|  |
| --- |
| AceTeaM |
| Projet Arduino |
| Applications Arduino |
|  |
| **Thomas AUGUEY – Vincent AUGUEY – Michael Melkior** |
| **31/05/2013** |

|  |
| --- |
| Développement logiciel du pilotage d’une plateforme domotique basé sur l’architecture Arduino. |

# Projet Arduino

Contenu

[Projet Arduino (Programme principale) 1](#_Toc357780828)

[Cahier des charges 2](#_Toc357780829)

[Environement matériel 2](#_Toc357780830)

[Equipement 2](#_Toc357780831)

[Licence Arduino 3](#_Toc357780832)

[Matériel libre et logiciel libre (source : Wikipedia) 3](#_Toc357780833)

[Usage commercial 3](#_Toc357780834)

[Analyse 4](#_Toc357780835)

[Environement 4](#_Toc357780836)

[Programmation 4](#_Toc357780837)

[Librairies de modules 4](#_Toc357780838)

[Carte de développement 4](#_Toc357780839)

[Développement 5](#_Toc357780840)

[Fichiers 5](#_Toc357780841)

[Architecture 5](#_Toc357780842)

[Configuration des équipements 5](#_Toc357780843)

## Cahier des charges

### Objectif

L’Application embarquée Arduino doit nous permettre de :

* Communiquer avec des applications de contrôles tierces et d’exécuter leurs ordres
* Recevoir et envoyer des informations avec les équipements et d’agir en conséquence
* Contenir la liste des équipements connectés et leurs types
* Contrôler l’identification des applications tierces pour assuré la sécurité

Ce programme est le centre nerveux de l’installation. Il se trouve directement dans la carte programmable et possède le contrôle total sur l’équipement. Il doit permettre une réaction rapide et une reconnaissance des équipements connectés.

### ****Environnement matériel****

**Pour permettre à l’application de prendre des décisions sur son environnement, le programme doit posséder une base d’informations permettant d’identifier tous les équipements de la maison.**

### ****Equipement****

**Un équipement est identifié selon les paramètres suivants :**

* **Un ou plusieurs connecteurs réservés sur la carte (rôle et numéros)**
* **Un intitulé**
* **Une zone géographique (numéro de la pièce)**
* **Un identifiant unique**

### Communication tierces

## Licence Arduino

### Matériel libre et logiciel libre (source : Wikipedia)

Le design matériel de l'Arduino est distribué sous **licence** **Creative Commons Attribution Share-Alike** **2.5** et est disponible sur le site d'Arduino.

Les schémas ainsi que les typons de circuits sont également disponibles. Le code source de l'environnement de programmation et les bibliothèques embarquées sont disponibles sous **licence LGPL**.

* <http://creativecommons.org/licenses/by-sa/2.5/>
* <http://fr.wikipedia.org/wiki/GNU_LGPL>

### Usage commercial

Ces licences nous donne le droit de :

* Commercialiser nos produits basés sur le matériel **Arduino**
* Commercialiser nos produits basés sur les logiciels **Arduino**

Ces licences nous imposes de :

* Utiliser les mêmes licences pour nos produits (redistribution libre)
* Citer les auteurs originels

## Analyse

Développer un logiciel de gestion Domotique qui devra permettre via la carte contrôleur **Arduino** de piloter les différents appareils de la maison.

### Environnement

Pour programmer avec Arduino il faut posséder l’environnement de base disponible sur le site officiel **(**<http://arduino.cc/en/Main/Software>)

**Pour nos tests nous utiliserons VisualStudio et son plugin visualmicro (voir Etude/cahier des charges) ou l’IDE officiel Arduino (**<http://arduino.cc/>)

