AutoMav

Project Heartbeat

Detailed Documentation
&
User Guide

Author: Siddhant Mahapatra

CONTENTS

References

- **Official Project Code Repository:** https://github.com/Robosid/Drone-Intelligence
- **Pixhawk Source:** https://pixhawk.org/
- **Cameron Roberts.** GPS Guided Autonomous Drone, University of Evansville, 2016
- **Firas Abdullah Thweny Al-Saedi.** Design and Implementation of Autopilot System for Quadcopter, Al-Nahrain University, 2015
- **Anitha G, Karthikeyan M, Karthic V.** Open Source Autopilots for Quadrotor Unmanned Aerial Vehicle, Department of Avionics, Madras Institute of Technology, Chennai, India, 2013
- **DiegoDomingos, GuilhermeCamargo, FernandoGomide.** Autonomous Fuzzy Control and Navigation of Quadcopters, University of Campinas, Campinas, Brazil, 2016
- **Md R Haque, M Muhammad, D Swarnaker.** Autonomous Quadcopter for product home delivery, Military Institute of Science and Technology, Dhaka, Bangladesh, 2014
- **Jin Kim, Yoon Suk Lee, Sang Su Han.** Autonomous flight system using marker recognition on drone, Chung-Ang University, Seoul, Korea, 2015
- **Andrea Patelli, Luca Mottola.** Model-based Real-time Testing of Drone Autopilots, Politecnico di Milano (Italy), +SICS Swedish ICT, 2017
- **Gordon Ononiwu, Arinze Okoye, James Onojo, Nnaemeka Onuekwusi.** Design and Implementation of a Real Time Wireless Quadcopter for Rescue Operations, Federal University of Technology Owerri, Nigeria, 2016
- **MAVLink Developer Guide:** https://mavlink.io/en/

Objectives

The objectives of this research are as follows:

- ➔ Develop an understanding of the drone's flight dynamics and Quadcopter motions by literature review and real world flying.
- ➔ Develop an understanding of a drone's mechanical structure and electronic components by literature review, real world flying, and system assembly and integration.
- ➔ Develop an understanding of autonomous flying principles and algorithms, by literature review and by conducting rudimentary autonomous test flights.
- ➔ Develop the skill to operate a Ground Control Station, upon an understanding of its capabilities and working.
- ➔ Develop an understanding of the fundamentals of the MAVlink protocol.
- ➔ Develop an understanding of the Autopilot Hardware (Flight Controller) and its simulation in SITL environment.
- ➔ Developing a custom Autopilot firmware including the development and implementation of various flight modes in simulation and real world, to have a stable and a semi-autonomous flight.
- ➔ Develop a custom API called **Advanced Autonomous Guidance Assisted Flying System (AAGAFS),** that allows programs to run on an onboard companion computer and these Onboard applications significantly enhances the autopilot, adding greater intelligence to vehicle behaviour.
- ➔ Finally, write and test algorithms and programs to deploy autonomous flight capabilities in a quadcopter. The official code repository for this project is stored in https://github.com/Robosid/Drone-Intelligence

Chapter 1

Introduction

This report expounds my work at an incubated tech startup, a drone outsourcing company based in Bangalore, India. I was solely responsible to impart intelligence and autonomous behavior to a drone powered by Pixhawk and Pi3B, communicating via the MAVlink protocol. This report will get off the ground by elucidating the key concepts involved in the construction of a drone and will eventually make headway into the algorithms and pipelines developed. The drone frame used in this work is of quad type.

1.1 Quadcopter Essentials:

This section expands upon essential concepts based on basics of a Quadcopter, required to comprehend the essence of this work.

1.1.1 What is a quadcopter?

A quadcopter, or multirotor, drone, or quadrotor, is a simple flying mechanical vehicle that has four arms, and in each arm there is a motor attached to a propeller. Multicopters with three, six or eight arms are also possible, but work on the same principle as a quadcopter. Two of the rotors turn clockwise, while the other two turn counter clockwise. These use independent variation of the speed of each rotor to achieve control. By changing the speed of each rotor it is possible to specifically generate a desired total thrust; to locate for the centre of thrust both laterally and longitudinally; and to create a desired total torque, or turning force. Quadcopters are aerodynamically unstable, and require a flight computer to convert the input commands into commands that change the RPMs of the propellers to produce the desired motion.

Quadcopters differ from a helicopter or a fixed wing aircraft in the way they generate lift and control forces. For an aircraft the lift is generated by the wings, but in a quadcopter the lift is generated by the propellers. A helicopter uses its main rotor to generate lift, but also have the ability to vary the pitch of the rotor blades to generate control forces.

The quadcopter concept is not new. Manned quadcopter designs appeared in the 1920s and 1930s, but these early concepts had bad performance, a high level of instability, and required a lot of pilot inputs. The advancement of electronic technology in flight control computers, coreless or brushless motors, smaller microprocessors, batteries, accelerometers, cameras, and even GPS systems made it possible to design and fly quadcopters. At a small size, quadcopters are cheaper to build and more durable than conventional helicopters due to their mechanical simplicity. Their smaller blades are also advantageous because they possess less kinetic energy, reducing their ability to cause damage. For small-scale quadcopters, this makes the vehicles safer for close interaction. It is also possible to fit quadcopters with guards that enclose the rotors, further reducing the potential for damage. However, as size increases, fixed propeller quadcopters develop disadvantages over conventional helicopters. Increasing blade size increases their momentum. This means that changes in blade speed take longer, which negatively impacts control. Helicopters do not experience this problem as increasing the size of the rotor disk does not significantly impact the ability to control blade pitch.

1.1.2 Flight Dynamics

Each rotor produces both a thrust and torque about its center of rotation, as well as a drag force opposite to the vehicle's direction of flight. If all rotors are spinning at the same angular velocity, with rotors one and three rotating clockwise and rotors two and four counterclockwise, the net aerodynamic torque, and hence the angular acceleration about the yaw axis, is exactly zero, which mean there is no need for a tail rotor as on conventional helicopters. Yaw is induced by mismatching the balance in aerodynamic torques (i.e., by offsetting the cumulative thrust commands between the counter-rotating blade pairs).

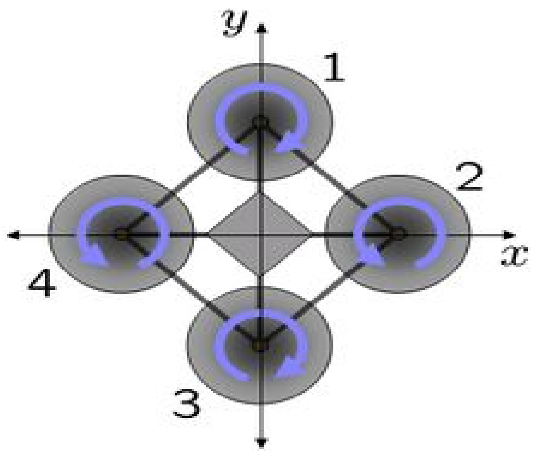


Figure 1.1: Schematic of reaction torques on each motor of a quadcopter aircraft, due to spinning rotors. Rotors 1 and 3 spin in one direction, while rotors 2 and 4 spin in the opposite direction, yielding opposing torques for control.

1.1.2.1 Drag

‘Drag’ is essentially a mechanical force that opposes the motion of any object through a fluid. Aerodynamic drag on multirotors is generated due to the difference in velocity between the multirotor and the air. This occurs is only if the quadcopter/multirotor is in motion (going up, down, forward, backward and taking turns) relative to the air. If the multirotor is stationary, there is no drag.

This drag force and the **weight** of the multirotor is what is needed to be overcome, in order for the craft to get up in the air and move around.

1.1.2.2 Thrust

‘Thrust’ is the force generated by the propellers of the multirotor, in order to work against one of the forces that needs to be overcome: the drag. Note that the thrust force is not the main force responsible for getting the multirotor up in the air. Instead, it is the force that lets the multirotor travel within the air, which is a fluid, overcoming its drag resistance.

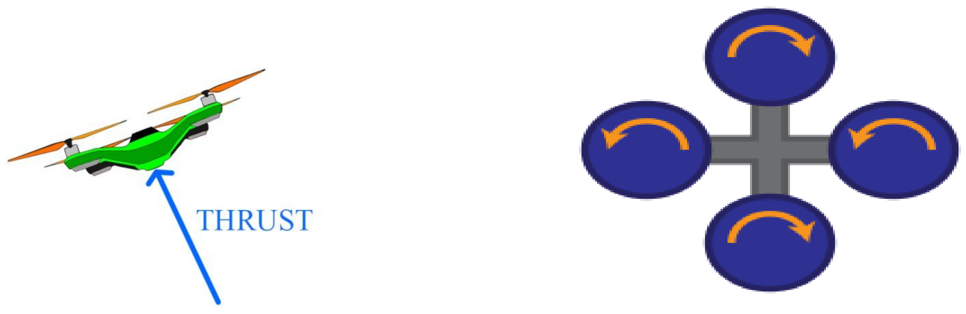


Figure 1.2: Illustrating Thrust force (Left) and quadrotor hovers or adjusts its altitude by applying equal thrust to all four rotors (Right)

1.1.2.3 Lift

The lift is the force that acts against the weight of the craft, taking it up in the air. The following are responsible for the lift in a wing:

1. Newton's third law (every action has an equal and opposite reaction) – generates a lift in a wing at the bottom, since the mass of air is pushed down and back (lift and drag).
2. Bernoulli's explanation is incomplete, but the pressure difference between the air at the top and at the bottom due to the Coanda Effect generates a lift towards the lower pressure at the top (read more in the quadcopter blade rotation post).

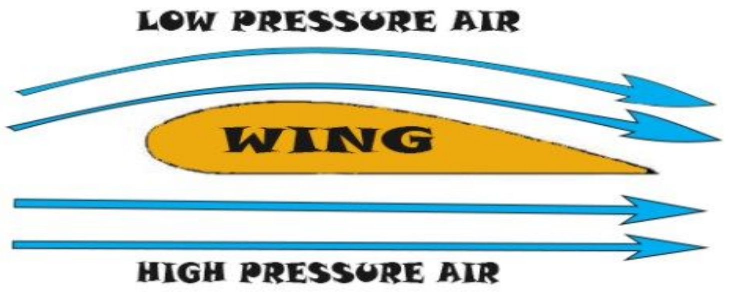


Figure 1.3: Illustrating the lift principle

The propellers on the multirotor generates a 'lift' force using similar principles (pushing the air downwards and the difference in air pressure). In order for the multirotor to get off the ground and be able to hover and fly around, this force must be greater than the weight of the craft.

1.1.3 The Radio Controller

The quadcopter used in this work has a 12 channel controller that sends commands to the drone to affect its throttle, yaw, pitch and roll along with an array of auxiliary operations. The communications frequency used by most controllers is 2.4 GHz. This is also the typical frequency used for WiFi connections. Though it is unlikely that interference takes place, it may be something worth checking if communications lag or dropouts with the quadcopter, is experienced.

The radio transmitter and receiver is used to control the quadcopter. In order for the basic operation of the quadcopter, four channels (throttle, elevator, aileron rudder) are used.

Quadcopter can be programmed and controlled in many different ways. However the one that is most commonly used in this work, is either guided or stable mode. In guided mode, the gyroscope is used to control the quadcopter balance, it does not self-level.

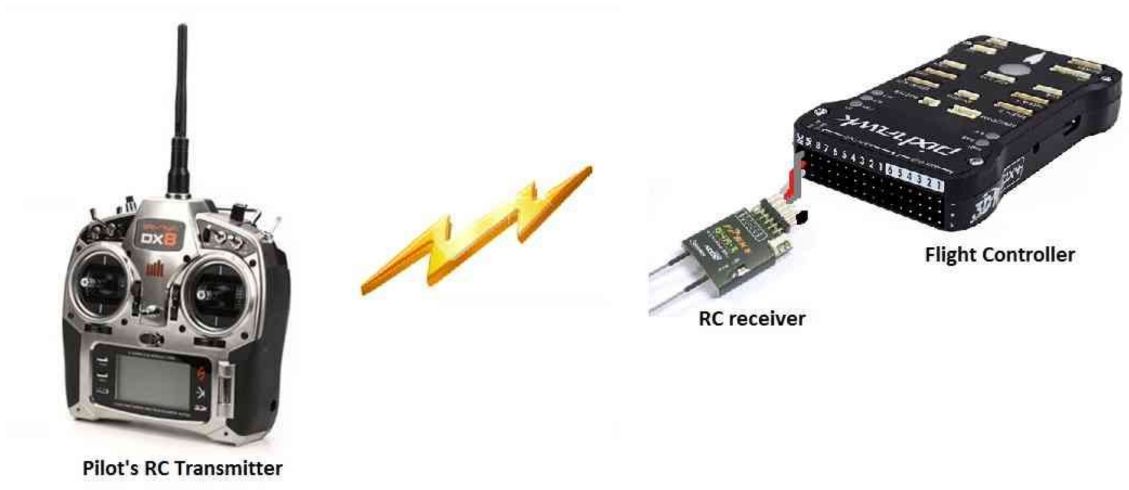


Figure 1.4: RC Operation

If switched to stable mode, the accelerometer gets activated, helping to stabilize the quadcopter. The speed of the 4 motors is adjusted automatically and perpetually to keep the quadcopter balanced.

A transmitter also has an FPV screen on which a camera mounted on the multirotor which beams video in real time.

The controller also has 4 trim buttons to allow minor corrections to the quadcopter’s flight behavior. With no control input, the drone drifts in a particular direction, applying trim for that control input in the opposite direction will remove the drift.

1.1.4 Hovering

Hovering takes place when the upward lift balances the downward force of gravity. To make the drone to higher, the lift force needs to be greater than the force of gravity. This is achieved by increasing RPM on all four propellers simultaneously, which produces more lift. To decrease the altitude, the RPM is decreased on all four propellers simultaneously. On the controller, this is accomplished by pushing the left stick up or down, which increases or decreases the rotor RPM, which makes the quadcopter go up or down.

In order to move the drone left, right, forwards, and backwards, the angle of the lifting force is changed so it has a vertical and horizontal component. The vertical component still serves to keep the quadcopter in the air, while the horizontal component allows produces control thrust.

1.1.5 Coaxial configuration

In order to allow more power and stability at reduced weight, a quadcopter, like any other multirotor can employ a coaxial rotor configuration. In this case, each arm has two motors running in opposite directions (one facing up and one facing down).

Chapter 2

QuadCopter Motion

To make quadcopter move forward or backwards, the pitch is adjusted using the right stick on the controller. What happens is that the front propellers decrease RPMs, while the back propellers increase RPMs. Now the lift force has a horizontal component which results in moving the quadcopter forward. The opposite happens to make the quadcopter go backwards. The front propeller RPM is increased, while the back propeller RPM is decreased.

To move the quadcopter move sideways, a similar change in RPM takes place, but this time it is on the left and right propellers. To make the drone move to left, the lift force is pointed slightly to the left. This is done by decreasing the RPM on the left rotors, and increasing the RPM on the right side. A similar change in the rotor RPMs takes place to move the quadcopter to the left.

2.1 Yaw Maneuvers

The ‘yaw’ or ‘rudder’ is a rotation movement of the quadcopter. In this case, the rotation speed of diametrically opposing pairs of motors are increased or decreased, varying the torque in the direction of rotation of that pair (remember that diametrically opposing motors in a quadcopter rotate in the same direction), causing the quadcopter to rotate in the direction of the increased torque. Yaw is bit trickier to visualize. On a helicopter, there is a tail rotor. It is there in order to keep the body of the helicopter from rotating. If the main blades are rotating anti-clockwise, like in most helicopters, the fuselage will start rotating clockwise due to torque reaction. This is a consequence of Newton’s Third Law of dynamics. The tail rotor balances out the torque reaction on the helicopter fuselage and prevents the fuselage from rotating.

On a quadcopter, if all four propellers rotated the same way, then the body of the quadcopter would rotate the opposite way. But it obviously doesn’t. This is because two of the propellers rotate clockwise, and the other two rotate anti-clockwise. So one set of propellers produces a torque in one direction, but the other two propellers are producing a torque in the opposite direction. These cancel out, so the quadcopter does not rotate.

For a yaw maneuver, the quadcopter has to rotate. This is done by reducing the RPM on one set of rotors, while increasing the RPM on the other set of rotors. Now there is a net torque in one direction, so the quadcopter rotates. To produce the opposite rotation, the RPMs on the propellers is simply reversed. The direction of the yaw maneuver is controlled by the left stick on the controller.

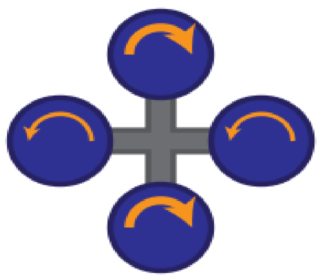


Figure 2.1: A quadrotor adjusts its yaw by applying more thrust to rotors rotating in one direction.

2.2 Pitch Maneuvers

The ‘pitch’ control tells the quadcopter whether to fly forward or backward. In order to pitch forward for example, the speed of the motors at the rear of the quadcopter must increase, relative to the speed of the motors on the front. This ‘pitches’ the nose (front) of the quadcopter down, resulting in the forward movement. This is achieved by either increasing the speed of the rear motors or decreasing the speed of the front motors. Conversely, in order to ‘pitch’ backwards, the speed of the motors at the front of the quadcopter must increase relative to the speed of the motors at the back.

2.3 Roll

The ‘roll’ control tells the quadcopter to move side to side. In order to ‘roll’ to the right for example, the speed of the motor at the left of the quadcopters must increase, relative to the speed of the motors on the right. This 'rolls down' the right side of the quadcopter, resulting in a sideways swaying movement.

Like pitch, this is achieved by either increasing the speed of the left motors or decreasing the speed of the right motors. Conversely, in order to 'roll' left, the speed of the motors of the right of the quadcopter should increase relative to the speed of the motors at the left.

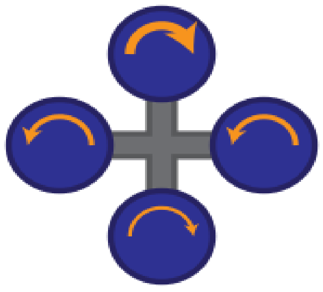


Figure 2.2: A quadrotor adjusts its pitch or roll by applying more thrust to one rotor and less thrust to its diametrically opposite rotor.

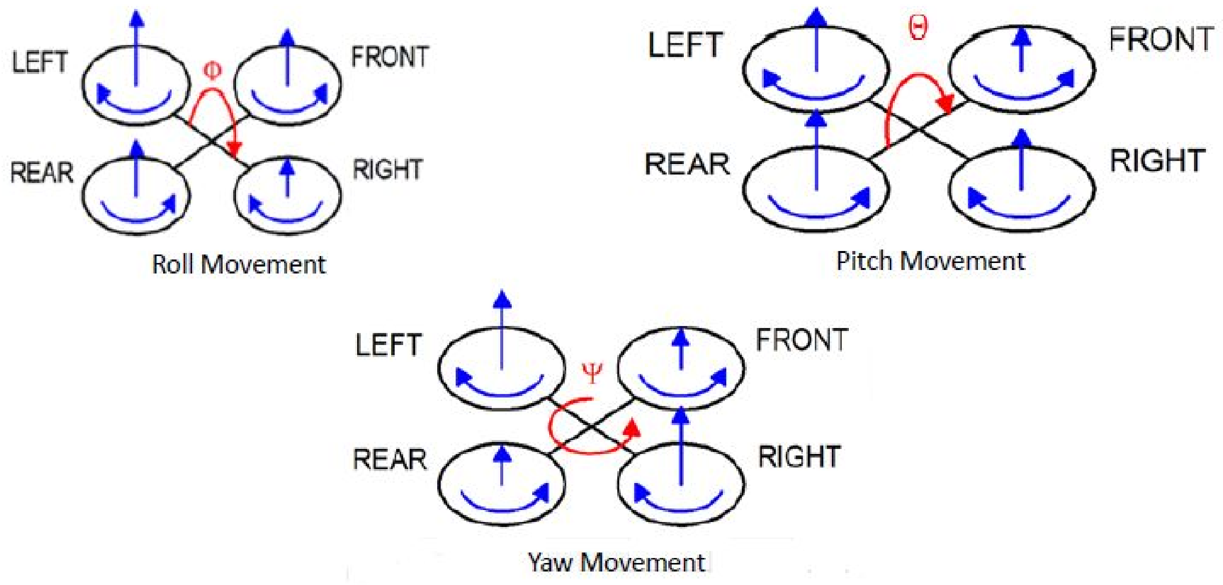


Figure 2.3: Illustrations for Roll, Pitch and Yaw Movements respectively

Chapter 3

Mechanical structure

The main mechanical components used for construction are the frame, propellers (either fixed-pitch or variable-pitch), and the electric motors. For best performance and simplest control algorithms, the motors and propellers have been placed equidistant. Recently, carbon fiber composites have been deployed due to their light weight and structural stiffness.

