

## TP M2M

# Mise en œuvre des ressources de la carte Arduino Uno

Pierre Gaucher

[pierre.gaucher@univ-tours.fr](mailto:pierre.gaucher@univ-tours.fr)

Ce document est en cours de rédaction et peut contenir des erreurs. Merci de bien vouloir les signaler afin d'en améliorer le contenu.

Reproduction et diffusion interdites sans autorisation préalable.



## Table des matières

1.	Présentation du matériel.....	4
1.1.	Carte microcontrôleur Arduino .....	4
1.2.	Description fonctionnelles des broches de la cartes Arduino.....	5
1.2.1.	Entrées sorties numériques.....	5
1.2.2.	Entrées sorties analogiques .....	6
1.3.	Shield Grove.....	6
1.4.	Périphériques d'E/S utilisés.....	7
1.4.1.	Module Led .....	8
1.4.2.	Module bouton .....	9
1.4.3.	Module potentiomètre .....	9
1.4.5.	Module détecteur de lumière_2.....	10
1.4.6.	Module capteur d'humidité et de température .....	10
1.4.7.	Module afficheur LCD .....	11
2.	Mise en œuvre.....	11
2.1.	Environnement de programmation.....	11
2.2.	Structure de base d'un programme Arduino .....	14
2.3.	Gestion des E/S analogiques .....	18
2.4.	Utilisation d'une bibliothèque : mesure température – humidité .....	18

# 1. Présentation du matériel

Le support matériel utilisé tout au long de ce projet sera composé d'une carte microcontrôleur Arduino, utilisée avec des shields particuliers, destinés à faciliter la mise en œuvre matérielle.

## 1.1. Carte microcontrôleur Arduino

La carte utilisée est la carte Arduino Uno Rev 3. Elle est architecturée autour d'un microcontrôleur Atmega328P de la société ATMEL. Les principaux éléments qui composent cette carte sont décrits ci-dessous (cf. Figure 1). Des informations complémentaires peuvent être consultées sur <http://arduino.cc/en/Main/ArduinoBoardUno>

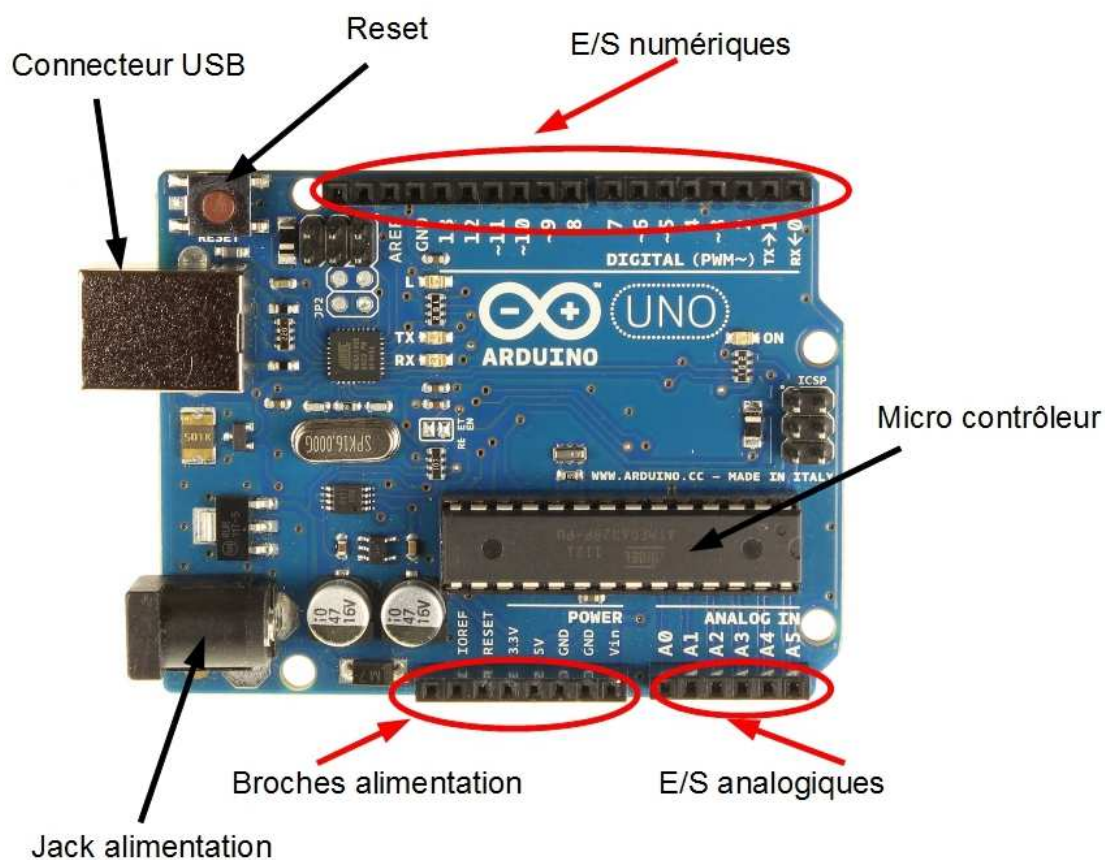


Figure 1 : Carte Arduino Uno Rev 3

- Microcontrôleur Atmega328P
- 14 broches d'E/S numériques
- 6 broches d'E/S analogiques
- Alimentation par port USB
- Bouton de reset permettant de redémarrer l'application chargée dans la mémoire de programme
- 32Ko de mémoire flash, dont 512 octets déjà occupés par le bootloader
- 2 Ko de mémoire SRAM
- EEPROM de 1Ko
- Fréquence horloge : 16 MHz
- Une Led déjà connectée sur la broche numérique 13

La documentation complète du microcontrôleur est disponible via le lien ci-après :  
Support TP M2M : Arduino

Afin de faire le lien entre la nomenclature des broches utilisées dans le contexte Arduino et celui utilisé par la société ATMEL, la figure ci-après présente le mapping utilisé selon le contexte (cf. Figure 2). Cette « carte de correspondance » pourra être utilisée afin de pouvoir accéder aux mêmes ressources matérielles, en fonction de l'environnement logiciel qui sera utilisé par la suite. Elle sera également utile afin de pouvoir se référer à la documentation du constructeur du microcontrôleur.

