CASA0019: Sensor Data Visualisation 24/25: SubRadar

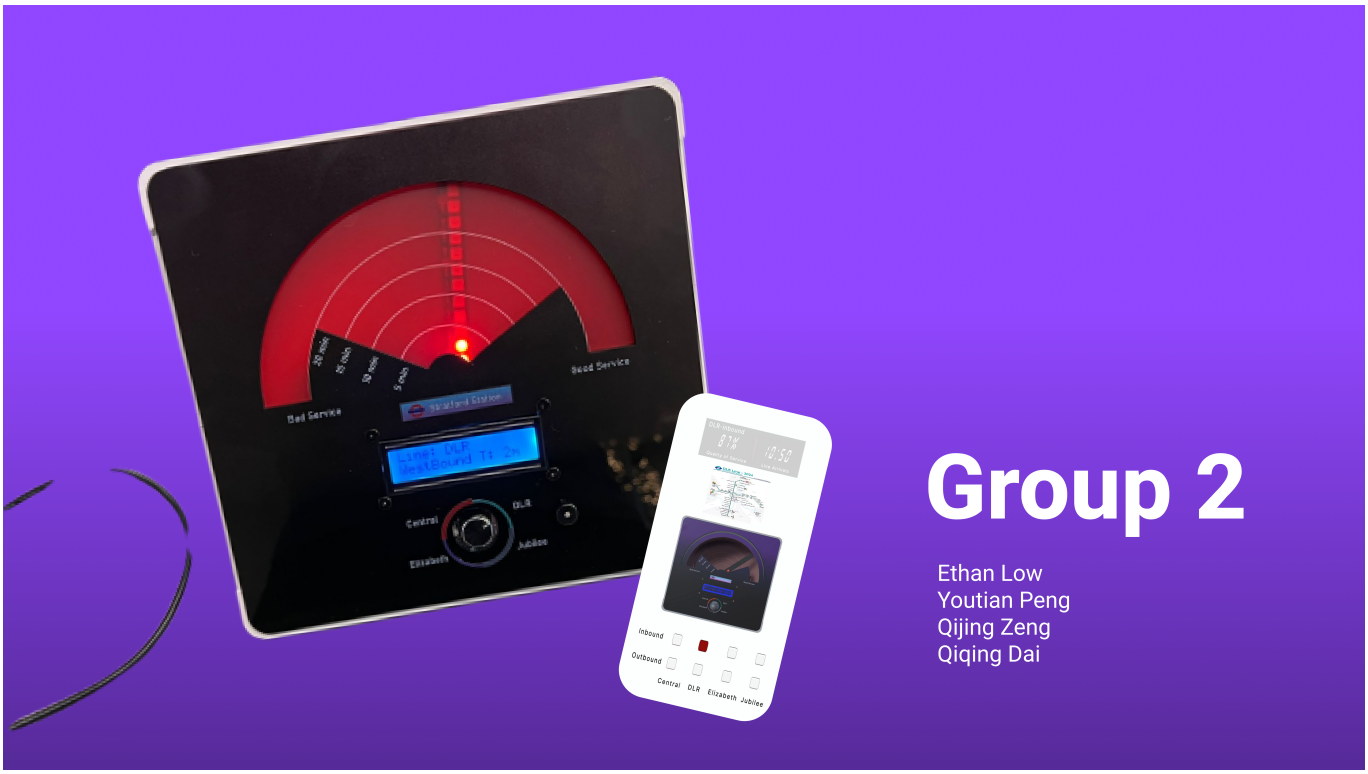


Table of Contents

Project Overview

SubRadar is an interactive device that visualizes real-time transportation data for Stratford Underground Station.

Rationale

SubRadar aims to visualise underground train arrival times and service information into a physical dial device, with digital elements, for students and staff preparing to leave UCL East after class or work. It was hoped that through a unique radar-inspired dial, this information could be presented to the UCL East audience in an accessible and engaging way.

UI Design Principles and Achievements

Design principles - Principles of design

The basic goals of our user interface design we promote are clearness and ease of understanding. In order to place essential information right under the user's fingertips, we followeds a layered information display structure: The outer semicircular cone pictures the quality at subways along with the movement of the pointer that clearly reflects the exact status of the trains in the real-time. The outer circle of the track uses lights that will light up the numbers of the minutes that go by, thus making it easy for the passengers to know when the next train will arrive. The over-arching design is based upon the wholesale reduction of cognitive load for the user by making information simple and straightforward. Furthermore, the real-world feedback loop provided by the actual physical pointer magnifies the collectedness of the interaction experience.

