

Internet delle Cose

IoT-Internet of Things

<https://bit.ly/ragnoIdC>



Perché IoT?

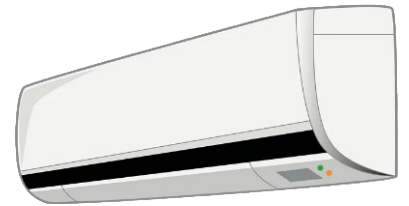
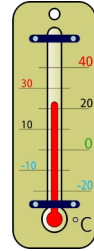
gestione/controllo di un ambiente di interesse



Sensori ed attuatori

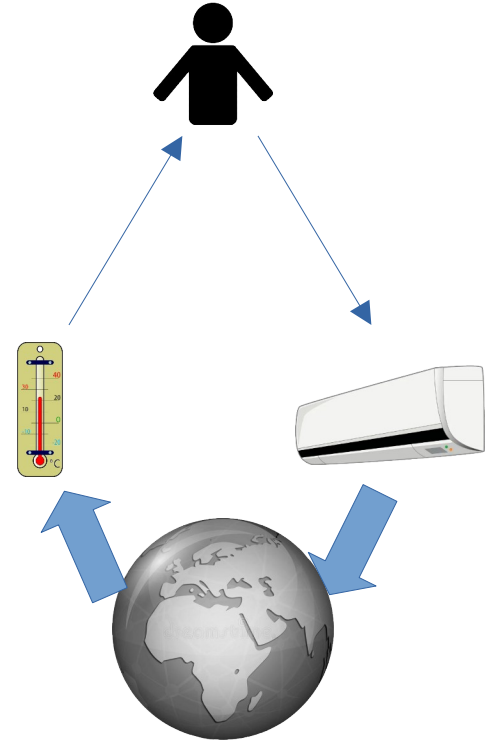
Per la gestione/controllo di un ambiente servono:

- Sensori che forniscono dati sullo stato dell'ambiente
- Attuatori che permettono di agire modificando lo stato dell'ambiente

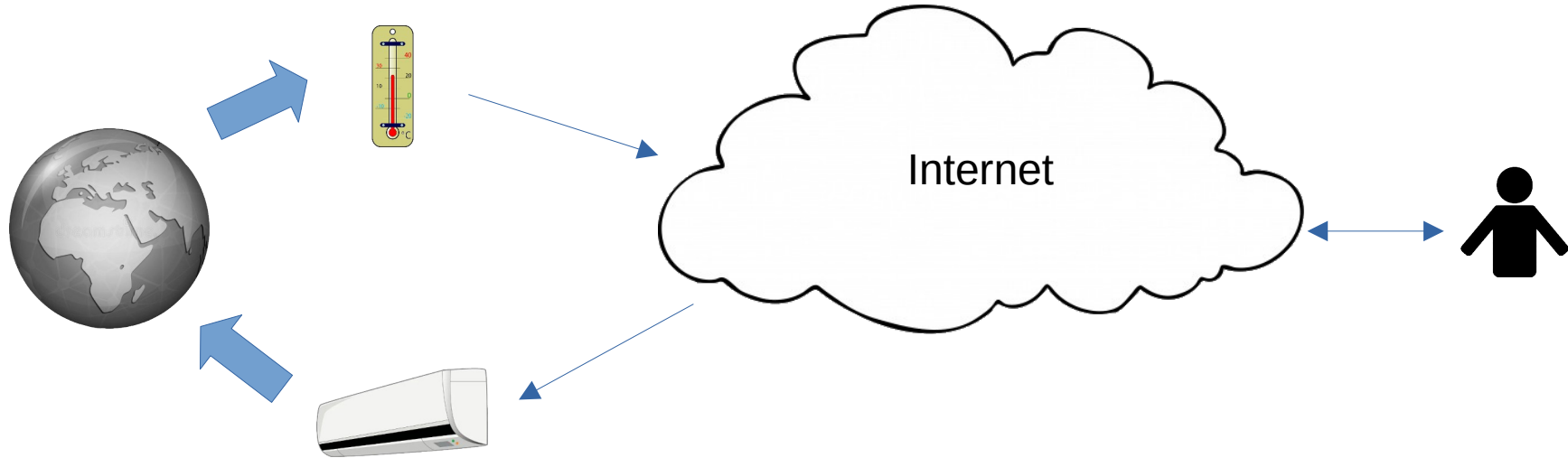


Catena di controllo

- Ambiente
- Sensore
- Controllore
- Attuatore
- Ambiente

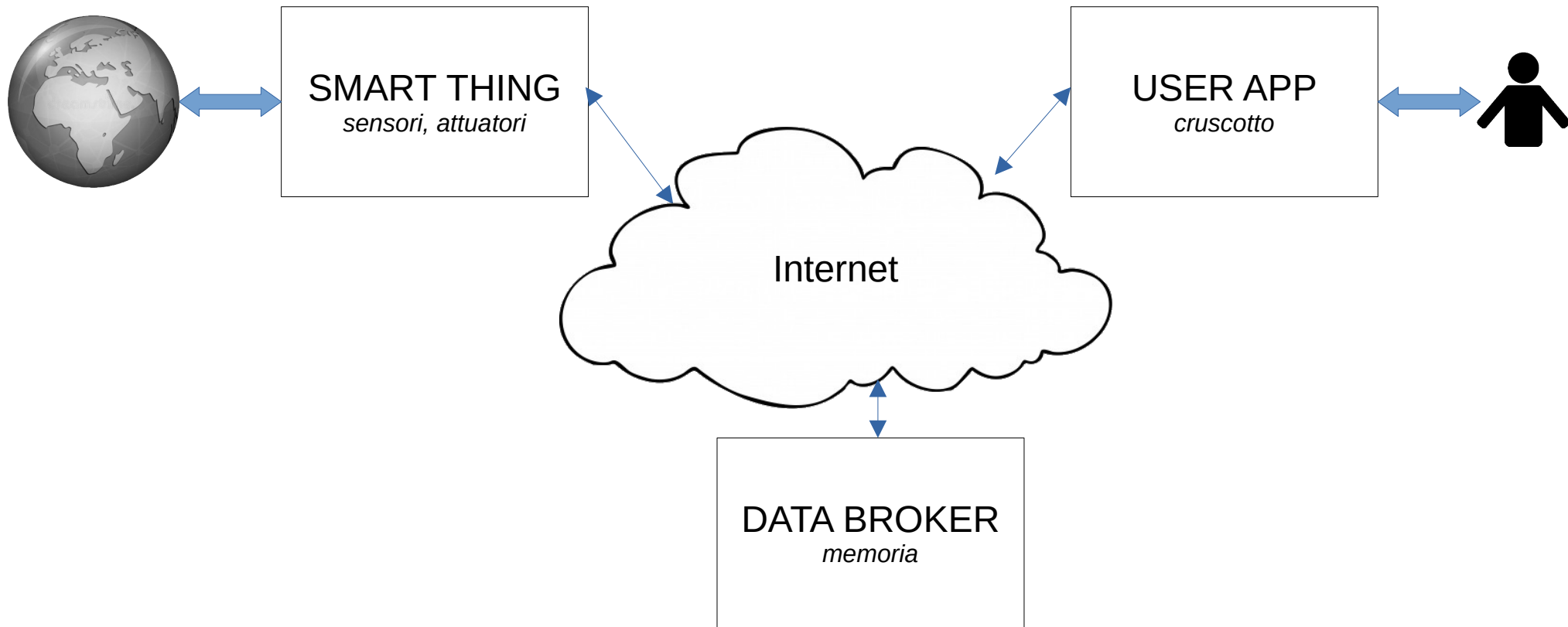


Controllo remoto



Sono necessari dispositivi di raccolta dati e/o comunicazione

Catena IoT



Architettura a tre livelli

Three-tier architecture



Broker

<https://www.schoolmakerday.it/broker/>

- Colloquia con chiamate http(s) che realizzano:
 - set : memorizza una coppia chiave-valore
 - get : restituisce il valore associato ad una chiave (se presente)
- Ad uso didattico, non prevede strumenti di protezione di dati personali

