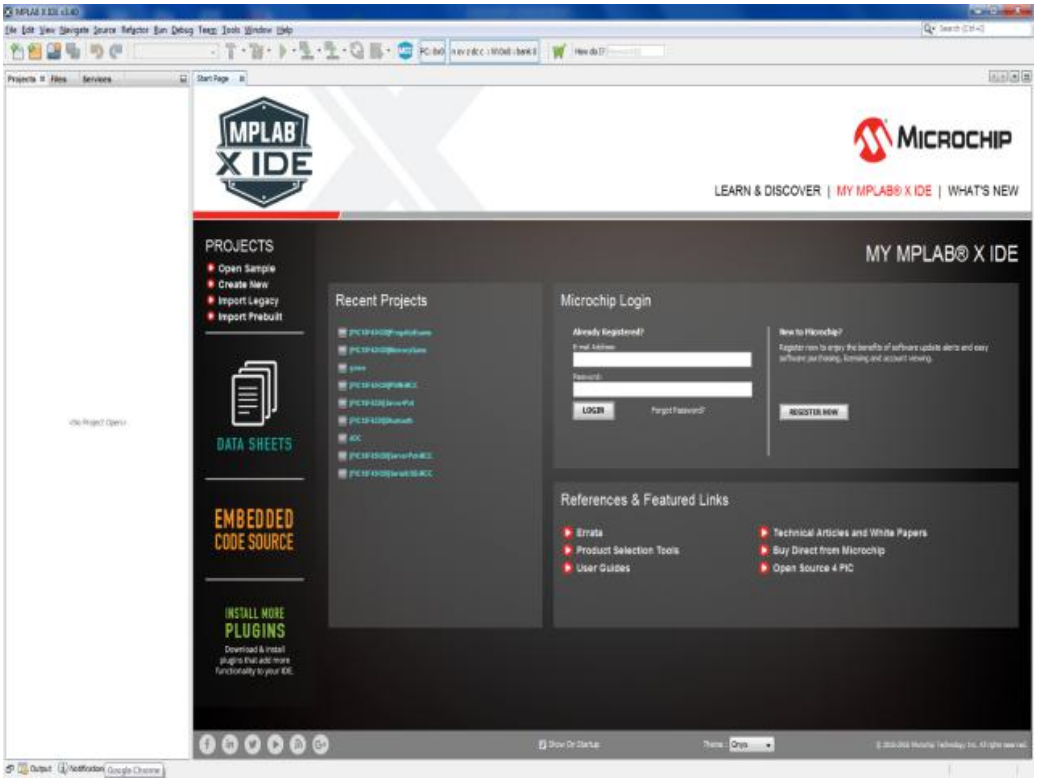
Guida all'utilizzo di Code Configurator

Obiettivo: far lampeggiare alternativamente tutti gli 8 led della nostra schedina collegati al PORTB, con tempi ben precisi, a seconda del pulsante da noi premuto.

Iniziamo il tutto aprendo MPLAB X IDE, che normalmente si trova sul nostro desktop. Appena aperto il programma ci troveremo di fronte ad una schermata del genere:

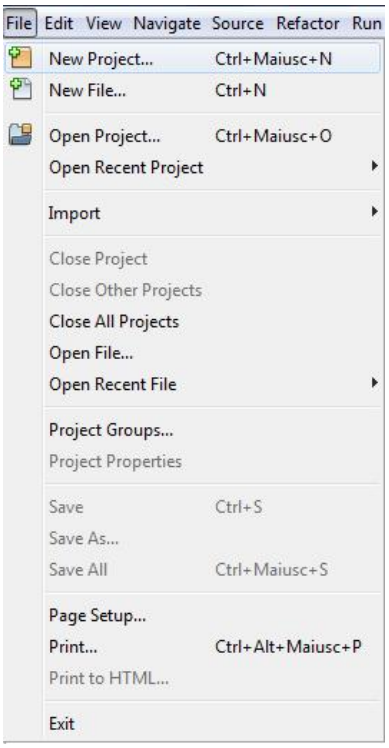


Qualora la schermata fosse leggermente diversa da questa significa che il vostro programma non è aggiornato a una delle ultime versioni, occorrerà quindi recarsi sul sito della MICROCHIP per scaricare l'ultima versione dell'IDE. ()

Creazione nuovo progetto:

Per iniziare un novo progetto possiamo seguire due diverse procedure:

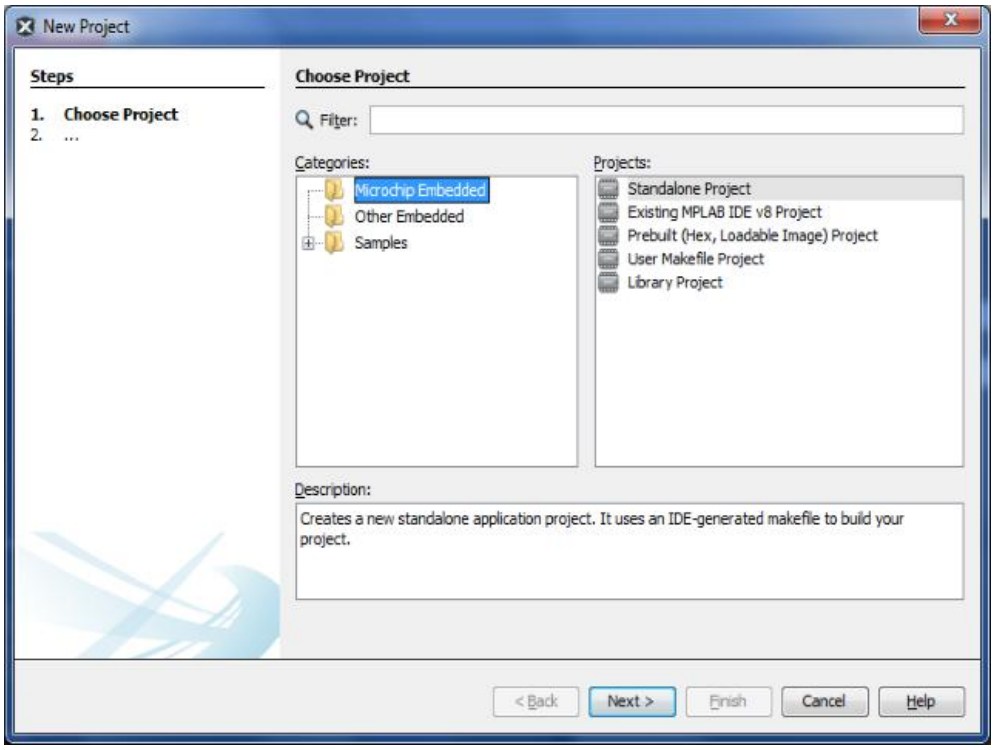
- Cliccare *File→New project…*



- Cliccare su questa icona: 

Si aprirà così una procedura guidata.

