

# APPOMI

2 TIPI DI DISPOSITIVI IOT

SENZORI

ATTUATORI

③ SENZORI

TEMPERATURA  
OSSIGENO  
BATITO

} MQTT

In questo caso sono generati automaticamente

- RETRIEVE EXTERNAL VALUES
- PUBBLICARE I DATI SU UN TOPIC

mosquitto\_pub -h localhost -t ossigeno -m 90

→ A questo punto sono processati dall'applicazione

② ATTUATORI

PILL DISPENSER

MASK

ACTUATOR NETWORK:

- Riceve chiamate CoAP

per ESPORRE RISPOSTE

Per fare certe operazioni

in risposta ai valori ricevuti

DUE DIVERSI COMPONENTI PER PROCESSARE I DATI

① Cloud Application →

- Si iscrive al TOPIC MQTT
- ottenere i valori e salvare su un DATABASE

② Remote Control Application →

- Prende i dati dal DATABASE e li PROCESSA TRAMITE

CHIAMATE COAP AGLI ATTUATORI

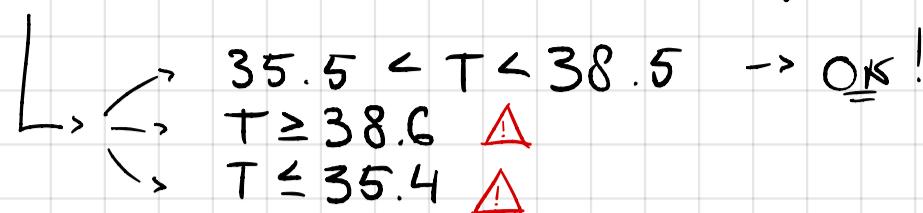
(in risposta a determinati valori)

- PROCESSA le RICHIESTE VENUTE
- inviate da TERMINALE

## SENSORS MQTT NETWORK

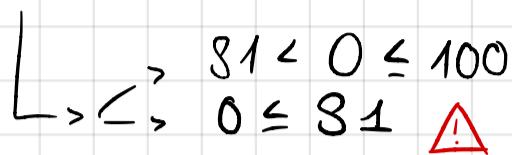
- TEMPERATURA : GENERA VALORI  $T \in [35, 45]$

TOPIC = TEMPERATURA



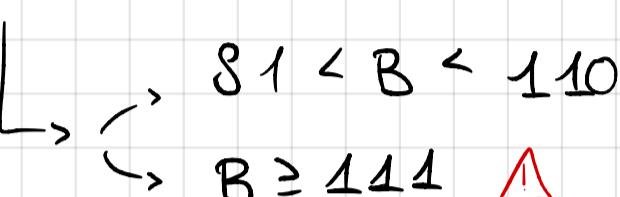
- OSSIGENO : GENERA VALORI  $O \in [65, 100]$

