Tag Recognition for Quadcopter Drone Movement

by

Song Hui Lee
14862

Dissertation submitted in partial fulfilment of
the requirements for the
Bachelor of Engineering (Hons)
(Electrical & Electronic)

JANUARY 2015

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

Tag Recognition for Quadcopter Drone Movement

by

Song Hui Lee
14862

A project dissertation submitted to the
Electrical & Electronic Engineering Programme
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
BACHELOR OF ENGINEERING (Hons)
(ELECTRICAL & ELECTRONIC)

Approved by,

[illegible]

____

(AP Dr Mohamad Naufal B Mohamad Saad)

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK
January 2015

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

[illegible]

SONG HUI LEE

ABSTRACT

Unmanned Aerial Vehicle (UAV) drone such as Parrot AR.Drone 2.0 is a flying mobile robot which has been popularly researched for the application of search and rescue mission. In this project, Robot Operating System (ROS), a free open source platform for developing robot control software is used to develop a tag recognition program for drone movement. ROS is popular with mobile robotics application development because sensors data transmission for robot control system analysis will be very handy with the use of ROS nodes and packages once the installation and compilation is done correctly. It is expected that the drone can communicate with a laptop via ROS nodes for sensors data transmission which will be further analyzed and processed for the close-loop control system. The developed program consisting of several packages is aimed to demonstrate the recognition of different tags by the drone which will be transformed into a movement command with respect to the tag recognized; in other words, a visual-based navigation program is developed.

ACKNOWLEDGEMENTS

I would like to take this golden opportunity to express my sincere gratitude to all parties who have assisted me in completing my Final Year Project.

First and foremost, special thanks to my supervisor, AP Dr Mohamad Naufal B Mohamad Saad for giving me full support in terms of motivation, knowledge and valuable advices throughout the two semesters of my project. His expertise and patience have granted me countless opportunities to explore, learn and experience new things in the field of robotics research.

Apart from that, my project will not be able to be accomplished without the assistance from Centre for Intelligent Signal and Imaging Research (CISIR) for borrowing me the UAV drone for experimentation purpose. Their kindness and hospitality gave me a comfortable and wonderful working environment in which I am able to work without burden and stress. I am also very grateful to Lee Kian Seng for assisting me in the software installation part in the early stage of my project with his experience in Robot Operating System.

Last but not least, many thanks to Final Year Project committees for the great arrangement of activities and funding which ensures the success and smooth flow of the project delivery.

TABLE OF CONTENTS

CERTIFICATION OF APPROVAL i
CERTIFICATION OF ORIGINALITY ii
ABSTRACT iii
ACKNOWLEDGEMENTS iv
TABLE OF CONTENTS v
LIST OF FIGURES vi
LIST OF TABLES vii
LIST OF ABBREVIATIONS vii

CHAPTER 1 : INTRODUCTION **1**
1.1 BACKGROUND 1
1.2 PROBLEM STATEMENTS 2
1.3 OBJECTIVES 3
1.4 SCOPE OF STUDY 5

CHAPTER 2 : LITERATURE REVIEW & THEORY **6**
2.1 CRITICAL LITERATURE ANALYSIS 6
2.2 QUADCOPTER DRONE DYNAMICS 9
2.3 PARROT AR.DRONE 2.0 SPECIFICATIONS 13
2.4 ROBOT OPERATING SYSTEM (ROS) 14
2.5 RECENTNESS OF LITERATURE SOURCES 15

CHAPTER 3 : METHODOLOGY OF PROJECT WORK **16**
3.1 RESEARCH METHODOLOGY 16
3.2 KEY MILESTONES 18
3.3 GANTT CHART 20
3.4 TOOLS & EQUIPMENT 21

CHAPTER 4 : RESULTS AND DISCUSSION **25**
4.1 DRONE COMMUNICATION & KEYBOARD TELE-OPERATION 26
4.2 TAG RECOGNITION PACKAGE 32
4.3 VISUAL-BASED DRONE MOVEMENT 36

CHAPTER 5 : CONCLUSION & RECOMMENDATION **39**
5.1 CONCLUSION 39
5.2 FUTURE RECOMMENDATION 40

REFERENCES **41**
APPENDIX A **43**

LIST OF FIGURES

FIGURE 1.1	Parrot AR.Drone 2.0	1
FIGURE 2.1	Aircraft Classification Based On Principle of Flight & Propulsion Mode	6
FIGURE 2.2	Quadcopter Drone Three Main Control Axes (Yaw, Pitch, Roll)	9
FIGURE 2.3	Angular Velocity, Torques and Accelerations Produced by Each Motor While Hovering	10
FIGURE 2.4	Simplified Diagram Representing Angular Velocity of Each Motor	10
FIGURE 2.5	Angular Velocity of Motors for Vertical Movement	11
FIGURE 2.6	Angular Velocity of Motors for Yaw Movement	11
FIGURE 2.7	Angular Velocity of Motors for Pitch Movement	12
FIGURE 2.8	Angular Velocity of Motors for Roll Movement	12
FIGURE 2.9	UAV Drone 1000mAh LiPo Battery	14
FIGURE 3.1	Research Methodology Flow Diagram	16
FIGURE 3.2	Specification of Laptop Used For the Project	21
FIGURE 3.3	Laptop Used For the Project	21
FIGURE 3.4	Linux Ubuntu 12.04	22
FIGURE 3.5	ROS Versions	22
FIGURE 3.6	Full Set of Parrot AR.Drone 2.0	23
FIGURE 3.7	Extra Drone Batteries for Experimentation Purpose	23
FIGURE 3.8	Fabricated Tags for Drone Movement	24
FIGURE 3.9	Spacious Hall for Drone Experimentation	24
FIGURE 4.1	ROS Packages Used	25
FIGURE 4.2	Running Drone Driver (ardrone_driver)	26
FIGURE 4.3	Information Displayed in Navdata Topic	27
FIGURE 4.4	Keyboard Tele-Operation Key Function Description for Drone Navigation Control	28
FIGURE 4.5	Drone Flight Test (Outdoor)	29
FIGURE 4.6	Drone Flight Test (Indoor)	29
FIGURE 4.7	Single Quadcopter in V-REP Simulator Environment	30
FIGURE 4.8	Two Quadcopters in V-REP with Two Camera Views	31
FIGURE 4.9	Activated 12 Models of Tag in ar_recog Package	33
FIGURE 4.10	Tag Information From echo /tags	34
FIGURE 4.11	Recognized Tag from Drone's Front Camera View	34
FIGURE 4.12	Condition for Recognized Tag from Drone's Front Camera View	36
FIGURE 4.13	Final Demo Video Uploaded In YouTube	37
FIGURE 4.14	Drone Flight Workstation Terminal Screen	38
FIGURE 4.15	Drone Flight Experimentation with Tags	38

LIST OF TABLES

TABLE 4.1 Tags and the Movement Command Assigned 35

LIST OF ABBREVIATIONS

ROS	Robot Operating System
UAV	Unmanned Aerial Vehicle
VTOL	Vertical Take Off and Landing
V-REP	Virtual Robot Experimentation Platform

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND

Unmanned Aerial Vehicle (UAV) is a flying robot without human on-board. It is also known as quadrotor or quadcopter which has Vertical Take Off and Landing (VTOL) ability with four rotors. The low cost UAV Parrot AR.Drone 2.0 which is initially designed for gaming purpose running on Android operating system will be used in this project. Basically, AR.Drone 2.0 has a robust structure design which comes with HD video recording camera, extreme precision control and automatic stabilization features on-board technology [1]. This quadcopter model is very famous among researchers since it is compatible with Robot Operating System (ROS) which is a free, widely used platform for developing robot control software. The quadcopter AR.Drone 2.0 used in this project will be addressed as "drone" in the later part of the report.

FIGURE 1.1 Parrot AR.Drone 2.0

A drone can be used for search and rescue purpose ranging from remote area such as jungle to destruction or disaster area. This will be more effective when a few drones carry out a common mission simultaneously. Once possible humans are detected, the coordinates of the victim can be sent to the rescue team. This will eventually reduce the surveillance cost, manpower, time and overcome the risk and capability of human to go personally into the surveillance area.

Apart from that, security is the top concern of every nation. Most crimes occurred in places without surveillance camera because it is not economical to install cameras in every corner which will require high installation and maintenance cost. Therefore, an UAV that does patrol on regular basis covering large area is a more idealistic idea. Additionally, an UAV can monitor the area with no view angle limitation since it can maneuver around. Upon detection of a crime, the security personnel can track and make the UAV chase after the criminal and constantly report his coordinates via the built-in GPS. Hence, UAV surveillance implementation can reduce cost while improving the security.

The implementation of UAV drone application as mentioned above can be developed by various approaches in terms of modeling the control system in multiple operating systems and platforms choices. However, a successful and strong foundation of the project will require years of continuous research and studies to be carried out. Therefore, as a head start for this goal, it is wise to choose a good platform to start with such as Robot Operating System (ROS) by focusing on a small area of UAV drone implementation followed by advancement in the project from time to time.

1.2 PROBLEM STATEMENTS

a) ***Ground mobile robot maneuverability limits the application of robotics in daily life.***

Many applications have been applied using the ground mobile robots in the past decades. However, as the technology advances and expectation grows, it is realized that there are many limitations in terms of the ground mobile robots mobility as compared to the flying robot. With the ability to hover in the air, the flying robot or drone has higher degree of freedom to carry out complex flying forms (pitch, roll, yaw, flip, etc.) which enables it to go through narrow openings. The drone capability in terms of maneuverability can overcome the incapability of human in the case of search and rescue in a building on fire, earthquake destroyed buildings or any other similar situation. Also, it can be used to transport medical aids to emergency patients in the rural area since travelling by air can be much faster with no congested traffic.

b) ***Manual navigation control of the robots requires operators constant monitoring.***

   The typical control of a robot is through the manual control either using the joystick or keyboard teleoperation, monitored by an operator as the decision maker. Hence, in the case where hundreds of robots are given a large-scale mission, numerous operators will be required to monitor the robot navigation which is not an efficient control. Therefore, to eliminate the need of operators as the decision maker for the robot navigation, it is best to develop a closed-loop control to have the robot acts as the brain to decide its own navigation through the autonomous control with the environment as the input element. For instance, visual input through the robot camera either in the form of fabricated tag, object or human recognition.

1.3 OBJECTIVES

The main objective of this project is to develop flying robot control software to communicate with the drone using a laptop by extracting the data from the sensors mounted on the UAV drone to perform tag recognition program. The objectives of this project are:

a) ***To establish communication between the drone and laptop to assist drone navigation control using Robot Operating System (ROS).***

   Parrot AR.DRONE 2.0 with Wi-Fi is to be used. The idea is to connect the drone to a workstation so that signals can be transmitted between them. Variety of data can be transmitted from different kind of sensors mounted on the drone to the laptop. Also, control commands can be sent via Wi-Fi to the drone to move it around or to command certain sensors to retrieve data.

b) ***To retrieve data from the sensors of drone.***

   There are numerous sensors attached to the drone. These sensors include inertia measurement unit (imu), ultrasonic sensor and camera which provide very ironic information. Being able to read the values from the sensors allow decisions to be made to increase or decrease the speed of certain motors to stabilize the flight. Also, camera allows vision-based navigation control such as fabricated tag recognition.

c) ***To process the data from the sensors of drone.***

Sensors like gyroscope and magnetometer can be used to estimate the velocity, orientation and position of the drone, which can be used for semi-autonomous navigation that helps in the control of a drone. The front camera provides very rich information which can be used to identify different types of tags with different orientations and thus, making control decision based on the objects detected. A more advanced use of a camera is to identify features or edges which can be used for obstacles avoidance or localization of drone.

d) ***To develop an autonomous control program based on the sensors data.***

Having the sensors data analyzed and processed, decisions based on the retrieved data can be decided by the robot. This includes a program that processes the sensors data and gives selected information as the output. On the other hand, another control program should then extract this output as the input of its control and sends corrective commands to the actuators or motors after comparing with the desired response of the drone.

e) ***To demonstrate an autonomous drone movement based on tag recognition.***

An intelligent system of giving command to the drone movements (left, right, up, down) simply through the visual of the recognized tag from the drone's front camera can be demonstrated by data extraction, data processing and decision making based on the information from the sensors. It uses the camera image stream to identify the identity of the tag, whether the tag has been initialized to be part of the system and if yes, differentiates the tags with the respective indicated navigation command for the drone movement in the program. Hence, an autonomous drone movement based on visual control (tag) is expected to be achieved.

1.4 SCOPE OF STUDY

This project involves the knowledge in the mobile robotics research specifically flying robot (drone) and Robot Operating System platform, specializing in the area of automation and control systems. The scope of study mentioned is highly relevant to the author because the author is majoring in Instrumentation and Control Systems in her degree of Electrical & Electronics engineering and has past experience in using the ROS platform during her internship which is related to robotics research in ground mobile robots. Therefore, given the time frame of two semesters (28 weeks), the project is highly feasible since the author will have plenty of time to study the dynamics of UAV drone and tag recognition program with the implementation of ROS platform. Also, the availability of the drone Parrot AR.Drone 2.0 in the lab can help the author to have an early start with the experimentation.

CHAPTER 2

LITERATURE REVIEW

2.1 CRITICAL LITERATURE ANALYSIS

Mobile robots have been the target agent for rescue purpose since year 2001 after the incident of World Trade Disaster in New York [2]. Mobile robots serving as the rescue agent can replace the risky job of the rescuers or trained dog in the case of searching victims in the earth quake or other disaster affected area [2], [3], [4], [5], [6]. Many studies has been done and it is learned that flying mobile robots can do a better job compared to ground mobile robots when it comes to the emergency rescue since it has a better freedom of movement [2], [3], [4], [5], [6], [7]. In general, flying mobile robot is a good platform for surveillance and common situation assessment in safety, security, and rescue missions [5], [6], [7].

Flying mechanism can be classified based on the principle of flight and propulsion mode [7], [8]. FIGURE 2.1 shows the categorization of the flying mechanisms.

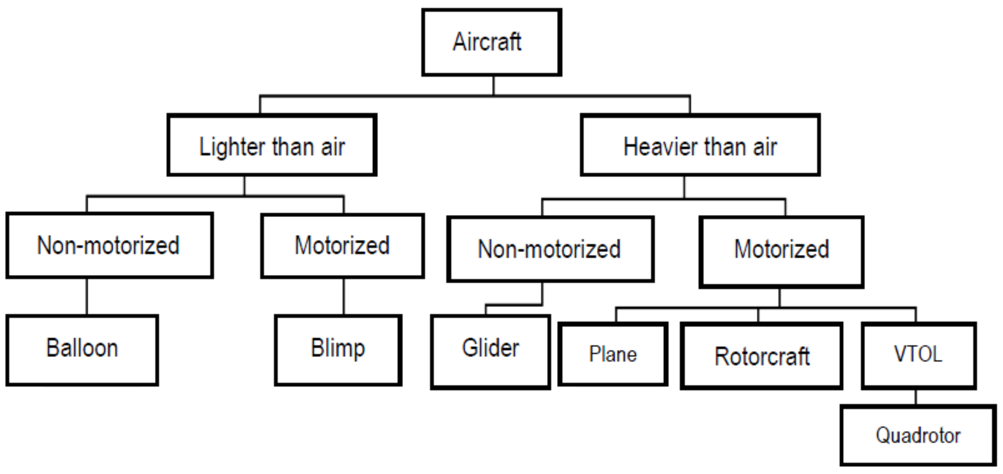


FIGURE 2.1 Aircraft Classification Based On Principle of Flight & Propulsion Mode

Quadcopter such as AR.Drone 2.0 is one of the UAVs with Vertical Take Off and Landing (VTOL) ability [5], [7]. In order for the UAV to perform search and rescue mission, the robot needs to be able to localize itself and move around autonomously. A direct approach on the drone localization is studied in [9] where the drone position is estimated using the dead reckoning method. This approach is not practical since the drone can only move autonomously on the learned path with limited coverage. Hence, drone localization for autonomous movement can be achieved better through visual-based approach such as ArUco visual markers in [10], [11], artificial vision in [12] and landmarks such as tag recognition in [13]. The implementation of localization using artificial approach in [12] with Astec Pelican drone is able to take off, navigate, land and recharge its battery by using a developed landing platform autonomously which is a great idea in overcoming the short battery life of drones without the need to change the battery manually.

The mission of search and rescue will be more efficient if more than one drone is performing the task. The issue of controlling a swarm of drones and its communication has been discussed in [5], [10], [11], [14] and in these papers, AR.Drone UAV is used and controlled using the packages in ROS. The drone control approach in [14] involving VICON Camera System inquires more cost as compared to the approaches in [5] that uses self-deployment capability based on Takahashi Self Deployment algorithms and [10], [11] that works on separate modules such as localization, obstacle avoidance and partner detection.

