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 (ACQUISITION+CENTRAL)*

Questa sezione è divisa in due parti:

* Acquisition: dove Arduino legge 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: dove un altro micro (potrebbe essere anche un altro Arduino) gestisce il riconoscimento dell’impronta digitale. Centralizzata perché ci sarà un unico riconoscitore per tutte le cassette. Qui ipoteticamente anche la gestione della Smart Camera.

I dati da acquisire nella parte “Acquisition” sono i seguenti:

* Temperatura: rilevazione della temperatura tramite il sensore (TODO: inserire il codice eprotocollo di comunicazione etc).
* Accelerometro: rilevazione di movimenti della cassetta tramite (TODO: inserire il codice, protocollo di comunicazione etc)

…

…

I tempi di refresh per ogni invio di pacchetti, possono essere anche lunghi (5 secondi ad esempio).

***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. In teoria a meno del Parser, il bridge farà solo da ponte per i dati, non gestirà nessuna logica aggiuntiva.

***CLOUD (THINGS)***

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

***SERVER CUSTOMERS (FLASK)***

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. Probabilmente questo sarà gestito con Flask.

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

# FUNZIONAMENTO

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 FLASK 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

Nel primo caso il BRIDGE legge la risposta e manda l’update al CENTRAL, che passa in stato VALID nel momento in cui c’è una cassetta aperta. 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.

// TODO: Descrivere la parte di validazione Fingerprint e quindi segnalazione del numero nel caso di corretta identificazione di SERVER CUST (coppia Fingerpint, numero di cassetta)

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.

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.

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, tutte le cassette devono entrare in lock e non possono essere aperte. L’innesco provoca un comando da parte di THINGS verso il BRIDGE, che inoltra questo comando a CENTRAL (in stato PROTECTION o BUSY) e tutte le ACQUISITION 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. Contestualmente si aggiorna anche THINGSBOARD.

# 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