

Electric Skateboard

Rahul Iyer, June-August 2015

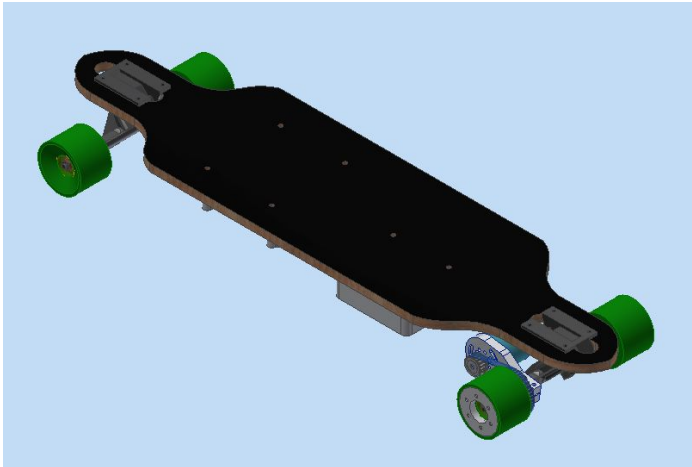


Figure 1: The electric skateboard concept in the Autodesk Inventor CAD software (left) and the completed, working device (right)

Overview

In the summer before my senior year, I made it a personal goal to design and build an electric skateboard completely from scratch. Over the course of three months, I designed, manufactured, and assembled the skateboard's drive system and its custom control and power distribution electronics, and wrote control software to translate joystick input to drive the board. I plan to make the skateboard's CAD model, electronics, and control software completely open-source.

Technical Specifications

- Maximum Speed (software limited) : 23 mph
- Drive Motor: *Turnigy Aerodrive SK3 5055-430kV Brushless Outrunner Motor*
- Drive Transmission System: *5mm HTD timing belt and 2.5x pulley reduction*
- Power Supply System: *2 x Zippy Flightmax 3S LiPo Battery, 5000mAh (in series)*
- Electronic Speed Controller: *Shenzhen Favourite Young Wolf 120Amp Car ESC*
- Drive Controller: *Arduino Uno R3*
- Remote Control Device: *Nintendo Wii Nunchuck*

Software Tools

- Computer-Aided Design: *Autodesk Inventor Professional*
- Computer-Aided Manufacturing: *Autodesk HSMWorks Professional*
- 3D Printing Program: *ReplicatorG*
- Electronic Circuit Design: *Fritzing*

Machinery

- 3D Printer
- CNC Mill
- Benchtop Manual Mill
- Drill Press
- Lathe

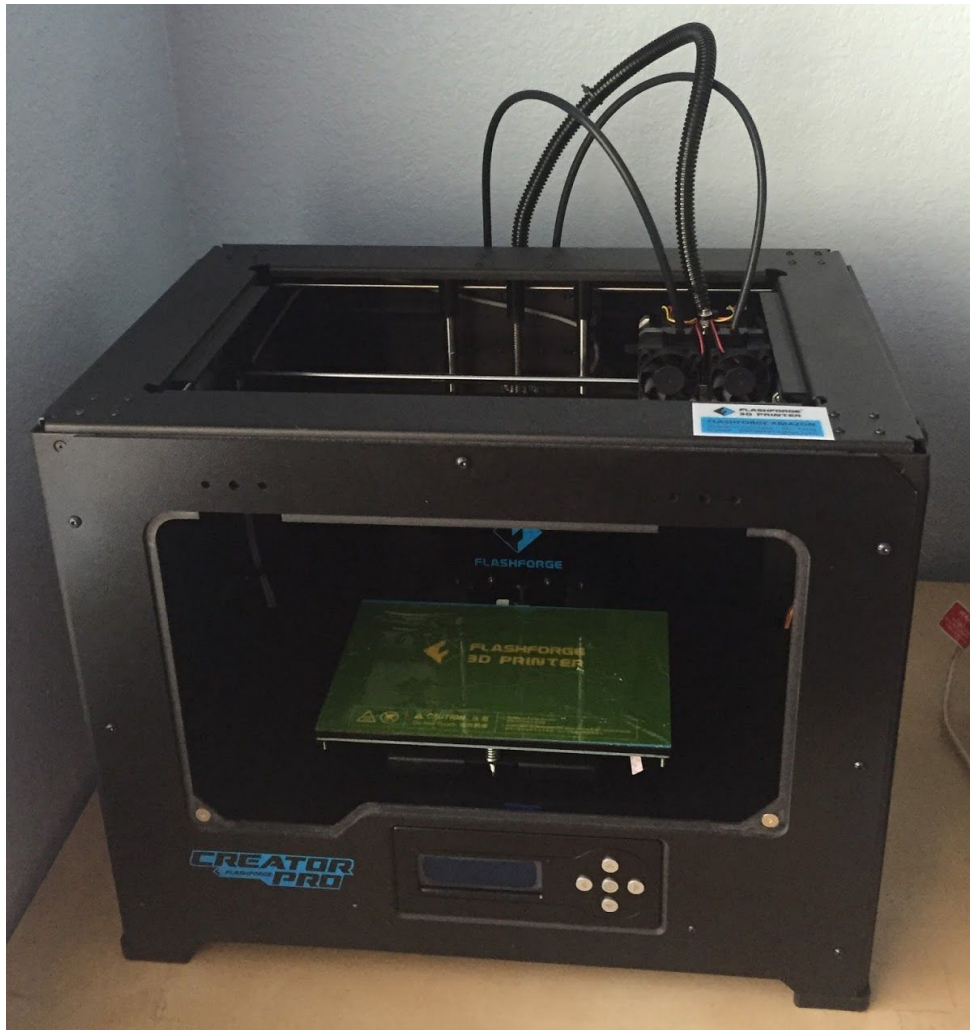


Figure 2: The Flashforge Creator Pro 3D Printer I used



Figure 3: Exploded view of the Electric Skateboard. The skateboard's parts were made through conventional and additive manufacturing technologies.

