LES PREMIERS PAS DE LA PROGRAMMATION AVEC UNE CARTE ARDUINO UNO

Bienvenue dans le monde de la programmation ou la seule limite sera entre tes deux oreilles …

Pour commencer, un tout petit peu de culture générale

Dans les exercices qui vont suivre, nous allons coder en langage de programmation appelé « C ».
C'est un langage très utilisé dans le monde de la programmation, c'est … la base !

Il existe de nombreux langages qui se sont inspirés sur celui –ci, c'est pourquoi on commence souvent par le « C ».

Si ce monde t'intéresse, tu verras qu'il est présent… partout !

Dans ton ordinateur, dans ton smartphone, dans tes jeux vidéo, dans tes applications mobile … et bien d'autres objets ou logiciels !

Présentation de la carte Arduino

Ok, alors la ça va être légèrement indigeste mais pas de panique on n'entre pas dans les détails. (Ton super professeur de techno pourras t'expliquer si tu as une question !)

Arduino est la marque d'une plateforme de prototypage open-source qui permet aux utilisateurs de créer des objets électroniques interactifs à partir de cartes électroniques matériellement libres sur lesquelles se trouve un microcontrôleur.

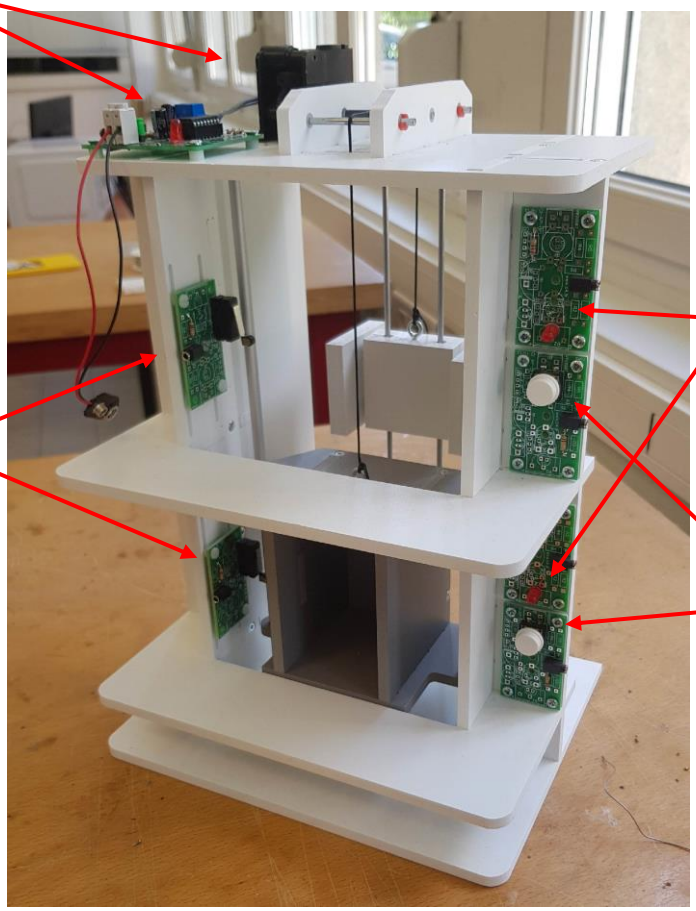
Un microcontrôleur est un circuit intégré qui rassemble les éléments essentiels d'un ordinateur : processeur, mémoires (mémoire morte et mémoire vive), unités périphériques et interfaces d'entrées-sorties.

