THE UNIVERSITY OF WESTERN AUSTRALIA

FINAL YEAR PROJECT

Autonomous Control of a Multirotor Unmanned Aerial Vehicle Real-time computer-vision guided navigation about moving points of interest

---

Author:
Brett DOWNING

Supervisors:
Prof. Thomas BRAUNL
Chris CROFT

May 22, 2015

Abstract

This document outlines a study into using computer-vision to guide navigation routines for object tracking and autonomous remote inspection. The low complexity and VTOL capability of multi-rotor Micro-UAVs, and the falling cost of parts lends these platforms to an ever increasing array of autonomous routines from remote inspection to extreme sport chase-cam applications.

Contents

1 Introduction

1.1 Motivation

Recent Developments in power and control electronics has allowed small, consumer level UAVs to lift enough processing capacity to navigate at least partly by computer vision.

A number of Commercial Micro UAVs are beginning to advertise the ability to act as an autonomous chase-cam [1] [2]. Almost all of these systems rely on a tracking device on the user, and many navigate entirely by GPS. While the performance of GPS is improving, the performance of a chase-cam will be limited to the update rate and fix accuracy of the beacon and drone GPS modules. It also requires a lock to be achieved in both devices before tracking can commence.

The power distribution and plant monitoring sectors have recently been taking on small fleets of drones to perform routine inspection of remote, hazardous or otherwise difficult locations [3]. Many of these applications are reasonably repetitive and well defined enough to fully automate, but cannot be navigated by GPS alone due to the poor repeatability of the GPS system.

This study will investigate the usability of computer vision alone to track and follow a moving target, and navigate an automated inspection routine of points of interest established by computer-vision. It is hoped that this work can be used to improve the performance of camera tracking routines in autonomous chase-cam and cinematographic applications, and partially automate remote inspection

Much of the technology related to multi-copters is applicable to most other forms of UAV. The vast array of applications micro UAVs have found suggests this work may find use in Agriculture, mapping, targeted crop dusting, Cinematography, extreme-sport photography, Data collection, remote observation and inspection and hazard and environmental monitoring.

1.2 Project History

1.2.1 2013

The 2013 team, O'Connor [4] and Venables [5] established the project with the purchase of a DJI flame-wheel F550 Hexacopter with a NAZA-lite flight controller. This copter was fitted and tested, and finally converted into an autonomous platform with the addition of a Raspberry Pi single-board computer (RPi) and a circuit to switch the control channels from the radio receiver to the RPi GPIO outputs. The NAZA-lite at this time did not feature telemetry outputs or way-point inputs, but it was able to loiter in a stiff wind using a GPS fix. This team gathered location information for the RPi using a Qstarz GPS unit, and bearing information using an X-sens IMU. The sensor duplication did not exceed the maximum payload capacity of the platform, but it did suffer a short flight-time. Under direct control from the RPi, the drone was able to execute basic way-point navigation.

1.2.2 2014

The 2014 team, Baxter [6], Mazur [7] and Targhagh [8] continued the project adding an internet accessible web UI to control the various autonomous features of the platform, displaying the flight-path of the copter and a live feed of the camera. The navigation routines were improved and extended to permit operation without reliable heading information, and the computer vision routines were extented.

1.2.3 2015

The project is being continued by Allen [9], Mohanty [10], Tan [11] and Downing [12]. Because of the rapid pace that the market is adopting new features and technologies, we saw fit to undertake a critical review of all aspects of the platform, and begin improvements that facilitate the current typical feature set.

2 Literature Review

2.1 Optical Search

Search and Rescue is a field that is extremely appropriate to automation due to the difficulty in mobilising teams in remote, harsh or dangerous conditions. A micro UAV can be deployed quickly and commence the search operation before boots can be placed on the ground. The UAV Outback Challenge [13] is a regular competition to perform various search and rescue missions autonomously. The performance criteria are deliberately set very high, and the competition frequently goes uncompleted. In 2012, CanberraUAV [14], a UAV development team from Canberra completed the search aspect of the competition. After trying a number of image recognition algorithms, the search algorithm that they flew with simply looked for the blue of the jeans of Outback Joe [15]. This was sufficient to locate the target in a 4x6 km area. This goes to show that even a minimally complex routine can be effectively applied in appropriate conditions.

2.2 Terrain Estimation via Optical Flow Methods

Adding sensors to allow the copter to understand its environment is a surprisingly difficult task. Conventional contact methods operate at ranges far too close for UAVs, ultrasonics are buffeted by down-wash and most depth sensors are either too heavy or suffer under intense light. In terms of simplicity of algorithms, biomimickry has turned up some surprising results. In 2004, a French research group applied a number of optical flow algorithms to the navigation of a small helicopter [16]. The computer vision routines were used to inform the navigation loops, following the middle of urban tunnels and reducing speed in dense clutter, without necessarily computing the distance to the obstacles. These routines were relatively expensive in terms of computational power, but extremely simple and parallelizeable.