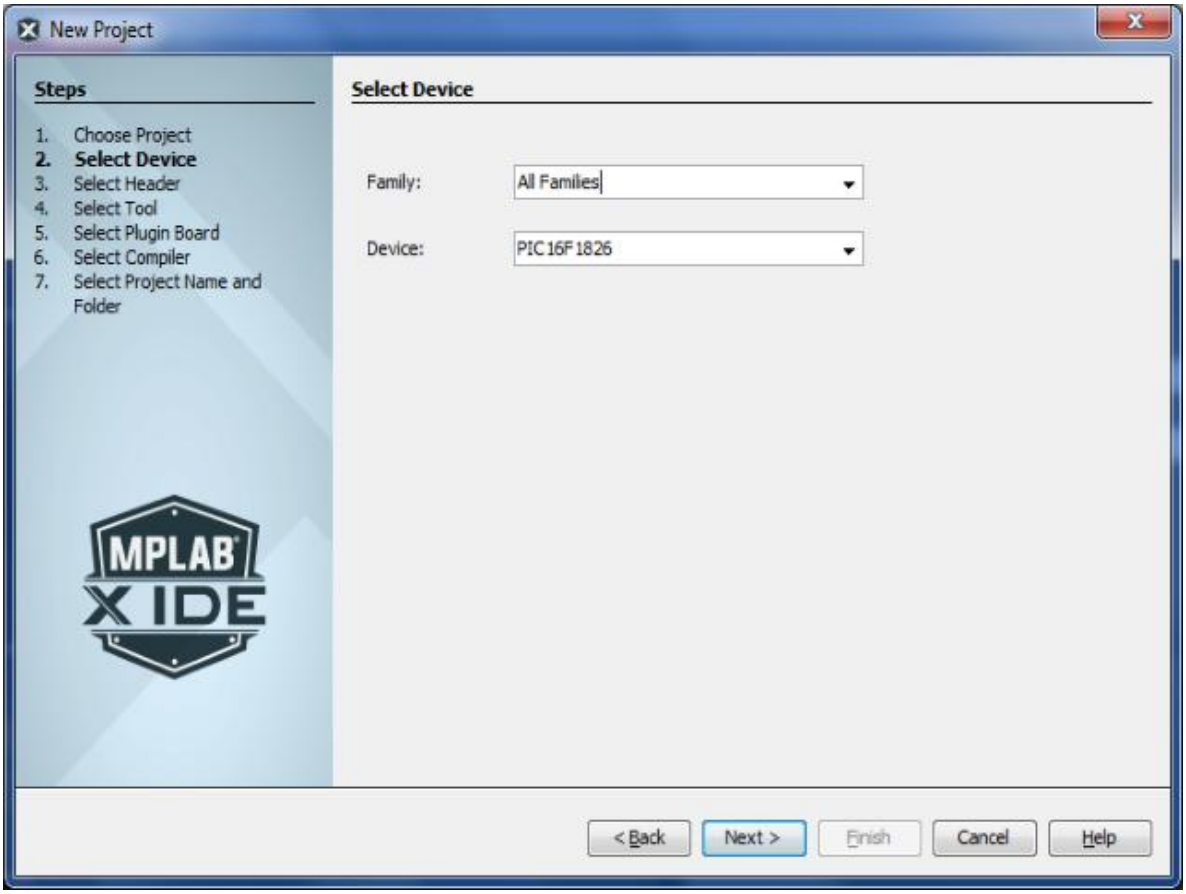
Scelta del tipo di progetto:



Il tipo di categoria (*Microchip Embedded*) e il tipo di progetto (*Standalone Projects*), selezionati di default dal programma, sono esattamente quelli che “servono” a noi quindi possiamo procedere cliccando su *Next* .

PIC utilizzato:

Una volta fatto questo dobbiamo scegliere il PIC che andremo ad utilizzare:



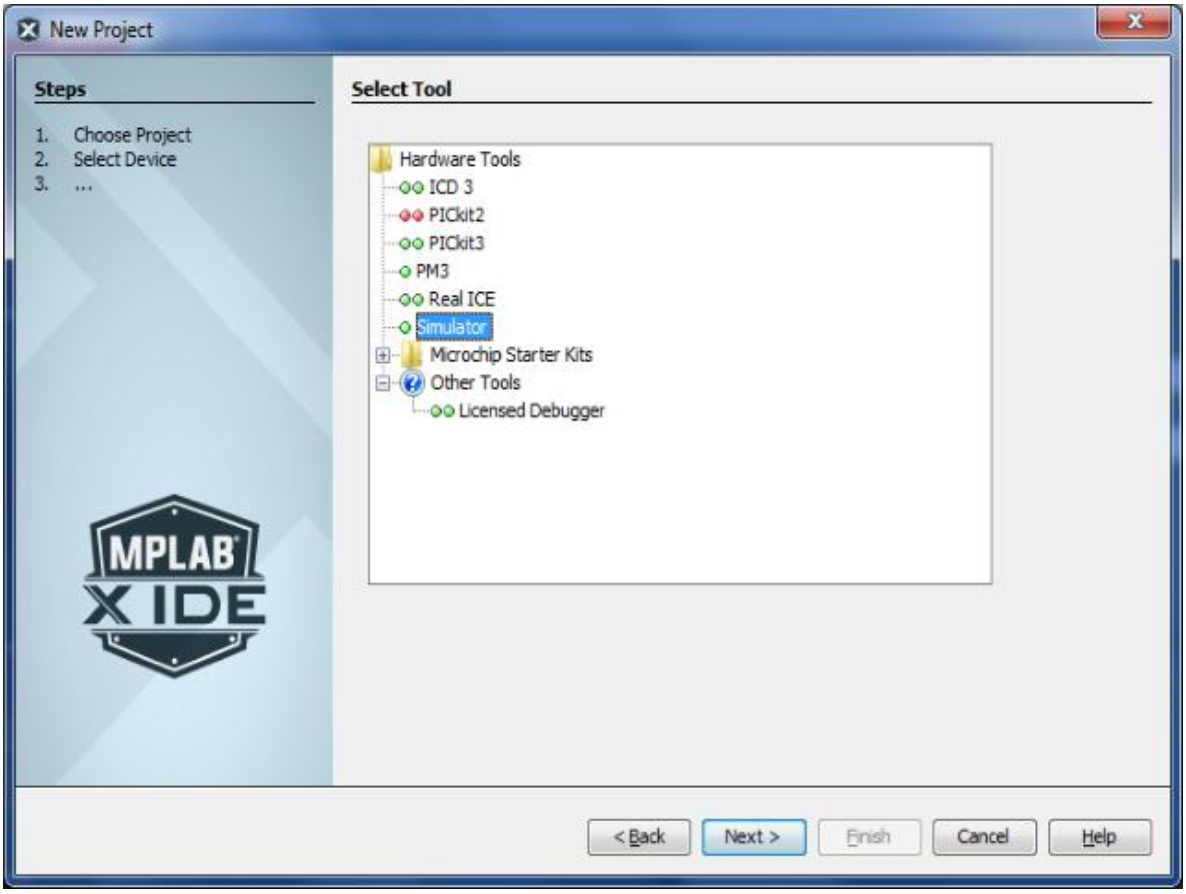
Questo lo possiamo ricercare attraverso il menù a tendina oppure semplicemente andando a scrivere in *Device* il nome del nostro microcontrollore, nel nostro caso il PIC16F1826. Proseguiamo ancora cliccando su *Next*.

Scelta opzionale dell'utilizzo di un debugger:

La seguente finestra ci dà la possibilità di utilizzare un debugger. A noi questo non serve quindi ci limitiamo a lasciare le cose così come stanno e ad andare avanti.

Scelta del programmatore utilizzato:

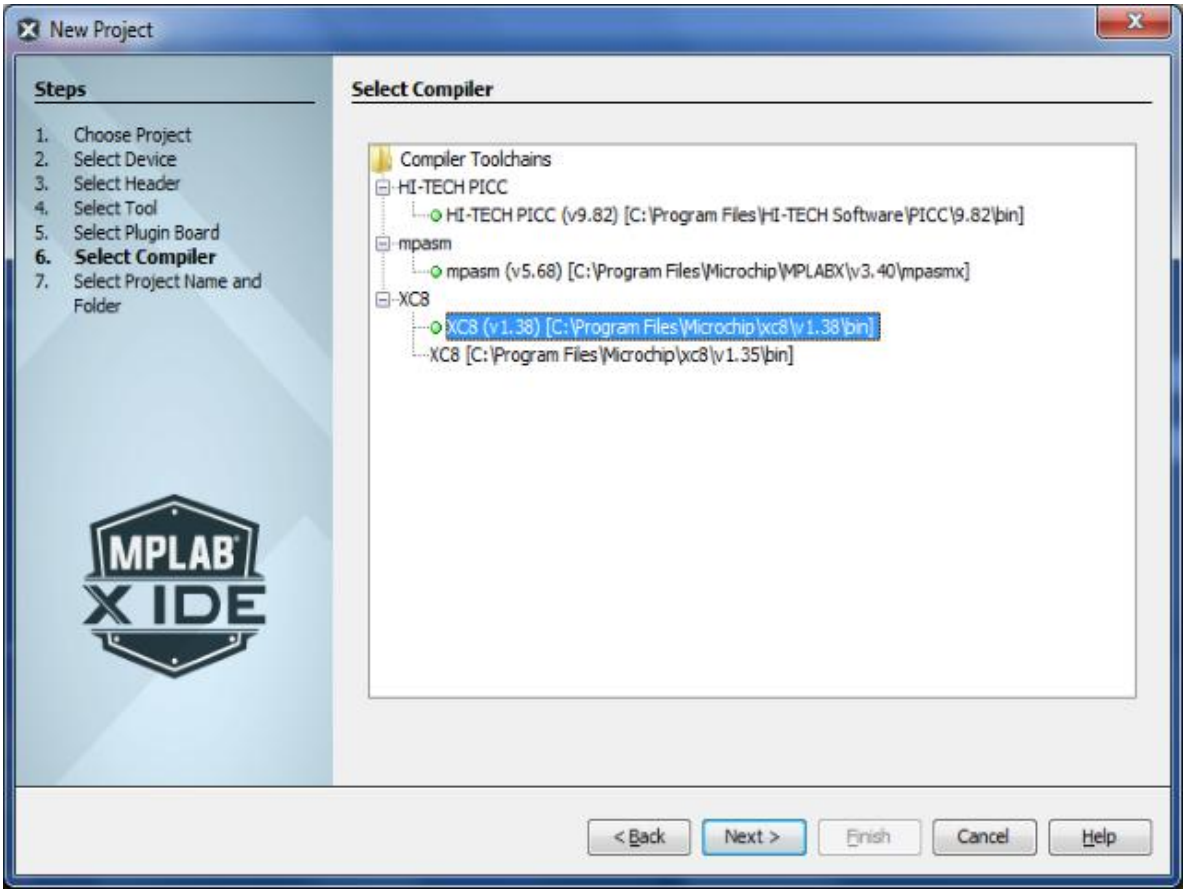
Non ci resta poi che scegliere il programmatore che andremo ad utilizzare, nel nostro caso il PICkit3 (che abbiamo a scuola), oppure eventualmente c'è la possibilità di utilizzare il simulatore e di verificare il funzionamento del nostro progetto, direttamente su MPLAB.



