SMASHBOX

# MACRO COMPONENTI software

1. **LOCAL**: parte di acquisizione dei dati di monitoring e spedizione verso il BRIDGE.
2. **BRIDGE**: parte di recupero e inoltro verso il CLOUD.
3. **CLOUD**: gestione in cloud dei Digital Twin.
4. **CLIENT**: agenti esterni che possono accedere al CLOUD tramite interfaccia

# DETTAGLIO COMPONENTI

**LOCAL**

Questa sezione è divisa in due parti:

* Acquisition: la board scelta per leggere i dati dai sensori. Gestisce strettamente l’acquisizione dei dati dai sensori e l’inoltro tramite un canale (seriale o Wi-Fi) verso il Bridge.
* Central: Un Arduino dedicato gestisce il riconoscimento dell’impronta digitale. Centralizzata perché ci sarà un unico riconoscitore per tutte le cassette.

**ACQUISITION**

I dati da acquisire nella parte “Acquisition” sono inviati tramite un pacchetto di un byte per ogni campo che segue (tutti riconosciuti come unsigned byte).

* ID della cassetta
* Presenza dell’oggetto nella cassetta
* Temperatura
* Umidità
* Stato di manomissione (0 se tutto regolare, 1 se è stata effettuata una manomissione)
* Stato di blocco (la cassetta è bloccata perché chiusa)
* Stato di apertura (la cassetta è attualmente aperta o chiusa)

In tutto il pacchetto che ACQ invierà a BRIDGE è quindi di 7 byte ad ogni ciclo.

**CENTRAL:**

L’altra parte del local, quella centralizzata che effettua sia le registrazioni che le richieste di sblocco da parte dell’utente.

Questa è composta da:

* Sensore di impronte (fingerprint)
* Display di servizio
* Un interruttore per passare dalla modalità di enroll a quella di check (v. dopo)
* Un pulsante per resettare la safe-mode (v. dopo)

**BRIDGE**

Il Bridge è uno script Python che ha due endpoint che lavorano in sinergia: uno lato LOCAL e uno lato CLOUD. L’idea di base è che il BRIDGE passi continuamente i dati che arrivano da LOCAL verso CLOUD (e viceversa). Il pacchetto di monitoring arriverà in un'unica trasmissione gestita da un protocollo (v. dopo). Il BRIDGE spacchetta tramite un Parser i dati ed effettua i comandi di update (HTTP/MQTT) verso il CLOUD in maniera separata.

Il BRIDGE è anche il punto di comunicazione tra CENTRAL e le varie ACQUISITIONS, dal momento che ci sono casi in cui il CENTRAL deve comunicare un informazione alle possibili N boards che sono le varie cassette.

**CLOUD (THINGSBOARD)**

La piattaforma scelta per il Cloud è Thingsboard. Qui avverrà il monitoring dei dati nel corrispondente Digital Twin e la gestione dei trigger scaturiti sulla base dei dati che arrivano dal Bridge.

Sostanzialmente raccoglie di ogni cassetta e comunica a tutte le altre di intervenire in caso di Safe Mode.

**TELEGRAM BOT**

E’ un endpoint in cui l’utente viene notificato di aperture della propria cassetta, di eventuali allarmi. Inoltre viene data la possibilità al cliente tramite comando al bot di capire quali sono stati i vari log della propria cassetta registrata.

***/\* MQTT SERVER(MQTT)***

*C’è un ulteriore blocco, che sarà un broker MQTT sul server dove appoggiarsi per scambiare dati tra BRIDGE e THINGS. L’idea è che i trigger di allarme siano mandati su questo server (test.mosquitto.org) e recuperati dal BRIDGE sullo stesso. Questo perché Thingsboard nella sua versione free possiede soltanto la versione per scambiare i dati verso il server e non dal server.*

***SERVER CUSTOMERS (FIREBASE)***

*L’idea di accedere ad un server per visualizzare il log e di poter sottoscrivere per una nuova cassetta potrebbe essere fatta tramite una app che fa delle richieste ad un Server dedicato in cui sono contenuti i log delle cassette. Lo stesso server contiene anche la lista dei clienti registrati che tentano di accedere allo sblocco. Questo non solo può essere usato per il log, ma anche come backup dei clienti registrati sul central per accedere alle cassette. Tutto questo sarà gestito con un Real Time Database di Firebase.*

*Lo schema generale potrebbe essere quello linkato al repository: \Doc\Blocks \*/*

# FUNZIONAMENTO

Il sistema nel prototipo è composto da due cassette di sicurezza (ACQUISITION), e una cassetta centralizzata (CENTRAL), un bridge (script Python) e un bot Telegram.

3.1 INIT

Allo startup, ACQ si mette in attesa del BRIDGE, mentre CNRTL manda sulla seriale un pacchetto identificativo (7 byte) di tutti 0 e si mette in attesa del BRIDGE stesso.

uint8\_t idle\_packet\_init[PACKET\_SIZE] = {0,0,0,0,0,0,0};

Il BRIDGE, una volta lanciato, scannerizza tutte le seriali e legge da ognuna. Quando legge il pacchetto identificativo di CRNTL divide in una lista le seriali collegate a ACQ e l’altra legata a CNTRL.