2.3 Position Estimation using Stereoscopic Methods

Any measurement will have an associated uncertainty, extracting the most information out of a collection of measurements rarely involves taking the most accurate measurements. It is possible to estimate the position of an object or feature using multiple images separated in either space or time, but making effective use the information requires an understanding of how the uncertainties behave. Error Modeling in Stereo Navigation [17] investigates a number of routines to estimate the position of a vehicle by tracking visible land-marks in a stereoscopic system, and notes the interaction between the geometry of the uncertainties, and the stability of the result.

2.4 Altitude Estimation by optical flow

Elementary methods are still interesting for the sake of biomimickry, and indeed, a 20 element photo-array was demonstrated sufficient to control the altitude of an aircraft [18], coupling the altitude to the velocity. Dedicated, low resolution optical flow sensors have become exceedingly cheap since the market saturation of the optical mouse, and many commodity flight computers already include doppler information from GPS modules. Combining this data allows the UAV to estimate the distance to the terrain, [19]. These commodity sensors typically do not include circuitry for computing rotation, but given the cost of the sensors, that limitation can be overcome using two sensors [20]. With the availability of these parts, a number of UAV systems such as ArduPilot already include support for these devices in their code-base [21]. This support also covers pitch and roll compensation and position stabilisation using odometetry.

2.4.1 Notable Commercial Systems

The rapidly expanding market for personal UAVs, and the compelling imagery from sponsored extreme sport drone cameras has created a call for personal chase-cams in the last year or so. The Lily [1] and the Air-Dog [2] are pre-release personal autonomous chase-cam systems.

3 Progress Report

3.1 Where we started

Figure 1: The hexacopter at a flight test shortly after hand-over

3.1.1 Flight Hardware

The Hexacopter is a DJI F550 frame with AXI2217 motors and DJI OPTO30A ESCs, 254mm props and a 6Ah 11.1V Lipo battery. This platform gives us a generous lifting capacity, and enough flight time to run multiple tests. We estimate the power consumption at take-off to be in the region of 50-100A. A switch had been installed in-line between the battery and the power distribution board. This switch was a 5A rocker switch rated to mains voltage. Surprisingly, it lasted two full years of testing and only melted shortly after hand-over in 2015.

3.1.2 Flight Electronics

The 2013 team decided to use a NAZA-V1-lite flight computer. This works well for free-flight, but does not make provisions for way-point navigation or telemetry.Interfacing is via Servo signals only, generated by ServoBlaster [22] on the RPi.

In order to switch between manual and autonomous modes, the 2013 team installed a switching circuit. This circuit featured a 555 timer and four relays. There was no schematic, and the only documentation was a copper layout [4, p. 27-28]. Only after meeting with the 2014 team did we become aware that there were dead channels on this board due to multiple failed relays.

3.1.3 Autonomy Electronics

In Autonomous mode, the hexacopter is directed by a Raspberry Pi Model-B Single-Board-Computer. This platform is extremely widely used for all manner of projects simply due to its cost. It supports Linux and has drivers for almost any hardware we could hope to add to the system. It supports a camera over a dedicated interface. The RPi is not a powerful computer it is a single ARM core running at 900MHz, and has band-width limitations that make stereoscopy difficult, The server has a WiFi connection allowing us to remotely control the autonomous features, and even re-configure and recompile components during manual flight.

The Wireless adapter, the GPS system, and the IMU were all connected via USB, which was powered via the RPi. The wireless connection would frequently drop out, and the GPS would lose lock or re-connect to USB, causing the navigation routine to lock up. The Wiring Harness was undocumented, with links that appeared entirely temporary, but looked as though they had been there for far too long.

3.1.4 Autonomy Software

The on-board server is currently doing all of the autonomous navigation processing. This works reasonably well for slow manoeuvres but the latency in data collection, and the generation and re-interpretation of control signals leaves the Pi at a distinct performance disadvantage to autonomous routines running on the flight-computer.

The Pi runs a basic webserver hosting a page with basic autonomy controls. The web interface has a small suite of tools for creating way-points, enabling camera tracking modes, and plots the course of the drone on a map.

3.1.5 Sensors and auxiliary Hardware

The camera is stabilised for pitch and roll by two servos. The tilt servo was damaged and tilt stabilisation was almost non-existent. The servos receive correction information from the flight computer with a configurable gain parameter. The servos are able to hold the camera level in the steady state, but have a significantly delayed response.

The server collected position data from a second GPS module (QStarz). This GPS module appears to have a filtering scheme with dead-reckoning configured for automotive applications; It exhibited lazy vertical response, and low accelerations.

Bearing data was to be collected from an X-sens branded IMU, though this was not installed at the time of hand-over, possibly due to an unreasonably heavy wiring harness. Instead, the navigation loops in the old code-base estimated the bearing by flying forward for a fixed interval before beginning navigation, relying on the flight computer to hold the bearing for the duration of the mission.

