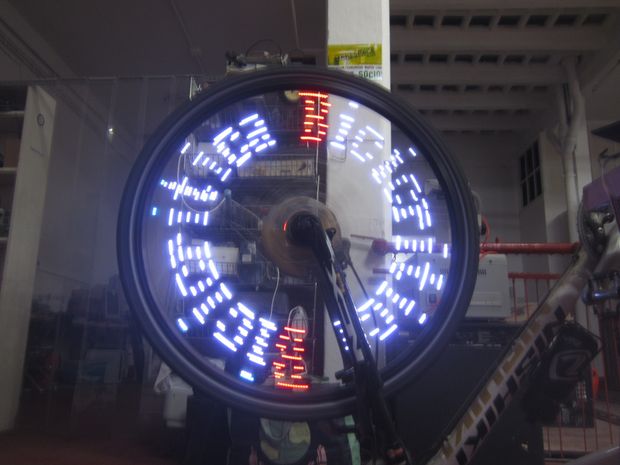
[[](http://cdn.instructables.com/FR5/PRHK/HY12NHP2/FR5PRHKHY12NHP2.LARGE.jpg)](http://cdn.instructables.com/FR5/PRHK/HY12NHP2/FR5PRHKHY12NHP2.LARGE.jpg)

[[](http://cdn.instructables.com/FP5/FDEK/HY12NHP8/FP5FDEKHY12NHP8.LARGE.jpg)](http://cdn.instructables.com/FP5/FDEK/HY12NHP8/FP5FDEKHY12NHP8.LARGE.jpg)

[TODO: insert a neat video or pictures from actual riding]

[Note the Arduino sketch works but is work in progress, see last step for link]

Persistence of Vision (POV) effects let you display arbitrary images with just a few controllable pixels on a fast moving object by changing the pixels' colours fast enough. There's plenty of existing POV projects using a LED strip on a spinning object, like a bike wheel or a spinning CD.

This is yet another implementation of POV on a bike wheel. Adafruit [has a similar project](http://www.ladyada.net/make/spokepov/), they sell the kits and have it well documented, there's probably a few more implementations. All the ones I've seen used custom LED controller circuits though. This instructable uses just the most obvious, off-the-shelf elements and requires some basic soldering and a lot of zip-ties.

It also allows you to switch the LED effect programmes without getting off your bike, by using short sequences of braking ("gestures") which are detected by the same sensors we use for keeping track of the wheel position. Wifi, bluetooth, etc. control would also be easy but I've not done that.

Here are the materials I used, but you may have similar components at home that are likely compatible and will work just as well.

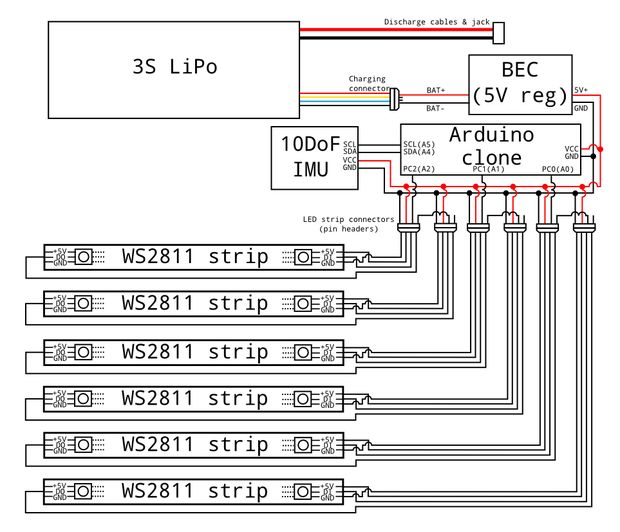
* 1.5m of the well-known WS2811 / WS2812B / compatible LED strip. I got a 5m reel of the "waterproof" variant at aliexpress.com for about $50 ([here's](http://www.aliexpress.com/item/FREE-Shipping-5M-60Pixel-M-300-5050-RGB-SMD-WS2811-IC-Built-in-WS2812B-Black-PCB/1200913601.html)one seller). You may not need 1.5m. I have the LEDs in 25cm 15-LED strips mounted on 6 spokes on a standard 28" wheel. If your wheels are 26" you might want 14 LEDs for example and you can start with just one spoke.
* A small Arduino-compatible board. I used the popular cheap $2.12 Pro Mini clones from aliexpress.com (10 MOQ - [here's](http://www.aliexpress.com/item/10Pcs-Lot-Pro-Mini-Module-Atmega328-5V-16M-For-Arduino-Compatible-With-Nano/1432639609.html) the seller). My firmware code should work with any Atmega328-based board.
* A 6+ DoF sensor board like the popular $10, 10DoF Acc+Gyro+Mag+Baro IMUs from e-bay. My code is only tested with the Invensense MPU6050 sensor chip ([here's](http://www.aliexpress.com/item/GY-86-10DOF-BMP085-Module-MS5611-HMC5883L-MPU6050-Module-MWC-Flight-Control-Sensor-Module-Free-Shipping/1121850654.html) my seller - but you can grab a similar board with the BMP085 instead of the higher-precision MS5611 baro and it'll be below $10 - we won't use the barometer here anyway)
* A lithium-polymer RC battery with enough capacity for your rides, the amount these LEDs consume depends a lot on how they're used, but they're quite power-hungry. I'm using a variety of batteries, for example [this 3S1P 3700mAh one from HobbyKing](http://www.hobbyking.com/hobbyking/store/uh_viewItem.asp?idProduct=21355), which should last for a good few hours in any configuration. You don't need that high a C rate, even 5C would do. Any voltage between 7V and 30V will do so a 2S or higher LiPo. Note the battery width matters because it needs to fit between the spokes of your wheel somewhere close to the hub end.
* A 5V step-down regulator, optimally a switching one. In RC airplane hobbies those are called BECs. I use a 25W one [similar to this](http://www.hobbyking.com/hobbyking/store/__18521__Turnigy_HV_SBEC_5A_Switch_Regulator_8_42V_input_.html). You may not need all 25W if using fewer LED strips than 6 but it's safe to get a bigger one, and my 25W BEC already gets hot.
* Some 4-wire cable, male & female pin headers for the plugs and the sockets, thin zip-ties, optimally some hot-glue. Soldering stuff. Perhaps some more mounting and water-proofing material if you can make use of it. My setup isn't water-proof and my mounting doesn't look very pretty.

