Configuration de Netbeans pour ESP32

Table des matières

Pour exploiter pleinement la puissance des cartes à base d'ESP8266 ou ESP32, L'IDE arduino est limité. La complétion de code, l'exploration du code en suivant un symbole, le refactoring, ... deviennent indispensables pour développer un projet un tant soit peu ambitieux.

1 Installation de Esp8266-Arduino-Makefile:

Tout d'abord installez le projet https://github.com/thunderace/Esp8266-Arduino-Makefile comme indiqué dans le README.md du dépôt.

```
git clone https://github.com/thunderace/Esp8266-Arduino-Makefile.git
```

Se déplacer dans le répertoire `Esp8266-Arduino-Makefile`
Lancer le script bash esp32-install.sh

```
~/Esp8266-Arduino-Makefile$ ./esp32-install.sh
```

Installer les dépôts suivants

```
sudo apt-get install libconfig-yaml-perl unzip sed
```

1-2 Installation de pySerial

Pour vérifier si pySerial est installé, ouvrir un shell Python

```
root@b113profs:# python3
Python 3.6.9 (default, Jul 17 2020, 12:50:27)
[GCC 8.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import serial
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ImportError: No module named serial
```

Nous devons au préalable installer **python3-pip** puis **pyserial**

```
root@b106tu4p4:~# apt install python3-pip
root@b106tu4p4:~# pip3 --version
pip 20.0.2 from /usr/lib/python3/dist-packages/pip (python 3.8)
root@b106tu4p4:~# pip3 install pyserial --upgrade
```

2 Modification du shebang des scripts Python

Le **shebang**, représenté par **#!**, est un en-tête d'un fichier texte qui indique au système d'exploitation linux que ce fichier n'est pas un fichier binaire mais un script (ensemble de commandes).

Sur la même ligne que le shebang est précisé l'interpréteur permettant d'exécuter ce script. Ici nous mettrons python3 sauf pour get.py. Modifier les fichiers suivants dans le répertoire tools:

/opt/Esp8266-Arduino-Makefile/esp32-1.0.4/tools# **ls *.py**

espota.py → #!/usr/bin/env python3

esptool.py → #!/usr/bin/env python3

gen_esp32part.py → #!/usr/bin/env python3

get.py → #!/usr/bin/env python2

Le script **get.py** téléchargera et extraira les outils requis dans le répertoire actuel. La liste des outils est obtenue à partir du fichier package /package_esp8266com_index.template.json.

3 Test de Esp8266-Arduino-Makefile

Pour tester le make nous allons compiler le programme **SimpleWiFIServer** donné dans les exemples du dépôt.

Se placer dans le répertoire des exemples esp32/SimpleWiFiServer

Modifiez le fichier makeFile pour l'adapter au modèle de votre carte, au port série utilisé.
(Pour les cartes ESP12E, le paramètre ARDUINO_VARIANT est nodemcuv2)

```
~/Esp8266-Arduino-Makefile/example/esp32/SimpleWiFiServer$ nano Makefile
ARDUINO_VARIANT = lolin32
SERIAL_PORT = /dev/ttyUSB0
# uncomment and set the right serail baud according to your sketch (default to
115200)
#SERIAL_BAUD = 115200
# uncomment this to use the 1M SPIFFS mapping
#SPIFFS_SIZE = 1
```

```
#ESP8266_VERSION=git
# uncomment to exclude this lib from dependencies
#EXCLUDE_USER_LIBS=ESP8266TrueRandom
ARDUINO_ARCH=esp32
USER_DEFINE = -D_SSID_=\"YourSSID\" -D_WIFI_PASSWORD_=\"YourPassword\"
OTA_IP = 192.168.1.184
OTA_PORT = 8266
OTA_AUTH = password
include ../../../espXArduino.mk
```

et lancer make

```
~/Esp8266-Arduino-Makefile/example/esp32/SimpleWiFiServer$ make
```

On obtient le résultat suivant :

```
/home/USERS/PROFS/psimier/Esp8266-Arduino-Makefile/esp32-1.0.4/tools/gen_esp32part.py -q
/home/USERS/PROFS/psimier/Esp8266-Arduino-Makefile/esp32-1.0.4/tools/partitions/
default.csv ./build.lolin32-1.0.4/SimpleWiFiServer.partitions.bin

/home/USERS/PROFS/psimier/Esp8266-Arduino-Makefile/esp32-1.0.4/tools/esptool.py --chip
esp32 elf2image --flash_mode dio --flash_freq 40m --flash_size 4MB -o ./build.lolin32-
1.0.4/SimpleWiFiServer.bin ./build.lolin32-1.0.4/SimpleWiFiServer.elf

esptool.py v2.8-dev


Memory usage
  Ram:    38856 bytes
  Flash: 646206 bytes
```

charger le programme sur la carte avec la commande make upload

```
~/Esp8266-Arduino-Makefile/example/esp32/SimpleWiFiServer$ make upload
```

Vous devez obtenir l'écran suivant

```
esptool.py v2.8-dev

Serial port /dev/ttyUSB0

Connecting........____

Chip is ESP32D0WDQ6 (revision 1)

Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None

Crystal is 40MHz...

Hash of data verified.

Compressed 647344 bytes to 386748...

Wrote 647344 bytes (386748 compressed) at 0x00010000 in 34.2 seconds (effective 151.3
kbit/s)...
```

```
Hash of data verified.

Compressed 3072 bytes to 128...

Wrote 3072 bytes (128 compressed) at 0x00008000 in 0.0 seconds (effective 1551.5
kbit/s)...

Hash of data verified.


Leaving...

Hard resetting via RTS pin...
```

Vérifiez le fonctionnement du programme téléchargé en utilisant minicom

```
$ minicom -s
```