3.2 What we've achieved

3.2.1 Flight Electronics

With the drop-outs in wireless and GPS very nearly causing a crash, We ran a full tear-down of the copter's electronics and uncovered quite a serious power supply contention issue; the NAZA's power supply was connected via the relay board to the RPi which was supposed to be powered by a different power module (a potted 5V regulator). Correcting the contention problem and repairing damaged links, the reliability of the copter was improved, but our faith was shaken. We reverse-engineered the switching circuit installed by the 2013 group, and traced the wiring harness into a net-list. Having talked to the previous teams, we seem to be working with better documentation about the hardware than the teams that installed it.

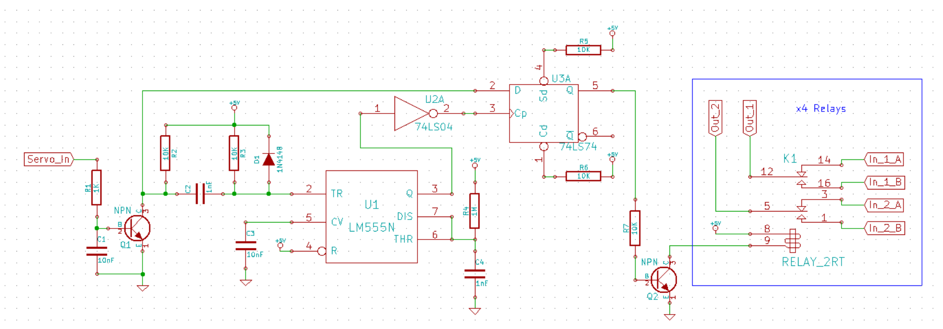

Figure 2: The circuit used to switch between the pilot and the autopilot (reverse-engineered from copper layout)

The power switch was replaced with a 15A toggle switch, still woefully under-rated, and mounted inconveniently close to the props. It was finally removed in favour of isolating the battery at the connector which was easier to get to anyway.

3.2.2 Landing Gear

We were encouraged to learn to pilot the copter, flying it on a regular basis. Naturally, the copter was crashed a couple of times, and was damaged in almost every instance. We inspected the damage and considered options to make the drone more tolerant to heavy landings, or at least make the crashes slightly cheaper. We removed the MDF arm extensions which appeared to be coupling the energy of the crash into torque on the arms, and installed long injection-moulded legs that flex a long way. This gives us more room underneath the copter for future enhancements, increases the total survivable impact energy, and should help to limit crash damage to single, cheap components that can be replaced easily.

Figure 3: A more recent photo of the copter including the landing legs.

3.2.3 Failsafe Recovery

At hand-over the copter was meant to implement a return-to-launch fail-safe in the event of transmitter failure. We found the fail-safe configurations empty on both the controller and the NAZA, and corrected this omission.

3.2.4 Software

The code-base of the previous year-groups did not appear to use version-control consistently, or at all in some cases. Many of the function files were duplicated, and functionality was not modularised. With the heroic efforts of Tan [11], we've implemented and tested a great deal of code that we were able to salvage from the previous year-groups. Oddly, our tests of the previous year-groups' code have produced better results than are published in the respective papers. However the claims of the previous groups still appear grossly overstated.

The object tracking code we ported over from the 2014 team took a relatively simplistic approach, feeding the pixel position from the image directly into two PID controllers. This did work, but altitude, camera angle, parallax, and other variables tended to make the controller go from a-bit-weak, to utterly unstable with what seemed like minor environmental changes.

We've built up the code-base to estimate the physical position of the target in coordinates relative to the copter, trying to sanitize the inputs to the control loop. The control loop now takes inputs of the target's physical coordinates in metres. Our controller is still a collection of basic PID loops, but already, the behaviour is far more consistent in flight.

3.2.5 Major Upgrades

We've made a proposal and ordered parts to remove the NAZA-V1-Lite flight computer and replace it with a 3D-Robotics Pixhawk [23] running the Ardupilot flight control software. (as of submission of this paper, work will be under way to install and test the Pixhawk.) This change allows us to utilise the vast array of supporting software that the Ardpilot community has written including ground-stations, telemetry loggers, Smart-phone apps, Kalman navigation filters and automatic flight control tuning. It also exposes all of the live flight parameters, raw and processed, available over a documented interface. This decision was partly motivated by our observations of an older APM-2.6 flight computer running similar firmware.

3.2.6 Autonomy Electronics

With the limited processing power of the RPi model B, the computer-vision routines were quite sluggish, we have upgraded to the Raspberry Pi Version 2 which has a higher core speed, multiple cores, and more advanced processor optimisations.

The mass of sensor duplication was a major contributing factor in the short flight-time of the platform. The NAZA already has GPS and IMU data, sharing it with the RPi seemed like an obvious solution. The X-sens IMU was only being used for bearing data, and the NAZA's GPS stand contained its compass. We found a forum that had a reverse-engineering effort for this protocol [24], and ported it to the Pi and its hardware serial port. This connection gives us access to the same GPS data that the flight computer is working with, reducing controller conflicts. It also gives us access to the NAZA's raw magnetometer data which could be used for bearing, but the inherent biases and changes due to unknown pitch and roll make this data difficult to use. The NAZA GPS system uses a basic TTL-level serial protocol, but the message payloads are XORed with an obscure bit combination from the payload itself. It looks as though the protocol was deliberately obfuscated by DJI.

