# Software Requirements Specification

**for**

# AQI Drone Project

**Version 1.0 approved**

**Prepared by Pavel, Fheng, Justin, Sebastian**

**UMN SASE Labs**

**09/25/2025**

# Table of Contents



# Revision History

| Name | Date | Reason For Changes | Version |
|---|---|---|---|
| | | | |

| | | | |
|---|---|---|---|
| | | | |

# 1. Introduction

## 1.1 Purpose

The purpose of this document is to track the requirements and specifications in the UMN SASE Labs AQI Drone project. It will explain the purpose and the features in this product and its software, the interfaces, what the product will do, and the constraints under which this product will operate. It will directly communicate with the engineering team and its users.

## 1.2 Document Conventions

This Document was created based on the IEEE template for System Requirement Specification Documents.

## 1.3 Intended Audience and Reading Suggestions

- Typical Users, who want to use this product in order to track the AQI of a given area
- Developers/Engineers, who are interested in working on the project by further developing it, fix existing bugs, or further develop its physical features
- Observers, who are interested in this product and wish to gain further insight about it

## 1.4 Product Scope

The AQI drone is a tool that can be used to determine the AQI of a certain area. Users can use it in order to obtain the average AQI of the area the drone is in. They are able to control the drone using a remote control within the Arduino App. The AQI metrics are then sent to a dashboard to be displayed.

## 1.5 References

AQI Drone Github:
https://github.com/Ptokarev74/SASE_AQI_Drone

Running Material List:
https://docs.google.com/spreadsheets/d/1LNIGE9SEOWnPPK-BiME-JE5M2TTcGpd6oZIThqegmDU/edit?pli=1&gid=0#gid=0

# 2. Overall Description

## 2.1 Product Perspective

The AQI Drone is being developed for everyone to use to gain better insights into their local air quality. It is a project being developed by SASE Labs, an active engineering team. It runs on C and various embedded systems.

Its physical scheme is shown below:
[Insert schematic of drone]

## 2.2 Material List

The following is a list of the materials required for this project:

1. AQI Sensor
2. 1 Gyroscope
3. 2 Inertial Measurement Unit
4. 5 Capicators
5. 1 pack of 4mm screws
6. 1 pack of 6mm screws
7. 5 Resistors
8. PCB Casing

More details can be viewed here.

## 2.3 Product Functions

Physical Functions:

1. Take off from the ground
    a. Take off from the ground. Takes into account ground effect, initial flight stabilization
        i. Hardware
            1. UAV Frame, Motors, Casing
        ii. Electrical
            1. Motor Controller, Flight controller
        iii. Software
            1. Software boot, Automated Takeoff

2. Reach and stabilize at a specific height
    a. Reach and hover around the desired height
        i. Hardware
            1. UAV Frame, Motors, Casing, Gyroscope
        ii. Electrical
            1. Motor Controller, Flight Controller, Altitude Sensor
        iii. Software
            1. Altitude Sensor reading, Motor speed control, Altitude Control (PID)

3. Sample Air Quality
    a. Read from the air quality sensor for 5 seconds, analyze, and average the data for measurement
        i. Hardware
            1. Air quality sensor mount
        ii. Electrical
            1. Air quality sensor
        iii. Software
            1. AQT sensor reading, Data analysis

4. Land on the ground
    a. Land UAV from at most 10 meters
        i. Hardware
            1. UAV Frame, Motors, Casing
        ii. Electrical
            1. Motor Controller, Flight Controller, Altitude Sensor
        iii. Software
            1. Altitude Sensor reading, Motor speed control, Altitude control (PID), Propeller shutoff

5. Emergency shut off
    a. Automatic/wireless remote failsafe in case of emergency
        i. Hardware
            1. Receiver mount, Ground station/controller
        ii. Electrical
            1. Receiver
        iii. Software
            1. Receiver sensor reading, Failsafe detection

