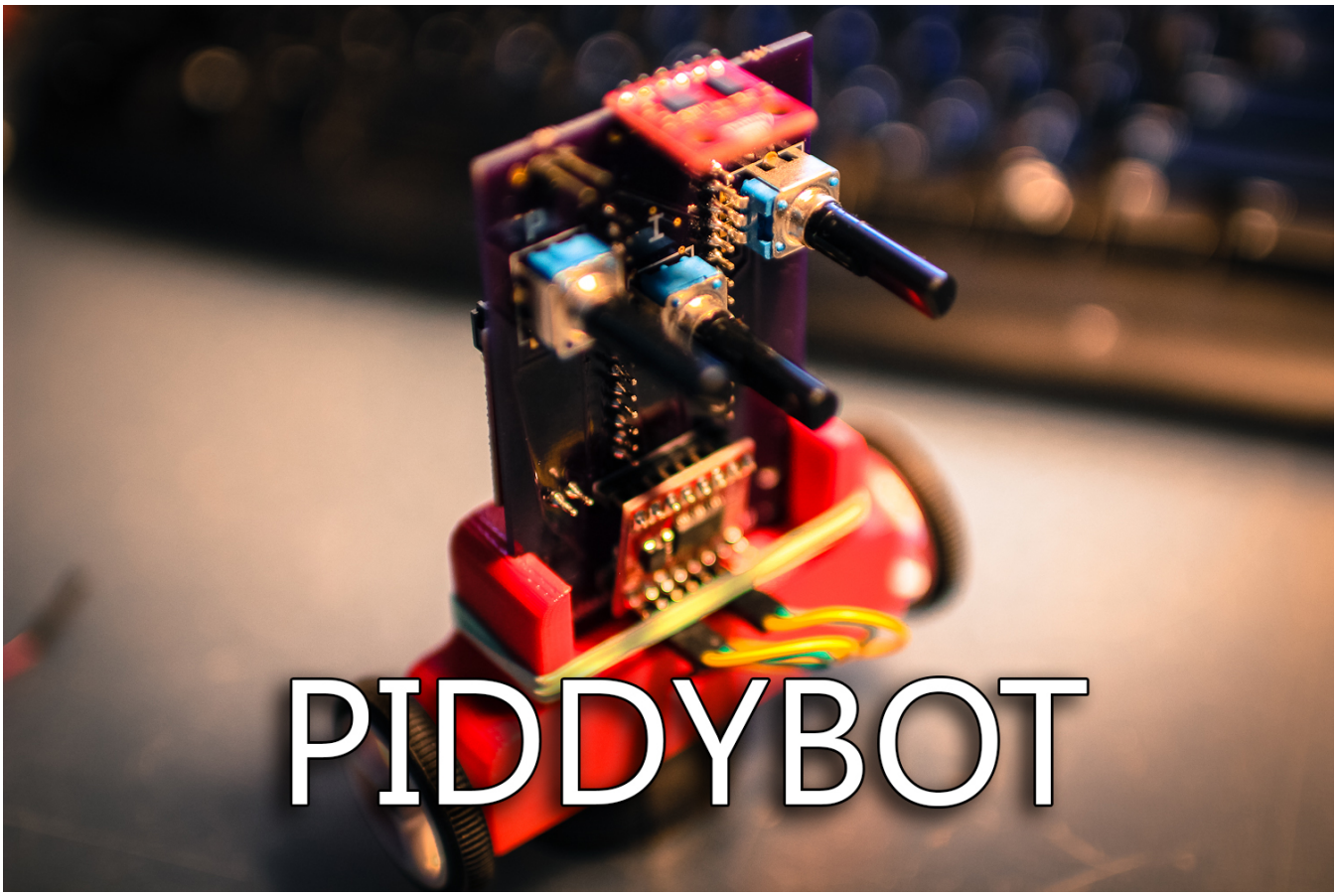
PIDDYBOT – A Self Balancing Teaching Tool.

