carte d'extension de ports

augmentez le nombre d'E/S de votre carte de développement

Alessandro Sottocornola (Italie)

Elettronica In
WWW.ELETTRONICAIN.IT

Êtes-vous fatigué de devoir contourner le manque de ports d'E/S lors du développement de vos projets sur votre plateforme préférée ? Vous en voulez plus pour gérer une multitude de signaux et d'actionneurs ? Grâce au bus I²C, chacune de ces cartes d'extension permet d'ajouter jusqu'à 16 E/S pour atteindre un maximum de 128 sur votre système existant.

On a souvent besoin de gérer plusieurs signaux et actionneurs avec un microcontrôleur ne disposant pas de suffisamment de broches d'E/S. Dans ces situations, on peut profiter de circuits intégrés appelés « extensions d'E/S », dont la fonction est d'ajouter un certain nombre de lignes d'entrée et de sortie et de les commander via une ligne de communication série I²C ou SPI.
Afin de couvrir ce besoin de multiplier les lignes d'E/S, nous avons fabriqué une carte de liaison basée sur le Microchip MCP23017 [1], une extension d'E/S à 16 bits. Nous avons complété la carte avec le matériel minimal nécessaire à son fonctionnement, ici un commutateur DIP et quelques résistances de rappel. Enfin, pour illustrer les possibilités d'une extension d'E/S, nous vous proposons un exemple d'application où la carte de liaison pilote avec des signaux TTL ou avec des boutons une platine équipée d'une série de huit relais. Mais allons-y dans l'ordre et voyons d'abord comment fonctionne cette puce.

Le MCP23X17

Cette puce intégrée polyvalente permet une extension générique des E/S série/parallèle à 16 bits et est disponible en deux versions : celle utilisée ici, le MCP23017, équipée d'une interface de bus I²C, et le MCP23S17, une variante SPI. La puce est une extension d'E/S à 16 bits, divisée en deux ports de 8 bits chacun, interfacés via le bus I²C. C'est ainsi qu'avec seulement deux fils, référencés à la masse, on peut scruter l'état de jusqu'à 16 lignes (en mode entrée) ou de définir l'état logique de chacune d'entre elles (en mode sortie). Les lignes E/S fonctionnent par défaut en entrées.
Le MCP23017 se compose de plusieurs registres de 8 bits pour la sélection des entrées, des sorties et de la polarité. Le système principal peut activer les E/S en tant qu'entrées ou sorties en écrivant les bits de configuration correspondants (IODIRA/B). Les données de chaque entrée ou sortie sont stockées dans le registre d'entrée ou de sortie correspondant. Le registre d'inversion de polarité permet d'inverser la polarité du registre du port d'entrée. Tous les registres peuvent être lus à partir du système principal. Le port d'E/S 16 bits est structurellement composé de deux ports 8 bits, à savoir le port A et le port B, pilotés respectivement par les broches 21...28 et 1...8. Le MCP23X17 peut être configuré pour fonctionner en mode 8 bits ou 16 bits. En outre, il dispose de deux broches d'interruption, INTA et INTB, que l'on peut affecter de deux manières :

- Les broches d'interruption fonctionnent indépendamment. INTA reflète les conditions d'interruption sur le port A et INTB reflète les conditions d'interruption sur le port B.
- Les deux broches d'interruption sont actives lorsqu'une interruption se produit sur l'un ou l'autre des ports.

Schéma du circuit

On peut constater sur le schéma-bloc de la **figure 1** que la carte de liaison est très basique, puisqu'on y trouve le

circuit intégré MCP23017 en version DIP, avec toutes ses broches connectées à des barrettes (qui sur le PCB ont un pas de 2,54 mm pour l'insertion dans d'autres cartes ou platines d'expérimentation). Un commutateur DIP à trois voies (SW1 sur le schéma de câblage) permet de positionner les trois bits de poids faible de l'adresse I²C du périphérique. Les broches A0, A1 et A2 sont maintenues au niveau haut par les résistances R1 à R3 lorsque les interrupteurs sont ouverts.

