Arduino

1. Cartes Arduino

Il s'agit de cartes électroniques libres utilisant des composants et des programmes simples à utiliser. Ces cartes peuvent lire des entrées – signal envoyé par un capteur (capteur de température, capteur de position,…), par bluetooth, par wifi. Ces entrées peuvent ensuite être analysées afin de piloter des sorties (commander un voyant, un afficheur, un relais, un moteur…). Le cœur de la carte arduino est un microcontrôleur.

Pour programmer la carte on utilise un environnement de développement (IDE) (basé sur Processing) et un langage (basé sur Wiring).

Différents modèles. – Une trentaine de cartes différentes sont disponibles : (Uno, Nano,…). La carte que vous allez utiliser est la carte Uno « officielle » (environ 20 €). C'est la carte la plus utilisée. Elle est équipée de connecteurs femelles permettant de réaliser et de modifier rapidement un montage.

Documentation.– De nombreux projets mettent en œuvre une carte Arduino et il est très facile de trouver des informations sur Internet. Voici quelques sites que j'ai eu l'occasion de consulter : le site officiel https://www.arduino.cc/ - Un site très intéressant réalisé par des amateurs de modélisme ferroviaire : https://www.locoduino.org/ - Un site qui permet de prendre conscience qu'on peut utiliser un arduino pour des applications nécessitant une grande fiabilité – allumage programmable pour Alpine Renault : http://a110a.free.fr/SPIP172/article.php3?id_article=142.

Remarques :

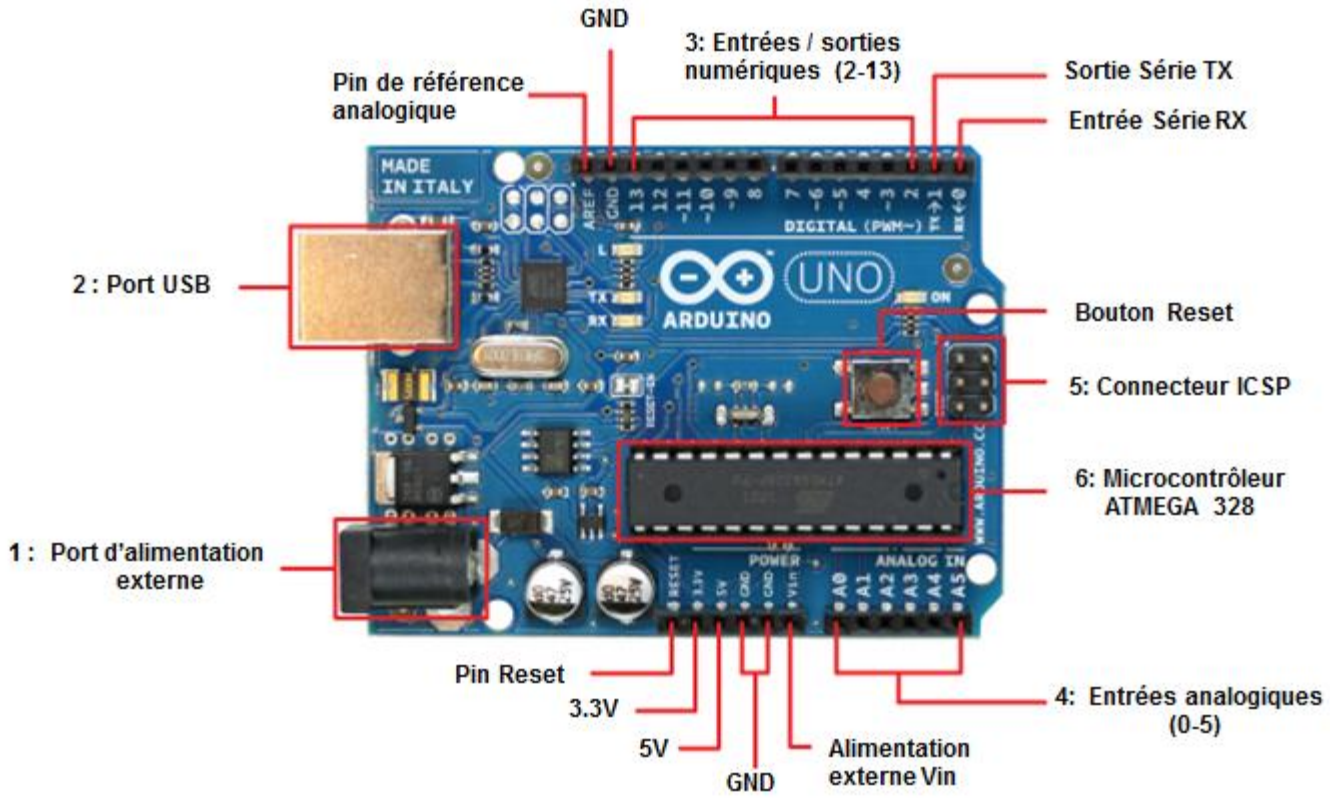
la carte Nano est plus petite et peut être très intéressantes pour des applications où la place est comptée ;

