

PROGETTO TOO(L)SMART

OUTPUT AZIONE 2 - O.2.d

Codice Output	O.2.d
Denominazione	Scheda componente tecnologica
Unità di Misura	Numero
Valore Target	1
Enti coinvolti	Ente Responsabile: UNIME Enti Partecipanti: tutti

Descrittivo:

L'output fa parte del pacchetto di strumenti del kit di riuso, volto a facilitare la diffusione della BP abilitando l'attivazione di un completo e autonomo trasferimento di soluzioni tra Amministrazioni e supportando le differenti fasi che compongono tali processi, e in particolare:

Fase "Ricerca e Selezione della buona pratica":

Elenco dei fattori tecnologici interni ed esterni che possono influenzare positivamente o negativamente il trasferimento e l'adozione della BP:

La **Piattaforma TOOLSMART** è una piattaforma in grado di raccogliere i dati provenienti dalle stazioni metereologiche, collezionarli rendendoli disponibili attraverso paradigma Open Data e fornire utili strumenti di visualizzazione, sia in tempo quasi-reale che in formato di report storicizzato.

La base di partenza di TOOLSMART è rappresentata dal progetto #SmartME, un progetto di crowdfunding per la realizzazione di un'infrastruttura di servizi "smart" all'interno della città di Messina.

I requisiti tecnologici di base si fondano, in entrambi i casi, sul paradigma open source, pertanto per software, hardware e dati sono stati adottati modelli di tipo "aperto": stack4Things è infatti il framework di base utilizzato che costituisce un'evoluzione di OpenStack. L'hardware di riferimento adottato è basato sulla scheda Arancino, sviluppata secondo modalità "open", che integra calcolo (Raspberry PI) e controllo (Arduino) in un unico dispositivo. I dati raccolti dal sistema sono infine resi disponibili in modalità Open data a chiunque voglia utilizzarli.

La **piattaforma TOOLSMART**, per assolvere le funzioni sopra specificate, fa riferimento 4 elementi principali:

- **Middleware Stack4Things**, uno strato software che consente la gestione remota dei dispositivi dislocati sul territorio. Consente tra l'altro la modifica remota del software in esecuzione e quindi la personalizzazione dei dispositivi a seconda delle esigenze. Stack4Things è dotato di un portale di gestione dei dispositivi che visualizza la loro posizione, il loro stato di connettività e tutte le informazioni necessarie agli amministratori della Smart City.

Stack4Things è un progetto open source appartenente all'ambiente OpenStack. Essendo un progetto appartenente all'ambiente OpenStack si compone di due sottocomponenti che ne gestiscono gli aspetti di coordinamento (*IoTronic* - lato cloud) e della gestione dei dispositivi

(*Lightning-Rod* - lato dispositivo). Le sue peculiarità sono: Comunicazione immune da problemi relativi al Natting; WebSocket based (Protocollo WAMP); Multi-tenancy; Gestione totale del dispositivo attraverso la piattaforma; Estendibilità del software operante sul dispositivo

- **CKAN** (repository), un sistema open source per l'immagazzinamento, la catalogazione e la distribuzione di dati, quali ad esempio fogli di calcolo o contenuti di database
- **Stazioni Meteorologiche basate su dispositivi Linux-based con software di gestione compatibile con Stack4Things** dislocati sul territorio e connessi tramite Wifi, 4G o LoRa. Tali dispositivi sono dotati di sensori di vario genere a bordo e inviano i dati raccolti ad un repository centrale
- **Mappa Interattiva:** consente la visualizzazione all'interno di una mappa OpenStreetMap-based dei dispositivi gestiti dal TOOLSMART, in forma aggregata e come elementi interattivi rappresentanti la stazione

Requisiti funzionali abilitanti in grado di influenzare positivamente il trasferimento della Buona Pratica sono:

- Messa a disposizione di sensori basati su dispositivi compatibili con Stack4Things
 - Due sistemi che agiscano da Server (bare metal o virtualizzati), raggiungibili attraverso IP pubblico in grado di eseguire i componenti costituenti la piattaforma
 - Connettività (4G – internet cablato o wifi)
- Tra i requisiti non funzionali si annoverano:
- Certificati di dominio associati ai sistemi Server

Focus: il modello della Città di Torino:

La Città di Torino ha operato nel seguente modo:

- Ha affidato alle risorse in forza ai propri sistemi informativi l'incarico di eseguire il deployment dei componenti "lato server" (infrastruttura Stack4Things e portale di raccolta e visualizzazione dati);
- Ha affidato a CSP l'incarico di installare le centraline presso i siti individuati, realizzare le interfacce per l'invio dei dati su rete LoRa e sviluppare dashboard personalizzate di visualizzazione dati;
- Ha affidato a Smartme.IO il ruolo di co-manutentore
- A livello di scelte architettoniche (dettagliate più approfonditamente nell'output O.3.e), la Città ha optato per:
 - impiego di virtual machine nel VDC CSI Piemonte
 - distribuzione delle funzioni su due server
 - impiego della distribuzione Debian Linux

A fronte di questa architettura, le difficoltà incontrate sono state:

- Stack4Things era stato testato solo su distribuzione Ubuntu e l'installazione eseguita su Debian, seguendo le guide, si è "scontrata" con situazioni non previste (in particolare il DBMS MySQL); Smartme.IO ha messo a punto una guida per Debian e la sperimentazione ha così permesso la rimozione di un potenziale lock-in tecnologico
- L'installazione di CKAN è stata piuttosto lunga e laboriosa

Fattori e skill tecnologiche che contribuiscono ad influenzare positivamente il trasferimento e l'adozione della Buona Pratica sono:

- amministrazione di sistemi Linux

- fondamenti di reti e sicurezza informatica
- fondamenti di programmazione Python e di Bash scripting

Focus: il modello delle altre Città:

Le altre città hanno operato seguendo principalmente due approcci :

A) Istanziazione di VM nei propri datacenter

B) Istanziazione di VM su soluzioni Cloud, quindi in outsourcing

A) Inerentemente alla prima opzione, le amministrazioni hanno ove possibile importato dal formato “.ova” messo a disposizione delle amministrazioni le componenti “Controller” e “Dataportal” preinstallate e con le configurazioni di base già impostate.

A seguito dell’importazione delle macchine, per ogni amministrazione si è seguita una procedura che può essere riassunta nei seguenti step:

- configurazione dei domini forniti dall’amministrazione, e configurazione dei sistemi dei certificati di dominio (dove forniti) per l’esposizione dei servizi in modalità sicura,
- Dataportal:
 - configurazione sulla dataportal del repository CKAN:
 - creazione utenze del comune
 - recupero dell’API-KEY di sistema dell’amministrazione per pubblicazione successiva dei dati dalle centraline
 - personalizzazioni dell’amministrazione (descrizione, disclaimer, etc)
 - configurazioni mappa interattiva
 - importazione dei flussi aggiornati
 - customizzazione mappa (geolocalizzazione “home” della mappa)
 - setup credenziali
 - configurazione influxdb
 - configurazione policy di retention dei dati
 - configurazione grafana
 - setup credenziali
 - importazione dashboard
 - configurazione datasource (InfluxDB)
- Controller
 - avvio e verifica dei container ospitanti i vari servizi (attraverso nome dominio dell’amministrazione)
 - personalizzazione configurazione IoTronic (porte esposizione servizi)
 - Creazione delle credenziali e degli endpoint admin dell’omonimo progetto
 - configurazione dashboard amministratore per IoTronic con dominio dell’ amministrazione
 - Registrazione su IoTronic delle centraline prima dell’installazione e verifica connessione tra dispositivo e Controller attraverso il dominio dell’amministrazione
- Registrazione dispositivi su dataportal attraverso i tool forniti

Maggiori dettagli sulle succitate operazioni sono reperibili nella guida “How to Deploy IoTronicJS”.

B) Nelle amministrazioni in cui non si sono potute importare le VM perchè istanziate in outsourcing o per altre difficoltà, prima di poter eseguire gli step descritti nei precedenti casi si è dovuta realizzare un’installazione basata su degli script docker-compose based.



Come nota a margine si vuole documentare alcune difficoltà riscontrate durante il processo di installazione:

- presso il comune di Lecce, uno dei comuni le cui VM sono state installate su piattaforma esterna all'amministrazione, tale sistema impediva l'esecuzione dei docker compose in quanto alcuni repository risultavano non raggiungibili dall'ambiente virtualizzato. In tal caso anzichè installare i container direttamente sulla macchina virtuale, si sono realizzate installazioni su testbed interni dai quali si sono importati i container direttamente negli ambienti virtualizzati in outsourcing.
- presso il comune di Messina, si era optato per l'importazione delle VM ma la piattaforma usata internamente al datacenter non consentiva l'importazione delle stesse VM importate in altre amministrazioni. Per tale ragione anche qui si è dovuta operare un'installazione a mezzo dei docker-compose script.
- presso le altre Pa coinvolte (Padova, Lecce e Siracusa) l'installazione - conclusa una lunga fase iniziale di comprensione dei prerequisiti di base - è avvenuta senza particolari criticità.

Fase "Trasferimento e adozione della BP":

Di seguito la documentazione utile da utilizzare ai fini dell'implementazione di tale fase:

- Codice sw sorgente compilato della soluzione con la debita documentazione a corredo, ivi inclusa la descrizione dell'infrastruttura tecnologica della soluzione con link al repository github (per entrambe fare riferimento alla versione 2.3.6):
 - <https://github.com/smartmeio/s4t-lightning-rod>
 - <https://github.com/smartmeio/s4t-iotronic-standalone>
- Documento di licenza d'uso individuata per il sw che ne titola la fruibilità come riuso di sw pubblico, corrispondente alla Licenza Apache2.0 presente all'interno del repository del codice
- Manuale tecnico per l'installazione della soluzione contenente la documentazione di descrizione e guida del processo di installazione e attivazione, consistente nelle seguenti Guide e documenti: Toolsmart Admin Guide_rev1.5.pdf , Guida Iotronic Standalone 2.3.6 - LR 2.3.7 - rev1.0.pdf, dashboard-smartme.io.pdf
- Guida Starter Kit (SME_2018_SK_MANUALE_ITA_1.1.pdf) e Guida WS (SME_2018_WS_GUIDA_ITA_1.1)
- Virtual Machine con soluzioni software preinstallate e pronte, a seguito dei passi relativi alla prima configurazione, per la registrazione delle centraline, disponibili sul Repository dello spinoff di SMartme.IO e accessibili tramite password dedicate <https://server.smartme.io/toolsmart/1.0/vms/>

Fase Gestione a regime della BP:

Tra le azioni necessarie, si individuano fra l'altro:

- Il monitoraggio dei dispositivi e dei due componenti Controller e Dataportal, consentendo il corretto funzionamento del sistema
- Una capillare formazione /attività di inclusione nelle scuole o rivolta alla cittadinanza
- La diffusione degli output relativi alla Buona Pratica, delle tecnologie in seno alla stessa adottate, delle applicazioni e i servizi sviluppati, oltre che della possibilità di ottenere nuovi casi d'uso ai quali applicare la BP
- La diffusione, attraverso incontri mirati con il tessuto imprenditoriale, di possibili partenariati atti a diffondere la BP ed eventuali evoluzioni della stessa al fine di costituire una base solida sulla quale applicare l'approccio Software-Defined (cfr. O3.b).

Allegati:

Oltre ai link sopra forniti, si elencano di seguito i seguenti documenti:

- Al. 1 - Toolsmart Admin Guide_rev1.5.pdf
- All. 2 - Guida Iotronic Standalone 2.3.6 - LR 2.3.7 - rev1.0.pdf
- All. 3 - Guida Starter Kit (SME_2018_SK_MANUALE_ITA_1.1.pdf)
- All. 4 - Guida WS (SME_2018_WS_GUIDA_ITA_1.1)
- All. 5 - Dashboard-smartme.io.pdf

Note:

Tale output accopra:

- l'Ex-output "Codice software sorgente compilato del middleware basato su Opensatck e cloud computing"
- l'Ex-Output "Manuale tecnico per l'implementazione"
- l'Ex-Output "Manuale su specifiche tecniche acquisto sensori low-cost basati su open hardware e /o applicazioni/servizi a valore aggiunto basati sui dati"



TOO(L)SMART - Admin guide

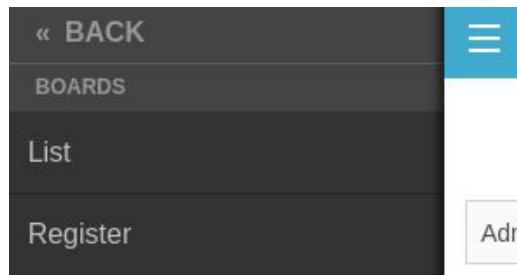
Rev. 1.5 - 09/06/2020

Indice

Registrazione device su lotronic	3
Rimozione di un device da lotronic	5
Reset identità device	6
Registrazione Device su CKAN	7
Script “ckan_register_device.py”	7
Struttura del CSV	8
Plugin tool_sender	9
Attivazione nuova centralina	12
Riavvio servizi sulle VM	14
Rinnovo certificati sulle VM	15

Registrazione device su Iotronic

Per collegare una nuova centralina e poterla gestire da remoto per mezzo di Iotronic è necessario preventivamente registrarla sulla piattaforma (VM controller).



Navigando nel menù di sistema, nella sezione “Boards” sarà possibile raggiungere il panel di registrazione “Register”.

Add new Board to the Cloud X

Board UUID
 New

Label

Password (between 4 and 60 digits)
 Show

Description

Connectivity

Layout

Latitude	Longitude	Altitude
e.g.: 38.123	e.g.: 15.123	e.g.: 50.123

Nel form di registrazione sarà necessario specificare:

- un device ID (generato automaticamente da Iotronic in formato UUID)
- una label

- una password per l'accesso ad Iotronic
- una descrizione (indirizzo del plesso dove la centralina sarà installata)
- (opzionale) informazioni sul tipo di connettività utilizzata dal device per connettersi ad Iotronic:
 - Ethernet (specifica del MAC)
 - WiFi (specifica del MAC)
 - Mobile (specifica del codice ICCID della SIM)
- il layout riconducibile al device (piattaforma HW e SW):
 - Arancino - Arancino OS
- le coordinate geografiche (le stesse del file CSV compilato per Toolsmart)

The screenshot shows a registration form for a new device. At the top, there is a section for "Extra user defined data (json)" with a "Scegli file" button and a message indicating "Nessun file selezionato". Below this is a large text area for "Project" and "User", both currently showing "--". Underneath, there is a dropdown for "Enable email notification (connection status)" set to "False". A prominent blue "Register" button is located in the center-right. At the bottom, there is a section labeled "Output" with a large empty text area.

- (opzionale) extra info (JSON):
 - informazioni supplementari specifiche di uno scenario.
- il progetto/flotta a cui afferirà il nuovo device (es. *Admin*)
- l'utente, il referente/responsabile della gestione del device (es. *Admin*)
- (opzionale) abilitare il sistema di notifiche (se abilitato in Iotronic) tramite email che avvertirà l'utente referente/responsabile del device in caso di disconnessione prolungata del suddetto.

Rimozione di un device da Iotronic

La rimozione di un device da Iotronic è possibile effettuarla dal panel “*Unregister*” raggiungibile dal menù di sistema, nella sezione “*Boards*”.

The screenshot shows the Iotronic interface with a sidebar on the left and a main content area on the right.

Sidebar (Left):

- « BACK
- BOARDS
- List
- Register
- Inject configuration
- Update Info
- Unregister

Main Content Area (Right):

Unregister Board

Enable / Disable per project delete

Show: 10 Search:

label	board_id	status
lede-mctx	lede-mctx	D
lede-test-00	lede-test-00	D
rosa4	rosa4	D
sme-arancino	5b04a27e-7b86-7e3d-2c63-bd7ee6288b16	C
sme-test-raspbian	70065bc7-000a-8608-8aaa-5f9ea49c793f	D
sme-test-ro	ad6167b8-9a6e-f436-203f-58b18d64c570	D
sme-ubuntu64	ffaa2c26-8a17-4d41-c837-61b9be3abd11	D
st-artik	af01f0fe-dc15-1a08-d27f-3602f6419744	C
st-dev-smartme	473a3576-e14b-ab91-51b5-ba3830dc4e6e	D

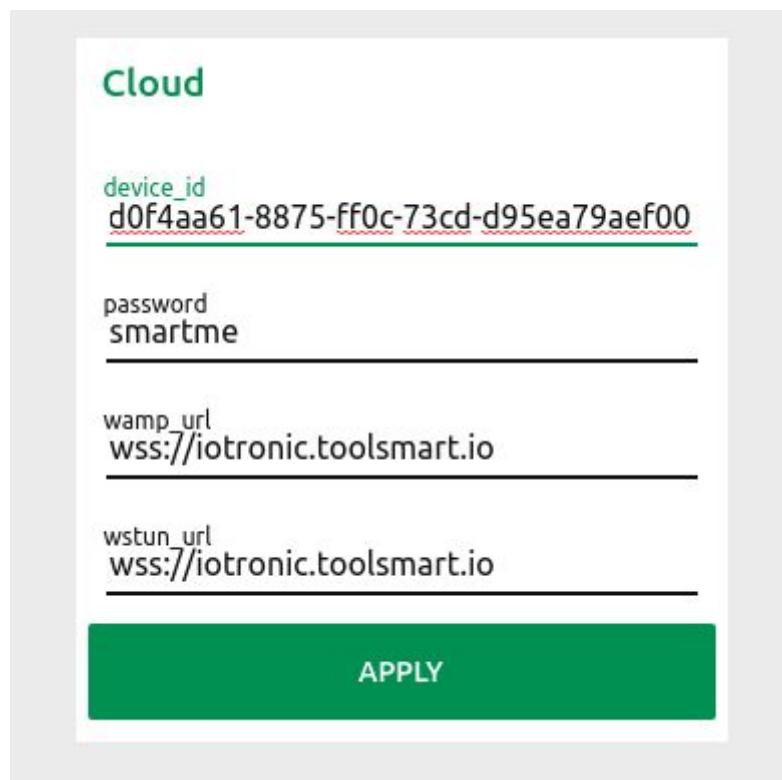
Showing 11 to 19 of 19 entries

Previous 1 Next

Sarà possibile rimuovere, uno o più device, oppure tutti i device di un intero progetto/flotta.

Reset identità device

1. Lato cloud (Iotronic)
 - a. Rimuovere da Iotronic il device [vedi sezione apposita]
 - b. Registrare il device con la nuova identità [vedi sezione apposita]
2. Lato device (configurazione di Lightning-rod)
 - a. Loggarsi sulla dashboard della centralina nella sezione “Cloud”:
 - i. <http://<IP-LOCALE-CENTRALINA>/ui/#!/3>
 - b. Immettere nel form i nuovi dati identificativi del device



1. specificare il nuovo “device_id”
2. la nuova “password”
3. gli url degli endpoint di Iotronic rimarranno immutati
4. applicare le modifiche e attendere la riconnessione del device ad Iotronic che vedremo riconnesso sulla dashboard di Iotronic.

Registrazione Device su CKAN

Script “ckan_register_device.py”

Il processo di registrazione dei dispositivi su CKAN è un processo automatizzato basato sull'esecuzione di uno script che usa un file csv come input da cui attingere le informazioni necessarie per la registrazione. Lo script è composto di due parti; nella prima parte, in cui sono immagazzinate le variabili che verranno utilizzate durante l'esecuzione, necessita dell'intervento dell'amministratore, per la corretta valorizzazione.

Le variabili da settare sono:

1. dataportal = "XXX.XXX.XXX.XXX" #<IPv4 DATAPORTAL>"
2. api_key = "XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX" #<API_KEY DI CKAN>'
3. metrics = ["temperature", "brightness", "humidity", "pressure", "gas", "noise", "part0.3", "part0.5", "part1.0", "part2.5", "part5.0", "part10", "pm1.0", "pm2.5", "pm10", "wind_direction", "wind_speed", "precipitation"]

In particolare il punto 2 è un parametro di cui l'amministratore è a conoscenza già dallo step iniziale di configurazione dell'ambiente CKAN preinstallato sulla vm “dataportal” fornитagli, mentre il punto 3 rappresenta l'insieme di metriche che sono fornite dal device e che saranno esposti da CKAN sotto forma di Datastore interni al Dataset rappresentante il device. Sarà possibile estendere o ridurre il quantitativo di metriche esposte a seconda delle possibilità di sensing del nodo (o device che dir si voglia) installato e del plugin iniettato all'interno dello stesso.