The electrical components used to construct the quadcopter are similar to those needed for a modern RC helicopter. They are the electronic speed control module, on-board computer, flight controller board, and battery (or fuel cell). An advanced RC transmitter has been also used to allow for human input.

3.1 The body/frame

The frame or body is what holds everything together. It is designed to be strong and lightweight and consists of a center plate where the main flight controller, on-board computer, battery and sensors are mounted and arms where the motors and motor controllers are mounted. It is made of carbon fibre.

3.2 Motors

The motivation behind using motors is to turn the propellers, which is responsible for providing thrust for countering gravity and drag. Every rotor is controlled separately by a speed controller. Motors are the the primary force behind how this quadcopters flies.

They are somewhat like typical DC motors in the sense that coils and magnets are utilised to drive the shaft. The brushless motors do not have a brush on the shaft that deals with iterating the power in the coils, hence, the ‘brushless’ reference.

The brushless motors have 3 coils on the inside center of the motor, which is settled to the mounting. On the external side, it contains multiple magnets mounted to the cylindrical structure that is appended to the turning shaft.

Hence, the coils are fixed and there is no need for brushless. Brushless motors turn a lot quicker and utilize less power at the same speed relative to DC motors. Unlike DC motors, they don't lose power in the brush-transition, so it is a lot more vitality productive.
The Kv (kilovolts) – rating in a motor demonstrates how various RPMs (Revolutions each moment) the motor will do per volt. The higher the kV rating is, quicker the motor rotates at a steady voltage. Motors with 1500 Kv has been utilized in this work.

3.3 ESCs

Motors spin, but in order to fully understand how quadcopters work, it is important to understand how the motor is controlled. The electronic speed controller or (ESC) is what tells the motor how to spin. It is responsible for controlling the rate at which the motor it is connected to, spins. Since the multirotor motors are supposed to spin at variable speeds, depending on control inputs, ESCs are crucial. In this multirotor, each motor has an associated ESC connected to it.
The ESCs are connected to the batteries via the power distribution board within the multirotor frame. The ESC used comes with a battery eliminator circuit (BEC) which acts as a voltage regulator, allowing other electronic components like the flight controller and receiver to power up without connecting them directly to a battery.

3.4 Batteries

Lithium Polymer (LiPo) battery is used as the power source for controlling multirotors. The principle explanation behind this is on the grounds that they are rechargeable and ordinarily have expansive limits.
LiPo batteries have discharge rates sufficiently expansive to control even probably the most taxing multirotors. This settles on them the favoured decision over different choices, for example, the Nickel Cadmium (NiCd) battery. This is likewise the essential reason they can be a genuine fuel source for multirotors.

3.5 The Propellers

It is important to understand that there are two sets of propellers, and two set of motors that rotate in the opposite direction. The pitch on the two sets of propellers are different. So if one switches a set of opposite propellers, they will be blowing air up instead of down. This results in a downward force on the quadcopter leading to drone flipping, and not flying. The propeller has a mark indicating which set of motors that it belongs to. Quadcopter propellers come in a huge variety of materials, dimensions and prices from bottom to top of the range. Generally, cheaper props are less precisely manufactured and more prone to creating vibration.

This applies especially to the relatively larger end of the prop spectrum, with differences becoming less perceptible for smaller craft. A Prop Balancer has also been used to check the quadcopter propellers every few flights.

There are three simple measurements to keep in mind while understanding the propeller choices:

3.5.1 Length – The first is length (Diameter), usually given in inches. The length of a propeller is the diameter of a disc the prop makes when it's spinning.

The higher the Kv rating of the motors, the smaller the props need to be. Smaller props allow for greater speeds, but reduced efficiency. A larger prop setup (with correspondingly low Kv motors) is easier to fly steadily. It also uses less current and lifts more weight. It primarily depends on the client's drone requirements.

3.5.2 Prop Pitch – This second measurement is also very important. Prop dimensions used in this work is in the form 21 x 7.0 inch (533 x 178 mm). The first number refers to the propeller length as above. The second is pitch, defined as the distance a prop would be pulled forward through a solid in a single full revolution. For example this propeller with a 7.0 inch pitch would move forward 7.0 inches in one revolution.

3.5.3 Bore – The last is known as bore measurement, which is simply the size of the hole in the center of the prop. This must be matched to the shaft of the chosen motors. Adapters have been used to downsize a prop's bore.

3.5.4 Self Locking

– The props used is called “Self locking”, because 2 motors are spinning clockwise and the other 2 are spinning counter clockwise. By using propeller threads that are the opposite to motor spin direction, the props automatically lock themselves down and won’t come loose when flying.

3.5.5 Large Or Smaller Quadcopter Propellers

The greater the pitch, then the higher the thrust and necessary motor output. Typically, multi-rotors use props with pitches in the range of 3 to 5 inches. Lower pitches are more efficient. The larger the prop (either increasing diameter, or pitch or both), the more energy it takes to spin it. However, larger propeller or higher pitch length will increase the aircraft’s speed but also use more power.

Props with smaller diameter or pitch are used as they spin faster (higher RPM), because the motor doesn’t need to work as hard to spin it so it pulls less current. They tend to run smoother and feels more responsive to the sticks. The faster change of RPM due to less inertia helps stability of the quadcopter.

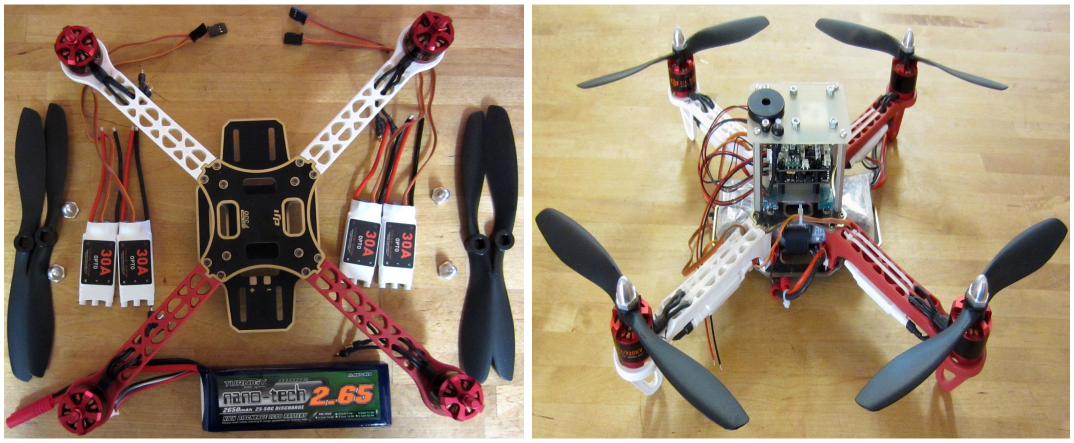


Figure 3.1: Mandatory mechanical and electronic components required for a flight, displayed in its unassembled (Left) and assembled states (Right)

Chapter 4

Autonomous Flight Capability

This Quadcopter can fly autonomously. Many modern flight controllers allows the user to mark "way-points" on a map, to which the quadcopter will fly and perform tasks, such as landing or gaining altitude. The PX4 autopilot system, is a software/hardware combination in development since 2009, that has since been adopted in this work to give the quadcopter flight-control capabilities. Other flight applications not included in this work, include crowd control between several quadcopters where visual data from the device is used to predict where the crowd will move next and in turn direct the quadcopter to the next corresponding waypoint.

4.1 Flight Control System

The flight controller is the mind or 'brain' of the multirotor. This board is what sits at the center, controlling the firmware within the ESCs, consequently controlling the spin rate of the motors. The quadcopter flies and moves by changing the RPMs of each propeller. So when a stick on the controller is moved, this command gets converted into the proper commands each of the four motors on the quadcopter. This is done by the flight control system. The purpose of the flight computer is to simplify the coordination of the control of all four propellers needed to make the drone fly. The flight control computer has the ability to connect to several other devices and sensors. The primary device that it connects to is the remote control receiver, which is linked to the remote transmitter.

This work revolves around an advanced quadcopter, where the flight controller also has a transmitter to communicate with the controller and with the on-board controller, to provide two way communication. Typical sensors included within the design of the quadcopter used in this work, include GPS, gyro compass, and barometer.

Now the central flight controller also takes information from IMU, Gyroscope, GPS modules and obstacle detection sensors on the quadcopter. It makes computation calculations using programmed flight parameters and algorithms, then sends this data to the electronic speed controllers.

In fact, the flight controller encompasses the IMU, GPS, Gyroscope and many more features to control the quadcopter flight and stability. It also has dual IMUs for redundancy and other safety features such as Return-To-Home.

4.2 Most important sensors used and their roles:

4.2.1 Accelerometer

The accelerometer measures the change in the object's speed (to tell whether it is going up or down). It senses both static gravity acceleration (which happens even when the object is not actively moving) and dynamic acceleration, to detect motion. The unit of measurement used is g (9.8 m/s^2) or in metre per second squared. Note that the accelerometer measures acceleration in three different axes in the 3D world (x, y and z axes). The accelerometer senses both **what direction the ground is,** by sensing the earth's gravitational pull and **linear motion**.

4.2.2 Gyroscope

Unlike the accelerometer, the gyroscope measures the rate of rotation of an object about its axis, in degrees per second or rotations per minute (RPM). The gyroscope is mounted on the quadcopter in a way that it is aligned with its axes, giving information on the orientation of the quadcopter. Three axes of rotations are measured (roll, pitch and yaw).

4.2.3 Inertial measurement unit (IMU)

In order to accurately measure the orientation, velocity and location of the quadcopter, an accelerometer or a gyroscope alone may not be enough. This is where the inertial measurement unit (IMU) comes in.

The IMU is a board that combines both multi axes gyroscope and accelerometer to get the best of both. The IMU may also contain a magnetometer to correct the errors in the gyroscope feedback.

4.2.4 Barometer

The barometer is essentially a pressure sensor that senses changes in air pressure. Air pressure changes with altitude. Barometer can detect even the smallest changes in pressure and so can be used to measure the altitude of the quadcopter. This is becoming increasingly important these days, with the advent of new regulations.

4.2.5 GPS

The global positioning system ('GPS') receives data from multiple satellites that are in orbit around the earth in order to pinpoint the geographical location of the quadcopter. With the GPS system in place, specific coordinates can be set for the quadcopter to fly to. GPS also enables the RTH (return to home) function which enables this craft to fly back to me in case it flies beyond my line of sight and beyond my controller's range.

4.2.6 Magnetometer

The magnetometer is used to measure the earth's magnetic field (like a compass). This sensor usually serves the purpose of correcting the drift of the gyro and also to serve as an ancillary to the GPS system.

Chapter 5

Ground Control Station

A custom built GCS has been developed in-house that can be used as a configuration utility or as a dynamic control supplement for this autonomous vehicle. Here are just a few things this GCS can do:

- Load the firmware into the flight controller that controls the vehicle.
- Setup, configure, and tune the vehicle for optimum performance.
- Plan, save and load autonomous missions into the autopilot with simple point-and-click way-point entry on Google or other maps.
- Download and analyze mission logs created by the autopilot.
- Interface with a PC flight simulator to create a full hardware-in-the-loop(HIL) UAV simulator.
- With our custom telemetry hardware, the pilot can:
    - Monitor the vehicle’s status while in operation.
    - Record telemetry logs which contain much more information the the on-board autopilot logs.
    - View and analyze the telemetry logs.
    - Operate the vehicle in FPV (first person view)

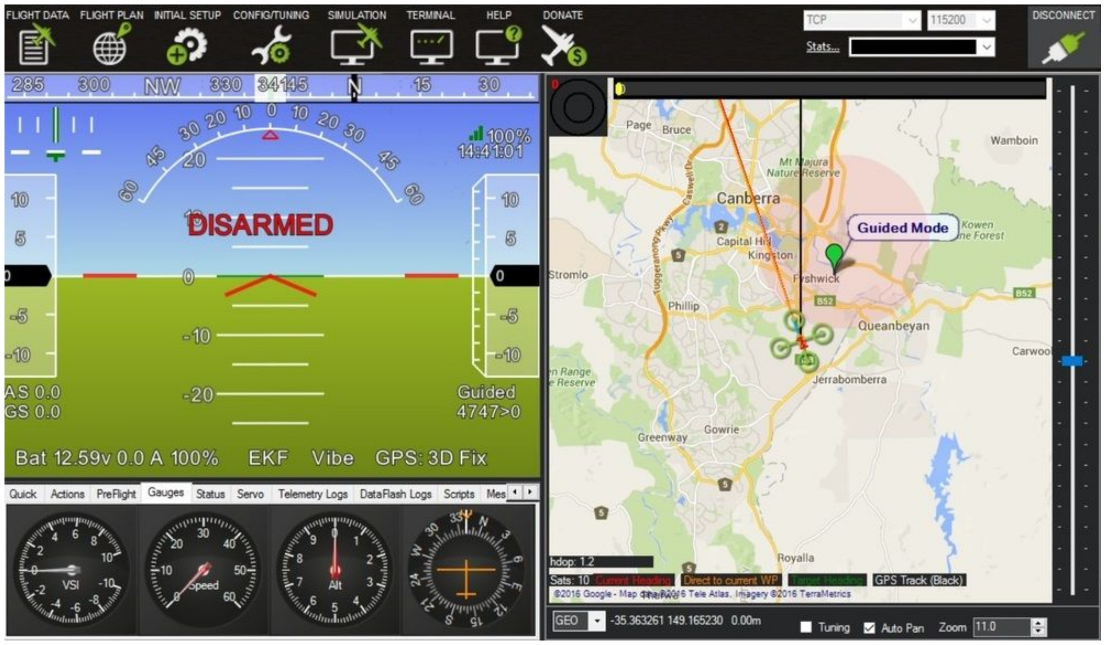
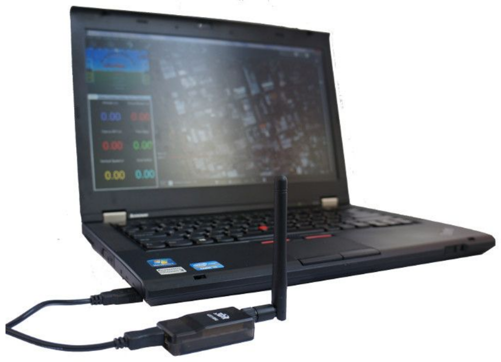


Figure 5.1 : Screenshot of the ground control station in action

5.1 Setting up the connection

To establish a connection to the on board flight controller, the communication method/channel has to be chosen, and the physical hardware and device drivers has to be set up. The PC and autopilot can be connected using USB cables, Telemetry Radios, Bluetooth, IP connections etc.



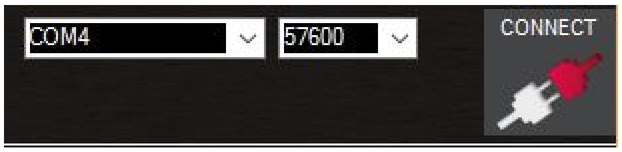
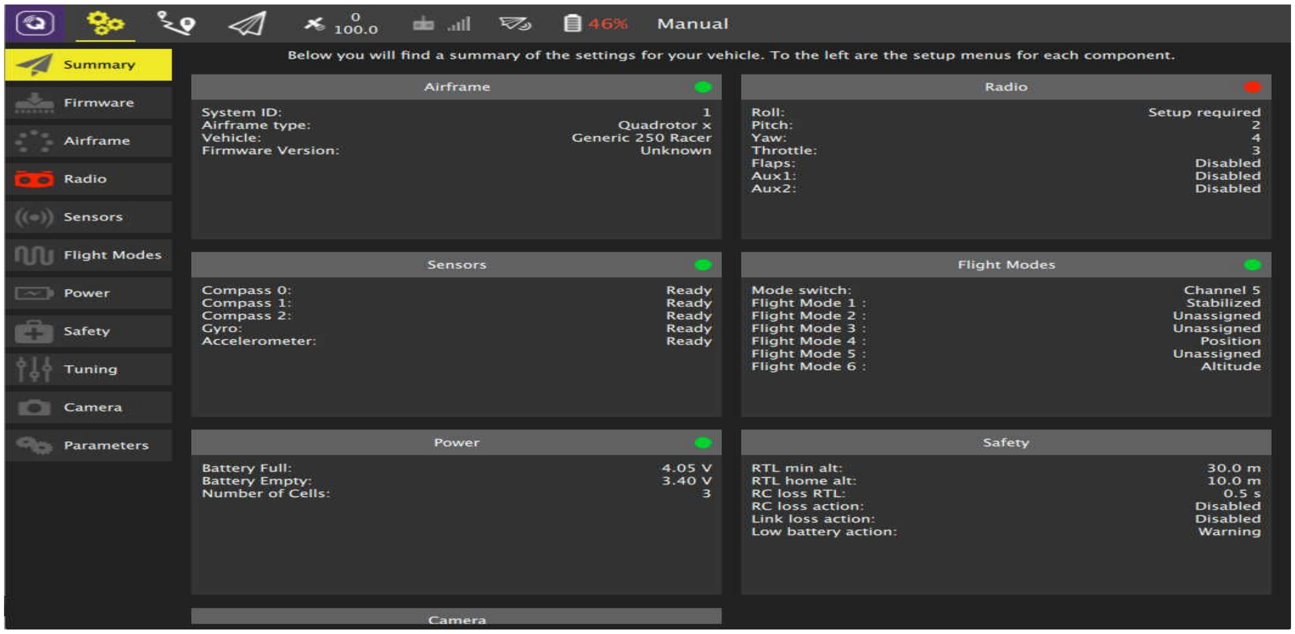


Figure 5.2 : Left: Connection using SiK Radio; Right: Setting the COM port and data rate for the connection

5.1.1 Initial Setup

Setup steps are designed to configure a new vehicle prior to first flight and/or tune a configured vehicle.



Below you will find a summary of the settings for your vehicle. To the left are the setup menus for each component.

