Progetto IOT: Beer Station

Sommario

Introduzione

Cogliendo l'occasione per lavorare con gli strumenti e le tecnologie utilizzate nel campo dell'IoT, ho voluto provare a ricreare un'idea prendendo spunto dal funzionamento dei distributori di bevande che si trovano in alcune catene fast food o generalmente nei luoghi di ristorazione.

Il progetto consiste nel provare a ricreare tale sistema, applicando ad esso alcune tecnologie viste a lezione del corso. In particolar modo, voglio automatizzare le sequenze di acquisto ed erogazione di una bevanda da parte di un utente, utilizzando principalmente la tecnologia RFID/NFC e usando le schede programmabili ESP 32/8266.

Inoltre, ho deciso di contestualizzare il progetto in accordo con la mia passione per la birra.

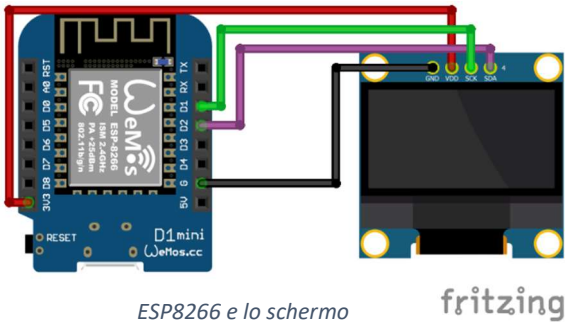
Sintesi del funzionamento

Il progetto consiste principalmente in due macro-componenti:

- La parte riguardante l'interazione dell'utente con la stazione di erogazione di birra, e tutte le componenti che supportano questo compito
- Un'applicazione per smartphone, che permette all'utente di visionare le consumazioni

Premesse

- Vista la complessità della gestione dei liquidi in presenza di componenti elettroniche e ai tempi per ottenere tali componenti da rivenditori, ho deciso di sostituire la pompa con uno schermo OLED da 0.96", in quanto riesce ad esprimere adeguatamente lo stato in cui la pompa si troverebbe nei vari momenti.



ESP8266 e lo schermo che sostituisce la pompa

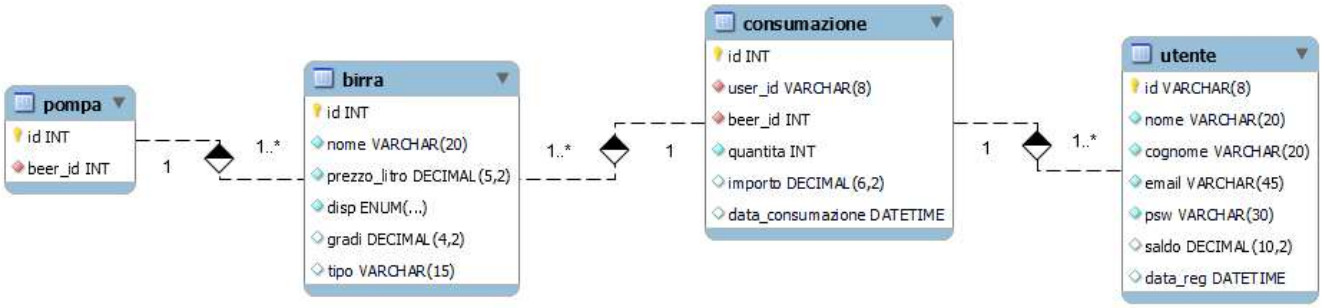
Tecnologie e Componenti

Componenti fisiche

- **DOIT DEVKIT v1 (ESP32):** Si interfaccia con il lettore RFID e invia messaggi tramite il protocollo MQTT, che vengono ricevuti da un server Python che esegue determinate procedure.
- **Wemos D1 mini (ESP8266):** Si interfaccia con una pompa a cui invia il segnale di erogazione ricevuto tramite protocollo MQTT dal server Python.
  Un'altra scheda viene invece usata assieme ad un lettore RFID per inserire nel tag NFC di un bicchiere l'ID di un utente usando la propria tessera personale.
- **Lettore RFID:** Il modulo MFRC522 accoppiato con ESP32 ed ESP8266. Permette l'accesso in lettura e scrittura alle EEPROM dei moduli NFC
- **Moduli NFC:** Vengono impiegati 2 tipi di PICC (Proximity Integrated Circuit Card) per identificare ciascun utente:
  - **NTAG213 (Tag):** Viene posto sul bicchiere per identificarlo e accedere alla stazione
  - **Mifare 1KB (Card):** Contiene le informazioni (ID) dell'utente registrato
- **Raspberry Pi:** Usato come hub di raccolta ed elaborazione delle richieste da parte delle schede ESP32, e come interfaccia al server che ospita il database. Ospita inoltre il broker del protocollo MQTT e il server.
- **PC/Server:** Un computer che ospita il database del progetto e i processi PHP dedicati a gestire le richieste dell'applicazione.

