Rendez vos projets communicants

FABRICE LASNE [1]

Peut-on mettre en œuvre (assez simplement) une application Android sur son mobile pour commander une carte Arduino sans passer par une connexion Internet ? C'est possible, grâce à l'Arduino Yùn.

mots-clés
hardware, projet, système d'exploitation, programmation

Cet article propose de réaliser un projet domotique pour la commande d'un portier programmable avec des badges RFID. Développé en 1re STI2D ITEC (Innovation technologique et écoconception) dans le cadre d'un projet tutoré, il peut aussi être utilisé pour un enseignement transversal ou un projet STI2D SIN (Systèmes d'information et numérique). Ce projet met en œuvre un capteur RFID, du wifi, du bluetooth et une base de données MySQL ; quant à la programmation, elle est réalisée en PHP pour produire des pages web dynamiques, en C et Python pour l'Arduino Yùn, le tout pour moins de 150 € !

Nous introduirons uniquement le début des programmes longs, avec un code de couleur identique à celui du tutoriel ; pour la totalité du programme, nous vous indiquerons la page web correspondante. Le code couleur utilisé est le suivant :

– commande pour l'installation des modules php et MySQL pour le système en bleu gris à taper dans un terminal Linux (*Putty*) ;

– commande pour créer les tables MySQL en bleu gris à taper dans un terminal Linux (*Putty*) ;

– code en Python à stocker sur la carte SD du Yùn dans le répertoire mnt/sda1 en vert ;

– code Arduino en bleu ;

– code PHP pour les pages dynamiques en rouge à stocker sur la carte SD dans le répertoire www de la carte SD de l'Arduino Yùn (WSCP) ; l'ensemble des pages avec le squelette est disponible sur le site.

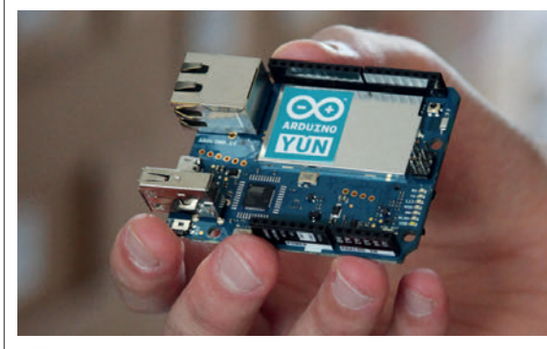
Objectif du projet

Dans un hôpital psychiatrique, les patients sont autorisés ou non à sortir en fonction de l'avis de leur médecin traitant. Celui-ci doit avoir un moyen simple et rapide de mettre à jour la liste des patients à tout instant, ainsi que leurs autorisations de sortie. On se propose de mettre à disposition du médecin une application Android pour qu'à partir de son smartphone il puisse gérer en direct les autorisations et s'en servir comme d'une télécommande. Les badges d'accès sont stockés à l'accueil ou en possession des patients, selon leur état de santé.

Matériel mis en œuvre

Pour mettre en œuvre le projet, nous allons nous appuyer sur du matériel ayant fait ses preuves dans le cadre des enseignements STI2D : une carte Arduino, présentée dans le n° 186 de *Technologie* « Une carte pour vos projets ». Mais pour utiliser un site web accessible en wifi, il faut une carte qui dispose d'un réseau wifi et un noyau Linux embarqué.

L'Arduino Yùn 1 est constitué d'une base Leonardo à laquelle a été ajouté en parallèle un processeur Atheros AR9331 qui utilise une distribution Linux Linino basée sur OpenWrt. Le processeur Atheros AR9331 est un processeur MIPS (architecture de type RISC, pour *Reduced Instruction Set Computer*), qui fonctionne à 400 MHz, avec une connexion Ethernet 100 Mb/s et Wifi 802.11b sur la bande des 2,4 GHz. Il est équipé également d'un port USB Host ou Device, dispose de 64 Mo de RAM et de 16 Mo d'espace de stockage flash



1 **La carte Arduino Yùn**

En ligne

Un tutoriel pas à pas ainsi que les programmes sont disponibles sur : http://eduscol.education.fr/sti/node/6809

Retrouvez tous les liens sur http://eduscol.education.fr/sti/revue-technologie

[1] Professeur en STS au lycée Louis-Armand, Paris (75015).

(dont une grande partie est occupée par la distribution Linux, il reste environ 7 Mo pour l'utilisateur), ainsi que d'un lecteur de carte micro-SD pour le stockage des pages web et du code en Python.

Si l'on prend en compte le coût des cartes shield Ethernet, l'Arduino Yùn reste très compétitif :

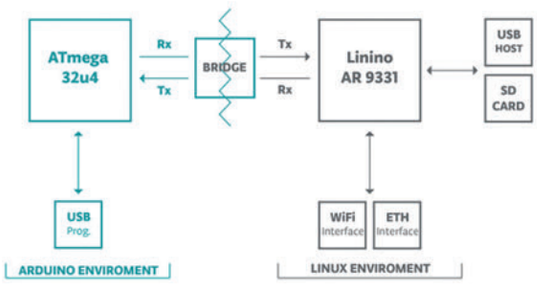
- Arduino Uno + shield Wifi : 20,00 € HT + 69,00 € HT = 89,00 € HT ;
- Arduino Yùn : 60,00 € HT.

