**Little EVG Library for PC Windows**

Reference: <https://github.com/littlevgl/pc_simulator_sdl_eclipse>

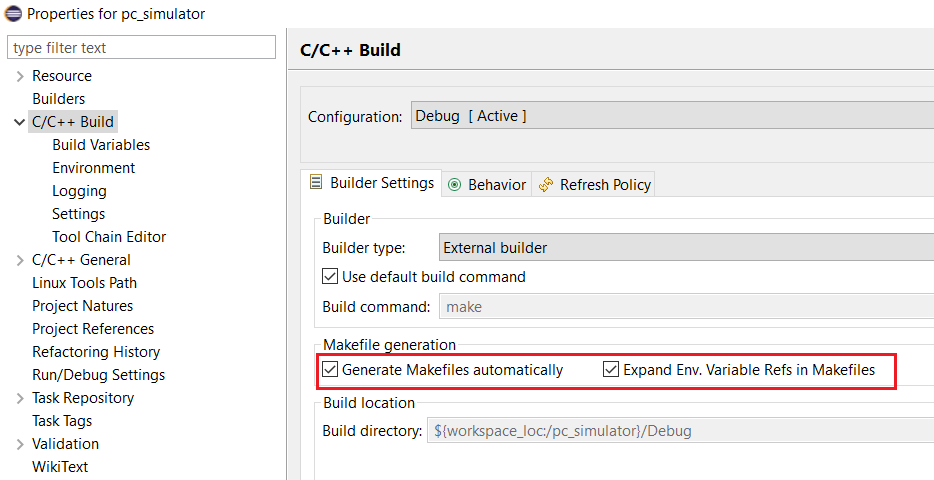
+ Get the pc simu ‘pc\_simulator\_sdl\_eclipse’ which already configured for Eclipse CDT.

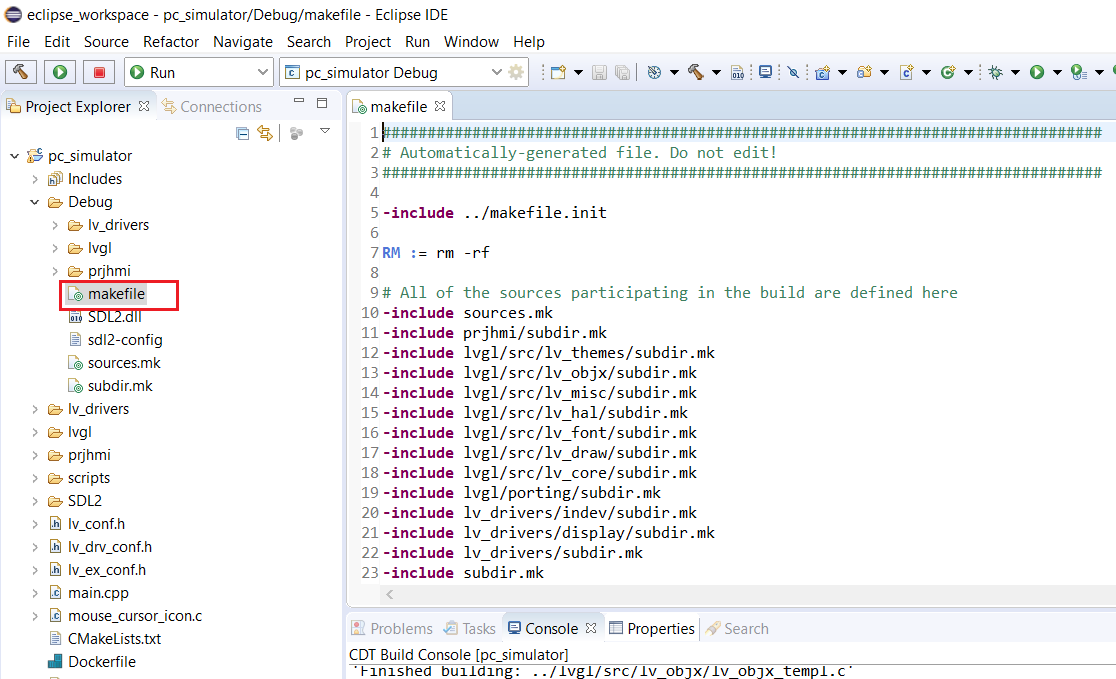
+ Download and install Eclipse CDT from <http://www.eclipse.org/cdt/>

+ Download SDL from <https://www.libsdl.org/>

Open ‘pc\_simulator\_sdl\_eclipse’: this project is configured with Cross GCC/G++ platform compiler with can work with SDL lib compiled by MingGW 64

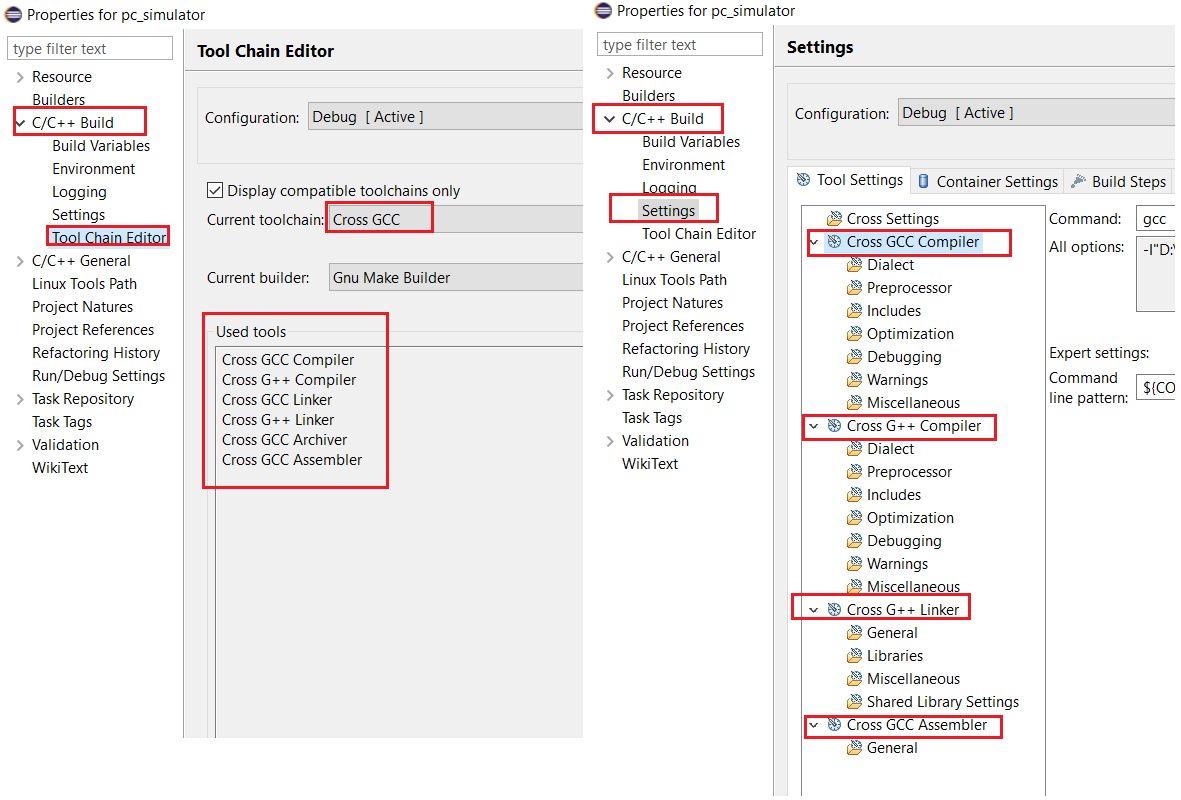
No need Makefile if use Eclipse CDT tool

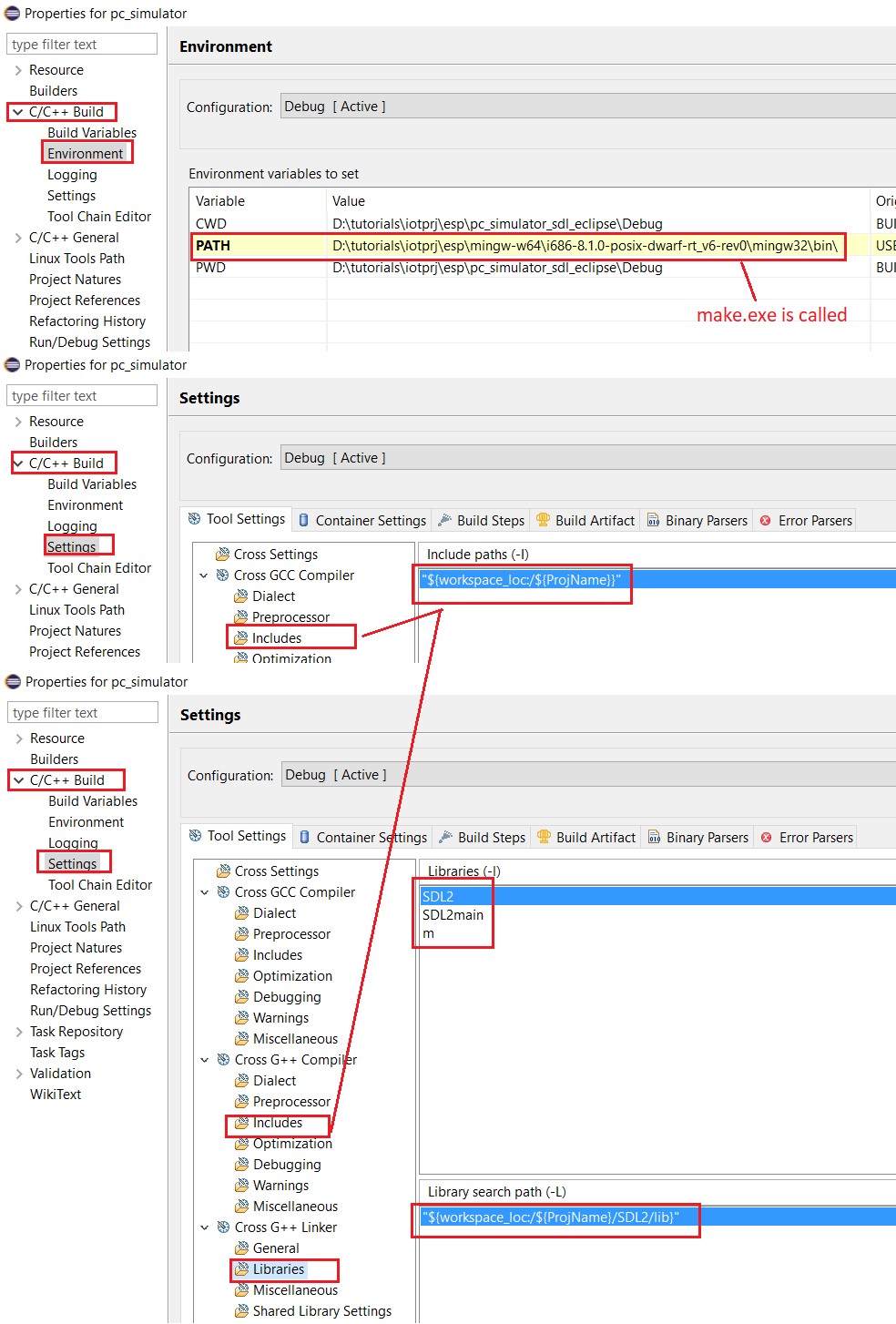




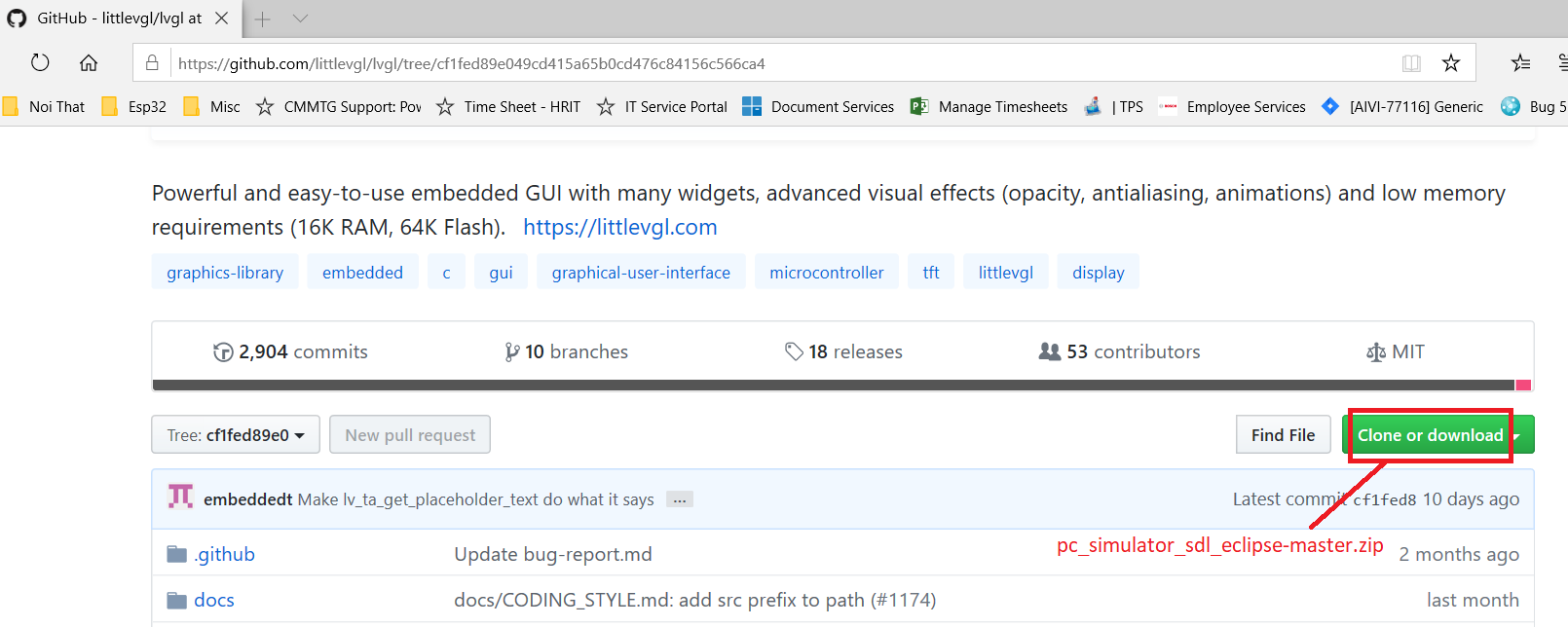
Convert C to CPP project (‘pc\_simulator\_sdl\_eclipse is a C project) using Cross G++ platform:

|  |  |
| --- | --- |
| \packages\pc\_simulator\_sdl\_eclipse:  + delete 2 folders: lvgl, SDL2 and lvgl\_examples  + copy folder \packages\lvgl-master\ into \packages\pc\_simulator\_sdl\_eclipse\lvgl  + copy folder \packages\SDL2-2.0.5\i686-w64-mingw32\include\SDL2 into \packages\pc\_simulator\_sdl\_eclipse\SDL2  + copy folder \packages\SDL2-2.0.5\i686-w64-mingw32\lib into \packages\pc\_simulator\_sdl\_eclipse\SDL2\lib  => if having new lvgl and SDL2, copy them as above |  |





<https://github.com/littlevgl/lvgl/tree/cf1fed89e049cd415a65b0cd476c84156c566ca4>

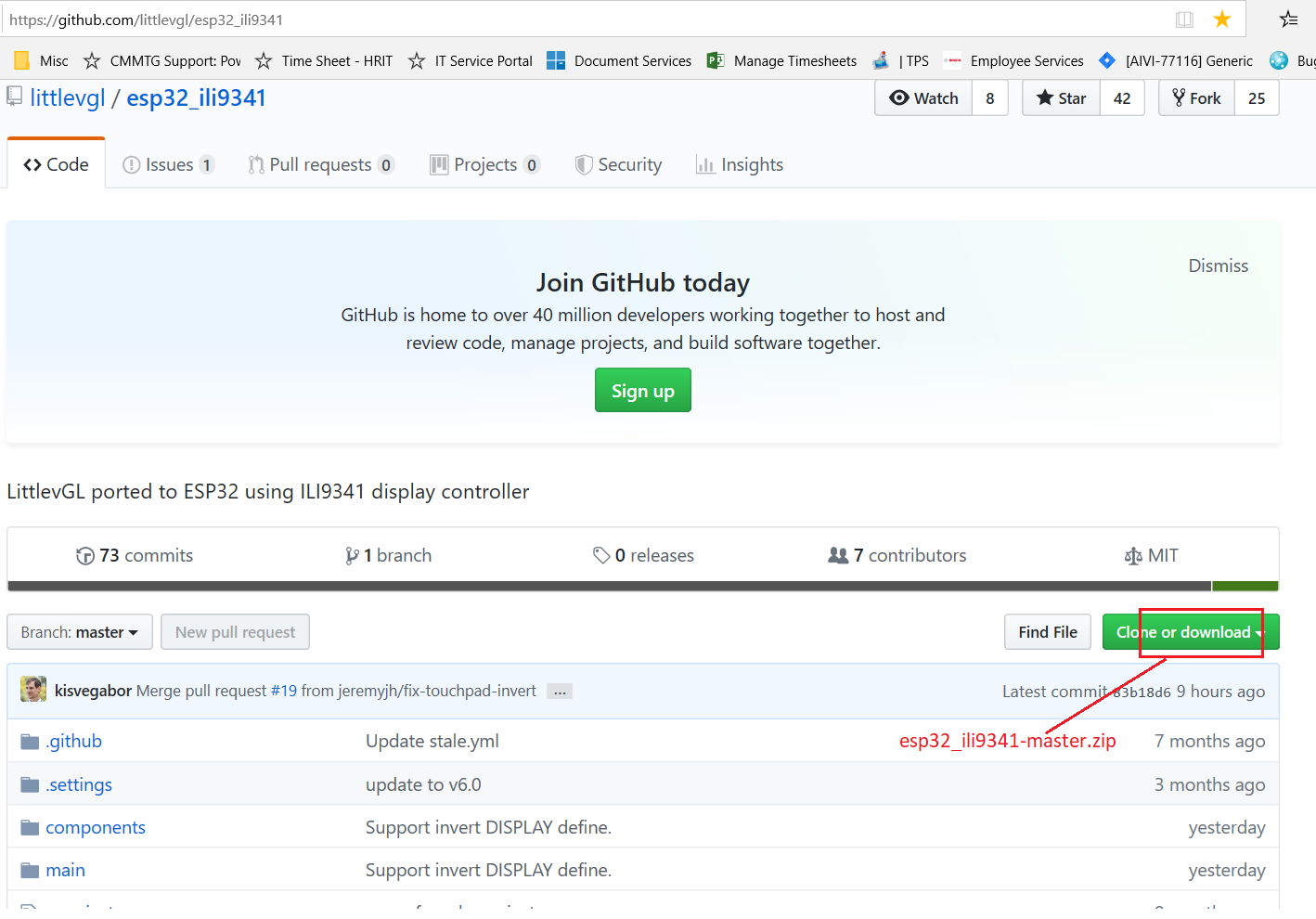


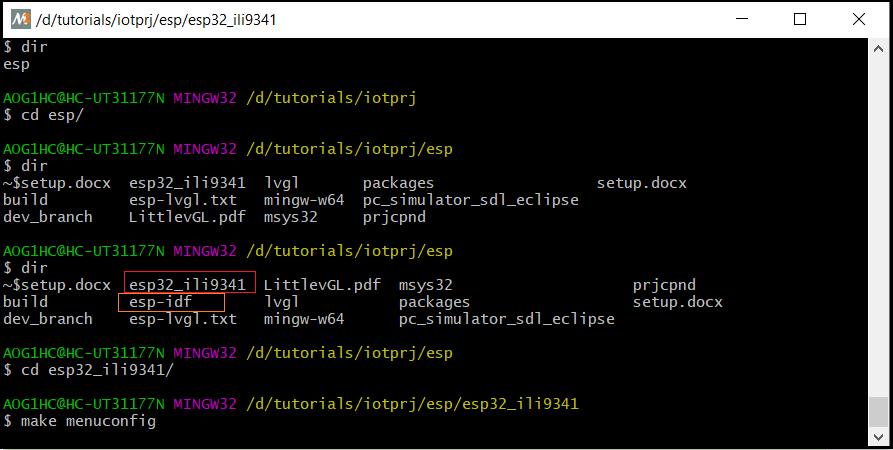
Little EVG Library for ESP Target

esp32 win32 toolchain: msys32  
<https://docs.espressif.com/projects/esp-idf/en/stable/get-started/windows-setup.html>

esp-idf: <https://github.com/espressif/esp-idf>

littlevgl for esp32: <https://github.com/littlevgl/esp32_ili9341>





Export IDF\_PATH:

$ export IDF\_PATH=/d/tutorials/iotprj/esp/esp-idf

\packages\esp32\_ili9341-master:

+ copy folder \packages\lvgl-master into \packages\esp32\_ili9341-master\lvgl

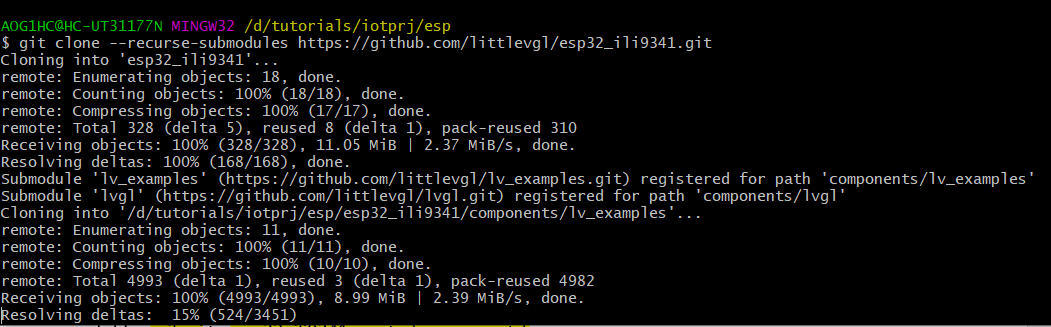
+ copy esp32\_ili9341-master\lvgl\_component.mk to esp32\_ili9341-master\components\lvgl\component.mk to build with ESP-IDF

+ copy folder prjhmi to esp32\_ili9341-master\components\

=> if having new lvgl, copy them as above

<https://github.com/littlevgl/esp32_ili9341>

git clone --recurse-submodules <https://github.com/littlevgl/esp32_ili9341.git>



make menuconfig -> serial flash COM7

drv/disp\_spi.h:

#define DISP\_SPI\_MOSI 23

#define DISP\_SPI\_CLK 19

#define DISP\_SPI\_CS 22

drv/ili9341.h:

#define DISP\_BUF\_SIZE (LV\_HOR\_RES\_MAX \* 40)

#define ILI9341\_DC 21

#define ILI9341\_RST 18

#define ILI9341\_BCKL 5

ESP-IDF for Eclipse

<https://gitdemo.readthedocs.io/en/latest/eclipse-setup.html>

<https://exploreembedded.com/wiki/Setting_up_Eclipse_for_ESP-IDF>

* Once Eclipse is running, choose File -> Import...
* In the dialog that pops up, choose “C/C++” -> “Existing Code as Makefile Project” and click Next.
* On the next page, enter “Existing Code Location” to be the directory of your IDF project. Don’t specify the path to the ESP-IDF directory itself.
* On the same page, under “Toolchain for Indexer Settings” choose “Cross GCC”. Then click Finish.

### Project Properties[¶](https://gitdemo.readthedocs.io/en/latest/eclipse-setup.html#project-properties)

* The new project will appear under Project Explorer. Right-click the project and choose Properties from the context menu.
* Click on the “Environment” properties page under “C/C++ Build”. Click “Add...” and enter name V and value 1.
* Click “Add...” again, and enter name IDF\_PATH. The value should be the full path where ESP-IDF is installed. Windows users: Use forward-slashes not backslashes for this path, ie C:/Users/MyUser/Development/esp-idf.

Windows users only, follow these two additional steps:

* On the same Environment property page, edit the PATH environment variable. Delete the existing value and replace it with C:\msys32\usr\bin;C:\msys32\mingw32\bin;C:\msys32\opt\xtensa-esp32-elf\bin (If you installed msys32 to a different directory then you’ll need to change these paths to match).
* Click on the “C/C++ Build” top-level properties page then uncheck “Use default build command” and enter this for the custom build command: bash ${IDF\_PATH}/tools/windows/eclipse\_make.sh.

All users, continue with these steps:

Navigate to “C/C++ General” -> “Preprocessor Include Paths” property page:

* Click the “Providers” tab
* In the list of providers, click “CDT Cross GCC Built-in Compiler Settings”. Under “Command to get compiler specs”, replace the text ${COMMAND} at the beginning of the line with xtensa-esp32-elf-gcc. This means the full “Command to get compiler specs” should be xtensa-esp32-elf-gcc ${FLAGS} -E -P -v -dD "${INPUTS}".
* xtensa-esp32-elf-gcc ${FLAGS} -E -P -v -dD "${INPUTS}"
* In the list of providers, click “CDT GCC Build Output Parser” and type xtensa-esp32-elf- at the beginning of the Compiler command pattern. This means the full Compiler command pattern should be xtensa-esp32-elf-(g?cc)|([gc]\+\+)|(clang)
* xtensa-esp32-elf-(g?cc)|([gc]\+\+)|(clang)
* Click OK to close the Properties dialog, and choose Project -> Build to build your project.