Una volta scelto proseguiamo con la creazione del nostro progetto.

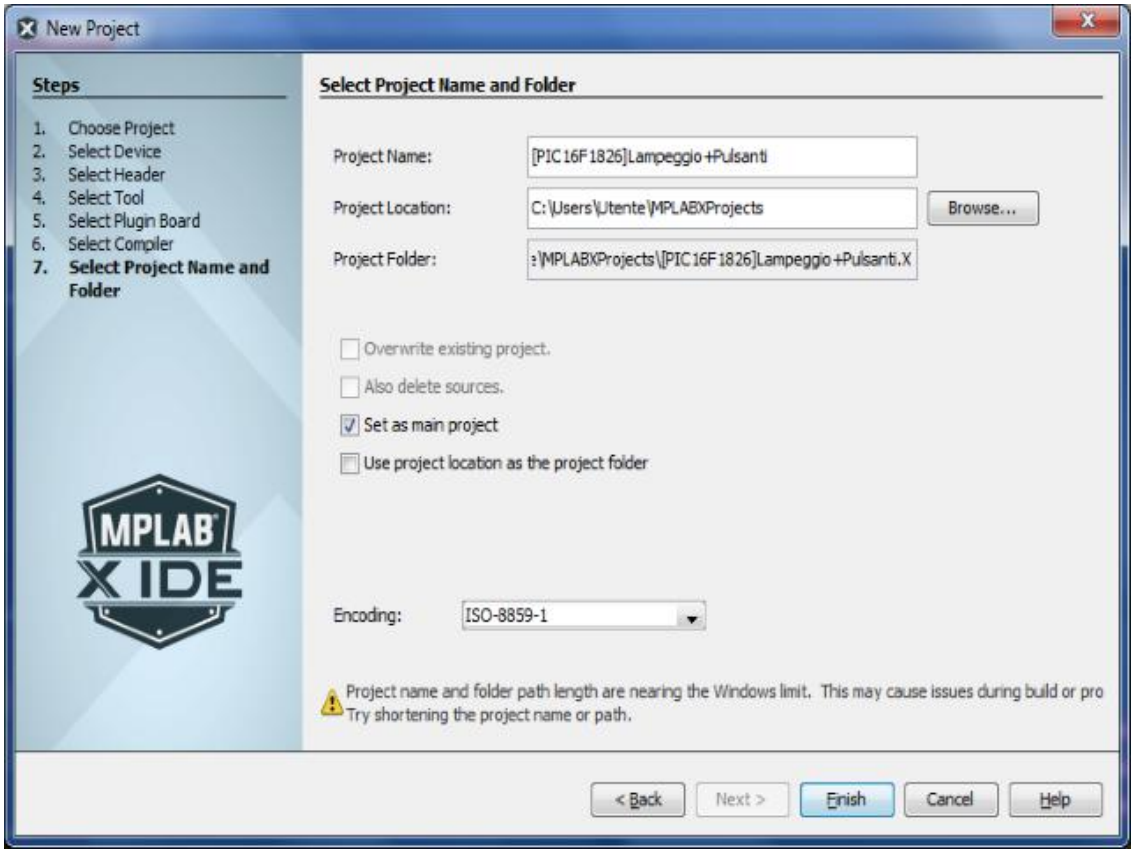
Scelta del compilatore:

Andremo poi a scegliere quindi il compilatore adatto alle nostre esigenze, ovvero XC8, un compilatore fornito direttamente dalla Microchip per i PIC a 8 bit.

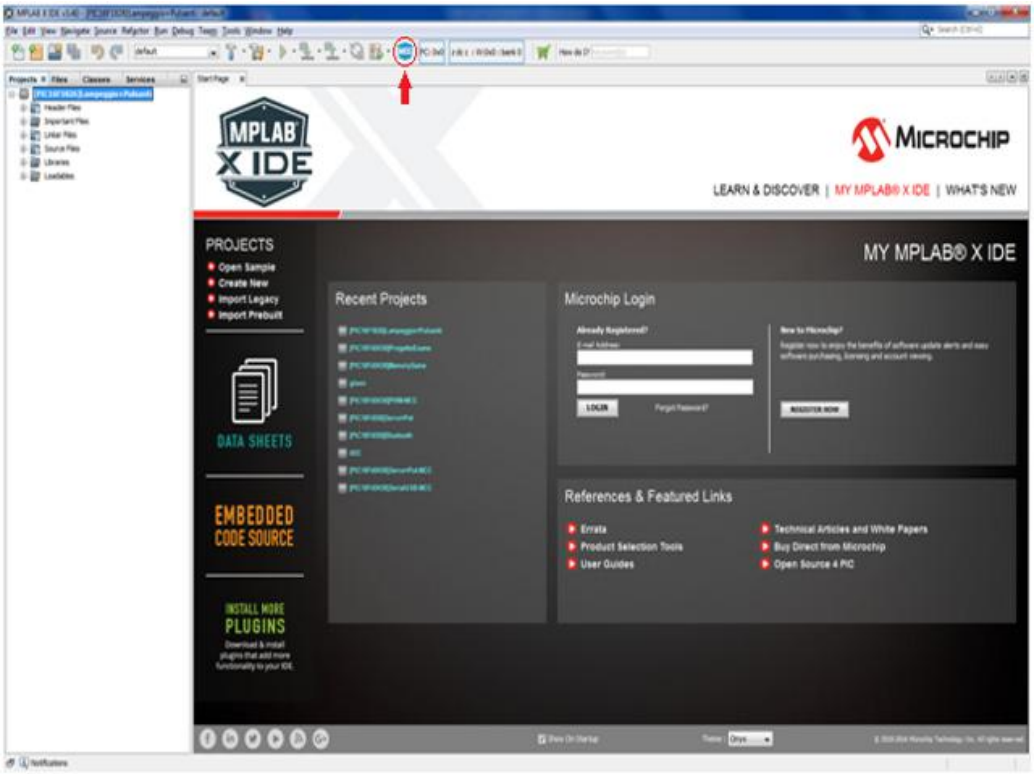


Percorso e nome della cartella entro cui salvare il nostro progetto:

Come ultimo passaggio non ci resta che scegliere la cartella di destinazione e il nome del nostro progetto:



Ora che la fase di creazione del nostro progetto è finita MPLAB ci riporterà automaticamente alla pagina iniziale:



Normalmente, se è la prima volta che utilizzate MPLAB X, Code Configurator non dovrebbe essere installato. Per sapere tutto ciò basta vedere se nella parte dei menù compare l'icona cerchiata nella figura precedente.