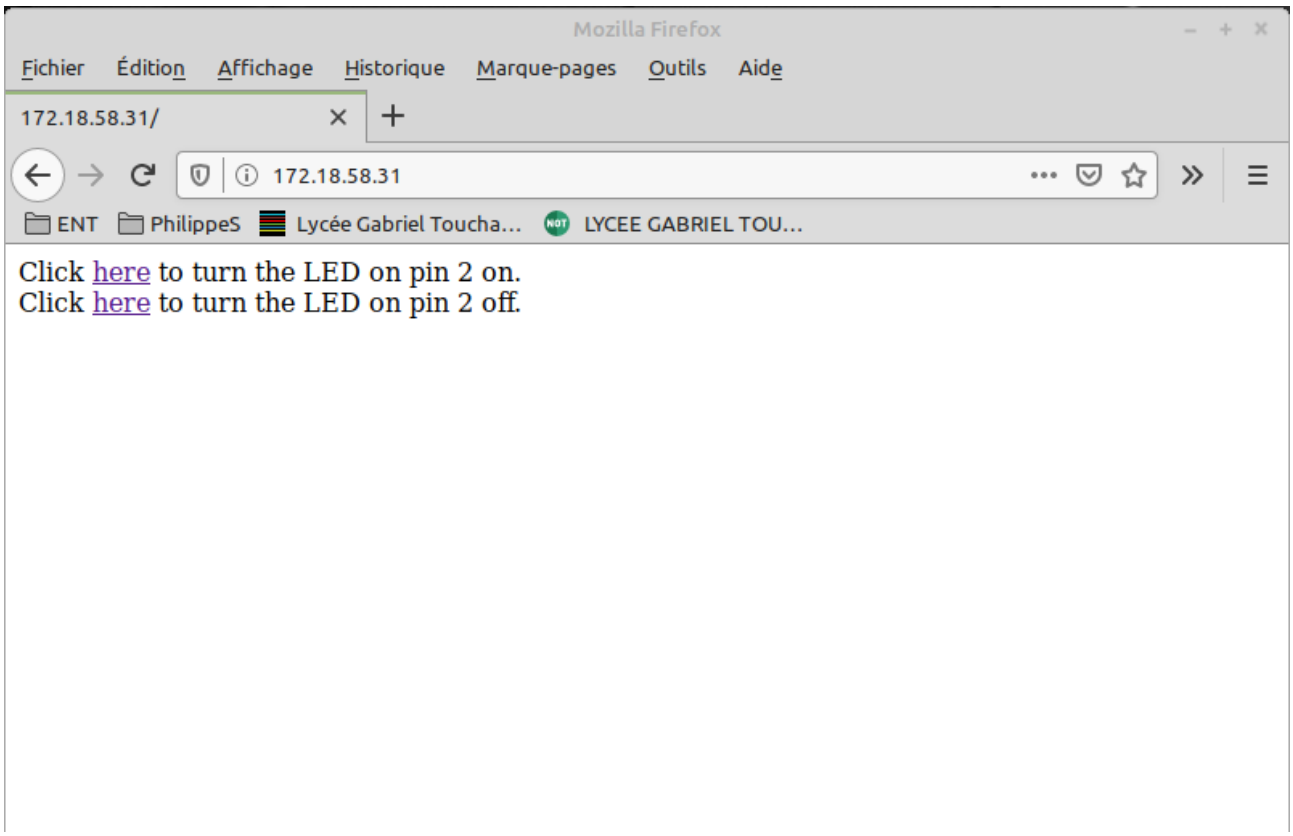
Configurer le port série */dev/ttyUSB0* et 115200 bauds

Appuyer sur le bouton poussoir EN

Le message suivant s'affiche sur la console :

```
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)

configsip: 0, SPIWP:0xee

clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00

mode:DIO, clock div:2

load:0x3fff0018,len:4

load:0x3fff001c,len:1044

load:0x40078000,len:8896

load:0x40080400,len:5828

entry 0x400806ac



Connecting to SNIR03

..

WiFi connected.

IP address:

172.18.58.31

Local Start

Global Start
```

Ouvrir un navigateur web pour obtenir la page web suivante :

