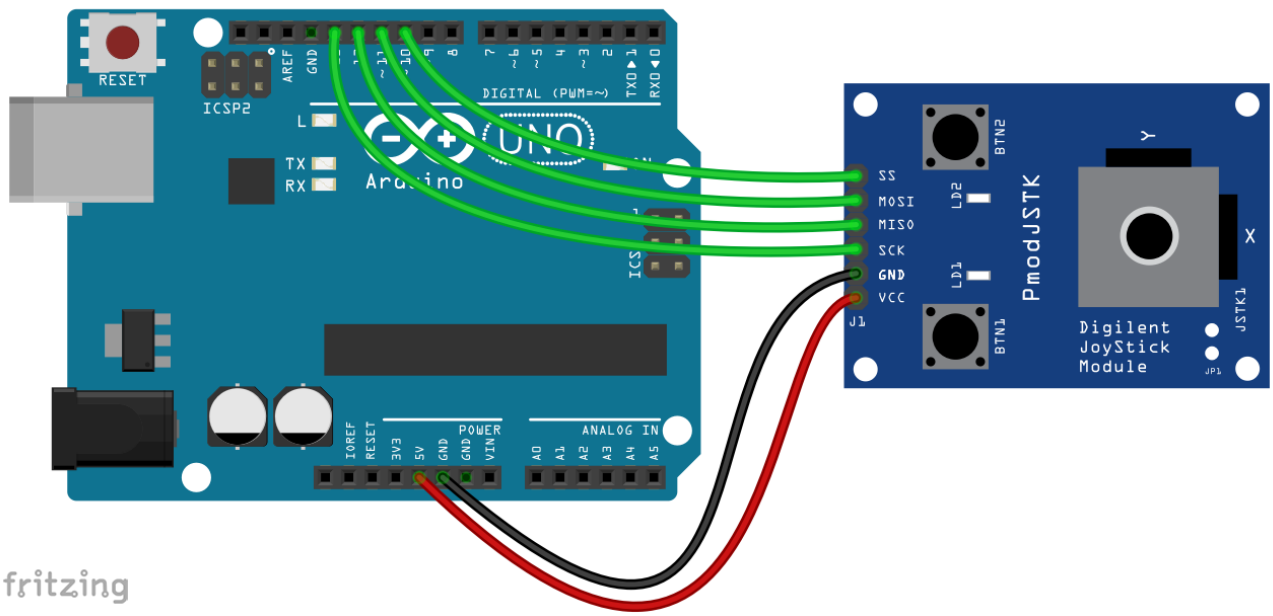


Schéma de câblage :



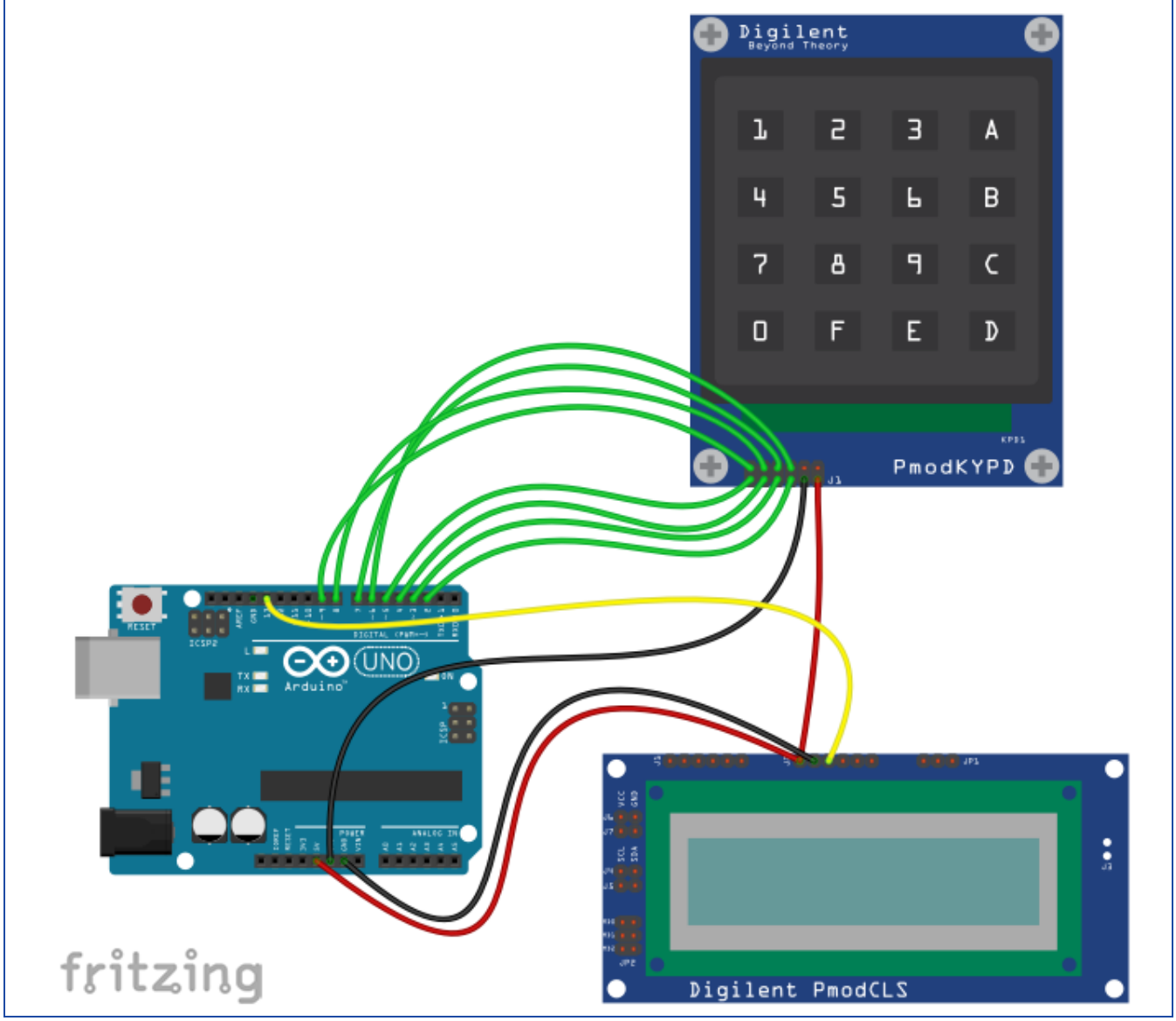
PMODKYPD : Module clavier 16 touches

Programme Arduino :

```
/************************************************************************
*
*    Test du module Pmod clavier 16 touches
*
************************************************************************
* Description:  Pmod_KYPD
* L'activation d'une touche du clavier est affichée sur un afficheur LCD Série.
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod KYPD (télécharger la bibliothèque http://playground.arduino.cc/Code/Keypad)
*       3. Module Pmod CLS (voir liste des instructions
https://reference.digilentinc.com/pmod/pmod/cls/user_guide)
*
************************************************************************/

//Déclaration d'un port série
#include <SoftwareSerial.h>
SoftwareSerial lcd(12,13);      // RX, TX

#include <Keypad.h>
const byte LIGNE = 4;           // 4 lignes
const byte COLONNE = 4;     // 4 colonnes
char touche;

//Déclaration des touches du clavier
char hexaKeys[LIGNE][COLONNE] =
{
 {'1','2','3','A'},
 {'4','5','6','B'},
 {'7','8','9','C'},
 {'0','F','E','D'}
};
//Affectation des broches du clavier
byte ligne_pin[LIGNE] = {2, 3, 4, 5};
byte colonne_pin[COLONNE] = {6, 7, 8, 9};
Keypad clavier = Keypad( makeKeymap(hexaKeys), ligne_pin, colonne_pin, LIGNE, COLONNE);  // création de l'objet
clavier

void setup()
{
 lcd.begin(9600);                    // initialisation de la liaison série de l'afficheur
```

```
 lcd.write("\x1b[j");                  // effacement de l'afficheur
 lcd.write("\x1b[0h");                 // configuration de l'afficheur en mode écriture du message sur deux lignes
}

void loop()
{
 lcd.write("\x1b[j");                 // effacement de l'afficheur
 lcd.write("\x1b[0;4H");              // positionnement du curseur 1ère ligne 4ème colonne
 lcd.print("Appuyer");
 lcd.write("\x1b[1;1H");              // positionnement du curseur 2nde ligne 1ère colonne
 lcd.print("sur une touche");
 delay(100);
 touche=clavier.getKey();             // acquisition de la touche
 if (touche!=0x00)                    // si aucune touche est active, la fonction getKey renvoie le caractère NULL (0x00)
 {
 lcd.write("\x1b[j");                 // effacement de l'afficheur
 lcd.print("Touche:");
 lcd.print(touche);                   // affichage de la touche
 delay(1000);
 }
}
```

Schéma de câblage :



PMODSSD : Module double afficheur

Programme Arduino :

```
/*************************************************************************
*
*   Test du module Pmod double afficheur
*
*************************************************************************
* Description:  Pmod_SSD
* Un compteur s'incrémente toutes les secondes.
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod SSD
*
*************************************************************************/

byte ledPin[8]={2, 3, 4, 5, 6, 7, 8, 9};                    // numéros de broches de l'Arduino
byte code[10]={63,6,91,79,102,109,124,7,127,103};           // code des chiffres 0 à 9
intunite=0;
int dizaine=0;
intduree;

voidsetup()
{
for (int i=0; i<=8; i++)                    // configuration des broches 2 à 9 en sortie
  {
pinMode(ledPin[i], OUTPUT);
  }
}

// Programme principal
voidloop()
{
duree=millis() / 1000;                  // chronomètre
dizaine=duree/10;                       // extraction des dizaines
unite=duree%10;                         // extraction des unités
if (duree>=100)                         // remise à 0 lorsque le compteur arrive à 99
 {
dizaine=0;
unite=0;
 }
digitalWrite(9,LOW);                    // sélection de l'afficheur des unités
afficher(code[unite]);                  // affichage des unités
delay(10);
```

```
digitalWrite(9,HIGH);              // sélection de l'afficheur des dizaines
afficher(code[dizaine]);           // affichage des dizaines
delay(10);
}

voidafficher(int x)                // procédure codant le chiffre en 7 segments
{
byte chiffre=x;
byte segment=0;
for (int i=2; i<9; i++)
  {
segment=chiffre&00000001;
digitalWrite(i,segment);
chiffre=chiffre>>1;
  }
}
```