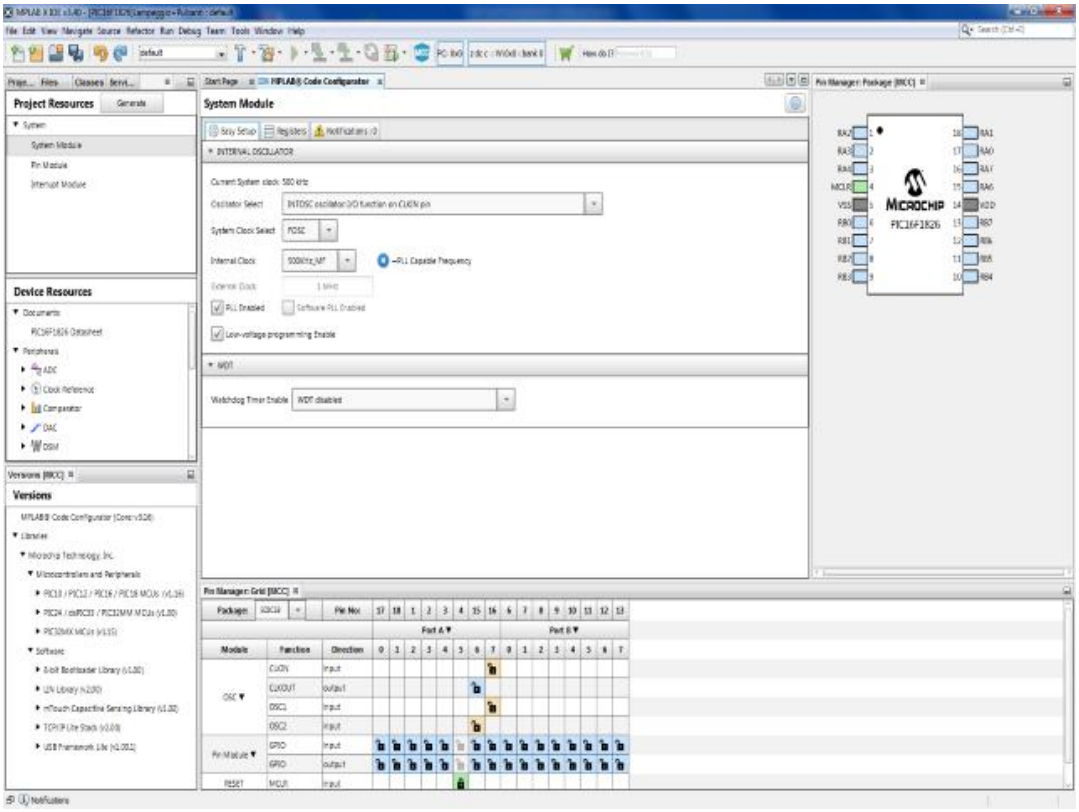
Installazione Code Configurator

Se non abbiamo installato Code Configurator dobbiamo eseguire alcuni semplici passaggi **(si necessita di una connessione a internet)**:

Clicchiamo su *Tools→Plugins→* nella tab *Available Plugins →* cerchiamo *MPLABCode Configurator* e spuntiamo la casella al suo fianco →*Install.*

Partirà così una procedura di installazione che seguirete alla lettera.

Una volta installato basterà cliccare sulla sua famosa icona per accedervi:

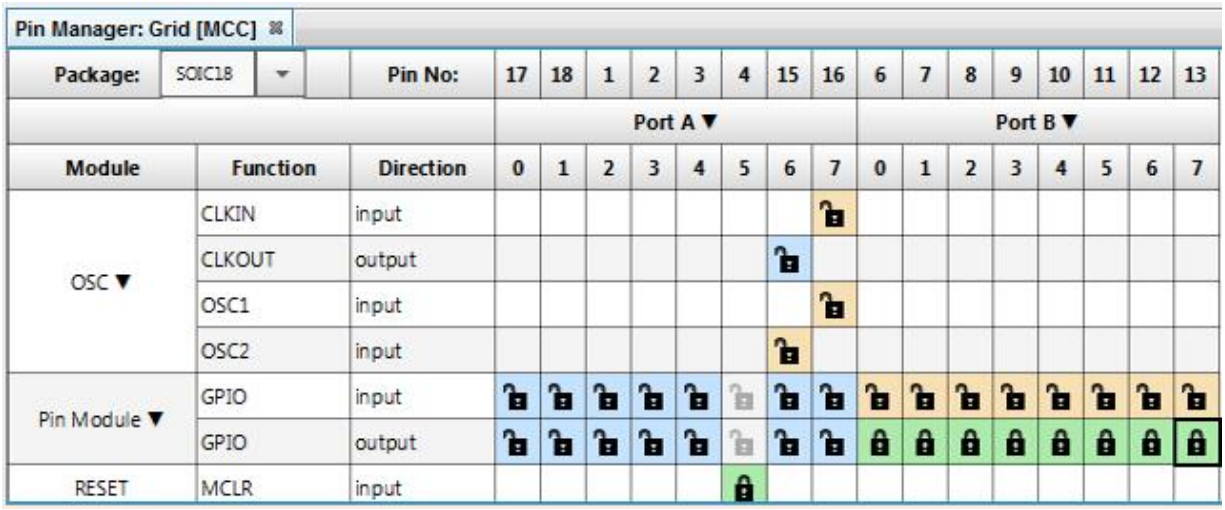


Da questo si potranno impostare diverse configurazioni:

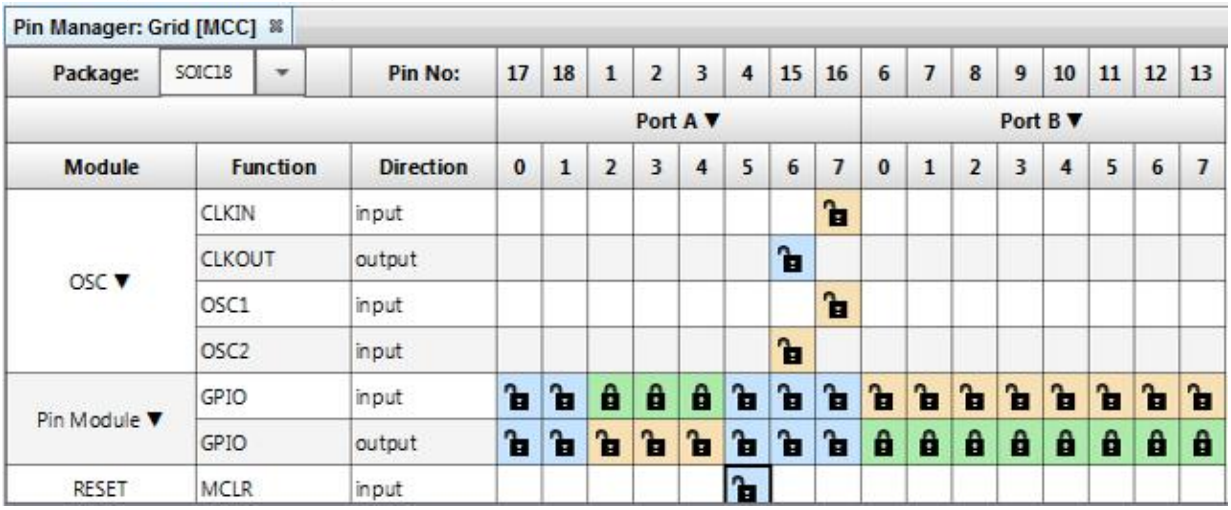
- Registri di configurazione del PIC
- I pin se utilizzati come input o output
- L'ADC, i CCP, le varie comunicazioni,…

A noi resterà soltanto la fatica di scrivere il programma!

