Informatics Large Practical

Coursework 2 Report

Duan Wenxing
(s1915425)
November 27, 2021

1. Software Architecture Description

1.1 Overview

The section explains the functionality of all classes, the architecture of the system and the class diagram and sequence diagram of the system.

1.2 Project Description

This project is an in-school drone delivery project. The purpose of the project is to write a planning and storage system for the drone delivery process. The system searches the orders of a given date, searches for the store location and destination of the order and plans a path. Finally, the order information, cost and path of the successfully delivered orders are written into the corresponding database. The system's path planning algorithm will try to find the shortest path and avoid the no-fly zone. At the same time, the path planning algorithm will also ensure that the drone has enough power to return to the starting point (Appleton Tower).

1.3 Class Description

1.3.1 App.java

This class is the main class of the project. It receives information of the date, month, year and port number of web and database from outside the system and transmit to PathBuilder.

1.3.2 DatabaseUtils.java

This class is responsible for writing and searching the database. It can search all order numbers based on date, search WhatThreeWord destinations and order items based on order numbers and can also write order information and flight path into the corresponding table.

1.3.3 GeoJsonUtils.java

This class is responsible for parsing and writing geojson files. This class can obtain the geojson data of landmarks and no-fly zones through serverUtils.java and parse it into LongLat type data then return. This class can also convert the final flight path into geojson data, create a corresponding geojson file and write the data.

1.3.4 LongLat.java

This class is used to store location information. This class contains the latitude and longitude of a location. It can calculate the straight-line distance between two positions and judge whether the two positions are close or not exactly the same. Also, it can also calculate the position of a point after moving one step in the specified direction, and judge whether a point is in the legal area.

1.3.5 LongLatCatcher.java

This class is responsible for parsing the address of the WhatThreeWord location. It receives the String of WhatThreeWord and returns the LongLat data at the center of this location.

1.3.6 MenuUtils.java

This class is responsible for searching the price and selling address of the item from the menu, and then return.

1.3.7 OrderDetail.java

This class is responsible for storing the data in an order, including the order number, date, student number of the person who ordered the meal, destination and items ordered.

1.3.8 PathBuilder.java

This class is responsible for completing the main processes of the system, including taking orders, finding coordinates, planning paths, and writing to the database. It first obtains the order detail data through DatabaseUtils, then obtains the destination that the order needs to pass through MenuUtils and LongLatCatcher, then plans a reasonable path through PathUtils, and finally writes the order information and flight path through DatabaseUtils. In this process, it will ensure that the flight always has enough power to return to the starting point (Appleton Tower) and that it will eventually return to the starting point. At the same time, it can also ensure that each step of the flight is 0.00015-unit length, and the angle is always a multiple of 10.

1.3.9 PathUtils.java

This class is responsible for providing tools for path planning. For example, use the mathematical method mentioned in **2.2.2** to plan the path between two points, determine whether the route can be safely flown (not through the no-fly zone or exceed the range), and find the path between the two points that passes the landmark. And some useful calculators such as calculating the length of a whole path, calculating the power consumption and so on.

1.3.10 ServerUtils.java

This class is responsible for obtaining information from the web server and returning it in the form of String.