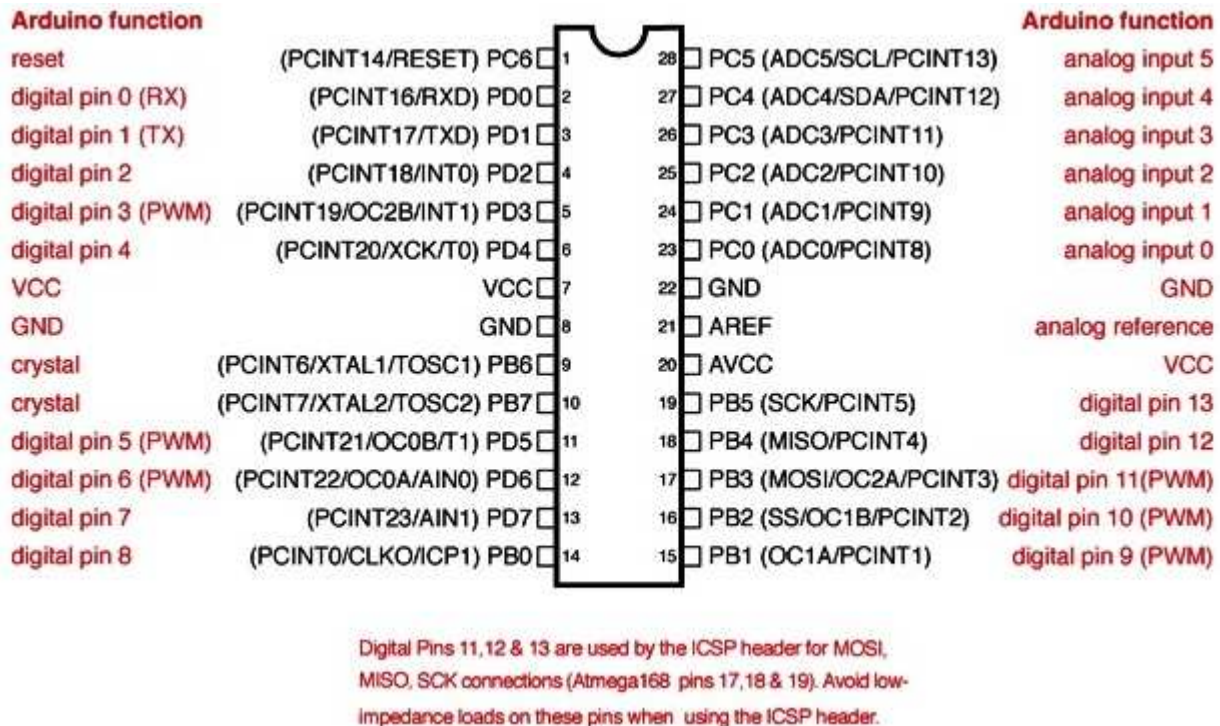


Figure 2 : Correspondance des broches du microcontrôleur Atmega328P et platine Arduino

## 1.2. Description fonctionnelles des broches de la cartes Arduino

### 1.2.1. Entrées sorties numériques

Elles correspondent aux broches de la carte Arduino D0 jusqu'à D13. Elles peuvent être utilisées aussi bien en entrée qu'en sortie. Il sera impératif de spécifier avant toute utilisation quel sera le « sens » d'utilisation, soit en initialisant directement les registres de contrôle correspondant, ou bien en utilisant des fonctions de plus haut niveau disponibles sous l'environnement de programmation Arduino (*pinMode*). Lorsque cette première étape est réalisée, alors il est possible de lire l'état d'une broche, ou bien de définir son état, aussi bien par le registre correspondant ou bien en utilisant les fonctions *digitalWrite()* ou *digitalRead()* sous l'environnement Arduino. Une broche numérique ne peut avoir que deux états logique, haut ou bas, correspondant à une tension respectivement de 5V ou 0V.

Sous l'environnement de développement Arduino, certaines broches numériques permettent de gérer des interruptions matérielles externes. Il s'agit des broches 2 et 3.

Sous l'environnement de développement Arduino, les broches D3, D5, D6, D9, D10 et D11 permettent également de générer des signaux PWM (Pulse Width Modulation ou Modulation par Largeur d'Impulsion). Cela permet de générer des tensions continues entre 0V et 5V, en utilisant la fonction *analogWrite()*, et de faire varier par exemple l'intensité lumineuse d'une Led.

Par défaut, sur la carte Arduino, une Led est déjà câblée sur la broche D13.

### 1.2.2. Entrées sorties analogiques

Elles correspondent aux broches A0 jusqu'à A5. Il peut être associé à chacune de ces 6 broches un convertisseur analogique / numérique sur 10 bits.

Les broches A4 et A5 jouent également un rôle particulier lors de l'utilisation du bus de communication I2C. Ce bus utilise 2 fils, SCL et SDA. SDA permet de gérer l'échange des données qui transitent sur le bus tandis que SCL permet de gérer l'horloge qui assure la synchronisation des échanges de données.

## 1.3. Shield Grove

Il s'agit d'une carte d'interface destinée à faciliter la mise en œuvre matérielle de périphériques d'E/S sur la carte Arduino (cf. Figure 3).

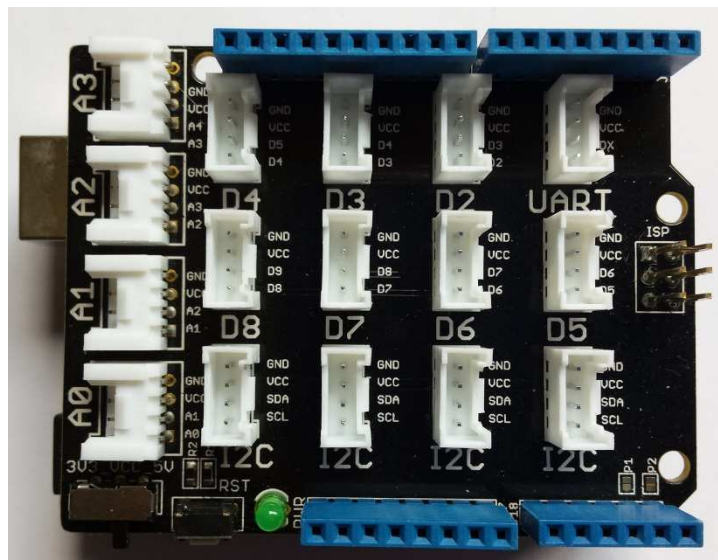


Figure 3 : Shield de connecteurs Grove

Les principales ressources du microcontrôleur (ports numériques et analogiques) sont réparties par famille de connecteurs (cf. Figure 4). Ce shield permet également d'accéder au reset du microcontrôleur de la carte Arduino.



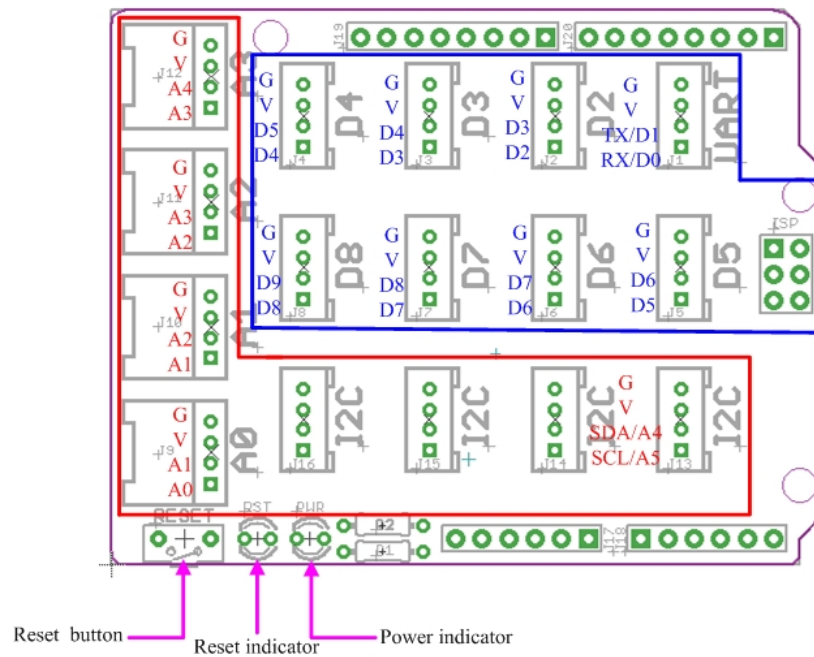


Figure 4 : Connectique shield Grove et ports E/S carte Arduino

Le shield sera connecté sur les borniers de la carte Arduino, comme le montre la figure ci-dessous (cf. Figure 5).

