Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Electric Power Engineering

Development of an Applied RFID-based Payment System

Bachelor's Thesis

Author
Gergely Szabó

Advisor
Dr. Zoltán Tóth, Ph.D.

2022

Contents

HALLGATÓI NYILATKOZAT

Alulírott *Szabó Gergely*, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy autentikált felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Budapest, 2021. december 10.

Szabó Gergely
hallgató

Kivonat

Az utóbbi időben a fesztiválokon bevett szokássá vált saját fizetési rendszer bevezetése, ami a készpénzkezeléssel járó kellemetlenségeket hivatott megoldani. A projekt célja az volt, hogy egy ilyen rendszer minden aspektusát úgy valósítsa meg, hogy az *megfizethető legyen* diákcsoportok számára zártkörű rendezvényeiken, úgy hogy a piacon lévő megoldásokhoz hasonló *funkcionalitást* és *biztonságot nyújtson.* A projekt egyik legfontosabb megkötése, hogy a rendszernek aktív internet hozzáférés nélkül is működnie kell. A megvalósításhoz több lehetséges módszer is megvizsgálásra került, és végül a legjobbnak ítélt alapján került elkészítésre. A felhasznált technológiák ingyenesek és nyílt forráskódúak, egyedül a fizikai eszköz elkészítéséhez kellett alkatrészeket vásárolni. A megvalósított projekt hardver prototípusok több iterációját tartalmazza, illetve a legújabb prototpus elektromágneses kompatibilitás (EMC) tesztelését. Hozzá tartozik még egy teljes értékű webalkalmazás adminisztrációs felülettel, a teljes rendszer biztonsági elemzése és egy üzemeltetési segédlet.

Abstract

Recently, it has become a common practice at festivals to introduce their own payment system, which is intended to solve the inconvenience of cash handling. The objective of the project was to implement all aspects of such a system so that it would be *affordable* for groups of students at their private events, providing *functionality* and *security* similar to the solutions on the market. One of the key goals of the project is that the system should work without active internet access. Several possible methods for implementation were examined, and finally developed on the basis of what was considered the best. The technologies used were free and open-source, only the parts for the physical had to be bought. The implemented project includes several iterations of hardware prototypes and an electromagnetic compatibility (EMC) testing of the latest prototype. It also includes a full-featured web application with an administration interface, security analysis of the entire system and operating instructions.

Chapter 1

Introduction

In the university, student groups regularly organize camps and events across the year. The organizers buy the needed resources such as food and drinks. To track personal consumptions, organizers use different ways.

The usage of regular cash could be one solution, but the problem with money is that a decent amount of change is required to maintain the flow. Users must carry their wallets and most importantly, bring cash to the camp, which has become less and less convenient with the widespread of bank cards in recent times.

Another solution can be a paper note where the organizers can write down the transactions. After the event is finished, the participants can pay their debts. This approach is better because it doesn't require change, but paper notes can be lost or damaged easily.

An even better solution would be to carry a token (that identifies the user) and use this token to track the transactions in a semi-automated system. An electronic system can solve the previously addressed problems, but there are technical challenges to solve in order to make the system viable.

The goal was to achieve a working prototype with a reliable backend and a working hardware that solves this problem.

The following requirements were used during the design and implementation process:

- *Affordability:* The maintenance and manufacturing costs must be affordable for student groups.
- *Usability:* Save more time than other existing and widely used solutions.
- *Scalability:* It should be modular and should be easy to upgrade.
- *Portability:* It must work without active internet connection to be compatible with rural application.
- *Security:* The system needs to be durable and secure against accidental damage and malicious users.
- *Applicability:* Must ready to be used in a real life environment.

Chapter 2

Technical approach to the goal

One of the technological challenges of the system is how to quickly and deterministically identify users. The different methods were evaluated according to the following principles:

- *Accuracy:* What is the chance that users will be accidentally or purposely mixed up?
- *Complexity:* How hard is it to use the solution for operators (bartenders) and users?
- *Costs:* The costs of the terminal and the tokens.
- *No internet:* Is the solution works without active internet connection?
- *Proven:* Is it used by anyone on the market for the same or similar purposes.

2.1 Evaluated Methods

2.1.1 Using quick response (QR) code variant I.

QR code Quick response (QR) code is a special type of two dimensional barcodes. "QR Code is a matrix symbology. The symbols consist of an array of nominally square modules arranged in an overall square pattern, including a unique finder pattern located at three corners of the symbol [...] and intended to assist in easy location of its position, size, and inclination. A wide range of sizes of symbol is provided for, together with four levels of error correction." [9]

Method Description Each user has a uniquely generated ID that they receive as a QR code on a printed (and laminated) page or electronically. There must be a QR reader in the terminal.

Advantages

- It is possible to read the full information even if the QR code is damaged.
- There are barcode reader models [1] for as low as $20.

[1] for example: GM61 QR code scanner

- The cost of printed QR codes are pretty low event if they are laminated.
- It is easy to show and read a QR code during the checkout process.
- It is possible to use without an active internet connection.
- It is used on public transport to check tickets purchased electronically in several places.

Disadvantages The code does not change over time so QR codes can easily be stolen by taking a picture of it by a malicious person. This is an acceptable security risk for small groups when they know each other. However, it is not acceptable for festivals or open events.

2.1.2 Using quick response (QR) code variant II.

Method Description It is similar to the method Using quick response (QR) code variant I discussed earlier. The main difference is that the QR codes are not static but generated periodically and only available around the time of generation. It is now not possible to print the QR codes out and requires a device to generate the codes.

Advantages

- Stealing QR codes does not make any sense in this way because they cannot be used twice or later.
- A mobile application or web application can be used for generation. The web application can have its own server and its own separate wireless network nearby the counter.

Disadvantages Users must carry and use their mobile phones for purchasing.

2.1.3 Using quick response (QR) code variant III.

Method Description The QR code is generated by the terminal, and users should scan them to complete the purchase. It requires a high-resolution display and additional infrastructure to connect the mobile devices to the server. A mobile app or web application is also required.

The security of the system depends on the exact algorithm. For example, an algorithm based on RSA can be a good solution to ensure integrity. [2] Such solution could utilize the property of asymmetric key pairs to make it easy to check whether that the keys belong together, thereby reducing network overhead during user authorization.

Neutral The terminal should have a display anyway.

Disadvantages

- It makes the system fragile because it depends on the mobile phones of the users and their QR code reading capability.
- It can work without an active internet connection, but a local network with enough capacity must be installed nearby the terminals.
- The backend server must trigger actions on the terminals (to show the status of a purchase and it makes the terminal design ever so slightly complex).

2.1.4 Using Magnetic stripe cards

Method Description A similar method as in regular payments with a non-contactless debit card.

Advantages

- This method was used long time ago for the same use-case.
- The validation process is fast.
- Magnetic stripe cards are cheap.

Disadvantages

- Magnetic stripe readers can be really expensive. Most models use USB to transport the card details, which makes it harder to integrate with micro-controllers.

2.1.5 Using passive RFID tokens

Method Description An RFID tag is provided for each user with a unique identifier. The RFID tags are passive components, the active part is located in the terminals. This method is widely used for access control applications and contactless payment solutions. There are many types of tags for different needs. It comes in the form of a card, keychain, bracelet, ring and even a sticker. For a festival, for example, an armband is one of the most ideal because it is hard to lose.

"NFC utilizes tap-to-pay technology where consumers can load their payment information into their NFC-enabled smartphones. Using these third party applications can securely store credit/debit card information which is then used for any kind of transactions. It is easy to use, saves time and highly mobile." [11] Even though it is very easy to support a 3rd party applications on this interface from an Android device, this cannot be said about Apple products. Apple does not support host-based card emulation (HCE) and any payment-related usages of their NFC chips for 3rd party applications. Only tag scanning is allowed. For this reason, RFID tags are not completely replaceable with phones. `Chrome for Android` and `Samsung Internet` supports NFC-HCE from the browser via the `Web NFC` [5] API. So on Android, an app support such a system does not have to be native.

Advantages

- Tag price is very low. Depending on the quantity purchased and the type of the tag, the price is between $0.05 and a few dollars.
- There are cheap RFID readers available in the market for as low as $5.
- Tags does not contains active parts, so they will technically never drains.
- Most of the labels are waterproof and durable.
- It is very easy to use. Users only need to take their cards or labels with them and place them near the reader during payment.
- Such system does not rely on any internet connection.
- The method is already in use for contactless payments.
- Tags uniquely identifies users and it is hard to duplicate them.

Disadvantages

- Cheaper RFID readers are not very reliable and are lack of functionalities such as AES encrypted transmission.
- Cheaper types of tags are often vulnerable to malicious label duplication.

2.1.6 Using TOTP physical tokens

Method Description Users receive a TOTP token upon entering the event and a unique user id that they need to memorize or write down. To complete a purchase they need to say their user id and the one-time password. (Which is about 6 digit long.) The one-time password is only depends on the seed, the time and the availability interval. Since the time is synchronized between the two sides (the physical token and the backend server) and the seed is shared at the beginning the two sides always generate the same one-time password. TOTP is further described in the RFC6238 [7] standard.

Advantages

- There's no need for a reader component since a keypad must already exists for other reasons.
- This solution is secure because it is really hard to break the seed from one or more one-time passwords. A generated password is only available once and only for a limited time such as 30 seconds.
- It does not require active internet connection.
- It is simple and reliable to authorize the purchases from a software point of view.

Disadvantages

- Using it requires to dictate roughly ten digits to the bartender until interval closes. This could be frustrating and uncomfortable for many users.
- This method is widely used widely but for a different use-case. TOTP tokens are used for two-factor authentications not for payments.
- Costs of the tokens start from $20 each.

2.1.7 Using TOTP software tokens

Method Description This is similar to the method described in Using TOTP physical tokens. The difference is that users read their QR codes upon entering the event with a TOTP client. This QR code contains a seed and an interval that a generated one-time password is valid. To complete the purchase they use the TOTP software client instead of the physical token.

This solution can be mixed with the physical TOTP token solution because the algorithm is the same.

Advantages No need to buy expensive tokens and all the pros from Using TOTP physical tokens.

Disadvantages

- This solution can still be frustrating and uncomfortable.
- Users must carry and use their mobile phones for purchasing.

2.1.8 Using web server

Method Description Users must login to a web application where they can see their account balance and the available terminals. Once the bartender has entered the items, the transaction is initiated. To complete the purchase the user must click on the menu associated with the terminal where the purchase started and approve the transaction. The

Advantages

- The method is as safe as the web application.
- Active internet connection is not required.

Disadvantages

- It takes a long time to approve, as users will need to sign in to the web application.
- It does not scale well and can be a strange solution at public events or with many terminals.
- Users must carry and use their mobile phones for purchasing.

- Requires a bigger network infrastructure. It makes it harder and more expensive to set up and operate the system.

2.2 Conclusion of evaluation

The objectives set were not always fulfilled to the examined solutions. Had to be considered between cheapness and comfort and had to be taken into account to be safe enough. The selected solution is the Using passive RFID tokens method that is already in use for contactless payments.

Method Name	Accuracy	Complexity	Costs	No internet	Proven	Other
Using quick response (QR) code variant I	++	++	++	++	+	−− (security)
Using quick response (QR) code variant II	++	−−	++	+	++	+
Using quick response (QR) code variant III	++	− − −	−	+	+	−− (comm.)
Using Magnetic stripe cards	+	++	−−	++	++	0
Using passive RFID tokens	++	++	++	++	+ + +	+ (durability)
Using TOTP physical tokens	+	− − −	− − −	++	+	0
Using TOTP software tokens	+	− − −	+ + +	++	+	0
Using web server	++	−−	−	+	−	0

Table 2.1: Evaluation of methods on the basis of the set criteria

Chapter 3

Backend Design

3.1 Architecture

The implementation was done by according to the 3-tier architecture model. This model is not only promoted by the Spring Framework but improves the readability of the project and makes further development of the application easier.

