Animez un atelier de prototypage avec Arduino

d'une plante Emoji, qui est un prototype électronique qui vous permettra de savoir quand vous devez arroser vos plantes grâce à une carte Arduino, un capteur d'humidité et une matrice de leds ! 😁

Nous verrons ensemble comment programmer cette carte électronique, relier tous les composants entre eux et faire fonctionner votre prototype.

A l'issue de ce cours vous saurez :

1. Vulgariser des concepts liés à l'électronique
2. Comprendre les concepts clés de la programmation (boucles, conditions, etc)
3. Programmer une carte Arduino
4. Relier différents composants électroniques entre eux
5. Utiliser les séquences pédagogiques proposées et/ou en créer de nouvelles

Table de matières :

La carte Arduino, qu'est-ce que c'est ?

Avant, pour faire de l'électronique il fallait tout faire de A à Z : assembler plusieurs circuits imprimé, utiliser plusieurs logiciels pour faire son programme, etc. **Il manquait une carte électronique facilement accessible et programmable par tous**.

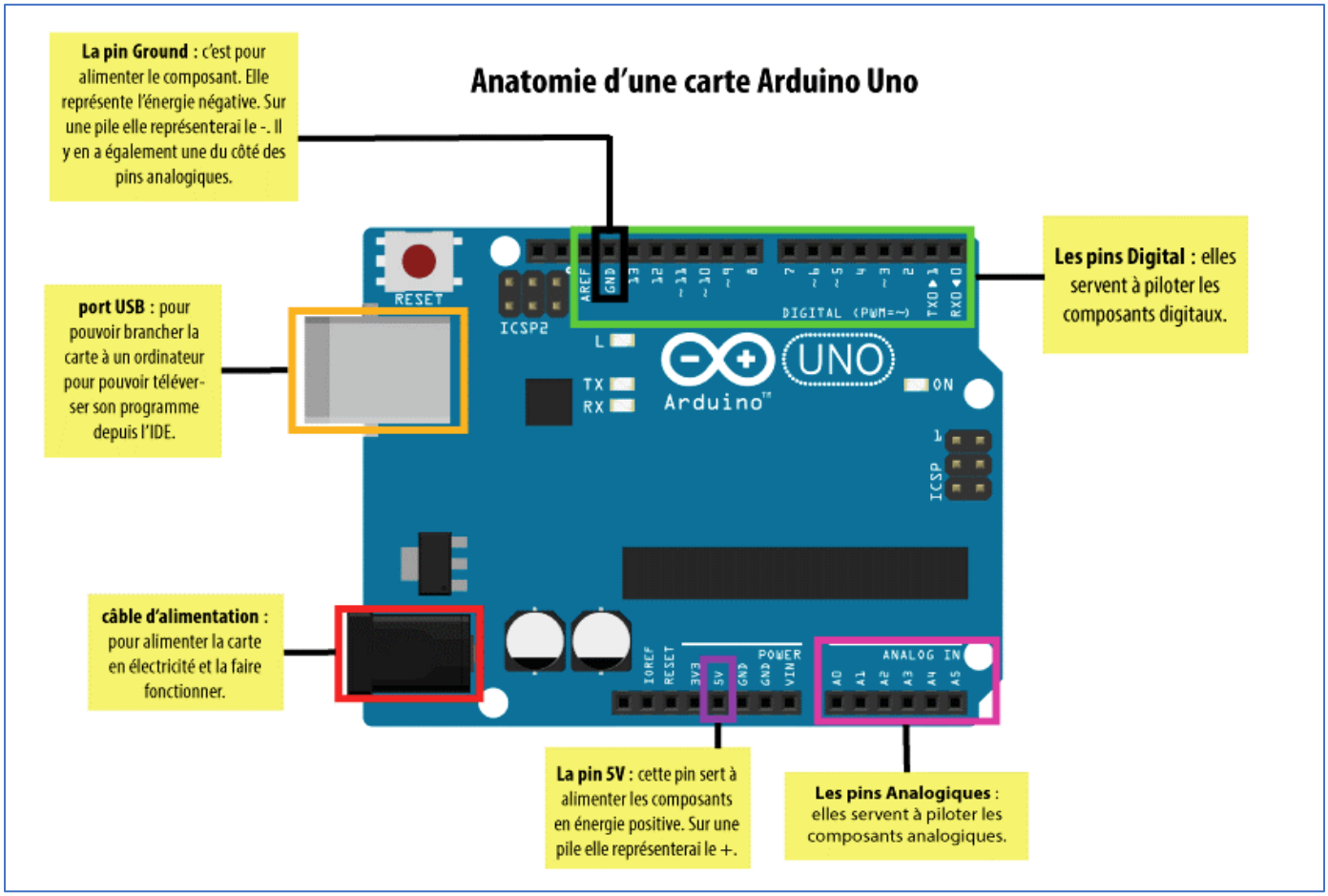
Arduino c'est tout simplement **une petite carte électronique programmable** et **un logiciel multiplateforme** (c'est à dire qui fonctionne sur tous les systèmes d'exploitation) qui permettent à tout le monde de **créer facilement un système électronique**.

À la base Arduino a été fait pour des designers et en vue de spectacles ou d'animations artistiques, pour des personnes qui ne connaissent donc rien à l'électronique et la programmation. C'est un "**plug and play**", il y a juste à connecter les fils au bon endroit, faire son programme et ça fonctionne !

Derrière Arduino il y a également une grande communauté, et donc plein de ressources pour apprendre facilement à l'utiliser !

En l'occurrence pour ce projet nous allons utiliser **la version Uno de l'Arduino**, qui est le modèle de carte le plus utilisé.

Voici un schéma qui vous explique la composition d'une carte Arduino Uno :



Pin Ground et Pin 5V : l'union fait la force.

Comme dans vos appareils électriques classiques, vos composants ont besoin d'être alimenté en électricité pour fonctionner.
Un peu à la manière d'une pile, notre Arduino comporte **un pôle positif** (+5V) et **un pôle négatif** (GND) qui vont agir ensemble **pour alimenter vos composants**.

Digital vs Analogique : quelle est la différence ?

Selon le type d'instructions que vous allez donner à votre carte électronique, celle-ci va effectuer deux types d'actions. Selon le type d'action que vous souhaitez lui faire effectuer, vous devrez brancher vos fils soit du côté des pins digitales, soit du côté des pin analogiques.