Les broches d'E/S des registres A et B ont été disposées de part et d'autre afin de faciliter la connexion de notre carte. Vous pouvez connecter tout ce que vous voulez sur ces broches d'E/S numériques, dans les limites du courant et de la tension supportés par le circuit intégré MCP23017. En plus des barrettes latérales au pas de 2,54 mm, quatre autres broches ont été exposées sur une barrette nommée *I2C* (également au pas de 2,54 mm), afin de pouvoir se connecter au bus directement sur le dessus du module pour plus de flexibilité. Ces broches, qui comprennent l'alimentation positive de 5 V et la masse, sont connectées en parallèle aux broches correspondantes sur les connecteurs latéraux, et qui peuvent donc aussi être utilisées. SDA et SCL sont tous deux dotés de résistances de rappel.

Notez que sur le circuit imprimé, par commodité, les broches du registre A ont été placées d'un côté et les broches B du côté opposé, toujours dans le but de simplifier les connexions. Comme nous l'avons dit, les broches appartenant au bus I²C ont été renvoyées sur la barrette marquée ***I²C***, de même que le positif 5 V et la masse ; les deux sont dotés d'une résistance de rappel. Dans notre carte de liaison, la broche *reset* de l'extension d'E/S n'est pas utilisée ; par conséquent, pour la désactiver, nous avons placé la broche correspondante (18, RST) au niveau logique haut par l'intermédiaire de la résistance R4.

Chaque barrette latérale de la carte comporte également une duplication des lignes 5 V et GND. L'ensemble du circuit est alimenté par le contact 5 V (il y a en réalité deux contacts : 1 et 24, situés sur les côtés longs de la carte de liaison) référencé à la masse (contacts GND, c.à.d. 2 et 23 des rangées latérales).

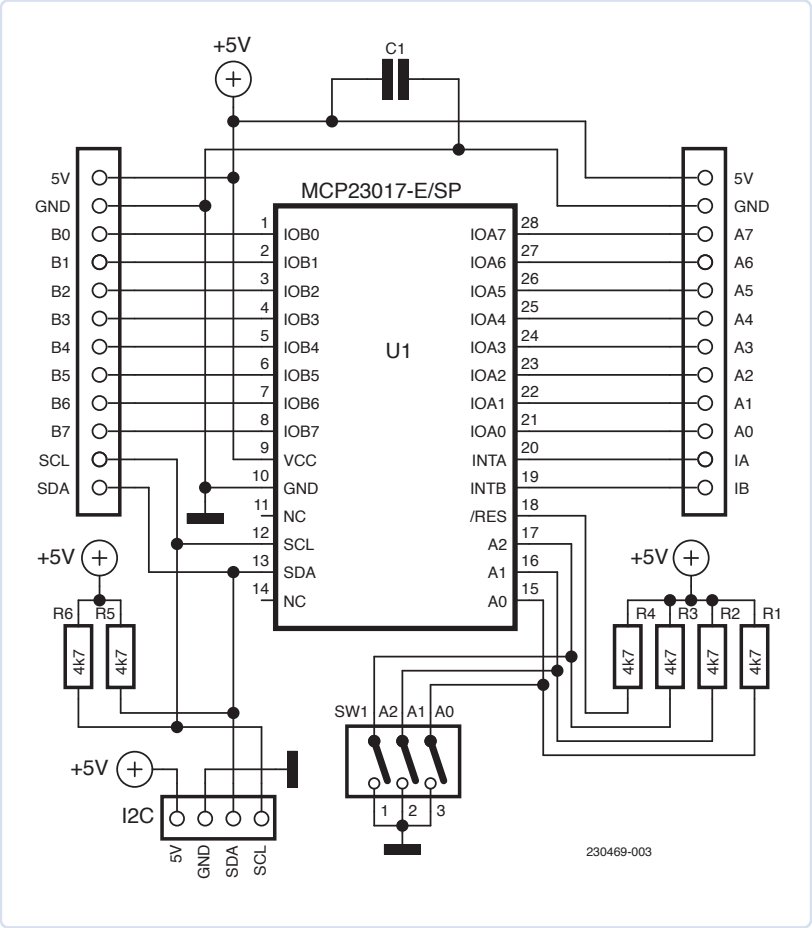


Figure 1. Schéma de la carte de liaison.