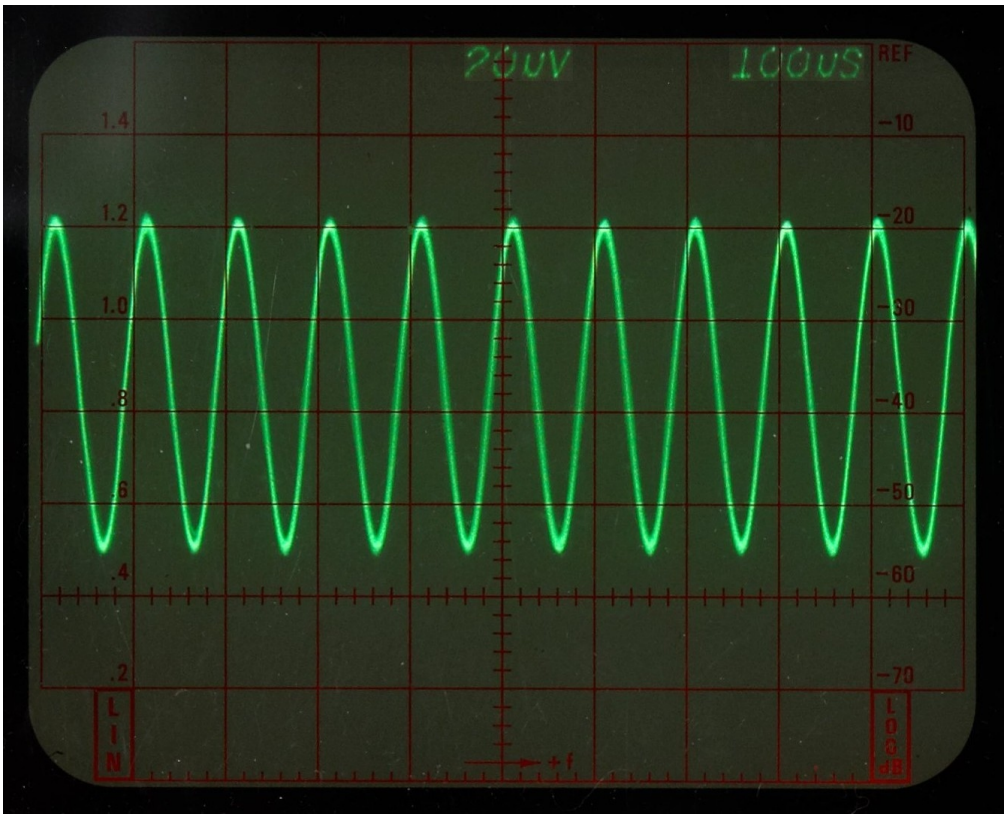
Achievements

The UI interface successfully achieves clear information segmentation through a dual-layer dial design, enabling users to quickly access both service quality and arrival time information. The light-marking feature effectively visualizes time data and received positive feedback from instructors during the demonstration. The UI interface is fully integrated with the hardware functionality, supporting users in selecting subway line directions and specific line information through the button and sliding potentiometer. This design successfully combines service quality and time data, meeting users' needs for quick and efficient commuting decisions.