Les pins digitales :

Celles-ci vont agir de manière binaire avec le courant électrique, **c'est à dire qu'elles ne peuvent être que dans deux états différents** : le courant passe (la tension électrique est à 5 Volts) ou ne passe pas (elle est à 0 Volt).
Vous allez pouvoir effectuer deux actions :

- **Agir** : piloter le courant électrique
- **Récupérer une information** : savoir l'état de la pin (le courant passe-t-il ou non?)

Pour vous donner un exemple concret : vous voulez faire un programme pour qu'une led clignote.
Elle n'a donc que deux actions à effectuer : **s'allumer et s'éteindre, et ce en boucle**. Cette action ne nécessite aucune nuance : elle est **soit dans un état** (allumée) **soit dans un autre** (éteinte).
Pour ceci vous devrez donc brancher vos fils de connexion sur **les pins digitales**.

Les pins analogiques :

Les pins analogiques sont utilisées dès qu'on cherche à récupérer une information qui est plus complexe que du simple binaire.
À l'inverse des pins digitales qui vous indique seulement si la tension est de 0 ou de 5 Volts, les pins analogiques vous permettent de connaître la graduation de cette tension. Par exemple si il est de 0 Volt, de 1 Volt, de 2 Volts, etc
Vous utiliserez également celles-ci lorsque vous chercherez simplement à récupérer des informations, sans interagir avec l'appareil auquel vous êtes connecté.
Par exemple : vous voulez faire un programme qui permet de récupérer le taux d'humidité de d'un pot de terre grâce à un capteur. Pour ceci vous devrez donc brancher vos fils de connexion sur les pins analogiques.

Prototypage électronique, objet connecté : quelle est la différence ?

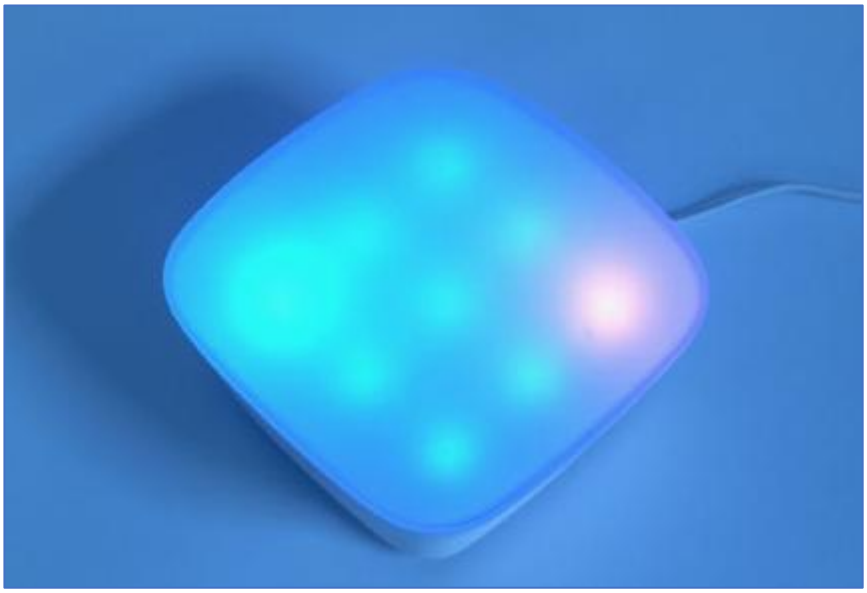
Les gens font souvent la confusion entre prototype électronique et objet connecté. Ce sont pourtant deux choses bien différentes. Tout ce qui est prototypage électronique n'est pas objet connecté, et tout ce qui est objet connecté n'est pas prototypage électronique. Commençons déjà par éclaircir ces deux notions :

Qu'est-ce qu'un prototypage électronique?

Un prototypage électronique est donc **une première version d'un objet avec des composants électroniques**.
Un prototype est la première étape de la réalisation d'un projet. Il peut être incomplet mais souvent il possède déjà les qualités techniques et les caractéristiques de fonctionnement du nouvel objet.
On le construit souvent pour démontrer ou infirmer le bien-fondé d'un ou plusieurs concepts, et ce avant sa commercialisation.
Un prototype n'est donc pas nécessairement un objet connecté. 😉
Qu'est-ce qu'un objet connecté ?
Un objet connecté est tout simplement un objet dont auquel on ajoute une connectivité (que ce soit via internet, bluetooth ou autre), dans le but de faire communiquer cet objet avec une application ou un service web.

Pour vous donner un exemple plus concret, prenons l'exemple du premier objet connecté : la lampe DAL.

On est d'accord pour se dire que la fonction première d'une lampe n'est pas d'aller sur le web ou de servir de périphérique informatique. 😊



La lampe DAL, sortie en 2003 et créée par Rafi Haladjian (entrepreneur français), possédait 9 leds qui permettent à la lampe de s'allumer de différentes couleurs selon des évènements variés : le cours de la Bourse, la météo, la pollution les alertes Google ou encore des envois de "messages couleurs" par SMS ou email. La lampe possède également un micro et des capteurs de "toucher".

Faites votre premier “Hello World” avec Arduino

La première chose à faire : installer l’IDE Arduino

Avant de pouvoir bidouiller votre carte Arduino, il va vous falloir installer le logiciel qui va vous permettre de la programmer. On l’appelle l’IDE Arduino.

