THIS IS THE DROID YOU'RE LOOKING FOR

HOW I BUILT AN ASTROMECH DROID

MAKERDAN (DAN MASSEY)

VERSION 1.0 (DEC.2019)

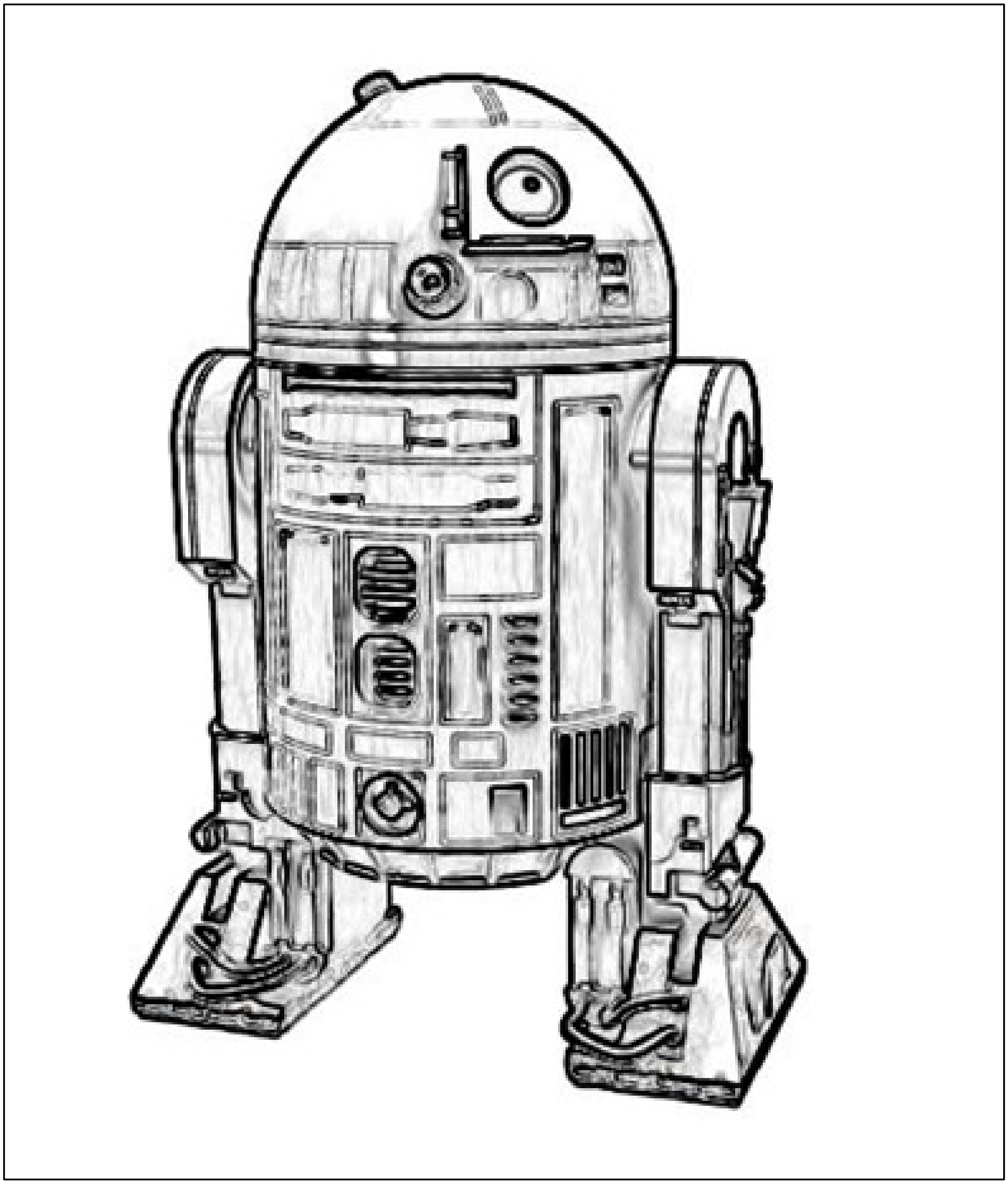


TABLE OF CONTENTS

This Handbook is a work in progress. As I complete my build I will be constantly updating it with new information.

BUILDING A STAR WARS DROID USING THE PADWAN 360 CONTROL SYSTEM

Introduction:

This isn't a handbook on how to build an Astromech Droid. This a handbook on how I built my own personal droid. There are hundreds of ways to go about making an Droid and everyone should find the way that best suits their talents, interests and abilities. Part A of this guide would be helpful for beginners to learn what to gather in terms of supplies and how to put the electronics together in order to get a functioning droid. After that, this handbook goes into details about how I ended up building my own droid. I would encourage anyone interested in pursuing this to spend time reading through many build logs on the Astromech site and taking notes along the way. I created a binder with tabs (Dome, Body, Skins, Electronics….) to help me organize all of the information I learned from various builds. You will find that in every build log you read, you will learn something new.

