Developing Data Dashboard Device For Queen Elizabeth Olympic Park

Dongyoung Lim Jingqi Cheng Patrick Whyte Yining An

https://github.com/CE-GPP/miniature-park

CASA0021

Connected Environments Group Prototype and Pitch 22/23

2023

Introduction

This report presents a project that aims to develop a real-time data dashboard device as an Objet d'art for the Queen Elizabeth Olympic Park (QEOP). The device is intended to provide users with a comprehensive understanding of the park before visiting, enabling them to plan their journey effectively while also serving as an aesthetically pleasing piece of art.

1.1. Background & Problem Statement

Data dashboards are typically designed by or for experts, focusing on functionality over aesthetics. This can be limiting for general users who seek accessible and appealing ways to access real-time information. Moreover, there is a lack of devices that combine practicality with sophisticated design. This project addresses these gaps by creating a data dashboard that is visually appealing and user-friendly, targeting general users rather than experts.

1.2. Objectives

The primary objectives of this project are:

1. To develop a real-time data dashboard device for QEOP that combines aesthetics and functionality.
2. To provide users with an intuitive experience, allowing them to correlate data with geographical context.
3. To ensure a high-quality, visually appealing design that complements the surrounding environment.
4. To create a device that can be easily updated with new data, offering a dynamic user interface.
5. To complete the project within a short timeline, focusing on rapid prototyping and efficient development methods.

Literature Review

2.1. Data Dashboard

Emerging sensing and computing technologies generate geo-located data streams, which can be visualized for real-time urban systems understanding (Hudson-Smith, 2014). However, challenges such as low data quality, insufficient resources, and maintenance issues must be overcome. Dashboards should help citizens comprehend situations quickly, minimizing search processes and information overload (Matheus et al., 2020).

2.2. Related Works

Research has been conducted to develop urban data dashboards, consolidating real-time data feeds from various city domains (Batty, 2015). McArdle and Kitchin (2016) developed the Dublin Dashboard (Figure 1), which serves as a guideline for creating urban dashboards. The Building City Dashboards project introduced "Data City," a 3D-printed city model enhanced with projected graphics using projection mapping (Figure 2). The London Data Dashboard by CASA collects data from various APIs, presenting real-time information visually.

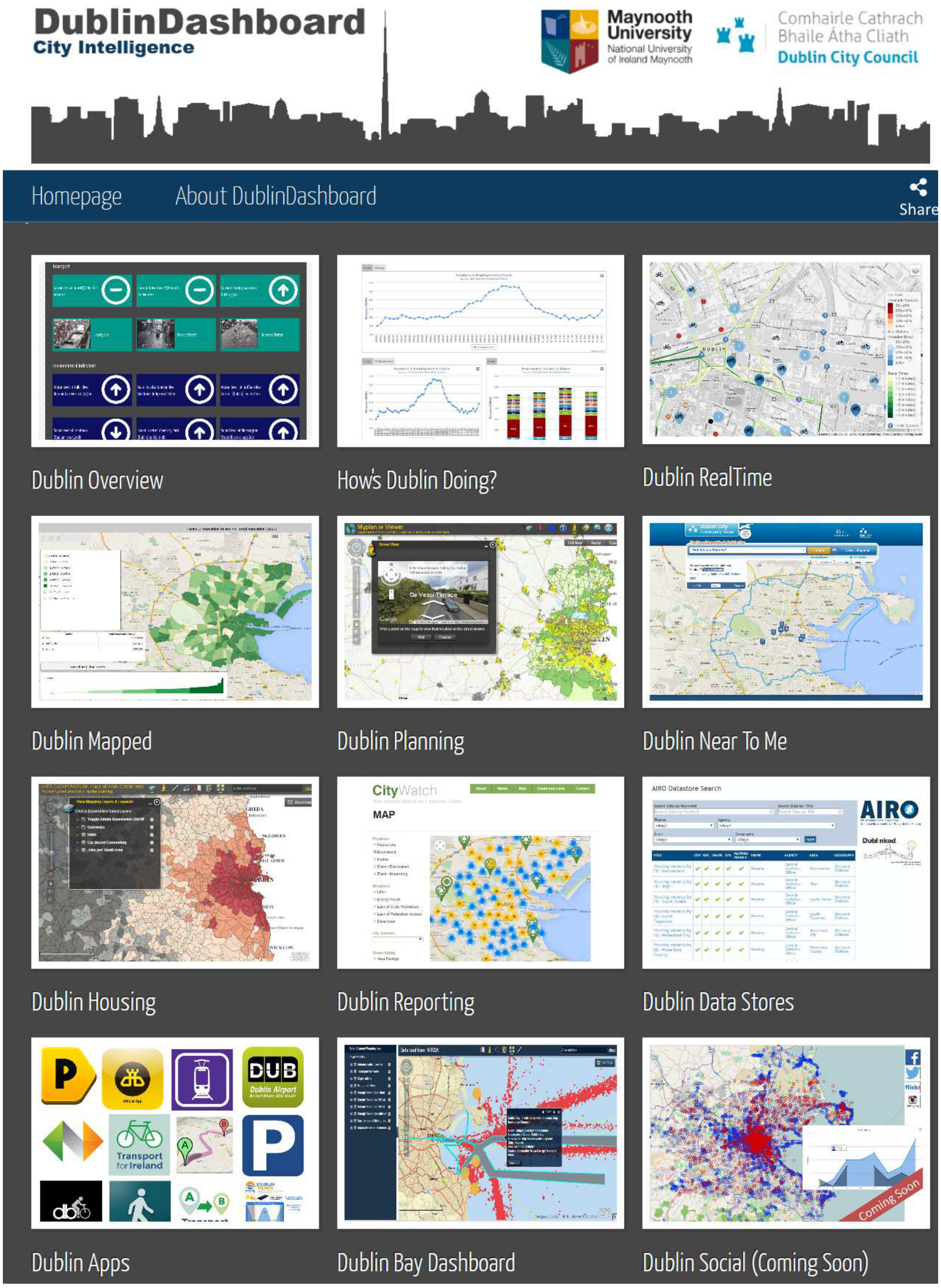

Figure 1. Dublin Dashboard Main Page

Figure 2. Projection Mapping with 3d printed model in Data City Project

Another related project is the London Data Dashboard (Figure 3) by the Centre for Advanced Spatial Analysis (CASA). This dashboard collects data from various application programming interfaces (APIs), making real-time information available and presenting it in useful ways (Batty, 2015). The dashboard enables users to visually compare and combine different aspects of the city, such as weather, transit system operations, and trending topics on Twitter.

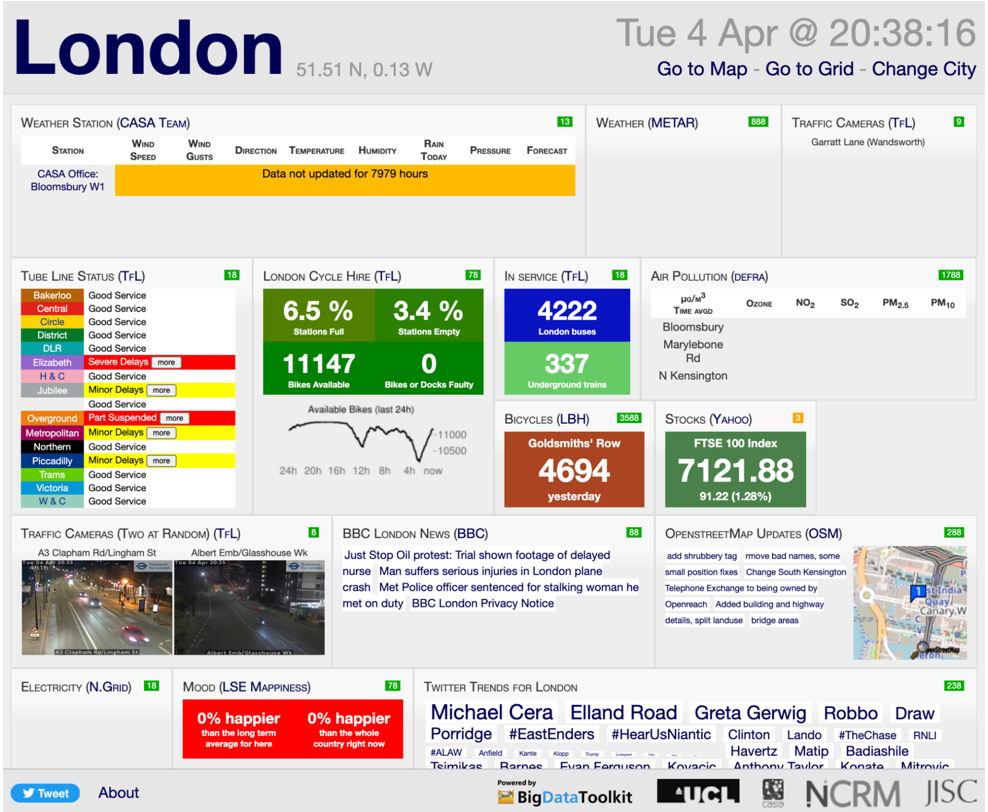

Figure 3. CityDashBoard Main Page

Existing researches offer real-time urban data visualisation through web applications or larger-sized devices. This project aims to develop a small-sized urban data dashboard, targeting potential users visiting QEOP for leisure or business purposes. This approach addresses a gap in existing research by offering a convenient, portable solution for real-time urban data access.