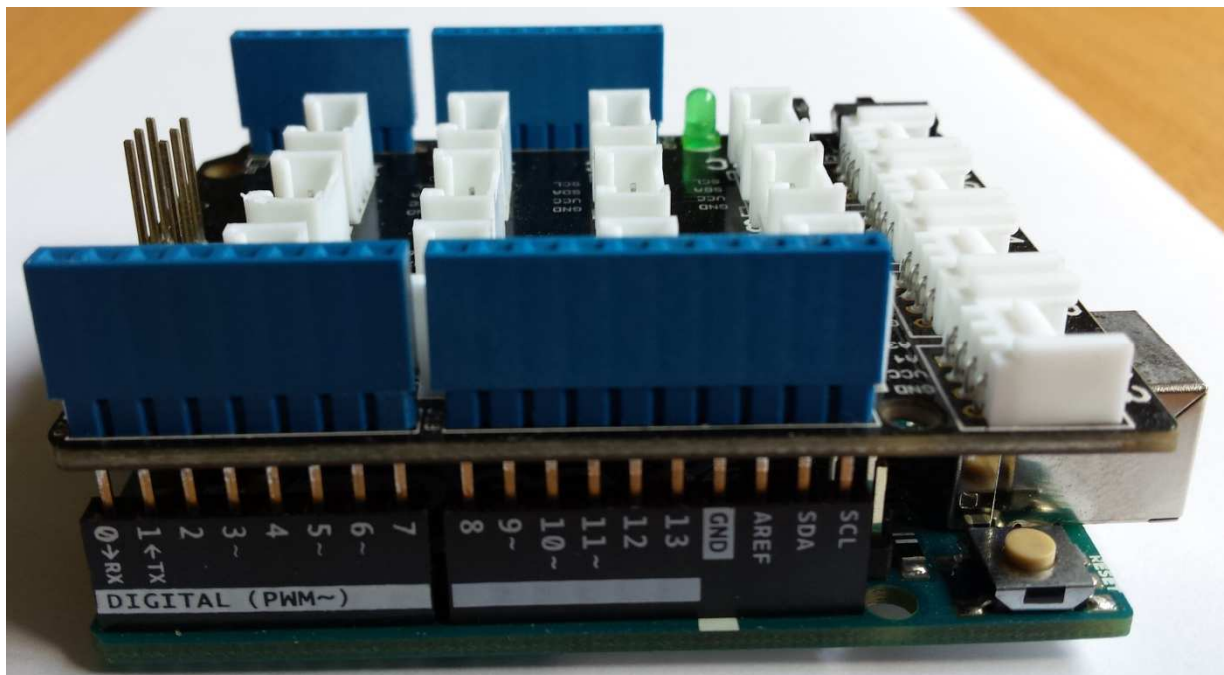


Figure 5 : Assemblage carte Arduino et Shield Grove

#### 1.4. Périphériques d'E/S utilisés

Cette section a pour but de présenter les ressources d'E/S qui pourront être utilisées tout au long du projet.

### 1.4.1. Module Led

Ce module permet de connecter une Led (Light-Emitting Diode) sur un des ports numériques de la carte Arduino (cf. Figure 6). Le shield intègre une résistance, sous la forme d'un potentiomètre, permettant de limiter le courant traversant la led (cf. Figure 7). Lorsque la broche 1 est à l'état haut (5V), la Led sera traversée par un courant et donc s'allumera.

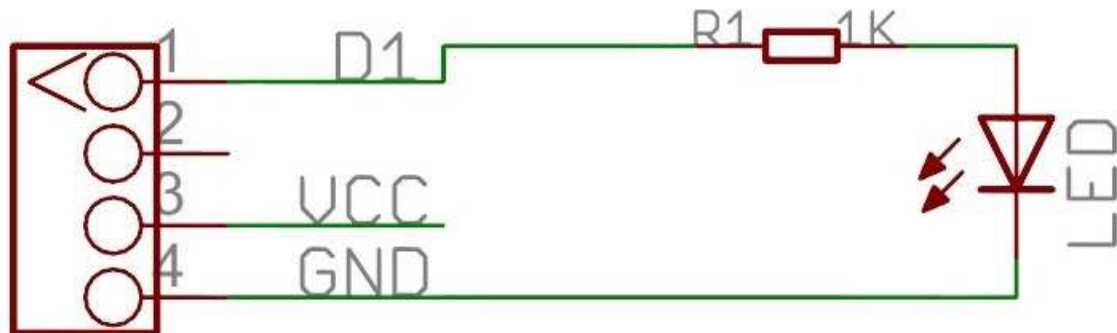


Figure 6 : Schéma de câblage du module Led



Figure 7 : Module Grove Led

Une Led est un composant polarisé. La cathode est connectée sur la masse tandis que l'anode est reliée sur une broche pouvant délivrer du 5V. Sur le module Grove, l'anode devra être connectée du côté '+'. Sur une Led, le repérage de l'anode et de la cathode s'effectue conformément à la figure ci-dessous (cf. Figure 8).

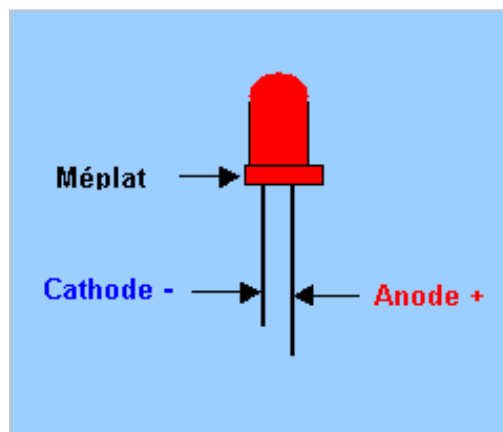


Figure 8 : Repérer l'anode et la cathode sur une Led



### 1.4.2. Module bouton

Ce module permettra de connecter un bouton poussoir sur un des ports numériques de la carte Arduino (cf. Figure 9).



Figure 9 : Module Grove bouton poussoir

Un appui sur le bouton poussoir provoque la mise à l'état haut (5V) de la broche « Sig », qui sera connectée sur un des ports numériques de la carte Arduino (cf. Figure 10).