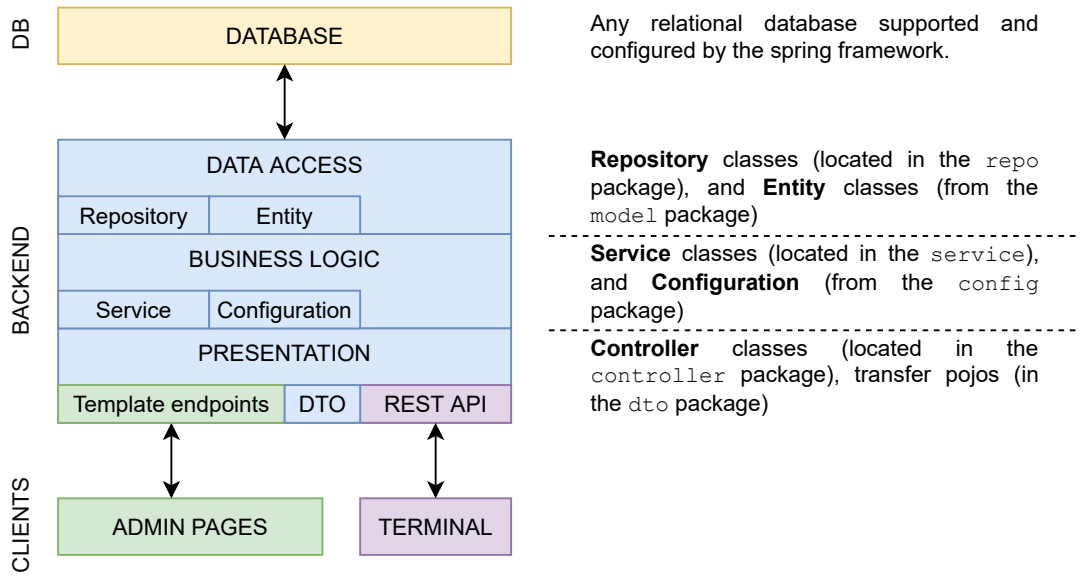

Figure 3.1: 3-tier architecture in this project

In this model the tiers are separated logically. The direction of the dependencies between tiers must go in one direction. It means that components from the Data Access layer never depend on the Business Logic and the Presentation. A component from the Business logic can only depend on the a data access component or another business logic component. Circular dependencies are possible between components of the same tier but contraindicated. Spring Framework can prevent users from creating circular dependencies if used well.

Components are the building blocks of a Spring application. The framework distinguish components from each of the three layers by annotations. A class annotated with @Repository represents a data access component, `@Service` and `@Configuration` represents a business logic component and the `@Controller` represents a presentation layer component.

3.2 Dependency Injection and IoC

The annotated components are managed automatically by the framework. The Spring Framework resolves the dependency tree between the components, instantiate them and injects the dependencies. This way the programmer don't have to deal with the component lifecycles. This pattern is called 'Dependency Injection' and it is a common implementation of the 'Inversion of Control' principle.

3.3 Data Access Layer

This layer is responsible of serving data between the business logic components and the database.

The most commonly used relational databases are supported by hibernate. Hibernate is an ORM (Object Relational Mapping) library that implements the JPA (Java Persistence API) specification. It maps the JPA entities (classes annotated with `@Entitiy`) to database tables. Automatic table generation and migration is enabled by default.

ORMs are generally supports the code-first approach when the developers don't have to worry about the database and its structure. They only have to create a convinient model for the application. The mapping is done by the library.

Entities on the java side follows the object oriented nature of the JVM but relational databases take a different approach. The conversion between the two approach is automatic, but some assistance is required in a form of carnality annotations (`@OneToMany`, `@ManyToMany`, and so on) and some other hints for the mapper.

Hibernate looks for interfaces annotated by @Repository and generates an implementation of the methods for every selected databases. If a repository method is valid it deterministically represents a database query.

An example from the project and what the parts mean:

```
List<ItemEntity> findAllByCodeAndActiveTrueOrderByPriceDesc(String code);
```

Listing 3.1: Example Repository method header

- `List<ItemEntity> findAllBy`: The result must be zero or more ItemEntities
- `code ... (String code)`: The ItemEntity.code equals the first String argument
- `And`: Logic gate between the condfitions
- `ActiveTrue`: The ItemEntity.active bool variable is true
- `OrderByPriceDesc`: Sort the results by the ItemEntity.price numeric variable in descending order

There are also support for NoSQL databases for Spring with similar interface but it is out the scope of this project.

3.4 Business Logic Layer

This layer is responsible of the internal functioning of the application. Connects the data access layer to the presentation.

Configuration

Configuration components represents business logic that needs to be done in the initialization phase. They can contain methods that changes other components or helps to assemble the running environment. Configurations can contains bean factory declarations.

Instances from configuration classes are typically never injected into other components.

Transactions

Service methods can access the database via repositories. Using transactions is recommended in this scenarios. The transaction lifecycle is managed by the spring framework if a service method is annotated with the `@Trasactional` annotation. The framework generates a proxy of the service object that encapsulates the transaction handling logic according to user defined settings. Such settings can be the isolation level, the type of propagation, the rollback strategy or whether the method is a read-only operation.

3.5 Presentation Layer

This layer is responsible of delivering information for the clients in the form of their requirements.

In a http web application (spring supports not just http endpoints but several other ways such as websocket and MQTT) the points of interactions are called endpoints. Endpoints are located inside Controller classes.

An endpoint is a method annotated with the `@RequestMapping` annotation or any of its child (`@GetMapping`, `@PostMapping` just to mention a few).

If the endpoint is annotated with @ResponseBody or the controller annotation is the `@RestController` than the result will be serialized with the default ObjectMapper. This mapper is the jackson databind object mapper by default for pojos. The output format is `application/json`. The logic can be replaced if needed. Since the transfer objects are pretty simple in this project, the restapi response type for the terminal has been changed to CSV string.

If neither of the `@ResponseBody` and `@RestController` annotations present and the response type is String, the servlet dispatcher identifies the result as a template or redirection. There is no default template resolver so one of the few supported libraries must be added to the classpath. This project uses Thymeleaf as such a library.

Chapter 4

Backend Implementation

4.1 API Endpoints for the terminal

The responses are serialized into CSV format with semicolon separator. It is easier to process it than for example a JSON.

Execute payment

Property	Value
Method	POST
Path	`/api/pay/{gatewayName}` The `gatewayName` path variable is used during the authorization.
Body	`card: String, amount: Integer, gatewayCode: String, details: String` card is the hashed card id, gatewayCode is the token, details contains the items of the payment separated by colons
Response	Any `PaymentStatus` enum values
Description	This endpoint handles payment requests, validate them, and return a status code accordingly.

Table 4.1: Pay Endpoint Details

Check account balance payment

Property	Value
Method	PUT
Path	`/api/balance/{gatewayName}` The `gatewayName` path variable is used during the authorization.
Body	`card: String, gatewayCode: String` card is the hashed card id, gatewayCode is the token
Response	`balance: Integer, loanAllowed: Boolean, allowed: Boolean` loanAllowed is whether the balance is allowed to go below 0, allowed is false when the account is locked
Description	This endpoint retrieves account informations if the code

Table 4.2: Check account balance endpoint details

Check terminal credentials

Property	Value
Method	PUT
Path	`/api/validate/{gatewayName}` The `gatewayName` path variable is used during the authorization.
Body	`card: String, gatewayCode: String` card is the hashed card id, gatewayCode is the token
Response	`OK` or `INVALID`
Description	It checks the gatewayName and the given token. Used by the *7002# command menu.

Table 4.3: Check terminal credentials endpoint details

Read the card id

Property	Value
Method	PUT
Path	`/api/reading/{gatewayName}` The `gatewayName` path variable is used during the authorization.
Body	`card: String, gatewayCode: String` card is the hashed card id, gatewayCode is the token
Response	`OK` or `INVALID` `OK` if the credentials and input structure valid
Description	Notifies the backend about a card id read. The backend attaches the id to the terminal. Administrators can initiate further actions form the GUI.

Table 4.4: Read card id endpoint details

Query the details of a named item

Property	Value
Method	POST
Path	`/api/query/{gatewayName}` The `gatewayName` path variable is used during the authorization.
Body	`card: String, gatewayCode: String` card is the hashed card id, gatewayCode is the token
Response	`valid: Boolean, name: String, price: Integer` valid if the code matches a stored item, name is the name of the stored item, and the price is how much it costs
Description	provides information about the named item for the terminals. It is allowed to use the * sign for telling the backend to add multiple amount of the given item.

Table 4.5: Query stored (named) item endpoint details

Check server status

Property	Value
Method	GET
Path	`/api/status`
Body	empty
Response	Five lines separated by semicolon
Description	Contains information about the backend version and the server time. Used by the *7001# command menu.

Table 4.6: Check server status endpoint details

4.2 Admin panel controllers and templates

The controllers in the admin panel have been separated based on which menu functions they implement. There is a separate controller for managing users and exporting for example. The endpoints in the controllers first perform their calculations and determine the return structure, and then tell which template the return model should be attached to. If a query is not authenticated, it will not even reach the controller.

Templates are designed so that a template resolver called Thymeleaf can replace the appropriate data and generate the necessary HTML elements. Thymeleaf templates conform to the XML standard, which makes them openable in a browser. This makes frontend and backend development separable.

An example of writing a variable called 'example' in a model to a <p> tag looks like listing 4.1.

```
<p th:text="${example}"></p>
```

Listing 4.1: Thymeleaf template engine usage example

Controller	Use-case	Associated Templates
`AccountsController`	list, create, modify accounts; start manual transactions; assign card to user	accounts, manual, manual-done, upload-money, upload-money-done, account-manipulate, assign-to-account
`AdminController`	show analytics; show audit log; show transactions	analytics, transactions
`ExportController`	export entities	export
`GatewaysController`	list, create, modify terminals	gateways, gateway-manipulate
`GuestController`	login page and redirections	login
`ItemsController`	list, create, modify, delete named items	items, item-manipulate

Table 4.7: Responsibilities of controllers

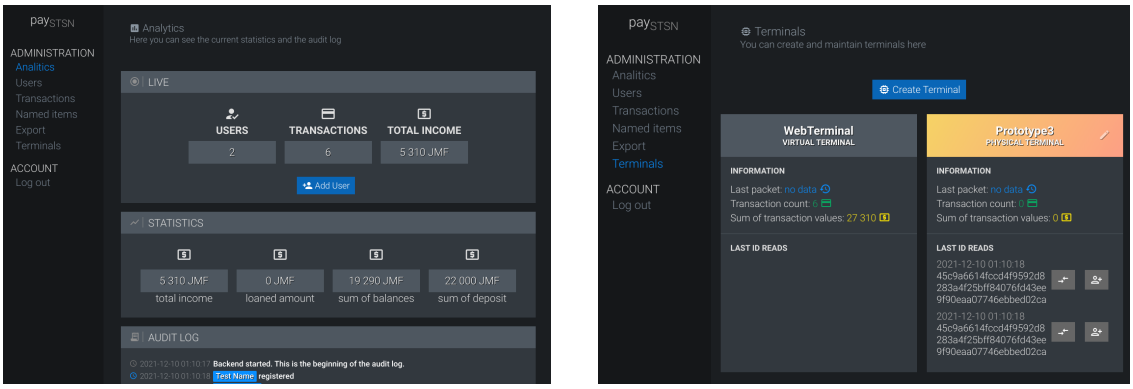

Figure 4.1: The analytic and audit log menu (left), active terminals (right)

