AeroQuad Flight Software

Developer’s Guide

# Introduction

The AeroQuad Flight Software uses a mixture of C/C++ to accommodate multiple hardware options and algorithms within the Arduino platform. The motivation behind using a mixture of C/C++ is to find the right balance between flexibility and commonality within the restrictive programming space and processing speed of a microcontroller environment. This also allows the end user to have a methodical way to add new hardware capabilities with minimal impact to the existing flight algorithms. As a result upgrading to new improved hardware sensors can be done easily and a clear defined method for users to customize and improve the flight software itself can be achieved. The purpose of this guide is to document the software architecture implemented, provide a description of what each component does, and to give a guideline on how to contribute new features for the future.

The software architecture documented here is considered a work in progress. With the possibility of larger and faster processors available in the future and with the general understanding and experience in using Object Oriented programming in the AeroQuad community, a more comprehensive C++ architecture can be developed. This current architecture is considered a learning step in this direction.

The AeroQuad Flight Software is provided as an open source project written within the Arduino development environment. Where possible it uses the Arduino libraries, but may rely on low level ATmega microcontroller programming for optimization of certain functions.

# Software Architecture

Each of the main functions depicted in dark blue in Figure 1 are kept in Arduino sketches (.pde files). Class definitions are placed in header files (.h) and are depicted in light blue. Header files will typically have the main class (for example Gyro) at the top of the header file and all available subclasses (such as Gyro:Gyro\_AeroQuad\_v2.0) will be listed underneath it. The following sections will describe each main function and any supporting header files needed.

## AeroQuad.pde

The AeroQuad.pde sketch contains the setup and main loop of the flight software. It’s primary responsibility is to maintain the timing that each of the mail functions are to execute at.

## SerialCom.pde

This sketch receives external serial commands and responds to telemetry requests.

## AeroQuadFlightSoftwareArchitecturev2.0.png

Figure - Software Architecture

## FlightCommand.pde

FlightCommand.pde is responsible for decoding transmitter stick combinations and for setting up AeroQuad modes such as motor arming/disarming and Acro/Stable flight modes. The main function relies on Receiver.h for receiving radio controlled signals from the pilot. Future classes are planned to receive pilot commands over wireless link from a laptop or mobile device.

## Sensors.pde

The Sensors.pde sketch is responsible for taking on-board sensor measurements and calculating flight attitude. The following header files are used by Sensors.pde:

* DataAcquisition.h – container for certain hardware configurations where a common sensor measurement call is needed. Examples of this are I2C communication for Wii sensors, SPI ADC communication for the APM and APM IMU Sensor.
* Filter.h – contains a simple low pass filter to remove noise from sensor measurements
* Accel.h – defines how to measure accelerometer data and convert to engineering units
* Gyro.h – defines how to measure gyro data and convert to engineering units
* GPS.h – defines how to communicate decode NMEA strings from a GPS
* FlightAngle.h – contains multiple algorithms to calculate multicopter vehicle attitude
* Compass.h – defines how to measure magnetometer or similar sensor data
* Distance.h – defines how to measure sonar or IR data
* Altitude.h – defines how to measure barometer or similar sensor data

## FlightControl.pde

This sketch combines sensor measurements and transmitter commands into motor commands for the defined flight configuration (X, +, etc.). The following header files are used by FlightControl.pde

* PID.h – contains the PID implementation used for control of the multicopter
* Motors.h – defines which motor control is used (for example PWM or I2C).

# Class Definitions

This section will describe each class defined in the AeroQuad Flight Software. Any new subclasses must conform to the methods (or function calls) defined in the main class. New methods required by the subclass will be defined in the subclass itself. The header files are listed in alphabetical order below.

## Accel.h

Under Construction

## Altitude.h

Under Construction

## Compass.h

Under Construction

## DataAcquisition.h

Under Construction

## Distance.h

Under Construction

## Filter.h

Under Construction

## FlightAngle.h

Under Construction

## GPS.h

Under Construction

## Gyro.h

Under Construction

## Motors.h

Under Construction

## PID.h

Under Construction

## Receiver.h

Under Construction

# Customizing Code

Under Construction

## How to Add a New Class

Under Construction

## How to Add a New Sub Class

Under Construction