

---

# **Software Requirements Specification**

**for**

## **AQI Drone Project**

**Version 1.0 approved**

**Prepared by Pavel, Feng, Justin, Sebastian**

**UMN SASE Labs**

**09/25/2025**

# Table of Contents

<b>Table of Contents</b>	<b>ii</b>
<b>Revision History</b>	<b>ii</b>
<b>1. Introduction</b>	<b>1</b>
1.1 Purpose	1
1.2 Document Conventions	1
1.3 Intended Audience and Reading Suggestions	1
1.4 Product Scope	1
1.5 References	1
<b>2. Overall Description</b>	<b>2</b>
2.1 Product Perspective	2
2.2 Product Functions	2
2.3 User Classes and Characteristics	2
2.4 Operating Environment	2
2.5 Design and Implementation Constraints	2
2.6 User Documentation	2
2.7 Assumptions and Dependencies	3
<b>3. External Interface Requirements</b>	<b>3</b>
3.1 User Interfaces	3
3.2 Hardware Interfaces	3
3.3 Software Interfaces	3
3.4 Communications Interfaces	3
<b>4. System Features</b>	<b>4</b>
4.1 System Feature 1	4
4.2 System Feature 2 (and so on)	4
<b>5. Other Nonfunctional Requirements</b>	<b>4</b>
5.1 Performance Requirements	4
5.2 Safety Requirements	5
5.3 Security Requirements	5
5.4 Software Quality Attributes	5
5.5 Business Rules	5
<b>6. Other Requirements</b>	<b>5</b>
<b>Appendix A: Glossary</b>	<b>5</b>
<b>Appendix B: Analysis Models</b>	<b>5</b>
<b>Appendix C: To Be Determined List</b>	<b>6</b>

## 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](https://github.com/Ptokarev74/SASE_AQI_Drone)

Running Material List:

<https://docs.google.com/spreadsheets/d/1LNIGE9SEOWnPPK-BiME-JE5M2TTcGpd6oZIThhegmDU/edit?pli=1&gid=0#gid=0>

## 2. Overall Description

### 2.1 Product Perspective

The AQI Drone is a self-contained environmental monitoring system being developed by the UMN SASE Labs. It is engineered as a mobile data-gathering platform to address the primary limitation of traditional air quality monitoring: fixed, ground-level sensor locations. By integrating an Air Quality Index (AQI) sensor onto an unmanned aerial vehicle (UAV), the project provides a solution for capturing local air quality data that is otherwise difficult to obtain.

The product is a complete, standalone system composed of two principal subsystems: the Aerial Platform and the Ground Control Interface.

The base is the physical drone itself. It is a mechanical system that operates on a C++-based embedded software environment. Its primary function is to transport the sensor, navigate to specified points, and maintain flight stability. Key onboard hardware includes a central flight controller, a gyroscope and Inertial Measurement Unit (IMU) for orientation and stabilization, and the essential AQI sensor for data collection.

The Ground Control Interface comprises the software components that enable user interaction. This includes an Arduino-based mobile application for real-time remote piloting and a data visualization dashboard. The dashboard displays critical flight and environmental data transmitted from the drone, such as: current AQI levels, flight altitude, and geographical coordinates.

This system is designed as an accessible, open platform intended for educational use, further academic research, and for environmental enthusiasts. The drone operates independently in the air but depends on the ground control interface for commands and for relaying its findings back to the user. Communication between the drone and the user's device is handled via a low-latency communications protocol.

### 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 Capacitors
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,** A non-technical user whose primary goal is to operate the drone via the mobile application to monitor the Air Quality Index (AQI) of a specific outdoor area

**Developers/Engineers,** A technical user responsible for the ongoing development of the project, including debugging software, fixing hardware issues, and implementing new physical or software features.

## 2.5 Operating Environment

**Physical Environment:** The drone is designed for outdoor operation in clear weather conditions, free from strong winds or precipitation.

**Software Environment:** The drone's onboard systems run in an embedded C environment. The user control software is compatible with desktop operating systems, including Windows, Linux and macOS.