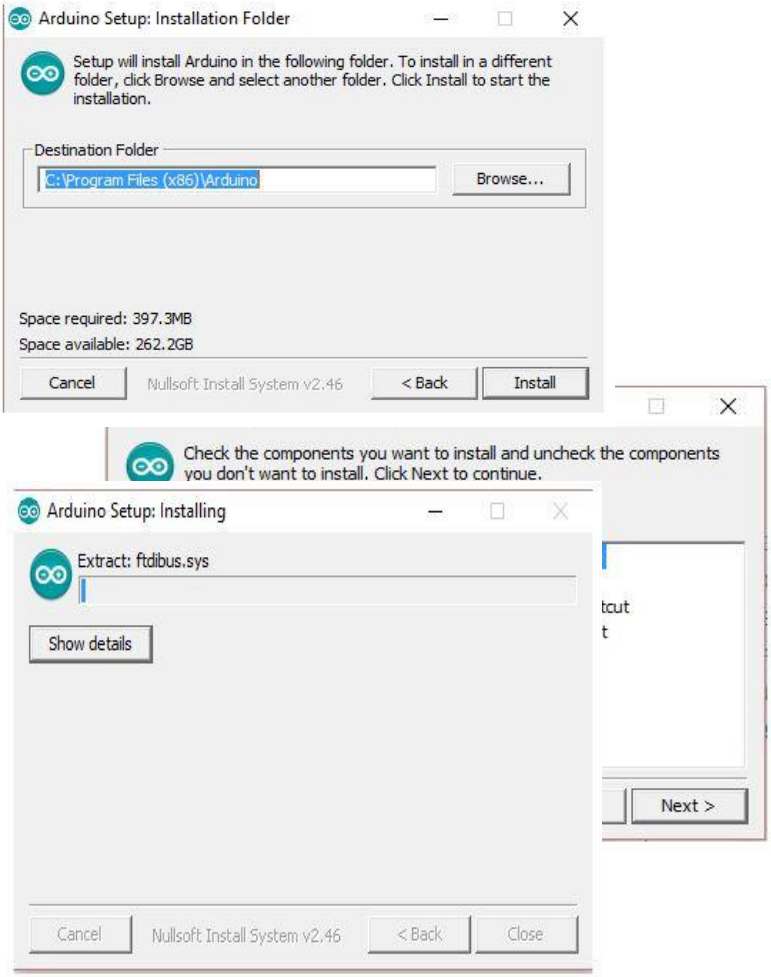
IDE est l’abréviation en anglais *Integrated Development Environment*. En français on dit EDI pour *Environnement de Développement Intégré*.

C’est un ensemble d'outils (éditeur de texte pour la programmation, des fonctions pré-écrites, etc) **pour augmenter la productivité des programmeurs qui développent des logiciels**. Certains environnements sont dédiés à un langage de programmation en particulier (ici le langage Arduino).

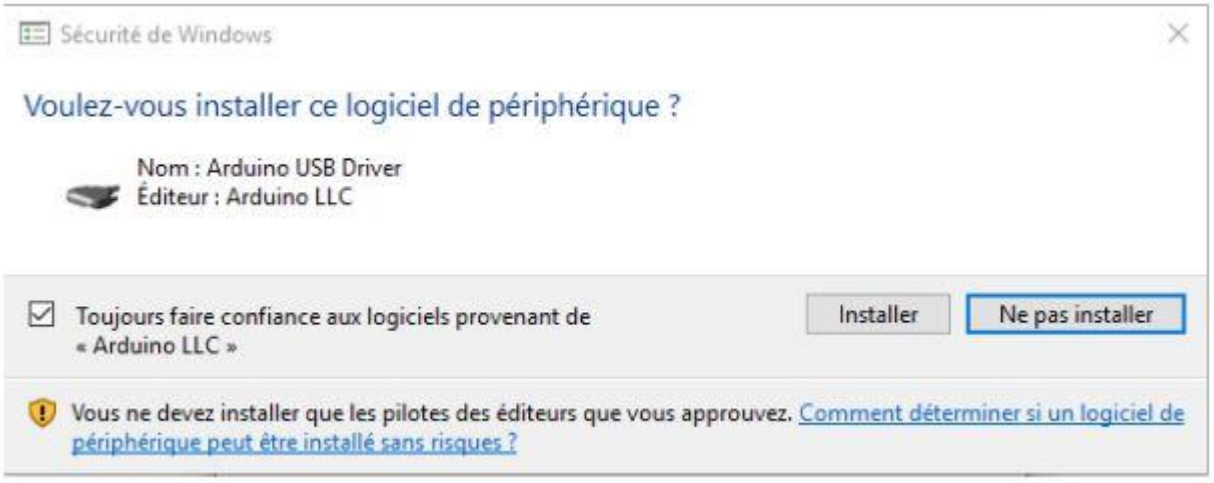
Pour l’installer rien de plus simple :

1. Rendez-vous sur le site d’Arduino : https://www.arduino.cc/en/Main/OldSoftwareReleases
2. Téléchargez le logiciel selon votre OS (Linux, Mac, Windows)
3. Acceptez les conditions générales du logiciel

4. Si ce n’est pas déjà fait, sélectionnez toutes les options du logiciel proposées
5. Cliquez sur “**Install**” pour installer le logiciel



6. Attendez quelques minutes que celui ci s’installe.
7. Si vous êtes sous Windows, acceptez l’installation.



8. Une fois l’installation terminé, cliquez sur “**close**”.
9. Le tour est joué ! Votre logiciel s’ouvre et vous êtes fin prêts à programmer votre carte! 😃

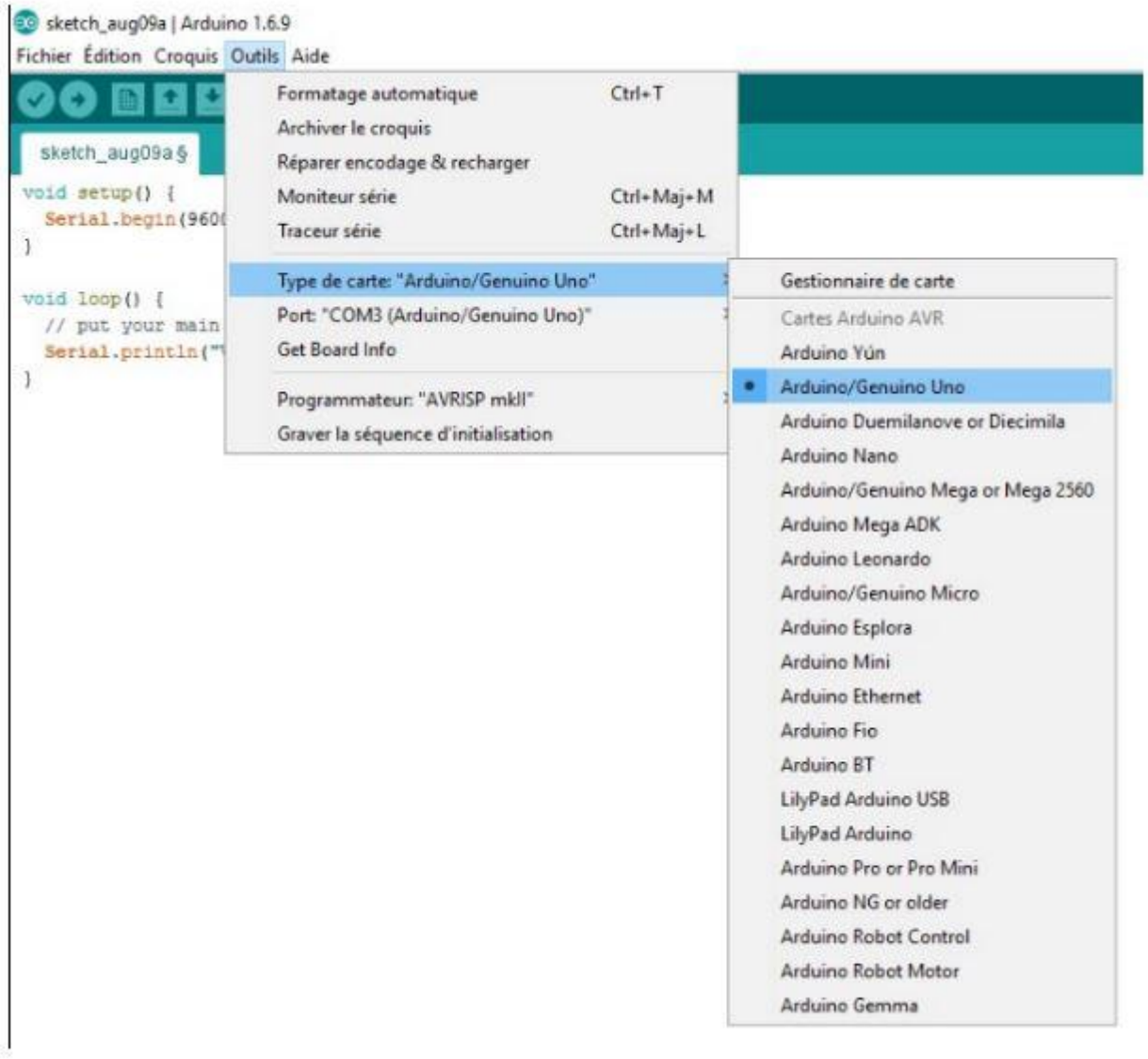
Un premier exercice pour se familiariser avec l'IDE : utiliser le moniteur