Airframe	
System ID:	1
Airframe type:	Quadrotor x
Vehicle:	Generic 250 Racer
Firmware Version:	Unknown

Radio	
Roll:	Setup required
Pitch:	2
Yaw:	4
Throttle:	3
Flaps:	Disabled
Aux1:	Disabled
Aux2:	Disabled

Sensors	
Compass 0:	Ready
Compass 1:	Ready
Compass 2:	Ready
Gyro:	Ready
Accelerometer:	Ready

Flight Modes	
Mode switch:	Channel 5
Flight Mode 1 :	Stabilized
Flight Mode 2 :	Unassigned
Flight Mode 3 :	Unassigned
Flight Mode 4 :	Position
Flight Mode 5 :	Unassigned
Flight Mode 6 :	Altitude

Power	
Battery Full:	4.05 V
Battery Empty:	3.40 V
Number of Cells:	3

Safety	
RTL min alt:	30.0 m
RTL home alt:	10.0 m
RC loss RTL:	0.5 s
RC loss action:	Disabled
Link loss action:	Disabled
Low battery action:	Warning

Figure 5.3 :An overview of all the important setup options for the vehicle

5.2 Mission Planning with Waypoints and Events

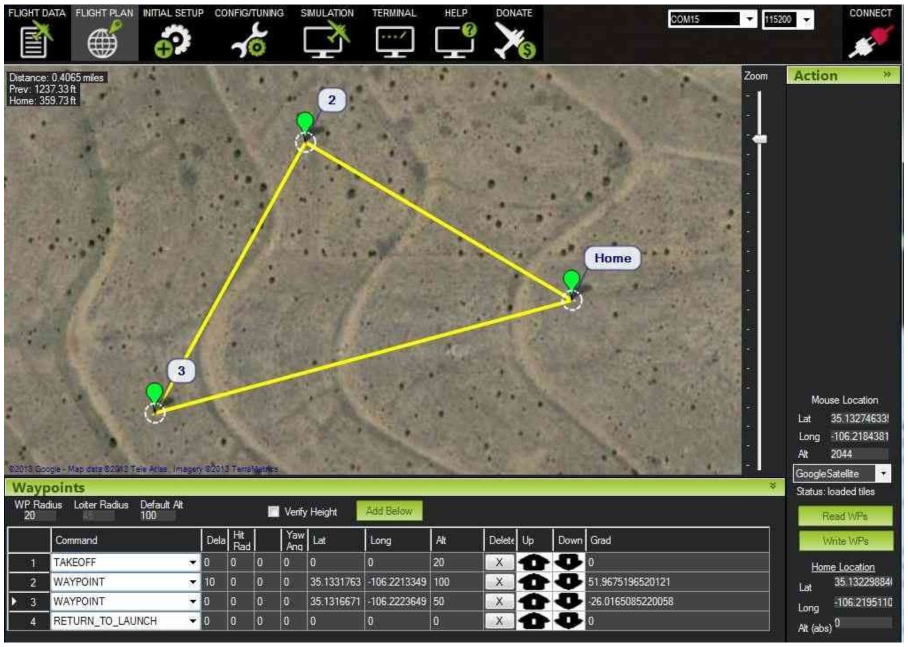
This section revolves around creating automated missions that will run when the Flight Controller is set to AUTO mode and describes generic waypoint setup for this copter.

5.2.1 Setting the Home Position

For our **Copter** the home position is set as the location where the copter was armed. This means if a RTL is executed in Copter, it will return to the location where it was armed, so the copter is always armed in the location it has to return to.

5.2.2 Mission Planning Example

In the screenshot below, our Copter mission starts with an auto takeoff to 20 meters altitude; then goes to WP 2 rising to 100 meters altitude on the way, then waits 10 seconds; then the craft will proceed to WP 3 (descending to 50 meters altitude on the way), then returns to launch. Since the default altitude is 100 meters, the return to launch will be at 100 meters. After reaching the launch position, the craft will land. The mission assumes that the launch position is set at the home position.



Waypoints

WP Radius 20 | Loiter Radius | Default Alt 100

	Command	Dela	Hit Rad		Yaw Ang	Lat	Long	Alt	Delete	Up	Down	Grad
1	TAKEOFF	0	0	0	0	0	0	20	X			0
2	WAYPOINT	10	0	0	0	35.1331763	-106.2213349	100	X			51.9675196520121
3	WAYPOINT	0	0	0	0	35.1316671	-106.2223649	50	X			-26.0165085220058
4	RETURN_TO_LAUNCH	0	0	0	0	0	0	0	X			0

Figure 5.4: Mission Planning Example

5.2.3 Features and Capabilities

Waypoints and other commands can be entered using this GCS. Lat and Lon can be entered by clicking on the map. Altitude is relative to the launch altitude/home position, so if 100m is set, for example, it will fly 100m above the user.

A default altitude can be set when entering new waypoints. It's also the altitude, RTL (return to launch) mode will fly at if Default Altitude hold is checked; if that is not checked, the aircraft will try to maintain the altitude it was at when RTL was switched on.

The GCS uses Google Earth topology data to adjust the desired altitude at each waypoint to reflect the height of the ground beneath. So if the waypoint is on a hill, if this option is selected the *the GCS* will increase the Altitude setting by the height of the hill.

Once done with the mission, it can be sent to the controller on-board and saved in it's EEPROM.

Multiple mission files can be saved to the user's local hard drive.

5.2.3.1 Special Features:

- Prefetch: Map data can be cached so there is no need of Internet access at the field.
- Grid: This allows to draw a polygon and automatically create waypoints over the selected area. Note that it does not do "island detection", which means if there is a big polygon and a little one inside of that, the little one will not be excluded from the big one Also, in the case of any polygon that partially doubles backs on itself (like the letter U), the open area in the center will be included as part of the flyover.
- The home location can be set to the current coordinates if the home location has to be pointing to the current location.
- Distance between waypoints can be measured. A dialog box will open with the distance between the two points.

5.2.4 Auto Grid

The GCS can create a mission, which is useful for function like mapping missions, where the aircraft should just go back and forth in a "lawnmower" pattern over an area to collect photographs.

To do this, a Polygon has to be selected and a box has to be drawn around the area to be mapped. After setting the altitude and spacing, the *GCS* will then generate a mission that looks something like this:

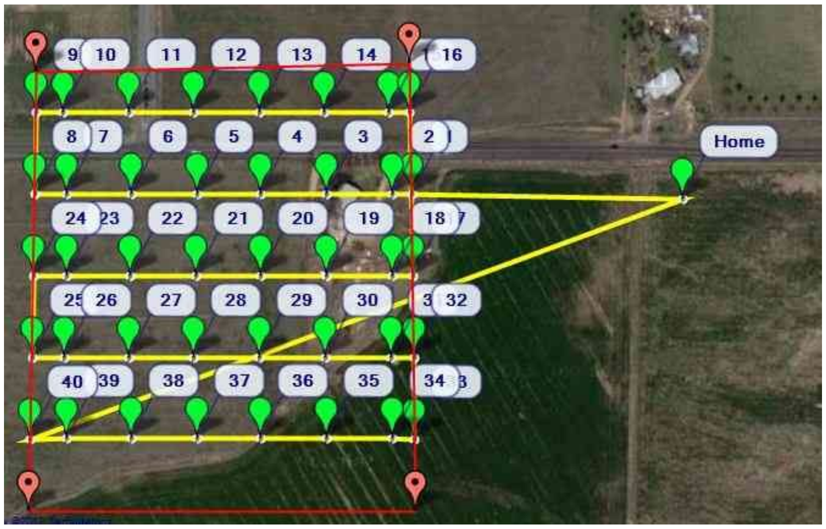


Figure 5.5: A auto-generated grid by GCS.

5.2.5 Rally Points

Ordinarily when the copter enters Return to Launch (RTL) mode (typically triggered by an autopilot failsafe), the default behaviour is to return to the Home point, but there are often cases when that can be undesirable. For example it may be an area full of people or property and a system running in RTL mode may very likely be in a state that merits extreme caution! It is also possible that the flight plan is large enough that should the aircraft enter RTL mode it is undesirable to traverse all the way back to the point of takeoff.

For this reason this GCS supports the creation of multiple Rally Points. Should an aircraft enter RTL and Rally Points have been defined then it will proceed to the closest Rally Point, rather than proceeding to the Home position. Plane will then loiter at that location, and Copter will perform an automated landing there.

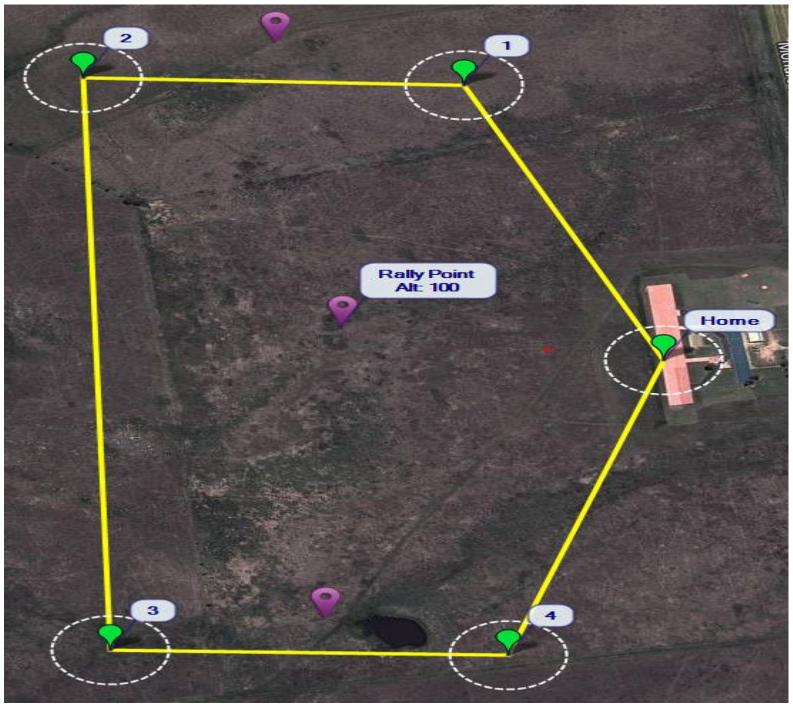


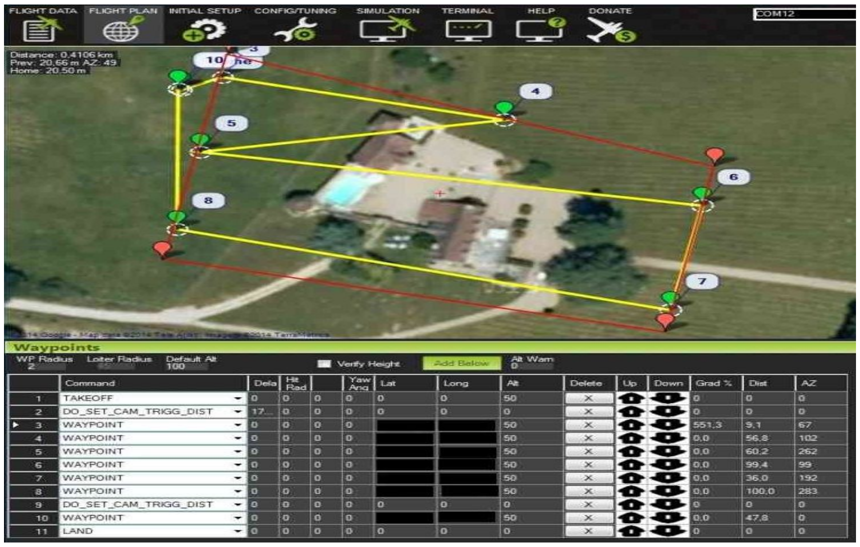
Figure 5.6: A flight plan with Rally Points. Rally Points are denoted with Purple markers. Mousing over a Rally Point gives its loiter altitude(as in the middle point above).

5.3 Camera Control and Auto Missions

Planning a camera mission is almost exactly the same as planning any other mission with waypoints and events. The only difference is that in a camera mission commands must be specified to trigger the camera shutter at waypoints or at regular intervals as the vehicle moves. As the camera is mounted on a gimbal, the gimbal orientation has to be set, or it can be programmed to track a particular point of interest.

For simple missions the required waypoints and camera commands can be manually programmed. For more complex paths and grid surveys this custom built GCS makes things easy by providing tools to automatically generate the required mission for arbitrary regions.

The test mission shown below was run. It generated 15 images, which have been stitched together. A thumbnail of the composite image is shown below (because the full size image is too large at about 107 MBytes).



WP Radius 2 | Loiter Radius | Default Alt 100 | Alt Warn 0

	Command	Dela	Hit Rad		Yaw Ang	Lat	Long	Alt	Delete	Up	Down	Grad %	Dist	AZ
1	TAKEOFF	0	0	0	0	0	0	50	X	⬆	⬇	0	0	0
2	DO_SET_CAM_TRIGG_DIST	17...	0	0	0	0	0	0	X	⬆	⬇	0	0	0
▶ 3	WAYPOINT	0	0	0	0	[redacted]	[redacted]	50	X	⬆	⬇	551,3	9,1	67
4	WAYPOINT	0	0	0	0	[redacted]	[redacted]	50	X	⬆	⬇	0,0	56,8	102
5	WAYPOINT	0	0	0	0	[redacted]	[redacted]	50	X	⬆	⬇	0,0	60,2	262
6	WAYPOINT	0	0	0	0	[redacted]	[redacted]	50	X	⬆	⬇	0,0	99,4	99
7	WAYPOINT	0	0	0	0	[redacted]	[redacted]	50	X	⬆	⬇	0,0	36,0	192
8	WAYPOINT	0	0	0	0	[redacted]	[redacted]	50	X	⬆	⬇	0,0	100,0	283
9	DO_SET_CAM_TRIGG_DIST	0	0	0	0	0	0	0	X	⬆	⬇	0	0	0
10	WAYPOINT	0	0	0	0	[redacted]	[redacted]	50	X	⬆	⬇	0,0	47,8	0
11	LAND	0	0	0	0	0	0	0	X	⬆	⬇	0	0	0

Figure 5.7:The final mission with waypoints and camera triggers looks like this.

Figure 5.8: A thumbnail of the composite image.

5.4 Calibration

5.4.1 Compass and Accelerometer:

- **Onboard:** The calibration routine runs on the flight controller. The vehicle is held in the air and rotated so that each side (front, back, left, right, top and bottom) points down towards the earth for a few seconds in turn.

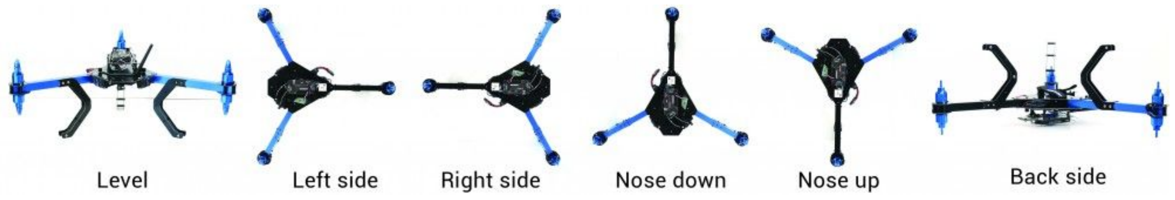
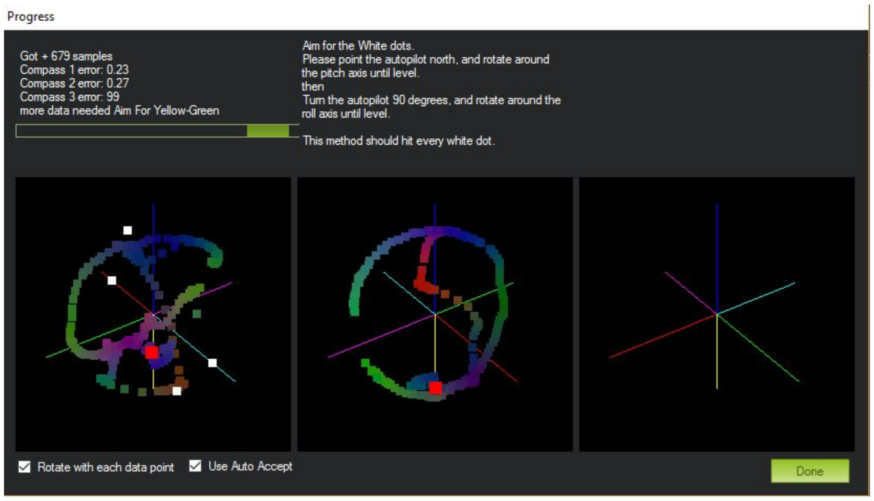


Figure 5.9: Calibration Positions

- **Offboard (For compass only):** The controller relies on the ground station to calculate the compass offsets.


Progress

Got + 679 samples
Compass 1 error: 0.23
Compass 2 error: 0.27
Compass 3 error: 99
more data needed Aim For Yellow-Green

Aim for the White dots.
Please point the autopilot north, and rotate around the pitch axis until level.
then
Turn the autopilot 90 degrees, and rotate around the roll axis until level.

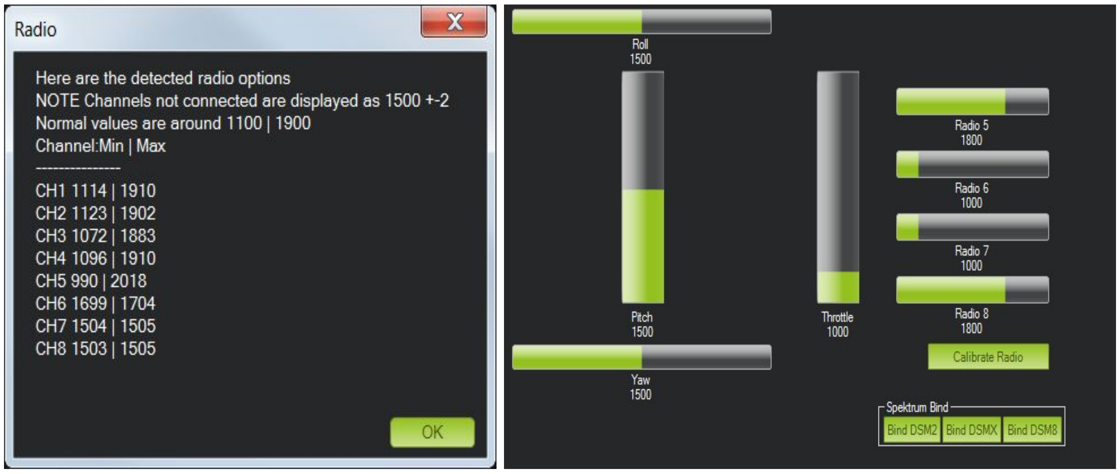
This method should hit every white dot.

Figure 5.10: Live Compass Calibration

5.4.2 Radio Control Calibration

RC transmitters are used to control vehicle movement and orientation. This Copter controls throttle, pitch, roll and yaw. Each of these control signals are mapped to transmitter stick/switch(s) and in turn to autopilot channels from the connected receiver.

To calibrating each of the transmitter controls/channels, each of the enabled sticks/switches is moved through their full range and the maximum and minimum positions are recorded.


Radio

Here are the detected radio options
NOTE Channels not connected are displayed as 1500 +-2
Normal values are around 1100 | 1900
Channel:Min | Max

CH1 1114 | 1910
CH2 1123 | 1902
CH3 1072 | 1883
CH4 1096 | 1910
CH5 990 | 2018
CH6 1699 | 1704
CH7 1504 | 1505
CH8 1503 | 1505

Figure 5.11: Left: Radio Calibration Results; Right: Radio Calibration Screen

Chapter 6

MAVlink Protocol

MAVLink or Micro Air Vehicle Link is a protocol for communication between GCS and the drone or between the on-board computer and the flight controller. It is designed as a very lightweight, header-only message marshaling library. MAVLink follows a modern hybrid publish-subscribe and point-to-point design pattern: Data streams are sent / published as **topics** while configuration sub-protocols such as the mission protocol or parameter protocol are point-to-point with retransmission.

6.1 Message

Every message is identifiable by the ID field on the packet, and the payload contains the data from the message.

Messages are defined within XML files, which is then generated into appropriate source code for each of the supported languages, python or C is this case. Each XML file defines the message set supported by a particular system, also referred to as a "dialect". The reference message set that is implemented by this particular ground control station and autopilot is defined in common.xml.

