Arduino YUN

Guide de démarrage (Sources MC HOBBY modifié)

Configuration du wifi de la carte lors de la première utilisation (ou changement de réseau)

1. Alimenter la carte avec un câble USB + une alimentation secteur (≥1A) (PB de déconnexion en Wifi si la carte est alimentée directement par le PC) et attendre que la Led USB s'éclaire (2mn maxi)
2. Activer la carte en point d'accès si elle n'y est pas (c.-à-d. actionner le bouton RST WLAN (wifi) entre 5 et 10s)
3. Se connecter à la carte dont le SSID doit être visible dans la liste des points d'accès du PC
4. Entrer **192.168.240.1** ou <nom>.local si le service Bonjour est installé <nom> = arduino lors de la première connexion. Entrer le mot de passe demandé (**arduino** lors de la première connexion)
5. Configurer la carte. Détails § Configuration du wifi de la carte … si nécessaire.
6. Effectuer une remise sous tension de la carte si elle ne se connecte pas.

Remarque : La led USB éclairée indique que la carte est prête (à la mise sous tension ou lors d'une reconfiguration.

Réseau wifi utilisé en SIN : voir l'annexe 3

Dernière révision le 7/5/2017

Table des matières

Introduction

Arduino Yún est une carte Arduino très différente des autres. Même si sa programmation est similaire à une **Arduino Leonardo** (*Anglais*, Arduino.cc) et qu'elle utilise le même processeur Atmel **ATmega32U4**, cette carte dispose également d'un processeur additionnel (**Atheros AR9331**) fonctionnant sous Linux avec une pile **Wifi OpenWrt**. La programmation de l'ATmega32U4 via l'USB est identique à celle d'une Arduino Leonardo. Une fois la carte Yún configurée pour se connecter à un réseau WiFi, vous pouvez l'utiliser pour programmer l'ATmega32U4. Pour connecter l'Arduino Yún à votre ordinateur, vous aurez besoin d'un câble **Micro-B** USB. Celui-ci fournit l'énergie et la connexion de donnée. Pour programmer une carte Yún, vous devez choisir **Arduino Yún** dans le menu **Outils > Carte** (**Tools > Board**) de l'IDE Arduino. **La carte Yún est supportée à partir de la version 1.5.4 d'Arduino IDE**.

Les différences avec une carte Arduino Leonardo

Crédit: Arduino arduino.cc

En plus de l'**ATmega32U4**, la carte Yún dispose d'un processeur **Atheros AR9331** fonctionnant avec la distribution **Linux Linino** basé sur **OpenWrt**. OpenWrt est un système Linux pour système embarqué. Linino inclus **Python 2.7.**

La carte Yún dispose d'un connecteur pour carte SD, d'un connecteur **Ethernet** (réseau filaire) et d'un connecteur **USB-A Hôte (Host)**. Il n'y a pas de connecteur d'alimentation car la carte peut être alimentée par le connecteur micro-USB (**Attention 1A min**).

Il n'y a pas de régulateur 5V sur la carte. Vous endommagerez votre Yún si vous l'alimentez avec une tension supérieure à 5V.

Si vous n'alimentez pas la carte Yun par l'intermédiaire de la connexion micro-USB, vous pouvez appliquer l'alimentation sur les broches **Vin et 5V**. **Il est cependant recommandé d'alimenter le Yún par l'intermédiaire de la connexion USB**.

Les connecteurs SD, Ethernet et USB-A sont tous physiquement connectés au processeur Linux AR9331 et non sur l'ATmega32U4.

Le processeur 32U4 du Yún fonctionne de la même façon que celui du Leonardo, excepté que **vous ne pouvez pas utiliser Serial1 car il est réservé à la communication avec le processeur AR9331 !**

La carte Yún dispose également d'un **module Wifi**. Il permet de la connecter à un routeur sans fil (Wifi) et de la **transformer en point d'accès wifi (AP)**. Quand la carte est utilisée en point d'accès, elle devient visible sur le réseau Wifi et il est alors possible de s'y connecter avec un ordinateur... exactement comme on le fait habituellement avec un routeur Wifi (un routeur WiFi est forcément un Point d'Accès).

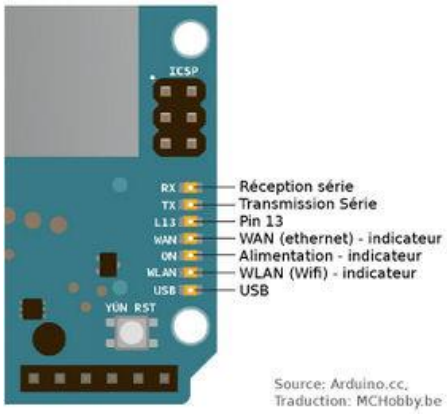
Le processeur ATmega32U4, le WiFi et le processeur Atheros AR9331 disposent chacun de leur propre **bouton "Reset**" (bouton de réinitialisation).

Crédit: Arduino arduino.cc Traduction: MCHobby.be

Il y a plusieurs LEDs de statut sur la carte Yun. Elles indiquent:

- L'état de l'alimentation,
- La connexion **WAN (réseau Ethernet filaire),**
- La connexion **WLAN (réseau Wireless... réseau sans fil),**
- La connexion au port USB.

En outre, **la broche 13** est connectée à une des LED d'état.



Crédit: Arduino arduino.cc Traduction: MCHobby.be

Linino

Le carte Yún exécute la distribution Linux **Linino**, basée sur **OpenWrt**. Même s'il est possible de configurer le système en ligne de commande, un serveur web embarqué permet de régler de nombreuses options. L'interface (appelée **LuCi**) permet d'accéder à la plupart des paramètres de gestion du module WiFi.

L'accès au serveur WEB est décrit plus loin dans ce document.

Pour installer des logiciels complémentaires sur une distribution Linino, il faut utiliser le **gestionnaire de paquet Linux opkg**. Des informations sur ce gestionnaire et les principales commandes sont disponibles sur la page du gestionnaire de paquet Yún.

On accède à Linino, en ligne de commande, par l'intermédiaire de la librairie Bridge d'Arduino (librairie de liaison), **avec une connexion SSH**.

Python

Python 2.7 est installé dans la distribution Linino. Avec Python, vous pouvez écrire des applications et des scripts.

Pour plus d'information sur Python :

- Le langage Python sur Wikipedia.fr.
- L'Association Francophone Python (AFPY)
- Apprenez à programmer en Python par Prolixe, sur fr.openclassrooms.com.
- Les pages de documentation sur Python 2.7 (*Anglais*, Pyhton.org).
- La section Python sur Developpez.com
- Les autres références se trouvent dans la section programmation du Raspberry Pi (MC Hobby).

Si vous apprenez Python, sachez qu'il existe de nombreuses et excellentes ressources sur le Net. Hormis les références ci-dessus, vous pouvez aussi consulter "Learn Python the Hard Way" (Apprendre Python par la pratique, *Anglais*, learnpythonthehardway.org) qui couvre tout ce que vous avez besoin de savoir pour commencer vos propres scripts.

Stockage externe sur la carte Yún

Il n'est pas conseillé d'utiliser la mémoire non-volatile interne de la carte Yún pour les opérations de stockage parce qu'elle a un nombre limité de cycles d'écriture.

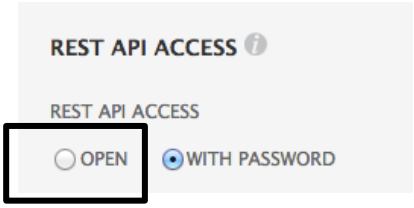
Vous pouvez utiliser une mémoire externe comme une carte microSD ou une clé USB pour stocker vos données, scripts, page web, etc.

Si vous voulez que l'ATmega32U4 de votre Yún stocke ou accède à ce périphérique, vous aurez besoin de créer un répertoire nommé *arduino* à la racine du volume.

Services WEB

Pour les clients et serveurs "Service WEB", Linino supporte REST. REST est l'acronyme de "**Representational State Transfer**" (*difficilement traduisible*). C'est une architecture logicielle exposant le matériel Arduino par l'intermédiaire d'URL.

Par défaut, les accès à l'interface de programmation REST (REST API) sont protégés par un mot de passe. Il est possible de modifier l'accès à ce service afin de ne pas utiliser de mot de passe. Pour modifier ce paramètre, entrer dans le panneau de configuration de la carte Yún. Vous trouverez l'interface permettant de basculer cette option d'accès en bas de la page.



Crédit: Arduino arduino.cc

Accès aux API (interface de programmation) REST. "**Open**" signifie que l'interface est *ouverte* donc librement accessible. "With Password" signifie qu'elle est protégée par un mot de passe.

Une introduction aux **concepts REST** est accessible ici (*Anglais*, StackOverflow.com) .

La carte Yún dispose de deux **points d'accès REST** (*REST end points*) reconnu:

POINTS d'ACCES

REST

- **/arduino**
- **/data**

Le répertoire "/arduino" n'est pas préconfiguré. Tout ce qui est ajouté dans l'URL derrière le point d'accès est transféré par le Serveur Web au sketch/croquis fonctionnant sur l'ATmega32U4. Vous pouvez définir vos propres interfaces de programmation (API) à l'intérieur de votre sketch/croquis. Consultez les exemples "Bridge" pour voir comment il est possible d'offrir un accès à une broche Arduino via l'interface REST.