This should come out at about $100 in total or less than $50 per wheel. You also need a LiPo charger if you don't have one. You'll also need a bike (or at least the wheel).

If you're new to Arduino, you should know that you'll probably need a USB-to-Serial adapter to program the board, unless the Arduino has a USB port already or unless you have a real serial port & cable in your computer. If you don't code, you probably want to reproduce my setup exactly.

If you're not new to Arduino, my code currently has just some basic LED programs, you'll probably want to modify/add your own. The code achieves about 400-700 Frames Per Second, so at, say, 20km/h, you'll have only about 200 virtual pixels per LED along the perimeter depending on the wheel diameter and how intensive the effect's code is on the Arduino. I'm not storing bitmaps on the Arduino, each effect (programme) is a function that calculates what the current colour of each LED needs to be. More info on the firmware in the final step of this instructable where we program the Arduino.

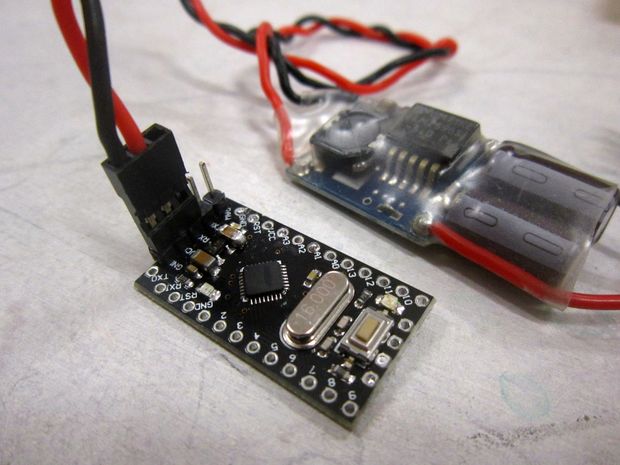
Let's connect the pieces together.

[[](http://cdn.instructables.com/F3N/NB3B/HXXHGJ2M/F3NNB3BHXXHGJ2M.LARGE.jpg)](http://cdn.instructables.com/F3N/NB3B/HXXHGJ2M/F3NNB3BHXXHGJ2M.LARGE.jpg)

Here's the diagram of my wiring. There's no individual electrical components, just the battery, LEDs, voltage regulator, the Arduino and cables.

One thing that I'd have added in retrospect is a connection between BAT+ and one of the Arduino analog inputs, e.g. A3 (PC3) through voltage divider resistors so the Arduino can measure the battery voltage. Discharging a LiPo cell below 3V destroys that cell.

[Note: I'm not sure what those exact LEDs are called, possibly WS2812B. In the diagram I used WS2811 as the name of the family of WS281x chips and their clones]

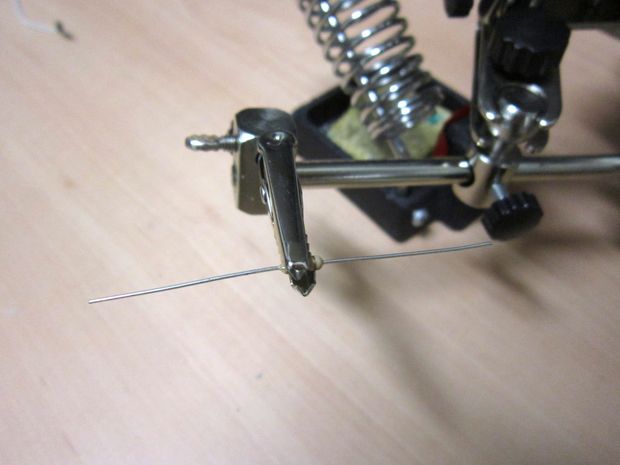
[[](http://cdn.instructables.com/F7F/LQ05/HY12NI9H/F7FLQ05HY12NI9H.LARGE.jpg)](http://cdn.instructables.com/F7F/LQ05/HY12NI9H/F7FLQ05HY12NI9H.LARGE.jpg)

To power the Arduino, we'll connect it to the voltage regulator through its VCC and GND pins. The regulator comes with a "servo connector" which looks like 3-pin standard female pin-header. So we just need some male pin-headers on the Arduino. On the Pro Mini the through holes for the power pins are on the opposite side from the reset button, together with three serial port pins: Tx, Rx and DTR. We'll also be using the three serial port pins for programming, so just solder a 5-pin male header row in the holes labelled:

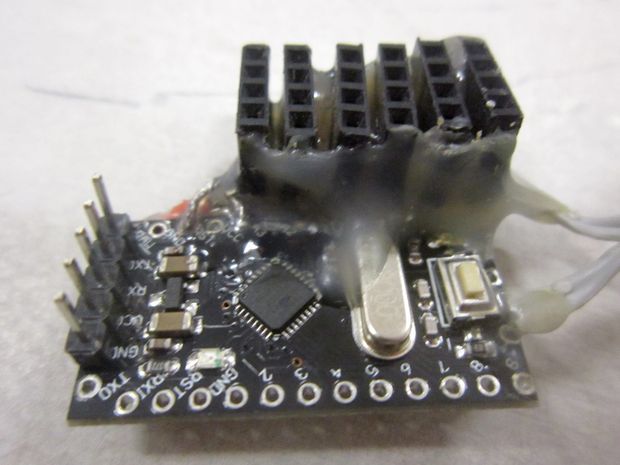
GND, VCC, RXT, TXD, DTR (the DTR label is on the bottom of the PCB)

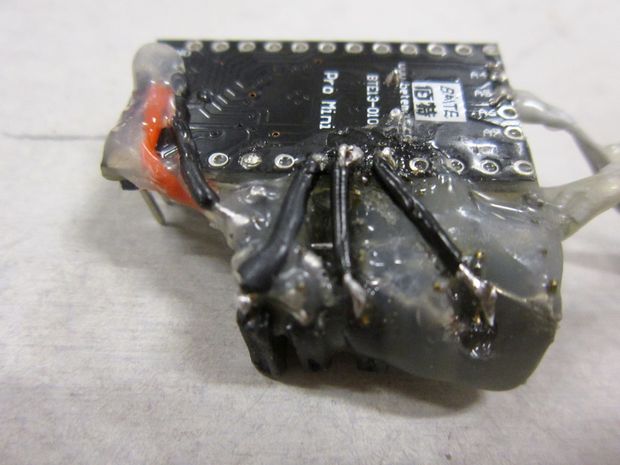
The power connector is ready.

When you connect the regulator's servo lead just make sure the black or brown wire connects to GND, red to VCC and the third pin, which will connect to RXT, may be unpopulated on the servo connector or it may have a dummy white or yellow wire.

[[](http://cdn.instructables.com/F8N/NV5G/HY12NTIR/F8NNV5GHY12NTIR.LARGE.jpg)](http://cdn.instructables.com/F8N/NV5G/HY12NTIR/F8NNV5GHY12NTIR.LARGE.jpg)

[[](http://cdn.instructables.com/FVW/Z6IR/HY12NTFE/FVWZ6IRHY12NTFE.LARGE.jpg)](http://cdn.instructables.com/FVW/Z6IR/HY12NTFE/FVWZ6IRHY12NTFE.LARGE.jpg)

[[](http://cdn.instructables.com/FNY/1J0W/HY12NTEZ/FNY1J0WHY12NTEZ.LARGE.jpg)](http://cdn.instructables.com/FNY/1J0W/HY12NTEZ/FNY1J0WHY12NTEZ.LARGE.jpg)

[[](http://cdn.instructables.com/FXK/9MD3/HY12NTF4/FXK9MD3HY12NTF4.LARGE.jpg)](http://cdn.instructables.com/FXK/9MD3/HY12NTF4/FXK9MD3HY12NTF4.LARGE.jpg)

I started with one strip of LEDs for testing but made connectors for 6 of them for expansion and later ended up using six. Each "connector" is a 4-wire female pin-header sliced off from a 40-pin row. I mounted the headers perpendicularly to the side of the Arduino board with some distance between each header so that the male connectors can be slightly thicker than a pin header, as in the photo.

The connector has the following lines starting from the side closer to the Arduino: GND, 5V, DOUT & DIN. I was too lazy to make a proper PCB to solder the headers in, so I sort of made an ugly ladder of wires initially held in the air by a third hand. The GND lines of all 6 headers are connected together by a piece of rigid wire like those of a through-hole resistor, same for the 5V lines of the connectors. If the pictures and the diagram from step 1 are enough for you to follow then you don't need to read all the details below. Sorry I don't have a picture from before the whole shebang was covered in hot glue, you can also see I had a few redesigns there... the thing holds together Ok though, I made a few hundred kilometres with it.

I'm not that great at soldering so this is what I did:

1. cut off 6 pieces of 4-pin female headers.
2. grab each one in turn in a third hand, tin all 4 pins so they're easier to solder things to. Whenever soldering connectors of any kind it's good practice to have them plugged into the opposite side of the connector (plug or socket) to increase heat capacity and make it difficult for you to accidentally melt the plastic. So put some male headers in the female connectors when soldering.
3. grab a small through-hole resistor in the third hand, so that it has two pieces of straight wire sticking out on both sides.
4. slide hot soldering iron along the wire feeding it a little tin so as to cover both wires in tin.
5. starting from one end grab a female header with pliers, touch one of its pins to the resistor wire, heat for a moment for the pin to stick to the wire with the tin already present.
6. once all 6 pieces of female headers are holding to the resistor wire, cut off that side of wire from the resistor.
7. now hold this structure with pliers again, and with the resistor still held by the third hand touch the second pin of all the header rows to the resistor wire and solder the second pin of each row.
8. cut the wire on the second side of the resistor, store the resistor for future projects.
9. you can add blobs of tin here and there to improve the connections.