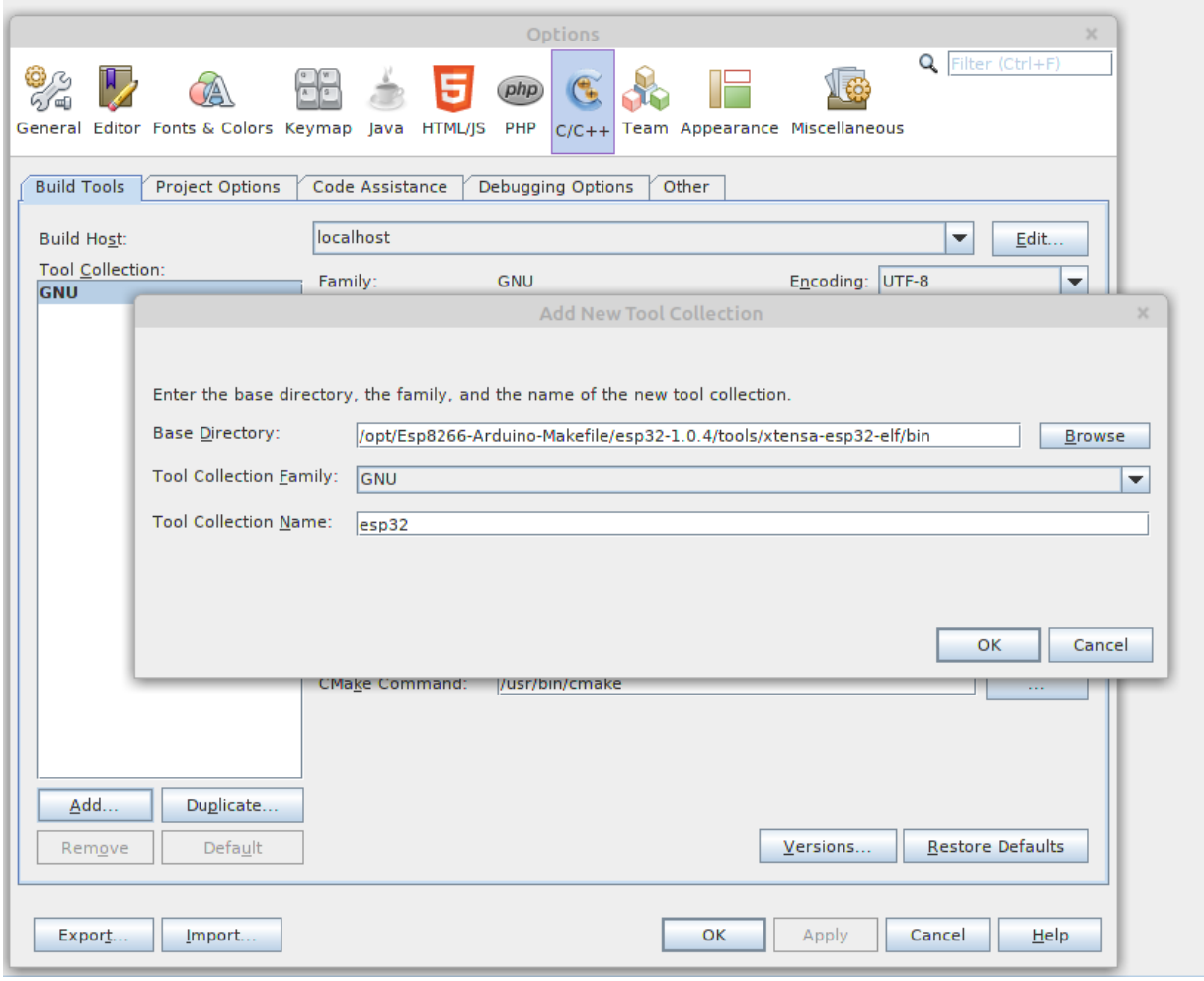

Mozilla Firefox
Fichier Édition Affichage Historique Marque-pages Outils Aide
172.18.58.31/
172.18.58.31
ENT PhilippeS Lycée Gabriel Toucha... LYCEE GABRIEL TOU...
Click here to turn the LED on pin 2 on.
Click here to turn the LED on pin 2 off.

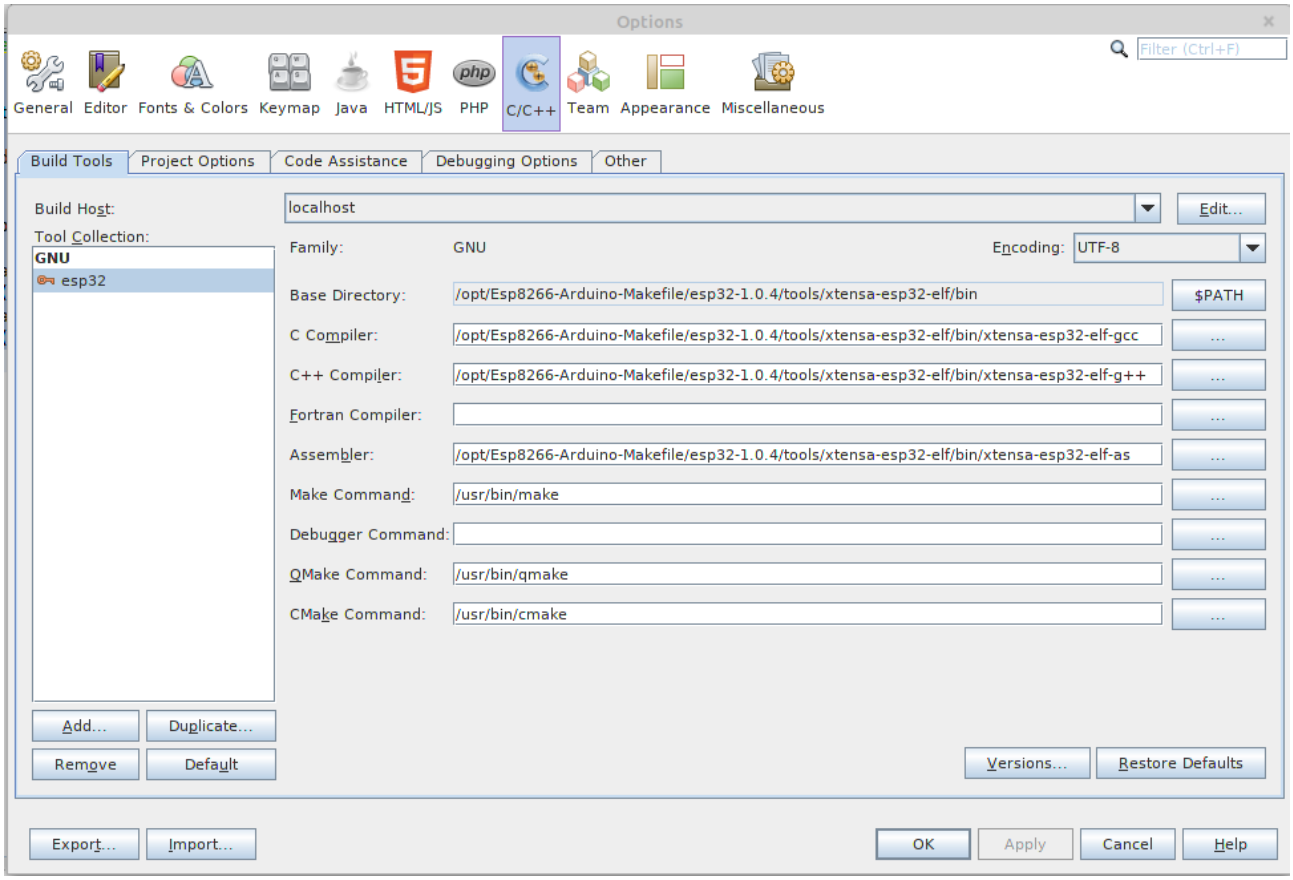
4 Configuration de NetBeans

Il faut enregistrer une nouvelle "tool chain" (chaîne d'outils : compilateur/linker/assembleur/...) dans NetBeans.

