



The Making of a DIY Brushless Gimbal With Arduino



by ArduinoDeXXX

This is a story of my third project with cheap gyro and Arduino.

After the earlier two projects, **Easy Inverted Pendulum** and **3D Calligraphy**, I have thought of making **Camera Gimbal with Arduino**. Then I had three policies for the project.

- Equipped with standard camera bigger than **GoPro**
- Simple as possible with **no special** device nor kit
- Bootstrap: Refer to fewer websites as possible and finish with **no imitation**

At the start of the project, I had intention of making **2-axis gimbal with Servo Motors**. But it has been made clear that it is difficult to see adequate performance using Servo Motor. Consequently a **3-axis gimbal with Brushless Motors and Arduino** has been made. **INTRODUCTION VIDEO** below shows the effectiveness of the DIY gimbal. In the video, pretty noise would be heard suddenly. It is left without any reduction to show strengths and weaknesses of the gimbal. Arbitrary film editing is avoided.

:

[INTRODUCTION VIDEO]

[//www.youtube.com/embed/3nkO4DhVGr0](https://www.youtube.com/embed/3nkO4DhVGr0)

:

It seems that the DIY 3-axis gimbal in this video provides **enough compensation** for practical use. But it was not easy to make it of this level. For example, I have met with **difficulties** below.

- Controlling **Brushless Motor** adequately with Arduino alone
- Getting a precision **frame** of gimbal by using rather poor tool
- Resolving the disturbance of **SPI** interface between gyros and Arduino
- Settling frame **chattering**
- Adding **intended and smoothPAN** or **TILT** to good compensation

The first problem above is a matter of **programming**. In contrast the second one is a pure **hardware** matter. And

the remaining ones are found in **intersection** of both matters.

Also **four problems** except for the first one have been encountered when **the second axis** was added to the single axis gimbal. It means that the **core of the program** of multi-axis gimbal here is not different from the single axis one. But the good frame for multi-axis one requires much more efforts to be gotten than single axis one.

:

Here "*How to make the DIY **single axis Brushless Gimbal with Arduino***" is told in detail in a later step. But it is rather a special topic to tell the whole story about the **process** of development a DIY camera gimbal with Arduino of level high enough for practical use. The story is as follows.

- An important point to make a gimbal with gyro
- DIY single axis gimbal with cheap Brush DC Motor
- DIY single axis gimbal with Servo Motor
- DIY single axis gimbal with Brushless DC Motor
- Multi-axis controlled
- Effectiveness of the DIY gimbal
- Unresolved problems for improvement

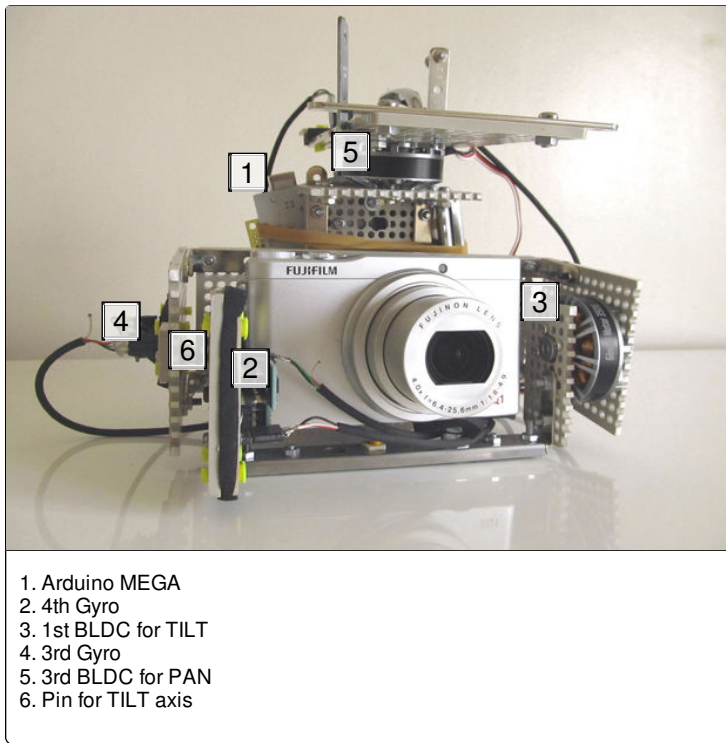
"How to ..." is told in **Step 6 to 10** mostly. And the minimum work in it might be able to be done just referring to content in **Step 10** after watching **DCUMENTARY (12)** in **Step 9**. On the other hand the performance of the DIY 3-axis gimbal would be seen approximately after watching **INTRODUCTION VIDEO** above, **DCUMENTARY (15)** in **Step 12** and **DCUMENTARY (17)** in **Step 13**.

:

By the way, another project entitled "**Brushless Gimbal with Arduino**" has been posted to Instructables in last year. It provided me the "***good resource for a starting point***". **But its approach contrasts with mine.** The difference between them would be told in **Step 5**.

:

I hope the attached videos and pictures would *compensate* for my poor English. The Japanese version of this instructable has three more steps additionally.



1. Arduino MEGA
2. 4th Gyro
3. 1st BLDC for TILT
4. 3rd Gyro
5. 3rd BLDC for PAN
6. Pin for TILT axis

Step 1: An Important Matter—Which Side Is Gyro Attached To?

Motion of a rigid body in space can be described with two independent elements; **rotation** and **parallel translation**. The former can be cancelled out or compensated by an instrument called **gimbal**. A camera mounted on the innermost gimbal could be free from rotary motion when the support of the gimbal moved, even though it might not be free from parallel shift. Hence for the gimbal driven by motor, **angle** (or **posture**) of camera becomes the important information to keep it still.

We have several alternatives to estimate the angle of object in motion. Here **cheap gyro module(s)** is/are used. The angular velocity of the object can be measured by gyro. And Arduino translates it into estimated degree of angle shift of the object. Now we have the next alternatives for the new matter which side of gimbal the gyro is attached to.

(1) Attaching gyro to the "camera side" (attaching it to the camera mount)

(2) Attaching gyro to the "support side" (attaching it to the handle base)

:

The most different point between them is whether the **feedback of the intentional rotation** of the motor controlling the gimbal is gotten or not. When attaching gyro to **(1) camera side**, intentional motion is not distinguished from un-intentional one and just the sum of them is measured. On the other hand when attaching gyro to **(2) support side**, only the un-intentional rotation is measured but intentional one is not. Hence they have good and bad points respectively as below.

(1) Attaching gyro to the **camera side** (attaching it to the **top** (or **rotor**) of the motor)

- Strategy: When the camera rotates a little bit, drive the motor to cancel out it immediately.
- Merits: It is not needed to estimate the angle correctly. (So it is not needed to know how much the motor rotates its spindle.)
- Demerits: The camera **vibrates** easily unless the motor is controlled adequately.

(2) Attaching gyro to the **support side** (attaching it to the **base** (or stator) of the motor)

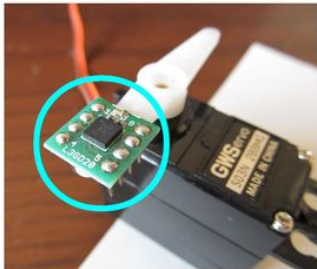
- Strategy: When the support rotates, estimate the degree and rotate motor as much as that to the reverse direction correctly.
- Merits: Vibration caused by the feedback of motor rotation never occurs.
- Demerits: The **correct estimates are needed for both rotation** of the support and camera.

:

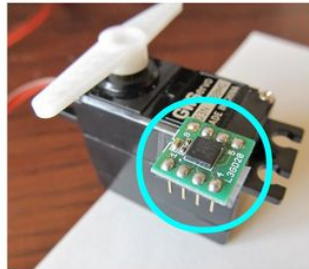
These differences are important for **motor choice**. When the gyro is attached to the **(2) support side**, both degrees of angles of the top and the base of the motor have to be estimated correctly. Therefore **Servo** or **Stepper** Motor is likely to be used, which could control the angle directly.

On the other hand, when the gyro is attached to the **(1) camera side**, just the rotation of the top of the motor should be restricted from the fixed viewpoint at somewhere outside. Then the degree of this rotation does not have to be estimated necessarily. So the **cheap Brush DC Motor** might work.

Now it appears that the **(1) camera side** is better to attach gyro to, from point of view of either freedom of motor choice or less efforts needed to estimate the correct degrees of angles.



(1) Gyro attached to the top of motor
On the camera mount



(2) Gyro attached to the base of motor
On the handle of gimbal

Step 2: Can Be Cheap Brush DC Motor Applied to Gimbal?

Some cheap Brush DC Motor might work for the gimbal when the gyro is attached to the **(1) camera side**. But the spindle of this kind of motor would turn at higher speed for gimbal. Therefore gear or pulley is needed to **reduce speed** properly.

Here a simple **inverted pendulum** robot ever made is used like a **single axis gimbal** to see whether Brush DC Motor could be applied to camera gimbal.

With a modified program for gimbal this robot can give rough compensation against the rotation of its support. Though it is not so bad, **large vibration** occurs, which cannot be resolved by the standard countermeasure. You can see detail in **DOCUMENTARY (1)** below.

:

be weakened. (See **DOCUMENTARY (2)** below.) But the gimbal is left uncontrolled while gears do not mesh. Hence a precision reduce gears such as no backlash is felt is needed to apply cheap Brush DC Motor to camera gimbal. But it seems too hard to get such gears in this project.

