

Informatics Large Practical: Specification

Stephen Gilmore and Paul Jackson

School of Informatics

Monday, 21st September 2020

Specification

- The task for the Informatics Large Practical is to **program an autonomous drone** which will collect readings from **air quality sensors** distributed around an **urban geographical area** as part of a (fictitious) research project to analyse urban air quality.
- The research project which is collecting and analysing the air quality readings is beginning with a **small-scale feasibility study** around the University of Edinburgh's Central Area.
- If this feasibility study is successful, the researchers will apply for funding to conduct their research on a much larger scale.

Your task

- Your task is to program the drone to fly around and **collect the sensor readings** of air quality and to produce **scientific visualisations of the data** which can be shown to funding councils in order to convince them to fund the large-scale version of the project.
- You should think that your implementation of the drone control software will be passed on to a team of research assistants and PhD students who will maintain and develop it.
- The **clarity and readability of your code is important**; you need to produce code which can be read and understood by others.

An air quality map



Drones and sensors

- The drone is fitted with a receiver which can download readings from sensors over-the-air, provided that it is within 0.0002 degrees of the air quality sensor.
- Degrees are used as the measure of distance throughout the project instead of metres or kilometres to avoid unnecessary conversions between one unit of measurement and another.
- The latitude and longitude of a sensor are expressed in degrees, so we stay with this unit of measurement throughout all our calculations.

Calculations: determining distances

- As a convenient simplification, locations expressed using latitude and longitude are treated as though they were **points on a plane**, not **points on the surface of a sphere**.
- This simplification allows us to use **Pythagorean distance** as the measure of the distance between points.
- That is, the distance between (x_1, y_1) and (x_2, y_2) is just

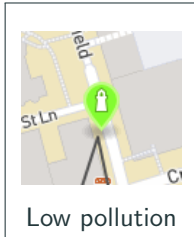
$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

The flight of the drone each day

- There are 99 sensors, of which 33 need to be read each day.
- The sensors are battery-powered, and when a receiver takes a reading from a sensor it has two components:
 - the *reading*: this is a character string which should represent a real value between 0.0 (no air pollution) and 256.0 (maximum possible reading); and
 - the *battery*: this is a real value expressing the percentage battery charge between 0.0 and 100.0.

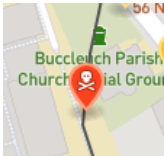
The flight of the drone each day

- There are 99 sensors, of which 33 need to be read each day.
- The sensors are battery-powered, and when a receiver takes a reading from a sensor it has two components:
 - the *reading*: this is a character string which should represent a real value between 0.0 (no air pollution) and 256.0 (maximum possible reading); and
 - the *battery*: this is a real value expressing the percentage battery charge between 0.0 and 100.0.



The flight of the drone each day

- There are 99 sensors, of which 33 need to be read each day.
- The sensors are battery-powered, and when a receiver takes a reading from a sensor it has two components:
 - the *reading*: this is a character string which should represent a real value between 0.0 (no air pollution) and 256.0 (maximum possible reading); and
 - the *battery*: this is a real value expressing the percentage battery charge between 0.0 and 100.0.



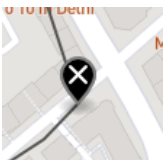
High pollution

Detecting, and reporting, a low battery

- If the battery has less than 10% charge then the sensor reading **cannot be considered to be trustworthy** because the sensor hardware gives false readings at low power levels.
- The sensor reading **at less than 10% charge** might be “**null**” or “**NaN**” (Not a Number).
- Even if the reading looks like a real number **it should be discarded**, and the sensor reported as needing a new battery.