Now we'll need to connect these headers to the corresponding Arduino pins and also make some connections for the LED signal between pairs of headers. Use another piece of wire to connect the ground line under the headers to the GND pin on the long side of the Arduino PCB, same for the 5V (VCC) line. Perhaps use a cable with plastic jacket to avoid shorting the wires where they cross.

To connect the signal lines you need to establish some numbering for the header connectors, it doesn't really matter what it is but we need to call them something. Use another piece of wire to connect pin 3 of connector one to pin 4 (the one furthest from the Arduino) of connector two. Same for pin 3 of connector 3 & pin 4 of connector 4. Same for pin 3 of connector 5 & pin 4 of connector 6.

Then connect the pin 4 (the one furthest from the Arduino) of connector 1 to Arduino A2 hole, pin 4 of connector 3 to Arduino A1 hole, pin 4 of connector 5 to Arduino A0 hole.

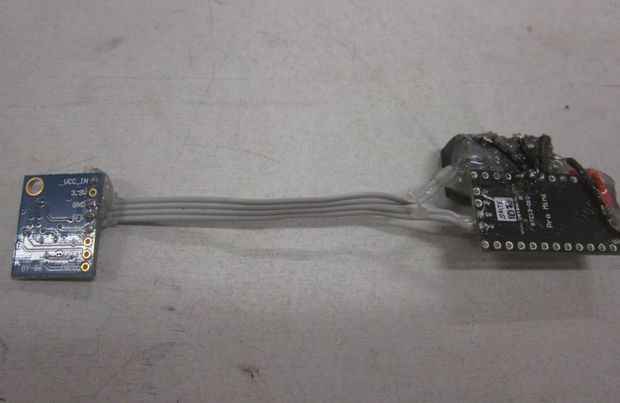
In summary, these are all the connections needed:

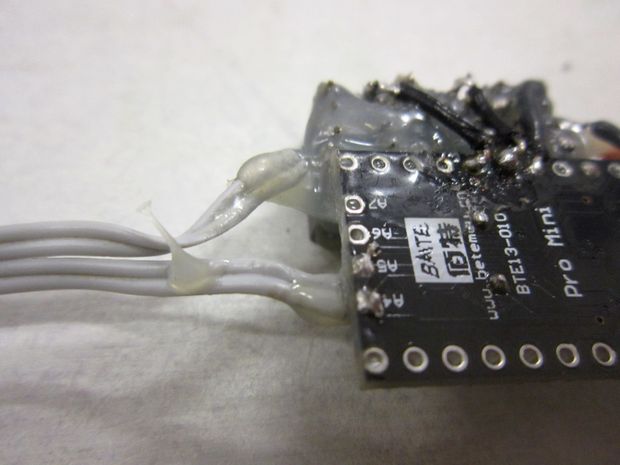
1. between Arduino GND and pin 1 of all 6 connectors,
2. between Arduino VCC and pin 2 of all 6 connectors,
3. between Arduino A2 and pin 4 of connector 1,
4. between Arduino A1 and pin 4 of connector 3,
5. between Arduino A0 and pin 4 of connector 5,
6. between pin 3 of connector 1 and pin 4 of connector 2,
7. between pin 3 of connector 3 and pin 4 of connector 4,
8. between pin 3 of connector 5 and pin 4 of connector 6,

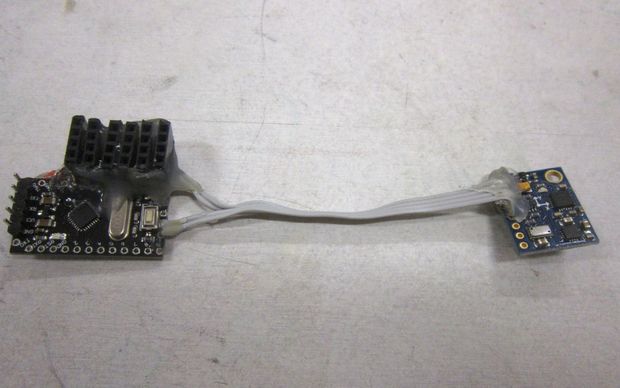
You'll notice pins 3 of connectors 2, 4 and 6 are left unconnected. Sorry for the pictures above, you can't see any of that there.

There's one more thing that I'd recommend doing: to avoid all of the current needed by the LEDs from passing through the very thing GND and VCC lines on the Arduino PCB, solder a piece of wire between the power connector's GND (the male headers we added in the previous step) to the GND wire under the 6 LED connectors, same for the VCC lines. You can solder these to the bottom end of the male headers that come out under the Arduino.

