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

001 : 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.");
  }
 }
}
```

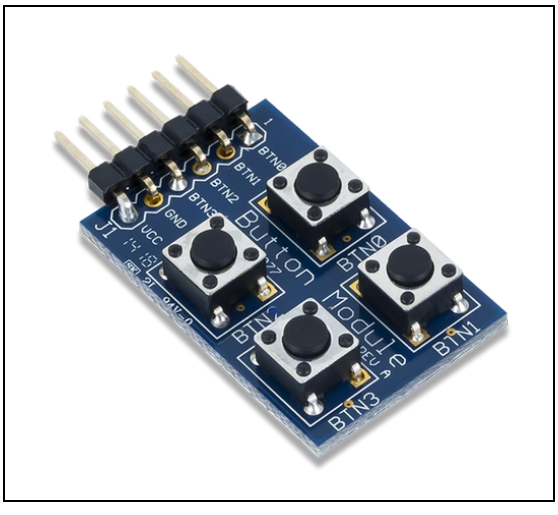
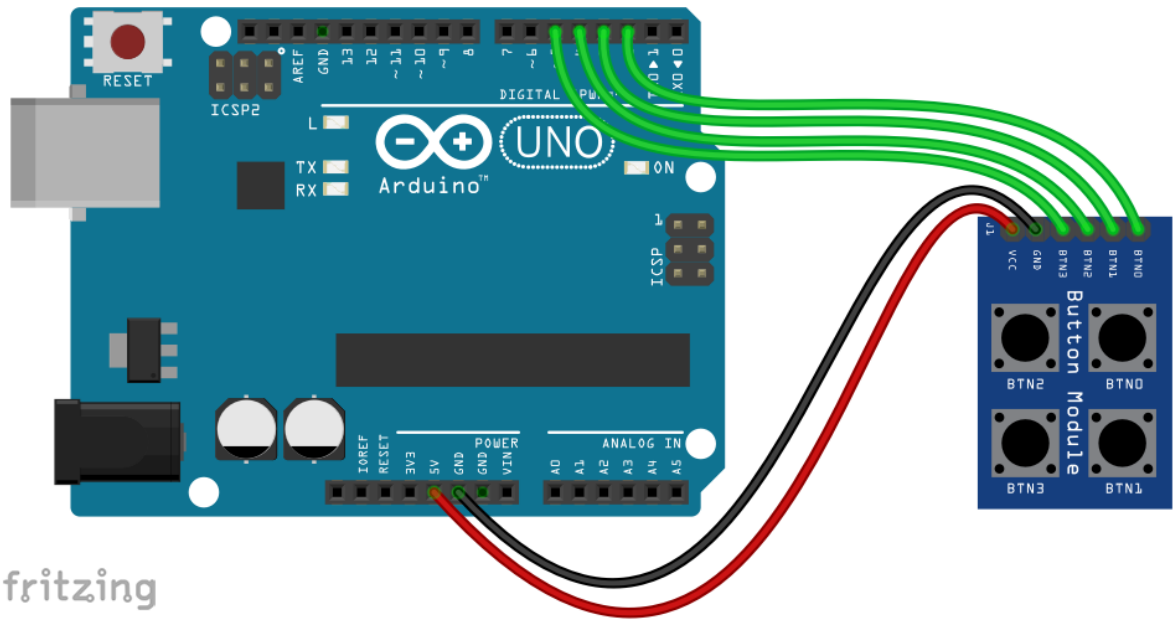


Schéma de câblage :



002 : 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);
}
```

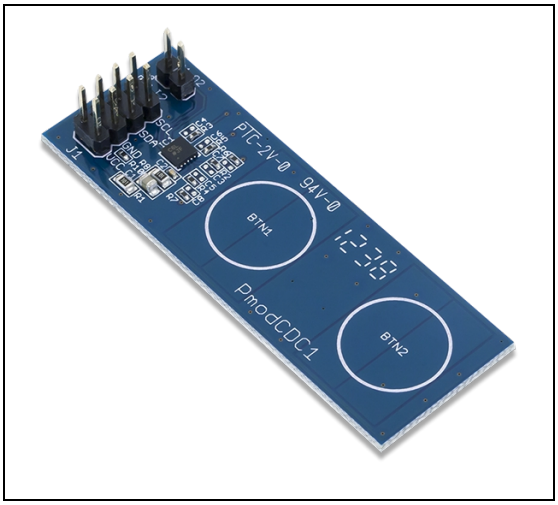
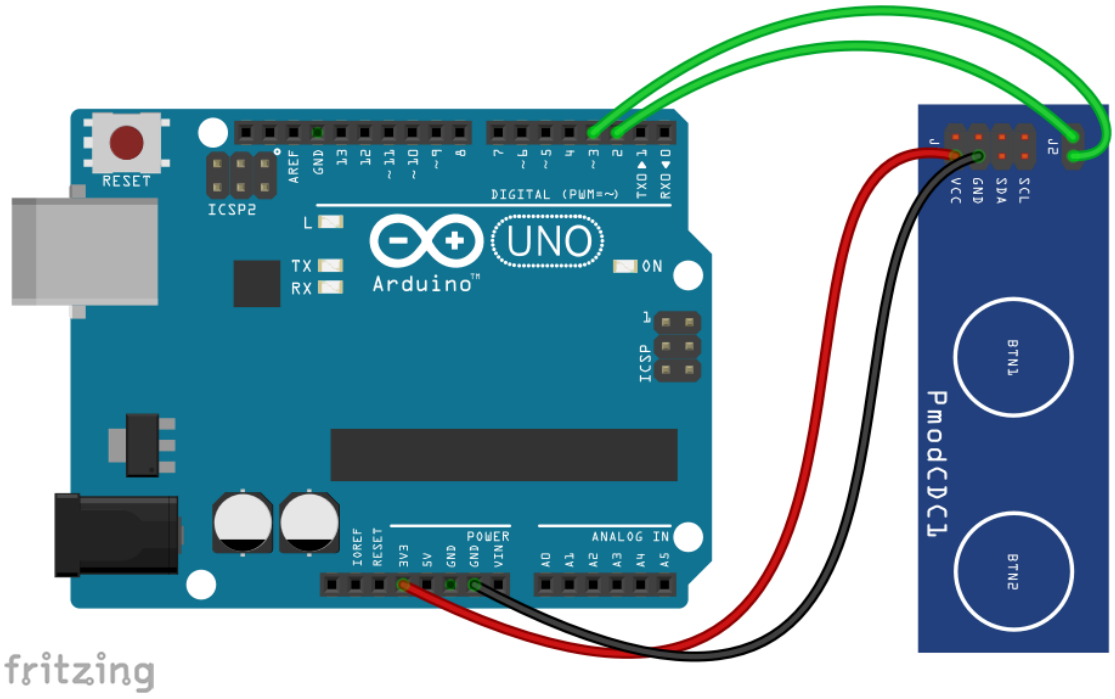


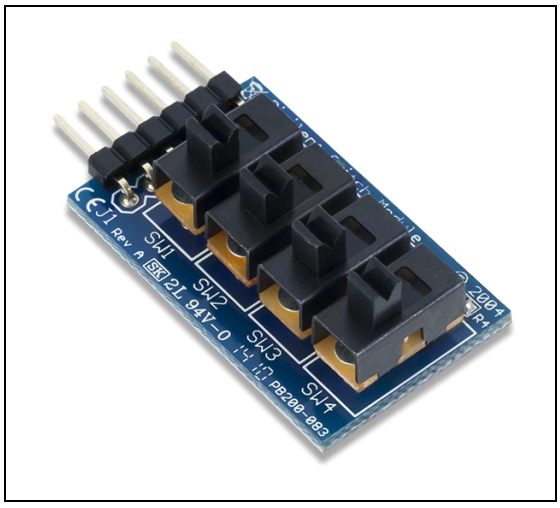
Schéma de câblage :



003 : PMODSWT : Module 4 interrupteurs

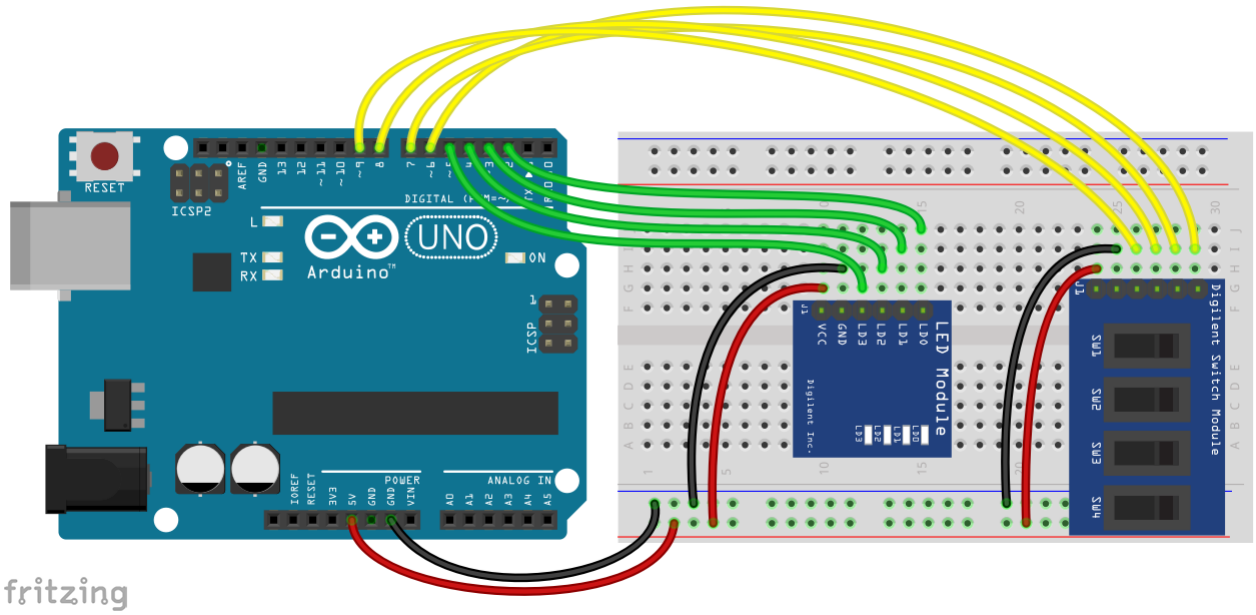
Programme Arduino :

```
/************************************************************************
*
*    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 :



004 : 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);
}
```

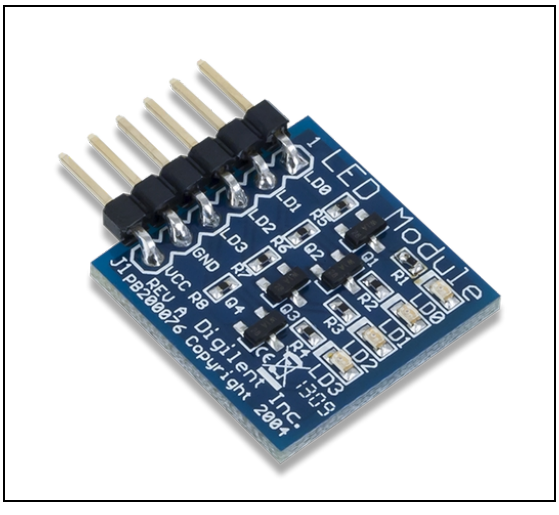
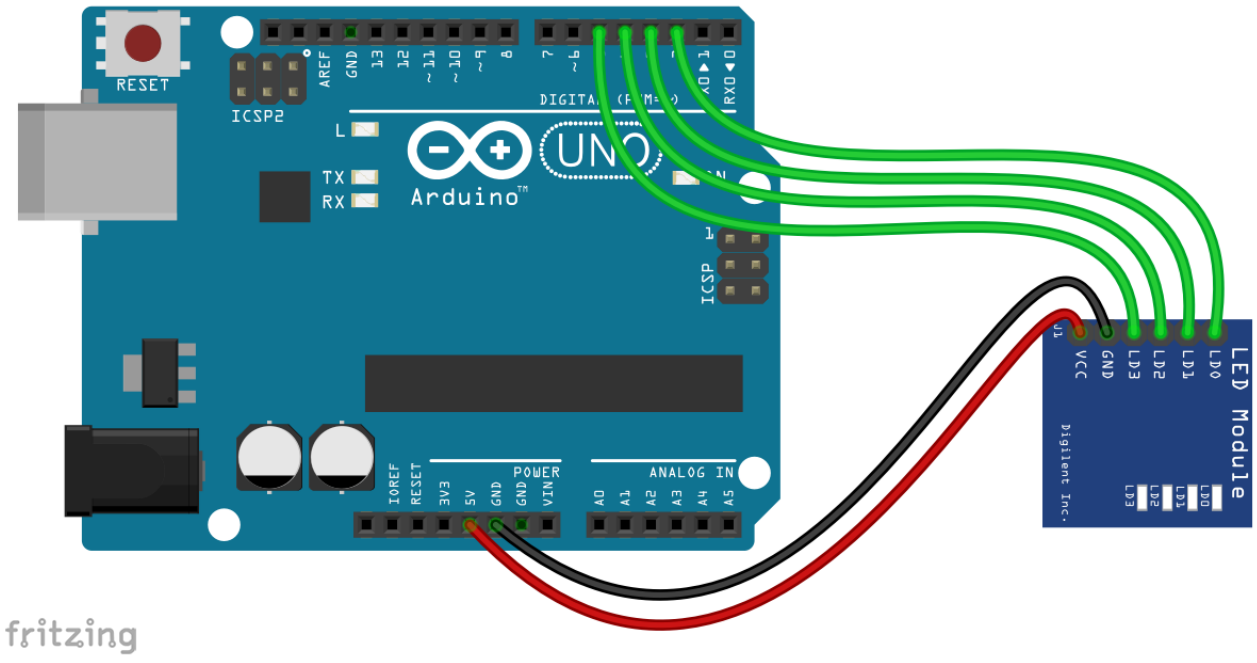


Schéma de câblage :



005 : 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);
}
```

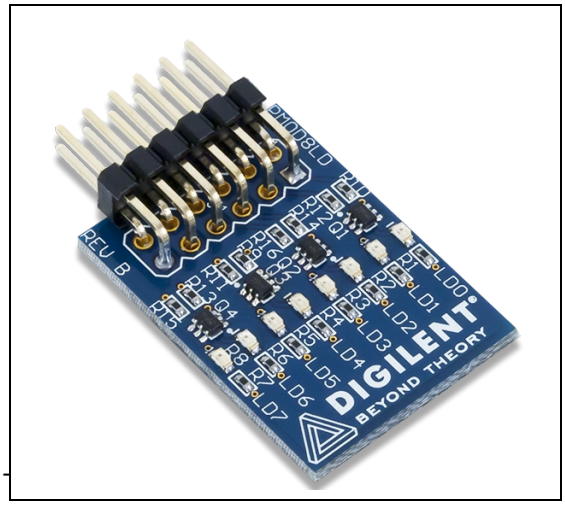
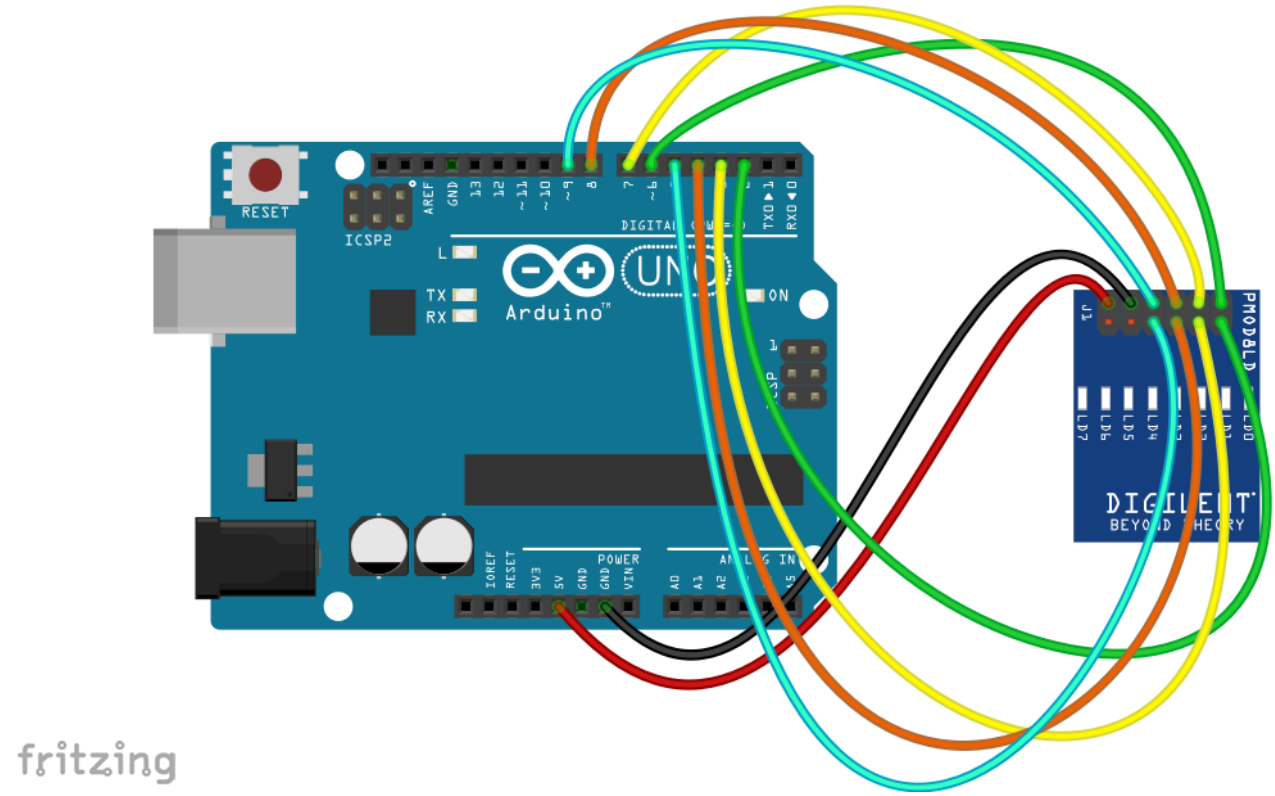


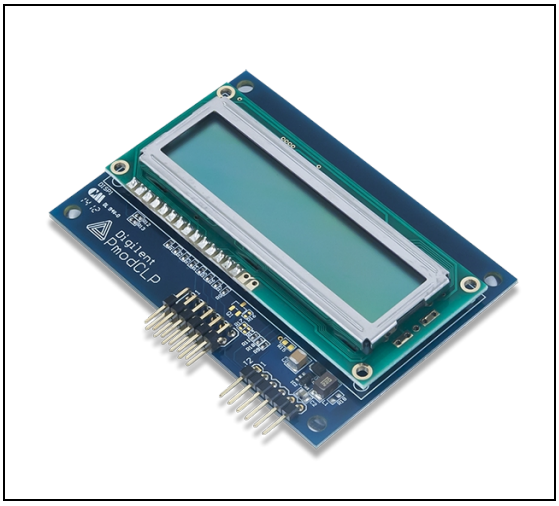
Schéma de câblage :



006 : PMODCLP : Afficheur LCD 16x2 à interface parallèle

Programme Arduino :

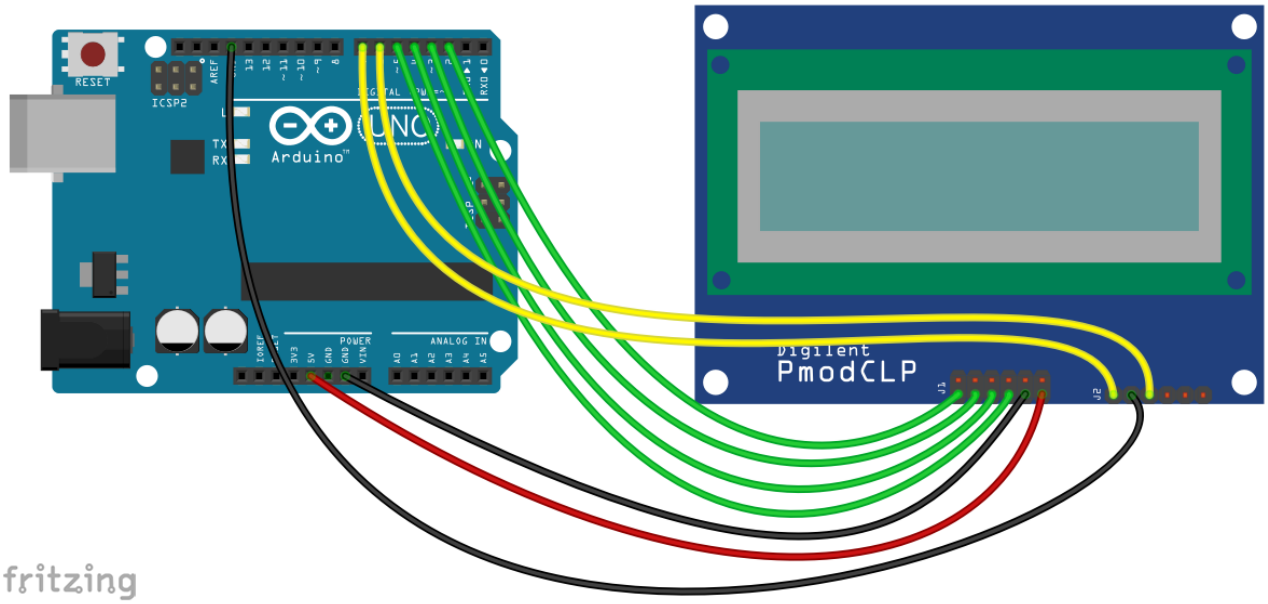
```
/************************************************************************
*
*    Test du module Pmod afficheur LCD interface parallèle
*
************************************************************************
* Description:  Pmod_CLP
* Le message "Test module Pmod Digilent partenaire de Lextronic" 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 :



007 : PMODCLS : Afficheur LCD 16x2 à interface série

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
}
```

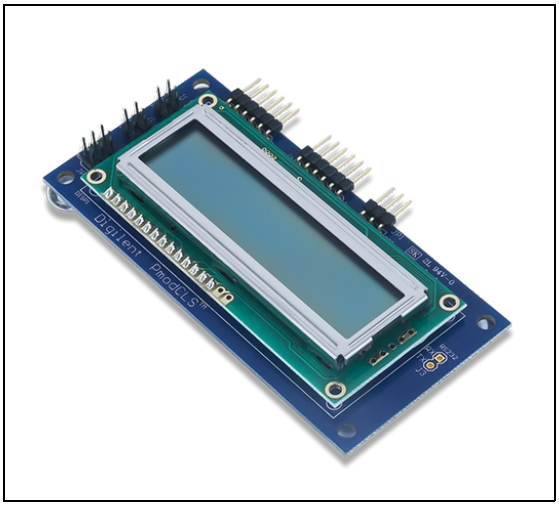
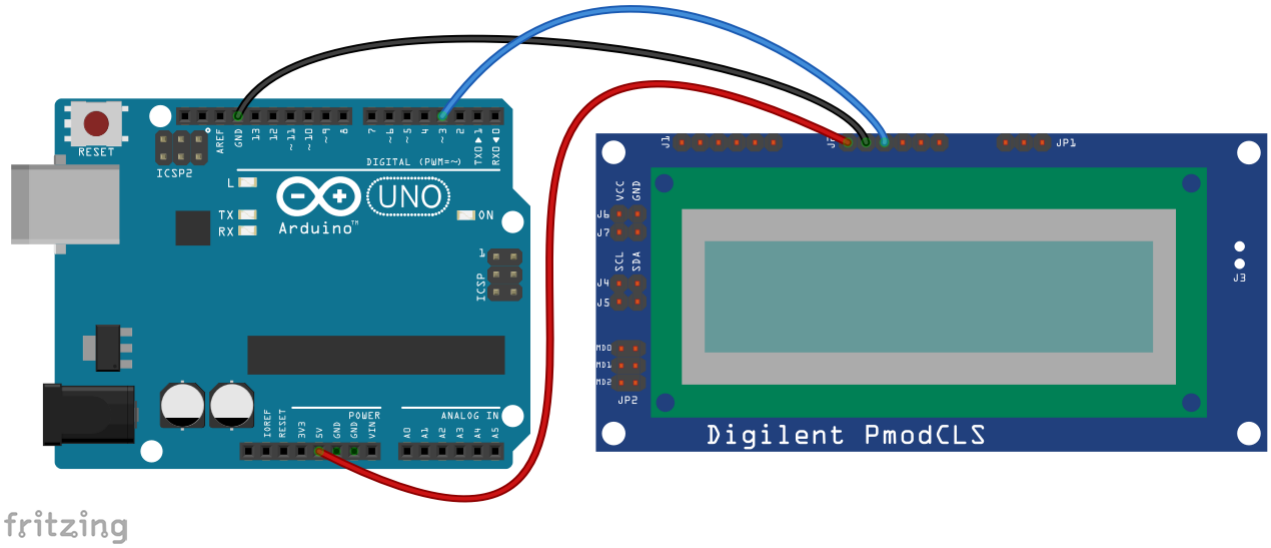


Schéma de câblage :



008 : PMODJSTK : Module joystick

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;
 }
}
```

