Progettazione e sviluppo di una piattaforma abilitante del **Web of Things** per **dispositivi embedded**

Relatore: Prof. Marco Di Felice

Correlatori: Dott. Lorenzo Gigli
Dott. Luca Sciullo

Candidato:
Daniele Rossi

Università di Bologna
Dipartimento di Informatica - Scienza e Ingegneria
Laurea Magistrale in Informatica

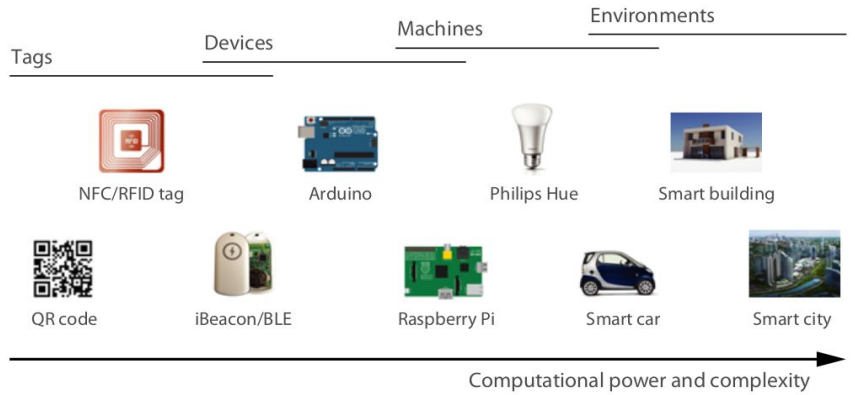
Stato dell’arte

Internet of Things (IoT)

Paradigma in cui la rete Internet viene integrata con oggetti fisici (**Thing**).

Da semplici oggetti muniti di codici QR, NFC/RFID tag a sistemi complessi come Smart Car, Smart Building o Smart City.

Difficile **interoperabilità** per utilizzo di protocolli, hardware e software differenti tra la miriade di prodotti esistenti in commercio.

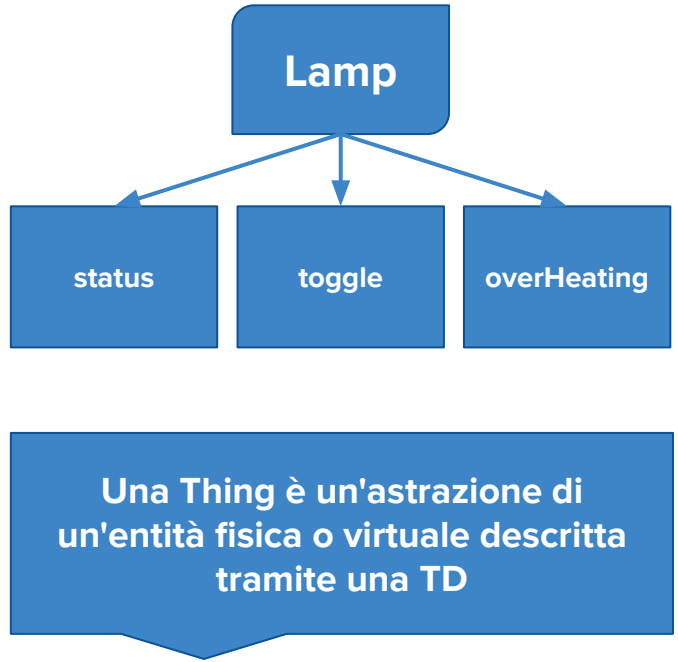


Tipologie di Thing
Dominique D. Guinard, Vlad M. Trifa. Building the Web of Things. Manning Publications, 2016

W3C Web of Things (WoT)

Il W3C ha istituito un Working Group per contrastare la frammentazione dell'IoT attraverso una serie di componenti standard (**building blocks**):

- **WoT Thing Description (TD)**
  L'entry point di una Thing: definisce un modello semantico di descrizione dei dati, dei modelli di interazione (Proprietà, Azioni ed Eventi), delle specifiche di sicurezza e dei pattern di comunicazione. La sua serializzazione standard è **JSON-LD**.
- **WoT Binding Templates**
  Collezioni di meta-dati per l'interoperabilità tra piattaforme IoT
- **WoT Scripting API**
  API per l'interazione T2T e la gestione della logica della Thing

