Basic design for drone detection system

By Pouria Akbari Mistani, November 2022

Approved

This document presents the design for the basic version of the drone detection system. We use this design as our guide for development. The document will be updated throughout the project lifetime.

Table of contents

Introduction

We want to design a simple geo-awareness service which allows pilots to specify a 4-dimensional spacetime in the airspace and get notifications (e.g. email, text, etc.) when a drone enters the airspace. We want to build a simulated digital twin of this service to visualize the concept of operations which may eventually be applied to real world scenarios. This document is concerned with the basic design of the system satisfying the requirements defined in a separate document accompanied by this design. A more advanced version of the design and requirements will be provided in separate documents.

System overview

The basic version of the geo-awareness service can be divided into a few components as shown in Figure 1. The drone detection system is responsible for detecting any drones entering the airspace of interest and sending detections to the notification server. The notification server is used to update the user interface and send emails to a list of known email addresses. The end user will be able to use their devices (e.g., phones, laptops, etc.) to check their notifications.

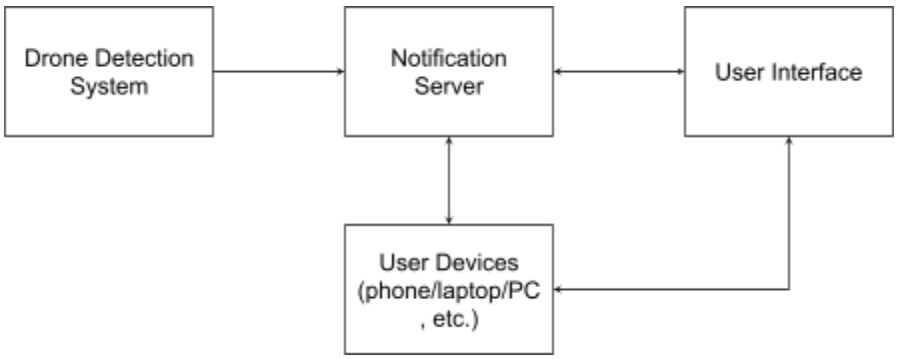

Figure 1. Main components of the service.

Design constraints

In this section, we provide a basic list of constraints to implement the proposed design.

1. The user interface should run on a typical laptop computer or PC with basic CPU and memory.
2. The system should run on common operating systems such as Windows, MacOS and Linux.
3. Notification server should run on a typical laptop computer or PC with basic CPU and memory.
4. The user interface is a single user application with no authentication
5. The user interface doesn’t require a database and all application data will be lost after closing it.

6. The system should run in a controlled development environment such as in a shell terminal by running a few basic commands provided to the user. Alternatively, if possible, an executable can be provided for the user to run on their device.

Future contingencies

This section presents contingency plans and risk mitigation plans for the proposed solution if we decide to extend this solution to an advanced version.

1. Losing user data after closing the application is a risk and could be mitigated in the future by adding a local/remote database to the system.
2. Authentication service can be added in the future to protect user information.
3. User management service can be added in the future to support multiple users at the same time.

Relevant documentation

1. This design is accompanied by a requirement document.

Glossary

If your design document has many terms and abbreviations, please add a list of them in this section.

1. **SDK**: Software development kit.
2. **API**: Application programming interface.

System architecture

System software

Python is chosen as the main programming language for this design for the following reasons:

1. Designer is familiar with the language and its features
2. There are a lot of open source libraries in Python we can use to expedite development of the first proof of concept of our proposed solution.
3. It is one of the popular programming languages that most developers are comfortable with.

Drone detection system

We assume the drone detection system is provided to developers and its design is outside the scope of this document assuming it comes with appropriate SDKs and APIs for developers. The developers are able to interact with the SDK/API to get the detections from the system and based on the detections we decide what to do next in our notification server and user interface.

Notification server

The simplest solution is to use an existing email server such as Gmail. The following diagram shows how to use Gmail for sending email notifications to our users.
Python provides native support for sending emails using existing standard email servers such as Gmail, Yahoo, Microsoft, etc. We take advantage of these features and use it to send email notifications to an email address provided by the user. We choose Gmail as our email server and for that we have to create a user account on Gmail so our Python email service can access and authenticate itself with Gmail server. Figure 2 presents the components of the notification server.

