Versione N° 1.0

LaurTec

Impostare Eclipse per la scheda di sviluppo ESP32 e Arduino

Autore : *Paolo Salvagnini*

ID: UT0004-IT

AVVERTENZE

I progetti presentati non hanno la marcatura CE, quindi non possono essere utilizzati per scopi commerciali nella Comunità Economica Europea.

Chiunque decida di far uso delle nozioni riportate nella seguente opera o decida di realizzare i circuiti proposti, è tenuto pertanto a prestare la massima attenzione in osservanza alle normative in vigore sulla sicurezza.

L'autore declina ogni responsabilità per eventuali danni causati a persone, animali o cose derivante dall'utilizzo diretto o indiretto del materiale, dei dispositivi o del software presentati nella seguente opera.
Si fa inoltre presente che quanto riportato viene fornito così com'è, a solo scopo didattico e formativo, senza garanzia alcuna della sua correttezza.

L'autore ringrazia anticipatamente per la segnalazione di ogni errore.

Indice

Introduzione

Il tutto ha origine da un progetto che stavo facendo pensando di usare un ESP8266 e quando ho finito di disegnare uno schema elettrico quasi definitivo, ho scoperto che nonostante avessi usato il bus I2C per collegare gli expanders per le porte e controllassi il display e un chip ADC esterno via SPI non rimaneva nemmeno un pin per una qualsiasi altra eventualità. Ho quindi cercato alternative e trovato schede basate sul SoC (System on Chip) ESP32 e ho deciso che valeva la pena di provarlo. La prima domanda è stata: quale IDE posso adoperare per scrivere il programma? La prima risposta, la più banale, è stata l'IDE di Arduino che avevo già usato per ESP8266 e che prevedeva un plug-in anche per ESP32. Questa soluzione non era però il massimo della felicità perché veniva a limitare le possibilità di utilizzo dell'ESP32, infatti, alcune funzioni non sono ancora state integrate, i Timer sono quasi ingestibili in modo corretto ma soprattutto la filosofia di Arduino che prevede un loop infinito non si adatta a porre in sleep la CPU. Questa funzione diventa invece determinante qualora ci si voglia connettere alla rete saltuariamente. L'altra possibilità che mi si affacciava alla finestra era Eclipse, un IDE che già avevo usato con controllori TI e Freescale e che fornivano adattato per i loro prodotti. Per il SoC ESP32 è invece necessario impostare Eclipse al fine di poterlo utilizzare come ambiente di sviluppo dedicato. Vorrei precisare che non ho inventato nulla, ho solo perso una certa quantità di tempo a leggere e mettere in pratica ciò che era raccontato e che al solito nelle singole descrizioni, mancava sempre, almeno per me, un uno per fare trentuno. Vedrò quindi di esser quanto prolisso possibile, cosi dice il Mauro, sperando di non dimenticare nulla.

Scopo

Gli obiettivi che volevo raggiungere erano due. Il primo era di configurare Eclipse per essere usato quale IDE per Arduino e quindi potesse supportare oltre alle schede native di Arduino anche ESP8266 ed ESP32. Il secondo era di utilizzare lo stesso IDE, ma configurato in C/C++ per supportare ESP32 con gli strumenti del costruttore. Le possibilità di configurare il vostro PC sono molto varie, io vi racconterò la mia soluzione che prevedeva un unico vincolo quello di avere un workspace in documenti, la directory che contiene tutti i miei lavori, di cui faccio spesso un backup.

Cosa vi serve

- Un PC con sistema operativo Windows
- Una scheda tipo Arduino
- Una scheda con ESP8266
- Una scheda con ESP32

Due parole sul sistema operativo. La mia descrizione è fatta e testata su Windows 7, ma dovrebbe funzionare anche su Windows 10, almeno così si racconta. Per gli altri sistemi operativi la linea logica della sequenza delle operazioni è la stessa, ma cambiano le istruzioni specifiche. Per questi casi vi rimando alla bibliografia che costituisce la fonte di ciò che vi scrivo e da cui ho attinto a piene mani. Le tre schede citate sono necessarie per verificare se tutto funziona correttamente.

Come funziona

L'ambiente di sviluppo Eclipse non contiene le librerie e gli strumenti dei singoli costruttori di CPU e quindi affinché possa svolgere le sue funzioni, è necessario dirgli dove li può trovare. Il luogo più ovvio, se esiste, è l'IDE del costruttore. In altre parole, perché Eclipse possa funzionare, dobbiamo aver preventivamente installato gli IDE di Arduino e di Espressif che contengono i tools necessari. Procederemo quindi installando, come prima cosa, l'IDE di Arduino, che configureremo anche per la scheda ESP8266 ed ESP32. Con l'IDE di Arduino che funziona correttamente, andremo a installare Eclipse e il plug-in di Sloeber. Questo plug-in specifico per Arduino ricrea sulla barra degli strumenti delle icone simili a quelle dell'IDE di Arduino, facendo si che gli appassionati di Arduino si sentano in famiglia, ma certamente avendo a disposizione tutte le opzioni di Eclipse cioè di un IDE professionale. Coloro che non fossero interessati a lavorare in C/C++ con ESP32 potrebbero fermarsi qui. Il passo successivo è quello di installare il tool previsto da Espressif per ESP32 e al solito una volta che questo sia funzionante configurare Eclipse per utilizzare questi tools. È quindi evidente la necessità di avere a disposizione le tre schede, o almeno le due basate sul SoC ESP, al fine di verificare l'installazione del software.

La scheda Arduino

Andate sul sito di Arduino e scaricate e installate l'ultima versione dell'IDE. Se avete una vecchia versione già installata, aggiornatela. La configurazione di Arduino si limita alla definizione del percorso di dove avete installato la vostra cartella degli sketch. Per far questo cliccate su *File* e quindi *Impostazioni*. Nella parte superiore della finestra che vi si apre, è chiesto di definire il percorso della cartella degli sketch ad esempio:

```
C:\users\[user_name]\Documenti\Arduino sketch
```

Nella cartella degli sketch dovrete creare una cartella Librerie dove alloggerete le vostre librerie. Le librerie di Arduino sono, infatti, essenzialmente di due tipi. Il primo gruppo è quello che costituisce il core di Arduino che avete installato automaticamente con il programma e si trovano nella directory di Arduino. Il secondo è quello che si trova della cartella creata e cioè quello delle librerie private che avete usato o userete nei vari lavori.

Driver Seriale USB

Se non lo avete già fatto, installate il driver del convertitore seriale/USB della vostra scheda. Su tutte le mie, di origine cinese, è installato il CH340, ma non è una regola. I driver si trovano facilmente in rete. Installato il driver, connettete una scheda all'USB e verificate che nel pannello di controllo del vostro PC, in *Sistema* → *Gestione dei dispositivi*, compaia la porta COMx a cui è stata connessa la vostra scheda. Prendete nota del numero di COM e dell'USB usata.

Scheda ESP8266

In Figura 1 è riportato un dettaglio della scheda di sviluppo ESP8266 e la relativa piedinatura.

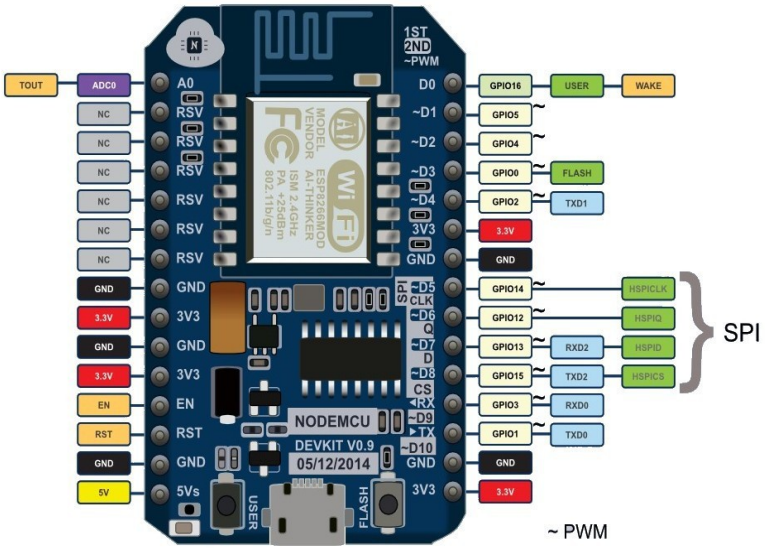

Figura 1: *Scheda di sviluppo ESP8266.*

Configuriamo ora l'IDE di Arduino per utilizzare il modulo ESP8266. Cliccate nuovamente in *File* e quindi *Impostazioni*, nella parte inferiore della videata nella riga a fianco a *URL aggiuntive per il Gestore schede*, copiate e incollate la riga sottostante:

```
http://Arduino.esp8266.com/stable/package_esp8266com_index.json
```

confermate e uscite.
Cliccate ora su *Strumenti* quindi nel Menù a tendina di *Scheda* cliccate su *Gestore Schede*. Comparirà un elenco di schede di cui generalmente l'ultima è definita come: esp8266 by ESP8266 community, cliccateci sopra e quindi installatela. Se tutto è andato a buon fine cliccando su *Strumenti* → *Scheda*, dovreste trovare elencate i tipi di schede più comuni per l'ESP8266. Provate ora a connettere una scheda e a programmarla con uno qualsiasi degli esempi che vi sono proposti. In questo modo vi assicurerete che il sistema funzioni correttamente. In seguito avremo bisogno di sapere dove sono stati installati i tools per la scheda ESP8266. Possiamo quindi verificare subito il percorso. Normalmente il PATH è il seguente:

```
C:\users\[users_name]\AppData\local\Arduino15
```