D:\iotprj\buildtools\msys32\usr\bin;D:\iotprj\buildtools\msys32\mingw32\bin;D:\iotprj\buildtools\msys32\opt\xtensa-esp32-elf\bin

### Flash from Eclipse[¶](https://gitdemo.readthedocs.io/en/latest/eclipse-setup.html#flash-from-eclipse)

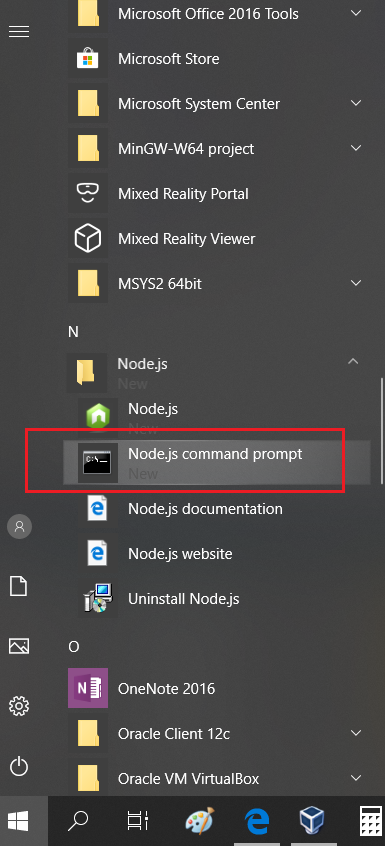
You can integrate the “make flash” target into your Eclipse project to flash using esptool.py from the Eclipse UI:

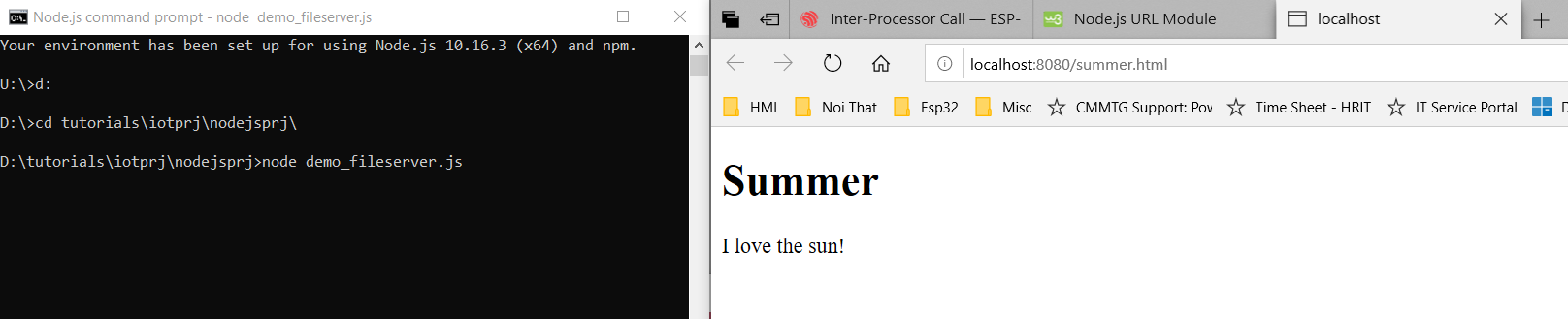
* Right-click your project in Project Explorer (important to make sure you select the project, not a directory in the project, or Eclipse may find the wrong Makefile.)
* Select Make Targets -> Create from the context menu.
* Type “flash” as the target name. Leave the other options as their defaults.
* Now you can use Project -> Make Target -> Build (Shift+F9) to build the custom flash target, which will compile and flash the project.

HTTP Server Node.js

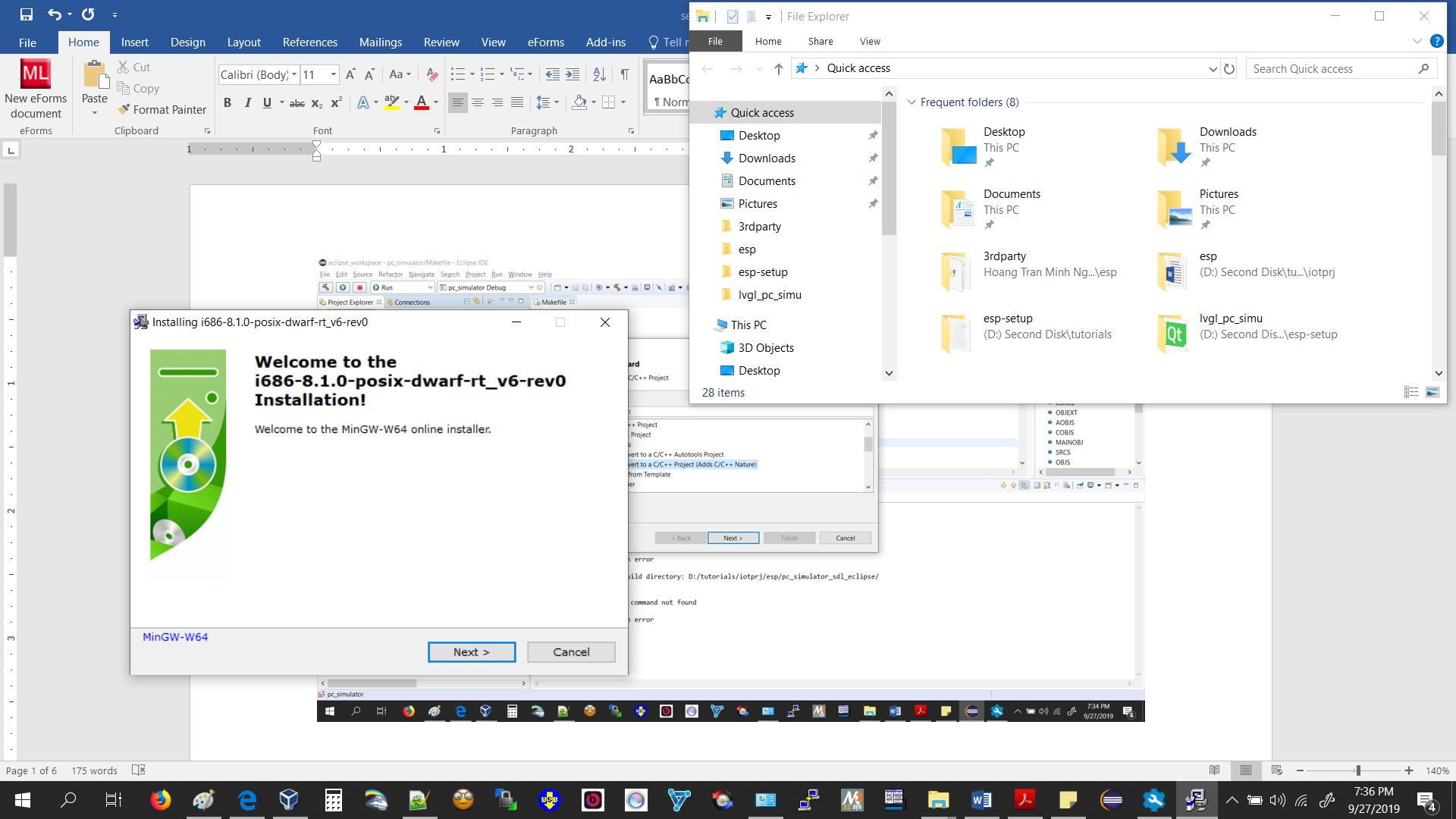
<https://nodejs.org/en/download/>

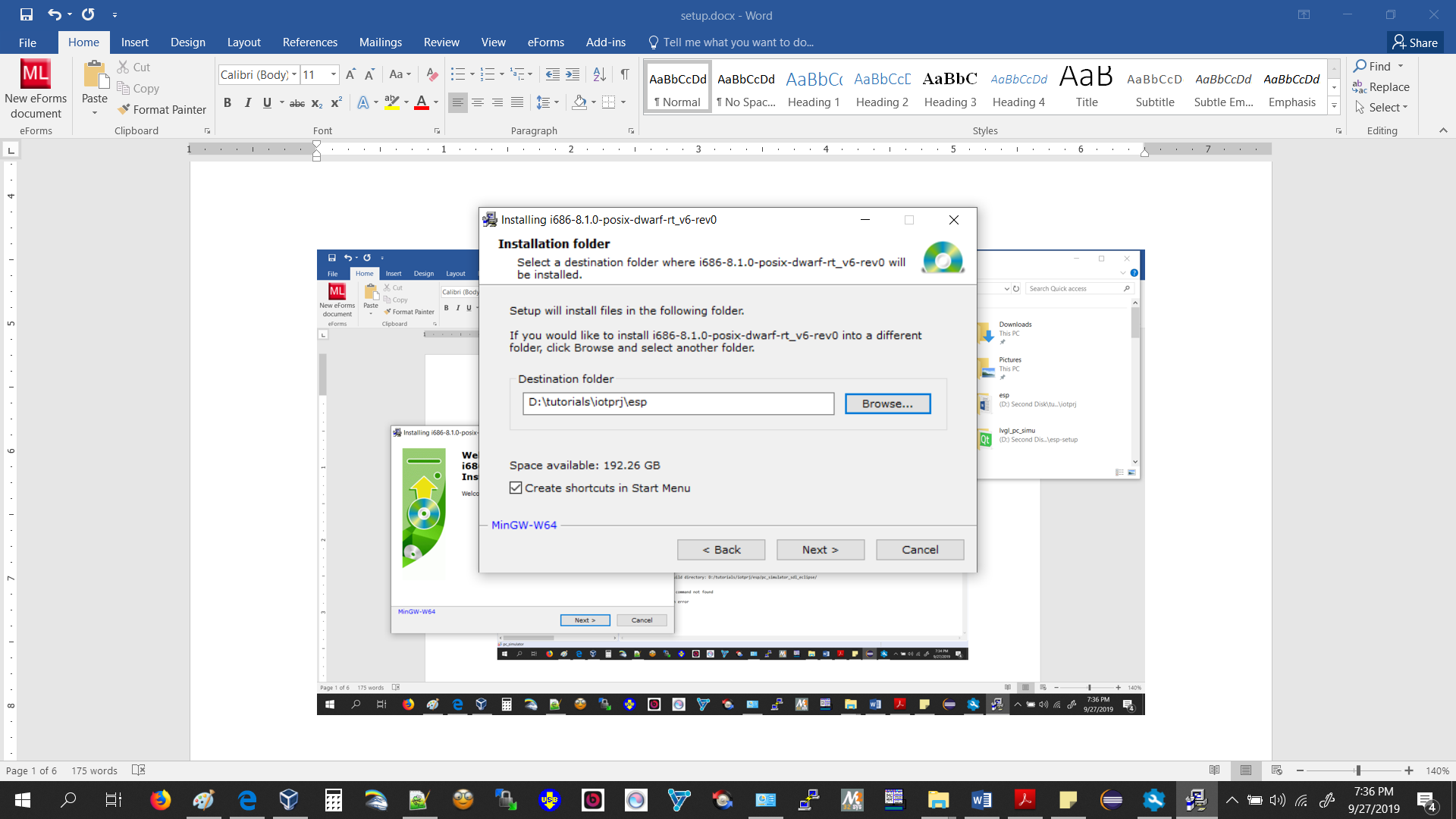
Nodejs tutorial <https://www.w3schools.com/nodejs/default.asp>





Ming





\iotprj\esp\packages

lvgl : https://github.com/littlevgl/lvgl

sdl2 mingw : https://www.libsdl.org/download-2.0.php

esp32 win32 toolchain: https://docs.espressif.com/projects/esp-idf/en/stable/get-started/windows-setup.html

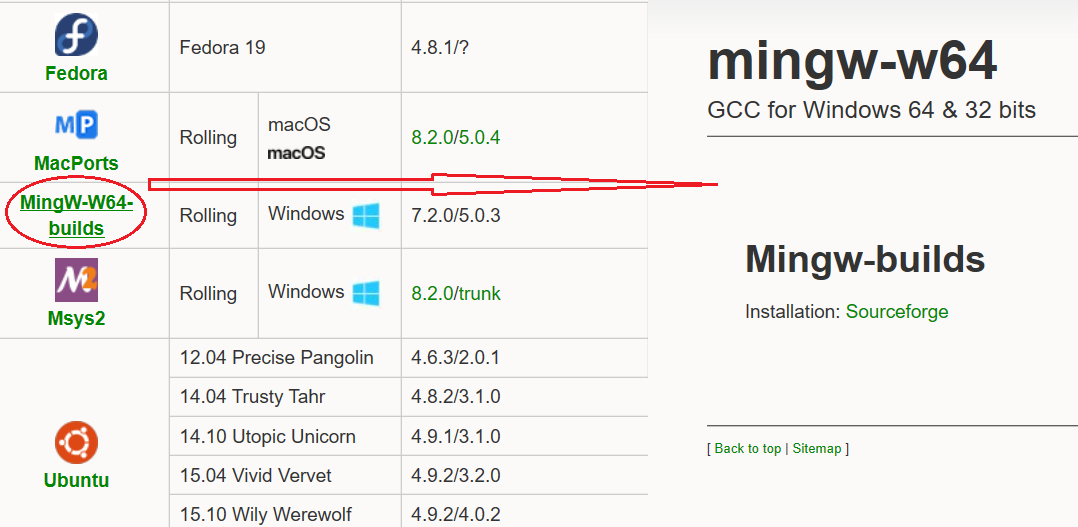
esp-idf : https://github.com/espressif/esp-idf

