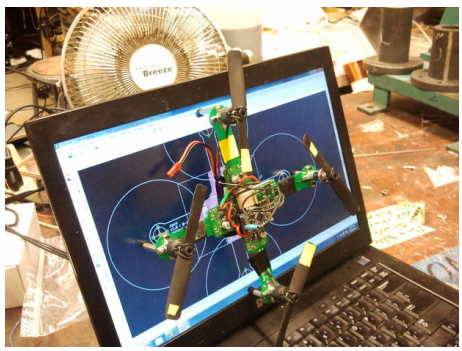
PCB Quadrotor (Brushless)

by scolton (/member/scolton/)

(/contest/makeitreal/) (/contest/rc2012/)

Download (/id/PCB-Quadrotor-Brushless/?download=pdf) (/id/PCB-Quadrotor-Brushless/) 20 Steps

(http://cdn.instructables.com/FK2/KLWQ/H2MYKEB7/FK2KLWQH2MYKEB7.LARGE.jpg)



(http://cdn.instructables.com/FQO/V74R/H2TJ4LV8/FQOV74RH2TJ4LV8.LARGE.jpg)

Quadrotors are the new Segways: a mesmerizing, somewhat magical, self-stabilizing platform that every tech person wants to have. You can't ride a quadrotor (well, maybe you can (http://www.engadget.com/2011/11/01/first-manned-multicopter-takes-flight-brave-human-sits-amidst-b/)), but they do fly, and you can

+ Collection I Made it! Favorite

About This Instructable

152,737 views

486 favorites

License: (cc) BY-NC-SA

scolton (/member/scolton/)

Follow 109

build one y

(http://cdn.instructables.com/FR3/SVM3/H2TIFZSD/FR3SVM3H2TIFZSD.LARGE.jpg)

I helped wit e here
(http://www fter flying it I decided I wanted
to make my one comprising a single printed
circuit boar electronics motherboard. The
end result w I designed it in EAGLE,
soldered it, nths. Here's some video of it
flying, after practice:

More flight video in the final Step!

The idea of making a PCB-based quadrotor isn't unique (see links below for other examples), and 4pcb definitely isn't the smallest (see the Picopter Instructable (http://www.instructables.com/id/Picopter/) for a really tiny one). But I think it strikes a good balance between size, cost, buildability, and flyability. It's also one of the only PCB quadrotors with integrated brushless motor drivers, so there's no need to wire up external ESCs. And it runs on XBee digital radios, so there's no RC receiver or servo-style wiring.

4pcb Specifications:
Size: 6.50" (165mm) motor center-to-center distance, diagonally
Battery: 3S (11.1V), 370mAh, 20-40C Lithium Polymer
Motors: HXM1400-2000 (http://www.hobbyking.com/hobbyking/store/uh_viewItem.asp?idProduct=6312) "hexTronik 5gram Brushless Outrunner 2000kv"
ESCs: Toshiba TB6588FG (http://www.toshiba-components.com/motorcontrol/pdfs/TB6588FG_E_P20_080424_.pdf) "3-Phase Full-Wave PWM Driver for Sensorless DC Motors"
Props: 4x2.5 (2), 4x2.5R (2)
Controller: Arduino Pro Mini 328 - 5V/16MHz (http://www.sparkfun.com/products/11113)
IMU: Pololu MinIMU-9 (http://www.pololu.com/catalog/product/1265)
Radio: XBee Series 1 (http://www.sparkfun.com/products/8665)
Total Weight: 138g
Additional Payload: <30g
Flight Time @138g: 8min

4pcb is a "low level" quadrotor build, by which I mean that there are very few black box components. The frame, motor control, flight control, radio interface, and ground station UI are all developed from component or sub-module level. Depending on your level of experience and interest, you may want to take a different approach where you buy commercial modules for some parts and DIY

(/member/scolton/)

More by scolton

(/id/Segstick)

Tags:

pcb (/tag/type-id/category-technology/keyword-pcb/)

quadrotor (/tag/type-id/category-technology/keyword-quadrotor/)

brushless (/tag/type-id/category-technology/keyword-brushless/)

IMU (/tag/type-id/category-technology/keyword-imu/)

xbee (/tag/type-id/category-technology/keyword-xbee/)

arduino (/tag/type-id/category-technology/keyword-arduino/)

Recommendations

Remote Control Cat Harness

The Ultimate FM Transmitter (Long

Digital Zoetrope

Infinity Mirror Clock

Make Your Own E-Textile Arduino

Bluetooth LE Go-Anywhere Sensor

Circuit Theory Primer

Unwashed Hands Alarm

others. (I included links to some kits and modules below.) This Instructable includes all the files and information you would need to build one completely from scratch.

There are a few small changes I would make if I did a second version of the board, but overall, I think it could make a good standalone project or, even better, a great starting point for your own modifications! (6pcb hexrotor, anyone?) Here are some resources that you might find useful, whether you are building this particular quadrotor or a different multirotor:

Other Quadrotor Instructables:
quadrotor (http://www.instructables.com/id/Quadrotor/) - Custom frame with Arduino-based controller.
RC Quadrotor Helicopter (http://www.instructables.com/id/RC-Quadrotor-Helicopter/) - Off-the-shelf frame with custom controller.
Picopter (http://www.instructables.com/id/Picopter/) - A very tiny custom PCB quadrotor.

Multirotor Frame Kits:
HobbyKing (http://www.hobbyking.com/hobbyking/store/__502__501__Multi_Rotors_Parts-Frames_Kits.html)
ArduCopter (https://store.diydrones.com/category_s/44.htm)

Multirotor Controller Kits:
HobbyKing (http://www.hobbyking.com/hobbyking/store/__504__501__Multi_Rotors_Parts-Electronics.html) (based on KK Multicontroller)
MultiWii (http://www.multiwii.com/)
ArduPilot (https://store.diydrones.com/category_s/1.htm)
OpenPilot CopterControl (http://www.openpilot.org/products/openpilot-coptercontrol-platform/)

Complete Solutions:
Parrot AR.Drone (http://ardrone.parrot.com/parrot-ar-drone/usa/) - Very stable iPhone-controlled quad.

General Multirotor Resources:
RCGroups Miniature Quadrotor Thread (http://www.rcgroups.com/forums/showthread.php?t=1335765)
DIY Drones Quadcopter Forum (http://www.diydrones.com/forum/categories/quadcopters-1/listForCategory)
OpenPilot Multirotor Forum (http://forums.openpilot.org/forum/26-multirotors/)
PID Tuning for Multirotor (http://vimeo.com/25839978) (OpenPilot TV)

Mini Quads (5" props)
TinyCopter (http://www.etotheipiplusone.net/?cat=87) (custom build)
BabyCopter (http://t3chnolochic.blogspot.com/2012/06/never-build-quadrotor-for-6115.html) (custom build)
Turnigy Integrated PCB Micro-Quad (KIT) (http://www.hobbyking.com/hobbyking/store/__22607__Turnigy_Integrated_PCB_Micro_Quad_KIT_.html) (commercial)
Blade mQX (http://thevariableconstant.blogspot.com/search/label/kawaiicopter) (commercial)

Micro Quads (4" props):
4pcb (http://scolton.blogspot.com/p/flying-things.html#4pcb) (custom build)
Kawaiicopter (http://thevariableconstant.blogspot.com/search/label/kawaiicopter) (custom build)

Nano Quads (3" props):

Nanocopter (http://forums.openpilot.org/topic/8831-nanocopter/) (custom build)
Nano quadcopter wii (http://www.youtube.com/watch?v=AZ7Xt2b61Us) (custom build)

Pico Quads (2" props):
Walkera QR Ladybird (http://www.helipal.com/walkera-qr-lady-bird-2-4ghz-black-value-edition.html?gclid=CL2o7IfEprACFdMAQAodtBP_xg) (commercial)
CrazyCopter (http://www.daedalus.nu/category/crazycopter/) (custom build)
Picopter (http://www.instructables.com/id/Picopter/) (custom build)
Chibicopter (http://www.etotheipiplusone.net/?cat=89) (custom build)
NC-ONE (http://www.nanokopter.at/tiki_v6/Entstehung) (custom build)

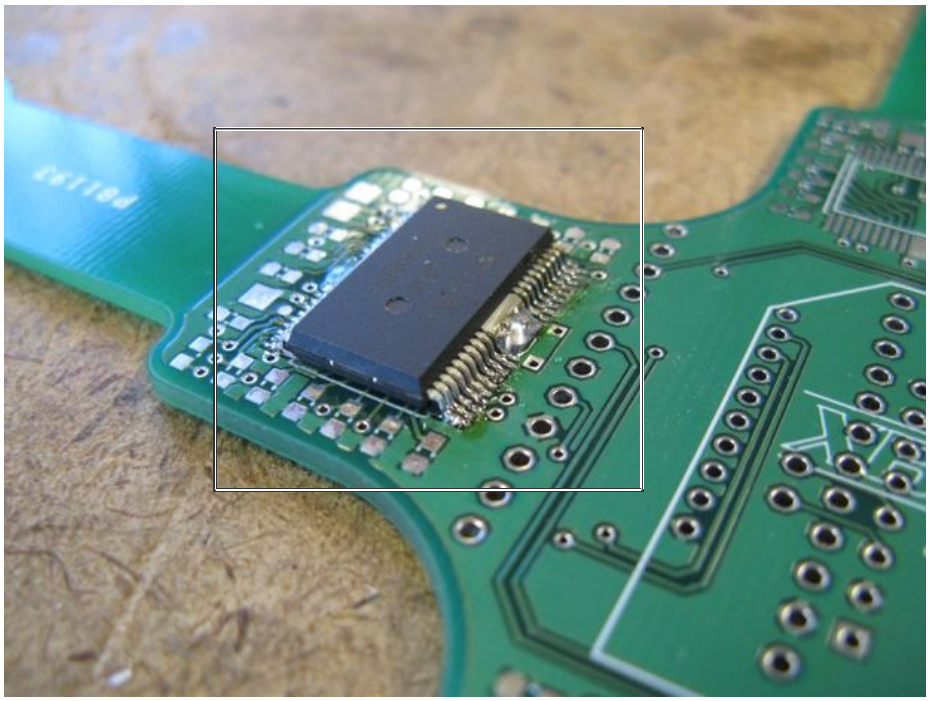
My Pages:
4pcb and other Flying Things (http://scolton.blogspot.com/p/flying-things.html)
The Balance Filter (http://web.mit.edu/scolton/www/filter.pdf) - Merging accelerometer and gyro signals.

Pcb Assembly Suppliers
alibaba.com/Pcb-Assembly
Top Deals at Factory Price. Contact Directly & Get Live Quotes!

Robot DC Motor Driver Kit

Remove these ads by **Signing Up** (/account/gopro?sourcea=removeads&nxtPgName=PCB+Quadrotor+%28Brushless%29&nxtPg=/id/PCB-Quadrotor-Brushless/?ALLSTEPS)

Step 1: The Setup: Parts, Tools, Software, and Files



(http://cdn.instructables.com/FTU/EN89/H2TJ4L3W/FTUEN89H2TJ4L3W.LARGE.jpg)

The attached file (4pcb_DOC.zip) contains all of the support files for making and flying 4pcb. Included in the zip file are:

4pcb_ARD (folder) - Arduino project (Arduino 0022, .pde).
4pcb_EAG (folder) - EAGLE board files and libraries (EAGLE 6.0.0 Light Edition).
4pcb_EXE (folder) - Ground station executable (requires .NET Framework 2.0 or later).
4pcb_GRB (folder) - Gerber files for PCB printing.
4pcb_VB (folder) - Ground station source (Visual Basic Express 2008 or later).
4pcb_BOM.xlsx - Bill of material in Excel format.
4pcb_DIR.jpg - Coordinate system of quadrotor and IMU.
4pcb_EXT.pdf - Details of external connections.
4pcb_IMU.jpg - Image showing vibration mounting and wiring of Pololu minIMU-9.
4pcb_SCH.pdf - PDF schematic of the board.

Bill of Materials / Cost:

The Bill of Materials (4pcb_BOM.xlsx) lists all the components required to put together one PCB quadrotor and ground station. The total cost to build the quadrotor is about **$240**. The ground station consists of a USB game controller, an XBee radio, and and XBee-to-USB adapter. If you don't already have these, they add an aditional **$80** or so.

Soldering:

This board requires a good amount of surface-mount soldering, including passives as small as 0603 and four TSOP36 ICs. They can all be hand-soldered (no BGA or leadless).

Additional Tools and Hardware:

- Wire (22AWG and 28AWG stranded would work) and wire cutters/strippers.
- Solder braid for cleaning up bridges.
- FTDI cable (http://www.sparkfun.com/products/9718) for programming the Arduino Pro Mini.
- Hex key set.
- Double-sided foam mounting tape.

Software:

-EAGLE
If you want to modify the printed circuit board, you'll need EAGLE v6.0.0 or later. You can download it here (http://www.cadsoftusa.com/download-eagle/?language=en). The free "Light Edition" is sufficient, even though the outline of the board is larger than 100x80mm limit (see Step 2). You will also need EAGLE to reference the board layout when placing components. (e.g. Type "show R32" in the board window command line to figure out where to put resistor R32.) There are no designators on the board itself.

-Arduino
The flight controller is written in the Arduino IDE. You can download the latest version from here (http://arduino.cc/en/Main/Software). Make sure you set the board type to "Arudino Pro Mini (5V/16MHz) w/ ATmega328".

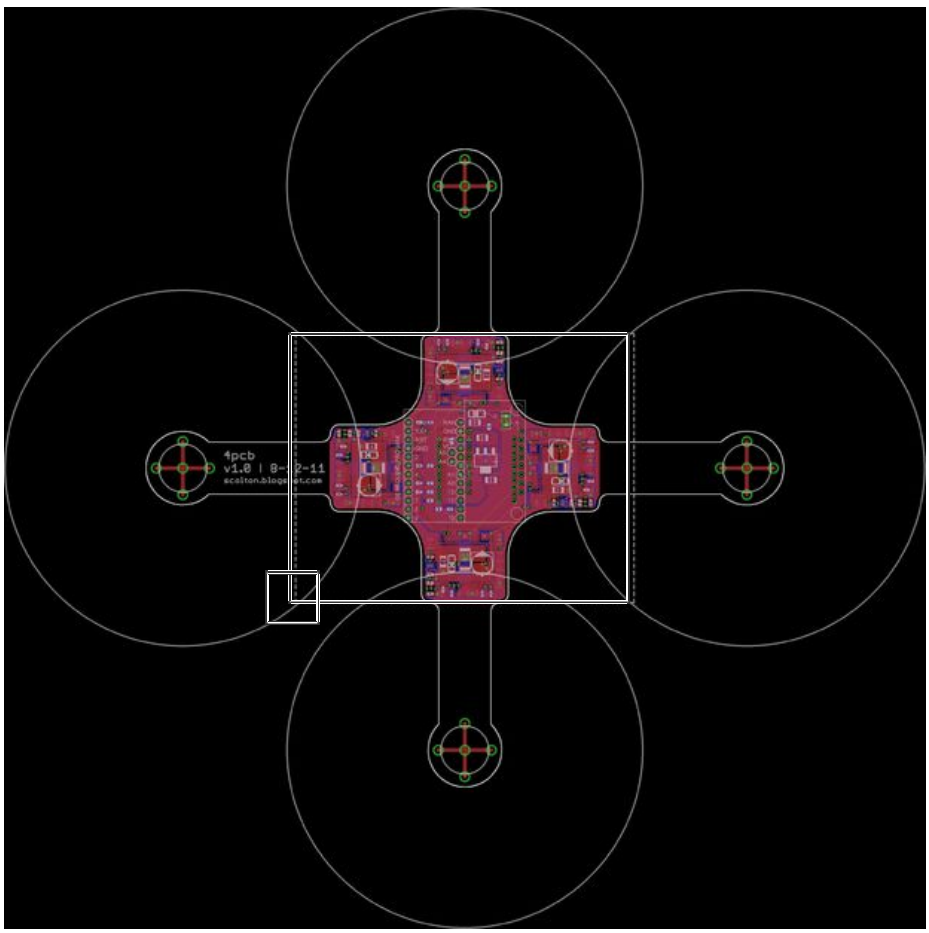
-Visual Basic Express (Optional)
The ground station is programmed in Visual Basic Express. If you want to modify the ground station software, you can download the free edition, Visual Basic Express 2010 from here (http://www.microsoft.com/visualstudio/en-us/products/2010-editions/visual-basic-express).

