Animatronic Iron Man Mk III suit

by Honus on April 26, 2014

Table of Contents

Author:Honus
I'm a former bicycle industry designer turned professional jeweler. I like working with my hands and am happiest when I'm in the shop building my creations. If you need help with your project just let me know!

Intro: Animatronic Iron Man Mk III suit

Iron Man costumes have been extremely popular lately and the number one question I am most often asked is "how can I add animatronics to my suit?" My friend Greg wanted to add animatronics to his MkIII fiberglass suit so he asked for my help and for this suit we went all out.

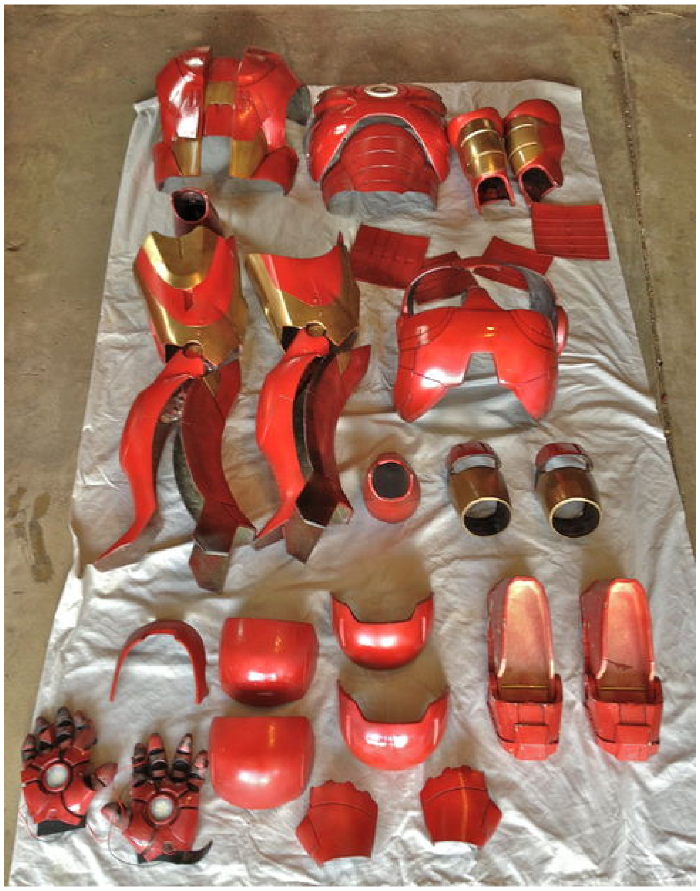
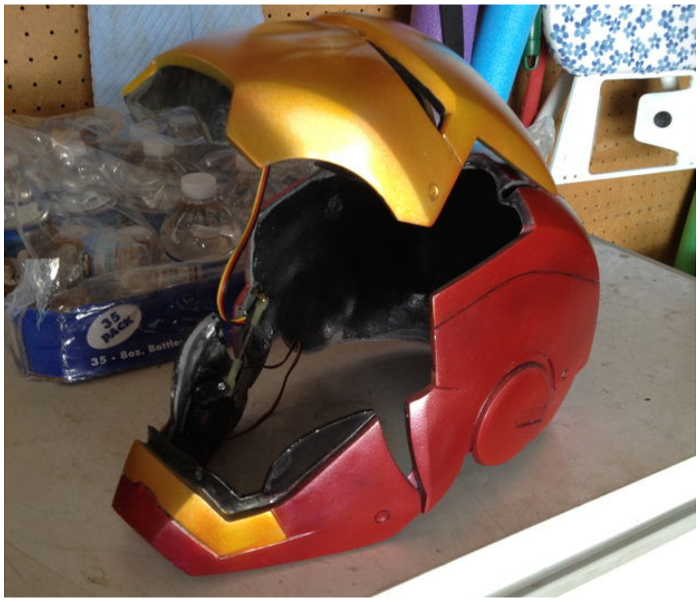
We wanted to add as many functions as seen in the movies as possible, which wasn't easy given that most of those sequences were not done using practical effects. The other issue was how should all of the functions be actuated? After considering several options we used RFID tags in the gloves to trigger the shoulder rocket pods, hip pods, forearm missile, back flaps and helmet. The helmet has wireless control via XBee radios. The boots light up and make sound while walking by using an infrared distance sensor in the boot to trigger the effect.

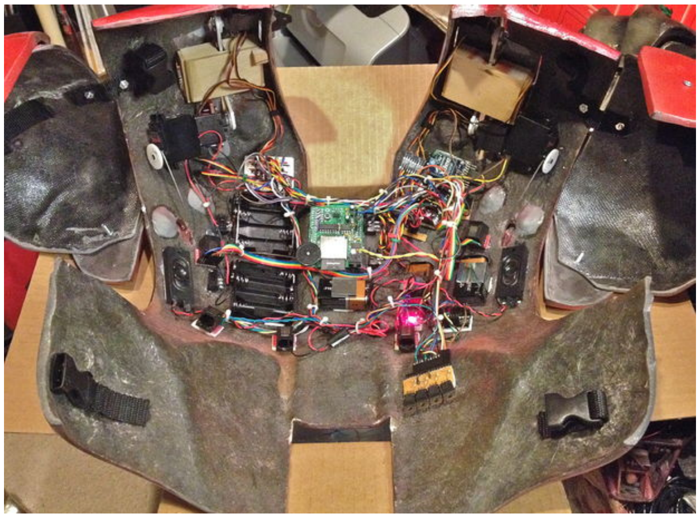
Here's a video that shows all of the suit functions-

This is certainly not an easy project but if you know your way around an Arduino and can wield a soldering iron this instructable will show you how to do it.

Be sure to click on any picture to get a larger version.

Let's get started!



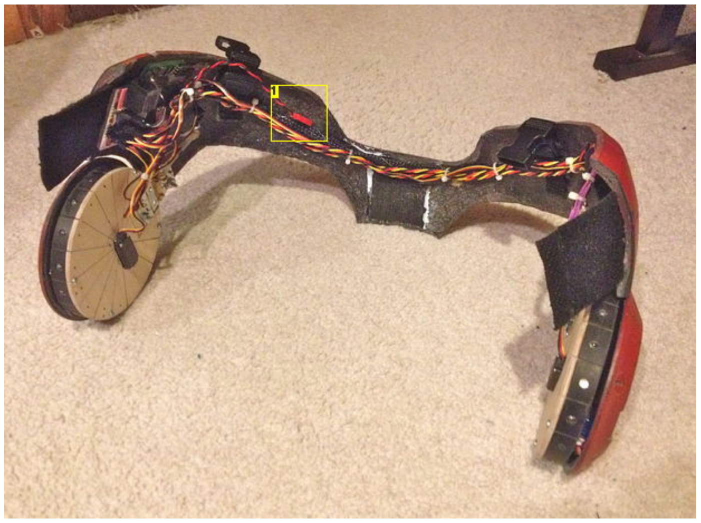


Image Notes
1. JST connector to provide power to the hip pods

Step 1: Build design/details

The suit is basically broken down into three systems: left side, right side and boots.

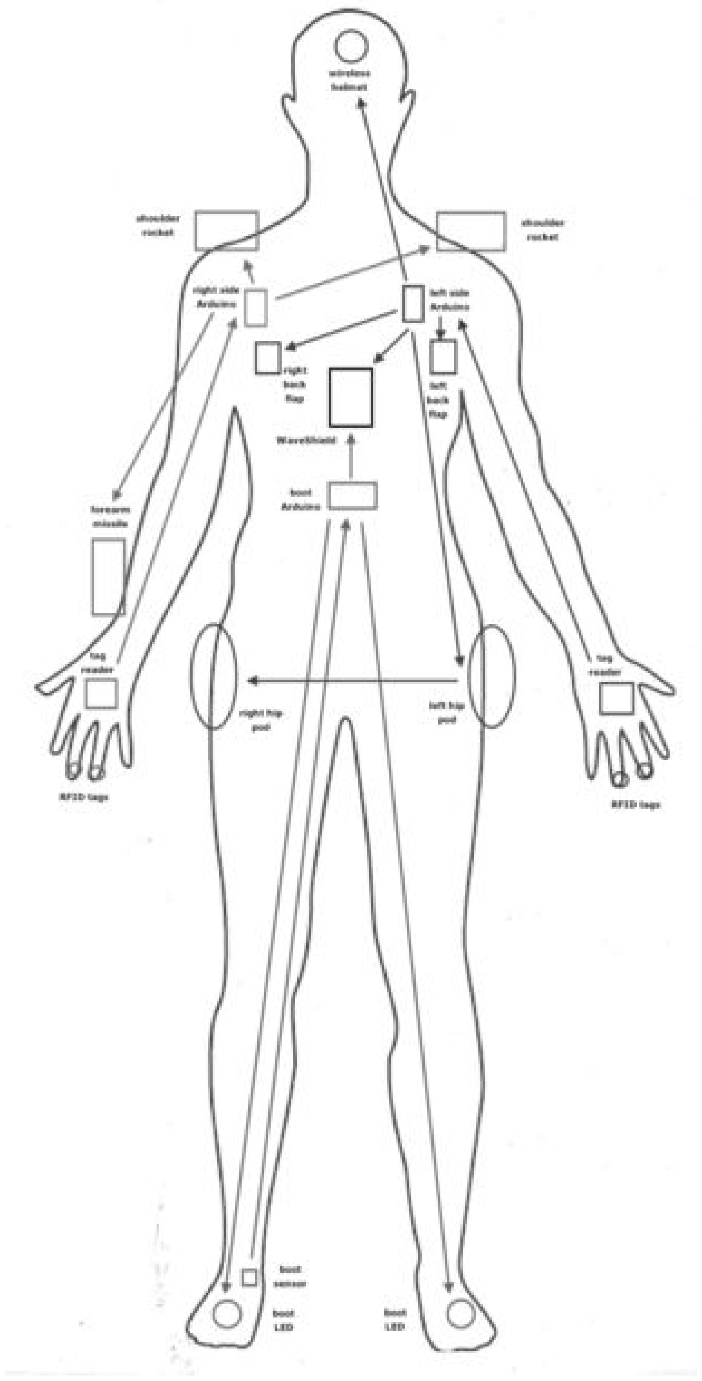
The left hand has two RFID tags that trigger programmed sequences for the helmet, hip pods and back flaps. The right hand RFID tags trigger programmed sequences for the forearm missile and shoulder rockets. The boots have an infrared sensor that triggers the boot lights and sound effect as soon as the boot is lifted from the ground.

The single most difficult thing about this build is that the suit fits like a glove- there's no room in it! The helmet has less than 1/2" of space around the head, there's about 1" depth for the shoulder rocket pods and the hip pod area has less than 1" depth available so the packaging of the mechanics and electronics is really tight. Another issue is that there's almost no flat surfaces so mounting servos and hardware gets really interesting.

The system is Arduino based and uses four ProMinis- one for each side, one for the boots and one in the helmet. Since we wanted the helmet to be easy to take on and off we decided to make it wireless using XBee radios to send the control signals. For the point to point wiring running from the electronics mounted in the back to the arms and feet we used Ethernet cables and jacks so they could be easily disconnected. The sound effects for the boots are handled by a WaveShield that sits on a Arduino Pro.

If you are not familiar with Arduino and XBee radios then please read through this instructable. It will explain a lot of the basics and you'll be up and running in no time!

Another issue with systems like this is the voltage and current requirements so we thought it best to power the servos separately using AA batteries, primarily for ease of availability if the suit is to be worn at conventions.



Step 2: Tools and materials

As far as tools go, most all of the work was done with basic tools -

I used a scroll saw to cut the plywood, soldering iron/torch, wire cutters, pliers, screwdrivers, glue gun and a Dremel tool. A multimeter will also come in very handy, as will a servo programmer . A servo programmer allows you to adjust several performance parameters of digital servos, such as speed, direction of rotation, etc. They can also be used to test a servo's movement, which is super handy when installing servos.

A bit of fiberglass work was done in the helmet and for this I really like epoxy resin- I use West Systems epoxies. Epoxy resin doesn't smell and it's a lot tougher than polyester resin. The West Systems epoxy resin also has an incredibly long shelf life and it's super easy to mix if you use their metered pumps. ProPoxy 20 (a two part epoxy putty) is something that I use a lot when bonding various materials- it's great for bonding metals to plastic. I also use a lot of super glue for tacking parts in place before securing them with epoxy putty. My favorite is Gorilla glue and I often use an accelerator to speed it's drying time. Apoxie Sculpt is used to fill gaps and add sculpted details.

West Sytems Epoxy
ProPoxy 20
Gorilla super glue
Apoxie Sculpt

Make sure to wear a respirator when cutting and sanding fiberglass! I also always wear disposable gloves when working with epoxies and resins.

Most of the build materials were nothing special. For pivots we used brass rod and music wire. The exact size isn't particularly important- I used whatever I had in my scrap bin. Brass and Aluminum sheet was also used to make parts for servo linkages. As long as it's it's not super thin material the exact size isn't really important. The

Aluminum sheet used for the shoulder rocket servo arms is around 1/8" thick.

As far as electronics are concerned it's quite the laundry list-

2 × ID-12 RFID tag reader Sparkfun part # SEN-11827 (LA version is replacement)

2 × XBee Series 1 module Sparkfun part # WRL-11215

2 × Adafruit XBee Adapter board Adafruit Product ID: 126

2 × Arduino ProMini 328 5V Sparkfun part # DEV-11113

1 × Arduino ProMini 328 3.3V Sparkfun part # DEV-11114

1 × Arduino Pro 328 5V Sparkfun part # DEV-10915

1 × Adafruit Wave Shield Adafruit Product ID: 94

1 × Pololu 5V DC/DC converter D24V5F5Steps down the helmet 7.4V battery voltage for the ProMini

5 × Hitec HS-5055MG sub-micro servo For the forearm gauntlet /hip pods- Servocity.com

8 × Hitec HS-85MG micro servo For the shoulder rocket pods/ hip pods- Servocity.com

3 × Hitec HS-82MG servo micro For the forearm gauntlet pod/shoulder rocket pods- Servocity.com

2 × Hitec HS-485HB standard servo For the back flaps- Servocity.com

1 × Hitec HS-5087MH micro servo For the helmet chin section- Servocity.com

1 × Hitec HS-7245MH mini servo For the helmet faceplate- Servocity.com

4 × 16mm RFID button tag (125 kHz) Sparkfun part # SEN-09417

2 × Luxeon Rebel high power white LED For the lights in the boots Sparkfun part # BOB-09656

1 × Sharp GP2D120XJ00F Infrared Proximity Sensor Sensor for the bottom of the boot Sparkfun part # SEN-08959

8 × RJ45 8-pin connector Ethernet cable connector Sparkfun part # PRT-00643

8 × Breakout Board for RJ45 Sparkfun part # BOB-00716

1 × 7027 "BuckToot" LED Driver Module to power the Luxeon LEDs Sparkfun part # COM-09642

3 × JST RCY Connector For the helmet battery pack Sparkfun part # PRT-10501

6 × AAA NiMH cells 1.2V For the helmet battery pack - local grocery store

2 × RFID reader breakout Sparkfun part # Product SEN-08423

1 × SD/MicroSD Memory Card (4 GB SDHC) Adafruit Product ID: 102

4 × Break Away Headers - Straight For making connectors/servo leads Sparkfun part # PRT-00116

4 × Female Headers For making connectors/servo leads Sparkfun part # PRT-00115

2 × Adafruit Perma-Proto Quarter-sized Breadboard for mounting connectors, etc. Adafruit Product ID: 589

Image Notes
1. My trusty soldering iron

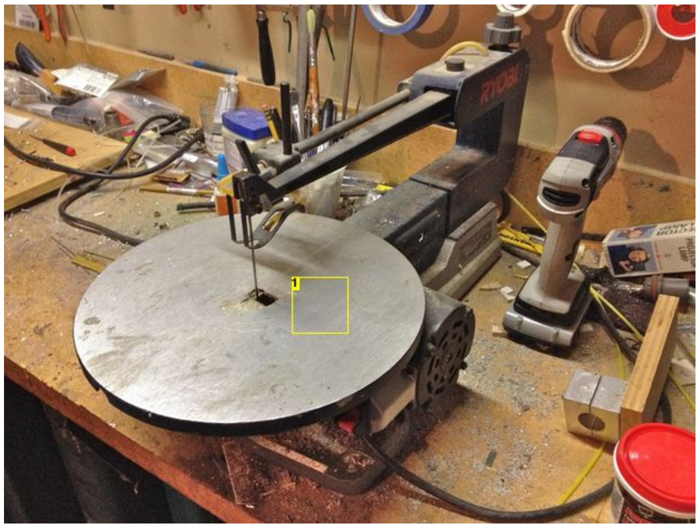


Image Notes
1. Scroll saw- one of the most used tools in my shop

HiTEC
HFP-25
Hitec Digital Servo Programmer & Servo Tester for ALL Servo Brands
DISPLAY
INPUT
UP/L
M
DN/R

Image Notes
1. Hitec servo programmer/tester- a very useful tool when working with servos

Step 3: Build tips

Notes about mechanics and suit construction:

Greg's suit is molded fiberglass. It is unlikely that a foam suit would be rigid enough to support the animatronic systems without some sort of reinforcement, especially in the shoulders as a large area has to be cut away.

It would also be best to have a finished assembled (but not painted) wearable suit before adding any animatronic system. It is important to know exactly how much room you have to work with and how it fits the suit performer before adding animatronics.

There are lots of different ways the mechanics of this suit could have been constructed. We tried to use readily available materials found in hobby shops and hardware stores whenever possible to save time and money. Items such as the shoulder rocket pods and hip pod hinge assemblies could easily be 3d printed if so desired.

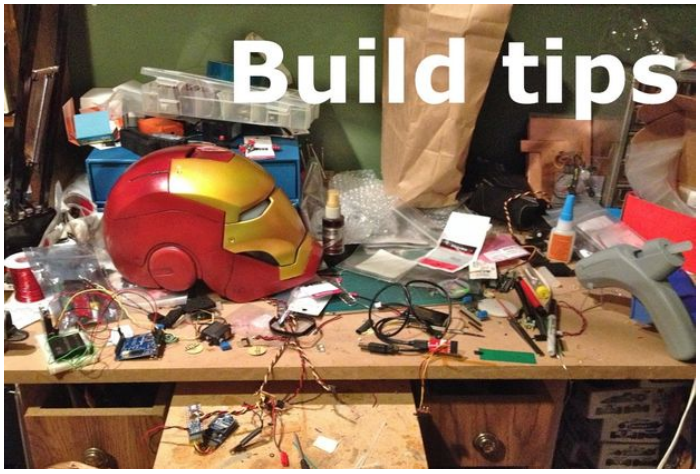
A few notes and tips concerning the use of servos in the build:

The servos listed are just what we used based on speed, power, durability and cost requirements. I would definitely recommend using metal gear servos with ball bearings. Nylon geared servos strip very easily, especially the micro and sub-micro variety.

Several of the servos will need to be changed to reverse rotation since the right and left side of various suit parts receive the same control signal. There are three options- use digital servos (easiest but most expensive), have the supplier change the rotation (Servocity.com does this) or change the rotation yourself by taking the servo apart and reversing the motor lead wires and the outer two pot wires. I changed them on the micro servos for the shoulder rockets and hip pods myself since I'm cheap. :)

We did elect to use high voltage digital servos in the helmet for the power they provide and the fact that their speed can be adjusted using a servo programmer. The servos for the hip pod sliding panels and gauntlet side panels also used digital servos since they needed to be powerful and swapping the wires on sub-micro servos can be really tricky. The other reason was because the standard sub-micro servos use nylon gears and we wanted to use metal gears for durability.

The servo position values in the code will probably need to be altered for another build since no two Iron Man suits are alike. The key when doing this is to make small changes in the values while paying attention to make sure the servos don't ever stall, as they can be easily damaged. If a servo does stall immediately turn off the power to the servos and make adjustments to the code.



Step 4: Wireless helmet

The first item on our list was making the helmet open.

Figuring out how to make the helmet work was pretty tricky. We definitely wanted it to be wireless so it could be easily taken off but there's barely any room in the helmet for servos, let alone electronics. The first system that was implemented used two identical high voltage digital servos with a rod system that moved the faceplate and chin at the same time. As the servo pulled the rod the arm raised the faceplate and a second pivoting rod pushed the chin section open. The top of the faceplate had two small pivoting links that ride in tracks in the top of the helmet. While this system worked really well it was eventually scrapped as it took up too much space around the sides of the helmet.

The next system worked a bit different. The tracks in the top of the helmet were kept but the links had a tendency to bind as they pivoted so they were reconstructed and epoxied to the top of the helmet using ProPoxy 20 epoxy putty in a fixed position.

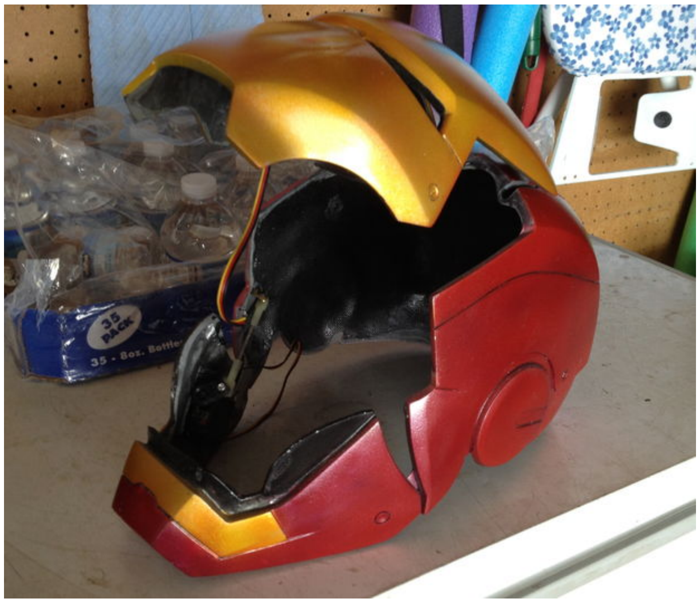
The servo mechanism was changed so one servo would open the faceplate while another would open the chin- that way the timing could be changed so the faceplate would open first and then the chin would open. When closing the helmet the chin would close first, then the faceplate. The faceplate servo has a brass arm that is silver soldered to a hinge that is fiberglassed into the top center section of the helmet. The brass plate that is soldered to the hinge tube is extended back a ways in order to support the helmet section as it would otherwise be too flexy to work properly. Both of the servos were attached to the helmet using high strength Velcro.

