Programmare i PIC con Linux

Tutorial per utilizzare la programmazione dei PIC negli IT informatici mediante strumenti Open Source

di Salvatore Salzano

Contenuto

Indice generale

Premessa

Didatticamente parlando, l'uso dei PIC ha un valore notevole soprattutto nella classe 4a degli IT Informatici, per i suoi evidenti collegamenti con lo studio dei sistemi operativi (concetti come architettura degli elaboratori, interrupt, gestione delle periferiche etc etc diventano molto più facilmente comprensibili).

Inoltre ci sono alcuni importanti aspetti interdisciplinari con Elettronica, per tutto quanto attiene a periferiche e attuatori vari, e con Informatica, visto che i PIC li programmiamo in C e, fra l'altro, sono anche l'occasione per poter praticamente utilizzare operatori bit-a-bit e operatori di shift.

Detto questo, veniamo al punto: questo tutorial ha lo scopo di presentare un metodo di utilizzo dei PIC nella didattica, tutto realizzato con strumenti Open Source.

Questo tutorial è rivolto a chi ha una minima conoscenza di Linux, quanto basta per aprire un menù o muoversi in una finestra terminale. Principale destinatario è lo studente che deve configurarsi a casa gli strumenti per esercitarsi, tuttavia chiunque potrà, spero, trovarvi qualche cosa di utile per utilizzare i PIC con gli strumenti Open Source

Per segnalare errori o dare qualche suggerimento, scrivetemi pure a:

salvatore.salzano@gmail.com

Ingredienti HW e SW

Non servono grandi cose. Essenzialmente abbiamo bisogno di tre cose: un corso o comunque del materiale didattico, un po' di hardware e del software.

Materiale didattico

Su internet c'è veramente di tutto, io personalmente trovo molto valido il corso di Giovanni Bernardo, su settorezero.com, che utilizzo come punto di partenza per molte delle mie lezioni. In particolare sono veramente importanti le prime lezioni:

1. http://www.settorezero.com/wordpress/corso-programmazione-pic-in-c-lezione-1-cose-un-microcontrollore-caratteristiche-note-introduttive-come-scegliere-programmatore-e-linguaggio-di-programmazione/
2. http://www.settorezero.com/wordpress/corso-programmazione-picmicro-in-c-lezione-2-installazione-strumenti-di-sviluppo-descrizione-del-circuito-di-base/
3. http://www.settorezero.com/wordpress/corso-programmazione-picmicro-in-c-lezione-3-il-primo-programma-in-c-scrivere-un-semplice-programma-in-c-per-picmicro-impostare-mplab-e-flashare-il-picmicro-con-il-pickit2/
4. http://www.settorezero.com/wordpress/corso-programmazione-picmicro-in-c-lezione-4-cosa-sono-gli-interrupt-concetti-di-base-per-sistemi-operativi-multitasking-su-picmicro/
5. http://www.settorezero.com/wordpress/corso-programmazione-picmicro-in-c-lezione-5-timer0-e-prescaler-come-si-impostano-per-generare-interrupt-nei-tempi-che-vogliamo-applicazione-pratica/
6. http://www.settorezero.com/wordpress/corso-di-programmazione-picmicro-in-c-lezione-6-collegamento-di-pulsanti-pilotare-un-led-in-on-off/
7. Ci sono poi altre importanti risorse, per le quali rimando all'ultimo capitolo di questo tutorial

Hardware

E' vero che adesso ci sono già in giro i PIC della serie 24 e 30, ma penso che per poter iniziare senza troppi problemi sia meglio partire dai cari vecchi PIC della serie 18, che sono già molto potenti da poter fare tante cose, e sono ancora abbastanza semplici da poter essere compresi dagli studenti delle nostre scuole. Inoltre usare i PIC della serie 18 ci consente di utilizzare come programmatore il PICKIT2, che costa veramente poco, e una demoboard economica (nel caso non volessimo autocostruircela, cosa tutt'altro che difficile).

Quindi, ricapitolando, abbiamo bisogno del seguente hardware:

8. Il **programmatore** PICKIT2, reperibile su vari siti:
    1. RS-ONLINE
    2. Laurtech
    3. Robot Italy
9. Una scheda di sviluppo, la “**demoboard**”, sulla quale possiamo alloggiare un PIC e

troviamo già alcuni trasduttori elementari (e non) con cui fare le prime prove: pulsanti, led, potenziometri, buzzer, display LCD....). Due bellissime schede (le ho entrambe e vi garantisco che sono favolose) sono quelle elencate di seguito, scegliete quella che preferite:

   1. http://www.laurtec.it/progetti-elettronici/sistemi-automatici/108-scheda-di-sviluppo-usb-easyusb
   2. http://www.laurtec.it/progetti-elettronici/sistemi-automatici/76-scheda-di-sviluppo-per-pic-freedom-ii

10. Potrebbe essere utile anche una **breadboard** collegata alla demoboard, in modo da realizzare i primi progetti autonomi. Questa è quella che uso io: http://www.laurtec.it/progetti-elettronici/laboratorio/98-prototipi-schede-mille-fori
11. Un **alimentatore** in corrente continua da 7,5 o 9 volt e che fornisca almeno 600mA. Io ho utilizzato l'alimentatore del mio vecchio modem US Robotics 56K, che da quando ho l'ADSL giace tristemente in un angolo della cantina. Questo vi serve per alimentare la demoboard
12. Un alternativa ai punti 2 e 3 potrebbe essere l'autocostruzione.

Software

Il processo di sviluppo con i PIC segue essenzialmente le seguenti fasi:

1. avere una idea e progettare correttamente il tutto
2. Scrivere il codice sorgente sul vostro abituale personal computer. Io utilizzo il linguaggio C
3. Compilare il codice utilizzando un compilatore che generi codice eseguibile adatto a girare sui PIC
4. Trasferire il file eseguibile dal PC al PIC
5. Testare il funzionamento

Per fare tutte queste operazioni sono possibili diverse configurazioni software. Come sistema operativo, manco a dirlo, ho scelto Linux.