Les environnements Arduino et Yùn ne sont pas complètement isolés l'un de l'autre, ils peuvent communiquer au moyen de la bibliothèque Bridge présente du côté Arduino et du coté Linux via une bibliothèque Python. La communication se fait par une liaison série entre les deux processeurs 2.

La communication entre les deux processeurs est bidirectionnelle. Elle permet de soulager les ressources de l'Arduino ; tous les fichiers situés dans un répertoire /arduino/www/ sur la carte micro-SD sont automatiquement disponibles pour le serveur web à l'adresse http://<ip>/sd/. Il est également possible d'exécuter des scripts (Shell, Python, Lua, PHP, Ruby, etc.) pour avoir des pages dynamiques et héberger une base MySQL.

Attention, il ne dispose pas de régulateur 5V intégré, il faut l'alimenter à partir d'un connecteur micro-USB de type B ou par les bornes Vin et +5V de la carte, en respectant scrupuleusement le niveau de tension.

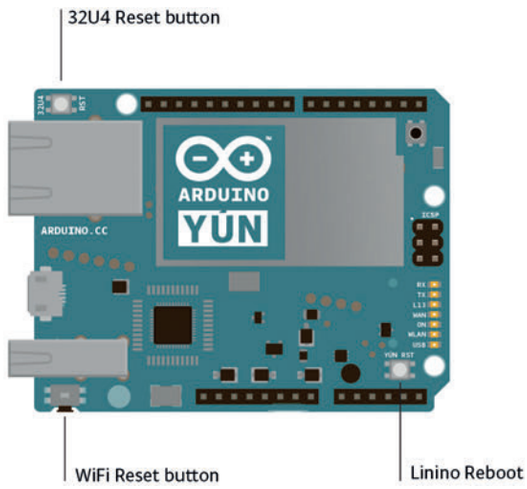
Il dispose de trois boutons Reset avec chacun sa fonction 3 : pour redémarrer l'AR9331, pressez le bouton Linino Reboot ; pour redémarrer l'Arduino, presser le bouton 32U4 Reset et pour redémarrer le wifi, presser le bouton WiFi Reset plus de 5 secondes, mais moins de 30 secondes pour réinitialiser l'adresse réseau.



2 Schéma de liaison entre Linux et Arduino (Bridge)

Le Yùn créera alors son propre réseau wifi « Arduino Yùn-XXXXXXXXXX ». Si on appuie plus de 30 secondes sur le bouton WiFi Reset, on remet le Yùn dans sa configuration usine, cela efface tous les fichiers installés et les configurations réseau ; le Yùn met alors environ 1 minute pour redémarrer.

On utilisera également les composants suivants : un lecteur de carte RFID 4, un module Bluetooth 5, un afficheur 6, des LED 7.



3 Les trois boutons « reset » de la carte

4 Lecteur et badge RFID

5 Module bluetooth

6 Afficheur LCD

7 LED de couleur

Principe de fonctionnement du projet

On utilise un lecteur de badge RFID RC522 branché sur l'Arduino Yùn qui lit l'identifiant unique du badge. Cet identifiant est transmis à la base de données par une fonction écrite en C (environnement Arduino) qui fait appel à un script en Python stocké sur la carte SD pour la recherche dans la base MySQL (environnement Linux) en utilisant le Bridge 2.

L'analyse de la base de données locale nous dit si le badge est référencé ou non. Si le numéro du badge existe, on vérifie si son propriétaire existe et s'il a le droit ou non de sortir ; en fonction du résultat, la fonction retourne une valeur 0, 1 ou 2.

Les droits peuvent être modifiés à tout moment à partir d'une application Android ou d'un site web 8. L'Arduino Yùn dispose d'un noyau Linux, héberge son serveur LAMP (Linux Apache MySQL PHP) et peut exécuter un script en Python en lien avec la base de données grâce à la bibliothèque Bridge. À partir de l'application Android ou d'un client web, le médecin se connecte sur l'interface wifi de l'Arduino Yùn, valide ou non les droits de sortie en cochant la case correspondante, la lecture des badges interroge la base et vérifie si la sortie est autorisée ou non. En cas de refus, une LED rouge s'allume et la porte reste fermée ; dans le cas contraire, une LED verte s'allume et la porte s'ouvre. En plus, les heures de sortie des patients sont enregistrées dans une autre table de la base de données et on affiche le nom du patient sur l'écran. Une liaison bluetooth permet d'ouvrir directement la porte à partir d'une application Android.

