Autonomous Drone Network: Non-Intrusive Control and Indoor Formation Positioning

P Harrington

PhD

2019

Autonomous Drone Network: Non-Intrusive Control and Indoor Formation Positioning

Peter Harrington

A thesis submitted in partial fulfilment
of the requirements of the
University of Northumbria at Newcastle
for the degree of
Doctor of Philosophy

Research undertaken in the
Faculty of Engineering and Environment

October 2019

Abstract

The Teal Group estimated worldwide drone expenditure in 2013 to be $5.2 billion. Since then, worldwide drone expenditure has risen considerably, with the International Data Corporation (IDC) forecasting worldwide spending on drones to total $12.3 billion in 2019, with a compound annual growth rate (CAGR) forecast of 30.6% to 2022. As of 2019, Goldman Sachs report military applications account for 70% of the total spend with consumer applications accounting for 17%, and commercial/civil applications accounting for the remaining 13% where the latter are showing the fastest growth. Applications in construction, agriculture, offshore oil and gas, policing, journalism, border protection, mining and cinematography are predicted to see the greatest drone investment. As the demands increase, and particularly for applications that are time critical or that span large geographical areas, the single drone solution may be inadequate due to its limited energy and payload.

A multiple drone solution, where the drones are networked and the drone's position is established by GPS (global positioning system), is able to complete demanding applications more efficiently. In such systems however, the accuracy of GPS can be substantially compromised when deployed near tall buildings, trees, or bridges or if deployed indoors or underground.

In this research, a drone position determination (DPD) algorithm, is proposed to overcome the shortcomings of GPS when satellite signals are compromised. An ad-hoc Wi-Fi network of autonomous quadcopter drones is constructed, as a platform to demonstrate the algorithm performance. To complement the DPD algorithm calculation, a method to estimate the distance flown, and also estimate the complete flightpath of a drone by considering the interaction of the angular velocities of a quadcopter's four rotors (AVQR), is presented. The flight plan to examine the AVQR algorithm yields results enabling the distance flown to be calculated to an accuracy of 95%.

Acknowledgements

I would like to express my sincere gratitude to my principle supervisor Dr. Wai Pang Ng for his superb guidance, assistance and patience throughout the period of this research.

I would like to thank my wife Ping for her assistance in collecting data at Coach Lane Gym and at a cold autumnal Blyth South Beach. I would also like to thank her for her patience and understanding throughout the whole duration of the research.

Finally, a thank you to my mother and father, my children, Charlotte, Christopher and Adam, and my grandchildren Katelyn, Christopher, Bailey and Leo, who are all a constant source of inspiration to me.

Declaration

I declare that the work contained in this thesis has not been submitted for any other award and that it is all my own work. I also confirm that this work fully acknowledges opinions, ideas and contributions from the work of others.
Any ethical clearance for the research presented in this thesis has been approved. Approval has been sought and granted by the Faculty Ethics Committee / University Ethics Committee on 5 March 2012.

I declare that the Word Count of this Thesis is 22,858 words

Name: Peter Harrington

Signature:

Date: 29 November 2019

Table of Contents

Abstract I

Acknowledgements II

Declaration III

Table of Contents IV

Glossary of Acronyms VII

Glossary of Symbols IX

List of Figures XI

List of Tables XIII

Chapter 1 Introduction 1

1.1 Introduction and Motivation 1

1.2 Problem Statement 3

1.3 Aims and Objectives 5

1.4 Original Contributions 6

1.5 Publications 7

1.6 Thesis Organisation 9

Chapter 2 Quadcopter Drone Theory and Applications 11

2.1 Introduction 11

2.2 Quadcopter Flight Theory 11

2.3 On Board Drone Sensors 16

2.3.1 Accelerometer 16

2.3.2 Gyroscope 16

2.3.3 Magnetometer 17

2.3.4 Ultra Sonic Transducer 17

2.3.5 Radio Control or Wi-Fi 17

2.4 Wi-Fi literature review 18

2.5 Applications of UAVs 21

2.5.1 Surveillance 21

2.5.2 Drone Journalism 22

2.5.3 Disaster Management Applications 23

2.5.4 Pipeline Monitoring and Leak Detection 23

2.5.5 Delivery by Drone 24
2.5.6 Key Performance Indicators Relating to Drone Applications 25
2.6 Control of a Commercially Available UAV 26
2.6.1 Analysis of Data Traffic Between a UAV and Flight Control Platform........ 26
2.7 Summary 33
Chapter 3 Autonomous Quadcopter Drone Control 34
3.1 Introduction 34
3.2 Autonomous UAV Hardware Platform 34
3.3 Autonomous UAV Flight Control Algorithm 38
3.3.1 Drone Distance Flown Calculation 40
3.4 Drone Flight Control Program Development 43
3.5 Autonomous UAV Flight Plan Control 46
3.6 Results and Analysis 57
3.6.1 Flight Function Modification - Braking 59
3.6.2 Flight Function Optimisation - Overshoot 61
3.6.3 Results of Autonomous UAV Following a Square Flight Plan 64
3.7 Summary 66
Chapter 4 Estimation of UAV Flightpath from the Relative Angular Velocity of the Quadcopter Rotors (AVQR) 67
4.1 Introduction 67
4.2 The Rotor Torque Equation 68
4.3 Rotor Torque Equation Test Results and Analysis 73
4.3.1 Analysis of the Pitch and Roll Components of the Torque Equation 73
4.3.2 Analysis of the Yaw Component of the Torque Equation 80
4.4 Summary 83
Chapter 5 Autonomous Dual-UAV Control 84
5.1 Introduction 84
5.2 Autonomous Dual-UAV Hardware Configuration 84
5.3 Autonomous Dual-UAV Network Algorithm 87
5.4 Results/Discussion 96
5.5 Summary 98
Chapter 6 Drone Node Discovery Algorithm and Drone Position Algorithm........ 99
6.1 Introduction 99

6.2 Current Methods of Node Discovery and Drone Positioning 100
6.2.1 Node Discovery Algorithms 100
6.2.2 Drone Positioning Algorithms 100
6.3 Drone Node Discovery Algorithm 101
6.3.1 Drone Discovery Algorithm Results 104
6.4 Drone Position Algorithm 107
6.4.1 The RSSI Value as a Measure of Distance on a Wi-Fi Network 107
6.4.2 RSSI Distance Measurement 110
6.4.3 RSSI Measurements Taken From Flying Drones 113
6.4.4 Proposed Drone Position Algorithm 119
6.5 Summary 123
Chapter 7 Conclusion and Future Work 126
7.1 Conclusion 126
7.2 Future work 130
References 131

Glossary of Acronyms

ADC	Analogue to Digital Converter
ADFC	Autonomous drone flight control function
ARM	Advanced (reduced instruction set computer) machines
AT	Attention
AVQR	Angular velocity of quadcopter rotors algorithm
CAA	Civil aviation authority
CBS	Columbia broadcasting system
CAGR	Compound annual growth rate
DDR2	Double data rate 2
DHCP	Dynamic host configuration process
DoD	Department of defense
DPD	Drone position determination algorithm
DSP	Digital signal processer
FAA	Federal aviation administration
FPS	Frames per second
GNSS	Global navigation satellite system
GPS	Global positioning system
IDC	International data corporation
IDE	Integrated development environment
IEEE	Institution of electrical and electronic engineers
IoT	Internet of things
JPEG	Joint photographic experts group
KPI	Key performance indicator
LAN	Local area network
MIMO	Multiple input multiple output

MUMIMO	Multi-user multiple input multiple output
NAVSTAR	Navigation satellite timing and ranging
NODEMCU	Node microcontroller unit
OFDM	Orthogonal frequency division multiplexing
OS	Operating System
PWM	Pulse width modulation
QoS	Quality of service
QVGA	Quarter video graphics array
RAM	Random access memory
RC	Radio control
RSSI	Received signal strength indicator
SDK	Software development kit
TOAL	Take off and land function
UAV	Unmanned aerial vehicle
UDP	User datagram protocol
USB	Universal serial bus

Glossary of Symbols

A	Cross sectional area of rotor
ρ	Density of air
g	Acceleration due to gravity
I	Inertia matrix relating to Euler rigid body equation
$\mathrm{k_t}$	Torque proportionality constant
$\mathrm{k_{PWM}}$	Constant attributed to PWM
k_τ	Rotor blade configuration constant
$\mathrm{k_v}$	Proportionality constant from back emf generated per rpm
m	Mass of the drone
M	Applied torque in Euler rigid body equation
P_C	Power consumed by a rotating rotor
PWM_i	PWM value attributed to rotor i
P_T	Power required to generate required thrust
r	Radius of rotor blade
T	Thrust generated by four drone rotors
T_{dg}	Drone drag
T_i	Torque generated by rotor blade *i*
T_{hor}	Horizontal component of thrust
T_{ver}	Vertical component of thrust
τ	Torque
τ_i	Torque generated by rotor blade *i*
τ_{Drone}	Torque attributed to overall drone movement
τ_{pitch}	Torque attributed to drone pitch movement
τ_{roll}	Torque attributed to drone roll movement

τ_{yaw}	Torque attributed to yaw movement
v_i	Velocity of air passing through rotor i
v_x	Velocity of drone in the x direction
v_y	Velocity of drone in the y direction
ω	Rotor angular velocity
$\dot{\omega}$	Rotor angular acceleration
ω_i	Rotor angular velocity of rotor *i*

List of Figures

Figure 1-1 Research Road Map... 8
Figure 2-1 Quadcopter rotor configuration 12
Figure 2-2 Diagram showing drone pitch, roll and yaw movement 13
Figure 2-3 Diagram illustrating thrust and angles created as the drone pitches or rolls 14
Figure 2-4 Flight control command packet transmitted from controller to drone 28
Figure 2-5 Subset of navigation data captured by wireshark 32
Figure 3-1(a) Drone flight controlled by laptop (b) Drone flight controlled by NodeMCU module (DWi-Fi_0.1) 35
Figure 3-2 Circuit diagram of NodeMCU controlled autonomous drone. 37
Figure 3-3 Diagram illustrating distance calculation method 41
Figure 3-4 Program illustrating the fly forward or backward structure 44
Figure 3-5 Diagram showing flight plan to take off, hover, climb, fly forward, hover and land 46
Figure 3-6 Flowchart and associated program required to achieve the flight plan of Figure 3-5 47
Figure 3-7 Flowchart describing take-off algorithm 48
Figure 3-8 Flowchart describing 9 second hover algorithm 48
Figure 3-9 Flowchart describing algorithm to ascend to 1200mm 49
Figure 3-10 Flowchart describing fly forward 12m algorithm 50
Figure 3-11 Flowchart describing 3s hover (brake) algorithm 51
Figure 3-12 Flowchart describing land algorithm 52
Figure 3-13 Flowchart describing module DWi-Fi_0.1 algorithm 54
Figure 3-14 Flowchart describing module DWi-Fi_0.0 algorithm 55
Figure 3-15 Flowchart describing module DWi-Fi_LT algorithm 56
Figure 3-16 Graph of pitch angle, roll angle, yaw angle and drone height plotted against time 57
Figure 3-17 Graph showing the velocity in the x direction and the distance travelled plotted against time 58
Figure 3-18 Graph of velocity in the x direction and pitch angle plotted against time . 60
Figure 3-19 Graph of velocity in the x direction (after acceleration and deceleration modification), and pitch angle plotted against time 60
Figure 3-20 Graph of drone distance travelled against time at different velocities 61
Figure 3-21 Graph of overshot plotted against drone velocity in the x direction 62
Figure 3-22 Graph showing the x-y plot of the autonomous controlled drone flying a 7m square 64
Figure 4-1 Graph showing the angular velocity of the four rotors (PWM values) plotted against time 74
Figure 4-2 Graph showing the torque generated in the pitch and roll components plotted against time. 75

Figure 4-3 Graph showing drone velocities for velocity argument values of -0.1, -0.2 and -0.3 76
Figure 4-4 Graph of distance travelled and pitch torque plotted against time 77
Figure 4-5 Graph showing the pitch and roll torque plotted against time for a square flight plan 78
Figure 4-6 Graph of yaw torque for a 270 degree rotation with a velocity argument of 0.3 plotted against time 80
Figure 4-7 Graph of yaw torque for a 270 degree rotation with a velocity argument of 0.3 plotted against time 80
Figure 5-1 Diagram of two drone network configuration 85
Figure 5-2 Flowchart describing the two drone flight algorithm 88
Figure 5-3 Flowchart describing DWi-Fi_0.0 program algorithm 92
Figure 5-4 Flowchart depicting DWi-Fi_0.1 program algorithm 93
Figure 5-5 Flowchart describing DWi-Fi_1.0 program algorithm 94
Figure 5-6 Flowchart describing DWi-Fi_1.1 program algorithm 95
Figure 5-7 Graph showing take-off, height, and landing of two networked drones 96
Figure 5-8 Photographs showing the two drones in mid flight (a) both drones hovering, (b) leading drone, drone 1 commences landing, (c) drone 1 almost landed, drone 0 commencing landing. 97
Figure 6-1 Drone number and associated drone formation position. 103
Figure 6-2 Results of discovery algorithm execution 106
Figure 6-3 Radiation pattern measured at intervals of 3m, 6m, and 12m. 111
Figure 6-4 Graph of RSSI (dBm) values plotted against distance (m) 112
Figure 6-5 Graph of distance travelled (cm) and RSSI (dBm) plotted against time 114
Figure 6-6 Two drone flight to capture RSSI 1 (a) drone 0 takes off, drone 1 on the ground, (b) drone 0 hovers at 0.75m and drone 1 on the ground, (c) drone 0 hovers at 0.75m and drone 1 hovering at 0.75m, (d) drone0 ascends to 1.2m, drone 1 hovering at 0.75m 115
Figure 6-7 Two drone flight to capture RSSI 2 (a) Drone 0 and drone 1 hovering at 1.2m, (b) drone 1 begins to fly forward, (c) drone 1 continues its forward flight of 12m 116
Figure 6-8 Two drone flight to capture RSSI 3 (a) drone 1 flies 12m and hovers for 10s 117
Figure 6-9 Two drone flight to capture RSSI 4 (a) drone 0 hovers, drone 1 commences landing, (b) drone 0 commences landing, drone 1 landed (c) drone 0 lands. 118
Figure 6-10 Zone quadrant labelling 119
Figure 6-11 Diagram depicting drone position algorithm 120
Figure 6-12 Diagram illustrating the angles required to determine the drone zone 121

List of Tables

Table 2-1 Wi-Fi Standards and ranges 20
Table 2-2 Specification of Parrot AR2 drone 27
Table 2-3 Navdata_demo option transmitted as subset of navigation data 30
Table 2-4 Navigation data structure definition 31
Table 2-5 Examples of navigation data options 32
Table 3-1 SDK argument functionality table 38
Table 3-2 Flight control functions 39
Table 3-3 Table showing results of drone velocity and average overshoot 62
Table 3-4 Table showing results of drone velocity, distance travelled, and calculated overshoot 63
Table 3-5 Table showing the deviation in direction and distance from the 7m square flight plan 65
Table 4-1 Table showing drone velocity arguments and corresponding drone velocity. 76
Table 4-2 Table showing values of d_a, d_b and drone velocity for a velocity 78
Table 4-3 Table showing the estimated distance travelled for a square flight plan and associated accuracy 79
Table 4-4 Results showing impulse response at different angular velocities 81
Table 4-5 Table showing calculated angle of rotation and accuracy 82
Table 5-1 Table showing drone flight path control codes 87
Table 6-1 C program structure created within each DWi-Fi_x.0 station drone 102
Table 6-2 Comparison of RSSI measurements from flying and stationary drone 114
Table 6-3 Table showing the comparisons of RSSI readings, and the resulting zone .. 122

Chapter 1

Introduction

1.1 Introduction and Motivation

In 1898, Nikola Tesla, visionary and electrical engineering pioneer, demonstrated the first remotely controlled model boat [1]. Marincic and Budimir describe the demonstration as spectacular, even though many of the observers did not appreciate the significance of the invention [2]. Tesla believed this was the first robot representing a new category of machines that would enhance human life in a new way. Only nineteen years later in 1917, and only sixteen years after the Wright Brothers first manned flight, the Ruston Proctor Aerial Target, based on Tesla's previous work, became the first unmanned aircraft in history [3]. Unmanned aerial vehicles (UAV) continued to be developed, but these were generally large aircraft very much resembling their manned equivalent. It was not until the 1960's, when miniaturised components became available, that small radio controlled fixed wing model aircraft, that we recognise today, could be produced. These fixed wing UAVs are not only popular with the hobbyist but are also ideal for long range reconnaissance, mapping and military applications [4]. The obvious shortcoming of the fixed wing drone is their inability to hover and hold position over a ground location, ruling out aerial photography and reconnaissance applications. In 2010 the French company Parrot, released the first commercially available, out of the box, ready to fly quadcopter, the AR drone [5]. The quadcopter has four independent rotors driven by four electric motors, providing excellent stability, vertical take-off and hover capability, and making the drone generally straightforward to fly and control. The AR drone has both vertical and horizontal mounted cameras enabling excellent quality aerial photography and video to be produced by the relative beginner at low cost. Similar drone products soon followed with manufacturers, DJI releasing the Phantom in 2013, and 3DR releasing the Iris drone in 2014 [6] [7].

Since 2013 the growth in drone expenditure worldwide has grown considerably. The Teal Group estimated annual worldwide expenditure across the whole sector to be $5.2 billion in 2013 and predicted growth to reach $11.6 billion by 2023 [8]. By 2018 the International Data Corporation (IDC) forecast an increase to the Teal Group estimate with annual predicted expenditure to total $12.3 billion in 2019 with a compound annual growth rate (CAGR) forecast of 30.6% to 2022 [9]. Drones have their origins with military applications, being able to fly missions without risk to flight crew and at lower cost, and the military will continue to consume the greatest proportion of drone expenditure (70%) [10]. The drone hobbyist (consumer applications) is predicted to account for 17% of drone expenditure [10]. In 2019 there is a vast choice of drone manufacturer and choice of drone, depending upon the required application. A number of platforms now publish the top 100 companies (drone manufacturers) to follow [11]. As of 2018, the number of registered drone owners in the US according to FAA (Federal Aviation Administration) figures exceeds one million [12]. Civil and commercial applications account for the remaining 13% of drone expenditure but it is in this area where the greatest growth is predicted. Construction, where drones are used in building surveys, inspections, health and safety inductions, maintenance inspections, progress reports, promotional photography and thermal imagery, receives approximately 50% of the commercial expenditure [13]. Other growth applications in the commercial and civil sector include agriculture, offshore oil and gas, policing, journalism, border protection, mining and cinematography [10]

The ability of the quadcopter drone to partake in missions such as surveillance, search and rescue, and aerial photography result from the drone’s capacity to hover [14] [15] [16]. The efficient completion of such missions however is significantly impeded by the inability of the quadcopter drone to remain in the air for extended periods of time [17] [18]. Due to inherent inefficiency, the quadcopter drone requires considerable energy just to keep itself airborne. Current battery technology demands that the flight time is limited to a maximum of thirty minutes [19] . This shortcoming of the single drone has seen the emergence of multiple drone systems. Drones are connected and communicate over a network enabling greater area coverage for the same flight time of the single drone [20]. The multi drone network, whilst improving the performance of the single drone, yields a number of technological challenges requiring solutions to ensure efficient completion of the required mission [21].

1.2 Problem Statement

Drones forming a Multi UAV network can be controlled from a base station on the ground or the UAVs themselves can act autonomously, making, communicating and acting on decisions as the required mission develops[22]. Although the type of network deployed, and the UAV's themselves may vary in sensing capability, depending upon the required mission, all UAV networks require the following essential interconnected building blocks [20] [21] [23]:

i) Communication and Networking

   Responsible for data flow across the network including data acquired from onboard sensors, and the communication of flight command information. Maintains network connectivity, handles routing and scheduling, and ensures quality of service (QoS) requirements are met.

ii) Coordination

   Utilises data and information received from other UAVs on the network, and from its own sensor data, to make decisions with regard to flight path planning. Tasks can be distributed to single or multiple UAVs depending upon mission requirements.

iii) Sensing

   Sensors provide data essential to the completion of the required mission. The type of sensors mounted on the UAV will therefore depend upon the particular mission being undertaken. UAV position and the determination of other UAVs position on the network, essential to avoid obstacle and UAV collision, is achieved using GPS (global positioning system) [20] [24] [25].

Although the required building blocks for UAV networks have been identified, algorithm development continues to provide areas of research in which the aim is to optimise the performance of the network of UAVs in whichever application it is deployed. Building block iii) which describes GPS as the method of calculating UAV position provides the focus of interest in this research.

The GPS method of calculating position on the earth has its origins with the US military. In 1978 the US Department of Defense (DoD) launched the 24 satellite NAVSTAR (GPS) System [26]. Although designed for the US military, the GPS system became available for civilian use in 1983 albeit with a significantly reduced accuracy of 100m. Known as selective availability, the intentional degradation was implemented for national security reasons. In 2000, President Clinton revoked selective availability making GPS more responsive for civil and commercial use worldwide [27]. Other global navigation satellite systems (GNSS) are also in operation; GLONASS (Russia), Beidou (China), and QZSS (Japan). All GNSS are operated by the military within the associated nation and could be switched off at any time due to conflict. The world has become so reliant on GNSS in for example, banking, aviation, transport and business that lack of access would cause massive disruption. For this reason, to have GNSS autonomy, for technological advancement in research and innovation and to facilitate the potential development of new products, the European Union developed and funded their own GNSS, – Galileo [28]. The Galileo system began initial services in 2016 with 26 satellites in its constellation and is expected to reach full capacity of 30 satellites by 2020 [29].

GNSS requires that the receiver be in view of at least three satellites. Upon receiving the three signals from the satellites the receiver calculates its position using trilateration [30]. GNSS signals are transmitted with a certain accuracy however the accuracy of the position calculation at the receiver can depend upon a number of factors. Reflections of satellite signals due to trees or tall buildings can have a significant impact on accuracy. For example, a GPS enabled smartphone under open sky has a typical accuracy of 4.9m

[31] [32]. Also the more satellites that are in view the greater the potential accuracy of the position calculation [33] [30]. Wing *et al.* report GPS accuracy of 5m under open sky, 7m in young forest conditions and 10m under closed forest canopies [34]. Modsching *et al.* concluded GPS accuracy in a built up city of 15m where results were taken on a wide street with four story houses on both sides [35]. These are examples of the very conditions in which a drone network, in the applications of search and rescue or disaster management, could be deployed. The aim of this research is thus to develop an alternative to the GPS method of determining position which is applicable to UAVs within a UAV network, particularly for use in situations where the GPS signals are compromised.

1.3 Aims and Objectives

The aim of this research project is to develop a drone position determination (DPD) algorithm. The algorithm utilises the RSSI (Received Signal Strength Indicator) received from multiple Wi-Fi sources to enable each individual drone to calculate its position within a drone network. Wi-Fi is chosen as the preferred networking technology to communicate data between drones in this research. The main reasons for this choice is due to the data transmission rate of Wi-Fi (300Mb/s) being significantly superior to other data network technologies such as Zigbee (10-250Kb/s) enabling faster data communication, and also because Wi-Fi is used by the majority of current drone manufacturers as the method of communicating commands to their drones [36]. Although the DPD algorithm is demonstrated using a network of drones, the method has potential to operate in other scenarios such as to calculate the position of autonomous intelligent carts moving around a factory floor environment.

The key objectives of this research involve:

(i) To design software algorithms to enable autonomous quadcopter drone flight path control.

(ii) To propose an effective method to estimate the flightpath of a drone from the relative angular velocity of the quadcopters four rotors (AQVR).

(iii) To develop a platform incorporating a network of quadcopter drones, to measure the performance of the DPD algorithm.

(iv) To develop and implement a drone position determination (DPD) algorithm enabling real time drone position calculation.

1.4 Original Contributions

The original contributions to knowledge described in this thesis are summarised as follows:

a) The flight control programs are constructed using a modular approach. Each possible flight movement e.g. forward, backward, roll left, roll right, is written as an individual ‘C’ function incorporating arguments to control the velocity of movement and the distance flown. Combining the functions in the required order enables any required flight plan to be realised. Navigation data transmitted by the drone is captured, not only to measure the flight control program performance, but also to enable conditional flight distance control. The distance flown is calculated in real time by effectively integrating the velocity in the *x* direction v_x and the velocity in the *y* direction v_y provided within the navigation data. By recording each calculated incremental movement in the x and y direction every 20ms, when the navigation data is read, a graph of the complete drone flight path can be plotted. To enable flight missions to be realised, the distance flown by the drone in both the x and the y direction should have an accuracy of 95%.

b) The movement of a quadcopter drone in any direction is achieved by creating a relative difference in the angular velocity of the four rotors. A torque equation, developed from first principles, demonstrates that the flight path of a quadcopter drone can be estimated from the angular velocity of the quadcopter's four rotors (AVQR). The distance travelled in any particular direction provided by the torque equation should be accurate to 95%.

c) A network of autonomous quadcopter drones is designed and developed utilising commercially available Parrot AR2 drones. Each drone on the network is augmented with two IoT (Internet of Things) Wi-Fi modules to control the flight of the drone and to enable network capability. Drones on the network are able to communicate with each other, transmitting, receiving and responding to flight control codes to realise any desired flight plan.

d) A drone position determination (DPD) algorithm is proposed providing each drone on the network with the ability to calculate its own position in real time. The algorithm utilises the received signal strength indicator (RSSI) value to estimate the distance between a Wi-Fi transmitter and receiver. Measurements of RSSI are taken whilst the drone is instructed to complete a prescribed flight plan to enable the drone position to be calculated.

1.5 Publications

1) P. Harrington, W. P. Ng, "Investigation of the speed-up performance of a dual microcontroller parallel processing system in the execution of a mathematical operation," in *IEEE Communications PGNet*, Liverpool 2012.

2) P. Harrington, T. Chen, W. P. Ng, "Establishing the flightpath of a quadcopter drone from the relative angular velocity of the four rotors," in *11th IEEE/IET International Symposium on Communication Systems, Networks & Digital Signal Processing* (CSNDSP), Budapest, Hungary, 2018.

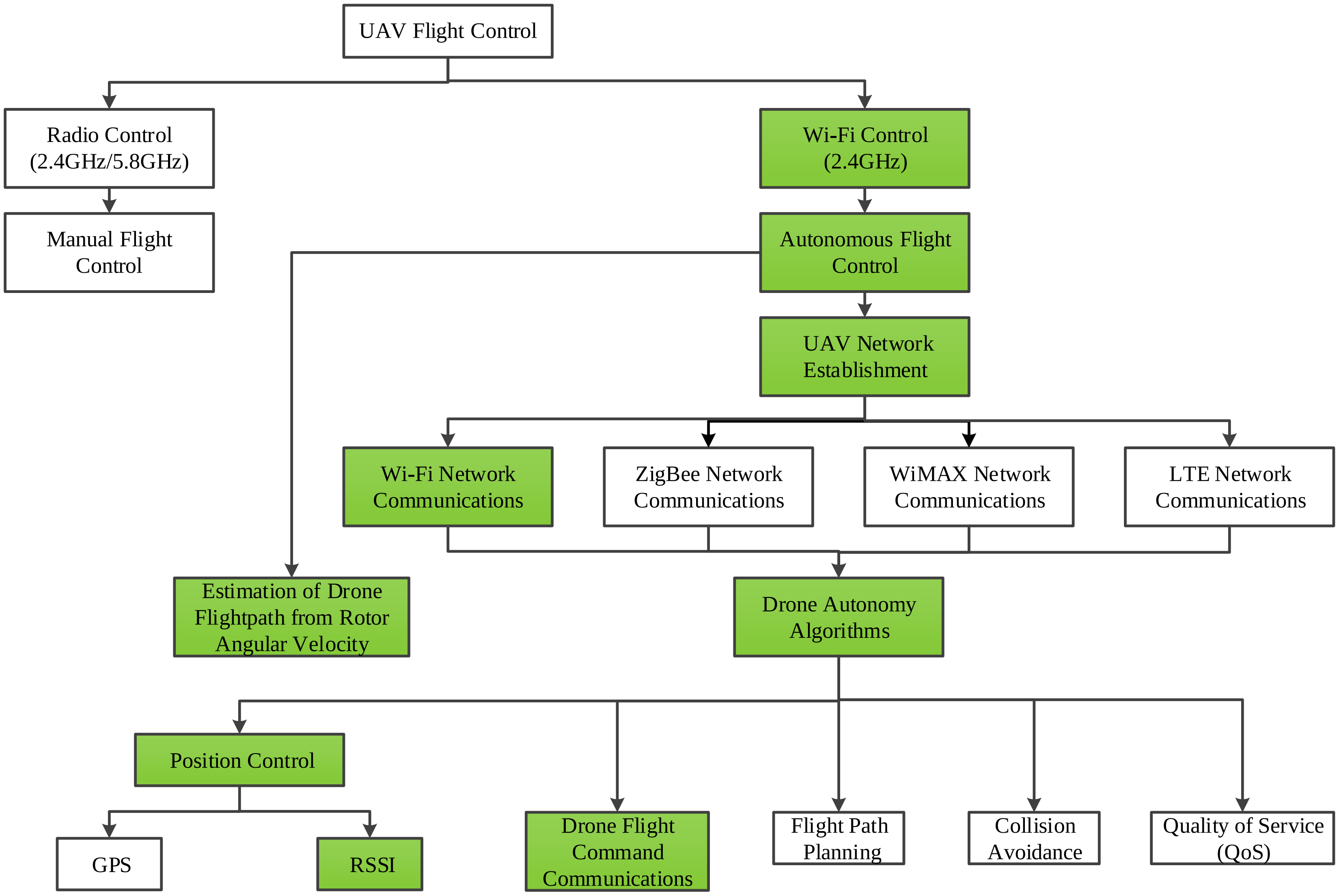

UAV Flight Control
Radio Control (2.4GHz/5.8GHz)
Wi-Fi Control (2.4GHz)
Manual Flight Control
Autonomous Flight Control
UAV Network Establishment
Wi-Fi Network Communications
ZigBee Network Communications
WiMAX Network Communications
LTE Network Communications
Estimation of Drone Flightpath from Rotor Angular Velocity
Drone Autonomy Algorithms
Position Control
GPS
RSSI
Drone Flight Command Communications
Flight Path Planning
Collision Avoidance
Quality of Service (QoS)

Figure 1-1 Research Road Map

1.6 Thesis Organisation

This thesis proposes an alternative method to GNSS which could be used in situations when GNSS signals are compromised. The delivery of this thesis is presented in 7 chapters.

Chapter 1 provides an introduction to UAVs, their history, and the evolution of the quadcopter drone. Applications of UAVs are briefly presented before the shortcomings of the single drone and the benefits of a multi drone network solution are discussed. Problem statement, aims and objectives, and original contribution are also discussed in this chapter. The theory behind the quadcopter drone are presented initially in **Chapter 2**, followed by a discussion of the sensors which assist the flight of a quadcopter drone. A literature review on Wi-Fi, which is commonly used as the communications channel for flight commands transmitted to drones is presented followed by a discussion of drone applications. The chapter concludes with a discussion of the attributes of the Parrot AR2 drone and a discussion of the commands and data transmitted and received by this drone.

Chapter 3 describes the development of the hardware platform and control algorithms to enable autonomous UAV control. Flight control programs are constructed utilising a modular approach, enabling developed modules to be combined to achieve any desired flight plan. Navigation data transmitted by the drone is captured enabling flightpath analysis.

In **Chapter 4** a method to estimate the flightpath of a quadcopter drone from a consideration of the angular velocity of the four rotors (AQVR) is presented. A torque equation developed from first principles incorporating the relative angular velocity of the four rotors not only provides a method for flightpath estimation but also provides a method to estimate of the distance travelled which could potentially be used to augment the calculation of distance travelled in Chapter 3.

Chapter 5 presents the development of an autonomous multi UAV platform. This 3-D platform is used to analyse the performance of the real time UAV position algorithm discussed in Chapter 6 to be analysed.

Chapter 6 describes the real time drone position algorithm. The algorithm utilises the received signal strength indicator (RSSI) values from multiple Wi-Fi sources, in order to determine UAV position.

Finally **Chapter 7** includes the thesis conclusion and recommendations for future work.

Chapter 2

Quadcopter Drone Theory and Applications

2.1 Introduction

In recent years quadcopter drones have received considerable interest from research institutions not only to explain the physics behind quadcopter drone flight, but also in developing drone applications. This chapter presents the theory behind quadcopter drone flight, describes a number of drone applications, and also explains the flight command packet transmitted, and the navigation data received, from a commercially available quadcopter – the Parrot AR2 drone

2.2 Quadcopter Flight Theory

Elevation of the quadcopter drone is provided by four rotors powered by four brushless dc motors. Unlike a standard helicopter the quadcopter does not have a rear rotor to provide stability and prevent rotation in the z axis. The diagram of Figure 2-1 illustrates how the quadcopter drone eliminates unwanted rotation by configuring adjacent rotors to rotate in the opposite direction. Rotor 1 thus rotates in the clockwise direction whilst rotor 2 rotates in the counterclockwise direction, and rotor 3 rotates in the clockwise direction whilst rotor 4 rotates in the counterclockwise direction.

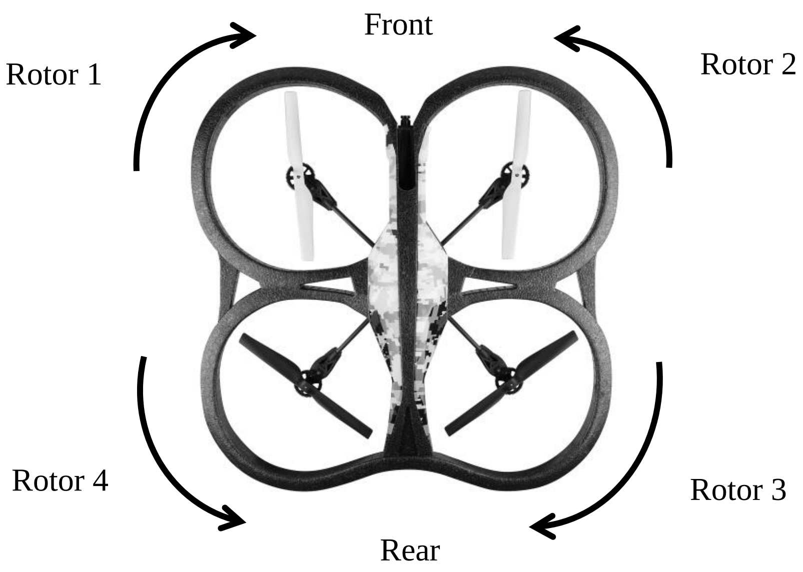


Figure 2-1 Quadcopter rotor configuration

The force, or thrust, required to elevate the drone from the ground is provided by the four rotors. The thrust is calculated from (Equation 2-1) where T_i is the thrust generated by a single rotor defined by subscript *i*, ρ is the density of air, A is the cross sectional area of the rotor, and v_i is the velocity of air generated by a particular rotor defined by the subscript *i* [37].

$$T_i = \rho A v_i^2 \qquad \text{(Equation 2-1)}$$

The total thrust *T*, is therefore determined by the sum of thrust provided by the four rotors.

$$T = \rho A \sum_{i=1}^{4} v_i^2 \qquad \text{(Equation 2-2)}$$

An important parameter when developing a quadcopter is the thrust to weight ratio. To elevate the drone the thrust to weight ratio must be greater than 1. In reality quadcopter drones are designed with a minimum thrust to drone ratio of 2 to ensure the rotors can elevate the drone from the ground and also have the capacity to ascend and manoeuvre once the drone is in the air [38]. When hovering the thrust delivered by the rotors is balanced by the force due to gravity as described in Equation 2-3, where *m* is the mass of the drone and *g* is the acceleration due to gravity.

$$\rho A \sum_{i=1}^{4} v_i^2 = mg \qquad \text{(Equation 2-3)}$$

The rotors of a quadcopter drone are fixed, i.e. they cannot change their angle relative to the drone to create movement. The drone can only move in a particular direction by pitching forward or backwards (x direction), or rolling to the left or right (y direction). Drone yaw movement is a rotation about the z axis. Pitch, roll and yaw angle movement is illustrated in Figure 2-2.

Pitch, roll and yaw angles are created due to a difference in the relative angular velocity of appropriate rotors. Pitching forward and therefore creating movement in the forward direction, for example, results from the two rotors at the rear increasing their relative velocity relative to the two rotors at the front. The diagram of Figure 2-3 illustrates the angles and thrust components, T_{ver} vertical thrust, T_{hor} horizontal thrust, and T thrust, created as the drone either pitches forward or rolls, and T_{dg} the drone drag.

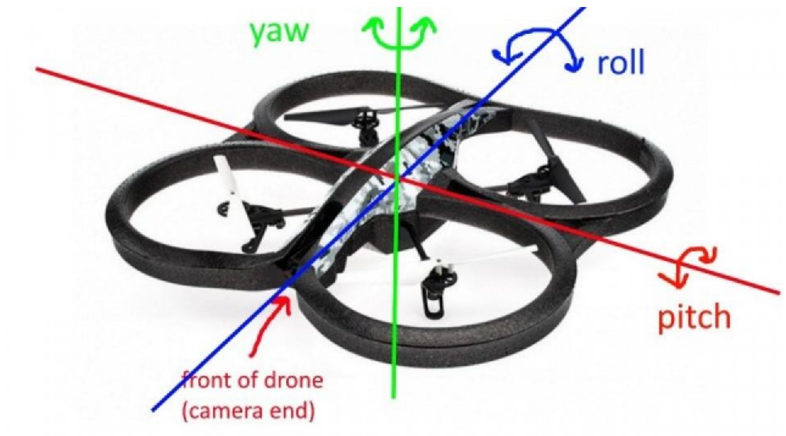


Figure 2-2 Diagram showing drone pitch, roll and yaw movement

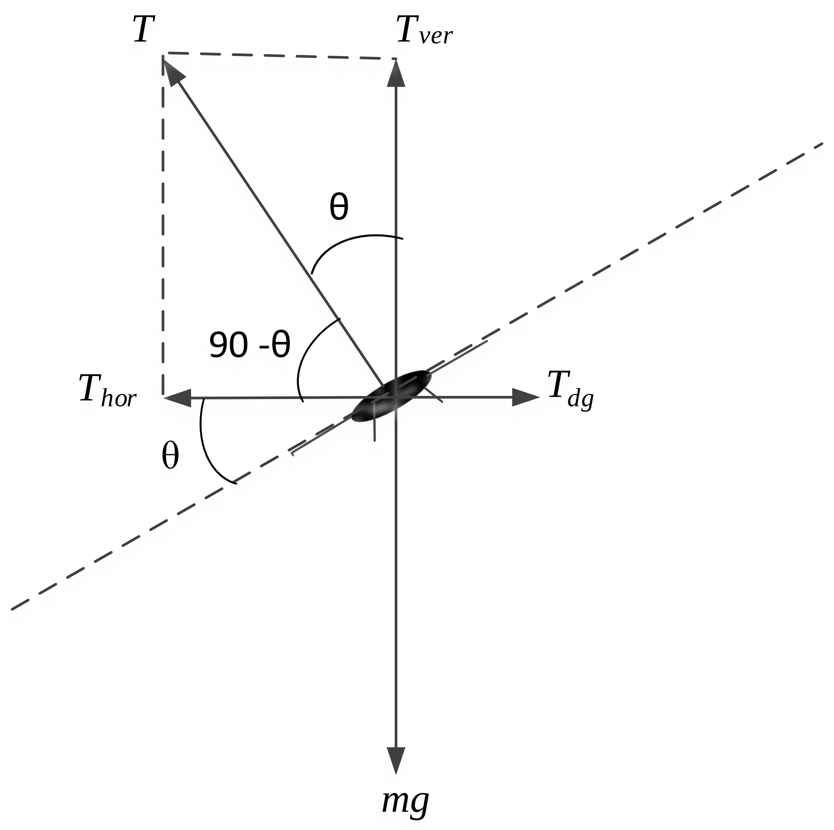


Figure 2-3 Diagram illustrating thrust and angles created as the drone pitches or rolls

In the diagram of Figure 2-3, *mg* is the vertical force due to gravity. As the drone pitches or rolls, the drone maintains the same height, therefore the vertical thrust is given by,

$$T_{ver} = mg \qquad \text{(Equation 2-4)}$$

The force in the horizontal direction is given by,

$$T_{hor} = T\cos(90 - \theta) \qquad \text{(Equation 2-5)}$$

As the drone pitches or rolls the overall thrust, *T*, provided by the drone rotors clearly increases and is given by,

$$T^2 = T_{hor}^2 + T_{ver}^2 \qquad \text{(Equation 2-6)}$$

The force to move the drone in the horizontal direction is given by *ma* which is equal to the thrust in the horizontal direction T_{hor} minus the drag T_{dg} or,

$$T\cos(90 - \theta) - T_{dg} = ma \qquad \text{(Equation 2-7)}$$

A summary of the forces required to be generated by the drone, and the external forces acting on the drone which must be overcome to enable drone flight, are listed below.

(i) For the drone to elevate itself from the ground, the thrust to weight ratio should be greater than 1, i.e. the thrust developed in the four rotors must exceed the force due to gravity, (Figure 2-3).

(ii) After take-off, in order for the drone to hover, the thrust developed by the four rotors must be equal to the force due to gravity, (Equation 2-4).

(iii) In order to ascend, the thrust developed by the four rotors must be greater than the force to due gravity and conversely for the drone to descend the thrust developed by the four rotors must be less than the force due to gravity, (Equation 2-4).

(iv) To fly in the forward direction the two rear rotors (rotor 3 and rotor 4) rotate at a greater angular velocity than the two front rotors (rotor 1 and rotor 2). As a result the drone pitches forward, but in order to maintain height the vertical component of thrust is now equal to the force due to gravity. In order to move in the forward direction a force is required in the horizontal direction which is equivalent to the horizontal component of thrust generated by the four rotors, minus the drone drag, (Equation 2-7).