Le répertoire "/data" est utilisé pour fournir un accès à un stockage interne de type **Clé/Valeur** (key/value). Les appels possibles sont:

- **/put/KEY/VALUE :** pour stocker une valeur *value* pour le clé *key*.
- **/get/KEY :** pour obtenir la valeur de la clé *key*. Retour au format JSON.
- **/get :** pour obtenir une liste des éléments stockés au format JSON.
- **/delete :** pour effacer le contenu du stockage interne

Reset des processeurs (AR9331, WiFi et ATmega32U4)

Reset Linino

Pour redémarrer le processeur AR9331 et Linino, pressez le bouton reset "**YÚN RST**". Il se trouve à côté des entrées analogiques et des LEDs d'état de la carte.

Reset Arduino

Pour redémarrer l'ATmega32U4 et votre sketch/croquis Arduino, pressez le bouton "**32U4 RST**" à côté du port Ethernet **deux fois**.

Reset WiFi

Le bouton reset du WiFi "**WLAN RST**" se trouve près du connecteur USB-A. Lorsque vous le pressez, la **LED WLAN commence à clignoter**.

Pour vous connectez à un nouveau réseau, appuyez sur "**WLAN RST**" **pendant plus de 5 secondes mais moins que 30 secondes**.

La configuration WiFi sera réinitialisée et le Yún redémarrera son propre réseau wiFi : Yún-XXXXXXXXXXXX. Toutes les autres modifications/configuration seront conservées.

Réinstallation de la distrubution Linino

Pour réinstaller la distribution Linino dans sa configuration d'usine (dite *default state*), pressez le bouton *reset* WiFi pendant au moins 30 secondes.

La carte reprendra sa configuration d'origine, celle qu'elle avait à sa sortie de la boite (où la dernière que vous auriez flashée).
Attention : Cette opération effacera tous les fichiers installés ainsi que la configuration réseau !

Ré-énumération série au Reset

Etant donné que la carte Yún ne dispose pas d'un composant dédié à la gestion de la communication série, cela implique que le port série est **virtuel** -- Cette communication est assurée par des routines logicielles sur la carte et sur votre ordinateur.

Cela fonctionne exactement comme lorsque vous connectez n'importe que Arduino à votre ordinateur. Celui-ci crée une instance du pilote de port série *virtuel* ET l'ATmega32U4 crée une instance de "port série" lorsqu'il démarre son bootloader. La carte devient alors une instance du pilote **USB's Connected Device Class (CDC)**.

Cela signifie que la connexion USB sera systématiquement perdue et ré-établie à chaque fois qu'il y a un *reset* du processeur ATmega32U4. La carte disparaitra de la liste des ports séries et cette liste sera ré-énumérée. Tous les programmes disposant d'une connexion série ouverte sur la carte Yun la perdront. C'est la différence avec une Arduino UNO puisqu'il est possible de faire un *reset* du processeur principal (the ATmega328P) sans perdre la connexion USB (maintenue par un second processeur ATmega8U2 ou ATmega16U2).

Ces différences ont des implications pour l'installation de pilotes, pour le télé versement de programmes et la communication.

Pas de reset à l'ouverture du port série

La carte Yún ne redémarrera pas votre sketch/croquis lorsque vous ouvrez le port série sur votre ordinateur. Cela signifie que vous ne verrez pas les données qui ont été envoyés par la carte vers l'ordinateur. C'est à dire la plupart des informations envoyées par la fonction setup().

Cela s'applique également à la Console, décrite ci-dessous.

Si vous utilisez n'importe quelle fonction Série (*Serial*) ou console avec print(), println() ou write() dans la fonction setup(), ces dernières n'afficherons rien lorsque vous ouvrirez le moniteur série ou une connexion Console.

Pour contourner ce problème, vous pouvez vérifier si le port est ouvert à l'aide de:

```
// Arrêter le programme jusqu'à ce que le moniteur série soit ouvert
while (!Serial);
```

ou

```
// Stopper le programme jusqu'à ce qu'une Console soit ouverte
while (!Console);
```

Qu'est-ce que la Console ?

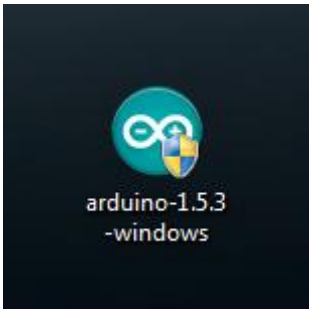
La **Console**, basée sur le *Bridge*, vous permet d'envoyer des informations depuis une carte Yún vers un ordinateur comme vous le feriez avec le moniteur série, mais grâce à la connexion sans fil.

Installation des pilotes pour la carte Yún

Windows

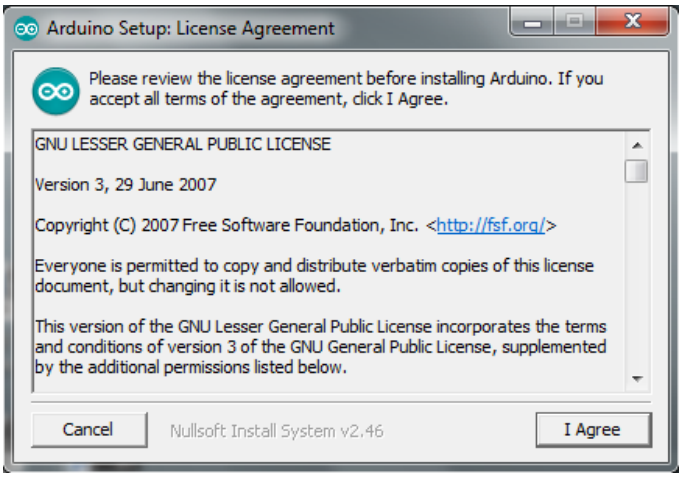
Windows dispose d'un programme d'installation pour l'IDE (*environnement de développement intégré*) et les pilotes (*drivers*).

Téléchargez le programme d'installation "Arduino IDE 1.5.4" **ou plus récent** et double cliquez sur l'icône d'installation.



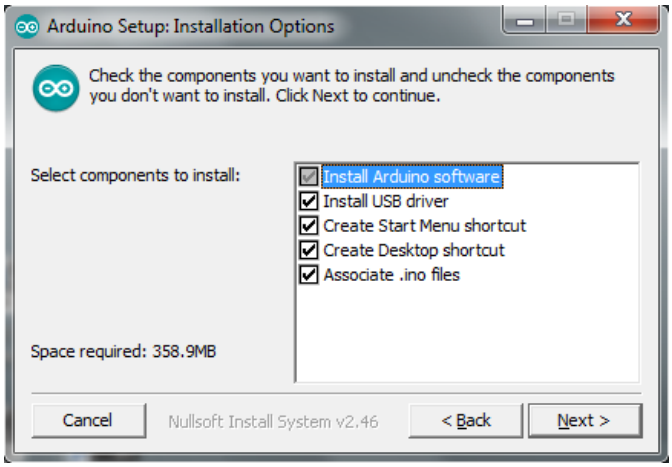
Crédit: Arduino arduino.cc

Une fois que vous avez pris connaissance de la licence, et si vous l'approuvez, cliquez sur le bouton "I agree" ("*Je suis d'accord*").



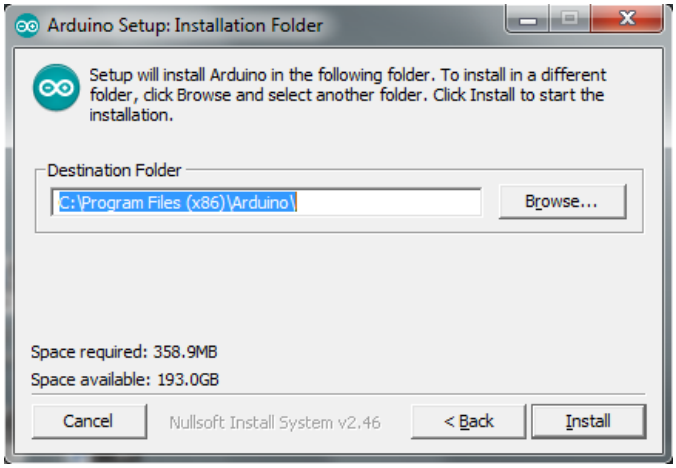
Crédit: Arduino arduino.cc

Tous les éléments sont sélectionnés par défaut. Cela inclus l'IDE, les pilotes (*drivers*) et les raccourcis (*shortcuts*)



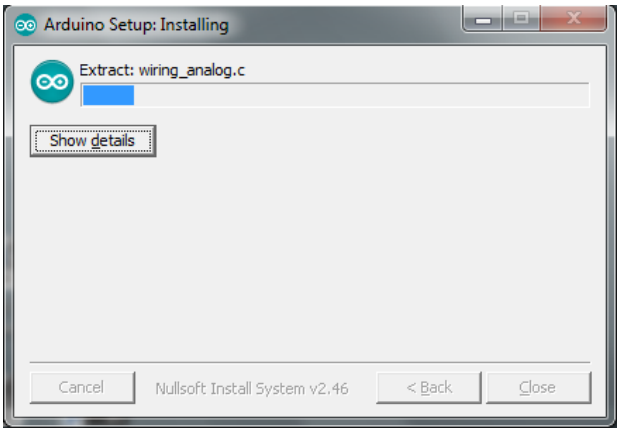
Crédit: Arduino arduino.cc

Sélectionnez l'endroit où vous désirez installer l'IDE (*environnement de développement intégré*).