:

[DOCUMENTARY (1)] Single Axis Gimbal with Brush DC Motor

[//www.youtube.com/embed/OAtdiOxKde4](http://www.youtube.com/embed/OAtdiOxKde4)

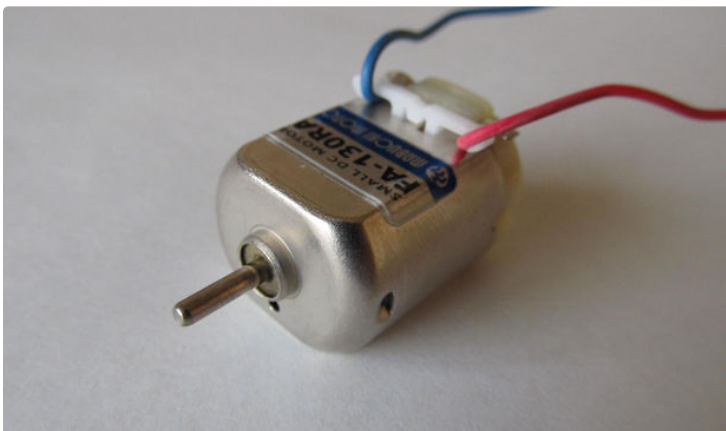
:

Backlash of the reduce gears causes this **persistent vibration**. It is needed to know at every moment whether gears mesh or not to resolve it. But it is not easy to know it in real time.

As lowering sensor sensitivity this vibration is able to

[DOCUMENTARY (2)] Modified Single Axis Gimbal with Brush DC Motor

[//www.youtube.com/embed/-WUvk5kHrd0](http://www.youtube.com/embed/-WUvk5kHrd0)



Step 3: Making Single Axis Servo Gimbal (Part 1)

In this step and the next, Servo Motor is tried out for a single axis camera gimbal.

In **Step 1** it is described that Servo Motor could work in either case where gyro is attached to **(1) camera side** or **(2) support side** of gimbal. In this step gyro is attached to the **(1) camera side**. Though it is not required to know the correct degrees of angles in this case necessarily, two alternative strategies were adopted respectively. The results observed are as follows.

- The strategy, "turning motor as much as estimated degree of rotation of the support of gimbal to the reverse direction," **cannot** cancel out the rotation **enough**.
- The other strategy, "turning motor until cancelling out camera rotation," results in **delayed compensation** or **vibration** of camera mount.
- Therefore it is not easy to decide how much Servo turns.

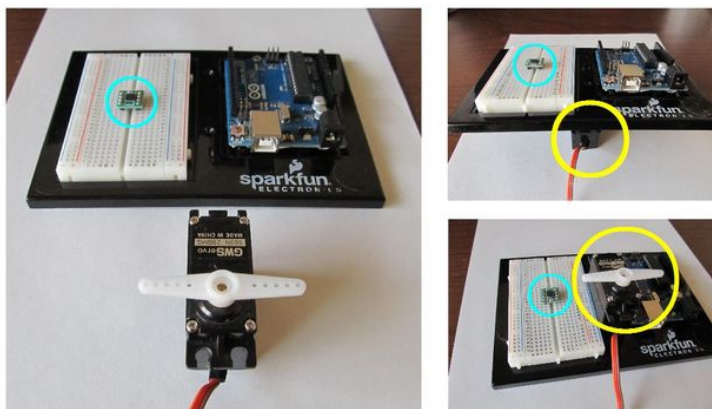
:

The former strategy above is prepared for the gyro attached to the **(2) support side**. Though it is thought to work as well for the gyro attached to the **(1) camera side** intuitively, it compensates only the half of rotation on reflection.

On the other hand the latter strategy is prepared for the case where gyro is attached to **(1) camera side**. Here Servo Motor receives directly the degree of angle to turn. So it is **not easy** to control angular velocity or acceleration.

By changing the degree to turn gradually, we can control angular velocity to some extent. But we would meet with vibration in higher speed or delayed compensation in lower speed easily.

The advantage of Servo Motor is correct turning with the degree received. Therefore when using Servo Motor it should be better that the gyro is attached to the **(2) support side** from the viewpoint of efficiency of control, where the vibration caused by feedback could not be observed.



Step 4: Making Single Axis Servo Gimbal (Part 2)

In the previous step we have learned that it is not better attaching gyro to **(1) camera side** when using Servo

Motor. So Servo Motor is tried out again with attaching gyro to (2) support side. The results observed are as follows.

- Servo Motor is **not good** where **quick acceleration or deceleration** is needed
- Especially it needs **more time upon deceleration** to stop
- It is **difficult to change** acceleration or deceleration of each Servo Motor

:

The figure above shows the change in the angular velocity of a Servo Motor from start to finish. There four different angles with 8, 15, 30 and 60 degrees are given and four lines in this figure represent these angles respectively. The slope of each line shows angular acceleration. We can know acceleration from start and deceleration to stop. So the figure tells us as follows.

- The acceleration from start to maximum speed is almost constant and common to all angles given.
- On the other hand the deceleration to stop is dropping stepwise but the pattern is similar to each other angle given.
- Therefore the time required to stop from maximum speed is about two or three times longer than the time needed to reach maximum speed from start.
- When the given angle is less than 15 degrees, the maximum speed of this motor is never reached from start to finish.

:

The first three points above mean that Servo Motor executes stopping motion more carefully than to starting. And the last point above means that Servo Motor cannot respond quickly enough to irregular change of angle. So it tells us that **rotating speed (as maximum speed in steady turning), one of the most popular performance measure of Servo Motor, is not so critical in this project.**

The careful deceleration of Servo Motor to stop is thought to intend to avoid risk of vibration. But this careful deceleration spoils quick response needed for camera gimbal. We can see both delayed starting and dropping deceleration to stopping of Servo Motor in the **DOCUMENTARY (3)**.

:

[DOCUMENTARY (3)] Testing Tracking Speed of Servo Motor

[//www.youtube.com/embed/lqgFQ1HsnRA](https://www.youtube.com/embed/lqgFQ1HsnRA)

:

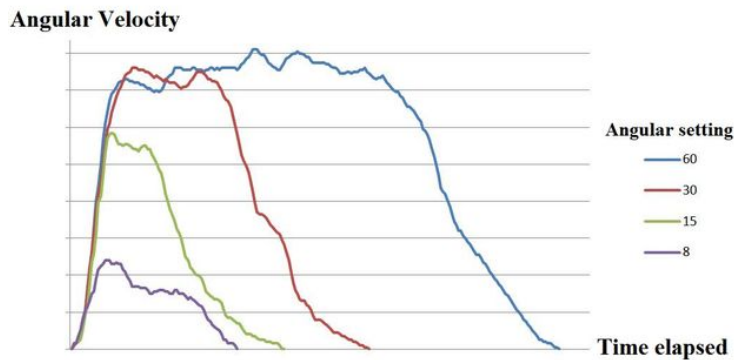
Camera gimbal must compensate against irregular rotation of its support. Hence the motor driving it is required not only higher maximum speed but also enough acceleration and deceleration for **quick tracking**.

From this point of view **Servo Motor** is thought to be too much careful for avoiding the risk of vibration **at the expense of acceleration and deceleration**. Though the extent of this trade-off depends on each model of Servo Motor, none provides enough compensation which I have tested. In the **DOCUMENTARY (4)** below the best Servo Motor among them is used, which has smaller body, higher speed and quicker response. But we **cannot recognize** that it works adequately in the video.

:

[DOCUMENTARY (4)] The Final Test of Single Axis Servo Gimbal

[//www.youtube.com/embed/Tf14-M0anfg](https://www.youtube.com/embed/Tf14-M0anfg)



Step 5: The Articles of DIY Brushless Gimbal in Web

In the previous step it has been made clear that Servo Motor would hardly provide enough compensation for gimbal. So **Brushless Motor** is considered to be applied to gimbal instead of Servo Motor.

But I had never seen Brushless motor and no special knowledge about it at that time. So I bought a relatively inexpensive one (\$22.75) for a research. But I found no paper like manual but a motor in the package. Then I got little information added to use it.

:

Now that things had come to this pass, I referred to websites to search useful information to use Brushless Motor. I found a considerable number of articles and videos about **DIY with Brushless Motor**. And I knew that **special device** named **ESC** or **Gimbal Controller** is used to drive each Brushless Motor in general.

On the other hand no special device would be used in this project as described in **Introduction**. Only Arduino should be used to control gimbal. Hence I focused on **a rare article** featured in Instructables in which neither **ESC** nor **Gimbal Controller** is used. Its title "**Brushless Gimbal with Arduino**" hits the mark of my project. But it is a pity that the gimbal shown in it does not seem to work adequately so the authors say, "*we never got the gimbal we*

were looking for”.

However, their article introduces some **other articles** consulted. And **one of them** shows how to rotate Brushless Motor using Arduino (by eLABZ). **It gave my project the starting point at that time**. The detail of the consulted article will be described in the next step. Here **the first article referring to it is mentioned**.

:

This article in Instructables describes an earlier project executed by two students belonging to a university in US. There seems to be **three points in common between their project and mine**.

- Both of us want to make a camera gimbal with Brushless Motor work well without special *Gimbal Controller*.
- In short we would use only Arduino to control gimbal.
- But we cannot find no similar project which meets either of these two requirements except ours.

On the other hand **their project is different from mine** as follows.