La cartella *AppData* è una cartella di sistema, quindi normalmente è nascosta, andate quindi in *Opzioni Cartella* nel pannello di controllo di Windows e attivate mostra tutte le cartelle. Prendete nota del percorso, che vi servirà in seguito per configurare Eclipse. Vedi anche:

```
https://github.com/esp8266/Arduino
```

Layout della scheda ESP32

In Figura 2 è mostrato il layout della scheda che contiene il SoC ESP32.

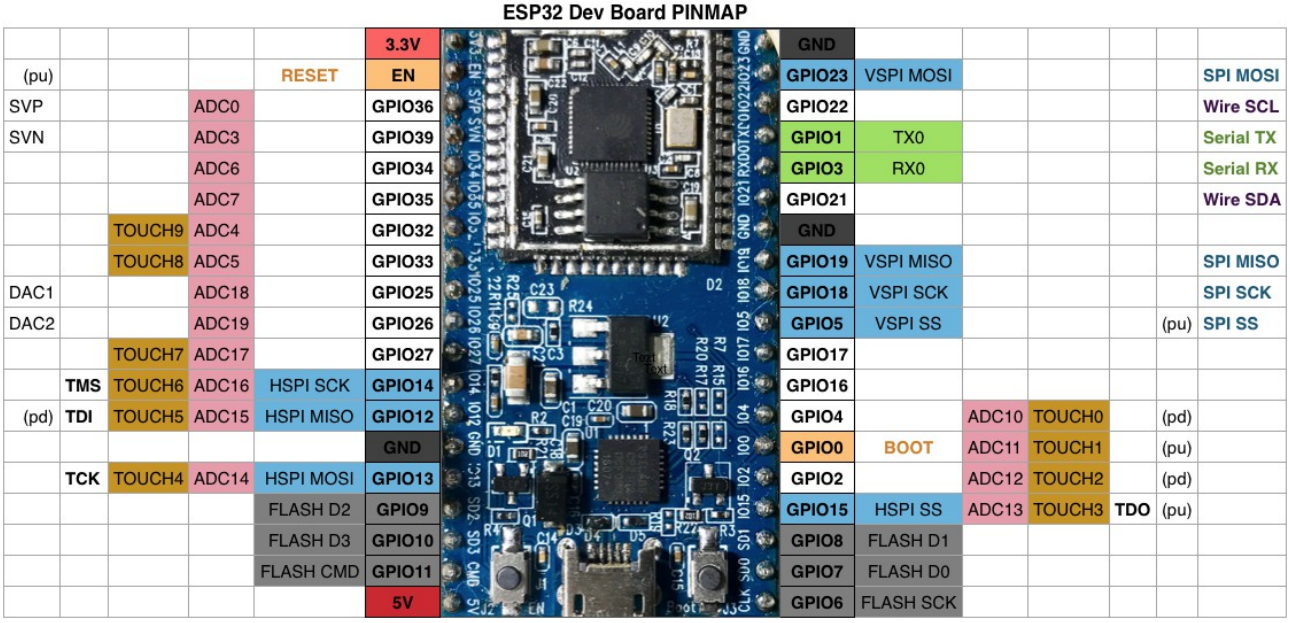

Figura 2: *Layout della scheda di sviluppo ESP32.*

La scheda di sviluppo ESP32 è da considerarsi la sorella maggiore di ESP8266 le cui caratteristiche ve le lascio scoprire dal datasheet, vi dico solo che ha una CPU Dual Core a 160MHZ semplicemente per istigarvi a pensare a cosa si potrebbe fare o cosa si faceva qualche anno fa con una simile CPU. Kolbman ha pubblicato un E-book per questa scheda che si può considerare un manuale, oltre ad una notevole quantità di video, che potete al solito trovare su YouTube e che vi consiglio di prendere in considerazione. Per l'installazione di questa scheda vi riporto quanto previsto al seguente indirizzo:

```
https://github.com/espressif/Arduino-esp32
```

Scaricate e installate Git da *git-scm.com*. Con questo programma eseguiremo il download dei tools di ESP32 di cui abbiamo bisogno. Lanciate Git GUI (Figura 3) ed eseguite le sottoelencate istruzioni. Selezionate *Clone Existing Repository*.

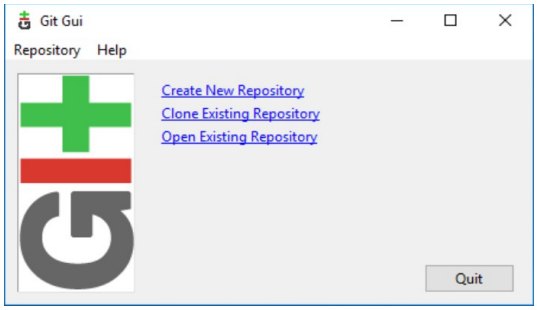

Figura 3: *Avvio di Git GUI.*

Selezionate la sorgente dei dati e la destinazione, *Source Location*:

```
https://github.com/espressif/Arduino-esp32.git
```

Target Directory:

```
C:/Users/[users_name ]/Documents/Arduino/hardware/espressif/esp32
```

Cliccate su *Clone* per iniziare il download.

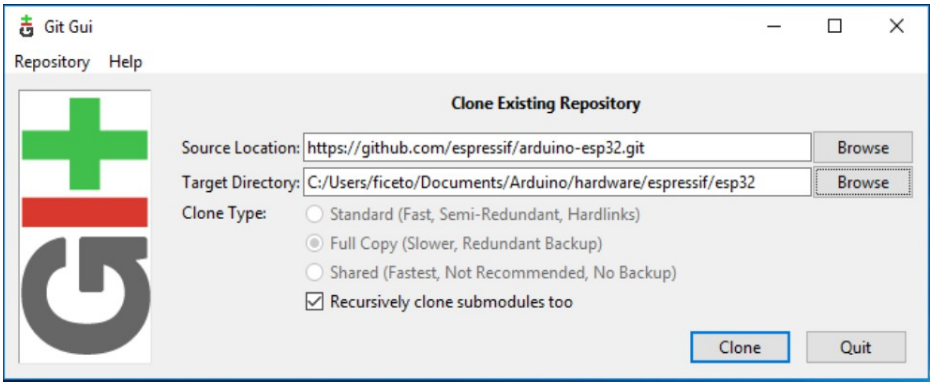

Figura 4: *Avvio del clone del repository.*

Aprite:

```
C:\Users/[user_name ]\Documents\Arduino\hardware\espressif\esp32\tools
```

Cliccate due volte su get.exe ed otterrete la finestra di Figura 5.