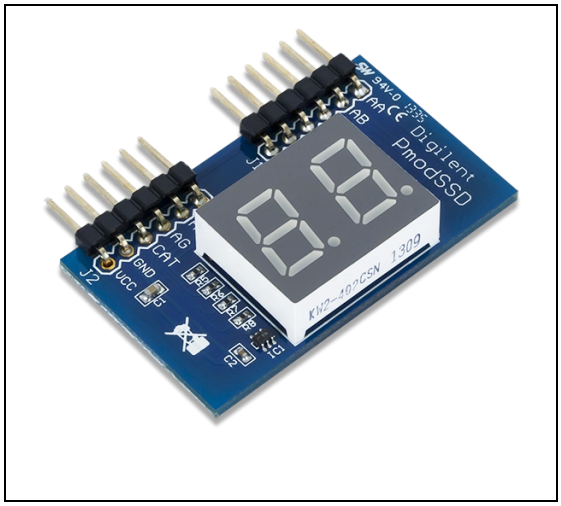
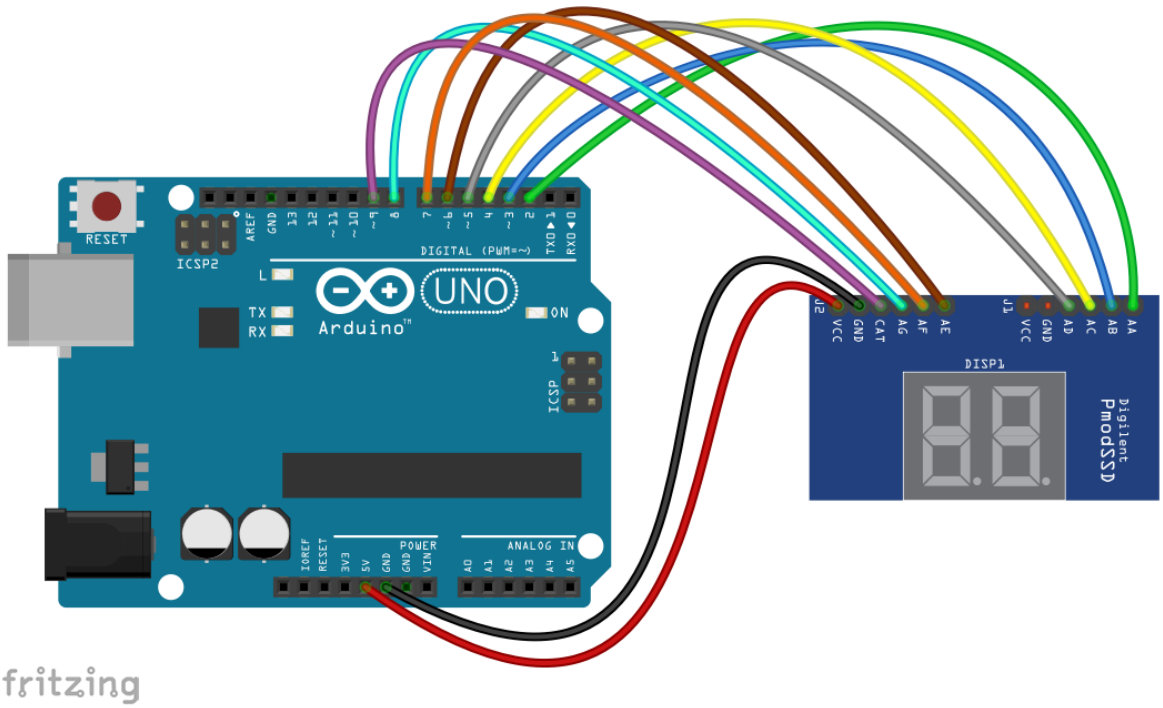


Schéma de câblage :



Programme Arduino :

```
/************************************************************************
*
*   Test du module Pmod convertisseur N/A 8 bit 4 sorties
*
************************************************************************
* Description:  Pmod_DA1
* Les sorties A1 et B1 délivrent des signaux carrés en opposition de phase
* avec une période programmable.
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod DA1
*
************************************************************************/

#define CS 10                                   // affectation de la broche CS
#include <SPI.h>                                // appel de la bibliothèque

int periode=10;
void setup()
{
 SPI.begin();                                   // initialisation du port SPI
 SPI.setDataMode(SPI_MODE0);                    // configuration de la liaison SPI en mode 0
 SPI.setClockDivider(SPI_CLOCK_DIV16);          // configuration de l'horloge à 1MHz
 pinMode(CS, OUTPUT);
}
void loop()
{
   digitalWrite(CS, LOW);                       // activation de la ligne CS
   SPI.transfer(16);                            // envoi des bit de commande (sortie A active et sortie B inactive)
   SPI.transfer(255);                           // envoi des bit de données
   digitalWrite(CS, HIGH);                      // désactivation de la ligne CS
   delay(periode/2);                            // pause
   digitalWrite(CS, LOW);                       // activation de la ligne CS
   SPI.transfer(12);                            // envoi des bit de commande (sortie A inactive et sortie B active)
   SPI.transfer(255);                           // envoi des bit de données
   digitalWrite(CS, HIGH);                      // désactivation de la ligne CS
   delay(periode/2);                            // pause
}
```

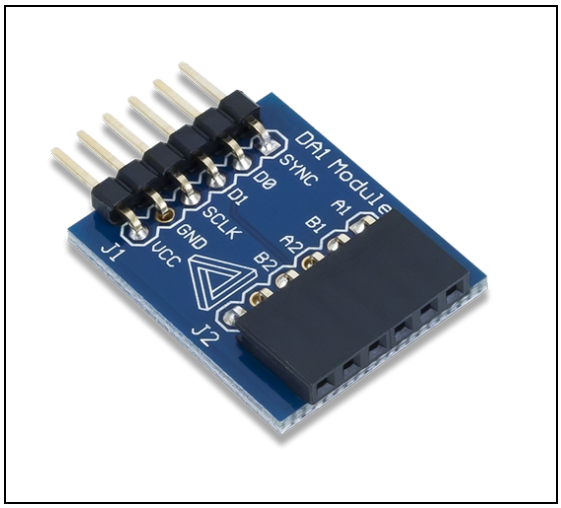
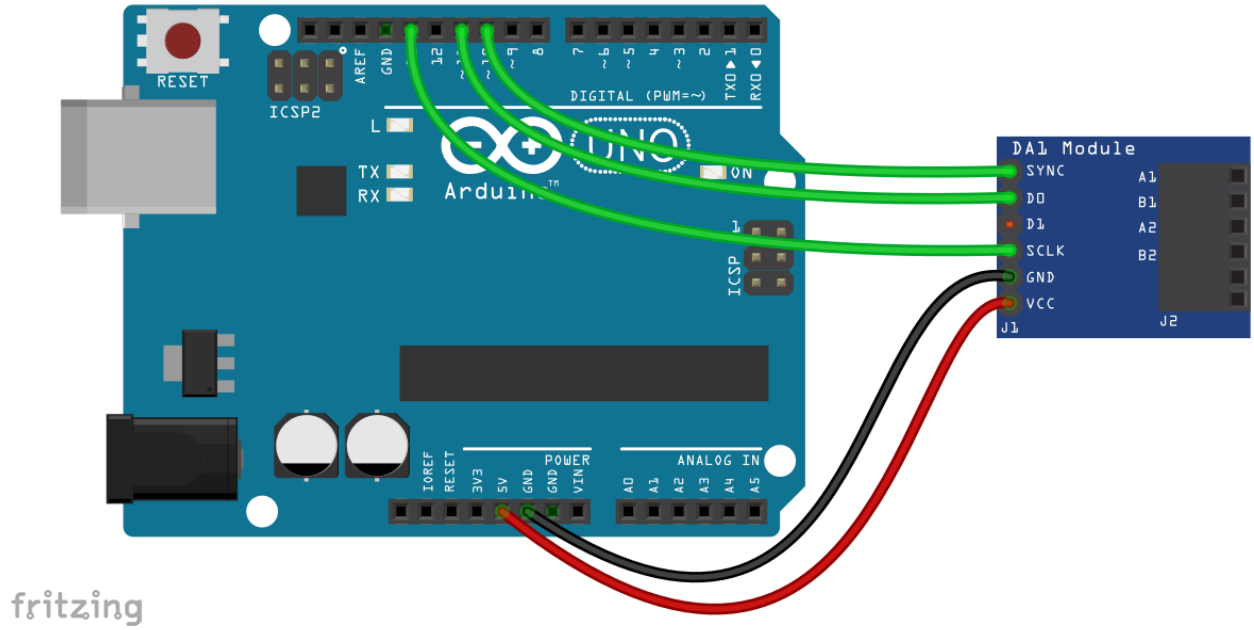
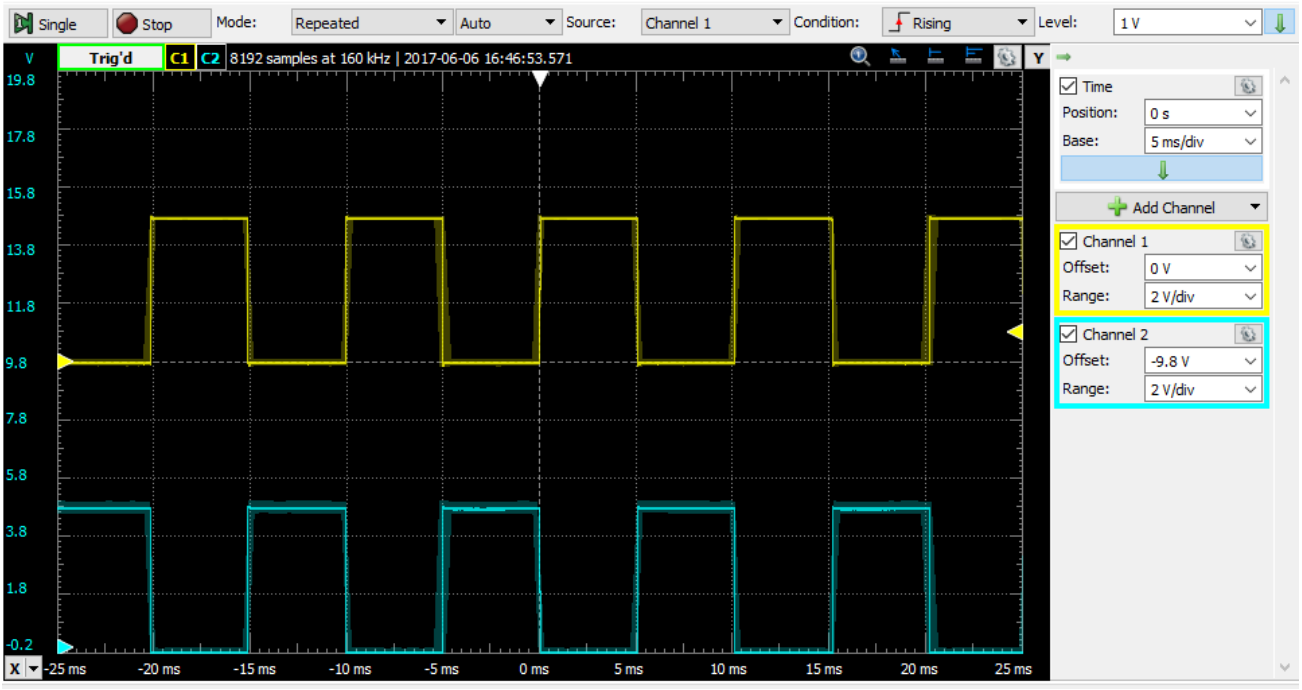


