Software/Hardware Requirements Specification

for

Anzhelka

Version 1.0 approved

Prepared by
Luke De Ruyter
Cody Lewis

University of California, Riverside

June 12, 2012

Table of Contents

Revisions

Current project status and files can be found at

blog.anzhelka.com
code.anzhelka.com

Version	Date	Changes	Commiter
0.01	April 30, 2012	Initial layout of file was created.	Cody
0.02	May 4, 2012	Purpose, Audience, Scope, Perspective, Functions, and Users was populated.	Cody, Luke
1.0	June 12, 2012	Typos removed, some sections grammatically modified.	Cody, Luke

1 Introduction

1.1 Purpose

Anzhelka is a complete system intended for autonomous quadrotor flight. The Anzhelka system includes both hardware and software such as the quadrotor frame, control electronics, ground station software, and the complete system documentation. Anzhelka is completely open source, and all project files are available for download.

1.2 Document Conventions

N/A

1.3 Intended Audience and Reading Suggestions

This document is written for Anzhelka developers. This document is intended to refine development direction, and to bring new developers up to speed. For this document, developers include software writers, hardware designers, and system testers. When the word "you" is written, it is referring to the developer who is reading this document. When the words "the user" is written, it is referring to the end user of the Anzhelka platform.

You should read this document based on your background with Anzhelka. Current developers can skip to the appropriate section to read. New developers should read the introduction, overall description, and system features sections. If you are not a developer for this project, and don't intend to ever become one, you should avoid this document. Look on the Anzhelka website for something more appropriate to your needs.

1.4 Product Scope

Anzhelka consists of four main components: a quadrotor hardware frame, custom quadrotor software, ground station software, and detailed documentation via the Anzhelka website. Even without a degree in control systems, you can use Anzhelka components to make an autonomous quadrotor system. By using these components you can customize the functionality of the system to suit your needs, or use them directly to perform predefined commands.

1.5 References

N/A

Figure 1: An artistic rendering of the frame used for Anzhelka.

2 Overall Description

2.1 Product Perspective

Anzhelka is a platform for experimentation in aerial robotics, designed from the ground up as an independent project. We are taking an existing open hardware frame as our base hardware platform, and building on top of this hardware with control electronics. We have designed a custom control board that will be able to run all of Anzhelka's software and house all the necessary components. In addition, a custom ground station provides realtime flight data to the user.

2.2 Product Functions

Anzhelka is designed to be programmatically controllable in a number of different aspects. The quadrotor can receive commands to set any of it's internal registers, which then affect the flight of the quadrotor. In addition, the quadrotor has a command interpreter that is able to read command sequences from onboard memory. With these commands, the user can control a number of quadrotor functions including attitude and position.

Much of Anzhelka's functionality is due to the hardware being used. Anzhelka is equipped with a Parallax Propeller multicore processor. The quadrotor has two separate batteries: one battery provides power for the motors and servos; the other battery powers the electronics. The motors and servos have voltage and current monitoring. The Anzhelka system is designed with expandability in mind by allowing for the easy attachment of additional electronics.

2.3 User Classes and Characteristics

2.3.1 User Example: Jason

Figure 2: Jason Lariart

Age: 28
Education: Some College
Marital Status: Open
Occupation: Videographer
Hobbies: Photography, Videography, Art, Protesting
About Jason: He is an avid videographer and likes to push the limits of his skills, crew, and equipment. Jason likes to take pictures and videos of things that not just any ordinary person can do. He is currently an amateur photographer, but wants to make a professional career out of his work.

Scenario: Jason is preparing for a remote video shoot and realized that he won't be able to get his aerial shot because a helicopter is out of his budget. Jason told the client that he was going to have to spend more money if wanted to get the aerial shot as planned. The client was upset to hear about this and had no choice but to drop the scene from the shoot unless Jason can complete the shot for less money.

2.3.2 User Example: Dan

Figure 3: Dan Knuth

Age: 82
Education: Doctorate in Mathematics
Marital Status: Widowed
Occupation: Software Developer
Hobbies: Open source programming, online forum posting, sitting
About Dan: Dan works on open source systems. He has literally been around since the begining, and he remembers the days of punch card programming. Throughout his career he has been a strong supporter of making computer science more accessible, and his work in the open source field is both notable and well respected.