-.NET Framework
The ground station requires the .NET Framework runtime files. (Unfortunately, this makes it Windows-only.) These files come with Visual Basic 2010, so if you plan on modifying the ground station software, there's no need to download them separately. If you just want to run the ground station executable, you can download the .NET Framework runtime files from here (http://www.microsoft.com/en-us/download/details.aspx?id=21).

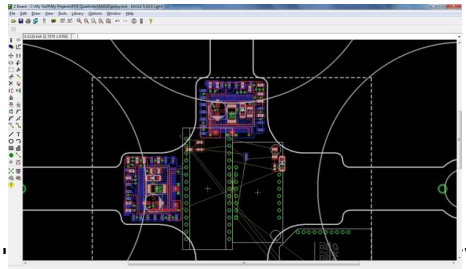
-Processing? (Optional)
Although I haven't done so myself, it is possible to port the ground station software over to Processing (http://www.processing.org/), which would make it compatible with other operating systems. To read from the USB game controller, there is a third-party library called ProCONTROLL (http://creativecomputing.cc/p5libs/procontroll/). I did some work with this for a XBee-based robot controller, the details of which are here (https://sites.google.com/site/2007arduino/example-code/controller-tutorials/usb-ps2-controller). This could be a good starting point for making a non-Windows ground station.


4pcb_DOC.zip (/files/orig/FAS/DOVV/H2WEJCSG/FASDOVVH2WEJCSG.zip)1 MB

Step 2: Board Design

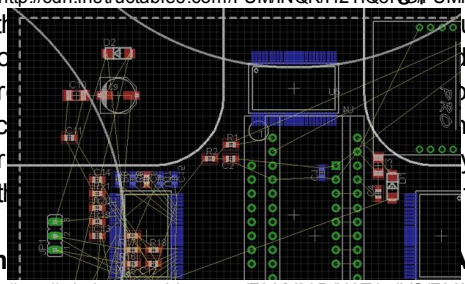


(http://cdn.instructables.com/FYF/OSAA/H2TJ4JA7/FYFOSAAH2TJ4JA7.LARGE.jpg)



(http://cdn.instructables.com/FUMNQKVH2TQ8NGA/FUMNQKVH2TQ8NGA.LARGE.jpg)

motherboard. The second most time-consuming part of the project was probably designing t uming part is programming the flight contro the Gerber files required to manufactur oject documentation files (Step 1). In that c talks about board manufactur yone who is interested in modifying t files and go for it!



How to Ch A Little):

(http://cdn.instructables.com/F26/V29D/H2TJ4JVQ/F26V29DH2TJ4JVQ.LARGE.jpg)

4pcb is quite a bit larger than the 100x80mm board size limit for the free EAGLE Light Edition. One reason I put off starting this project was because I didn't want to pay for the full version or switch to a different board layout software. But then I discovered something interesting: The board size limit applies only the placement of components. You can route traces, add holes, and create a board outline that extend far outside the 100x80mm box.

4pcb's centroid is right at the middle of the 100x80mm bounding box. All the components fit in the center part of the board, inside this restricted area. The arms, including the motor mounting holes, stretch out in all directions including going into negative coordinates. This is fine - the Gerber files still seem to render properly.

Board Outline:

4pcb's board outline ("Dimension" layer in EAGLE) is actually the shape of the quadrotor frame. Most board printing companies will route the outside dimension of your board to any shape you specify. (Make sure this is an option before you order.) Drawing the shape took some practice using the arc tools in EAGLE. One idea that's been suggested is to add diagonal supports between the arms. However, this would create areas of internal routing, and most board companies charge extra for that.

Rotational Symmetry and Copy-Paste:

The four motor controller blocks are rotationally symmetric. I routed this board in EAGLE 5.11, when Copy-Paste of component groups was more difficult. I think it's much more straightforward in EAGLE 6.0.0 and later. But it definitely still saved a lot of routing time to be able to make four identical copies of the motor controller block. **If you're using an older version of EAGLE** (prior to 6.0.0), the rough process for Copy-Paste is as follows:

1. Create and route the component block to be copied. Do this first, before any non-copied components are placed or routed. This will help auto-numbering not break.

2. Close the schematic. This breaks the back-annotation check so that you can Copy-Paste on the board itself.

3. Select the group and perform a group copy (Ctrl + Right Click). Paste, move, and rotate as desired.

4. Re-open the schematic. Ignore all the errors. Copy-paste the schematic group

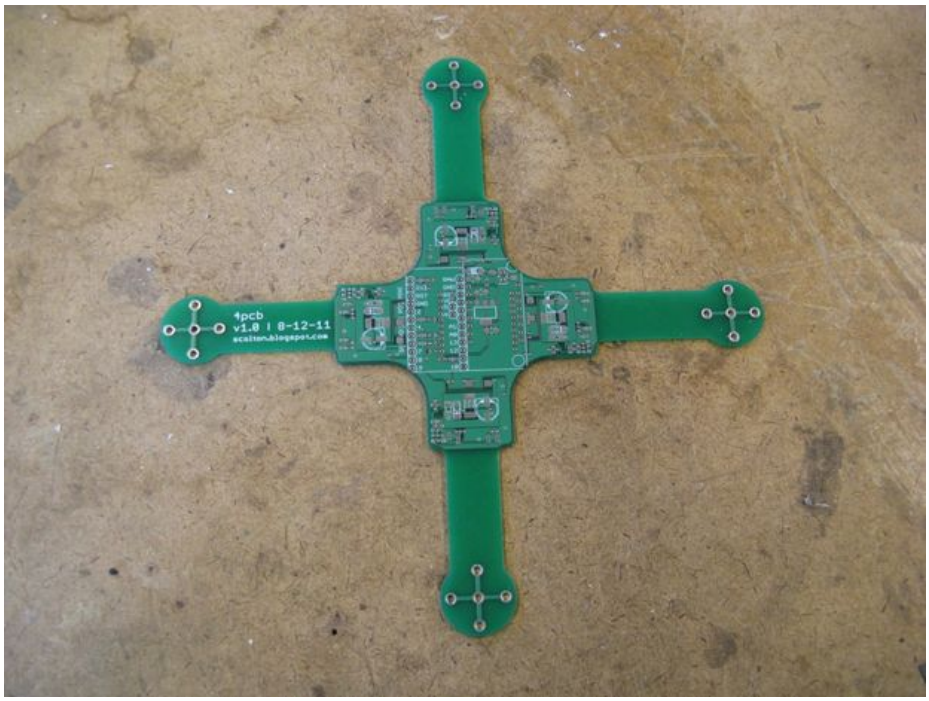
containing **exactly** the same components and nets as the board group.

5. The auto-numbering should work, but if it doesn't you may have to clean up some designators manually to make them match the board copy.

6. Repeat.

The process should be much simpler in EAGLE 6.0 or later, and shouldn't require breaking the back-annotation. I haven't gotten around to trying it yet, though.

Step 3: Board Printing



(http://cdn.instructables.com/FCG/YMK5/H2TJ4L9W/FCGYMK5H2TJ4L9W.LARGE.jpg)

Make it real.

Below is a list of just a few board printing companies. They should all take the standard set of Gerber files. For a two layer board, there should be up to seven files. The Gerber files for 4pcb are included in the project documentation files (Step 1).

Layer Definitions:

Top Silkscreen (pcbqr.GTO)
Top Solder Mask (pcbqr.GTS)
Top Copper (pcbqr.GTL)

NC Drill (pcbqr.TXT)

Bottom Copper (pcbqr.GBL)
Bottom Solder Mask (pcbqr.GBS)
Bottom Silkscreen (pcbqr.GBO)

You can probably live without the silkscreen, since the component density is too

high for designators to be useful anyway. (You'll have to reference the EAGLE files to place all the parts.) The soldermask is absolutely necessary for this board, though. A bare board will be too difficult to solder.

Board Printing Services:

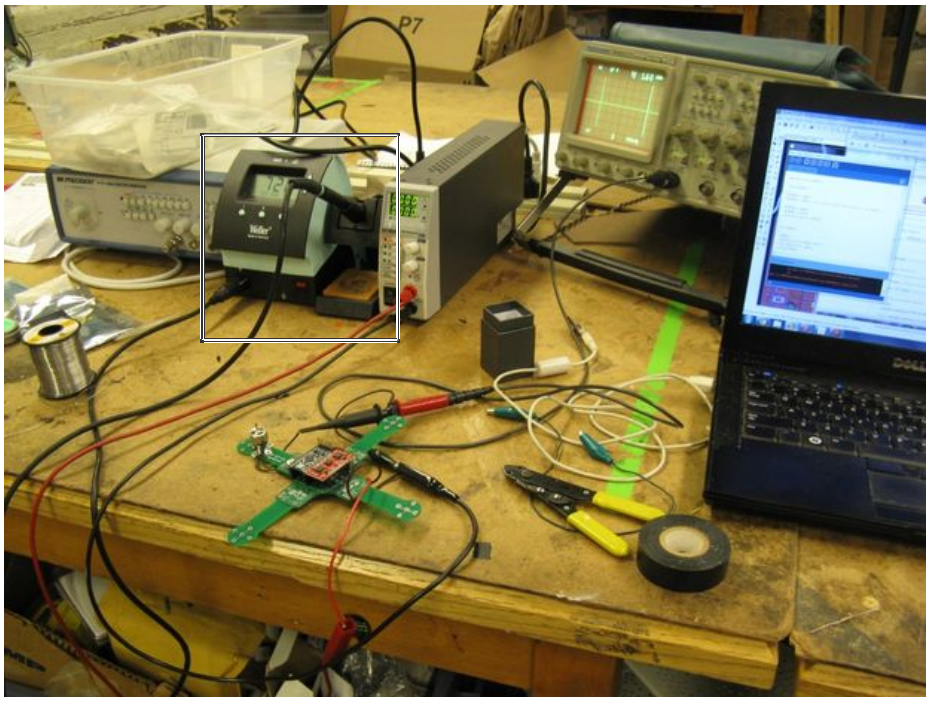
-Advanced Circuits (www.4pcb.com (http://www.4pcb.com/))
Fast and reliable (US-based), but expensive shipping. If you are a student, use the 33each discount to get a single board for $33 + shipping. This is where I got mine printed. And yes, the name "4pcb" comes from this site's URL, though they are not an official project sponsor or anything.

-MyRO PCB (www.myropcb.com (http://www.myropcb.com/))
Inexpensive and usually fast.

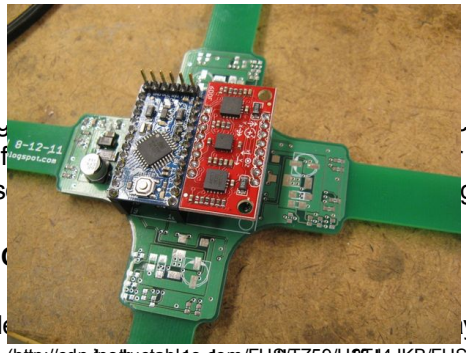
-Dorkbot PDX (www.dorkbotpdx.org/wiki/pcb_order (http://dorkbotpdx.org/wiki/pcb_order) > now www.oshpark.com (http://oshpark.com/)?)
Group order, so it could be slower, but good pricing.

There are many others. Shop around. Just make sure that external routing to non-rectangular shapes and soldermask on both sides are standard options.

Step 4: Solder Lots of Tiny Parts



(http://cdn.instructables.com/FOG/XDV7/H2TIBAOB/FOGXDV7H2TIBAOB.LARGE.jpg)

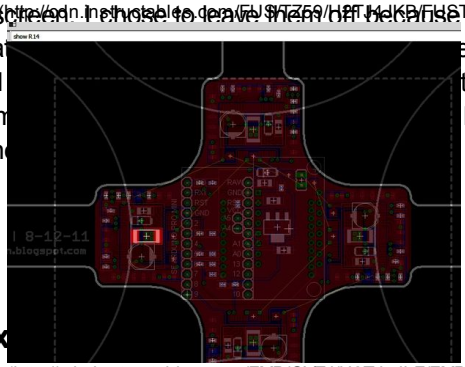


Depending[illegible]ry relaxing or very boring. Definitely f[illegible] if you are more into hot-plate or reflow s[illegible]gs to solder everything.

Locating C[illegible]

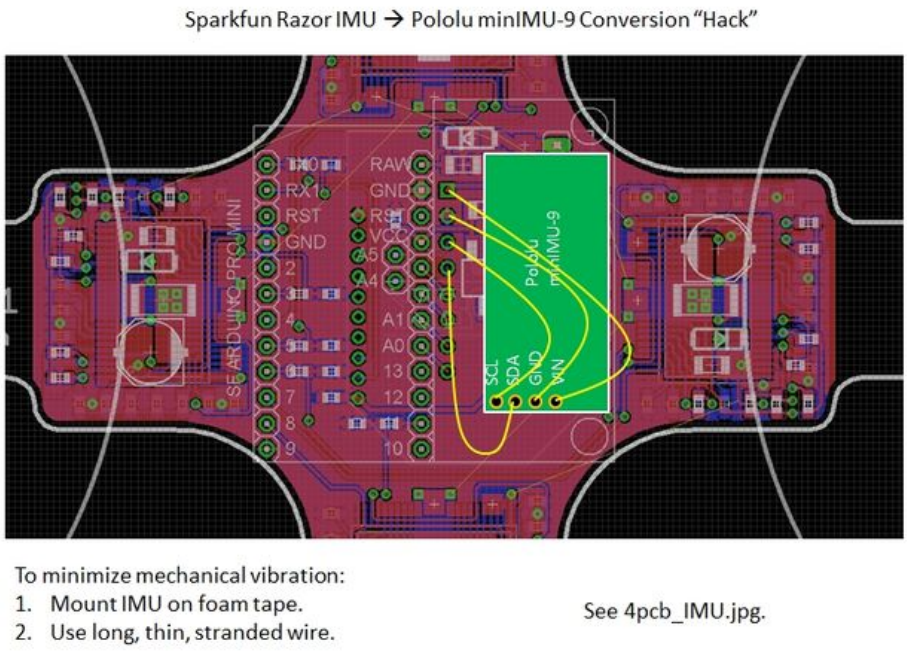
The part de[illegible]yer in EAGLE are not rendered on the silkscreen. I chose to leave them off because the component density is high enough tha[illegible]e consuming process. Instead, the method [illegible]the EAGLE board open on a nearby com[illegible] highlight each component as I go. (See th[illegible]

(http://cdn.instructables.com/FUSRTZ59/H2TJ4JKP/FUSRTZ59H2TJ4JKP.LARGE.jpg)