Mechanical Subsystem

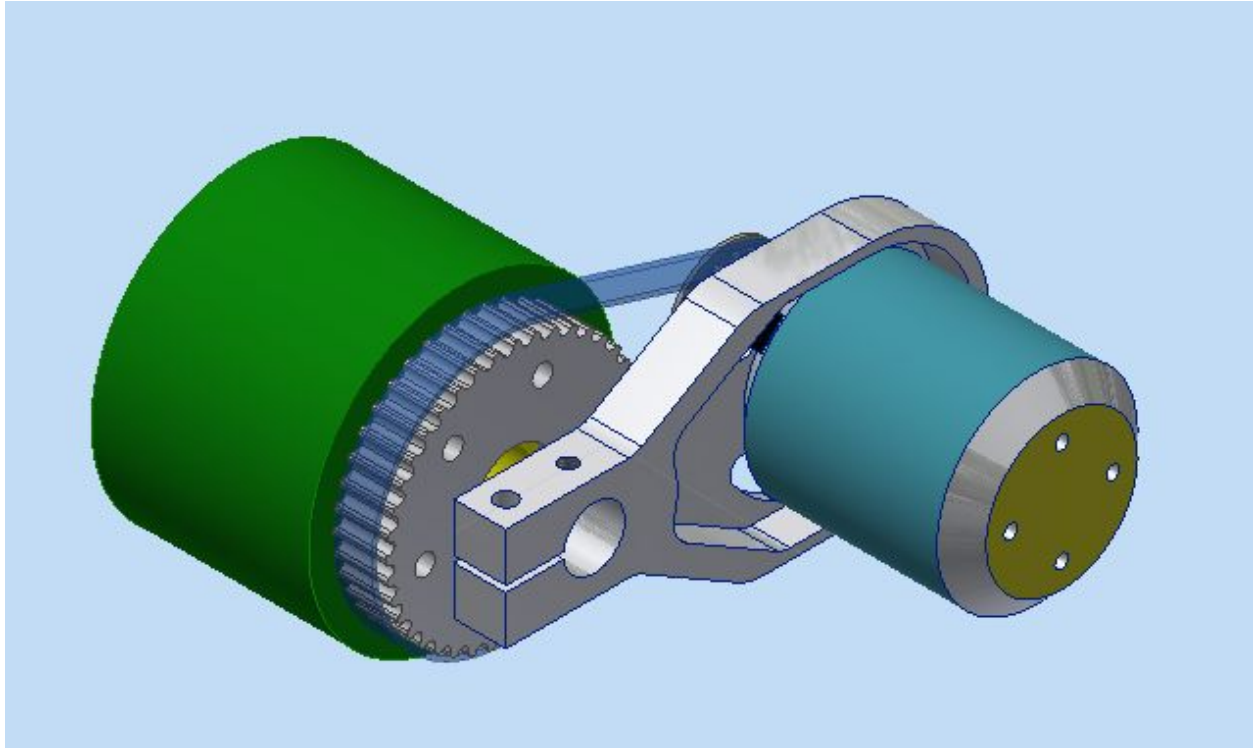


Figure 3: CAD model of the skateboard's belt transmission and motor mount. The outrunner motor mounts to a custom mounting bracket that is secured to the skateboard's front truck.

The mechanical subsystem of the skateboard is the transmission assembly that drives the skateboard. The brushless outrunner motor drives one of the skateboard's wheels through a timing belt system, which consists of a timing belt, one large pulley, and one small pulley.

Gearing Calculations		
Free Speed Wheel Velocity	168.9	ft / sec
Software Speed Limit (Factor)	2	ul
Target Wheel Velocity (Software Limited)	25	mph
Pinion Sprocket	16	teeth
Output Sprocket	40	teeth
Reduction	2.5	ul
Wheel Speed	67.5	ft/sec
Wheel Speed (free)	46.1	mi/hr
Wheel Speed (Software Limited)	23.0	mi/hr

Table 1: Gearing Calculations for the skateboard. I defined a target speed, and then selected a pair of pulleys that would move the wheels at that speed.

I decided to use a timing belt mounted on two pulleys instead of a pair of spur gears to offset the motor from the skateboard's trucks. To calculate the reduction in the belt pulley system, I calculated the ratio between the motor's maximum output revolutions per minute (RPM) and the RPM I wanted the skateboard's wheel to travel at.

To choose the pulleys for the drive system, I had to consider two parameters: the ratio between the pulleys' teeth, and the outside diameters of the larger pulley. At a constant pitch--the space between the teeth of a pulley--the number of teeth of a pulley is directly proportional to its pitch diameter, the midline diameter of a pulley's teeth where the teeth and the belt mesh. Therefore, I selected a pair of pulley's whose ratio of teeth were the same as the reduction I previously calculated, because this would mean that the ratio of the pulley's pitch diameters would be the same. However, if I selected a pulley whose outside diameter was larger than the diameter of the wheel, the skateboard's point of contact with the ground would become the pulley instead of the wheel.