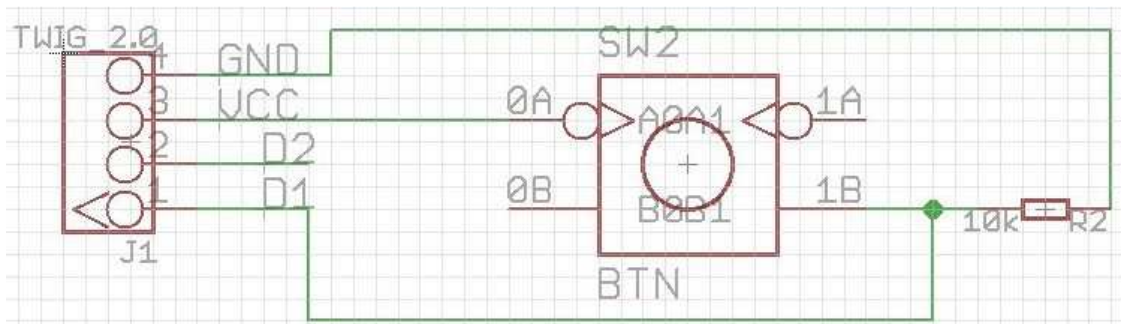


Figure 10 : Schéma de câblage du module Grove bouton poussoir

### 1.4.3. Module potentiomètre

Ce module permet de générer un signal analogique en utilisant un potentiomètre (résistance variable) de 10 kΩ (cf. Figure 11).



Figure 11 : Module Grove potentiomètre

La sortie de ce module sera donc connectée sur une des entrées analogiques de la carte Arduino.

#### 1.4.4. Module capteur de luminosité\_1

Ce module permet d'effectuer une mesure de 'intensité lumineuse amiante. Son principe de fonctionnement repose sur l'utilisation d'une résistance dont la valeur varie en fonction du flux lumineux (LDR : Light Dependent Resistor). (cf. Figure 12).

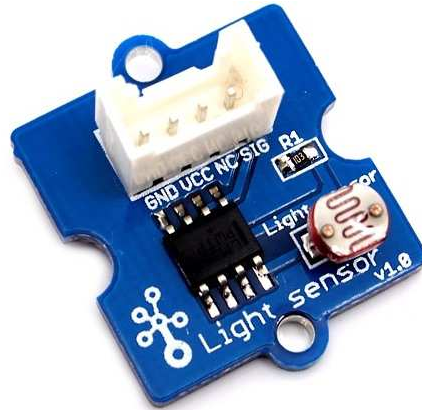


Figure 12 : Module Grove capteur de luminosité

#### 1.4.5. Module détecteur de lumière\_2

Ce module permet de réaliser des mesures d'intensité lumineuse, selon trois gammes de spectres lumineux (cf. Figure 13). La mesure de l'intensité lumineuse est obtenue sous forme numérique, via le bus de communication I2C. La gestion de ce bus I2C nécessitera la mise en œuvre de la bibliothèque de fonction Arduino « Wire » ainsi qu'une seconde bibliothèque spécifique au capteur utilisé. Cette bibliothèque sera fournie ultérieurement.

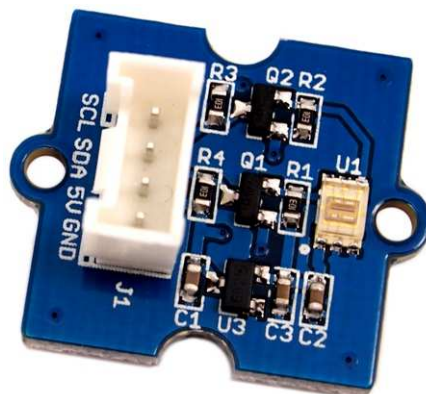


Figure 13 : Module Grove détecteur de lumière

#### 1.4.6. Module capteur d'humidité et de température

Ce module permet la mesure du taux d'humidité relative ainsi que de la température (cf. Figure 14). Il est basé sur l'utilisation du circuit DHT22. L'acquisition des mesures s'appuie sur l'utilisation du bus I2C et d'une bibliothèque spécifique afin de récupérer les mesures du taux d'humidité et de la température. Cette bibliothèque spécifique sera fournie ultérieurement.

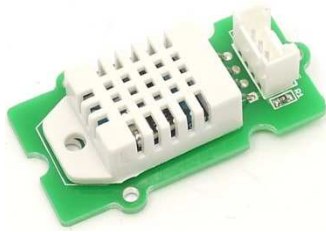


Figure 14 : Module Grove capteur d'humidité et température

#### 1.4.7. Module afficheur LCD

Cet afficheur permet de gérer 2 lignes de 16 caractères, ainsi que la couleur du rétroéclairage (cf. Figure 15). L'envoi des données s'effectue sur le bus I2C. Une bibliothèque spécifique pour gérer cet afficheur sera également fournie.



Figure 15 : Module Grove afficheur LCD\_I2C

#### 1.4.8. Module Led RGB

Il est possible d'utiliser une Led RGB, disponible sous la forme d'un shield Grove.



L'ensemble des informations utiles à sa mise en œuvre est accessible via un lien fourni en annexe.

## 2. Mise en œuvre

### 2.1. Environnement de programmation

#### 2.1.1. Installation

Les ressources logicielles peuvent être téléchargées depuis le site Arduino (cf. <http://arduino.cc/en/Main/Software>). Par la suite, nous travaillerons sous environnement Windows,


même si d'autres systèmes d'exploitation sont supportés. La version de l'environnement de programmation utilisée est la 1.0.6.

A l'issue de l'installation de l'environnement au sein du répertoire choisi, on doit retrouver les fichiers et répertoires organisés comme ci-dessous (cf. Figure 16).

Nom	Modifié le	Type	Taille
drivers	24/11/2014 16:07	Dossier de fichiers	
examples	24/11/2014 16:07	Dossier de fichiers	
hardware	24/11/2014 16:07	Dossier de fichiers	
java	24/11/2014 16:07	Dossier de fichiers	
lib	24/11/2014 16:07	Dossier de fichiers	
libraries	24/11/2014 16:07	Dossier de fichiers	
reference	24/11/2014 16:07	Dossier de fichiers	
tools	24/11/2014 16:07	Dossier de fichiers	
arduino	16/09/2014 15:46	Application	844 Ko
arduino_debug	16/09/2014 15:46	Application	383 Ko
cygiconv-2.dll	16/09/2014 15:46	Extension de l'application	947 Ko
cygwin1.dll	16/09/2014 15:46	Extension de l'application	1 829 Ko
libusb0.dll	16/09/2014 15:46	Extension de l'application	43 Ko
revisions	16/09/2014 15:46	Document texte	39 Ko
rxTxSerial.dll	16/09/2014 15:46	Extension de l'application	76 Ko
uninstall	24/11/2014 16:08	Application	402 Ko

Figure 16 : Répertoires et fichiers environnement Arduino 1.0.6

### 2.1.2. Description des fonctions de l'éditeur

Pour ouvrir et lancer l'environnement de programmation, il faut cliquer sur l'icône  , ce qui permet l'accès aux différentes fonctionnalités de l'éditeur. Selon la version de Windows utilisée, il est possible qu'une fenêtre d'avertissement s'ouvre. A l'issue du lancement de l'environnement, on obtient (cf. Figure 17). Par la suite, un programme est dénommé croquis ou sketch.

Cet environnement dispose de fonctions également accessibles par des raccourcis présents dans la barre contenant les icônes suivants (cf. Figure 18).