TOPIC = OSSIGENO

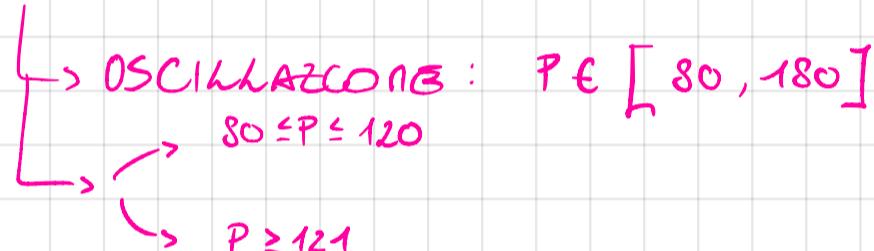


- BATTITO : GENERA VALORI  $B \in [70, 150]$

TOPIC = BATTITO



IDEA: ① Considerare come valore la  
PRESSIONE ARTERIOSA



## ACTUATORS COAP NETWORK

- PER PRIMA COSA DOVRAO FARSI COAP REGISTRATION SERVER



AFFINCHÉ UN ATTUATORE SIA RAGGIUNGIBILE  
DALLA REMOTE CONTROL APPLICATION

↳ ALL'UTIZIO NON È A CONOSCENZA  
DELL'INDIRIZZO IPV6

- REGISTRAZIONE : OGNI ATTUATORE MANDA IN JSON

UN KEY-VALUE =  $\{ \text{"type": "gas"} \}$

- MASK →  $O \leq 81$  AND  $B \geq 111$



MASK ATTIVA = LED BLUE

↗ NOTIFICA IL PERICOLO

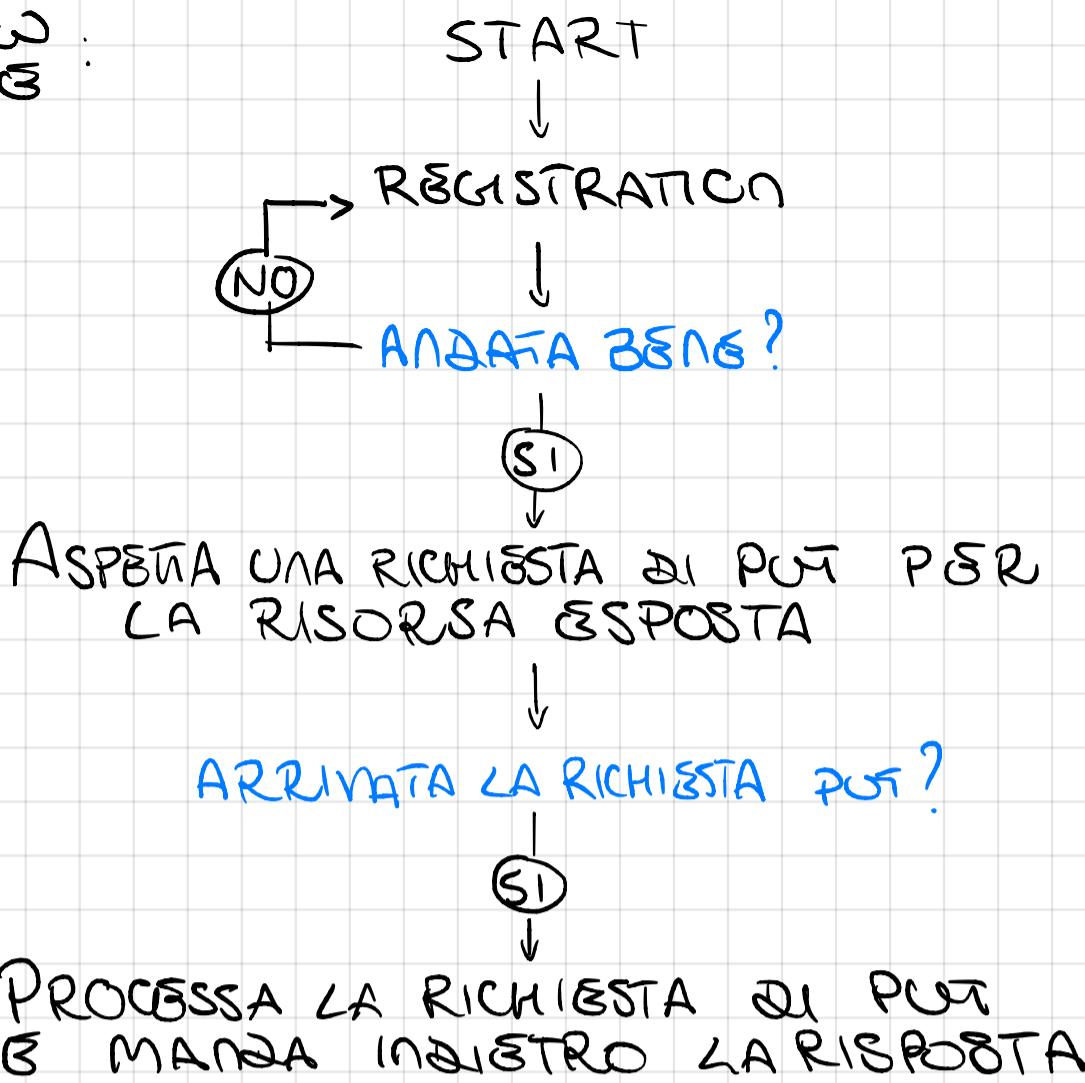
- PIU DISPENSER →  $(O \leq 90$  AND  $B \geq 95)$  OR  $(T > 38.5)$



