Software Design Descriptions

An Application of Motion Planning on a Quadrotor

1 December 2013
## Change History

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dec 1, 2013</td>
<td>1.0</td>
<td>Created</td>
</tr>
</tbody>
</table>
### Signature

<table>
<thead>
<tr>
<th>Member</th>
<th>Student ID</th>
<th>Signature</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abdullah Ömer Yamaç</td>
<td>1885458</td>
<td></td>
<td>01.12.2013</td>
</tr>
<tr>
<td>Özgür URAL</td>
<td>1819622</td>
<td></td>
<td>01.12.2013</td>
</tr>
<tr>
<td>Jonard DOÇİ</td>
<td>1786862</td>
<td></td>
<td>01.12.2013</td>
</tr>
<tr>
<td>Ömer Faruk ÖZARSLAN</td>
<td>1751429</td>
<td></td>
<td>01.12.2013</td>
</tr>
</tbody>
</table>
Preface

This document contains the software design descriptions for “An Application of Motion Planning on a Quadrotor” project. This document is prepared according to the “IEEE Standard for Information Technology – Systems Design – Software Design Descriptions – IEEE STD 1016 – 2009”.¹

This Software Design Documentation provides a complete description of all the system design and views of the “An Application of Motion Planning on a Quadrotor” project.

The first section of this document includes an overview. The second section of this document includes definitions. The third section of this document includes conceptual model for software design descriptions. The fourth section of this document includes Design verification and design role in validation. Lastly, the fifth section includes the design viewpoints of the system.
# Table of Contents

Change History ............................................................................................................. 2

Signature ....................................................................................................................... 3

Preface ......................................................................................................................... 4

1 Overview.................................................................................................................. 8
  1.1 Scope ................................................................................................................. 8
  1.2 Purpose .............................................................................................................. 8
  1.3 Intended audience .............................................................................................. 8
  1.4 References ......................................................................................................... 9
  1.5 Definitions ......................................................................................................... 9

2 Conceptual model for software design descriptions .............................................. 10
  2.1 Software design in context ............................................................................... 10
  2.2 Software design descriptions within the life cycle ......................................... 11
    2.2.1 Influences on SDD preparation .................................................................. 11
    2.2.2 Influences on software life cycle products ............................................... 11
    2.2.3 Design verification and design role in validation .................................. 11

3 Design description information content ................................................................. 11
  3.1 Introduction ....................................................................................................... 11
  3.2 SDD identification ............................................................................................ 12
  3.3 Design stakeholders and their concerns ......................................................... 12
  3.4 Design views .................................................................................................... 12
  3.5 Design viewpoints ............................................................................................ 13
  3.6 Design rationale ............................................................................................... 13
  3.7 Design languages ............................................................................................. 14

4 Design viewpoints .................................................................................................. 15
  4.1 Introduction ....................................................................................................... 15
  4.2 Context Viewpoint ........................................................................................... 15
  4.3 Patterns Use Viewpoint ................................................................................... 16
4.4 Dependency Viewpoint

4.5 Interfaces Viewpoint
  4.5.1 Quadrotor-PC interface
  4.5.2 PC-Joystick Interface

4.6 State dynamics viewpoint

4.7 Logical Viewpoint
Figure 1 Axes of a flying vehicle

Figure 2 Use case diagram of the simulation menu

Figure 3 MVC Pattern

Figure 4 A GUI prototype for controlling the quadrotor written in MATLAB

Figure 5 Dependency Diagram

Figure 6 Zigbee Network Topology

Figure 7 Quadrotor-PC interface

Figure 8 Statechart diagram of the quadrotor functionalities

Figure 9 Statechart diagram of the simulation functionalities

Figure 10 UML class diagram of the simulator
1 Overview

This document is a Software Design Document to our graduation project which involves applying motion planning algorithms on a quadrotor. It includes the details for how the intended software should be built. The details are represented by using graphical notations such as context diagrams, use case models, sequence diagrams, class diagrams, and other supporting requirement information.

1.1 Scope

This Software Design Document will serve as a reference for the system designers and other stakeholders while developing and reviewing the final tangible product resulting from this project. In this document we will reveal the structure of the software we will develop, all the details about its member subsystems, the organization of the data flow, all designed in such a manner that the initial software requirements are fulfilled. The information contained within this document shall be enough for the designers to fully develop the software.