- They use **GoPro** as mounted camera.
- They use **ready-made set** on the market as frame and motors
- They use the program provided in **other website** to rotate Brushless Motor
- They use **a special device** as gyro module (a *MPU6050* carrier)
- They use the program provided by **others** to exploit this module

:

Both of us share a goal (Driving Brushless Motor Adequately by Using Arduino Alone without Special Controller) and a situation recognition (Finding No Article Achieving This Goal for Camera Gimbal). And they use rather positively both existing programs provided by others and ready-made set sold on the market.

This approach seems like a strategy such as "**Achieving a Unique Objective with Help of Others**" or "**Getting a General Function with Special Materials or Methodologies**". At that time I brought the term in my mind impressively, ***Horizontal Divisions of Labor*** (*).

(*) I had heard that *Horizontal Divisions of Labor* harass Japanese Industry which has advantage in ***Vertical Integration***. The third policy described in **Introduction**, "Bootstrap: Refer to less websites and finish with no imitation", is felt nearer *Vertical Integration* than *Horizontal Divisions of Labor*.

:

Back to topic, in the article in Instructables introduced here a special gyro module is used, which has **MPU6050** sensor. This sensor has a special and very powerful function named **DMP (Digital Motion Processor)**. This function provides not only greater noise reduction than standard gyro sensor but also **estimate of 3D rotary coordinate system** (*1) in real time (*2).

(*1) It is not a rotation around a fixed axis on gyro but a rotation of 3D coordinate system in the fixed system.

(*2) This is not an easy calculation. Hence a famous freeware working in PC, *Processing*, is used not in real time in my earlier project.

Though the sensor with *DMP* is amazing, it would be "**special device**" at the present moment. Therefore I decided not using the module with this sensor in this project. By the way a popular motor driver, **L298**, is used in this project, which is employed in the article above. I am grateful to the authors of this article.



Step 6: Rotating Brushless Motor Using Existing Programs

Going back to my project, "*How to ...*" is described here for the first time in this instructable.

In the article mentioned in the previous step authors rotate Brushless Motor by using Arduino. There they refer to an earlier article written by eLABZ. And they use the program provided in it. In the consulted article the author **eLABZ** explains how to make a *Stroboscope* with Brushless Motor took out of an old DVD drive.

As far as I know (*), **this is the only article that contains the full array of information to rotate Brushless Motor by using Arduino alone**. This article is constructed of 3 sections. (*I found another video right now. It has just the minimum information.)

1. Learning fundamental knowledge of (3-phase) Brushless Motor
2. Rotating Brushless Motor by using Arduino
3. Putting materials together to get Stroboscope

:

In the second section above two alternative programs for Arduino are provided to rotate Brushless Motor. I read them and its commentary with an illustration in the article. And I understood that I should just supply **three AC voltages** which phases are shifted by 120 degrees to **the three wires** with Brushless Motor in no particular order. Indeed the Brushless Motor I have bought rotates easily with Arduino and an imitation program, which contains the only lines to supply these three voltages to the motor. The **DOCUMENTARY (5)** shows the Brushless Motor rotates with Arduino only.

:

[DOCUMENTARY (5)] Rotating 3-Phase Brushless Motor with Arduino and AC Square Wave

//www.youtube.com/embed/rzWfxoOVmjU

:

Here I explain how to rotate Brushless Motor by using Arduino UNO. You need to get a **3-Phase Brushless Motor**. I think it has three wires.

1. Connect these wires to pin 9, 10 and 11 on Arduino UNO in no particular order
2. Download the two **pdf files** attached at the end of this step to your PC
3. Open them in proper reader application
4. Copy the whole of text in either of them
5. Paste the copy to IDE for Arduino and correct misprints
6. Upload corrected program to Arduino using IDE

Each of these programs is based on the original one provided by **eLABZ**. A high-torque-type motor for gimbal would rotate immediately at upload finished. On the other hand some high-speed-type for RC model might not rotate with the power from Arduino. Then some outer battery and motor driver IC should be used. In the **DOCUMENTARY (6)** below outer batteries and driver IC are used. In the former half of this video a high-torque-type is rotated and in the latter a high-speed-type is rotated respectively.

:

[DOCUMENTARY (6)] Rotating Two Types of Brushless Motor with Arduino, Outer Batteries and Sine Wave AC

//www.youtube.com/embed/55_e6Mnpl88



Arduino with the eLABZ's sketch drives BLDC.

<http://www.instructable...>

Download

Step 7: Testing a Single Axis Brushless Gimbal Using Existing Program

In the previous step we saw that simple programs uploaded to Arduino can rotate Brushless Motor easily. These programs **imitate** the core of existing programs written by **eLABZ**. The uploaded programs can be seen once downloading the pdf files attached to the end of the previous step. We can control rotational speed of motor to some extent by changing the value of the variable *motorDelayActual* in line 1 in these programs.

Here a **test bench** as a **single axis Brushless Gimbal with Arduino** is made. And the program for this test bench is prepared also based on one used in the previous step. Please note that the **rotational angle of motor is not** measured as well as the case with a cheap Brush DC Motor in **Step 2**. Hence gyro should be attached to the **(1) camera side** and some proper countermeasure is required to prevent **vibration**. The **DOCUMENTARY (7)** below shows the performance of this test bench controlled by the program with standard countermeasure to vibration.

:

[DOCUMENTARY (7)] Single Axis Brushless Gimbal Using 3-Phase Sine Wave AC

//www.youtube.com/embed/Kxeb1wU57XM

:

The test bench can compensate the rotation of its support (or base) roughly in the video above. But some problems are shown as follows.

- **Delayed** starting for compensation
- Unsteady rotation under compensating (**pulsatory** motion)
- Some **vibration** remaining at stopping

So the program uploaded undergoes repeated improvements and tests to resolve these problems. The **DOCUMENTARY (8)** below shows the performance of the test bench controlled by the finally improved program.

:

[DOCUMENTARY (8)] Improved Brushless Gimbal Using 3-Phase Sine Wave AC

:

The improved program can weaken the problems above at some extent. But it could not seem to work well for the camera gimbal. Also the gimbal with it could **hardly work better than Servo Gimbal in Step 4.**



Step 8: Copernican Revolution and Brushless As Servo

In **Step 5** an article in Instructables is introduced, which describes a challenge to get a 2-axis Brushless Gimbal controlled by Arduino. This challenge is supported by an existing program provided by **eLABZ**. But as authors recognizing, it is a pity that the gimbal gotten there does not seem to work adequately.

In the previous step **I imitated the same original program** provided by **eLABZ** and applied it to the single axis Brushless Gimbal. But it has **never worked well** in spite of executing **any improvements** I could think of.

Indeed it is written in the article about the original program that *"this circuit was designed for a rather simple application (...) – the load is so light (...). If (...) your motor does not have **Hall-effect sensors** (many BLDC motors do), then this simplified circuit is **not suitable** for your application."*

:

At the end of my resources, as dejectedly gazing at the gimbal working, it seemed to be quite unreliable as yet. I almost gave up. Then an idea hit me, *"The most important here is **not to rotate the motor quickly or smoothly but to keep it stopping tightly at the point**"*.

It was felt a *revolutionary shift of the goal* for me. So I drew a figure simplifying Brushless Motor, and based on it **I calculated the best operation to maximize the stopping power of motor at arbitrary angle** (*1). Next, I prepared a whole new program to execute the operation and uploaded it to Arduino. Then I saw the Brushless Motor was kept stopping at the objective angle very strongly as never seen (*2).

And changing the angle given, the Brushless Motor rotated immediately and stopped at the changed angle tightly again.

(*1) I learned fundamental knowledge of Brushless Motor by referring to an article in the site of Ijima-san (in Japanese). Though it is similar to the article by **eLABZ** in **Step 6**, an additional topic is mentioned about **alternative connections, Y or Delta**. Every Brushless Motor used in this project has **Y connection**. (At first I had the wrong idea that each motor had Delta connection. But later I found Y connection in the motor disassembled. I appreciate Ramirez_MecaUPQ's question.)

(*2) The earliest calculation was based on the assignment of slots (coils) explained in the site of Ijima-san. But no torque was gotten. After research I found a manual to build Brushless Motor kit (in Japanese). It taught me that there are alternatives among Y or Delta connections.

:

The new operation and program could **make a Brushless Motor work like as a Servo Motor**. So they were modified to be able to cooperate with gyro. The **DOCUMENTARY (9)** shows they work well on the test bench and suggests much better performance could be expected than Servo Gimbal.

:

[DOCUMENTARY (9)] Brushless Motor Made Work Like as Servo Motor

[//www.youtube.com/embed/6cYyLiSGuHE](https://www.youtube.com/embed/6cYyLiSGuHE)



Step 9: Making a Precision Single Axis Brushless Gimbal With Arduino

The **new operation** introduced in the previous step would make it possible to get a Brushless Gimbal with Arduino of practical use. Now **a Brushless Motor can work like as a Servo.**

In this operation gyro can be attached to the **(2) support side**. The test bench prepared in **Step 7** becomes a new single axis Brushless Gimbal under this operation. In the **DOCUMENTARY (10)** below the new gimbal equipped with a camera is tested.

:

[DOCUMENTARY (10)] The First Brushless Gimbal Operated by the New Program

[//www.youtube.com/embed/IB_O69B8D-c](https://www.youtube.com/embed/IB_O69B8D-c)

:

In the video above we can see the performance of the new single axis Brushless Gimbal from two viewpoints like **DOCUMENTARY (4)** in **Step 4** with Servo Gimbal. Now we can recognize that the new Brushless Gimbal works better than Servo Gimbal in Step 4 though the new one has some rough compensations as yet. Fortunately we have still several rooms for improvement such as value selection for parameter set in the program.