On utilise le port SPI (ICSP) de l'Arduino Yùn pour l'alimentation du lecteur RFID ; la broche 10 est utilisée pour le SDA et le module est alimenté en 3,3 V. Le schéma de câblage complet avec l'afficheur et les LED est réalisé avec le logiciel gratuit Fritzing 9. On utilise la bibliothèque RFID pour faire fonctionner l'Arduino avec le lecteur RC522 disponible à l'adresse : https://github.com/miguelbalboa/rfid

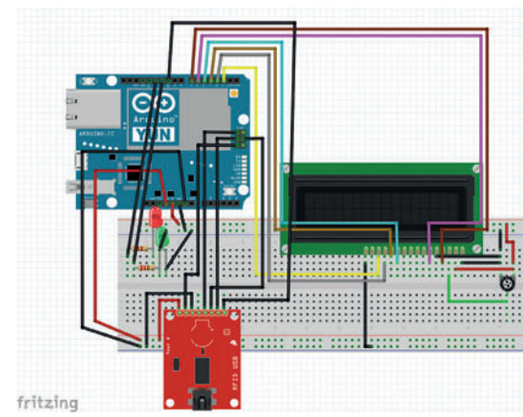
Un conseil : faire un test avec le programme d'exemple pour vérifier le fonctionnement qui doit vous permettre de visualiser le contenu des valeurs stockées dans la carte Mifare en la passant devant le lecteur 10.

Réalisation du programme de gestion des badges

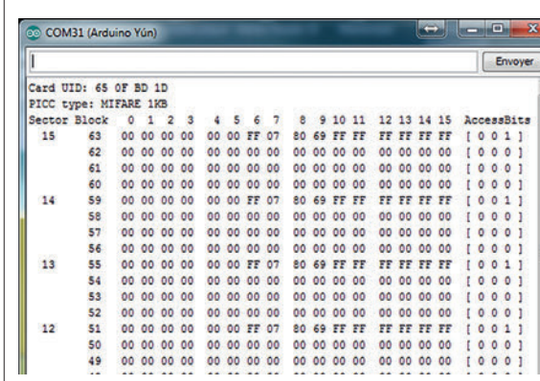
Il y a trois façons de se connecter à l'Arduino Yùn 11 : avec la liaison série en connectant un cordon USB au PC, avec le programme Arduino et l'interface série du Yùn ou bien avec un câble RJ45. Si le réseau possède un DHCP, on peut se connecter sur le serveur web du Yùn à l'adresse 192.168.0.47 (Arduino Yùn) 12. Avec un navigateur, on peut configurer la liaison wifi – le password par défaut est « arduino » – et le wifi en master à l'adresse 192.168.240.1. Une fois connecté, on peut modifier les paramètres de la carte, y compris en mode avancé avec le menu Luci très complet.

La dernière méthode consiste à utiliser le mode master du wifi de l'Arduino Yùn, qui nous fournit une adresse IP en 192.168.240.x. Une fois les programmes transférés par le réseau local ou la liaison série, on se connecte avec un appareil wifi sur le réseau Yùn sans avoir besoin de connexion Internet, seul le site hébergé par le Yùn est accessible 13. Le wifi apparaît sous le nom « Arduino Yùn-XXXXXXXX ».

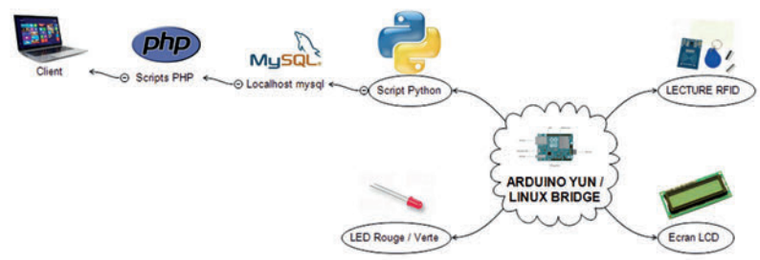
Pour gérer les sorties des patients, il faut disposer d'une base de données ; le plus simple est d'installer MySQL sur le serveur Linux de l'Arduino. Pour installer des paquets sur l'Arduino, on peut utiliser un client



9 Schéma de câblage du projet



10 Vérification des informations de la carte MIFARE



11 Synoptique du projet sous la forme de carte mentale

8 Interactions entre les différents acteurs

SSH 14 tel que *Putty* et faire une mise à jour du système avec OPKG update ; la procédure complète est détaillée dans le tutoriel.

```
opkg install libpthread libncurses libreadline Mysql-server
sed -i 's,^datadir.*,datadir      = /srv/Mysql/,g' /etc/my.cnf
sed -i 's,^tmpdir.*,tmpdir              = /tmp/,g' /etc/my.cnf
mkdir -p /srv/Mysql
Mysql_install_db --force
```