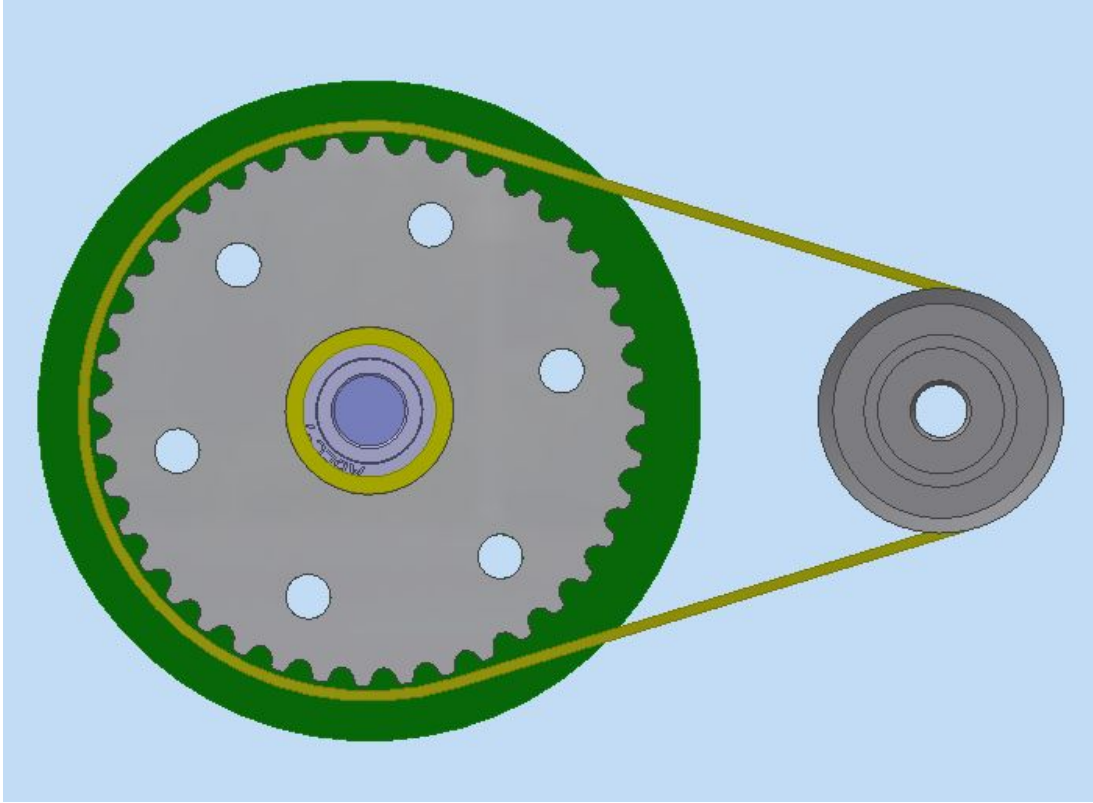


Figure 4: CAD model of the belt run that connects the pinion pulley (right) to the output pulley (left)

I decided to use a 16-tooth pinion pulley and 40-tooth output pulley pair (a reduction of 2.5) because this combination best fit the design parameters. Using Autodesk Inventor, I made sure that neither the pulley nor the belt touched the ground when mounted on the skateboard wheel.

To keep the motor rigid while maintaining a constant center-to-center distance between the pinion and output pulleys, I mounted the motor to the skateboard's truck with a custom bracket.

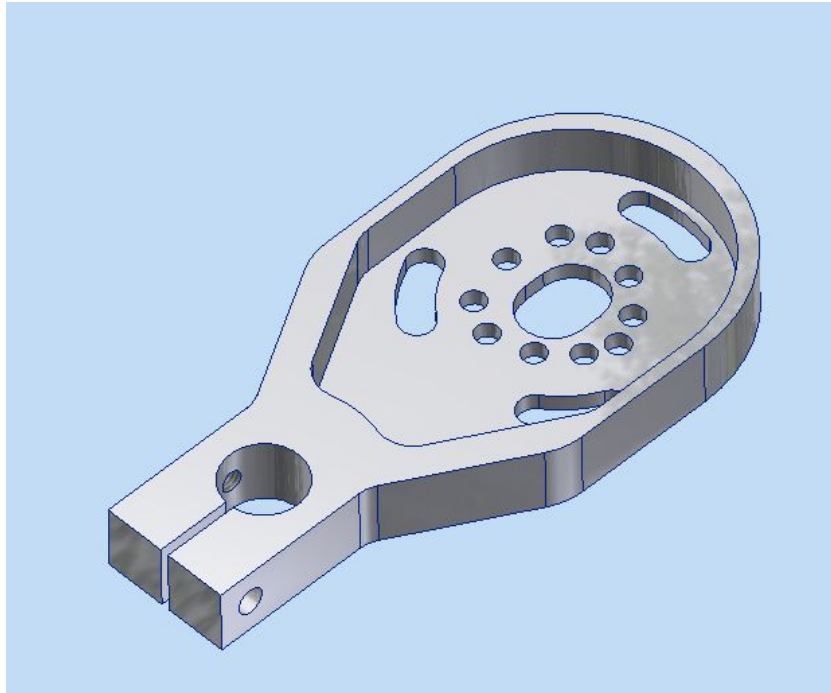


Figure 5: CAD model of the motor mounting bracket

To allow the bracket to clamp onto the skateboard truck, I added a slot that intersected the mounting hole that slid onto the truck. When the screw that runs perpendicular to the slot is tightened, the two sides of the slot are pulled together, which causes the mounting hole's diameter to decrease and tighten around the truck.