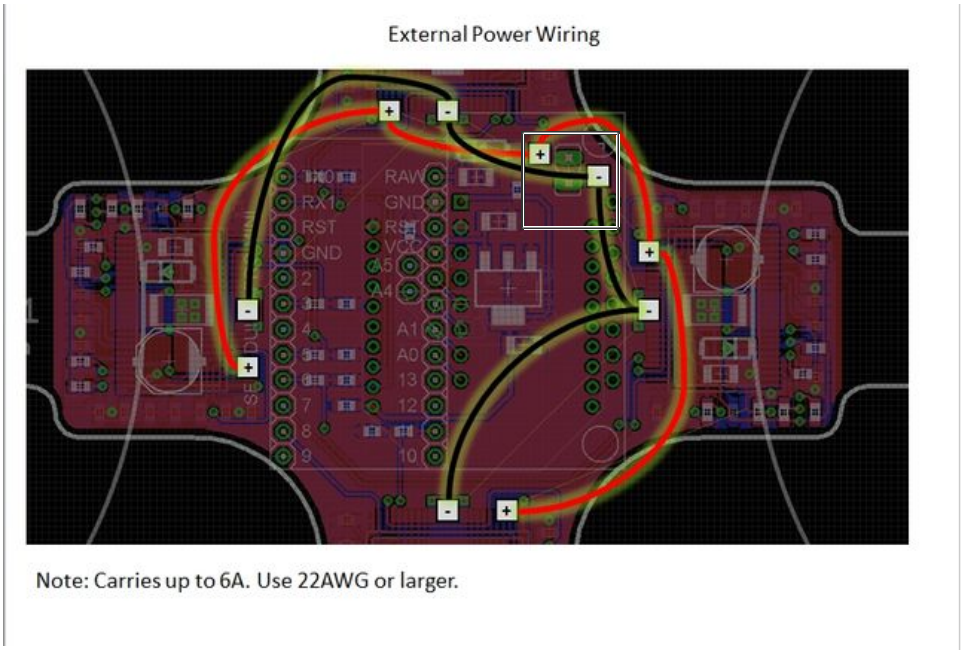


(http://cdn.instructables.com/FMP/GVTJ/H2TJ4JLF/FMPGVTJH2TJ4JLF.LARGE.jpg)

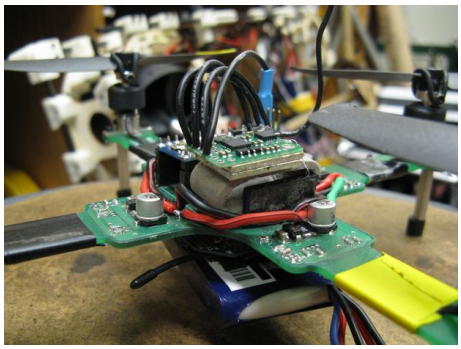
Step 5: Ex[illegible]



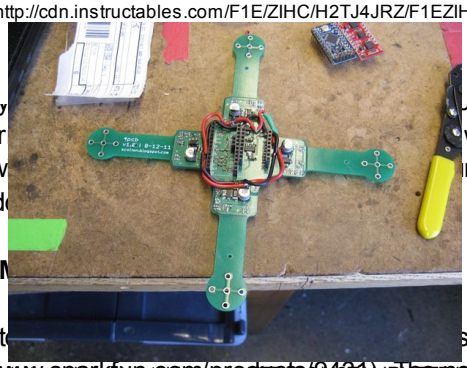
(http://cdn.instructables.com/FU4/SR0M/H2TIFWU5/FU4SR0MH2TIFWU5.LARGE.jpg)



(http://cdn.instructables.com/FSO/KEJV/H2TIYE8I/FSOKEJVH2TIYE8I.LARGE.jpg)



(http://cdn.instructables.com/F1E/ZIHC/H2TJ4JRZ/F1EZIHCH2TJ4JRZ.LARGE.jpg)



(http://cdn.instructables.com/FSO/KEJV/H2TIYE8I/FSOKEJVH2TIYE8I.LARGE.jpg)

I nailed in my [illegible] motor circuit to a single PCB with no external wires to connect, 12 wires in total, each wire summarized in 4pcb_EXT.pdf in the project download.

Pololu minIMU-9

This quadrotor uses the obsolete Sparkfun 6DOF Razor IMU (http://www.sparkfun.com/products/9431). The new design uses the Pololu minIMU-9 (http://www.pololu.com/catalog/product/1265). The minIMU-9 is a digital IMU, which has some nice benefits (higher resolution ADC than the Arduino, self-zeroing). It's also smaller than the Razor IMU was and only requires four connections to communicate with the Arduino Pro Mini over I2C serial.

I didn't modify the board to have a minIMU footprint because I found that the IMU has to be mounted on foam tape anyway in order to minimize the effects of mechanical vibration transmitted through the frame. (The vibration really messes with the gyros, in ways that can't even be filtered out.) I found that making a loop of foam tape works even better than one or two layers alone. Also, I found that adding mass to the IMU, by taping small pieces of aluminum to the bottom, helped. (See the third image.)

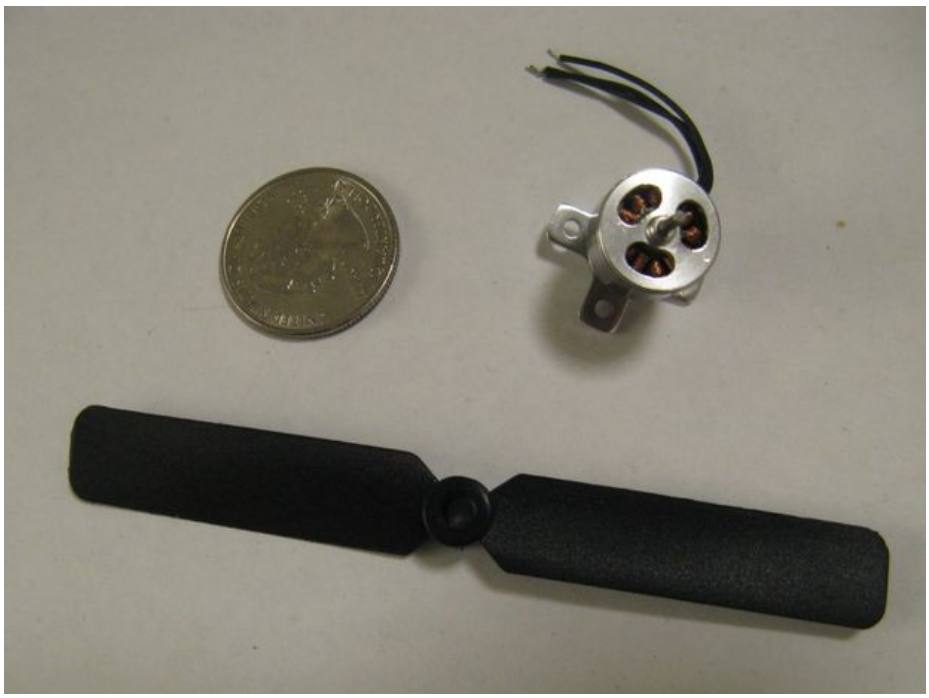
After mounting the IMU to the frame, the four connections should be made with thin, flexible external wires (also to minimize vibration). Conveniently, the connectors can go right to the first four pins of the now-unused Sparkfun IMU header. (See first image.)

External Power Wiring:

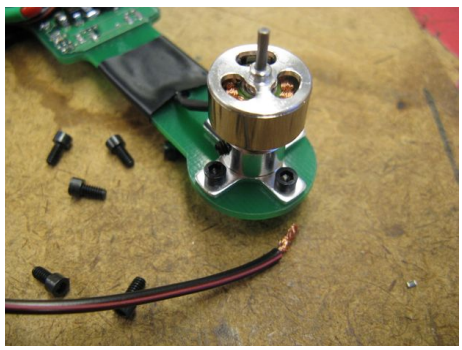
The power wiring carries high current, so I thought it would be a good idea to keep it off-board. In retrospect, I wish I had put the external power wiring on the board. This is something I would definitely modify in v2. There is enough room to route a few large traces around to carry power and ground to the four motor controllers, especially with the Sparkfun IMU headers out of the way.

For now, though, the power wiring is done externally with 22AWG or larger, preferably stranded wire. There are a total of 8 wire jumps to make, four for power and four for ground. (See the second image.) To eliminate ground loops, a star configuration from the battery power input to each motor controller could be used. I found that a bus works okay, though. (Keep in mind that in a bus, the wires feeding out from the main power input carry the most current.) **Be very careful not to reverse any of these connections, or it will instantly destroy the motor controller on power-up.**

Step 6: Motors!



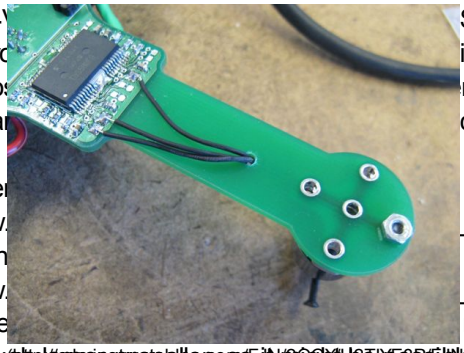
(http://cdn.instructables.com/FC7/YXYB/H2TIUAGM/FC7YXYBH2TIUAGM.LARGE.jpg)



(http://cdn.instructables.com/FLP/3QR1/H2TIUAGN/FLP3QR1H2TIUAGN.LARGE.jpg)

This PCB quad uses four tiny brushless motors for propulsion. They're the HXM1400-2000 (http://www.hobbyking.com/hobbyking/store/uh_viewItem.asp?idProduct=6312) "hexTronik 5gram Brushless Outrunner 2000kv". They're designed

for 2S (7.4V[illegible]S battery. However, the minimum vo[illegible]iver chip is 7V, which is cutting it a little clos[illegible]ry, even one that it slightly heavier than[illegible]cause the motors to overheat.



The 10g ver[illegible]
(http://www.[illegible]__18_11_2000kv_Micro_Brushless_Outrun[illegible]
(http://www.[illegible]__hexTronik_20gram_Brushless_Outrunne[illegible] mounting pattern and can swing a much larger propeller, so it might be possible to make a larger version just by extending the arms. Stiffness would be a concern at that point. The 2g version (http://www.hobbyking.com/hobbyking/store/__7230__hexTronik_2_gram_Brushless_Outrunner_7700kv.html) might also fit the same mount, for an even smaller quad, but its rpm/V is too high to run effectively on 3S and it's speed might exceed the RPM limit of the Toshiba TB6588FG.

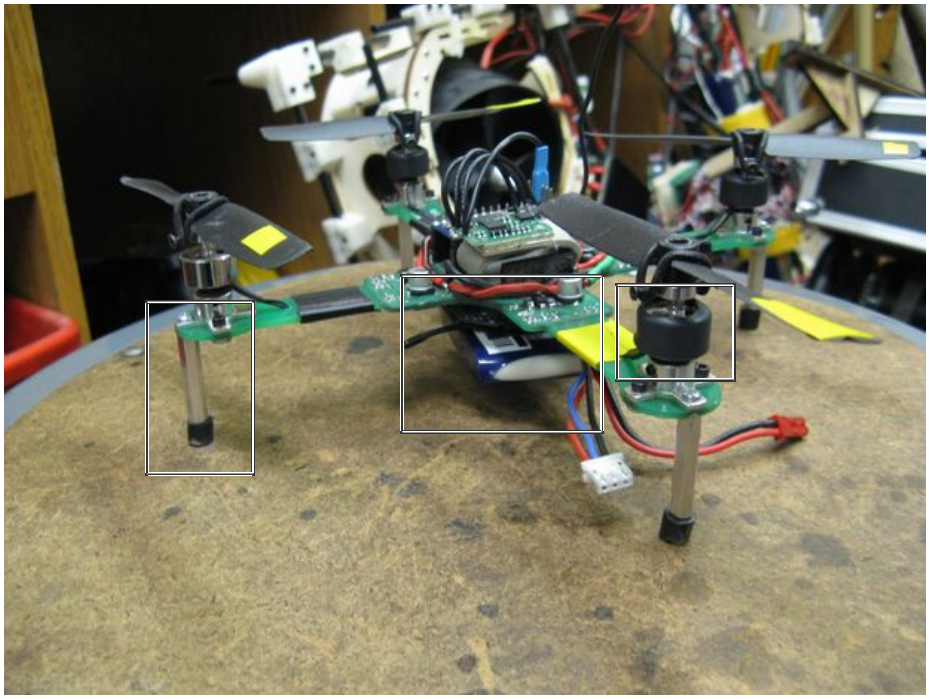
Mounting:

The motors bolt directly to the PCB with short 2-56 machine screws. The wires should be facing inward and the hole directly under the wires may be inaccessible. That's okay: three screws will do. The two side screws should get 2-56 nuts. The outermost screw will be used to attach the landing gear in the next Step, so it doesn't need a nut. **Use threadlocker (Loctite)** on all the screws otherwise vibrations will cause them to loosen.

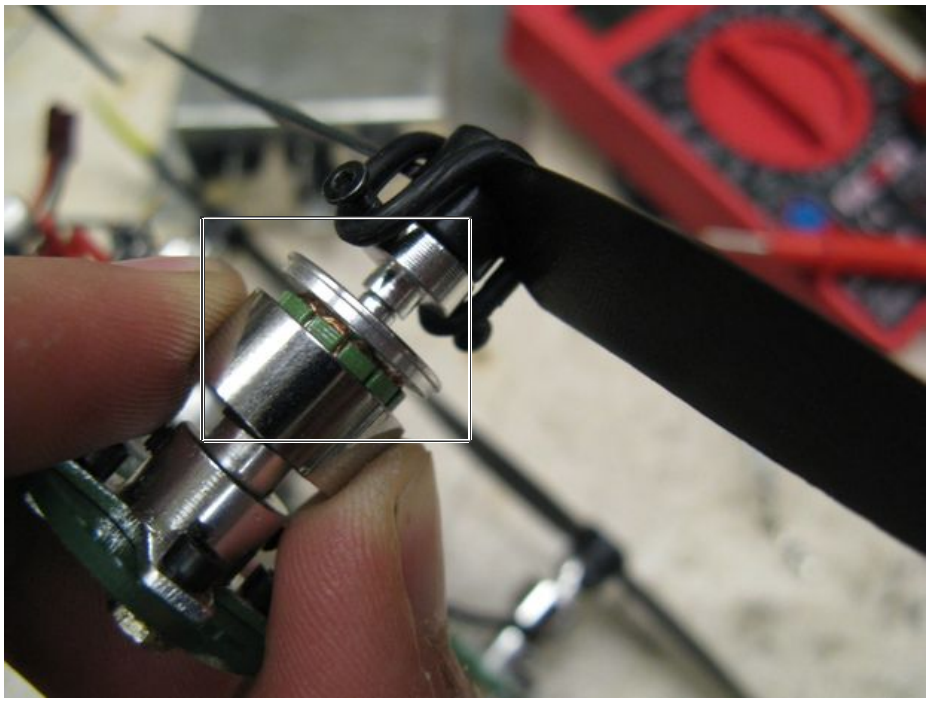
Wiring:

The motor wires are pre-tinned and can be soldered directly to the three output pads of the motor controller component block, which are on the bottom layer of the board. Sometimes, the wires come long enough to reach around the arms. Other times, they are too short. You can either make wire extensions or drill a hole in the arm for a more direct line to the pads. (See the third image.)

Step 7: Landing Gear and Motor Shock Proofing



(http://cdn.instructables.com/FBI/LT72/H2ULR3CX/FBILT72H2ULR3CX.LARGE.jpg)



(http://cdn.instructables.com/FKU/QLI1/H2ULR3FV/FKUQLI1H2ULR3FV.LARGE.jpg)

Building a quad means lots of crash landings.

Small quadrotors are nice because they can survive crashes pretty well. There are a few weak links, though, and this step tries to address those.

Landing Gear:

The landing gear I chose were 1"-long aluminum 2-56 standoffs. I originally tried plastic ones, but they broke off on tough landings. The aluminum ones are much more durable, for not very much weight penalty. They attach directly to the outermost motor mounting screw (use threadlocker). To cushion the landings, I added four small silicone feet to the landing gear. Without these, the shock from the landings is transmitted directly into the motor, which can sometimes case its rotor press fit to fail (see below).