The chin servo was swapped out for a slightly smaller servo and it pushes a 4-40 threaded rod that opens the chin section- it's a pretty simple arrangement. Eventually the chin servo was relocated closer to the center of the chin piece to make more room for one of the battery packs.

On the electronics side, a split battery pack was constructed using six AAA NiMH cells for a total of 7.4V to power the high voltage digital servos. Originally we had planned on using LiPo packs but felt that NiMH cells would be a lot safer and less hassle in terms of battery management. The two battery packs were wired up to an Arduino ProMini (3.3V version) along with a XBee radio. Power to the ProMini was stepped down to 5V using a Pololu DC/DC converter. The electronics were covered with heat shrink for protection.

The two battery packs were wired up to an Arduino ProMini (3.3V version) along with a XBee radio. Power to the ProMini was stepped down to 5V using a Pololu DC/DC converter. The electronics were covered with heat shrink for protection.

Finally all of the electronics were secured in the chin section of the helmet. Everything just barely fit in there without being able to be seen when the helmet is open. Overall we're very pleased with how the helmet turned out. It opens very quickly and we can change the speed of the servos along with the timing of the opening/closing sequence very easily.



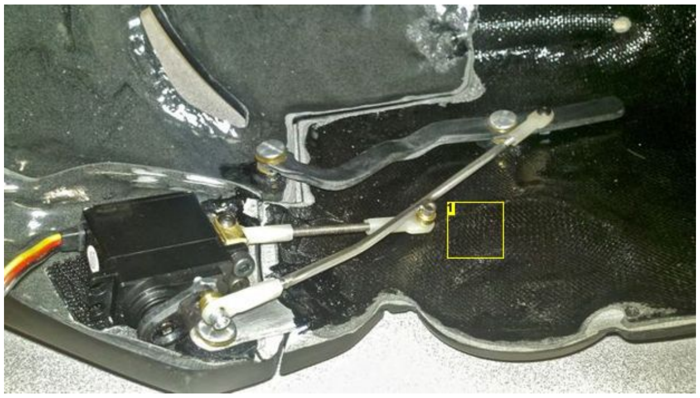


Image Notes
1. Earlier helmet linkage system that was abandoned as it took up too much room on the side of the helmet

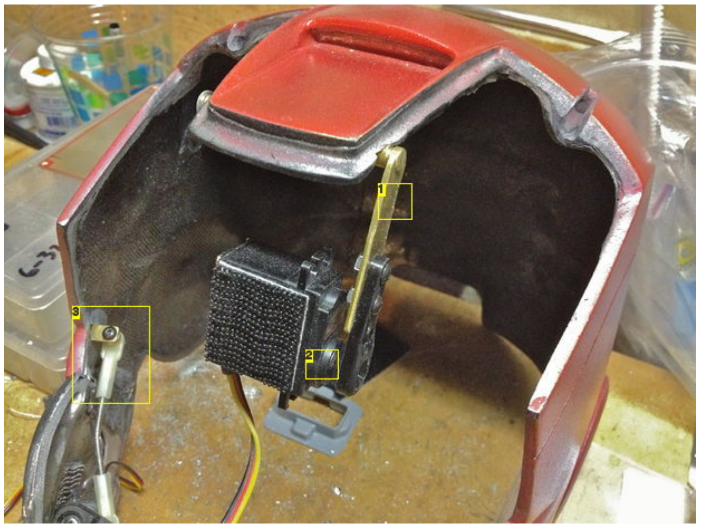


Image Notes
1. Brass pivot arm
2. Faceplate servo
3. Swivel link mount for chin servo made from brass sheet

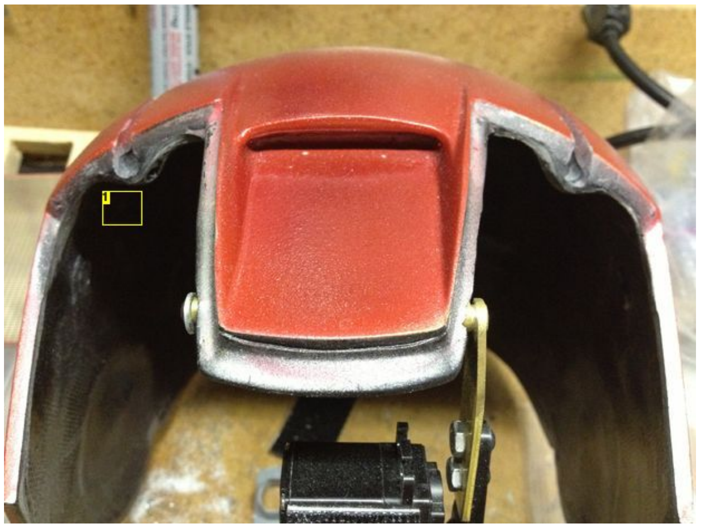


Image Notes
1. Slots for faceplate top links

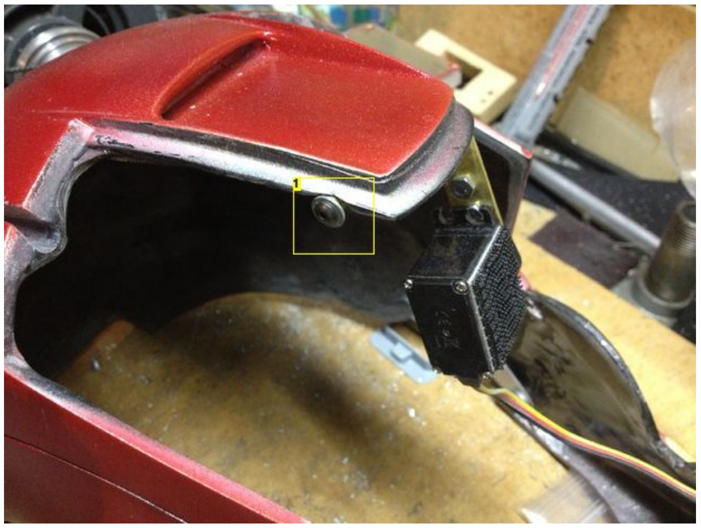


Image Notes
1. Screw threads into brass pivot rod to hold the rod in place. You could also cut a groove in the pivot rod and use a circlip to hold it.

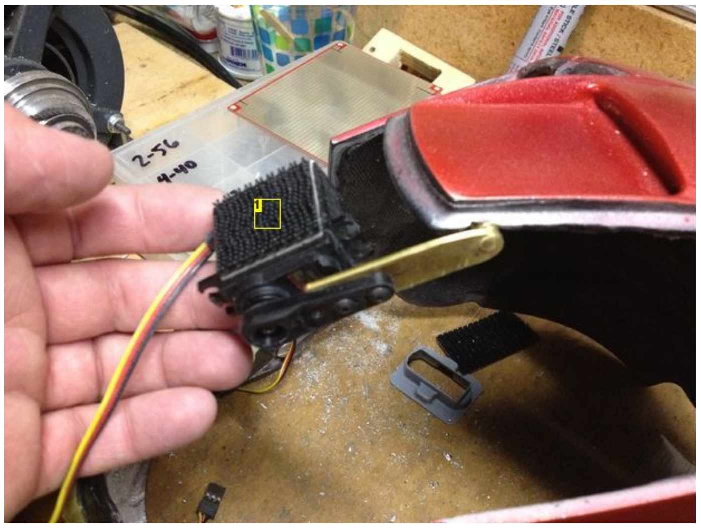


Image Notes
1. High strength Velcro secures the faceplate servo

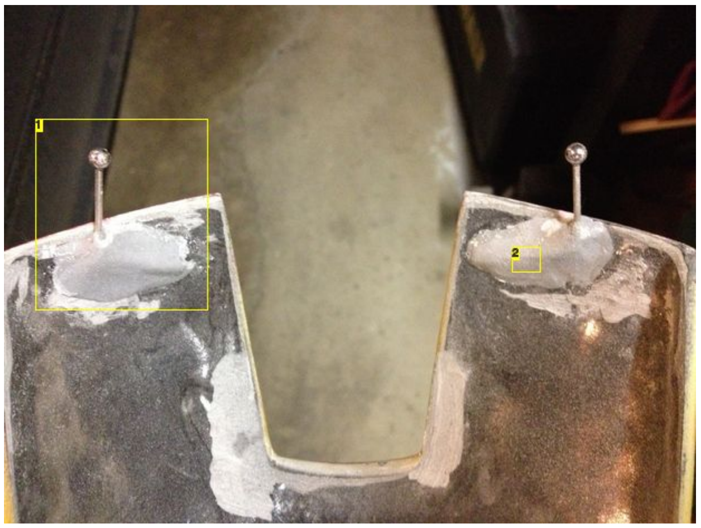


Image Notes
1. Faceplate top links were made by silver soldering a bearing to a stainless

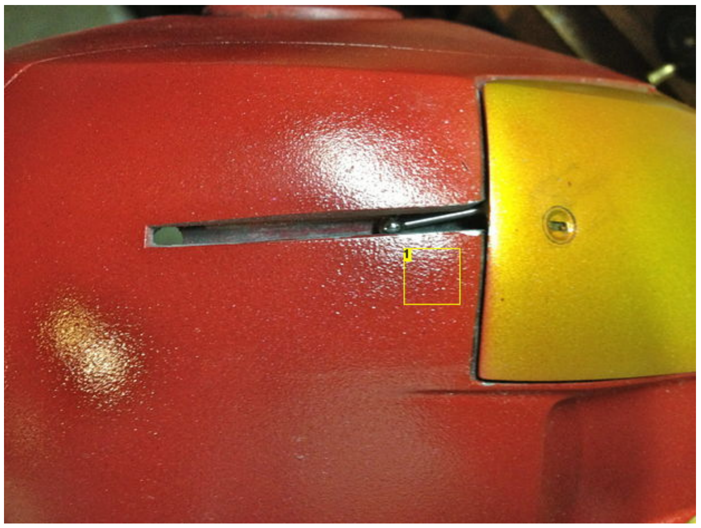


Image Notes
1. The faceplate top links slide in slots in the top of the helmet

steel rod
2. The link rods are bent in an "L" shape and are held in place with epoxy putty

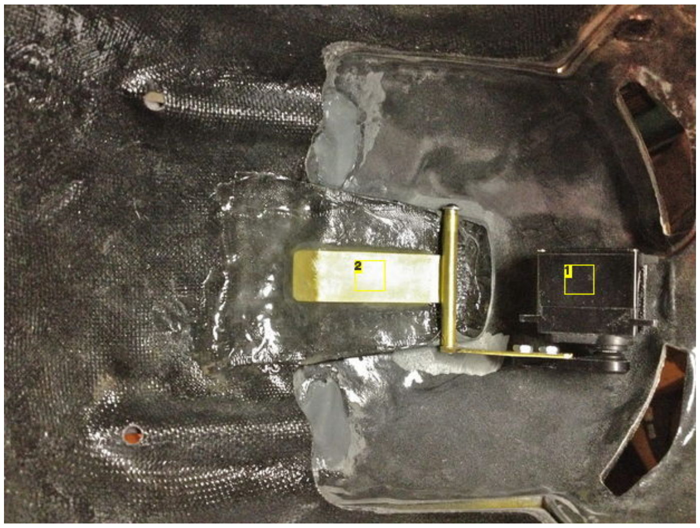


Image Notes
1. Faceplate servo
2. Faceplate pivot is silver soldered to a brass plate. The brass plate is fiberglassed into the helmet center section.

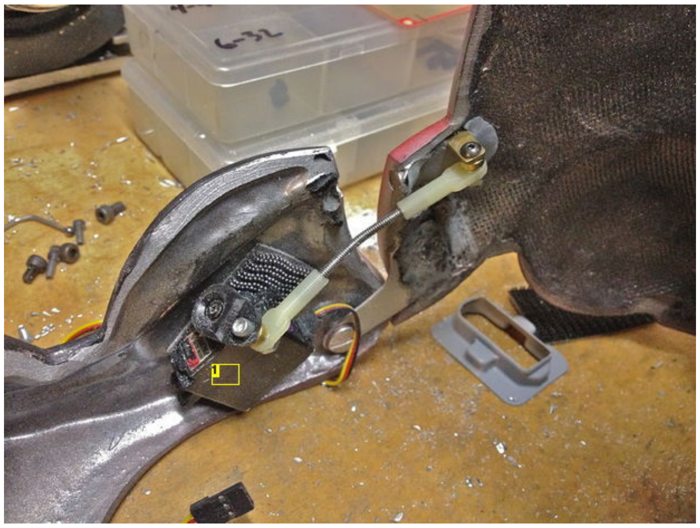


Image Notes
1. Chin servo. This was later relocated to the forward chin section to make room for the batteries.

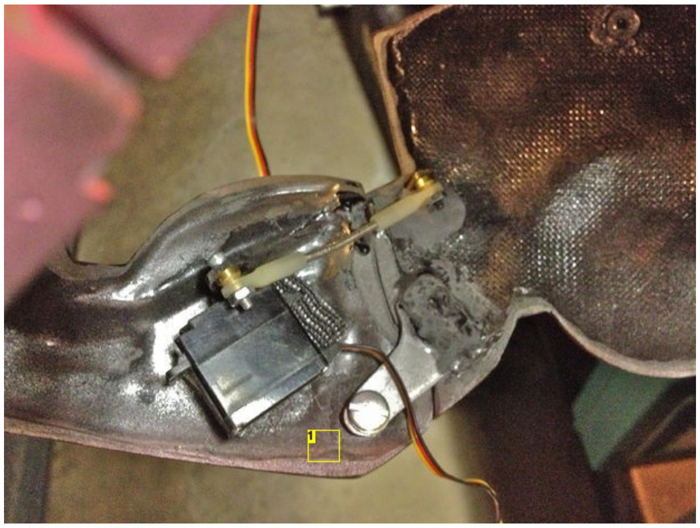


Image Notes
1. Hinge for the chin section

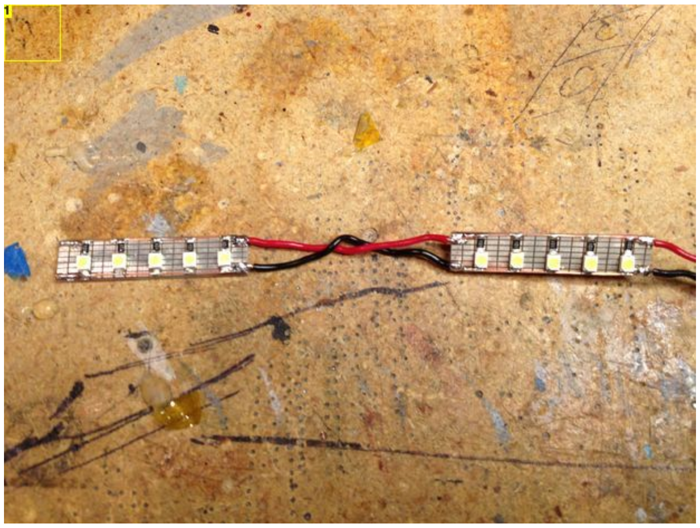
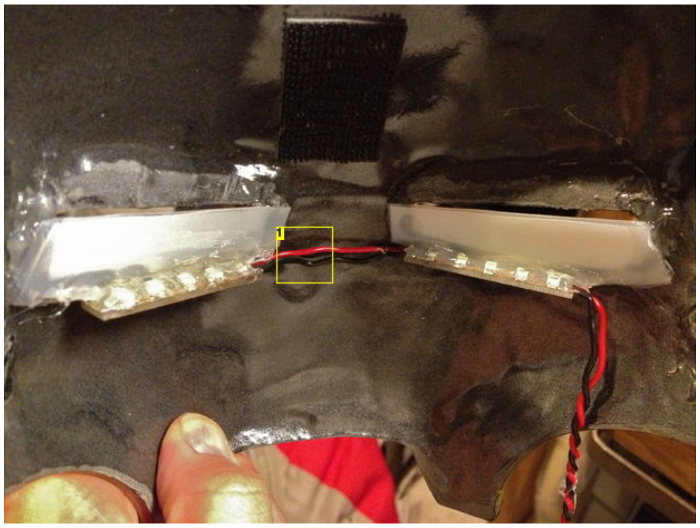


Image Notes
1. Surface mount LEDs wired in parallel were used to light up the eyes. Standard LEDs could also be used.



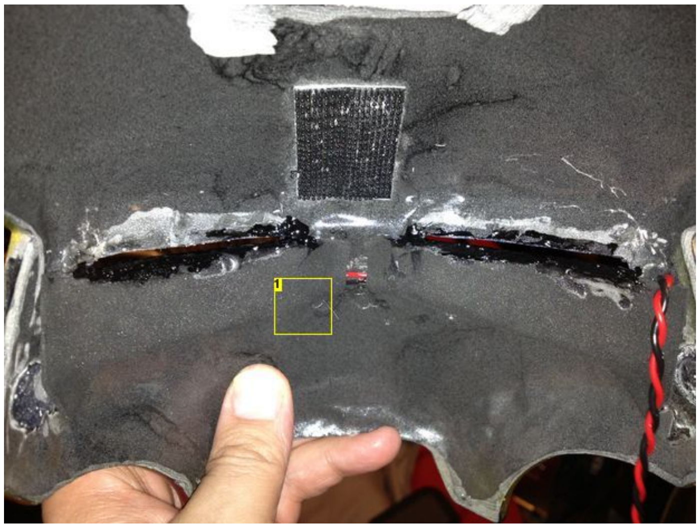
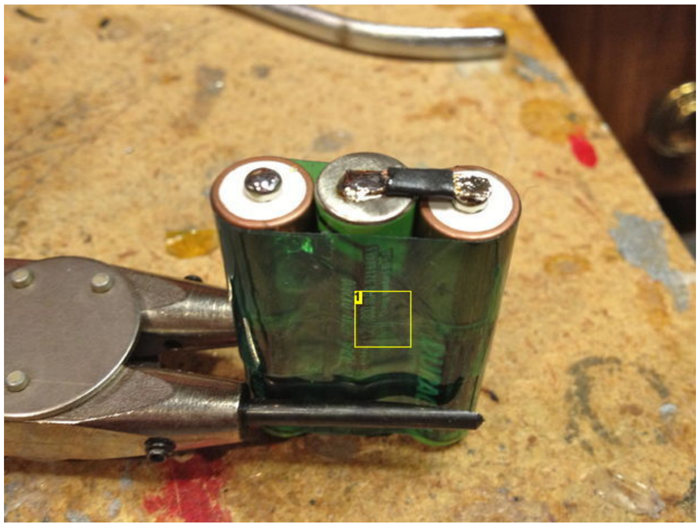


Image Notes
1. The LEDs were hot glued to the eye lenses. The eye lenses are made from milk jug material and a slot is cut at the top so the wearer can see out.

Image Notes
1. Black sheet foam was attached over the eyes to block out the light and the top edges of the lenses were painted black



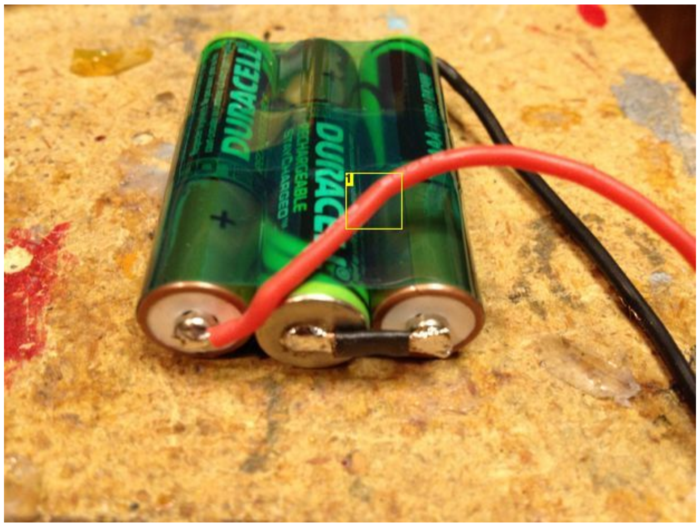
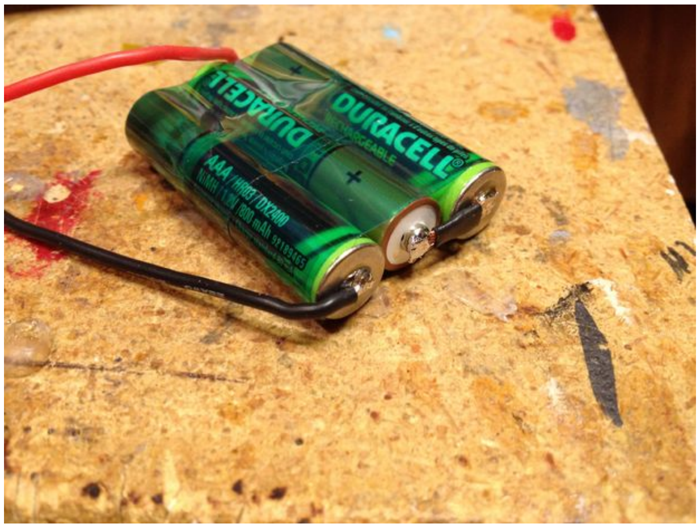


Image Notes
1. The helmet battery packs are made from two sets of three AAA NiMH cells. Copper braid was used to solder the cells together. Heat shrink was placed over the exposed section of copper braid prior to soldering.

