IOT Review

Francesco Tarsia

ftarsia@hotmail.com

.

Internet Of Things

Con lo sviluppo delle varie tecnologie di comunicazione si sta avendo un continuo aumento di dispositivi che possono accedere alla rete ed interagire con essa.
Con **Internet Of Things (IOT)** si intende una rete di oggetti interconnessi, individuati in modo univoco, che possono comunicare tra loro. Uno strumento importante per lo sviluppo dell'IOT è il web 3.0 o web semantico.
Il web 3.0 si basa sull'unione del web 2.0 (in cui gli utenti possono, a differenza del web 1.0 o web statico, interagire condividendo con altri utenti contenuti da loro stessi creati) con il web semantico, ossia un web in cui ciascuna risorsa è descritta utilizzando XML e è messa poi in relazione con altre risorse attraverso dei link.

Wireless Sensor Network (WSN)

Grazie allo sviluppo che sta avendo la tecnologia *wireless*, le WSN sono utilizzate sempre più per operazioni di *sensing*. I nodi di una WSN sono dei sensori, disposti all'interno di un ambiente, con lo scopo di rilevare determinati dati, inviarli a un sensore con capacità di elaborazione detto *sink* (concentratore), affinchè essi vengano elaborati. La rete di sensori è una rete dinamica che riesce a supportare anche un incremento del numero di sensori connessi oppure anche una diminuzione degli stessi dovuta a dei guasti.
Quello che sino ad ora è stato chiamato nodo sensore generalmente è conosciuto anche con il termine ***mote***, coniato dai ricercatori dell'Università di Berkley in California nel 1998 con il progetto Smartdust.
Un ***mote*** è un dispositivo di elaborazione in miniatura dotato di memoria, di un apparato ricetrasmittente, di sensori per la rilevazione di dati ambientali e di una batteria per l'alimentazione. I ***mote*** vengono progettati in modo che, una volta accesi e inseriti nel luogo di funzionamento, siano in grado di operare a lungo senza intervento umano.
I ***mote*** sfruttano le trasmissioni radio per comunicare e per trasmettere i dati raccolti. L'assenza di cavi di alimentazione e di trasmissione dati facilita il dispiegamento delle WSN, consentendone l'utilizzo in tutte quelle situazioni dove i tradizionali sensori cablati non possono essere impiegati efficacemente.

Sensori

I *mote* hanno sia sensori built-in sia un bus di espansione per mezzo del quale è possibile aggiungere altri sensori necessari per svolgere funzioni specifiche.
Un sensore è composto generalmente da un trasduttore e da un convertitore di segnali analogici in digitali. I trasduttori sono costruiti sfruttando le caratteristiche di certi materiali che variano le loro caratteristiche elettriche al variare delle condizioni ambientali. Un **ADC** (*Analog to Digital Converter*) converte il valore di tensione su un trasduttore in un valore binario che verrà poi utilizzato per le successive elaborazioni.
Molti dei trasduttori utilizzati sui *mote* sono dei **MEMS** (*MicroElectroMechanical Systems*): si tratta di dispositivi in grado di rilevare una vasta gamma di fenomeni fisici in maniera efficiente ed economica dal punto di vista energetico.

Attualmente possono essere reperiti sul mercato svariati tipi di mote. I più comuni sono:

- Famiglia Mica (Mica, Mica2, Mica2dot, MicaZ) della Crossbow Technology Inc.
- Famiglia Telos (Telos, Telosa, Telosb, TmoteSky) della Moteiv Corporation.
- Intelmote2 della Intel.
- Famiglia Eyes (EyesIFX, EyesIFXv1, EyesIFXv2) della Ember Corporation.
- IRIS della Crossbow Technology Inc.
- Famiglia Fleck (Fleck1, Fleck2, Fleck3, FleckNano) della CSIRO ICT Centre.

Per quanto riguarda il software non vi è un unico sistema operativo ma ve ne sono diversi, tra cui i più diffusi sono TinyOS. **NesC** è stato il primo linguaggio utilizzato per scrivere le applicazioni per *mote*. TinyOS è diventato uno standard industriale, sia per essere stato il primo sistema operativo per WSN sia perchè i suoi sorgenti sono pubblici e liberamente utilizzabili. I *mote* sono dei dispositivi di elaborazione particolari, perchè "dormono" per la maggior parte del tempo. L'elaborazione svolta è guidata dagli eventi: solo quando il sensore acquisisce nuovi dati oppure quando vengono ricevuti dei messaggi si avvia un processo di calcolo.
L'unità base di programmazione e di compilazione nel linguaggio NesC è il "componente", ogni programma è ottenuto assemblando uno o più componenti tra loro. Ogni componente è specificato da una o più interfacce, le quali, in modo simile a quanto fatto da altri linguaggi, dichiarono un insieme di funzioni e procedure da implementare chiamate "comandi".

Confronto tra diversi hardware per WSN

Piattaforma per reti di sensori	**EyesIFX v.2.0 Ember Corporation**	**TelosB Moteiv Corporation**	**MICA-Z Crossbow Technology Inc**	**BTnode ETH Zurigo**	**Squidbee Libelium**
Tensione di alimentazione	3 V	2,1÷3,6 V	2,7÷3,3 V	2÷3 V	2.8÷3.4 V
Consumo di corrente in modalità di risparmio energetico/con radio in trasmissione/con radio in ricezione	9 µA/12 mA/9,42 mA	21 µA/21 mA/23 mA	20 µA/11 mA/19.7 mA	100 nA/5-24 mA/7.4-9.6 mA	10 µA/45 mA/50 mA
Massima potenza in trasmissione	4 dBm	-24 dBm a 0 dBm	-24 dBm a 0 dBm	-20 dBm a - 10 dBm	+ 0 dBm
Modulazione	FSK	O-QPSK	O-QPSK	FSK	O-QPSK
Frequenza di trasmissione	868,3 MHz	2.4-2.48 GHz ISM	2.4-2.48 GHz ISM	433-915 MHz	2,4 GHz 915 MHz 868 MHz
Raggio massimo di copertura outdoor/indoor	~80 m/~40 m	75-100 m/20-30 m	75-100 m/20-30 m	300 m/10 m	90 m/30 m
Velocità di trasmissione	19200 bit/s	250 Kbits/s	250 Kbits/s	0.6-76.8 Kbits/s	20-250 Kbits/s
Capacità di memoria RAM/ROM	10 KB/48KB	10 KB/45KB	4KB/128KB Program Flash Memory	4 KB RAM, 4 KB EEPROM, 128 KB Flash-ROM, 256 KB external SRAM	Flash: 16 Kbytes, EEPROM: 512 bytes, SRAM: 1Kbytes
Trasduttori	Luce, Temp., RSSI	Luce, Temp., Umidità	Sicurezza, Acustica, Video	Luminosità, Temp., Acceler., Suono	Luce, Temp., Umidità
Costo	~70€	77 €	77 €	150 €	120 €
Processore	TI MSP430	TI MSP430	ATMega128L	ATMega128L	ATMega168
Radio chip	TDA 5250	TI - CC 2420	TI-CC2420	CC1000	Xbee – DiGI
Sistema Operativo	TinyOS	TinyOS	TinyOS	TinyOS	NO-caricato da bootloader

Un denominatore comune della tabella è che diverse famiglie di mote utilizzano come microprocessore (µP) un ATmega128L/168 dell'azienda californiana Atmel. La piattaforma Arduino si basa sulla famiglia di chip per µP AVR ATmega di Atmel. I microcontrollori sono il motore che fa funzionare i sensori, sono l'ultimo baluardo della elaborazione dati a 8 bit in un mondo che è già passato ai 32 bit e oltre. I µP hanno funzionalità molto limitate e per questo motivo si utilizzano ancora componenti a 8 bit, anche se il prezzo dei µP a 32 bit sta scendendo. La RAM dei µP si misura in Kbyte e l'unità di storage in decine di Kbyte. Nonostante queste limitazioni, i risultati che si possono ottenere sono notevoli, i chip sono molto più piccoli rispetto a quelli degli anni Ottanta, consumano meno energia e lavorano con una velocità che è quasi cinque volte superiore a quella dei componenti degli anni Ottanta.
A differenza del mercato dei processori per personal computer, dominato da due produttori (Intel e AMD), quello dei µP è composto da molti produttori (Atmel, Microchip, NXP, Texas Instruments, solo per citarne alcuni), ciascuno dei quali propone una serie di chip da impiegare in applicazioni diverse tra loro.

System-on-Chip

Tra un µP di basso livello e un personal computer si collocano i sistemi SoC. Come il µP, i SoC associano in un unico chip un processore e molte periferiche, cui si aggiungono altre funzionalità. L'unità di storage dei SoC non è di solito inclusa nel chip e prevede tipicamente l'impiego di SD card.

Le funzionalità avanzate dei SoC richiedono la presenza di un sistema operativo. Vi sono molti sistemi operativi, di tipo closed o open source, forniti da produttori di soluzioni embedded e dai principali produttori di sistemi operativi, per esempio Microsoft e Linux.

Scelta della piattaforma

Scegliere la piattaforma adeguata per il device dell'Internet delle cose è una questione complessa. Ciò non significa che è impossibile fare la scelta più corretta, ma va ricordato che ci sono tante risposte quanti sono i device disponibili.
La piattaforma dipende dalla combinazione di molti fattori, tra cui il prezzo, prestazioni e funzionalità che meglio si adattano al raggiungimento dei nostri obiettivi. Si deve iniziare con la scelta di una piattaforma per creare il prototipo.

Velocità del processore

La velocità di clock del processore indica la rapidità di esecuzione delle singole istruzioni nel codice macchina del programma in esecuzione. Questo vuol dire che un processore con una velocità superiore è in grado di eseguire le istruzioni in meno tempo. La velocità di clock è il parametro più semplice che misura la capacità di calcolo, anche se non è l'unico. Si può fare affidamento sul valore di MIPS (*Millions of Instructions Per Second*), un parametro riportato sulle specifiche o nel datasheet delle varie piattaforme.
In generale, la velocità del processore è uno dei fattori che si mettono a confronto per valutare sistemi simili. I µP hanno di solito velocità di clock espresse nell'ordine delle decine di MHz, mentre le schede SoC lavorano a centinaia di MHz e perfino con velocità nell'ordine dei GHz.
I µP vanno bene in tutte quelle situazioni dove non è richiesto un carico eccessivo di elaborazione dati, per esempio quando si devono effettuare semplici trasmissioni in rete e misure con i sensori. Se invece il device deve elaborare una grande quantità di dati, per esempio nella gestione di video in tempo reale, è preferibile affidarsi alla velocità di una piattaforma SoC.

RAM

La RAM è la memoria di lavoro del sistema. I µP con una RAM inferiore a 1KB sono poco interessanti, e per eseguire i protocolli di cifratura standard servono almeno 4 KB (meglio di più).
Nelle schede SoC è preferibile una RAM di almeno 256 MB, soprattutto se si vuole installare un sistema operativo Linux.

Connessione in rete

La modalità di connessione del device con il resto del mondo è un fattore determinante per ogni prodotto dell'Internet delle cose. Un collegamento Ethernet cablato è spesso la soluzione più immediata per l'utente (è di tipo *plug and play*) ed è economica, ma richiede la presenza di un cavo fisico. Le opzioni wireless escludono l'impiego di cavi, ma necessitano di una configurazione più complessa.
La soluzione Wi-Fi è la più diffusa quando si tratta di predisporre un'infrastruttura per le connessioni in rete, anche se può essere più costosa ed è meno ottimizzata in termini di consumo di energia.
Esistono tecniche wireless a corto raggio che hanno un minore consumo di energia o che hanno costi più contenuti del Wi-Fi, ma in genere hanno una larghezza di banda più piccola. ZigBee è una tecnologia di questo tipo, orientata principalmente per le reti di sensori e per l'home automation. Il protocollo Bluetooth LE (noto anche come BLE acronimo di Bluetooth Low Energy o Bluetooth 4.0) è simile a ZigBee per quanto riguarda il consumo di energia, sta avendo una rapida diffusione perchè comprende i chip Bluetooth standard inclusi nei telefoni cellulari e nei portatili.
Il mercato offre anche soluzioni "pesanti" ma economiche, per esempio, lo standard RFM12B funziona nello spettro di frequenze radio a 434 MHz, invece che nello spettro a 2,4 GHz adottato per esempio nei router wireless domestici.
Per quanto riguarda le trasmissioni remoti o in ambienti esterni, la soluzione più semplice implica l'utilizzo delle reti di telefonia mobile. Per le comunicazioni in banda larga ridotta e con alta latenza si può impiegare un servizio di base come quello degli SMS; per le trasmissioni più veloci si utilizzano le connessioni dati degli smartphone, per esempio la tecnologia 3G.

