LoM Relazione

Samuele Evangelisti

a.a. 2021/2022

Contents

1 Introduzione

Il sistema ha lo scopo di implementare un controllo degli accessi fisico. Nello specifico l'MCU ha lo scopo di azionare una serratura o un cancello. Ai fini del progetto il comportamento di azionamento della serratura o del cancello è stato simulato con un *piezo buzzer*. L'azione veicolata dall'MCU può essere innescata in due modi:

- **PIN o RFID**: è posibile fornire all'MCU un PIN o un RFID, in entrambi i casi il codice inserito verrà verificato, in caso il codice sia abilitato verrà innescata l'azione
- **API**: è possibile innescare l'MCU direttamente dall'interfaccia del gateway, in questo caso è necessario autenticarsi prima di poter interagire con l'MCU

2 Struttura

Il sistema si compone di tre componenti:

- **Database**: contiene i dati relativi agli utenti, agli MCU collegati (nel nostro caso sarà uno) e registra le operazioni di accesso effettuate;
- **Server**: opera come gateway ricevendo i codici forniti all'MCU e convalidando o no l'accesso mediante i dati presenti nel database. Fornisce anche una web application per poter operare sugli utenti e sui dispositivi registrati;
- **MCU**: consente l'inserimento di un codice tramite PIN o RFID, lo comunica al gateway e, in caso di autenticazione confermata, esegue l'azione. Permette anche di registrare o rimuovere un RFID per un dato utente tramite il PIN;

La struttura viene descritta in figura 1 a pagina 3.

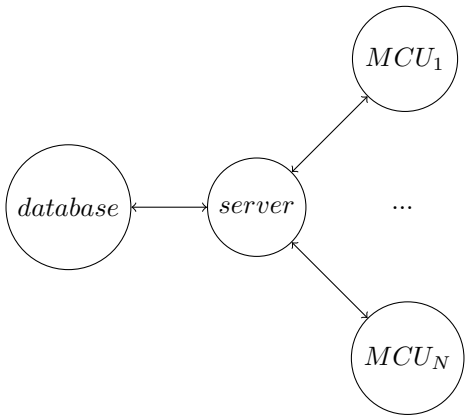

Figure 1: Struttura del sistema

3 Implementazione

3.1 Database

Come database è stato scelto mariadb. All'interno vengono generate le seguenti tabelle:

- **Devices**: informazioni relative agli MCU registrati (tabella 1 a pagina 3);
- **Users**: informazioni relative agli utenti registrati e ai rispettivi permessi di accesso (tabella 2 a pagina 4);
- **Sessions**: informazioni relative al meccanismo di autenticazione per l'utilizzo della web application (tabella 3 a pagina 4);
- **Logs**: registrazione delle operazioni di accesso eseguite dagli utenti (tabella 4 a pagina 4);

Column	**Key**	**Foreign**	
id	primary	-	id del dispositivo
name	unique	-	nome user-friendly
url	unique	-	url del dispositivo

Table 1: Tabella Devices

Column	Key	Foreign	
id	primary	-	id dell'utente
username	unique	-	username dell'utente
password	-	-	hash della password
pin	unique	-	pin dell'utente
rfid	unique	-	rfid dell'utente
level	-	-	livello dei permessi
active	-	-	stato di attivazione

Table 2: Tabella Users

Column	Key	Foreign	
id	primary	Users.id	id dell'utente
token	-	-	token assegnato
expire_datetime	-	-	scadenza del token

Table 3: Tabella Sessions

Column	Key	Foreign	
id	primary	-	id del log
creation_datetime	-	-	data e ora di creazione
operation	-	-	operazione effettuata
status	-	-	stato dell'operazione
log	-	-	informazioni aggiuntive

Table 4: Tabella Logs

3.2 Server

Il server è stato implementato in python usando flask. È stata definita un'API REST con i seguenti path:

GET **/devices**: restituisce la lista di MCU registrati;

POST **/devices/<driver>/<method>**: richiama uno specifico metodo di uno specifico driver passandogli in input il body della chiamata;

POST **/devices**: registra un nuovo MCU;