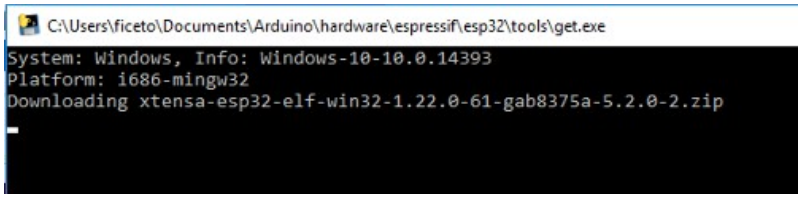

Figura 5: *Avvio del programma get.exe.*

A questo punto dovreste aver scaricato il plug-in di ESP32 per Arduino. Questo è costituito dalla cartella espressif definita nel percorso appena sopra. Purtroppo vi è un errore poiché la cartella espressif non va posta nella directory:

```
C:\Users\[user_name ]\Documents\Arduino\hardware
```

bensì nella cartella hardware del programma Arduino. Ad esempio, nel mio caso, ma reputo nella maggior parte dei casi il percorso è il seguente:

```
C:\Program Files (X86)\Arduino\hardware
```

Copiate quindi la cartella espressif e incollatela dentro la cartella hardware di quest'ultimo

percorso. Se ora lanciate l'IDE di Arduino cliccando su *Strumenti* e quindi *Scheda*, vi dovrebbero comparire le schede ESP32 supportate, se così non fosse fermatevi e verificate tutti i passaggi. Se invece la programmazione è andata a buon fine connettete una scheda ESP32, aprite un esempio e provate a compilarlo e a programmare la memoria flash. Se tutto funziona correttamente potete cancellare la directory prima creata:

```
C:\Users \[user_name] \Documents\Arduino\hardware
```

Ora possiamo installare Eclipse per programmare in stile Arduino.

Eclipse

Di Eclipse ne esistono molte versioni, con differenti nomi. A noi serve una versione che sia adatta al C/C++. Andare ora sul sito:

```
https://www.eclipse.org/downloads/packages/eclipse-ide-cc-
developers/neon3rc3
```

e scaricate la versione NEON3 a 32 0 64 Bit. Avrete così scaricato un file compresso che vi suggerisco di decomprimere in C:\. Effettuata tale operazione, verrà creata una cartella con il nome Eclipse, contenente l'eseguibile che non abbisogna di installazione. Createvi una cartella, dove riterrete opportuno, assegnandogli un nome senza spazi bianchi.
Questa cartella costituirà il vostro workspace. Lanciate Eclipse. All'apertura vi sarà chiesto di definire quale sarà la cartella di workspace, scegliete la cartella che avete creato. Dopo aver avviato Eclipse cliccando in alto a sinistra, chiudete la cartella di benvenuto. Vi troverete così di fronte alla schermata di default di Eclipse. Installiamo ora il plug-in per Arduino. Per far questo cliccate su *Help* → *Eclipse Marketplace*, nel riquadro che si aprirà, digitate Arduino. Vi apparirà la schermata di cui sotto e installate *The Arduino Ecipse plugin named Sloeber V4*, come mostrato in Figura 6. Dopo aver istallato il plug-in, Eclipse si presenterà così, naturalmente con il Project Explorer vuoto.

Come si può notare, sulla barra degli strumenti sono comparse delle icone che ripropongono le funzioni dell'IDE di Arduino.

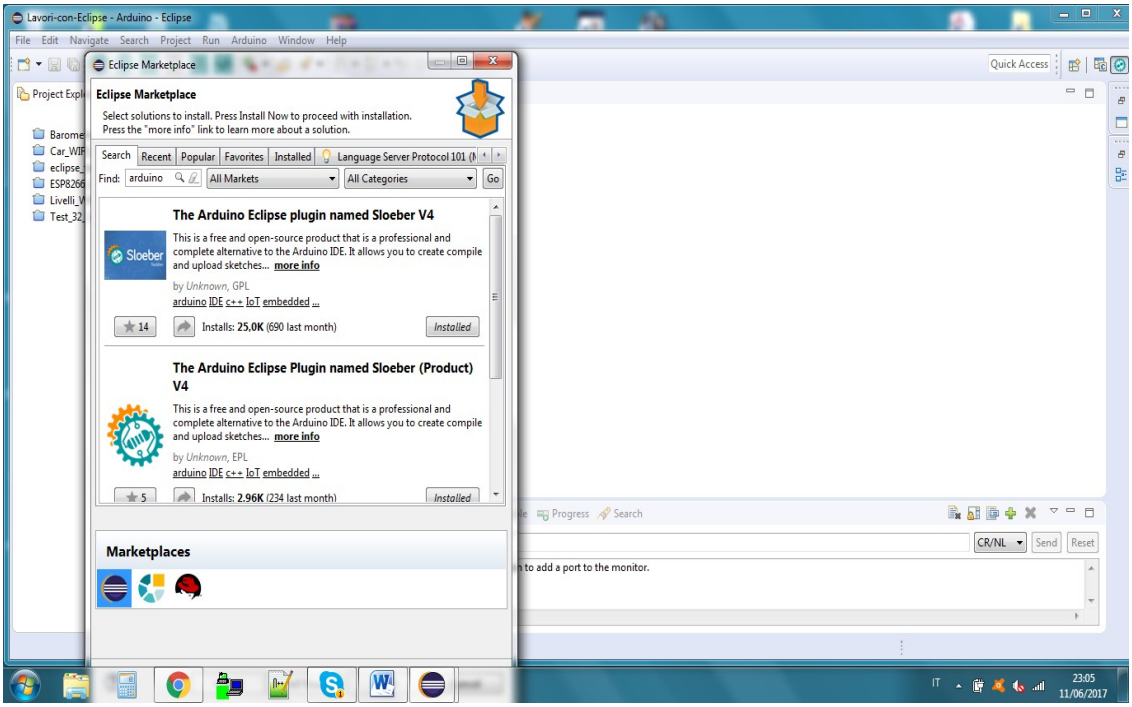

Figura 6: *Eclipse - Installazione dei plug-in aggiuntivi.*

Configuriamo Eclipse

Per poter configurare Eclipse bisogna impostare diversi percorsi affinché l'ambiente di sviluppo possa trovare le varie librerie installate. Per fare questo bisogna selezionare *Windows* → *Preferenze* e selezionare Arduino sulla finestra che sia apre (Figura 7). Nelle finestre *Private Library path* e *Private hardware path* rimuovete gli eventuali percorsi che non corrispondono alla vostra installazione, selezionandoli e poi cliccando su *Remove* quindi con *New* uno per volta aggiungete i percorsi corretti.

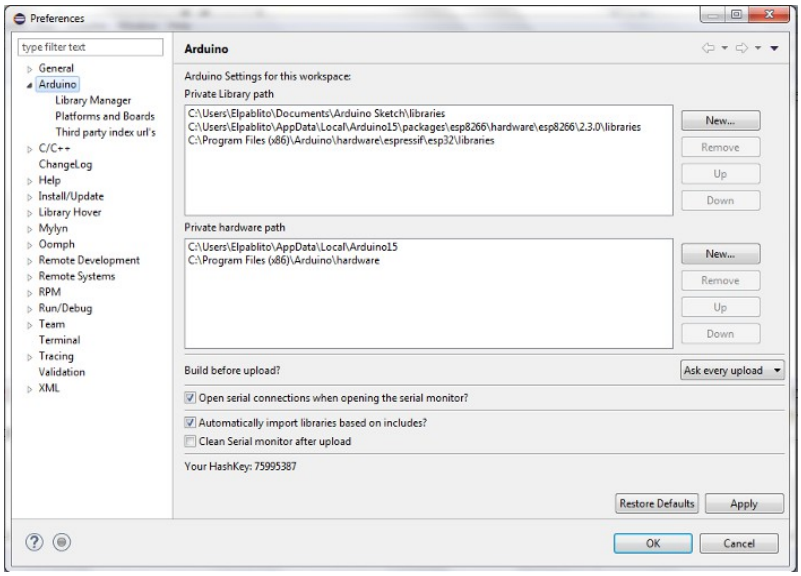

Figura 7: *Selezione dei percorsi per la scheda Arduino.*

Private library path

Il primo percorso è quello che vi serve per le schede di Arduino e definisce dove si trovano le vostre librerie, ad esempio:

```
C:\users\[user_name]\documenti\Arduino Sketch\libraries
```

Il secondo è quello che definisce dove si trovano le librerie della ESP8266, ad esempio:

```
C:\users\[UserName]\AppData\Local\Arduino15\pakages\esp8266\hardware\
esp8266\2.3.0\libraries
```

Il terzo è quello che definisce dove si trovano le librerie della ESP32, ad esempio:

```
C:\Program Files (x86)\Arduino\hardware\espressif\esp32\ libraries
```

Private hardware path

Il primo percorso identifica l'hardware di ESP8266, ad esempio:

```
C:\users\[user_name]\AppData\Local\Arduino15
```

Il secondo percorso è quello dell'ESP32, ad esempio:

```
C:\Program Files (x86)\Arduino\Hardware
```

Dopo aver definito dove si trovano i tools nel nostro PC, verifichiamo cosa ha installato di default il plug-in. Cliccate su *Third party index URL*'s e verificate di avere i tre indirizzi di Figura 8, nel caso aggiungete ciò che vi manca.