Various recognition methods ranging from ArUco visual markers [10], [11], [15], fabricated tag recognition [13], objects recognition [4], [6], [16], [17] to human recognition [2], [3] have proven to be efficient. These recognition methods are important to ensure that the drone has a specific target to track on for the mission of search and rescue. The ArUco visual markers are recognized as obstacles to be avoided in [10], [11] while in [15], ArUco is recognized as a target to be followed both in the real environment as well as in the virtual simulator tool V-REP. Collision avoidance can be achieved either using ArUco visual markers in [10], [11] or FastSLAM approach as in [18]. As for the fabricated tag recognition in [13], the parameters used for recognition is solely based on the drone’s downward facing camera and it is proven to be an excellent method of localization with the absence of GPS data.

The object recognition for the case of military surveillance [4] uses Adaboost Classifier and Pinhole Algorithm has 71% object detection accuracy utilizing ROS platform. Moving object can also be recognized through the tracking of motion in a captured scene with either improved Fourier Mellin Invariant method (iFMI) in [6] or MATLAB algorithm [17]. The search of living human using a ground mobile robot [2] uses passive infrared sensor (PIR) to detect body heat radiation as the first level inspection followed by neural networks technique as the second level inspection to confirm the existence of a human shape before contacting the rescue team. The similar approach is explained in [3] with more explanation on the possible drawbacks.

For a higher accuracy of recognition regardless of human, object or tag, the drone must always be in the stable flight state [19] where four ultrasonic sensors are mounted below the blades to keep track of a fixed distance from the ground. Hence, a drone tends to have a stable hovering state while capturing images for recognition process. In the outdoor environment, the flight accuracy decreases due to the weather factor and an intelligent wind detection system [20] proves that three detection equations could help to improve the reliability of wind adaptability of the drone.

Although a drone is mechanically simpler than an ordinary helicopter, it is inherently unstable. [21] Hence, its control is rather difficult as compared to ground robots. A beginner in drone research will eventually have to consume a lot of time and effort in solving several implementation and ‘low level’ issues.

2.2 QUADCOPTER DRONE DYNAMICS

A drone consists of four rotor blades and each of these rotors produces a certain air flow which in turn creates an acceleration force onto the body of the drone. The resultant force of all four acceleration moves the drone around. [22] FIGURE 2.2 shows the three main control axes of the drone motion.

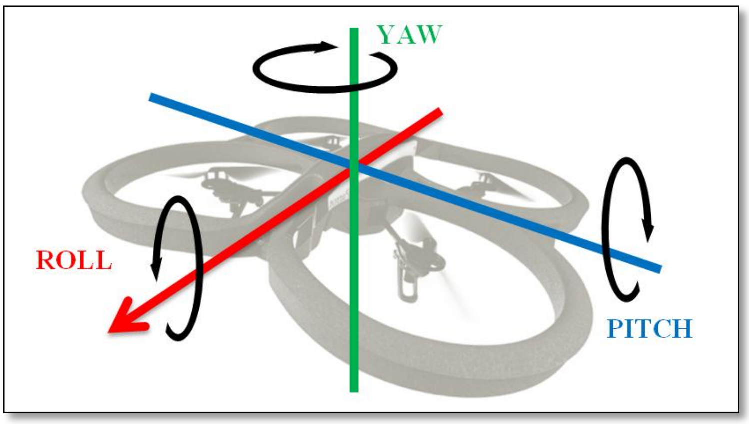


FIGURE 2.2 Quadcopter Drone Three Main Control Axes (Yaw, Pitch, Roll)

- **Hovering On The Same Spot**

  When all of the four motors rotate at a certain speed, the drone takes off and if the thrust of all the four motors combined together exactly compensates for earth gravity, then the drone will vertically stay in place. The sum of all the four motors torque must be exactly zero for the drone to hover on the same spot. Otherwise, a contour-torque will be induced which would start to rotate the drone around its yaw axis.

  This is physically implemented with two diagonal motors rotate in the same direction and the other two motors rotate in the opposite direction. In this way, the torque of all the four rotors will sum to zero while the trust compensates for earth gravity in hovering mode. FIGURE 2.3 shows the angular velocity (represented by small, black arrow), torques (represented by red, thick arrow) and forces (represented by green arrow) generated by each rotating rotors. A simplified diagram with arrow corresponds to the angular velocity of the rotors is in FIGURE 2.4.

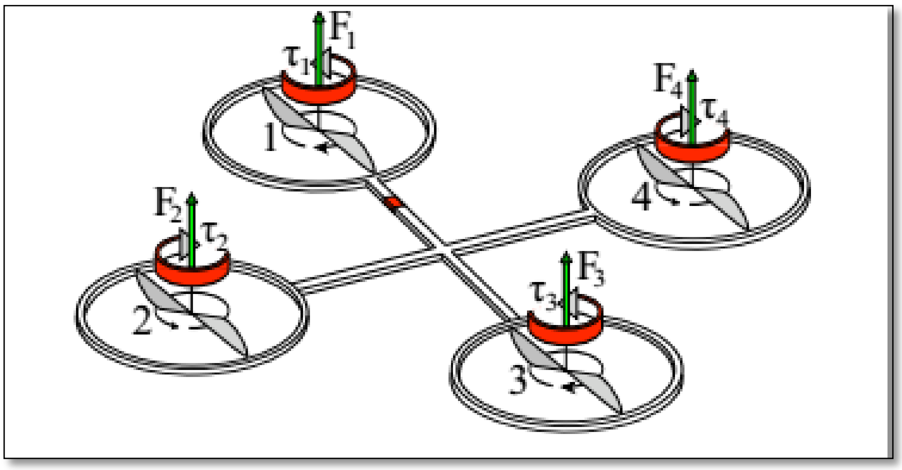


FIGURE 2.3 Angular Velocity, Torques and Accelerations Produced by Each Motor While Hovering

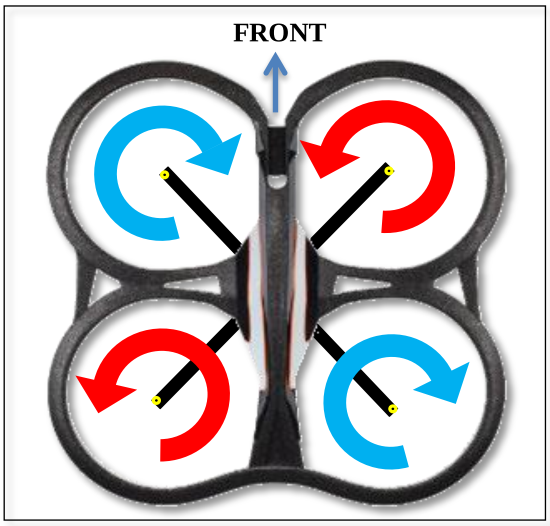


FIGURE 2.4 Simplified Diagram Representing Angular Velocity of Each Motor

- **Vertical Movement**

  In order for the drone to move up vertically, the angular velocity of all the four motors shall be increased equally as shown in FIGURE 2.5 (a). On the other hand, the speed of all four motors shall be reduced equally as in FIGURE 2.5 (b) to reduce the overall thrust, which will bring down the drone vertically. For keeping the orientation, it is important that the torques of all four motors sum exactly to zero.

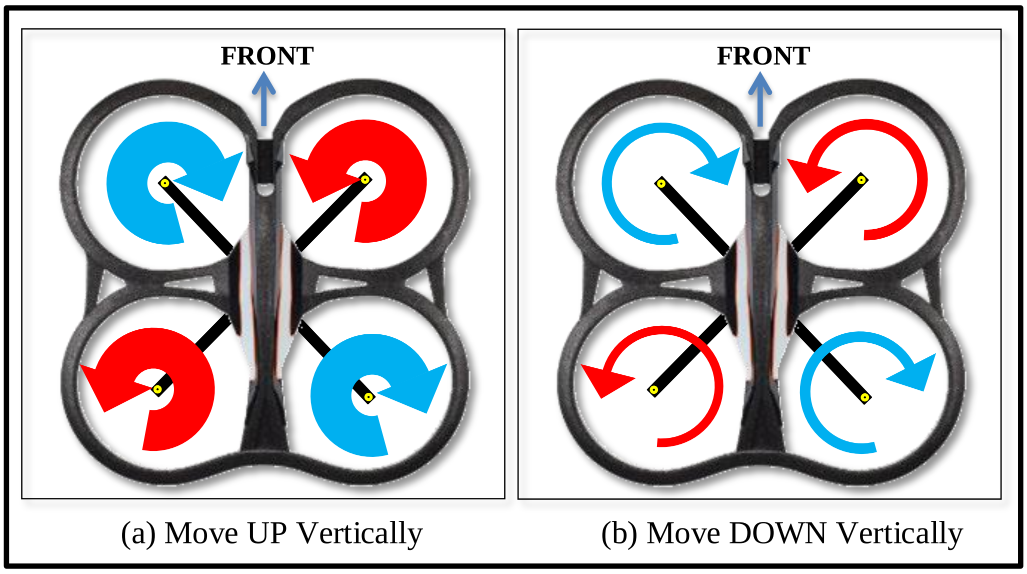


FIGURE 2.5 Angular Velocity of Motors for Vertical Movement

- **Yaw Movement**

  To make the drone rotate in the right direction, increase the speed of the anti-clockwise rotating motors, which induces clockwise torque as shown in FIGURE 2.6 (a). On the other hand, make the drone rotate in the left direction, increase the speed of the clockwise rotating as shown in FIGURE 2.6 (b). Note that to make sure the drone keep its vertical position, the overall thrust of the four motors exactly equals earth gravity.

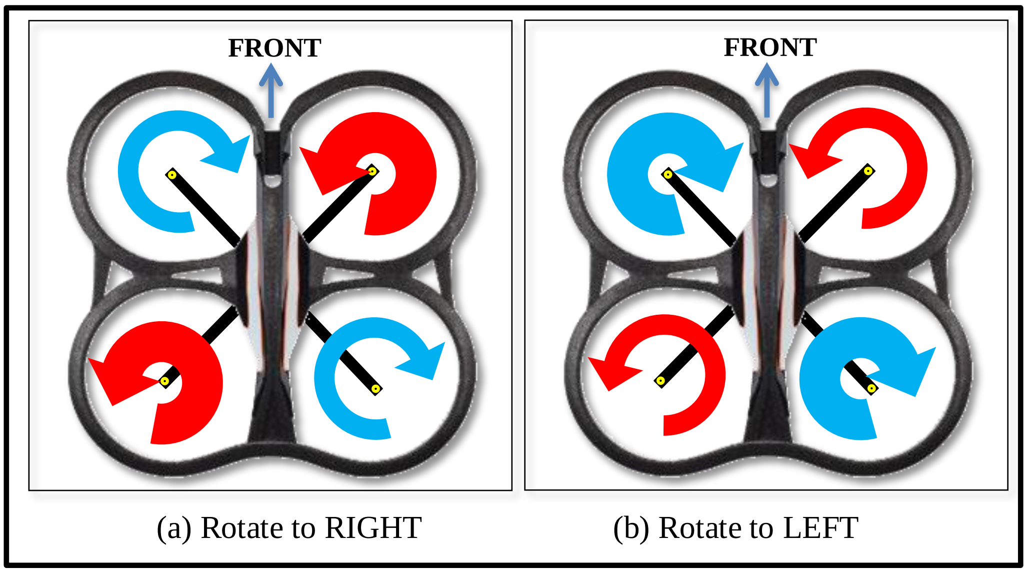
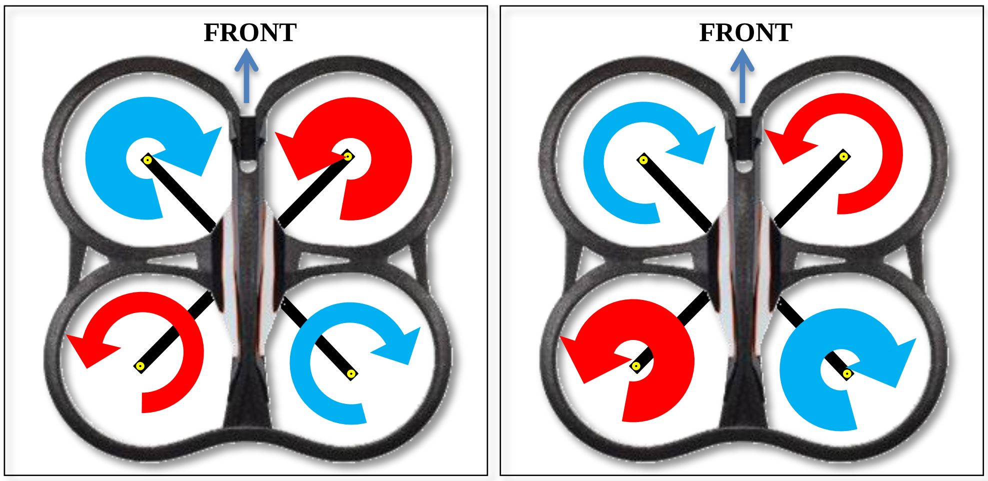


FIGURE 2.6 Angular Velocity of Motors for Yaw Movement

- **Pitch Movement**

  To pitch backward, reduce the speed of the back motors and increase the speed of the front motors as in FIGURE 2.7 (a). In this way, the drone will tilt backward and then lead to acceleration in the backward direction. Similarly, to pitch forward, decrease the speed of the front motors and increase the speed of the back motors as in FIGURE 2.7 (b). That will tilt the drone forward and then lead to a horizontal forward acceleration.

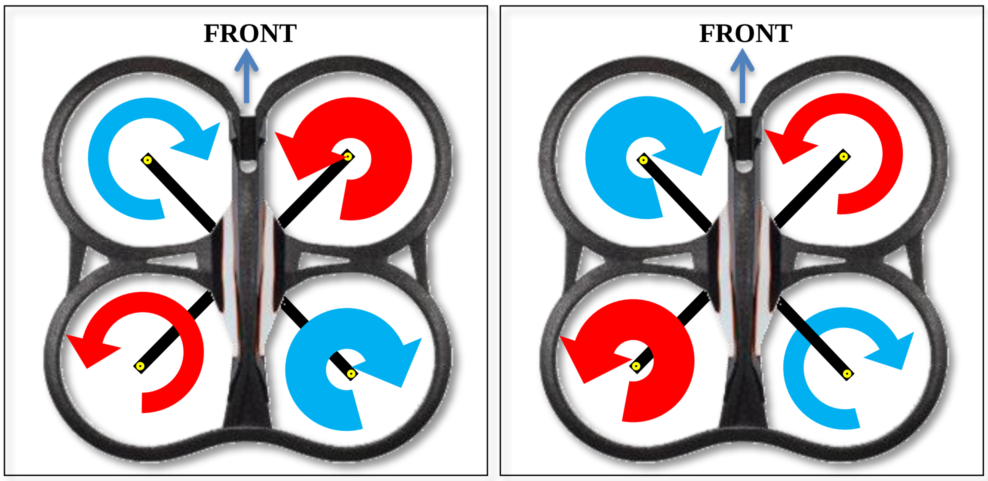


(a) Pitch BACKWARD (b) Pitch FORWARD

FIGURE 2.7 Angular Velocity of Motors for Pitch Movement

- **Roll Movement**

  Similar to the pitch movement concept, the drone can be moved to the left and to the right as in FIGURE 2.8 (a) and FIGURE 2.8 (b) respectively.



(a) Move LEFT (b) Move RIGHT

FIGURE 2.8 Angular Velocity of Motors for Roll Movement

2.3 PARROT AR.DRONE 2.0 SPECIFICATIONS

The unmanned aerial vehicle that is to be hacked with ROS is manufactured by a French company named Parrot. The specifications were looked into to have better understanding of the robot. The detailed technical specification of Parrot AR.Drone 2.0 can be referred in *APPENDIX A*.

- **Motion sensors**

  Parrot AR.Drone 2.0 comes with many motions sensors and they are located below the central hull. Basically, it features a 6 degree of freedom (DOF), micro-electro-mechanical (MEMS)-based, miniaturized inertial measurement unit. It provides the software with pitch, roll and yaw measurements. On the other hand, ultrasound telemeter provides the altitude measurement for altitude stabilization and assisted vertical speed control whereas pressure sensor allows altitude measurements at any height. Lastly, the down camera provides the drone with ground speed measurements for hovering and trimming.

- **Engines**

  The drone is powered with brushless engines with three phase current controlled by a micro-controller. It detects if all the engines are turning or are stopped. In case a rotating propeller encounters any obstacle, the drone will detect if there is any of the propellers is blocked and stops all engines immediately. This protection system prevents repeated shocks.

- **Video Streaming & Tags Detection**

  The front camera is a CMOS sensor with a 92 degrees angle lens (field of view). The drone uses 720p (1280x720) image resolutions and the video stream frame rate can be adjusted between 15 and 30 FPS.