Dopo aver configurato le variabili dello script sarà sufficiente lanciarlo con il comando:

python3 ckan_register_device.py

Lo script dopo aver letto i valori presenti all'interno del file csv in maniera iterativa comincia il processo di registrazione del device, processo che si divide in quattro fasi:

1. Registrazione del dataset.
2. Registrazione dei metadati relativi ai datastore del dataset. (in *Figura 2 STEP 1*)
3. Registrazione del datastore riepilogativo delle caratteristiche del device, denominato “sensors”. (in *Figura 2 STEP 2*)
4. Registrazione e creazione dei datastore. (in *Figura 2 STEP 3*)

Ogni iterazione è preceduta da un report a video delle informazioni che stanno per essere registrate come metadati del dataset e datastore in CKAN, corrispondenti alle informazioni recuperate dal file csv come mostrato nella *Figura 1*

```

Argument List: ['ckan_register_device.py']
Board ID: be7a9e15-9894-426d-a73b-21013a7861f5
Label: ws-0002
Altitude: 0
Latitude: 39.0001
Longitude: 16.0001
Manufacturer: Smartme.IO
Model: Arancino
Model: Arancino

```

Figura 1: Dati riepilogativi della registrazione

```

[User@pmpc01:~/Downloads/TOOLS/CKAN/SCRIPT_CONF]$ python3 ckan_register_device.py
Fine lettura CSV
Argument List: ['ckan_register_device.py']
Board ID: be7a9e15-9894-426d-a73b-21013a7861f5
Label: ws-0002
Altitude: 0
Latitude: 39.0001
Longitude: 16.0001
Manufacturer: Smartme.IO
Model: Arancino
Model: Arancino
('server': PasteGISServer/0.5 Python/2.7.9', 'date': 'Wed, 30 Oct 2019 09:35:04 GMT', 'content-type': 'application/json;charset=utf-8', 'content-length': '1770', 'status': '200'}
b'{"help": "http://212.189.207.99:5000/api/3/action/help_show?name=package_create"}', 'success': true, 'result': {'license_title': None, 'relationships_as_object': [], 'private': false, 'maintainer_email': null, 'num_tags': 1, 'id': '10928652-5f86-4db8-8997-43594a444943', 'name': 'testbed', 'version': '1.0', 'tags': [], 'resources': [], 'num_resources': 0, 'tags': []}, 'vocabulary_id': null, 'state': 'active', 'display_name': 'testbed', 'id': 'la3c6795-1064-4090-83fb-5272bf394a3', 'name': 'testbed'}, 'groups': [], 'license_id': null, 'relationships_as_subject': [], 'organization': {'oe_description': 'Created: 2019-09-17T16:15:39.527300', 'title': 'toolsmart', 'name': 'toolsmart', 'is_organization': true, 'state': 'active', 'image_url': 'https://pbs.twimg.com/profile_images/1173931809316491264/WK3tDw_486x486.jpg', 'revision_id': '7bb8a14c-7b05-4d85-beab-000000000000', 'id': '10928652-5f86-4db8-8997-43594a444943', 'version': '1.0', 'notes': 'Open Data', 'owner_org': 'e6a37721-3487-43ea-8d2c-2b9595d73e84', 'extras': [{"key": "Altitude", "value": "0"}, {"key": "Label", "value": "ws-0002"}, {"key": "Latitude", "value": "39.0001"}, {"key": "Longitude", "value": "16.0001"}, {"key": "Manufacturer", "value": "Smartme.IO"}, {"key": "Model", "value": "Arancino"}, {"key": "ProdID", "value": "SMESSGAR000002XXXXXX"}], 'key': 'label', 'value': 'ws-0002'}, {'key': 'Latitude', 'value': '39.0001'}, {'key': 'Longitude', 'value': '16.0001'}, {'key': 'Manufacturer', 'value': 'Smartme.IO'}, {"key": "Model", "value": "Arancino"}, {"key": "ProdID", "value": "SMESSGAR000002XXXXXX"}}, 'revision_id': '9e6a2e2f-b3de-422d-88ef-21013a7861f5'}, 'STEP 1 - Sensors datastore UUID: 84bd7bbc-6249-419b-a717-d86781b66dc5'
STEP 2 - Metrics datastores creation...
('server': PasteGISServer/0.5 Python/2.7.9', 'date': 'Wed, 30 Oct 2019 09:35:06 GMT', 'content-type': 'application/json;charset=utf-8', 'content-length': '200', 'status': '200'}
b'{"help": "http://212.189.207.99:5000/api/3/action/help_show?name=datastore_create"}', 'success': true, 'result': {'id': '731e4976-72d9-422f-1561d9e125e5', 'name': 'Metrics datastores', 'type': 'group', 'description': 'A group of datastores for metrics data.', 'private': false, 'version': '1.0', 'tags': [], 'resources': [], 'num_resources': 0, 'tags': []}, 'vocabulary_id': null, 'state': 'active', 'display_name': 'Metrics datastores', 'id': '10928652-5f86-4db8-8997-43594a444943', 'name': 'Metrics datastores'}, 'groups': [], 'license_id': null, 'relationships_as_subject': [], 'organization': {'oe_description': 'Created: 2019-09-17T16:15:39.527300', 'title': 'toolsmart', 'name': 'toolsmart', 'is_organization': true, 'state': 'active', 'image_url': 'https://pbs.twimg.com/profile_images/1173931809316491264/WK3tDw_486x486.jpg', 'revision_id': '7bb8a14c-7b05-4d85-beab-000000000000', 'id': '10928652-5f86-4db8-8997-43594a444943', 'version': '1.0', 'notes': 'Open Data', 'owner_org': 'e6a37721-3487-43ea-8d2c-2b9595d73e84', 'extras': [{"key": "Label", "value": "ws-0002"}, {"key": "Latitude", "value": "39.0001"}, {"key": "Longitude", "value": "16.0001"}, {"key": "Manufacturer", "value": "Smartme.IO"}, {"key": "Model", "value": "Arancino"}, {"key": "ProdID", "value": "SMESSGAR000002XXXXXX"}], 'key': 'label', 'value': 'ws-0002'}, {'key': 'Latitude', 'value': '39.0001'}, {'key': 'Longitude', 'value': '16.0001'}, {"key": "Manufacturer", "value": "Smartme.IO"}, {"key": "Model", "value": "Arancino"}, {"key": "ProdID", "value": "SMESSGAR000002XXXXXX"}}, 'revision_id': '9e6a2e2f-b3de-422d-88ef-21013a7861f5'}, 'STEP 3 - Datastore UUID: 84bd7bbc-6249-419b-a717-d86781b66dc5'
Datastore UUID: 1a1fffd1-731e-4976-82df-1561d9e125e5
('server': PasteGISServer/0.5 Python/2.7.9', 'date': 'Wed, 30 Oct 2019 09:35:06 GMT', 'content-type': 'application/json;charset=utf-8', 'content-length': '154', 'status': '200'}
b'{"help": "http://212.189.207.99:5000/api/3/action/help_show?name=datastore_create"}', 'success': true, 'result': {'id': '731e4976-72d9-422f-1561d9e125e5', 'name': 'Metrics datastores', 'type': 'group', 'description': 'A group of datastores for metrics data.', 'private': false, 'version': '1.0', 'tags': [], 'resources': [], 'num_resources': 0, 'tags': []}, 'vocabulary_id': null, 'state': 'active', 'display_name': 'Metrics datastores', 'id': '10928652-5f86-4db8-8997-43594a444943', 'name': 'Metrics datastores'}, 'groups': [], 'license_id': null, 'relationships_as_subject': [], 'organization': {'oe_description': 'Created: 2019-09-17T16:15:39.527300', 'title': 'toolsmart', 'name': 'toolsmart', 'is_organization': true, 'state': 'active', 'image_url': 'https://pbs.twimg.com/profile_images/1173931809316491264/WK3tDw_486x486.jpg', 'revision_id': '7bb8a14c-7b05-4d85-beab-000000000000', 'id': '10928652-5f86-4db8-8997-43594a444943', 'version': '1.0', 'notes': 'Open Data', 'owner_org': 'e6a37721-3487-43ea-8d2c-2b9595d73e84', 'extras': [{"key": "Label", "value": "ws-0002"}, {"key": "Latitude", "value": "39.0001"}, {"key": "Longitude", "value": "16.0001"}, {"key": "Manufacturer", "value": "Smartme.IO"}, {"key": "Model", "value": "Arancino"}, {"key": "ProdID", "value": "SMESSGAR000002XXXXXX"}], 'key': 'label', 'value': 'ws-0002'}, {'key': 'Latitude', 'value': '39.0001'}, {'key': 'Longitude', 'value': '16.0001'}, {"key": "Manufacturer", "value": "Smartme.IO"}, {"key": "Model", "value": "Arancino"}, {"key": "ProdID", "value": "SMESSGAR000002XXXXXX"}}, 'revision_id': '9e6a2e2f-b3de-422d-88ef-21013a7861f5'}, 'Datastore UUID: dddbab27-c408-4a61-b605-a71586277cb
('server': PasteGISServer/0.5 Python/2.7.9', 'date': 'Wed, 30 Oct 2019 09:35:11 GMT', 'content-type': 'application/json;charset=utf-8', 'content-length': '308', 'status': '200')

```

Figura 2- Output prodotto dallo script ckan_register_device.py

Struttura del CSV

Il file *boardList.csv* è un file che deve essere posto all'interno della stessa directory in cui risiede lo script di registrazione dei device su CKAN. la struttura è quella classica di un csv per cui i campi saranno separati da una semplice “,” in tabulazione e ogni nuova riga è realizzata con un a capo come indicato in *Figura 3*.

deviceid	label	altitude	longitude	latitude	model	productid
ae7a9e15-9894-426d-a73b-21013a7861f5	ws-0001	0	16.0001	39.0001	Arancino	SMESSGAR000010XXXX
be7a9e15-9894-426d-a73b-21013a7861f5	ws-0002	0	16.0001	39.0001	Arancino	SMESSGAR000020XXXX
ce7a9e15-9894-426d-a73b-21013a7861f5	ws-0003	0	16.0001	39.0001	Arancino	SMESSGAR000030XXXX

Figura 3: Esempio di file csv

I campi che dovranno essere contenuti sono i seguenti:

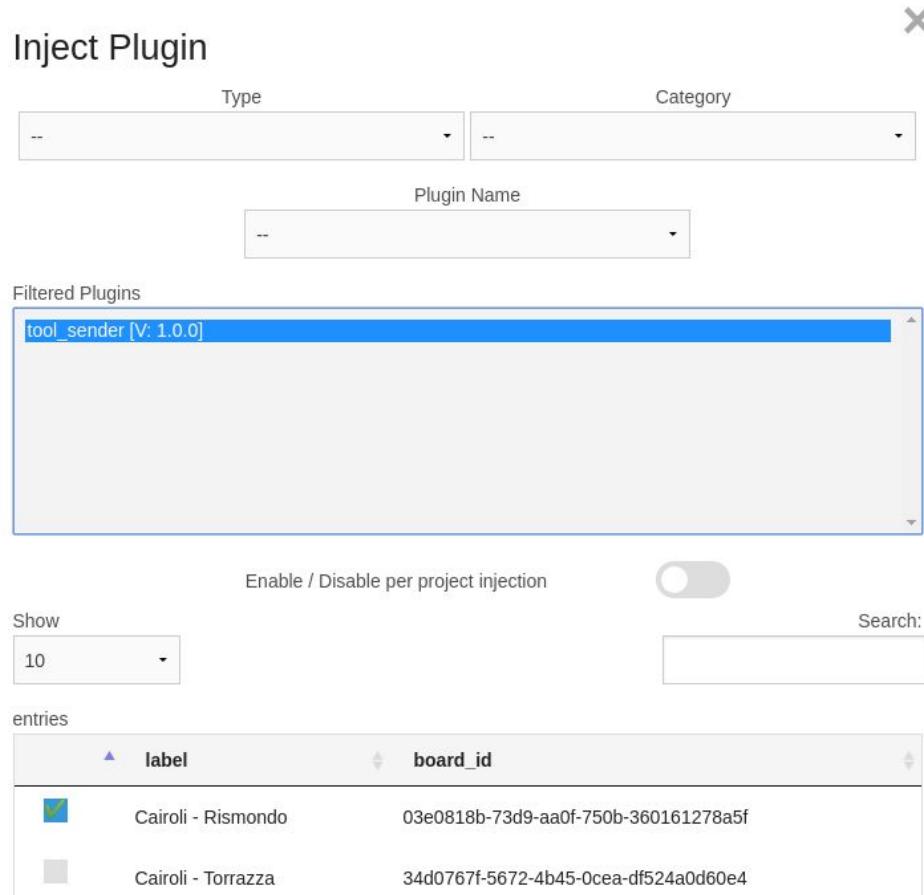
- deviceid, Identificativo fornito da IoTTronic all'atto della registrazione del device sul framework di gestione del device
- label, Etichetta¹ che si vuole assegnare al device. altitude,
- longitude, La logitudine corrispondente al punto di installazione del dispositivo
- latitude, La latitudine corrispondente al punto di installazione del dispositivo
- model, Il modello del dispositivo, attualmente tutti i modelli sono di tipo Arancino
- productid, Questo campo è presente sull'etichetta fisica presente sul case del device.

¹ NB: l'etichetta non deve contenere spazi nè frasi nè disclaimer o altro che non sia una semplice etichetta che aiuta mnemonicamente ad identificare il device nella sua visualizzazione sulla mappa

Plugin tool_sender

Smartme.IO ha realizzato il plugin “*tool_sender*” per inviare le metriche delle stazioni meteo su CKAN.

Il plugin verrà iniettato all'interno delle centraline mediante lotronic, tramite il panel “Inject Plugin”:

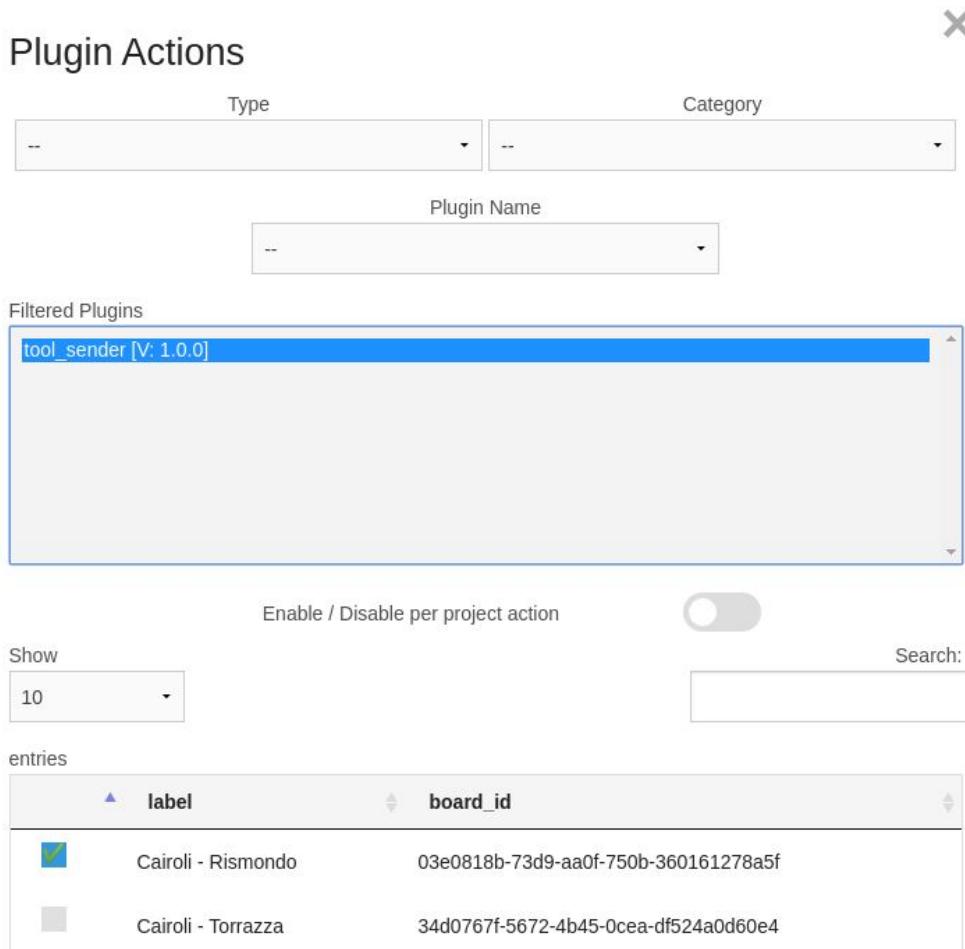


Una volta selezionato il plugin “*tool_sender*” e la centralina è necessario settare a “True” il flag “On Boot”, per far sì che il plugin venga messo in esecuzione automaticamente all'avvio della centralina:

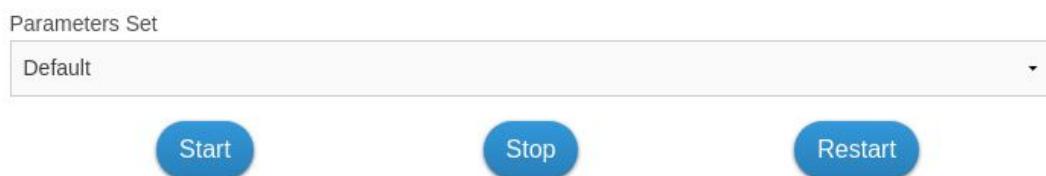
On Boot	Force
True	False
<input type="button" value="Inject"/>	

Lightning-rod (LR), che è il probe in esecuzione sui device, riceve il plugin con i parametri di input, da lotronic e li memorizza all'interno dello storage della centralina.

Il plugin dovrà essere messo in esecuzione dopo la fase di inject, tramite il panel “*Plugin Actions*”, selezionando il plugin e la centralina:



Sarà possibile specificare il set di parametri :



- **Default:** sono i parametri specificati in fase di creazione del plugin;
- **Latest:** set di parametri specificati nell'ultima esecuzione del plugin;
- **New:** selezionando tale opzione verrà data la possibilità di caricare un nuovo set di parametri per questa specifica esecuzione e che verranno memorizzati da lotronic come nuovi parametri “*Latest*”.