certaines cartes sont livrées sans connecteurs soudés. Cela nécessitent de souder les câbles de liaisons aux entrées et aux sorties, ce qui peut être un gros avantage si l'on recherche la fiabilité ;

on peut se procurer, sur Internet, des cartes « compatibles Arduino » pour environ 5 €.

Arduino Nano

2. Carte Arduino Uno



Alimentation électrique de la carte. – La carte doit être connecté à une source d'alimentation électrique continue pour fonctionner. Il y a 2 possibilité pour alimenter la carte :

Utiliser le port USB (2) relié à l'ordinateur sur lequel on travaille. C'est la solution que l'on utilise dans la phase de prototypage, le port USB permettant l'alimentation de la carte (en 5 V) et le transfert du programme.

Utiliser le port d'alimentation (1). C'est la solution que l'on utilise dans la phase d'utilisation. La tension recommandée est comprise entre 7 et 12 V, les valeurs limites étant 6 et 20 V. On utilise alors une pile 9V, une batterie, un adaptateur 240 V / 9V

Transfert du programme, visualisation des entrées. – Le port USB (2) Permet de communiquer avec la carte (et de l'alimenter).

Entrées/sorties numériques (3). – 14 entrées/sorties numériques dont 6 peuvent assurer une sortie PWM. peuvent actionner de nombreux composants (LED, Afficheurs,...). **Le courant délivré par une sortie doit être inférieur à 20 mA** (40 mA selon certaines sources, avec un courant cumulé sur l'ensemble des sorties inférieur à 200 mA). Pour commander des circuits nécessitant une intensité supérieure, il faut utiliser des transistors ou des relais.

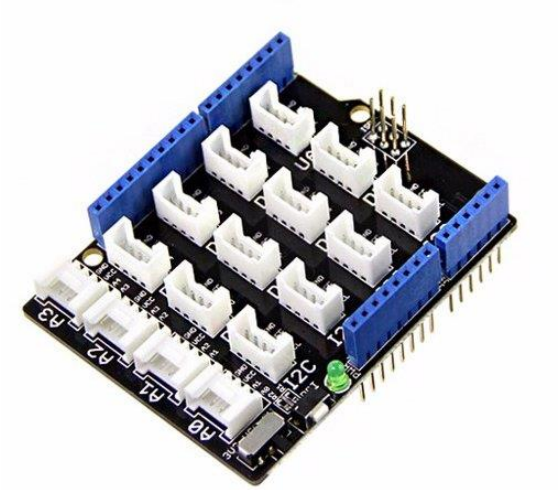
Entrées analogiques (4). – Ces entrées permettent de mesurer une tension variable comprise entre 0 et 5 V. Les capteurs avec sortie analogique (certains capteurs de température, certains capteurs d'humidité, potentiomètre…) seront connectés sur ces entrées.