DELETE **/devices/id** : elimina l'MCU registrato identificandolo con il suo id;

GET **/users**: restituisce la lista di utenti registrati;

POST **/users**: crea un nuovo utente

PATCH **/users/<id>**: modifica i dati di un utente

DELETE **/users/<id>**: elimina un utente

Il server espone anche altri entry point per le operazioni di registrazione (/register), di login (/login) e di interazione con la piattaforma (/dashboard). All'interno del server è anche presente un proxy per permettere di comunicare direttamente con gli MCU evitando problemi di CORS. Questi path non sono da utilizzare direttamente ma vengono utilizzati durante il funzionamento della web application.

Dopo il login, all'utente vengono forniti il proprio id e un token valido. Durante le successive interazioni con il server, gli header di ogni chiamata vanno arricchiti con:

- **LOM_id**: id utente
- **LOM_token**: token valido

La piattaforma, insieme ai dari relativi alla chiamata, restituirà un nuovo token valido. In questo modo il token assume la valenza di un OTP. Di default i token hanno validità per un'ora.

Per quanto riguarda le interazioni con i singoli MCU, queste non vengono autenticate. Attualmente la sicurezza viene garantita dal gateway WiFi al quale si è collegati. Tuttavia è possibile implementare un livello di sicurezza all'interno degli MCU e del server in modo che sia possibile interagire con gli MCU solo passando attraverso il proxy, il quale attualmente controlla il token dell'utente che effettua la chiamata. In questo modo non sarà più possibile interagire direttamente son gli MCU ma solo passano attraverso il proxy o implementando lo stesso layer di sicurezza all'interno del software custom.

Per poter collegare un dispositivo al sistema è necessario identificare l'URL di quel dispositivo. Il dispositivo deve poi implementare l'API standard (definita nella sezione 3.3 a pagina 6). Inoltre deve essere fornito al server un driver python per poter gestire i metodi dell'MCU. Supponendo di voler creare il driver con chiamato *mydriver*, sarà necessario creare il file *Server/driver/mydriver.py*. La struttura base del file sarà:

```
def my_function(data_dict):
    ...

method_map = {
    'method_name': my_function
}
```

All'interno del file *Server/driver/__init__.py* sarà necessario definire:

```
from drivers import mydriver
...

driver_map = {
    'driver_name': mydriver
}
```

Ora sarà possibile, da parte dell'MCU, richiamare il metodo tramite il path del gateway */driver_name/method_name*. Inoltre per tutti gli altri dispositivi che condividono lo stesso driver non sarà necessario reimplementare niente ma semplicemente indirizzare le rispetitve chiamate al driver corretto.

3.3 MCU

Per l'implementazione del dispositivo sono stati utilizzati i seguenti componenti:

- **ESP32**: l'MCU utilizzato è ESP32
- **LCD**: il display LCD ha dimensioni 16x2 ed è collegato tramite I2C
- **keypad**: un keypad 4x4 con 10 cifre, 4 lettere e i caratteri '' e '*'
- **RFID-RC522**: lettore rfid collegato tramite SPI e relative schede magnetiche
- **piezo buzzer**: garantisce feedback alle interazioni con i dispositivi di input e simula l'esecuzione dell'azione desiderata