Scenario: Dan wants to spend more time focusing on his hobbies, but he doesn't have much time left and he needs to eliminate routine activities. Dan wants to develop a robot to automatically pick up his medicine from the pharmacy and carry it to him. This robot needs to be able to fly, and to be user programmable. Dan has lots of experience in working with software, so it's ok if the product is a bit rough and requires the use of a command line. Dan would be particularly appreciative if the instructions could be programmed in by a series of toggle switches and incandescent lights.

2.4 Operating Environment

N/A

2.5 Design and Implementation Constraints

N/A

2.6 User Documentation

N/A

2.7 Assumptions and Dependencies

N/A

3 External Interface Requirements

N/A

3.1 User Interfaces

N/A

3.2 Hardware Interfaces

N/A

3.3 Software or Hardware Interfaces

N/A

3.4 Communications Interfaces

N/A

4 System Features

This section describes Anzhelka features, and assigns priorities to each. Priorities are assigned in ascending order, with a priority of 1 being the most important.

4.1 Attitude Control

4.1.1 Description and Priority

An quadrotor must be able to maintain a stable attitude (orientation) at all times during flight. If the quadrotor loses control of it's attitude then catastrophic failure is certain to happen. Such a failure would likely result in the destruction of the vehicle, and possibly collateral damage. Therefore, attitude control is *the* top priority of Anzhelka.

Priority: 1

Business Requirement: Allow the user to have a precisely orientable platform.

4.1.2 Stimulus/Response Sequences

Stimulus	Response
1. Hit By High Velocity Paintballs	(a) Paintball hits, creates sudden attitude disturbance (b) Quadrotor loses altitude, increase in attitude error (c) Attittude mathematical control loop gains change to compensate (d) Quadrotor adjusts motor speed based on control loop result (e) Quadrotor achieves desired attitude

4.1.3 Functional Requirements

Attitude control assumes that the quadrotor has sufficient thrust in each motor to maintain a given attitude. In addition, the quadrotor must have a valid desired attitude to work towards. If there is a disturbance to the attitude the quadrotor must be able to correct it's orientation quickly before the position changes enough to risk collision.

4.2 Altitude Control

4.2.1 Description and Priority

Altitude control refers to the ability of the quadrotor to maintain a set height above ground level. This can be sensed through a number of different methods, but each must have the same result. Altitude control is important in order to prevent the quadrotor from crashing to the ground or going too high. Altitude control failure can result in damage to the vehicle or nearby objects, hence altitude control is very important.

Priority: 2

Business Requirement: Gives the user the ability to set a desired altitude for flight.

4.2.2 Stimulus/Response Sequences

Stimulus	Response
1. Altitude drops by 5cm	(a) Quadrotor loses 5cm in altitude (b) Altitude sensor on the quadrotor triggers an alarm within control loop (c) Altitude mathematical control loop gains change to compensate (d) Quadrotor adjusts motor speed based on control loop result (e) Quadrotor achieves desired altitude

4.2.3 Functional Requirements

Altitude hold assumes that the quadrotor can produce enough thrust to maintain the desired altitude. Maximum thrust may not be enough when a)the quadrotor is overloaded and too heavy or b)the quadrotor is flying at a severe angle. If either of these cases are true, then the quadrotor should be expected to lose altitude.

4.3 Highly Maneuverable

4.3.1 Description and Priority

One of the main advantages of an *autonomous* (vs. remotely operated) quadrotor is that the computer control should allow for aerial acrobatics. Most human pilots do not have suitable skill to perform complex movements. By allowing for complex manuevers, the quadrotor is able to fly in areas that it may not normally be able to.

This feature is an important reason to select autonomous flight, and hence an important part of the flight experience. It is not, however, necessary for stable flight.

Priority: 3

Business Requirement: Users have the ability to control precise aerobatics.

4.3.2 Stimulus/Response Sequences

Stimulus	Response
1. Quadrotor is programmed to do a barrel roll	(a) Quadrotor is instructed to perform a barrel roll (b) Current IMU and altitude readings are stored (c) The trajectory of the roll is calculated by the barrel roll function (d) Motors are changed to the desired speeds calculated by the control loop (e) Motor speeds are adjusted throughout the duration of the roll to maintain control (f) The quadrotor is then returned back to the starting attitude and altitude.