Schéma de câblage :



Visualisation des sorties du module à l'aide d'un boitier de mesure “Analog Discovery 2” de Digilent
https://www.lextronic.fr/P37273-botier-analog-discovery-2.html

Oscillogramme :



Programme Arduino :

```
/************************************************************************
*
*   Test du module Pmod convertisseur N/A 12 bit 2 sorties
*
************************************************************************
* Description:  Pmod_DA2
* La tension de sortie du module est choisie par l'utilisateur depuis le
* moniteur série
*
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod DA2
*
************************************************************************/

#define CS 10    // affectation de la broche CS
#include <SPI.h> // appel de la bibliothèque
char tableau[4] = {0,0,0,0};
int i = 0;
long valeur;
long consigne;
int consigne_basse;
int consigne_haute;
int val=0;
float tension;

void setup()
{
 Serial.begin(115200);                    // initialisation de la liaison série
 SPI.begin();                             // initialisation du port SPI
 SPI.setDataMode(SPI_MODE3);              // configuration de la liaison SPI en mode 3
 SPI.setClockDivider(SPI_CLOCK_DIV16);    // configuration de l'horloge à 1MHz
 pinMode(CS, OUTPUT);
}

void loop()
{
 Serial.println("Entrer la tension en mV sur 4 chiffres puis Envoyer");
 while (Serial.available()==0);           // attente de données
 for(i = 0;i<4;i++)                       // boucle pour écrire les données dans un tableau
   {
```

```
  tableau[i]=Serial.read();
  delay(1);
  }
 delay(100);
 // recomposition du nombre tension
 valeur=1000*(tableau[0]-48)+100*(tableau[1]-48)+10*(tableau[2]-48)+(tableau[3]-48);
 consigne=4095*valeur;
 consigne=consigne/5000;
 consigne_haute=consigne>>8;                  // consigne_haute récupère les 4 bit de poids fort de consigne
 consigne_basse=consigne&0XFF;                // consigne_basse récupère les 8 bit de poids faible de consigne
 digitalWrite(CS, LOW);                       // activation de la ligne CS
 SPI.transfer(consigne_haute);                // envoi de la consigne
 SPI.transfer(consigne_basse);
 delay(5);
 digitalWrite(CS, HIGH);                      // désactivation de la ligne CS
 delay(100);
 Serial.print("La tension de consigne est : ");
 Serial.print(valeur);
 Serial.println(" mV");
}
```

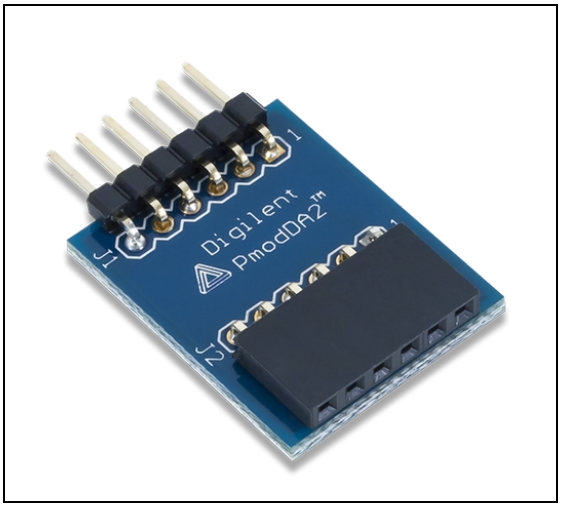
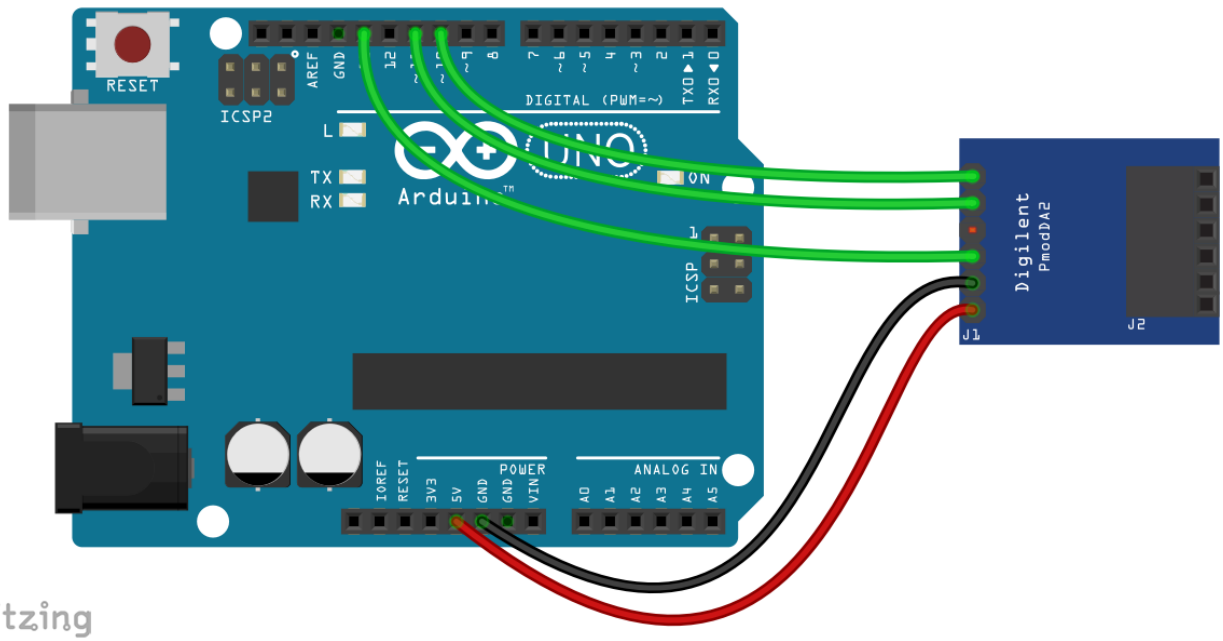


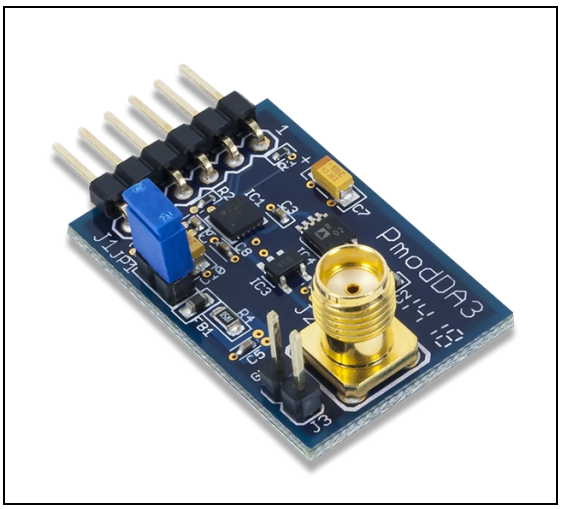
Schéma de câblage :