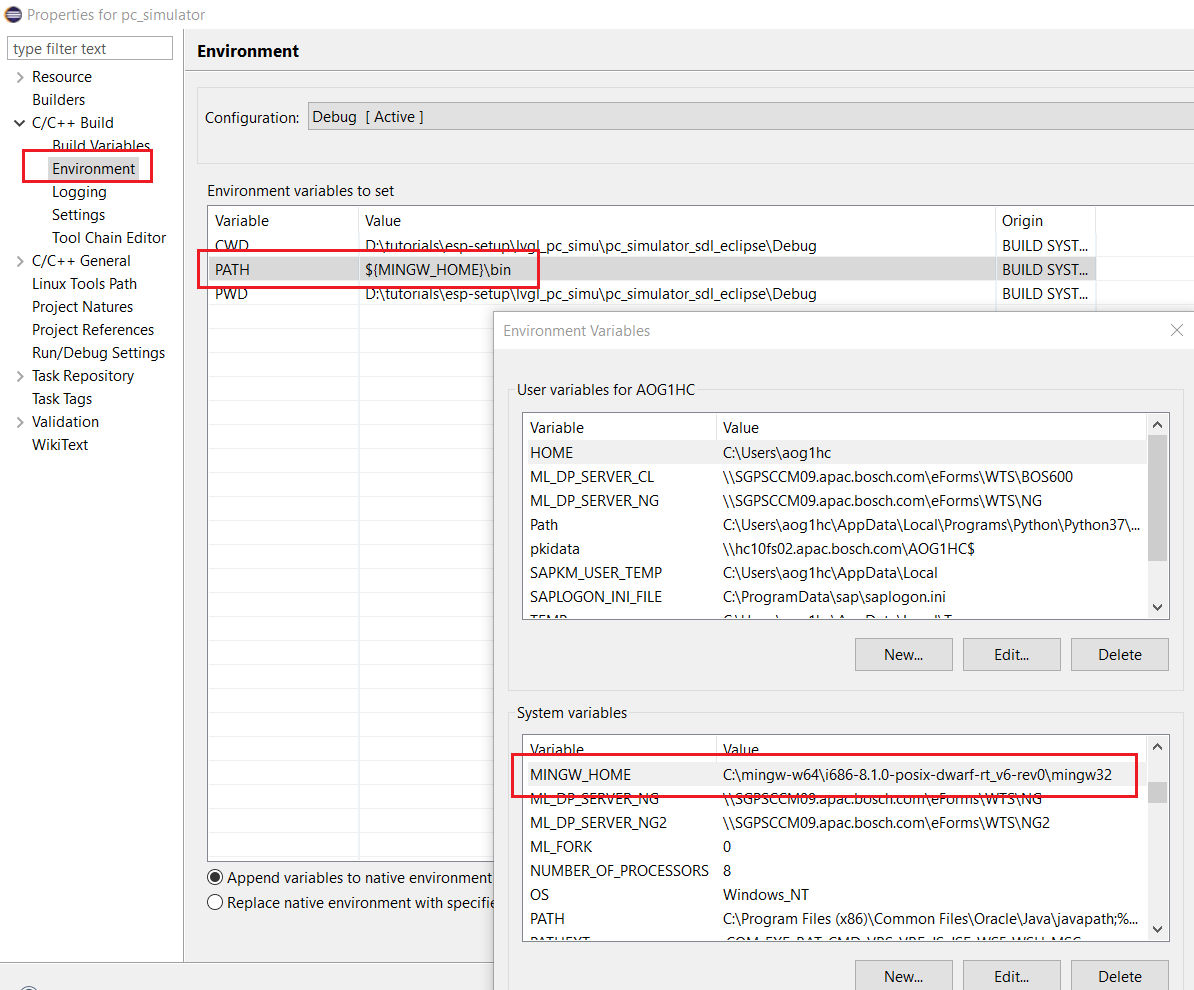
lvgl pc simulator : https://github.com/littlevgl/pc\_simulator\_sdl\_eclipse

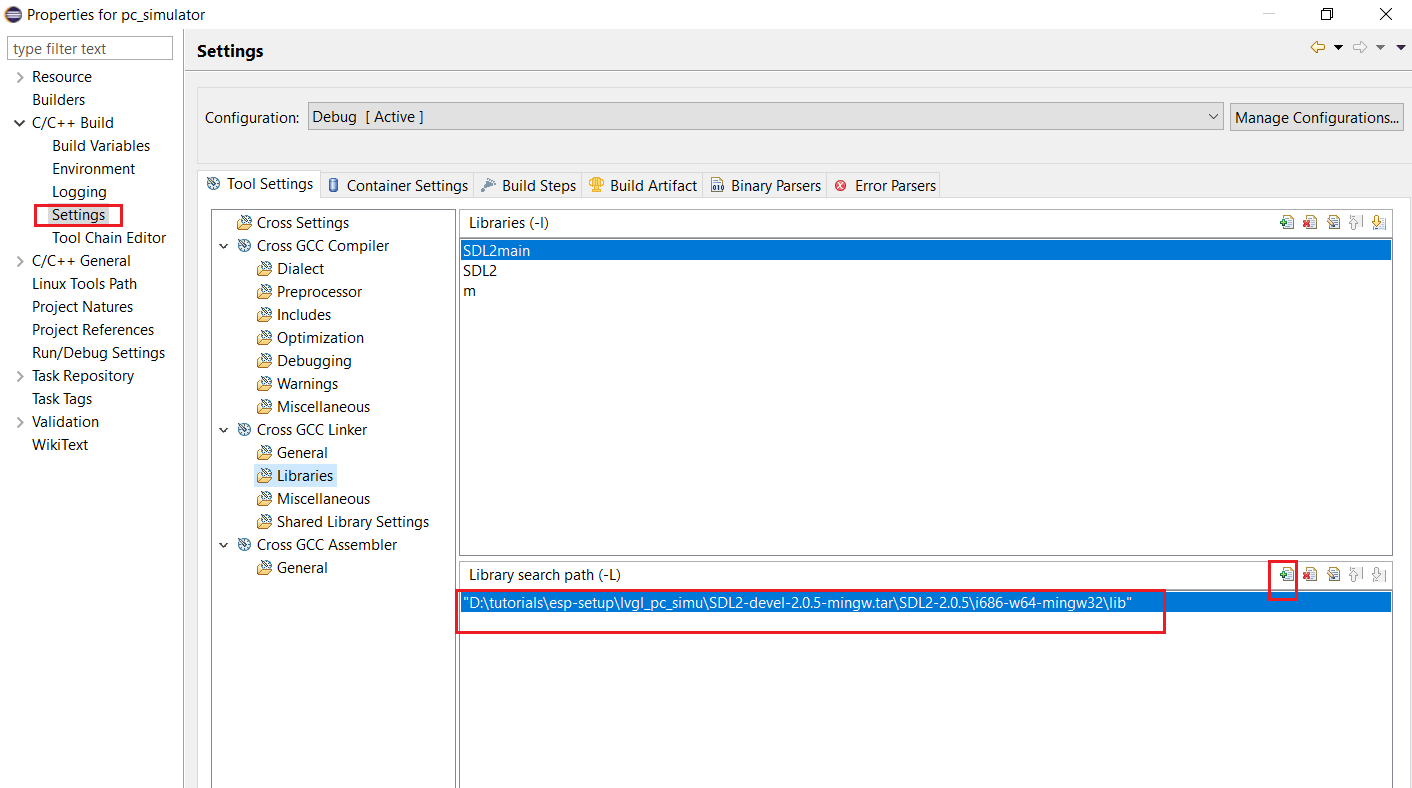
lvgl esp32 : https://github.com/littlevgl/esp32\_ili9341

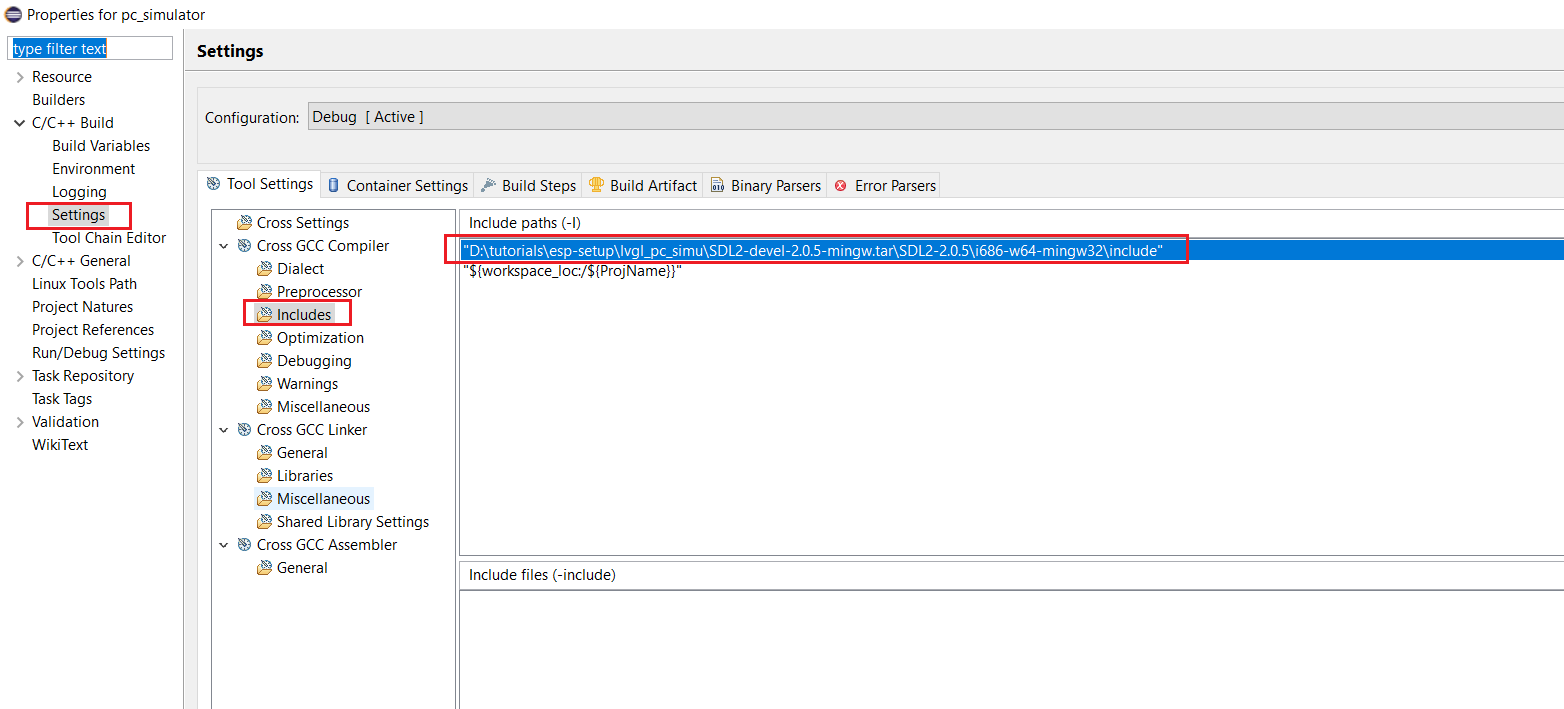
build for pc simu : Mingw64: http://mingw-w64.org/doku.php/download

