ECE 445

SENIOR DESIGN LABORATORY

FINAL REPORT

---

# Campus Tour Guide by AI-Powered Autonomous System

---

**Team #21**

XUANBO JIN
(xuanboj2@illinois.edu)
HAO REN
(haor2@illinois.edu)
YUNTONG GU (yuntong7@illinois.edu)
WEIANG WANG
(weiangw2@illinois.edu)

TA: Xinlong Huang

March, 18, 2024

# Contents


**1 Introduction** **1**
1.1 Problem and Solution Overview . . . 1
1.2 Motivation . . . 1
1.3 Visual Aid . . . 1
1.4 High-level requirements list . . . 2
1.5 External Subsystem: Motion Control Subsystem (UAV) . . . 2

**2 Design** **2**
2.1 Block Diagram . . . 2
2.2 Internal Subsystems Overview . . . 2
2.3 AI-powered response generation subsystem . . . 3
2.3.1 System Architecture and Design Overview . . . 3
2.3.2 Data side effort: Multi-media Data Collection . . . 5
2.3.3 Agent side effort: Intent Identification and Retrieval . . . 5
2.3.4 Agent side effort: Protection Sub-Unit . . . 6
2.3.5 Goals and Verification . . . 6
2.3.6 AI-powered Response Generation Subsystem Interface . . . 7
2.4 User Interface . . . 8
2.4.1 Subsystem Architecture and Design . . . 8
2.4.2 Input and Output of the Subsystem . . . 10
2.4.3 Frontend Development . . . 10
2.4.4 Remote Server Setup . . . 10
2.4.5 Connection with AI Subsystem . . . 11
2.4.6 Verification and Results . . . 11
2.5 Planning & Control Subsystem . . . 13
2.5.1 Notation and Explanation . . . 13
2.5.2 Subsystem Architecture and Design . . . 14
2.5.3 Input and Output of the Subsystem . . . 14
2.5.4 Data Collection and Tagging . . . 15
2.5.5 Algorithms for Subsystem . . . 15
2.5.6 Integration with QGroundControl Platform . . . 16
2.5.7 Verification and Results . . . 16
2.6 Sensor System . . . 18
2.6.1 Introduction . . . 18
2.6.2 Temperature and Humidity Sensor . . . 18
2.6.3 Accelerometer Sensor . . . 19
2.6.4 LCD Display . . . 20
2.6.5 PCB design . . . 20
2.6.6 Verification and Requirement Plan . . . 21
2.7 Tolerance Analysis . . . 22
2.7.1 Route Planning Stability . . . 22
2.7.2 GPS Locating Error . . . 22

**3 Cost and schedule** **23**
3.1 Cost Analysis . . . 23

3.1.1 Labor . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 23
3.1.2 Parts . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 23
3.1.3 Grand Total . . . . . . . . . . . . . . . . . . . . . . . . . . . 23

**4 Conclusion** **24**
4.1 Summary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 24
4.2 Ethical and Safety Considerations . . . . . . . . . . . . . . . . . 24

**References** **25**

**Appendix A Requirement & Verification Table** **26**

# 1 Introduction

## 1.1 Problem and Solution Overview

Anyone entering a place for the first time, like an university, can be quite challenging. Knowing where you are, how to get to your destination, how to optimize your routes, knowing factors that will influence your routes can be complicated. Having a real-time interactive system that guides people through this process is needed. It has been possible yet not able to scale because it's not open-sourced, and its hardware isn't standardized, and is expensive. The interaction isn't versatile enough to adapt well under the ever-changing applications. A cheap and versatile solution is needed.

## 1.2 Motivation

The most traditional paradigm is have human tour guide guiding a group of visitors. Then when the era of electrical and electronics engineering came, engineers started automating this process. They designed specific pipelines to emulate this process. However, this automation has intrinsic problem with cost, scalability, as well as generality. Currently, when artificial intelligence prevails, inserting AI as a component in the pipeline has significant downsides. Apart from previous problems, most of the components in the pipeline is not AI-powered, wasting computational resources and efficiency. In light of these problems, we completely shift the pipeline design to a Multi-agent AI operation network design, with each of its component AI powered. It can scale easily as all agents have simple and general interfaces. The average cost is also amortised as the number of agents increase. Most importantly, the paradigm of all the down stream applications, including campus tour guide, is shifted from 1 guide leading 100 visitors to 100 guides leading 1 visitors, letting users to these applications fully exploit the power of AI.

## 1.3 Visual Aid

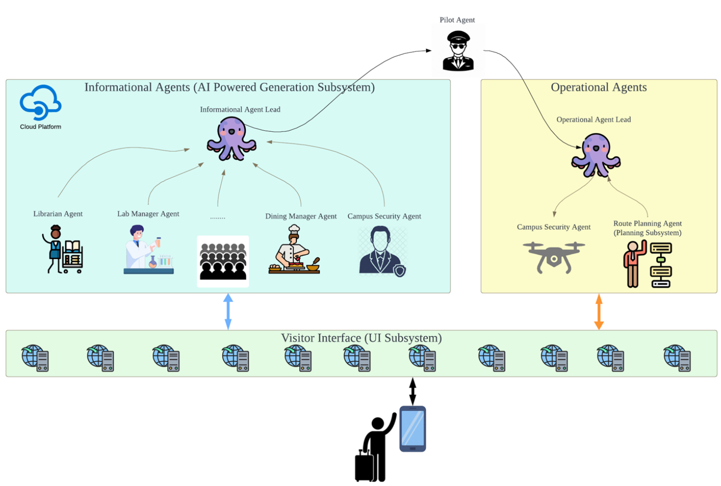


Figure 1: Multi-agent AI operation Visual Illustration

### 1.4 High-level requirements list

- The AI agents network system should be **responsive**. It should respond to user's request appropriately. It should give appropriate guidance to user in both **informational and operational** ways.
- The **interface** must be clean and useful. It should gives user easy access to the service.
- The system must distribute and **map** the request to the correct agent who is most useful in a certain service and is able to merge and **reduce** the response from many agents into one organized response. The choice of agent must be optimized to ensure the best holistic accuracy.

### 1.5 External Subsystem: Motion Control Subsystem (UAV)

The UAV model used in our project is the MFP450, a medium-sized drone platform with a 410mm wheelbase. It is equipped with a Pixhawk 6C open-source flight controller, M8N-GPS, brushless motors, custom hard-shell batteries, Minihomer telemetry, an integrated optical flow ranging module, camera, and other devices. This UAV meets the requirements for stable flight both indoors and outdoors, and it is suitable for various applications including teaching and development. This is an off-the-shelf open source UAV, so we won't go in depth here.

# 2 Design

## 2.1 Block Diagram

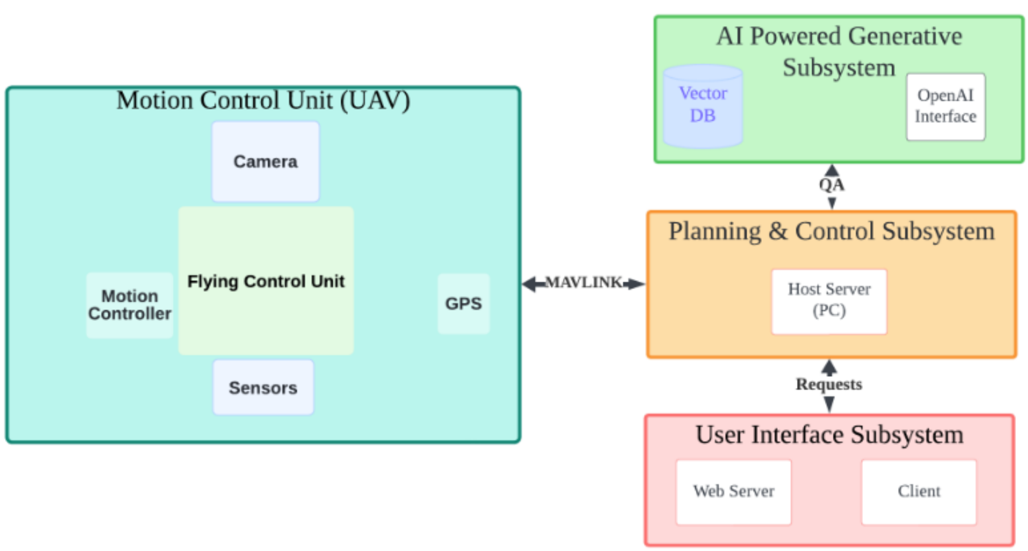


Figure 2: Block Diagram

## 2.2 Internal Subsystems Overview

This projects consist of 4 subsystems:

- AI-powered response generation system
- User Interface
- Planning & Control system
- Sensor System

These sub-systems sense the context of the tour, planning the tour, and interact with users in a natural manner. The emphasis of the sub-systems are the integration of strong ability of Retrieval-Augment Generation into a mobile system. **These subsystems link the cloud, the PC, and the embedded mobile systems into an interactive tour guiding assistant**.