Une fois MySQL installé, nous allons créer deux tables : l'une avec les noms, les numéros RFID et l'autorisation ; l'autre avec les noms et les heures de sortie. Nous créons une base *patients* avec « pass » comme password et deux tables *patientsrfid* et *controle* ; nous utilisons les commandes MySQL suivantes :

```
Mysql -u root -p patients
CREATE TABLE `patientsrfid` ( `id` int(255) NOT NULL
AUTO_INCREMENT, `nombre` varchar(300) NOT NULL,
`nom` varchar(300) NOT NULL, `prenom` varchar(300)
NOT NULL, `rfid` varchar(300) NOT NULL, `autorisation`
varchar(300) NOT NULL,`date` varchar(300) NOT NULL,
`heure` varchar(300) NOT NULL, PRIMARY KEY (`id`) )
ENGINE=MyISAM DEFAULT CHARSET=latin1
AUTO_INCREMENT=0;
CREATE TABLE `controle` ( `id` int(255) NOT NULL
AUTO_INCREMENT, `rfid` varchar(300) NOT NULL,`heure`
varchar(300) NOT NULL,`date` varchar(300) NOT NULL,
PRIMARY KEY (`id`) ) ENGINE=MyISAM DEFAULT
CHARSET=latin1 AUTO_INCREMENT=0;
```

Pour traiter les requêtes à partir d'un serveur web, il faut installer PHP5 :

```
opkg install php5 php5-cgi
```

Autoriser le PHP dans Linino pour ensuite installer les passerelles entre PHP et Python :

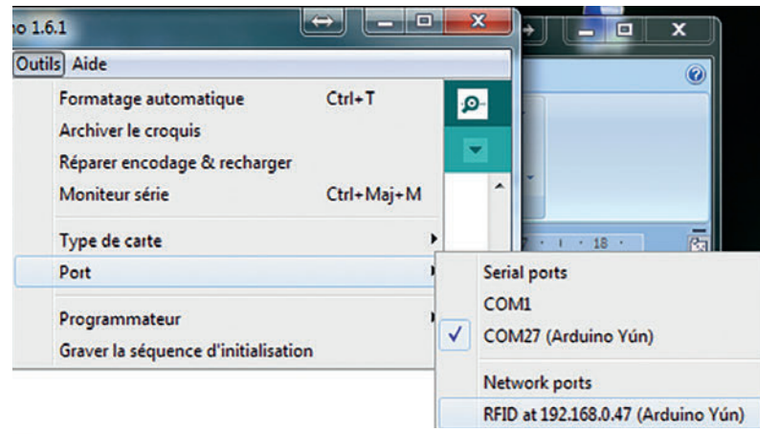
```
opkg install php5-mod-Mysql
sed -i 's,;extension=Mysql.so,extension=Mysql.so,g' /etc/
php.ini
opkg install python-Mysql
```

Une fois PHP, MySQL et Python installés et configurés, nous pouvons commencer le programme de lecture des badges, dans le croquis Arduino. La bibliothèque Bridge.h permet la liaison entre Linux et Arduino.

Exemple de lecture de badge

Nous vérifions si une carte est présente devant le lecteur de carte, puis nous déterminons le GUID de celle-ci.

```
// Look for new cards
if ( ! mfrc522.PICC_IsNewCardPresent())
  {
```

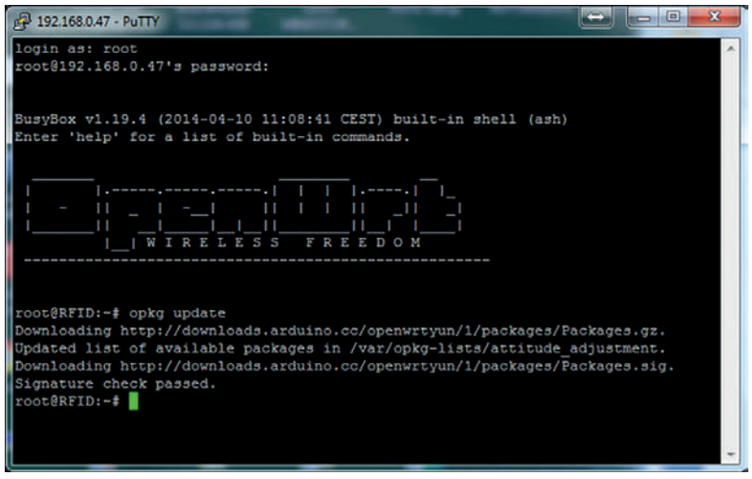


12 **Accès à la carte Arduino par adressage IP**

13 **Vérification de l'accès sans fil**

```
    return;
  }
  // Select one of the cards
if ( ! mfrc522.PICC_ReadCardSerial())
  {
    return;
  }
for (byte i = 0; i < mfrc522.uid.size; i++)
  {
   contenu.concat(String(mfrc522.uid.uidByte[i]
   < 0x10 ? "0" : ""));
   contenu.concat(String(mfrc522.uid.uidByte[i], HEX));
    }
```

Puis nous vérifions dans la base MySQL, en appelant un script en Python auquel on passe en paramètre, le n° IDrfid de la carte qui vient de passer devant le lecteur de badges.