[[](http://cdn.instructables.com/FG6/CMS8/HY12NUTM/FG6CMS8HY12NUTM.LARGE.jpg)](http://cdn.instructables.com/FG6/CMS8/HY12NUTM/FG6CMS8HY12NUTM.LARGE.jpg)

[[](http://cdn.instructables.com/FS8/MH8K/HY12NUTQ/FS8MH8KHY12NUTQ.LARGE.jpg)](http://cdn.instructables.com/FS8/MH8K/HY12NUTQ/FS8MH8KHY12NUTQ.LARGE.jpg)

[[](http://cdn.instructables.com/FX7/DWY7/HY12NUTZ/FX7DWY7HY12NUTZ.LARGE.jpg)](http://cdn.instructables.com/FX7/DWY7/HY12NUTZ/FX7DWY7HY12NUTZ.LARGE.jpg)

[[](http://cdn.instructables.com/FNM/WCLV/HY12NUU6/FNMWCLVHY12NUU6.LARGE.jpg)](http://cdn.instructables.com/FNM/WCLV/HY12NUU6/FNMWCLVHY12NUU6.LARGE.jpg)

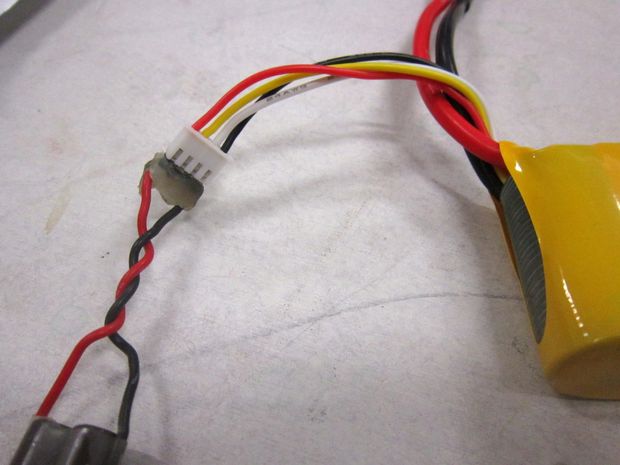
Most of the $10 IMUs (sensor boards) from E-bay / aliexpress / etc. have an onboard 3.3V regulator so you can connect a 5V source directly. You only need to connect four lines between the Arduino and the sensor board, and that accounts both for powering the board and for two-way communication.

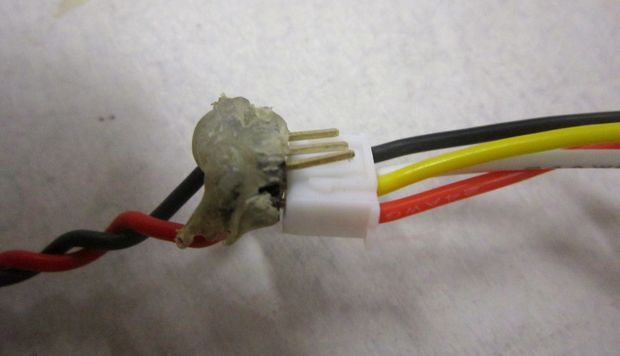
The pin names on the sensor board will be normally: GND, VCC\_IN, SDA and SCL, the other pins can be left unconnected. I connected my GND and VCC lines to the GND and VCC lines under my LED strip connectors.

The SDA line connects to the Arduino's SDA line which is labelled A4, and the SCL is labelled A5. On the popular, cheap Pro Mini clones, the holes for those pins are located on the same side of the board as the reset button, together with A7 & A8. I believe on the original Arduino Pro Mini the A4 & A5 pins are set apart on the longer side of the PCB close to A2 & A3.

I used a flexible 5cm 4-wire cable for the connection so that I can orient the sensor board independently from the Arduino if needed, since the sensor board is going to sense the wheel's orientation. I used no pin headers because I don't plan to disconnect the sensor board.

[[](http://cdn.instructables.com/F6R/JSGV/HY12NVWI/F6RJSGVHY12NVWI.LARGE.jpg)](http://cdn.instructables.com/F6R/JSGV/HY12NVWI/F6RJSGVHY12NVWI.LARGE.jpg)

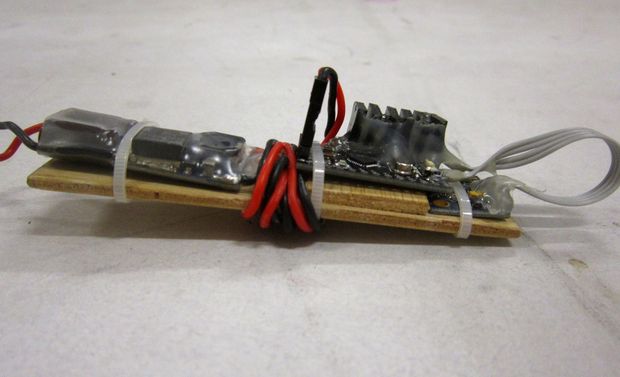
[[](http://cdn.instructables.com/FRD/90IA/HY12NVX5/FRD90IAHY12NVX5.LARGE.jpg)](http://cdn.instructables.com/FRD/90IA/HY12NVX5/FRD90IAHY12NVX5.LARGE.jpg)

