BRIDGE

INTRODUZIONE

Questo software è stato realizzato utilizzando Visual Studio Code e il linguaggio di programmazione Python. È parte integrante del progetto “Neighborhood Delivery” creato per l’esame di IoT & 3D Intelligent Systems dal gruppo Kaido, Merolla e Calvano.

Il software viene eseguito su computer fisso o laptop domestico qualsiasi, ma a livello definitivo potrebbe lavorare su Raspberry Pi. È lo script che permette la comunicazione tra il lato dei sensori e il lato Cloud del progetto. Comunica tramite socket con gli ESP-32 e con gli script “Find Neighbour” e “Update Presence” tramite richieste http. È connesso al Wi-Fi di casa ed è in ascolto sulla rete locale alla porta 5050.

FUNZIONAMENTO

Lo script Python lavora come un server utilizzando le socket e rimane in attesa di connettersi con un dispositivo ESP-32. Utilizza i thread per gestire più connessioni in contemporanea, chiudendo le connessioni e spegnendo i thread qualora risultassero inattive o con errori. Le funzionalità principali sono due e sono collegate ai due sistemi ESP-32.

La prima consiste nel gestire ingresso ed uscita delle persone dall’abitazione, tenendo un conto del numero di persone presenti all’interno della casa e aggiornando tale conto tramite i messaggi inviati da parte dei sensori del sistema “Sonar”. Il conteggio è successivamente utilizzato per inviare lo stato (digital twin) dell’abitazione sul Cloud, tramite richiesta http POST al file “Update Presence”.

La seconda riguarda il sistema “RingDoor” e serve per fornire al corriere la più vicina abitazione disponibile al reperimento del pacco, qualora non ci fosse nessuno nell’abitazione del destinatario dello stesso. La richiesta viene inviata tramite socket da parte del sistema ESP-32 e ricevuta quindi dal Bridge, che la inoltra tramite http GET al file “Find Neighbour”. La risposta di quest’ultimo, contenente il codice da visualizzare sullo schermo LCD di “RingDoor”, viene opportunamente adattata dal Bridge, che la invia immediatamente al sistema ESP-32.

CODICE E FUNZIONI PRINCIPALI

Lo script, scritto in Python, è scomposto in due classi principali, che corrispondono ai thread dei due tipi, RingDoorThread e SonarThread. Ogni classe è specifica per un determinato sistema ESP-32 e contiene svariate funzioni per portare a termine i compiti richiesti. Durante l’esecuzione del programma possono collegarsi al Bridge più sistemi della stessa tipologia, ognuno verrà identificato ed assegnato ad un thread specifico. Questo viene fatto durante il flusso principale (main) del codice, che rimane in attesa di ricevere una connessione socket e, quando ciò accade, esegue i controlli necessari per stabilire quale thread inizializzare ed eventualmente quale thread chiudere. A causa dell’utilizzo elevato dei thread, alcune funzioni sono state rese thread-safe, ovvero accessibili solo da un thread alla volta, per evitare errori durante l’accesso e la modifica delle variabili.

Qui di seguito vengono elencate le funzioni principali del software.

* ***def start()***

Inizializza il server socket e rimane in ascolto per stabilire le connessioni con i client. Esegue i controlli dopo aver stabilito la connessione per poi iniziare il thread corretto.

* ***def check\_mac(mac, dict)***

Controlla che il nuovo dispositivo connesso al server non abbia già una connessione attiva. In quel caso, viene chiusa la vecchia connessione e relativo thread.

* ***def run()***

Funzione di esecuzione del thread, rimane in attesa di ricevere messaggi dal dispositivo connesso e, a seconda della tipologia, svolge le operazioni relative.

* ***def check\_presenza(move)***

Il parametro move è il messaggio inviato da “Sonar” e relativo all’ingresso o uscita delle persone, viene aggiornato il conteggio e, se necessario, comunicato al Cloud un cambiamento di stato dell’abitazione.

* ***def find\_package\_receiver()***

Dopo aver ricevuto una richiesta da parte del sistema “RingDoor”, la funzione cerca di individuare la risposta adatta, controllando il conteggio interno al Bridge (per sapere se sono presenti persone in casa) oppure richiedendo al Cloud.

* ***def receive\_from\_cloud()***

Invia la richiesta http al Cloud per la ricerca del vicino disponibile alla consegna. La risposta contiene il codice di tale vicino.

DIPENDENZE

Il file utilizza diverse librerie esterne per svolgere le sue funzioni.

1. **Socket**, per configurare il server in ascolto.
2. **http**, per inviare le richieste GET e POST agli script del Cloud.