## 2.3 AI-powered response generation subsystem

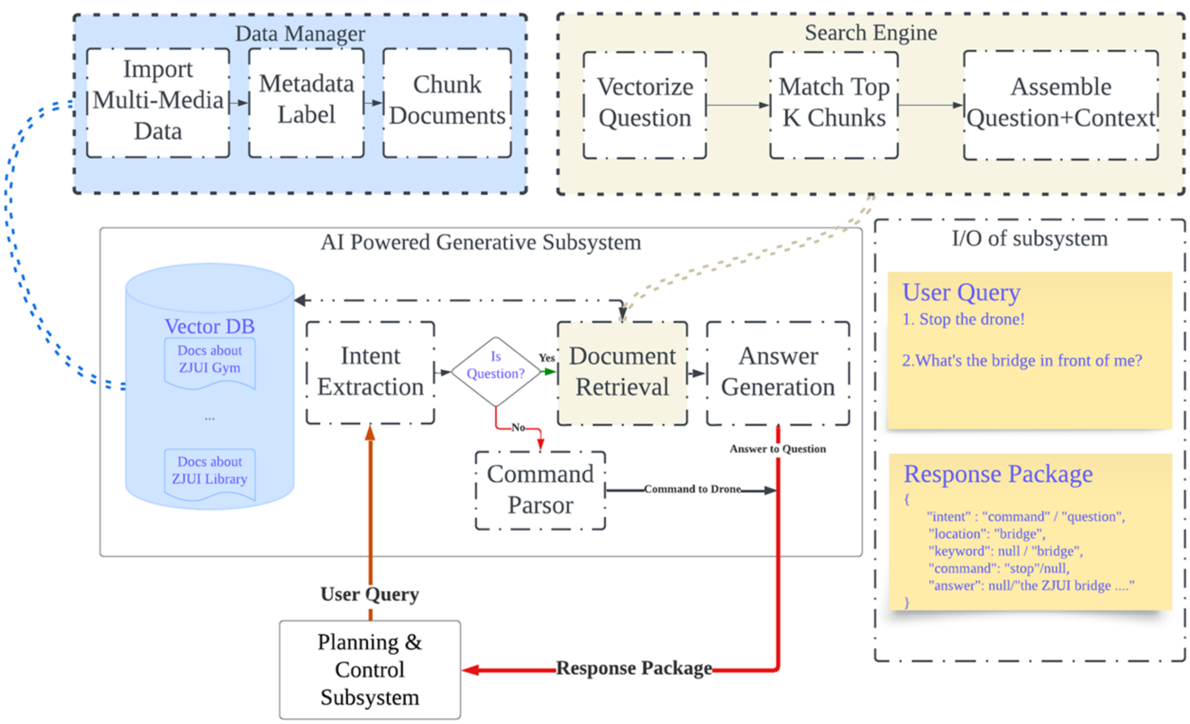


Figure 3: AI-powered response generation system

The AI-powered response generation subsystem focuses on building an assistant to respond to user queries about the ZJU-UIUC campus. The user's query can be a question about the campus or a command to the drone. The text generation and embedding modules are powered by OpenAI[1].

### 2.3.1 System Architecture and Design Overview

The system is architected to efficiently handle two distinct types of queries: informational and operational. The architecture is modular, with each module specializing in tasks such as intent detection, data retrieval, and response generation and validation. This modular design facilitates scalability and maintenance while ensuring the system can evolve to incorporate future enhancements or functionalities.

The system can be broken into the following parts:

- Vector Database(DB) storing related campus material
- Intent Extraction module extracting user's intent.
- Search Engine module search for related entries in Vector DB
- Answer Generation

To build this subsystem, we have two tasks of two aspects:

- Data side tasks: tasks related to multi-media data collection and processing.
- Agent side tasks: tasks related to response generation process.

The data side tasks consist of various work items. First, one must spend times to mine insights from data. This data mining process is crucial to determine the methods used to process data. For instance, If the data for each location is not too long, then we can categorize data by location and handle queries related to each location separately. Based on the insights gained from the multi-media data, we can handle different types of data with different methods. Once we finalize the methods, we can scale up the data size and systematically collect and process data. Hence, the third work item is multi-media data gathering, cleaning and tagging. Eventually when the data gathering is completed, we need to present statistics in clear and organized ways.

The agent side also has many components in its pipeline. The first module of the pipeline is intent identification. It identifies and categorizes user's intent. Then we have a search engine module that extracts the corresponding data which is processed by tremendous data-side effort. Then we have a answer generation module which references the data extracted from search engine, generating a ungrounded response. By ungrounded, we mean the response generated by answer generation module can be nonideal, meaning it can be too long, too dangerous, inaccurate. Hence, we need a protection-unit module to ground the output generated by answer generation module. These modules form a pipeline for the AI powered response generation agent.

The interface of the subsytem is listed as follows:

Table 1: Input and Output of the AI-Powered Response Generation Subsystem

| Field Name | Type | Meaning |
|---|---|---|
| User Query | Input | Campus related query or a command to drone |
| User Location | Input | Current GPS location of user |
| Answer | Output | Answer to user's question |
| Command | Output | Parsed output Command to the drone |

### 2.3.2 Data side effort: Multi-media Data Collection

The GPT model does not have the ability to answer questions related to ZJU-UIUC. It failed to answer the 100 testing questions completely. To supply pertinent information regarding to users' query, we need a set of data tailored for our application. The data we will use are in multiple forms as shown, it could be digital pdfs, or paper-based materials placed at many places at different locations within the campus. We utilize several methods to handle these different forms of information.

1. OCR tool backed by Wechat.
2. python tools to parse pdfs.

The code to parsing multi-media material is in this directory.

We convert these data-sources into uniform textual information so our agent can answer user's query with accurate and diverse information.

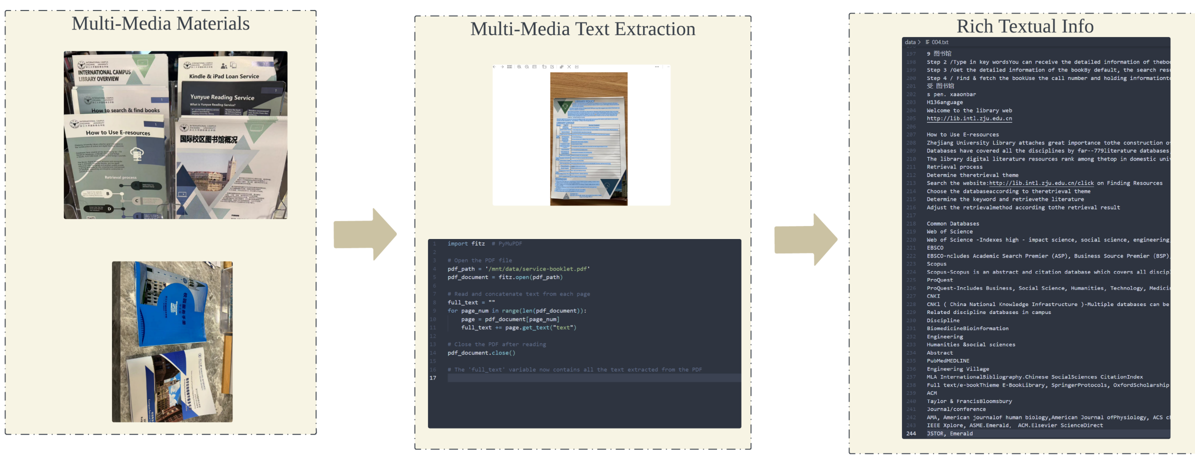


Figure 4: Workflow of Data Collection

Table 2: Vector DB Data Store Table

| Field Name | Type | Meaning |
|---|---|---|
| d_id | INTEGER | Indexing of the entries |
| location | STRING | Location of the described place |
| outlook | STRING | Appearance of the location |
| keyword | STRING | keyword of the description |
| description | STRING | a text describing related info about a location in ZJUI campus |

### 2.3.3 Agent side effort: Intent Identification and Retrieval

The intent identification module classifies users' input into three categories, as specified in the instruction section of the prompt. This module relies on a straightforward query to GPT

3.5. However, given that GPT models typically achieve only about 70% accuracy on general multiple-choice tests [1], we must evaluate its performance on our specific task with caution. Enhancements might be achieved through refined prompt engineering [2].

The information retrieval module employs a Retrieval-Augmented-Generation approach. After extensive review [3], I opted for a sophisticated modular configuration. This agent integrates multimedia input processing, a search engine as the retriever, and a generation unit that processes and outputs the retrieved information.