Il cablaggio dei componenti viene mostrato in figura 2 a pagina 7

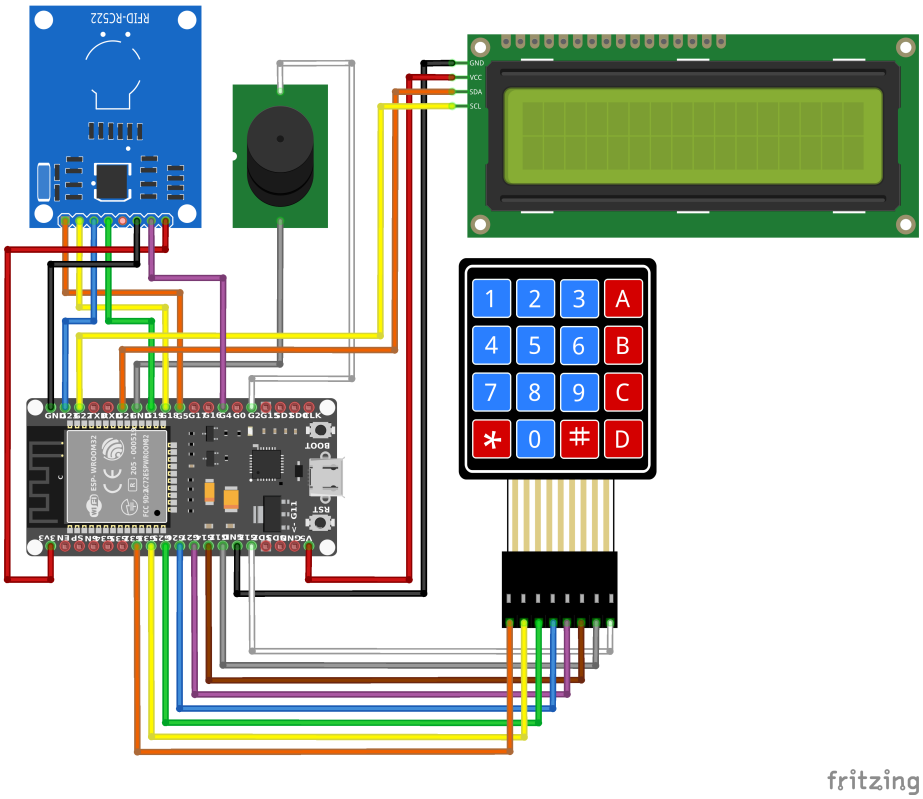

Figure 2: Schematics dell'MCU implementato

Durante l'avvio, il dispositivo si cerca di collegarsi alla rete impostata. Fino a che la connessione non avviene il dispositivo continuerà a cercare di connettersi. Per motivi di sicurezza SSID e password vengono definiti all'interno di Globals.h e il collegamento al gateway WiFi non può essere modificato dall'esterno.

I dispositivi che vogliono collegarsi al sistema devono implementare l'API standard:

GET /: informazioni sullo stato attuale del dispositivo, la risposta deve essere in JSON e contenere i seguenti attributi:

- **success**: true or false;
- **data**: JSON;
  - **attribute_name**: valore
- **error**: errore (se success: false);

PATCH /: settaggio del dispositivo, prende in input un oggetto contenente gli attributi del dispositivo e il valore da assegnare, se la chiave non è presente il valore non è da modificare, la risposta deve avere i seguenti attributi:

- **success**: true or false;
- **error**: errore (se success: false);

GET **/info**: informazioni sugli attributi disponibili all'interno del dispositivo, la risposta deve avere i seguenti attributi:

- **success**: true or false;
- **data**: JSON;
  - **attribute_name**: configurazione;
- **error**: errore (se success: false);

Le configurazioni possono essere oggetti, se si vuole definire altre configurazioni annidate, oppure uno tra i seguenti valori:

- **"boolean"**: per valori booleani
- **"number"**: per numeri interi
- **"string"**: per stringhe di testo
- [**"value1", ..., "valueN"**]: se si vuole definire una lista di valori, questi devono essere sempre stringhe

Il dispositivo implementato ai fini del progetto permette di inserire un PIN tramite il keypad oppure fornire un RFID tramite l'apposito sensore. In entrambi i casi il dispositivo produrrà un breve suono e comunicherà con il server per verificare i permessi di accesso. Questo è possibile perché PIN e RFID sono univoci per ogni utente. Se l'autenticazione va a buon fine, il dispositivo produrrà di nuovo un breve suono e mostrerà a schermo "Access granted", viceversa produrrà tre beep e mostrerà la scritta "Access denied". Tramite il tasto "" è possibile verificare l'indirizzo IP del dispositivo. Tramite il tasto "A" è possibile registrare un RFID per un dato utente mediante il rispettivo PIN. Tramite il tasto "D" è possibile rimuovere un RFID per un dato utente tramite il rispettivo PIN. Gli altri tasti funzione non sono implementati. Le operazini descritte sono effettuabili anche tramite web application