Image Notes
1. Power leads were soldered to the battery packs



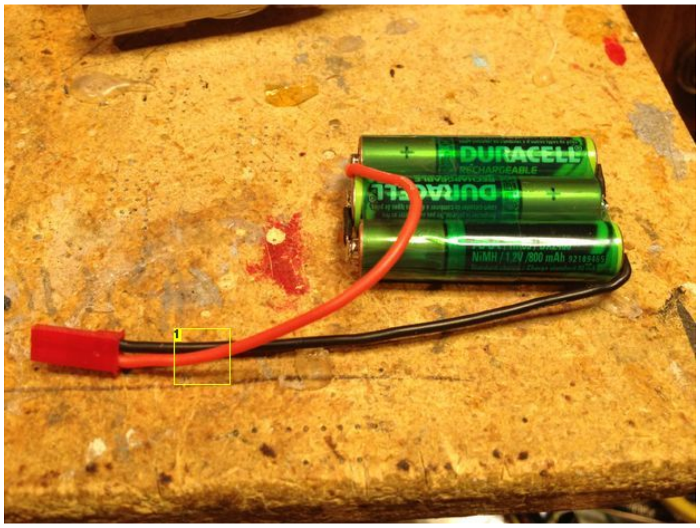


Image Notes
1. JST connectors were used to connect the battery packs in series to give a

total of 7.4V

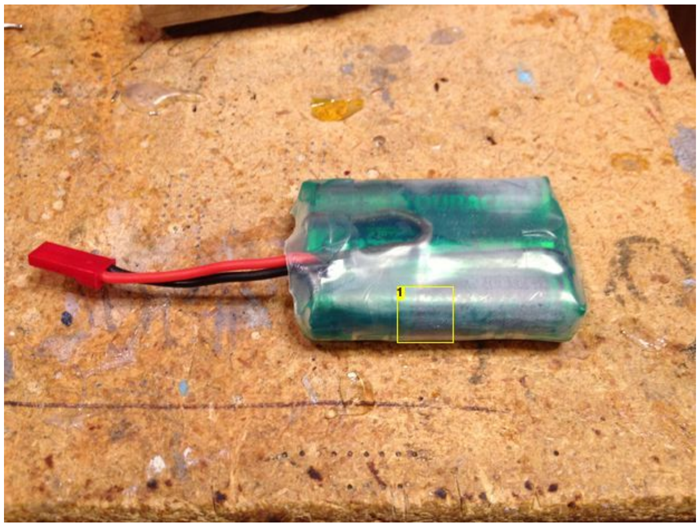


Image Notes
1. The finished battery packs were covered with heatshrink for protection

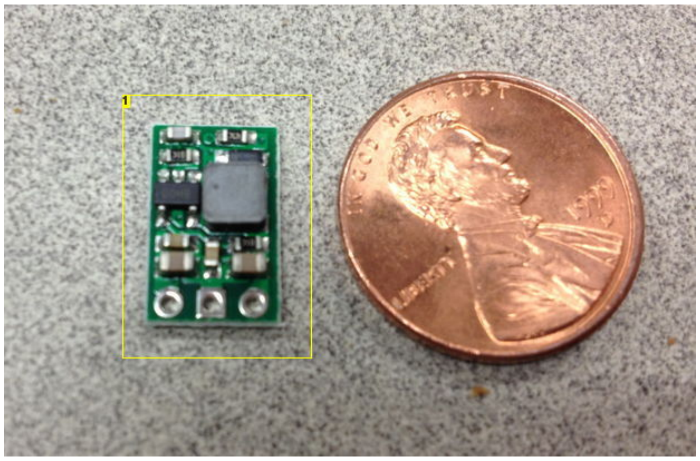


Image Notes
1. Pololu 5V DC/DC converter. These are a lot more efficient than a 7805 voltage regulator. This was used to drop the pack voltage to 5V for the Arduino, XBee radio and eye LEDs.

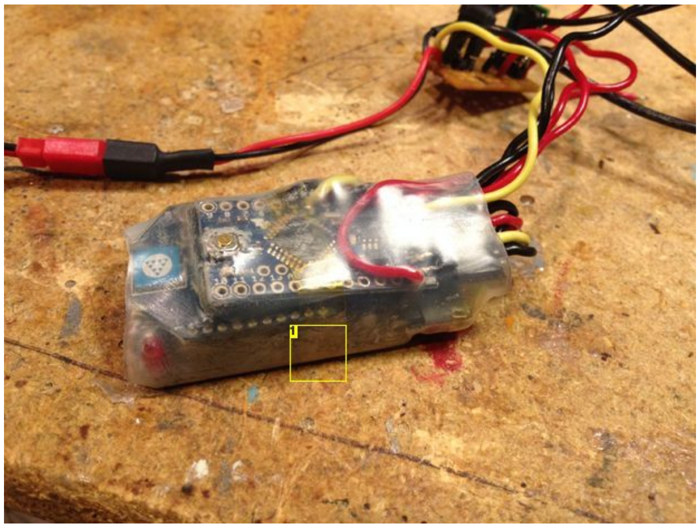


Image Notes
1. The Arduino ProMini and XBee radio were covered with heatshrink material after all of the connections were soldered

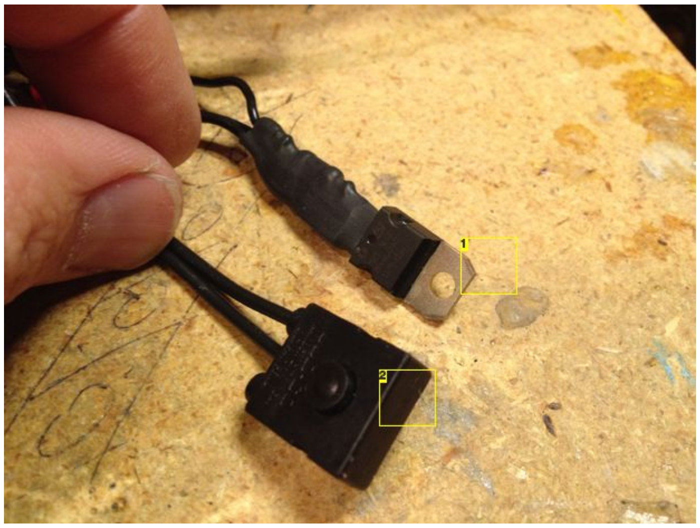


Image Notes
1. TIP120 transistor was used to turn the LED eyes on and off
2. Sealed pushbutton power switch for the helmet

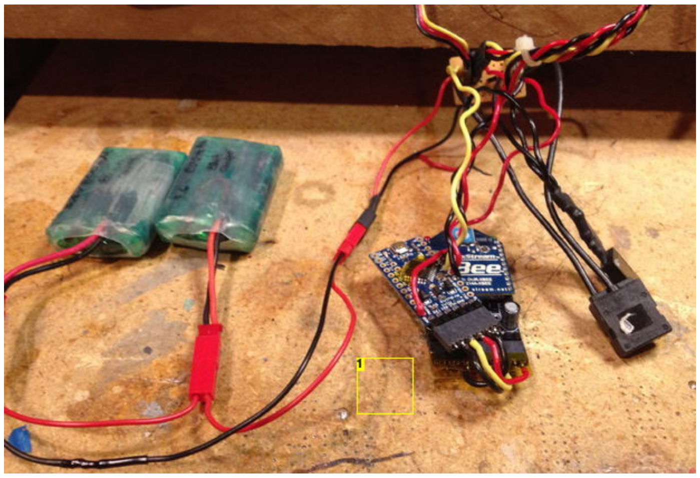
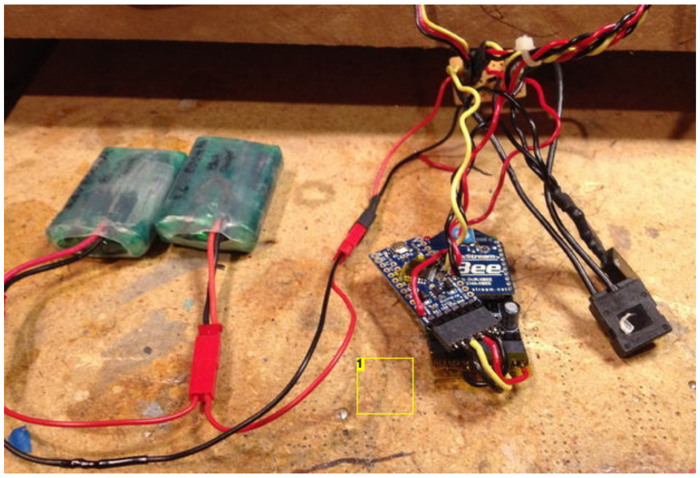


Image Notes
1. Helmet electronics wired up



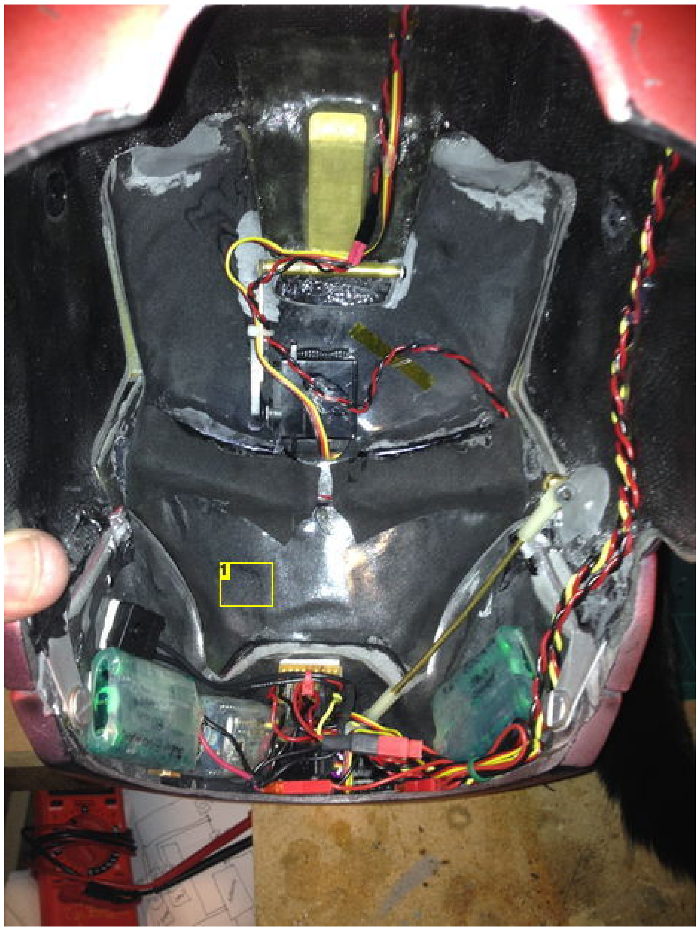


Image Notes
1. Helmet electronics in place. No room to spare!

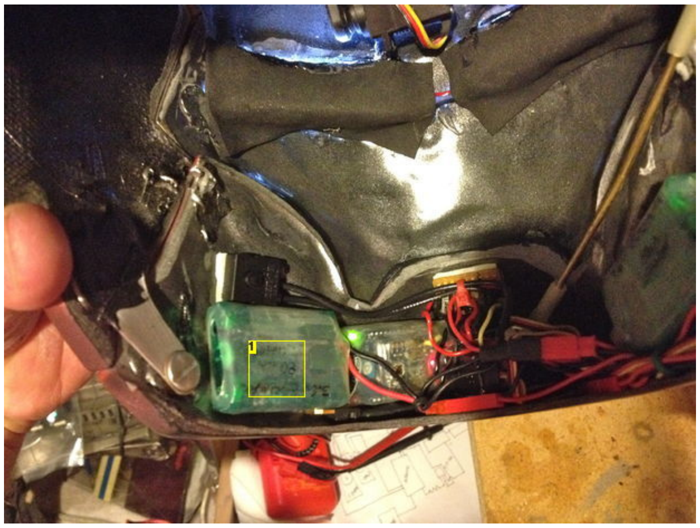


Image Notes
1. Left inside of the helmet showing placement of battery pack, power switch and Arduino

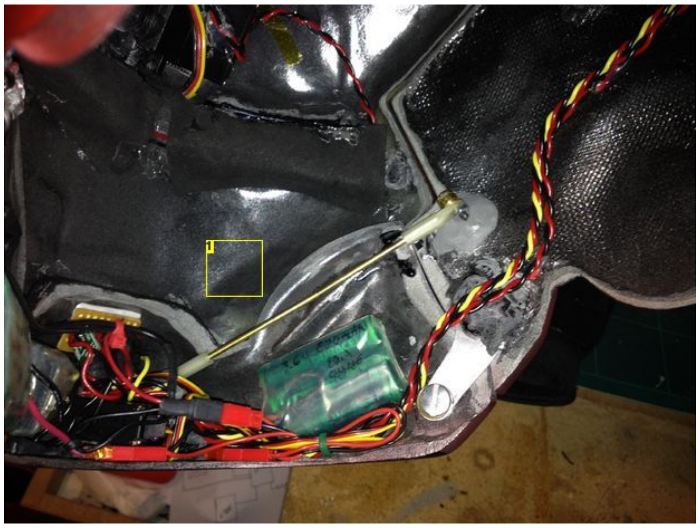


Image Notes
1. Right inside of the helmet showing battery pack placement and chin servo rod to open chin section

Step 5: Forearm gauntlet missile

Other than the helmet, one of the features we viewed as a "must have" item for the suit was the forearm missile.

Much like the helmet, the forearm missile gauntlet has some really tight packaging. There was also the issue that in the film when the gauntlet opens it's done as a visual effect (as are most all of the suit functions.)

We had seen other missile compartments that people had done but they usually only had the top open- we wanted the sides to fold down as well. The other thing we wanted was for the missile pod to open up and back, just like in the movie. And we wanted the missile to extend outward- in for a penny, in for a pound!

To make the opening compartment the fiberglass gauntlet was first cut apart into the appropriate sections and a plastic base plate was epoxied into the opening. Each of the cut side panels had a 4-40 screw epoxied onto the inside forward point using Propoxy 20. Onto this screw was fitted a swivel link with a rod that was epoxied into the forward section of the gauntlet. This formed a very simple single pivot hinge for the front of each of the side panels.

To make the side panels fold down two sub-micro servos had their output arms fitted with a small section of bent music wire that was epoxied to the back of each of the side panels. The servos are attached to the base plate using high strength Velcro. As the servo arms rotate the side panels drop down. A very simple and reliable system.

The missile housing was constructed using thin birch plywood sheet. To make the housing move up and back a small parallelogram linkage was fabricated and attached to the micro servo that raises the housing. The servo is attached to the gauntlet base plate using high strength Velcro. The forward hinge uses a 4-40 ball link attached to the plywood housing and the rear link uses music wire that pivots in brass tubes epoxied to both the servo and the plywood housing. The face of the missile housing was built up using Apoxie Sculpt.

The missile is moved forward in the housing by a sub-micro servo with a long arm. The servo is mounted to the underside of the pod and the arm extends through the bottom of the pod. A short length of music wire is attached to the servo arm and goes through the back center of the missile so as the servo arm rotates the missile extends and retracts through the pod opening. The missile is a short length of wood dowel sanded to shape.

The AA battery holders are held underneath the base plate along with a power switch and an Ethernet connector. The connector is mounted to a small PCB with connectors for the servos and it also has extended wires that reach the gloves. A Ethernet cable runs from the Arduino in the back of suit, down the arm and then connects to the gauntlet. This cable carries the signals for the servos as well as signals and power to the RFID reader in the glove.

That's a whole lot of hardware in a very small space. Believe it or not his arm fits in there no problem- the opening is a lot bigger than it looks in the photo. The gauntlet worked perfect right out of the gate, which was awesome.

Image Notes
1. The gauntlet panels were first cut apart

Image Notes
1. Side panels were held in place with tape while the epoxy cured for the hinge mechanisms

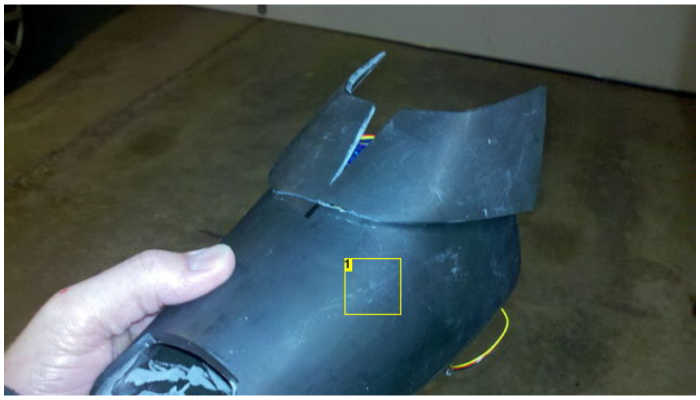


Image Notes
1. Side panels opening during testing

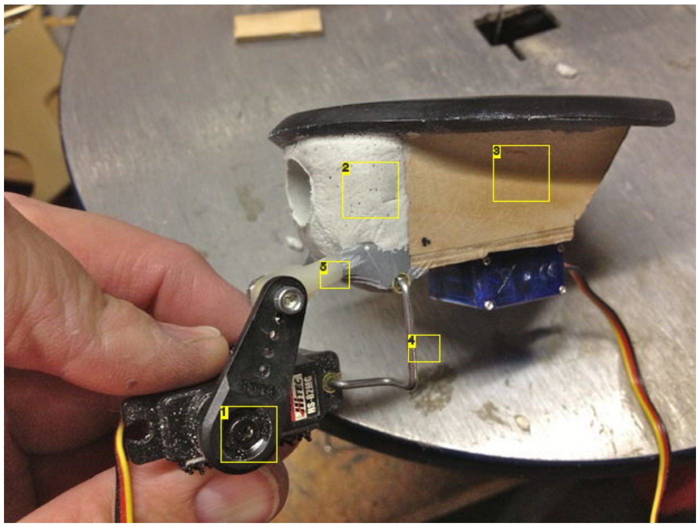


Image Notes
1. Missile pod servo. The rear pivots are brass tubes held in place with epoxy putty.
2. Apoxie Sculpt was used to fill in the front pod area
3. Missile pod was constructed using plywood
4. Music wire rear arm
5. Forward pod arm is made using a 4-40 ball link and threaded rod epoxied into place

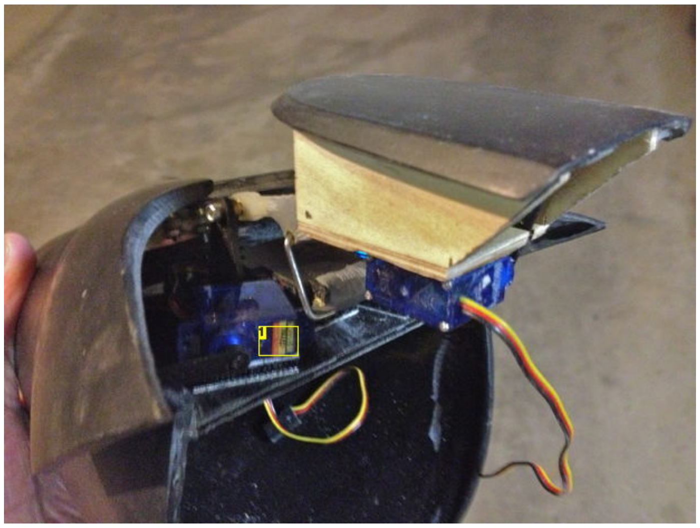


Image Notes
1. Servo to move gauntlet side panel

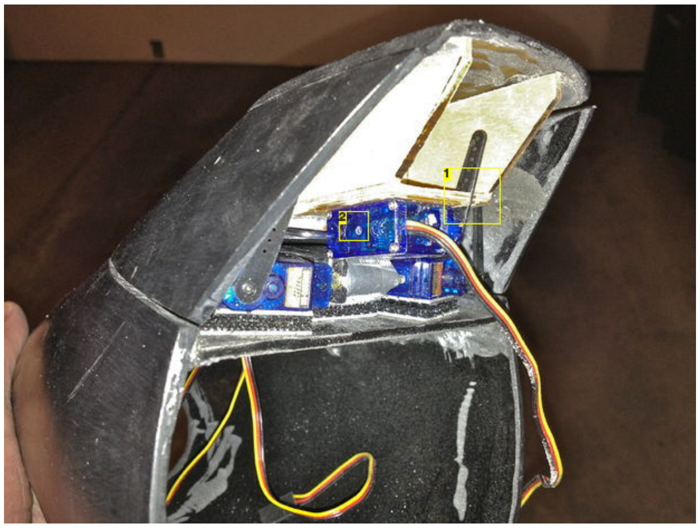


Image Notes
1. Long servo arm reaches through slot in underside of missile pod
2. Servo to move pod missile forward/back