Our final software product is a system simulation running in Unity and an embedded software written in Arduino, coupled with a Matlab application. The purpose of our application is to employ maneuvers on a quadrotor and apply motion planning algorithms on it. To achieve our objective, a multitude of systems will be designed to cooperate with each other.

1.2 Purpose

The purpose of this document is to aid the designers in the building of the software. For this sole reason, the SDD is of the highest importance among the documents. The main purpose is to explain the features and functionality of the software. Aside from the design details, this document shall also be a guide to the implementers of the system. This document will cover the in-Depth design and development details by providing with a set of diagrams that will help its readers visualize the functionality of the software.

1.3 Intended audience

This document is aimed at the current developers of this project, developers that may join the team in the future, developers that may extend or modify any part of this software. Its audience is also consultants and future investors or stakeholders. Considering this document has an eclectic audience, it shall be structured in such a way that its content will be intelligible to every party included in the audience.
1.4 References


1.5 Definitions

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESC</td>
<td>Electronic Speed Controller</td>
</tr>
<tr>
<td>PWM</td>
<td>Pulse width modulation</td>
</tr>
<tr>
<td>EEPROM</td>
<td>Electronically Erasable Programmable Read-Only Memory</td>
</tr>
<tr>
<td>SRAM</td>
<td>Static random-access memory</td>
</tr>
<tr>
<td>Arduino</td>
<td>Microprocessor</td>
</tr>
<tr>
<td>XBee</td>
<td>Wireless Serial Communication Hardware</td>
</tr>
<tr>
<td>Arduino-XBee Shield</td>
<td>It is a component which provides plug in XBee on to Arduino</td>
</tr>
<tr>
<td>CW</td>
<td>Clockwise</td>
</tr>
<tr>
<td>CCW</td>
<td>Counter Clockwise</td>
</tr>
<tr>
<td>X-CTU</td>
<td>It is a software which enables to configure Xbees</td>
</tr>
<tr>
<td>RTS</td>
<td>Request to Send</td>
</tr>
<tr>
<td>CTS</td>
<td>Clear to Send</td>
</tr>
<tr>
<td>Pitch</td>
<td>Rotation about y-axis, as seen in Figure 1</td>
</tr>
<tr>
<td>Roll</td>
<td>Rotation about x-axis, as seen in Figure 1</td>
</tr>
<tr>
<td>Yaw</td>
<td>Rotation about z-axis, as seen in Figure 1</td>
</tr>
<tr>
<td>PID</td>
<td>Proportional-Integral-Derivative</td>
</tr>
<tr>
<td>IMU</td>
<td>Inertial Measurement Unit</td>
</tr>
</tbody>
</table>

Table 1
2 Conceptual model for software design descriptions

In our project mainly focuses on academic and research purposes; but application of this project contains several areas. We are trying to apply motion planning algorithm on a quadrotor. This algorithm may be changed to get better results by us.

Let me define how our product looks like. There is a quadrotor which may have camera. We will set the initial position then we will give next position. Quadrotor will move towards to destination automatically. Although this movement done by auto. We have a mode to fly quadrotor manually. When the quadrotor arrived the destination camera may took a photo. This was the one possible application of our project. Another possible application is preventing the collision with quadrotor. Firstly, we must detect the object goes through the quadrotor and then we can apply motion planning algorithm to escape from incoming object.

2.1 Software design in context

Our project contains two parts. One of them is quadrotor and the other one is simulation.
In the quadrotor part we won’t design as object-oriented. The reason of this decision is mostly efficiency. Each function must run as fast as possible. Therefore we will design specific functions for each specific problem. Speed is very important for us; because this project may be used for military purposes. Any latency may cause a possible danger. Communication is also important. It means that stability is important. Any communication error may cause a collision. Quadrotor must response same output for each command which comes from PC. As a result we will design this part based on efficiency.

2.2 Software design descriptions within the life cycle

2.2.1 Influences on SDD preparation

The basic influences are SRS report, project proposal and also other similar projects which we researched. Until the SDD report, we started the project and experiments. These experiments also give us idea what should we do or shouldn’t. Although we have an academic project, some people which live in our habitat shares their ideas about our project. These ideas also shape SDD report.