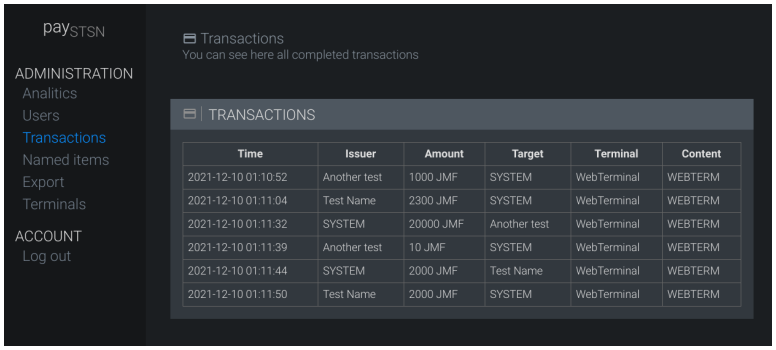

Figure 4.2: Transactions menu (right)

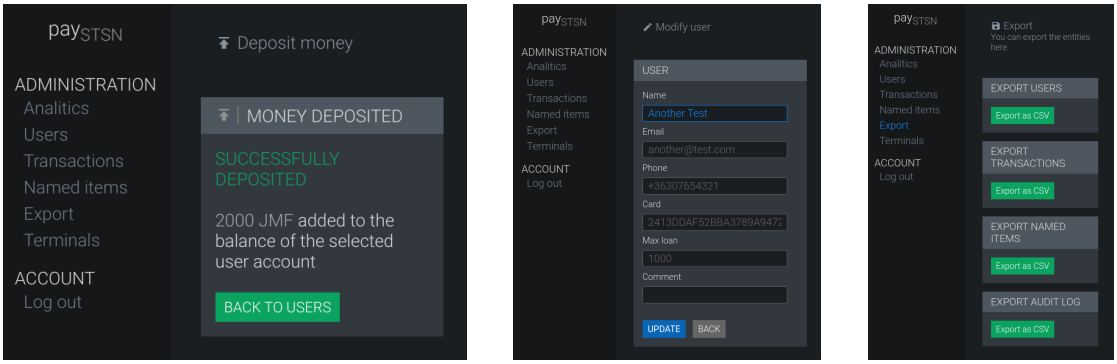

Figure 4.3: Deposit finished (left), update user details menu (center), export entities (right)

4.3 Business logic

The authentication and authorization process was configured in the SecurityConfig configuration component. The required authorization level of the endpoints was specified through their request path. Login and api endpoints does not require authentication but the admin panel (below `/admin/**`) does require administrator privileges.

Automatic export jobs are specified as well. There are rolling updates with 10 minutes and one hour period (those are overridden every time it saves) and a 30 minutes period

permanent save (that creates new export file every time it runs). The target of the exports are the account entities, the transactions, the named entities and the audit log.

4.4 Database Model

The database model consists of five entities.

1. The one for transactions contains information that were true during its creation. Such fields are the current time, the value of the selected items, the actual card id of the buyer and ids of the participating partners.
2. One entity to represent terminals with its name and token.
3. User accounts with personal information (for example: name, email address, phone), card id, balance and loan limit.
4. Administrator entities that are the users of the admin panel. Their name and hashed password is stored in the database as well.
5. Named items with their name, value and id.

They had foreign keys and relations during the design period but all of them got removed because it either left unused or looked easier to store redundantly. The core aspect was to make it fast and easily maintainable which did not required the model to be in any of the normal forms.

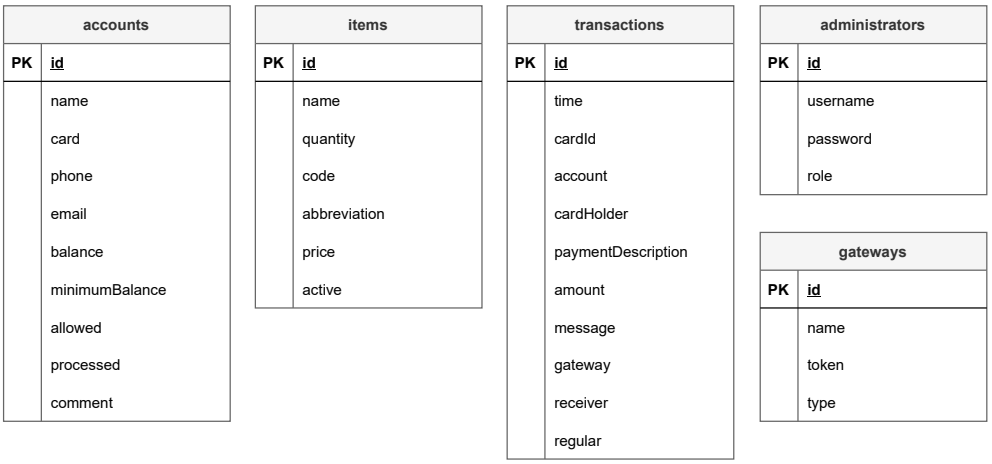

Figure 4.4: ER diagram of the database model

4.5 Backend Setup

The maven build tool is packed with the project. To build the backend the JDK 11 must be installed on the computer.

4.5.1 Build

To build a new binary from source execute the `install` goal of the maven build tool (as shown in listing 4.2).

```
./mvnw clean install
```

Listing 4.2: Command to build the applications

The binaries will be generated below the `target` directory.

4.5.2 Configure

The application can start without any additional configuration, but a lot of can be customized.

To apply a custom configuration value pass the configuration key to the application as launch arguments (starting with double dash `--key=value`) or create an `application.properties` file to the same folder as the JVM working directory.

Property	Default value	Description
server.port	8080	The port where the server will start to listen.
paymentsystem.admin.username	admin	The default admin username. If this username is not found in the database, the initialization script will create it.
paymentsystem.admin.password	1234	Password for the default admin user.
paymentsystem.gateways.file	config/gateways.csv	Terminals will loaded from this CSV file.
spring.datasource.url	jdbc:h2:file:./db/paymentsystem	The JDBC address of the database. It is configured for H2 file backed by default, but other databases are supported as well.
server.ssl.key-store	classpath:keystore/default.p12	The name of the keystore file for SSL
server.ssl.key-store-password	password	The password for the keystore
server.ssl.key-alias	paymentsystem	The key alias inside the keystore

Table 4.8: Most important configuration parameters

4.5.3 Launch

To start the application on the 443 port with maximum of 512 MB RAM usage the command from listing 4.3 can be used.

```
java -Xmx512M -jar payment-system.jar --server.port=443
```

Listing 4.3: Command to start the application from binary

The jar is packed with a web server and all of its dependencies.

Chapter 5

Firmware Design and Implementation

5.1 Tech-stack and dependencies

The firmware was written in C++ in top of the Arduino stack because ESP32 boards support it. To support any IDE the project uses the arduino-cli [1] and a makefile to cross-compile and flash.

The following 3rd party libraries must be installed via Arduino IDE or arduino-cli:

- `ESP32 board` – to support the ESP32 DEVKIT V1 30 pins board
- `lcdgfx 1.0.4` [2] – graphical functions and connection between the board and the SSD1331 display controller
- `MFRC522-spi-i2c-uart-async 1.5.1` [3] – connection between the board and the RC-522 RFID reader

5.2 Operation modes

5.2.1 Boot sequence

The following tasks are being executed during the boot phase:

1. The serial debugger initialization. This is the most important, because log messages can make troubleshooting easier. The baud rate is 115200.
2. EEPROM library initialization.
3. DisplayManager component initialization. (pin configuration, SPI periphery, display splash screen)
4. Keypad Component initialization. (pin configuration: GPIO, reset state in memory)

[1] `https://github.com/arduino/arduino-cli`
[2] `https://github.com/lexus2k/lcdgfx`
[3] `https://github.com/MakerSpaceLeiden/rfid`

5. Piezo buzzer initialization. (pin and PWM config, a beep sound)
6. Debug messages are being printed to the serial.
7. PermanentMemory module initialization. (read values from the flash)
8. Setup mode status check.

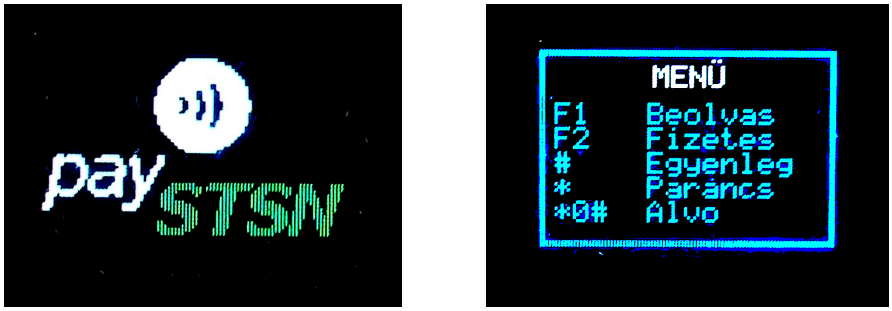

Figure 5.1: Splash screen – pay-station (left), main menu (right)

5.2.2 Setup mode

The EEPROM memory and keypad is monitored by the boot sequence. The setup mode is initiated when the `setupMode` bit is true or the '*' key is pressed during the boot sequence. The `setupMode` bit was set to true by default.

The setup mode activation is followed by the following four tasks:

1. Two beep sounds are played to indicate setup mode.
2. A new wifi access point is initiated with an SSID and password pair declared in the Firmware.h.
3. A web server is also initiated with a configuration form.
4. The ssid, the password and the IP address of the device is displayed by the screen of the device.

If a user connects to the network and enters the specified IP address, they will be able to edit the variables in the non-volatile memory through the web page.

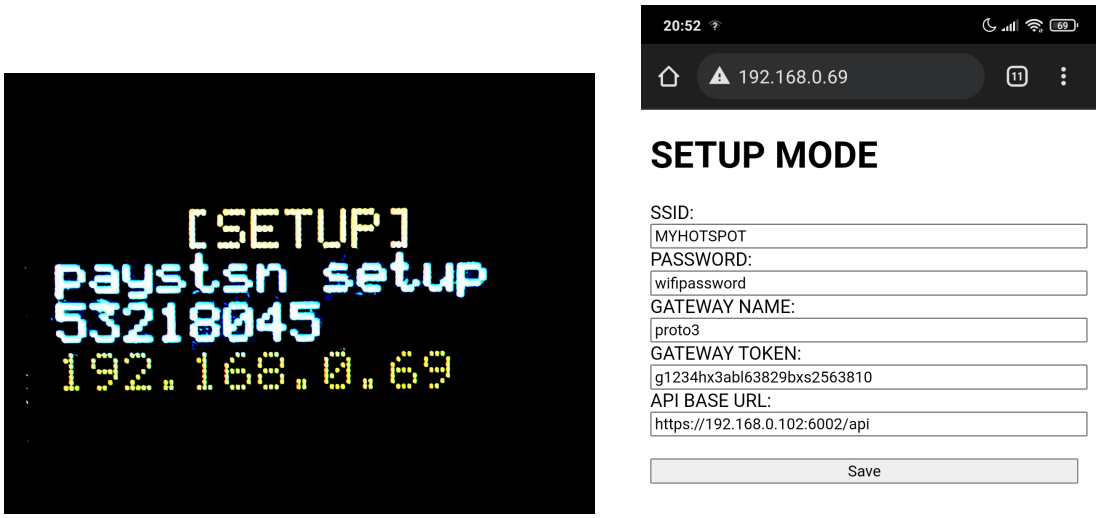

Figure 5.2: Setup mode parameters shown on the display (left), setup menu opened from a mobile web browser (right)

ESP32 comes with a 512 byte non-volatile memory. More than 71% of this memory is utilized. After enabling the settings, the 'Now reboot the device' will be displayed.

Variable name	Usage	Range start	Length in memory
state	Locked (0x02) and setup mode (0x01) bits	0	1 byte
pin code	unused, reserved for locking feature	1	8 + 1 bytes
ssid	Wifi SSID to connect to	10	32 + 1 bytes
wifi password	Wifi password	43	63 + 1 bytes
terminal name	Terminal name used in authorization	107	63 + 1 bytes
token	Token used in authorization	171	64 + 1 bytes
base url	Base URL to access backend api	236	128 + 1 bytes
			sum: 365 bytes

Table 5.1: Stored variables and its sizes, plus one bytes are for null terminators.