Sources : Wikipédia

Présentation de la maquette

Le monte-charge

Le Moteur et sa carte d'alimentation

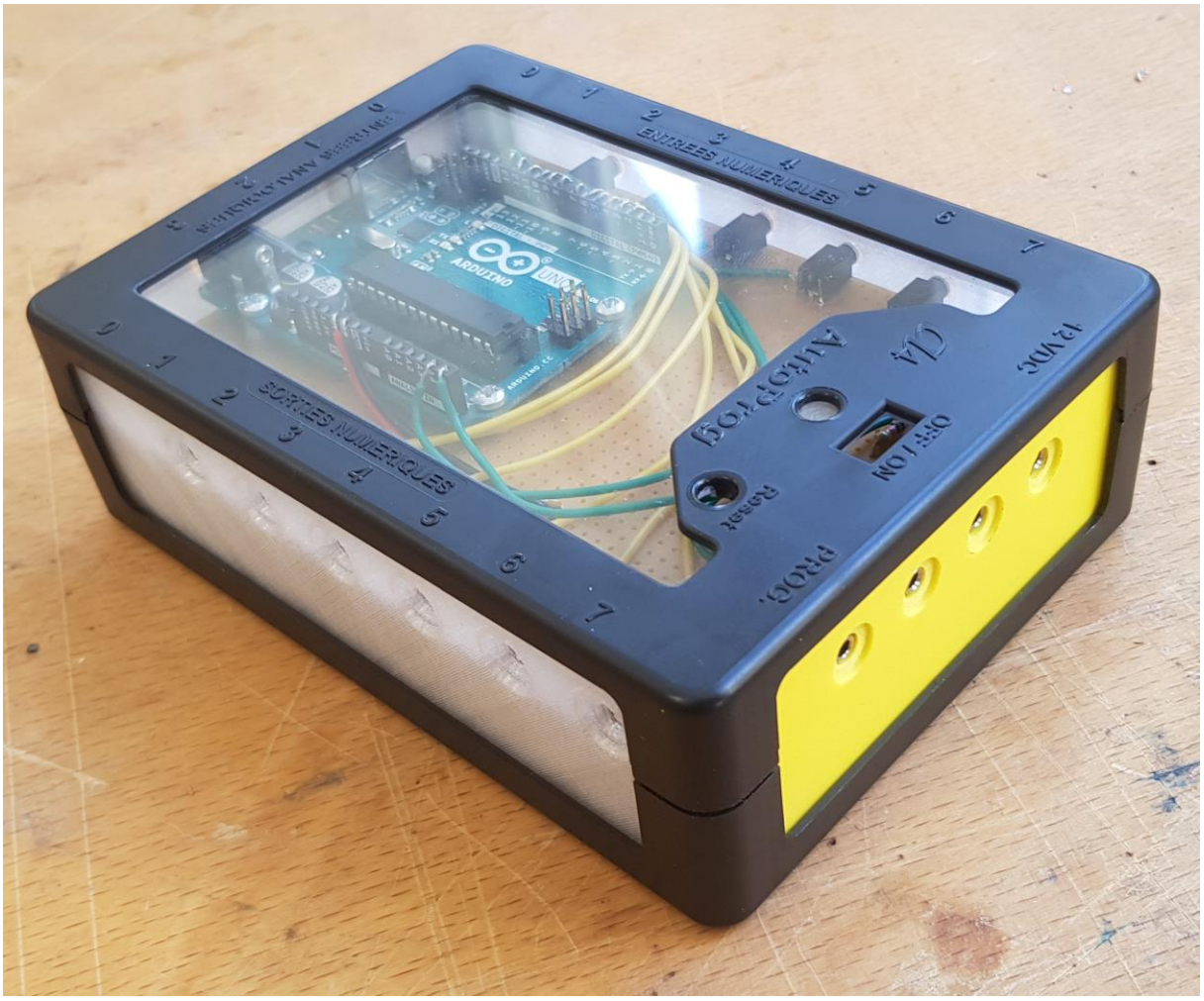


Led d'étage

Fin de course (fdc)

