di **Cristina Paveri**

Fonte: www.thecsiac.com

Open source e strumenti collaborativi

Vediamo quali vantaggi offre il 'software free e open source' in ambito industriale: sistemi operativi come Linux, software di automazione conformi a IEC 61499, soluzioni a basso costo

Nonostante il timore di divulgare il know-how aziendale, che rappresenta un prezioso capitale di differenziazione dalla concorrenza, il concetto di 'open innovation' si sta affermando anche nel settore dell'automazione industriale, in quanto assicura diversi vantaggi soprattutto in termini economici. In particolare, il Foss 'Free open source software' porta a un risparmio nei costi di sviluppo poiché più aziende, anche se concorrenti, collaborano contemporaneamente (e non in parallelo) al suo sviluppo, ognuna mettendo a disposizione le proprie risorse con la sponsorizzazione delle associazioni di riferimento e spingendosi fino al livello 'divulgabile'. Inoltre, il software Foss permette l'utilizzo, la modifica e la distribuzione senza discontinuità (la licenza non è infatti vincolata a una data di scadenza o a terzi come avviene per i software commerciali); inoltre, in quanto risultato del lavoro congiunto di un'ampia comunità di sviluppatori che possono accedere al codice sorgente, assicura maggiore stabilità e una qualità superiore del risultato. Lo sviluppo da parte di un vasto numero di partecipanti ha anche un altro effetto positivo: garantire la standardizzazione, poiché si basa sull'utilizzo di interfacce e protocolli definiti congiuntamente. Porta quindi a una reale 'globalizzazione' dello sviluppo software.

Linux nell'automazione industriale

Il software free e open-source sempre più utilizzato in numerosi controllori di automazione e nei sistemi embedded, Linux, si è rivelato un sistema operativo affidabile, in grado di fornire servizi senza discontinuità agli utenti finali. È anche una soluzione idonea ad applicazioni più specializzate per diversi motivi, quali i bassi requisiti di CPU, l'efficienza di memoria e la sicurezza nei confronti delle minacce da virus e malware. Linux assicura stabilità e affidabilità, poiché permette di aggiornare parti di un sistema senza dover riavviare l'intero sistema, nonché disponibilità e produttività anche in caso di interruzione di un servizio, perché tutti i servizi funzionano in modo indipendente.
Nello specifico, nei sistemi embedded il codice open source

di Linux permette agli sviluppatori di scalare il sistema operativo per adattarlo alla propria piattaforma target e per soddisfare al meglio i requisiti dell'applicazione specifica. Per contro, Linux richiede conoscenze tecniche maggiori per la minore disponibilità di interfacce grafiche user-friendly tipiche dell'ambiente Windows e presenta dei limiti nel tempo di risposta a causa delle ridotte risorse della CPU. È il caso, per esempio, di una tipica architettura basata su polling (a interrogazione), in cui il sistema embedded deve controllare lo stato di centinaia di I/O a intervalli di millisecondi. Questa limitazione è comunque superabile con l'utilizzo di dispositivi di I/O remoti intelligenti, che determinano autonomamente quando richiedere la comunicazione e inviano rapporti senza la necessità di un polling continuo dello stato degli I/O.

Fonte: img.brajeshwar.com

IEC 61499 e open source

La norma IEC 61131 di standardizzazione dei cinque linguaggi di programmazione dei PLC ha posto le basi per la modellazione dei programmi di controllo. Tuttavia, le piattaforme e gli strumenti di automazione continuano a presentare limiti di portabilità e interoperabilità. Lo stesso comitato tecnico che ha sviluppato la norma ha quindi deciso di stilare uno standard aperto per l'automazione ponendosi obiettivi più ambiziosi quanto a portabilità, ovvero la capacità del software di accettare e interpretare correttamente elementi della libreria generati da altri software; configurabilità, cioè la capacità dei dispositivi e dei relativi componenti di essere configurati con diversi software, e infine interoperabilità, ossia la capacità dei dispositivi di diversi fornitori di funzionare in modo combinato per eseguire le funzioni specificate da una o più applicazioni distribuite.

Lo standard IEC 61499 definisce la realizzazione di intere applicazioni a partire da blocchi funzionali di tipo base o composto (blocchi funzione che contengono a loro volta altri blocchi funzione) per mezzo di algoritmi: ogni blocco funzione presenta sia eventi, sia dati in ingresso e in uscita. In un blocco funzione base l'esecuzione di un algoritmo è attivata dal verificarsi di un evento in ingresso; l'algoritmo in esecuzione genera quindi nuovi dati di uscita a partire dai dati ricevuti in ingresso e, al termine dell'esecuzione, lo stesso algoritmo genera un evento in uscita che potrebbe a sua volta diventare l'evento in ingresso a un altro blocco funzione e così via. Lo standard open IEC 61499 ha dato impulso a diverse iniziative di sviluppo di software di tipo sia commerciale, come IsaGraf, sia open source.