Setup mode makes the physical installation easier because the terminal is configurable without having to compile and flash the device.

5.2.3 Normal mode

If the setup mode is not triggered the following initialization tasks are being executed:

1. Wifi connection is being established and the `NetworkHelper` component is being initialized.
2. RfidReader component initialization. (pin configuration, SPI periphery, device initialization)
3. LED PWM configuration

Keyboard actions are being monitored periodically and events are being sent towards the active screen module.

5.3 Components

The business logic has been split into components to make the responsibilities separated and easily replaceable.

5.3.1 Keypad component

This component (`Keypad.cpp`, `Keypad.h`) handles the the key down and up events events. A key press event is triggered when the state of the key is changed to pressed and at least 100 *ms* passed from the last state change.

The detection algorithm checks the columns separately so it is possible to handle multiple key presses at the same time. The event listener function pointer is handled by the DisplayManager component.

5.3.2 DisplayManager component

This component (`DisplayManager.cpp`, `DisplayManager.h`) handles the OLED screen periphery. The `DisplayManager` class contains the utility methods that sends the draw commands to the OLED display.

5.3.3 Screens

The inputs of the different menus are similar, only their behavior differs. A pure virtual abstract class (`ScreenBase`) has been created that handles these inputs and the menu lifecycle. Depending on the implementation, it can contain internal states, respond to keystroke events, and run code when it is created, idle, or closed.

Drawing and state initialization operations are typically locaded in the `onActivate()` function. The `onKeyPressEvent(char)` function alters the initial state or changed the active screen. The `onIdle()` runs every 10 *ms* or so, and it is mainly used for entering sleep mode after one minute of inactivity. The `onDeactivated()` functions contains cleaning functionalities for example listener cleanups.

The orientation of the screens are also determined here. Menus aimed for the operators are displayed normally, but menus for customers are rotated by 180° so they don't have to read upside down texts.

The main loop periodically calls the idle and `handleKeyEvenets` functions.

5.3.4 RfidReader component

This component is responsible of handling the RFID reader periphery and its operations. When needed it reads the RFID tags nearby the antenna and executes a listener function while passing the hashed version of the tag id.

The purpose of the hashing the cards is to make them the same length and make identity theft attacks harder. A typical tag id lengths is 4 or 7 bytes. It uses the SHA256 algorithm. Adding salts to the hash might increase the security but adds more complexity to the system because the salts have to be distributed to all the terminals to work properly.

Setting the gain of the antenna and self checks are also possible using this component.

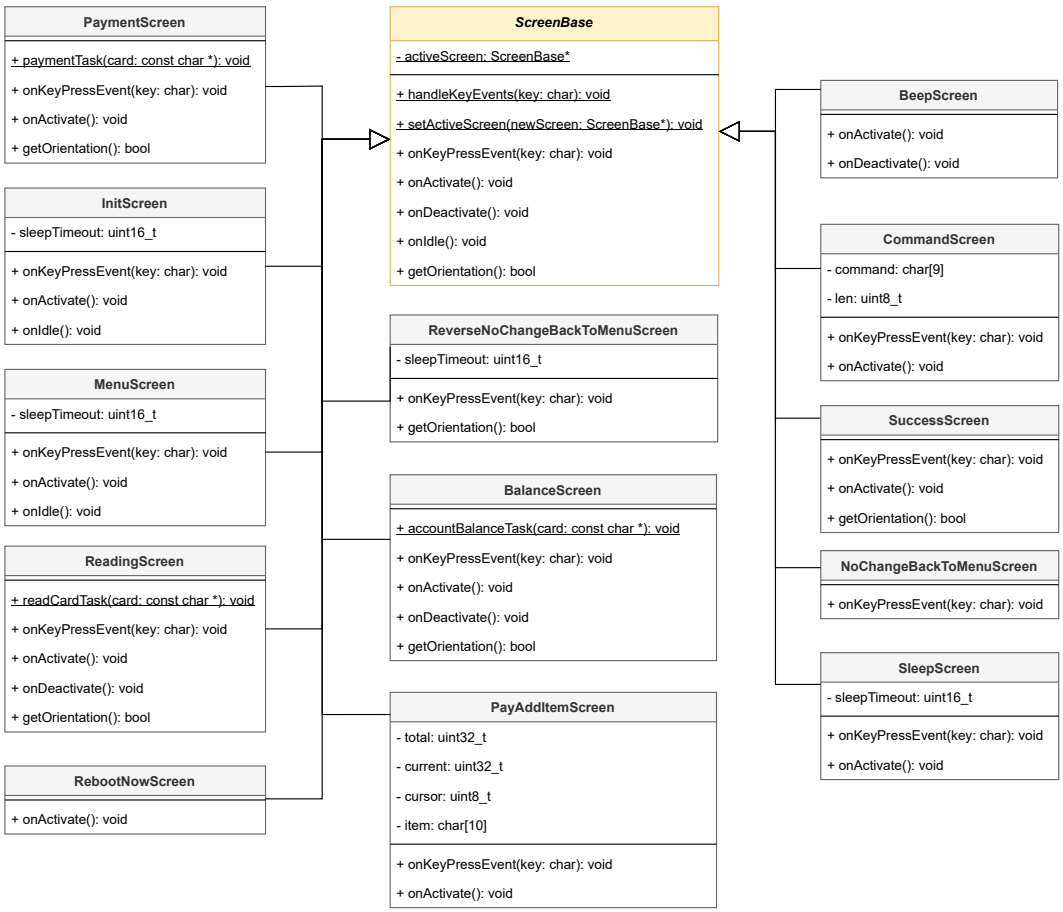

Figure 5.3: Class hierarchy of the Screen classes

5.3.5 NetworkHelper component

This component is responsible of handling the connection between the terminal and the backend via http requests. Before executing the request the methods checks whether the WiFi connection is active and tries to connect if not. The response data is passed to the associated functions. When a http request fails an error message is shown describing the possible cause of the error.

5.3.6 SetupMode component

This component is responsible of creating and maintaining a web server when the setup mode is activated.

5.4 Menus

The three most important use-cases for a regular user point of view are `Read Card Id`, `Show Account Balance` and `Make Payment`.

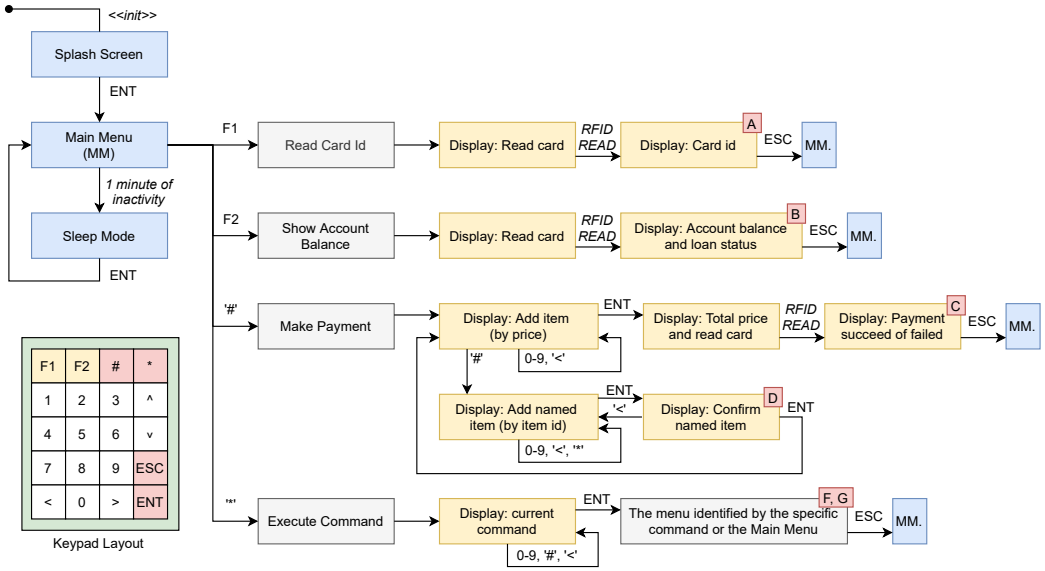

Figure 5.4: Menu structure and transitions

Red boxes in figure 5.4 indicates that a HTTP request is required to complete to show the result. Exiting to the main menu is possible in most of the states via pressing the `ESC` button.

5.4.1 Read Card Id

Triggered by the `F1` button from the menu or the `*103#` command.

In order to attach an RFID tag to a user account the hash must be known by an administrator. This menu reads a card, hashes the id, prints the first few characters to the OLED display and sends a http request to the backend.

After the `PUT /api/reading` (figure 5.4 A) endpoint is triggered, the readings are displayed in the admin panel. Administrators can attach the id to existing user account or create a new one.

5.4.2 Show Account Balance

Triggered by the `'#'` button from the menu or the `*102#` command.

Users sometimes want to know how much money they have left. This menu reads a card and sends the hashed id to the `PUT /api/balance` (figure 5.4 B). The response structure contains the account balance, whether going below 0 is supported and whether is the account blocked. This information then printed to the display.

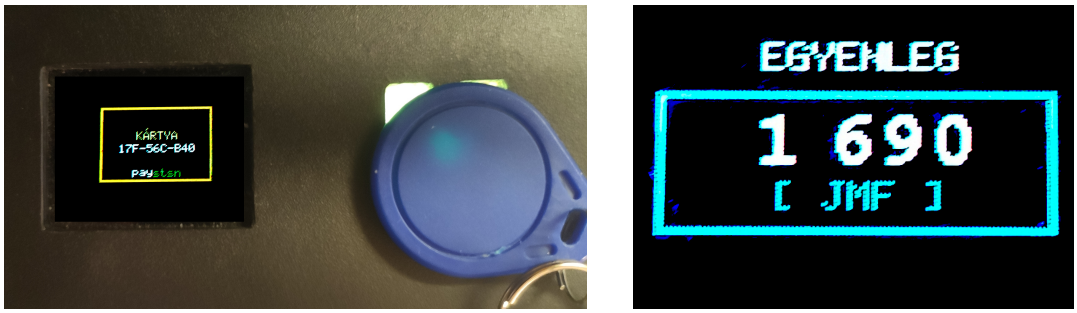

Figure 5.5: Card reading screen (left), account balance screen (right)

5.4.3 Make Payment

Triggered by the `F2` button from the menu or the `*104#` command.

A payment consists of at least one item. There are two ways to add items to payments.

- Add them by it value: simply type the integer value using the keypad and hit enter
- Add item by name: press the `'#'` button and type the id if the item. Hit enter to retrieve the value from the server via the `POST /api/query` endpoint (figure 5.4 D).

The total amount is shown in the bottom left corner. Hitting enter again will start the checkout process. The full amount is shown in he screen and the RFID readers waits for a tag to read. When a read is successful the hashed id and the payment details is sent to the serve via the `POST /api/pay` endpoint (figure 5.4 C).

Figure 5.6: Snapshots of the payment process menus

The result is printed to the screen. A green tick is shown (and the backlight LEDs turns to green) when the payment was successful. A red cross is shown with an error message (and with red led backlight) when it failed.

Result Code	Successful	Meaning
ACCEPTED	yes	The transaction is processed.
INTERNAL_ERROR	no	An unexpected error happened while executing the payment transaction. Check backend logs for more details.
NOT_ENOUGH_CASH	no	There is not enough funds in the account.
VALIDATION_ERROR	no	The read card id is not attached to any of the accounts.
CARD_REJECTED	no	The card is blocked. It might indicate that this card is lost or stolen.
UNAUTHORIZED_TERMINAL	no	The credentials of the terminal is invalid.
else	no	Unexpected error type
no response	no	Communication error

Table 5.2: Payment result codes and its meanings

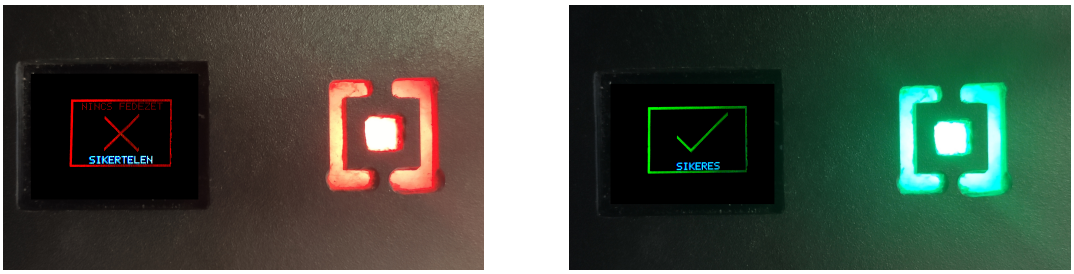

Figure 5.7: Transaction declined menu (left), transaction accepted (right)

5.5 Commands

Every functionality is available trough the command menu. This menu provides useful actions for debug and maintenance purposes.