A questo punto, la parte CNTRL che era in attesa viene risvegliata dal bridge tramite l’invio sulla seriale (dedicata a central) un carattere identificativo per procedere. Le ACQ restano ancora in attesa.

Quello che stanno attendendo è l’ID. Poiché si sta simulando il primo power on del sistema, le cassette fin quando non sono registrate, non possono partire. E per partire hanno bisogno di un ID.

Il CNTRL, risvegliato, entra in fase di enroll, dove aspetta due registrazioni (come le cassette del sistema). Si simula a questo punto la registrazione di due dita (che simulano diversi clienti). Ogni volta che viene registrata una cassetta, da CNTRL viene inviato un pacchetto con l’ID verso il bridge (2 byte), uno identifica l’header (PACKET\_ENROLL) e l’altro byte l’ID stesso.

Il bridge riceve questo pacchetto con l’ID, lo estrae e la manda ad una delle box non già registrate (questo viene tracciata in un dizionario python). Il pacchetto che viene inviato in questo caso sono 10 byte con il meno significativo contenente l’ID. L’ACQ nella sua attesa legge 10 byte sulla serial e quando li riceve (grazie all’enroll) assegna l’ID alla cassetta ed entra in loop.

Nel mentre viene ripetuta la seconda registrazione dell’impronta in modo da replicare lo stesso processo per la cassetta non ancora registrata.

Immagine che contiene testo, schermata, Carattere, numero

Il contenuto generato dall'IA potrebbe non essere corretto.

Un cliente tenta l’accesso tramite il riconoscimento di impronta. CENTRAL si mette in uno stato BUSY per cui non può accettare nessun altro comando dal momento che è in corso un tentativo di accesso. Quindi verrà mandata una richiesta al BRIDGE che tramite il server verifica la richiesta attesa. A questo punto si possono verificare due ipotesi:

* La richiesta è accolta perché il cliente esiste ed è registrato
* La richiesta non è accolta perché non esiste il cliente

3.1 VALID AUTH

Nel primo caso il BRIDGE legge la risposta che sarà composta da una coppia (ID\_CLIENTE + NUMERO CASSETTA) e manda l’update al CENTRAL. Se esiste il cliente, CENTRAL attende il feedback della camera per verificare che effettivamente c’è corrispondenza tra cliente e numero. Lato ACQUISITION, BRIDGE manderà lo sblocco solo alla cassetta associata con l’ID(ACQUISITION\_X). Lo sblocco può essere soltanto fatto dal BRIDGE, l’utente non può (altrimenti rientriamo in manomissione). Lato THINGS deve aggiornare lo stato del Digital Twin in maniera che sia notificato il fatto che sia aperta.

Quando termina l’operazione e quindi la chiusura manuale, ACQUISITON\_X manda un feedback di avvenuta chiusura (tramite sensore di luce?), BRIDGE aggiorna CENTRAL (che torna in stato IDLE) e THINGS.

3.2 INVALID AUTH

Nel secondo caso, se viene negato l’autorizzazione BRIDGE manda a CENTRAL un errore e viene notificato e torna in IDLE. ACQUISITION non viene coinvolto.

3.3 SAFE MODE

Lo stato di monitoring deve essere sempre attivo. ACQUISITION deve continuamente mandare i dati ogni x secondi (TBD quanti). BRIDGE spacchetta i dati e aggiorna i campi su THINGS. Se viene superata una qualsiasi soglia si entra in *SafeMode*.

Quando viene innescata, viene inviato un messaggio MQTT su un server esterno (MQTT) che viene prelevato dal bridge. A questo punto, le cassette devono entrare in lock e non possono essere aperte. L’innesco provoca un inoltro del comando ricevuto da BRIDGE verso CENTRAL (in stato PROTECTION o BUSY in modo che non avvenga nessuna identificazione su FINGERPRINT) e verso tutte le ACQUISITION, che devono essere lockate. Si esce da questa modalità solo un uno sblocco su CENTRAL (pulsante manuale ad esempio). Lo sblocco manuale da CENTRAL (stato IDLE a quel punto) manda una notifica a BRIDGE che sblocca la possibilità di essere aperte tutte le ACQUISITION. Automaticamente, aggiornando lo stato delle acquisition, thingsboard si aggiornerà in automatico.

# PROTOCOLLO ACQUISITION -> BRIDGE

L’idea è di mandare un pacchetto unico di dati che comunichi tutto il monitoring. I vari campi potrebbero avere tipi di dato diversi (anche per più blocchi)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **START BYTE 0x1D** | **Temperature** | **Accel** | **Presence** | **Lock (1byte)** | **END BYTE**  **0xFF** |

/\* TODO: Aggiungere qui la descrizione di cosa si riceve dai vari sensori, come viene acquisito e tutto quello che serve da descrivere \*/

# CLIENT