Tecnologie

- Il **database** è stato creato e gestito con **MySQL**, e mi ci sono interfacciato con *MySQL Workbench* ed *Heidi SQL*. È composto da 4 tabelle *Utente*, *Consumazione*, *Birra* e *Pompa*.



Sono presenti alcuni trigger che semplificano l'inserimento di nuovi record nel sistema. Ad esempio, il trigger *updateImporto* calcola il nuovo saldo da pagare di un utente ogni volta che effettua una nuova consumazione, ed aggiorna il suo valore nellla tabella.

- **MQTT**: Protocollo di messaggistica basato sul modello di pubblicazione/sottoscrizione. Ho scelto questo protocollo di comunicazione perché permette lo scambio di informazioni in maniera veloce ed efficiente.
- **Applicazione APK**: Sviluppata con Android Studio su linguaggio Flutter, permette di:
  - Effettuare il login con i propri dati
  - Consultare lo storico delle consumazioni dell'utente
  - Saldare il conto tramite una pagina di pagamento
- **Server PHP**: Gestisce le richieste da parte dell'applicazione (login, registrazione, saldo, …)
- **Server Python**: Iscritto a diversi topic con radice "beer/", verifica che l'ID del tag sul bicchiere corrisponda ad un utente registrato e nel caso di un riscontro positivo, invia il segnale di erogazione alla pompa e successivamente registra la consumazione nel database

Approfondimento

Una visione generale

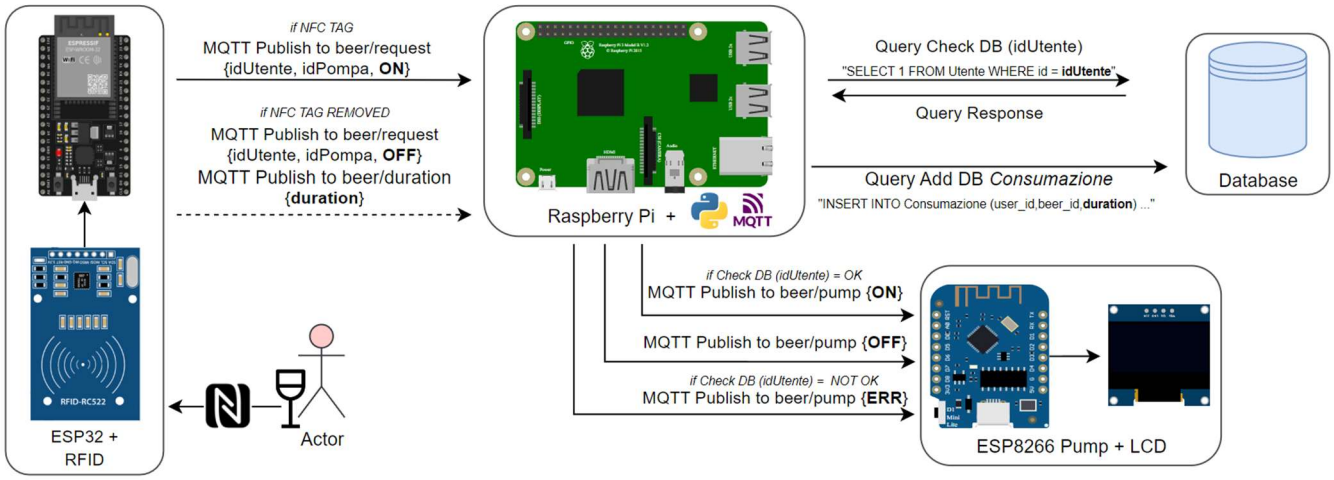


Diagramma di interazione generale dei componenti principali

Prima di scendere nel dettaglio analizzando le varie componenti del sistema, vorrei riassumere brevemente il suo funzionamento nel caso d'uso generale:

- L'utente già registrato al sistema, presenta la propria tessera sulla quale è salvato il proprio user ID. La tessera viene letta da un lettore RFID collegata ad una scheda ESP8266 che si occupa di trasferire tale ID su un tag applicato ad un bicchiere, con il quale l'utente avrà accesso alle varie stazioni di erogazione di birra.
- Quando l'utente appoggia il bicchiere sulla stazione, la scheda ESP32 collegata ad un lettore RFID si occupa di leggere lo user ID ed inviare un payload tramite MQTT al server Python che, dopo aver controllato la presenza (dell'utente) nel DB tramite query, invia il comando (erogazione/errore) alla pompa corrispondente. In caso di errore, verrà visualizzato un messaggio sullo schermo.
  Fino a quando il bicchiere rimane sul lettore nella stazione, la pompa continuerà ad erogare liquido. Questo approccio seppur più complesso a livello di realizzazione, permette all'utente di erogare la quantità preferita al contrario di dover decidere tra un set statico imposto dal sistema.
- Non appena l'utente rimuove il bicchiere, la scheda ESP32 invia sempre tramite MQTT un altro payload al server Python che inoltra la richiesta alla pompa di terminare l'erogazione. Oltre a tale richiesta, la scheda invierà un altro messaggio che contiene il tempo di erogazione della pompa.
- Il server Python, alla ricezione di tale messaggio, si occupa di inserire la consumazione nel DB.

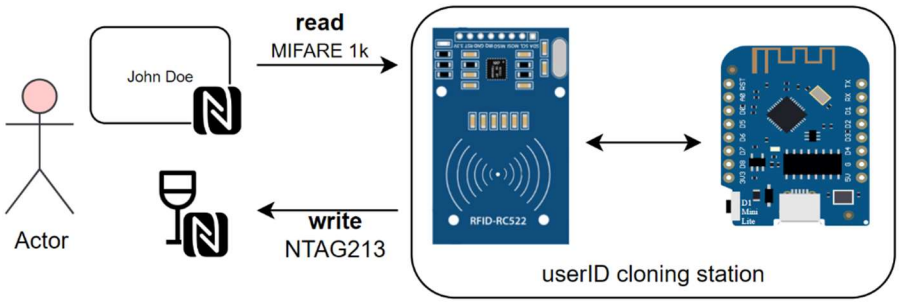
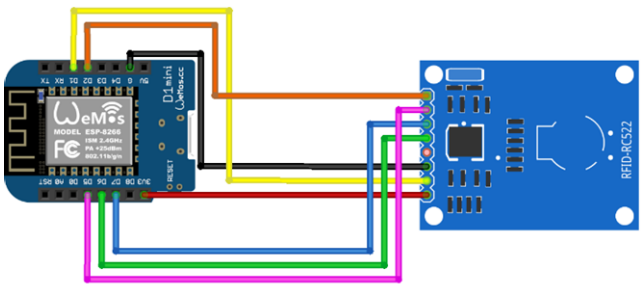


Diagramma di interazione tra l'utente e la stazione di trasferimento dell'ID

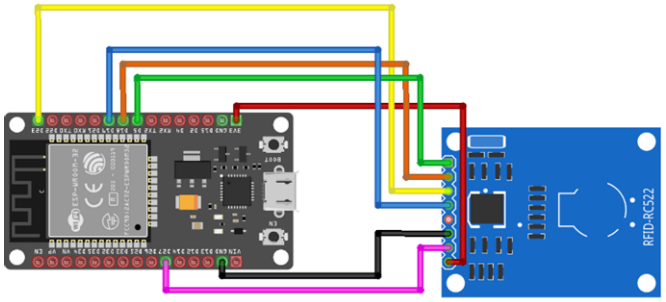
La lettura dei moduli NFC

Ho scelto una scheda con ESP32 da collegare al modulo RFID MFRC522 (almeno per rilevare il tag del bicchiere) in quanto tra i due moduli a disposizione, ho ritenuto che fosse in grado di sopportare meglio il flusso di dati continuo rispetto alla sua controparte con ESP8266.

Per trasferire invece l'ID dell'utente dalla propria Card al Tag sul bicchiere ho proposto un sistema analogo con una scheda ESP8266 al posto di ESP32.



Il collegamento tra ESP32 e MFRC522



Il collegamento tra ESP32 e MFRC522

I tipi di moduli NFC

Anche se i due moduli usati possiedo una struttura dati apparentemente simile, presentano sostanziali differenze nella gestione e dimensione della memoria. La libreria usata MFRC522.h, non supporta completamente tali moduli. Tuttavia, è stato comunque possibile usarla visto che le uniche limitazioni in cui mi sono imbattuto, riguardano potenzialmente il numero di indirizzi logici delle memorie raggiungibili, ma per il mio progetto prevede l'uso di una piccola quantità di memoria e quindi non ho riscontrato problemi.