Il software IsaGraf, conforme a IEC 61131 e 61499, è basato su Microsoft Visual Studio e sulla tecnologia plug-in open in base alla quale ogni componente viene sviluppato nella nuova piattaforma di automazione collaborativa ATP (Automation Collaborative Platform).

Per quanto concerne l'open source, l'iniziativa 4Diac ha dato vita a un framework per l'automazione industriale e il controllo distribuito sviluppato per standardizzare le diverse piattaforme proposte da più fornitori, con l'obiettivo di realizzare un ambiente di automazione e controllo conforme a IEC 61499, utilizzabile in diversi settori: building automation, automazione dei processi e di laboratorio, smart grid, controllo macchine ecc. 4Diac è composto dall'ambiente di sviluppo integrato modulare IDE, basato sul framework open Eclipse, dall'ambiente runtime in tempo reale RTE (Forte) per piccoli sistemi embedded (16/32 bit) sviluppato in C++, dalla libreria

Glossario del software free open source (*)

Software libero (Free Software): software che può essere utilizzato per ogni scopo, modificato e adattato alle proprie esigenze, condiviso e disponibile anche per uso, sviluppo e distribuzione commerciali. Si può ottenere software libero pagandolo o non pagandolo, ma rimane sempre la libertà di copiare e modificare il software, persino di venderne copie.

Open Source: modello di sviluppo software collaborativo, in cui chiunque può contribuire fornendo correzioni, migliorie, segnalazioni di errori, traduzioni... Ampiamente adottato da numerosi progetti 'free software'. Non necessariamente il 'software libero' è anche 'open source' e non tutto l''open source' è anche 'software libero'.

GPL (General Public License): è la licenza più diffusa per il software libero e consente all'utente libertà di utilizzo, copia, modifica e distribuzione. Prevede inoltre il vincolo che l'eventuale ridistribuzione di un software modificato mantenga la stessa licenza.

GNU/Linux: la combinazione tra la collezione di applicativi GNU e il kernel Linux è il sistema GNU/Linux.

Linux: è un sistema operativo, alternativo a Windows e MacOS e il primo software libero, ovvero distribuito con una licenza che ne permette non solo l'utilizzo da parte di chiunque e in qualsiasi circostanza, ma anche la modifica, la copia e l'analisi. La differenza sostanziale tra software libero e un software proprietario è il libero accesso al codice sorgente.

GNU GPL: la maggior parte dei progetti Foss attivi utilizza la licenza GNU GPL (General Public Licence), che concede i diritti di utilizzo del software per qualsiasi scopo, studio del funzionamento e modifica del programma per adattarlo alle esigenze dell'utente, ridistribuzione di copie, miglioramento del programma.

(*) Definizioni tratte da www.linux.it/external/VademecumSoftwareLibero.pdf

LIB di blocchi funzione utilizzabili per diversi tipi di applicazioni di controllo e dagli esempi applicativi (Systems).

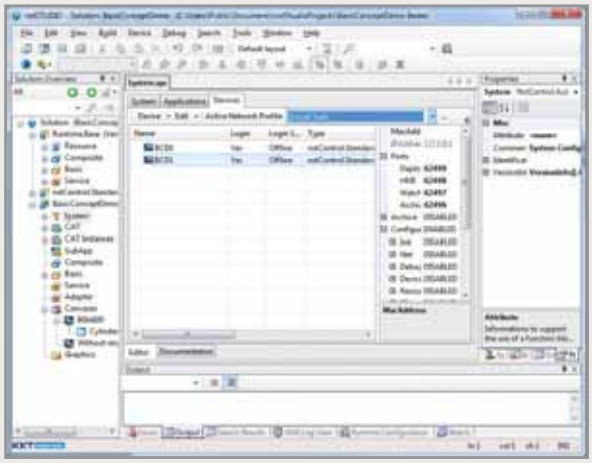
Soluzioni a basso costo

Il basso costo delle piattaforme di sviluppo open source basate su una semplice scheda a microcontrollore, come Raspberry PI e Arduino, che permettono di sviluppare oggetti interattivi e di controllare oggetti fisici quali LED e motori, insieme a un'ampia disponibilità di materiale didattico di supporto hanno contribuito alla diffusione di tali soluzioni all'interno della comunità open source. La sezione 'Newbie' del portale Web Arduino della community di sviluppatori italiani insegna come utilizzare Arduino a partire da zero: dopo aver acquistato la scheda, si scaricano e si installano il software IDE di programmazione e i driver per Arduino. Dopo aver acquisito le basi di programmazione nel linguaggio C e C++, si approfondisce l'elenco di strutture, variabili e funzioni di Arduino (il 'reference' di programmazione). A questo punto si può iniziare a scrivere il codice a partire dagli esempi già disponibili,

