**CSCI 5551 Fall 2015 Term Project**

**UAV Applications – Image Based Tracking of Mobile Object**

John Erickson

[eric0870@umn.edu](mailto:eric0870@umn.edu)

Computer Science and Engineering Department

The University of Minnesota at Twin Cities



# Abstract

The intent of this project is to develop an image based tracking application for a quadrotor UAV. Much work has been done in industry and academia in the field of hobbyist level quadcopter development, resulting in open source software and hardware that are readily available and affordable. This project will leverage COTS hardware and open source software libraries in order to develop a client based control application that enables a drone to track a mobile object. The drone will autonomously maintain a desired distance and orientation from the object as it moves through space. This is not a novel application, but rather a milestone on the way to building up an autonomous flying system with marketable capabilities. This project will address a small number of trade studies, including the development platform, software development environment and software library support. The objective of these trades is early validation of the project concept. The majority of the work performed in this project will be in the domain of software development, specifically focusing on computer visualization and wireless communication. The outcome of this project will be twofold: an application potentially reusable by the development community, and a start on the development of a future robotic system.

Contents

[Abstract 1](#_Toc437613367)

[1. Introduction 3](#_Toc437613368)

[1.1. Problem Description 3](#_Toc437613369)

[1.2. Motivation 3](#_Toc437613370)

[1.3. Related Work 3](#_Toc437613371)

[1.4. Work Plan 3](#_Toc437613372)

[2. Background 4](#_Toc437613373)

[2.1. Requirements 4](#_Toc437613374)

[2.2. Trade Studies 5](#_Toc437613375)

[2.2.1. Development Platform Trade 5](#_Toc437613376)

[2.2.2. Development Tools Trade 5](#_Toc437613377)

[2.3. Development Tools 6](#_Toc437613378)

[2.3.1. AR.Drone 2.0 6](#_Toc437613379)

[2.3.2. AR.Drone Software Development Kit (SDK) 2.0 7](#_Toc437613380)

[2.3.3. Robot Operating System (ROS) 8](#_Toc437613381)

[2.3.4. AR.Drone Autonomy 8](#_Toc437613382)

[2.3.5. Open Computer Vision (Open CV) 9](#_Toc437613383)

[3. Work 9](#_Toc437613384)

[3.1. Proof of Concept 9](#_Toc437613385)

[3.2. Follow Me Application 9](#_Toc437613386)

[4. Results and Conclusions 10](#_Toc437613387)

[4.1. Overview 10](#_Toc437613388)

[4.2. Future Work 10](#_Toc437613389)

[4.3. Acknowledgements 10](#_Toc437613390)

[5. Reference 11](#_Toc437613391)

[5.1. Citations 11](#_Toc437613392)

[5.2. Resources 12](#_Toc437613393)

[5.3. Images 12](#_Toc437613394)

[5.4. Terminology 13](#_Toc437613395)

# Introduction

## Problem Description

The intent of this project is to develop an image based tracking application for a quadrotor UAV, more specifically, an application capable of allowing an autonomous flying system to track a mobile object using image recognition algorithms. The application must allow the user to command the drone to begin and end tracking, as well as indicate the desired distance and orientation the drone should maintain from the object. Initially a detector tag may be used for tracking; however, it is highly desirable to eventually migrate to tracking a user provided digital image.

In order to accomplish this goal, a system capable of autonomous flight control and real time feedback will be needed. At a minimum this system must possess the following: inertial measurement unit capable of reporting acceleration and attitude, altitude sensor, front facing camera, and wireless communication with client device. The following support is also highly desired: side and down facing cameras, magnetometer, user application processor resident on device, flight data and video stream recording capability. The tracking and controlling application may be hosted on either the drone itself or a client device (PC or mobile system).

This paper provides an Introduction to the project, including the problem description, motivation, related work and the work plan. A Background section is included to provide the reader with an understanding of project requirements, trade studies performed and development tools used. The development activities performed in this project are provided in the Work section, including the proof of concept and Follow Me application implementation. Future work is also included in this section. Finally, the project Results and Conclusions are presented, followed by a Reference section. A Terminology section is included at the end to allow the paper to flow without inline definitions for acronyms and industry standard terms.