I parametri attuali di “Default” del plugin “*tool_sender*” sono i seguenti:

```
{  
    "metrics":  
        ["temperature", "humidity", "pressure", "part0.3", "part0.5", "part1.0", "part2.5",  
         "part5.0", "part10", "pm1.0", "pm2.5", "pm10", "wind_direction", "wind_speed",  
         "precipitation"],  
    "period": 600,  
    "ckan_key": "<CKAN-API-KEY>",  
    "ckan_endpoint": "<CKAN-URL>",  
    "debug": "true"  
}
```

Descrizione:

- **“metrics”**: lista metriche abilitate nello scenario
- **“period”**: tempo in secondi ogni quanto verrà inviata una metrica a CKAN
- **“ckan_key”**: API key di CKAN per consentire l’invio delle metriche nei datastore del device; la KEY deve essere associata ad un user di tipo “Editor” dell’Organization “toolsmart”
- **“ckan_endpoint”**: indirizzo di CKAN comprensivo:
 - del suffisso “http://” o “https://”
 - dell’IP e porta (5000) o del dominio tramite cui raggiungere CKAN
- **“debug”**: [“true” | “false”] per abilitare i log all’interno del plugin

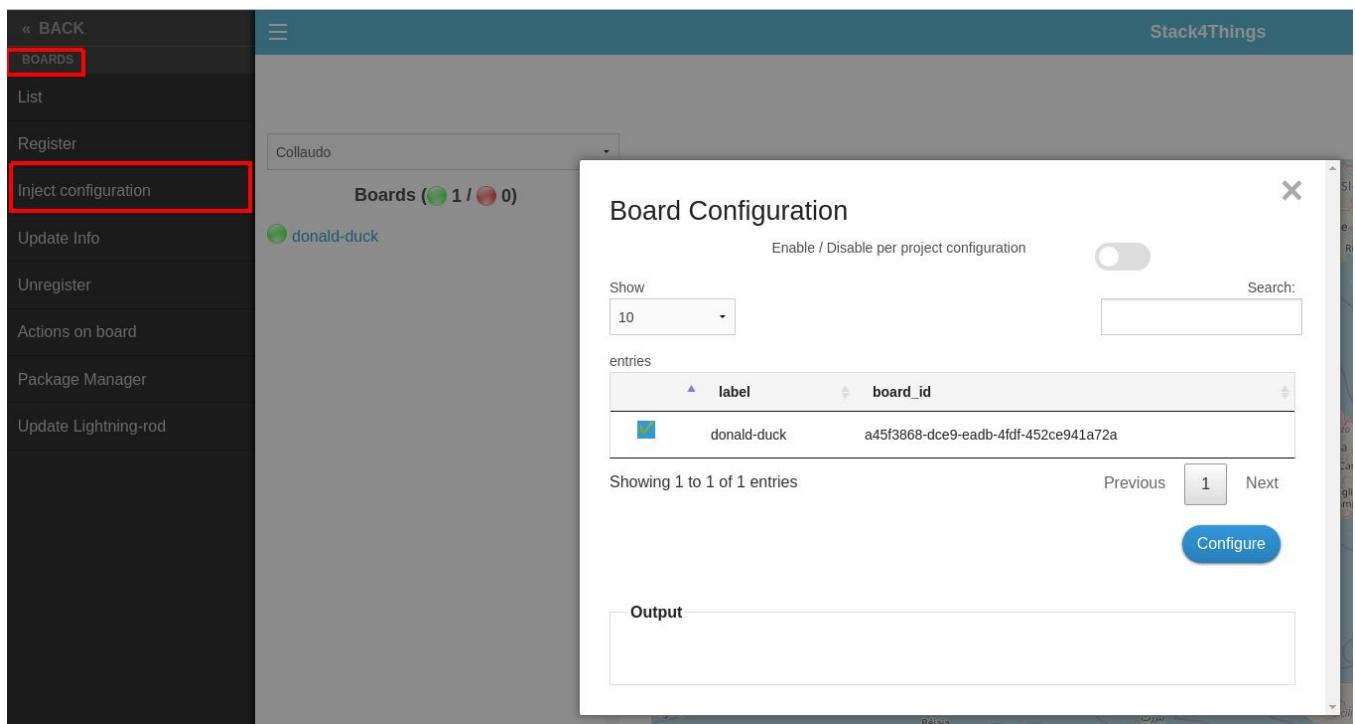
Attivazione nuova centralina

Requisiti:

- registrazione ad lotronic
- centralina online su lotronic
- plugin “**tool_sender**” correttamente disponibile su lotronic per l’inject sulle centraline

Attivazione:

1. Inject nuova configurazione: Lightning-rod (LR) si riavvia automaticamente:
 - a. dal menù principale della dashboard di lotronic, selezionare il menù **“Boards”** e successivamente **“Inject configuration”** come riportato qui di seguito:



2. Dopo il riavvio di LR effettuare l’inject del plugin **“tool_sender”** con parametri di **default** e flag **“onBoot”** attivo (per ulteriori dettagli vedi sezione **“Plugin tool_sender”**)
3. Start plugin **“tool_sender”**
4. Dopo aver verificato la ricezione dei dati su CKAN, riavviare la centralina da remoto per fissare i cambiamenti sul file system:
 - a. dal menù principale della dashboard di lotronic, selezionare il menù **“Boards”** e successivamente **“Actions on board”** come riportato qui di

seguito, e selezionare come “**Action**” --> “**reboot**”:

The screenshot shows the Stack4Things application interface. On the left, there is a sidebar with the following menu items:

- « BACK
- BOARDS** (highlighted with a red box)
- List
- Register
- Inject configuration
- Update Info
- Unregister
- Actions on board** (highlighted with a red box)
- Package Manager
- Update Lightning-rod

The main area displays a list of boards under the heading "Boards (1 / 0)". One board is listed: "donald-duck". A modal dialog titled "Board Action" is open over the main content. The dialog has the following fields and controls:

- Action:** A dropdown menu set to "reboot".
- Enable / Disable per project action:** A toggle switch.
- Show:** A dropdown menu set to "10".
- Search:** An input field.
- entries:** A table listing one entry:

label	board_id
donald-duck	a45f3868-dce9-eadb-4fdf-452ce941a72a
- Showing 1 to 1 of 1 entries**
- Previous** and **Next** buttons.
- Delay Time:** An input field with "(in seconds)" placeholder text.
- Execute:** A blue button.
- Output:** A large text area.

Riavvio servizi sulle VM

I servizi su entrambe le VM sono containerizzati mediante il servizio Docker. Ogni servizio di default è abilitato affinchè parta al boot della VM.

Se dovesse essere necessario riavviare uno dei servizi:

- sulla VM controller:
 - docker restart iotronic_db
 - docker restart iotronic-crossbar
 - docker restart iotronic-wstun
 - docker restart iotronic-standalone
 - docker restart iotronic-web
- sulla VM dataportal:
 - riavvio di CKAN:
 - docker restart db redis solr datapusher ckan
 - docker restart influxdb
 - docker restart grafana
 - docker restart nodered

NB: se dovesse essere necessario riavviare tutti i servizi sulle singole VM, procedere col loro riavvio seguendo l'ordine sopra riportato.

Rinnovo certificati sulle VM

Si dovrà agire su entrambe le VM. Nello specifico una volta loggati ci si dovrà recare nella cartella /etc/nginx/sites-available/

- nella VM controller dovremo modificare i seguenti file:
 - iotronic-standalone
 - iotronic-ui
- nella VM dataportal:
 - ckan
 - grafana
 - nodered

Per ognuno di tali file si dovrà aggiornare i nomi dei nuovi certificati ed il loro eventuale cambio di locazione; nello specifico si dovranno aggiornare queste due voci

- ssl_certificate <LOCAZIONE-PUBLIC-KEY>;
- ssl_certificate_key <LOCAZIONE-PRIVATE-KEY>;

Nei file dove è presente, si dovrà modificare anche questa voce:

- ssl_client_certificate <LOCAZIONE-CHAIN-CERTIFICATE>;

Se invece si volesse procedere con la rinominazione dei file dei certificati recarsi nella cartella dove risiedono attualmente i certificati e procedere con la sostituzione dei vecchi certificati con i nuovi, ponendo attenzione a rinominarli come i vecchi.

Come ultimo step bisognerà riavviare il servizio NGINX:

- systemctl restart nginx



Stack4Things

Iotronic v2.3.6 - LR 2.3.7

Rev. 1.0 - 31/10/2019



Premessa	4
Stack4Things	5
Struttura modulare	6
Iotronic	7
File di sistema	7
settings.json	8
API documentation	13
Lightning-rod	14
Moduli Manager in Lightning-rod	14
Stati di funzionamento	15
File di sistema	15
authentication.json	16
settings.json	17
API exposed	19
Iotronic-UI	20
WSTUN	23
Platform administration	24
Users management	25
Projects/Fleets management	26
Layouts management	27
Requests management	28
Device Manager	30
Gestione identità del device in Iotronic	31
Registrazione di un device in Iotronic	31
Rimozione di un device da Iotronic	34
Aggiornamento delle informazioni di un device	35
Inject di una nuova configurazione nel device	37
Lista device in Iotronic	38
Esecuzione di azioni predefinite sui device	39
Gestione pacchetti software del device	40
Aggiornamento di Lightning-rod	41
Procedure operative	42
Primo collegamento del device ad Iotronic	42
Backup/restore configurazione di un device	43
Service Manager	44
Esposizione servizio	46
Ripristino esposizione servizio	48



Revoca esposizione servizio	49
Plugin Manager	50
Plugin workflow	51
Creazione di un nuovo plugin	52
Plugin sincrono	53
Plugin asincrono	57
Plugin API	61
PluginExec development tool	62
Upload plugin in Iotronic	65
Versioning	66
Review & Release di un plugin	67
Inject del plugin in un device	69
Plugin actions	71
Update plugin	73
Rimozione del plugin dal device	75
Rimozione del plugin da Iotronic	77
Show plugin logs	78
Show plugin in device	79
Plugins in Lightning-rod	80
File di sistema	80



Premessa

Il documento fa riferimento alle versioni attualmente rilasciate:

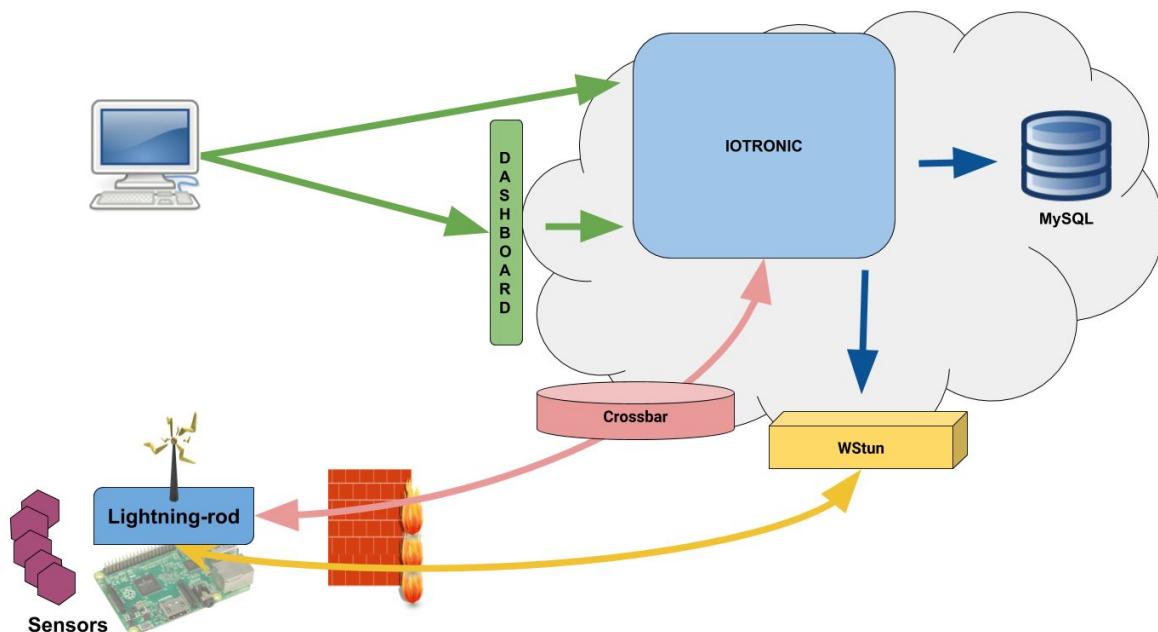
- Iotronic v2.3.6
- Lightning-rod v2.3.7
- Iotronic-UI v2.3.9

Stack4Things

Stack4Things è un progetto open-source che permette il management di device IoT remoti senza la necessità di aver conoscenza della loro configurazione di rete (con o senza firewall) e della tecnologia hardware usata.

Fornisce un out-of-the-box experience per i più popolari sistemi embedded e sistemi mobile. Sviluppato in *NodeJS*, si compone di una parte server (cloud-side), **iotronic**, una parte client (device-side) **Lightning-rod (LR)** e di un'interfaccia web di management.

Il canale di comunicazione tra iotronic e le varie istanze di LR è basato sul protocollo **WAMP (Web Application Messaging Protocol)**, sfruttando la libreria **Autobahn**. La gestione della comunicazione avviene lato cloud per mezzo dell'istanza server di **Crossbar.IO**.



Actors:

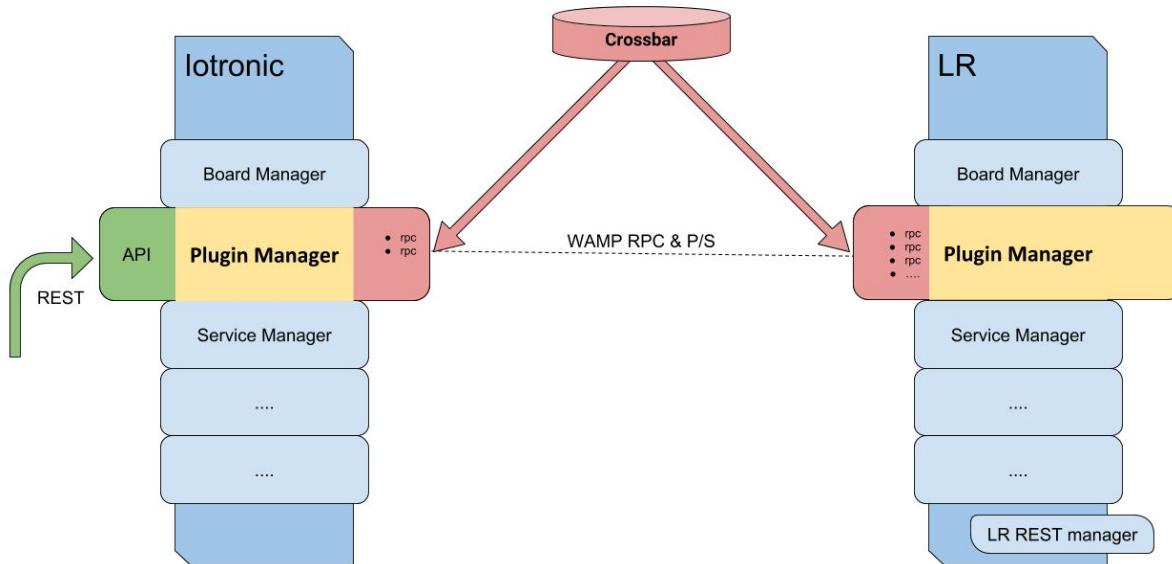
- iotronic
- Lightning-rod
- iotronic-UI (admin dashboard)
- Crossbar server
- WSTUN server (e istanze client sui device)
- Database (Mysql)

Struttura modulare

Sia Iotronic che Lightning-rod posseggono una struttura altamente modulare. Ogni modulo, detto Manager, si occupa di esporre un proprio set di funzionalità.

Lato Iotronic ogni modulo espone un set di API RESTful e optionalmente un set di funzionalità richiamabili col paradigma RPC (*Remote Procedure Call*).

Lato device, Lightning-rod espone un set di chiamate RPC che IoTronic richiamerà per soddisfare le richieste pervenute dalle API RESTful o da IoTronic stesso.



Nell'attuale versione di Stack4Things sono stati implementati e rilasciati i seguenti manager:

- **Device Manager:** modulo base per la gestione delle flotte di device.
 - **Service Manager:** modulo opzionale, per l'esposizione dei servizi in esecuzione sui device.
 - **Plugin Manager:** modulo opzionale, per l'esecuzione sul device di codice (*Python* e *NodeJS*).

Iotronic

Iotronic, nell'architettura *Stack4Things (S4T)*, rappresenta il core cloud-side dell'intera infrastruttura di gestione delle flotte di device IoT. La gestione dei device avviene mediante l'interazione con il probe Lightning-rod - LR (vedi sezione successiva) istanziato sui device.

Iotronic è implementato in linguaggio *NodeJS* e presenta una struttura altamente modulare. Ogni set di funzionalità per la gestione dei device è implementato come un modulo a se stante e definito in S4T “*Manager*”. Ogni manager è indipendente dagli altri e si innesta al core di Iotronic.

Ogni Manager in Iotronic espone un set di API RESTful e opzionalmente, in funzione delle interazioni logiche con LR, può esporre anche delle funzioni RPC, che LR può richiamare da remoto per comunicare eventuali cambiamenti di stato del device o per richiedere informazioni ad Iotronic.

File di sistema

- Iotronic configuration files:
 - /var/lib/iotronic
 - settings.json
 - [OPZIONALE] ssl/
 - cartella dove risiedono i certificati per esporre in HTTPS le API
 - [OPZIONALE] <MANAGERS_FOLDERS>
 - LR source code path:
 - /usr/lib/node_modules/@mdslab/iotronic-standalone/



settings.json

In questo file sono contenute le configurazioni di Iotronic:

- “**server**”:
 - “public_ip”: IP/FQDN dove raggiungere pubblicamente la dashboard di amministrazione (Iotronic-UI)
 - “interface”: DEPRECETED
 - “api_ip”: IP/FQDN dove raggiungere pubblicamente le API
 - “http_port”: porta HTTP su cui sono esposte le API
 - “https”: impostazioni per esporre le API in HTTPS
 - “enable”: [“true” | “false”] abilitare l’esposizione delle API in HTTPS
 - “port”: porta HTTPS su cui sono esposte le API
 - “key”: chiave privata certificato Iotronic
 - “cert”: chiave pubblica certificato Iotronic
 - “db”: configurazioni per permettere ad Iotronic di accedere al database
 - “host”: “<MYSQL-IP>”,
 - “port”: “<MYSQL-PORT>”,
 - “user”: “<IOTRONIC-DB-USER>”,
 - “password”: “<IOTRONIC-DB-PASSWORD>”,
 - “db_name”: “s4t-iotronic”
 - “log”: configurazione del logging
 - “logfile”: “/var/log/iotronic/s4t-iotronic.log”,
 - “loglevel”: “[info | debug | …]”
 - “notifier”: configurazione del sistema di notifiche di Iotronic (email)
 - “email”:
 - “address”: indirizzo email *sender* delle notifiche
 - “password”: “<PASSWORD-EMAIL-SENDER>”
 - “smtp”:
 - “smtp_server”: “<SMTP-SERVER>”
 - “smtp_port”: “<SMTP-PORT>”
 - “smtp_secure”: [“true” | “false”]
 - “enable_notify”: [“true” | “false”]
 - “docs”: configurazioni per la gestione della documentazione delle API da parte di Iotronic

- Iotronic da la possibilità di esporre la propria documentazione delle API e di crearla automaticamente, come descritto nell'apposita [guida](#).
- “auth”: configurazione del sistema di autenticazione dei device e del sistema di API
 - “encryptKey”: chiave scelta in fase di installazione di Iotronic, usata per criptare/decriptare le password in fase di autenticazione e rilascio token per le API
 - “adminToken”: superAdmin token scelto in fase di installazione di Iotronic, utilizzato per la creazione iniziale dell’utente Admin ed il progetto Admin; può essere utilizzato dall’amministratore per richiamare le API senza la necessità di richiedere un token temporaneo.
 - “backend”: modalità di autenticazione; ad oggi unicamente gestita da Iotronic; in futuro si prevede di integrare altri meccanismi di autenticazione (Ldap, Keystone, ecc). [“iotronic”]
 - “expire_time”: validità dei token rilasciati da Iotronic [“30m”, “1h”, “7d”, ecc]
 - “auth_lr_mode”: modalità di autenticazione di LR su Iotronic;
 - “basic”: basata sulla semplice valutazione della registrazione del device ad Iotronic. <NON-SICURA>
 - “password”: basata sulla verifica di una password fornita da LR al momento del login ad Iotronic. [da 4 a 36 caratteri]
 - “certs” <SPERIMENTALE>: basata su certificati gestiti da Iotronic; Iotronic verifica la signature fornita da LR al momento del login: tale signature è generata da LR firmando con la propria *pub-key* (del device) il proprio device-ID. Iotronic verificherà tale signature tramite la stessa *pub-key* del device memorizzata in fase di registrazione del device.
- “wamp”: configurazione di accesso al server Crossbar
 - “url”: websocket url del server Crossbar (wss://)
 - “port”: porta del server Crossbar
 - “ssl”: [“true” | “false”]
 - “realm”: “s4t”
 - “topic_connection”: “board.connection”
 - “crossbar_pub_ip”: IP/FQDN pubblico del server Crossbar utilizzato dalle istanze di LR per collegarsi ad Iotronic.
- “modules”:
 - in questa sezione del file di configurazione viene gestita l’abilitazione dei moduli Manager all’interno di Iotronic. Ogni sezione di un modulo prevede la presenza di un flag “enabled” per abilitare (“true”) o disabilitare (“false”) il modulo.



```
{  
    "plugins_manager": {  
        "enabled": true  
    },  
    etc  
}
```

- ogni modulo potrà avere un proprio set di parametri di configurazione; ad esempio:

```
"services_manager": {  
    "enabled": true,  
    "wstun": {  
        "port_range": {  
            "high": 40100,  
            "low": 40001  
        },  
        "public_ip": ""  
    }  
}
```

Template

```
{
    "config": {
        "server": {
            "interface": "",
            "public_ip": "",
            "api_ip": "",
            "http_port": "8888",
            "https": {
                "enable": "false",
                "port": "",
                "key": "",
                "cert": ""
            },
            "auth": {
                "encryptKey": "",
                "adminToken": "",
                "backend": "iotronic",
                "expire_time": "1h",
                "auth_lr_mode": "basic"
            },
            "db": {
                "host": "localhost",
                "port": "3306",
                "user": "root",
                "password": "",
                "db_name": "s4t-iotronic"
            },
            "log": {
                "logfile": "/var/log/iotronic/s4t-iotronic.log",
                "loglevel": "info"
            },
            "docs": {
                "embedded": {
                    "enable": false,
                    "path": ""
                },
                "exposed": {

```



```
        "enable": false,
        "url":"",
        "url_spec":""
    }
},
"notifier":{
    "email": {
        "address": "",
        "password": "",
        "smtp":{
            "smtp_server": "",
            "smtp_port": "",
            "smtp_secure": ""
        }
    },
    "enable_notify":"false"
}
},
"wamp": {
    "url": "ws://localhost",
    "port": "8181",
    "ssl": "false",
    "realm": "s4t",
    "topic_connection": "board.connection",
    "crossbar_pub_ip": ""
},
"modules": {
    "plugins_manager": {
        "enabled": true
    },
    "services_manager": {
        "enabled": true,
        "wstun": {
            "port_range":{
                "high":40100,
                "low": 40001
            },
            "public_ip": ""
        }
    }
}
```

```
        }
    },
    "nodered_manager": {
        "enabled": true
    },
    "vnets_manager": {
        "enabled": false,
        "socat": {
            "ip": "20.0.0.0",
            "server": {
                "port": "10000"
            }
        }
    },
    "gpio_manager": {
        "enabled": false
    },
    "drivers_manager": {
        "enabled": false
    },
    "vfs_manager": {
        "enabled": false
    }
}
}
```

API documentation

La documentazione sulle API di Iotronic è consultabile online al seguente indirizzo:
<https://iotronic-dev.smartme.io/iotronic-api-docs/>



Lightning-rod

Lightning-rod (LR), nell'architettura di *Stack4Things*, è il probe in esecuzione sui device ed interagisce con Iotronic per la gestione del device stesso.