Bien souvent, vous aurez besoin d'interagir avec l'Arduino et de visualiser les informations qu'il reçoit. C'est le cas lorsque vous voulez afficher un message en texte sur votre écran d'ordinateur par exemple.

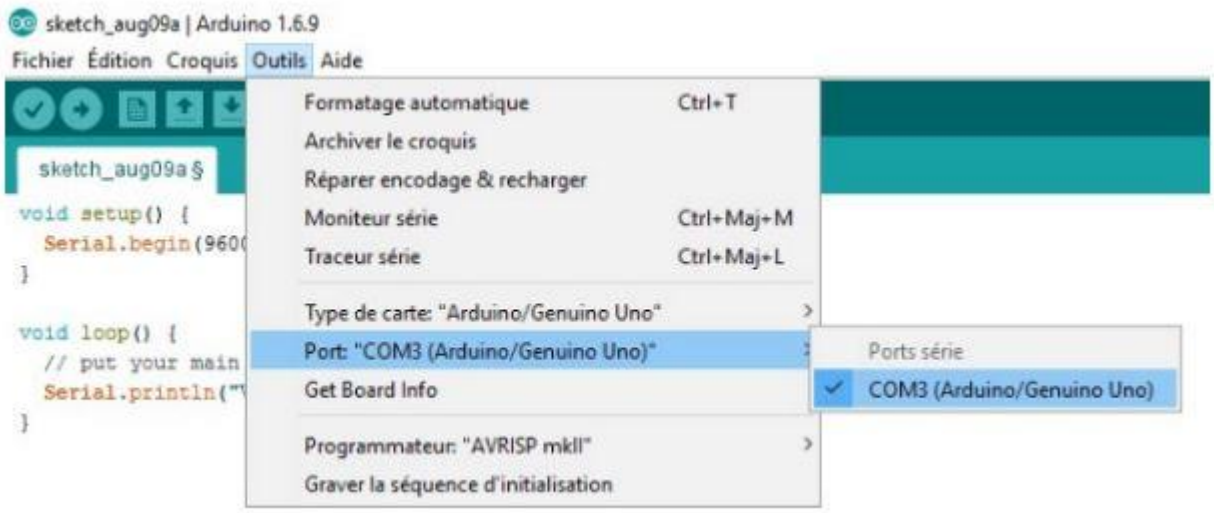
Pour cela vous allez utiliser ce qu'on appelle **une connexion "série"** entre l'Arduino et notre ordinateur. Une connexion "série" c'est tout simplement **une connexion qui permet de s'échanger du texte**.

Pour que le message s'affiche, vous allez devoir faire plusieurs manipulations.

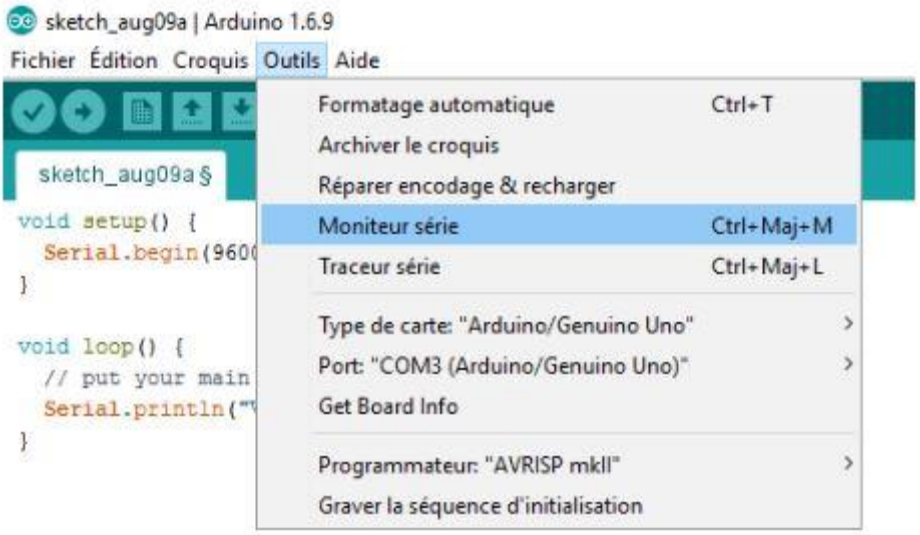
1. Connectez votre Arduino à votre ordinateur.
2. Sélectionnez dans l'IDE Arduino **le bon type de carte** et **le bon port USB**.
3. Pour sélectionner la carte, allez dans "**Outils**" et dans "**Type de carte**".



Pour sélectionner le port, allez dans “**Outils**” puis dans “**Port**”



4. On va utiliser ce qu’on appelle **un moniteur**. Vous le trouverez dans l’onglet “**Outils**”, puis “**Moniteur série**”.