PMODDA3 : Module convertisseur « numérique/analogique » 16 bit

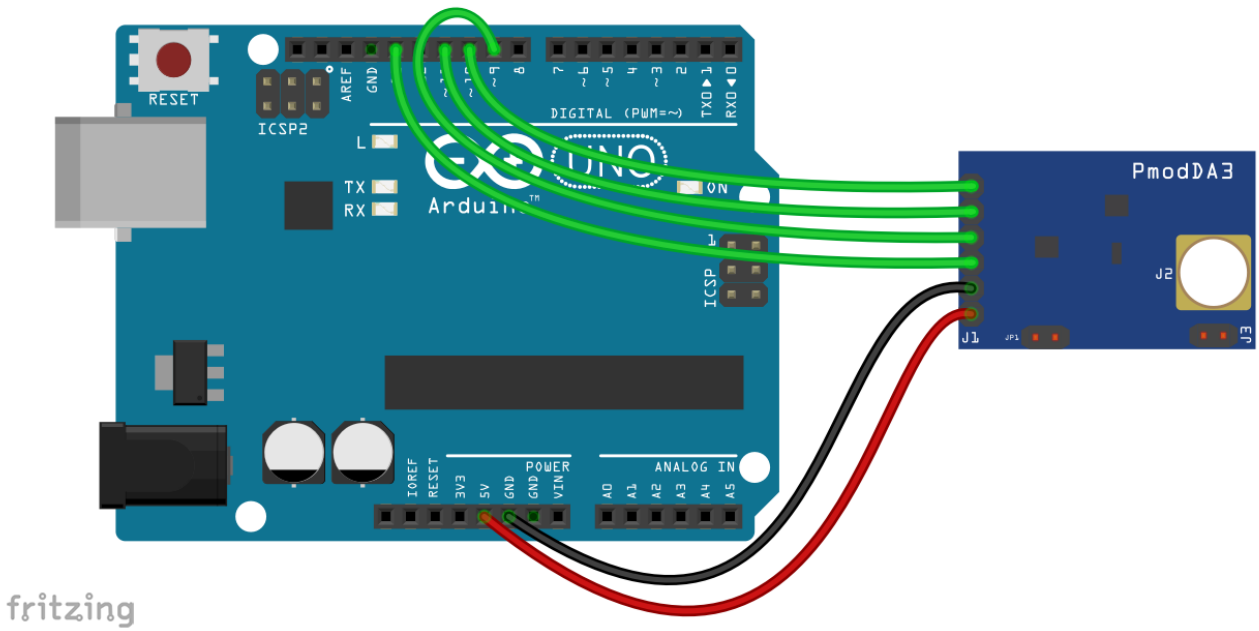
Programme Arduino :

```
/************************************************************************
*
*   Test du module Pmod convertisseur N/A 16 bit
*
************************************************************************
* Description:  Pmod_DA3
* La tension de sortie du module évolue de 0 à 2,5 V pour générer un signal
* en dent de scie.
*
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod DA3 (laisser le cavalier JP1 en place)
*
************************************************************************/

#define CS 10                                   // affectation de la broche CS
#define LDAC 9                                  // affectation de la broche LDAC
#include <SPI.h>                                // appel de la bibliothèque

int i;
int j;

void setup()
{
 Serial.begin(9600);                            //initialisation de la liaison série
 SPI.begin();                                   // initialisation du port SPI
 SPI.setDataMode(SPI_MODE0);                    // configuration de la liaison SPI en mode 0
 SPI.setClockDivider(SPI_CLOCK_DIV16);          // configuration de l'horloge à 1MHz
 pinMode(CS, OUTPUT);
 pinMode(LDAC, OUTPUT);
}

void loop()
{
 for (i=0;i<256;i=i+1)
   {
    for (j=0;j<256;j=j+1)
    {
    digitalWrite(LDAC, HIGH); // désactivation de la ligne LDAC
    digitalWrite(CS, LOW);      // activation de la ligne CS
    SPI.transfer(i);            // envoi des bit de poids fort
```



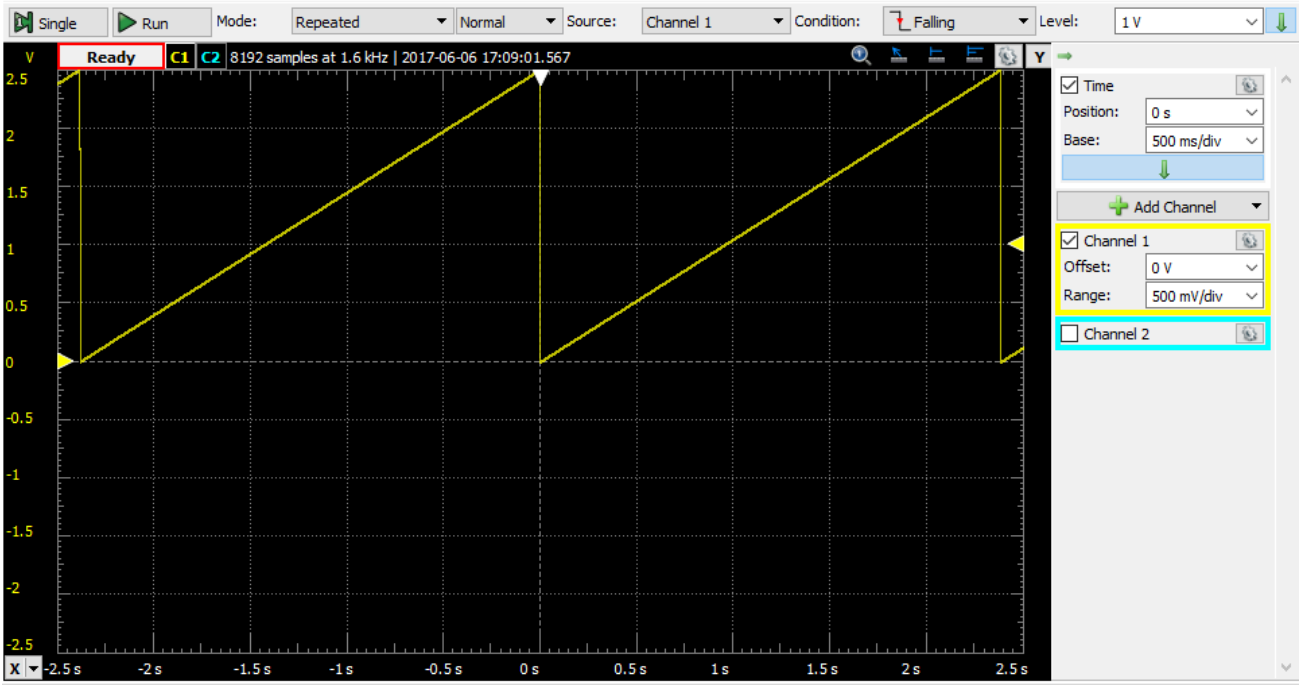
```
  SPI.transfer(j);                // envoi des bit de poids faible
  digitalWrite(CS, HIGH);                 // désactivation de la ligne CS
  digitalWrite(LDAC, LOW);                // activation de la ligne LDAC
  /* la période du signal est de 2,4 s environ.
   * Si on souhaite l'augmenter, on introduit à chaque passage dans la boucle un retard
   * delay(1);  ou delayMicroseconds(50);
   */
  }
}
```

Schéma de câblage :



Visualisation des sorties du module à l'aide d'un boitier de mesure “Analog Discovery 2” de Digilent
https://www.lextronic.fr/P37273-botier-analog-discovery-2.html

Oscillogramme :



Programme Arduino :

```
/************************************************************************
*
*   Test du module Pmod convertisseur N/A 12 bit 8 sorties
*
************************************************************************
* Description:  Pmod_DA4
* La tension des sorties A à E vont de 2,5V à 0,5V par pas de 0,5V
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod DA4
*
************************************************************************/

#define CS 10                           // affectation de la broche CS

#include <SPI.h>                        // appel de la bibliothèque

void setup()
{
 SPI.begin();                           // initialisation du port SPI
 SPI.setDataMode(SPI_MODE0);            // configuration de la liaison SPI en mode 0
 SPI.setClockDivider(SPI_CLOCK_DIV16);  // configuration de l'horloge à 1MHz
 pinMode(CS, OUTPUT);
 digitalWrite(CS, LOW);                 // activation de la ligne CS
 SPI.transfer(0b00001000);              // configuration du convertisseur N/A (configuration du registre REF)
 delay(1);
 SPI.transfer(0);
 delay(1);
 SPI.transfer(0);
 delay(1);
 SPI.transfer(0b00000001);              // configuration du convertisseur N/A (tension de référence interne
active VREF=1,25V)
 delay(1);
 digitalWrite(CS, HIGH);                // désactivation de la ligne CS
 digitalWrite(CS, LOW);                 // activation de la ligne CS
 SPI.transfer(0b00000011);              // configuration du convertisseur N/A (écriture dans les voies du
convertisseur)
 delay(1);
 SPI.transfer(0b11110000);              // configuration du convertisseur N/A (les 8 voies du convertisseur sont
actives)
 delay(1);
```

```
 SPI.transfer(0);
 delay(1);
 SPI.transfer(0);
 delay(1);
 digitalWrite(CS, HIGH);                    // désactivation de la ligne CS
}

void loop()
{
 // la sortie A du convertisseur est à 2,5V
 digitalWrite(CS, LOW);                     // activation de la ligne CS
 SPI.transfer(0b00000011);
 delay(1);
 SPI.transfer(0b00001111);
 delay(1);
 SPI.transfer(0b11111111);
 delay(1);
 SPI.transfer(0);
 delay(1);
 digitalWrite(CS, HIGH);                    // désactivation de la ligne CS
 // la sortie B du convertisseur est à 2V
 digitalWrite(CS, LOW);                     // activation de la ligne CS
 SPI.transfer(0b00000011);
 delay(1);
 SPI.transfer(0b00011100);
 delay(1);
 SPI.transfer(0b11001101);
 delay(1);
 SPI.transfer(0);
 delay(1);
 digitalWrite(CS, HIGH);                    // désactivation de la ligne CS
 // la sortie C du convertisseur est à 1,5V
 digitalWrite(CS, LOW);                     // activation de la ligne CS
 SPI.transfer(0b00000011);
 delay(1);
 SPI.transfer(0b00101001);
 delay(1);
 SPI.transfer(0b10011010);
 delay(1);
 SPI.transfer(0);
 delay(1);
 digitalWrite(CS, HIGH);                    // désactivation de la ligne CS
 // la sortie D du convertisseur est à 1V
 digitalWrite(CS, LOW);                     // activation de la ligne CS
 SPI.transfer(0b00000011);
 delay(1);
 SPI.transfer(0b00110110);
 delay(1);
 SPI.transfer(0b01100110);
 delay(1);
```

```
SPI.transfer(0);
delay(1);
digitalWrite(CS, HIGH);                    // désactivation de la ligne CS

// la sortie E du convertisseur est à 0,5V
digitalWrite(CS, LOW);                     // activation de la ligne CS
SPI.transfer(0b00000011);
delay(1);
SPI.transfer(0b01000011);
delay(1);
SPI.transfer(0b00110011);
delay(1);
SPI.transfer(0);
delay(1);
digitalWrite(CS, HIGH);                    // désactivation de la ligne CS
}
```

