ARDUINO
Une carte pour vos projets

CHRISTOPHE ULTRÉ [1]

Les projets interdisciplinaires impliquent de faire l'acquisition d'un matériel adaptable et peu coûteux qui réponde à des problématiques communes, comme le contrôle-commande ou l'interface homme-machine. La carte Arduino répond parfaitement à ces exigences. Découvrons-la à travers un exemple de projet.

mots-clés
automatismes, partie commande, programmation, projet

Le projet doit constituer un moment fort dans la scolarité des élèves de terminale. Leur implication et leur motivation se doivent d'être importantes. Il est donc nécessaire de leur offrir un maximum de liberté pour la concrétisation de leurs idées. Et cela commence par le choix du projet, qui doit répondre au moins à une des thématiques du référentiel : mobilité ; santé ; confort ; protection ; énergie ; environnement ; assistance au développement. Une fois un sujet choisi par chaque groupe d'élèves, les professeurs élaborent des notes de cadrage du projet à destination d'une commission académique de validation. Dès que cette commission donne son accord, l'aventure commence, ou plutôt continue...

Le projet

Une équipe d'élèves de terminale S option sciences de l'ingénieur du lycée Jacquard, dans le 19e arrondissement de Paris, s'est lancée dans un projet créatif et original : une boîte aux lettres qui communique avec le facteur par RFID. Cette activité de conception s'inscrit dans le cadre institutionnel du projet interdisciplinaire de terminale.

La technologie à mettre en œuvre devant être peu coûteuse, le choix matériel s'est rapidement porté sur une carte de contrôle de type Arduino pour assurer l'automatisation du produit.

Lors de la constitution du cahier des charges, les élèves ont proposé que la boîte aux lettres :

– renseigne, de la façon le plus simple possible, l'utilisateur sur la quantité de courrier reçu. La solution retenue a été de placer une balance électronique au fond de la boîte ;

– interdise à toute personne non autorisée la possibilité d'y mettre quelque chose (par exemple de la publicité), tout en laissant la possibilité au facteur d'y accéder. La solution retenue a été de déverrouiller l'accès à la boîte pour le facteur grâce à une carte RFID La Poste. Le cas exceptionnel où une personne autre que le facteur aurait à déposer quelque chose dans la boîte a été envisagé. La solution retenue consiste à utiliser la fonction « communication en champ proche » (NFC, pour *near field communication*) du lecteur de cartes RFID pour reconnaître le téléphone mobile de cette personne.

[1] Professeur d'électrotechnique au lycée Jacquard de Paris (75019).

L'équipe pédagogique encadrant le projet, composée de professeurs en sciences industrielles de l'ingénieur et d'un professeur de physique, y a ajouté une dimension : une recherche sur la manière d'alimenter tous les équipements de la boîte aux lettres en énergie électrique et éventuellement, pour une habitation individuelle, de la rendre indépendante de ce point de vue ; elle a proposé au groupe d'élèves que la fonction « traiter » de la chaîne d'information (qui reçoit des informations des capteurs et qui pilote les préactionneurs de la boîte aux lettres) soit assurée par un type de microcontrôleurs présent sur les cartes Arduino.

Pourquoi les cartes Arduino ?

Les aspects matériels

Dès son origine, ce projet se devait d'être économique, car les clients potentiels ne vont pas dépenser beaucoup d'argent pour ajouter des fonctions à leur boîte aux lettres. Par conséquent, il était nécessaire de trouver un dispositif capable de piloter le système et de communiquer avec l'utilisateur à un prix raisonnable (environ 20 €). De plus, il fallait trouver le moyen de le programmer simplement sans être un expert en langage informatique. Les cartes Arduino répondent à ces deux attentes – nous verrons comment. Et, petit plus, elles sont fabriquées dans un pays de la Communauté européenne : l'Italie.

Signalons que, si les cartes Arduino possèdent leur propre interface de programmation, la version 5 du logiciel FlowCode possède en option le compilateur pour cartes Arduino, de même que Matlab et LabView.

Nous allons ici nous intéresser à la carte Arduino Uno, mais il en existe d'autres avec des caractéristiques différentes : ATmega2560, Leonardo, Micro et la dernière en date, DUE.