Maintenant que vous avez compris comment fonctionne l’IDE, vous êtes fin prêt à programmer votre premier “Hello World” !

Réalisez les branchement électroniques de votre plante Emoji

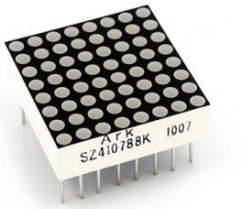
Avant de mettre les mains dans le code, il nous faut d'abord faire les bons branchements électroniques pour que notre prototype soit prêt à recevoir le programme que vous allez vouloir lui téléverser.
Voici la liste du matériel dont vous allez avoir besoin pour ce chapitre :

- une carte Arduino Uno
- une matrice de leds
- un capteur d'humidité
- un matériel de branchement de votre carte : câble d'alimentation micro USB (Chargeur) et un câble usb/micro usb pour pouvoir le connecter à votre ordinateur.
- 8 fils de connexion
- votre ordinateur





La matrice de leds

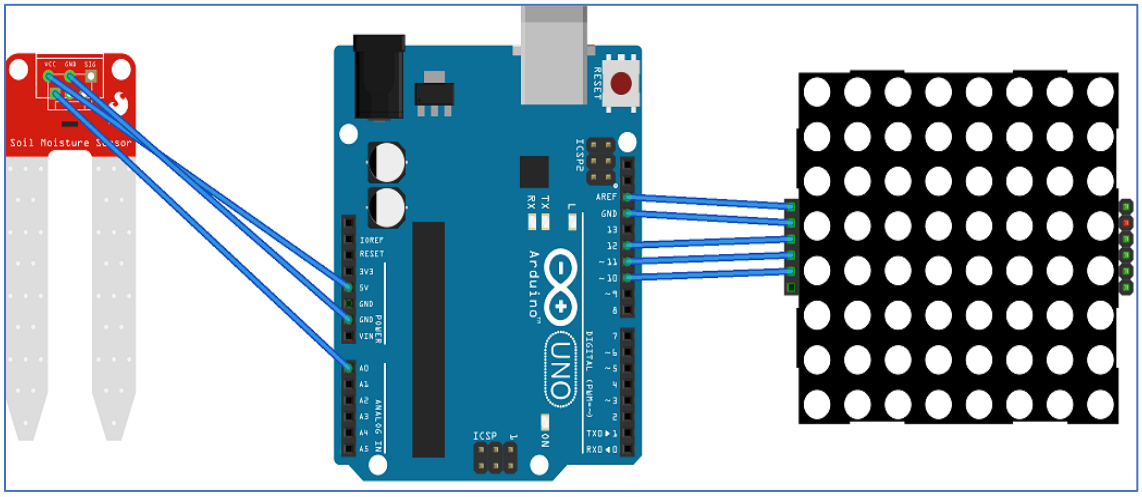


Une matrice de leds c'est quoi ? C'est un ensemble de LED qui constitue un tableau où l'on peut éclairer chacune de celles-ci indépendamment. Les LED sont rangées en lignes et en colonnes.

Notre matrice de leds est un composant un peu spécial, celle-ci utilise seulement 3 pins digitales pour piloter ses 64 leds. Comment est-ce possible ? C'est simple, notre Arduino communique avec notre capteur en envoyant des messages via un circuit intégré. Le dit circuit décode les messages envoyés par la matrice et pilote l'allumage des leds.

Faire nos branchements

Voici une photo pour vous aider. N'hésitez pas à vous référer au plan de l'anatomie des composants de votre prototype et à la photo des branchements ci-dessous afin de vous aider.



Le plus important est de comprendre où brancher quel fil, la définition précise des acronymes n'a que peu d'importance et vous n'aurez pas forcément le temps de l'expliquer en atelier.

Pour la matrice de leds :

VCC -> 5V (ou AREF selon les cartes)

GND -> GND

DIN -> 12 (permet d'envoyer les données : telle pin s'allume, telle pin tu s'éteint)

CS -> 11 (permet de choisir avec qui communiquer, dans le cas où il y a plusieurs composants)

CLK -> 10 (permet de régler la vitesse de communication entre l'Arduino et le composant)

Pour le capteur d'humidité:

A0 -> A0 (c’est la pin dont la tension variera en fonction du taux d’humidité de la plante)

VCC -> 5V

GND -> GND

Pourquoi faire les branchements de cette façon et pas d’une autre ?

Notre matrice de Leds doit être branchée du côté digital et notre capteur d’humidité du côté analogique.

Pourquoi forcément brancher la matrice de led du côté digital et le capteur d’humidité du côté analogique ? Rappelons-nous rapidement de la différence entre les deux :

- **Pins digitales** : lorsque l'on cherche à faire des actions “binaires”, sans nuances. En l’occurrence, on veut juste indiquer sur la matrice quelle led s’allume ou ne s’allume pas.
- **Pins analogiques** : lorsque l’on veut récupérer la valeur de la tension qui sort du capteur d’humidité (0 Volts, 3 Volts, etc...).

Notre capteur d’humidité va faire varier sa tension en fonction de l’humidité de la plante. Plus la terre est humide, plus la tension est élevée et approchera des 5 Volts.
Par exemple, si la terre est moyennement humide, la tension sera aux alentours des 2-3 Volts ; si la terre est très sèche, la tension approchera plutôt de 0 Volt.

Récupérez le taux d'humidité de votre plante Emoji !

Etape 1 : Comment fonctionne notre capteur d'humidité ?

Notre capteur d'humidité va faire varier la tension du pin A0 entre 0 Volt et 5 Volts. Mais notre Arduino ne peut récupérer qu'une valeur analogique comprise entre 0 et 1023.

	valeur minimum	*valeur maximum*
Valeur en volt (analogique)	0	5
Valeur numérique	0	1023

Une valeur analogique correspond à une valeur numérique.
La valeur récupérée en volt donnera forcément **une valeur proportionnelle** à la valeur numérique.
Cela veut dire que si la tension est, par exemple, de 2.5 Volts, la valeur récupérée par l'Arduino sera de 512.
Comment a-t-on réussi à calculer cette valeur ?
Pour calculer cette valeur, nous avons fait **un simple produit en croix** (comme lorsque vous calculez des pourcentages).