Iotronic comunica con LR mediante chiamate RPC sfruttando il protocollo WAMP. La gestione della comunicazione e la registrazione delle chiamate RPC di LR è demandata al server Crossbar.

Ogni modulo di LR espone un proprio set di RPC, richiamate dal rispettivo modulo di Iotronic.

Moduli Manager in Lightning-rod

In LR i Manager sono implementati come moduli JS a se stanti, abilitabili dal file di configurazione *settings.json*.

Ogni Manager di LR espone un proprio set di metodi RPC, richiamate dal rispettivo modulo Manager lato Iotronic.

I moduli che implementano un Manager di LR, per essere definiti tali devono implementare:

- un metodo *Init()*:
 - al collegamento di LR ad Iotronic, tale metodo regista sul server Crossbar le proprie funzioni RPC, che rappresentano le funzionalità esposte dal Manager.
 - tutti i Manager devono implementare tale metodo
 - nel file *settings.json*, per ogni modulo, vi è un flag “enabled” che consente il caricamento del modulo (registrazione RPC su Crossbar server)
- un metodo *Boot()*:
 - all'avvio di LR, se abilitato dal file *settings.json*, verranno eseguite delle procedure specifiche del Manager
 - non tutti i Manager devono implementare necessariamente tale metodo
 - nel file *settings.json*, per ogni modulo, vi è un flag “boot” che consente di abilitare l'esecuzione del metodo *Boot()*
- i metodi che implementano le funzioni RPC che espongono le funzionalità specifiche del modulo.
- eventuali metodi di servizio utilizzati all'interno del modulo stesso (privati).

Stati di funzionamento

LR nella sua ultima versione (v2.3.7) in combinazione con la versione di Iotronic 2.3.6 può avere i seguenti stati di funzionamento:

- **new:** è lo stato in cui si trova il device quando LR deve ancora effettuare la prima connessione ad Iotronic; alla prima connessione Iotronic muta lo stato del device in “registered”;
- **registered:** è lo stato di funzionamento standard di LR.

File di sistema

- LR configuration files:
 - (persistent confs) /etc/iotronic/
 - authentication.json
 - (variable confs) /var/lib/iotronic/
 - settings.json
 - <MANAGER_FOLDERS>
- LR source code path:
 - /usr/lib/node_modules/@mdslab/iotronic-lightning-rod/



authentication.json

In questo file sono contenute le configurazioni persistenti di Lightning-rod, come:

- identità del device
- credenziali di autenticazione ad Iotronic
- riferimenti al cloud (crossbar e wstun endpoint)

Template

```
{  
    "auth": {  
        "board": {  
            "code": "<DEVICE-ID>",  
            "layout": "[raspberry_pi | server | ... ]",  
            "authentication": {  
                "auth_lr_mode": "[ password | basic | certs (exp.) ]",  
                "password": "<IOTRONIC_DEVICE_PASSWORD>"  
            }  
        },  
        "wamp": {  
            "url_wamp": "<wss://URL_CROSSBAR_SERVER>",  
            "port_wamp": "8181",  
            "realm": "s4t",  
            "wamp_socket_recovery": "true",  
            "rpc_alive_time": 600,  
            "check_skt_time": 30  
        },  
        "wstun": {  
            "ws_url": "<wss://URL_WSTUN_SERVER>",  
            "ws_port": "8080",  
            "bin": "/usr/lib/node_modules/@mdslab/wstun/bin/wstun.js"  
        }  
    }  
}
```



settings.json

In questo file sono contenute le configurazioni variabili di Lightning-rod, come:

- device info:
 - label: nome del device
 - position: coordinate geografiche
 - health: porta rest server di diagnostica (descritto nella sezione successiva)
- modules: configurazioni dei vari Manager; ogni Manager avrà dei settaggi:
 - comuni:
 - enabled:
abilita l'esecuzione alla connessione di LR ad Iotronic della funzione *Init()* che espone le funzionalità del Manager.
 - boot:
abilita l'esecuzione allo start di LR della funzione di *Boot()* presente in ogni Manager.
 - specifici del Manager
- log: configurazioni del logger di LR
 - path del log file
 - log level
- extra: JSON in cui si ha la possibilità di specificare informazioni specifiche dello scenario.



Template

```
{
```

```
  "config":{  
    "log": {  
      "logfile": "/var/log/iotronic/lightning-rod.log",  
      "loglevel": "info"  
    },  
    "extra":{},  
    "board": {  
      "label": "",  
      "position":{  
        "altitude": "",  
        "longitude": "",  
        "latitude": "",  
        "timestamp": ""  
      },  
      "health":{  
        "local_port": "8888"  
      },  
      "modules": {  
        "plugins_manager": {  
          "enabled": true,  
          "boot": true,  
          "alive_timer": 60  
        },  
        "services_manager": {  
          "enabled": true,  
          "boot": false  
        },  
        ...  
      }  
    }  
  }  
}
```

API exposed

LR espone inoltre un set di API RESTful richiamabili localmente sul device (*port: 8888*) per consentire al manutentore dell'infrastruttura, che si trova in loco per un intervento, di valutare il corretto funzionamento di LR:

```
curl localhost:8888/diagnostics
```

che restituisce con un oggetto JSON simile al seguente:

```
{
    "lr_pid": <LR-PID>,
    "internet_connection": {
        "status": "<true | false>",
        "reason": <possible warn/err message>
    },
    "wamp_connection": {
        "status": "<true | false>",
        "reason": {
            "message": "<IOTRONIC STATUS MESSAGE>",
            "result": "<SUCCESS | ERROR>"
        }
    },
    "board_id": "<DEVICE-ID>",
    "timestamp": "<UTC-TIMESTAMP>"
}
```



Iotronic-UI

La dashboard di amministrazione di Iotronic consente all'amministratore di sistema di gestire le flotte di device e monitorarne lo stato.

The screenshot shows the Iotronic-UI dashboard interface. At the top, there's a header bar with the Stack4Things logo and a 'Support' link. Below the header is a sidebar on the left containing a dropdown menu set to 'ALL' and a list of 'Boards'. The main area features a map of the Alpine region, specifically the French Alps and the Ligurian coast, with several green location pins indicating device locations. On the right side of the map, there are two blue buttons labeled 'Services' and 'Plugin' with corresponding icons.

Board	Status
alanno_top	Connesso
bussoleno	Connesso
dora_pulk	Connesso
gad	Connesso
geve	Connesso
lodo	Connesso
perilleux	Connesso
pratorosso	Connesso
providenza_dam	Connesso
ss335_d	Connesso
ss335_s	Connesso
tana	Connesso
valbona_molinazzo	Connesso
alanno_bot	Connesso
alveo_vecchio	Connesso
borio_1	Connesso
borio_2	Connesso
datalogger1	Connesso
petino	Connesso
polcevera	Connesso
prova_test_agrare	Connesso
taranco	Connesso

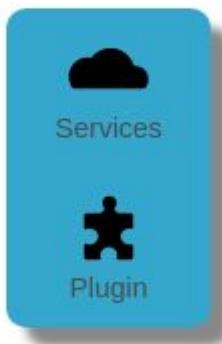
La dashboard nello specifico è organizzata come segue.

Nella sezione a sinistra della mappa sono elencati i device raggruppati per flotte/progetti. In ogni flotta i device sono ordinati in funzione del loro stato di connessione e funzionamento:

This screenshot provides a detailed look at the device list and its connection status. The list is filtered for the 'Admin' user. It shows 5 connected devices (green icons) and 14 devices in an attention state (yellow icons). A legend on the right defines these states: a green circle for 'Connesso' (Connected), a yellow circle for 'in stato di attenzione' (Attention State), and a red circle for 'Disconnesso' (Disconnected).

Device	Status
dev_sme1	Connesso
dev_sme2	Connesso
dev_st1	Connesso
sme-arancino	in stato di attenzione
st-artik	Connesso
LEDERO	in stato di attenzione

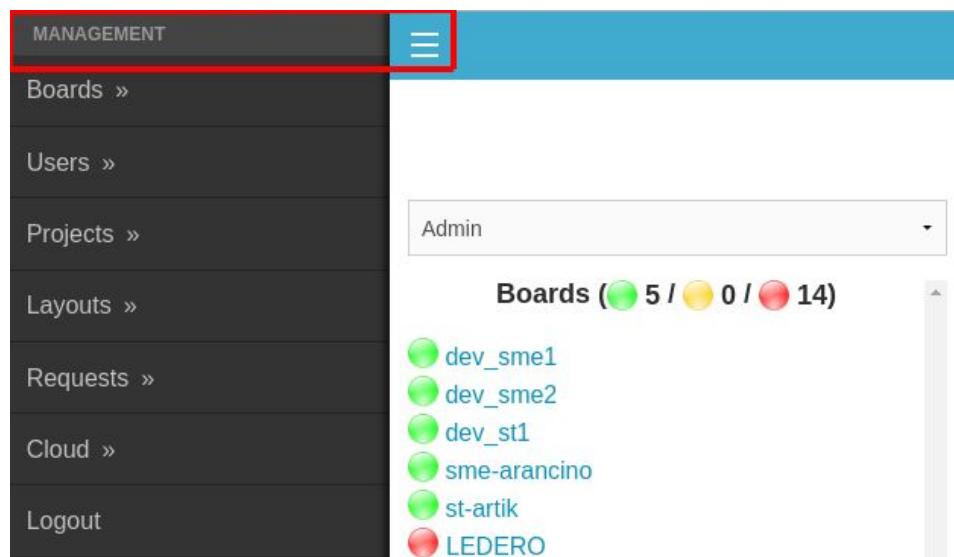
Lo stato di attenzione ☺ invece, viene stabilito mediante controlli automatici da parte della dashboard stessa; al momento tali controlli sono dipendenti dallo scenario e implementati in funzione delle esigenze specifiche. In futuro tali controlli saranno elaborati da un apposito Manager/modulo di Iotronic e saranno esposti mediante apposite API.



Nella icon-bar di destra sono accessibili i menù gestionali dei manager abilitati durante il deploy di Iotronic (es.: *Service Manager*, *Plugin Manager*).

Tramite le icone rappresentative dei *Manager* è possibile accedere ai menù che espongono le funzionalità dello stesso.

La gestione dei device e di Iotronic avviene invece mediante il menù principale  :



The screenshot shows the Iotronic management interface. On the left is a dark sidebar with the following menu items:

- Boards »
- Users »
- Projects »
- Layouts »
- Requests »
- Cloud »
- Logout

To the right of the sidebar is a light-colored main area. At the top right of the main area is a dropdown menu set to "Admin". Below it is a section titled "Boards" with a status indicator:  5 /  0 /  14. A scrollable list of device icons and names follows:

-  dev_sme1
-  dev_sme2
-  dev_st1
-  sme-arancino
-  st-artik
-  LEDERO

Da ogni sottomenù è possibile accedere alle diverse funzionalità di sistema:

- Boards - accedere alle funzionalità del Device Manager.
- Users - gestire gli utenti del sistema:
 - l'utente “admin”
 - gli utenti gestori/referenti associabili ad ogni device
- Projects - gestire i progetti/lotte in Iotronic in cui raggruppare i device.
- Layouts - gestire i layout (Hw - Sw) dei device gestiti da Iotronic.
- Requests - visualizzare i risultati delle azioni svolte sui device e le flotte.
- Cloud - collegamenti ai servizi esterni ad Iotronic (Log manager, ecc)



Le funzionalità del *Device manager* esposte dal sopracitato menù “Boards”, sono le seguenti:

« BACK
BOARDS
List
Register
Inject configuration
Update Info
Unregister
Actions on board
Package Manager
Update Lightning-rod

- **List:**
lista dei device registrati in Iotronic sui vari progetti/flotte
- **Register:**
accesso al form di registrazione di un nuovo device
- **Inject configuration:**
panel per eseguire l'inject del file di configurazione di LR su uno o più device.
- **Update Info:**
form di aggiornamento delle informazioni specificate in fase di registrazione.
- **Unregister:**
cancellazione da Iotronic di un device precedentemente registrato.
- **Action on board:**
panel per eseguire azioni su uno o più device.
- **Package Manager:**
panel per la gestione delle operazioni sui pacchetti software del device (installazioni, update, ecc)
- **Update Lightning-rod:**
panel per eseguire l'aggiornamento di LR da remoto su uno o più device.

Le funzionalità esposte dal *Device Manager* e dagli altri *Manager* di Iotronic, saranno descritte nelle sezioni successive.

WSTUN

WSTUN è un servizio sviluppato all'interno dei laboratori del gruppo di ricerca *MDSLab* (*Università di Messina*), in collaborazione con *Smartme.IO*, che consente la creazione di reverse tunnel basati su websocket.

WSTUN è utilizzato in Iotronic dal *Service Manager* per l'esposizione di servizi in esecuzione sui device, in maniera sicura attraverso la tecnologia websocket.

Tale servizio nello specifico si compone di due parti:

- client, istanziato sui device, che permette di esporre tramite il cloud i servizi in esecuzione sui device;
- server, in ascolto sulle seguenti porte:
 - 8080, porta principale su cui è in ascolto il servizio e a cui le istanze client si collegano;
 - range di porte (scelte in fase di deploy di Iotronic); Iotronic attinge random da tale pool, per destinare una delle porte disponibili all'esposizione di un servizio in esecuzione sul device.

Riferimenti: <https://github.com/MDSLab/wstun>



Platform administration

In tale sezione verranno descritti i vari manager che si occupano della gestione delle funzionalità di amministrazione di lotronic:

- Users management:
 - operazioni CRUD sugli utenti a cui vengono associati i device ([API](#))
- Projects/Fleets management:
 - operazioni CRUD sui progetti/flotte di dispositivi ([API](#))
- Layouts management:
 - operazioni CRUD sui layout dei dispositivi ([API](#))
- Requests management:
 - visualizzazione dei risultati delle richieste/operazioni eseguite sui device e/o le flotte di device ([API](#))

Users management

Allo stato attuale la gestione degli utenti prevede la presenza di solo un utente di tipo admin predisposto in fase di deploy dell'istanza di lotronic.

Gli utenti che possono essere creati all'interno di lotronic (escluso l'admin pocanzi descritto) sono semplicemente riferimenti ai responsabili/referenti/proprietari del device. Non hanno ancora, ad oggi, alcuna possibilità di effettuare il login né, più in generale, di richiamare le API esposte da lotronic. L'utente associato al device verrà notificato per email automaticamente da lotronic, in caso di prolungata disconnessione del suo device.

Add new User to the Cloud

Username

Password

Verify Password

Email (example: paolo.rossi@gmail.com)

Valid email!

First Name

Last Name

Output

lotronic mette a disposizione le funzionalità di Update, Delete e List per gli utenti precedentemente creati.

API di riferimento:

- **POST** [/v1/users/](#)



Projects/Fleets management

All'interno di iotronic ogni device è associato ad uno ed un solo progetto/flotte. Non è possibile allo stato attuale che un device possa appartenere a più progetti/flotte. Con il termine "progetto" o "flotta" ci riferiamo al concetto di "raggruppamento di device" che condividono uno scenario di utilizzo comune.

A screenshot of the iotronic Admin interface. On the left, there is a vertical sidebar with a dropdown menu labeled "Admin" which is currently set to "Admin". Below this, there are four options: "ENEL", "ST - Lift", and "ALL", each preceded by a small green circular icon. To the right of the sidebar, there is a list of three devices: "dev_sme2", "dev_st1", and "sme-arancino", each also preceded by a small green circular icon.

Dal menù “Projects” di amministrazione possiamo effettuare le operazioni CRUD sui progetti/flotte di iotronic per lo specifico scenario.

Per la creazione di un nuovo progetto/flotta dovremo specificare nel form “Create”:

1. il nome del progetto/flotta
2. una descrizione

A screenshot of a modal dialog titled "Add new Project to the Cloud". The dialog contains three input fields: "Name" (with placeholder "Project Name"), "Description" (with placeholder "Description"), and "Output" (an empty text area). At the bottom right of the dialog is a blue "Register" button.

Come per l'utenza di "admin", in fase di deploy di iotronic verrà creato anche il progetto/flotta "Admin".

API di riferimento:

- **POST** [/v1/projects/](#)

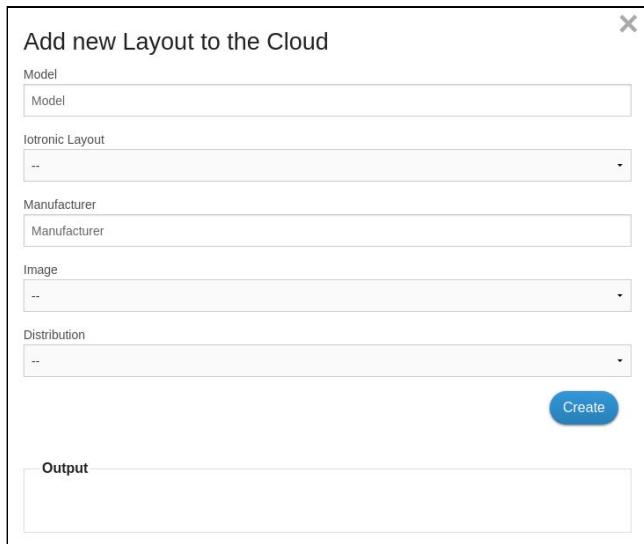
Layouts management

In Iotronic il concetto di “layout” identifica una configurazione/modello in cui viene specificata sia l’architettura hardware del device IoT, che l’immagine di sistema operativo (OS) caricata sul device.

Dal menù “Layout” di amministrazione possiamo effettuare le operazioni CRUD sui layout di Iotronic per lo specifico scenario.

Per la creazione di un nuovo layout dovremo specificare, nel form “Create” i seguenti campi:

1. il nome del nuovo modello/configurazione di layout nel campo “Model”
2. il layout hardware del compute module del device IoT nella lista di quelli supportati da Iotronic:
 - raspberry_pi
 - artik
 - arduino_yun
 - server (generic)
3. l’immagine di OS caricata sul device (es: Raspbian, Ubuntu, ecc)
4. la famiglia a cui appartiene la distribuzione del OS sopra specificato, tra quelli supportati in Iotronic:
 - debian
 - openwrt



The screenshot shows a modal dialog titled "Add new Layout to the Cloud". It contains the following fields:

- Model:** A text input field with the placeholder "Model".
- Iotronic Layout:** A dropdown menu showing two options: "--" and "raspberry_pi".
- Manufacturer:** A text input field with the placeholder "Manufacturer".
- Image:** A dropdown menu showing two options: "--" and "raspbian".
- Distribution:** A dropdown menu showing two options: "--" and "debian".
- Output:** A large text area for notes.

A blue "Create" button is located at the bottom right of the dialog.

API di riferimento:

- **POST** [/v1/layouts/](#)

Requests management

Le azioni che Iotronic esegue sui device IoT possono avere diverse modalità di esecuzione:

- “*syscall*”: azioni eseguite sul device che ritornano prontamente un risultato.
- “*batch*”: azioni eseguite su tutti i device di un progetto/fлотта
- “*long-running*”: azioni eseguite sui device e la cui esecuzione richiede un tempo di attesa non ben determinato.