Broker – API set

<https://www.schoolmakerday.it/broker/set.php?key=KKK&value=VVV>

- registrazione da parte del server di dati inviati in forma di coppie chiave-valore
- **key** string max 10 caratteri, **value** string max 255 car.
- viene aggiunto il timestamp (data e ora)
- restituisce oggetto JSON con i dati ottenuti da una rilettura dei dati inseriti:
 - **status**: deve avere valore 'OK', altrimenti c'è stato un errore
 - (solo se status OK) **data**: oggetto con i campi "key","value","ts"

Broker – API get

<https://www.schoolmakerday.it/broker/get.php?key=KKK>

- restituisce il valore più recente associato alla chiave fornita, in forma di oggetto JSON con campi
 - **status**: deve avere valore 'OK', altrimenti c'è stato un errore
 - (solo se status OK) **data**: oggetto con i campi "key", "value", "ts" o null se chiave mancante.

Dashboard

<https://www.schoolmakerday.it/broker/dashboard/>

- app per la consultazione/modifica di chiavi e valori su broker
- utilizzabile online o installabile su device (Progressive Web App per chrome/edge)

SMD Broker Dashboard

Gestione multipla di chiavi

Per la gestione di chiavi su broker inserire nel form qui sotto una o più chiavi di proprio interesse, tenendo presente che per chiavi destinate a comando è necessario indicarle come editabili (senza spunta a ReadOnly), mentre per chiavi destinate a stato è bene renderle non modificabili (spuntando ReadOnly)

Inserisci una chiave da aggiungere in dashboard:

Chiave: ☐ ReadOnly

Server time: 08:27:24

Realizzato per SchoolMakerDay EDU 2023
Copyright CC BY-SA 4.0

Smart Thing

Realizziamo un semplice dispositivo dotato di:

- led
- sensore di temperatura

con cui interagiremo a distanza utilizzando broker e dashboard

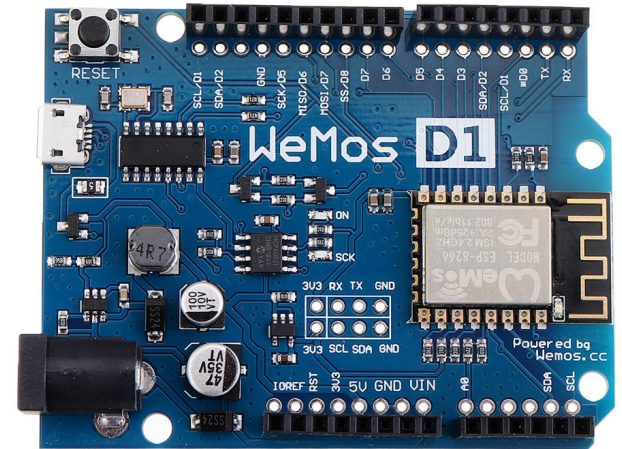
Microcontrollore ESP8266

- Wi-Fi integrato con supporto al protocollo TCP/IP
- Processore 32 bit
- 64 KiB di RAM istruzioni, 96 KiB di RAM dati
- Programmabile:
 - C++ (Arduino compatibile)
 - MicroPython
 - altro...