Valeur en volt (analogique)	**2.5**	**5**
Valeur numérique	512	**1023**

Ici vous multipliez donc 2.5 par 1023 et divisez le tout par 5 pour obtenir votre valeur numérique :
(2.5*1023)/5 = 511.5
Etant donné que l'on doit toujours arrondir le résultat à l'entier supérieur, **l'équivalent numérique de 2.5 Volts est donc 512**.

Etape 2 : Récupérer le taux d'humidité de la plante

La commande que nous allons utiliser est la suivante : `int humidity = analogRead(0);`

Nous disposons d'une fonction analogRead qui va nous permettre de récupérer la valeur numérique retournée par le capteur d'humidité.
La fonction prend en paramètre **le numéro de la pin** (pour nous ce sera la pin 0).
Pour réutiliser notre résultat, on le stocke dans ce que l'on appelle **"une variable".**
Qu'est-ce qu'une variable ?
Prenons un exemple imagé : **une variable c'est comme un tiroir.** On lui met une étiquette et on peut ranger quelque-chose dedans. Par exemple, j'ai un tiroir où je vais mettre une étiquette "tiroirOutil" et ranger un outil dedans.
On va maintenant définir quel type d'outil on va mettre dans notre tiroir. Par exemple on ne pourra mettre que des marteaux. Notre marteau peut prendre des arguments, c'est à dire qu'on peut lui donner des caractéristiques.

`int` Le type de la variable

`humidity` Le nom de la variable

`analogRead` Le nom de la fonction qui permet de récupérer la valeur numérique d'un pin analogique

`(0)` Le numéro de la pin

Etape 4 : Adapter l'Emoji en fonction du taux d'humidité reçu

Maintenant que l'on a récupéré l'humidité de la plante et que nous l'avons dans une variable, le mieux serait d'afficher sa valeur dans le moniteur série afin de vérifier que tout fonctionne et de faire des tests.
Rappelez vous que pour utiliser le moniteur série il faut initialiser la vitesse de communication entre Arduino et l'ordinateur (c'est une obligation) en ajoutant cette commande dans la fonction **setup** :

```
Serial.begin(9600);
```

Après avoir ajouté cette commande (et seulement après!) dans la fonction **loop** vous pouvez afficher la variable "humidity" dans notre cas à l'écran avec la commande suivante :

```
Serial.println(humidity);
```

Pour que le message s'affiche, on va devoir faire plusieurs manipulations. La première chose à faire est de sélectionner dans notre logiciel Arduino le bon type de carte et le bon port USB.
Vous trouverez le moniteur dans l'onglet "**Outils**", puis "**Moniteur série**".
À vous de faire des tests en branchant le capteur dans le pot d'une plante sec, bien portante ou même dans un verre d'eau.
Pour notre plante Emoji, nous considérons que si le capteur retourne une valeur numérique inférieure à 300 : la plante a besoin d'eau. On a choisi cette valeur de 300, mais dans votre code vous pouvez choisir de faire varier cette valeur entre 300 et 900 selon votre préférence et d'après vos tests.
On va utiliser ce que l'on appelle une condition pour vérifier si la valeur numérique du capteur d'humidité est inférieure à 300 et afficher l'émoji correspondant.

Une condition en programmation c'est quoi ?
C'est tout simplement le fait de **tester une hypothèse**. Si l'hypothèse se vérifie, une action se déclenche , sinon ça en déclenche une autre.

Dans le cas de notre plante Emoji :

Si la valeur numérique du capteur d'humidité est inférieure à 300, alors la matrice de leds affichera le smiley triste. ☹

Sinon, la matrice de leds affichera le smiley heureux. 😃

En code ça donne quoi ?

```
if ( humidity < 300 ) {

<code pour le smiley triste>

 } else {

<code pour le smiley content>

 }
```

Programmez le module LED

Deux fonctions : “setup” et “loop”

Lorsque vous ouvrez votre IDE Arduino, deux fonctions vides sont déjà écrites :

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

A quoi servent-elles ?

Les instructions se trouvant dans **la fonction setup** ne s’exécutent **qu’une seule fois** : au démarrage.

En revanche, les instructions se trouvant dans **la fonction loop** s'exécutent **à l’infini** (comme dans le chapitre 2 lorsque nous avons fait clignoter notre petite led).

En l'occurrence nous allons renseigner la fonction **setup** avec un morceau de code qui va mettre **toutes les leds de votre matrice à zéro**.

Dans la fonction **loop** nous allons mettre le code du programme qui va permettre à **nos leds de réagir selon le taux d’humidité récupéré par notre capteur**

Etape 1 : Inclure la bibliothèque : “#include <LedControl.h>”

Comme nous l’avons vu dans le chapitre 3, notre matrice de leds est un composant “intelligent” qui communique avec l’Arduino par message. Il serait compliqué d’écrire nous même ces messages et de les envoyer à la matrice.