Enclosure and Industrial Design

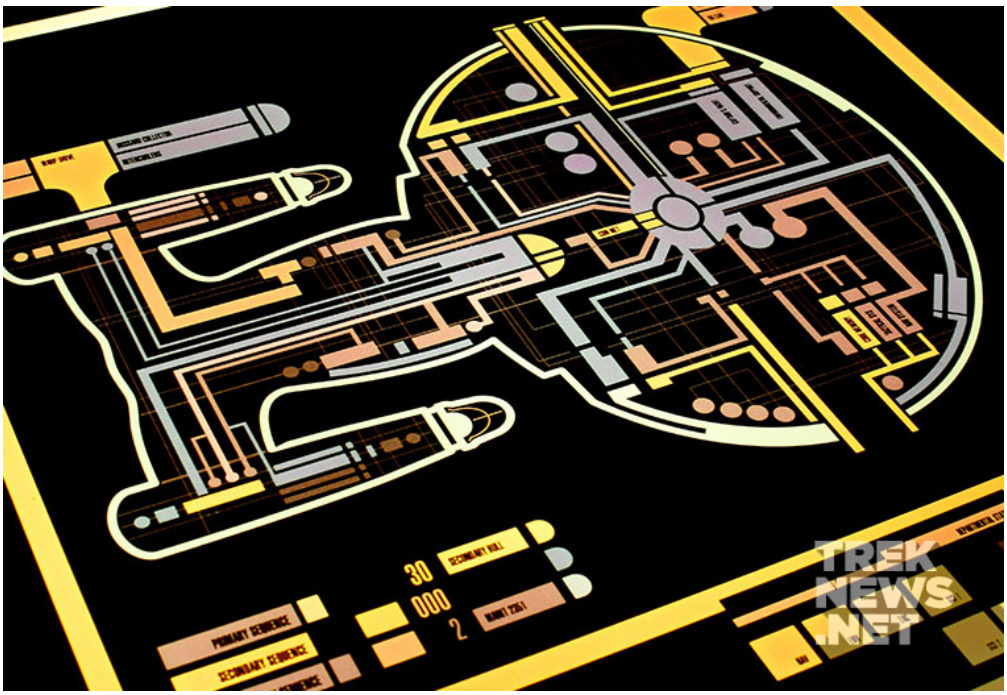
Inspiration

SubRadar emulates a vaccum-tube type device in order to make reference to the classic green or red circular radar scopes of the past. The homage to historical radar is intentional; we wanted to reference a typology of device designed to show the proximity of moving vehicles to a fixed station.



An oscilloscope, a type of device that traditionally used a vaccum-tube display (Wikipedia, 2025)

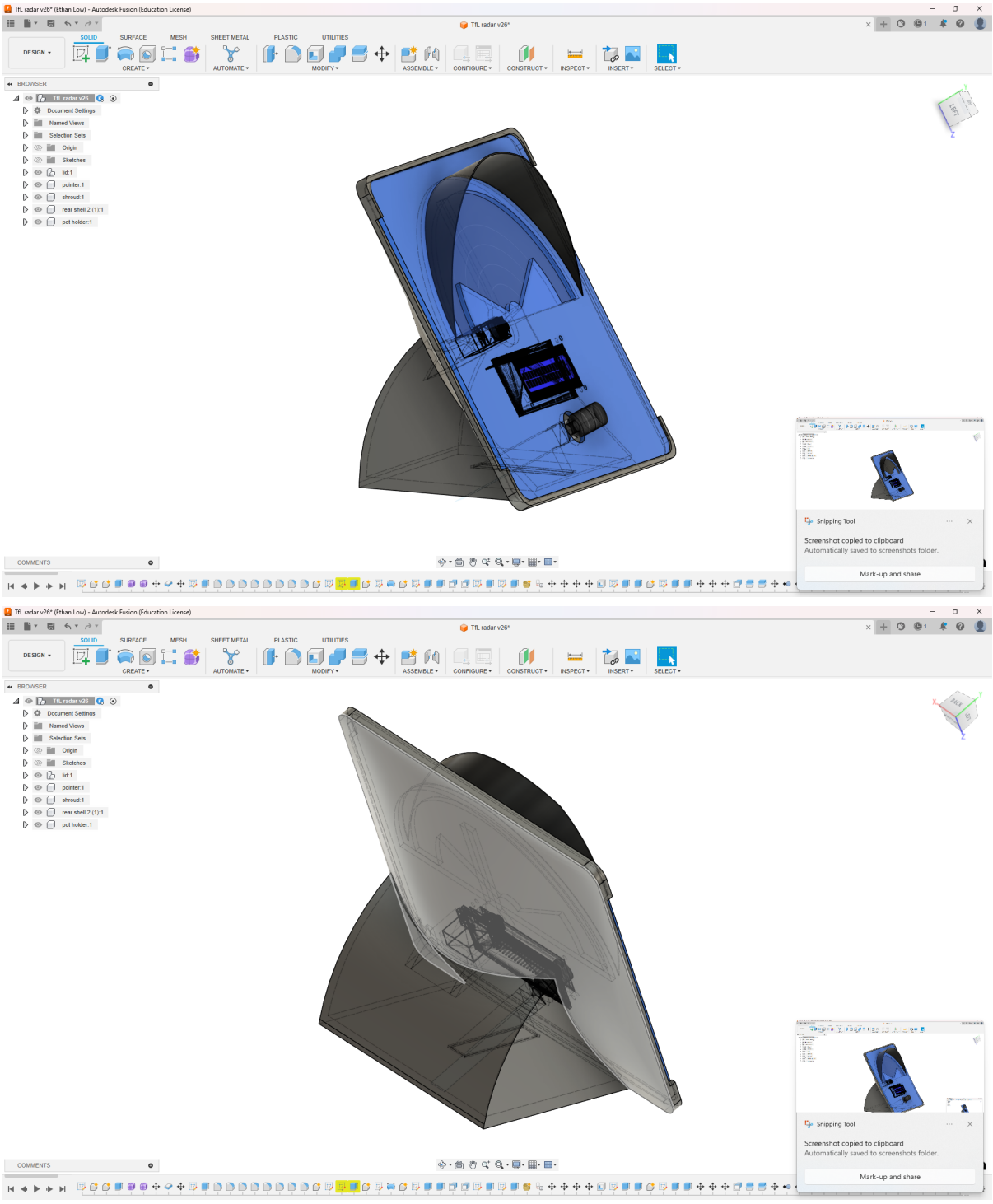
However, without an actual vaccuum-tube display available to us, we had to simulate the effect using stage special effect techniques. Taking inspiration from Michael Okuda, the graphic designer for Star Trek: The Next Generation, (Savage, 2024) we decided to place LEDs mounted on a servo pointer behind frosted acrylic to evoke the grainy effect of a vaccum-tube display.