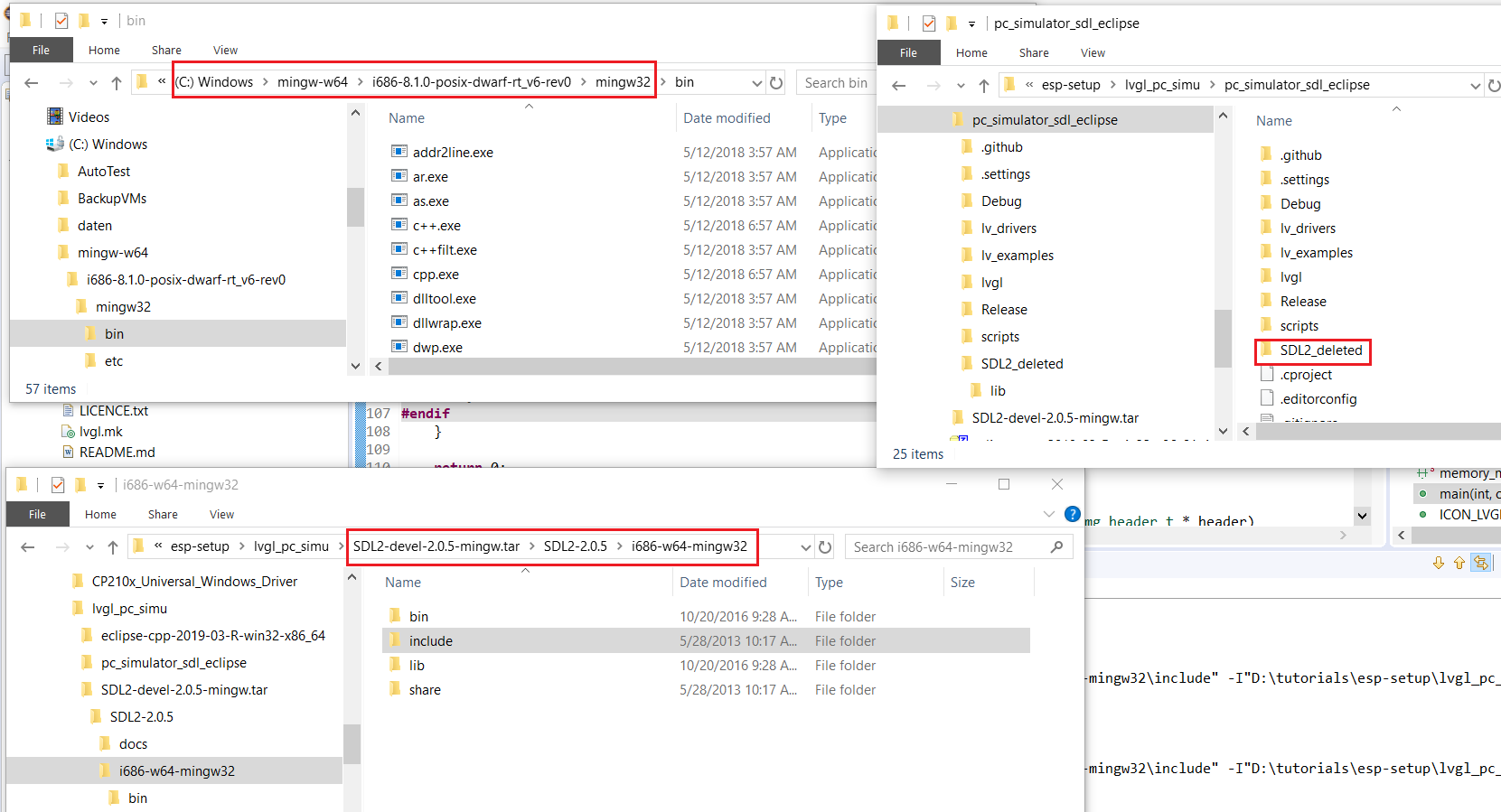
Mingw64: <http://mingw-w64.org/doku.php/download>











lvgl : https://github.com/littlevgl/lvgl

sdl2 mingw : https://www.libsdl.org/download-2.0.php

esp32 win32 toolchain: https://docs.espressif.com/projects/esp-idf/en/stable/get-started/windows-setup.html

esp-idf : https://github.com/espressif/esp-idf

lvgl pc simulator : https://github.com/littlevgl/pc\_simulator\_sdl\_eclipse

1. PC Simu:

..\3rdparty\pc\_simulator\_sdl\_eclipse:

+ delete 2 folders: lvgl and SDL2

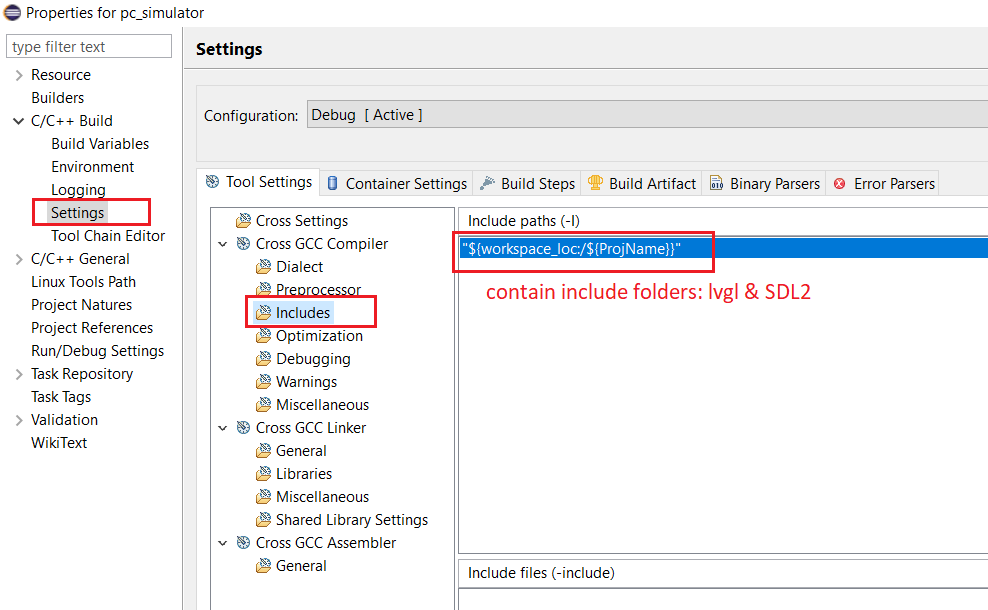
+ copy folder \3rdparty\lvgl-master\lvgl into \3rdparty\pc\_simulator\_sdl\_eclipse

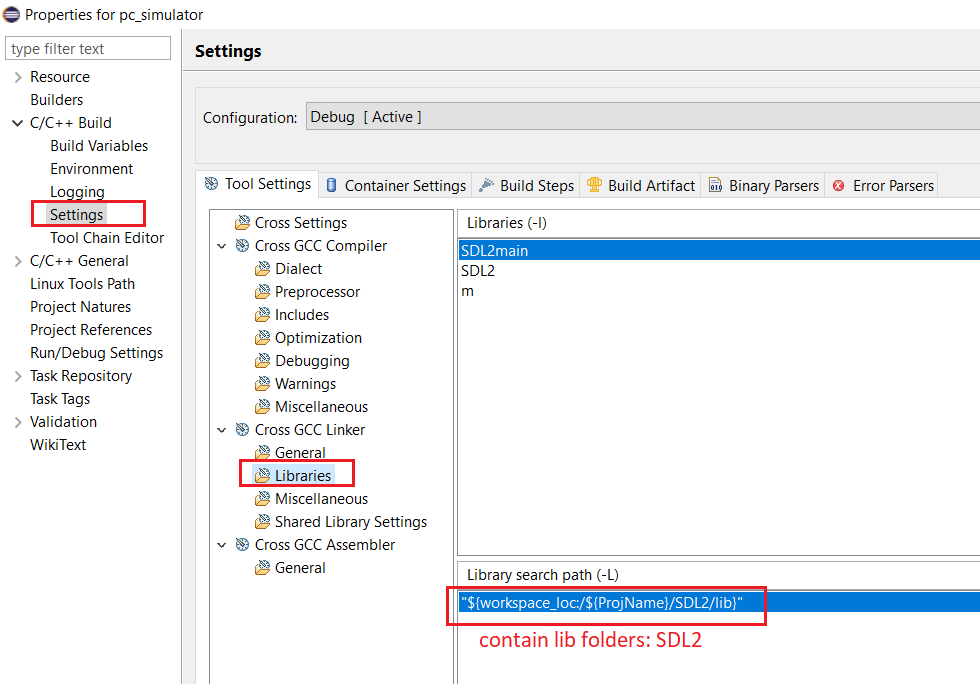
+ copy folder \3rdparty\SDL2-2.0.5\i686-w64-mingw32\include\SDL2 into \3rdparty\pc\_simulator\_sdl\_eclipse

+ copy folder \3rdparty\SDL2-2.0.5\i686-w64-mingw32\lib into \3rdparty\pc\_simulator\_sdl\_eclipse\SDL2

=> if having new lvgl and SDL2, copy them into \3rdparty\pc\_simulator\_sdl\_eclipse folder

Configure Eclipse include/lib to folder





1. PC Simu:

..\3rdparty\pc\_simulator\_sdl\_eclipse:

+ delete 2 folders: lvgl and SDL2

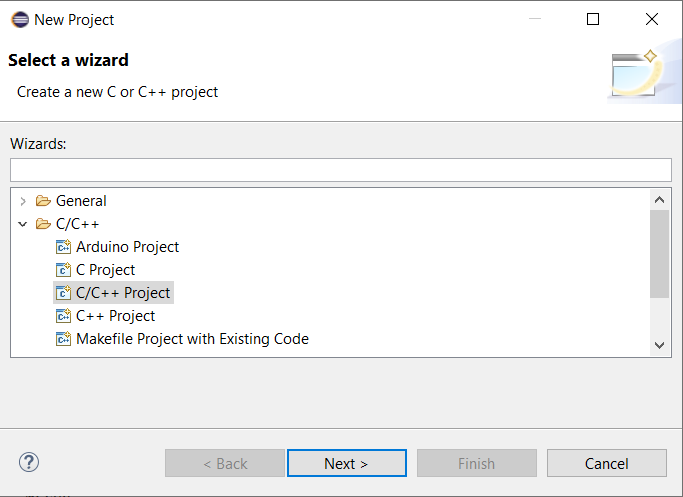
+ copy folder \3rdparty\lvgl-master\lvgl into \3rdparty\pc\_simulator\_sdl\_eclipse

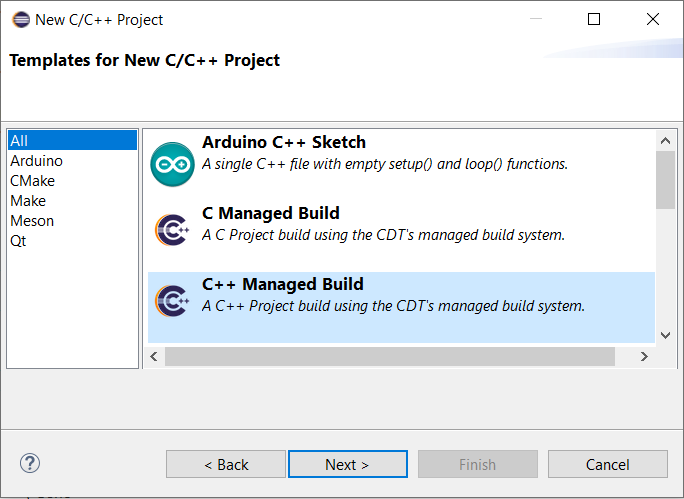
+ copy folder \3rdparty\SDL2-2.0.5\i686-w64-mingw32\include\SDL2 into \3rdparty\pc\_simulator\_sdl\_eclipse

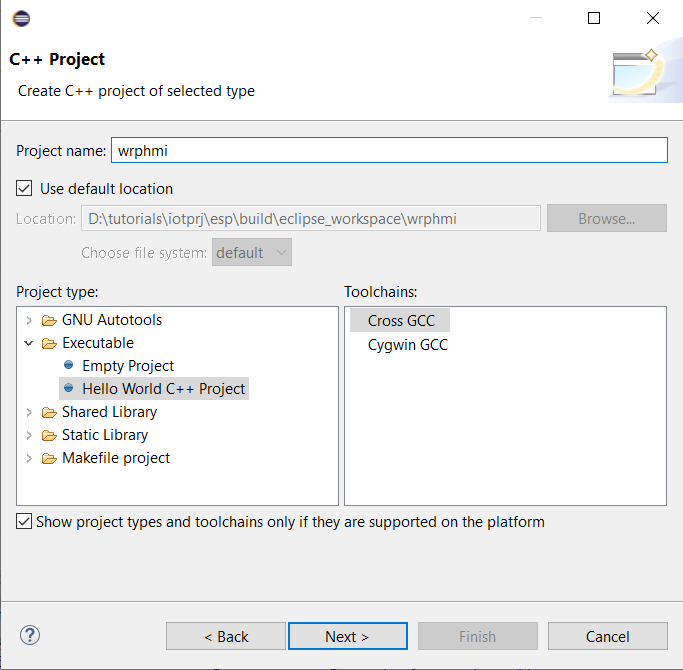
+ copy folder \3rdparty\SDL2-2.0.5\i686-w64-mingw32\lib into \3rdparty\pc\_simulator\_sdl\_eclipse\SDL2

