# GCS Software Design Document

## Data Design

The GCS will hold data relative to the state of the collision avoidance for all aircraft involved. This state will be developed as a data structure and displayed on the user interface.

State will be displayed as text.

## Architecture Design

Current Work?

* Shall display collision avoidance activity.
* Shall ensure reliable communication.
* Shall have a communication specified defined for reliable communication and collision avoidance.

### Collision Avoidance

Current Work?

Collision Avoidance is dictated by messages passed from the UAV to GCS. The GCS probes for collision avoidance through heartbeats. The messages are one byte in size and sent over XBEE transceivers through a serial connection. The message will be a boolean value and will dictate whether to report CA activity. Communication shall be reliable via serial communication with an active heartbeat coming from the UAV.

## Interface Design

A command-line and graphical interface will be developed for the GCS.

### Command-Line Interface (CLI)

A command-line interface (CLI) will be initially developed for reporting collision avoidance data. This will occur during the first milestone. It shall be developed in the C++ Programming Language using the Qt toolkit. The data will be read through a reliable serial connection. This interface is primarily to be used for debugging and proof of concept.

Qt was chosen for developing the CLI to keep consistency with the GUI that will be developed in milestone 2.

The CLI will support Windows, but capable of running on any Qt-supported platform. For the purpose of this task, the team has decided that focusing on one platform will allow for more streamlined development going into this year’s project demo

### Graphical User Interface (GUI)

A graphical user interface (GUI) will be developed during milestone 2. The GUI will display collision avoidance data, as well as attitude and altitude data coming from connected vehicles.

The GUI will be developed in the C++ programming language using the Qt toolkit. It shall utilize the Model-View-Controller architectural pattern. This will allow us to separate our data, logic, and design. The logic and data layers will be tied together though object models produced through object relational mapping.

## Data

The GCS will use the SQlite database system. This was chosen due to its server-less self-contained nature. Object relational mapping will be accomplished using QxOrm.

## Procedural Design

Current Work?

Spiral development.

Use unit tests for regression testing.

Port from console to GUI.

## in