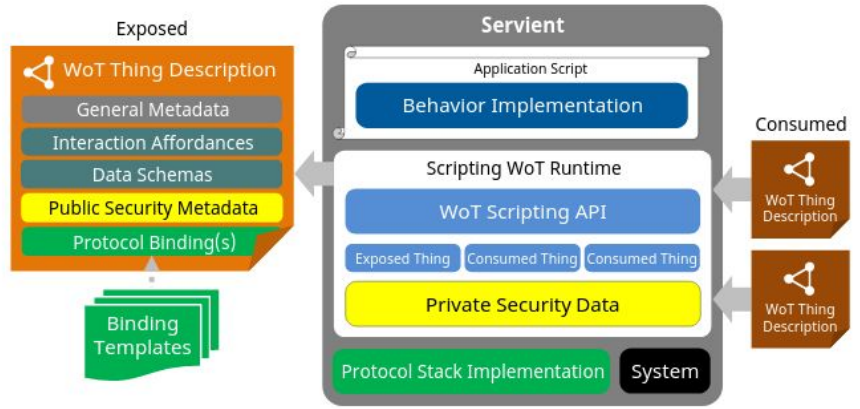


W3C Web of Things (WoT): Servient

Un Servient è uno **stack software** che implementa i componenti dell'architettura del W3C WoT.

Un Servient è in grado di **eseguire**, **esporre** e/o **consumare** le Thing, svolgendo così sia la funzione di server sia di client.

Un Servient offre un'implementazione concreta dell'architettura del WoT attraverso i suoi quattro moduli: **Behaviour Implementation**, **WoT Runtime**, **Protocol Stack Implementation** e **System API**.



Implementazione di un WoT Servient
https://www.w3.org/TR/wot-architecture/

Embedded WoT Servient

Obiettivi e funzionalità

Obiettivo primario: studio e realizzazione dello stack software **W3C WoT Servient** in grado di **esporre Thing** a partire da **sistemi embedded** dell'**Internet of Things**.

Alla base del progetto si è presa in esame l'implementazione del WoT Servient di Eclipse Foundation: **thingweb.node-wot**.

Funzionalità principali:

- **Creazione** ed **esposizione** di Thing Description
- **Creazione** di file di scripting (sketch)
- **Compilazione** e **flashing** di file di scripting

Architettura

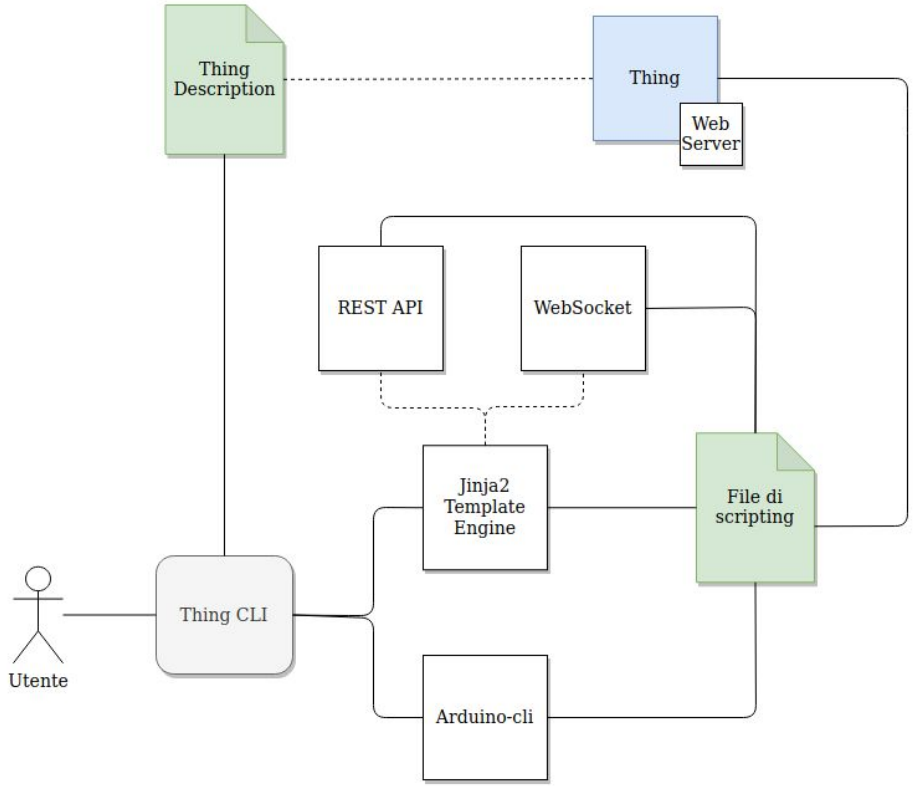
1. **Thing CLI**
   Interfaccia a riga di comando per definire TD, configurare ed installare file di scripting su sistemi embedded