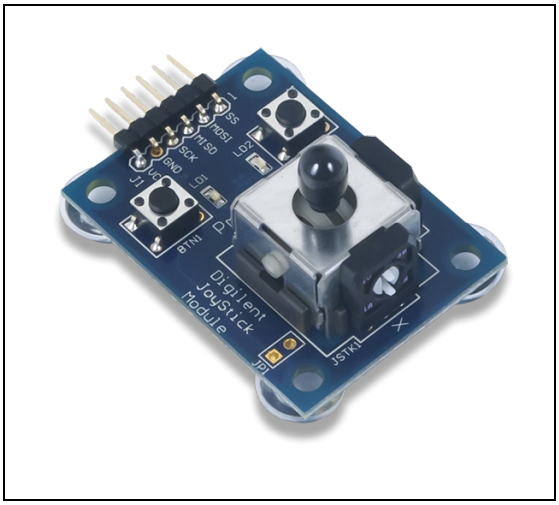
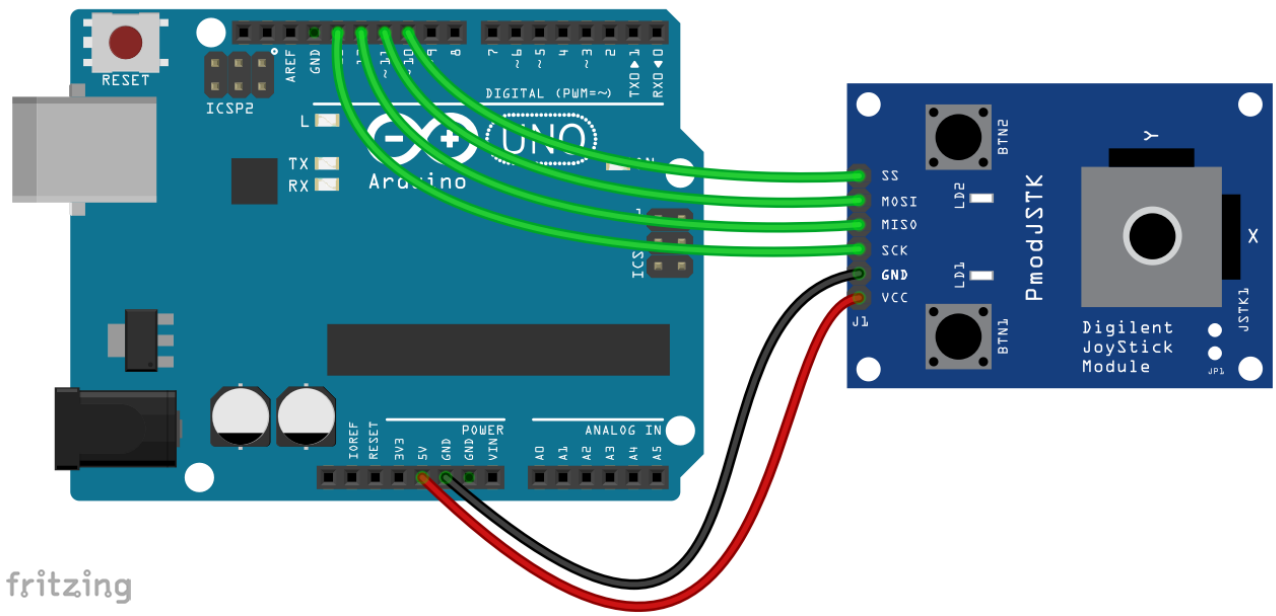


Schéma de câblage :



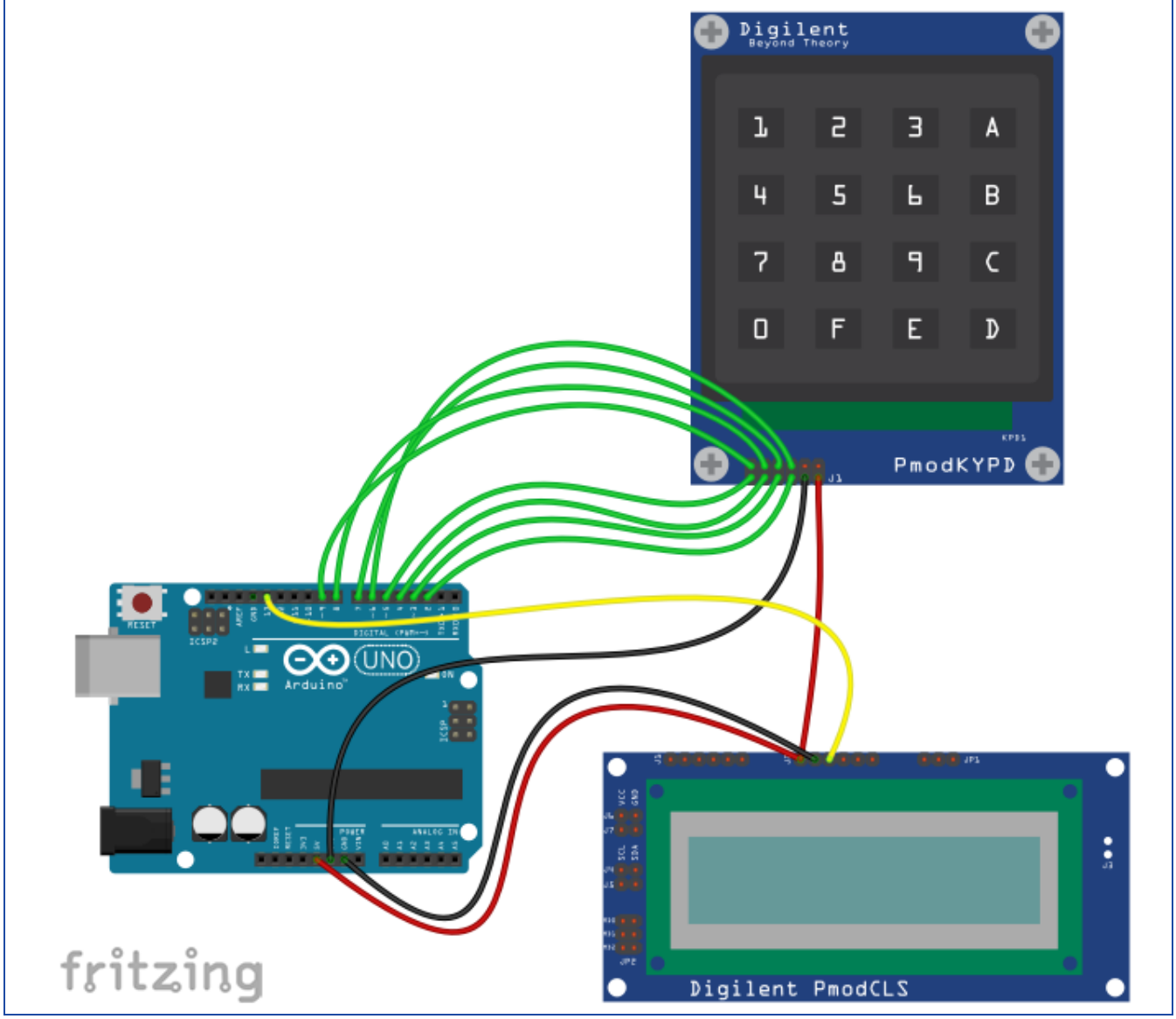
009 : PMODKYPD : Module clavier 16 touches

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 :



010 : 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;
  }
}
```

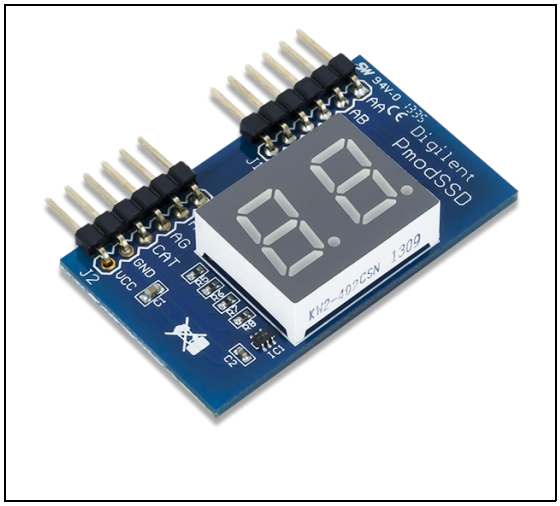
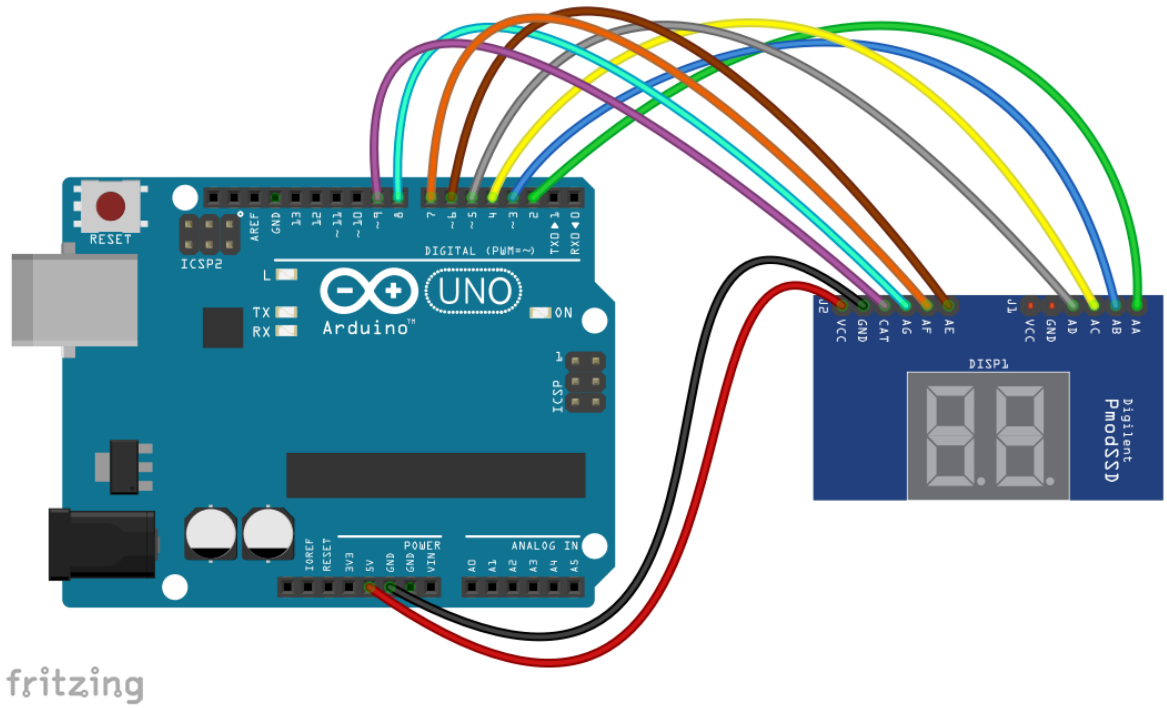


Schéma de câblage :



011 : PMODDA1 : Module convertisseur « numérique/analogique » 8 bit 4 sorties

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
}
```

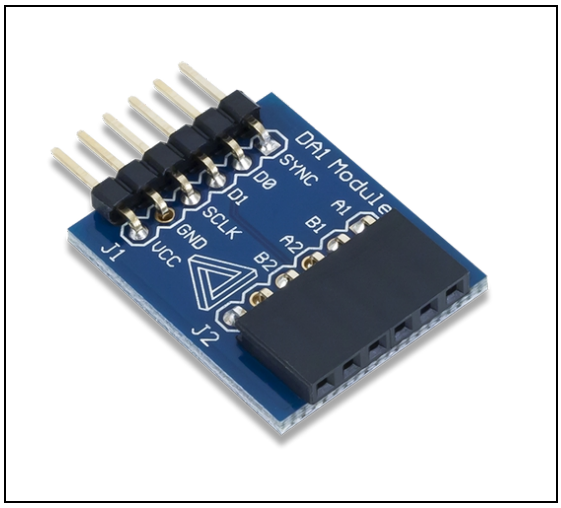
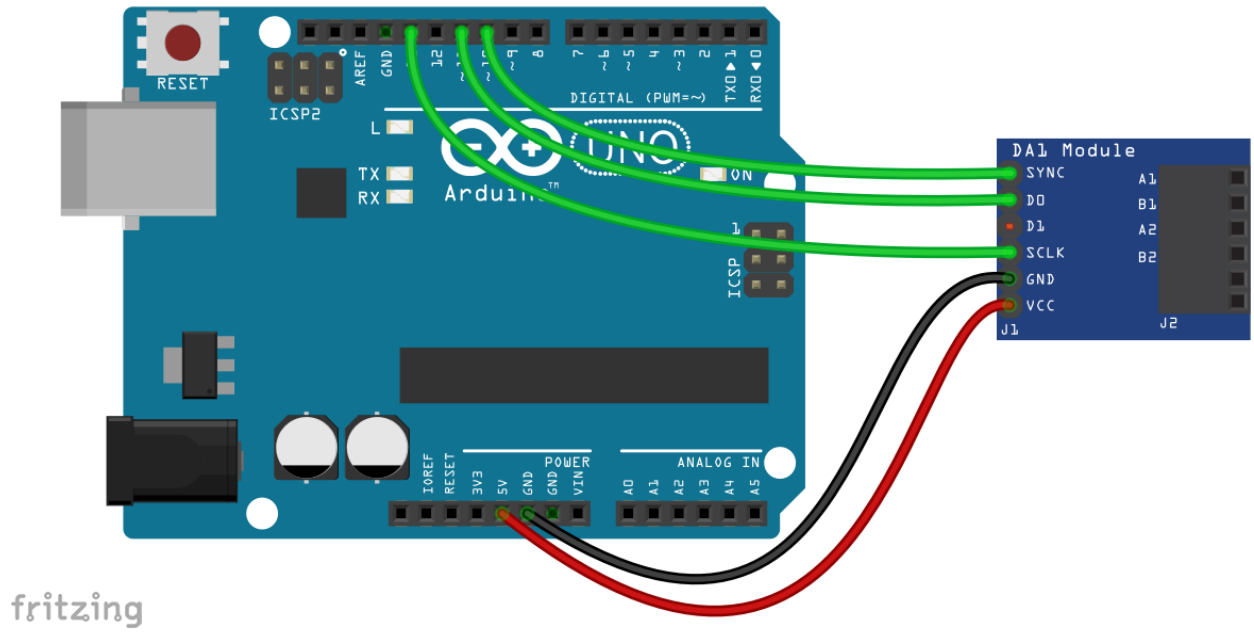
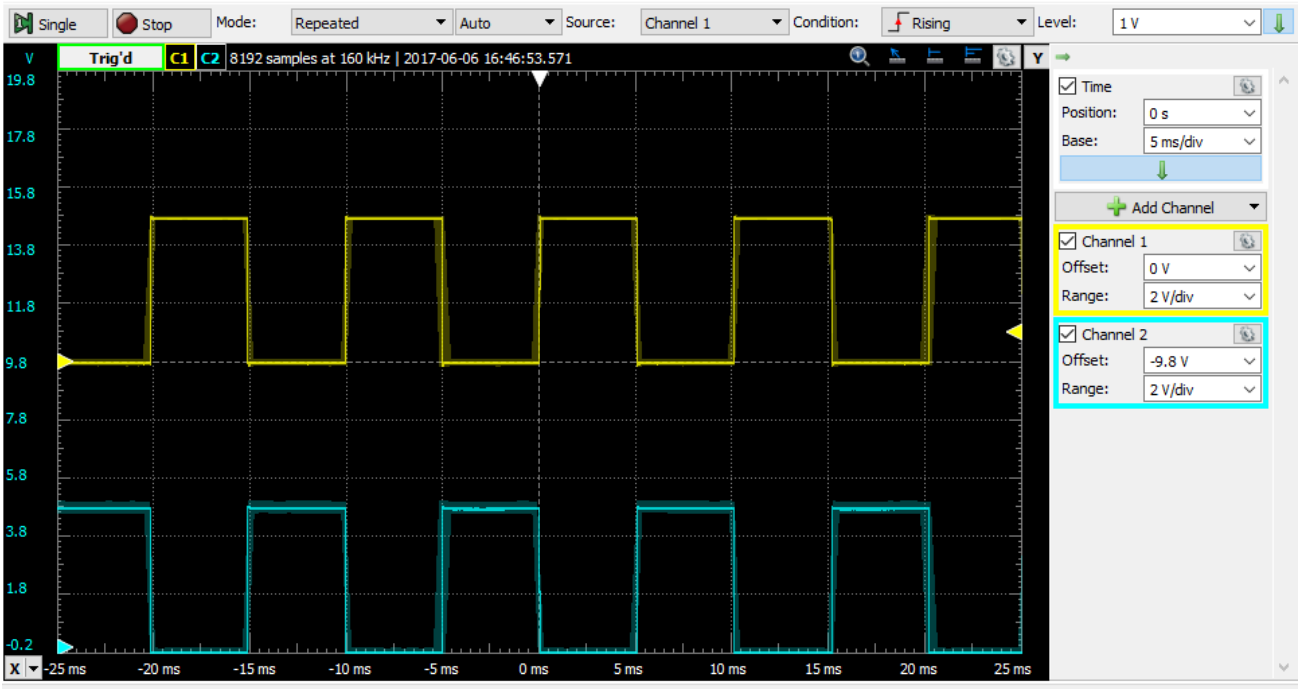


Schéma de câblage :



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 :



012 : PMODDA2 : Module convertisseur « numérique/analogique » 12 bit

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");
}
```

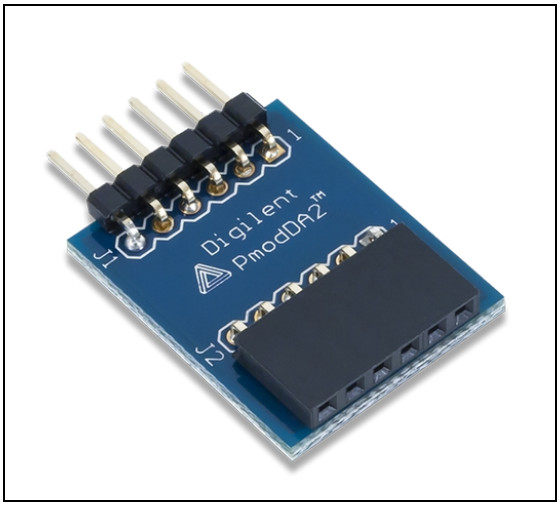
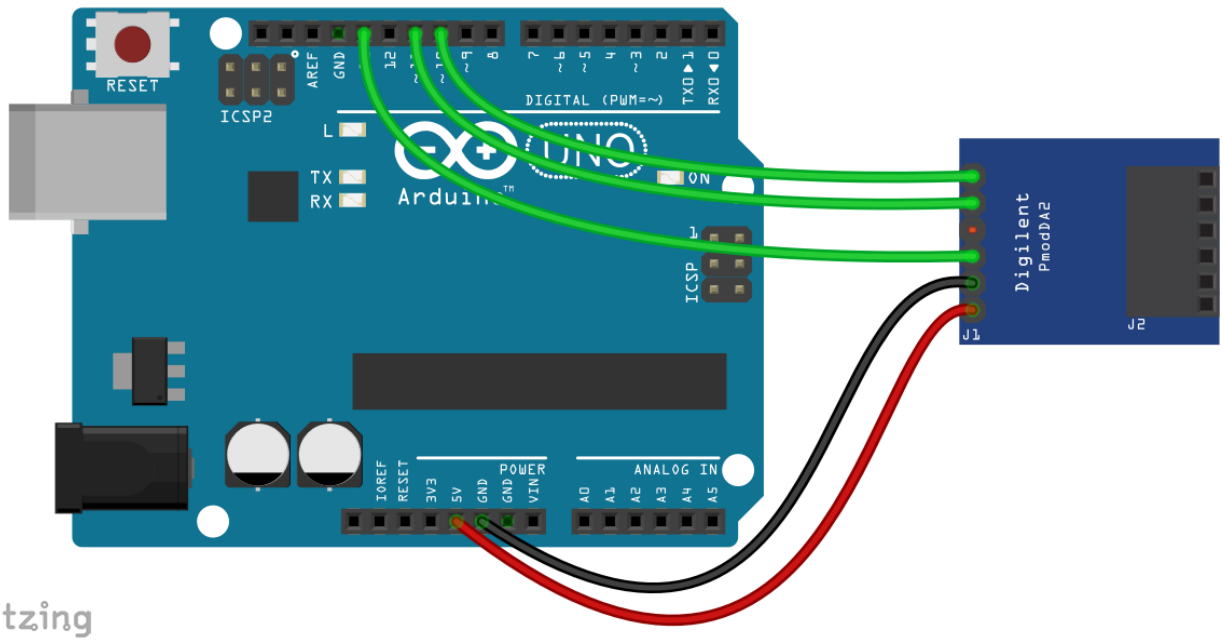


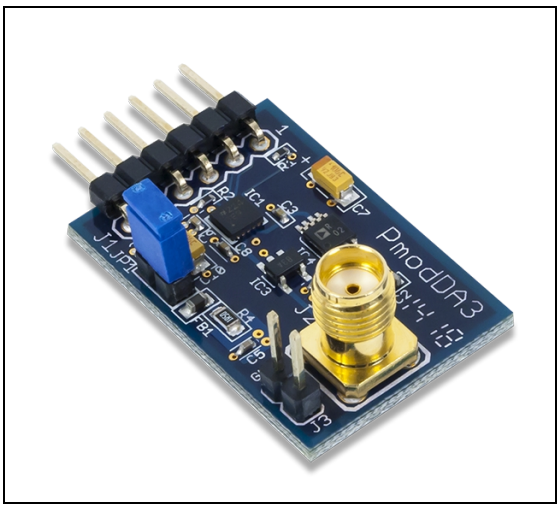
Schéma de câblage :



013 : PMODDA3 : Module convertisseur « numérique/analogique » 16 bit

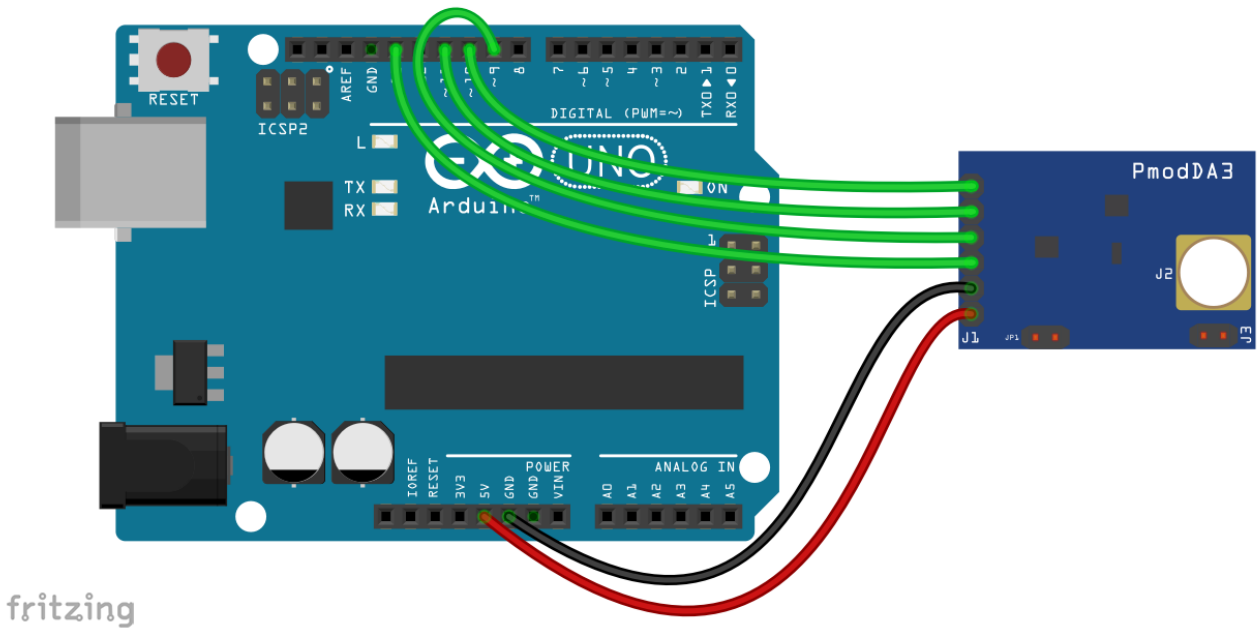
Programme Arduino :