Design Process

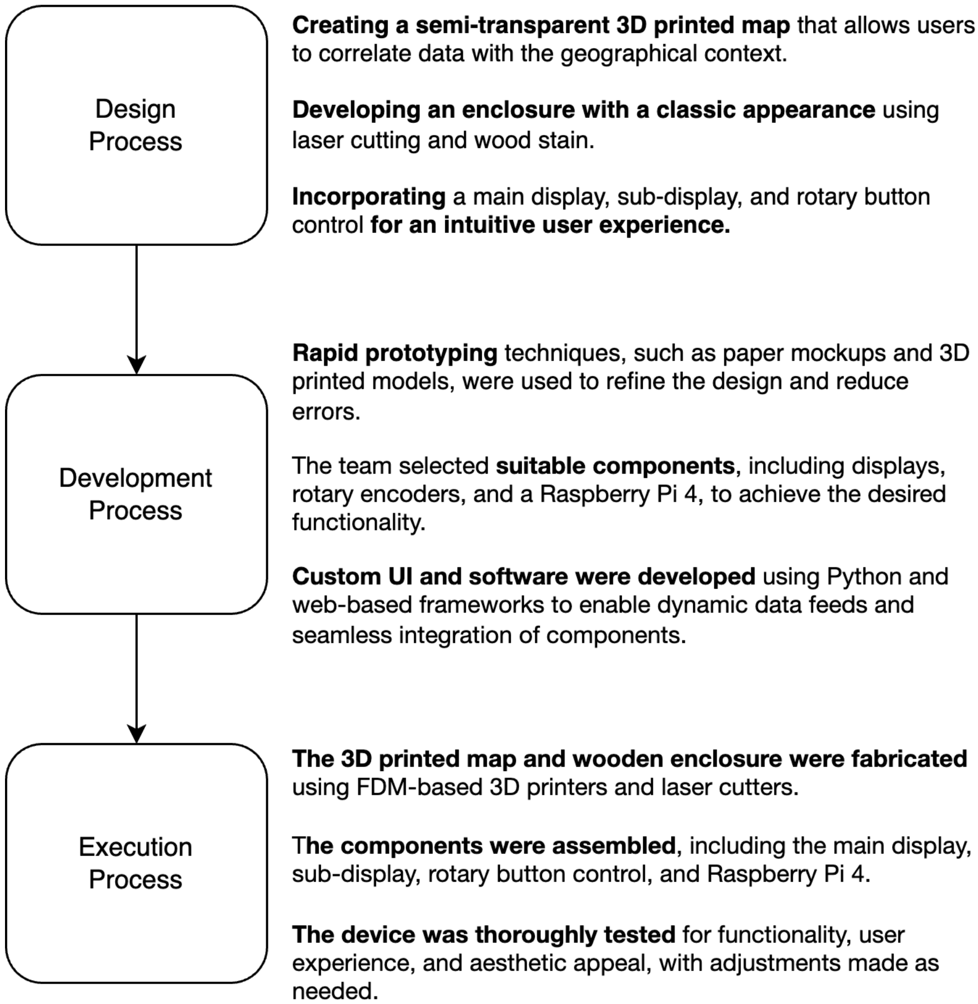

Figure 4. Design process.

In the Design Process (Figure 4), we present a comprehensive flow chart illustrating the various stages of our project's development. This figure effectively encapsulates the decision-making, design iterations (Figure 5 and Figure 6), and implementation strategies employed throughout the project, providing a clear visual overview of our methodology and progress.

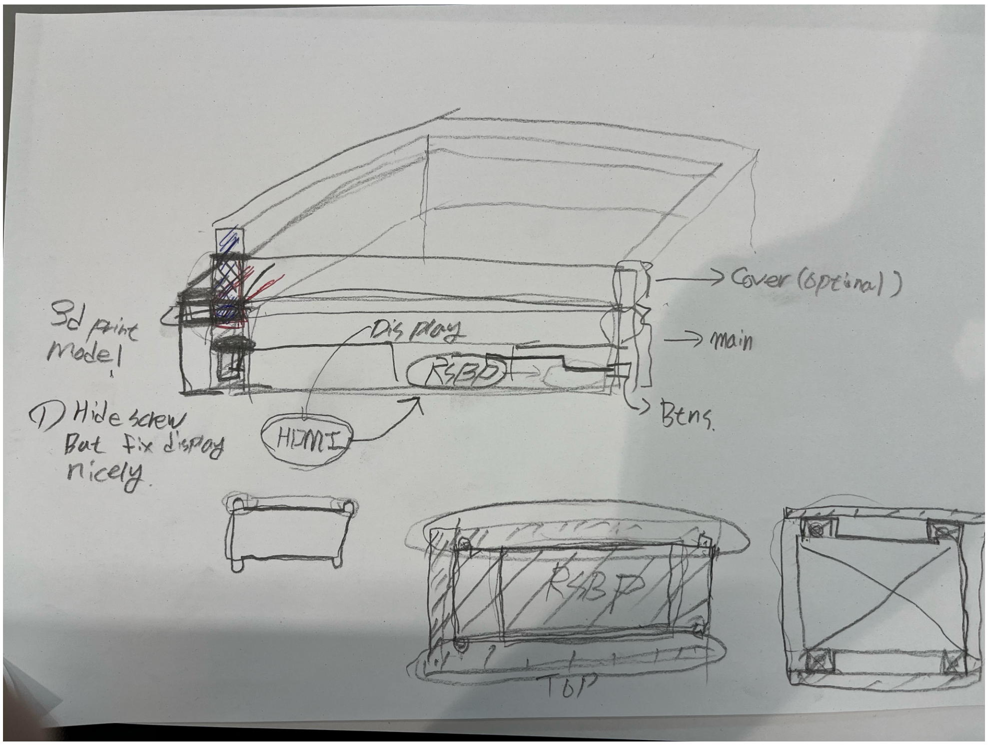

Figure 5. Initial design sketch.

Figure 6. Prototyping process from using paper to plywood.

Device Components

4.1 Raspberry Pi

The Raspberry Pi 4, with its high computing power, compact size, and low power consumption, was chosen as the primary computing unit for this device. It runs on the Linux operating system, supporting a wide range of programming languages.