Michael Okuda's backlit graphic designs for Star Trek, affectionately known by fans as 'Okudagrams' (Nguyen, 2015)

Main enclosure

The physical design was developed in Fusion360 and printed on a Bambu Lab 3D printer. To make the design look more appealing, inspiration was taken from Jony Ive's earlier designs for Apple, (Ravenscroft, 2019) using Fusion360's NURBS surface manipulation tool to create a curved back shell that creates an optical illusion, making the device look much thinner than it is in reality.



Screenshots from Fusion360 showing the curved design of the back shell

User interface and experience

Wireframes

Front panel with final UI

Test fitting the front face, made of acrylic, to the 3D printed rear enclosure shell

The major device components coming together, showing the difference made by adding the UV printed graphics

The final UI was adapted from the wireframe designs to fit the dimensions of the pysical enclosure as built. These were then prepared for UV printing using GIMP and printed on a laser-cut piece of black acrylic. The graphics were printed with 2 layers of white laid down before the final colour print, in an attempt to ensure the print had sufficient opacity.

Data

Data Source

The project utilizes the Transport for London (TfL) Unified API to access real-time data on public transportation. This API provides comprehensive information across various transport modes, including live arrivals, line statuses, and station details. By integrating this API, the project ensures that accurate and up-to-date data is sent to the device for users to consume.

Used Data:

Field Name	Meaning	Purpose
`lineName`	Line name	Used to distinguish data for different lines
`expectedArrival`	Scheduled arrival time	Used to calculate service interval variability
`timeToStation`	Remaining time to arrival	Used to calculate passenger waiting time
`direction`	Train direction	Used to distinguish service levels for different directions

In order to measure the Overall Service Level, the group proposed an integrated index - **Overall Service Level**, which offers a comprehensive evaluation of transport service quality by combining key metrics such as `Service Interval Variability` and `Passenger Waiting Time`. It provides an integrated measure of overall service stability and efficiency, with higher scores reflecting better service quality.

Metric	Method	Required Fields	Explanation
`Service Interval Variability`	Calculate the time intervals between consecutive trains using `expectedArrival` and compute the standard deviation.	`expectedArrival`, `lineName`	A smaller standard deviation indicates more stable and reliable services, while larger values suggest irregular and less dependable operations.
`Passenger Waiting Time`	Analyze the average timeToStation (remaining time for train arrival).	`timeToStation`, `lineName`	Shorter waiting times indicate higher service efficiency, while longer times suggest inadequate service, impacting passenger experience.
`Overall Service Level`	0.5 * **Service Interval Variability** + 0.5* **Passenger Waiting Time**	`Service Interval Variability`, `Passenger Waiting Time`	A composite metric assessing overall service stability and efficiency. Higher scores indicate better service quality.

Output Data:

Field Name	Meaning	Purpose
`timeToStation`	Time for the nearest tube to the station	Notify user
`serviceLevel`	Line's overall service level(50%*Passenger Waiting Time + 50%*Service Interval Variability)	Demonstrate the line's service level

Data Processing

The preprocessing of raw data involves three key steps: **data cleaning**, **filtering**, and **transformation**. These steps ensure that the data is clean, consistent, and ready for further analysis.

1. Data Cleaning

Data cleaning focuses on handling missing or incomplete data. For instance, the script `fetch_and_process.py` checks whether the API returns valid data. If no data is received for a specific line or direction, the system prints a warning message and skips further processing for that update:

```
if not api_data:
    print(f"No data received for line {line_name}, direction {direction}. Skipping update.")
    continue
```

2. Data Filtering

To ensure relevance and avoid redundancy, the data filtering process includes:

- **Removing Duplicate Train Information**: Only unique train data is retained to prevent duplication.
- **Direction-Based Filtering**: The data is further refined by selecting train information specific to a given direction. For example, in `fetch_and_process.py`, unique train data is filtered based on the specified direction. If no valid data is found, the system skips further processing:

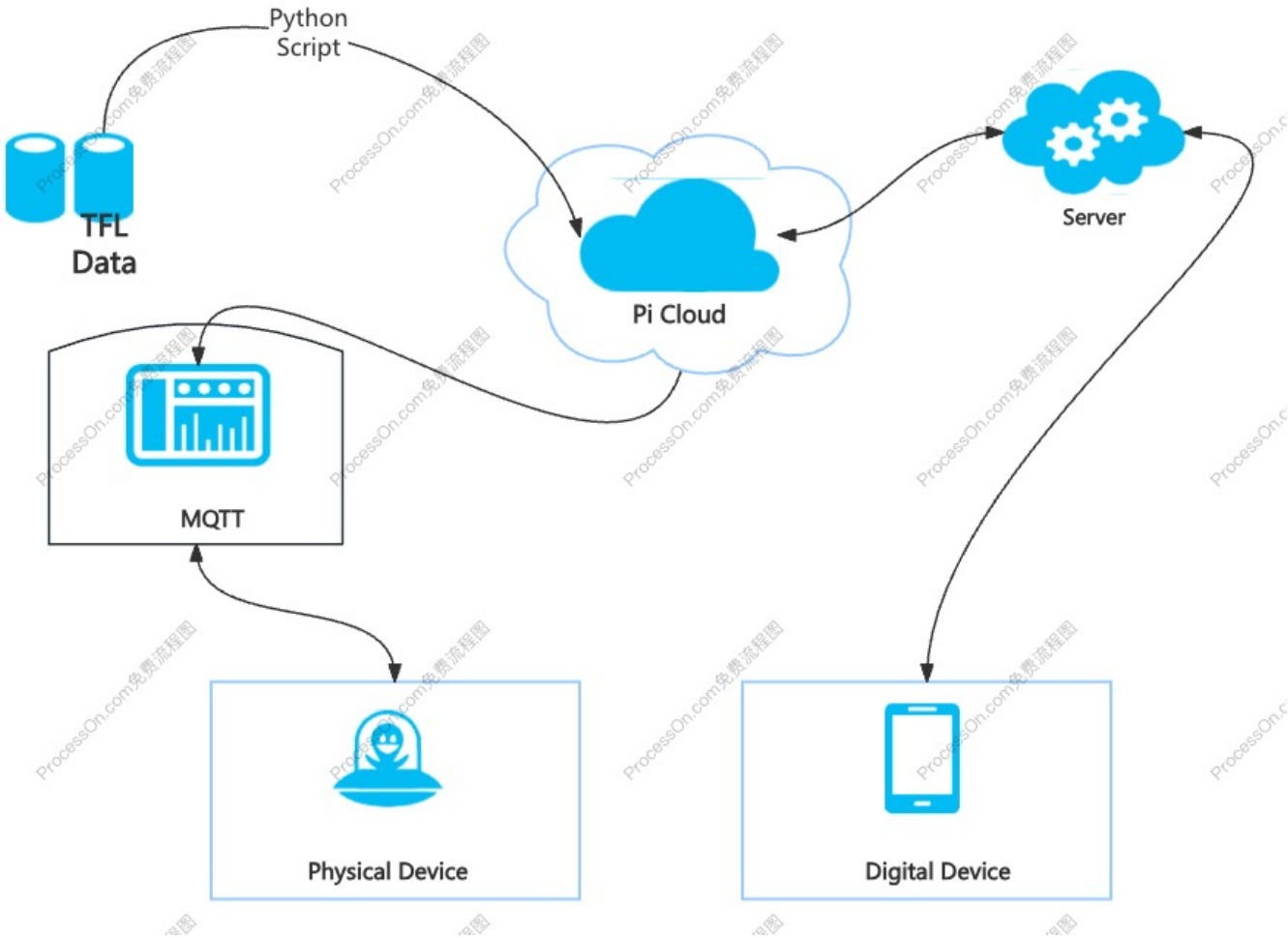
  ```
  unique_trains = filter_unique_trains(api_data, direction)

  if not unique_trains:
      print(f"No valid unique trains data for line {line_name}, direction {direction}. Skipping update.")
      continue
  ```

3. Data Transformation

The transformation process ensures that the data is standardized and ready for use by converting raw API responses into a consistent JSON format. Train arrival times are reformatted to follow a uniform time structure, making the data easier to process and analyze. Additionally, directions such as "inbound" and

"outbound" are mapped to standardized terms like "westbound" and "eastbound" to enhance clarity and consistency across the system.

Data Management System



The data management system depicted in the diagram ensures efficient handling and flow of information across the project. Data from TfL (Transport for London) is retrieved using Python scripts and processed in the Pi Cloud, where it is further distributed. The server, connected to the cloud, facilitates advanced computations and real-time updates.