6.2 Packet Structure

The packet structure is the following:

Field name	**Index (Bytes)**	**Purpose**
Start-of-frame	0	Denotes the start of frame transmission (v1.0: 0xFE)
Payload-length	1	length of payload (n)

Packet sequence	2	Each component counts up their send sequence. Allows for detection of packet loss.
System ID	3	Identification of the SENDING system. Allows to differentiate different systems on the same network.
Component ID	4	Identification of the SENDING component. Allows to differentiate different components of the same system, e.g. the IMU and the autopilot.
Message ID	5	Identification of the message - the id defines what the payload “means” and how it should be correctly decoded.
Payload	6 to (n+6)	The data into the message, depends on the message id.
CRC	(n+7) to (n+8)	Check-sum of the entire packet, excluding the packet start sign (LSB to MSB)

6.3 CRC field

To ensure message integrity a cyclic redundancy check (CRC) is calculated to every message into the last two bytes. Another function of the CRC field is to ensure the sender and receiver both agree in the message that is being transferred. It is computed using an ITU X.25/SAE AS-4 hash of the bytes in the packet, excluding the Start-of-Frame indicator (so 6+n+1 bytes are evaluated, the extra +1 is the seed value).

Additionally a seed value is appended to the end of the data when computing the CRC. The seed is generated with every new message set of the protocol, and it is hashed in a similar way as the packets from each message specifications. The system used in this work uses a precomputed array for this purpose.

The CRC algorithm of MAVLink in this work has been implemented in Python.

Chapter 7

Pixhawk Flight Controller

The Pixhawk PX4 is a high-end industry standard autopilot hardware.The Pixhawk autopilot module runs a very efficient real-time operating system (RTOS), which provides a POSIX-style environment (i.e. printf(), pthreads, /dev/ttyS1, open(), write(), poll(), ioctl(), etc).

Figure 7.1: Pixhawk PX4

7.1 Key Features

- 168 MHz / 252 MIPS Cortex-M4F
- 14 PWM / Servo outputs (8 with failsafe and manual override, 6 auxiliary, high-power compatible)
- Abundant connectivity options for additional peripherals (UART, I2C, CAN)
- Integrated backup system for in-flight recovery and manual override with dedicated processor and stand-alone power supply (fixed-wing use)
- Backup system integrates mixing, providing consistent autopilot and manual override mixing modes (fixed wing use)
- Redundant power supply inputs and automatic failover
- External safety switch
- Multicolor LED main visual indicator

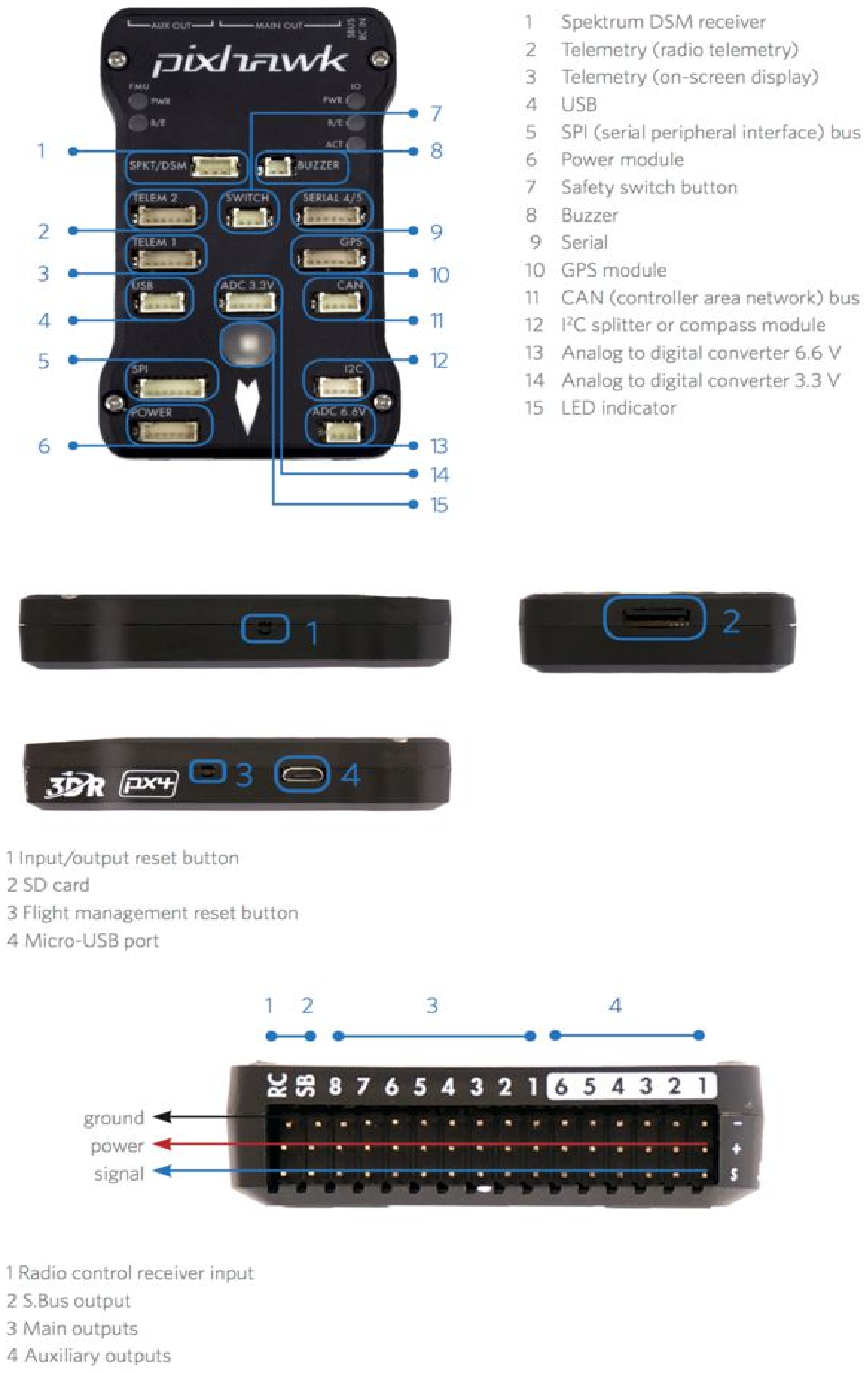

1 Spektrum DSM receiver
2 Telemetry (radio telemetry)
3 Telemetry (on-screen display)
4 USB
5 SPI (serial peripheral interface) bus
6 Power module
7 Safety switch button
8 Buzzer
9 Serial
10 GPS module
11 CAN (controller area network) bus
12 I²C splitter or compass module
13 Analog to digital converter 6.6 V
14 Analog to digital converter 3.3 V
15 LED indicator
pixhawk
SPKT/DSM
BUZZER
TELEM 2
SWITCH
SERIAL 4/5
TELEM 1
GPS
USB
ADC 3.3V
CAN
SPI
I2C
POWER
ADC 6.6V
1 Input/output reset button
2 SD card
3 Flight management reset button
4 Micro-USB port
ground
power
signal
RC
SB
1 Radio control receiver input
2 S.Bus output
3 Main outputs
4 Auxiliary outputs

Figure 7.2: Pixhawk Connectors

7.2 Specifications

7.2.1 Processor

- 32bit STM32F427 Cortex M4 core with FPU
- 168 MHz
- 256 KB RAM
- 2 MB Flash
- 32 bit STM32F103 failsafe co-processor

7.2.2 Inbuilt Sensors

- ST Micro L3GD20H 16 bit gyroscope
- ST Micro LSM303D 14 bit accelerometer / magnetometer
- Invensense MPU 6000 3-axis accelerometer/gyroscope
- MEAS MS5611 barometer

7.2.3 Inbuilt Interfaces

- 5x UART (serial ports), one high-power capable, 2x with HW flow control
- 2x CAN (one with internal 3.3V transceiver, one on expansion connector)
- Spektrum DSM / DSM2 / DSM-X® Satellite compatible input
- Futaba S.BUS® compatible input and output
- PPM sum signal input
- RSSI (PWM or voltage) input
- I2C
- SPI
- 3.3 and 6.6V ADC inputs
- Internal microUSB port and external microUSB port extension

7.2.4 Power System and Protection

- Ideal diode controller with automatic failover
- Servo rail high-power (max. 10V) and high-current (10A+) ready

Chapter 8

Autopilot Firmware Design

This work primarily revolves around developing a custom built autopilot which is on its path to become one of the most advanced, full-featured and reliable autopilots on successful testing. Coupled with ground control software, this unmanned vehicle running the autopilot firmware, has advanced functionality including real-time communication with the operator. In this section, autopilot controlled flying is discussed with RC inputs, and intelligent - autonomous flying is covered in the section 10.

8.1 Attributes:

- The autopilot has Autonomous flight modes that execute fully scripted missions with advanced features.
- This autopilot also provides Advanced failsafe options which comes in handy in the event of lost control signal, low battery conditions, or other system failures.
- This work is aided by this autopilot for three Axis camera control and stabilization, shutter control and live video link with programmable on-screen-display.
- This work uses a Real-time two-way communication between the GCS and controller, including GPS position, battery status, and other live information, which are facilitated by the autopilot.
- Full data logging for comprehensive post mission analysis, with graphing and Google Earth mapping tools is possible as well.

8.2 Key Abilities of the autopilot:

- High precision acrobatic mode: performs aggressive maneuvers including flips.
- Auto-level and Altitude Hold modes: Fly level and straight with ease or add simple mode which removes the need for the pilot to keep track of the vehicle's heading. The stick is pushed the way the vehicle needs to go, and the autopilot figures out what that means for whatever orientation the copter is in.
- Loiter and PosHold modes: the vehicle will hold its position using its GPS, accelerometers and barometer.
- Return to launch: Copter can fly back to the launch location and land automatically.
- Ad-hoc commands in Flight : With a two-way telemetry radio installed, clicking on the map will make the vehicle will fly to the desired location.
- Autonomous missions: The ground station is used to define complex missions with up to hundreds of GPS wapoints. Once the vehicle switches to "AUTO", it can be seen take-off, execute the mission, then return home, land and disarm, all without any human intervention.
- Failsafes: The software monitors the state of the system and triggers an autonomous return-to-home in case of loss of contact with the pilot, low battery or the vehicle strays outside a defined geofence.

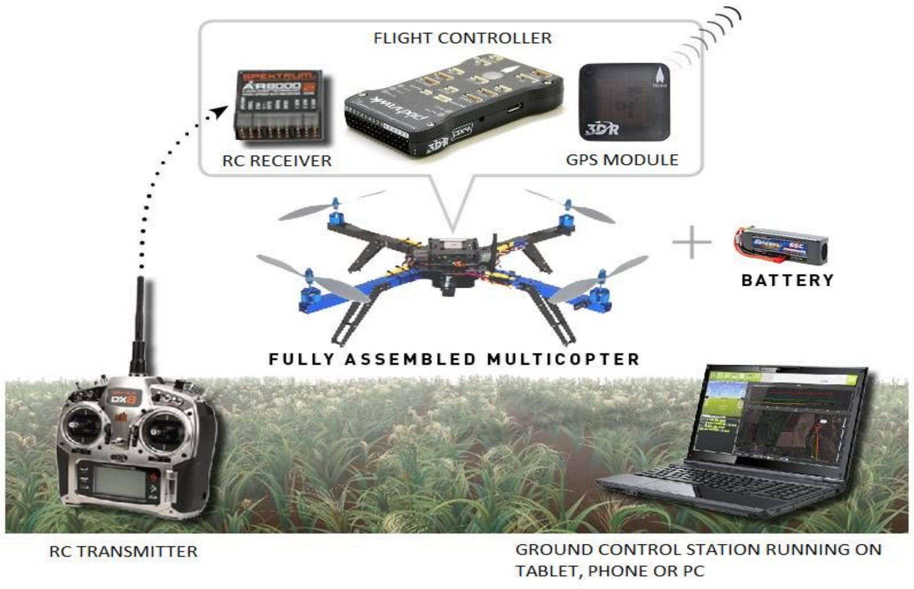

FLIGHT CONTROLLER
RC RECEIVER
GPS MODULE
BATTERY
FULLY ASSEMBLED MULTICOPTER
RC TRANSMITTER
GROUND CONTROL STATION RUNNING ON TABLET, PHONE OR PC

Figure 8.1: Autopilot run system, control intelligence.

8.3 Flight Modes

Copter has 20 flight modes developed, 8 of which were developed in this work .

Flight modes are controlled through the radio, via mission commands, or using commands from a ground station (GCS) or companion computer.

In this section, Radio control is highlighted, but the next section covers the companion computer controlled flying of the drone.

8.3.1 Primary Flight Modes

8.3.1.1 Stabilize Mode

Stabilize mode allows the pilot to fly the vehicle manually, but self-levels the roll and pitch axis.

- Pilot's roll and pitch input control the lean angle of the copter. When the pilot releases the roll and pitch sticks the vehicle automatically levels itself.
- Pilot will need to regularly input roll and pitch commands to keep the vehicle in place as it is pushed around by the wind.
- Pilot's yaw input controls the rate of change of the heading. When the pilot releases the yaw stick the vehicle will maintain it's current heading.
- Pilot's throttle input controls the average motor speed meaning that constant adjustment of the throttle is required to maintain altitude. If the pilot puts the throttle completely down the motors will go to their minimum rate and if the vehicle is flying it will lose attitude control and tumble.
- The throttle sent to the motors is automatically adjusted based on the tilt angle of the vehicle (i.e. increased as the vehicle tilts over more) to reduce the compensation the pilot must do as the vehicle's attitude changes.

8.3.1.2 Altitude Hold

In altitude hold mode, Copter maintains a consistent altitude while allowing roll, pitch, and yaw is controlled normally, ie, the throttle is automatically controlled to maintain the current

altitude. Roll, Pitch and yaw operate the same as in Stabilize mode meaning that the pilot directly controls the roll and pitch lean angles and the heading.
If the throttle stick is in the middle (40% ~ 60%) the vehicle will maintain the current altitude. Outside of the mid-throttle deadzone (i.e. below 40% or above 60%) the vehicle will descend or climb depending upon the deflection of the stick. When the stick is completely down the copter will descend at 2.5m/s and if at the very top it will climb by 2.5m/s. These speeds can be adjusted by altering the value of a designated variable in the source code.

8.3.1.3 Loiter Mode

Loiter Mode automatically attempts to maintain the current location, heading and altitude. The pilot may fly the copter in Loiter mode as if it were in a more manual flight mode but when the sticks are released, the vehicle will slow to a stop and hold position.
Horizontal location can be adjusted with the the Roll and Pitch control sticks with the default maximum horizontal speed being 5m/s, however, this can be adjusted while tuning. When the pilot releases the sticks the copter will slow to a stop.
Altitude can be controlled with the Throttle control stick just as in AltHold mode
The heading can be set with the Yaw control stick.

8.3.1.4 RTL Mode

RTL mode (Return To Launch mode) navigates Copter from its current position to hover above the home position. RTL is a GPS-dependent move. In RTL mode the flight controller uses a barometer which measures air pressure as the primary means for determining altitude ("Pressure Altitude").

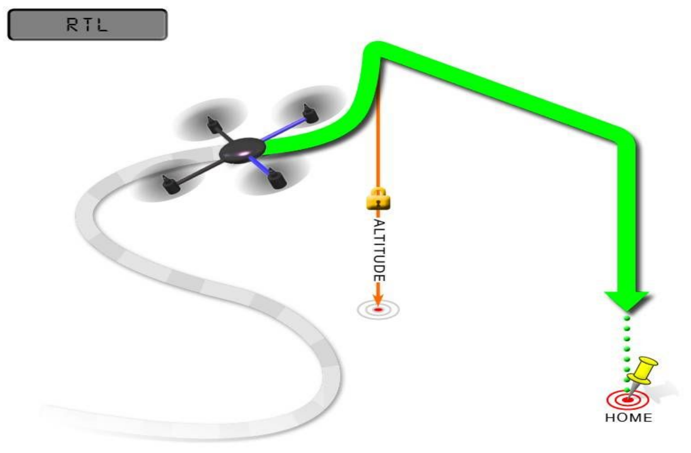


Figure 8.2: RTL Mode

8.3.1.5 AUTO Mode

In Auto mode the copter will follow a pre-programmed mission script stored in the autopilot which is made up of navigation commands (i.e. waypoints) and "do" commands (i.e. commands that do not affect the location of the copter including triggering a camera shutter).

AUTO mode incorporates the altitude control from AltHold mode and position control from Loiter mode. All the same apply including ensuring that vibration levels and compass interference levels are acceptable and that the GPS is functioning well including returning an HDOP of under 2.0. Copter has to be armed before engaging AUTO mode.

If starting the mission while the copter is on the ground the pilot should ensure the throttle is down, then switch to the Auto flight mode, then raise the throttle. The moment that the throttle is raised above zero, the copter will begin the mission.

If starting the mission from the air the mission will begin from the first command the moment that the flight mode switch is moved to Auto. If the first command in the mission is a take-off command but the vehicle is already above the take-off command's altitude the take-off command will be considered completed and the vehicle will move onto the next waypoint.

At any time the pilot can retake control from the autopilot by returning the flight mode switch to another flight mode such as Stabilize or Loiter. If the pilot then switches to AUTO again, the mission will restart from the first command.

During the mission the pilot’s roll, pitch and throttle inputs are ignored but the yaw can be overridden with the yaw stick. This allows the pilot to for example aim the nose of the copter (which might have a hard mounted camera on it) as the copter flies the mission. The autopilot will attempt to retake yaw control as the vehicle passes the next waypoint.

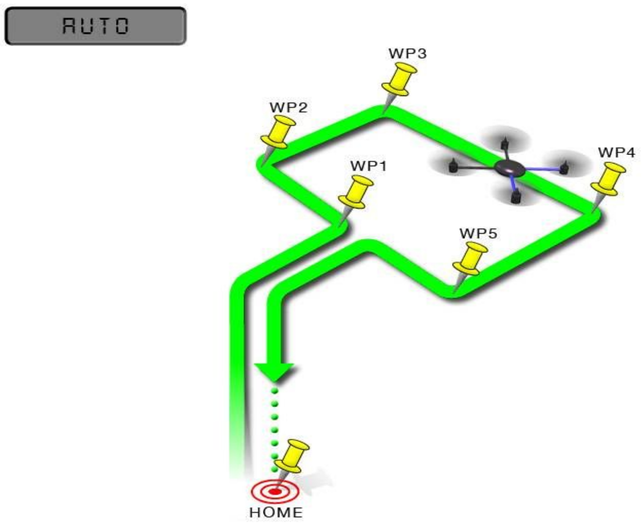


Figure 8.3: AUTO Mode

8.3.2 Secondary Modes

8.3.2.1 Guided Mode

Guided mode is a capability to dynamically guide the copter to a target location wirelessly using a telemetry radio module and GCS. This capability allows the pilot to interactively command the copter to travel to a target location by specifying a Geolocation Coordinate on the Flight Data map. Once the location is reached, the copter will hover at that location, waiting for the next target.

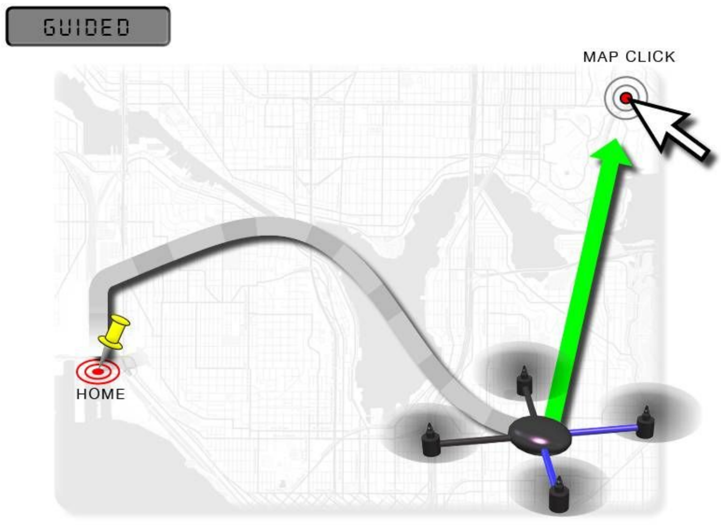


Figure 8.4: GUIDED Mode

The Copter has to be set up at the field and a MAVLink connection needs to be established over wireless telemetry between the copter and the laptop / companion computer.
The code pipeline will make the copter take off in Stabilize Mode and once at a reasonable altitude, will switch to Loiter Mode.
Specify the guided mode altitude(in meters) and the geolocation coordinates.
The vehicle will fly to the target location and wait there until another location is specified by the code or will switch to another mode.