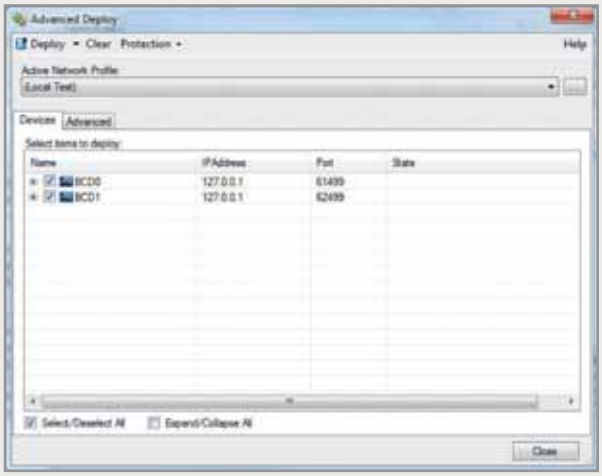
Tutorial per nxtStudio

Vediamo ora come sviluppare una soluzione di automazione conforme allo standard IEC 61499 con il software nxtStudio di nxtControl per il controllo di cilindri pneumatici.

- Per aprire il file 'demo' avviare il programma nxtStudio e fare clic su 'File/Unarchive Solution' nella barra del menu principale. Dopo l'apertura del file, doppio clic sulla voce 'System' del menu 'BasicConceptDemo' nella scheda 'Solution Overview' a sinistra. Passare quindi alla scheda 'Devices' e selezionare 'Local test per Active Network Profile' (figura 1).
- Selezionare 'Yes' nella colonna 'Login' per entrambi i dispositivi, selezionare entrambi i dispositivi dell'elenco e aprire il menu contestuale facendo clic con il tasto destro del mouse nell'area selezionata e selezionare 'Start Soft PLC' dal menu (figura 2).



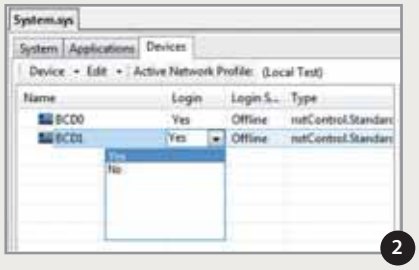
1



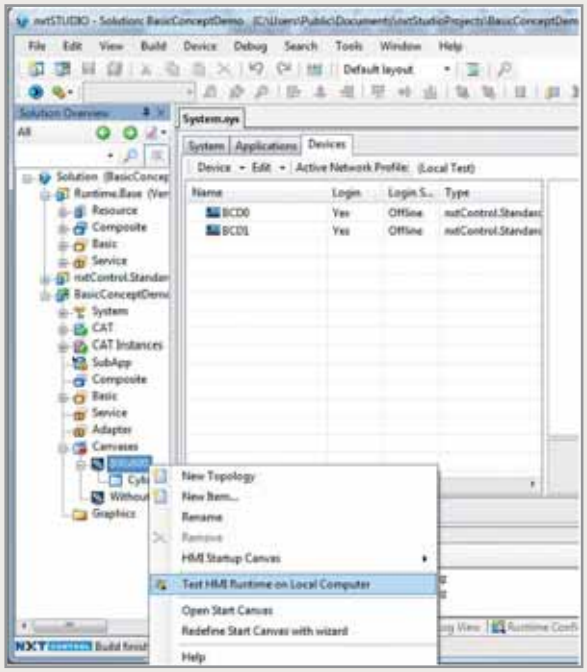
3

- La macchina avvia entrambi i PLC soft. Quindi aprire di nuovo il menu come descritto in precedenza e selezionare 'Deploy' e 'Advanced' (figura 3).
- Nella finestra di dialogo 'Advanced Deploy' selezionare entrambi i dispositivi e fare clic su 'Deploy', poi chiudere la finestra. Ora il progetto è in esecuzione sui PLC soft. Per visualizzare il funzionamento è necessario avviare l'interfaccia HMI espandendo la voce 'Canvases' nel menu di 'Solution Overview' e facendo clic su 'Test HMI Runtime on Local Computer' (figura 4).
- Quando l'interfaccia è in esecuzione ed è collegata correttamente si può controllare il movimento dei due cilindri agendo sul joystick nell'angolo in basso a sinistra dell'HMI (figura 5).