After breaking into the NAZA GPS, and in preparation to change to the Pixhawk, the X-sens IMU was never added to the copter, the drivers were removed from the code-base, and we permanently removed Q-starz GPS module.

3.3 Aims

3.3.1 Hardware Modifications

Following the critical review of the hexacopter hardware, we intend to continue development to bring the platform to a respectable standard. These changes started with corrections to the wiring harness, replacement of the landing gear and will continue with the change to the Pixhawk flight computer as discussed in Appendix A.

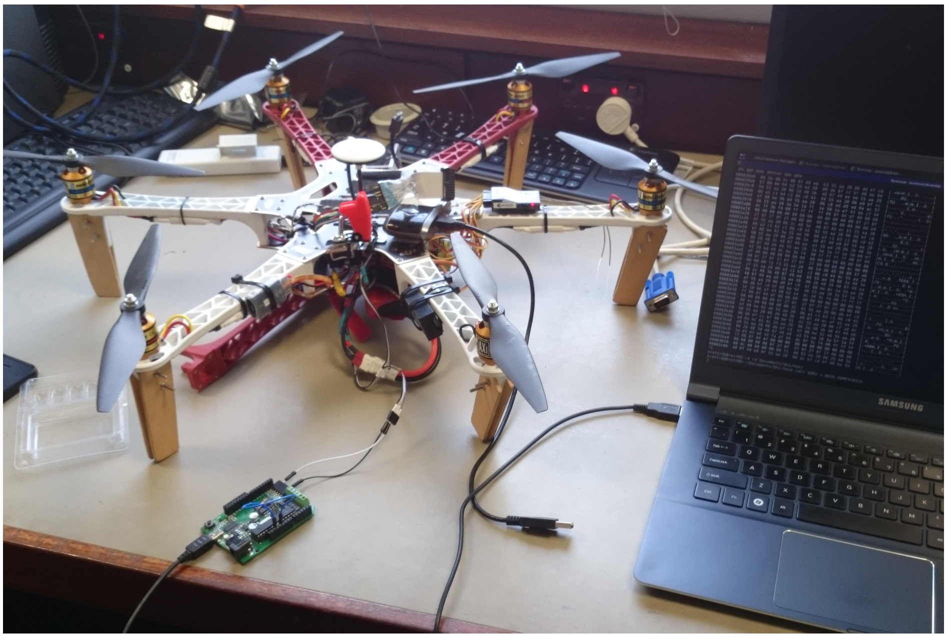

Figure 4: First tests at breaking into the NAZA GPS. Visible here is a hexdump of the GPS byte-stream via an Arduino clone operating as a TTL adapter

The servo driven camera gimbal is showing limitations in its abilty to stabilise the camera, but more testing is required before we can make an honest evaluation of its performance and recommend changes. A large number of the algorithms we have discussed to identify or track objects require slightly more information than is readily available from one camera alone. A variety of additional algorithms and sensors will be investigated during the course of this project to fill these information gaps.

3.3.2 Go Fetch

The focus of this project is to position the UAV based largely on information from the camera. We intend to continue the development of the object tracking code initially by improving the copter's ability to follow a highly visible target. This will allow us to perform reasonably aggressive manoeuvres under computer-vision guidance that should highlight weaknesses in the existing software chain.

3.3.3 Lost and Found

The copter is already beginning to exhibit strong tracking behaviour, but we have encountered a camera stability issue (Discussed in 3.4.2). We've not yet been able to apply aggressive parameters to the chase algorithm because of the instabilities it introduces, and the tracked object frequently leaves the field of view of the camera at the extremes of the controller oscillations. We intend to write a search algorithm to attempt to locate an object shortly after it has left the frame, and address the camera stability problem to improve tracking performance.

3.3.4 Everything in Its Place

We expect to improve on the physicality of our control systems, estimating latitude and longitude of the target to assist in lost-target recovery (Section 3.3.3). Logging the estimates of the target's global coordinates is likely to lead to insights into the stability and validity of our work, and suggest possible improvements.

We'd like to convert the loop to calculate the desired location of the drone based on the location of the targets that have been spotted, regardless of which objects are currently in the frame. This would provide a foundation for a Simultaneous Localisation and Mapping (SLAM) system.

Ideally, we'd use this information to calculate appropriate observation locations as way-points for remote inspection routines, and construct no-fly-zones around visible obstructions. Getting this far will rely heavily on tightly integrating work from all members of the team, and may not be feasible in light of our time constraints.