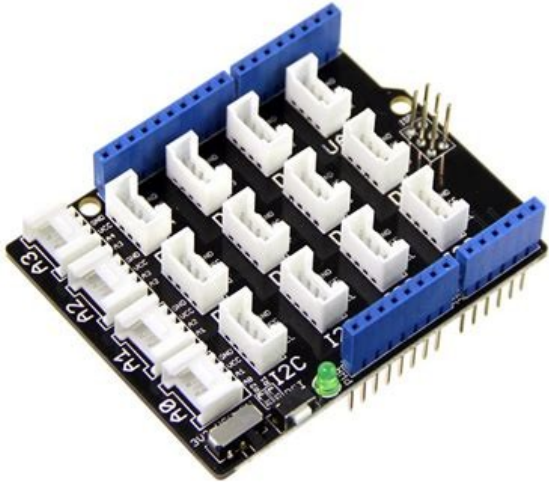
WeMosD1

- Scheda hardware-compatibile con Arduino equipaggiata con ESP8266

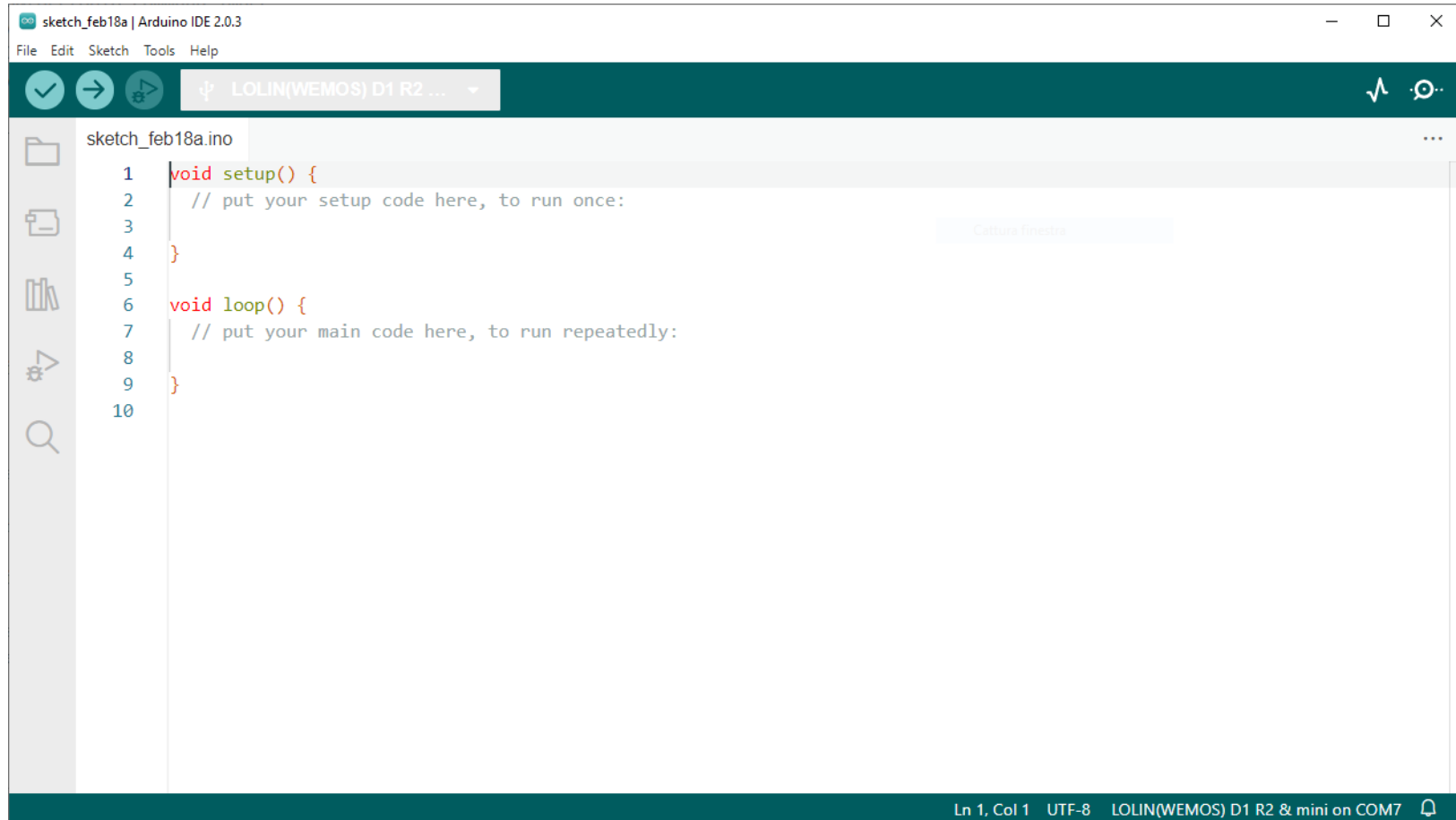


Grove

- Kit di dispositivi con connessione semplificata compatibile con Arduino



Arduino IDE



Step 0 - Struttura base del software

```
void setup() {  
    //attiva il canale seriale  
    initSerial();  
}  
  
void loop() {  
  
    //aggiorna comandi da broker  
  
    //campiona stato  
  
    //azioni coerenti ai comandi/stati  
  
    //le azioni potrebbero aver modificato lo stato, aggiorna stato  
  
    //se c'è una modifica di stato aggiorna in locale e su broker  
  
    //attesa sospensiva  
    //in un progetto realtime occorre una attesa senza sospensione  
    delay(1000);  
}
```

```
/**  
 * inizializzazione del canale seriale  
 * @return void  
 */  
void initSerial(){  
    Serial.begin(SERIAL_BAUD_RATE);  
  
    Serial.println();  
    Serial.println();  
    Serial.println();  
  
    for (uint8_t t = 4; t > 0; t--) {  
        Serial.printf("[main] [SERIAL SETUP] WAIT %d...\n", t);  
        Serial.flush();  
        delay(1000);  
    }  
}
```

L'uso del canale seriale è previsto per la diagnostica

Step 1 – blink (locale)

```
// librerie -----  
#include <Arduino.h>  
// classi del progetto -----  
#include "Led.h"  
// configurazione -----  
//velocità per la seriale  
#define SERIAL_BAUD_RATE 115200  
// pin dei dispositivi  
#define LED_PIN 4  
// fine configurazione -----  
// oggetti di interazione con i dispositivi  
Led led(LED_PIN); //si occupa del led in comando e lettura di stato  
// -----
```

```
void setup() {  
    //attiva il canale seriale  
    initSerial();  
    led.off();  
}  
void loop() {  
    //aggiorna comandi da broker  
    //campiona stato  
    //azioni coerenti ai comandi/stati  
    azioneBlink();  
    //le azioni potrebbero aver modificato lo stato, aggiorna stato  
    //se c'è una modifica di stato aggiorna in locale e su broker  
    //attesa sospensiva  
    //in un progetto realtime occorre una attesa senza sospensione  
    delay(1000);  
}  
/**  
 * azione di controllo blink, ad ogni chiamata inverte lo stato del led  
 * @return void  
 */  
void azioneBlink(){  
    if (led.isOn()){  
        led.off();  
    }  
    else {  
        led.on();  
    }  
}
```

Step 2 – blink con notifica al broker

```
// librerie -----
#include <Arduino.h>
// classi del progetto -----
#include "Led.h"
#include "Network.h"
#include "Broker.h"
// configurazione -----
// ssid e password per wifi
#ifndef STASSID
#define STASSID "SSID"
#define STAPSK "PASSWORD"
#endif
//velocità per la seriale
#define SERIAL_BAUD_RATE 115200
// pin dei dispositivi
#define LED_PIN 4
// chiavi da utilizzare nel broker
String BASE_KEY = "idc";
String STA_ON_KEY = BASE_KEY+"s_on"; //chiave per lo stato del l
// fine configurazione -----
// oggetti di interazione con i dispositivi
Led led(LED_PIN); //si occupa del led in comando e lettura di sta
Network net(STASSID,STAPSK); //si occupa della connessione di ret
Broker broker(&net); //si occupa del colloquio con il broker e de
// variabili per immagine di comando e stato-----
String staOn=""; // ON/OFF status image
// -----
```