Bouton d'appel

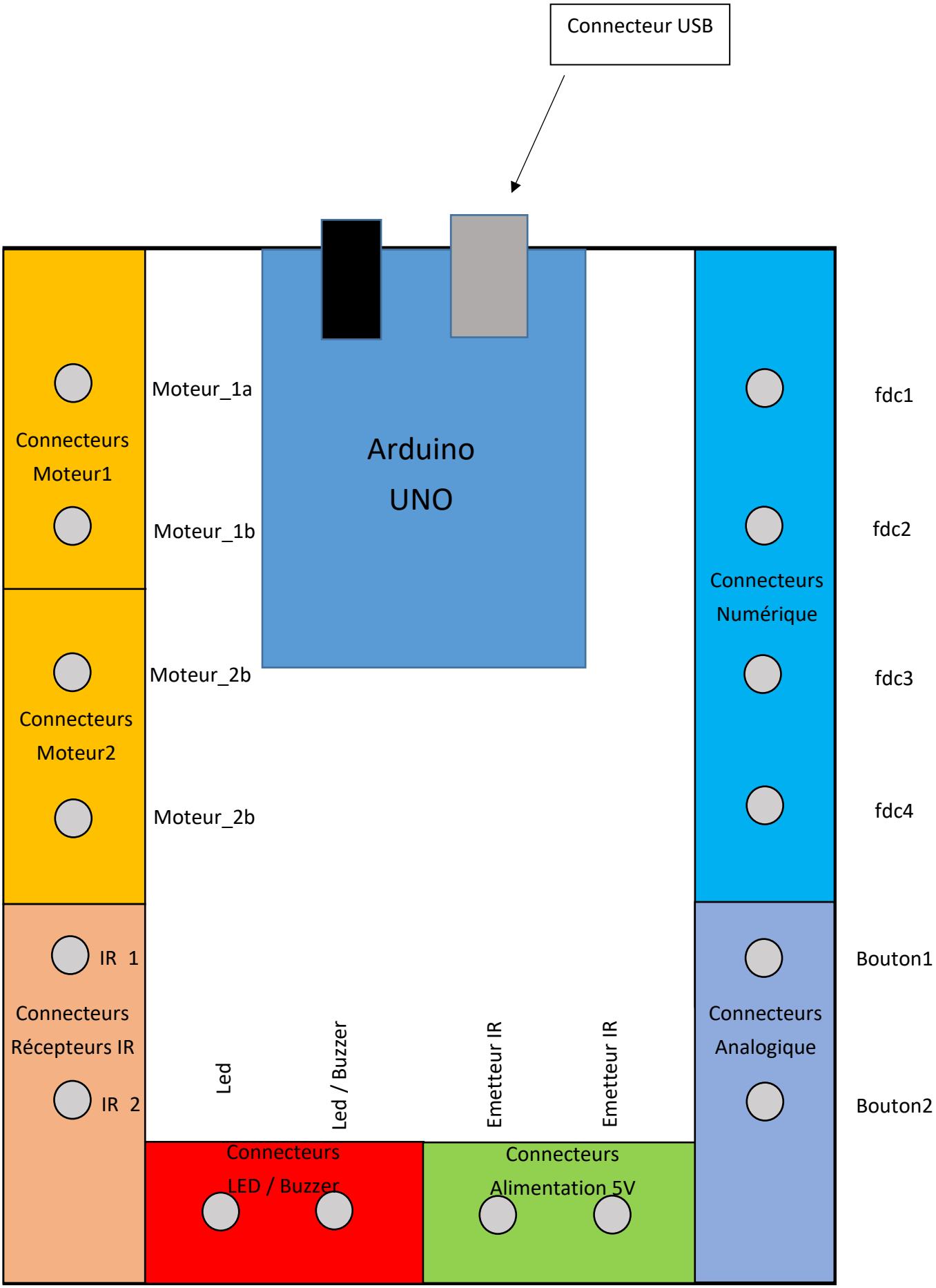
Présentation de l’interface de programmation



Voici, Le cerveau des maquettes ! Grace à lui tu vas pouvoir définir qu’est-ce qui doit fonctionner quand et comment ! La carte Arduino Uno est au milieu, et les connecteurs qui servent à raccorder les différents composants aux maquettes, sont sur les côtés.

C’est un prototype, donc les annotations sur le plastique servent uniquement de décoration … Mais si ce n’est pas déjà fait, des étiquettes vont être mises en place.

Pour te repérer voici un schéma qui pourra bien te servir :



Hop c’est parti on va commencer !

Pour coder, il va te falloir le logiciel Arduino appelé « Arduino IDE » (IDE = Integrated Development Environment ce qui veut dire : l'environnement de développement intégré Arduino). S’il n’est pas déjà sur ton ordinateur, tu vas pouvoir le télécharger sur le site : www.arduino.cc, ce site contient également énormément de ressources si tu recherches des informations, des fonctionnalités, de l’aide.

Une fois le logiciel lancé, et que tu auras écrit tes premières lignes de codes, pour l’essayer, il faudra les « téléverser » vers l’Arduino.

Pour cela tu auras besoin du câble USB type A / type B :

Mais avant cela il faudra aussi câbler les différents composants vers le boitier à l’aide des câbles mini-jack :

Allons-y étape par étape.

Avant de commencer à coder il faut savoir quel fonctionnement nous voulons obtenir.

Exercice 1 :

Pour notre premier programme, nous allons simplement utiliser les deux boutons haut et bas, ainsi que le moteur.

Le fonctionnement : Si on reste appuyé sur le bouton haut, notre monte-charge monte, et si on reste appuyé sur le bouton bas, notre monte-charge descend. Et sinon, il s'arrête. C'est un fonctionnement à pression maintenu.

« Ok, c'est cool tes explications, mais comment on code ça ?! »

Alors, si t'as ouvert le logiciel, t'as dû remarquer qu'il y a déjà des petites choses écrites.

Void setup() : Sert à déclarer à l'Arduino nos différents composants, de déclarer les entrée / sorties. Cette partie sera lue une seule fois au démarrage de l'Arduino ou lors d'un Reset.

Il faut bien comprendre la différence entre une entrée et une sortie.
Une entrée c'est une lecture de données.
Une sortie c'est de l'écriture de données.

Exemple :

Imagine que l'Arduino est un humain, il a donc des oreilles pour écouter, et une bouche pour parler.

Pour faire tourner un moteur, on doit lui dire de tourner, l'Arduino doit lui parler. Donc, c'est un son qui sort de sa bouche … Une information qui sort … C'est une sortie.

Pour savoir si un bouton est enclenché, il doit écouter ce que lui dit le bouton pour savoir s'il est enfoncé.
Il doit donc utiliser ses oreilles, c'est un son qui entre ... Une information qui entre ... C'est une entrée.