### Programmation

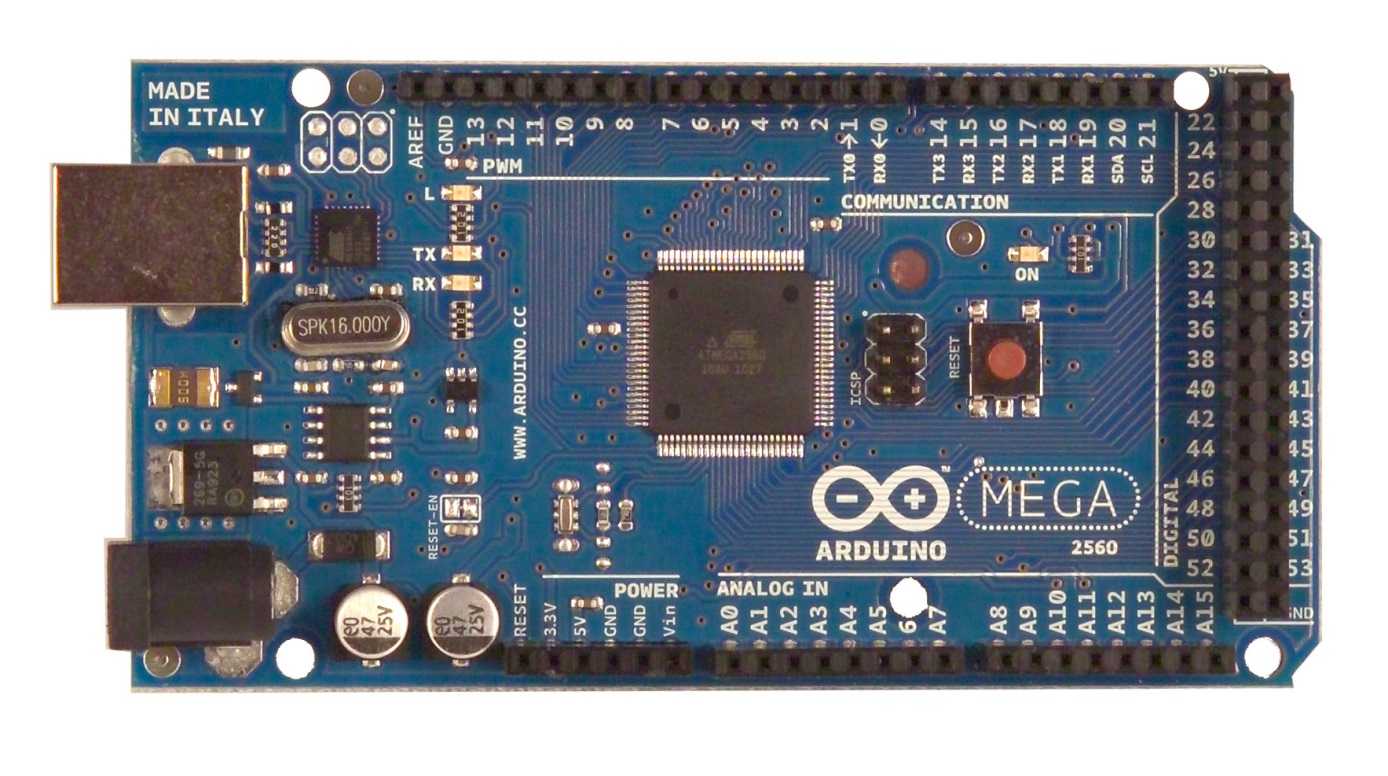
Les programmes transmissibles dans la carte **Arduino** sont programmés en **C/C++. Les programmes sont** compilés est transférés via l’IDE.

### Librairies de modules

Il existe de nombreux modules de librairies utilisables avec les cartes Arduino. Les librairies et documentations sont disponibles dans le dossier **arduino-*[version]*\reference\index.html de l’environnement de développement Arduino.**

### ****Cartes****

#### **Arduino Mega 2560**



**Tous les tests de développement sont réalisés sur une carte de type Arduino Mega 2560**

<http://arduino.cc/en/Main/Products>

## Applications

### Server

**Fonction**

Ce programme est le point d’entrée de la communication entre l’installation et l’utilisateur.

La communication est basée sur un système de message formaté (voir description plus bas).

Les messages ainsi reçus de l’extérieur son transmit aux équipements nommés via les entrées/sorties de la carte.

**Format de message**

Un message est un texte formaté et structuré qui peut être transmit de diverses façons (Ethernet, Internet, Port Série, …). Le format est basé sur un model simple d’association Nom/Valeur (ou tableau associatif).

#### Messages

|  |  |
| --- | --- |
| **Messages** | Liste des messages traités par l’application |
| **Identifiant** | **Désignation** |
| **LIST** | Liste les équipements |
| **REM** | Supprime un équipement |
| **ADD** | Ajoute un équipement |
| **GET** | Obtient les informations sur un équipement |
| **SWITCH** | Active / Désactive un équipement |
| **CHECK** | Obtient l’état d’un équipement |
| **CMD** | Transmet une commande à un équipement |

### Equipments

#### LedStrip

Contrôle une série de LED rouges avec plusieurs modes de défilement.

Cet équipement simple permet un déploiement rapide de test.

#### Objets

|  |  |  |
| --- | --- | --- |
| **Type** | **Identifiant** | **Désignation** |
| **LedStripObjet** | **strip** | Contrôleur de LED en série |

#### LedStripObjet

|  |  |
| --- | --- |
| **Etats** | **Désignation** |
| **-** | - |
| **Configurations** | **Désignation** |
| FirstPin | Numéro du premier PIN digital |
| PinCount | Nombre total de PIN utilisés |
| AnimationMode | Mode d’animation :  0 : Linéaire  1 : Décalé |
| **Commandes** | **Désignation** |
| **-** | - |

## Librairies

Chacune des applications ci-dessus utilise des fonctionnalités partagées qui sont écrites dans des librairies à intégrer à l’IDE Arduino avant compilation.