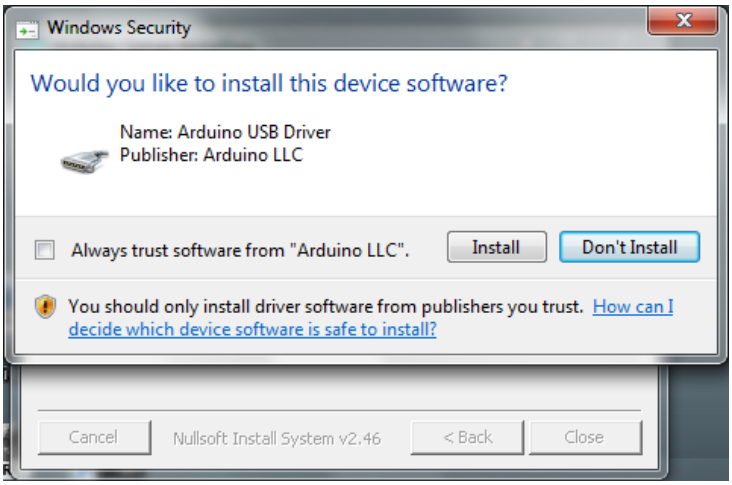
Crédit: Arduino arduino.cc

Le programme d'installation affiche une barre de progression pendant qu'il extrait et installe les fichiers à l'emplacement choisi.



Crédit: Arduino arduino.cc

Une fenêtre de confirmation s'affichera si vous décidez d'installer les pilotes (*drivers*). Vous devrez confirmer votre sélection.



Crédit: Arduino arduino.cc

Lorsque l'installation est terminée, vous pouvez presser le bouton "Close" (*Fermer* ou *Terminer*).

NB: La carte Yún utilise le service "**Bonjour"** pour permettre sa détection automatique sur les réseaux WiFi. Ce service n'est pas inclus par défaut dans les distributions de Windows. Si "**Bonjour"** n'est pas installé sur votre système, vous pouvez le télécharger ici. En cas de problème, il peut être nécessaire de configurer l'anti-virus ou le firewall pour ne pas bloquer les communications sur le **port 5353**. La carte est accessible en entrant : **http://<nom>.local**

Linux

Il n'est pas nécessaire d'installer les pilotes pour Ubuntu 10.0.4 ou suivant.

Configuration du WiFi de la carte

La carte Yún peut fonctionner en point d'accès (*Access Point*) et se connecter sur un réseau existant.

Elle peut se connecter sur des réseaux non protégés (*non cryptés*) et sur des réseaux crypter en **WEP, WPA et WPA2**.

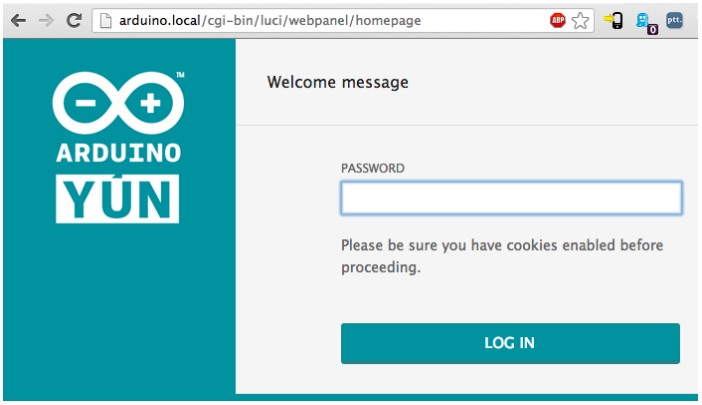
Lorsque vous mettez la carte sous tension pour la première fois, elle crée un réseau WiFi nommé **ArduinoYun-XXXXXXXXXXXX**.

Connectez votre ordinateur à ce réseau.

La YUN comme POINT d'ACCES

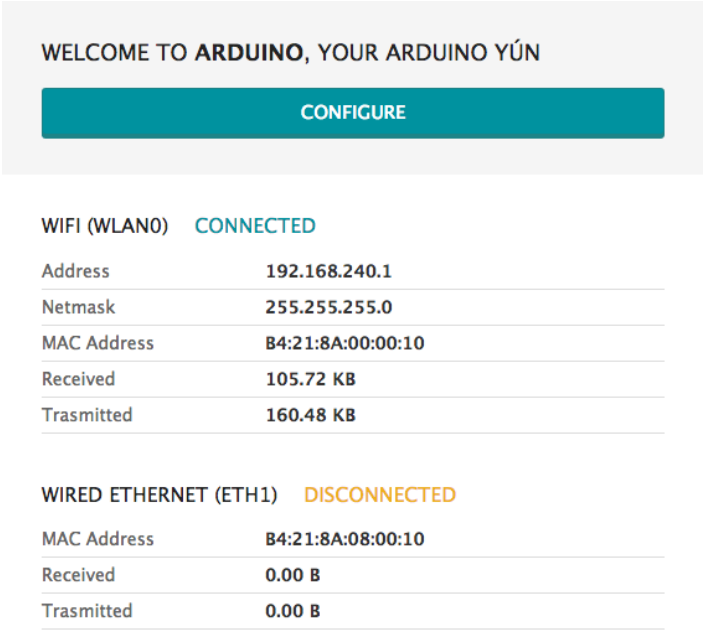
Une fois que la carte **Yún** a délivré une adresse IP à votre ordinateur, ouvrez un navigateur et entrez l'adresse http://arduino.local (si le service Bonjour est installé) ou **192.168.240.1**. Une page Web vous demandant un mot de passe (*password*) doit s'afficher.

Entrez "**arduino**" et cliquer sur "Log In" (*se connecter*).



Crédit: Arduino arduino.cc

Une page contenant des informations relatives aux connexions actuelles s'ouvre. Elle affiche les paramètres de l'interface Wifi et ceux de la connexion Ethernet. Pressez le bouton "**Configuration**" pour continuer.



Crédit: Arduino arduino.cc

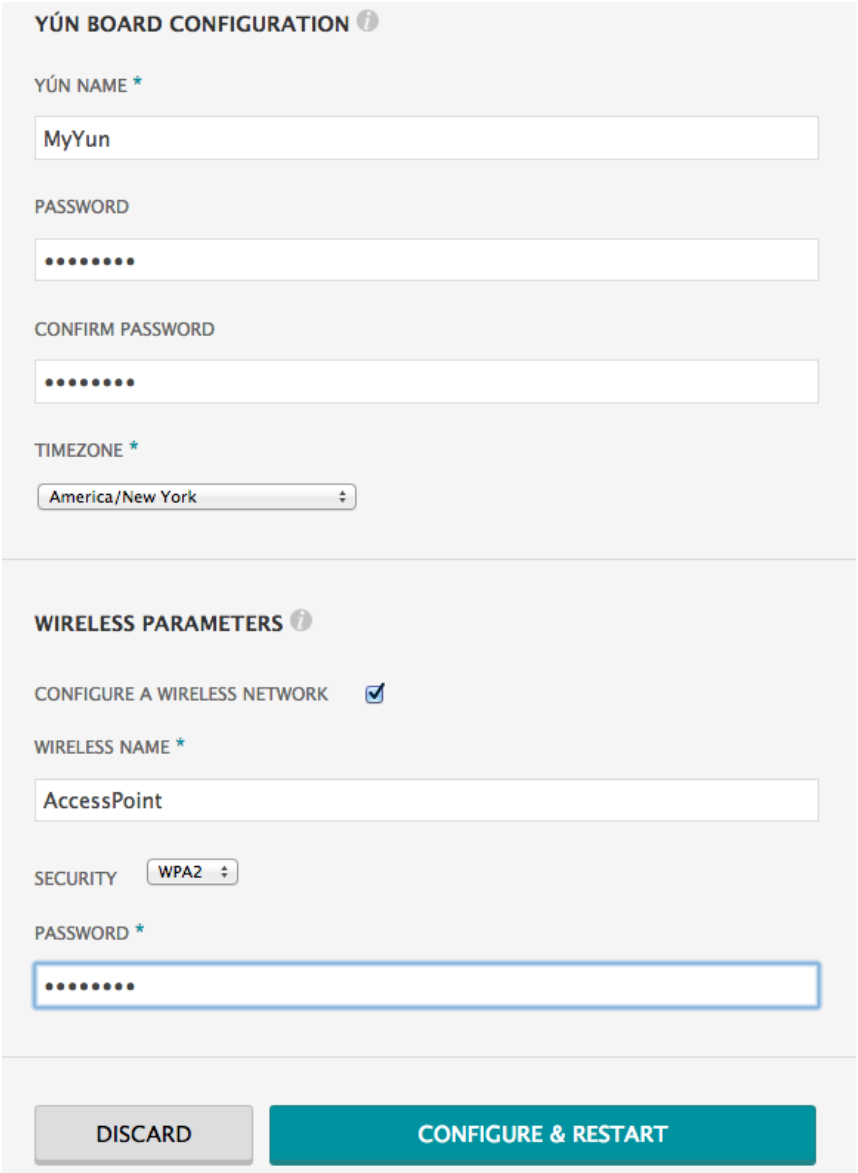
Configurerez la carte Yùn en lui donnant un **nom unique**.

Choisissez un mot de passe d'au moins **8 caractères**. Si vous laissez cette zone vide, le système maintiendra le mot de passe par défaut.

