Le module Arduino

L'objectif de ce chapitre est de connaître le module Arduino.

1 Le module Arduino c'est quoi ?

Le module (ou la carte) Arduino est une petite carte électronique équipée d'un microcontrôleur. Le microcontrôleur permet, à partir d'événements détectés par des capteurs, de commander des actionneurs selon son programme. Elle joue aussi le rôle d'interface entre un ordinateur et des capteurs.

Le programme est écrit sur l'ordinateur puis téléversé dans la mémoire de la carte Arduino. Celle-ci devient à ce moment-là autonome et l'ordinateur n'est plus nécessaire au fonctionnement de la carte. Le programme reste inscrit tant qu'il n'est pas remplacé par un autre.

Le module Arduino est peu couteux (moins de 30€), relativement simple à mettre en œuvre et à programmer. Il est issu d'un projet « open source » : toutes les informations techniques et son logiciel de programmation sont disponibles gratuitement. Il est très utilisé par les artistes, les concepteurs de machines, les laboratoires,...

2 Présentation physique du module

Nous nous intéresserons ici au modèle de carte UNO. C'est le plus répandu.

2.1 Les entrées

Les broches d'entrée permettent d'acquérir les informations provenant des capteurs. Ces informations arrivent sous la forme d'une tension électrique.

2.1.1 Les entrées analogiques

La carte Arduino Uno possède 6 entrées analogiques, reliées à un convertisseur analogique/numérique qui renvoie un code numérique sur 10 bits, soit une valeur comprise entre 0 et 1023.

La pleine échelle est de 5V, c'est à dire que la valeur numérique 0 correspond à 0V et la valeur numérique 1023 correspond à 5V.

Ainsi le pas de quantification est de 5V/1024 soit environ 5 mV.

2.1.2 Les entrées numériques

La carte Arduino Uno possède 14 entrées numériques.

Une tension de 5V sera traduite par le nombre 1, une tension de 0V par le nombre 0.

2.2 Les sorties

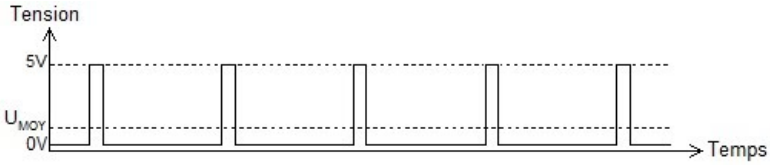
Les broches de sortie permettent d'envoyer les ordres. Si la sortie vaut 1, une tension de 5V sera présente sur sa broche. Si elle vaut 0, la tension sera de 0V. ATTENTION : chaque broche ne peut fournir que 40mA.

2.2.1 Les sorties numériques

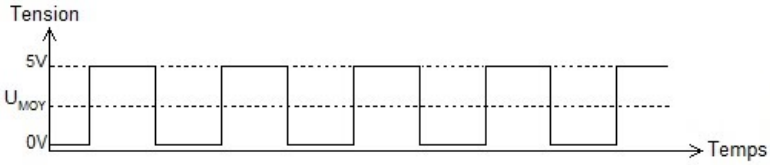
Ce sont les mêmes que les entrées. C'est par la programmation que l'on choisit si la broche est une entrée ou une sortie.

2.2.2 Les sorties MLI

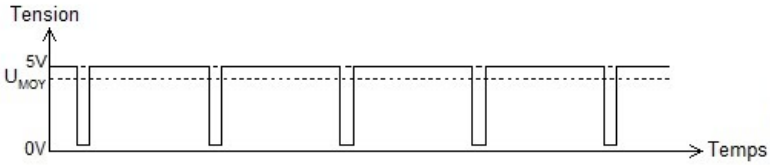
Les broches 3, 5, 6, 9, 10 et 11 peuvent envoyer un signal MLI (Modulation par Largeur d'Impulsion) ou PWM (Pulse Width Modulation) de fréquence variable.