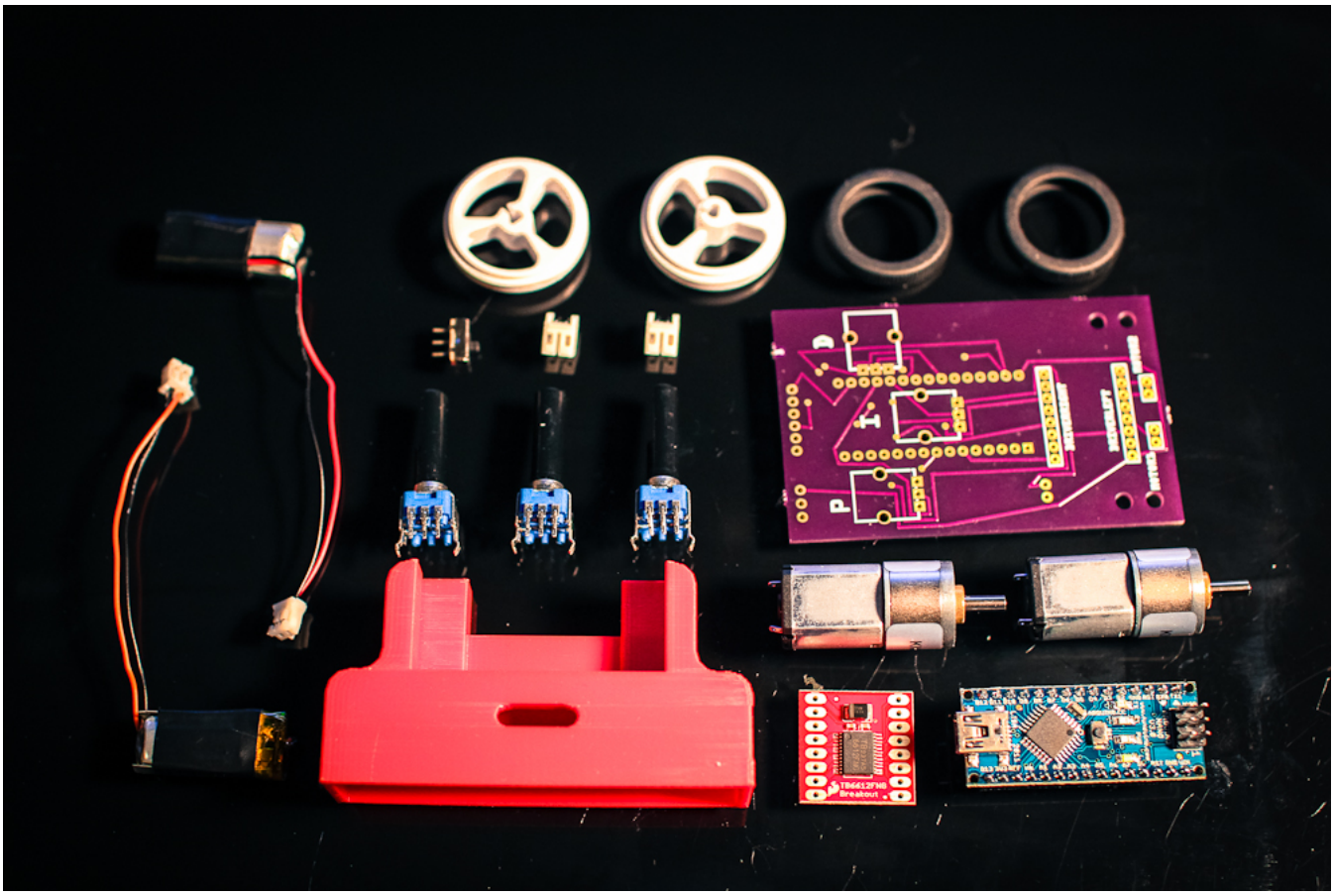


After I built the tiny balancing robot using an IR sensor for stabilization, there was a comment on the youtube video about how it would be a good thing for kids to build and learn about PID controls. I thought that was a great idea, the only problem was that that tiny robot was the simplest of balancing robots. It was just an on off switch for telling the motor which way to move. There was no actual PID implementation in

that system. So that got me thinking about how it would have been really cool if in one of my classes where I was learning about control theory I had a robot that actually let you see the changes in a PID system in real time. I decided to take it upon myself to create such a robot.