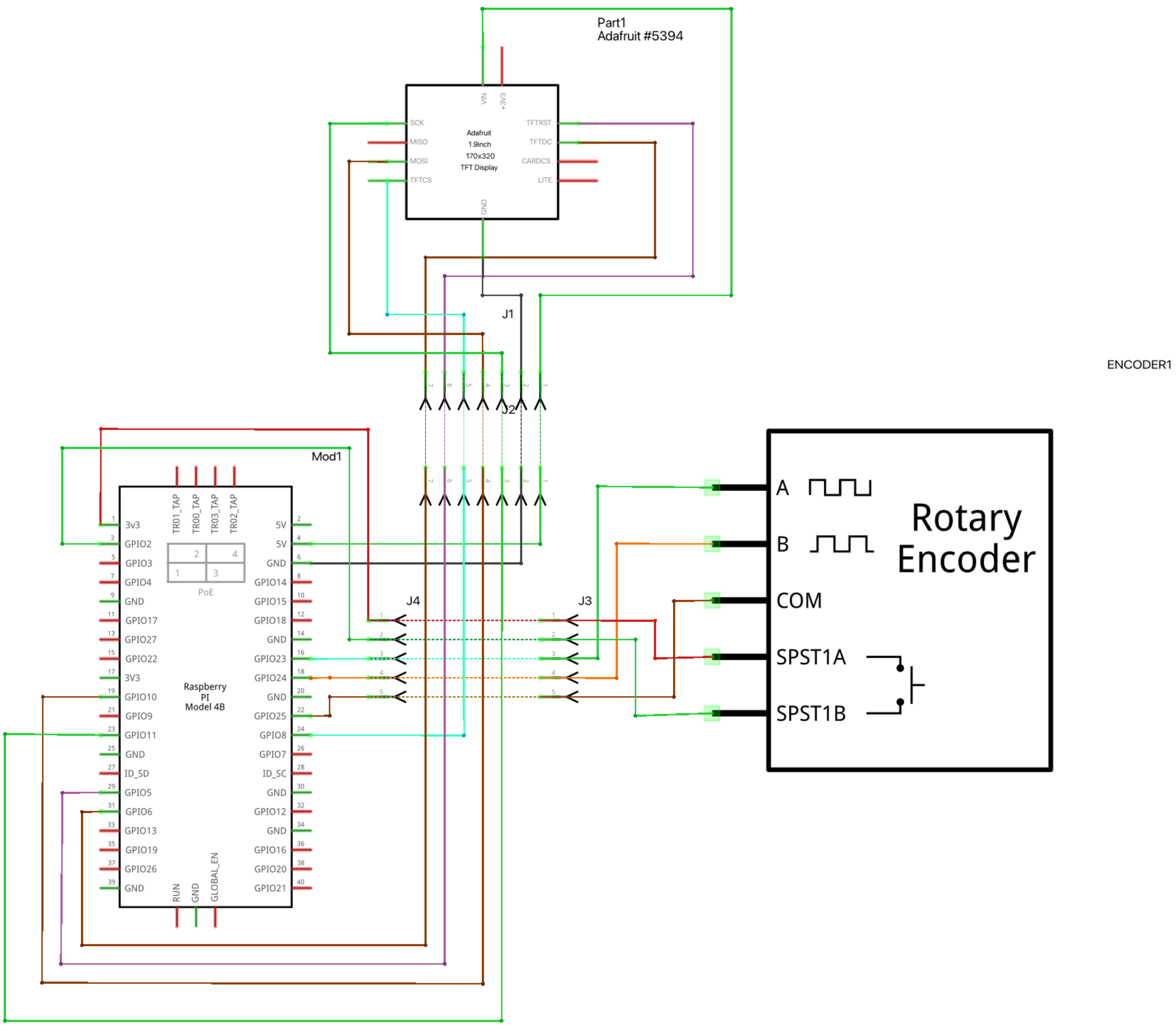

Figure 7. Device schematic diagram.

Figure 7 shows the connections between the Raspberry Pi and other components. Components are linked to a hub, which serves as an intermediary connection point, allowing for easier component replacement or upgrades without disturbing the core Raspberry Pi setup.

4.2. Main Display

Figure 8. 7 Inch HDMI LCD Display with Touch Screen (Elecrow, 2014)

The main display used in the device is a 7-inch screen with HDMI support. This display size was chosen because it is the most popular display size for similar devices, and larger screens were deemed too big for the intended use.

4.3. Sub-Display

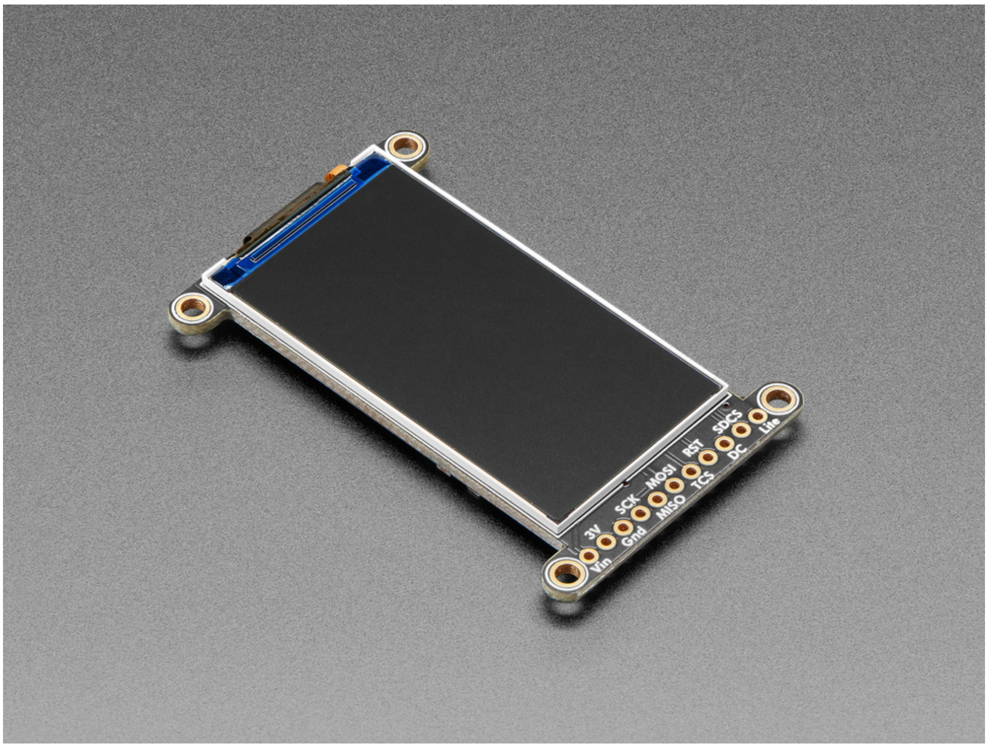

Figure 9. Adafruit 1.9" 320x170 Color IPS TFT Display (M. LeBlanc-Williams, 2022)

A 1.9-inch sub-display was chosen to handle secondary functions such as menu navigation and project descriptions. This display communicates with the Raspberry Pi using the SPI protocol. A custom user interface was developed from scratch, allowing for dynamic data feeds and automatic layout updates. This feature ensures that developers can efficiently update the UI without the need for manual reorganisation whenever new data or features are added.

4.4. Rotary Button Control

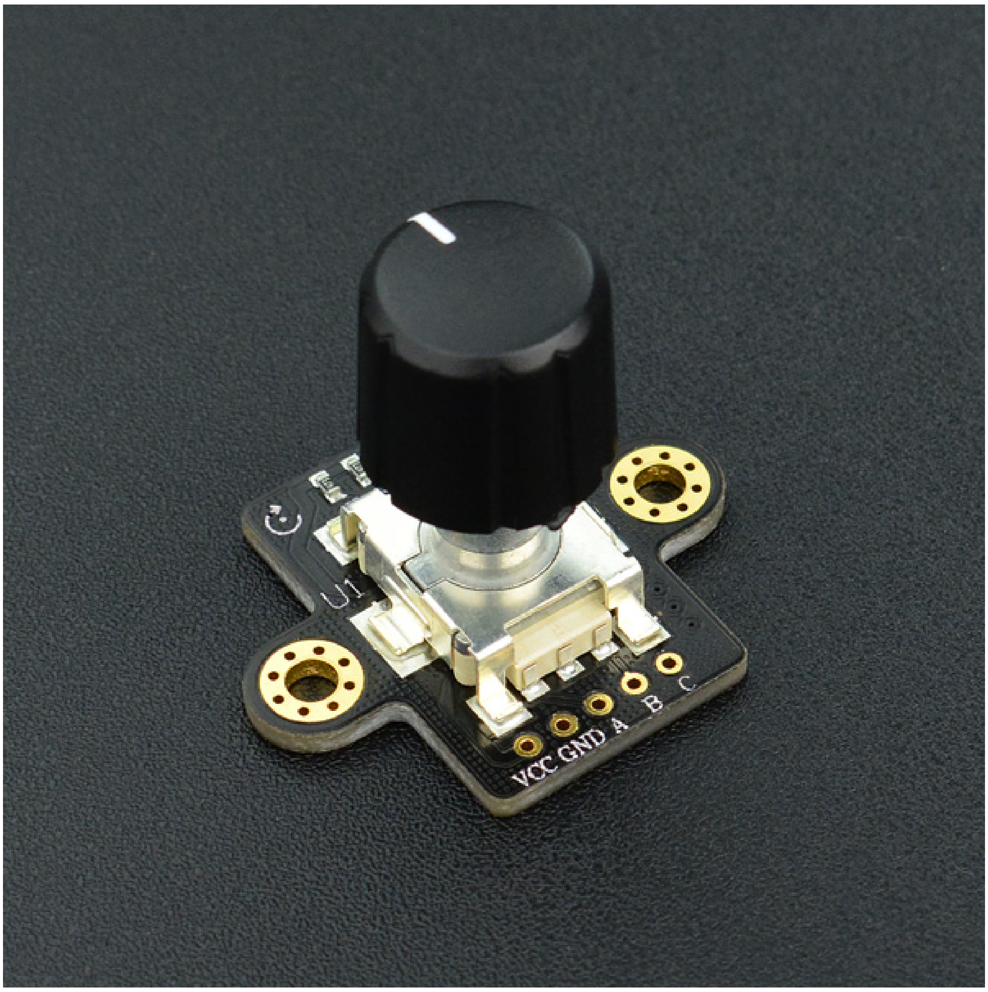

Figure 10. EC11 Rotary Encoder Module (DFRobot, 2018)

