Drone Control and Monitoring: An Android Application with Real-Time Navigation Features

1st Feras Mohammad
Computer Engineering
University of Jordan
0208687
Amman, Jordan
fra0208687@ju.edu.jo

2nd Amr Abuzayyad
Computer Engineering
University of Jordan
0214597
Amman, Jordan
amr0214597@ju.edu.jo

3rd Saleh Al-Ghobari
Computer Engineering
University of Jordan
0214044
Amman, Jordan
sal0214044@ju.edu.jo

***Abstract*—This project focuses on developing an Android smartphone application that enables remote navigation and monitoring of an unmanned aerial vehicle (UAV). The primary objective is to provide a simple user-friendly interface for defining a search area using GPS coordinates, setting a destination point, and controlling the drone's movement via satellite communication. Although the application is designed to incorporate TCP communication for enhanced control and data transfer, this functionality is not implemented yet.**

The application integrates OpenStreetMap and Google Maps to display real-time drone location updates on a map interface, supporting both street and satellite views. Key functionalities include translating GPS coordinates into scaled screen coordinates for precise control and dynamically changing drone icons based on status.

The outcome is an intuitive application that enhances UAV navigation, offering real-time feedback and visualization. This project demonstrates the effective integration of mapping technologies and user-centric design to achieve seamless drone control.

***Index Terms*—Drone Control, Remote Navigation, Dynamic Marker Icons, Street and Satellite Maps, Android Application, Coordinate Scaling, Real-Time Position Tracking**

I. Introduction

The integration of unmanned aerial vehicles (UAVs) into various applications has opened new possibilities to enhance efficiency and precision in diverse fields. However, effectively controlling and monitoring drones from a remote location remains a challenge. This project addresses this issue by developing an Android smartphone application that provides a robust interface for remotely controlling a drone and monitoring its operations in real time.

The core problem is enabling Dr. Kamal, the intended user, to define a search area, set a destination for the drone, and monitor its real-time location with ease. By addressing this, the application aims to simplify UAV operations by transforming complex drone navigation tasks into intuitive user interactions. OpenStreetMap was used as the mapping platform to display the search area and drone movement. The app scales and converts GPS coordinates into x-y screen coordinates, allowing precise control over the drone's destination.

Key features include two map views (street and satellite),[1] dynamic icon selection for drone visualization, and a location generator that simulates the drone movement to the destination defined through the sliders. [2] The motivation for this project lies in the growing demand for accessible and user-friendly tools to manage UAVs in scenarios such as search and rescue, surveillance, and geographic exploration. This application bridges the gap between technical UAV capabilities and user-centered design, providing an efficient solution for remote drone operation.

Through this project, the focus was on delivering a scalable, interactive, and efficient application to meet the requirements of a wide range of UAV users while prioritizing usability and integration with existing technologies.

II. System Design

The proposed application for drone control and monitoring utilizes a smartphone interface to streamline the interaction with the unmanned aerial vehicle (UAV). The system design is structured to address two fundamental requirements: control and monitoring. The control aspect allows users to define destination coordinates for the drone within a specified search area, while the monitoring component enables real-time retrieval and display of the drone's current location.

A. Control System Features:

- Two sliding bars (SeekBars) allow for precise adjustments of X and Y coordinates that are scaled to fit the boundaries of a user-defined rectangular search area. These values are converted into GPS coordinates before being transmitted to the drone.
- The "GO" button ensures seamless communication with the drone by establishing a TCP connection over a satellite link. TCP connection is implied; the exact implementation is left to be added.

B. Application Architecture and Data Flow:

The application architecture is modular, comprising distinct UI layers that communicate seamlessly to provide a user-friendly and efficient experience. This design ensures a clear separation of concerns, making the system easier to debug, maintain, and expand. The data flow is considered as:

1) Input Phase: On the first screen, users input the bounding box coordinates for the search area, defining the operational region for the drone. These inputs are processed and validated before proceeding.

2) Map Retrieval: The app retrieves map data from internet services. Based on user selection, either a street map or a satellite image is displayed on the second screen.