Il concetto di “*request*” in Iotronic entra in gioco per queste ultime due modalità (“*batch*” e “*long-running*”). Entrambe le modalità hanno in comune un tempo di attesa potenzialmente lungo prima di restituire un risultato; nel caso di chiamate di tipo:

- “*batch*”, il tempo di completamento di un’azione eseguita su decine o centinaia di device di un progetto/fлотта, richiederà sicuramente un tempo non determinabile a priori;
es.: inject di un file (un plugin) su tutti i device di una flotta.
- “*long-running*”, si prevede che il tempo di completamento dell’azione richiederà un tempo non determinabile a priori;
es.: installazione di un pacchetto software in un device.

In questi due casi Iotronic, dopo aver lanciato la richiesta di esecuzione di un’azione, rilascerà all’utente amministratore (tramite dashboard o API) un “*request-id*”.

Requests				
Show	10	Search:		
entries				
	id_request	subject	timestamp	result
	ab091221-cee7-43f6-aa98-bd8bfb6f56b8	plugins configuration inject	2019-10-14T15:50:03.000Z	pending
	f3babe91-ff65-4305-b31a-d5a038f6ad92	plugins configuration inject	2019-10-08T12:39:01.000Z	pending
	25cbba33-84e5-4509-af20-1288f36a405f	plugins configuration inject	2019-10-08T12:38:06.000Z	pending
	21cbbffa-6967-4bce-8b93-1f5c29c7025a	plugins configuration inject	2019-10-08T12:35:16.000Z	pending
	3b3fe480-8dc6-483b-a0d6-c4b78f5757db	plugins configuration inject	2019-10-08T12:34:26.000Z	pending
	35190430-2318-4bad-84e5-db0a0c0bbfcf	plugin inject: 127	2019-10-04T14:02:29.000Z	completed
	e484d2d7-b4a8-46c4-9d32-0e262c0e78bf	plugin inject: 127	2019-10-04T14:02:06.000Z	completed
	a8128636-75a9-4f81-b372-bcb5fb169be3	plugin inject: 127	2019-10-04T13:59:26.000Z	completed
	e4630d69-2819-4ba9-95b1-80245fbf0839	install package vim	2018-10-29T14:44:23.000Z	completed
	24aabce8-3cfa-46e5-8947-23968914452f	LR update	2018-10-29T14:34:10.000Z	completed

Showing 1 to 10 of 76 entries

Previous 1 2 3 4 5 ... 8 Next

Tale ID verrà utilizzato successivamente sia per identificare la richiesta effettuata,

sia per visualizzare:

- i risultati dell'azione per ogni device della flotta;

Request

Request ID: 21cbbbfa-6967-4bce-8b93-1f5c29c7025a

Subject: plugins configuration inject

Show: 10 | Search:

entries		
board_id	timestamp	result
linkit	2019-10-08T12:35:16.000Z	WARNING
yun-22	2019-10-08T12:35:16.000Z	WARNING
st-dev-smartme	2019-10-08T12:35:16.000Z	pending
laptop-14	2019-10-08T12:35:16.000Z	SUCCESS
rasp-nicola	2019-10-08T12:35:16.000Z	WARNING
smartme-messina-old	2019-10-08T12:35:16.000Z	WARNING

Showing 1 to 6 of 6 entries

Previous 1 Next

Previous

- il risultato dell'azione long-running sul device specificato.

Request

Request ID: e4630d69-2819-4ba9-95b1-80245fbf0839

Subject: install package vim

Show: 10 | Search:

entries		
board_id	timestamp	result
laptop-14	2018-10-29T14:44:23.000Z	SUCCESS

Showing 1 to 1 of 1 entries

Previous 1 Next

Previous

Per ogni device verrà riportato l'esito con relativo timestamp; ad ogni esito è associato il messaggio/log dell'azione eseguita sul device.



Message for board st-dev-smartme

Result: SUCCESS

```
Downloading http://download.arancino.cc/distro/LEDE/releases/17.01.3-1-glibc/targets/brcm2708/bcm2710/packages/Packages.gz
Updated list of available packages in /var/opkg-lists/reboot_core
Downloading http://download.arancino.cc/distro/LEDE/releases/17.01.3-1-glibc/targets/brcm2708/bcm2710/packages/Packages.sig
Signature check passed.
Installing nano (2.9.6-1) to root...
Downloading http://download.arancino.cc/distro/LEDE/releases/17.01.3-1-glibc/targets/brcm2708/bcm2710/packages/nano_2.9.6-1_arm_cortex-a53_neon-vfpv4.ipk
Configuring nano.
```

[Previous](#)

API di riferimento:

- **GET** [`/v1/requests/`](#) get IoTronic requests
- **DELETE** [`/v1/requests/`](#) delete all IoTronic requests
- **GET** [`/v1/requests/{request}`](#) get request results
- **DELETE** [`/v1/requests/{request}`](#) delete IoTronic request
- **GET** [`/v1/projects/{project}/requests`](#) get project's requests
- **DELETE** [`/v1/projects/{project}/requests`](#) delete project's requests

Device Manager

Il *Device Manager* si occupa di gestire a vario livello le flotte di device registrati in Iotronic. Rappresenta il manager principale di Iotronic, mentre gli altri *Manager*, che descriveremo successivamente, sono opzionali e attivabili in funzione dello scenario d'utilizzo di Iotronic in fase di installazione/configurazione o anche successivamente.

Le funzionalità principali sono elencate qui di seguito e descritte nelle relative sezioni di questo documento:

- Gestione identità del device in Iotronic
- Esecuzione di azioni predefinite sui device
- Gestione pacchetti software del device
- Aggiornamento di Lightning-rod

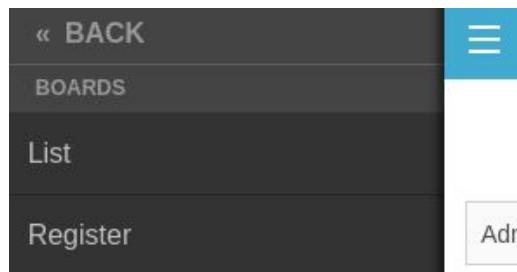
Gestione identità del device in Iotronic

Iotronic gestisce le seguenti fasi:

- registrazione di un nuovo device
- rimozione di un device
- aggiornamento delle informazioni specificate in fase di registrazione
- inject di un nuovo set di configurazioni sul device
- visualizzazione della lista dei device di ogni progetto/flotta

Registrazione di un device in Iotronic

Per collegare un nuovo device IoT e poterlo gestire da remoto per mezzo di Iotronic è necessario preventivamente registrarla sulla piattaforma.





Navigando nel menù di sistema, nella sezione “*Boards*” sarà possibile raggiungere il panel di registrazione “*Register*”.

Add new Board to the Cloud

Board UUID

 New

Label

Password (between 4 and 60 digits)

 Show

Description

Connectivity

Ethernet MAC

Layout

--

Latitude	Longitude	Altitude
e.g.: 38.123	e.g.: 15.123	e.g.: 50.123

Nel form di registrazione sarà necessario specificare:

- un device ID (“*Board UUID*” in formato UUID)
- una label
- una password per l’accesso ad Iotronic
- una descrizione
- (opzionale) informazioni sul tipo di connettività utilizzata dal device per connettersi ad Iotronic:
 - Ethernet (specifica del MAC)
 - WiFi (specifica del MAC)
 - Mobile (specifica del codice ICCID della SIM)
- il layout riconducibile al device (piattaforma HW e SW); es:
 - Raspberry Pi 3 - Raspbian
 - Raspberry Pi 3 - Ubuntu 64
 - Arancino - Arancino OS
 - Arancino - Raspbian

- le coordinate geografiche

Extra user defined data (json)

Nessun file selezionato

Project	User
--	--

Enable email notification (connection status)

Output

- extra info (JSON):
 - informazioni supplementari specifiche di uno scenario.
- il progetto/flotta a cui afferirà il nuovo device
- l'utente, il referente/responsabile della gestione del device
- (opzionale) abilitare il sistema di notifiche (se abilitato in lotronic) tramite email che avvertirà l'utente referente/responsabile del device in caso di disconnessione prolungata del suddetto.

API di riferimento:

- **POST** [/v1/boards/](#)



Rimozione di un device da Iotronic

La rimozione di un device da Iotronic è possibile effettuarla dal panel “*Unregister*” raggiungibile dal menù di sistema, nella sezione “*Boards*”.

Unregister Board

Enable / Disable per project delete

Show Search:

label	board_id	status
lede-mctx	lede-mctx	D
lede-test-00	lede-test-00	D
rosa4	rosa4	D
sme-arancino	5b04a27e-7b86-7e3d-2c63-bd7ee6288b16	C
sme-test-raspbian	70065bc7-000a-8608-8aaa-5f9ea49c793f	D
sme-test-ro	ad6167b8-9a6e-f436-203f-58b18d64c570	D
sme-ubuntu64	ffaa2c26-8a17-4d41-c837-61b9be3abd11	D
st-artik	af01f0fe-dc15-1a08-d27f-3602f6419744	C
st-dev-smartme	473a3576-ef4b-ab91-51b5-ba3830dc4e6e	D

Showing 11 to 19 of 19 entries Previous 1 Next

[Remove](#)

Sarà possibile rimuovere, uno o più device, oppure tutti i device di un intero progetto/flotta.

API di riferimento:

- **DELETE** </v1/boards/{board}>
- **DELETE** </v1/projects/{project}/boards>

Aggiornamento delle informazioni di un device

L'aggiornamento delle informazioni di un device da lotronic è possibile effettuarla dal panel “*Update Info*” raggiungibile dal menù di sistema, nella sezione “*Boards*”.

Come primo step sarà necessario selezionare il device da aggiornare:



Successivamente, verrà visualizzato il form di aggiornamento delle informazioni, in cui verranno precaricate le informazioni già presenti.



Una volta completato l'aggiornamento delle informazioni, queste, se il device dovesse risultare connesso, verranno riportate anche sul device. Se il device in questa fase dovesse invece risultare disconnesso, sarà compito dell'amministratore, alla riconnessione, iniettare



la nuova configurazione dall'apposito panel “*Inject configuration*”, descritta nella sezione successiva.

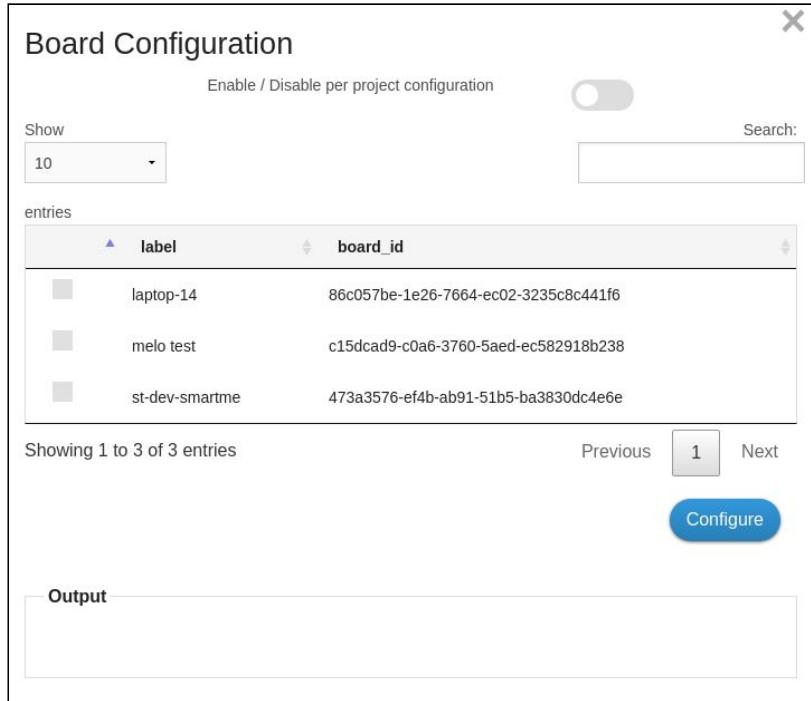
N.B.: se il campo password non verrà modificato, la password del device rimarrà inalterata.

API di riferimento:

- **PATCH** [/v1/boards/{board}](#)

Inject di una nuova configurazione nel device

Iotronic permette di iniettare sul device una nuova configurazione (parametri di connessione esclusi) che andrà a sovrascrivere il file settings.json. Tale azione è possibile effettuarla dal panel “Inject configuration” raggiungibile dal menù di sistema, nella sezione “Boards”.



The screenshot shows the "Board Configuration" dialog box. At the top, there is a toggle switch labeled "Enable / Disable per project configuration". Below it, a "Show" dropdown is set to 10, and a "Search:" input field is empty. The main area is titled "entries" and contains a table with three rows:

label	board_id
laptop-14	86c057be-1e26-7664-ec02-3235c8c441f6
melo test	c15dcad9-c0a6-3760-5aed-ec582918b238
st-dev-smartme	473a3576-ef4b-ab91-51b5-ba3830dc4e6e

Below the table, it says "Showing 1 to 3 of 3 entries". There are "Previous" and "Next" buttons, and a page number "1". A blue "Configure" button is located to the right of the page number. At the bottom, there is a section labeled "Output" with a large empty text area.

L'inject della configurazione all'interno del device comporta un restart di Lightning-rod che riavviandosi carica la nuova configurazione.

Tale operazione sarà possibile effettuarla su uno o più device allo stesso tempo o su tutti i device di un progetto/flotta.

API di riferimento:

- **PUT** </v1/boards/{board}/conf>
- **PUT** </v1/projects/{project}/conf>



Lista device in lotronic

Tramite il panel “*List*” raggiungibile dal menù di sistema, nella sezione “*Boards*” è possibile visionare la lista dei device registrati in lotronic per i vari progetti/flotte.

BoardsX

Show

10

Search:

entries

label	status	latest_update	notify	board_id
rosa4	D	2019-07-17T09:42:39.000Z	false	rosa4
Santa Lucia Logger	C	2019-10-30T01:10:12.000Z	false	slucia-loggo
smartme-messina	D	2019-07-18T14:25:41.000Z	true	smartme-messina-ba3830dc4e6e
sme-arancino	C	2019-10-25T11:16:24.000Z	true	5b04a27e-7b86-7e3d-2c63-bd7ee6288b16
sme-test-raspbian	D	2019-04-09T09:17:27.000Z	true	70065bc7-000a-8608-8aaa-5f9ea49c793f
sme-test-ro	D	2019-05-29T10:50:20.000Z	false	ad6167b8-9a6e-f436-203f-58b18d64c570
sme-ubuntu64	D	2019-09-06T11:15:21.000Z	false	ffaa2c26-8a17-4d41-c837-61b9be3abd11
st-artik	C	2019-10-30T03:31:10.000Z	true	af01f0fe-dc15-1a08-d27f-3602f6419744
st-dev-mimmo	D	2019-04-30T15:21:50.000Z	false	1181fd23-958c-4669-397e-4fd9e3c43ab1
st-dev-smartme	D	2019-10-04T14:00:37.000Z	false	473a3576-ef4b-ab91-51b5-ba3830dc4e6e

Showing 21 to 30 of 30 entriesPrevious123Next

API di riferimento:

- [GET /v1/boards/](#)

Esecuzione di azioni predefinite sui device

Iotronic gestisce un set di azioni predefinite che è possibile eseguire su:

- un singolo device
- device multipli
- intere flotte di device (device afferenti ad un progetto)

Nello specifico l'attuale versione di Iotronic, mette a disposizione le seguenti azioni:

- **ping di un device “hostname”**: tale azione andrà a contattare il device che risponderà con una stringa

<label-device>@<timestamp>

- **restart di Lightning-rod**
- **reboot del device**

Board Action

Action

reboot

Enable / Disable per project action

Show

10

Search:

label	board_id
laptop-14	86c057be-1e26-7664-ec02-3235c8c441f6
test	c15dcad9-c0a6-3760-5aed-ec582918b238

Showing 1 to 2 of 2 entries

Previous
1
Next

Delay Time

(in seconds)

Execute

Output

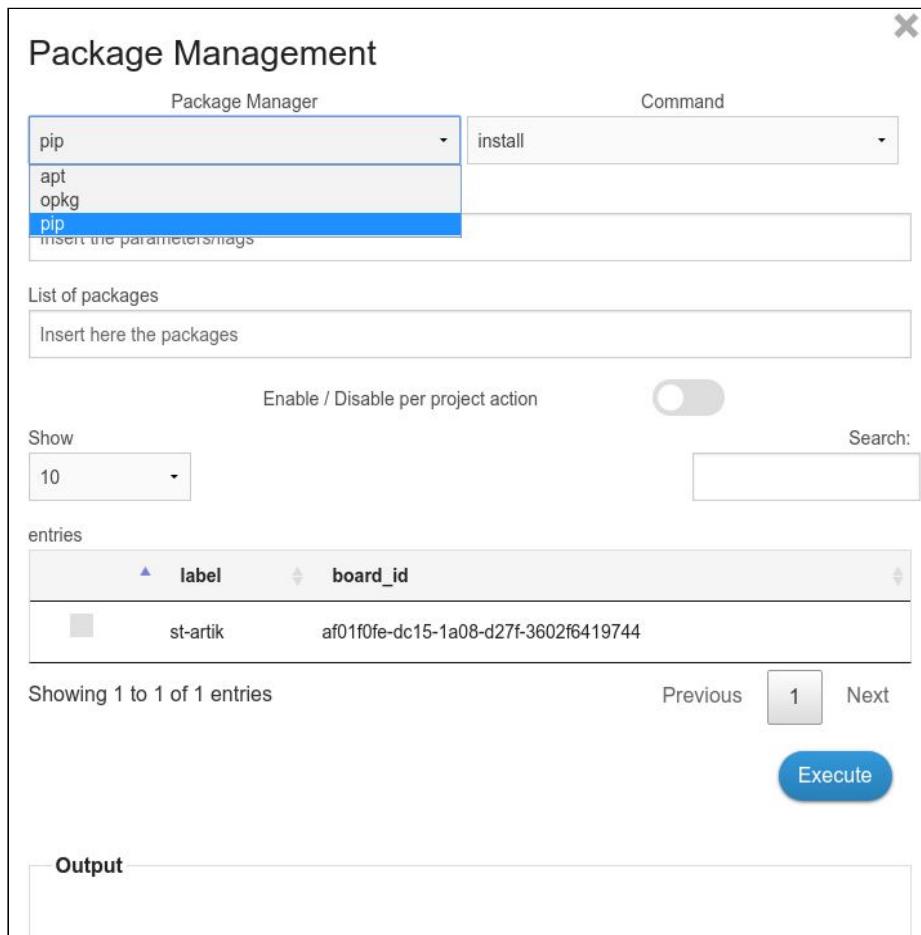
API di riferimento:

- **POST** </v1/boards/{board}/action>
- **POST** </v1/projects/{project}/action>

Gestione pacchetti software del device

Iotronic consente di installare, aggiornare e rimuovere pacchetti software sui device, interagendo con i più comuni package manager delle distribuzioni di SO attualmente supportate da Iotronic. Nello specifico i package manager gestiti dall'attuale versione di Iotronic sono:

- apt
- opkg
- pip/3
- npm (coming soon)



Package Management

Package Manager

Command

pip

apt
opkg
pip

insert the parameters

List of packages

Insert here the packages

Enable / Disable per project action

Show: 10

Search:

entries

label	board_id
st-artik	af01f0fe-dc15-1a08-d27f-3602f6419744

Showing 1 to 1 of 1 entries

Previous 1 Next

Execute

Output

API di riferimento:

- **POST** </v1/boards/{board}/package>
- **POST** </v1/projects/{project}/package>

Aggiornamento di Lightning-rod

Da Iotronic è possibile effettuare da remoto l'aggiornamento di Lightning-rod (LR). Ogni aggiornamento comporta un restart dell'istanza di LR in esecuzione sul device. Al riavvio LR comunicherà ad Iotronic la versione durante la procedura di login, in modo che Iotronic possa tenerne traccia.

Lightning-rod Management

Enable / Disable per project action

Search:

Show entries

label	board_id	distro	lr_version
0123181022007	0123181022007	openwrt	2.3.3
kitra-dev-mimmo	bc1d3f61-5aab-0c0f-9a67-02e17792b84c	debian	2.3.3
lede-p0	lede-p0	openwrt	2.3.3
smartme-messina	smartme-messina-ba3830dc4e6e	openwrt	2.3.3
st-dev-mimmo	1181fd23-958c-4669-397e-4fd9e3c43ab1	openwrt	2.3.0
st-dev-smartme	473a3576-ef4b-ab91-51b5-ba3830dc4e6e	openwrt	2.3.3

Showing 1 to 6 of 6 entries

Previous
1
Next

Version (only if debian)

Update

API di riferimento:

- **POST** </v1/boards/{board}/lr>
- **POST** </v1/projects/{project}/lr>



Procedure operative

Primo collegamento del device ad Iotronic

Una volta che il device è stato registrato in Iotronic, sarà necessario configurarlo per eseguire la sua prima connessione ad Iotronic.