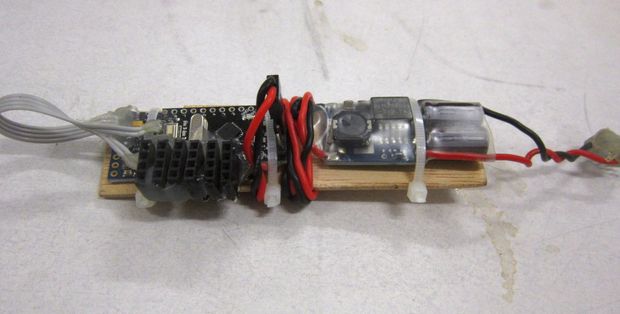
[[](http://cdn.instructables.com/FJO/V67K/HY12NVXP/FJOV67KHY12NVXP.LARGE.jpg)](http://cdn.instructables.com/FJO/V67K/HY12NVXP/FJOV67KHY12NVXP.LARGE.jpg)

Unless your regulator has an input connector compatible with the discharge connector of the battery (T-dean, XT60, etc.) it's probably easier to cut off whatever input connector it has and use male pin-headers to plug into the battery's charger jack. For a 2-cell LiPo the charger connector will have 3 wires, for a 3-cell there will be 4 wires, etc. We want to connect the regulator's red wire to the charger connector's red and the regulator's black to black. In other words we only need the first and the last wire from that connector and ignore those in the middle. But we'll use a pin row with all 4 pins (or as many as the charger connector has) to make a firmer grip between the "plug" and the "socket". So just solder the regulator's two input wires to the two pins at the ends of the row.

Since the headers doesn't have anything to prevent you from plugging them in reverse, be very careful to connect the red wire side to the battery's red wire.

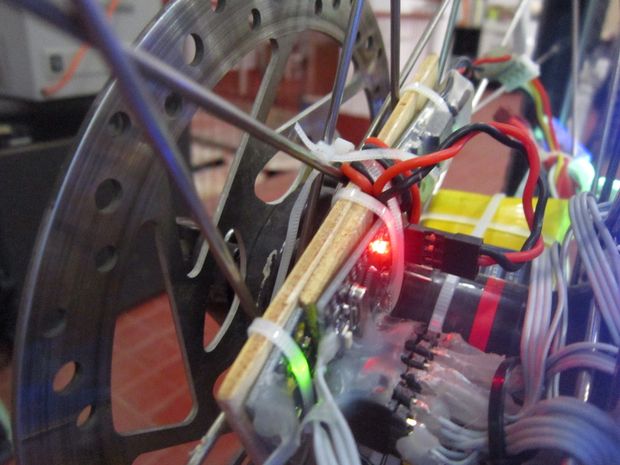
In my case I actually have both a 3-pin and a 4-pin male header strip so that I can use either a 2S or a 3S battery. Again sorry for not having a good pic.

[[](http://cdn.instructables.com/FV7/FNZ3/HY12NXY1/FV7FNZ3HY12NXY1.LARGE.jpg)](http://cdn.instructables.com/FV7/FNZ3/HY12NXY1/FV7FNZ3HY12NXY1.LARGE.jpg)

[[](http://cdn.instructables.com/FEJ/M7HD/HY12NY0A/FEJM7HDHY12NY0A.LARGE.jpg)](http://cdn.instructables.com/FEJ/M7HD/HY12NY0A/FEJM7HDHY12NY0A.LARGE.jpg)

[[](http://cdn.instructables.com/F1I/WGIY/HY12NY5D/F1IWGIYHY12NY5D.LARGE.jpg)](http://cdn.instructables.com/F1I/WGIY/HY12NY5D/F1IWGIYHY12NY5D.LARGE.jpg)

[[](http://cdn.instructables.com/F3A/1VKB/HY12NY60/F3A1VKBHY12NY60.LARGE.jpg)](http://cdn.instructables.com/F3A/1VKB/HY12NY60/F3A1VKBHY12NY60.LARGE.jpg)

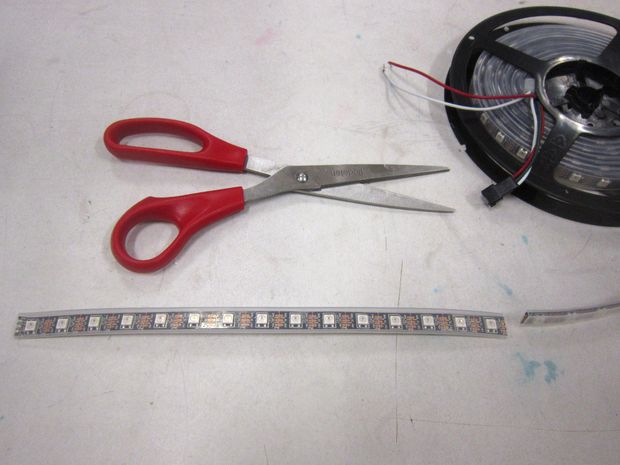
[[](http://cdn.instructables.com/F9T/36EJ/HY12NY33/F9T36EJHY12NY33.LARGE.jpg)](http://cdn.instructables.com/F9T/36EJ/HY12NY33/F9T36EJHY12NY33.LARGE.jpg)

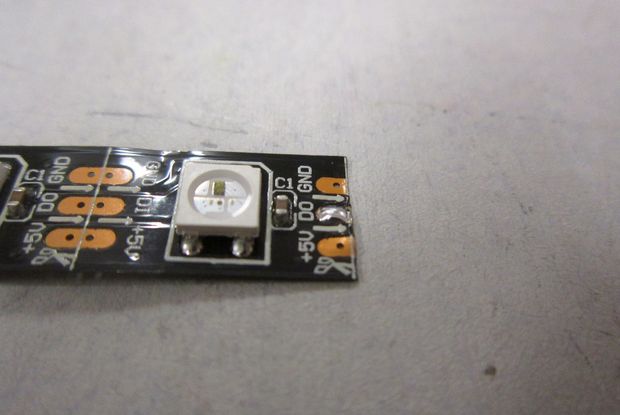
You might want to mount the whole thing now or later after testing that it works with the LED strips that we'll prepare next.