(v) To fly in reverse the forces are as described in (d) but with rotors 1 and 2 rotating at a greater angular velocity than rotors 3 and 4.

(vi) To roll to the left, the forces are as described in (d) above but with rotors 2 and 3 rotating at a greater angular velocity than rotors 1 and 4.

(vii) Finally to roll to the right, the forces are as described in (d) above but with rotors 1 and 4 rotating at a greater angular velocity than rotors 2 and 3.

2.3 On Board Drone Sensors

In order to maintain a stable controlled flight the drone requires a number of on board sensors which are discussed below.

2.3.1 Accelerometer

The accelerometer is a device, found in most drones, to determine the acceleration of the drone in the x, y and z planes [39]. Electronic accelerometers use the piezoelectric effect, where an acceleration in the device creates a stress on the crystals inducing a variation in capacitance. The change in capacitance is converted to a voltage which is interpreted as a measure of acceleration. The acceleration can be used to assist in the calculation of the drone velocity.

2.3.2 Gyroscope

The classic gyroscope consists of a disc or wheel which can rotate rapidly about an axis which is itself free to rotate. Solid state gyroscopes which are generally used in drones, use the Coriolis force [40] to measure angular velocity, and assist in maintaining drone stability and orientation. The Coriolis force occurs as a result of an angular rotation being applied to a moving body. Gyroscopes of this type have many applications including, in aircraft to assist in orientation stabilisation, in cameras to determine hand movement and in motor vehicles to assist in accident prevention.

2.3.3 Magnetometer

The accelerometer and gyroscope determine acceleration and yaw movement respectively, but only relative to the drone’s starting position. The magnetometer uses the Hall-effect to measure magnetic field strength. The drone uses a magnetometer to determine the earth’s magnetic field and thus provide the drone with directional information. In conjunction with the accelerometer and gyroscope the drone has information regarding acceleration, yaw movement and direction.

2.3.4 Ultra Sonic Transducer

The drone utilises an ultra-sonic transducer to determine altitude. The ultra-sonic transducer transmits ultra-sonic pulses towards the ground which are reflected back and are received by the ultra-sonic transducer. The time of flight of the ultra-sonic pulses is proportional to the distance travelled, enabling altitude calculation [41].

2.3.5 Radio Control or Wi-Fi

Historically model aircraft and boats have been controlled via RC (radio control), however more recently UAVs are being developed to be controlled via Wi-Fi. This enables any platform with Wi-Fi capability such as a tablet or mobile phone running the required app to control the flight of the UAV. Since Wi-Fi control can only provide a range of approximately 50m compared with RC which can provide ranges up to 18km (conditions dependant), drone user forums suggest RC is the preferred method for drone control [42] [43] . They also prefer the ergonomics and tactile feel that joystick control provides over on screen control buttons available on a tablet or smart phone [44]. Wi-Fi control however, does provide the possibility of creating drone Wi-Fi networks and since the drone used in this research, the Parrot AR2 drone, is controlled via Wi-Fi, a Wi-Fi literature review follows.

2.4 Wi-Fi literature review

Internet usage has grown from 16 million in 1995 to 4.4 billion (56.8% of the world population) in 2019 worldwide [45]. In Great Britain, by 2016, 82% of adults went on line daily (or almost daily) compared with 35% in 2006 [46]. By 2019, 91% of all adults had recently (within three months) used the internet with 99% of adults between 16 to 44 years being recent internet users [47]. This growth has been made possible in part by the evolution of Wi-Fi which has enabled wireless internet connectivity to laptops, smartphones and tablets [48]. Through the availability of Wi-Fi, wireless internet connectivity has been made possible in Universities, Colleges and schools, buses trains and aircraft, shops and hotels, in cities and within the home. As well as providing internet connectivity, Wi-Fi has also been at the heart of the IoT explosion with a plethora of Wi-Fi devices with built in sensors becoming available, enabling for example, thermostats, ovens and fridges to become networked and controlled over the internet [49].

Historically, it was back in the mid-1980s, when it was realised that a common wireless standard to enable the interconnection of network equipment from different vendors was required. In 1988 an Institute of Electrical and Electronic Engineers (IEEE) committee chaired by engineer Victor Hayes was convened to draw up a standard that would be acceptable to all vendors [50]. The new committee was named 802.11, a title which has become synonymous with Wi-Fi, and which is still in use today. It was not however until 1997 that the committee agreed the 802.11 specification. The standard allowed for a data transfer rate of one or two megabits per second (Mbit/s) on the 2.4GHz frequency band [48]. Amendments to the original 802.11 standard were soon developed. Some of the more common of these amendments are described below [51].

802.11b (1999)	Still operating in the 2.4GHz band but with a maximum data rate of 11MBits/s. The increase in data throughput led to the rapid acceptance of 802.11b as the definitive wireless LAN technology. Can experience interference from products using the same 2.4GHz frequency e.g. microwave ovens.

802.11a (2000)	Operating in the 5 GHz band, employing OFDM (Orthogonal Frequency Division Multiplexing) allowing data rates of up to 54Mbit/s (including error correction code, effective speed 50% of rated speed). The shorter wavelength of 802.11a compared with 802.11b means that signals are more easily absorbed by walls and solid obstacles. In theory, 802.11b has a greater range than 802.11a.
802.11g (2003)	Designed to combine the best of 802.11a and 802.11b. Operates in the 2.4GHz band, employs OFMD and 54Mbits/s data rate (22Mbits/s including error detection codes). Rapidly adopted due to high data rates and lower manufacturing costs.
802.11n (2009)	Utilises multiple wireless signals and antennas, Multiple Input and Multiple Output (MIMO). Maximum theoretical data rate of 450MBits/s. Operates on both 2.4GHz and 5 GHz frequency bands.
802.11ac (2013)	The standard found in the majority of current Wi-Fi devices, builds on the performance of 802.11n operating in the 5GHz frequency band. Development enhancements including Multiple MIMO (MUMIMO) provide a data rate of up to 1.3GBits/s.
802.11ax (2019)	Due for ratification by the IEEE in 2019. Demonstrations show a maximum data rate of 11GBits/s.

As well as providing wireless network connectivity, Wi-Fi is now also being used as the communications channel to control the flight of quadcopter drones. Traditionally remote control aircraft, cars and boats were controlled by a dedicated radio transmitter. Manufacturers of quadcopter drones have taken advantage of technology by enabling flight commands to be transmitted to drones by mobile phones or tablets via a Wi-Fi communications channel. The requirement of the manufacturer to provide a radio transmitter to control the flight of the drone is thus eliminated, reducing costs. Manufacturers supplying such drones include DJI (Mavic 2 Zoom, Mavic 2 Pro), Skydio (Skydio R1), Yuneec (Mantis Q), Parrott (Minidrone Spider, AR2), [6] [52] [53] [42]. Utilising Wi-Fi as the method of communications to control drones opens up the potential of creating a Wi-Fi drone network.

Using Wi-Fi as a drone communication channel does however have limitations. A list of Wi-Fi standards and ranges is presented in Table 2-1. From Table 2-1 it is observed that the different IEEE standards offer a very similar indoor range with the exception of 802.11n standard. Also, as the distance between transmitter and receiver increases, the signal strength at the receiver falls, lowering the bandwidth and data transmission rate. Experiments carried out by Galbraith and Brown show a reduction in data transmission rate from 547Mbps at 2m to 12Mbps at 60m [54]. Wi-Fi thus exhibits limitations as a communications channel, however the impact of these limitations in this research, is minimal. The largest distance between transmitter and receiver during experimentation in this research is governed by the size of the gym in which the experimentation takes place (25m x 15m). The greatest distance is therefore the diagonal of the 25m x15m rectangle which is 29.15m.

Table 2-1 Wi-Fi Standards and ranges

Wi-Fi Standard	Indoor range (m)
802.11b	35m
802.11a	35m
802.11g	38m
802.11n	70m
802.11ac	35m

This distance is less than the maximum range specified in Table 2-1 and therefore does not impact on the integrity of the communications channel. Also the maximum amount of data communicated in one packet between transmitter and receiver in this research is 990 bytes, the data transmitted as the flight control command packet depicted in Figure 2-4. The impact of the reduced transmission rate over larger distances is minimal for the relatively small number of bytes transmitted to the drone via the Wi-Fi communications channel.

2.5 Applications of UAVs

Drones are found in an ever increasing number of applications. A number of these applications are discussed in this section.

2.5.1 Surveillance

The vast majority of drones carry an onboard camera capable of taking video as well as photographs. Gandhi and Ghosal discuss a drone designed for military surveillance. The size, hovering capability and relatively low cost of the drone make it an excellent option for deployment in such applications. The drone can produce photographs and video of areas of interest removing the necessity for direct manual intervention and minimising the potential for loss of life whilst performing the operation [55]. As well as military surveillance drones also find applications in tactical surveillance in police applications. Such a system discussed by Rangel and Terra carries out urban surveillance relaying real time information to the ground for evaluation and action. Since its deployment in 2015 they describe effective results in drug control, mapping and pest control [56]. Depending on the subject of surveillance, drones are often modified with appropriate sensors to monitor the area of interest. Patel *et al.* develop a bespoke drone for use in an agricultural application incorporating an infra-red camera which will provide a colour image showing the difference between diseased and mature crops [57]. The major shortcoming of drone deployment in surveillance applications is the limited time the drone can remain in the air. Current battery technology limit drone flight times to a maximum of approximately 30 minutes. Williams and Yakemenko present a solution to the battery issue by implementing a system of drone swapping enabling the area of surveillance to be monitored for an indefinite period [58]. An alternative solution to the limited flight time of a single drone is to adopt a system of a multi drone network. Capitain, Marino and Ollero propose such a system where data is communicated between drones and a base station, and enables greater coverage than would be possible with a single drone [59].

2.5.2 Drone Journalism

Capturing images of places of interest, areas of natural disaster, and areas devastated by war activity is not a new idea. An article in the Daily Mail reports in 1906 George R. Lawrence captured birds eye view photographs of American cities by rigging a 49 pound camera to a system of kites [60]. The photographs are truly remarkable providing images from a position that had never before been seen. Until the emergence of drones aerial journalism was the remit of small aircraft and helicopters. One of the first examples of drone aerial journalism occurred after a disaster in Arkansas in 2014 when tornados devastated cities across the state [61]. News organisations struggled to provide information and helicopters could not fly due to the bad weather. A drone was flown to capture some of the worst case areas and shared across multiple news websites. Drones have since grown in popularity to cover news stories resulting in the term ‘drone journalism’ being coined for such activities. Tremayne and Clarke, and Schroyer, emphasise the importance of using drones in modern day journalism [62] [63]. The use of drones for journalistic applications does incur a number of issues however. Photographing or videoing a battlefield, village or town where there may be a large number of casualties raises ethical issues [64]. Filming a celebrity’s property, or a member of the general public raises the issue of privacy [65]. Flying a drone near an airfield raises the issue of danger with potentially catastrophic consequences due to the possibility of collision with aircraft in the process of taking off or landing [66] . To counter these issues the FAA (Federal Aviation Authority) in the USA require that for all commercial use, for drones under 55 pounds, the drone is registered and the pilot has a Part 107 license [67]. For all recreational use the drone must be registered [68]. In the UK, airspace is regulated by the CAA (Civil Aviation Authority). For commercial use a permission from the CAA is required [69]. For recreational use drones between 250g and 20kg must be registered by the end of November 2019 [70].

2.5.3 Disaster Management Applications

Considerable work has been conducted utilising drones in the area of disaster management. Drones can provide effective assistance in several stages of such scenarios. Erdelj and Natalizio propose a three stage operational life cycle. The first is pre-disaster preparedness, effectively setting up early warning systems. The second is disaster assessment, providing real time data of the disaster to enable logistical planning. The third is disaster response and recovery including search and rescue [71]. Developing their previous work Erdelj and Natalizio suggest drones could also re-establish damaged communication infrastructure and deliver essential medical supplies [72]. Saha suggests a similar drone solution incorporating an autonomous drone with GPS [73]. Camera proposes utilising a fleet of drones scanning a disaster stricken area to provide real time data from which response strategies can be formulated [74]. Although applications of drones in disaster management are in their infancy, drones have been used in disasters to assist with search and rescue operations and provide damage information. Dozens of drones were deployed both in Houston in response to Hurricane Harvey and Florida in response to Hurricane Irma [75].

2.5.4 Pipeline Monitoring and Leak Detection

The use of drones in pipeline and leak detection is being researched. Due to the drone being non-intrusive and highly mobile the use of drones in this activity is receiving increasing attention. Shukla *et al.* discuss the use of an autonomous drone to discover and track an oil or gas pipeline at low altitude. Utilising the on board cameras for navigation and techniques for edge and line detection they develop algorithms to realise the detection and tracking of pipelines [76]. Essentially to minimise costs Bretschneider and Shetti propose a method of methane leakage detection from pipelines utilising a drone mounted with optical and laser based detectors [77]. A method to detect water leakage in buried pipes by mounting a thermal imaging camera on a quadcopter drone is proposed by Shakmak and Al-Abaibeh [78].

2.5.5 Delivery by Drone

In 2013 Amazon CEO announced on CBS news that his company had developed a fleet of unmanned aerial vehicles (UAVs) for small parcel delivery [79]. Delivery by drone poses problems requiring innovating solutions and is receiving interest from the research community. Murray and Chu propose models to optimise routing and scheduling of unmanned aircraft [80]. Similarly Dorling *et al*. propose two solutions to vehicle routing problems also taking into account the effect of the battery, and payload weight on energy consumption [81]. Delivery by drone does yield potential benefits. Goodchild and choy discuss the benefits of drone delivery in reducing CO2 emissions [82]. The reduction of motor vehicles and associated benefits using drones in making last mile deliveries is discussed by Gulden [83].

Although it is difficult to imagine that governments would permit the delivery of parcels to the general public by this method, drones are already making deliveries in exceptional circumstances. Reuters reports the first authorized unmanned delivery of goods in Europe: the delivery of essential medical goods to the German island of Juist[84]. The delivery of medicines to Haiti following the devastating earthquake in 2010 is discussed by Choi-Fitzpatrick *et al.* [85].

2.5.6 Key Performance Indicators Relating to Drone Applications

A number of key performance indicators (KPIs) can be derived from the applications discussed in the previous sub-sections. Drone surveillance and disaster management applications require a number of aspects of drone control.

KP1: The drone must take-off and land when commanded to do so.

KP2 The drone must be able to ascend to the required height in order to be able to survey the area of interest.

KPI3 The drone must be able to hover over any required position for the time required to complete the mission.

KPI4 The drone must have the ability to communicate data relative to the associated application back to the control station e.g. photographs and video.

As well as the above, additional KPIs can be derived for pipeline monitoring and leak detection applications.

KPI5 The drone must be able to follow a pre-defined flight path e.g. to follow a pipeline looking for leaks or to fly to a delivery destination.

KPI6 The drone must be able to return from the destination back to the starting location.

KPI7 The drone must be able to detect and avoid obstacles in its path.

KPI8 For delivery applications the drone must be able to complete the required flight with the additional payload of what is to be delivered.

Although this section has covered the main applications and the current areas of research related to drones, new applications are continuously emerging as the technology evolves.

2.6 Control of a Commercially Available UAV

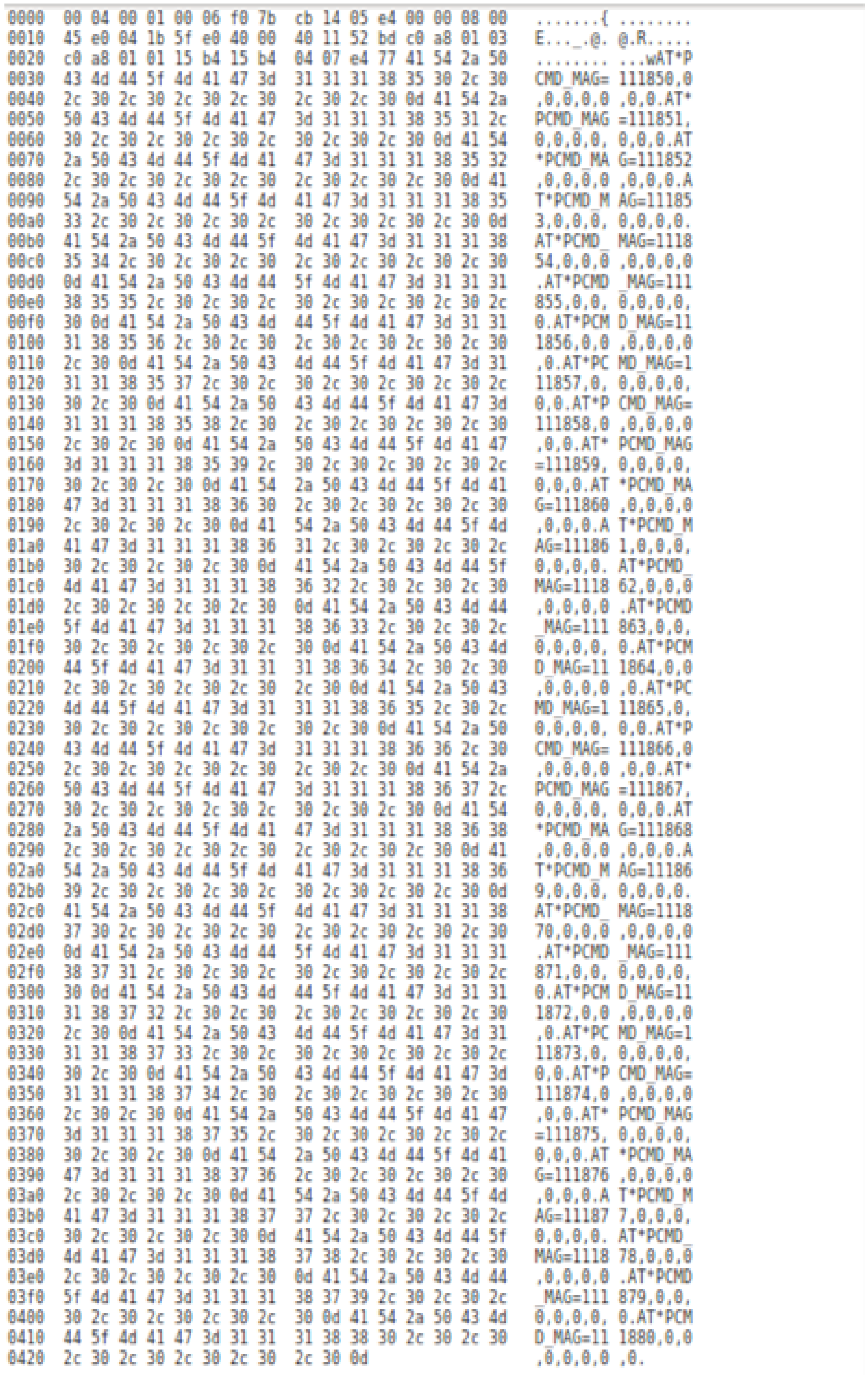
The Parrot AR2 drone is controlled via an app running on a smartphone or a tablet. Due to its general low cost and ease of flight control, the Parrot AR2 drone has been used extensively in research projects and is used as the drone of choice in this project [86] [87] [88] [89]. In 2012, the Parrot AR2 drone replaced the original Parrot AR drone. Although similar in physical appearance, the AR2 drone benefits from enhancements to on board equipment and sensors. The on board camera quality was increased to 720p. Sensor improvements included an upgraded 3-axis gyroscope along with a 3-axis accelerometer and magnetometer. Improvements to the battery enabled 50% longer flight times. A complete specification of the Parrot AR2 drone is described in Table 2-2 [41].

2.6.1 Analysis of Data Traffic Between a UAV and Flight Control Platform

When power is applied to the AR2 drone, it configures itself as a Wi-Fi access point. When the control platform (smartphone, tablet or computer), connects to the access point a Wi-Fi communications channel is established, which enables flight control AT commands to be transmitted from control platform to drone, and navigation data to be transmitted from drone to control platform. Flight control AT commands are transmitted as a packet consisting of 30 flight commands (actually the same command) every 20ms via UDP (user datagram protocol) on port 5556. The data of Figure 2-4 depicts a flight control command packet captured via the network traffic capture software ‘wireshark’, whilst the drone is hovering. The first column of Figure 2-4 displays the memory location of the byte values of the flight command packet that are transmitted to the drone. The second and third columns, both consisting of eight bytes, contain the hexadecimal values of the flight command packet transmitted to the drone. The fourth and fifth columns, again consisting of eight values each, contain the alphanumeric values corresponding to the hexadecimal values of columns two and three. The first command in the data packet begins at location 0x002c with the hexadecimal value $41 equivalent to alphanumeric character A. The final character of the command is located at memory location 0x004C with the hexadecimal value 0x0d equivalent to the alphanumeric character ‘full stop’.

Table 2-2 Specification of Parrot AR2 drone

HD VIDEO RECORDING	720p 30fps HD camera
	Wide-angle lens: 92° diagonal
	Basic encoding profile: H264
	Photo format: JPEG
	Connection: Wi-Fi
ELECTRONIC ASSISTANCE	Processor: ARM Cortex A8 1 GHz 32-bit processor with DSP video 800MHz
	TMS320DMC64x
	OS: Linux 2.6.32
	RAM: DDR2 1 GB at 200 MHz
	USB: High-speed USB 2.0 for extensions
	Wi-Fi 802.11 b/g/n
	Gyroscope: 3 axes, accuracy of 2,000°/second
	Accelerometer: 3 axes, accuracy of +/- 50mg
	Magnetometer: 3 axes, accuracy of 6°
	Pressure sensor: Accuracy of +/- 10Pa
	Altitude ultrasound sensor: Measures altitude
	Vertical camera: QVGA 60 FPS to measure the ground speed
MOTORS & WEIGHT	4 "inrunner" type brush-free motors: 14.5 watts and 28,500 rev/min
	Micro ball bearing:
	Nylatron Gears:
	Bronze self-lubricating ball bearings:
	Weight: indoor frame 380 g/ outdoor frame 420g



```
0000  00 04 00 01 00 06 f0 7b  cb 14 05 e4 00 00 08 00   ........{ ........
0010  45 e0 04 1b 5f e0 40 00  40 11 52 bd c0 a8 01 03   E..._.@. @.R.....
0020  c0 a8 01 01 15 b4 15 b4  04 07 e4 77 41 54 2a 50   ........ ...wAT*P
0030  43 4d 44 5f 4d 41 47 3d  31 31 31 38 35 30 2c 30   CMD_MAG= 111850,0
0040  2c 30 2c 30 2c 30 2c 30  2c 30 2c 30 0d 41 54 2a   ,0,0,0,0 ,0,0.AT*
0050  50 43 4d 44 5f 4d 41 47  3d 31 31 31 38 35 31 2c   PCMD_MAG =111851,
0060  30 2c 30 2c 30 2c 30 2c  30 2c 30 2c 30 0d 41 54   0,0,0,0, 0,0,0.AT
0070  2a 50 43 4d 44 5f 4d 41  47 3d 31 31 31 38 35 32   *PCMD_MA G=111852
0080  2c 30 2c 30 2c 30 2c 30  2c 30 2c 30 2c 30 0d 41   ,0,0,0,0 ,0,0,0.A
0090  54 2a 50 43 4d 44 5f 4d  41 47 3d 31 31 31 38 35   T*PCMD_M AG=11185
00a0  33 2c 30 2c 30 2c 30 2c  30 2c 30 2c 30 2c 30 0d   3,0,0,0, 0,0,0,0.
00b0  41 54 2a 50 43 4d 44 5f  4d 41 47 3d 31 31 31 38   AT*PCMD_ MAG=1118
00c0  35 34 2c 30 2c 30 2c 30  2c 30 2c 30 2c 30 2c 30   54,0,0,0 ,0,0,0,0
00d0  0d 41 54 2a 50 43 4d 44  5f 4d 41 47 3d 31 31 31   .AT*PCMD _MAG=111
00e0  38 35 35 2c 30 2c 30 2c  30 2c 30 2c 30 2c 30 2c   855,0,0, 0,0,0,0,
00f0  30 0d 41 54 2a 50 43 4d  44 5f 4d 41 47 3d 31 31   0.AT*PCM D_MAG=11
0100  31 38 35 36 2c 30 2c 30  2c 30 2c 30 2c 30 2c 30   1856,0,0 ,0,0,0,0
0110  2c 30 0d 41 54 2a 50 43  4d 44 5f 4d 41 47 3d 31   ,0.AT*PC MD_MAG=1
0120  31 31 38 35 37 2c 30 2c  30 2c 30 2c 30 2c 30 2c   11857,0, 0,0,0,0,
0130  30 2c 30 0d 41 54 2a 50  43 4d 44 5f 4d 41 47 3d   0,0.AT*P CMD_MAG=
0140  31 31 31 38 35 38 2c 30  2c 30 2c 30 2c 30 2c 30   111858,0 ,0,0,0,0
0150  2c 30 2c 30 0d 41 54 2a  50 43 4d 44 5f 4d 41 47   ,0,0.AT* PCMD_MAG
0160  3d 31 31 31 38 35 39 2c  30 2c 30 2c 30 2c 30 2c   =111859, 0,0,0,0,
0170  30 2c 30 2c 30 0d 41 54  2a 50 43 4d 44 5f 4d 41   0,0,0.AT *PCMD_MA
0180  47 3d 31 31 31 38 36 30  2c 30 2c 30 2c 30 2c 30   G=111860 ,0,0,0,0
0190  2c 30 2c 30 2c 30 0d 41  54 2a 50 43 4d 44 5f 4d   ,0,0,0.A T*PCMD_M
01a0  41 47 3d 31 31 31 38 36  31 2c 30 2c 30 2c 30 2c   AG=11186 1,0,0,0,
01b0  30 2c 30 2c 30 2c 30 0d  41 54 2a 50 43 4d 44 5f   0,0,0,0. AT*PCMD_
01c0  4d 41 47 3d 31 31 31 38  36 32 2c 30 2c 30 2c 30   MAG=1118 62,0,0,0
01d0  2c 30 2c 30 2c 30 2c 30  0d 41 54 2a 50 43 4d 44   ,0,0,0,0 .AT*PCMD
01e0  5f 4d 41 47 3d 31 31 31  38 36 33 2c 30 2c 30 2c   _MAG=111 863,0,0,
01f0  30 2c 30 2c 30 2c 30 2c  30 0d 41 54 2a 50 43 4d   0,0,0,0, 0.AT*PCM
0200  44 5f 4d 41 47 3d 31 31  31 38 36 34 2c 30 2c 30   D_MAG=11 1864,0,0
0210  2c 30 2c 30 2c 30 2c 30  2c 30 0d 41 54 2a 50 43   ,0,0,0,0 ,0.AT*PC
0220  4d 44 5f 4d 41 47 3d 31  31 31 38 36 35 2c 30 2c   MD_MAG=1 11865,0,
0230  30 2c 30 2c 30 2c 30 2c  30 2c 30 0d 41 54 2a 50   0,0,0,0, 0,0.AT*P
0240  43 4d 44 5f 4d 41 47 3d  31 31 31 38 36 36 2c 30   CMD_MAG= 111866,0
0250  2c 30 2c 30 2c 30 2c 30  2c 30 2c 30 0d 41 54 2a   ,0,0,0,0 ,0,0.AT*
0260  50 43 4d 44 5f 4d 41 47  3d 31 31 31 38 36 37 2c   PCMD_MAG =111867,
0270  30 2c 30 2c 30 2c 30 2c  30 2c 30 2c 30 0d 41 54   0,0,0,0, 0,0,0.AT
0280  2a 50 43 4d 44 5f 4d 41  47 3d 31 31 31 38 36 38   *PCMD_MA G=111868
0290  2c 30 2c 30 2c 30 2c 30  2c 30 2c 30 2c 30 0d 41   ,0,0,0,0 ,0,0,0.A
02a0  54 2a 50 43 4d 44 5f 4d  41 47 3d 31 31 31 38 36   T*PCMD_M AG=11186
02b0  39 2c 30 2c 30 2c 30 2c  30 2c 30 2c 30 2c 30 0d   9,0,0,0, 0,0,0,0.
02c0  41 54 2a 50 43 4d 44 5f  4d 41 47 3d 31 31 31 38   AT*PCMD_ MAG=1118
02d0  37 30 2c 30 2c 30 2c 30  2c 30 2c 30 2c 30 2c 30   70,0,0,0 ,0,0,0,0
02e0  0d 41 54 2a 50 43 4d 44  5f 4d 41 47 3d 31 31 31   .AT*PCMD _MAG=111
02f0  38 37 31 2c 30 2c 30 2c  30 2c 30 2c 30 2c 30 2c   871,0,0, 0,0,0,0,
0300  30 0d 41 54 2a 50 43 4d  44 5f 4d 41 47 3d 31 31   0.AT*PCM D_MAG=11
0310  31 38 37 32 2c 30 2c 30  2c 30 2c 30 2c 30 2c 30   1872,0,0 ,0,0,0,0
0320  2c 30 0d 41 54 2a 50 43  4d 44 5f 4d 41 47 3d 31   ,0.AT*PC MD_MAG=1
0330  31 31 38 37 33 2c 30 2c  30 2c 30 2c 30 2c 30 2c   11873,0, 0,0,0,0,
0340  30 2c 30 0d 41 54 2a 50  43 4d 44 5f 4d 41 47 3d   0,0.AT*P CMD_MAG=
0350  31 31 31 38 37 34 2c 30  2c 30 2c 30 2c 30 2c 30   111874,0 ,0,0,0,0
0360  2c 30 2c 30 0d 41 54 2a  50 43 4d 44 5f 4d 41 47   ,0,0.AT* PCMD_MAG
0370  3d 31 31 31 38 37 35 2c  30 2c 30 2c 30 2c 30 2c   =111875, 0,0,0,0,
0380  30 2c 30 2c 30 0d 41 54  2a 50 43 4d 44 5f 4d 41   0,0,0.AT *PCMD_MA
0390  47 3d 31 31 31 38 37 36  2c 30 2c 30 2c 30 2c 30   G=111876 ,0,0,0,0
03a0  2c 30 2c 30 2c 30 0d 41  54 2a 50 43 4d 44 5f 4d   ,0,0,0.A T*PCMD_M
03b0  41 47 3d 31 31 31 38 37  37 2c 30 2c 30 2c 30 2c   AG=11187 7,0,0,0,
03c0  30 2c 30 2c 30 2c 30 0d  41 54 2a 50 43 4d 44 5f   0,0,0,0. AT*PCMD_
03d0  4d 41 47 3d 31 31 31 38  37 38 2c 30 2c 30 2c 30   MAG=1118 78,0,0,0
03e0  2c 30 2c 30 2c 30 2c 30  0d 41 54 2a 50 43 4d 44   ,0,0,0,0 .AT*PCMD
03f0  5f 4d 41 47 3d 31 31 31  38 37 39 2c 30 2c 30 2c   _MAG=111 879,0,0,
0400  30 2c 30 2c 30 2c 30 2c  30 0d 41 54 2a 50 43 4d   0,0,0,0, 0.AT*PCM
0410  44 5f 4d 41 47 3d 31 31  31 38 38 30 2c 30 2c 30   D_MAG=11 1880,0,0
0420  2c 30 2c 30 2c 30 2c 30  2c 30 0d                  ,0,0,0,0 ,0.
```

Figure 2-4 Flight control command packet transmitted from controller to drone

A single complete command requires a minimum of 28 characters depending upon the value following the ‘equal to’ sign in the command. The command is duplicated thirty times and the packet is transmitted to the drone. The first command in the packet is shown below.

AT*PCMD_MAG=111850,0,0,0,0,0,0,0.

The beginning of the command, AT*PCMD_MAG, informs the drone that it is receiving a flight control command. The value following the ‘equals to’ sign is the sequence number which is effectively a command counter and is required for every AT command transmitted to the drone. The sequence number starts from zero for the first command transmitted and is incremented for every subsequent command transmitted. In the flight command shown the sequence number is 111850 indicating that this is the 111850th command that has been transmitted. It is observed in Figure 2-4 that the next command in the packet has a sequence number of 111851. The drone will only execute a command whose sequence number is greater than the sequence number of the command previously executed. The seven values following the sequence number describe the velocity and direction in which the drone should fly. When all seven values are 0, the drone is being instructed to hover. The detail of how these values impact on the drone when they are not zero are described in Table 3-1.

As well as receiving AT flight commands, the drone also transmits flight data, known as navigation data, every 20ms to the flight control platform. The navigation data is grouped into sections of similar type (e.g. v_x velocity in the x direction, v_y velocity in the y direction) that Parrot call options. In total there are twenty one available options that the drone can transmit. The controller however can also request to receive a subset of the available options by transmitting the relevant command to the drone as required. The most relevant option is the so called navdata_demo option which contains the navigation data described in Table 2-3.

Table 2-3 Navdata_demo option transmitted as subset of navigation data

typedef struct _navdata_demo_t {		
uint16_t	tag;	/* Navdata block ('option') identifier */
uint16_t	size;	/*set this to the size of this structure */
uint32_t	ctrl_state;	/*Flying state defined in CTRL_STATES enum */
uint32_t	vbat_flying_percentage;	/*battery voltage (%) */
float32_t	theta;	/*UAV's pitch in milli-degrees */
float32_t	phi;	/*UAV's roll in milli-degrees */
float32_t	psi;	/*UAV's yaw in milli-degrees */
int32_t	altitude;	/*UAV's altitude in millimeters */
float32_t	vx;	/*UAV's estimated linear velocity in x direction */
float32_t	vy;	/*UAV's estimated linear velocity in y direction */
float32_t	vz;	/*UAV's estimated linear velocity in z direction */
}_ATTRIBUTE_PACKED_ navdata_demo_t		

As well as describing the data contained in the navdata_demo option, Table 2-3 also shows the number of bytes required for each value transmitted. The altitude data for example follows the yaw data and consists of a 32 bit, or 4 byte, integer value. The first value transmitted is the 16 bit option tag number. The navdata_demo is always the first data option transmitted and has a tag number of $0000. The second value transmitted is the size in bytes of the data transmitted in the option including the tag and the size values. Potentially useful data then follows including battery percentage, pitch, roll and yaw angles, altitude, and velocity in the x, y, and z directions. Following the last value transmitted, velocity in the z direction v_z, data relating to the camera is also transmitted but is not included in Table 2-3 as it is now deprecated.

The navigation data is transmitted as a structure as defined in Table 2-4. The 32 bit header are the first four bytes of navigation data packet transmitted. At the receiving end the header is checked and once verified ensures the validity of the navigation data that follows.

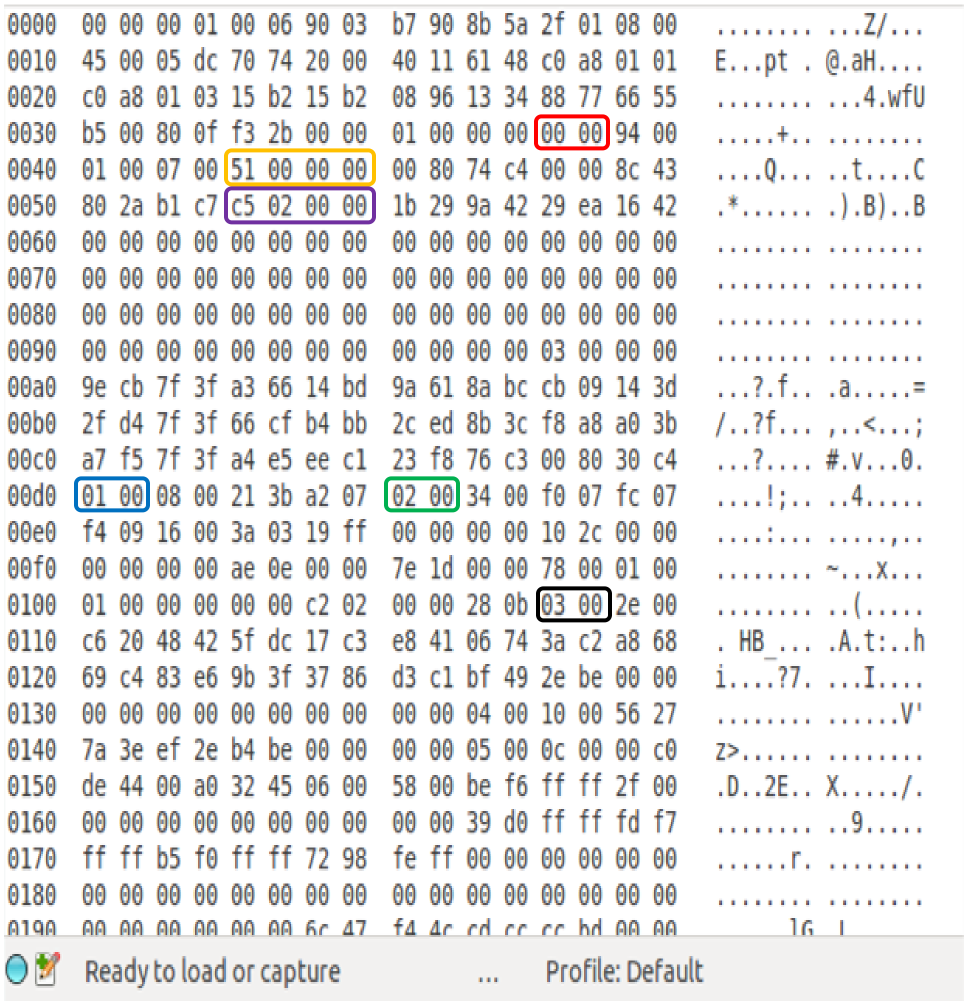
Table 2-4 Navigation data structure definition

typedef struct navdata_t {		
uint32_t	header;	/*Always set to NAVDATA_HEADER */
uint32_t	ardrone_state;	/*Bit mask from def_ardrone_state_mask_t */
uint32_t	sequence;	/*Sequence number incremented for each sent packet */
bool_t	vision_defined;	
navdata_option_t	options[1];	
}_ATTRIBUTE_PACKED_ navdata_t;		

Following the header is a 32 bit ardrone state word. This word contains bits which define the current state of the drone for example bit 12 indicates motor status, bit 15 indicates low battery voltage. Following the 32 bit state word is the 32 bit sequence number. This value defines the number of the current navigation data packet. The value is incremented after each packet transmission and is effectively a running total of the number of navigation data packets that have been transmitted. A 32 bit vision defined value precedes the navigation data options. A subset of the navigation data transmitted by the drone and captured by 'wireshark' is shown in Figure 2-5.

The navigation data header which is defined as $55667788 is observed in Figure 2-5 to begin at location $002c. Following the header is the 32 bit state word which in this case is $0f8000b5. The next value is the 32 bit sequence number $00002bf3 (decimal 11,251), indicating that this is the 11,251th navigation data packet transmitted. The 32 bit vision defined value $00000001 follows before the navigation data options begin at location $003c. The tag values of the first four navigation data options transmitted are highlighted in red, blue, green, and black in Figure 2-5. The two bytes following the tag number is the number of bytes transmitted as part of that option.

Within the navdata_demo option two data values are highlighted to indicate where the battery voltage, and altitude as an example are located. In yellow is the 32 bit vbat_flying_percentage value which is indicating that the battery is currently at 51% of capacity.



```
0000  00 00 00 01 00 06 90 03  b7 90 8b 5a 2f 01 08 00   ........ ...Z/...
0010  45 00 05 dc 70 74 20 00  40 11 61 48 c0 a8 01 01   E...pt . @.aH....
0020  c0 a8 01 03 15 b2 15 b2  08 96 13 34 88 77 66 55   ........ ...4.wfU
0030  b5 00 80 0f f3 2b 00 00  01 00 00 00 00 00 94 00   .....+.. ........
0040  01 00 07 00 51 00 00 00  00 80 74 c4 00 00 8c 43   ....Q... ..t....C
0050  80 2a b1 c7 c5 02 00 00  1b 29 9a 42 29 ea 16 42   .*...... .).B)..B
0060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0090  00 00 00 00 00 00 00 00  00 00 00 00 03 00 00 00   ........ ........
00a0  9e cb 7f 3f a3 66 14 bd  9a 61 8a bc cb 09 14 3d   ...?.f.. .a.....=
00b0  2f d4 7f 3f 66 cf b4 bb  2c ed 8b 3c f8 a8 a0 3b   /..?f... ,..<...;
00c0  a7 f5 7f 3f a4 e5 ee c1  23 f8 76 c3 00 80 30 c4   ...?.... #.v...0.
00d0  01 00 08 00 21 3b a2 07  02 00 34 00 f0 07 fc 07   ....!;.. ..4.....
00e0  f4 09 16 00 3a 03 19 ff  00 00 00 00 10 2c 00 00   ....:... .....,..
00f0  00 00 00 00 ae 0e 00 00  7e 1d 00 00 78 00 01 00   ........ ~...x...
0100  01 00 00 00 00 00 c2 02  00 00 28 0b 03 00 2e 00   ........ ..(.....
0110  c6 20 48 42 5f dc 17 c3  e8 41 06 74 3a c2 a8 68   . HB_... .A.t:..h
0120  69 c4 83 e6 9b 3f 37 86  d3 c1 bf 49 2e be 00 00   i....?7. ...I....
0130  00 00 00 00 00 00 00 00  00 00 04 00 10 00 56 27   ........ ......V'
0140  7a 3e ef 2e b4 be 00 00  00 00 05 00 0c 00 00 c0   z>...... ........
0150  de 44 00 a0 32 45 06 00  58 00 be f6 ff ff 2f 00   .D..2E.. X...../.
0160  00 00 00 00 00 00 00 00  00 00 39 d0 ff ff fd f7   ........ ..9.....
0170  ff ff b5 f0 ff ff 72 98  fe ff 00 00 00 00 00 00   ......r. ........
0180  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0190  00 00 00 00 00 00 6c 47  f4 4c cd cc cc bd 00 00         lG  L
```

Ready to load or capture ... Profile: Default

Figure 2-5 Subset of navigation data captured by wireshark

The 32 bit values of pitch, roll and yaw angles follow, and then the 32 bit altitude value, \$000002c5, highlighted in purple. Converting to decimal, the altitude data informs the controlling consul that the drone is flying at an altitude of 709mm. The titles of the four navigation data options highlighted are as shown in Table 2-5.