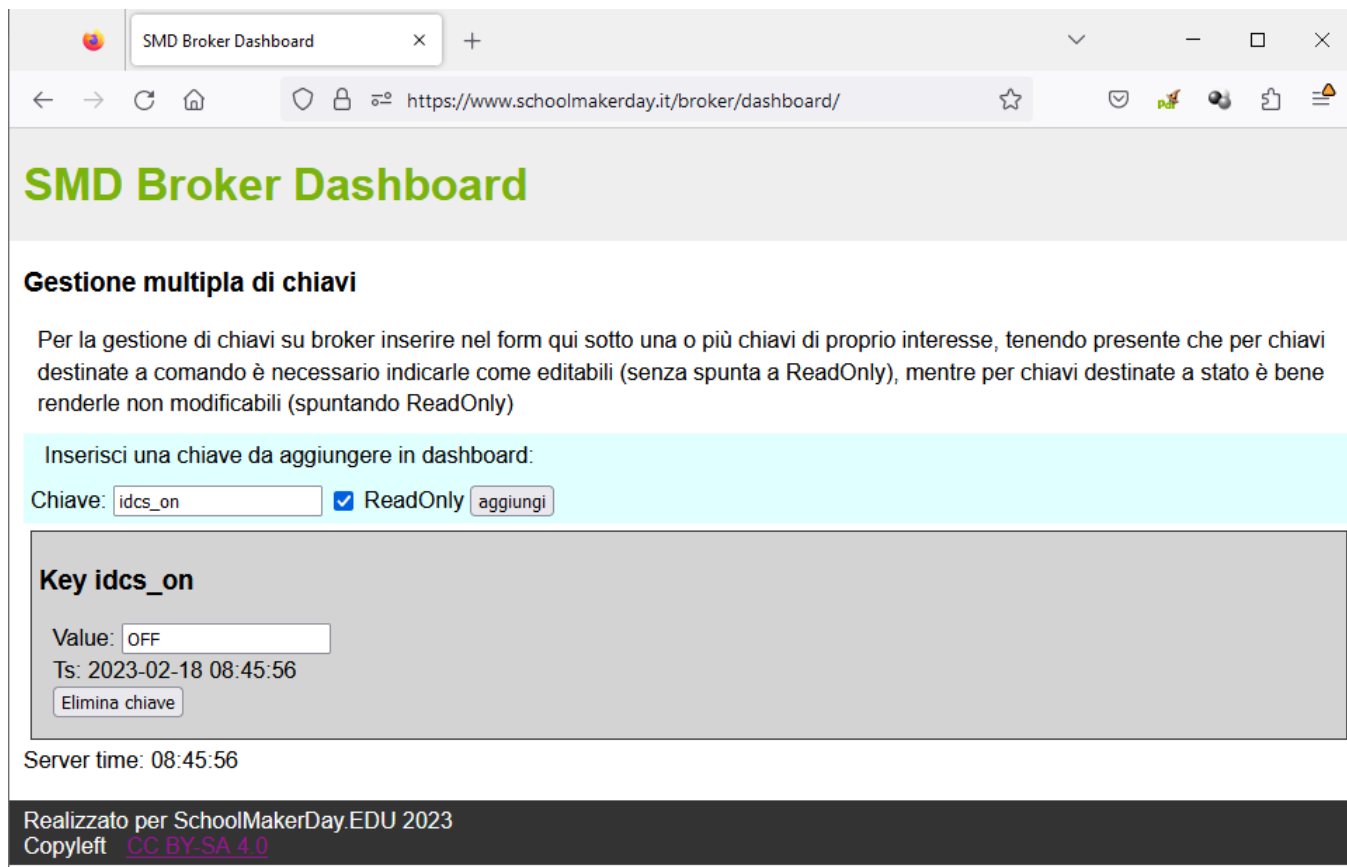
```
void setup() {
    //attiva il canale seriale
    initSerial();
    led.off();
    staOn=aggiornaStaOn();
    broker.set(STA_ON_KEY, staOn);
}

void loop() {
    //aggiorna comandi da broker
    //campiona stato
    String newStaOn=aggiornaStaOn();
    //azioni coerenti ai comandi/stati
    azioneBlink(newStaOn);
    //le azioni potrebbero aver modificato lo stato, aggiorna stato
    newStaOn=aggiornaStaOn();
    //se c'è una modifica di stato aggiorna in locale e su broker
    if (newStaOn!=staOn){
        staOn=newStaOn;
        broker.set(STA_ON_KEY, staOn);
    }
    //attesa sospensiva
    //in un progetto realtime occorre una attesa senza sospensione
    delay(1000);
}
```

```
/**
 * azione di controllo blink, ad ogni chiamata inverte lo stato del led
 * @param sta stato corrente
 * @return void
 */
void azioneBlink(String sta){
    if (sta=="ON"){
        led.off();
    }
    else {
        led.on();
    }
}

/**
 * richiede all'oggetto led il suo stato
 * e ne restituisce stringa corrispondente
 * @return String
 */
String aggiornaStaOn(){
    if (led.isOn()){
        return "ON";
    }
    return "OFF";
}
```

Step 2 / dashboard



The screenshot shows a web browser window with the title "SMD Broker Dashboard". The address bar shows the URL "https://www.schoolmakerday.it/broker/dashboard/". The page content includes a green header "SMD Broker Dashboard", a section titled "Gestione multipla di chiavi", and a form for adding a new key. The form has a text input for the key name (containing "idcs_on"), a checked checkbox for "ReadOnly", and an "aggiungi" button. Below the form, a table displays the details of the added key: "Key idcs_on", "Value: OFF", "Ts: 2023-02-18 08:45:56", and an "Elimina chiave" button. The server time "08:45:56" is shown at the bottom. The footer contains the text "Realizzato per SchoolMakerDay.EDU 2023" and "Copyleft CC BY-SA 4.0".

SMD Broker Dashboard

Gestione multipla di chiavi

Per la gestione di chiavi su broker inserire nel form qui sotto una o più chiavi di proprio interesse, tenendo presente che per chiavi destinate a comando è necessario indicarle come editabili (senza spunta a ReadOnly), mentre per chiavi destinate a stato è bene renderle non modificabili (spuntando ReadOnly)

Inserisci una chiave da aggiungere in dashboard:

Chiave: ☒ ReadOnly

Key
Key idcs_on
Value: <input type="text" value="OFF"/>
Ts: 2023-02-18 08:45:56
<input type="button" value="Elimina chiave"/>

Server time: 08:45:56

Realizzato per SchoolMakerDay.EDU 2023
Copyleft [CC BY-SA 4.0](#)

Step 3 – comando da remoto

```
// librerie -----
#include <Arduino.h>
// classi del progetto -----
#include "Led.h"
#include "Network.h"
#include "Broker.h"
// configurazione -----
// ssid e password per wifi
#ifndef STASSID
#define STASSID "SSID"
#define STAPSK "PASSWORD"
#endif
//velocità per la seriale
#define SERIAL_BAUD_RATE 115200
// pin dei dispositivi
#define LED_PIN 4
// chiavi da utilizzare nel broker
String BASE_KEY = "idc";
String CMD_ON_KEY = BASE_KEY+"c_on"; //chiave per il comando
String STA_ON_KEY = BASE_KEY+"s_on"; //chiave per lo stato
// fine configurazione -----
// abilitazione ai messaggi su seriale per il debug
#define MAIN_DEBUG true
// oggetti di interazione con i dispositivi
Led led(LED_PIN); //si occupa del led in comando e lettura
Network net(STASSID,STAPSK); //si occupa della connessione
Broker broker(&net); //si occupa del colloquio con il broker
// variabili per immagine di comando e stato-----
String cmdOn=""; // ON/OFF/AUTO command image
String staOn=""; // ON/OFF status image
```