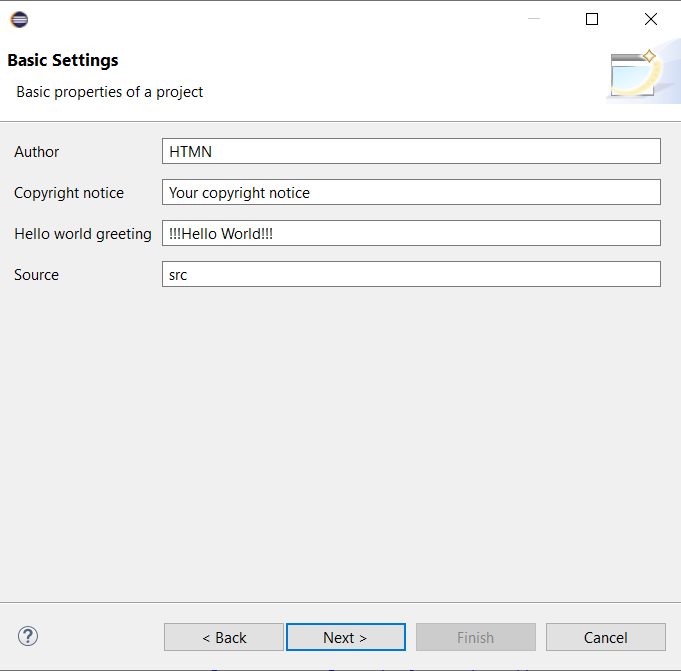
=> if having new lvgl and SDL2, copy them into \3rdparty\pc\_simulator\_sdl\_eclipse folder