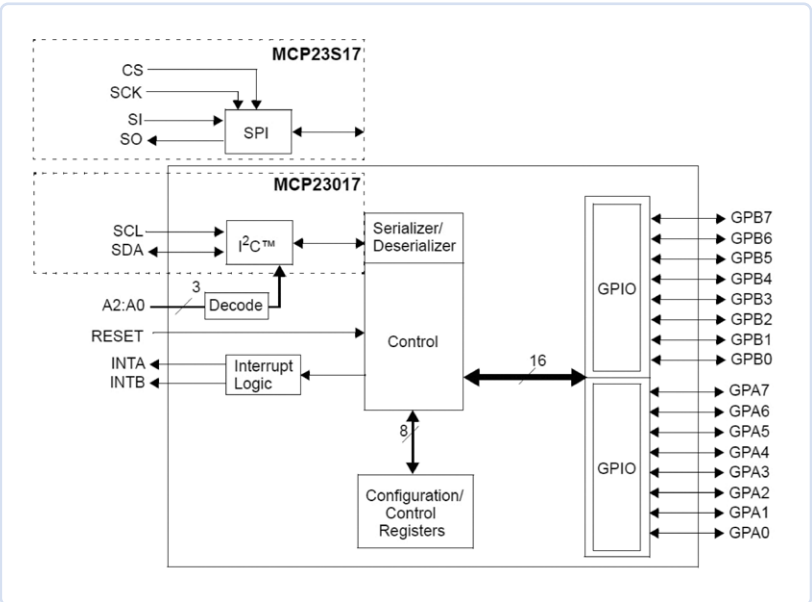


Figure 2. Schéma-bloc du MCP23017. La variante SPI, appelée MCP23S17, est également représentée. (Source : Microchip [3])

L'extension d'E/S

L'élément principal du circuit est, bien sûr, le MCP23017 fabriqué par Microchip (marqué U1), que l'on peut considérer comme un convertisseur bus I²C/parallèle. Le circuit intégré, dont on voit le schéma-bloc interne à la **figure 2**, fonctionne comme un périphérique (agent) du bus I²C et prend en charge deux modes : entrée et sortie. Dans le premier, il permet de transférer les états des E/S des registres A et B sur le bus en format série, un octet pour chaque registre, à la demande du dispositif principal du bus I²C ; dans le second, il va régler les lignes d'E/S en convertissant les données entrantes sur le bus I²C à l'état correspondant des lignes des registres A et B. La sortie d'interruption peut être configurée pour

Le CI MCP23017 en bref

La puce au cœur de la carte décrite ici est une extension d'E/S à 16 bits, divisée en deux ports de 8 bits chacun, interfacée via le bus I²C. Cela signifie qu'avec seulement deux fils, référencés à la masse, il vous permet d'acquérir l'état, ou de le définir en sortie, de pas moins de 16 lignes. Ses caractéristiques techniques sont les suivantes :

- Interface de données I²C à grande vitesse, fonctionnant à 100 kHz, 400 kHz ou 1,7 MHz
- Adresse du bus I²C réglable à 8 combinaisons
- Broches d'interruption configurables, par niveau et fonction logique
- Source d'interruption configurable Registre d'inversion de polarité

Pour les entrées :

- Entrée de réinitialisation externe
- Courant de veille de 1 µA max.
- Tension d'alimentation de 1,8 V à 5,5 V

se déclencher sous deux conditions (mutuellement exclusives) :

- Lorsqu'un état d'entrée diffère de l'état du registre du port d'entrée correspondant. Cette condition est utilisée pour indiquer au système principal qu'un état d'entrée a changé.
- Lorsque l'état d'une entrée est différent de la valeur préconfigurée du registre (registre DEFVAL).