8.3.2.2 Position Hold Mode

It is similar to Loiter in that the vehicle maintains a constant location, heading, and altitude. The copter will arm in PosHold mode but only once the GPS has 3D lock and the HDOP has dropped to 2.0 or lower. Horizontal location can be adjusted with the the Roll and Pitch control sticks with the default maximum lean angle being 45 degrees (default; can be altered in source).
When the pilot releases the sticks the copter will lean back to bring the vehicle to a stop. Altitude can be controlled with the Throttle control stick. The heading can be set with the Yaw control stick.

8.3.2.3 Land Mode

LAND Mode attempts to bring the copter straight down and descends to 10m (or until the sonar senses something below the copter) using the regular Altitude Hold controller which will descend at the speed specified in a parameter variable in the source code. Below 10m the copter will descend at the rate specified in a particular parameter variable in the source code, which defaults to 50 cm/s. Upon reaching the ground the copter will automatically shut-down the motors and disarm the copter if the pilot's throttle is at minimum. Copter will recognise that it has landed if the motors are at minimum but it's climb rate remains between -20cm/s and +20cm/s for one second. It does not use the altitude to decide whether to shut off the motors except that the copter must also be below 10m above the home altitude.
If the vehicle has GPS lock the landing controller will attempt to control it's horizontal position but the pilot can adjust the target horizontal position just as in Loiter mode.
If the vehicle does not have GPS lock the horizontal control will be as in stabilize mode so the pilot can control the roll and pitch lean angle of the copter.

8.4 Pre-Arm Safety Check

Pre-arm Safety Checks which will prevent the vehicle from arming if any of a fairly large number of issues are discovered before take-off including missed calibration, configuration or bad sensor data. These checks help prevent crashes and fly-aways.

8.4.1 Arming / Disarming the motors

Arming the motors causes the autopilot to apply power to the motors, which will cause them to start spinning. Disarming the motors will cause the motors to stop spinning.

8.5 Vibration

In this project, the Flight controller accelerometer is sensitive to vibration. As accelerometer data is used to estimate vehicle position (along with barometer and GPS readings), excessive vibration can lead to poor performance in modes that rely on accurate position calculation.

Vibration affects all vehicle types, but is most critical for the Copter (particularly in AltHold, Loiter, RTL, Guided, Position and AUTO flight modes). The algorithm for calculating the vibration levels involves calculating the standard deviation.

8.6 Extended Kalman Filter (EKF)

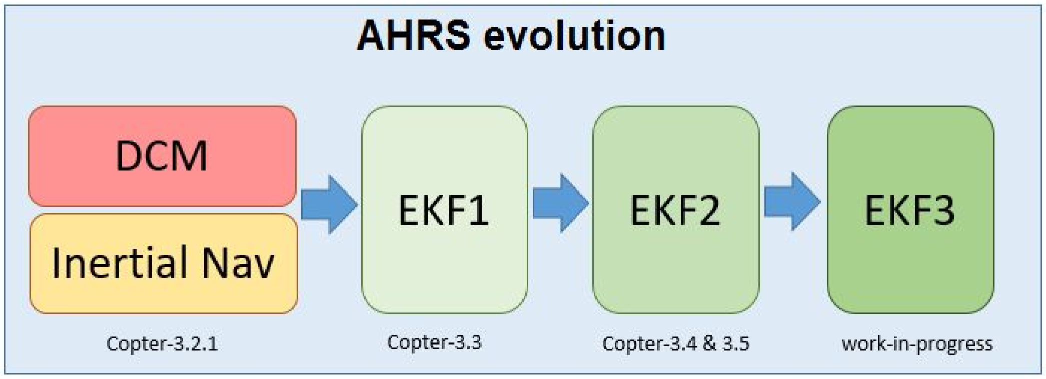


Figure 8.5: EKF Evolution

This work uses an Extended Kalman Filter (EKF) algorithm to estimate vehicle position, velocity and angular orientation based on rate gyroscopes, accelerometer, compass, GPS, airspeed and barometric pressure measurements.

The advantage of the EKF over the simpler complementary filter algorithms (i.e. “Inertial Nav”), is that by fusing all available measurements it is better able to reject measurements with significant errors. This makes the vehicle less susceptible to faults that affect a single sensor. EKF also enables measurements from optional sensors such as optical flow and laser range finders to be used to assist navigation.

Current stable version of the autopilot uses the EKF2 as it’s primary attitude and position estimation source with DCM running quietly in the background. The flight controller has two IMUs available, thus two EKF “cores” (i.e. two instances of the EKF) runs in parallel, each using a different IMU. At any one time, only the output from a single EKF core is ever used, that core being the one that reports the best health which is determined by the consistency of its sensor data.

8.7 Ground Effect Compensation

Ground effect compensation reduces the weighting of the barometer (in favour of the accelerometers) when the vehicle is likely taking off or landing. This reduces the bounce sometimes seen when landing vehicles with short legs compared to the length of their propellers.

8.8 Motor Thrust Scaling

Motor thrust scaling compensates for the non-linear thrust output of most ESCs and motors.

8.9 Sensor Position Offset Compensation

The autopilot has the capability for the compensation for sensor placement on the vehicle.

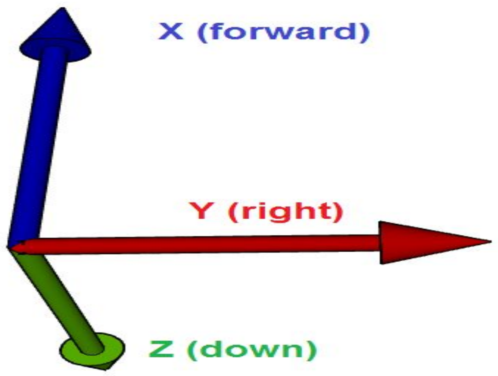


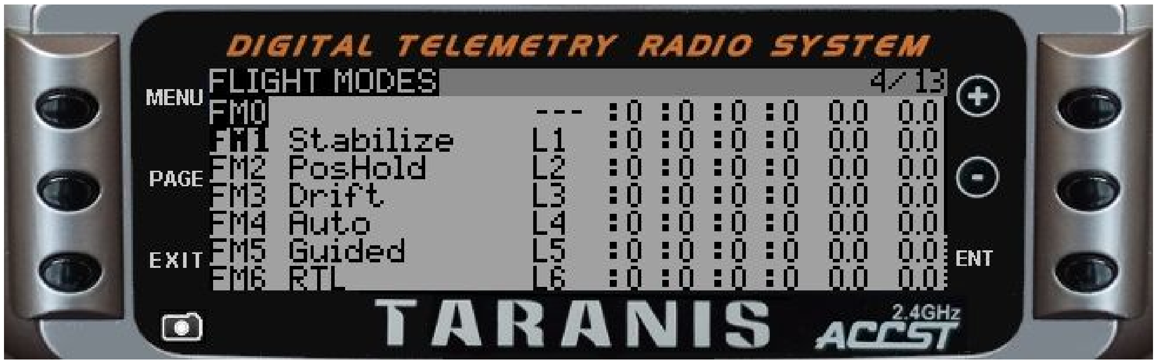
Figure 8.6: sensor's position offsets

The sensor's position offsets are specified as 3 values (X, Y and Z) which are distances in meters from the IMU (which can be assumed to be in the middle of the flight controller board) or the vehicle's center of gravity.

- X : distance forward of the IMU or center of gravity. Positive values are towards the front of the vehicle, negative values are towards the back.

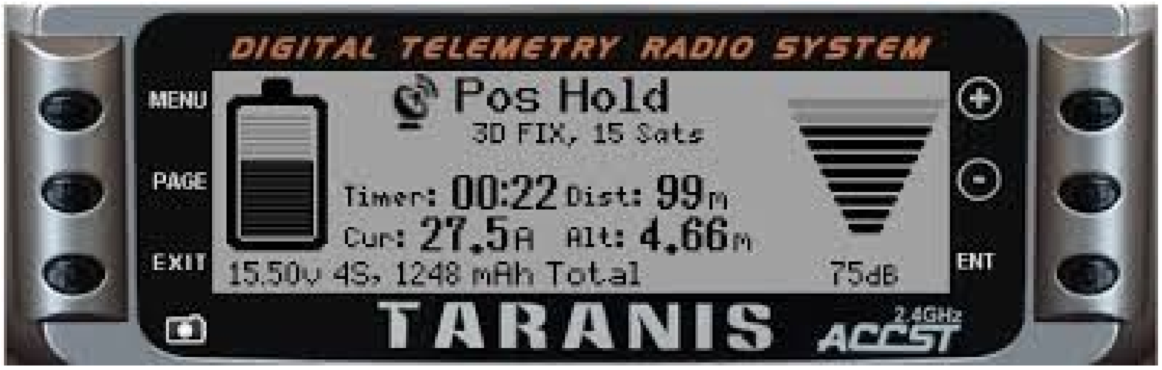
- Y : distance to the right of the IMU or center of gravity. Positive values are towards the right side of the vehicle, negative values are towards the left.
- Z : distance *below* the IMU or center of gravity. Positive values are *lower*, negative values are *higher*.

The compensation is only *partial* because the autopilot can correct the vehicle's velocity and position estimate but it does not correct the acceleration estimate. For example if the flight controller was placed on the nose of a vehicle and the vehicle suddenly leans back (i.e. rotates so that it's nose points up) with no offset compensation the vehicle velocity estimate would momentarily show the vehicle is climbing when it's not. With the position offsets added the velocity would not show this momentary climb. The EKF would still show a momentary vertical acceleration and because the acceleration is used in our altitude hold controllers this could still lead to the vehicle momentary reducing throttle.



```
FLIGHT MODES                                   4/13
FM0                 ---   :0 :0 :0 :0  0.0  0.0
FM1 Stabilize       L1    :0 :0 :0 :0  0.0  0.0
FM2 PosHold         L2    :0 :0 :0 :0  0.0  0.0
FM3 Drift           L3    :0 :0 :0 :0  0.0  0.0
FM4 Auto            L4    :0 :0 :0 :0  0.0  0.0
FM5 Guided          L5    :0 :0 :0 :0  0.0  0.0
FM6 RTL             L6    :0 :0 :0 :0  0.0  0.0
```

Figure 8.7: TARANIS DX9 2 Switches 1 TX channel 6 Flight Modes



```
Pos Hold
3D FIX, 15 Sats
Timer: 00:22 Dist: 99m
Cur: 27.5A  Alt: 4.66m
15.50v 4S, 1248 mAh Total        75dB
```

Figure 8.8: Pos Hold activated on the TARANIS

Chapter 9

Auxiliary Hardwares

9.1 OSD Board (On Screen Display)

A custom developed OSD board performs an overlay of telemetry data from the autopilot over a First Person View monitor.

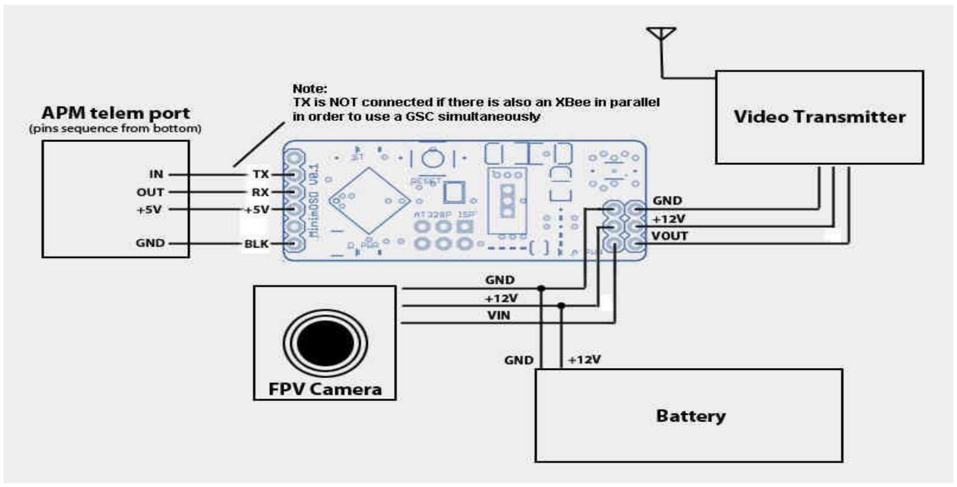

APM telem port
(pins sequence from bottom)
IN
OUT
+5V
GND
Note:
TX is NOT connected if there is also an XBee in parallel in order to use a GSC simultaneously
TX
RX
+5V
BLK
Video Transmitter
GND
+12V
VOUT
GND
+12V
VIN
FPV Camera
GND
+12V
Battery

Figure 9.1: OSD Board interfacing

9.2 Optical Flow Sensor

The (Optical Flow) Sensor is a specialized high resolution downward pointing camera module and a 3-axis gyro that uses the ground texture and visible features to determine aircraft ground velocity. It is used in place of a GPS, but is not within this work's scope.

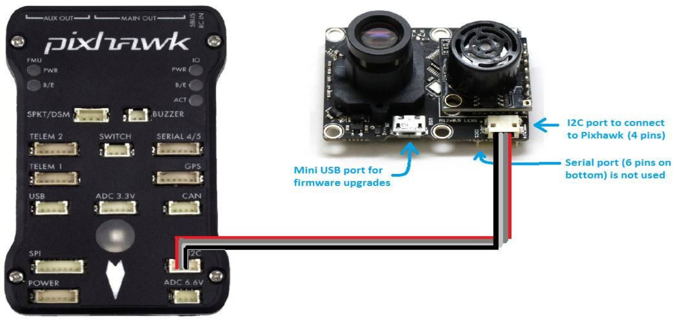

pixhawk
I2C port to connect to Pixhawk (4 pins)
Serial port (6 pins on bottom) is not used
Mini USB port for firmware upgrades

Figure 9.2: OFS Interfacing with the Flight Controller

9.3 Object Avoidance Sensor

It is a simultaneous multi-axis scanner for SLAM and collision avoidance.

It is totally silent and has zero moving parts, resulting in greater efficiency and reliability. It combines 8 sensors and a hub to create a lightweight, fully eye-safe scanner for SLAM (Simultaneous Localization and Mapping) and collision avoidance on the fast-moving airborne robots. Calibrated distance data is streamed from a USB or UART port as an array of synchronized distances, in millimetres.

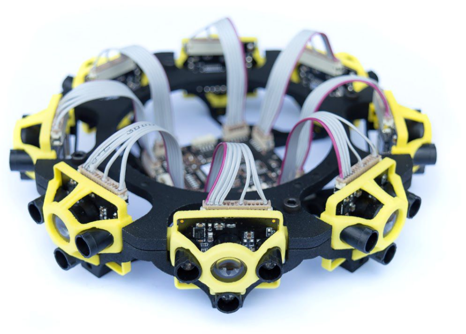


Figure 9.3: Multi-axis stationary collision avoidance sensor

9.4 Crop Sprayer

This allows the flight controller connected to a PWM operated pump and (optionally) spinner, to control the rate of flow of liquid fertilizer based on the vehicle speed. The drone that was being developed, was intended for the agriculture industry and this feature is unique to this market specific design of the drone.

9.5 Cameras and Gimbals

This drone is being developed further to use 3-axis gimbals, including advance features like automated aiming of the camera at a Region of Interest (ROI), and automatic triggering of a camera shutter. Vision is the next stage in the development cycle of the drone, of which this work forms a part.

Chapter 10

Advanced Autonomous Guidance Assisted Flying System (AAGAFS)

A custom API co-developed by the author of this report as a part of an open source and community-driven project, allows programs to run on an onboard companion computer, Raspberry Pi in this case, and communicate with the flight controller using a low-latency link. Onboard applications significantly enhances the autopilot, adding greater intelligence to vehicle behaviour, and performing tasks that are computationally intensive or time-sensitive (for example, computer vision, path planning, or 3D modelling).

The API communicates with vehicles over MAVLink. It provides programmatic access to a connected vehicle's telemetry, state and parameter information, and enables both mission management and direct control over vehicle movement and operations.

The API provides classes and methods to:

- Connect to a vehicle (or multiple vehicles) from a script
- Get and set vehicle state/telemetry and parameter information.
- Receive asynchronous notification of state changes.
- Guide a UAV to specified position (GUIDED mode).
- Send arbitrary custom messages to control UAV movement and other hardware (GUIDED mode).
- Create and manage waypoint missions (AUTO mode).
- Override RC channel settings.

10.1 Communicating with Raspberry Pi via MAVLink

This sub-section briefly explains how to connect and configure a Raspberry Pi (RPi) so that it is able to communicate with a Pixhawk flight controller using the MAVLink protocol over a serial connection. This can be used to perform additional tasks such as image recognition which simply cannot be done by the Pixhawk due to the memory requirements for storing images.