- **Parrot AR.Drone Software Development Kit**

  The manufacturer (Parrot) releases the software development kit (SDK) to the public to create more applications. The SDK allows users to write their own applications to remotely control the drone from any Linux personal computer with Wifi connectivity.

- **LiPo Batteries**

  The drone set comes with 1000mAh, 11.1V LiPo battery to fly. While flying, the battery voltage decreases from full charge (12.5V) to low charge (9V). The microcontroller monitors battery voltage and converts into a battery life percentage (100% when battery is full, 0% when battery is low). When the drone detects a low battery voltage, it will first send a warning message to the user, and then lands automatically. If the voltage reaches a critical level, the whole system will shut down to prevent any unexpected behavior. This battery capacity can last the drone flight in between 10-15 minutes duration.

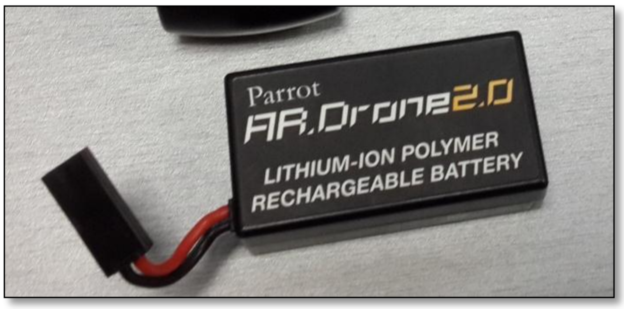


FIGURE 2.9 UAV Drone 1000mAh LiPo Battery

2.4 ROBOT OPERATING SYSTEM (ROS)

Robot Operating System (ROS) is a collection of software frameworks for robot software development, providing operating system-like functionality on a heterogeneous computer cluster. [23] ROS provides standard operating system services such as hardware abstraction, low-level device control, implementation of commonly used functionality, message-passing between processes, and package management. Running sets of ROS-based processes are represented in a graph architecture where processing takes place in nodes that may receive, post and multiplex sensor, control, state, planning, actuator and other messages. Despite the importance of reactivity and low latency in robot control, ROS, itself, is not a Realtime OS, though it is possible to integrate ROS with realtime code.

2.5 RECENTNESS OF LITERATURE SOURCES

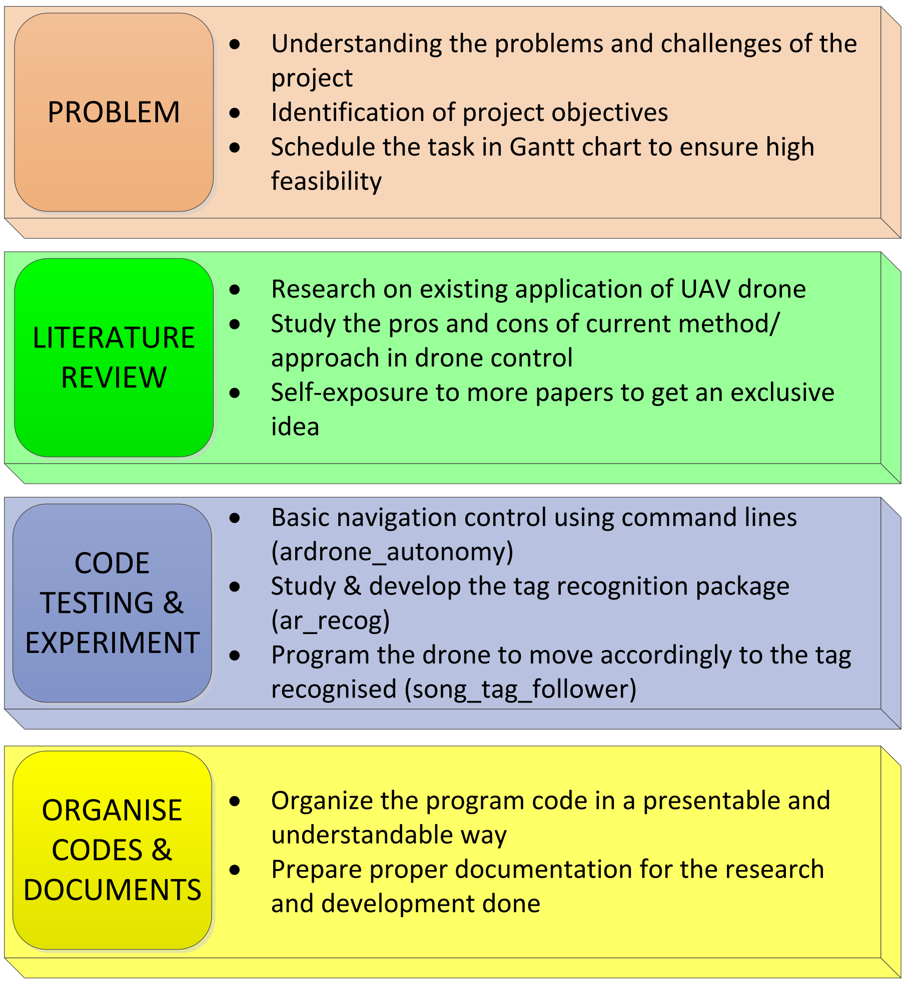
There are a total 23 literature references used in this extended proposal report and all these references are very recent where the year of publication of the papers ranges from year 2009 up to year 2014. The references used are highly reliable since the source or publisher is either from IEEE, International Journal of Computer Applications or valid official websites. From the references used, there are one 2009 paper, one 2010 paper, four 2011 papers, five 2012 papers, three 2013 papers, seven 2014 papers and two official website which is originated in year 2014. Therefore, the literature review sources used in this report is recent and reliable.

CHAPTER 3

METHODOLOGY OF PROJECT WORK

3.1 RESEARCH METHODOLOGY

The basic framework of the research methodology for this project is described as in FIGURE 3.1.

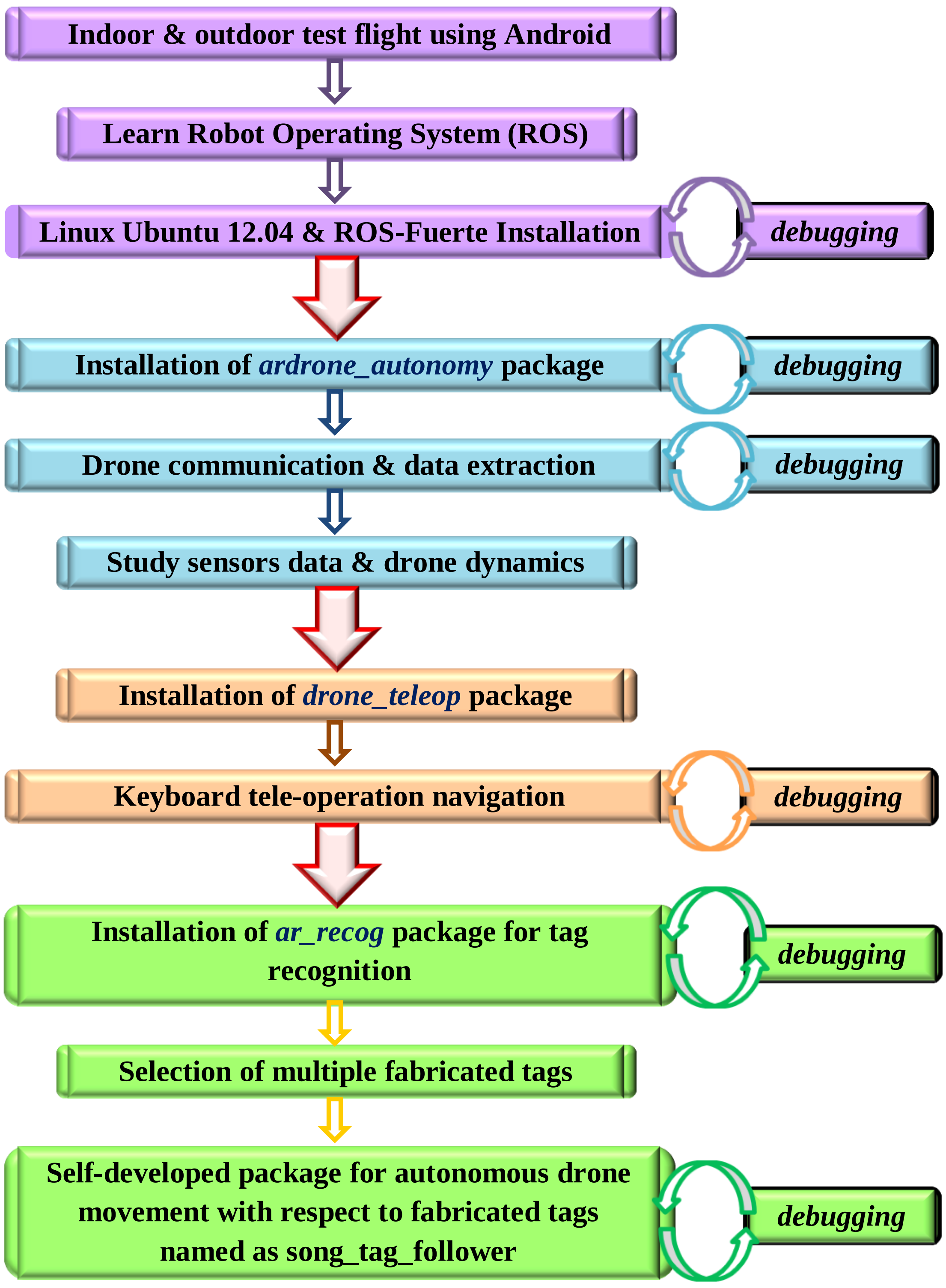


PROBLEM
Understanding the problems and challenges of the project
Identification of project objectives
Schedule the task in Gantt chart to ensure high feasibility
LITERATURE REVIEW
Research on existing application of UAV drone
Study the pros and cons of current method/ approach in drone control
Self-exposure to more papers to get an exclusive idea
CODE TESTING & EXPERIMENT
Basic navigation control using command lines (ardrone_autonomy)
Study & develop the tag recognition package (ar_recog)
Program the drone to move accordingly to the tag recognised (song_tag_follower)
ORGANISE CODES & DOCUMENTS
Organize the program code in a presentable and understandable way
Prepare proper documentation for the research and development done

FIGURE 3.1 Research Methodology Flow Diagram

With the brief idea on how the project is to be conducted, time management is highlighted as to ensure that the project objectives are feasible within the time frame of 28 weeks.

The technical methodology of the project is divided into four parts as indicated by different colors in the flowchart. The first part is to acquire basic knowledge on Robots Operating System (ROS) and the UAV drone. Second part is to establish the communication between the workstation and the drone. Third part is to perform tele-operation of the drone. Lastly, it is the most complex part of the project that is to design an autonomous tag recognition program for the drone movement.


Indoor & outdoor test flight using Android
Learn Robot Operating System (ROS)
Linux Ubuntu 12.04 & ROS-Fuerte Installation
debugging
Installation of *ardrone_autonomy* package
debugging
Drone communication & data extraction
debugging
Study sensors data & drone dynamics
Installation of *drone_teleop* package
Keyboard tele-operation navigation
debugging
Installation of *ar_recog* package for tag recognition
debugging
Selection of multiple fabricated tags
Self-developed package for autonomous drone movement with respect to fabricated tags named as song_tag_follower
debugging

3.2 KEY MILESTONES

The key milestones of the project are as follows:

- **Installation of the drone driver**

  Having the driver successfully compiled in ROS (ardrone_autonomy package) marks the first step of the project. It means the communication between the workstation and UAV is achieved. The sensors data can be retrieved. Different nodes and topics are readily accessible.

- **Keyboard tele-operation of drone**

  Keyboard tele-operation requires a program that recognizes the buttons pressed on keyboard and sends control commands to the node that control the movement of the drone. All the four degrees of freedom (pitch, roll, yaw and acceleration along vertical axis) are able to be controlled from keyboard.

- **Compilation of tag recognition package**

  Once the data from the sensors can be retrieved and control commands can be sent, the next milestone is to compile the tag recognition package. The tag is represented in pixel matrix. This software uses the front camera of the drone as input, compares the image stream with previously stored pixel matrix to search for the tag and outline the tag with green line.

- **Creation of package to autonomously control drone movement based on the tag recognition**

  After confirming the performance of the tag recognition program, more tags of different identity are initialized to indicate different drone movement commands (left, right, up, down). The identity of the tags is identified accurately via the echo/ tags info. Hence, an autonomous control program based on the visual-based tag tracked by the drone’s front camera image stream can be developed. This requires the development of a closed loop control system program to navigate the drone movement based on the identity of the tag recognized. The package is created with the support of the information extracted from drone_teleop, ar_recog and ardrone_autonomy packages. The program parameters are improved by enhancing the movement control performance of the drone flight through numerous experimentations.

- **Project documentation for future work continuation**

  All the crucial steps in achieving the final goal of tag recognition for drone movement are well documented both in written guidelines and video. The video is uploaded in YouTube for easier access to future users as well as to give a better illustration of the project outcome. With proper documentations, all the work will not be wasted and can be appreciated by future user where they can start working on the project enhancement from the point my project ended.

3.3 GANTT CHART

FINAL YEAR PROJECT SEMESTER	**Final Year Project 1 (22nd September 2014 – 26th December 2014)**														**Final Year Project 2 (12th January 2015 – 17th April 2015)**														
WEEK NO / TASK	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
Selection of Project Topic	🟥	🟥																											
Preliminary Research Work		🟥	🟥	🟥	🟥																								
Submission of Extended Proposal						✓																							
Proposal Defense								✓	✓																				
•Drone flight testing using Android Apps •ROS installation for research platform •Perform drone communication with laptop for navigation control using ROS •Purchase drone extra battery (2 units) •Learn V-REP virtual platform simulator							🟥	🟥	🟥	🟥	🟥	🟥																	
Submission of Interim Draft Report													✓																
Submission of Interim Report														✓															
•Program fabricated tags for drone tag recognition (ar_recog) •Create a package to differentiate recognized tags for the drone navigation movement control (song_tag_follower)															🟦	🟦	🟦	🟦	🟦	🟦	🟦								
Submission of Progress Report																						✓							
•Prepare detailed documentation of the project •Finalize the programming code & folder •Prepare poster and presentation slides																						🟦	🟦	🟦	🟦	🟦			
Pre-SEDEX																									✓				
Submission of Draft Final Report																										✓			
Submission of Dissertation (soft bound)																											✓		
Submission of Technical Paper																											✓		
Viva																												✓	
Submission of Project Dissertation (hard bound)																													✓

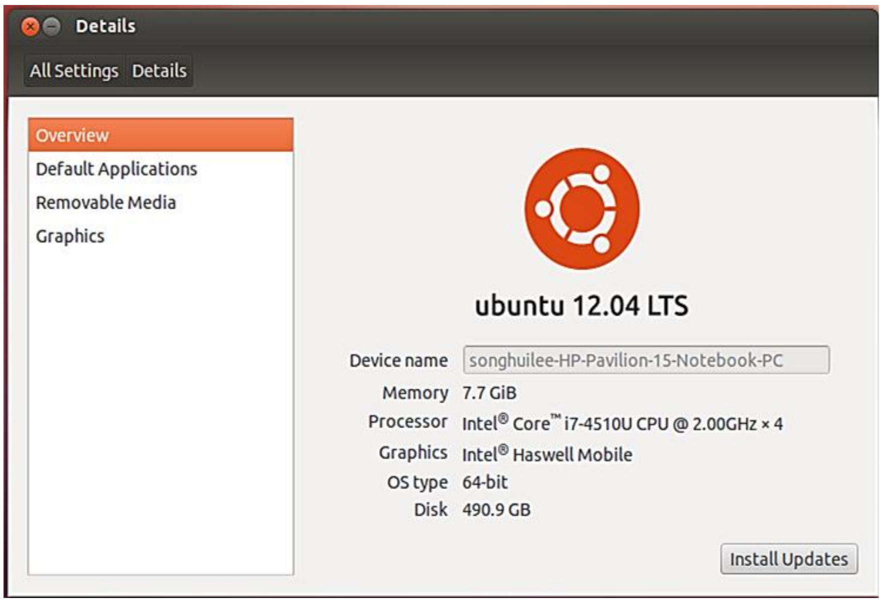
✓ *Important dates (Submissions/presentation)*

3.4 TOOLS & EQUIPMENT

This section states the tools and equipment needed throughout the project.

- **Workstation**

  A laptop is required because the experiments are conducted in a spacious area where it eases the mobility of the experimentations. The specification of the laptop is shown in FIGURE 3.2. The specification of the laptop for the current time is sufficient to support image processing and commands computation for this project. FIGURE 3.3 is the laptop used.



