3rd Year Project

Quadcopter Project

John Walsh

Table of Contents

Introduction

The project I'm undertaking has a number of different technologies involved, you might think this project is more suited to electrical engineering student because it involves building and assembling most of the electronics and airframe of a quadcopter, however the flight controller used in this project allows the user to completely customize the software running on the device to control the movement of the craft. Later in the project I'll be going over why I choose the parts I did and how I'm going to integrate them into my project.

Moving onto the software side of things I'll be using mainly C / C++ to write the basic flight stabilization software that will be loaded onto the flight controller. The specific libraries I'll be using to develop this type of the software is called 'ArduPilot'. To interface with flight controller I'll need a customized version of the Arduino's IDE to load the software. Other tools like 'Install Mission Planner' will also be required to calibrate the flight controller initially configure the device. The flight controller hardware is designed by 3DR and is fully capable of autonomous flight, which is why I choose this unit as I could further adapt the controller for other projects.

Drone

My overall goal of this project is to research and develop my own flight stabilization software for the quadcopter to allow fly, this involves first building the frame of the craft and adding the necessary electronics like the speed controllers, wiring loom, motors, radio receiver, LiPo battery, propellers and so on and then finally learn how to actually fly a quadcopter. I already have experience in flying R/C helicopters and small planes so it shouldn't take too long.

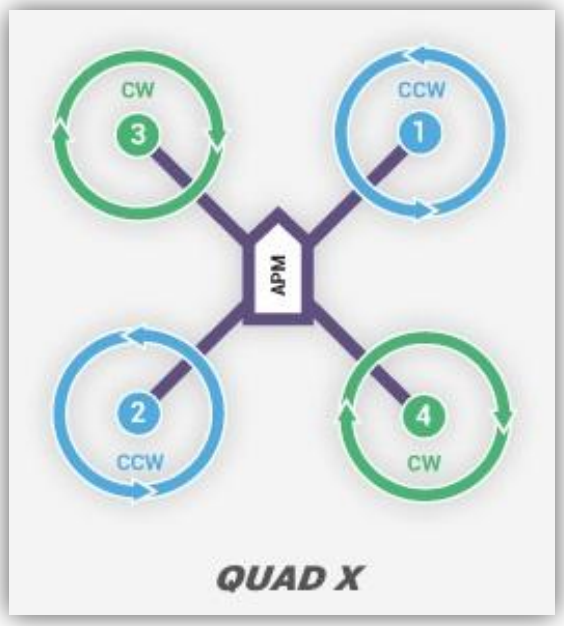
The Building of the Quadcopter

In this chapter I'll be going through the process of building this little R/C machine. I'll be going over the different options available when building one of these machines, from the flight controllers, transmitters, motors and other essential parts required.

The Frame

Currently there are many different frames available to users when building a R/C drone. For my own project I've opted for a 250mm X pattern style of frame for my own quad rotor build. Other designs and patterns are available that suit other applications like lifting objects or racing. FPV (First-person view) racers usually pick the X and H pattern frames for lightness and compacted designs for high speed racing through buildings and forests for example.

The X style Quadcopter



As you can see in the picture above you'll notice I'll have to label the different propellers CW (Clock-wise) and CCW (Counter Clock-wise) for the flight controller to apply the correct upward lift needed to stabilize the craft. This can be easily achieved by switching the polarity of the motor wires to the ESC's (Electronic Speed Controllers).

The frame comes disassembled and requires me to route all the wires to and from the ESC's, receiver and flight controller. The frame I choose also has an option of adding camera gimbal to the underside of the frame to allow a user to control the camera movement, with either a switch on the transmitter or FPV goggles to control the camera with head tracking if available. I purchased the frame on eBay and was best suited for my needs, it's a light weight and tough frame for FPV racers and hobbyist.

In this section of the report I'll detail the building process of the quadcopter. I know this may not be completely relevant to the software development side of things but I found it quite interesting and wanted to cover this area.

The frame came in a flat pack, I assembled the frame making sure to use thread-locking fluid on the screws to prevent losing due to vibrations during flight.

First Stage

The Motors & ESCs

After making sure everything was assembled correctly, I moved on to mounting the motors. The motors I bought were packaged with the ESC's for the quadcopter. The matching sets was helpful because I didn't need to worry about getting correctly matching set of ESC's and motors. I bought them on eBay, it's also a great deal because propellers with the package.

Motors & ESC's

During the assemble of any quadcopter it's important to have a balanced layout of the components. If the quadcopter were unbalanced, the flight controller would have a difficult time keeping it in flight. I'll include a number of screenshots throughout the building process to show you where I located the parts of the quadcopter.

Adding the motors to the frame is the next step in the building of the quadcopter. I had a problem when attaching the motors with the supplied screws, they were for some reason binding against the internals of the motor, so I bought some titanium screws of the correct length.