## Motivation

The motivation behind this project stems from a number of factors. One, to broaden experience working with autonomous flying vehicles from a control and communication perspective. Two, to develop an application potentially reusable by the community, academic or commercial. And three, to develop an application that may eventually be incorporated into a marketable product.

## Related Work

Much work has been done in industry and academia in the field of hobbyist level quadcopter development, resulting in open source software and hardware that are readily available and affordable. From universities to start-ups to large corporations, time and money is being spent furthering the technology in the interest of utility, convenience, and profits. The concepts developed in this project are by no means novel; rather they are readily available in forms ranging from proprietary to open source. Companies like 3D Robotics, Parrot and EZ-Robot, among numerous others, have invested in software and hardware development, resulting in modular plug-and-play systems readily available to anyone with an interest in robotics.

**< Add paragraph(s) on specific projects from References:**

* **Boyers thesis,**
* **CMU project**
* **Object Tracking Using Camshift Alg**
* **Vision-Based Guidance Control**
* **Visual Navigation of a Quadrotor**
* **Navigation and Control Technology inside the AR.Drone**

## Work Plan

The first phase of the project will be a series of trade studies to identify the development platform, software development environment and software library support. A number of development platforms will be traded, ranging from DIY to ready-to-fly systems. The selected development platform must be able to meet all of the system requirements of the project. The development environment and software library trades will assess the capabilities of available software development kits and community libraries supporting the selected development platform.

After selection of the development platform has been completed, a proof-of-concept task will be conducted. The primary goal of this task is to quickly prove out the ability to accomplish the overall project goal of mobile image tracking on the development platform. A community developed and/or commercially available application will be used for this task.

The project specific application development will follow the proof-of-concept phase. The software development will be conducted in the selected development environment, utilizing selected SDK’s and software libraries. The output of this project phase will be a custom image based “Follow Me” application capable of tracking a mobile user. The validation of the final product will also be included in this phase.

Upon completion of the project, future work will be addressed and presented in the final report.

Configuration management of artifacts developed for the project will be conducted using Git [E]. All artifacts are available on GitHub under the following projects:

* Design: [https://github.com/eric0870/CSCI\_5551\_Proj](https://github.com/Eric0870/CSCI_5551_Proj)
* Code: <https://github.com/eric0870/ardrone_followme>

# Background

## Requirements

The system requirements for the project are provided in Table 1 below. These requirements were derived in the interest of completing the project within a reasonable budget and an aggressive schedule. They were used to govern the procurement of development hardware and tools, as well as drive the system and software development effort of the project.

Table 1: Project Requirements

| **ID** | **Requirement** | **Threshold** | **Objective** |
| --- | --- | --- | --- |
| 1 | Cost of development Hardware | < $500 | < $300 |
| 2 | Cost of development Software | < $100 | Free |
| 3 | Availability of development Hardware/Software | Now |  |
| 4 | Project development schedule | 2 months |  |
| 5 | Development platform form factor | Quadcopter UAV |  |
| 6 | Development toolchain support | Mature community/ commercial support |  |
| 7 | Development platform sensor suite   1. Inertial Measurement Unit 2. Altitude sensor | 1. 6 DOF (accel, gyro) 2. Ultrasound telemeter | 1. 9 DOF (accel, gyro, mag) 2. Pressure sensor |
| 8 | Development platform video   1. Camera(s) 2. Video stream frame rate | 1. 1 (front facing) 2. 15 FPS | 1. 2 (front, down facing) 2. 30 FPS |
| 9 | Development platform support for user application software | Onboard | Remote |
| 10 | Development platform real-time communication with user application   1. Communication type 2. Data: user app to drone 3. Data: drone to user app 4. Rate | 1. Wired or WIFI 2. Command / control 3. Nav data, video 4. 1 Hz updates | 1. -- 2. -- 3. -- 4. 100 Hz updates |
| 11 | The system shall provide an application capable of image based tracking of a mobile object  *Goal: the application should be agnostic to both the development platform and host device* | Tracking based on detector tag image supplied to application | Tracking based on generic digital image supplied to application |
| 12 | The system shall provide a driver capable of routing images to the application, and control commands to the drone |  |  |
| 13 | The image tracking software program shall be hosted on a user computer capable of wireless internet | PC running Linux based OS | Mobile phone running Android OS |

