Smart Greenhouse

Progetto di Pervasive Computing

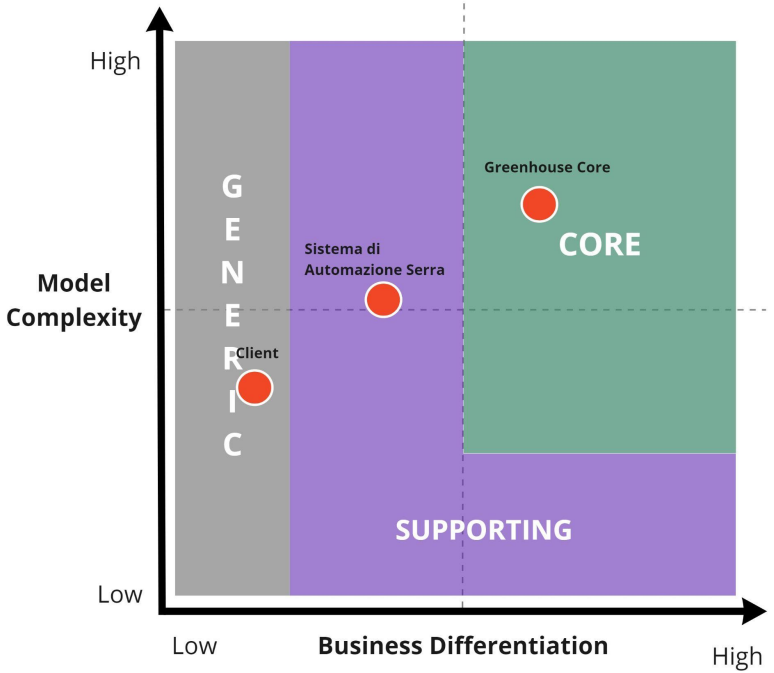
Mengozzi Maria, Vitali Anna

Introduzione

- Per il progetto si è pensato di realizzare un’applicazione che consenta la **gestione** e il **monitoraggio** di un **complesso** di **serre intelligenti**
- All'interno delle serre è possibile coltivare **diverse piantagioni**. All’interno di **una serra** viene coltivata un’**unica** tipologia di **pianta**
- I **parametri** che si vogliono andare a monitorare sono:
  - luminosità
  - temperatura ambientale
  - umidità dell'aria
  - umidità del terreno

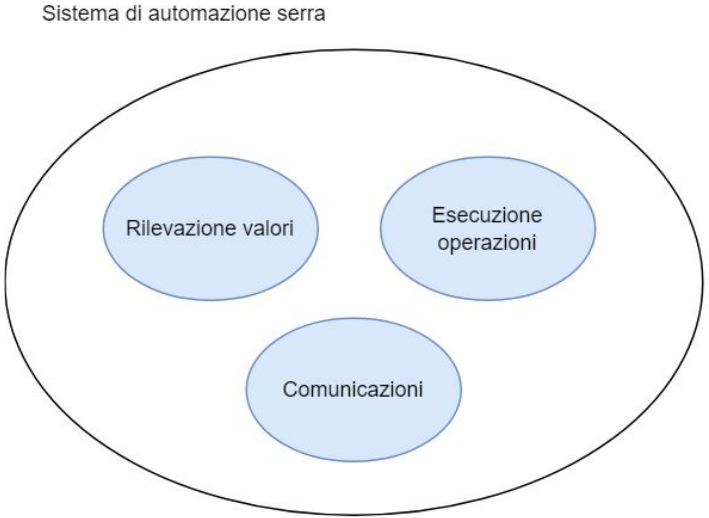
Domain Driven Design - Subdomain

- i **sottodomini** individuati sono tre:
  - **Sistema di automazione serra**
  - **Greenhouse core**
  - **Client**
- ciascuno costituito da un certo numero di **bounded-context**