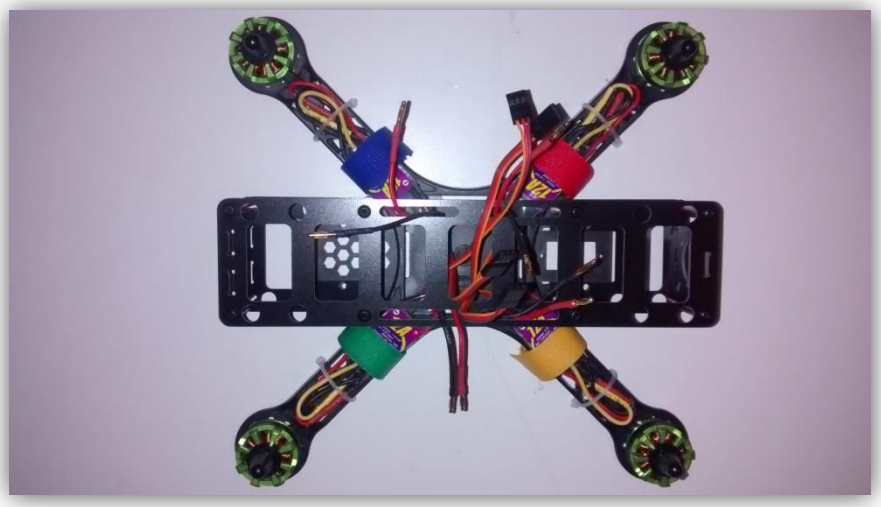
Motors Added

For the motors to work correctly I need to add the ESC's for each rotor. The ESC's came bundled with the motors as stated above. For the motors to work correctly the ESC's must output more than the motors can consume to prevent stalling. Stalls are like brown outs that happen when there's a partial drop in power.

ESC's

Mounting the devices is fairly straight forward, I've attached an ESC to each arm of the frame to balance the weight.

Added ESC's



As you can see, the ESC's have many inputs and outputs. The signal cable will hook up to the flight controller. The three outputs from each ESC will connect to their corresponding motor. I'll include a wiki link here to cover the workings of these brushless type ESC's.

The GPS / Compass Module

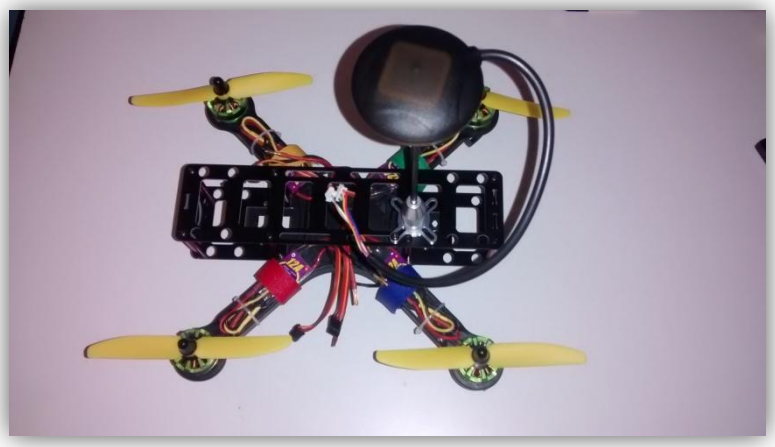
The next part of build is the GPS / Compass module, although it's not necessary to install one of these modules if you plan on flying it normally without any autonomous behaviour. I may include some autonomous functionality so I decided to include it in the build. The GPS device I'll be installing is the 'Ublox-NEO-6M-GPS' module.

GPS Module

You can see there's also a securing kit included to fasten the module to the quadcopter, it's also important to isolate the device away from the rest of the electronics because electrical interference can mess with the compass's orientation.

From the picture below you can see the progress of the build. With the GPS module now installed I can move onto the flight controller installation.

GPS - Installed



The GPS module's cables are fairly short so I need to mount the flight controller in an area that's close to the cables. I planned on placing the flight controller inside the cage of the quadcopter's frame but that didn't work out so the next best place was on top of the frame, although I'm quite worried if I were to crash the quadcopter it could damage the sensitive electronics. I've also temporarily installed the props to roughly show how it'll look when it's finished.