In Linux sono disponibili due principali alternative.

La prima è quella ufficiale della Microchip, MPLABX, ottima soluzione, la consiglio vivamente a chi vuole farne un uso professionale, ma nel campo della didattica preferisco utilizzare strumenti totalmente open-source e fra l'altro, da docente, preferisco che i miei studenti si abituino a smanettare con diversi programmi da far funzionare insieme piuttosto che fornire loro la pappa pronta.

La seconda possibilità è quindi l'uso di programmi Open Source.

In questo tutorial proporrò due metodi.

Il primo è proprio da “smanettoni”, e cioè:

   1. utilizzare un editor per scrivere il programma C
   2. compilare da linea di comando in una “comoda” finestra di terminale :-)
   3. programmare sempre da linea di comando da terminale :-))

Credo che sia molto formativo, almeno una volta, vedere l'intero processo.

Successivamente si può passare al secondo metodo, cioè integrare tutte le operazioni viste precedentemente utilizzando un apposito IDE.

Programmare i PIC da linea di comando

La soluzione che ho adottato io è la seguente:

1. un editor a sintassi evidenziata: tra i tanti, ho scelto Geany
2. compilatore: se si utilizzano i PIC, la scelta quasi obbligata per Linux è SDCC
3. programma per programmare il PIC: pk2cmd, ce lo fornisce la Microchip stessa

Installare i software necessari

Per installare Geany in una distribuzione Debian o derivata da essa (ad esempio, io attualmente uso Lubuntu 11.04) (ricordatevi che il comando da digitare è quello che viene dopo il simbolo "**$**" per gli utenti normali e dopo il simbolo "**#**" per l'utente "root", mentre la parte che viene prima è il "prompt")

```
salvatore@satellite:~$ sudo su      <invio>
root@satellite:/home/salvatore# apt-get install geany      <invio>
```

Come compilatore l'ideale è SDCC, sigla che significa "small device C compiler", cioè compilatore C per piccoli dispositivi. Come si capisce facilmente dal nome SDCC è in grado di scrivere codice per diversi microcontrollori, non solo per i PIC.

A questo indirizzo:

http://www.8052.it/download/sdccman.pdf

trovate un manuale tradotto in italiano per installare SDCC sul vostro computer.

Invece il manuale più aggiornato, direttamente dal sito di SDCC, è al seguente indirizzo:

http://sdcc.sourceforge.net/doc/sdccman.html/

Questo invece è il sito ufficiale del progetto dove potrete trovare tutte le informazioni più importanti:

http://sdcc.sourceforge.net/

Per installare SDCC occorrono anche le gputils, ma non preoccupatevi, se state usando apt-get ci pensa lui a proporvele. Quindi:

```
root@satellite:/home/salvatore# apt-get install sdcc      <invio>
```

A questo punto ci manca solo il programma per trasferire il file nel PIC. Se stiamo usando PICKIT2 ci serve un apposito programma fornito direttamente dalla Microchip: pk2cmd.

Per installarlo basta seguire questo ottimo tutorial di Massimo Spataro:

http://mcmax.blogspot.com/2009/12/pk2cmd-pickit2-microchip-con-linux.html

Comunque, per i più pigri, riassumo brevemente:

1. E' necessario, prima di tutto, che sul vostro PC sia stato installato l'ambiente di sviluppo (gcc & company). Se non lo avete ancora fatto, allora è meglio che lo facciate adesso:

   ```
   root@satellite:/home/salvatore# apt-get install build-essential <invio>
   ```

2. Occorre adesso scaricare il pk2cmd dal sito della Microchip:
   http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1960&fragment3_NextRow=31
   (selezionate la versione “Linux e MacOs” più recente)
3. Dopo aver scaricato il file, scompattatelo in una directory a vostra scelta (esempio, creo una cartella pic sotto la mia home e copio lì dentro il file scaricato):

   ```
   root@satellite:/home/salvatore# tar -xvf <nome del file scaricato> <invio>
   ```

4. entrate nella directory del pk2cmd (“**ls**” per vedere come si chiama la directory e “**cd**” per entrarci)
5. compilate il pk2cmd:

   ```
   root@satellite:/home/salvatore# make linux     <invio>
   ```

6. installate il :pk2cmd

   ```
   root@satellite:/home/salvatore# make install     <invio>
   ```

7. Seguite le istruzioni indicate nel link di Massimo Spataro per testare se tutto funziona. Se è tutto OK, allora adesso siamo pronti per scrivere il nostro primo programma.

Scrivere il primo programma

Il primo programma sarà quello per accendere un led. Il programma in questione è molto semplice, come si capisce leggendo le lezioni presenti su settorezero.com (vedi introduzione al tutorial), tuttavia è fortemente dipendente dal PIC che decidiamo di utilizzare.

Qui sono obbligatorie due parole.

La programmazione di un PIC richiederebbe di accedere direttamente a registri di memoria, e ciò andrebbe fatto utilizzando direttamente gli indirizzi di tali registri. Questo significa, fra le altre cose, che occorre avere una conoscenza profonda del modello di PIC che abbiamo deciso di

utilizzare.

A questo si può porre rimedio, in quanto sul sito della Microchip sono disponibili tutti i datasheet per i PIC disponibili.

Ad esempio, se decidiamo di utilizzare il PIC18F4550, il datasheet lo troviamo al seguente indirizzo:

http://ww1.microchip.com/downloads/en/devicedoc/39632c.pdf

Se non vogliamo diventare matti con indirizzi di memoria, Assembly e simili, conviene però utilizzare un altro sistema: includere all'inizio del programma un apposito file che contiene tutte le #define necessarie per poterci riferire alle porte e alle altre periferiche del PIC utilizzando un linguaggio "umano"!!!!