The document retriever connects queries to relevant locations and retrieves the associated data. However, as data for each location increases, it may surpass GPT's token limit—the maximum number of tokens an LLM can process at one time. To manage this, the retriever breaks down data into manageable "chunks," summarizing each chunk and converting them into vectors using an OpenAI tool. The vector distance indicates the textual semantic similarity, which aids in identifying the most relevant chunks for a query.

#### 2.3.4 Agent side effort: Protection Sub-Unit

Just like any hardware system, the subsystem's IO and behavior needs appropriate amount of protection. For instance, we must ground the I/O of the module so that the module would not output illegal commands or commands that exceed the tolerance of the other units.

The specific groundings are listed in the following table:

Table 3: AI-Powered Response Generation Subsystem Protection Unit

| Index | Protection Object | Protection Explanation |
|---|---|---|
| 0 | Input text | Either campus-related question or a command to drone |
| 1 | Output Text | Must have a length between 0 to 255 words |
| 2 | Output Command | Valid command to drone |
| 3 | Output Text | Must exceed confidence threshold |

#### 2.3.5 Goals and Verification

In order to verify the design, our system has the following 3 main target functionalities:

- Identify the intent of the input to the subsystem
- Fetch the correct external materials
- Output proper answer

In order to verify these functionalities, we design the following verification methods:

- The intent is evaluated automatically against the ground truth labeled by human on a testing dataset containing 100 sample questions.

- The retrieval accuracy will be evaluated automatically against ground truth labeled by human on a testing question set containing 30 human labeled questions on 4 testing locations.
- The end-to-end answer accuracy will be evaluated on the same dataset as retrieval accuracy. This evaluation will be conducted by human, giving a score of 5 and a comments pointing out potential issue. The comments are visualized and analyzed through word cloud.

The initial verification process will be shown below:

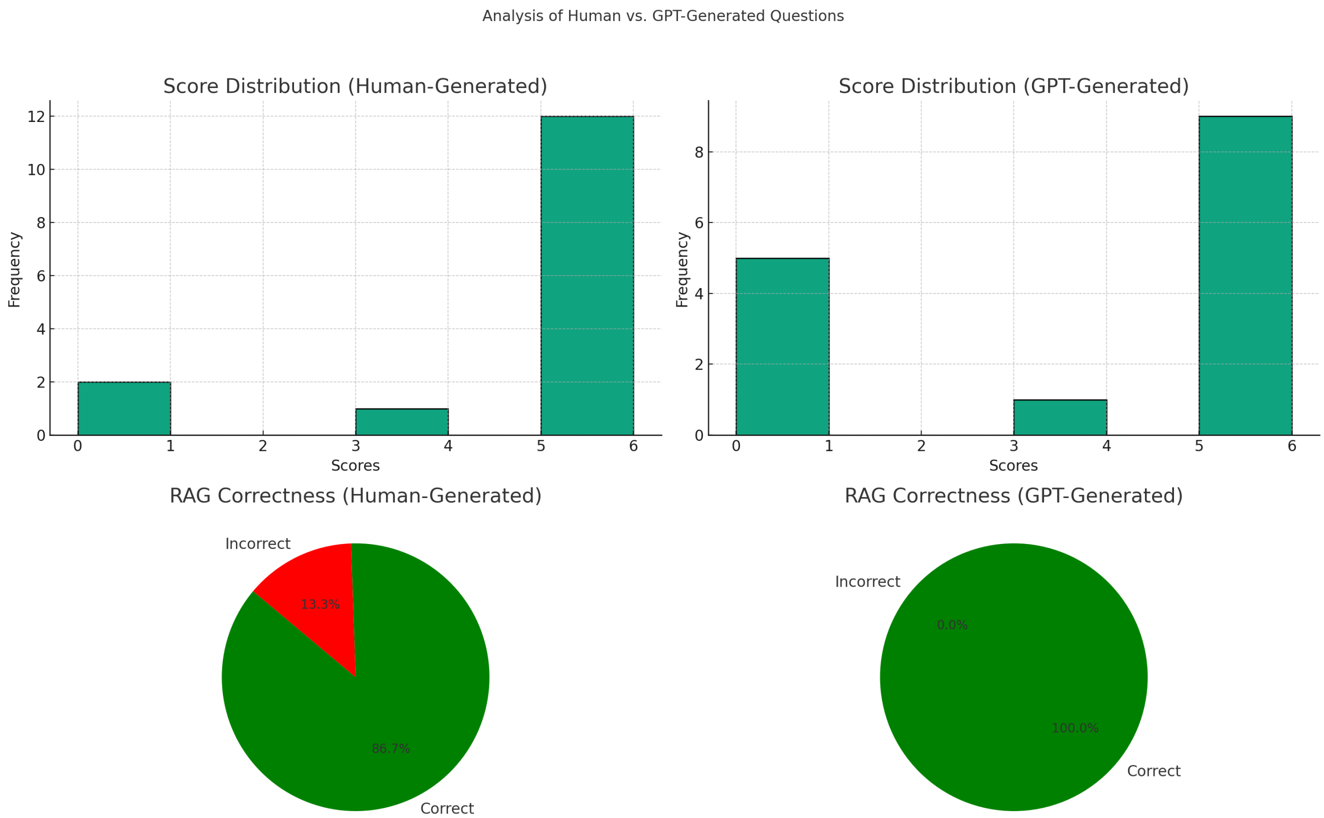


Figure 5: Analysis of End to End Accuracy as well as RAG accuracy

#### 2.3.6 AI-powered Response Generation Subsystem Interface

The code is available at project_repos

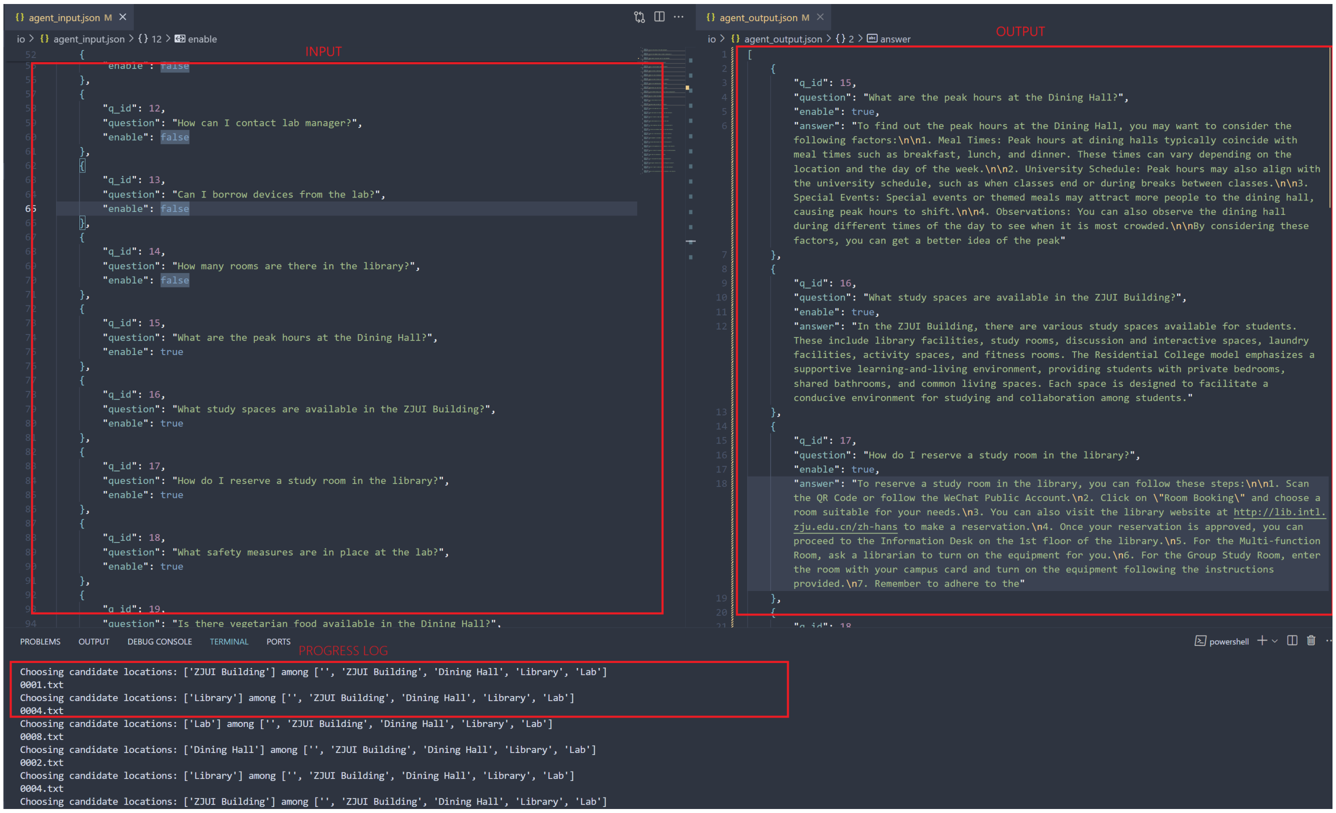


Figure 6: Interface Demo

## 2.4 User Interface

The User Interface (UI) subsystem serves as the primary point of interaction between the users and the AI-powered response generation system. It is designed to be intuitive and user-friendly, enabling users to easily submit queries about the ZJU-UIUC campus and issue commands to the UAV.

