

Gérer vos fichiers Arduino

! Arduino 1.18 en français n'est pas utilisable !

1 Remarques générales sketch=croquis

Arduino utilise sur l'environnement "processing", mais nous ne ferons pas la différence. Les "sketch" arduinos (croquis) sont des dossiers qui contiennent un fichier .ino et d'autres fichiers utiles ou inutiles, en particulier des .h qui permettent un concept de librairie locale.

On peut être amené à gérer directement les dossiers et leurs contenu en utilisant en particulier Notepad++ ou tout autre éditeur familial.

Si on change le nom du dossier, il faut aussi changer le nom du .ino à l'intérieur. Si au sauvetage la correspondance n'existe pas, Arduino averti et crée un sous-dossier qui ne peut qu'amener des confusions ultérieures.

2. Dossiers pour les exercices

Gérer ses fichiers disques est assez personnel. Les programmes associés à notre cours sont accessibles dans un zip, pour éviter de les retaper. Notre proposition est de créer un dossier "LearnC" et de dézipper nos exercices dans des dossiers lc1 lc2 etc. Les noms à l'intérieur de ces dossiers doivent mettre en évidence la fonctionnalité et la chronologie.

2 Charger un fichier

L'installation d'Arduino a créé un raccourci sur le bureau. En cliquant sur cette icône, Arduino ouvre un sketch vide ou recharge un certain nombres de fichiers précédemment ouverts, selon le fichier preferences.txt que l'on accède via l'onglets "fichier-préférences".

On ouvre un nouveau programme avec "Fichier – Ouvrir";

On le vérifie, on le télécharge si correct avec les icones de gauche.

Si on veut modifier le programme, il faut commencer par le sauver avec "Fichier – Enregistrer sous" et mettre à jour les première lignes qui résumant ce que fait le nouveau programme. En renommant on perd le programme. On peut le reprendre et il ira dans une nouvelle fenêtre.

C'est utile de garder dans l'écran les derniers programmes ouverts. Il peut y en avoir plusieurs sans problèmes.

Attention !

Arduino ne sauve pas les programmes à chaque compilation. Il faut cliquer sur l'icône "Enregistre" ou quitter en confirmant la demande d'enregistrement.

Quant on crée une variante d'un programme qui marche, c'est conseillé de le sauver et donner un nouveau nom au programme que l'on continue à travailler. Ne pas oublier de sauver avant de passer par "Fichier – Enregistrer sous".

Ne pas donner d'extension. Ne pas ouvrir le dossier avec l'ancien nom. Taper le nouveau nom et sauver; un croquis avec à l'intérieur un .ino est créé.

Pour créer un nouveau programme, on procède de façon similaire: on charge un programme qui a des éléments réutilisables (#include, set-up, ..) . On renomme, on enlève ce qui est inutile et on complète en allant prendre des morceaux dans d'autres programmes.

Fichiers importés sous Arduino (semaine 3)

Une bonne pratique de programmation, quand les programmes deviennent longs et que des parties de programmes sont utilisables dans différents programmes, est de mettre les définitions, fonctions, modules dans des fichiers séparés et les appeler avec l'ordre #include.

Le mécanisme est bien connu pour les bibliothèques Arduino, qui sont dans une zone mémoire réservée et connue du compilateur. Nous parlons ici de bibliothèques locales.