3.4 Major Challenges

3.4.1 Limiting Controller Complexity

A method for the copter to chase an object on a screen is a relatively straight-forward problem to grasp intuitively, but include camera geometry, position estimation, velocity and acceleration limits and such; just defining the problem rapidly turns into a wall of mathematics. We've made a lot of progress towards sensible estimation and chase routines already, but already the single-input-single-output PID controllers are becoming inadequate. The control scheme will need to be analysed in a more comprehensive manner, incorporating data sets from multiple sources to coordinate coherent actions in a multi-input-multi-output controller.

Limiting the scope and complexity of the controller may well become a matter of identifying diminishing returns. Rigorous flight performance analysis using sensible metrics, monitoring the time spent coding those incremental improvements, and assessing the wider applicability of the possible changes may help to identify when the chase controller should be declared "good enough".

3.4.2 Flying with a Steady Cam

As the platform stands, the camera gimbal does not sufficiently isolate platform motion to stabilise a simple control loop. For example, a forward motion instruction from the chase algorithm causes the copter's rear rotors to throttle up, then the platform pitches forward, then the copter accelerates forward. The pitch data from the IMU is sent to the camera stabilisation servo which has a slow slew rate and a small delay time. This delay couples pitch motion into the image processing loop and adds a brief but intense upward swing to the location of the target in the image as the copter accelerates. This behaviour is controlled by the physical properties of the copter, the flight-computer's control loops, the servo's controller and the shutter lag on the camera. While all of this is technically possible to compensate for, calibrating against mechanical and electrical properties of a commodity servo and an advanced flight controller introduces a large number of variables and doesn't lend itself easily to empirical validation.

It makes more sense to solve single whole problems; either cleanly stabilising the camera with a higher performance gimbal, or applying software image stabilisation using motion data from the flight controller to calculate through bulk motions, followed by optical flow methods to remove the remaining jitter. The change to the Pixhawk flight computer will give us access to IMU data required for software stabilisation, but it may prove too computationally intensive for the RPiV2 to process. Many of the parts involved in image stabilisation were not selected with any great care, and we fully expect to continue swap out or modify older parts of the copter during the course of the project.

3.4.3 Reliability and Repeatability

Between dead servos, incomplete configurations, undocumented custom circuits with supposedly known but undocumented faults, and a wiring loom that was unreliable, undocumented, and even simply incorrect; we've been set back by days at a time with intermittent faults. Maintaining a high quality of work over a long project is difficult; small, temporary hacks tend to become permanent fixtures and intermittent faults are left undiagnosed with time being prioritised to flashy achievements.

The change to the Pixhawk is set to remove, modify, and simplify a large portion of the wiring harness. This is probably the best opportunity to do clean-room engineering on the hardware. We believe we have the technical skill in the group to leave the wiring harness at least presentable, if not bullet-proof. Though it may become necessary to dedicate significant blocks of time to removing temporary fixtures periodically throughout the course of the project.

3.4.4 Extensible Code

As with incremental hardware modifications, maintaining a high degree of quality across the board is difficult. For this to happen, our code needs to reasonably well documented, modular, conform to some kind of standard, and be logged in version control. Our team has selected projects that are able to begin very independently and achieve clearly defined goals, but as these new capabilities mature, they will be used to enhance aspects of the projects of other team members. We hope that this cross-project extensibility measure will force some level of accountability to documentation and code cleanliness.

4 Conclusions

The versatility of the multirotor platform has made possibilities for research into to a truly vast field. Between automated landing routines, optical navigation methods, obstacle avoidance, the applications for multirotors are limited only by the skill of the engineer in applying the state-of-the-art. The 2015 team hopes to demonstrate a number of robust autonomous, and semi-autonomous behaviours, and lay a solid ground-work to facilitate further development of specific applications.

References

[1] "Lily - The Camera That Follows You." https://www.lily.camera/, 2015.

[2] "Auto-Follow Drone for GoPro Camera." https://www.airdog.com/, 2015.

[3] "UAV Inspection." http://www.ropeaccess-wa.com/unmanned-aerial-vehicle-inspection/, 2015.

[4] R. OConnor, "Developing a multicopter uav platform to carry out research into autonomous behaviours, using on-board image processing techniques," 2013.

[5] C. Venables, "Multirotor unmanned aerial vehicle autonomous operation in an industrial environment using on-board image processing," 2013.

[6] M. Baxter, "Autonomous hexacopter software design," 2014.

[7] A. Mazur, "Autonomous operation and control of a multirotor unmanned aerial vehicle through 4g lte using onboard gps and image processing," 2014.

[8] O. Targhagh, "Autonomous navigation of unmanned aerial vehicles," 2014.

[9] Richard Allen. https://github.com/20739367, 2015.

[10] Manish Mohanty. https://github.com/manishmohanty09, 2015.

[11] Jeremy Tan. https://github.com/jtanx/, 2015.

[12] Brett Downing. https://github.com/brettrd/, 2015.

[13] "Uav ouback challenge." http://www.uavoutbackchallenge.com.au/.

[14] C. Anderson, "Open source civilian uav development." http://canberrauav.org.au/, 2012.

[15] Andrew Tridgel, "2012 UAV Outback Challenge." Personal discussion at Make-Hack-Void following Linux Conf Canberra, 2013.