2. **Server Web**
   Espone la TD e gestisce le richieste verso TD e modelli di interazione attraverso **API REST**

3. **Server WebSocket**
   Gestisce le notifiche push degli eventi

4. **Template Engine**
   Genera il codice del file di scripting

5. **Arduino-cli**
   Interfaccia a riga di comando per compilare e trasferire il codice eseguibile sul dispositivo



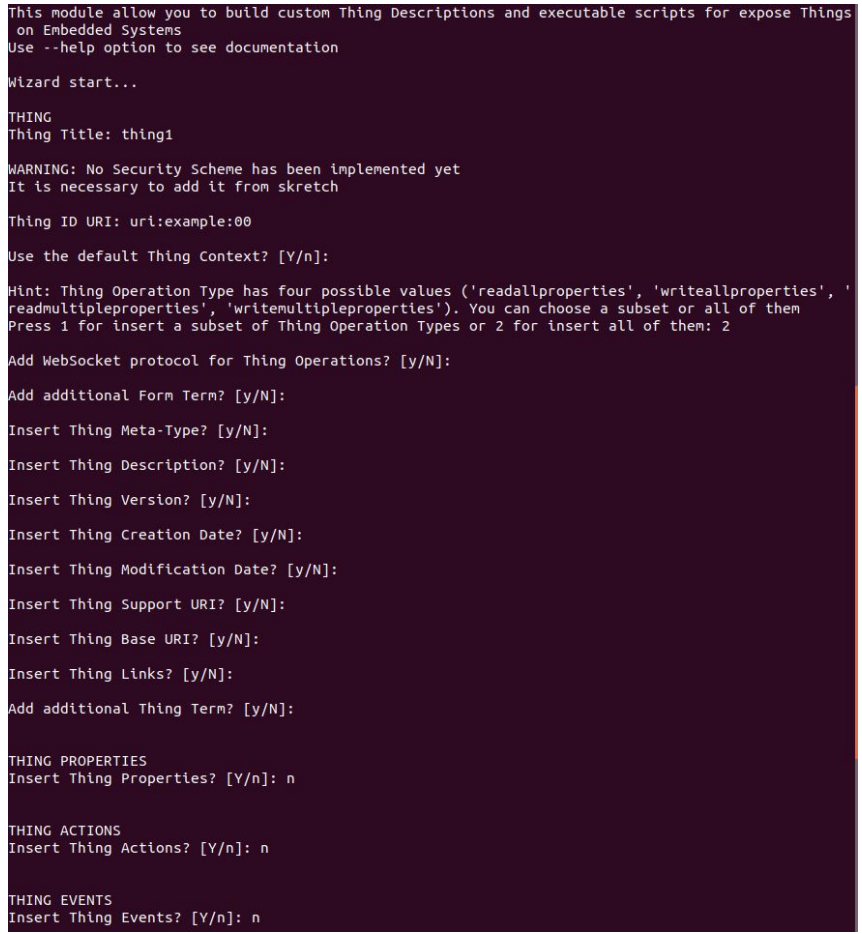
Thing CLI

È un applicazione a riga di comando per terminale scritta in **Python** (v3.6) tramite la libreria **Click** (v7).

Permette all'utente di definire la logica della Thing tramite una procedura guidata completamente interattiva.

Fornisce le seguenti funzionalità:

- **Definizione** TD e modelli di interazione
- **Definizione** campi per generazione di sketch
- **Configurazione** principali costrutti di programmazione
- **Compilazione** e **flashing** di sketch



Server Web

Il sistema embedded espone la Thing attraverso un Server Web (***home: ip/nome_thing/***) implementato tramite la libreria **ESP8266WebServer** per NodeMCU.

Espone un'**API REST** per la gestione delle richieste verso TD, proprietà e azioni.