- « Vérifier » : permet de détecter des erreurs de syntaxe d'un programme ;
- « Téléverser » : permet de transférer le code exécutable d'un programme vers la mémoire du microcontrôleur;
- « Nouveau » : pour ouvrir un nouveau croquis ;
- « Ouvrir » : pour ouvrir un croquis déjà existant ;
- « Enregistrer » : pour sauvegarder un croquis ;
- « Moniteur Série » : permet d'ouvrir une fenêtre texte pour afficher des messages texte via le port série.

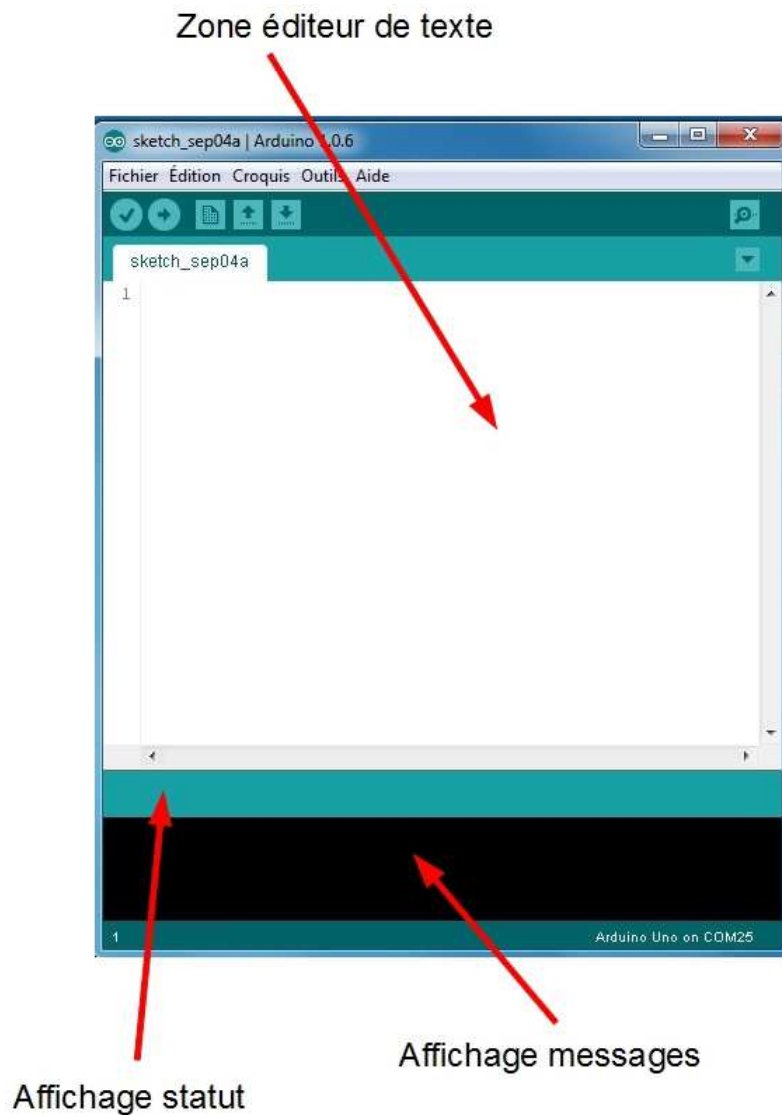


Figure 17 : Environnement de programmation Arduino

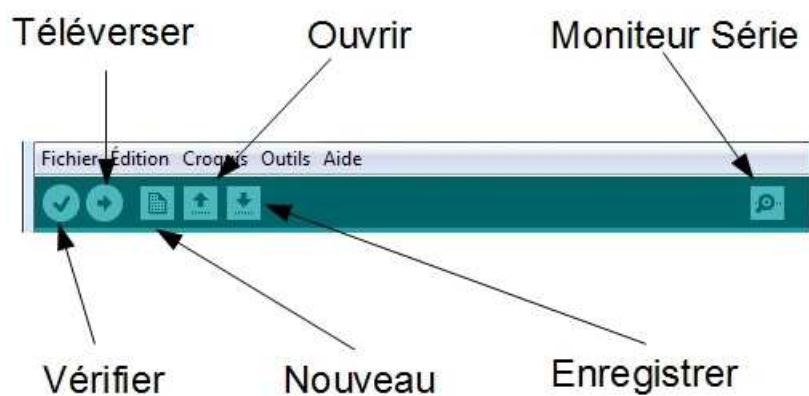


Figure 18 : Icones raccourcis environnement de programmation Arduino

### 2.1.3. Accès aux ressources logicielles

L'environnement de développement permet d'accéder aux ressources logicielles suivantes :

- Une base de programmes selon différentes rubriques, en passant par l'onglet *Fichier* puis *Exemples* (cf. Figure 19).
- Un accès direct à des ressources documentaires internet sur le site Arduino, en passant par l'onglet *Aide*.

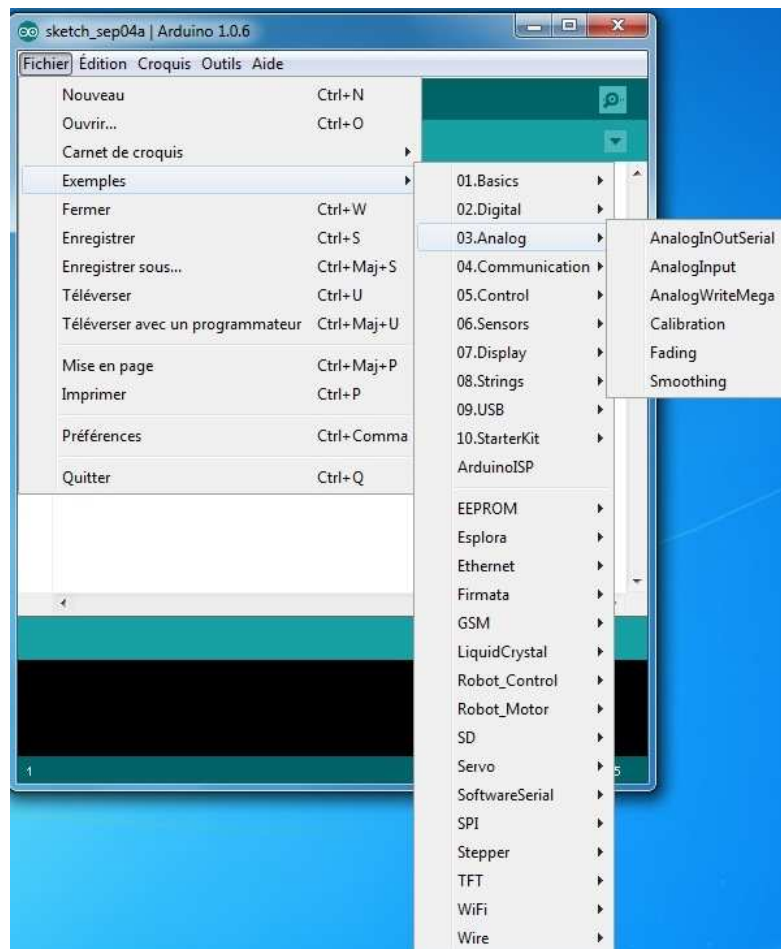


Figure 19 : Bases exemples de croquis Arduino

## 2.2. Structure de base d'un programme Arduino

### 2.2.1. Faire clignoter une Led

Il s'agit de reprendre les notions étudiées dans la première partie, mais sous un environnement de programmation intégré, l'environnement Arduino.