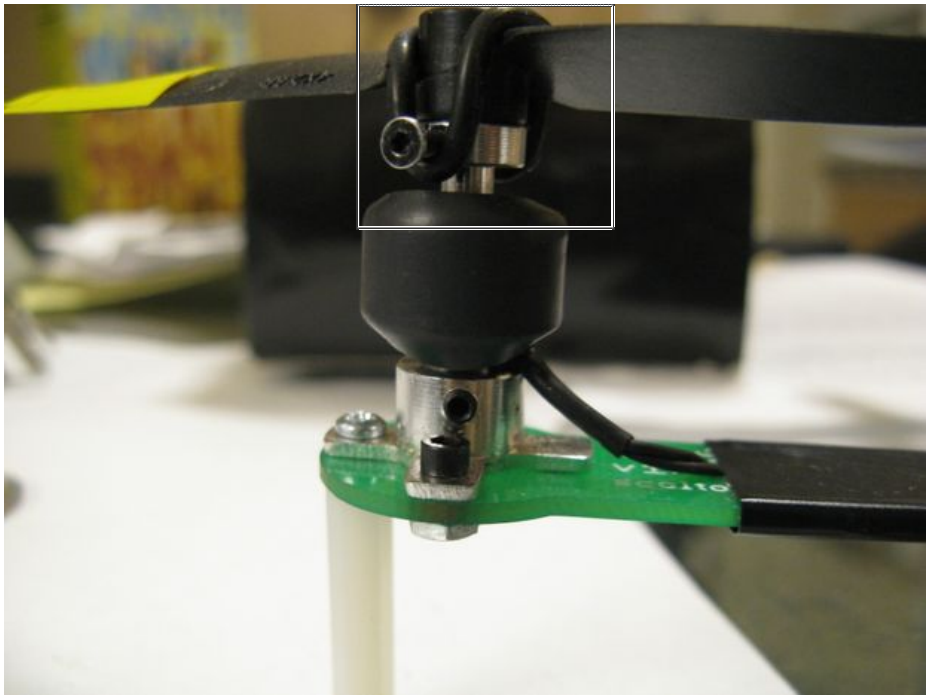
Shock-Proofing the Motors:

By far the most common failure mode I found during crash landings was the motor cans coming apart. (See the second image.) The shock from a crash would cause the narrow interference fit to fail, and no amount of superglue or Loctite would keep it together. The problem with adhesives like that is that the joint is brittle, so the shock just cracks the adhesive off.

I found a much better solution was to actually heat shrink the entire motor. You can see this in the first image: three of the four motors are heat-shrinked. This probably reduces motor cooling a bit, but the compliant heat shrink does a much better job of holding the rotor together under the shock loads of a crash landing. Just be careful to cut and shrink the tubing evenly, or it will throw off the balance of the rotor a lot.

Step 8: Props and Prop Balancing

(http://cdn.instructables.com/FXM/IA0C/H2TJ4OIY/FXMIA0CH2TJ4OIY.LARGE.jpg)



(http://cdn.instructables.com/FD5/WZ5D/H2TJ4OJ1/FD5WZ5DH2TJ4OJ1.LARGE.jpg)

4pcb uses four 4x2.5 propellers, two standard and two counter-rotating. (It needs the counter-rotating props to control the yaw.) You can buy them in sets of six here (http://www.hobbyking.com/hobbyking/store/__11327___4x2_5_Propellers_Standard_and_Counter_Rotating_6pc_.html). Buy plenty of spares.

The props are inexpensive, but they're also poorly balanced. Other than mounting the IMU on foam tape, balancing the props was the single most important thing I did to reduce vibrations. You can feel the effect of vibrations on the frame, but to really see how large they are I filmed one arm of the quadrotor under a strobe light set to flash slightly faster or slightly slower than the prop speed:

If that doesn't convince you of the importance of balancing the props, I don't know what will. You can buy prop balancing jigs, but I think they're mostly useless for small props and I wouldn't waste the money.

For a rough pass at balancing the props, you can place them on any small, smooth shaft and see which way they tend to fall. If they get "stuck", tap the shaft a little or spin the prop with your fingers. If they have a tendency to fall one way, add small pieces of electrical tape to the opposite side until they seem to have no preference. I use yellow electrical tape to add some visibility to the edges of the props as well.

For more precise balancing, you can use a 2mm shaft and two prop adapters to

make a simple gravity-based balancing jig like the one shown in the first image. The bubble levels are optional - a machine vise or any other reasonably flat and smooth surfaces could be used instead. I found this to be far more precise than commercial jigs.

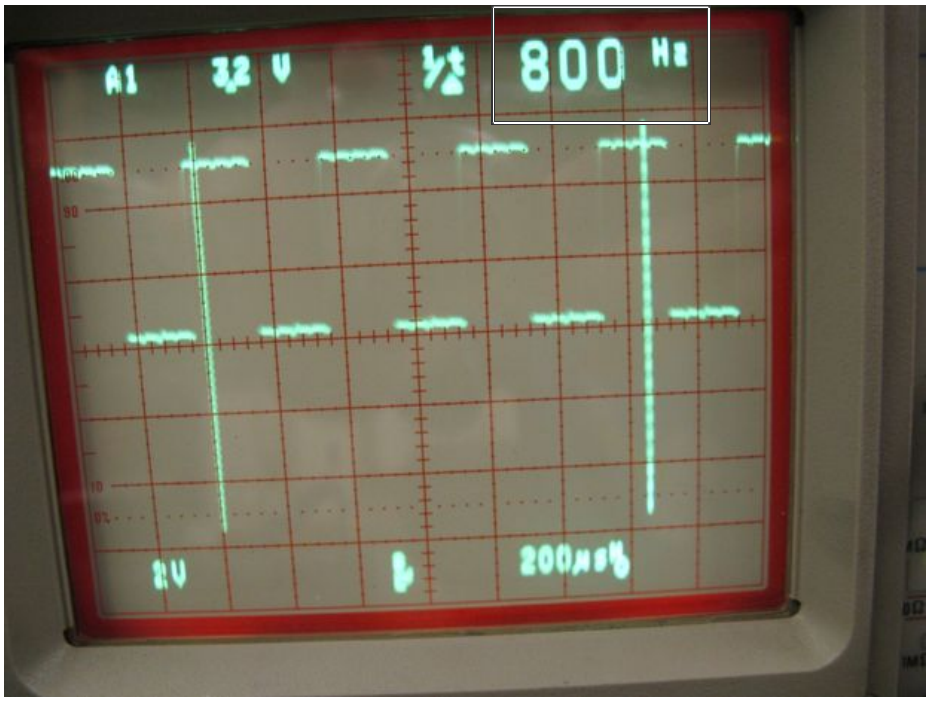
For some reason, the counter-rotating props (4x2.5R) always seem to require more balancing. I've sometimes even had to use two layers of tape!

Prop Adapters and Prop Savers:

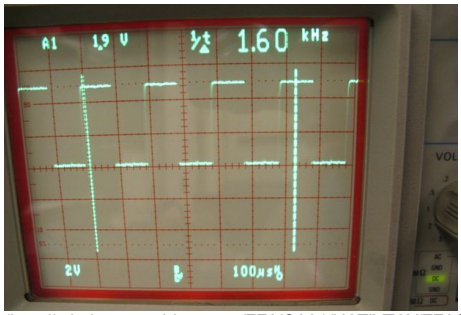
The motors come with prop adapters, small hubs that reduce the bore of the prop down to 2mm so it is a nice fit on the motor shaft. They also come with prop savers, which are just o-rings. The prop savers are designed to absorb the impact of a prop strike, where the propeller hits the ground or a fixed object. This way, the propeller itself is less likely to break. They work fairly well on small props. I've crashed many times and only break a prop maybe 1 out of 10 crashes.

The second image shows how the prop adapter and prop saver are used to mount the propeller to the motor. It also shows the motor heat shrinking from the previous step.

Step 9: Integrated Brushless ESCs

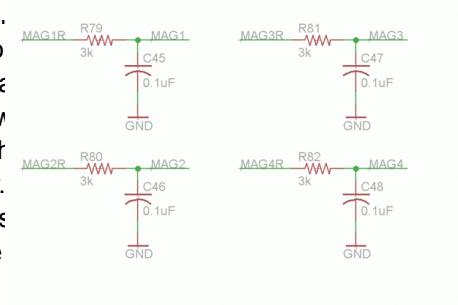


(http://cdn.instructables.com/FO9/LHPI/H2TJ4LXI/FO9LHPIH2TJ4LXI.LARGE.jpg)



(http://cdn.instructables.com/FFA/C261/H2TIYE8X/FFAC261H2TIYE8X.LARGE.jpg)

Brushless m… …d Controller (ESC) to switch current into … …osition of the rotor. Normally, this would ta… …ESC, but for this quad I especially w… …nto the PCB itself. Luckily, after search… …ba TB6588FG (http://www.… components… …20_080424_.pdf) driver chips, which were…



Well...almost perfect. (http://cdn.instructables.com/FV4/QPZY/H2TIBBQ0/FV4QPZYH2TIBBQ0.LARGE.gif)

Overclocking the TB6588FG:

The TB6588FG has a variety of options that can be "programmed" simply by placing resistors to pull pins up or down. One such option is a fault protection which disables the motor if a maximum commutation frequency is exceeded. The maximum frequency is the chip's internal clock frequency divided by 3*2^11. On the default clock speed of 5MHz, this is 814Hz. The datasheet, which is a little sparse, neglects to mention what happens if you ignore the fault and keep telling the motor to speed up. As it turns out, the driver will keep increasing the drive voltage while holding the frequency at 814Hz, causing the motor to draw a lot of current without producing any more power.

Unfortunately, the HXM1400-2000 motors can get up to significantly higher frequencies. At 2,000rpm/V and 11.1V, the no-load speed is 22,200rpm, or 367Hz. But the motor has 14 poles, so the no-load frequency is 7*367rps = 2,590Hz. The loaded speed, with a propeller, is much lower than this, but still higher than 800Hz.

Luckily, the datasheet also neglects to mention why the clock speed is 5MHz; it's just the default setting used in all the examples. The clock speed is set by a resistor and capacitor which form the time basis of an oscillator. I halved the value of the resistor, to 10k (R9, R27, R46, and R65), pushing the clock to 10MHz and the maximum commutation frequency to 1,628Hz. This seems to be enough to cover the range of loaded speeds for the HXM1400-2000 on 11.1V. I'm guessing the clock frequency can go even higher, but haven't tested it. As a side effect of the clock frequency doubling, the PWM frequency also doubled. This can increase switching losses, but I opted for the lowest of the possible PWM frequency settings to compensate for this.

Analog Inputs

Another nice thing about the TB6588FG motor driver chips is that they take analog inputs to command motor voltage, rather than the servo-style PWM inputs of commercial brushless ESCs. Servo PWMs (http://arduino.cc/en/Reference/Servo) have a refresh rate of 20ms by default, which is a little on the slow side for a small quadrotor like this. Analog inputs can be arbitrarily fast (up to the sampling rate inside the TB6588FG).

To generate the analog signals, I use the Arduino's PWM capabilities, but at very high frequency. (62.5kHz if I got the settings correct.) Then, I filter the PWMs with a first-order RC filter (third image). The filter time constant is:

tau = 3kΩ * 0.1μF = 0.3ms.

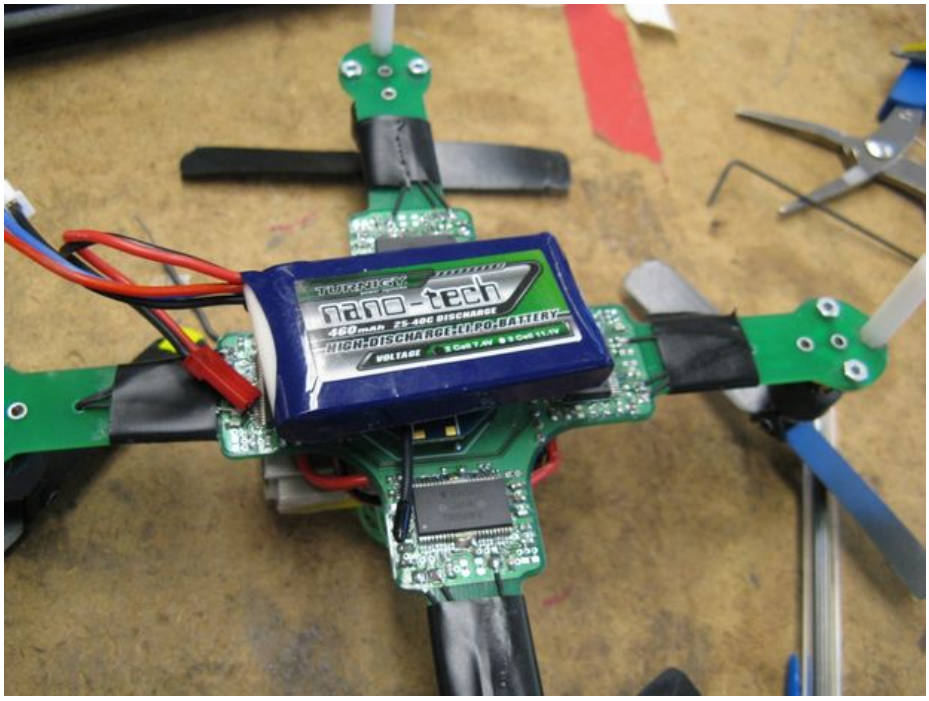
The cutoff frequency is:

f_c = 1 / (2 * pi * tau) = 531Hz.

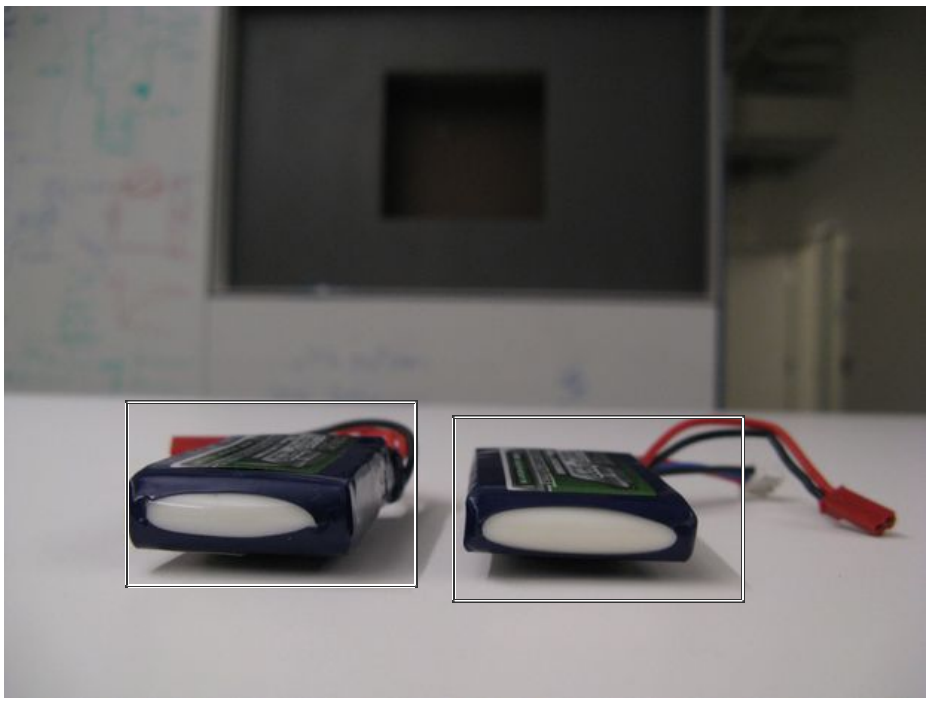
This is fast enough to capture transient speed commands, but slow enough to filter out most of the PWM ripple.

Overall, the TB6588FG is a pretty amazing little chip. It can drive each of the motors with up to 3A, and can even reverse them using a digital input. This can be convenient in case you solder the motor wires on in the wrong order. You might even be able to fly upside down, although I haven't gotten this to work yet. If you want to know more about the TB6588FG, here's the Toshiba datasheet (http://www.toshiba-components.com/motorcontrol/pdfs/TB6588FG_E_P20_080424_.pdf) for it.

Step 10: Battery



(http://cdn.instructables.com/FQT/7R2V/H2TIQ9T7/FQT7R2VH2TIQ9T7.LARGE.jpg)



(http://cdn.instructables.com/FLT/U552/H2ULR3T5/FLTU552H2ULR3T5.LARGE.jpg)

(LiPo) battery.

I've tested it with two different batteries:
- A 2S (7.4V), 460mAh, 25-40C Turnigy nano-tech LiPo (http://www.hobbyking.com/hobbyking/store/__11896__Turnigy_nano_tech_460mah_2S_25_40C_Lipo_Pack.html).
- A 3S (11.1V), 370mAh, 25-40C Turnigny nano-tech LiPo (http://www.hobbyking.com/hobbyking/store/__11895__Turnigy_nano_tech_370mah_3S_25_40C_Lipo_Pack.html).

