



Dronology Ground Station

Controlling UAVs through WebSockets

Joshua Huseman

<https://dronology.info/>



Core Dronology Controller/User Interface

- Java-based core controller dispatches flight routes
- Vaadin (Java web server) user interface handles planning and visualizing flights

The screenshot displays a web-based interface for drone management. At the top, there are tabs for "Active Flights" and "Flight Routes". The main interface is divided into several sections:

- 1 Active UAVs:** A list of active drones. The first entry is "TollRoad-SurveillanceUAV" with a status of "FLYING" and a battery life of 14.73%. It shows real-time coordinates: Latitude: 41.716627, Longitude: -86.22966, Altitude: 25.0 meters, and Ground Speed: 0.0 m/s. Below this information are controls: a "Hover in Place" toggle (currently OFF), a "Return to Home" button, and an "Assign New Route" button.
- Following UAV(s):** A section titled "Following UAV(s): TollRoad-SurveillanceUAV" with a "Stop Following" button. It features a map showing a flight route (orange line) starting from a cluster of black dots near Cleveland Road and Indiana Toll Road, and ending at a drone icon near Indiana Toll Road and Juniper Road.
- Map View Operations:** Two buttons: "Follow Selected UAVs on Map" and "View All UAVs on Map".
- Emergency Operations:** Two buttons: "All UAVs Hover in Place" and "All UAVs Return to Home".



Ground Control Station

- Communication bridge between core controller and UAVs
- Relays status messages from UAVs to Dronology Core
- Forwards commands from Dronology to the UAVs
- Pulls RTK GPS correction data from Raspberry Pi server
- Written in Python:
 - DroneKit (<http://python.dronekit.io/>) for controlling UAVs
 - Socket for communicating with Dronology Core (and RTK server)
- Simple JSON-based communication protocol



JSON Communication Protocol

Each message is a JSON object, with a carriage return (\r) character to signal the end of a message.

```
{
  "type": "handshake",
  "uavid": "PHYS_0",
  "sendtimestamp": 1512454376145,
  "data": {
    "home": {
      "x": 41.698184,
      "y": -86.233975,
      "z": 0.0
    },
    "safetycase": {}
  }
}
```

```
{
  "type": "state",
  "uavid": "PHYS_0",
  "sendtimestamp": 1512454389145,
  "data": {
    "location": {"x":0.0,"y":0.0,"z":0.0},
    "attitude": {"x":0.0,"y":0.0,"z":0.0},
    "velocity": {"x":0.0,"y":0.0,"z":0.0},
    "batterystatus": {
      "current": 0.1,
      "voltage": 12.4,
      "level": 100.0
    },
    "status": "ACTIVE",
    "armable": true,
    "groundspeed": 0.0,
    "armed": true,
    "mode": "GUIDED"
  }
}
```

```
{
  "command": "takeoff",
  "uavid": "PHYS_0",
  "msgid": 1,
  "sendtimestamp": 1512454393145,
  "data": {
    "altitude": 100
  }
}
{
  "command": "gotoLocation",
  "uavid": "PHYS_0",
  "msgid": 2,
  "sendtimestamp": 1512454399145,
  "data": {
    "x": 10,
    "y": -10,
    "z": 100
  }
}
```

```
{"command":"gotoLocation","uavid":"PHYS_0","msgid":2,"sendtimestamp":1512454399145,"data":{"x":10,"y":-10,"z":100}}
```



RTK Server

- Broadcasts satellite correction data from a stationary GPS receiver to moving GPS on UAVs
 - Corrects position based off of atmospheric disturbance and delay in the GPS signal
 - https://en.wikipedia.org/wiki/Real_Time_Kinematic
- Reads GPS data using u-blox8 protocol (<https://www.u-blox.com/en>)
- Forwards data over socket to Ground Control Station
- Small python server to run on a Raspberry Pi (<https://www.raspberrypi.org/>)