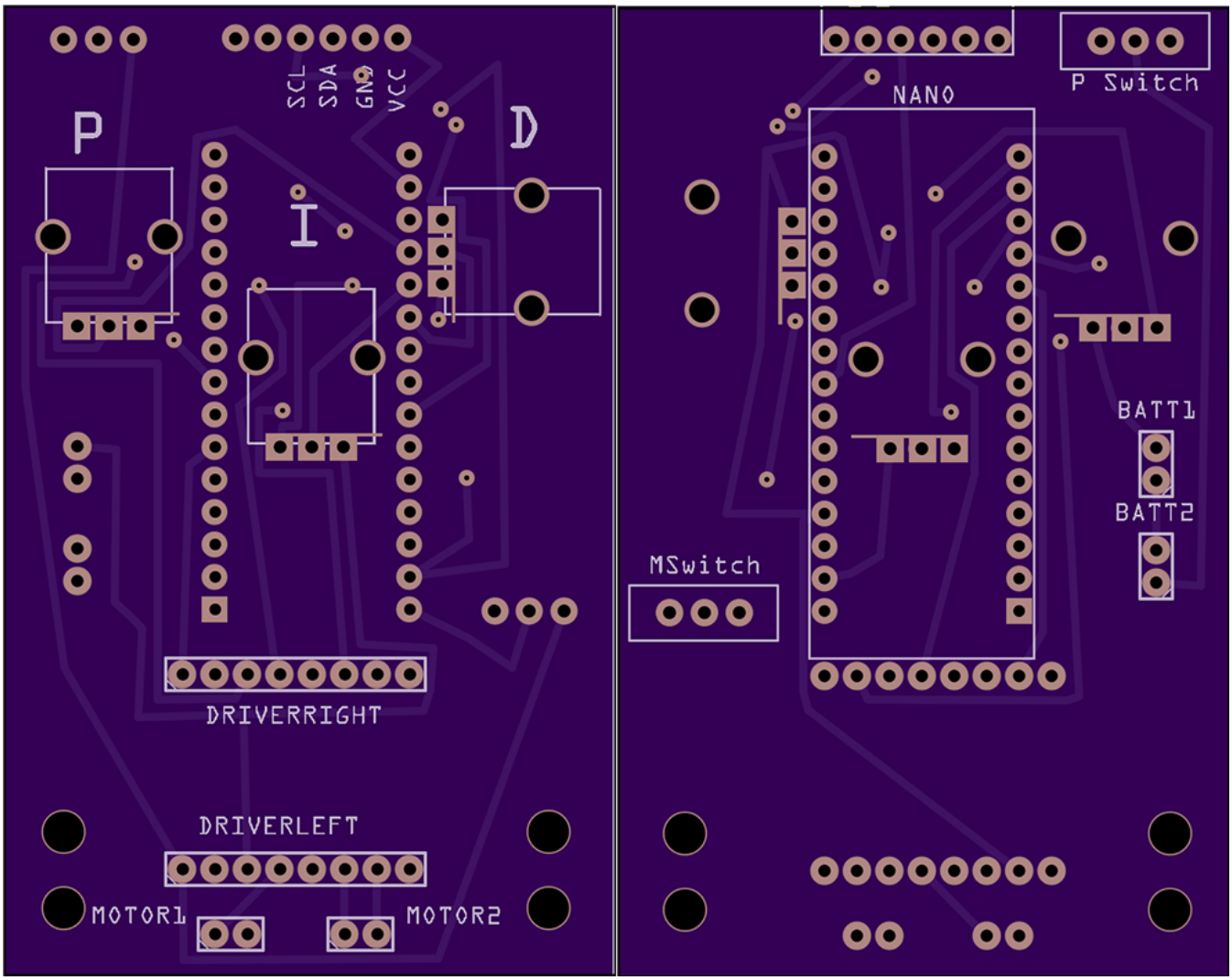
The basic plan I had in my head was to make it as simple and small as possible – while making it possible for anyone to build themselves. The components consist of:

- 3 Potentiometers (might become 4)
- Arduino Nano (Obvously!)
- A custom PCB – on OSHpark – Want to Mod it before Buying? Frizting File Here
- 2 Geared Motors 26:1 or similar
- Sparkfun 1A Dual Motor Driver Board
- Wheel set – Bigger Might work Better…
- 6 DOF IMU from SparkFun
- 3D printed Body. – Thingiverse
- 2 x Battery Connectors
- 2 x Lithium Batteries 110mAh
- 4 x M2 Screws for Motor Mounting
- 2 x Female Jumper Cables
- 2 x SPDT Slide Switches
- 4 or more Right angle Headers
- A bunch of Headers.
- My PIDDYBOT Arduino Code – V3 Now with Location PID Control – PIDDYPROGRAM_V3_Upload
- Assembly Instructions… Kind Of — NEW!



Always be Knolling (No IMU Whoops!)

First I figured out how small I could make the PCB and still have everything fit. I actually made the PCB a long time ago and it was one of the first I made, so I probably could have made it smaller; I'm happy with the size now.



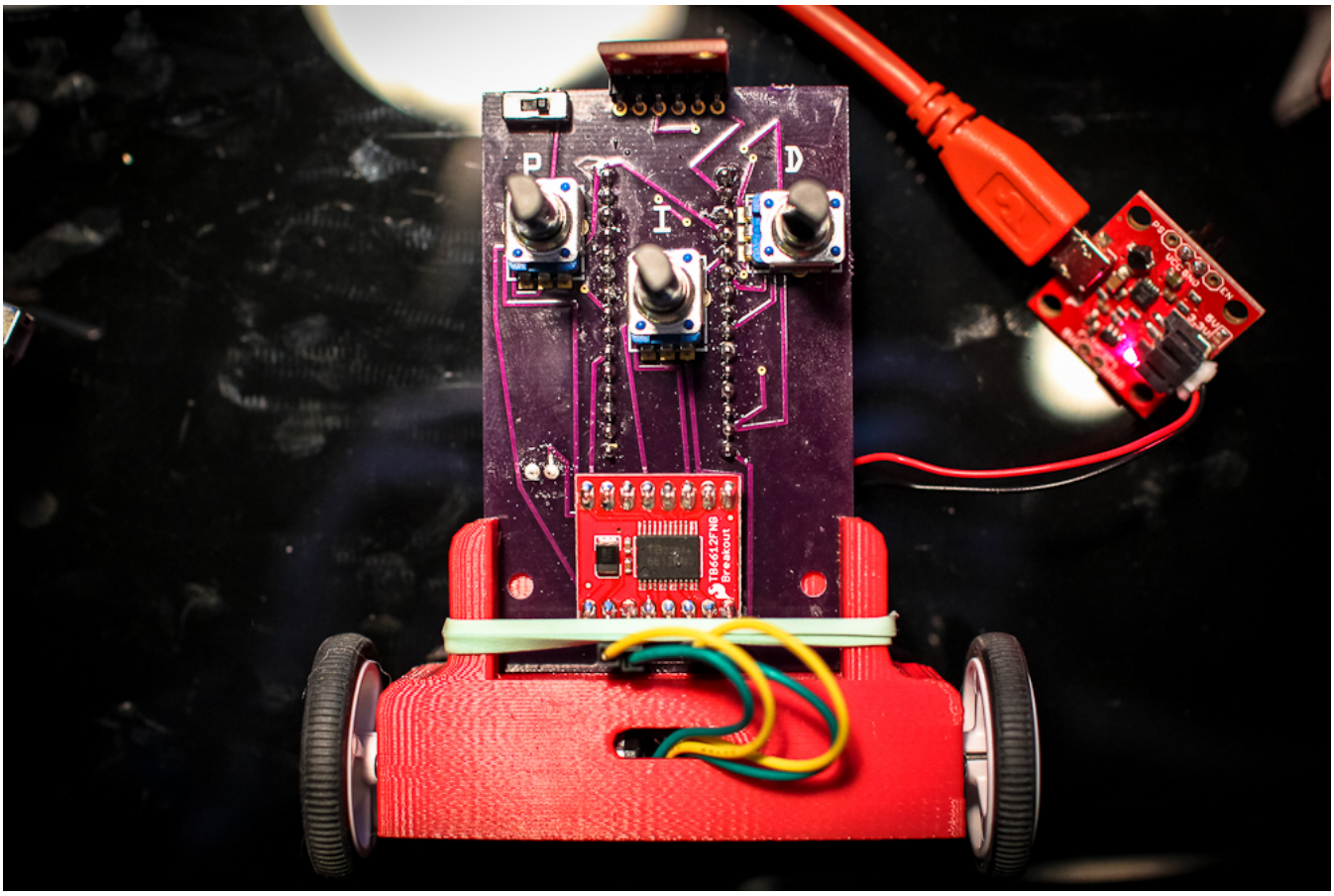
Front – Back OSHpark

I designed it to have the Arduino just basically fit right into the center of it, that way all the connections are good and secure and it just makes it easier. I know I could have designed a micro-controller onto my own board, maybe in the future. The size of the board determined the size of the body. I made it so the PCB was basically the only thing requiring any assembly. I also made mine with the IMU was facing upwards. Now, I don't think it actually matters which way you mount the IMU as long as its not transversely angled with regards to the pivot point. All you would have to do is alter the code for the correct reading to be used. The reason two batteries are used is to increase the voltage. I was using SparkFun's voltage booster to use a single battery, but when it hits 600mA it shuts off. When the robot suddenly jerks the motors, it can possibly go higher than the 600mA limit and shuts the system down. By having two batteries you get 3.7+3.7 = ~7.4V, which is fine for all of these components, and the IMU is run off of the Arduino(3v3).

Next was the body(chassis) design. I knew what motors I was going to be using, and thankfully the dimensions were readily available so I started up my favourite modelling software and went to town. I somehow designed it perfectly(this never happens) so that when the PCB was put into the slot, it was snug enough that no mounting hardware was required. If your print is a little loose, there are mounting holes in the PCB. I actually messed up a little with the battery tray in the back, its too close so for these to fit in with the Arduino there; you need to use an elastic or something to secure them, no big deal. The updated model on Thingiverse is corrected for more space, so they will fit.