Dans la zone de l'éditeur de texte, taper le programme ci-dessous (cf. Figure 20). Le sauvegarder dans le croquis `Clignote_Led_V1_Arduino`. Ensuite, vérifier que le code compile sans erreur, en utilisant le raccourci « Vérifier ». Lorsque le code est compilé sans erreur, alors il est possible de le télécharger dans la mémoire de programme du microcontrôleur. Pour cela, il faut configurer l'environnement de programmation en spécifiant :

- Le type de carte Arduino cible au programme ;
- Le port COM de communication.



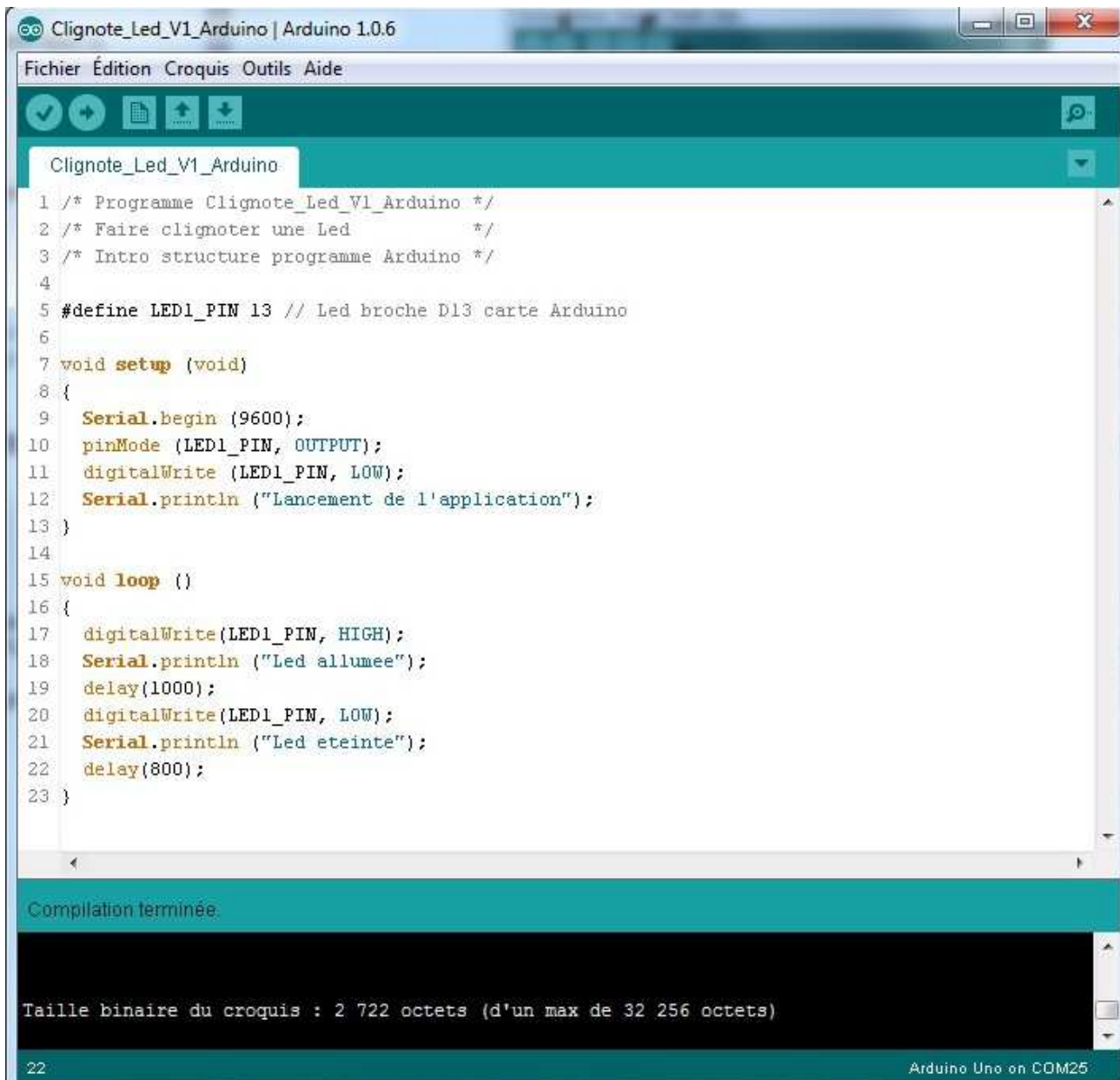


Figure 20 : Source de Clignote\_Led\_V1\_Arduino

Cette configuration s'effectue en sélectionnant l'onglet « Outils » puis « Type de carte » et en choisissant « Arduino Uno ».

Pour le choix du port de communication, il suffit de sélectionner « Outils » puis « Port série » et dans la liste qui apparaît, le numéro de port COM attribué lorsque la carte Arduino a été connectée sur un des ports USB (cf. Figure 21).

