Arduino/C include files

C programs make an extensive usage of header files called with a `#include "xxx.h"` or `#include <xxx.h>`.

A programmer-defined header file is a file containing C declarations and macro to be shared between several source files.

The difference between " " and < > header files is clearly explained on
https://stackoverflow.com/questions/21593/what-is-the-difference-between-include-filename-and-include-filename

"For #include "filename" the preprocessor searches in the same directory as the file containing the directive, and then like for #include <filename>. This method is normally used to include programmer-defined header files.
For #include <filename> the preprocessor searches in an implementation dependent manner, normally in search directories pre-designated by the compiler/IDE. This method is normally used to include standard library header files."

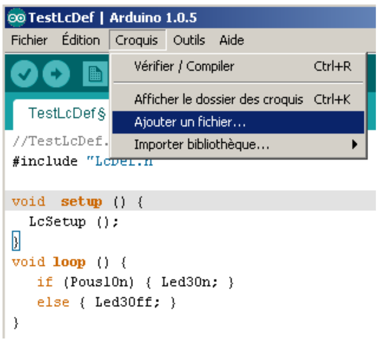
We are concerned here only with " " files, fully under the control of the programmer. Well handled, they play the role of a "Meccano" to build programs. Arduino libraires are rigid, you have to get familiar with their names, you cannot modify without a good C++ knowledge.
".h" libs encapsulate parts of program of any size. Up to you to be consistent.
For instance, you have two leds that must blink. Build a module that defines the connections, the setup, the functions. When you re-use it in another program, modify the wiring definitions and possibly some timings. This module looks like this:

```
//TwoLeds.h
#define Led1 8   // Arduino pin 8
#define Led2 9
#define Led1On digitalWrite (Led1,0)   // 0v for on
. . .
void SetupTwoLeds () {    pinMode (Led1,OUTPUT);
   pinMode (Led1,OUTPUT);
}
void BlinkLed1 (byte howmany,byte period) {
   . . .
}
```

The main program will refer to that module adding 2 lines:

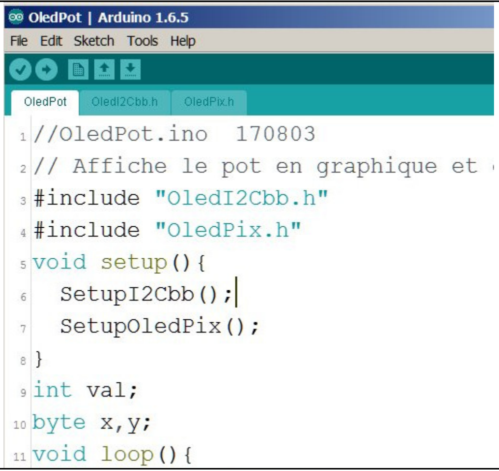
```
#include "TwoLeds.h"
. . .          // other includes
void setup() {
   SetupTwoLeds();
   . . .      // other Setups
}
```

Arduino IDE show the include files as tabs one can
dit. Compilation take care of the files on screen buffer.
o not forget to save, or set-up the preference on
automatic save".


TestLcDef | Arduino 1.0.5
Fichier Édition Croquis Outils Aide
Vérifier / Compiler Ctrl+R
Afficher le dossier des croquis Ctrl+K
Ajouter un fichier...
Importer bibliothèque...
TestLcDef §

```
//TestLcDef.
#include "LcDef.h

void  setup () {
  LcSetup ();
}
void loop () {
   if (Pous1On) { Led3On; }
   else { Led3Off; }
}
```



```
//OledPot.ino  170803
// Affiche le pot en graphique et
#include "OledI2Cbb.h"
#include "OledPix.h"
void setup(){
  SetupI2Cbb();
  SetupOledPix();
}
int val;
byte x,y;
void loop(){
```

The objective is not to write a big application, but do rather small projects, testing sensors, using them with motors for creative projects.

2 places for the ".h" files

C is a little bit tricky and there are too much freedom in the implementations. We report here of our experience with Arduino.

Understand ".h" files can be within the same sketch (croquis) or at the place reserved by the Arduino IDE for libraries. As documented at many places:

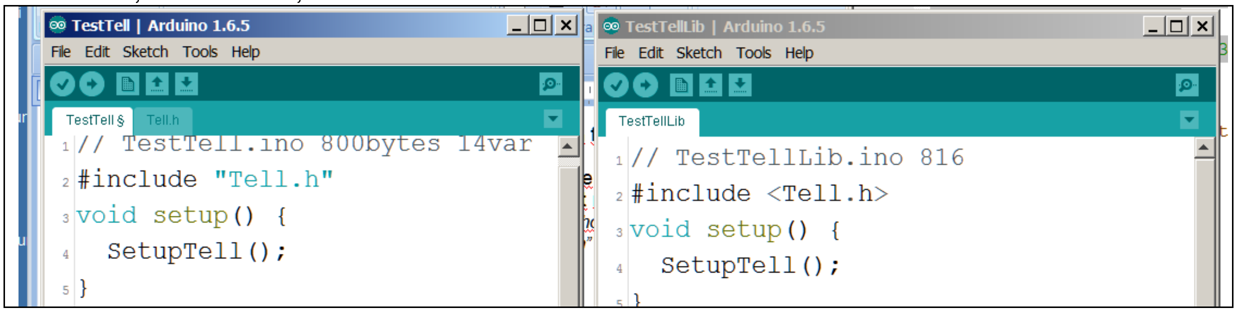
"Open your Arduino IDE, and from the menu, choose Sketch / Include library / Add file. A file requester will open. Navigate to your file, and click the “Open” button. The library will be installed and ready to use."