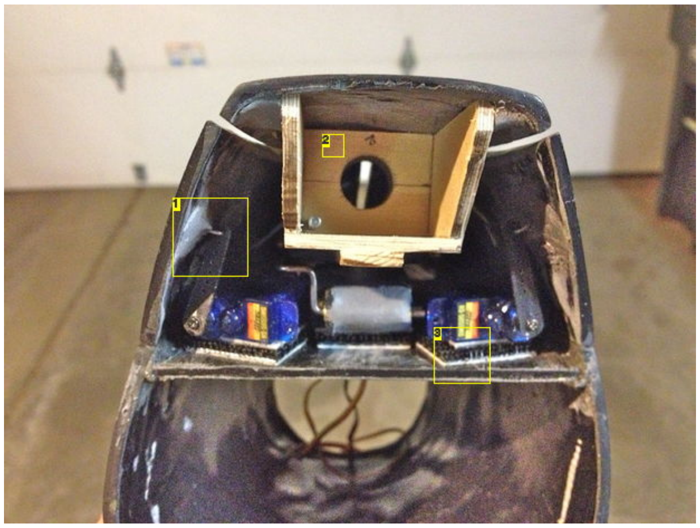
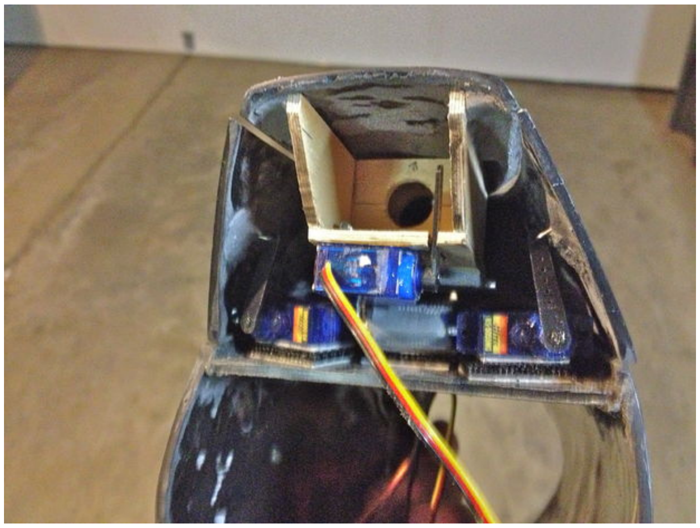


Image Notes
1. "U" shaped piece of music wire attaches servo arms to side panels
2. Hole for pod missile
3. Servos are held in place with heavy duty Velcro



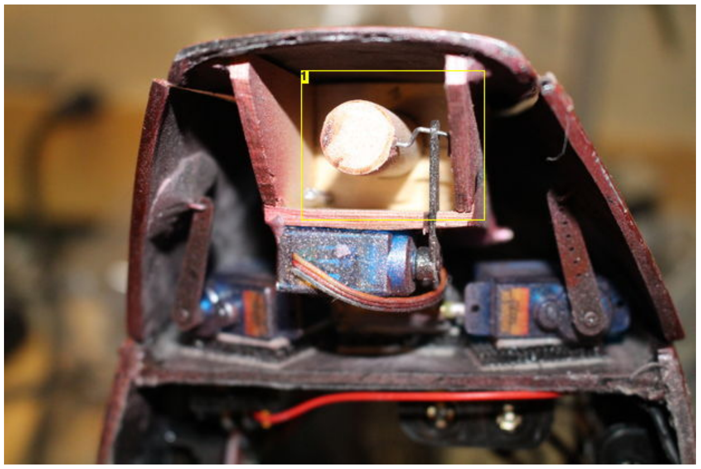
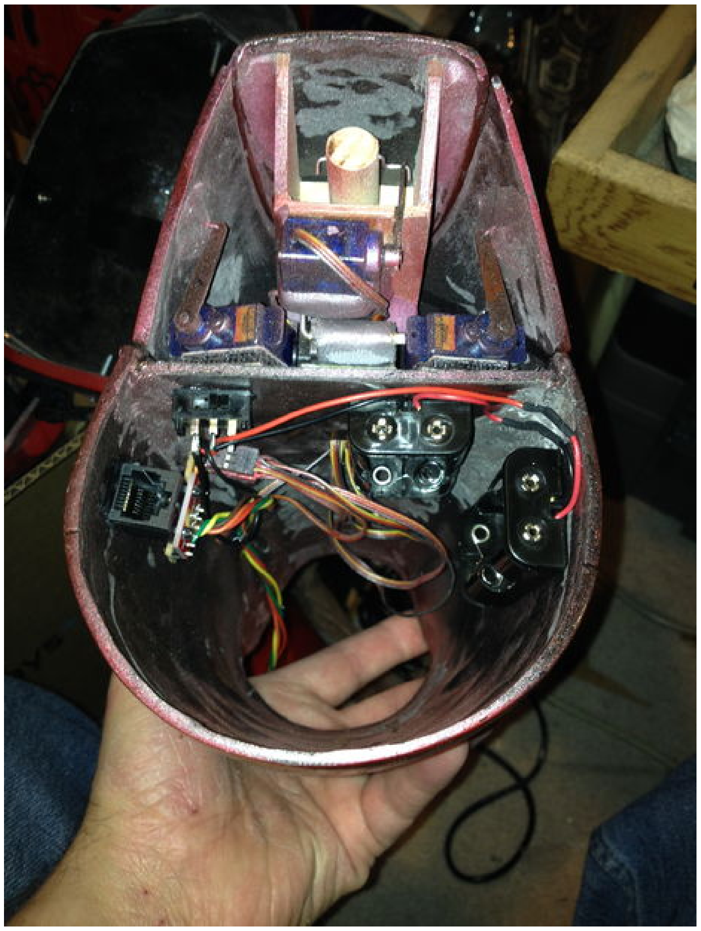
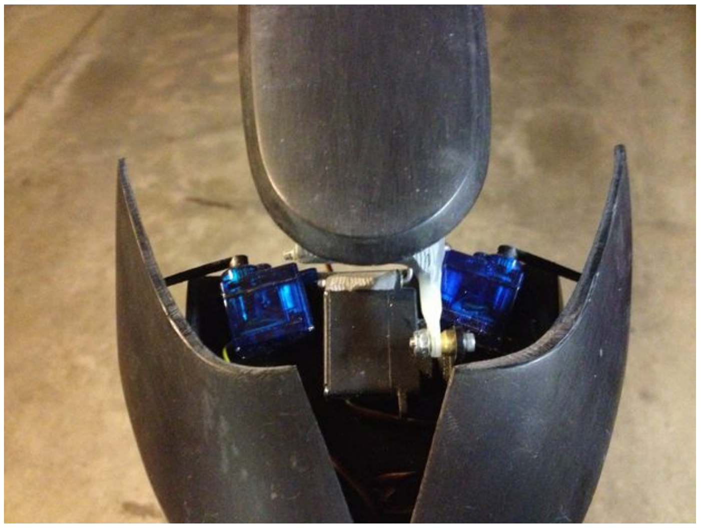


Image Notes
1. Servo arm is attached to missile with a link made from music wire





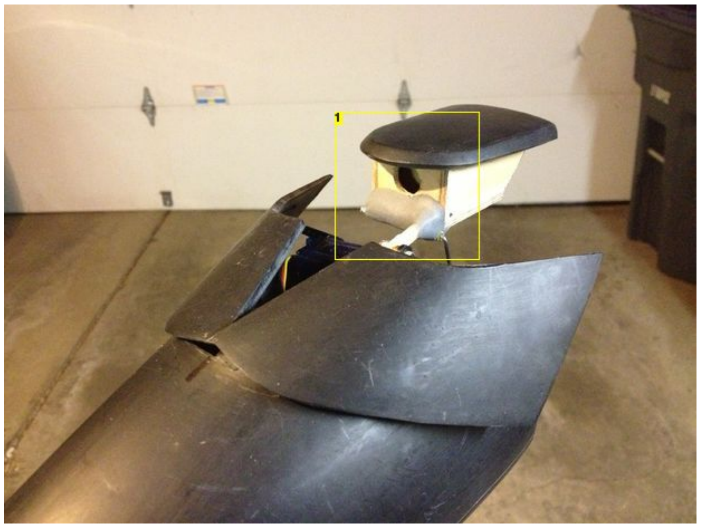


Image Notes
1. Pod before Apoxie Sculpt was used to shape the front

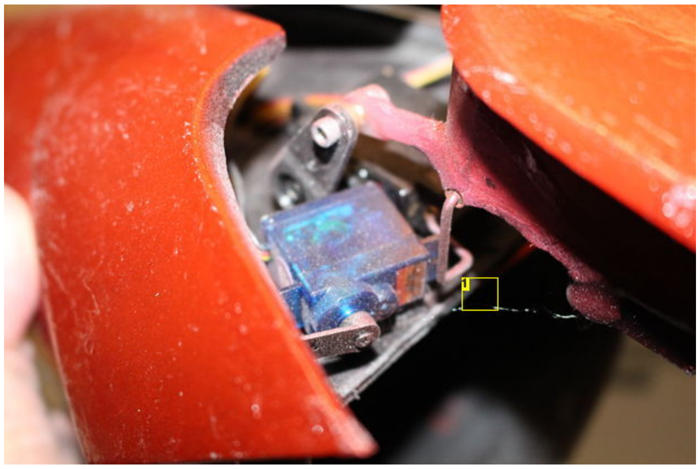


Image Notes
1. Here you can see how the rear arm needed to be bent in order to clear the side panel servo

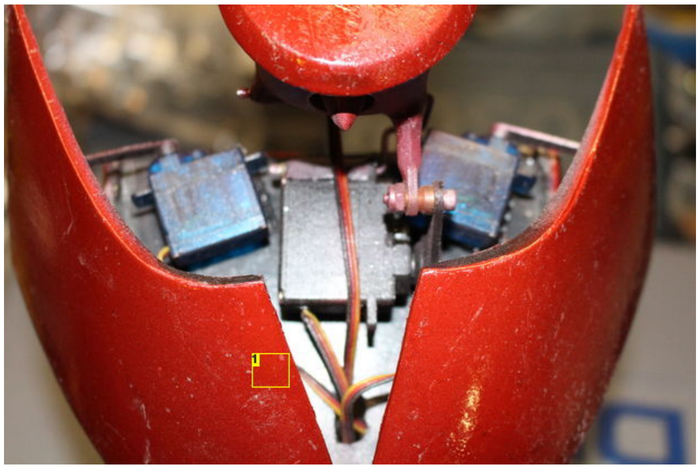


Image Notes
1. Servo wires run through a hole in the base plate

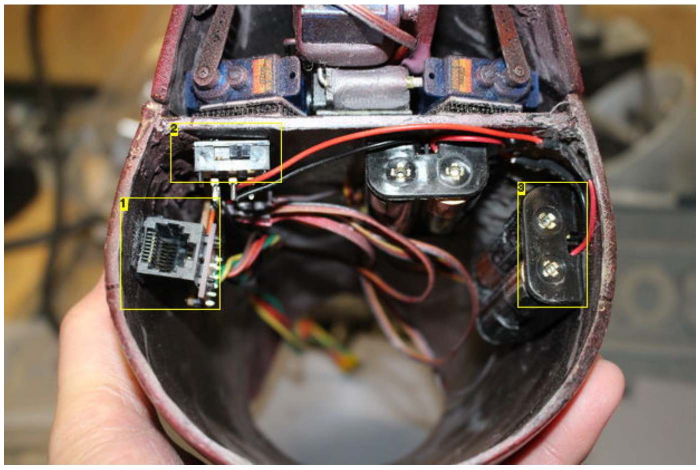


Image Notes
1. Ethernet connector
2. Power switch
3. AA battery holders

Step 6: Shoulder rocket pods

We really wanted opening shoulder rocket pods. Bad.

We just thought it would be so cool to see them open up and unfold like in the movie. The trouble was there was no way they could ever fit in there and perform in a similar fashion- the suit simply didn't have the necessary internal volume. There is about one inch of usable depth in the shoulder area. Those darn visual effects again...

Once I began taking measurements with Greg in the suit I wasn't so sure it could be done. I think I probably sketched a couple hundred designs trying to figure out a way to have the pod raise up and fold over. This was a key feature I really, really wanted. I didn't want to have a pod that opened like one of those flip up car headlamps. I knew I couldn't have it perform like in the film with the entire shoulder section moving but I felt I could get something that looked really cool.

I finally settled on having a section of the shoulder split into two panels- one rotating forward and one rotating backward. This would give a decent sized opening for a proper size rocket pod. The big trick was making the pod an open box so the servo could hide inside it- without that there was no way it would ever fit in the shoulder cavity.

The pod is constructed of birch plywood and pivots are fabricated from brass tube and music wire. As the servo arm rotates the rear of the rocket pod is pushed away from the servo. The pod servo is mounted to an Aluminum plate that is attached to the forward shoulder panel servo. The rear panel servo has a similar Aluminum plate attached to it. The plates have threaded holes in them for attaching the shoulder panels to them with small brass angle brackets. The holes in the brass angle brackets are slightly slotted to allow for a small range of height adjustment of the shoulder panel.

There was an enormous amount of trial and error fitting as once the shoulder was cut out of the fiberglass suit there would be no turning back- we had to be right on the money the first time. The panel openings were taped off and cut out with a Dremel tool and then the servos were mounted inside the shoulder area using plywood mounts epoxied into place. Getting the panels to fit just right was a real challenge!

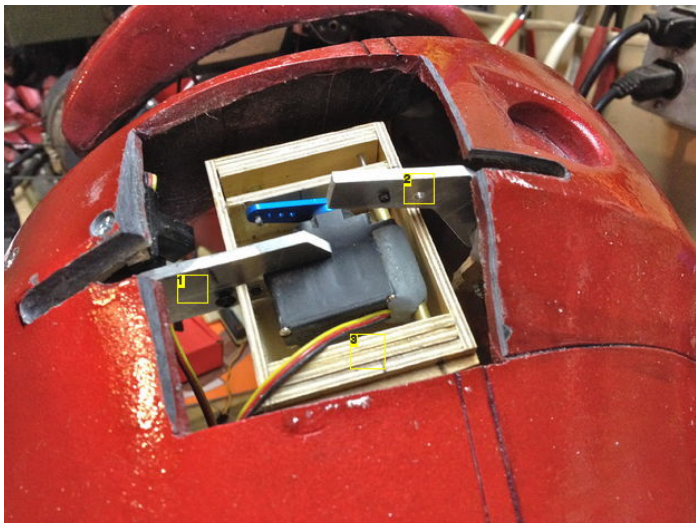


Image Notes
1. Aluminum arm for the rocket pod and forward panel
2. Aluminum arm for the rear panel
3. Left shoulder rocket pod with panels removed

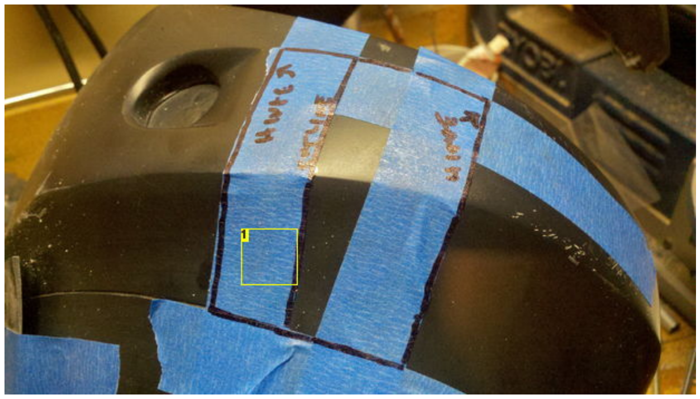


Image Notes
1. Marking the area needed for the shoulder rocket pods

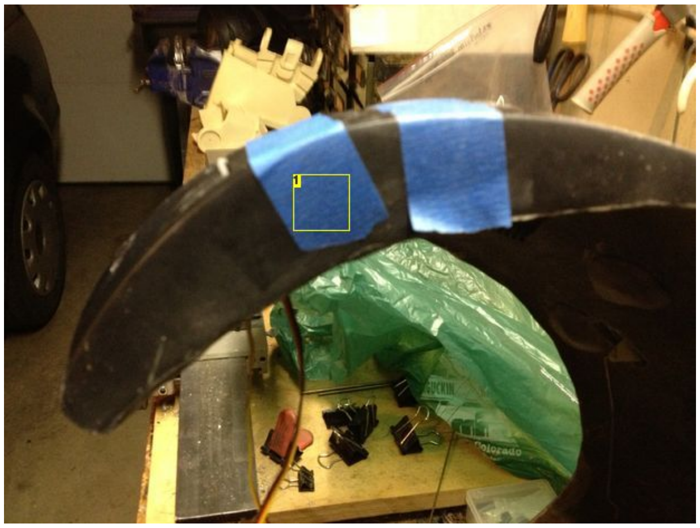


Image Notes
1. There's not much room in there! There's only about 1" of space available from the shoulder to the bottom of the inside of the armor.

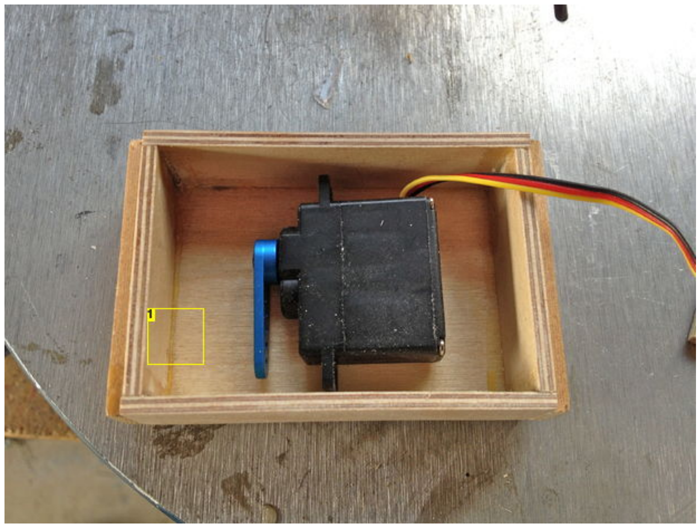


Image Notes
1. Plywood box based on the size of the servo and available room in shoulder cavity

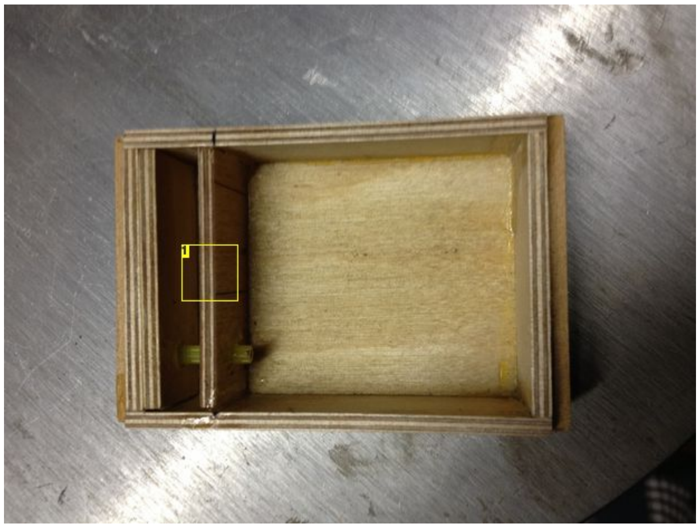


Image Notes
1. Support brace for rear pivot tube

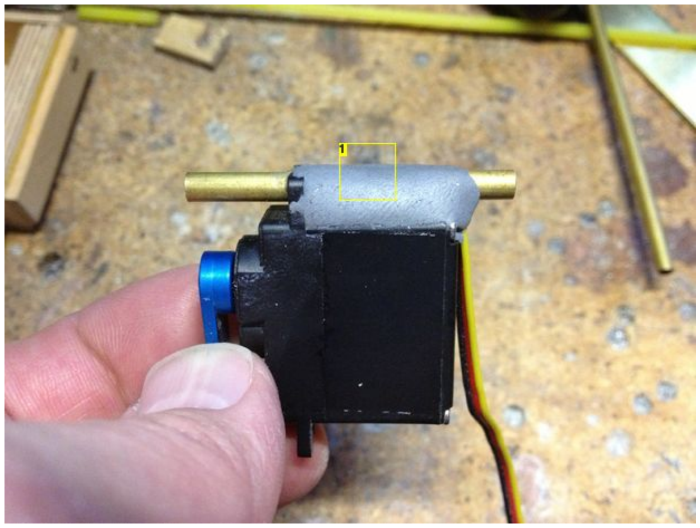


Image Notes
1. Brass forward pivot tube secured in place with epoxy putty. The tube runs

right through the servo mounting hole.

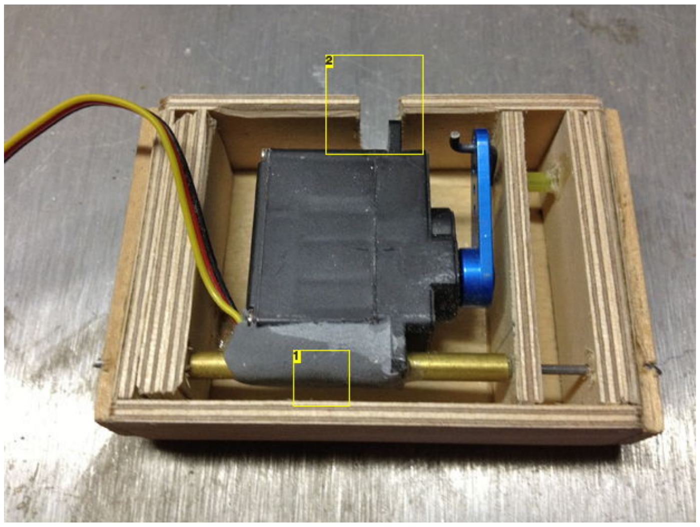


Image Notes
1. Pivot pins are made from music wire.
2. Notch cut in back of rocket pod for Aluminum servo mounting arm.