## Trade Studies

## Development Platform Trade

A number of development platforms were traded, ranging from DIY to ready-to-fly systems. The outcome of the study was the selection of the AR.Drone 2.0 by Parrot. The AR.Drone is equipped with an onboard sensor suite and proprietary set of control algorithms that provide assistance to the user for all supported maneuvers [22]. Ergo, less time will be spent fine tuning vehicle stability, leaving more time to accomplish project specific goals. The AR.Drone is also widely used by the community, resulting in numerous user applications available for flight demos, proof-of-concept and open source examples. Table 2 provides a summary of each of the development platforms included in the trade study.

Table 2: Development Platforms

|  |  |  |  |
| --- | --- | --- | --- |
| **Platform** | **Cost** | **Pros** | **Cons** |
| Ardupilot, by 3D Robotics [1][3] | $500 | Open SW/HW | Does not meet project development schedule requirement |
| Phantom 3, by DJI [7] | $700 | High quality camera and sensors | Does not meet project cost requirement |
| AR.Drone 2, by Parrot [19] | $300 | Meets project requirements,  Widely used by community | Closed onboard processor (but has well defined client interface) |
| Bebop, by Parrot [20] | $500 | High quality camera and sensors,  Open onboard processor | Does not meet project development toolchain requirement |
| Spiri, by Pleiades [21] | $1200 | High quality camera and sensors,  Open onboard processor | Does not meet project cost and availability requirements |
| Hummingbird, by AscTec [2] | $5000 | High quality camera and sensors,  Open onboard processor | Does not meet project cost requirement |

## Development Tools Trade

A number of free tools are available to support development activities related to robotics [14]. They range from the development environment to libraries associated with the development platform and project application to examples provided by the community. The tools targeted for this project include ROS (primary) and OpenCV (secondary). The development environment and software library trades were combined to ensure compatibility. The primary objective of this study was to select a development environment supporting the integration of the tools noted above, along with the AR.Drone SDK 2.0. Secondary objectives included community support, quick start and ease of use. Table 3 and Table 4 below provide a summary of the development tools traded. The outcome of the study was the selection of the Eclipse development environment. Due to lack of ROS support in Windows, the software development for the project will be done in Linux. A number of software libraries and development kits were included in the trade, the majority of these will be leveraged during the development of the Follow Me application.

Table 3: Development Environments

|  |  |  |  |
| --- | --- | --- | --- |
| **IDE [9]** | **OS** | **Pros** | **Cons** |
| Eclipse [K] | Linux, Windows | Cross platform support,  Multi-language support,  Well supported, mature toolchain |  |
| EZ-Builder [A] | Windows | High level IDE for quick start,  Well supported in community | No Linux support |
| Qt Creator [L] | Linux, Windows | Cross platform support,  Multi-language support,  Native support for CMake projects |  |
| Visual Studio [F] | Windows | Well supported, mature toolchain | No Linux support,  Limited capability in free version |

Table 4: Development Libraries

|  |  |  |  |
| --- | --- | --- | --- |
| **Library / SDK** | **OS** | **Pros** | **Cons** |
| AR.Drone Autonomy Lab [J] | Linux | Integrated with AR.Drone SDK,  AR.Drone Examples |  |
| AR.Drone SDK 2.0 [H] | Linux, Windows | High and low level support of AR.Drone specific features |  |
| EZ-SDK [B] | Windows | High level IDE for quick start,  Well supported in community | No Linux support |
| EZ-SDK Mono [C] | Linux | High level support for quick start |  |
| OpenCV [D] | Linux, Windows | Rich set of CV libraries,  Multi-language support,  Well supported in community |  |
| ROS [G] | Linux | Rich set of robotics libraries,  Well supported in community |  |

## Development Tools

## AR.Drone 2.0

The AR.Drone 2.0, hereafter referred to as the AR.Drone, is a second generation quadrotor UAV developed by Parrot [19]. Parrot’s intention for the drone is to provide a platform for academics and hobbyists that allows the developer to focus their effort on software application development, leveraging COTS OEM hardware and a small set of aftermarket accessories. The progression from first to second generation included sensor and camera upgrades, the addition of a pressure sensor, magnetometer, and USB port as well as software/firmware updates. A brief discussion of the AR.Drone hardware and software is included here to familiarize the reader with the drones components and capabilities.