Also we can get some additional data by attaching another gyro to the (1) camera side of the gimbal. Indeed comparing data from both gyros I have understood there are three matters remaining.

- **Time lag** at the start or the end of compensation
- Attenuating **shake** at the start or the end of compensation
- Angular **shift** after compensation: Uncompensated shift remaining

Among these three matters the second would be resolved by the standard PD control using the added gyro in the

operation newly. Also the third is thought to be resolved by giving proper values for parameter set or using the added gyro. In the **DOCUMENTARY (11)** below we can see the effectiveness of exploiting the added gyro in the operation.

:

[DOCUMENTARY (11)] Effectiveness of Exploiting Added Gyro in the New Operation

//www.youtube.com/embed/dy7rGhf7-IM

:

In the video above we can recognize that each of the three matters seen in the previous video **(10)** can be much improved respectively by exploiting the gyro added. And we have still other rooms to improve.

For example the power supplied to the motor is about **5V** in these videos above though the recommended voltage is **11 to 15V**. In the **DOCUMENTARY (12)** below four more AA batteries are added to **supply enough power to the motor** and the values of parameter set are adjusted.

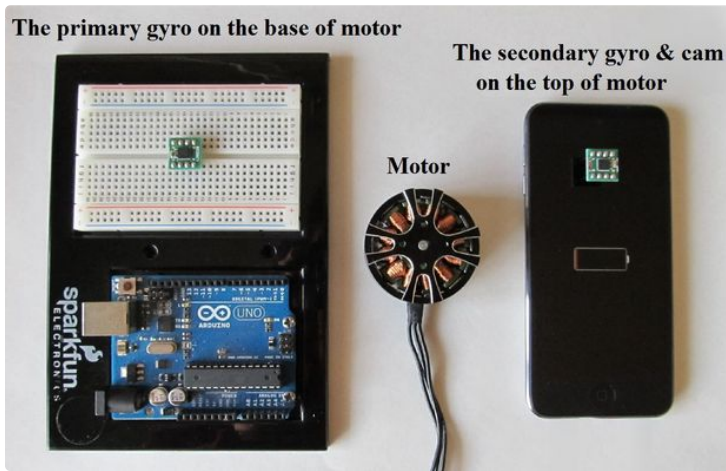
:

[DOCUMENTARY (12)] Effectiveness of Adequate Power Supply to Motor

//www.youtube.com/embed/ty4WfmcjwUc

:

In the video above we can see that the first problem, "Time lag at the start or the end of compensation", is weakened so much under the enough power supplied. Now we should move from the stage using the test bench as single axis gimbal to the next stage making 2-axis gimbal of practical use.



Step 10: How to Make the Single Axis Brushless Gimbal With Arduino

Before moving the next stage where 2-axis gimbal is made, I show here how to get the **DIY single axis Brushless Gimbal with Arduino** which works in the **DOCUMENTARY (12)** in the previous step. To avoid the long recipe I suppose readers here have enough experience with Arduino and gyro such as my earlier project.

:

[Materials]

- 3-phase Brushless Motor for gimbal (x1): Limited to **14-pole, 12-slot** (Though both Y and Delta connection are acceptable, assignment of slots have to be same as the figure shown in this manual ↓)
- Arduino UNO (x1)
- **L3GD20 gyro chip** carriers (x2): SPI interface with 4 lines (*1)
- AA batteries (x4-8)
- **L298 motor driver** (x1)
- Plates (x2): For the support of gimbal and the camera mount
- Others: Popular stuff for work with Arduino

(*1) I used Akizuki's carrier here. I know another L3GD20 carrier produced by Pololu. The latter is available also **outside Japan**. If it were used instead of the former, the picture with schematic above should be viewed.

:

[Program (or Sketch)]

- Download the pdf file attached at the end of this step
- Open it in a proper reader application
- Copy the whole of text in it
- Paste the copy to IDE for Arduino and correct misprints
- Upload corrected program to Arduino using IDE

- Though this sketch is optimized for Delta connection, it is applicable for Y connection
- The sign of output of the secondary gyro at the **(1) camera side** depends on the manner to attach it
- If the gimbal does not stop nor compensate but rotates the wrong direction (*1), try the last point of **"Wiring"** below first
- If the gimbal still does not work well, the **two lines** commented out in this program should be made effective instead of commenting out the alternative **two lines, A and B**
- If the gimbal still does not work well, try the last point of **"Wiring"** below again

:

[Wiring]

- See the **figure** at the top of this step and follow it
- Three wires with Brushless Motor are connected to pin 3, 5 and 6 on Arduino UNO in arbitrary order
- If the gimbal does not stop nor compensate but rotates the wrong direction (*2), change this connection appropriately

(*1, *2) To make the gimbal work well, both of **outputs of gyros** and **wiring** have to be set correct. Sometimes the output of gyro might become **abnormal** in spite of correct setting. Then cut the power off to Arduino and gyros once.

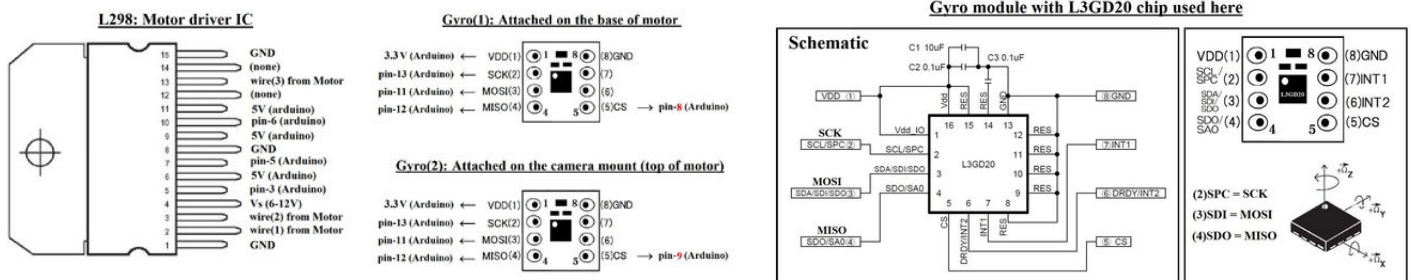
:

[Calibration]

- Reset Arduino
- Don't touch the gimbal for 6-10 seconds: Keep both gyros still

:

The rough flow of the program in the attached pdf file below could be understood by seeing it on a block-by-block basis. On the other hand the details of some blocks might be not so clear. It needs much more steps to explain them. If many requests were gotten in future they might be told in another story.



<http://www.instructable...>

Download

Step 11: Making the 2-Axis Brushless Gimbal With Arduino

We have gotten the single axis Brushless Gimbal with Arduino on a test bench, which works as expected at the start of this project. Now the multi-axis gimbal of practical use should be made. Here **2-axis Brushless Gimbal with Arduino** is made and tested.

It compensates un-intended rotation around two axes: **TILT** (looking up/down) and **ROLL** (rotating horizontal line). It needs two motors and six pins with PWM. So in this step a richer Arduino board, **MEGA**, is used instead of UNO. The main materials to build the frame of this gimbal are plastic plates and stainless steel brackets. The former plate is a goods for hobby named "**Universal Plate**". It has many regular pinholes and needs no special tool to be cut and so on.

:

The next two points take much time to build the frame.

- Hunting good material for camera mount
- Installing a pair of pin and pinhole for the **TILT** axis

In the former, some steel flat bar with regular pinholes is looked for, which is suitable both to **the center of gravity of the camera** and to the existing pinholes on the frame. On the other hand the latter work is required to be equipped with a camera bigger than **GoPro**.

Generally a Brushless Motor works as a good pair of pin and pinhole on arbitrary axis. It supports a cantilever by itself. So an ultra-lightweight camera like **GoPro** could be mounted on a cantilever with few problems. But cantilever could not support a bigger camera well. Hence **a pair of pin and pinhole** should be installed on the axis at the opposite side of the motor across the camera. The pin, pinhole and the motor should provide a good pair of fulcrums on the **TILT** axis to rotate camera smoothly without little backlash.

:

Once the frame is built, the 2-axis Brushless Gimbal with an augmented program for two axes becomes available to test. This **naive gimbal** has **two problems** as follows. The detail and the cause of them, and the countermeasures to these problems are described in the final step.

- Chattering of the frame
- Biased output from gyro to Arduino: Disturbance on SPI interface

Most of electrical components of the gimbal made here are plugged into a solderless breadboard and connected to Arduino, gyro or motors by jumper wires as yet.

In the **DOCUMENTARY (13)** below we can see the 2-axis gimbal tested here is working, which is given some countermeasures to the problems above and allowed intentional **TILTing** additionally. In this video the gimbal seems to work well.