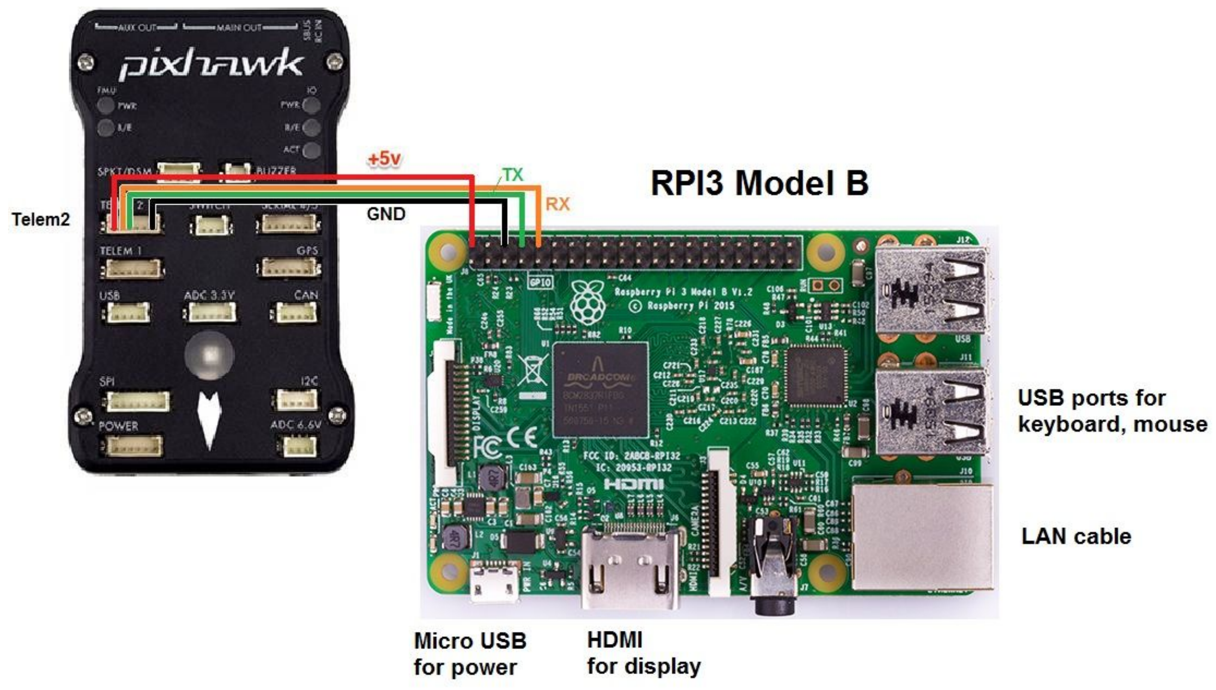


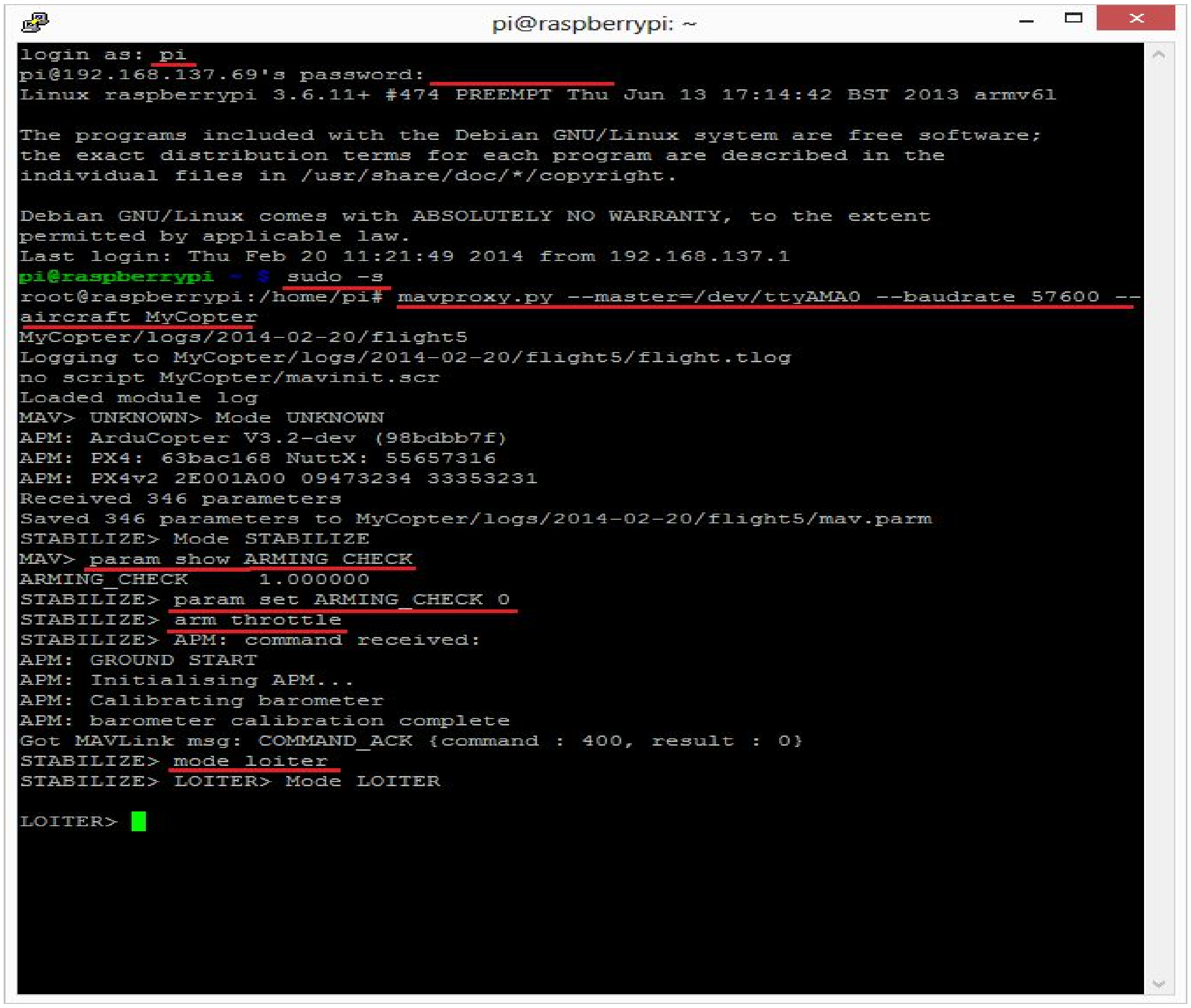
Figure 10.1: Pixhawk-RPi hardware connection

The Pixhawk's TELEM2 port is connected to the RPi's Ground, TX and RX pins as shown in the image above.
The RPi can be powered by connecting the red V+ cable to the +5V pin (as shown above) **or** from USB in (for example, using a separate 5V BEC hooked up to the USB power).
Powering via USB is recommended as it is typically safer - because the input is regulated.

The Pixhawk is connected with a the ground station and the following parameters are set:

- SERIAL2_PROTOCOL = 1 to enable MAVLink on the serial port (TELEM 2).
- SERIAL2_BAUD = 921 (BAUD rate of TELEM 2 port) so the Pixhawk can communicate with the RPi at 921600 baud.
- LOG_BACKEND_TYPE = 3 to stream the dataflash log files to the RPi

After connecting to RPi with an SSH client, installing the required packages on the Raspberry Pi, and disabling the OS control of the serial port, the following is a snippet of what is obtained while testing the connection:



```
login as: pi
pi@192.168.137.69's password:
Linux raspberrypi 3.6.11+ #474 PREEMPT Thu Jun 13 17:14:42 BST 2013 armv6l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Feb 20 11:21:49 2014 from 192.168.137.1
pi@raspberrypi ~ $ sudo -s
root@raspberrypi:/home/pi# mavproxy.py --master=/dev/ttyAMA0 --baudrate 57600 --
aircraft MyCopter
MyCopter/logs/2014-02-20/flight5
Logging to MyCopter/logs/2014-02-20/flight5/flight.tlog
no script MyCopter/mavinit.scr
Loaded module log
MAV> UNKNOWN> Mode UNKNOWN
APM: ArduCopter V3.2-dev (98bdbb7f)
APM: PX4: 63bac168 NuttX: 55657316
APM: PX4v2 2E001A00 09473234 33353231
Received 346 parameters
Saved 346 parameters to MyCopter/logs/2014-02-20/flight5/mav.parm
STABILIZE> Mode STABILIZE
MAV> param show ARMING_CHECK
ARMING_CHECK     1.000000
STABILIZE> param set ARMING_CHECK 0
STABILIZE> arm throttle
STABILIZE> APM: command received:
APM: GROUND START
APM: Initialising APM...
APM: Calibrating barometer
APM: barometer calibration complete
Got MAVLink msg: COMMAND_ACK {command : 400, result : 0}
STABILIZE> mode loiter
STABILIZE> LOITER> Mode LOITER

LOITER>
```

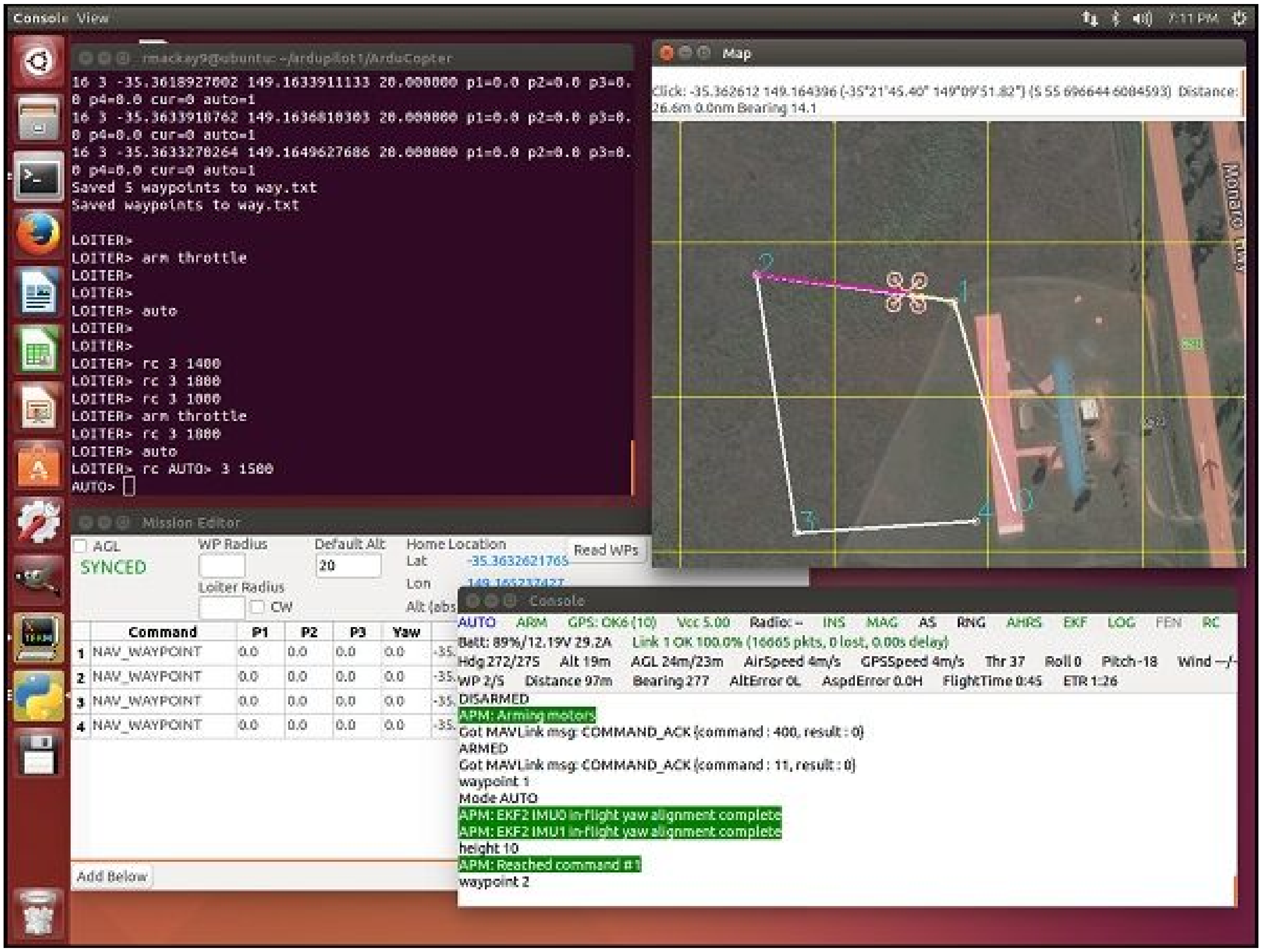
Figure 10.2: Connection Testing Result

10.2 Simulated Vehicle (SITL)

The SITL (Software In The Loop) simulator allowed the author to create and test apps without a real vehicle. It is a build of the autopilot code using an ordinary C++ compiler, giving the author a native executable that allowed the author to test the behaviour of the code without hardware.

When running in SITL the sensor data comes from a flight dynamics model in a flight simulator.

A big advantage of SITL is that it gave the author access to the full range of development tools available to desktop C++ development, such as interactive debuggers, static analyzers and dynamic analysis tools.



```
16 3 -35.3618927002 149.1633911133 20.000000 p1=0.0 p2=0.0 p3=0.
0 p4=0.0 cur=0 auto=1
16 3 -35.3633918762 149.1636810303 20.000000 p1=0.0 p2=0.0 p3=0.
0 p4=0.0 cur=0 auto=1
16 3 -35.3633270264 149.1649627686 20.000000 p1=0.0 p2=0.0 p3=0.
0 p4=0.0 cur=0 auto=1
Saved 5 waypoints to way.txt
Saved waypoints to way.txt

LOITER>
LOITER> arm throttle
LOITER>
LOITER>
LOITER> auto
LOITER>
LOITER>
LOITER> rc 3 1400
LOITER> rc 3 1800
LOITER> rc 3 1000
LOITER> arm throttle
LOITER> rc 3 1000
LOITER> auto
LOITER> rc AUTO> 3 1500
AUTO>
```

Click: -35.362612 149.164396 (-35°21'45.40" 149°09'51.82") (S 55 696644 6084593) Distance: 26.6m 0.0nm Bearing 14.1

AGL SYNCED WP Radius Loiter Radius CW Default Alt 20 Home Location Lat -35.3632621765 Lon 149.165237427 Alt (abs

	Command	P1	P2	P3	Yaw	
1	NAV_WAYPOINT	0.0	0.0	0.0	0.0	-35.
2	NAV_WAYPOINT	0.0	0.0	0.0	0.0	-35.
3	NAV_WAYPOINT	0.0	0.0	0.0	0.0	-35.
4	NAV_WAYPOINT	0.0	0.0	0.0	0.0	-35.

```
AUTO  ARM  GPS: OK6 (10)  Vcc 5.00  Radio: --  INS  MAG  AS  RNG  AHRS  EKF  LOG  FEN  RC
Batt: 89%/12.19V 29.2A   Link 1 OK 100.0% (16665 pkts, 0 lost, 0.00s delay)
Hdg 272/275  Alt 19m  AGL 24m/23m  AirSpeed 4m/s  GPSSpeed 4m/s  Thr 37  Roll 0  Pitch -18  Wind --/-
WP 2/5  Distance 97m  Bearing 277  AltError 0L  AspdError 0.0H  FlightTime 0:45  ETR 1:26
DISARMED
APM: Arming motors
Got MAVLink msg: COMMAND_ACK {command : 400, result : 0}
ARMED
Got MAVLink msg: COMMAND_ACK {command : 11, result : 0}
waypoint 1
Mode AUTO
APM: EKF2 IMU0 in-flight yaw alignment complete
APM: EKF2 IMU1 in-flight yaw alignment complete
height 10
APM: Reached command #1
waypoint 2
```

Figure 10.3: SITL in action during Waypoint Nav test.

10.3 Algorithm Example: Take-Off

The programs written with this API, communicates with vehicle autopilots using the MAVLink protocol, which defines how commands, telemetry and vehicle settings/parameters are sent between vehicles, companion computers, ground stations and other systems on a MAVLink network. At high level, the steps are: check that the vehicle is *able* to arm, set the mode to GUIDED, command the vehicle to arm, takeoff and block until we reach the desired altitude. *Unfortunately, explaination or mention of any code snippets is not within the scope of this report, hence this report will not contain any code or code explanations and will contain only algorithms, especially the ones that can be explained without any code reference*.

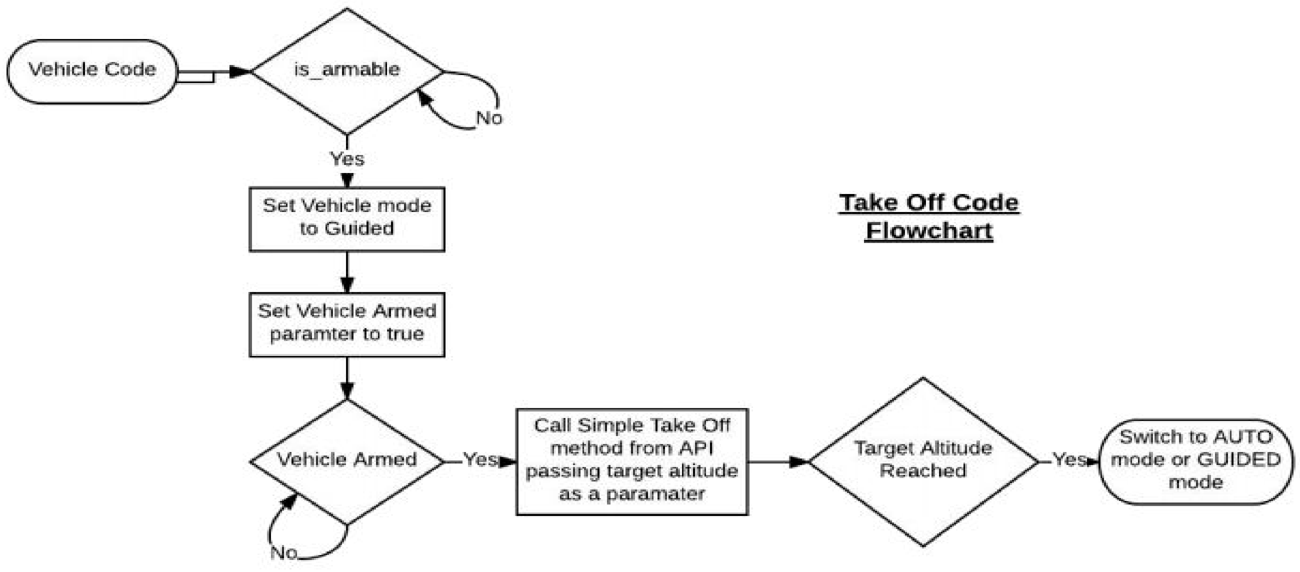

Vehicle Code
is_armable
No
Yes
Set Vehicle mode to Guided
Set Vehicle Armed paramter to true
Vehicle Armed
No
Yes
Call Simple Take Off method from API passing target altitude as a paramater
Target Altitude Reached
Yes
Switch to AUTO mode or GUIDED mode
Take Off Code Flowchart

Figure 10.4: Take-off Algorithm

10.4 Vehicle information

Vehicle state information is exposed through vehicle *attributes* which is read and observed and vehicle settings which is read, written, iterated and observed using *parameters* (a special attribute). Attributes are populated by MAVLink messages from the vehicle.

10.5 Observing attribute changes

Any of the vehicle attributes can be observed and monitored for changes without the need for polling.

Listeners ("observer callback functions") are invoked differently based on the *type of observed attribute*. Attributes that represent sensor values or other "streams of information" are updated whenever a message is received from the vehicle. Attributes which reflect vehicle "state" are only updated when their values change.

10.6 Parameters

Vehicle parameters provide the information used to configure the autopilot for the vehicle-specific hardware/capabilities. The API downloads all parameters when first connected to the UAV (forcing parameter reads to wait until the download completes), and subsequently keeps the values updated by monitoring vehicle messages for changes to individual parameters. This process ensures that it is always safe to read supported parameters, and that their values will match the information on the vehicle.

Parameters can be read, set, observed and iterated using an attribute of a particular object reserved for parameter operations.

The vehicle parameters can be observed and monitored for changes without the need for polling. The parameters are cached, so that callback functions are only invoked when parameter values change.

10.7 Missions (AUTO Mode)

AUTO mode is used run pre-defined waypoint missions on this drone. The API provides methods to download and clear the current mission commands from the vehicle, to add and upload new mission commands, to count the number of waypoints, and to read and set the currently executed mission command. High-level mission planning functionality can be built upon these abilities.

There are three types of commands:

- ***NAVigation commands*** are used to control vehicle movement, including takeoff, moving to and around waypoints, changing altitude, and landing.
- ***DO commands*** are for auxiliary functions that do not affect the vehicle's position (for example, setting the camera trigger distance, or setting a servo value).
- ***CONDITION*** *commands* are used to delay *DO commands* until some condition is met. For example *MAV_CMD_CONDITION_DISTANCE* will prevent DO commands executing until the vehicle reaches the specified distance from the waypoint.

During a mission at most one *NAV* command and one *DO* or *CONDITION* command can be running **at the same time**. *CONDITION* and *DO* commands are associated with the last *NAV* command that was sent: if the UAV reaches the waypoint before these commands are executed, the next *NAV* command is loaded and they will be skipped.

At the end of the mission the vehicle will enter LOITER mode. New commands can be added to the mission, but needs to toggle from/back to AUTO mode to start it running again.

10.8 Few Important Use-Cases

This section contains the documentation that demonstrates common use-cases, and show (among other things) how to use the API to query vehicle state and parameters, and how to control a vehicle during missions and outside missions using custom commands.

10.8.1 Go To

This demonstrates the arming and launching the UAV in GUIDED mode, and it travels towards a number of target points, and then returns to the home location. The target locations are centered around the home location when the Simulated Vehicle is booted. The code has three distinct sections: arming and takeoff, flight to two (multiple) locations, and return-to-home.