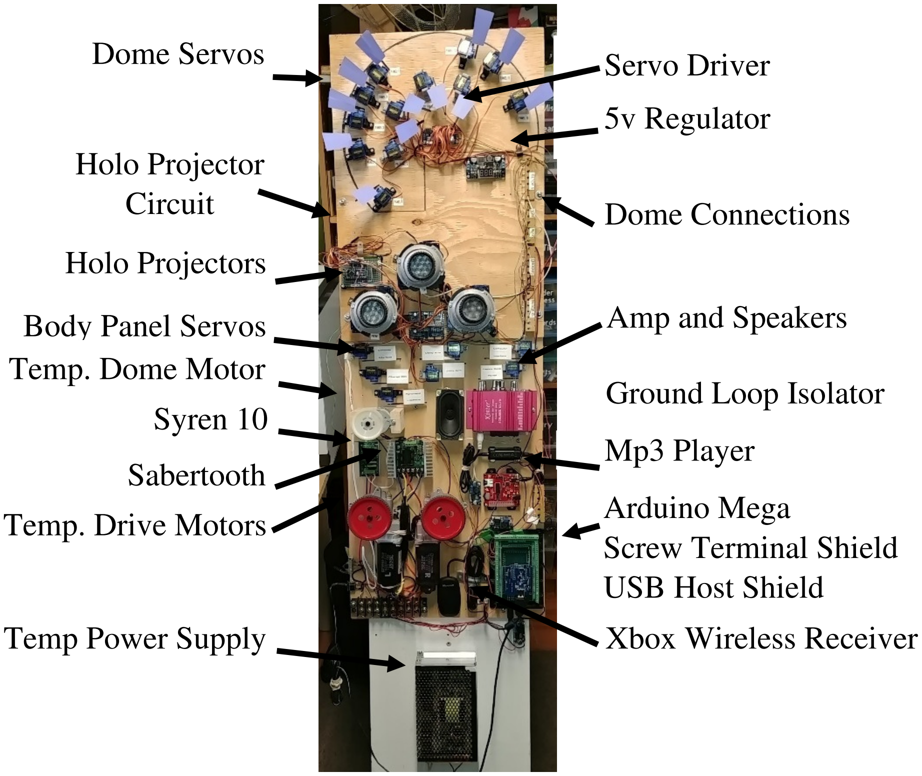
Thank You: I certainly had a lot of help from my fellow Atromech builders and YouTubers along the way. Every now and then in my handbook you will see a shout out to people using this symbol. Right now, I want to formally thank everyone that has given me advice, taught me some very useful lessons and encouraged me during my build. I would especially like to thank Maxstang for his encouragement and very helpful advice.

Thank You!
Maxstang

Order of Build: You can start your build any number of ways. Some plan on starting with the dome and others, the body section. I decided to start with the electronics and programming and build a working prototype of these mechanisms before starting to build my droid. I created a large wooden easel that I could screw most of the electronic parts to while I tested them out. I called this my "Proof-of-Concept-board". Here is a photo of it:

Here is a link to a video that explains it in more detail:

https://youtu.be/prtZo56cxwo



Control System: There are several different ways to control your droid. I decided to use an Xbox 360 controller that sends signals to an Arduino Mega which in turn sends commands to the droid using an Arduino sketch provided by Dan Kraus.

Dan Kraus

Here is a shout out to Dan Kraus for creating and sharing the code for the Xbox system. Not only did he make this available for everyone but also spends time maintaining and updating it at his Github Repository found here:

https://github.com/dankraus/padawan360

PART A: Basic setup (Controller, Software, Sound, Motor Drivers for Feet and Dome)

Step 1: Gathering and building the Basic Parts for the Electrical System

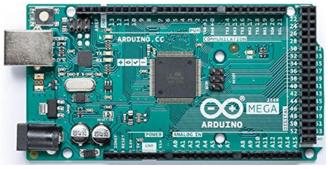
There can be some variation in the parts needed to use this system but if you stick with what is described below it will be much easier to make the control setup a "plug and play" type system. You do not need to gather all of the parts listed at the same time. It is advisable to put together the system in stages and test each stage as you complete it.

Basic Necessary Parts: (see next page for photos)

- Arduino Mega
- USB Host Shield (used to connect an Xbox receiver to the system)
- Wireless Xbox controller
- Wireless Xbox Receiver
- Sparkfun MP3 Trigger Board
- Sabertooth Motor Controller (2x25 or 2x32) (made by Dimension Engineering)
- SyRen 10 Motor Controller (made by Dimension Engineering)
- Amplifier and Speakers (any size will do based on preference)
- DC-DC Buck Converter (converts a higher voltage to a lower voltage).
    - You may need more than one. Some of the parts above can only be powered from a certain voltage such as 5 volts or 12 volts. A Buck Converter will drop the incoming voltage down to the required amount for the part you are connecting it to. Some converters come with a pre-set output voltage but the one in the diagram can be adjusted to any voltage you need.
- Fuse Panel
    - The type of fuse panel you choose will depend on how many accessories your droid will have. Many club members have chosen the type in the diagram made by Blue Sea. (see part (TBA) for the one I used)
- Main Power Switch (to turn on/off all of the power to the Droid)
- Wire
    - You will need a lot of wire of varying gauges. Also, a variety of colors will help. Typically, you want to use 12-14-gauge wire for items that require a lot of power such as the foot drives. You can get away with 16 gauge and smaller for items that don't require as much power (the higher the gauge the thinner the wire is).

- Power Supply
    - Eventually you will be using some sort of battery setup to power your droid (more on that in the "Power" section) but for now you may want to use a wall adaptor just for test purposes. I used a 12-volt 20-amp supply that I had left over from a previous build but if your drive motors will only be turning to see if they work and not actually driving your droid you can get away with something with far less amps.
- Slip Ring and Adaptors
    - A Slip Ring connects the Body electronics to the Dome electronics. It allows wires to go from body to dome and allows the dome to spin 360 degrees without tangling the wires. You will need it as well as the adaptors when it is time to connect the dome electronics to the body. More information on this can be found on page 13.