[16] L. Murateta, S. Doncieuxa, Y. Briereb, and J.-A. Meyera, "A contribution to vision-based autonomous helicopter flight in urban environments," *Robotics and Autonomous Systems*, vol. 50, p. 195209, March 2004.

[17] L. Matthies and S. Shafer, "Error modeling in stereo navigation," *IEEE Journal Of Robotics And Automation*, vol. 3, pp. 239–248, June 1987.

[18] T. Netter and N. Franceschini, "A robotic aircraft that follows terrain using a neuromorphic eye," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2002.

[19] S. Griffiths, "Remote terrain navigation for unmanned air vehicles," Master's thesis, Brigham Young University, 2006.

[20] J. Kim and G. Brambley, "Dual optic-flow integrated navigation for small-scale flying robots."

[21] "Documentation for the installation of optical flow sensors on the ardupilot." http://copter.ardupilot.com/wiki/common-optional-hardware/common-optical-flow-sensors-landingpage/common-mouse-based-optical-flow-sensor-adns3080/.

[22] "ServoBlaster source code." https://github.com/richardghirst/PiBits/tree/master/ServoBlaster.

[23] 3D Robotics, "3DR Pixhawk." http://3drobotics.com/pixhawk/, 2015.

[24] RC-Groups online community, "DJI NAZA GPS communication protocol - NazaDecoder Arduino library." http://www.rcgroups.com/forums/showthread.php?t=1995704, 2013.

Appendix A ArduPilot (Pixhawk) Proposal

ENGINEERING PROPOSAL
Hexacopter flight controller upgrade
24 April 2014

This document outlines the proposal for replacing and upgrading the flight controller used on the hexacopter to a 3DR Pixhawk [1] running the open-source flight control software ArduPilot [2]. This marks a significant departure from the current 'black-box' NAZA flight controller. The primary motivation for replacement is based on the strong concern that the current system is not sustainable: The current system restricts any automation improvements, provides no value to the wider community, and limits opportunities for future students.

The key benefits of switching to ArduPilot are:

- Tighter integration and feedback of the hexacopter state with the automation software;
- Avoiding sensor duplication
- Access to existing ground control and simulation software
- Well documented communication interfaces
- Wide community support
- Already being used in the research field.

ArduPilot was used in the winning entries in the 2012 and 2014 Canberra outback challenges [3, 4] and is the platform of choice in the forum DIYdrones.

The Pixhawk will use the MAVLink communications protocol to communicate with the Raspberry Pi. It is acknowledged that there are risks associated in the switchover, which will be discussed and addressed in this document.

1 Scope

The scope of this project involves:

1. Removing the NAZA-M V1 flight controller, power monitor (PMU) and GPS/Compass module
2. Removing the autonomous/manual switching circuit
3. Installing the 3DR Pixhawk, uBlox Neo-7m GPS/Compass module, power monitor, LED and wireless telemetry kit
4. Reworking the wiring harness between the flight controller and the Raspberry Pi
5. Reconfiguring the automation software to interface with the Pixhawk

2 Feature comparison

Feature	**Old (NAZA) system**	**Proposed (ArduPilot) system**
Failsafe (return to base) mode	Yes.	Yes.
Low-power failsafe	Yes, voltage monitor.	Yes, voltage and current monitor.
Auto/Manual switch	Yes (via switching circuit).	Yes, built-in to Pixhawk.
Buzzer	Yes, but not controllable. Pi has separate buzzer.	Yes, controllable. Pi can also have separate buzzer as before.
Fully autonomous (takeoff and waypoint travel)	No.	Yes. Can be programmed for fully automated missions.
IMU (pitch, roll, heading, altitude) and GPS data access	Limited (undocumented, unsupported, deliberately obfuscated interface). Current break-in cable to the GPS is unreliable and may cause damage to the GPS.	Yes. Well documented communications protocol (MAVLink).
Radio telemetry	Possible (not currently installed).	Yes.
Pan/Tilt gimbal for camera	Limited (tilt control only from the Pi).	Yes.
Simulation environment	No.	Yes, both hardware and software (HITL/SITL) with physics simulation [5]
Existing software tools	Flight assistant software performs calibration only.	1. Mission planner - handles system configuration, calibration and programming for autonomous flight [6]. 2. Tower - an Android mobile app for controlling the hexacopter [7]
Modifiable	No, closed system.	Yes. It is possible to modify all aspects of the system (open-source), but not recommended for our application without thorough testing.
Autonomous control from the Raspberry Pi	Yes, via the switching circuit and PWM output.	Yes, digital control via MAVLink (a communication protocol).
Built-in support for the PIKSI GPS	No.	Yes, but it is unclear how mature this support is at this stage.

2.1 Further discussion of features

2.1.1 Failsafe

The ArduPilot configuration has a fail-safe mode equivalent to the NAZA, activated in exactly the same way. Once configured, the flight computer will return-to-land on loss of hand-held transmitter signal. The failsafe can be configured as situations demand.