Category	Command	Action
Menus		
	*0#	Sleep mode
	*100#	Main menu
	*101#	Show splash screen
	*102#	Account balance menu
	*103#	Read card menu
	*104#	Pay menu
Testing		
	*7000#	Dump firmware info
	*7001#	Server status check without auth
	*7002#	Check connection and authentication
	*7003#	Empty payment: payment test
	*7004#	LED color test
	*7005#	PWM LED spectrum fade test
	*7006#	Buzzer self test
	*7007#	RFID self test
Resets		
	*8001#	Reset RC522
	*8002#	Re-init RC522 (hard reset)
Advanced Settings and Debugging		
	*9101#	Set RC522 antenna gain to 18 dB
	*9102#	Set RC522 antenna gain to 33 dB
	*9103#	Set RC522 antenna gain to 48 dB
Factory modes		
	Hold * on boot	Enters setup mode
	*707172#	Forces setup mode
	*909192#	Resets the memory and sets setup mode flag

Table 5.3: All available commands

Server status check uses the `GET /api/status` endpoint (figure 5.4 F) and authentication check uses the `PUT /api/validate` endpoint (figure 5.4 G)

Figure 5.8: Command menu (left), server status menu (center), connection test menu (right)

5.6 Setup build environment

First generate the configuration file, and then add the espressif index url 'https://dl.espressif.com/dl/package_esp32_index.json' into the `board_manager -> additional_urls` array inside the configuration file. When its finished update the index and install the ESP32 board support.

```
# Create arduino-cli config file
```

```
arduino-cli config init
# add the url to the config file
arduino-cli core update-index
# install ESP32 board
arduino-cli core install esp32:esp32
```

Listing 5.1: Arduino-cli environment config setup

This step is only required on linux. Add your user to the `tty` and the `dialout` group. On others operating system additional permission configuration might be required. You must log out and then in to your user after the usermod commands finished.

```
sudo usermod -a -G tty <user>
sudo usermod -a -G dialout <user>
# log out and in now
arduino-cli board list
```

Listing 5.2: User permission config

To complete the initialization install the required libraries.

```
arduino-cli lib install "lcdgfx"
arduino-cli lib install "MFRC522-spi-i2c-uart-async"
```

Listing 5.3: Installation of the required libraries

5.7 Build firmware

To fully compile and build the project execute the commands from listing 5.4. Note that assigned port might be different than `/dev/ttyUSB0` for other system and operating systems.

```
# Cleaning the output directory
rm output/* -rf
# Compile the sources
arduino-cli compile -b esp32:esp32:esp32 src/src.ino -v --build-path output/
# Upload the compiled binary
arduino-cli upload -b esp32:esp32:esp32 -v --port /dev/ttyUSB0 --input-dir output/
```

Listing 5.4: Commands to build the sources with aurdino-cli

To connect to the device for debugging a serial console application is required. During the development the `minicom` was used. An example of the usage is shown in listing 5.5. Note that it must be closed before a software update.

```
minicom -b 115200
```

Listing 5.5: Starting minicom with baud rate 115200

Chapter 6

Hardware Iterations

6.1 Prototype 1

An important aspect was to make the physical part cheap but also functional. The selected parts were the following:

- `ESP32 DEVKIT V1` with 30 pins for controller. It has a lot of pins, a relatively fast CPU and an integrated WiFi module. The dev kit also comes with a debugger which makes it easy to upload new firmware versions. The most important part of the component is the `WROOM-32` [12] module.
- `RC-522 RFID reader` This is one of the cheapest options and it has fairly well documented library. The chip on it is the `MFRC522` [10] module.
- `0.95" RGB OLED Display SSD1331` [6] It is really small in size but at least it is colorful.
- `Membrane switch 4x5` It is waterproof and has 20 different keys.
- A `piezo buzzer` For the iconic beep sound.
- Some resistors ($5 * 10\ k\Omega$, $2 * 100\ \Omega$, $150\ \Omega$) and jumpers.

It was first assembled on a breadboard. This made it easier to try and test the peripherals, as minor modifications could easily be made. The power supply was solved by connecting the device to a PC via the debugger.

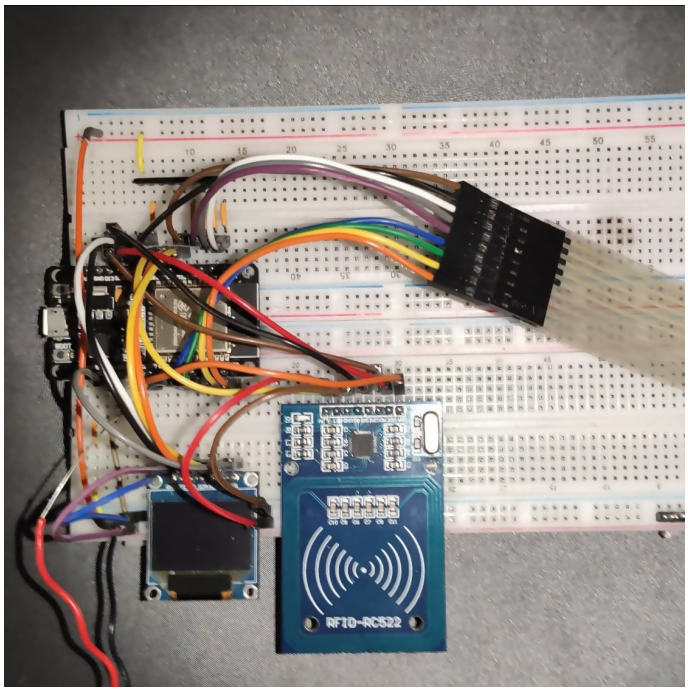

Figure 6.1: Prototype 1 assembled on a bread board

Much of the firmware was created during this time. It was later dismantled and the same parts were used for the second prototype.

6.2 Prototype 2

The parts were soldered to a prototype board and the jumpers were replaced with plain wires. The whole thing was placed into an instrument box made out of plastic. Bolts, nuts and hot glue were used for fastening. The power was supplied by a power bank connected to the debugger.

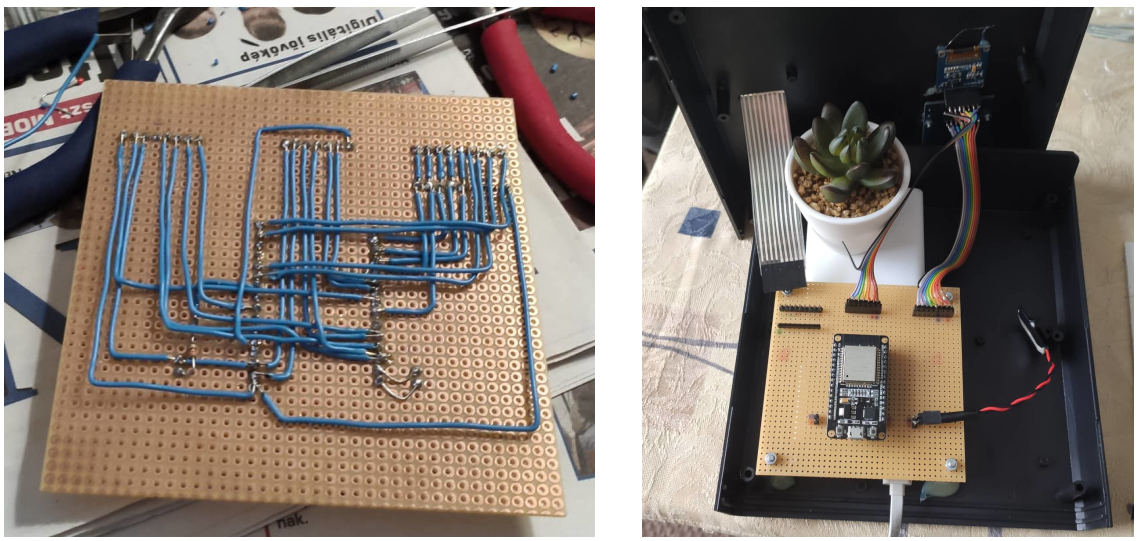

Figure 6.2: Prototype 2 assembled on a test board

The end result was not really aesthetic but functional.

6.3 Prototype 3

Things observed during testing of the second prototype were used to build the new prototype. A larger hole for the screen was drilled in the box and a waterproof glass foil was placed in front of the screen. The glass foil was attached to the box with instant glue.

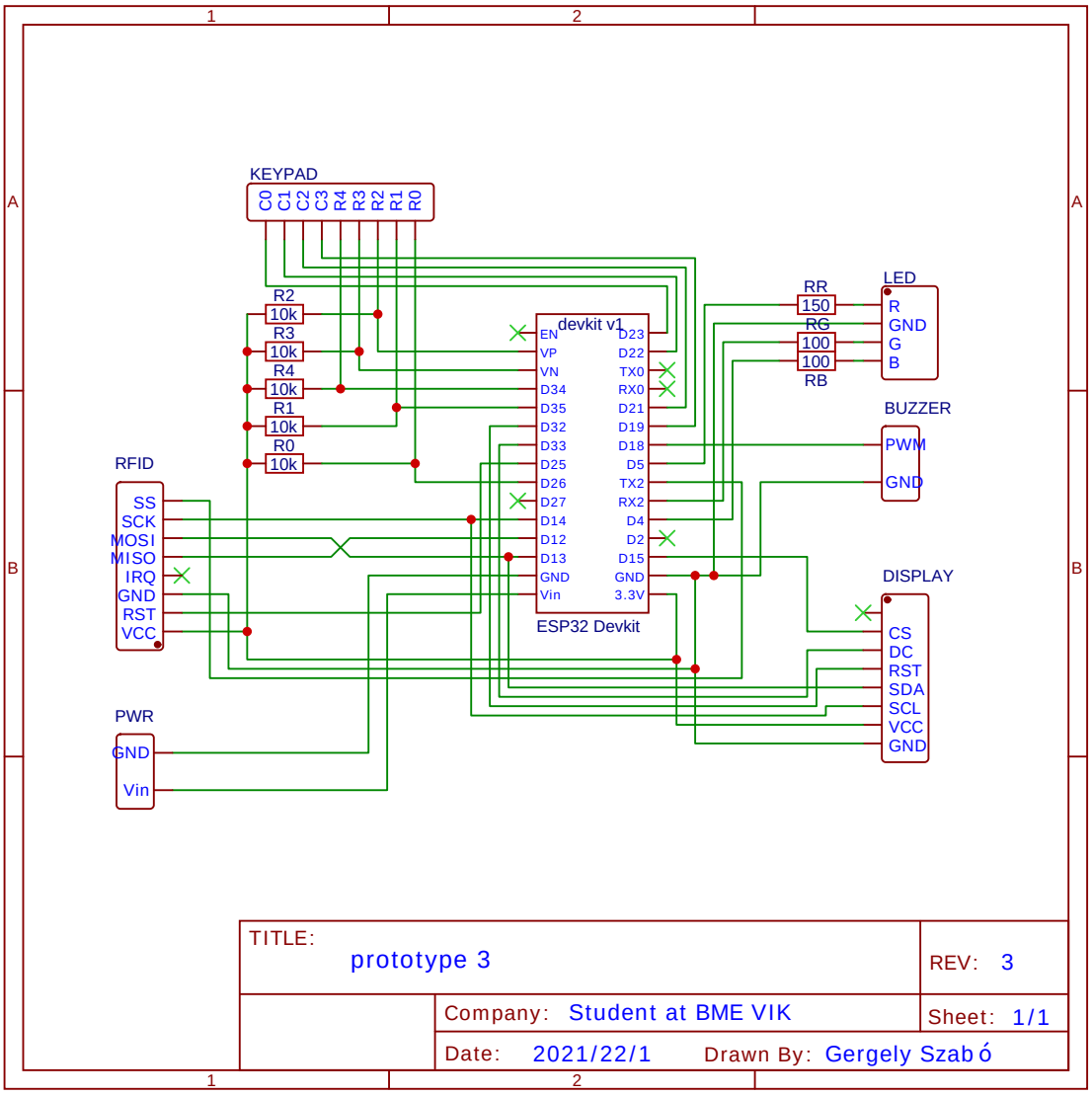

Figure 6.3: Prototype 3 - Schematic

The box was machined using CNC. The gcode file was created using `CamBam` [1]. A new hole was drilled for the feedback LED which was a new thing in this prototype. The RGB LED can be controlled using PWM and its color indicates the status of the transaction. The screen hole and the keypad wiring hole is relocated to match the new (smaller) size of

[1] `http://www.cambam.info/`

the box. The plan was to put creamy plexi glass next to the led to scatter its light better, but eventually a thin layer of hot glue was put there for that purpose.