In particolare:

- la TD viene esposta sulla root in formato **JSON-LD**
- ad ogni proprietà viene associato un endpoint accessibile tramite **richieste GET**
- ad ogni azione viene associato un endpoint accessibile tramite **richieste POST**

Tramite il metodo on() si impostano gli endpoint su cui il server rimane in attesa di richieste

```
server.on(req1, HTTP_GET, handleReq1);
server.on(req2, HTTP_GET, handleReq2);
server.on(req3, HTTP_GET, handleReq3);
server.on(req4, HTTP_POST, handleReq4);
server.on(req5, HTTP_GET, handleReq5);
```

Server WebSocket

Il sistema embedded espone un Server WebSocket per gestire la **comunicazione asincrona** tipica degli **eventi**. Il server è implementato tramite la libreria **WebSockets**.

Espone un endpoint per ogni evento, proprietà e azione definiti nella TD.

Gestisce tre tipi di messaggi per gli eventi:

1. **sottoscrizione**: il client si sottoscrive ad un evento per ricevere le notifiche push
2. **notifica**: messaggio inviato al client ogniqualvolta si verifica un evento a cui è sottoscritto
3. **cancellazione**: il client cancella la sua sottoscrizione ad un evento per non ricevere più notifiche push

Il server websocket processa tutti i messaggi (eventi) ricevuti tramite la funzione webSocketEvent

```
webSocket.onEvent(webSocketEvent);
```

Template Engine 

Il motore di template **Jinja2** (v2.10) genera lo sketch nel linguaggio di programmazione **Embedded-C** (Arduino IDE).

Interagisce con la **Thing CLI** per assegnare i valori alle variabili di cui è composto.

Fornisce le seguenti funzionalità:

- Inclusione di **librerie**
- Dichiarazione **variabili locali** e **globali**
- Dichiarazione **funzioni**
- Configurazione **server web** e **websocket**
- Configurazione **setup()** e **loop()**
- Gestione **proprietà**, **azioni** ed **eventi**

- **load() -> caricamento del file di template**
- **render() -> resa del template**

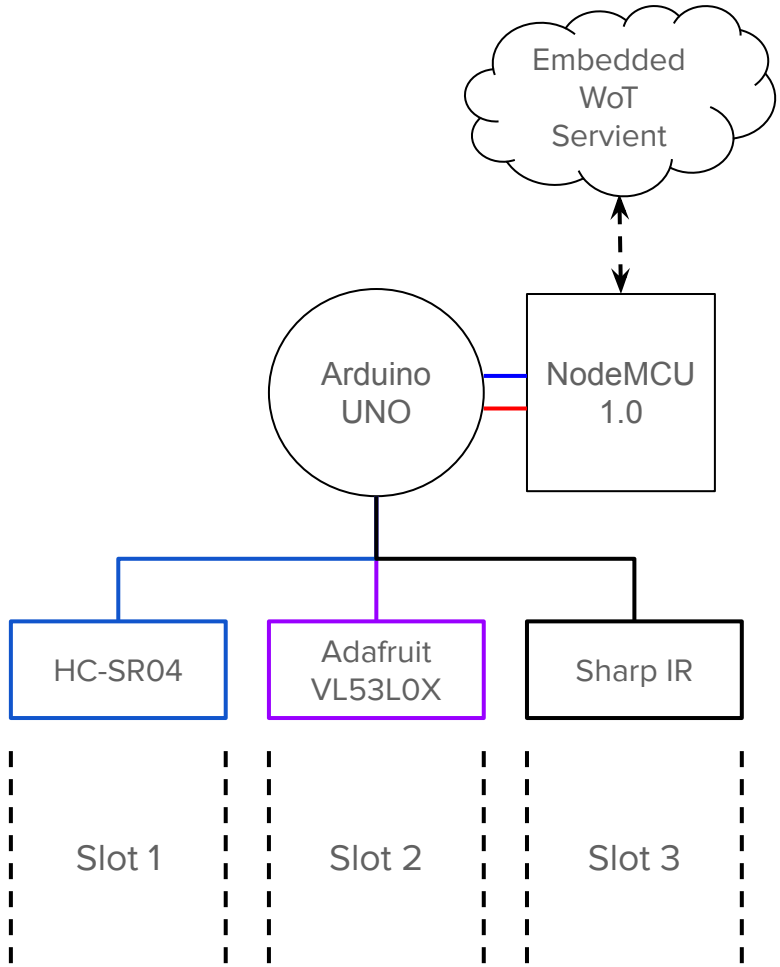
Validazione: Smart Parking

Applicazione dell'IoT al sistema di parcheggio integrando **sensori eterogenei** per il rilevamento degli slot.