Queste sono alcune delle principali tecnologie di comunicazione che il mercato offre agli sviluppatori:

- **Bluetooth;**
- **ZigBee;**
- **Z-Wave;**
- **6LoWPAN;**
- **Thread;**
- **WiFi;**
- **Cellular - GSM/GPRS/EDGE (2G), UMTS/HSPA (3G), LTE (4G);**
- **NFC;**
- **Sigfox;**
- **Neul;**
- **LoRaWAN.**

Sigfox

Una tecnologia WAN alternativa è Sigfox, che in termini di gamma si frappone tra Wi-Fi e cellulare. Esso utilizza le bande ISM, quelle libere che si possono usare, senza la necessità di acquisire le licenze, per la trasmissione di dati su uno spettro molto stretto da e per oggetti connessi. Sigfox nasce dall'idea che per molte applicazioni M2M[1] che girano con una piccola batteria e che richiedono bassi livelli di trasferimento dati, la tecnologia Wi-Fi viene utilizzata per un periodo breve, mentre la tecnologia cellulare è troppo costosa e consuma anche troppa energia.
Sigfox utilizza una tecnologia chiamata UNB (*Ultra Narrow Band*) ed è progettato solo per gestire basse velocità di trasferimento dati da 10 a 1000 bit al secondo. Consuma solo 50 microwatt rispetto al 5000 microwatt per la comunicazione cellulare, oppure può fornire un tipico stand-by di 20 anni con una batteria di 2.5Ah mentre è solo 0,2 anni per un cellulare.
Già distribuito in decine di migliaia di oggetti connessi, la rete è attualmente in fase di implementazione nelle principali città di tutta Europa, tra cui dieci città del Regno Unito. Sigfox è adatto per varie applicazioni M2M dove si prevede di inserire contatori intelligenti, monitoraggio dei pazienti, dispositivi di sicurezza, illuminazione stradale e sensori ambientali in aree di diversi chilometri quadrati.
Il sistema Sigfox utilizza ricetrasmettitori wireless EZRadioPro di Silicon Labs (*EZRadioPro wireless transceivers*), che offrono prestazioni leader del settore wireless, e un consumo energetico molto basso per le applicazioni di rete wireless che operano nella banda sotto 1 GHz.

Neul

Simile nel concetto a Sigfox e operante nella banda sotto 1 GHz , Neul fa leva sulle molto piccole slice messe a disposizione dallo spazio bianco dello spettro della TV per fornire elevata scalabilità , alta copertura, bassa potenza e reti wireless a basso costo . I sistemi sono basati sul chip Iceni, che comunica utilizzando lo spazio bianco radio per accedere allo spettro UHF di alta qualità, ora disponibile grazie alla transizione dalla TV analogica alla TV digitale. La tecnologia delle comunicazioni è chiamata Weighless, che è una nuova tecnologia di rete wireless WAN progettata per l' Internet delle cose che compete in gran parte contro GPRS esistenti , soluzioni 3G, CDMA e LTE WAN. Il data rate può essere qualsiasi, da pochi bit per secondo fino a 100 Kbps sullo stesso collegamento singolo; e i device consumano poco, da 20 a 30 mA con batterie 2xAA, il che significa dai 10 a 15 anni.

LoRaWAN

Simile per certi aspetti a Sigfox e Neul , LoRaWAN è orientato verso applicazioni di rete su reti geografiche (WAN) ed è progettato per fornire WAN a bassa potenza con caratteristiche specificamente necessarie per sostenere a basso costo una comunicazione bidirezionale sicura per l'Internet delle cose, M2M, smart city e applicazioni industriali.
Ottimizzato per il consumo a bassa potenza e per sostenere reti di grandi dimensioni , con milioni e milioni di dispositivi , la velocità di trasferimento dati varia da 0.3 Kbps a a 50 Kbps.

[1] M2M è acronimo di Machine to Machine e indica le tecnologie e i servizi che permettono il trasferimento automatico delle informazioni da macchina a macchina con limitata o nessuna interazione umana.

Un discorso a parte merita 6LoWPAN basato sul Protocollo Internet IPv6 (Internet Protocol version 6) che dovrebbe sostituire IPv4 negli anni avvenire. La standardizzazione del protocollo IPv6 avvenne quando poche persone avrebbero potuto prevedere che i 2^{32} = 4.294.967.296 (4,3 miliardi) di indirizzi ammessi da IPv4 sarebbero andati esauriti, e la diffusione dell'Internet delle cose potrà solo accelerare questa tendenza.
Si calcola che nel 2020 ci saranno 28 miliardi di oggetti collegati in Rete, esclusi pc, tablet e smartphone. Trenta volte più che nel 2009.
Il protocollo IPv6 utilizza indirizzi a 128 bit, lo spazio di indirizzi (2^{128}) è talmente grande che consente di assegnare lo stesso numero di indirizzi ammessi dall'intero IPv4 a ciascuna persona che vive sul pianeta.
Il nuovo standard venne studiato negli anni Ottanta e fu rilasciato definitivamente nel 1996.
È facile ipotizzare che in futuro l'aumento di device dell'Internet delle cose richiederà l'adozione del protocollo IPv6. Occorre però considerare anche il consumo di energia elettrica necessaria per questi dispositivi. I device devono consumare poca corrente elettrica ed essere molto affidabili, mantenendo la possibilità di collegarsi a Internet. Per ottenere questi risultati, dovranno funzionare in una rete mesh, come previsto da 6LoWPAN, un gruppo di lavoro IETF (acronimo di Internet Engineering Task Force) che propone soluzioni "IPv6 over Low power Wireless Personal Area Networks", ovvero IPv6 su reti personali wireless a bassa potenza, sfruttando tecnologie particolari, per esempio la IEEE 802.15.4.

La seguente tabella mostra le differenze fra le principali tecnologie di comunicazione:

Protocolli IoT	**Standard**	**Frequenza**	**Range**	**Data Rates**
Bluetooth	Bluetooth 4.2	2.4GHz (ISM)	50-150m (Smart/BLE)	1Mbps (Smart/BLE)
ZigBee	ZigBee 3.0 based on IEEE802.15.4	2.4GHz	10-100m	250kbps
Z-Wave	Z-Wave Alliance ZAD12837 / ITU-T G.9959	900MHz (ISM)	30m	9.6/40/100kbit/s
6LoWPAN	RFC6282	(adapted and used over a variety of other networking media including Bluetooth Smart (2.4GHz) or ZigBee or low-power RF (sub-1GHz)	N/A	N/A
Thread	Thread, based on IEEE802.15.4 and 6LowPAN	2.4GHz (ISM)	N/A	N/A
WiFi	Based on 802.11n (most common usage in homes today)	2.4GHz and 5GHz bands	Approximately 50m	600 Mbps maximum, but 150-200Mbps is more typical, depending on channel frequency used and number of antennas (latest 802.11-ac standard should offer 500Mbps to 1Gbps)
Cellular	GSM/GPRS/EDGE (2G), UMTS/HSPA (3G), LTE (4G)	900/1800/1900/2100MHz	35km max for GSM; 200km max for HSPA	(typical download): 35-170kps (GPRS), 120-384kbps (EDGE), 384Kbps-2Mbps (UMTS), 600kbps-10Mbps (HSPA), 3-10Mbps (LTE)
NFC	ISO/IEC 18000-3	13.56MHz (ISM)	10cm	100–420kbps
Sigfox	Sigfox	900MHz	30-50km (rural environments), 3-10km (urban environments)	10-1000bps
Neul	Neul	900MHz (ISM), 458MHz (UK), 470-790MHz (White Space)	10km	Few bps up to 100kbps
LoRaWAN	LoRaWAN	Various	2-5km (urban environment), 15km (suburban environment)	0.3-50 kbps

Indirizzi MAC

Oltre all'indirizzo IP, ogni device collegato in rete ha anche un indirizzo MAC che viene impiegato per distinguere i dispositivi presenti in una rete fisica, allo scopo di consentire lo scambio reciproco di pacchetti.
L'acronimo MAC vuol dire *Media Access Control* (controllo per l'accesso multimediale). Un indirizzo MAC è definito da un numero a 48 bit composto da sei gruppi di cifre esadecimali separati da due punti, come nell'esempio che segue:

F8:32:E4:47:D0:9C.

Gli indirizzi MAC sono univoci, ma nonostante questo non sono in genere sfruttati al di fuori della rete Ethernet, per esempio oltre il router di casa. L'instradamento di un messaggio IP implica il trasferimento da nodo a nodo fintanto che non raggiunge il nodo che sa dove si trova il dispositivo fisico cui corrisponde il device associato all'indirizzo MAC che deve ricevere il messaggio.
La maggior parte dei dispositivi, per esempio i portatili, nasce con l'indirizzo MAC inciso nei chip della scheda Ethernet, anche se alcuni chip particolari non hanno codificato l'indirizzo MAC, per esempio i chip WizNet (produttore coreano che produce chip di rete per device incorporati) della scheda Ethernet di Arduino.
Si potrebbe memorizzare l'indirizzo nel firmware del chip, ma in questo caso ogni chip andrebbe costruito con un codice personalizzato da compilare nel firmware. In alternativa, è possibile fornire un semplice chip dati che memorizza solo l'indirizzo MAC da far leggere al chip WizNet. La maggior parte dei device consuner adotta una procedura simile a questa per garantire che il dispositivo venga sempre avviato con il medesimo indirizzo MAC.

I protocolli a livello Applicazione

Il modello ISO/OSI è uno standard *de iure* per le reti di calcolatori stabilito nel 1978 dall'ISO, il principale ente di standardizzazione internazionale, esso stabilisce uno stack di protocolli di comunicazione di rete suddivisa in 7 livelli:

- Livello Applicazione
- Livello Presentazione
- Livello Sessione
- Livello Trasporto
- Livello Rete
- Livello Collegamento
- Livello Fisico

A livello implementativo lo standard *de facto* affermatosi per architetture di rete a livelli è il TCP/IP, in funzione dei due importanti protocolli in essi definiti: il *Transmission Control Protocol* (TCP) e l'*Internet Protocol* (IP):
A differenza del modello ISO/OSI sono presenti quattro livelli invece di sette:

- Livello Applicazione
- Livello Trasporto
- Livello Rete
- Livello Accesso alla Rete

Consideriamo il livello più alto dello stack, ossia il livello Applicazione. È il livello con il quale avremo a che fare per le interazioni del nostro progetto dell'Internet delle cose.
A livello Applicazione il protocollo HTTP (HyperText Transfer Protocol) è importante quando si ha a che fare con l'Internet delle cose. HTTP è in sostanza un protocollo semplice. Il client richiede una risorsa inviando un comando a un URL (Uniform Resource Locator), completo di alcuni header.
Tramite l'HTTP il client ed il server negoziano la richiesta di una risorsa informativa. Tipicamente questo genere di informazioni sono organizzate in pagine, realizzate utilizzando dei marcatori speciali (chiamati tag) che ne definiscono la struttura e la formattazione. L'insieme dei marcatori costituisce un linguaggio meglio conosciuto come HTML (HyperText Markup Language) adoperato per la creazione di ipertesti anche distribuiti.
Per avere connessioni più sicure esiste il protocollo HTTPs che è la combinazione tra il contenuto HTTP e il protocollo SSL (Secure Socket Layer). Un server HTTPs è in ascolto generalmente su una porta differente (in genere la porta 443) e in fase di connessione imposta un collegamento sicuro e cifrato con il client.

Il protocollo HTTP non è idoneo per le applicazioni dell'Internet delle cose che invece utilizza:

- Protocollo MQTT
- Protocollo XMPP
- Protocollo CoAP

Protocollo MQTT

MQTT (*Message Queuing Telemetry Transport*) è un protocollo di messagistica piuttosto leggero (*http://mqtt.org*), progettato esplicitamente per situazioni nelle quali si ha una larghezza di banda limitata. Le specifiche vennero sviluppate inizialmente da IBM, ma sono state poi pubblicate come standard aperto e ora sono disponibili molte implementazioni che comprendono librerie scritte in numerosi linguaggi di programmazione.
Al posto del modello client/server di HTTP, il protocollo MQTT adotta un meccanismo di pubblicazione e sottoscrizione per scambiare messaggi tramite un apposito *message broker*. Invece di inviare messaggi a un determinato set di destinatari, i mittenti pubblicano i messaggi su un certo argomento (*topic*) sul message broker. Ogni destinatario si iscrive agli argomenti che lo interessano e, ogni volta che un nuovo messaggio viene pubblicato su quel determinato argomento, il message broker lo distribuisce a tutti i destinatari interessati. In questo modo è molto più semplice configurare una messagistica uno-a-molti.
Esiste inoltre un protocollo affine a questo, MQTT-S (*MQTT for Sensors*), per le piattaforme eccessivamente limitate o per le reti nelle quali non è disponibile il protocollo TCP; le sue specifiche permettono al protocollo MQTT di arrivare alle reti di sensori come ZigBee.