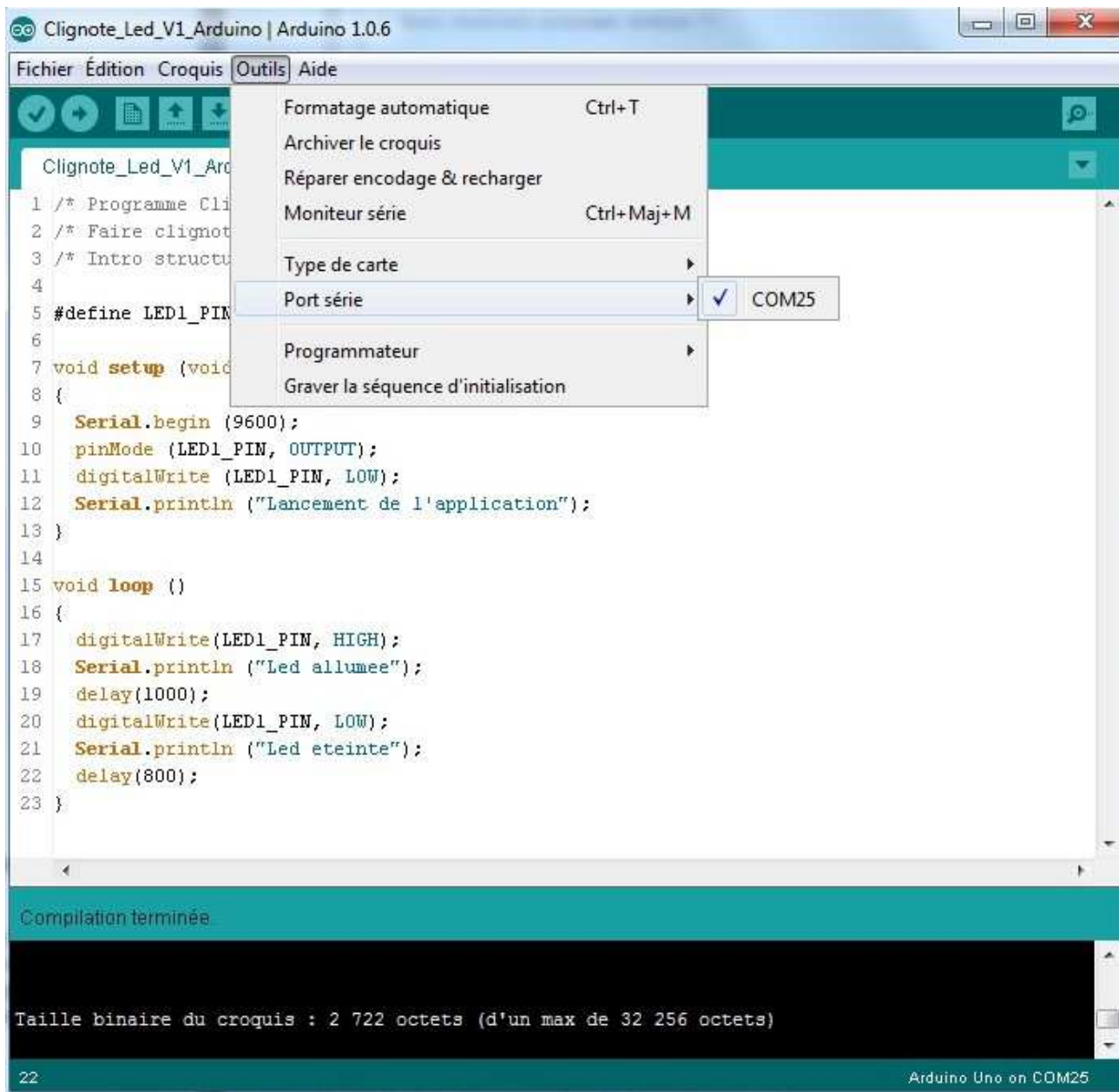
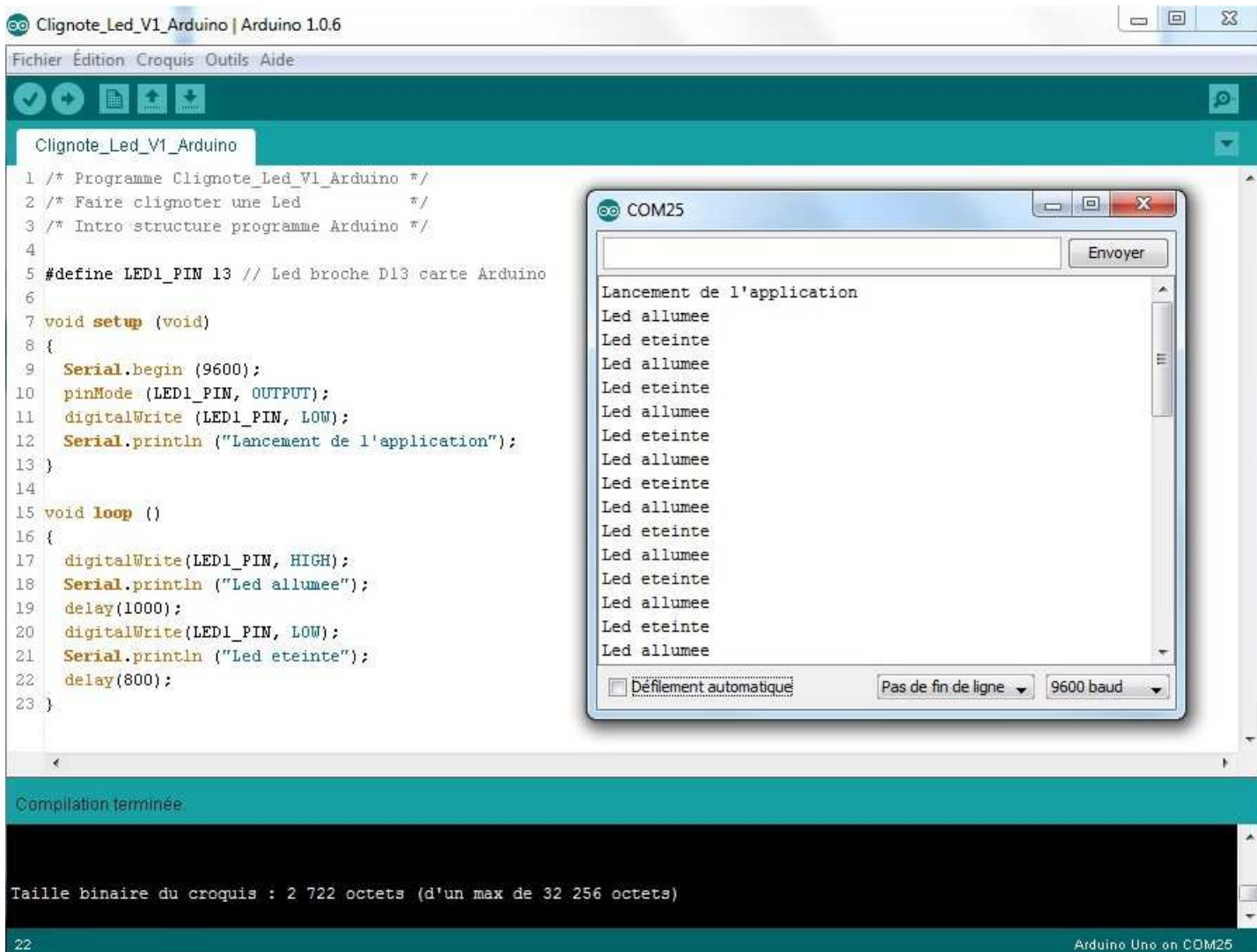


Figure 21 : Configuration matérielle de l'environnement Arduino

En cliquant sur le raccourci « Téléverser », cela provoque la compilation du programme ainsi que le téléchargement du code exécutable dans la mémoire de programme du microcontrôleur.

Dès que le téléchargement est achevé, le programme s'exécute sur le microcontrôleur. En ouvrant le moniteur série, il est alors possible de vérifier le résultat de cette exécution (cf. Figure 22).



**Figure 22 : Exécution de Clignote\_Led\_V1\_Arduino**

### 2.2.2. Etude du code Clignote\_Led\_V1\_Arduino

Un programme Arduino est structuré autour de deux fonctions de base, *setup* et *loop*.

La fonction *setup* est exécutée une seule fois lors du lancement du programme. Elle contient toutes les initialisations.

La fonction *loop* est exécutée de façon répétitive. C'est en fait une « boucle infinie ». **Aucune** fonction *main* n'apparaît.

Les principales ressources concernant les fonctions et les ressources utilisées sont disponibles sur le portail Arduino à l'adresse :

<http://arduino.cc/en/Reference/HomePage>

### 2.2.3. Travail à réaliser

A partir du code du programme Clignote\_Led\_V1 et des ressources disponibles depuis le portail Arduino, étudier les spécifications des différentes fonctions utilisées et rajouter dans le code source les commentaires permettant une meilleure compréhension du code et des fonctions utilisées.