The Flight Controller & Battery

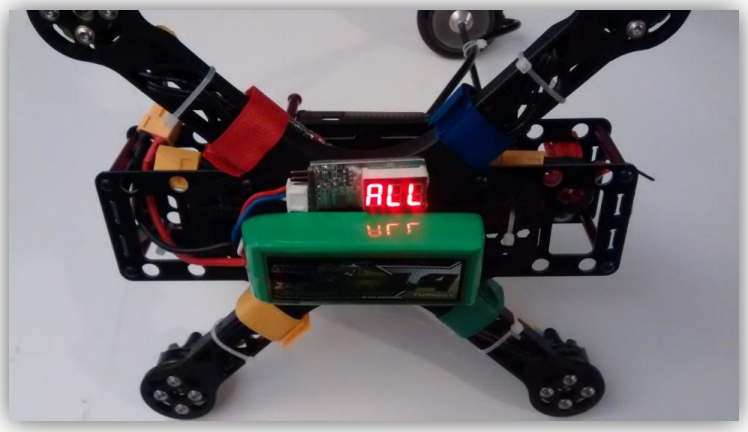
The flight controller is next on the list of things to install to the frame of the quadcopter. As I said before it'll be mounted on the top side of the frame, exposed to the elements. For now I'll install these aluminium posts that came with another frame I got on eBay to help protect the flight controller from rough landings.

Flight Controller - Ardupilot

You might notice the large cable next to the flight controller is the power module I'll be using to power the flight controller of the quadcopter. This will supply 5V at @2A of dedicated current to the controller. This is by far the most reliable way of powering the electronic systems of the quadcopter.

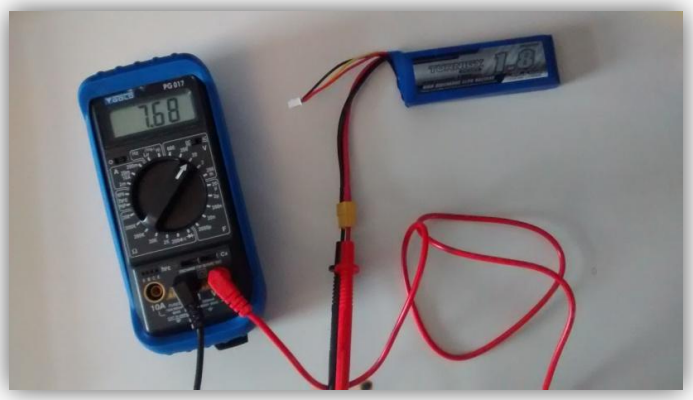
On the topic about power, the entire quadcopter will be powered by a 3S C40-80 1400mAh lithium polymer battery. This battery can roughly give the quadcopter about 10mins of run time I believe. I still need to a proper flight test to see if the battery is up to the job.

The Battery Location



I also installed a LiPo battery monitor near the balancing cable of the battery to check the status of the cells in the battery in real-time. LiPo cells are fairly sensitive to the voltage they store. If the voltage is too high the cells can start expand outwards meaning excessive pressure has built up in the cell that can lead to a battery exploding, which doesn't sound like fun. Another thing to keep an eye on is the batteries in storage, their voltage needs to be monitored from time to time to see if the cell voltages are above 2.8 - 3 volts. If not damage can also incur to the cells. The correct cell voltage to store these types of batteries is around 3.8 volts, I use a multi-meter to check the voltage of the individual cells.

Multi-Meter Testing



The Radio Receiver

The radio receiver in the quadcopter build is the Hitec 2.4Ghz Optima 7, this is a compact 7 Channel receiver that has few km range.

Hitec Optima 7

The receiver is powered by the flight controller's input pins. There's nothing too special about this device, it will simply take the input from my transmitter and output the signals to the flight controller. The flight controller will then make adjustments to those input signals and spin the four motors appropriately.

The Radio Transmitter

The transmitter I'll be using is the Hitec Eclipse 7 Pro. It's considered a good transmitter by most of the flying community so that's why I choose to buy it and use it fly my quadcopter.

The Transmitter - Hitec Eclipse 7 Pro

This transmitter supports up to 7 channels of control, plenty for my quadcopter which only requires 4 channels to fly properly.

Different options

In this section of the report I'll include why I chose each specific part, especially when it comes to the flight controllers because I really need a platform that I can openly develop my own software.

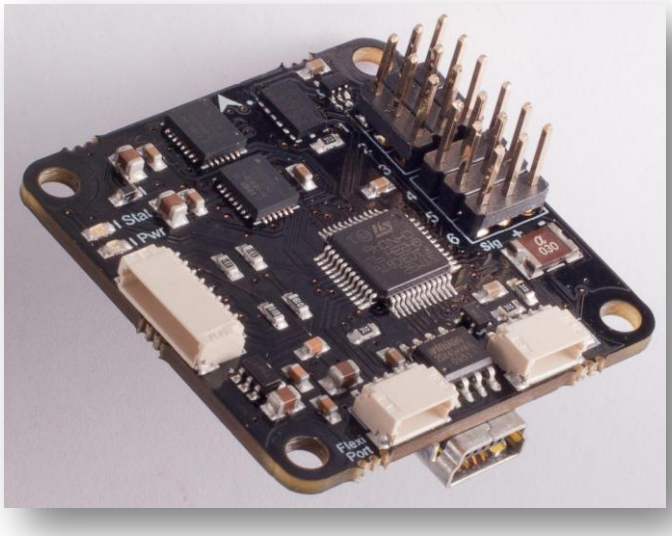
Flight Controllers

On the market today there are plenty of flight controllers to choose from to control drones. But I needed a controller that would allow me to develop and experiment on that will allow me to upload my own software. There's also a need to match the flight controller with flying experience and flying style. As I'm a beginner to developing on these types of devices, I choose to develop on a platform that has plenty of well documented instructions for the software side of things. That platform is the ArduPilot I mention earlier on in the report provides a great environment to develop and build my own autonomous drone. Although I'm not sure I'll be adding the autonomous feature in my own software build because I may inadvertently make my drone fly miles away without ever finding it again!.