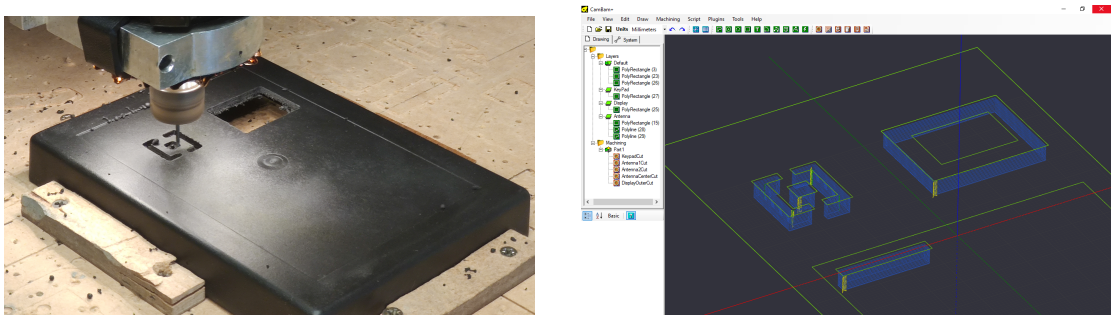

Figure 6.4: Prototype 3 - CNC machining (left), CamBam design view (right)

The status led lights green if the transaction was a success, red if it failed and yellow when the RFID reader is enabled. It is turned off otherwise.

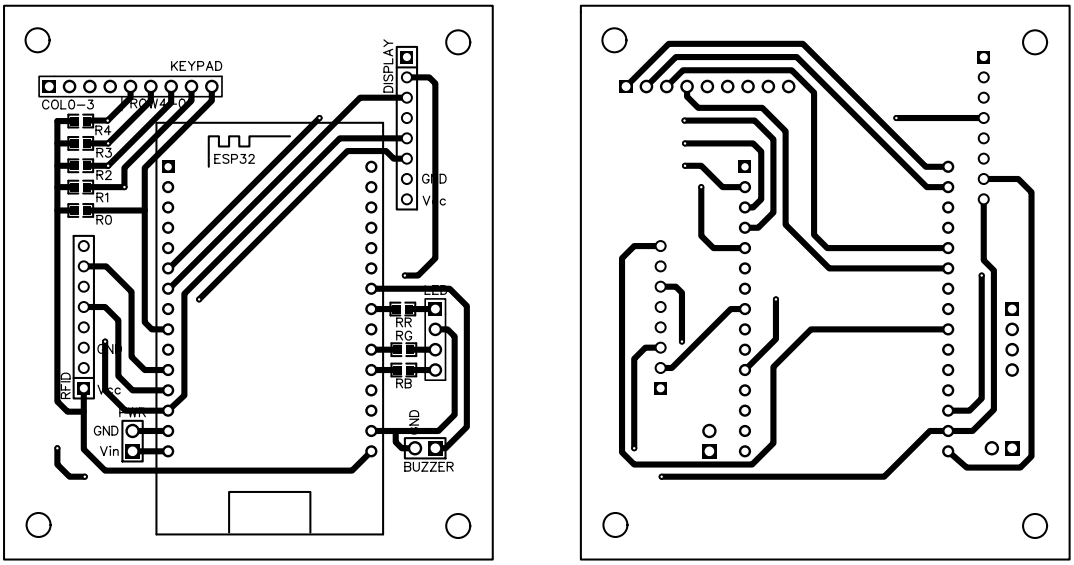

Figure 6.5: Prototype 3 - PCB top layer + silk (left), PCB bottom layer only (right)

Another development was that the components were connected in a custom made printed circuit board. The schematic (see figure 6.3) and the gerber file was created with `EasyEDA` [2]. The board was ordered from a PCB manufacturer.

[2] `https://easyeda.com/`

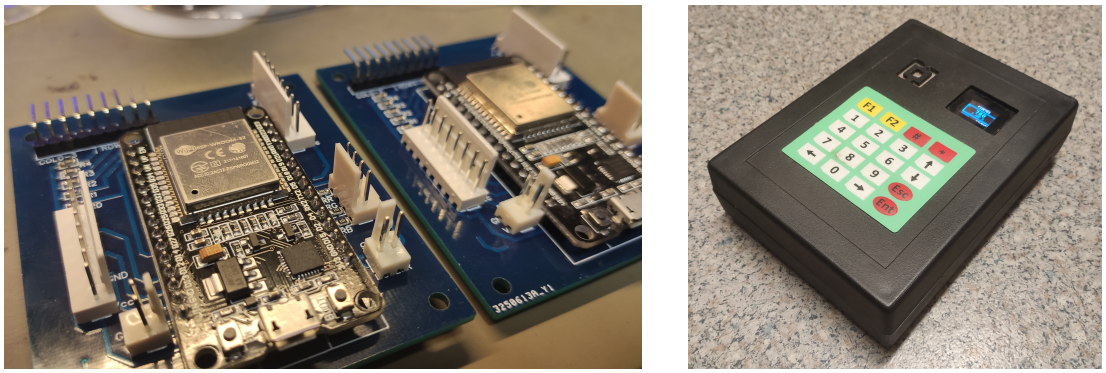

Figure 6.6: Prototype 3 - assembled board (left), assembled device (right)

The power supply pins have been routed out so there is no need to connect power supply via the debugger anymore.

Chapter 7

Security Considerations

7.1 Admin panel security

The admin panel uses spring security with username and password authentication, and authorization. The accounts stored in the same database as the transactions and other entities.

Passwords stored as BCrypt hashes. BCrypt uses salts by default and it is still considered a secure hashing algorithm. The strength parameter is configurable between 4 and 31.

The periodic export makes it recoverable in case of database failures. During one of the test a power outage caused the database to corrupt but the periodical saves were in good condition.

The system also saves audit logs for fraud detection and debugging.

7.2 Investigate possible security fixes

To improve the security the following things could be done.

7.2.1 Improve password security

Enforce minimum password length and special characters

Strong and uncommon passwords can help against dictionary and broute-force attacks.

Enforcing does not directly improve the security. Administrators can set complex passwords without this requirement as well. There's no perfect restrictions to protect against non-complex and common passwords.

Add expiry date for passwords

This method is not very useful if the average operation period of the system is one to three days.

Captcha

Captchas are randomly generated images with humanly readable characters and complex noise. These distorted words are purposely hard to recover with computer vision but easy to read with human eyes. The purpose of captchas is to protect against brute-force attacks.

While captchas are powerful, the common providers requires internet connection to work. A self-hosted captcha solution would satisfy the requirements of the projects and improve security.

SSO

A single sing-on service where the authentication and sometime authorization parts are provided by a third-party entity. Some common SSO providers are big tech companies such as Facebook [1], Google [2] and Github [3] just to mention a few.

It's core advantage is that the passwords are stored in the providers database so the provider is responsible of securing the credentials. SSOs are easy to integrate with spring, but requires internet connection, so not suitable for this project.

There are self hosted alternatives but they does not come with the advantage of being a trusted third-party.

7.2.2 Filter attack sources

Create IP whitelist for accessing the backend

Logging in from outside of these IP regions would be prohibited. It is possible to check it on backend side but it is better to be controlled by the router.

Block DB access outside the server itself

The (default) H2 database can't handle multiple connections, but the project is compatible all sorts of relational databases. Make sure that the access levels and zones are configured correctly.

Use at WPA2 PSK for wireless security

This way the packets will not be readable without the wifi password and it is harder to interfere with the communication in general. Use isolated wireless network and use it for this purpose only.

[1] `https://developers.facebook.com/docs/facebook-login/`
[2] `https://cloud.google.com/architecture/identity/single-sign-on`
[3] `https://docs.github.com/en/developers/apps/building-oauth-apps/authorizing-oauth-apps`

7.2.3 Two-factor authentication

Cryptographic two-factor authentications

Cryptographic two-factor authentications are based on time and a shared secret. One example could be the TOTP [7] (Time-Based One-Time Password Algorithm, RFC6238).

This method is feasible because it doesn't require internet connection. The secret sharing procedure can be done via a QR code read. The only factor left is time which needs to be synchronized between the auhenticator device (for example a mobile phone with the `Google Authenticator` [4] application installed) and the backend server.

2FA using SMS or email messages

While this solution improves security in an order of magnitude, without active internet connection it is not feasible.

7.2.4 Connection between the backend and the terminal

The API endpoints validates the incoming packages and only accept them when the gateway name and associated token is valid.

The endpoints are secured with SSL unless in is turned off manually. The connection is encrypted between the terminal and the backend. If the SSL is turned off the name and the token can easily be stolen.

7.3 Security of RFID tags

The `RC-522` library supports multiple tag types. The two tested types are the `Mifare Classic 1K` and the `Mifare Ultralight EV 1`.

These models comes with 4 or 7 bytes [10] of manufacturer programmed unique ids. Even the 4 byte uid is safe enough against brute force attacks because only place when the attacker can check the validity is the terminal and the terminals are controlled by operators. Attackers would become suspicious after 2 or 3 tries.

The library does not support AES encryption currently. This makes the tags vulnerable to certain card duplication attacks.

These tags can store additional data. To improve security some other id could be stored on the tag. The downside of this is that it makes the reading process slower and the token must be written to the tags.

7.4 Phyisical security instructions

- Make sure terminals will not be left unattended.
- The terminals are not waterproof. It is equivalent of a IP44 protection level but not certified. It means that it is resistant to intrusion by objects up to 1 mm in diameter (eg. screws, cables) and resistant to water splashing from any direction.

[4] `https://play.google.com/store/apps/details?id=com.google.android.apps.authenticator2`

- Place the access point and the backend server physically away from users.

7.5 Security conclusion

While there are possible and feasible improvements available – with a dedicated wireless network, secure SSL connection and complex tokens and passwords – the system is considered *safe enough* against the most common attack vectors.

Chapter 8

Deployment

This chapter introduces in detail how to use the whole system.

8.1 Preparations

- Build the software components according to sections Backend Setup and Build firmware.
- Decide whether the system is used in a prepaid, post-paid or mixed mode. Send information about the system to the attenders accordingly. For example send them how can they transfer money to their virtual account if the the prepaid solution is selected.
- Show the bartenders how can they use it if they are not familiar with the system. The available menus described on the section 5.4

8.2 Infrastructure setup

Some of the steps can be done in different order.