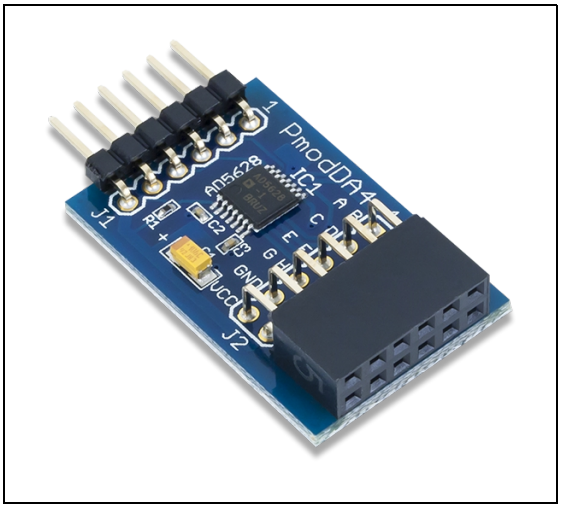
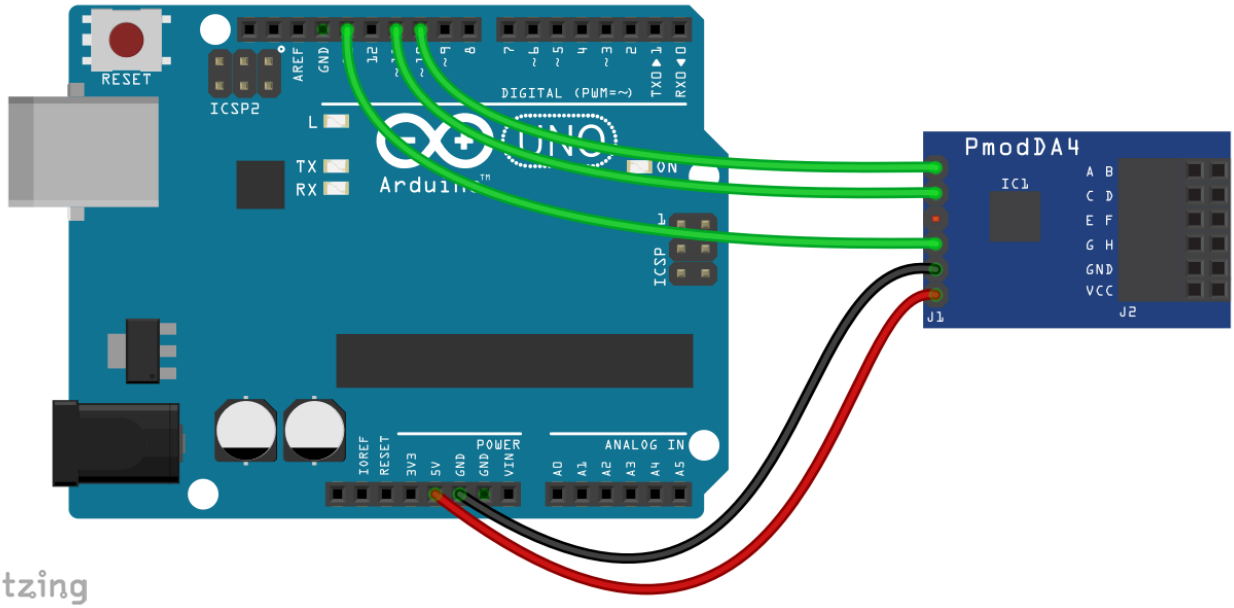


Schéma de câblage :



PMODPOT : Module potentiomètre digital

Programme Arduino :

```
/************************************************************************
*
*   Test du module Pmod potentiomètre digital
*
************************************************************************
* Description:  Pmod_DPOT
* La tension de sortie du module évolue de 0 à 5 V toutes les secondes.
*
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod DPOT
*
************************************************************************/

#define CS 10                                   // affectation de la broche CS

#include <SPI.h>                                // appel de la bibliothèque

int i;
int val=0;
float tension;

void setup()
{
 Serial.begin(9600);                            // initialisation de la liaison série
 SPI.begin();                                   // initialisation du port SPI
 SPI.setDataMode(SPI_MODE0);                    // configuration de la liaison SPI en mode 0
 SPI.setClockDivider(SPI_CLOCK_DIV16);          // configuration de l'horloge à 1MHz
 pinMode(CS, OUTPUT);
}

void loop()
{
 for (i=0;i<256;i=i+1)
     {
     digitalWrite(CS, LOW);                     // activation de la ligne CS
     delayMicroseconds(15);
     SPI.transfer(i);                           // envoi de la variable i (i=0->Vout=0V i=255->Vout=Vcc)
     val=analogRead(A0);                        // conversion AN
     tension = map(val, 0, 1023, 0, 5000);      // tension varie de 0 à 5000 pour une variation de val de 0 à 255
     tension=tension/1000;
```

```
        Serial.print("i=");
        Serial.print(i);
        Serial.print('\t');                    // tabulation
        Serial.print("val=");
        Serial.print(val);
        Serial.print('\t');                    // tabulation
        Serial.print("Tension=");
        Serial.print(tension);
        Serial.println("V");
        digitalWrite(CS, HIGH);                // désactivation de la ligne CS
        delay(1000);
        }
}
```

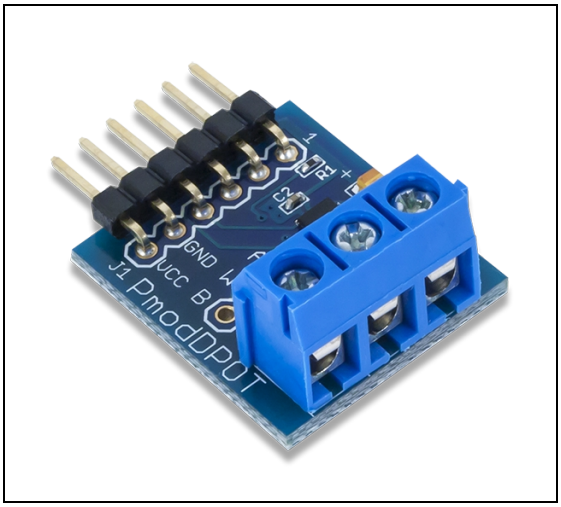
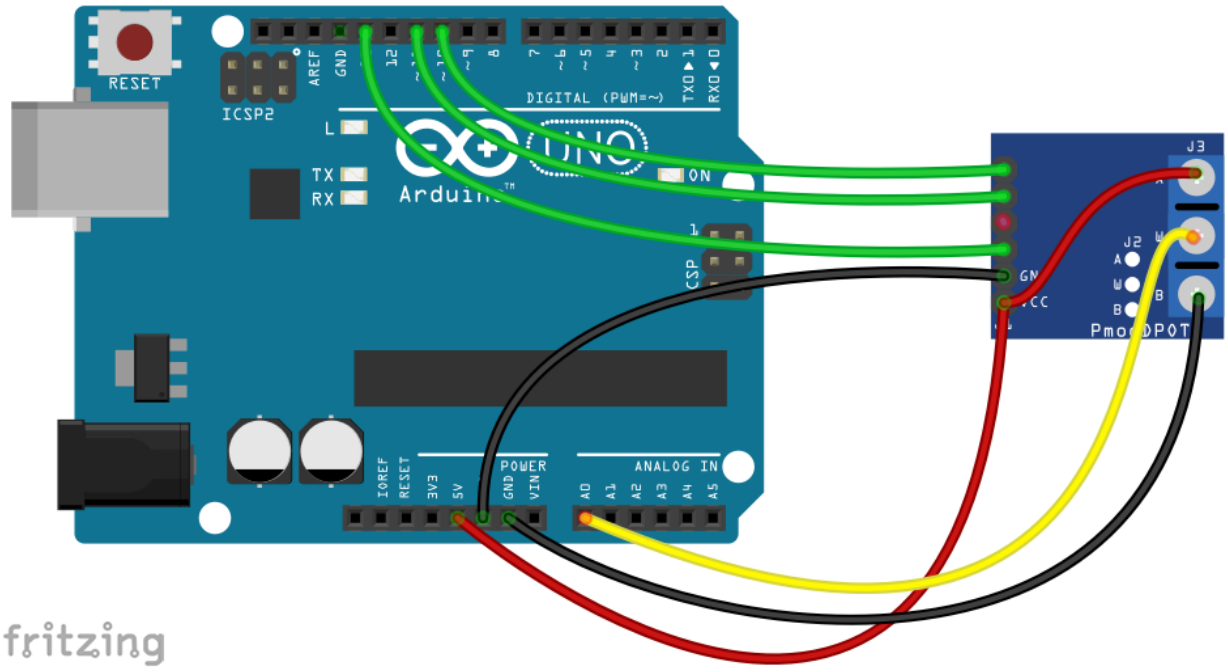