```
/************************************************************************
*
*   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
```



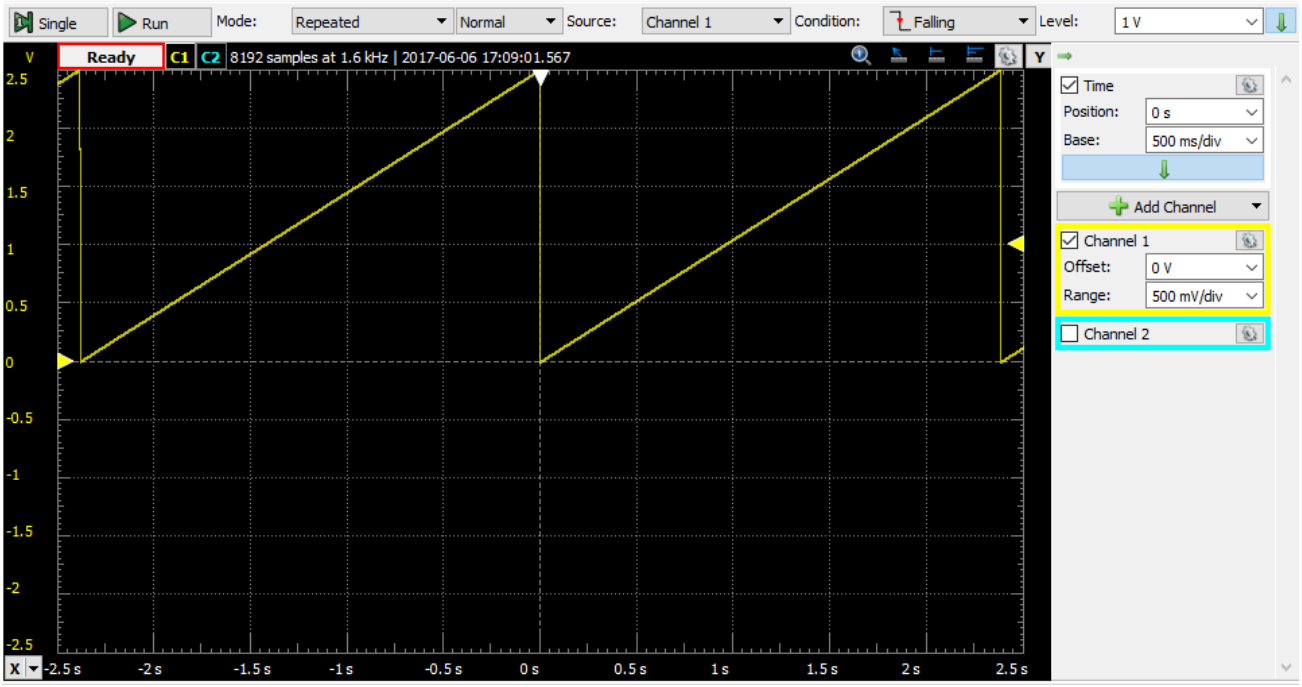
```
  SPI.transfer(j);                // envoi des bit de poids faible
  digitalWrite(CS, HIGH);                 // désactivation de la ligne CS
  digitalWrite(LDAC, LOW);                // activation de la ligne LDAC
  /* la période du signal est de 2,4 s environ.
   * Si on souhaite l'augmenter, on 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 :



014 : PMODDA4 : Module convertisseur « numérique/analogique » 12 bit 8 sorties

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
}
```

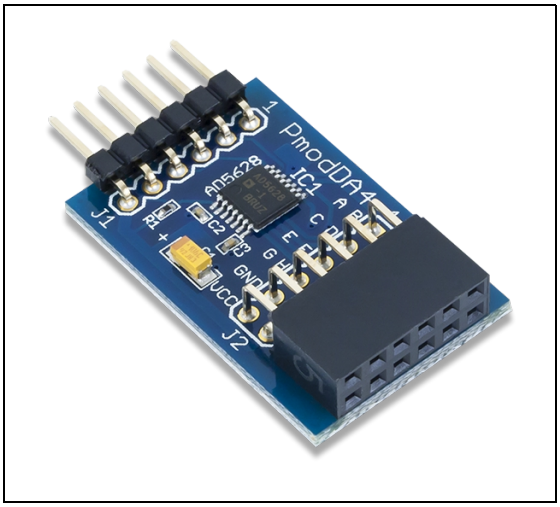
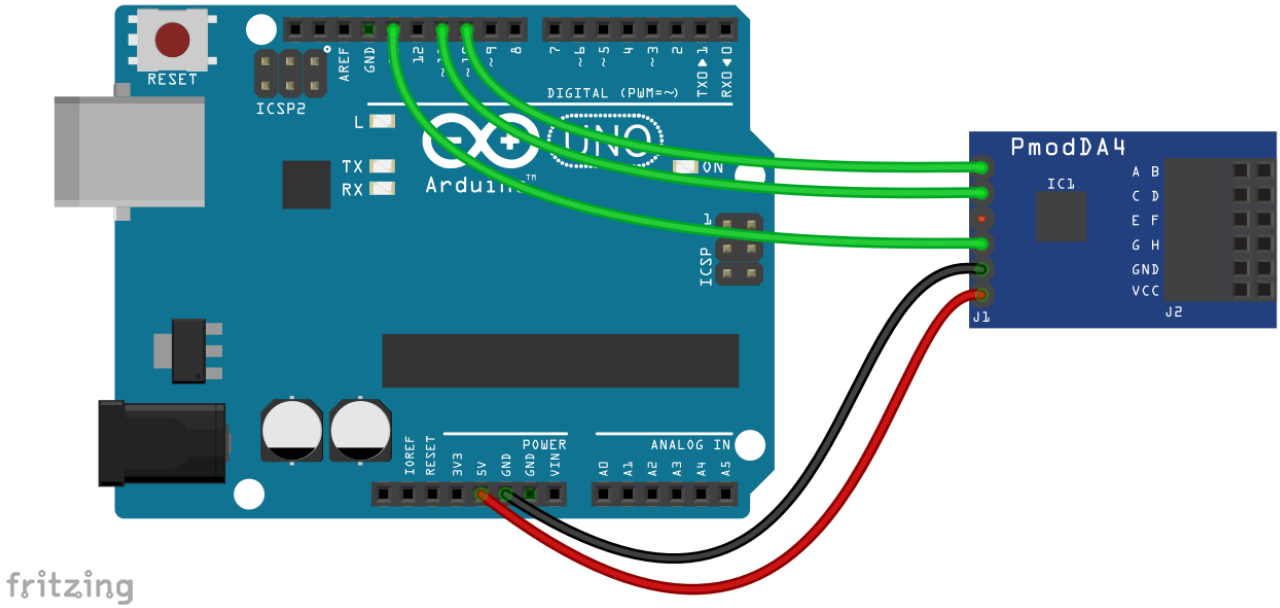


Schéma de câblage :



015 : 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);
        }
}
```

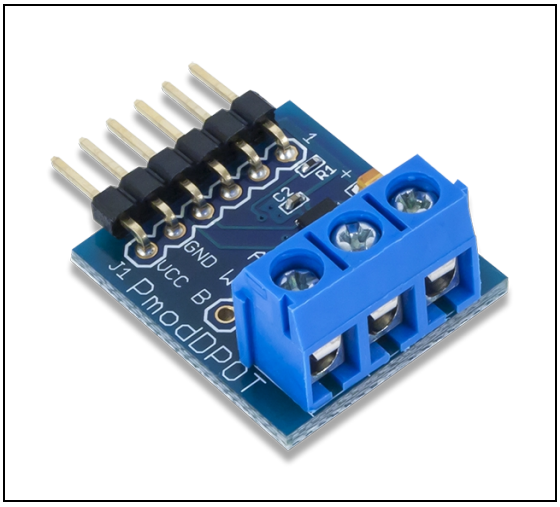
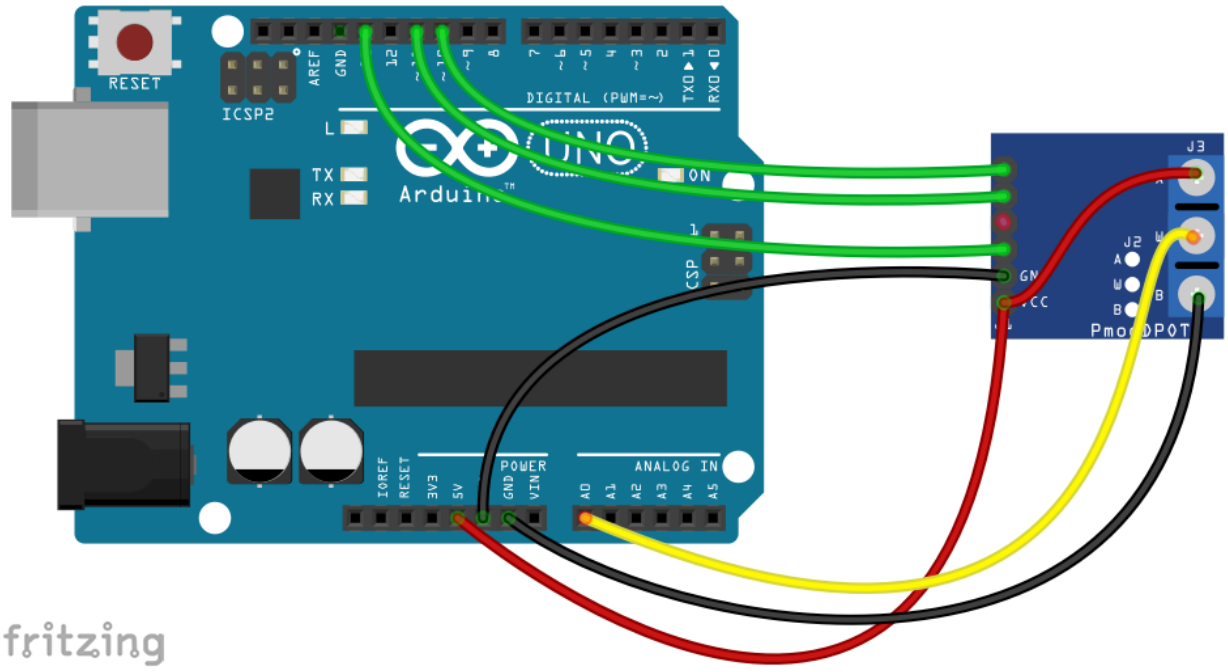


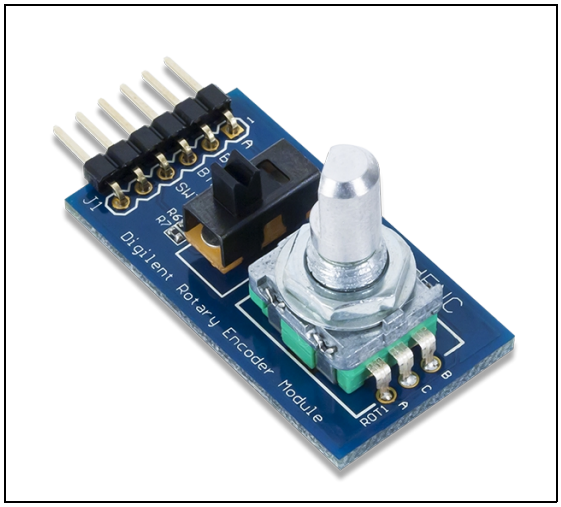
Schéma de câblage :



016 : PMODENC : Module encodeur rotatif

Programme Arduino :

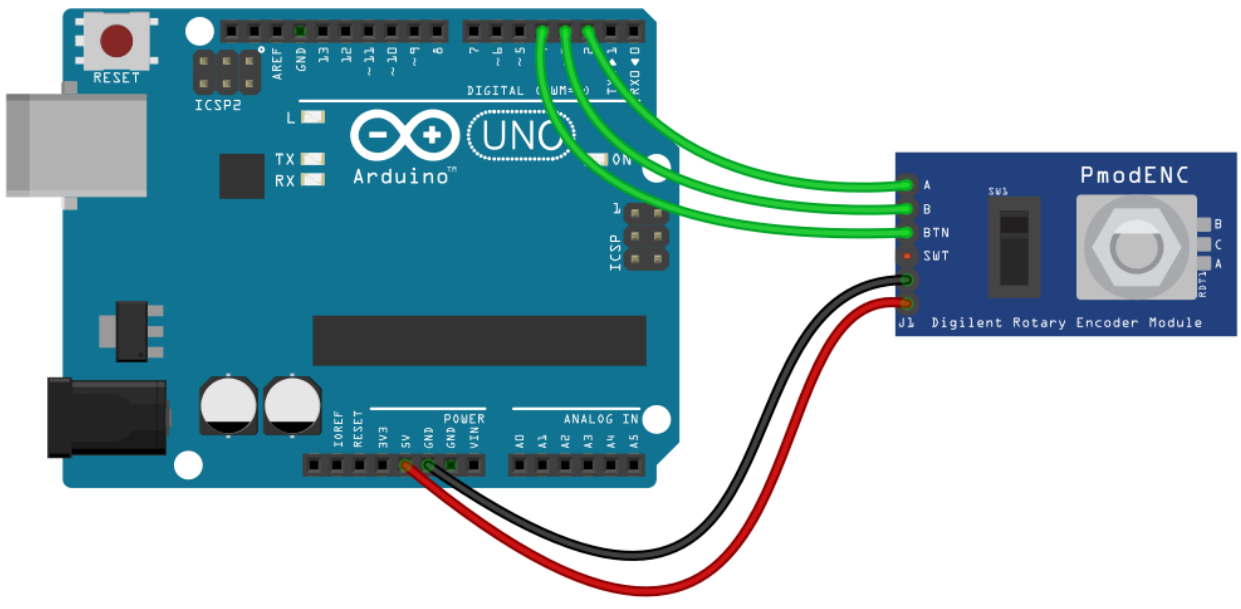
```
/************************************************************************
*
*    Test du module Pmod Encodeur rotatif
*
************************************************************************
* Description:  Pmod_ENC
* Le sens de rotation de l'axe de l'encodeur est affiché dans le moniteur
* série. 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 :



017 : PMODJSTK2 : Module joystick 2

Programme Arduino :

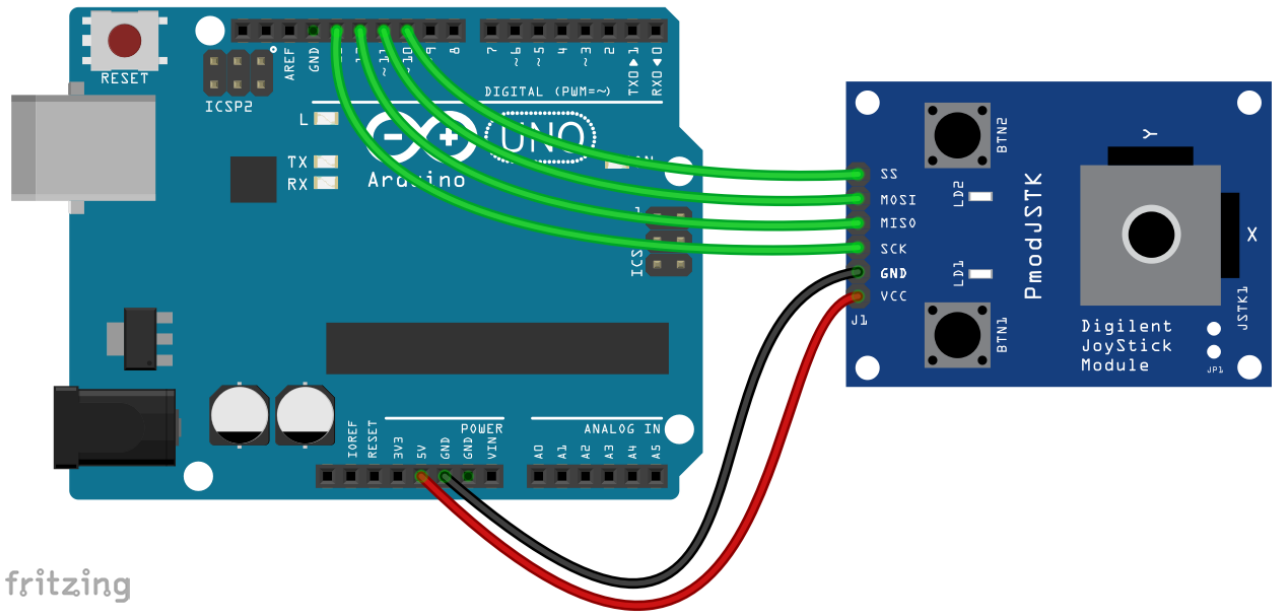
```
/************************************************************************
*
*    Test du module Pmod joystick 2
*
************************************************************************
* Description:  Pmod_JSTK2
* Les valeurs X et Y sont affichées dans le moniteur série sous forme de
* tableau 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



018 : PMODOLED : Module afficheur OLED

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);
  }
}
```

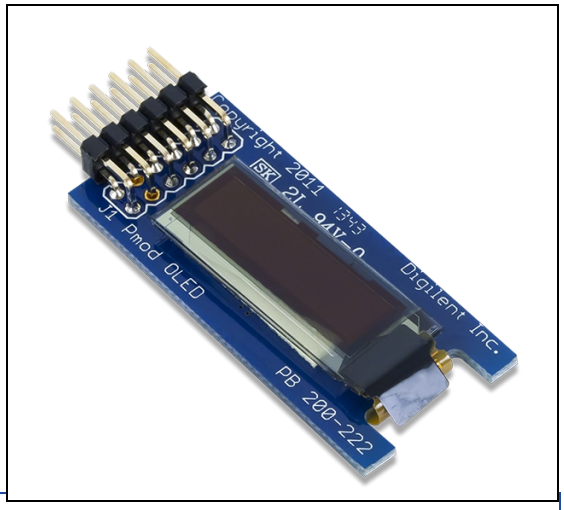
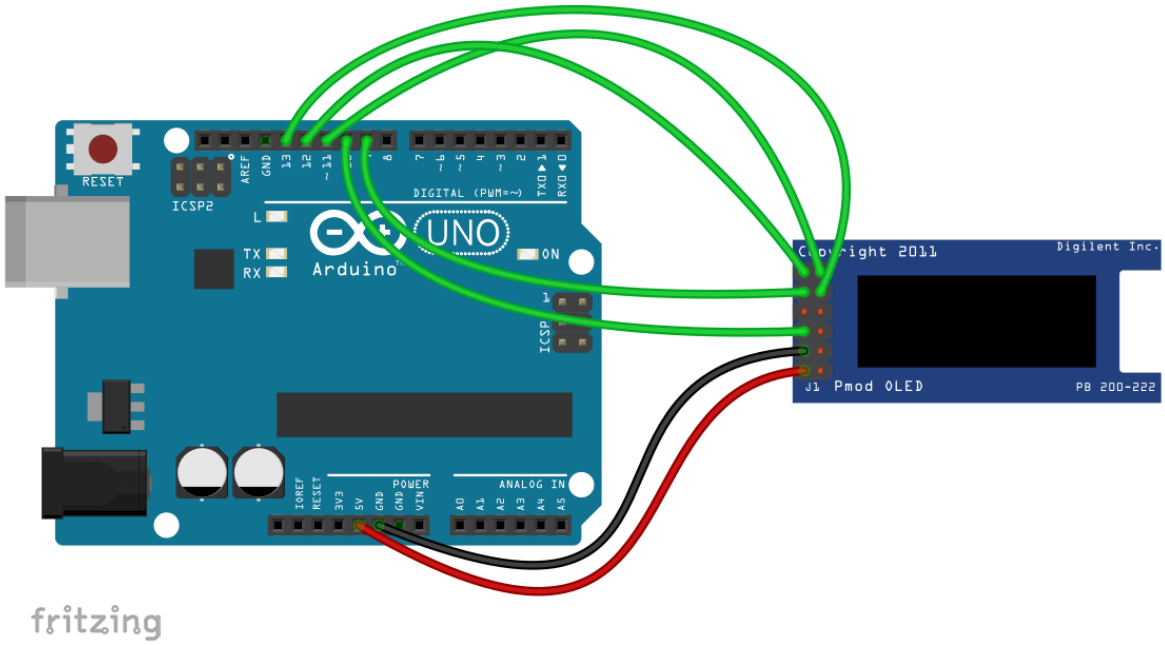


Schéma de câblage :



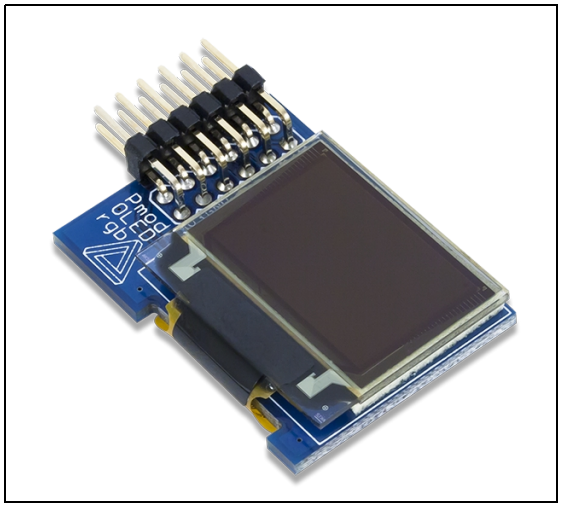
019 : PMODOELDRGB :

Module afficheur OLEDRGB

Attention, ce module fonctionne sous 3,3 V

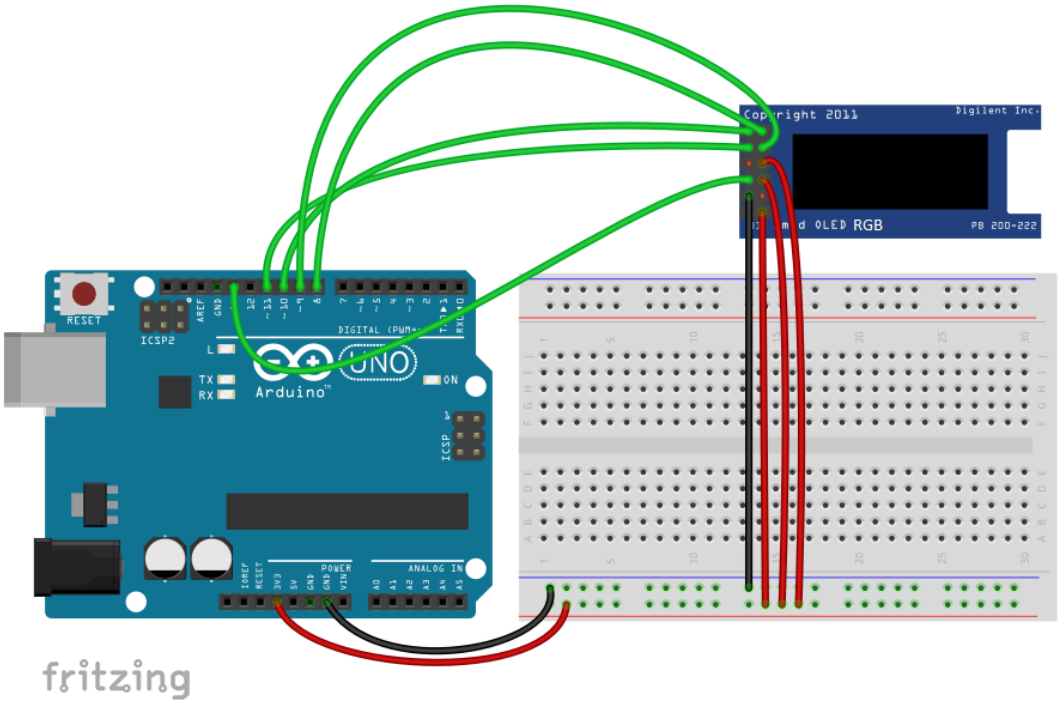
Programme Arduino :

```
/************************************************************************
*
*   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 :