Void loop() : Est la partie principale, c'est ici que tu vas mettre ton code. Loop, signifie boucle en anglais.
Tout simplement parce que cette partie est lue, puis relue, puis relue …. En boucle.

Il faut se mettre ça dans la tête, ça va t'aider à comprendre la logique, et la façon de construire ton code.

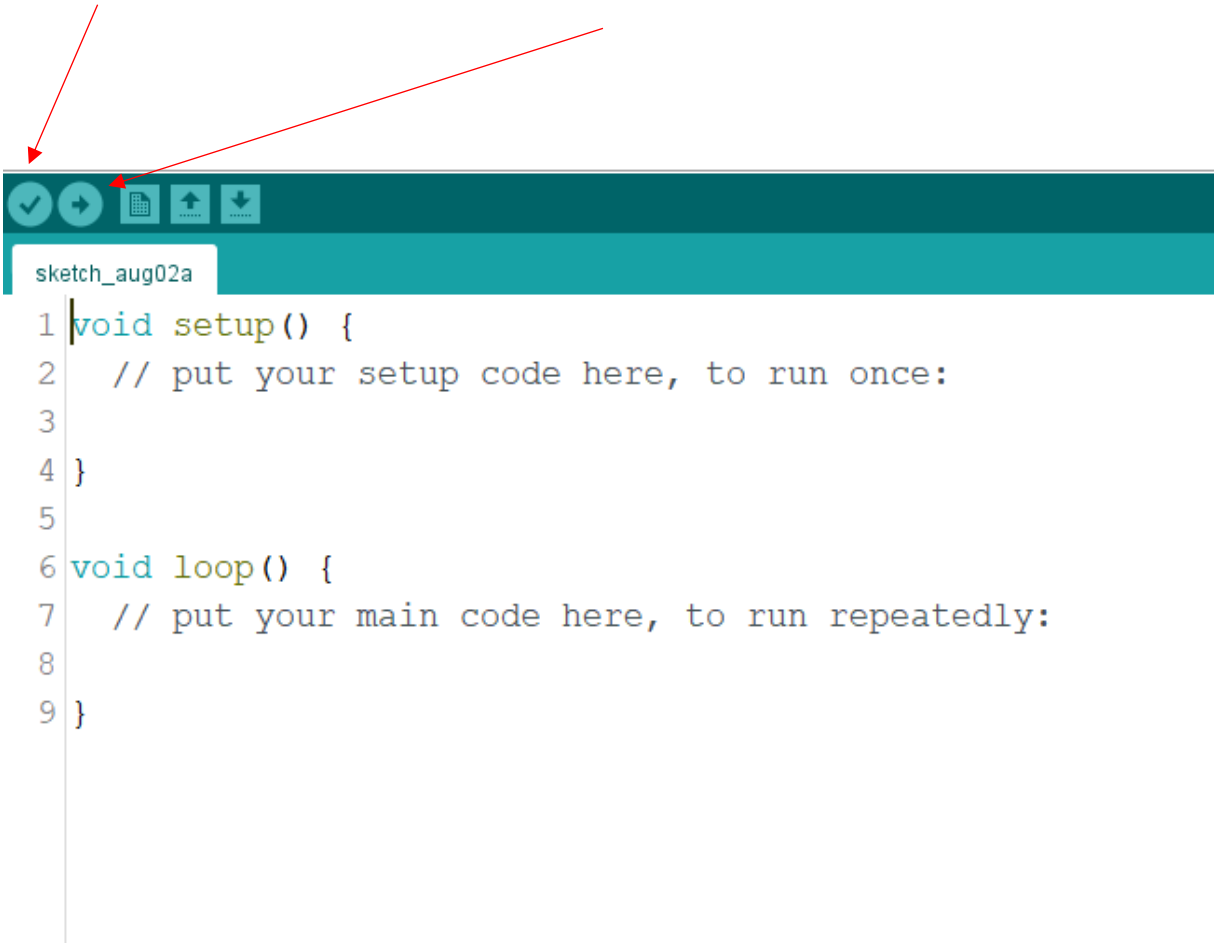
Le logiciel :

Une fois le logiciel ouvert, il faudra importer la bibliothèque avec des fonctionnalités qui t'aideront à coder le monte-charge.
Pour ça il faut aller dans : Croquis >> Inclure une bibliothèque >> Ajouter la bibliothèque .ZIP

Puis, sélectionner la bibliothèque.