- **NTAG213** è organizzato in 45 pagine, ognuna contenente 4 Byte, per un totale di 180 Byte.
  I primi 9 byte contengono il numero seriale univoco (UID) e due byte di controllo, mentre i byte nelle pagine 4-15 sono dedicati ai dati utente, ed è dove si trova l'ID dell'utente nel mio progetto. Ho scelto questo tipo di tag per il costo e dimensioni ridotte e dispone di memoria sufficiente.

NTAG213

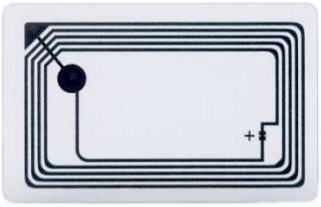
	NUMERO BYTE PAGINA				
PAG	0	1	2	3	INFO
0	UID0	UID1	UID2	CB0	SN + Check
1	UID3	UID4	UID5	UID6	SN
2	CB1		LCK0	LCK1	Check + Lock
3					
4 … 39	*user id : xxxxxxxx*				USER DATA
.. 44					CONFIG BYTES

Una visione semplificata della struttura della memoria nel modulo NTAG213

- **Mifare 1K** è organizzato in 16 settori composti da 4 blocchi da 16 Byte ciascuno, totale 1KB.

  Il suo UID si trova nei primi 4 Byte di memoria. Presenta un meccanismo di protezione complesso che può portare al blocco totale della card; quindi, ho optato per un ruolo più statico.

  L'ultimo blocco (4°) di ogni settore è denominato "trailer block" ed è riservato per 2 chiavi e dei bits di controllo agli accessi.



MIFARE 1K

		NUMERO BYTE BLOCCO								
SEC	BLK	0	..	5	6	..	9	10	…	15
15	3	KEY A			ACCESS BITS			KEY B		
	2	USER DATA								
	1									
	0									
…	…									
1	3	KEY A			ACCESS BITS			KEY B		
	…	*user id : xxxxxxxx*								
	0									
0	3	KEY A			ACCESS BITS			KEY B		
	2	USER DATA								
	1									
	0	UID			MANUFACTURER DATA					

Una visione semplificata della struttura della memoria nel modulo MIFARE 1K

La comunicazione

La comunicazione tramite schede ESP e Server avviene tramite il protocollo MQTT.

- La scheda ESP32 pubblica sul topic "beer/requests" un payload contenente la richiesta di un utente di volere consumare una bevanda.
- Analogamente, quando il bicchiere viene rimosso dal lettore, pubblica sullo stesso topic il comando per fermare la pompa.
- Il server riceve entrambe le richieste iscrivendosi al topic, e pubblica i messaggi sul topic "beer/pump {*idPompa*}" per inviare i comandi adeguati alla pompa identificata da *idPompa.*

Oltre alla suddivisione del dominio del topic, evitando quindi che una pompa “rubi” una erogazione ad un’altra, MQTT offre anche un livello di QoS (Quality of Service) che riduce il numero di messaggi persi o duplicati, tramite accorgimenti del broker.