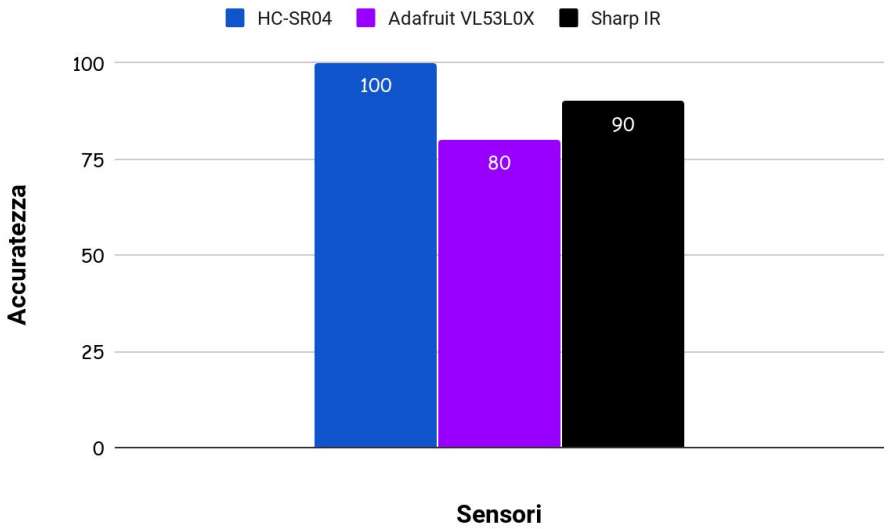
Tre sensori: **HC-SR04**, **Adafruit VL53L0X**, **Sharp IR**.

Implementazione concreta:

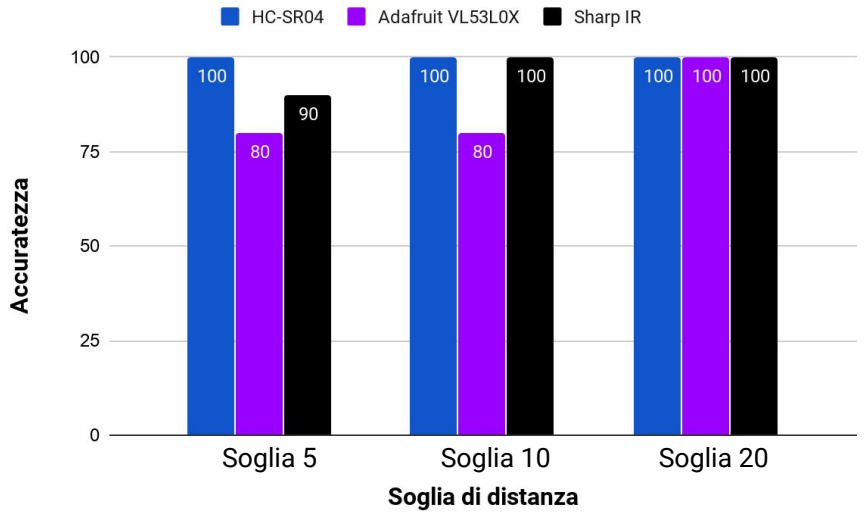
- Una proprietà per **slot disponibili**
- Tre proprietà per ogni **sensore**: stato parcheggio e valore di soglia di distanza
- Tre azioni per **soglia di distanza**



Validazione: Risultati



Differenze tra i sensori in termini di accuratezza dello stato del parcheggio con soglia pari a 6



Differenza tra i sensori in termini di accuratezza con soglia pari a 5, 10, e 20

Conclusioni e sviluppi futuri

Tentativo di portare l'articolata architettura **WoT Servient** nel mondo dei **sistemi embedded**.

Proposta di un sistema fruibile e intuitivo per l'utente in grado di:

- **Esporre** TD
- **Generare** file di scripting
- **Compilare** e **caricare** sketch su sistemi embedded

1. Passaggio da un'interfaccia a riga di comando ad un'**interfaccia grafica**
2. Mantenere il lavoro allineato con lo **standard W3C WoT**

Grazie per la cortese attenzione