Vous pouvez également configurer le fuseau horaire (*Timezone* en anglais) et le pays (*country*). Il est recommandé de renseigner ces options car elles peuvent faciliter la connexion sur le réseau Wifi local. En effet, sélectionner un fuseau horaire (*timezone*) permet de présélectionner le domaine de la réglementation qui y est applicable (*country's regulatory domain*).

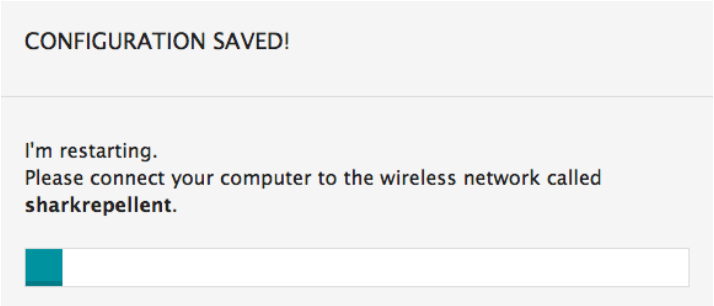
Entrez le nom du réseau WiFi sur lequel vous désirez qu'elle se connecte dans la zone "Wireless Name".

Sélectionnez le type de la sécurité à appliquer (*security type*) et entrez le mot de passe (*password*) applicable à cette connexion Wifi.



Crédit: Arduino arduino.cc

En pressant le bouton "Configure & Restart", la carte se réinitialise et se connecte sur le réseau spécifié. Le réseau Arduino (le *point d'accès*) s'interrompt au bout d'un moment. Le message "Configuration Saved !" indique que votre carte Yùn a enregistré les nouveaux paramètres et se connecte au réseau.



Crédit: Arduino arduino.cc

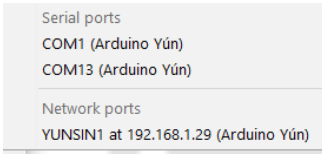
Vous pouvez maintenant connecter le PC au réseau sur lequel vient de se connecter la carte Yùn.

Note: pressez "Discard" si vous ne désirez pas appliquer ces nouveaux paramètres.

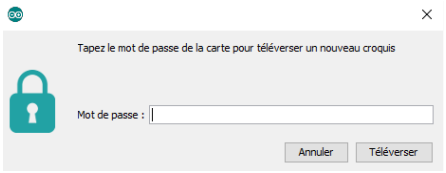
Programmer l'ATmega32U4 par l'intermédiaire du réseau sans fil WiFi

Si votre carte Yún est située sur le même réseau que votre ordinateur alors vous pouvez la programmer par l'intermédiaire du réseau sans fil.

Ouvrez l'IDE "Arduino".



- Dans le menu **Outils > Port** (*Tools* > *Port*), vous devriez voir la ou les cartes Yùn présentes sur le réseau (avec leur nom et leur adresse IP). Sélectionnez la vôtre.
- Sélectionnez la carte "**Arduino Yùn**" dans le menu **Outils > Cartes** (*Tools* > *Board*).
- Ouvrez l'exemple *Blink* (*Fichier* > *Exemples* > *01Basics* > *Blink*) et téléverser le croquis dans la carte. Le mot de passe administrateur de la carte Yùn vous sera demandé durant cette opération. utiliser le mot de passe configuré lors de son paramétrage.



Le processeur ATmega32U4 redémarre après le transfert du programme.

Vous devriez voir clignoter la LED branchée sur la broche 13.

Utiliser le connecteur Ethernet

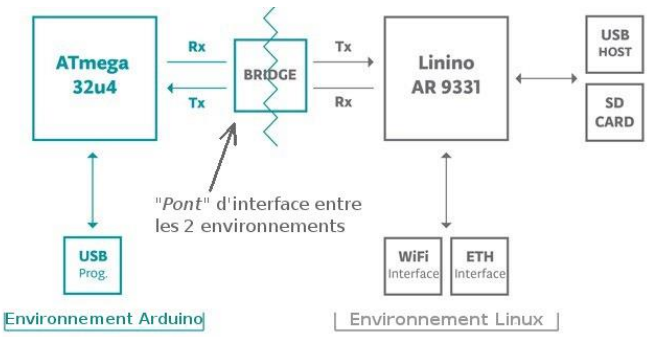
Lorsque vous connectez une carte Yùn sur un réseau avec un câble Ethernet, elle essaie de s'y connecter automatiquement (IP automatique). La carte est alors visible dans le menu *Port série* de l'IDE Arduino comme précédemment.

Si vous souhaiter connecter directement une carte Yùn à un PC, celui-ci doit avoir été configuré en serveur DHCP ou le PC est la carte doivent disposés d'une adresse IP statique. Dans les deux cas la connexion s'effectuera avec un câble croisé.

Note: l'interface **Ethernet est eth1** et non eth0

Communiquer avec la distribution Linino par l'intermédiaire du "Bridge"

La librairie "Bridge" (signifiant littéralement "pont") permet la **communication entre Arduino et Linino**. Des classes facilitent la communication entre ces deux environnements. Celles-ci sont décrites plus loin dans ce document mais également en détails dans les pages de la librairie Bridge (ou bien Bridge library reference Arduino.cc, Anglais).



Crédit: Arduino arduino.cc Traduction par MCHobby.be

La Console

La "Console", basée sur "Bridge", permet d'envoyer des informations depuis la carte Yún vers un ordinateur exactement comme si vous utilisiez le moniteur série (**sauf que c'est sans fil)**. La "Console" crée une connexion SSH sécurisée entre la carte Yùn et votre ordinateur.

Pour tester la console, télé versez le croquis ci-dessous sur votre carte. Les interfaces WiFi et Ethernet, le port USB et le support de carte SD sont tous connectés sur l'AR9331. La librairie "Bridge" permet de travailler avec ces périphériques, d'exécuter des scripts et de communiquer avec des WebServices.

```
#include <Console.h>

const int ledPin = 13; // La broche sur laquelle la LED est branchée
int incomingByte;      // Une variable pour lire les données séries entrantes

void setup() {
  // Initialisation de la communication série:
  Bridge.begin();
  Console.begin();

  while (!Console){
    ; //Attendre la connexion du port Console.
  }
  Console.println("Vous êtes connecté sur la Console!!!!");
  // Initialiser la broche de la LED comme sortie:
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // Vérifier s'il y a une donnée entrante:
  if (Console.available() > 0) {
    // Lire le byte (l'octet) le plus vieux stocké dans
    // la mémoire tampon série (serial buffer):
    incomingByte = Console.read();
    // Si c'est un H majuscule (ASCII 72), allumer la LED:
    if (incomingByte == 'H') {
      digitalWrite(ledPin, HIGH);
    }
    // Si c'est un L (ASCII 76), éteindre la LED:
    if (incomingByte == 'L') {
      digitalWrite(ledPin, LOW);
    }
  }
}
```

Pour voir la Console, sélectionnez le nom de votre carte Yun (+ son adresse IP) dans le menu Outils ->Ports.

Remarque : La carte Yún est visible si l'ordinateur est sur le même réseau (LAN). Vous ne la verrez pas si elle se trouve sur un réseau différent !

Ouvrez le moniteur dans Outils -> Moniteur Série. Entrez le mot de passe de la carte Yùn.

Vous pouvez également accéder à la Console en ouvrant une fenêtre « **terminal** » et en tapant la commande:

```
ssh root@yourYunsName.local 'telnet localhost 6571'
```

Puis presser la touche Retour/Entrée.

Note: Si vous utilisez Windows alors vous devrez installer un émulateur de terminal. PuTTY est un choix raisonnable mais vous devrez entrer les deux commandes ci-dessus séparément.

Tapez 'H' pour allumer la LED sur la broche 13 et 'L' pour l'éteindre.

Les commandes Process

Les commandes *Process* permettent **d'exécuter des processus Linux sur Linino par l'intermédiaire d'Arduino**.

Dans l'exemple suivant, Linino va se connecter sur un serveur en utilisant curl et télécharger du texte ASCII. Il affiche ensuite le texte sur la connexion série.

```
#include <Process.h>

void setup() {
  // Initialisation du Bridge
  Bridge.begin();
  // Initialisation du port série
  Serial.begin(9600);
  // Attendre qu'un moniteur série soit connecté.
  while (!Serial);
  // Exécuter les différents processus
  runCurl();
}
void loop() {
  // Rien à faire.
}

void runCurl() {
  // Exécuter la commande "curl" et obtenir un logo Arduino en "ascii art" depuis le Net
  // curl est un programme en ligne de commande pour transférer des données en utilisant
  // différents protocols Internet
  Process p;        // Créer un processus nommé p
  p.begin("curl");  // Le processus démarre la commande "curl"
  p.addParameter("http://arduino.cc/asciilogo.txt"); // Ajouter le paramètre URL à "curl"
  p.run();      // Exécuter le processus et attendre la fin de son exécution
  // Afficher le logo Arduino sur le port série
  // La sortie d'un processus (''process output'') peut être lue à l'aide
  // des méthodes de streaming (gestion de flux)
  while (p.available()>0) {
    char c = p.read();
    Serial.print(c);
  }
  // S'assurer que les derniers bits de donnée sont bien envoyés
Serial.flush();}
```

Bridge = échange entre les processeurs

Le Bridge permet de passer des informations entre les deux processeurs en utilisant des paires **clé/valeur** ("*key/value pairing*").