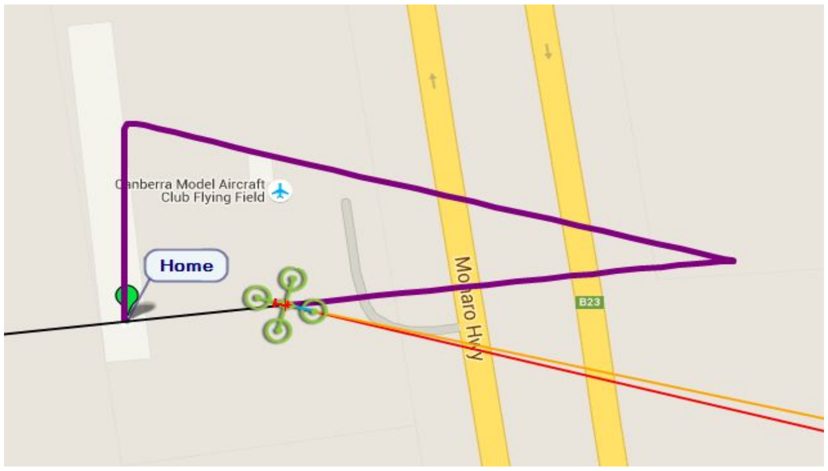


Figure 10.5: GOTO Flight path with a single waypoint

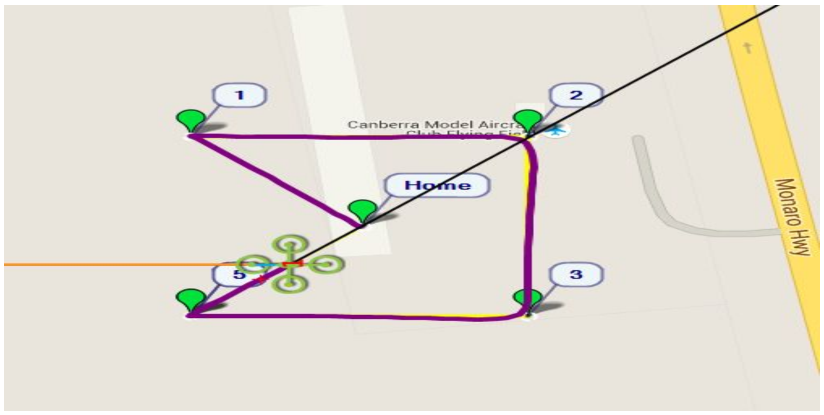
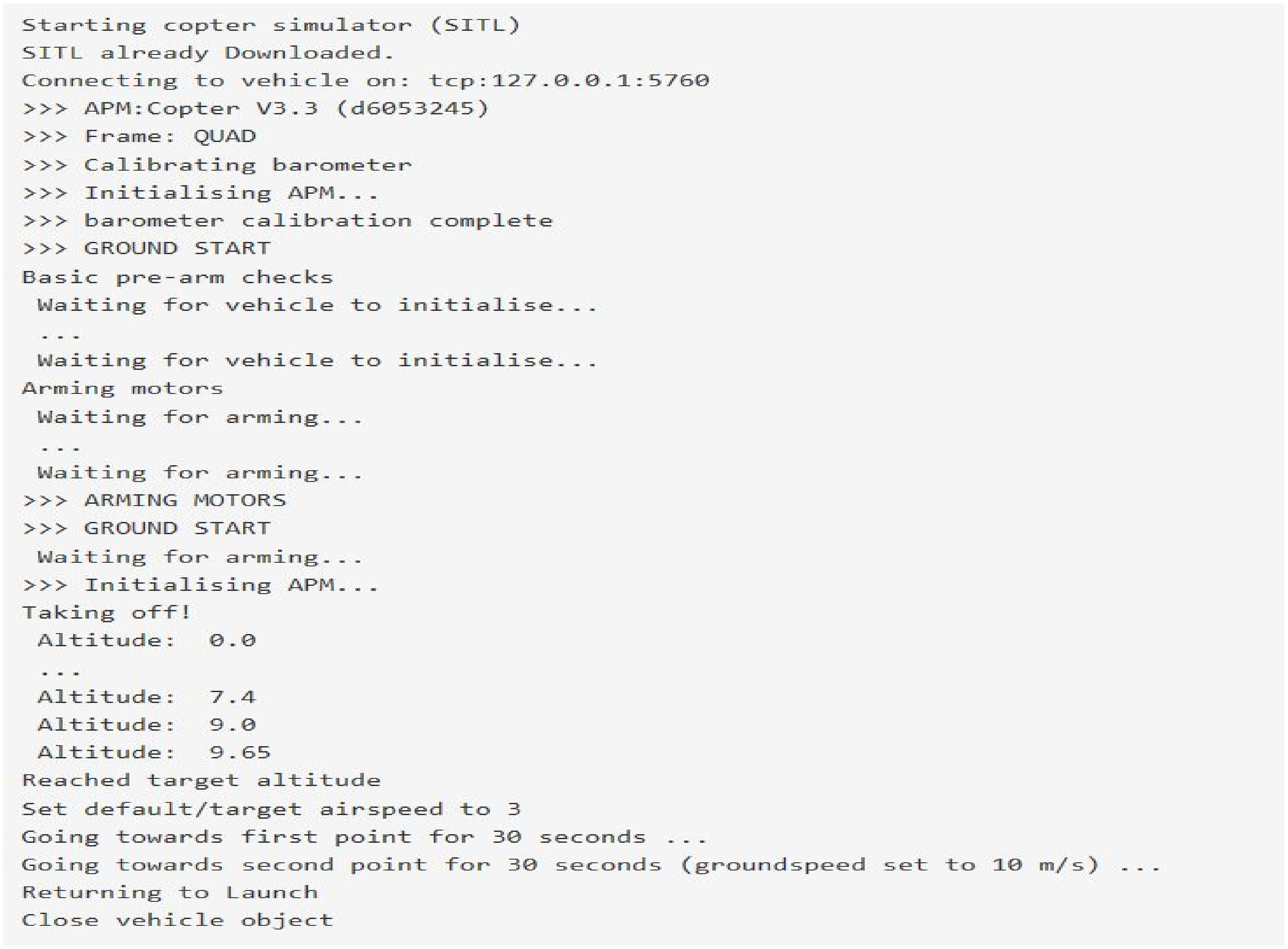


Figure 10.6: GOTO flight path with multiple goto waypoints

- **Command Line based Test Result**:



```
Starting copter simulator (SITL)
SITL already Downloaded.
Connecting to vehicle on: tcp:127.0.0.1:5760
>>> APM:Copter V3.3 (d6053245)
>>> Frame: QUAD
>>> Calibrating barometer
>>> Initialising APM...
>>> barometer calibration complete
>>> GROUND START
Basic pre-arm checks
 Waiting for vehicle to initialise...
 ...
 Waiting for vehicle to initialise...
Arming motors
 Waiting for arming...
 ...
 Waiting for arming...
>>> ARMING MOTORS
>>> GROUND START
 Waiting for arming...
>>> Initialising APM...
Taking off!
 Altitude:  0.0
 ...
 Altitude:  7.4
 Altitude:  9.0
 Altitude:  9.65
Reached target altitude
Set default/target airspeed to 3
Going towards first point for 30 seconds ...
Going towards second point for 30 seconds (groundspeed set to 10 m/s) ...
Returning to Launch
Close vehicle object
```

Figure 10.7: Command Line based Test Result of GOTO algorithm

10.8.2 Follow Me

The *Follow Me* algorithm moves the vehicle to track the pilot's position, using location information from a USB GPS attached to the pilot's (Linux) laptop.

This connects to the vehicle, and performs the usual taking off, and closing the vehicle object.

All this does in addition to the usual, is attempt to get a gps socket and read the location in a two second loop. If it is successful it reports the value and uses the GOTO feature to move to the new position. The loop exits when the mode is changed.

- **Command Line based Test Result**:

```
Starting copter simulator (SITL)
SITL already Downloaded.
Connecting to vehicle on: tcp:127.0.0.1:5760
>>> APM:Copter V3.4-dev (e0810c2e)
>>> Frame: QUAD
Link timeout, no heartbeat in last 5 seconds
Basic pre-arm checks
Waiting for GPS...: None
...
Waiting for GPS...: None
Taking off!
 Altitude:  0.019999999553
 ...
 Altitude:  4.76000022888
Reached target altitude
Going to: Location:lat=50.616468333,lon=7.131903333,alt=30,is_relative=True
...
Going to: Location:lat=50.616468333,lon=7.131903333,alt=30,is_relative=True
Going to: Location:lat=50.616468333,lon=7.131903333,alt=30,is_relative=True
User has changed flight modes - aborting follow-me
Close vehicle object
Completed
```

Figure 10.8: Command Line based Test Result of Follow Me algorithm

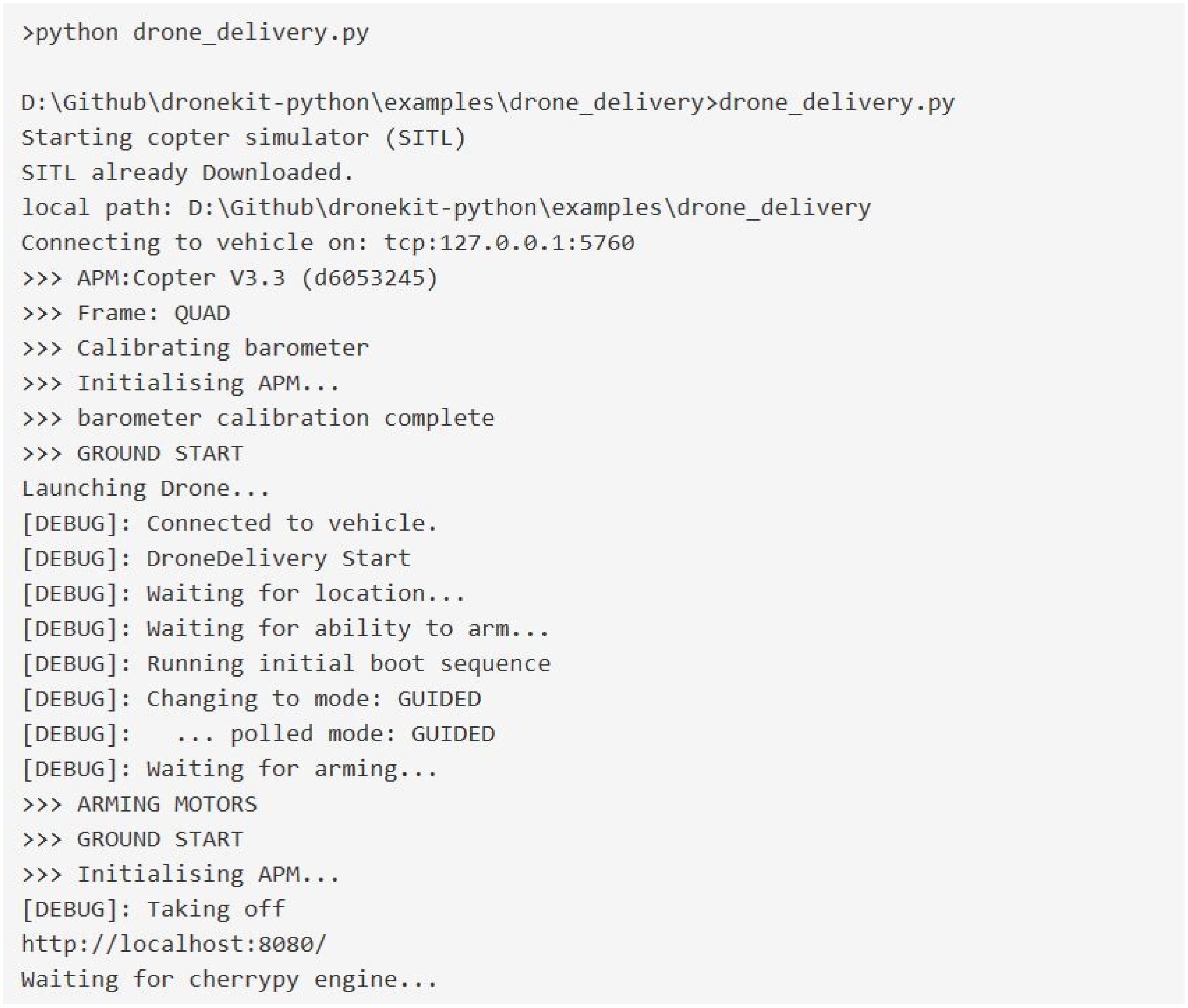
10.8.3 Drone Delivery

This use case shows a CherryPy based web application that displays a mapbox map to let the pilot view the current vehicle position and send the vehicle commands to fly to a particular latitude and longitude.

New functionality demonstrated by this test feature includes:

- Using attribute observers to be notified of vehicle state changes.
- Starting *CherryPy* from a DroneKit application.

➔ **Command line based Test Result:**

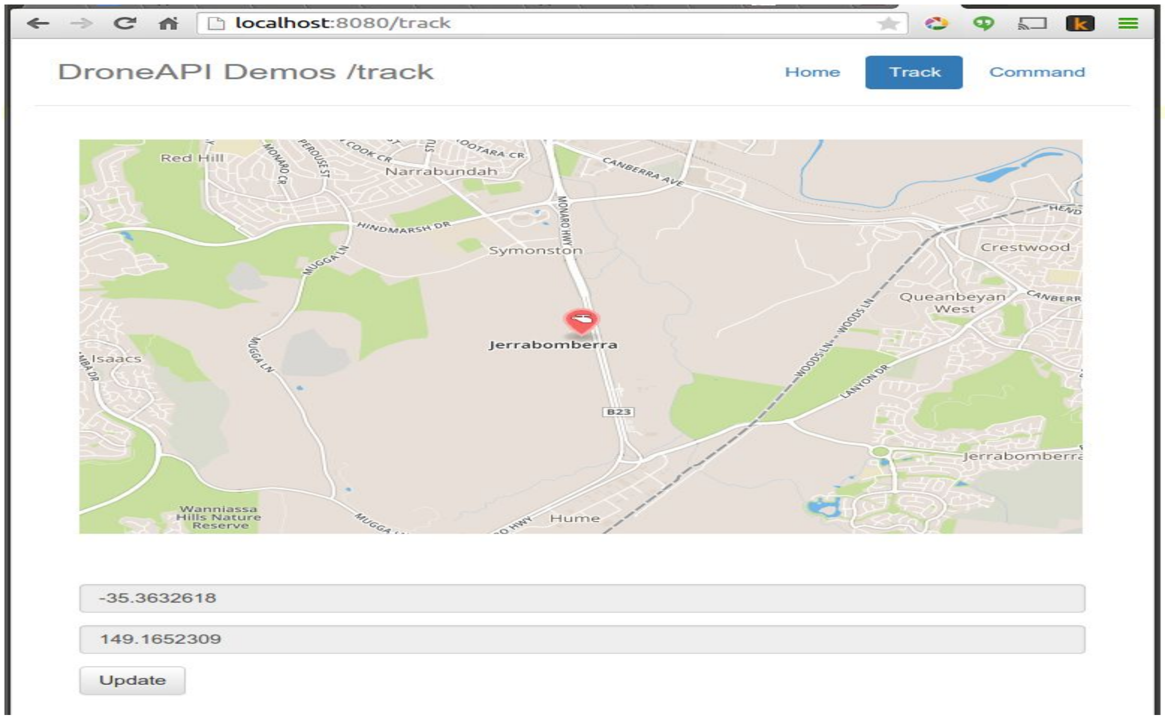


```
>python drone_delivery.py

D:\Github\dronekit-python\examples\drone_delivery>drone_delivery.py
Starting copter simulator (SITL)
SITL already Downloaded.
local path: D:\Github\dronekit-python\examples\drone_delivery
Connecting to vehicle on: tcp:127.0.0.1:5760
>>> APM:Copter V3.3 (d6053245)
>>> Frame: QUAD
>>> Calibrating barometer
>>> Initialising APM...
>>> barometer calibration complete
>>> GROUND START
Launching Drone...
[DEBUG]: Connected to vehicle.
[DEBUG]: DroneDelivery Start
[DEBUG]: Waiting for location...
[DEBUG]: Waiting for ability to arm...
[DEBUG]: Running initial boot sequence
[DEBUG]: Changing to mode: GUIDED
[DEBUG]:   ... polled mode: GUIDED
[DEBUG]: Waiting for arming...
>>> ARMING MOTORS
>>> GROUND START
>>> Initialising APM...
[DEBUG]: Taking off
http://localhost:8080/
Waiting for cherrypy engine...
```

Figure 10.9: Command Line based Test Result of Drone Delivery application

➔ **Track Feature test result:**



-35.3632618

149.1652309

Figure 10.10: Demo of Tracking feature

➔ **Command Feature test result:**

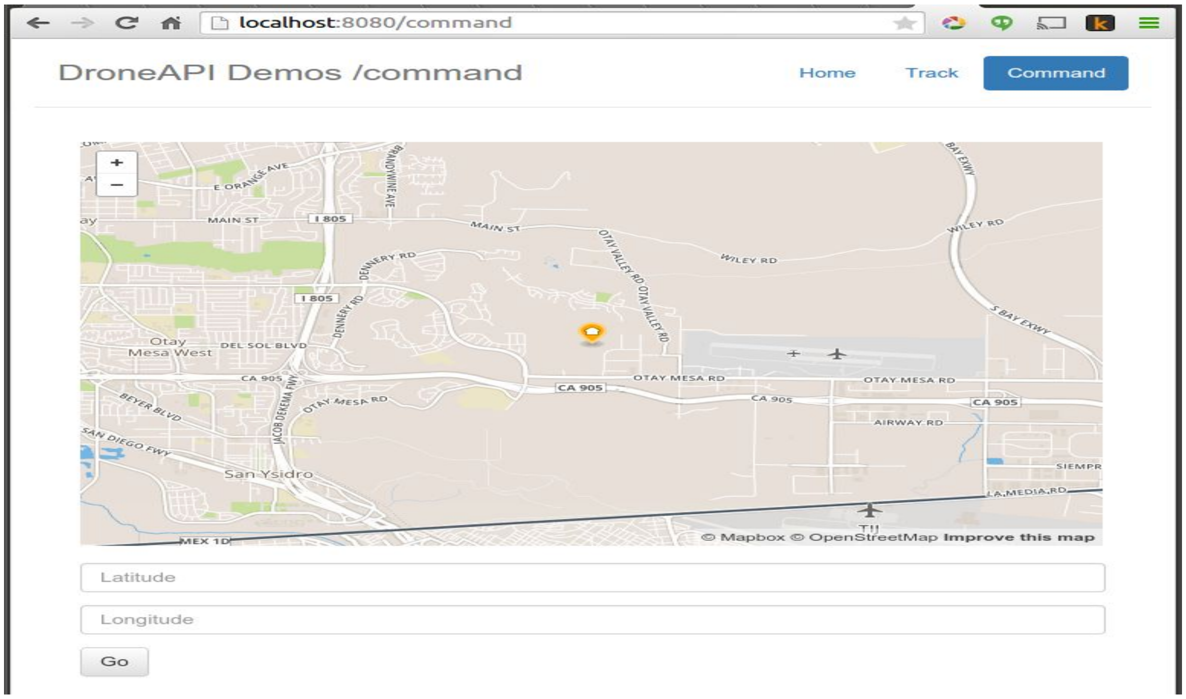


Figure 10.11: Demo of Command Interface

10.8.4 Flight Replay

This test case creates and runs a waypoint mission using position information from a TLOG file.

The log used in this contains around 2700 points. This is too many points to upload to the autopilot (and to usefully display). Instead only those points are added that are more than 3 metres away from the previously kept point, and only 99 points are stored in total. After 60 seconds the mission is ended by setting the mode to RTL (return to launch).