MQTT is used to transmit processed data from the Pi Cloud to the physical device, enabling it to display real-time information such as train schedules and service quality. Simultaneously, the digital device receives updates directly from the server, ensuring synchronized data representation across both devices. This system leverages a cloud-based architecture to manage data acquisition, processing, and distribution seamlessly.

Data Device

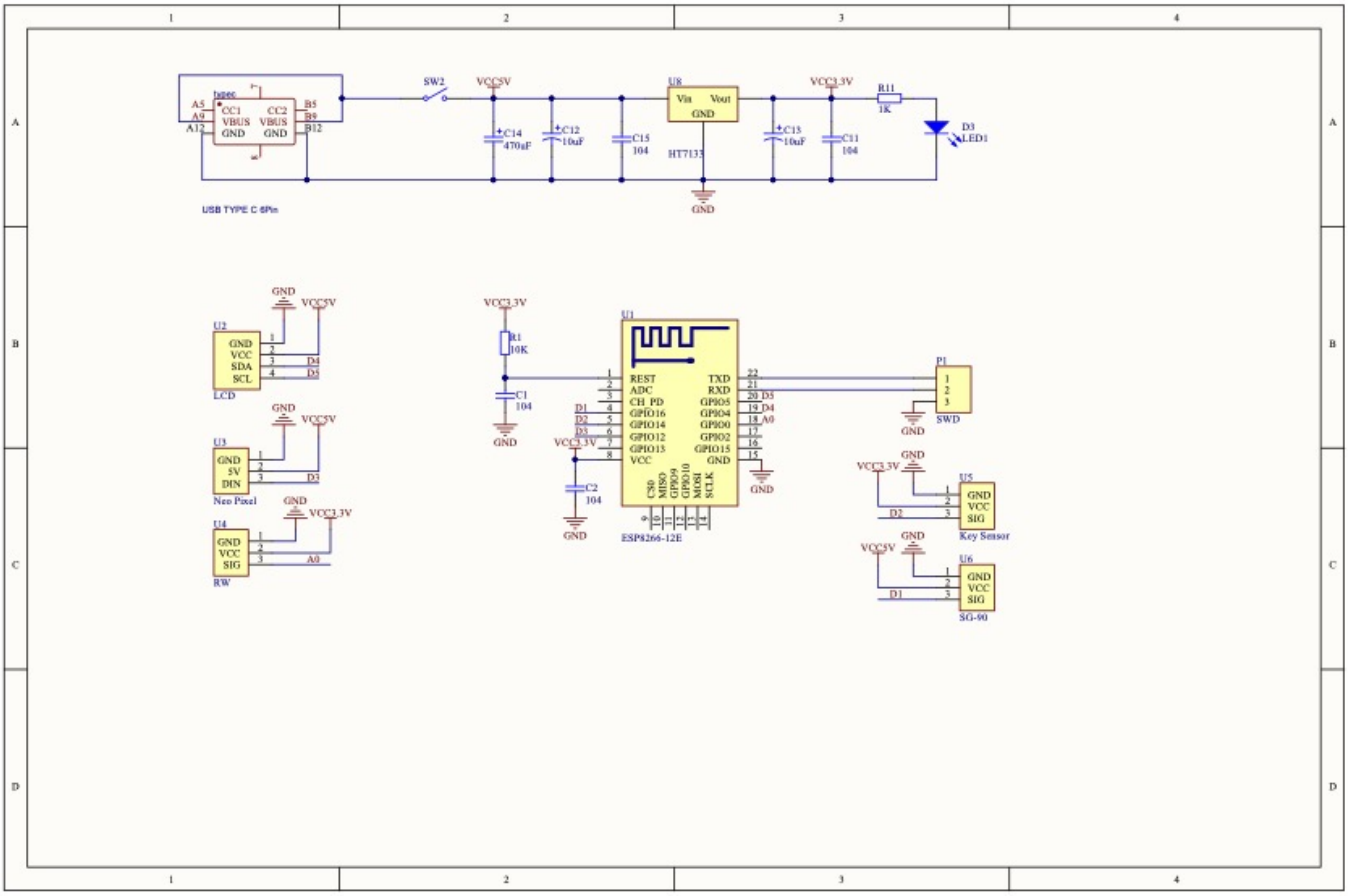
DATA DEVICE (Physical)