Detecting, and reporting, a low battery

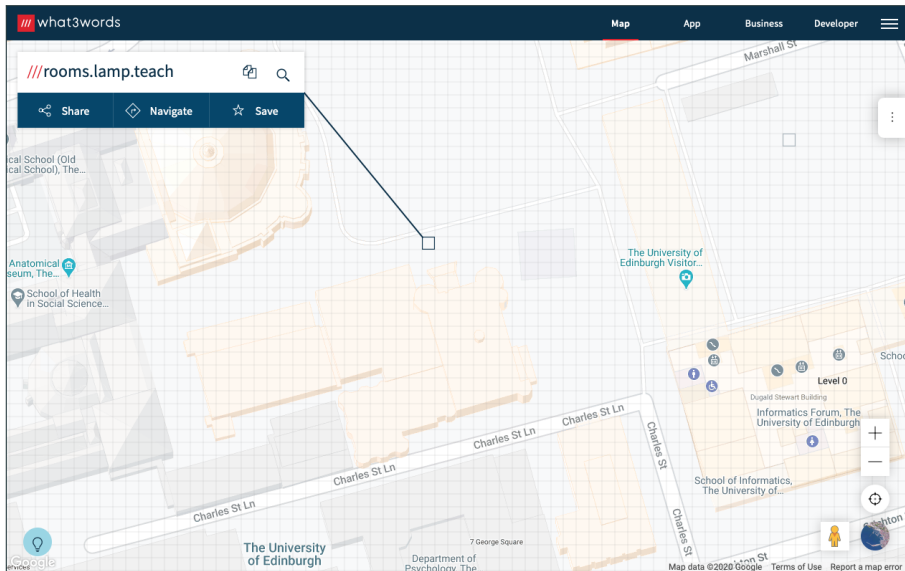
- If the battery has less than 10% charge then the sensor reading **cannot be considered to be trustworthy** because the sensor hardware gives false readings at low power levels.
- The sensor reading **at less than 10% charge** might be “**null**” or “**NaN**” (Not a Number).
- Even if the reading looks like a real number **it should be discarded**, and the sensor reported as needing a new battery.



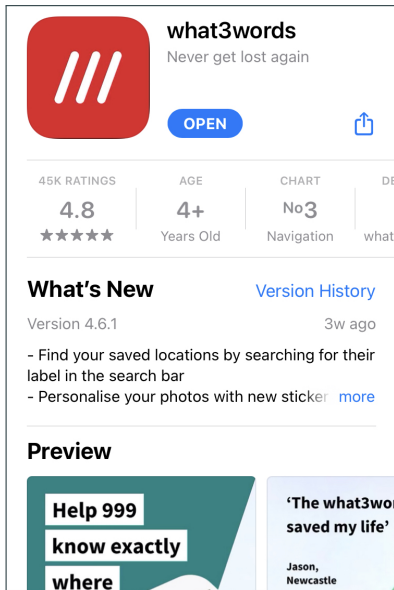
Low battery

Replacing a low battery

- Replacing the battery in a sensor is a **manual process** whereby a researcher from the project is sent to the location of a faulty sensor to swap out the drained battery for a fully-charged one.
- A novel **location addressing system** is used to help locate the sensor where the battery needs to be replaced: the *What3Words* encoding.
- The What3Words system maps the earth's surface using 3m^2 tiles, **each with a unique three word address**.

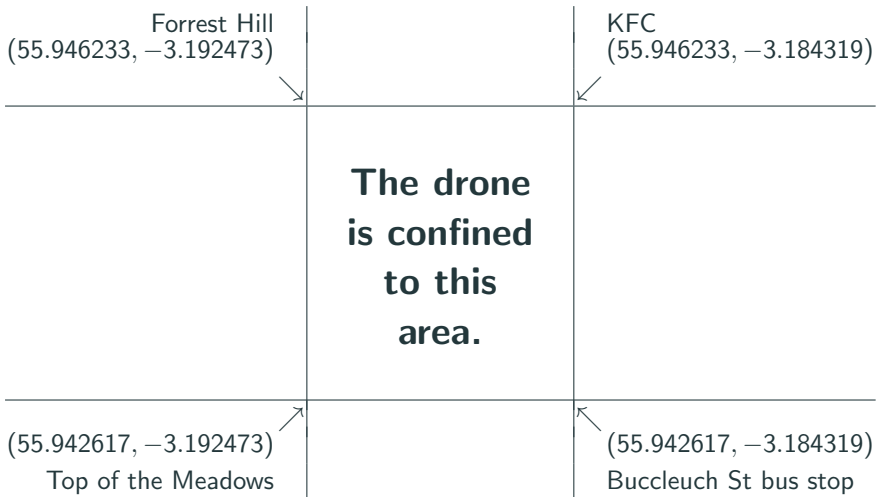


What3Words mobile phone app



The research assistants on the project will be able to use the mobile phone app to locate the sensors using addresses like "sports.topic.clocks"

Drone confinement area



Data used by the drone

- In order to help with the development of the software for the drone, the researchers on the project have made a **web server** with synthetic data which represents the **sensor and battery level data** that a drone would be able to read from a sensor on a given day if it was **in range** (i.e. within 0.0002 degrees) of the air quality sensor.
- The web server has three top-level folders, **maps**, **words**, and **buildings**.