3) Coordinate Transformation: Scaled X and Y slider values are converted into GPS coordinates that the drone can interpret. This transformation ensures that user input aligns accurately with real-world positions.

4) Feedback Phase: The drone's current GPS location is retrieved, processed, and displayed on the map and in message boxes for user review.

The event-driven design ensures that every user interaction — be it slider adjustment, map type selection, or button presses — triggers the appropriate logic, maintaining a smooth and responsive user experience.

C. Details of the Map Retrieval and Coordinate Conversions

The application integrates two map types: street map sourced from OpenStreetMap, and a satellite map provided by Google Satellite services. These maps are retrieved dynamically based on user input, ensuring that the displayed area corresponds to the defined search boundaries.

The map type can be switched seamlessly using a dropdown menu. This feature enhances usability, allowing users to select the view most suited to their operational needs.

The application uses Mercator projection to convert GPS coordinates into x-y positions that can be accurately displayed on the phone's screen, as well as calculate the correct zoom level. This process involves two key transformations:

1) Latitude to Y Conversion: The conversion employs the following formula:-

$$Y = \left(1 - \frac{\log(\tan(\text{rad}) + \sec(\text{rad}))}{\pi}\right) \times \frac{2^z}{2}$$

Here, 'rad' represents the latitude in radians, and 'z' is the zoom level. This ensures that the y-coordinate reflects the vertical position accurately across varying zoom levels.

2) Longitude to X Conversion: Longitude values are transformed into x-coordinates using the formula:-

$$X = \frac{\text{longitude} + 180}{360} \times 2^z$$

This calculation ensures a one-to-one mapping of longitude values to horizontal positions on the map.

3) Zoom Level Selection: The function determines the appropriate zoom level for displaying the map by iterating through zoom levels in descending order, starting from the maximum zoom level (`maxZoomLevel` = 22) down to the minimum zoom level (`minZoomLevel` = 0). For each zoom level, the following computations are performed:

$$x_{\text{min}} = \text{lonToX}(\text{lonMin}, z), \quad x_{\text{max}} = \text{lonToX}(\text{lonMax}, z)$$

$$y_{\text{min}} = \text{latToY}(\text{latMax}, z), \quad y_{\text{max}} = \text{latToY}(\text{latMin}, z)$$

Here, `lonToX` and `latToY` are functions that convert longitude and latitude values into their corresponding x and y coordinates on the map, based on the given zoom level z.

The zoom level z is selected when the following condition is satisfied:

$$x_{\text{max}} - x_{\text{min}} \leq 4 \quad \text{and} \quad y_{\text{max}} - y_{\text{min}} \leq 4$$

Once this condition is met, the function returns the zoom level z, ensuring that the map view accurately fits the defined bounding box.

III. Implementation Details

A. Development Environment

The development of the application was carried out using Android Studio and Flutter. Android Studio served as the primary integrated development environment (IDE) for coding, testing, and debugging. Its robust suite of tools facilitated the seamless integration of features such as map retrieval, GPS handling, and user interface design.

Flutter, a cross-platform framework by Google, was employed for creating the application's user interface. [3] Flutter's widget-based architecture allowed for building a simple user interface, while its support for platform-specific functionalities enabled the integration of native Android features. [4] The combination of Android Studio and Flutter ensured an efficient development workflow, providing a scalable and maintainable architecture for the application.

B. User Interface Components

- SeekBars: These sliding bars allow users to define scaled X and Y values for the drone's destination. The sliders offer intuitive control and provide real-time updates to the marker position on the map.
- Dropdown Menus: Two dropdown menus enhance user interaction, where the first allows switching between map types (Street Map and Satellite View), The second menu toggles between different drone icons, offering options for better visualization.
- Buttons: The "GO" button transmits the user-defined coordinates to the drone, while the "GET" button retrieves the drone's current GPS location and displays it in the UI.