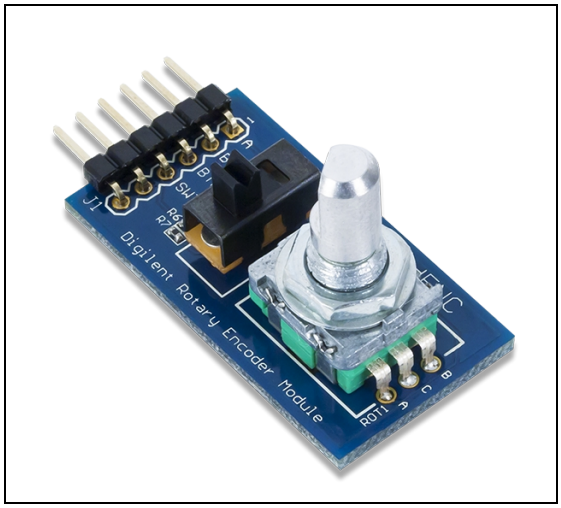
Schéma de câblage :



PMODENC : Module encodeur rotatif

Programme Arduino :

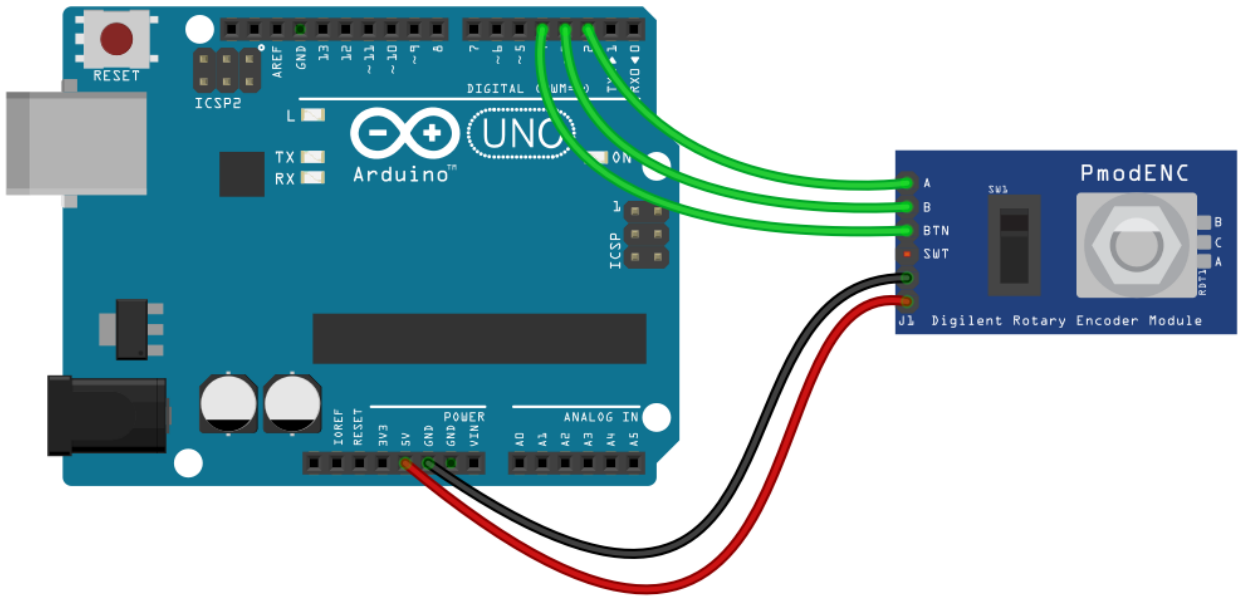
```
/*****************************************************************************
*
*    Test du module Pmod Encodeur rotatif
*
*****************************************************************************
* Description:  Pmod_ENC
* Le sens de rotation de l'axe de l'encodeur est affiché dans le moniteur
* série. Un appui sur le bouton poussoir mémorise l'état d'un compteur dans
* une variable.
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod ENC
*
*****************************************************************************/
// Affectation des broches
#define A 2                     // sortie A de l'encodeur
#define B 3                     // sortie B de l'encodeur
#define BTN 4                   // bouton de l'encodeur

volatile boolean rotation;
volatile boolean sens;
int compteur = 0;
int validation = 0;
```




```
// Programme d'interruption
void Interruption ()
{
 if (digitalRead(A)==HIGH)
   {
   sens = digitalRead(B);             // sens horaire
   }
 else
   {
     sens = !digitalRead(B);          // sens anti-horaire
   }
 rotation = HIGH;
}
void setup()
{
 attachInterrupt (0,Interruption,FALLING);          // interruption front descendant sur la sortie A
```

```
 Serial.begin(9600);                    // initialisation du moniteur série
 pinMode(A,INPUT);                      // configuration de la broche N°2 en entrée
 pinMode(B,INPUT);                      // configuration de la broche N°3 en entrée
 pinMode(BTN,INPUT);                    // configuration de la broche N°4 en entrée
}

void loop()
{
 if (rotation==HIGH)                    // si une rotation a été détectée
 {
  if (sens==HIGH)                       // si sens horaire
    {
    compteur++;                         // compteur s'incrémente
    }
  else
    {
    compteur--;                         // compteur se décrémente
    }
 rotation = LOW;
 Serial.print ("Compteur=");            // écriture de la variable compteur
 Serial.println (compteur);
 }
 if (digitalRead(BTN)==HIGH)            // si le bouton est actif
  {
   validation=compteur;
   delay(200);                          // anti-rebonds
   Serial.print ("Validation=");        // écriture de la variable validation
   Serial.println (validation);
  }
}
```

Schéma de câblage :



Programme Arduino :