Cet exemple montre comment utiliser la bibliothèque Bridge pour accéder aux broches analogiques et digitales de la carte par l'intermédiaire d'appels REST (*Wikipedia.fr*). Il montre comment créer votre propre API en utilisant le style REST pour faire des appels par l'intermédiaire d'un navigateur internet.

Assurez-vous que votre ordinateur partage le même réseau que votre Yùn.

Lorsque la carte est programmée, vous pouvez demander la valeur d'une broche, écrire une valeur sur une broche et configurer une broche comme une entrée ou une sortie.

Lorsque le mot de passe REST est désactivé, vous pouvez transmettre les URL suivantes en utilisant votre navigateur:

- http://myArduinoYun.local/arduino/digital/13 : effectue un *digitalRead(13);*
- http://myArduinoYun.local/arduino/digital/13/1 : effectue un *digitalWrite(13,1);*
- http://myArduinoYun.local/arduino/analog/9/123 : effectue un *analogWrite(9,123);* (donc un contrôle PWM en sortie)
- http://myArduinoYun.local/arduino/analog/2 : effectue un *analogRead(2);* pour effectuer une lecture analogique.
- http://myArduinoYun.local/arduino/mode/13/input : effectue un *pinMode(13, INPUT);* pour assigner une broche en entrée.
- http://myArduinoYun.local/arduino/mode/13/output : effectue une *pinMode(13, OUTPUT);* pour assigner une broche en sortie.

A la place de votre navigateur Web, vous pouvez également utiliser les commandes curl depuis une ligne de commande.

Vous devez inclure les bibliothèques Bridge, YunServer et YunClient :

```
#include <Bridge.h>
#include <YunServer.h>
#include <YunClient.h>
```

(1) Instancier un YunServer (donc un "serveur", dit "server" en anglais) permet au Yun d'écouter les clients connectés.

```
YunServer server;
```

(2) Dans la fonction **setup()**, démarrez la communication série pour faire du débogage et activer la LED branchée sur la broche 13 lors du démarrage du *bridge* ("*Bridge begins*" en anglais). **Bridge.begin()** est bloquant et devrait prendre environ 2 secondes pour s'achever. Une fois que le "Bridge" est démarré, la LED est éteinte.

```
void setup() {
  Serial.begin(9600);
  pinMode(13,OUTPUT);
  digitalWrite(13, LOW);
  Bridge.begin();
  digitalWrite(13, HIGH);
```

(3) Dans la seconde partie de **setup()**, **informez l'instance du YunServer pour qu'il écoute les connexions entrantes depuis localhost** (uniquement). Les connexions réalisées avec Linino seront passées au processeur ATmega32U4 pour les opérations de 'parsing' et de contrôle des broches. Cela se passe sur le port 5555. Démarrer le serveur avec **server.begin()**.

```
  server.listenOnLocalhost();
  server.begin();
}
```

(4) Dans la fonction **loop()**, vous allez créer une **instance de YunClient pour gérer la connexion**. Si un client se connecte, la requête est traitée ("*processed*" en anglais) par une fonction dédiée nommée **process**. On clôture ensuite la connexion.

Placez une instruction **delay** (attente) à la fin de **loop()** pour éviter que le processeur ne travaille trop.

```
void loop() {
  YunClient client = server.accept();

  if (client) {
    process(client);
    client.stop();
  }

  delay(50);
}
```

(5) Créez ensuite une fonction nommée **process** ("*traitement*") qui accepte un YunClient comme argument, **lit la commande et stocke l'information entrante dans une chaine de caractère**, Parse/décompose la commande REST pour extraire sa fonction (digital, analog ou mode) et passe l'information à la routine de traitement adéquate (et nommée de façon intelligible).

```
void process(YunClient client) {
  // Extraction de la commande.
  // readStringUntil() signifie Lire_chaine_jusqu'a()
  // Donc extraire ce qui se trouve dans l'url soit "http://myArduinoYun.local/arduino/"
  // et avant le "/" suivant!
  String command = client.readStringUntil('/');

  if (command == "digital") {
    digitalCommand(client);
  }
  if (command == "analog") {
    analogCommand(client);
  }
  if (command == "mode") {
    modeCommand(client);
  }
}
```

(6.1) Créez une fonction qui gère les commandes « *digital* ».

Exemple : http://myArduinoYun.local/arduino/**digital**/13/1.

La fonction accepte un *client* comme argument. Créez quelques variables locales pour commander la broche ("*pin*") et la valeur ("*value*") de la commande.

```
void digitalCommand(YunClient client) {
  int pin, value;
```

Faites un parsing (découpage) de la requête client pour obtenir la broche ("*pin*") à utiliser à l'aide de la fonction **client.parseInt()**.

Si le caractère après la broche est un "/", cela signifie que l'URL devrait aussi contenir une valeur 1 ou 0 (après le "/"). Cette valeur initialisera la "valeur de la broche" en plaçant sont état à HIGH ou LOW. S'il n'y a pas de "/" après le numéro de broche alors nous lisons la valeur de la broche spécifiée.

```
  // Extraction du numéro de broche (un nombre entier)
  pin = client.parseInt();
  // S'il y a un "/" après l'entier du numéro de
  // broche comme dans ".../arduino/digital/13/1"
  if (client.read() == '/') {
    // Alors lire un autre entier après le "/"
    // cette valeur est l'état de la broche
    // 1 = HIGH, 0 = LOW
    value = client.parseInt();
    digitalWrite(pin, value);
  } else {
    // Sinon, sans "/" après ne numéro de broche
    // comme dans ".../arduino/digital/13", il
    // faut simplement lire l'état de l'entrée.
    value = digitalRead(pin);
  }
```

Afficher/envoyer la valeur de/vers le client et faire une mise-à-jour de la clé dans le datastore avec la valeur de la broche.

En incluant la valeur renvoyée au client dans la fonction **F()**, vous pouvez afficher/renvoyer du texte provenant de la mémoire Flash. Cela permet d'économiser de la mémoire SRAM, très précieuse pour traiter de longues chaines de caractères telles que des URLs.

La clé (*key* en anglais) du datastore correspond au numéro de broche et au type (analogique ou digital). Par exemple, *D2* est la clé pour sauver la valeur de la broche digitale #2. La valeur correspondant à la clé est soit la valeur actuelle de la broche, soit la valeur que l'on assigne volontairement à la broche.

```
  client.print(F("Pin D"));
  client.print(pin);
  client.print(F(" set to "));
  client.println(value);
  // Clé pour le DataStore
  String key = "D";
  key += pin;
  // Stocker la ''clé = valeur'' dans le datastore du Bridge.
  Bridge.put(key, String(value));
}
```

(6.2) Mettez en place une fonction qui gère les appels analogiques. Cette fonction est similaire à celle qui gère les appels digitaux, excepté que la clé utilise un A à la place d'un D pour stocker la valeur de la broche d'entrée dans le DataStore :

```
void analogCommand(YunClient client) {
  int pin, value;
  // Obtenir le numéro de la broche depuis l'URL
  pin = client.parseInt();

  // S'il y a un "/" après...
  if (client.read() == '/') {
    // cela signifie que l'on fixe la valeur de la broche
    value = client.parseInt();
    analogWrite(pin, value);

    // Renvoyer une réponse au client
    client.print(F("Pin D"));
    client.print(pin);
    client.print(F(" set to analog "));
    client.println(value);

    // Valeur de la clé pour le DataStore
    String key = "D";
    key += pin;
    // Stockage de la "Clé = Valeur" dans le DataStore
    Bridge.put(key, String(value));
  }
  else {
    // Si pas de "/" après le numéro de broche, c'est que
    // l'on désire lire la valeur de la broche
    value = analogRead(pin);

    // renvoyer une réponse au client
    client.print(F("Pin A"));
    client.print(pin);
    client.print(F(" reads analog "));
    client.println(value);

    // Stockage de la "Clé = Valeur" dans le DataStore
    String key = "A";
    key += pin;
    Bridge.put(key, String(value));
  }
}
```

(6.3) Créez une ou plusieurs fonctions pour gérer les changements de mode des broches. Accepter un YunClient comme argument, et créez une variable locale pour maintenir le numéro des broches. Lire la valeur de la broche comme nous l'avons fait dans les fonctions de traitement des commandes "digital" et "analog".

```
void modeCommand(YunClient client) {
  int pin;
  // Obtenir le numéro de la broche depuis l'URL
  pin = client.parseInt();
```

Vérifiez que l'URL est bien valide. Il faut un paramètre en plus, et donc un "/" après le numéro de broche

```
  if (client.read() != '/') {
    client.println(F("error"));
    return;
  }
```

Si l'URL est correcte alors extraire le mode (jusqu'à la fin de l'URL qui se termine par un retour clavier... donc \r). Si le mode est *input* (entrée) ou *output* (sortie) alors configurer la broche comme demandé et informer le client. Si la chaine de caractère "mode" ne contient pas l'une de ces valeurs alors il faut informer le client qu'il y a une erreur.