2.2.2 Influences on software life cycle products

In this project, we decided to design simulator and quadrotor simultaneously. First thing which we will plan to do is calculating physical properties such as gravitational center, lift power of quadrotor. These calculations will shape both simulation and quadrotor. After these steps quadrotor and simulation is splitting.

In quadrotor part, we will control quadrotor manually, then we apply motion planning algorithm which is automatically. This method eases to implement second part.

2.2.3 Design verification and design role in validation

Validation and Verification will be tested in some periods which will depend on the process. These processes can be found in weekly progress such as when wireless communication is done, it must be tested. After the test is successful, then we will go on next step.

3 Design description information content

3.1 Introduction

In this section of the SDD, we will describe the structure of this SDD, as well as give information about the stakeholders of the design process, the design views that will be used to shape our final product, the inherent design viewpoints part of the design process, the different design
elements, as well as design constraints, design overlays, the design rationale and last but not least the design language used in the composing of this document.

3.2 SDD identification

This document concerning the design of our project for the CENG490 senior course shall be delivered to the stakeholders on the 1st of November, 2013. The authors are the members of our team “Clover”:

- Abdullah Omer Yamac
- Jonard Doci
- Ozgur Ural
- Omer Faruk Ozarslan

A reviewed version of this document will be issued after consulting from the stakeholders and including additional design features.

This document shall contain references to any used material outside the scope of the writers’ ownership.

This document shall be written in context with the previously written SRS document which specified the requirements for this project.

This document shall clearly and unambiguously indicate the details of the design of this project with a comprehensible language that shall not fall subject to misinterpretation.

3.3 Design stakeholders and their concerns

The stakeholders of this project include but are not limited to, our development team, the consultants/mentors of our team, the sponsors of our project and also the parties that may purchase our product in the future, or even academic reviewers that may want to reference our work in their own papers.

The stakeholders concerned with the design of our project are the development team itself and our consultants, but the other parties are encouraged to give feedback and interfere at a small scale with the design process.

3.4 Design views

The design views of the SDD are to be selected in a symbiotic harmony between the mindset of the developers, the requests of the stakeholders, the initial requirements specifications. This SDD shall not provide conflicting design views and viewpoints as they are selected to be well-matched. Our project shall be developed in a modular fashion following the ‘open for extension, closed for
modification’ approach of software development. Each subsystem will be developed separately preserving encapsulation features, and each subsystem will be unaware of the inner details of other subsystems, it only need to know the interface of that subsystem in order to communicate, give input to that subsystem in order to retrieve the required output. After developing each subsystem, they shall be merged together in a single unified system. This merging shall be observed carefully in order to examine the emergent properties that may arise from the interaction of the smaller subsystems. The design views are built while having in mind the various system constraints, costs and limitations. In addition, team coordination is essential when building design views, each member shall be assigned different tasks depending on that member’s capabilities in order to increase the efficiency and to create proper scheduling. The context views show the interaction between the system, the user, and its surroundings. The dependency views show the components and the subsystems, thus revealing the interdependencies between them. The interface views express the interaction between the subsystems. The state dynamic views display the dynamic behavior of the application and the logical views show the abstraction of the systems by displaying the designed classes and interfaces.

3.5 Design viewpoints

In our project we have selected several design views which are inherently compatible with its nature, an embedded systems project bundled together with a simulation of an unmanned flying vehicle, a quadrotor. For such a project selecting proper design viewpoints may be rather intricate and challenging compared to straightforward software projects, but still we managed to find viewpoints germane to our project.

We have a context viewpoint, patterns viewpoint, dependency viewpoint, interfaces viewpoint, state dynamics viewpoint and logical viewpoint in our project. Each of these viewpoints is explained in details in the fifth section of this report. Evaluation and analysis techniques will be applied to the views

3.6 Design rationale

While designing our project, the decisions that are made regarding the design choices are based especially on the software requirements, since conforming to the requirements stipulated by the user is one of the crucial aspects of software developing. In addition to adhering to the specified requirements, other important aspects that give shape to the design conditions that we make are, developing a system that is reliable, thus perform the required functionalities by the user at any time, usable, it will have an proper user interface, extensible, it will provide the capability of adding new features to the product easily, performance, since some of our software will operate together
with an embedded device, it should run at a speed above a certain threshold. So before making any
design decision, we deeply analyze whether that decision will result in a system that has the
previously stated system properties.