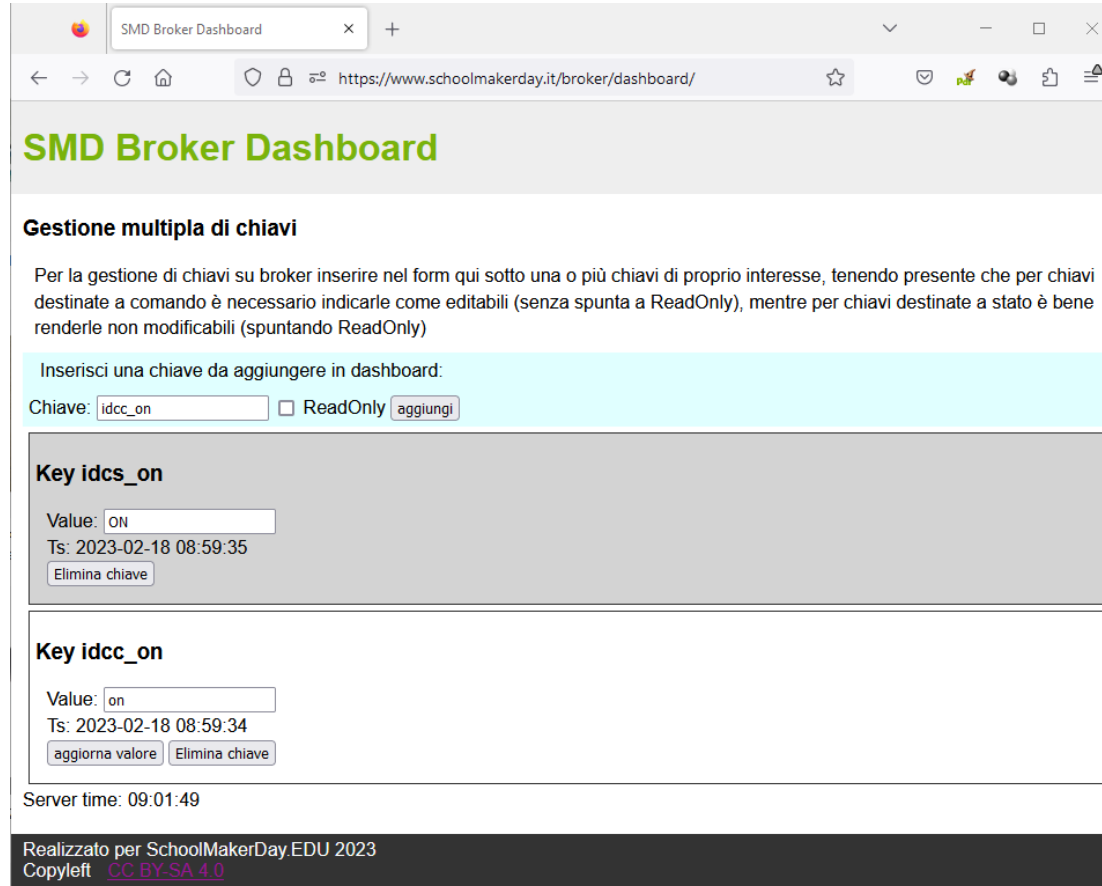
```
// -----
void setup() {
    //attiva il canale seriale
    initSerial();
    led.off();
    staOn=aggiornaStaOn();
    broker.set(STA_ON_KEY, staOn);
}
void loop() {
    //aggiorna comandi da broker
    aggiornaCmdOn();
    //campiona stato
    String newStaOn=aggiornaStaOn();
    //azioni coerenti ai comandi/stati
    azioneOnOff(newStaOn);
    //le azioni potrebbero aver modificato lo stato, aggiorna stato
    newStaOn=aggiornaStaOn();
    //se c'è una modifica di stato aggiorna in locale e su broker
    if (newStaOn!=staOn){
        staOn=newStaOn;
        broker.set(STA_ON_KEY, staOn);
    }
    //attesa sospensiva
    //in un progetto realtime occorre una attesa senza sospensione
    delay(1000);
}
```

```
/**
 * richiede al broker il valore di cmdOn
 * effettua verifica di validità
 * @return void
 */
void aggiornaCmdOn(){
    String newCmdOn=broker.get(CMD_ON_KEY);
    newCmdOn.toUpperCase();//converte in tutto maiuscolo
    if (
        newCmdOn=="ON" ||
        newCmdOn=="OFF"
    ) {
        //comando valido
        cmdOn=newCmdOn;
        if (MAIN_DEBUG) {
            Serial.println("[main] CmdOn: '"+newCmdOn+"'");
        }
    } else {
        // comando sconosciuto, ignora
        if (MAIN_DEBUG) {
            Serial.println("[main] CmdOn sconosciuto: '"+newCmdOn+"'");
        }
    }
}
/**
 * azione di controllo diretto on/off
 * @param newStaOn stato corrente
 * @return String
 */
void azioneOnOff(String newStaOn){
    //verifico se stato è coerente a comando
    if (newStaOn!=cmdOn){
        //devo agire
        if (cmdOn=="ON"){
            led.on();
        }
        else {
            led.off();
        }
    }
    // else nothing to do!
}
```

Step 3 /serial monitor

```
[main] CmdOn: 'ON'
[broker] deserializeJson() success:
      status: OK
      key: idcc_on
      val: on
      ts: 2023-02-18 08:59:34
[main] CmdOn: 'ON'
[broker] deserializeJson() success:
      status: OK
      key: idcc_on
      val: on
      ts: 2023-02-18 08:59:34
[main] CmdOn: 'ON'
[broker] deserializeJson() success:
      status: OK
      key: idcc_on
```

Step 3 /dashboard



The screenshot shows a web browser window with the title "SMD Broker Dashboard". The address bar shows the URL "https://www.schoolmakerday.it/broker/dashboard/". The page content includes a green header "SMD Broker Dashboard", a section "Gestione multipla di chiavi" with explanatory text, a form to add a new key, and two panels for existing keys: "Key idcs_on" and "Key idcc_on". The server time is displayed as 09:01:49. The footer mentions "Realizzato per SchoolMakerDay.EDU 2023" and "Copyleft CC BY-SA 4.0".

SMD Broker Dashboard

Gestione multipla di chiavi

Per la gestione di chiavi su broker inserire nel form qui sotto una o più chiavi di proprio interesse, tenendo presente che per chiavi destinate a comando è necessario indicarle come editabili (senza spunta a ReadOnly), mentre per chiavi destinate a stato è bene renderle non modificabili (spuntando ReadOnly)

Inserisci una chiave da aggiungere in dashboard:

Chiave: ☐ ReadOnly

Key idcs_on

Value:
Ts: 2023-02-18 08:59:35

Key idcc_on

Value:
Ts: 2023-02-18 08:59:34

Server time: 09:01:49

Realizzato per SchoolMakerDay.EDU 2023
Copyleft [CC BY-SA 4.0](#)