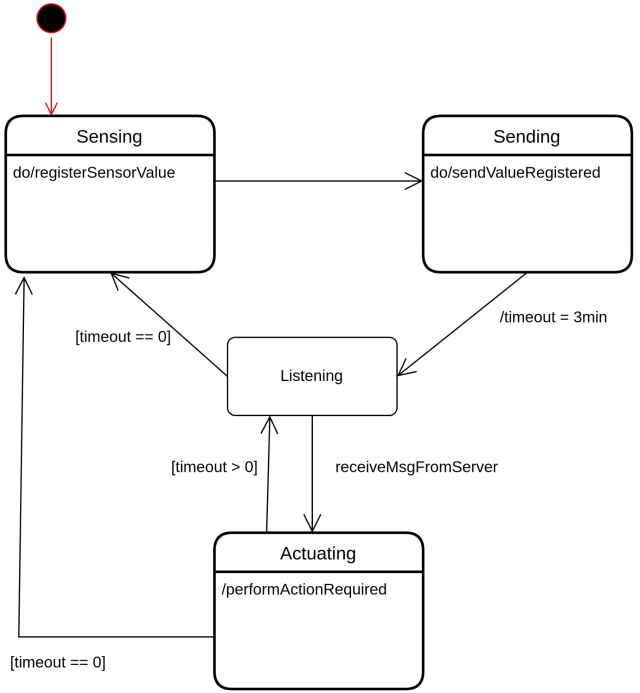
Sistema di automazione serra

- **Rilevazione valori**, racchiude i diversi sensori necessari per poter rilevare i parametri delle piante
- **Esecuzione operazioni**, racchiude gli elementi che si occupano di gestire la logica per l'esecuzione delle diverse operazioni
- **Comunicazioni**, comprende tutti i diversi elementi necessari per poter inviare i dati rilevati e ricevere i comandi relativi alle operazioni da eseguire

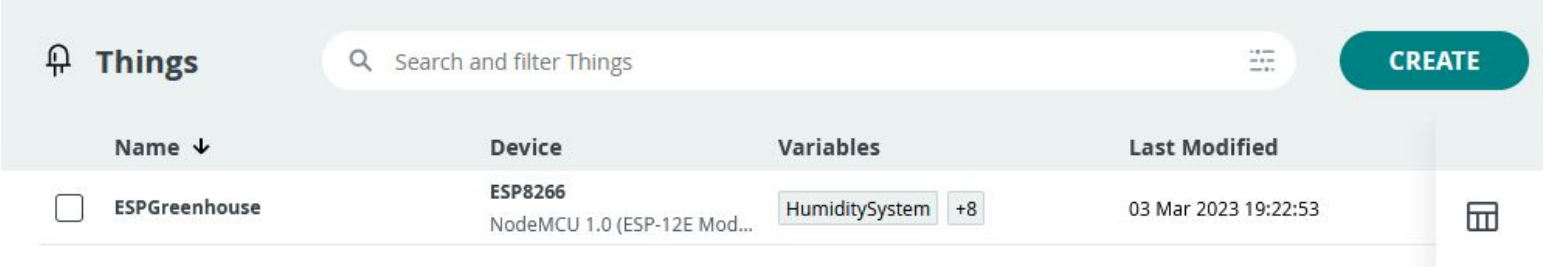


Architettura del Microcontrollore: Super-loop

- il **super-loop** rappresenta una struttura di programmazione costituita da:
  - una fase di **inizializzazione**
  - un **loop** che si ripete all'infinito
- Nel loop vengono eseguiti i diversi tasks del sistema, che alternano fasi di *sense* *decide* e *act*
- L'esecuzione dei diversi task viene alternata grazie ad uno **scheduler**