## 2.6 Design and Implementation Constraints

**Schedule Constraints:** The project is subject to a hard deadline. A fully functional and presentable prototype must be completed by the end of April 2026 to be showcased at the annual UMN SASE Labs Symposium. This fixed timeline necessitates a focused development scope and efficient project management.

**Budgetary Constraints:** There is a total budget of \$200 allocated for all physical materials. This includes the UAV frame, all electronic components, sensors, and any necessary hardware. This constraint requires careful component selection to balance cost against performance and reliability.

**Platform:** The system's onboard software must be developed using the Arduino platform and its associated toolchain. This limits the choice of microcontrollers and available processing power.

**Programming Language:** The embedded software for the flight controller must be written in C or C++, as specified in the project outline.

**Communication:** The interface between the drone and the ground control station will rely on standard communication protocols like I2C and UART, as mentioned in the interface requirements.

**Regulatory and Safety Constraints:** The project must adhere to all relevant regulations for operating unmanned aerial vehicles. As the project is based in the United States, all flight tests must comply with FAA (Federal Aviation Administration) guidelines for recreational or educational drones, including maintaining a visual line of sight and avoiding restricted airspace.

## 2.7 User Documentation

No external user documentation is or will be provided.

## 2.8 Assumptions and Dependencies

**Third-Party Software:** The project's development is dependent on the continued availability and stability of the Arduino IDE and its associated libraries.

**Wireless Environment:** The drone's remote control functionality is dependent on a stable wireless connection (e.g., Bluetooth). The performance can be affected by radio interference in the operating environment.

# 3. External Interface Requirements

## 3.1 User Interfaces

The main screen for piloting the drone and managing its state. Its design will be modeled after standard drone control interfaces to ensure a familiar user experience.

- **Virtual Joysticks:** Two on-screen virtual joysticks (thumbsticks) will provide real-time manual control over the drone's movement:
  - **Left Stick:** Controls throttle (altitude) and yaw (rotation).
  - **Right Stick:** Controls pitch (forward/backward) and roll (left/right).
- **Core Action Buttons:** A clear, accessible panel will feature buttons for critical flight actions:
  - ARM DRONE / DISARM DRONE
  - START AQI MISSION
  - LAND
  - EMERGENCY MOTOR CUTOFF (for safety).

- **Status Display:** A persistent header or footer will display essential telemetry at all times, including drone battery level, connection strength, and current flight mode (e.g., Manual, Mission, Landing).

A secondary screen is dedicated to displaying the environmental data collected by the drone in a clean, easy-to-read format.

- **AQI Readout:** A large gauge or numerical display will show the primary Air Quality Index reading in real-time.
- **Positional Data:**
  - **Altitude (Z-axis):** A numerical indicator will display the drone's current height above the ground.
  - **Location (X/Y-axis):** A simple map view or coordinate display will show the drone's geographical position.
- **Flight Vitals:** Additional readouts will display secondary information such as total flight time and sensor status.

## 3.2 Hardware Interfaces

**Motor Control (via ESCs):** The flight controller software commands the four Electronic Speed Controllers (ESCs) to adjust motor speed. This interface uses Pulse Width Modulation (PWM) signals, where the duration of the electrical pulse sent from the controller's output pins dictates the rotational speed of each motor.

**Inertial Measurement Unit (IMU):** The software continuously reads orientation and acceleration data (e.g., pitch, roll, yaw) from the onboard IMU. This critical data is used for flight stabilization. The communication occurs over a two-wire I2C (Inter-Integrated Circuit) serial bus.

**AQI Sensor:** To perform its primary function, the software interfaces with the Air Quality Index sensor to retrieve environmental data. Depending on the final sensor model selected, this connection will use either an I2C or UART (Universal Asynchronous Receiver-Transmitter) serial communication protocol.

**Radio Receiver / Bluetooth Module:** The flight controller receives commands from the user's GCS mobile application via the onboard radio receiver or Bluetooth module. This interface uses a UART serial connection to transmit data packets containing the user's control inputs (e.g., desired throttle, pitch, and roll).