Les aspects pédagogiques

La programmation des cartes Arduino est aisée, et l'ensemble des outils nécessaires au développement des applications est *open source*, c'est-à-dire que tous les schémas, programmes, bibliothèques de périphériques, etc., sont 100 % accessibles en consultation et/ou en téléchargement sur internet. La communauté des personnes utilisant ces cartes est très nombreuse, et une recherche internet aboutit généralement à des ébauches de solutions mises en ligne. On trouve aussi de plus en plus d'ouvrages traitant de ce type de cartes (voir le « S-SI le mag » p. 14). Par conséquent, en prérequis, un apprentissage minimal des fonctions algorithmiques de base et une initiation aux microcontrôleurs sont suffisants pour l'exploiter.

Comme nous allons le voir, l'utilisation de macrofonctions permet la concrétisation aisée d'une idée sans que l'on soit informaticien. Ces cartes sont donc un outil parfaitement adapté pour le prototypage.

Les retours d'expérience d'équipes pédagogiques d'autres établissements ayant déjà utilisé des cartes Arduino en enseignement technologique prébac ont conforté notre choix : les cartes Arduino sont fiables, robustes et faciles à programmer.

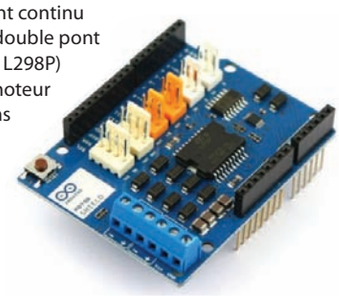
La carte Arduino Uno

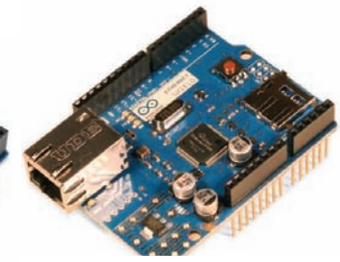
Quand on regarde une carte Arduino Uno, on est dérouté par sa sobriété ! On y retrouve le microcontrôleur entouré du minimum de composants nécessaires à son fonctionnement et à sa programmation 1. La carte, de conception très robuste, tient dans la main (75 × 54 × 15 mm). Autre avantage, une connectique très astucieuse permet aux cartes périphériques – baptisées *shields* 2 en langage Arduino – de venir se raccorder sur la carte principale en formant une sorte de mille-feuille.

Quelques précisions techniques :

- La gestion du port USB se fait grâce à un microcontrôleur indépendant qui établit une liaison série RS232 entre la carte et le PC, ce qui diminue la capacité des transferts. Si cet aspect constitue un inconvénient, il faudra utiliser la carte Arduino Leonardo qui permet une gestion native du port USB, étant vue par le PC comme un périphérique HID (*human interface device*).
- La connexion AREF permet d'utiliser une tension de référence autre que les 5 volts de la carte (configuration par défaut). L'utilisation de cette fonction nécessite quelques précautions (voir les données du fabricant).
- Le symbole ~ qui précède le numéro de certaines connexions du port d'entrées/sorties numérique désigne leur capacité à délivrer un signal PWM (pour *pulse width modulation*, ou MLI pour modulation de largeur d'impulsion). Le microcontrôleur ne possédant pas de convertisseur numérique-analogique, le seul moyen de disposer d'une tension variable dépendant d'une donnée numérique est de générer un signal rectangulaire de fréquence 490 Hz dont le rapport cyclique est variable. La donnée numérique pourra aller de 0 à 255 et correspondra à un rapport cyclique allant de 0 à 100 %.