020 : PMODR2R : Module convertisseur « numérique/analogique » R/2R

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);
}
```

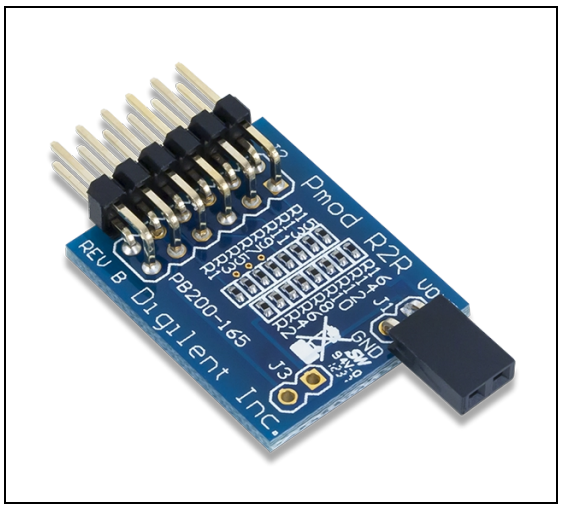
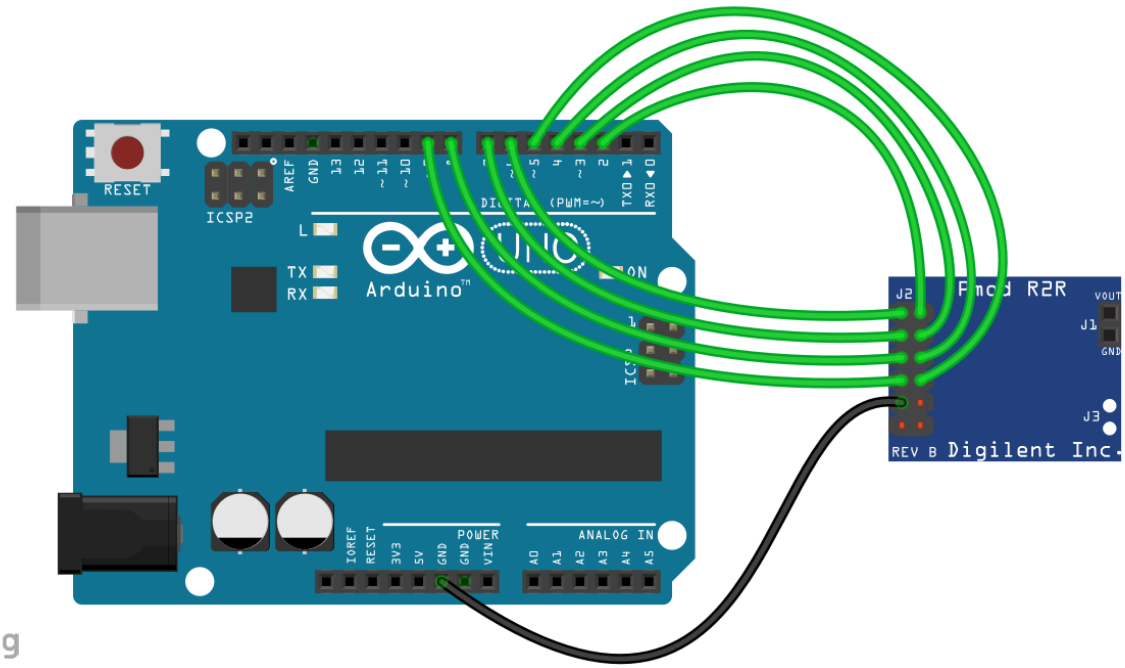


Schéma de câblage :

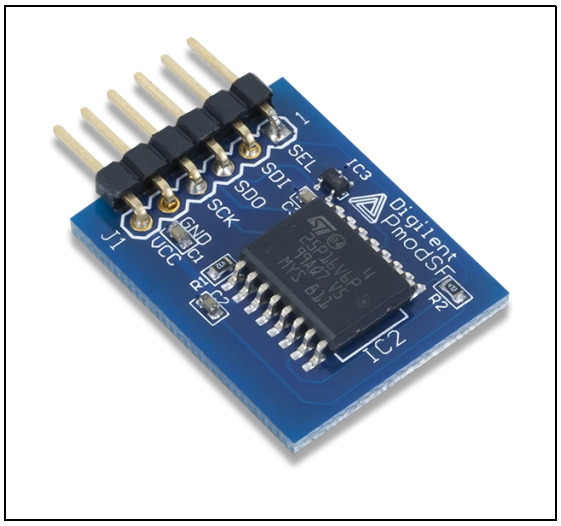


021 : PMODSF2 : Mémoire Flash

Attention, ce module fonctionne sous 3,3 V

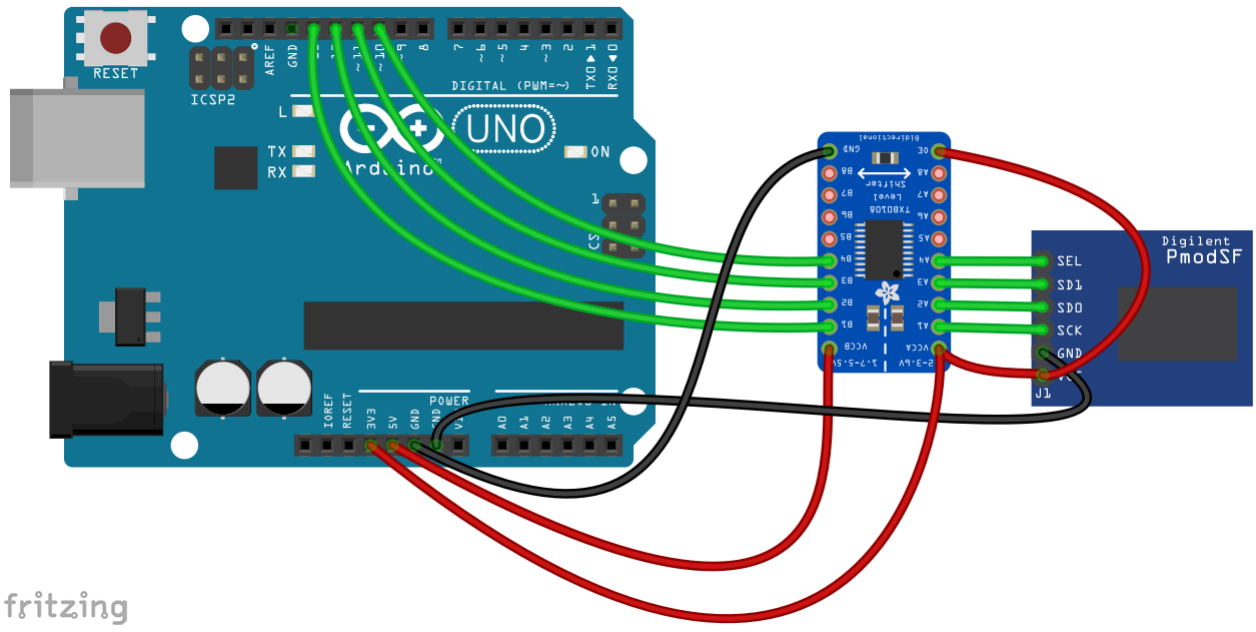
Programme Arduino :

```
/************************************************************************
*
*    Test du module Pmod mémoire Flash PCM série
*
************************************************************************
* Description:  Pmod_SF2
* Une donnée est mémorisée à l'adresse spécifiée puis relue et affichée
* dans le moniteur série.
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod SF2
*       3. Module Adafruit TXB0108
*
************************************************************************/

#define CS 10    // affectation de la broche CS

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

int i;
int adresse=0xFF;
int donnee=0x55;
byte recu[3];
byte envoi[5];

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);
 envoi[0] = adresse >> 16;
 envoi[1] = adresse >> 8;
 envoi[2] = adresse;
 envoi[3] = donnee;
 // validation écriture dans la mémoire
 digitalWrite(CS, LOW);                 // activation de la ligne CS
 SPI.transfer(0x06);                    // validation écriture
 digitalWrite(CS, HIGH);                // désactivation de la ligne CS
```



```
 // effacement complet la mémoire
 digitalWrite(CS, LOW);                  // activation de la ligne CS
 SPI.transfer(0xC7);                     // effacement
 digitalWrite(CS, HIGH);                 // désactivation de la ligne CS
 // écriture d'une donnée à l'adresse spécifiée
 digitalWrite(CS, LOW);                  // activation de la ligne CS
 SPI.transfer(0x02);                     // écriture d'une donnée à l'adresse
 for (i=0;i<4;i=i+1)
    {
    SPI.transfer(envoi[i]);
    }
 digitalWrite(CS, HIGH);                 // désactivation de la ligne CS
 digitalWrite(CS, LOW);                  // activation de la ligne CS
 SPI.transfer(0x04);                     // blocage écriture
 digitalWrite(CS, HIGH);                 // désactivation de la ligne CS
 delay(1000);
}

void loop()
{
 digitalWrite(CS, LOW);                  // activation de la ligne CS
 SPI.transfer(0x03);                     // lecture d'une donnée à l'adresse
 for (i=0;i<3;i=i+1)
    {
    SPI.transfer(envoi[i]);
    }
 recu[0]=SPI.transfer(0);
 digitalWrite(CS, HIGH);                 // désactivation de la ligne CS
 Serial.println(recu[0],HEX);
}
```

Schéma de câblage :

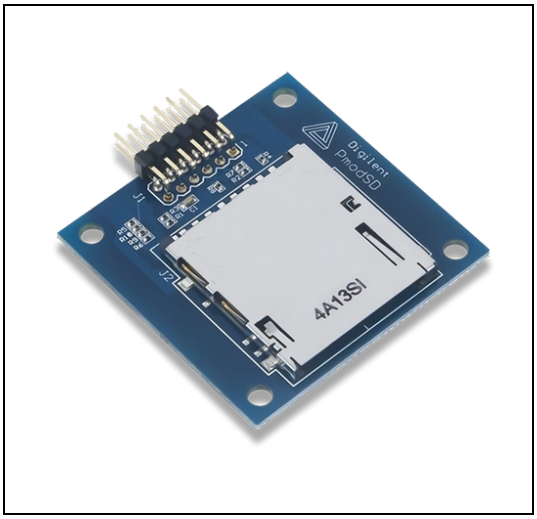


022 : PMODSD : Module lecteur de carte SD

Attention, ce module fonctionne sous 3,3 V

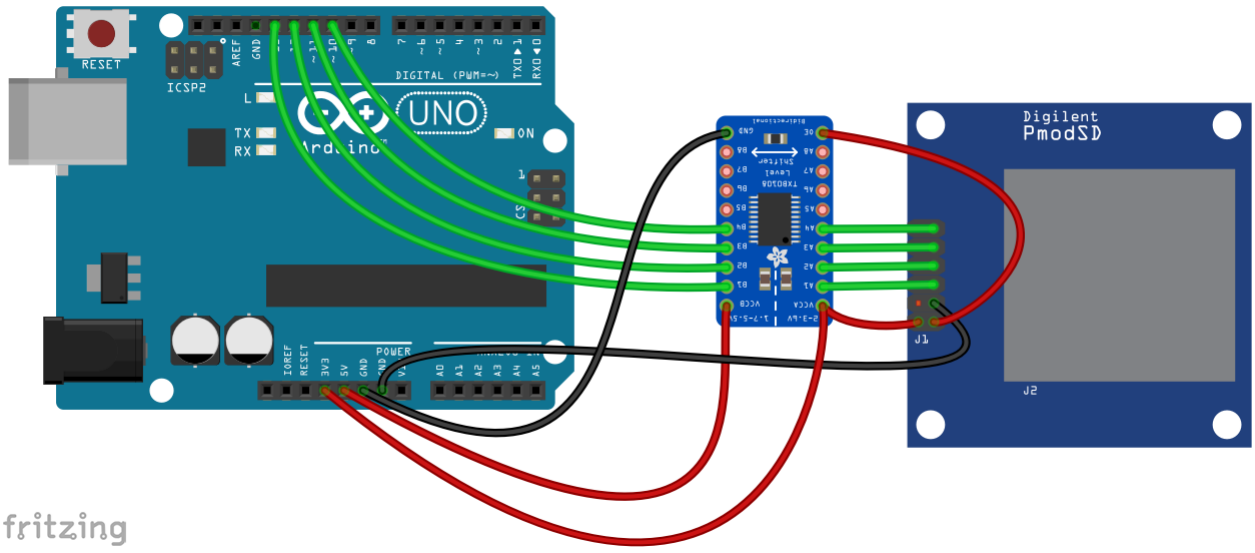
Programme Arduino :

```
/*************************************************************************
*
*    Test du module Pmod lecteur carte SD
*
*************************************************************************
* Description:  Pmod_SD
* Les tensions appliquées sur les entrées A0 et A1 sont enregistrées dans un fichier
* de la carte SD.
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod SD
*       3. Module Adafruit TXB0108
*
*************************************************************************/

#define CS 10                                   // affectation de la broche CS
#include <SPI.h>                                // appel des bibliothèques
#include <SD.h>
int tension;

void setup()
{
 Serial.begin(9600);                            // initialisation de la liaison série
 Serial.println("Initialisation carte SD");
 pinMode(CS, OUTPUT);

 // Initialisation carte SD
 if (!SD.begin(CS))
 {
  Serial.println("    * Carte absente");
  Serial.println("    * Erreur de cablage");
  return;
 }
 Serial.println("Carte initialisee.");
 delay(2000);

}

void loop()
{
```



```
String donnee = "";                          // chaîne de caractères pour le stockage des valeurs à enregistrer
for (int i = 0; i < 2; i++)                  // conversion des tensions et ajout à la chaîne de caractères
{
  tension = analogRead(i);
  donnee+= String(tension);
  if (i < 1)
  {
    donnee += ",";                           // placement d'une virgule entre les deux données
  }
}

File fichier = SD.open("datalog.txt", FILE_WRITE);          // ouverture du fichier datalog.txt
if (fichier)                                 // si le fichier est disponible
{
  fichier.println(donnee);                   // écriture des données dans le fichier
  fichier.close();
  Serial.println(donnee);                    // écriture des données dans le moniteur série
}
else                                         // si le fichier n'existe pas, afficher une erreur
{
  Serial.println("Erreur d'ouverture fichier");
}
delay(1000);                                 // attente d'une seconde entre deux mesures et enregistrement
}
```

Schéma de câblage :

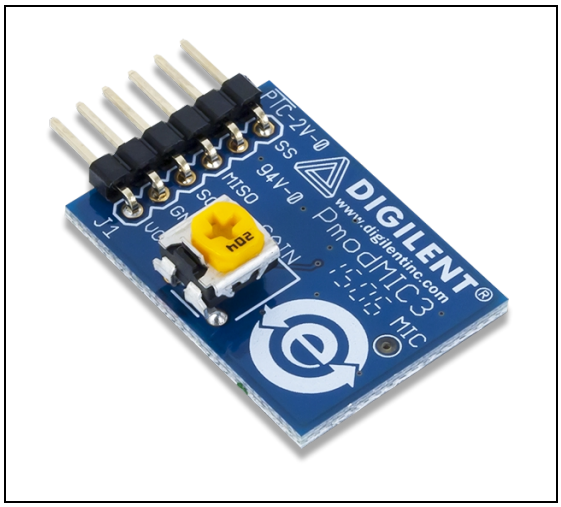


023 : PMODMIC3 : Module micro amplifié

Programme Arduino :

```
/************************************************************************
*
*   Test du module Pmod micro amplifié
*
************************************************************************
* Description:  Pmod_MIC3
* Le son capté par le module est affiché dans le traceur série.
*
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod MIC3
*
* Le câblage est identique au module Pmod AD1
************************************************************************/

#define CS 10                                   // affectation de la broche CS

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

int i;
byte recu[3];                                   // stockage des données du module
int X;
long somme = 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(20);
 for (i=0;i<2;i=i+1)
     {
     recu[i] = SPI.transfer(0);                 // envoi de 2 données pour récupérer la valeur de la conversion sur deux
octets
```

```
      delayMicroseconds(20);
      }
  digitalWrite(CS, HIGH);                  // désactivation de la ligne CS
  X = recu[1];                             // X a un format de 12 bit
  X |= (recu[0] << 8);

   for(int i=0; i<32; i++)                 // élaboration de la courbe de son
   {
      somme = somme + X;
   }

   somme >>= 5;
   Serial.println(somme);                  // affichage dans le traceur série
   delay(10);
}
```



024 : PMODAD1 : Module convertisseur analogique/numérique 12 bit 2 entrées

Programme Arduino :

```
/************************************************************************
*
*   Test du module Pmod convertisseur A/N 12 bits  2 voies
*
************************************************************************
* Description:  Pmod_AD1
* Le résultat de la conversion A/N de la voie A0 est affichée sur le moniteur série.
* Matériel
*       1. Arduino Uno
*       2. Module Pmod AD1
*
************************************************************************/

#define CS 10                                   // affectation de la broche CS

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

int i;
byte recu[3];                                   // stockage des données du module
int X;

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(20);
 for (i=0;i<2;i=i+1)
    {
    recu[i] = SPI.transfer(0);     // envoi de 2 données pour récupérer la valeur de la conversion sur deux octets
    delayMicroseconds(20);
    }
 digitalWrite(CS, HIGH);                        // désactivation de la ligne CS
 X = recu[1];                                   // X a un format de 12 bit
 X |= (recu[0] << 8);
```

```
  for (i=0;i<2;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.println(X);
  delay(100);
}
```

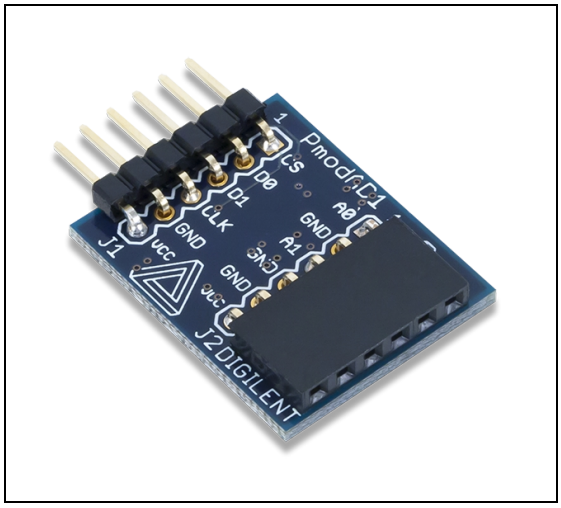
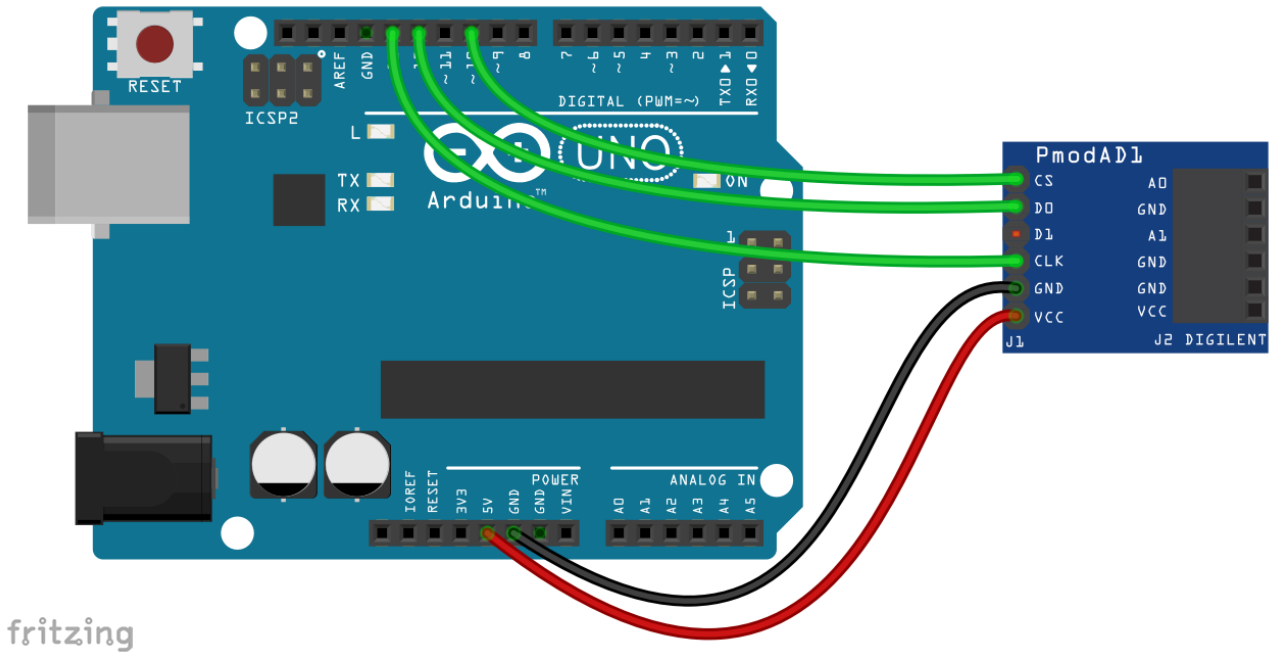


Schéma de câblage :



025 : PMODA2 : Module convertisseur analogique/numérique 12 bit 4 entrées

Programme Arduino :

```
/************************************************************************
*
*   Test du module Pmod convertisseur A/N 12 bits 4 voies
*
************************************************************************
* Description:  Pmod_AD2
* Le résultat de la conversion A/N de la voie A0 est affiché sur le
* moniteur série et sur un afficheur LCD série .
*
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod AD2 (position du cavalier sur V4)
*       3. Module Pmod CLS
*
************************************************************************/

#include <Wire.h>                       // appel de la bibliothèque
#define AD7991_Adresse 0x28             // adresse I2C du module Pmod AD2

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

int MSB;
int LSB;
int valeur;

void setup()
{
 Serial.begin(9600);                    // initialisation de la liaison série
 Wire.begin();                          // initialisation de la liaison I2C
 Init_AD7991();                         // initialisation du module Pmod AD2
 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()
{
 Wire.beginTransmission(AD7991_Adresse);      // lancement de la mesure
 Wire.endTransmission();
```

```
 delay(10);
 Wire.requestFrom(AD7991_Adresse, 2);          // récupération des deux octets MSB et LSB
 if(Wire.available() <=2)
 {
  MSB = Wire.read();
  LSB = Wire.read();
 }
 valeur=MSB<< 8 |LSB ;
 Serial.print("MSB=");                          // affichage dans le moniteur série
 Serial.println(MSB);
 Serial.print("LSB=");
 Serial.println(LSB);
 Serial.print("Valeur=");
 Serial.println(valeur);
 lcd.write("\x1b[j");                           // effacement de l'afficheur
 lcd.print("MSB=");                             // écriture sur l'afficheur
 lcd.print(MSB);
 lcd.print(" LSB=");                            // écriture sur l'afficheur
 lcd.print(LSB);
 lcd.write("\x1b[1;0H");                        // positionnement du curseur 2nde ligne 1ère colonne
 lcd.print("Valeur=");                          // écriture sur l'afficheur
 lcd.print(valeur);
 delay(1000);
}

 // Initialisation du module Pmod AD2
void Init_AD7991(void)
{
 Wire.beginTransmission(AD7991_Adresse);
 Wire.write(0x08);                  // configuraion de la liaison I2C en mode HIGH SPEED
 Wire.write(0x10);                  // configuration du module Pmod AD2 (lecture de V0)
 Wire.endTransmission();
}
```

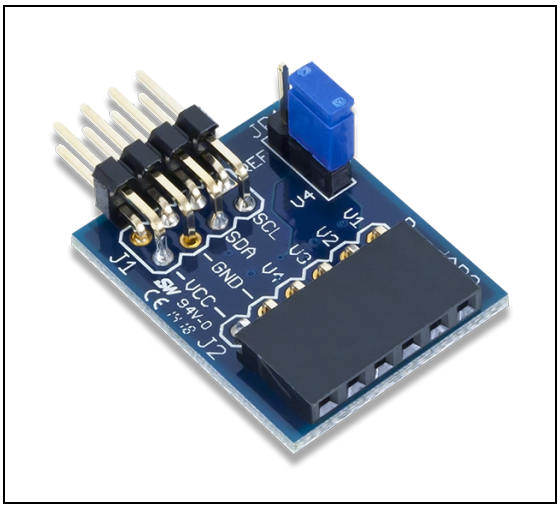
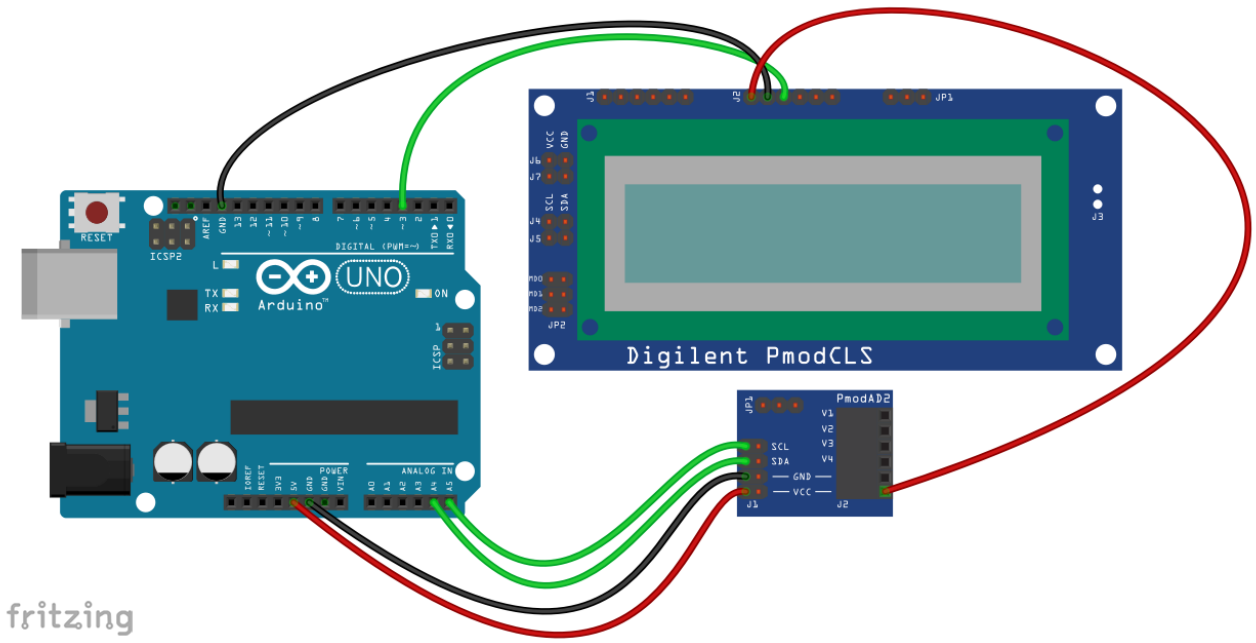


Schéma de câblage :



026 : PMODAD5 : Module convertisseur analogique/numérique 24 bit 8 entrées

Programme Arduino :

```
/************************************************************************
*
*   Test du module Pmod convertisseur A/N 24 bits 8 voies
*
************************************************************************
* Description:  Pmod_AD5
* Le résultat de la conversion A/N de la voie AIN1 est affiché sur le moniteur série.
*
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod AD5 (laisser le cavalier en place et
*       télécharger la bibliothèque https://github.com/annem/AD7193)
*
************************************************************************/