Rapport cyclique proche de 0%



Rappot cyclique de 50%



Rappot cyclique proche de 100%

2.3 Les communications

Le module peut envoyer et recevoir des informations avec d'autres appareils : ordinateurs, circuits intégrés, modules divers,… et d'autres Arduino.

2.3.1 La communication série

Utilise les broches numériques 0(Réception) et 1(Transmission) par l'intermédiaire du port USB. C'est la plus courante.

2.3.2 La communication i2C

Utilise les broches analogiques A4(SDA) et A5(SCL). Utilisée par des capteurs, mémoires,... que l'on peut connecter en chaîne. Chaque élément possède sa propre adresse.

2.3.3 La communication SPI

Même principe que la communication i2C. Utilise les broches numériques 10(/SS), 11(MOSI), 12(MISO) et 13(SCK).

2.4 L'alimentation électrique

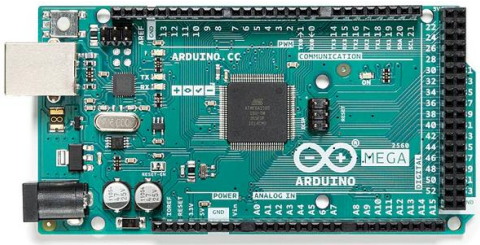
Il y a 4 modes d'alimentation électrique de la carte :

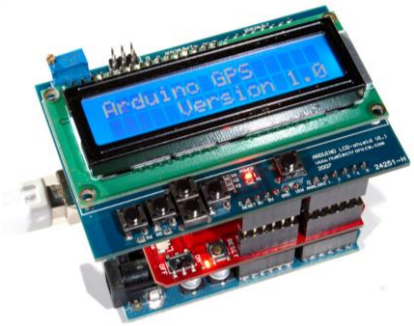
- Une alimentation entre 7V et 12V sur la prise d'alimentation.
- Le câble USB.
- Une alimentation de 5V sur la broche 5V.
- Une alimentation entre 7V et 12V sur la broche Vin.

Quand la carte est correctement alimentée une DEL verte est allumée sur la carte.

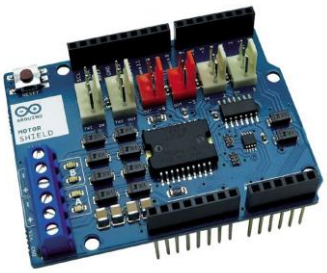
2.5 Les autres cartes

La gamme Arduino comporte de nombreuses autres modèles :





Ainsi que des « shields », cartes additionnelles qui se connectent sur l'Arduino et rajoutent des fonctionnalités