Table 2-5 Examples of navigation data options

navdata_demo (red),	tag number 0000, consists of 148 (\$0094) bytes,
time (blue),	tag number 0001, consists of 8 (\$0008) bytes,
raw_measures (green)	tag number 0002 consists of 52 (\$0034) bytes,
magneto (black),	tag number 0003 consists of 46 (\$002e) bytes.

2.7 Summary

Initially this chapter discusses the physics behind the quadcopter drone. Formulae relating to the thrust required by the four rotors to both elevate the drone, and move the drone forwards, backwards, or laterally, are derived. The sensors present within drones to enable stable and controlled flight are discussed. The varying number of applications to which drones are now being applied are presented and discussed. Finally a detailed analysis of the flight command packet transmitted, and the navigation data received from the Parrott AR2 drone are presented. This information will be used extensively in the next chapter in the development of an autonomous quadcopter drone.

Chapter 3

Autonomous Quadcopter Drone Control

3.1 Introduction

This chapter describes the development of the hardware platform and control algorithms to enable autonomous UAV 3-dimensional flight control. The Parrot AR2 drone is designed to be controlled by a tablet or smartphone, however, autonomous flight control requires a computer or microcontroller which can execute a flight program stored in memory. Since the drone's on board microcontroller is not accessible, a laptop solution, and a solution incorporating Wi-Fi modules with on board microcontrollers, are used to meet the hardware platform requirements of the autonomous drone. Programs are developed to run on the hardware platforms to control the flight of the drone and enable the drone to follow any desired flight plan. The hardware platforms and flight control programs are evaluated experimentally and compared with expected results.

3.2 Autonomous UAV Hardware Platform

An autonomous UAV requires a hardware platform which is capable of executing a stored flight program in its memory and be able to communicate commands to the drone (achieved via Wi-Fi in this case). Two hardware platforms, (a) and (b) below, are implemented and are depicted in Figure 3-1.

(a) Control via laptop
(b) Control via NodeMCU Wi-Fi module

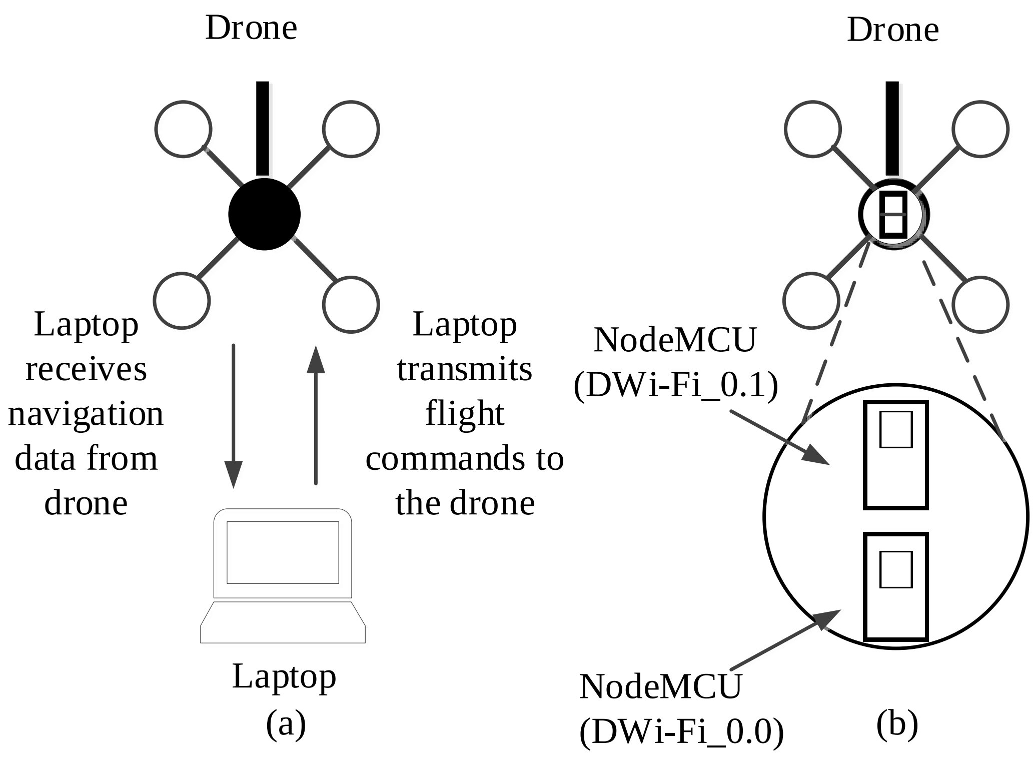

Drone
Drone
Laptop receives navigation data from drone
Laptop transmits flight commands to the drone
NodeMCU (DWi-Fi_0.1)
Laptop
(a)
NodeMCU (DWi-Fi_0.0)
(b)

Figure 3-1(a) Drone flight controlled by laptop (b) Drone flight controlled by NodeMCU module (DWi-Fi_0.1)

Control via laptop requires the installation of the software development kit (SDK), provided by the drone manufacturer, Parrot. The SDK, whilst enabling user flight control programs to be constructed and executed, also handles drone Wi-Fi connection, navigation data reception, and flight command transmission to the drone, as depicted in Figure 3-1a.

In Figure 3-1b above the NodeMCU module effectively replaces the laptop. The NodeMCU is an IoT platform which incorporates an ESP8266 Wi-Fi module, and a microcontroller with 128k of application program memory. The microcontroller has thirty accessible pins enabling I/O (input/output), serial data communications, PWM pulse width modulation (PWM) output, analogue to digital converter (ADC) inputs, and is readily programmed utilizing the Arduino IDE (integrated development environment). The device is 4.8cm long by 2.5cm wide and weighs 10g, which enables it to be mounted on the drone without compromising the drone during flight.

In order for the drone to fly a controlled flight plan in 3-dimensional space, commands must be received by the drone every 20ms, otherwise the drone drifts in a random and uncontrollable manner. In a network environment of potentially multiple networked drones, searching and connecting to appropriate Wi-Fi networks can take in excess of one second. A single NodeMCU module is therefore unable to control the flight of the drone and handle network requirements. A second NodeMCU module is thus incorporated into the hardware configuration to enable the possible construction of a drone network of similarly configured drones. The flight control NodeMCU module is labelled DWi-Fi_x.1 and the network module is labelled DWi-Fi_x.0, where x is the drone number. In the following description the drone is numbered as drone 0.

The function of the two NodeMCU modules mounted on drone 0 is described below.

DWi-Fi_0.1:

(i) Connects to the drone Wi-Fi network to which it is attached.

(ii) Communicates flight control commands to the drone.

(iii) Reads navigation data from the drone.

(iv) Transmits navigation data via serial port connections to DWi-Fi_0.0.

(v) Receives flight control codes from DWi-Fi_0.0 and responds accordingly.

DWi-Fi_0.0

(i) Enables drone network capability by connecting to DWi-Fi_0.0 modules to other DWi-Fi_x.0 modules of similarly configured drones.

(ii) Transmits flight control codes across the network.

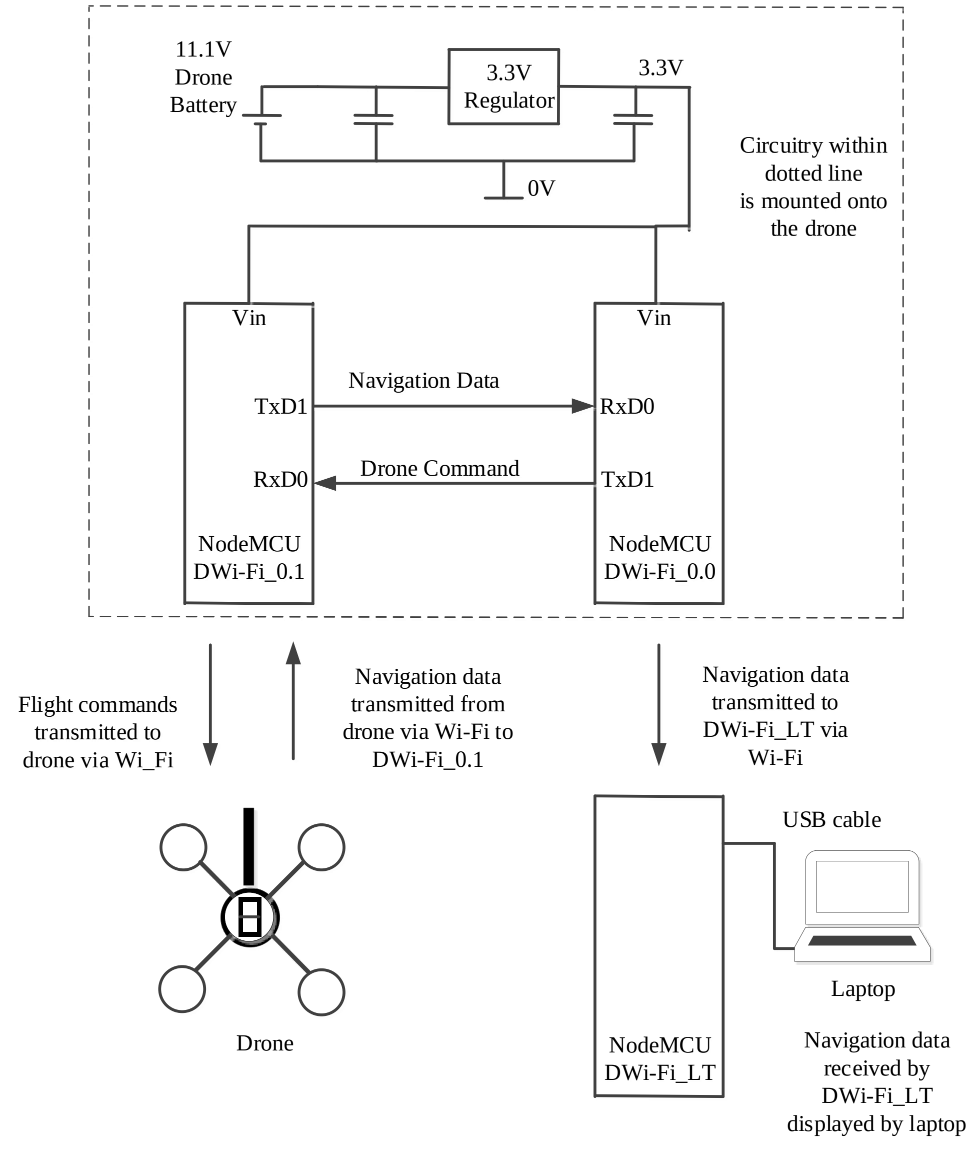
(iii) Receives flight control codes that have been transmitted across the network.

(iv) Communicates flight control codes to DWi-Fi_0.1 via serial port.

(v) Receives navigation data from DWi-Fi_0.1.

(vi) Transmits received navigation data from DWi-Fi_0.1 across the network.

A 3.3V voltage regulator circuit also mounted on the drone receives power from the drone’s 11.1V lithium battery and provides power to both NodeMCU modules. The hardware circuit for the NodeMCU controlled drone with network capability, is shown in Figure 3-2. A third NodeMCU module, DWi-Fi_LT, captures navigation data transmitted from DWi-Fi_0.0. The navigation data is originally transmitted by the drone to DWi-Fi_0.1 via Wi-Fi, which then transmits to DWi-Fi_0.0 via the serial port connection.



11.1V Drone Battery
3.3V Regulator
3.3V
0V
Circuitry within dotted line is mounted onto the drone
Vin
Vin
Navigation Data
TxD1
RxD0
Drone Command
RxD0
TxD1
NodeMCU DWi-Fi_0.1
NodeMCU DWi-Fi_0.0
Flight commands transmitted to drone via Wi_Fi
Navigation data transmitted from drone via Wi-Fi to DWi-Fi_0.1
Navigation data transmitted to DWi-Fi_LT via Wi-Fi
USB cable
Laptop
Drone
NodeMCU DWi-Fi_LT
Navigation data received by DWi-Fi_LT displayed by laptop

Figure 3-2 Circuit diagram of NodeMCU controlled autonomous drone.

3.3 Autonomous UAV Flight Control Algorithm

Autonomous UAV flight control in 3-dimensional space via a laptop, utilises the SDK provided by the drone manufacturer Parrot, to facilitate the construction of flight control programs. Two functions, made available within the SDK, can be called from the flight control program, to i) enable drone take off and land, and ii) autonomous drone flight control. The two SDK functions are shown below.

i) Take-off and land function (TOAL)
ardrone_tool_set_ui_pad_start(t/l);

ii) Autonomous drone flight control function (ADFC)
ardrone_at_set_progress_cmd_with_magneto(a, b, c, d, e, f, g);

The argument functionality table of Table 3-1 describes how each argument in the above functions impacts on the drone.

Table 3-1 SDK argument functionality table

Function	Argument	Value	Function
TOAL	t/l	1	Take-off
TOAL	t/l	0	Land
ADFC	a	1	Enables combined yaw mode
ADFC	a	0	Enables hover mode
ADFC	b	-1.0 max to +1.0 max	Roll left/Roll right
ADFC	c	-1.0 max to +1.0 max	Pitch forward/Pitch backward
ADFC	d	-1.0 max to +1.0 max	Ascend/Descend
ADFC	e	-1.0 max to +1.0 max	Anticlockwise/Anticlockwise rotation
ADFC	f	0	Drone points to North after calibration
ADFC	g		Magnetometer accuracy

The SDK translates functions TOAL and ADFC into the AT commands that are transmitted to the drone as described in Section 2.6.1. An example of an AT command to fly the drone in the forward direction is shown below.

AT*PCMD_MAG= seq_num,1,0,-1110651699,0,0,0,0;

In the above example, the negative value of argument c indicates that the drone should fly in the forward direction. The value of argument c is the IEEE 754 equivalent of -0.1 and indicates that the drone should fly in the forward direction at the corresponding velocity. The SDK builds a packet of thirty AT commands, and transmits the whole packet to the drone every 20ms.

For autonomous 3-dimensional flight control via the NodeMCU Wi-Fi module the AT commands described above are constructed directly. The program running in the NodeMCU module mimics the SDK by building the required thirty AT command packet before transmitting complete packets to the drone every 20ms.

The seven flight control functions for both laptop and NodeMCU platforms, are depicted in Table 3-2. By combining the available functions in the required sequence, any desired flight plan can be realised.

Table 3-2 Flight control functions

Flight Function	**argument a**	**argument b**
take_off()	-	-
land()	-	-
hover(a)	Time to hover (s)	-
forward_backward(a,b)	Distance to travel (mm)	Velocity (-0.1 to 0.1)
roll_left_roll_right(a,b)	Distance to travel (mm)	Velocity (-0.1 to 0.1)
ascend_descend(a,b)	Height required (mm)	Velocity (-0.1 to 0.1)
rotate_clock_anticlock(a,b)	Number of degrees (mm)	Velocity (-0.1 to 0.1)

The seven flight control functions operate completely independently from each other. The execution of any function in a desired flight plan is completed before the execution of the next function in the flight plan can begin. The current functions therefore cannot instruct the drone to fly a combination of flight instructions, e.g. to fly forward and ascend simultaneously for example. Additional functions can be developed, however, to permit simultaneous flight instructions to be completed by the drone. For example, to simultaneously fly forward and ascend, the function call would require four arguments; (w) a velocity argument to ascend and (x) the height the drone should ascend to, and (y) a velocity argument for the forward flight and (z) the distance the drone should fly in the forward direction. Within the function to fly forwards, backwards, ascend and descend, the velocity value to ascend would be inserted into position d, and the velocity value to fly forwards would be inserted into position c, of the flight control function described in Table 3-1. The flight control function would be executed continuously until the required height, and the required distance flown in the forward direction have been realised. A possible function call to fly forward and ascend simultaneously is described below.

```
forward_backward_ascend_descend(w, x, y, z);
```

In a similar fashion additional functions could be written to combine drone flight movement, e.g. to fly diagonally, if required.

3.3.1 Drone Distance Flown Calculation

The drone has no means of determining the distance it is required to fly, as specified in argument ‘b’ in the forward_backward(a,b) and roll_left_roll_right(a,b) functions. The distance travelled is therefore calculated in the program by effective integration of received navigation data, v_x velocity of flight in the x direction (forwards or backwards), and v_y velocity of flight in the y direction (lateral movement). The calculation of the distance travelled in the x direction is described in Figure 3-3.

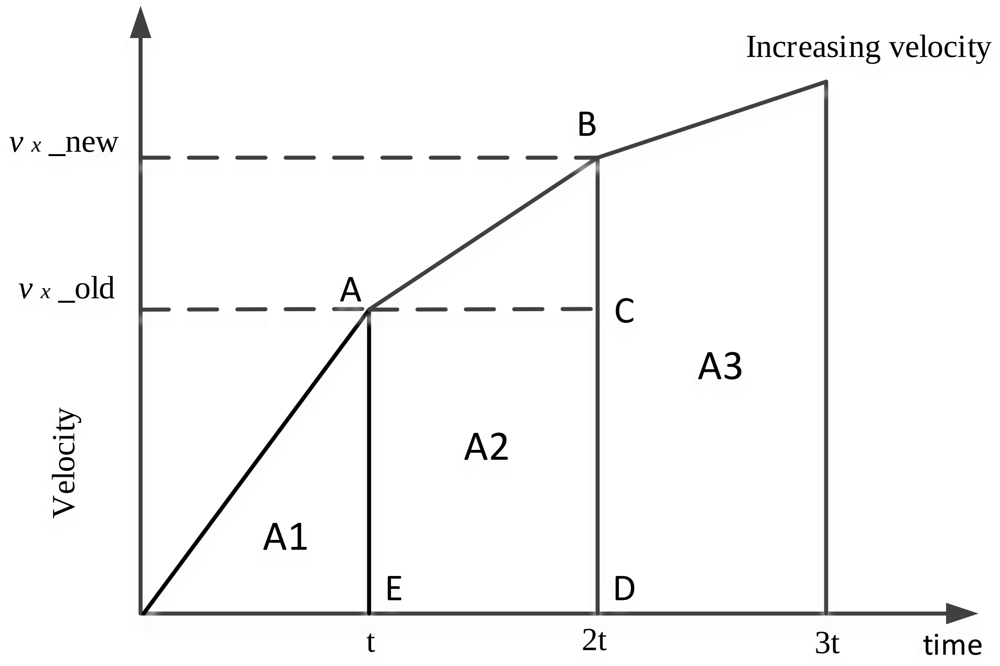


Figure 3-3 Diagram illustrating distance calculation method

The velocity v_x_old in Figure 3-3 is the drone velocity received and stored from the previous navigation data received at time t. The time intervals t, $2t$, $3t$, are the times at which navigation data is received. The velocity v_x_new is the current drone velocity received at time $2t$. In the diagram of Figure 3-3, the distance travelled between time t and $2t$ in the x direction, *distance_x*, is, calculated from the area of the trapezoid A2 (ABDE) in Figure 3-3. The area of trapezoid A2 is given by the sum of the areas of the triangle ABC, and the rectangle ACDE.

$$\text{Area of triangle ABC} = 0.5(v_x\_new - v_x\_old)(2t - t) \qquad \text{(Equation 3-1)}$$

$$\text{Area of rectangle ACDE} = (v_x\_old)(2t - t) \qquad \text{(Equation 3-2)}$$

If the difference in the velocity v_{diff} is given by:

$$v_{diff} = v_x\_new - v_x\_old \qquad \text{(Equation 3-3)}$$

Then by substituting into Equation 3-1, the area of triangle ABC is given by:

$$\text{Area of ABC} = 0.5v_{diff}(2t - t) \qquad \text{(Equation 3-4)}$$

The area of trapezoid ABDE, equivalent to the distance travelled in the x direction, *distance_x,* is therefore given by:

$$\text{Area of trapezoid ABDE} = (0.5v_{diff}(2t - t)) + ((v_x\_old)(2t - t)) \qquad \text{(Equation 3-5)}$$

Similarly, distances travelled in the y direction (*distance_y),* can be calculated from v_y_old, v_y_new and *t*. The drone position in terms of *x,y* co-ordinates are thus calculated for every increment of time *t.* These co-ordinates are saved and enable a complete *x y* plot of the drone flight plan to be drawn at the end of the flight. The total distance travelled in the *x* direction and the *y* direction are calculated from the sum of the *distance_x* values and the *distance_y* values respectively.

The accuracy of the calculated distances depend upon the accuracy of the velocity data supplied by the drone, however the accuracy of this velocity data varies depending upon the terrain the drone is flying over. The manufacturing company Parrot, suggest that for the most accurate velocity values the drone should be flown over a uniform patterned floor such as square tiles. The accuracy of the distances calculated in this thesis can vary between 91% and 99% when compared with the actual measured flight distance. The calculated flight distance can therefore only be considered as an estimate of the actual distance flown.

3.4 Drone Flight Control Program Development

The autonomous drone flight control functions were developed utilising the SDK, to run initially on the laptop, and then modified to be made available to the Node MCU Wi-Fi modules. The SDK provides a number of programs written as threads to assist autonomous flight control program development. The essential threads are the ardrone_control thread which generates the flight control packet illustrated in Figure 2-4 and transmits the packet to the drone, and the navdata_update thread, which reads the navigation data transmitted by the drone every 20ms. Since there was potential for the navigation data to assist with flight control, the initial functions were constructed within the navdata_update thread. A function to fly the drone in the forward direction was constructed which simply executed the autonomous flight control function described in Table 3-1 with the value -0.1 in the c argument position. The function was executed within a control loop to ensure it was executed a number of times. Upon execution the drone was observed to take off and fly in the forward direction until the loop counter in the control loop decremented to zero. Since there was no flight distance control, care was taken to ensure that the number in the control loop was of a size to ensure that the drone did not crash into the surrounding walls of the gym in which the tests were carried out.

It would be expected that the velocity of the drone would be constant once the terminal velocity of the drone is reached, however upon closer inspection, this was found not to be the case. During the flight, the drone did not maintain its pitch angle, and was observed to decelerate, then pitch forward and accelerate again several times during the forward flight. The reason for this was discovered after closer inspection of the navdata_update function. This function should execute every 20ms, however on occasion it was observed to skip, and execute after 40ms or even 60ms. Commands should be transmitted to the drone every 20ms to ensure correct uninterrupted flight, however, if the navdate_update function where the flight control program is located, does not execute for 60ms, then this will not be the case. This was the cause of the drone being interrupted during the time when it was expected to fly in the forward direction at a constant velocity. Quite why the navdata_update function, on occasion, skips execution, could not be ascertained.

To overcome this undesirable situation a completely separate flight control thread entitled ‘fly’ was constructed. The thread contains a number of flight control functions which when executed informs the drone of the desired flight. The three main flight control functions to instruct the drone to fly forwards or backwards, roll left or roll right, and ascend or descend were constructed in a similar fashion.

All three functions contain two arguments. One argument informs the drone of the velocity at which it should fly and the second argument informs the drone the distance it should fly. For the fly forward/backward function and the roll left/roll right function the distance the drone flies is calculated during the flight using the method described in Figure 3-3 and Equation 3-5. For the ascend/descend function the distance (height) the drone should fly is provided by the drone in the ‘height’ navigation data. The fly forward or backward program function is shown in Figure 3-4 below.

```
void forward_backward(int n, float m)
{
extern float distance_new;
extern int distance_flag;
extern float distance;

while (distance_new < =n)
    {
    if (distance_flag != 1)
            {
            ardrone_at_set_progress_cmd_with_magneto(1, 0, m, 0, 0, 0, 0);
            }


    else
            {
            distance_new = distance_new + distance;
            distance_flag = 0;
            ardrone_at_set_progress_cmd_with_magneto(1, 0, m, 0, 0, 0, 0);
            }
    }
distance_new = 0;
printf("man finished\n");
}
```

Figure 3-4 Program illustrating the fly forward or backward structure

The function of Figure 3-4 is executed by a function call, for example forward_backward(15000,-0.1). The arguments passed to the function, 15000, (n, in the forward_backward function) instructs the drone to fly 15000mm, and -0.1, (m, in the forward_backward function) instructs the drone to fly at the velocity corresponding to this value. The variables distance_new, distance_flag, and distance are declared in the navdata_update function where the distance flown by the drone is calculated. The variables are declared as extern in the forward_backward function to make theses variables calculated in navdata_update available to the forward_backward function. The distance_new variable is the calculated distance the drone has flown and is initially cleared to zero in the navdata_update function.

While the distance required for the drone to fly (n), is less than distance_new (initially zero) the main section of the function is executed. To ensure that the distance travelled (distance_new) is only updated after the navdate_update has been executed and the distance travelled (distance) has therefore been calculated, a flag system is introduced. A flag, 'distance_flag', is set to 1 at the end of navdata_update which is examined at the beginning of the forward_backward function. If the flag is cleared to '0', the distance travelled will not be updated, and the command to fly the drone forward as described in Section 3.3 is executed with the argument -0.1 inserted into the correct argument. If the flag is set to '1', the distance travelled is updated, the distance_flag is cleared to zero and the command to fly the drone forwards is executed as before. Once the calculated distance travelled is equal to or greater than the distance required specified in the original argument, the distance_new variable is cleared to zero and the function is terminated. The functions to roll left or roll right and to ascend or descend are constructed in a very similar manner to the description above. The three functions, when combined in the required order, enables the drone to be instructed to fly to any desired flight plan in 3-dimensional space. Once fully tested, the programs were transposed into programs and uploaded to the NodeMCU module, thus establishing a fully self-contained, autonomous flying drone.

3.5 Autonomous UAV Flight Plan Control

In the flight plan of Figure 3-5, the drone is required to take off and hover for 9s (red functions), climb to 1.5 m (blue function), fly forward in the x direction for 12m (yellow function), hover for 3s, and finally land (green function).

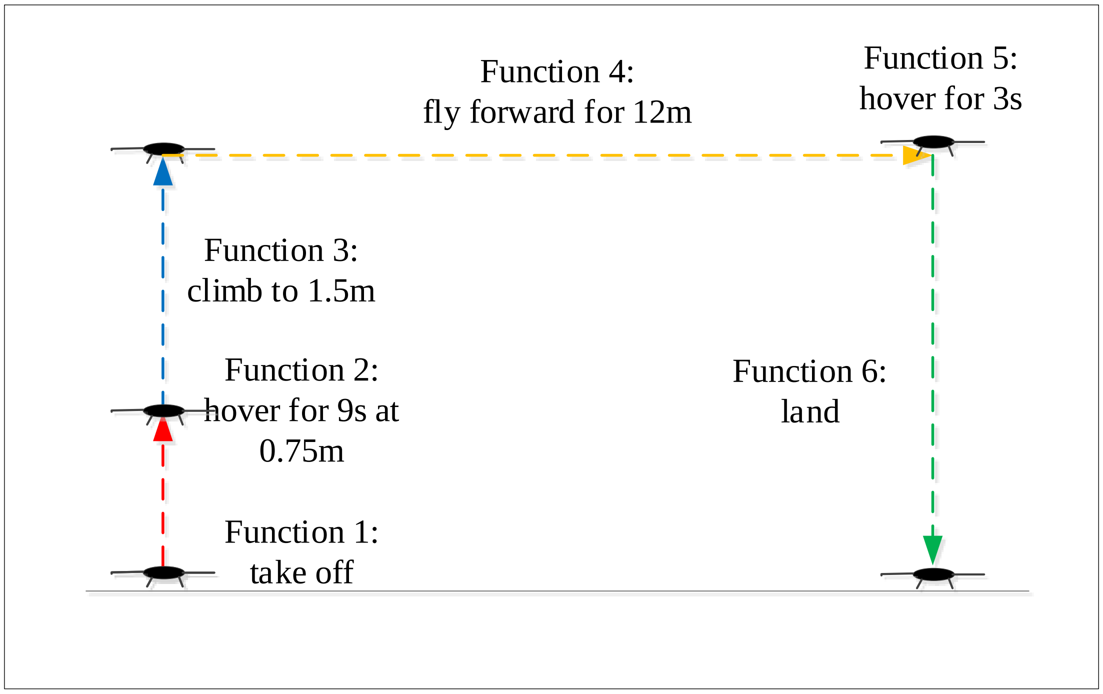


Figure 3-5 Diagram showing flight plan to take off, hover, climb, fly forward, hover and land

The flowchart and associated program required to achieve the flight plan of Figure 3-5 is depicted in Figure 3-6.

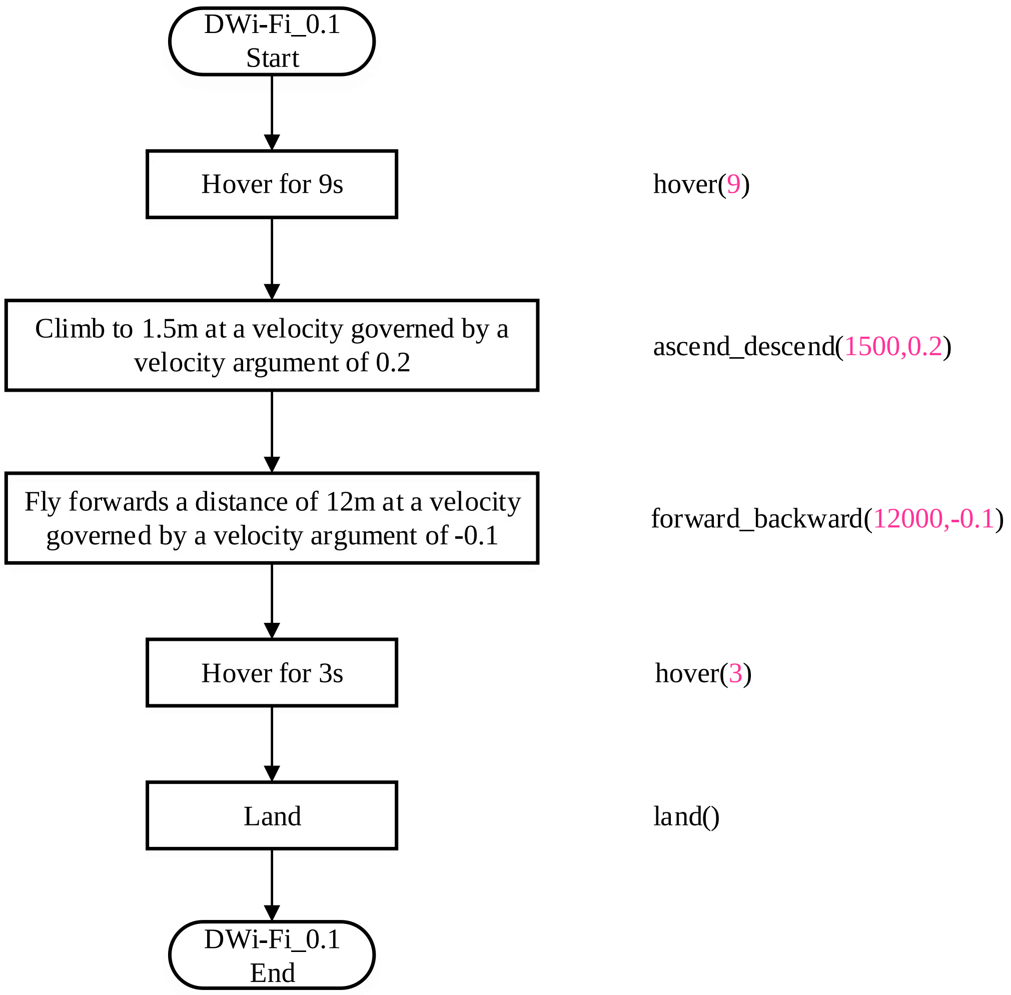

DWi-Fi_0.1
Start
Hover for 9s
hover(9)
Climb to 1.5m at a velocity governed by a velocity argument of 0.2
ascend_descend(1500,0.2)
Fly forwards a distance of 12m at a velocity governed by a velocity argument of -0.1
forward_backward(12000,-0.1)
Hover for 3s
hover(3)
Land
land()
DWi-Fi_0.1
End

Figure 3-6 Flowchart and associated program required to achieve the flight plan of Figure 3-5

The flowcharts in Figure 3-7 to Figure 3-12 describe the functions of Table 3-2 required to achieve the flight plan of Figure 3-5.

The take_off and hover functions (Figure 3-7 and Figure 3-8)

The take-off and hover functions are executed for every desired flight plan. The command that instructs the drone to take-off in Figure 3-7 is repeated for 2 seconds to ensure that the drone responds. The hover command in Figure 3-8 is repeated for 9 seconds to ensure the drone has completed the take-off phase, is stable in flight, and is ready to accept further flight commands.

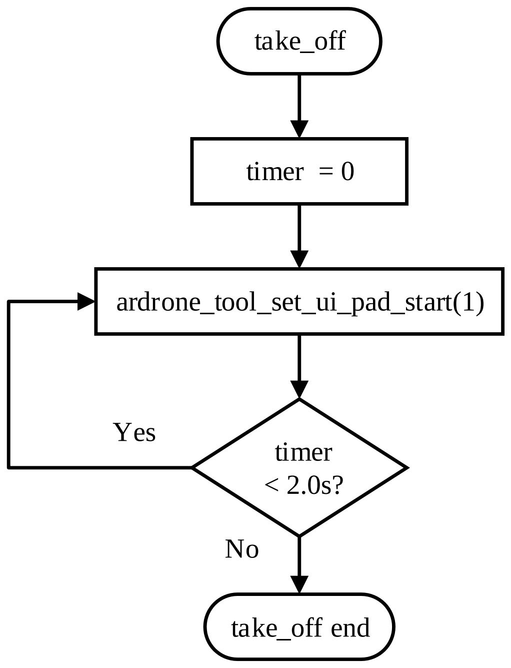


Figure 3-7 Flowchart describing take-off algorithm

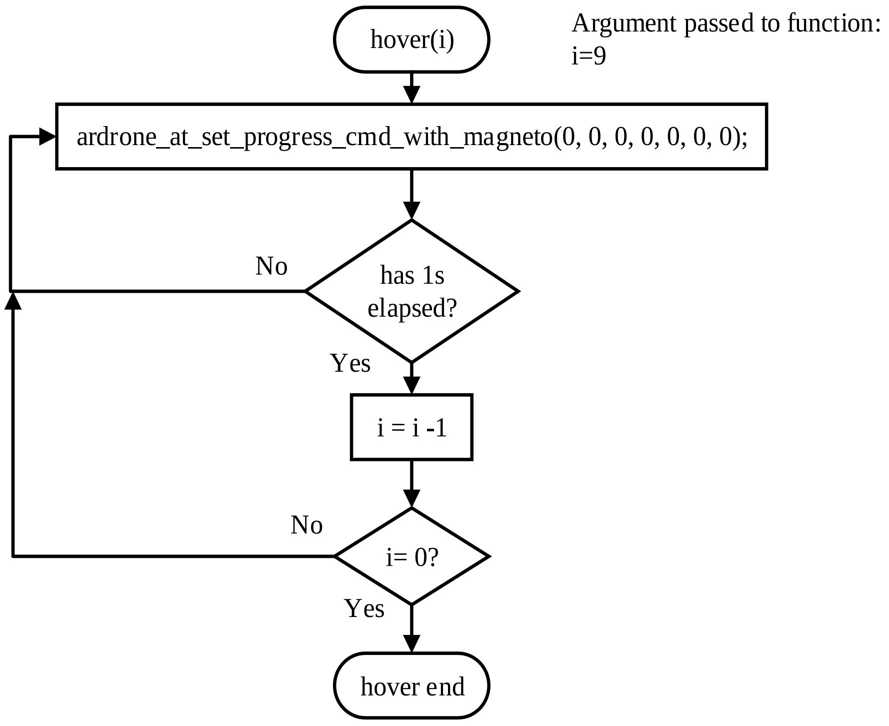

hover(i)
Argument passed to function: i=9
ardrone_at_set_progress_cmd_with_magneto(0, 0, 0, 0, 0, 0, 0);
No
has 1s elapsed?
Yes
i = i -1
No
i= 0?
Yes
hover end

Figure 3-8 Flowchart describing 9 second hover algorithm

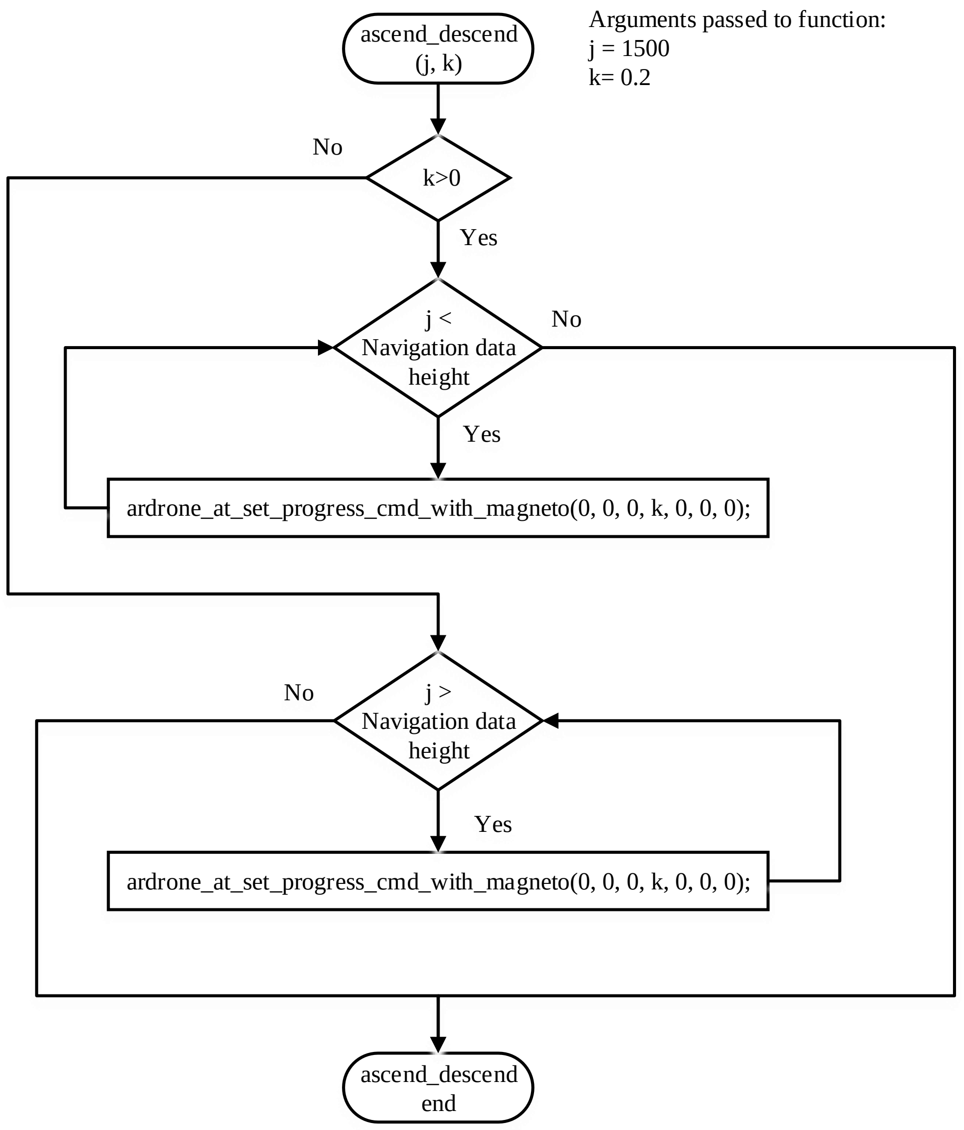

ascend_descend (j, k)
Arguments passed to function:
j = 1500
k= 0.2
No
k>0
Yes
j < Navigation data height
No
Yes
ardrone_at_set_progress_cmd_with_magneto(0, 0, 0, k, 0, 0, 0);
No
j > Navigation data height
Yes
ardrone_at_set_progress_cmd_with_magneto(0, 0, 0, k, 0, 0, 0);
ascend_descend end

Figure 3-9 Flowchart describing algorithm to ascend to 1200mm

The ascend_descend function (Figure 3-9)

The arguments passed to the function (1500,0.2) inform the height to which the drone should ascend (1500mm) and the velocity of ascent (the velocity corresponding to 0.2). The function first examines the polarity of the velocity argument. A positive value, 0.2 in this case, instructs the drone to ascend. The height of the drone, provided by the navigation data, is compared continuously with the required height (1500mm) and the autonomous flight control function causing the drone to ascend continuously executed until the required height is reached when the function terminates.