OpenPilot

Other flight controllers exist that allow to build and develop your own drone. The OpenPilot board allows you to build an inexpensive flying drone. Although this flight controller does not allow you to completely and openly develop on without hacking some of the boards features. The board's firmware is written in C while the ground station software used to communicate with the board is written in C++ with some extra libraries.

The OpenPilot Board



The board has a much smaller footprint than the Ardupilot, it would've be easier to work with this board as I could integrate the board inside the frame of the

quadcopter with exposing it as much as the ArduPilot. The ArduPilot also has a few extra built in features on the board, like the GPS and real-time telemetry support.

Naza-M V2

A more expensive and locked down option is the Naza flight controller which allows users to fully control and interface with the device in many different ways like sending commands over a Bluetooth link from an iPad to the drone to follow you or follow other drones, cars, bikes or whatever you wish depending on what camera and GPS modules you have installed and enabled. This flight controller is a product of DJI and have perfected the device and software over years of development.

Naza-M V2

However this flight controller is completely proprietary and wouldn't of suited my project needs as I require a platform with open-source software and can be freely changed and modified. Don't get me wrong this is a great flight controller to install on your quadcopter, especially if you need a ready to run type setup that doesn't require much configuration.

Again the ArduPilot covers all the requirements I need to develop my own stabilization software, unlike this Naza flight controller.

Moving onto the next section will bring to the software development side of project that I'll be writing to eventually stabilize the quadcopter in mid-air, although it'll be quite basic I need to be careful not to make any obvious mistakes in the code that could potentially spell disaster for my quadcopter project.

Software Development

In this section of the report I'll be going over the different areas of the software development stages of my project. This will include all of the tools and sources of information I've found to help me develop my own code to help keep my quadcopter in flight. I've never developed stuff like this before and have only developed software over the years that runs on devices like PCs, Phones and Tablets, so I will be borrowing some ideas from different sources to help me develop this type of software. This type of software development also comes with its risks, you have to realize that this craft is not a toy, if for example there was a bug in my code that would prevent me from lowering the throttle it could easily fly anywhere, it could end up hurting someone. Another example, a craft like this travelling at 40km/h - 60km/h at a weight just over a half a kilo with four 120 watt motors spinning props anywhere from 2,000 rpm to 22,000 rpm, were to strike someone's head, it could seriously injure them, this is why I'm taking my time in developing this software to make sure I cover every aspect of safety while I research and develop my own running firmware.

ArduPilot Software

The ArduPilot has great software libraries that allow you to develop your own running software and upload it to the device. ArduPilot doesn't just allow to develop for one platform but for many different types of craft. You could if you wished, run the device on a land based rover, plus planes and helicopters can also be controlled by the adaptable ArduPilot. It's a great piece of kit, I highly recommend it to anyone wishing to build a project similar to what I'm doing here.

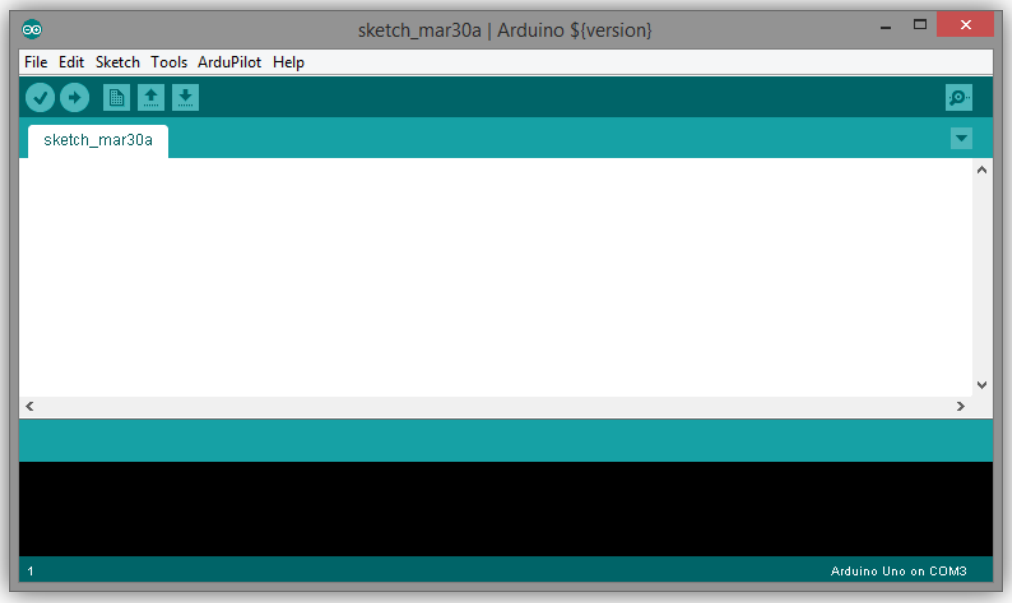
The code base for the ArduPilot is here, I'll be borrowing some ideas from the original base code and to build on top of the existing libraries. As a I stated before safety plays a big part in developing the code for a flying craft as you could imagine. I'm not going to try and develop everything from scratch here, I plan on only building on top of proven and tested libraries.