#include <SPI.h>                                  // appel des bibliothèques
#include <AD7193.h>

AD7193 AD7193;                                    // création de l'objet AD7193

unsigned long valeur;
float tension;

void setup()
{
  Serial.begin(9600);                             // initialisation de la liaison série
  Init_AD7193();
}


void loop()
{
 valeur = AD7193.ReadADCChannel(0);               // conversion A/N de la voie 1
 valeur = valeur >> 8;                            // extraction de la valeur
 tension = AD7193.DataToVoltage(valeur);          // récupération de la tension
 Serial.println("");
 Serial.print("Valeur=");
 Serial.print(valeur);
 Serial.print('\t');                              // tabulation
 Serial.print("Tension=");
 Serial.print(tension);
```

```
  Serial.println("V");
}

// Initialisation du module Pmod AD5
void Init_AD7193(void)
{
  AD7193.begin();                          // initialisation du module Pmod AD5
  AD7193.AppendStatusValuetoData();        // configuration du module Pmod AD5
  AD7193.SetPGAGain(1);
  AD7193.SetAveraging(100);
  AD7193.Calibrate();
  AD7193.ReadRegisterMap();
}
```

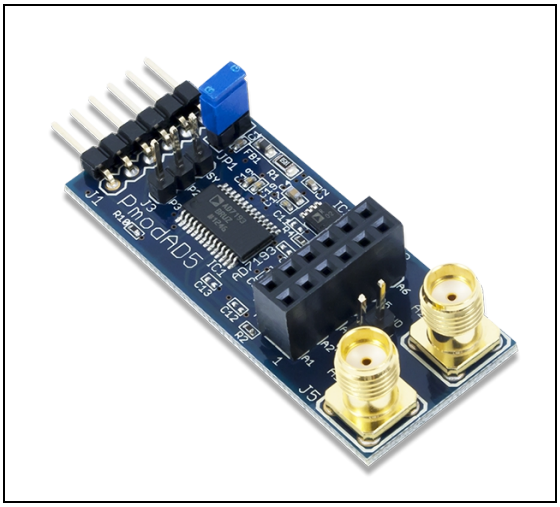
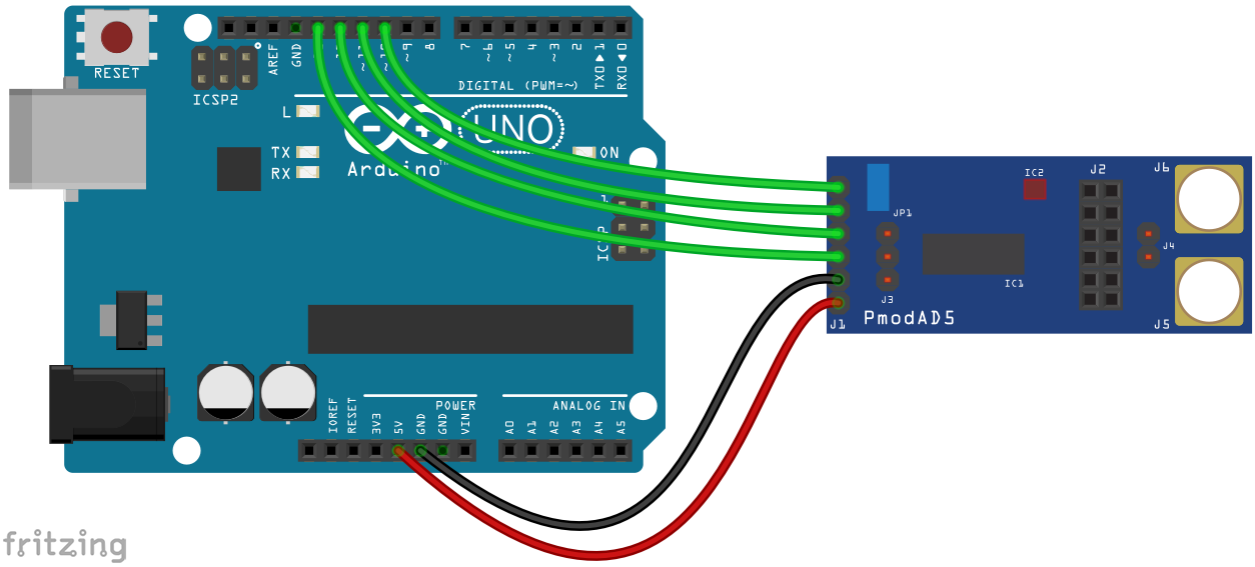


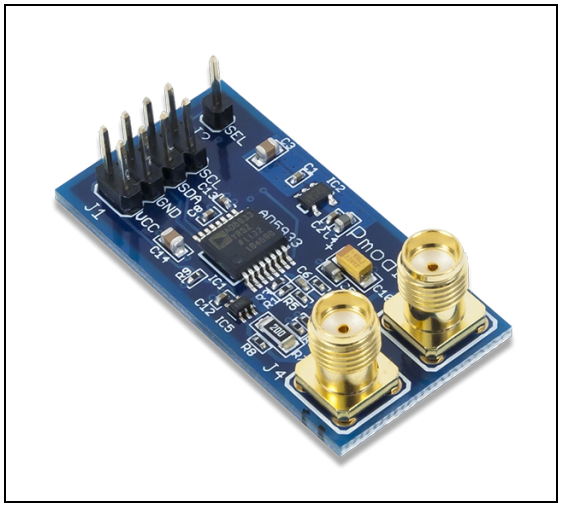
Schéma de câblage :



027 : PMODIA : Module analyseur d'impédance

Programme Arduino :

```
/******************************************************************************
*
*   Test du module Pmod analyseur d'impédance (basé sur le programme de Michael Meli)
*
******************************************************************************
* Description:  Pmod_IA
* Le résultat de la mesure est affiché sur le moniteur série.
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod IA (télécharger le programme et la bibliothèque https://github.com/mjmeli/arduino-ad5933)
*
******************************************************************************/
```



028 : PMODACL : Module accéléromètre 3 axes ADXL345

Attention, ce module fonctionne sous 3,3 V

Programme Arduino :

```
/************************************************************************
*
*    Test du module Pmod accéléromètre ADXL345
*
************************************************************************
* Description:  Pmod_ACL
* Les 3 composantes X, Y  et Z de l'accéléromètre sont affichés
* dans le moniteur série
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod ACL
*
************************************************************************/

// Déclaration des adresses du module
#define ADXL345_Adresse  0x53                  // adresse de l'ADXL345
#define POWER_CTL  0x2D                        // registre Power Control
#define DATA_FORMAT  0x31                      // registre Data Format
#define DATAX0  0x32                           // bit de poids faible axe X
#define DATAX1  0x33                           // bit de poids fort axe X
#define DATAY0  0x34                           // bit de poids faible axe Y
#define DATAY1  0x35                           // bit de poids fort axe Y
#define DATAZ0  0x36                           // bit de poids faible axe Z
#define DATAZ1  0x37                           // bit de poids fort axe Z

// Configuration du module
#define ADXL345_Precision2G   0x00
#define ADXL345_Precision4G   0x01
#define ADXL345_Precision8G   0x02
#define ADXL345_Precision16G  0x03
#define ADXL345_ModeMesure    0x08


// Appel de la bibliothèque
#include <Wire.h>

byte buffer[6];                                // stockage des données du module
int i = 0;
int composante_X;
int composante_Y;
int composante_Z;
```

```
void setup(void)
{
  Serial.begin(9600);                                  // initialisation de la liaison série
  Wire.begin ();                                       // initialisation de la liaison I2C
  Wire.beginTransmission (ADXL345_Adresse);            // configuration du module
  Wire.write (DATA_FORMAT);
  Wire.write (ADXL345_Precision4G);
  Wire.endTransmission ();
  Wire.beginTransmission (ADXL345_Adresse);
  Wire.write (POWER_CTL);
  Wire.write (ADXL345_ModeMesure);
  Wire.endTransmission ();
}

void loop()
{
  Wire.beginTransmission (ADXL345_Adresse);
  Wire.write(DATAX0);
  Wire.endTransmission ();
  Wire.beginTransmission (ADXL345_Adresse);
  Wire.requestFrom(ADXL345_Adresse, 6);                // récupération des 6 composantes
  i=0;
  while(Wire.available())
  {
   buffer[i] = Wire.read();
   i++;
  }
  Wire.endTransmission();
  composante_X=(buffer[1] << 8) | buffer[0];           // élaboration des 3 composantes
  composante_Y=(buffer[3] << 8) | buffer[2];
  composante_Z=(buffer[5] << 8) | buffer[4];
  Serial.print("X=");                                  // affichage des composantes dans le moniteur série
  Serial.print (composante_X);
  Serial.print('\t');                                  // tabulation
  Serial.print("Y=");
  Serial.print (composante_Y);
  Serial.print('\t');
  Serial.print("Z=");
  Serial.print (composante_Z);
  Serial.println("");
  delay(500);
}
```

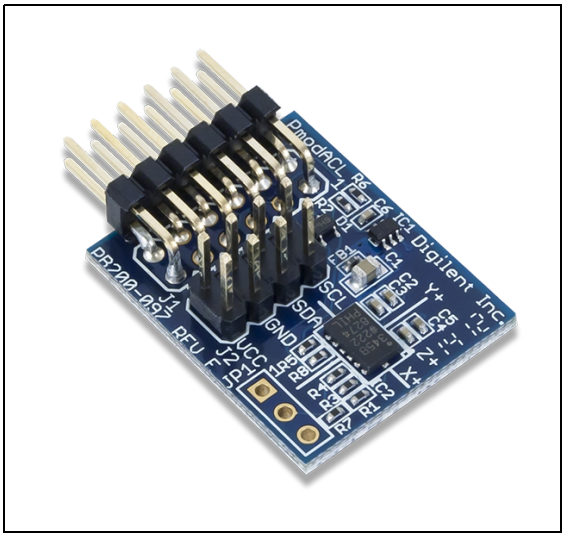
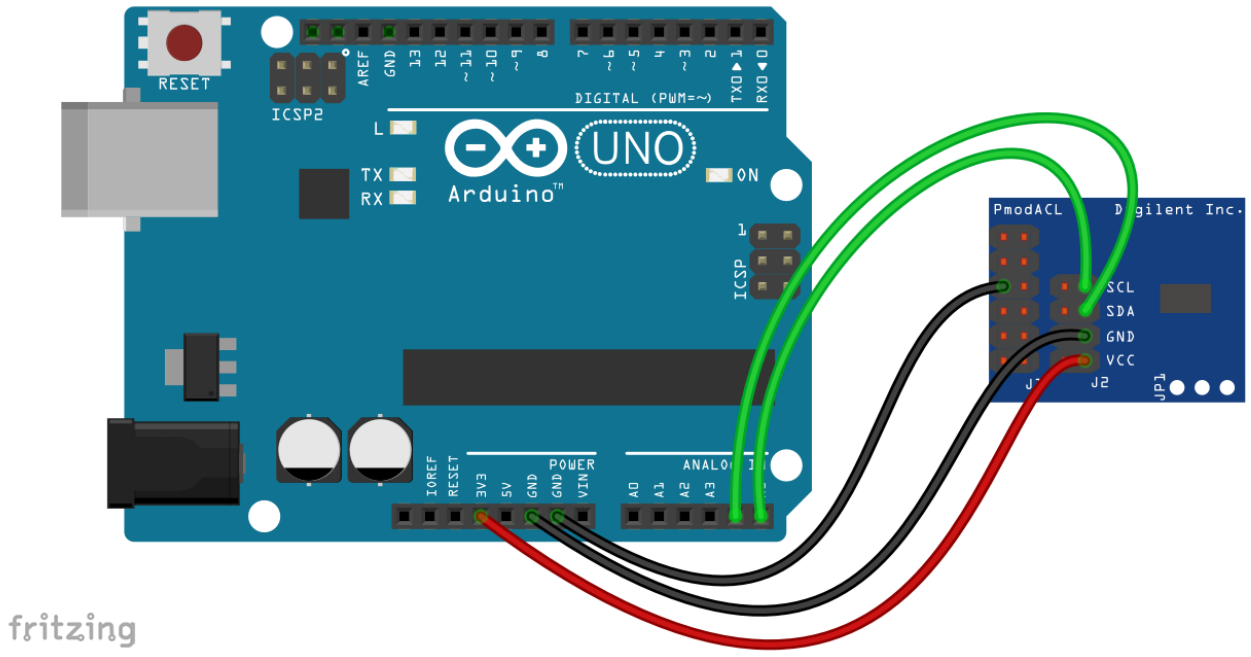


Schéma de câblage :



029 : PMODCMPS : Module boussole

Attention, ce module fonctionne sous 3,3 V

Programme Arduino :

```
/************************************************************************
*
*    Test du module Pmod boussole
*
************************************************************************
* Description:  Pmod_CMPS
* Les 3 composantes (X, Y, Z) du champ magnétique puis son module et son angle
* sont affichés dans le moniteur série
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod CMPS
*
************************************************************************/
#include <Wire.h>                                   // appel de la bibliothèque
#define HMC5803L_Addresse 0x1E                      // adresse I2C du module

double composante_X;
double composante_Y;
double composante_Z;
double module;
double angle;

void setup()
{
 Serial.begin(9600);
 Wire.begin();                                      // initialisation du bus I2C
 Init_HMC5803L();                                   // initialisation du module boussole
}

void loop()
{
 Wire.beginTransmission(HMC5803L_Addresse);         // lancement de la mesure
 Wire.write(0x02);
 Wire.write(0x01);
 Wire.endTransmission();
 delay(10);
 Wire.requestFrom(HMC5803L_Addresse, 6);            // récupération des composantes
 if(Wire.available() <=6)
 {
  composante_X = Wire.read() << 8 | Wire.read();
  composante_Y = Wire.read() << 8 | Wire.read();
```

```
   composante_Z = Wire.read() << 8 | Wire.read();
  }
  Serial.print("X=");
  Serial.print (composante_X);
  Serial.print('\t');
  Serial.print("Y=");
  Serial.print (composante_Y);
  Serial.print('\t');
  Serial.print("Z=");
  Serial.print (composante_Z);
  Serial.print('\t');
  // Calcul du module
  module = sqrt(composante_X * composante_X + composante_Y * composante_Y + composante_Z *
composante_Z);
  Serial.print("M=");
  Serial.print(module);
  Serial.print('\t');
  //Calcul de l'angle
  angle= atan2(composante_Y,composante_X) * (180 /
3.14159265);       // angle en degrés
  if (angle<0) {angle=angle+360;}
  Serial.print("Angle=");
  Serial.println(angle);
 delay(1000);
}
// Initialisation du module HMC5803L
void Init_HMC5803L(void)
{
  Wire.beginTransmission(HMC5803L_Addresse);
  Wire.write(0x00);
  Wire.write(0x70);
  Wire.write(0x01);
  Wire.write(0xA0);
  Wire.endTransmission();
}
```

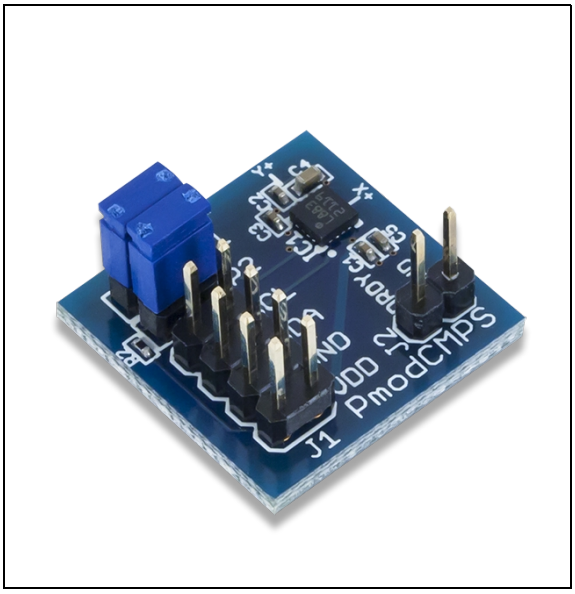
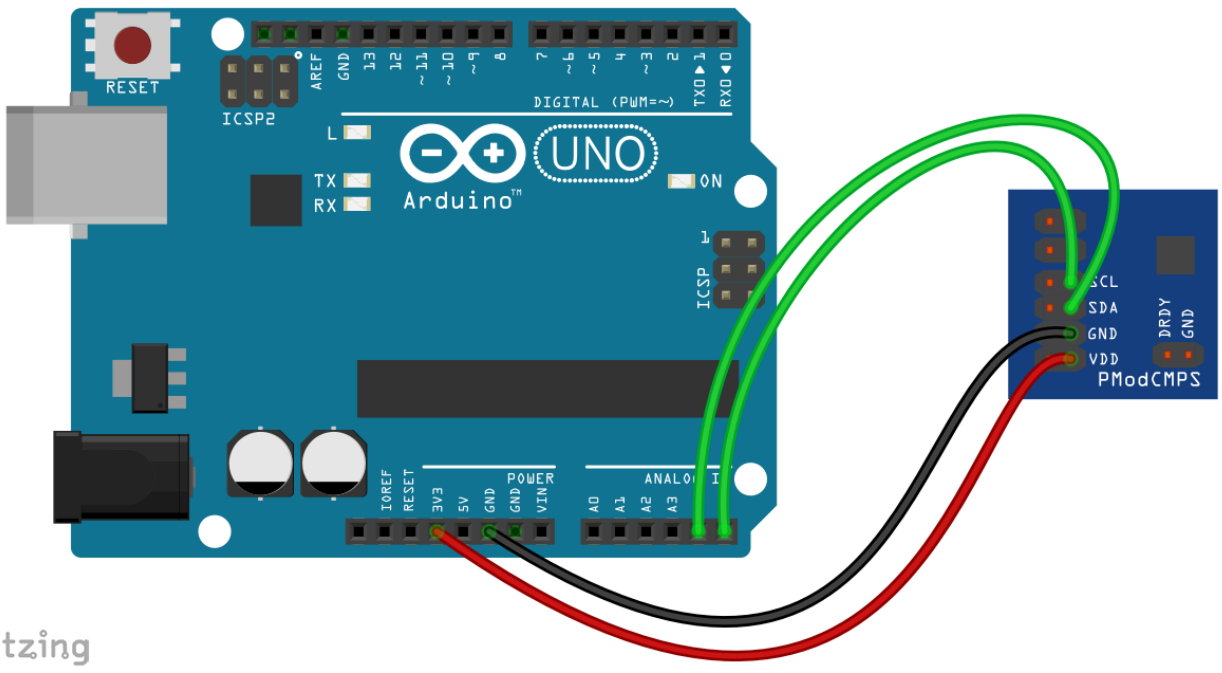


Schéma de câblage :



030 : PMODGYRO : Module gyroscope 3 axes L3G4200D

Attention, ce module fonctionne sous 3,3 V

Programme Arduino :

```
/************************************************************************
*
*    Test du module Pmod gyroscope 3 axes
*
************************************************************************
* Description:  Pmod_GYRO
* Les 3 composantes X, Y  et Z du gyroscope sont affichés
* dans le moniteur série
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod GYRO
*
************************************************************************/

// Déclaration des adresses du module
#define L3G4200D_Adresse  0x69                  // adresse du L3G4200D
#define CTRL_REG1 0x20
#define CTRL_REG2 0x21
#define CTRL_REG3 0x22
#define CTRL_REG4 0x23
#define CTRL_REG5 0x24
#define REGISTRE_MSB_X  0x29                    // bit de poids fort axe X
#define REGISTRE_LSB_X  0x28                    // bit de poids faible axe X
#define REGISTRE_MSB_Y  0x2B                    // bit de poids fort axe Y
#define REGISTRE_LSB_Y  0x2A                    // bit de poids faible axe Y
#define REGISTRE_MSB_Z  0x2D                    // bit de poids fort axe Z
#define REGISTRE_LSB_Z  0x2C                    // bit de poids faible axe Z


// Appel de la bibliothèque
#include <Wire.h>

int composante_X;
int composante_Y;
int composante_Z;

byte composante_MSB_X;
byte composante_LSB_X;
byte composante_MSB_Y;
byte composante_LSB_Y;
byte composante_MSB_Z;
```

```
byte composante_LSB_Z;



void setup(void)
{
  Serial.begin(9600);                    // initialisation de la liaison série
  Wire.begin();                          // initialisation de la liaison I2C
  Init_L3G4200D(2000);
  delay(1000);
}

void loop()
{
 Lecture_Module();

 Serial.print("X=");                     // affichage des composantes dans le moniteur série
 Serial.print (composante_X);
 Serial.print('\t');                    // tabulation
 Serial.print("Y=");
 Serial.print (composante_Y);
 Serial.print('\t');
 Serial.print("Z=");
 Serial.print (composante_Z);
 Serial.println("");
 delay(500);
}

// Initialisation du module L3G4200D
void Init_L3G4200D(int echelle)
{
 Wire.beginTransmission(L3G4200D_Adresse);
 Wire.write(CTRL_REG1);
 Wire.write(0b00001111);
 Wire.endTransmission();
  Wire.beginTransmission(L3G4200D_Adresse);
 Wire.write(CTRL_REG2);
 Wire.write(0b00000000);
 Wire.endTransmission();
 Wire.beginTransmission(L3G4200D_Adresse);
 Wire.write(CTRL_REG3);
 Wire.write(0b00001000);
 Wire.endTransmission();
 if (echelle==250)
 {
  Wire.beginTransmission(L3G4200D_Adresse);
  Wire.write(CTRL_REG4);
  Wire.write(0b00000000);
  Wire.endTransmission();
 }
```

```
 if (echelle==500)
 {
  Wire.beginTransmission(L3G4200D_Adresse);
  Wire.write(CTRL_REG4);
  Wire.write(0b00010000);
  Wire.endTransmission();
 }
 else
 {
  Wire.beginTransmission(L3G4200D_Adresse);
  Wire.write(CTRL_REG4);
  Wire.write(0b00110000);
  Wire.endTransmission();
 }
 Wire.beginTransmission(L3G4200D_Adresse);
 Wire.write(CTRL_REG5);
 Wire.write(0b00000000);
 Wire.endTransmission();
}

// Lecture des valeurs X, Y et Z
 void Lecture_Module()
 {
 composante_MSB_X=Lecture_Registre(REGISTRE_MSB_X);          // lecture MSB_X
 composante_LSB_X=Lecture_Registre(REGISTRE_LSB_X);          // lecture LSB_X
 composante_X=((composante_MSB_X << 8) | composante_LSB_X);  // reconstitution composante_X
 composante_MSB_Y=Lecture_Registre(REGISTRE_MSB_Y);          // lecture MSB_Y
 composante_LSB_Y=Lecture_Registre(REGISTRE_LSB_Y);          // lecture LSB_Y
 composante_Y=((composante_MSB_Y << 8) | composante_LSB_Y);  // reconstitution composante_Y
 composante_MSB_Z=Lecture_Registre(REGISTRE_MSB_Z);          // lecture MSB_Z
 composante_LSB_Z=Lecture_Registre(REGISTRE_LSB_Z);          // lecture LSB_Z
 composante_Z=((composante_MSB_Z << 8) | composante_LSB_Z);  // reconstitution composante_Z
 }