Ce bouton sert à vérifier ton code

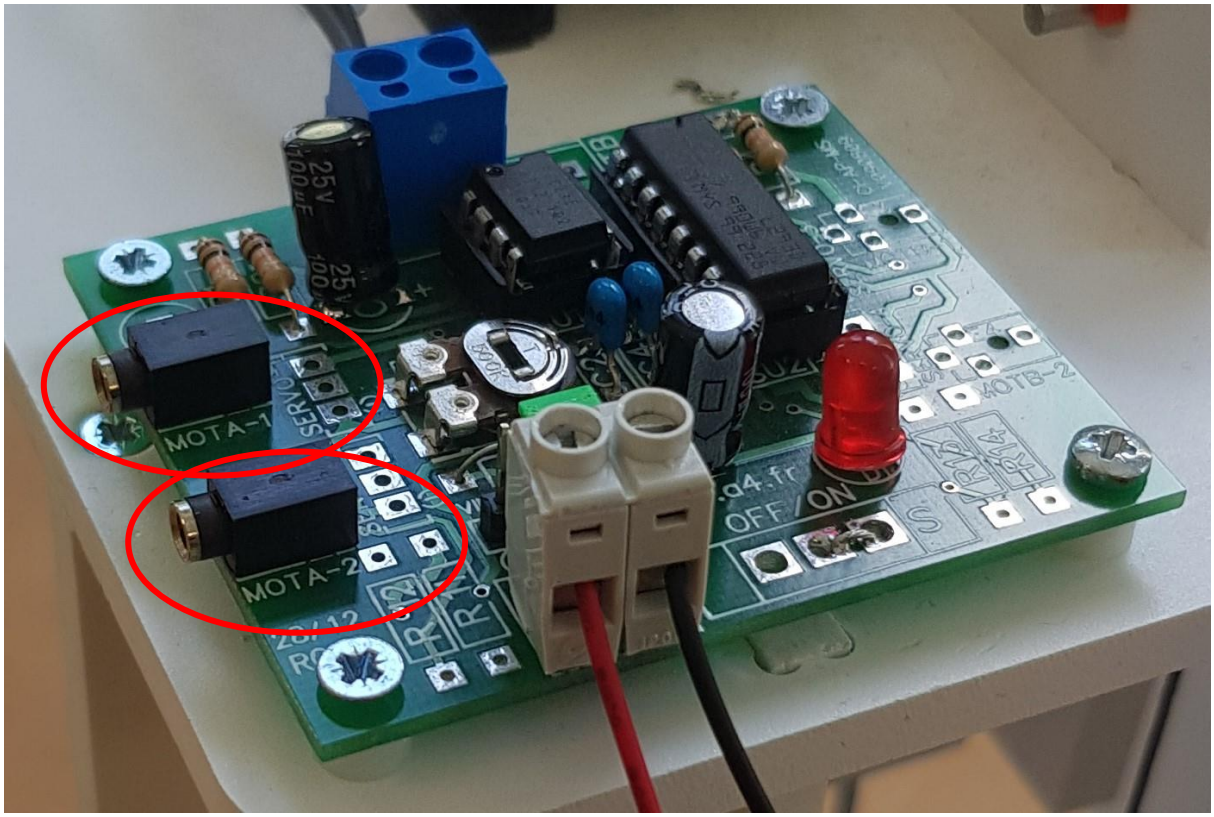
Celui-ci sert à le téléverser



Ok commençons !

Dans un premier temps on va brancher le moteur sur les connecteurs dédiés, en branchant un connecteur sur la sortie « Moteur1a » de notre boitier, vers l'entrée « MOTA-1 » de la carte de notre moteur sur la maquette.

Puis même chose pour le « Moteur1b » et « MOTA-2 ».



Maintenant, on va pouvoir écrire notre première ligne à l'intérieur des accolades du « void setup() » :

Moteur_config(« nom de moteur ») ;

Exemple :

```
1 void setup() {
2 Moteur_config("Moteur1") ;
3
4 }
5
```

N’oublie pas les « ; » sinon tu auras des erreurs !

Cette première ligne sert à dire à l’Arduino ou est branché ton moteur, et de le configurer comme une sortie.

⚠ Pense à bien respecter le schéma vu précédemment pour raccorder les différents composants ⚠

Ensuite, les boutons il va aussi falloir les déclarer dans le « setup ».

Mais avant ça nous allons définir des constantes que l’on va appeler BOUTON1 et BOUTON2.

« Des constantes … ? »
Et oui tant de choses à apprendre et découvrir !!!

Une constante est un identificateur associé à une valeur fixe…
C’est très simple, ça veut dire que si tu décides que ta constante va s’appeler Bob, elle s’appellera Bob et ça ne pourra pas changer tout au long de ton programme. Et une règle de programmation est : qu’il faut toujours écrire les constantes en MAJUSCULE !