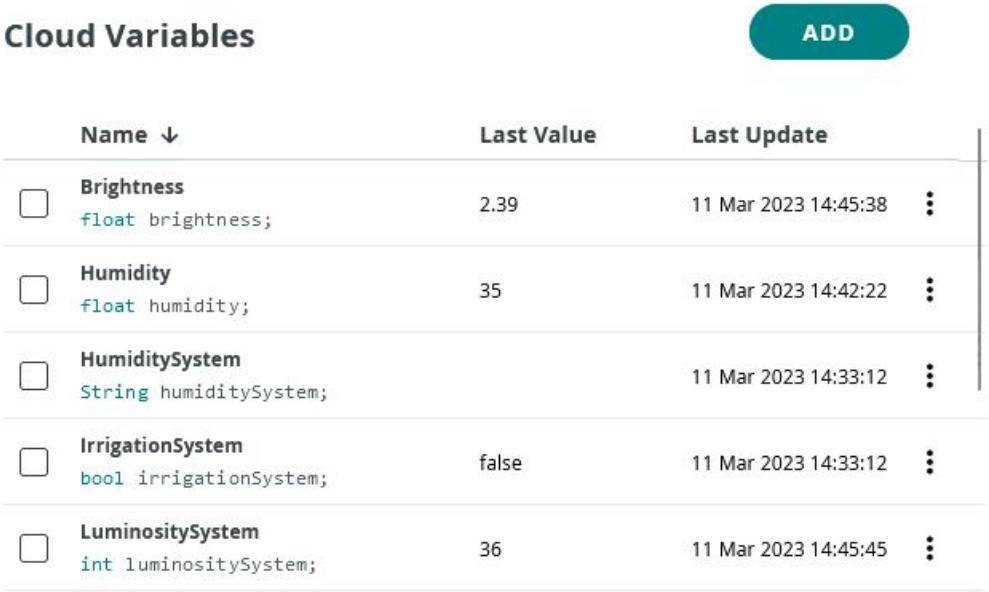


Arduino IOT Cloud



- Arduino Cloud offre la possibilità di creare diverse “**Things**” ovvero dispositivi virtuali, associati a dispositivi fisici
  - mette a disposizione un web editor per poter scrivere il codice da caricare sui dispositivi
    - Consentendo anche **Over-The-Air-Uploads**
  - È possibile associare al dispositivo delle **variabili Cloud**

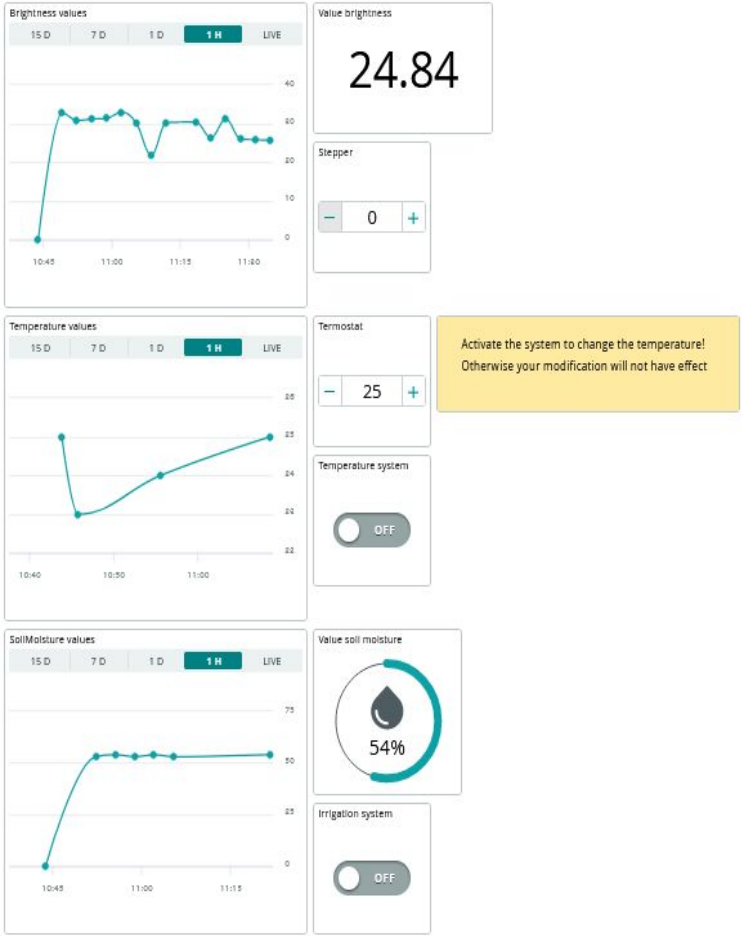
Arduino IoT Cloud - Cloud Variables



- Sono state create **otto variabili** che ci consentono di:
  - visualizzare l'**andamento** dei parametri
  - lo **stato** dei sistemi di attuazione
- Sono **sincronizzate** con il cloud e vengono **aggiornate** automaticamente