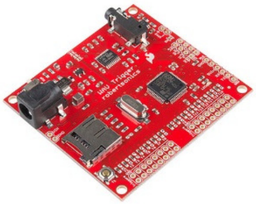
Photos:

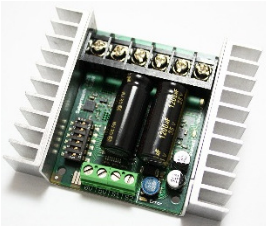

Arduino Mega

USB Host Shield

Xbox 360 Wireless Controller

Xbox 360 Wireless Receiver


Sparkfun MP3 Trigger Board


Sabertooth Motor Driver

SyRen 10 Motor Controller

DC-DC Buck Converter

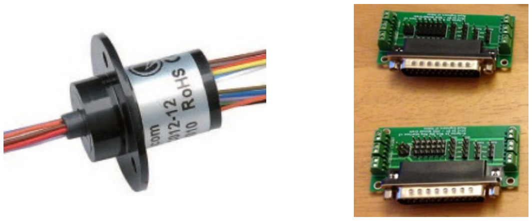
Amplifier and Speakers

Fuse Panel

Power Switch

Wire

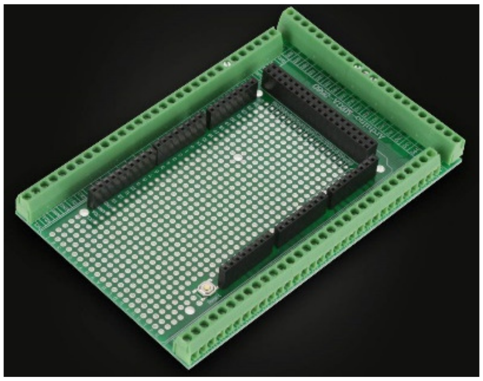
12v 5amp Power Supply

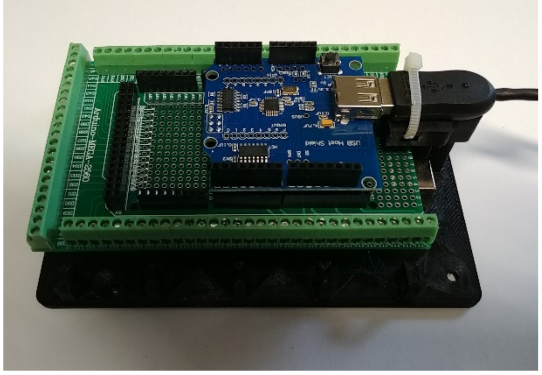

Slip Ring and Adaptor Boards

Optional Parts:

- Ground Loop Isolator (used to reduce buzzing from the speakers) Place this between your amplifier and the MP3 Trigger board.

- Screw Terminal Block Shield Board Kit (for Arduino Mega)





Screw Terminal Block Shield (for Arduino Mega) Explanation:

You will soon find out that while you are building your droid, there will be several connections needed that go from the Arduino Mega to various accessories such as the USB Host Shield, I2C connections, Tx and Rx connections, servos etc… There are 3 ways to make these connections to the Arduino Mega pins:

a. Push the wires directly into the pin headers on the Arduino board (not recommended as the wires can easily pop out during usage although some people secure them with hot glue).

b. Solder the connections directly to the underside of the Arduino at the pin locations (hard to solder small, clean joints and it is difficult to change connections if needed in the future).

c. Purchase the Screw Terminal Block Shield pictured above (recommended) and plug the Arduino directly into this shield from the bottom. This allows you to plug the USB Host Shield into the top of the board and provides you with screw terminal connections for all the pins on the Arduino. Also, if your Arduino Mega ever needs replacing you can easily pop if out and replace it with another one. In order to make this even easier, I have created a 3D printable mounting setup that holds all 3 of the components together and can be screwed to any platform (photo above right). It even has a support platform for the USB connector on the wireless Xbox receiver (zip tie it to the platform).

   Here is a link to the Thingiverse page for my platform: https://www.thingiverse.com/thing:3950484

Step 2: Testing the Wireless Connection and Pairing the Xbox Controller

This step assumes you know how to navigate around the Arduino environment. This means you know how to plug in an Arduino to your computer, open the Arduino IDE, download a sketch to the Arduino and download Arduino Libraries to the IDE. If you are unfamiliar with this, then spend some time watching tutorials on how complete these tasks.

a. Once you have the Arduino Mega, connect it to your computer and download the blink sketch found under the File > Examples > Basics tabs. If you are able to successfully upload this sketch and the small LED on the Mega board is blinking, then it means you are able to download sketches to the Arduino Mega successfully and you are ready to move on to the next step.

b. Next, we will do a test to make sure the USB host shield and the Xbox receiver are functioning correctly and that your wireless setup is working. Plug the USB Host Shield into the Arduino Mega. Make sure you line up the 6-pin ICSP header underneath as well. If your shield came unassembled, double check your soldering job to make sure there are no "cold" solder joints.

c. Plug the Xbox wireless receiver into the host shield (the light on the receiver should turn on)

d. Press the button on the receiver. The LED on the receiver should blink. Press the center Guide button on the controller. It will turn on the controller and will also blink. Press the little sync button located on the top edge of the controller. The two devices should sync up and the blinking pattern on the controller will change to a steady LED light. This indicates that they are paired. If you need further assistance, there is a diagram found here.