### 2.4.1 Subsystem Architecture and Design

The User Interface (UI) subsystem is the principal conduit for user interactions within the AI-powered response generation system. It is meticulously crafted to be both intuitive and user-friendly, enabling seamless submission of queries and UAV command issuance.

The subsystem is comprised of several key components:

- A web server that accepts and distributes messages.
- Client interfaces for visitors to interact with the system.

The architecture delineates the following operational flow:

1. The web server receives messages from various visitors through the user interface.
2. Depending on the message type, identified as a question or a command, the server routes the message to the respective subsystem for processing.

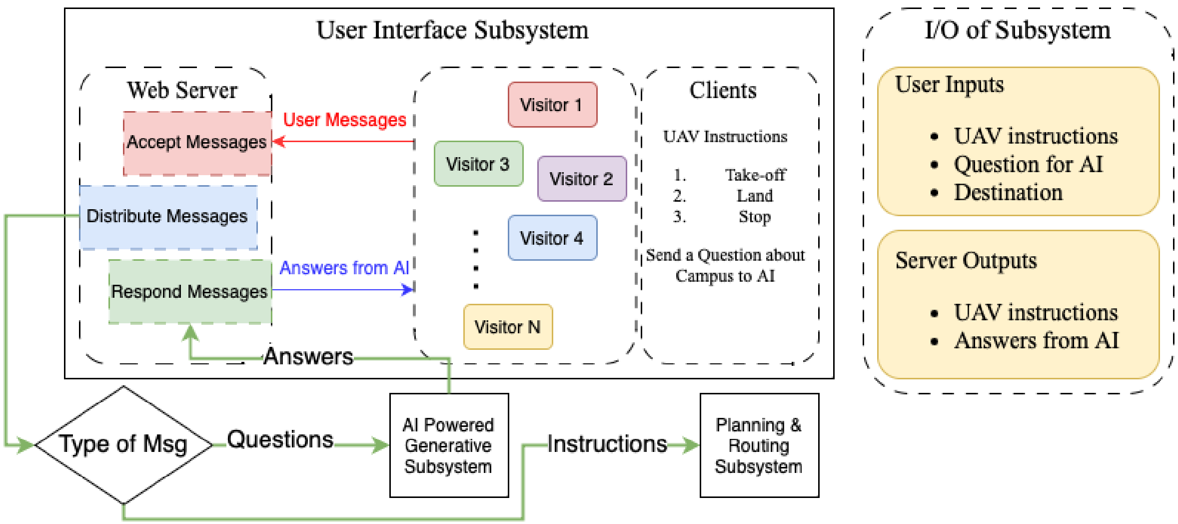


Figure 7: User Interface Diagram

The design leverages a set of defined APIs to manage the interactions between the UI and other subsystems, promoting real-time processing and ensuring data consistency and reliability. The modular nature of the design allows for scalable and maintainable enhancements, critical for future integration and functionality expansion.

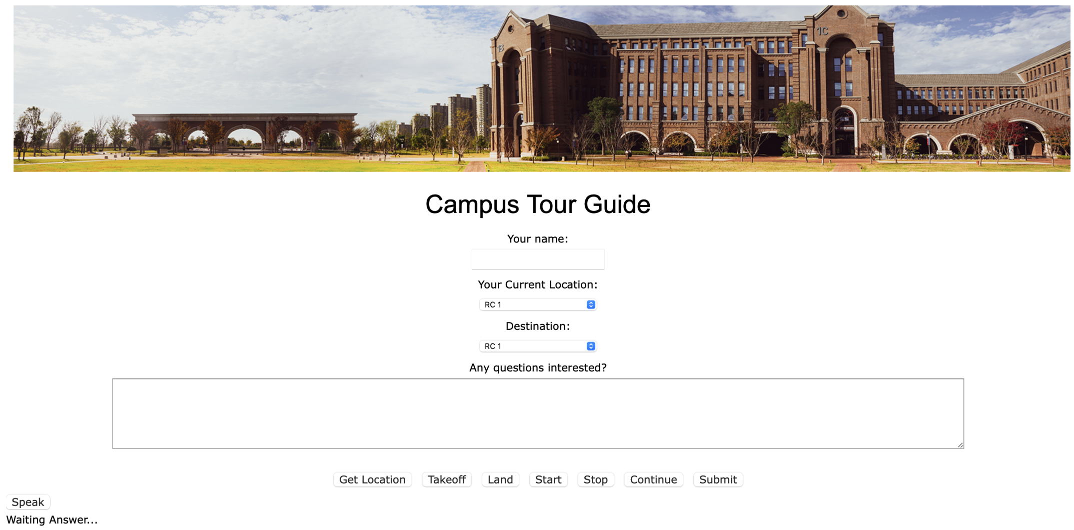


Figure 8: User Interface

The envisioned deliverables include:

- Hosting the web service on a remote server.
- Developing an easy-to-navigate, full-stack framework.
- Establishing a robust connection between the web server and the AI and Planning subsystems.

This architecture is designed to provide a seamless, efficient, and secure user experience, whether it's for informational queries or operational control over the UAV.

#### 2.4.2 Input and Output of the Subsystem

Table 4: Input and Output of the User Interface Subsystem

| Field Name | Type | Meaning |
|---|---|---|
| UAV Instructions User2Server | Input | Take-off, Land, Stop instruction to UAV |
| User Questions | Input | Questions about ZJUI Campus |
| User Destination | Input | User's destination |
| AI Answer | Output | Answer to user's question |
| UAV Instructions Server2UAV | Output | Take-off, Land, Stop instruction to UAV |

The input to the user interface subsystem is **the answer to the user's questions and the status of the drone**. The output of the subsystem is **questions by the user and commands to the drone**.

#### 2.4.3 Frontend Development

The frontend of the UI subsystem is developed using React and JavaScript, offering a dynamic and responsive web interface. The design features a minimalist layout to enhance user experience and facilitate ease of use. Key elements of the UI include:

- **Question Input Block**: A dedicated area where users can type in or voice their queries about the ZJU-UIUC campus.
- **Instruction Buttons**: Several interactive buttons designed to issue predefined commands to the UAV, such as "Take off," "Land," and "Stop". Besides, there is an additional button to send questions to AI-powered Generative System "Send".
- **Campus Image Display**: An image block that dynamically displays photographs of the ZJUI campus, which could be relevant to the user's queries or for showcasing UAV functionalities.

This design ensures that users have a straightforward and efficient way to interact with the system, whether seeking information or controlling the UAV.

#### 2.4.4 Remote Server Setup

The subsystem utilizes Ali Cloud for hosting the remote server, establishing a robust and scalable infrastructure. The connection to the server is secured via SSH, ensuring encrypted communication channels. This server plays a critical role in:

- Managing connections between end-users and the Planning & Control subsystem.

- Facilitating data exchange between the UI and AI subsystems, ensuring seamless integration and real-time response capabilities.

#### 2.4.5 Connection with AI Subsystem

The integration between the UI and the AI subsystem is essential for the real-time processing of user queries and drone commands. This connection is established through a well-defined API that:

- Allows for the seamless transmission of user inputs from the UI to the AI subsystem, where they are processed to generate responses or drone commands.
- Enables the AI subsystem to send back the generated responses or status updates directly to the UI, ensuring that users receive immediate and relevant feedback.

This integration is designed to be highly efficient, minimizing latency and maximizing the accuracy and relevance of the information provided to the users.

#### 2.4.6 Verification and Results

The verification of the User Interface Subsystem involved a series of rigorous tests and evaluations to ensure its functionality, usability, and performance met the project requirements. The following methodologies were employed:

1. **Functional Testing:** Comprehensive functional tests were conducted to verify that all features of the UI subsystem, including question input, instruction buttons, and campus image display, worked as intended.
2. **Usability Testing:** Usability testing sessions were conducted with a diverse group of users to evaluate the UI's ease of use, intuitiveness, and effectiveness in facilitating interaction with the AI-powered Campus Tour Guide UAV.
3. **Performance Testing:** Performance tests were carried out to assess the responsiveness and reliability of the UI subsystem under various load conditions, ensuring it could handle multiple concurrent user requests without degradation in performance.
4. **Integration Testing:** Integration tests were performed to validate the seamless communication between the UI subsystem and other project modules, including the AI subsystem and the Planning & Control subsystem.


```
{
  "userID": "966",
  "destination": "ZJUI Buidling",
  "question": "How many staffs are there in ZJUI Building?",
  "instruction": "Take-off"
}
```


Figure 9: User Infomation

The results of the verification process are summarized as follows:

1. **Functional Testing:** All features of the UI subsystem were successfully tested and found to be functioning according to specifications. Users were able to input questions, issue instructions to the UAV, and view relevant campus images without encountering any major issues.