Although it flew with both of these batteries, the flight time and stability were much better with the 3S battery, despite the slightly increased size and weight.

I mounted the battery to the XBee radio on the bottom of the board with Velcro. The extra weight of the battery would sometimes cause the XBee to fall off in hard landings, so I also superglued the XBee into its header. **Don't ever reverse the battery connections.** It will cause instant destruction of the motor controllers. Use a polarized connector such as this (http://www.hobbyking.com/hobbyking/store/uh_viewItem.asp?idProduct=9683) for connecting the battery to the board.

A bit about what the battery ratings mean:

3S = Three cells in series, each cell is nominally 3.7V.
11.1V = 3.7V * 3
370mAh = The capacity of the battery. It can put out 370mA for an hour, or 3.7A for 6 minutes.
25-40C = Maximum discharge rating, as a multiple of the 1h discharge rate.
25*0.370A = 9.7A.

Energy and Power:

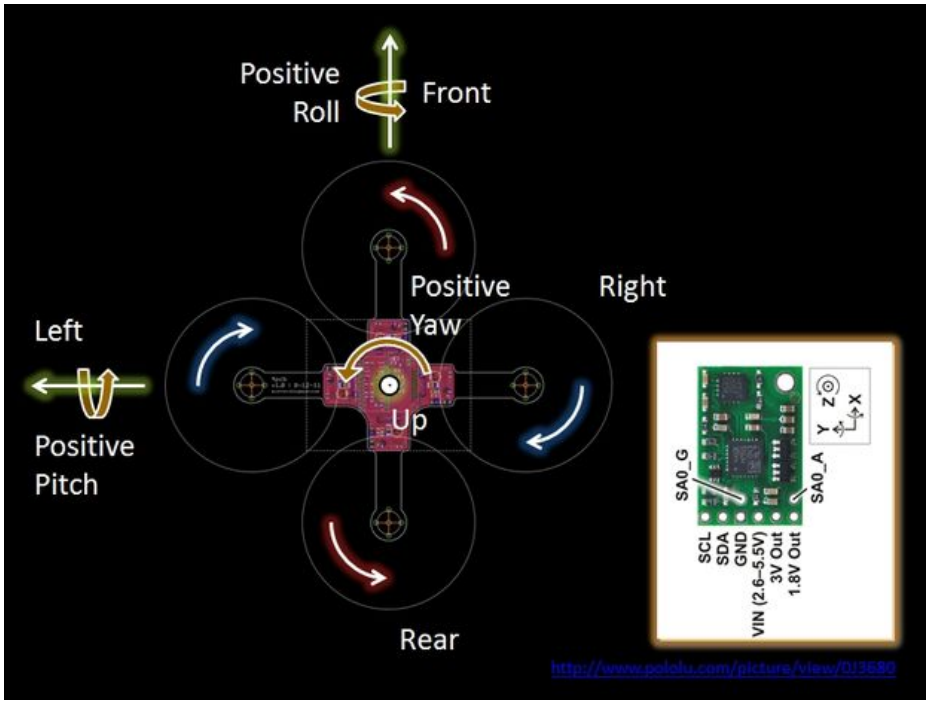
The total energy stored in the battery can be found by multiplying the voltage by the capacity:

11.1V * 0.370A * 1h = **4.1Wh**.

The average power used by the quadrotor can be found by dividing by the average flight time:

4.1W * 1h / 8min = 4.1W * 60 / 8 = **30.8W**.

Step 11: Axes and Orientation



(http://cdn.instructables.com/FU8/RP4M/H2ULR39V/FU8RP4MH2ULR39V.LARGE.jpg)

Before getting into details about the control of the quadrotor, it's important to come up with a consistent definition of the directions, axes, and orientation for the quad. The image shows the coordinate system used on 4pcb. It's a "+-mode" quadrotor, meaning that the forward direction is parallel to one of the motor arms. (The alternative is "x-mode", where forward is directly between two arms, and everything is rotated by 45º.)

The image also shows the Pololu minIMU-9 in the orientation it would be mounted on the quadrotor. (Image source: http://www.pololu.com/picture/view/0J3680 (http://www.pololu.com/picture/view/0J3680).)

In +-mode:
- The "front" direction corresponds to the IMU's X axis.
- The "left" direction corresponds to the IMU's Y axis.
- The "up" direction corresponds to the IMU's Z axis.

Note that the coordinate system used is a right-handed one, so:
- Curling the fingers of your right hand from X to Y, your thumb points in the Z direction.
- Positive rotation is defined as the direction your fingers would curl if you grab the axis with your thumb pointed in the positive direction.

Also note that:
- Positive pitch is pitching downward.
- Positive roll is rolling to the right.
- Positive yaw is yawing to the left.

Step 12: Command Matrix

	Front	Right	Rear	Left
Throttle	+	+	+	+
Pitch	-	0	+	0
Roll	0	-	0	+
Yaw	-	+	-	+

(http://cdn.instructables.com/FL1/3Y9A/H2TIUBHI/FL13Y9AH2TIUBHI.LARGE.gif)

A quadrotor has four motors that can spin at different speeds, so it seems reasonable that it has four "degrees of freedom" that can be controlled. The four controllable DOFs are:

1. Throttle
2. Pitch
3. Roll
4. Yaw

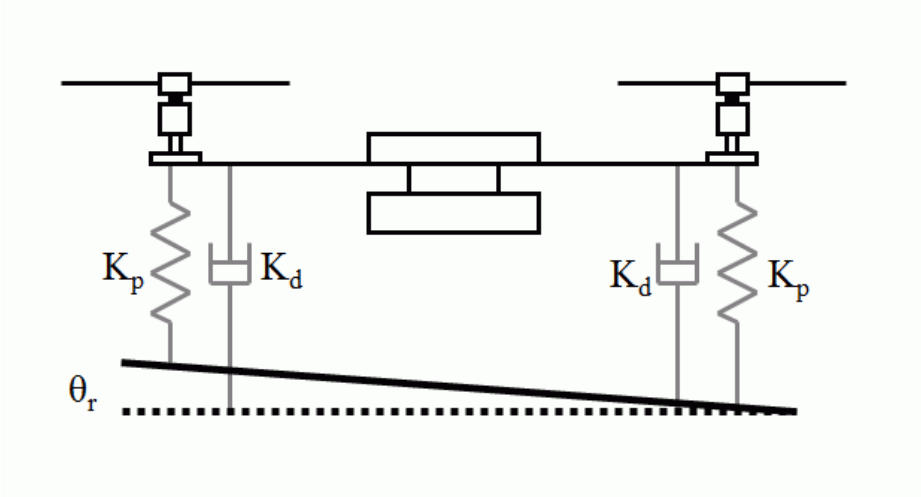
How those control inputs are derived and sent as radio commands to the quadrotor will be covered in a later Step. This Step focuses on how to get from those four commands (the inputs) to the four motor speeds (the outputs).

You might be able to look at the image in the previous Step and intuitively figure out what the input/output mapping for a quadrotor should be. It's easiest to represent as a 4x4 matrix, which is just a table where the rows are inputs and the columns are outputs (or vice versa). This Step's image show the command matrix for a +-mode quadrotor with propeller directions indicated in the previous Step's image.

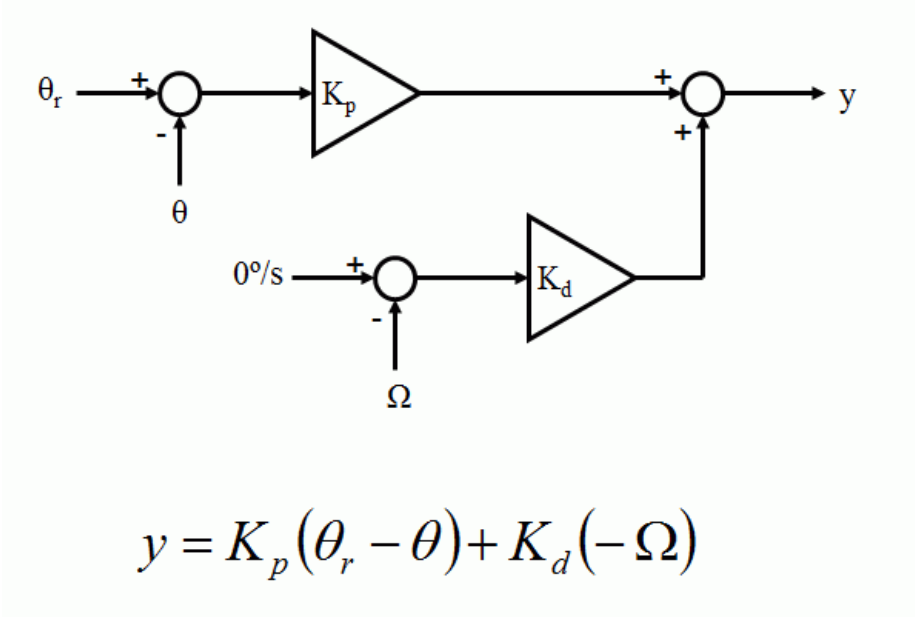
A [+] indicates that the motor should be sped up in response to a positive input command.
A [-] indicates that the motor should be slowed down in response to a positive input command.
A [0] indicates that the motor speed is unaffected by a positive input command.

Throttle, pitch, and roll work by increasing or decreasing the thrust produced by particular motors to create a net torque on the quadrotor frame. Yaw works on a different principle, using reaction torque from the motor to rotate the frame. The reaction torque for each motor is in the opposite direction as the propeller and is proportional to the drag seen by the propeller as it pushes through the air.

Step 13: Feedback Control



(http://cdn.instructables.com/F5H/218E/H2TIUBI5/F5H218EH2TIUBI5.LARGE.gif)



$$y = K_p(\theta_r - \theta) + K_d(-\Omega)$$

(http://cdn.instructables.com/FKV/LLFR/H2TIBBTR/FKVLLFRH2TIBBTR.LARGE.gif)

In an ideal world, you could just take the four inputs (throttle, pitch, roll, yaw) and directly map them to the four motor outputs using the command matrix from the previous step. But there are some things which make it impractical to do this in real life:

1. Disturbances such as wind and non-idealities such as differences in the motors and propellers cause the real-life dynamics to be noisy and variable. Direct command mapping doesn't take these into account, and our mind, eyes, and hands might not be fast enough to react to these in real time, especially on a small quadrotor.

2. We want the quadrotor to have some degree of "autonomy". Particularly, it would be nice if the quadrotor could self-level, returning to nearly horizontal when we command zero pitch, roll, and yaw. With direct mapping, we are commanding the quadrotor to rotate but it won't know to return to horizontal when we're done.

This is where **feedback control** comes in. Despite its complex mathematical notations, the concept of feedback control is simple. *Imagine washing your hands in a sink you've never used before*. You set the faucet knob to some middle position. If the water's too cold, you turn it up. If the water's too warm, you turn it down. How much you turn it up or down depends on how hot or cold it is compared to your liking. If the water starts cold, but then rapidly heats up, you may

preemptively turn the knob back down to prevent it from overshooting and burning your hands. *All these concepts are mathematically formalized in feedback control.*

Feedback Control on a Quadrotor:

A common structure for feedback control is called PID (Proportional-Integral-Derivative) control. The thing to be controlled in this case is the **angle** (pitch, roll, or yaw angle) of the quadrotor. This is analogous to the temperature of the water in the sink. With no input commands, we try to control the angle to be zero. However, we can also command a non-zero angle to make the quadrotor move. The commands we send to the motors are based on the error between the angle we want and the angle we actually have, as measured by the IMU.

- **Proportional (P):** The command is proportional to how much angular error we have. It helps return the quadrotor to zero angle, or push it to whatever angle you command.

- Integral (I): The command is proportional to the accumulated error over time. It can help fight disturbances like wind or asymmetric motor performance. I don't use this on my quad, although I left placeholder code for it in the Arduino project.

- **Derivative (D)**: The command is proportional to the rate of change of the error*. It resists motion and it keeps the angle from overshooting the target.

Mass-Spring-Damper Analogy:

The effects of PD control (no integral term) on a quadrotor are similar to adding a virtual spring and virtual damper (shock absorber) to the quadrotor, which is the mass. See the first image for a graphical depiction of this. The spring stiffness is set by a constant, *Kp*, the "proportional gain". The damping rate is set by another constant, *Kd*, the "derivative gain". The desired angle, θr, defines the angle at which the springs are evenly stretched. Imagine attaching the springs to a movable plank, and rotating the plank to command the quadrotor to go to a certain angle.

- Increasing *Kp* pushes the quadrotor toward the reference angle faster, but can also result in more overshoot and oscillation.

- Increasing *Kd* slows down the rotation rate of the quadrotor, but can also damp out oscillations.

Note that the gains can be increased to a point where this model breaks down. If *Kp*, *Kd,* or both are too high, the controller will start amplifying noise, leading to oscillations and instability. These oscillations tend to be at a faster frequency than the oscillations that would be seen from a high *Kp*:*Kd* ratio. If you see fast oscillations, the best thing to do is turn both gains down.

Tuning the gains takes practice and experience, and depends on your exact frame and flying preference. A really good guide to PID tuning for multirotors, with video example to show the various types of oscillations, can be found here (http://wiki.openpilot.org/display/Doc/Stabilization+Tuning++Multirotor). Hopefully, the spring/damper analogy helps you think about the gains intuitively.

*Actually, in practice it is sometimes easier to use the measured rate of rotation directly, instead of trying to measure the rate of change of the error. The error only exists in software and computing the derivative of it can be noisy. Practically speaking, using the measured rate of rotation directly from the gyro works just fine. To map this into the mass-spring-damper analogy, the dampers are connected to ground (zero angle) instead of to the plank. In this configuration, they resist all rotation, even commanded rotation.

Pitch, Roll, and Yaw Controllers:

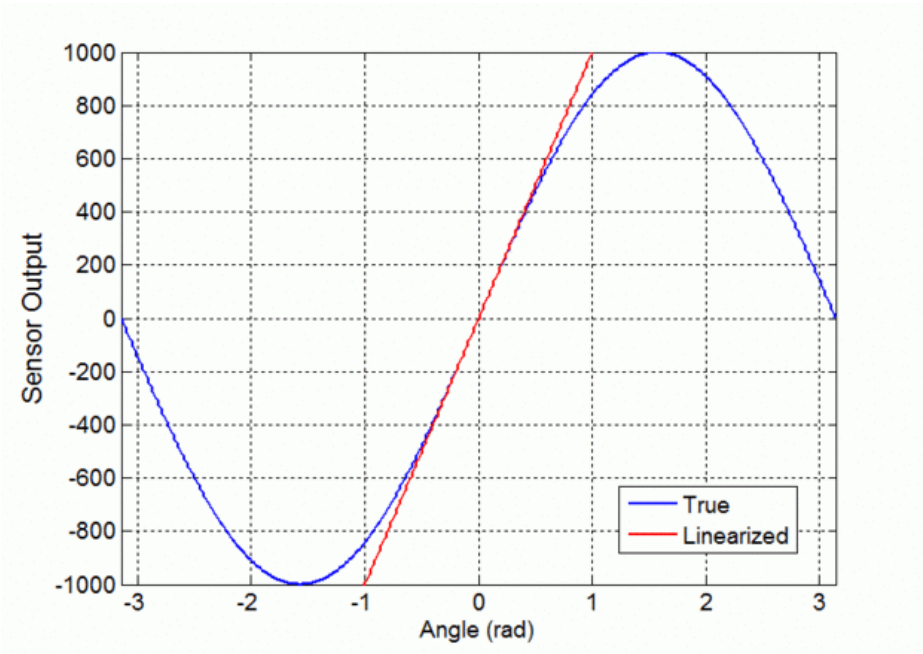
The quadrotor actually has three independent feedback controllers, one each for pitch, roll, and yaw. Throttle is directly mapped to all four motors with no feedback control in this quadrotor. With an altitude sensor, a fourth feedback controller could be added.