The mounting is all up to you, myself I just used a lot of zip-ties and pieces of 3-mm plywood to hold it all together. The sensor board should be mounted roughly perpendicular to the wheel axis or within the wheel "disc" plane, but doesn't need to align exactly -- the software can account for some misalignment.

Here are some additional notes on what to avoid:

1. Avoid squeezing the LiPo battery too hard with the zip ties and make sure the centrifugal force won't squeeze or pinch it against something once you're riding the bike and the wheel is spinning. These batteries are fragile and can be explosive or catch fire when damaged. The centrifugal force obviously acts in the direction away from the wheel hub and is smallest close to the centre. So mount it close to the centre and perhaps use a piece of something flat between the battery and the zip ties to maximise the surface on which the force is applied.
2. Avoid putting the sensor board too far away from the wheel hub too because the code needs to use the accelerometer readings for gyro drift compensation. My sensor board is about 6cm from the wheel centre. Too far away and, at higher speeds, the centrifugal force will saturate the accelerometer's range and the program won't be able to make any sense from the data.

[[](http://cdn.instructables.com/FEJ/X1JY/HY12NZAL/FEJX1JYHY12NZAL.LARGE.jpg)](http://cdn.instructables.com/FEJ/X1JY/HY12NZAL/FEJX1JYHY12NZAL.LARGE.jpg)

[[](http://cdn.instructables.com/F6P/01F1/HY12NZCW/F6P01F1HY12NZCW.LARGE.jpg)](http://cdn.instructables.com/F6P/01F1/HY12NZCW/F6P01F1HY12NZCW.LARGE.jpg)

[[](http://cdn.instructables.com/FHU/G7DY/HY12NZCZ/FHUG7DYHY12NZCZ.LARGE.jpg)](http://cdn.instructables.com/FHU/G7DY/HY12NZCZ/FHUG7DYHY12NZCZ.LARGE.jpg)

[[](http://cdn.instructables.com/F5K/FOKQ/HY12NZF8/F5KFOKQHY12NZF8.LARGE.jpg)](http://cdn.instructables.com/F5K/FOKQ/HY12NZF8/F5KFOKQHY12NZF8.LARGE.jpg)

I use 6 strips of LEDs on 6 spokes, you may want to use fewer for a start. This is how to prepare one strip.

Cut a pieces of tape 15-LEDs long, cut it off at the white dashed lines marked on the tape between the chips. You can use scissors. If the tape is protected by a silicon rectangular tube like mine just cute the tubing too. If the tape comes with some wires pre-soldered at the end, desolder them.

For now slide the tape out of the silicon jacket. Prepare a piece of 4 wire cable long enough to connect the LED strip to the sockets we added to the Arduino board. For the strips to be mounted on the same side of the wheel as the controller board, you can use a shorter cable, for the ones on the opposite side of the hub they need to be longer. Obviously an excess cable is less of a problem than a short cable.

You also need a single wire of about the same length as the LED strip to connect the output (DO pin) from the last WS2811 chip. Solder one end of the long wire to the DO pin. Them the four-wire cable's wires need to be connected in the following order:

1. GND pin on the input end (the first LED) of the strip,
2. 5V pin on the input end,
3. DO pin from the opposite end through the single wire we just added,
4. DI pin on the input end.

Solder a 4-pin male header on the other end of the cable. Also slide the jacket back on.

[[](http://cdn.instructables.com/F5K/OJG5/HY12O2FG/F5KOJG5HY12O2FG.LARGE.jpg)](http://cdn.instructables.com/F5K/OJG5/HY12O2FG/F5KOJG5HY12O2FG.LARGE.jpg)

[[](http://cdn.instructables.com/F6K/QJTP/HY12O2HA/F6KQJTPHY12O2HA.LARGE.jpg)](http://cdn.instructables.com/F6K/QJTP/HY12O2HA/F6KQJTPHY12O2HA.LARGE.jpg)

[[](http://cdn.instructables.com/F8D/N4OJ/HY12O2D0/F8DN4OJHY12O2D0.LARGE.jpg)](http://cdn.instructables.com/F8D/N4OJ/HY12O2D0/F8DN4OJHY12O2D0.LARGE.jpg)

[[](http://cdn.instructables.com/FUM/X63G/HY12O2J9/FUMX63GHY12O2J9.LARGE.jpg)](http://cdn.instructables.com/FUM/X63G/HY12O2J9/FUMX63GHY12O2J9.LARGE.jpg)