Les lignes d'interruption INTA et INTB peuvent être configurées comme actives-hautes, actives-basses ou à drain ouvert. Le registre de capture d'interruption capture les valeurs des ports au moment où l'interruption est déclenchée, stockant ainsi la condition qui a provoqué l'interruption. La réinitialisation au démarrage (POR) ramène les registres à leurs valeurs par défaut et initialise la machine à états de l'appareil. La nécessité d'un fonctionnement bidirectionnel est due au fait que chaque périphérique du bus I²C doit à la fois être capable de lire (par exemple des commandes) et d'envoyer des données acquises sur 8+8 bits sur le bus.

Comme toutes les unités d'un bus I²C, le MCP23017 autorise le réglage de son adresse parmi une plage de huit adresses. Il dispose pour cela des broches A0, A1 et A2 qui permettent de fixer l'adresse de l'appareil pour y accéder directement à partir du bus I²C. Chacune de ces lignes est positionnée par le commutateur DIP SW1 : un DIP fermé définit l'état logique 0 sur la ligne correspondante, tandis qu'inversement un DIP ouvert définit l'état logique 1. La possibilité de définir huit adresses permet de placer jusqu'à huit extensions d'E/S sur le même bus et de commander ainsi un maximum de 128 E/S avec seulement trois lignes. Pour l'affectation de la carte de liaison à votre application, le **tableau 1** fournit la correspondance entre les adresses et le réglage des commutateurs DIP.

Tableau 1. Configuration de l'adresse de périphérique du MCP23017.

ADDR	A2	A1	A0
0x20	ON	ON	ON
0x21	ON	ON	OFF
0x22	ON	OFF	ON
0x23	ON	OFF	OFF
0x24	OFF	ON	ON
0x25	OFF	ON	OFF
0x26	OFF	OFF	ON
0x27	OFF	OFF	OFF

Avec ce matériel, la logique de fonctionnement est la suivante : chaque fois qu'il reçoit une chaîne sur la ligne SDA du bus I²C (rythmée par le signal d'horloge sur la ligne SCL), le circuit intégré MCP23017 exécute la commande qu'elle contient (dans ce cas, celle qui indique de charger l'octet de données) et dispose les huit lignes de sortie IOA0...IOA7 et IOB0...IOB7 comme les bits correspondants. Par exemple, IOA0 prendra l'état du premier bit de l'octet 1, IOA1 celui du deuxième bit, et ainsi de suite. Il en va de même pour les IOB0...IOB7, qui reproduiront exactement les bits du deuxième octet de données.

Bien entendu, la conversion et la sortie ne se produisent que si la chaîne reçue contient l'adresse du bus I²C correspondant à celle définie, via les commutateurs DIP de SW1, pour U1. À la réception de chaque chaîne, le circuit intégré met à jour l'état de ses sorties, et les niveaux logiques respectifs déterminent si les circuits en aval (par ex. des LED ou des segments d'afficheurs) sont activés ou restent éteints ; si aucune chaîne n'est envoyée par la suite, l'état de sortie conserve le dernier profil de l'octet car les sorties du MCP23017 sont maintenues. Ce qui précède s'applique au mode sortie, c.à.d. à l'écriture de l'état des deux octets du bus I²C dans les registres de sortie A et B. Si, en revanche, la commande provenant du bus est une lecture, le MCP23017 acquiert l'état E/S de chaque registre et génère deux octets, le premier contenant l'état de IOA0...IOA7 et le second celui de IOB0...IOB7 ; il les envoie ensuite comme réponse sur le bus I²C.

Réalisation pratique

Bien, maintenant que nous avons décrit le schéma de câblage, nous pouvons passer aux instructions de construction. Nous proposerons ensuite un exemple d'application basé sur l'interfaçage avec une carte Arduino accompagné du croquis correspondant. Comme d'habitude, nous avons dessiné un circuit imprimé dont nous mettons à disposition pour téléchargement les deux couches (il s'agit d'un circuit double face) à la page de ce projet sur Elektor Labs [2]. À partir de ces schémas, vous pouvez procéder à la préparation du circuit imprimé par photogravure. Après l'avoir gravé et