Dans le menu **Tools** choisir l'item **Options** une fenêtre s'ouvre
Cliquer sur l'onglet **C/C++** pour ouvrir la fenêtre suivante

Créez une tool chain esp32 (de type GNU) et renseignez les chemins des différents outils.
Cliquer sur Add... pour ajouter un outil de compilation. La fenêtre **Add New Tool Collection** s'ouvre





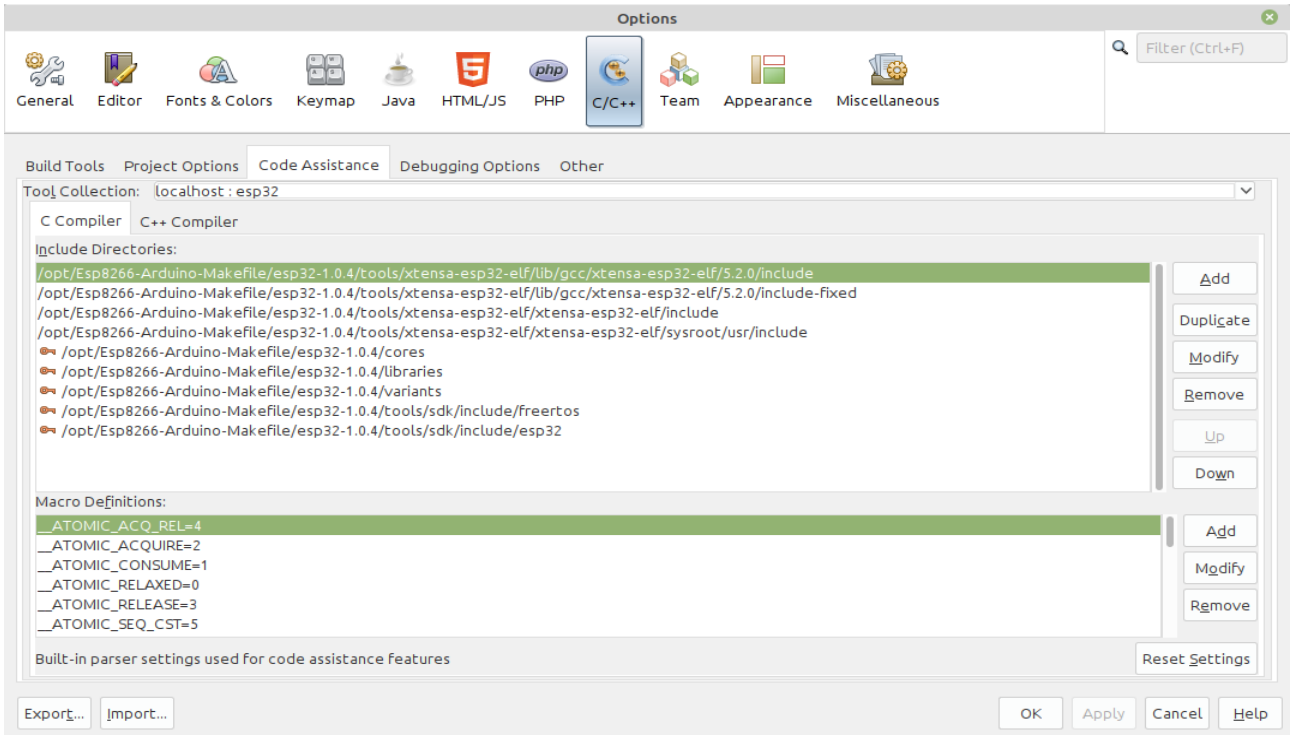
Base Directory : /opt/Esp8266-Arduino-Makefile/esp32-1.0.4/tools/xtensa-esp32-elf/bin
C Compiler : /opt/Esp8266-Arduino-Makefile/esp32-1.0.4/tools/xtensa-esp32-elf/bin/xtensa-esp32-elf-gcc
C++ Compiler : /opt/Esp8266-Arduino-Makefile/esp32-1.0.4/tools/xtensa-esp32-elf/bin/xtensa-esp32-elf-g++
Assembler : /opt/Esp8266-Arduino-Makefile/esp32-1.0.4/tools/xtensa-esp32-elf/bin/xtensa-esp32-elf-as

Ensuite, et c'est le plus fastidieux, il faut renseigner les répertoires "code assistance" pour permettre la reconnaissance des #include et la complétion de code. Il faut le faire pour le compilateur c et pour compilateur c++.