Data used by the drone: **maps**

- The **maps** folder contains files containing **lists of sensors to be visited**, one for each day in the years 2020 and 2021.
- Within each of these folders is a JSON (JavaScript Object Notation) format file named **air-quality-data.json** which specifies the list of sensors which are to be visited on that day, together with the **air quality reading** and the **battery level** of that sensor.

The file `maps/2021/06/15/air-quality-data.json`

```
[  
  {  
    "location": "slips.mass.baking",  
    "battery": 94.53979,  
    "reading": "141.81"  
  },  
  {  
    "location": "sports.topic.clocks",  
    "battery": 1.0801673,  
    "reading": "null"  
  },  
  :  
]
```

What3Words address

battery OK: ~95%

sensor reading: 141.81

What3Words address

low battery! : ~1%

Data used by the drone: words

- The **words** folder contains JSON files with the What3Words information corresponding to a What3Words address.
- The details for a What3Words address **"first.second.third"** will be stored in the file **words/first/second/third/details.json**.
- The most important field in this record for our purposes is the **coordinates** field which allows us to associate a What3Words address with a specific longitude (**"lng"**) and latitude (**"lat"**).

The file `words/slips/mass/baking/details.json` (1/2)

```
{  
  "country" : "GB",  
  "square" : {  
    "southwest" : {  
      "lng" : -3.187428,  
      "lat" : 55.945936  
    },  
    "northeast" : {  
      "lng" : -3.18738,  
      "lat" : 55.945963  
    }  
  },  
  :  
  :
```

The What3Words square

South-west corner

North-east corner

The file `words/slips/mass/baking/details.json` (2/2)

```
⋮  
"nearestPlace" : "Edinburgh",  
"coordinates" : {  
  "lng" : -3.187404,  
  "lat" : 55.945949  
},  
"words" : "slips.mass.baking",  
"language" : "en",  
"map" : "https://w3w.co/slips.mass.baking"  
}
```

Location of the sensor

Sensor longitude

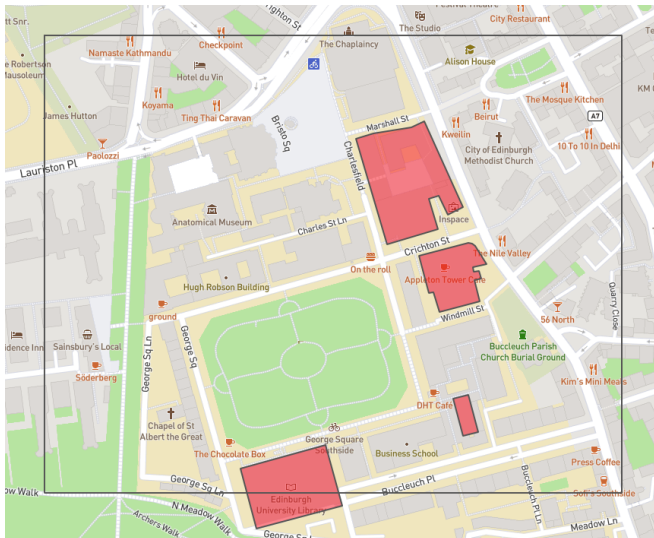
Sensor latitude

Metadata

Data used by the drone: **buildings**

- Within the drone confinement area are **four regions where the drone is not allowed to fly**.
- These are known as the ***No-Fly Zones*** for the drone.
- The details of the No-Fly Zones are given in the file **`buildings/no-fly-zones.geojson`**.
- This file is in **Geo-JSON format**, which is a standard way of encoding geographic data structures

The No-Fly Zones and the containment area



The flight path of the drone and its life-cycle (1/2)

1. The drone flight path has at most 150 moves, each of which is a straight line of length 0.0003 degrees;
2. the drone cannot fly in an arbitrary direction: it can only be sent in a direction which is a multiple of ten degrees where, 0 = East, 90 = North, 180 = West, and 270 = South, with the other multiples of ten between 0 and 350 representing the obvious directions between these four major compass points.

The flight path of the drone and its life-cycle (2/2)











3. As near as possible, the drone flight path should be a closed loop, where the drone ends up close to where it started from (where close to x means less than 0.0003 degrees from x); and
4. the drone life-cycle has a pattern which iterates (i) making a move; and (ii) taking one sensor reading (if in range).