1.4 UML Class Diagram

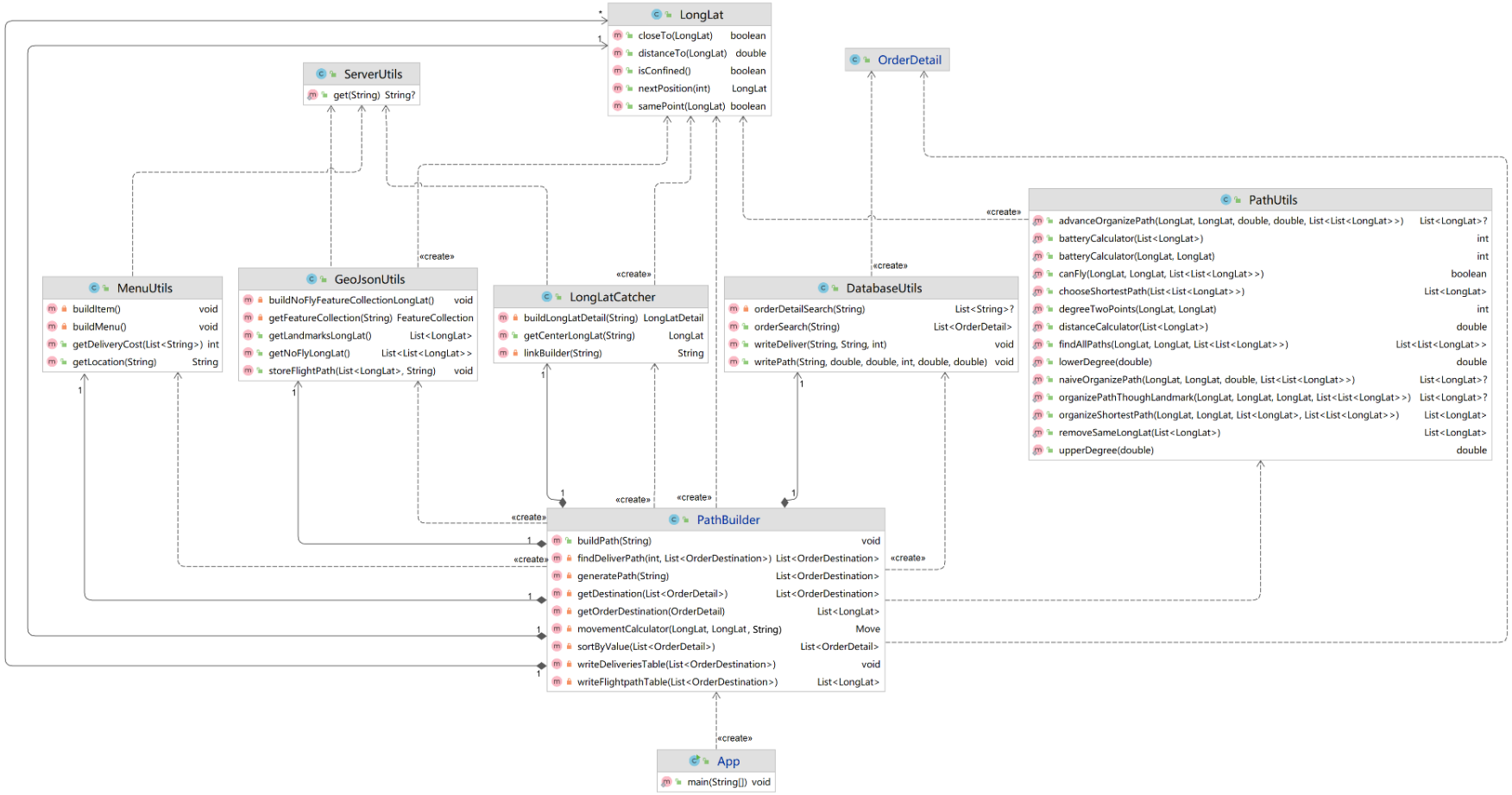


Figure 1: UML Class Diagram

1.5 UML Sequence Diagram

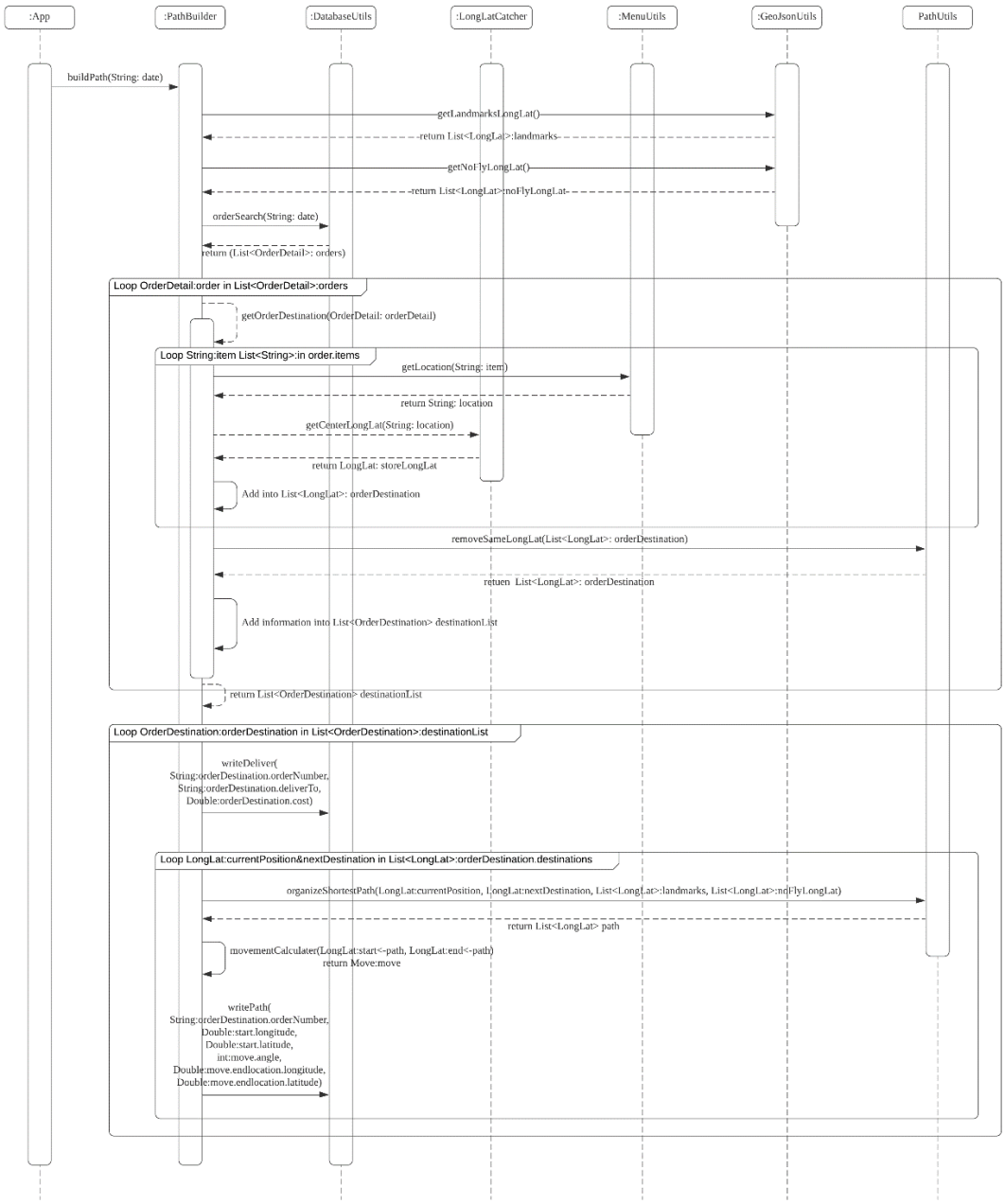


Figure 2: UML Sequence Diagram

1.6 Dependence

- The Mapbox Java SDK[1]
- Package java.awt.geom[2]
- Google Gson API[3]

[1] https://docs.mapbox.com/

[2] https://sites.google.com/site/gson/

[3] https://docs.oracle.com/javase/7/docs/api/java/awt/geom/package-summary.html

2. Drone Control Algorithm

2.1 Overview

The section explains the mathematical principles and work process of the drone control algorithm. The implementation of some core functions (mainly implemented in PathBuilder and PathUtils) will be explained in this part.