:

[DOCUMENTARY (13)] 2-Axis Brushless Gimbal with Arduino Working (1)

[//www.youtube.com/embed/bKCKZnVS_FI](http://www.youtube.com/embed/bKCKZnVS_FI)

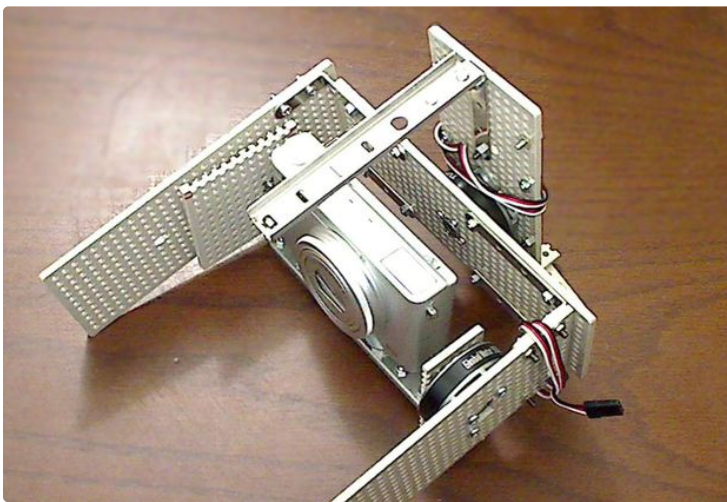
:

On the other hand, in the **DOCUMENTARY (14)** below, we can observe not only the gimbal working but also film shot by the camera on the gimbal in one picture. This video shows that there remains some room to improve for the compensation to **ROLL** (rotating horizontal line).

:

[DOCUMENTARY (14)] 2-Axis Brushless Gimbal with Arduino Working (2)

[//www.youtube.com/embed/FmXjmh3nzUo](http://www.youtube.com/embed/FmXjmh3nzUo)



Step 12: Making the 3-Axis Brushless Gimbal With Arduino

The two problems met in making the 2-axis gimbal in the previous step, Gimbal Chattering and SPI Disturbance, are able to be weakened by some conventional symptomatic treatments. Their details are described in the final step. And the room to improve observed in **DCUMENTARY (14)** could be resolved or reduced by adjusting values of parameter set in the program.

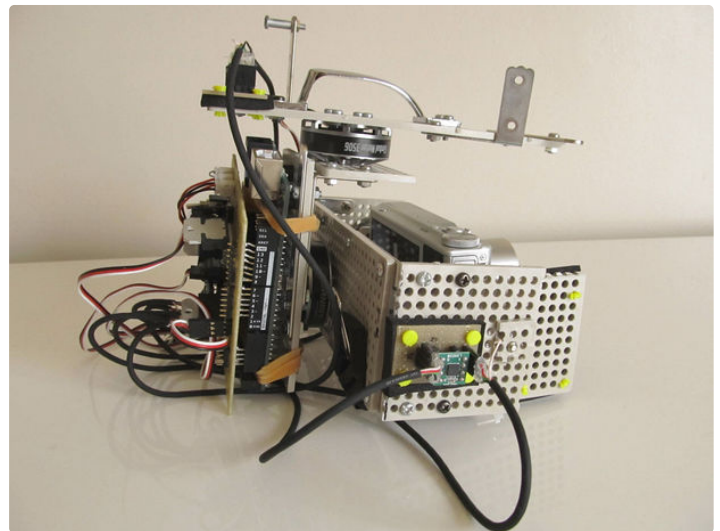
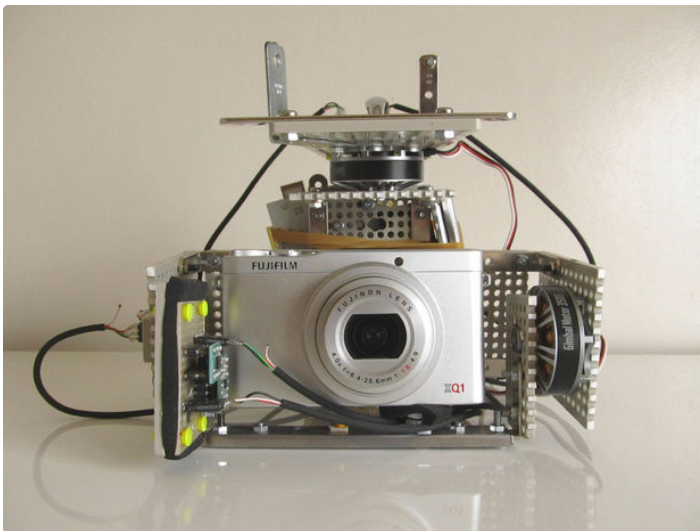
So here, the third Brushless Motor is attached additionally to the top of the frame of the gimbal, which could compensate un-intentional rotation around the axis for PAN (looking at right/left) newly. Next, electrical components are moved from a breadboard to a **DIY Shield for Arduino MEGA**.

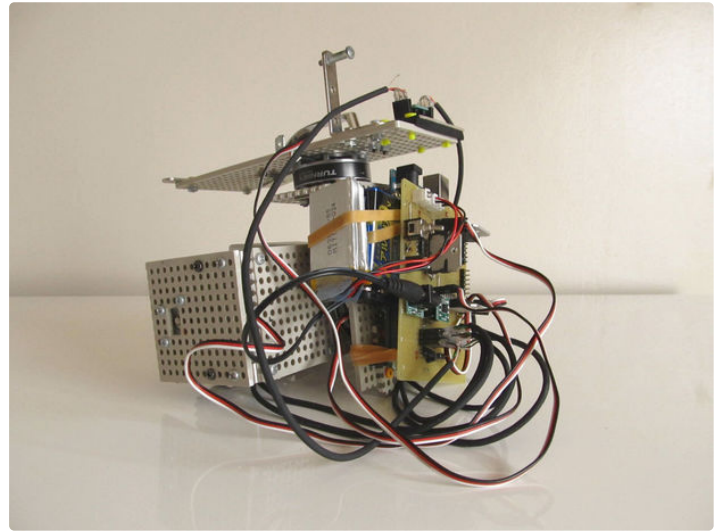
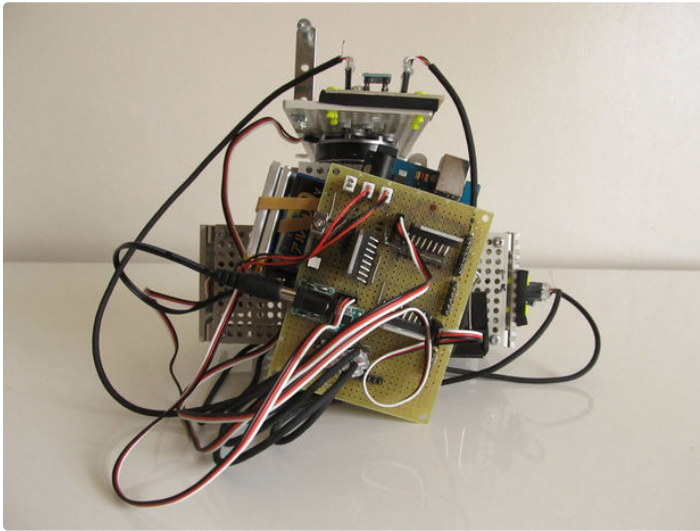
At the same time, the program is augmented again to compensate the un-intentional **PAN** and to allow the intentional PANing. And the values of parameter set in the program are adjusted to reduce the room for improvement observed in **DCUMENTARY (14)**. Now the DIY 3-axis Brushless Gimbal with Arduino for practical use has been gotten. Its performance can be seen in the **DCUMENTARY (15)**.

[DCUMENTARY (15)] 3-axis Brushless Gimbal with Arduino Working

[//www.youtube.com/embed/0wAiGEn1WRo](https://www.youtube.com/embed/0wAiGEn1WRo)

In the video above the camera mounted on the 3-axis gimbal shoots itself **in front of a mirror**. We can see both the gimbal working and the stability of mounted camera. As comparing this video with the DCUMENTARY (14) in the previous step, it is clear that the compensation for un-intentional ROLL is much improved in this step. This is the moment to go out with the gimbal and shoot various scenes to see its performance.





Step 13: Shooting With DIY 3-Axis Brushless Gimbal

We can see new matters in outdoor shooting, which are met with hardly in indoor testing. Therefore the values of parameter set are needed to be adjusted repeatedly again by shooting after shooting. We can see the films shot for this adjusting in the **DOCUMENTARY (16)** below. There we should remember that a gimbal cancels out some rotation but it does not compensate any parallel shifts. Hence the vertical (up-and-down) motion in walking is not subject to compensation in the videos below.

:

[SAMPLE (2)] Up and Down Stairs (From Running to Quick Steps: Uncut)

[//www.youtube.com/embed/H3c5qk_TF5E](https://www.youtube.com/embed/H3c5qk_TF5E)

:

[MAKING VIDEO (16)] Films Shot in Outdoor for Fine Adjusting

[SAMPLE (2-a)] Up and Down Stairs (with Arduino DUE instead MEGA)

:

[//www.youtube.com/embed/m6spew4GQPQ](https://www.youtube.com/embed/m6spew4GQPQ)

[SAMPLE (3)] Walking in a Park (Flowers and Persons)

:

In the adjusting above, the main issue is the adequate compensation for the un-intentional ROLL. Watching **a horizontal line in distant view** in the video above, the degree of its rotational stability is different between scenes (shot under different values

:

[//www.youtube.com/embed/s9Slrp3ic3w](https://www.youtube.com/embed/s9Slrp3ic3w)

for parameter set).

The **DOCUMENTARY (17)** below is a film shot in the adjusting also. It is good to see the performance of the gimbal adjusting here, because it has more stable compositions where the motion of horizontal line is observed well and it has a scene of battery is dead suddenly.

:

[DOCUMENTARY (17)] Film Shot in Outdoor for Fine Adjusting (Shooting Forward/Backward)

[//www.youtube.com/embed/4e7myGchBCQ](https://www.youtube.com/embed/4e7myGchBCQ)

:

On the other hand the six videos below are films shot with some adjusted values for parameter set. In the first one, **SAMPLE (1), intentional PANing** is used so frequently that we can see its effect well. The second **SAMPLE (2)** is uncut video shot by going up and down the stairs at a quick pace. And the rest videos, **SAMPLE (3) to (6)**, have scenes shot under so various conditions that they would give us good information to estimate the efficiency of the DIY camera gimbal gotten in this project.