3.7 Design languages

In the content of this document standardized UML will be used to illustrate the design
viewpoints. UML\textsuperscript{[2]} is chosen because of its widespread presence in the software engineering world
and because it serves as a bridge of communication between the developers and the stakeholders.
4 Design viewpoints

4.1 Introduction

The standard for SDD document is organized into viewpoints. Viewpoints specify a design concern that is addressed by a design view. And thus for each design view, there is exactly one design viewpoint governing it. Design views correspond to one single or multiple related UML diagrams.

4.2 Context Viewpoint

The following use case diagram is presented to show an overview of the relationships and interactions between actors and the system.

![Use case diagram of the simulation menu](Figure 2 Use case diagram of the simulation menu)
4.3 Patterns Use Viewpoint

This viewpoint addresses design ideas as collaboration patterns involving abstracted roles and connectors.

Motion planning on a quadrotor’s Simulation part will be implemented by using Model View Controller (MVC) pattern\(^3\). In this pattern, user will interact with the system by a user interface. When user makes an action such as mouse click, this input will be handled by Simulator. The simulator notifies the model of the user action. Then simulator updates the display.

![MVC Pattern](image)

Figure 3 MVC Pattern

4.4 Dependency Viewpoint

We have two main subsystems in our main system. The first one is the subsystem running on the PC, which will contain the controller of the quadrotor. This controller is implemented in Matlab and is made possible through a simple GUI illustrated in Figure 3 and through a simple joystick. The controller sends serial data to the arduino microcontroller through a serial port which is open on the pc. The data is send through the Xbee wireless transmitter, which will be connected to the PC and the arduino.
The second subsystem is the most important one. It contains the Arduino microcontroller which serves as the brain of the subsystem. This subsystem also contains the two sensors: a Gyroscope and an Accelerometer. It also contains 4 ESCs which are used to control the 4 rotors. An Xbee shield is mounted on top of the Arduino to facilitate wireless communication. The microcontroller collects the data provided by the sensors and uses them to calculate the optimal rotation speeds of the rotors to achieve its stabilization. After doing so it sends the proper values to the ESCs to actuate the rotation speed of the motors.

The two systems communicate together through serial communication through the Xbee communication protocol. We will use the analog inputs of a gamepad joystick to make the controlling of the quadrotor more usable and to make the interaction between the user and the quadrotor more enjoyable. After getting the input from the gamepad analog sticks, the MATLAB application sends serial data to the arduino that controls the motion of the quadrotor. Each command contains a value between 0 and 255, 0 being no rotation at all and 255 maximum rotation capacity of the motor will be utilized. Thus each command for a single motor will take up only 8 bits of data, thus speeding up the communication process. The communication between the two subsystems is particularly important because a failure or any bug may greatly compromise the success of the project and may even lead to damage of the quadrotor.

In Figure 4, you can find the illustration of the two subsystems and their respective components.
4.5 Interfaces Viewpoint

In this part of document, interfaces between the subsystems will be defined.

There must be at least 2 interfaces between sub-systems. Quadrotor –Matlab(PC) interface and Matlab(PC)-Game Pad (Joystick) interface.

Communications between computer and quadrotor will be established via Xbee (Zigbee module). We will create a network through Xbees. One of Xbee is coordinator, is connected to PC, the other Xbees, it may connect more than one Quadrotor, are rooter or end-device. At below you may see the simple network which is similar to our network.
We will use usb-explorer for Xbee to connect to PC. Xbees can communicate through RS232 Serial Port; but we want to use with usb port. For this problem a solution was created. This was the virtual com port. We will create VCP (virtual com port).\(^6\)

On the other hand we will use Arduino Xbee shield to plug-in Xbee on to Arduino.

Also communication between PC and Game Pad is through usb port.

### 4.5.1 Quadrotor-PC interface

This interface will be used to connect the quadrotor and PC.