2.2 Path Organization Process

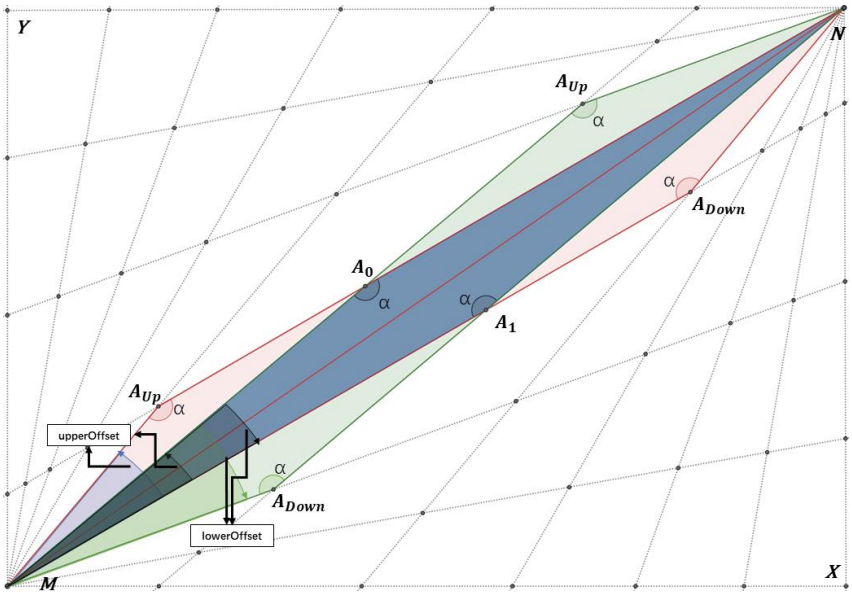
2.2.1 Preparation before organization

The system will obtain the geographic location information of the no-fly zone and landmarks through other classes. Meanwhile, the system will sort the orders by the total cost from high to low, and search for the stores needs to pass through and final destinations of each order.

2.2.2 Path organization between two locations

Path organization is mainly divided into two ways: with landmarks and without landmarks. The way of organization with landmarks can also be simply regarded as adding a new destination between the two-point path then organize the path between starting point to landmark and the landmark to ending point in the way of organizing without landmarks. However, to improve operating efficiency, the system uses a simpler path plan method for the way of using landmarks, but it will result in a slightly longer distance. It will be explained in detail below.

- **Path organization without landmarks (PathUtils.advanceOrganizePath)**
  This method takes 5 inputs which are: starting location, ending location, counterclockwise offset degree, clockwise offset degree, and no-fly zones location, and return a path that is safe for fly.



Here is a schematic diagram of planning a path according to the road network. The angle of each grid line is a multiple of 10. We assume the starting point is **M**, ending point is **N**. And we assume the coordinate of **M** and **N** is (x_M, y_M) and (x_N, y_N). In this case, the angle and distance between **M** and **N** is

$$\angle NMX = \tan^{-1}\left(\frac{y_N - y_M}{x_N - x_M}\right);\ MN = \sqrt{(y_N - y_M)^2 + (x_N - x_M)^2}$$

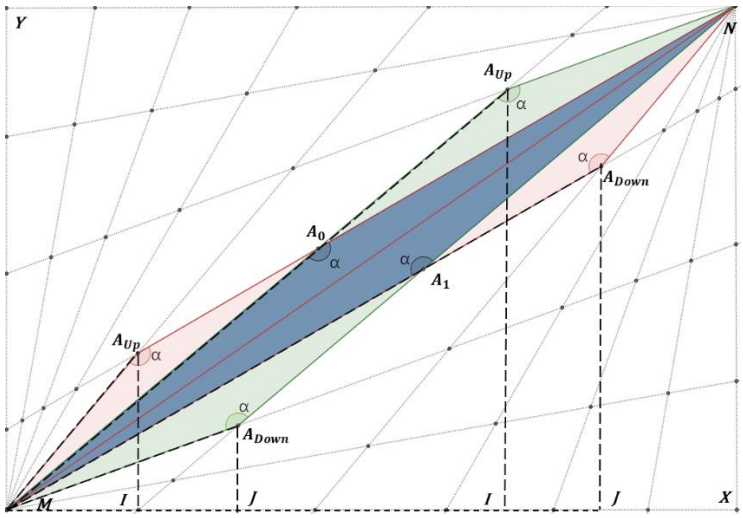
Then, the system will find $\angle A0$Then, system will find $\angle A_0MX$ and $\angle A_1MX$ which is the smallest multiple of 10 angle greater than $\angle NMX$ and the greatest multiple of 10 angle less than $\angle NMX$. Therefore,

$$\angle A_0MX - \angle A_1MX = \angle A_0MA_1 = 10°$$
$$(\angle A_0MX = 10 * z_1, \angle A_1MX = 10 * z_2 | z \in \boldsymbol{Z}\ \&\ z \in [0,36))$$