Board & Sensor

Specification	ESP8266	Rotary Encoder	Servo	NeoPixel	16x2 LCD

Specification	ESP8266	Rotary Encoder	Servo	NeoPixel	16x2 LCD
Microcontroller	Tensilica L106 32-bit RISC processor, clocked at 80 MHz (can be overclocked to 160 MHz)	-	-	-	-
Operating Voltage	3.3V	-	4.8V to 6V	5V	5V
Digital I/O Pins	16	-	-	**Data Pin**: GPIO0 (D2) **Brightness**: 30 (0-255) **Number of LEDs**: 16	**SDA Pin**: GPIO2 (D4) **SCL Pin**: GPIO14 (D5)
Analog Input Pins	1 (10-bit ADC)	-	-	-	-
Library Used	-	-	Servo	Adafruit NeoPixel	LiquidCrystal_I2C
Documentation	ESP8266 Documentation	Rotary Encoder Link	Servo Motor Documentation	NeoPixel Documentation	16x2 LCD Documentation

Schematics



Firmware Design

Firmware Modules and Logic

- Initialization Module

  Upon powering up, the firmware's Initialization Module configures the LCD, LEDs, servos, and network connections, including WiFi and MQTT setup for data communication. It also schedules periodic tasks like button monitoring and rotary encoder detection to enable user interaction.

- Input Processing Module

  The Input Processing Module interprets rotary encoder signals and button presses to execute actions like menu navigation or function selection, ensuring prompt and accurate device response.

- Display Control Module

  The Display Control Module manages LED indicators, the LCD screen, and servos to provide real-time updates, display line and direction information, and indicate service levels through visual outputs.

Development Tools and Frameworks

The firmware is developed in C++ using the *Arduino Integrated Development Environment (IDE)*, a robust platform with extensive libraries and community support. It utilizes Arduino core libraries for hardware abstraction and peripheral management, simplifying interactions with sensors and actuators. For MQTT

communication, the `PubSubClient` library is employed to facilitate efficient message handling and seamless data exchange with the digital twin system.

Digital Dashboard

The dashboard in Android application is shown in figures below, it includes four main components:

- Overview Panel

  This panel includes the train name, line direction, real-time service value, and expected arrival time. The function is to present important data in a direct and prominent way, allowing users to capture key information as soon as they open the app.

- Map of Each Train Line

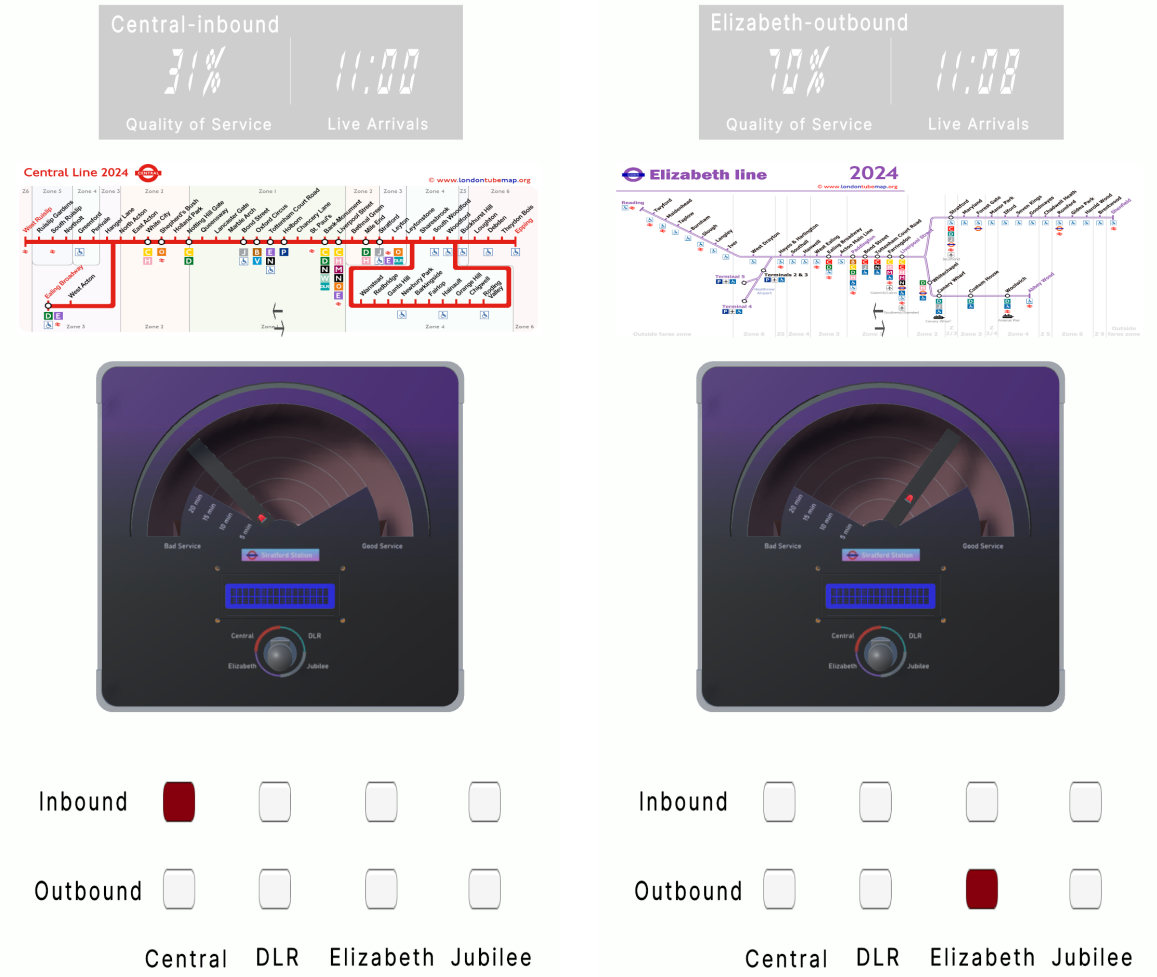
  This map indicates all of the stations in the current train line, it can help users to navigate towards their destinations.