Protocollo XMPP

Un'altra soluzione per la messagistica è costituita dal protocollo XMPP (*Extensible Messaging and Presence Protocol*). Le sue specifiche (*http://xmpp.org*) derivano dal sistema di messagistica istantanea Jabber (http://www.jabber.it), e per questo motivo XMPP ha avuto un supporto talmente ampio da diventare un protocollo generale su Internet, infatti il protocollo è molto noto ed è ampiamente distribuito, ma non è stato progettato esplicitamente per le applicazioni incorporate, perciò utilizza il linguaggio XML per formattare i messaggi. La scelta di XML però rende i messaggi molto lunghi, il che preclude la possibilità di impiegare XMPP nei µP che hanno poca RAM.

Protocollo CoAP

Il protocollo CoAP (*Constrained Application Protocol*) è stato progettato per risolvere lo stesso tipo di problemi dell'HTTP ma, analogamente a MQTT-S, si rivolge alle reti che non dispongono del protocollo TCP. Si sta progettando di eseguire CoAP in UDP (a differenza del TCP, l'UDP è un protocollo di tipo connectionless), con i messaggi SMS dei telefoni cellulari e di integrarlo con 6LoWPAN.
Le specifiche CoAP traggono molte caratteristiche di design dal protocollo HTTP e includono un meccanismo definito per i proxy che ammette la mappatura da un protocollo all'altro. Le specifiche del protocolo CoAP sono nella RFC 7252 (https://tools.ietf.org/pdf/rfc7252.pdf).

USB

Se il device può fare affidamento sulla presenza di un computer potente, il collegamento tra i due dispositivi via USB è la soluzione più facile per fornire alimentazione elettrica e connessione in rete. In commercio esistono versioni di µP che includono il supporto USB.
Invece di presentare il µP come un device, si può fare in modo che agisca come un "host" USB. La configurazione host consente di collegare elementi che in genere sono connessi a un computer, per esempio telefoni cellulari tramite Android ADK (ADK acronimo di Accessory Development Kit), unità esterne di storage oppure dongle Wi-Fi.
Alcuni dispositivi, tra cui i dongle Wi-Fi, dipendono da un software che deve essere installato nel sistema host, per esempio istruzione per la connessione in rete, perciò si adattano meglio a soluzioni che si basano su schede SoC, che hanno un funzionamento più simile a quello di un computer.

Consumo di energia

I processori più veloci consumano spesso più energia elettrica di quelli più lenti. I device portatili o che si affidano a fonti di energia non convenzionali (batterie, energia solare) e che dipendono dal luogo in cui sono installati, possono avere problemi in termini di consumo. Anche quando il device è collegato alla presa elettrica il consumo di energia non va trascurato, perchè un minore consumo può diventare un requisito fondamentale.
I processori offrono una modalità "sleep" a basso consumo di energia. In questo modo si utilizza un processore più veloce per eseguire velocemente determinate operazioni, dopodiché si ritorna in modalità sleep a bassa potenza. Un processore più veloce può così non essere uno svantaggio anche in un device incorporato a basso consumo di energia.
Le prestazioni e la vita della batteria sono elementi che procedono di pari passo: ciò che va bene per le une va bene anche per l'altra. I device alimentati a batteria o da celle solari e quelli che devono rispondere immediatamente dopo che l'utente ha premuto un pulsante dipendono fortemente dalle prestazioni e dal consumo di corrente.
Gran parte del consumo di energia deriva dal design dell'hardware. In particolare, è da preferire il device che può disattivare i moduli del sistema che non sono in uso oppure se il processore può passare alla modalità *sleep* a basso consumo di energia quando si conclude l'esecuzione del codice o si attende che accada qualcosa. È sempre importante ottimizzare il codice in considerazione del fatto che, prima si conclude l'esecuzione del codice principale, prima si può mettere l'hardware a riposo.
Uno dei modi più semplici per rendere efficiente il codice richiede di passare a un modello guidato da eventi. Il motivo è legato alla possibilità di portare il device in modalità sleep per un tempo maggiore e di rimetterlo in azione quando serve, invece di lasciarlo sempre in funzione per verificare se è successo qualcosa che richiede un lavoro effettivo di elaborazione dati. Tecniche come i protocolli simili a MQTT permettono di eludere situazioni di questo tipo.
Riguardo all'hardware, bisogna utilizzare funzioni del processore, per "svegliare" il processore e richiamare il codice di elaborazione dati soltanto quando si verificano determinate condizioni rilevate dai sensori.

Interfaccia con i sensori e altri circuiti

Oltre a dialogare in Internet, il device deve interagire con altre cose, ovvero con i sensori che raccolgono dati dall'ambiente circostante, oppure con motori, LED, schermi e altro ancora, per attuare le operazioni di output. Il circuito può essere collegato sfruttando un bus per periferiche (i più diffusi sono i bus SPI "*Serial Peripheral Interface*" e I2C "*Inter-Integrated Circuit*"), oppure attravesro moduli ADC (*Analog to Digital Converter*) o DAC (*Digital to Analog Converter*).
Si possono anche impiegare i pin GPIO (*General Purpose Input/Output*), che forniscono input o output digitali. I µP e le soluzioni SoC offrono svariate combinazioni di interfacce di ogni genere.

Piattaforme Embedded

Il mercarto offre svariate piattaforme di computing incorporato:

- Arduino
- Raspberry Pi
- Mbed
- Beaglebone Black
- Atmel – Xpro
- Sakura

Arduino

Arduino è il "manifesto" dell'Internet delle cose e del physical computing in generale. Il Progetto Arduino nasce a Ivrea nel 2005. Un gruppo di docenti dell'IDII (*Interaction Design Institute Ivrea*) voleva realizzare una scheda per i propri studenti di design allo scopo di realizzare progetti interattivi e proposero una scheda poco costosa (intorno ai 25 €), nella quale fu inclusa una connessione seriale per consentire una programmazione semplice. In combinazione con l'ambiente di sviluppo software Wiring (2003), la scheda ebbe un impatto che rivoluzionò il mondo del physical computing.
La decisione iniziale di rendere open source il codice e gli schemi elettronici permise che la scheda Arduino sopravvivesse alla scomparsa di IDII. Il gruppo Arduino si concentrò sulla semplicità di programmazione piuttosto che sulle prestazioni, e ciò ha reso Arduino la scheda preferita da quasi tutti coloro che si avvicinavano ai primi progetti di physical computing.
La scheda Arduino "standard" ha avuto molte repliche:

- **Arduino NG**
- **Arduino Diecimila**
- **Arduino Duemilanove**
- **Arduino Uno**

La scheda Uno ha un µP ATmega328 e una presa USB per il collegamento con un computer. L'unità di storage è di 32 KB e la RAM di 2 KB.
Uno mette a disposizione 14 pin GPIO e 6 pin ADC. La porta seriale ATmega è disponibile da tutti i pin IO e, con un chip aggiuntivo, tramite il connettore USB.
La scheda Arduino Mega 2560 include 256 KB di storage flash, 8 KB di RAM, tre porte seriali aggiuntive, 54 pin GPIO e 16 pin ADC.
Arduino Due ha un µP ARM a 32 bit ed è la prima scheda Arduino ad adottare questa architettura. Le sue specifiche sono simili alla scheda Mega, ma la RAM è di 96 KB.
Un singolo cavo USB permette non solo di alimentare la scheda, ma anche di caricare il programma e comunicare con la scheda, per esempio quando si vuole utilizzare il computer per memorizzare i dati rilevati dai sensori collegati ad Arduino.
In Arduino lo sviluppo software avviene con l'impiego dell'ambiente IDE che il team fornisce all'indirizzo *http://arduino.cc*. È un IDE ricco di funzioni ed è basato sul linguaggio Processing (http://processing.org). La maggior parte dei progetti Arduino include un solo file di codice, i controlli più utilizzati riguardano la verifica del codice in fase di compilazione oppure il caricamento del programma sulla scheda.
Arduino non esegue di default un sistema operativo vero e proprio ma solo il bootloader per semplificare la procedura di caricamento del programma specifico. L'accensione della scheda si limita a eseguire il codice fino a quando non si spegne la scheda (o fino a quando il programma va in crash).
In Arduino è possibile caricare un sistema operativo costituito da una distribuzione RTOS (*Real-Time Operating System*), per esempio FreeRTOS/DuinOS. Il vantaggio principale di questi sistemi operativi è rappresentato dal supporto built-in del multitasking, anche se in molti casi si ottengono risultati accettabili con un'apposita libreria, più semplice da utilizzare.
Si può anche compilare il codice senza l'uso dell'IDE, sfruttando per esempio il toolset *avr-gcc* (*http://www.nongnu.org/avr-libc*).
Il linguaggio di programmazione per Arduino è un dialetto leggermente modificato di C++, che deriva dalla piattaforma Wiring e include librerie per la lettura e la scrittura di dati rilevati dai pin I/O di Arduino e per gestire le condizioni di "interrupt" di base (tecnica che permette di gestire il multitasking a un livello molto basso). La variante di C++ è piuttosto tollerante riguardo l'ordine delle istruzioni; per esempio, consente di chiamare funzioni prima che siano definite.

Il codice deve comprendere due sole routine:

- *setup()*: routine da eseguire una sola volta, quando si accende la scheda. Permette di impostare le modalità dei pin I/O di input e di output oppure di organizzare una struttura dati da impiegare in tutto il programma.

- *loop()*: routine da ripetere ciclicamente fintanto che la scheda Arduino rimane accesa. In genere, le istruzioni verificano lo stato di input, effettuano alcuni calcoli sui dati rilevati e intraprendono le azioni di risposta in output.

C++ è un linguaggio compilato, perciò in fase di compilazione si rileva un certo numero di errori, dovuti per esempio a una sintassi non corretta oppure alla presenza di variabili non dichiarate. Dato che ciò si verifica nel computer, il compilatore è in grado di fornire informazioni dettagliate che permettono all'utente di capire il problema che ha generato l'errore.
Le cose cambiano non appena il programma è caricato su Arduino che di solito non è collegato a uno schermo, ciò rende difficile cercare eventuali errori nel programma. Si possono verificare errori anche se il codice è stato compilato senza problemi. La situazione peggiore si verifica quando un errore di programmazione continua a far funzionare il programma ma si ottengono risultati completamente errati.
Gli errori di programmazione in runtime sono difficili da rilevare perchè il compilatore *avr-gcc* non supporta la gestione delle eccezioni, a differenza del linguaggio C++, probabilmente a causa del "carico" di memoria relativamente alto della gestione delle eccezioni; la piattaforma Arduino non consente perciò di utilizzare le tipiche istruzioni *try... catch...*
In assenza di uno schermo, Arduino permette di scrivere informazioni via cavo USB utilizzando l'istruzione *Serial.write()*.
Arduino mette a disposizione molti pin GPIO e comprende in genere una serie di morsetti (o *header*), parti in plastica che si sovrappongono ai fori dei pin per avere una connessione comoda e senza saldature, per i fili di collegamento e in particolare per una connessione "a ponte". I morsetti sono ideali per i prototipi e permettono di modificare facilmente la configurazione della scheda Arduino.
Sono presenti gli output di alimentazione a 5 V oppure a 3,3 V.
Oltre alle schede standard, ve ne sono altre dedicate ad alcune applicazioni; per esempio, la scheda Arduino Ethernet ha installato un chip Ethernet e trasforma la presa USB in una presa Ethernet, semplificando la connessione a Internet. Questa è un'ottima candidata a far parte dei progetti dell'Internet delle cose.
Nel mondo Arduino alcune schede prendono il nome di *shield* (scudi), alcune schede shield offrono funzionalità di rete, per esempio Ethernet, Wi-Fi o ZigBee; altre si collegano agli schermi LCD dei telefoni cellulari, altre ancora riproducono file MP3 o file WAV memorizzati su una SD card. Un sito web è dedicato al confronto e alla documentazione delle schede shield (*http://shieldlist.org*).
In termini di funzionalità, una scheda Arduino standard associata a una shield Ethernet è equivalente a una scheda Arduino Ethernet, anche se questa è più sottile, ma perde la comoda connessione USB. Si può comunque utilizzare un apposito adattore per trasformare la porta seriale in una presa USB, utile per caricare il programma o per effettuare trasmissioni di altro tipo.
Il progetto Arduino è di tipo open hardware, l'unica parte protetta del progetto è il marchio Arduino. La scheda wireless Arduino Fio, che all'inizio era stata sviluppata da terze parti (si chiamava Funnel IO) è stata poi adottata come scheda ufficiale approvata da Arduino.