After that, the system will use the inputted counterclockwise offset degree and clockwise offset degree to determine the location of A_{Up} and A_{Down}. In the algorithm, upperOffset indicate counterclockwise offset and lowerOffset indicate clockwise offset. In the graph, counterclockwise offset is $\angle A_1MA_{Up}$ and $\angle A_0NA_{Down}$. And clockwise offset is $\angle A_0MA_{Down}$ and $\angle A_1NA_{Up}$. Therefore, by using these two parameters, the system forms a parallelogram $MA_{Up}NA_{Down}$, The upper and lower sides of the diagonal ***MN*** of the parallelogram are the organized paths. The way the system stores the path is by storing the waypoint on the path. In this example, it would be:

$$\{M,\ A_{Up}/A_{Down},\ N\}$$

In the figure, the intersection point of every grid line above MN can be an A_{Up}, and the intersection point of every grid line below MN can be an A_{Down}. The goal is to find the coordinate of A_{Up} and A_{Down} with respect to the inputted counterclockwise and clockwise offset. The system will calculate the length of $A_{Up}M$ and $A_{Down}M$, and then calculate the coordinates of A_{Up} and A_{Down} according to the trigonometric function



It is easy to see that:

$$\angle A_{Up}MX = \angle A_{Up}MA_1 + \angle A_1MX\ ; \angle A_{Down}MX = \angle A_oMX - \angle A_{Down}MA_0$$
$$\angle A_{Up}MN = \angle A_{Up}MX - \angle NMX; \angle A_{Down}MN = \angle NMX - \angle A_{Down}MX$$
$$\angle \alpha = 180 - \angle A_{Up}MX - \angle A_{Up}MN$$

Since **MN** has been calculated, Through the law of sines, system will calculate the length of $A_{Up}M$ and $A_{Down}M$

$$A_{Up}M = \frac{MN}{\sin(\alpha)} * \sin\left(\angle A_{Up}MN\right)$$
$$A_{Down}M = \frac{MN}{\sin(\alpha)} * \sin\left(\angle A_{Down}MN\right)$$

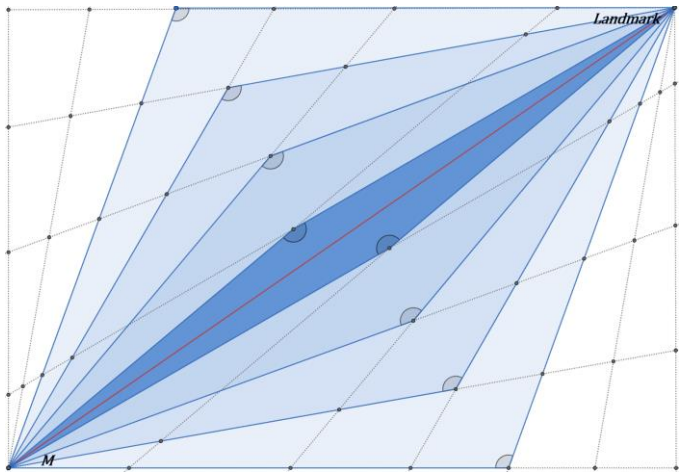
Therefore, the coordinate of A_{Up} and A_{Down} is:

$$A_{Up}: \left(x_M + A_{Up}M * \cos\left(\angle A_{Up}MX\right), y_M + A_{Up}M * \sin\left(\angle A_{Up}MX\right)\right)$$
$$A_{Down}: \left(x_N + A_{Down}M * \cos\left(\angle A_{Down}MX\right), y_N + A_{Down}M * \sin\left(\angle A_{Down}MX\right)\right)$$

- **Path organization with landmarks (PathUtils.naiveOrganizePath)**
  For situations that need to pass landmarks, because the map has more than one landmarks, the system uses another similar but faster algorithm to find the way.
  When using the landmark, the system organizes the path from the starting point to landmark, then from landmark to ending point. If the system uses the organize algorithm described before, since that algorithm will calculate over all possible paths, the number of organize attempts is

$$2 * number\ of\ landmarks * 18 * 18$$

Because the system needs to find all possible paths for each step and find the shortest one, increasing the speed of this algorithm can greatly increase the overall system operating speed.



Instead of using counterclockwise offset degree and clockwise offset degree to determine the location of A_{Up} and A_{Down}, this algorithm set two parameters to with same value from 0 to 180 and calculate two points with the same offset value. Although the distance may be slightly extended, it increases the operating speed of the system.

2.2.3 No-Fly Zone

After finish organize the path between two points, the system will check whether the path has the intersection point with the edge of the no-fly zone, whether the path is out of the fly range, and whether the waypoint of the path is too close to the no-fly zone. If any of the condition above is true, this path will be dropped.