// Lecture d'un registre du module L3G4200D
int Lecture_Registre(byte registre)
{
int v;
Wire.beginTransmission(L3G4200D_Adresse);
Wire.write(registre);
Wire.endTransmission();
Wire.requestFrom(L3G4200D_Adresse, 1);
while(!Wire.available());
v = Wire.read();
return v;
}
```

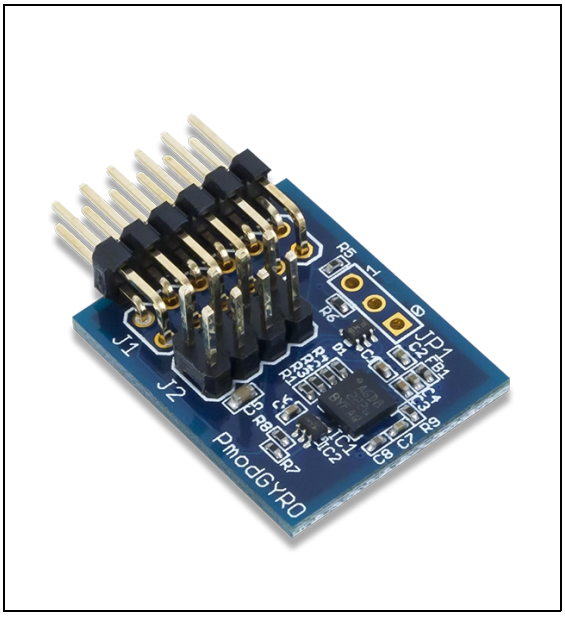
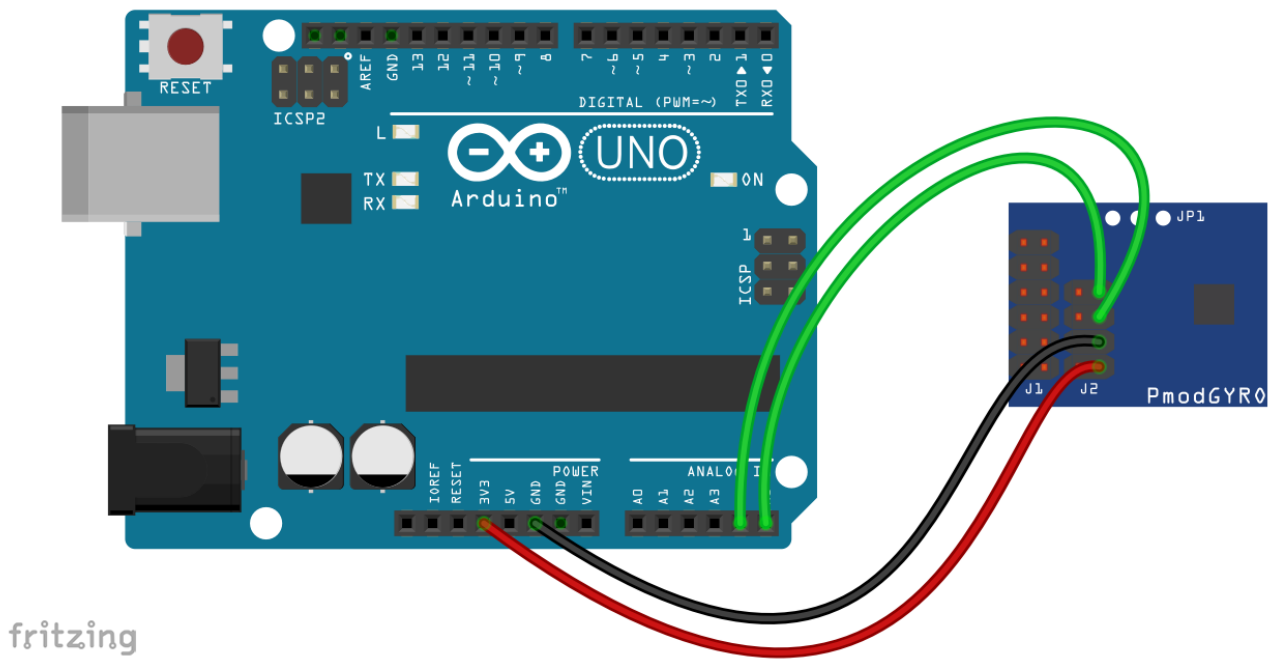


Schéma de câblage :



031 : PMODOC1 : Module 4 collecteurs ouverts

Programme Arduino :

```
/*************************************************************************
*
*    Test du module Pmod 4 collecteurs ouverts
*
*************************************************************************
* Description:  Pmod_OC1
* Les led s'allument et s'éteignent successivement les unes après les autres.
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod 4 collecteurs ouverts (laisser le cavalier en place)
*       3. 4 led avec résistance 330 ohm
*
*************************************************************************/

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

void loop()
{
 for(int i = 2; i < 6; i++)
 {
   digitalWrite(i,HIGH);             // saturation des transistors
   delay(500);
   digitalWrite(i,LOW);              // blocage des transistors
 }
}
```

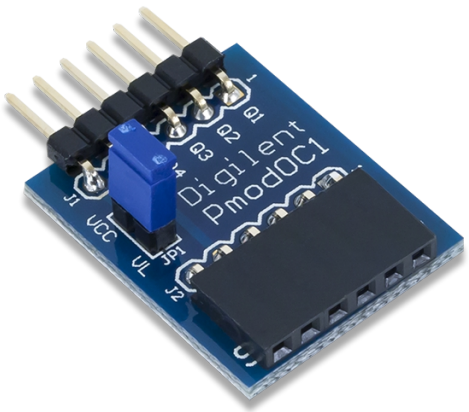
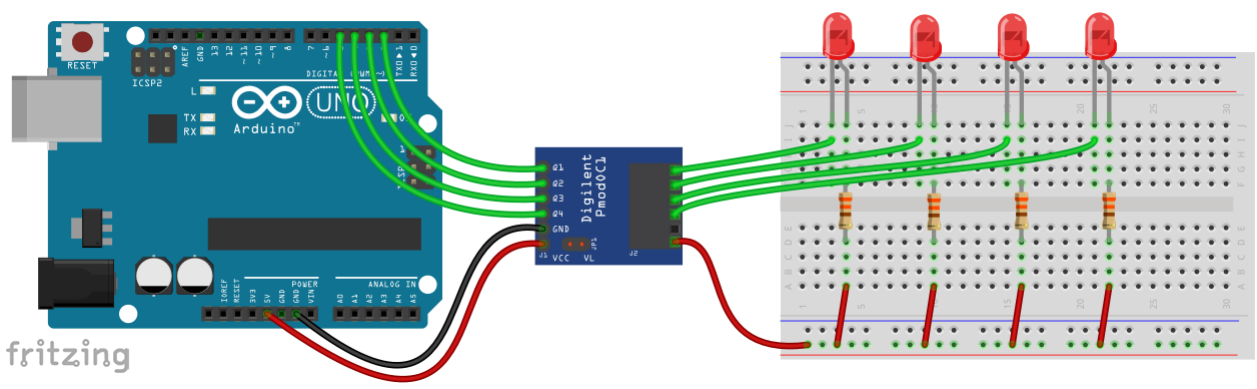


Schéma de câblage :



032 : PMODOD1 : Module FET de puissance

Programme Arduino :

```
/*****************************************************************************
*
*    Test du module Pmod FET de puissance
*
*****************************************************************************
* Description:  Pmod_OD1
* Un moteur accélère progressivement pendant 2.5 s puis reste en régime
* permanent pendant 5 s et décélère pendant 2.5 s puis reste arrêté
* pendant 5s.
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod FET de puissance (laisser le cavalier en place)
*       3. Moteur à courant continu 5 V (modèle à faible consommation)
*
*****************************************************************************/
#define mot 3
int i;

void setup()
{
 pinMode(mot,OUTPUT);          // broche 3 configurée en sortie (PWM)
}

void loop()
{
 for(i = 0; i < 256; i++)
 {
  analogWrite(mot,i);         // le moteur accélère
  delay(10);
 }
 delay(5000);
 for(i = 0; i < 256; i++)
 {
  analogWrite(mot,255-i);         // le moteur décèlere
  delay(10);
 }
 delay(5000);
}
```

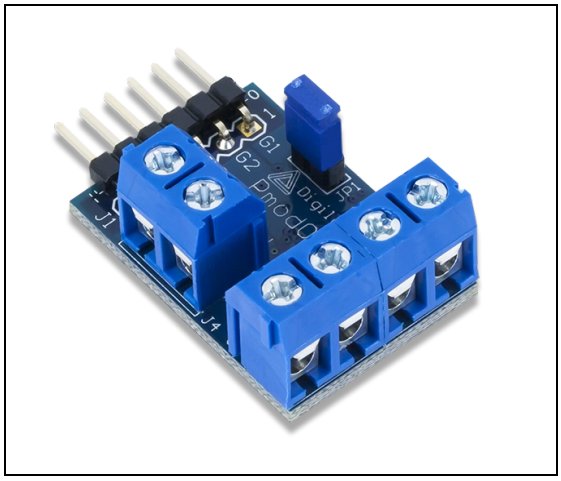
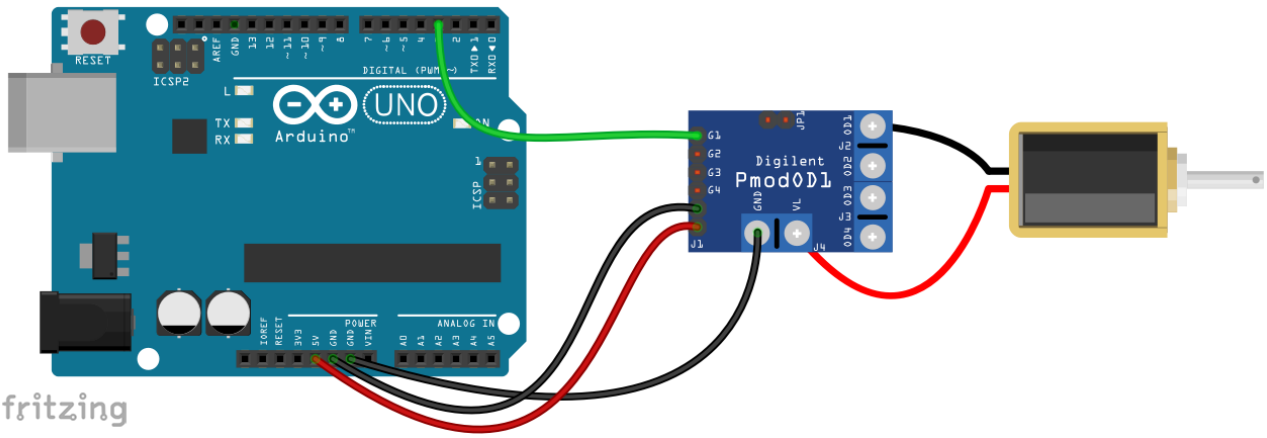
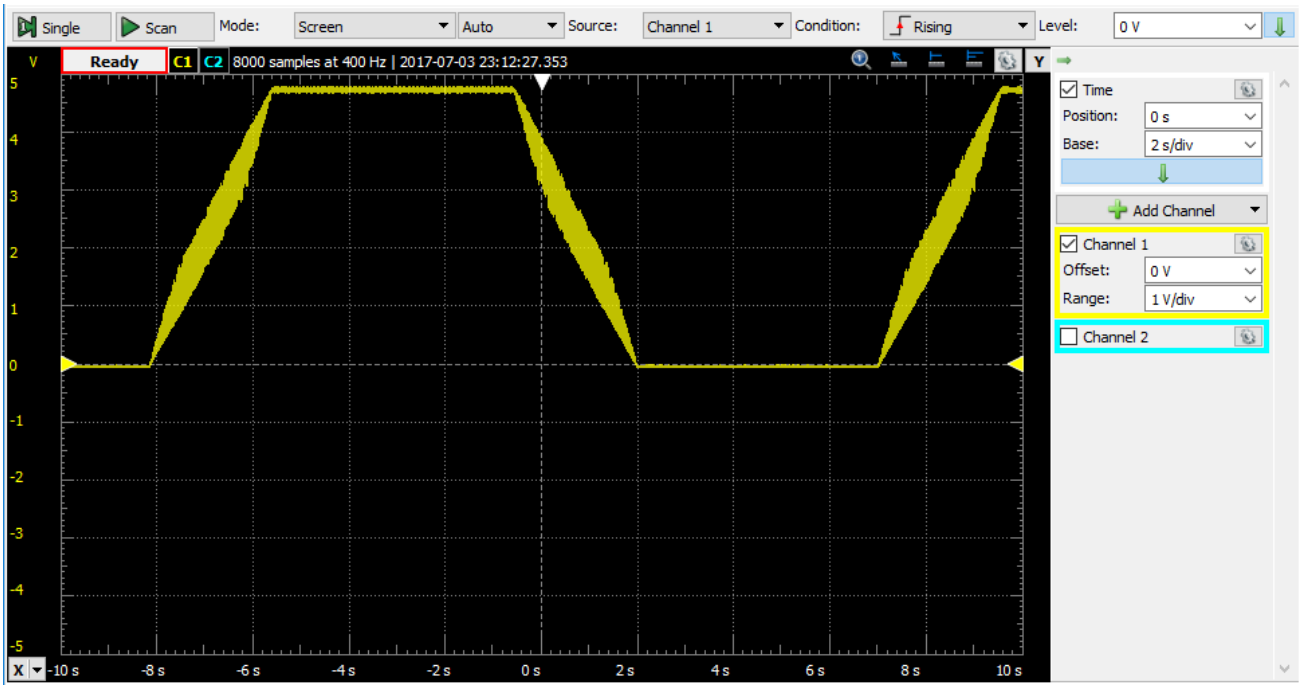


Schéma de câblage :



Oscillogramme :

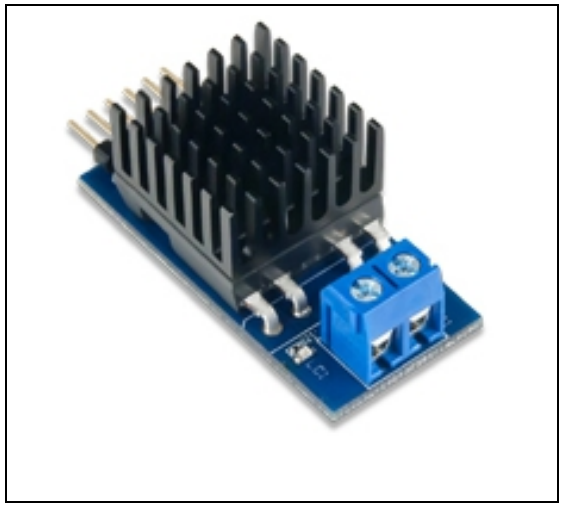


033 : PMODSSR : Module relais statique

Programme Arduino :

```
/*************************************************************************
*
*    Test du module Pmod relais statique
*
*************************************************************************
* Description:  Pmod_SSR
* Une charge est commandée depuis le moniteur série (0=OFF et 1=ON)
* et via le module pouvant commuter une puissance importante.
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod relais statique (branchement identique au module Pmod OD1)
*
*************************************************************************/
#define relais 3
char octetReception=0;

void setup()
{
 Serial.begin(9600);                    // initialisation du moniteur série
 pinMode(relais,OUTPUT);                // broche 2 configurée en sortie
}

void loop()
{
 Serial.println("Relais actif:Taper 1 ou Relais inactif:Taper 0");

  while (Serial.available()==0);            // attente réception d'un octet
  {
   octetReception=Serial.read();          // lecture de l'octet reçu et stockage dans la variable
   switch (octetReception)
   {
    case '0':                             // cas 0
     {
     digitalWrite(relais,LOW);            // relais inactif
     Serial.println("Relais inactif");
     Serial.println();
     break;
     }

    case '1':                             // cas 1
     {
```

```
    digitalWrite(relais,HIGH);              // relais actif
    Serial.println("Relais inactif");
    Serial.println();
    break;
    }

   default:                                 // cas par défaut
    {
   Serial.println("Commande non reconnue");
    }
  }
 }
}
```



034 : PMODACL2 : Module accéléromètre 3 axes ADXL362

Attention, ce module fonctionne sous 3,3 V

Programme Arduino :

```
/************************************************************************
*
*   Test du module Pmod accéléromètre ADXL362
*
************************************************************************
* Description:  Pmod_ACL2
* Les 3 composantes X, Y  et Z de l'accéléromètre sont affichés
* dans le moniteur série
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod ACL2 (télécharger la bibliothèque https://github.com/annem/ADXL362)
*       3. Module Adafruit TXB0108
************************************************************************/

// Appel des bibliothèques
#include <SPI.h>
#include <ADXL362.h>

ADXL362 accelerometre;                    // création de l'objet

int composante_X;
int composante_Y;
int composante_Z;
int temperature;

void setup(void)
{
  Serial.begin(9600);                     // initialisation de la liaison série
  accelerometre.begin(10);                // initialisation de l'accéléromètre
  accelerometre.beginMeasure();
}

void loop()
{
 // acquisition des données de l'accéléromètre
 accelerometre.readXYZTData(composante_X, composante_Y, composante_Z, temperature);
 Serial.print("composante_X=");
 Serial.print(composante_X);
 Serial.print(" composante_Y=");
 Serial.print(composante_Y);
 Serial.print(" composante_Z=");
```

```
  Serial.println(composante_Z);
  delay(500);
}
```

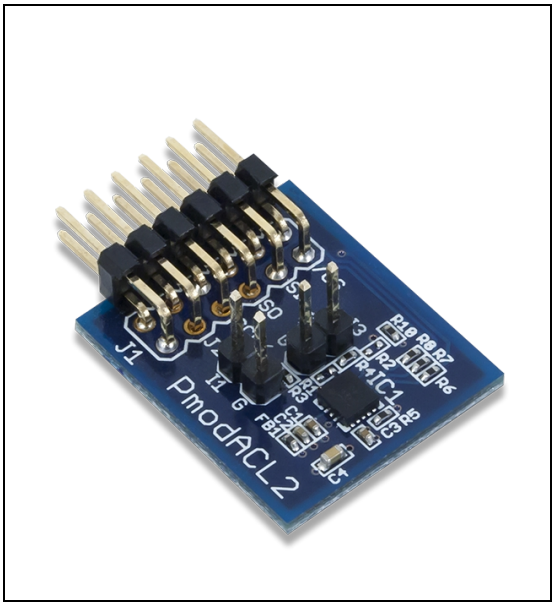
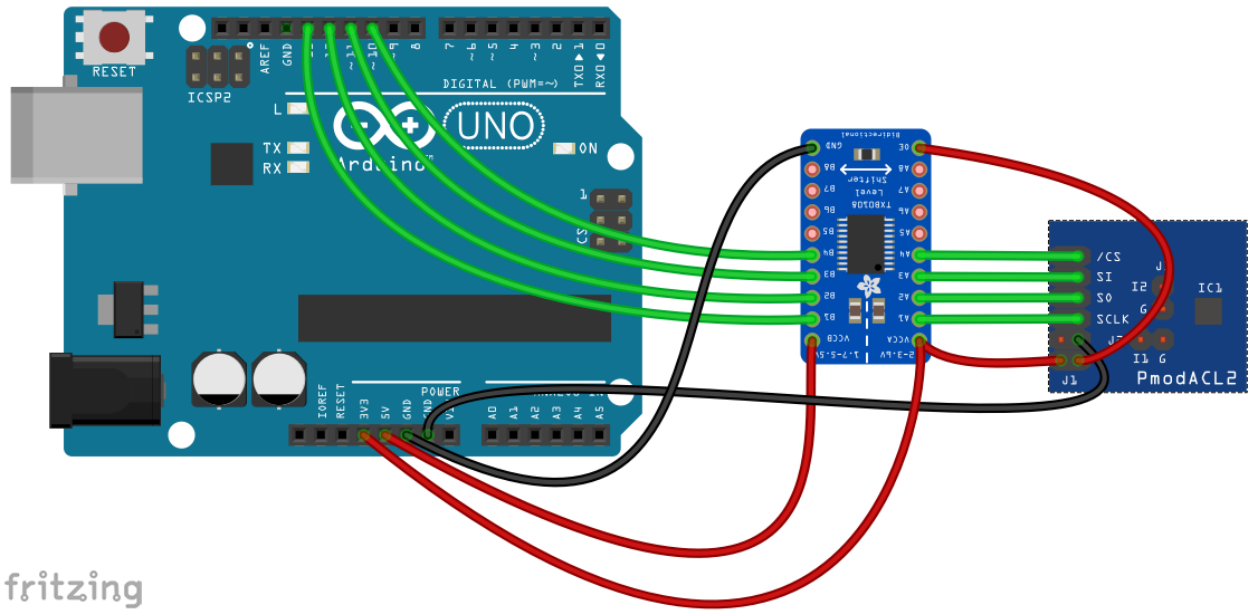


Schéma de câblage :



035 : PMODALS : Module capteur de lumière

Programme Arduino :

```
/************************************************************************
*
*    Test du module Pmod capteur de lumière
*
************************************************************************
* Description:  Pmod_ALS
* La valeur de la quantité de lumière est affichée sur le moniteur série.
* (Le capteur est saturé à une valeur de 125 environ)
*
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod ALS
*
************************************************************************/

#define CS 10                                   // affectation de la broche CS

#include <SPI.h>                                // appel de la bibliothèque
int i;
int recu[2];                                    // stockage des données du module
int lumiere;

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
 for (i=0;i<2;i=i+1)
     {
     recu[i] = SPI.transfer(0);                 // acquisition des 2 octets de données
     }
 digitalWrite(CS, HIGH);                        // désactivation de la ligne CS
 for (i=0;i<2;i=i+1)                            // écriture dans le moniteur série
     {
```

```
    Serial.print("i");
    Serial.print(i);
    Serial.print("=");
    Serial.println(recu[i]);
    }
  lumiere=(recu[0]<<3)|(recu[1]>>4);     //reconstitution de la variable lumiere sur 8 bits
  Serial.print("Lumiere=");
  Serial.println(lumiere);
  delay(1000);
}
```

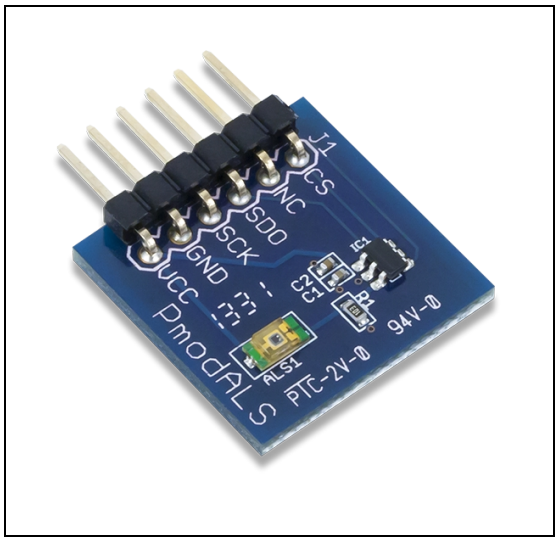
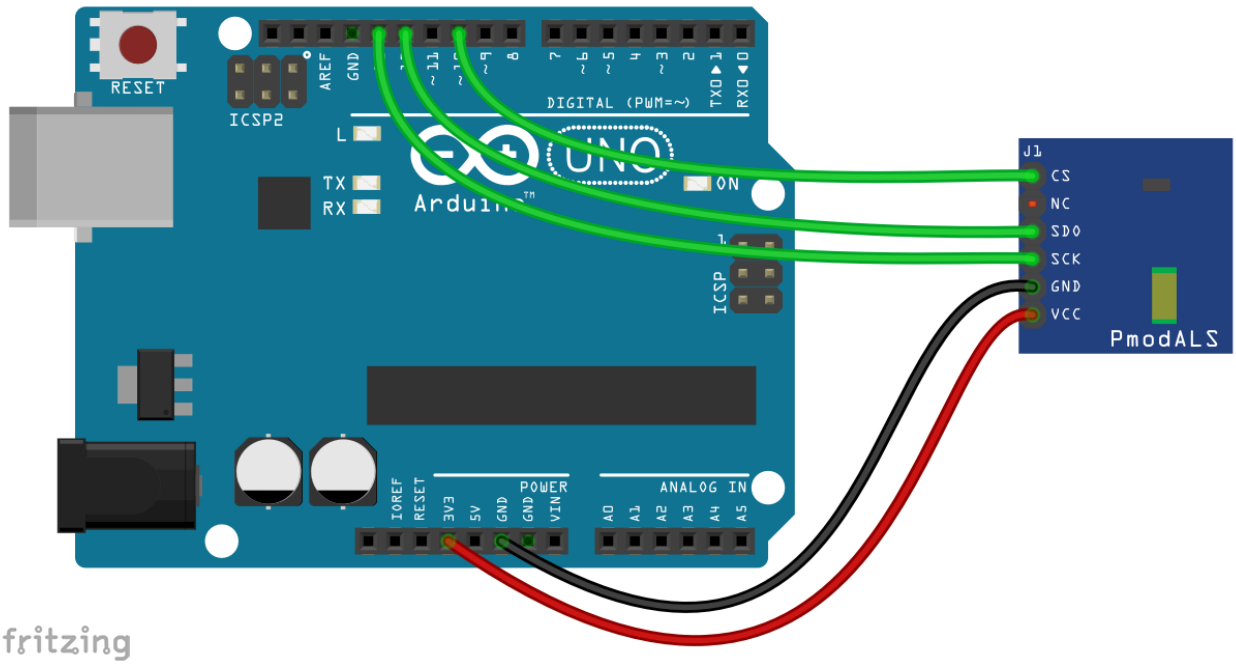


Schéma de câblage :

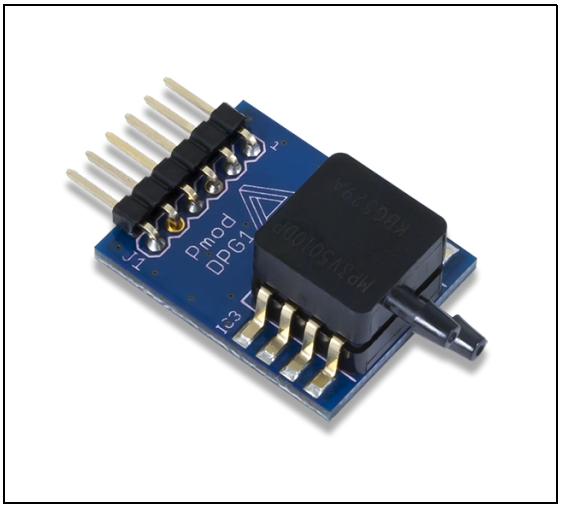


036 : PMODDPG1 : Module capteur de pression différentiel

Programme Arduino :

```
/************************************************************************
*
*   Test du module Pmod capteur de pression différentiel
*
************************************************************************
* Description:  Pmod_DPG1
* La pression est affichée sur le moniteur série.
*
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod DPG1
*       3. Module Adafruit TXB0108
*
* Câblage
*       Module<----------> TXB0108 <-----------> Arduino
*       J1 broche 6                 3.3 V
*       J1 broche 5                 GND
*       J1 broche 4                 13
*       J1 broche 3                 12
*       J1 broche 1                 10
*
************************************************************************/

#define CS 10                                   // affectation de la broche CS

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

int i;
byte recu[3];                                   // stockage des données du module
int valeur;
float pression;