Ad esempio, se decidiamo di utilizzare il PIC18F4550, il nostro programma dovrebbe iniziare così:

```
#include <pic18f4550.h>
```

Prima di continuare, vorrei dire due parole su questi "include". Con una normale installazione di sdcc questi dovrebbero essere nella cartella **/usr/share/sdcc/include/pic16** (se non ci sono mi sa che avrete qualche problema al momento della compilazione :-)). Provate ad aprirne uno (ad esempio proprio **pic18f4550.h**), e confrontate le righe dopo la 477 con la tabella 5.1 a pagina 66 del datasheet del PIC: capito cosa fanno le define? Invece di scrivere in esadecimale "F81" per riferirci all'indirizzo di memoria dove si trova la porta B del PIC, noi possiamo scrivere semplicemente PORTB. Molto comodo, vero?

Dopo aver visto come iniziare ci scontriamo subito con il primo grosso problema: come facciamo a dare le impostazioni iniziali per i registri importanti (frequenza oscillatore, WDT etc etc.... se non sapete cosa sono allora è meglio tornare a leggere le lezioni 1, 2 e 3 del corso di settorezero.com)?

Per facilitarvi il compito, e senza costringervi a leggere attentamente le righe dalla 1476 in avanti del file **pic18f4550.h** oltre che il relativo datasheet, vi scrivo una configurazione iniziale che va bene per le demoboard consigliate e in genere per quarzi da 4Mhz in su.

Subito dopo l'include, inserite le seguenti righe che imposteranno correttamente i registri:

```
code char at __CONFIG1L CONFIG1L
= _PLLDIV_NO_DIVIDE__4MHZ_INPUT__1L
& _CPUDIV__OSC1_OSC2_SRC___1__96MHZ_PLL_SRC___2__1L
& _USBPLL_CLOCK_SRC_FROM_OSC1_OSC2_1L;

code char at __CONFIG1H CONFIG1H
= _OSC_HS__USB_HS_1H & _FCMEN_OFF_1H & _IESO_OFF_1H;
```

```
code char at __CONFIG2L CONFIG2L
= _PUT_ON_2L & _BODEN_ON_2L & _BODENV_2_0V_2L & _VREGEN_OFF_2L;

code char at __CONFIG2H CONFIG2H
= _WDT_DISABLED_CONTROLLED_2H & _WDTPS_1_32768_2H;

code char at __CONFIG3H CONFIG3H
= _CCP2MUX_RC1_3H &
_PBADEN_PORTB_4_0__CONFIGURED_AS_ANALOG_INPUTS_ON_RESET_3H
& _LPT1OSC_OFF_3H & _MCLRE_MCLR_ON_RE3_OFF_3H;

code char at __CONFIG4L CONFIG4L
= _STVR_ON_4L & _LVP_ON_4L & _ENICPORT_OFF_4L
& _ENHCPU_OFF_4L & _BACKBUG_OFF_4L;

code char at __CONFIG5L CONFIG5L
= _CP_0_OFF_5L & _CP_1_OFF_5L & _CP_2_OFF_5L & _CP_3_OFF_5L;

code char at __CONFIG5H CONFIG5H = _CPB_OFF_5H & _CPD_OFF_5H;

code char at __CONFIG6L CONFIG6L
= _WRT_0_OFF_6L & _WRT_1_OFF_6L & _WRT_2_OFF_6L & _WRT_3_OFF_6L;

code char at __CONFIG6H CONFIG6H
= _WRTC_OFF_6H & _WRTB_OFF_6H & _WRTD_OFF_6H;

code char at __CONFIG7L CONFIG7L
= _EBTR_0_OFF_7L & _EBTR_1_OFF_7L & _EBTR_2_OFF_7L & _EBTR_3_OFF_7L;

code char at __CONFIG7H CONFIG7H = _EBTRB_OFF_7H;
```

Attenzione, questa che vi ho proposto è la configurazione valida per il PIC18F4550 e non per gli altri.

Su questo punto occorre leggere sempre il datasheet relativo al PIC che vogliamo utilizzare.

Infine non ci resta che scrivere il primo programma, quello che accende tutti i led eventualmente collegati alla PORTB (ovviamente ipotizzando che abbiate veramente collegato otto led alla PORTB):

```
void main() {
TRISB = 0b00000000;
PORTB = 0b11111111;
}
```

Ora che abbiamo scritto il programma dobbiamo compilarlo. Per fare questo non é necessario

essere “root” ma si lavora come utente normale.

Io ho memorizzato il file utilizzato come esempio nella cartella ~/tutorialpic della mia /home e l'ho chiamato “primo.c”

Per compilarlo dobbiamo innanzitutto chiamare il compilatore sdcc passandogli le seguenti opzioni:

- quale architettura di PIC stiamo usando (-mPIC16)
- quale PIC stiamo usando (-p18f4550)
- abilitiamo i messaggi del compilatore per venire avvertitit degli errori (-V)
- inseriamo i simboli per poter usare il debugger (--debug)
- chiediamo solo di compilare e non di linkare (-c)
- indichiamo il nome del file sorgente (primo.c)

Ecco il comando da dare:

```
salvatore@satellite:~/tutorialpic$ sdcc -mpic16 -p18f4550 -V --debug
-c primo.c    <invio>
```

Volendo sarebbe stata utile anche l'opzione che indicava la directory corrente come quella che poteva contenere dei nostri eventuali <include> aggiuntivi. Bastava aggiungere la seguente opzione al compilatore:

```
-I/home/salvatore/tutorialpic
```

Se tutto è OK, ci troveremo con i seguenti file nella directory:

```
primo.adb
primo.asm
primo.c
primo.lst
primo.o
```

Adesso dobbiamo provvedere a chiamare il linker, e lo facciamo sempre con sdcc:

```
salvatore@satellite:~/tutorialpic$ sdcc -mpic16 -p18f4550 -V --debug
-Wl-c -Wl-m -I/home/salvatore/tutorialpic/ -oprimo.hex primo.o
<invio>
```

Se tutto è OK, adesso nella directory c'è anche il file **primo.hex** , che è il nostro file binario da trasferire nel PIC, come vedremo nel prossimo paragrafo

“Flashiamo” il primo PIC

Il termine “flashare”, non esistente nella lingua italiana, deriva da “flash”, ed è riferito al fatto che i PIC hanno memorie riscrivibili elettricamente, come tutte le EEPROM. In parole povere significa “*trasferire il file .hex dal PC al PIC*”.