Nel file *authentication.json* (descritto nelle sezioni precedenti) sarà necessario compilare i seguenti campi:

1. **auth -> board:**
 - a. **code:** il device UUID rilasciato in fase di registrazione
 - b. **authentication**
 - i. assicurarsi che il metodo di autenticazione sia lo stesso utilizzato in Iotronic; nel caso di “*password*” (metodo di default), sarà necessario specificarla nell’omonimo campo.
2. **wamp:**
 - a. **url_wamp:** indirizzo websocket del server Crossbar associato all’istanza di Iotronic a cui si è registrato il device; dovrà essere rispettato il seguente formato dell'url websocket:
 - i. `wss://< IP-FQDN-CROSSBAR-SERVER >` in caso di connessione sicura
 - ii. `ws://< IP-FQDN-CROSSBAR-SERVER >` in caso di connessione non protetta
 - b. **port_wamp:** porta in cui è in ascolto il server Crossbar (di default 8181);
3. **wstun**

nella configurazione di default Iotronic e LR prevedono che il *Service Manager* sia abilitato e data la sua importanza nel management dei device la configurazione del servizio WSTUN è stata inclusa di default nel file di autenticazione

 - a. **ws_url:** indirizzo websocket dell’istanza server di WSTUN; il formato dell'url websocket è il seguente:
 - i. `wss://< IP-FQDN-CROSSBAR-SERVER >` in caso di connessione sicura
 - ii. `ws://< IP-FQDN-CROSSBAR-SERVER >` in caso di connessione non protetta
 - b. **ws_port:** porta in cui è in ascolto il server WSTUN (di default 8080).

Backup/restore configurazione di un device

Lightning-rod mette a disposizione un tool per il backup e successivo restore della sua configurazione; tale backup comprende:

- il file `/etc/iotronic/authentication.json`
- le configurazioni in `/var/lib/iotronic/` comprendenti:
 - il file `settings.json`
 - le configurazioni dei vari Manager (es. i plugin iniettati nel device tramite il *Plugin Manager*)

Tale tool, denominato “*board_bkp_rest*”, è eseguibile via shell per effettuare:

- il backup:
 - `board_bkp_rest backup`
 - tale comando produce un file compresso nella stessa locazione in cui viene eseguito, con il seguente formato di filename:
 - `bkp_<IOTRONIC-DEVICE-ID>_<TIMESTAMP>.tar.gz`
- il restore:
 - `board_bkp_rest restore <backup-filename-to-restore>`
 - dopo il restore sarà necessario effettuare il restart di LR

Service Manager

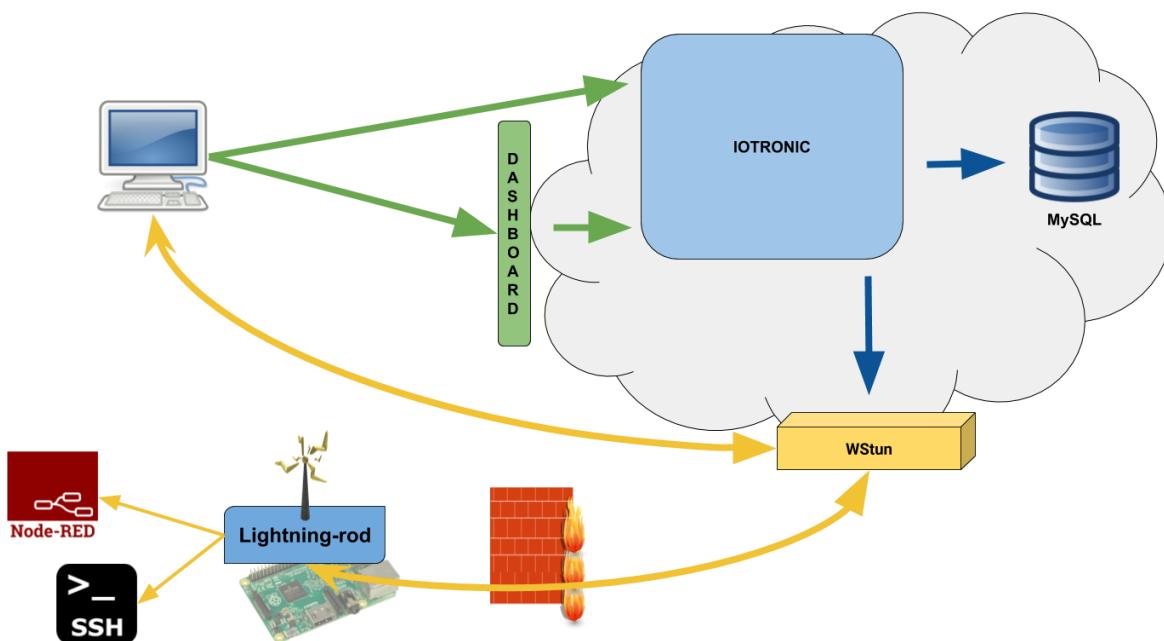
Il *Service Manager* in Iotronic permette ad un amministratore di esporre un servizio in running sul device (*ssh*, *http/s*, ecc), tramite la creazione di un tunnel sicuro basato sul protocollo websocket (*wss*).

La creazione del tunnel viene resa possibile tramite l'utilizzo del servizio **WSTUN** (<https://www.npmjs.com/package/@mdslab/wstun>).

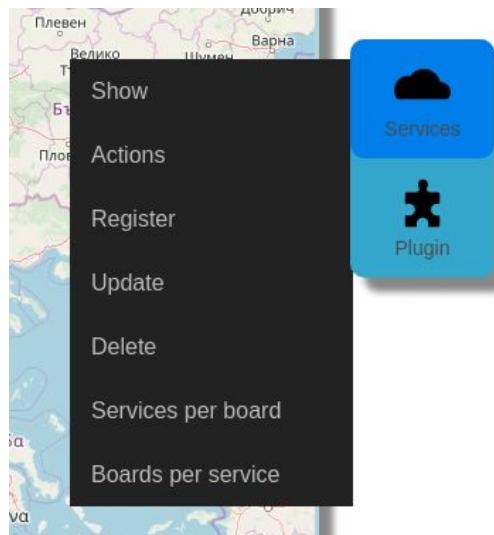
Tale servizio si compone di una parte server-side e di una client-side. L'istanza server è in esecuzione sul cloud (porta 8080), mentre quella client viene installata sui device insieme a LR.

Durante l'installazione di Iotronic dovranno essere dedicate a tale scopo un pool di porte esposte lato cloud (es.: *dalla 40001 alla 4000X*). Quando Iotronic riceve una richiesta di esposizione di un servizio di un device, Iotronic si occuperà di prelevare in maniera random una porta libera fra quelle specificate nel pool.

L'esposizione di un tunnel websocket consentirà di contattare un servizio in running su un device, sfruttando l'indirizzo pubblico del cloud e la suddetta porta generata random. Così facendo non sarà necessario né conoscere l'IP pubblico del device con cui si espone su Internet, né esporre lato device porte per raggiungere il servizio. L'unico vincolo che il device deve garantire è la possibilità di raggiungere il cloud dove risiede Iotronic.



Le funzionalità esposte dal Service Manager nella lotronic-UI sono le seguenti:



- *Show*:
mostra i servizi preimpostati esponibili da lotronic.
- *Actions*:
azioni sui tunnel che espongono i servizi dei device (enable, disable, restore).
- *Register | Update | Delete*:
azioni CRUD sul servizio da esporre
- *Services per board*:
visualizzazione dei servizi esposti da un device
- *Boards per service*:
visualizzazione dei device che espongono un dato servizio



Esposizione servizio

Dal menù “Actions” del *Service Manager* è possibile esporre un servizio in running su un device.

Service Actions

Service Name
SSH [SSH: 22]

Enable / Disable per project action

Show 10 Search:

label	board_id
dev_sme1	20R0030019052400001
dev_sme2	20R0030019052400002
dev_st1	20R0030019052900001
sme-arancino	5b04a27e-7b86-7e3d-2c63-bd7ee6288b16
st-artik	af01f0fe-dc15-1a08-d27f-3602f6419744

Showing 1 to 5 of 5 entries Previous 1 Next

Buttons: Enable, Disable, Restore

Dal questo panel si dovrà:

1. selezionare il servizio che si vuole esporre.

Service Name

SSH [SSH: 22]

2. scegliere se:

- a. esporlo su tutti i device di un progetto/flotta, abilitando il flag

Enable / Disable per project action

- b. esporlo su uno o più device selezionandoli dalla tabella

label	board_id
dev_sme1	20R0030019052400001
dev_sme2	20R0030019052400002
dev_st1	20R0030019052900001
sme-arancino	5b04a27e-7b86-7e3d-2c63-bd7ee6288b16
st-artik	af01f0fe-dc15-1a08-d27f-3602f6419744

Nella sezione output del panel verranno restituiti i riferimenti necessari per collegarsi al servizio mediante il tunnel appena esposto.

La lista dei tunnel esposti dal device è visualizzabile dal pannello “*Board Info*” raggiungibile cliccando sul nome del device nella sezione di sinistra della schermata principale di *iotronic-UI*.

Board Information: **laptop-14**

Services				
name	public_port	local_port	protocol	shortcut
SSH	7009	22	SSH	ssh -p 7009 root@192.168.17.247

API di riferimento:

- **POST** </v1/boards/{board}/services/{service}/action>
- **POST** </v1/projects/{project}/services/{service}/action>



Ripristino esposizione servizio

I tunnel esposti per mezzo di Iotronic, a causa di condizioni di rete instabili o anche dopo un non ben determinato numero di ore, può verificarsi che il tunnel si disconnetta.
In tale condizione, tramite il panel “Service Actions” dal menù del *Service Manager*, è possibile effettuarne il “Restore”, mantenendo lo stesso numero di porta assegnata in fase di abilitazione (vedi sezione precedente).

Service Actions

Service Name
SSH [SSH: 22]

Show: 10 | Search:

Enable / Disable per project action

label	board_id
dev_sme1	20R0030019052400001
dev_sme2	20R0030019052400002
dev_st1	20R0030019052900001
sme-arancino	5b04a27e-7b86-7e3d-2c63-bd7ee6288b16
st-artik	af01f0fe-dc15-1a08-d27f-3602f6419744

Showing 1 to 5 of 5 entries | Previous | **1** | Next

Enable | **Disable** | **Restore**

Selezionare:

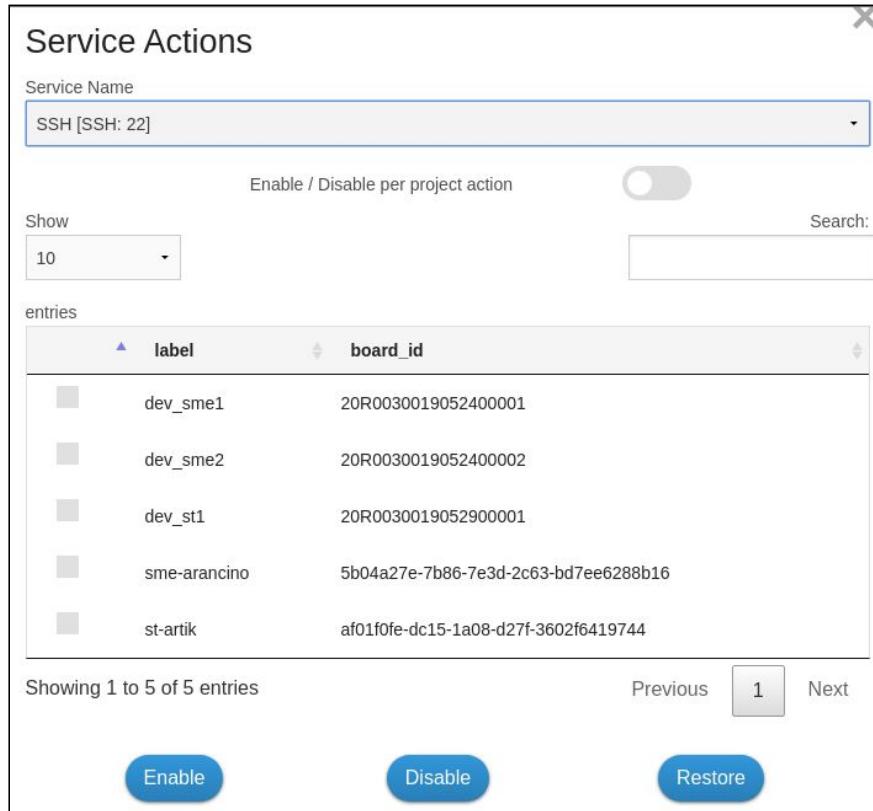
1. il servizio da restartare
2. uno o più device (o l'intera flotta/progetto)
3. “Restore”

API di riferimento:

- **POST** </v1/boards/{board}/services/{service}/action>
- **POST** </v1/projects/{project}/services/{service}/action>

Revoca esposizione servizio

Un servizio precedentemente esposto è possibile disabilitarlo dal menù “Actions” del Service Manager:



The screenshot shows the "Service Actions" interface. At the top, there is a "Service Name" dropdown set to "SSH [SSH: 22]". Below it is a toggle switch labeled "Enable / Disable per project action" which is currently off. A "Show" dropdown is set to "10" and a "Search" input field is empty. The main area is titled "entries" and contains a table with two columns: "label" and "board_id". The table lists five entries:

label	board_id
dev_sme1	20R0030019052400001
dev_sme2	20R0030019052400002
dev_st1	20R0030019052900001
sme-arancino	5b04a27e-7b86-7e3d-2c63-bd7ee6288b16
st-artik	af01f0fe-dc15-1a08-d27f-3602f6419744

At the bottom, there are three buttons: "Enable", "Disable", and "Restore". The "Disable" button is highlighted in blue.

Selezionare:

4. il servizio da disabilitare
5. uno o più device (o l'intera flotta/progetto)
6. “*Disable*”

API di riferimento:

- **POST** </v1/boards/{board}/services/{service}/action>
- **POST** </v1/projects/{project}/services/{service}/action>



Plugin Manager

Il *Plugin Manager* (PM) consente di iniettare e mettere in esecuzione, sui dispositivi, codice sorgente, che in questo contesto denominiamo *plugin*; i plugin possono essere sviluppati in linguaggio *NodeJS* e *Python*.

Il PM gestisce l'intero ciclo di vita dei plugin, dalla sua creazione alla sua gestione (start, stop, restart).

In Stack4Things distinguiamo due tipologie di plugin:

- **sincroni**: implementano al loro interno una funzione che restituisce un risultato dopo una breve computazione.
- **asincroni**: plugin che implementano algoritmi in long-running, la cui esecuzione termina solo quando viene eseguito il comando di arresto da lotronic.

Per ogni plugin si potranno specificare dei parametri d'ingresso che dovranno essere specificati in formato JSON.

Un plugin istanziato da Lightning-rod:

- se sviluppato in Python sarà messo in esecuzione mediante un *thread Python* istanziato da Lightning-rod;
- se sviluppato in NodeJS risulterà essere invece un sottoprocesso *Node* istanziato sempre da Lightning-rod.

Plugin workflow

Il Plugin Manager gestisce l'intero ciclo di vita di un plugin; di seguito in ordine logico di esecuzione le operazioni che vanno dalla creazione del plugin alla sua gestione all'interno dei dispositivi:

1. Creazione di un nuovo plugin
2. Caricamento (Upload) del plugin in lotronic
3. Revisione (Review) e Rilascio (Release) del plugin da parte dell'amministratore di sistema/lotronic
4. Iniezione (Inject) del plugin in un device
5. Start/Execute del plugin precedentemente iniettato
6. Stop del plugin (solo nei plugin asincroni)
7. Aggiornamento (Update) del plugin
8. Rimozione del plugin dal device
9. Rimozione del plugin da lotronic



Creazione di un nuovo plugin

I plugin in *Stack4Things* possono essere sviluppati in linguaggio *NodeJS* o *Python*. Nelle attuali versioni di lotronic e Lightning-rod, il plugin sarà sviluppato in un unico file (*.js* | *.py*) affiancato da un file di configurazione in formato JSON che conterrà i parametri d'ingresso con i quali verrà messo in esecuzione il plugin. Nella fase di creazione tali parametri d'ingresso rappresenteranno quelli di default; successivamente, nelle fasi di messa in esecuzione del plugin, sarà possibile specificare nuovi parametri d'ingresso.

Per aiutare lo sviluppo di plugin, attualmente viene fornito un tool di supporto per il *Type Python*, *PluginExec*, che verrà descritto nelle successive sezioni.

Plugin sincrono

Plugin utilizzato per eseguire sul device una porzione di codice che ritorna un risultato.

Sviluppo in Python

Il file Python in cui si articola un generico plugin sincrono deve prevedere la definizione di una funzione “main” che prenda in ingresso i seguenti parametri:

- **plugin_name:** <STRING> nome del plugin;
- **params:** <JSON> oggetto JSON contenente i parametri d'ingresso con cui il plugin verrà messo in esecuzione;
- **api:** <OBJECT> oggetto/libreria contenente utility per lo sviluppo dei plugin (vedi sezione Plugin API).

I parametri sopracitati saranno valorizzati esternamente da lotronic (*params*) e Lightning-rod (*plugin_name*, *api*). Di seguito viene riportato il template di un plugin sincrono *Python*.

```
# User imports
< SECTION TO SPECIFY MODULES IMPORTS >

def main(plugin_name, params, api):

    # SET PLUGIN LOGGER
    logging = api.getLogger(plugin_name)

    # -----
    # USER CODE
    input = params['< PARAMS JSON KEY >']

    ...

    result = < USER CODE RESULT >
    # -----

    return result
```



Nello specifico vi sarà:

- una sezione dedicata agli *import* delle librerie specifiche del plugin
- la funzione *main* sopracitata che contiene:
 - l'implementazione della logica del plugin
 - l'abilitazione del sistema di logging (opzionale)
 - il return del risultato
 - l'acquisizione/elaborazione dei parametri d'ingresso (*params*)

Esempio plugin Python sincrono:

```
# User imports
from datetime import datetime

#input_parameters -> {"name": "S4T"}


def main(plugin_name, params, api):

    logging = api.getLogger(plugin_name)

    # -----
    # USER CODE
    extra = api.getExtraInfo()
    logging.info("Board extra-info: " + str(extra))

    now = datetime.now().strftime( "%d %b %Y %H:%M:%S.%f" )

    result = { 'time':now, 'name':str(params['name']) }

    logging.info(result)
    # -----


    return result
```

Sviluppo in NodeJS

Il file NodeJS in cui si articola un generico plugin sincrono deve prevedere la definizione di una funzione “main” (`exports.main`) che prenda in ingresso i seguenti parametri:

- **plugin_name:** <STRING> nome del plugin;
- **arguments:** <JSON> oggetto JSON contenente i parametri d'ingresso con cui il plugin verrà messo in esecuzione;
- **api:** <OBJECT> oggetto/libreria contenente utility per lo sviluppo dei plugin (vedi sezione *Plugin API*).
- **callback:** <OBJECT> NodeJS callback object per il ritorno del risultato.

I parametri sopracitati saranno valorizzati esternamente da lotronic (*params*) e Lightning-rod (*plugin_name*, *api*). Di seguito viene riportato il template di un plugin sincrono NodeJS.

```
exports.main = function (plugin_name, arguments, api, callback) {  
  
    var logger = api.getLogger(plugin_name, 'debug');  
  
    // -----  
    // USER CODE  
    var input = arguments.< ARGUMENTS JSON KEY >;  
  
    ...  
  
    result = < USER CODE RESULT >  
    // -----  
  
    callback("OK", result);  
};
```

Nello specifico vi sarà:

- la funzione *main* sopracitata che contiene:
 - eventuali *require* delle librerie specifiche del plugin
 - l'implementazione della logica del plugin
 - l'abilitazione del sistema di logging (opzionale)
 - la restituzione del risultato tramite *callback*
 - l'acquisizione/elaborazione dei parametri d'ingresso (*arguments*)



Esempio plugin NodeJS sincrono:

```
//input_parameters -> {"says": "Hello!"}

exports.main = function (plugin_name, arguments, api, callback) {

    var logger = api.getLogger(plugin_name, 'debug');

    // -----
    // USER CODE

    var says = arguments.says;
    logger.info(says);

    // -----

    callback("OK", says);

};
```

Plugin asincrono

Plugin utilizzato per mettere in esecuzione sul device una porzione di codice sviluppata dall'utente. L'esecuzione del plugin prevede che venga interrotta solo alla ricezione del comando di stop da parte di lotronic.