An EC11 rotary encoder was used to control the sub-display and navigate through the device's various features. The rotary encoder supports button actions, allowing users to select menu options by rotating the encoder and confirming their choices by pushing it. This design decision enhances the device's aesthetics and provides a user-friendly control mechanism.

Implementation

5.1. Application Architecture

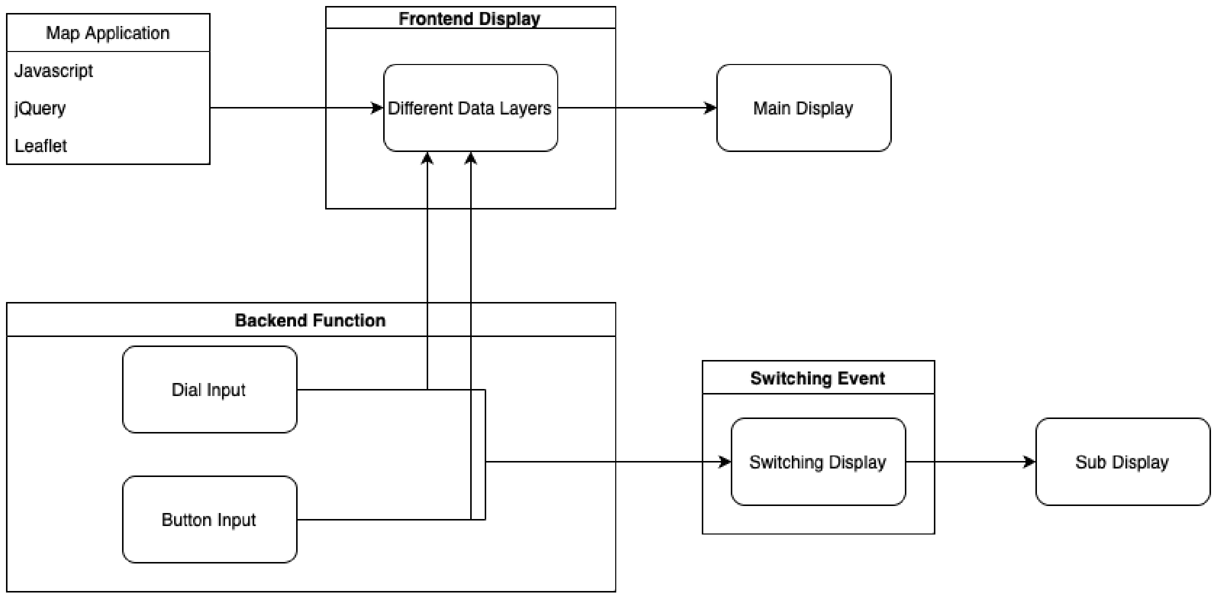

Figure 11. The web application structure

For the web application architecture, there are two main tasks:

1. Implement different types of data to each map layer and shown on the page.
2. Connect backend and frontend web application for users to choose the wanted layers to be shown.

5.2. Map Application Development

5.2.1. Leaflet Library Integration

We built an interactive map using Leaflet (Leaflet, 2023), a lightweight, open-source JavaScript library. It offers various plugins to extend its functionality. For our project, we used Open Street Map and Dark Canvas Map tiles for base layers. To provide smooth zooming, we implemented the Smooth Wheel Zoom plugin.

5.2.2. Data Layers Implementation

After the development of the fundamental map on the web application, we then build the overlayers of different types of data.

Weather Widget Layer

Figure 12. Weather widget on the screen.

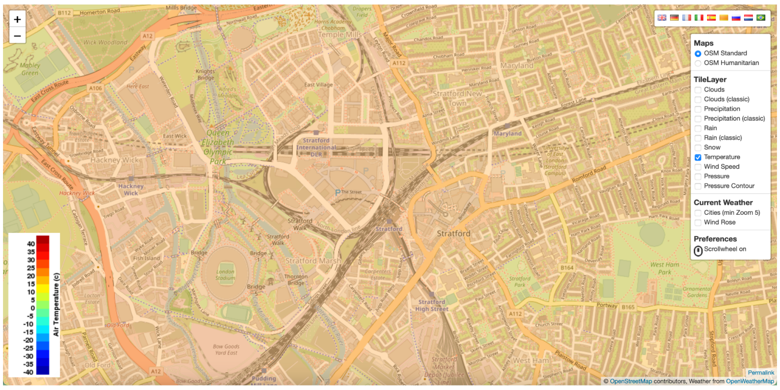

Figure 13. Temperature weather map.

We chose the OpenWeatherMap API (OpenWeatherMap, 2023) for our project due to its free subscription and ease of use. The API provides weather maps with a scale legend for use as layers in direct tiles in Leaflet, as shown in Figure 12. However, since our project requires a map of a small area, weather maps would display similar outcomes across the whole map, making it pointless and disorganised for users to fetch weather data (Figure 13). As a result, we developed a weather widget to display weather data on a small popup dashboard using an open-sourced Leaflet plugin and the OpenWeatherMap API. The widget displays weather information such as the icon, temperature, precipitation, humidity, visibility range, wind speed, and wind direction.

Network Coverage Layer

Figure 14. Network coverage layer

As shown in Figure 14, we built the network coverage layer using the Leaflet heatmap layer plugin (Wied, 2023), which allows for the customisation of options such as radius, opacity, data point coordinates, and data values. We transformed the data source (Ofcom, 2023) to a CSV format and used the median download speed as the value.

3D Buildings Layer

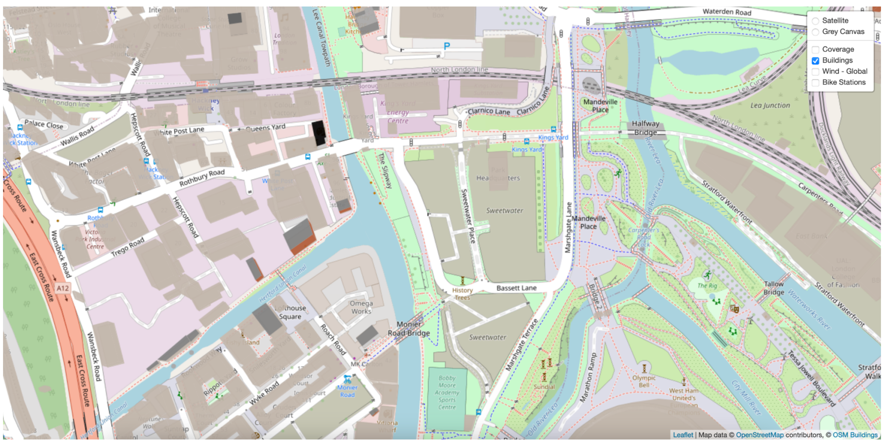

Figure 15. 3D building layer

We used the OSM 3D buildings library (OpenStreetMap contributors, 2017) to show building geometry on interactive maps, which supports Leaflet. OSM uses building GeoJson data such as height, colour, and shape.