Raspberry Pi

A differenza di Arduino, Raspberry Pi non nasce per il physical computing ma per il mondo dell'istruzione. L'idea di Eben Upton, amministratore e cofondatore di Raspberry Pi Foundation, consisteva nella realizzazione di un computer piccolo, poco costoso e progettato per essere programmato e per fare esperimenti.
Upton sviluppò la scheda SoC BCM2835, che aveva una unità GPU (*Graphics Processing Unit*) molto potente, capace di gestire video ad alta definizione e la riproduzione veloce di grafica. Includeva anche una CPU ARM a 700 MHz, a bassa potenza, poco costosa, inserita nel device come se si trattasse di un elemento aggiuntivo (http://www.gamesindustry.biz/articles/digitalfoundry-inside-raspberry-pi).
I nomi dei modelli di Raspberry Pi sono "Model A" e "Model B" più potente, il prezzo è quasi uguale a quello di Arduino.

La tabella che segue confronta le caratteristiche di Arduino Due, con la scheda Raspberry Pi Model B.

	Arduino Due	**Raspberry Pi Model B**
Velocità della CPU	84 MHz	700 MHz ARM11
GPU	No	Broadcom Dual-Core VideoCore IV Media Co-Processor
RAM	96 KB	512 MB
Storage	512 KB	SD card (4 GB +)
Sistema Operativo	Bootloader	Diverse distribuzioni Linux; sono disponibili altri sistemi operativi
Connessioni	54 pin GPIO	8 pin GPIO
	12 output PWN	1 output PWN
	4 UART	1 UART
	SPI bus	SPI bus con due chip select
	I^2C bus	I^2C bus
	USB 16U2 + native host	2 socket USB
	12 input analogici (ADC)	Ethernet
	2 output analogici (DAC)	HDMI out
		Componente video e audio out

Raspberry Pi è a tutti gli effetti un computer che esegue un sistema operativo reale e moderno, è in grado di comunicare tramite tastiera e mouse, dialoga su Internet e pilota un TV/monitor ad alta risoluzione grafica; Pi Model B ha incorporato una scheda Ethernet (analogamente a Arduino Ethernet, mentre non è presente in Due), può anche impiegare dongle USB Wi-Fi invece di utilizzare una scheda "shield" aggiuntiva.
Raspberry Pi ha una certa importanza nelle soluzioni multimediali, come media centre che non richiede un prototipo con nuovi componenti elettronici.
Un contenitore di Raspberry Pi lo si può trovare all'indirizzo https://shop.pimoroni.com/products/pibow-raspberry-pi-case .
Un altro kit interessante è Gertboard (https://www.raspberrypi.org/blog/tag/gertboard/).
Sono molti i sistemi operativi in grado di lavorare con le schede Pi.

- Raspbian: è una distribuzione basata su Debian, è la distribuzione di default ed è un'ottima scelta per lavorare con le schede Pi.
- Occidentalis: è la distribuzione Raspbian personalizzata per Adafruit (azienda hardware open-source fondata nel 2005). A differenza di Raspbian, questa distribuzione presume che venga impiegata in modalità "headless", ovvero senza essere collegata a tastiera e monitor, perciò la configurazione predefinita implica una connessione da remoto. Raspbian invece richiede una fase preventiva di configurazione.

Per l'Internet delle cose è preferibile una distribuzione simile ad Adafruit. Le configurazioni principali sono:

- il daemon sshd (relativo al protocollo SSH "Secure Shell" che permette di stabilire una sessione remota cifrata tramite interfaccia a riga di comando): è attivato di default;
- i registri del device: utilizzano la tecnica zeroconf (zero-configuration networking) con il nome *raspberrypi.local*, perciò non occorre conoscere o ricavare l'indirizzo IP di rete per effettuare una connessione.

Una volta scelta la distribuzione, occorre scompattarla nella scheda SD card. Dopo aver acceso la scheda Pi, si può stabilire una comunicazione come si fa di solito con un computer.
Il comando che da riga di comando Linux permette di accedere a Pi come se si trattasse del login a un server remoto è:
$ ssh root@raspberrypi.local
In Windows si utilizza un client SSH, per esempio PuTTY. Una volta collegato il device, si può sviluppare un'applicazione software.
Per quanto riguarda il linguaggio e l'ambiente di programmazione, le linee guida della fondazione suggeriscono l'impiego di Python. In effetti, il nome "Pi" deriva proprio da Python.

Di seguito è riportato l'"Hello World" del physical computing, ovvero l'esempio delle "luci lampeggianti":

```
import RPi.GPIO as GPIO
from time import sleep

GPIO.setmode(GPIO.BOARD)   # imposta lo schema di numerazione
                           # uguale a quello della scheda
GPIO.setup(8,  GPIO.OUT)   # imposta il pin GPIO 8 in output mode

led = False

GPIO.output(8,   led)      # inizializza il LED spento

while 1:
  GPIO.output(8,   led)
  led = not led            # cambia lo stato acceso/spento del LED
                           # per l'iterazione successiva
sleep(10)                  # attende un secondo
```

L'esempio mostra istruzioni simili a quella del codice C++ per Arduino. L'unica differenza riguarda i dettagli della modularizzazione: il codice GPIO e perfino la funzione *sleep()* devono essere specificati.
L'impiego di un linguaggio "ad alto livello" come Python, rende più semplici alcune operazioni comuni, per esempio:

- Gestire stringhe di dati carattere.
- Evitare completamente di dover gestire la memoria e i bug correlati.
- Effettuare chiamate a servizi Internet ed esaminare i dati ricevuti.
- Collegarsi a database ed effettuare elaborazioni dati complesse.
- Astrarre pattern comuni o comportamenti complessi.

La presenza di librerie immediatamente disponibili su PyPi (https://pypi.python.org/pypi) può consentire un semplice riutilizzo di codice che altre persone hanno scritto, utilizzato e testato per intero.
Si può fare un confronto tra Python e C++, dal momento che C++ è il linguaggio utilizzato per la programmazione di Arduino.

- Python, come la maggior parte dei linguaggi di alto livello, compila un codice piuttosto grande (in termini di uso della memoria) e lento rispetto a C++. La prima non rappresenta in genere un problema, dato che Pi ha memoria più che a sufficienza. La velocità di esecuzione può invece diventarlo: è probabile che Pi sia "abbastanza veloce" per tutto ciò che riguarda le comunicazioni su Internet, il rallentamento principale è dovuto alla trasmissione in rete. Python può diventare troppo lento nel caso in cui l'elettronica dei sensori e degli attuatori richieda risposte nell'ordine delle frazioni di secondo.
- Python gestisce la memoria in automatico, questo si traduce nella riduzione dei bug e in un'esecuzione più corretta del programma. L'elaborazione automatica va comunque pianificata e richiede tempo, quindi si possono verificare pause di funzionamento, che dipendono dalla strategia di garbage collection che influenzano la temporizzazione di eventi successivi. Si possono verificare casi in cui Python trattiene più memoria di quella necessaria. Nel peggiore dei casi, la memoria non è rilasciata fino al termine del processo: si ha il cosiddetto *memory leak* (perdita di memoria). Dato che un device dell'Internet delle cose è lasciato solo per lunghi periodi di tempo, le perdite di memoria si accumulano e finiscono per esaurire la memoria del device e lo mandano in crash.
- Linux presenta dei problemi nell'uso "in tempo reale". Dato che si tratta di un sistema operativo abbastanza grande, si possono avere molti processi in esecuzione simultanea, e la temporizzazione può dipendere dalle priorità assegnate alla CPU dal runtime Python in un dato istante.
- Arduino esegue solo un set di istruzioni, in un loop rigido, fino a quando non viene spento o va in crash. Pi esegue costantemente un certo numero di processi. Se uno dei processi mostra dei malfunzionamenti o se due processi hanno un conflitto di risorse, si possono avere inconvenienti che non sono legati al codice.

Comunque sia, chi conosce Linux troverà lo sviluppo di una scheda Pi relativamente semplice, anche se non si avvicina a quello dell'IDE Arduino. Arduino avvia il codice nell'istante in cui si alimenta la scheda. Lo stesso

comportamento richiede in Linux un certo numero di meccanismi, per esempio la presenza di uno script di inizializzazione in */etc/init.d*.
A differenza di Arduino che ha funzioni di debugging limitate, la maggior parte delle quali riguarda l'output di dati attraverso la porta seriale o l'utilizzo di effetti secondari quali le luci lampeggianti, il codice Python in Linux offre il vantaggio tipico di un linguaggio e di un sistema operativo. È possibile esaminare il codice tramite il debugger integrato in Python, collegarlo al processo tramite il comando Linux *strace*, visualizzare i registri, scoprire quanta memoria si sta impegnando e altro ancora.
Pi è un computer general-purpose ed è privo delle rigide limitazioni di memoria di Arduino, si può adottare una logica *try...cath...* per intercettare gli errori del codice Python e stabilire cosa fare quando si verificano.
La scheda Raspberry Pi ha 8 pin GPIO, a differenza di quelli di Arduino, i pin di Raspberry Pi non sono identificati da singole etichette, ciò ha senso a causa del gran numero di componenti di Pi.
Il blocco di pin include output a 5 V oppure a 3,3 V, mentre i pin GPIO ammettono solo tensioni a 3,3 V. La scheda Pi non è protetta dalle sovratensioni, perciò si rischia di rovinare il circuito se lo si alimenta a 5 V.
Raspberry Pi non ha ingressi analogici (ADC); ciò significa che non sono molte le opportunità di collegare sensori elettronici, pronti da usare, agli ingressi digitali della scheda. La lettura di fotocellule, sensori di temperatura, potenziometri e così via richiede il collegamento di un convertitore ADC via bus SPI.
Dal momento che il chip Broadcom è difficile da reperire rispetto ai chip più diffusi di Atmel per Arduino, renderà più difficile il passaggio dal prototipo al prodotto industriale.

BeagleBone Black

BeagleBone Black è il device più recente tra quelli realizzati dal gruppo BeagleBone. Il team è composto da dipendenti Texas Instruments e nonostante i prodotti non siano schede TI, utilizzano molti componenti TI.
La scheda BeagleBone originale non include output audio o video, ci sono i pin ADC per l'input analogico che mancano in Raspberry Pi.
L'influenza di Pi si nota nella versione più recente della piattaforma BeagleBone, ovvero BeagleBone Black. Non sono presenti i connettori per video e audio analogici, ma è stato aggiunto un connettore microHDMI per fornire output digitali audio e video. Il prezzo del connettore microHDMI è di circa 40 € (in aggiunta ai 70 € di BeagleBone) e adotta funzioni di interfaccia migliori di quella della scheda BeagleBone originale.
La tabella che segue mette a confronto le specifiche di Raspberry Pi Model B con la scheda BeagleBone Black.

	BeagleBone Black	**Raspberry Pi Model B**
Velocità della CPU	1 GHz ARM Cortex-A8	700 MHz ARM11
GPU	SGX530 3D	Broadcom Dual-Core VideoCore IV Media Co-Processor
RAM	512 MB	512 MB
Storage	Card MMC incorporata da 2 GB, più µSD card	SD card (4 GB +)
Sistema Operativo	Diverse distribuzioni Linux, Android e altri sistemi operativi	Diverse distribuzioni Linux; sono disponibili altri sistemi operativi
Connessioni	65pin GPIO	8 pin GPIO
	8 output PWN	1 output PWN
	4 UART	1 UART
	SPI bus	SPI bus con due chip select
	I^2C bus	I^2C bus
	USB client + host	2 socket USB
	Ethernet	Ethernet
	Micro-HDMI out	HDMI out
	CAN bus con 7 input analogici (ADC)	Componente video e audio out

Per l'impiego nell'Internet delle cose è importante che le schede dispongano di una connessione Ethernet e che possano sfruttare i poco costosi dongle USB Wi-Fi, se necessario.
Le schede di espansione di BeagleBone prendono il nome di *cape* (mantello) invece di chiamarsi *shield* che è il termine usato per Arduino. Le schede cape aggiungono controllori per display LCD, e per svariate applicazioni di rete e di misura con sensori.
Nonostante BeagleBone sia una soluzione valida per il computing general-purpose, il suo mercato di riferimento è decisamente ristretto; forse per questo motivo, il gruppo responsabile del suo sviluppo ha proposto una soluzione intermedia: ogni scheda ha installato la distribuzione Linux Angstrom.