Dati i prerequisiti del nostro progetto che, come ripeto, prevede di far lampeggiare alternativamente tutti gli 8 led della nostra schedina collegati al PORTB, con tempi ben precisi, a seconda del pulsante da noi premuto, possiamo procedere con l'impostare i pin del PORTB come OUTPUT recandoci nella tab *Pin Manager: Grid [MCC]*, che si trova nella parte bassa di MPLAB:



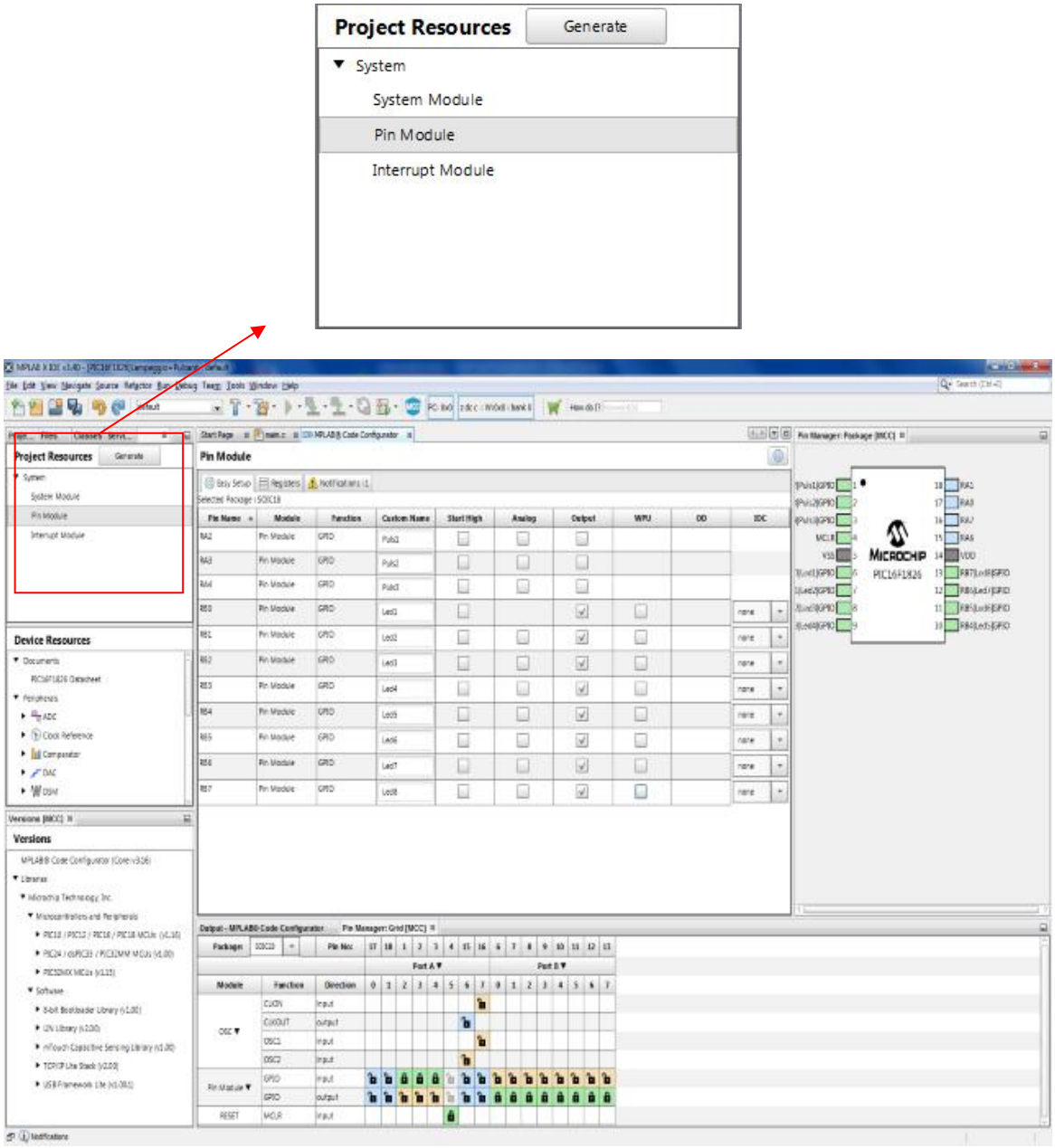
Come si può notare in figura il lucchetto chiuso in corrispondenza della linea ***output*** imposto il pin come un uscita, viceversa, se il lucchetto chiuso è in corrispondenza della linea ***input*** imposto il pin come ingresso. Esistono anche altre possibilità di configurazione dei pin, ma queste variano a seconda delle esigenze del nostro progetto, per questo si rimanda il lettore a approfondire eventualmente tutto ciò.
E impostare i pulsanti collegati ad RA2, RA3 ed RA4 come INPUT:



Da notare bene che in quest'ultima immagine il pin di reset è stato impostato per non eseguire questa funzione, infatti il lucchetto è stato tolto, questo perché la nostra schedina non prevede che il pin RA5 sia collegato tramite una resistenza di pull-up a 5V e con un pulsante poi a massa.

*La tab **Pin Module**:*

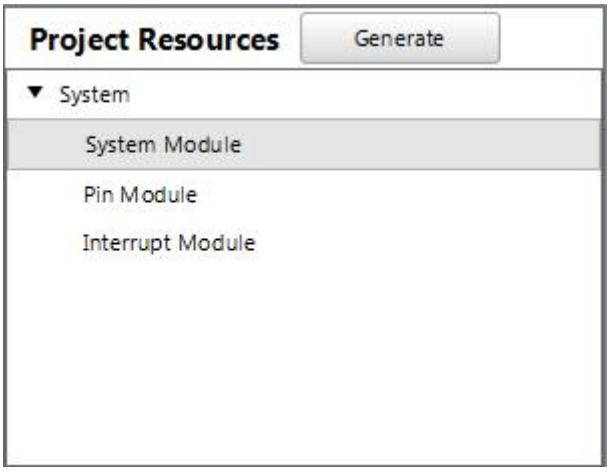
Il settaggio dei pin però non è ancora finito perché infatti dobbiamo recarci nella tab *Pin Module* e configurare il tutto a seconda delle nostre esigenze.

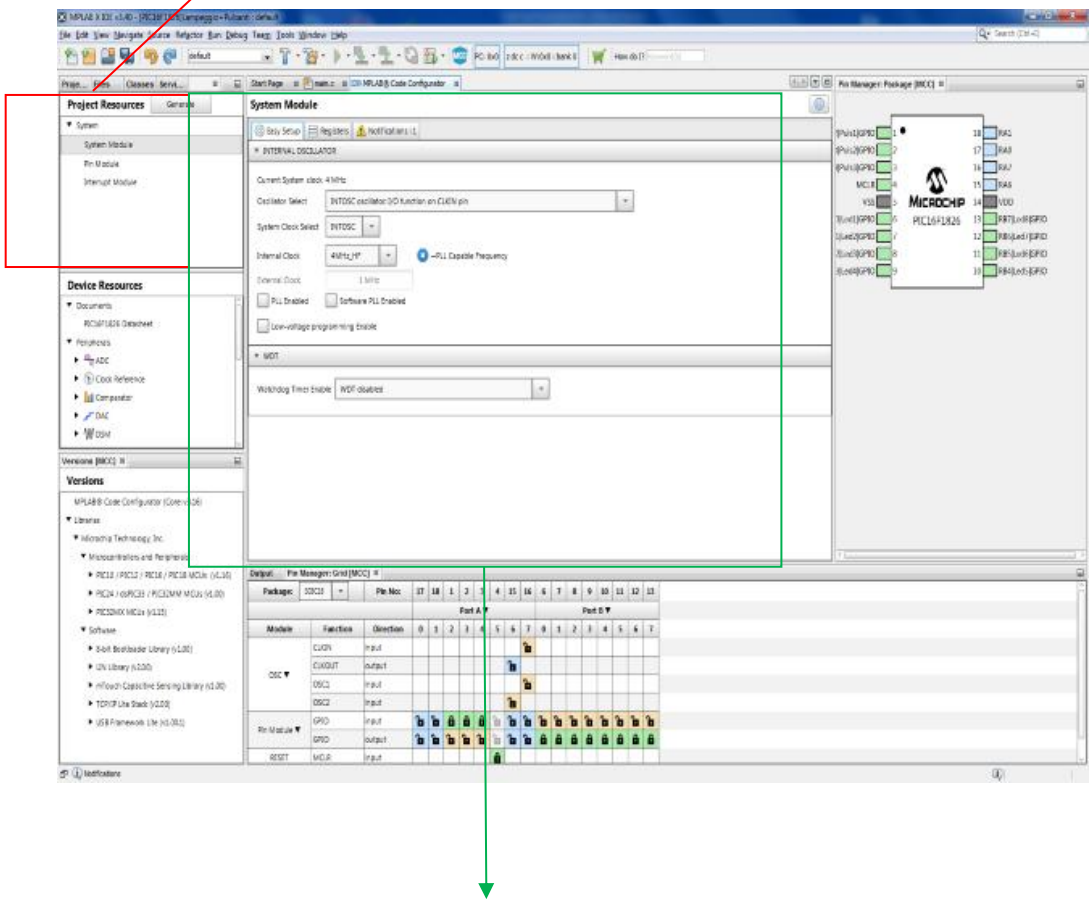


Spuntando le relative caselle possiamo scegliere se impostare un pin a livello alto 5V (*Start High*), come analogico o digitale (*Analog*), come uscita o ingresso (*Output*) o con la resistenza di pull-down (*WPU*).