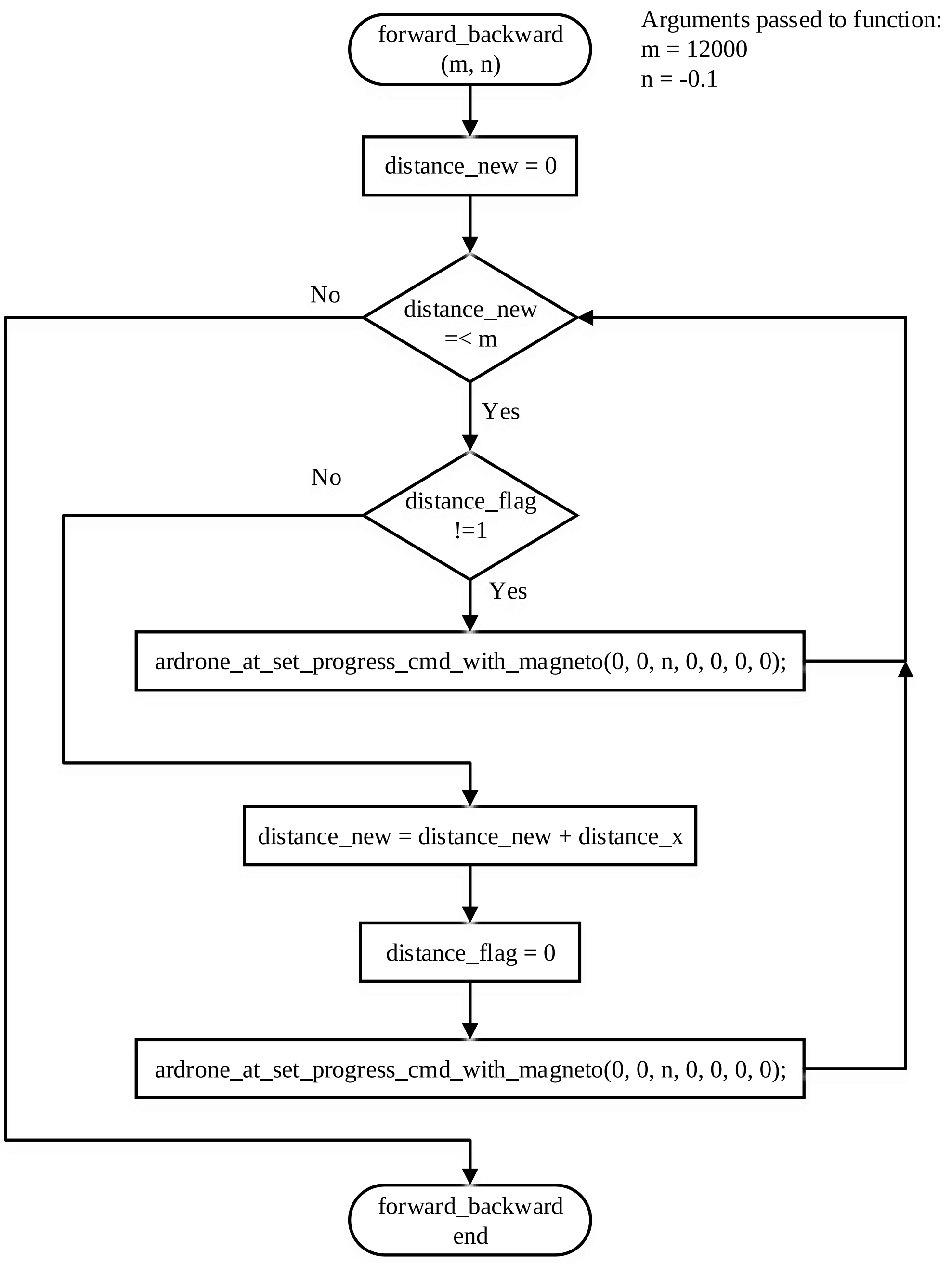

forward_backward (m, n)
Arguments passed to function:
m = 12000
n = -0.1
distance_new = 0
distance_new =< m
No
Yes
distance_flag !=1
No
Yes
ardrone_at_set_progress_cmd_with_magneto(0, 0, n, 0, 0, 0, 0);
distance_new = distance_new + distance_x
distance_flag = 0
ardrone_at_set_progress_cmd_with_magneto(0, 0, n, 0, 0, 0, 0);
forward_backward end

Figure 3-10 Flowchart describing fly forward 12m algorithm

The forward_backward function (Figure 3-10)

The arguments passed to the function (12000,-0.1), determine the distance of travel, 12m, the direction of travel (negative velocity argument indicates forwards) and the drone velocity. The distance_new variable keeps a running total of the distance travelled in the x direction and is initially cleared to zero. While distance_new is less than or equal to

12m, the distance_flag is examined to confirm whether it is currently 0 or 1. The distance_flag is declared in the routine that reads the navigation data and is also made available within the forward_backward function. The distance_flag is set to 1 at the end of the routine that reads the navigation data and informs the forward_backward function that new navigation is available and distance_new can be updated. The distance flag is cleared after distance new has been updated. The distance flag ensures that distance_new is only updated when new navigation data is available. Irrespective of the value of the distance_flag the autonomous flight control function is executed to fly the drone forward. When distance_new is equal to or greater than 12m the function terminates.

The hover function (brake) (Figure 3-11)

Upon completion of the previous function, the hover function is executed for 3 seconds which effectively acts as a brake. When flying in the forward direction the drone is pitched forwards. When the hover function is executed the drone pitches backwards effecting a braking action. The drone then hovers until the 3 seconds has elapsed.

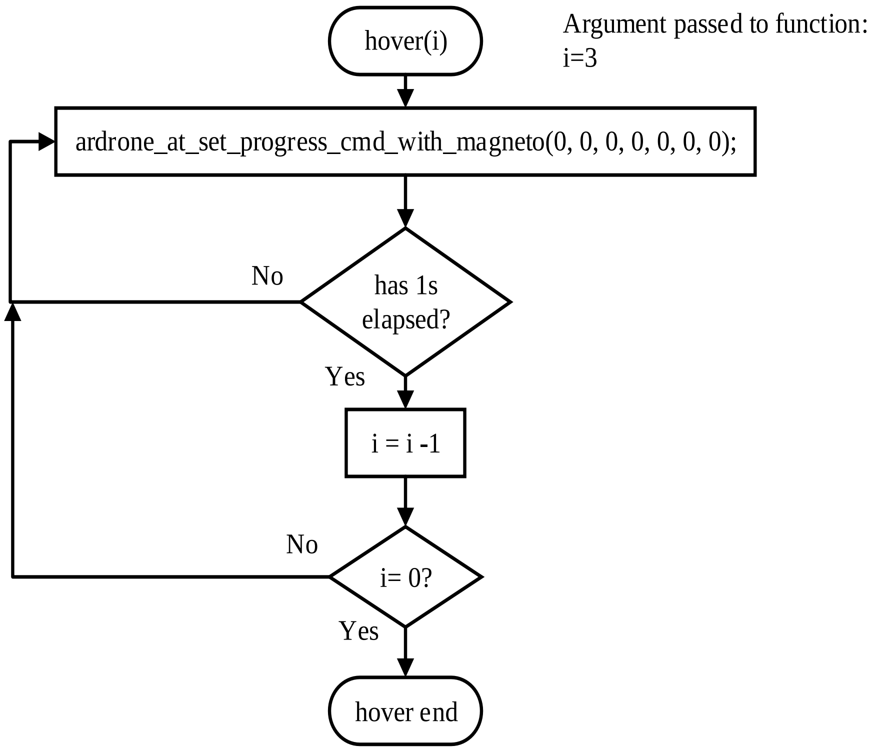

hover(i)
Argument passed to function:
i=3
ardrone_at_set_progress_cmd_with_magneto(0, 0, 0, 0, 0, 0, 0);
No
has 1s elapsed?
Yes
i = i -1
No
i= 0?
Yes
hover end

Figure 3-11 Flowchart describing 3s hover (brake) algorithm

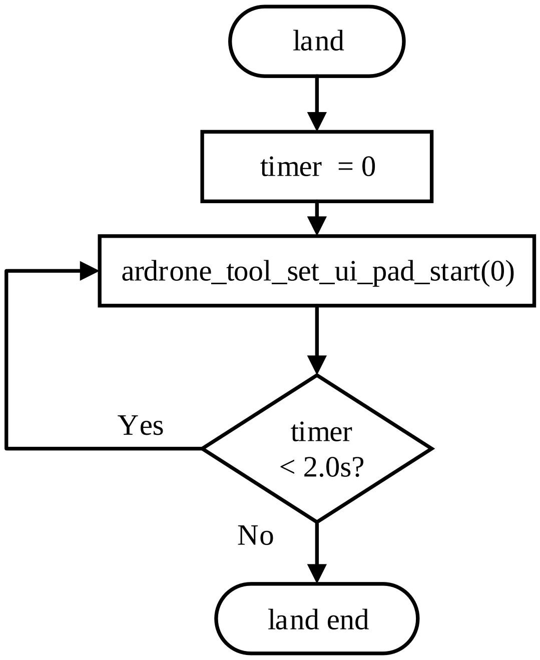


Figure 3-12 Flowchart describing land algorithm

The land function (Figure 3-12)

The land function is executed at the end of every flight plan. The command that instructs the drone to land is executed continuously for 2 seconds to ensure that the drone responds to the instruction and lands.

Under laptop control the program to deliver the flight plan of Figure 3-5 can be executed directly from within the SDK. Under NodeMCU control, the algorithms described in the flowcharts of Figure 3-13, Figure 3-14 and Figure 3-15 are uploaded to NodeMCU modules DWi-Fi_0.1, DWi-Fi_0.0, and DWi-Fi_LT respectively. The interaction during execution of the three programs is as follows:

(i) DWi-Fi_0.1 configures as a station and connects to drone.
DWi-Fi_LT configures as an access point.

(ii) DWi-Fi_0.1 informs DWi-Fi_0.0 that it is connected to the drone.

(iii) DWi-Fi_0.0 reads DWi-Fi_0.1's connection confirmation message and configures itself as a station.

(iv) DWi-Fi_0.0 connects to DWi-Fi_LT and transmits take off code 'TO' to DWi-Fi_0.1.

(v) DWi-Fi_0.1 reads take off code and executes take off function.

(vi) DWi-Fi_0.1 executes next function in flight plan list.

(vii) If function in (vi) is not complete, DWi-Fi_0.1 delays 20ms and transmits navigation data to DWi-Fi_0.0.

(viii) DWi-Fi_0.0 transmits navigation data to DWi-Fi_LT.

(ix) DWi-Fi_LT displays navigation data on laptop screen.

(x) Repeat (vii) to (ix) until execution of current function is complete.

(xi) Repeat (vi) to (x) until last function in flight plan is executed and DWi-Fi_0.1 transmits 'goodbye" to DWi-Fi_0.0.

(xii) DWi-Fi_0.0 transmits 'goodbye" to DWi-Fi_LT.

(xiii) DWi-Fi_LT transmits 'goodbye" to laptop and is displayed on the screen.

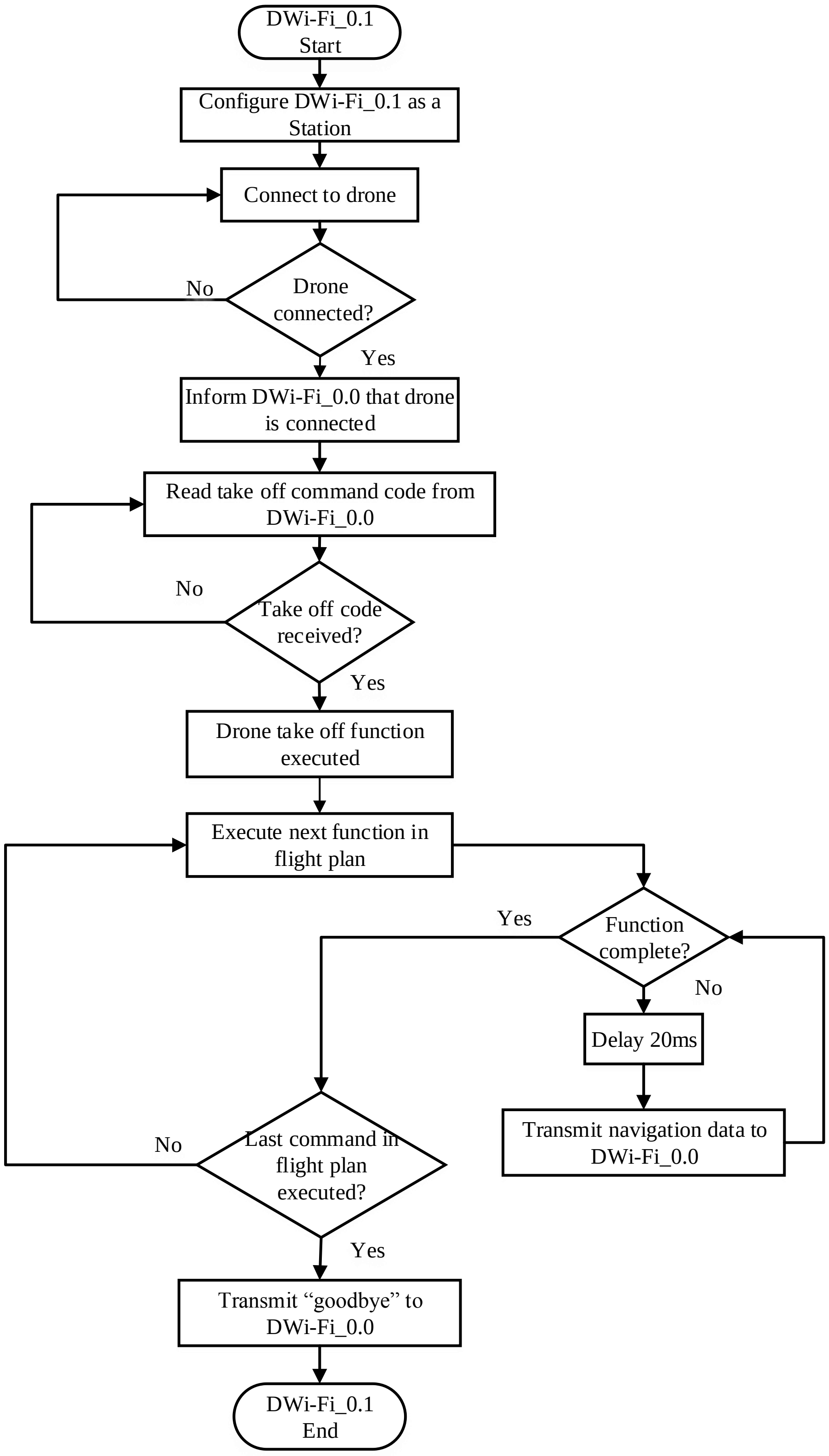

DWi-Fi_0.1 Start
Configure DWi-Fi_0.1 as a Station
Connect to drone
Drone connected?
No
Yes
Inform DWi-Fi_0.0 that drone is connected
Read take off command code from DWi-Fi_0.0
Take off code received?
No
Yes
Drone take off function executed
Execute next function in flight plan
Function complete?
Yes
No
Delay 20ms
Transmit navigation data to DWi-Fi_0.0
Last command in flight plan executed?
No
Yes
Transmit “goodbye” to DWi-Fi_0.0
DWi-Fi_0.1 End

Figure 3-13 Flowchart describing module DWi-Fi_0.1 algorithm

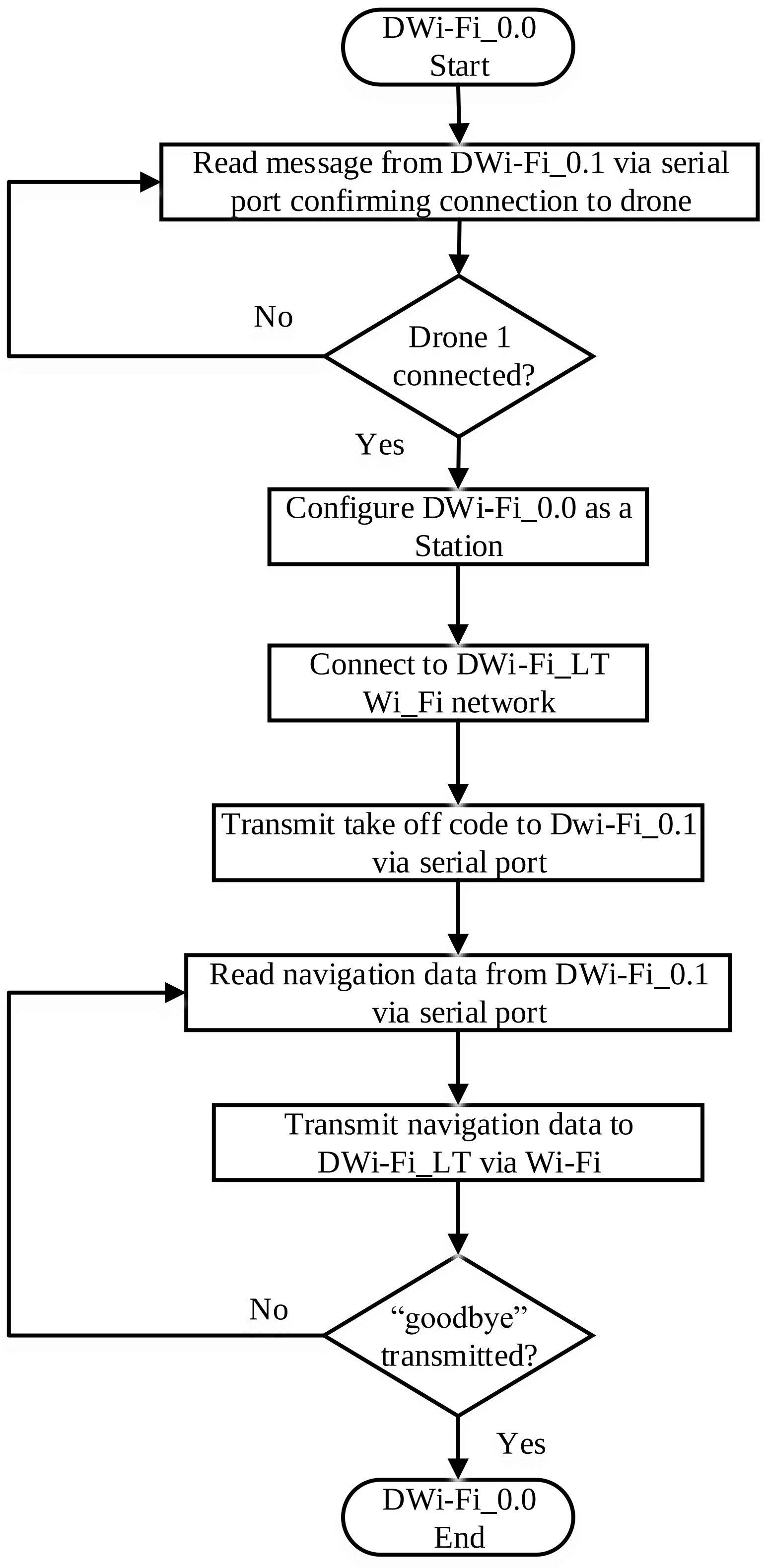

DWi-Fi_0.0
Start
Read message from DWi-Fi_0.1 via serial port confirming connection to drone
Drone 1 connected?
No
Yes
Configure DWi-Fi_0.0 as a Station
Connect to DWi-Fi_LT Wi_Fi network
Transmit take off code to Dwi-Fi_0.1 via serial port
Read navigation data from DWi-Fi_0.1 via serial port
Transmit navigation data to DWi-Fi_LT via Wi-Fi
"goodbye" transmitted?
No
Yes
DWi-Fi_0.0
End

Figure 3-14 Flowchart describing module DWi-Fi_0.0 algorithm

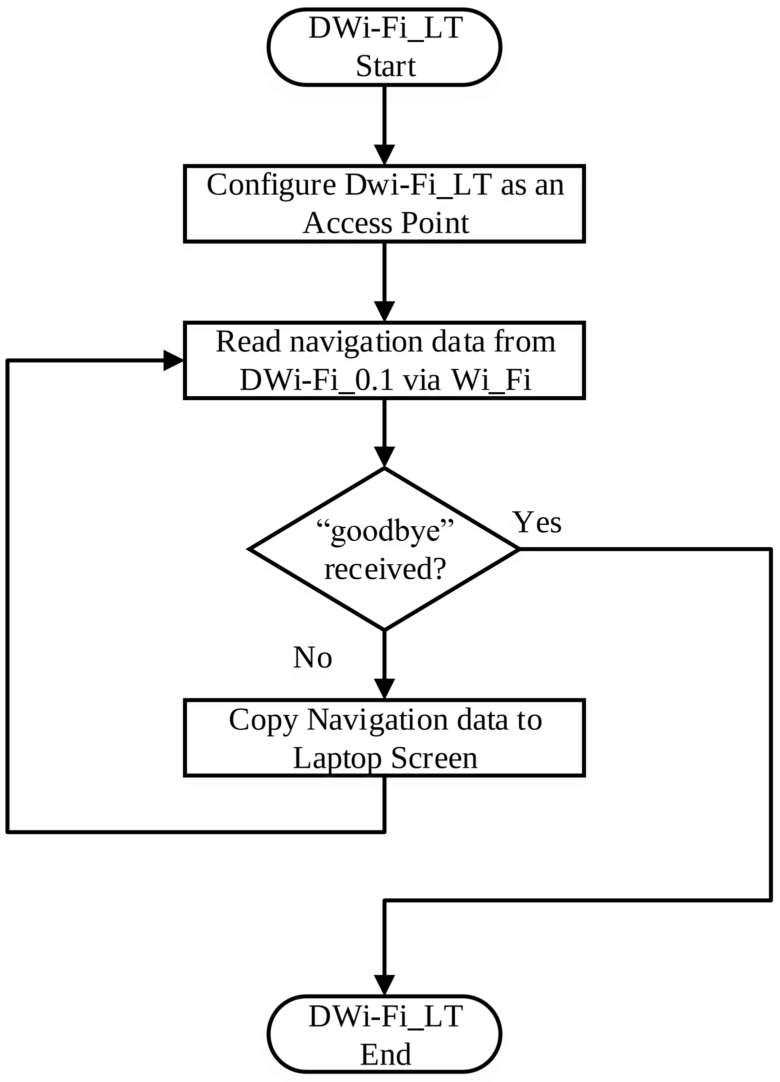

DWi-Fi_LT Start
Configure Dwi-Fi_LT as an Access Point
Read navigation data from DWi-Fi_0.1 via Wi_Fi
“goodbye” received?
Yes
No
Copy Navigation data to Laptop Screen
DWi-Fi_LT End

Figure 3-15 Flowchart describing module DWi-Fi_LT algorithm

3.6 Results and Analysis

The flight plan of Figure 3-5, which contains five of the seven developed flight control functions, is executed to examine the response of the drone to the flight control functions. The navigation data captured during the flight enables important parameters such as the height, and distance flown to be examined and compared with the expected values from the arguments included in the flight control functions. The test flight plan also enables the drone movement in pitch, roll and yaw angles to be analysed as the flight control program is executed. The graph of Figure 3.16 depicts the flight of the drone resulting from the execution of the flight control program of Figure 3-5.

The graph of Figure 3-16 shows the flight program commencing at 3.6 seconds at which time the take-off function begins execution. Two seconds later at 5.6 seconds, the take-off function is complete, movement is observed in the pitch, roll and yaw angles, and at 6.6 seconds, the drone lifts off the ground. The hover function begins execution at 5.6 seconds (before the drone has actually left the ground) and continues for 9 seconds, as requested in the program, until completion at 14.6 seconds.

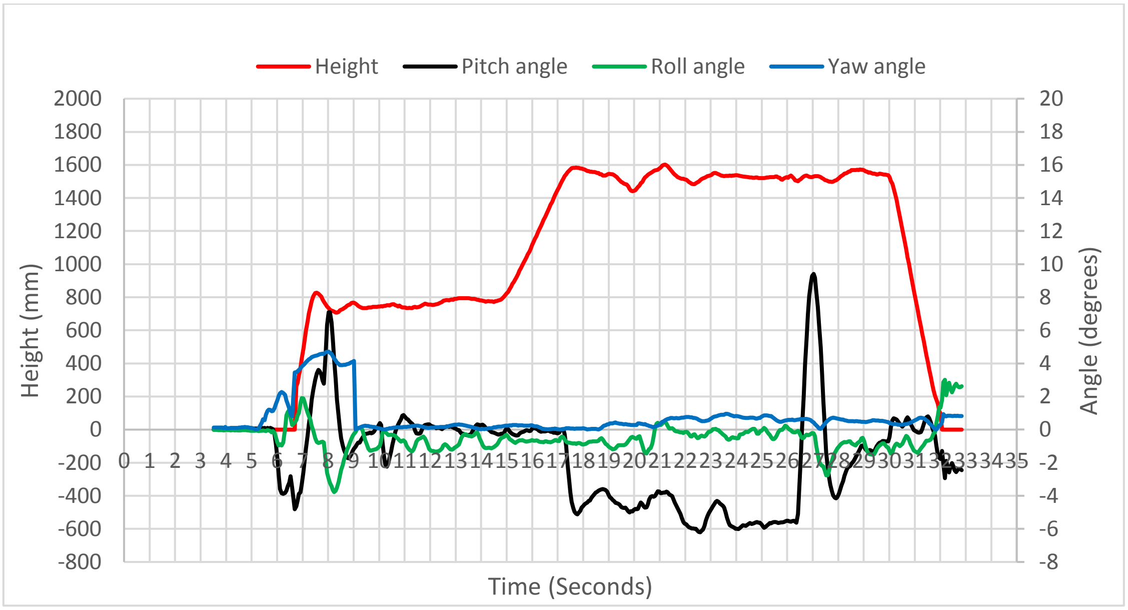


Figure 3-16 Graph of pitch angle, roll angle, yaw angle and drone height plotted against time

During the execution of the hover function the drone is observed to hover at a height between 740 and 800cm. At 14.6s, the third phase of the flight plan (ascend to 1500mm) begins execution. The drone is observed to climb initially to 1600mm before finally settling at 1520mm. The ascend function completes at 17.2 seconds at which time the next phase of the flight plan, fly forwards 12m, begins. The pitch angle decreases to 5 degrees initially, indicating the drone is pitching forward and therefore moving in the forward direction.

The graph of Figure 3-17 shows the velocity in the x direction beginning to increase at 17.2 seconds until levelling out at approximately 1.5m/s. The drone is observed to fly in the forward direction until the required 12m is reached at 25.9 seconds. The graph of Figure 3-16 shows the drone pitching backwards 9.1 degrees, producing a braking action, at 26.4 seconds. The drone comes to a halt after flying a total of 14m, overshooting the required distance by 2m, due to the drone's momentum. The drone remains in the air hovering for 3 seconds as requested in the flight program before being instructed to land when the final function in the flight control program is executed.

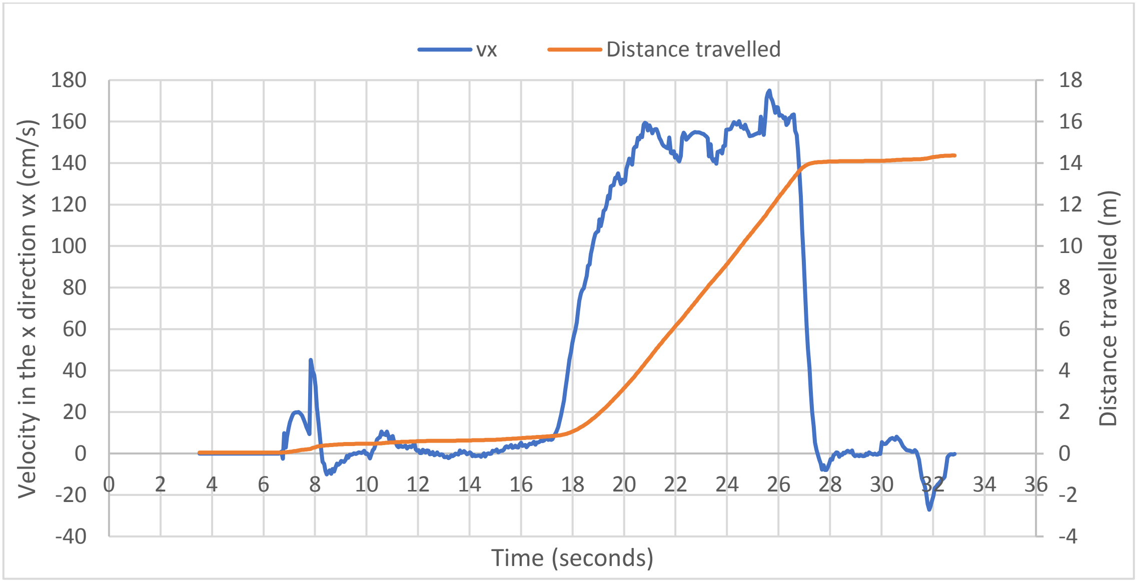


Figure 3-17 Graph showing the velocity in the x direction and the distance travelled plotted against time

The drone completes the flight plan of Figure 3-5 as expected, however the test flight also reveals two issues.

(i) When the brake is applied the drone pitches back 9.1 degrees and comes to a halt. Although not so much of an issue at lower velocities, 1.5m/s in this test flight, at higher velocities the harsh brake can cause significant deviation in height and direction.

(ii) Although requested to fly a distance of 12m, in function 4 of the flight plan, the drone overshoots the requested distance by 2m. The overshoot is due to the momentum of the drone as it accelerates to its terminal velocity of approximately 1.5m/s.

Modifications to function 4 of the flight plan which flies the drone in the forward direction are discussed in the next sections to address the points raised in (i) and (ii) above.

3.6.1 Flight Function Modification - Braking

The worst case scenario of the harsh brake application is depicted in Figure 3-18 when the brake is applied when the drone has reached maximum velocity. The pitch angle is observed to swing from -19 degrees when flying forward, to 31 degrees in the reverse direction, in 1.2 seconds, resulting in a rapid deceleration, impacting drastically on the drone height and direction of flight.

The velocity of flight is determined by the velocity argument (b), which can vary from 0 to -0.4 for forward flight and 0 to +0.4 for reverse flight, in the forward_backward(a,b) flight control function. The greater the magnitude of the velocity argument, the greater the flight velocity. When flying in the forward direction, a soft brake is introduced by increasing the velocity argument by 0.1 increments in the flight control function, with a short delay in between increments, until the velocity argument value is equivalent to the magnitude of the original velocity argument, but is now positive. The brake is thus applied to the drone in a more controlled manner over a greater period of time.

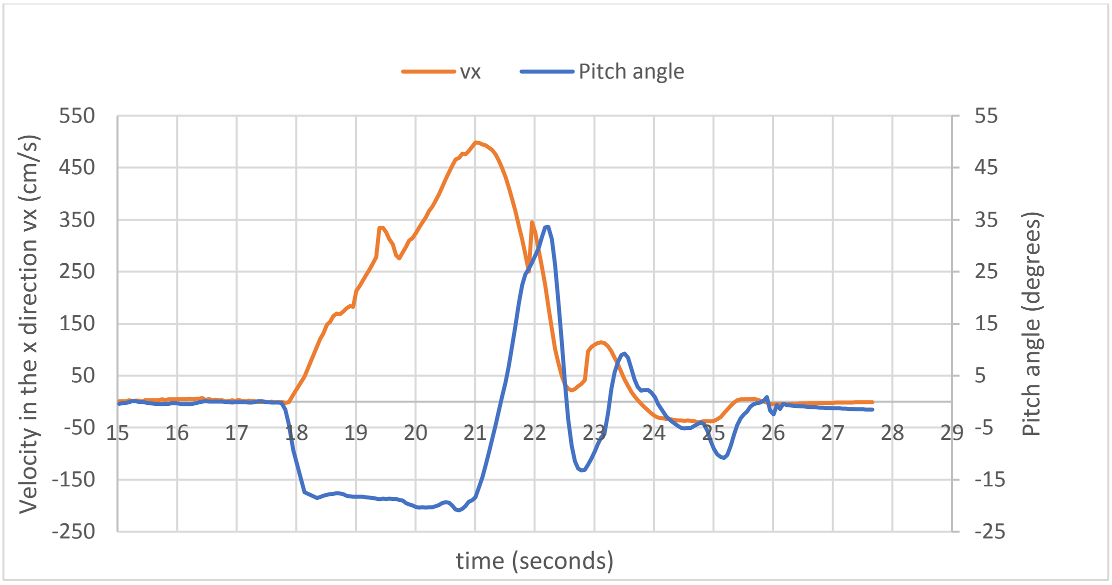


Figure 3-18 Graph of velocity in the *x* direction and pitch angle plotted against time

In a similar fashion the forward_backward function is modified to accelerate the drone in controlled increments. The graph of Figure 3-19 depicts the result of the drone flight incorporating the modified program.

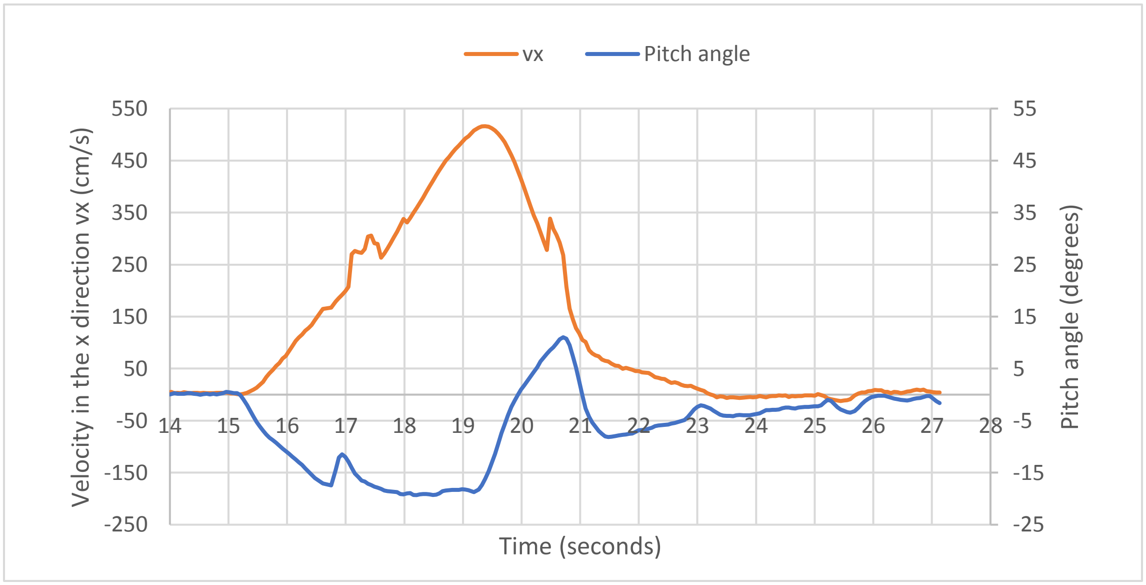


Figure 3-19 Graph of velocity in the x direction (after acceleration and deceleration modification), and pitch angle plotted against time

Although the drone takes longer to accelerate to the maximum velocity as would be expected, a significant improvement is observed when the brake is applied. The soft brake ensures the drone only pitches back 11 degrees compared to 31 degrees previously. The issues of uncontrolled movement of the drone, created by a hard brake, are thus nullified and the drone comes to a controlled halt.

3.6.2 Flight Function Optimisation - Overshoot

To further examine the overshoot experienced by the drone at the moment in time when the brake is applied, the drone is flown for 12m at velocity arguments of -0.1,-0.15, -0.2 and-0.25. The resulting overshoot for the four test flights is depicted in Figure 3-20. The distance overshoot results displayed in Figure 3-20 were taken with the acceleration and braking enhancement included in the flight program. The results clearly show the increase in overshoot from the required flight distance as the velocity is increased. The drone also comes to a halt earlier as the velocity increases as expected, since the drone reaches the required distance in a shorter space of time.

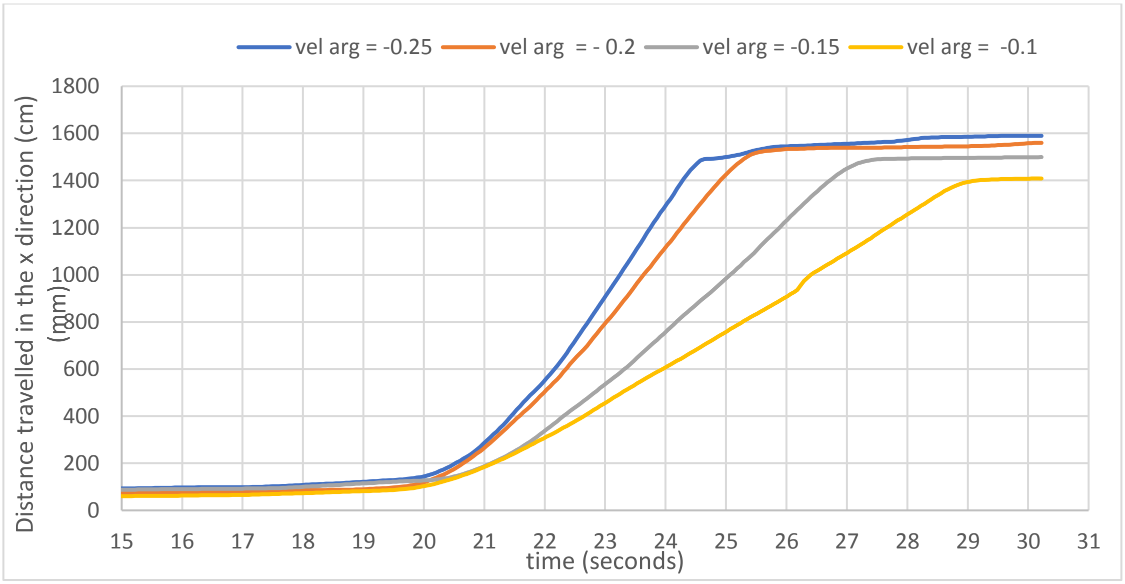


Figure 3-20 Graph of drone distance travelled against time at different velocities

Table 3-3 Table showing results of drone velocity and average overshoot

Velocity Overshoot data			
Velocity Argument	Average velocity v_x (cm/s)	Distance required (cm)	Average overshoot (cm)
-0.1	144.3	1500	220.3
-0.15	223.9	1500	343
-0.2	314.9	1500	452.6
-0.25	391.8	1500	612.8
-0.3	467.7	1200	800.6

To eliminate the overshoot, a practical algorithm is adopted, by considering the relationship between the velocity of the drone, at the moment in time when the brake is applied. The drone is flown for four flights at velocities corresponding to velocity arguments of -0.1, -0.15, -0.2, -0.25, -0.3, i.e. twenty flights in total, and the average velocity and overshoot for the four flights plotted to determine the relationship. The results are presented in Table 3-3 and plotted in the graph of Figure 3-21.

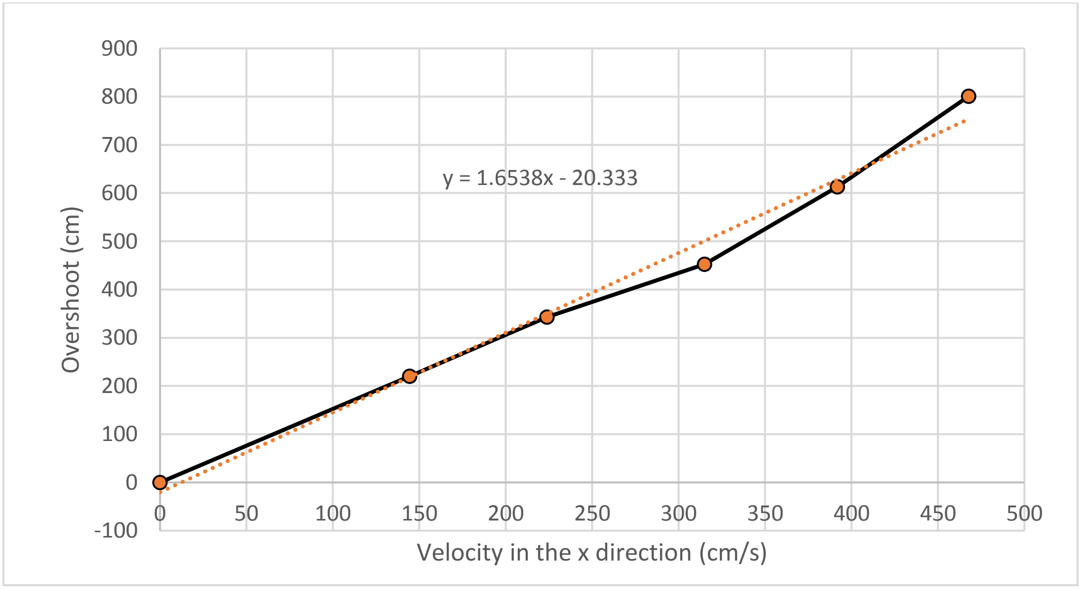


Figure 3-21 Graph of overshot plotted against drone velocity in the *x* direction

The graph of Figure 3-21 can be approximated to a linear relationship between the overshoot and the velocity in the *x* direction presented in Equation 3-6.

$$\text{overshoot} = 1.65v_x - 20.33 \qquad \text{(Equation 3-6)}$$

The equation is inserted into the forward_backward function enabling the projected overshoot to be calculated as the drone is flying forward in real time. The calculated overshoot value is subtracted from the required distance passed to the function so that the actual distance flown will be as expressed in Equation 3-7.

$$\text{distance\_new} = \text{distance\_required} - \text{overshoot} \qquad \text{(Equation 3-7)}$$

To examine the performance of the overshoot elimination algorithm, the two equations above are inserted into the forward_backward program and tests flights flown for velocity arguments of -0.1, -0.15, -0.2, -0.25 and -0.3 for a number of required distances. The actual distance travelled, the calculated overshoot, and the percentage error in the distance travelled for the test flights, are shown in Table 3-4.

Table 3-4 Table showing results of drone velocity, distance travelled, and calculated overshoot

Velocity Argument	Velocity (cm/s)	Distance Required (cm)	Distance Travelled (cm)	Calculated Overshoot (cm)	Distance error (%)
-0.1	152.3	1500	1490	232	-0.7
-0.1	154.0	2000	2017	234	0.8
-0.15	221.1	1500	1586	345	5.4
-0.15	227.8	2000	1985	356	-0.7
-0.2	294.8	1500	1517	467	1.1
-0.25	380.6	1500	1540	609	2.7
-0.3	405.0	1500	1643	650	8.7

The ‘Calculated Overshoot’ column entries are calculated from Equation 3-6 in real time and are subtracted from the ‘Distance Required’ in the program to determine when the brake should be applied. In the first row for example the brake is applied after the drone has flown 1500cm – 232cm = 1268cm. The drone’s momentum carries the drone forward a further 222cm so that the drone actually flies forward a distance of 1490cm. The results show good distance accuracy when flown at lower velocities. As the velocity increases the distance error is also observed to increase with a maximum error of 8.7% when flown at a velocity of 405cm/s. An acceptable error for the drone flying a relatively short distance at high speed.