14 **Console Linux de l'Arduino Yùn**

```
void verifRFID(String IDrfid){
Process verif;
  verif.begin(«python»);
  verif.addParameter("/mnt/sda1/RFID/valid.py");
  verif.addParameter(IDrfid);
                    // Appel de la fonction en Python valid.py
verif.run();
```

Le programme en Python vérifie si le n° IDrfid est reconnu dans la base de données et s'il a le droit de sortie ou pas. Ce programme *valid.py* est stocké à un endroit bien précis sur la carte SD : /mnt/sda1/RFID ; nous le transférons via un programme FTP 15.

```
import MySQLdb
import sys
host = "127.0.0.1"
user = "root"
passw = "pass"
base = «patients»
while True:
        db= MySQLdb.connect(host,user,passw,base)
        cur = db.cursor()
```

suite page 13 du tutoriel

Ce script nous renvoie une valeur 0, 1 ou 2 suivie du nom du patient qui sera affiché sur l'écran. Le résultat de cette fonction est retourné sous forme de chaîne de caractère dont on enlève les espaces.

```
String resultatPython = verif.readString();
                    // lecture du résultat de la fonction
resultatPython.trim();
                                    // 0 ou 1 ou 2 en string
String val = resultatPython.substring(0,1);
   Serial.println(val);
```

Nous retournons dans le sketch ou croquis pour le traitement en fonction du résultat de la requête :

```
if (val == «1»){                    //Badge vérifié et OK
Serial.println("Acces Autorise");
   resultatPython = resultatPython.substring(5,(resultatPython.
length()-5));
      Serial.println(resultatPython);
      Serial.println();
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("Bonjour M ");
      lcd.print(resultatPython);
      lcd.setCursor(0,1);
      lcd.print(«Ouverture porte!»);
      digitalWrite(13, HIGH);       // LED vert allumée
delay(3000);
      controle(IDrfid);             // ajout de l'heure de sortie
                                                dans la base
mensageminicial();
      break;
    }
```

Pour enregistrer les heures de sortie des patients, nous faisons appel à la fonction contrôle suivante :

```
void controle(String IDrfid){
Serial.println("ajout contrôle");
  Process control;
  control.begin("python");
  control.addParameter("/mnt/sda1/RFID/control.py");
  control.addParameter(IDrfid);     // Appel de la fonction
          contrôle en Python qui ajoute l'heure de sortie
control.run();
}
```

Le script Python *control.py* suivant va remplir la table des sorties avec le n° IDrfid de la carte.

```
import MySQLdb
import sys
import time

host = "127.0.0.1"
user = «root»
passw = «pass»
base = «patients»
temps = time.strftime("%d/%m/%Y %H:%M:%S", time.
localtime())
db= MySQLdb.connect(host,user,passw,base)
cur = db.cursor()
resultat = cur.execute(" INSERT INTO controle (rfid,heure)
VALUES (%s,%s) ",(sys.argv[1],temps,))
if (resultat == 1 ):
      print 1
sys.exit(1)
else:
      print 2
sys.exit(1)
```

La fonction retourne 1 si l'opération s'est déroulée correctement.

Pour créer et gérer les accès, nous allons développer des pages en PHP qui vont alimenter la base de données. Nous allons créer une interface pour gérer les accès à partir de pages web hébergées dans l'Arduino, une page pour valider les autorisations, une page pour le contrôle des entrées/sorties et une dernière page pour entrer les noms et les numéros des badges.

Réalisation du site

La première page sera celle qui autorise les sorties avec les cases à cocher index.html.

Nous listons les entrées de la base et on coche les patients autorisés à sortir ; la trame sera constituée des fichiers PHP habillés à l'aide d'un squelette de site en java script aisément disponible sur le net.

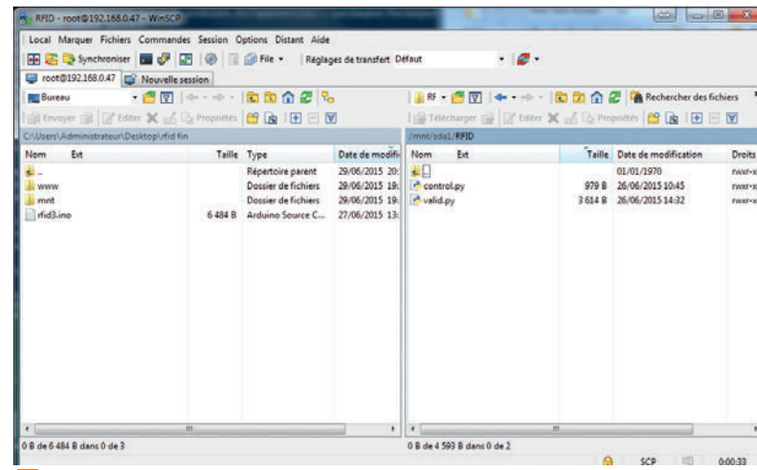
```
<?php
include('config.php');          //on inclut le fichier
                    de configuration avec les mots de passe
if(isset($_POST['id']))         //la condition sera true s'il y a
                                              un tableau d'id


{
$listId=$_POST['id'];
   if(isset($_POST['CBTest']))              // s'il y a au moins
                                              un checkbox coché
{
;
```