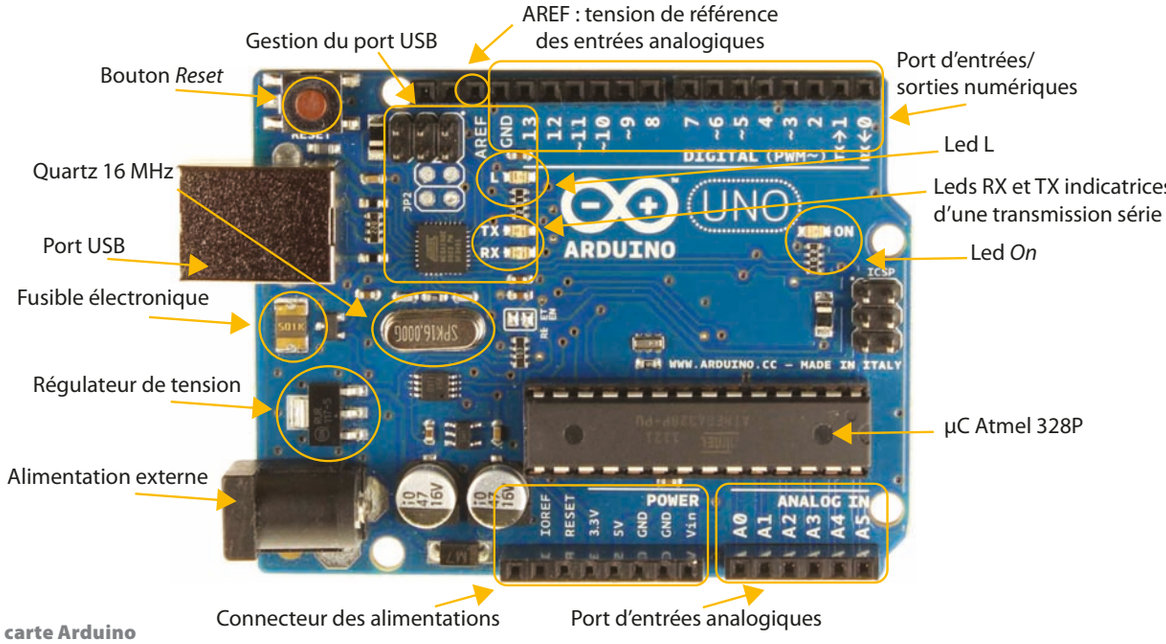
a La carte Motor Shield Rev3 qui permet de commander 2 moteurs à courant continu par un double pont en H (CI L298P) ou un moteur pas à pas





b La carte Ethernet Shield

2 Exemples de *shields*



1 La carte Arduino

● La led L est directement reliée à la connexion 13, elle permet donc de vérifier l'état d'une sortie d'un programme sans que l'on ait à effectuer le moindre câblage.
● Une sortie de l'Arduino Uno peut délivrer un courant d'une intensité de 40 mA, cependant l'intensité totale de toutes les sorties ne pourra pas excéder 200 mA.
● La carte possède 6 entrées analogiques associables avec un convertisseur 10 bits. La durée de la conversion analogique-numérique est d'environ 100 ms.
● La connexion Vin permet de récupérer, en liaison directe, la tension de l'alimentation externe.

Le câblage d'un *shield*

Pour le projet, il a fallu associer à la carte Arduino Uno un *shield* RFID pour l'autorisation d'accès. Un *shield* LCD 2 x 16 (2 lignes, 16 colonnes) a été ajouté pour l'affichage des informations utiles à l'utilisateur. Les deux *shields* communiquent via le bus série I²C, et leur alimentation est fournie par la carte mère. Il n'y a aucun câblage à faire, il suffit d'empiler les cartes 3. Pour des raisons évidentes de lisibilité, le *shield* LCD est installé au sommet de la pile.

Remarque : Il faut vérifier, lors de l'achat d'un *shield* compatible Arduino, que les connecteurs permettant d'empiler les cartes sont fournis, car ce n'est pas systématiquement le cas.

La programmation

Illustrons par un exemple (sans lien avec la boîte aux lettres intelligente) la simplicité de programmation de la carte Arduino.

Le langage utilisé n'est pas propriétaire, il utilise les instructions et la structure du langage C et du langage C++ (plus récent). Les instructions disponibles sont limitées, car adaptées aux possibilités du microcontrôleur. L'interface de programmation 4 est téléchargeable en version française à cette adresse :

http://arduino.cc/fr/Main/DebuterInstallation

Notons que ce logiciel est portable, c'est-à-dire qu'on peut l'installer sur une clé USB, par exemple. Il permet de programmer n'importe quel microcontrôleur avec un langage de haut niveau sans restrictions pour un coût nul.