Ici la constante qu’on va créer c’est BOUTON1. On va associer BOUTON1 à la sortie A5 de notre carte Arduino. Et ça s’écrit comme ça :

```
const int BOUTON1 = A5 ;
```

C’est la méthode de base, mais qui n’optimise pas ton programme et qui nécessite un minimum de connaissance des différents type de constantes.
Nous allons utiliser la méthode :

```
#define BOUTON1 A5
```
Sur cette ligne pas de point-virgule !!

Qui ne prend pas de mémoire dans ton Arduino et en plus, **beaucoup plus simple à déclarer**.

Puis le setup, ce coup-ci on va utiliser la fonction :

pinMode(« numéro de broche de l’arduino », « Entrée ou sortie ») ;

Les numéros de broches sont dans l’annexe 1.

Comme nous avons défini la constante BOUTON1 = A5

Si on écrit BOUTON1 l’arduino verra A5, donc nous allons pouvoir écrire le pinMode comme suit pour le bouton du haut de notre monte-charge :

```
1 #define BOUTON1 A5
2
3 void setup() {
4 pinMode(BOUTON1,INPUT);
5 }
6
```

Bien évidemment en programmation, tout est en anglais, *(et oui ça sert à quelque chose les cours d'anglais au final …)* donc entrée = INPUT et sortie = OUTPUT. J'ai volontairement écris en majuscule, c'est ainsi que le logiciel le comprend.

Maintenant tu peux faire la même chose pour le bouton 2.

OK ! La config est terminée.
Place au code !

En programmation, pour faire fonctionner notre code en fonction de ceci ou cela on utilise des « conditions ».

Si Le bouton est enfoncé, Alors je fais quelque chose.

En « C » ça donne :

If (le bouton est enfoncé)

{

Je fais quelque chose ;

}

If = Si, en anglais {…} = Alors, en « C »

Aussi, pour lui dire de comparer deux éléments on utilise le double égale « == », le simple égale « = » étant réservé pour les affectation (comme pour notre constante).

Enfin, « HIGH » ça veut dire que quelque chose est activé ou allumé et « LOW » veut dire qu'il est désactivé ou éteint.

Donc dans notre programme nous allons écrire la première condition ensemble.

En utilisant la fonction : Bouton(« Nom_bouton ») ;

If (Bouton("bouton1") == HIGH){

Utilisation de la fonction :

Moteur(« Moteur1 ou Moteur2 », « Montée, Descente ou Arrêt ») ;

}

Else

{

Utilisation de la fonction : Moteur(« Moteur1 », « Arrêt ») ;

}

Else = Sinon, en anglais

Pour ajouter une condition de descente, il faut ajouter un « else if » puis la condition.

Voici le programme pour ce premier exercice :

```
#include "Projet_zero.h"
#define BOUTON1 A5
#define BOUTON2 A4

void setup() {
  pinMode(BOUTON1, INPUT);
  pinMode(BOUTON2, INPUT);
  Moteur_config("Moteur1");
}

void loop() {

  if (Bouton("bouton1") == HIGH)
  {
    Moteur("Moteur1", "Montée");
  }
  else if (Bouton("bouton2") == HIGH)
  {
    Moteur("Moteur1", "Descente");
  }
  else   {
    Moteur("Moteur1", "Arrêt");
  }
}
```

On se connecte à la bibliothèque de fonction préprogrammé

On déclare nos constantes

On définit les entrées/sorties

On conditionne le fonctionnement

Si le bouton1 est activé alors on monte.

Si le bouton2 est activé alors on descend.

Sinon on s'arrête.

Exercice 2

Ok, maintenant que tu as deux trois bases, voyons voir si tu as compris ce qu'on a vu avant.

Fonctionnement : Nous allons ajouter des contacts « fin de course » à notre montage précédent, pour que l'ascenseur s'arrête tout seul quand il est au niveau haut ou bas.

Donc si on ajoute un élément, il faut le déclarer.

Autre petit détail, nos « fin de course » qu'on appellera « fdc » (parce que c'est super long à écrire tout le temps) fonctionnent comme nos boutons.

Pour le fdc_haut on utilisera

#define FDC1 10 *et 11 pour FDC2*.

Puis on fera notre : pinMode(FDC1, entrée ou sortie) ;

Seulement que, comme ils ne sont pas câblés de la même façon, il faudra noter « INPUT_PULLUP » au lieu du « INPUT » classique. !!! UNIQUEMENT POUR LES FDC !!!

Allons-y !

Dans tes conditions pour lire l'état de tes fdc, il faudra utiliser la fonction :

fdc (« fdc1 », « fdc2 », « fdc3 » ou « fdc4 ») ;

Exemple : if (fdc(« fdc1 ») == HIGH) {Moteur(« Moteur1,Arrêt) ;}

Je te laisse compléter ton code cette fois-ci. Ne va pas voir le corrigé si ça ne fonctionne pas ou si tu n'as pas cherché pendant au moins 3 jours … !!! (Temps à ajuster en fonction du prof …).