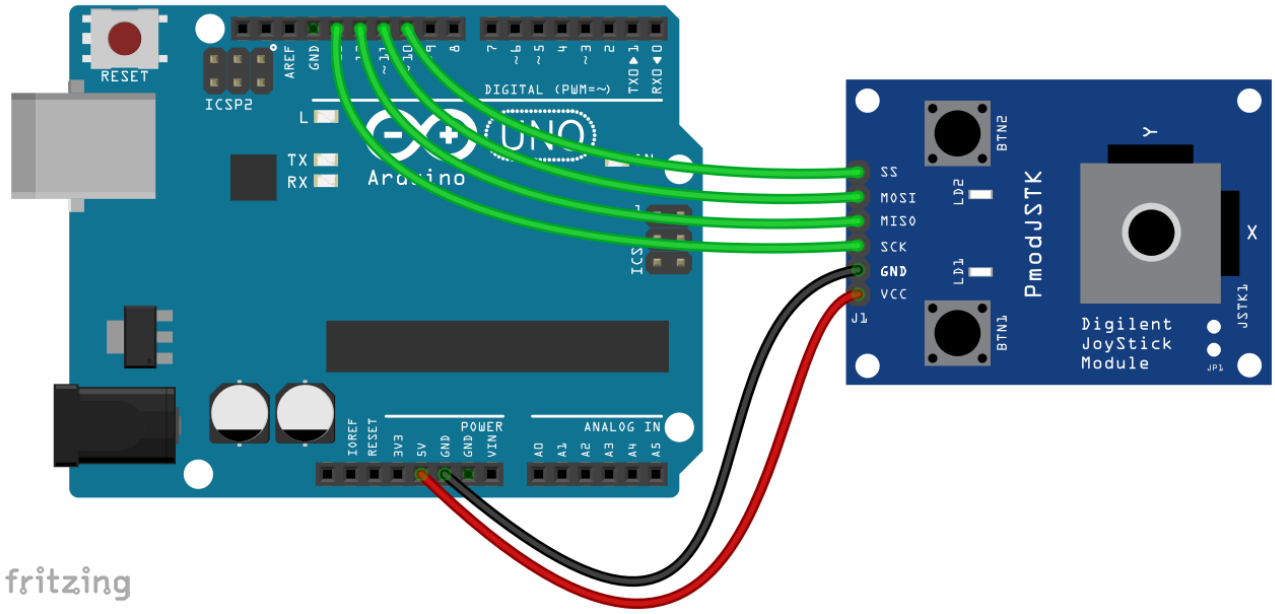
```
/************************************************************************
*
*   Test du module Pmod joystick 2
*
************************************************************************
* Description:  Pmod_JSTK2
* Les valeurs X et Y sont affichées dans le moniteur série sous forme de
* tableau et la led du module s'allume avec une couleur différente lorsque
* les boutons sont actifs.
*
*
* Matériel
*      1. Arduino Uno
*      2. Module Pmod JSTK2
*
************************************************************************/

#define CS 10                                  // affectation de la broche CS

#include <SPI.h>                               // appel de la bibliothèque

int i;
byte recu[7];                                  // stockage des données du module
int X;
int Y;
int cmd=0;
void setup()
{
 Serial.begin(9600);                           // initialisation de la liaison série
 SPI.begin();                                  // initialisation du port SPI
 SPI.setDataMode(SPI_MODE0);                   // configuration de la liaison SPI en mode 0
 SPI.setClockDivider(SPI_CLOCK_DIV16);         // configuration de l'horloge à 1MHz
 pinMode(CS, OUTPUT);
}

void loop()
{
 digitalWrite(CS, LOW);                        // activation de la ligne CS
 delayMicroseconds(15);                        // voir doc: pause de 15us après l'activation de la ligne CS
 for (i=0;i<6;i=i+1)
    {
    recu[i] = SPI.transfer(cmd);               // envoi de 6 données pour récupérer les données du module
```

```
    delayMicroseconds(10);                // voir doc: pause de 10us après chaque envoi
    }
digitalWrite(CS, HIGH);                   // désactivation de la ligne CS
X = recu[0];                              // X a un format de 10 bit
X |= (recu[1] << 8);
Y = recu[2];                              // Y a un format de 10 bit
Y |= (recu[3] << 8);

for (i=0;i<6;i=i+1)                       // écriture dans le moniteur série
    {
    Serial.print("i");
    Serial.print(i);
    Serial.print("=");
    Serial.print(recu[i]);
    Serial.print('\t');                   // tabulation
    }
Serial.print("X=");
Serial.print(X);
Serial.print('\t');                       // tabulation
Serial.print("Y=");
Serial.println(Y);
delay(10);
switch (recu[4])
{
case 1:                                   // bouton joystick actif
 // allumage de la led en rouge
 digitalWrite(CS, LOW);                   // activation de la ligne CS
 SPI.transfer(0x84);
 delayMicroseconds(15);
 SPI.transfer(255);
 delayMicroseconds(15);
 SPI.transfer(0);
 delayMicroseconds(15);
 SPI.transfer(0);
 delayMicroseconds(15);
 digitalWrite(CS, HIGH);                  // désactivation de la ligne CS
 delay(1000);
 // extinction de la led
 digitalWrite(CS, LOW);                   // activation de la ligne CS
 SPI.transfer(0x84);
 delayMicroseconds(15);
 SPI.transfer(0);
 delayMicroseconds(15);
 SPI.transfer(0);
 delayMicroseconds(15);
 SPI.transfer(0);
 delayMicroseconds(15);
 digitalWrite(CS, HIGH);                  // désactivation de la ligne CS
 break;
case 2:                                   // bouton trigger actif
```

```
  // allumage de la led en vert
  digitalWrite(CS, LOW);                  // activation de la ligne CS
  SPI.transfer(0x84);
  delayMicroseconds(15);
  SPI.transfer(0);
  delayMicroseconds(15);
  SPI.transfer(255);
  delayMicroseconds(15);
  SPI.transfer(0);
  delayMicroseconds(15);
  digitalWrite(CS, HIGH);                 // désactivation de la ligne CS
  delay(1000);
  // extinction de la led
  digitalWrite(CS, LOW);                  // activation de la ligne CS
  SPI.transfer(0x84);
  delayMicroseconds(15);
  SPI.transfer(0);
  delayMicroseconds(15);
  SPI.transfer(0);
  delayMicroseconds(15);
  SPI.transfer(0);
  delayMicroseconds(15);
  digitalWrite(CS, HIGH);                 // désactivation de la ligne CS
  break;
 case 3:                                  // deux boutons actifs
  // allumage de la led en bleu
  digitalWrite(CS, LOW);                  // activation de la ligne CS
  SPI.transfer(0x84);
  delayMicroseconds(15);
  SPI.transfer(0);
  delayMicroseconds(15);
  SPI.transfer(0);
  delayMicroseconds(15);
  SPI.transfer(255);
  delayMicroseconds(15);
  digitalWrite(CS, HIGH);                 // désactivation de la ligne CS
  delay(1000);
  // extinction de la led
  digitalWrite(CS, LOW);                  // activation de la ligne CS
  SPI.transfer(0x84);
  delayMicroseconds(15);
  SPI.transfer(0);
  delayMicroseconds(15);
  SPI.transfer(0);
  delayMicroseconds(15);
  SPI.transfer(0);
  delayMicroseconds(15);
  digitalWrite(CS, HIGH);                 // désactivation de la ligne CS
  break;
 default:
```

```
  break;
  delay(1000) ;
  }
}
```

Schéma de câblage : identique au module PmodJSTK



Programme Arduino :

```
/************************************************************************
*
*   Test du module Pmod afficheur OLED
*
************************************************************************
* Description:  Pmod_OLED
* Un nuage de points est affiché aléatoirement puis lE MESSAGE lextronic apparaît
* et clignote 3 fois
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod OLED
*       (télécharger les bibliothèques https://github.com/adafruit/Adafruit_SSD1306 et
*        https://github.com/adafruit/Adafruit-GFX-Library)
*
************************************************************************/

// Affectation des broches
#define OLED_MOSI   9
#define OLED_CLK   10
#define OLED_DC    11
#define OLED_CS    12
#define OLED_RESET 13

// Appel des bibliothèques
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <SPI.h>

Adafruit_SSD1306 afficheur(OLED_MOSI, OLED_CLK, OLED_DC, OLED_RESET, OLED_CS);    //création de l'objet

int x;
int y;

void setup(void)
{
  afficheur.begin();           // initialisation de l'objet afficheur
  afficheur.display();         // rafraichissement de l'écran
  afficheur.clearDisplay();    // effacement de l'écran
  afficheur.display();         // rafraichissement de l'écran
}
```

```
void loop()
{
  afficheur.clearDisplay();              // effacement de l'écran
  afficheur.display();                   // rafraichissement de l'écran
  for (int i=0; i <= 50; i++)            // apparition du nuage de points
  {
    x=random(128);                       // x prend une valeur aléatoire comprise entre 0 et 128
    y=random(32);                        // y prend une valeur aléatoire comprise entre 0 et 32
    afficheur.drawPixel(x, y, WHITE);    // affichage d'un pixel en (x,y)
    afficheur.display();                 // rafraichissement de l'écran
    delay(50);                           // pause de 50 ms
  }
  afficheur.setTextSize(2);              // configuration de la taille des caractères
  afficheur.setTextColor(WHITE);
  afficheur.setCursor(10,10);            // placement du curseur en x=10 et y=10
  afficheur.println("LEXTRONIC");        // écriture de LEXTRONIC
  afficheur.display();                   // rafraichissement de l'écran
  delay(1000);
  for (int i=0; i <= 3; i++)             // clignotement du message
  {
    afficheur.setCursor(10,10);
    afficheur.println("LEXTRONIC");
    afficheur.display();
    delay(1000);
    afficheur.clearDisplay();
    afficheur.display();
    delay(500);
  }
}
```

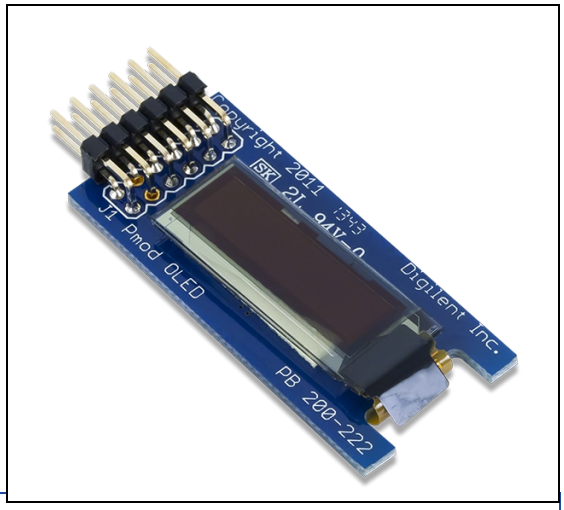
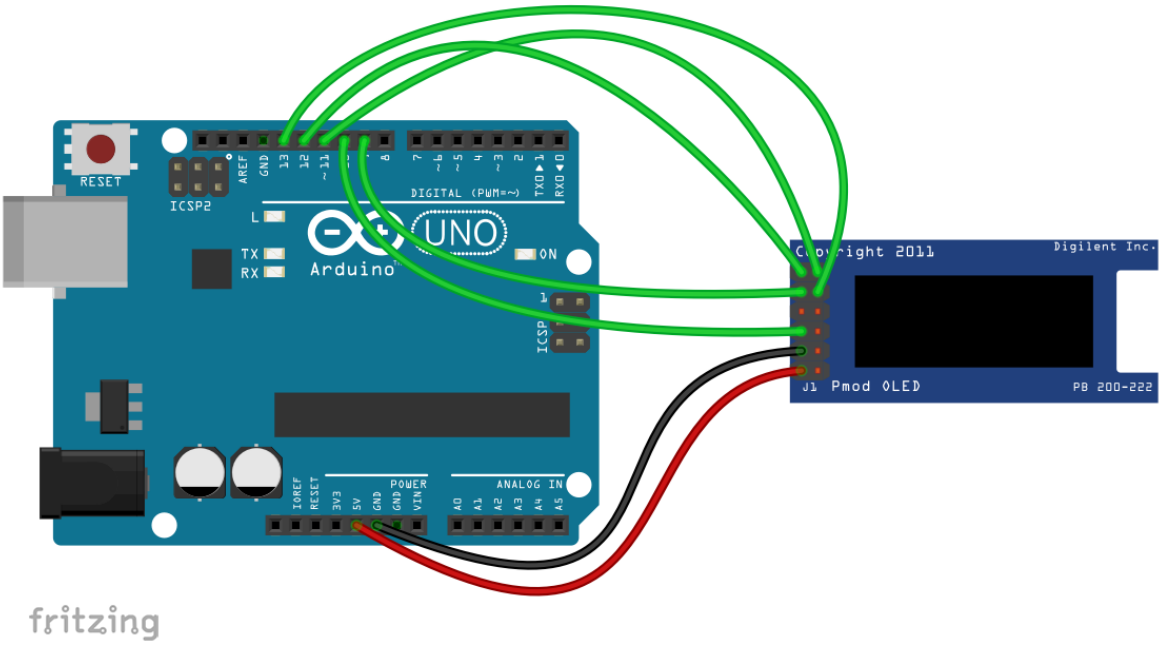