   https://support.xbox.com/en-US/xbox-on-windows/accessories/xbox-360-wireless-gaming-receiver-windows

e. Next, we will download the Xbox Wireless Library's example sketch to see if the Xbox controller is working correctly (actually sending the correct signals to the receiver).
   Go to Dan Kraus's Github Repository and download all of the Libraries he has provided there into your Arduino IDE. Check to make sure the libraries have been included in your Arduino IDE (Sketch > Include Library).

f. While you are at his Repository, save the Arduino Mega Body sketch as well as the Dome sketch to your computer for later use. (padawan360_body_mega_i2c_ino). His Github repository is located here:

   https://github.com/dankraus/padawan360

g. Next, exit out of the Arduino IDE and then open the program again. This makes sure your new libraries will be accessible. Open the Xbox Wireless Library's example sketch.
   - File > Examples > USB Host Shield > Xbox > Xbox Recv
   - Upload that to your Arduino Mega.
   - Look at the code in the sketch and you will see "Serial.begin". Take note of the number after this (9600, 115200 etc).
   - Go to Tools > Serial Monitor and set the baud rate to that same number in the small window near the bottom right of your serial monitor screen (__________baud) from the drop down.

- Make sure your controller is turned.
- If you are paired, you should see the button names on the serial monitor appear as you press the buttons on your Xbox controller. If you are able to do this, then your system is ready to move on to the next step.

h. Open the Padwan 360 Body sketch.

i. Verify the sketch to make sure it verifies without any errors. If there are errors it most likely means you have not included all of the libraries needed. Once it verifies correctly, upload it to the Arduino Mega. In order to test the system so far, move on to step 3.

Step 3: Making Sounds Using the MP3 Trigger Board

The MP3Trigger Board will hold all of the sounds your R2D2 will make. If you need a copy of the sounds, they can be found here: https://host-a.net/f/97459-padawanr2soundszip

These sounds need to be saved to a micro SD card and placed onto the board. Connect the Micro SD card to your computer and upload the files one by one in the numbered order. If you don't do this, you will not be able to control which sound files are triggered. The MP3Trigger Board will not automatically sort the files by name or number (only by the order you copied them to the SD card). You can also use a Windows program called Drivesort to help sort them very quickly and easily (I would highly recommend doing this). Here's a helpful video from builder Balders on how to load and sort the sounds with Drivesort.

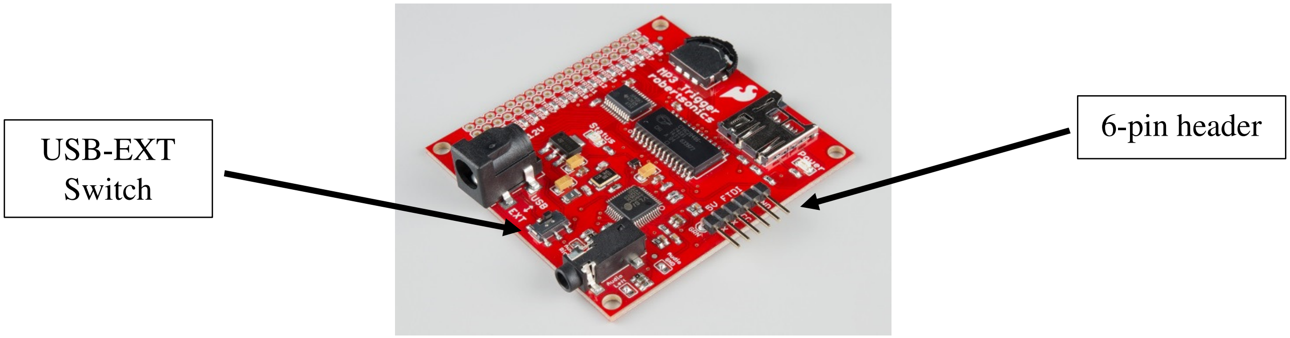
https://www.youtube.com/watch?v=UsMI2gW7Q40&feature=youtu.be

Balders

I would like to take this opportunity to thank Balders from Astromech for providing the information needed to use Drivesort as well as all of the many other contributions he has made to this club. You have been extremely helpful.

a. Once the files have been added to the SD card, place the card into the MP3 Trigger Board.

b. Your trigger board should have a 6-pin header soldered in place as shown in the diagram. If it doesn't, then you will have to solder one in. It can be either horizontal or vertical and is the only header you will need (you actually only need 3 pins).



c. The MP3 board can be powered 2 different ways. There is a small switch at the bottom of the board labeled USB - EXT, make sure that it is pointing to the USB side in order to power it via the Arduino (you would run a wire from the 6 pin header named 5v to the 5v pin on the Arduino board). It can also be powered by an external battery through the barrel jack. If you do that, you need to move the switch to the EXT position. Some users have experienced some issues of sounds freezing up and going a bit haywire when they used power from the Arduino (browning out). This was resolved by using the barrel jack power connector. The MP3 Trigger can be powered from 4.5v to 12v.

d. Add the wires from the Arduino to the MP3 Trigger Board's 6-pin header using the following chart.

MP3 Trigger	Arduino Mega
RX	Pin 1 (TX0)
USBVCC	5V
GND	GND

e. Connect an audio cable from the MP3 Board to your amplifier. Some people have added a ground loop isolator between the MP3 Board and the amplifier. You may want to do this if you experience interference sounds (humming or buzzing) from your speakers when the sounds are playing.

THE BIG TEST OF THE SYSTEM!