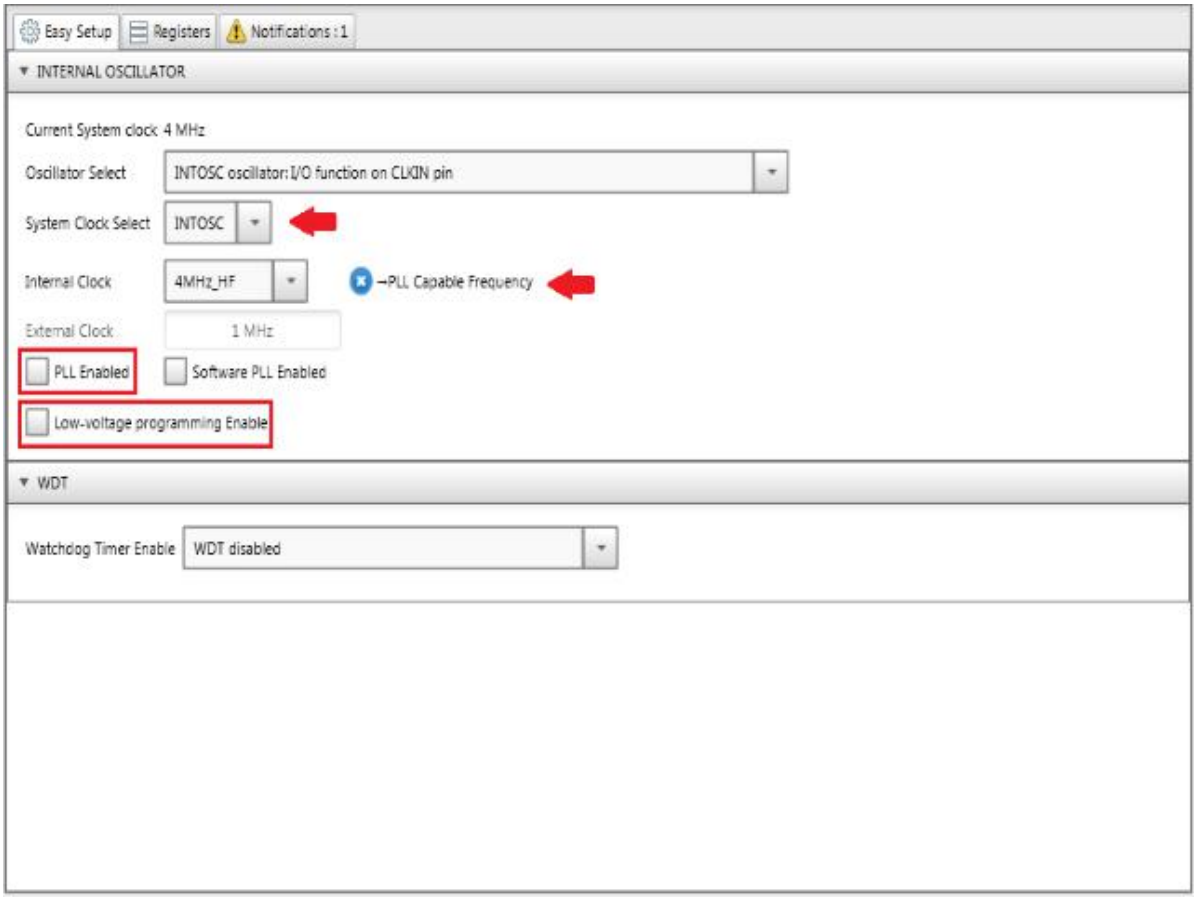
Settaggio oscillatore:

Una volta fatto tutto questo non ci resta che settare oscillatore, clicchiamo quindi sulla tab *System Module*:





Vedi pag. seguente

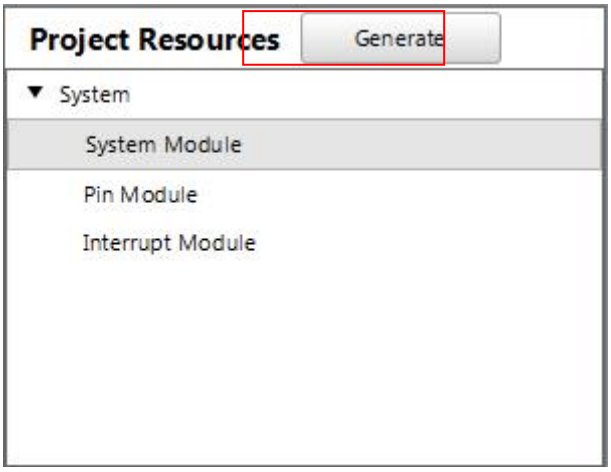


I parametri che sono stati cambiati sono quelli evidenziati da dei segni rossi:

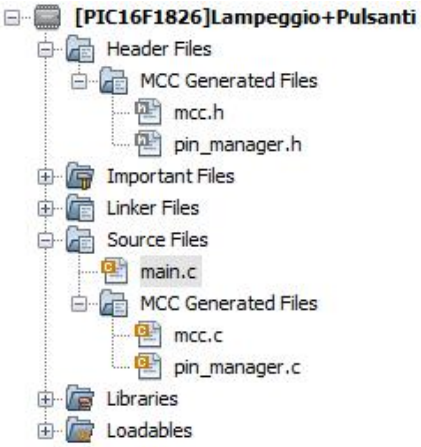
- *System Clock Select:* non volendo utilizzare il quarzo esterno, al posto di *FOSC* abbiamo messo *INTOSC*.
- *Internal Clock:* abbiamo cambiato la frequenza del clock a 4MHz, volendo si può scegliere tra più valori però questo è quello che utilizziamo noi solitamente.
- *PLL Enable:* abbiamo disabilitato questa funzione perché nel progetto non ci serve.
- *Low-voltage programming Enable:* abbiamo disabilitato la programmazione a basso voltaggio, perché prevede diverse impostazioni come ad esempio l'uso del sesto pin del programmatore, che all'interno della nostra schedina non è collegato a nulla.

Generazione file MCC:

Al termine di tutte queste operazioni e dopo aver correttamente controllato i valori da noi inseriti, clicchiamo su *Generate.*



All'interno del nostro progetto si verranno a creare diverse cartelle e file:



I file creati sonodi due tipi:

- *Header Files*,con estensione *.h*
- *Source Files*, con estensione *.c*

Andando ad analizzare dettagliatamente ogni singolo file possiamo vedere che:

- *mcc.h* contiene l'inclusione di tutte le librerie necessarie al corretto funzionamento del nostro programma (*<xc.h>, "pin_manager.h", <stdint.h>, <stdbool.h>*), oltre che alla dichiarazione della frequenza del quarzo utilizzato (*_XTAL_FREQ 4000000*). Nel file sono dichiarate anche le funzioni utili al "sistema" che saranno poi esplicitate all'interno del file *mcc.c*.
- *pin_manager.h* contiene alcune funzioni utili per scrivere il nostro programma, nel nostro progetto abbiamo utilizzato ad esempio *Puls2_GetValue()* che, controllando in questo file, sarebbe come scrivere *PORTAbits.RA3.*
- *mcc.c* contiene tutte le configurazioni del PIC, come l'utilizzo del quarzo interno, la disabilitazione del WATCHDOG,... (le configurazioni sono contraddistinte da *#pragma*). Sono inoltre contenute le funzioni dichiarate all'interno del file *mcc.h.*

- *pin_manager.c* contiene l'impostazione dei nostri pin attraverso i relativi registri.
- *main.c* Code Configurator crea anche un main, all'interno del quale vengono richiamati i file sopra citati e all'interno del quale andremo a scrivere il nostro programma.