Device name	songhuilee-HP-Pavilion-15-Notebook-PC
Memory	7.7 GiB
Processor	Intel® Core™ i7-4510U CPU @ 2.00GHz × 4
Graphics	Intel® Haswell Mobile
OS type	64-bit
Disk	490.9 GB

FIGURE 3.2 Specification of Laptop Used For the Project

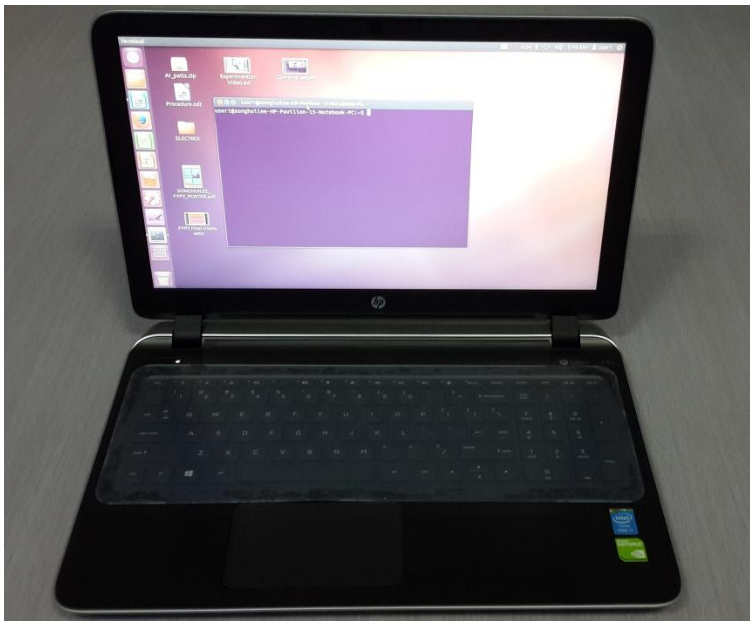


FIGURE 3.3 Laptop Used For the Project

- **Workstation Operating System And Software**

  The operating system used is Ubuntu 12.04 LTS which is an updated and stable version of Ubuntu as shown FIGURE 3.4.

FIGURE 3.4 Linux Ubuntu 12.04

The Robot Operating System (ROS) installed is Groovy version as shown in FIGURE 3.5 (a). However, due to some compatibility issues with tag recognition package, its predecessor, Fuerte version is then used as shown FIGURE 3.5 (b).

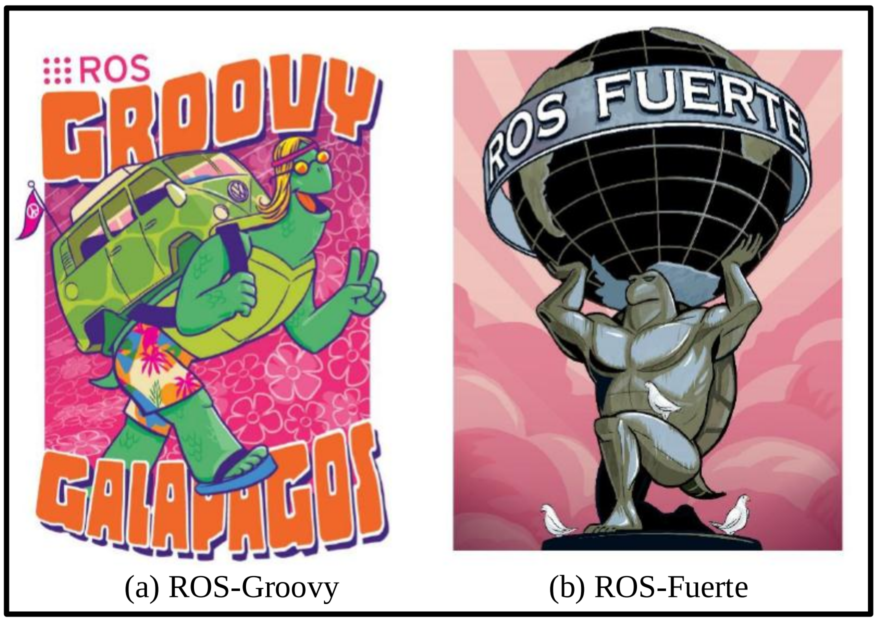


FIGURE 3.5 ROS Versions

- **Parrot UAV Drone Robot**

  The flying robot used is Parrot AR.Drone 2.0 and the full set of the drone with its outdoor hull, indoor hull, battery with the charger, stcikers and manuals are shown in FIGURE 3.6. The drone set belongs to the university property and the approximate value is MYR 1,000 since it comes with numerous sensors which can be utilized for advance applications.

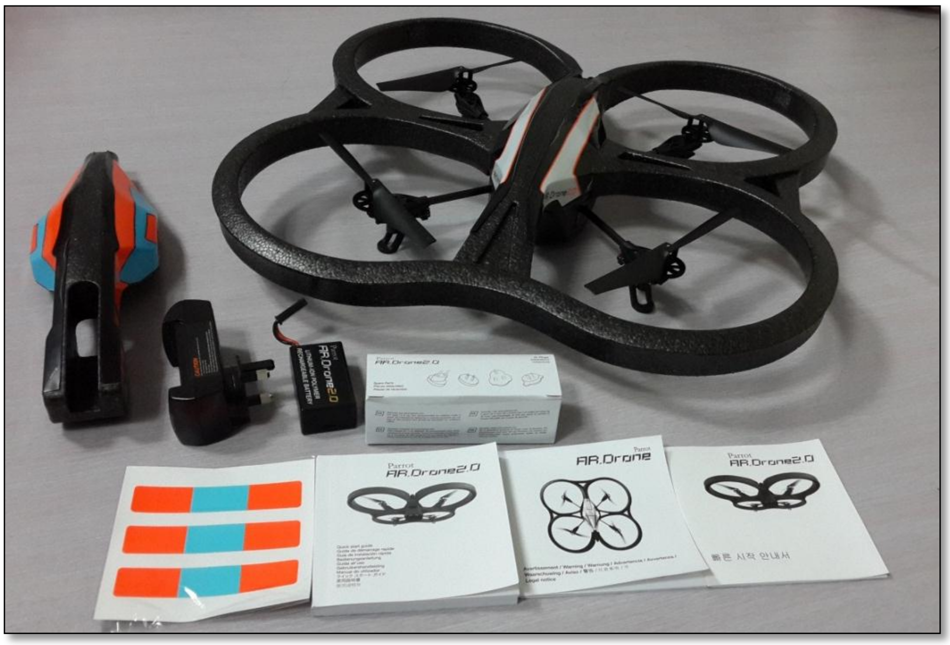


FIGURE 3.6 Full Set of Parrot AR.Drone 2.0

- **Extra Drone Batteries for Experimentation Purpose**

FIGURE 3.7 shows the batteries for the drone. The original set of drone comes with only one 1000mAh LiPo battery which can only last the drone flight of maximum 15 minutes duration. To aid the smooth flow of the experimentation, two extra batteries are purchased with 50% extra capacity (1500mAh) which can last the drone flight to about 25 minutes each.

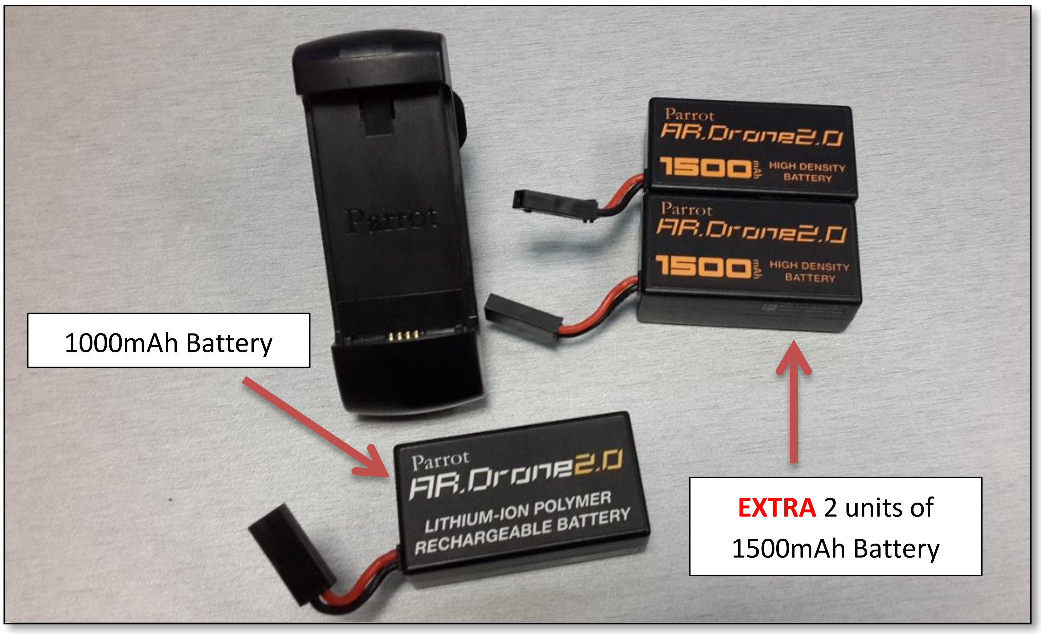

1000mAh Battery
EXTRA 2 units of 1500mAh Battery
Parrot AR.Drone2.0 LITHIUM-ION POLYMER RECHARGEABLE BATTERY
Parrot AR.Drone2.0 1500 mAh HIGH DENSITY BATTERY
Parrot AR.Drone2.0 1500 mAh HIGH DENSITY BATTERY
Parrot

FIGURE 3.7 Extra Drone Batteries for Experimentation Purpose

➢ **Fabricated Tags for Drone Movement**

The fabricated tags for the drone movement with the commands of left, right, up and down are shown in FIGURE 3.8.

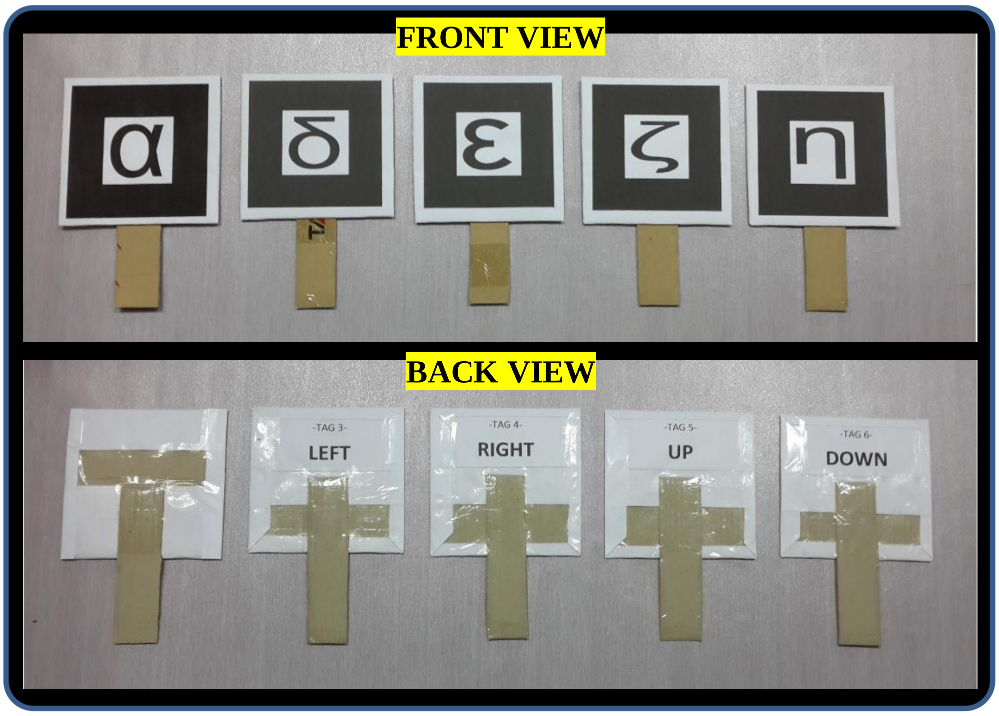


FIGURE 3.8 Fabricated Tags for Drone Movement

➢ **Experimentation Space**

For safety and to reduce crashes, a large space to conduct the experiment is necessary. A spacious area such as in FIGURE 3.9 is used for the experimentation.

FIGURE 3.9 Spacious Hall for Drone Experimentation

CHAPTER 4

RESULTS AND DISCUSSION

Throughout the entire project, there are four main packages used; three packages obtained from the Robot Operating System open source website with little modification and enhancement, one package is self-developed for the drone autonomous movement based on visual tag recognition. FIGURE 4.1 shows the packages used in this project.

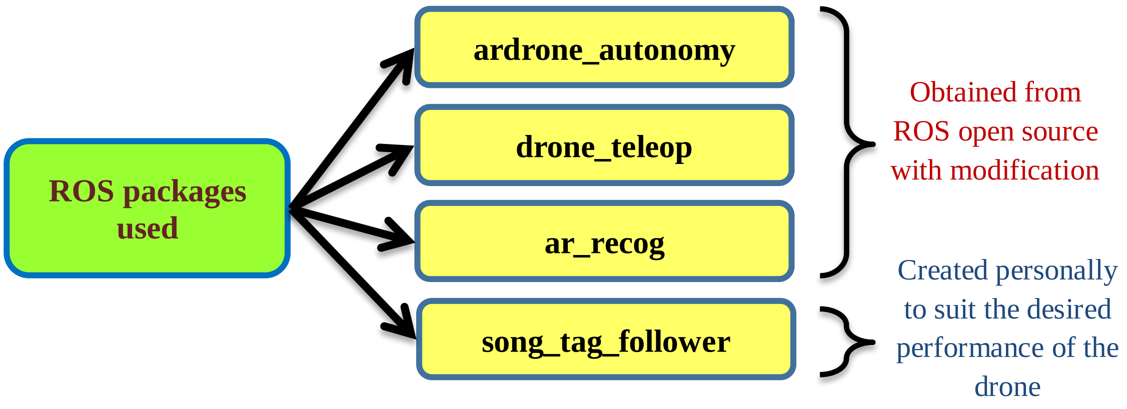

ROS packages used
ardrone_autonomy
drone_teleop
ar_recog
song_tag_follower
Obtained from ROS open source with modification
Created personally to suit the desired performance of the drone

FIGURE 4.1 ROS Packages Used

Before starting with the algorithm development, the workstation platform is prepared by installing Linux Ubuntu 12.04. Then, several versions of ROS have been installed (Fuerte, Groovy, Hydro) but due to the tag recognition package (ar_recog) compatibility problem, ROS-Fuerte is finalized as the platform for the project development. The software and ROS packages must be installed and compiled correctly in order for it to function without errors. Throughout the 28 weeks project development, many low-level programming errors are encountered and it is very time-consuming to solve the bug. The results and discussion of the project will be further divided into three main parts for easier understanding of the project outcome.