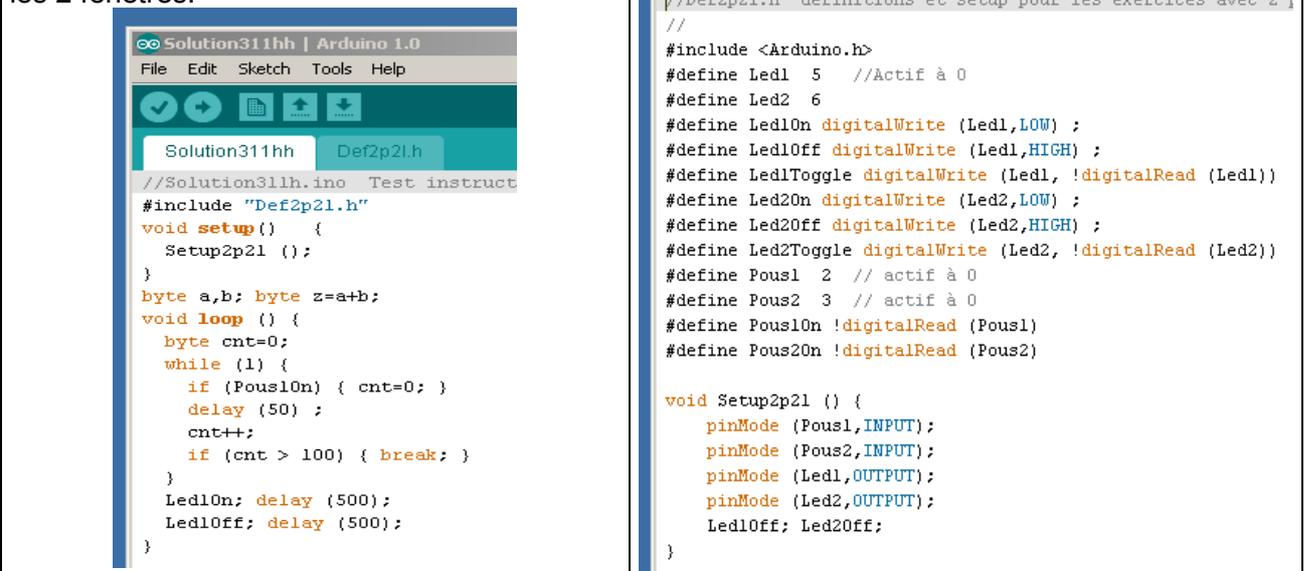
Prenons l'exemple de l'ensemble des définitions pour les exercices du cours Microcontrôleur. On crée un fichier LcDef.h qui les contient, et dans le programme, on note #include "LcDef.h" pour que le préprocesseur ajoute ce fichier avant d'envoyer le programme au compilateur.

Pour le set-up, on préfère définir une fonction qui est appelée dans le setup. De cette façon, on a dans le fichier définit tout ce qui concerne l'aspect matériel de l'application.

On peut parler de bibliothèques propres (elles sont entre "", alors que les bibliothèques communes sont entre <>). La contrainte Arduino est que les fichiers insérés soient dans le même sketch (croquis) que le programme. Le sketch du programme contient le fichier .ino et les fichiers .h associés.

Exemple; chargez Axxxx.ino que vous trouvez dans le dossier Axxxxh le programme solution311h.ino, accompagné par LcDef.h et LcSetup.h

Après chargement on peut visualiser et éditer les 2 fenêtres.



```
//Solution311h.ino Test instruct
#include "Def2p21.h"
void setup() {
  Setup2p21 ();
}
byte a,b; byte z=a+b;
void loop () {
  byte cnt=0;
  while (1) {
    if (Pous10n) { cnt=0; }
    delay (50);
    cnt++;
    if (cnt > 100) { break; }
  }
  Led10n; delay (500);
  Led10ff; delay (500);
}

//Def2p21.h définitions et setup pour les exercices avec 2
//
#include <Arduino.h>
#define Led1 5 //Actif à 0
#define Led2 6
#define Led10n digitalWrite (Led1,LOW) ;
#define Led10ff digitalWrite (Led1,HIGH) ;
#define Led1Toggle digitalWrite (Led1, !digitalRead (Led1))
#define Led20n digitalWrite (Led2,LOW) ;
#define Led20ff digitalWrite (Led2,HIGH) ;
#define Led2Toggle digitalWrite (Led2, !digitalRead (Led2))
#define Pous1 2 // actif à 0
#define Pous2 3 // actif à 0
#define Pous10n !digitalRead (Pous1)
#define Pous20n !digitalRead (Pous2)

void Setup2p21 () {
  pinMode (Pous1,INPUT);
  pinMode (Pous2,INPUT);
  pinMode (Led1,OUTPUT);
  pinMode (Led2,OUTPUT);
  Led10ff; Led20ff;
}
```

La coupure en morceaux d'un programme existant n'est pas évidente. Il faut parfois passer par un éditeur comme NotePad++.

Sous l'onglet "sketch" le menu "add file" permet d'ajouter un fichier, mais apparemment on ne peut pas ajouter un fichier vide que l'on nomme à ce moment. Dans Energia, l'option "New Tab" crée un fichier vide.

Pour des modifications successives de programmes, c'est facile, les fichiers inclus sont automatiquement transportés. Ne pas oublier de sauver à chaque étape.

A noter que si le fichier inclus comporte des fonctions Arduino, il faut ajouter la ligne #include <Arduino.h>

Programmation avec un compilateur C

Le changement est mineur si vous utilisez un compilateur C à la place d'Arduino, qui a défini les deux fonctions void setup() et void loop() alors que le C n'a qu'un point d'entrée. Après les premières instructions de configuration, on a la boucle infinie avec un while (1).

```
#include "LcDef.h"
int main () {
  LcSetup ();
  while (1)
    . . .
}
}
```