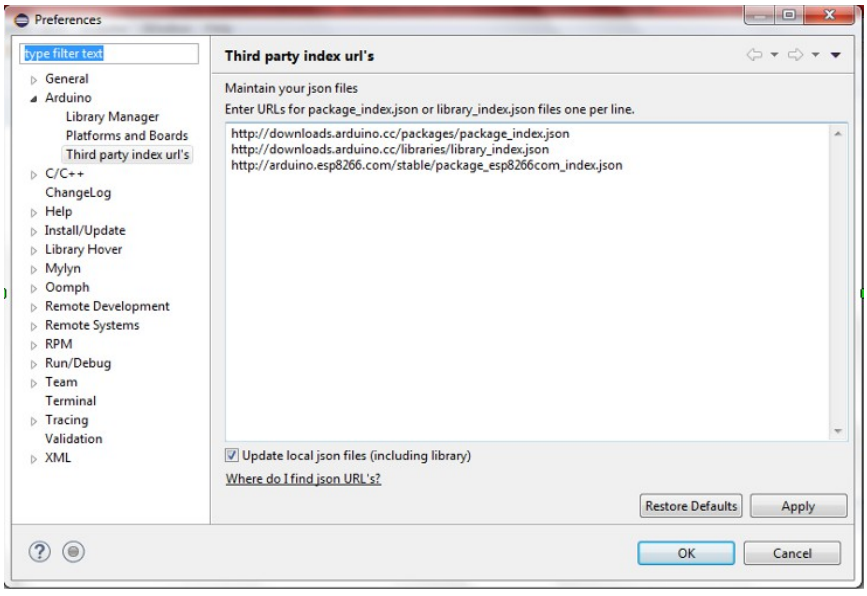

Figura 8: *Finestra Third party index.*

Andate poi in *Platforms and Boards*. Deselezionate *Hide 3th party, json files*, espandete i livelli, vi apparirà l'elenco degli hardware che potete usare. Selezionate e aggiungete ciò che vi serve.

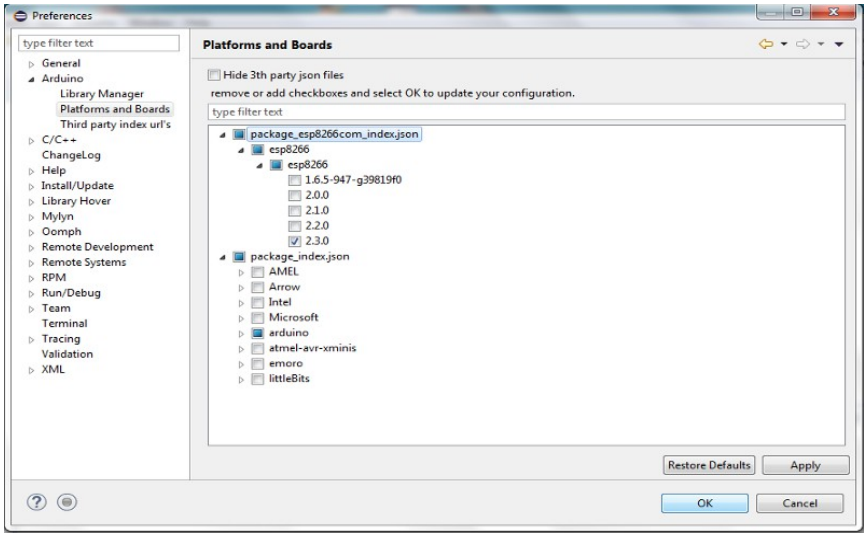

Figura 9: *Finestra di dialogo Platform and Boards.*

Al fine di rendere attive le nuove impostazioni riavviare l'ambiente di sviluppo Eclipse. Alla riapertura del programma, avendo finito di configurare Eclipse, possiamo al solito, verificare che tutto funzioni correttamente. Prima di fare questo test vi suggerisco, tuttavia, di far comparire sulla barra degli strumenti l'icona di Arduino, qualora non sia presente, che vi permetterà di passare velocemente dalla configurazione C/C++ a quella di Arduino e viceversa.
Cliccate su *Windows* → *Perspective*, quindi *Open Perspective* → *Others* . Cliccando su *Arduino* farete comparire sulla barra degli strumenti, in alto a destra, la corrispondente icona che vi permetterà di passare velocemente dalla configurazione C++ a quella di Arduino.

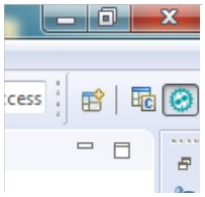

Figura 10: *Pulsante per passare dalla perspective Arduino a C/C++.*

La configurazione di Eclipse è finita e quindi verifichiamo se tutto funziona correttamente. Per fare questo suggerirei di rilanciare Eclipse, scegliere il workspace e chiudere la pagina di benvenuto. Sulla barra degli strumenti cliccare sull'icona che configura la presentazione per Arduino, poi cliccate sull'icona con il foglio che da luogo alla creazione di in nuovo sketch. Digitate il nome del progetto e premere *next*. Vi apparirà la finestra di Figura 11.

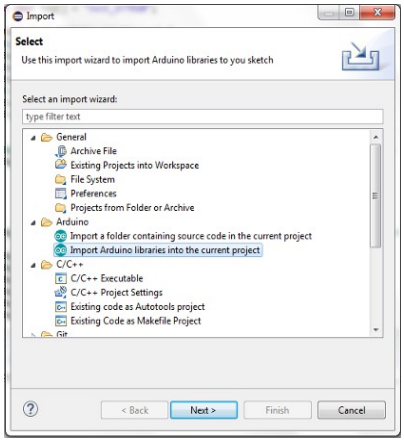

Figura 11: *Import di un progetto Arduino.*

Nella finestra dovete selezionare la *Platform Folder* con le caratteristiche della board che andrete a usare e la porta seriale che avevate annotato, poi cliccate su finish; avete creato uno sketch vuoto. In *Project Explorer* cliccate due volte sul nome del progetto per aprirlo e quindi sul file xxx.ino. Ciò che vi apparirà è estremamente simile a quanto sarebbe apparso con l'IDE di Arduino, con un'unica differenza, rappresentata dall'istruzione:

```
#include "Arduino.h"
```

con cui Eclipse include il core di Arduino nel nuovo progetto. Se avete fretta di provare e non volete lavorare, aprite uno sketch di esempio per la board che avete scelto, copiatelo e incollatelo nello sketch vuoto appena aperto cancellando tutto tranne `#include "Arduino.h"` che costituirà la prima istruzione. Compilate il nuovo sketch e programmate la memoria flash. Ricordatevi che per aggiungere librerie al vostro progetto dovete importarle quindi cliccate su *File* → *Import*. Allo stesso modo verificate di poter programmare correttamente i tre gruppi di schede cioè Arduino, ESP8266 e ESP32. Se non siete interessati alla programmazione dell'ESP32 in C/C++ fermatevi qui e incominciate a scoprire ciò che vi può offrire l'IDE di Eclipse.

Vedi anche:

```
https://www.youtube.com/watch?v=MDocFcK73_Q
```

EspressIF ESP-IDF

Per poter lavorare in C/C++ con Eclipse e i tools del costruttore, è necessario partire da un file preventivamente configurato in modo idoneo, è necessario utilizzare un file configurato con i tools del costruttore, prima di poterlo gestire con Eclipse. Il primo passo e quindi quello di scaricate i tools del costruttore che troverete all'indirizzo:

```
https://dl.espressif.com/dl/esp32_win32_msys2_environment_and_toolchain-20170330.zip
```

La versione originale di quanto andrò raccontandovi potete trovarla sul sito:

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

Decomprimete il file.zip che avete scaricato nella directory C:\ o in un'altra di vostro gradimento. Noi, per semplificare le impostazioni successive, seguiremo il suggerimento del costruttore, quello di decomprimere il file in C:\. Verrà così creata una directory con il nome msys32. Msys32 contiene tra le altre cose i tools per configurare, compilare e programmare la memoria flash di ESP. Dobbiamo ora scaricare le librerie specifiche della scheda ESP32.

- Create una nuova cartella in C:\ col nome esp.
- Aprite msys32 e lanciate mingw32.exe.

Vi comparirà la shell di Figura 12.