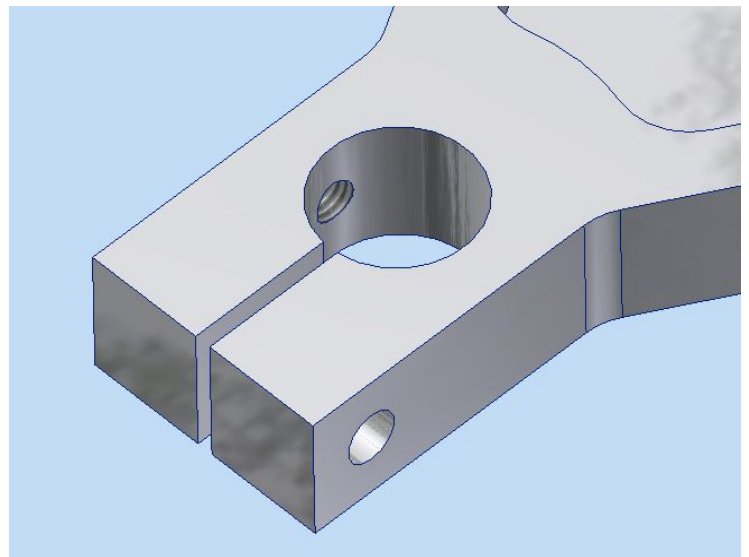


Figure 6: The clamp features on the bracket hold it firmly to the skateboard truck

Technical drawing of a mechanical part, likely a cross-section of a cylindrical component. The drawing shows concentric circles and radial lines. Key dimensions and features include:

- Outer diameter: 35.62
- Inner diameter: 37.69
- Radial distance from center to outer edge: 33.23
- Radial distance from center to inner edge: 220
- Central feature labeled: $fx: .984$
- Small dimension: 0.03

In my Inventor sketch, I specified different center-to-center distances for each pattern and slot dimension for my tensioning scheme. The center of the sketch indicate the center of the motor in the different patterns. The pattern of smaller circles around the center hole indicates the locations of the new holes.

Electronics Subsystem

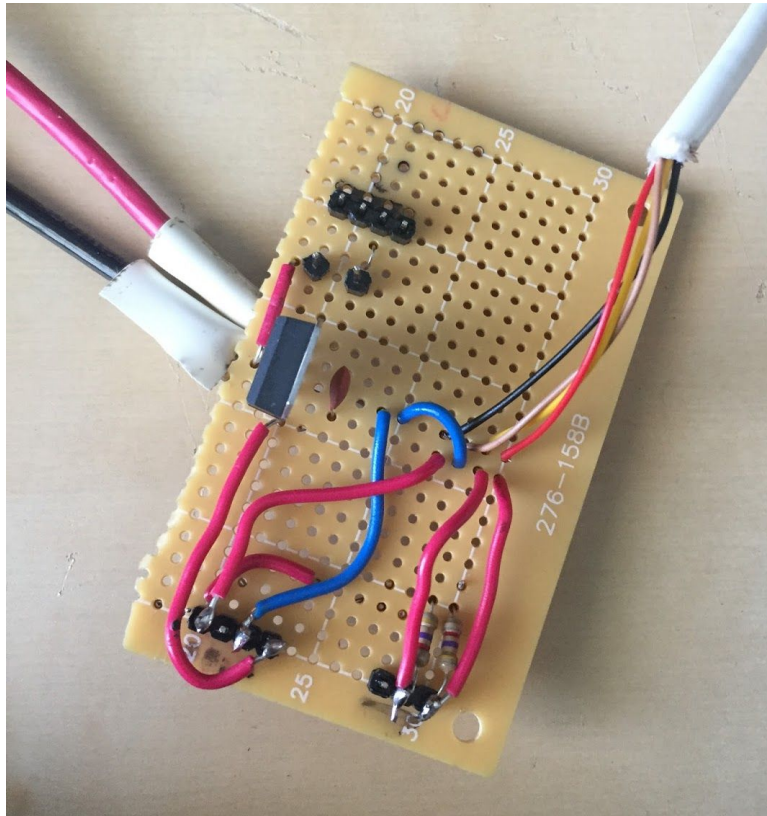


Figure 8: The custom Arduino shield powers the Arduino, connects the Wii Nunchuck, and bridges the Arduino to the ESC

I decided to use the Arduino Uno as the brains for the skateboard because it is versatile, can connect to multiple peripheral circuits, and can be easily re-programmed for different or additional functionality. I made a custom expansion board, or “shield,” for the Arduino that powers it with regulated +9 Volts, attaches the Wii Nunchuck to the Arduino I²C SDA and SCL pins, and attaches the ESC’s signal wire to a Pulse Width Modulation (PWM) on the Arduino.

I started out by drawing the circuit for the shield in Fritzing. The +22.2V and Ground wires from the two LiPo batteries in series first connected to the Input and Ground leads on the 9V Voltage Regulator. The Output lead of the Voltage Regulator was connected to the Voltage In pin on the Arduino, and the Ground lead of the Voltage Regulator was connected to the Ground pin on the Arduino. I also added a .1 μ F capacitor between the Ground and Output leads of the Voltage Regulator to smoothen the output to the Arduino.