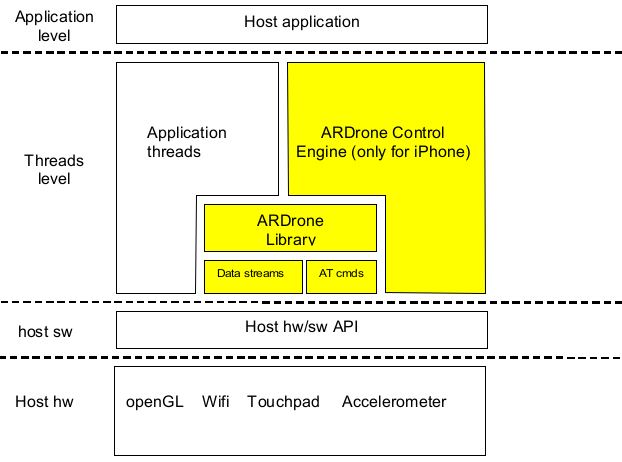
Figure 1: Parrot AR.Drone 2.0

The AR.Drone is an indoor/outdoor quadrotor powered with a rechargeable 12V 3-cell LiPo battery pack. The drone’s movement is controlled by four 3-phase DC brushless motors, each connected to a propeller. The drone’s sensor suite includes a 9DOF MEMS IMU (accelerometer, gyroscope, and magnetometer), an ultrasound telemeter, and pressure sensor. Data provided by this sensor suite allows the drone’s onboard mission processor to resolve heading, pitch, roll, tilt and altitude. The drone also has two onboard cameras, one front facing, and the other downward facing. Both cameras are capable of up to 720p image resolution and 30 FPS. In addition to capturing photos and video, the downward facing camera is also used resolve ground speed measurements. Communication between the drone and a client device is facilitated by an onboard Wi-Fi controller. The drone generates its own wireless network, allowing a client to connect and interact with the drone.

The mission processor software is hosted on a TI OMAP 3 series processor containing an ARM Cortex A8 core running a Linux based real-time operating system. The application is responsible for managing multiple threads, including Wi-Fi communications, video data sampling, video compression (for wireless transmission), image processing, sensor acquisition, state estimation and closed-loop control [18].

The AR.Drone was developed with safety and convenience in mind. The drone ships from the factory with a detachable foam propeller guard for indoor flight. It has casualty modes capable of landing the UAV in cases where a propeller obstruction or low battery is detected. Parrot also built in a handful of high level commands for the most common maneuvers (take-off, trimming, hovering in place and landing), as well as configurable flight control limits on altitude, yaw speed, vertical speed and tilt angle.

## AR.Drone Software Development Kit (SDK) 2.0

The AR.Drone SDK 2.0, hereafter referred to as the AR.Drone SDK, is an open source software development kit written for AR.Drone application development [22]. The SDK is a collection of libraries and examples written in C and compatible with Windows, Linux, Android and iOS operating systems. The library collection ranges from a hardware level Application Programming Interface (API) to application level routines. The API abstracts hardware management providing functions for Wi-Fi, Bluetooth, etc. The application level provides routines for memory allocation, thread management, communication, video stream processing, etc. The examples included with the SDK range from bare minimum communication modules to full up mobile applications. Figure 2 illustrates the software architecture for a user application built on the AR.Drone SDK.

The AR.Drone Library (and file structure) is grouped into modules as indicated in Table 5 below. Within the “Soft” component is a set of AR.Drone specific code, including: header files defining the drone’s communication structures, an AR.Drone Tool kit, and a set of utilities. The AR.Drone Tool kit is a library providing functions for all Client / drone communication. A breakdown of the AR.Drone Tool is provided in Table 6. The communication services supported by the AR.Drone are grouped into four categories, each with its own communication port. A listing of these services is provided in Table 7. The AR.Drone Tool provides a means of assembling and parsing data over each of these communication ports, including codecs for video processing.