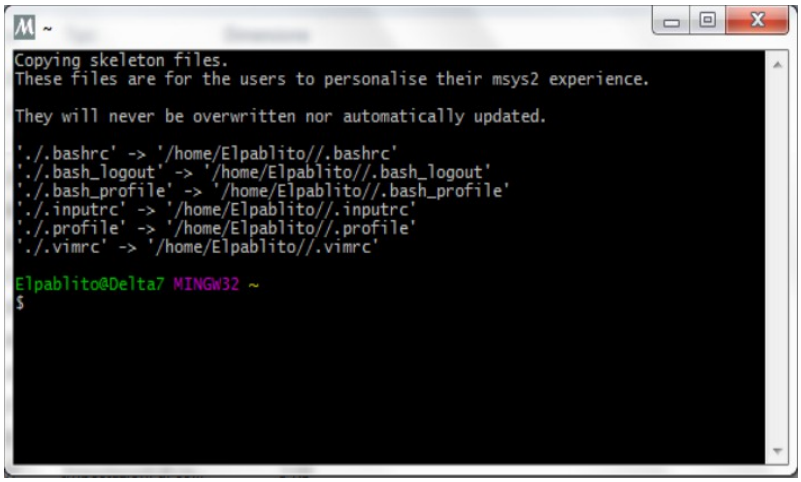

Figura 12: *Esecuzione del programma mingw32.exe.*

Digitare le istruzioni che seguono, verificando dalla shell che tutto procede correttamente

```
$ cd ..
$ cd ..
$ cd c:\esp
$ git clone --recursive https://github.com/espressif/esp-idf.git
```

Con le prime due istruzioni ritornate in radice C:\

La terza istruzione vi porta nella directory esp
Con la quarta scaricate le librerie. La schermata finale è come riportato in Figura 13.

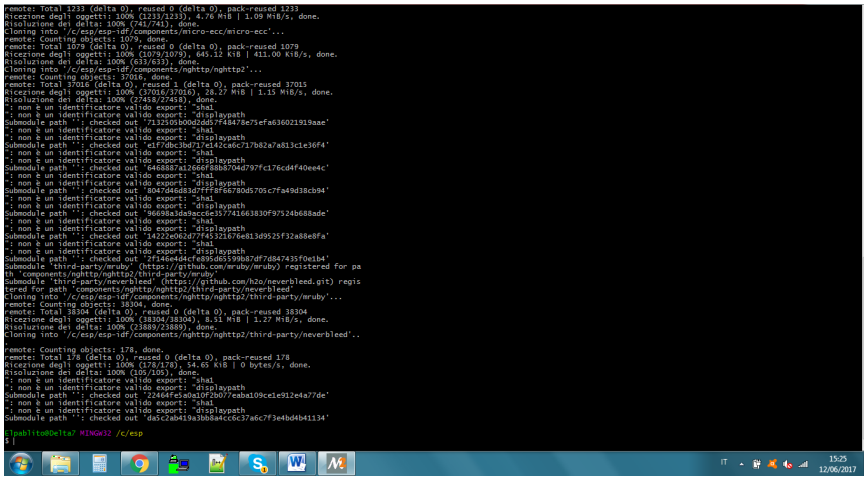

Figura 13: *Schermata ad installazione ultimata.*

Gli avvisi del tipo ": non è un identificatore valido export:" sono da ritenersi corretti.
La toolchain dei programmi si interfaccia con esp-idf tramite un'istruzione IDF_PATH che ne definisce il percorso e che sarebbe quindi necessario definire di volta in volta, cioè ridigitare. Per evitare che questo avvenga facciamo una piccola modifica come segue:

- Aprite un editor qualsiasi, es. notepad e aprite un nuovo file.
- Copiate e incollate la seguente istruzione: export IDF_PATH="C:/esp/esp-idf"
- Salvate il file con il nome: export_idf_path.sh
  nella cartella C:\msys32\etc\profile.d
- Chiudete la shell mingw32 e riapritela. Verificate se la modifica è corretta.
- Digitate $ printenv IDF_PATH. La shell vi deve restituire il percorso corretto e quindi C:/esp/esp-idf.

Digitate ora

```
$ cd ..
$ cd ..
$ cd c:\esp
$ git clone https://github.com/espressif/esp-idf-template.git myapp
```

Con le prime tre istruzioni vi portate nella directory esp dove create una cartella a nome myapp che contiene un'applicazione di esempio. In effetti viene creata una cartella strutturata in modo tale da poter gestire il file di esempio contenuto. Infatti, come vedremo, questa cartella costituisce anche l'elemento base per poter lavorare con Eclipse. Per le ragioni che vi saranno chiare nel seguito. Prima di proseguire vi invito a farne una copia e di salvarla magari cambiandole nome es. Cartella-base. Salvata la cartella, proseguite digitando:

```
$ cd myapp
$ make menuconfig
```

Vi si aprirà la finestra di Figura 14, con cui andate a configurare il vostro harware.

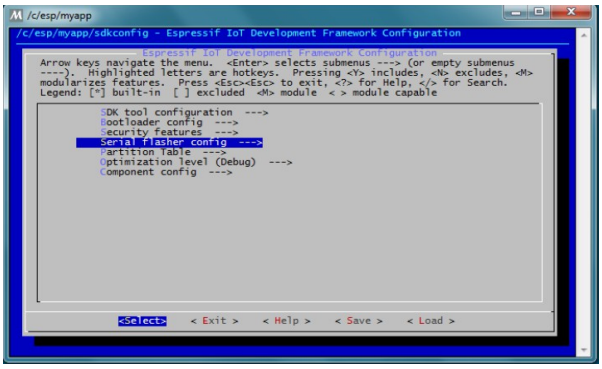

Figura 14: *Configurazione dell'Hardware.*

Spostatevi con le freccette su serial Flasher config, e con enter andate alla finestra successiva, riportata in Figura 15.

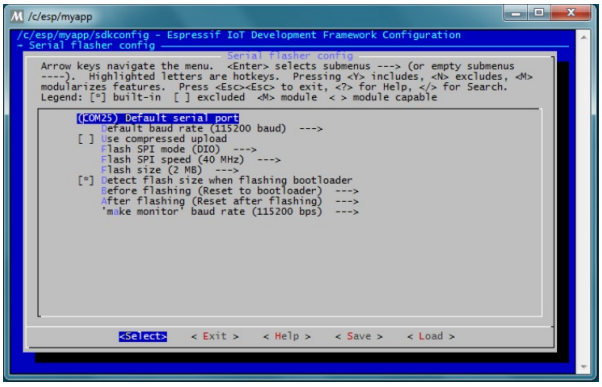

Figura 15: *Flasher Config* .

Modificate il Default serial port digitando la porta COMxx a cui è, o sarà connessa, la vostra ESP32 (Figura 16).

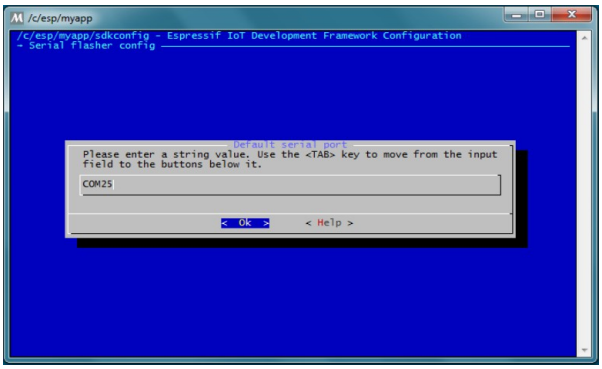

Figura 16: *Selezione della porta seriale.*

Confermate con OK, poi con exit fino a ritornare al quadro iniziale, salvate e uscite. Avete creato una nuova configurazione per ESP. Questa modifica ha l'unico scopo di

poter programmare la memoria flash per mezzo dell'adattatore UBS, cosa che faremo al passo successivo. Per le altre opzioni di configurazione non posso far altro che rimandarvi alla documentazione ufficiale, vista la quantità di opzioni possibili.
Se non lo abbiamo ancora fatto, colleghiamo il modulo ESP32 alla porta USB corrispondente alla COM che abbiamo definito nella configurazione, e digitiamo:

```
$ make
```

con cui compileremo il progetto myapp (Figura 17)