Per far questo dobbiamo utilizzare il programma **pk2cmd**, che abbiamo installato in precedenza.

Dovremmo essere in grado di far tutto ciò dalla directory dove è memorizzato il programma.

La prima prova da fare, per essere certi di non aver problemi, è assicurarci che con il nostro utente (non con “root”) siamo in grado di accedere alla porta USB.

Con questo comando:

```
salvatore@satellite:/home/salvatore/tutorialpic$ pk2cmd –PPIC18F4550
–I –B/usr/share/pk2    <invio>
```

dovremmo ricevere un responso simile a questo:

```
Device ID = 1200
Revision  = 0007
Device Name = PIC18F4550
```

Se non è così allora vuol dire che forse c'è un problema con i permessi di lettura e scrittura sulla porta USB, in tal caso rivedeteli (nella peggiore delle ipotesi, aggiungete il vostro utente al gruppo root in /etc/group, io non ho avuto questa necessità, ma non si sa mai che installazione avete fatto). Altrimenti siamo pronti per programmare.

Colleghiamo il PICKIT2 alla demoboard (vedere il manuale della demoboard in vostro possesso per sapere come fare). quindi diamo il comando che trasferisce il file .hex nel PIC18F4550:

```
salvatore@satellite:~/tutorialpic$ pk2cmd –PPIC18F4550 –F primo.hex
–M –B/usr/share/pk2    <invio>
```

Per provare il funzionamento, stacchiamo il PICKIT2 dalla demoboard e alimentiamola con il suo alimentatore. Se tutto è stato fatto correttamente vedrete accendersi i led sulla PORTB.

Piklab: un semplice ma potente IDE per i PIC

A questo punto, avendo visto quale è il percorso che occorre compiere, vediamo se possiamo semplificarci l'esistenza utilizzando un software che consenta di compiere tutte le azioni indicate.

Ci sarebbero diversi modi. Innanzitutto si potrebbero impostare editor come Geany in modo da dare i comandi di compilazione e programmazione del PIC in modo automatico. Provate a selezionare la voce di menù “genera” e quindi “set build commands”. Questa soluzione ha il vantaggio di semplificare le operazioni di compilazione, link e programmazione, ma non risolve il problema principale con i PIC: avere un aiuto per impostare i registri all'inizio del programma.

Una valida soluzione è usare Piklab, che fa proprio questo: ci consente di impostare un progetto e selezionare il PIC che vogliamo usare, scegliere il modello di programmatore (per chi non ha PICKIT2 ma ha, ad esempio, ICD2), scegliere il compilatore (non esiste solo sdcc) e, cosa più importante, imposta correttamente i registri in base alle nostre richieste, e infine ci crea un file template già pronto per l'uso.

Come prima cosa occorre installare Piklab.

Questo software è stato scritto con le librerie Qt3, e quindi la sua installazione non è fra quelle normalmente previste in una Debian o derivata, soprattutto fra quelle più recenti. Tuttavia non è difficile, occorre solo fare una piccola variazione alla configurazione di apt-get e magicamente ci troveremo Piklab fra i programmi installabili automaticamente.

Modificare apt-get

Come noto la configurazione del sistema di gestione dei pacchetti, apt-get, dipende dall'elenco dei siti “repository”, quelli cioè dove vengono cercati i pacchetti da installare.

Per poter trovare facilmente Piklab occorre aggiungere il suo repository. La distribuzione che utilizzo io attualmente è la Lubuntu 11.04 (natty) tuttavia non c'è il pacchetto debian compilato per questa distribuzione, ma per quella precedente, la 10.10, “maverick”.

La riga da aggiungere in fondo al file /etc/apt/sources.list è la seguente:

```
deb http://ppa.launchpad.net/michael-gruz/elektronik/ubuntu maverick main
```

che, come si può vedere, è relativa alla distribuzione “maverick”.

Aggiungere la riga indicata, poi fare un bel

```
apt-get update
```

e quindi

```
apt-get install piklab
```

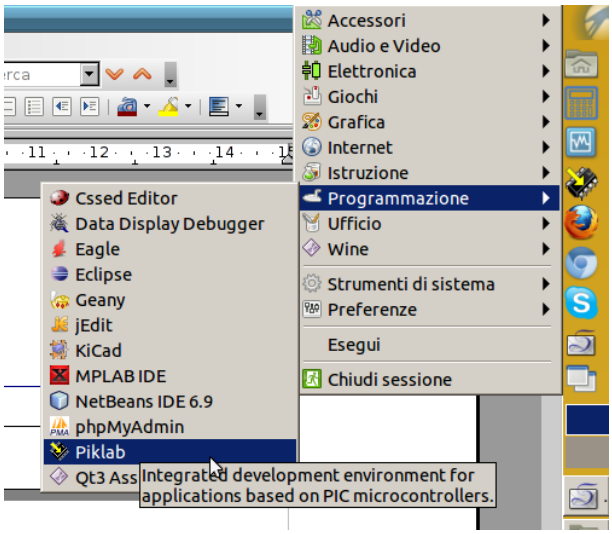
e tutto andrà per il meglio. Il sistema di gestione dei pacchetti vi chiederà di confermare l'installazione in quanto il repository non è tra quelli ufficialmente ritenuti fidati, voi accettate e tutto filerà liscio

Non posso garantire che con le prossime versioni il “trucco” funzionerà ancora, tuttavia c'è sempre la possibilità di scaricare il sorgente e compilarlo, direttamente dal sito ufficiale del progetto Piklab.