2.1.2 Modification of the flight controller

While it is possible to modify ArduPilot itself, changes to the low-level controls is not recommended nor required. This option will remain open for future work but will require extensive testing. The ArduPilot's default flight modes are sufficient for the scope of our work.

2.1.3 Camera gimbal

The pan servo cannot be controlled by the Pi due to a limitation of the NAZA. This is an out-of-the-box feature on the ArduPilot.

2.1.4 Sensor availability

The Raspberry Pi cannot access pan or tilt data as collected by the NAZA. Any pan/tilt data received will be different to the NAZA, causing the Pi to fight the NAZA. MAVLink provides access to the GPS and IMU data directly from the flight computer, which allows both the QStarz GPS and the XSens IMU to be removed. MAVLink also reports remaining battery capacity, which will allow us to write software that issues warnings and aborts missions intelligently.

2.1.5 Telemetry downlink

A new redundant data channel will permit monitoring outside range of WiFi, or in the case of server failure, which has caused near misses in the past.

The ArduPilot software supports fully autonomous missions, with instruction sent over this link. However, we expect to maintain the availability of manual override at all times and will configure the fail-safe to reflect this.

2.1.6 Control debugging

Using an external monitor interferes with the current control interface between the NAZA and Raspberry Pi. The proposed modification will save a great deal of time and effort in working around this problem.

The PWM signal from ServoBlaster must be calibrated with the NAZA flight control software. With no hardware PWM outputs, exact calibration with the NAZA flight computer is difficult. This is not an issue using MAVLink as it is a digital protocol.

2.1.7 Additional sensor compatibility

The ArduPilot community has added support for a wide variety of sensors over a wide variety of interfaces. This support includes integration into the navigation Kalman filter where appropriate.

2.1.8 Forward compatibility

DJI has released two new flight computers since the NAZA-M V1. The NAZA-M V1 is still supported, but the latest firmware updates require a CAN bus expander valued at A$80 to make use of any new features. Very few of these new features address our current concerns.

The Pixhawk was designed to replace the APM2.6 as the firmware files became too large for the 8-bit processor it carried. It was designed to be largely future-proof, boasting a generous amount of ram, flash and CPU power. Even if the ArduPilot project out-grows the Pixhawk, it is likely to have continued support and backported features, such as the APM2.6 owners currently enjoy.

2.1.9 Portability to future platforms

The ArduPilot project began with fixed wing aircraft and has since been extended to various configurations of multi-rotor, Helicopters, Rovers and Boats. Any work done building functionality against ArduCopter (the multirotor firmware) is immediately relevant to almost any other autonomous platform.

2.2 Alternative solutions

2.2.1 Upgrade the current flight controller

As a commercial system, there is *no* official interface that allows for automation or access to any of its sensors. In the past, this project achieved automation by using a remotely controlled switching circuit and the Raspberry Pi to emulate the PWM signals of joystick commands (aileron, elevator and rudder controls). This approach also requires the Raspberry Pi to have its own set of sensors, where progress would be bogged down in trying to interface with the low-level components and having to re-invent, test and tune basic control functions.

Although there is some limited upgrade capability (upgrading to the PMU V2), which would provide access to the IMU data, this is only possible through an unofficial, undocumented, reverse-engineered system. Even with this extra information, this does not address the other issues outlined in terms of community support, the camera gimbal, telemetry and the existing level of software available for the ArduPilot platform.

2.2.2 APM 2.6

The APM2.6 is a known-good ArduPilot board. The hardware is cheaper than the Pixhawk, but it is version capped as of last year. This board is not future-proof and will not benefit from new features in ArduPilot.

2.2.3 MultiWii

MultiWii is another general purpose open-source flight controller. Originally developed to use the gyroscope and accelerometer system from the Nintendo Wii game controller, it has developed into a general flight controller that operates on the Arduino platform. Community support for this platform is less than that for the ArduPilot. The platform also has lesser specs than the out-moded APM2.6.

2.2.4 Paparazzi

Paparazzi is another open-source flight controller system, which has been developed since 2003. It is an older project that appears to be extremely versatile, but not very beginner friendly. Their focus appears to be on modularity and wide applications. The community encourages significant modification of the core flight control software, which is likely to remain out of scope for this platform. Many of the demonstrated applications are single-purpose scientific flights.

2.2.5 OpenPilot

The OpenPilot appears to have a strong community backing for FPV and acrobatics, but the ArduPilot appears to have a sample of simpler flight modes and a stronger autonomous focus, and a stronger Australian community. The OpenPilot hardware does support ArduCopter, but does not have the extensive feature set of the Pixhawk, nor its sensor redundancy.