Shadow Simulator Layer

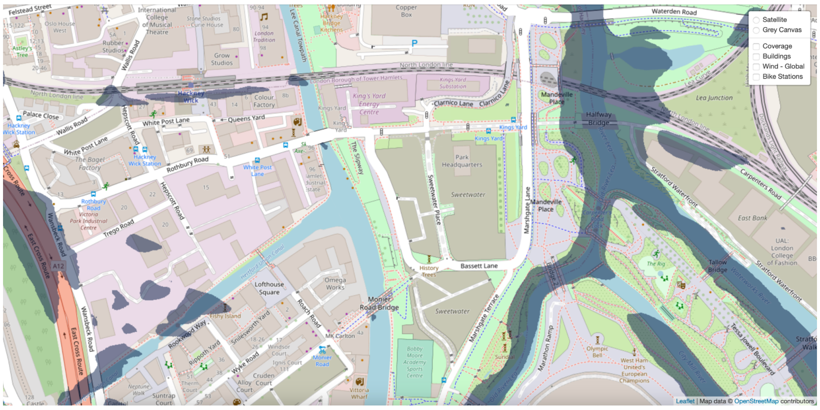

Figure 16. Shadow layer at the specified time

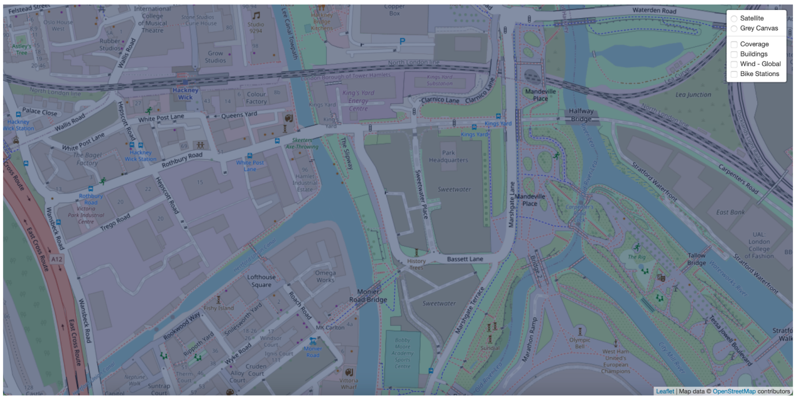

Figure 17. Shadow layer when sunset

We used the Leaflet-shadow-simulator plugin (Piotrowski, 2022) to show building shadows according to the time of day. However, we encountered issues with the shadows' appearance and alignment with the buildings and sun exposure (Figure 16).

Santander Cycles Availability Layer

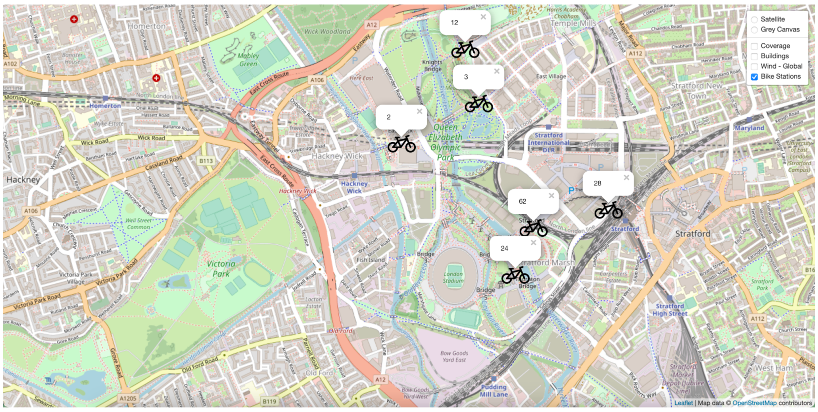

Figure 18. Available bikes with numbers

We built the bike layer using the marker layer in Leaflet, with data sourced from the TFL API (Transport for London, 2023) and filtered to only include bike stations within the Queen Elizabeth Park area (Figure 18). We attached popups to the markers, displaying the current number of bikes at each station. We also added a bicycle-style icon for better visual effects.

Wind Layer

Figure 19. Wind layer on dark canvas map

For the wind map layer (Figure 19), we initially tried Wind-JS but switched to the wind-js-leaflet plugin (Wild, 2023) due to its flexibility and more up-to-date features. The plugin adds a visualisation overlay of wind direction, velocity, and temperature, with customisation options for better aesthetics. The wind data is 1 degree, 6 hourly data downloaded from the NOAA website (National Centers for Environmental

Prediction, 2022), then we used grib2json (Beccario, 2023) to decode the grib2 file into a JSON structure with the grid represented as an array to be used in the plugin.

5.3. Backend Server Development

5.3.1. MQTT Feed Integration

We used MQTT for communication between the physical sensor (i.e., button) and the web application. When users select layers with the button, a JSON object is published to an MQTT broker, and the web application receives the message.

5.3.2. User Preference Management

We implemented a function to manage user-selected layers. The function iterates through the JSON array every 2 seconds, updating the layers on the map based on the user's preferences. This approach, however, is computationally inefficient, and future development should focus on improving its efficiency.

5.4. UI Design for Sub-Display

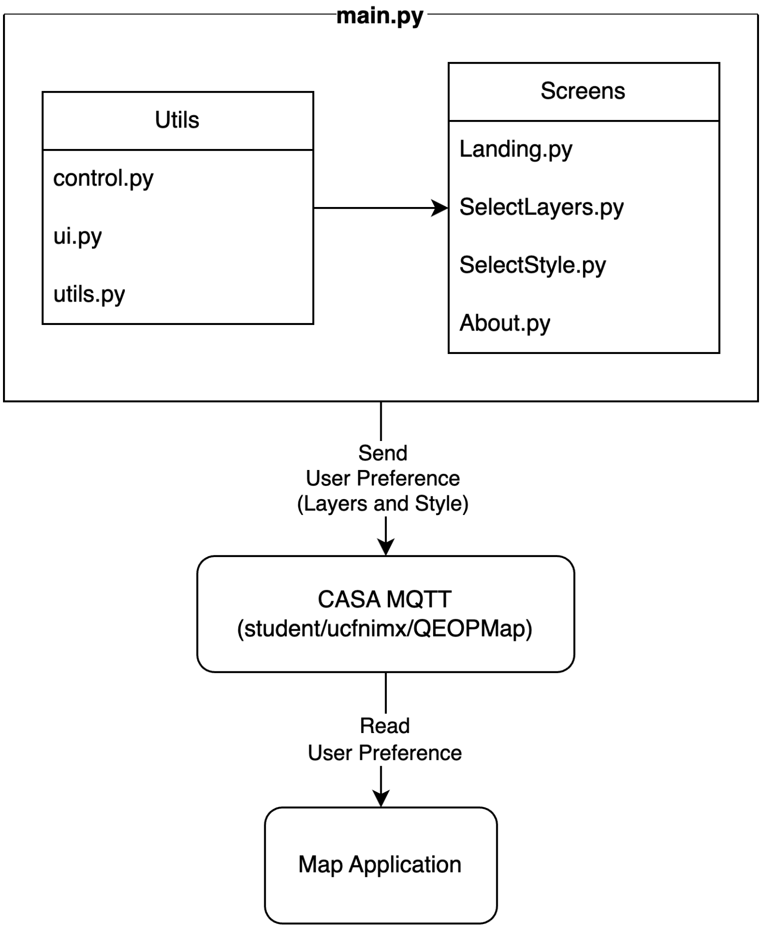

Figure 20. The programme structure of the menu display.

Figure 20 shows the program structure for the sub-display, designed to present map information and focus on the user interface. Adafruit's open-source graphics library for Python offers basic graphics functions (M. LeBlanc-Williams, 2022), making it easier to work with such display modules.

Figure 21. Layer selection screens.

We built a custom library on top of Adafruit to create a dynamic project menu, as shown in Figure 21. This menu facilitates smooth navigation through map layers and styles. User preferences are sent to the CASA MQTT server in JSON format for display on the main screen. The menu is developed using Python, while the map application operates in JavaScript. The MQTT server enables the communication between the components.

5.6. Enclosure

5.6.1 3D Printed Map

In order to optimise the 3D printed map for the device, we experimented with several prototypes using different settings (Figure 22). Typically, Stereolithography (SLA) printers are employed to achieve fully transparent and highly detailed models. However, the resin material used in SLA printers tends to be environmentally unfriendly and expensive.