Plotting the information returned by the drone











In order to help communicate the behaviour of the drone to others, the drone flight path is to be plotted to a **Geo-JSON map**. The markers on the map have four properties:

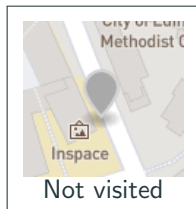
- **location** — the What3Words location string;
- **rgb-string** — the HTML colour of this marker represented as a hexadecimal Red-Green-Blue string;
- **marker-color** — identical to **rgb-string**, but will be rendered by <http://geojson.io>; and
- **marker-symbol** — a symbol from the Maki icon set (<https://labs.mapbox.com/maki-icons/>).

The **colour mapping** for markers on the Geo-JSON map

Range	RGB string	Marker colour	Marker symbol
$0 \leq x < 32$	#00ff00		lighthouse
$32 \leq x < 64$	#40ff00		lighthouse
$64 \leq x < 96$	#80ff00		lighthouse
$96 \leq x < 128$	#c0ff00		lighthouse
$128 \leq x < 160$	#ffc000		danger
$160 \leq x < 192$	#ff8000		danger
$192 \leq x < 224$	#ff4000		danger
$224 \leq x < 256$	#ff0000		danger
low battery	#000000		cross
not visited	#aaaaaa		no symbol

The **colour mapping** for markers on the Geo-JSON map


Range	RGB string	Marker colour	Marker symbol
$0 \leq x < 32$	#00ff00		lighthouse
$32 \leq x < 64$	#40ff00		lighthouse
$64 \leq x < 96$	#80ff00		lighthouse
$96 \leq x < 128$	#c0ff00		lighthouse
$128 \leq x < 160$	#ffc000		danger
$160 \leq x < 192$	#ff8000		danger
$192 \leq x < 224$	#ff4000		danger
$224 \leq x < 256$	#ff0000		danger
low battery	#000000		cross
not visited	#aaaaaa		no symbol



An air quality map (**note:** not a closed loop)




Investigating the markers on the map using <http://geojson.io>


marker-size	medium
location	slips.mass.baking
rgb-string	#ffc000
marker-color	
marker-symbol	danger

[+ Add row](#) ☒ Show style properties

Properties Info

[Save](#) [Cancel](#) [Delete feature](#)




marker-size	medium
location	sports.topic.clocks
rgb-string	#000000
marker-color	
marker-symbol	cross

[+ Add row](#) ☒ Show style properties

Properties Info

[Save](#) [Cancel](#) [Delete feature](#)



Investigating the markers on the map using <http://geojson.io>

marker-size	medium
location	sculpture.shot.melon
rgb-string	#40ff00
marker-color	■
marker-symbol	lighthouse

[+ Add row](#) ☒ Show style properties

Properties Info

[Save](#) [Cancel](#) [Delete feature](#)

marker-size	medium
location	hurt.green.filier
rgb-string	#80ff00
marker-color	■
marker-symbol	lighthouse

[+ Add row](#) ☒ Show style properties

Properties Info

[Save](#) [Cancel](#) [Delete feature](#)

Programming language: Java

- The programming language to be used for your software is **Java**. The researchers on the project have chosen **Java version 11** as the version of Java which will be used throughout the project.
- This version of Java has been selected because it provides **local variable type inference** and other helpful features and is the most recent **LTS (Long Term Support)** version of Java which is available.
- You should ensure that the code which you submit can be compiled and run on a Java 11 installation.

Using third-party software and libraries

- This practical exercise allows you to use **free software**, but not commercial software which you must pay for or license.
- One free software development kit (SDK) which you should find useful is the **Mapbox Java SDK** which provides classes and methods for **parsing and generating Geo-JSON maps**.

Informatics Large Practical

Stephen Gilmore and Paul Jackson
School of Informatics, University of Edinburgh

Document version 1.0.0

First issued on: September 21, 2020

About

The Informatics Large Practical is a 20 point Level 9 course which is available for Year 3 undergraduate students on Informatics degrees. It is not available to visiting undergraduate students or students in Year 4 or Year 5 of their undergraduate studies. It is not available to postgraduate students. Year 4, Year 5 and postgraduate students have other practical courses which are provided for them.

Scope

The Informatics Large Practical is an individual practical exercise which consists of one large design and implementation project, with two coursework submissions. Coursework 1 involves creating a new project and implementing one important component of the project. Coursework 2 is the implementation of the entire project together with a report on the implementation.