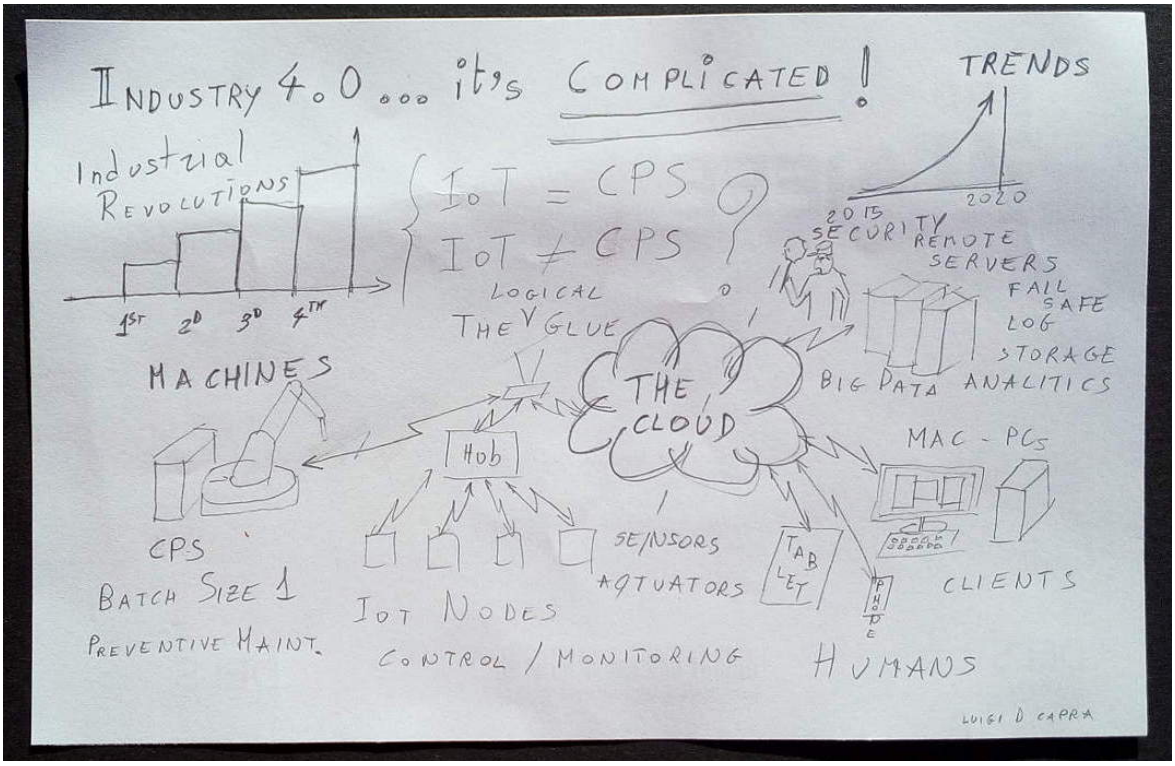
Prototipazione rapida di applicazioni IoT e aggiornamento di applicazioni *legacy* al contesto Fabbrica 4.0.

Luigi D. CAPRA
http://www.luigidcapra.com



Parlare di **Fabbrica 4.0** è difficile poiché si tratta di un coacervo di tecnologie eterogenee e complesse.
Se il termine "coacervo" vi mette a disagio provate a pensare ai vostri interlocutori sentendovi parlare di *Cyber Physical Systems* o di *Internet of Things*.
Il minimo che vi possa succedere è che vi chiedano di spiegare la differenza fra i due concetti.
Quando avrete finito sarà già l'ora di pranzo ed i vostri clienti saranno ben felici di congedarsi adducendo agli impegni che hanno nel pomeriggio…
C'è bisogno di concretezza. Nessuno è disposto a staccare assegni in bianco lasciandosi sedurre dalle sirene della tecnologia. I titolari delle PMI, in particolare, non vogliono più sentire discorsi in generale ma capire, esattamente, quali vantaggi può offrire Fabbrica 4.0 nel loro caso, in riferimento ai loro prodotti e ai loro processi produttivi.
Data queste permesse bisognerebbe andare dai clienti muniti di una "valigetta" di soluzioni già quasi pronte o di un sistema di prototipazione rapida che permetta di abbozzare una soluzione sotto gli occhi dei clienti, invece di chiedere tempo per redigere uno studio di fattibilità. Poi si vedrà!
Poiché è difficile improvvisare delle soluzioni complesse sul momento occorre prepararsi prima, in modo da avere il materiale pronto quando serve.