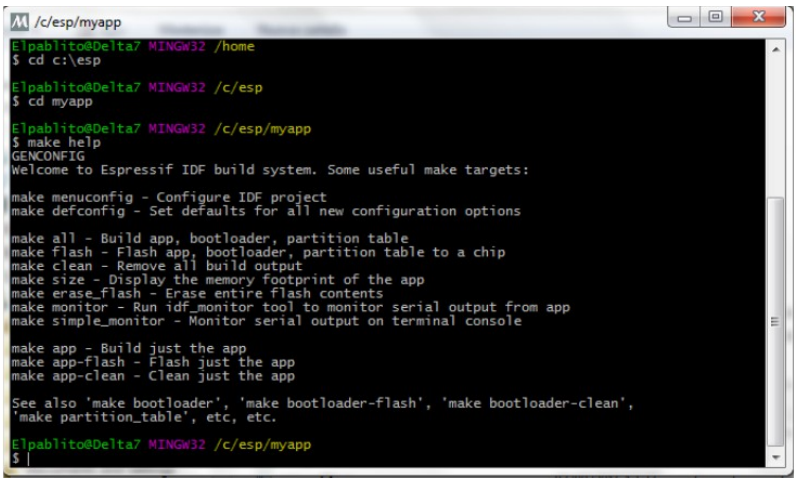

Figura 17: *Compilazione del progetto myapp.*

e quindi

```
$ make flash
```

per scrivere il codice compilato nella memoria flash del dispositivo. Per alcune board bisogna premere nell'ordine il pulsante RST poi quello FLASH e quindi rilasciare RST. Quindi tenendo premuto il pulsante di FLASH premere invio quindi rilasciarlo. Per altre, tipo la mia *easy kit esp-32* b1, ma non è l'unica, non è necessaria questa procedura.
Attendete la fine dell'operazione, che impiegherà un poco di tempo, poi verificate che il processo di programmazione sia andato a buon fine.
Abbiamo così fatto il minimo indispensabile per poter passare all'ultima fase quella della configurazione di Eclipse.

Vedi anche:

```
https://espressif.com/en/products/hardware/esp32/overview
http://esp-idf.readthedocs.io/en/latest/get-started/index.html#
https://www.youtube.com/watch?v=6YMSeGxJlRc
```

Eclipse C/C++ e ESP32

Anche questa volta comincio con il presentarvi il documento da cui ho tratto le informazioni per quello che andrò raccontandovi, lo trovate all'indirizzo:

```
https://github.com/espressif/esp-idf/blob/master/docs/get-started/eclipse-setup.rst#id1
```

Diciamo subito che Eclipse non può configurare il modulo ESP32 e sarà quindi necessario farlo con il programma del costruttore. Ciò che faremo come esempio è anche ciò che è necessario fare per creare un file nuovo. Mi spiego meglio, per creare un nuovo file è necessario avere un file configurato opportunamente da caricare in Eclipse, poi nessuno ci vieta di svuotare il main esistente e di scrivere il nostro nuovo programma. Supponiamo di voler creare un progetto con il nome test1_esp32. Copiamo quella cartella che vi avevo detto di salvare nello workspace di Neon3 e rinominiamola in test1_esp32. Usciamo da Eclipse e configuriamo il progetto secondo la necessità.
Per far ciò apriamo la shell MINGW32.exe, come abbiamo fatto prima, e ci porteremo nella cartella test1_esp32.
Lanciamo *make menuconfig* e apportiamo le modifiche del caso. Abbiamo cosi il nostro progetto configurato come un progetto ESP_IDF pronto per essere importato in Eclipse con un nuovo flie sdkconfig (quello preesistente è stato rinominato in sdkconfig.old).
Verificate l'esistenza del file eclipse_make,py in

```
C:\esp\esp-idf\tools\windows\
```

qualora non fosse presente copiatelo e incollatelo.

Lanciate Eclipse e scegliete *File* → *New* → *C++Project* (Figura 18)

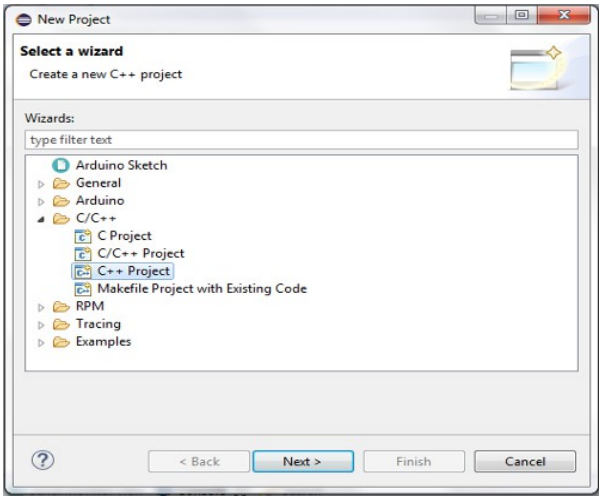

Figura 18: *Creazione di un nuovo progetto.*

Nella finestra che si aprirà scegliete *Makefile Project* e quindi *Empty Project* selezionando *Cross GCC*.
Il *Project Name* dovrà essere rigorosamente quello che avete assegnato alla cartella copiata, nel nostro caso test1_esp32. Nella parte alta del riquadro vi comparirà l'avviso *Directory*

with specified name already exist, come mostrato in Figura 19.

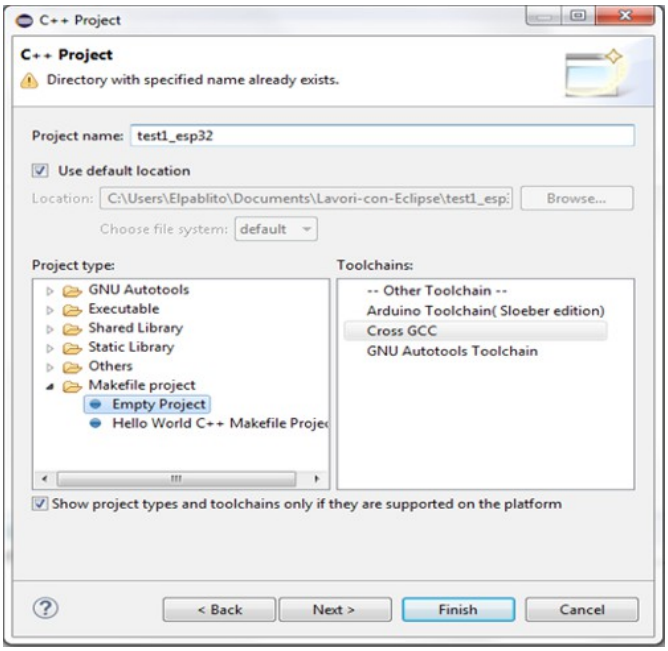

Figura 19: *Messaggio di avviso "Directory with specified name already exist".*

Proseguite con *Finish*. In Project Explorer vi comparirà il nuovo progetto. Clicchiamo due volte sul progetto per aprirlo. Aperto il progetto clicchiamo su *Project* → *Properties*. Nella pagina delle proprietà clicchiamo su C/C++ build. Togliamo il flag a *Use default build command*. In build command, al posto di Make, inserite:

```
python ${IDF_PATH}/tools/windows/eclipse_make.py
```

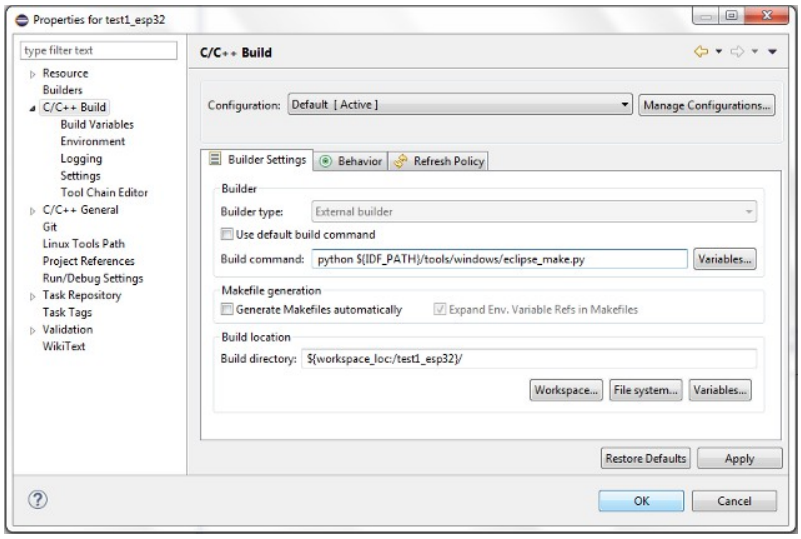

Figura 20: *Proprietà del progetto.*

Dobbiamo ora definire i percorsi dei tools

- Espandiamo C/C++ BUILD e apriamo *Enviroment*.