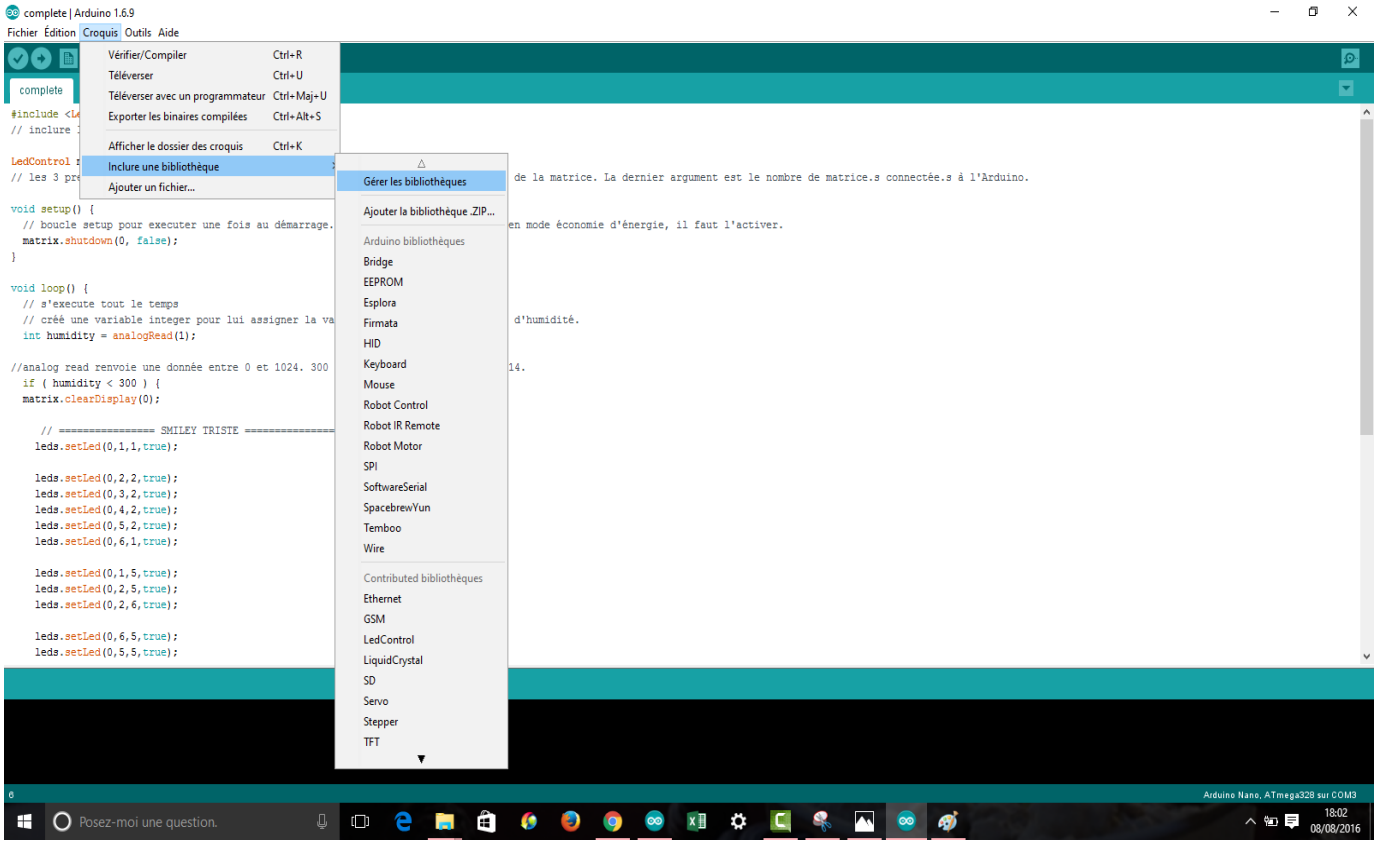
Vous allez donc devoir nous servir d’une bibliothèque qui vous permet de faire fonctionner la matrice de leds.

C’est quoi une bibliothèque en programmation ?

Pour prendre un exemple imagé, une bibliothèque c’est un peu comme un guide de conversation. Un guide de conversation, c’est un petit livre qui vous donne des phrases toutes faites pour communiquer dans une langue que vous ne connaissez pas.

En programmation, une bibliothèque vous donne un ensemble de lignes de codes toutes faites pour vous éviter de les écrire vous-même.
La bibliothèque **LedControl** va servir à contrôler la matrice de leds.

Pour que l'on puisse l'utiliser quand on en a besoin, il faut installer cette bibliothèque.
Pour ce faire, allez dans l'onglet "**croquis**", cliquez sur "**inclure une bibliothèque**" puis sur "**gérer les bibliothèques**".



Ensuite, la première ligne de code que vous allez renseigner sera :

```
#include <LedControl.h>
```

Vous demandez à votre programme d'inclure la bibliothèque **LedControl**, qui vous permet de contrôler les différentes leds de la matrice.

Etape 2 : Initialiser le programme : "LedControl matrix=LedControl(12,11,10,1);"

Comme je l'ai déjà expliqué précédemment, votre carte Arduino est un peu bête. Vous devez lui indiquer chaque instruction. Votre matrice de leds est d'un côté et votre carte Arduino est de l'autre. Si vous n'indiquez pas à votre carte que cette matrice existe : elle ne pourra pas le deviner.

Maintenant vous devez donc créer **une variable de type LedControl** et définir quelles pins vous allez utiliser pour que votre arduino sache comment communiquer avec votre matrice de leds.

Lorsque l'on initialise un LedControl, celui-ci prend 4 arguments :

- les 3 premiers arguments sont les 3 pins (ou épingles) utilisées sur la carte Arduino
- le dernier indique si il y a une ou plusieurs matrices connectées à l'Arduino

Concrètement, si on le décompose, ce code signifie :

LedControl Maintenant, nous créons une nouvelle variable de type LedControl.
matrix= Je vais ranger des informations dans la variable "matrix" (matrice en anglais).
LedControl(12,11,10... Nous utilisons les pins 12,11 et 10 sur l'Arduino.
...1); Il est possible d'utiliser plusieurs matrice de leds. Dans notre cas, il n'y aura qu'une seule matrice de leds attachée à l'Arduino.
Il est important de ne mettre ce code ni dans void setup ni dans void loop. Pourquoi ? **Parce qu'on veut qu'elle ait une portée globale et non locale.**

Etape 3 : Activer la matrice de leds : matrix.shutdown(0, false);

Avant de faire notre programme, il faut dire à notre matrice de se réveiller.
Vous vous souvenez de notre fonction setup qui ne s'exécute qu'une fois au démarrage ? Et bien il va être temps de l'utiliser !
C'est dans cette fonction qu'on va demander à notre matrice de leds de se réveiller car on va avoir besoin d'elle ! On lui indique tout simplement d'utiliser cette commande :

```
matrix.shutdown(0, false);
```

Ce qui traduit littéralement en français veut dire : matrice, tu n'es pas éteinte ! 😁

Etape 4 : Premier test pour voir si l'Arduino et la matrice de leds sont connectés

Essayons maintenant d'allumer une led !

Dans la fonction loop, écrivons notre première ligne de code pour vérifier que notre Arduino et notre matrice de leds sont bien connectés.