Chassis

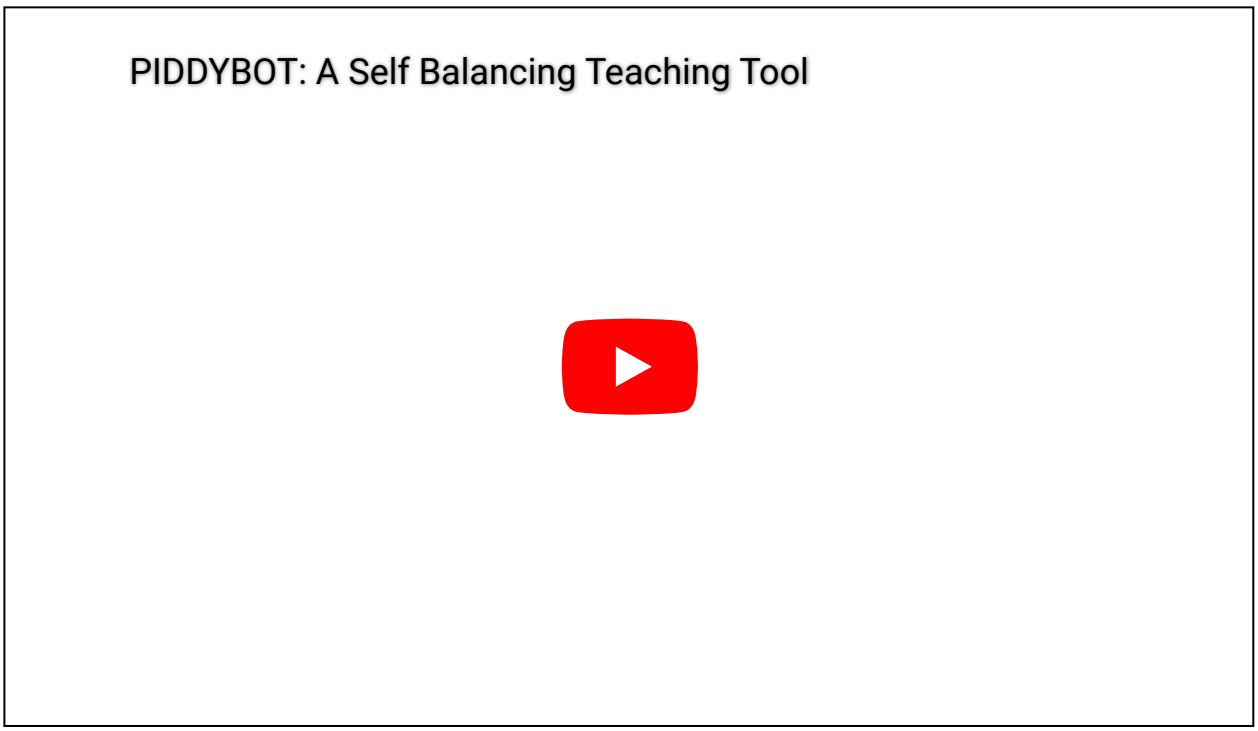
Before you install the motors you need to take two jumper cables and cut them in half and solder them onto the motor connections. These will be the leads from the motors to the motor controller. It doesn't really matter which you solder to, as you can just switch them around on the PCB connections. Then its just as simple as putting the wires through the hole and screwing the motors onto the chassis. The space will be tight near the connections, but should fit perfectly. I opted for two motors in this design so in the future I could turn it into a remote controlled or autonomous vehicle. It needed to have the ability to turn. I'm planning on making a xBee version of the PCB that accepts an xBee module for remote controlling.



Assembled 2

Now on to the code... This is in no way finished, but it does allow for the tuning of the PID and all that. It is not very neat or tidy and its assembled from several different projects. I have to give thanks to this bildr tutorial as well as the FreeSixIMU Library which was modified from this FreeIMU Library from varesano. You will need to download the library from the bildr tutorial. You can download my program here. PIDDYPROGRAM_V2_Upload – It is pretty rough, if there are any questions at all, don't hesitate to ask! – There is a little extra feature in the code, it actually has a sort of positioning return system even without encoders. It basically takes the amount of time the motors are tuning in each direction and the speed and figures out how far away it is from its first position, you can see this in the video. I will be updating the code, but I probably won't be making a GitHub folder for it. If there are major updates I will post them here and announce them on twitter. Be sure to follow!

Once you upload it you can start figuring out how to tune your own self balancing PID system. Don't get too frustrated!



Share

This entry was posted in 3D Printing, Arduino, CAD, Electronics, Projects, Robotics, Sparkfun, Uncategorized and tagged 3dprinting, Arduino, balancing, electronics, robot, robotics, self, sparkfun on January 16, 2014 [http://www.idlehandsproject.com/piddybot-a-self-balancing-teaching-tool/] .