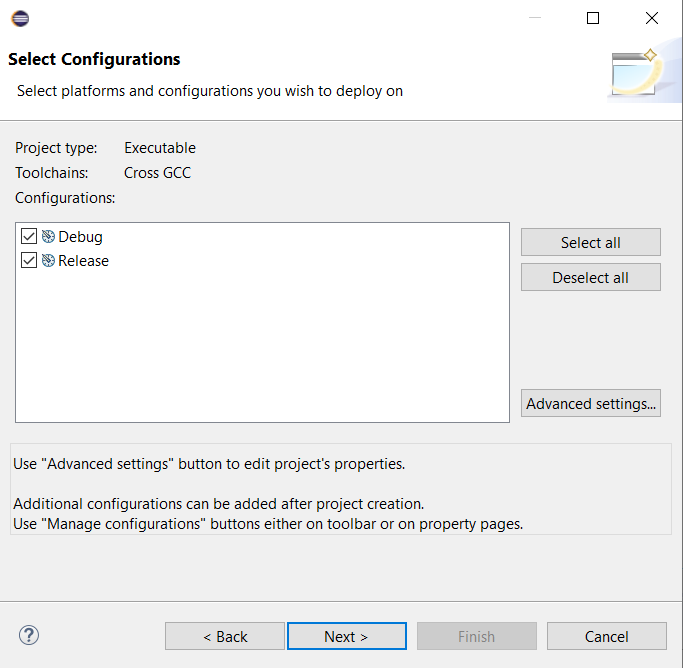
Configure Eclipse include/lib to folder

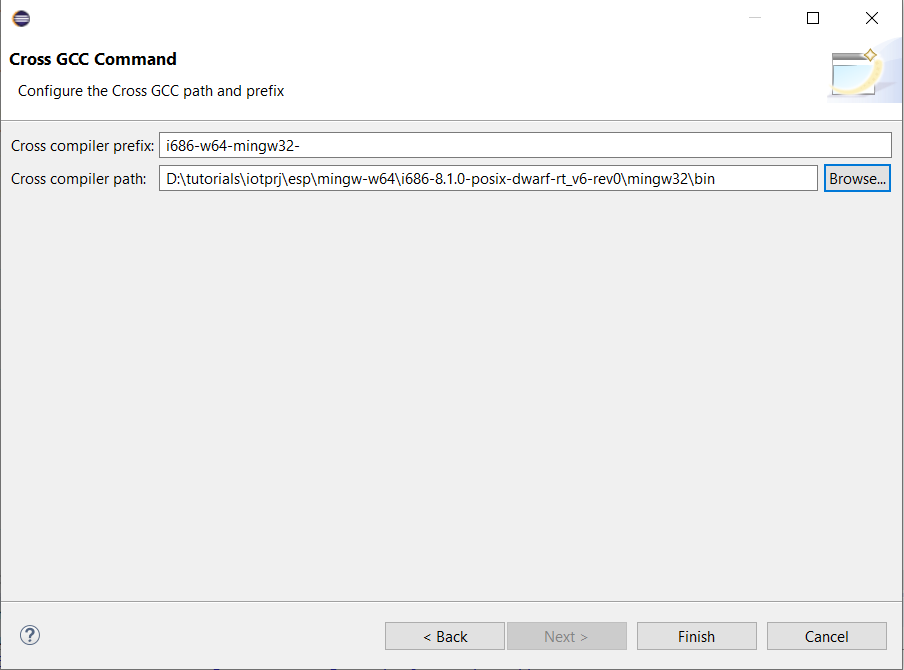


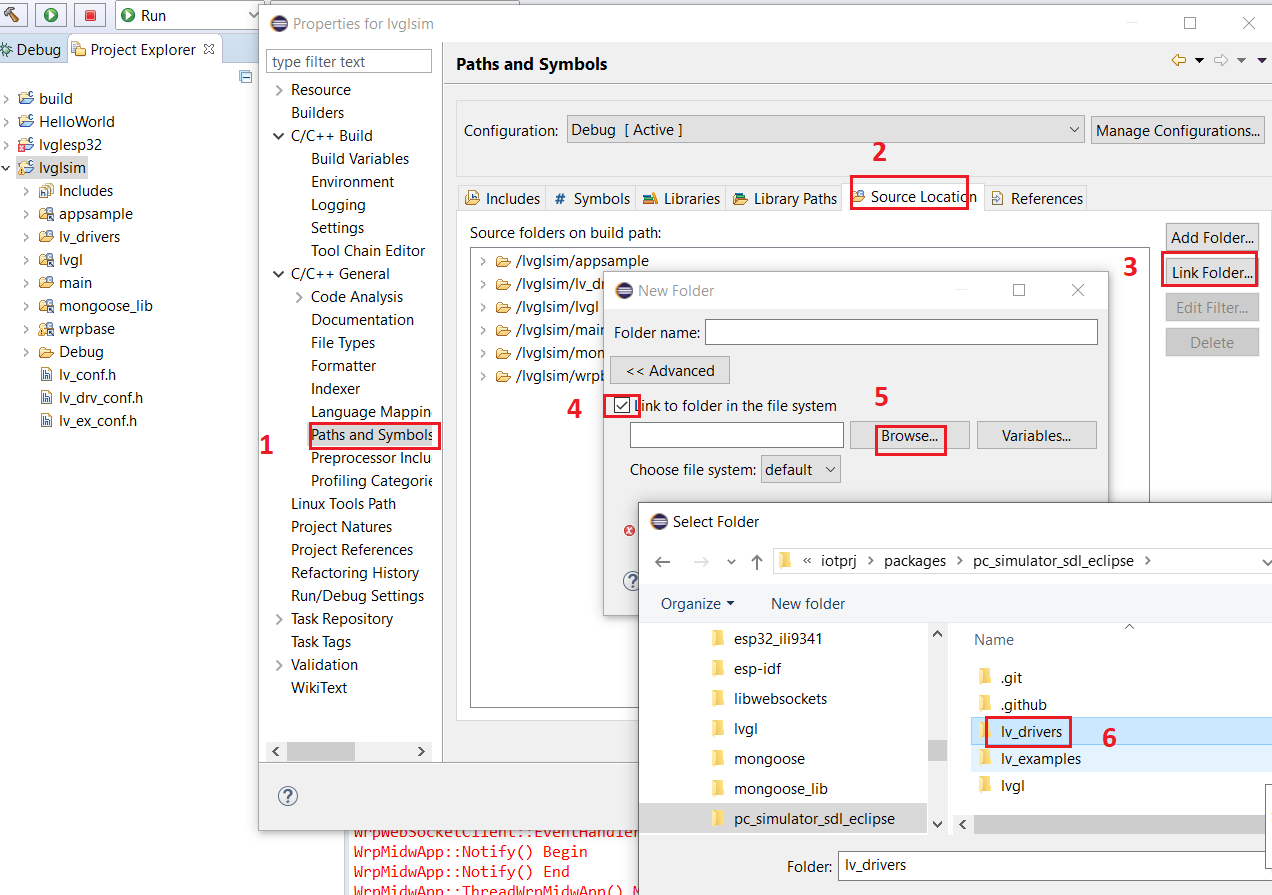




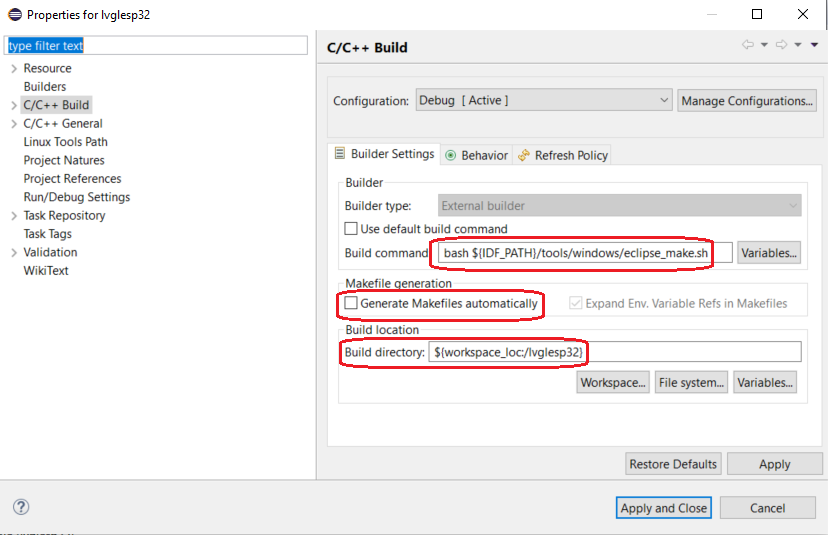


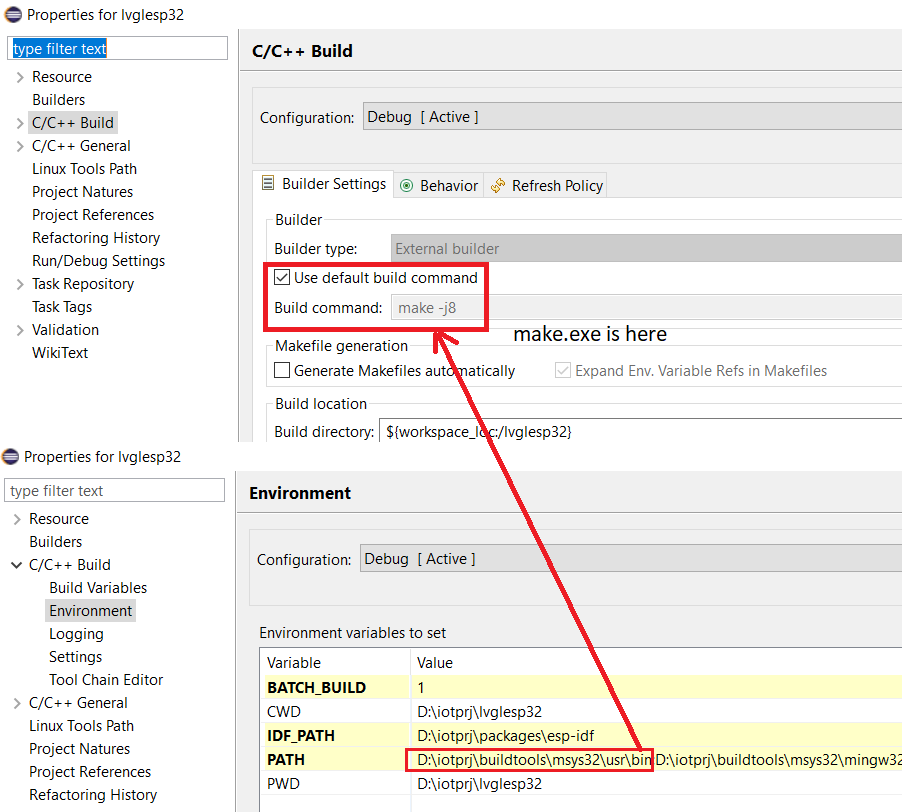


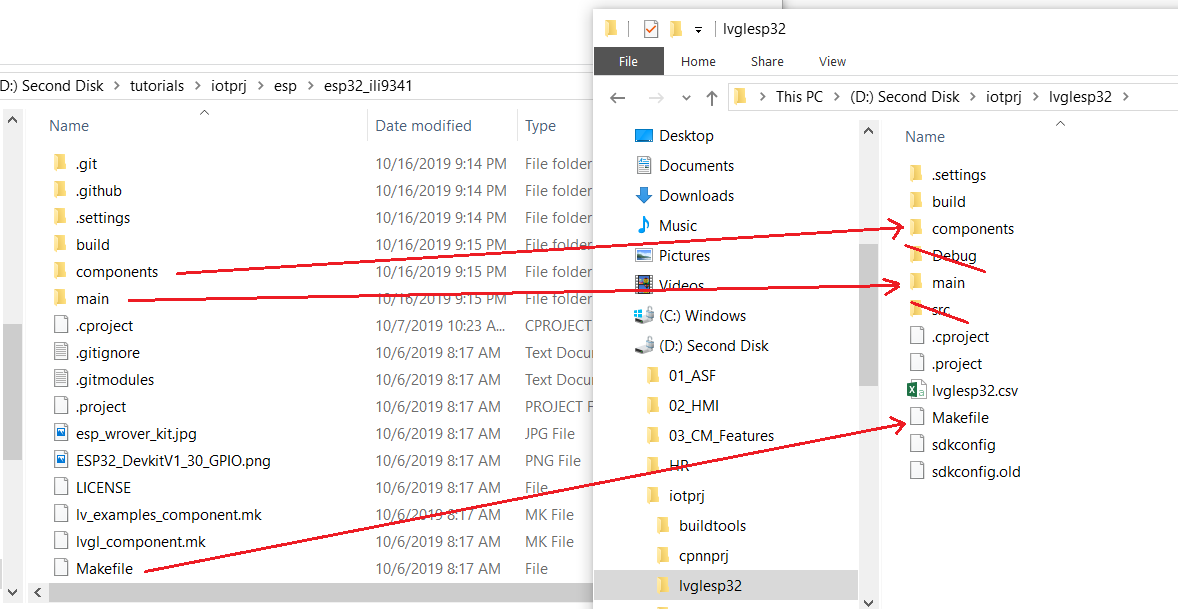




For ESP32

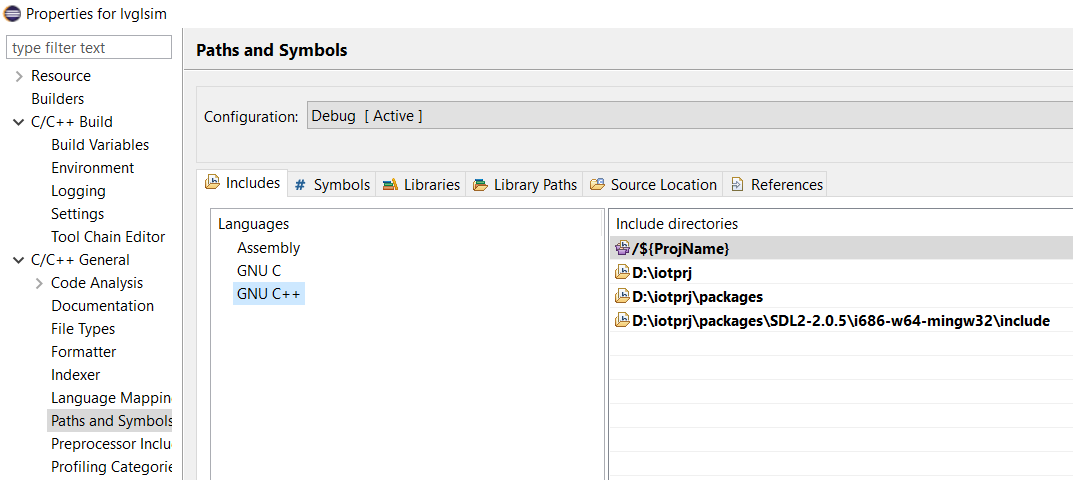






Copy components & main folders to lvglesp32

Copy Makefile to lvglesp32



MAKE file cua Eclipse va Make file cua ESP-IDF is different!!!

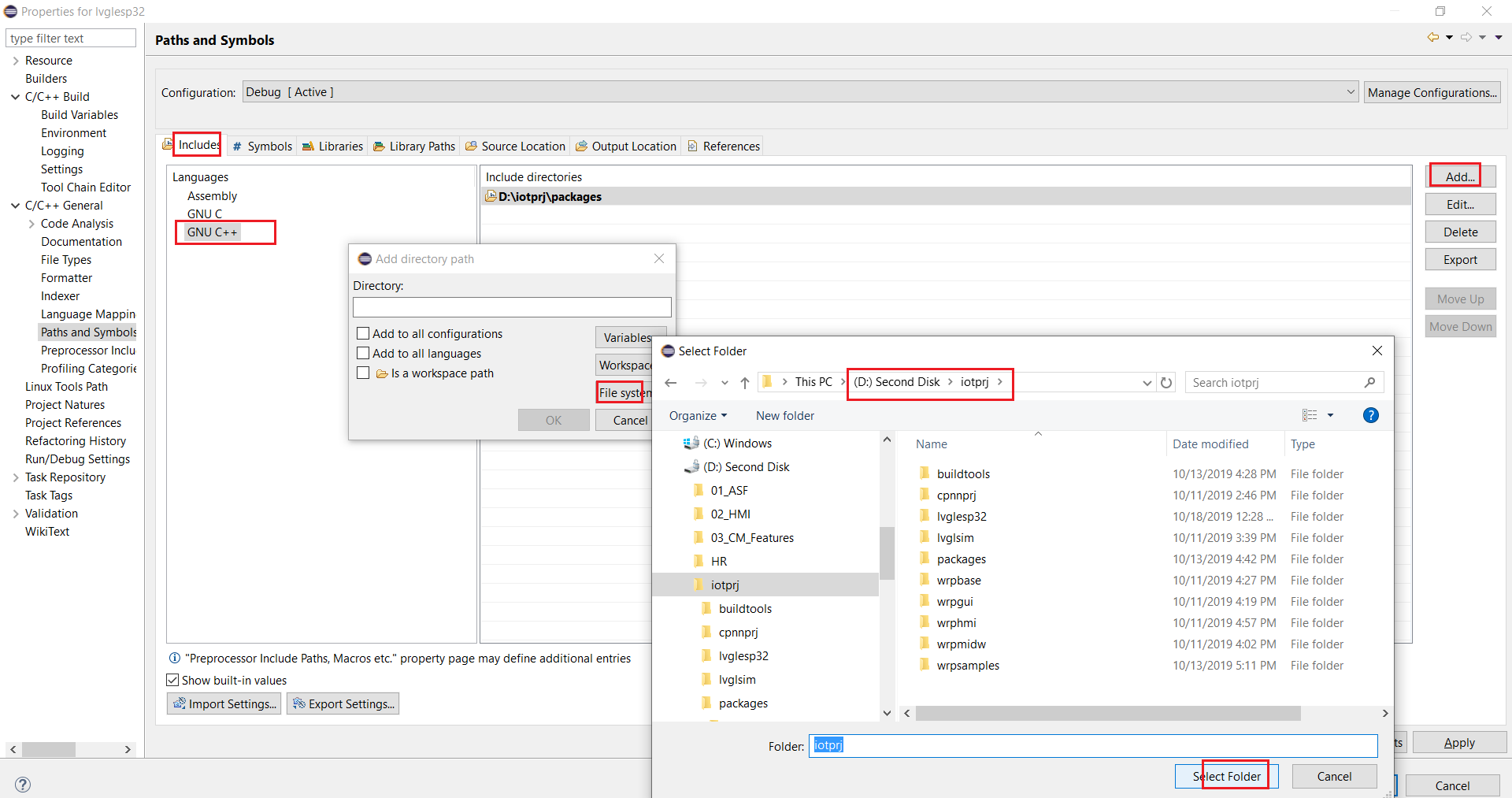
Now build packages/lvgl as component of ESP-IDF and use package/lvgl in project lvglesp32:

* + Modify D:\iotprj\packages\esp-idf\make\component\_wrapper.mk COMPONENT\_EXTRA\_INCLUDES ?= $(IDF\_PATH)/../ $(IDF\_PATH)/../esp32\_ili9341/components  
    (D:\iotprj\lvglesp32\components\drv can include lvgl/lvgl.h file)
  + Modify D:\iotprj\packages\esp-idf\make\project.mk to include lvgl: EXTRA\_COMPONENT\_DIRS ?= $IDF\_PATH/../lvgl  
    (build lvgl as new component of ESP IDF)
  + Modify D:\iotprj\packages\esp-idf\make\project.mk to include lv\_conf.h: CFLAGS ?= -DLV\_CONF\_INCLUDE\_SIMPLE

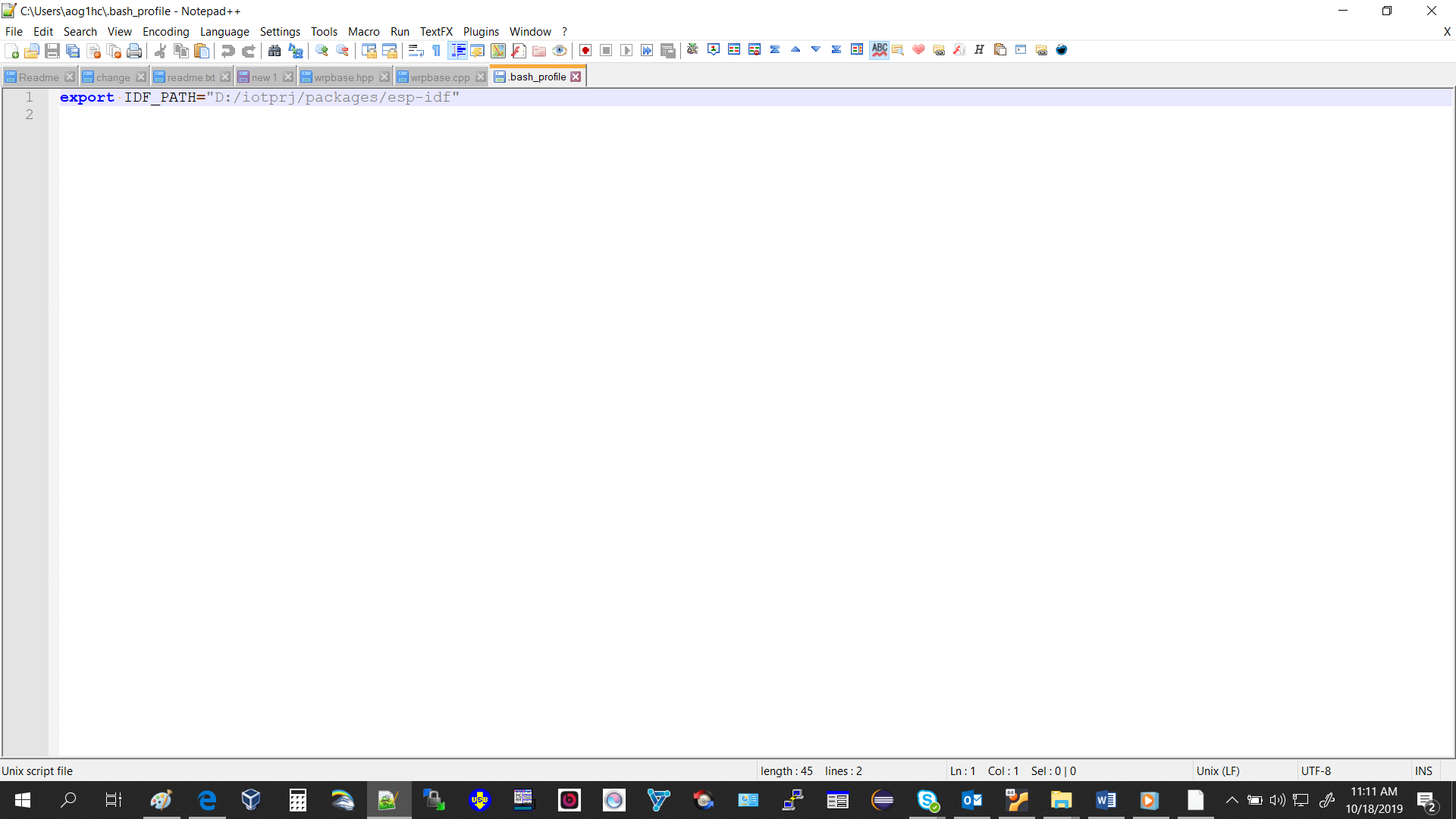
Copy iotprj\packages\esp32\_ili9341\components\drv to D:\iotprj\lvglesp32\components\drv

Copy iotprj\packages\esp32\_ili9341\components\lv\_conf.h, lv\_ex\_conf.h to D:\iotprj\lvglesp32\components\drv\lv\_conf.h, lv\_ex\_conf.h

Copy iotprj\packages\esp32\_ili9341\ lvgl\_component.mk to D:\iotprj\packages\lvgl\component.mk