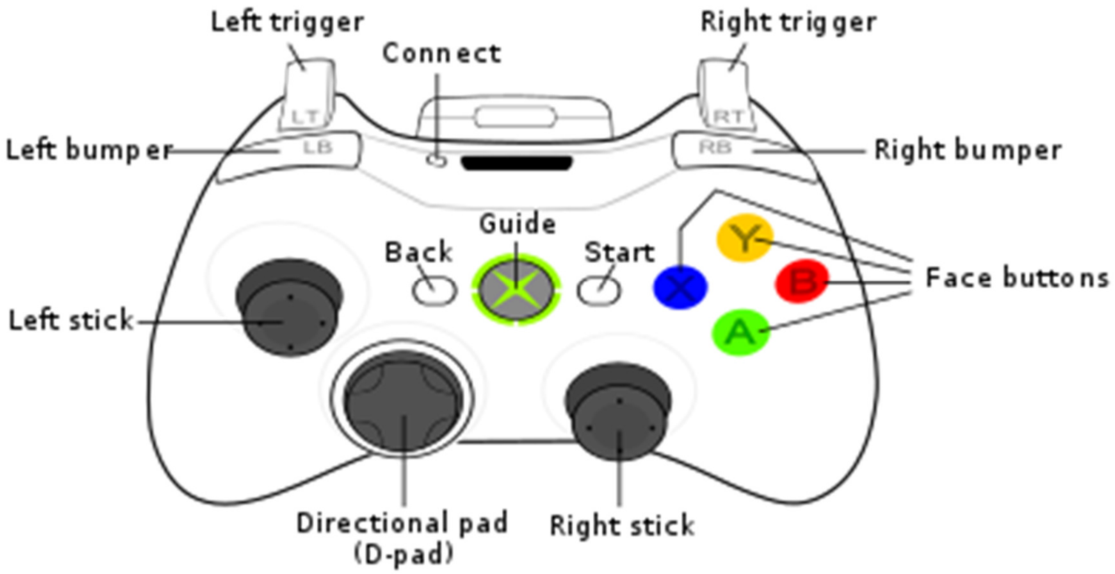
f. Start up the Xbox Controller and make sure you have power to your Arduino board as well as your amplifier. You should get a sound confirmation from the MP3 Trigger that confirms you are connected (sound clip #21). If the sound played for you then congratulation! You can move on otherwise check the "troubleshooting section below.

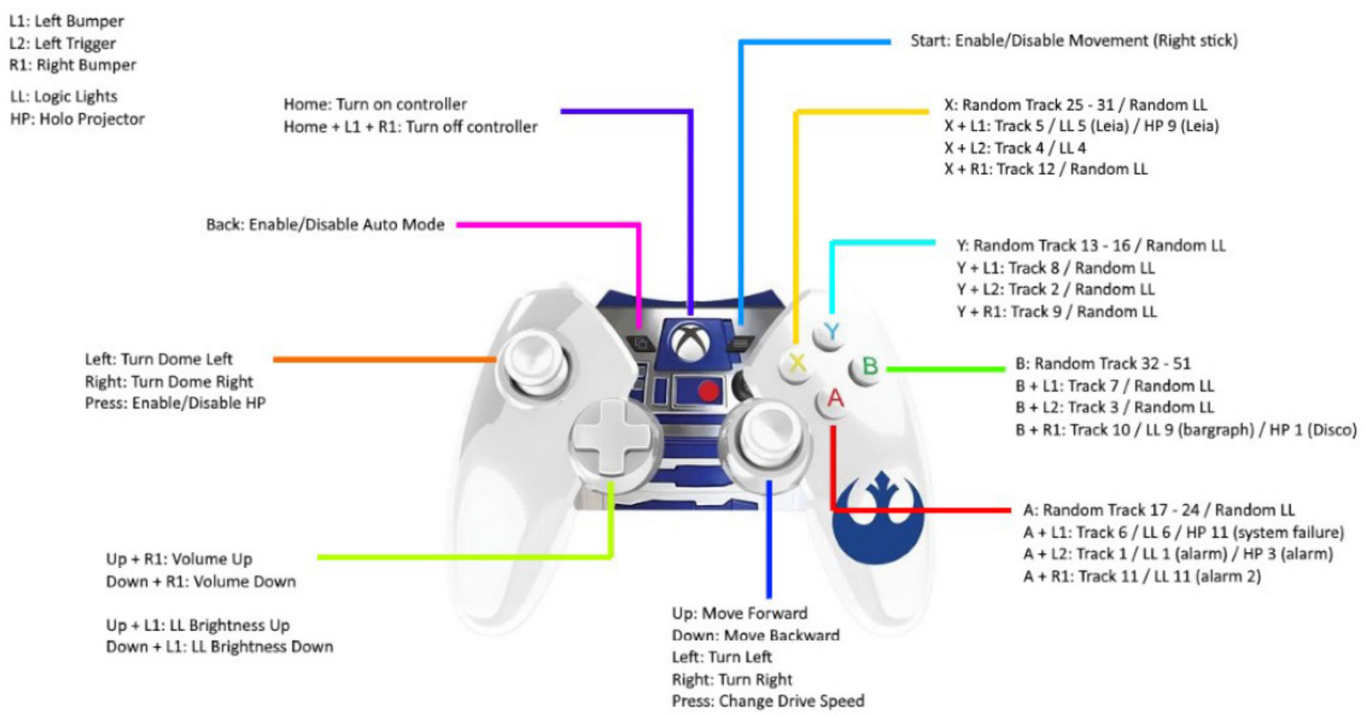
g. If you have gotten this far without any problems you can go ahead and try out some of the sounds by clicking buttons on the controller. Use the diagram on the next page to find out which buttons to press to get various sounds:

Troubleshooting:

- if you are experiencing any troubles with the MP3 Trigger Board, please reference the help provided on Dan's repository page. It may be that your MP3 Trigger board firmware is out of date.
- If it is "sort of working", try uploading the Padawan 360 Body sketch to the Arduino again while the MP3 Trigger Board is attached and powered up.