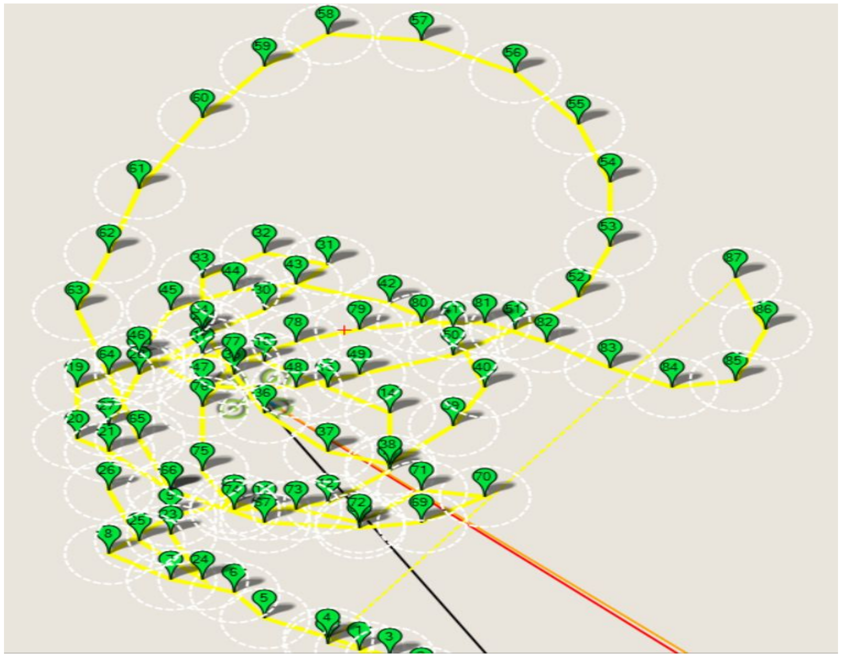


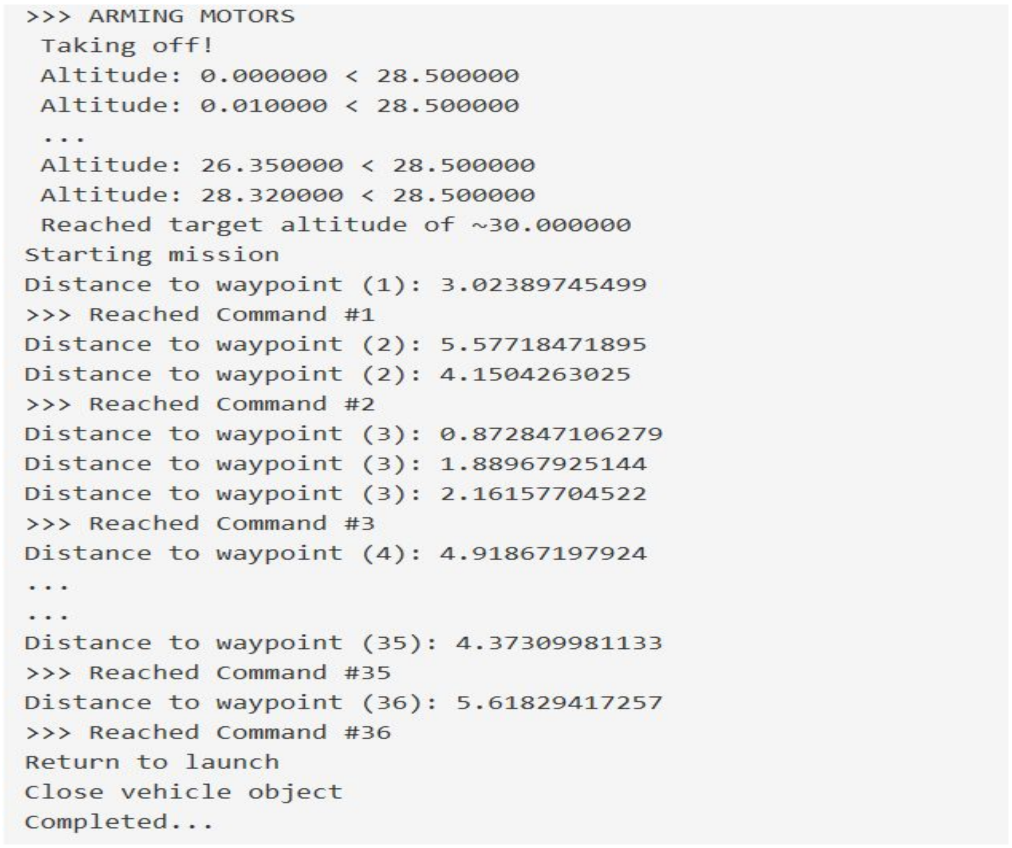
Figure 10.12: 99 point mission generated from log

The test parses a **flight.tlog** file for position information. First all the points are read. Then the first 99 points are kept that are at least 3 metres separated from the preceding kept point.

For safety reasons, the altitude for the waypoints is set to 30 meters (irrespective of the recorded height).

The method used to reduce the number of points is fairly effective, but it can be improved by grouping some of the waypoints, and mapping others using spline waypoints.

➔ **Command Line Based Test Result:**



```
>>> ARMING MOTORS
 Taking off!
 Altitude: 0.000000 < 28.500000
 Altitude: 0.010000 < 28.500000
 ...
 Altitude: 26.350000 < 28.500000
 Altitude: 28.320000 < 28.500000
 Reached target altitude of ~30.000000
Starting mission
Distance to waypoint (1): 3.02389745499
>>> Reached Command #1
Distance to waypoint (2): 5.57718471895
Distance to waypoint (2): 4.1504263025
>>> Reached Command #2
Distance to waypoint (3): 0.872847106279
Distance to waypoint (3): 1.88967925144
Distance to waypoint (3): 2.16157704522
>>> Reached Command #3
Distance to waypoint (4): 4.91867197924
...
...
Distance to waypoint (35): 4.37309981133
>>> Reached Command #35
Distance to waypoint (36): 5.61829417257
>>> Reached Command #36
Return to launch
Close vehicle object
Completed...
```

Figure 10.13: Command Line based Test Result of Flight Replay test case

10.8.5 Circular Trajectory Maneuver

The system waits for a valid mission to be uploaded. It needs only one (the first) waypoint. The drone takes off and then performs a circular trajectory around the way point. It keeps constant forward velocity, while the lateral velocity is calculated in order to keep the drone at the same distance from the waypoint. The heading is calculated for maintaining a 90 degrees angle respect to the bearing.

The outline of the algorithm is as follows:

--> Follow circular trajectory while keeping the forward speed constant and adjusting its heading to be tangent to the trajectory.

--> Offtrack error needs to be controlled by adjusting the lateral speed, Vy.

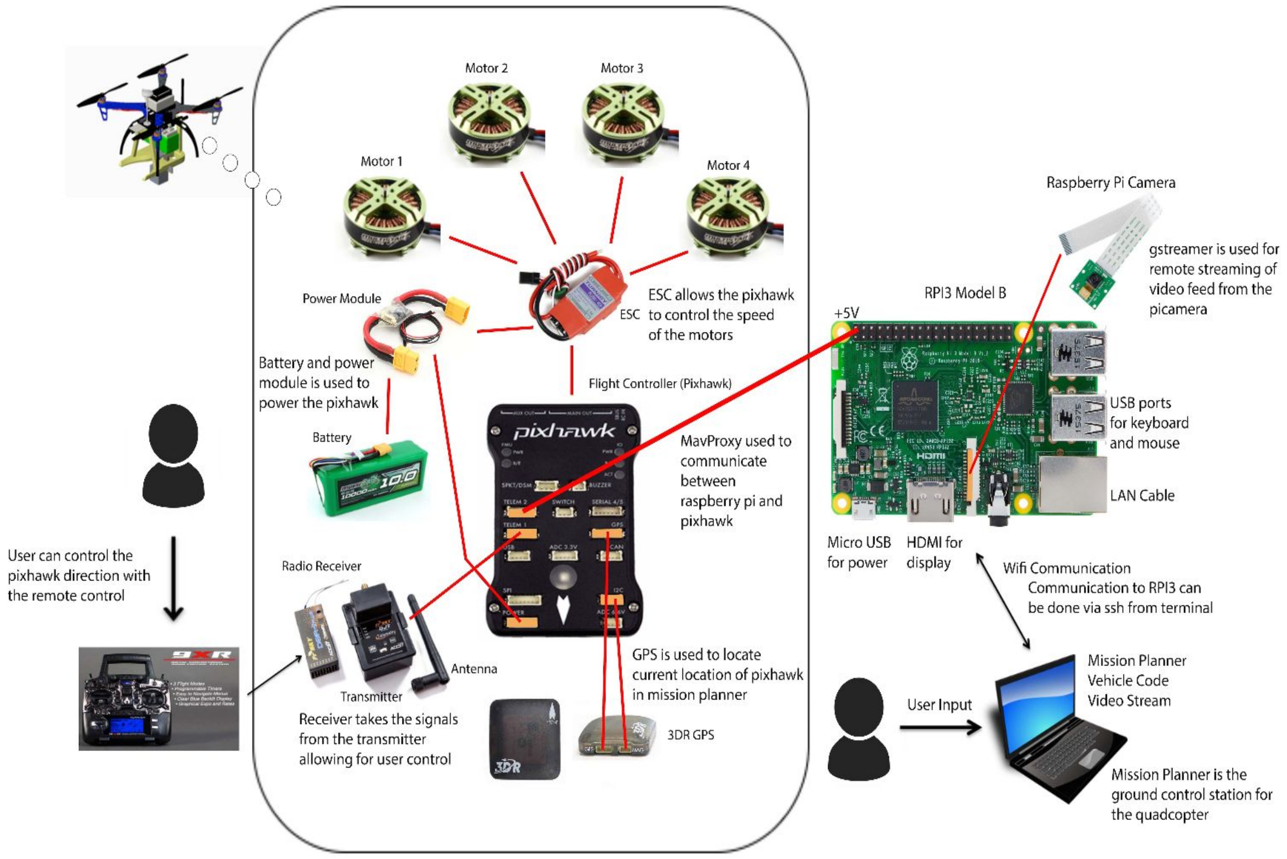
--> Lateral speed will be used to keep the drone on the track.

--> Provide WP with GCS, and the script will create a simple circular path around the waypoint
--> The drone will be on Velocity mode. Set the forward speed, Vx, as constant
and the heading as tangent to the trajectory and Vy will be used to keep the drone on the trajectory.
--> In Mission state, implement feedback control loop, where heading is set as 90 deg from Bearing, Vx as constant ground speed and Vy is proportional to offtrack error (Thus, its the distance between vehicle position and circumference).
--> Position will be check wrt radius.
--> After n turns, return.

Chapter 11

Conclusion

This report concludes my work at the Indian drone company. The work highlighted in this report primarily revolves around imparting intelligence and autonomous behavior to a drone (Quadcopter) powered by Rpi and Pixhawk, communicating via the MAVlink protocol. This report began with Quadcopter basics and stretched through to the primary focus, the Advanced Autonomous Guidance Assisted Flying System (AAGAFS). Midway, it traversed through critical concepts that are required to be understood to be able to absorb the essence of AAGAFS.
These concepts include various components and UAV subsystems at play to either assist AAGAFS or to act independently in parallel to AAGAFS, but all serving the same purpose of creating a stable autonomous flight. Most tests were successful which further propels the idea of the existence of an era where autonomous UAV's will be available commercially for sorts of purposes and industries ranging from Agriculture to Search&Rescue to Delivery Service, and will be the solution to the world's aching concerns. Drones will not just serve the commercial purposes, but also has the potential to impact the lives of people, and this work compiles into a building block, a part of a much bigger challenge. The figures below summarise the overall system design, autopilot design and the interface design between the systems levels.



Motor 2
Motor 3
Motor 1
Motor 4
Raspberry Pi Camera
gstreamer is used for remote streaming of video feed from the picamera
RPI3 Model B
+5V
Power Module
ESC
ESC allows the pixhawk to control the speed of the motors
Battery and power module is used to power the pixhawk
Flight Controller (Pixhawk)
Battery
MavProxy used to communicate between raspberry pi and pixhawk
USB ports for keyboard and mouse
LAN Cable
Micro USB for power
HDMI for display
User can control the pixhawk direction with the remote control
Radio Receiver
Wifi Communication
Communication to RPI3 can be done via ssh from terminal
Antenna
Transmitter
GPS is used to locate current location of pixhawk in mission planner
Receiver takes the signals from the transmitter allowing for user control
3DR GPS
User Input
Mission Planner
Vehicle Code
Video Stream
Mission Planner is the ground control station for the quadcopter

Figure 11.1: Full system overview

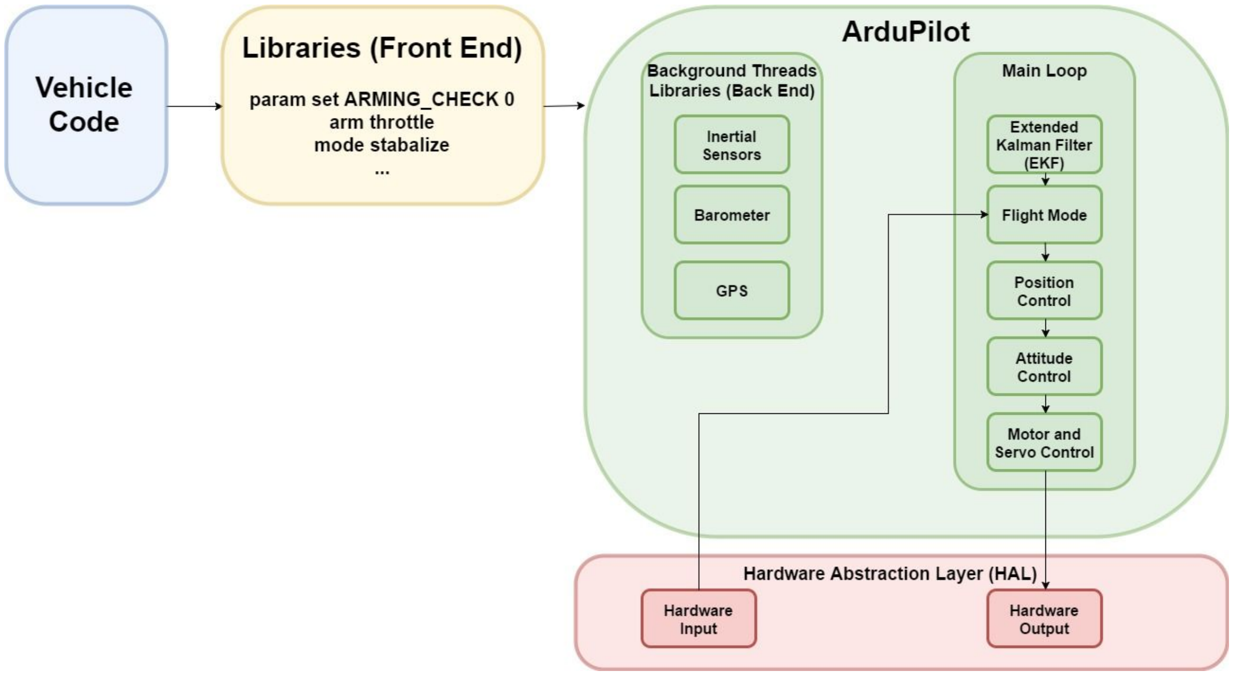


Figure 11.2: Autopilot software block diagram

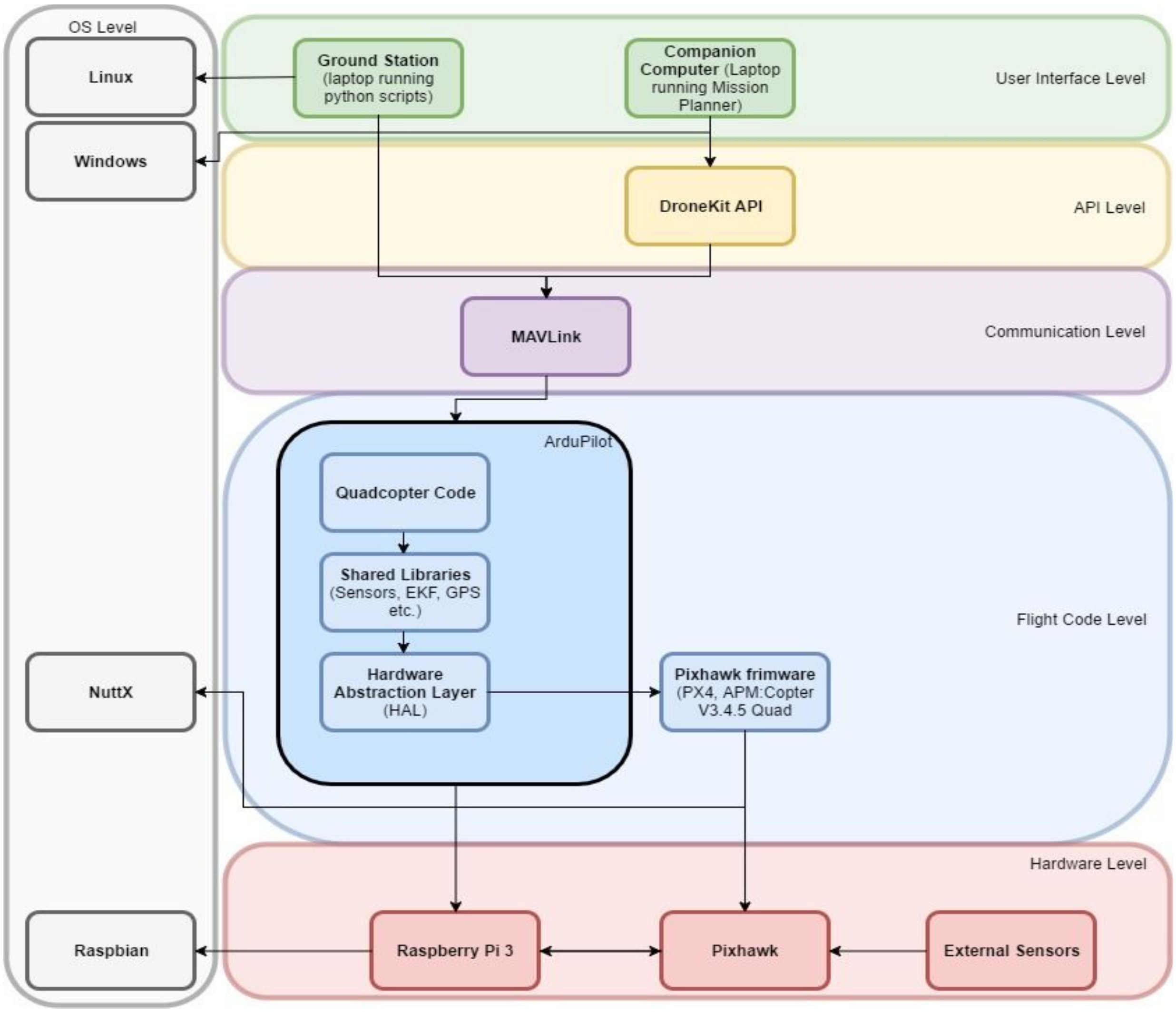

OS Level
Linux
Windows
NuttX
Raspbian
Ground Station (laptop running python scripts)
Companion Computer (Laptop running Mission Planner)
User Interface Level
DroneKit API
API Level
MAVLink
Communication Level
ArduPilot
Quadcopter Code
Shared Libraries (Sensors, EKF, GPS etc.)
Hardware Abstraction Layer (HAL)
Pixhawk frimware (PX4, APM:Copter V3.4.5 Quad
Flight Code Level
Hardware Level
Raspberry Pi 3
Pixhawk
External Sensors

Figure 11.3: Interface Design Block Diagram