# Design Notes for the E-Vtol Simulation Project

Kodiak North

## 10/09/2024 – Initial list of notes/tasks/questions to guide the design

* First and foremost, I’ll need to configure VS Code to compile and run C++ executables.
* What are some unit tests I can incorporate into the code?
  + The first idea that comes to mind is a test to ensure all 5 vehicle types end up in the random list of 20 every time it is created.
  + I can also design a test to remove battery life from the equation and ensure each vehicle type will fly the correct distance given 1 hour of simulated time.
  + Similarly, I can test that the batteries drain in the right amount of time.
* I should obviously create a parent class to define an e-vtol vehicle with the provided characteristics, and then child classes for each specific vehicle type will handle the intricacies between different velocities, number of passengers, etc.
* How will I handle the possible generation of a fault?
  + A 25% chance of a fault per hour does not necessarily mean that a fault will occur every 4 hours!
* So the simulation does not actually take 3 hours, 1 loop step can be considered 1 second, or even 1 minute of elapsed time.
* How will the equation for energy consumption work?
  + Say cruise speed is 100 miles/hour, battery capacity is 50 kWh, and energy use at cruise is 2 kWh/mile…
  + How long the battery will last flying at cruise speed?
    - 50 kWh / (2 kWh/mile) = 25 miles… => 25 miles / (100 miles/hour) = 0.25 hours… => 0.25 hours \* (60 min/hour) = 15 min.
  + So the equation is:
    - ((Battery capacity [kWh] / energy use at cruise [kWh/mile]) / cruise speed [miles/hour]) \* 60 [min/hour] = duration in minutes
    - Will have to incorporate the loop step into this! Or perhaps this value could be stored for each vehicle type. As the system loops, it increments a counter. Once the counter equals the calculated duration, the battery is dead.
* Design a state machine for each vehicle. Upon initial thoughts, the states could be:
  + Idle, flying, charging, or waiting to charge
* Will need create a queue to hold vehicles that are waiting to charge, and also track chargers that are available or not.
* How will I assign a vehicle to a charger? It can probable be as simple as chargers 1, 2, and 3 are in the queue of available chargers. When a vehicle needs a charger, it is dequeued and its ID populates the ‘current charger’ member variable for that vehicle. Once charging is complete, the charger’s ID is enqueued back into the queue of available chargers.
  + If the queue of available chargers is empty, that means there are no available chargers and a vehicle will have to be placed in line.
* As for recording metrics:
  + If I track the flight time, I think I’ll know each vehicle’s distance travelled based on the cruise speed.
  + I can record vehicle charge time and store it in a member variable.
  + Store number of faults in a member variable.
  + Total passenger miles can likely also be calculated at the end of the simulation based on flight time.

Okay! So this is a lot of information spewed out. It’s time to organize it into a list of priorities, and concretely define the data structures that I’ll use.

## 10/10/2024 – The Beginnings of a Plan

1. ~~Install C++ compiler and build a simple hello world application to make sure the toolchain is squared away.~~
2. Define the data structures for the E-Vtol parent class, and one specific vehicle type child class. Let’s just start with the Alpha Company.
3. Create one instance of an Alpha aircraft and perform some tests!
   1. Will need the main loop to step through time. Can initially loop the corresponding number of times for 1 simulation hour. This will be 60 times if the desired loop step is 1 minute.
   2. Start in the idle state, move to the flying state, fly until the battery is dead, and then return to the idle state.
   3. Record metrics during the flight, and verify that everything looks correct at the end! At this point in time, we’re not worrying about charging or faults. Data such as flight time, distance traveled, and total passenger miles should be easy to test now.
      1. The expected flight time for the Alpha aircraft is 100 minutes using the equation calculated yesterday. If the loop is kept at 1 hour for this first test, the Alpha plane should still be in the flying state at the end of the loop, with 60 minutes of recorded flight time. This will be an excellent test to perform!
      2. If all passes, then loop for 2 hours, and expect the state to be idle at the end with 100 minutes of flight time.
4. Now we have an aircraft that can fly until its battery dies, and then stop flying. It would be a good time to incorporate one charger for our one aircraft to recharge with, and record its total time spent charging
   1. With only one charger, we don’t have to worry about creating the queue right away.
   2. When the battery dies, the Alpha plane can be placed directly into the charging state.
   3. It will take 36 minutes to recharge its battery.
   4. For our 1-hour test, there should be no recharging.
   5. With the 2-hour test, the total time spent charging should be 20 minutes, and the final state will be charging at the end of the loop.
   6. It’d be good to now perform a 3-hour test. What will we expect to see then?
5. With one vehicle flying, recharging, and flying again; we’re at a very good starting point. Now will be a good time to incorporate another vehicle and another charger. How will we have to modify the code to implement this?