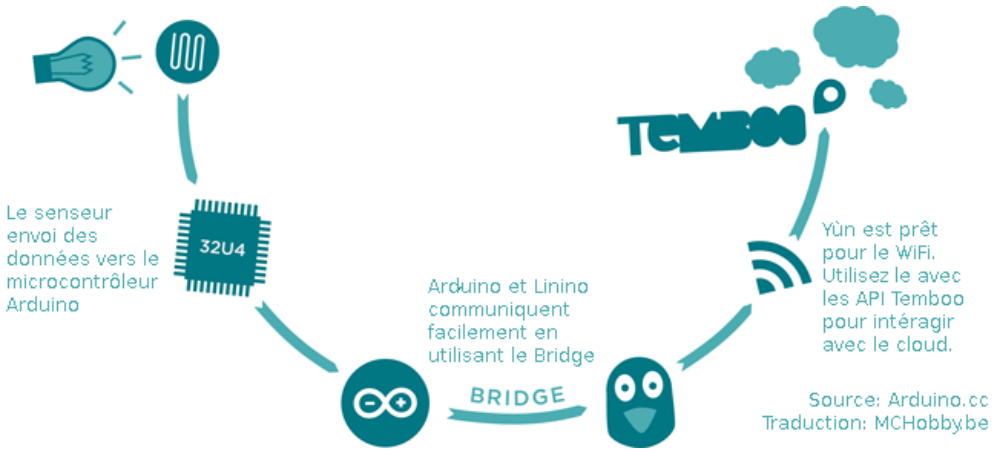
```
  String mode = client.readStringUntil('\r');
  if (mode == "input") {
    // configurer la broche en entrée
    pinMode(pin, INPUT);
    // Informer le client
    client.print(F("Pin D"));
    client.print(pin);
    client.print(F(" configured as INPUT!"));
    return;
  }

  if (mode == "output") {
    // configurer la broche en sortie
    pinMode(pin, OUTPUT);
    // informer le client
    client.print(F("Pin D"));
    client.print(pin);
    client.print(F(" configured as OUTPUT!"));
    return;
  }

  // Si cette portion de code est exécutée alors
  // la valeur de mode n'est ni "input" ni "output".
  // Si c'est le cas, alors il y a une erreur
  client.print(F("error: invalid mode "));
  client.print(mode);
}
```

Vous trouverez plus de détails sur cet exemple dans les pages d'exemples dédiés aux "exemples Bridge".

Se connecter à l'internet des objets avec Temboo



Crédit: Arduino arduino.cc traduction par MCHobby.be

Yún dispose de nombreuses fonctions réseaux, mais certains des aspects les plus excitants sont certainement ceux permettant d'interagir avec des plateformes en-ligne (online). Arduino a développé un partenariat avec Temboo pour rendre la connexion à vos services favoris aussi simple que possible. Temboo fournit un accès normalisé à plus de 100 APIs depuis un simple point de contact vous permettant ainsi de mixer et coupler des données provenant de plusieurs plateformes (par exemple Twitter, Facebook, Foursquare mais aussi FedEx ou PayPal).

Il y a de nombreux exemples dans le répertoire *Fichier > Exemples > Bridge > Temboo*. C'est un bon point de départ pour commencer à utiliser un Yùn dans le cloud. Vous pouvez en apprendre plus sur l'intégration de Temboo et Arduino dans les pages "Temboo getting started" (*Temboo.com, anglais*).

Spacebrew

Spacebrew est "une façon simple de connecter ensemble des objets interactifs". Spacebrew repose sur le modèle client/serveur et utilise des WebSockets pour établir la communication entre les deux éléments. Il fait fonctionner un web serveur customisé en python sur la carte Yún pour faciliter la communication.

Il y a plusieurs exemples concernant **Spacebrew** sur la carte Yún inclus dans le logiciel. Pour plus d'information sur Spacebrew, voyez les pages de documentation du projet.

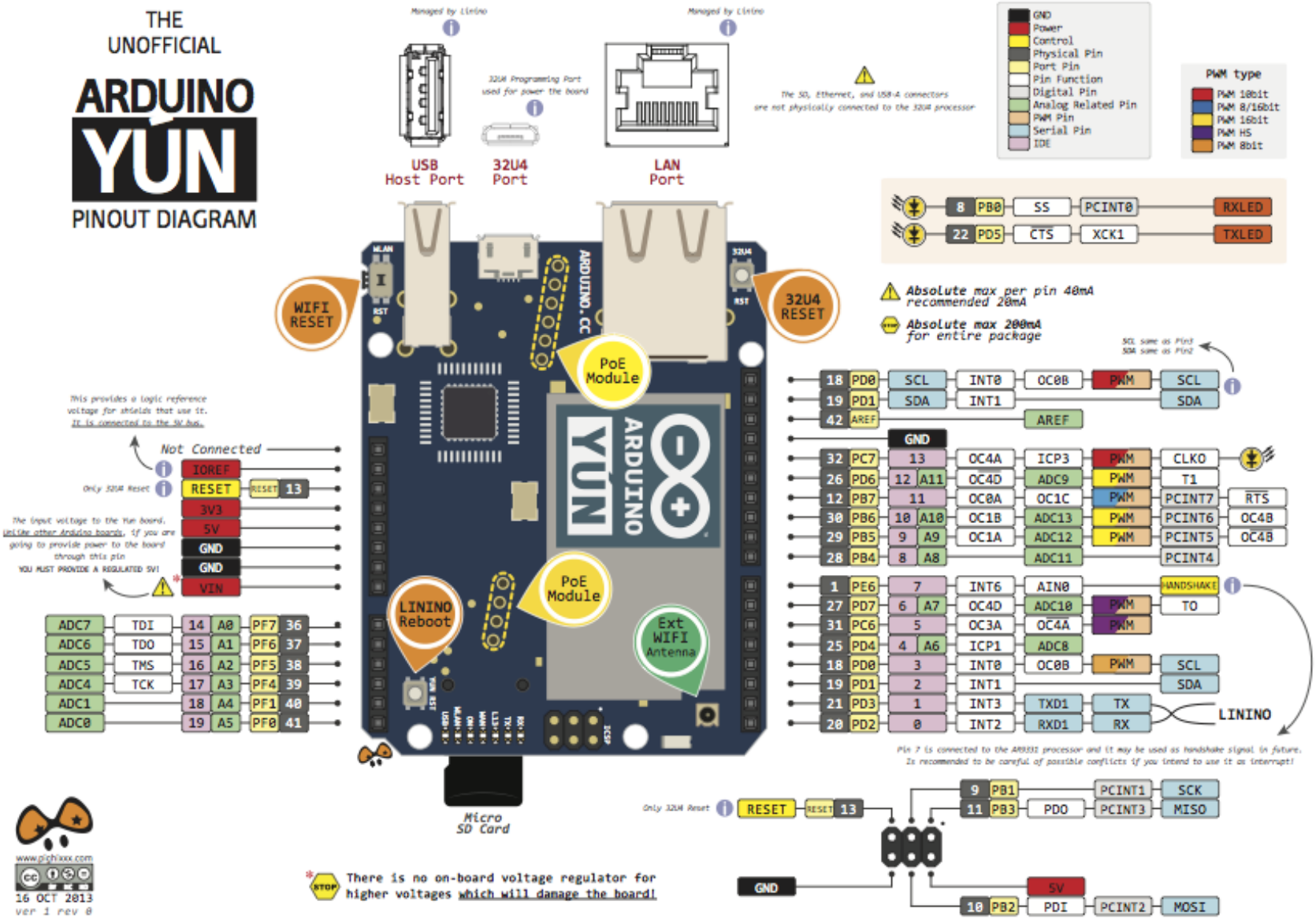
Installer des logiciels sur Linux

Yún dispose de plusieurs logiciels pré-installés sur Linino, incluant "curl" et "python". Vous pourriez avoir besoin d'installer d'autres logiciel sur Yún. Sur un système **Linux**, d'autres logiciels additionnels peuvent être installés à l'aide d'un outil appelé "**système de gestion de paquet**" (*package management system*). Référez-vous au Gestionnaire de paquet Yun pour plus d'information.

Guide en vidéo

Une vidéo YouTube "Getting started with Yun" réalisée par Arduino.