Turning Off the Controller: The code has a function built into it that allows you to turn off the controller without having to remove the batteries each time as long as the Arduino is still running the sketch. To do this, hold down the L1, R1 and center Guide button at the same time.





L1: Left Bumper
L2: Left Trigger
R1: Right Bumper

LL: Logic Lights
HP: Holo Projector

Start: Enable/Disable Movement (Right stick)

Home: Turn on controller
Home + L1 + R1: Turn off controller

X: Random Track 25 - 31 / Random LL
X + L1: Track 5 / LL 5 (Leia) / HP 9 (Leia)
X + L2: Track 4 / LL 4
X + R1: Track 12 / Random LL

Back: Enable/Disable Auto Mode

Y: Random Track 13 - 16 / Random LL
Y + L1: Track 8 / Random LL
Y + L2: Track 2 / Random LL
Y + R1: Track 9 / Random LL

Left: Turn Dome Left
Right: Turn Dome Right
Press: Enable/Disable HP

B: Random Track 32 - 51
B + L1: Track 7 / Random LL
B + L2: Track 3 / Random LL
B + R1: Track 10 / LL 9 (bargraph) / HP 1 (Disco)

A: Random Track 17 - 24 / Random LL
A + L1: Track 6 / LL 6 / HP 11 (system failure)
A + L2: Track 1 / LL 1 (alarm) / HP 3 (alarm)
A + R1: Track 11 / LL 11 (alarm 2)

Up + R1: Volume Up
Down + R1: Volume Down

Up + L1: LL Brightness Up
Down + L1: LL Brightness Down

Up: Move Forward
Down: Move Backward
Left: Turn Left
Right: Turn Right
Press: Change Drive Speed

Tracks

1 SCREAM2	11 Emperor	21 MISC17	31 OOH7	41 SENT10	51 SENT20
2 CHORTLE	12 Chorus	22 MISC25	32 SENT1	42 SENT11	52 HUM19
3 DOODOO	13 ALARM3	23 MISC30	33 SENT2	43 SENT12	53 HUM20
4 WOLFWSTL	14 ALARM5	24 MISC34	34 SENT3	44 SENT13	
5 LEIA	15 ALARM7	25 OOH1	35 SENT4	45 SENT14	
6 SHORTCKT	16 ALARM8	26 OOH2	36 SENT5	46 SENT15	
7 PATROL1	17 MISC3	27 OOH3	37 SENT6	47 SENT16	
8 ANNOYED	18 MISC7	28 OOH4	38 SENT7	48 SENT17	
9 Theme	19 MISC14	29 OOH5	39 SENT8	49 SENT18	
10 Cantina	20 MISC16	30 OOH6	40 SENT9	50 SENT19	

Notes

* Everything (including auto mode) is disabled if controller loses power or goes out of range.

* LED ring around the Home button indicates speed setting.

Larryj

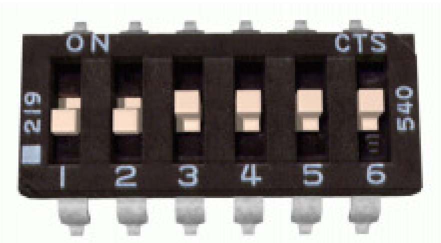
Thanks to LarryJ for the last diagram. It is very helpful. (note, in the new sketch Dan has given you have the ability to assign the left joystick to power the dome or the drive wheels)

Note: The button assignments in the diagram are not the same as the ones I am going to use. For more information on my button assignments see Appendix: A.