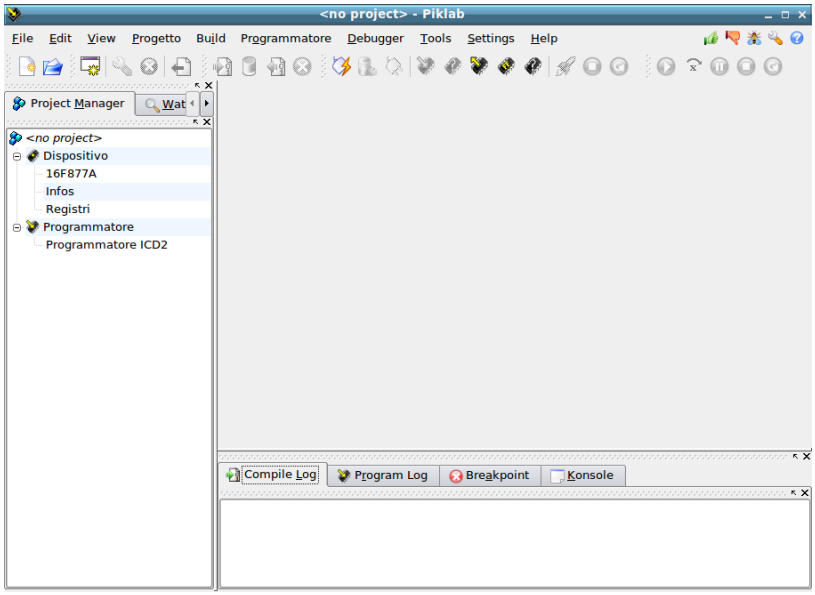
Primi passi con Piklab: creare un nuovo progetto

L'uso di Piklab non è molto complesso: è il classico IDE che lavora “per progetti”:

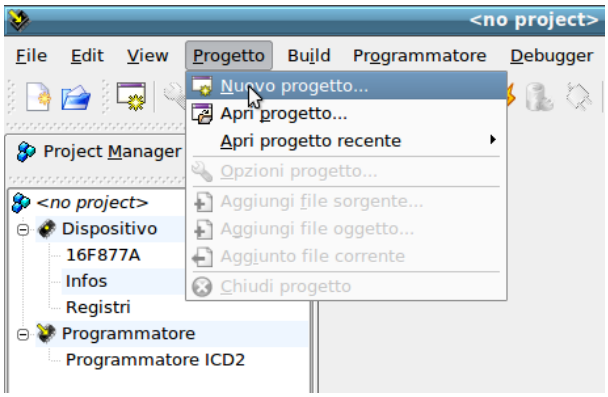
Innanzitutto lo selezionate dal menù e lo avviate:



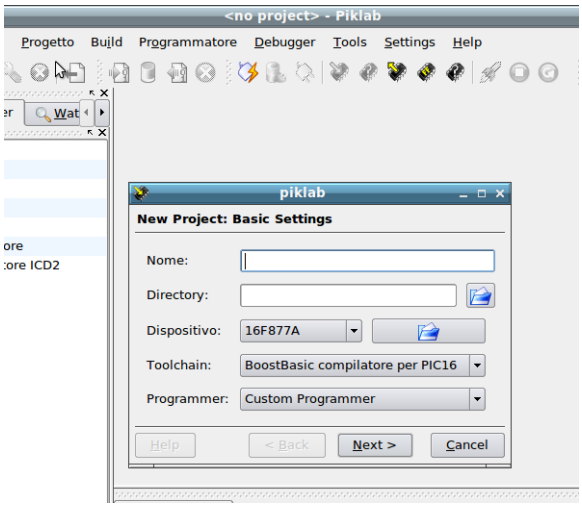
Vi troverete davanti la finestra di Piklab che dovrebbe apparire più o meno così:



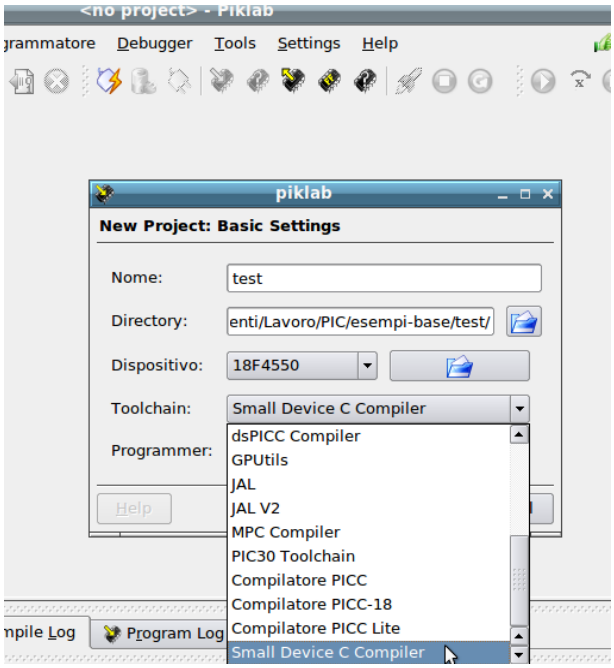
A questo punto apriamo il menù “Progetto”:

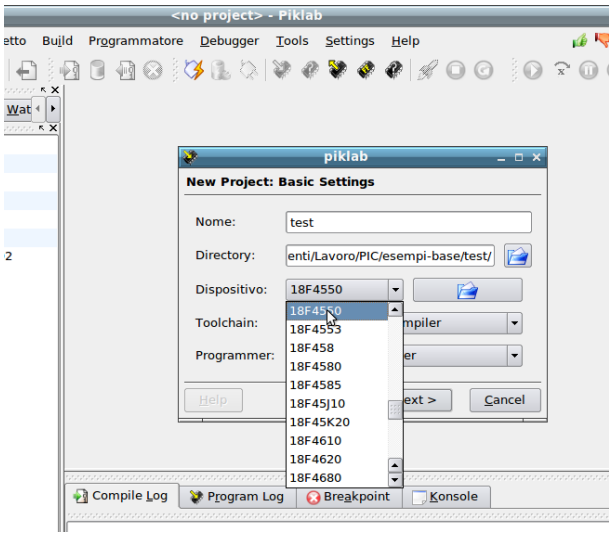


e, dopo aver selezionato la voce “Nuovo progetto”, ci troveremo con il seguente “wizard”