suite page 24 du tutoriel

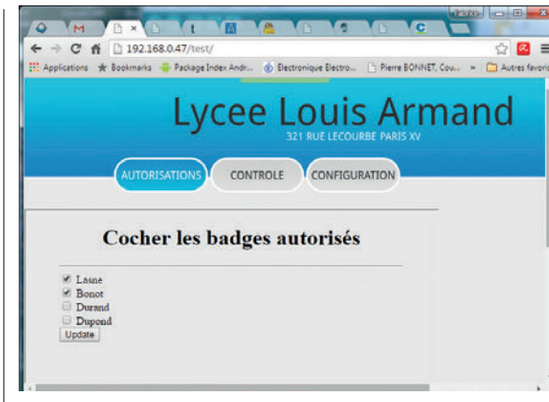
À l'ouverture de la page, nous vérifions si les autorisations sont actives ou non et nous vérifions chaque case lors de la validation ; il y a quelques subtilités à prendre en compte avec les cases à cocher, comme le montre la capture d'écran 16. Une fois les modifications effectuées, nous devons valider les choix avec le bouton Update. Les modifications de droits sont enregistrées dans la base de données. Le fichier *config.php* vérifie les droits d'accès à la base.

```
<?php
                                  //Consultation base patients
$server=»127.0.0.1»;
$username="root";
$password="pass";
$database_name="patients";

                        //Connexion serveur base de données
$connexion=Mysql_connect($server, $username, $password)
or die(«Probleme de connexion base de données»);

                                //Sélection de la base patients
$bd_sel=Mysql_select_db($database_name) or die
(«Probleme de connexion base de données»);
?>
```

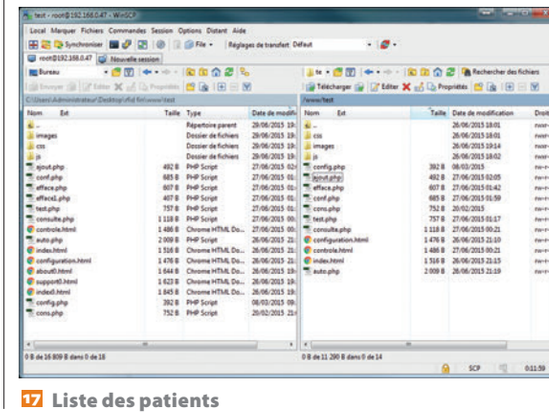
Pour la page de contrôle, nous proposons la liste des patients et les heures de sortie pour chaque patient 17. Pour gérer les attributions de badge, la page configuration permet de gérer les associations. Nous utilisons le programme avec la liaison série pour avoir le numéro des badges. L'ensemble des pages est disponible en téléchargement sur le site (avec un squelette en java script disponible en version gratuite). Le fichier *index.html* qui est issu du squelette de site trouvé sur le web fait appel à la balise iframe qui permet d'insérer la frame en PHP dans la page web.



15 Client FTP pour le transfert de programme



16 Écran d'accueil



17 Liste des patients

```
<!DOCTYPE html PUBLIC «-//W3C//DTD XHTML 1.0 Transi-
tional//EN» «http://www.w3.org/TR/xhtml1/DTD/xhtml1-tran-
sitional.dtd»>
<html xmlns=»http://www.w3.org/1999/xhtml»>
<head>
```

suite page 25 du tutoriel

Les fichiers pour le site web sont envoyés dans l'Arduino Yùn avec un client FTP, ici avec WinSCP, dans un répertoire test. Le lien vers le site est donc : http://192.168.0.47/test/index.html si on se connecte avec un navigateur en RJ45 (DHCP réseau en 192.168.x.x) ou en 192.168.0.1/test/index.html si on se connecte sur le wifi master de l'Arduino Yùn.