The IDE

The IDE being used to develop the software in this project is a modified version of the Arduino IDE. It's a version developed for the ArduPilot community. I could also build and compile the source code within Visual Studio but will require modifications to work with the APM board.

The IDE can be found here, I'll include some information next in the report on how to connect, verify and upload software to the APM board.

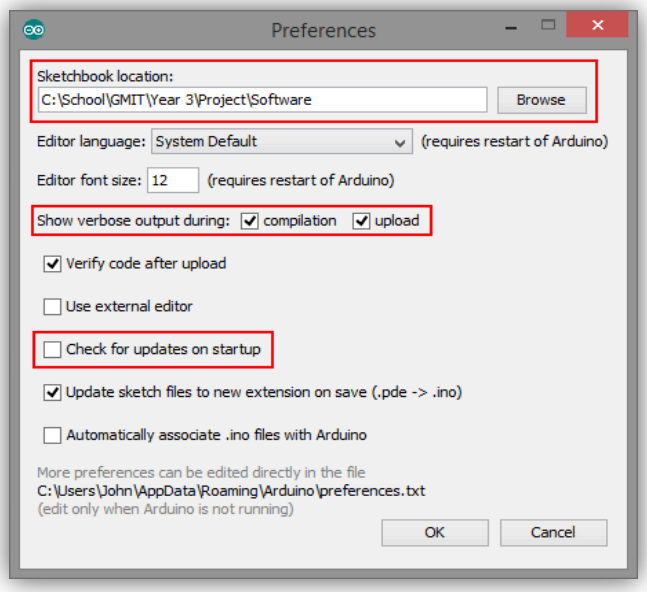
The IDE



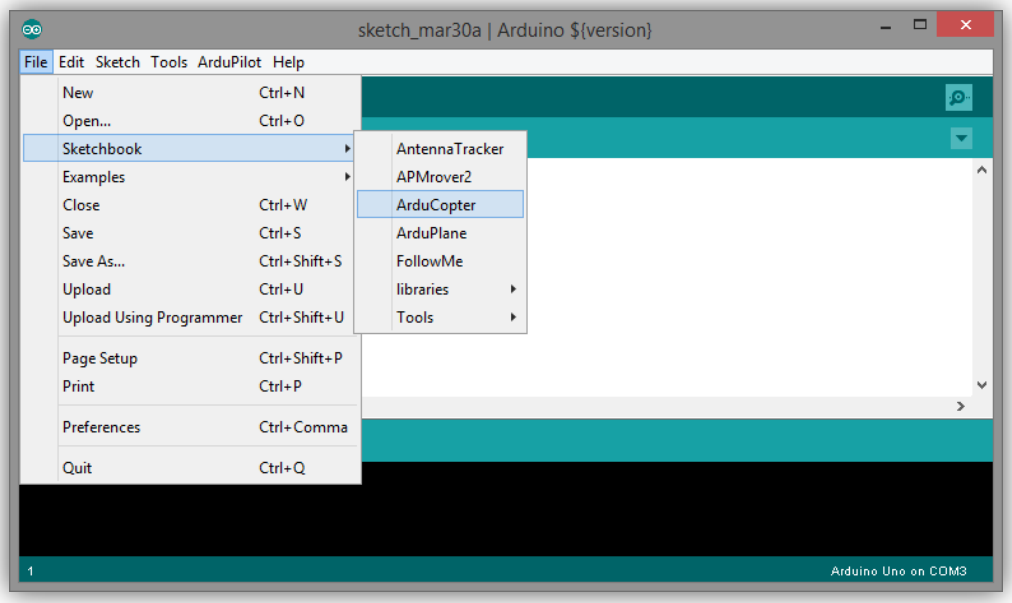
The IDE is version 1.0.3, it's fairly straight forward to use. You must create a new sketch before you can start writing anything. I'll first show how the standard version of the ArduPilot base code is configured, verified and uploaded to the APM board.