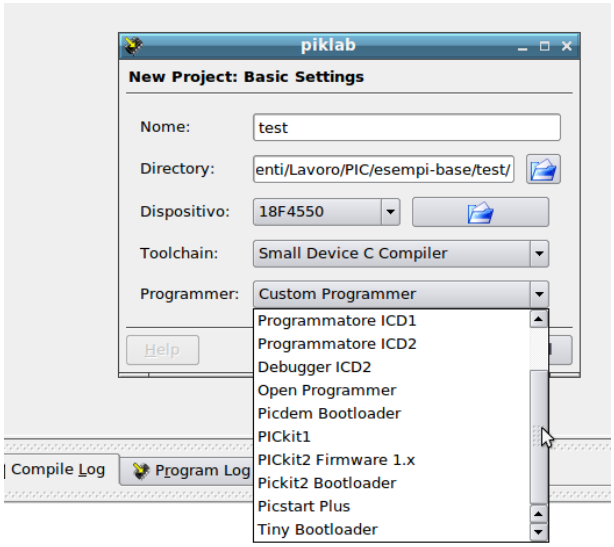


Diamo un nome al progetto e selezioaniamo una cartella che lo conterrà, quindi dagli appositi menù di scelta impostiamo il modello di PIC e il compilatore SDCC:

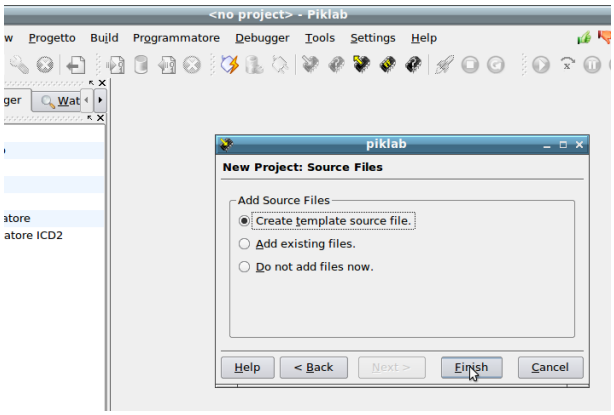




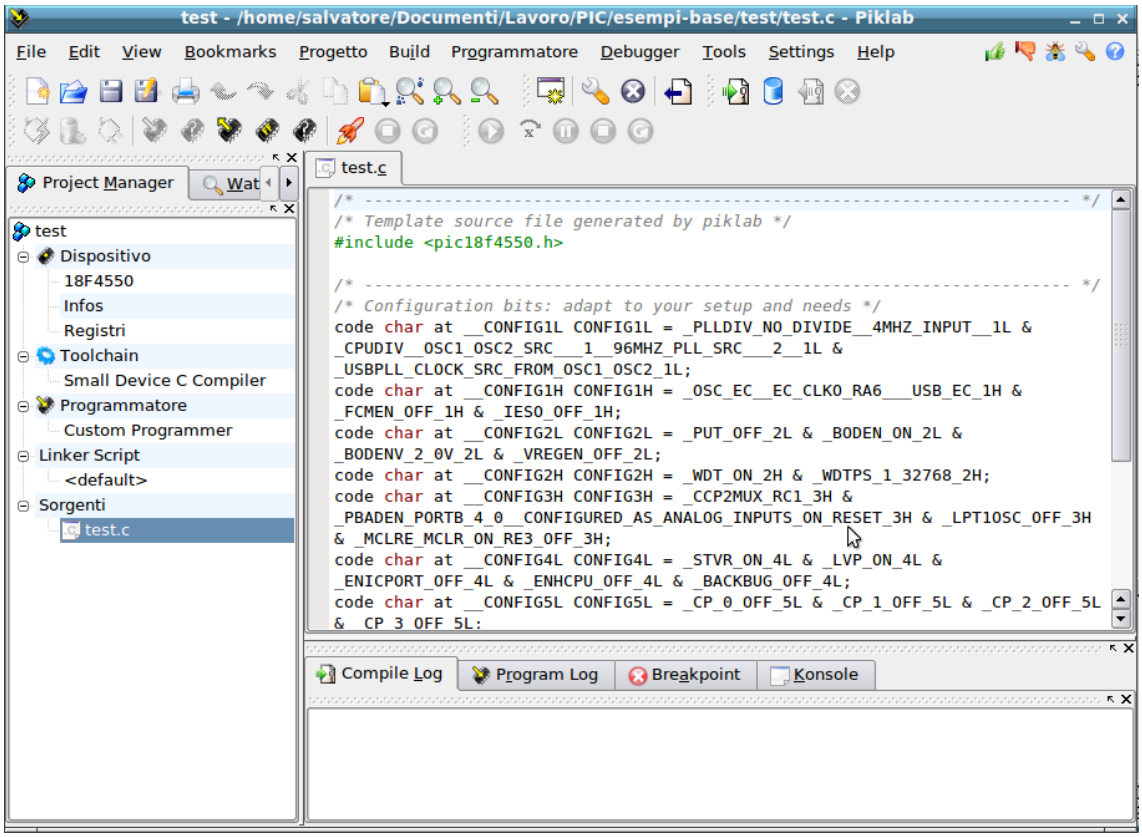
A questo punto viene una parte molto importante. Dobbiamo impostare il programmatore. Anche se dal menù a tendina vi appare la possibilità di selezionare il PICkit2, non fatelo: la voce indicata è relativa ai vecchi modelli con il firmware della serie 1.x. Noi dovremo creare una configurazione personalizzata, come mostrato nel prossimo paragrafo, quindi adesso selezioniamo “Custom Programmer” e proseguiamo



Dopo aver selezionato il programmatore, ci appare un'ultima finestra di scelta, che ci chiede se vogliamo generare i file template, in modo da non partire con un file vuoto. Ovviamente, SI!!!