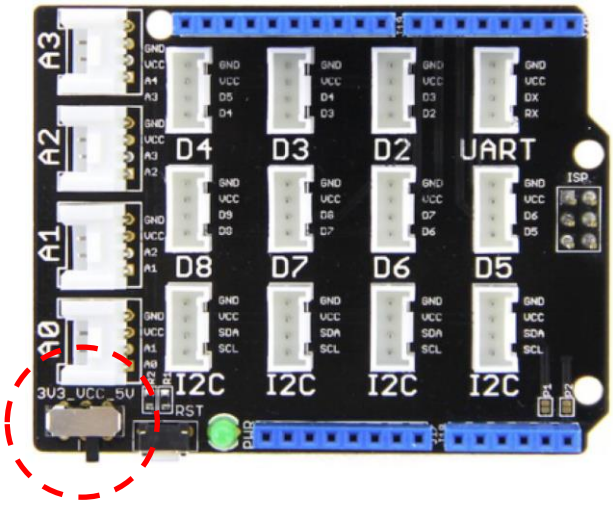
3. Shields

On peut rajouter sur la carte différentes plateformes – Shield – qui permettent de faciliter les connexions (shield de base Grove) ou d'étendre les capacités de la carte (enregistrer des données sur une carte mémoire SD, commander un moteurs,…).

Base Shield V2

Cette plateforme permet une connexion facile de différents éléments muni d'un connecteur **Grove** : capteur, afficheur,…





Attention au petit commutateur permettant d'alimenter les connecteurs en 3,3 V ou en 5V. L'afficheur Grove-LCD RGB Backlight V4.0 ne fonctionne qu'avec une alimentation 5V !!!

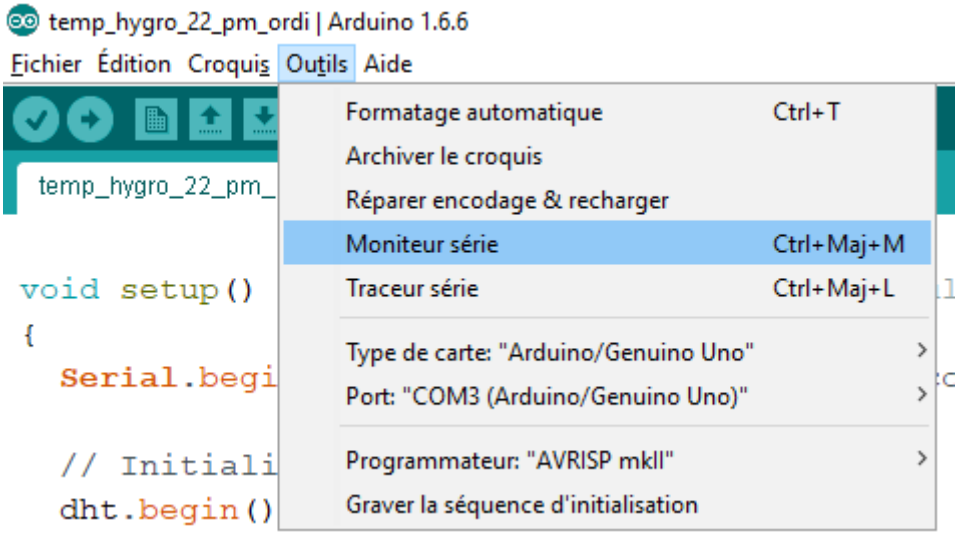
4. Phase de mise au point : utilisation de la sortie sur port série

Il est souvent utile de connaître l'évolution des variables (température issue d'un capteur,…) sans utiliser un afficheur. Le moyen le plus simple est « d'écrire » la ou les valeur(s) sur le port série. On utilise ensuite la fonction « moniteur série » de l'IDE Arduino qui permet de suivre l'évolution des variables.