2. **Usability Testing:** Feedback from usability testing sessions indicated that users found the UI to be intuitive and easy to navigate. The question input block, instruction buttons, and campus image display were all praised for their clarity and responsiveness.

3. **Performance Testing:** Performance tests demonstrated that the UI subsystem could handle a significant number of concurrent user requests without noticeable slowdowns. Response times remained within acceptable limits even under heavy load conditions.

4. **Integration Testing:** Integration tests confirmed seamless communication between the UI subsystem and other project modules. Data exchange with the AI subsystem for question answering and the Planning & Control subsystem for UAV instructions occurred without errors.

## 2.5 Planning & Control Subsystem

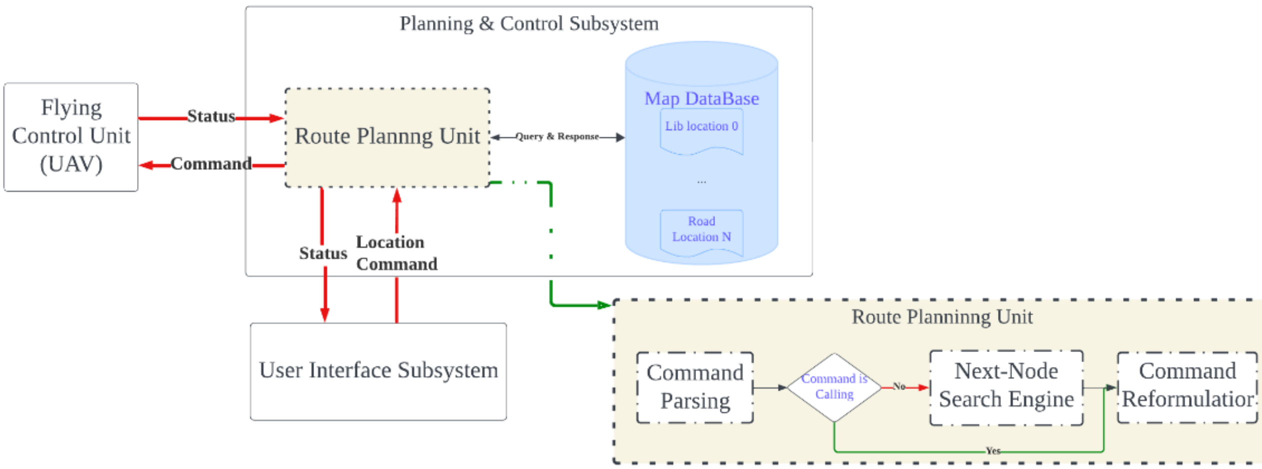


Figure 10: Planning and Control Block Diagram

The Planning and Control Subsystem is an integral component of our UAV operational framework, designed to process commands from the user interface, assess the current status of the drone, and issue precise navigational instructions based on the drone's specifications. This subsystem interfaces directly with PX4 APIs, an open-source flight control software, to monitor and control the UAV's flight parameters. The primary input to this subsystem is **the user's command and the drone's status**, while its output is the **command to the UAV**, ensuring that each operation is executed safely and efficiently.

### 2.5.1 Notation and Explanation

Table 5: Notations Used for Planning and Control Subsystem

| Name | Meaning |
|---|---|
| Node | The map is continuous, we extract a sets of locations as nodes |
| Map Database | A data-store for the node |
| Next Node | The next node we plan to go to |
| Link | The minimum route unit linking 2 nodes |
| Command | An instruction to UAV or from user |
| Parsed Command | An instruction equivalent to a set of MAVSDK APIs |
| MAVSDK | a software dev kit consists of APIS instructing UAV |
| Search Engine | A module searching for next node given current location and command |
| Reformulation | A process formulating instruction to comply with MAVSDK APIs |

### 2.5.2 Subsystem Architecture and Design

This subsystem obtains a command and it reformulates the command to the drone to execute. The reason why this module is essential and vital is this module helps visitor has a safe and comfortable trip. This subsystem objective is to find a short and comfortable route for user to take while taking the tour inside ZJU-UIUC campus.

This subsystem establishes a robust connection with a remote server to access vital flight data, including starting points, destinations, flight altitudes, and other navigational parameters. This connection is crucial for retrieving real-time information necessary for flight planning and control. The communication between the Planning & Control Subsystem and the remote server is facilitated through a secure, encrypted channel, ensuring the integrity and confidentiality of transmitted data. This setup allows the subsystem to:

- Obtain real-time updates on weather conditions, no-fly zones, and other environmental factors that may affect flight paths.
- Receive user-defined flight parameters such as starting location, destination, and preferred flight height.
- Update the UAV's mission parameters in response to changing conditions or user commands.

### 2.5.3 Input and Output of the Subsystem

Table 6: **Inputs and Outputs**

| **Inputs** | **Outputs** |
|---|---|
| **From the Unmanned Aerial Vehicle (UAV):**<br>• Latitude and longitude position of the UAV.<br>• Current velocity, acceleration, and attitude (orientation) of the UAV. | **To the Unmanned Aerial Vehicle (UAV):**<br>• Attitude control commands for the UAV.<br>• Velocity and acceleration control commands for the UAV. |
| **From the AI-Powered Generative System:**<br>• Responses generated by the AI system based on user queries. | **To the AI-Powered Generative System:**<br>• User queries directed to the AI system. |
| **From the User Interface System:**<br>• User commands and requests submitted through the interface. | **To the User Interface System:**<br>• Responses provided to users based on their queries. |

### 2.5.4 Data Collection and Tagging

We used an n*n adjacency matrix to represent the entire map, where n is the number of locations in the map. A link is a pair of nodes. A link is a viable path. Any connections between nodes that are not linked are not a valid path. For instance, if the connection between 2 nodes cross a lake which visitor is impossible to follow, it won't be our data store.

### 2.5.5 Algorithms for Subsystem

Let's define the mathematical model for the planning and control subsystem:

- **Nodes** $N$: A set of extracted locations from the continuous map, which serve as possible waypoints for the UAV.
- **Map Database** $D$: A datastore that contains the nodes and links information.
- **Links** $L$: Directed edges between nodes representing the minimum navigable path for the UAV. Each link connects two nodes and has associated costs (like distance, time, or energy consumption).
- **Commands** $C$: Instructions from the user or system that need to be executed by the UAV.
- **Parsed Commands** $P$: Translated commands into MAVSDK API calls.
- **MAVSDK** $M$: The software development kit used to control the UAV.

Given:

- **Current Node** $n_{\textbf{current}}$: The UAV's current position represented as a node.
- **Destination Node** $n_{\textbf{destination}}$: The target position the UAV needs to reach.

**Objective**

Find the optimal path $\Pi$ from $n_{\text{current}}$ to $n_{\text{destination}}$ minimizing the cost function $F$, which could include factors like distance, time, energy, etc.

**Constraints**

- The UAV can only travel along links in $L$ from one node to another.
- The path must start near $n_{\text{current}}$

**BFS Algorithm for Pathfinding**

BFS algorithm can be used to find the path from $n_{\text{current}}$ to $n_{\text{destination}}$ in a graph represented by nodes and links.

1. Initialize a queue $Q$ and enqueue the starting node $n_{\text{start}}$.
2. Create a set $S$ to track visited nodes and add $n_{\text{start}}$ to $S$.
3. While $Q$ is not empty:
    - Dequeue the front node $n$ from $Q$.
    - If $n$ is the destination node $n_{\text{destination}}$, terminate the search and process the result as needed.

- For each neighbor $n_{\text{next}}$ of $n$:
  - If $n_{\text{next}}$ has not been visited (not in $S$):
    - Enqueue $n_{\text{next}}$ into $Q$.
    - Add $n_{\text{next}}$ to $S$.

The result of BFS algorithm is the path $\Pi$ that optimizes the cost function $F$, providing an efficient route for the UAV from the starting point to the destination.

**Implementation with MAVLink Python Package:** The MAVLink Python package provides a comprehensive set of tools for communicating with the UAV, offering functionalities such as:

- Sending navigational commands and mission updates to the UAV.
- Receiving real-time status information, including location, battery level, and flight mode.
- Managing telemetry data to monitor and adjust flight parameters as needed.

This package is instrumental in bridging the gap between high-level operational commands and the low-level directives understood by the UAV, ensuring that the subsystem can effectively translate user intentions into actionable flight paths.

#### 2.5.6 Integration with QGroundControl Platform

QGroundControl (QGC) represents a pivotal tool in the UAV operational toolkit, offering an intuitive and feature-rich ground control station interface for UAV management. This open-source platform provides full flight control and mission planning capabilities for any MAVLink-enabled drone, making it an indispensable asset for our project.