Ed ecco creato il file con l'impostazione di base per il PIC18F4550 e il compilatore SDCC. Notate come nella finestra a sinistra, il Project Manager, compaiono tutti gli elementi del progetto, mentre nella finestra di destra si può vedere il file “test.c”, che si presenta già con il giusto “#include” e le impostazioni standard per i registri del PIC:



Una volta creato il progetto, adesso ci sono due importanti passi: configurare il Piklab per usare il PicKit2 e configurare i registri del PIC18F4550 come piace a noi.

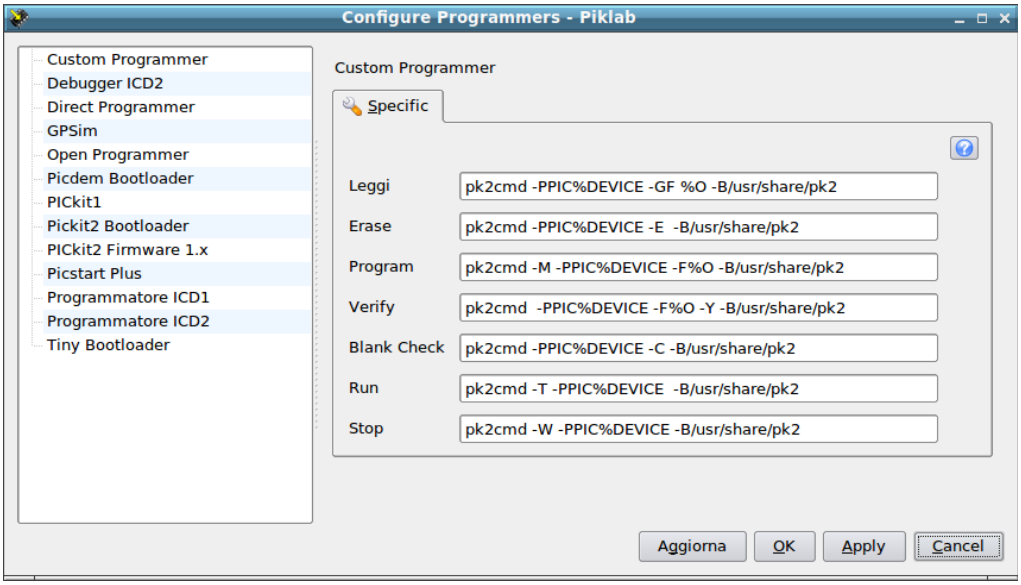
Configurare Piklab per usare PicKit2 con il firmware 2.x

L'utilizzo di PICKIT2 con firmware serie 2.x non è supportato da Piklab, tuttavia è possibile risolvere il problema.

Il programma pk2cmd, visto nella parte iniziale di questo tutorial, utilizzato da linea di comando, può venire integrato direttamente dentro Piklab, configurando quello che nell'elenco dei programmatori appare come “Customer programmer”.

Vediamo i passi da seguire.

1) selezionare la voce di menù Settings/Configure Programmer
2) Appare la finestra di selezione dei programmatori. Selezionare Custom Programmer e quindi impostare le singole voci in modo da utilizzare pk2cmd come da figura:



(come si vede le stringhe inserite sono le stesse che avremmo dovuto digitare da linea di comando)

3) A questo punto il nostro Piklab vedrà il PICKIT2 come un “Custom Programmer” e ogni volta che selezioneremo i comandi per programmare il PIC, verranno mandati in esecuzione i comandi elencati nelle caselle di testo che abbiamo appena configurato. Ad esempio se nella barra strumenti selezioneremo la prima icona a sinistra

(cioè la freccia che entra nel PIC) verrà eseguito il comando

`pk2cmd -M -PPIC%DEVICE -F%O -B/usr/share/pk2`

Gli altri comandi sono associati alle rispettive icone: verificare, scaricare il file dal Pic,

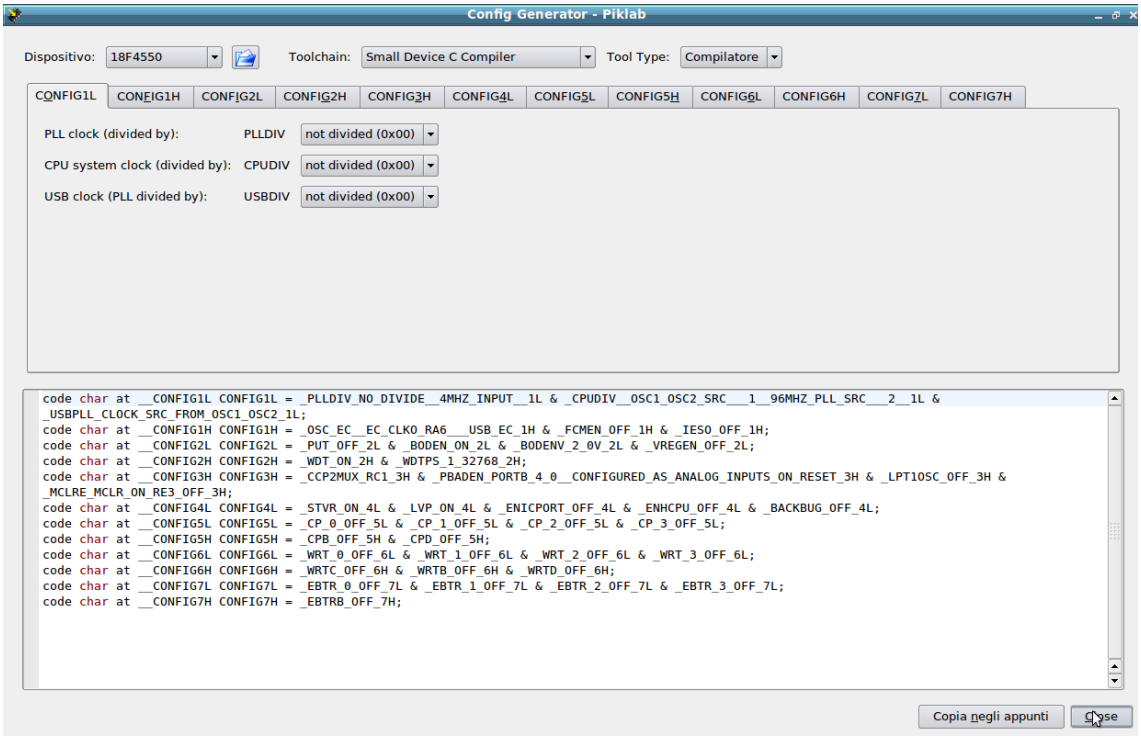
cancellare tutto il PIC, etc etc....