IV. Testing and Results

A. Testing Methodology

The application underwent extensive testing to validate its functionality, ensuring all features worked as intended. The first step involved entering bounding box coordinates on the initial page. The purpose of this test was to confirm that the map was accurately retrieved and displayed according to the provided coordinates. Users could define specific regions of interest, and the application successfully generated precise map views, reflecting the entered data without any noticeable errors. This functionality was critical as it served as the foundation for subsequent interactions within the application.

Next, the sliders for adjusting the X and Y values were tested to evaluate their responsiveness and real-time effect on the map. As users manipulated these sliders, the marker on the map moved dynamically, providing immediate visual feedback. This test confirmed that the application could handle smooth, real-time updates, a crucial feature for enhancing the user experience. The responsiveness of the marker movement demonstrated the application's ability to process and reflect user inputs seamlessly, ensuring accurate representation of positional adjustments.

The "GET" button functionality was also rigorously tested. This button was designed to retrieve simulated GPS coordinates from the application. Upon pressing it, the retrieved coordinates were promptly displayed on the interface, matching expectations. This process validated the application's capability to handle data retrieval operations efficiently and confirmed that the displayed coordinates were accurate and aligned with the simulated GPS data. This step was vital for verifying the application's core purpose of integrating mapping functionalities with GPS simulation.

Finally, the dropdown menus for switching between different map types and drone icons were tested to ensure seamless functionality. Users could effortlessly toggle between various map views, such as satellite or street-view, and change drone icons to customize the interface. These tests highlighted the application's versatility and ability to cater to diverse user preferences. The transitions between selections were smooth, without any glitches, affirming the application's robustness in managing visual elements and configurations.

These comprehensive tests collectively validated the application's functionality, demonstrating its reliability and user-centric design.

B. Results

The application successfully retrieved and displayed maps based on user-defined coordinates. The maps were scaled accurately, and the zoom levels adjusted dynamically to fit the specified search area. This feature ensured that users could define precise operational regions for the drone with minimal effort, showcasing the reliability of the map retrieval and scaling process.

The drone marker on the map responded smoothly to slider adjustments, providing real-time feedback to the user. As the sliders were manipulated, the marker moved dynamically across the map, accurately reflecting the updated destination coordinates. This responsiveness demonstrated the effectiveness of the coordinate transformation algorithms and highlighted the application's ability to process and display positional data seamlessly.

All user interface components, including buttons, dropdown menus, and sliders, functioned as intended. The "GET" button retrieved and displayed simulated GPS coordinates promptly, And Dropdown menus allowed users to switch between map types and customize drone icons effortlessly, enhancing the overall user experience and versatility of the application.

V. Conclusion

The developed application successfully met its objectives by providing robust control and monitoring functionalities for UAVs. The integration of real-time map retrieval, coordinate conversion, and intuitive UI components resulted in a user-friendly experience.

A. Achievements

- Accurate map retrieval and scaling based on user-defined coordinates. - Real-time drone position monitoring and marker movement. - Reliable data transmission via TCP over satellite links.

B. Suggestions for Improvement

- Incorporate additional map layers, such as topographic maps, for enhanced situational awareness. - Add functionality for automated route planning and obstacle detection. - Add proper TCP connection with another device to send information to a receiver

This project lays the groundwork for future enhancements, ensuring scalability and adaptability for a wide range of UAV applications.

Acknowledgment

The authors would like to express their gratitude to Dr. Talal Edwan for his invaluable guidance and dedication in the Mobile Computing course. His efforts in fostering a deep understanding of the subject and his continuous support have greatly contributed to the successful completion of this work.

References

[1] CARTO API, "CARTO API Documentation," 2024. [Online]. Available: https://api-docs.carto.com/. [Accessed: Dec. 26, 2024].
[2] Google Developers, "Google Maps Platform — Google Developers," 2019. [Online]. Available: https://developers.google.com/maps/documentation.
[3] Flutter, "Flutter documentation," docs.flutter.dev. [Online]. Available: https://docs.flutter.dev/.
[4] The Net Ninja, "Flutter Tutorial for Beginners - YouTube," YouTube. [Online]. Available: https://www.youtube.com/playlist?list=PL4cUxeGkcC9jLYyp2Aoh6hcWuxFDX6PBJ.