4.3.3 Functional Requirements

The quadrotor must be able fly at any specified orientation. The quadrotor is not expected to maintain position while in an acrobatic movement state. The system should be able to perform these movements with either a human pilot or through a programmable interface.

4.4 User Programmable

4.4.1 Description and Priority

The quadrotor must be able to accept commands from the user in a high level format. This allows for users to make the best use of the autonomous features of the quadrotor vehicle.

Priority: 3

Business Requirement: Makes the quadrotor unique to the users needs.

4.4.2 Stimulus/Response Sequences

Stimulus	Response
1. The user wants to program the quadrotor with a new trajectory plan	(a) The user plugs the SD card into a computer (b) The user launches the programming software (c) Trajectories are entered into the software (d) When the user is ready they can press the "Write to SD Card" Button (e) The user then plugs the SD card into the quadrotor

4.4.3 Functional Requirements

The quadrotor should be programmable in an easy to understand scripting language type format, in plain text. The user program should be able to be stored onboard the quadrotor, and edited on most computers. The scripting language developed should be able to support advanced features such as condition testing and branching. In addition, there should be a graphical user interface to allow for automated programming of the quadrotor.

4.5 Comprehensive Datalogging

4.5.1 Description and Priority

The quadrotor should document and log all of it's internal parameters and measurements for later review. This is especially important for debugging and assessing system stability. Every variable should be stored on a relatively secure and crash-safe memory device. This is important for debugging intermittent errors of the system, and hence is relatively important.

Priority: 3

Business Requirement: Know how the system performed and how to better achieve the desired actions.

4.5.2 Stimulus/Response Sequences

Stimulus	Response
1. The user wishes to log all flight data	(a) The user locates the logging switch on the control board (b) The user toggles the switch into the logging position (c) Recording begins once the quadrotor has been initialized

4.5.3 Functional Requirements

The data should be stored onboard the quadrotor, preferably in a removable medium such as an SD card. The data should also be stored in a plain text file on the medium in order to allow for easy data processing by a wide range of software.

4.6 Modularized

4.6.1 Description and Priority

The quadrotor must be able to be easily assembled and disassembled. This is especially important when the quadrotor is expected to crash frequently. However, as the project matures, development emphasis of the quadrotor frame can change to minimize the importance of modularity.

Priority: 4

Business Requirement: Interchangeable parts makes a reliable system.

4.6.2 Stimulus/Response Sequences

Stimulus	Response
1. The user wants to fix one of the motor booms	(a) The motor mount is removed from the damaged boom (b) The broken motor boom is then removed from the frame (c) A new replacement motor boom is mounted on the frame (d) The motor mount is reattached to the boom

4.6.3 Functional Requirements

The design of the quadrotor frame should use several techniques to maximize modularity. The quadrotor should use threaded fasteners for all joints. The use of glues and other adhesives should be minimized. All fasteners must use the same head type and thread sizes to minimize the number of different tools required. All components should be easy to manufacture or inexpensive to purchase, and be easily removable and replaceable when necessary.

4.7 Ground Station

4.7.1 Description and Priority

A ground station allows the quadrotor operator to view realtime statistics about the operation of the vehicle. Operational data is transmitted wirelessly to a receiver that is connected to a laptop computer for display. The data can include both measured data (such as motor speed or battery level) and calculated data (such as orientation error or next command). While useful and informative, this feature is not required for every flight.

Priority: 4

Business Requirement: Know what the quadrotor is doing while operating.

4.7.2 Stimulus/Response Sequences

Stimulus	Response
1. The user wants to get data from the quadrotor in real time	(a) The ground station software is initialized on a computer (b) The user sets the switch labeled "wireless" on the control board to on (c) The ground station hardware is then plugged into the USB port on the computer

4.7.3 Functional Requirements

The ground station should be self contained, and be available in one of two forms: a small USB device, or a tripod mounted and motorized device. The USB version of the ground station should be able to receive a signal from the quadrotor when it is operating in close proximity, and transmit packets to the quadrotor. The tripod mounted ground station must be able to track the quadrotor with a directional antenna, and must be powered from a USB port.