:

[SAMPLE (1)] Effect of Intentional PANing (Turning 540 Degrees)

[//www.youtube.com/embed/wVzUWIkOTrY](https://www.youtube.com/embed/wVzUWIkOTrY)

[SAMPLE (4)] Walking on Dirt Path in Woods

[//www.youtube.com/embed/OtDR9s76GUk](https://www.youtube.com/embed/OtDR9s76GUk)

:

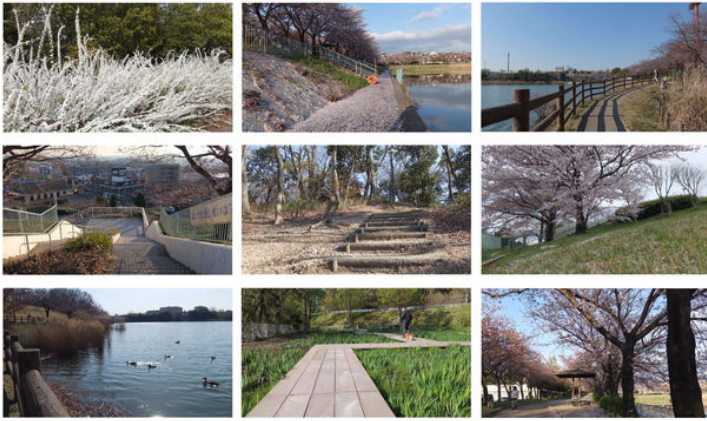
[SAMPLE (5)] Uncut of the Gimbal Working from Power On to Off

[//www.youtube.com/embed/4KoywRbmzDI](https://www.youtube.com/embed/4KoywRbmzDI)

:

[SAMPLE (6)] Low Angle Shooting

[//www.youtube.com/embed/x__cy9W4qvk](https://www.youtube.com/embed/x__cy9W4qvk)



Step 14: Room for More Improvement

I have shown a story getting the DIY 3-axis Brushless Gimbal with Arduino of practical use. As searching web, there are few sites which describe driving Brushless Motor using only Arduino without other special devices. And it is a pity that the rare program provided in a precious article describing it has turned out to be unsuitable for the camera gimbal. (See **Step 7.**) Therefore I have prepared a new program independently and developed the DIY gimbal in order of single, two- and three- axis while augmenting the program.

The policies shown in **Introduction**, *finishing this project with no imitation and no special device*, seem to be kept finally. On the other hand I don't know the merits and demerits of this DIY gimbal well because I have not seen camera gimbal except for it. Here I describe the left room for improvement at the end of story.

:

(1) High Frequency Noise [: Resolved]

The program prepared in **Step 8** makes Brushless Motor have strong torque in stopping or rotating. But the operated motor generates **noise around 500Hz** continuously. This noise has little negative effect to the picture of video. But recorded **sound** is so contaminated by it.

(* Added:)

It was found that the coils in Brushless Motor generate this noise by **PWM pulse** applied. The noise can be shifted into non-audible range (31kHz) with making the frequency of the pulse higher. It can be done in **Step10** by correcting two points below. I appreciate the comments of [EricL50](#) and ThaddeusW3.

Change Arduino-pin assignment in both the wiring and the sample program.

- Pin8 > 5
- Pin9 > 6
- Pin5 > 9
- Pin6 > 10

Add four lines below in "setup { }" in the sample program.

- TCCR1B &= B11111000;
- TCCR1B |= B00000001;
- TCCR2B &= B11111000;
- TCCR2B |= B00000001;

The video "Sample (7)" below was shot with the modified sketch. The high-pitched loud noise cannot be heard in it. The values of parameters in the sketch should be adjusted again to keep good compensation in this modification. There seems to be room for it still. The next video "Sample (8)" was shot with adjusted parameters. We can recognize higher stability even though shot in a high wind.

:

[SAMPLE (7)] Noise Eliminated

//www.youtube.com/embed/YaUE7wMpwlc

[SAMPLE (8)] Parameter Adjusted for Noise Eliminated Version

//www.youtube.com/embed/Dg2LEi1aHcY

:

And long Shots for 15 minutes: TEST(1), TEST(2), TEST(3)

:

(2) Chattering of the Gimbal Frame [: Almost Resolved with Arduino DUE]

I met with the persistent **chatter vibration** of frame in the stage where the 2-axis gimbal of practical use was made from the single axis test bench. In contrast to the noise above, this chattering negatively effects picture. In the video shot with this chattering, the shape of object undulates. Though I do not know the accurate frequency of the chattering it is felt like about 50Hz. As touching some point of the frame by a finger, this chattering stops immediately. But it would disturb gimbal working.

:

[DOCUMENTARY (18)] Disturbed Picture Caused by Gimbal Chattering

//www.youtube.com/embed/qaChMXT3pOk

:

Though the root cause of this persistent chattering is not clear, the function of vibration damper in the sketch to compensate unintended tilting causes it. In this project the chattering is reduced by some **conventional symptomatic treatments** as follows. But more drastic measures are required for the fundamental solution such as **improving frame rigidity** and so on.

- Adjusting values for parameter set in the program: Loosening compensation a little
- Lowering power: Decreasing supply voltage to motor (7.5V) at two thirds of the lower level recommended one (11 to 15V)
- Increasing size of the handle and attaching some brackets to it: Absorbing the vibration by the palm or changing the natural frequency of the gimbal by clutching bigger handle with brackets tightly.

:

(* Added in May 2018:)

I have described that we may be able to solve the insistent chattering or vibration with using Arduino **DUE** instead of Arduino **MEGA**. See the comments of the_3d6 and the replies for them. (They were posted about 11 months ago.) DUE has a 32-bit ARM core and its clock speed is 84MHz, though MEGA has a 8-bit AVR core and its clock speed is 16MHz. I also found that it is easy to change the PWM frequency on DUE. And fortunately in L298N, a motor driver IC, the minimum input voltage at HIGH status is 2.3V. Hence DUE, 3.3V board, can work without logic level shifter. However pin 50, 51 and 52 on DUE are not assigned for SPI, though these pins can be used for SPI on MEGA. Hence additional wiring is required on the DIY motor driver shield gotten in Step 12 to try DUE newly for 3-axis gimbal. After that, I tried DUE with my gimbal and found that the insistent chattering or vibration can be much reduced or almost disappeared but the ad hoc parameter set for MEGA should be changed for DUE. I recommend DIY makers should begin with DUE instead of MEGA when building a multi-axis brushless gimbal.

[SAMPLE (9)] With Arduino DUE instead MEGA (Parameter Adjusted)

//www.youtube.com/embed/LSbYXa8GsTM

:

The **horizon** shot with gimbal tells us how good it is working.

:

(3) Un-intentional ROLL Still Remaining [: Almost Resolved with Arduino DUE]

When seen from the outside, the DIY gimbal seems to compensate enough against the un-intentional rotation of its support. But seen from the mounted camera, the rotational stability of a horizontal line is not perfect. It is thought important to improve it that the following matters are resolved.

- Inappropriate values for parameter set in the program: Loosening compensation to reduce the gimbal chattering
- Insufficient power supply: Decreasing supply voltage to motor at two thirds of the recommended one to reduce the gimbal chattering
- **Discrepant coordinate system** between gyros and frame of gimbal

The first two matters come from the conventional countermeasure against the gimbal **chattering**. Then the fundamental solution to the chattering is required here also. On the other hand the last matter concerns the **calibration of gyro**.

The program prepared to control the gimbal in this project is assuming that all four gyros and frame of gimbal have the same coordinate system (or the same three orthogonal axes). But in reality, each of them should be thought **specific** (*) and the three axes of each system would **not** be orthogonal strictly. It is possible to know the posture of each system of gyros comparing with the system of the frame. But it does not seem to be so easy and its cost-effectiveness is not clear. Hence it has not been executed in this project as yet.

(*) No special measuring or adjusting has not been done when the four gyros were attached to the frame of the 3-axis gimbal.

(* Added in May 2018:)

The lens axis of the camera in the gimbal is **not always parallel** to the shaft of the roll motor or Z-axis of the main gyro to compensate camera rolling. And this gyro is attached off-set on the special shield on Arduino at the back side of gimbal. See the pictures in Step 12. They are considered explicitly in the sketches for the video "Sample (7)" and the later. Though this revision could contribute to an obstinate vibration, a faster controller Arduino DUE can settle it. See the video "Sample (9)" above.

:

(4) Disturbance on SPI interface

The outputs of gyros are transmitted to Arduino by SPI interface in this project. Though it is used for a short range communication between digital devices, there ought to be no problem within three feet of distance. Indeed few matters has been met with while the electrical materials are plugged into a breadboard and connected to Arduino by jumper wires. But after replacing these jumper wires with **ribbon cables**, the behavior of the gimbal has become abnormal.

For a while (some days) the cause has not been unknown at all. Through a considerable process of trial and error it was made clear that SPI interface is sometimes disturbed when some special lines is put close. Especially the disturbance occurs very often when **Clock-line (SCK)** is brought near **Output-line (MISO)**. No article which describes such a phenomenon has been able to be found. Hence it is thought that this is a specific problem between Arduino and the gyro sensor used here. Though the true cause has not been known, the problem can be resolved by choice the proper combination of bundled wires.