Per quanto mi concerne, considerate le due tipologie di problemi che più interessano i miei clienti:

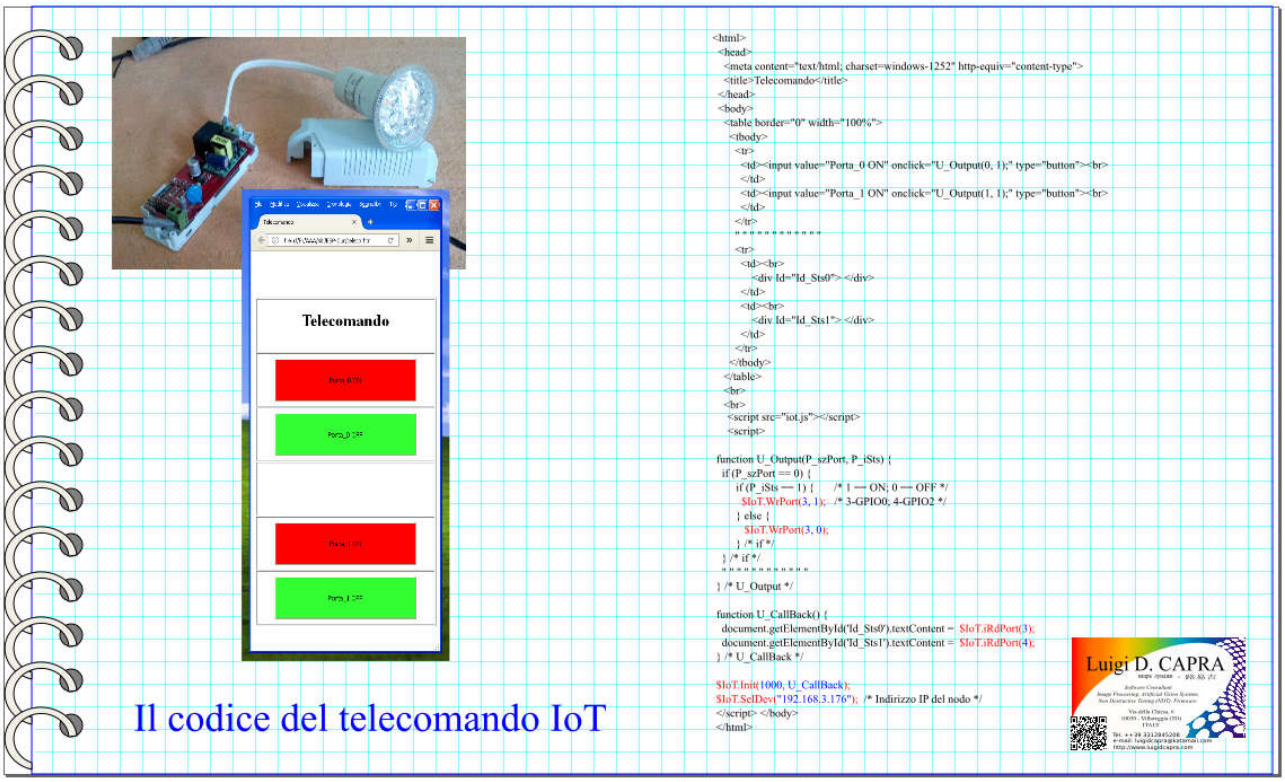
- sviluppo di nuove applicazioni IoT (di cui non sono ben chiare le specifiche e le implicazioni);
- migrazione di HW o SW già esistenti al contesto 4.0 (evitando una riprogettazione completa);

mi sono proposto di dissipare riserve dei clienti circa l'entità dell'impegno richiesto, dimostrando che non solo le competenze richieste per lo sviluppo e la gestione di nuove applicazioni IoT

possono essere ridotte al minimo ma che in molti casi ciò potrebbe essere evitato adeguando i sistemi esistenti alle nuove esigenze con alcuni interventi mirati.
L'originalità delle soluzioni che mi accingo a presentare non risiede nelle tecnologie impiegate ma nell'uso che ne è stato fatto; più precisamente nello sforzo compiuto per riorganizzare il processo di sviluppo minimizzando le competenze richieste, in modo da rendere possibile un maggiore coinvolgimento degli esperti del settore applicativo nello sviluppo delle "loro" applicazioni IoT.