Schéma de câblage :



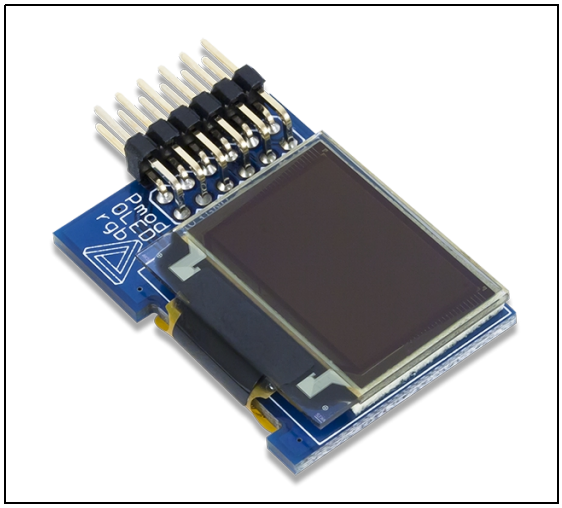
PMODOELDRGB :

Module afficheur OLEDRGB

Attention, ce module fonctionne sous 3,3 V

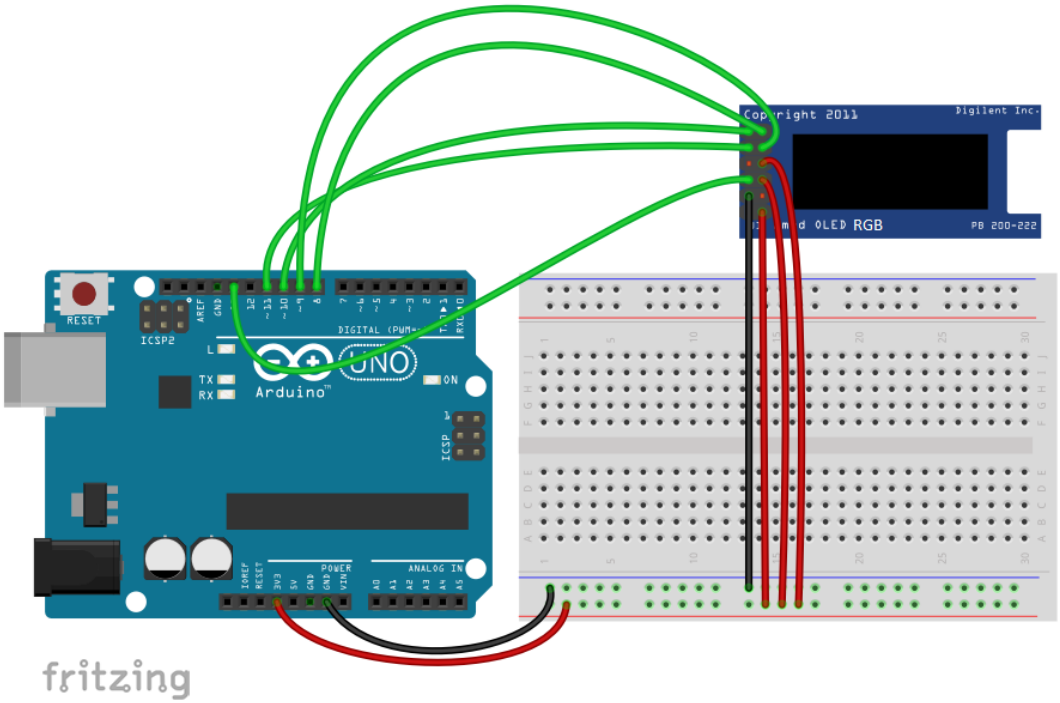
Programme Arduino :

```
/************************************************************************
*
*   Test du module Pmod afficheur OLEDRGB
*
************************************************************************
* Description:  Pmod_OLEDRGB
* Le message "Test module Pmod Digilent Lextronic" est affiché sur
* l'afficheur avec des couleurs et des tailles différentes
*
* Matériel
*      1. Arduino Uno
*      2. Module Pmod OLEDRGB (télécharger les bibliothèques
*        https://github.com/adafruit/Adafruit-SSD1331-OLED-Driver-Library-for-Arduino
*        https://github.com/adafruit/Adafruit-GFX-Library)
*
************************************************************************/

// Affectation des broches
#define sck 13
#define mosi 11
#define cs 10
#define res 9
#define dc 8

// Définition des couleurs
#define NOIR        0x0000
#define BLEU        0x001F
#define ROUGE       0xF800
#define VERT        0x07E0
#define CYAN        0x07FF
#define MAGENTA     0xF81F
#define JAUNE       0xFFE0
#define BLANC       0xFFFF

// Appel des bibliothèques
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1331.h>
#include <SPI.h>

Adafruit_SSD1331 afficheur = Adafruit_SSD1331(cs, dc, mosi, sck, res);    //création de l'objet
```



```
void setup(void)
{
 afficheur.begin();                                          // initialisation de l'objet afficheur
}

void loop()
{
 afficheur.fillScreen(NOIR);                                  // fond de l'écran noir
 afficheur.setTextColor(CYAN);                                // couleur du texte en cyan
 afficheur.setCursor(0,0);                                    // placement du curseur en x=0 et y=15
 afficheur.print("Test module Pmod");                         // écriture du texte
 delay(500);                                                  // pause de 500 ms
 afficheur.setCursor(0,15);                                   // placement du curseur en x=0 et y=15
 afficheur.setTextSize(2);                                    // taille du texte
 afficheur.setTextColor(ROUGE);                               // couleur du texte en rouge
 afficheur.println("DIGILENT");                               // écriture du texte
 afficheur.setCursor(20,40);                                  // placement du curseur en x=20 et y=40
 afficheur.setTextSize(1);                                    // taille du texte
 afficheur.setTextColor(VERT);                                // couleur du texte en vert
 afficheur.println("LEXTRONIC");                              // écriture du texte
 afficheur.drawFastHLine(1, 60, afficheur.width()-1, BLEU);   // ligne bleue de x=1 à largeur de l'écran-1 et y=60
 delay(2000);                                                 // pause de 2 s
 afficheur.fillScreen(NOIR);                                  // fond de l'écran noir (effacement de l'écran)
 afficheur.fillRoundRect(5, 5, 30, 40, 5, BLEU);              // drapeau bleu blanc rouge
 afficheur.fillRoundRect(35, 5, 30, 40, 5, BLANC);
 afficheur.fillRoundRect(65, 5, 30, 40, 5, ROUGE);
 afficheur.fillCircle(90, 55, 5, JAUNE);                      // cercle jaune de rayon=5 en x=90 et y=55
 delay(2000);                                                 // pause de 2 s
}
```

Schéma de câblage :



Programme Arduino :

```
/************************************************************************
*
*   Test du module Pmod convertisseur N/A R/2R
*
************************************************************************
* Description:  Pmod_R2R
* La tension de sortie est paramétrable en fonction des niveaux logiques
* appliqués sur les broches 2 à 9 de l'Arduino.
* La tension de sortie est la somme de toutes les composantes.
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod R2R
*
************************************************************************/
void setup()
{
 for (int i=2; i<=9; i++)              // configuration des broches 2 à 9 en sortie
 {
   pinMode(i,OUTPUT);
 }
}

void loop()
{
 // L'entrée D7 du module à l'état haut impose une composante de VCC/2
 digitalWrite(9,HIGH);
 // L'entrée D6 du module à l'état haut impose une composante de VCC/4
 digitalWrite(8,LOW);
 // L'entrée D5 du module à l'état haut impose une composante de VCC/8
 digitalWrite(7,LOW);
 // L'entrée D4 du module à l'état haut impose une composante de VCC/16
 digitalWrite(6,LOW);
 // L'entrée D3 du module à l'état haut impose une composante de VCC/32
 digitalWrite(5,HIGH);
 // L'entrée D2 du module à l'état haut impose une composante de VCC/64
 digitalWrite(4,HIGH);
 // L'entrée D1 du module à l'état haut impose une composante de VCC/128
 digitalWrite(3,HIGH);
 // L'entrée D0 du module à l'état haut impose une composante de VCC/256
 digitalWrite(2,HIGH);
}
```

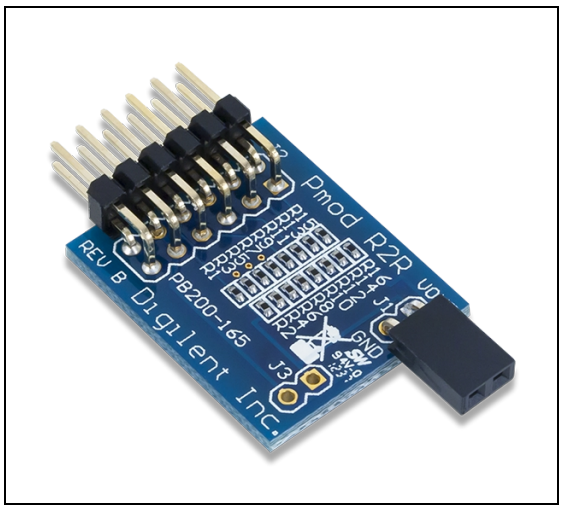


Schéma de câblage :