Step 4 – sensore di temperatura

```
// librerie -----
#include <Arduino.h>
// classi del progetto -----
#include "Led.h"
#include "TempSensor.h"
#include "Network.h"
#include "Broker.h"
// configurazione -----
// ssid e password per wifi
#ifdef STASSID
#define STASSID "SSID"
#define STAPSK "PASSWORD"
#endif
//velocità per la seriale
#define SERIAL_BAUD_RATE 115200
// pin dei dispositivi
#define LED_PIN 4
#define TEMP_SENSOR_PIN A0
// chiavi da utilizzare nel broker
String BASE_KEY = "idc";
String CMD_ON_KEY = BASE_KEY+"c_on"; //chiave per il com
String STA_ON_KEY = BASE_KEY+"s_on"; //chiave per lo st
String STA_TEMP_KEY = BASE_KEY+"s_temp"; //chiave per l
// fine configurazione -----
// abilitazione ai messaggi su seriale per il debug
#define MAIN_DEBUG true
// oggetti di interazione con i dispositivi
Led led(LED_PIN); //si occupa del led in comando e lettura
TempSensor ts(A0); //si occupa del sensore di temperatura
Network net(STASSID,STAPSK); //si occupa della connessione
Broker broker(&net); //si occupa del colloquio con il broker
// variabili per immagine di comando e stato-----
String cmdOn=""; // ON/OFF command image
String staOn=""; // ON/OFF status image
String staTemp=""; // Temperature status image
// -----
```

```
void setup() {
    //attiva il canale seriale
    initSerial();
    led.off();
    staOn=aggiornaStaOn();
    broker.set(STA_ON_KEY, staOn);
    staTemp=aggiornaStaTemp();
    broker.set(STA_TEMP_KEY, staTemp);
}

void loop() {
    //aggiorna comandi da broker
    aggiornaCmdOn();
    //campiona stato
    String newStaOn=aggiornaStaOn();
    String newStaTemp=aggiornaStaTemp();
    //azioni coerenti ai comandi/stati
    azioneOnOff(newStaOn);
    //le azioni potrebbero aver modificato lo stato, aggiorna stato
    newStaOn=aggiornaStaOn();
    //se c'è una modifica di stato aggiorna in locale e su broker
    if (newStaOn!=staOn){
        staOn=newStaOn;
        broker.set(STA_ON_KEY, staOn);
    }
    if (newStaTemp!=staTemp){
        staTemp=newStaTemp;
        broker.set(STA_TEMP_KEY,staTemp);
    }
    //attesa sospensiva
    //in un progetto realtime occorre una attesa senza sospensione
    delay(1000);
}
```

```
/**
 * richiede all'oggetto ts il suo stato
 * e ne restituisce il valore con 1 decimale
 * @return String
 */
String aggiornaStaTemp(){
    float t = ts.getTemperature();
    if (MAIN_DEBUG) {
        Serial.println("[main] Temperatura = "+String(t));
    }
    return String(t,1);
}
```


Step 4 /serial monitor

```
    val: on
    ts: 2023-02-18 08:59:34
[main] CmdOn: 'ON'
[main] Temperatura = 22.50
[broker] deserializeJson() success:
    status: OK
    key: idcc_on
    val: on
    ts: 2023-02-18 08:59:34
[main] CmdOn: 'ON'
[main] Temperatura = 22.50
[broker] deserializeJson() success:
    status: OK
    key: idcc_on
    val: on
    ts: 2023-02-18 08:59:34
[main] CmdOn: 'ON'
[main] Temperatura = 22.50
```

Step 4 / dashboard

The screenshot shows a web browser window with the title "SMD Broker Dashboard". The address bar shows the URL "https://www.schoolmakerday.it/broker/dashboard/". The page content includes a green header "SMD Broker Dashboard", a section "Gestione multipla di chiavi" with explanatory text, a form to add a new key, and three key entries: "idcs_on", "idcc_on", and "idcs_temp". Each entry shows its value, timestamp, and action buttons. The footer contains the server time and copyright information.

SMD Broker Dashboard

Gestione multipla di chiavi

Per la gestione di chiavi su broker inserire nel form qui sotto una o più chiavi di proprio interesse, tenendo presente che per chiavi destinate a comando è necessario indicarle come editabili (senza spunta a ReadOnly), mentre per chiavi destinate a stato è bene renderle non modificabili (spuntando ReadOnly)

Inserisci una chiave da aggiungere in dashboard:

Chiave: ☒ ReadOnly

Key idcs_on

Value:
Ts: 2023-02-18 09:13:18

Key idcc_on

Value:
Ts: 2023-02-18 08:59:34

Key idcs_temp

Value:
Ts: 2023-02-18 09:19:58

Server time: 09:19:58

Realizzato per SchoolMakerDay.EDU 2023
Copyleft [CC BY-SA 4.0](#)

Step 5 – controllo di temperatura

```
// librerie -----
#include <Arduino.h>
// classi del progetto -----
#include "Led.h"
#include "TempSensor.h"
#include "Network.h"
#include "Broker.h"
// configurazione -----
// ssid e password per wifi
#ifndef STASSID
#define STASSID "SSID"
#define STAPSK "PASSWORD"
#endif
//velocità per la seriale
#define SERIAL_BAUD_RATE 115200
// pin dei dispositivi
#define LED_PIN 4
#define TEMP_SENSOR_PIN A0
// chiavi da utilizzare nel broker
String BASE_KEY = "gr";
String CMD_ON_KEY = BASE_KEY+"c_on"; //chiave per il c
String STA_ON_KEY = BASE_KEY+"s_on"; //chiave per lo
String STA_TEMP_KEY = BASE_KEY+"s_temp"; //chiave per
String CMD_TEMP_KEY = BASE_KEY+"c_temp"; //chiave per
// fine configurazione -----
// abilitazione ai messaggi su seriale per il debug
#define MAIN_DEBUG true
// oggetti di interazione con i dispositivi
Led led(LED_PIN); //si occupa del led in comando e let
TempSensor ts(A0); //si occupa del sensore di temperat
Network net(STASSID,STAPSK); //si occupa della conness
Broker broker(&net); //si occupa del colloquio con il
// variabili per immagine di comando e stato-----
String cmdOn=""; // ON/OFF/AUTO command image
String cmdTemp=""; // Temperature command image
String staOn=""; // ON/OFF status image
String staTemp=""; // Temperature status image
```

```
void setup() {
    //attiva il canale seriale
    initSerial();
    led.off();
    staOn=aggiornaStaOn();
    broker.set(STA_ON_KEY, staOn);
    staTemp=aggiornaStaTemp();
    broker.set(STA_TEMP_KEY, staTemp);
}

void loop() {
    //aggiorna comandi da broker
    aggiornaCmdOn();
    aggiornaCmdTemp();
    //campiona stato
    String newStaOn=aggiornaStaOn();
    String newStaTemp=aggiornaStaTemp();
    //azioni coerenti ai comandi/stati
    if (cmdOn=="AUTO") {
        //il cmdOn AUTO richiede controllo della temperat
        azioneAuto(newStaTemp);
    }
    else {
        //è ON oppure OFF
        azioneOnOff(newStaOn);
    }
    //le azioni potrebbero aver modificato lo stato, ag
    newStaOn=aggiornaStaOn();
    //se c'è una modifica di stato aggiorna in locale e
    if (newStaOn!=staOn){
        staOn=newStaOn;
        broker.set(STA_ON_KEY, staOn);
    }
    if (newStaTemp!=staTemp){
        staTemp=newStaTemp;
        broker.set(STA_TEMP_KEY,staTemp);
    }
    //attesa sospensiva
    //in un progetto realtime occorre una attesa senza
    delay(1000);
}
```

```
/**
 * richiede al broker il valore di cmdTemp
 * effettua verifica di validità
 * @return void
 */
void aggiornaCmdTemp(){
    String newCmdTemp=broker.get(CMD_TEMP_KEY);
    //newCmdTemp deve essere un dato float
    //se non è float valido viene convertito a 0
    // se 0 è un valore ammissibile bisogna fare un check più raffi
    float t=newCmdTemp.toFloat();
    if (t!=0) {
        cmdTemp=t;
        if (MAIN_DEBUG) {
            Serial.println("[main] CmdTemp: '"+String(t)+"'");
        }
    }
    else {
        // comando sconosciuto, ignora
        if (MAIN_DEBUG) {
            Serial.println("[main] CmdTemp sconosciuto: '"+newCmdTemp+"'");
        }
    }
}

/**
 * azione di controllo automatico di temperature
 * @param newStaTemp stato corrente
 * @return String
 */
void azioneAuto(String newStaTemp){
    //confrontando interi ho isteresi di un grado
    int tEff=newStaTemp.toInt(); //temperatura attuale
    int tCmd=cmdTemp.toInt(); //temperatura richiesta per comando
    if (tEff>tCmd){
        //spegni
        led.off();
    }
    else if (tEff<tCmd){
        //accendi
        led.on();
    }
    // else nothing to do!
}
```

```
/**
 * richiede al broker il valore di cmdOn
 * effettua verifica di validità
 * @return void
 */
void aggiornaCmdOn(){
    String newCmdOn=broker.get(CMD_ON_KEY);
    newCmdOn.toUpperCase(); //converte in tutto maiuscolo
    if (
        newCmdOn=="ON" ||
        newCmdOn=="OFF" ||
        newCmdOn=="AUTO"
    ) {
        //comando valido
        cmdOn=newCmdOn;
        if (MAIN_DEBUG) {
            Serial.println("[main] CmdOn: '"+newCmdOn+"'");
        }
    }
    else {
        // comando sconosciuto, ignora
        if (MAIN_DEBUG) {
            Serial.println("[main] CmdOn sconosciuto: '"+newCmdOn+"'");
        }
    }
}
```