4.1 DRONE COMMUNICATION & KEYBOARD TELE-OPERATION

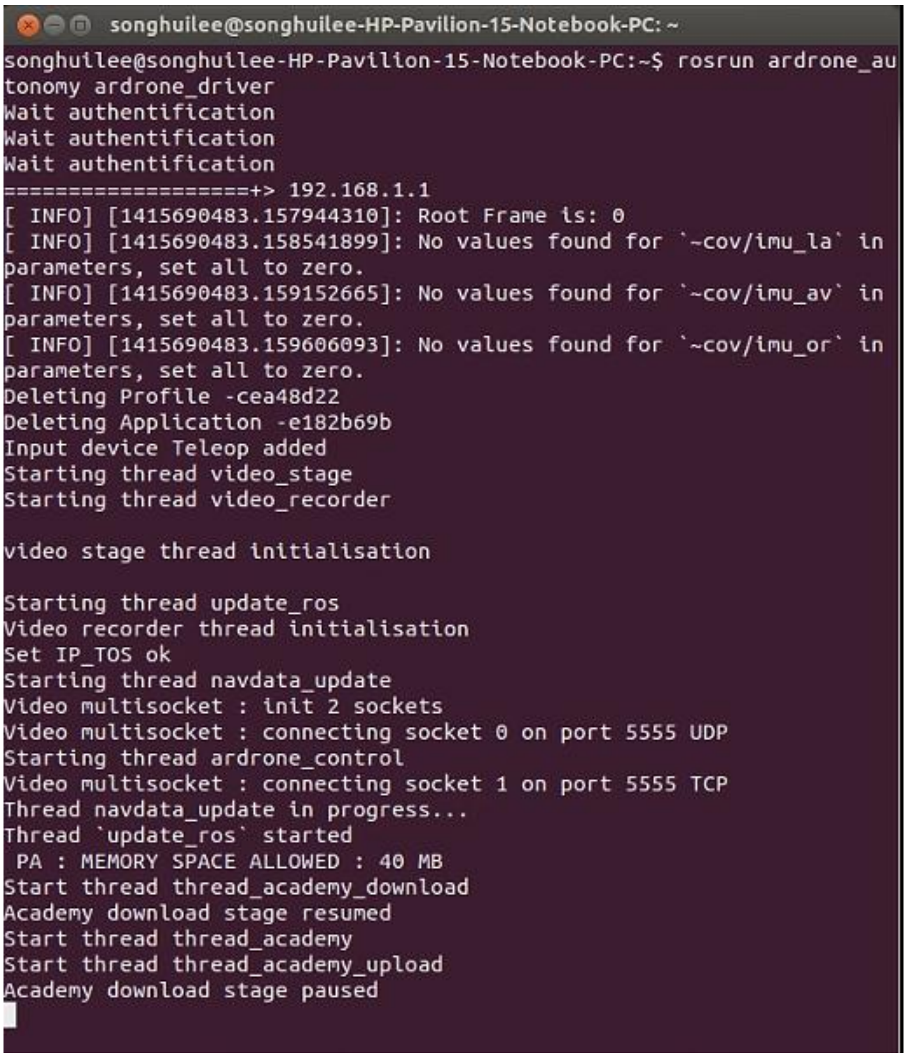
The first step for the drone communication with the laptop is to have the drone driver successfully compiled in ROS. It means the communication between the drone and laptop is established. The sensors data can be retrieved. Also, different nodes and topics are readily accessible.

a) To start up the system, run *roscore*.

```
$ roscore
```

b) To establish communication with drone from laptop, run the following:

```
$ rosrun ardrone_autonomy ardrone_driver
```

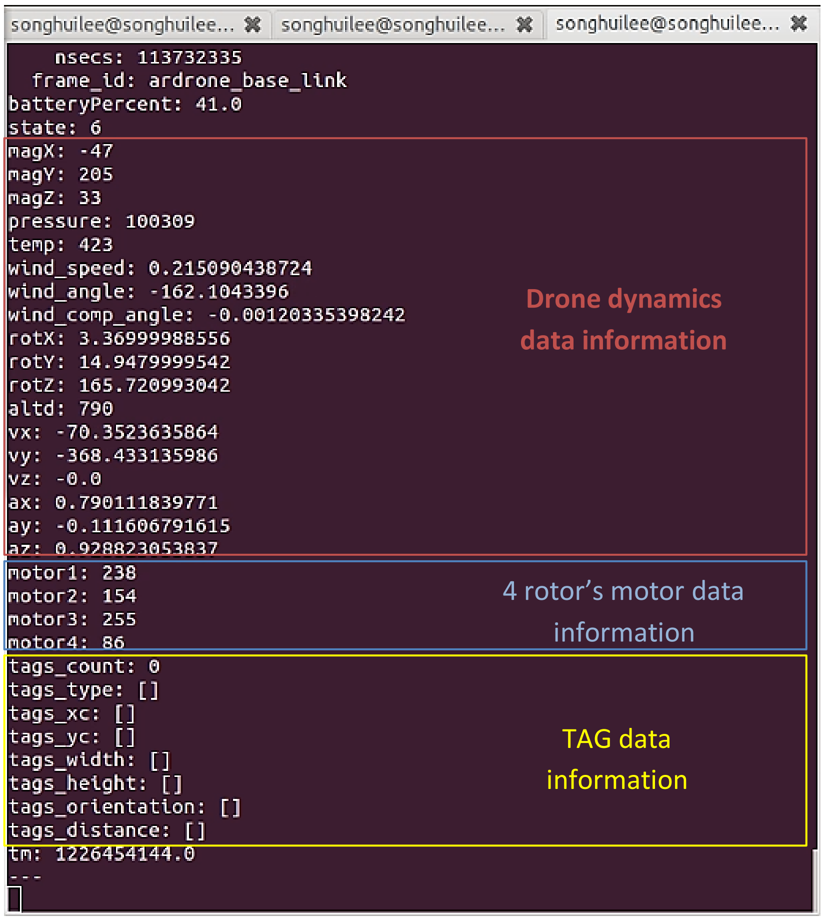


```
songhuilee@songhuilee-HP-Pavilion-15-Notebook-PC: ~
songhuilee@songhuilee-HP-Pavilion-15-Notebook-PC:~$ rosrun ardrone_au
tonomy ardrone_driver
Wait authentification
Wait authentification
Wait authentification
===================+> 192.168.1.1
[ INFO] [1415690483.157944310]: Root Frame is: 0
[ INFO] [1415690483.158541899]: No values found for `~cov/imu_la` in
parameters, set all to zero.
[ INFO] [1415690483.159152665]: No values found for `~cov/imu_av` in
parameters, set all to zero.
[ INFO] [1415690483.159606093]: No values found for `~cov/imu_or` in
parameters, set all to zero.
Deleting Profile -cea48d22
Deleting Application -e182b69b
Input device Teleop added
Starting thread video_stage
Starting thread video_recorder

video stage thread initialisation

Starting thread update_ros
Video recorder thread initialisation
Set IP_TOS ok
Starting thread navdata_update
Video multisocket : init 2 sockets
Video multisocket : connecting socket 0 on port 5555 UDP
Starting thread ardrone_control
Video multisocket : connecting socket 1 on port 5555 TCP
Thread navdata_update in progress...
Thread `update_ros` started
 PA : MEMORY SPACE ALLOWED : 40 MB
Start thread thread_academy_download
Academy download stage resumed
Start thread thread_academy
Start thread thread_academy_upload
Academy download stage paused
```

FIGURE 4.2 Running Drone Driver (ardrone_driver)

c) To extract navdata from the drone sensors, run the following:

```
$ rostopic echo ardrone/navdata
```



```
songhuilee@songhuilee...  songhuilee@songhuilee...  songhuilee@songhuilee...
    nsecs: 113732335
  frame_id: ardrone_base_link
batteryPercent: 41.0
state: 6
magX: -47
magY: 205
magZ: 33
pressure: 100309
temp: 423
wind_speed: 0.215090438724
wind_angle: -162.1043396
wind_comp_angle: -0.00120335398242
rotX: 3.36999988556
rotY: 14.9479999542
rotZ: 165.720993042
altd: 790
vx: -70.3523635864
vy: -368.433135986
vz: -0.0
ax: 0.790111839771
ay: -0.111606791615
az: 0.928823053837
motor1: 238
motor2: 154
motor3: 255
motor4: 86
tags_count: 0
tags_type: []
tags_xc: []
tags_yc: []
tags_width: []
tags_height: []
tags_orientation: []
tags_distance: []
tm: 1226454144.0
---
```

Drone dynamics data information
4 rotor's motor data information
TAG data information

FIGURE 4.3 Information Displayed in Navdata Topic

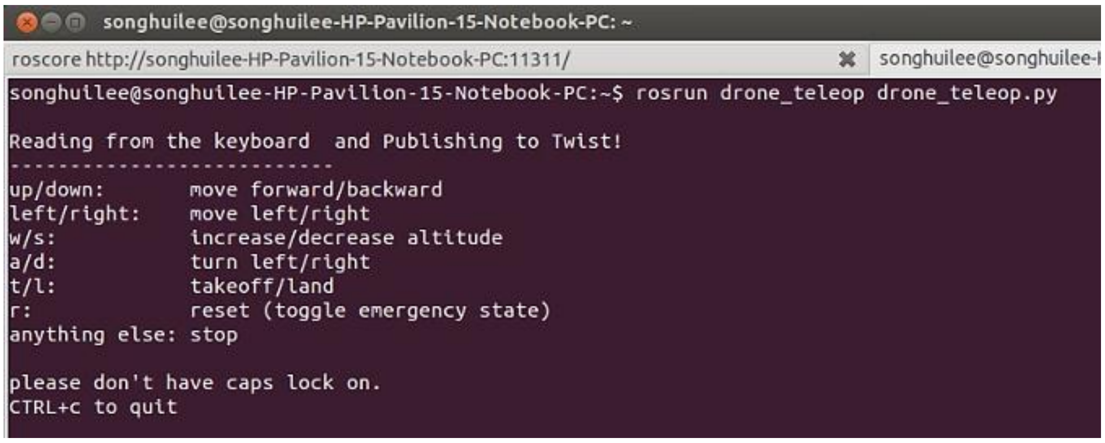
The navdata topic displays sensors information about the dynamics of the drone. The 3-axis gyroscope sensor gives information on the rotation of the drone in x, y, z-axis (rotX, rotY, rotZ). The drone rotation data given in degrees is crucial for speed control as well as flight stabilization.

The altitude is measured (in meter) by the ultrasonic sensor when the drone is less than 6 meters from the ground. Higher than that, the ultrasonic sensor is no longer providing accurate readings. Hence, the pressure sensor now will be used to infer the altitude of the drone based on the phenomenon that the higher the altitude, the lower the pressure.

Velocity is estimated by the downward optical flow sensor together with the altitude information for ground speed estimation using triangulation approach. Acceleration on the other hand, is measured by the accelerometer sensor.

d) To navigate the drone manually using keyboard tele-operation, run the following:

```
$ rosrun drone_teleop drone_teleop.py
```



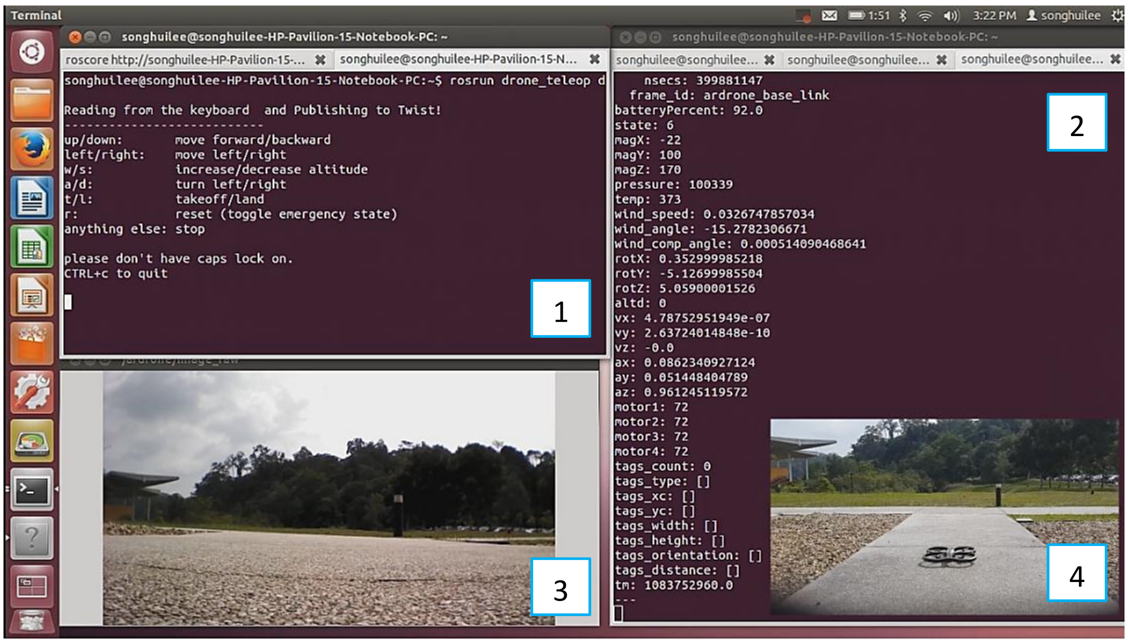
songhuilee@songhuilee-HP-Pavilion-15-Notebook-PC: ~
roscore http://songhuilee-HP-Pavilion-15-Notebook-PC:11311/
songhuilee@songhuilee-

```
songhuilee@songhuilee-HP-Pavilion-15-Notebook-PC:~$ rosrun drone_teleop drone_teleop.py

Reading from the keyboard  and Publishing to Twist!
---------------------------
up/down:       move forward/backward
left/right:    move left/right
w/s:           increase/decrease altitude
a/d:           turn left/right
t/l:           takeoff/land
r:             reset (toggle emergency state)
anything else: stop

please don't have caps lock on.
CTRL+c to quit
```

FIGURE 4.4 Keyboard Tele-Operation Key Function Description for Drone Navigation Control

In the python script *drone_teleop.py*, specific key is assigned to execute particular movement command as displayed in the terminal window of the laptop. This is much more convenient than to type in long navigation commands one by one manually because in the case of controlling the drone flight, it is best if the user can give quick command for the drone flight movement. For example, in the case where the drone is about to crash onto the wall, it will be faster to press a key to perform emergency stop rather than to type in the command in the terminal window. The delay in sending the navigation command might put the drone in high risk of getting into accident. Also, the advantage of keyboard tele-operation over manual type-in command is that the command will be stopped once the key pressed is released as compared to the manual type-in command where the entered command will always be executed until another command instructs it to stop.

Upon the success of communication establishment between the drone and laptop together with keyboard tele-operation for drone navigation control, the drone flight is tested both indoors and outdoors to evaluate how accurate is the flight performance prior to the manual navigation command in relative with the external environment factor (e.g. strong wind in outdoor environment).



```
songhuilee@songhuilee-HP-Pavilion-15-Notebook-PC:~$ rosrun drone_teleop d

Reading from the keyboard  and Publishing to Twist!
---------------------------
up/down:       move forward/backward
left/right:    move left/right
w/s:           increase/decrease altitude
a/d:           turn left/right
t/l:           takeoff/land
r:             reset (toggle emergency state)
anything else: stop

please don't have caps lock on.
CTRL+c to quit
```

```
    nsecs: 399881147
  frame_id: ardrone_base_link
batteryPercent: 92.0
state: 6
magX: -22
magY: 100
magZ: 170
pressure: 100339
temp: 373
wind_speed: 0.0326747857034
wind_angle: -15.2782306671
wind_comp_angle: 0.000514090468641
rotX: 0.352999985218
rotY: -5.12699985504
rotZ: 5.05900001526
altd: 0
vx: 4.78752951949e-07
vy: 2.63724014848e-10
vz: -0.0
ax: 0.0862340927124
ay: 0.051448404789
az: 0.961245119572
motor1: 72
motor2: 72
motor3: 72
motor4: 72
tags_count: 0
tags_type: []
tags_xc: []
tags_yc: []
tags_width: []
tags_height: []
tags_orientation: []
tags_distance: []
tm: 1083752960.0
---
```

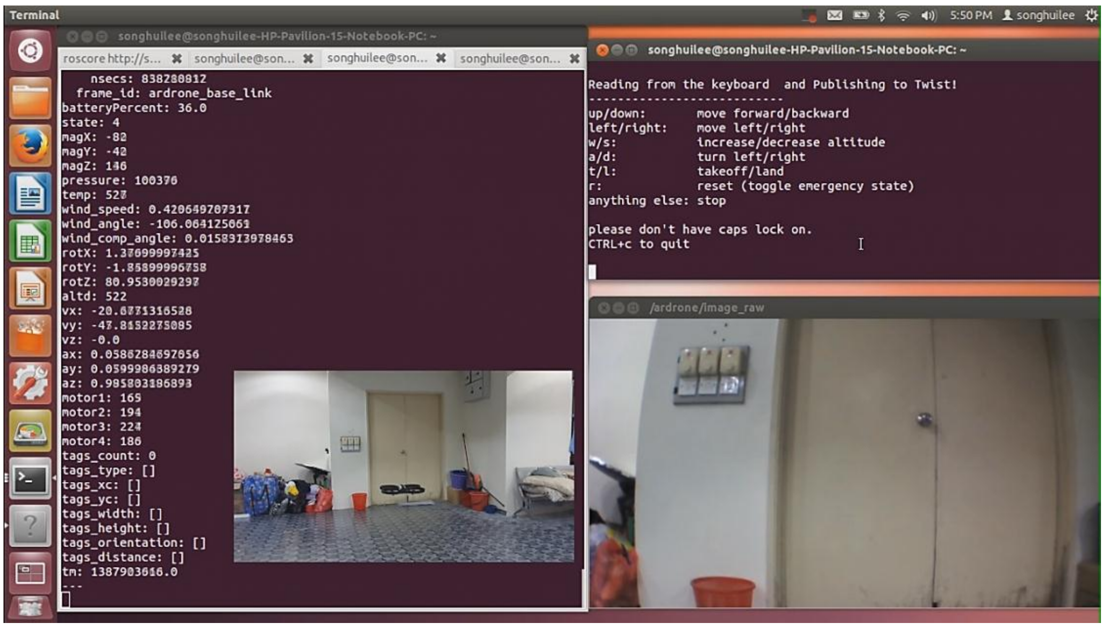
FIGURE 4.5 Drone Flight Test (Outdoor)

FIGURE 4.5 shows drone flight experimentation laptop print screen in the outdoor environment with the descriptions of the labeled boxes as below:

1. Terminal for the keyboard tele-operation for the drone navigation control.
2. Terminal displaying Navdata topic with the drone dynamics information.
3. Drone HD front camera view launched from the terminal, by running:

```
$ rosrun image_view image _view image:=/ardrone/image_raw
```