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(20);
 for (i=0;i<2;i=i+1)
    {
    recu[i] = SPI.transfer(0);     // envoi de 2 données pour récupérer la valeur de la conversion sur deux octets
    delayMicroseconds(20);
    }
 digitalWrite(CS, HIGH);                                 // désactivation de la ligne CS
 valeur = (recu[0] << 8|recu[1]);
 for (i=0;i<2;i=i+1)                                     // écriture dans le moniteur série
    {
    Serial.print("i");
    Serial.print(i);
    Serial.print("=");
    Serial.print(recu[i]);
    Serial.print('\t');                                  // tabulation
    }
 pression=(valeur/4096.0-0.08)/0.09;                     // formule donnée par la documentation
 Serial.print("Pression=");
 Serial.print(pression);
 Serial.println(" kPa");
 delay(1000);
}
```



Programme Arduino :

```
/************************************************************************
*
*    Test du module Pmod double pont en H
*
************************************************************************
* Description:  Pmod_DHB1
* La commande du moteur est réalisée à l'aide du moniteur série.
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod HB1 (positionner les cavaliers JP1 et JP2 sur 1)
*
************************************************************************/
// Affectation des broches
#define DIRECTION 2
#define VALIDATION 3

char sens=0;
char recu[3];
boolean dir;
int vitesse=0;
int vitesse_moteur;

void setup()
{
 Serial.begin(9600);                                    // initialisation du moniteur série
 pinMode(DIRECTION,OUTPUT);                             // configuration des broches
 pinMode(VALIDATION,OUTPUT);
}

void loop()
{
 Serial.println("Rentrer le sens (A: avance ou R:recule) puis Envoyer");
 while (Serial.available()==0);                          // attente du paramètre sens
 sens=Serial.read();                                    // réception du paramètre sens
 Serial.println("Rentrer la vitesse (000..100) puis Envoyer");
 while (Serial.available()==0);                         // attente du paramètre vitesse
    for (int i=0;i<3;i++)                               // réception du paramètre vitesse
    {
    recu[i]=Serial.read();
    delay(10);
    }
```

```
 vitesse=(recu[0]-48)*100+(recu[1]-48)*10+recu[2]-48;          // reconstitution de la variable vitesse
 vitesse_moteur=map(vitesse,0,100,0,255);                      // changement d'échelle pour la variable vitesse du
moteur (0 à 255)
 Serial.print("Le moteur tourne dans le sens ");
 if (sens==65)                                                 // code ASCII de A
  {
    dir=HIGH;
    Serial.print("horaire");
  }
 if (sens==82)                                                 // code ASCII de R
  {
    dir=LOW;
    Serial.print("anti-horaire");
  }
 Serial.print(" avec une vitesse de ");
 Serial.print(vitesse);
 Serial.println(" %");
 Serial.println(" ");
 digitalWrite(DIRECTION,dir);
 analogWrite(VALIDATION,vitesse_moteur);
 delay(100);
}
```

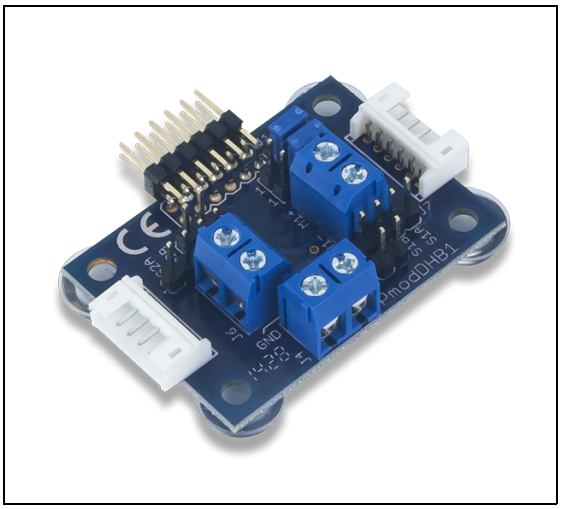
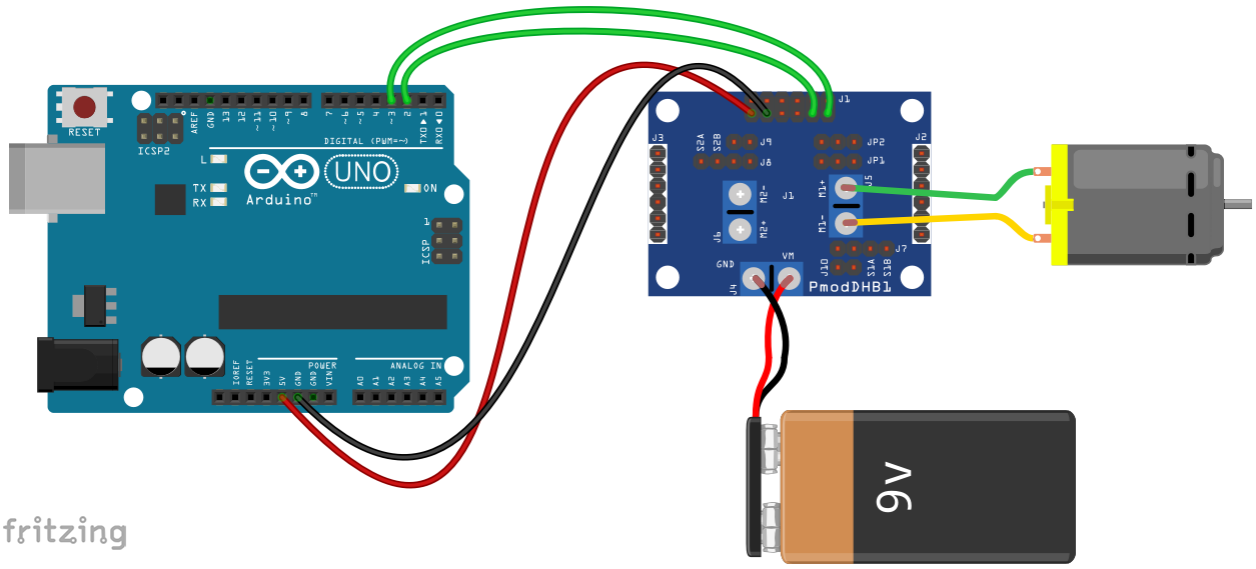


Schéma de câblage :



038 : PMODHB3 : Module commande moteur cc HB3

Programme Arduino :

```
/************************************************************************
*
*   Test du module Pmod commande moteur cc HB3
*
************************************************************************
* Description:  Pmod_HB3
* L'interrupteur SW1 commande l'arrêt ou la rotation du moteur.
* L'interrupteur SW2 commande le sens de rotation du moteur.
* L'état du moteur est affiché sur le moniteur série.
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod HB3
*       3. Module Pmod SWT
*
************************************************************************/
// Affectation des broches
#define DIRECTION 2
#define VALIDATION 3
#define SWT_1 4
#define SWT_2 5

boolean inter_1;
boolean inter_2;

void setup()
{
 Serial.begin(9600);                    // initialisation du moniteur série
 pinMode(DIRECTION,OUTPUT);             // configuration des broches
 pinMode VALIDATION,OUTPUT);
 pinMode(SWT_1,INPUT);
 pinMode(SWT_2,INPUT);
}

void loop()
{
 inter_1=digitalRead(SWT_1);            // lecture de l'interrupteur SW1
 inter_2=digitalRead(SWT_2);            // lecture de l'interrupteur SW2
 if (inter_1==HIGH)                     // arrêt du moteur
  {
   digitalWrite(DIRECTION,LOW);
   digitalWrite(VALIDATION,LOW);
```

```
   Serial.println("Le moteur est arrete");     // affichage dans le moniteur série
  }
 else                                          // le moteur tourne
  {
   if (inter_2==HIGH)
   {
    digitalWrite(DIRECTION,HIGH);
    digitalWrite(VALIDATION,HIGH);
    Serial.println("Le moteur tourne dans le sens horaire");
   }
   else
   {
    digitalWrite(DIRECTION,LOW);
    digitalWrite(VALIDATION,HIGH);
    Serial.println("Le moteur tourne dans le sens anti-horaire");
   }
  }
}
```

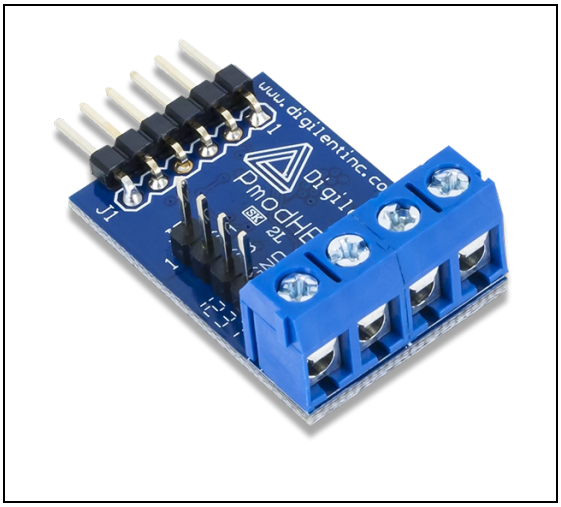
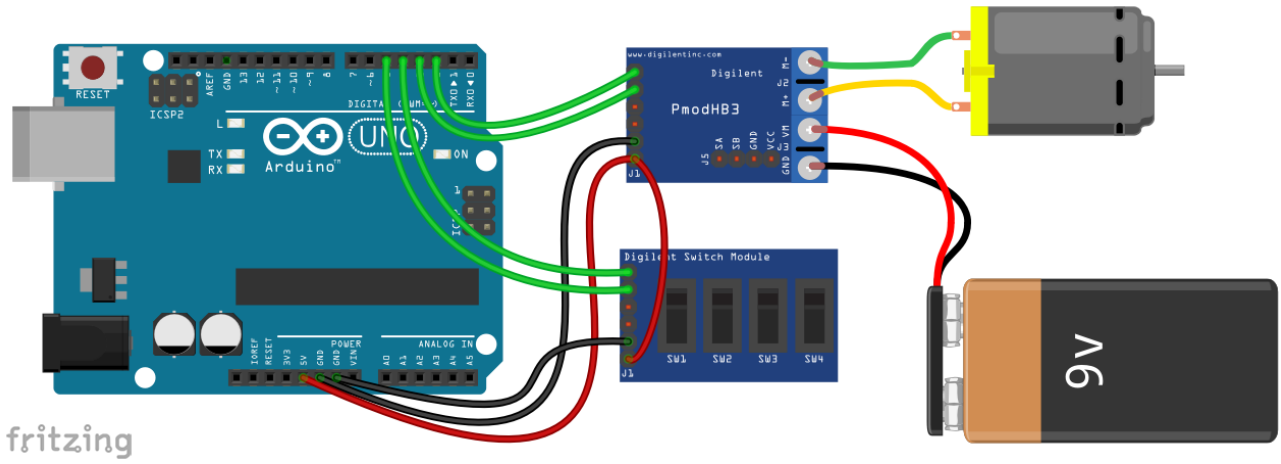


Schéma de câblage :



Programme Arduino :

```
/************************************************************************
*
*    Test du module Pmod commande moteur cc HB5
*
************************************************************************
* Description:  Pmod_HB5
* Le bouton poussoir BTN0 commande l'arrêt du moteur.
* Le bouton poussoir BTN1 commande l'augmentation de la vitesse du moteur.
* Le bouton poussoir BTN2 commande la diminution de la vitesse du moteur.
* La vitesse du moteur est affichée sur le moniteur série.
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod HB5
*       3. Module Pmod BTN
*
************************************************************************/
// Affectation des broches
#define DIRECTION 2
#define VALIDATION 3
#define BTN_0 4
#define BTN_1 5
#define BTN_2 6


boolean etat_1;
boolean etat_2;
boolean etat_3;
int vitesse=0;
int aff_vitesse;

void setup()
{
 Serial.begin(9600);                    // initialisation du moniteur série
 pinMode(DIRECTION,OUTPUT);             // configuration des broches
 pinMode(VALIDATION,OUTPUT);
 pinMode(BTN_0,INPUT);
 pinMode(BTN_1,INPUT);
 pinMode(BTN_2,INPUT);
}

void loop()
```

```
{
 etat_1=digitalRead(BTN_0);          // lecture de l'interrupteur BTN_0
 etat_2=digitalRead(BTN_1);          // lecture de l'interrupteur BTN_1
 etat_3=digitalRead(BTN_2);          // lecture de l'interrupteur BTN_2
 if (etat_1==HIGH)                   // arrêt du moteur
  {
    vitesse=0;
  }
 if (etat_2==HIGH)                   // augmentation de la vitesse
  {
   vitesse=vitesse+10;
   if(vitesse>255)                   // le rapport cyclique est à 100 %
    {
    vitesse=255;
    }
  }
 if (etat_3==HIGH)                  // diminution de la vitesse
  {
   vitesse=vitesse-10;
   if(vitesse<0)                    // le rapport cyclique est à 0 %
    {
    vitesse=0;
    }
  }

digitalWrite(DIRECTION,HIGH);      // pour faire tourner le moteur dans l'autre sens, mettre LOW à la place de HIGH
analogWrite(VALIDATION,vitesse);
aff_vitesse=map(vitesse,0,255,0,100);              // changement d'échelle pour l'affichage (0 à 100 %)
Serial.print("La vitesse du moteur est egale a ");
Serial.print(aff_vitesse);
Serial.println(" %.");
delay(100);
}
```

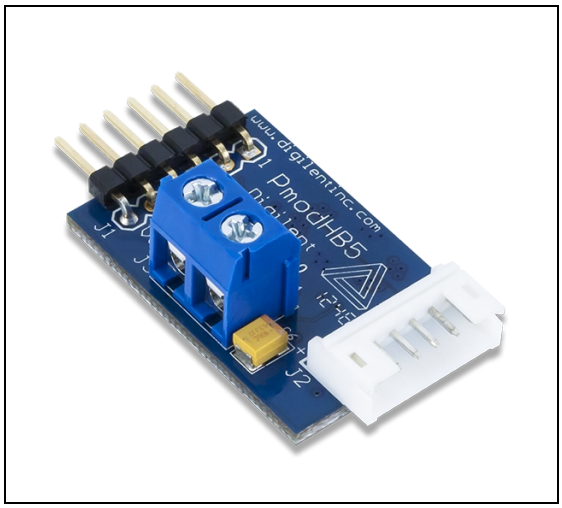
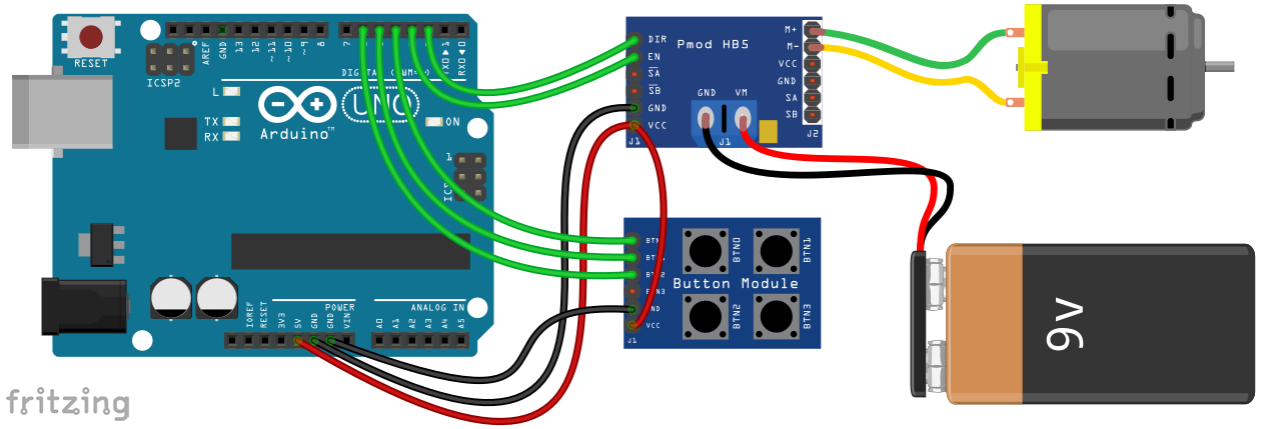


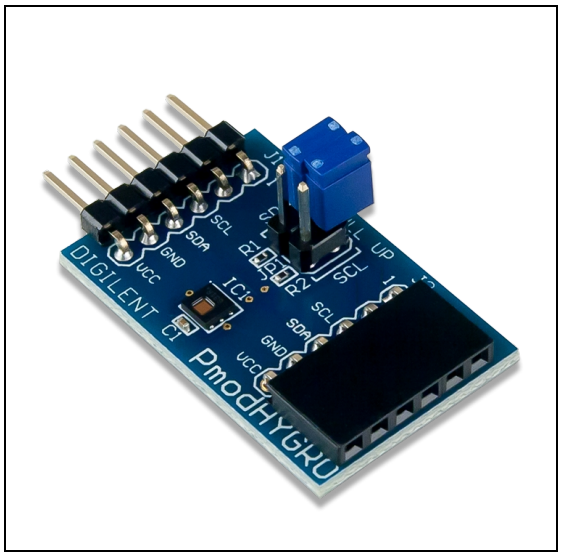
Schéma de câblage :



040 : PMODHYGRO : Module hygromètre/temp

Programme Arduino :

```
/************************************************************************
*
*   Test du module Pmod hygromètre/temp
*
************************************************************************
* Description:  Pmod_HYGRO
* La valeur de la température et de l'humidité sont affichées
* sur le moniteur série.
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod HYGRO (télécharger la bibliothèque
*       https://github.com/closedcube/ClosedCube_HDC1080_Arduino)
*
* Câblage
*       Module<----------> Arduino
*       J1 broche 6       5 V
*       J1 broche 5       GND
*       J1 broche 4       A5
*       J1 broche 2       A4
*
************************************************************************/
// appel des bibliothèques
#include <SPI.h>
#include "ClosedCube_HDC1080.h"

ClosedCube_HDC1080 hdc1080;                    // création de l'objet

void setup()
{
 Serial.begin(9600);                           // initialisation de la liaison série
 hdc1080.begin(0x40);                          // initialisation du capteur
}

void loop()
{
 Serial.print("Temperature en C = ");
 Serial.println(hdc1080.readTemperature());    // acquisition de la température
 Serial.print("Humidite en % = ");
 Serial.println(hdc1080.readHumidity());       // acquisition de l'humidité
 delay(1000);
}
```


SCL
SDA
GND
Vcc
DIGILENT
PmodHYGRO

041 : PMODISNS20 :

Module capteur de courant

Attention, ce module fonctionne sous 3,3 V

Programme Arduino :

```
/************************************************************************
*
*   Test du module Pmod capteur de courant
*
************************************************************************
* Description:  Pmod_ISNS20
* La valeur du courant capté est affiché sur le moniteur série.
*
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod ISNS20
*
************************************************************************/
#define Correction 100          // pourcentage de correction de la sensibilité du capteur
#define Filtre 0                //  Filtre=1 si utilisation des filtres du module
#define Echantillon 5000        // nombre d'échantillons
#define CS 10                   // affectation de la broche CS

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

int MSB;
int LSB;
signed int valeur;
signed int total = 0;
signed int milli_amps;

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()
{
 if(Filtre == 1)
 {
   for(int i; i < Echantillon; i++)
```

```
  {
   digitalWrite(CS, LOW);                    // activation de la ligne CS
   MSB=SPI.transfer(0x00);                   // récupération des bit de poids forts
   LSB=SPI.transfer(0x00);                   // récupération des bit de poids faibles
   digitalWrite(CS, HIGH);                   // désactivation de la ligne CS
   valeur = (Correction / 100) * (10000  * (((MSB<<8) | LSB) - 2048)) / 899; //formule donnée dans la
documentation
   total = total + valeur;
   }
  milli_amps = total / Echantillon;
 }
 else
 {
  digitalWrite(CS, LOW);                     // activation de la ligne CS
  MSB=SPI.transfer(0x00);                    // récupération des bit de poids forts
  LSB=SPI.transfer(0x00);                    // récupération des bit de poids faibles
  digitalWrite(CS, HIGH);                    // désactivation de la ligne CS
  milli_amps = (Correction / 100) * (10000  * (((MSB<<8) | LSB) - 2048)) / 899;  //formule donnée dans la
documentation
 }
 Serial.print("Courant=");
 Serial.print(milli_amps);
 Serial.println(" mA");
 delay(10);
}
```

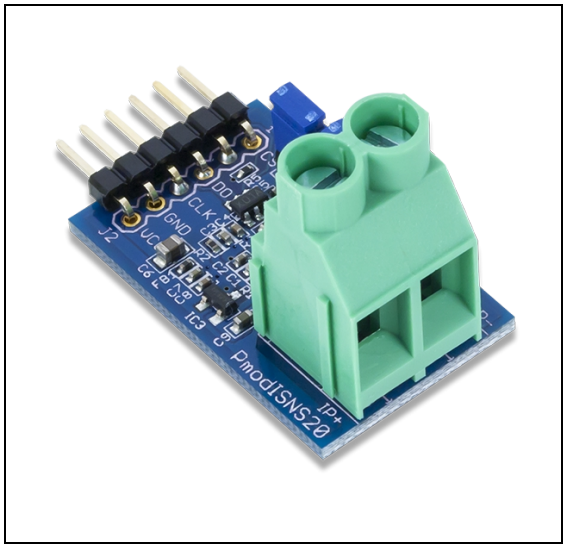
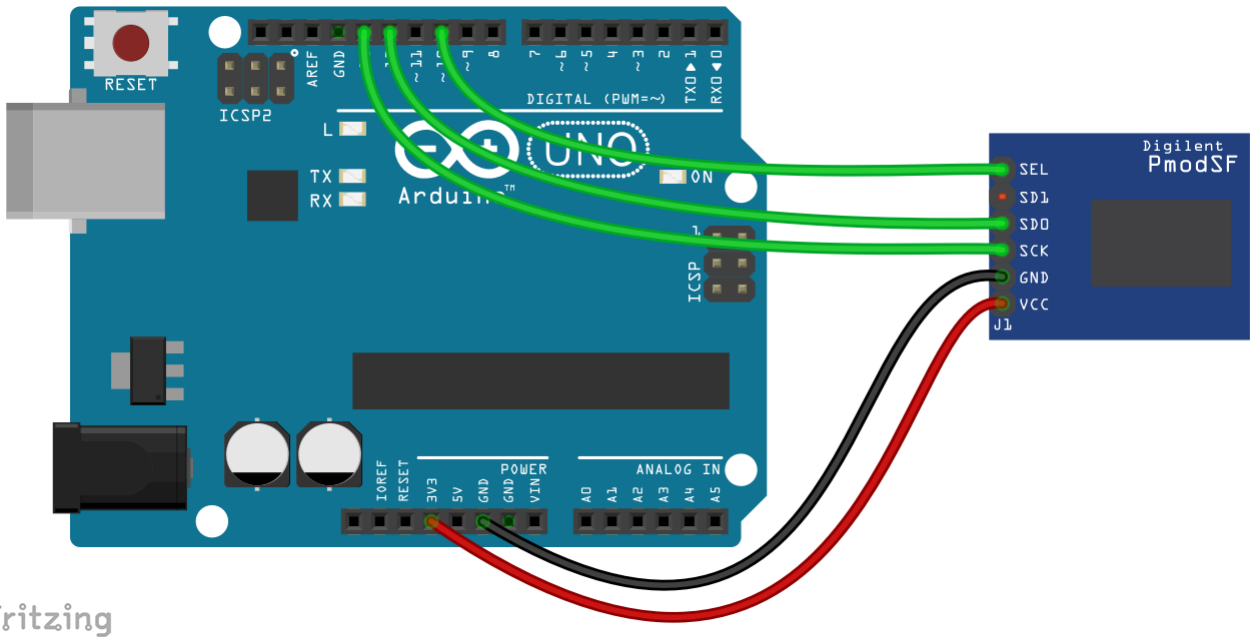


Schéma de câblage :



042 : PMODLS1 : Module comparateur

Programme Arduino :

```
/************************************************************************
*
*   Test du module Pmod comparateur
*
************************************************************************
* Description:  Pmod_LS1
* La tension de sortie du module potentiomètre digital évolue de 0 à 5 V
* et le module comparateur renvoie un niveau haut si la tension est
* supérieure au seuil.
* La tension d'entrée et le niveau de sortie du module comparateur est affiché
* dans le moniteur série.
*
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod DPOT
*       3. Module Pmod LS1
*
************************************************************************/

#define CS 10                                   // affectation de la broche CS
#define comparateur 2                           // affectation de la broche comparateur
#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 de la liaison 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(comparateur, INPUT);
}