1. Build and upload the latest firmware version.
2. Generate a unique token for each terminal.
3. Build and start the web server. Configure administrator accounts. Add terminals using the admin panel. Make sure the server has safe power supply.
4. Setup the WiFi access point. See subsection 7.2.2 for security advice.
5. Charge the batteries.
6. Enter setup mode (see subsection 5.2.2) and update SSID, WiFI password, terminal name, and the token value. Then reboot the terminals.
7. Optional: Test the connection with the `*7002#` command.
8. Optional: Setup named items using the admin panel.
9. Register attenders, attach tag ids to them. Deposit money to user accounts.

8.3 During the event

- Give participants opportunity to deposit additional resources during the event.
- Optional: Save the automatically created backups regularly to another drive.
- Check the charge of the batteries multiple times a day. To approximate the required battery size see section 9.1.

8.4 Post event tasks

- Close accounts. Depending on the scheme (prepaid or post-paid) collect money or return the unused amount to the user.
- Save the transaction log and stop the system.

Chapter 9

Power Consumption Measurement

A measurement was conducted to approximate power consumption, which is directly tied to the required battery size.

- Location: Schönherz Zoltán Dormitory - Schönherz Elektronikai Műhely SCH-320
- The power bank is a `Xiaomi Mi Power Bank 2C` with 20000 mAh cell capacity
- The subject of the measurement was the `third prototype` (section 6.3) of the current project
- The multimeter is an `Extech EX350` [1]
- The oscilloscope is a `Rigol DS1054Z` [2]

9.1 Measurement 1 - Multimeter

The `multimeter` has been configured to act as a current meter. It was then attached to the positive pole of the `power bank` and the `VIN` pin of the prototype. The negative pole of the power bank is attached to the GND pin of the prototype. Current values were measured in several operating modes. The measured values were determined by sampling for at least half a minute in each mode.

[1] `http://www.extech.com/products/resources/EX35x_DS-en.pdf`

[2] `https://beyondmeasure.rigoltech.com/acton/attachment/1579/f-0504/1/-/-/-/-/MSO1000Z_Datasheet.pdf`

Figure 9.1: Measurement 1 setup - Power bank (left), Prototype 3 (center), Multimeter (right)

Operation Mode	Power consumption
Display sleep (main menu without display)	$5\ V * 73\ mA = 365\ mW$
Main menu	$5\ V * 77\ mA = 385\ mW$
Indicator LED and RFID is on with (gain 48dB)	$5\ V * 91\ mA = 455\ mW$
Indicator LED and RFID is on with (gain 33dB)	$5\ V * 100\ mA = 500\ mW$
Indicator LED and RFID is on with (gain 18dB)	$5\ V * 100\ mA = 500\ mW$
Setup mode	$5\ V * 159\ mA = 795\ mW$

Table 9.1: Power consumption measurement values per mode

A 4 mA difference between the main menu and sleep mode was expected. Higher consumption of setup mode was also expected, as the device acts as an Access Point and also runs a web server. An interesting anomaly is that the RFID module was used 9 mA more current if it was not set to the maximum gain. The gain level was manually tested with an RFID token and appeared to be valid.

Considering that the main menu is used 90 percent and the reading menu (indicator LED and RFID is on) is used 10 percent of the time: a fully charged 20000 mAh power bank could last for 255 hours supplying power to this device.

9.2 Measurement 2 - Oscilloscope

In order to find out how uniform the power consumption of the device is, another measurement was made using an oscilloscope.

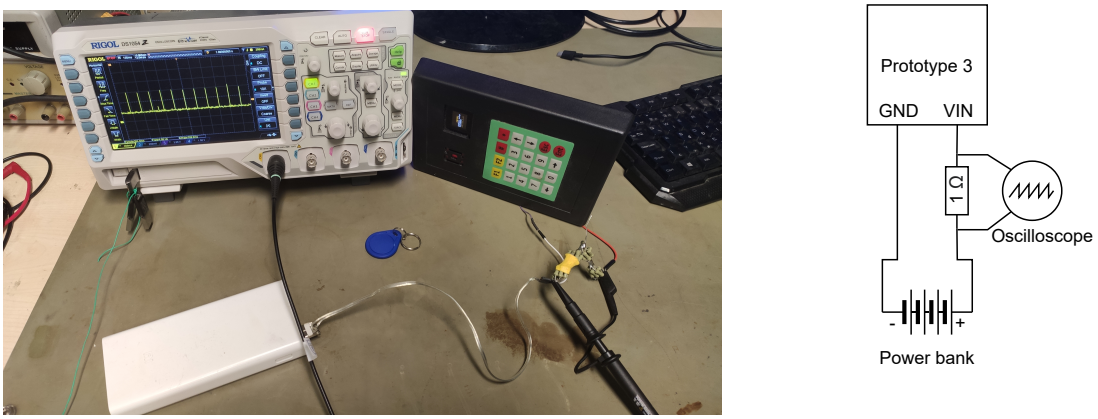

Figure 9.2: Measurement 2 setup - Power bank (bottom left), Oscilloscope (top left), Prototype 3 (top right), Resistor (bottom left)

The measured data correlated roughly with the values in the first measurement. It was expected that the measured values in current would not be as accurate as the shape of the signal was observed here. What was clear in this measurement is that there were occasional peaks in the current.

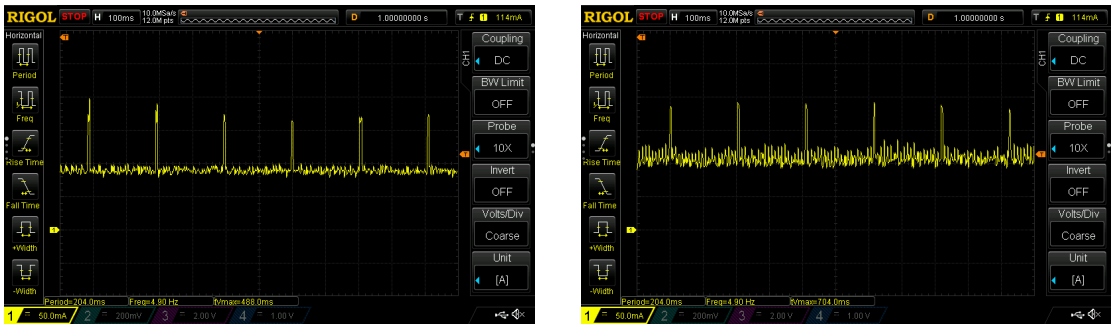

Figure 9.3: Used current over time (grid: 50 mA | 100 ms) - Main menu (left) and Reading menu (right)

Non-periodic peaks in power can be observed in the 'one active connection' (figure 9.4 right) image due to the increased computing power and the maintenance of the WiFi connection compared to the one without active connection (figure 9.4 left).

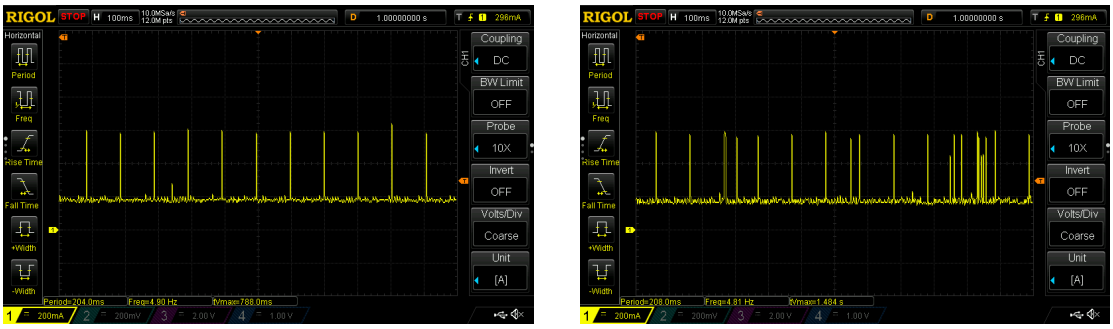

Figure 9.4: Used current over time (grid: 200 mA | 100 ms) - Setup mode without active connection (left) and Setup mode with an active connection (right)

Chapter 10

Electromagnetic Compatibility (EMC) Measurement

A measurement was performed to verify the electromagnetic compatibility of the device.

- Location: BME building V1 4th floor
- Receiver: `Rohde&Schwarz ESPI Test Receiver` 9 kHz...3 GHz
- Test cell: `ETS EMC GTEM` [8]
- Test subject: `Prototype 3` (section 6.3) with an internal battery (`Xiaomi Mi Power Bank 2C` 20000 mAh)

Figure 10.1: Environment (left), Inner view of the GTEM cell (right)

GTEM Test Cell The purpose of using a GTEM cell (figure 10.1) is to obtain a *homogeneous field strength* of adequate quality created in a properly shaded area. The cell can be used to test the immunity of electronic equipment and to analyze its radiated electromagnetic emissions in the frequency range of about 9 kHz to 5 GHz. The cell is a rectangular hollow pyramid-shaped 50 Ω transmission line. [8]

Class	Frequency range	Level limit
A	$[30, 230]\ MHz$	$40\ dB(\mu V/m)$
	$(230, 1000]\ MHz$	$47\ dB(\mu V/m)$
B	$[30, 230]\ MHz$	$30\ dB(\mu V/m)$
	$(230, 1000]\ MHz$	$37\ dB(\mu V/m)$

Table 10.1: Evaluation of methods on the basis of the set criteria

The criteria (table 10.1) was set according to the `IEC 61000-4-20:2010` [3] standard.

	reference value (0 dB)
	47 dB
	40 dB
	37 dB
	30 dB

Table 10.2: EMC diagram legend

Note The markings are only *approximate*.

10.1 Device orientation 1 - Side

Conditions: The device is turned on, the RFID reader is active, the WiFi module is turned on. The *left* side of the device points in the direction of capture (as shown on figure 10.1 right).

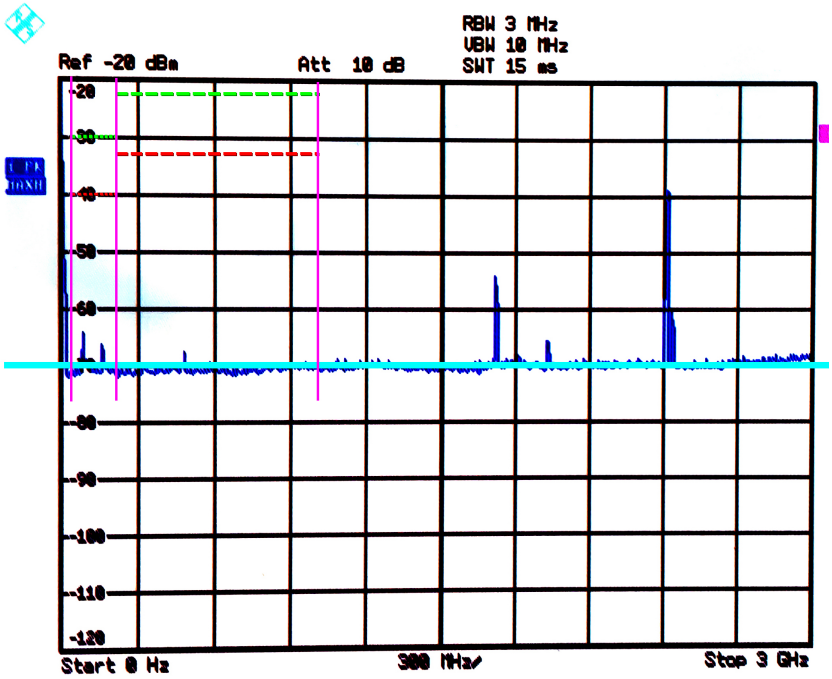

Figure 10.2: Device is turned on and its RFID is active - capture: left side [0, 3] GHz

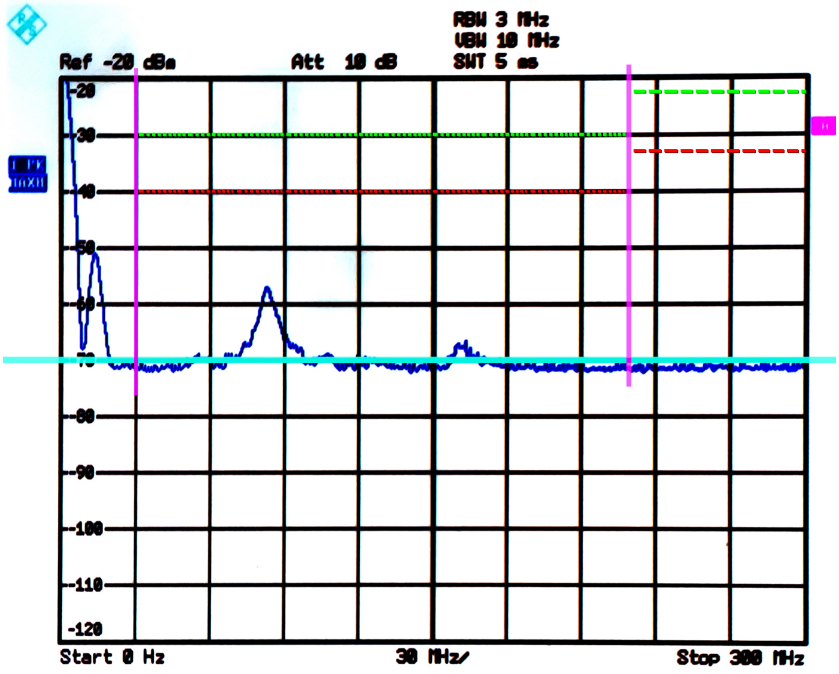

Figure 10.3: Device is turned on and its RFID is active - capture: left side [0, 300] MHz

10.2 Device orientation 2 - Front

Conditions: The device is turned on, the RFID reader is active, the WiFi module is turned on. The *front side* of the device points in the direction of capture.