4. External camera capturing the drone flight, separated from ROS platform.



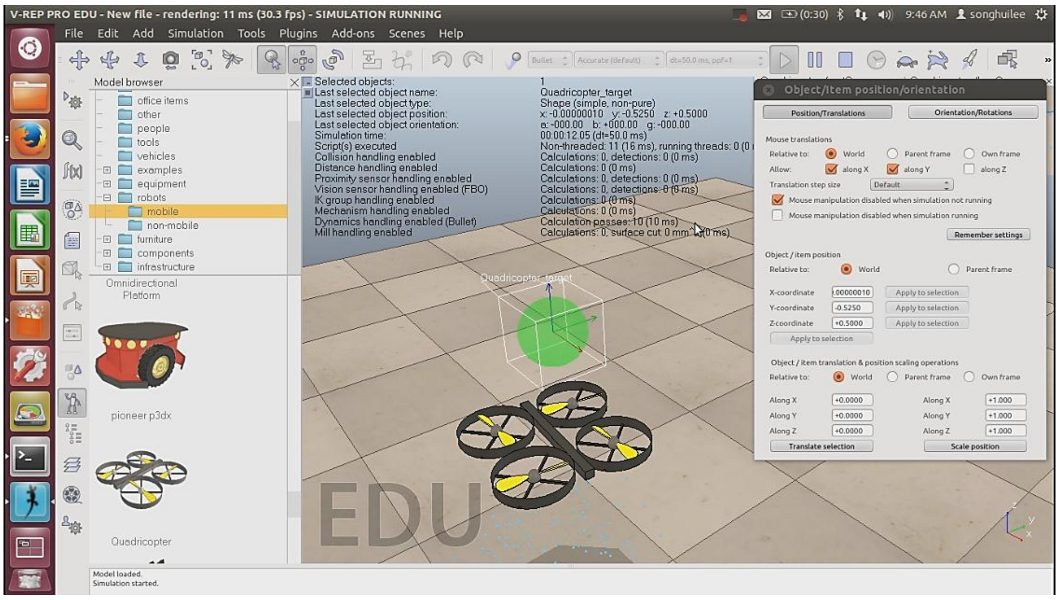
```
    nsecs: 838280812
  frame_id: ardrone_base_link
batteryPercent: 36.0
state: 4
magX: -82
magY: -42
magZ: 146
pressure: 100376
temp: 527
wind_speed: 0.420649707317
wind_angle: -106.064125061
wind_comp_angle: 0.0158313978463
rotX: 1.37699997425
rotY: -1.85899996758
rotZ: 80.9530029297
altd: 522
vx: -20.6771316528
vy: -47.8152275085
vz: -0.0
ax: 0.0586284697056
ay: 0.0599986389279
az: 0.985803186893
motor1: 165
motor2: 194
motor3: 224
motor4: 186
tags_count: 0
tags_type: []
tags_xc: []
tags_yc: []
tags_width: []
tags_height: []
tags_orientation: []
tags_distance: []
tm: 1387903616.0
---
```

```
Reading from the keyboard  and Publishing to Twist!
---------------------------
up/down:       move forward/backward
left/right:    move left/right
w/s:           increase/decrease altitude
a/d:           turn left/right
t/l:           takeoff/land
r:             reset (toggle emergency state)
anything else: stop

please don't have caps lock on.
CTRL+c to quit
```

FIGURE 4.6 Drone Flight Test (Indoor)

Similarly, FIGURE 4.6 shows the drone flight experimentation laptop print screen in indoor environment. The drone flight is very unstable in the outdoor experimentation due to the strong wind condition. For example, when the drone is expected to move forward via the keyboard tele-operation command, it tends to move sideways in the direction of the wind. This is because the navigation command to the drone is an open-loop system and there is no corrective feedback to deal with the external factor influence such as wind. In order to counter this problem, a complex close-loop system shall be designed taking into consideration the flight stabilization with six degree of freedom. This approach is possible but it will take up too much time, hence the drone will be experimented indoor for the time being to prioritize the goal for tag recognition program development. The same drone flight experimentation has been carried out indoor and it is proven that the flight is very much stable compared to the outdoor environment.

Apart from the real drone experimentation, the basic Virtual Robot Experimentation Platform (V-REP) simulator tool is explored. This simulator tool is chosen because it is very powerful in displaying the simulation almost similar to the real environment depending on the setting made. Since this V-REP simulator tool is released recently (year 2014), there is insufficient reference material specifically for the quadcopter model. For instance, the V-REP installation in ROS platform is successfully performed after five trials due to the inadequate installation steps from the available online resources. Therefore, given such condition, the progress of the V-REP tool will mostly depend on self-exploration.



```
Selected objects:                          1
Last selected object name:                 Quadricopter_target
Last selected object type:                 Shape (simple, non-pure)
Last selected object position:             x: -0.00000010   y: -0.5250   z: +0.5000
Last selected object orientation:          a: -000.00   b: +000.00   g: -000.00
Simulation time:                           00:00:12.05 (dt=50.0 ms)
Script(s) executed                         Non-threaded: 11 (16 ms), running threads: 0 (0
Collision handling enabled                 Calculations: 0, detections: 0 (0 ms)
Distance handling enabled                  Calculations: 0 (0 ms)
Proximity sensor handling enabled          Calculations: 0, detections: 0 (0 ms)
Vision sensor handling enabled (FBO)       Calculations: 0, detections: 0 (0 ms)
IK group handling enabled                  Calculations: 0 (0 ms)
Mechanism handling enabled                 Calculations: 0 (0 ms)
Dynamics handling enabled (Bullet)         Calculation passes: 10 (10 ms)
Mill handling enabled                      Calculations: 0, surface cut 0 mm^3 (0 ms)
```

Mouse translations
Relative to: [✓ World] Parent frame Own frame
Allow: [✓ along X] [✓ along Y] along Z
Translation step size: Default
[✓ Mouse manipulation disabled when simulation not running]
Mouse manipulation disabled when simulation running

Object / item position
Relative to: [✓ World] Parent frame
X-coordinate: .00000010
Y-coordinate: -0.5250
Z-coordinate: +0.5000

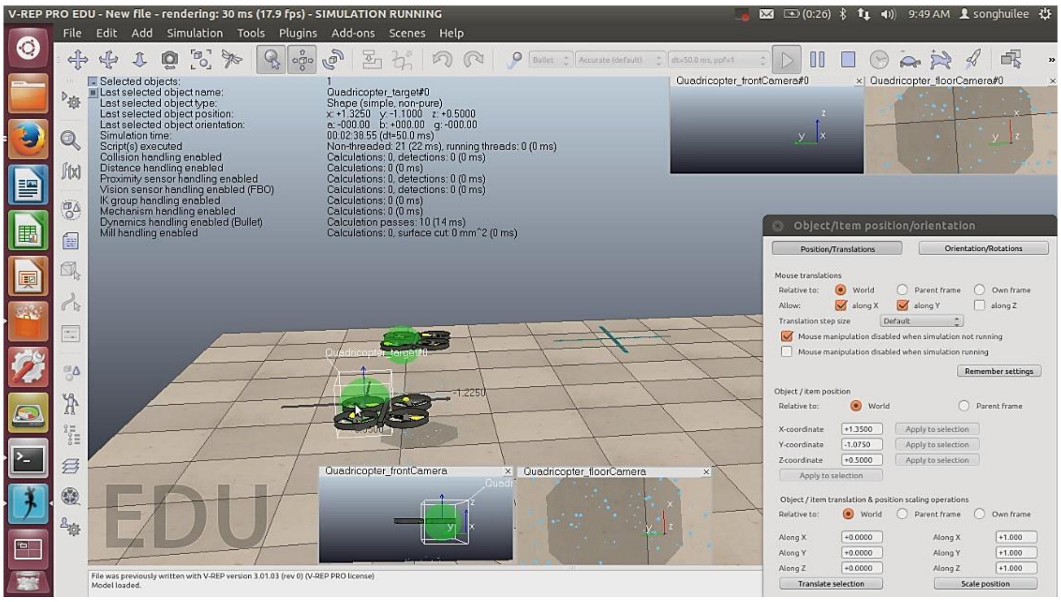
Object / item translation & position scaling operations
Relative to: [✓ World] Parent frame Own frame

Along X	+0.0000	Along X	+1.000
Along Y	+0.0000	Along Y	+1.000
Along Z	+0.0000	Along Z	+1.000

```
Model loaded.
Simulation started.
```

FIGURE 4.7 Single Quadcopter in V-REP Simulator Environment

As for now, the basic icon tool in controlling the quadcopter in the V-REP environment is mastered. More than one quadcopter model can be placed in the V-REP environment. Also, it can display the quadcopter's front and down camera views as shown in FIGURE 4.8. The challenge in using this V-REP tool is the integration of the real drone program code into the simulation environment since the programming language code might not be compatible and hence, more exploration will be carried out in this direction.


V-REP PRO EDU - New file - rendering: 30 ms (17.9 fps) - SIMULATION RUNNING
File
Edit
Add
Simulation
Tools
Plugins
Add-ons
Scenes
Help
Quadricopter_frontCamera#0
Quadricopter_floorCamera#0
Quadricopter_frontCamera
Quadricopter_floorCamera
Quadricopter_target#0
EDU

```
Selected objects:                       1
Last selected object name:              Quadricopter_target#0
Last selected object type:              Shape (simple, non-pure)
Last selected object position:          x: +1.3250   y: -1.1000   z: +0.5000
Last selected object orientation:       a: -000.00   b: +000.00   g: -000.00
Simulation time:                        00:02:38.55 (dt=50.0 ms)
Script(s) executed                      Non-threaded: 21 (22 ms), running threads: 0 (0 ms)
Collision handling enabled              Calculations: 0, detections: 0 (0 ms)
Distance handling enabled               Calculations: 0 (0 ms)
Proximity sensor handling enabled       Calculations: 0, detections: 0 (0 ms)
Vision sensor handling enabled (FBO)    Calculations: 0, detections: 0 (0 ms)
IK group handling enabled               Calculations: 0 (0 ms)
Mechanism handling enabled              Calculations: 0 (0 ms)
Dynamics handling enabled (Bullet)      Calculation passes: 10 (14 ms)
Mill handling enabled                   Calculations: 0, surface cut: 0 mm^2 (0 ms)
```

Object/item position/orientation

Mouse translations
Relative to: [✓ World] Parent frame Own frame
Allow: [✓ along X] [✓ along Y] along Z
Translation step size: Default
[✓ Mouse manipulation disabled when simulation not running]
Mouse manipulation disabled when simulation running

Object / item position
Relative to: [✓ World] Parent frame
X-coordinate: +1.3500
Y-coordinate: -1.0750
Z-coordinate: +0.5000

Object / item translation & position scaling operations
Relative to: [✓ World] Parent frame Own frame

Along X	+0.0000	Along X	+1.000
Along Y	+0.0000	Along Y	+1.000
Along Z	+0.0000	Along Z	+1.000

```
File was previously written with V-REP version 3.01.03 (rev 0) (V-REP PRO license)
Model loaded.
```

FIGURE 4.8 Two Quadcopters in V-REP with Two Camera Views

Although V-REP tool is very good and advance, the significance can only be seen when used in higher level of application such as object recognition or 3D-mapping reconstruction. Hence, the tag recognition drone movement in this project will act as the fundamental step to be advanced by next user to make better applications which then can be integrated with V-REP tool. The documentation of the V-REP tool exploration will be very handy to the next user.

In addition, to smoothen the real drone experimentation and prevent drone battery short flight time to limit the exploration progress, two additional drone batteries of 1500mAh capacity each have been purchased.

4.2 TAG RECOGNITION PACKAGE

Drone tag recognition package (ar_recog) for ROS-Fuerte has been installed for multiple times but was not able to be compiled without errors despite following each and every step stated in the webpage where the open-source package is downloaded. After a couple of weeks of settling the 'low-level' problem, a definite installation and compilation of the package is summarized as below:

ar_recog Package Installation & Compilation Guide

1. Enter your password to get permission upon the command using "sudo"

```
$ sudo apt-get install python-rosinstall
```

2. Initialize and source your ros_workspace

```
$ rosws init ~/fuerte_workspace /opt/ros/fuerte
$ source ~/fuerte_workspace/setup.bash
```

3. Set your workspace folder location to place the package

```
$ rosws set ~/fuerte_workspace/sandbox
```

4. Add a line after going into the .bashrc file and save

```
$ gedit .bashrc
```

Add these 2 lines at the bottom of the file document script:

```
source /opt/ros/fuerte/setup.bash
source ~/fuerte_workspace/setup.bash
```

5. Download ar_recog package directly from the website into the sandbox location.

```
$ svn http://devel.iri.upc.edu/pub/labrobotica/ros/iri-ros-pkg/stacks/kinton_robot/trunk
```

6. Locate the ar_recog package using "rospack find" and rosmake (compile) the package with no failures (no error).

```
$ rospack find ar_recog
$ rosmake ar_recog
```

After the successful compilation of the ar_recog package, a tag (id: 0) is used for the tag recognition program testing. The commands in an orderly manner will be described with the aid of the command window snapshot.

ar_recog Tag Recognition Testing On Drone Guide

1. *Run "roscore" and the drone driver on separate terminals*

#Terminal 1

```
$ roscore
```

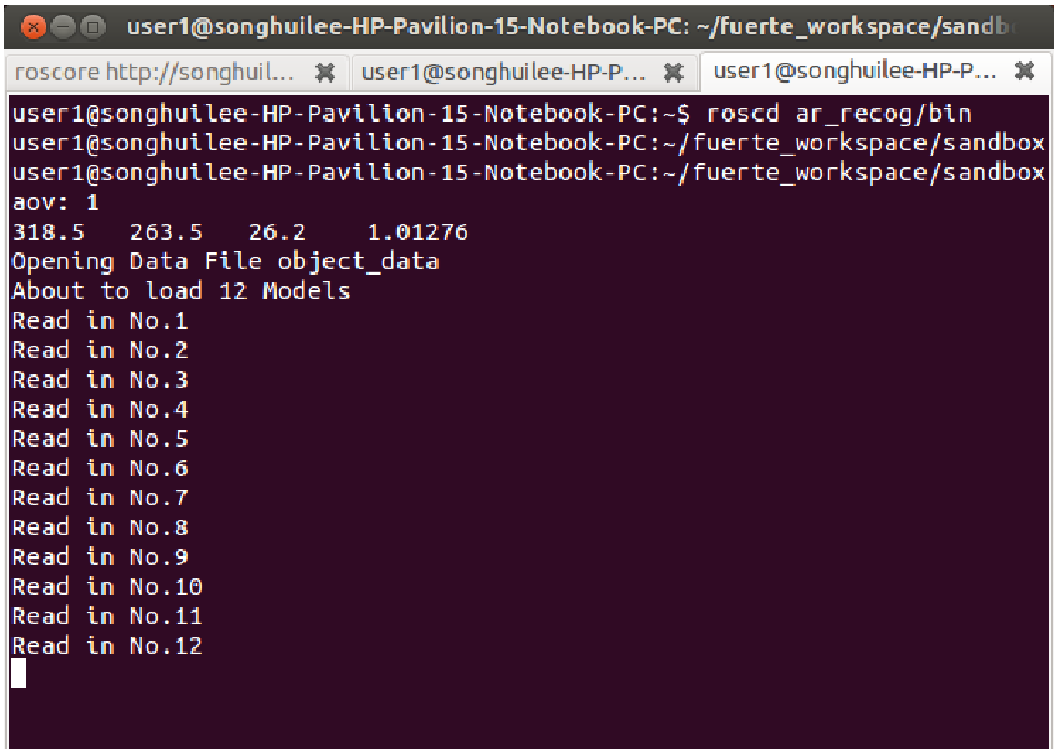
#Terminal 2

```
$ rosrun ardrone_autonomy ardrone_driver
```

2. *Go into the ar_recog/bin location to set the aov (angle of view) of the camera and initialize the tags identity (altogether 12 tags are registered and ready to be recognized)*

#Terminal 3

```
$ roscd ar_recog/bin
$ rosparam set aov 1.0
$ rosrun ar_recog ar_recog image:=/ardrone/image_raw
```