3.6.3 Results of Autonomous UAV Following a Square Flight Plan

To examine the drone’s response to a more complex 3-dimensional flight plan, the drone is requested to fly a square of side 7m at a velocity of 1.5m/s (velocity argument -0.1). The x, y plot of the drone flight is depicted in Figure 3-22 and flight accuracy measurements displayed in Table 3-5.

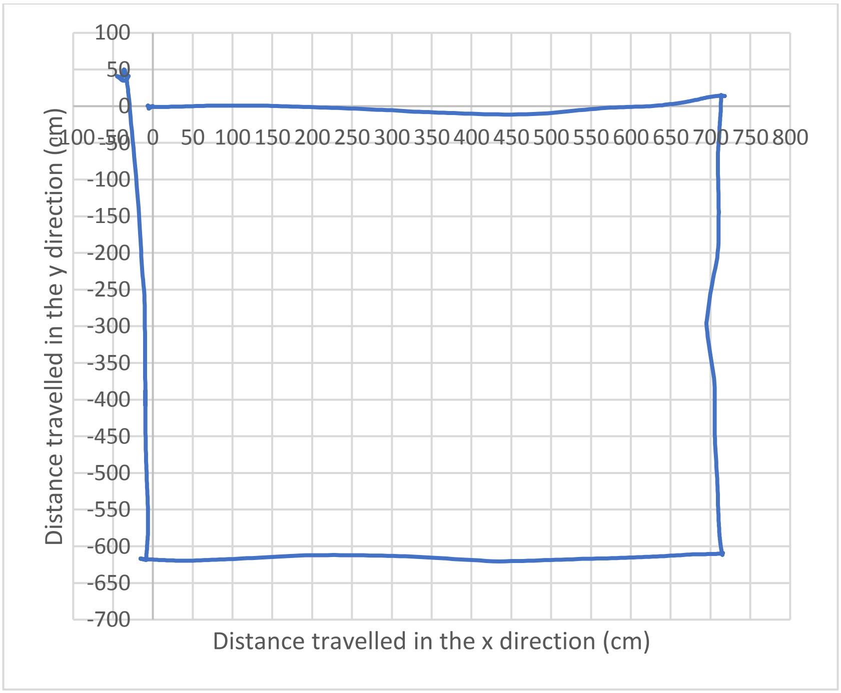


Figure 3-22 Graph showing the *x-y* plot of the autonomous controlled drone flying a 7m square

Table 3-5 Table showing the deviation in direction and distance from the 7m square flight plan

Direction of flight	Distance travelled (cm)	Deviation from required distance (%)	Deviation from required direction (%)
Forward +x	713	2 (long)	2 (to the left)
Right –y	618	12 (short)	0
Backward -x	724	3 (long)	3 (to the right)
Left +y	659	6 (short)	4 (to the back)

The diagram of Figure 3-22 shows the drone commencing the 3-dimensional flight plan from the origin. The drone then flies the first phase in the x direction, flies the second phase rolling in the –y direction, flies the third phase backwards in the –x direction, and finally completes the square by rolling in the +y direction.

The results of Table 3-5 show the drone flying further than the requested distance in the x direction, and shorter than the requested distance in the y direction, when completing the square. Although relatively low deviations from the required distances are observed, these percentage deviations will reduce further, if greater distances are required in the control program. Deviations of less than 5% are observed in the direction of flight.

3.7 Summary

Results taken during programmed flights show that the laptop and NodeMCU Wi-Fi modules provide the hardware platform required for autonomous drone flight control. The programs developed to run on the hardware platforms, are written as functions enabling them to be combined to produce any required 3-dimensional flight plan. The distance travelled by the drone in both the x and y directions, is calculated by integrating the available velocity in the x direction v_x and the velocity in the y direction v_y, with respect to time. The resulting incremental distance values in the x and y directions enables the flight to be drawn as an x-y plot. The enhancement made to the flight control functions to minimise the braking effects on the drone caused by a harsh brake, are shown by results, to significantly reduce deviations to the drone flight. Results also show that the distance overshoot modification to functions, improves the accuracy of the requested distance of flight. The NodeMCU hardware platform and the developed flight control programs enable autonomous flight control where the drone can be programmed to fly any desired 3-dimensional flight plan.

Chapter 4

Estimation of UAV Flightpath from the Relative Angular Velocity of the Quadcopter Rotors (AVQR)

4.1 Introduction

The flightpath of a quadcopter drone can be determined and controlled by GPS or waypoint methods. These methods however require additional specialist components, and in the case of GPS, outdoor flights. This chapter introduces a unique method to estimate the drone flightpath, utilising the relative angular velocity of the four rotors and thus eliminating the shortcomings of GPS techniques and the requirement of additional components or equipment.

The differences in the angular velocity of the four rotors of a quadcopter drone creates torque, causing the drone to pitch, roll or yaw in the appropriate direction. The torque is thus proportional to the angle of pitch, roll or yaw generated and the resulting drone directional movement. A torque equation incorporating the torque components of pitch, roll and yaw is developed from first principles from which the flightpath of the drone can be estimated.

The accuracy of the estimated flightpath is presented by analysing the results generated from the torque equation for a number of 3-dimensional test flights.

4.2 The Rotor Torque Equation

The relationship between the rotor speed and the movement of the drone can be shown to be a function of the torque developed by the four rotors. The overall torque can be calculated by a consideration of the torque developed in the three aspects of pitch, roll and yaw movement. The torque developed by pitch and roll aspects can be calculated from the thrust developed by the four rotors (i). The torque developed by the yaw aspect is determined by the angular velocity of the rotors in the Euler rigid body equation (ii) [90].

(i) Torque equations for pitch and roll movement.

Whilst the rotors rotate, the blades generate downward thrust. The power required to generate the required thrust is given by [90].

$$P_T = Tv_h \qquad \text{(Equation 4-1)}$$

Where P_T is the power required to generate the ideal thrust, T is the thrust and v_h is the velocity of air passing through the blades.

From momentum theory, the power required to produce a given thrust is given by:

$$P_T = \sqrt{\frac{T^3}{2\rho A}} \qquad \text{(Equation 4-2)}$$

Where ρ is the density of air, and A is the area described by the rotating blade [90].

The power consumed while a rotor is rotating can be derived from the following equation:

$$P_C \approx \frac{\mathrm{k_v}}{\mathrm{k_t}} \tau\omega \qquad \text{(Equation 4-3)}$$

Where P_C is the power consumed, τ is the torque, $\mathrm{k_v}$ is a proportionality constant and $\mathrm{k_t}$ represents a torque proportionality constant. The constant $\mathrm{k_v}$ is the motor velocity constant (or back EMF constant) applied to brushless DC motors. The motor velocity constant is defined as the ratio of the motors' angular velocity without load, to the peak voltage across the motor coils (the back EMF). The maximum value of $\mathrm{k_v}$ is therefore limited by the maximum no load rotor angular velocity and the minimum peak voltage across the motor coils. The torque constant $\mathrm{k_t}$ is defined by the ratio of the DC motor's output torque to the current flowing through the motor windings. It is a function of the motor design including the armature length and the number of wire turns. The maximum value of torque constant $\mathrm{k_t}$ is therefore determined by the maximum output torque output and the minimum current flowing through the motor windings.

Combining Equation 4-2 and Equation 4-3, the relationship between thrust T and angular velocity ω can be derived as follows:

$$\sqrt{\frac{T^3}{2\rho A}} = \frac{\mathrm{k_v}}{\mathrm{k_t}} \tau\omega = \frac{\mathrm{k_v}}{\mathrm{k_t}} \mathrm{k_\tau} T\omega \qquad \text{(Equation 4-4)}$$

and

$$T = \left(\frac{\mathrm{k_v k_\tau}\sqrt{2\rho A}}{\mathrm{k_t}} \omega \right)^2 = \mathrm{k}\omega^2 \qquad \text{(Equation 4-5)}$$

The thrust T is thus directly proportional to the square of the angular velocity ω of each rotor.

A quadcopter drone has four rotors. To ensure the rotors are balanced, and the drone does not rotate when the rotors are rotating, the Parrot AR2 drone's front left rotor (rotor 1) rotates in a clockwise direction, the front right rotor (rotor 2) rotates in an anticlockwise direction, the rear right rotor (rotor 3) rotates in a clockwise direction and the rear left rotor (rotor 4) rotates in an anticlockwise direction. The relative variation in velocity of the rotors impacts on the pitch angle, yaw angle and roll angle of the drone which ultimately governs the drone's flightpath.

Forward flight requires an increase in velocity of rotors 3 and 4 and a decrease in velocity in rotors 1 and 2. The rotor velocity imbalance from the rear to the front generates torque causing the drone to pitch forwards and rotate about its centre of gravity. As the rotors angle away from the horizontal plane the drone moves forwards. The angle the drone pitches forwards (the pitch angle) will determine the velocity of flight. To continue forward flight at constant velocity the torque applied to the drone is required to be maintained, ensuring that the pitch angle remains constant. To fly in reverse, the above also applies but with an increase in speed of rotors 1 and 2 and a decrease in speed of rotors 3 and 4. A similar explanation is true for lateral (roll) movement [91].

The torque generated for pitch or roll can be determined by consideration of the standard torque formula:

$$\tau_i = rT_i \qquad \text{(Equation 4-6)}$$

Where τ_i is the torque generated by a rotor i, r is the radius between the drone body to each rotor (common for each rotor), and T_i is the thrust generated by rotor i. For forward flight i.e. the drone is pitching forwards, the torque due to the pitch angle is determined by the difference between the front thrust and the rear thrust.

$$\tau_{pitch} = rT_1 + rT_2 - rT_3 - rT_4 \qquad \text{(Equation 4-7)}$$

Substituting Equation 4-5

$$\tau_{pitch} = r(k_1\omega_1{}^2 + k_1\omega_2{}^2 - k_1\omega_3{}^2 - k_1\omega_4{}^2) \qquad \text{(Equation 4-8)}$$

Assuming the PWM rotor speed value has a linear relationship with the angular velocity ω.

$$PWM_i = \mathrm{k}_{\mathrm{PWM}}\omega_i \qquad \text{(Equation 4-9)}$$

Substituting into Equation 4-8 yields

$$\tau_{pitch} = \frac{r\mathrm{k}_1}{\mathrm{k}^2_{\mathrm{PWM}}}\left(PWM_1{}^2 + PWM_2{}^2 - PWM_3{}^2 - PWM_4{}^2\right) \qquad \text{(Equation 4-10)}$$

Applying the same process to the torque due to the roll angle yields:

$$\tau_{roll} = \frac{r\mathrm{k}_1}{\mathrm{k}^2_{\mathrm{PWM}}}\left(PWM_1{}^2 + PWM_4{}^2 - PWM_2{}^2 - PWM_3{}^2\right) \qquad \text{(Equation 4-11)}$$

(ii) Torque equation for yaw angle:

The torque due to the movement in the yaw angle can be determined by consideration of the Euler rigid body equation shown in Equation 4-12, which identifies the relationship between the torque and the angular velocity for each rotor [90]

$$\mathbf{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega}\left(\mathbf{I}\boldsymbol{\omega}\right) = \mathbf{M} \qquad \text{(Equation 4-12)}$$

where **I** is the inertia matrix, **ω** is the angular velocity about the principle axes and **M** is the applied torque.

The drone has four rotors each of which will generate torque, impacting on the whole system. The torque due to yaw movement is given by the sum of the torque generated by each individual rotor, represented by the following equation [92].

$$\tau_{yaw} = \sum_{i=1}^{4} \mathbf{M}_i = \sum_{i=1}^{4} \mathbf{I}\dot{\boldsymbol{\omega}}_i + \boldsymbol{\omega}_i(\mathbf{I}\boldsymbol{\omega}) \qquad \text{(Equation 4-13)}$$

In steady state flight the acceleration component $\dot{\boldsymbol{\omega}}$ approximates to zero and can be removed from the equation yielding:

$$\tau_{yaw} = \sum_{i=1}^{4} \mathbf{M}_i = \boldsymbol{\omega}_i^{2}\mathbf{I}_i \qquad \text{(Equation 4-14)}$$

Applying the same previous assumption that the PWM rotor speed value has a linear relationship with the angular velocity $\boldsymbol{\omega}$ yields:

$$\tau_{yaw} = \frac{\mathbf{I}}{\mathrm{k}^{2}_{\mathrm{PWM}}}\left(PWM_1^{2} + PWM_3^{2} - PWM_2^{2} - PWM_4^{2}\right) \qquad \text{(Equation 4-15)}$$

The overall rotor torque equation attributed to pitch, roll and yaw angle movement is thus given by the following matrix equation.

$$\tau_{Drone} = \begin{pmatrix} \tau_{pitch} \\ \tau_{roll} \\ \tau_{yaw} \end{pmatrix} = \begin{pmatrix} \mathrm{k}\left(PWM_1^{2} + PWM_2^{2} - PWM_3^{2} - PWM_4^{2}\right) \\ \mathrm{k}\left(PWM_1^{2} + PWM_4^{2} - PWM_2^{2} - PWM_3^{2}\right) \\ \mathrm{a}\left(PWM_1^{2} + PWM_3^{2} - PWM_2^{2} - PWM_4^{2}\right) \end{pmatrix} \qquad \text{(Equation 4-16)}$$

4.3 Rotor Torque Equation Test Results and Analysis

To evaluate the pitch and roll components of the rotor torque equation, the drone is flown under laptop control in a 3-dimensional, 7m square flight plan, at a velocity of approximately 1.5m/s. The yaw component of the torque equation is evaluated by rotating the drone about its z-axis, 90 degrees, 180 degrees and 270 degrees at velocity arguments of -0.1, -0.2 and -0.3. The angular velocities of the rotors are captured as navigation data transmitted from the drone to the laptop during the test flights. The rotor angular velocity data is expressed as a PWM value, where 0 is the minimum value, and 255 is the maximum value. The angular velocity (PWM) values for each rotor are substituted into the rotor torque equation, enabling graphs to be plotted of the rotor torque equation against time for the different test flights. The three components of the torque equation are analysed to establish the significance in the variation of these values as the drone completes the test flights.

4.3.1 Analysis of the Pitch and Roll Components of the Torque Equation

The graph of Figure 4-1 depicts the raw angular velocity data plotted against time for the drone following a 3-dimensional flight plan where it takes off and hovers, climbs to 1.2m, flies forward 7m at a velocity of approximately 1.5m/s and then brakes and hovers. The rotors are numbered such that rotor 1 is front left, rotor 2 is front right, rotor 3 is rear right and rotor 4 is rear left.

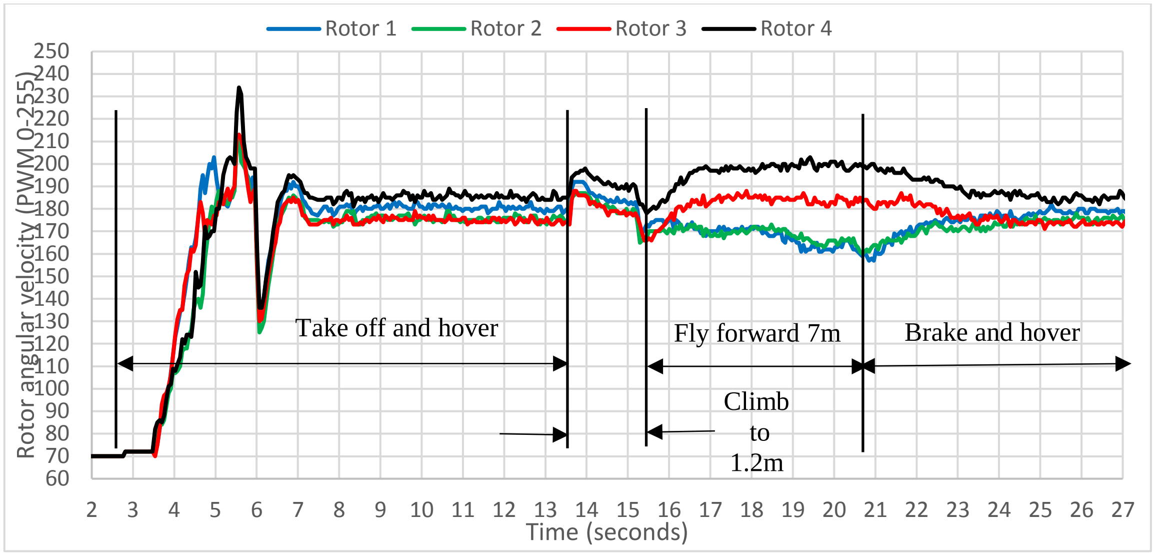


Figure 4-1 Graph showing the angular velocity of the four rotors (PWM values) plotted against time

It is clear from the graph of Figure 4-1 that realising the 3-dimensional flight plan from the raw angular velocity data is challenging. It is also observed from Figure 4-1 that when hovering (between 7s and 13s) the rotors do not rotate at the same angular velocity but differ from a minimum PWM value of 175 to a maximum PWM value of 185. As the developed torque equation assumes the angular velocity of all rotors is equal whilst hovering in the horizontal position, the angular velocity values are normalised to the average rotor angular velocity when the drone is hovering. The normalised angular velocity values for the four rotors are inserted into the torque equation resulting in the graph of Figure 4-2 which depicts the pitch torque (blue) and the roll torque (red) for the drone flight rotor data. The yaw torque is not shown in this graph as there is no rotational movement of the drone in this test flight. There is minimal movement in the roll torque in Figure 4-2 as would be expected for the drone flying in the forward direction. The pitch torque also oscillates about the zero torque position until 15.2s has elapsed when the drone begins to move forward. The decrease in the pitch torque indicates that the two rear rotors are rotating at a greater angular velocity than the two front rotors initiating forward movement.

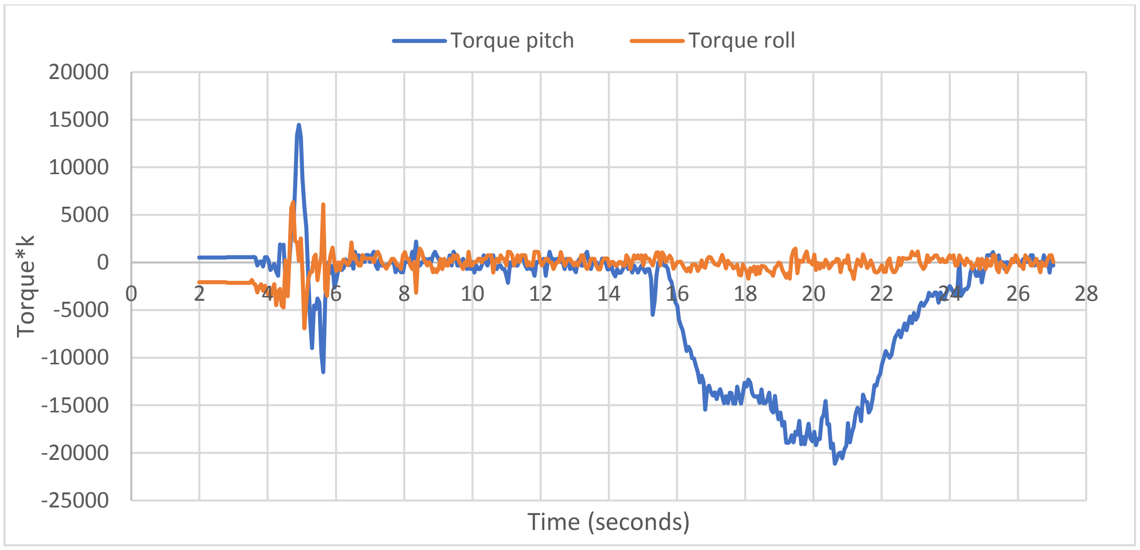


Figure 4-2 Graph showing the torque generated in the pitch and roll components plotted against time.

The pitch torque continues to decrease to a minimum torque value of approximately -180000. At 20.2 seconds the brake is applied and the pitch torque increases until it reaches 0 torque at approximately 25s when the drone hovers in the horizontal plane.

The distance the drone has flown in the forward direction can be estimated by determining the sum of:

i) d_a = the distance the drone flies during the acceleration phase to terminal velocity.

ii) d_v = the distance the drone flies at terminal velocity when the acceleration is equal to zero.

iii) d_b = the distance the drone flies during the deceleration phase from terminal velocity to zero velocity.

The terminal velocity of the drone when flown with velocity arguments of -0.1, -0.2, and -0.3 is depicted in. Figure 4-3.

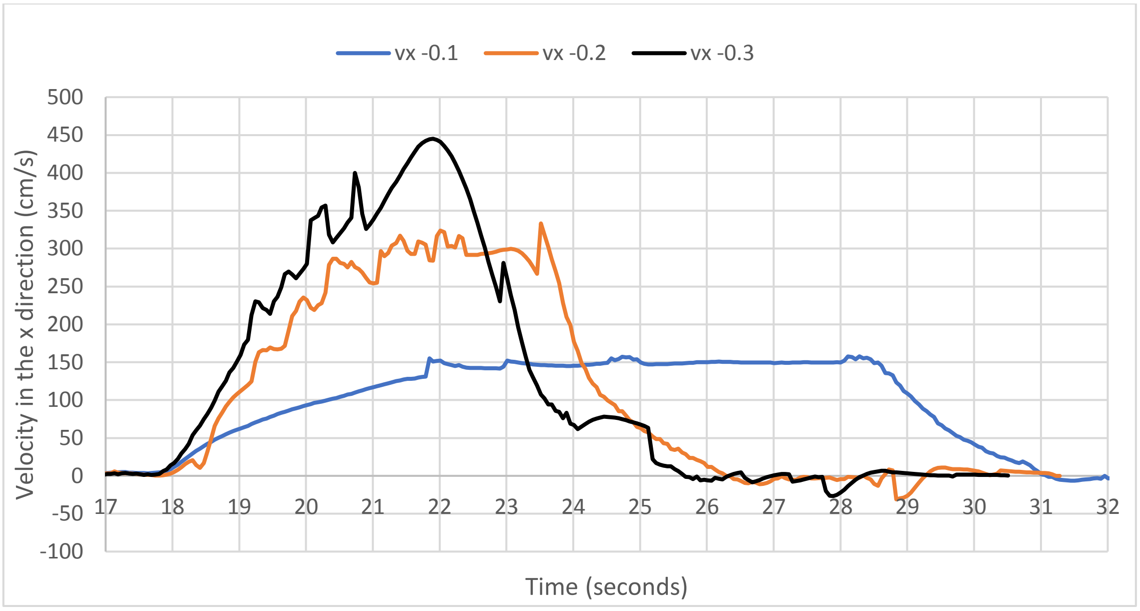


Figure 4-3 Graph showing drone velocities for velocity argument values of -0.1, -0.2 and -0.3

The graph of Figure 4-3 enables the velocity of the drone for differing velocity argument values to be determined and is summarised in Table 4-1.

Since the velocity argument in the program is set to -0.1 then the drone in the test flight accelerates to a velocity of 1.5ms. The distance that the drone flies in the forward direction can be determined from the graph of Figure 4-4.

Table 4-1 Table showing drone velocity arguments and corresponding drone velocity.

Velocity argument	Velocity cm/s	Velocity m/s
-0.1	150	1.5
-0.2	300	3
-0.3	450	4.5

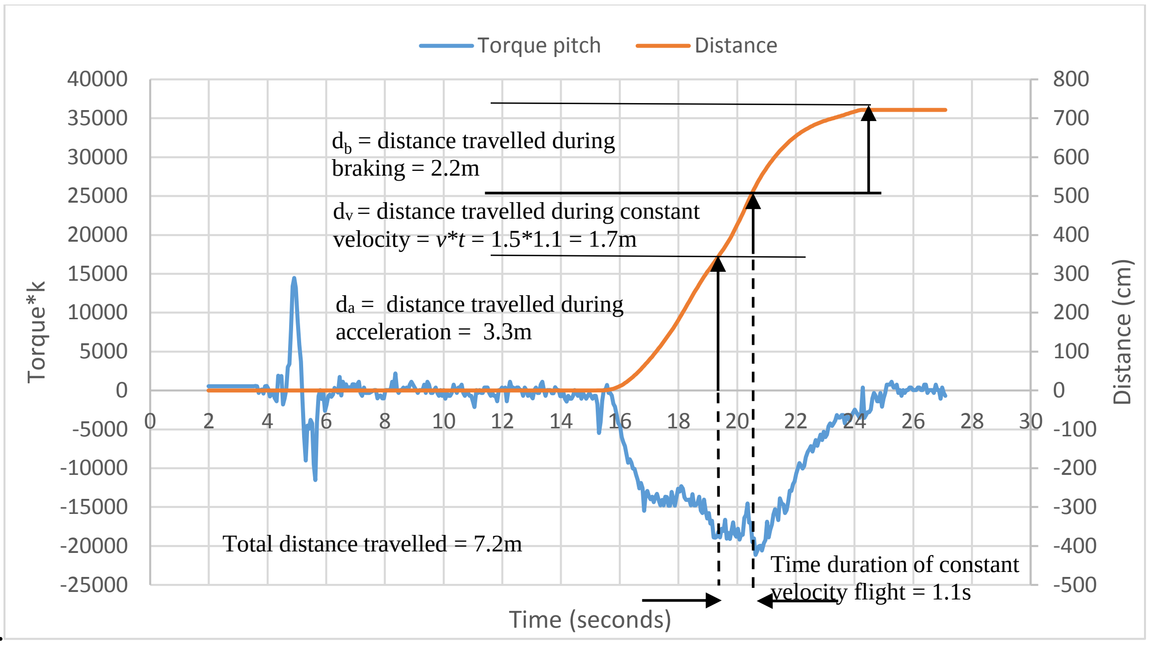


Figure 4-4 Graph of distance travelled and pitch torque plotted against time

The distance travelled during the acceleration phase, d_a, is determined from the graph as 3.3m, which is the distanced travelled until the pitch torque reaches steady state. The drone has then reached terminal velocity, 1.5m/s in this case, which it flies at for 1.1s. The distance travelled during the constant velocity, phase, d_v, is thus 1.7m. Finally, during the braking phase, d_b, the drone flies a further 2.2m making a total of 7.2m, for the whole flight.

For any distance flown in the forward direction greater than 7m at a velocity argument of -0.1, the values of d_a and d_b will be the same. The only calculation required in determining the total distance flown, is that of d_v, which can be determined from a graph of pitch torque plotted against time, similar to that of Figure 4-4. The total distance flown can then be realised by the sum of d_a, d_v and d_b,

Using the same technique described above, values of d_a, d_b and the terminal velocities, are determined for the drone flying in reverse, rolling to the right and rolling to the left and are displayed in Table 4-2.

Table 4-2 Table showing values of d_a, d_b and drone velocity for a velocity argument of -0.1.

Direction of flight	Distance flown during acceleration d_a (m)	Distance flown during breaking d_b(m)	Velocity m/s
Forward	3.3	2	1.5
Reverse	3	2	1.5
Roll Left	3	1.3	1.25
Roll Right	3	1.3	1.25

To examine the performance for a more complex drone flight, the rotor angular velocities for the 7m square flight plan of Chapter 4 are substituted into the torque equation yielding the graph shown in Figure 4-5.

The 3-dimensional flight plan of the drone can be determined by interpreting the graph of Figure 4-5. At 15.8s, the pitch torque (blue) goes negative whilst the roll torque remains at zero, indicating the drone is moving in the forward direction.

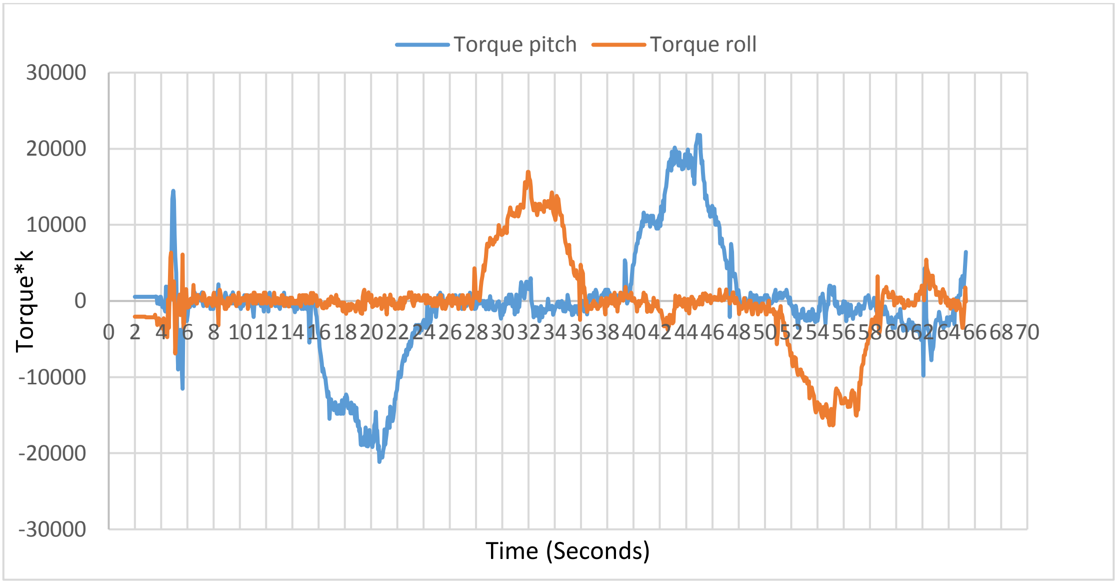


Figure 4-5 Graph showing the pitch and roll torque plotted against time for a square flight plan

From previous analysis, the drone movement is interpreted to be moving in the forward direction, 7.2m, during the time the pitch torque is negative. As the torque pitch returns to zero, a two second delay is followed by the roll torque (red) going positive. During the time the roll torque is positive, the pitch torque is approximately zero indicating the drone is rolling to the right. The distance the drone rolls to the right can be estimated from the values associated with left roll, available in Table 4-2. The distance travelled during acceleration d_a is 3m. The time the drone is flying at terminal velocity is approximately 1.8s which means the distance travelled during this time period d_v is 2.25m. The distance travelled during deceleration d_b is 1.25m, giving a total roll distance of 6.5m. After a further two second delay the torque pitch goes positive indicating a drone movement in the reverse direction. Finally the roll torque goes negative indicating a roll to the left. Considering the graph of Figure 4-5, the pitch torque indicating reverse movement and the roll torque indicating left movement are almost mirror images of the torque pitch forward and the torque roll right respectively, identifying an approximate square flight plan. The estimated distances travelled as the drone traverses a 7m square are illustrated in Table 4-3.

Table 4-3 Table showing the estimated distance travelled for a square flight plan and associated accuracy

Drone movement	Distance travelled from Figure 3-21 (m)	Estimated distance travelled from torque equation (m)	Accuracy (%)
Forward	7.1	7.2	99
Roll right	6.2	6.6	94
Backward	7.2	7.0	97
Roll left	6.6	6.6	99

The estimated distance travelled in Table 4-3 is shown to have a worst case accuracy of 94%. The pitch and roll components of the torque equation are thus observed to provide an excellent estimation of the distance travelled in the *x* and the *y* directions.

4.3.2 Analysis of the Yaw Component of the Torque Equation

Results of the drone rotating 270 degrees at a velocity argument of 0.3, and at a velocity argument of 0.2, are shown in Figure 4-6 and Figure 4-7 for comparison.

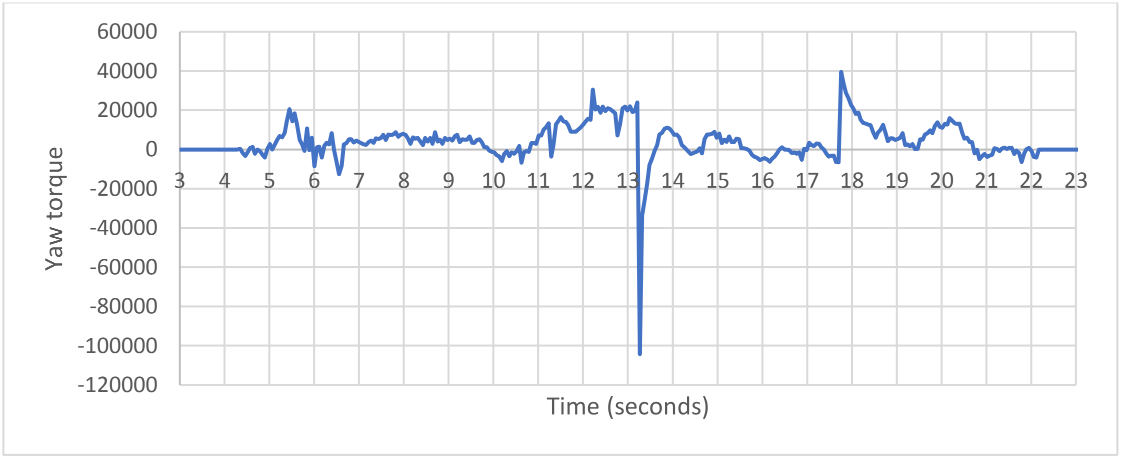


Figure 4-6 Graph of yaw torque for a 270 degree rotation with a velocity argument of 0.3 plotted against time

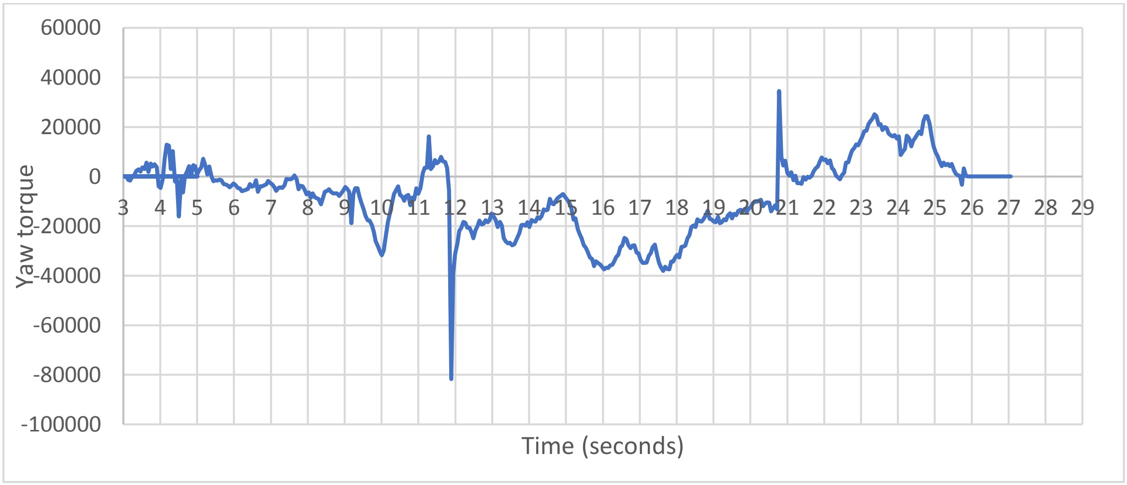


Figure 4-7 Graph of yaw torque for a 270 degree rotation with a velocity argument of 0.3 plotted against time

Prior to the negative going impulse on graphs of Figure 4-6 and Figure 4-7, the drone has taken off and is hovering. The negative impulses at 13.3 seconds and 11.9 seconds respectively depicted in the graphs of Figure 4-6 and Figure 4-7, are as a result of the torque created in the rotors initiating rotational movement. From the yaw component of the torque equation, (equation 5-16), rotors 2 and 4 (rotating clockwise) have a greater angular velocity than rotors 1 and 3 (rotating anticlockwise) creating a clockwise rotational movement. A negative impulse thus implies a clockwise rotation. A positive impulse would imply an anticlockwise rotation. The drone rotates through 270 degrees when it is informed to stop which initiates an impulse in the positive direction. The positive impulses are observed to occur at 17.8 seconds in Figure 4-6 and 20.8 seconds in Figure 4-7. The drone thus completes the 270 degree rotation in 4.5 seconds, (Figure 4-6) when rotating with a velocity argument of 0.3 equating to an angular velocity of 60 degrees/second, and 9.1 seconds (Figure 4-7) with a velocity argument of 0.2 equating to an angular velocity of 30 degrees/second. As would be expected the drone completes the 270 degree rotation faster when rotating at a greater angular velocity. The angular velocity of rotation can be determined from the magnitude of the initial impulse initiating rotational movement. From the graphs of Figure 4-6 and Figure 4-7 a rotational angular velocity of 60 degrees/second is initiated by an impulse of magnitude -104000 and a rotational angular velocity of 30 degrees/second is initiated by an impulse of magnitude 81000. Table 4-4 shows results of the drone rotating through 270, 180 and 90 degree angles and the resulting magnitude of the impulse at different angular velocities.

Table 4-4 Results showing impulse response at different angular velocities

Velocity argument	Rotation (degrees)	Time for rotation (s)	Angular velocity (degrees/second)	Magnitude of initial impulse
0.3	270	4.5	60	-104000
0.3	180	3.3	54.5	-106000
0.3	90	1.6	56.3	-100000
0.2	270	9.1	29.7	-81000
0.2	180	4.7	38.3	-61000
0.2	90	2.9	31	-95000

From the results of Table 4-4 it is observed that the angular velocity of rotation can be identified from the magnitude of the impulse initiating rotation. Knowing the angular velocity, and the time taken to complete the rotation, an approximation of the number of degrees rotated by the drone can be realised. The average angular velocities are calculated as 57 degrees/second and 33 degrees/second for angular velocity arguments of 0.3 and 0.2 respectively. Calculated angles for the time taken to complete the rotation and accuracy calculations are displayed in Table 4-5.

From the results of Table 4-5 it is concluded that a greater accuracy of calculated degrees of rotation is observed when the angular velocity of drone rotation is 57 degrees per second compared to 33 degrees per second. As the angular velocity of the drone rotation decreases, the differences between the angular velocities of the rotors also decreases, limiting the performance of the torque equation in the yaw component. When the angular velocity argument governing the drone angular velocity rotation is set to 0.1, the differences in angular velocity of the rotors is so reduced that meaningful rotor angular velocity data analysis cannot be realised.

Table 4-5 Table showing calculated angle of rotation and accuracy

Average angular velocity degrees/s	Rotation completion time (s)	Actual angle of rotation (degrees)	Calculated angle of rotation (degrees)	Accuracy (%)
57	4.5	270	256.5	95
57	3.3	180	188.1	96
57	1.6	90	91.2	99
33	9.1	270	300.3	90
33	4.7	180	155.1	86
33	2.9	90	95.7	95

4.4 Summary

In this chapter a torque equation, incorporating pitch, roll, and yaw components, is developed to estimate the 3-dimentional flightpath of a quadcopter drone. The performance of the torque equation in the pitch and roll components is examined by analysing the results generated from flying the drone in a 7m square flight plan. The yaw component of the torque equation is examined by analysing the results generated from rotating the drone through 90°, 180° and 270°decreases at different angular velocities.

The pitch and roll components of the torque equation generate graphical results which enable:

(i) the direction of flight to be realised,

(ii) the terminal velocity and hence the distance travelled, calculated to a minimum accuracy of 94% for the flight of test.

The yaw component of the torque equation generate graphical results which enable:

(i) an estimation of the drone rotational angular velocity,

(ii) an estimation of the angle of rotation. Most accurate results (95% minimum) are obtained when the drone is rotating at a higher angular velocity (57°/s). At angular velocities of less than 33°/s meaningful results cannot be realised.

For indoor applications, e.g. inside a warehouse, and applications taking place underground, e.g. in the mining industry, where GPS is not available, it is essential for a drone to be able to estimate how far it has flown in a particular direction. If a drone can estimate how far it has flown, then it can estimate its current position. The ability for a drone to be able to establish its position is necessary for it to be able to successfully complete missions for which it has been deployed. In this Chapter, a novel method which considers the relative angular velocities of the four rotors to estimate the distance travelled by a drone in a particular direction, is developed and implemented.

Chapter 5

Autonomous Dual-UAV Control

5.1 Introduction

The autonomous drone, developed in Chapter 3 incorporating two NodeMCU modules, is further developed in this Chapter to enable two drones to be flown in 3-dimensions and controlled over a network. The DWi-Fi_x.0 NodeMCUs previously discussed in Chapter 3 are able to create a network of similarly configured drones, over which flight control codes and navigation data can be transmitted and received. The network control algorithm required to enable and control multiple autonomous drones, is developed and discussed. The algorithm performance is evaluated experimentally and compared with expected results.