BeagleBone esegue di default la configurazione di rete zeroconf. Il collegamento può avvenire in molti modi.
Un utile riferimento si trova all'indirizzo https://beagleboard.org/static/beaglebone/latest/Docs/Hardware/BONE_SRM.pdf .
Per scrivere il codice per il device viene utilizzato l'IDE Cloud9 che è un ambiente di programmazione online. Una versione hosted di Cloud9 è disponibile all'indirizzo https://c9.io/. L'ambiente online Cloud9 supporta anche Ruby e Python, mentre la versione gratuita supporta solo Node.js, un framework scritto in JavaScript. Node.js è una piattaforma basata sul runtime JavaScript di Chrome per realizzare applicazioni di rete veloci e scalabili. Node.js adotta un modello I/O guidato da eventi, perfetto per applicazioni che devono gestire in tempo reale molti dati rilevati da device distribuiti.
Un altro compromesso da valutare quando si studia la programmazione in tempo reale è il modello multitasking. Se una o più attività devono essere effettuate in tempo reale, per esempio il pilotaggio di un motore, la rilevazione della pressione su un pulsante, l'accensione e lo spegnimento di più luci in base a diversi schemi, allora l'ambiente di runtime deve adottare una strategia che permetta di stabilire quando passare da un'attività a un'altra. Node.js utilizza un modello cooperativo, per cui si esegue ogni azione in base al coordinamento operato da un processo centrale. Dato che in realtà si può eseguire solo un'azione alla volta, il coordinamento richiede di attendere il completamento dell'azione precedente prima di avviare quella successiva; ciò significa che non è possibile garantire che una certa azione verrà intrapresa entro un certo numero di millisecondi.
Di seguito è riportato l'"Hello World" del physical computing, ovvero l'esempio delle "luci lampeggianti" quando si utilizza l'IDE Cloud9:

```
require('bonescript');

setup = function () {

pinMode(bone.USR3, OUTPUT);        // Attiva il LED USR3
}

loop = function () {
  digitalWrite(bone.USR3, HIGH);    // Accende il LED USR3
  delay(100);                       // Aspetta 100 ms
  digitalWrite(bone.USR3, LOW);     // Spegne il LED USR3
  delay(100);                       // Aspetta 100 ms
}
```

Il codice è praticamente sempre lo stesso, dal punto di vista concettuale, anche in questo caso si hanno due funzioni, *setup* e *loop*; si deve impostare la modalità output di un certo pin; si alterna la configurazione dei valori *HIGH* e *LOW* in output a intervalli regolari.
Analogamente a Python, anche JavaScript è un linguaggio di alto livello, perciò non appena superata la configurazione di input e output dei componenti elettronici, il dialogo con i servizi internet diventano istruzioni più facili da impostare rispetto a quelle relativamente a basso livello del C++ adottato da Arduino.
Node.js è un ambiente ricco di librerie che consentono di integrare un'applicazione, include anche strumenti di test automatici come *node-unit*. Dal momento che Node.js si basa sul multitasking collaborativo tramite i callback, il percorso di esecuzione può diventare complesso: i test consentono di controllare che si verifichino tutti i percorsi attesi, nell'ordine corretto.
Tutto l'hardware di BeagleBoard.org è open source, si può scaricare il materiale di progetto che include gli schemi elettronici, l'elenco componenti e il layout dei circuiti stampati all'indirizzo https://beagleboard.org/.

Telefoni cellulari e tablet

I telefoni moderni e i tablet, a prescindere che utilizzino iOS (iPhone/iPad), Android, Blackberry o perfino Windows, sono corredati di sensori, giroscopi e termometri, una o più fotocamere in grado di registrare immagini fisse e video, microfoni, GPS, Wi-Fi, Bluetooth, USB, pulsanti e uno schermo multitouch. I telefoni hanno ovviamente una connettività always-on a Internet, via Wi-Fi o tramite la rete telefonica; così si comportano anche molti tablet. Questi device hanno anche una serie di output: uscite audio ad alta fedeltà, video di qualità HD e uno o più elementi che vibrano. Hanno anche una potenza di elaborazione simile a quella di Raspberry Pi (più veloci per l'elaborazione dati e più lenti per l'elaborazione grafica, a causa della GPU di Pi, molto potente).
Anche se un device mobile ha molte caratteristiche interessanti, non è indicato per l'Internet delle cose per svariati motivi: prima di tutto questi device sono piuttosto grandi e costosi e poi perchè non è possibile collegare il telefono a un circuito, a sensori o attuatori.

L'ADK (*Android Development Kit*) cerca di risolvere il problema suddetto associando il device Android con un microcontrollore compatibile come la scheda Arduino Due basata su chip ARM. Dati il costo del device Arduino e di Arduino Due, l'ADK non sembra avere le caratteristiche di una piattaforma funzionale.

Servizi web

I componenti fondamentali dell'Internet delle cose sono:

Oggetto fisico
+
Controllore, sensori e attuatori
+
Internet
=
Internet delle cose

L'espressione "Internet delle cose" suggerisce che Internet deve essere parte fondamentale dell'equazione. Internet aggiunge la dimensione delle comunicazioni. La rete consente al device di informare noi e altri in merito alla presenza di eventi oppure di raccogliere dati e consentirci di agire in tempo reale, e permette di unire informazioni che provengono da località diverse e da diversi tipi di sensori.
Esistono servizi già operativi, per esempio Xively (https://xively.com/) è un servizio web gratuito che permette di creare uno spazio in rete per la trasmissione di dati di un sensore collegato ad Arduino, ma se ne può creare uno personalizzato attraverso un'API. Un'API definisce la modalità di accesso a un servizio destinato a macchine, e non a persone.
Un'API definisce i messaggi inviati da client a server e da server a client. In sostanza, si possono inviare dati nel formato che si preferisce, ma è quasi sempre meglio adottare uno standard esistente, in quanto si trovano comode librerie per il client e per il server. Di seguito sono riportati gli standard più diffusi:

- REST (*Representational State Transfer*): accede a un set di URL web utilizzando metodi HTTP, per esempio comandi *GET* e *POST*, ma anche *PUT* e *DELETE*. Il risultato è spesso in formato XML[2] oppure JSON[3], ma il più delle volte dipende dai meccanismi di negoziazione HTTP content-type.

- JSON-RPC[4] accede a un singolo URL passando una stringa JSON del tipo *{'method':'update','params': [{'timer-id':1234, 'description':'Writing API'}], 'id':12}*. Anche il valore restituito è in formato JSON, per esempio *{'result':'OK', 'error':null, 'id':12}*.

- XML-RPC: standard simile a JSON-RPC che utilizza il formato XML al posto di JSON.

---

[2] Il linguaggio XML (*Extensible Markup Language*) utilizza una coppia < > per distinguere i suoi elementi e tende a essere più verboso dello standard JSON, quindi è meno adatto ai sistemi vincolati alle risorse come i device dell'Internet delle cose. Le specifiche XML sono dettate dal W3C (*World Wide Web Consortium*) *http://www.w3.org/standards/xml*, un consorzio che si occupa anche di HTML, CSS e di altri standard web.

[3] Lo standard *JavaScript Object Notation* (JSON) *http://json.org*, pronunciato "Jason", definisce un modo per formattare i dati che si possono così scambiare con facilità tra sistemi diversi. Il nome dello standard suggerisce la sua derivazione dal linguaggio JavaScript, anche se può lavorare con altri linguaggi, per esempio Ruby e Python.
Alla base dello standard ci sono alcune proprietà, espresse nella forma

"nome proprietà": "valore proprietà"

I valori delle proprietà possono essere una stringa, un numero, un valore booleano, oppure un altro oggetto JSON o un array. Le singole proprietà si distinguono perché separate da una virgola, e si possono raggruppare proprietà differenti in un oggetto utilizzando una coppia di parentesi graffe. Gli array sono raggruppati tramite una coppia di parentesi quadre.

[4] La sigla RPC (*Remote Procedure Call*) identifica una modalità di chiamata di un codice di programmazione che non si trova sullo stesso computer dove è presente il codice che si sta scrivendo.

- SOAP (*Simple Object Access Protocol*): standard che utilizza il formato XML per la trasmissione, analogamente a XML-RPC, ma che fornisce una serie aggiuntiva di funzionalità, da sfruttare nei sistemi molto complessi.

La configurazione di una richiesta HTTP implica una serie di trasmissioni di andata e ritorno con il server. Il protocollo TCP prevede uno scambio handshake in tre fasi composto da una richiesta SYN da parte del client, una trasmissione SYN-ACK dal server che riconosce (*acknowledge*) la richiesta e infine l'invio del comando ACK dal client.
Il protocollo HTTP impiegato dai servizi web si colloca al di sopra del protocollo TCP. In genere, l'API che colloquia direttamente con il livello TCP è nota come *sockets API*. Il nome deriva dal fatto che la comunità web studiò queste funzionalità a livello HTTP e lanciò la soluzione chiamandola *WebSockets*.
WebSockets ha il vantaggio di essere bidirezionale, è ideale per i timer delle attività. Dopo aver configurato un socket, il timer deve inviare le informazioni per le attività che sono state avviate, modificate o cancellate, oltre a leggere le informazioni che riguardano modifiche via software.

Node.js

Node.js, disponibile all'indirizzo https://nodejs.org, è un framework basato sul runtime JavaScript V8 di Chrome per realizzare applicazioni di rete veloci e scalabili. Node.js adotta un modello I/O guidato da eventi, privo di blocchi, che lo rende un framework leggero ed efficiente, perfetto per applicazioni che devono gestire in tempo reale molti dati rilevati da device distribuiti.
Grazie al progetto Node.js JavaScript, un linguaggio prettamente lato client, è stato utilizzato come linguaggio per server web. Node.js permette di creare un server web gestito e sviluppato in JavaScript, esegue codice asincrono ed è più veloce rispetto ai classici server che eseguono codice PHP o un altro linguaggio lato server.
Per mantenere aggiornate le librerie collegate a un generico progetto Node.js utilizza come package manager npm (Node Package Manager), la più grande collezione di librerie open source al mondo in grado di installare in completa autonomia una serie di pacchetti software che aiutano lo sviluppo del progetto.

Node-RED

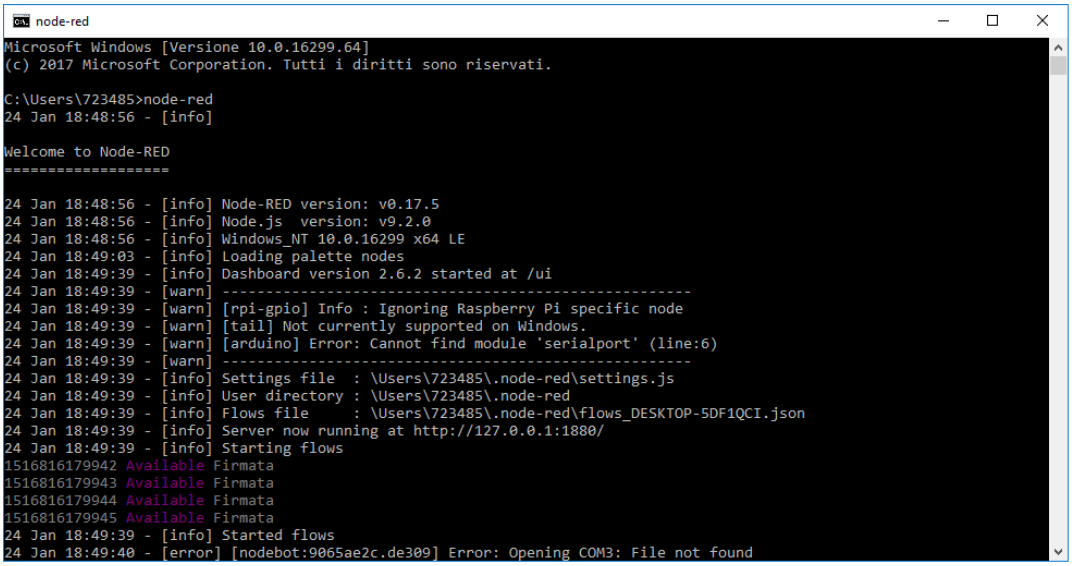
Node-RED è uno tra i più noti tool di flow-based programming per l'Internet of Things. Nasce con l'obiettivo di collegare tra loro diversi dispositivi (con eventuali relativi sensori ed attuatori), oltre a API e servizi online per poter realizzare sistemi altamente integrati e complessi in modo del tutto semplice ed intuitivo. Questo strumento può rendersi molto utile anche in fase di prototipazione di un progetto per individuare da subito funzionalità e potenzialità.
Node-RED nasce dall'idea di alcuni ricercatori della IBM, Nicholas O'Leary, Dave Conway-Jones e Andy Stanford-Clark, che nella prima versione si focalizzarono sulla possibilità di utilizzare questo strumento per poter trasferire messaggi attraverso il protocollo MQTT costruendo dei flussi in maniera del tutto visuale.
MQTT (*Message Queuing Telemetry Transport*) è un protocollo di messagistica piuttosto leggero (*http://mqtt.org*), progettato esplicitamente per situazioni nelle quali si ha una larghezza di banda limitata. Le specifiche vennero sviluppate inizialmente da IBM, ma sono state poi pubblicate come standard aperto e ora sono disponibili molte implementazioni che comprendono librerie scritte in numerosi linguaggi di programmazione.
Al posto del modello client/server di HTTP, il protocollo MQTT adotta un meccanismo di pubblicazione e sottoscrizione per scambiare messaggi tramite un apposito *message broker*. Invece di inviare messaggi a un determinato set di destinatari, i mittenti pubblicano i messaggi su un certo argomento (*topic*) sul message broker. Ogni destinatario si iscrive agli argomenti che lo interessano e, ogni volta che un nuovo messaggio viene pubblicato su quel determinato argomento, il message broker lo distribuisce a tutti i destinatari interessati. In questo modo è molto più semplice configurare una messagistica uno-a-molti.
L'interconnessione **publisher** (chi pubblica) e **subscriber** (chi riceve/sottoscrive) è gestita attraverso un server centrale (**broker**) che ha il compito di "smistare" i messaggi applicando appunto il pattern *publish/subscribe*.
Esiste inoltre un protocollo affine a questo, MQTT-S (*MQTT for Sensors*), per le piattaforme eccessivamente limitate o per le reti nelle quali non è disponibile il protocollo TCP; le sue specifiche permettono al protocollo MQTT di arrivare alle reti di sensori come ZigBee.