Votre code va ressembler à ceci :

```
matrix.setLed(0,1,1,true);
```

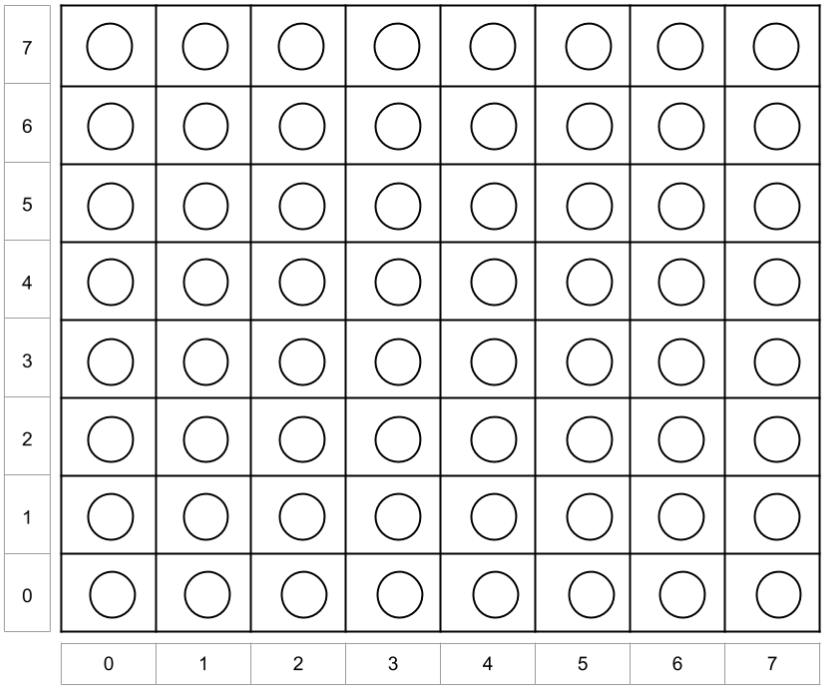
Comme pour l'initialisation de notre matrice, nous allons décortiquer cette phrase pour bien la comprendre :

matrix la première chose qu'on va faire est d'indiquer que l'on veut interagir avec une des leds de la matrice en utilisant l'objet matrix.

`.setLed` cette action nous permet de modifier l'état d'une led c'est à dire de l'allumer ou de l'éteindre.
`(0,1,1,true);` on va indiquer quatre paramètres : les trois indices de position de notre led et son état (allumée ou éteinte).
Représentez-vous votre matrice de leds comme un tableau sur lequel vous pouvez cocher certaines cases.

Dans ce tableau un peu spécial, on commence à compter à partir de 0. Donc par exemple si vous voulez allumer la led qui est à la deuxième ligne et dans la deuxième colonne, vous allez devoir indiquer :

- que vous utilisez la première matrice de leds
- que vous voulez allumer la led qui se situe sur la ligne 1
- et qui se situe dans la colonne 1



Vous pouvez télécharger une version vierge de la matrice de leds pour en distribuer à vos élèves. Ça sera plus facile pour eux (et pour vous) de savoir ce qu'ils vont devoir faire comme code si ils peuvent avoir le modèle papier sous les yeux 😁

C'est une notion importante, n'hésitez pas à y rester un petit moment si vous sentez que certains élèves ont du mal à comprendre et à faire plusieurs petits exercices de retranscription.

Par exemple, si je vous donne ce schéma saurez-vous le retranscrire en code ? 😁

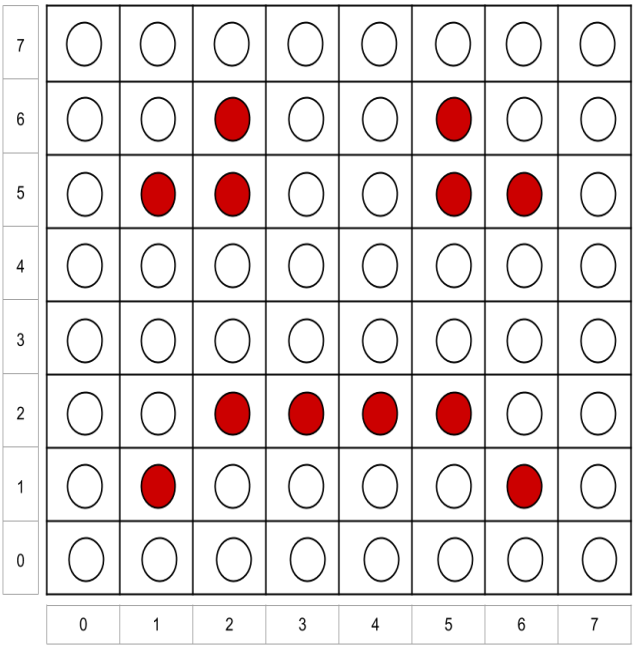
Voici la réponse :

```
// ================ SMILEY TRISTE ================
matrix.setLed(0,1,1,true);

matrix.setLed(0,2,2,true);
matrix.setLed(0,3,2,true);
matrix.setLed(0,4,2,true);
matrix.setLed(0,5,2,true);
matrix.setLed(0,6,1,true);

matrix.setLed(0,1,5,true);
matrix.setLed(0,2,5,true);
matrix.setLed(0,2,6,true);

matrix.setLed(0,6,5,true);
matrix.setLed(0,5,5,true);
matrix.setLed(0,5,6,true);
```



Vous pouvez également éteindre toutes les leds avec la commande suivante :

```
matrix.clearDisplay(0);
```

Pourquoi éteindre les leds ? Tout simplement parce qu'au moment du changement d'émoticône, celles-ci risquent de "s'entre afficher". Elles vont apparaître simultanément sur la matrice et vous ne saurez plus si votre plante est contente ou si elle est triste !

Le code final

```
#include<"LedControl.h">

LedControl matrice = LedControl(12, 11, 10, 1);

void setup() {
  matrice.clearDisplay(0);
  matrice.shutdown(0, false);
  Serial.begin(9600);
}

void loop() {
  int humidity = analogRead(0);
  Serial.println(humidity);

  if(humidity > 300){
```

```
    matrice.clearDisplay(0);
    // ================ BAD ================
    matrice.setLed(0,1,1,true);

    matrice.setLed(0,2,2,true);
    matrice.setLed(0,3,2,true);
    matrice.setLed(0,4,2,true);
    matrice.setLed(0,5,2,true);

    matrice.setLed(0,6,1,true);


    matrice.setLed(0,1,5,true);
    matrice.setLed(0,2,5,true);
    matrice.setLed(0,2,6,true);

    matrice.setLed(0,6,5,true);
    matrice.setLed(0,5,5,true);
    matrice.setLed(0,5,6,true);
  } else {
    // ================ SMILE ================
    matrice.clearDisplay(0);
    matrice.setLed(0,1,2,true);

    matrice.setLed(0,2,1,true);
    matrice.setLed(0,3,1,true);
    matrice.setLed(0,4,1,true);
    matrice.setLed(0,5,1,true);

    matrice.setLed(0,6,2,true);


    matrice.setLed(0,1,5,true);
    matrice.setLed(0,2,5,true);
    matrice.setLed(0,2,6,true);
    matrice.setLed(0,1,6,true);

    matrice.setLed(0,6,5,true);
    matrice.setLed(0,6,6,true);
    matrice.setLed(0,5,5,true);
    matrice.setLed(0,5,6,true);
  }
}
```