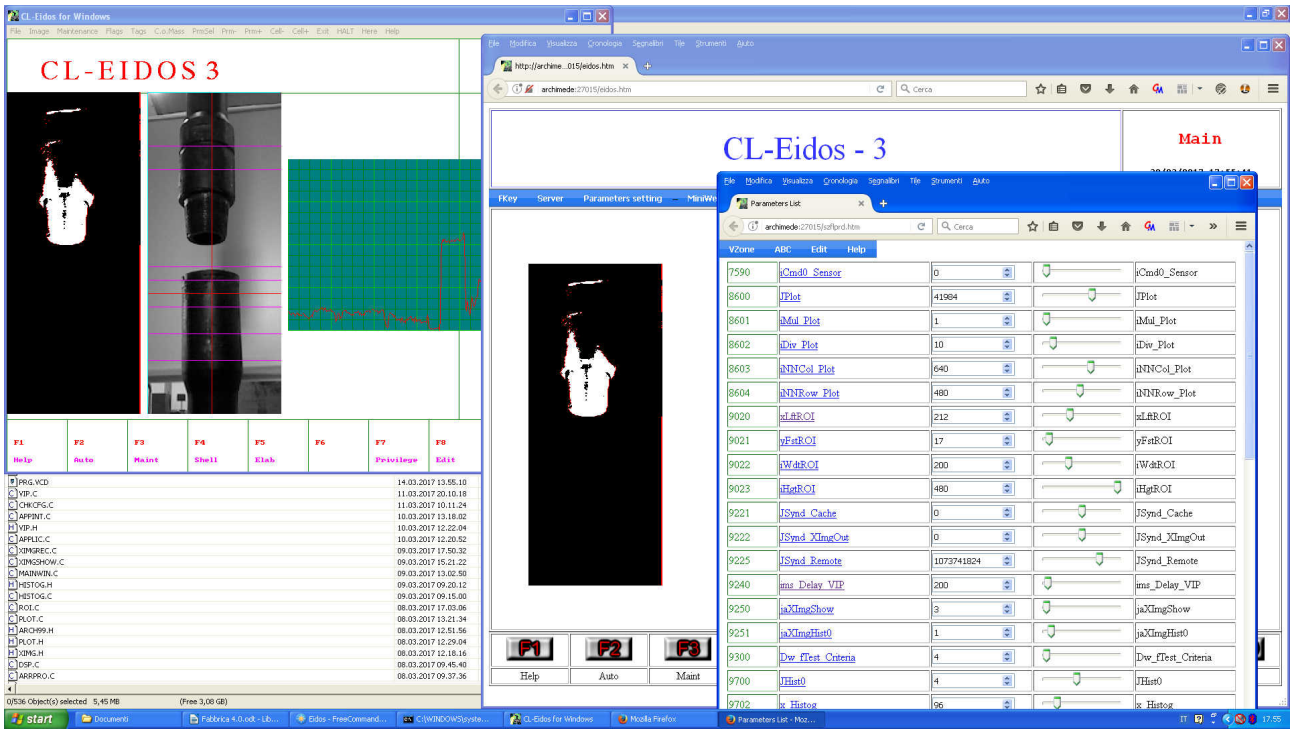
Tale filosofia è particolarmente evidente nell'ambiente di prototipazione rapida *Facciamo/Let's See* (FLM) che idealmente vorrebbe consentire lo sviluppo di applicazioni IoT da parte degli utenti finali.
Perseguendo il suddetto obiettivo mi sono focalizzato sulle schede basate su di un chip piuttosto versatile, l'ESP8266, per le quali ho realizzato un *firmware general purpose* capace di accettare dei comandi via TCP/IP e restituire delle risposte. Come logico complemento del precedente ho poi sviluppato un IDE costituito da: un terminale *wireless* (utilizzabile ad esempio per effettuare la diagnostica dei nodi senza essere costretti ad accedere ad essi fisicamente), un generatore di codice (che facilita la configurazione dei parametri di controllo del FW) e uno strumento per la gestione di gruppi di nodi a livello di sciame.
Previa disponibilità di schede dotate di FW preinstallato, volendo improvvisare una rete IoT, basta collegare l'alimentazione e configurarle assegnando loro gli indirizzi IP.
Dopodiché, se è previsto che che tutti i nodi si comportino allo stesso modo, basterà selezionare il file di configurazione preparato in precedenza e confermare. L'IDE si incaricherà di fare il resto caricando lo *script* sui nodi selezionati.
Nel caso in cui fosse invece richiesta una programmazione specifica, i nodi dovranno essere indirizzati e configurati individualmente. Anche in questo caso l'IDE viene in soccorso dell'operatore fornendo un apposito pannello per la configurazione dei parametri. Al termine basterà premere un tasto di conferma per avviare il processo di generazione del codice e l'*uploading* nel nodo prescelto.