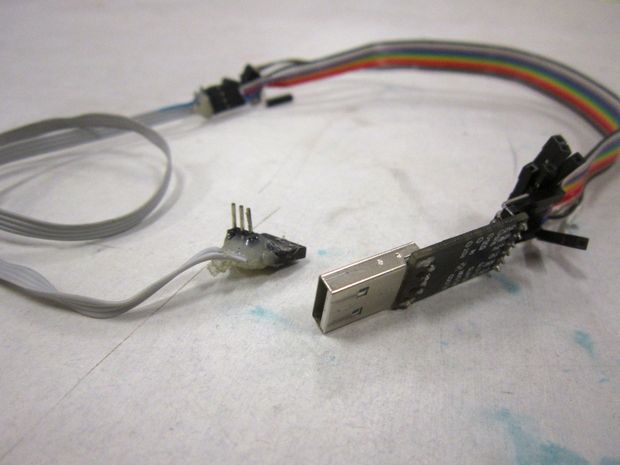
[[](http://cdn.instructables.com/FL3/M5ST/HY12O2FB/FL3M5STHY12O2FB.LARGE.jpg)](http://cdn.instructables.com/FL3/M5ST/HY12O2FB/FL3M5STHY12O2FB.LARGE.jpg)

Before mounting the LEDs you may want to secure the ends of the silicon tube around the LED strip with some hot glue, try to inject some of it into the tube. This will at least prevent humidity from staying inside the LED strip, so you don't have to take them all off the wheel when there's a chance of rain. You'd probably still want to leave the other electronics at home though, and you definitely want to take out the battery.

If you know of a good way to waterproof the whole setup, please comment. I hear that WD-40 is great at protecting SMD electronics from water, just spray some of it over all the PCBs. Note however that you shouldn't spray anything into the barometer chip aperture on the sensor board or cover it with hot-glue unless you don't plan on ever using it.

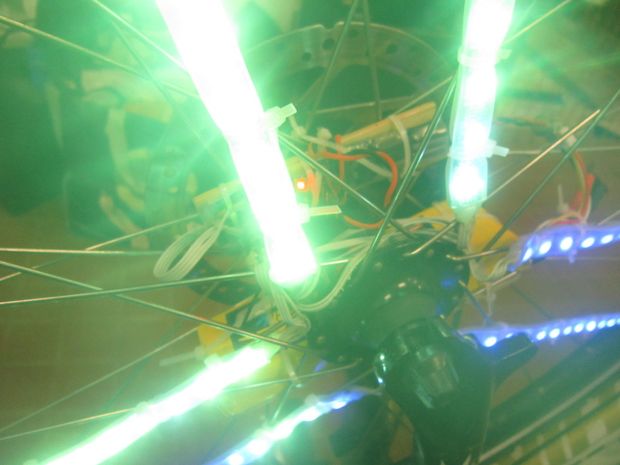
I like to secure all the hand soldered elements like the male and female headers, with hot-glue to also add some structural support for the connectors and make them easier to grab with fingers or pliers without risk of breaking the solder joints. With the hot-glue gun the blobs always come out quite ugly but you can later use a hot-air gun with a wide nozzle to melt the glue, force it into holes or even spread it over the whole Arduino PCB.

Finally the LED strips are just zip-tied to the spokes, make sure they don't slide along the spoke. In my case I mounted the 6 strips on pairs of adjacent spokes on three opposite sides of the wheel due to the symmetry present in the spoke pattern. I suppose this spoke pattern is quite standard. Having strips close together helps make POV text readable at a low speed.

[[](http://cdn.instructables.com/FBR/LWQP/HY12O5QB/FBRLWQPHY12O5QB.LARGE.jpg)](http://cdn.instructables.com/FBR/LWQP/HY12O5QB/FBRLWQPHY12O5QB.LARGE.jpg)

[[](http://cdn.instructables.com/FDJ/G07G/HY12O5QI/FDJG07GHY12O5QI.LARGE.jpg)](http://cdn.instructables.com/FDJ/G07G/HY12O5QI/FDJG07GHY12O5QI.LARGE.jpg)

[[](http://cdn.instructables.com/FDX/9O0C/HY12O5QD/FDX9O0CHY12O5QD.LARGE.jpg)](http://cdn.instructables.com/FDX/9O0C/HY12O5QD/FDX9O0CHY12O5QD.LARGE.jpg)

[[](http://cdn.instructables.com/FJV/EP15/HY12O5QA/FJVEP15HY12O5QA.LARGE.jpg)](http://cdn.instructables.com/FJV/EP15/HY12O5QA/FJVEP15HY12O5QA.LARGE.jpg)

[[](http://cdn.instructables.com/FQI/69YY/HY12O5Q7/FQI69YYHY12O5Q7.LARGE.jpg)](http://cdn.instructables.com/FQI/69YY/HY12O5Q7/FQI69YYHY12O5Q7.LARGE.jpg)

Check out my firmware code from <https://github.com/balrog-kun/blinkenbike>, not everything works yet -- check the github page for current status. The sketch depends on the following Arduino libraries: I2Cdev, MPU60X0 (both ship with the FreeIMU library), and WS2811. The WS2811 library needs to be the one from <https://github.com/balrog-kun/ws2811-multi> because that version supports significantly faster updating of the LED's colours.

Once the libraries are installed, just load wheel.ino into the Arduino IDE and upload it to the Pro Mini board.

If your laptop doesn't have USB overcurrent protection, make sure the electronics are powered from the LiPo battery during flashing, or disconnect the LEDs. For development I used a long cable that connects the Arduino's GND, TXD, RXD and DTS lines to the USB-to-Serial adapter, but the VCC (and GND) to the voltage regulator & battery. The cable is about 70cm allowing the wheel to make a few turns without disconnecting from PC.