The **pitch** and **roll** controllers are PD controllers that match up exactly with the first image. To the extent that the quadrotor is symmetric, the pitch and roll gains should be the same. The outputs from the proportional and the derivative gains are summed together and sent to the motors through the command matrix in the previous Step. The second image shows the pitch and roll PD controllers in block diagram form. The variables are:

Kp - Proportional Gain
Kd - Derivative Gain
θr - Reference/Command Angle
θ - Measured Angle from the IMU
Ω - Measured Rate of Rotation from the IMU.
y - The output command.

The **yaw** controller is only based on rate, so it doesn't exactly match the images. The error is just the difference between the commanded yaw rate and the measured yaw rate from the IMU. The magnetometer in the IMU could be used to measure absolute yaw angle to implement a full PD controller, but I haven't tried this yet.

Step 14: IMU Part 1: Sensors



(http://cdn.instructables.com/F1B/M3QM/H2TJ4TMR/F1BM3QMH2TJ4TMR.LARGE.gif)

Feedback control is only useful if there is a quality measurement signal to feed back. In the case of a quadrotor, it's critical to have an accurate measurement of both **angle** and **angular rate**. This requires two different types of sensors. Sound

familiar (http://www.instructables.com/id/Segstick/step9/Two-sensors-one-angle/)? I wasn't joking when I said that quadrotors are the new Segway. In fact, they're more like two flying Segways, one for pitch and one for roll. Each axis has its own accelerometer and gyro measurement, and the way the signal from these two sensors can be combined is identical to what I used in the Seg...stick Instructable (http://www.instructables.com/id/Segstick/) and in this white paper (http://web.mit.edu/scolton/www/filter.pdf). Before looking at the Complementary Filter, though, it's important to understand what each of the sensors does:

The Accelerometer:

The accelerometer actually doesn't measure acceleration. It measures force per unit mass along each axis. You can use this measurement to determine the direction of gravity, which exerts a force downward, as long as the quadrotor isn't accelerating.

The Polulu minIMU-9 (http://www.pololu.com/catalog/product/1265) is a lot less hassle than the original analog sensor because it is self-zeroing and reports a digital value with a known scaling factor. The scaling factor is even a nice simple number: **1mg/digit** with the most sensitive range. That is, 1/1000th the acceleration due to gravity per digit. I got this value directly from the accelerometer datasheet (http://I got this value directly from the accelerometer datasheet.). If you point the sensor's X axis directly downward, it should report back a value of 1,000.

Converting from an acceleration measurement to an angle requires either a lot of trigonometry or a linearization. Since the quadrotor spends most of it's time horizontally and deviates by at most 30º in any direction, I chose the linearization. The first image shows how to linearize about 0º. The slope of the line that is tangent to the sensor output vs. angle curve at 0º is 1000digit/rad. The inverse is what is called A_GAIN in the code:

A_GAIN = 0.001rad/digit = 0.0573º/digit.

This is the number by which to multiply the raw accelerometer value to get an approximate angle in degrees.

The Gyro:

The gyroscope isn't actually a gyroscope; there are no spinning flywheels inside the chip. It's more accurately called an angular rate sensor, and uses a MEMS tuning fork structure to measure rate of rotation based on Coriolis force. Like the accelerometer, there's one for each axis.

Since this is a nice digital IMU, the value reported is directly in physical units, although the scaling factor is a bit odd. It's given in the datasheet for the gyro (http://www.pololu.com/file/download/L3G4200D.pdf?file_id=0J491) as 8.75mdps/digit, or 0.00875(º/s)/digit. I use this value directly as G_GAIN in the code.

G_GAIN = 0.00875(º/s)/digit.

This is the number by which to multiply the raw gyro value to get an angular rate in degrees per second.

Motivation for Merging Sensors to Estimate Angle:

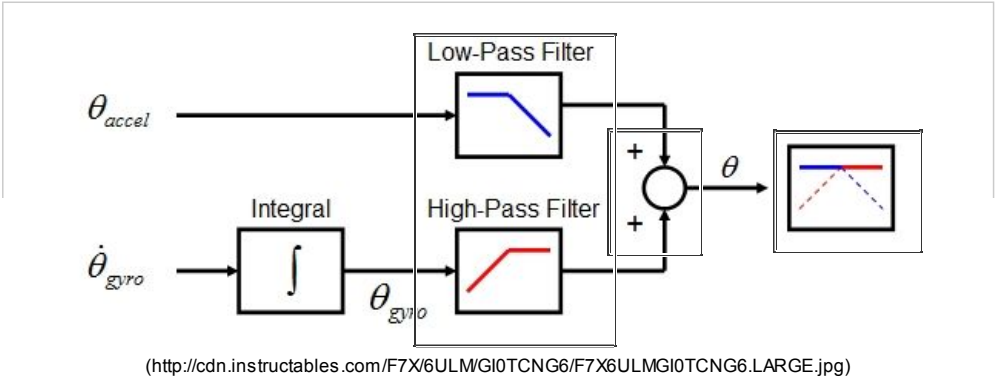
The gyro directly measures angular rate, so the rate feedback is taken care of by the gyros alone. In fact it should be possible to fly the quadrotor on rate feedback only, but it would need a different control structure and wouldn't self-level. To get good attitude control, angle feedback is also required.

Unfortunately, the accelerometer estimate of angle isn't perfect. Horizontal acceleration will disturb the angle estimate, since the accelerometer can't distinguish the difference between accelerating forward and leaning backwards. The quadrotor doesn't spend much time accelerating horizontally, but when it does, it can easily pull large fractions of 1g, which will disturb the angle estimate substantially.

The gyro can also produce an angle estimate, by numeric integration: If you know how fast you're rotating, you can predict what angle you'll be at in the next time step by multiplying the rate by the sample time. This estimate of angle can work for short periods of time, but over long a duration it will drift away from the true angle. No matter how good and well-zeroed the gyro is, numeric integration will always cause the angle estimate produced by the gyro alone to drift over time.

So, the accelerometer is bad for short-term angle estimate, but provides a stable long-term average equal to the gravity vector. The gyro is good for short-term angle estimates, but will drift over long periods of time. The Complementary Filter, discussed in the next Step, merges the sensors together smoothly at different time scales to obtain a better angle estimate than either sensor alone could produce.

Step 15: IMU Part 2: Complementary Filter



(http://cdn.instructables.com/F7X/6ULM/GI0TCNG6/F7X6ULMGI0TCNG6.LARGE.jpg)

My method of choice for combining the accelerometer and gyro signals into a clean and stable angle estimate is the Complementary Filter. It's computationally efficient and it directly addresses both transient horizontal acceleration and long-term gyro drift. The idea is to use the gyro for short term angle estimates, by numeric integration, but switch to the accelerometer for long-term estimates, by averaging. To do this in a continuous way, low-pass and high-pass filters are used. This is identical to the method I used in the Seg...stick Instructable (http://www.instructables.com/id/Segstick/), Step 10 (http://www.instructables.com/id/Segstick/step10/A-very-flattering-filter/).

The image shows the a Complementary Filter in block diagram form. The sensor signals, converted to degrees and degrees per second, are the inputs. The accelerometer angle is low-pass filtered, reducing the influence of short-duration signals but maintaining a long-term average (from gravity). The gyro angular rate is first integrated, to get an estimate of angle, and then high-pass filtered, to remove any long-term drift. The two filtered signals are summed together to create a single angle estimate that combines the best parts of both sensor signals.

It's one of the few cases where the code is more concise than the block diagram or the written explanation:

```
// Excerpt from pd_digitalIMU.pde in 4pcb_ARD:
// --------------------------------------------------------------------
rate_pitch = (float) g_pitch * G_GAIN;
angle_pitch = AA * (angle_pitch + rate_pitch * DT);
angle_pitch += (1.0 - AA) * (float) a_pitch * A_GAIN;
// --------------------------------------------------------------------
```

The first line converts the raw gyro signal, **g_pitch**, into º/s, as discussed in the previous Step. The end of the second line is the numeric integration. **DT** is the loop time, 0.01s. A small change in angle, **rate_pitch * DT**, is added to the previous angle estimate.

The rest of the second line and all of the third line form the high-pass and low-pass filters, in discrete form. The high-pass acts on the integrated angle estimate and the low-pass acts on the accelerometer signal, **a_pitch**. The constant **AA** determines the time scale of the high- and low-pass filters, and is a value you can tweak depending on your quadrotor. Larger **AA** trusts the integrated gyro signal for longer. Shorter **AA** merges the accelerometer signal in faster.

The value for **AA** used in 4pcb corresponds with a time constant of **1s**. This is the approximate length of time for which the gyro angle estimate is trusted, and the approximate length of time over which the accerometer signal is averaged. Since these are infinite impulse-response (IIR) filters, the transition is smooth and continuous and the time constant only represents where the weighting of the two filters is equal.

I wrote up a more thorough explanation of the Complementary Filter as applied to self-balancing system in this white paper (http://web.mit.edu/scolton/www/filter.pdf). It covers some of the details of how **AA** is related to the time constant of the low-pass and high-pass filters, how rate sensor bias affects the filter, and some more details.

Step 16: Putting it Together: The Code

```
readGyro();
readAcc();

// START OF + MODE ------------------------------------------------------------------------
a_pitch = (accel_x_zero - accel_x);
a_roll = (accel_y - accel_y_zero);
// a_z = (signed int) analogRead(A_Z) - a_z_z;
g_pitch = (gyro_y - gyro_y_zero);
g_roll = (gyro_x - gyro_x_zero);
g_yaw = (gyro_z - gyro_z_zero);

rate_pitch = (float) g_pitch * G_GAIN;
rate_roll = (float) g_roll * G_GAIN;
rate_yaw = (float) g_yaw * G_GAIN;

angle_pitch = AA * (angle_pitch + rate_pitch * DT);
angle_pitch += (1.0 - AA) * (float) a_pitch * A_GAIN;
pitch_error = (float) (pitch_command - 127) / 127.0 * MAX_ANGLE - angle_pitch;
//pitch_error_integral += pitch_error * DT;
//if(pitch_error_integral >= INT_SAT) { pitch_error_integral = INT_SAT; }
//if(pitch_error_integral <= -INT_SAT) { pitch_error_integral = -INT_SAT; }

angle_roll = AA * (angle_roll + rate_roll * DT);
angle_roll += (1.0 - AA) * (float) a_roll * A_GAIN;
roll_error = (float) (roll_command - 127) / 127.0 * MAX_ANGLE - angle_roll;
//roll_error_integral += roll_error * DT;
//if(roll_error_integral >= INT_SAT) { roll_error_integral = INT_SAT; }
//if(roll_error_inttegral <= -INT_SAT) { roll_error_integral = -INT_SAT; }
```

(http://cdn.instructables.com/FAQ/HI2X/H2TJ4U6C/FAQHI2XH2TJ4U6C.LARGE.gif)

(http://cdn.instructables.com/F2X/D49E/H2TJ4U69/F2XD49EH2TJ4U69.LARGE.gif)

The code for my POD quad is written in Arduino-style C/C++ in the Arduino IDE. If you've never used an Arduino before, you can get up to speed pretty quickly using the resources on the Arduino hopepage (http://www.arduino.cc/). The Arduino project is included in the documentation (Step 1). In total, the code is approximately 500 lines, not including includes. Even in the main file, a lot of the code is running setup or background stuff like functions to read the IMU serial data. (I stripped down the Pololu libraries from here (https://github.com/pololu/L3G4200D) and here (https://github.com/pololu/LSM303) to include only functions I needed.)

The most interesting part of the code, which implements the concepts from the past four Steps, fits on one page (or the two images in this Step). Here it is, with brief descriptions of what each part does:

```
  readGyro();
  readAcc();

  // START OF + MODE ---------------------------------------------------------------------
  a_pitch = (accel_x_zero - accel_x);
  a_roll = (accel_y - accel_y_zero);
  // a_z = (signed int) analogRead(A_Z) - a_z_z;
  g_pitch = (gyro_y - gyro_y_zero);
  g_roll = (gyro_x - gyro_x_zero);
  g_yaw = (gyro_z - gyro_z_zero);
```

This part grabs the raw accelerometer and gyro signals. It also does zeroing, although I have all my zeros set to...zero...since the Pololu minIMU-9 does a pretty good job of zeroing its outputs already. If necessary, the zero values could be changed to trim the quadrotor. **Note: a_pitch is reversed.** This is a physical issue. Under positive pitch, the gravity vector points toward the X axis of the IMU. This

registers as negative acceleration in X, since the accelrometer can't distinguish between leaning forwards and accelerating backwards. a_roll doesn't have this reversal.

```
  rate_pitch = (float) g_pitch * G_GAIN;
  rate_roll = (float) g_roll * G_GAIN;
  rate_yaw = (float) g_yaw * G_GAIN;
```

Convert the raw gyro signals into physical units of º/s.

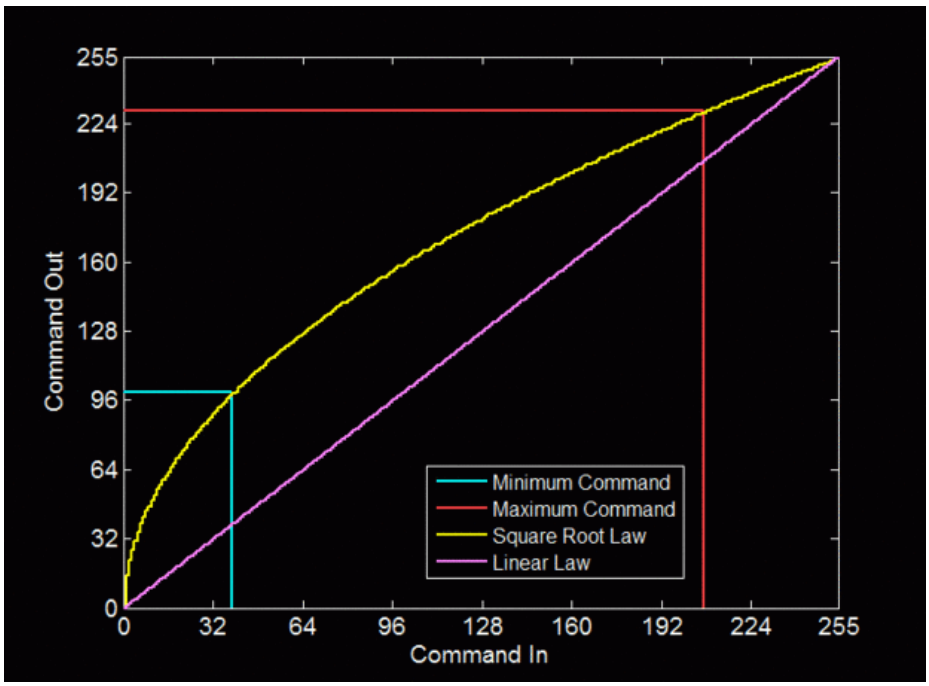
```
  angle_pitch = AA * (angle_pitch + rate_pitch * DT);
  angle_pitch += (1.0 - AA) * (float) a_pitch * A_GAIN;
  pitch_error = (float) (pitch_command - 127) / 127.0 * MAX_ANGLE - angle_pitch;
  //pitch_error_integral += pitch_error * DT;
  //if(pitch_error_integral >= INT_SAT) { pitch_error_integral = INT_SAT; }
  //if(pitch_error_integral <= -INT_SAT) { pitch_error_integral = -INT_SAT; }

  angle_roll = AA * (angle_roll + rate_roll * DT);
  angle_roll += (1.0 - AA) * (float) a_roll * A_GAIN;
  roll_error = (float) (roll_command - 127) / 127.0 * MAX_ANGLE - angle_roll;
  //roll_error_integral += roll_error * DT;
  //if(roll_error_integral >= INT_SAT) { roll_error_integral = INT_SAT; }
  //if(roll_error_inttegral <= -INT_SAT) { roll_error_integral = -INT_SAT; }
```

Pitch and roll Complementary Filters, as well as angle error calculation. **pitch_command** and **roll_command** come from the radio, and are transmitted from the ground station based on user inputs. **MAX_ANGLE** is a user-define constant that sets the maximum commanded pitch and roll angle in degrees. The integral control terms are all commented out, but feel free to play with them.

```
  // Control
  front_command_f -= pitch_error * KP;
  front_command_f -= pitch_error_integral * KI;
  front_command_f += rate_pitch * KD;
  front_command_f -= ((float) (yaw_command - 127) / 127.0 * MAX_RATE -
rate_yaw) * KY;

  rear_command_f += pitch_error * KP;
  rear_command_f += pitch_error_integral * KI;
  rear_command_f -= rate_pitch * KD;
  rear_command_f -= ((float) (yaw_command - 127) / 127.0 * MAX_RATE -
rate_yaw) * KY;

  right_command_f -= roll_error * KP;
  right_command_f -= roll_error_integral * KI;
  right_command_f += 1.3 * rate_roll * KD;
  right_command_f += ((float) (yaw_command - 127) / 127.0 * MAX_RATE -
rate_yaw) * KY;

  left_command_f += roll_error * KP;
  left_command_f += roll_error_integral * KI;
  left_command_f -= rate_roll * KD;
  left_command_f += ((float) (yaw_command - 127) / 127.0 * MAX_RATE -
rate_yaw) * KY;
  // END OF + MODE ----------------------------------------------------------------------
```

This is the command matrix from Step 12. The yaw controller is in-lined, which is kind of silly. **yaw_command** comes from the radio, and is transmitted by the ground station. **MAX_RATE** is a user-defined constant that sets the maximum commanded yaw rotation rate in degrees per second.