6. Emergency Teleop Control
    a. In case of testing/emergency
        i. Hardware
            1. Receiver Mount, Ground station/controller
        ii. Electrical
            1. Receiver
        iii. Software
            1. Receiver sensor reading, Reading Receiver Data

Software Functions:

1. A dashboard that displays different data from the drone:

    a. Air Quality

    b. Drone Height

    c. Drone Location

## 2.4 User Classes and Characteristics

- Typical Users, who want to use this product in order to track the AQI of a given area

- Developers/Engineers, who are interested in working on the project by further developing it, fix existing bugs, or further develop its physical features

## 2.5 Operating Environment

Physical Operating Environment:
- The drone must be operated in clear weather, free from strong winds

Software Operating Environment:
- The software must run in an embedded environment
- Windows & Mac

## 2.6 Design and Implementation Constraints

- Initial funding of $200, materials purchased must not exceed this amount.
- This project must be finished by the end of April to be presented at the SASE Labs Symposium.

## 2.7 User Documentation

No external user documentation is provided.

## 2.8 Assumptions and Dependencies

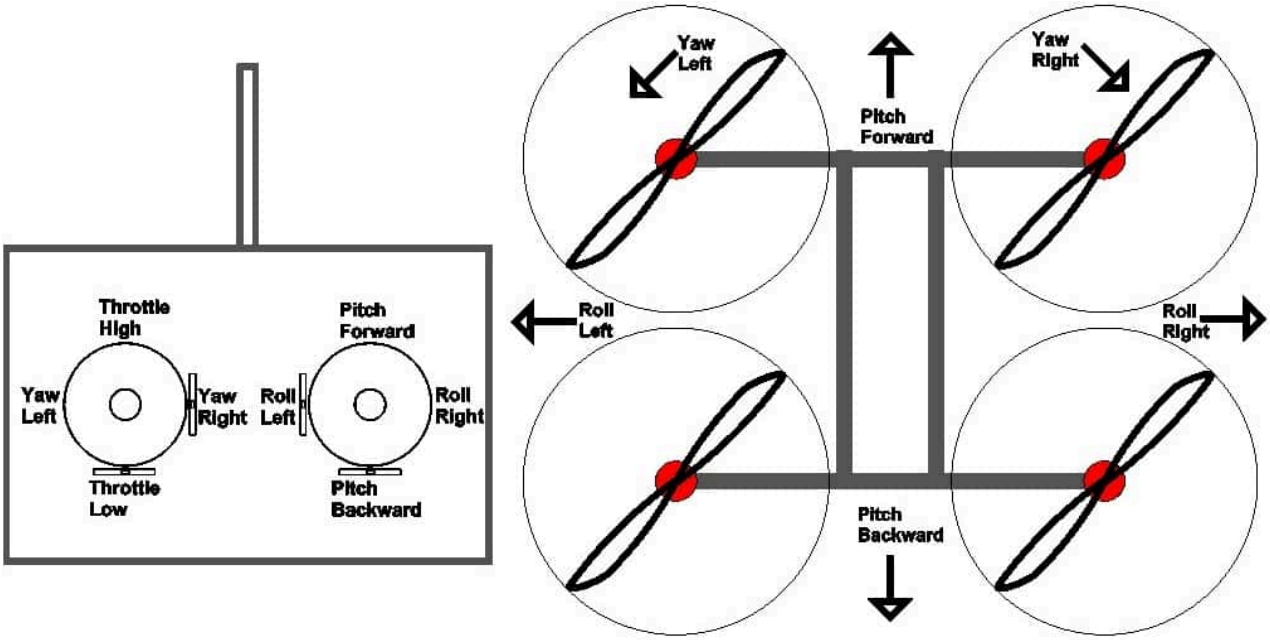
<List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also, identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project, unless they are already documented elsewhere (for example, in the vision and scope document or the project plan).>