Hello? I am a college student majoring in engineering in Korea. I was impressed with your project and learned a lot. Thank you very much.

May I ask you a few questions?

I read Mr **Abil FidaA**'s comments. And I finally knew I had to **solve the optimization problem** in order to maximize the stopping power and You answered like this :

1. Specify the degree of magnetic force on a slot (coil) at arbitrary point (angle).
2. Sum up forces of all slots at the point.
3. Solve maximum stopping force problem at arbitrary point (angle).
4. Approximate the curve of solution with some lower order function

```
void calcPwms() {
int degPwm1000 = deg1000 % 8571;
if ( degPwm1000 < 0 ) { degPwm1000 = degPwm1000 + 8571; }
long linePwm = ( LnLv110 + ( slope1000 * degPwm1000 ) / 100000 + 5 ) / 10;
long degNL1000 = abs( degPwm1000 - LnT1000 - nLnT1000 );
long nLinePwm = 255 - ( ( - 2935 * ( sq( sq(degNL1000)/1000 ) /1000 )
+ 8413 * ( ( sq(degNL1000)/1000 ) * degNL1000 /1000 )
- 11421 * ( sq(degNL1000)/1000 )
+ 32929 * ( degNL1000 )
) / 100000 + 5
) / 10;
```

Maybe the code here came out of the problem of optimization. Am I understanding correctly?

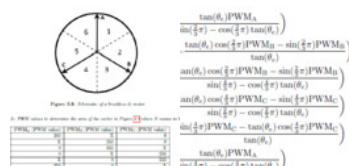
But I have no idea how that equation was induced. It is not easy for you to explain it, but can you draw some data, clues, or pictures that I can understand?

I studied Space Vector PWM and found a theory about BLDC PWM Position Control. Please check the attached file. It may be a geometric solution to the problem.

Your comment is a great help to me.

Thank you very much.

Jong-hyeon



- I saw those formulas, but I could not use it because if you see the first part when the angle is between 0° and 60° , you are going to find that the output value (X) does vary from 0 to 255, when the angle is 0° and 60° , respectively, nevertheless, as the angle increments it gets negative values and values greater than 255.

Blessings!



I think the pictures you attached are close to my idea. However, I cover only 51.429 degrees to control, though they seem covering 360 degrees. Here $51.429 = 360 / (14/2)$ and 14 is the number of poles in BLDC. Also I use no trigonometric function, because I approximate the arc with 51.429 degrees as a short line.

I recommend you to draw two figures, each of which has 3 line charts. One is based on the sample sketch attached in Step10, and the other is based on the pictures you attached. I think they will be similar to each other. If you draw them, post them please(*).

(*) Reader vonkross has posted good comments and figures a few years ago. One of them is the former figure above. Now they are disappeared unfortunately.



How did you get the power(hold torque) to stop the motor? Did you rewind the coil of the stator? or Did you increase the number of turns in the coil? I think position control is necessary to obtain holding torque. But you didn't use encoder or hall sensor. Please explain more thoroughly.



I replied you in Questions in this page. See above.



Hello!

Just had a quick question about your gimbal, would this design work if I were to flip it completely upside down? Or does it have to be held from the 12 o'clock position?



Hi,

Pistol Grip Gimbal you say. It works better for a smaller camera such as GoPro. Download the cover picture above and rotate it upside down. You can see that it is near to Pistol Grip. Only the camera attachment cradle must to be set upside down again with the same tilt axis. It requires enough clearance not to interfere with the pan motor on the bottom.

- Hi ArduinoDeXXX,

First of all, I have to say I'm loving your work!

I'm trying to understand in depth your Brushless Gimbal With Arduino project and having some questions i would love you to answer for me about the calcPWM routine, if you don't mind.

1- I would love to know how you get the linePwm formula and its value meaning, as well for nLinePwm.

2- How do you get the int LnLvl10 = 390 and LnT1000 = 4291 values? I've read that they are levels of the reference points of PWM in a cycle, but how do you get them?

Thanks a lot!



I have gotten several comments like your question ever. See earlier comments and the replies to them*. You can see all of them by clicking "More Comments" button. If you search on this page for some key-word, you could find answers or get good information.

(*) For example, Abil FidaA, game123, EricL50 and **eggll** have posted technical questions and I have replied to them.



Supplement:

vonkross, a reader, posted some comments with good figures. One of these figures explains the function "calcPwm()" and some constant variables well. 10 (or 1000) in the name of variables means multiplying the original value to 10 (or 1000) times.

- Thanks for your reply but I think my question was confusing. I was referring to what mathematical procedure you have done to arrive at the formula of variPwm which is similar to a sinusoidal function, since I imagine that it has not been a casual adjustment by trial and error.

Also, I have a problem with which I would need some help. I have adjusted your code to my hardware and I have verified with the Serial.print () function that my motor controller and the 3-axis gyroscope sensor as they should, but the motor only turns in phases 2 and 6 of the code and in an ineffective way. So, I have performed the following test on the gimbal motor (vonkross comments):

```
void setup () {  
  pinMode (9, OUTPUT);  
  pinMode (10, OUTPUT);  
  pinMode (11, OUTPUT);  
}  
void loop () {  
  analogWrite (9, 0);  
  analogWrite (10, 255);  
  analogWrite (11, 0);  
  delay (1000);  
  analogWrite (9, 0);  
  analogWrite (10, 255);  
  analogWrite (11, 255);  
  delay (1000);  
}
```

I have seen that the motor turns + - 8.5° every second as it should. Then I tried out the two tests of step 6, and the engine rotates approximately + -20° every second like in the previous test instead of turning continuously as it should. I also checked line brakings with a multimeter and everything is OK.

I have the following gimbal engine: https://hobbyking.com/en_us/turnigy-hd-2212-brush... It has the same configuration as the one used in your project (12N14P and star termination).

What do you think could be the problem?



Both two sample sketches in Step 6 could work with every 3-phase BLDC motor. See the eLABZ's commentary in his/her site described in Step 6.

If a 3-phase BLDC with no wire broken doesn't rotate with these sketches, it should be doubted that the rotation speed too high. You can change it slower. Constant "motorDelayActual" in these sketches decides it.

I think you can get good information by three steps below.

- 1) Click "More Comments" button at the end of page until it is not displayed.
 - 2) Search this page for the key word "motorDelayActual".
 - 3) Search this page for two key term "it is not easy" and "game123".
-

- hie i am currently working on these project.. but i am little bit problem with circuit diagram so if u can give me the circuit diagra, that would be very helpful
-



Hi,

The upper picture in Step10 is almost the same as the schematic, though lines omitted. I think you can get the schematic with lines by referring to the picture. If you posted your schematic, we might be able to check it and say something.

- Hi ArduinoDeXXX

I keep reading your one-axis code.

And Could I ask something about that code?

Is this right that "1000" means about 1000 times degree and angle?

And what do "51429 and so many number" means?

I thought I can do it use PID. But I stopped 2 months after just running BLDC.

Would you help me just a minutes? :))

I stuck and I need this.



Hi,

I have gotten many comments for this project. See earlier comments and replies for them. I think you can find what you want to know. Have fun!



Hi ArduinoDeXXX

Thank you so much for a great article - I think you did a superb job with the written English, because, perhaps, your spoken English is not good enough for spoken commentary. The way you educated us through several iterations was superb, and very clever videos. I help run a Hackspace in Leicester, UK, and I feel this would make a great project for our members, demonstrating advanced motor driving and complex feedback in closed loop systems. I appreciate that you wish to hold on to your final code, but it is great that you have answered various posters' questions.

Thank you

Tony

www.leicesterhackspace.org.uk



I appreciate your thoughtful comment. I see your site of Hackspace in Leicester. It seems very good studio and people look like having so much fun. I would be glad if this article could help you and them. Thank you.



hello

I'm impressed. normally I like inventions that are new, but camera gimbals are super cool but super expensive. I am a rookie, I just made my first hackintosh, fast computer for less money. I make music and started with making video. I just discovered the world of microprocessors and I'm full of ideas. I build things since kid, e guitars, analog fx I can solder and I have passion.

maybe someone can give me an advice how I should start with micro comp?

I think a raspberry pi could be a thing for me to start. actually I have no clue. but with the hackintosh it was the same and now, with help from forums and a lot of time, it's my main computer.

I think when I start to know Arduino and stuff, you guys will have the perfect DIY gimbal.

all the best

mo



Thank you for your comment. I think you could enjoy DIY projects with microprocessor. If your projects require fast image analysis, Arduino cannot work well (Raspberry Pi can work). If not, you can get start your project with Arduino at a lower cost and skill.

◦ Hello ArduinoDeXXX,

I really want to thank you for the great article. I read it maybe for 30th time.

Since a few weeks I am struggling with my gimbal and I have a bit progress. In this video you can see the latest approach. I use arduino nano, cheap brushless, 12V and no ESC.

[//www.youtube.com/embed/UqkEfOHl3OY](https://www.youtube.com/embed/UqkEfOHl3OY)

I have following problem: There are some positions where the rotor oscillates when it comes to the desired equilibrium. You can see this in the video, it oscillates maybe two times when it comes to the zero state. In the video I have a P control. I tried to add a D block with the idea to damp the system but it didn't work. When I make the P parameter very small than there is no oscillation but it is way too slow.

I thought that the problem can be in the array with the sine wave. I increased the number of the array elements from 96 to 576 and it got a bit better but with 1100 it is the same but slower.