Step 17: A Minor Tweak: Square-Root Control Law



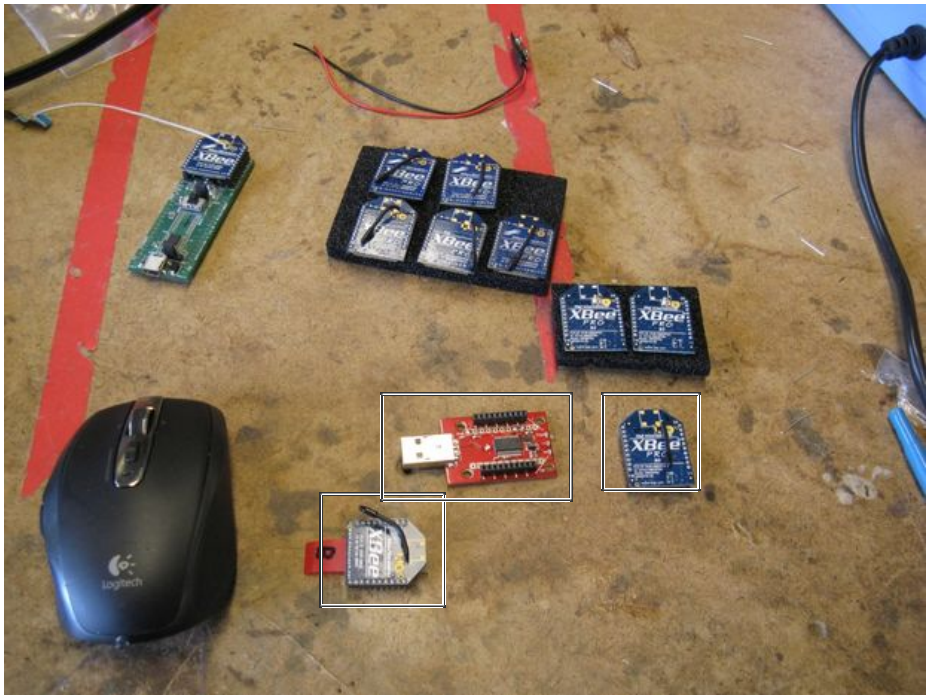
(http://cdn.instructables.com/FI5/0QGW/H2TIL75I/FI50QGWH2TIL75I.LARGE.gif)

A lot of the control theory used in this project, particularly the mass-spring-damper analogy for the PD controller, relies on the fact that the motors produce thrust proportional to angle error, or angular rate. The commands sent to the motor are not thrust values, though. They represent a voltage, or more precisely a fraction of the battery voltage, applied at the motor terminals. The voltage and speed of the motor are closely related, which is why motor controllers are often called speed controllers. But thrust is more closely related to the *square* of propeller speed. (Just like lift (http://en.wikipedia.org/wiki/Lift_coefficient) is proportional to airspeed squared.)

Since the quadrotor spends most of its time operating in a small band of speeds around hover, the difference may not matter very much. But, I played with the idea of "undoing" the square relationship by passing the motor commands through a square root function before sending them to the motors. The square root function has to be normalized (see the image), but it can be done with a look-up table to spare the CPU time.

I found that this modification did improve stability, especially for manually holding altitude.

Step 18: Radio Interface



(http://cdn.instructables.com/F3Z/J5EY/H2TIL78D/F3ZJ5EYH2TIL78D.LARGE.jpg)

(http://cdn.instructables.com/FKH/VF5J/H2TIQBEO/FKHVF5JH2TIQBEO.LARGE.gif)

4pcb's radio interface is based on XBee Series 1 (http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/point-multipoint-rfmodules/xbee-series1-module#overview) modules. These are small 2.4GHz digital radio modules that behave like wireless serial ports. In the Bill of Materials, I specified two of the higher-power XBee Pro transceivers. At $32, these are about 50% more expensive than the normal XBee modules. But they will give you much longer radio range, which is important for a quadrotor.

(http://cdn.instructables.com/FSS/H5AF/H2VARRSB/FSSH5AFH2VARRSB.LARGE.gif)

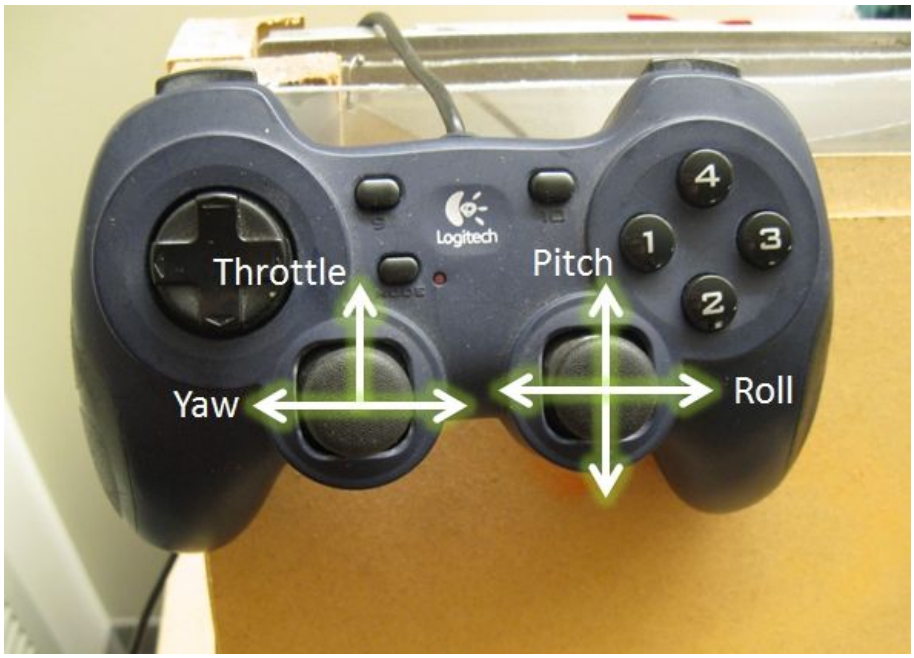
One XBee connects directly to the quadrotor, on the bottom of the board. (I superglued mine in, so it doesn't fall out during hard landings.) The other connects to a ground station computer using an XBee-to-USB adapter such as the Sparkfun XBee Explorer (http://www.sparkfun.com/products/8687). One advantage XBee radios have over standard RC equipment is that the data flows both ways. Commands are sent from the ground station to the quadrotor, and telemetry is sent back.

Data is sent in packets, one byte at a time, at 57600 bits per second. I chose this bit rate because it's also the default rate for the Arduino bootloader, so you can program the Arduino Pro Mini over XBee without needing a separate programming cable. (You still have to hit reset on the Pro Mini.) The second image shows the flow of data and the byte definitions of the command and telemetry packets. The third image shows how to recover the angle and angular rates in degress (per second) from the raw telemetry data.

The quadrotor transmits and reads data at 100Hz. You can send commands to it at up to this rate. (Slower or slightly faster is okay too.) You can look through the source code in the Arduino project to see how the radio interface is implemented in

detail.

Step 19: Ground Station



(http://cdn.instructables.com/F5G/6F69/H2ULR6OE/F5G6F69H2ULR6OE.LARGE.jpg)

(http://cdn.instructables.com/FPB/P3UL/H2TIFZQV/FPBP3ULH2TIFZQV.LARGE.jpg)

The ground station transmits the commands {throttle, pitch, roll, yaw} to the quadrotor based on user inputs. It consists of three parts: a user input device, an XBee radio as discussed in the previous step, and some software to map input device axes to commands.

Input Device: Logitiech Gamepad

(http://cdn.instructables.com/FHT/DEEW/H2ULR7X0/FHTDEEWH2ULR7X0.LARGE.gif)

I chose to use a Logitech Dual Action gamepad as my input device. The first image shows how the Logitech gamepad's joystick axes get mapped to the four quadrotor commands. Experienced RC pilots might recognize this joystick layout as Mode 2 (http://www.rc-airplane-world.com/rc-transmitter-modes.html). Unlike an RC radio, the throttle stick on the gamepad will be spring loaded. This makes it a little more difficult to hold altitude, but it's not too hard to get used to.

You can use any other USB HID device as the controller, but you'll have to modify the ground station software to correctly map the joystick axes to the commands. I've also tried a flight simulator joystick (http://www.amazon.com/Logitech-Extreme-Joystick-Silver-Black/dp/B00009OY9U). I didn't like the feel as much, but this was back when I was first learning how to fly it.

You can also use a standard RC flight transmitter. In this case, you don't even need a computer for interfacing. An Arduino can read the pulse position modulation (PPM) signals from the transmitter's trainer port, and send out commands via XBee. Here's a guide (http://www.etotheipiplusone.net/?p=1820) on how to map signals from the transmitter's trainer port to the XBee.

Interfacing Software:

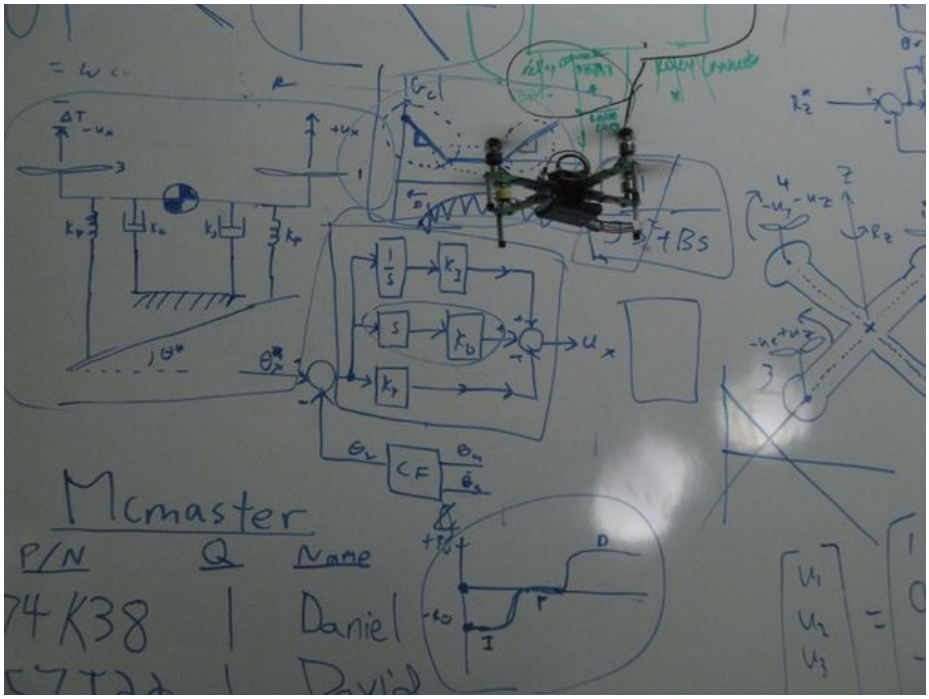
If you do decide to use a USB HID controller, you'll need some software running on a computer to interface with the controller and to send commands to the XBee radio. This software can also grab telemetry from the quadrotor and display it, or log it for later analysis.

I wrote a simple ground station program in Visual Basic, which is included in the project documentation (Step 1). The executable is in the 4pcb_EXE folder and the source is in the 4pcb_VB folder. To run the executable, you'll need the .NET Framework runtime files, which can be downloaded from here (http://www.microsoft.com/en-us/download/details.aspx?id=21). If you're interested in modifying the ground station software, you can use Visual Basic 2010 Express (http://www.microsoft.com/visualstudio/en-us/products/2010-editions/visual-basic-express), which is free.

My ground station is nowhere near as functional as it could be. First, only some of the trims are active and they don't save their values when you close the program. It also only logs the telemetry (in a text file called 4pcbdata.txt). It could do so much more - like display a virtual 3D quadrotor on the screen that mimics the orientation of the real quadrotor.

The .NET-based ground station will only run in Windows. If you're on a different OS, you can still write ground station software in any programming language that can interface to HID game controllers and a virtual serial port (so, pretty much any language). One easy option is Processing (http://www.processing.org/), an application/GUI programming language with and Arduino-like IDE. There is a third-party library for Procssing called ProCONROLL (http://creativecomputing.cc/p5libs/procontroll/) which interfaces to HID controllers. I wrote a very quick Processing/ProCONTROLL-based ground station for controlling robots (https://sites.google.com/site/2007arduino/example-code/controller-tutorials/usb-ps2-controller) that could easily be modified to send the command packet needed for the quadrotor.

Step 20: Learn to Fly!



(http://cdn.instructables.com/FJ6/7QR2/H2TIL761/FJ67QR2H2TIL761.LARGE.jpg)

This isn't the type of quadrotor that can fly itself. Even when properly tuned and trimmed, it still requires a good amount of practice to keep it in one place for an extended period of time. It definitely took me some time to get used to the controls, although I expect an experienced RC helicopter pilot could pick it up and fly it right away.

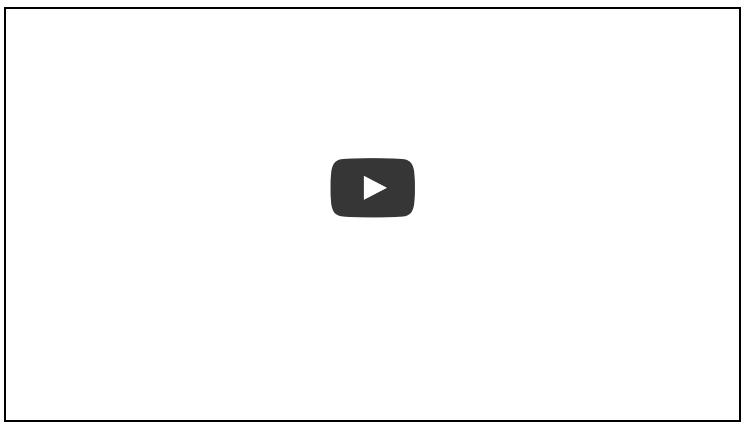
Overall, it took me a few months of free time to improve and tune the controller, learn the controls, and become comfortable flying it. Along the way, I crashed it probably dozens of times, replaced several motors, probably three full sets of propellers, and some landing gear parts, and resoldered a bunch of things. The PCB frame has never broken, though.

To show that it wasn't all magic and instant success, here are some videos that show a chronological progression through a few months of control refinement *and* flying practice:

Very early flight testing with old IMU and no foam tape.

Testing different control loop structures.

The first big breakthrough, foam-tape mounting the IMU to reduce vibrations.



One of the motors was weak/damaged, so it would only move in one direction. :(

First test flight on the Pololu minIMU-9. Not well-configured yet.

Using the IMU's 12.5Hz low-pass filter. Much better.