# 3. External Interface Requirements

## 3.1 User Interfaces

*<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>*

*Users will interact with the Drone using Arduino Bluetooth app.*




- *Between the yokes will be a display that will log different positions and states of the Drone in an X, Y, Z position*
    - *Z position can be static for maintaining altitude*

## 3.2 Hardware Interfaces

*<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>*

-

## 3.3 Software Interfaces

*<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>*

- *Arduino App to run program to fly*
    - *Can interface API calls e.g. fly up, fly down, left, right*
    - *Custom scripts*

## 3.4 Communications Interfaces

*<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>*

- *I2C, UART*

# 4. System Features

*<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>*

## 4.1 System Feature 1

*<Don't really say "System Feature 1." State the feature name in just a few words.>*

### 4.1.1 Description and Priority

*<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>*

#### 4.1.2 Stimulus/Response Sequences

*<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>*

#### 4.1.3 Functional Requirements

*<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use "TBD" as a placeholder to indicate when necessary information is not yet available.>*

*<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>*

REQ-1:
REQ-2:

### 4.2 System Feature 2 (and so on)

# 5. Other Nonfunctional Requirements

## 5.1 Performance Requirements

*<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>*

## 5.2 Safety Requirements

*<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product's design or use. Define any safety certifications that must be satisfied.>*

## 5.3 Security Requirements

*<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication*

*requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.>*

### 5.4 Software Quality Attributes

*<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>*

### 5.5 Business Rules

*<List any operating principles about the product, such as which individuals or roles can perform which functions under specific circumstances. These are not functional requirements in themselves, but they may imply certain functional requirements to enforce the rules.>*

## 6. Other Requirements

*<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>*

## Appendix A: Glossary

*<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>*

## Appendix B: Analysis Models

*<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>*

## Appendix C: To Be Determined List

*<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>*

**Use Case Template**

| | |
|---|---|
| **Name** | |
| **ID** | |
| **Description** | |
| **Actors** | |
| **Organizational** | |

| | |
|---|---|
| **Benefits** | |
| **Frequency of Use** | |
| **Triggers** | |
| **Preconditions** | |
| **Postconditions** | |
| **Main Course** | |
| **Alternate Course** | |
| **Exceptions** | |

| **Name** | **System Startup** |
|---|---|
| **ID** | UC_001 |
| **Description** | When the program is run, a seat count and the type of election must be specified. On start-up, the user must be asked in an interface to input how many seats there are to be filled and the algorithm to use to determine the election winners. |
| **Actors** | System User |
| **Organizational Benefits** | Allows the system to be functional in running elections. |

| | |
|---|---|
| **Frequency of Use** | It must be used by 100% of the users, 100% of the time, for every election. |
| **Triggers** | The user starts up the system. |
| **Preconditions** | The user must have determined how many open seats there are for an election.<br>The user must have determined which of the two valid election types took place.<br>There must exist a ballot file of proper format (See section 6.1) in the directory |
| **Postconditions** | The number of open sets is now selected<br>The election type is specified<br>The ballot file(s) name is specified |
| **Main Course** | 1. User starts the system<br>2. User is prompted for the number of open seats in the interface<br>3. User inputs the total number of seats (see EX1) into the interface<br>4. User is prompted for the type of election in the interface<br>5. User inputs the election type (see EX2) into the interface.<br>6. User is directed towards ballot counting initiation<br>7. User is prompted for the ballot file(s) name in the interface<br>8. User inputs the ballot name(s) (see EX3) into the interface.<br>9. Continue to UC_002 |
| **Alternate Course** | AC1 User prompts for help in the interface<br>1. Direct user to UC_011 (Help Window) |
| **Exceptions** | EX1 A non-integer was the input<br>1. User is notified to input an integer<br>2. User is directed back to Main Course step 2<br>EX2 A non-valid election type is the input<br>1. User is notified that the input was not a valid option<br>2. User is directed back to Main Course step 4<br>EX3 A non-valid file is the input<br>1. User is notified that the file name is wrong<br>2. User is directed back to Main Course Step 7 |