- Clicchiamo su *Add*, quindi su *Name*, inseriamo BATCH_BUILD e su *Value* il valore1.
- Clicchiamo su *Add* e inseriamo il *Name* IDF_PATH. Il percorso, nel caso abbiate seguito i suggerimenti, è :

```
C:/esp/esp-idf
```

e va scritto in *Value* con gli slash come mostrato (C:\esp\esp-idf è sbagliato).

- Clicchiamo sulla *Variable* PATH per selezionarla, quindi su *Edit* e sostituiamo il percorso esistente con il seguente:

```
C:\msys32\usr\bin;C:\msys32\mingw32\bin;C:\msys32\opt\xtensa-esp32-elf\bin
```

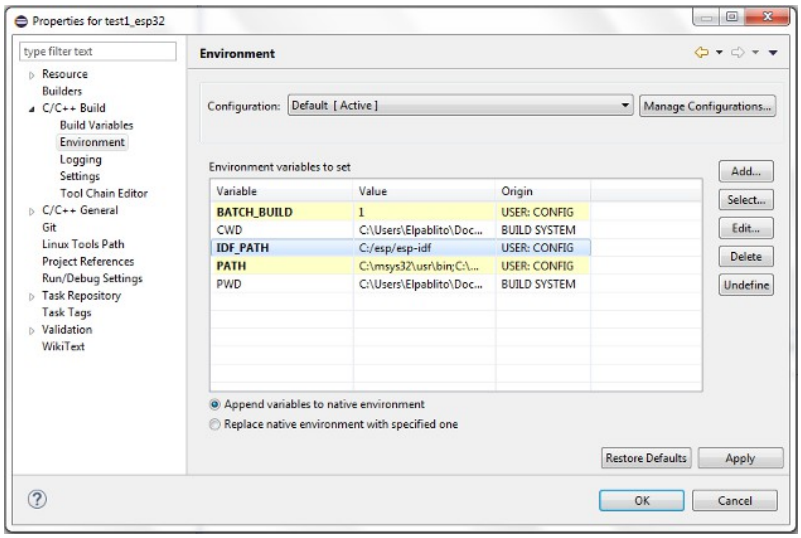

Figura 21: *Cambio dei percorsi Environment.*

Clicchiamo su *C/C++ General* e espandiamo la struttura:

- Clicchiamo su *Preprocessor Include Path*, *Macros*, etc.
- Clicchiamo sul *Tab Providers* nella lista dei *providers*, clicchiamo su *CDT Cross GCC Built-in Compiler Setting*.

Nella riga sotto *Command to get compiler spec* sostituite $(command) con

```
xtensa-esp32-elf-gcc
```

così da ottenere il comando completo

```
xtensa-esp32-elf-gcc ${FLAGS} -E -P -v -dD "${INPUTS}"
```

Nella stessa lista, clicchiamo ora su *CDT GCC Build Output Parser* e al comando *Compiler command pattern* aggiungete all'inizio :

```
xtensa-esp32-elf
```

Dovrete avere un comando completo del tipo:

```
xtensa-esp32-elf-(g?cc)|([gc]\+\+)|(clang)
```

come mostrato in Figura 23.

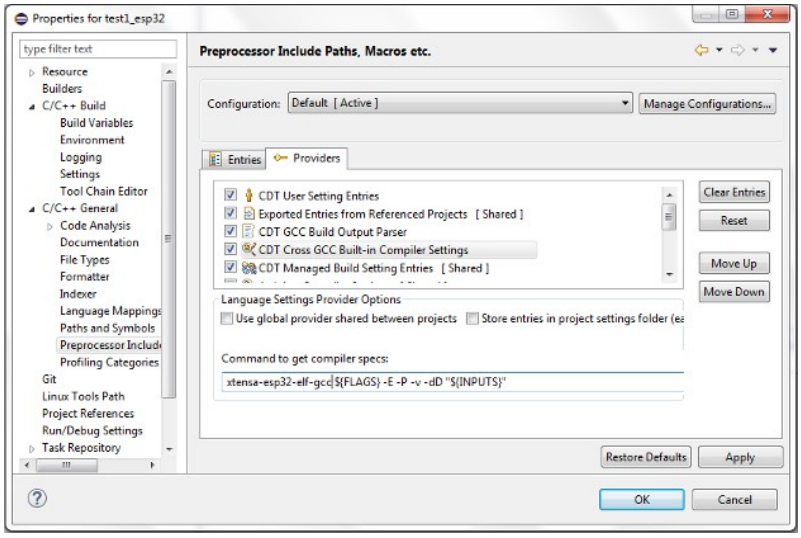

Figura 22: *Processors Include Path, Macros etc (CDT Cross GCC).*

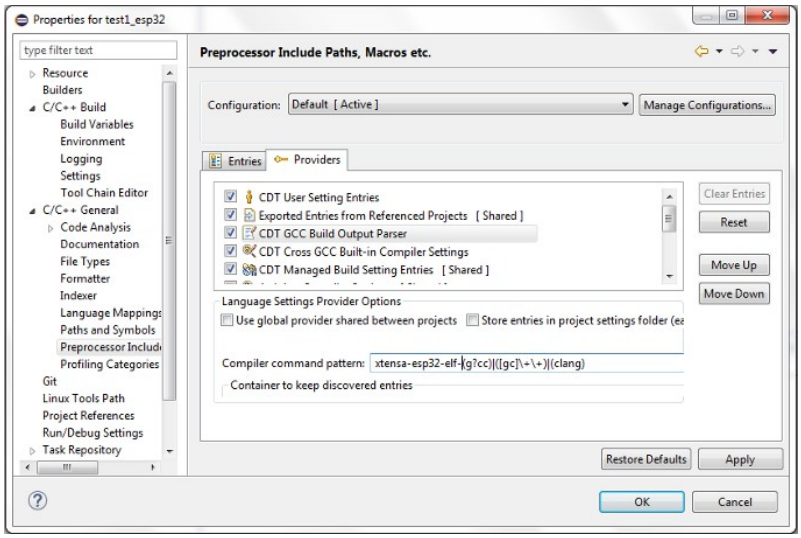

Figura 23: *Processors Include Path, Macros etc (CDT GCC Build Output Parser).*

Dobbiamo ora definire i percorsi per includere le librerie che ci servono.

- Clicchiamo su *Path and Symbols*. Nella finestra che si apre selezionate GNU e inseriamo cliccando su *Add* i percorsi degli include del nostro programma.
- Cliccate sul pulsante *Import Setting* poi con *Browse* selezionate il file …\c_incude2.xml quindi *Finish* .

Se, in seguito ad una modifica degli include, vi salvate la lista dei percorsi con *Export Setting*, vi esonerate dalla fatica di doverli riscrivere.

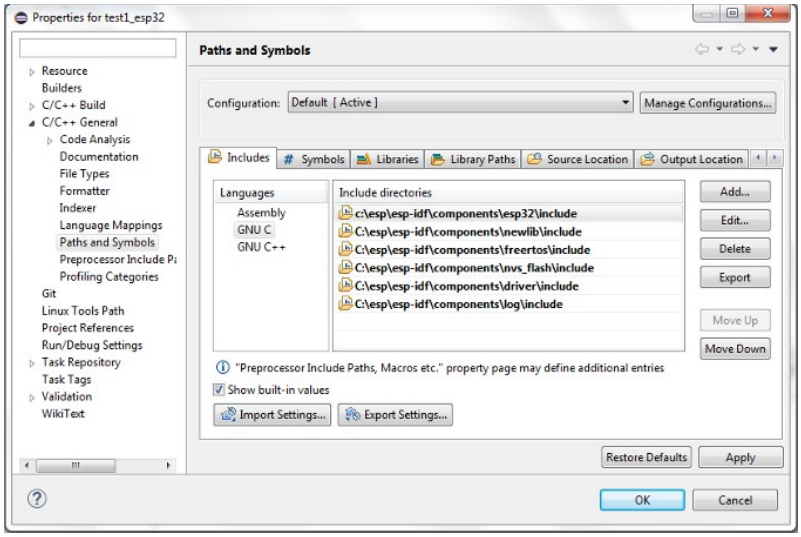

Figura 24: *Path and Symbols.*

A questo punto il vostro progetto, dovrebbe presentarsi come si può vedere più sotto. Sono scomparsi i simboli di errore dalla lista degli include, ciò sta a significare che sono stati riconosciuti i percorsi delle librerie. Rimangono ancora una serie di errori sulle istruzioni del programma.
In *Project Explorer*, clicchiamo col destro sul programma (test1-esp32-py) e scegliamo *Index* e quindi *Rebuild*. Così facendo dovrebbero scomparire tutti gli errori e si può quindi procedere con la compilazione.