Annexe 1 : PINOUT DIAGRAM



Annexe 2 : les projets du répertoire ./Yun/Projets/ Proj_Yun_Pem

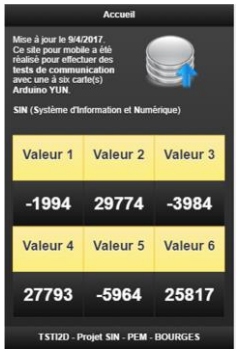
Répertoire	Sous répertoire IDE Arduino	Sous répertoire Web	Commentaires
a_Guide_D_Test_Console*	testComYunMega		Commande de la led L13 de la carte avec la console. Entrer H ou L
b_Guide_D_Process*	testYunProcess		? Linino se connecte sur un serveur en utilisant curl et télécharge du texte ASCII dans la console.
c_Guide_D_Bridge_Serveur_HTTP*	testYunServer		Utilisation de la bibliothèque « Bridge » pour accéder aux broches analogiques et numériques de la carte (REST). Exemple : http://192.168.1.94/arduino/digital/13/0 =>éteint L13
d_IoT_CH1_YunWheatherSation	YunWheatherSation		Matériels: Carte Yun + shield Tinkerkit (DHT11 sur I0), SEN11302P sur I1. Affichage dans la console.
e_IoT_CH4_Robot			A voir
f_IoT_Ch4_Lecture_Pot	yun_read_Pot	Site_Yun_Web_Pot	Matériels : Carte Yun + shield Tinkerkit + 2 potentiomètres. Connecter Pot1 à I0 et Pot2 à I1 Vérifier que l'url entrée dans les fichiers *get_pot1.php* et *get_Pot2.php* correspond à celle de la carte. Description : Affichage données page Web Yun.docx
g_Yun_Web_Led	Web_Led	Site_Yun_Web_Led	Matériels: Carte Yun Vérifier que l'url entrée dans le fichier *update_state.php* correspond à celle de la carte. Commande de la Led L13 avec les deux boutons à l'écran.
h_Yun_PO_PEM_SERRE	Yunpopem	Site_Yun_PO_PEM	Matériels : Carte Yun + shield Tinkerkit + DHT11 Affichage de l'humidité et de la température ambiante dans une page web Commande de la led L13 de la carte.
i_ Yun_PO_PEM_SERRE _V2	Yunpopem2	Site_Yun_PO_PEM_V2	Matériels : Carte Yun + shield Tinkerkit + DHT11 + Relay-Shield v1.3 Affichage de de l'humidité et de la température ambiante dans une page web (site web pour mobile). Commande d'un relais pour éclairer la serre.
J_Projet_Test_YUN	Yuntestino	SiteTestYun	Matériels : Carte(s) Yun Affichage de 6 grandeurs pouvant être issues de 1 à 6 cartes dans un tableau (site web mobile).
k_Projet_Yun_REST_JQxGauge	Yuntestino	RESTJauge	Matériel : Carte Yun Affichage d'une grandeur dans une jauge jQWidget.

* Description dans le guide de démarrage
? Pb !

Annexe 3 : Réseau privé sans fil de la section SIN

Le réseau wifi ci-dessous permet de s'affranchir des câbles Ethernet. Il est utilisé pour programmer les cartes **Arduino YUN** (logiciel Arduino ou MSVS + Visual Micro) et lors de leur utilisation en tant que client ou serveur. Le routeur **Dlink DIR-600** (ou autre) est le serveur DHCP. Il est configuré pour distribuer des adresses IP aux PC (100 à 254) et des adresses réservées aux cartes YUN (3 à 99 mais 24 au maximum). Les cartes Arduino YUN ne sont donc pas configurées avec des adresses IP statiques !

Schéma



SiteTestYun : @IP : 192.168.200.2 :10000

NasTest
Synology DS112 et dongle wifi (N300 –F7D2102)
user : **eleve** password : **projet**

Wifi (IP statique) indépendant du réseau lycée
➔ 192.168.200.2/24

Ethernet (IP statique) sur le réseau lycée
➔ 192.168.231.188/21

DlinkSinTest

Routeur Dlink DIR600 (DHCP + wifi)
Firmware mis à jour : v2.18 ou +
user : **admin** password : **test**

SSID : DlinkSinTest
Mdp : 12345678 (WPA)

IP Statique : 192.168.200.1/24

YUNSINx

Arduino YUN
DHCP (IP réservée et distribuée par le routeur Dlink DIR600)
IP : 192.168.200.y/24 y ∈ [3,99]

Procédure

1. Configurer le routeur Dlink (réserver les adresses DHCP pour les YUN).
2. Configurer le wifi du NAS Synology. Lui faire rejoindre le réseau du routeur Dlink. (IP fixe)
3. Nommer les YUN. **Leur faire rejoindre le réseau du routeur**.

Détails pages suivantes.

PC et carte wifi (PCI-E TL-WN881ND)

Wifi (DHCP)
Ethernet (DHCP) sur le réseau lycée

1. Configuration du routeur DLINK DIR600

Généralités

ROUTER SETTINGS

Use this section to configure the internal network settings of your router. The IP address that is configured here is the IP address that you use to access the Web-based management interface. If you change the IP address here, you may need to adjust your PC's network settings to access the network again.

Router IP Address :	192.168.200.1
Default Subnet Mask :	255.255.255.0
Host Name :	DlinkSinTest
Local Domain Name :	(optional)
Enable DNS Relay :	

DHCP SERVER SETTINGS

Use this section to configure the built-in DHCP server to assign IP address to the computers on your network.

Enable DHCP Server :	
DHCP IP Address Range :	100 to 199 (addresses within the LAN subnet)
DHCP Lease Time :	10080 (minutes)

Exemples pour la réservation d'adresses

24 - DHCP RESERVATION

Remaining number of rules that can be created: 22

	Computer Name	IP Address	MAC Address	
☑	YUNSINR1	192.168.200.3	b4:21:8a:f0:18:2e	<< Computer Name
☑	YUNSINR2	192.168.200.4	b4:21:8a:f0:00:58	<< Computer Name
☐				<< Computer Name

Configuration du réseau sans fil

WIRELESS NETWORK SETTINGS

Enable Wireless :	Always, New Schedule
Wireless Network Name :	DlinkSinTest (Also called the SSID)
Enable Auto Channel Selection :	
Wireless Channel :	11
Transmission Rate :	Best (automatic) (Mbit/s)
Wireless Mode :	802.11 Mixed(n/g/b)
Band Width :	20 MHz
Enable Hidden Wireless :	(Also called the SSID Broadcast)

WIRELESS SECURITY MODE

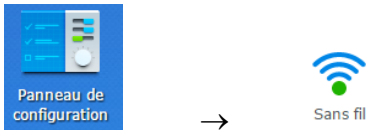
Security Mode : Enable WPA/WPA2 Wireless Security (enhanced)

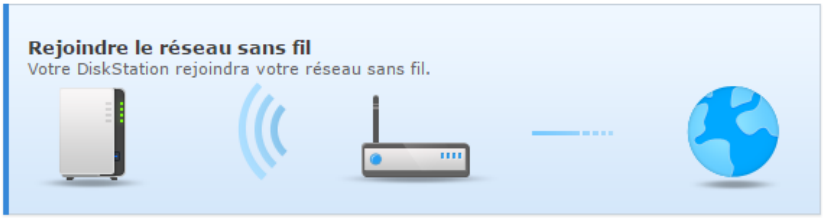
WPA/WPA2

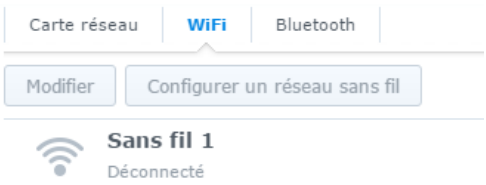
WPA/WPA2 requires stations to use high grade encryption and authentication.

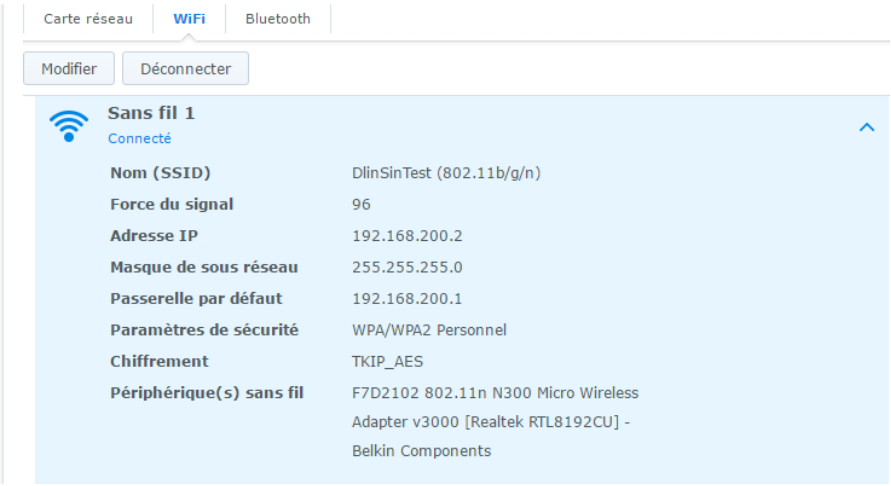
Cipher Type :	AES
PSK / EAP :	PSK
Network Key :	12345678
	(8~63 ASCII or 64 HEX)

2. Configuration du NAS DS112 Synology





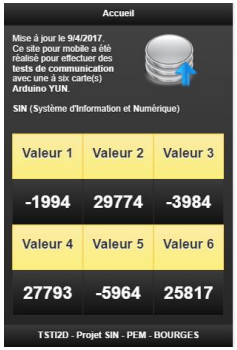




3. Configuration des cartes YUN

Suivre la procédure proposée sur la page de garde (ou § **Configuration du wifi de la carte** dans ce dossier) pour donner un nom à la carte et la connecter au réseau DlinkSinTest.
Placer « REST API Acces » sur OPEN. (Ne pas configurer la carte avec une IP statique, l'IP réservée et attribuée par la Box)