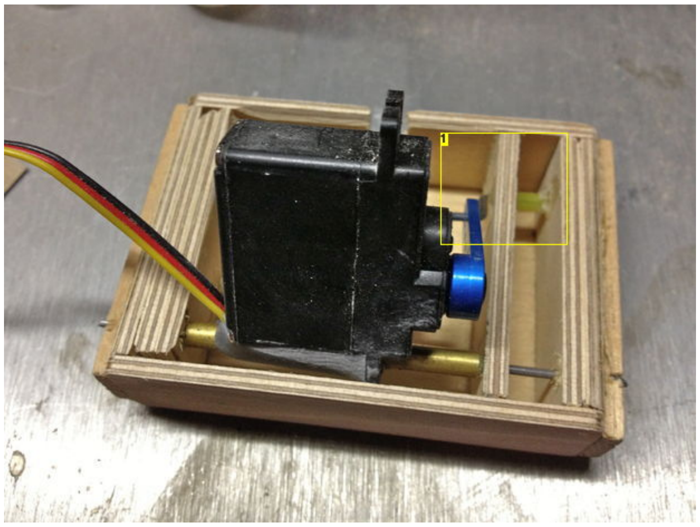


Image Notes
1. As the servo arm rotates the secondary pivot arm pushes the rocket pod away from the servo

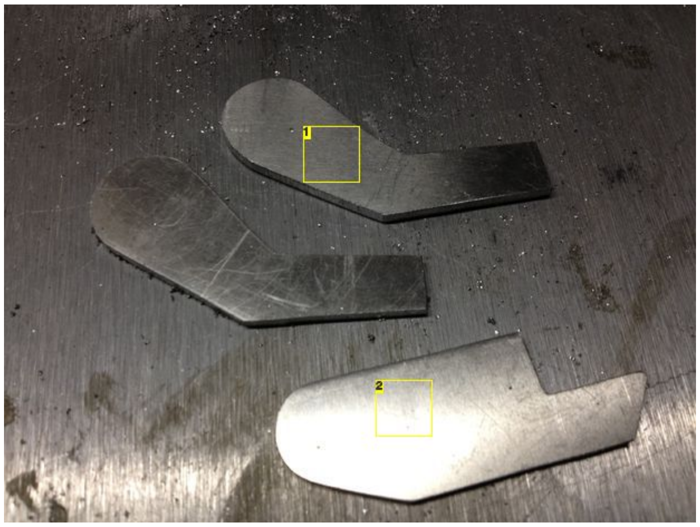


Image Notes
1. Aluminum arm for the rear shoulder panel
2. Aluminum arm for the forward shoulder panel/rocket pod servo mount

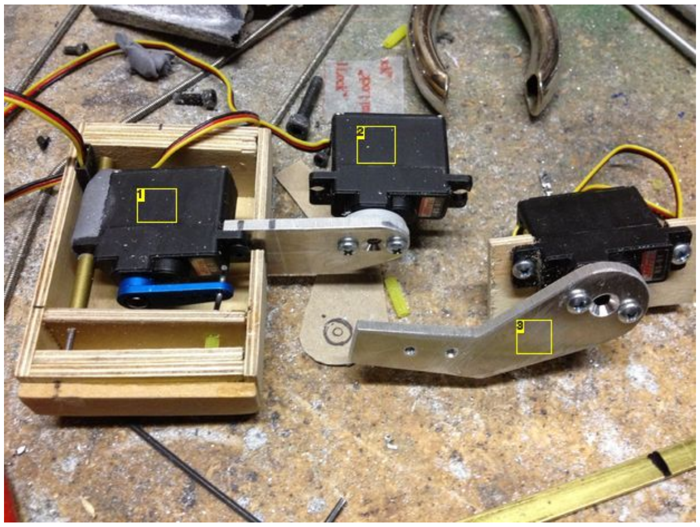
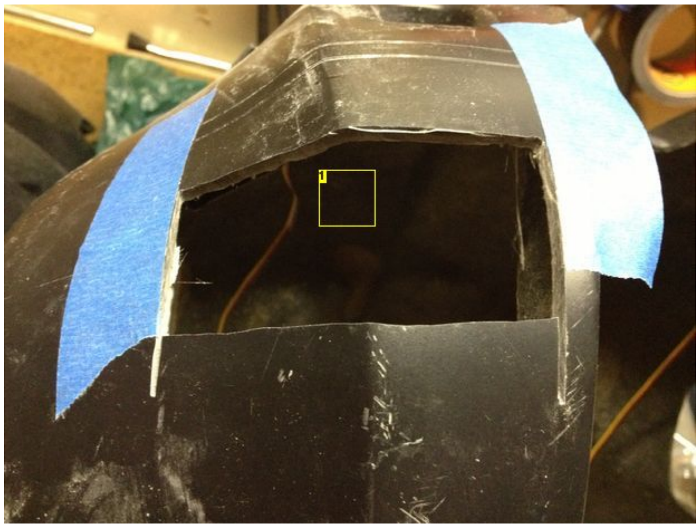


Image Notes
1. Rocket pod servo
2. Forward panel servo
3. Rear panel servo



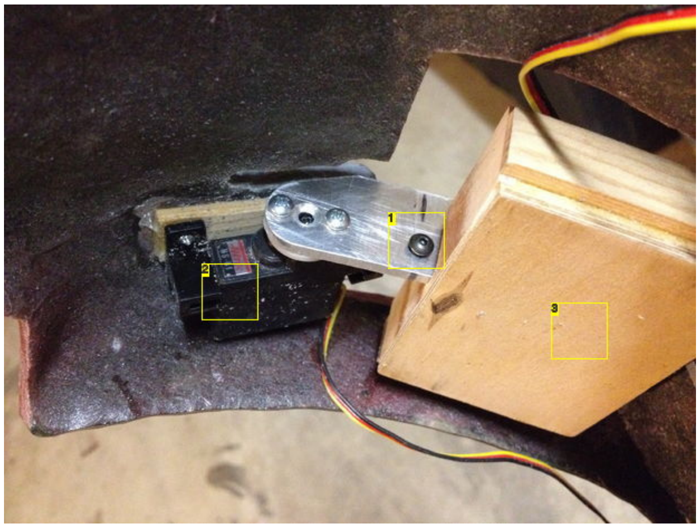


Image Notes
1. Cutting out the shoulder panels. No turning back now!

Image Notes
1. 4-40 mounting screw that holds the panel support bracket
2. Left forward panel servo mounted in place
3. Left rocket pod

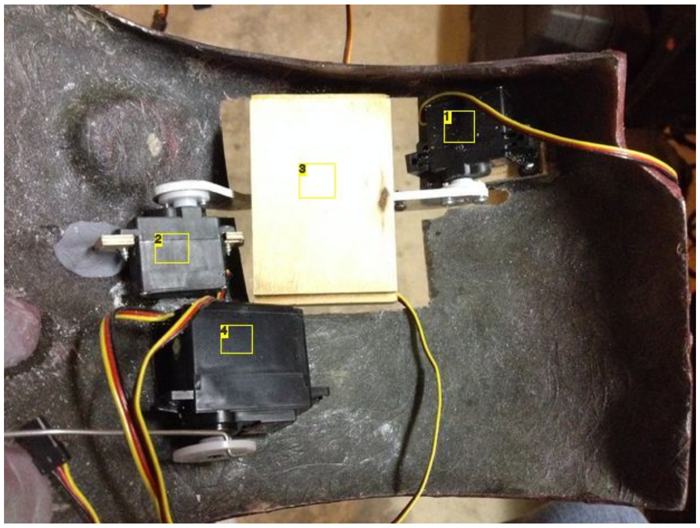


Image Notes
1. Left shoulder forward panel servo
2. Left shoulder rear panel servo
3. Left shoulder rocket pod
4. Left shoulder back flap servo

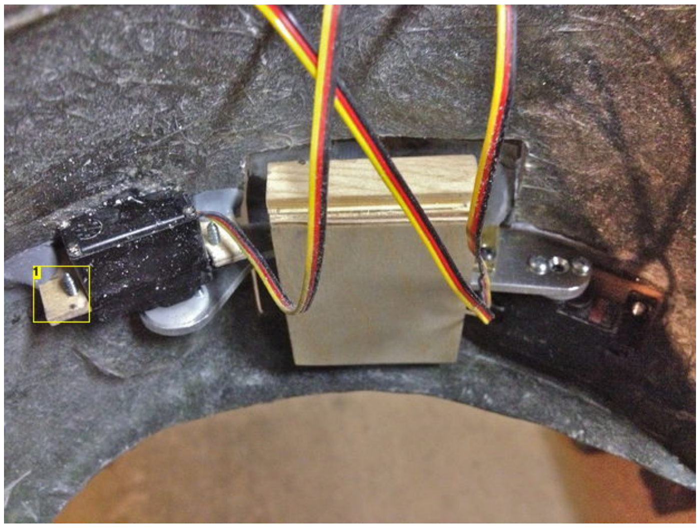


Image Notes
1. Plywood servo mount epoxied in place

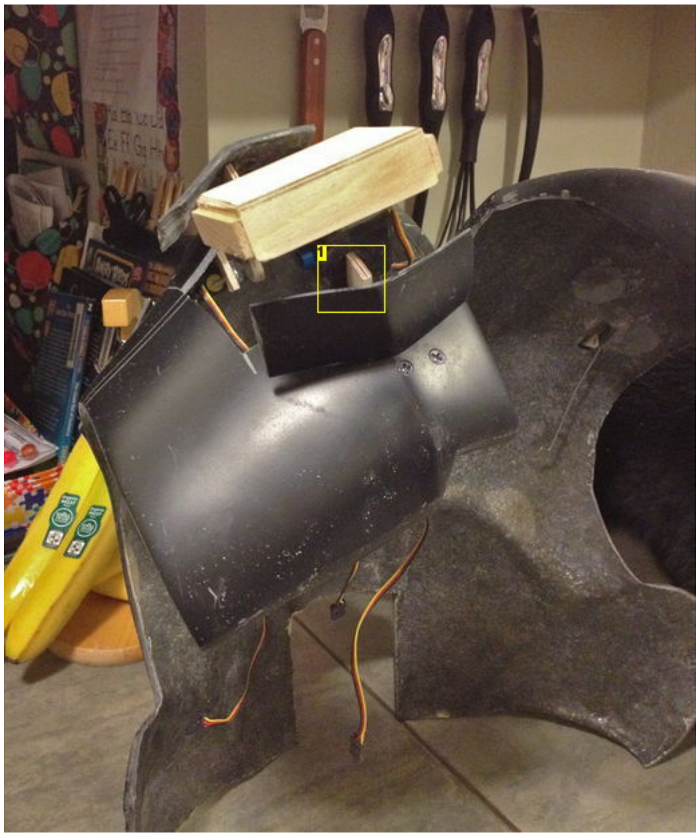


Image Notes
1. Early test of the shoulder rocket pods. Early tests were done with plywood panel arms before cutting them from Aluminum.

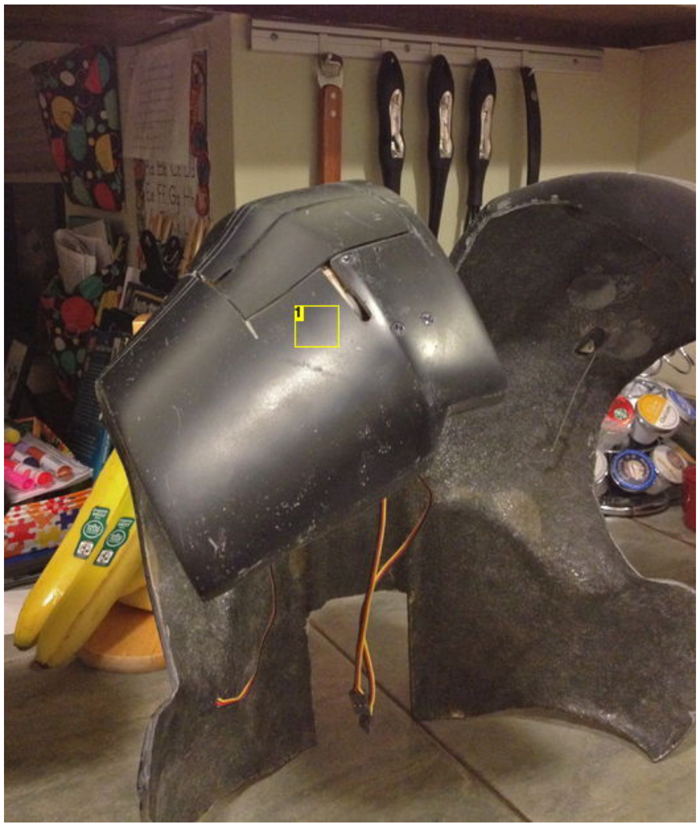


Image Notes
1. The shoulder panels closed. Note the slot needed for the forward panel arm.

Image Notes
1. Testing out the shoulder rocket pods

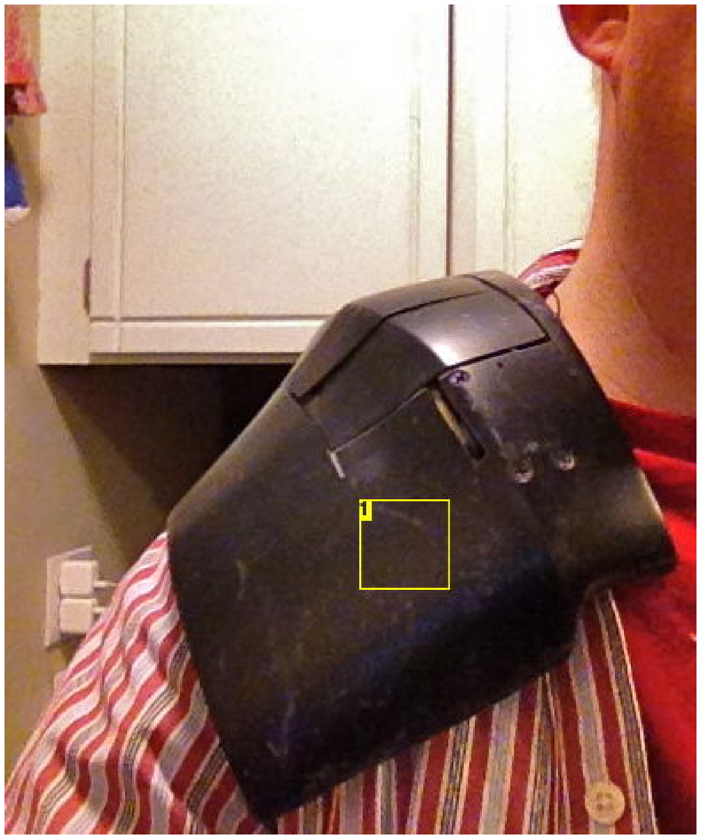


Image Notes
1. Everything actually fit! You can see the panel fit still needs to be adjusted.

Step 7: Hip pods

Originally we hadn't planned on adding the hip pods.

Greg now knows me well enough to know that I tend to first say it can't be done or isn't probable and then fifteen minutes later I figure it out- that's just my M.O. The hip pods are a perfect example of this. I first looked at the suit and said "no way." But of course it bugged me because it would be so cool to have them move... So first I figured out how to make them pop out of the hip section. Then I thought it would be great if they could rotate. Then I thought, well heck- might as well make the lever on the cover slide and have them light up!

To make the pods pop out I took two small hinges and welded them together to make a parallelogram linkage and then added a micro servo to move the linkage. The finished mechanism is very low profile. Getting the pods to fit right was tricky because the fiberglass hip section with was molded as one piece so everything had to be cut apart and reconstructed. The faceplate of the pod was cut away and hollowed out to make a shell and a housing was made from ABS pipe. A backing plate was cut from birch plywood.

Several ideas were tried for the pod rotation system but ultimately the faceplate was driven directly by a servo as that took up the least amount of space. The rotation servo is mounted to a piece of plywood that is bolted to the hinge assembly. LEDs were mounted in the ABS ring to simulate the flares.

A servo wheel is mounted to the rotation servo and it drives the pod faceplate. To make the lever on the faceplate slide open a sub-micro servo is mounted to a plywood plate that is attached to the servo wheel. The servo output arm has a small slot cut in it and it is attached to the sliding lever using with a small section of music wire that is epoxied in place. The lever slides on a small hinge made from music wire and brass tubing- the hinge is attached to the large servo wheel. As the servo lever moves the hinge rotates slightly outward and the lever slides open. This particular mechanism required a lot of trial end error fitting to get it to move smoothly with very little friction.

Since the suit hip section had been cut away in order to use the pods we had to reconstruct the flanges on the back hip section. Sintra sheet was cut and formed to shape and was epoxied in place and then the seams were filled in with Apoxie Sculpt. In the end the hip pods worked really well and I'm glad we went to the trouble of adding them!

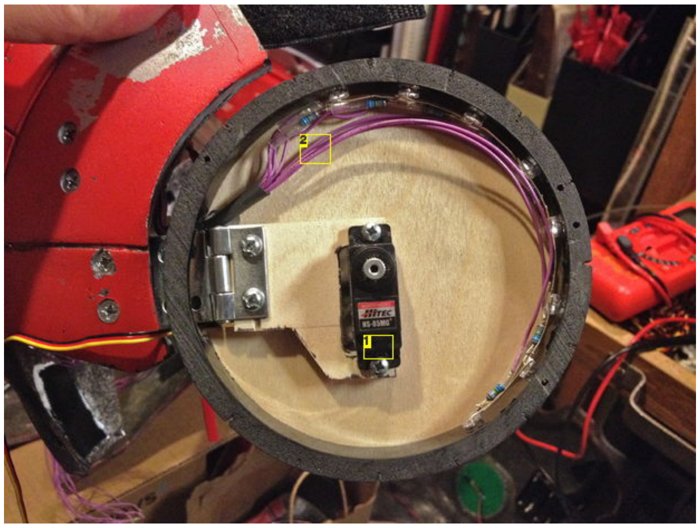


Image Notes
1. Pod cover rotation servo
2. Wiring for hip pod LEDs with 100 Ohm resistors

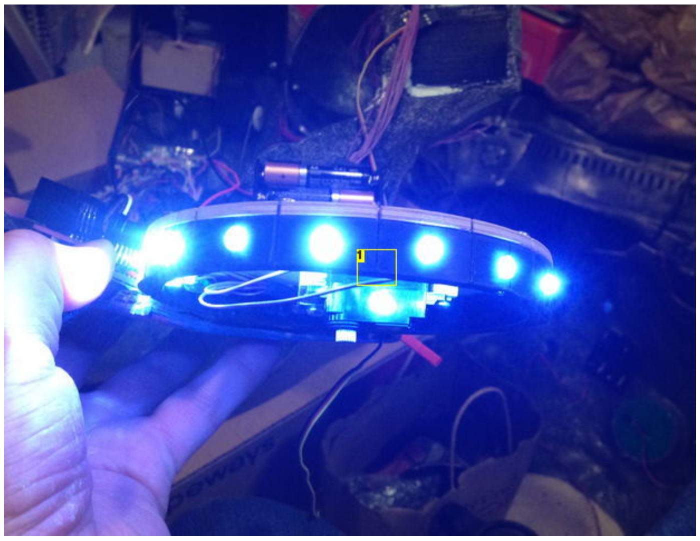


Image Notes
1. Hip pod LEDs

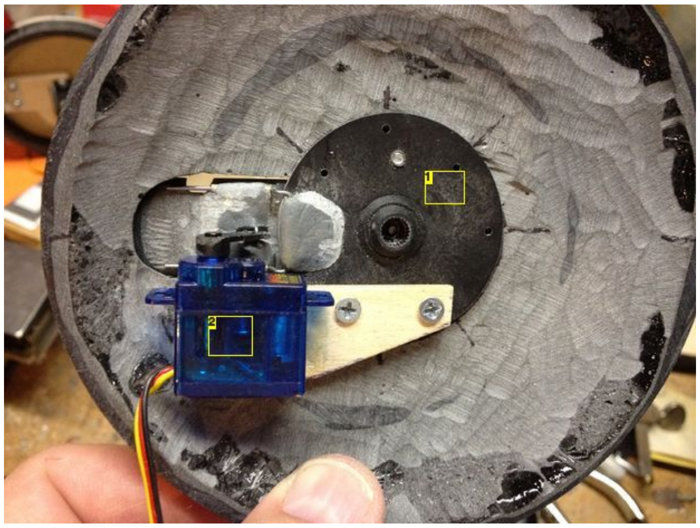


Image Notes
1. Servo wheel is screwed to the hip pod shell
2. Servo to move hip pod sliding panel. This is epoxied to the plywood plate that is screwed to the servo wheel.

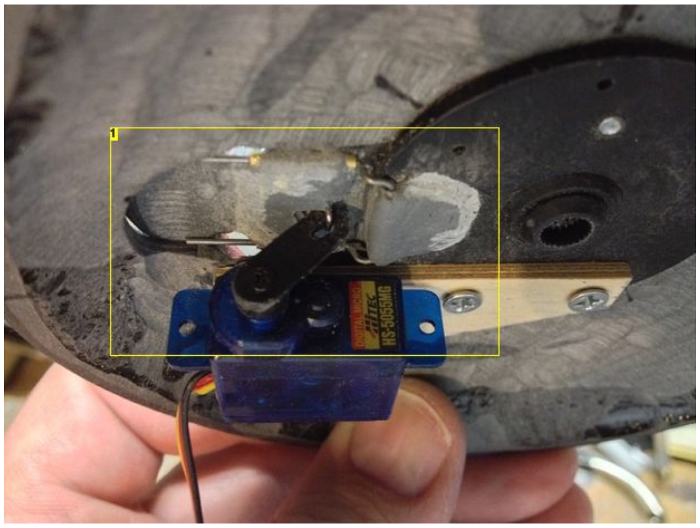


Image Notes
1. Mechanism for sliding hip pod panel. The servo arm is slotted and a "U" shaped piece of music wire is epoxied to the panel. As the servo rotates the panel slides on the music wire hinge that is epoxied to the servo wheel.