**Battery Voltage Monitor:** To monitor the drone's remaining power and ensure a safe landing, the software reads the main battery's voltage. This is achieved through a direct connection from the power distribution board to one of the flight controller's analog input pins, allowing the Analog-to-Digital Converter (ADC) to measure the voltage level.

## 3.3 Software Interfaces

**Arduino IDE:** The primary development environment for the drone's onboard software is the **Arduino Integrated Development Environment (IDE)**. All custom C/C++ scripts (firmware) are written, compiled, and uploaded to the flight controller's microcontroller using this tool.

**Third-Party Libraries:** The firmware depends on several external, open-source libraries to communicate with hardware components. These libraries provide a pre-built Application Programming Interface (API) that simplifies tasks like reading sensor data. Key examples include:

- An **IMU-specific library** (e.g., for the MPU-6050) to process gyroscope and accelerometer data.
- An **AQI sensor library** to handle communication and data interpretation from the air quality sensor.
- A **communication library** (e.g., `SoftwareSerial`) to manage the UART interface with the Bluetooth module.

**Custom Communication Protocol:** The GCS mobile application and the drone's firmware communicate using a custom-defined serial protocol sent over Bluetooth. The GCS application sends command packets, and the drone sends back telemetry and data packets. This acts as a simple API for controlling the drone's flight.

- **Example Outgoing Commands (GCS to Drone):** `ARM_DRONE`, `SET_THROTTLE(value)`, `LAND`.
- **Example Incoming Data (Drone to GCS):** `SEND_AQI_DATA`, `SEND_ALTITUDE`, `SEND_BATTERY_VOLTAGE`.

**Operating System API:** The user-facing GCS application interfaces with the host operating system's (OS) native APIs. This is necessary to render the graphical user interface (GUI) on screen and to access the device's Bluetooth hardware for communication with the drone.

-

## 3.4 Communications Interfaces

**2C (Inter-Integrated Circuit):** This is a two-wire serial bus used to connect the flight controller to multiple onboard sensors simultaneously. In this project, the I2C bus is the primary interface for reading data from the Inertial Measurement Unit (IMU) and may also be used for the AQI sensor.

**UART (Universal Asynchronous Receiver-Transmitter):** This is a point-to-point serial protocol used for communication with components that require a dedicated connection. The primary use of UART is to interface the flight controller with the onboard Bluetooth module, enabling the drone to send and receive data wirelessly.

**Bluetooth:** The drone uses a Bluetooth Classic or Low Energy (BLE) module to establish a wireless link with the user's mobile device. This connection is used to transmit all command and control signals from the user to the drone and to send all telemetry and sensor data (like AQI readings and altitude) from the drone back to the user.

- **Range:** The operational range is limited by the Bluetooth standard, typically up to **30 meters (approx. 100 feet)** in an open environment.

## 4. System Features

### 4.1 Feature: Flight Control and Stabilization

**4.1.1 Description and Priority** This feature includes all functions related to the drone's ability to fly in a stable and controllable manner, both manually and through automated sequences. It is the foundational feature upon which all others depend.

#### 4.1.2 Stimulus/Response Sequences

**Stimulus:** User provides "throttle up" input via the GCS application.

**Response:** The system increases power to all motors, causing the drone to gain altitude.

**Stimulus:** The drone is tilted by an external force, like a gust of wind.

**Response:** The system's flight controller detects the unintended change in orientation via the IMU and automatically adjusts the speed of individual motors to return the drone to a stable, level state.

**Stimulus:** User presses the EMERGENCY MOTOR CUTOFF button.

**Response:** The system immediately ceases power to all motors.

#### 4.1.3 Functional Requirements

**REQ-FC-01:** The system shall process user commands from the GCS app to control the drone's pitch, roll, yaw, and throttle.

**REQ-FC-02:** The system shall use data from the IMU to actively stabilize its flight orientation.

**REQ-FC-03:** The system shall require a specific "arming" command from the user before the motors can be powered for takeoff.

**REQ-FC-04:** The system shall be able to execute an automated landing sequence upon user command.

**REQ-FC-05:** The system shall provide an immediate motor cutoff function for emergency situations.

### 4.2 Feature: Environmental Data Collection

**4.2.1 Description and Priority** This feature covers the drone's primary mission: activating the AQI sensor, collecting a valid air quality sample, and processing the data.

