**PATUAKHALI SCIENCE AND TECHNOLOGY UNIVERSITY**

**COURSE CODE EEE-212**

**Electrical Technology Sessional**

**Project Report**

**SUBMITTED TO:**

### **Md. Naimur Rahman Professor Department of Electrical and Electronics Engineering Faculty of Computer Science and Engineering**

**SUBMITTED BY:**

**Md. Sadman Kabir Bhuiyan**ID: **2102020**,   
Registration No: **10147**Faculty of Computer Science and Engineering

**Md. Sharafat Karim**ID: **2102024**,   
Registration No: **10151**Faculty of Computer Science and Engineering

Date of submission: **29 October 2024**Project title: **Metro Recharge Point**

**Metro Recharge Point**

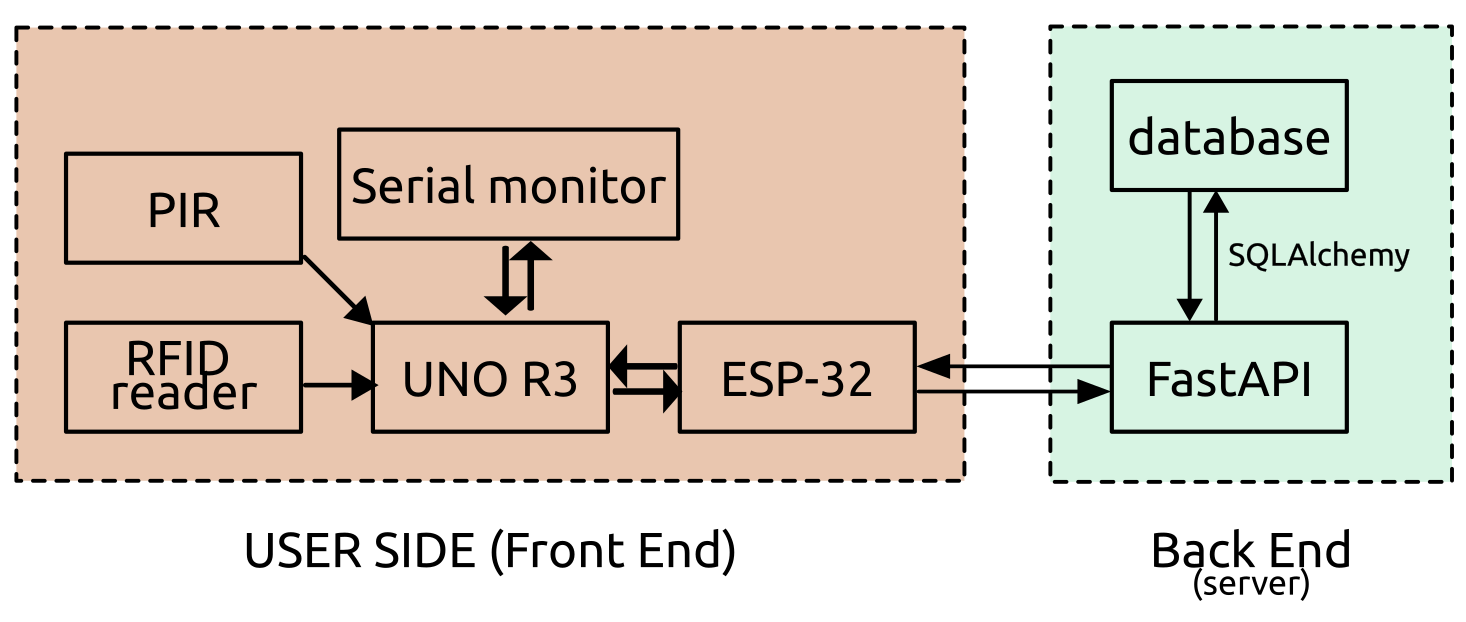
## ****Description****

Metro rail card's self recharging utility with mobile banking ability. Where users can scan their card and access their accounts with simple clicks.

## Brief History

In general in a metro rail system, whenever a user wants to travel, he has to go to the ticket counter and form a query, in order to collect the ticket, which makes the whole entire process more time-consuming. To solve this issue, we can implement a backend server for database and API, and use our system as frontend.

## Objectives

1. Developing an Arduino-based system with RFID readers for identifying metro cards.
2. Integrate a secure mobile banking platform for users to recharge their metro cards on the go. Where user can use mobile banking/ any specific hardware to recharge cards.
3. Implement a backend system to track user balances, recharge, and discharge transactions using **FastAPI** and **SQLAlchemy**.
4. Ensure seamless communication between hardware components (Arduino, RFID reader) and backend services via HTTP requests.
5. Auto check balance on card punch, and LED color to indicateb balance.

## Scope

* **Hardware Integration**: Utilize components such as RFID Readers, ESP32, PIR sensors, and buzzers for card detection and system feedback.
* **Backend Development**: Develop a FastAPI-based RESTful API to handle user balance management, including recharge, discharge, and user creation functionalities.
* **Mobile Banking Integration**: Enable secure mobile banking transactions for recharging metro cards.
* **Real-time Data Processing**: Ensure real-time updates to user balances via cloud-hosted services. It can also be deployed on a local server.