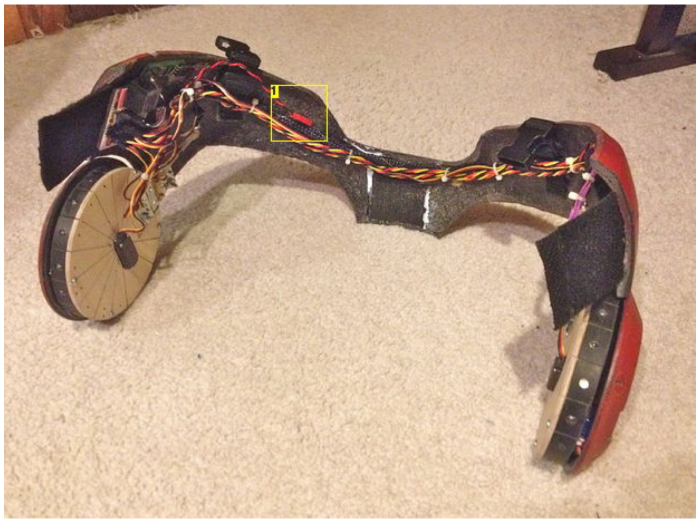
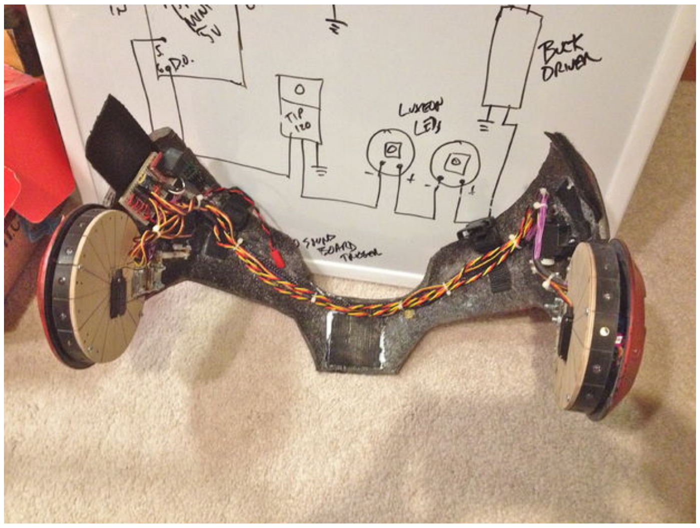


Image Notes
1. JST connector to provide power to the hip pods



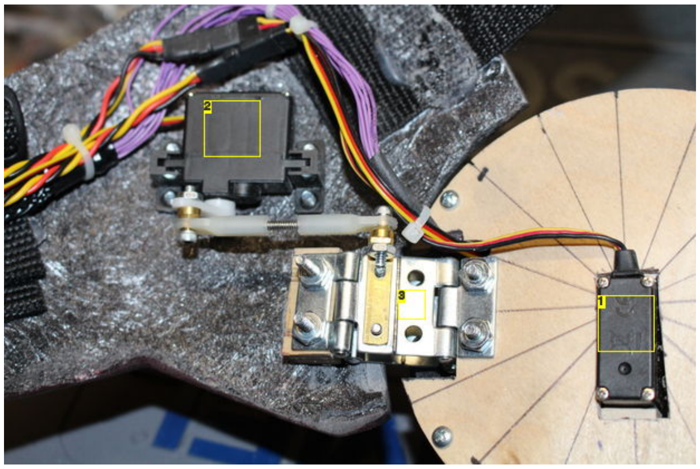


Image Notes
1. Right hip pod rotation servo
2. Right hip pod servo
3. A pair of hinges makes up the parallelogram that pushed the hip pod outward

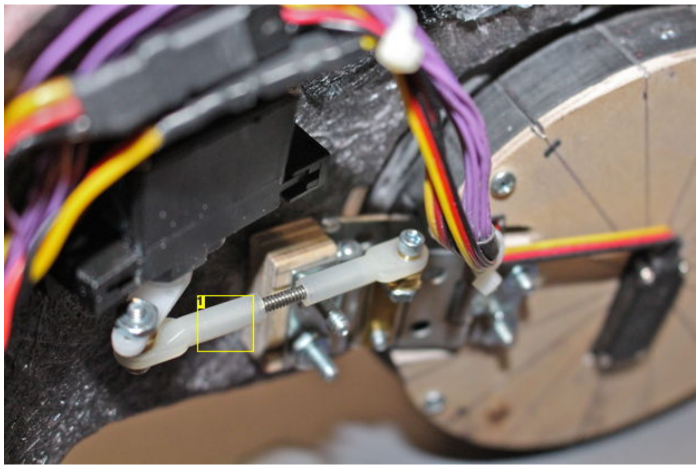


Image Notes
1. 4-40 linkage that moves the hip pod

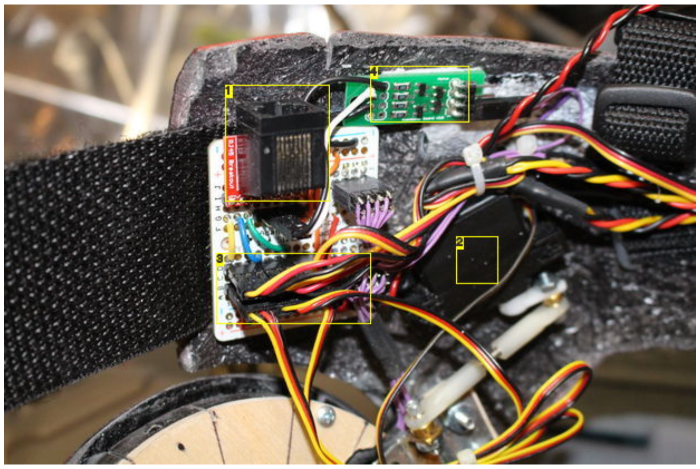


Image Notes
1. Hip pod Ethernet connector
2. Left hip pod servo
3. Servo connectors
4. Transistors to turn on the hip pod LEDs

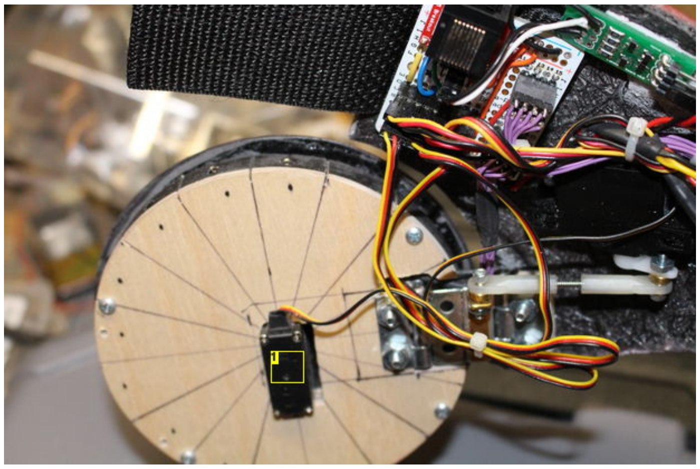


Image Notes
1. Left hip pod rotation servo

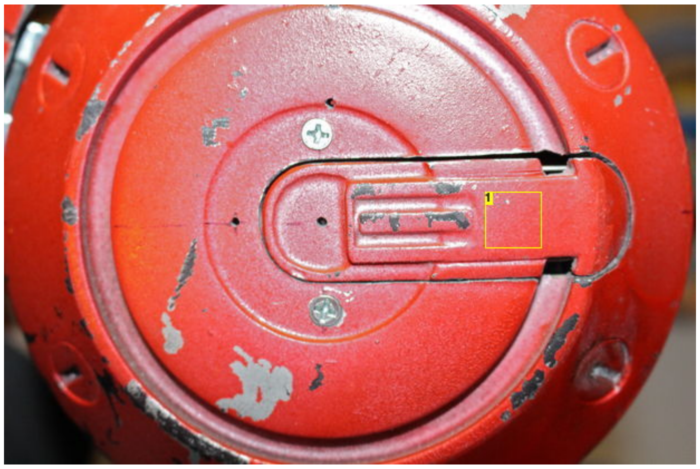


Image Notes
1. Hip pod sliding panel

Step 8: Back flaps

The back flaps are fairly simple to animate.

The flaps were first held in place with tape in order to locate the hinges. Hinges were constructed using brass rod and tube and they were epoxied to the back of the flaps. The hinges were then mounted to the back of the upper torso by drilling holes and using epoxy putty to bond them in place. Control horns were mounted to the flaps and slots were cut in the upper back torso section so the control horns and connecting rods could fit through. The servos to move the back flaps were mounted to the inside of the upper back torso using high strength Velcro.

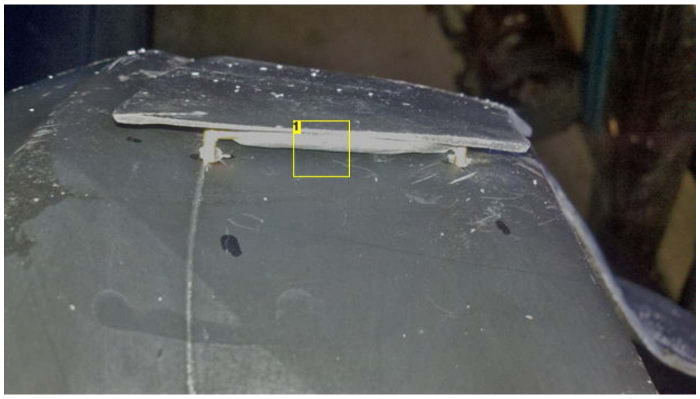


Image Notes
1. Brass rod and tube hinge epoxied to the underside of the flap

Image Notes
1. ProPoxy 20 epoxy putty is formed around the hinge rods to hold the hinge in place

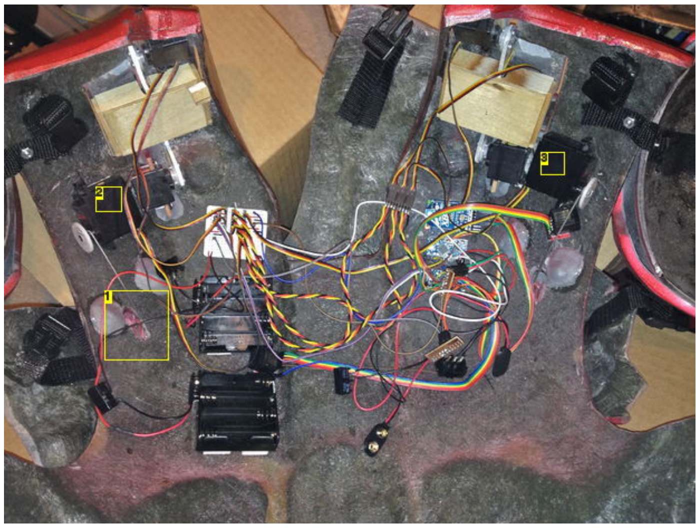


Image Notes
1. Slot is cut in the back for the control rods, which are made from music wire. The electronics are temporarily held in place for testing.
2. Right back flap servo
3. Left back flap servo

Step 9: Boots

We knew that we wanted to boots to light up and make a robotic clanking sound while walking.

I figured the easiest way to trigger this effect was to use a distance sensor on the underside of the boot- in this case the Sharp GP2D120XJ00F IR sensor. The sensor reads the distance from the boot to the ground and then a threshold value is set in the code so when the value exceeds the threshold the sensor tells the Arduino to turn on the light and activate the sound. Simple!

A cavity was carved out of the bottom of the boot with a Dremel tool and the sensor was mounted in place. A hole was drilled through for the sensor wires as well as the wires for the high power Luxeon LED. The wires for these were bundled together and a connector was soldered on so they could be easily disconnected from the wires that ran down through the legs of the suit. Only one sensor is needed to trigger the Arduino so the sensor was mounted in the right boot since Greg begins walking with his right foot first.

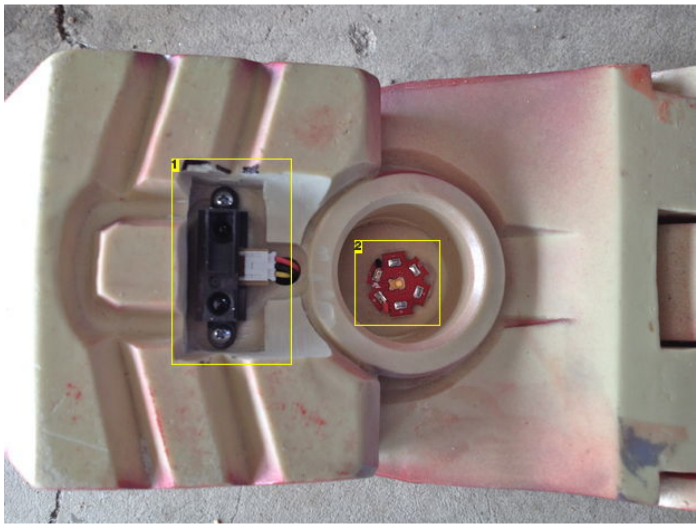


Image Notes
1. Sharp IR sensor
2. Luxeon LED

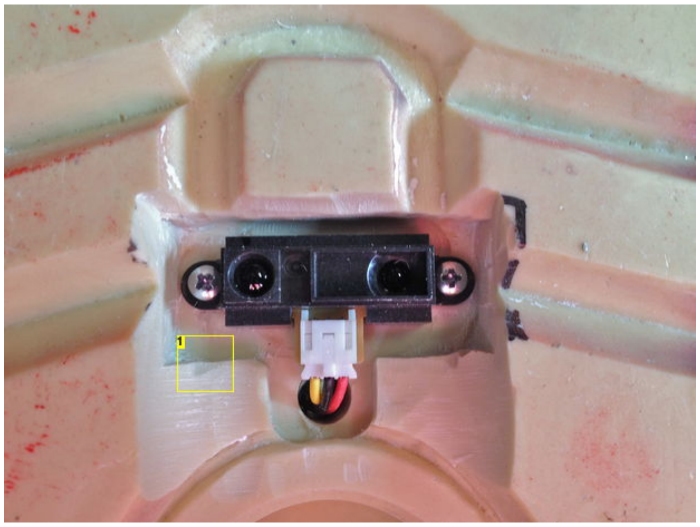


Image Notes
1. Cavity is cut in the bottom of the boot so the sensor is below the surface of the bottom of the boot

Step 10: Electronics

The electronics for this suit aren't really that complex once it's broken down into separate systems.

The reason I wanted to make the left side/right side and boots separate systems was so they could be run at the same time and if one system went down the rest of the suit would still operate.

The left side takes the RFID tag input from two fingers on the left hand and then the Arduino will operate either the helmet functions (via the XBee radio) or it will have the hip pods and back flaps run through a programmed sequence.

The right side takes the RFID tag input from two fingers on the right hand and then a second Arduino operates either the hip pods or shoulder rockets, depending on which tag is read. If the hip pods are selected then the WaveShield is also triggered to play a sound effect.

The IR sensor in the right boot sends a signal to another Arduino that operates the lights for the boots and triggers the WaveShield to play a sound effect.

For testing purposes I glued RFID tags and a tag reader to a glove to get an idea as to how easy it would be to operate, since the tag reader can only read one tag at a time. Reading two tags at the same time gives zero output. I was worried that since the fingers were in close proximity to one another this could be a problem, but it turns out it worked just fine.

The tag reader was then mounted to the inside of the fiberglass suit glove shell using adhesive foam tape. The back side of the board was then taped over to protect it and the lead wires. The glove clam shells fit over a batting glove so the wearer's hand never comes in contact with the board. The gloves have extension leads that connect to the gauntlets, which have Ethernet jacks for connecting to the Ethernet cables that run through the arms. The left gauntlet is pretty much empty while the right gauntlet has AA battery holders as well as a small connector board for the servo wires.

The three Arduino ProMinis are mounted in the back of the upper torso section along with batteries and power switches for each system. The WaveShield sits atop the Arduino Pro in the center. The transmitting XBee radio for the helmet is visible in the upper corner. There are also several Ethernet jacks visible- two for the arms, two for the legs and one for the Ethernet cable that runs to the hip section. Also visible is a board that has a few transistors on it- these take the signals from the Arduinos and turn on the boot lights and trigger the sound effects via the Wave Shield. The WaveShield output is boosted by a small amplifier board. There is a small breadboard PCB in the upper corner that has connectors for the shoulder rocket servos and back flap servos. The two speakers were salvaged from an old monitor.

The boards are secured using foam tape as it holds them securely but they can still be removed and if wires get pulled nothing will get damaged. The Ethernet cables were also secured using hot glue a bit away from the connectors in order to provide some strain relief.

If I was to do it again I would probably create a single board down the center and have the Arduinos socketed along with a socketed transistor board using SMD transistors. I would also have the servo connectors on the center board. That would go a log way toward cleaning up the wiring.

The hip pod section uses a small breadboard PCB with an Ethernet connector to route the signals for the servos and LEDs in the pods. There is a small transistor board that is used to turn on the pod LEDs. The wires from the LEDs were run through the back of the pods near the hinge and heatshrink tubing was used protect the wires from potential damage caused by hinge movement.

Finally the AA batteries that provide power to the hip pod servos were mounted to the inside of the chest section near the chest light along with a switch. A power lead with a JST connector was run to the hip pod section.

Much to my surprise, everything actually worked the first time it powered up (even though I had breadboarded the circuits for testing.) The only thing I had to do was make adjustments to timing and servo movements in the code.

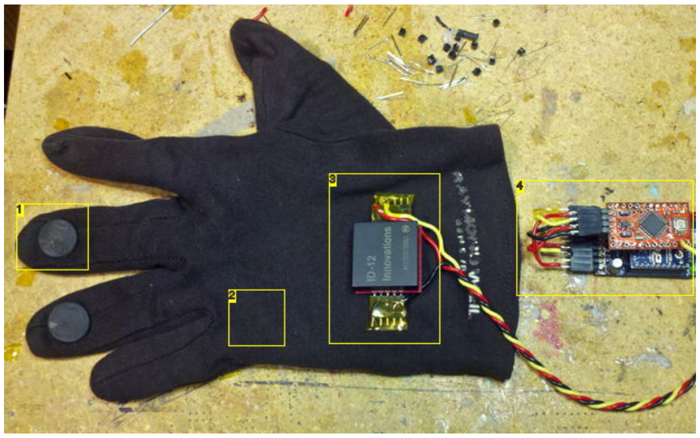


Image Notes
1. RFID tags
2. Glove for testing the tag reader
3. ID12 RFID tag reader
4. Arduino ProMini and XBee radio. The XBee radio is plugged into a Adafruit XBee adapter.

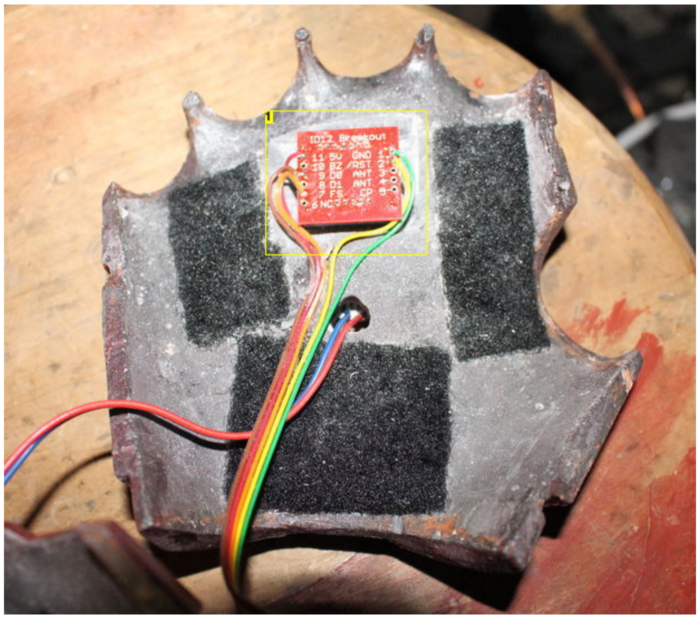


Image Notes
1. ID12 RFID tag reader. This is secured to the inside of the glove shell with double sided foam tape

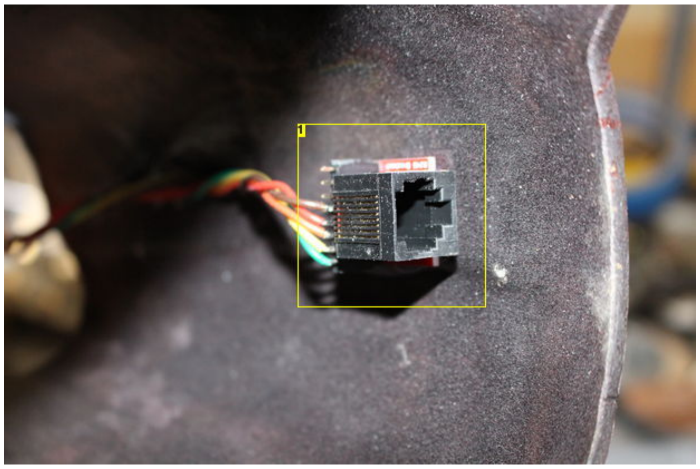


Image Notes
1. Ethernet connector in left gauntlet

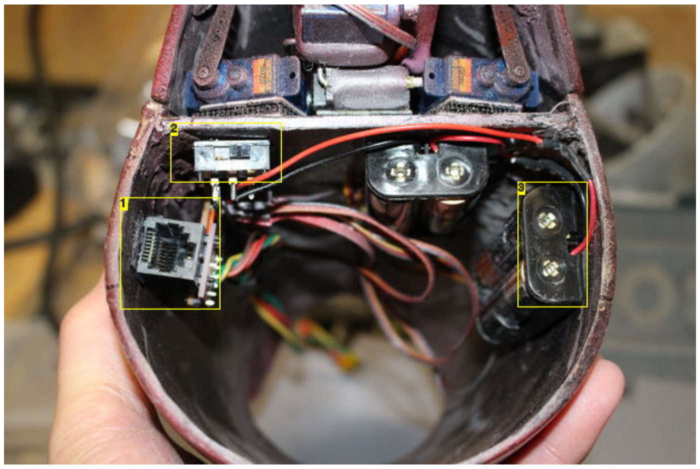


Image Notes
1. Ethernet connector
2. Power switch
3. AA battery holders

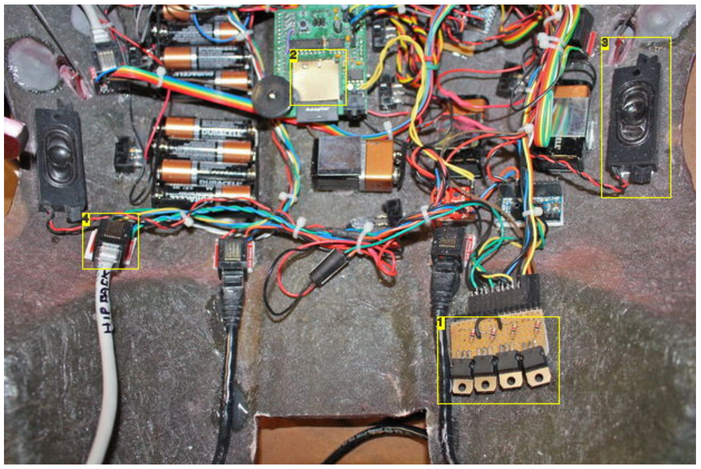


Image Notes
1. TIP120 transistors
2. Adafruit WaveShield on top of Arduino Pro
3. Speakers were salvaged from an old monitor
4. Ethernet connectors