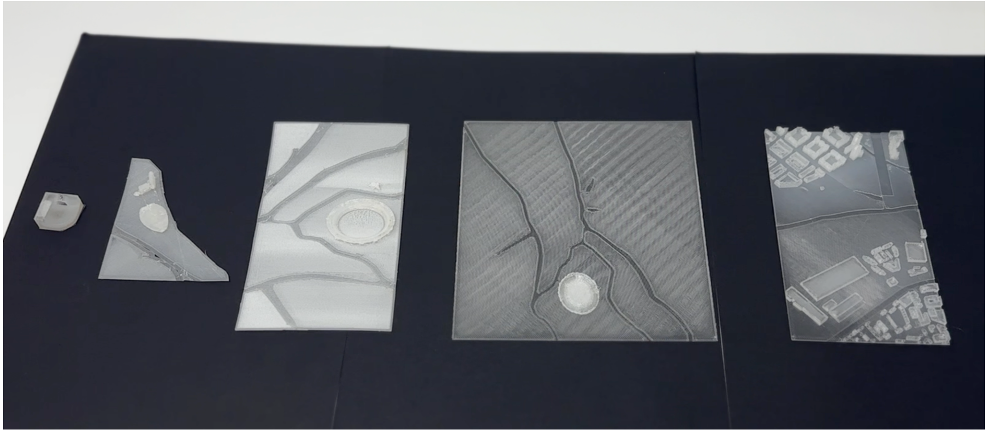

Figure 22. 3d printing prototypes.

One challenge with FDM printers is that they produce models by stacking layers, which causes light to diffuse through the layers and create a foggy appearance. To counter this issue, it is crucial to melt and align all the layers together (CNC KITCHEN, 2022).

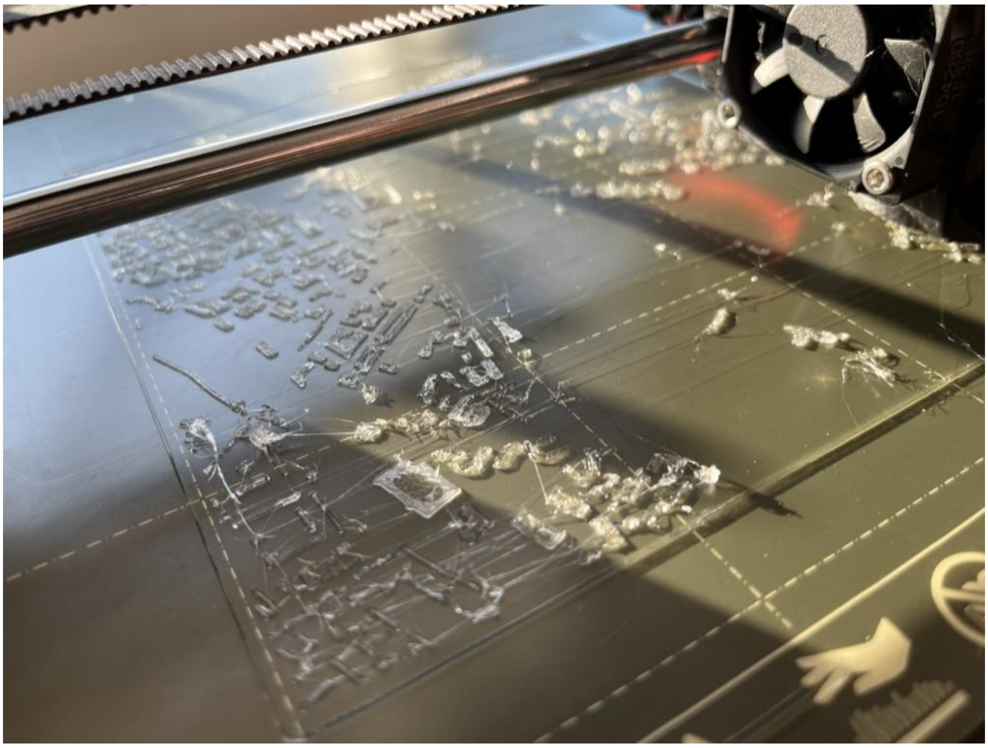

Figure 23. Tiny size buildings using the FDM printer.

However, our model required numerous small buildings in a compact area, and the transparency techniques we employed could not print such tiny pieces, as demonstrated in Figure 23. To overcome this limitation, we modified the map location, and zoom level to a more suitable area for printing and modified the transparency techniques. The selected area, Eastcross Bridge (Figure 24), provided a balance between the level of detail and the printer's limitations.

Figure 24. Eastcross Bridge area.

5.6.2 Wood Frame

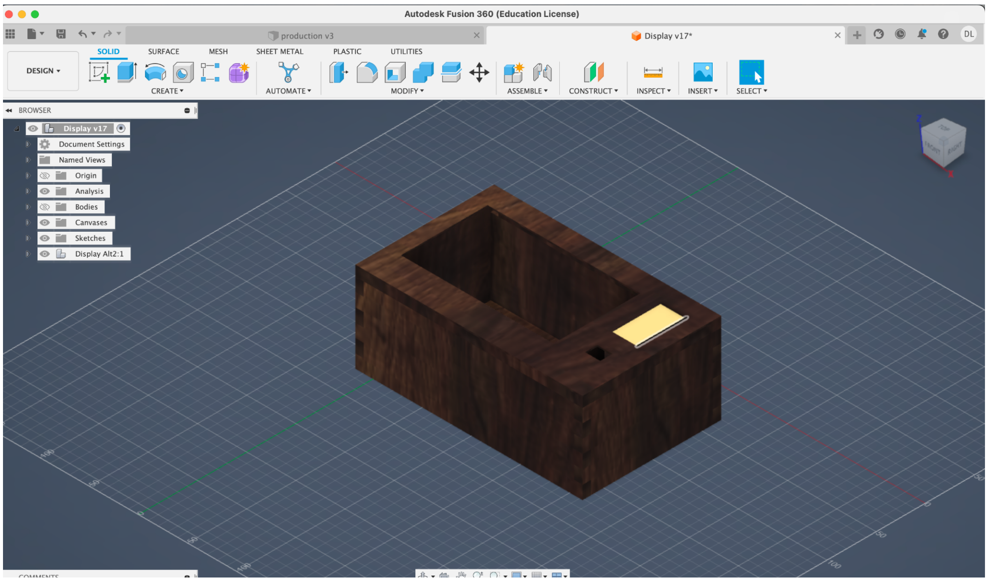

Figure 25. Virtual device model in Fusion 360.

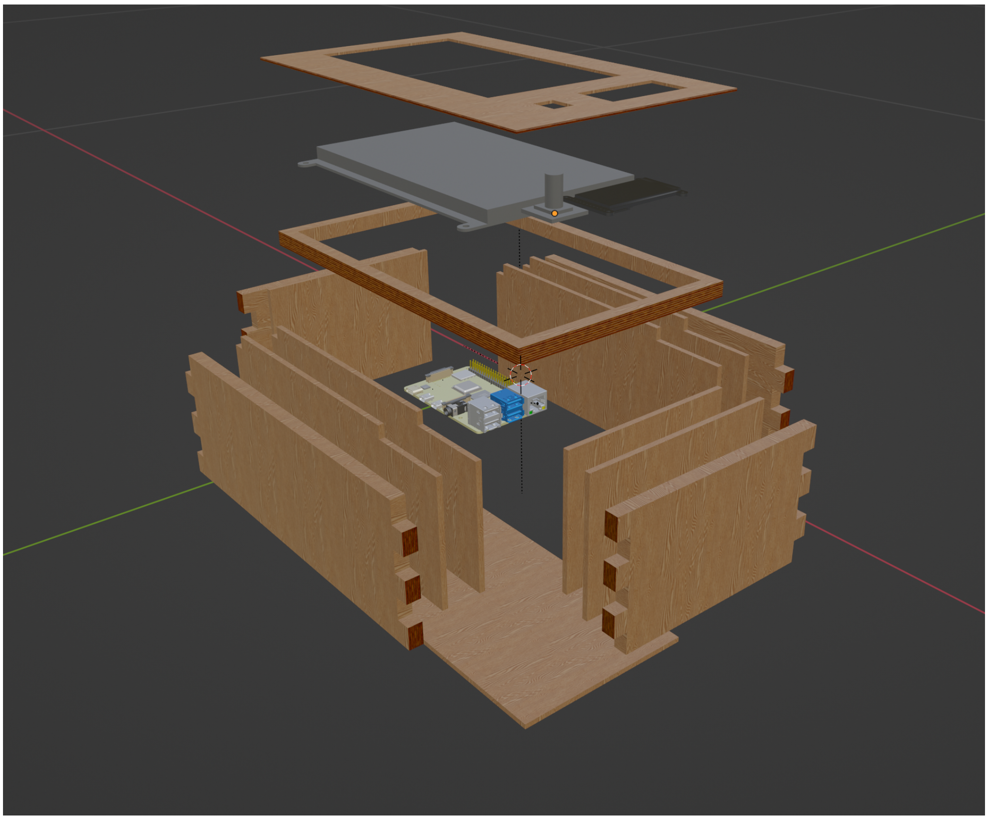

Figure 26. Assembly model in Blender.