The synergy between QGroundControl and our subsystems significantly enhances our UAV's operational capabilities, enabling sophisticated mission planning, real-time control, and detailed performance analysis. Through this integration, we aim to achieve a level of precision, safety, and efficiency that sets a new standard for UAV operations within the ZJU-UIUC campus context.

#### 2.5.7 Verification and Results

The verification of the Planning & Control Subsystem involved a comprehensive evaluation of its capabilities in interpreting flight missions, generating optimal routes, and controlling the UAV during flight operations. The following methodologies were employed:

1. **Mission Interpretation Testing:** Testing was conducted to ensure that the subsystem could accurately interpret flight missions specified in text files, including waypoints, altitudes, speeds, and pause durations.
2. **Route Generation Testing:** Tests were carried out to verify the efficiency and reliability of the BFS algorithm implementation in generating optimal flight routes while considering factors such as terrain obstacles and no-fly zones.

3. **Flight Control Testing:** Flight control tests were performed to assess the subsystem's ability to execute flight missions autonomously, including takeoff, landing, and waypoint navigation, using MAVSDK scripts.
4. **Integration Testing:** Integration tests were conducted to validate the subsystem's seamless integration with other project modules, including the User Interface Subsystem, AI subsystem, and Sensing subsystem.

The results of the verification process for the Planning & Control Subsystem are summarized as follows:

1. **Mission Interpretation Testing:** The subsystem successfully interpreted various flight missions specified in text files, accurately translating waypoints and other parameters into executable commands for the UAV.
2. **Route Generation Testing:** BFS algorithm consistently generated optimal flight routes, avoiding obstacles and adhering to operational constraints. The subsystem demonstrated reliability in finding efficient paths even in complex environments.
3. **Flight Control Testing:** Flight control tests confirmed the subsystem's capability to execute flight missions autonomously, including takeoff, landing, and waypoint navigation. MAVSDK scripts facilitated smooth operation and precise control of the UAV.
4. **Integration Testing:** Integration tests revealed seamless communication between the Planning & Control Subsystem and other project modules. The subsystem effectively received user-defined destinations from the UI subsystem, integrated AI-generated insights, and utilized real-time environmental data from the Sensing subsystem for adaptive flight planning.

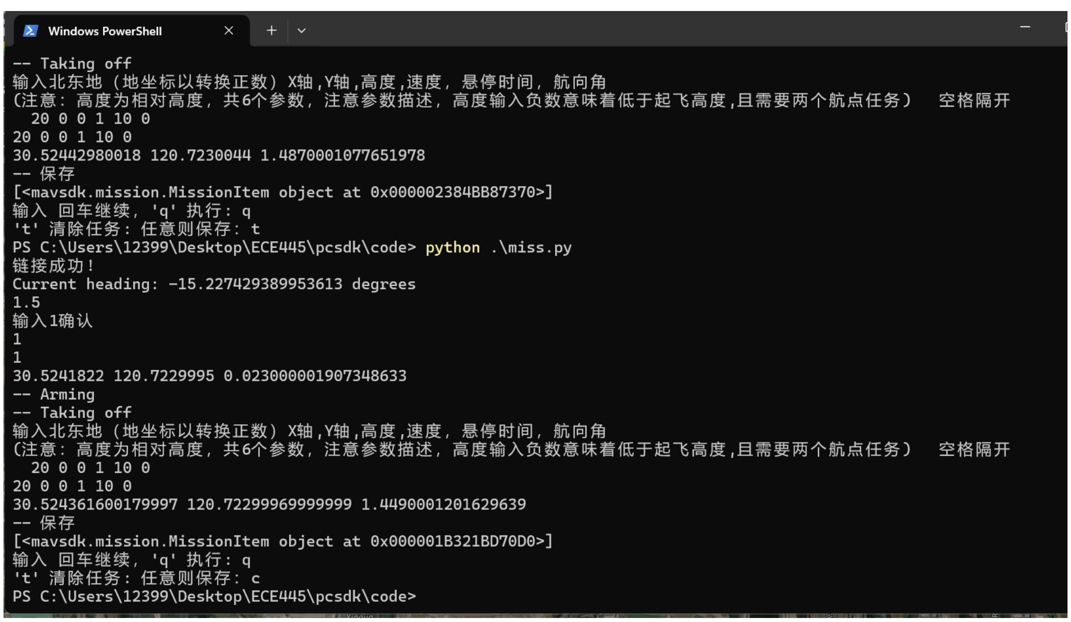


Figure 11: UAV Flight mission by Planning & Routing Subsystem

## 2.6 Sensor System

This section provides an overview of the sensor subsystem, detailing the specific sensors used for measuring humidity, temperature, and angular rate. Each sensor's type, model, and primary functionality are outlined to give a clear understanding of how they contribute to the overall system.

### 2.6.1 Introduction

We want to connect a temperature and humidity sensor to a PCB board, powered by a drone, with the sensor's temperature and humidity measurements displayed on an external LCD screen. This way, as the drone follows the user, the user can view real-time weather conditions.

Here is the schematic diagram of our overall sensor section. From the diagram, we can see that the entire circuit is controlled by the STM chip. Firstly, it converts the 5V regulated power supply from the aircraft input to a 3.3V voltage. Then, it connects to the temperature and humidity sensors as well as the speed sensor to obtain data. Subsequently, the acquired data is transmitted to the LCD display screen by the microcontroller for displaying the results.

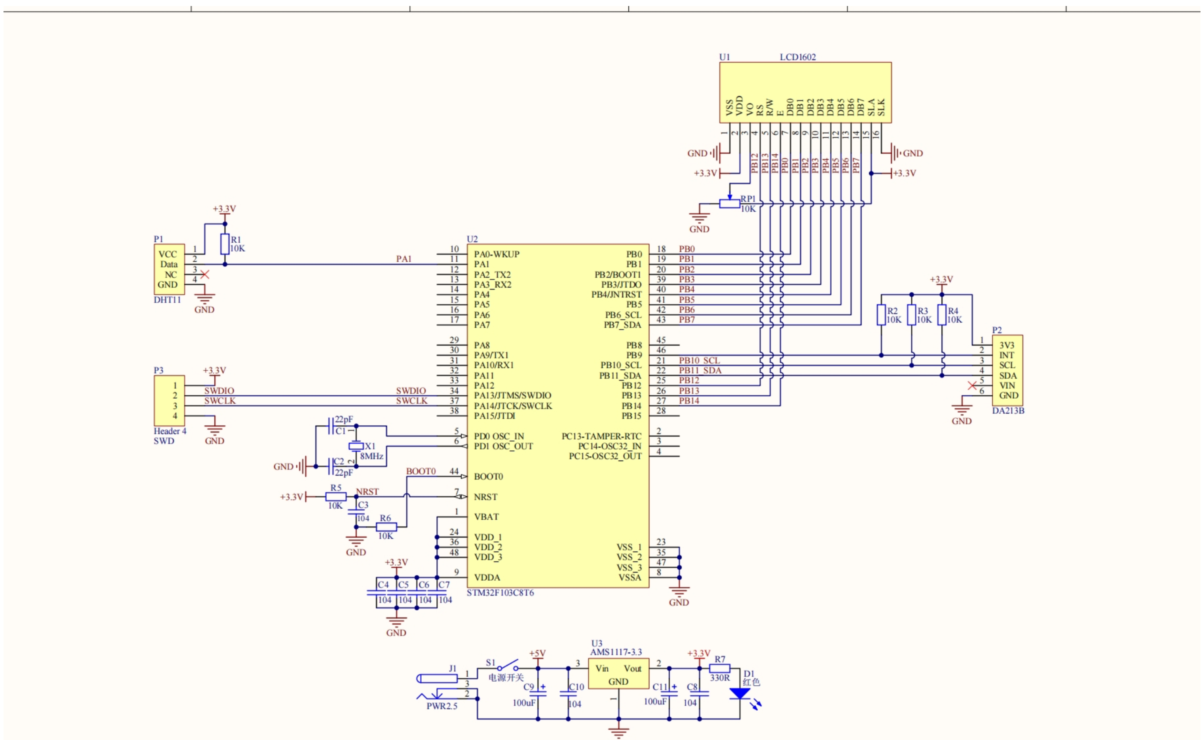


Figure 12: Circuit Design

### 2.6.2 Temperature and Humidity Sensor

- **Type & Model:** DHT-11

- **Functionality:** We selected the DHT11 sensor mainly because of its simplicity, ease of use, and low cost, as well as its ability to accurately measure environmental temperature and humidity. Its digital output makes data reading and processing more convenient, and it offers high accuracy and stability, providing reliable temperature and humidity measurements in various environmental conditions to ensure flight safety and stability for our project.[4]