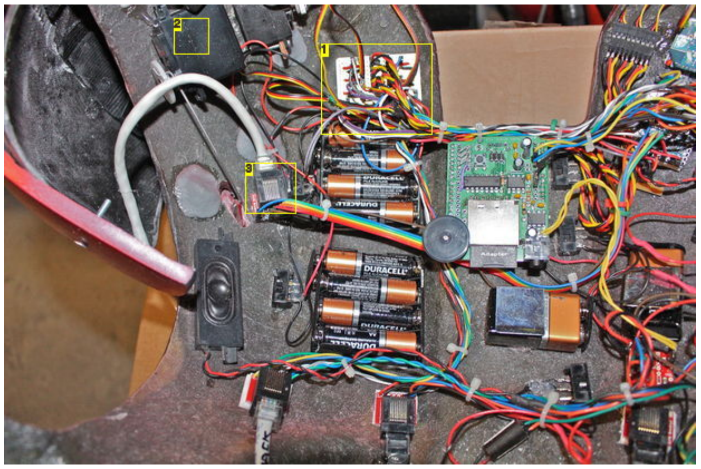


Image Notes
1. Servo connector board for shoulder rockets and back flaps
2. Right flap servo
3. Ethernet connector for right arm

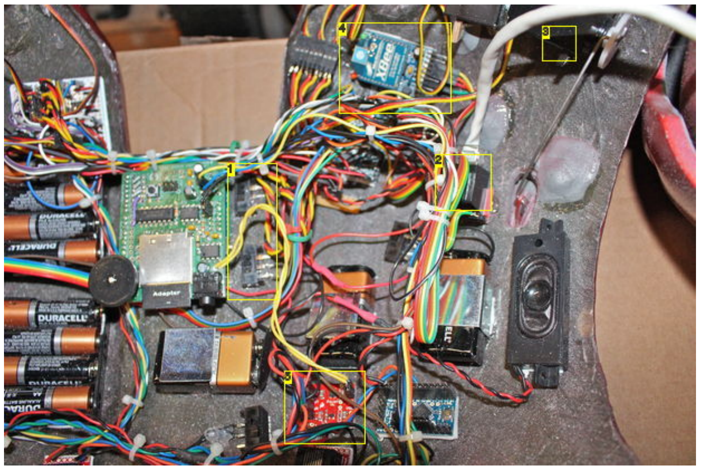


Image Notes
1. Power switches
2. Ethernet connector for left arm
3. Left flap servo
4. XBee radio
5. Amp board to boost output of WaveShield

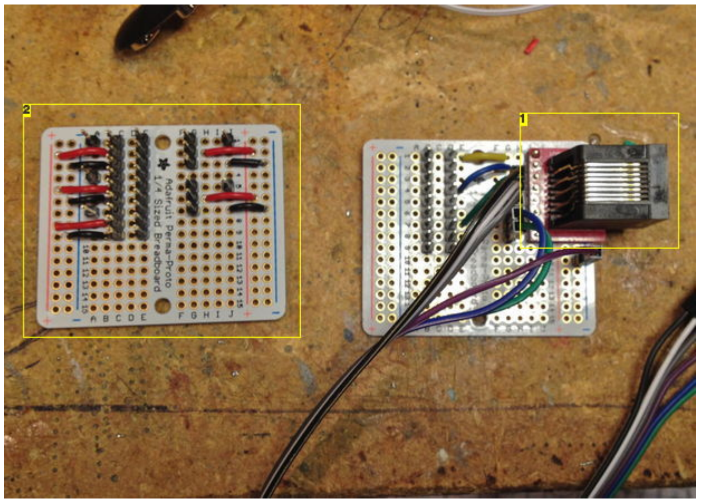


Image Notes
1. Hip pod board with Ethernet connector
2. Servo connector board for shoulder rockets and back flaps

Step 11: Electronics- left side schematic and code

Here's the schematic for the left side control system.

There are a few notes on the diagram but it's pretty straightforward. When parts were purchased for the suit the ID12 tag reader was only available in a 5V version, which was powered by the Arduino. Since the servos are powered by a 6V battery pack it was easiest to just use a 9V battery for the Arduinos and 6V packs for the servos because you need to isolate the power supply for the Arduino due to the electrical noise generated by the servos.

Now that the ID12 is available in a low voltage version it would be simpler to power everything from a 6V battery pack and use a 3.3V Arduino ProMini and use a 3.3V DC/DC converter to supply isolated power the Arduino.

Here's the code-

```
#include <NewSoftSerial.h>
#include "Servo.h" // include the servo library

Servo podServo; // servo to move hip pods
Servo leverServo; // servo to move hip pod levers
Servo rotateServo; // servo to rotate hip pods
Servo leftflapServo; // servo to move left back flap
Servo rightflapServo; // servo to move right back flap

NewSoftSerial mySerial = NewSoftSerial(2, 3);

int RFIDResetPin = 13;
int ledPin1 = 6; // control pin for left hip pod LEDs
int ledPin2 = 5; // control pin for right hip pod LEDs
int servoPin1 = 10; // control pin for left flap servo
int servoPin2 = 11; // control pin for right flap servo
int servoPin3 = 9; // control pin for pod servo
int servoPin4 = 8; // control pin for lever servo
int servoPin5 = 7; // control pin for rotate servo
int soundPin = 12; // control pin for flare sound

//Register your RFID tags here
char tag1[13] = "440085E77452";
char tag2[13] = "440085FC330E";
char tag3[13] = "440085F97840";
char tag4[13] = "4400863914EF";


void setup(){
Serial.begin(9600);
mySerial.begin(9600);

podServo.attach(servoPin3); // attaches the servo on pin 9 to the servo object
leverServo.attach(servoPin4); // attaches the servo on pin 8 to the servo object
rotateServo.attach(servoPin5); // attaches the servo on pin 7 to the servo object
leftflapServo.attach(servoPin1); // attches the servo on pin 10 to the servo object
rightflapServo.attach(servoPin2); // attaches the servo on pin 11 to the servo object
podServo.write(155); // rotate the pod servo to 135 degrees
leverServo.write(145); // rotate the lever servo to 135 degrees
rotateServo.write(165); // rotate the pod rotation servo to 170 degrees
leftflapServo.write(170); // rotate the left flap servo to 170 degrees
rightflapServo.write(10); // rotate the right flap servo to 10 degrees
pinMode(ledPin1, OUTPUT); // sets the LED pin as an output
pinMode(ledPin2, OUTPUT); // sets the LED pin as an output
digitalWrite(ledPin1, LOW); // turn off LEDs
digitalWrite(ledPin2, LOW); // turn off LEDs
pinMode(soundPin, OUTPUT); // sets the sound pin as output
digitalWrite(soundPin, LOW); // turn off sound pin


pinMode(RFIDResetPin, OUTPUT);
digitalWrite(RFIDResetPin, HIGH);

}

void loop(){

char tagString[13];
int index = 0;
boolean reading = false;

while(Serial.available()){

int readByte = Serial.read(); //read next available byte

if(readByte == 2) reading = true; //begining of tag
if(readByte == 3) reading = false; //end of tag

if(reading && readByte != 2 && readByte != 10 && readByte != 13){
//store the tag
tagString[index] = readByte;
index ++;
}
}

checkTag(tagString); //Check if it is a match
clearTag(tagString); //Clear the char of all value
resetReader(); //eset the RFID reader
}

void checkTag(char tag[]){
///////////////////////////////////
//Check the read tag against known tags
///////////////////////////////////

if(strlen(tag) == 0) return; //empty, no need to contunue

if(compareTag(tag, tag1)){ // if matched tag1, do this
mySerial.print('A');

}else if(compareTag(tag, tag2)){ //if matched tag2, do this
podServo.write(90); // rotate the pod servo to 90 degrees
delay(500); // wait half a second
leverServo.write(95); // rotate the lever servo to 90 degrees
delay(1000);
rotateServo.write(5); // rotate the pod rotation servo to 10 degrees
delay(1500);
leverServo.write(145);
delay(500);
digitalWrite(soundPin, HIGH); // turn sound on
delay(10); // wait ten milliseconds
```

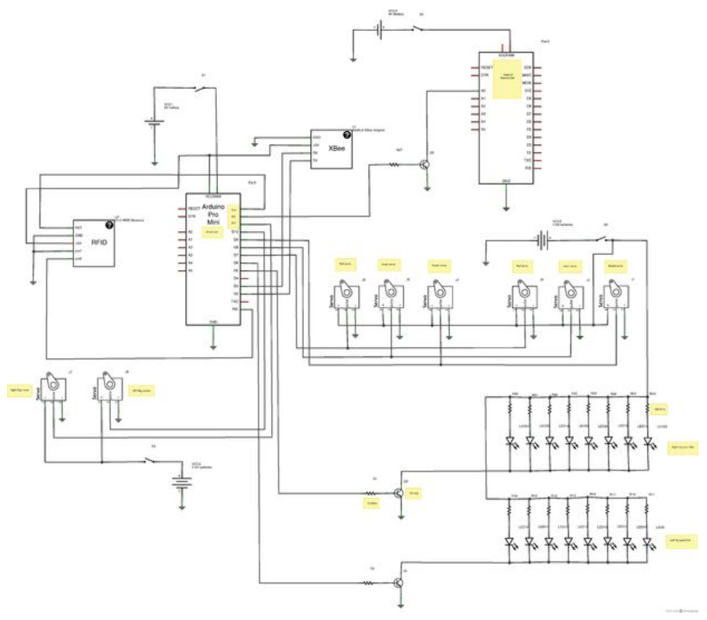
```
digitalWrite(soundPin, LOW); // turn sound off
digitalWrite(ledPin1, HIGH); // turn on LEDs
digitalWrite(ledPin2, HIGH); // turn on LEDs
delay(50); // wait 50 milliseconds
digitalWrite(ledPin1, LOW); // turn off LEDs
digitalWrite(ledPin2, LOW); // turn off LEDs
delay(50);
digitalWrite(ledPin1, HIGH); // turn on LEDs
digitalWrite(ledPin2, HIGH); // turn on LEDs
delay(50);
digitalWrite(ledPin1, LOW); // turn off LEDs
digitalWrite(ledPin2, LOW); // turn off LEDs
delay(50);
digitalWrite(ledPin1, HIGH); // turn on LEDs
digitalWrite(ledPin2, HIGH); // turn on LEDs
delay(50);
digitalWrite(ledPin1, LOW); // turn off LEDs
digitalWrite(ledPin2, LOW); // turn off LEDs
delay(50);
digitalWrite(ledPin1, HIGH); // turn on LEDs
digitalWrite(ledPin2, HIGH); // turn on LEDs
delay(50);
digitalWrite(ledPin1, LOW); // turn off LEDs
digitalWrite(ledPin2, LOW); // turn off LEDs
delay(50);
digitalWrite(ledPin1, HIGH); // turn off LEDs
digitalWrite(ledPin2, HIGH); // turn off LEDs
delay(50);
digitalWrite(ledPin1, LOW); // turn off LEDs
digitalWrite(ledPin2, LOW); // turn off LEDs
delay(50);
digitalWrite(ledPin1, HIGH); // turn on LEDs
digitalWrite(ledPin2, HIGH); // turn on LEDs
delay(50);
digitalWrite(ledPin1, LOW); // turn off LEDs
digitalWrite(ledPin2, LOW); // turn off LEDs
delay(50);
digitalWrite(ledPin1, HIGH); // turn on LEDs
digitalWrite(ledPin2, HIGH); // turn on LEDs
delay(50);
digitalWrite(ledPin1, LOW); // turn off LEDs
digitalWrite(ledPin2, LOW); // turn off LEDs
delay(50);
digitalWrite(ledPin1, HIGH); // turn on LEDs
digitalWrite(ledPin2, HIGH); // turn on LEDs
delay(50);
digitalWrite(ledPin1, HIGH); // turn off LEDs
digitalWrite(ledPin2, HIGH); // turn off LEDs
delay(50);
digitalWrite(ledPin1, LOW); // turn off LEDs
digitalWrite(ledPin2, LOW); // turn off LEDs
delay(50);
digitalWrite(ledPin1, HIGH); // turn on LEDs
digitalWrite(ledPin2, HIGH); // turn on LEDs
delay(50);
digitalWrite(ledPin1, LOW); // turn off LEDs
digitalWrite(ledPin2, LOW); // turn off LEDs
delay(50);
digitalWrite(ledPin1, HIGH); // turn on LEDs
digitalWrite(ledPin2, HIGH); // turn on LEDs
delay(50);
digitalWrite(ledPin1, LOW); // turn off LEDs
digitalWrite(ledPin2, LOW); // turn off LEDs
delay(50);
digitalWrite(ledPin1, HIGH); // turn on LEDs
digitalWrite(ledPin2, HIGH); // turn on LEDs
delay(50);
digitalWrite(ledPin1, LOW); // turn off LEDs
digitalWrite(ledPin2, LOW);
digitalWrite(ledPin1, HIGH); // turn on LEDs
digitalWrite(ledPin2, HIGH); // turn on LEDs
delay(50);
digitalWrite(ledPin1, LOW); // turn off LEDs
digitalWrite(ledPin2, LOW); // turn off LEDs
delay(50);
digitalWrite(ledPin1, HIGH); // turn on LEDs
digitalWrite(ledPin2, HIGH); // turn on LEDs
delay(50);
digitalWrite(ledPin1, HIGH); // turn off LEDs
digitalWrite(ledPin2, HIGH); // turn off LEDs
delay(50);
digitalWrite(ledPin1, LOW); // turn off LEDs
digitalWrite(ledPin2, LOW); // turn off LEDs
```

```
delay(50);
digitalWrite(ledPin1, HIGH); // turn on LEDs
digitalWrite(ledPin2, HIGH); // turn on LEDs
delay(50);
digitalWrite(ledPin1, LOW); // turn off LEDs
digitalWrite(ledPin2, LOW); // turn off LEDs
delay(50);
digitalWrite(ledPin1, HIGH); // turn on LEDs
digitalWrite(ledPin2, HIGH); // turn on LEDs
delay(50);
digitalWrite(ledPin1, LOW); // turn off LEDs
digitalWrite(ledPin2, LOW); // turn off LEDs
delay(50);
digitalWrite(ledPin1, HIGH); // turn on LEDs
digitalWrite(ledPin2, HIGH); // turn on LEDs
delay(50);
digitalWrite(ledPin1, LOW); // turn off LEDs
digitalWrite(ledPin2, LOW);
digitalWrite(ledPin1, HIGH); // turn on LEDs
digitalWrite(ledPin2, HIGH); // turn on LEDs
delay(50);
digitalWrite(ledPin1, HIGH); // turn off LEDs
digitalWrite(ledPin2, HIGH); // turn off LEDs
delay(50);
digitalWrite(ledPin1, LOW); // turn off LEDs
digitalWrite(ledPin2, LOW); // turn off LEDs
delay(50);
digitalWrite(ledPin1, HIGH); // turn on LEDs
digitalWrite(ledPin2, HIGH); // turn on LEDs
delay(50);
digitalWrite(ledPin1, LOW); // turn off LEDs
digitalWrite(ledPin2, LOW); // turn off LEDs
delay(50);
digitalWrite(ledPin1, HIGH); // turn on LEDs
digitalWrite(ledPin2, HIGH); // turn on LEDs
delay(50);
digitalWrite(ledPin1, LOW); // turn off LEDs
digitalWrite(ledPin2, LOW); // turn off LEDs
leverServo.write(95); // rotate the lever servo to 90 degrees
delay(1500);
rotateServo.write(165); // rotate the pod servo to 135 degrees
delay(1000);
leverServo.write(145);
delay(500);
podServo.write(155); // rotate the pod servo to 135 degrees
delay(2000);
leftflapServo.write(125); // rotate the left flap servo to 125 degrees- full up
rightflapServo.write(55); // rotate the right flap servo to 55 degrees- full up
delay(500);
leftflapServo.write(170); // rotate the left flap servo to 170 degrees- full down
rightflapServo.write(10); // rotate the right flap servo to 10 degrees- full down
delay(500);
leftflapServo.write(125); // left flap full up
delay(500);
leftflapServo.write(170); // left flap full down
delay(500);
rightflapServo.write(55); // right flap full up
delay(500);
rightflapServo.write(10); // right flap full down



}else{
Serial.println(tag); //read out any unknown tag
}

}

void lightLED(int pin){
///////////////////////////////////
//Turn on LED on pin "pin" for 250ms
///////////////////////////////////
Serial.println(pin);

digitalWrite(pin, HIGH);
delay(250);
digitalWrite(pin, LOW);
}

void resetReader(){
///////////////////////////////////
//Reset the RFID reader to read again.
///////////////////////////////////
digitalWrite(RFIDResetPin, LOW);
digitalWrite(RFIDResetPin, HIGH);
delay(150);
}

void clearTag(char one[]){
///////////////////////////////////
//clear the char array by filling with null - ASCII 0
//Will think same tag has been read otherwise
///////////////////////////////////
for(int i = 0; i < strlen(one); i++){
one[i] = 0;
}
}

boolean compareTag(char one[], char two[]){
///////////////////////////////////
//compare two value to see if same,
//strcmp not working 100% so we do this
///////////////////////////////////

if(strlen(one) == 0) return false; //empty

for(int i = 0; i < 12; i++){
if(one[i] != two[i]) return false;
}

return true; //no mismatches
}
```

Here's the code for the WaveShield, courtesy of Adafruit-

```
#include <FatReader.h>
#include <SdReader.h>
#include <avr/pgmspace.h>
#include "WaveUtil.h"
#include "WaveHC.h"


SdReader card; // This object holds the information for the card
FatVolume vol; // This holds the information for the partition on the card
FatReader root; // This holds the information for the filesystem on the card
FatReader f; // This holds the information for the file we're play

WaveHC wave; // This is the only wave (audio) object, since we will only play one at a time

#define DEBOUNCE 100 // button debouncer

// this handy function will return the number of bytes currently free in RAM, great for debugging!
int freeRam(void)
{
extern int __bss_end;
extern int *__brkval;
int free_memory;
if((int)__brkval == 0) {
free_memory = ((int)&free_memory) - ((int)&_bss_end);
}
else {
free_memory = ((int)&ree_memory) - ((int)__brkval);
}
return free_memory;
}

void sdErrorCheck(void)
{
if (!card.errorCode()) return;
putstring("\n\rSD I/O error: ");
Serial.print(card.errorCode(), HEX);
putstring(", ");
Serial.println(card.errorData(), HEX);
while(1);
}

void setup() {
// set up serial port
Serial.begin(9600);
putstring_nl("WaveHC with 6 buttons");

putstring("Free RAM: "); // This can help with debugging, running out of RAM is bad
Serial.println(freeRam()); // if this is under 150 bytes it may spell trouble!

// Set the output pins for the DAC control. This pins are defined in the library
pinMode(2, OUTPUT);
pinMode(3, OUTPUT);
pinMode(4, OUTPUT);
pinMode(5, OUTPUT);
```

```
// pin13 LED
pinMode(13, OUTPUT);

// enable pull-up resistors on switch pins (analog inputs)
digitalWrite(14, HIGH);
digitalWrite(15, HIGH);
digitalWrite(16, HIGH);
digitalWrite(17, HIGH);
digitalWrite(18, HIGH);
digitalWrite(19, HIGH);

// if (!card.init(true)) { //play with 4 MHz spi if 8MHz isn't working for you
if (!card.init()) { //play with 8 MHz spi (default faster!)
putstring_nl("Card init. failed!"); // Something went wrong, lets print out why
sdErrorCheck();
while(1); // then 'halt' - do nothing!
}

// enable optimize read - some cards may timeout. Disable if you're having problems
card.partialBlockRead(true);

// Now we will look for a FAT partition!
uint8_t part;
for (part = 0; part < 5; part++) { // we have up to 5 slots to look in
if (vol.init(card, part))
break; // we found one, lets bail
}
if (part == 5) { // if we ended up not finding one :(
putstring_nl("No valid FAT partition!");
sdErrorCheck(); // Something went wrong, lets print out why
while(1); // then 'halt' - do nothing!
}

// Lets tell the user about what we found
putstring("Using partition ");
Serial.print(part, DEC);
putstring(", type is FAT");
Serial.println(vol.fatType(),DEC); // FAT16 or FAT32?

// Try to open the root directory
if (!root.openRoot(vol)) {
putstring_nl("Can't open root dir!"); // Something went wrong,
while(1); // then 'halt' - do nothing!
}

// Whew! We got past the tough parts.
putstring_nl("Ready!");
}

void loop() {
//putstring("."); // uncomment this to see if the loop isnt running
switch (check_switches()) {
case 1:
playcomplete("SOUND1.WAV");
break;
case 2:
playcomplete("SOUND2.WAV");
break;
case 3:
playcomplete("SOUND3.WAV");
break;
case 4:
playcomplete("SOUND4.WAV");
break;
case 5:
playcomplete("SOUND5.WAV");
break;
case 6:
playcomplete("SOUND6.WAV");
}
}

byte check_switches()
{
static byte previous[6];
static long time[6];
byte reading;
byte pressed;
byte index;
pressed = 0;

for (byte index = 0; index < 6; ++index) {
reading = digitalRead(14 + index);
if (reading == LOW && previous[index] == HIGH && millis() - time[index] > DEBOUNCE)
{
// switch pressed
time[index] = millis();
pressed = index + 1;
break;
}
previous[index] = reading;
}
// return switch number (1 - 6)
return (pressed);
}


// Plays a full file from beginning to end with no pause.
void playcomplete(char *name) {
// call our helper to find and play this name
playfile(name);
while (wave.isplaying) {
// do nothing while its playing
}
// now its done playing
}

void playfile(char *name) {
// see if the wave object is currently doing something
if (wave.isplaying) {// already playing something, so stop it!
wave.stop(); // stop it
}
// look in the root directory and open the file
if (!f.open(root, name)) {
putstring("Couldn't open file "); Serial.print(name); return;
}
// OK read the file and turn it into a wave object
if (!wave.create(f)) {
putstring_nl("Not a valid WAV"); return;
}

// ok time to play! start playback
wave.play();
}
```



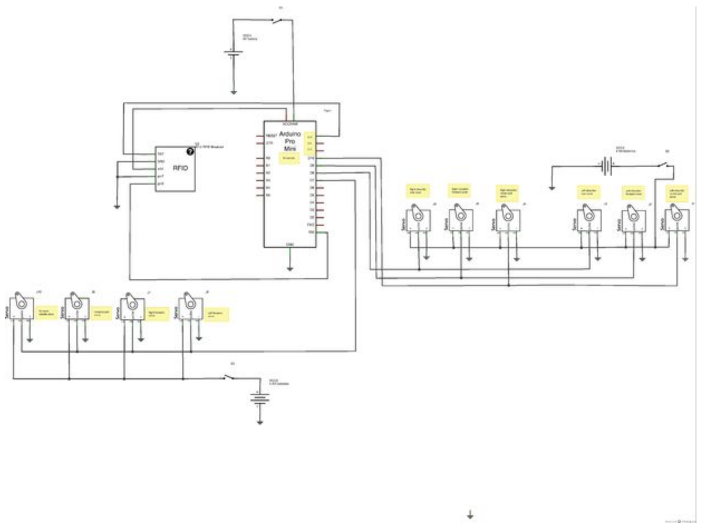
Step 12: Electronics- right side schematics and code

Here's the schematic and code for the right side.