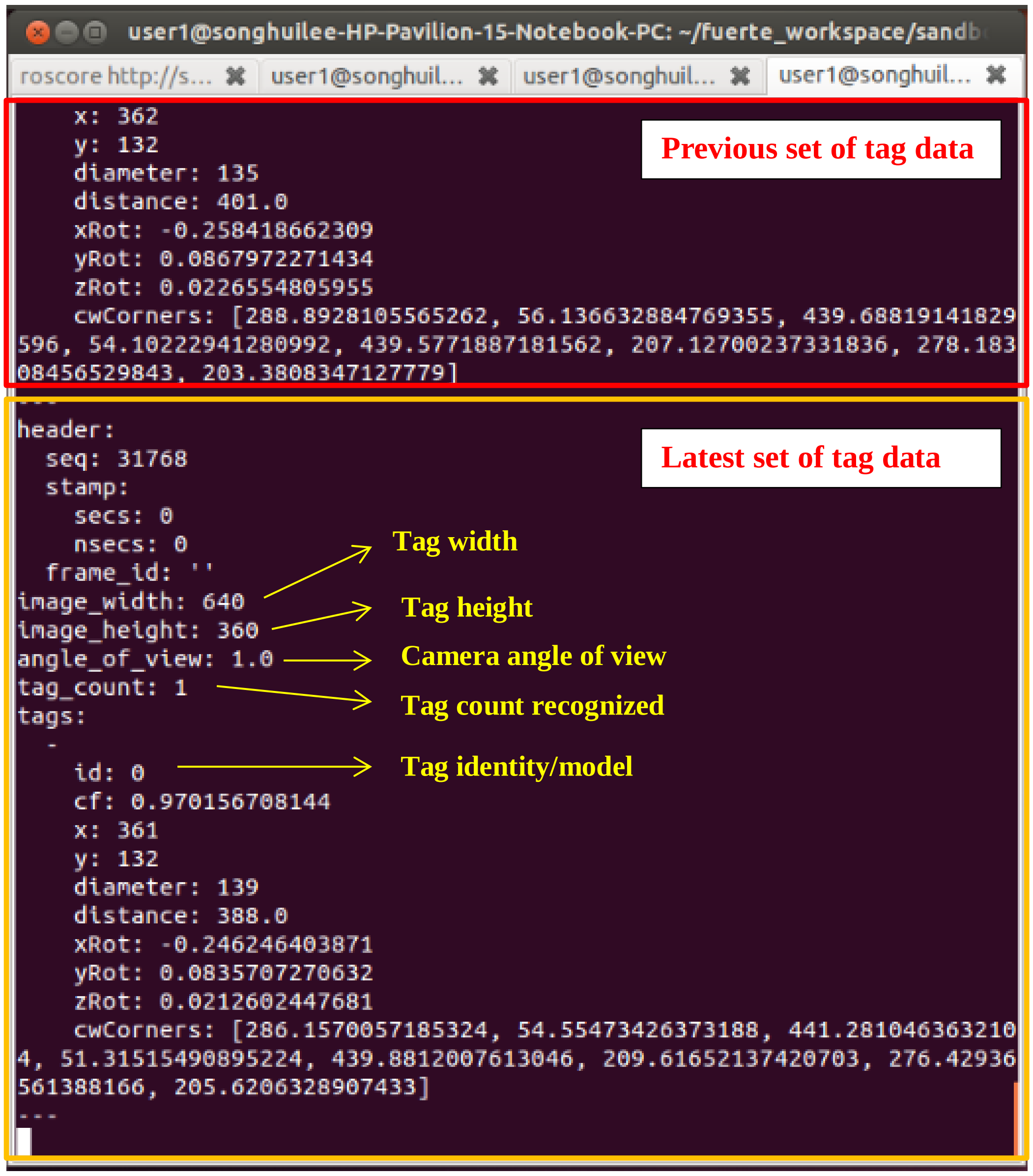
```
user1@songhuilee-HP-Pavilion-15-Notebook-PC: ~/fuerte_workspace/sandb
roscore http://songhuil...   user1@songhuilee-HP-P...   user1@songhuilee-HP-P...
user1@songhuilee-HP-Pavilion-15-Notebook-PC:~$ roscd ar_recog/bin
user1@songhuilee-HP-Pavilion-15-Notebook-PC:~/fuerte_workspace/sandbox
user1@songhuilee-HP-Pavilion-15-Notebook-PC:~/fuerte_workspace/sandbox
aov: 1
318.5    263.5    26.2     1.01276
Opening Data File object_data
About to load 12 Models
Read in No.1
Read in No.2
Read in No.3
Read in No.4
Read in No.5
Read in No.6
Read in No.7
Read in No.8
Read in No.9
Read in No.10
Read in No.11
Read in No.12
```

FIGURE 4.9 Activated 12 Models of Tag in ar_recog Package

3. *Go into the ar_recog/bin location and display the tag info identified.*

#Terminal 4

```
$ roscd ar_recog/bin
$ rostopic echo /tags
```



```
    x: 362
    y: 132
    diameter: 135
    distance: 401.0
    xRot: -0.258418662309
    yRot: 0.0867972271434
    zRot: 0.0226554805955
    cwCorners: [288.8928105565262, 56.136632884769355, 439.68819141829
596, 54.10222941280992, 439.5771887181562, 207.12700237331836, 278.183
08456529843, 203.3808347127779]
---
header:
  seq: 31768
  stamp:
    secs: 0
    nsecs: 0
  frame_id: ''
image_width: 640
image_height: 360
angle_of_view: 1.0
tag_count: 1
tags:
  -
    id: 0
    cf: 0.970156708144
    x: 361
    y: 132
    diameter: 139
    distance: 388.0
    xRot: -0.246246403871
    yRot: 0.0835707270632
    zRot: 0.0212602447681
    cwCorners: [286.1570057185324, 54.55473426373188, 441.281046363210
4, 51.31515490895224, 439.8812007613046, 209.61652137420703, 276.42936
561388166, 205.6206328907433]
---
```

FIGURE 4.10 Tag Information From echo /tags

4. Launch the camera front view and tags will be identified with a green line box.

#Terminal 5

```
$ rosrun image_view image_view image:=/ar/image
```

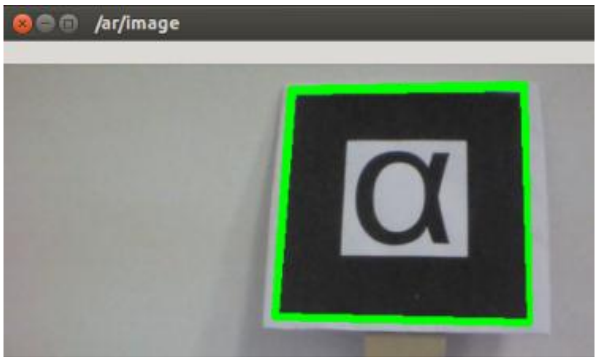
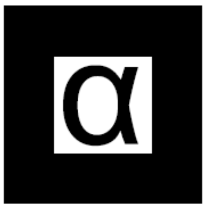
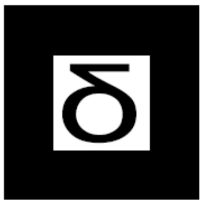
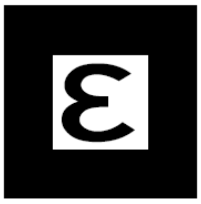
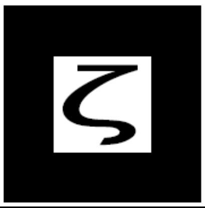
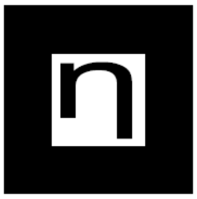


FIGURE 4.11 Recognized Tag from Drone's Front Camera View

TABLE 4.1 summarizes the tag id with their corresponding movement command to the drone when the tag is recognized. These tags with the following symbols are generated using Unicode characters from the DejaVu Sans font. Hence, they are chosen to be used as the tags since the characters are unique which will not easily detected in the daily life environment.

TABLE 4.1 Tags and the Movement Command Assigned

TAGS	**MOVEMENT COMMAND**
id = 0 (Other than id = 3,4,5,6) or no tag 	*Hover at the same spot (do nothing)*
id = 3 	*Move to the LEFT*
id = 4 	*Move to the RIGHT*
id = 5 	*Move UP (increase altitude)*
id = 6 	*Move DOWN (reduce altitude)*

Also from the tag recognition program, it is concluded from numerous experimentations that the orientation of the tag facing the camera is very important. The tag will not be recognized if the angle is too deviated or not fully shown within the angle of view as in FIGURE 4.12. Hence, it is best that the tag is exposed in a good lightning room condition, no sunlight exposure and the orientation does not exceed 30 degrees from the drone front camera for the best performance.

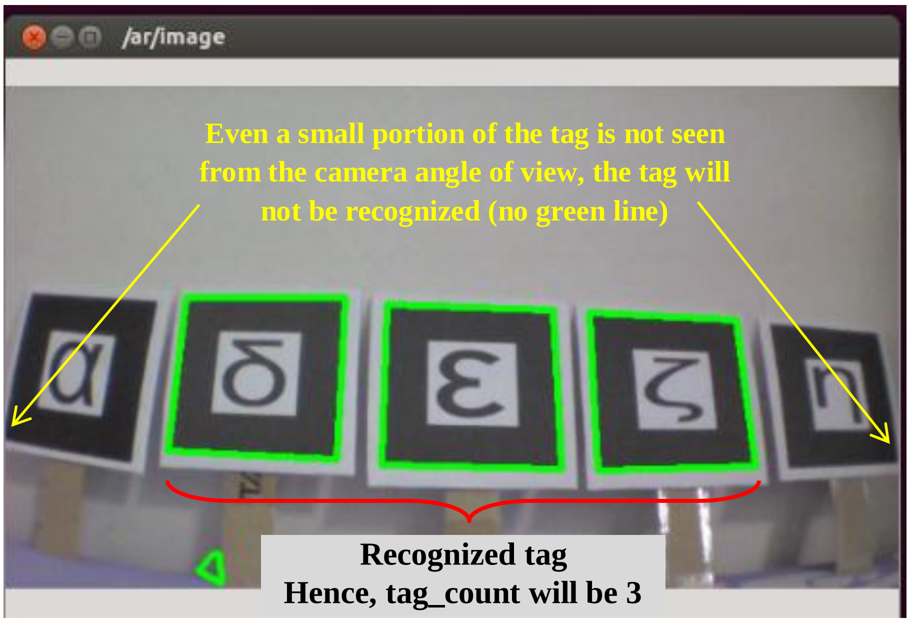

/ar/image
Even a small portion of the tag is not seen from the camera angle of view, the tag will not be recognized (no green line)
Recognized tag
Hence, tag_count will be 3

FIGURE 4.12 Condition for Recognized Tag from Drone's Front Camera View

4.3 VISUAL-BASED DRONE MOVEMENT

Since the tag recognition testing is a success, several tags are prepared as the visual control for the drone movement. To be more precise, each tag with their own defined identity or id number will instruct the drone to move in different direction when the tag itself is detected and within the angle of view of the drone front camera. The package song_tag_follower is created from scratch with other necessary package dependencies; especially the ar_recog package to obtain the tags information.

Creating & Building Package song_tag_follower Guide

1. *Go into the the ~/fuerte_workspace/sandbox directory*

```
$ cd ~/fuerte_workspace/sandbox
```

2. *Create the package with the dependencies list*

```
$ roscreate-pkg song_tag_follower rospy roscpp roar_recog geometry_msgs
ardrone_autonomy std_msgs
```

3. To ensure that ROS can find the new created package, do the following:

```
$ rospack profile
$ rospack find  song_tag_follower
```

4. Try moving to the directory for the package and take a look at the manifest file

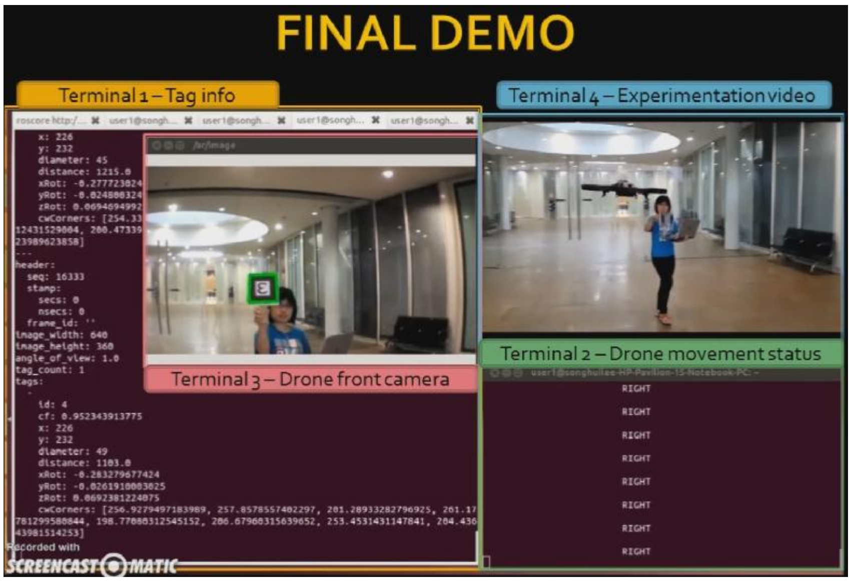
```
$ roscd song_tag_follower
$ cat manifest.xml
```

5. To build the package created

```
$ rosmake song_tag_follower
```

The algorithm of the tag recognition for drone movement is written in the script named tag_movement.py in the song_tag_follower package folder. The parameters for the drone flight are modified for the best flight condition in either indoor or outdoor environment. The script must be changed into an executable script; otherwise there will be no response from the drone.

The whole project is then finalized with the tag_movement.py script and the video of the whole experimentation is uploaded in YouTube which made it easier to access by public as well as the future user that will continue this project. FIGURE 4.13 shows the screenshot of the video. The video is part of the project documentation which explains the whole project key milestone and final demo of the drone flight based on visual tag recognition. The link of the video is: https://youtu.be/yCHDoaer2qk

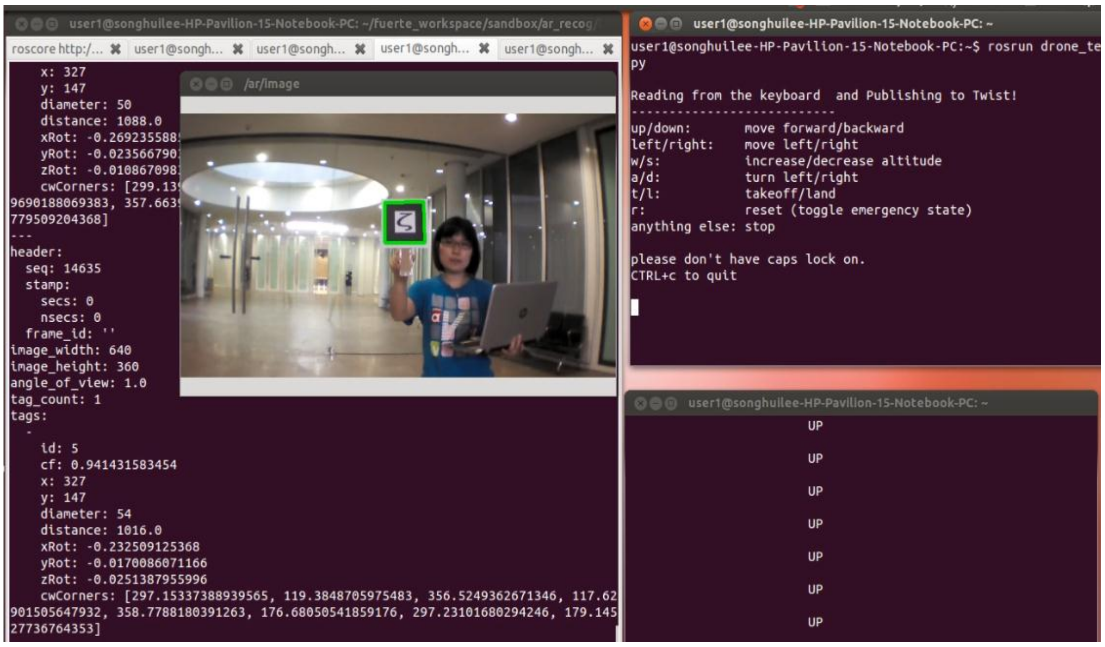

FINAL DEMO
Terminal 1 – Tag info
Terminal 4 – Experimentation video
Terminal 3 – Drone front camera
Terminal 2 – Drone movement status
roscore http:/...
user1@songh...
Recorded with SCREENCAST-O-MATIC

```
    x: 226
    y: 232
    diameter: 45
    distance: 1215.0
    xRot: -0.277723824
    yRot: -0.024880324
    zRot: 0.0694694992
    cwCorners: [254.33
12431529004, 200.47339
23989623858]
---
header:
  seq: 16333
  stamp:
    secs: 0
    nsecs: 0
  frame_id: ''
image_width: 640
image_height: 360
angle_of_view: 1.0
tag_count: 1
tags:
  -
    id: 4
    cf: 0.952343913775
    x: 226
    y: 232
    diameter: 49
    distance: 1103.0
    xRot: -0.283279677424
    yRot: -0.0261910003825
    zRot: 0.0692381224075
    cwCorners: [256.9279497183989, 257.8578557402297, 201.28933282796925, 261.17
781299588844, 198.77088312545152, 206.67968315639652, 253.4531431147841, 204.436
43981514253]
```

```
RIGHT
RIGHT
RIGHT
RIGHT
RIGHT
RIGHT
RIGHT
RIGHT
```

FIGURE 4.13   Final Demo Video Uploaded In YouTube

FIGURE 4.14 shows the terminal screen of the workstation while FIGURE 4.15 shows the experimentation of the drone.