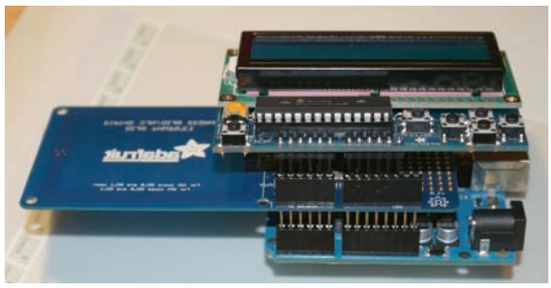
De plus, les *shields* ou les périphériques que l'on vient ajouter à la carte mère possèdent tous leur bibliothèque (*library*), téléchargeable ou présente par défaut dans le compilateur. Cette bibliothèque fonctionne exactement comme le pilote d'un périphérique de PC. Au niveau de la programmation, ce pilote va nous permettre d'utiliser des macrofonctions qui ne nécessitent pas de réelles compétences en programmation. Par conséquent, il n'est pas nécessaire d'entrer dans les détails de la structure du traitement du périphérique… mais un niveau minimal d'anglais technique est souhaitable.

La structure du programme, classique, est baptisé *sketch,* ou croquis dans la version française. Le programme commence par la déclaration des variables et par l'appel aux bibliothèques (si nécessaire). Puis viennent deux parties :
● La partie `void setup( )` est la partie du programme qui ne va s'exécuter qu'une seule fois, à la mise en route (ou après un *reset,* ou quand on se connecte au port COM). On y écrit généralement les fonctions d'initialisation.
● La partie `void loop( )` est une boucle infinie dans laquelle le programme écrit s'exécute tant que la carte est alimentée.

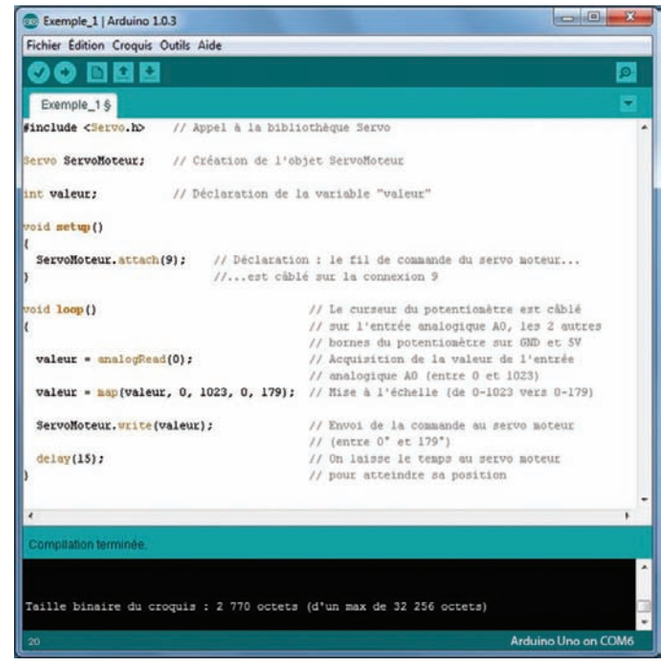
Le programmeur peut également écrire ses propres fonctions annexes, qu'il pourra utiliser dans les fonctions `setup( )` et `loop( )`.

Remarque : On vérifiera avant de lancer un téléversement du programme vers le microcontrôleur que la carte Arduino et le port COM utilisés sont correctement sélectionnés dans le menu Outils. Ces renseignements sont rappelés en bas à droite de l'interface de programmation 4.

Le programme 4 et le câblage 5 permettent de piloter un servomoteur en fonction de la position d'un potentiomètre. Le convertisseur analogique-numérique convertit la tension présente à l'entrée AO allant de 0



3 Le mille-feuille de cartes



4 Le programme de l'exemple dans la version 1.0.3 du logiciel de programmation

à 5 V issue du potentiomètre en une grandeur codée sur 10 bits allant de 0 à 1023. La macrofonction `ServoMoteur.write(valeur)` pilote intégralement le servomoteur dans les deux sens. Nul besoin de connaître la logique de commande des servomoteurs en fonction de la largeur et de la fréquence des impulsions.