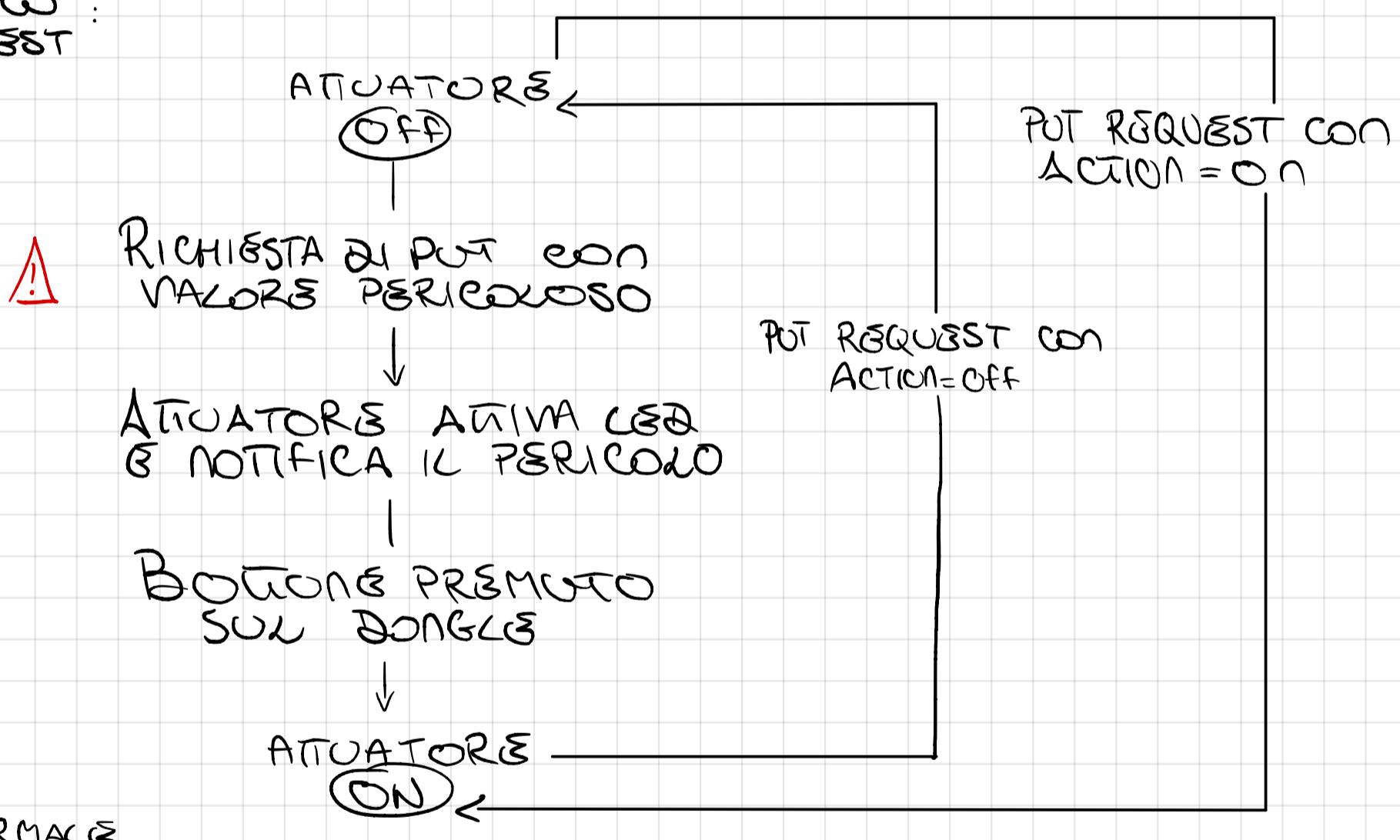
PIU DISP. ATTIVO = LED ROSSO

↗ NOTIFICA IL PERICOLO

## WORKFLOW ATTUATORE:



## WORKFLOW PUT REQUEST:



VALORE ANORMALE

↳ ATTUATORE = LED ATTIVO ACCESO

↳ BOTTONE = "OK, HO CAPITO CHE C'È UN PERICOLO.  
PREMO UN BOTTONE PER SEGNALARE  
CHE HO RICEVUTO IL MESSAGGIO"



## ① PERIODIC DATA RETRIEVAL

- ↳ STORIAGE SOGGEZ DI THRESHOLD
- ↳ RUN :
  - RECUPERA I DATI PIÙ RECENTI PER MONITORARE EVENTUALI SITUAZIONI DI PERICOLO
    - ↳ VALORES OK => ATTUATORI OFF (RISPARMIO ENERGETICO)
    - ↳ VALORES >ε => ATTUATORI ON + LED ACCESO
    - ↳ VALORES >ε - (30% ε) => ATTUATORI ON + LED SPENTO