To begin, you need to configure the IDE for the ArduPilot, going into 'File' -> 'Preferences' to view the current configuration of the IDE. Change the 'Sketch location' to the location of the ArduPilot code base, get version 3.2.1 from GitHub, (the latest doesn't fit). Then change the 'Show verbose output during' to both being ticked for debugging purposes. And finally un-tick 'Check for updates on startup' to stop checking for updates, because if you were to update the IDE it will break a couple of features that allow this IDE to communicate properly with the APM board.

Preferences Window

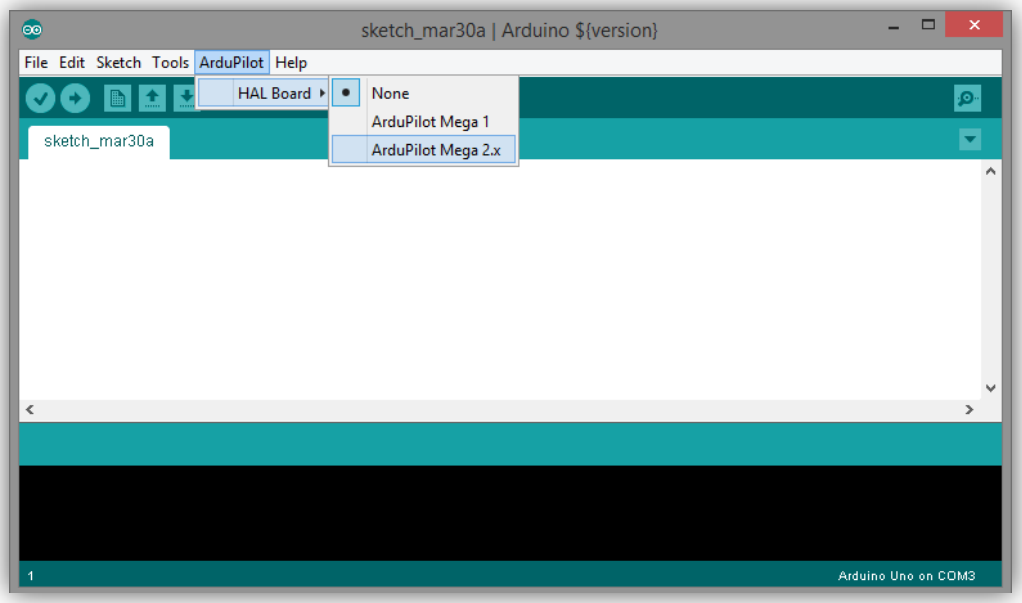


Configure IDE 1



The sketchbook must point to the ArduCopter version of the code base. Different platforms also exist in the repository, versions for helicopters and Rovers for example. After that you need to point the HAL Board settings to the 'ArduPilot 2.x'. To do this go to 'ArduPilot' -> 'HAL Board' -> 'ArduPilot 2.x' and select it, you may receive a warning stating you need restart IDE to apply current settings.

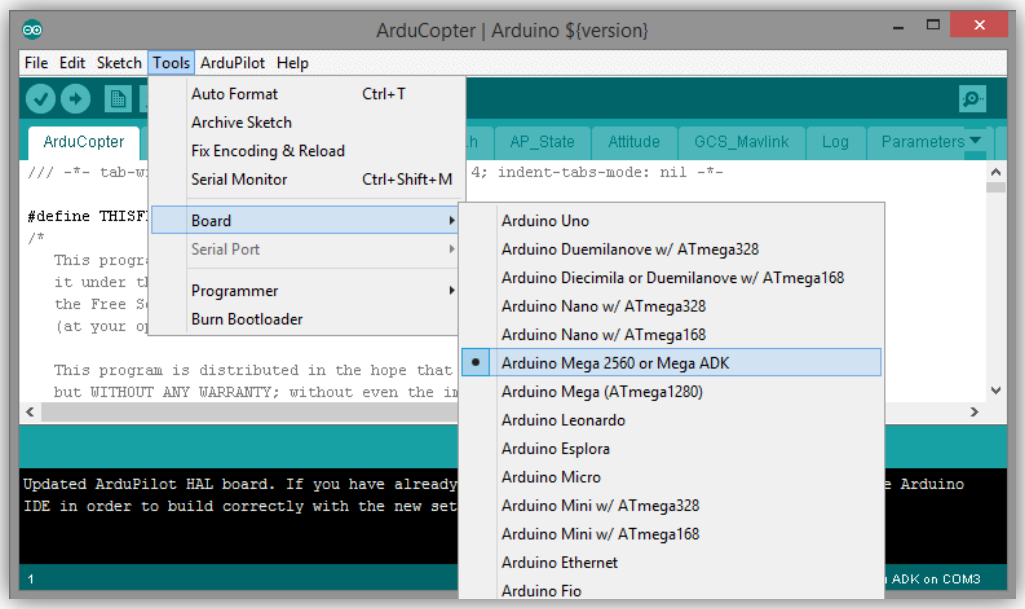
Configure IDE 2



To finish with the configurations, you need to select the correct board version your using to correctly interface with the device.