user1@songhuilee-HP-Pavilion-15-Notebook-PC: ~/fuerte_workspace/sandbox/ar_recog
roscore http:/...
user1@songh...
user1@songh...
user1@songh...
user1@songh...
/ar/image
user1@songhuilee-HP-Pavilion-15-Notebook-PC: ~
user1@songhuilee-HP-Pavilion-15-Notebook-PC: ~

```
    x: 327
    y: 147
    diameter: 50
    distance: 1088.0
    xRot: -0.269235588
    yRot: -0.023566790
    zRot: -0.010867098
    cwCorners: [299.13
9690188069383, 357.663
779509204368]
---
header:
  seq: 14635
  stamp:
    secs: 0
    nsecs: 0
  frame_id: ''
image_width: 640
image_height: 360
angle_of_view: 1.0
tag_count: 1
tags:
  -
    id: 5
    cf: 0.941431583454
    x: 327
    y: 147
    diameter: 54
    distance: 1016.0
    xRot: -0.232509125368
    yRot: -0.0170086071166
    zRot: -0.0251387955996
    cwCorners: [297.15337388939565, 119.3848705975483, 356.5249362671346, 117.62
901505647932, 358.7788180391263, 176.68050541859176, 297.23101680294246, 179.145
27736764353]
```

```
user1@songhuilee-HP-Pavilion-15-Notebook-PC:~$ rosrun drone_te
py

Reading from the keyboard  and Publishing to Twist!
---------------------------
up/down:       move forward/backward
left/right:    move left/right
w/s:           increase/decrease altitude
a/d:           turn left/right
t/l:           takeoff/land
r:             reset (toggle emergency state)
anything else: stop

please don't have caps lock on.
CTRL+c to quit
```

```
                         UP

                         UP

                         UP

                         UP

                         UP

                         UP

                         UP
```

FIGURE 4.14   Drone Flight Workstation Terminal Screen

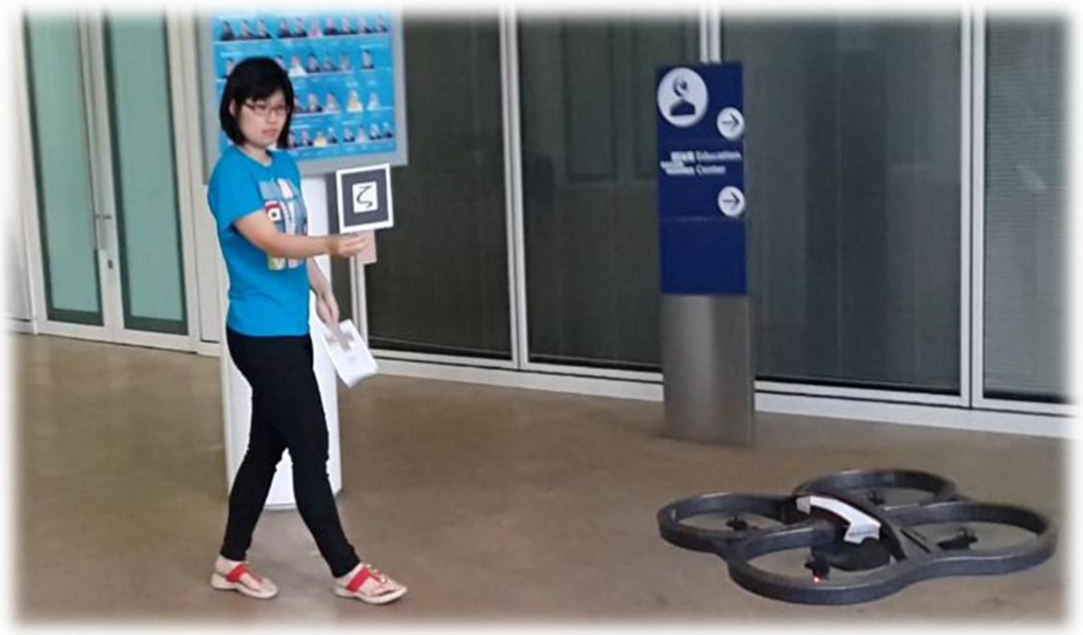


FIGURE 4.15   Drone Flight Experimentation with Tags

6. *To run the tag_movement.py script, run the following command in a new terminal*

```
$ rosrun song_tag_follower tag_movement.py
```

CHAPTER 5

CONCLUSION AND RECOMMENDATION

5.1 CONCLUSION

In conclusion, this project is highly feasible and the progress has been on track as stated in the Gantt chart. Several key milestones have been achieved such as preparing the ROS platform laptop, establishing the communication between the drone and laptop, extracting drone dynamics data as well as the front HD camera raw image view, manual navigation of drone using keyboard tele-operation, testing the drone flight in indoor and outdoor environment, installing and compiling the ar_recog package for tag recognition and lastly, the creation of song_tag_follower package to make the drone movement responds autonomously with the tag detected.

In the meantime, the knowledge and documentations are being transferred to several students which will be continuing this project especially for those who will be in the CISIR Drone Research group. The algorithm of the program is enhanced with comments and documentation of the project (program guide and video) is well prepared for future user reference. The outcome of this project can be further developed for the applications of localization, search and rescue mission, construction and transportation by a swarm of drone and etc.

Overall, the objectives of this tag recognition for drone movement program is successfully achieved using ROS.

5.2 RECOMMENDATION

Drone tag recognition is indeed a very interesting topic to be further enhanced in many areas for the purpose of search and rescue mission. However, I will only cover the research of the tag recognition and tracking on the drone due to time limitation. The part that I am trying to achieve is basically the initial research stage for the drone development for the search and rescue mission because in order for the drone to be officially used for the rescue mission in real-time disaster condition, many other criteria have to be taken into consideration which requires the researches to spend full time for the experimentation in the real time scenario. For the drone to be a practical rescue agent, it is recommended that the drone is equipped or modified using ROS framework to have the following abilities:

- Functional in extreme outdoor weather (rain, snow, strong wind) with stable flight condition
- Long battery life or self-rechargeable capability for long duration mission
- Differentiate detection of human (dead/living), objects (stationary/moving)

REFERENCES

[1] AR.Drone Academy. (2014). AR.Drone 2.0 Technical Specifications. Available: http://ardrone2.parrot.com/

[2] R. Shamroukh and F. Awad, "Detection of surviving humans in destructed environments using a simulated autonomous robot," in *Mechatronics and its Applications, 2009. ISMA '09. 6th International Symposium on*, 2009, pp. 1-6.

[3] S. Bhatia, H. S. Dhillon, and N. Kumar, "Alive human body detection system using an autonomous mobile rescue robot," in *India Conference (INDICON), 2011 Annual IEEE*, 2011, pp. 1-5.

[4] M. A. Ma'sum, M. K. Arrofi, G. Jati, F. Arifin, M. N. Kurniawan, P. Mursanto*, et al.*, "Simulation of intelligent Unmanned Aerial Vehicle (UAV) For military surveillance," in *Advanced Computer Science and Information Systems (ICACSIS), 2013 International Conference on*, 2013, pp. 161-166.

[5] M. S. Alvissalim, B. Zaman, Z. A. Hafizh, M. A. Ma'sum, G. Jati, W. Jatmiko*, et al.*, "Swarm quadrotor robots for telecommunication network coverage area expansion in disaster area," in *SICE Annual Conference (SICE), 2012 Proceedings of*, 2012, pp. 2256-2261.

[6] S. Schwertfeger, A. Birk, and H. Bulow, "Using iFMI spectral registration for video stabilization and motion detection by an Unmanned Aerial Vehicle (UAV)," in *Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on*, 2011, pp. 61-67.

[7] S. Gupte, P. I. T. Mohandas, and J. M. Conrad, "A survey of quadrotor Unmanned Aerial Vehicles," in *Southeastcon, 2012 Proceedings of IEEE*, 2012, pp. 1-6.

[8] Markus Achtelik, Abraham Bachrach, Ruijie He, Samuel Prentice and Nicholas Roy, “Autonomous Navigation and Exploration of a Quadrotor Helicopter in GPS-denied Indoor Environments”, http://iarc.angel-strike.com/2009SymposiumPapers/2009MIT.pdf

[9] T. Krajnik, M. Nitsche, S. Pedre, L. Preucil, and M. E. Mejail, "A simple visual navigation system for an UAV," in *Systems, Signals and Devices (SSD), 2012 9th International Multi-Conference on*, 2012, pp. 1-6.

[10] J. L. Sanchez-Lopez, J. Pestana, P. de la Puente, R. Suarez-Fernandez, and P. Campoy, "A system for the design and development of vision-based multi-robot quadrotor swarms," in *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*, 2014, pp. 640-648.

[11] J. Pestana, J. L. Sanchez-Lopez, P. de la Puente, A. Carrio, and P. Campoy, "A Vision-based Quadrotor Swarm for the participation in the 2013 International Micro Air Vehicle Competition," in *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*, 2014, pp. 617-622.

[12] F. Cocchioni, A. Mancini, and S. Longhi, "Autonomous navigation, landing and recharge of a quadrotor using artificial vision," in *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*, 2014, pp. 418-429.

[13] L. Jayatilleke and Z. Nian, "Landmark-based localization for Unmanned Aerial Vehicles," in *Systems Conference (SysCon), 2013 IEEE International*, 2013, pp. 448-451.

[14] D. I. Montufar, F. Munoz, E. S. Espinoza, O. Garcia, and S. Salazar, "Multi-UAV testbed for aerial manipulation applications," in *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*, 2014, pp. 830-835.

[15] M. A. Olivares-Mendez, S. Kannan, and H. Voos, "Setting up a testbed for UAV vision based control using V-REP &amp; ROS: A case study on aerial visual inspection," in *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*, 2014, pp. 447-458.

[16] J. Pestana, J. L. Sanchez-Lopez, P. Campoy, and S. Saripalli, "Vision based GPS-denied Object Tracking and following for unmanned aerial vehicles," in *Safety, Security, and Rescue Robotics (SSRR), 2013 IEEE International Symposium on*, 2013, pp. 1-6.

[17] Tripty Singh, Sanju S and Bichu Vijay. Article: A New Algorithm Designing for Detection of Moving Objects in Video. *International Journal of Computer Applications* 96(2):4-11, June 2014.

[18] J. Mendes and R. Ventura, "Safe teleoperation of a quadrotor using FastSLAM," in *Safety, Security, and Rescue Robotics (SSRR), 2012 IEEE International Symposium on*, 2012, pp. 1-6.

[19] C. Oh-hoon, B. Kyeong-Jin, and K. Eung-kon, "Stabilized UAV flight system design for structure safety inspection," in *Advanced Communication Technology (ICACT), 2014 16th International Conference on*, 2014, pp. 1312-1316.

[20] S. Zhou, Y. Wang, S. Ye, G. Zhu, and L. Cheng, "The Intelligent Wind Detection System of Weather UAV Based on the Multi-mode Variable Structure," in *Intelligent Computation Technology and Automation (ICICTA), 2011 International Conference on*, 2011, pp. 765-768.

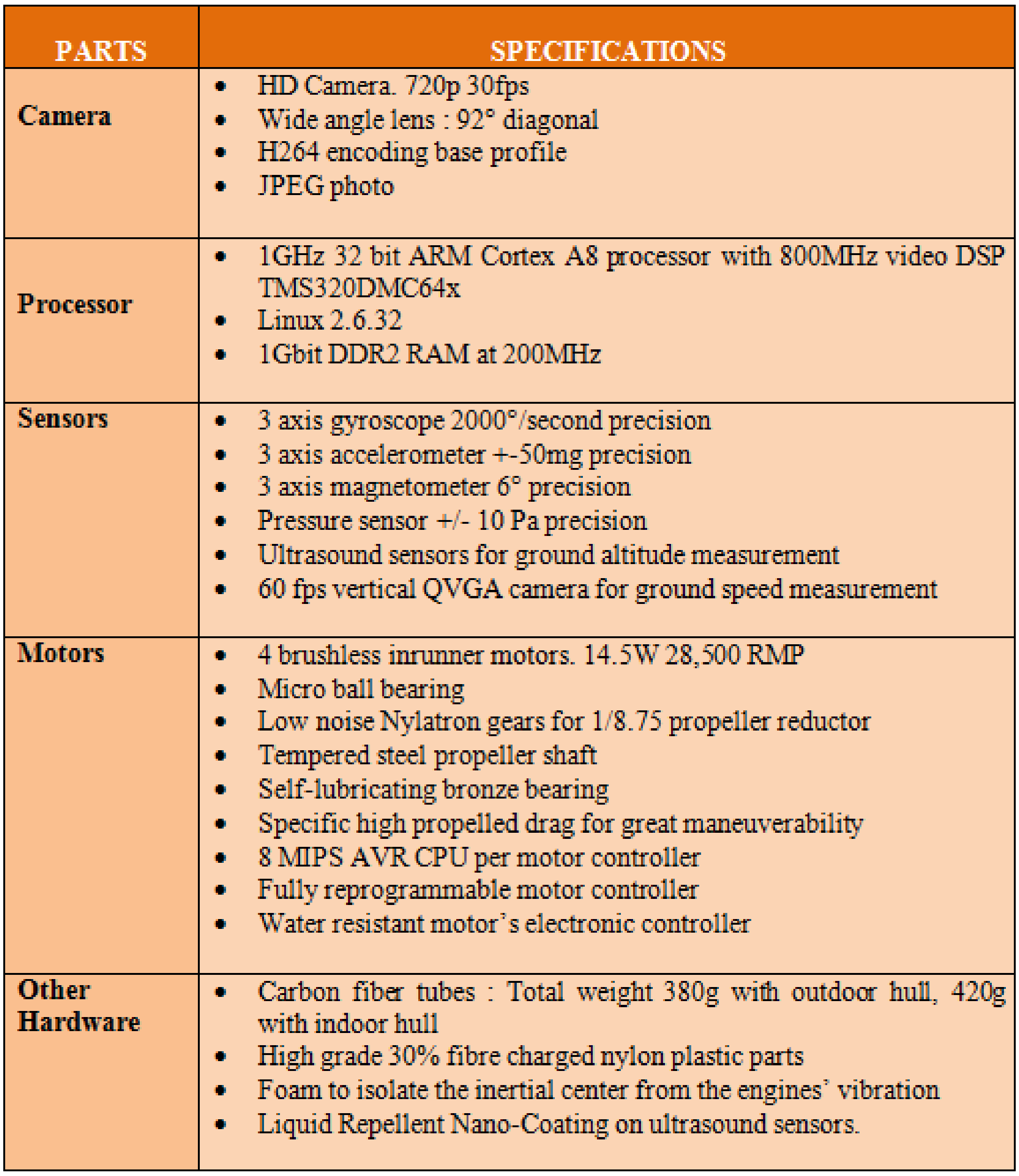
[21] Krajn´ık, T., Von´asek, V., Fiˇser, D., Faigl, J.: AR-Drone as a Platform for Robotic Research and Education. In *Research and Education in Robotics: EUROBOT 2011*, Heidelberg, Springer (2011)

[22] Raza, S. A. & Gueaieb, W. (2010). Intelligent Flight Control of an Autonomous Quadrotor. In Casolo, F. (Eds.), Motion Control. ISBN: 978-953-7619-55-8, InTech, DOI: 10.5772/6968.

[23] Thomas, D. (2014, May 22). ROS Introduction. Retrieved July 18, 2014, from http://wiki.ros.org/ROS/Introduction

APPENDIX A

➢ **Parrot AR.Drone 2.0 Technical Parts Specification**



PARTS	SPECIFICATIONS
Camera	• HD Camera. 720p 30fps • Wide angle lens : 92° diagonal • H264 encoding base profile • JPEG photo
Processor	• 1GHz 32 bit ARM Cortex A8 processor with 800MHz video DSP TMS320DMC64x • Linux 2.6.32 • 1Gbit DDR2 RAM at 200MHz
Sensors	• 3 axis gyroscope 2000°/second precision • 3 axis accelerometer +-50mg precision • 3 axis magnetometer 6° precision • Pressure sensor +/- 10 Pa precision • Ultrasound sensors for ground altitude measurement • 60 fps vertical QVGA camera for ground speed measurement
Motors	• 4 brushless inrunner motors. 14.5W 28,500 RMP • Micro ball bearing • Low noise Nylatron gears for 1/8.75 propeller reductor • Tempered steel propeller shaft • Self-lubricating bronze bearing • Specific high propelled drag for great maneuverability • 8 MIPS AVR CPU per motor controller • Fully reprogrammable motor controller • Water resistant motor's electronic controller
Other Hardware	• Carbon fiber tubes : Total weight 380g with outdoor hull, 420g with indoor hull • High grade 30% fibre charged nylon plastic parts • Foam to isolate the inertial center from the engines' vibration • Liquid Repellent Nano-Coating on ultrasound sensors.

(Retrieved from http://ardrone.parrot.com/ardrone/specifications/)