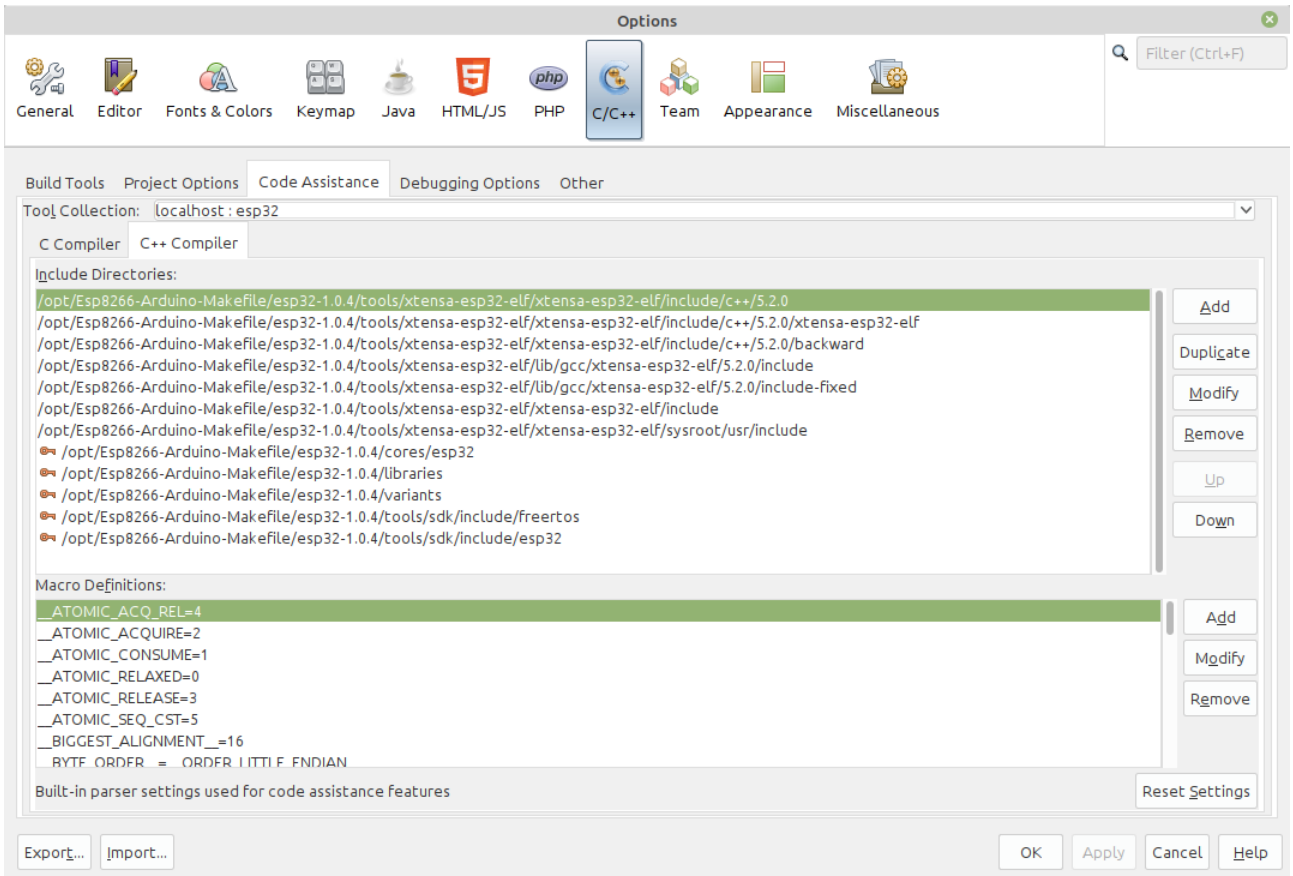
Dans l'onglet **Code Assistance**
pour Tool Collection : **esp32**

Si vous avez des objets non reconnus, il faut rajouter des répertoires dans la "Code Assistance".