N.B. Ogni qualvolta si apporta una modifica su Code Configurator, bisogna cliccare su *Generate.*

```
/**
  Generated Main Source File

  Company:
    Microchip Technology Inc.

  File Name:
    main.c

  Summary:
    This is the main file generated using MPLAB(c) Code Configurator

  Description:
    This header file provides implementations for driver APIs for all modules selected
in the GUI.
    Generation Information :
        Product Revision  :  MPLAB(c) Code Configurator - 3.16
        Device            :  PIC16F1826
        Driver Version    :  2.00
    The generated drivers are tested against the following:
        Compiler          :  XC8 1.35
        MPLAB             :  MPLAB X 3.20
 */

/*
    (c) 2016 Microchip Technology Inc. and its subsidiaries. You may use this
    software and any derivatives exclusively with Microchip products.

    THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER
    EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY IMPLIED
    WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A
    PARTICULAR PURPOSE, OR ITS INTERACTION WITH MICROCHIP PRODUCTS, COMBINATION
    WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

    IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE,
    INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND
    WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS
    BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE
    FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN
    ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY,
    THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

    MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF THESE
    TERMS.
 */

#include "mcc_generated_files/mcc.h"

void RitardoMS(signed int count){
    while(count > 0){
        __delay_ms(1);
        count--;
    }
}
```

```
/*
                         Main application
 */
void main(void) {
    // initialize the device
    SYSTEM_Initialize();

    // When using interrupts, you need to set the Global and Peripheral Interrupt
Enable bits
    // Use the following macros to:

    // Enable the Global Interrupts
    //INTERRUPT_GlobalInterruptEnable();

    // Enable the Peripheral Interrupts
    //INTERRUPT_PeripheralInterruptEnable();

    // Disable the Global Interrupts
    //INTERRUPT_GlobalInterruptDisable();

    // Disable the Peripheral Interrupts
    //INTERRUPT_PeripheralInterruptDisable();

    while (1) {
        // Add your application code
        if (Puls2_GetValue() == 1 && Puls3_GetValue() == 0) {
            __delay_ms(30);
            if (Puls2_GetValue() == 1 && Puls3_GetValue() == 0) {
                PORTB = 0b00001111;
                RitardoMS(2000);
                PORTB = 0b00000000;
                RitardoMS(1000);
            }
        }
        if (Puls2_GetValue() == 1 && Puls3_GetValue() == 1) {
            __delay_ms(30);
            if (Puls2_GetValue() == 1 && Puls3_GetValue() == 1) {
                PORTB = 0b00000000;
                RitardoMS(10000);
                PORTB = 0b11110000;
                RitardoMS(10000);
            }
        }
        else PORTB = 0x00;
    }
}
/**
 End of File
 */
```

All'interno del programma si possono notare diverse nuove funzioni:

- *SYSTEM_Initialize();* che inizializza le varie impostazioni/configurazioni.
- *Puls2_GetValue();* che restituisce il valore del pin (0→0V,1→5V).

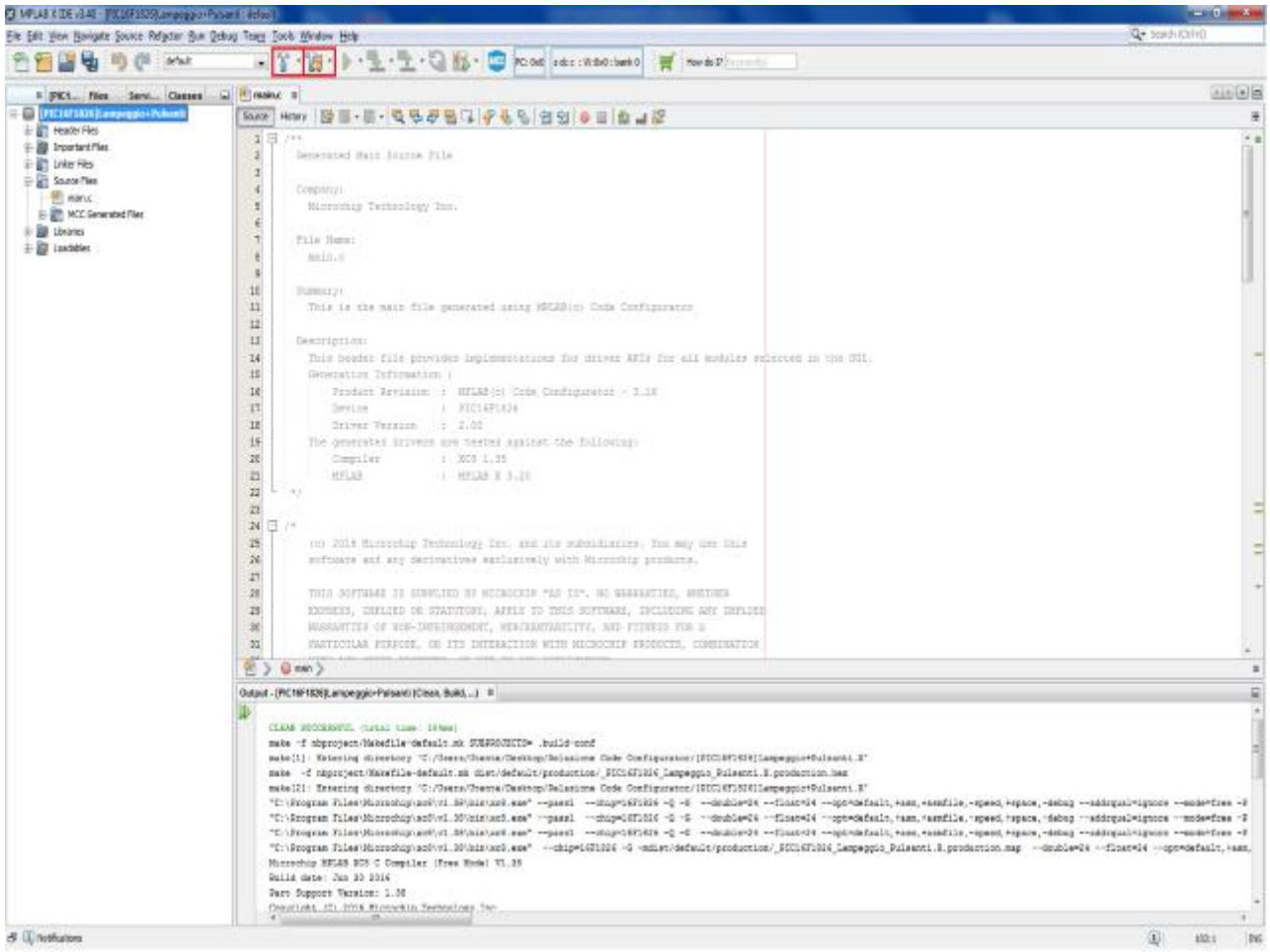
Si può notare anche l'utilizzo di una funzione ausiliaria, da me creata:

```
void RitardoMS(signed int count){
    while(count > 0){
        __delay_ms(1);
        count--;
    }
}
```

Creata vista l'impossibilità da parte della funzione `__delay_ms();` del compilatore di accettare valori troppi grandi.