Ricezione delle richieste, update DB e risposta

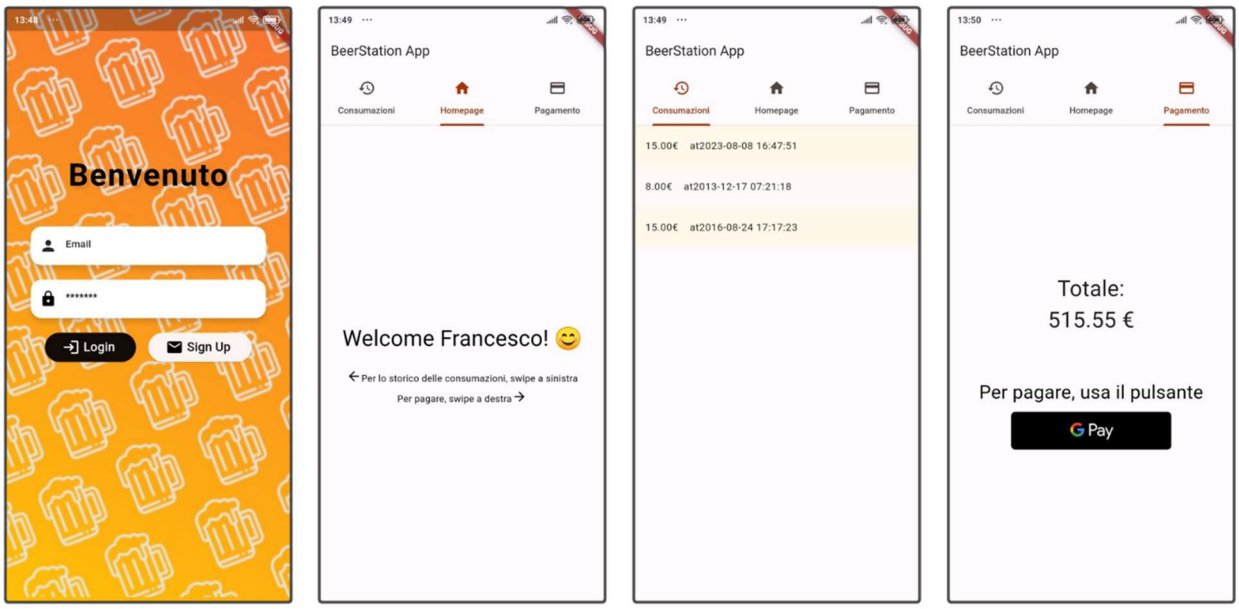
A ricevere la richiesta di erogazione vi è un server Python, che è iscritto tramite protocollo MQTT al topic “beer/richiesta”: al suo arrivo, estrare il payload e lo converte in formato json dai cui preleva l’id dell’utente, la pompa selezionata ed il comando. Invia tramite MQTT il comando sul topic “beer/eroga/PUMP*” alla pompa corrispondente, e una volta ottenuto il feedback di terminare l’erogazione, si collega al DB per addebitare il costo della consumazione all’utente.

L’applicazione Android

Ho sviluppato usando Flutter, un’applicazione per Android per consultare in maniera sintetica le consumazioni che l’utente ha effettuato precedentemente e di poter pagare il conto. Suddivisa in:

- Pagina di login: Se l’utente è registrato, tramite l’inserimento di mail e password permette di accedere ai propri dati
    - Nel caso un utente non sia registrato, può farlo accedendo alla pagina di registrazione con il pulsante Sign Up
- Una volta fatto l’accesso si accede alla Homepage, dove tramite scorrimento orizzontale si arriva alle sezioni Consumazioni e Pagamento .

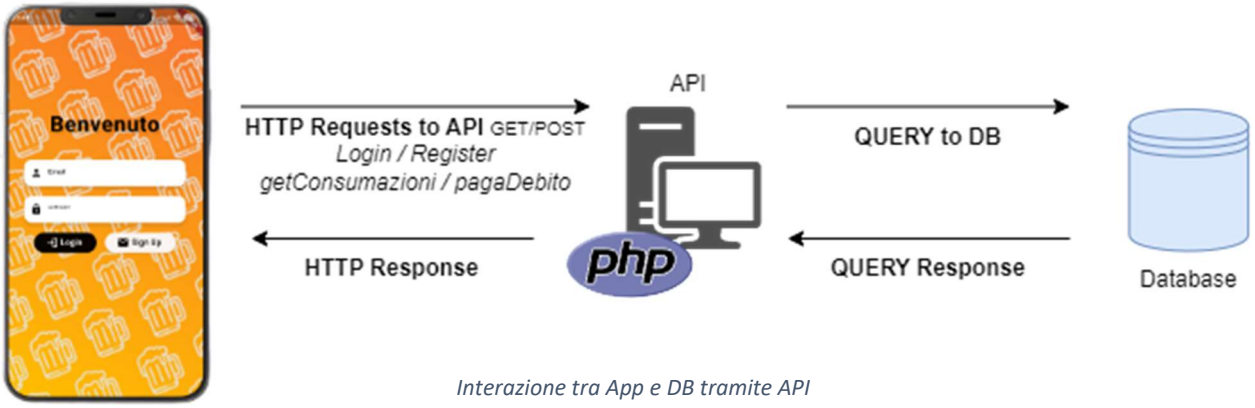
Oltre a poter consultare l’applicazione usando l’apk, sono disponibili degli screenshot nella repository.