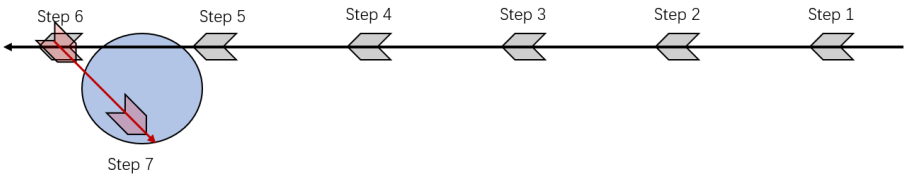
2.2.4 Organize All Path

The system will organize the path of the waypoints adjacent in every order in the order list. While organize the nearby waypoints, system will try every possible legal combination of counterclockwise offset and clockwise offset using PathUtils.advanceOrganizePath. Meanwhile, system will also try to find all possible paths with landmarks. Also, all path that belongs to the condition in 2.2.3 will be removed. Then, the path with the shortest distance will be selected for the final path for these two points. If the system fails to find a path in the end, it will keep generate random landmarks and try to find the path through it.

2.2.5 Writing into Database

None of the previous mentioned system planning considers the condition that one step is 0.00015 length. Therefore, when writing into the database, the system will put the drone at the starting point (Appleton Tower) and start to move along the organized path step by step until it get close to (distance within 0.00015) the next destination. Also, system will write into database at the same time. In addition, if the system found that there is not enough power to return to Appleton Tower after sending the next order, it will order the drone to return immediately after finishing the current order.

Because under the limit of 0.00015 length of each step, the actual position will deviate from the previously planned position, so this situation may occur.



Because of the position deviation, the plan to approach the destination area in the fifth step was invalidated. System can detect this situation and change the angle in step 6 to move into the destination area.

2.3 Example of Organized Path Figure

- The first example is from 2022-08-23, in this day, drone able to deliver all orders and return to Appleton Tower.

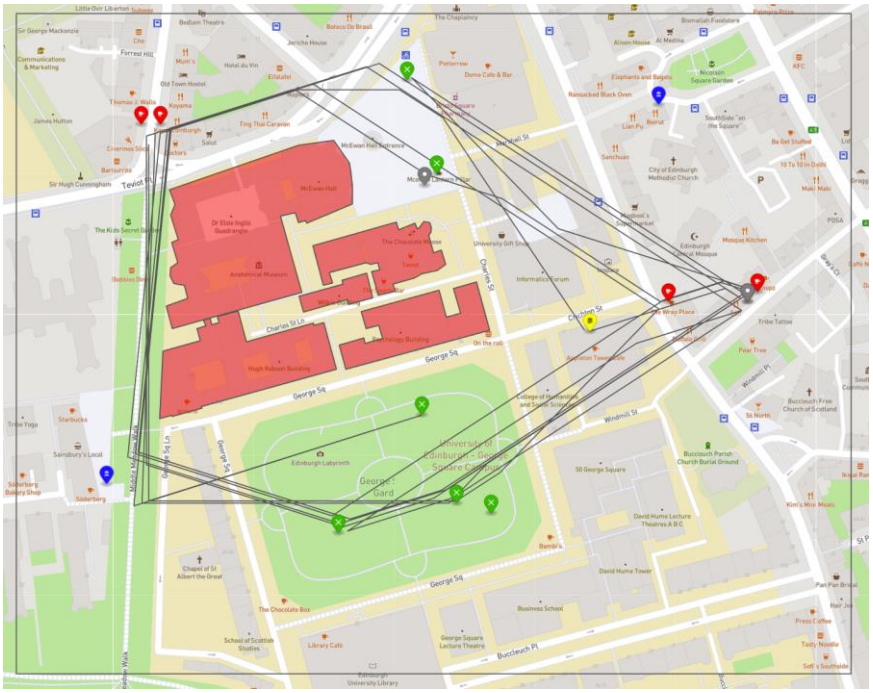


Figure 3: Flight Path 2022-08-23

- The second example is 2023-12-28, in this day, drone is unable to deliver all orders. As can be seen from the picture, although the power is not enough to deliver all the orders, the drone can still safely return to Appleton Tower.

Figure 4: Flight Path 2023-12-28