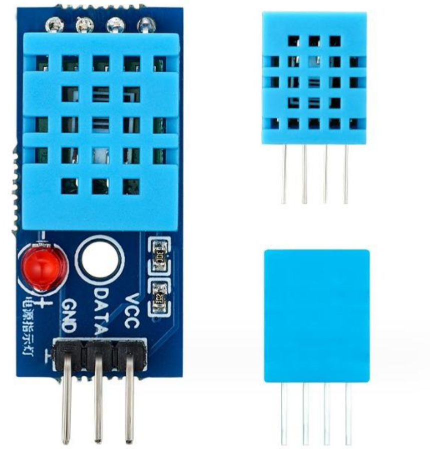


Figure 13: DHT 11 Outside view

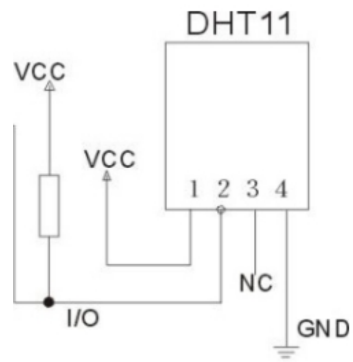


Figure 14: DHT11 pin diagram

### 2.6.3 Accelerometer Sensor

- **Type & Model:** DA213B

- **Functionality:** The DA213B accelerometer sensor was chosen primarily for its high precision, sensitivity, and stability. Capable of accurately measuring acceleration in three axial directions with digital output, it offers convenient data reading and processing. Additionally, its low power consumption and small footprint make it suitable for embedded systems and mobile device applications. In our project, the DA213B sensor provides precise acceleration data for attitude control and motion tracking, enhancing flight stability and accuracy.[5]

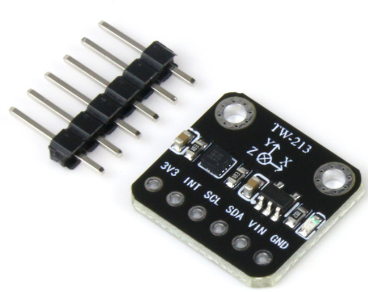


Figure 15: DA213B Outside view

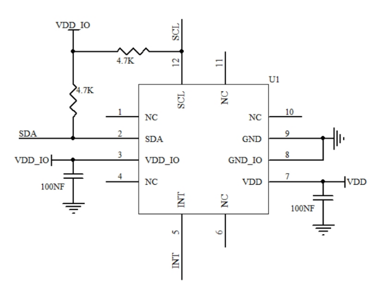


Figure 16: DA213B pin diagram

### 2.6.4 LCD Display

- **Type & Model:** LCD1602

- **Functionality:** The LCD1602 screen was selected primarily for its simplicity and affordability. It can display 16 columns and 2 rows of characters, providing basic text display functionality. Utilizing the standard HD44780 controller, it communicates easily with microcontrollers through a simple interface. Moreover, the LCD1602 screen has low power consumption, making it suitable for embedded systems and mobile devices. In our project, it provides a convenient way to display sensor data, system status, and user interface elements, simplifying user interaction and making system operation more intuitive.


Figure 17: LCD1602 Outside view

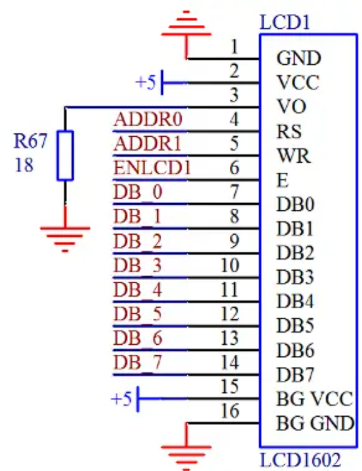


Figure 18: LCD1602 pin diagram

### 2.6.5 PCB design

Here is our PCB design diagram. We have designed the PCB board to be 40cm * 60cm in size to match the dimensions of the drone. Two sensors are directly soldered onto the PCB board, while the LCD screen is connected externally using ribbon cables. The input terminal of the PCB board is supplied with 5V voltage from the drone, and the output terminal is connected to the LCD screen.

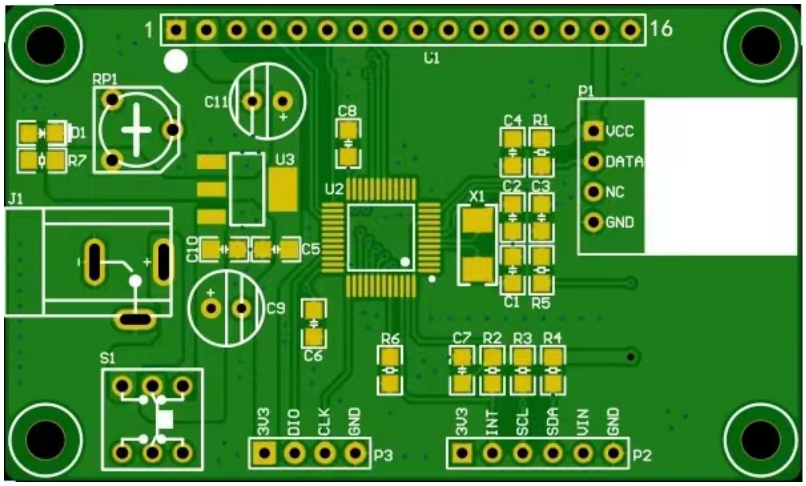


Figure 19: PCB Design

#### 2.6.6 Verification and Requirement Plan

There are several functionalities we want to achieve for the sensor unit:

1. The stability of converting 1.5V voltage to 3V voltage.
2. Whether the two sensors can operate properly, achieve data measurement, and transmit data.
3. Whether the LCD screen can display normally and show the data we need in real-time.

We plan to verify these goals using the following verification methods.

1. Utilize a stable power supply to provide a 1.5V input voltage, connecting it to the voltage conversion circuit. Measure the output voltage using an oscilloscope or digital multimeter and record its value. Continuously observe the stability of the output voltage over a period of time (e.g., 30 minutes to 1 hour), noting any fluctuation range. Repeat the test steps to check the stability of the circuit under different temperature and load conditions.
2. Connect the sensors to the test board and use an oscilloscope or microcontroller to read the sensor's output data. Simulate sensor operation under different environmental conditions, such as varying temperature, humidity, and light levels, and observe the sensor's response. Verify whether the sensor output data matches the expected values and record any anomalies. Test the data transmission functionality to ensure that sensor data can be successfully transmitted to the microcontroller or other devices and correctly interpreted.
3. Connect the LCD screen to the test board and provide appropriate power. Send test data to the LCD screen and observe whether the screen displays correctly. Check whether the LCD screen can display various types of data, including text, numbers, and graphics. Under simulated working conditions such as temperature changes and vibration, verify whether the LCD screen can stably display data. If possible, conduct long-term testing to confirm the stability and durability of the LCD screen.

## 2.7 Tolerance Analysis

Ensuring the robustness of the UAV system involves conducting a thorough tolerance analysis. This analysis focuses on identifying potential vulnerabilities within the system, assessing risks, and devising strategies to mitigate these risks.

### 2.7.1 Route Planning Stability

Route planning is susceptible to various environmental factors that can affect the UAV's ability to navigate effectively. These factors include wind, physical obstacles, and the presence of visitors within the campus.

**Risks:**

- Wind can significantly alter the UAV's course, leading to deviations from the planned route and potentially unsafe conditions.
- Obstacles such as trees and buildings may not only hinder the UAV's path but also pose a risk of collision.
- Visitors moving unpredictably through the campus can introduce dynamic variables, complicating the UAV's navigation and safety protocols.

Wind's impact on UAV navigation can be analyzed using vector mathematics, specifically the wind triangle theory. The ground speed vector ($\vec{V}_g$) of the UAV is the vector sum of its airspeed vector ($\vec{V}_a$), which is the speed and direction relative to the air, and the wind speed vector ($\vec{V}_w$), which represents the speed and direction of the wind. This relationship is given by:

$$\vec{V}_g = \vec{V}_a + \vec{V}_w \tag{1}$$

### 2.7.2 GPS Locating Error

Global Positioning System (GPS) technology is crucial for UAV navigation, offering real-time location data that guides the UAV's flight path. However, GPS signals can be subject to interference from environmental factors, such as atmospheric conditions, buildings, and signal jamming, leading to potential errors in location accuracy.

**Risks:**

- **Mission Failure:** Critical missions requiring precise location data, such as aerial photography or targeted delivery, could fail due to inaccurate positioning.