### NP (Noyau Portable)

Est une librairie C / C++ reprenant quelques concept de base de la programmation. Elle est indépendante de toutes autres librairies ou implémentations.

**Includes :**

**Xarg.h** : Formatage (sérialisation) de tableau associatif sous forme de chaine

#### Installer la librairie Noyau Portable (NP) dans l’IDE Aduino

L’Application utilise la librairie **NoyauPortable**.

Pour permettre l’inclusion depuis le projet procédez comme suit :

1. Télécharger le code source depuis le dépôt (<https://github.com/Ace4teaM/nplib>)
2. Créer un dossier nommé **nplib** dans les librairies Arduino (C:\Program Files (x86)\Arduino\libraries)
3. Copier le contenu la librairie NP de niveau 1 (nplib/src/l1)
4. Coller les sources dans le dossier **nplib** créé ci-dessus
5. Redémarrer l’IDE Arduino

Ou depuis un lien symbolique :

1. Télécharger le code source depuis le dépôt (<https://github.com/Ace4teaM/nplib>)
2. Créer un dossier nommé **nplib** dans les librairies Arduino (C:\Program Files (x86)\Arduino\libraries)
3. Ouvrir l’invite de commande DOS
   1. Ouvrir le dossier des librairies Arduino « **cd "C:\Program Files (x86)\Arduino\libraries"** »
   2. Créer le lien « **mklink /D nplib "C:\Users\developpement\Documents\GitHub\nplib\src\l1"** »

Redémarrer l’IDE Arduino

## Développement

### Libraries

Pour pouvoir compiler les programmes Arduino vous devez ajouter la librairie partagée « **Serveur** » à aux bibliothèques de l’IDE Arduino.

Pour cela, 2 solutions :

1. Copier le dossier ‘[Arduino-Project]/Arduino App/Serveur’ dans le répertoire des librairies Arduino (généralement «C:\Program Files (x86)\Arduino\libraries »)
2. Créer un lien symbolique vers le dossier ‘[Arduino-Project]/Arduino App/Serveur’ dans le répertoire des librairies Arduino (généralement «C:\Program Files (x86)\Arduino\libraries »)

Exemple de commande :

**cd "C:\Program Files (x86)\Arduino\libraries"**

**mklink /D Base "C:\Users\...\Arduino-Project\Arduino App\Libraries\Base"**

### Configuration des équipements

La configuration est stockée dans un fichier XML permettant ainsi une configuration facilement manipulable. Le programme se charge de le parser en mémoire pour être rapidement consultable.

(Pour stockable la configuration, la carte doit intégrer un module SD)

### Communication avec les applications tierces

**L’implémentation d’une classe d’interface au sein de l’API permet l’usage transparent de divers méthodes de transport de données (WIFI, Radio, Ethernet, ...)**

### Wifi Shield ( IMPORTANT )

**Note Importante : *Pour utiliser le module Wifi avec son Firmware d’origine. Le programme doit OBLIGATOIREMENT être compilé et transféré avec l’IDE Arduino 1.0.2***

***Si vous disposez de l’IDE Arduino 1.0.5+ merci de mettre à jour le Firmeware du Wifi Shield***

**Symptômes : Le Wifi Shield se connecte au réseau mais ne reçoit pas les requêtes TCP/IP**

**Articles connexes :**

* <http://stackoverflow.com/questions/17357650/i-cannot-connect-to-my-arduino-wifi-shield-server-but-i-can-ping-it>
* <http://www.dfrobot.com/community/how-to-upgrade-arduino-wifi-shield-firmware-on-windows.html>

[http://forum.arduino.cc/index.php?topic=167794.0](http://forum.arduino.cc/index.php?topic=167794.0Mémoire)

### [Mémoire](http://forum.arduino.cc/index.php?topic=167794.0Mémoire)

La faible quantité de mémoire vive (SRAM) embarqué sur les cartes Arduino et source de problème et de bug pour le programmeur.

En effet, le compilateur ne pourra pas vous prévenir d’un dépassement de l’utilisation de la mémoire vive (SRAM).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Models** |  | Arduino nano |  | Arduino mega |
| **Chips** | **ATMega168** | **ATMega328P** | **ATmega1280** | **ATmega2560** |
| **Flash  (1 Kbyte utilisé pour le bootloader)** | 16 KBytes | 32 KBytes | 128 KBytes | 256 KBytes |
| **SRAM** | 1024 bytes | 2048 bytes | 8 KBytes | 8 KBytes |
| **EEPROM** | 512 bytes | 1024 bytes | 4 KBytes | 4 KBytes |

#### Définition des chaines

…