Architettura di Node-RED

Node-RED è completamente scritto in JavaScript e gira su Node.js, il modello event-driven di JavaScript e l'esecuzione asincrona delle operazioni di I/O, consentono di sviluppare soluzioni scalabili ed efficienti per l'analisi real-time di flussi di dati, con un forte orientamento all'Internet of Things.
L'ulteriore vantaggio di utilizzare Node.js è quello di poter utilizzare package già pronti e disponibili attraverso npm, da installare per aggiungere ulteriori funzionalità a Node-RED.
Le applicazioni Node-RED sono definite come "flow" (flusso) e il codice JavaScript non è presente solo nel runtime che garantisce l'esecuzione di uno o più flussi ma anche nei "blocchi" che li compongono e che sono chiamati "*nodi*".
Ciascun nodo ha una parte che potremmo definire di "*code-behind*" che contiene l'implementazione del processo e dell'elaborazione eseguita dal relativo nodo. L'altra "faccia" del nodo è descritta attraverso codice HTML per quanto riguarda la *parte visuale* e di configurazione; anche in questo caso il JavaScript viene usato come codice client-side per la user experience.
Le connessioni sono realizzate attraverso le "porte", di input e di output, di cui ciascun nodo è dotato; in Node-RED sono così evidenti tutti i concetti tipici della flow-based programming.
A questo punto risulta naturale pensare al browser come ambiente in cui "vive" Node-RED. Tuttavia va precisato che l'ambiente visuale non è indispensabile, anche se permette di realizzare flussi in modo semplice, immediato e intuitivo. Ciascun flusso infatti è descritto da un file in formato JSON, che possiamo anche scrivere con un editor di testo.
Per avviare il runtime di Node-RED basta digitare il comando:

node-red

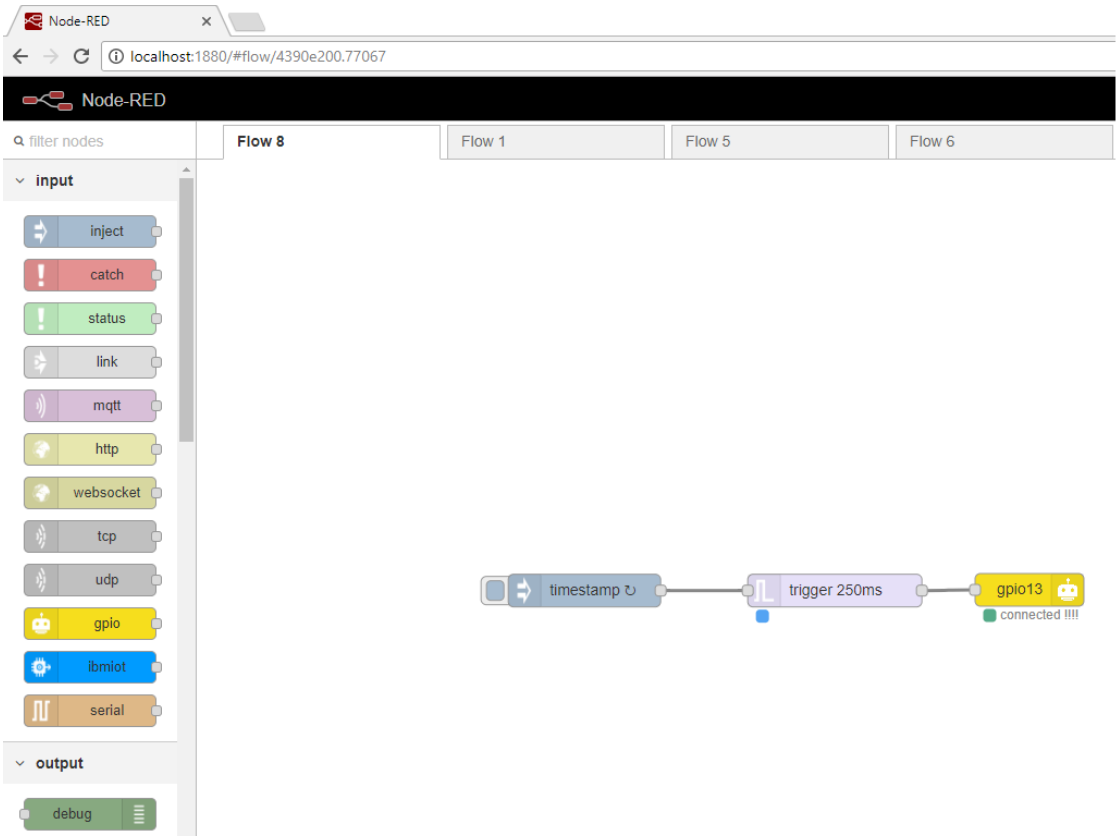
dal Prompt dei comandi.



Se il processo di avvio è avvenuto correttamente, possiamo lanciare il nostro browser e digitare l'indirizzo:

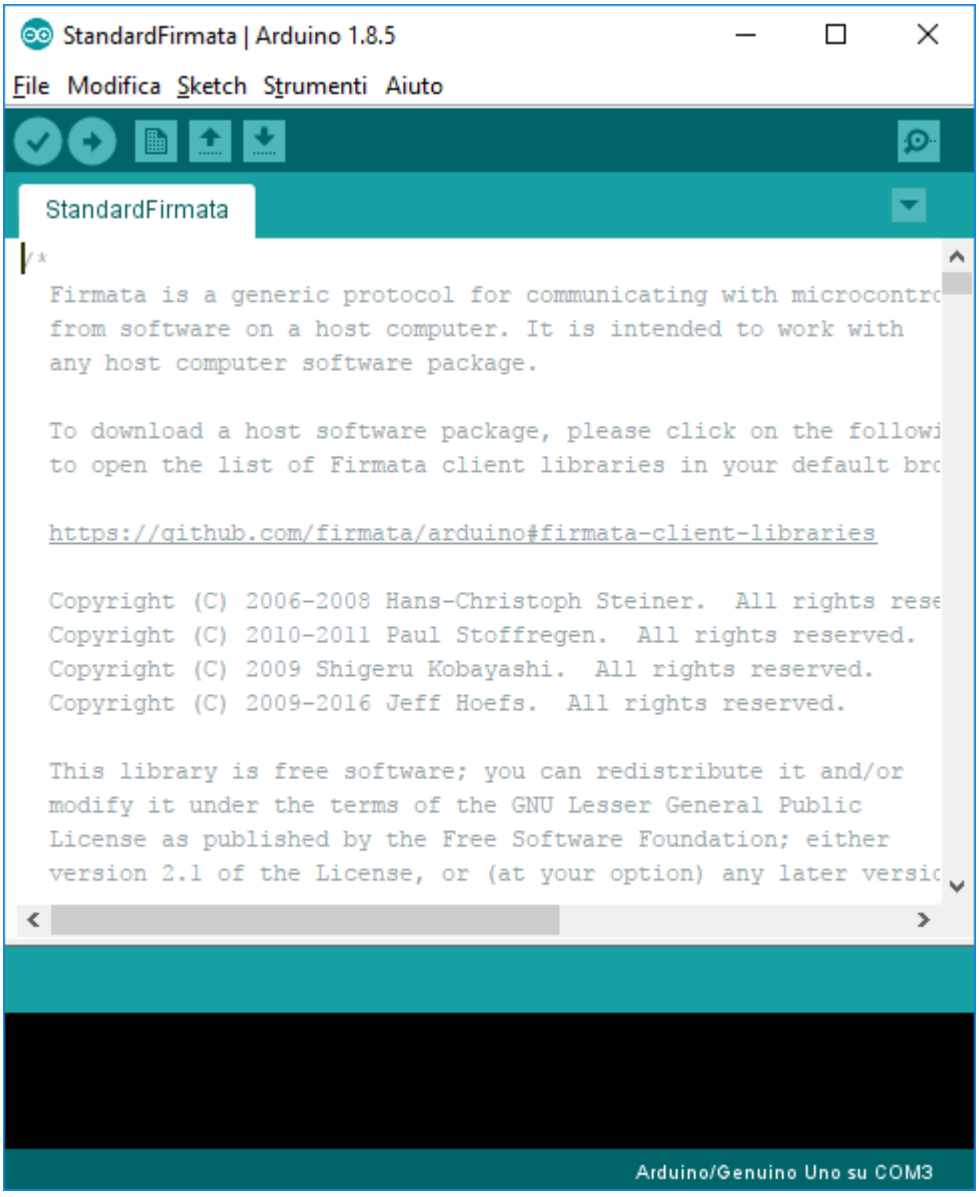
http://127.0.0.1:1880

A questo punto apparirà l'interfaccia utente di Node-RED.



Firmware Firmata

Il firmware Firmata fornisce l'accesso a tutte le funzionalità di Arduino tramite la porta seriale (cavo USB). Per installare Firmata basta collegare la scheda Arduino al computer tramite il cavo USB e aprire l'IDE di Arduino.
Nel menu File ➔ Esempi si trova un sottomenu denominato Firmata, selezionare il firmware Standard.



Per esigenze particolari si può andare direttamente sul sito Firmata Builder (pagina di configurazione, progetto GitHub) che permette di scegliere le funzionalità e la velocità della porta seriale.

Installazione del Modulo Serialport in Node.js

Arrestare Node-RED e posizionarsi nella directory node-red:

cd ~/.node-red/.

Installare il modulo serialport con il comando:

npm install serialport --unsafe-perm –build-from-source.

Non ho installato il modulo Arduino in Node-RED node-red-node-arduino perchè ho avuto difficoltà nella connessione a Firmata.

Installazione del Modulo Johnny-Five (Johnny 5)

Johnny-Five , un progetto open source, è una piattaforma JavaScript Robotics & IoT.
Rilasciato da Bocoup nel 2012, Johnny-Five è mantenuto da una comunità di appassionati sviluppatori e di ingegneri hardware.
All'interno di Node-RED Johnny-Five è disponibile come plugin, comunica con il firmware Firmata di Arduino sulla porta seriale.
Per l'installazione posizionarsi nella directory node-red:

cd ~/.node-red/.