In effetti, io ho notato un comportamento anomalo che segnalo perché non ho ancora scoperto se è un baco o meno, e il cui comportamento è random. Rimane un errore che in effetti non è tale.

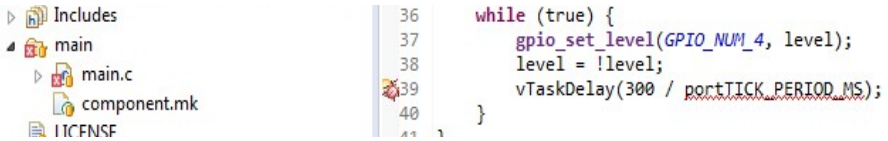

Figura 25: *Errore dopo le impostazioni.*

Gironzolando per internet ho trovato che ovviamente non ero l'unico con questo problema, ma che si poteva ovviare aggiungendo `#include "sdkconfig.h"` alla lista degli include. Questa sembrerebbe la ragione del perché in vari esempi compare quest'include ma che poi nessuno giustifica.

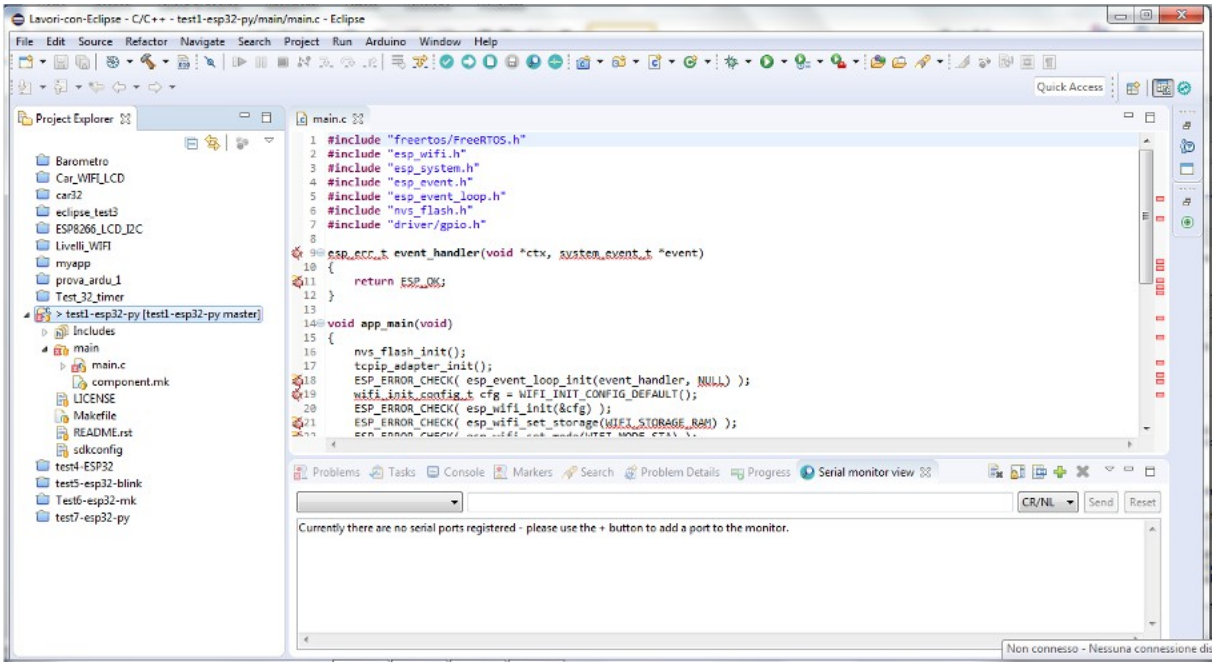

Figura 26: *Progetto da compilare.*

Risolto il problema, abbiamo raggiunto il duplice scopo, quello di avere un progetto funzionante e editabile in Eclipse, e quello di avere creato un nuovo file C/C++ per il nostro nuovo progetto. Nessuno vi vieta ora di riscrivere o sostituire il main.
Ci creeremo ora qualche comoda scorciatoia. Cliccate poi su *Project* quindi *Build Target* e infine su create.

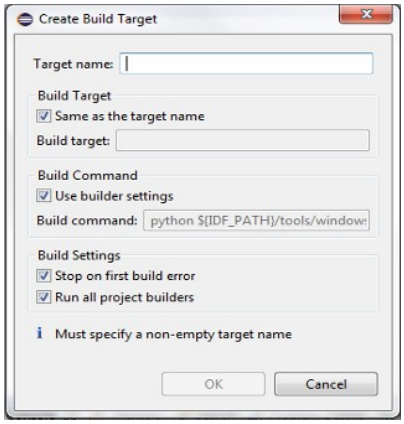

Figura 27: *Create Build Target.*

Nella finestra che si apre (Figura 27) in *Target Name*, scrivete uno dei *job* di *make* ad esempio all, flash, clean, app, app-flash e ripetete questa procedura per tutti i jobs a cui siete interessati. La Figura 28 vi mostra il risultato:

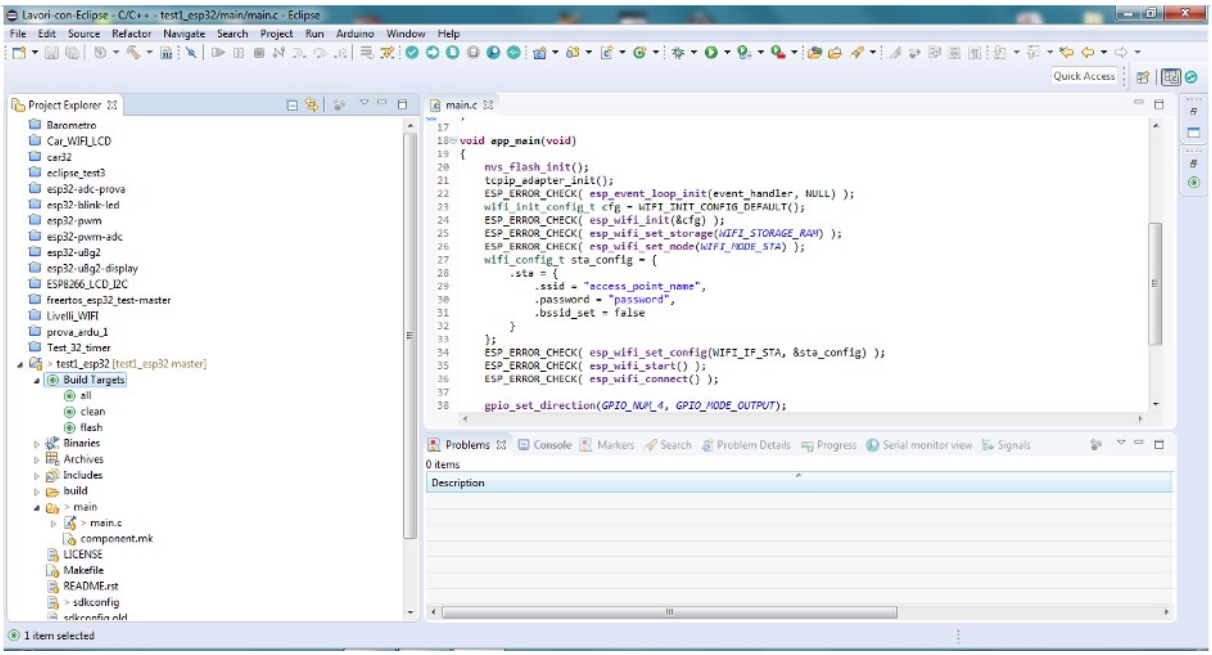

Figura 28: *Progetto completato.*

La gestione della vostra scheda ESP32 si può cosi effettuare con un doppio clic sui vari job.

Conclusioni

Spero di aver trattato l'argomento in maniera abbastanza dettagliata per farvi arrivare fino a questo punto senza troppi problemi. Non vi racconterò della programmazione delle schede ESP, la sto imparando anch'io, quello che mi sembrava necessario era avere uno strumento per incominciare a lavorare.

Indice Alfabetico

Bibliografia

[1] www.LaurTec.it: sito ufficiale dove poter scaricare gli aggiornamenti dell'articolo oltre ad altri Tutorial e Progetti.

History

Data	Versione	Autore	Revisione	Descrizione Cambiamento
16/02/18	1.0	Paolo Salvagnini	Mauro Laurenti	Versione Originale.