Figure 2: AR.Drone SDK 2.0 Layered Architecture

Table 5: AR.Drone SDK Components

|  |  |  |
| --- | --- | --- |
| **Component** | **Sub-Component(s)** | **Description** |
| Soft | Common | Header files describing the drones communication structures |
|  | Lib/ardrone\_tool | API to handle the client command and receive threads |
|  | Lib/utils | Utilities to support application development |
| VLIB | -- | Video processing library |
| VPSDK | VP Stages | API for video processing |
|  | VP OS | API for system level support (memory allocation, thread mgmt., etc) |
|  | VP COM | API for communication support (Wi-Fi, Bluetooth) |
|  | VP API | API for video pipeline |

Table 6: AR.Drone Tool Components

|  |  |
| --- | --- |
| **Tool** | **Description** |
| AT | Contains functions for configuring and commanding the drone |
| Com | Contains functions for communication configuration for serial and Wi-Fi |
| Control | Contains functions for critical drone control over the Control Port (5559) |
| Navdata | Contains functions for Navdata parsing |
| Video | Contains functions for video parsing and recording |

Table 7: AR.Drone Communication Ports

|  |  |
| --- | --- |
| **Port** | **Description** |
| 5554 | UDP protocol for sending Navdata from drone to Client |
| 5555 | TCP protocol for sending video stream from drone to Client |
| 5556 | UDP protocol for sending configuration/control (AT) commands from Client to drone |
| 5559 | TCP protocol for sending critical mission data from Client to drone |

**<Highlight AR.Drone SDK components used in project>**

## Robot Operating System (ROS)

Overview – note using Jade version

Framework used in project (Messages, Topics, Nodes, etc)

Launch scripts:

Provide general description from ROS wiki;

(how used in project addressed in implementation section)

Utilities used in project (rosbag, etc)

## AR.Drone Autonomy

The ardrone\_autonomy package is an ROS driver for the Parrot AR.Drone quadcopter. This package is based on the official AR.Drone SDK [3]. It was originally developed by the Autonomy Lab at Simon Frasier University [24], though it is now maintained as an open source project on Github.

Overview

Stuff used in project

Ardrone\_autonomy: launch scripts,

## Open Computer Vision (Open CV)

Overview – note version

Framework used in project

Utilities used in project

# Work

## Proof of Concept

The primary goal of this task was to quickly prove out the ability to accomplish the overall project goal of mobile image tracking on the development platform. The Windows based EZ-Builder [A] application was leveraged for this task due to its inherent support for the AR.Drone quadcopter. EZ-Builder is a high level development environment created by EZ-Robot [6]. The application is targeted to developers interested in programming and interacting with the EZ-Robot products, however it also contains libraries for third party robots and even supports low level libraries for DIY development. EZ-Robot has made a number of tutorials available to the developer who wishes to come up to speed quickly. One of such tutorials [5] discusses using the application to control the AR.Drone to track an object based on its color. To conduct the proof-of-concept, the suggestions provided in this tutorial were used.

An EZ-Builder project was created that included an AR.Drone Movement Panel (third party add on) and a Camera controller. The Camera controller was configured to command the drone to track objects with the color red. During the test, the EZ-Builder application communicated with the drone over the PCs WIFI network connection. The drone was commanded to Take-Off and then commanded to track the red object. At a high level, the tracking was controlled by the application as follows: The drone telemetered camera image and vehicle state information to the EZ-Builder application over its WIFI network. The application used the camera image as input into its control algorithms to generate acceleration commands that would allow the drone to maintain a desired orientation with respect to the red object. These acceleration commands were then passed back to the drone via the WIFI network.

The proof-of-concept was successfully accomplished in a relatively short timeframe. The AR.Drone quadcopter demonstrated the ability to track a mobile object based on image processing performed real time on a remote device. A video of this test is available at the Github URL provided in section 1.4 above.

## Follow Me Application

Paragraph(s) on Follow Me

Overview

Design (including state machine, etc)

Discuss why used python was preferred over C++:

Preferred ease of development inherent in high level language over performance (timing) of C/C++

Notes on implementation (open source reuse, custom code)

Launch script(s) - include description (configure drone and launch processes needed) and snippet in paper

ROS custom package, my Python source – include description and snippet in paper

Include pseudo code of all implementation in this section – prof specifically asked for this.