Step 5 / monitor seriale

```
        key: grc_temp
        val: 25
        ts: 2023-02-17 14:19:53
[main] CmdTemp: '25.00'
[main] Temperatura = 21.70
[broker] deserializeJson() success:
        status: OK
        key: grc_on
        val: auto
        ts: 2023-02-17 14:19:23
[main] CmdOn: 'AUTO'
[broker] deserializeJson() success:
        status: OK
        key: grc_temp
        val: 25
        ts: 2023-02-17 14:19:53
[main] CmdTemp: '25.00'
[main] Temperatura = 21.80
```

Step 5 / dashboard

SMD Broker Dashboard

https://www.schoolmakerday.it/broker/dashboard/

SMD Broker Dashboard

Gestione multipla di chiavi

Per la gestione di chiavi su broker inserire nel form qui sotto una o più chiavi di proprio interesse, tenendo presente che per chiavi destinate a comando è necessario indicarle come editabili (senza spunta a ReadOnly), mentre per chiavi destinate a stato è bene renderle non modificabili (spuntando ReadOnly)

Inserisci una chiave da aggiungere in dashboard:

Chiave: ☐ ReadOnly

Key idcs_on

Value:

Ts: 2023-02-18 09:13:18

Key idcc_on

Value:

Ts: 2023-02-18 09:37:26

Key idcs_temp

Value:

Ts: 2023-02-18 09:34:15

Key idcc_temp

Value:

Ts: 2023-02-18 09:37:01

Server time: 09:41:33

Realizzato per SchoolMakerDay EDU 2023
Copyleft [CC BY-SA 4.0](#)

Le classi di progetto

- Ci siamo concentrati sulla gestione di dati e messaggi senza entrare nello specifico dei componenti led e sensore di temperatura e della comunicazione network e broker
- Di seguito il dettaglio per chi vuole approfondire

Led

```
#ifndef LED_H
#define LED_H
/**
 * @file Led.h
 * @date 23 03 2022
 * @author Giovanni Ragno
 * @copyright https://creativecommons.org/licenses/by-sa/4.0/
 */
#include <Arduino.h>
/**
 * @class Led
 * @brief Manage an LED.
 * Managed actions: Off/On
 */
class Led{
protected:
| byte _pin; /**< @var pin number */
public:
/**
 * constructor
 * @param pin An Led needs a pin number
 */
Led(byte pin);
/**
 * Switch on the led using digitalWrite
 * @return void
 */
void on();
/**
 * Switch off the led
 * @return void
 */
void off();//digital off
/**
 * level getter
 * @return current level
 */
byte getLevel() const ;
/**
 * pin getter
 * @return pin number
 */
byte getPin() const ;
/**
 * abstract status getter
 * @return true if level >0
 */
boolean isOn() const ;
/**
 * abstract status getter
 * @return true if level==0
 */
boolean isOff() const ;
};
#endif
```

```
/**
 * @file Led.cpp
 * @date 23 03 2022
 * @author Giovanni Ragno
 * @copyright https://creativecommons.org/licenses/by-sa/4.0/
 */
#include "Led.h"
Led::Led(byte pin){
    // initialize the digital pin as an output.
    _pin=pin;
    pinMode(pin, OUTPUT);
    //initialize level OFF
    off();
}
void Led::on(){
    digitalWrite(_pin, HIGH);
}
void Led::off(){
    digitalWrite(_pin, LOW);
}
byte Led::getLevel() const {
    return digitalRead(_pin);
}
byte Led::getPin() const {
    return _pin;
}
boolean Led::isOn() const {
    return digitalRead(_pin)==HIGH;
}
boolean Led::isOff() const {
    return digitalRead(_pin)==LOW;
}
```

TempSensor

```
#ifndef TEMP_SENSOR_H
#define TEMP_SENSOR_H
/**
 * @file TempSensor.h
 * @date 04 02 2023
 * @author Giovanni Ragno
 * @copyright https://creativecommons.org/licenses/by-sa/4.0/
 */
#include <Arduino.h>
#define TEMP_FILTER_1 1
#define TEMP_FILTER_2 2
#define TEMP_FILTER_5 5
#define TEMP_FILTER_10 10
/**
 * @class TempSensor
 * @brief Gestisce il sensore di temperatura Grove.
 * Azioni: lettura di temperatura corrente raw o filtrata
 */
class TempSensor{
protected:
    byte _pin; /**< @var numero di pin */
    byte _filter; /**< @var base del filtro */
public:
    /**
     * constructor
     * @param pin Il sensore di temperatura richiede un pin analogico
     */
    TempSensor(byte pin);
    /**
     * Imposta il filtro
     * si raccomanda l'uso delle costanti TEMP_FILTER, altri valori saranno ignorati
     * @param f valori ammessi : 1,2,5,10 (decimi) di arrotondamento
     */
    void setFilter(byte f);
    /**
     * Restituisce il filtro
     * @return byte valori ammessi : 1,2,5,10 (decimi) di arrotondamento
     */
    byte getFilter();
    /**
     * Campiona la temperatura corrente
     * @return float temperatura corrente grezza
     */
    float getRawTemperature();//
    /**
     * Campiona e filtra la temperatura
     * @return float temperatura corrente arrotondata in base al valore di _filter
     */
    float getTemperature();//
};
#endif
```

```
/**
 * @file TempSensor.cpp
 * @date 04 02 2023
 * @author Giovanni Ragno
 * @copyright https://creativecommons.org/licenses/by-sa/4.0/
 */
#include "TempSensor.h"
const int B = 4275; // B value of the thermistor
const int R0 = 100000; // R0 = 100k
TempSensor::TempSensor(byte pin){
    // initialize _pin
    _pin=pin;
    _filter=TEMP_FILTER_1;
}
void TempSensor::setFilter(byte f){
    //accetta solo i valori previsti, li altri sono ignorati
    switch (f){
        case TEMP_FILTER_1:
        case TEMP_FILTER_2:
        case TEMP_FILTER_5:
        case TEMP_FILTER_10:
            _filter=f;
            break;
    }
}
byte TempSensor::getFilter(){
    return _filter;
}
/**
 * vedi https://wiki.seedstudio.com/Grove-Temperature_Sensor_V1.2/
 */
float TempSensor::getRawTemperature(){
    int a = analogRead(_pin);
    float R = 1023.0/a-1.0;
    R = R0*R;
    return 1.0/(log(R/R0)/B+1/298.15)-273.15; // convert to temperature via datasheet
}
float TempSensor::getTemperature(){
    return round((10.0/_filter)*getRawTemperature())*_filter/10.0;
}
```