Go to 'Tools' -> 'Board' -> '(Your board version)'. Now you should be ready to write, build, verify and upload the code to your APM.

Configure IDE 3



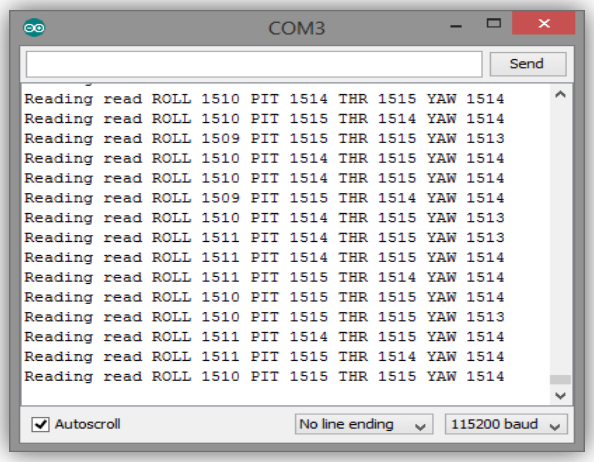
Once the IDE was setup I could start playing with the configurations files of the ArduPilot base code to the way I like them. NextI'll move onto describing the code I've experimented with that'll be running on the ArduCopter board.

My Development

I'll start by describing the code I've written and developed to run on the ArduPilot flight controller. At the beginning, I needed to write a section of code that would read the inputs from the transmitter to get the min and max values of the different channels. By using the serial monitor I could view these values from the console output. I'll include the different source files throughout development, find the files with the prefix Stage 1, Stage 2 etc.

In Stage 1.txt contains the code for reading the inputs from the radio, below is a screenshot from the serial monitor.

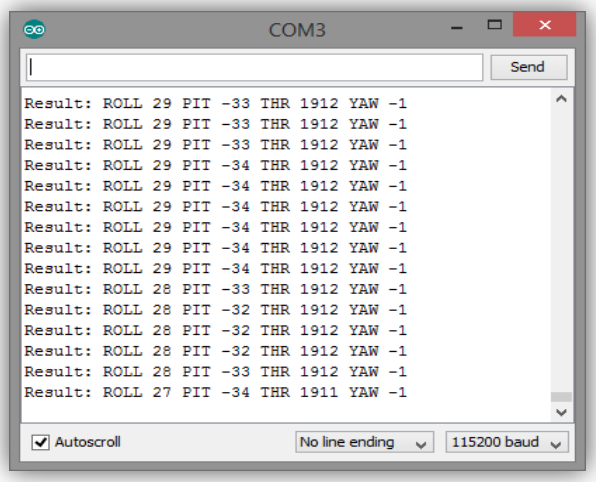
Serial Monitor - Stage 1



The values you see printing out originate from RC channels 1, 2, 3 and 4, channel 5 is also in use but is not implemented in my code, the default ArduCopter code at 700K lines of code uses channel 5 to select different flight modes in real-time. I hope to use this type of functionality to further develop the project later, right now I'm only developing one flight mode to stabilize to the craft while it's flying.

After writing down the min and max values from the different channels I could move onto the next stage of development, which involves converting those values of input from the roll, pitch and yaw channels to readable values in degrees.

Serial Monitor - Stage 2



A function called 'map' converts the values from the radio channels into the required values in degrees, these values are currently only used for testing purposes to display the roll, pitch and yaw values from their respective channels. By taking the current value from the roll, pitch and yaw channels, it outputs a scaled value between the min and max values passed into the function. The Stage 2.txt file contains all the changes from the first stage.

The next step in development would be initializing the motors and applying input from the throttle channel to the motors. The 'setup' function will handle the initialization the motors while a few extra lines in the main loop will write the value from the throttle channel to each motor. I also #defined the motors just under the includes to improve the readability of the source code. The Stage 3.txt contains all of these changes. It's also important to note I haven't included any console prints and delays in the main loop for safety reasons as it could delay the throttle response.

