Project Task

Plan an optimal drone operation for surveying a sugar beet field

The specific requirements are:

1. The drone must survey multiple fields, whose coordinates are provided in the attached file. Assume that each field is a square with a side length of 200 meters.

2. Determine the optimal location for the drone dock, preferably near a farm, and compute the optimal flight route.

3. Consider the following constraints:
- The drone flies at 15 meters per second between fields.
- Each field takes 10 minutes to survey.
- The drone has a maximum flight time of 40 minutes before it needs to return to the dock.
- The drone requires 35 minutes of charging before it can resume flights.

4. Expected Implementation
- Implement a C# program to compute the optimal dock location and flight path.
- The program should read the field coordinates from the input file.
- Use an appropriate algorithm to minimize total operation time while ensuring all constraints are met.
- Provide a visualization to illustrate the flight path and dock location.
- Handle edge cases such as disconnected fields or infeasible operations.

5. The input file containing the coordinates is expected to be in Excel.
Both latitude and longitude are expected to be in the first column.
The type of location, i.e. field or building, is expected to be in the second column.
Example:

First Column	Second Column
49.15868902252248, 9.111073073485683	Field
49.1582580129513, 9.113612777241652	Building

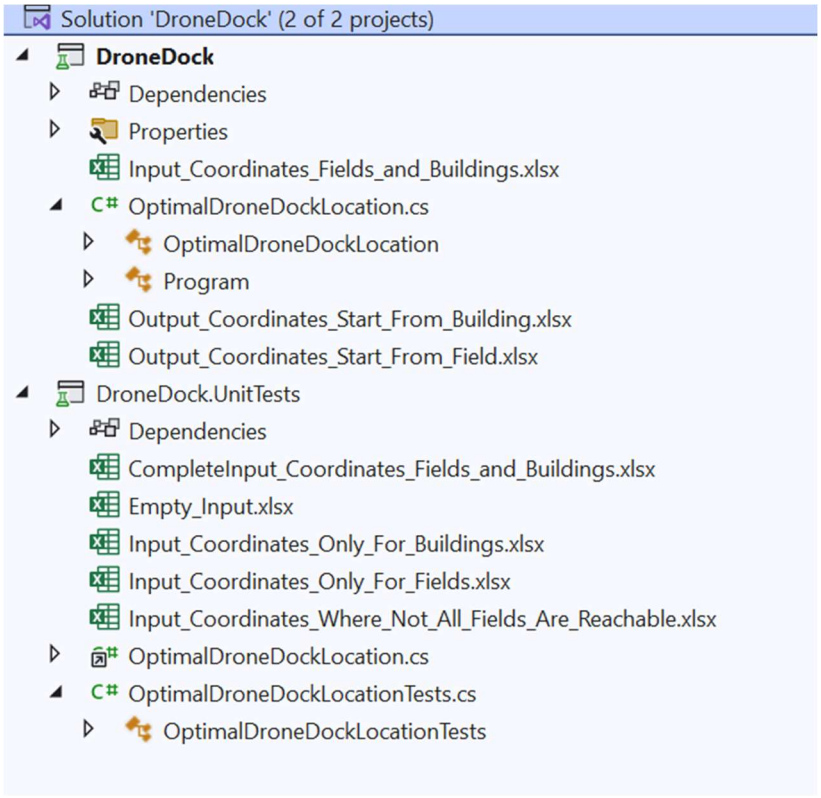
etc.

6. It is assumed that the start of the survey for each field is from the center (the coordinates in the input file) and that it is completed also at the center. The distance between each field, respectively the flight time, is strictly calculated in the basis of these centers. Each field must be surveyed at once, during one flight, so there are no partial surveys of a field that can be completed during two or more flights.

Project Implementation

Programming language: C#
Framework: .NET 9.0
Testing: xUnit
IDE: Microsoft Visual Studio 2022

Overview:




```
namespace DroneDock, public class OptimalDroneDockLocation
```

Contains the logic of the application. Each method and field is appropriately named, and where necessary, accompanied by comments. All that is needed to run the application is to supply a valid input file (as described in the project task) to the constructor.

Then the application will run the relevant methods and generate an output in the form of a list containing two objects `SurveyStatistics`: the first one has data for a drone dock on a field and the second one has data for a drone dock on a building.

In addition to that, for the purposes of paths and drone dock visualization, output files in Excel are generated for each type of drone dock: Output_Coordinates_Start_From_Field' and 'Output_Coordinates_Start_From_Building'.

The first column of the output file contains latitude coordinates.
The second column of the output file contains longitude coordinates.

Since the visualization is done at https://www.gpsvisualizer.com/map?output_google to visualize all flights per survey with different colors (when visualizing all flights per survey at once) the row preceding the coordinates for each flight has 'latitude' as title in the first column and 'longitude' as title in second column. Example:

First Column Second Column

latitude longitude
49.15742599 9.116101442
49.15621042 9.116235216
… the rest of the coordinates for first flight of current survey …
latitude longitude
… coordinates for second flight of current survey …
etc.

`public class Program`

Contains the method `static void Main()` and a quick initialization of `class OptimalDroneDockLocation`, ready to run.

`namespace DroneDock.UnitTests, public class OptimalDroneDockLocationTests`

Contains the unit tests, implemented through the xUnit framework. Here again, each method and field is appropriately named, and where necessary, accompanied by comments.

Concluding Notes

As mentioned, the visualization of the flight paths and drone dock location is done through https://www.gpsvisualizer.com/map?output_google and if other visualization tools are to be applied, the format of the output files might need modification.

A sample of the output, both in terms of visualization and text output (saved screenshots) is provided for the given example of input file.

The paths for the input and output files in all classes must be adjusted for the local system where the application will be run.