Spostando orizzontalmente il joystick si comanda il movimento del cilindro sinistro, mentre agendo verticalmente si sposta quello destro. Al raggiungimento di uno dei due finecorsa, il LED di controllo corrispondente si accende. Rilasciando il tasto del mouse, il joystick rimane nella posizione impostata e per azzerare la sua posi-



2



4

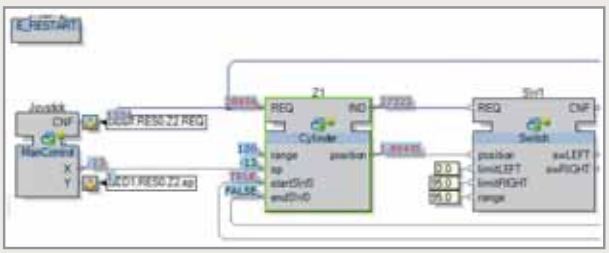
che illustrano l'utilizzo delle funzioni digitali, analogiche e di comunicazione delle strutture di controllo ('if', 'for', 'while', 'switch'), dei sensori ecc.

Fonti: White paper "Free and Open Source Software (Foss): Open Source software and virtualization in the automation industry – a change of fashion or a paradigm change?" Open Source Automation Development Lab; White Paper "How to use Linux in idustrial automation" Moxa; IEC 61499; nxtControl; arduino.cchttp://playground.arduino.cc//Italiano/Newbie

zione è sufficiente fare doppio clic nell'area circolare bianca.
- È possibile anche visualizzare il trend dei cilindri mediante diagrammi temporali della distanza. Con un clic su un cilindro si apre la rispettiva finestra con il grafico del trend (figura 6).
- Per facilitare l'analisi degli errori e ottenere informazioni dettagliate sul progetto, il software permette di controllare interfacce specifiche e visualizzare i valori che sono cambiati (visualizzati in rosso) e quelli inalterati (nero/blu); permette inoltre di aprire un blocco funzione per mostrare il contenuto e aggiungere punti di monitoraggio (figura 7).

5

6



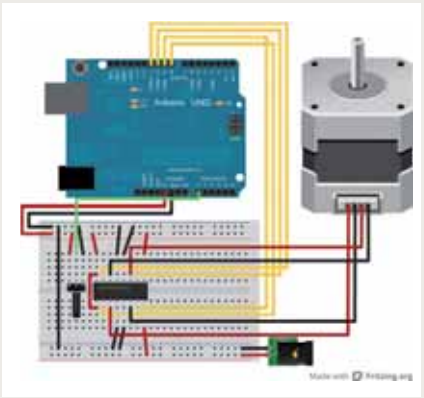
7

Tutorial per Arduino

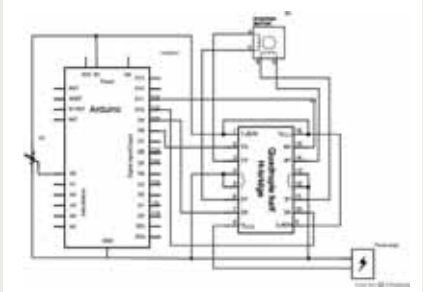
A titolo esemplificativo analizziamo un tutorial dedicato al controllo di motori passo-passo alimentati esternamente nelle versioni unipolare con array Darlington e bipolare con ponte a H mediante un potenziometro da 10 k sull'ingresso analogico 0.

Codice:

```
#include <Stepper.h>
// numero di passi del motore passo-passo
#define STEPS 100
// crea un'istanza della classe del motore passo-passo, specificando
// il numero di passi del motore e i pin
// a cui è collegato
Stepper stepper(STEPS, 8, 9, 10, 11);
// la lettura precedente dall'ingresso analogico
int previous = 0;
void setup()
{
  // imposta il numero di giri del motore su 30 giri/min
  stepper.setSpeed(30);
}
void loop()
{
  // ottiene il valore del sensore
  int val = analogRead(0);
  // muove di un numero di passi pari alla variazione nella
  // lettura del sensore
  stepper.step(val - previous);
  // ricorda il valore precedente del sensore
  previous = val;
}
```



8



9