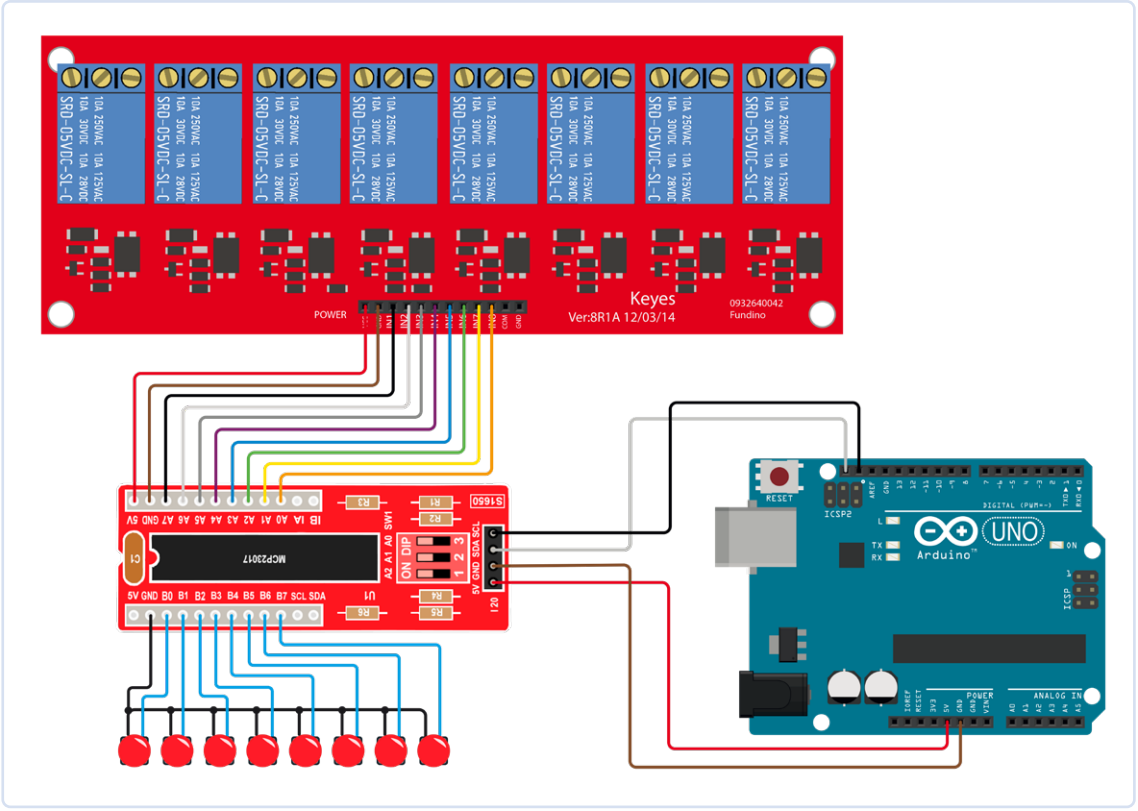


Figure 3. Schéma de câblage de l'application de commande de relais actionnés par boutons-poussoirs.

Liste des composants

Résistances

R1…R6 : 4,7 k

Condensateur

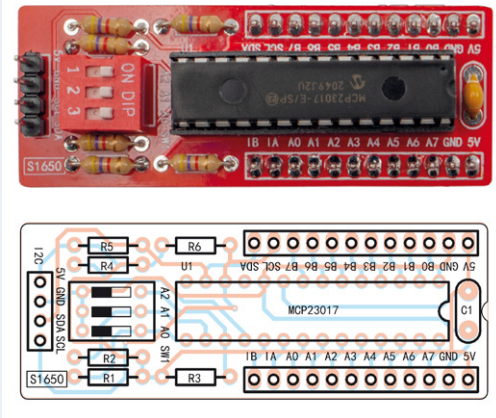
C1 = 100 nF, céramique

Semi-conducteurs

U1 = MCP23017-E/SP

Divers

SW1 = Commutateur DIP à 3 voies
Embase DIL 2 x 14 broches
2 x barrettes mâles à 12 broches
1 x barrette mâle à 4 broches
1 x circuit imprimé (voir texte)



percé, vous pouvez assembler les quelques composants nécessaires, qui, dans ce projet, sont tous traditionnels, de type traversant, pour ceux qui n'aiment pas trop la technique CMS. Commencez par insérer et souder les résistances, puis passez à l'embase pour le circuit intégré (à placer avec l'encoche orientée comme indiqué dans le plan de montage que vous voyez dans cet article) et au commutateur DIP à trois voies, à monter avec l'interrupteur 1 orienté vers la gauche, en regardant la carte avec l'embase pour U1 sur le dessus.