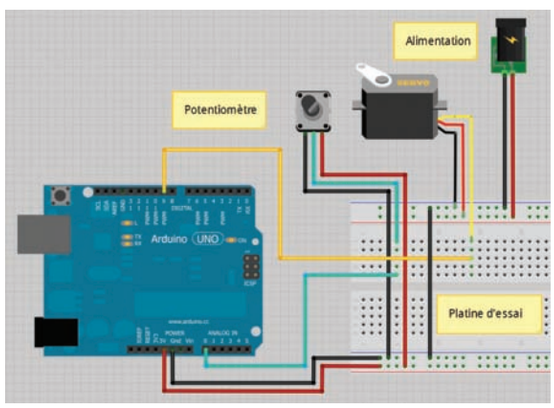
Remarque : la macrofonction `ServoMoteur.write(valeur)` ayant besoin, en argument, de la variable valeur de 0 à 179, nous avons procédé à une mise à l'échelle avec la fonction :

```
map (donnée, limite_basse_source, limite_haute_source, limite_basse_destination, limite_haute_destination);
```

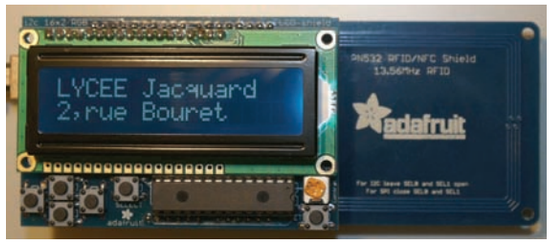
Nous constatons dans cet exemple que la mobilisation des connaissances et la réflexion principale vont vers l'architecture du programme et du prototype. Il n'y a pas de perte de temps en programmation fastidieuse, et en quelques lignes on est capable de piloter un petit système ludique.

Une production des élèves

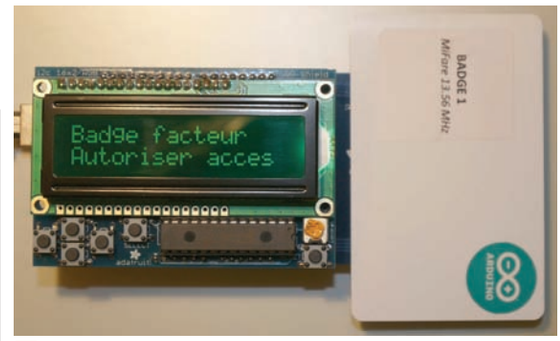
Pour le projet la boîte aux lettres, l'encadré montre une partie de programme qui permet d'obtenir les résultats illustrés par les figures de 6 à 9. Elle concerne la résolution du problème de l'identification du facteur. Ce programme semble plus complexe que celui de l'exemple précédent, mais son écriture n'a pourtant pas nécessité énormément plus de travail. En effet, les élèves sont partis des exemples fournis avec les bibliothèques du *shield* RFID et du *shield* LCD, en ont analysé les solutions, puis ils ont modifié les programmes en fonction de ce dont ils avaient besoin.



5 Le câblage nécessaire au programme de l'exemple
(schéma réalisé avec le logiciel gratuit Fritzing)



6 L'affichage de l'adresse de la boîte aux lettres



7 La présentation du badge « facteur »

8 La présentation d'un badge inconnu

Dans un premier temps, le programme teste que le *shield* RFID est bien connecté à la carte Arduino, puis il se met en attente d'un badge et affiche l'adresse de la boîte aux lettres en bleu sur l'écran LCD 6.