Exercice 3

Bravo, tu es déjà à l'exercice 3 … on va un petit peu corser les choses avant d'arriver au fonctionnement voulu.

Fonctionnement : Maintenant, si tu appuies sur l'un des deux boutons, le monte-charge monte jusqu'au fdc haut, puis descend jusqu'au fdc bas, puis remonte, puis redescend etc…

Pour cette condition, tu auras besoin **du « OU » logique**.

« OUUUAIS ok… »

Le « OU » logique ça sert à quoi ?
C'est pour encore une fois conditionner notre programme, laisser plusieurs possibilités pour faire quelque chose.

Exemple : SI Bob OU Patrick mange le chocolat ALORS Plancton se vengera.

On peut l'écrire comme ça :

If ((Bob == mange le chocolat) || (Patrick == mange le chocolat)) {Plancton se vengera ;}

Je te laisse méditer là-dessus… Bonne chance !

Exercice 4

Ça y est, nous y sommes le fonctionnement optimal de notre maquette !

Fonctionnement : Pour cette exercice nous voulons que le monte-charge vienne tout seul à l'étage quand on l'appel.
C'est-à-dire, j'appuie sur le bouton du haut puis je relâche (comme pour un vrai ascenseur), et l'ascenseur monte, puis s'arrête à l'étage. Même chose pour l'étage du bas.

C'est beau sur le papier n'est-ce pas ? Mais comment programmer ça ?

Alors, nous allons avoir besoin d'une !!! Variable !!!

Qu'est-ce qu'une variable ? *C'est un ami à constante, seulement que constante sait ce qu'elle veut alors que variable change plus ou moins souvent de décision !*

En gros une variable c'est par exemple un mot genre … « Prénom » et on peut lui attribuer une valeur.

Par exemple : Prénom = « Bob ».

Mais dans le programme si on veut modifier la variable et dire que finalement

Prénom = « Patrick »

On peut le faire, car une **variable** comme son nom l'indique… est **variable**.

Il y a différents types de **variable**, dans notre programme pour le fonctionnement que nous voulons, on aura besoin d'une **variable booléenne**.

Tu ne connais pas le booléen ? Matrix ça te parle ? 01001111001010100111001 10…

Zéro ou un.

C'est une variable qui peut varier entre « **0** » et « **1** ».

Le programme peut aussi le comprendre avec « VRAI » ou « FAUX » … Mais en anglais :

« true » ou « false », ou encore « HIGH » ou « LOW » … *oh ! Mais on a déjà utilisé du booléen pour faire fonctionner notre moteur et conditionner nos boutons !!*

Et oui le booléen est partout !!!!

Comment utiliser un variable ?
Il faut commencer par la déclarer comme notre constante en entête du programme :

```
bool nom_de_ma_variable = 0 ;
```

Puis, on lui donne une valeur de départ, ici j'ai mis « 0 ».

Et voilà, maintenant on peut l'utiliser dans notre programme… Mais comment ? Comment l'utiliser pour que ça fonctionne comme je le souhaite ?!

Je suis sympa je vais te guider 😎.

En fait, ce que tu vas pouvoir faire, c'est créer une variable, pour *mémoriser* l'état de notre bouton haut, par exemple :

```
SI ( Bouton("bouton1") == HIGH) {
 mémoire_bouton1 = 1 ;
}
SI (mémoire_bouton1 == 1){
Moteur(« Moteur1 », « Montée ») ;
}
```

Et voilà, maintenant il faut gérer nos montée, descente et arrêt en prenant en compte cette mémoire des boutons.

Bonne chance 😉 !

Bonus

Si tu es vraiment super fort et que tu aimes aller au bout des choses, ce bonus est fait pour toi !

Nous allons exploiter la maquette jusqu'au bout !
Il reste un élément que nous n'avons pas pris en compte sur la maquette, ce sont les LED d'étage.

Fonctionnement : Ce que nous voulons est simple, presque trop facile pour toi déjà …

SI l'ascenseur est à l'étage, la LED de l'étage s'allume, sinon elle s'éteint.

Pour ce bonus il va falloir configurer tes LED dans le setup en utilisant la fonction :

led_config(« nom_led ») ;

Ici : led_config(« led_haut ») ;

Dans le code, pour allumer et éteindre tes LED, tu pourras utiliser la fonction :

Led(« nom_led », « type de fonctionnement ») ;

Il y a trois fonctionnement possible, « Clignotement », comme son nom l'indique, la LED va clignoter, « Normal », la LED restera allumée fixe, ou bien « Eteint ».

Il y a deux noms de LED : led_haut ou led_bas.