Code assistant pour l'onglet C Compiler



Code assistant Pour l'onglet C++ compiler

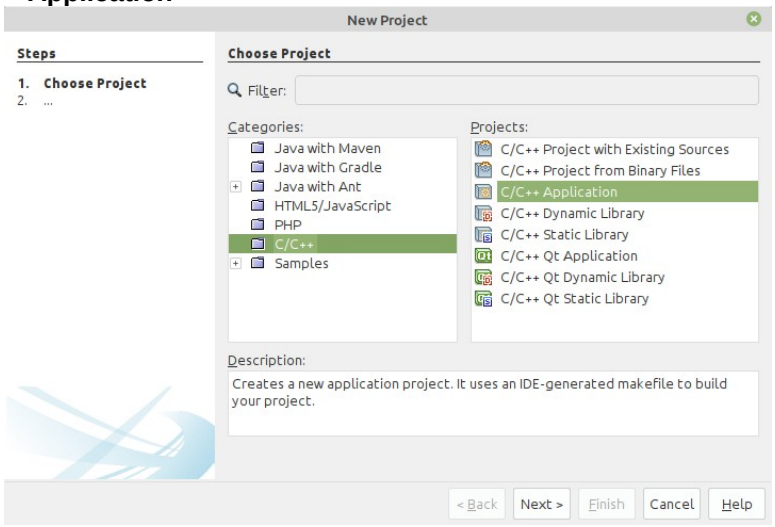


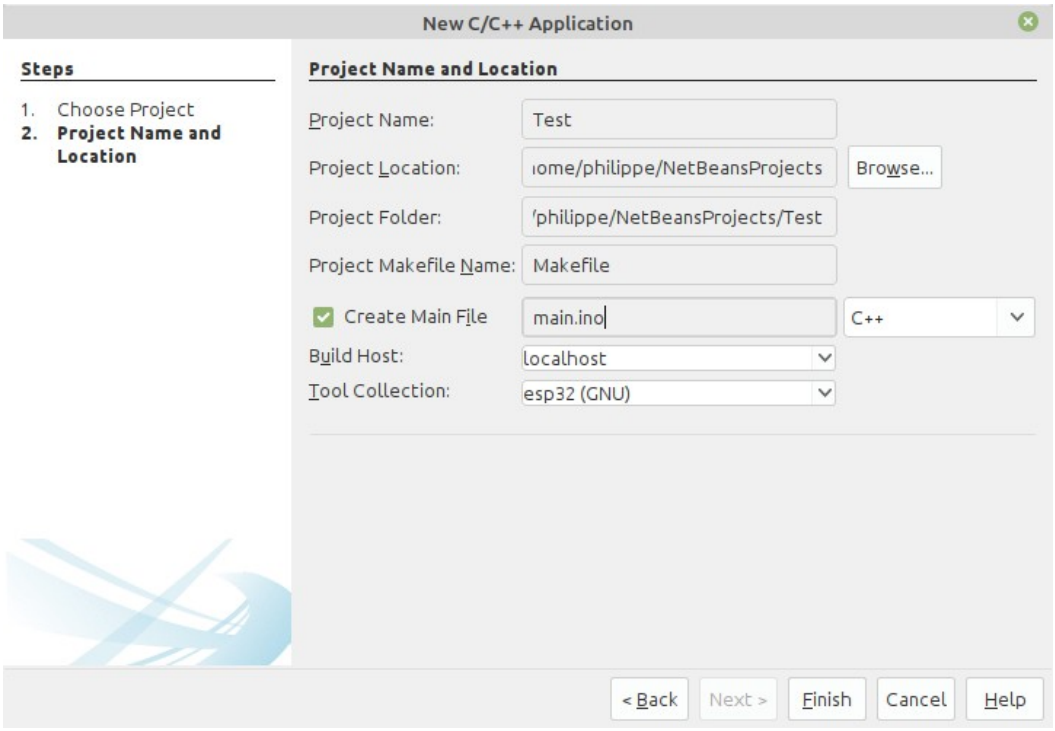
Si vous avez des objets non reconnus, il faut rajouter des répertoires dans la "Code Assistance".

5 Création d'un nouveau projet esp32

Maintenant nous pouvons créer un nouveau projet pour l'esp32

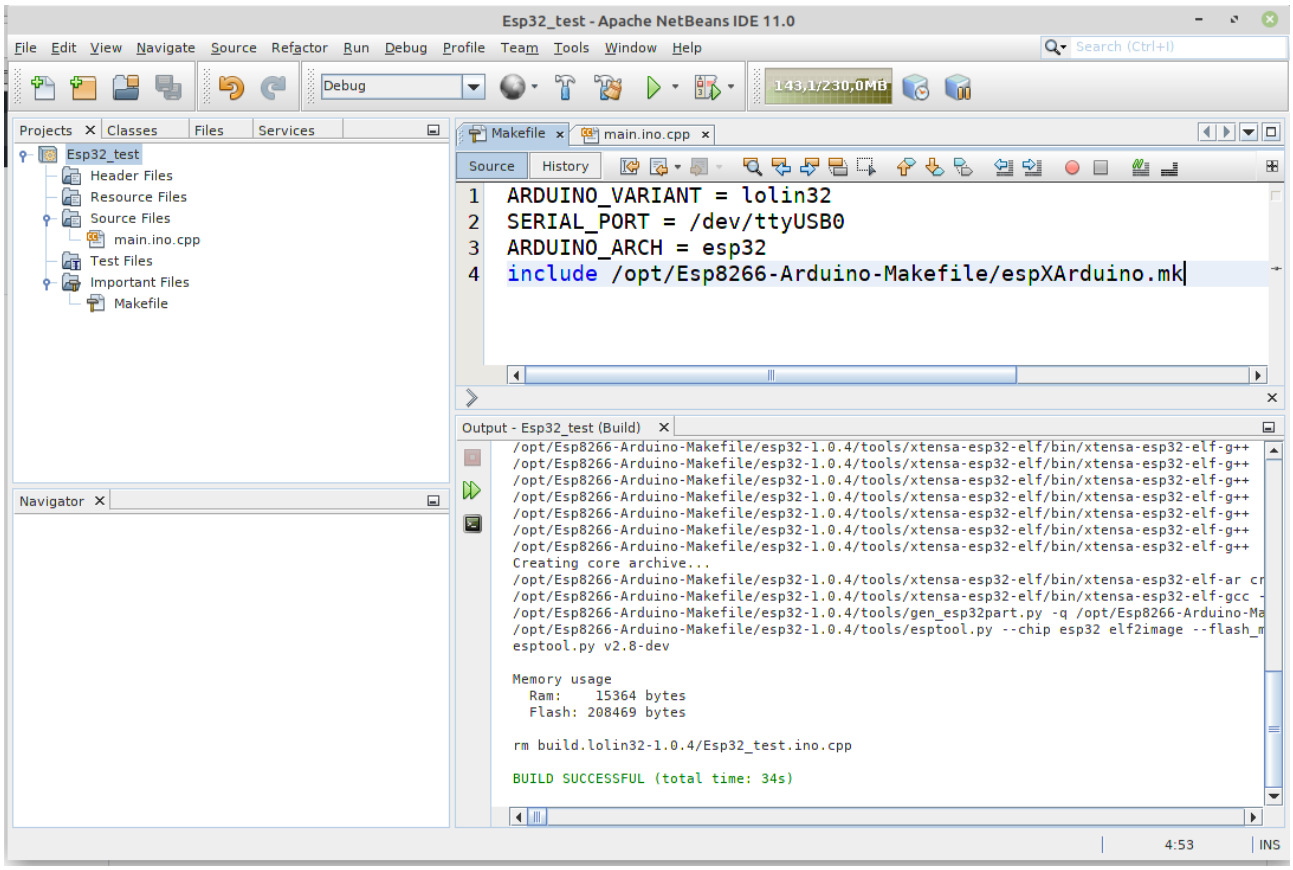
Dans le menu **File** choisir **New Project**
puis application **C/C++ Application**





renommer main en **main.ino**
Choisir pour Tool Collection **esp32 (GNU)**

Dans **Makefile** remplaçer les lignes par les lignes suivantes




```
ARDUINO_VARIANT = lolin32
SERIAL_PORT = /dev/ttyUSB0
ARDUINO_ARCH = esp32
include /opt/Esp8266-Arduino-Makefile/espXArduino.mk
```