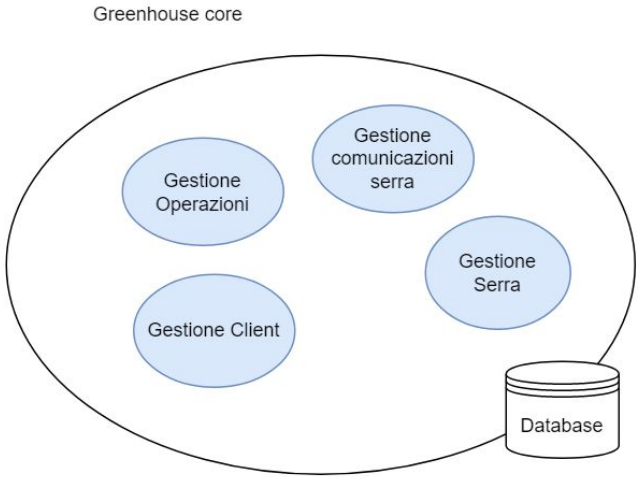
Arduino IoT Cloud - Dashboard

- Strumento che consente di associare **elementi grafici** alle variabili Cloud
  - Le dashboards possono essere **condivise** con altri utenti
  - Possibile accedervi anche tramite **l'applicazione cellulare**
- Possibile utilizzare elementi che **modificano** il **valore** o lo **stato** delle variabili
- I dati vengono mantenuti **salvati** per un certo periodo di tempo



Greenhouse core

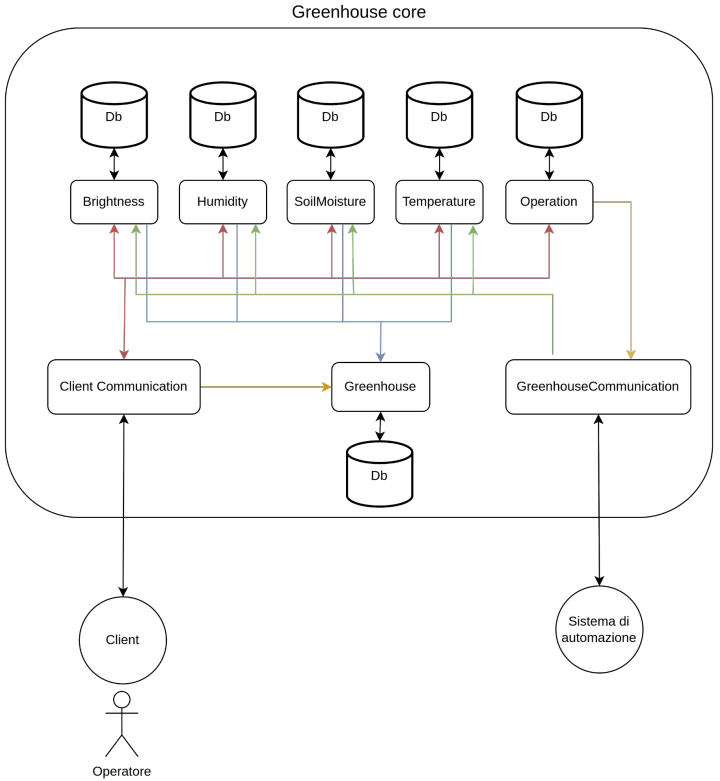
- **Gestione comunicazioni serra**, gestisce tutte le interazioni che avvengono con il sistema di automazione;
- **Gestione Serra**, contiene la logica che permette di automatizzare la serra;
- **Gestione Operazioni**, amministra tutte le operazioni che avvengono all'interno della serra
- **Gestione Client**, gestisce le interazioni con i diversi client del sistema, quello mobile e desktop