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
    if (digitalRead(comparateur)==HIGH)
    {
     Serial.println("La sortie du comparateur est au niveau haut");
    }
    else
    {
     Serial.println("La sortie du comparateur est au niveau bas");
    }
    delay(200);
    }
}
```

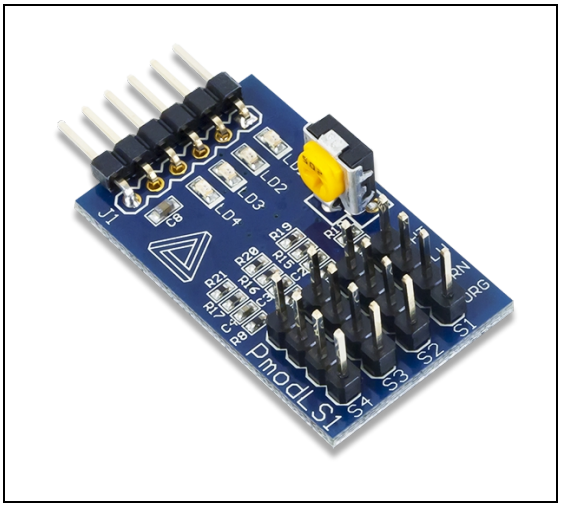
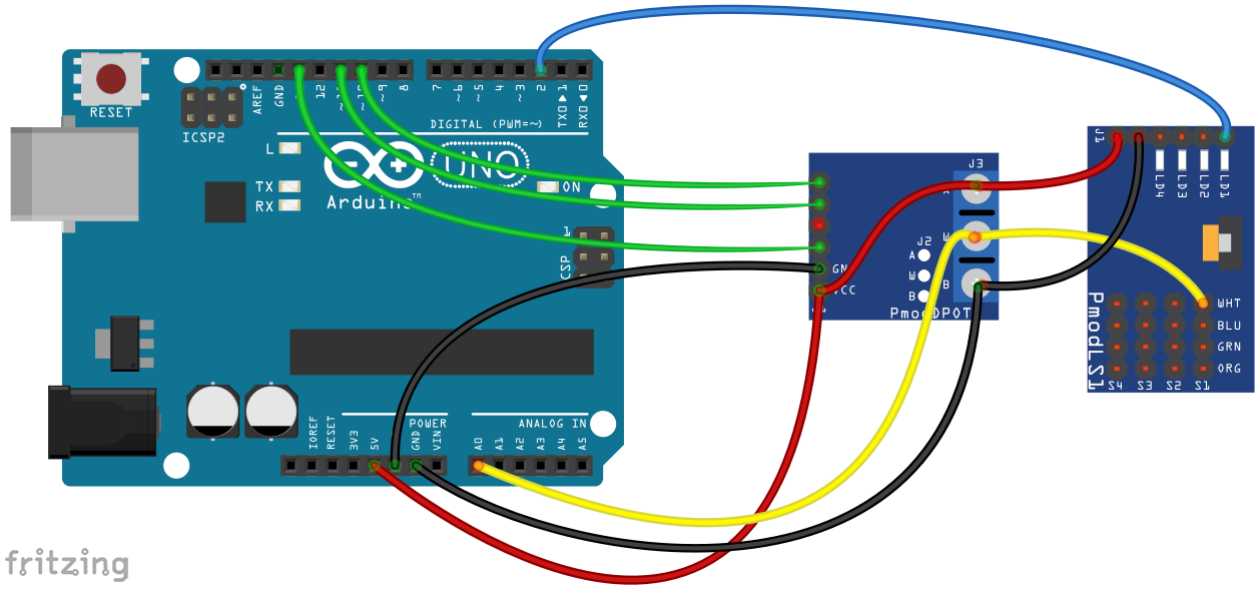


Schéma de câblage :



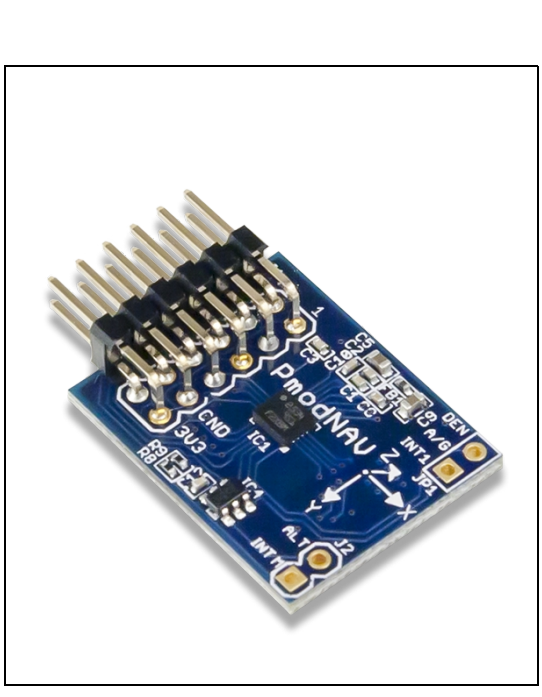
043 : PMODNAV : Accéléromètre IMU 9 AXES + baromètre

Attention, ce module fonctionne sous 3,3 V

Programme Arduino :

```
/************************************************************************
*
*   Test du module Pmod IMU 9 axes + Baromètre  (basé sur le programme de Jim Lindblom)
*
************************************************************************
* Description:  Pmod_NAV
* Toutes les données (accéléromètre, gyroscope, magnétomètre)  sont affichées
* dans le moniteur série
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod NAV (télécharger la bibliothèque
*       https://github.com/sparkfun/SparkFun_LSM9DS1_Arduino_Library)
*       Licence Beerware
*
* Câblage
*       Module<----------> Arduino
*       J1 broche 6       3.3V
*       J1 broche 5       GND
*       J1 broche 4       A5
*       J1 broche 2       A4
************************************************************************/
// Appel des bibliothèques
#include <Wire.h>
#include <SparkFunLSM9DS1.h>

// Déclaration des adresses du module
#define LSM9DS1_M  0x1E
#define LSM9DS1_AG  0x6B

LSM9DS1 imu;                                    // création de l'objet imu

// Configuration du module
#define PRINT_CALCULATED
#define PRINT_SPEED 250
static unsigned long lastPrint = 0;

// Le champ magnétique terrestre varie en fonction de sa localisation.
// Il faut ajouter ou soustraire une constante pour obtenir la bonne valeur
// du champ magnétique à l'aide du site suivant
// http://www.ngdc.noaa.gov/geomag-web/#declination
#define DECLINATION -0.33                       // déclinaison (en degrés) pour Paris.
```

```
void setup(void)
{
  Serial.begin(115200);                        // initialisation de la liaison série
  imu.settings.device.commInterface = IMU_MODE_I2C;    // initialisation du module
  imu.settings.device.mAddress = LSM9DS1_M;
  imu.settings.device.agAddress = LSM9DS1_AG;
  if (!imu.begin())
 {
   Serial.println("Probleme de communication avec le LSM9DS1.");
   while (1);
 }
}

void loop()
{

 if ( imu.gyroAvailable() )
 {
   imu.readGyro();           // acquisition des données du gyroscope
 }
 if ( imu.accelAvailable() )
 {
   imu.readAccel();          // acquisition des données de l'accéléromètre
 }
 if ( imu.magAvailable() )
 {
   imu.readMag();            // acquisition du magnétomètre
 }

 if ((lastPrint + PRINT_SPEED) < millis())
 {
   printGyro();  // Print "G: gx, gy, gz"
   printAccel(); // Print "A: ax, ay, az"
   printMag();   // Print "M: mx, my, mz"
   printAttitude(imu.ax, imu.ay, imu.az,-imu.my, -imu.mx, imu.mz);
   Serial.println();

   lastPrint = millis();
 }
}

void printGyro()
{
 Serial.print("G: ");
#ifdef PRINT_CALCULATED
 Serial.print(imu.calcGyro(imu.gx), 2);
 Serial.print(", ");
 Serial.print(imu.calcGyro(imu.gy), 2);
 Serial.print(", ");
```



```
  Serial.print(imu.calcGyro(imu.gz), 2);
  Serial.println(" deg/s");
#elif defined PRINT_RAW
  Serial.print(imu.gx);
  Serial.print(", ");
  Serial.print(imu.gy);
  Serial.print(", ");
  Serial.println(imu.gz);
#endif
}

void printAccel()
{
  Serial.print("A: ");
#ifdef PRINT_CALCULATED
  Serial.print(imu.calcAccel(imu.ax), 2);
  Serial.print(", ");
  Serial.print(imu.calcAccel(imu.ay), 2);
  Serial.print(", ");
  Serial.print(imu.calcAccel(imu.az), 2);
  Serial.println(" g");
#elif defined PRINT_RAW
  Serial.print(imu.ax);
  Serial.print(", ");
  Serial.print(imu.ay);
  Serial.print(", ");
  Serial.println(imu.az);
#endif

}

void printMag()
{
  Serial.print("M: ");
#ifdef PRINT_CALCULATED
  Serial.print(imu.calcMag(imu.mx), 2);
  Serial.print(", ");
  Serial.print(imu.calcMag(imu.my), 2);
  Serial.print(", ");
  Serial.print(imu.calcMag(imu.mz), 2);
  Serial.println(" gauss");
#elif defined PRINT_RAW
  Serial.print(imu.mx);
  Serial.print(", ");
  Serial.print(imu.my);
  Serial.print(", ");
  Serial.println(imu.mz);
#endif
}
```

```
void printAttitude(float ax, float ay, float az, float mx, float my, float mz)
{
  float roll = atan2(ay, az);
  float pitch = atan2(-ax, sqrt(ay * ay + az * az));
  float heading;
  if (my == 0)
    heading = (mx < 0) ? PI : 0;
  else
    heading = atan2(mx, my);
   heading -= DECLINATION * PI / 180;
   if (heading > PI) heading -= (2 * PI);
   else if (heading < -PI) heading += (2 * PI);
   else if (heading < 0) heading += 2 * PI;
   heading *= 180.0 / PI;
   pitch *= 180.0 / PI;
   roll  *= 180.0 / PI;
  Serial.print("Pitch, Roll: ");
  Serial.print(pitch, 2);
  Serial.print(", ");
  Serial.println(roll, 2);
  Serial.print("Heading: "); Serial.println(heading, 2);
}
```

044 : PMODSTEP : Module pilotage moteur pas à pas

Programme Arduino :

```
/*****************************************************************************
*
*    Test du module Pmod pilotage de moteur pas à pas
*
*****************************************************************************
* Description:  Pmod_STEP
* Le moteur fait un tour complet dans un sens puis revient à sa position
* d'origine
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod STEP (positionner le cavalier JP1 en position 2-3)
*
*****************************************************************************/
#include <Stepper.h>                          // appel de la bibliothèque

#define enroulement_1 2                       // affectation des broches
#define enroulement_2 3
#define enroulement_3 4
#define enroulement_4 5

const int nombre_pas=2048;                    // nombre de pas du moteur

Stepper moteur(nombre_pas, enroulement_1, enroulement_2, enroulement_3, enroulement_4);    // création de
l'objet moteur

void setup()
{
Serial.begin(9600);
moteur.setSpeed(10);                          // initialisation la vitesse de rotation du moteur en tour par minute
pinMode(enroulement_1, OUTPUT);               // configuration des broches en sortie
pinMode(enroulement_2, OUTPUT);
pinMode(enroulement_3, OUTPUT);
pinMode(enroulement_4, OUTPUT);
}

void loop()
{
Serial.println("Sens 1");
for (int i=1; i<=nombre_pas; i++)             // boucle avance du moteur en fonction du nombre de pas
{
 moteur.step(1);                              // un pas en sens positif
```

```
/* Temporisation utile pour voir le déroulement des séquences sur les led du module.
   Diminuer la temporisation pour augmenter la vitesse de rotation du moteur.*/
  delay(100);
}
Serial.println("Sens 2");
for (int i=1; i<=nombre_pas; i++)                    // boucle retour du moteur en fonction du nombre de pas
{
  moteur.step(-1);                                   // un pas en sens négatif
/* Temporisation utile pour voir le déroulement des séquences sur les led du module.
   Diminuer la temporisation pour augmenter la vitesse de rotation du moteur.*/
  delay(100);
}
}
```

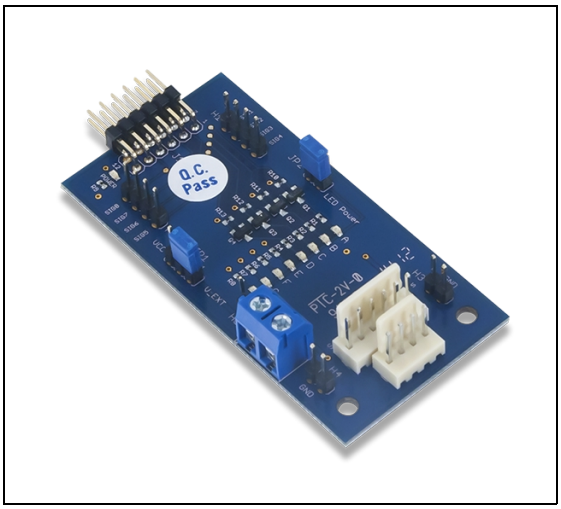
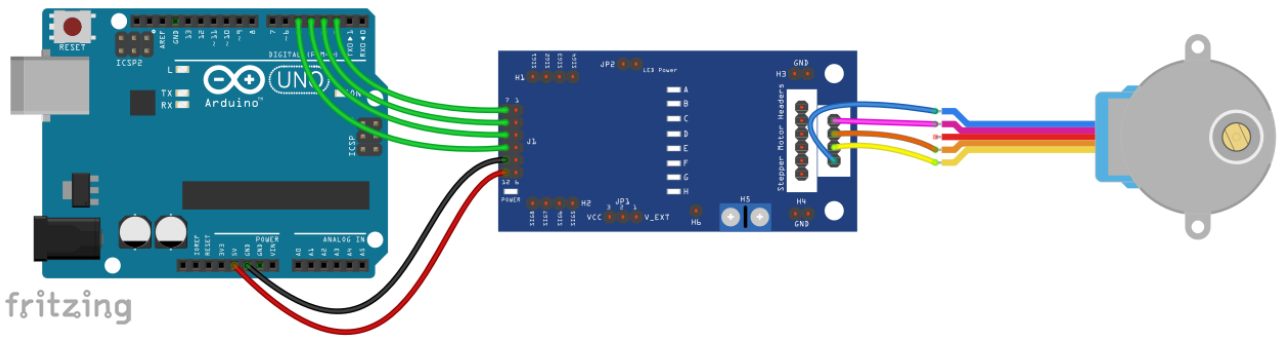


Schéma de câblage :



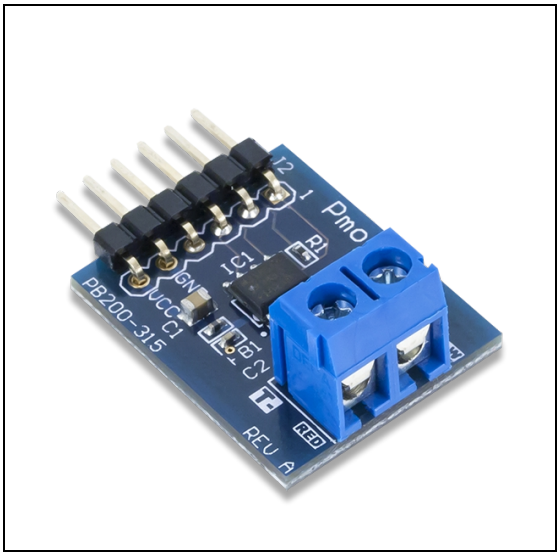
045 : PMODTC1 : Thermocouple K

Attention, ce module fonctionne sous 3,3 V

Programme Arduino :

```
/************************************************************************
*
*   Test du module Pmod thermocouple type K
*
************************************************************************
* Description:  Pmod_TC1
* La valeur de la température est affichée sur le moniteur série.
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod TC1 (télécharger la bibliothèque
*       https://github.com/adafruit/Adafruit-MAX31855-library)
*       3. Module Adafruit TXB0108
*
* Câblage
*       Module<----------> TXB0108 <-----------> Arduino
*       J1 broche 6                             3.3 V
*       J1 broche 5                             GND
*       J1 broche 4                              13
*       J1 broche 3                              12
*       J1 broche 1                              10
************************************************************************/

// affectation des broches
#define CS 10
#define MISO 12
#define SCLK 13

// appel des bibliothèques
#include <SPI.h>
#include "Adafruit_MAX31855.h"

Adafruit_MAX31855 Temp(SCLK, CS, MISO);        // création de l'objet

void setup()
{
 Serial.begin(9600);                           // initialisation de la liaison série
 delay(500);
}

void loop()
{
```

```
 Serial.print("Temperature en C = ");
 Serial.println(Temp.readCelsius());          // acquisition de la température en degré Celcius
 Serial.print("Temperature en F = ");
 Serial.println(Temp.readFarenheit());        // acquisition de la température en degré Fahrenheit
 delay(1000);
}
```



046 : PMODTMP2 : Module thermostat\thermomètre

Programme Arduino :

```
/************************************************************************
*
*    Test du module Pmod thermostat\thermomètre
*
************************************************************************
* Description:  Pmod_TMP2
* La température ambiante est affiché sur le moniteur série et
* sur un afficheur LCD série .
*
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod TMP2
*       3. Module Pmod CLS
*
************************************************************************/

#include <Wire.h>                       // appel de la bibliothèque
#define ADT7420_Adresse 0x4B            // adresse I2C du module Pmod TMP2

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

int MSB;
int LSB;
int valeur;
float temperature;

void setup()
{
 Serial.begin(9600);                    // initialisation de la liaison série
 Wire.begin();                          // initialisation de la liaison I2C
 Init_ADT7420();                        // initialisation du module Pmod AD2
 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()
{
 Wire.beginTransmission(ADT7420_Adresse);       // lancement de la mesure
```

```
 Wire.endTransmission();
 delay(10);
 Wire.requestFrom(ADT7420_Adresse, 2);          // récupération des deux octets MSB et LSB
 if(Wire.available() <=2)
 {
  MSB = Wire.read();
  LSB = Wire.read();
 }
 valeur=(MSB<<8)|LSB ;
 if (((valeur>>15)&1)==0)                        // si la température est positive
 {
  temperature=valeur/128.0;
 }
 else                                            // si la température est négative
 {
  temperature=(valeur-65535)/128.0;
 }
 Serial.print("MSB=");                           // affichage dans le moniteur série
 Serial.println(MSB);
 Serial.print("LSB=");
 Serial.println(LSB);
 Serial.print("Valeur=");
 Serial.println(valeur);
 Serial.print("Temperature=");
 Serial.println(temperature);
 lcd.write("\x1b[j");                            // effacement de l'afficheur
 lcd.print("Temperature:");                      // écriture sur l'afficheur
 lcd.write("\x1b[1;0H");                         // positionnement du curseur 2nde ligne 1ère colonne
 lcd.print(temperature);
 lcd.print((char)223);
 lcd.print("c");
 delay(1000);
}

// Initialisation du module Pmod TMP2
void Init_ADT7420(void)
{
// configuration de l'ADT7420 en mode 16 bit
 Wire.beginTransmission(ADT7420_Adresse);
 Wire.write(0x03);
 Wire.write(0x80);
 Wire.endTransmission();
}
```

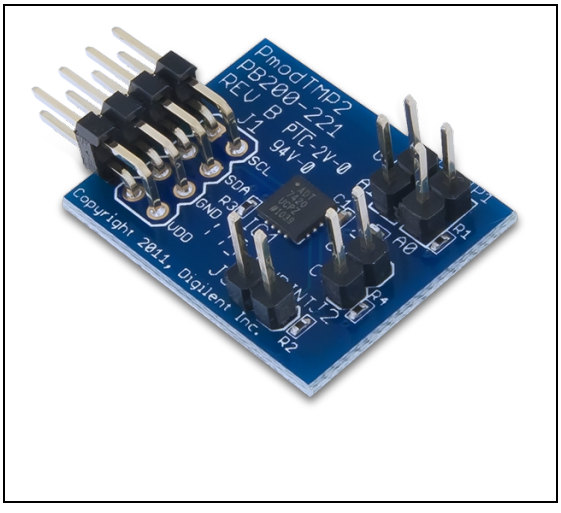
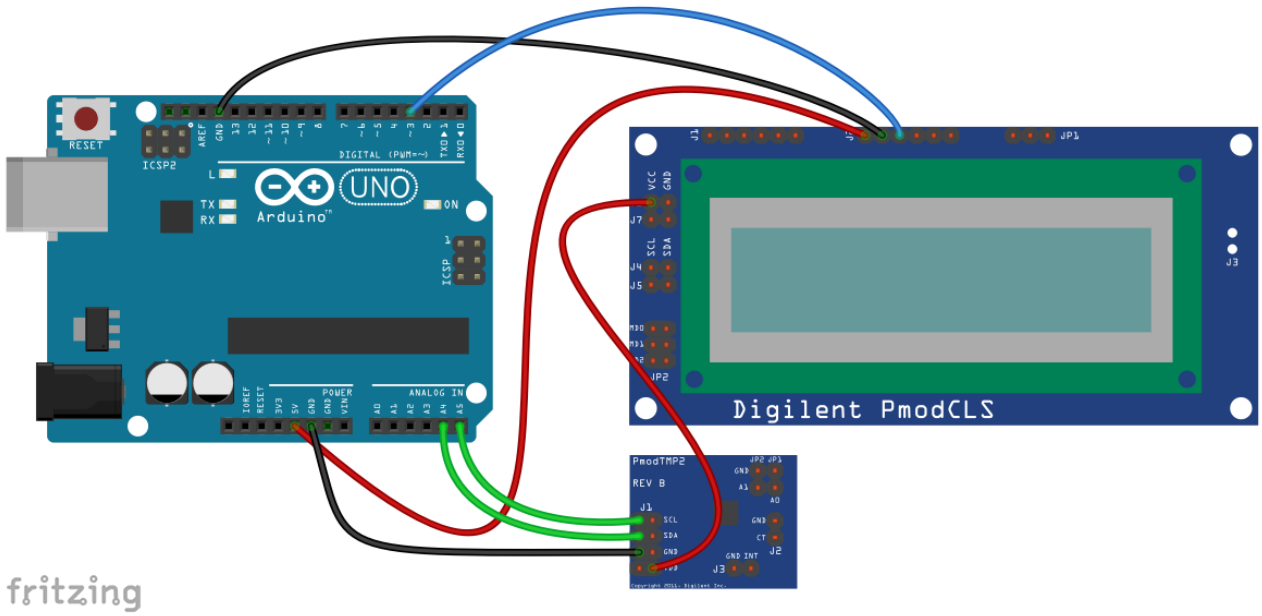


Schéma de câblage :



047 : PMODTMP3 : Module thermomètre

Programme Arduino :

```
/************************************************************************
*
*   Test du module Pmod thermomètre
*
************************************************************************
* Description:  Pmod_TMP3
* La température ambiante (en °F et °C) est affichée dans le moniteur série .
*
*
* Matériel
*       1. Arduino Uno
*       2. Module Pmod TMP3 (position des cavaliers JP1, JP2 et JP3  sur GND)
*
************************************************************************/

#include <Wire.h>                          // appel de la bibliothèque
#define TCN75A_Adresse 0x48                // adresse I2C du module Pmod TMP3

int MSB;
int LSB;
int valeur;
float temperature;

void setup()
{
 Serial.begin(9600);                       // initialisation de la liaison série
 Wire.begin();                             // initialisation de la liaison I2C
 Init_TCN75A();                            // initialisation du module Pmod TMP3
}

void loop()
{
 Wire.beginTransmission(TCN75A_Adresse);   // lancement de la mesure
 Wire.endTransmission();
 delay(10);
 Wire.requestFrom(TCN75A_Adresse, 2);      // récupération des deux octets MSB et LSB
 if(Wire.available() <=2)
 {
   MSB = Wire.read();
   LSB = Wire.read();
 }
 valeur=LSB |(MSB<<8) ;
 valeur=valeur>> 4;
 temperature=valeur/16.00;
```

```
  Serial.print("MSB=");                    // affichage dans le moniteur série
  Serial.println(MSB);
  Serial.print("LSB=");
  Serial.println(LSB);
  Serial.print("Valeur=");
  Serial.println(valeur);
  Serial.print("Température en F = ");
  Serial.print(temperature*9/5 + 32,2);
  Serial.print("   ");
  Serial.print("Température en C = ");
  Serial.println(temperature, 2);
  delay(1000);
}

// Initialisation du module Pmod TMP3
void Init_TCN75A(void)
{
  Wire.beginTransmission(TCN75A_Adresse);
  Wire.write(0x01);
  Wire.write(0x60);
  Wire.endTransmission();
  Wire.beginTransmission(TCN75A_Adresse);
  Wire.write(0x00);
  Wire.endTransmission();
}
```

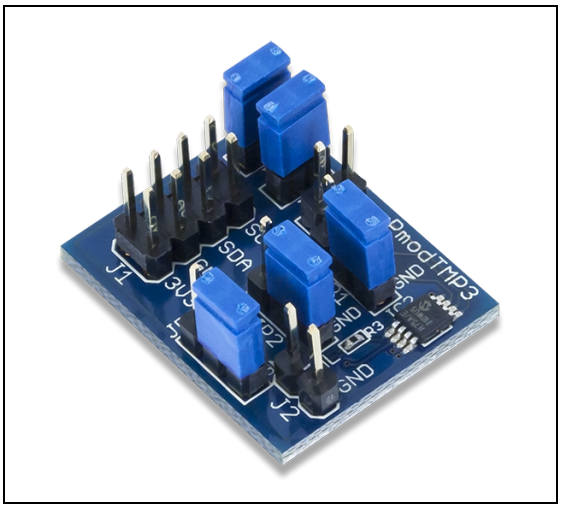


Schéma de câblage :