Sviluppo in Python

Il file Python in cui si articola un generico plugin asincrono deve prevedere la definizione di una funzione “main” che prenda in ingresso i seguenti parametri:

- **plugin_name:** <STRING> nome del plugin;
- **params:** <JSON> oggetto JSON contenente i parametri d'ingresso con cui il plugin verrà messo in esecuzione;
- **api:** <OBJECT> oggetto/libreria contenente utility per lo sviluppo dei plugin (vedi sezione *Plugin API*).

I parametri sopracitati saranno valorizzati esternamente da lotronic (*params*) e Lightning-rod (*plugin_name*, *api*). Di seguito viene riportato il template di un plugin asincrono Python.

```
# User imports
< SECTION TO SPECIFY MODULES IMPORTS >

def main(plugin_name, params, api):

    # SET PLUGIN LOGGER
    logging = api.getLogger(plugin_name)
    input = params['< PARAMS JSON KEY >']

    while(True):
        # -----
        # USER CODE

        ...

        # -----
```



Nello specifico vi sarà:

- una sezione dedicata agli *import* delle librerie specifiche del plugin
- la funzione *main* sopracitata che contiene:
 - l'implementazione della logica del plugin all'interno ad esempio di un *loop* che mantenga il plugin in esecuzione
 - l'abilitazione del sistema di logging (opzionale)
 - l'acquisizione/elaborazione dei parametri d'ingresso (*params*)

Esempio plugin Python asincrono:

```
# User imports
import time
from datetime import datetime

#input_parameters -> {"name": "S4T" }

def main(plugin_name, params, api):

    logging = api.getLogger(plugin_name)

    while(True):
        // -----
        // USER CODE
        now = datetime.now().strftime( "%-d %b %Y %H:%M:%S.%f" )
        logging.info("I'm "+str(params['name'])+" @ "+now)
        time.sleep(1)
        // -----
```

Sviluppo in NodeJS

Il file NodeJS in cui si articola un generico plugin asincrono deve prevede la definizione di una funzione “main” (`exports.main`) che prenda in ingresso i seguenti parametri:

- **`plugin_name`:** <STRING> nome del plugin;
- **`arguments`:** <JSON> oggetto JSON contenente i parametri d'ingresso con cui il plugin verrà messo in esecuzione;
- **`api`:** <OBJECT> oggetto/libreria contenente utility per lo sviluppo dei plugin (vedi sezione *Plugin API*).

I parametri sopracitati saranno valorizzati esternamente da lotronic (`params`) e Lightning-rod (`plugin_name, api`).

All'interno della funzione main è possibile richiamare le api fornite per lo sviluppo dei plugin e deve essere specificato un costrutto NodeJS (es. `setInterval`). Tale costrutto consente di implementare al suo interno una funzione che rimanga in esecuzione e che potrà essere terminata da lotronic mediante il comando di stop del plugin.

Di seguito viene riportato il template di un plugin asincrono NodeJS.

```
exports.main = function (plugin_name, arguments, api){  
  
    var logger = api.getLogger(plugin_name, 'debug');  
  
    // -----  
    // USER CODE  
    var input = arguments.< ARGUMENTS JSON KEY >;  
  
    setInterval(function(){  
  
        ...  
  
    }, < TIMER - MILLISECONDS > );  
  
    // -----  
};
```



Nello specifico vi sarà:

- la funzione **main** sopracitata che contiene:
 - eventuali *require* delle librerie specifiche del plugin
 - l'implementazione della logica del plugin all'interno ad esempio di un *loop* che mantenga il plugin in esecuzione
 - l'abilitazione del sistema di logging (opzionale)
 - l'acquisizione/elaborazione dei parametri d'ingresso (*arguments*)

Esempio plugin *NodeJS* asincrono:

```
//input_parameters -> {"name": "Iotronic"}\n\nexports.main = function (plugin_name, arguments, api){\n\n    var logger = api.getLogger(plugin_name, 'info');\n\n    // -----\n    // USER CODE\n    var name = arguments.name;\n\n    logger.info("Hello plugin starting...");\n\n    setInterval(function() {\n\n        logger.info('Hello '+name+'!');\n\n    }, 3000);\n    // -----\n\n};
```

Plugin API

Set di utility fornite agli sviluppatori dei plugin per accedere a dati/informazioni del device.

Sviluppo in Python

- `getLogger(plugin_name, console=None)`
Restituisce un oggetto “logger” utilizzabile all’interno del plugin per interfacciarsi con il sistema di logging del device.
- `getExtraInfo()`
Restituisce un oggetto JSON corrispondente al campo “extra” che contiene le informazioni supplementari relative allo scenario che opzionalmente è possibile specificare in fase di registrazione di un device in lotronic.

Sviluppo in NodeJS

- `getLogger(plugin_name, loglevel)`
Vedi sezione Python. Nel caso specifico qui è possibile specificare con “loglevel” una stringa col livello di debug.
- `getExtraInfo()`
Vedi sezione Python.
- `getPosition()`
Restituisce un oggetto JSON contenente le informazioni di geolocalizzazione: latitudine, longitudine e altitudine; tali valori sono estrapolati localmente dal file settings.js di Lightning-rod e quindi valorizzati da remoto ad ogni update delle info del device (o in fase di registrazione dello stesso).
- `getBoardId()`
Restituisce il device ID di lotronic.
- `getLocalTime()`
Restituisce il timestamp della propria timezone.
- `getUtcTime()`
Restituisce il timestamp in formato UTC.



PluginExec development tool

Questo tool di sviluppo ha lo scopo di permettere l'esecuzione e test di plugin sul device dove è installato Lightning-rod (LR): questa guida è utilizzabile dalla versione di LR v2.3.4 in poi.

Nello specifico è data la possibilità di mettere in esecuzione i plugin Python:

- sia che essi siano stati caricati localmente (senza passare dal workflow di deploy di lotronic); questo scenario è stato denominato “**local**”.
- sia che essi siano già stati iniettati tramite lotronic; questo scenario è stato nominato invece “**injected**”.

Installazione

1. Creare l'ambiente di sviluppo dei plugin:
 - a. eseguire il comando “create_plugin_env” nella locazione scelta; output del comando:

```
PluginExec Environment Creation
  - folder pluginExec created.
  - environment created.
```
 - b. verrà creata in questa locazione una cartella “pluginExec/” che conterrà:
 - i. **exec_plg**: eseguibile per mettere in esecuzione i plugin in fase di sviluppo e test.
 - ii. **lib/**: cartella che contiene le stesse librerie “wrapper-python” usati da Lightning-rod (LR) per l'esecuzione dei plugin Python; differiscono da quelle incluse in LR solo per la gestione degli output/logging e path di sistema.
 - iii. **node_modules/**: link simbolico al modulo NodeJS “python-shell” installato con LR
(/usr/lib/node_modules/@mdslab/iotronic-lightning-rod/node_modules/python-shell/)
 - iv. **plugins/**: cartella che conterrà i plugin caricati localmente sul device nelle fasi di sviluppo prima di essere iniettati da lotronic.

Esecuzione

Come descritto nelle sezioni precedenti, si identificano due possibili scenari di utilizzo: “*local*” e “*injected*”.

Lanciando il comando “`exec_plg`”, verrà restituito un “*help*”:

Usage:

```
./exec_plg <local|injected> <sync|async> <PLUGIN-NAME>  
./exec_plg <local|injected> list
```

Local scenario

Si presuppone che il plugin da testare/sviluppare non sia stato iniettato sul device da Iotronic.

Step:

1. Copiare/trasferire nella cartella “*pluginExec/plugins/*”, la cartella del plugin che si vuole eseguire. Tale cartella dovrà:
 - essere avere lo stesso nome del plugin (es. *py_async*)
 - es: *plugins/py_async/*
 - contenere il plugin ed il suo file di configurazione (anch'essi nominati con lo stesso nome del plugin) rispettivamente con estensione “.py” e “.json”:
 - *py_async/*
 - *py_async.py*
 - *py_async.json*
2. Spostarsi all'interno della cartella “*pluginExec/*” creata con la procedura descritta nella sezione precedente.
3. Eseguire dalla cartella “*pluginExec/*”:
 - `./exec_plg local <sync|async> <PLUGIN-NAME>`
 - es: `./exec_plg local async py_async`
 - l'output del plugin verrà restituito sia a shell che sul file di log (`/var/log/iotronic/plugins/<PLUGIN-NAME>.log`)



Injected scenario

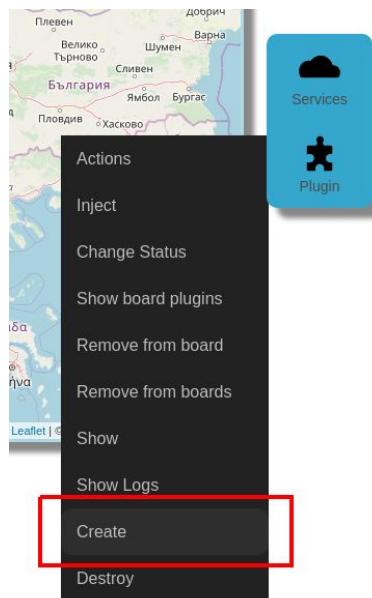
Si presuppone che il plugin rilasciato ufficialmente da Iotronic sia stato iniettato con successo sul device.

Step:

1. Spostarsi all'interno della cartella “*pluginExec/*” creata con la procedura descritta nella sezione precedente.
2. dalla cartella “*pluginExec/*” eseguire:
 - a. `./exec_plg injected <sync|async> <PLUGIN-NAME>`
 - i. `es: ./exec_plg injected async py_async`
 - b. l'output del plugin verrà restituito sia a shell che sul file di log
`(/var/log/iotronic/plugins/<PLUGIN-NAME>.log)`

Upload plugin in Iotronic

Per l'upload di un plugin in Iotronic attraverso la dashboard di amministrazione è necessario selezionare il menù “Create” del *Plugin Manager (PM)*, come mostrato qui di seguito:



Tramite il form di registrazione del plugin dovranno essere specificate le seguenti informazioni:

- nome del plugin
- versione (vedi sezione *Versioning*)
- la tipologia di plugin:
 - Sync (sincrono) | Async (asincrono)
 - NodeJS | Python
- una breve descrizione
- il file contenente il codice sorgente (*.js* | *.py*)
- il file JSON contenente i parametri d'ingresso di default con cui il plugin verrà messo in esecuzione.



Create Plugin

Plugin Name: Version:

Type: Category:

Description:

Plugin Parameters:
 Nessun file selezionato

Code:
 Nessun file selezionato

API di riferimento:

- [POST /v1/plugins/](#)

Versioning

Per taggare un plugin con una versione specifica occorre rispettare il seguente formato

MAJOR.MINOR.PATCH

costituito da tre numeri che insieme formano il numero di versione. I tre numeri di versione, separati da punti, dovranno essere incrementati secondo le seguenti *best practices*:

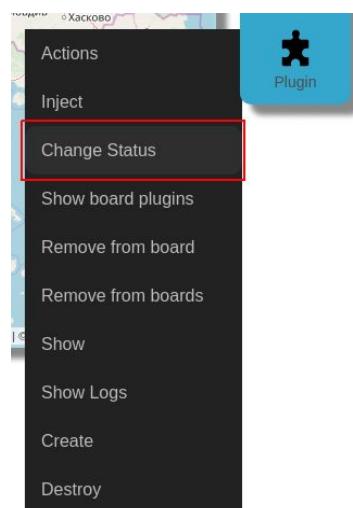
1. *MAJOR* quando le modifiche apportate sono significative e la nuova versione è incompatibile con le precedenti (in funzionalità, interfacce, formato dati, ecc);
2. *MINOR* quando vengono aggiunte o modificate funzionalità comunque retrocompatibili;
3. *PATCH* solitamente incrementato dopo bug-fixing o modifiche minimali non relative a cambiamento nella funzionalità esposta.

Review & Release di un plugin

Tra i compiti di un amministratore della piattaforma lotronic vi è il compito di revisionare il codice del plugin prima di poterlo rilasciare e rendere disponibile ad essere iniettato sui device.

Una volta che il plugin verrà caricato su lotronic di default sarà taggato come “*Unreleased*” e in tale condizione non sarà possibile iniettarlo su un device.

Per poter visionare il codice del plugin e rilasciarlo, un amministratore della piattaforma dovrà accedere tramite la dashboard al sottomenù del *Plugin Manager*, “Change Status”:



dal quale si accede alla schermata dove sono elencati tutti i plugin caricati su lotronic.

Change Tag to Plugin

entries								
id	name	category	version	type_id	tag_id	updated_at	created_at	
156	hello	async	0.0.1	NodeJS	Released	2018-11-21T11:29:16.000Z	2018-11-21T11:28:44.000Z	
163	hello	async	0.0.2	NodeJS	Released	2018-11-21T11:29:24.000Z	2018-11-21T11:29:04.000Z	
65	extra	sync	0.0.1	Python	Released	2018-09-25T08:06:25.000Z	2018-09-25T08:06:15.000Z	
173	except_call	sync	0.0.1	Python	Released	2019-03-14T10:14:45.000Z	2019-03-14T10:14:33.000Z	
172	except	async	0.0.1	Python	Released	2019-03-14T10:11:56.000Z	2019-03-14T10:11:44.000Z	
121	echo	sync	1.0.0	NodeJS	Unreleased	2019-04-10T10:37:12.000Z	2018-10-05T15:46:35.000Z	
186	echo	sync	2.0.0	NodeJS	Released	2019-04-10T10:44:53.000Z	2019-04-10T10:38:20.000Z	
181	ciao	async	0.0.1	Python	Released	2019-03-21T14:46:30.000Z	2019-03-21T14:45:57.000Z	
135	basethreshold	async	0.0.1	Python	Released	2018-11-27T16:42:58.000Z	2018-11-05T09:09:20.000Z	
187	acc_raw_plugin	async	1.0.0	Python	Released	2019-07-01T16:28:19.000Z	2019-07-01T16:28:08.000Z	

Showing 11 to 20 of 20 entries

Previous 1 2 Next



Selezionando il plugin apparirà la schermata da cui sarà possibile visionare il codice del plugin e cambiarne lo stato in “*released*”:

Tag Plugin py_async

Tag

Unreleased

Released

Unreleased

Version

0.0.1

Description

test python async plugin

Plugin Parameters

{"name": "LR"}

Code

```
## See the License for the specific language governing permissions and
## limitations under the License.
#####
#####
# User imports
import time
from datetime import datetime

def main(params):

    while(True):
        now = datetime.now().strftime( "%d %b %Y %H:%M:%S.%f" )
        print("I'm "+str(params['name'])+" @ "+now)
        time.sleep(1)
```

Previous

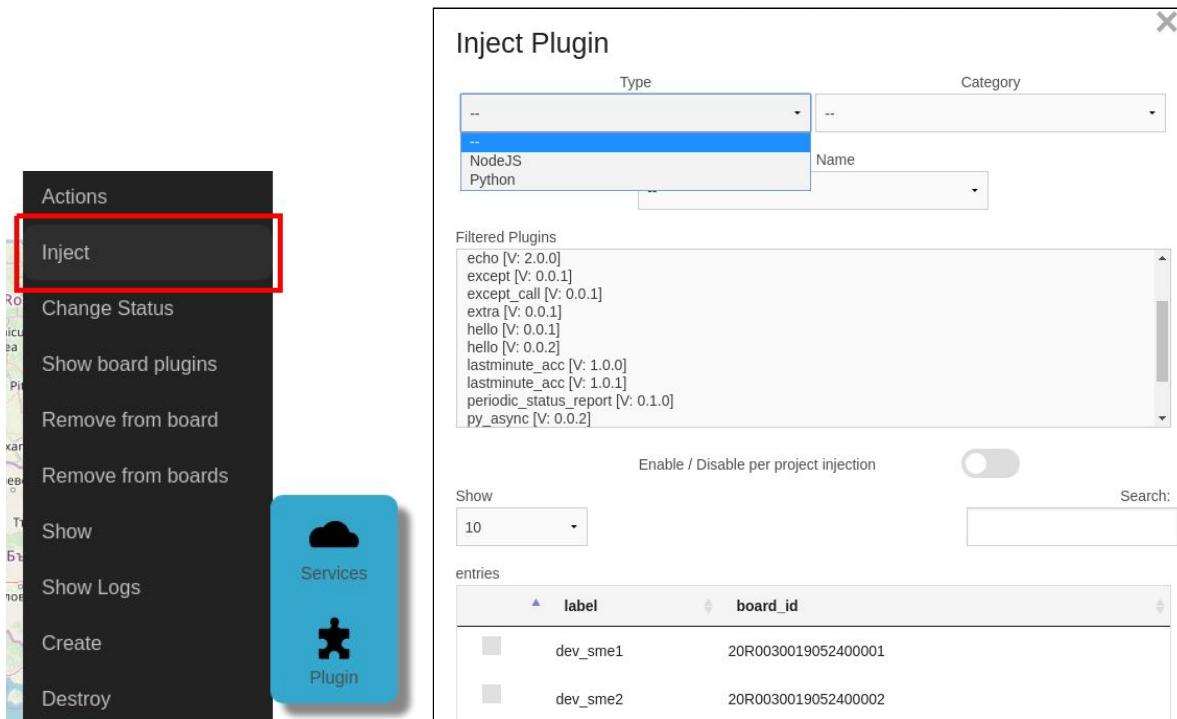
Update Tag

API di riferimento:

- **POST** </v1/plugins/{plugin}/tag>

Inject del plugin in un device

Mediante la dashboard di amministrazione di lotronic, dal sottomenù “*Inject*” del *Plugin Manager (PM)* l’amministratore può iniettare un plugin “*released*” su uno o più device (modalità *batch*). Dal panel di “*Inject Plugin*” è possibile selezionare il plugin di interesse: la ricerca prevede l’ausilio di filtri per “category” (*sync* | *async*), per “type” (*NodeJS* | *Python*) o per nome:



E’ tuttavia necessario selezionare uno o più device dove iniettare il plugin; se un plugin deve essere iniettato su tutti i device di un progetto occorre abilitare l’apposita opzione:

Enable / Disable per project injection

Infine, come ultimo step, è possibile:

- tramite l’opzione “*On Boot*”, specificare se il plugin dovrà essere messo automaticamente in esecuzione al boot di Lightning-rod;
- tramite l’opzione “*Force*”, forzare l’inject (sovrascrittura) del plugin sul device.