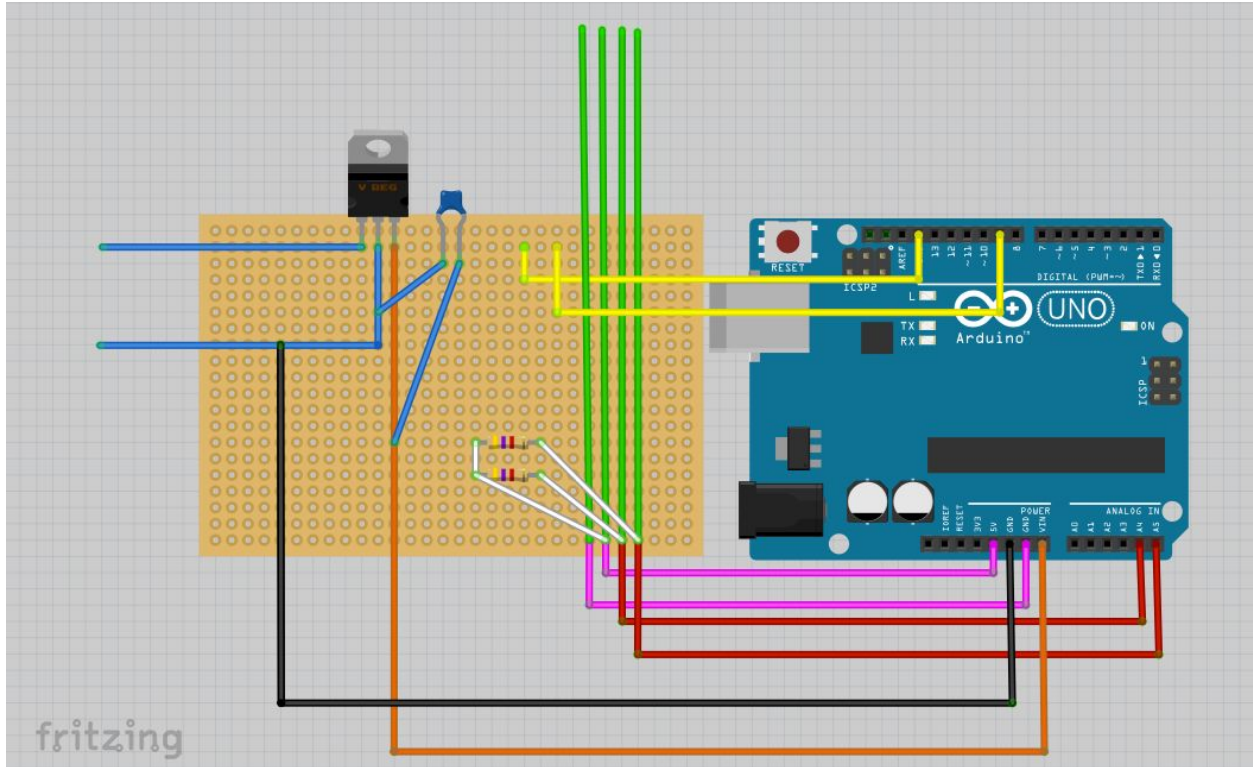


Figure 9: Fritzing diagram of the connections between the Arduino and my custom shield. I color-coded the wires to differentiate connections to different peripherals.

Because the Wii nunchuck communicates to a control board via the I²C protocol, I connected the signal pins from the nunchuck it to the Arduino's I²C pins, SDA pin 4 and SCL pin 5, with two 470kΩ pull-up resistors to connect the pins to a "high" signal of +5V.

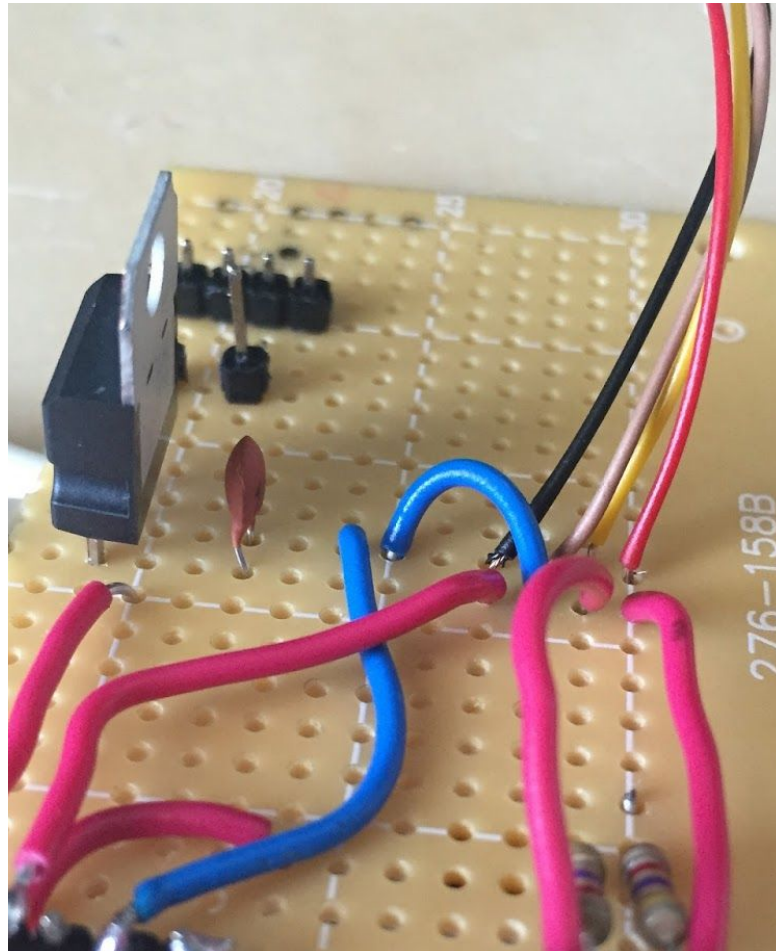


Figure 10: The two pull-up 470kΩ resistors (left) and the 9V Voltage regulator with the .1μF smoothing capacitor