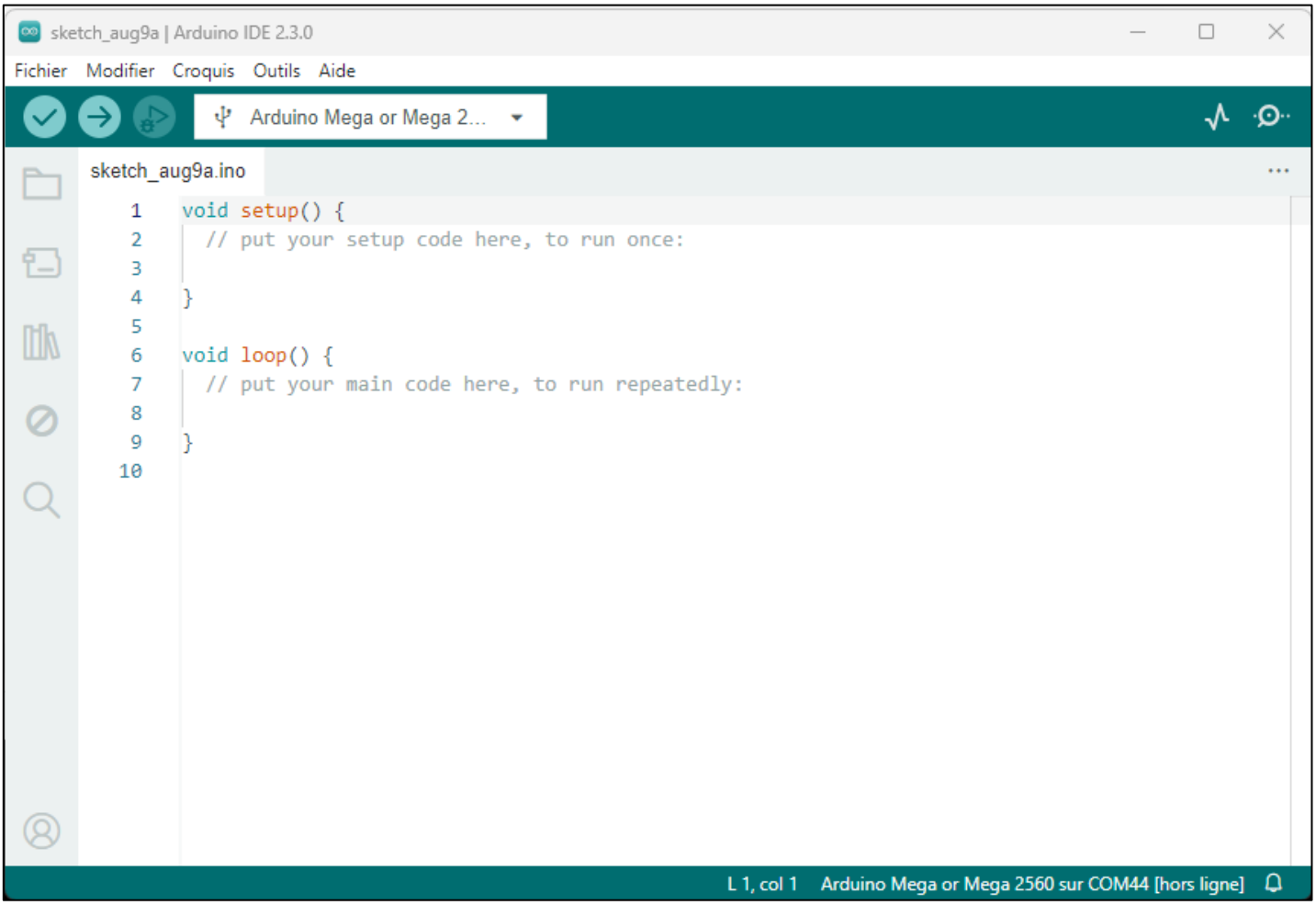
3 La programmation de l'Arduino

La programmation de l'Arduino s'effectue en 3 phases :

1. L'écriture du programme en langage « naturel », le C.
2. La compilation du programme. C'est la traduction du C en instructions compréhensibles par le microcontrôleur.
3. Le téléchargement du programme dans la mémoire de l'Arduino.

Le logiciel nécessaire est l'IDE (Environnement de Développement Intégré) Arduino.

3.1 L'interface de programmation



Exemple de programme :

```
1   #include <HCSR04.h>
2
3   HCSR04 hc(5, 6);
4
5   int distance;
6
7   void setup() {
8     pinMode(7, OUTPUT);
9     Serial.begin(9600);
10  }
11
12  void loop() {
13    distance = hc.dist();
14    Serial.println(distance);
15    delay(60);
16    if (distance < 10) {
17      digitalWrite(7, HIGH);
18    }
19    else {
20      digitalWrite(7, LOW);
21    }
22  }
```

Exécution du programme :

```
1   #include <HCSR04.h>
2
3   HCSR04 sonar(5, 6);
4
5   int distance;
6
7   void setup() {
8     Serial.begin(9600);
9   }
10
11  void loop() {
12    distance = sonar.dist();
13    Serial.println(distance);
14    delay(60);
15  }
```

Les fonctions setup() et loop() sont obligatoires. Il est possible (et souvent souhaitable) de créer ses propres fonctions et de les appeler dans le programme.