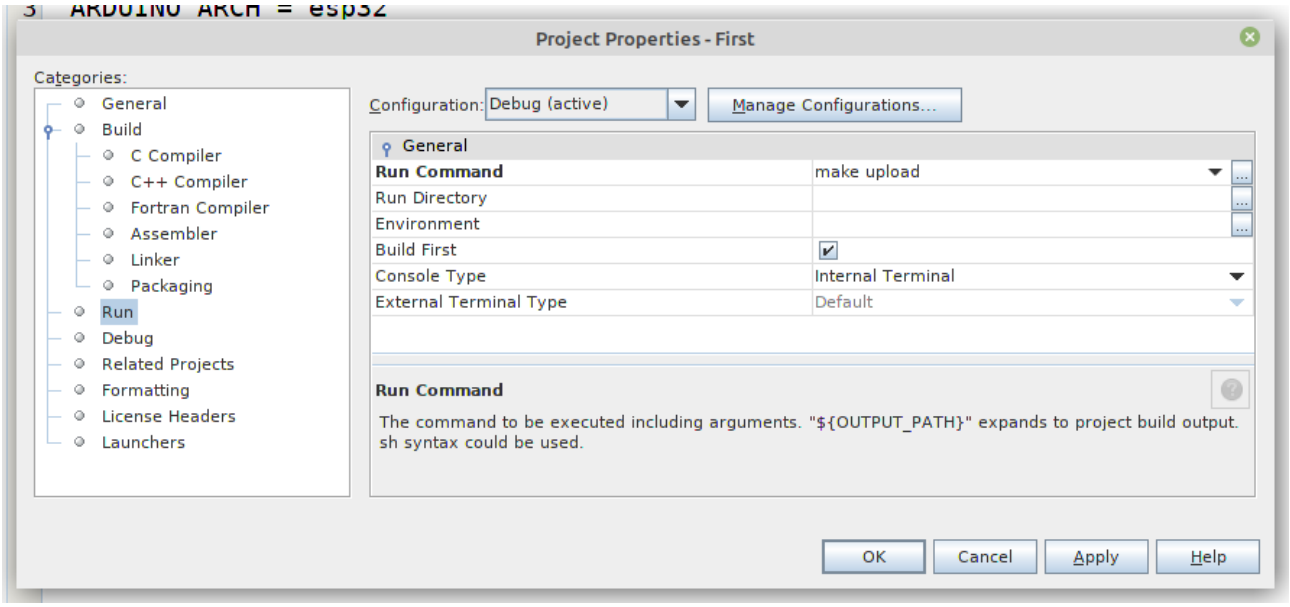
Pour tester la compilation, dans main.ino.cpp copier le programme suivant

```
#include <Arduino.h>

/*
  Turns on an LED on for 0,2 second, then off for one second, repeatedly.
  The ESP32 has an internal blue LED at D2 (GPIO 02)
*/
int led = 2;
void setup()
{
  pinMode(led, OUTPUT);
}
void loop()
{
  digitalWrite(led, digitalRead(led) ^1);   // turn the LED
  delay(200);                               // wait for a 0.2 second
}
```

Votre projet est maintenant utilisable. Vérifiez qu'il se compile correctement.

Pour téléverser le projet avec la commande Run, il suffit de modifier les propriétés du projet pour indiquer la commande "**make upload**" :

Dans propriétés du projet



6 Librairie supplémentaires

Si vous utilisez des librairies supplémentaires vous pouvez placer les sources dans le répertoire ou dans un sous répertoire de librairies

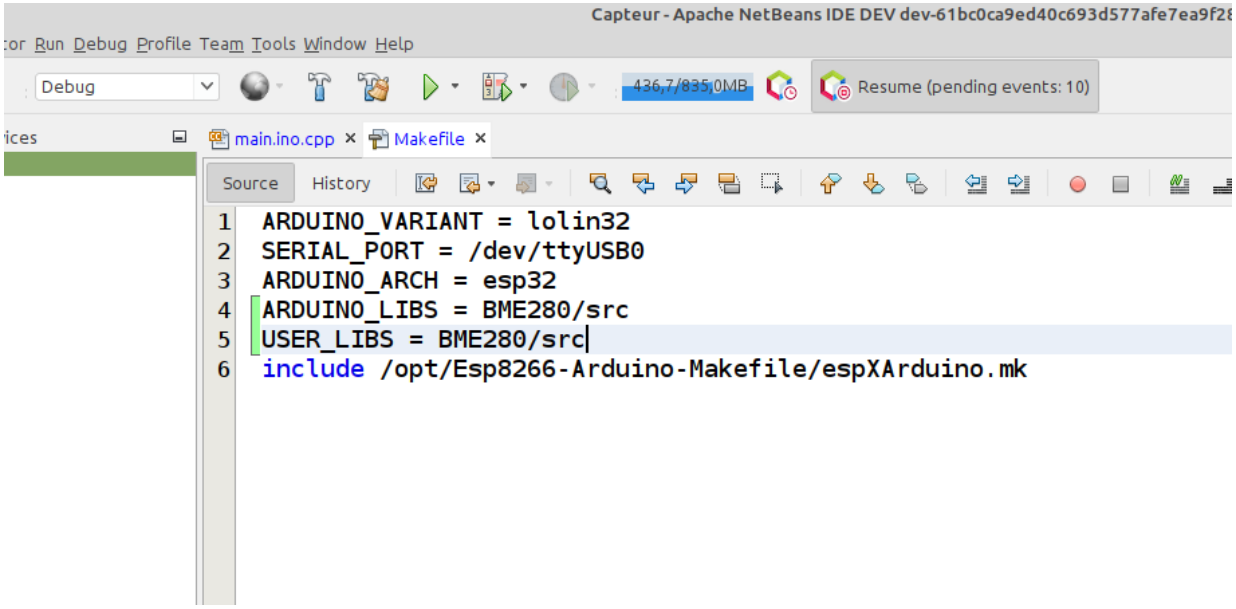
```
/opt/Esp8266-Arduino-Makefile/libraries/
```