Exemple pour la LED du haut : *Led(« led_haut », « Normal ») ;*

Et on n'oublie pas les points virgule 😁

Annexes

Annexe 1 : Les numéros de broches

Composants de la maquette	Broches de l'Arduino
BOUTON1	A5
BOUTON2	A4
FDC1	10
FDC2	11
FDC3	12
FDC4	13

Les autres composants sont définis automatiquement avec les fonctions que je t'ai créées

Annexe 2 : Le corrigé de l'exercice 2

```
#include "Projet_zero.h"
#define BOUTON1 A5
#define BOUTON2 A4
#define FDC1 10
#define FDC2 11

void setup() {
  pinMode(BOUTON1, INPUT);
  pinMode(BOUTON2, INPUT);
  pinMode(FDC1, INPUT_PULLUP);
  pinMode(FDC2, INPUT_PULLUP);
  Moteur_config("Moteur1");
}

void loop() {

  if ((Bouton("bouton1") == HIGH) && (fdc("fdc1") == LOW))
  {
    Moteur("Moteur1", "Montée");
  }
  else if ((Bouton("bouton2") == HIGH) && (fdc("fdc2") == LOW))
  {
    Moteur("Moteur1", "Descente");
  }
  else   {
    Moteur("Moteur1", "Arrêt");
  }
}
```

Annexe 3 : Le corrigé de l'exercice 3

```
#include "Projet_zero.h"
#define BOUTON1 A5
#define BOUTON2 A4
#define FDC1 10
#define FDC2 11

void setup() {
  pinMode(BOUTON1, INPUT);
  pinMode(BOUTON2, INPUT);
  pinMode(FDC1, INPUT_PULLUP);
  pinMode(FDC2, INPUT_PULLUP);
  Moteur_config("Moteur1");
}

void loop() {

  if ((Bouton("bouton1") == HIGH) || (Bouton("bouton2") == HIGH)
{
  Moteur("Moteur1", "Montée");
    if (fdc("fdc1") == HIGH) {
      Moteur("Moteur1", "Descente");
    }
    else if (fdc("fdc2") == HIGH) {
      Moteur("Moteur1", "Montée");
    }
  }
}
```

Annexe 4 : Le corrigé de l'exercice 4

```
#include "Projet_zero.h"
#define BOUTON1 A5
#define BOUTON2 A4
#define FDC1 10
#define FDC2 11
bool memoire_montee = 0;
bool memoire_descente = 0;

void setup() {
  pinMode(BOUTON1, INPUT);
  pinMode(BOUTON2, INPUT);
  pinMode(FDC1, INPUT_PULLUP);
  pinMode(FDC2, INPUT_PULLUP);
  Moteur_config("Moteur1");
}
```

```
void loop() {
  if (Bouton("bouton1") == HIGH) {
    memoire_descente == 0;
    memoire_montee == 1;
  }
  if (Bouton("bouton2") == HIGH) {
    memoire_montee == 0;
    memoire_descente == 1;
  }
  if (memoire_montee == 1) {
    Moteur("Moteur1", "Montée");
  }
  if (memoire_descente == 1) {
    Moteur("Moteur1", "Descente");
  }

  if (fdc("fdc1") == HIGH) || (fdc("fdc2") == HIGH) {
    memoire_montee == 0;
    memoire_descente == 0;
    Moteur("Moteur1", "Arrêt");
  }
}
```

Annexe 5 : Le corrigé du Bonus

```
#include "Projet_zero.h"
#define BOUTON1 A5
#define BOUTON2 A4
#define FDC1 10
#define FDC2 11
bool memoire_montee = 0;
bool memoire_descente = 0;

void setup() {
  pinMode(BOUTON1, INPUT);
  pinMode(BOUTON2, INPUT);
  pinMode(FDC1, INPUT_PULLUP);
  pinMode(FDC2, INPUT_PULLUP);
  led_config("led_haut");
  led_config("led_bas");
  Moteur_config("Moteur1");
}
```

```
void loop() {
  if (Bouton("bouton1") == HIGH) {
    memoire_descente == 0;
    memoire_montee == 1;
  }
  if (Bouton("bouton2") == HIGH) {
    memoire_montee == 0;
    memoire_descente == 1;
  }
  if (memoire_montee == 1) {
    led("led_bas","Eteint");
    Moteur("Moteur1", "Montée");
  }
  if (memoire_descente == 1) {
    led("led_haut","Eteint");
    Moteur("Moteur1", "Descente");
  }

  if (fdc("fdc1") == HIGH){
    memoire_montee == 0;
    led("led_haut","Normal");
    Moteur("Moteur1", "Arrêt");
  }
  if (fdc("fdc2")== HIGH){
    memoire_descente == 0;
    led("led_bas","Normal");
    Moteur("Moteur1", "Arrêt");
  }
}
```