3 Risks

3.1 Risk matrix

	Very unlikely	Unlikely	Possible	Likely	Very Likely
Negligible	Very low	Very low	Very low	Low	Low
Minor	Very low	Very low	Low	Low	Medium
Moderate	Low	Low	Medium	Medium	High
Significant	Medium	Medium	High	High	Very high
Severe	Medium	High	High	Very high	Very high

3.2 Risk register

No.	Description	Probability	Impact	Rating	Mitigation	Contingency
1.1	The UAV community moves to a different platform	Unlikely	Moderate	Low	Many people, including developers have invested in ArduPilot compatible hardware which represents a small community lock-in	Find alternative, compatible flight controller software
1.2	Unforeseen hardware incompatibilities	Possible	Moderate	Medium	Conduct prior investigation and identify any potential hardware issues	Consult community for potential fix. OR replace incompatible hardware
2.2	Late deliveries or manufacturing faults	Possible	Significant	High	Request priority delivery, AND Much of the integration testing can be performed with an existing ArduPilot owned personally by a team member.	Source parts elsewhere

3.3 Further risk comments

Risk 1.1 - The UAV community moves to a different platform

The developer forums are extremely active, and new features are being added regularly. Of the open-source systems discussed, ArduPilot is the most actively developed [8, 9, 10, 11]. If an open-source flight controller is to be used, it makes sense to use the most actively developed platform to avoid project stagnation. Should ArduPilot development cease, there are a number of compelling platforms with a great deal of support - OpenPilot, MultiWii and Paparazzi.

Consequences
External development of the ArduPilot software would halt, but it would still be available in its current state. Being an open source project, many of the alternative drone software projects already support the Pixhawk hardware, just as ArduPilot has been ported to hardware from other projects.

Risk 1.2 - Unforeseen hardware incompatibilities

The Hexacopter has 30A Opto ESCs and a Futaba R7008SB receiver [12]. Support for these components with the ArduPilot has been demonstrated in other UAV projects, so hardware incompatibility is unlikely. The Raspberry Pi has been demonstrated to work with ArduPilot [13].

Consequences
Hardware incompatibilities would result in additional debugging work and lost time. It could also result in the replacement of other hardware, or the purchase of additional interfacing hardware.

Controls
Preliminary research into each subsystem of current hexacopter.

Risk 2.2 - Late deliveries or manufacturing faults

Online purchases carry the risk of components not arriving on time. It may also make it more difficult to return components, should they be faulty.

Consequences
Late deliveries would lead to lost time, potentially putting the project on hold or incurring additional costs.

4 Costs

Component	Cost ($AUD)	Source	Comments
Pixhawk	$256	3DR store	
UBlox GPS + Honeywell Compass	$41	eBay	
Telemetry	$32	eBay	Not strictly necessary, but allows in-flight full recovery after a complete server crash
Indicator LED	$12	eBay	

Total cost: A$341 (24/04/2015)

5 Conclusion

We fully expect the components listed above to serve as a drop-in replacement for the NAZA and other ancillary components.

The removal of the NAZA flight computer and the inclusion of the Pixhawk radically simplifies the wiring harness, removing many of the components and links that the 2014 team complained about being ‘unreliable’ and the cause of multiple crashes.

The move to the Pixhawk will greatly increase the maintainability of the platform and significantly widen the plausible scope of any future work with this hardware.

References

[1] 3D Robotics, "3DR Pixhawk." http://3drobotics.com/pixhawk/, 2015.

[2] DIY Drones team, "ArduPilot." http://ardupilot.com/, 2015.

[3] C. Anderson, "CanberraUAV (ArduPlane) wins Outback Challenge!." http://diydrones.com/profiles/blogs/canberrauav-arduplane-wins-outback-challenge, 2012.

[4] G. Mortimer, "Congratulations Canberra UAV, stunning Outback Challenge win." http://diydrones.com/profiles/blogs/congratulations-canberra-uav-stunning-outback-challenge-win, 2014.

[5] Ardupilot community, "Simulation." http://dev.ardupilot.com/wiki/simulation-2/, 2015.

[6] M. Oborne, "Mission planner home." http://planner.ardupilot.com/, 2015.

[7] 3D Robotics, "Tower (DroidPlanner 3)." https://play.google.com/store/apps/details?id=org.droidplanner.android, 2015.

[8] A. Tridgell, R. Mackay, P. Hickey, *et al.*, "ArduPilot source code." https://github.com/diydrones/ardupilot/, 2015.

[9] OpenPilot Foundation, "OpenPilot source code." https://github.com/openpilot/OpenPilot, 2015.

[10] F. Ruess, G. Hattenberger, C. D. Wagter, *et al.*, "Paparazzi source code." https://github.com/paparazzi/paparazzi, 2015.

[11] MultiWii Authors, "MultiWii source code." https://github.com/multiwii/multiwii-firmware, 2015.

[12] NoLimitLab, "Setting up Futaba 14SG and R7008SB to work with the pixhawk." https://nolimitlab.wordpress.com/resources/setting-up-futaba-14sg-and-r7008sb-to-work-with-the-pixhawk/, 2015.

[13] ArduPilot community, "Communicating with Raspberry Pi via MAVLink." http://dev.ardupilot.com/wiki/companion-computers/raspberry-pi-via-mavlink/, 2015.