Prenons comme exemple un capteur BME280. Les classes sont disponible sur le dépôt officiel https://github.com/finitespace/BME280

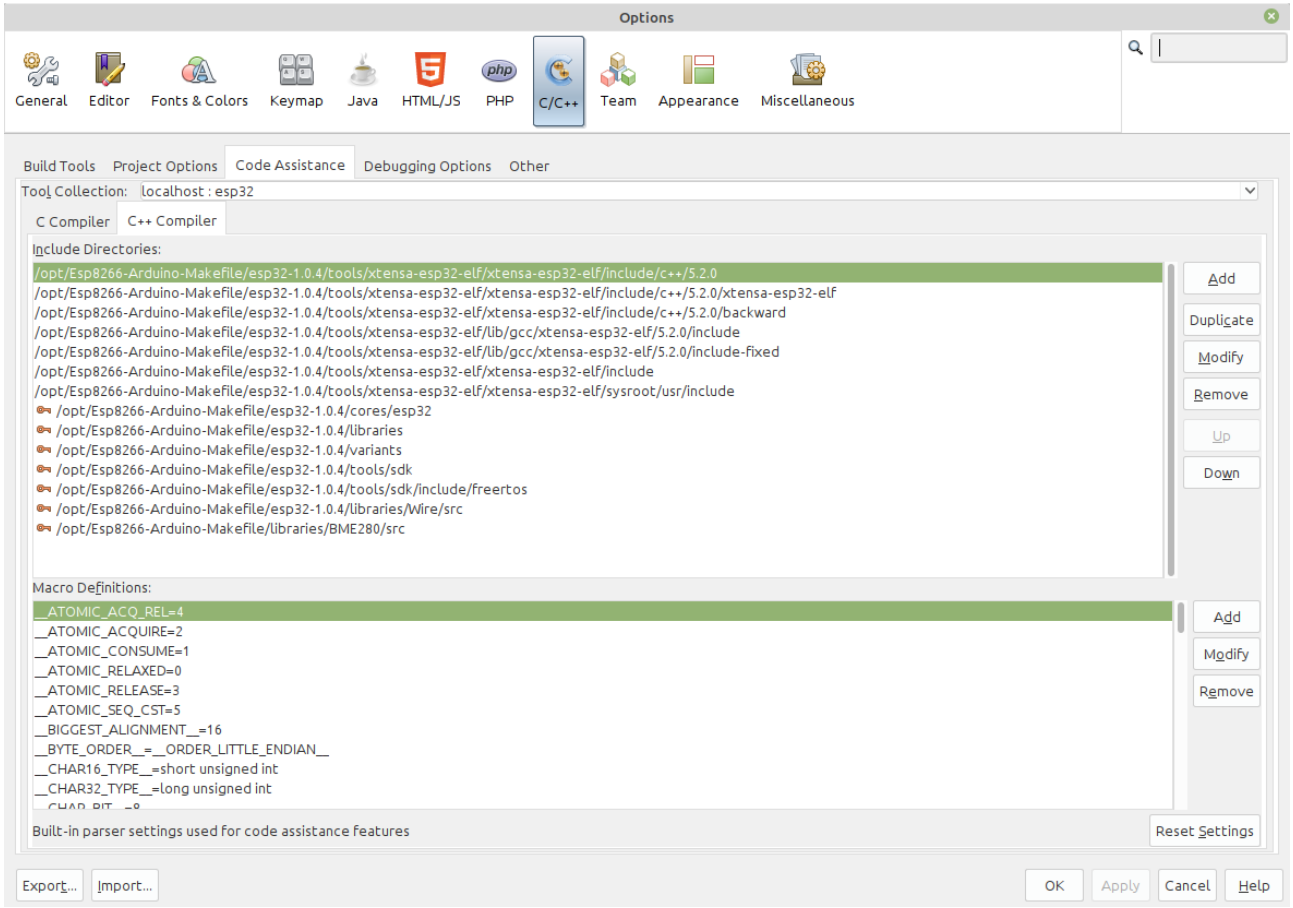
Cloner ce dépôt dans le répertoire **librairies**

```
git clone https://github.com/finitespace/BME280.git
```

Dans le fichier **Makefile**, Il faut alors mettre à jour la liste des librairies utilisées dans les variables ARDUINO_LIBS et USER_LIBS, comme le montre l'écran suivant.



Pour obtenir l'auto-complétion, dans code Assistance, ajouter le chemin vers la bibliothèque. Dans notre exemple **BME280/src** et **Wire/src**



Nous pouvons à partir de maintenant utiliser la bibliothèque BME280 dans notre code

```
#include <Arduino.h>
#include <Wire.h>
#include <BME280I2C.h>
#define SERIAL_BAUD 115200

BME280I2C::Settings settings(
   BME280::OSR_X1,
   BME280::OSR_X1,
   BME280::OSR_X1,
   BME280::Mode_Forced,
   BME280::StandbyTime_1000ms,
   BME280::Filter_Off,
   BME280::SpiEnable_False,
   BME280I2C::I2CAddr_0x76      // 0x77 I2C address pour BME 280 Adafruit.
);

BME280I2C bme(settings);
...
```