Ma non è tutto poiché FLM offre un ulteriore *bonus*. Essendo scritto in un formalismo ben conosciuto (HTML-5 + JavaScript) e rilasciato con licenza *open source*, il codice di FLM si presta ad essere modificato ed impiegato come punto di partenza per lo sviluppo di nuove applicazioni.
In occasione del Linux Day 2016 ad Ivrea, ho presentato una semplice applicazione IoT basata sulle librerie FLM, che consentiva di accendere e spegnere una lampadina a distanza utilizzando un telefono cellulare come telecomando, realizzata da un ragazzino di 14 anni, che aveva sulle spalle non più di sei ore di lezione di programmazione!
Come si può vedere in figura, le librerie FLM hanno consentito di ridurre significativamente la complessità del problema. Il codice IoT è costituito dalle poche istruzioni evidenziate in rosso, le restanti linee di programma definiscono la grafica.



All'estremo opposto, considerato il problema consistente nell'adeguamento al contesto Fabbrica 4.0 di applicazioni e macchine industriali già esistenti, ho sviluppato un kit incentrato su di un mini server web che si propone di minimizzare il tempo richiesto per consentire il monitoraggio e/o controllo remoto di sistemi *legacy* evitando di riscriverli completamente.
Il tutto si basa sull'osservazione che le funzioni di monitoraggio e controllo e più in generale l'interazione uomo-macchina o fra due macchine (M2M) coinvolgono generalmente un numero limitato di variabili di programma che riepilogano lo stato del sistema.
Per adeguare un sistema esistente dotandolo di connettività occorrerà in primo luogo individuare le variabili di cui sopra e quindi sviluppare le routine atte a consentire la loro lettura e scrittura in risposta ad eventi esterni.
Un compito apparentemente semplice, se non fosse per la difficoltà di reperire le specifiche dei vecchi sistemi, per cui è sempre presente il rischio di malfunzionamenti riconducibili a vincoli di cui non si è tenuto conto poiché se ne ignorava l'esistenza.
L'alternativa che ho elaborato si basa sul principio in base al quale il funzionamento degli eseguibili deve essere indipendente dall'ordine di allocazione delle variabili (salvo esplicita richiesta di assegnare di indirizzi fissi ad alcune di esse), quale premessa per il funzionamento delle routine di ottimizzazione del codice.

Sotto queste premesse è consentito a chiunque (compilatore, linker ed anche a noi) di scambiare o spostare le variabili in memoria, trasferendole persino in un altro segmento, magari in una *shared memory* accessibile dall'esterno.
Il trasferimento delle variabili di I/O in una memoria condivisa è un intervento assai meno invasivo rispetto a quello delineato in precedenza poiché si ha la garanzia che il programma *legacy* continuerà a funzionare normalmente se viene eseguito da solo; d'altro canto introduce la possibilità di monitorare e/o controllare lo stato dell'applicativo per mezzo di un secondo processo avviato in parallelo, come fanno i *debugger.*
La soluzione proposta offre numerosi vantaggi rispetto quella consistente nell'integrazione delle funzioni di comunicazione nel programma aggiornato, poiché preserva le funzionalità dell'applicativo originale, evitando i rischi di destabilizzazione connessi alla redistribuzione del tempo macchina disponibile fra le *thread* del SW; ma il principale beneficio consiste nel drastico abbattimento delle tempistiche che si ottiene ricorrendo ad un modulo preconfezionato evitando lo sviluppo di un nuovo modulo complesso.



La soluzione è risultata sufficientemente performante da consentire l'aggiornamento di un complesso programma di visione artificiale al fine di consentirne la gestione remota.

Bibliografia e Riferimenti:

http://www.luigidcapra.com/

Dizionario di IoT e Fabbrica 4.0
Una breve introduzione a Fabbrica 4.0: rischi e opportunità per le PMI (PDF)