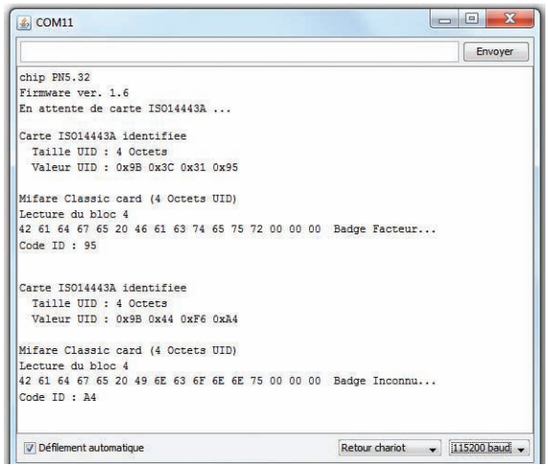
Lorsque l'on présente un badge à proximité de l'antenne du *shield* RFID, le programme l'identifie et récupère les 4 octets de son UID (*Unique IDentification number*). L'identification du facteur se fait grâce au 4e octet de l'UID de son badge (la valeur hexadécimale 0x95 dans notre cas). L'UID du badge est stocké de façon permanente dans la puce et n'est pas modifiable. Pour simplifier la reconnaissance du badge, les élèves ont choisi de ne tester que le dernier octet.

Par contre, le contenu du bloc 4 étant modifiable par l'utilisateur, les élèves y ont inscrit (avec un autre programme) « Badge Facteur » 7 et « Badge Inconnu » 8 en guise de vérification visuelle immédiate des différents badges sur le « moniteur série » de dialogue avec le microcontrôleur 9 (voir l'encadré).

Si le badge du facteur est identifié, le programme affiche en vert sur l'écran LCD que l'accès à la boîte aux lettres est autorisé 7. Pour tout autre badge, il affiche en rouge que l'accès à la boîte aux lettres est interdit 8.

La fin du programme permet aux élèves d'être avertis, par un affichage sur le moniteur série, d'un éventuel problème avec le badge ou qu'un badge est reconnu par le lecteur mais incompatible avec cette application.

Remarque : L'interface de programmation ne permet pas de simuler le fonctionnement d'un programme. C'est pourquoi, dans une phase de mise au point, il est très utile d'insérer dans le programme des lignes qui vont permettre d'envoyer et de recevoir des informations sur le moniteur série – en fait une fenêtre qui s'ouvre sur un PC permettant d'afficher



9 **Le moniteur série de dialogue avec le microcontrôleur**

des messages texte reçus de la carte Arduino et d'envoyer des caractères vers la carte Arduino. Le dialogue entre la carte et le PC s'effectue par une liaison série.

Une séduisante simplicité

Le choix de la technologie Arduino s'est révélé très fructueux : après quelques tâtonnements, les élèves affectés à ce projet de boîte aux lettres intelligente ont rapidement obtenu des résultats concluants. Nous avons également constaté un intérêt des élèves des autres groupes, pour le projet desquels le choix des solutions technologiques n'était pas complètement fixé, devant la facilité d'utilisation d'un tel matériel. Ils ont été séduits par les progrès rapides de ce groupe et ont tous voulu intégrer les cartes Arduino dans leur projet. Par exemple, dans le cadre d'un projet de maison thermodynamique, des élèves ont réalisé le pilotage en puissance du module à effet Peltier ainsi que l'acquisition et le traitement des mesures de température à l'aide de ce type de cartes. ■

Pour aller plus loin

TAVERNIER (Christian), *Arduino : Maîtrisez sa programmation et ses cartes d'interface (« shields »)*, Dunod, 2011

www.arduino.cc/fr
www.adafruit.com
https://github.com/adafruit/Adafruit-PN532
https://github.com/adafruit/Adafruit-RGB-LCD-Shield-Library
www.fritzing.org

Remerciements

À Bruno Guilbert (professeur en STS IRIS au lycée Jacquard) pour son assistance technique sur les cartes Arduino et la programmation en C++, et également aux élèves du groupe « boîte aux lettres intelligente » : Gustave Corpet, Samy Kettani, Bilel Souabni et Ugo Vidberg.