Suite de l’annexe 3 : Sources du code du site de test (index.php)




```
<!DOCTYPE html>
<html>
<!--------------------------------------------------------------------------------
Lycée : Pierre Emile Martin 1 av de Gionne 18000 Bourges
Classe : Terminale STI2D option SIN
Projet : Site de test pour la carte YUN        Version : 1
Année scolaire : 2015 - 2016
Révision le : 9/4/2017
Type de site : pour mobile
--------------------------------------------------------------------------------
Fichier : index.php
--------------------------------------------------------------------------------
Description : Page d'accueil du site mobile.
-------------------------------------------------------------------------------->
<head>
	<title>Test YUN</title>
	<meta name="viewport" content="width=device-width, initial-scale=1" />
	<link rel="stylesheet" href="JQuery/jquery.mobile-1.2.0/jquery.mobile-1.2.0.min.css" />
</head>

<body>

<div data-role="page" data-theme="a">
	<div data-role="header">
		<h1>Accueil</h1>
	</div><!-- /header -->

	<div data-role="content">
		<div class="ui-grid-a">
			<div class="ui-block-a" style="text-align:left">
				<div>Mise à jour le <strong>9/4/2017</strong>. <br /> Ce site pour mobile a été réalisé pour effectuer des <strong>tests de communication</strong> avec une à six carte(s)
                                                                                                                                                                <strong>Arduino YUN</strong>. </div>
			</div>
			<div class="ui-block-b" style="text-align:center;">
				<div id="iconeMeteo"><?php echo "<img src=\"img/test.png\"/>"; ?></div>
	</div>
		</div><!--class="ui-grid-a"-->
		<br />
		<div><strong>SIN </strong>(<strong>S</strong>ystème d'<strong>I</strong>nformation et <strong>N</strong>umérique)</div>
		<br /> <br />
		<div class="ui-grid-b" style="text-align:center;">
			<div class="ui-block-a">
				<div class="ui-bar-e" style="height:80px;"><h2>Valeur 1</h2></div>
			</div>

			<div class="ui-block-b">
				<div class="ui-bar-e" style="height:80px;"><h2>Valeur 2</h2></div>
	</div>
```

```
	<div class="ui-block-c">
				<div class="ui-bar-e" style="height:80px;"><h2>Valeur 3</h2></div>
	</div>

	<div class="ui-block-a">
				<div class="ui-bar-a" style="height:80px;"><h1 id="val1"></h1></div>
	</div>

	<div class="ui-block-b">
				<div class="ui-bar-a" style="height:80px;"><h1 id="val2"></h1></div>
	</div>

				 <div class="ui-block-c">
				<div class="ui-bar-a" style="height:80px;"><h1 id="val3"></h1></div>
	</div>
				<div class="ui-block-a">
				<div class="ui-bar-e" style="height:80px;"><h2>Valeur 4</h2></div>
				</div>

				<div class="ui-block-b">
				<div class="ui-bar-e" style="height:80px;"><h2>Valeur 5</h2></div>
	</div>

				<div class="ui-block-c">
				<div class="ui-bar-e" style="height:80px;"><h2>Valeur 6</h2></div>
	</div>

	<div class="ui-block-a">
				<div class="ui-bar-a" style="height:80px;"><h1 id="val4"></h1></div>
	</div>

	<div class="ui-block-b">
				<div class="ui-bar-a" style="height:80px;"><h1 id="val5"></h1></div>
	</div>

				 <div class="ui-block-c">
				<div class="ui-bar-a" style="height:80px;"><h1 id="val6"></h1></div>
	</div>
    </div> <!--class="ui-grid-a"-->
        </div><!-- /content -->

        <div data-role="footer" data-theme="a">
            <h4>TSTI2D - Projet SIN - PEM - BOURGES</h4>
        </div><!-- /footer -->

</div><!-- /page -->
        <script src="JQuery/jquery-1.8.2.min.js"></script>
        <script src="JQuery/jquery.mobile-1.2.0/jquery.mobile-1.2.0.min.js"></script>
        <script src="scripts/script.js"></script>
</html>
```

Suite de l’annexe 3 : Sources du code du site de test (get_val1.php)

```
<?php
 // Create cURL call, make sure to change it with your Yun name
 $service_url = "http://192.168.200.3/arduino/grandeur/val1";
 $curl = curl_init($service_url);

 // Send cURL to Yun board
 curl_setopt($curl, CURLOPT_IPRESOLVE, CURL_IPRESOLVE_V4);
 curl_exec($curl);
 curl_close($curl);
?>
```

Annexe 3 suite : Sources du code du site de test (scripts.js)

```
$(document).ready(function(){
                                $("#val1").load('scripts/get_val1.php');
                                $("#val2").load('scripts/get_val2.php');
                                $("#val3").load('scripts/get_val3.php');
                                $("#val4").load('scripts/get_val4.php');
                                $("#val5").load('scripts/get_val5.php');
                                $("#val6").load('scripts/get_val6.php');
//-----------------------------------------------------------------
        setInterval(function() {
                                $("#val1").load('scripts/get_val1.php');
                                $("#val2").load('scripts/get_val2.php');
                                $("#val3").load('scripts/get_val3.php');
                                $("#val4").load('scripts/get_val4.php');
                                $("#val5").load('scripts/get_val5.php');
                                $("#val6").load('scripts/get_val6.php');
                        }, 5000);
//-----------------------------------------------------------------
});
```

Suite de l'annexe 3 : Sources du code de la carte yun (Yuntestino.ino)

```
// ----------------------------------------------------
// Test de la carte YUN
// Matériel : Carte Yun
// Projet : Test
// Version: 1
// Date: 6/12/2015
// Fichier : Yuntestino.ino
// Commentaires :Envoi de 2 valeurs
// ----------------------------------------------------
// Bibliothèques
#include <Bridge.h>
#include <YunServer.h>
#include <YunClient.h>
// ----------------------------------------------------
// Create Yun server
YunServer server;
// ----------------------------------------------------
// Variable globale
int val = 0;
// ----------------------------------------------------
void setup(void)
{
  // Start bridge
  Bridge.begin();

  // Start server
  server.listenOnLocalhost();
  server.begin();
}
//-----------------------------------------------------
// ----------------------------------------------------
void loop(void)
{
  // Get clients coming from server
  YunClient client = server.accept();

  // There is a new client ?
  if (client) {
    // Process request
    process(client);

    // Close connection and free resources.
    client.stop();
  }

  // Poll every 50ms
  delay(50);
}
// ----------------------------------------------------
// Process incoming command
void process(YunClient client) {
  // read the command
  String command = client.readStringUntil('/');

  // is "grandeur" command?
  if (command == "grandeur") {
    GCommand(client);
  }
}
// ----------------------------------------------------
int CalculVal(int n){
  return n*val++;
}
// ----------------------------------------------------
void GCommand(YunClient client) {
  // Read command and react accordingly
  String command = client.readStringUntil('\r');
```

```
  // Return temp measurement
  if (command == "val1"){
    client.print(CalculVal(2));
  }

  if (command == "val2"){
    client.print(CalculVal(3));
  }
    if (command == "val3"){
    client.print(CalculVal(4));
  }

  if (command == "val4"){
    client.print(CalculVal(5));
  }
    if (command == "val5"){
     client.print(CalculVal(6));
  }

  if (command == "val6"){
     client.print(CalculVal(7));
  }

}
// ---------------------------------------------------
```

Annexe 4 : Opérations à réaliser pour mettre en œuvre un serveur Web avec la carte Arduino YUN

(0) Bibliothèques

```
#include <Bridge.h>
#include <YunServer.h>
#include <YunClient.h>
```

(1) Créer un objet server

```
YunServer server;
```

(2) Démarrer le bridge entre l'arduino et le µC Linux

```
Bridge.begin();
```

(3) Démarrer le serveur

```
server.listenOnLocalhost();
server.begin();
```

(4) Créer un objet client dans la boucle principale et attendre une requête

```
// Get clients coming from server
YunClient client = server.accept();

// There is a new client?
if (client) {
  // Process request
  process(client);

  // Close connection and free resources.
  client.stop();
}
```

(5) Créer le process permettant de détecter le type de requête

```
void process(YunClient client) {
 // read the command
 String command = client.readStringUntil('/');

 // is "digital" command?
 if (command == "digital") {
   digitalCommand(client);
 }

 // is "analog" command?
 if (command == "DHT") {
   DHTCommand(client);
 }
Etc..
}
```

(6) Créer une fonction pour chaque requête

```
void digitalCommand(YunClient client) {
 int pin, value;

 // Read pin number
 pin = client.parseInt();

 // If the next character is a '/' it means we have an URL
 // with a value like: "/digital/13/1"
 if (client.read() == '/') {
   value = client.parseInt();
   digitalWrite(pin, value);
 }
 else {
   value = digitalRead(pin);
 }

 // Send feedback to client
 client.print(F("Pin D"));
 client.print(pin);
 client.print(F(" set to "));
 client.println(value);
}
Etc
```

Remarque : Une lecture dans la chaîne de caractère incrémente un pointeur. Celui-ci est passé dans l'objet client.

Annexe 5 : OPENWRT

Se connecter en SSH avec Putty : root@<nom>.local ou root@<IP>

Gestionnaire de paquets
opkg update
opkg install
opkg remove
opkg list-installed

Gestion des répertoires
ls : liste (-R : sous répertoires)
mkdir : création d'un répertoire
cd : changer de répertoire
chmod : pour changer les permissions d'un fichier
cp : pour copier un fichier

Annexe 6 : Upgrade your Arduino Yun with sysupgrade

```
$ cd /tmp

$ wget http://download.linino.org/linino_distro/master/latest/openwrt-ar71xx-generic-linino-yun-16M-250k-squashfs-sysupgrade.bin (pas de place pb!)

$ sysupgrade -v openwrt-ar71xx-generic-linino-yun-16M-250k-squashfs-sysupgrade.bin
```

sysafter about 1 minute the board will reboot and the new image will be loaded on the board.

- if you want to overwrite all the configuration settings present on the previous image, you have to add the **-n** option to the sysupgrade command

```
$ sysupgrade -v -n openwrt-ar71xx-generic-linino-yun-16M-250k-squashfs-sysupgrade.bin
```