5.2 Autonomous Dual-UAV Hardware Configuration

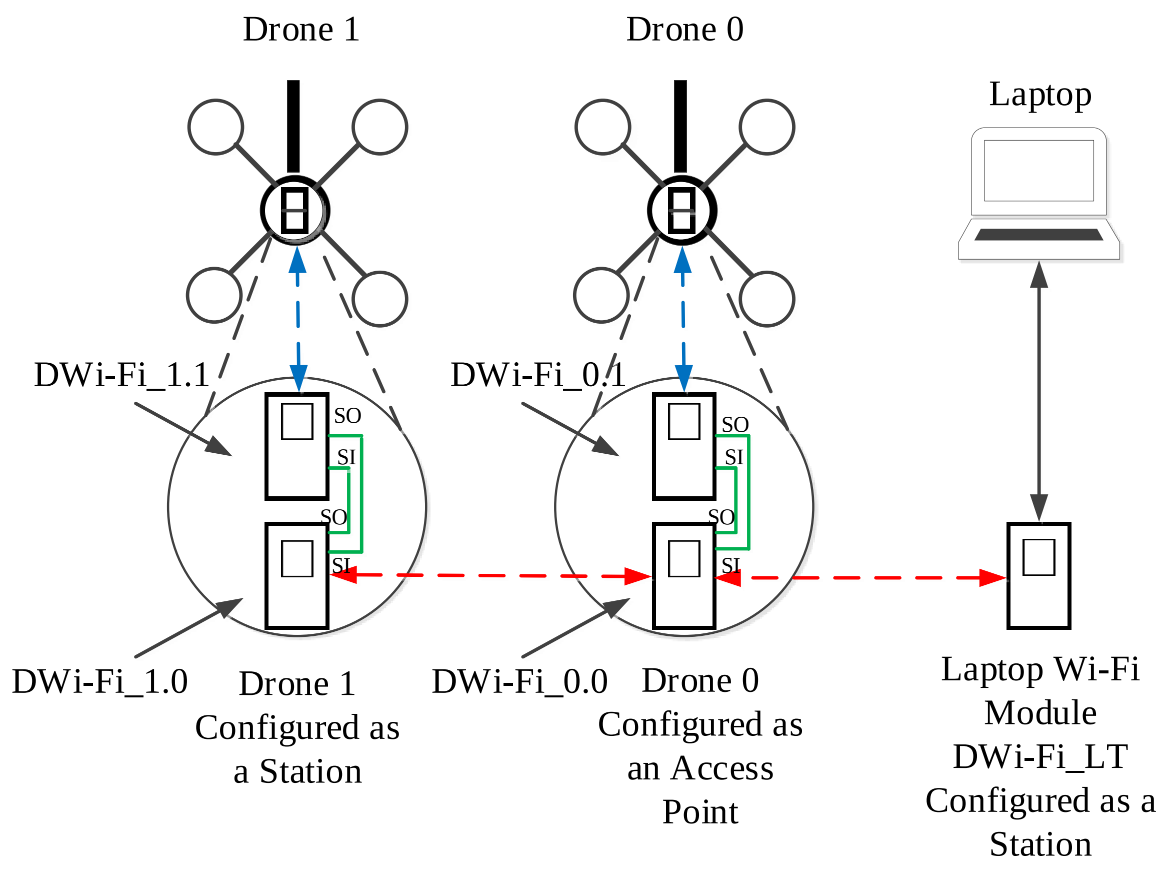
The hardware configuration for the two drone network is depicted in Figure 5-1. The two drone network of Figure 5-1 incorporates five Wi-Fi modules and three communications channels. Two Wi-Fi modules are required per drone, the function of which are as follows.

1) Module DWi-Fi_x.1 (flight control module)

a) Connects to the drone Wi-Fi (blue channel), and enables flight commands to be transmitted to the drone.
b) Reception of navigation data transmitted by the drone (blue channel).

c) Connects to DWiFi_x.0 via serial port connections (green channel). The serial port connections enable DWi-Fi_x.0 to transmit flight codes to DWi-Fi_x.1.

d) Transmission of navigation data, (received by DWi-Fi_x.1 in a), to DWi-Fi_x.0 (green channel).


Drone 1
Drone 0
Laptop
DWi-Fi_1.1
DWi-Fi_0.1
SO
SI
SO
SI
SO
SI
SO
SI
DWi-Fi_1.0
Drone 1 Configured as a Station
DWi-Fi_0.0
Drone 0 Configured as an Access Point
Laptop Wi-Fi Module DWi-Fi_LT Configured as a Station

Legend

🟥 Wi-Fi communication link between DWi-Fi_x.0 modules and DWi_Fi_LT

🟦 Wi-Fi communication link between DWi-Fi_x.1 modules and respective drone

🟩 Serial communications link between DWi-Fi_x.0 module and DWi-Fi_x.1 on the same drone

Figure 5-1 Diagram of two drone network configuration

2) Module DWi-Fi_1.0 (Network Station)

a) Connects to network created by module DWi-Fi_0.0, enabling reception of flight codes from DWi-Fi_0.0 (red channel).

b) Transmission of flight codes to DWi-Fi_1.1 (green channel).

c) Transmission of navigation data to DWi-Fi_0.0 (red channel).

3) Module DWi-Fi_0.0 (Network Access Point)

a) Transmission of flight codes to DWi-Fi_1.0 (red channel).

b) Transmission of flight codes to DWi-Fi_0.1 (green channel).

c) Reception of navigation data from DWi-Fi_1.0 (red channel).

d) Transmission of navigation data to Laptop Wi-Fi module (red channel)

The DWi-Fi_LT module is configured as a station and connects to the DWi-Fi_0.0 network to receive navigation data and display on the laptop screen.

5.3 Autonomous Dual-UAV Network Algorithm

The flowchart of Figure 5-2 describes the functionality of the overall dual drone network algorithm. After network formation both drones take off and hover for five minutes before landing

The table of Table 5-1 lists the available flight codes which can be transmitted across the network between DWi-Fi_x.0 modules, and transmitted to DWi-Fi_1.x modules via the serial port connection, to control the flight plan of the networked drones. The flight path control codes are read and interpreted by DWi-Fi_1.x modules, The DWi-Fi_1.x modules then instruct the drone to react accordingly to the received flight control code.

Table 5-1 Table showing drone flight path control codes

Flight code	Flight code function
TO	Take off
FOx	Fly forward (x metres)
REx	Fly in reverse (x metres)
RLx	Roll to the left (x metres)
RRx	Roll to the right (x metres)
LA	Land

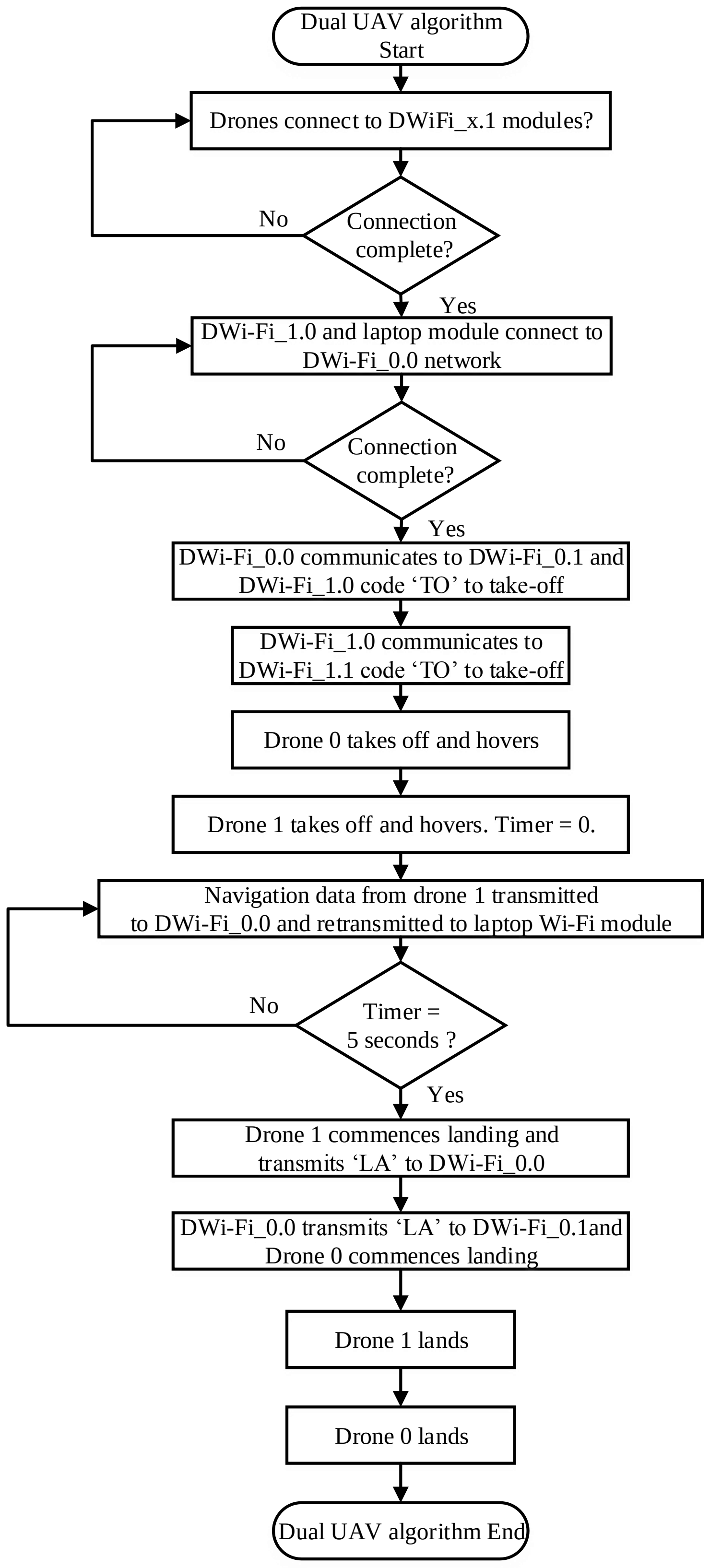

Dual UAV algorithm Start
Drones connect to DWiFi_x.1 modules?
No
Connection complete?
Yes
DWi-Fi_1.0 and laptop module connect to DWi-Fi_0.0 network
No
Connection complete?
Yes
DWi-Fi_0.0 communicates to DWi-Fi_0.1 and DWi-Fi_1.0 code 'TO' to take-off
DWi-Fi_1.0 communicates to DWi-Fi_1.1 code 'TO' to take-off
Drone 0 takes off and hovers
Drone 1 takes off and hovers. Timer = 0.
Navigation data from drone 1 transmitted to DWi-Fi_0.0 and retransmitted to laptop Wi-Fi module
No
Timer = 5 seconds ?
Yes
Drone 1 commences landing and transmits 'LA' to DWi-Fi_0.0
DWi-Fi_0.0 transmits 'LA' to DWi-Fi_0.1and Drone 0 commences landing
Drone 1 lands
Drone 0 lands
Dual UAV algorithm End

Figure 5-2 Flowchart describing the two drone flight algorithm

As described in the flowchart the algorithm begins with a network formation phase which establishes the network, followed by a flight control phase where flight control codes are transmitted between drones to enable the flight mission.

Network Formation Phase

(i) DWi-Fi_0.1 configured as a station.

DWi-Fi_1.1 configured as a station.

DWi-Fi_LT configured as a station.

(ii) DWi-Fi_0.1 attempts to connect to drone 0 Wi-Fi network and sends a message via serial port to DWi-Fi_0.1 when connected.
DWi-Fi_1.1 attempts to connect to drone 1 Wi-Fi network and sends a message via serial port to DWi-Fi_1.1 when connected.

(iii) DWi-Fi_0.0 reads confirmation of drone network connection message from DWi-Fi_0.1 via serial port and configures itself as an access point.
DWi-Fi_1.0 reads confirmation of drone network connection message from DWi-Fi_1.1 via serial port and configures itself as a station.

(iv) DWi-Fi_1.0 and DWi-Fi_LT attempt to connect to DWi-Fi_0.0 access point.
DWi-Fi_0.0 waits until both DWi-Fi_1.0 and DWi-Fi_LT are connected.

Network Formation Phase Complete

Dual Drone Flight Control Phase

(i) DWi-Fi_0.0 transmits take off command code ‘TO’ via serial port to DWi-Fi_0.1. DWi-Fi_0.0 transmits take off command code ‘TO’ via Wi-Fi to DWi-Fi_1.0.

(ii) DWi-Fi_0.1 receives ‘TO’ from DWi-Fi_0.0 and transmits the take-off command to drone 0 and drone 0 takes off.
DWi-Fi_1.0 receives ‘TO’ from DWi-Fi_0.0 and retransmits to DWi-Fi_1.1 via serial port.

(iii) DWi-Fi_1.1 receives ‘TO’ from DWi-Fi_1.0 and transmits the take-off command to drone 1 and drone 1 takes off.

(iv) DWi-Fi_1.1 transmits hover command for 5 seconds to drone.
DWi-Fi_0.1 transmits hover command to drone.
Navigation data transmitted from drone 1 to DWi-Fi_1.1 (Wi-Fi), DWi-Fi_1.1 transmits navigation data to DWi-Fi_1.0 (serial port), DWi-Fi_1.0 transmits navigation data to DWi-Fi_0.0 (Wi-Fi), and finally DWi-Fi_0.0 transmits navigation data to DWi-Fi_LT for laptop display.

(v) When 5 seconds hover time has elapsed DWi-Fi_1.1 transmits land command to drone 1 and drone 1 commences landing.
DWi-Fi_1.1 transmits land code ‘LA’ to DWi-Fi_1.0.

(vi) DWi-Fi_1.0 transmits land code ‘LA” to DWi-Fi_0.0, DWi-Fi_0.0 retransmits ‘LA’ to DWi-Fi_0.1 and DWi-Fi_0.1 transmits the land command to drone 0. Drone 0 commences landing

Dual Drone Flight Control Phase Complete

In the example the drones follow a simple flight plan of take-off, hover for 5 seconds and land. The flight plan is stored as a program in NodeMCU module DWi-Fi_1.1. Any required 3-dimensional flight plan could be followed in place of the hover function. Required flight plans can also be transmitted between networked modules DWi-Fi_x.0 using the available flight codes described in Table 5-1. Although the example presented relates to two drones, the algorithm is scalable, and can be modified with minor adjustments, to include more drones.

Flowcharts in Figure 5-3, Figure 5-4, Figure 5-5, and Figure 5-6 describe the detail of the programs uploaded to DWi-Fi_0.0, DWi-Fi_0.1, and DWi-Fi_1.0, DWi-Fi_1.1 respectively.

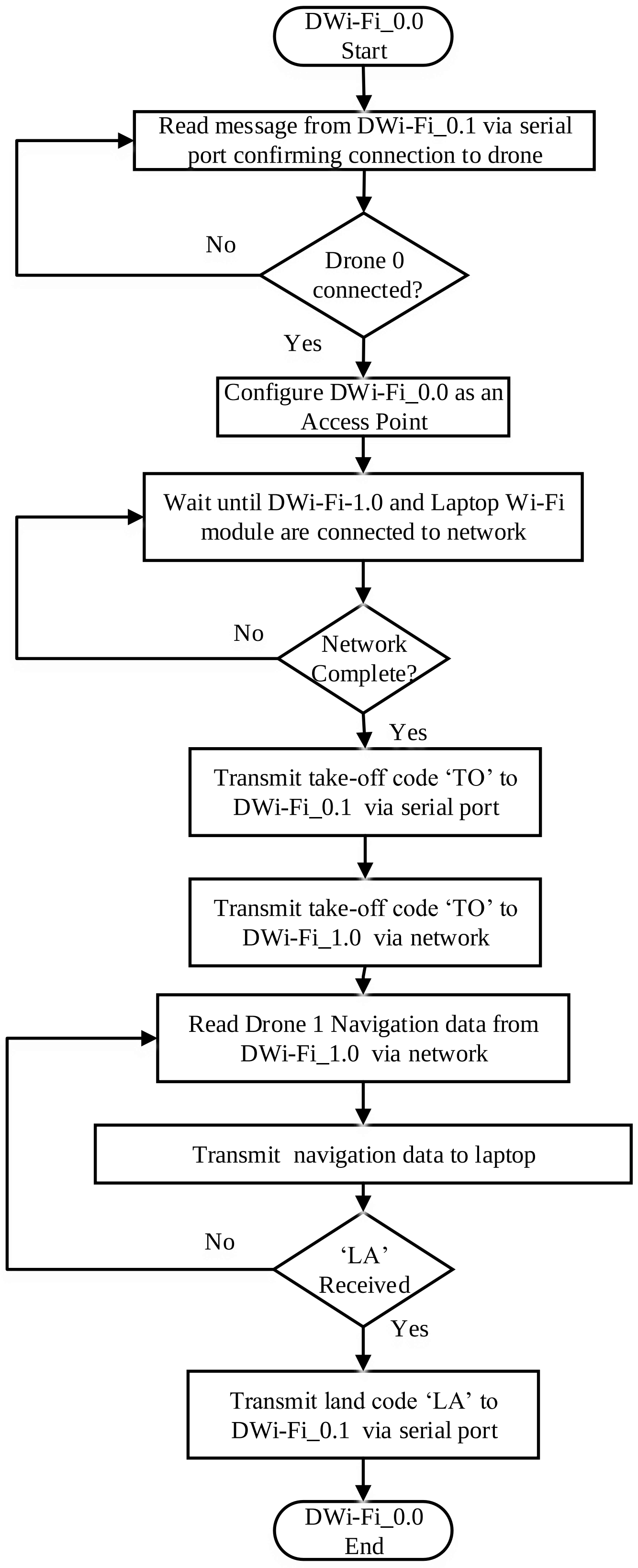

DWi-Fi_0.0 Start
Read message from DWi-Fi_0.1 via serial port confirming connection to drone
Drone 0 connected?
No
Yes
Configure DWi-Fi_0.0 as an Access Point
Wait until DWi-Fi-1.0 and Laptop Wi-Fi module are connected to network
Network Complete?
No
Yes
Transmit take-off code 'TO' to DWi-Fi_0.1 via serial port
Transmit take-off code 'TO' to DWi-Fi_1.0 via network
Read Drone 1 Navigation data from DWi-Fi_1.0 via network
Transmit navigation data to laptop
'LA' Received
No
Yes
Transmit land code 'LA' to DWi-Fi_0.1 via serial port
DWi-Fi_0.0 End

Figure 5-3 Flowchart describing DWi-Fi_0.0 program algorithm

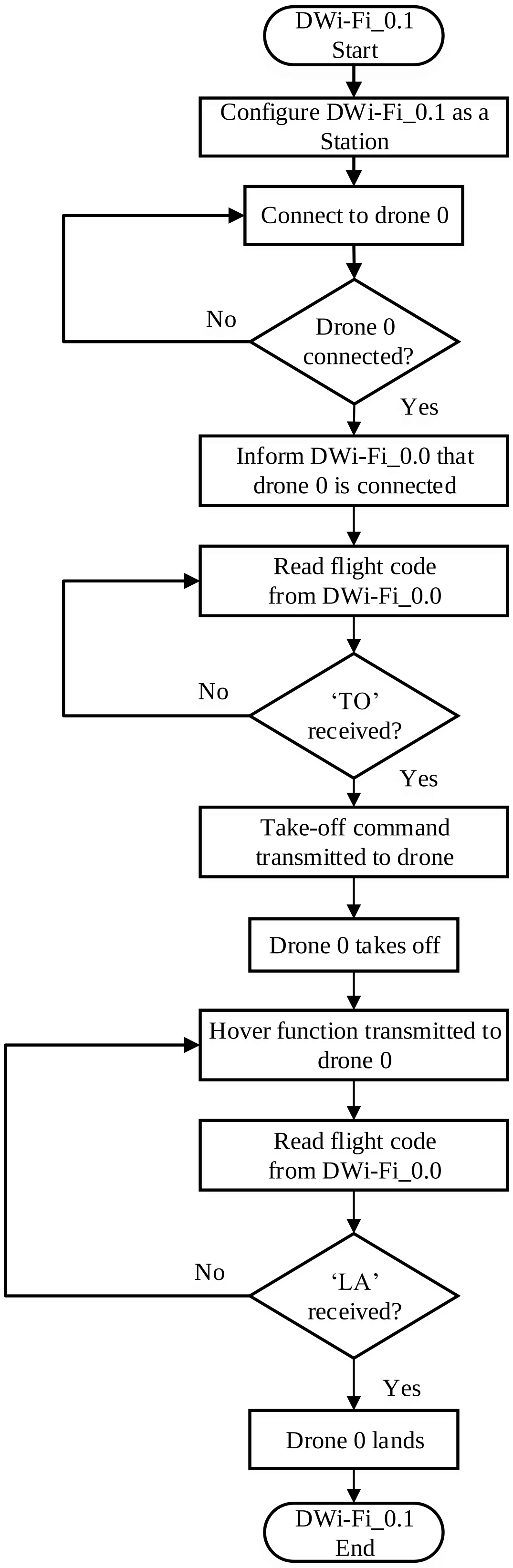

DWi-Fi_0.1 Start
Configure DWi-Fi_0.1 as a Station
Connect to drone 0
Drone 0 connected?
No
Yes
Inform DWi-Fi_0.0 that drone 0 is connected
Read flight code from DWi-Fi_0.0
'TO' received?
No
Yes
Take-off command transmitted to drone
Drone 0 takes off
Hover function transmitted to drone 0
Read flight code from DWi-Fi_0.0
'LA' received?
No
Yes
Drone 0 lands
DWi-Fi_0.1 End

Figure 5-4 Flowchart depicting DWi-Fi_0.1 program algorithm

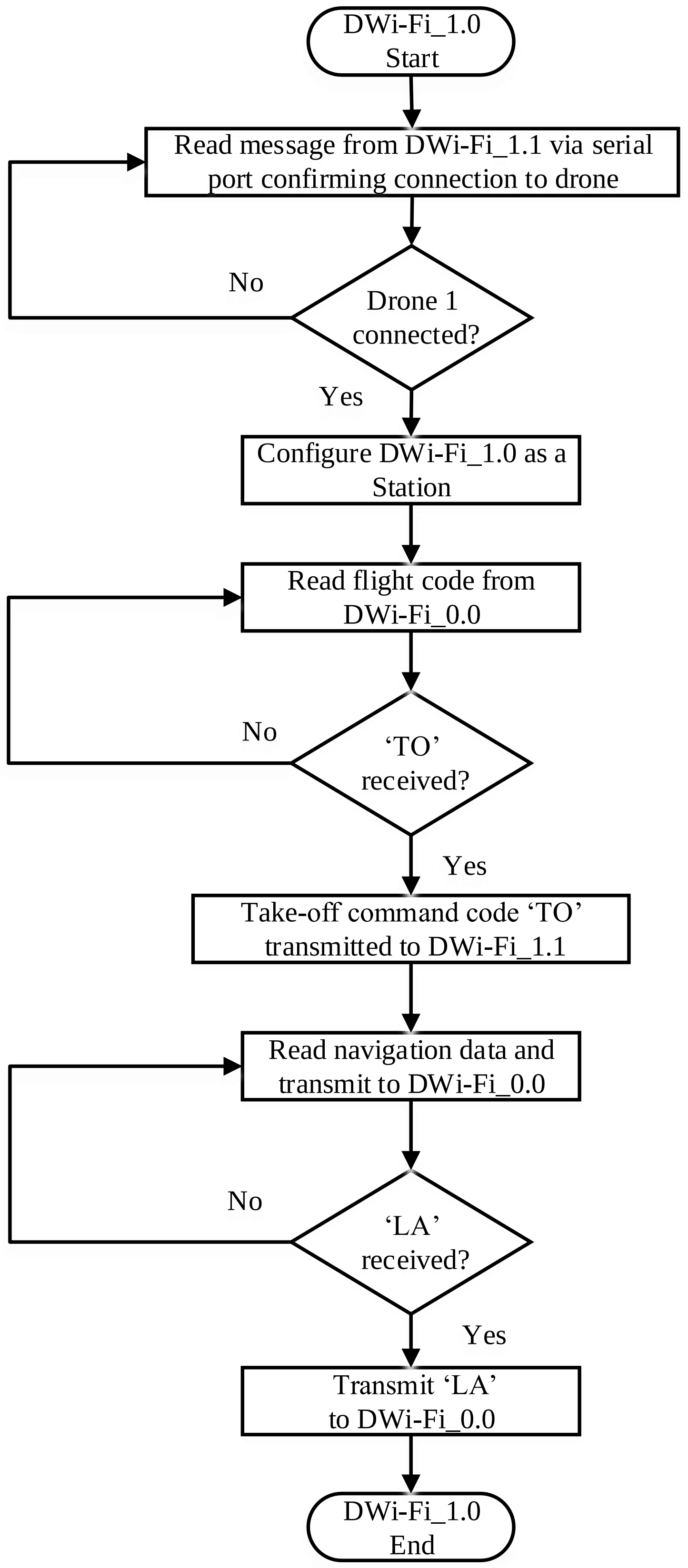

DWi-Fi_1.0
Start
Read message from DWi-Fi_1.1 via serial port confirming connection to drone
Drone 1 connected?
No
Yes
Configure DWi-Fi_1.0 as a Station
Read flight code from DWi-Fi_0.0
'TO' received?
No
Yes
Take-off command code 'TO' transmitted to DWi-Fi_1.1
Read navigation data and transmit to DWi-Fi_0.0
'LA' received?
No
Yes
Transmit 'LA' to DWi-Fi_0.0
DWi-Fi_1.0
End

Figure 5-5 Flowchart describing DWi-Fi_1.0 program algorithm

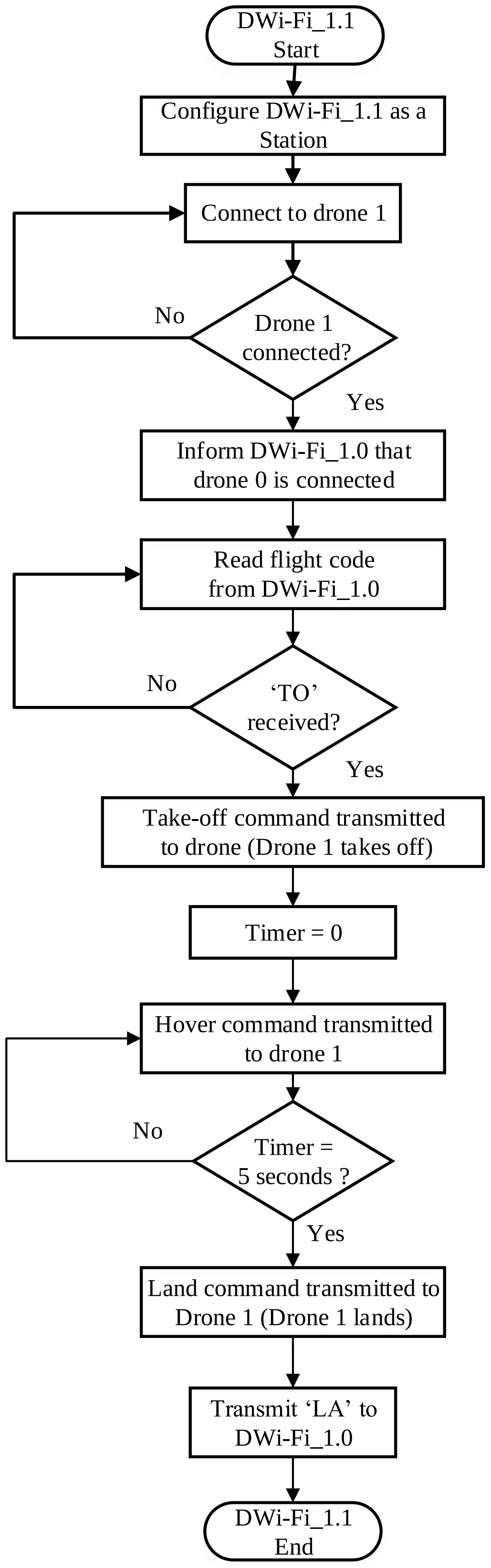

DWi-Fi_1.1 Start
Configure DWi-Fi_1.1 as a Station
Connect to drone 1
Drone 1 connected?
No
Yes
Inform DWi-Fi_1.0 that drone 0 is connected
Read flight code from DWi-Fi_1.0
'TO' received?
No
Yes
Take-off command transmitted to drone (Drone 1 takes off)
Timer = 0
Hover command transmitted to drone 1
Timer = 5 seconds ?
No
Yes
Land command transmitted to Drone 1 (Drone 1 lands)
Transmit 'LA' to DWi-Fi_1.0
DWi-Fi_1.1 End

Figure 5-6 Flowchart describing DWi-Fi_1.1 program algorithm

5.4 Results/Discussion

The programs described in the flowcharts of the previous section are uploaded to the respective Wi-Fi modules. Power from the on board lithium batteries is applied to the two drones and to the Wi-Fi modules. A few seconds is required for the Wi-Fi networks to be established, before the rotors of both drones begin to rotate, and the drones leave the ground. After hovering for approximately five seconds, the drones land. A plot of the height and duration of hover for both drones is depicted in Figure 5-7.

The graph of Figure 5-7 shows the two drones taking off, climbing to approximately 73cm, hovering for approximately 3.5 seconds, and then landing. Although instructed to hover for 5 seconds the hover function actually begins execution before the drone takes off as described in Chapter 4. The graph of Figure 5-7 shows a clear latency of approximately 0.2 seconds between the drones during take-off. This result is due to the structure of the controlling programs. Drone 0 is instructed to take off before transmitting the take-off code to drone 1. Similarly Drone 1 is instructed to land before transmitting the land code to drone 0. In both cases, the transmission of the command code across the Wi-Fi network, the interpretation of the command code, the transmission of the code via

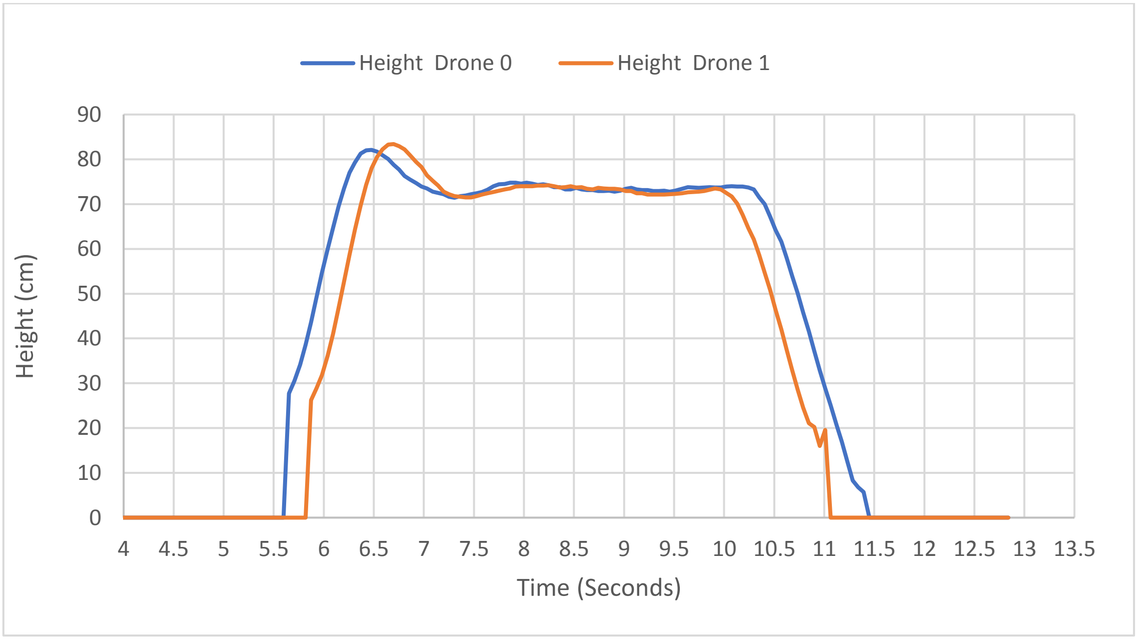
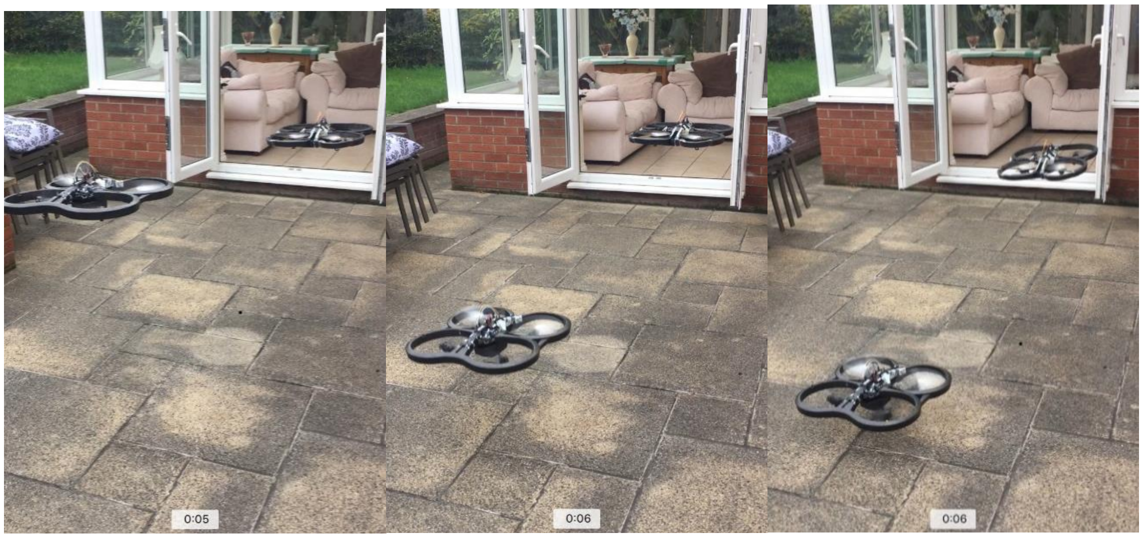


Figure 5-7 Graph showing take-off, height, and landing of two networked drones

the serial port to DWi-Fi_x.1, and finally the interpretation of the code by the drone incurs the 0.2 second delay between both drones during take-off and landing.

The photographs of Figure 5-8 show three shots in time during the two drone networked flight, where drone 0 is at the rear and drone 1 at the front. In Figure 5-8(a) the shot captures both drones during the hover period. Figure 5-8(b) shows a shot at approximately 10.25 seconds into the flight time when drone 1 has begun to descend but drone 0 is still hovering. Finally, Figure 5-8(c) shows the time in the process at approximately 10.75 seconds, when drone 1 has almost landed and drone 0 is commencing decent.



(a) (b) (c)

Figure 5-8 Photographs showing the two drones in mid flight (a) both drones hovering, (b) leading drone, drone 1 commences landing, (c) drone 1 almost landed, drone 0 commencing landing.

5.5 Summary

The hardware of the autonomous flying drone, developed in Chapter 4, provides the platform for a dual-drone system. The DWi-Fi_x.0 Wi-Fi modules, strapped to the drones, enable drones to form a two drone network. Although the discussion in this chapter largely describes the formation of a two drone network, the algorithms have been constructed to be expandable. Chapter 6 develops the algorithms further, to enable additional drones in range of the access point drone, to join the network.

Two or three character flight codes communicated over the network control the flight of the two drones. Although the example discusses the two drones following a basic flight plan of take-off, hover, and land, any combination of the available flight codes, can be communicated across the network to follow any required 3-dimensional flight plan. However, since there is currently no method for the drones to determine their relative position, flying more complex flight plans incurs the potential of flight collision.

A network discovery algorithm, enabling multiple station drones to form a drone network, and a method to enable the networked drones to calculate their relative positions, prevent random collisions, and allow the potential for formation flying, is presented in Chapter 6.

Chapter 6

Drone Node Discovery Algorithm and Drone Position Algorithm

6.1 Introduction

The dual UAV implementation of Chapter 5 is enhanced in this chapter by enabling the possibility of multi drone network formation. A drone discovery algorithm is developed to enable any station drone in the range of the access point drone to make a connection and join the drone network. The drone discovery algorithm is tested and results presented for analysis.

A multi drone network requires that each drone is aware of its position to avoid collision. The received signal strength indicator, RSSI, is a variable, inherently available to all stations on the Wi-Fi network and is used to determine the distance between the access point drone and the station drone. Results of the distances determined by the RSSI measurements between stationary access point and stations drones, and flying access point and station drones are compared for accuracy and analysis.

Finally, using the distance information provided by the RSSI data, an algorithm to determine the position of the station drones with respect to the access point drone as a set of Cartesian co-ordinates is proposed.

6.2 Current Methods of Node Discovery and Drone Positioning

A number of node discovery algorithms and drone positioning algorithms currently exist. This following subsections discuss a number of these algorithms and explain why the development of new algorithms is appropriate for this research.

6.2.1 Node Discovery Algorithms

To establish a drone network, a drone discovery algorithm is required. The drone discovery algorithm should enable all drones with the appropriate configuration to connect and form a network. A number of discovery algorithms have been discussed and implemented in published research however the deployment of a particular algorithm is largely dependent upon the network application and the number of nodes in the network concerned. A paper published by Konwar *et al.,* discuss a method of node discovery such that only nodes with mutual association form a network. This is achieved by the nodes sharing knowledge by transmitting and receiving gossip messages [93]. Dyo and Mascolo discuss an efficient node discovery in mobile sensor networks by ensuring nodes are awake when they are likely to encounter other nodes [94] . An energy efficient neighbour discovery protocol is discussed by Kohvakka *et al.,* which reduces the power consumed by nodes during the discovery phase [95]. The method of drone discovery in this research will follow a method similar to that of Konwar *et al.*. In this research however, the messages communicated between access point and potential nodes are minimised to ensure a speedy network formation, and also to ensure all non-drone nodes are discarded.

6.2.2 Drone Positioning Algorithms

A multi drone network requires that each drone is aware of its position to avoid collision. For an indoor network of drones, GPS cannot be used for positioning, and therefore an alternative method is required. Mustafah *et al.* propose a method of positioning using two video cameras The cameras are not mounted on the drone but are able to calculate the drones' position, and communicate this information to the drone via a Wi-Fi communication link. They claim reasonable results but the method would not be appropriate for a multi drone system if a pair of stereo cameras are required for each drone [96]. The method of drone positioning proposed by Mustafah *et al*., is further developed by Jin *et al.*, but now with the two cameras mounted on the drone. After take-off the drone maintains a constant height and uses the cameras to scan and determine its position. The method is more of a theoretical proposal and there are no actual results of the proposed method presented in the published paper [97]. A different approach is implemented by Bahiki *et al.*, who suggest a method of indoor positioning by combining infra-red and ultrasonic ranging sensors. The presented results enable relative drone position to be estimated but not their absolute position [98]. Furthermore, additional hardware is required to be mounted on the drone. A method of drone positioning is required in this research which does not require any additional hardware to be mounted on the drone.

6.3 Drone Node Discovery Algorithm

The drone node discovery algorithm effectively establishes the drone network enabling a number of drones configured as stations to connect to the drone configured as an access point. The hardware configuration is essentially as depicted in Figure 5-1 but with the possibility of additional station drones, having the ability to connect to the drone configured as an access point, thus establishing a multi drone network.

The C program uploaded to every DWi-Fi_x.0 network module configured as a station is identical. Additional drones can therefore be added to the network without the requirement to make any change to the network program. A C-program structure of type struct_drone_info is created within each DWi-Fi_x.0 module as described in Table 6-1.

The structure contains essential information pertinent to each station drone on the network. The first value within the structure is the drone number which is assigned by the access point drone during the drone discovery process. The Received Signal Strength Indicator (RSSI) value is the second value stored in the structure and is the most recently read RSSI value between station and access point.

Table 6-1 C program structure created within each DWi-Fi_x.0 station drone

```
struct drone_info {int drone_no;
                   int  RSSI_val;
                   IPAddress ipC;
                   };
```

The final value stored in the structure is the value of the drones IP address allocated by the access point drone during the Dynamic Host Configuration Protocol (DHCP) process.

Prior to commencement of the discovery process all DWi-Fi_x.1 modules connect to their respective drones. The DWi-Fi_x.1 modules report to their DWi-Fi_x.0 modules confirming that the connection is made. The DWi-Fi_0.0 module then configures itself as an access point with an IP address of 192.168.04.01. All other DWi-Fi_x.0 network modules configure themselves as stations and attempt to connect to the access point. A node connection time limit within the program of approximately 15 seconds is introduced to enable all network modules to connect to the access point. The function, WiFi.softAPgetStationNum(), running within the DWi-Fi_0.0 module, returns the number of current connections. This function is executed continuously within the 15 second connection time and the final assigned variable becomes the number of station nodes connected to the network. The access point module allocates IP addresses of 192.168.4.02, 192.168.4.03, 192.168.4.04, 192.168.4.05 to the connected nodes. The network thus consists of an access point drone, a number of station drones and a laptop node. The access point drone allocates itself as ‘Drone 0’ and allocates a drone number to each additional station drone. The drone number is important as this number determines the position that the drone will ultimately take within the drone formation. The drone formation and associated drone numbers are illustrated in Figure 6-1.

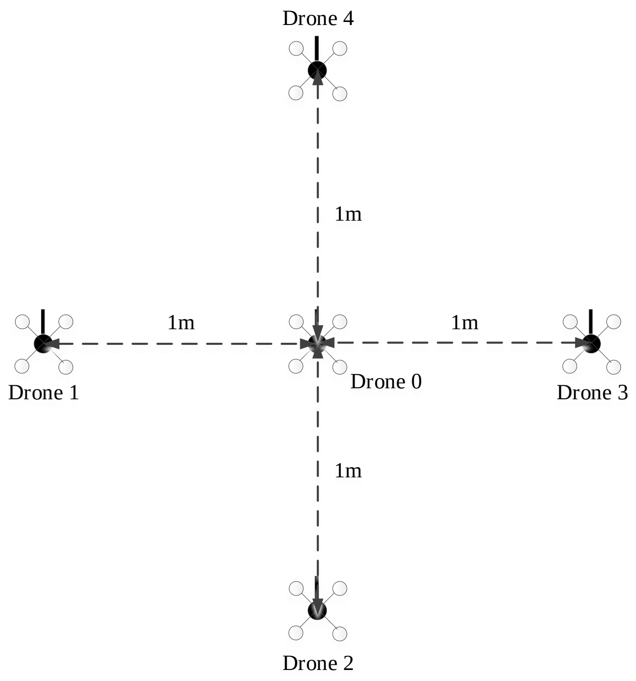


Figure 6-1 Drone number and associated drone formation position.