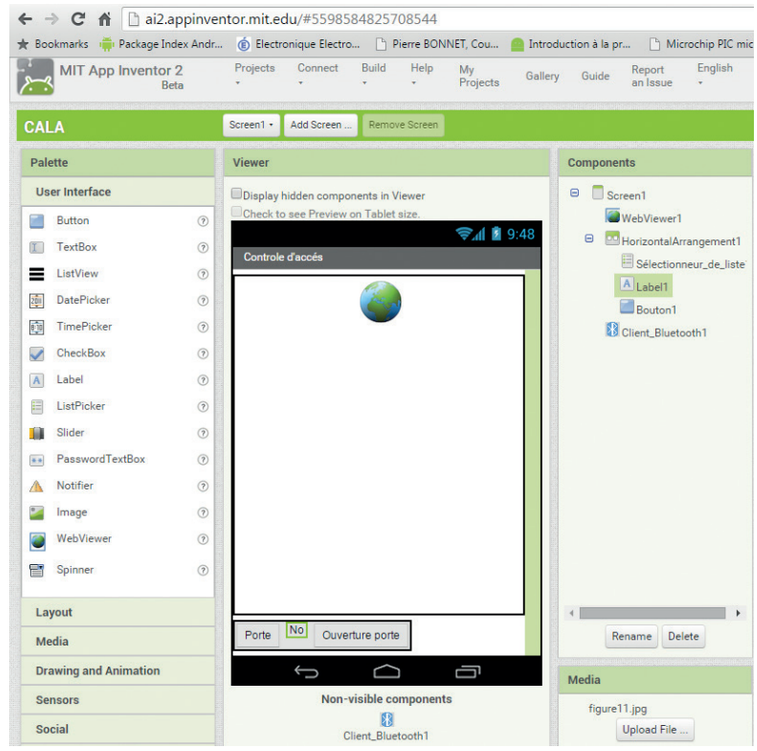
L'application Android

Pour réaliser l'application Android 18, le plus rapide est de passer par le site AppInventor : http://appinventor.mit.edu/explore/

Une fois inscrit avec un compte Google, nous créons un nouveau projet, ici CALA, on place un WebViewer sur la page avec comme HomeUrl le lien : http://192.168.240.1/test/index.html

Nous pouvons donner un titre à la fenêtre et une icône pour l'application 19. Nous ne définissons pour l'instant aucun bloc ; pour transférer le programme dans l'appareil, nous passons par l'application AI Companion. Dans le menu connecté, nous choisissons AI Companion : un code-barres apparaît, nous lançons l'application MIT AI2 Companion sur le téléphone Android en scannant le QRcode ; le programme est directement envoyé à l'appareil si bien sûr ils sont sur le même réseau wifi 20. Pour créer une application avec une icône sur l'appareil, nous utilisons « construire APP ».

Pour rajouter un bouton qui permet d'ouvrir la porte directement à partir du téléphone, il faut utiliser une connexion bluetooth. J'utilise un HC06 en 3,3 V, donc avec un pont diviseur sur la patte RXD, les pattes 10 et 11 de l'Arduino (même montage mais avec les pattes 10 et 11 à la place des pattes 1 et 2) 21.



19 **Capture d'écran de la configuration AppInventor**

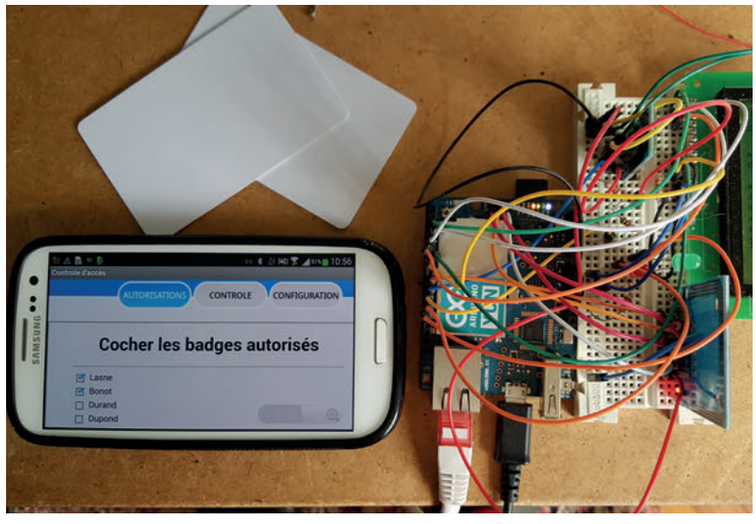
```
// Broches utilisées pour la connexion du module blue
                                                tooth HC06
SoftwareSerial mySerial(10, 11); // RX, TX
String messageRecu;
```

Nous connectons le port série :

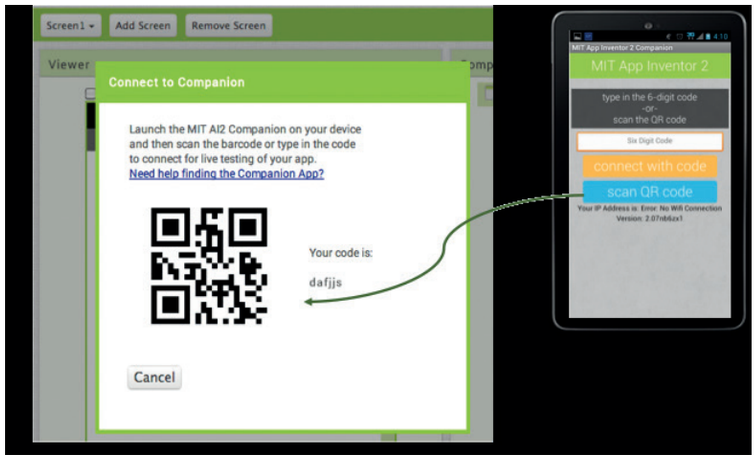
```
mySerial.begin(9600);   // Rajout d'un port com pour
    lecteur bluetooth (pas serial1 à cause du bridge)
```

Nous vérifions la présence de la commande CMD=ON\n :