Setup() est exécutée une fois au démarrage de la carte (ou lors d'un appui sur reset).

Loop() est exécutée en boucle

3.2 Les bibliothèques

Les bibliothèques (libraries en anglais) contiennent des fonctions permettant de faciliter la programmation et l'utilisation d'un composant (capteur, afficheur,…).

Parfois, Le composant doit être initialisé : broches de connexion, type,…

Exemple :

```
#include <HCSR04.h>
HCSR04 sonar(5, 6);
```

Le radar ultrason est appelé « sonar » et est connecté aux broches 5 et 6.

3.3 Les variables

Il est impératif de définir chaque variable utilisée. On définit son type, son nom et éventuellement sa valeur. La portée d'une variable dépend de l'endroit où elle est déclarée.

```
1 int a = 3; // utilisable dans tout le programme
2
3 void setup() {
4   int b = 5; // utilisable que dans la fonction setup
5 }
6
7 void loop() {
8   int c = 7; // utilisable que dans la fonction loop
9   for( int i = 0; i < 5 ; i++) {  // utilisable que dans le for
10    // ...
11  }
12  while(c < 10) {
13    int d = 9; // utilisable que dans le while
14    c++;
15  }
16 }
```

Les types les plus courants sont :

Type	Contenu	Plage de valeur
boolean	nombre	0 ou 1 (false ou true)
byte	nombre	0 à 255
int	nombre	-32768 à 32767
unsigned int	nombre	0 à 65535
long	nombre	-2^{31} à $2^{31} - 1$
Unsigned long	nombre	0 à $2^{32} - 1$
float	nombre	$-3{,}4.10^{38}$ à $-3{,}4.10^{-38}$ 0 et $3{,}4.10^{-38}$ à $3{,}4.10^{38}$
double	nombre	$-1{,}7.10^{308}$ à $-1{,}7.10^{-308}$ 0 et $1{,}7.10^{-308}$ à $1{,}7.10^{308}$
char	caractère	-127 à 127 (code ascii) ou caractère («A»)
String	Chaîne de caractères	En fait un assemblage de char

3.4 Les instructions de base

3.4.1 L'affectation

Cela consiste à donner une valeur à une variable. La variable doit avoir être déclarée.

```
1136    E = 20;
1137    F = 60;
```

3.4.2 Les opérations mathématiques

+ addition
- soustraction
* multiplication
/ division
% modulo (reste de la division euclidienne)

```
234   tempsactuel = (now.hour() * 10000L) + (now.minute() * 100) + now.second();
235   debutjour = (heure_debut * 10000L) + (minute_debut * 100);
236   finjour = (heure_fin * 10000L) + (minute_fin * 100);
```

3.4.3 Lire une entrée

Lire une entrée analogique

```
22    potentiometre = analogRead(0);
```

Ici lecture de l'entrée 0. Sa valeur est comprise entre 0 et 1023.

Lire une entrée numérique

```
110    mesure_num_9 = digitalRead(9);
111    mesure_num_10 = digitalRead(10);
```

La valeur vaut 0 ou 1.

ATTENTION : La broche aura dû être déclarée comme une entrée dans le setup().

```
107    pinMode(17, INPUT_PULLUP);
```

3.4.4 Activer une sortie

ATTENTION : La broche aura dû être déclarée comme une sortie dans le setup().

```
109    pinMode(23, OUTPUT);
```

Activer une sortie numérique

```
756    digitalWrite(4, HIGH);
```

Ici la sortie 4 est activée. 0 ou LOW pour désactiver, 1 ou HIGH pour activer.

Activer une sortie PWM

```
228    analogWrite(9, 128);
```

La sortie 9 envoie une tension moyenne de 2,5V.

3.4.5 Les tests conditionnels

Une série d'instruction sera exécutée ou non en fonction du résultat d'un test logique.