First flight with square-root control law.

Just after precision prop balancing. First attempts at hand launching/landing.

As good as it's ever looked!

Make Comment

1-40 of 50 Next » (http://www.instructables.com/id/PCB-Quadrotor-Brushless/?&sort=ACTIVE&limit=40&offset=40#DISCUSS)

satterfieldmartha (/member/satterfieldmartha/) 1 month ago Reply (CF9L07CHWAB0ET1)

(/member/satterfieldmartha/)

Only Wow, this project is amazing!
(first sorry for my bad English)
I tried to adapt the code in my own quad to do it fly in x mode, but without success. The difference is that my IMU is turned 45 degrees compared to the project that there is here, the pitch of the IMU is at the center between the motors. This orientation of the IMU creates problems?
Thanks to the help and the patience.

satterfieldmartha (/member/satterfieldmartha/) 1 month ago Reply (C4GUT90HWAB0ESZ)

(/member/satterfieldmartha/)

Only Wow, this project is amazing!
(first sorry for my bad English)
I tried to adapt the code in my own quad to do it fly in x mode, but without success. The difference is that my IMU is turned 45 degrees compared to the project that there is here, the pitch of the IMU is at the center between the motors. This orientation of the IMU creates problems?
Thanks to the help and the patience.

cybernetic (/member/cybernetic/)

2 months ago Reply (CTTJMAKHUQIFA2X)

(/member/bernald/)

Only Wow, this project is amazing!
(First sorry for my bad English)
I tried to adapt the code in my own quad to do it fly in x mode, but without success. The difference is that my IMU is turned 45 degrees compared to the project that there is here, the pitch of the IMU is at the center between the motors. This orientation of the IMU creates problems?
Thanks to the help and the patience.

---

jiffymanager (/member/jiffymanager/) 3 months ago Reply (CD25ZJOHUFW6K40)

(/member/jiffymanager/)

good

---

calmlunch (/member/calmlunch/) 3 months ago Reply (C9N32CBHU843YJM)

(/member/calmlunch/)

super

---

clickyummy (/member/clickyummy/) 4 months ago Reply (CK0EPCFHSICYV82)

(/member/clickyummy/)

found this blogspot by shane: http://scolton.blogspot.com/2011/08/something-like-flight.html this particular blog about the control structure really helped me. the entire blog site, matter fact, is a gold mine of information. go check it out if you're also having trouble or would just like to see how amazing shane really is.

---

bearblue (/member/bearblue/) 5 months ago Reply (C9BOP1DHS3GTA19)

(/member/bearblue/)

super

---

patriots21 (/member/patriots21/) 6 months ago Reply (C31C6Z5HQC1N47C)

(/member/patriots21/)

Would like to thank you man. Your balance filter theory helped us a lot get through our final undergraduate research last year. Worked better than any DCM or kalman filter code we ever wrote, probably because we understood your filter the most. I just got back to the hobby after a half-year lay-off, and just now remembered I should thank you for posting all these wonderful control information, without which I'm sure we would have all stayed grounded.

cgrrty (/member/cgrrty/) 7 months ago Reply (C9I4LOYHOSHKBJ7)

(/member/cgrrty/)

mark !very cool

Combatraffi (/member/Combatraffi/) 9 months ago Reply (CQ3SG1THMVJOMYW)

(/member/Combatraffi/)

Great Instructable. I just have one question... In a video that was added its mentioned that activating the 12.5HZ low pass filter on the gyro really helped stability. After reading the datasheet on the gyro it says that there is a low pass filter but the sheet only gives instructions for controlling the High-Pass. Did you add a software filter or am I missing something on the datasheet?

---

scolton (/member/scolton/) (author) Combatraffi 9 months ago Reply (C9PXO2HHMVJ6ZS7)

(/member/scolton/)

It is a little unclear in the datasheet. Right above the High Pass settings, starting on p.29 of

http://www.pololu.com/file/downlo
file_id=0J491, there is a setting for "DR and BW" (data rate and bandwidth) which are the low-pass filter and output data rate settings. The lowest setting is 12.5Hz cut-off for the LPF. It's rather harsh, but it seemed to work better than 25Hz. If you could cut the vibration down a lot (by careful balancing of propellers and such), a higher frequency cutoff would be better.

bgumelar (/member/bgumelar/) 1 year ago Reply (CPFZ4U0HGU2EXGK)

how long this quadrotor can fly??

(/member/bgumelar/)

scolton (/member/scolton/) (author) bgumelar 9 months ago Reply (CEUXL8CHMVJ6ZR0)

Flight time is about 8 minutes, depending on the size of the battery.

(/member/scolton/)

susan111 (/member/susan111/) 1 year ago Reply (CSZA9PQHIXPVTWK)

Hi, i think u did a great job. at least, i think it's a tall order for me. though i love href="http://www.rctophobby.com/rc-helicopters.html">rc helicopters, i still knew nothing about the details of the machine. however, after your presentation, i think i have a notice note about it. thanks for share it, good job!

(/member/susan111/)

susan111 (/member/susan111/) 1 year ago Reply (C4R3FHZHIXPVTW7)

Hi, i think u did a great job. at least, i think it's a tall order for me. though i love href="http://www.rctophobby.com/rc-helicopters.html">rc helicopters, i still knew nothing about the details of the machine. however, after your presentation, i think i have a notice note about it. thanks for share it, good job!

(/member/susan111/)

patriots21 (/member/patriots21/) 1 year ago Reply (C5GPJRGHH2VN0F1)

found this blogspot by shane: http://scolton.blogspot.com/2011/08/something-like-flight.html. this particular blog about the control structure really helped me. the entire blog site, matter fact, is a gold mine of information. go check it out if you're also having trouble or would just like to see how amazing shane really is.

(/member/patriots21/)

patriots21 (/member/patriots21/) 1 year ago Reply (C9SS2HCHH2VL3WK)

Hi shane. great work by the way. I've been trying to implement the same control structure in my own quad. My quad is way bigger than yours but I'm using the same IMU. My propellers are 10x4.5. I also use smaller ones which are 8x4.5 for testing. As such, I'm having a hard time determining what value to start my kp and kd constants at. I've been trying to tune for a month now and I haven't had a successful flight yet.

(/member/patriots21/)

hdman1203 (/member/hdman1203/) 1 year ago Reply (CTVZ56AHAUNMHKA)

excellent project and videos. i saw a TED video of this a while back really wanted to do this.

(/member/hdman1203/)

i am new to this so was wondering if there are any books/literature you would suggest

to better how this particular system works as a whole and how each of its parts interact with each other? also, if one wanted the altitude feedback and return to horizontal position, would it be just as easy as rewriting some of the code for the Arduino board or would some extra hardware/electronics such as the gyros be required as well? Thanks!!


(/member/scolton/)

scolton (/member/scolton/) (author) hdman1203
1 year ago Reply (CF7I8MQHB58N3Q6)

I don't know of any books in particular, but many of the links at the bottom of Step 1 have great information on the different components involved in multirotor systems. As written, the code already has "return to horizontal" capability (see Step 13), but if you wanted to add altitude hold, you would need some way of measuring absolute height, such as a barometer or ultrasonic distance sensor.


(/member/danblakemore/)

danblakemore (/member/danblakemore/) 1 year ago Reply (CE2U7MZHABC552B)

I'm surprised you got it so small and it could still carry its own battery. Nice work. I'm working on a similar project with a much larger form factor (Turnigy Talon v1) and we're having massive trouble with vibration on our gyro. I have the picture of your foam mounting but I can't really tell how it's set up. Is the sensor board weighted and set on top of a foam loop, or is the gray foam wrapped around something? Thanks.


(/member/scolton/)

scolton (/member/scolton/) (author) danblakemore
1 year ago Reply (CWHIE3XHABC56J7)

The weighted IMU sat on top of the foam loop.

I also have a Talon v1...it's a great frame. But yeah I haven't had much luck with mounting the controller or IMU directly to it. Some kind of isolation is needed.


(/member/zaphv/)

zaphv (/member/zaphv/) 1 year ago Reply (C7PBEC5H8I43IES)

In the schematic , Obviously where it says DNP is open circuit however where it lists the resistors as 0 is that meant to be a closed circuit. For example R18 is a direct connection from pin 15 to the 3 resistors but there is no cap to ground ?

(/member/scolton/)

scolton (/member/scolton/) (author) zaphv
1 year ago Reply (C2J30GQHABC56IN)

Yes, all DNP and zero-ohm resistors are for configuring the various settings on the TB6588FG. They're either shorts or open-circuits. So you could use a solder bridge instead of an actual zero-ohm resistor.

Alastor136 (/member/Alastor136/)

1 year ago Reply (CJNPT12H742ZR5H)

I was about to ask if you were affiliated with MIT, then I noticed you were testing it in Building 7!

(/member/Alastor196/)

Bill WW (/member/Bill+WW/) 1 year ago Reply (CNG7HQZH67NSEOT)

Congrats on the win!
You provide excellent specs on components.

Bill

(/member/Bill+WW/)

TechNotes (/member/TechNotes/) 1 year ago Reply (CF3WM3DH5CR572Y)

How did you choose your motors?

(/member/TechNotes/)

scolton (/member/scolton/) (author) TechNotes 1 year ago Reply (C3ENTMBH52FXVLH)

For this size quadrotor, there weren't many options. But in general, you would choose the motor and propeller at the same time. There are suggested propellers listed for most motors. Sometimes you can also find data on thrust/RPM/current values with different propellers.

The total maximum thrust should be 2-3x greater than total weight, so that you have plenty of extra thrust available for maneuvering and stability. I think on this quad it was much closer to 1x when I was using the 7.4V battery. Switching to 11.1V made it closer to 2x, which improved the stability a lot.

(/member/scolton/)

TechNotes (/member/TechNotes/) scolton 1 year ago Reply (C2RGHGTH5A36KEK)

Would this battery be a good choice for this quadrotor?

http://www.hobbyking.com/hobbyking/stc

(/member/TechNotes/)

scolton (/member/scolton/) (author) TechNotes 1 year ago Reply (CXJJVU7H52FXVPZ)

Yes, that's the exact one I used.

(/member/scolton/)

TechNotes (/member/TechNotes/) 1 year ago Reply (CJM4CIKH5CR5748)

As of August, 2012, there is MinIMU-9 version 2. Did you use the version 1 or 2, and would the other one work with the quad just as well?

(/member/TechNotes/)

scolton (/member/scolton/) (author) TechNotes 1 year ago Reply (C5NDUSKH52FXVLM)

I'm using the version 1, but I think the version 2 would also work fine.

(/member/scolton/)

WYE_Lance (/member/WYE_Lance/) 2 years ago Reply (C6349POH4AGJDAK)

(/member/WYE_Lance/)

Allow me to be the first to congratulate you on the grand prize win! Well deserved!

scolton (/member/scolton/) (author) WYE_Lance 2 years ago Reply (CLDE9GJH4AGK83G)

(/member/scolton/)

Thank you! I'm glad people like it and I hope there will be swarms of PCB quadrotors in the near future.

francisroan (/member/francisroan/) 2 years ago Reply (C2JD5KZH401UUBO)

(/member/francisroan/)

guys i have an idea...see i wanted to build this but it is too expensive i mean like 300$? ya like ma parents are gonna give me that so i wanted to make the worlds most cheapest quadrocopter!!! and i really mean it ..in this one he used an xbee which is really expensive plus he didnt add the cost to make the transmitter so add another 50 to 100 bucks so total about less than 500$?
so i had abt module laying around so i thought hey why cant we make a bt quadrocopter instead of an r/c?(cheaper u can get two modules for like 15 $ at ebay one for transmitter and another for reciever

most probably ill make this one and maybe make an instructable for u guys

by the way awesome instructable(this) well explained well done!!!

scolton (/member/scolton/) (author) francisroan 2 years ago Reply (CHGYY60H401VBNV)

(/member/scolton/)

The total cost for everything including the ground station transmitter hardware (XBee, XBee Explorer, Game Controller) is about $320. Without the transmitter hardware it's $240. It's definitely not the cheapest project, mostly because I built everything from scratch.

Bluetooth would be cool, but the range isn't as good as the XBee Pro. It would be perfect for indoor use though, and then you don't need an adapter for your computer if it already supports Bluetooth. Or maybe a smartphone-based controller. That would certainly bring the cost down.

tabare.perez (/member/tabare.perez/) 2 years ago Reply (C38YK6AH337JIGV)

(/member/tabare.perez/)

VERY NICE INSTRUCTABLE!!
You can try KiCad Open Source solution instead of Eagle.

Best regards from Uruguay!
TabarÃ©

blanchae (/member/blanchae/)

(/member/blanchae/)

2 years ago Reply (C2QL1PHH2WESWI5)

When two worlds collide, I read your excellent instructable then came across this artist who turned his dead pet cat into a helicopter (http://www.dailymail.co.uk/news/article-2154283/Cats-away-Artist-turns-dead-pet-flying-helicopter-killed-car.html?ITO=1490) (yes, you read right!). Probably the funniest and most morbid thing I read in a while. So if you want to expand your quadrotor in a new direction.....

---

udawatabhimanyu4 (/member/udawatabhimanyu4/)

(/member/udawatabhimanyu4/)

2 years ago Reply (CW9PE9YH2WESUVE)

Really, there should be a category for Super-Awesome-Advanced-Level-Instructables. Also, you could have taken it to an even higher level by including the ATmega328 from the pro mini on the PCB itself and maybe the inertial measurement unit too. Extremely detailed Instructable +10

---

scolton (/member/scolton/) (author) udawatabhimanyu4

(/member/scolton/)

2 years ago Reply (C7CD7UWH2WEVB1M)

Thanks! Putting the ATmega328 on-board is definitely something I want to do in a future revision. Or maybe somebody here will do it first. ;)

I'm not sure about putting the accelerometer and gyro on-board, though, because they wouldn't be isolated from mechanical vibrations.

Geraldes (/member/Geraldes/)

(/member/Geraldes/)

2 years ago Reply (CMIFNONH2WESV2A)

Awesome instructable.
Have you tried using any Kalman filtering?

1-40 of 50 Next » (http://www.instructables.com/id/PCB-Quadrotor-Brushless/?&sort=ACTIVE&limit=40&offset=40#DISCUSS)

Make Comment

About Us

Wh e (/about/)
A vertise/)
C t/contact.jsp)
Jobs (/community/Positions-available-at-Instructables/)
Help (/community?categoryGroup=Help)

Join our newsletter! | enter email | Join!

English

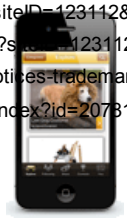
Find Us

Facebook (http://www.facebook.com/instructables)
Youtube (http://www.youtube.com/user/instructablestv)
Twitter (http://www.twitter.com/instructables)
Pinterest (http://www.pinterest.com/instructables)
Google+ (https://plus.google.com/+instructables)
Tumblr (http://instructables.tumblr.com)

Terms of Service (http://usa.autodesk.com/adsk/servlet/item?siteID=123112&id=21959721) |
Privacy Statement (http://usa.autodesk.com/adsk/servlet/item?s 123112 2079) |
Legal Notices & Trademarks (http://usa.autodesk.com/legal-notices-trademarks/) | Mobile Site (http://m.instructables.com)

Resources

For Teachers (/teachers/)
Artists in Residence (/group/air/)
Gift Pro Account (/account/give?sourcea=footer)
Forums (/community/)
Answers (/tag/type-question/?sort=RECENT)
Sitemap (/sitemap/)



Mobile

Download our new apps for iOS, Android and Windows 8!

Android (https://play.google.com/store/apps/details?id=com.adsk.instructables)

iOS (https://itunes.apple.com/app/instructables/id586765571)

Windows (http://apps.microsoft.com/windows/en-us/app/7afc8194-c771-441a-9590-54250d6a8300)

(http://usa.autodesk.com/adsk/servlet/pc/index?id=20731458&siteID=123112)
© 2014 Autodesk, Inc.

Go Pro Today » (/account/gopro?sourcea=footer)

We're Hiring! » (/community/Positions-available-at-Instructables/)