Alcune schermate delle pagine principali nell'applicazione

La comunicazione tra applicazione e database avviene tramite una API realizzata con pagine PHP, che permettono di ottenere e manipolare i dati nel database:

- *Login.php* verifica se l'utente ha inserito la mail e la password corretti presenti nel DB
- *Register.php* inserisce i dati di un nuovo Utente
- *Resetdebt.php* azzera debito dell'utente aggiornando la tabella Utente
- *Getconsumazioni.php* recupera le consumazioni di un Utente



Interazione tra App e DB tramite API

Conclusioni

Riflessioni

Questo progetto mi ha permesso di approfondire l'uso delle schede ESP di cui avevo già una conoscenza basilare. Grazie all'utilizzo di MQTT ho capito come gestire meglio la comunicazione tra diverse schede, e come strutturare il progetto in maniera efficiente.

Ho anche avuto modo di poter lavorare su una Raspberry Pi, che ho scoperto essere estremamente versatile e facile da utilizzare. Mi piacerebbe molto espandere la mia conoscenza su di essa e sulle sue potenzialità nell'ambito dell'IoT.

Ho avuto modo di poter studiare la struttura dei tag NFC, che anche se complessa ha innumerevoli utilizzi.

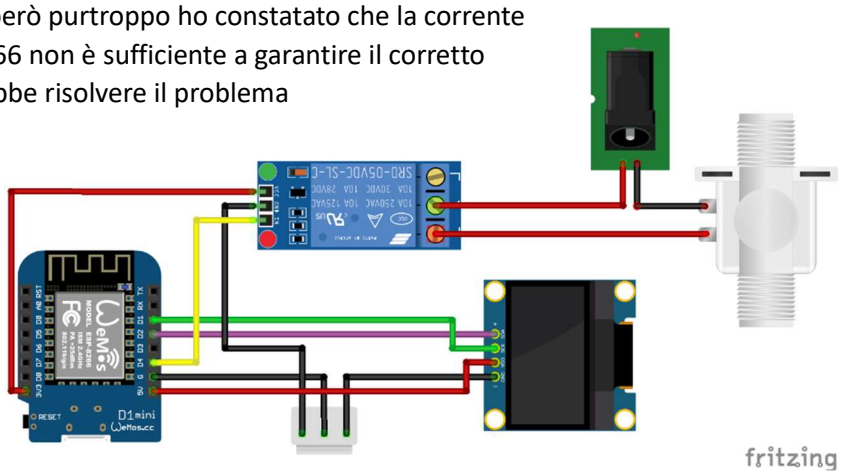
Anche se è stata la mia prima applicazione Android, è stato interessante poterla sviluppare in un linguaggio che non avevo mai usato finora.

TODO e miglioramenti futuri

Come ho accennato nella sezione Premesse, purtroppo ho dovuto accettare dei compromessi durante lo sviluppo del progetto, a causa delle conoscenze necessarie per utilizzare al meglio le tecnologie.

Oltre al display avrei voluto inserire anche il relé che controllerebbe la pompa per dare un secondo feedback sull'erogazione della birra, però purtroppo ho constatato che la corrente fornita dal pin 5V della scheda ESP8266 non è sufficiente a garantire il corretto funzionamento di entrambi. Si potrebbe risolvere il problema usando un alimentatore esterno.

A lato lo schema di come potrebbe risultare un circuito completo:



Il circuito della pompa, comprensivo di schermo, relé e pompa

Alcuni possibili miglioramenti al progetto potrebbero essere:

- L'aggiunta di altre stazioni di erogazione di birre diverse, che lavorano in parallelo (scaling)
- Aggiunta una pagina nell'app con le birre attualmente disponibili
- Riuscire ad avere un prototipo che includa l'utilizzo di una pompa e lavorare con liquidi
- Ottimizzare l'uso della memoria dei tag NFC, criptandone i dati
- Utilizzare meglio la componente QoL di MQTT
- Implementare un pannello informativo, che illustra le birre disponibili e la quantità rimasta.
- Implementare un meccanismo di stop all'erogazione in base al volume massimo del bicchiere salvato nel tag nfc, così da impedire una possibile erogazione infinita per sbaglio

Riferimenti e Bibliografia

Tutto il codice scritto per il progetto è disponibile su Github.

Per la creazione dei diagrammi di collegamento delle schede ai componenti ho utilizzato Fritzing.

Lo schema ER è stato creato tramite MySQL Workbench.

La documentazione per i moduli ESP8266 e ESP32.

La libreria per interagire con il modulo RFID è MFRC522.h