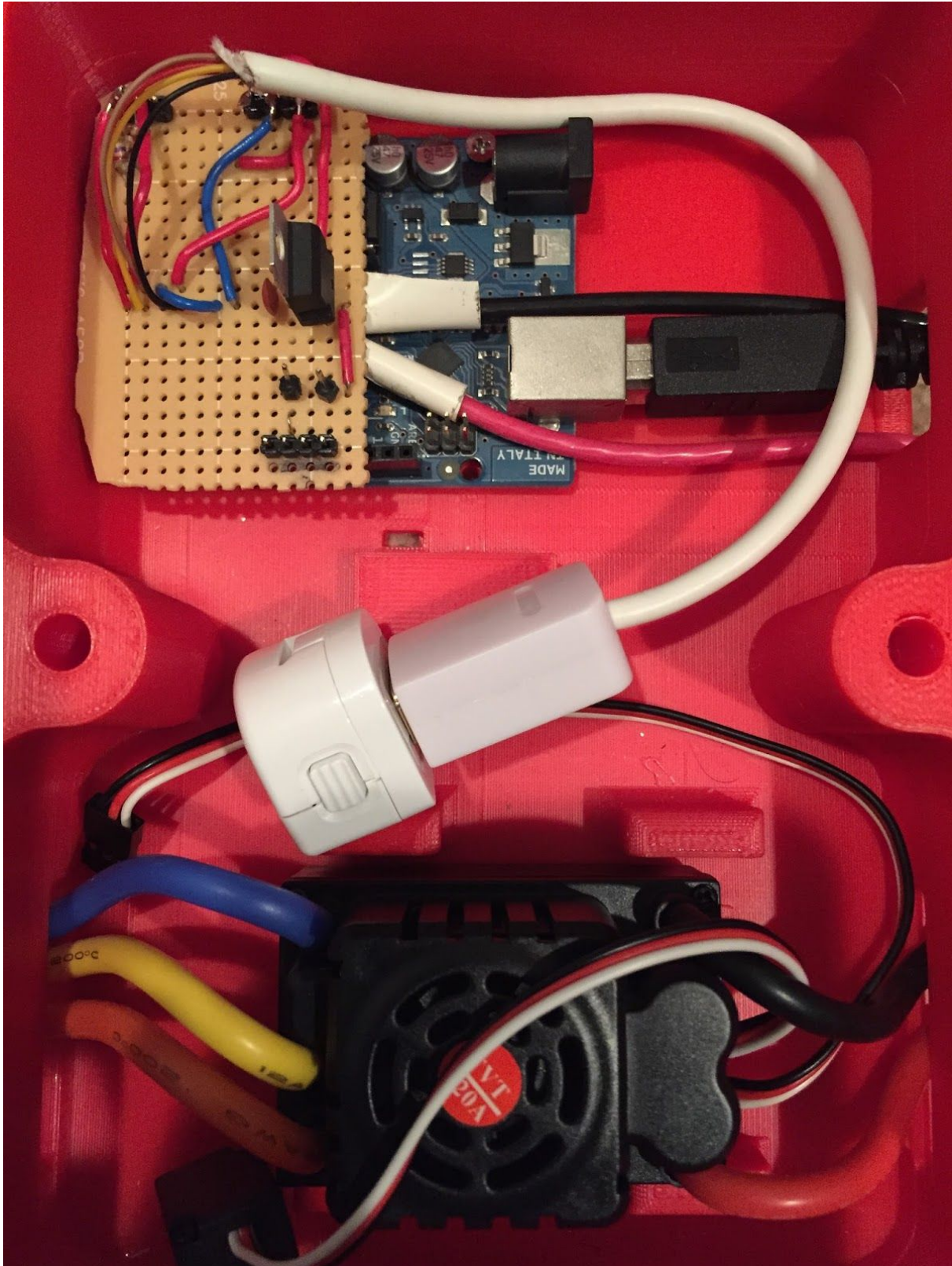
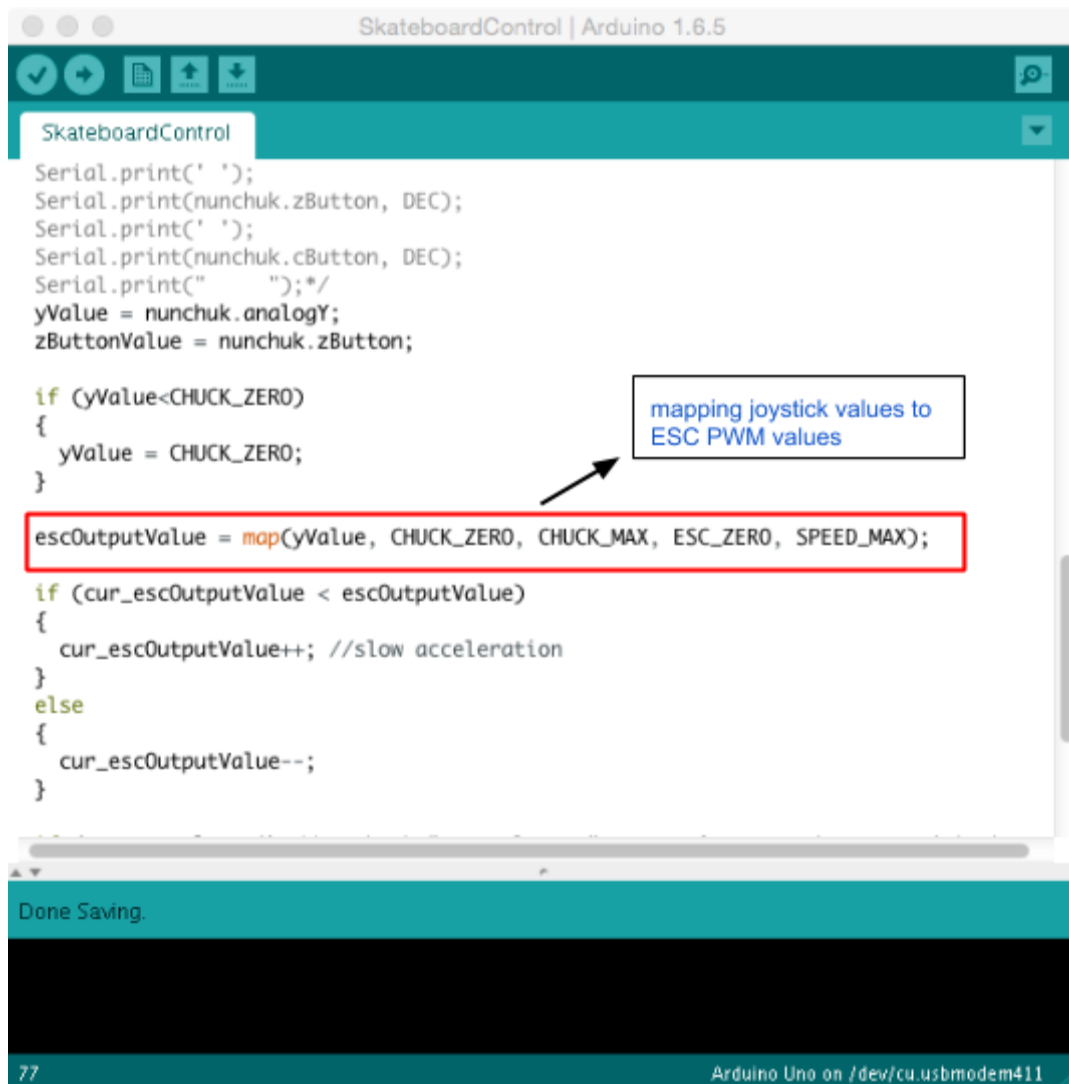