GPS locating error ($E_{gps}$) can be influenced by several factors, including signal propagation delay, atmospheric conditions, and multipath errors. The total error can be modeled as a combination of these factors:

$$E_{gps} = E_{propagation} + E_{atmospheric} + E_{receiver} \tag{2}$$

# 3 Cost and schedule

## 3.1 Cost Analysis

### 3.1.1 Labor

The labor cost is calculated based on the working hours and wage pricing of each team member. We set the hourly wage at 100 RMB based on market research and the skill levels of team members. Considering the total project duration of 8 weeks with 40 hours of work per week, the total working hours per team member are: 320 hours. Therefore, the labor cost per team member is:

$$100 \text{ RMB/hour} \times 320 \text{ hours} = 32000 \text{ RMB}$$

We chose an hourly wage of 100 RMB, which is based on market wage levels and the skill and experience levels of team members. According to survey data from the Institute of Electrical and Electronics Engineers (IEEE) [6], the average salary for graduates in Electrical and Computer Engineering (ECE) is around 200,000 RMB per year. Calculated on a full-time basis, the average hourly wage is approximately 100 RMB.

### 3.1.2 Parts

The table below provides a breakdown of the parts and their estimated costs:

| Description | Manufacturer | Part # | Quantity | Cost (RMB) |
|---|---|---|---|---|
| Drone | PixHawk | MFP450 | 1 | 5174 |
| Mavlink Module | Amovlab | - | 1 | 680 |
| Temperature Sensor | Aosong | DHT11 | 2 | 10 |
| Acceleration Sensor | MiraMEMS | DA213B | 2 | 15 |
| LCD Screen | Touglesy | LCD1602 | 1 | 20 |
| PCB Board | Custom | - | 1 | 50 |
| Simple Application Server | Alibaba Cloud | - | 1 | 49 per month |
| ChatGPT4 API | OpenAI | - | 1 | 240 per month |

Table 7: Parts List and Estimated Costs

### 3.1.3 Grand Total

The grand total cost of the project can be calculated by summing up the labor cost and the cost of parts:

- Labor: 32,000 RMB
- Parts: 6,819 RMB

Grand Total: 32000 + 6819 = 38,819 RMB

# 4 Conclusion

## 4.1 Summary

In this project, we implemented an AI-based campus tour guide assistant. We utilized a drone as the guiding platform to facilitate a seamless campus tour experience while providing informative responses to users' inquiries. The AI agent network system we developed prioritizes responsiveness, ensuring prompt and relevant answers to users' requests. This system not only offers guidance on campus exploration but also addresses user queries comprehensively.

Moreover, we emphasized the importance of a clean and user-friendly interface, enabling easy access to the service. A well-designed interface enhances user engagement and satisfaction, enhancing the overall tour experience.

Additionally, our system incorporates a sophisticated agent distribution and mapping mechanism. Requests are intelligently routed to the most suitable agent capable of addressing specific services. Furthermore, the system optimizes agent selection to ensure the highest accuracy in response aggregation. By merging and organizing responses from multiple agents, we deliver a cohesive and comprehensive user experience, enhancing the effectiveness of our AI-based campus tour guide assistant.

## 4.2 Ethical and Safety Considerations

The development and deployment of the AI-guided tour guide drone raise important ethical considerations that must be addressed. One key concern is privacy. The use of drones for guided tours may intrude upon individuals' privacy rights, particularly if they capture images or videos without consent. It's imperative to prioritize privacy and ensure that data collection and usage adhere to ethical standards and legal regulations.[7]

Safety is paramount in the development and operation of the AI-guided tour guide drone. Several safety measures will be implemented to mitigate risks and ensure the well-being of users and developers.

Firstly, the drone's hardware and software systems will undergo rigorous testing and validation to ensure their reliability and stability. Emergency shutdown protocols will be in place to address malfunctions or emergencies promptly.

Secondly, strict battery safety protocols will be enforced to prevent accidents related to lithium polymer (LiPo) batteries. This includes regular inspection, proper storage, and careful handling to minimize the risk of fire or explosion.

# References

[1] OpenAI, J. Achiam, S. Adler, *et al.*, *Gpt-4 technical report*, 2024. arXiv: 2303.08774 [cs.CL].
[2] L. Weng, "Prompt engineering," *lilianweng.github.io*, Mar. 2023. [Online]. Available: https://lilianweng.github.io/posts/2023-03-15-prompt-engineering/.
[3] P. Zhao, H. Zhang, Q. Yu, *et al.*, *Retrieval-augmented generation for ai-generated content: A survey*, 2024. arXiv: 2402.19473 [cs.CV].
[4] Aosong Electronics Co., Ltd. "Aosong Product 21." (2024), [Online]. Available: http://www.aosong.com/products-21.html (visited on 04/19/2024).
[5] MiraMEMS. "MiraMEMS Product 1." (2024), [Online]. Available: http://www.miramems.com/product-1.html (visited on 04/19/2024).
[6] "IEEE (Institute of Electrical and Electronics Engineers) Salary Survey," 2022.
[7] IEEE. ""IEEE Code of Ethics"." (2016), [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html (visited on 02/08/2020).

# Appendix A Requirement & Verification Table

Table 8: Subsytem Index Table

| Subsystem Name | Subsystem Index(S-Index) |
|---|---|
| AI Powered Response Generation Subsystem | 1 |
| User Interface Subsystem | 2 |
| Planning and Control Subsystem | 3 |
| Sensor Unit Subsystem | 4 |

Table 9: Requirement & Verification Table

| S-Index | Requirement | Verification | Points |
|---|---|---|---|
| 1 | Detect user intent with at least 25% accuracy | Test with a diverse set of input queries and verify the accuracy of intent detection. | 2 |
| 1 | Detect user intent with at least 50% accuracy | Continue testing and refining to achieve higher accuracy. | 2 |
| 1 | Generate response within 60 seconds | Measure retrieval time with various queries to ensure performance within the initial time limit. | 2 |
| 1 | Generate response within 40 seconds | Measure retrieval time with various queries to ensure performance within the initial time limit. | 2 |
| 1 | Generate response within 30 seconds | Optimize system to improve performance and meet the final time requirement. | 1 |
| 1 | Can fetch correct external material with 20% accuracy | Testing the agent's RAG accuracy by using manually labeled dataset containing 30 questions for 4 testing locations. | 1 |
| 1 | Can fetch correct external material with 50% accuracy | Testing the agent's RAG accuracy by using manually labeled dataset containing 30 questions for 4 testing locations. | 2 |

Continued on next page

**Table 9 continued from previous page**

| S-Index | Requirement | Verification | Points |
|---|---|---|---|
| 1 | Can fetch correct external material with 70% accuracy | Testing the agent's RAG accuracy by using manually labeled dataset containing 30 questions for 4 testing locations. | 1 |
| 1 | Generated answers must match the user's intent with an accuracy of at least 25% in given dataset | Compare generated answers from the dataset containing 30 questions with human labeled answer to calculate initial accuracy. | 3 |
| 1 | Generated answers must match the user's intent with an accuracy of at least 50% in given dataset | Compare generated answers from the dataset containing 30 questions with human labeled answer to calculate initial accuracy. | 3 |
| 2 | The web server must handle and route messages with less than 2 seconds latency | Test message routing on the server under load and measure latency | 2 |
| 2 | The client interface must provide intuitive access for users to submit queries and control the UAV | Conduct usability testing with participants to assess ease of use and intuitiveness | 1 |
| 2 | UAV command buttons must send correct instructions to the UAV subsystem with 100% accuracy | Test each button and verify that the correct command is sent to the UAV subsystem | 1 |
| 2 | The web server must send instructions and questions separately to different hosts | Test two hosts if they receive correct messages | 1 |
| 2 | Obtain the user's GPS position as the starting position | Test if the two hosts receive the GPS signal | 1 |
| 2 | The host which process the questions can display answers correctly | Check if the UI can display reasonable answers | 2 |
| 2 | The host which process the instructions can send commands to UAV | UAV can take-off, Stop, Continue, Land correctly and in time | 2 |
| 3 | Must accurately process user commands and drone status within 10 second | Perform stress testing with simultaneous user commands and verify response time | 3 |

Continued on next page

**Table 9 continued from previous page**

| S-Index | Requirement | Verification | Points |
|---|---|---|---|
| 3 | Must accurately process user commands and drone status within 1 second | Perform stress testing with simultaneous user commands and verify response time | 3 |
| 3 | Must optimize the UAV route based on the current status | Test with different scenarios (no-fly zones, different areas) to verify route optimization | 3 |
| 3 | Should maintain a secure and encrypted connection to the remote server | Verify the encryption standards and conduct penetration testing to assess security | 1 |
| 3 | Must integrate seamlessly with the PX4 APIs for flight control | Execute a series of flight tests to ensure proper integration and control | 1 |
| 4 | Power supply successfully power the hardware unit | Power supply LED works correctly | 1 |
| 4 | Sensors can operate properly | Connect the sensors to the test board Use oscilloscope to read the output data. | 2 |
| 4 | LCD screen can display normally | Connect the LCD screen to the test board Provide appropriate power. Send test data to the LCD screen Check if the screen displays correctly. | 2 |
| 1,2 3,4 | End to End works correctly | Can complete a guide for appointed locations while iteracting with visitors | 5 |