## 2.3. Gestion des E/S analogiques

### 2.3.1. Contrôler l'intensité lumineuse d'une Led

Il s'agit de revenir sur la gestion des E/S analogiques au travers de l'application décrite ci-après. A partir de la lecture de la tension délivrée par un potentiomètre, on souhaite allumer une première Led si la valeur de tension aux bornes du potentiomètre est supérieure à un seuil donné  $T_0$ . De plus, on veut asservir l'intensité lumineuse d'une seconde Led en fonction de la tension délivrée par le même potentiomètre.

La sortie du potentiomètre sera connectée sur l'une des entrées analogiques A0, A1, A2, ou A3.

La première Led sera connectée sur un port numérique. La seconde Led peut être connectée sur l'une des broches D3, D5, D6, D9, D10 ou D11. Ce sont les seules broches qui permettent de générer en sortie un signal modulé en largeur d'impulsions (MLI) (ou PWM : Pulse Width Modulation)

Pour la lecture de la tension en sortie du potentiomètre, on pourra utiliser la fonction *analogRead()* dont la description est disponible via le portail Arduino :

<https://www.arduino.cc/en/Reference/AnalogRead>

Pour la génération d'un signal PWM, on pourra utiliser la fonction *analogWrite()*, dont la description est fournie via le lien ci-après :

<https://www.arduino.cc/en/Reference/AnalogWrite>

### 2.3.2. Travail à réaliser

Ecrire le programme Arduino *Arduino\_ES\_Led\_PWM* permettant de réaliser les fonctionnalités demandées. Vérifier que le résultat obtenu est conforme aux fonctionnalités spécifiées.

Vous préciserez les affectations des ressources du microcontrôleur ainsi que le mode de prise en compte de l'interruption externe.

## 2.4. Utilisation d'une bibliothèque : mesure température – humidité

### 2.4.1. Introduction

Afin d'illustrer l'utilisation de bibliothèques de fonctions, nous allons mettre en œuvre un capteur permettant d'effectuer une mesure de la température et du taux relatif d'humidité. Pour cela, nous allons utiliser le capteur DHT22, présenté dans un précédent paragraphe (cf. 1.4.6).

Les spécifications techniques de ce capteur sont disponibles sur [http://www.seeedstudio.com/wiki/Grove - Temperature and Humidity Sensor Pro](http://www.seeedstudio.com/wiki/Grove_-_Temperature_and_Humidity_Sensor_Pro).

Une seconde illustration sera proposée par l'utilisation d'un écran LCD, avec communication LCD (cf. 1.4.7). La documentation permettant sa mise en œuvre est disponible via le lien ci-après [http://www.seeedstudio.com/wiki/Grove - LCD RGB Backlight](http://www.seeedstudio.com/wiki/Grove_-_LCD_RGB_Backlight).

### 2.4.2. Prise en compte d'une bibliothèque de fonctions

L'environnement Arduino permet l'utilisation de bibliothèques additionnelles de fonctions, le plus souvent associées à la mise en œuvre de matériels périphériques. C'est le cas du capteur de

température – humidité DHT22. Au travers de cet exemple, nous verrons comment gérer ces bibliothèques.

### 2.4.3. Gestion des bibliothèques de fonctions sous Arduino

Toutes les bibliothèques de fonctions sont regroupées dans le répertoire *libraries*. Les conventions suivantes doivent être respectées afin de pouvoir les utiliser lorsque l'on se trouve dans l'environnement de programmation Arduino.

- Une bibliothèque est toujours constituée :
  - o D'un fichier source : C++ source file
  - o Du fichier d'en-tête associé : header file

Ces deux fichiers sont toujours placés dans le répertoire *libraries*.

Les règles de nommage suivantes doivent être suivies :

- Les suffixes du fichier source et du fichier d'en tête doivent être identiques au nom du répertoire qui contiendra ces deux fichiers.

Sous votre environnement Arduino, créer le répertoire nécessaire pour utiliser la bibliothèque de fonctions associée à l'utilisation du capteur de température et humidité. Ces deux fichiers sont disponibles sous Celene.

Répéter la même opération pour l'écran LCD.

Afin qu'une nouvelle bibliothèque soit prise en compte, il est indispensable de fermer puis de lancer de nouveau l'environnement Arduino.

Remarques : vous pouvez éditer les fichiers sources et d'en tête afin de connaître les fonctions proposées. Pour ouvrir ces deux fichiers, vous pouvez utiliser, par exemple, Codeblocks.

### 2.4.4. Mise en œuvre : **travail à réaliser**

Ecrire une application Arduino qui permettra d'afficher sur le **terminal série** les valeurs de température et de taux d'humidité, avec une périodicité de l'ordre de la seconde.

Reprendre ensuite le programme précédent pour afficher les mesures de température et de taux d'humidité **également** sur l'écran LCD.

# **Annexe**

## **Ressources bibliographiques**

### **Arduino Uno**

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

### **Environnement de développement Arduino**

<https://www.arduino.cc/>

### **Micro contrôleur**

Micro contrôleur Atmega 328P : <http://www.atmel.com/Images/doc8161.pdf>

### **Shields Grove**

Platine de connexion :

[http://www.seeedstudio.com/wiki/Grove-Base\\_Shield\\_V1.3](http://www.seeedstudio.com/wiki/Grove-Base_Shield_V1.3)

Module Led :

[http://www.seeedstudio.com/wiki/Grove\\_-\\_LED](http://www.seeedstudio.com/wiki/Grove_-_LED)

Module bouton poussoir :

[http://www.seeedstudio.com/wiki/index.php?title=GROVE\\_-\\_Starter\\_Kit\\_v1.1b#Grove\\_-\\_Button](http://www.seeedstudio.com/wiki/index.php?title=GROVE_-_Starter_Kit_v1.1b#Grove_-_Button)

Module détecteur de lumière\_1:

[http://www.seeedstudio.com/wiki/Grove\\_-\\_Light\\_Sensor](http://www.seeedstudio.com/wiki/Grove_-_Light_Sensor)

Module détecteur de lumière\_2:

[http://www.seeedstudio.com/wiki/Grove\\_-\\_Digital\\_Light\\_Sensor](http://www.seeedstudio.com/wiki/Grove_-_Digital_Light_Sensor)

Module capteur humidité et mesure de température :

[http://www.seeedstudio.com/wiki/Grove\\_-\\_Temperature\\_and\\_Humidity\\_Sensor\\_Pro](http://www.seeedstudio.com/wiki/Grove_-_Temperature_and_Humidity_Sensor_Pro)

Module afficheur LCD :

[http://www.seeedstudio.com/wiki/Grove\\_-\\_LCD\\_RGB\\_Backlight](http://www.seeedstudio.com/wiki/Grove_-_LCD_RGB_Backlight)

Module Led RGB

[http://www.seeedstudio.com/wiki/index.php?title=Twig\\_-\\_Chainable\\_RGB\\_LED](http://www.seeedstudio.com/wiki/index.php?title=Twig_-_Chainable_RGB_LED)

### **Codeblocks et environnement Arduino**

CodeBlocks pour Arduino

<http://arduino.dev/codeblocks/>