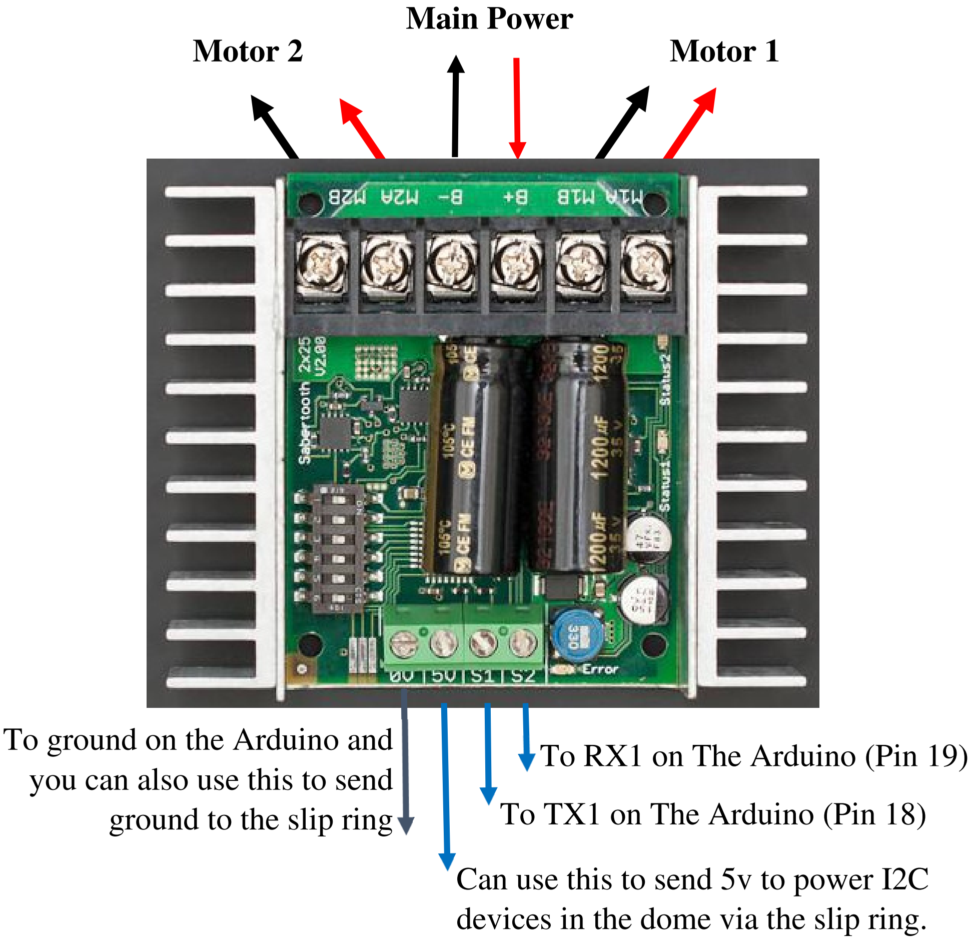
Step 4: Connect the Sabertooth motor driver:

Whether you have the 2x25 or the 2x32 motor drivers, the connections will be the same.

a. Set the dip switches on the Sabertooth. #1 and #2 off and the others all on.



b. Make the following connections to the motors, battery and Arduino: Make sure you are using the correct gauge wire for the main power and drive motors (see Part (TBA) for more information on this).



Main Power

Motor 2

Motor 1

To ground on the Arduino and you can also use this to send ground to the slip ring

To RX1 on The Arduino (Pin 19)

To TX1 on The Arduino (Pin 18)

Can use this to send 5v to power I2C devices in the dome via the slip ring.

c. Test your setup.

   a. Turn on the Xbox controller first and then the main power switch that sends power to the Arduino and all of the other components that make up the system including the Sabertooth. An LED on the Sabertooth should turn on confirming power.

   b. The MP3 Trigger should emit a confirmation melody.

   c. Press “Start” on the Xbox controller and you should get another confirmation sound.

   d. Try moving the joysticks to find out which one controls the motors (the motors should spin).

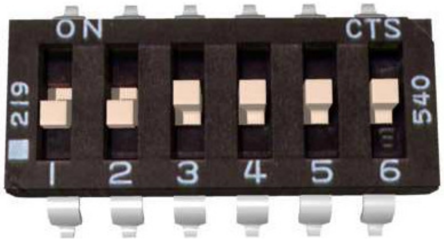
Troubleshooting:

- If one or more motors are spinning in the wrong direction you can flip the wires that go to M1A, M1B, M2A, M2B on the Sabertooth. Play around with it until the motors spin in the directions needed.
- More troubleshooting tips can be found at Dan’s repository

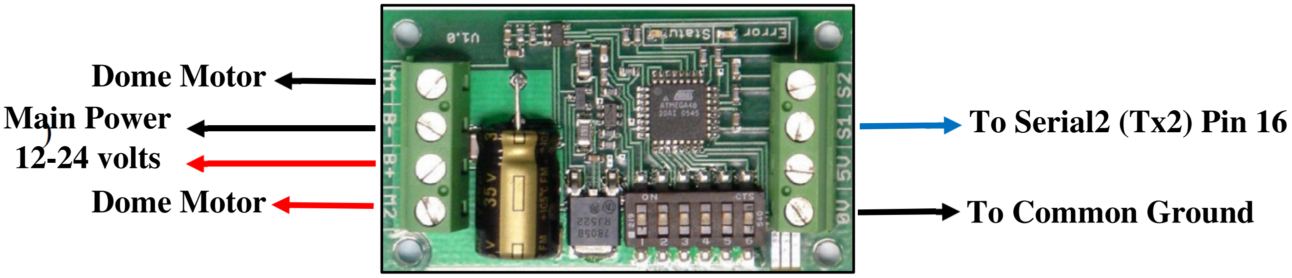
d. If your drive motors are working correctly, move on to the next step.

Step 5: Connect the Syren 10 Dome Motor Driver:

a. Set the dip switches on the Syren 10. #1 and #2 off and the others all on (for packetized serial mode).



b. Make the following connections to the motors, battery and Arduino: Remember to use the correct gauge wire to the dome motor and power connections.