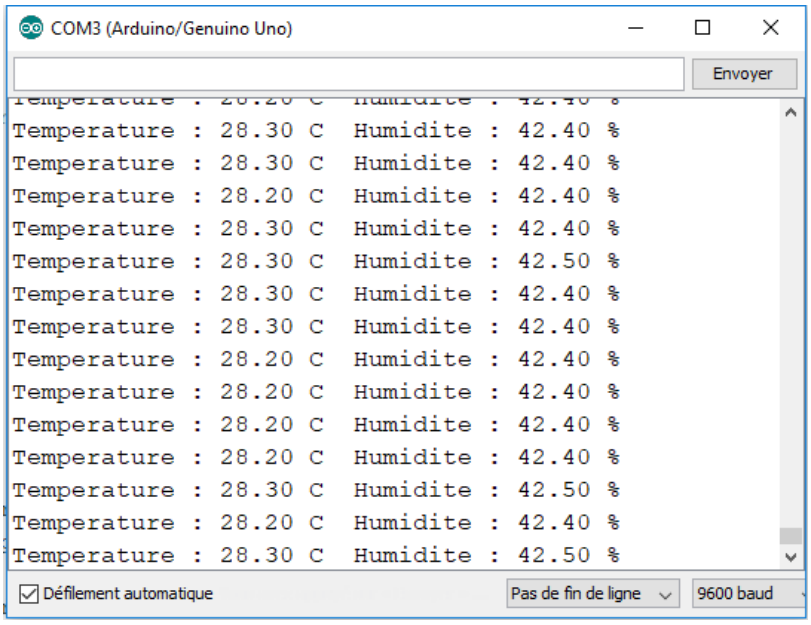
Extrait de code

```
// Transmission des données au port série
Serial.println("");                 // on va à la ligne
Serial.print("Temperature : ");     // on écrit du texte
Serial.print(t,2);                  // on écrit la valeur de t avec 2 chiffres après la virgule
Serial.print(" C ");                // on écrit du texte
```

IDE Arduino : ouvrir le Moniteur Série



IDE Arduino : le Moniteur Série

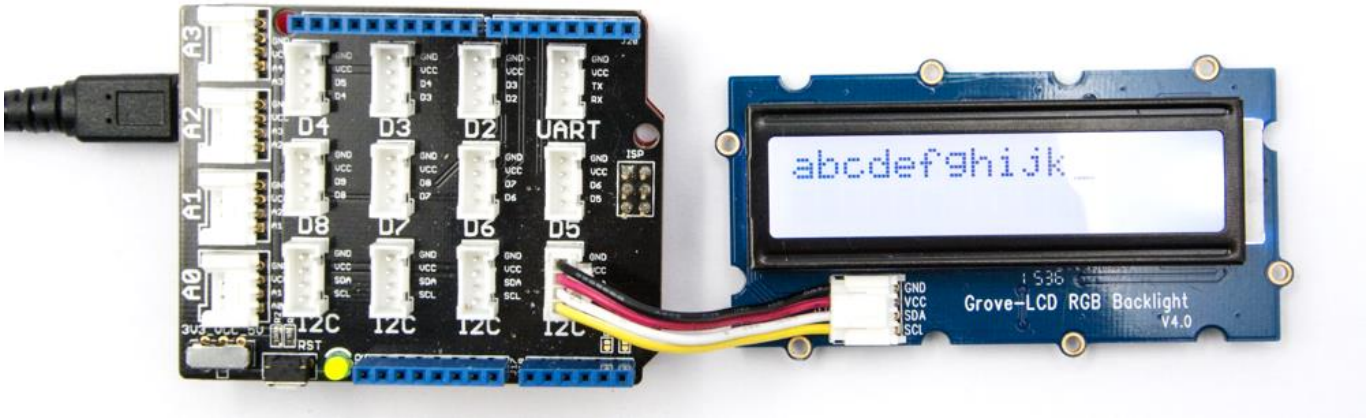


5. Afficheur Grove-LCD RGB Backlight V4.0

Cet afficheur se connecte à l'aide d'un câble Grove. Il nécessite une bibliothèque spécifique : Grove_LCD_RGB_Backlight-master, que l'on peut télécharger sur ce site :

http://wiki.seeedstudio.com/Grove-LCD_RGB_Backlight/

Cet afficheur doit être alimenté (par le cordon Grove) en 5V. Si les caractères ne s'affichent pas, il est fort probable que le commutateur du shield est en position 3,3 V.