It's pretty similar to the left side, minus the XBee radio. The servos for the forearm rocket all receive the same signal- one of the servos that opens the side cover will need to be reversed rotation. Two of the servos that open the forward and rearward shoulder rocket covers will also need to have their rotation reversed as they receive the same signal as the servos on the opposite shoulder.

Here's the code-

```
#include "Servo.h" // include the servo library

Servo forearmServo; // servos to move forearm missile
Servo rearcoverServo; // servo to move rear shoulder rocket pod cover
Servo forwardcoverServo; // servo to move forward shoulder rocket pod cover
Servo podServo; // servo to move shoulder rocket pod



int RFIDResetPin = 13;
int servoPin1 = 7; // control pin for forearm missile servos
int servoPin2 = 8; // control pin for rear shoulder rocket pod cover servo
int servoPin3 = 9; // control pin for forward rocket pod cover servo
int servoPin4 = 10; // control pin for shoulder rocket pod servo



//Register your RFID tags here
char tag1[13] = "440085E77452";
char tag2[13] = "440085FC330E";
char tag3[13] = "440085F97840";
char tag4[13] = "4400863914EF";


void setup(){
Serial.begin(9600);


forearmServo.attach(servoPin1); // attaches the servo on pin 7 to the servo object
rearcoverServo.attach(servoPin2); // attaches the servo on pin 8 to the servo object
forwardcoverServo.attach(servoPin3); // attaches the servo on pin 9 to the servo object
podServo.attach(servoPin4); // attaches the servo on pin 10 to the servo object
forearmServo.write(45); // rotate the forearm servos to 45 degrees
rearcoverServo.write(45); // rotate the rear cover servo to 45 degrees
forwardcoverServo.write(45); // rotate the forward cover servo to 45 degrees
podServo.write(45); // rotate the left flap servo to 45 degrees



pinMode(RFIDResetPin, OUTPUT);
digitalWrite(RFIDResetPin, HIGH);



}

void loop(){

char tagString[13];
int index = 0;
boolean reading = false;

while(Serial.available()){

int readByte = Serial.read(); //read next available byte

if(readByte == 2) reading = true; //begining of tag
if(readByte == 3) reading = false; //end of tag

if(reading && readByte != 2 && readByte != 10 && readByte != 13){
//store the tag
tagString[index] = readByte;
index ++;
}
}

checkTag(tagString); //Check if it is a match
clearTag(tagString); //Clear the char of all value
resetReader(); //reset the RFID reader
}

void checkTag(char tag[]){
///////////////////////////////////
//Check the read tag against known tags
///////////////////////////////////

if(strlen(tag) == 0) return; //empty, no need to contunue
if(compareTag(tag, tag3)){ // if matched tag3, do this
forearmServo.write(135);
delay(2500);
forearmServo.write(45);

}else if(compareTag(tag, tag4)){ //if matched tag4, do this
rearcoverServo.write(70); // rotate the pod servo to 90 degrees
delay(500); // wait half a second
forwardcoverServo.write(100); // rotate the forward cover servo to 110 degrees
delay(500);
podServo.write(80); // rotate the pod servo to 80 degrees
delay(4000);
podServo.write(45); // rotate the pod servo to 45 degrees
delay(500);
forwardcoverServo.write(45); // rotate the forward coverservo to 90 degrees
delay(500);
rearcoverServo.write(45); // rotate the pod servo to 135 degrees


}else{
Serial.println(tag); //read out any unknown tag
}

}

void lightLED(int pin){
///////////////////////////////////
//Turn on LED on pin "pin" for 250ms
///////////////////////////////////
Serial.println(pin);

digitalWrite(pin, HIGH);
delay(250);
digitalWrite(pin, LOW);
}

void resetReader(){
///////////////////////////////////
//Reset the RFID reader to read again.
///////////////////////////////////
digitalWrite(RFIDResetPin, LOW);
digitalWrite(RFIDResetPin, HIGH);
delay(150);
}

void clearTag(char one[]){
///////////////////////////////////
//clear the char array by filling with null - ASCII 0
//Will think same tag has been read otherwise
///////////////////////////////////
for(int i = 0; i < strlen(one); i++){
one[i] = 0;
}
}

boolean compareTag(char one[], char two[]){
///////////////////////////////////
//compare two value to see if same,
//strcmp not working 100% so we do this
///////////////////////////////////

if(strlen(one) == 0) return false; //empty

for(int i = 0; i < 12; i++){
if(one[i] != two[i]) return false;
}

return true; //no mismatches
}
```



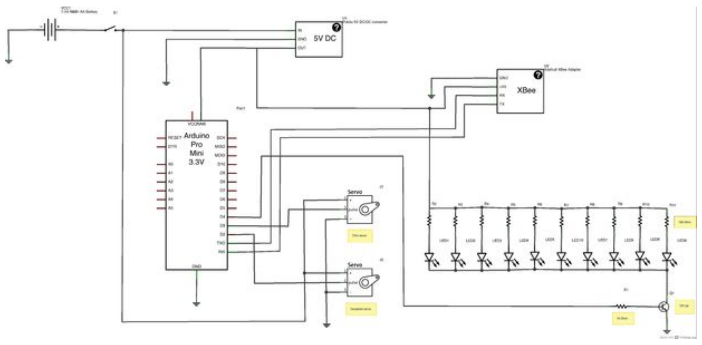
Step 13: Electronics- wireless helmet schematic and code

Here's the schematic and code for the wireless helmet.

Since there's very little room in the helmet the wireless system was powered by a single 7.4V NiMH battery pack. The digital servos used in the helmet are designed to be operated on 7.4V so a 5V DC/DC converter is used to provide power for the Arduino, XBee and LEDs.

Here's the code-

```
#include "Servo.h" // include the servo library

Servo faceplateServo;
Servo chinServo;

int ledPin1 = 4; // control pin for LED eyes
int servoPin1 = 2; // control pin for face plate servo
int servoPin2 = 3; // control pin for chin

void setup() {

faceplateServo.attach(servoPin1); // attaches the servo on pin 2 to the servo object
chinServo.attach(servoPin2); // attaches the servo on pin 3 to the servo object
faceplateServo.write(30); // rotate face plate servo to 30 degrees
chinServo.write(95); // rotate chin servo to 95 degrees
pinMode(ledPin1, OUTPUT); // sets the LED pin as output
digitalWrite(ledPin1, HIGH); // turn on LED eyes

Serial.begin(9600);
}

void loop() {


// look for a capital A over the serial port and turn off LED
if (Serial.available() > 0) {
if (Serial.read() == 'A') {
digitalWrite(ledPin1, LOW); // turn off LED eyes
delay(500); // wait half a second
faceplateServo.write(95); // rotate the face plate servo to 95 degrees
chinServo.write(20); // rotate the chin servo to 20 degrees
delay(4000); // wait 4 seconds
chinServo.write(95); // rotate the chin servo to 95 degrees
faceplateServo.write(30); // rotate the face plate servo to 30 degrees
digitalWrite(ledPin1, HIGH); // turn on LED eyes

}
}
}
```



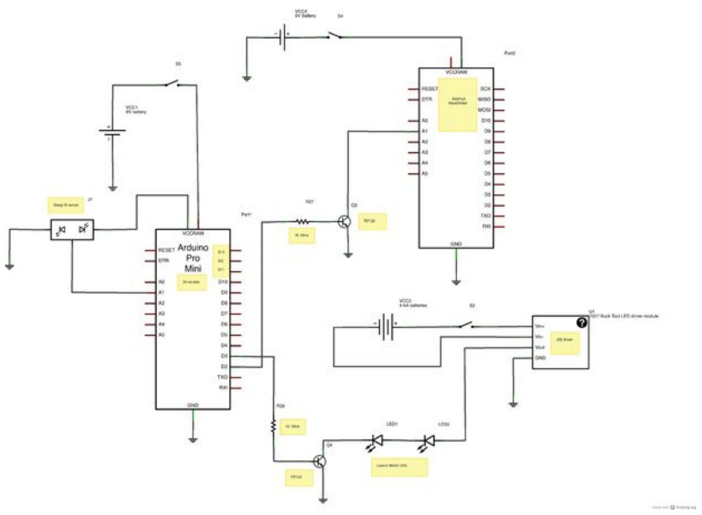
Step 14: Electronics- schematic and code for the boots

Here's the schematic and code for the boots.

This is a pretty simple circuit. The Sharp IR sensor inputs a value into the Arduino which triggers the Luxeon boot lights and the WaveShield to play an audio file.

Here's the code-

```
// these constants won't change:

int triggerSensor = 1; // the sensor is connected to analog pin 1
int threshold = 750; // threshold value to decide when the sensor input triggers
int ledPin = 3; // control pin for LED
int soundPin = 2; // control pin for sound board

// these variables will change:
int sensorReading = 0; // variable to store the value read from the sensor pin


void setup() {
Serial.begin(9600); // use the serial port
pinMode(ledPin, OUTPUT); // sets the LED pin as an output
pinMode(soundPin, OUTPUT); // sets the sound pin as output
digitalWrite(ledPin, LOW); // turn off LED
digitalWrite(soundPin, LOW); // turn the sound off

}

void loop() {

// read the sensor and store it in the variable sensorReading:
int val = analogRead(triggerSensor);

// if the sensor reading is greater than the threshold:
if (val >= threshold) {

Serial.println(val);
digitalWrite(soundPin, HIGH); // turn the sound on
delay(10); // wait ten milliseconds
digitalWrite(soundPin, LOW); // turn the sound off
digitalWrite(ledPin, HIGH); // turn the LED on
delay(2400); // wait two seconds
digitalWrite(ledPin, LOW); // turn the LED off
}


}
```

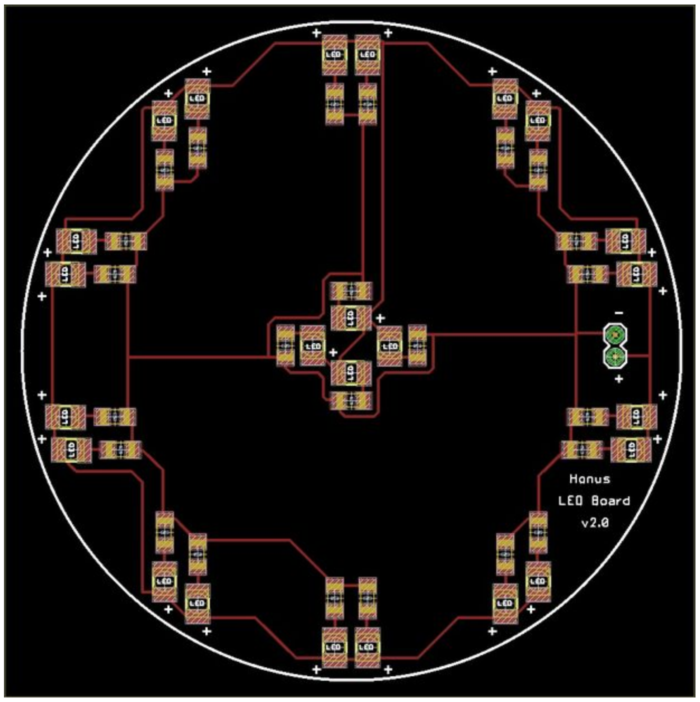


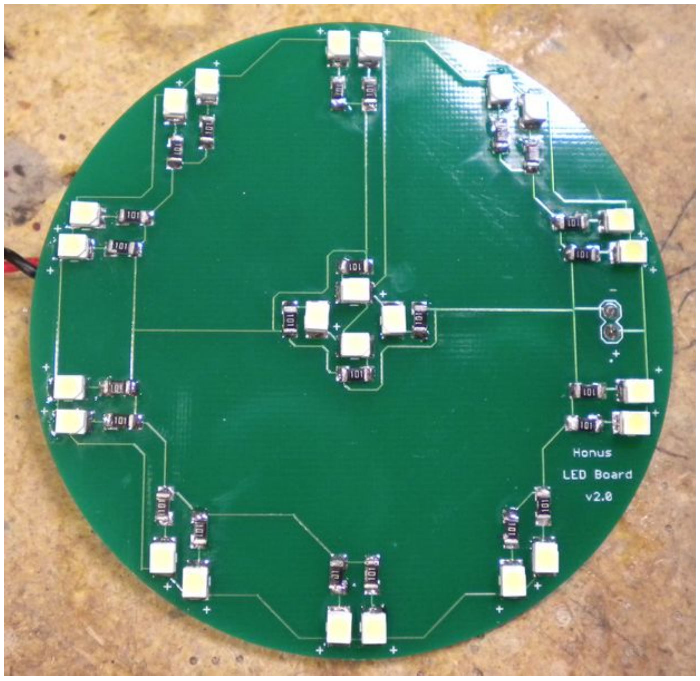
Step 15: Chest and repulsor lights

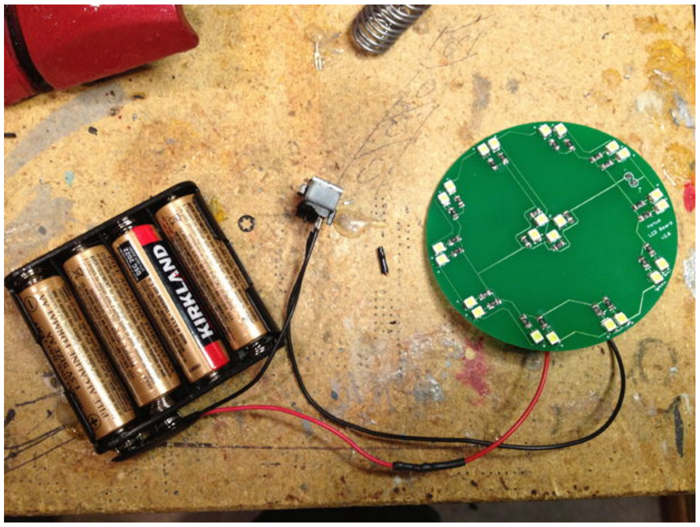
What Iron Man suit would be complete without lights?

For the chest light I created a simple PCB in EAGLE and soldered on the surface mount resistors and LEDs. The finished PCB was wired up and mounted behind the translucent chest piece. With the hands we lucked out and didn't need to build the lighting system from scratch as Greg already had a repulsor light and sound system that was donated by a friend.

You could easily add repulsor lights to this system and trigger them with additional RFID tags by modifying the code. For an idea have a look here.







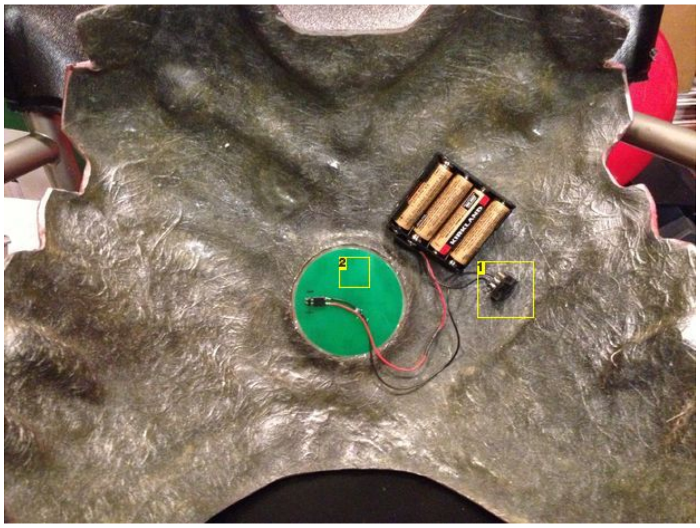


Image Notes
1. Power switch
2. PCB is hot glued in place

File Downloads

LEDBoardv2.0.sch (23 KB)
[NOTE: When saving, if you see .tmp as the file ext, rename it to 'LEDBoardv2.0.sch']

LEDBoardv2.0.brd (19 KB)
[NOTE: When saving, if you see .tmp as the file ext, rename it to 'LEDBoardv2.0.brd']

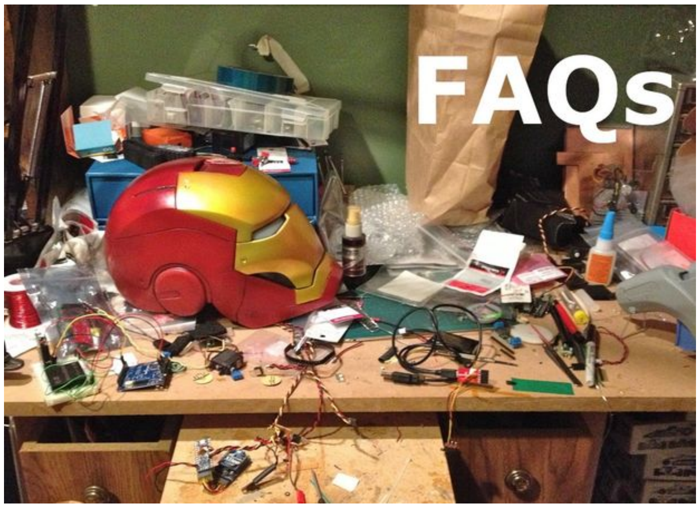
Step 16: Frequently asked questions

Q: I want to buy Iron Man armor. Do you sell Iron Man armor?
A: Sorry I do not produce nor do I sell Iron Man armor or helmets. Your best bet is to check out the Junkyard on The Replica Prop Forum.

Q: Can I send my Iron Man costume to you to have animatronics installed?
A: Sorry, but I cannot take on any more Iron Man projects at this point in time.

Q: What is the difference between analog and digital servos?
A: Compared to analog servos, digital servos have a different type of control circuitry that allows for higher resolution (more accurate positioning), increased holding power and the ability to adjust control parameters such as direction of rotation, speed and deadband width. The downside is that digital servos can be noisier (often heard as a high pitch hum) and they consume more power than analog servos.

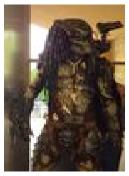
Q: Why did you use RFID sensors in the gloves?
A: I chose RFID because I basically needed a non contact switch to trigger the suit functions. The way that Iron Man gloves are constructed makes it very difficult to use a traditional push button switch. Plus with RFID you can add more tags to the glove fingers later on for more functions without adding additional control wires- something that you would have to do with a traditional switch.

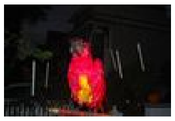


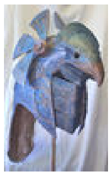
Related Instructables

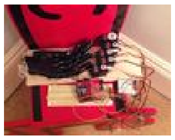
Arduino animatronics- make your awesome costumes more awesome! by Honus

How to create simple animatronics- part one: using the MAKE controller by Honus


Building a replica Predator costume by Honus


Animatronic Fawkes the Phoenix by makermike


Animatronic Stargate helmet by Honus


Arduino Wireless Animatronic Hand by leadzeplin

Comments

1 comments Add Comment

matson23 says: Apr 28, 2014. 9:34 AM REPLY

Wow! I'm speechless! Awesome isn't strong enough a word ! Superb work!