Network

```
#ifndef NETWORK_H
#define NETWORK_H
/**
 * @file Network.h
 * @date 06 02 2023
 * @author Giovanni Ragno
 * @copyright https://creativecommons.org/licenses/by-sa/4.0/
 */
#include <Arduino.h>
#include <ESP8266WiFiMulti.h>
/**
 * @class Network
 * @brief Gestisce connessione con rete wifi
 * Azioni disponibili: chiamata http get
 */
class Network{
protected:
    ESP8266WiFiMulti WiFiMulti; /**< @var oggetto di connessione alla wifi */
public:
    /**
     * constructor
     * @param ssid identificativo di wifi
     * @param password per l'accesso alla wifi
     */
    Network(String ssid, String password);
    /**
     * Chiamata http get
     * @param url indirizzo da chiamare
     * @return String il messaggio restituito dal server, vuoto se in errore
     */
    String httpGET(String url);//
};
#endif
```

```
/**
 * @file Network.cpp
 * @date 04 02 2023
 * @author Giovanni Ragno
 * @copyright https://creativecommons.org/licenses/by-sa/4.0/
 */
#include "Network.h"
#include <ESP8266HTTPClient.h>
#include <WiFiClient.h>
#define NETWORK_DEBUG false
Network::Network(String ssid, String password){
    WiFi.mode(WIFI_STA);
    WiFiMulti.addAP(ssid.c_str(), password.c_str());
}

String Network::httpGET(String url) {
    String payload="";
    // wait for WIFI connection
    if (WiFiMulti.run() == WL_CONNECTED) {
        WiFiClient client;
        HTTPClient http;
        if (NETWORK_DEBUG) {
            Serial.print("[Network] HTTP: begin...\n");
        }
        if (http.begin(client, url)) { // HTTP
            if (NETWORK_DEBUG) {
                Serial.print("[Network] HTTP: GET...\n");
            }
        }
        // start connection and send HTTP header
        int httpCode = http.GET();
        // httpCode will be negative on error
        if (httpCode > 0) {
            // HTTP header has been send and Server response header has been handled
            if (NETWORK_DEBUG) {
                Serial.printf("[Network] HTTP: GET... code: %d\n", httpCode);
            }
            // file found at server
            if (httpCode == HTTP_CODE_OK || httpCode == HTTP_CODE_MOVED_PERMANENTLY) {
                payload = http.getString();
                if (NETWORK_DEBUG) {
                    Serial.println("[Network] HTTP: GET received msg: "+payload);
                }
            }
        }
        else {
            if (NETWORK_DEBUG) {
                Serial.printf("[Network] HTTP: GET... failed, error: %s\n", http.errorToString(httpCode).c_str());
            }
        }
        http.end();
    }
    else {
        if (NETWORK_DEBUG) {
            Serial.printf("[Network] HTTP: Unable to connect\n");
        }
    }
}

return payload;
}
```

Broker

```
#ifndef BROKER_H
#define BROKER_H
/**
 * @file Broker.h
 * @date 06 02 2023
 * @author Giovanni Ragno
 * @copyright https://creativecommons.org/licenses/by-sa/4.0/
 */
#include <Arduino.h>
#include "Network.h"
/**
 * @class Broker
 * @brief Gestisce colloquio con broker utilizzando risorse di rete fornite da un oggetto network
 * azioni gestite: chiamata delle api get e set e relativi valori di ritorno
 */
class Broker{
protected:
    Network* _net; /**< @var oggetto di connessione alla rete */
    /**
     * emissione di messaggi su seriale per debug della deserializzazione da JSON
     * @return void
     */
    void deserializaDbg();
public:
    /**
     * constructor
     * @param net oggetto di connessione alla rete
     */
    Broker(Network* net);
    /**
     * chiamata alla api get
     * @param key chiave da ricercare
     * @return String il valore associato alla chiave, NULL se chiave non trovata
     */
    String get(String key);
    /**
     * chiamata alla api set
     * @param key chiave da impostare
     * @param value valore associato alla chiave
     * @return String conferma il valore in caaso di successo, NULL se in errore
     */
    String set(String key, String value);
};
#endif
```

```
/**
 * @file Broker.cpp
 * @date 04 02 2023
 * @author Giovanni Ragno
 * @copyright https://creativecommons.org/licenses/by-sa/4.0/
 */
#include "Broker.h"
#include "Network.h"
#include <ArduinoJson.h>
#define BROKER_DEBUG true
String BASE_URL = "http://www.schoolmakerday.it/broker/";
//String BASE_URL = "http://192.168.1.8/broker/";
String GET_URL = BASE_URL+"get.php?key=";
String SET_URL_1 = BASE_URL+"set.php?key=";
String SET_URL_2 = "&value=";
/* per dimensioni vedi https://arduinojson.org/v6/assistant/ */
const int docCapacity = 192; // > di JSON_OBJECT_SIZE(1) + JSON_OBJECT_SIZE(3);
StaticJsonDocument<docCapacity> answerDoc;
DeserializationError error;
Broker::Broker(Network* net){
    _net=net;
    if (BROKER_DEBUG) {
        Serial.printf("[broker] docCapacity: %d\n", docCapacity);
    }
}
String Broker::get(String key){
    const char * val;
    String msg=_net->httpGET(GET_URL + key);
    if (msg){
        error=deserializeJson(answerDoc,msg);
        if (BROKER_DEBUG) {
            deserializaDbg();
        }
        val=answerDoc["data"]["value"];
    }
    else {
        val=NULL;
    }
    return val;
}
String Broker::set(String key, String value){
    const char * val;
    String msg=_net->httpGET(SET_URL_1 + key + SET_URL_2 + value);
    if (msg){
        error=deserializeJson(answerDoc,msg);
        if (BROKER_DEBUG) {
            deserializaDbg();
        }
        val=answerDoc["data"]["value"];
    }
    else {
        val=NULL;
    }
    return val;
}
void Broker::deserializaDbg(){
    if(error){
        Serial.print("[broker] deserializeJson() failed: ");
        Serial.println(error.c_str());
    }
    else {
        Serial.println("[broker] deserializeJson() success: ");
        String sta=answerDoc["status"];
        Serial.println("\tstatus: "+sta);
        String key=answerDoc["data"]["key"];
        Serial.println("\tkey: "+key);
        String val=answerDoc["data"]["value"];
        Serial.println("\tval: "+val);
        String ts=answerDoc["data"]["ts"];
        Serial.println("\tts: "+ts);
    }
}
```