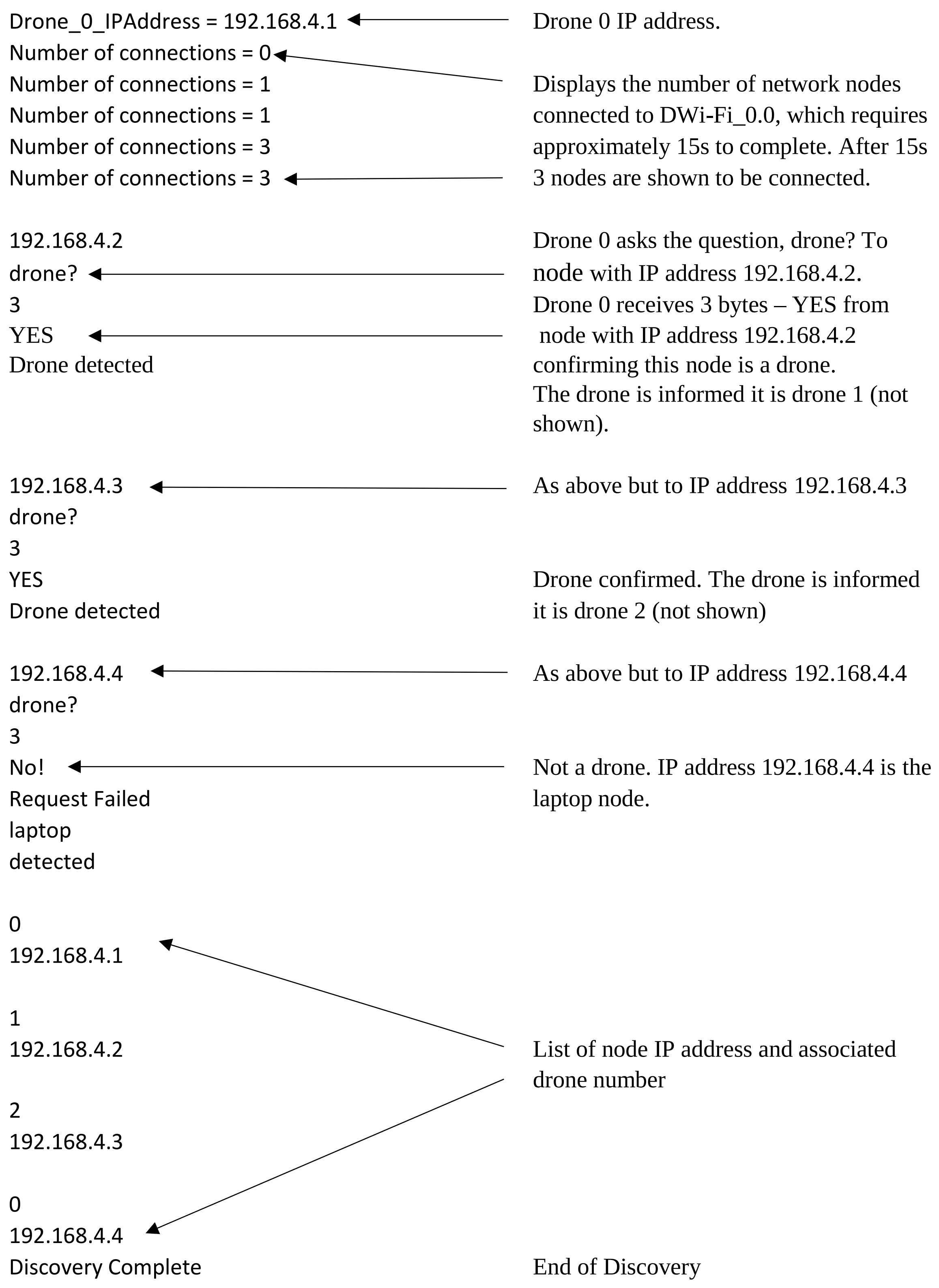
The laptop node is allocated with an IP address but should not be allocated a drone number. The access point node however does not know which IP address has been allocated to the laptop node. To ensure therefore that the laptop does not receive a drone number the access point node cycles through each allocated IP address transmitting the question, drone? If the current node receiving the question is a drone then it responds ‘Yes’ and the access point allocates the drone with a drone number. If the current node responds with a ‘No’ (i.e. it is the laptop node) the node is not numbered. All valid drones on the network are thus numbered drone 1, drone 2, drone 3, and drone 4.

6.3.1 Drone Discovery Algorithm Results

The discovery algorithm is examined with an access point drone (Drone 0), two station drones, and a laptop node. In order to examine network activity the drone 0 network node is connected to the laptop enabling network activity to be printed to the laptop screen. The laptop node for test purposes is powered by a 9V battery. Results of the execution of the discovery algorithm with respect to Drone 0 are depicted in Figure 6-2.

a. Initially ‘Drone 0 Connected’ is printed to the laptop screen indicating that DWi_Fi_0.0 has received a message from DWi_Fi_0.1, via the serial port connection, informing DWi_Fi_0.0 that DWi_Fi_0.1 is successfully connected to the drone AP.
b. DWi_Fi_0.0 IP address 192.168.4.1 is printed to the screen.
c. The next five lines illustrate the station connections. The section of code that generates this output requires approximately 15 seconds to complete to provide time for the station nodes to connect to the access point. The last line of this section indicates that three station nodes are connected to the access point as expected (two drone stations and the laptop station).
d. At this stage, although IP address have been allocated to the station nodes, it is not known which nodes are drones and which node is the laptop. DWi_Fi_0.0 thus transmits the question ‘drone?’ to IP address 192.168.4.2 and awaits a response. In this example DWi_Fi_0.0 receives three bytes, ‘YES’ from node IP address 192.168.4.2 indicating that this node is a drone. DWi_Fi_0.0 transmits ‘1’ to IP address 192.168.4.2 informing this node that it is drone 1. This process is repeated for the remaining nodes detected on the network.
e. When the question is asked of node with IP address 192.168.4.4 it responds with the answer ‘No’. The laptop has been discovered.
f. Finally, when all nodes have responded to the ‘drone’ question, all IP addresses of discovered nodes and their associated drone numbers are listed on the laptop screen. The discovery program is now complete.

The results show that the implementation of the discovery algorithm detects and numbers all drone nodes as required. The IP addresses and associated drone numbers for all drones on the network are stored in a structure array within DWi_Fi_0.0 for later use. It is noted also that the results shown in Figure 6-2 are variable i.e. the drone nodes and laptop could be allocated different IP addresses and node numbers for subsequent execution of the discovery program.



```
Drone_0_IPAddress = 192.168.4.1
Number of connections = 0
Number of connections = 1
Number of connections = 1
Number of connections = 3
Number of connections = 3

192.168.4.2
drone?
3
YES
Drone detected

192.168.4.3
drone?
3
YES
Drone detected

192.168.4.4
drone?
3
No!
Request Failed
laptop
detected

0
192.168.4.1

1
192.168.4.2

2
192.168.4.3

0
192.168.4.4
Discovery Complete
```

Drone 0 IP address.

Displays the number of network nodes connected to DWi-Fi_0.0, which requires approximately 15s to complete. After 15s 3 nodes are shown to be connected.

Drone 0 asks the question, drone? To node with IP address 192.168.4.2.
Drone 0 receives 3 bytes – YES from node with IP address 192.168.4.2 confirming this node is a drone.
The drone is informed it is drone 1 (not shown).

As above but to IP address 192.168.4.3

Drone confirmed. The drone is informed it is drone 2 (not shown)

As above but to IP address 192.168.4.4

Not a drone. IP address 192.168.4.4 is the laptop node.

List of node IP address and associated drone number

End of Discovery

Figure 6-2 Results of discovery algorithm execution

6.4 Drone Position Algorithm

The aim of the drone position algorithm is to determine the position of each drone on the network as a set of x,y co-ordinates. Drone 0 is in the reference position at the origin, and the distance between drone 0 and any other drone on the network is determined by the value of the RSSI (Received Signal Strength Indicator) between drone 0 and the respective drone. Once the position of each drone is known, they should fly in turn to the correct formation position depending on the drone number as depicted in Figure 6-1. The first requirement is to establish the relationship between the RSSI value and the distance between drone 0 and a drone on the network which is established in the next section.

6.4.1 The RSSI Value as a Measure of Distance on a Wi-Fi Network

The RSSI value is a measure of the power of a signal at the receiving node. The unit of RSSI is dBm, a dimensionless unit which uses 1mW as the reference power level. The RSSI value is negative and moves closer to 0 as the receiver approaches the signal source. Distance measurement between two nodes within an indoor environment can prove challenging due to multipath effects.

The recognised equation for RSSI distance measurement is described by Equation 6-1 below [99]. Known as the log-normal shadowing model, Equation 6-1 is suitable for indoor and outdoor environments and includes a number of parameters to account for varying environmental conditions [100]

$$P(d) = X_\sigma + P_0 - 10\eta_p \log(d/d_0) \qquad \text{(Equation 6-1)}$$

In Equation 6-1 $P(d)$ is the received power in dBm at distance d. P_0 is the received power at a short distance d_0, where d_0 is typically 1m. η_p is the path loss exponent, has a typical value of between 2 and 4, and is dependent upon environmental conditions such as room size and room contents. X_σ is a Gaussian random variable with zero mean which is used to describe the Gaussian interference on the received signal. A large X_σ value is derived from multipath influences however Ladha *et al*. suggest that this can be reduced significantly by simply taking an average from a number of sample RSSI readings [99]. Ladha *et al.* also suggest a method to calculate the path loss exponent η_{p}, however the method relies on calculating angles between a node and known reference points which would be variable within differing environments and is therefore considered not to be valid for this research [99].

Capriglione *et al.* explore the challenges of distance measurement using RSSI within an indoor environment. The conclusions from their results are listed below.

(i) Multipath fading and shadowing of the Wi-Fi channel due to the size and shape and the presence of walls and obstacles within the indoor environment.

(ii) Variability of the signal power at the transmitter. Transmitted signal power can vary from similar transmitters resulting in differing power signal strength received at the receiver causing error.

(iii) Sensitivity at the receiver. The sensitivity of similar receivers can vary, resulting in different RSSI values being recorded for a common distance creating an error in the distance measurement.

(iv) The orientation of the antenna. Antennae have their own radiation pattern which is generally not orthogonal. The RSSI value at the receiver can thus vary for the same distance between transmitter and receiver dependent on their mutual orientation.

The conclusions of Capriglione *et al*. provide a useful insight into the potential source of error when attempting to measure distance from RSSI values [101].

Xu *et al*. explore the impact of environmental temperature on the RSSI value. They report a decrease in the RSSI value as the temperature increases. Specifically, an increase in temperature from 22° to 34° saw an average decrease in RSSI of 7.7% when the distance between transmitter and receiver is set at 5m and a decrease in RSSI of 6.2% when a distance between transmitter and receiver is set at 10m [102]. Research by Luomala and Hakala and also Sabu *et al.*, attribute the decrease in RSSI with increasing temperature to the effect on the electronic components within transmitter and receiver as the temperature rises. Essentially, the mobility of charged carriers within CMOS transistors which are used in transmitter and receiver circuitry, reduces with increasing temperature. This reduction in mobility impacts on the performance of the circuitry, resulting in a decrease in RSSI as the temperature is increased [103] [104].

Xu *et al*. also explore how the height of the sensor nodes, the influence of antenna type, and the influence of the human body impact on the RSSI value.

With regard to the height of the receiver, RSSI readings are taken with a receiving node at 135cm from the ground and at 6cm from the ground with a 5m distance between transmitter and receiver. An average increase of 8.5% in the RSSI value is reported at the greater height of 135cm. With a 10m distance between transmitter and receiver the average RSSI reading increases by 5.9% at the greater distance of 135cm from the ground. They conclude that multipath effects impact on the RSSI value when the receiver is closer to the ground.

A helix antenna is reported to provide higher values of RSSI, and RSSI values with less variation than a patch antenna when RSSI values are measured with the different antennae over the same distance.

In examining the influence of the human body on RSSI, the receiving sensor node is worn on the human body. Although the procedure of the test is not clearly defined or explained they report a 3.7% fall in RSSI at 5m and a 2.9% fall in RSSI at 10m due to the influence of the human body [102].

Research by Ivanic and Mezei, investigates the RSSI and distance relationship both indoors and outdoors with the receiving sensor oriented at differing angles. RSSI readings are taken as the receiving node is rotated at 45° intervals from 0° to 315° at increasing distances from 0.5m to 6m. Results for the outdoor measurement depict an expected logarithmic decay as the distance between transmitter and receiver is increased however variations in RSSI are observed as the angle of the receiver is varied with a worst case difference of -10dBm between RSSI values recorded at 0° and 90°. Results taken indoors depict a significant dispersion in the RSSI values. The expected logarithmic decay is not clearly defined as the distance between transmitter and receiver is increased. Significant variation in RSSI values for varying angles is also observed. The variation between the results taken indoors and the results taken outdoors is attributed to multipath effects impacting on the indoor measurements. What is clear from the results is the effect of the orientation of the receiving node, which from the results presented, significantly impacts on the RSSI values obtained over the range of distances measured [105].

Barai *et al.* describe a study using the very same NodeMCU modules utilised in this thesis. They program one NodeMCU as an access point and the second NodeMCU as a station and measure the RSSI value with respect to the distance between the two modules. They take 300 RSSI samples at distances from 0.3m to 10m and incorporate a curve fitting technique to generate an equation to estimate the distance. They conclude that their technique proves a minimum distance accuracy estimation of 91.68% [106]. Unfortunately there is very little detail of the experimental configuration employed other than that the experiment is carried out on the roof top of a two story building. Information regarding the height of the NodeMCU modules, the orientation of the modules, the area of the roof top, or the presence of any rooftop furniture, all of which can impact the RSSI values read by the receiving node, is not provided.

6.4.2 RSSI Distance Measurement

To establish the distance between a Wi-Fi transmitter and a Wi-Fi receiver utilising the RSSI value obtained at the receiving node, conditions mitigating the accuracy of RSSI measurements described in the previous section are considered as discussed below.

i) To minimise multipath effects measurements of RSSI values are taken indoors in a large gymnasium measuring 15m x 25m.

ii) To reduce the effects of the X_σ Gaussian random variable of Equation 6-1, ten RSSI readings are taken at each measurement position and the average value calculated.

iii) The height of the transmitter and receiver are fixed at 1.2m above the ground to mitigate ground multipath effects.

iv) The radiation pattern for the Wi-Fi modules is determined by taking RSSI measurements at 45° intervals from 0° to 315°. Whilst varying the angle, the orientation of the Wi-Fi modules remains constant i.e. both transmitter and receiver point to 0°.

v) RSSI measurements are taken at 3m, 6m and 12m at each angle as described in iv).

RSSI values are taken at distance intervals of 3m, 6m and 12m to provide a good demarcation between RSSI measurements. To establish the radiation pattern, RSSI values are measured at angles from 0° to 315° at 45° intervals between the transmitter and receiver at each measured distance. The resulting radiation pattern is depicted Figure 6-3.

The radiation pattern of Figure 6-3 displays results with similar patterns of RSSI against angle, measured over the distances measured. As expected, a clear demarcation between the radiation patterns is also observed with increasing RSSI values at the differing angles as the distance between transmitter and receiver is increased.

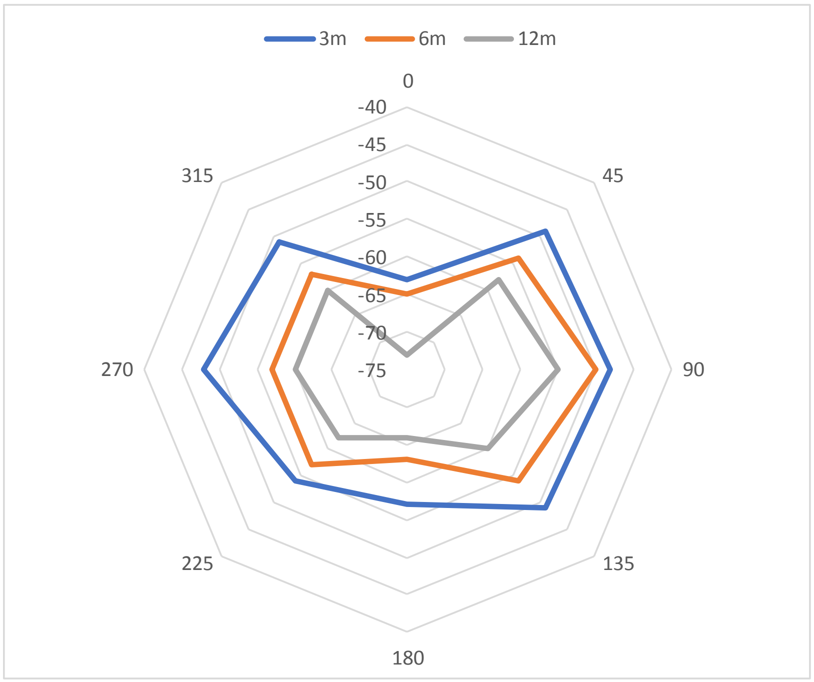


Figure 6-3 Radiation pattern measured at intervals of 3m, 6m, and 12m.

The radiation patterns result from the relative orientation between the antennae of the transmitter and receiver. Figure 6-2 shows that the largest RSSI values are obtained at angles of 90°and 270° when the NodeMCU modules are oriented side by side. The NodeMCU antenna is mounted on the top left of the module. The greatest RSSI value may therefore be expected to be received when the transmitter and receiver are side by side for a particular distance.

The graph of Figure 6-4 depicts the measured RSSI values plotted against distance at varying angles. The RSSI value is observed to decrease as the distance increases as expected. Results taken at 0° and at 180° show the lowest values of RSSI when the transmitter and receiver NodeMCU modules are positioned directly behind and in front of each other which is the orientation providing the weakest signal reception. There is however a considerable variation in the RSSI value depending upon the angle of measurement. At 90° and 270°, at a distance of 3m for example, an RSSI value of -48dBm is measured, whereas at 0° at 3m, an RSSI value of -63dBm is measured. This is a difference of 15dB for a measurement at the same distance, only differing in the orientation of the measurement.

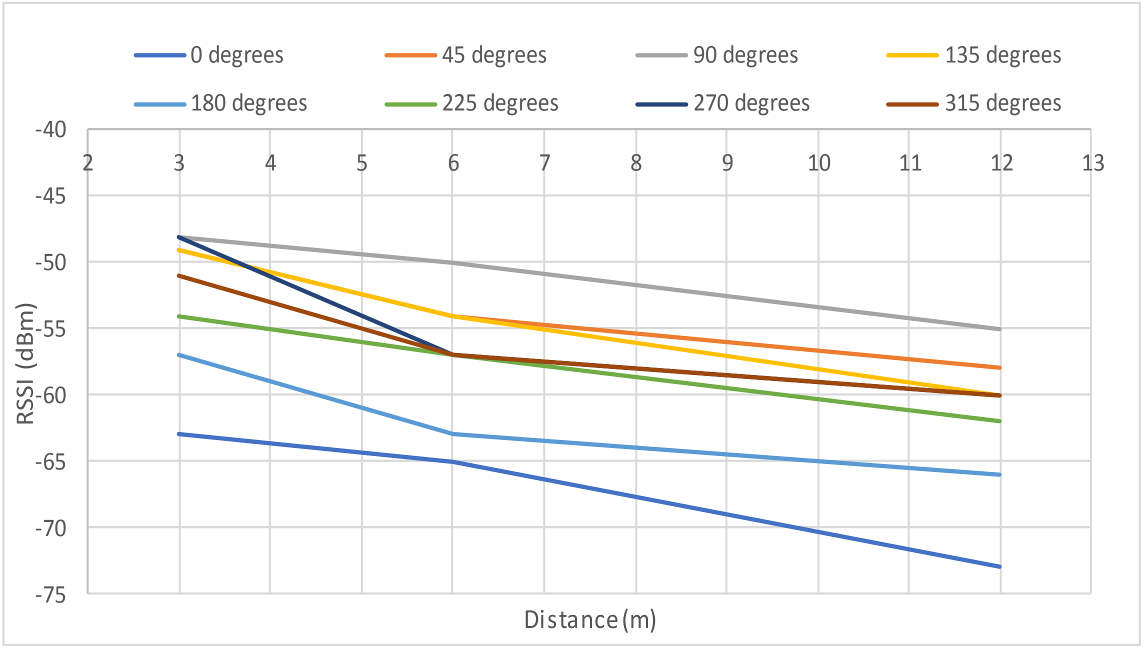


Figure 6-4 Graph of RSSI (dBm) values plotted against distance (m)

The results however, do show an approximate linear response for decreasing RSSI as the distance increases for the varying angles measured. An important factor in the determination of the distance between transmitter and receiver from the RSSI value.

6.4.3 RSSI Measurements Taken From Flying Drones

The RSSI measurements in the previous section are taken from two static drones where the access point drone is the reference point and remains stationary and the station drone is manually moved the incremental measurement distances. It is important, now that distances can be determined from the RSSI value provided the angle is known, that measurements are taken from flying drones. It is possible that the rotating motors driving the propellers on the drone, could impact on the integrity of the Wi-Fi signal with induced harmonics. Tests are therefore carried out with two flying drones. The access point drone and the station drone begin two metres apart oriented at 0° i.e. the access point drone is positioned directly behind the station drone both pointed at 0°. Both drones take-off and climb to 1.2m before the station drone (drone1) flies forward 12m, hovers for 10s and then flies backwards 12m to the starting position. RSSI measurements are taken by the station drone during the 12m forward flight and the 12m flight in the reverse direction. Results of the test flight are depicted in Figure 6-5. The distance travelled by the drone follows the expected path, flying 12m in the forward direction, hovering for 10s and then flying 12m in reverse to the drone’s original position. Photographs of the two drone flight to measure the RSSI are depicted in Figure 6-6, Figure 6-7, Figure 6-8 and Figure 6-9.

During the flight the RSSI value read by the station drone is observed to fluctuate considerably but more so on the outward journey rather than on the inward journey. This is due to the pitch of the drone. When flying forwards the NodeMCU module DWi-Fi_1.0 is obscured by the drone itself as the drone pitches forward. On the return journey when the drone pitches backwards there is a clear line of sight from the receiver to the transmitter. The results of the shape of the RSSI graph yield expected results. As the drone flies the outward stage of the journey the RSSI value is seen to fall and on the return stage of the journey the RSSI value is seen to increase back to the starting RSSI value as the return journey is completed.

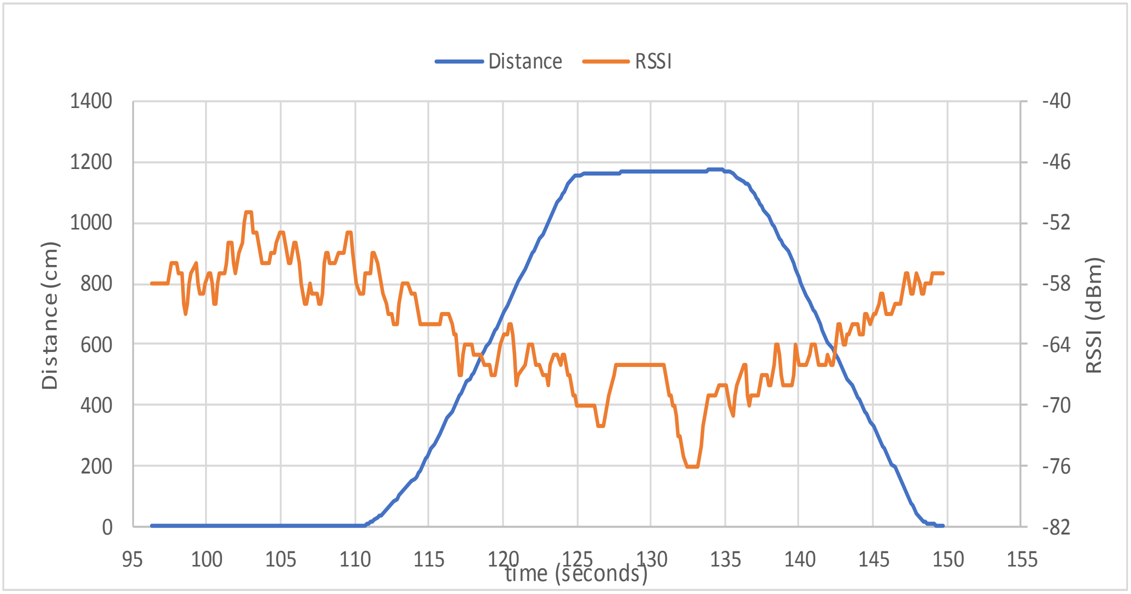


Figure 6-5 Graph of distance travelled (cm) and RSSI (dBm) plotted against time

Unexpected results are only obtained when the drone is hovering for 10s after completing the outward journey. The RSSI value is observed to fluctuate drastically between -66dBm and -76dBm, a variation of 10dB. At this time in the flight the drone was very close to the internal wall and this fluctuation could be due to multipath effects.

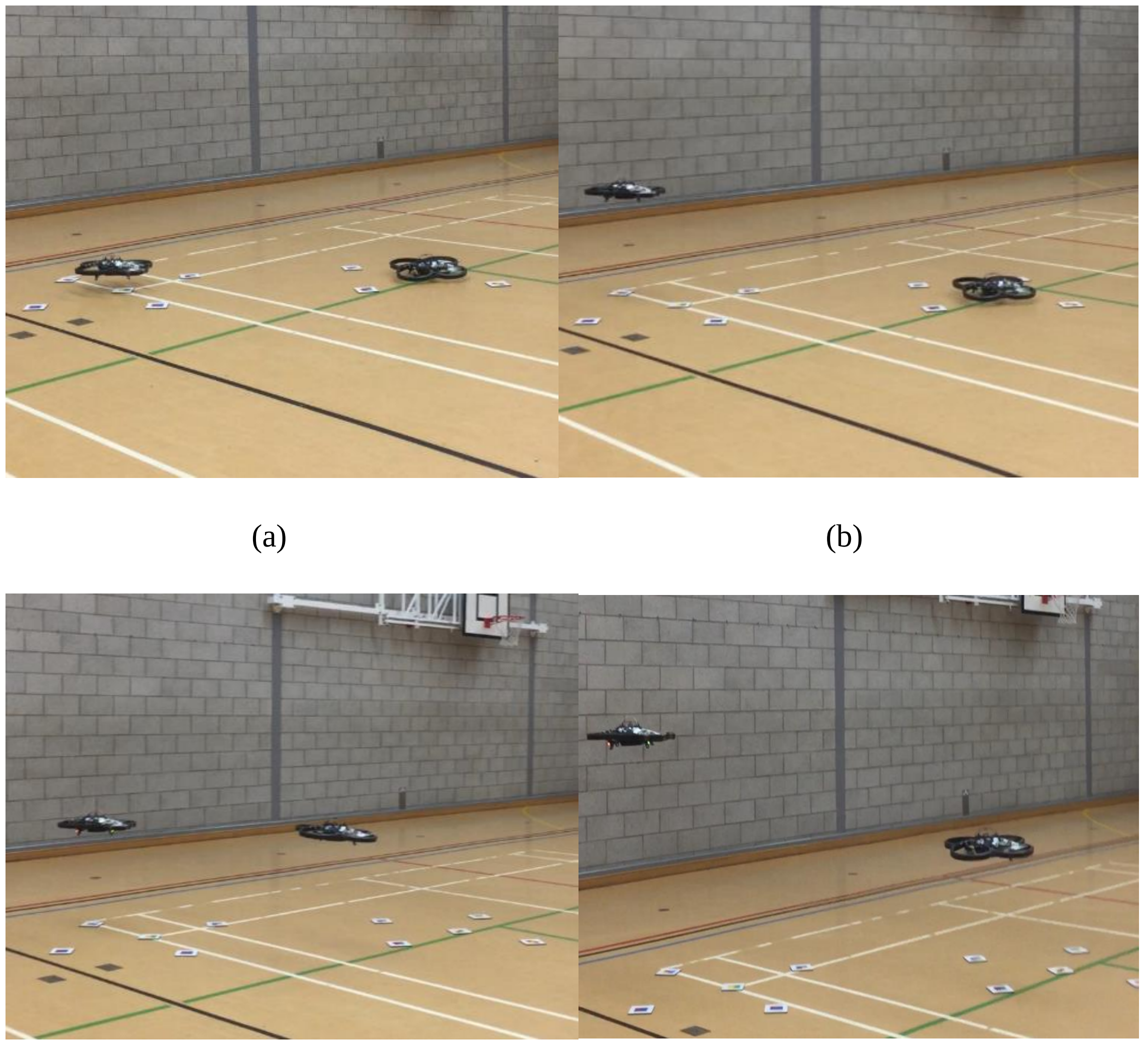
As a guide to the accuracy of the measurements, RSSI values from the drone flight are compared with the RSSI values measured with the standing drone from Figure 6-4 displayed in Table 6-2. The RSSI values correlate very well, with a worst case difference of 5dB, representing a distance of approximately 3m.

Table 6-2 Comparison of RSSI measurements from flying and stationary drone

Distance (m)	Standing drone RSSI dBm	Flying drone outward RSSI dBm	Flying drone inward RSSI dBm	Difference (Standing - Outward) dB	Difference (Standing- Inward) dB
3	-63	-62	-60	-1	-3
6	-65	-64	-63	-1	-2
12	-73	-68	-68	-5	-5

Good correlation is observed on the outward journey with the RSSI values only differing by 1dB. The inward journey is not so good with a worst case difference of 3dB. Taking into consideration the results obtained, it is possible that a measurement of RSSI can be used to determine the distance between transmitter and receiver and will be used in the algorithm to establish drone position proposed in the next section.

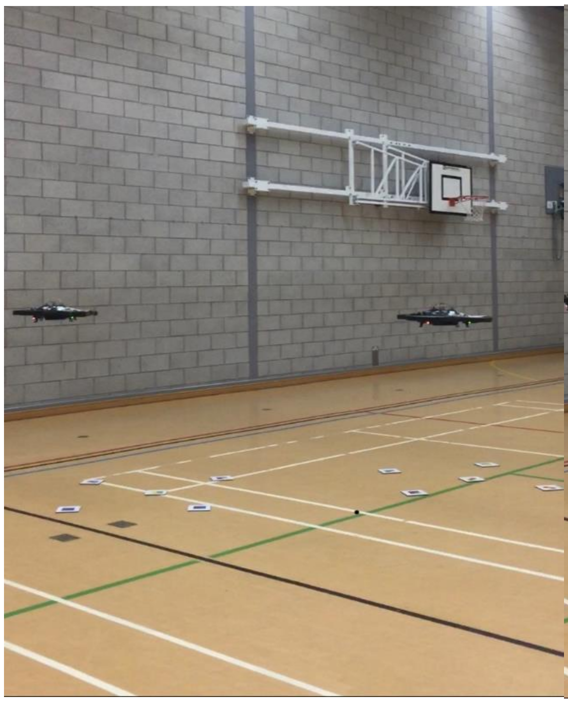
In the following photographs illustrating the flight of the two drones to capture the RSSI value, drone 0 is on the left of each photograph, and drone 1 is on the right of each photograph.



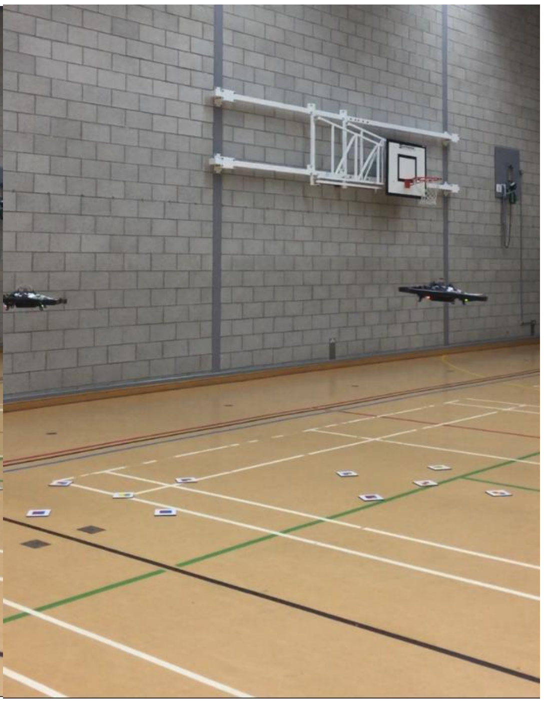
(a) (b)

(c) (d)

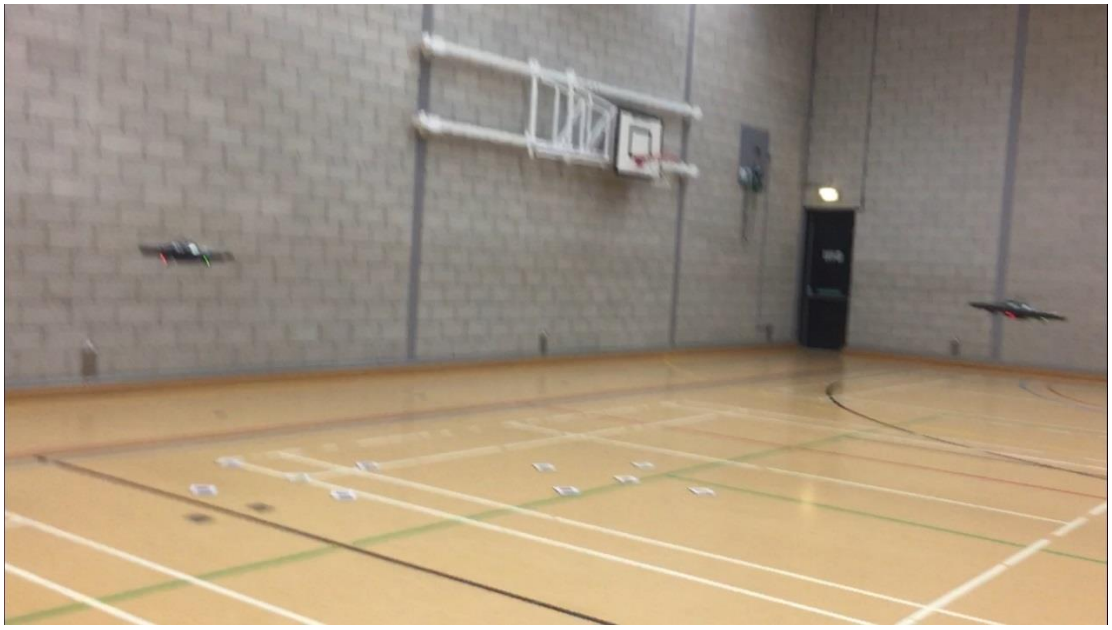
Figure 6-6 Two drone flight to capture RSSI 1 (a) drone 0 takes off, drone 1 on the ground, (b) drone 0 hovers at 0.75m and drone 1 on the ground, (c) drone 0 hovers at 0.75m and drone 1 hovering at 0.75m, (d) drone0 ascends to 1.2m, drone 1 hovering at 0.75m



(a)



(b)



(c)

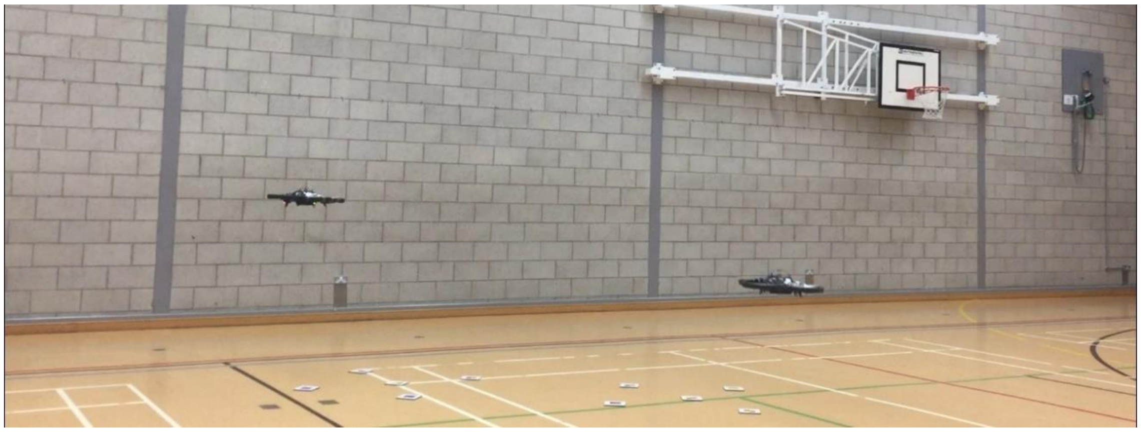
Figure 6-7 Two drone flight to capture RSSI 2 (a) Drone 0 and drone 1 hovering at 1.2m, (b) drone 1 begins to fly forward, (c) drone 1 continues its forward flight of 12m

(a)

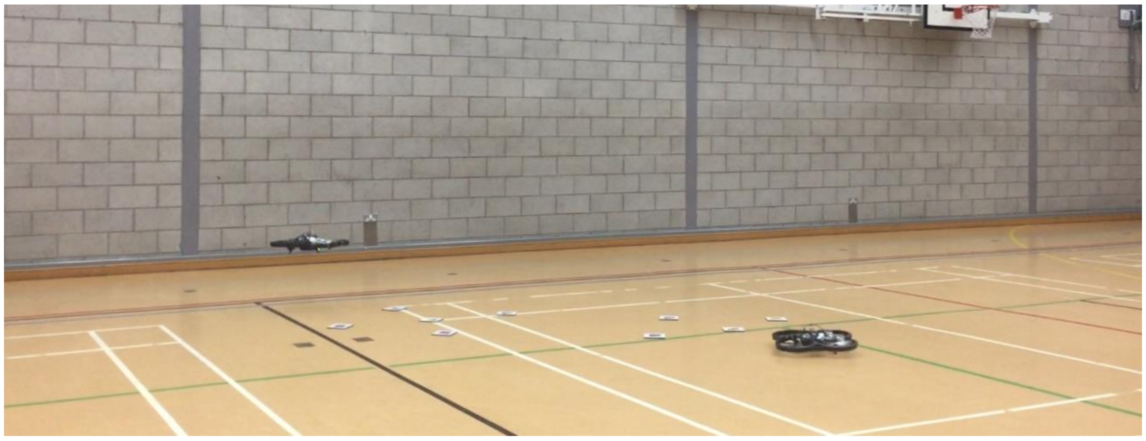
(b)

(c)

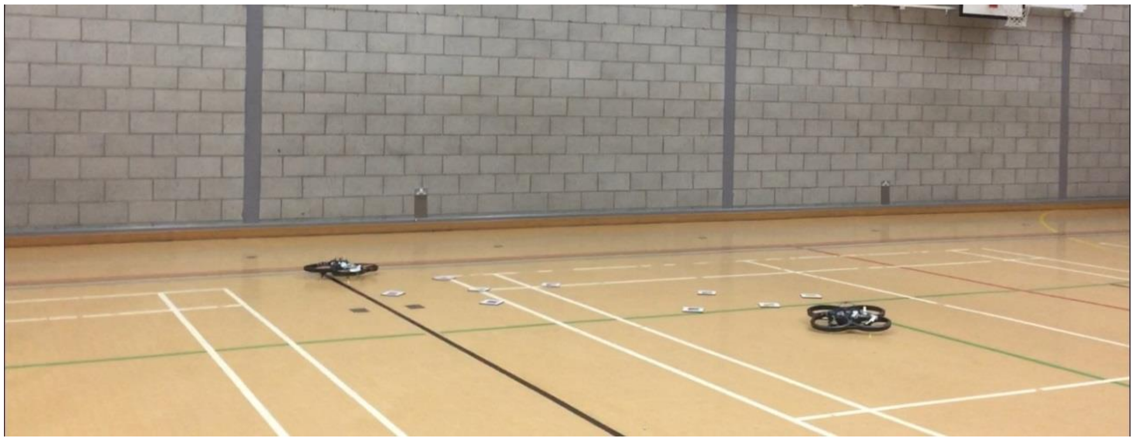
Figure 6-8 Two drone flight to capture RSSI 3 (a) drone 1 flies 12m and hovers for 10s (b) drone 1 flies backwards 12m, (c) drone 1 arrives at original position and hovers.



(a)



(b)



(c)

Figure 6-9 Two drone flight to capture RSSI 4 (a) drone 0 hovers, drone 1 commences landing, (b) drone 0 commences landing, drone 1 landed (c) drone 0 lands.

6.4.4 Proposed Drone Position Algorithm

A drone position algorithm is proposed to determine the position of the drones that have been detected on the network during the discovery process, as a set of Cartesian co-ordinates. Once the position of the drones has been determined then they can fly to the required location depending upon the allocated drone number as depicted in Figure 6-1.

The proposed drone position algorithm is completed in two phases. The first phase determines the quadrant on a Cartesian graph in which the drone is situated. The second phase then calculates the drone position as a pair of Cartesian co-ordinates within that quadrant.

Phase 1

Each of the quadrants are labelled as zones as depicted in Figure 6-10. Drone 0 is positioned at the origin, and drone 1, whose position is to be determined, is located within any one of the four zone regions.