Quindi installare i moduli **johnny-five** e **node-red-contrib-gpio johnny-cinque** con i comandi:

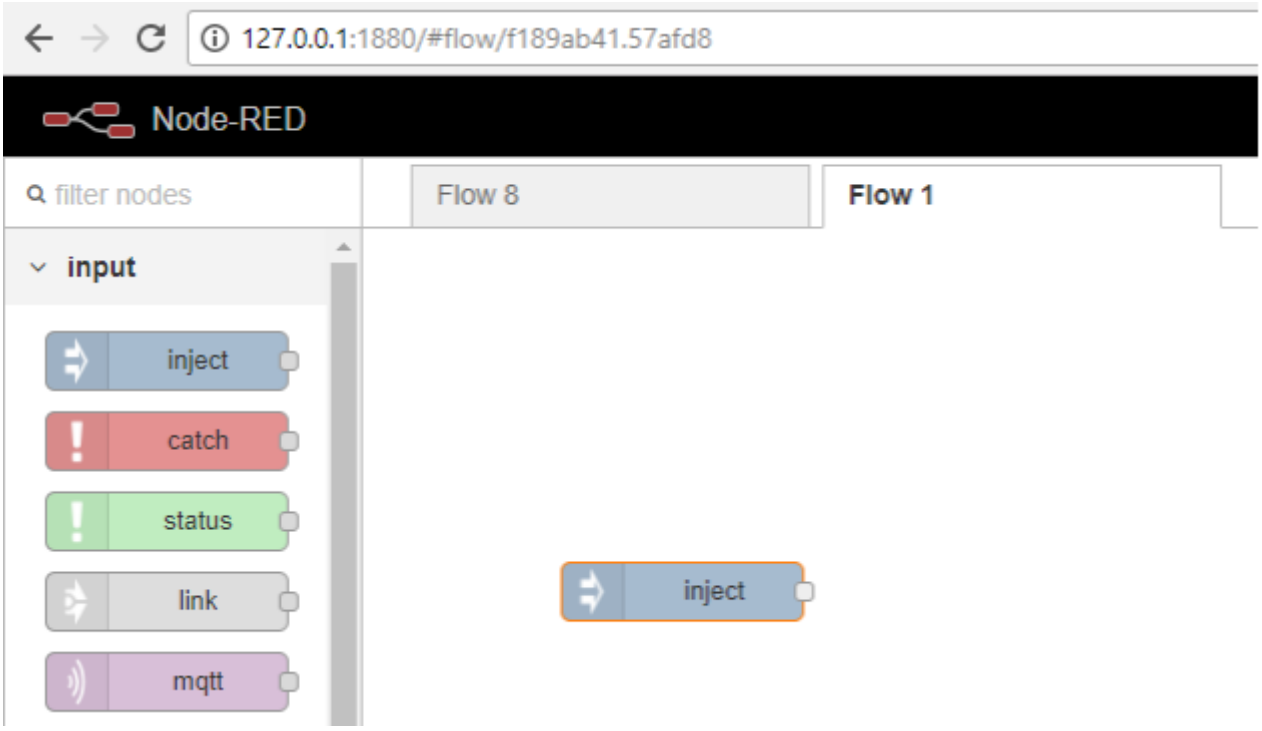
npm install -g johnny-five --unsafe-perm –force

npm install -g node-red-contrib-gpio --unsafe-perm –force

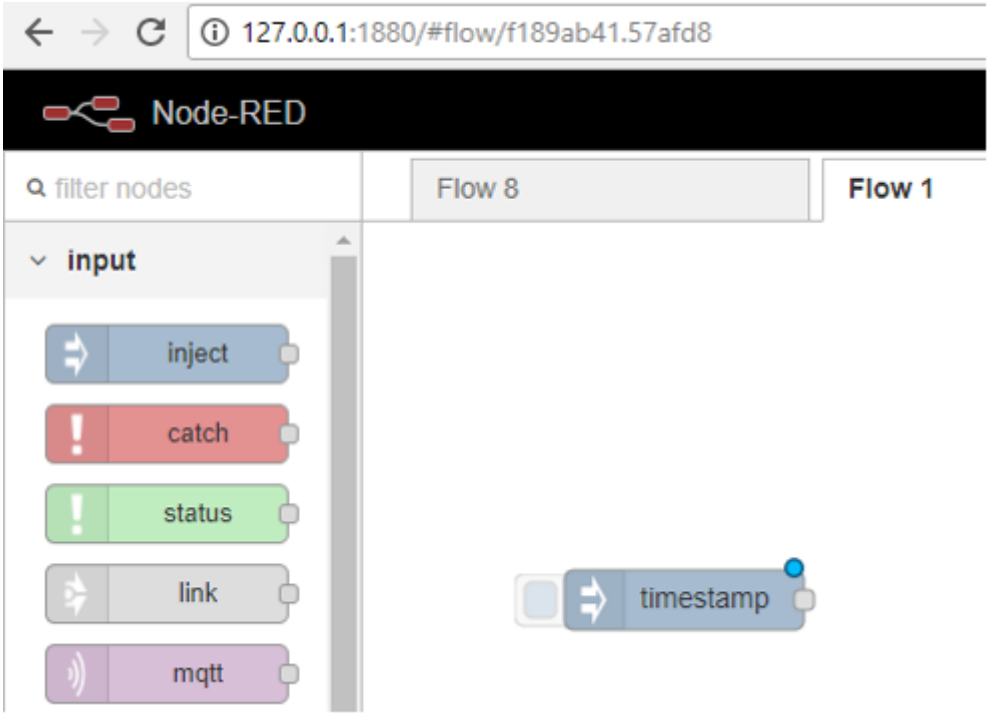
Una volta completata l'installazione, si dispone di 3 nuovi strumenti (gialli) nel toolbox:

Pilotare il GPIO di Arduino usando il nodo GPIO Out

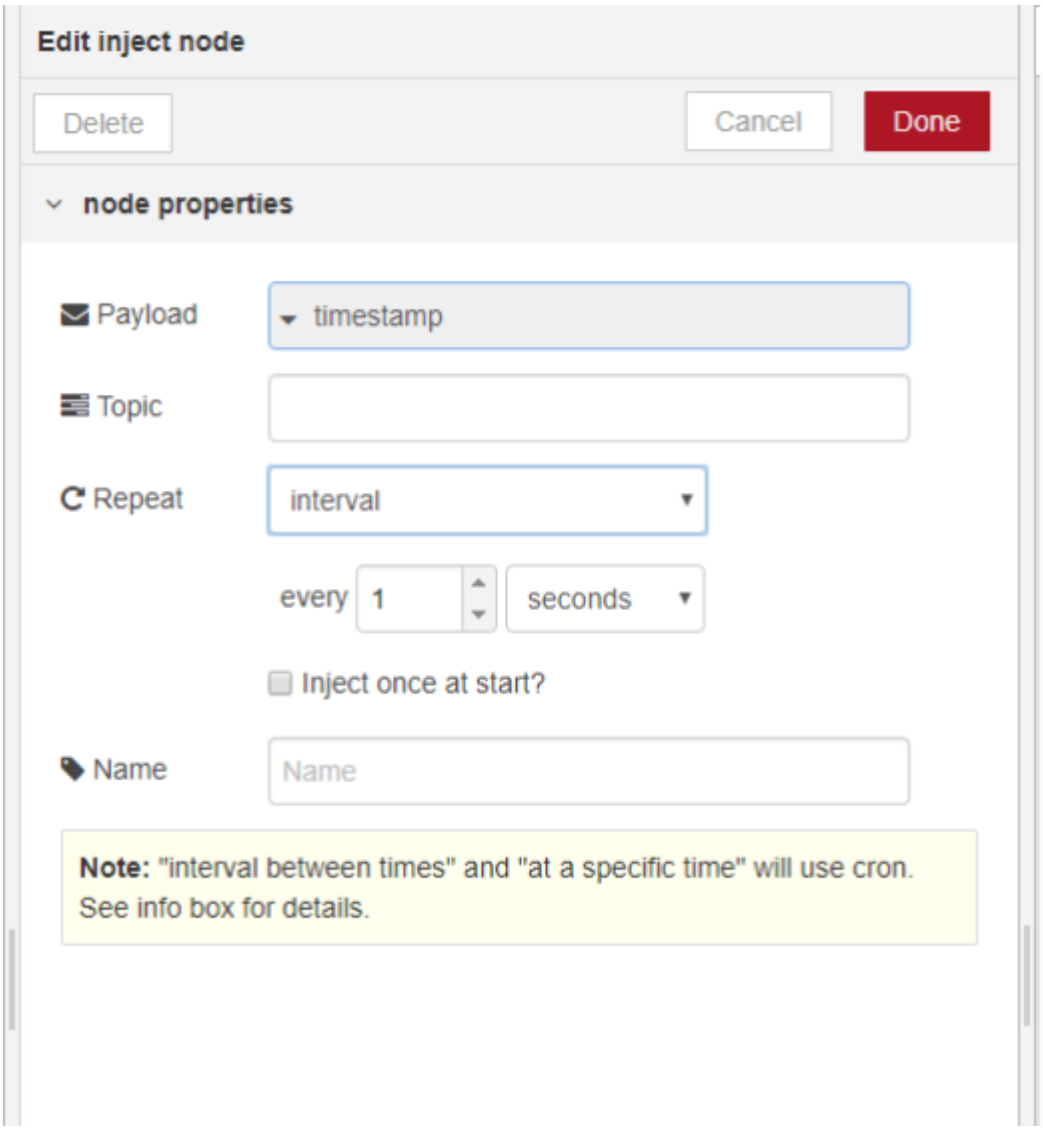
Il GPIO di Arduino può essere controllato attraverso i GPIO flows. Qui, per esempio, un LED collegato al pin 13 di Arduino lampeggerà per un breve periodo di tempo (1 secondo). Il primo passo è quello di inserire (trascinandolo con il mouse) un node inject all'interno del Flow.



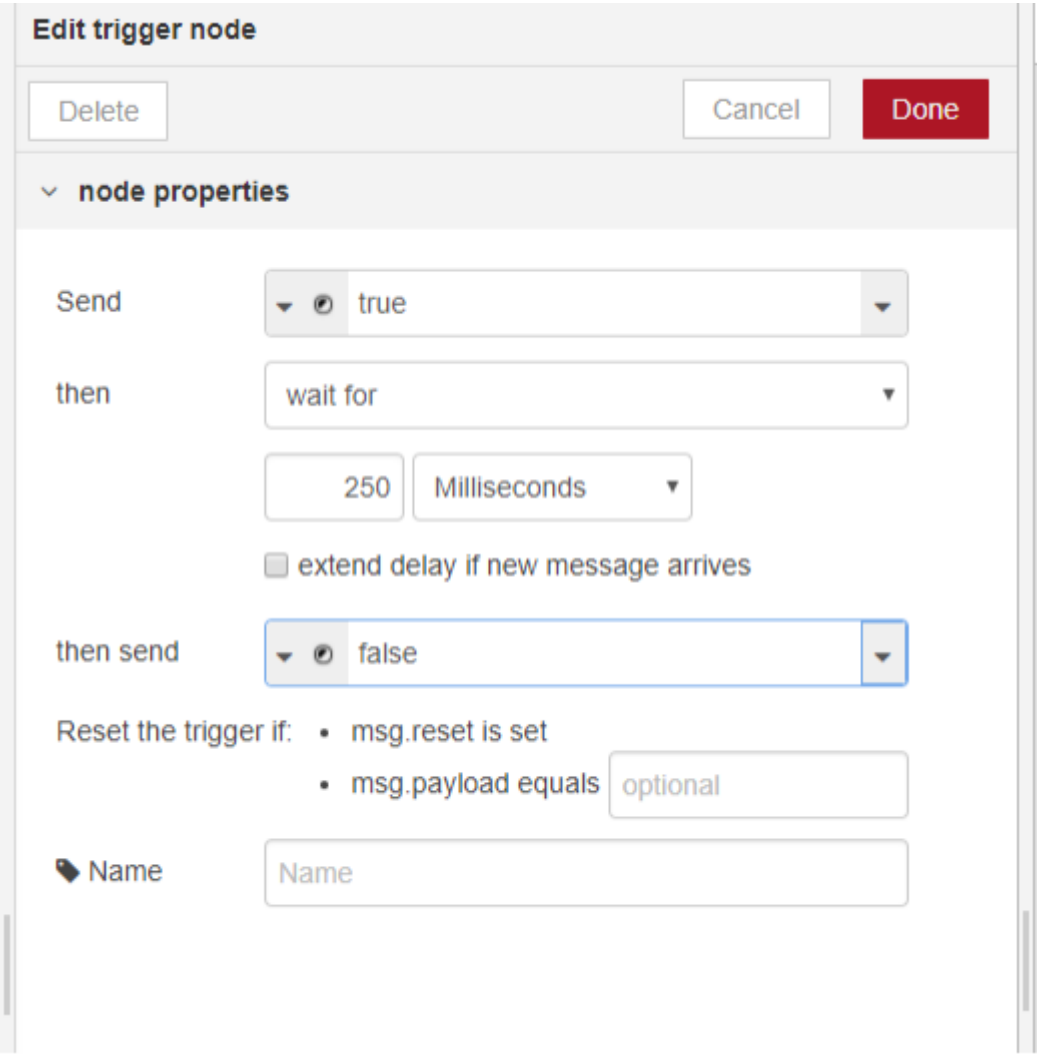
Una volta rilasciato, il nodo cambia nome in *timestamp*.