**Diagram**

![Quadrotor-PC interface diagram](image)

**Figure 7 Quadrotor-PC interface**

**Functions**
### Function Name

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Caller</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>changeMotorRPM()</td>
<td>PC/Matlab</td>
<td>This function will be used to change produced PWM signal by Arduino (Microprocessor of quadrotor). This PWM signals changes the rpm (motor rotation per minute) value through ESCs.</td>
</tr>
<tr>
<td>getSensorData()</td>
<td>Quadrotor</td>
<td>This function will be used to get data of sensors which are located on quadrotor. These sensors are gyroscope, accelerometer and sonar. Information is collected by Arduino then a meaningful data send through PC. Some of data will be used by quadrotor.</td>
</tr>
</tbody>
</table>

### Messages

<table>
<thead>
<tr>
<th>Message Name</th>
<th>Sender</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>newValue</td>
<td>PC/Matlab</td>
<td>This message will have at most 4 characters. First character will decide which motor’s or motors’ rpm value will change. The other 3 character will change the rpm value to speed up or down.</td>
</tr>
<tr>
<td>newValue</td>
<td>Quadrotor</td>
<td>This message will have at most 4 characters. First character will decide which axes and the other 3 character will state the magnitude.</td>
</tr>
</tbody>
</table>

### 4.5.2 PC-Joystick Interface

This interface will be used to connect the PC and joystick when the quadrotor in manual mode. Also this interface was defined before in matlab. Therefore we don’t need to recreate. Description of the interface is at below.\[5\]

### Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>axis</td>
<td>( a = \text{axis}(\text{joy}, \ n) ) reads the status of joystick with axis number ( n ). Axis status is returned in the range of -1 to 1. The ( n ) parameter may be a vector to return multiple buttons.</td>
</tr>
<tr>
<td>button</td>
<td>( b = \text{button}(\text{joy}, \ n) ) reads the status of joystick button number ( n ). Button status is returned as logical 0 if not pressed and logical 1 if pressed. The ( n ) parameter may be a vector to return multiple buttons.</td>
</tr>
<tr>
<td>caps</td>
<td>( c = \text{caps}(\text{joy}) ) returns joystick capabilities, such as the number of axes, buttons, POVs, and force-feedback axes. The return value is a structure with fields named \text{Axes}, \text{Buttons}, \text{POVs}, \text{and Forces}.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>close</td>
<td>close(joy) closes and invalidates the joystick object. The object cannot be used once it is closed.</td>
</tr>
<tr>
<td>force</td>
<td>force(joy, n, f) applies force feedback to joystick axis n. The n parameter can be a vector to affect multiple axes. f values should be in range of -1 to 1, and the number of elements in f should either match the number of elements of n, or f can be a scalar to be applied to all the axes specified by n.</td>
</tr>
<tr>
<td>pov</td>
<td>p = pov(joy, n) reads the status of joystick POV (point of view) of control number n. pov is usually returned in degrees, with -1 meaning &quot;not selected.&quot; n can be a vector to return multiple POVs.</td>
</tr>
<tr>
<td>read</td>
<td>[axes, buttons, povs] = read(joy) reads the status of axes, buttons, and POVs of the specified joystick. [axes, buttons, povs] = read(joy, forces) applies feedback forces, in addition, to a force-feedback joystick.</td>
</tr>
</tbody>
</table>

### 4.6 State dynamics viewpoint

In this viewpoint we give a layout of the states and state transitions of our project. This viewpoint helps in visualizing the overall functionality and the dynamic nature of applications.

In Figure 8 Statechart diagram of the quadrotor functionalities you can see the main state transition diagram of the quadrotor.
The above Statechart diagram describes the behavior of the quadrotor. Initially the quadrotor is in OFF state, after turning it on, it is in a stationary state and waits for serial input from the joysticks connected to the Matlab application. Immediately after receiving any input from the joysticks, the quadrotor’s rotors will start rotating, thus the quadrotor will move in the flight state. While in this state it may continue to receive additional motion inputs from the joystick, it may receive the LAND command, or it may receive the Emergency POWER OFF command. Each of these commands will move the quadrotor in the flight state, stationary state, and final state respectively.

In Figure 9 Statechart diagram of the simulation functionalities you can see the main state transition diagram of the simulation part of the project.

4.7 Logical Viewpoint

The purpose of this viewpoint is to elaborate existing and designed types and their implementations as classes and interfaces with their structural static relationships.

The project heavily depends on Gazebo framework; therefore Figure 10 is the diagram which doesn’t depend on Gazebo framework⁶. Further details (which should include some internals about Gazebo) will be placed and the diagram will be updated accordingly in next versions of this document.
Figure 10 UML class diagram of the simulator