Prof mentioned that Python is very good to learn.. maybe reason enough to prefer over C++ for project..

# Results and Conclusions

## Overview

Discuss limitations of application as implemented, and what could be done as future work to address

Discuss any assumptions made to simplify implementation

Validation of application – how well did it work, where did it fail

The idea here is to critically analyze the results of your work. Show you understand what you did, and what the results were and why you guys the results you did. Eg, it didn’t work very well because of noise, uncontrolled variables, etc.

## Future Work

More work w OpenCV, did not get to this during project

Assess requirements that were left out or partially implemented (ie Android hosted client)

Take a look at Boyers paper, page 81

## Acknowledgements

A special thanks goes out to the groups that made open source robotics code available to the community, including Parrot, the Open Source Robotics Foundation, and Simon Frasier University. Gratitude is also extended to those who provided online tutorials for the AR.Drone including Mike Hammer and D.J. Sures.

# Reference

## Citations

1. 3D Robotics. “ArduPilot.” Internet: <http://ardupilot.com>, [10/10/2015].
2. Ascending Technologies. “AscTec Hummingbird.” Internet: <http://www.asctec.de/en/uav-uas-drones-rpas-roav/asctec-hummingbird>, [10/10/2015].
3. Autonomy Lab, “Documentation.” Internet: <http://ardrone-autonomy.readthedocs.org>. [11/1/2015].
4. C. Anderson, J. Munoz. “3D Robotics, Inc.” Internet: <https://store.3drobotics.com>, [10/10/2015].
5. D. Sures. “Color Following AR Parrot Drone.” Internet: <http://www.instructables.com/id/Color-Following-AR-Parrot-Drone/?ALLSTEPS>, [10/10/2015].
6. D. Sures. “EZ-Robot.” Internet: <http://www.ez-robot.com>, [10/10/2015].
7. DJI. “DJI Phantom 3 Series.” Internet: <http://www.dji.com/products/phantom-3-series>, [10/10/2015].
8. E. Yuan. “Eric Yuan’s Blog.” Internet: <http://eric-yuan.me/continuously-adaptive-shift>, [11/1/2015].
9. I. Gerg, A. Ickes, J. McCulloch. “Camshift Tracking Algorithm.” Internet: <http://www.gergltd.com/cse486/project5/>, [11/1/2015].
10. J. Allen, R. Xu, J. Jin. “Object Tracking Using CamShift Algorithm and Multiple Quantized Feature Spaces,” in Proceedings of the Pan-Sydney Area Workshop on Visual Information Processing, 2003, pp. 3-7.
11. J. Courbon, Y. Mezouar, N. Guenard, P. Martinet. “Visual Navigation of a Quadrotor Aerial Vechicle,” in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009, pp. 5315-5320.
12. M. Hammer. “Parrot AR.Drone Tutorial.” Internet: <http://robohub.org/tag/parrot-ar-drone-tutorial/>, [11/1/2015].
13. M. Ryosuke, K. Hirata, T. Kinoshita. “Vision-Based Guidance Control of a Small-Scale Unmanned Helicopter,” in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2007, pp. 2648-2653.
14. N. Papanikolopoulos, W. Beksi. CSCI 5551. Class Lecture, Topic: “Robot Hardware and Software.” Keller Hall 3-115, University of Minnesota, Minneapolis, MN, [10/15/2015].
15. O. Boyers. “An Evaluation of Detection and Recognition Algorithms to Implement Autonomous Target Tracking With a Quadrotor.” B.S. Thesis, Rhodes University, South Africa, 2013.
16. Open Source Robotics Foundation, “ROS IDEs.” Internet: <http://wiki.ros.org/IDEs>, [11/1/2015].
17. OpenCV. “Meanshift and Camshift.” Internet: <http://docs.opencv.org/master/db/df8/tutorial_py_meanshift.html#gsc.tab=0>, [11/1/2015].
18. P. Bristeau, F. Callou, D. Vissiere, N. Petit. “The Navigation and Control Technology Inside the AR.Drone Micro UAV,” in Proceedings of the 18th IFAC World Congress, 2011, pp. 1477-1484.
19. Parrot. “AR.Drone 2.0.” Internet: <http://ardrone2.parrot.com>, [10/10/2015].
20. Parrot. “Bebop Drone.” Internet: <http://www.parrot.com/products/bebop-drone>, [10/10/2015].
21. Pleiades. “Meet Spiri.” Internet: <http://pleiades.ca>, [10/10/2015].
22. S. Piskorski, N. Brulez, P. Eline, F. D’Haeyer. *AR.Drone Developer Guide SDK 2.0*. Dec. 2012.
23. S. Singh, B. Grocholsky, S. Nuske, M. Dille. “UAV / UGV Air-Ground Collaboration.” Internet: <http://www.frc.ri.cmu.edu/projects/agc/Overview.html>, [10/31/2015].
24. Simon Frasier University, “Autonomy Lab.” Internet: <http://autonomylab.org>, [11/1/2015].
25. Wiki. “HSL and HSV.” Internet: <https://en.wikipedia.org/wiki/HSL_and_HSV>, [11/1/2015].