4) Una cosa molto importante da sottolineare sono i parametri che vengono passati a pk2cmd:
    1. %DEVICE è il modello di PIC che abbiamo selezionato nel progetto, ad esempio se stiamo usando il PIC18F4550, allora %DEVICE sarà uguale a 18F4550, per capire cosa ciò significhi rileggere attentamente come si programmava a mano il PIC da linea di comando.
    2. %O indica il nome del file .hex da utilizzare
5) Anche con Piklab per testare il lavoro occorre disconnettere PICKIT2 dalla demoboard.

Impostare i registri del PIC

La configurazione di base che si ottiene quando si crea un progetto ha alcuni “difetti”. Ad esempio, la configurazione standard prevede il “watch dog timer” abilitato, mentre a noi conviene lasciarlo disabilitato. Facendo un altro esempio, molto probabilmente avremo bisogno di impostare la frequenza dell'oscillatore.

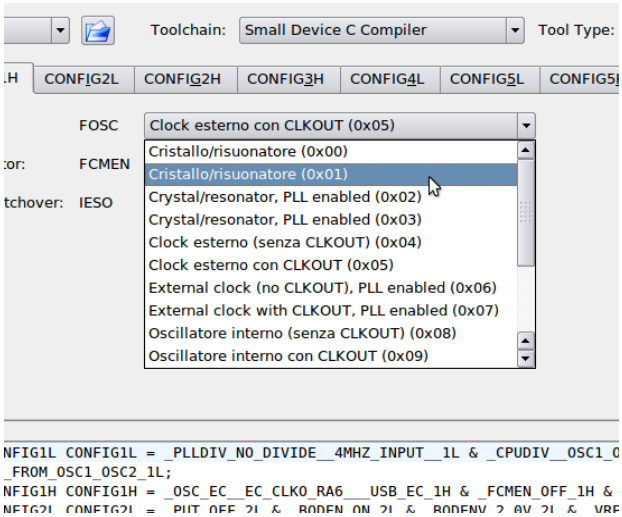
Tutti questi problemi si risolvono con il “Config generator”, che si trova nel menù “Tools”:

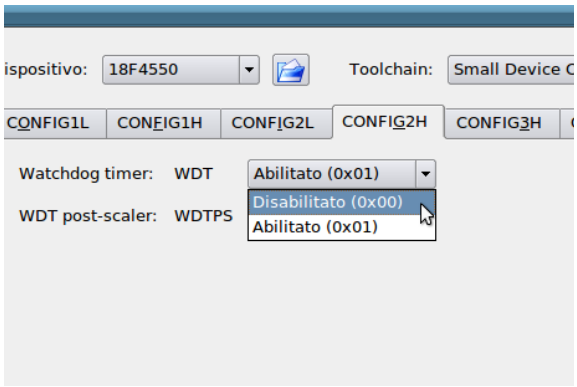


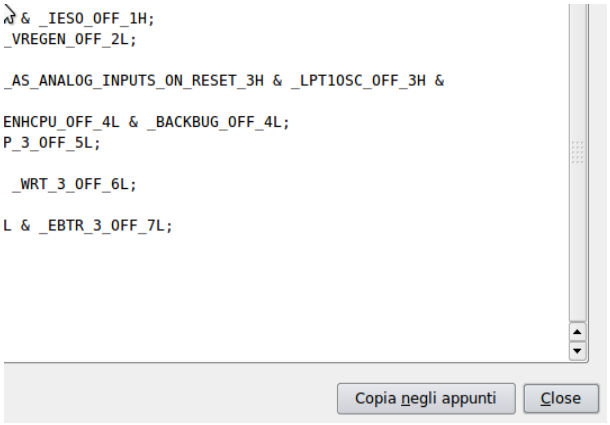
Questo utile strumento consente di impostare i registri del PIC selezionando le opzioni da comodi menù a tendina. Va detto che la finestra che vi apparirà, ovviamente, dipenderà dal tipo di PIC che state utilizzando, in quanto ciascun modello ha i suoi propri registri.

Le tre immagini seguenti mostrano come:

1. impostare l'oscillatore al quarzo come sorgente di clock
2. disattivare il WDT
3. utilizzare il pulsante “Copia negli appunti” per trasferire le impostazioni generate dal configuratore direttamente nel file sorgente “test.c” (in pratica: copiate il contenuto del generatore negli appunti, poi tornate al file test.c, cancellate le righe con la vecchia configurazione e incollate la nuova







A questo punto siete in grado di generare il primo progetto, con le configurazioni che più si adattano alle vostre esigenze. Spero che questa guida possa risultarvi utile per fare i primi passi nel mondo dei PIC con Linux.