After successful testing to see if the motors were accepting input from the transmitter I moved onto to implementing sensors into the running code to determine the orientation of the quadcopter. The ArduPilot board has a MPU6050 chip that contains the accelerometer and gyroscope that I'll use to get the required data to determine the correct orientation of the quadcopter. The specs of the chip

state it has a feature that fuses both values from accelerometer and gyroscope to get a good accurate value of orientation. It's called DMP (Digital Motion Processing), by using this built-in function I can get accurate values directly from the unit without messing around and compensating for errors like vibrations affecting the accelerometer which can be a major problem, this is why I mounted the flight controller on anti-vibration pads. There are plenty of guides on how to develop your own flight modes using this feature, this is partly why I choose to develop on this board as its quite friendly to developers. It also packs a lot of features usually found in expensive flight controllers like GPS / Compass support and real-time telemetry. For the moment I won't be implementing the GPS / Compass into my program. The Stage 4.txt file contains most of these changes to the source file.

Moving onto the next part of development, I needed to configure the PID values that'll determine how much / fast the quadcopter will need to roll, pitch and yaw to meet the pilots commands. PID is short for proportional-integral-derivative controller, these values will used to determine the rate in which the quadcopter will respond to the pilots requests. If for example I wanted to pitch by 25 Degrees forward, I need to determine how much throttle I'll need to apply to the front and rear props to get the required pitch. By using PIDs I can employ an algorithm to calculate the amount of adjustment required to the front and rear props to achieve the pilots commands.

The file Stage 5.txt contains most of these changes. I'll make sure to fully comment the final working code for you to review.

The final stage of development , from calculating the stabilize PID values I can pass these values directly to the throttle output amount and take it away from the current position of the quadcopter to generate the required throttle to compensate and attempt to level the craft. This is where I've struggled for some time during debugging. It's difficult to debug code with its flying around you, this is why I haven't develop more functions and features into the firmware. I still need to put more time into developing this software. Believe it or not even this basic firmware requires a few minutes to compile. The libraries are written in a number of different languages including assembly, which is quite complex and difficult to understand. I believe this is why it requires a lot of time to compile. Another point to cover is the amount of memory available to the developer. It's about 256,000 bytes of memory, I've already used about 40,000 bytes so can imagine how difficult it can be to fit lots of cool features to the board's flash. It states on the ArduPilot website that the official firmware from version 3.3 and above no longer fits on this APM board, so already this board has been taken to its limit in terms of software. Version ArduCopter 3.2.1 is the last official firmware release for the this board. The last stage of development requires me to mix the values calculated from the three z, y and z axis's to the

motors. I've found this troublesome, I went and found helpful information on various ArduPilot community forums to gather some helpful pointers.

The Stage 6.txt file contains the last few changes to the firmware.

Conclusion

This project from the beginning has been a difficult and time consuming task. From researching the different components to reading up on forums and online messaging boards for help me choose the best and most cost effective flight controller for my project needs. I can honestly say it's quite difficult to develop on devices like the ArduPilot, only because debugging challenging and time consuming, not to mention frustrating. Imagine trying to debug and fly the quadcopter at the same time without crashing it. After reading up online about other experiments, I really have a long way to go before companies like Amazon who are also currently experimenting with drones to deliver package with consider hiring someone like me with basic knowledge of drone software development. I believe this project has been more of a research project then a full blown development project. The amount of time spent on researching the different parts of the quadcopter also took longer than expected, from the various flight controllers, to the available radio setups and power system requirements, it's been overall quite an interesting project to work on. The software development of this project may seem fairly small, only because I ran into more problems than I expected, mainly hardware problems delayed this project. This was an area I not really comfortable with, mainly because I haven't experimented with much electronics in the past, but I still progressed and eventually achieved 90% of my goal which was to build and develop my own quadcopter and to develop my own software to run on flight controller. The software itself still needs a lot of work before I can really say its anywhere near complete or 100% stable.

I'd also like to point out that I wouldn't run my own un-tested firmware out in the open, because if something were to go wrong with the running firmware, I could lose complete control over the craft. It could seriously hurt someone. This is why the standard official firmware at 700K lines of source code, contains many safety features my firmware lacks. I will try and implement some of these features later into my firmware, but because I'm new to this type of development and that time is short on this project, I'm afraid I can't implement those features just yet.

The ArduPilot board really is a versatile device that has great potential in anyone's hands, it's libraries have great documentation online and would recommend anyone whose looking for a platform that's easy to develop and experiment on. I look forward to developing this quadcopter more in the future.

Appendix & Sources

In this section I'll be leaving relevant information relating to my project's development and different sources of information I've found online.

- DiyDrones.com

This site has helped me immensely over the last 2 - 3 months, its contains tons of up to date information on drone development and project based quadcopter builds.

- ArduPilot.com

The home of the flight controller I've used on this project, I've read most the developer pages on this website, it contains a lot of information and has references to the main ArduCopter firmware.

- 3DRobotics.com

The creators of the APM 2.5 / 2.6 hardware, these guys are an industry leader in drone development.

- Blade Hobbies - eBay

I've bought most of the parts off this seller on eBay, the seller is relatively in-expensive.

- AMain.com

A Main contains tons of parts for various different model R/C planes, cars and multi-rotors. I've bought the transmitter used in this project from this site.