- Digital Twin of Physical Gauge

  The interface of this gauge is same with physical gauge, the pointer can smoothly transitions to the corresponding angle based on the “quality of service” value, the specific animation effect achieved through Mathf.Lerp. In addition, there are five lights on the pointer, and they will show a "light up" effect based on the train's expected arrival time. This effect is achieved by ShowObjectBasedOnRange method, which makes objects appear or disappear according to time value.

- Line Selection Buttons

  Each button means a specific direction of a train line. For instance, the following figures display the information for the Central line with inbound direction and the Elizabeth line with outbond direction. Users can interact by clicking the buttons, and when a button is pressed, the three components above will switch to display the related information. The eight buttons are arranged in a table-like layout, making it convenient for users to understand the difference of each button.



Challenges and Future Development

Enclosure and product design

The enclosure provided significant challenges to print, and several attempts resulted in delamination. In the end, the final print was successful in no small part due to the precision of the Bambu printer as opposed to the Prusa Mk 4. In future developments, it would be good to test breaking down or reorienting the rear shell 3D model to see if it would be easier to print in such as configuration.

For the UV printing of the front face with the UI, print opcacity was a persistent problem due to the dark coloured acrylic. If we were to develop another iteration of the device, lighter coloured acrylic could be used, and black paint applied to the interior face to reduce light leakage.

Digital dashboard

Although a 3D model was constructed using Unity software during the setup of the digital dashboard, in order to ensure that users can intuitively view specific train line information, this project ultimately adopted a 2D presentation effect. Future research could focus on how to present components in a 3D effect while ensuring the practicality of the functions, to enhance the user's interactive experience.

Data Challenges and Solutions

Managing data synchronization with the TFL API was challenging due to connectivity interruptions, causing delays in real-time updates. This was resolved by adding a retry mechanism in Python scripts. Inconsistent and incomplete data required cleaning and standardization to ensure usability. Future improvements will focus on implementing caching and optimizing data handling for better reliability and performance.

Physical Device Challenges and Solutions

Hardware reliability was an issue, with a faulty rotary encoder and unstable MKR 1010 board. Replacing these with a new encoder and ESP8266 resolved the problems. MQTT connectivity issues were addressed by using a non-blocking code structure. Future plans include refining the circuit design and exploring alternative communication protocols for improved scalability.

HOW TO USE

Installation

Download unity project and export into android phone(note: the whole unity project package can be download from: Unity download)

Demo

https://github.com/user-attachments/assets/52968817-6eab-4319-b941-27107830a150

Video demo of the physical device

https://github.com/user-attachments/assets/e4ada6dc-514a-41d3-86cf-59e24c7f435a

Video demo of the digital twin Unity mobile application

References

Nguyen, W. (2015). *An Ode to the Okudagram.* [online] TREKNEWS.NET. Available at: https://treknews.net/2015/07/23/ode-to-okudagram/ [Accessed 11 Jan. 2025].

Ravenscroft, T. (2019). *Jony Ive's 10 most revolutionary designs for Apple*. [online] Dezeen. Available at: https://www.dezeen.com/2019/06/28/jony-ive-designs-apple-imac-ipod-iphone/ [Accessed 11 Jan. 2025].

Savage, A. (2024). *Why Star Trek's Graphic Design Makes Sense.* [online] YouTube. Available at: https://youtu.be/D24tYFIVyv0?si=6iS26g7v1mN3KYin [Accessed 11 Jan. 2025].

Wikipedia (2025). *Oscilloscope.* [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Oscilloscope [Accessed 11 Jan. 2025].