Figure 11: The completed electronics box with the Arduino (top), Wii Nunchuck Receiver (middle), and the Electronic Speed Controller (bottom)

Control Software

The C++ program I wrote provided instructions to the Arduino to control the speed of the motor based on the Wii Nunchuck's joystick input. The "ArduinoNunchuk" open-source library provided the methods to retrieve bytes of data from the Nunchuck via I²C.

I mapped the range of values that the Nunchuck joystick provided to a range of speed values for the ESC. By testing different upper limits on the ESC speed values, I determined the range of speed values that would make the skateboard both fast enough and safe for city travel.



```
SkateboardControl | Arduino 1.6.5

Serial.print(' ');
Serial.print(nunchuk.zButton, DEC);
Serial.print(' ');
Serial.print(nunchuk.cButton, DEC);
Serial.print("    ");*/
yValue = nunchuk.analogY;
zButtonValue = nunchuk.zButton;

if (yValue<CHUCK_ZERO)
{
  yValue = CHUCK_ZERO;
}

escOutputValue = map(yValue, CHUCK_ZERO, CHUCK_MAX, ESC_ZERO, SPEED_MAX);

if (cur_escOutputValue < escOutputValue)
{
  cur_escOutputValue++; //slow acceleration
}
else
{
  cur_escOutputValue--;
}
```

mapping joystick values to ESC PWM values

Done Saving.

77 Arduino Uno on /dev/cu.usbmodem411

Figure 12: The Arduino C++ IDE. I used Arduino's built-in mapping function to convert Nunchuck joystick input values to ESC PWM values

Additive Manufacturing

In addition to CNC Milling and Lathe conventional machining, which I used to create and modify aluminum parts for the skateboard's motor assembly, I used 3D printing to create components that had complex features that could not easily be machined in a conventional manner.

I used the ReplicatorG open-source 3D printing program to turn my CAD models into G-Code for the 3D printer. In the program, I could specify the percentage of plastic filled into the part, the thickness of each layer of plastic deposited, and the temperature of the heating element that extruded the plastic onto the 3D printer's bed.

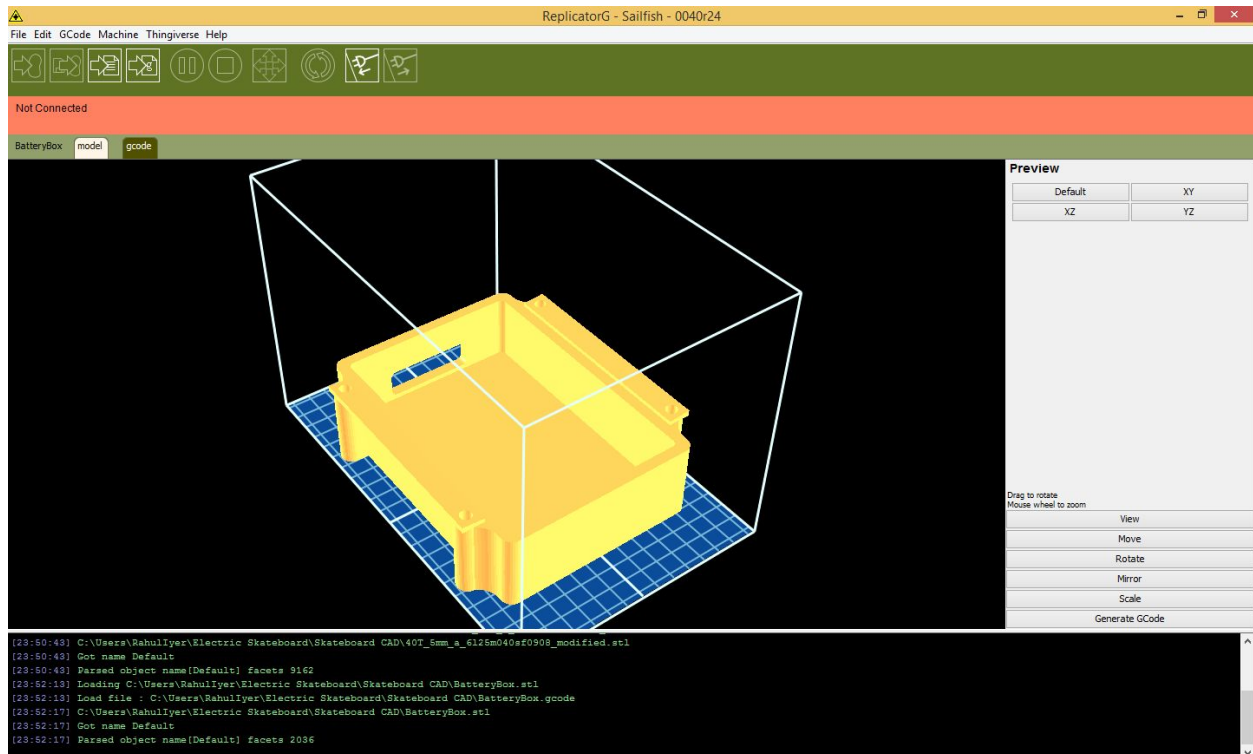


Figure 13: Viewing the battery box CAD model on the print surface in Replicator G. After positioning the model on the print surface, the user can select different print parameters and upload the G-Code to the printer.