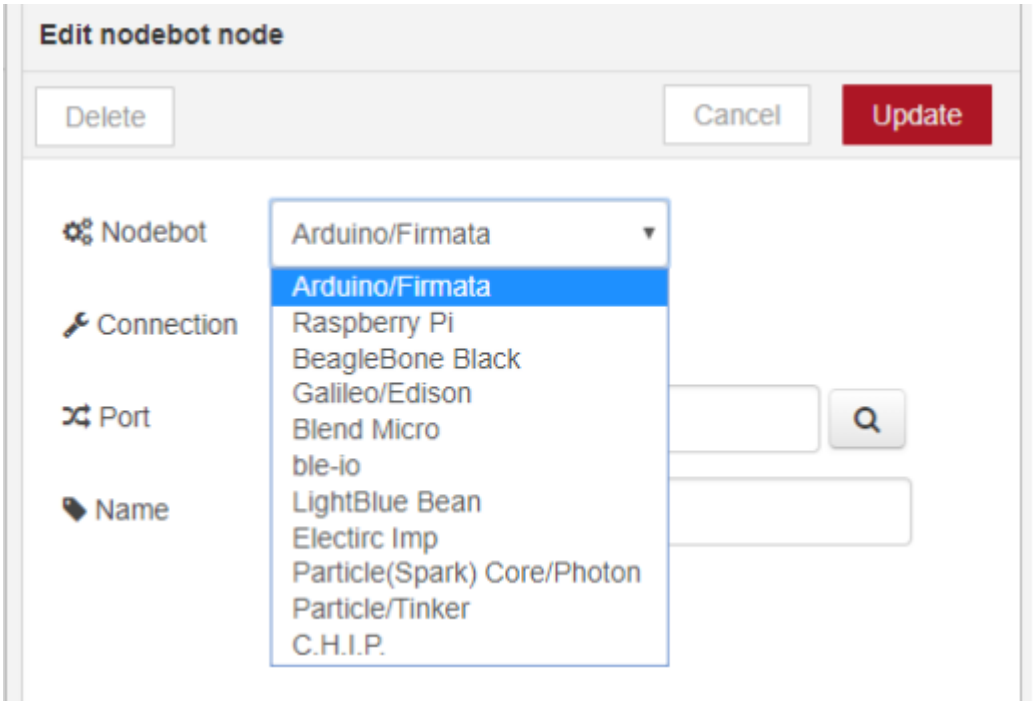
Per modificare le proprietà del nodo *timestamp* ed impostare l'intervallo di 1 secondo basta fare doppio click sul nodo, scegliere *interval* e alla fine cliccare su *Done*.

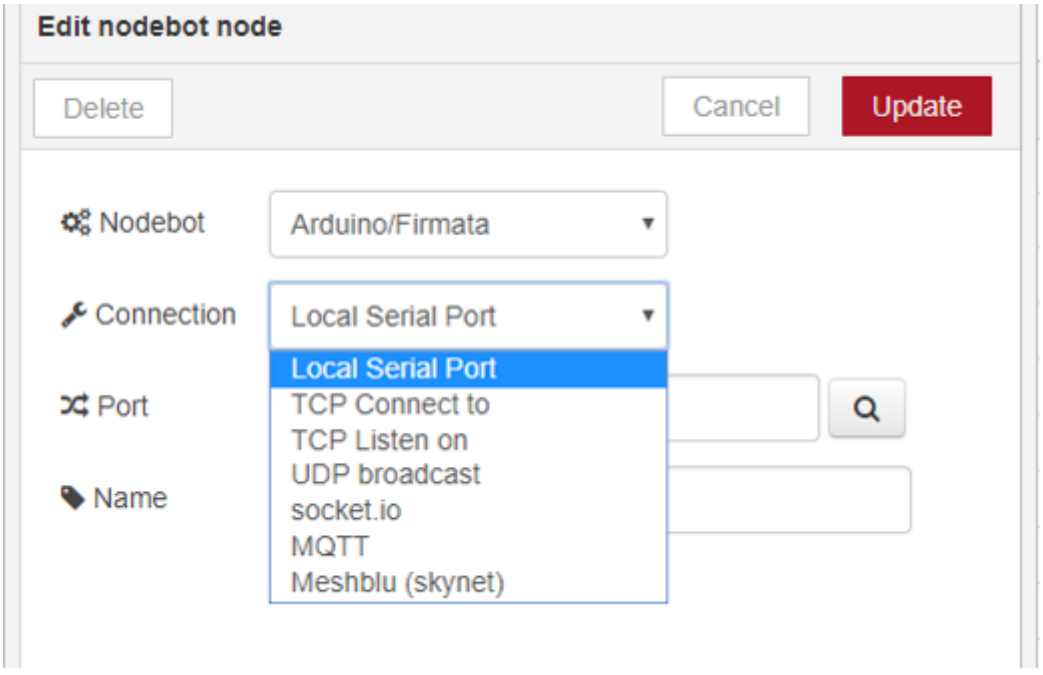


A questo punto occorre aggiungere un nodo *Trigger* che invierà un segnale (On/Off) per 250 ms, (si può modificare la durata). La logica di un nodo *Trigger* consiste nel generare due messaggi consecutivi in uscita, separati da un intervallo di tempo personalizzabile noto come timeout, alla ricezione di un qualsiasi messaggio in ingresso.

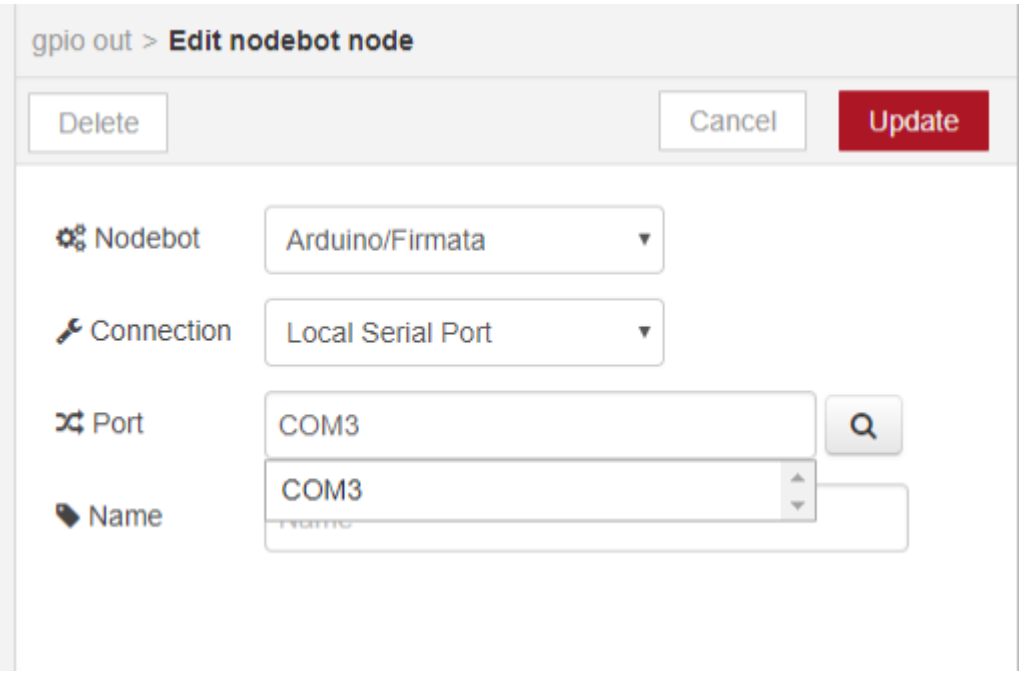


Alla fine aggiungere un nodo *GPIO Out* e scegliere Arduino/Firmata dall'elenco Nodebot e Local Serial Port dall'elenco Connection dal momento che Arduino è collegato tramite USB.

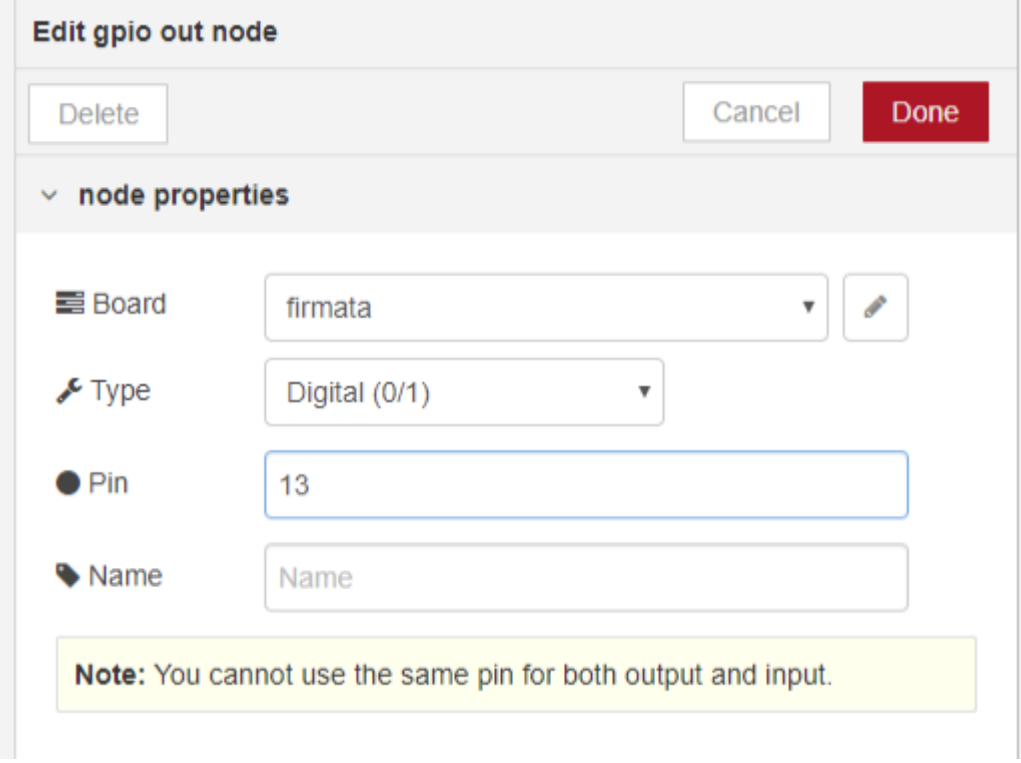




Per trovare la porta, fare clic sulla lente di ingrandimento. Dopo alcuni secondi, la porta a cui è collegato Arduino/Firmata sarà presente nell'elenco. In questo caso la porta è COM3.

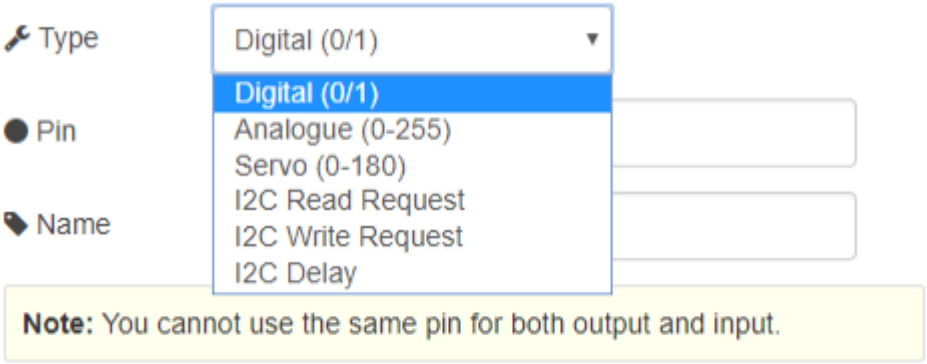


Nell'elenco Board, bisogna scegliere firmata che si è appena creato.



Nell'elenco Type scegliere **Digital (0/1)**. Sono disponibili anche:

- Analog (0-255);
- Servo (0-180);
- I2C (Inter Integrated Circuit) è un sistema di comunicazione seriale utilizzato tra circuiti integrati. Più avanti vedremo l'utilizzo del nodo johnny5 per recuperare le misurazioni da un sensore I2C.



Una volta terminato, è necessario effettuare il deploy del flusso utilizzando il tasto di "Deploy" che troviamo sul lato destro del tool visuale. Al termine di questa operazione, Node-RED salva il nostro flusso in un file JSON e ne avvia l'esecuzione. Se tutto è corretto, il led collegato al pin 13 inizierà a lampeggiare.