On Boot	Force
False	False

Inject

API di riferimento:

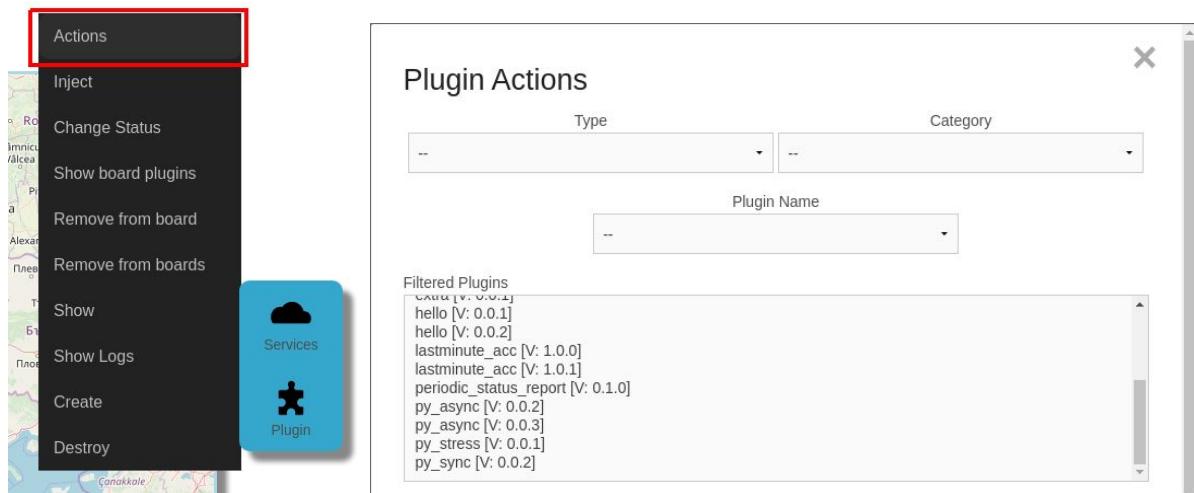
- **POST** [`/v1/plugins/{plugin}/tag`](#)

Plugin actions

Dal panel “*Plugin Actions*” è possibile gestire l’esecuzione dei plugin iniettati sui device. Nello specifico è possibile eseguire le seguenti azioni in funzione del fatto che il plugin preso in esame sia asincrono o sincrono:

- asincroni: *start | stop | restart*
- sincroni: *exec*

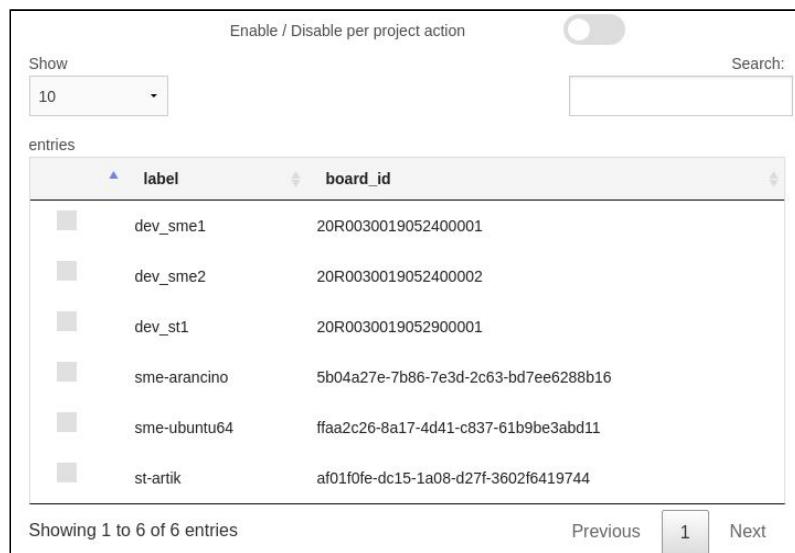
Da questo panel dopo aver selezionato il plugin:



The screenshot shows the 'Plugin Actions' panel. On the left, a sidebar menu is open, with the 'Actions' item highlighted by a red box. The main area displays a list of 'Filtered Plugins' with the following entries:

Label	board_id
dev_sme1	20R0030019052400001
dev_sme2	20R0030019052400002
dev_st1	20R0030019052900001
sme-arancino	5b04a27e-7b86-7e3d-2c63-bd7ee6288b16
sme-ubuntu64	ffaa2c26-8a17-4d41-c837-61b9be3abd11
st-artik	af01f0fe-dc15-1a08-d27f-3602f6419744

e i device su cui eseguire l’azione:



The screenshot shows the 'entries' panel. At the top, there is a toggle switch labeled 'Enable / Disable per project action'. Below it are 'Show' and 'Search' fields. The main area is a table titled 'entries' with columns 'label' and 'board_id', containing the following data:

label	board_id
dev_sme1	20R0030019052400001
dev_sme2	20R0030019052400002
dev_st1	20R0030019052900001
sme-arancino	5b04a27e-7b86-7e3d-2c63-bd7ee6288b16
sme-ubuntu64	ffaa2c26-8a17-4d41-c837-61b9be3abd11
st-artik	af01f0fe-dc15-1a08-d27f-3602f6419744

At the bottom, it says 'Showing 1 to 6 of 6 entries' and has 'Previous' and 'Next' buttons.



in funzione del plugin selezionato verranno visualizzate le azioni disponibili; per gli asincroni:

A screenshot of a user interface titled "Parameters Set". It shows a dropdown menu with "Default" selected. Below the menu are three rounded rectangular buttons labeled "Start", "Stop", and "Restart".

per i sincroni:

A screenshot of a user interface titled "Parameters Set". It shows a dropdown menu with "Default" selected. Below the menu is a single button labeled "Exec".

Per le azioni “start” ed “exec” sarà necessario specificare il set di parametri d’ingresso con cui mettere in esecuzione il plugin:

A screenshot of a user interface titled "Parameters Set". A dropdown menu displays four options: "Default", "Default" (which is highlighted in blue), "New", and "Latest".

- **Default:** sono i parametri specificati in fase di creazione del plugin;
- **Latest:** set di parametri specificati nell’ultima esecuzione del plugin;
- **New:** selezionando tale opzione verrà data la possibilità di caricare un nuovo set di parametri per questa specifica esecuzione e che verranno memorizzati da lotronic come “*latest parameters*” associati al device selezionato.

API di riferimento:

- **POST** `/v1/boards/{board}/plugins/{plugin}`
- **POST** `/v1/projects/{project}/plugins/{plugin}`

Update plugin

Tramite il panel “Show” del *Plugin Manager* (PM) è possibile modificare un plugin.

Plugins								
Show								
entries								
id	name	category	version	type_id	tag_id	updated_at	created_at	
22	py_sync	sync	0.0.1	Python	Unreleased	2018-06-01T14:19:37.000Z	2018-05-05T16:54:58.000Z	
29	py_async	async	0.0.1	Python	Unreleased	2018-06-01T14:19:27.000Z	2018-05-08T10:08:05.000Z	
43	py_sync	sync	0.0.2	Python	Released	2018-05-30T13:18:51.000Z	2018-05-30T13:18:03.000Z	
50	py_async	async	0.0.2	Python	Released	2018-05-30T13:30:01.000Z	2018-05-30T13:29:50.000Z	
65	extra	sync	0.0.1	Python	Released	2018-09-25T08:06:25.000Z	2018-09-25T08:06:15.000Z	
114	py_stress	async	0.0.1	Python	Released	2018-10-04T13:12:23.000Z	2018-10-04T13:11:44.000Z	
121	echo	sync	1.0.0	NodeJS	Unreleased	2019-04-10T10:37:12.000Z	2018-10-05T15:46:35.000Z	
135	basethreshold	async	0.0.1	Python	Released	2018-11-27T16:42:58.000Z	2018-11-05T09:09:20.000Z	
149	py_async	async	0.0.3	Python	Released	2018-11-21T09:11:46.000Z	2018-11-21T09:11:28.000Z	

Selezionando un plugin si entrerà nel panel di modifica in cui si potranno modificare, senza che ciò necessiti un avanzamento di versione del plugin, i seguenti campi:

- *Description*
- *Plugin Parameters*

Invece se verrà apportata una modifica al codice sarà necessario incrementare la versione (vedi sezione *Versioning*).



Update Plugin py_async

Plugin Name	Version
py_async	0.0.3

Description

async py plugin

Plugin Parameters

Scegli file Nessun file selezionato

```
{"name": "S4T"}
```

Code

Scegli file Nessun file selezionato

```
#####
#####
## Copyright (C) 2018 Nicola Peditto
##
## Licensed under the Apache License, Version 2.0 (the "License");
## you may not use this file except in compliance with the License.
## You may obtain a copy of the License at
##
## http://www.apache.org/licenses/LICENSE-2.0
##
## Unless required by applicable law or agreed to in writing, software
## distributed under the License is distributed on an "AS IS" BASIS,
## WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
## See the License for the specific language governing permissions and
## limitations under the License.
```

[Previous](#) [Update](#)

N.B.: il cambiamento del nome comporterà la creazione di un nuovo plugin in stato “unreleased”.

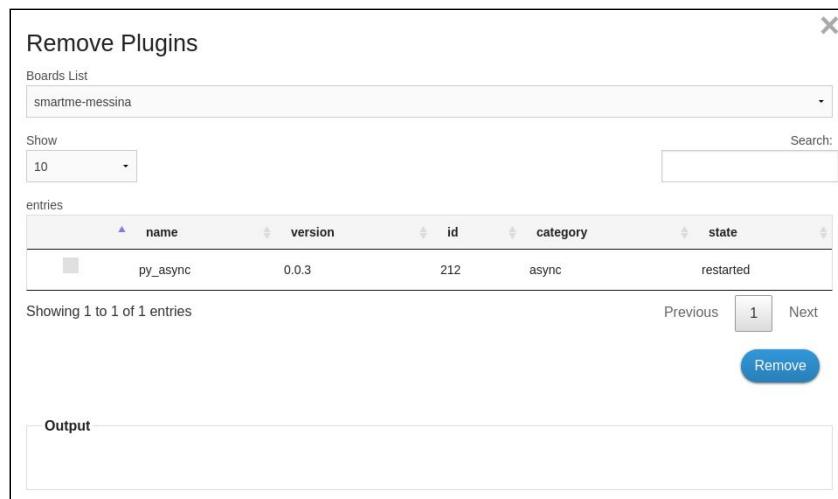
API di riferimento:

- **PATCH** </v1/plugins/{plugin}>

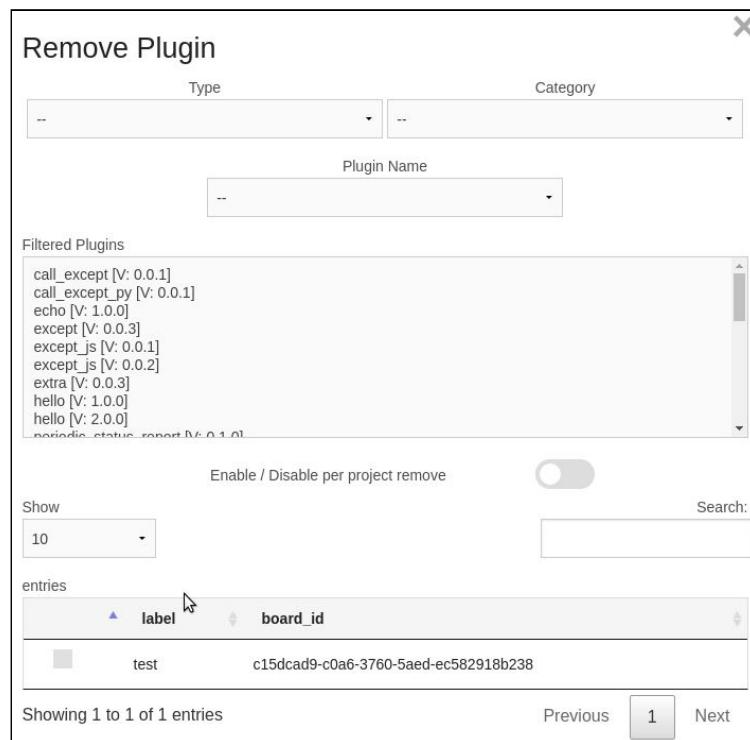
Rimozione del plugin dal device

Per rimuovere un plugin da un device è possibile procedere in due modi:

- partendo dalla selezione del device è possibile rimuovere uno o più plugin dal panel **“Remove from board”**.



- partendo invece dalla selezione del plugin è possibile rimuovere un plugin da uno o più device dal panel **“Remove from boards”**:



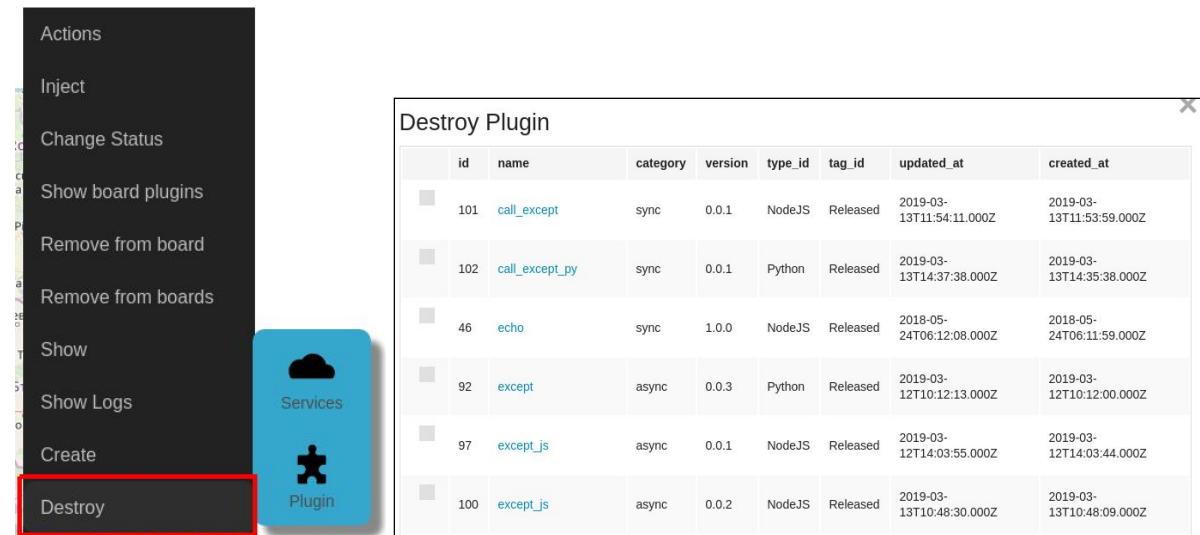


API di riferimento:

- **DELETE** [`/v1/boards/{board}/plugins/{plugin}`](#)
- **DELETE** [`/v1/projects/{project}/plugins/{plugin}`](#)

Rimozione del plugin da lotronic

Dal panel “*Destroy*” del *Plugin Manager* è possibile selezionare uno o più plugin da rimuovere da lotronic.



The screenshot shows the 'Actions' sidebar with various options like 'Inject', 'Change Status', and 'Destroy'. The 'Destroy' button is highlighted with a red box. To its right is a 'Services' icon, and below it is a 'Plugin' icon. A modal window titled 'Destroy Plugin' is open, displaying a table of plugins with columns: id, name, category, version, type_id, tag_id, updated_at, and created_at. The table contains the following data:

	id	name	category	version	type_id	tag_id	updated_at	created_at
1	101	call_except	sync	0.0.1	NodeJS	Released	2019-03-13T11:54:11.000Z	2019-03-13T11:53:59.000Z
2	102	call_except_py	sync	0.0.1	Python	Released	2019-03-13T14:37:38.000Z	2019-03-13T14:35:38.000Z
3	46	echo	sync	1.0.0	NodeJS	Released	2018-05-24T06:12:08.000Z	2018-05-24T06:11:59.000Z
4	92	except	async	0.0.3	Python	Released	2019-03-12T10:12:13.000Z	2019-03-12T10:12:00.000Z
5	97	except_js	async	0.0.1	NodeJS	Released	2019-03-12T14:03:55.000Z	2019-03-12T14:03:44.000Z
6	100	except_js	async	0.0.2	NodeJS	Released	2019-03-13T10:48:30.000Z	2019-03-13T10:48:09.000Z

N.B.: la rimozione sarà possibile se il plugin non risulta essere iniettato in nessun device. In tal caso è necessario rimuoverlo prima da tutti i device e poi eseguire la rimozione dal Cloud.

API di riferimento:

- **DELETE** </v1/plugins/{plugin}>



Show plugin logs

Il *Plugin Manager* (PM) espone la funzionalità di recupero dei log per un plugin iniettato in un device. In particolare è possibile specificare il numero di righe di log che si vuole recuperare (vedi “*Rows*”).

The screenshot shows the smartme.IO interface. On the left, there is a sidebar with various actions: Inject, Change Status, Show board plugins, Remove from board, Remove from boards, Show, Show Logs (which is highlighted with a red box), Create, and Destroy. Below the sidebar are two icons: Services and Plugin. To the right of the sidebar is a main panel titled "Plugin Logs". It contains a search bar for "Plugin Name" (set to "py_async [v: 0.0.3]") and a "Rows" input field (set to "10"). There is also a toggle switch for "Enable / Disable per project logs" and a "Search:" input field. A table titled "entries" lists one entry: "smartme-messina" with "a640cd28-81c7-8bd3-d791-68eed3ab64ec". At the bottom, it says "Showing 1 to 1 of 1 entries" and has "Previous" and "Next" buttons, along with a "Get" button.

Una volta che si è selezionato il plugin, il numero di rows e il device (un device per richiesta), apparirà la schermata dove visionare i log recuperati.

The screenshot shows a message for the board "smartme-messina". The message header says "Message for board smartme-messina" and there is a "Refresh" button. Below the header, it says "Result: SUCCESS". The main area displays a scrollable log output:
INFO:root:I'm S4T @ 6 Sep 2019 08:59:18.227969
INFO:root:I'm S4T @ 6 Sep 2019 08:59:19.229692
INFO:root:I'm S4T @ 6 Sep 2019 08:59:20.231414
INFO:root:I'm S4T @ 6 Sep 2019 08:59:21.233135
INFO:root:I'm S4T @ 6 Sep 2019 08:59:22.234862
INFO:root:I'm S4T @ 6 Sep 2019 08:59:23.236587
INFO:root:I'm S4T @ 6 Sep 2019 08:59:24.238310
INFO:root:I'm S4T @ 6 Sep 2019 08:59:25.240032
INFO:root:I'm S4T @ 6 Sep 2019 08:59:26.241757
INFO:root:I'm S4T @ 6 Sep 2019 08:59:27.243514

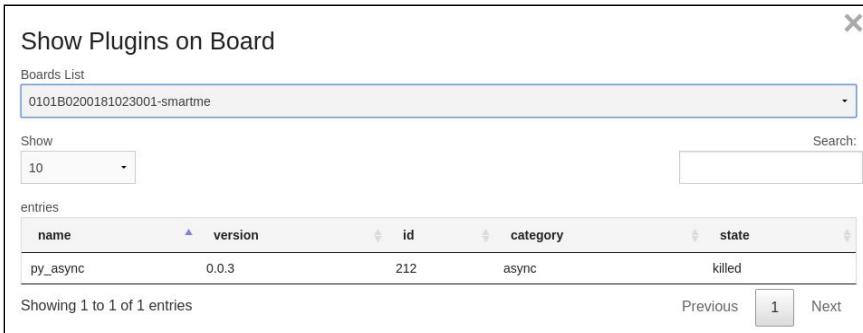
API di riferimento:

- [**GET** /v1/boards/{board}/plugins/{plugin}/logs](#)
- [**GET** /v1/projects/{project}/plugins/{plugin}/logs](#)

Show plugin in device

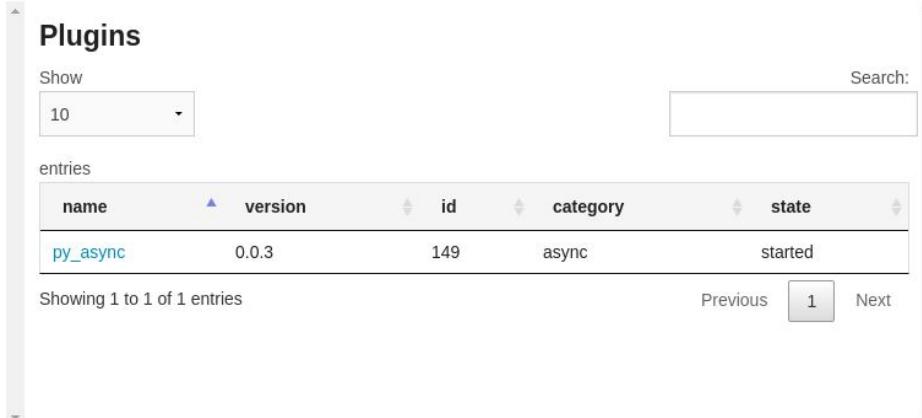
Dalla dashboard di amministrazione di lotronic è possibile visualizzare i plugin iniettati su un device in due modi:

- dal menù “Show board plugins” del *Plugin Manager*:



name	version	id	category	state
py_async	0.0.3	212	async	killed

- dal “Board Information” panel raggiungibile cliccando sul nome del device dall’homepage della dashboard:



name	version	id	category	state
py_async	0.0.3	149	async	started

API di riferimento:

- **GET** </v1/boards/{board}/plugins>



Plugins in Lightning-rod

File di sistema

Il *Plugin Manager* gestisce localmente le informazioni dei plugin iniettati all'interno del seguente data path:

- /var/lib/iotronic
 - plugins/
 - *plugins.json*
 - <PLUGIN_DIR>/
 - ...

All'interno di ogni cartella di un plugin (<PLUGIN_DIR>) saranno presenti due file:

- il file del codice sorgente del plugin: <PLUGIN_NAME>.py|js
- il file JSON dei parametri d'ingresso del plugin: <PLUGIN_NAME>.json

Il file *plugins.json* viene utilizzato dal *Plugin Manager* per tenere traccia delle informazioni relative alle caratteristiche intrinseche dei plugin; tali informazioni sono utilizzate da LR in fase di boot per avviare in automatico i plugin che sono stati iniettati sul device con il parametro “*autostart*” abilitato.



SmartMe.IO S.r.l.

Sede Legale:

Via Osservatorio, 1 - 98121 Messina (ME) – ITALIA

Sede Operativa:

Dipartimento di Ingegneria, C/da Di Dio, 1

98166 Villaggio S.Agata, Messina (ME) – ITALIA

Num. REA ME238676

P.IVA/C.F. 03457040834

Tel. +39 090 676 3644

Email: info@smartme.io

Web: <http://smartme.io/>