Enfin, insérez et étamez dans les trous respectifs la barrette à quatre broches marquée *I2C*, puis, de l'autre côté du PCB, insérez et étamez deux rangées de barrettes à 12 broches qui permettront le montage sur des platines ou l'insertion sur d'autres cartes, par exemple pour finalement se connecter à Arduino en utilisant de classiques cavaliers mâle/femelle. Une fois les composants soudés,

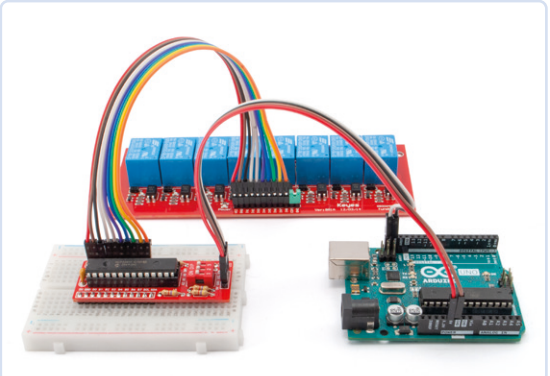


Figure 4. Le matériel pour tester le croquis d'exemple.

Listage 1. Projet de démonstration

```
#include <Adafruit_MCP23X17.h>.
#include <Adafruit_MCP23X17.h>
Adafruit_MCP23X17 mcp;
int i = 0;
int OUT[] = ;      //Represents MCP23017 PIN (A7...A0)
int IN[] = ;       //Represents MCP23017 PIN (B0...B7)
int STATO[] = ;    //For each output, every toggle the status is being saved
void setup()
{
 Serial.begin(9600);
 Serial.println("MCP23017 INPUT/OUTPUT");
 if (!mcp.begin_I2C(0x20))         //0x20 is MCP23017's address with A0=A1=A2 > ON(GND)
 {
    Serial.println("MCP Error!");  //If MCP is not found, the error is visualized
    while (1);
 }
 //bank A Pin set as outputs and B bank as inputs
 //The STATO variable to 0 to indicate idling outputs
 for (i=0; i<8; i=i+1)
 {
    mcp.pinMode(OUT[i], OUTPUT);
    mcp.pinMode(IN[i], INPUT_PULLUP);
    STATO[i] = 0;
 }
}

//******************* L O O P **********************
void loop()
{
 String Testo_Debug = "";
 for (i=0; i<8; i=i+1)
 {
    //If button pressed or output not activated, I activate it
    if ((mcp.digitalRead(IN[i])==0) && (STATO[i]==0))
    {
      STATO[i] = 1;
      Testo_Debug = "Pulsante " + String(i+1) + " premuto";
      Serial.println(Testo_Debug);
      mcp.digitalWrite(OUT[i], HIGH);
    }
    //If button released and output is active,I de-activate it
    if ((mcp.digitalRead(IN[i])==1) && (STATO[i]==1))
    {
      STATO[i] = 0;
      Testo_Debug = "Pulsante " + String(i+1) + " rilasciato";
      Serial.println(Testo_Debug);
      mcp.digitalWrite(OUT[i], LOW);
    }
 }
 delay(10);
}
```

insérez le MCP23017 dans son embase, en le tenant avec l'encoche orientée comme indiqué dans le plan de montage sur ces pages. Ceci fait, votre carte de liaison est prête pour l'expérimentation ou le prototypage.