Le test if :

```
704    if (hygrometrie < hygrometrie_jour - 3) {
705      etat[3] = 1;
706    }
707    else if (hygrometrie > hygrometrie_jour + 3) {
708      etat[3] = 0;
709    }
```

Le test switch :

En fonction de la valeur de la variable « menu », le « case » correspondant sera effectué.

```
260    switch (menu) {
261      case 1:
262        date = (date + 1) % 32;
263        Reglage_temps();
264        break;
265      case 2:
266        mois = (mois + 1) % 13;
267        Reglage_temps();
```

3.4.6 Les boucles

Un bloc d'instruction sera effectué plusieurs fois.

La boucle for Le nombre de tours est connu.

```
24    for (int i = -1; i <= num_led; i++) {
25      ruban_leds.setPixelColor(i, ruban_leds.Color(0, 255, 0));
26    }
```

La variable i (le compteur) varie de -1 à la valeur de num_led en augmentant de 1 à chaque tour. Dès que i est égal à num_led, le programme sort de la boucle.

<u>La boucle while</u> Le nombre de tours dépend d'une variable. ATTENTION : si la condition de sortie n'est jamais réalisée le programme boucle à l'infini et l'Arduino est bloqué jusqu'au prochain reset.

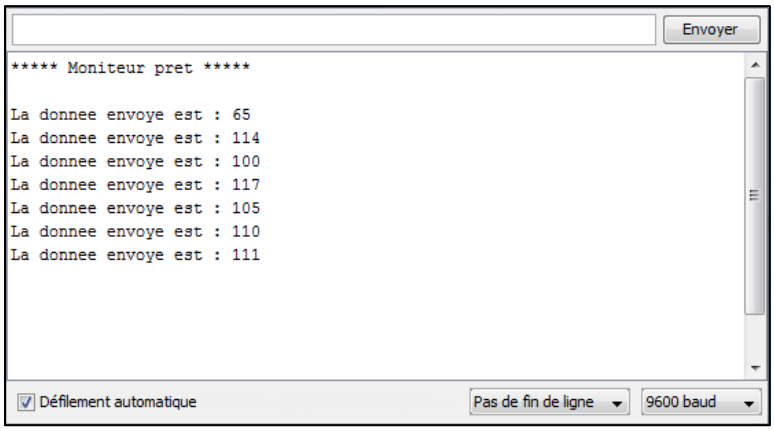
```
53  while (digitalRead(buttonPin) == HIGH) {
54    calibrate();
55  }
```

Tant que l'entrée numérique « buttonPin » est activée, la fonction calibrate est appelée.

3.4.7 La console série

Permet la communication entre l'ordinateur et Arduino. Utile pour afficher des valeurs envoyées par Arduino vers le PC. La vitesse de communication doit être identique entre le programme et la console (ici 9600).

```
1   #include <HCSR04.h>
2
3   HCSR04 sonar(5, 6);
4
5   int distance;
6
7   void setup() {
8     Serial.begin(9600);
9     Serial.println("***** Moniteur pret *****");
10    Serial.println();
11  }
12
13  void loop() {
14    distance = sonar.dist();
15    Serial.print("La donnee envoyee est : ");
16    Serial.println(distance);
17    delay(60);
18  }
```



Envoyer des données à Arduino :

```
45    if (Serial.available() > 0) {
46      delay(10);
47      message = Serial.readString();
48      Serial.flush();
49
50      if (message.indexOf("LED1") >= 0) {
51        digitalWrite(13, HIGH);
52      }
53      if (message.indexOf("LED0") >= 0) {
54        digitalWrite(13, LOW);
55      }
```

Ici l'envoi du code « LED1 » active la sortie 13, « LED0 » la désactive. Ce type de séquence est placée dans la fonction « loop ».

3.4.8 Quelques instructions utiles

delay(1000) ; Met Arduino en pause pendant 1000 millisecondes. La valeur 1000 peut être changée.

Sources et références iconographiques :
Si pas précisé : Étienne LANGLOIS
Cartes Arduino : https://store.arduino.cc/products