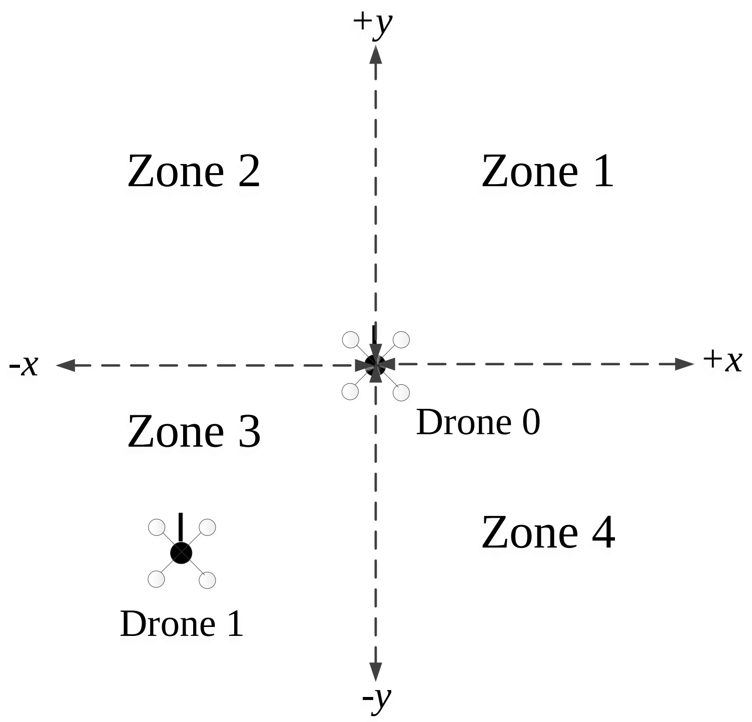


Figure 6-10 Zone quadrant labelling

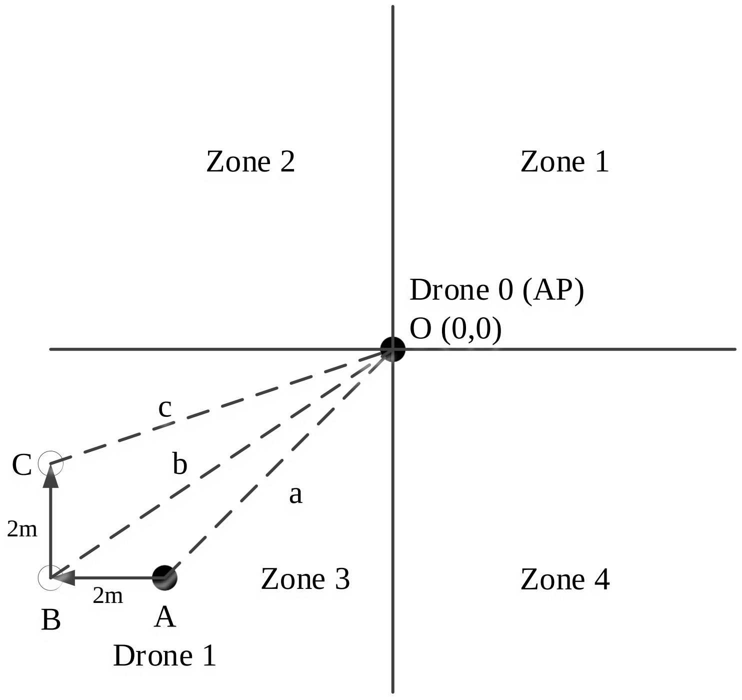


Figure 6-11 Diagram depicting drone position algorithm

Before execution of the drone positioning algorithm, drone 0 takes off and instructs drone 1 to do likewise. The drone positioning algorithm is executed entirely by drone 1 whilst drone 1 and drone 0 are airborne.

Once in the air, Drone 1 can take an RSSI reading but this alone cannot determine drone 1's position. A single RSSI reading provides a distance measurement *r*, which potentially positions drone 1 at any point on a sphere of radius *r* with respect to the origin. In order to determine drone 1's position drone 1 is required to move. The movement of drone 1 to determine its position is described in Figure 6-11. In the example shown in Figure 6-11 drone 1 is positioned at point A within Zone 3, and drone 0, the access point drone, is positioned at the origin, point O.

<u>Drone Movement to determine the Zone in which the drone is situated</u>

i) Drone 1 takes 10 RSSI readings at position A. The average of these 10 readings is recorded as the definitive RSSI, labelled RSSI_1, for drone 1 at position A.

ii) Drone 1 moves 2m to the left to position B and hovers for 5 seconds to brake and stabilise the drone and then hovers for a further 2 seconds. At the beginning of the two second hover period, whilst drone 1 is stationary, 10 RSSI readings are taken at position B. The average of these ten readings is recorded as the definitive RSSI reading for drone 1 at position B, labelled as RSSI_2.

iii) Finally Drone 1 moves forwards 2m to position C hovers for 5 seconds and then hovers for a further 2 seconds as before. At the beginning of the 2 second hover period whilst the drone is stationary the average of 10 RSSI readings is recorded as the definitive RSSI reading for drone 1 at position C, labelled as RSSI_3.

The three RSSI values recorded, RSSI_1, RSSI_2 and RSSI_3 are then utilised in determining the zone in which drone 1 is located. If the value of RSSI_1 taken at point A is greater than the value of RSSI_2 taken at point B as would be the case in the example of Figure 6-12, then drone 1 must be in the $-x$ region i.e. either zone 2 or zone 3.

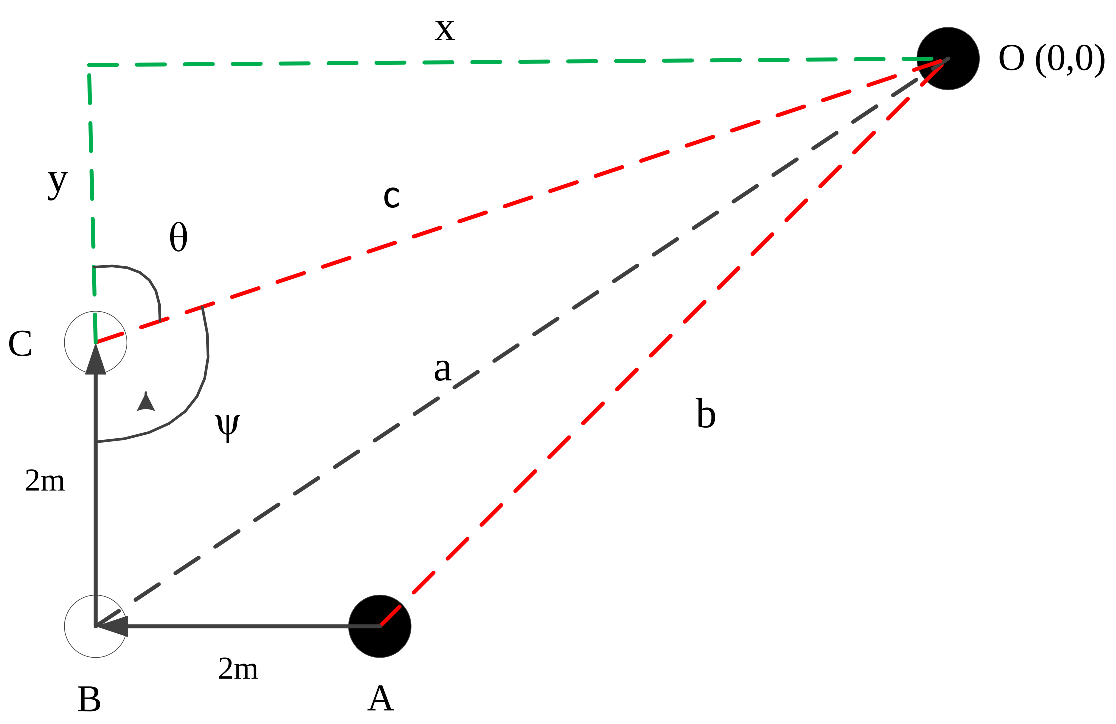


Figure 6-12 Diagram illustrating the angles required to determine the drone zone

Table 6-3 Table showing the comparisons of RSSI readings, and the resulting zone

Comparison 1	Comparison 2	Zone
RSSI_1 < RSSI_2	RSSI_2 > RSSI_3	1
RSSI_1 > RSSI_2	RSSI_2 > RSSI_3	2
RSSI_1 > RSSI_2	RSSI_2 < RSSI_3	3
RSSI_1 < RSSI_2	RSSI_2 < RSSI_3	4

If RSSI_2 is less than the value of RSSI_3 taken at point C as it is in the example, then drone 1 must also be in the $-y$ region i.e. either zone 1 or zone 2. By combining the results determined by the two comparisons, the zone in which the drone is positioned can be realised. In the example since RSSI_1 is greater than RSSI_2 and RSSI_3 is greater than RSSI_2 then drone 1 is located in zone 3. A table showing the comparisons of the distances and the resulting zone is shown in Table 6-3.

Phase 2

To improve the accuracy of the *x*-*y* position calculation, the distances a, and c of Figure 6-12 are calculated from the RSSI, distance graph of Figure 6-4 using the mid angle value i.e. either 45°, 135°, 215° or 315° depending upon the zone in which the drone is situated. Angle ψ can then be calculated knowing the lengths of the a, c and 2m triangle. Knowing angle ψ, angle θ can be determined and knowing distance c, the *x*–*y* position can finally be calculated. Drone 1, now aware of its x-y position, can fly to its required location, 1m to the left of drone 0 as illustrated in Figure 6-1. In the case of the example illustrated in Figure 6-12, drone 1 would fly a distance *y*m forwards, followed by a distance of *x*-1m to the right to find its correct location in the formation. Once drone 1 is in position, drone 0 instructs drone 2 to take off. Drone 2 then follows the same process as drone 1, in determining the zone in which it is situated, its *x*-*y* position, and finally flying to its formation position. Any additional drones connected to drone 0 during the discovery procedure continue the process as described until all drones are in position within the required formation.

6.5 Summary

The dual UAV network of Chapter 5 has been enhanced in this chapter to enable the formation of multi-drone networks. A discovery algorithm, which allows any station drone in range of an access point drone, to join the network of drones has been successfully developed and implemented. During the discovery process every station drone which finally connects to the network is provided with a unique drone number. This number provides each station drone the information it requires to find its position in the drone network formation.

In order to avoid collision, a requirement of a drone network is that each drone is aware of its own position. The received signal strength indicator RSSI is investigated as a possible measure of the distance between the access point drone and station drones. Tests completed at differing distances and orientation of Wi-Fi modules show that the RSSI value does fluctuate when measured for the same distance and orientation by as much as 6dB which corresponds to a distance of approximately 4m. This error can be reduced by averaging the RSSI values over a number of samples. Results show that the RSSI values can be used as a distance measurement particularly over shorter distances (between 2 and 12m) when the variation of RSSI as the distance increases is at its greatest.

An algorithm, utilising measured RSSI values is proposed to estimate the drone position. The algorithm involves performing a calculation using three RSSI measurements taken at three different locations after take-off. The algorithm can be experimented with to determine the optimum three locations providing the RSSI readings to generate the most accurate drone position.

The research completed and presented in this Chapter thus provides contributions to knowledge in the areas of drone discovery and network formation, autonomous drone control, and drone positioning within a drone network. A summary of the contributions to knowledge within these areas follows.

Drone Discovery and Network Formation

- A discovery algorithm is developed to discover all potential drone nodes within range.

  This is essential to ensure that all drone nodes that are within range are included in the network to be formed.
- An algorithm is developed to enable the unique numbering of all discovered drones.

  This unique number is vital to each drone so that it is aware of where it should appear within the network formation.
- The formation of a network allowing flight control commands to be transmitted between all discovered drones.

  Formation of the drone network is essential to enable flight commands to be transmitted between drones.

Autonomous Drone Control

- A hardware platform mounted on the drone to facilitate autonomous drone control and network capability.

  The hardware platform consisting of one NodeMCU module to control the autonomous flight of the drone, and a second NodeMCU module to enable network capability, is essential to enable autonomous control and drone network formation.
- Transmission and reception of developed flight command codes between drones, enabling any drone on the network to inform another drone on the network to follow a desired flight plan.

  The developed flight command codes and the algorithms to permit their transmission, are vital to ensure the required drone receives the correct flight command code.

- The development and implementation of the flight control algorithms are critical to enable autonomous flight control.

<u>Drone positioning</u>

- The measurement of RSSI values between two stationary (non flying) drones and relating these values to the distance between the two drones.
  In using the RSSI as a distance measurement, it is essential to establish the correlation between the RSSI value and the distance between drones.
- Measurements taken of RSSI values between two flying drones, as one drone flies a controlled distance away from the other and then returns back to its original position.
  It is essential that flying drones are able to take RSSI values between one another and the values are not compromised by harmonics induced by the drone rotors.
- The development of an algorithm to establish drone position.
  In order to avoid collision, and also to be able to complete a desired mission, it is vital that drones are aware of their own position and the position of other drones within the network. The proposed method of drone positioning has the potential to determine a drone's position within a drone network.

The research thus provides an essential platform for future development in establishing drone networks, autonomous drone control, and drone positioning within a drone network.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

This thesis presents the development of an autonomous quadcopter drone network. In developing an autonomous drone, flight commands are required to be executed by a device running a stored program, without human interaction. Flight control programs running on a lap top computer provide autonomy but not a practical solution. In this thesis a fully autonomous drone is implemented by strapping a microcontroller module (NodeMCU) containing flight path programs, to the drone. Upon execution of an example flight path program the drone has been shown to follow the flightpath to a minimum accuracy of 88%. An additional algorithm developed which estimates the drone flight path from the angular velocity of the four rotors (AVQR) enables the distance travelled for the particular flight plan under test to an accuracy of 95%.

Communication and transmission of flight instructions between a number of drones is achieved by creating a drone network. Drone network capability is established by the introduction of a second NodeMCU module which is able to connect to other drones with the same hardware configuration. Flight command codes can now be transmitted between drones to instruct an identified drone to follow a particular flightpath.

To avoid collision it is essential within a drone network that all drones are aware of their own position. The received signal strength indicator RSSI, between transmitter and receiver on the network, is introduced as a measure of the distance between drones and an algorithm utilising this distance calculation is proposed to enable drone position calculation.

This thesis thus presents all aspects required in the development of a quadcopter drone network.

The theory behind the flight of a quadcopter drone is presented in Chapter 2. An understanding of the theory behind drone flight is essential before embarking on a drone project. Justification of the project by considering a number of drone applications and in particular applications where a network of drones would potentially enhance the performance is also presented. Data transmitted between the Parrot AR2 drone and the controlling device is presented and interpreted. This data, consisting of commands transmitted to the drone, and navigation data transmitted by the drone is essential in the flight control algorithms developed in Chapter 3.

The development of the autonomous drone is discussed in Chapter 3. The autonomous drone is initially developed utilising the software development kit (SDK) made available from the manufacturer Parrot. The SDK implements thread programming to enable quadcopter flight control. The user is able to develop their own threads to carry out any desirable flight plan. A thread is thus developed to enable the drone to fly a number of flight plans incorporating take-off, fly forwards, fly backwards, fly to the left, fly to the right, to ascend, to descend and to land. Each of these individual flight control instructions are written as a C program function. Combining the functions in the desired order enables the drone to follow any desired flight plan. The distance travelled by the drone is calculated from two variables v_x – velocity in the x direction, and v_y –velocity in the y direction, transmitted by the drone as part of the navigation data. The velocity values are effectively integrated with respect to time to provide an estimate of the distance travelled. The accuracy of the distance calculation is therefore dependent upon the accuracy of the velocity data values obtained from the drone. Unfortunately these values do vary in accuracy depending upon the terrain the drone is flying over. Parrot suggest flying over a uniform repeated pattern such as a floor of square tiles to provide the most accurate velocity data. Results captured in tests show that the accuracy of the distance calculated can vary between 91% and 99%.

The practical autonomous drone is implemented by modifying the programs developed on the lap top and uploading to a NodeMCU module. The programs running in the NodeMCU do not incorporate thread programming but are able to read navigation data transmitted by the drone and transmit flight commands to the drone. When strapped to the drone, a fully autonomous drone is implemented.

An algorithm is developed in Chapter 4 to estimate the flightpath of the quadcopter drone by considering the relative angular velocity of the four rotors. A differing angular velocity in the rotors of the quadcopter creates torque and initiates drone movement. If for example the rear motors are rotating at a greater angular velocity than the front rotors, the generated torque causes the drone to pitch forward and move in the forward direction. A torque equation is developed to enable the flight plan of the drone and the distance travelled in any particular direction to be estimated. The magnitude of the resulting output from the torque equation indicates the drone velocity, and the time the resulting output is active enables the distance travelled to be calculated. Results from experimentation show the distance travelled calculated to minimum accuracy of 95%,

Chapter 5 describes the development of a dual drone network. Each drone incorporates an additional NodeMCU module to enable network capability. The additional NodeMCU module (DWi-Fi_x.0) is connected to the original NodeMCU (DWi_Fi_x.1) module via serial communication links. A set of three byte flight command codes have been developed which DWi-Fi_x.0 communicates to DWi_Fi_x.1 via the serial communication connection to instruct the flight plan to be followed. The flight command codes can also be transmitted from one drone to another across the network to enable one drone to control the flightpath of another drone. DWi-Fi_x.0 also receives navigation data from DWi_Fi_x.1 via the serial communication connection which is transmitted on to the laptop to capture the flight information data. In the dual drone network test, drone 0 instructs drone 1 to take-off and then takes-off itself. The drones hover for 5 seconds before drone 1 lands and also instructs drone 0 to land. The captured data shows a lag of approximately 0.2 seconds between the two drones responding to both the take-off command and lag command. This can also be observed visually. This is explained by the time taken to receive and interpret the command and the network delay.

The dual drone network of Chapter 5 is developed in Chapter 6 to produce a multi drone network. A discovery algorithm is developed and implemented to enable any station drone within range of the access point drone to join the drone network. Each drone on the network is allocated a drone number by the access point drone for identification purposes and so the drone knows where within the drone formation it should position itself. The program code developed for the network modules, DWi-Fi_x.0, are all identical and are therefore transferable meaning that they could be strapped to any drone from any manufacturer. To produce a complete drone network for drones from a different manufacturer would require the flight control modules DWi-Fi_x.1 to be programmed with the flight command for the particular drones concerned. The current algorithm caters for one access point drone and 4 station point drones. The received signal strength indicator, RSSI, at the station drone is utilised to determine the distance between access point and station drones. Initial measurements of RSSI are taken between access point and station drones at varying distances and orientation whilst the drones are not flying to establish the relationship between the distance and the RSSI value. The RSSI values are measured during a flight where both drones take off, climb to 1.2m, the station drone flies forward 12m, hovers for 10s, flies backwards to the starting point and then both drones land. Results show a good correlation between the RSSI measurements taken when the drone is not flying and the RSSI measurements taken when the drone is flying the described flight plan. The RSSI values are observed to fluctuate considerably during the data captured during the flight test. More accurate RSSI measurements could be taken when flying if the drone is stationary and a number of RSSI values are taken and the average calculated.

Finally a drone position algorithm has been proposed to determine the position of the drones on the network. The position of the drones are represented as a set of Cartesian co-ordinates with respect to the access point drone which is positioned at the origin. The algorithm utilises the RSSI measurement between access point and drone to determine the distance. By completing a series of movements and measuring the RSSI values at each new location the position of the drone can be calculated. It would be interesting to determine the accuracy of the proposed algorithm.

7.2 Future work

Although the discovery and flight control communication algorithms have been developed for a multi drone system they have only been implemented and tested using two drones. The examination of the algorithms with an increased number of drones should be performed to ensure their integrity. The current algorithms caters for one access point drone and 4 station point drones. Some applications, such as surveillance, may require more drones and in differing configurations such as a ‘daisy chain’. Consideration should be given to modify the algorithms to cater for such applications.

In determining the distance between the access point drone and a station drone an RSSI value from a single source is utilised. In fact between the access point drone and the station drone there are three access points. Each drone has its own access point and the access point drone NodeMCU is also configured as an access point. A more reliable RSSI measurement could therefore be potentially achieved by triangulating the three RSSI values available to provide a more accurate estimation of the distance between two drones.

Flight control programs have been developed to implement the drone position algorithm. The flight control programs have been tested and the drones perform the desired flight plan. What was not realised when the algorithm was tested was that the drones were flying too low, and the floor was impacting on the RSSI values measured. Flying the drones at a greater height, at for example 1.2m, appears to solve the multipath reception problem. Once implemented, the 2m distances flown by the drone in the algorithm to calculate the position can be varied to obtain the optimum drone position result.

References

[1] N. Tesla, *Dr. Nikola Tesla - Complete Patents, Vols 1 and 11*. Milbrae, CA: Tesla Book Co. 1979 Library of Congress Catalog Card 79-67722, patent No. 613809.

[2] A. Marincic and D. Budimir, "Tesla's multi-frequency wireless radio controlled vessel," in *2008 IEEE History of Telecommunications Conference*, 11-12 Sept. 2008 2008, pp. 24-27, doi: 10.1109/HISTELCON.2008.4668708.

[3] N. Budanovic. (2017) War History Online: The Early Days of Drones Unmanned aircraft from World War 1 and World War 2. [Online]. Available: www.warhistotyonline.com

[4] A. Chapman. (2016) Types of Drones-Rotor vs Fixed-Wing vs Single Rotor vs Hybrid VTOL. [Online]. Available: www.auav.com.au

[5] E. Darack. (2017) A Brief History of Quadrotors. *Air and Space/Smithsonian Magazine* [Online]. Available: www.airspacemag.com

[6] DJI. [Online]. Available: https://www.dji.com/uk

[7] 3DR. [Online]. Available: https://3dr.com/

[8] (June 2013) The Teal Group Corporation: Teal Group Predicts Worldwide UAV Market Will Total $89 Billion in Its 2013 UAV Market Profile and Forecast. [Online]. Available: https://www.tealgroup.com/index.php/pages/press-releases/40-teal-group-predicts-worldwide-uav-market-will-total-89-billion-in-its-2013-uav-market-profile-and-forecast

[9] (2018) International Data Corporation: Forecast to Total $115.7 Billion in 2019, According to New IDC Spending Guide. [Online]. Available: https://www.idc.com/getdoc.jsp?containerId=prUS44505618

[10] (2016) Goldman Sachs: Drones: Reporting for Work. [Online]. Available: https://www.goldmansachs.com/insights/technology-driving-innovation/drones/

[11] M. Sheehan. (2019) Top 100 drone companies to follow in 2019. [Online]. Available: https://mydroneauthotity.com

[12] (2018) US Department of Transport: FAA Drone Registration Tops one million. [Online]. Available: https://www.transportation.gov/briefing-room/faa-drone-registry-tops-one-million

[13] D. Ayemba. (2018) Utilising drone technology in construction. [Online]. Available: https://constructionreviewonline.com/2018/03/drones-in-construction/

[14] G. Ding, Q. Wu, L. Zhang, Y. Lin, T. A. Tsiftsis, and Y. Yao, "An Amateur Drone Surveillance System Based on the Cognitive Internet of Things," *IEEE Communications Magazine*, vol. 56, no. 1, pp. 29-35, 2018, doi: 10.1109/MCOM.2017.1700452.

[15] J. Ye, H. Chen, and W. Tsai, "Panorama Generation Based on Aerial Images," in *2018 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, 23-27 July 2018 2018, pp. 1-6, doi: 10.1109/ICMEW.2018.8551548.

[16] D. Yavuz, H. Akbıyık, and E. Bostancı, "Intelligent drone navigation for search and rescue operations," in *2016 24th Signal Processing and Communication Application Conference (SIU)*, 16-19 May 2016 2016, pp. 565-568, doi: 10.1109/SIU.2016.7495803.

[17] S. Bernardini, M. Fox, and D. Long, "Planning the behaviour of low-cost quadcopters for surveillance missions," in *Twenty-Fourth International Conference on Automated Planning and Scheduling*, 2014.

[18] D. Lee, J. Zhou, and W. T. Lin, "Autonomous battery swapping system for quadcopter," in *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, 9-12 June 2015 2015, pp. 118-124, doi: 10.1109/ICUAS.2015.7152282.

[19] (2019) Dronethusiast: Best Drones with Longest Flight Times. [Online]. Available: https://www.dronethusiast.com/best-drones-with-longest-flight-times/

[20] E. Yanmaz, S. Yahyanejad, B. Rinner, H. Hellwagner, and C. Bettstetter, "Drone networks: Communications, coordination, and sensing," *Ad Hoc Networks*, vol. 68, pp. 1-15, 2018/01/01/ 2018, doi: https://doi.org/10.1016/j.adhoc.2017.09.001.

[21] L. Gupta, R. Jain, and G. Vaszkun, "Survey of Important Issues in UAV Communication Networks," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1123-1152, 2016, doi: 10.1109/COMST.2015.2495297.

[22] E. Yanmaz, R. Kuschnig, M. Quaritsch, C. Bettstetter, and B. Rinner, "On path planning strategies for networked unmanned aerial vehicles," in *2011 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 10-15 April 2011 2011, pp. 212-216, doi: 10.1109/INFCOMW.2011.5928811.

[23] H. Shakhatreh *et al.*, *Unmanned Aerial Vehicles: A Survey on Civil Applications and Key Research Challenges*. 2018.

[24] K. Geon-Hwan, N. Jae-Choong, I. Mahmud, and C. You-Ze, "Multi-drone control and network self-recovery for flying Ad Hoc Networks," in *2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*, 5-8 July 2016 2016, pp. 148-150, doi: 10.1109/ICUFN.2016.7537004.

[25] M. H. Tareque, M. S. Hossain, and M. Atiquzzaman, "On the routing in Flying Ad Hoc Networks," in *2015 Federated Conference on Computer Science and Information Systems (FedCSIS)*, 13-16 Sept. 2015 2015, pp. 1-9, doi: 10.15439/2015F002.

[26] (2017) NASA: Global Positioning System History. [Online]. Available: https://www.nasa.gov/directorates/heo/scan/communications/policy/GPS_History.html

[27] (2018) GPS.gov: Selective Availability. [Online]. Available: https://www.gps.gov/systems/gps/modernization/sa/

[28] European Commission: Galileo. [Online]. Available: https://ec.europa.eu/growth/sectors/space/galileo_en

[29] (2019) ESA: Galileo General Introduction. [Online]. Available: https://gssc.esa.int/navipedia/index.php/Galileo_General_Introduction

[30] IOP: How Does GPS Work. [Online]. Available: http://www.physics.org/article-questions.asp?id=55

[31] GPS.Gov: GPS Accuracy. [Online]. Available: https://www.gps.gov/systems/gps/performance/accuracy/

[32] F. Van Diggelen and P. Enge, "The worlds first gps mooc and worldwide laboratory using smartphones," in *Proceedings of the 28th international technical meeting of the satellite division of the institute of navigation (ION GNSS+ 2015)*, 2015, pp. 361-369.

[33] GPS.gov: How GPS Works. [Online]. Available: https://www.gps.gov/multimedia/poster/

[34] M. G. Wing, A. Eklund, and L. D. Kellogg, "Consumer-Grade Global Positioning System (GPS) Accuracy and Reliability," *Journal of Forestry,* vol. 103, no. 4, pp. 169-173, 2005, doi: 10.1093/jof/103.4.169.

[35] M. Modsching, R. Kramer, and K. ten Hagen, "Field trial on GPS Accuracy in a medium size city: The influence of built-up," in *3rd workshop on positioning, navigation and communication*, 2006, vol. 2006, pp. 209-218.

[36] L. Li, H. Xiaoguang, C. Ke, and H. Ketai, "The applications of WiFi-based Wireless Sensor Network in Internet of Things and Smart Grid," in *2011 6th IEEE Conference on Industrial Electronics and Applications*, 21-23 June 2011 2011, pp. 789-793, doi: 10.1109/ICIEA.2011.5975693.

[37] M. Khan, "Quadcopter flight dynamics," *International Journal of Science and Technology Research*, vol. 3, no. 8, pp. 130-135, 2014.

[38] A. Javir, K. Pawar, S. Dhudum, N. Patale, and S. Patil, "Design, analysis and fabrication of quadcopter," *Journal of The International Association of Advanced Technology and Science,* vol. 16, 2015.

[39] D. Floreano and R. J. Wood, "Science, technology and the future of small autonomous drones," *Nature*, vol. 521, p. 460, 05/27/online 2015, doi: 10.1038/nature14542.

[40] K. Tanaka *et al.*, "A micromachined vibrating gyroscope," *Sensors and Actuators A: Physical,* vol. 50, no. 1-2, pp. 111-115, 1995.

[41] P.-J. Bristeau, F. Callou, D. Vissiere, and N. Petit, "The navigation and control technology inside the ar. drone micro uav," *IFAC Proceedings Volumes,* vol. 44, no. 1, pp. 1477-1484, 2011.

[42] Parrot. [Online]. Available: https://www.parrot.com/uk/drones/parrot-ardrone-20-elite-edition

[43] DJI: Mavic 2. [Online]. Available: https://www.dji.com/uk/mavic-2/info

[44] DJI Forum: RC vs WiFi. [Online]. Available: https://forum.dji.com/thread-124974-1-1.html

[45] Internet World Statistics 1995 -2019. Usage and Population Statistics [Online]. Available: https://www.internetworldstats.com/emarketing.htm#top

[46] C. Prescott. (2016) Office of National Staistics, Internet access - housholds and individuals, Great Britain [Online]. Available: https://www.ons.gov.uk/peoplepopulationandcommunity/householdcharacteristics/homeinternetandsocialmediausage/bulletins/internetaccesshouseholdsandindividuals/2016

[47] C. Prescott. (2019) Office for National Statistics - Internet Users, UK. [Online]. Available: https://www.ons.gov.uk/businessindustryandtrade/itandinternetindustry/bulletins/internetusers/2019

[48] (2004) A Brief History of Wi-Fi - Case History. *The Economist*.

[49] W. Watterson. (2016) The Next Web - WiFi and the coming IOT invasion. [Online]. Available: https://thenextweb.com/insider/2016/11/30/wifi-coming-iot-invasion/

[50] (2019) Victor Hayes - IEEE Computer Society. [Online]. Available: https://www.computer.org/profiles/victor-hayes

[51] (2017) CableFree - Wireless Technology - The History of Wi-Fi 1971 to today. [Online]. Available: https://www.cablefree.net/wireless-technology/history-of-wifi-technology/

[52] Skydio. [Online]. Available: https://www.skydio.com/

[53] Yuneec. [Online]. Available: http://yuneec.uk/

[54] J. Galbraith. (2013) Fact or Fiction: What affects Wi-Fi speed. *Macworld*. Available: https://www.macworld.com/article/2058324/fact-or-fiction-what-affects-wi-fi-speed-.html

[55] D. A. Gandhi and M. Ghosal, "Novel Low Cost Quadcopter for Surveillance Application," in *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*, 11-12 July 2018 2018, pp. 412-414, doi: 10.1109/ICIRCA.2018.8597391.

[56] R. K. Rangel and A. C. Terra, "Development of a surveillance tool using UAV's," in *2018 IEEE Aerospace Conference*, 3-10 March 2018 2018, pp. 1-11, doi: 10.1109/AERO.2018.8396603.

[57] P. N. Patel, M. A. Patel, R. M. Faldu, and Y. R. Dave, "Quadcopter for agricultural surveillance," *Advance in Electronic and Electric Engineering*, vol. 3, no. 4, pp. 427-432, 2013.

[58] A. Williams and O. Yakimenko, "Persistent mobile aerial surveillance platform using intelligent battery health management and drone swapping," in *2018 4th International Conference on Control, Automation and Robotics (ICCAR)*, 20-23 April 2018 2018, pp. 237-246, doi: 10.1109/ICCAR.2018.8384677.

[59] J. Capitan, L. Merino, and A. Ollero, "Cooperative decision-making under uncertainties for multi-target surveillance with multiples UAVs," *Journal of Intelligent & Robotic Systems*, vol. 84, no. 1-4, pp. 371-386, 2016.

[60] "The Incredible Aerial Photographs of American Cities Taken by the World's First Drone: Pioneer Took Images in 1900s with Cameras Attached to Kites. https://www.dailymail.co.uk/news/article-2347122/The-incredible-aerial-photographs-American-cities-taken-worlds-drone-Pioneer-took-images-1900s-cameras-attached-kites.html," in *Daily Mail*, ed, 2013.

[61] A. E. Holton, S. Lawson, and C. Love, "Unmanned Aerial Vehicles: Opportunities, barriers, and the future of “drone journalism”," *Journalism practice*, vol. 9, no. 5, pp. 634-650, 2015.

[62] M. Tremayne and A. Clark, "New Perspectives from The Sky," *Digital Journalism*, vol. 2, no. 2, pp. 232-246, 2014/04/03 2014, doi: 10.1080/21670811.2013.805039.

[63] M. Schroyer, "Seven more reasons why journalists should learn to fly unmanned aircraft," *Professional Society of Drone Journalists*, 2013.

[64] K. B. Culver, "From Battlefield to Newsroom: Ethical Implications of Drone Technology in Journalism," *Journal of Mass Media Ethics*, vol. 29, no. 1, pp. 52-64, 2014/01/02 2014, doi: 10.1080/08900523.2013.829679.

[65] B. Jenkins, "Watching the watchmen: Drone privacy and the need for oversight," *Ky. LJ*, vol. 102, p. 161, 2013.

[66] J. M. Loffi, R. J. Wallace, J. D. Jacob, and J. C. Dunlap, "Seeing the threat: Pilot visual detection of small unmanned aircraft systems in visual meteorological conditions," *International Journal of Aviation, Aeronautics, and Aerospace,* vol. 3, no. 3, p. 13, 2016.

[67] (2018) Federal Aviation Administration.

. *Small Unmanned Aircraft Regulations (Part 107)* [Online]. Available: https://www.faa.gov/news/fact_sheets/news_story.cfm?newsId=22615

[68] (2019) Federal Aviation Administration. Recreational Flyers & Modeler Community-Based Organizations. [Online]. Available: https://www.faa.gov/uas/recreational_fliers/

[69] (2016) UK Civil Aviation Aouthority. Regulations relating to the use of small drones. [Online]. Available: https://www.caa.co.uk/Commercial-industry/Aircraft/Unmanned-aircraft/Small-drones/Regulations-relating-to-the-commercial-use-of-small-drones/

[70] (April 2019) UK Civil Aviation Authority. Updates About Drones. [Online]. Available: https://www.caa.co.uk/Consumers/Unmanned-aircraft/Our-role/Updates-about-drones/

[71] M. Erdelj and E. Natalizio, "UAV-assisted disaster management: Applications and open issues," in *2016 International Conference on Computing, Networking and Communications (ICNC)*, 15-18 Feb. 2016 2016, pp. 1-5, doi: 10.1109/ICCNC.2016.7440563.

[72] M. Erdelj, E. Natalizio, K. R. Chowdhury, and I. F. Akyildiz, "Help from the Sky: Leveraging UAVs for Disaster Management," *IEEE Pervasive Computing,* vol. 16, no. 1, pp. 24-32, 2017, doi: 10.1109/MPRV.2017.11.

[73] H. Saha *et al.*, "A low cost fully autonomous GPS (Global Positioning System) based quad copter for disaster management," in *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, 8-10 Jan. 2018 2018, pp. 654-660, doi: 10.1109/CCWC.2018.8301782.

[74] D. Câmara, "Cavalry to the rescue: Drones fleet to help rescuers operations over disasters scenarios," in *2014 IEEE Conference on Antenna Measurements & Applications (CAMA)*, 16-19 Nov. 2014 2014, pp. 1-4, doi: 10.1109/CAMA.2014.7003421.

[75] M. Hutson. (2017) Hurricanes Show Why Drones Are the Future of Disaster Relief. NBC News. [Online]. Available: https://www.nbcnews.com/mach/science/hurricanes-show-why-drones-are-future-disaster-relief-ncna799961

[76] A. Shukla, H. Xiaoqian, and H. Karki, "Autonomous tracking of oil and gas pipelines by an unmanned aerial vehicle," in *2016 IEEE 59th International Midwest Symposium on Circuits and Systems (MWSCAS)*, 16-19 Oct. 2016 2016, pp. 1-4, doi: 10.1109/MWSCAS.2016.7870114.

[77] T. R. Bretschneider and K. Shetti, "UAV-based gas pipeline leak detection," in *Proceedings of the Asian Conference on Remote Sensing, Nay Pyi Taw, Myanmar*, 2014, pp. 27-31.

[78] B. Shakmak and A. Al-Habaibeh, "Detection of water leakage in buried pipes using infrared technology; A comparative study of using high and low resolution infrared cameras for evaluating distant remote detection," in *2015 IEEE Jordan*

Conference on Applied Electrical Engineering and Computing Technologies (AEECT), 3-5 Nov. 2015 2015, pp. 1-7, doi: 10.1109/AEECT.2015.7360563.

[79] C. Rose. (2013) Amazon's Jeff Bezos looks to the future. CBS News. [Online]. Available: https://www.cbsnews.com/news/amazons-jeff-bezos-looks-to-the-future/

[80] C. C. Murray and A. G. Chu, "The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery," *Transportation Research Part C: Emerging Technologies*, vol. 54, pp. 86-109, 2015.

[81] K. Dorling, J. Heinrichs, G. G. Messier, and S. Magierowski, "Vehicle Routing Problems for Drone Delivery," *IEEE Transactions on Systems, Man, and Cybernetics: Systems,* vol. 47, no. 1, pp. 70-85, 2017, doi: 10.1109/TSMC.2016.2582745.

[82] A. Goodchild and J. Toy, "Delivery by drone: An evaluation of unmanned aerial vehicle technology in reducing CO2 emissions in the delivery service industry," *Transportation Research Part D: Transport and Environment*, vol. 61, pp. 58-67, 2018.

[83] T. R. Gulden, "The Energy Implications of Drones for Package Delivery: A Geographic Information System Comparison," 2017.

[84] V. Bryan, "Drone delivery: DHL 'parcelcopter' flies to German isle," ed: Reuters, 2014.

[85] A. Choi-Fitzpatrick *et al.*, "Up in the Air: A Global Estimate of Non-Violent Drone Use 2009-2015," 2016.

[86] Z. Tianqu and J. Hong, "Landing system for AR.Drone 2.0 using onboard camera and ROS," in *2016 IEEE Chinese Guidance, Navigation and Control Conference (CGNCC)*, 12-14 Aug. 2016 2016, pp. 1098-1102, doi: 10.1109/CGNCC.2016.7828941.

[87] R. Schurig, T. Désesquelles, A. Dumont, E. Lefranc, and A. Lux, "3D stereoscopic measurements from AR Drone Squadron," in *2014 IEEE International Conference on Control Science and Systems Engineering*, 29-30 Dec. 2014 2014, pp. 23-26, doi: 10.1109/CCSSE.2014.7224501.

[88] T. Saito and K. Mase, "DronePilot.NET Development: AR.Drone SDK Supporting Native and Managed Code," in *2013 International Conference on Advanced Computer Science Applications and Technologies*, 23-24 Dec. 2013 2013, pp. 60-64, doi: 10.1109/ACSAT.2013.19.

[89] V. Indrawati, A. Prayitno, and G. Utomo, "Comparison of two fuzzy logic controller schemes for position control of AR.Drone," in *2015 7th International Conference on Information Technology and Electrical Engineering (ICITEE)*, 29-30 Oct. 2015 2015, pp. 360-363, doi: 10.1109/ICITEED.2015.7408972.

[90] A. Gibiansky, "Quadcopter dynamics, simulation, and control," *Andrew. gibiansky. com*, 2012.

[91] A. L. Salih, M. Moghavvemi, H. A. F. Mohamed, and K. S. Gaeid, "Modelling and PID controller design for a quadrotor unmanned air vehicle," in *2010 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*, 28-30 May 2010 2010, vol. 1, pp. 1-5, doi: 10.1109/AQTR.2010.5520914.

[92] T. Luukkonen, "Modelling and control of quadcopter," *Independent research project in applied mathematics, Espoo*, vol. 22, 2011.

[93] K. M. Konwar, D. Kowalski, and A. A. Shvartsman, "Node discovery in networks," *Journal of Parallel and Distributed Computing,* vol. 69, no. 4, pp. 337-348, 2009.

[94] V. Dyo and C. Mascolo, "Efficient node discovery in mobile wireless sensor networks," in *International Conference on Distributed Computing in Sensor Systems*, 2008: Springer, pp. 478-485.

[95] M. Kohvakka, J. Suhonen, M. Kuorilehto, V. Kaseva, M. Hännikäinen, and T. D. Hämäläinen, "Energy-efficient neighbor discovery protocol for mobile wireless sensor networks," *Ad Hoc Networks,* vol. 7, no. 1, pp. 24-41, 2009.

[96] Y. M. Mustafah, A. W. Azman, and F. Akbar, "Indoor UAV positioning using stereo vision sensor," *Procedia Engineering,* vol. 41, pp. 575-579, 2012.

[97] Y.-H. Jin, K.-W. Ko, and W.-H. Lee, "An indoor location-based positioning system using stereo vision with the drone camera," *Mobile Information Systems,* vol. 2018, 2018.

[98] M. Bahiki, N. Talib, and S. Azrad, "Relative positioning-based system with tau control for collision avoidance in swarming application," in *IOP Conference Series: Materials Science and Engineering*, 2016, vol. 152, no. 1: IOP Publishing, p. 012025.

[99] C. Ladha, B. S. Sharif, and C. C. Tsimenidis, "Mitigating propagation errors for indoor positioning in wireless sensor networks," in *2007 IEEE International Conference on Mobile Adhoc and Sensor Systems*, 2007: IEEE, pp. 1-6.

[100] J. Xu, W. Liu, F. Lang, Y. Zhang, and C. Wang, "Distance measurement model based on RSSI in WSN," *Wireless Sensor Network,* vol. 2, no. 08, p. 606, 2010.

[101] D. Capriglione, L. Ferrigno, E. D'Orazio, V. Paciello, and A. Pietrosanto, "Reliability analysis of RSSI for localization in small scale WSNs," in *2012 IEEE International Instrumentation and Measurement Technology Conference Proceedings*, 2012: IEEE, pp. 935-940.

[102] L. Xu, F. Yang, Y. Jiang, L. Zhang, C. Feng, and N. Bao, "Variation of received signal strength in wireless sensor network," in *2011 3rd International Conference on Advanced Computer Control*, 2011: IEEE, pp. 151-154.

[103] J. Luomala and I. Hakala, "Effects of temperature and humidity on radio signal strength in outdoor wireless sensor networks," in *2015 Federated Conference on Computer Science and Information Systems (FedCSIS)*, 2015: IEEE, pp. 1247-1255.

[104] S. Sabu, S. Renimol, D. Abhiram, and B. Premlet, "A study on the effect of temperature on cellular signal strength quality," in *2017 International Conference on Nextgen Electronic Technologies: Silicon to Software (ICNETS2),* 2017: IEEE, pp. 38-41.

[105] M. Ivanić and I. Mezei, "Distance Estimation Based on RSSI Improvements of Orientation Aware Nodes," in *2018 Zooming Innovation in Consumer Technologies Conference (ZINC),* 30-31 May 2018 2018, pp. 140-143, doi: 10.1109/ZINC.2018.8448660.

[106] S. Barai, D. Biswas, and B. Sau, "Estimate distance measurement using NodeMCU ESP8266 based on RSSI technique," in *2017 IEEE Conference on Antenna Measurements & Applications (CAMA),* 4-6 Dec. 2017 2017, pp. 170-173, doi: 10.1109/CAMA.2017.8273392.