Utilisons-la avec Arduino

La carte de liaison a été créée pour être reliée à un microcontrôleur, étant donné qu'on utilise généralement les extensions d'E/S avec des appareils équipés d'une interface série avec le bus I²C. Comme Arduino prend en charge ce bus, nous avons créé un exemple de code téléchargeable à [2] pour lire et gérer les E/S du MCP23017 via Arduino. Ce croquis permet essentiellement d'écrire l'état du registre du Port A en fonction d'un octet envoyé par Arduino sur le bus, dont les bits correspondent à l'état lu sur le Port B, qui sert cette fois d'entrée. Pour donner à l'exemple une application concrète, nous avons décidé d'utiliser les états logiques des E/S du Port A, qui fonctionneront ici comme des sorties numériques, pour piloter une carte à relais. Plus précisément, nous devons connecter les huit lignes de commande de sortie des relais d'une carte à relais à 8 canaux à la banque d'E/S du port A, tandis que huit boutons-poussoirs normalement ouverts doivent être connectés au port B, avec en commun le pôle connecté à GND. Pour réaliser cette application, il est nécessaire de connecter Arduino UNO, la carte de liaison, la carte à relais et les boutons, comme le montre le schéma de câblage proposé à la **figure 3**, tandis que la **figure 4** montre le prototype réel de cette application. Puisqu'il s'agit de boutons poussoirs assez communs (normalement ouverts) et qu'ils n'ont pas d'électronique externe, les résistances de rappel internes du MCP ont été activées (via la bibliothèque) pour les gérer et reconnaître le changement d'état, de sorte que nous avons activé la sortie respective lorsque le bouton est amené à la masse (GND).

Pour réaliser ce petit projet de démonstration, un code simple basé sur l'Arduino UNO a été écrit, en tirant parti de la bibliothèque Adafruit, qui, comme vous le voyez dans le **listage 1**, est incluse à la première ligne du croquis.

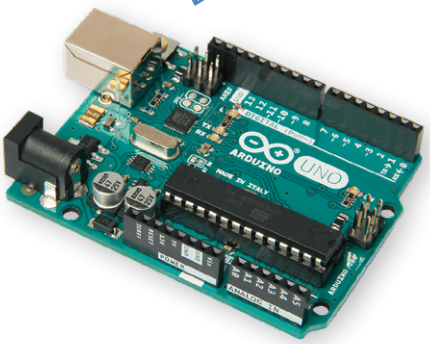
Avant de charger le code dans la mémoire programme de notre carte Arduino, il est essentiel de télécharger la bibliothèque à partir de *www.adafruit.com* et de l'installer en utilisant le gestionnaire de bibliothèque inclus dans l'EDI, ou simplement d'extraire le contenu du fichier ZIP et de copier le dossier *Adafruit_MCP23017_Arduino_Library* dans le répertoire *libraries* que l'on trouve normalement dans le système d'exploitation dans *Documents\Arduino\libraries*. Après avoir chargé la bibliothèque, il suffira de charger notre code d'exemple et de le télécharger dans la carte après avoir choisi le port COM correct dans le menu *Outils* de l'EDI.

Dans le code, un octet contenant l'état des boutons est envoyé au MCP23017 et les données correspondantes sont traitées puis écrites dans un octet dirigé vers le circuit intégré, qui définira l'état des lignes du port A de manière stable, jusqu'au rafraîchissement. Pour que l'interfaçage fonctionne, les commutateurs DIP A0, A1 et A2 doivent être réglés correctement, car si l'adresse 0x20 n'était pas attribuée au capteur, un message d'erreur serait affiché sur le port série ; l'adresse de la carte de liaison est attribuée avec la combinaison 000 des trois bits A0, A1, A2 (c'est-à-dire en fermant les trois commutateurs DIP à la masse). Si vous souhaitez modifier l'adresse, reportez-vous au tableau 1, étant entendu que vous devez également modifier l'adresse indiquée dans le croquis

VF : Denis Lafourcade — 230469-04

Produits

- **Arduino UNO Rev3**
  www.elektor.fr/15877



LIENS

[1] Page Web du MCP23017 de Microchip : https://microchip.com/en-us/product/mcp23017
[2] Ce Projet sur Elektor Labs : https://tinyurl.com/9sh3ct5t
[3] Fiche technique de Microchip : https://tinyurl.com/yc39n93v