Node-RED, flow-based programming per l'Internet of Things

Flow-based programming, introduzione

Utilizzare flussi di informazioni per realizzare, in modo visuale, veri e propri programmi per sfruttare i dati generati con board Arduino (o Raspberry PI

PP

Paolo Patierno
Pubblicato il 1 dic 2014

Quando parliamo di **Flow-Based Programming** ci riferiamo ad un paradigma di programmazione. Tutti i linguaggi sono creati facendo riferimento a specifici paradigmi con i quali si stabiliscono lo stile, le modalità, i componenti fondamentali e l'interazione tra questi per la realizzazione del software.

In questa lezione cominciamo quindi a considerare un approccio alla programmazione che può aiutare in molti casi anche lo **sviluppo in breve tempo di applicazioni complesse** (magari prototipali).

Inizieremo poi a parlare in maniera più approfondita di Node-RED e di quale sia il suo ruolo nell'Internet of Things. Prima di addentrarci in **Node-RED** però, ci sembra utile passare in rapida rassegna i principali paradigmi di programmazione in uso:

Campo	Descrizione
Structural programming (programmazione strutturata)	Storicamente contrapposto al noto fenomeno "spaghetti code", propone di evitare l'utilizzo sconsiderato di salti nel codice (istruzioni come il "goto") che rendono difficile la leggibilità e la manutenibilità del sorgente. Essa pone le sue fondamenta in una serie di costrutti che rappresentano la base anche per gli altri paradigmi : subroutine, blocchi e le strutture di controllo. Tra i più noti linguaggi che supportano paradigma c'è il Pascal, non più ampiamente utilizzato;
Procedural programming (programmazione procedurale)	Deriva dalla programmazione strutturata e si basa appunto sul concetto di "procedura" (anche nota come 'funzione', routine o subroutine) che contiene una sequenza di istruzioni che definiscono i passi caratteristici dell'applicazione. Ciascuna procedura può essere chiamata più volte in differenti punti del programma (anche all'interno di altre procedure). Il Pascal può considerarsi anche procedurale ma il principe di questo paradigma è senza ombra di dubbio il linguaggio C, utilizzato moltissimo per lo sviluppo dei sistemi embedded che rientrano anche nell'ambito dell'Internet of Things.
Object-Oriented Programming (OOP - programmazione orientata agli oggetti)	Si basa sul concetto di "classe" e delle istanze note come "oggetti". Una classe rappresenta un'entità che ha un proprio stato caratterizzato da campi (quindi dati) ed un proprio comportamento descritto attraverso i cosiddetti metodi, che possono determinare una variazione dello stato interno quando invocati (dall'esterno o dall'interno della classe stessa). Ad oggi, Sono numerosi i linguaggi che appartengono a questo paradigma da quelli storici come il C++ fino a quelli relativamente moderni come Java e C#, utilizzati in numerosi ambiti dalle applicazioni mobile a quelle Web, passando per le applicazioni desktop.
Functional programming (programmazione funzionale)	Paradigma "dichiarativo" strettamente legato al concetto di funzione matematica che evita la relazione con lo stato (presente ad esempio nella OOP) ed il suo cambiamento (considerato un "side effect") in modo tale che il valore di ritorno di una funzione dipenda esclusivamente dai parametri di ingresso della stessa. Uno dei principali esponenti di questa categoria è il linguaggio Python anche se considerato impuro perché contiene caratteristiche dei linguaggi "imperativi".

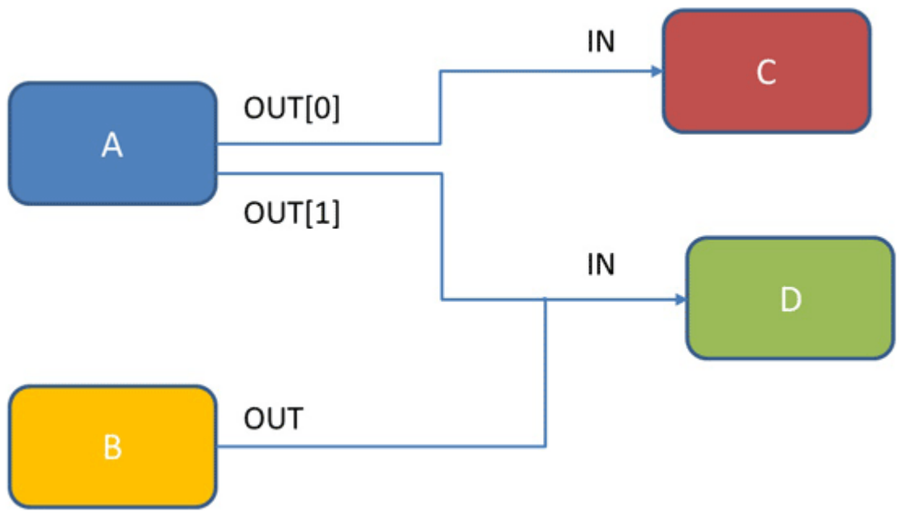
Flow-Based Programming

L'elenco potrebbe considerarsi concluso ma qui si innesta invece il nostro paradigma basato su flussi.