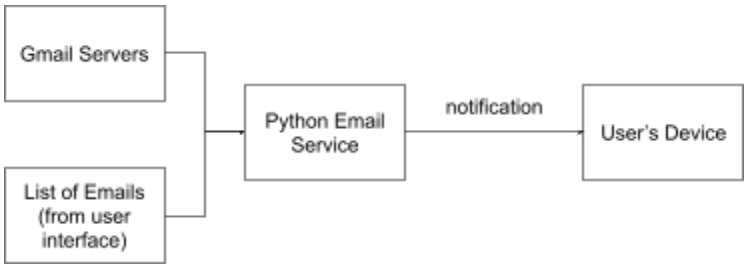

Figure 2. Components of the notification server.

We develop the Python email service shown in Figure 2 for sending email notifications to users. For simplicity, we provide the Gmail user account credentials here in this design document. Note that for sending emails from our Python scripts, we need to create a specific "app generated password" for the Gmail account. Instructions for creating the "app generated password" for Gmail can be found in the following link: https://support.google.com/mail/answer/185833?hl=en

Username	utm.geoawareness@gmail.com
Password	A3fancyutm
App generated password	fosgnogvrbeszhnf
Login address	mail.google.com

The Python email service uses SMTP protocol for sending emails. The following are the SMTP configuration for the Gmail server.

Port number	465
SMTP server address	smtp.gmail.com

User interface

For the user interface, we choose VPython library for the following reasons:

1. Ease of use for creating 3D worlds and objects which we believe is useful for creating a basic digital twin of our environment
2. Has features for adding common user interface components such as input boxes, buttons, sliders and text boxes.
3. Support for common geometry operations required to build the digital twin
4. It can run on a typical computer with minimal compute resources with minimal dependencies.
5. Support for 3D animation and use on touch screens
6. Open source commercial-friendly licenses

The user interface can be broken down into two components. One component is responsible for handling the user data and based on the data update the visual components of the user interface. Another component is responsible for rendering the interface including the text, buttons, 3D world, etc. To run the user interface application, a few instructions will be provided in the repository containing the source code.

System hardware

In this section all the hardware components involved in the design are described.

Drone detection system

The hardware for the drone detection system is outside the scope of this document. There are commercially available services such as Aerial Armor we can use in this proof of concept. This design is concerned with the software development aspects of the project. As mentioned before in this document, we assume SDKs and APIs for interacting with the drone detection system are provided.

Common systems are based on Radio Frequency (RF) detection that can triangulate multiple drones and their controllers accurately. RF systems don't require a license to operate and support tracking of drones. However, RF systems cannot detect autonomous drones (non-piloted) and their efficacy is reduced in areas with a mix of multiple radio frequencies. These systems also have a limited range specially when physical obstacles are present. Considering their shortcomings, they are still capable of satisfying the requirements of this project. More advanced detection systems include radars and optical/thermal sensors (cameras).

User devices

For simplicity, we assume everything runs on a basic computer with enough CPU (e.g., a 1GHz+ quad core CPU) and memory (e.g., at least 8 GB). Also, we assume the users have an email address and they know how to use their email service.

Interfaces

The current design requires an interface between the notification service and the drone detection system. In this design, the notification service subscribes to a signal/event from the drone detection system. We use the following interfaces:

1. The notification service uses the SDK/API provided by the drone detection system
2. The email service uses APIs provided by Google to send email notifications using Gmail service.

Integration

Currently there are no integration concerns as all the programs will be running from a single standalone application. In the future, if new components are added to the design, integration concerns will be documented.

Discussion and recommendations

Current design is a very basic solution that satisfies the requirements. Since the requirements are not very accurate and are not verified by users of the system, we recommend implementation of the proposed minimalistic design in this document. After the first proof of concept is implemented, market research and user interviews can be done by the product development and business development teams to verify and/or change the requirements for the next phase of the project.

Future work

The current requirements are vague and highly speculative which makes the future work almost impossible to predict. However there are certain high level next steps we can define for the project. Below is a list of high level activities:

1. More market research after development of the first proof of concept.
2. Developing more accurate requirements for the project
3. Evaluating different options for the drone detection system satisfying the requirements
4. Exploring options for handling user data either locally on user devices or in the cloud.
5. Change the logic for UI and backend to have separate processing threads.

References

1. S. Kunze and A. Weinberger, "Concept for a Geo-Awareness-System for Civilian Unmanned Aerial Systems," 2021 31st International Conference Radioelektronika (RADIOELEKTRONIKA), 2021, pp. 1-6, doi: 10.1109/RADIOELEKTRONIKA52220.2021.9420196.
2. D. Scherer and B. Sherwood, "Web VPython", 2011 https://vpython.org