The battery box, the enclosure for the skateboard's electronics, the large pulley, and the case for the arming key were all 3D printed.

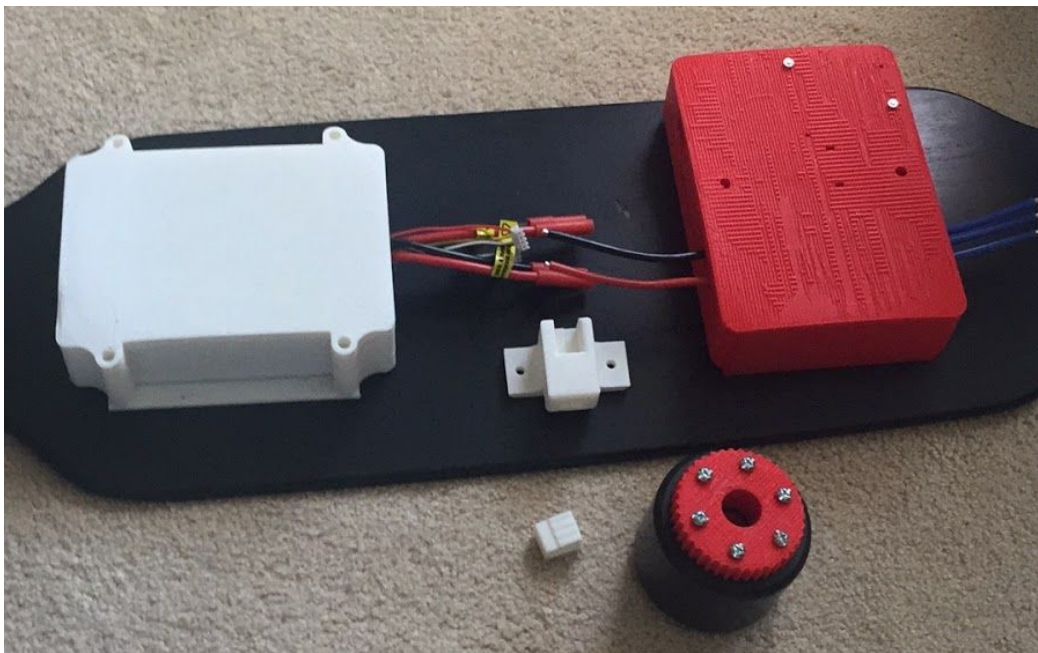
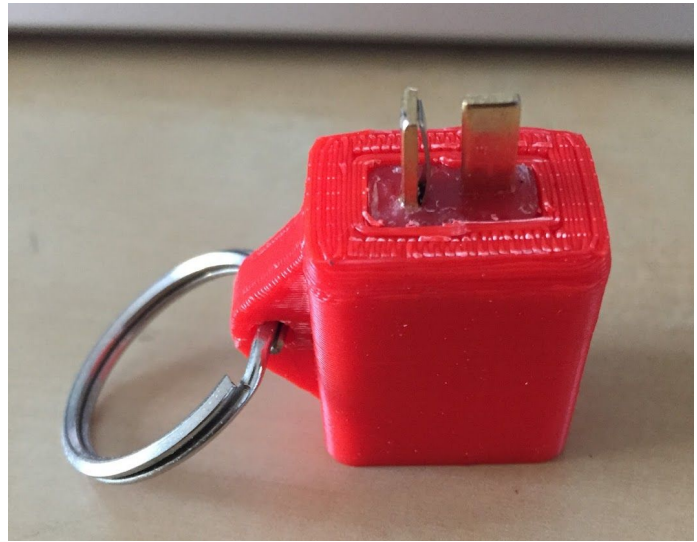


Figure 14: The 3D printed components of the skateboard.

Clockwise from top left: Large pulley and support plate mounted to skateboard wheel, arming key case with Dean's T connector press-inserted, battery box (white) and electronics box (red) with wheel assembly and parts for the arming key assembly

Future Projects and Improvements

After building and testing this electric skateboard, I want to improve both the design and the user experience through the following modifications:

- **Wireless Nunchuck:** Replacing the wired Wii Nunchuck with a wireless model like the Nyko Kama will improve the user experience, as the user will be able to move his or her hands freely while riding. The wired nunchuck introduces a point of failure on the electric skateboard, as tugging on the nunchuck cable too hard may cause it to disconnect.
- **Custom PCB with integrated Arduino:** Having one PCB for all the control electronics will not only make the design look more professional, but will also make the skateboard more reliable, as there will be less assembled parts.
- **Voltage Readout:** Adding a voltage readout to the skateboard will allow the user to know when the batteries will need to be recharged. With a voltage readout, the user can plan rides based on how much charge is left in the onboard batteries.
- **Encoder:** Attaching an encoder to the skateboard will allow me to create a closed-loop control system. Data from the encoder about the board's current velocity and acceleration can be used to improve the user's control of the board, and can even be used to create braking mechanisms to slow down the skateboard.
- **Regenerative Braking:** Regenerative braking will let the skateboard quickly come to a full stop. I hope to implement a regenerative braking circuit to give the user more control over the skateboard while travelling at high speeds.