Architettura a microservizi

Microservizi: componenti **indipendenti** che eseguono ciascun processo applicativo come un servizio

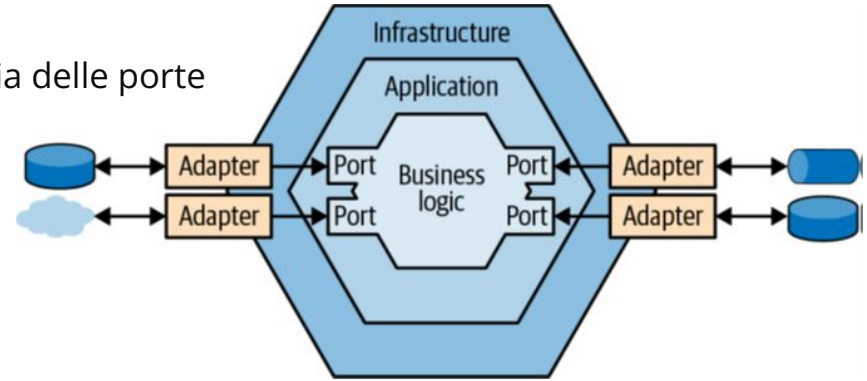
- comunicano attraverso la rete ed espongono un'**interfaccia** ben definita
- possono essere **aggiornati**, **distribuiti** e **ridimensionati** per rispondere alla richiesta di funzioni specifiche



Dettagli micro-servizio: Architettura Esagonale

Il microservizi sono stati realizzati tramite un'architettura **esagonale port and adapters**

- la **logica di dominio** è racchiusa nella parte più interna dell'architettura e definisce delle porte che devono essere implementate
- prevede l'applicazione del ***principio di inversione delle dipendenze***
- gli **adapter**:
  - sono la concreta implementazione dell'interfaccia delle porte
  - filtrano la comunicazione



API REST

- permette di inviare e ricevere in modo **semplice ed interoperabile** dati tra due diversi servizi
- **identifica le risorse** tramite HTTP URI
- sfrutta **metodi predefiniti** per operare sulle risorse
    - GET, POST, PUT, DELETE ...

Web of Things e thing description

Una **Thing Description** definisce un modello informativo di una *thing* basato sul vocabolario semantico e una serializzazione basata su ***JSON***

- ***properties***: che descrivono gli attributi del dispositivo
- ***actions***: che descrivono le funzioni che possono essere eseguite su un dispositivo
- ***events***: che definiscono i tipi di eventi che possono essere emessi da un dispositivo

```
"actions":{
      "fade":{
        "@type":"FadeAction",
        "title":"fade",
        "description":"fade the lamp to a given level.",
        "input":{"type":"string"},
        "links":[{
          "href":"mqtt://broker.mqtt-dashboard.com:1883",
          "op":{
            "level":{
              "type":"integer",
              "minimum":"0",
              "maximum":"255"
              }
            },
          "mqv:topic":"LUMINOSITY"
        }]
      }
  }
```

```
"properties":{
    "value":{
      "title":"current value",
      "description":"the level of light registered.",
      "type":"object",
      "properties":{
        "greenhouseId":{"type":"string"},
        "date":{"type":"date"},
        "value":{"type":"float"}
        }
    },
    "links":[{"href":"/brightness/history?id=&limit="}],
    "history":{
      "title":"history values",
      "description":
         "the history value of the light registered.",
      "type":"list",
      "properties":{
        "greenhouseId":{"type":"string"},
        "date":{"type":"date"},
        "value":{"type":"float"}
      }
    }
  }
```

```
"events":{
    "newData":{
      "title":"new data",
      "type":"string",
      "description":"new luminosity data.",
      "unit":"lux",
      "links":[{
        "href":"mqtt://broker.mqtt-dashboard.com:1883",
        "mqv:topic":"dataSG"}]
    }
  }
```

The end