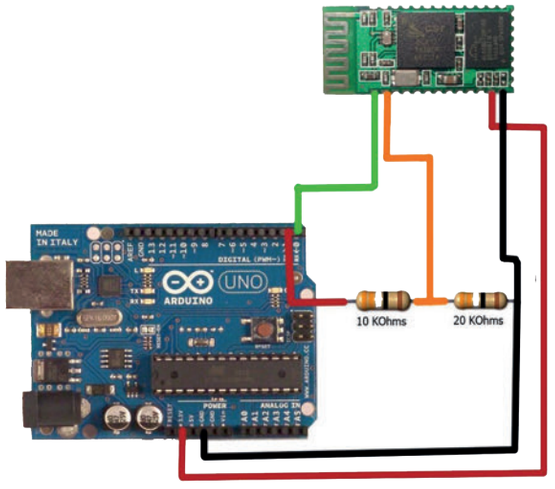
```
// if (mySerial.available() > 0) {serialA = mySerial.read();Serial.
println(serialA);}
        while(mySerial.available())
        {
         delay(3);
         char c = mySerial.read();
         messageRecu += c;
        }
        if (messageRecu.length() >0)
        {
         Serial.println(messageRecu);
          if (messageRecu == "CMD=ON\n")
```



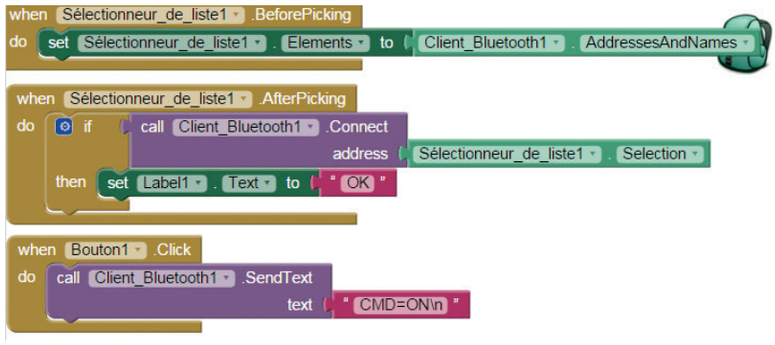
18 **Maquette complète de l'application Android**



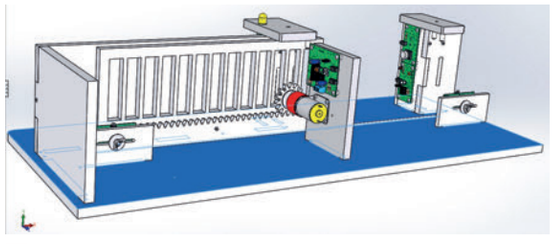
20 QRcode pour le transfert du programme sur le téléphone



21 Schéma de câblage de la carte bluetooth



22 Programme AppInventor



23 Maquette Solidworks du projet collaboratif

```
{digitalWrite(13, HIGH);
        delay(3000);
        Serial.println("led on");
        }
    }
```

Il suffit, dans l'application AppInventor, de rajouter un client bluetooth (menu connectivité avec les options par défaut) et de placer un sélecteur de liste pour se connecter au bluetooth 19.

Pour le code, nous sélectionnons le module bluetooth avec le bloc sélectionneur de ligne ; une fois connecté, nous utilisons la commande Send text du client bluetooth pour envoyer la chaîne de caractère CMD=ON de l'application au Yùn 22.

Conclusion

La mise en œuvre de ce projet ne présente pas de difficultés majeures, mais demande une certaine rigueur dans la mise en place des programmes, car le nombre de concepts abordés avec ce système est important, ce qui en fait la richesse :

– communication en bluetooth, liaison série, avec une application Android ;
– connexion RFID wifi et leurs paramétrages ;
– mise en place d'une base de données ;
– mise en place d'un site web dynamique ;
– programmation en C et en Python.

Il est actuellement utilisé en première année de BTS électrotechnique et sera amélioré dans le cadre d'un projet collaboratif avec un trinôme d'élèves en 1re STI2D pour la réalisation d'une maquette 23. La répartition proposée sera la suivante : un élève en spécialité EE (Énergie et environnement) chargé de la motorisation et de l'alimentation en énergie avec un panneau solaire, un élève en spécialité ITEC pour la réalisation de la partie mécanique, avec découpage laser pour les plaques et impression en 3D pour le pignon à partir d'un fichier Solidworks, et un élève en spécialité SIN pour une partie de la programmation (AppInventor avec le module bluetooth). ■

En ligne

Site officiel de la carte arduino Yùn :
http://www.arduino.cc/en/Main/ArduinoBoardYùn

Exemple d'utilisation des cartes RFID :
http://playground.arduino.cc/Learning/MFRC522/

Application ayant inspiré le projet :
http://www.instructables.com/id/Control-Access-of-Arduino-Y%C3%9AN-with-Mysql-PHP5-and-/

Retrouvez tous les liens sur http://eduscol.education.fr/sti/revue-technologie