Figures 25 and 26 show the virtual device model and assembly model in Fusion 360 and Blender, respectively. This streamlined process facilitated collaboration with UCL East's Fabrication Workshop during the laser cutting and assembly stages (Figures 27, 28 and 29).

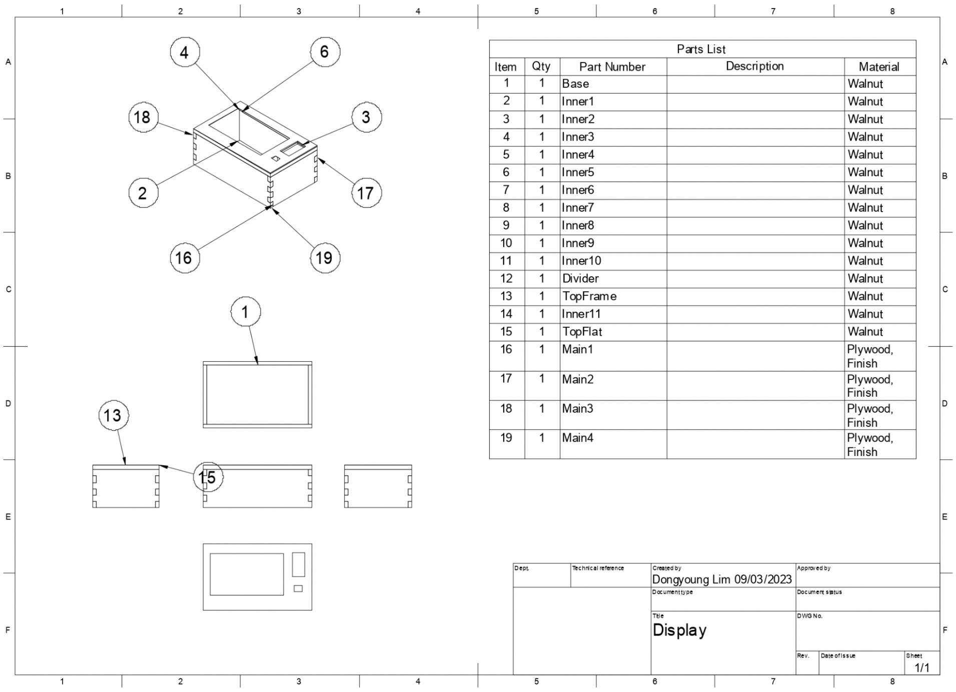

Figure 27. Assembly drawing.

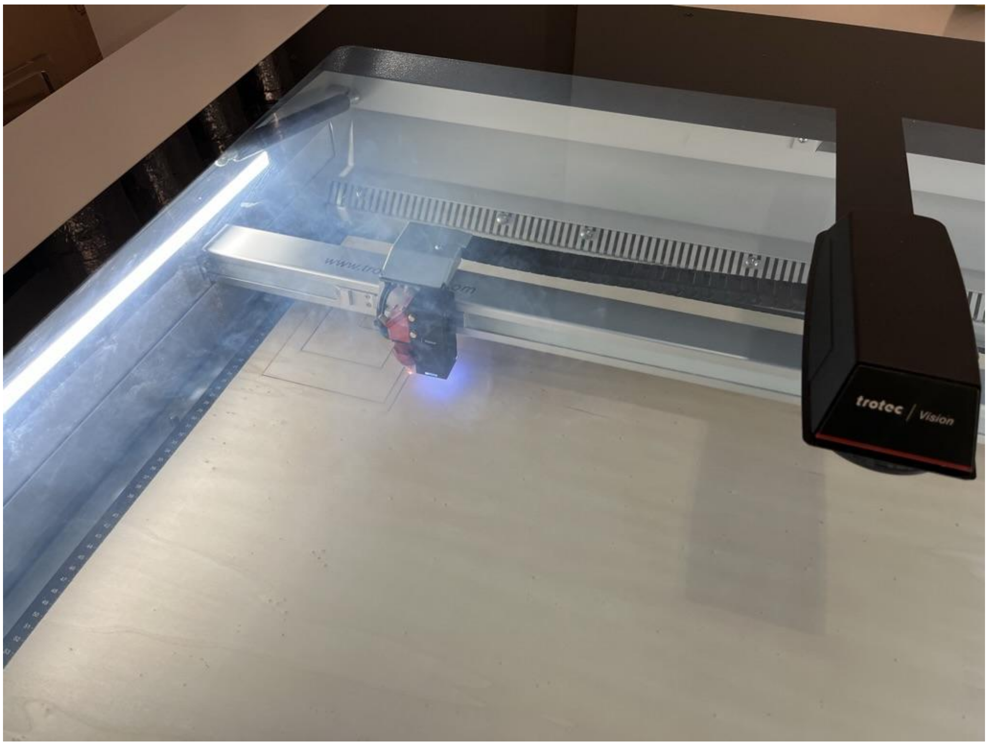

Figure 28. Plywood laser cutting in Fabrication Workshop

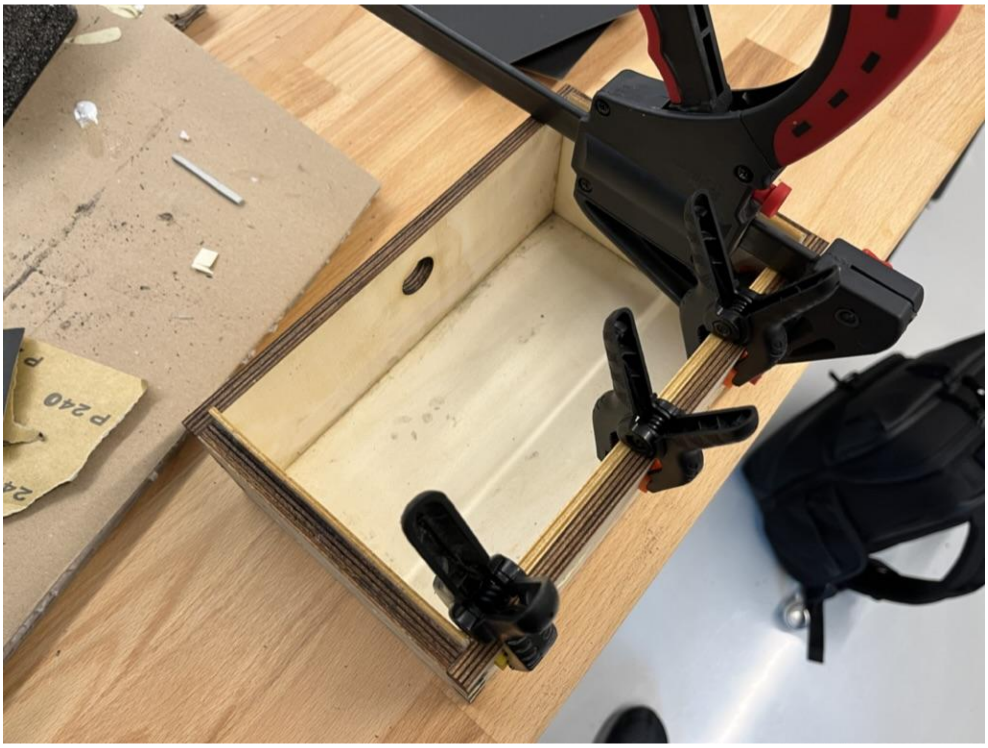

Figure 29. Assembling the parts in Fabrication Workshop

To achieve an oak-like appearance, we applied a wood stain to the plywood surfaces (Figures 30 and 31). To ensure durability, we utilised finger joints in the construction of the enclosure. The combination of these design choices resulted in a robust and visually appealing wood frame for the device.

Figure 30. Before applying the wood stein.

Figure 31. After applying the wood stein.