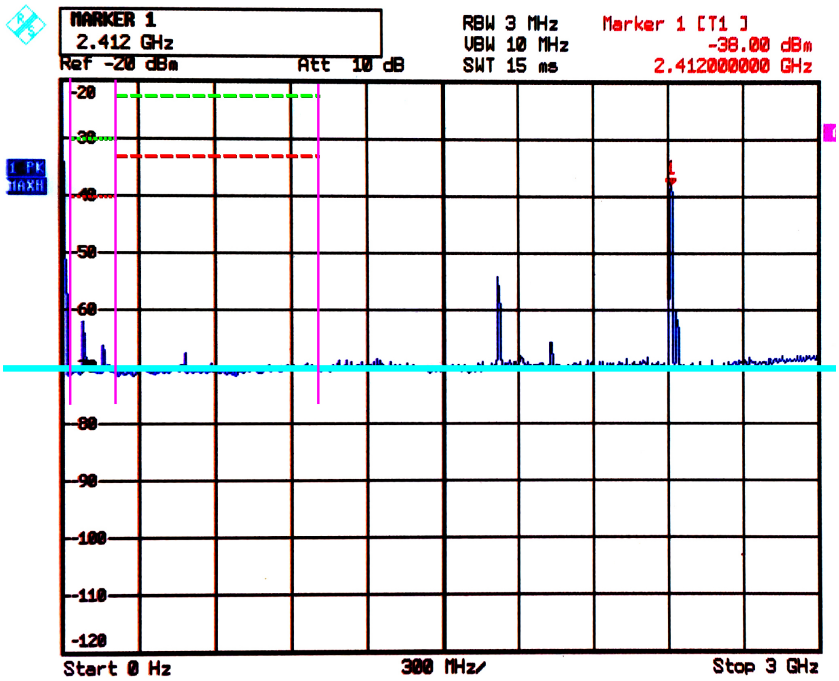

Figure 10.4: Device is turned on and its RFID is active - capture: front side [0, 3] GHz

10.3 Device orientation 3 - Standing

Conditions: The device is turned on, the RFID reader is active, the WiFi module is turned on. The device points with its *top side* towards the direction of capture.

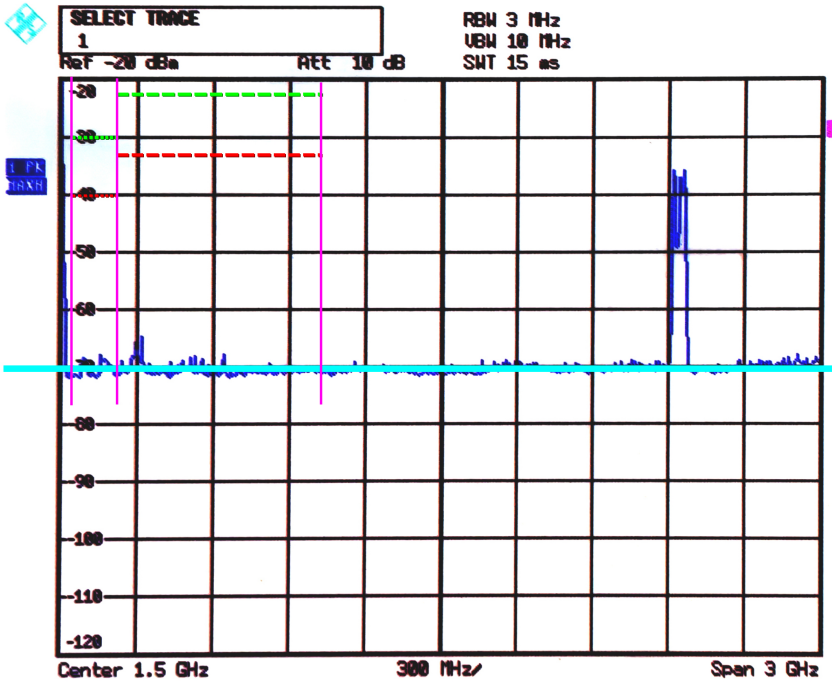

Figure 10.5: Device is turned on and its RFID is active - capture: standing [0, 3] GHz

It has been clearly shown by the measurement (figure 10.4) that the WiFi module is trying to communicate on multiple channels.

10.4 Reference measurement

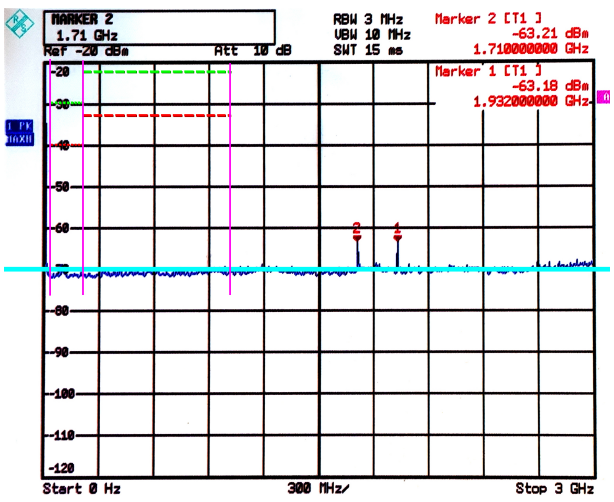

Figure 10.6: The STEM cell without the device [0, 3] GHz

During the reference measurements (without the device) two peaks were measured, one with 1.71 GHz and one with 1.932 GHz frequency. These frequencies were also measured during other (see figure 10.4) measurements.

10.5 Conclusion

Electromagnetic radiation emissions of the device were everywhere within the limits. The device intentionally broadcasts in two ranges.

RFID module implements the `ISO/IEC 14443-2` standard. "The frequency of the RF operating field shall be 13.56 MHz ± 7 kHz." [4] This is an ISM band and it is reserved for civilian use. The measurements were showed that the radiation of the RFID reader were not distinguishable from the noise from the distance of the measurement. It was expected since its maximum operating range is about $10cm$.

Above the 1 GHz range, only the WiFi module operates. The WiFi module implements the `IEEE 802.11` standard [1] and uses only the ISM $2.4 - 2.5$ GHz free frequency range.

Acknowledgements

Thanks to Dr. Zoltán Tóth for being my consultant and helping me perform the EMC measurement.

I am very grateful to the SEM [1] electronics workshop for being able to use their CNC machinery, soldering stations, multimeter and oscilloscope for free.

[1] Schönherz Elektronikai Műhely

List of Figures

List of Tables

Bibliography

[1] Ieee standard for information technology–telecommunications and information exchange between systems - local and metropolitan area networks–specific requirements - part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. *IEEE Std 802.11-2020 (Revision of IEEE Std 802.11-2016)*, pages 1–4379, 2021. DOI: 10.1109/IEEESTD.2021.9363693.

[2] Lina Ahmad, Rania Al-Sabha, and Ali Al-Haj. Design and implementation of a secure qr payment system based on visual cryptography. In *2021 7th International Conference on Information Management (ICIM)*, pages 40–44, 2021. DOI: 10.1109/ICIM52229.2021.9417129.

[3] International Electrotechnical Commission. Electromagnetic compatibility (emc) – part 4-20: Testing and measurement techniques – emission and immunity testing in transverse electromagnetic (tem) waveguides. *IEC 61000-4-20:2010*, (STD-570430): 1–151, 2010.

[4] ISO Central Secretary. Cards and security devices for personal identification - contactless proximity objects - Part 2: Radio frequency power and signal interface. Standard ISO/IEC 14443-2:2020, International Organization for Standardization, 7 2020. URL https://www.iso.org/standard/73597.html.

[5] Kenneth Rohde Christiansen, Zoltan Kis, François Beaufort, and Alexander Shalamov. Web nfc – draft community group report. Draft community group report, W3C, 12 2021. URL https://w3c.github.io/web-nfc/.

[6] Solomon Systech Limited. Ssd1331 96rgb x 64 dot matrix oled/pled segment/common driver with controller. https://cdn-shop.adafruit.com/datasheets/SSD1331_1.2.pdf, 11 2007. Rev 1.2.

[7] D. M'Raihi, S. Machani, M. Pei, and J. Rydell. Totp: Time-based one-time password algorithm. RFC 6238, RFC Editor, May 2011. URL http://www.rfc-editor.org/rfc/rfc6238.txt. http://www.rfc-editor.org/rfc/rfc6238.txt.

[8] BME Department of Electric Power Engineering High Voltage Laboratory. Mérési útmutató: Emc observation of lighting sources with gtem cell. Technical report, BME. URL http://nfl.vet.bme.hu.

[9] ISO Central Secretary. Information technology – automatic identification and data capture techniques – QR Code bar code symbology specification. Standard ISO/IEC 18004:2015, International Organization for Standardization, 2 2015. URL https://www.iso.org/standard/62021.html.

[10] NXP Semiconductors. Mfrc522: Standard performance mifare and ntag frontend. https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf, 4 2016. Rev. 3.9.

[11] Nahar Sunny Suresh Shobha, Kajarekar Sunit Pravin Aruna, Manjrekar Devesh Parag Bhagyashree, and Kotian Siddhanth Jagdish Sarita. Nfc and nfc payments: A review. In *2016 International Conference on ICT in Business Industry Government (ICTBIG)*, pages 1–7, 2016. DOI: 10.1109/ICTBIG.2016.7892683.

[12] Espressif Systems. Esp32-wroom-32 datasheet. https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf, 2021. specifications for the ESP32-WROOM-32 module.

Appendix

A.1 Abbreviations

2FA	Two-factor authentication
CNC	Computerized Numerical Control
CSV	Comma Separated Values
EMC	Electromagnetic Compatibility
ER (diagram)	Entity Relationship Diagram
GPIO	General Purpose Input/Output
HTML	HyperText Markup Language
IDE	Integrated Development Environment
ISM	Industrial, Scientific and Medical
JDBC	Java Database Connectivity
JDK	Java Developement Kit
NFC	Near-Field Communication
OLED	Organic Light-Emitting Diode
PWM	Pulse-Width Modulation
RF	Radio Frequency
RFID	Radio Frequency Identification
RSA	Rivest–Shamir–Adleman
SPI	Serial Peripheral Interface
SSL	Secure Sockets Layer
SSO	Single sign-on
XML	Extensible Markup Language