Update with the same procedure.

You can find the exact place in the "verbose" preference mode. I see for instance in the beginning of that long procedure: "Using library Tell in folder: C:\Users\jdn\Documents\Arduino\libraries\Tell (legacy)" But W7 Explorer do not accept the link!

Imagine we have the TestTell.ino programm the call a function declared in the Tell.h file

1) if `#include "Tell.h"` is not at the beginning of the program, there will be error when referring to Tell.h definitions and functions.
2) If `#include "Tell.h" is` at the beginning of the program but there are no Tell.h file either on the sketch or on the Arduino library, there will be the same errors.
3) If the Tell.h is on one of these two places, ok
4) If the Tell.h is on both places (not recommended), selection ".h" will select locally, <.h> in the Arduino lib
5) Last strange case, selection with ".h" but .h file is only inside Arduino lib. The compiler search on the sketch, if not found, it takes on the lib if it does exist.


TestTell | Arduino 1.6.5
File Edit Sketch Tools Help
TestTell §
Tell.h
TestTellLib | Arduino 1.6.5
File Edit Sketch Tools Help
TestTellLib

```
// TestTell.ino 800bytes 14var
#include "Tell.h"
void setup() {
  SetupTell();
}
```

```
// TestTellLib.ino 816
#include <Tell.h>
void setup() {
  SetupTell();
}
```

One have to live with this. When playing with programs over the years, it is easy to get lost. The advantage of programmer-defined header file is their flexibility. The danger is multiple variants of a lib under the same name. A recommendation is to well define a naming convention. For the Xlib, all lib files start with an x, e.g. xPfm2a.h one of the PFM files, well documented and saved in a common directory before being put on the Arduino lib. For a new program that will need modifications of the xPfm2a.h, due to Arduino requirement to have the same sketch and .ino name, and the stupidity that an include file cannot be renamed or deleted under Arduino,

there are several options

1) Take a program using the lib, replace the < > with " " and under Sketch, load the lib from the "AllLibs" directory. What is modified and saved is local. Then either you update the Lib (if no effect on existing programs) or you vreate a new lib with a new name.
2) Get an empty sketch, save it and add the include files you plan to re-use. Add "Empty" files to add and put a beginning of information inside. Quit, change the names under Explorer, bo back to Arduino, check the names of #include files and setup files. Edit with Notepad when easier. If you are French, do not use accent letters and be coherent with upper and lower cases.

If you understand the logic behind, it is easy. Of course, it is more complicated than using an Arduino lib. It is like for transportations. Either you relax to a place somebody has decided it is a good place for you, or you go where you want and live tiring and enriching experiences.

?????jdn170915 190328

When you develop, file.h is next to the .ino program inside the sketch.

When file.h is well documented, you can put it in the reserved place that may depends on the Arduino version. The Arduino procedure to deposit a lib is valid définitions, fonctions, modules dans des fichiers séparés et les appeler avec l'ordre #include.

The mechanism is well known for Arduino libraries, which are in a reserved memory area and known to the compiler.
Take the example of the set of definitions for the exercises of the Microcontroller course. We create a file LcDef.h which contains them, and in the program, we note #include "LcDef.h" so that the preprocessor adds this file before sending the program to the compiler.
For the set-up, we prefer to define a function that is called in the setup. Any necessary initializations, such as Serial.begin (9600), are added to the setup;
In this way, we have in the file LcDef.h all that concerns the material aspect of the application.
One can speak of own libraries (they are between "" whereas the common libraries are between <>). The constraint for clean libraries is that the inserted files must be in the same folder (sketch, sketch) as the program. The sketch of the program contains the .ino file and associated .h files.
In the index "Sketch" we can fetch files to add in other folders, but we can not remove them, we must intervene with the file manager.

One can naturally make imported files with functions. The constraint is that the variables used must be defined in a previously called module.

Cutting into pieces of an existing program is not obvious. Sometimes you have to go through an editor like NotePad ++.
Under the "sketch" tab, the "add file" menu allows you to add a file, but apparently you can not add an empty file that is named at this time. In Energia, the "New Tab" option creates an empty file.
For successive program changes, it's easy, the included files are automatically transported. Do not forget to save at every step.
Note that if the included file has registry names or Arduino functions, you must add the line #include <Arduino.h>.

Programmation sous C
Le changement est mineur si vous utilisez un compilateur C à la place d'Arduino.
	`#include "LcDef.h"`
	`int  main () {`
	`   LcSetup ();`
	`   while (1) {   // remplace void loop () {`
	`     . . .`
	`   }`
	`}`

jdn131114/140508