Questo paradigma stabilisce che un'applicazione sia costituita da una serie di blocchi dei quali dobbiamo conoscerne il tipo di elaborazione eseguita ma non necessariamente la corrispondente implementazione interna; per questo motivo essi possono essere considerati come "**black box**".

Tali blocchi utilizzano una o più **porte** (sia in ingresso che in uscita) per poter essere collegati tra loro attraverso delle connessioni in modo da costituire una rete nell'ambito della quale comunicano scambiandosi dati sotto forma di messaggi noti anche come "information packets" (**IP**).

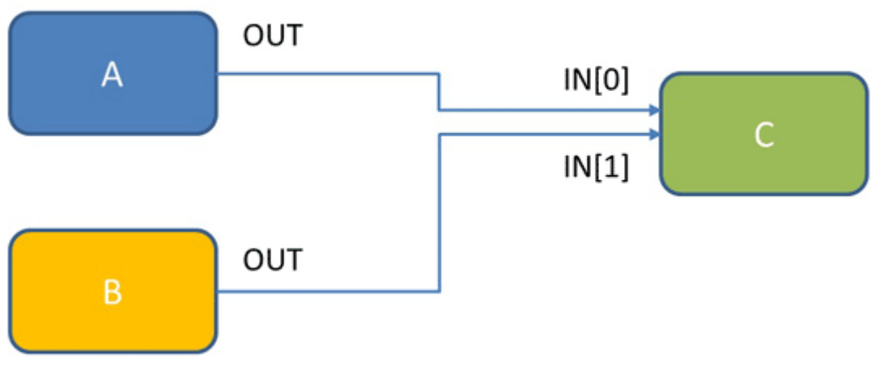
Flow con blocco ad una o più porte di uscita



I blocchi possono essere riutilizzati senza alcun cambiamento interno per poter sviluppare altri programmi che richiedano le medesime elaborazioni in determinati contesti.

Abbiamo dunque la possibilità di avere più blocchi che eseguono le medesime elaborazioni nello stesso programma, ciascuno in modo indipendente dagli altri anche in termini di memoria di lavoro e di risorse utilizzate (a meno di casi particolari in cui potrebbero sussistere anche delle risorse condivise).

Flow con blocco a più porte di ingresso



Gli "**informarion packets**" possono trasportare con se del contenuto informativo ed essere quindi dei dati a tutti gli effetti oppure possono essere usati semplicemente come dei segnali, per poter innescare l'avvio di un processo relativo al blocco cui sono destinati.

Il programma non è più una sequenza di istruzioni ma è caratterizzato da un insieme di **flussi di dati** che vengono scambiati tra i blocchi in maniera completamente asincrona.

L'esecuzione viene garantita da un software noto come "**scheduler**" che può o meno sfruttare le caratteristiche della piattaforma in cui si trova per garantire anche l'eventuale parallelismo nell'esecuzione dei processi su più flussi.

Il concetto di flussi separati è determinato proprio dal fatto che l'applicazione può avere più ingressi indipendenti attraverso i quali far fluire i dati attraverso i blocchi sfruttando le relative connessioni. Le "strade" percorse dai dati possono essere molteplici e parallele.

I tool di sviluppo visuali: Node-RED e l'Internet of Things

Data la sua natura basata sul concetto di rete (blocchi e connessioni), la programmazione a flussi è molto spesso realizzata con tool di tipo visuale tra i quali, come vedremo nel corso di questa guida, spicca Node-RED con la sua focalizzazione al mondo dell'Internet of Things. Va sottolineato che i blocchi di ciascun flusso sono comunque caratterizzati da codice da eseguire, per cui lo sviluppo di un'applicazione prevede due livelli : quello visuale e quello implementativo per ciascun blocco.

Nel caso particolare di Node-RED, la rappresentazione visuale è realizzata attraverso HTML (ed in parte anche JavaScript) mentre la parte implementativa ed elaborativa esclusivamente in JavaScript, in quanto tutto si basa sul framework NodeJS (JavaScript server side). Ovviamente, l'ambiente in cui "vive" il tool di sviluppo non può che essere un normalissimo browser !

Node-RED : flow-based programming per l'Internet of Things