#### 4.2.2 Stimulus/Response Sequences

**Stimulus:** User presses the START AQI MISSION button on the GCS app.

**Response:** The system commands the AQI sensor to begin sampling, collects data for a set duration, calculates the result, and transmits it to the GCS.

#### 4.2.3 Functional Requirements

**REQ-DC-01:** The system shall be able to power on and initialize the AQI sensor via a command from the GCS app.

**REQ-DC-02:** The system shall collect sensor readings over a predefined sampling period (e.g., 5 seconds).

**REQ-DC-03:** The system shall calculate an average value from the readings collected during the sampling period.

**REQ-DC-04:** The system shall transmit the final calculated AQI value to the GCS application for display.

### 4.3 Feature: Telemetry and Data Visualization

**4.3.1 Description and Priority** This feature involves the transmission of real-time flight data (telemetry) and environmental data from the drone to the GCS app for display to the user.

#### 4.3.2 Stimulus/Response Sequences

**Stimulus:** The drone's battery voltage drops below a predefined critical level.

**Response:** The drone transmits the low voltage data to the GCS app; the GCS app displays a prominent "LOW BATTERY" warning to the user.

#### 4.3.3 Functional Requirements

**REQ-DV-01:** The drone shall periodically transmit its battery voltage level to the GCS application.

**REQ-DV-02:** The drone shall transmit its current altitude to the GCS application.

**REQ-DV-03:** The GCS application shall display the received AQI data in a clear, graphical format on its data dashboard.

**REQ-DV-04:** The GCS application shall display the received flight telemetry (battery, altitude, connection status) in a persistent status display.

**REQ-DV-05:** The GCS application shall present a visual alert to the user when the drone's battery level is critically low.

## 5. Other Nonfunctional Requirements

### 5.1 Performance Requirements

These requirements define how the system must perform under specific conditions.

- **Flight Duration:** The drone **shall** be capable of a minimum continuous flight time of **5 minutes** on a fully charged battery.
- **Control Latency:** The end-to-end latency between a user's command on the GCS application and the drone's physical response **shall** be less than 200 milliseconds.
- **Telemetry Update Rate:** The GCS application **shall** receive and display telemetry updates (e.g., battery voltage, altitude) from the drone at a minimum rate of 2 Hz.

### 5.2 Safety Requirements

These requirements are concerned with preventing harm, loss, or damage resulting from the drone's operation.

- **Loss-of-Communication Failsafe:** In the event that the wireless communication link is lost for more than 3 consecutive seconds, the drone **shall** automatically initiate a controlled landing sequence at its current position.
- **Low-Battery Failsafe:** When the battery voltage drops to a predefined critical level, the drone **shall** automatically initiate a controlled landing sequence.
- **Motor Arming Protocol:** The system **shall** require an explicit "arming" sequence from the user before the propellers can spin. The system **shall** prevent arming if it detects that it is not in a stable, level orientation.
- **Physical Safeguards:** The drone **shall** be equipped with propeller guards to minimize the risk of injury or damage from contact with the spinning propellers.

### 5.3 Security Requirements

Currently there are no security requirements

### 5.4 Software Quality Attributes

These attributes define the desired quality characteristics of the software.

1. **Reliability:** The flight controller firmware **shall** be designed to operate without resets or critical failures during any single flight session.
2. **Maintainability:** The C/C++ source code **shall** be well-commented and adhere to a consistent coding style to allow for easy understanding and modification by future SASE Labs developers.
3. **Usability:** The GCS mobile application **shall** be designed to be intuitive, allowing a new user to learn basic flight and data collection operations in under 10 minutes.

## **5.5 Business Rules**

These are operational rules or policies that the project must adhere to.

- **Regulatory Compliance:** All flight operations **shall** be conducted in strict accordance with the current FAA (Federal Aviation Administration) regulations for recreational or educational drone use.
- **Intended Use:** This system is designed as an educational prototype and **shall** be used for academic and hobbyist purposes only. It is not certified for commercial or safety-critical applications

## **6. Other Requirements**