## Benefits

* **User Convenience**: Quick and easy metro card recharges from any location using mobile banking or specific machines. Further systems can be easily developed for using the same backend. For example, bus ticket system.
* **Real-time Updates**: Users can track balances and manage their metro cards through a user-friendly interface. Or, they can also use an interface hosted on web to quickly access. Besides, there can be admin panels and administrations control access, in order to debug things out.
* **Scalability**: Designed to handle large volumes of users and transactions efficiently. Besides, it can be integrated into other systems, as well as handling multiple purposes.
* **Data Analyze:** Backend servers can also implement anonymous data collection. Which will be a great help for future crowd controlling and preparing more efficient schedule for the entire train system.

## Backend Server

* **Python FastAPI**
* **SQLAlchemy**
* **SQLite/MySQL** (for database)
* **Python**
* **Uvicorn** (for ASGI server)
* **Pydantic** (for data validation)
* **HTTP/HTTPS**

## Frontend Budget

|  |  |
| --- | --- |
| **Equipments Name** | **Approx. Price** |
| PIR (Passive infrared sensor) | 110 |
| Arduino UNO R3 SMD | 574 |
| Cable | 55 |
| Breadboard | 150 |
| Buzzer | 15 |
| RFID Reader | 195 |
| RFID Cards | 70 |
| Resistor | 30 |
| ESP 32 | 470 |
| LED | 5 |
| Matrix keyboard | 75 |
| LCD display | 273 |

## ****1. Project Overview****

The **Metro Recharge System** is a microcontroller-based, smart recharge station for metro cards. It incorporates RFID card scanning, balance management, and interactive features to enhance user experience at recharge stations. The system provides easy card balance checking and recharge options via both on-site and online methods. It combines multiple technologies, including Arduino, ESP32, and a Python-based graphical display.

## ****2. Objectives****

* To enable balance checks and recharges for metro cards via RFID.
* To use an intuitive user interface for easy navigation and user engagement.
* To provide an online component that supports data storage and update via RESTful APIs.
* To utilize visual indicators (LEDs and screen displays) and sound signals to enhance user feedback.

## ****3. System Architecture****

1. **Arduino Metro Station Control**: Arduino manages the RFID card reading, balance display, and navigation of options for users.
2. **ESP32 Internet Connectivity**: Communicates with a remote server to retrieve or update card balances. Visual feedback is provided via LEDs.
3. **Metro Backend API**: A RESTful API with FastAPI and SQLAlchemy allows users to store, retrieve, and update balances remotely.
4. **Python GUI**: Displays balance information in real-time, using Tkinter for the full-screen GUI.

## ****4. Technologies Used****

### Hardware

* **Arduino**: Controls the RFID reader, keypad, LCD, LEDs, and other input-output devices.
* **ESP32**: Provides Wi-Fi connectivity for online balance update requests.
* **Ultrasonic Sensor**: Detects user proximity to the station.
* **LDR (Light Dependent Resistor)**: Adjusts screen brightness based on ambient light.
* **LEDs and Buzzer**: Visual and auditory feedback indicators.

### Software

* **FastAPI**: Backend API framework for managing user balances.
* **SQLAlchemy**: ORM for database interaction in the backend API.
* **Tkinter**: Python library for GUI display on full-screen devices.
* **Python**: For serial communication and display interaction with the Arduino.

## ****5. Code Implementation****

Code is

## ****6. Backend API****

The **Metro-API** backend is built with FastAPI and SQLAlchemy to manage user balances. It includes endpoints for creating users, fetching balances, and updating balances upon recharge or discharge.

### Endpoints

* GET /users/{id} - Retrieves a user’s balance.
* POST /users - Creates a new user.
* PUT /users/+/{id} - Recharges a user’s balance.
* PUT /users/-/{id} - Deducts from a user’s balance.
* DELETE /users/{id} - Deletes a user account.

## ****7. Testing and Troubleshooting****

### Common Issues and Solutions:

* **Serial Communication Errors**: Ensure baud rates match across Arduino and Python scripts.
* **Wi-Fi Connection Issues**: Double-check SSID and password for ESP32 connection.
* **Display Delays**: If data doesn’t show immediately on the GUI, check the Arduino-Python serial connection.

## ****8. Future Improvements****

* Implement additional security measures, such as user authentication.
* Expand the system to handle multiple RFID cards concurrently.
* Provide more payment options, such as mobile wallet integration.

## ****9. Conclusion****

This project presents a comprehensive solution for metro card recharge stations, using a blend of hardware components and software technologies. It effectively demonstrates how microcontrollers, Wi-Fi modules, and server communication can be used to develop an efficient, user-friendly recharge system. The balance-checking and recharge features allow users to have an easy and seamless experience at metro recharge points.