6. Quelques exemples :

temp_hygro_22_pm_ordi : utilisation d'un capteur de température et d'humidité DHT 22, affichage des valeur via le port série.

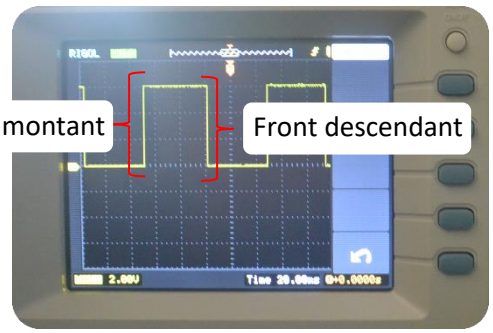
Code minimaliste. Il faudrait rajouter la vérification du bon déroulement de la lecture des données du capteur

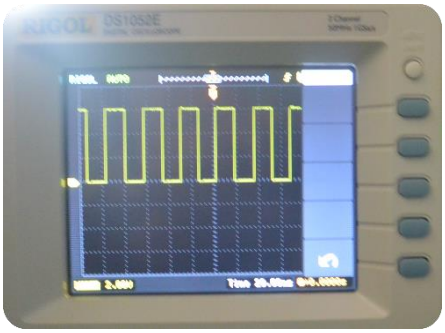
temp_hygro_22_pm_mini : utilisation d'un capteur de température et d'humidité DHT 22 et d'un afficheur

Code minimaliste. Il faudrait rajouter la vérification du bon déroulement de la lecture des données du capteur.

7. Les interruptions

Bien souvent, on lit la valeur d'une (ou plusieurs) entrée(s) régulièrement afin d'agir sur les sorties en fonction des valeurs relevées sur la ou les entrées. Dans certains cas il n'est pas possible de possible de procéder de cette manière, parce que l'entrée à laquelle on s'intéresse change d'état en permanence, c'est le cas notamment lorsque l'on utilise un capteur à effet hall pour mesurer une vitesse de rotation. Celui-ci délivre une tension nulle sauf lorsque la partie aimantée liée à la partie tournante passe devant le capteur. Il délivre alors une tension non nulle.
C'est ce que l'on observe à l'aide d'un oscilloscope :





Le signal de sortie est une succession de créneaux dont la largeur diminue lorsque la vitesse augmente. Il suffit donc de compter le nombre de créneaux par seconde pour déterminer la vitesse de rotation. Pour cela, on peut soit s'intéresser au moment où le signal passe de 0 V à 5V (dans le cas de l'exemple) – on parle alors de détection d'un front montant, soit s'intéresser au moment où le signal passe de 5V à 0V – on parle alors de détection d'un front descendant. Plus généralement on parlera d'état bas (LOW) lorsque le signal a une tension nulle et d'état haut (HIGH) lorsque le signal à une tension non nulle.

Cette ligne : attachInterrupt(0, demo_bp, RISING); indique que l'on souhaite détecter un front montant. Celle-ci : attachInterrupt(0, demo_bp, FALLING); indique que l'on souhaite détecter un front descendant.

Fichier exemple : Interruption. Écrit pour être utilisé avec un bouton poussoir (avec une résistance pour assurer l'état bas). Cet exemple permet de comprendre l'interruption sur la détection d'un front montant ou d'un front descendant. Il permet également de constater le problème de « rebond » d'un bouton poussoir…

Webographie : https://www.locoduino.org/spip.php?article64