## ② COAP Client

- ↳ STORIAGE INFORMAZIONI
- ↳ MANDA COMANDI AGGI ATTUATORI
  - ↓
  - COAP PUT REQUEST => ON/OFF

## ③ COMMAND LINE INTERFACE

- ↳ PROCESSARE LE RICHIESTE UTENTE
  - 1. HELP
  - 2. CAMBIO THRESHOLD
  - 3. CONTROLLO STATO
  - 4. CAMBIO COERCIVO STATO ATTUATORI

## CLOUD APPLICATION

① MQTT COLLECTOR

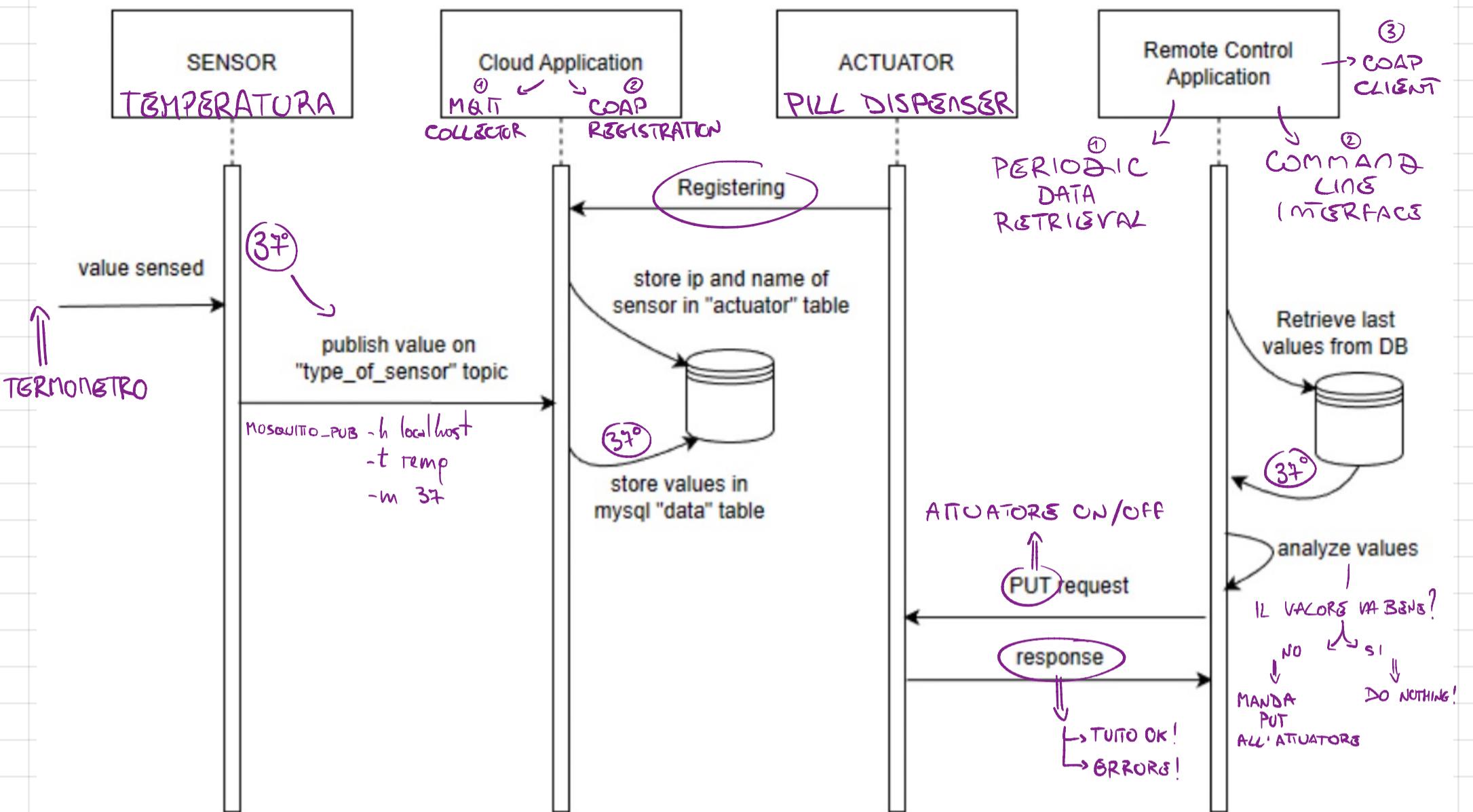
② CLOUD REGISTRATION

### ① MQTT COLLECTOR

- ↳ AGISCE COME SUBSCRIBER SUCC CODI DEI 3 SENSORI CHE VENGONO MANDATE LE MISURAZIONI
- ↳ DATO RICEVUTO => METODO MESSAGE ARRIVED
  - ↳ PRENDE IL PAYLOAD
    - ↓
    - SALVA L'INFORMAZIONE SUL DATABASE