IDF\_PATH for MSYS2 startup

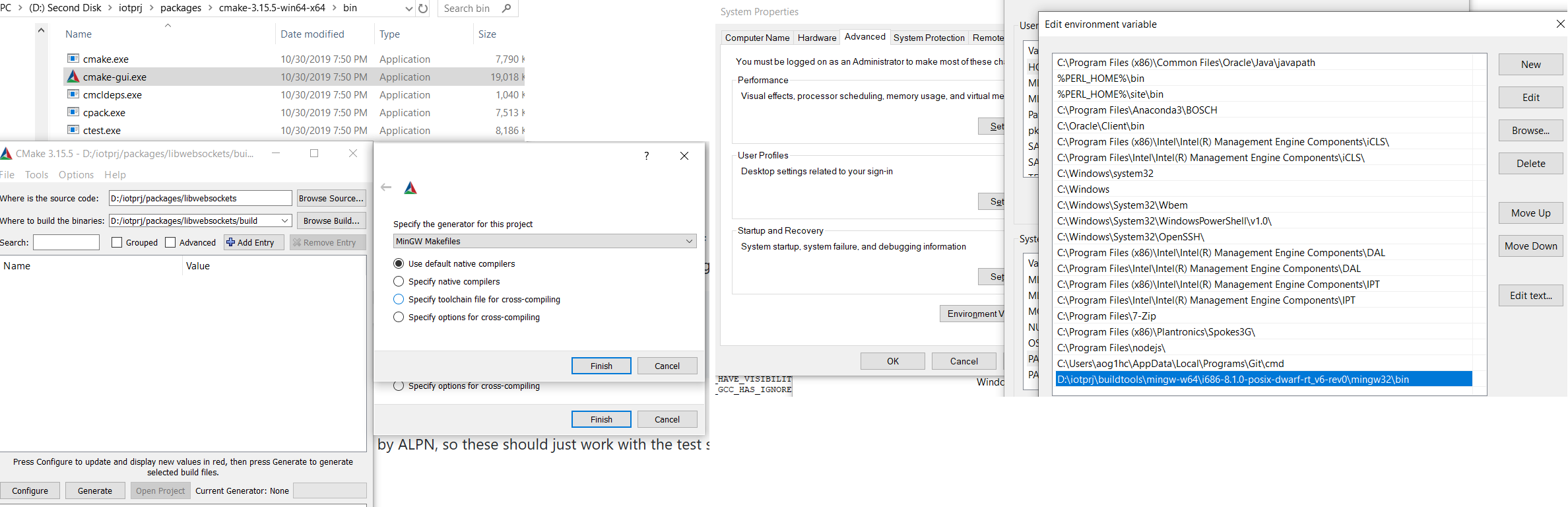


C:\Users\aog1hc\.bash\_profile -> export IDF\_PATH=”path”

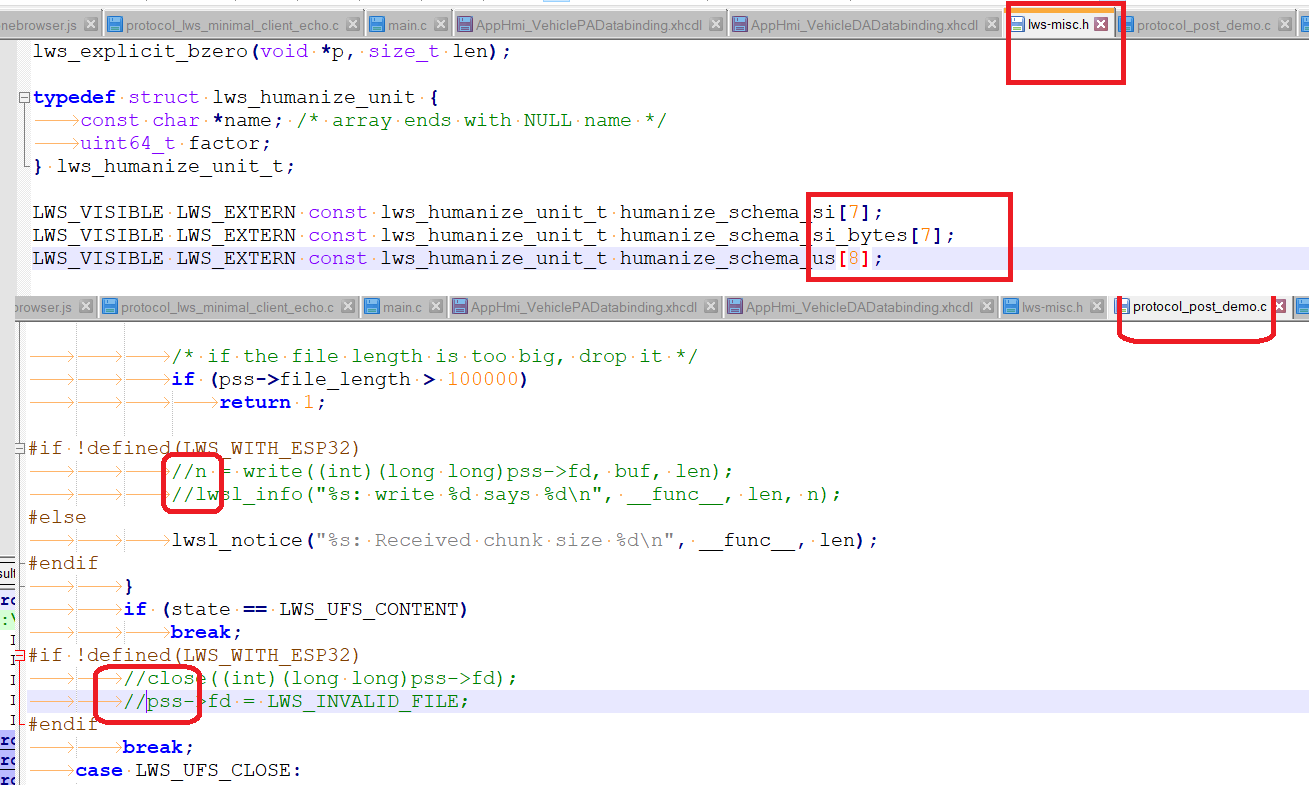
CMAKE to build libwebsocket

<https://cmake.org/download/>

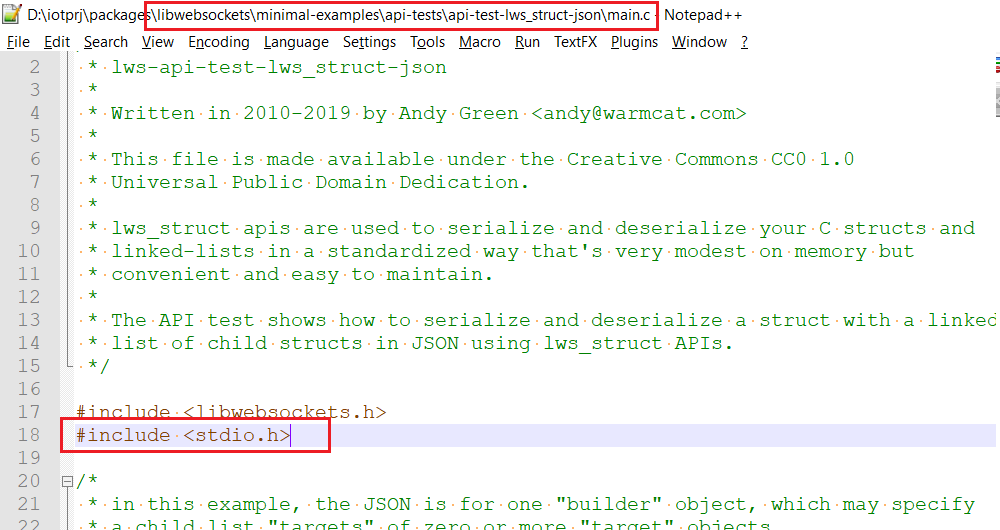
libwebsocket: git checkout v3.2-stable



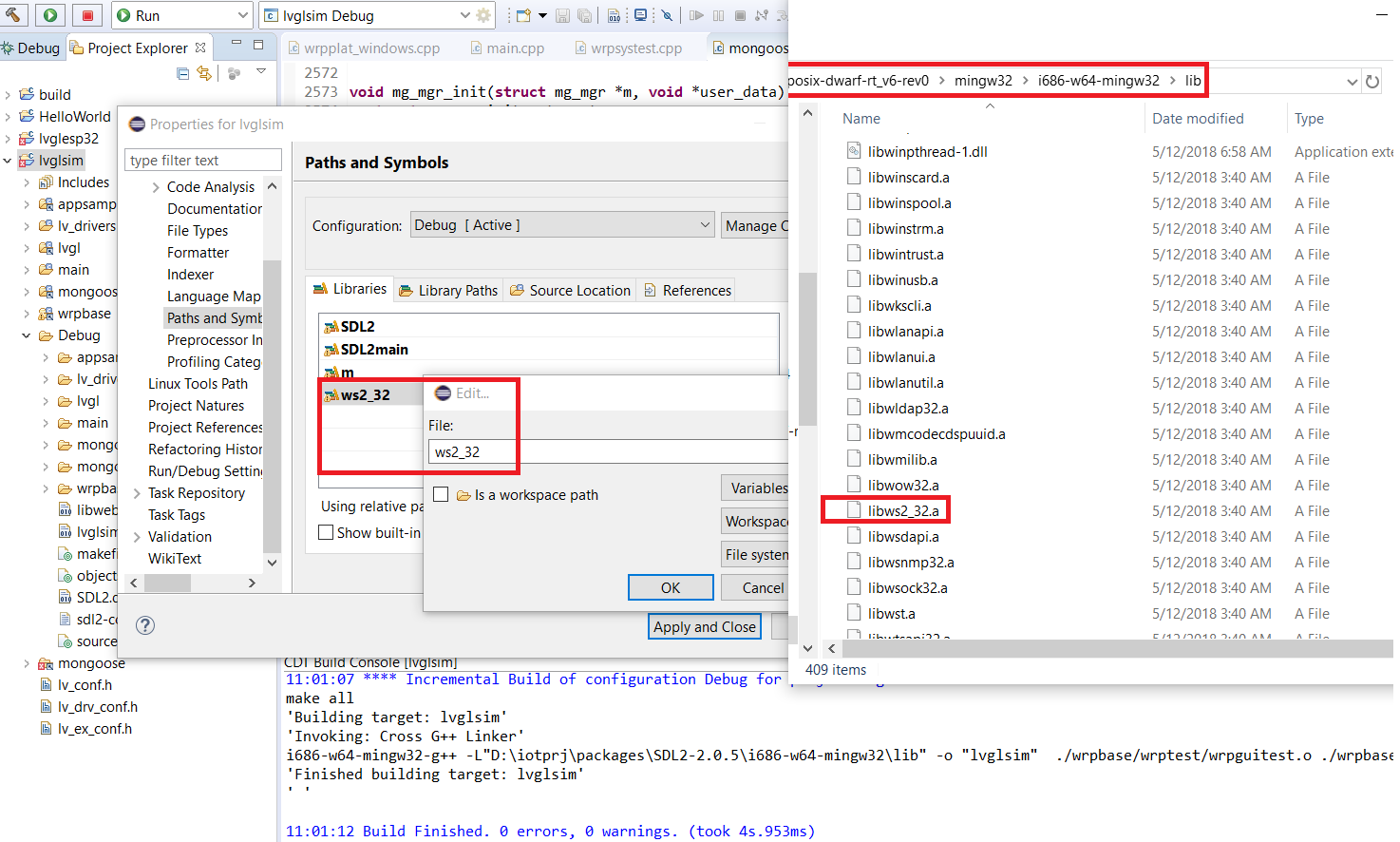
Uncheck LWS\_SSL\_CLIENT\_USE\_OS\_CA\_CERTS, LWS\_WITH\_SSL, LWS\_IPV6, LWS\_WITH\_MINIMAL\_EXAMPLES



MASTER:



Mongoose Websocket Lib:



Working with file binary on SDCard

gpio\_pullup\_en((gpio\_num\_t)12); // must pull up this resistor

sdmmc\_host\_t host = SDMMC\_HOST\_DEFAULT();

host.max\_freq\_khz = SDMMC\_FREQ\_PROBING;

host.flags = SDMMC\_HOST\_FLAG\_1BIT;

sdmmc\_slot\_config\_t slot\_config = SDMMC\_SLOT\_CONFIG\_DEFAULT();

// To use 1-line SD mode, uncomment the following line:

//slot\_config.width = 1;

// GPIOs 15, 2, 4, 12, 13 should have external 10k pull-ups.

// Internal pull-ups are not sufficient. However, enabling internal pull-ups

// does make a difference some boards, so we do that here.

gpio\_set\_pull\_mode((gpio\_num\_t)15, GPIO\_PULLUP\_ONLY); // CMD, needed in 4- and 1- line modes

gpio\_set\_pull\_mode((gpio\_num\_t)2, GPIO\_PULLUP\_ONLY); // D0, needed in 4- and 1-line modes

gpio\_set\_pull\_mode((gpio\_num\_t)4, GPIO\_PULLUP\_ONLY); // D1, needed in 4-line mode only

gpio\_set\_pull\_mode((gpio\_num\_t)12, GPIO\_PULLUP\_ONLY); // D2, needed in 4-line mode only

gpio\_set\_pull\_mode((gpio\_num\_t)13, GPIO\_PULLUP\_ONLY);