Future Work and Improvements

Despite our efforts to complete the project within the given timeline, we faced some challenges that led to certain compromises. Potential enhancements and expansions for future work are outlined below:

7.1. Design Enhancements

We attempted to reduce the device size for easy placement in various home settings, but the HDMI cable's thickness and length required a larger enclosure. To address this issue, we could use a different display unit or a thinner cable.
(Figure. Cable in the device)

As shown in Appendix 1, the Raspberry Pi 4 accounts for more than half of the device's cost due to the recent chip shortage. Moreover, its computing power far exceeds the project's requirements. Alternative options (Figure 32) include using the more affordable Raspberry Pi Zero, which supports the Linux operating system. Another option is replacing it with a microcontroller like Arduino or ESP series, which are highly energy-efficient and affordable compared to the Raspberry Pi. However, this would require optimizing the map application to run on such microcontrollers. Also, the saved cost can be partially used to replace the main display with higher brightness for better visibility.

Figure 32. Alternative controllers.

Additionally, we could optimize data communication by replacing MQTT with internal data communication between Python-based and JavaScript-based programs within the operating system. This change could result in faster and more concise data processing.

7.2. Feature Expansions

Our initial design aimed to support custom locations beyond QEOP. While the map application is designed for this feature, several improvements could be made for a seamless user experience.

For example, the device could recognize the 3D printed map using a tag, such as NFC or a colour sensor. NFC is highly precise but requires embedding a chip in the map, adding extra cost. Alternatively, a colour sensor could be used, allowing users to mark a specific area on the map for the device to read and display accordingly.

Several sensors are available in QEOP, such as bat sensors and weather stations, but the layers were not included due to technical issues and time constraints. Future work could incorporate data from LoRa networks and collaborations with the London Legacy Development Corporation (LLDC) to expand the device's features and capabilities, such as ongoing events and the water flow of river Lea. This expansion could lead to a more versatile and powerful device for users in various locations.

Conclusion

In conclusion, the project aimed to bridge the gap between parks and homes by allowing users to preview and experience a location before physically visiting the park, with Queen Elizabeth Olympic Park as our pilot location. We developed a data dashboard device with an integrated 3D printed map, focusing on creating an affordable, aesthetically pleasing, and intuitive product.

Throughout the project, we learned the importance of planning, execution, and integration of components to ensure an optimal user experience. The device allows users to engage with their favourite locations virtually, enriching their overall experience and motivating them to explore these areas in person.

While we achieved considerable progress, there are opportunities for further research and development. By broadening the scope and refining the design, the device could become a more versatile and powerful tool for users to connect with various locations and enhance their exploration of the world around them.

References

adafruit, 2023. adafruit/Adafruit-ST7735-Library [WWW Document]. URL https://github.com/adafruit/Adafruit-ST7735-Library (accessed 4.8.23).

Batty, M., 2015. A perspective on city dashboards. Regional Studies, Regional Science 2, 29–32. https://doi.org/10.1080/21681376.2014.987540

Beccario, C., 2023. grib2json.

CNC KITCHEN, 2022. Transparent FDM 3D Prints are Clearly Stronger! [WWW Document]. CNC Kitchen. URL https://www.cnckitchen.com/blog/transparent-fdm-3d-prints-are-clearly-stronger (accessed 4.10.23).

DFRobot, 2018. EC11_Rotary_Encoder_Module_SKU__SEN0235-DFRobot [WWW Document]. URL https://wiki.dfrobot.com/EC11_Rotary_Encoder_Module_SKU__SEN0235 (accessed 4.8.23).

Elecrow, 2014. 7 Inch 1024*600 HDMI LCD Display with Touch Screen [WWW Document]. URL https://www.elecrow.com/wiki/index.php?title=7_Inch_1024*600_HDMI_LCD_Display_with_Touch_Screen (accessed 4.8.23).

Hudson-Smith, A., 2014. Tracking, Tagging and Scanning the City. Architectural Design 84, 40–47. https://doi.org/10.1002/ad.1700

Hudson-Smith, A., Hügel, S., Roumpani, F., 2020. Self-monitoring, analysis and reporting technologies: Smart cities and real-time data, in: The Routledge Companion to Smart Cities. Routledge.

Leaflet, 2023. Leaflet/Leaflet.

M. LeBlanc-Williams, 2022. Adafruit 1.9" Color IPS TFT Display [WWW Document]. Adafruit Learning System. URL https://learn.adafruit.com/adafruit-1-9-color-ips-tft-display/overview (accessed 4.8.23).

Mark Linnane, n.d. Data City. mark linnane. URL https://marklinnane.net/portfolio/building-city-dashboards/ (accessed 4.4.23).

Matheus, R., Janssen, M., Maheshwari, D., 2020. Data science empowering the public: Data-driven dashboards for transparent and accountable decision-making in smart cities. Government Information Quarterly 37, 101284. https://doi.org/10.1016/j.giq.2018.01.006

McArdle, G., Kitchin, R., 2016. The Dublin Dashboard: Design and development of a real-time analytical urban dashboard. Presented at the First International Conference on Smart Data and Smart Cities; ISPRS Annals Photogrammetry, Remote Sensing and Spatial Information Sciences, III-4/W1, Split, Croatia, pp. 19–25.

mutsuyuki, 2023. Leaflet.SmoothWheelZoom.

National Centers for Environmental Prediction, 2022. NCEP Data Products GFS and GDAS [WWW Document]. URL https://www.nco.ncep.noaa.gov/pmb/products/gfs/ (accessed 4.19.23).

Ofcom, 2023. Open data [WWW Document]. Ofcom. URL https://www.ofcom.org.uk/research-and-data/data/opendata (accessed 4.19.23).

OpenStreetMap contributors, 2017. Planet dump retrieved from https://planet.osm.org.

OpenWeatherMap, 2023. OpenWeatherMap for Leaflet based maps.

Piotrowski, T., 2022. leaflet-shadow-simulator.

Rygar1432, n.d. How To Print Glass by Rygar1432 | Download free STL model [WWW Document]. Printables.com. URL https://www.printables.com/model/15310-how-to-print-glass (accessed 1.23.23).

Transport for London, 2023. Our Unified API [WWW Document]. Transport for London. URL https://www.tfl.gov.uk/info-for/open-data-users/api-documentation (accessed 4.19.23).

Wied, P., 2023. heatmap.js.

Wild, D., 2023. wind-js-leaflet.

Appendix 1. Bill of Material

Bill of Materials

Assembly List

Label	Part Type	Properties	Price	Link
ENCODER1	EC11 Rotary Encoder Module	Module Size: 33.8 * 22.4 (mm) /1.3 * 1.1 (inches)	£3.00	https://shop.pimoroni.com/products/ec11-rotary-encoder-module
J1	Generic female header - 7 pins	package THT; pins 7; hole size 1.0mm,0.508mm; row single; pin spacing 0.1in (2.54mm); form ♀ (female)	£2.00	https://shop.pimoroni.com/products/female-headers
J2	Generic female header - 7 pins	package THT; pins 7; hole size 1.0mm,0.508mm; row single; pin spacing 0.1in (2.54mm); form ♀ (female)		
J3	Generic female header - 5 pins	package THT; pins 5; hole size 1.0mm,0.508mm; row single; pin spacing 0.1in (2.54mm); form ♀ (female)		
J4	Generic female header - 5 pins	package THT; pins 5; hole size 1.0mm,0.508mm; row single; pin spacing 0.1in (2.54mm); form ♀ (female)		
Mod1	Raspberry Pi 4B	processor Broadcom 2711, Quad-core Cortex-A72 64-bit SoC @ 1.5GHz; variant Raspberry Pi 4B	£62.40	https://shop.pimoroni.com/products/raspberry-pi-4?variant=29157087445075
Part1	Adafruit 1.9in 320x170 TFT	variant variant 1; part # Adafruit #5394	£17.10	https://shop.pimoroni.com/products/adafruit-1-9-320x170-color-ips-tft-display-st7789
	Elecrow RC070LCD Display with Touch Screen	7 Inch 1024*600 HDMI	£50.00	https://www.amazon.co.uk/gp/product/B076J8ZWFF
		Total	£134.50	

Enclosure

Label	Part Name	Size	Price	Link
	Plywood - 3mm	600mm x 300mm	£4.00	https://www.amazon.co.uk/Trustleaf-300x600mm-Plywood-Sheets-SECONDS/
	Plywood - 8mm	600mm x 300mm	£12.45	https://www.amazon.co.uk/Birch-Plywood-making-furniture-300mm/
	Furniture Clinic Wood Stain	250 ml	£9.45	https://www.amazon.co.uk/Furniture-Clinic-Stain-250ml-Mahogany/
		Total	£25.90	
		Grand Total	£160.40	