"Buildare" un programma

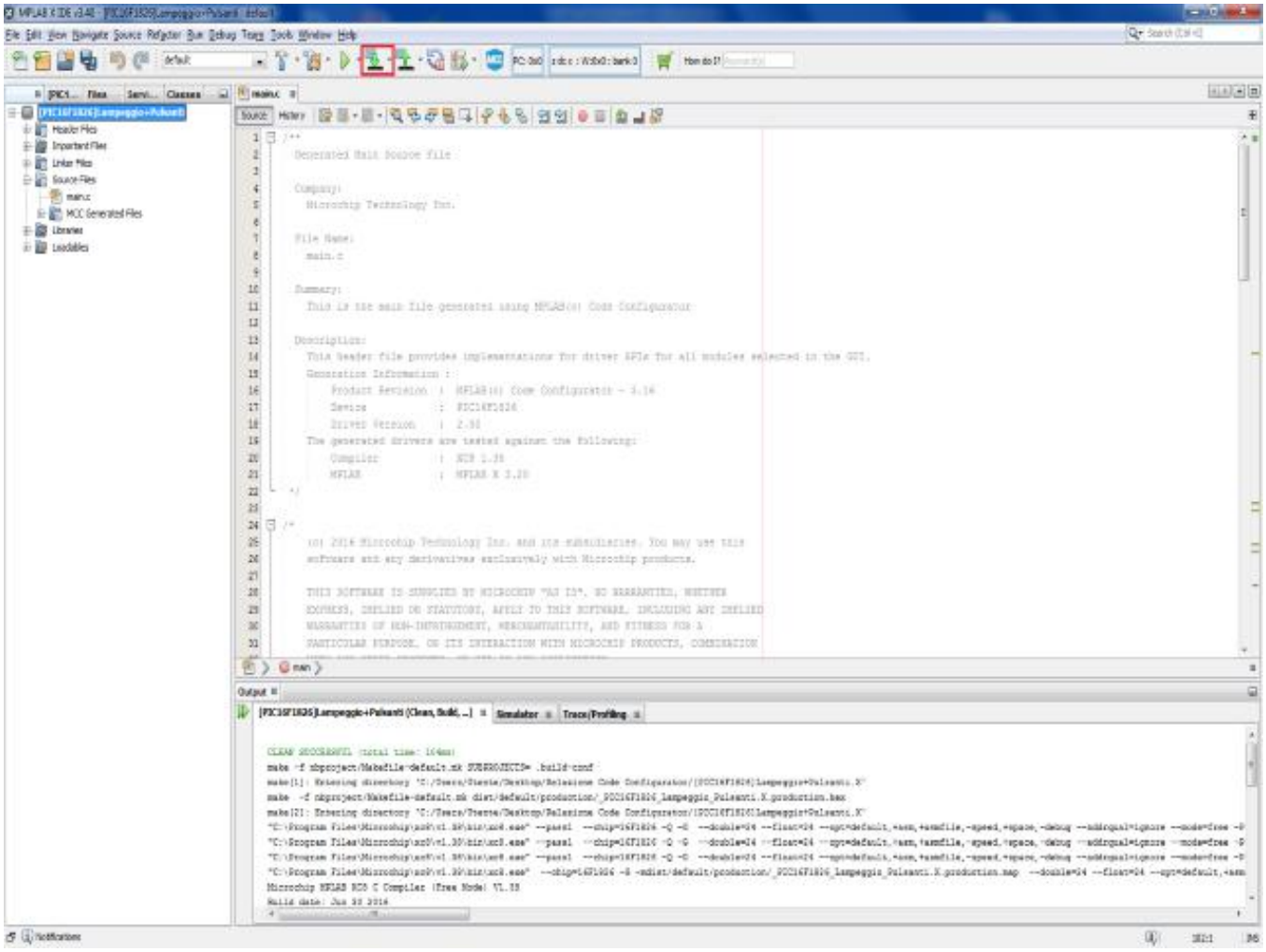
Appena abbiamo fatto tutto ciò non ci resta che verificare l'eventuale presenza di errori all'interno del nostro progetto. Per fare questo non dobbiamo che schiacciare l'icona con il martello nella parte alta di MPLAB.
N.B. L'icona con il martello e la scopa ripulisce prima i file creati dal compilatore e poi compila nuovamente il progetto, per questo molte volte è preferibile usare questa.



Programmare il PIC

Se non ci sono errori possiamo procedere con la programmazione del PIC:

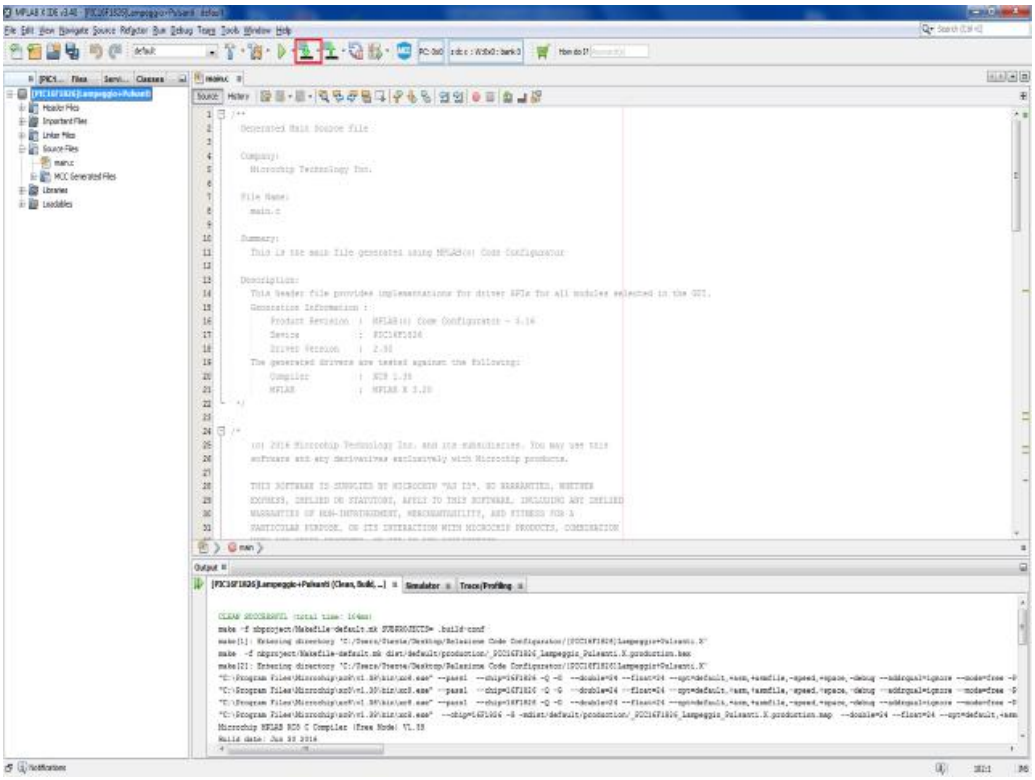
Se la schedina è alimentata → Collego il PICkit3 al PC e alla schedina → Clicco su *Program* (si trova nella parte alta di MPLAB) → Attendo che il caricamento si andato a buon fine



Se la schedina non è alimentata → Bisogna fornire alimentazione direttamente dal PICkit3:

Clicclo su *File* → *Projects Propertis(nome progetto)* → *PICkit3* → Apro il menù a tendina in alto e seleziono *Power* → Spunto la casella di fianco a *Power target circuit from PICkit3* → Clicco su *Apply* → *OK*

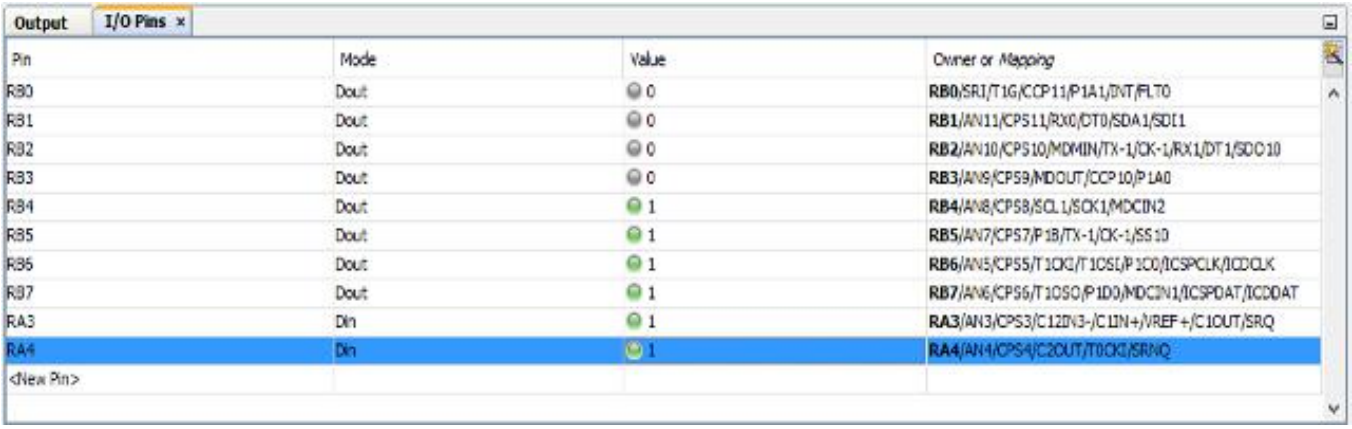
Ora la nostra schedina è alimentata direttamente dal PICkit3 → Clicco su *Program* (si trova nella parte alta di MPLAB) → Attendo che il caricamento si andato a buon fine



Nella pagina successiva si sono alcune immagini della simulazione e della prova su schedina.

Prova su schedina :

Simulazione:



Realizzato da Legni

Testato da Fiuzzi’s beta Tester