**\*\*Remove this note after verifying all references are cited within the paper\*\***

## Resources

1. EZ-Robot, EZ-Builder, <https://www.ez-robot.com/EZ-Builder/windows>.
2. EZ-Robot, EZ-SDK, <https://www.ez-robot.com/EZ-Builder/sdk>.
3. EZ-Robot, EZ-SDK Mono, <https://www.ez-robot.com/EZ-Builder/mono>.
4. Itseez, Open CV, <http://opencv.org>.
5. Linus Torvalds, Git, <https://git-scm.com>.
6. Microsoft, Visual Studio, [https://www.visualstudio.com](https://www.visualstudio.com/).
7. Open Source Robotics Foundation, ROS, <http://www.ros.org>.
8. Parrot, AR.Drone SDK 2.0, <http://developer.parrot.com/ar-drone.html>.
9. Parrot, AR.FreeFlight, <https://play.google.com/store/apps/details?id=com.parrot.freeflight&hl=en>.
10. Simon Frasier University, AR.Drone Autonomy Lab, <http://autonomylab.org/ardrone_autonomy>.
11. The Eclipse Foundation, Eclipse IDE, <https://eclipse.org/ide>.
12. The Qt Company, Qt Creator, <http://www.qt.io/ide/>.

## Images

|  |  |  |
| --- | --- | --- |
| Figure | Image Title | Reference |
| Cover page | AR.Drone splash | <http://www.augmentedplanet.com/2011/11/parrot-ar-drone-review> |
| 1 | AR.Drone 2.0 | <http://ardrone2.parrot.com> |
| 2 | AR.Drone SDK Layered Architecture | AR.Drone SDK 2.0 Reference Manual |

## Terminology

|  |  |
| --- | --- |
| **Term** | **Description** |
| Client | Reference to remote application communicating with drone |
| COTS | Commercial Off-the-Shelf |
| DC | Direct Current |
| DIY | Do it yourself |
| DOF | Degrees of Freedom |
| Drone | Reference to UAV |
| FPS | Frames per second |
| Host | Reference to system containing client software (may be used in place of client) |
| HSL | Hue, saturation, lightness color model |
| HSV | Hue, saturation, value color model |
| LiPo | Lithium-ion Polymer [battery] |
| MEMS | Micro-electro-mechanical System |
| Node | ROS term for an executable that is connected to the ROS network |
| Open CV | Open source Computer Visualization software library |
| OEM | Original Equipment Manufacturer |
| OMAP | Open Multimedia Application Platform |
| PC | Personal Computer |
| Publisher | ROS term for node that sends messages on a specific topic |
| Quadcopter | Quadrotor UAV |
| ROS | Robotics Operating System |
| SDK | Software Development Kit |
| Subscriber | ROS term for node that listens for messages on a specific topic |
| Target | Mobile object tracked by drone |
| TI | Texas Instruments |
| Topic | ROS term for communication channel |
| UAV | Unmanned aerial vehicle |
| Wi-Fi | Wireless Fidelity (wireless communication standard) |