### ② COAP REGISTRATION

- ↳ SERVER CHE ESPONE LA RISORSA PER REGISTRARSI
  - È IN ASCOLTO ASPETTANDO REGISTRAZIONI
  - ↓
  - METODO POST (L'IP VIENE RECUPERATO DALLA RICHIESTA)
    - ↓
    - CREA UNA NUOVA RISORSA CON STATO INIZIALE OFF
- ↳ RIMANE ATTIVO PER TUTTA LA DURATA DELL'APPLICAZIONE



## COSE DA FARE

- SENSORI MQT $\bar{U}$   $\xleftrightarrow{\quad}$  { TEMPERATURA  
OSSIGENO  
BATITTO } VANO PERFEZIONATI
  - ↳ . VALORI SENSATI
  - ↳ . CONTROLLO SOGLIE THRESHOLD
  - ↳ . INSERIRE UNITÀ DI MISURA
- CLOUD APPLICATION  $\stackrel{(1)}{\equiv}$  MQTT COLLECTOR  
 $\downarrow$   
VA PERFEZIONATO
  - ↳ . VALORI SENSATI
  - ↳ . CONTROLLO SOGLIE THRESHOLD
  - ↳ . INSERIRE UNITÀ DI MISURA

⚠ COLLEGARE I 2 CONNECTORS AD UN DATABASE

- $\left[ \begin{array}{l} \rightarrow \text{STORARE I DATI DEL DATABASE} \\ \rightarrow \text{STORARE LE INFORMAZIONI CIRCA UNA RISORSA} \end{array} \right]$
- CLOUD APPLICATION  $\stackrel{(2)}{\equiv}$  CoAP REGISTRATION  
 $\downarrow$ 
  - ↳ VA FATTO  $\rightarrow$  DEVE POTER ESPORRE UNA RISORSA (METODO PUT)  
PER REGISTRARSI
    - ↳ IP
    - ↳ Nome Attivatore

NEXT  $\rightarrow$  REMOTE CONTROL APPLICATION

## DATABASE Access

```
public class DatabaseAccess {
    private static final String url = "jdbc:mysql://localhost:3306/smartsuit";
    private static final String username = "root";
    private static final String password = "ubuntu"; } DATABASE CONNECTION DETAILS

    public static int updateActuators(String address, String actuatorType, String status) throws SQLException {
        Connection connection = DriverManager.getConnection(url, username, password);
        PreparedStatement ps = connection.prepareStatement(sql:"REPLACE INTO actuators (ip, actuator_type, status) VALUES(?, ?, ?);");
        ps.setString(parameterIndex:1, address.substring(beginIndex:1)); //substring(1)
        ps.setString(parameterIndex:2, actuatorType);
        ps.setString(parameterIndex:3, status);
        ps.executeUpdate();
        return ps.getUpdateCount();
    }

    public static HashMap<String, String> retrieveActuator(String actuatorType) throws SQLException {
        HashMap<String, String> result = new HashMap<>();
        Connection connection = DriverManager.getConnection(url, username, password);
        PreparedStatement ps = connection.prepareStatement(sql:"SELECT ip, status FROM actuators WHERE actuator_type = ?");
        ps.setString(parameterIndex:1, actuatorType);
        ResultSet rs = ps.executeQuery();
        if(!rs.next()){
            return result; //empty at this point
        }else {

            result.put(key:"ip", rs.getString(columnLabel:"ip"));
            result.put(key:"status", rs.getString(columnLabel:"status"));
            rs.close();

            return result;
        }
    }
}
```

### • UPDATE ACTUATORS (IP, TIPO\_ATT, STATO\_ATT)

Aggiorna lo stato di un attivatore

- Si stabilisce una connessione al Database
- PreparedStatement è creato per eseguire una query SQL che inserisce/sostituisce nuovi dati nella tabella ACTUATORS