Programme de gestion d'accès par badge RFID avec afficheur LCD

```
#include <Wire.h>
#include <Adafruit_NFCShield_I2C.h>        // Appel aux bibliothèques
#include <Adafruit_MCP23017.h>
#include <Adafruit_RGBLCDShield.h>

#define ROUGE   0x1                        // Couleurs d'éclairage de l'afficheur
#define VERT    0x2
#define BLANC   0x7

Adafruit_RGBLCDShield lcd = Adafruit_RGBLCDShield();
Adafruit_NFCShield_I2C nfc( 2, 3);         // Déclaration du protocole I2C

void setup(void) {

Serial.begin( 1152 00);
lcd.begin(16, 2);                          // Attribution des dimensions de l'afficheur LCD
nfc.begin();

uint32_t versiondata = nfc.getFirmwareVersion();
if (! versiondata) {
      Serial.print ("Pas de carte PN53x");
      while (1);                           //  Arrêt si pas de shield NFC/RFID reconnu
      }
                                           // Renseignements à propos de la carte PN53x
Serial.print (" Chip PN5 ."); Serial.println ((versiondata>>24) & 0xFF,   HEX);
Serial.print (" Firmware ver.  "); Serial.print ((versiondata>>16) & 0xFF,   DEC);
Serial.print (' .'); Serial.println ((versiondata>>8) & 0xFF,   DEC);

nfc.SAMConfig();                           //  Configuration pour lire la carte RFID
Serial.println (" En attente de carte ISO14443A  ..."); // Voir fig. 9
}

void loop(void) {

int IDcarte = 0;
boolean acquisition;                       // Identification de la carte
uint8_t uid[] = { 0, 0, 0, 0, 0, 0, 0 };   // Tableau pour retourner l'adresse UID
uint8_t uidLength;                         // Taille de l'UID (4 ou 7 octets)
lcd.clear();                               // Effacer l'écran LCD
lcd.print (" LYCÉE Jacquard ");            // Affichage de l'adresse sur l'écran LCD
lcd.setCursor (0, 1);                      // de la boite aux lettres
lcd.print (" 2, rue Bouret ");                          // Voir fig. 6
lcd.setBacklight (BLANC);

                                           // En attente de la carte
acquisition = nfc.readPassiveTargetID(PN532_MIFARE_ISO14443A, uid, &uidLength);
if (acquisition) {
      Serial.println (" Carte ISO14443A identifiée  ");   // Info sur la carte détectée
      Serial.print ("  Taille UID :  "); Serial.print (uidLength, DEC); Serial.println (" Octets");

if (uidLength == 4) {                      //  Si l'UID de la carte comporte 4 octets
      IDcarte  =uid[3];                    // Génération du code d'identification
      Serial.println (" Mifare Classic  (4 Octets UID) ");   // Carte Mifare Classic
      uint8_t keya[6] = { 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF };
      acquisition = nfc.mifareclassic_AuthenticateBlock(uid,uidLength, 4, 0, keya);
      if (acquisition)  {
            Serial.println (" Lecture du bloc 4 ");
            uint8_t data[16];
            nfc.Print HexChar (data, 16);          // Vérification du bloc 4 et du code
            Serial.println (" Code ID : "); Serial.print (IDcarte,  HEX); Serial.println ("");

            if (IDcarte ==0x95 ) {                 //  Code de la carte du facteur
                  lcd.clear ();
                  lcd.setCursor (0, 0);
                  lcd.print (" Badge facteur ");           //  Voir fig. 7
                  lcd.setCursor (0, 1);
                  lcd.print (" Autoriser accè s");
                  lcd.setBacklight (VERT);                 // Rétroéclairage vert
                  delay(6000);
                  }
            else {                                 //  Tous les autres codes
                  lcd.clear ();
                  lcd.setCursor (0, 0);
                  lcd.print (" Badge inconnu");            //  Voir fig. 8
                  lcd.setCursor (0, 1);
                  lcd.print (" Accès interdit ");
                  lcd.setBacklight (ROUGE);                // Rétroéclairage rouge
                  delay(6000);}
                  }
      else {
            Serial.println (" Secteur illisible ") ;       // Prise en compte d'autres cas
            }                                              // de figure
      }
else {
      Serial.println (" DÉFAUT d'identification  ");
      }
}

if (uidLength == 7) {
      Serial.println (" Mifare Ultralight (7 Octets UID) non utilisée ");
      }
}
```