Another thing I see is that the motor can't run really smooth when it rotates slow. It is always a bit jumpy. When the currency is turned off and I rotate the rotor with hand I can feel pretty strong magnetic resistance (EMF*). Maybe the problem is that the motor is not the right one for gimbal but it was sold for this purpose. It has 12 cogs and 14 poles.

I discribed some additional specifications in the description of the youtube video.

Thank you very much!!



I saw your comment and video. I think that your code would be similar to the second sample sketch in Step6. To get a good gimbal, I have tried this sketch and abandoned it. See the two videos in

Step7; DOCUMENTARY (7) and (8). In these videos the board attached to the top of BLDC motor cannot stay at a point. The board is bigger and heavier than yours. You might see the same situation if you attached a bigger and heavier board to your gimbal. You should try the sample sketch in Step10, which could work well with your BLDC for a single axis gimbal.

Thank you for your reply!!

You are right, if there is a bigger moment of inertia I will land at exactly at the behavior of DOCUMENTARY (7) and (8).

I am doing this project with the purpose to get experience in control theory and arduino and I want to follow your approach from DOCUMENTARY (7) and (8) to DOCUMENTARY (9) by myself and not just copy paste it.

Could you please advice where should I focus and which topics should I research to overcome my problem?

You mention that the difference between Y and Delta winding was a big point for you and also the change of the goal to "make the motor stop at the right position".



You can make your BLDC rotate. Hence you should not stay at Step6 or 7. I think you should move Step10 and try to get a single axis gimbal with the sample sketch at first. You will be able to know more with reading earlier messages and replies and changing the sample sketch.

Hello ArduinoDeXXX

Very interesting project. Can you please tell me where I can find the arduino code for the project?



Three sample sketches are shown in this article. The first two attached in Step6 could not work well for a gimbal. The last one for a single axis gimbal is attached in Step10.



Three sample sketches are shown in this article. The first two in Step6 could not work well for a gimbal. The last one for a single axis gimbal is attached in Step10.

Hello ArduinoDeXXX,

I've read your wonderful instructable on making a gimbal brushless, and I have been able to make one working on two axis (x and y). But I have come across a problem that I can't solve :

My start position for having setup leveled is different than yours, and when I modify the following lines in void calibrate :

```
analogWrite(motorPin1, 217);
```

```
analogWrite(motorPin2, 0);
```

```
analogWrite(motorPin3, 255);
```

with my values (0, 210, 255 for x-axis motor and 255, 188, 0 for y-axis motor)

it goes to the correct position for the calibration but returns to the previous position after that.

(see in the video, the calibration position (0, 210, 255) 00:01-00:03

and then the return to "standard position" (217, 0, 255) when in the main loop 00:04-00:07.)

<https://www.youtube.com/watch?v=kCzqO0uaX04>

I feel like I need to change something in the nLinePwm calculations but despite reading the comment you made to "game123" I don't really know what to change to make it keep the new start

position.

Thanks for the help!



In "void loop()" in the sample sketch in Step10, the degree to rotate BLDC is decided by variable "deg1000" in the first function "chkAndCtl()". And the value of "deg1000" decides PWM output as variable "variPwm" in the the second function "calcPwms()". The angle of BLDC rotating can be shifted by adding a constant to "deg1000" between these two functions. See below.

```
chkAndCtl( );  
deg1000 = deg1000 + 12345;  
calcPwms( );
```

- Oh yes I didn't thought of adding a constant there...
Now the position is correct !
Thank you so much =)



hi ArduinoDeXXX!

i see your project is very good, and i like it
so i made my own, but i am using other sensor
i just don't understand with deg1000 's value, if this value from reading the sensor
what kind of that value? is it degree or RAW data?
i'm sorry for my english, thanks before



Variable deg1000 represents 1000 times of the degree to rotate BLDC. You can find some topics about it in the comments below.



thanks for your reply!
so that you mean value from deg1000 is the actual angle*1000?
for example the value of reading sensor is 90 degree, so deg1000 now is 90000?



Yes. See the comments of EricL50 and replies for them posted about 2 years ago. You can know more detail.



i see thank you very much!
and i think your program is fixed program, i mean fixed constant value
when i use your program, i just get a little bit something error, so i try to setting with change the value of that constant value, like LnLvl10 and slope1000 and the other. But it didn't make a change. Actually where is the value that i can setting or tuning in your program?
like if i use PID control, that i can tuning by set the value of Kp,Ki, or Kd



Two sample sketches attached in Step6 could work with any 3 phase BLDC motors. On the other hand, the sample sketch attached in Step10 should be limited to 3 phase BLDC motors with 14-

pole/12-slot. Though it could work with many 14-pole/12-slot BLDC motors, it might not work with a special Delta connection. However I have never been told such one. See Step8 and early comments of Ramirez_MecaUPQ and replies which contain pictures.



here i using same motor with 14P12S

and when i upload your program in step10 without any load,it work so well

but when the load is placed, it won't stop at the point and also make a little shake

it has same result when i reduce the load with the thing that not heavy enough like the load before.



If your BLDC is the same model shown in Step5 or its successor model, the sample sketch in Step10 ought to work well. It holds rotator of BLDC strongly at a point when three PWM values are kept constant like "217, 0, 255". If not, breaking of wire in BLDC is suspected. See earlier comments about this topic.



thank you

i will try to fix it



i see! thank you very much

i think your program is fixed program,i mean fixed constant value, when i use your program it just a little bit something error. So i try to setting by change the value like LnLvl10 and slope1000 and the other, then i upload it. But nothing changes. Actually where is the value that i can setting or tuning? like if i use PID, that i can setting by the KP, KI or KD 's value



Hi ArduinoDeXXX!

I bought this motor:

http://www.hobbyking.com/hobbyking/store/__45366__...

But when I use your code (DOCUMENTARY 6) my motor behave like on movie:

[//www.youtube.com/embed/3A-7pYXajS0](http://www.youtube.com/embed/3A-7pYXajS0)

(it isn't smooth)

Can you help me and say why it is so?

Thanks! :)



It is hard to make BLDC rotate slowly and smoothly with the sample sketches attached to Step6. You will execute it with the sample sketch attached to Step10. See the comments of vonkross and the replies to them (*).

(*) The hyper link to earlier comment (under "More Comments" button) doesn't work well.



Hi! Thanks for your reply! :)

I made some progress, now my "gimbal" works like your in DOCUMENTARY 9 (its shaky)

But i can't understand how you improved gimbal by adding second gyro, with single gyro I know that I need eg. 10degree, currently is 50, my error is 40degree so a need to compenaste this, but how is it works with two gyros?

Have you done some other improvements?

Thank you! :)

Hello Tomasz,

would you please upload the sketch you used for this video :) Thank you!!!



In your video, a small shake of camera is observed at the end of compensation (such as 0:04-0:06). The shake of camera is observed in the video, DOCUMENTARY (11), in Step9 at 1:51-1:58 also. There the secondary gyro attached to the camera-mount does not work. On the other hand, the shake does not appear in the same video at 2:14-2:24. There the secondary gyro attached to the camera-mount works to counteract the shake.

I think you can get good answer by yourself when you complete your gimbal and execute trials and error in practice.



I finally go it!

Thank you very much ArduinoDeXXX! :)



Wow, a great effort and very detailed report! Thanks!

Once we made a similar project, but it turned out that atmega328 isn't enough to handle 3-axis stabilization at proper speed (it wasn't able to compensate for really fast vibrations), so we switched to stm32 instead. Gimbal chattering was almost completely gone (appeared only in some very special positions) when we managed to complete feedback cycle in less than 1 millisecond. On atmega it was around 6 milliseconds, and I'm almost sure it isn't possible to completely get rid of chattering at such speed.



Thank you for your good comment. I also have suspected that Arduino Mega with ATmega2560 does not have enough power for my DIY gimbal. Hence I have gotten Arduino Due which has ARM Cortex-M3 CPU in AT91SAM3X8E.

I have corrected the original sketch for Mega to work with Due. Though it works well with default PWM frequency, the loud noise is heard. I am searching a good code for Due which can change PWM frequency higher to shift the noise into non-audible range (31kHz) with no trouble.



Due is fast enough, but I highly doubt you can find good enough PWM library for it (actually I don't know a good library even for atmega328, have to write registers directly anyway). Your best shot is to go through its datasheet, and find out how to switch timers into PWM mode using their registers. But it is quite complicated, I just took a look here: <http://www.atmel.com/Images/Atmel-11057-32-bit-Cor...> - and pages from 970 describe its PWM module, can't even really understand how to make it (on page 996 there is "simple 19-step guide" and pages 1003-1006 describe "user interface" registers).

STM32 is so much easier with their HAL library! After stm32Cube makes all initialization code for you, controlling timer takes just a few rather intuitive lines. I really recommend to try it (there are a lot of Nucleo and Discovery boards with stm32F0, F1, F3 and F4 series processors, here F1 is optimal, F4 is much more powerful, but also somewhat more complicated)



Thank you. I appreciate your suggestion. I get a good alternative.

I have looked for topics in web about changing PWM frequency of Due. Then I found a forum about it. A very simple way is described in it. It tells that all we should do are simple changes in **variant.h** in the Arduino library.

However I cannot find variant.h in my old IDE or the latest version in web. I have downloaded earlier versions between the former and the latter. At last I found it in IDE 1.6.0 and 1.6.1. Why it cannot be found in the later versions?

I felt something disquieting. I stopped and left it for other new projects. I am going to come back to variant.h when these new projects are finished.