4.8 Waypoint Control

4.8.1 Description and Priority

The quadrotor should be able to follow a predetermined waypoint path in 3D space. This allows for precise control of quadrotor position, and can be used for tasks such as surveillance or transportation of payloads. The waypoints should be able to be defined both statically, before flight begins, and dynamically, after flight begins.

Priority: 4

Business Requirement: The system must be able to monitor an area continuously.

4.8.2 Stimulus/Response Sequences

Stimulus	Response
1. The user wants to control the quadrotor to fly in an continuous pattern	(a) While using the configuration software the user selects the waypoint tab (b) The user then selects the waypoint pattern (c) The waypoints are programmed onto the quadrotor.

4.8.3 Functional Requirements

The quadrotor should be able to track it's current position and travel to a specified waypoint. The quadrotor should consider itself at the waypoint once it has reached a specified distance error radius from the waypoints. The waypoint specification must allow for both a two dimensional specification and a full three dimensional specification. In the case of two dimensions, the quadrotor should maintain it's current position in the unspecified dimension.

4.9 Object Tracking

4.9.1 Description and Priority

The quadrotor should be able to locate and identify obstacles in it's immediate environment. This is useful for both avoiding obstacles and tracking the movement of a target. This is especially important when flying in a restricted area such as a forest or urban setting. Should the quadrotor fail to accurately track and avoid objects it will likely hit something, leading to a crash. While this is a serious issue, it is reasonable to expect most flights to take place in an open area, in which case most users would not need object avoidance.

Priority: 5

Business Requirement: The system must be able to safely fly in a complex environment.

4.9.2 Stimulus/Response Sequences

Stimulus	Response
1. The user wants to enable the object avoidance software	(a) The user locates the object avoidance switch on the control board (b) The user toggles the switch into the ON position

4.9.3 Functional Requirements

The object tracking feature must be able to create at least a two dimensional map of the surrounding area, out to within three seconds of travel at the current speed. This map must be updated at least two times per second.

4.10 Intuitive User Control

The quadrotor must have a manual flight capability. Ideally, this would be done through an intuitive user interface that does not require much training or experience to operate. Practically this is a low priority requirement since the quadrotor must be operable manually, but this control need not be *intuitive* for early adopters.

Priority: 6

Business Requirement: Gives the user the ability to fly the quadrotor with no prior flight experience

4.10.1 Description and Priority

4.10.2 Stimulus/Response Sequences

Stimulus	Response
1. The user wants to control the quadrotor manually, without programming flight data	(a) The user locates the manual flight switch on the control board (b) The user toggles the switch into the manual position (c) The remote flight control board is then able to control the orientation and altitude of the quadrotor

4.10.3 Functional Requirements

The user must be able to have complete control of the orientation of the quadrotor body, and the z axis thrust produced by the quadrotor. The user

control must be wireless and not require any connection to external electronics. The user controller should allow for rapid rotation of the quadrotor body.

4.11 Open Community Support

A good open source product must have a central location to store user generated technical support. Forums are one of the best methods available, and reduce the support burden of Anzhelka developers. However, forum based support is only useful once a community of users has been established.

Priority: 7

Business Requirement: An open community means open and free support

4.11.1 Description and Priority

4.11.2 Stimulus/Response Sequences

Stimulus	Response
1. The user has a problem with troubleshooting a problem with the quadrotor.	(a) The user browses to our forums website at forums.anzhelka.com (b) The user then selects the appropriate forum, based on the problem (c) The user then log in or create a new account (d) Click on the "Create a New Thread" button (e) In title box the user creates a descriptive title for the problem (f) In the body text box the user enters what the problem is and any troubleshooting that has already been done. (g) The user then posts the thread (h) The user waits for a helpful reply

4.11.3 Functional Requirements

The forums must be publicly viewable without an account, since most user questions are likely to repeat. The forums must be hosted so that they are accessible from anywhere in the world. In addition, the forums must be moderated, either by users or staff, to ensure a friendly environment free of profanity and politics.

5 Other Nonfunctional Requirements

5.1 Performance Requirements

N/A

5.2 Safety Requirements

N/A

5.3 Security Requirements

N/A

5.4 Software or Hardware Quality Attributes

N/A

5.5 Business Rules

N/A

6 Other Requirements

A Glossary

N/A

B Analysis Models

N/A

C To Be Determined List

N/A