Troubleshooting:

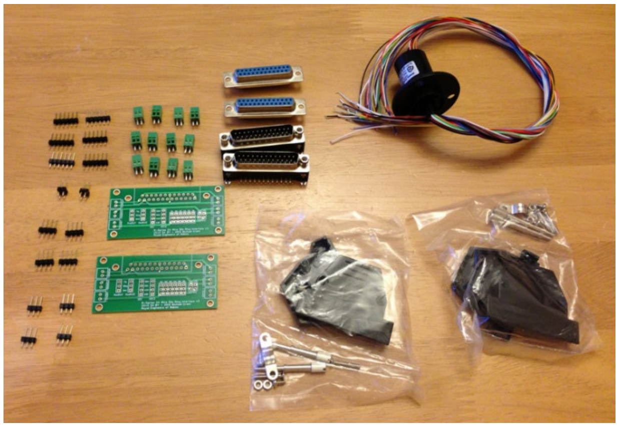
- If you find that the dome spins the opposite direction, flip M1 and M2. When standing behind the droid, moving left on the left analog stick should rotate the dome left.
- In some cases, we've noticed that the dome may behave erratically after starting up. If this is the case plugging a 10k resister between the S1 and 0V screw terminals. Simply bend the pins and screw them in along with the wires.
- If your dome motor doesn't spin at all then you probably have to change the baud rate in the code from 2400 to 9600 or another value that can be found in Dan's repository.
- More troubleshooting tips can be found at Dan's repository

Slip Rings: A Slip Ring connects the Body electronics to the Dome electronics. It allows wires to go from body to dome and allows the dome to spin 360 degrees without tangling the wires. It is mounted in the center base of the dome and looks like this:

Slip Ring Adaptors are used on either side of the Slip Ring to connect the slip ring wires to the top and bottom of the dome.

Thank You!
glynharper

Here is a setup available from glynharper at the Astromech site. Thanks glynharper!



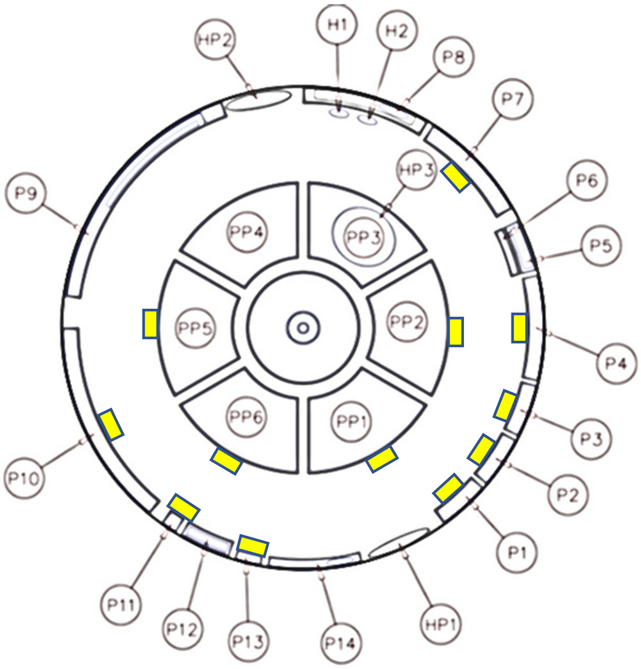
For my "Proof of Concept Board" I just ran wires to all of the parts and eliminated the slip ring for now. Once I start adding the electronics to my droid I will include the slip ring and adaptors. I will discuss this in more detail further on when I get to installing these devices but for now, I will move on to explaining how all of the servos will work.

This concludes Part A of this handbook. In the next section I will explain how I added lights and servos

PART B: Adding Servos

Dome Servos: There are many different types of servos that could operate the dome panels. I chose to use these common ones for testing purposes. I will probably go with some that have metal gears for the final version:

Not all of my dome panels will open via servos but 12 of them will. I want to leave one of the pie panels alone in case I want to add a periscope or life form scanner in the future (Pie Panel 4). Another pie panel will also house one of the Holo Projectors as well (Pie Panel 3). Here is a diagram of the panels that will open as well as numbers assigned to them.



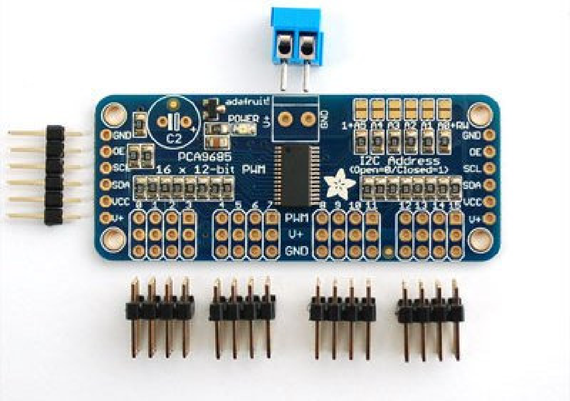
Servos Attached to: 🟨

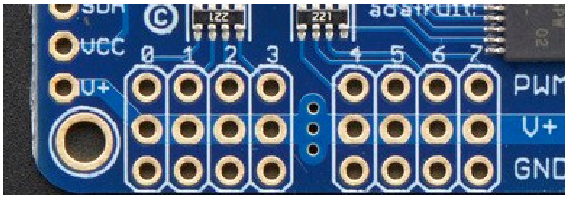
Pie Panel 1
Pie Panel 2
Pie Panel 5
Pie Panel 6

Dome Panel 1
Dome Panel 2
Dome Panel 3
Dome Panel 4
Dome Panel 7
Dome Panel 10
Dome Panel 11
Dome Panel 13

Step 1: Controlling the Dome Servos

I chose to control the Dome servos using an Adafruit 16 channel servo driver board which is fed I2C signals from the main code in the Arduino Mega. This may sound complicated, but it really isn't. This is what the servo board looks like unassembled:





Servo #1 is attached to position zero on the board
(0-15) = 16 servos

The Adafruit 16 channel servo driver board can control up to 16 servos using only 2 wires (SDA and SCL) plus the power and ground connections of course.

Connecting I2C lines to Arduinos can only be made in certain locations:

For an Arduino Mega:

SDA → Pin 20

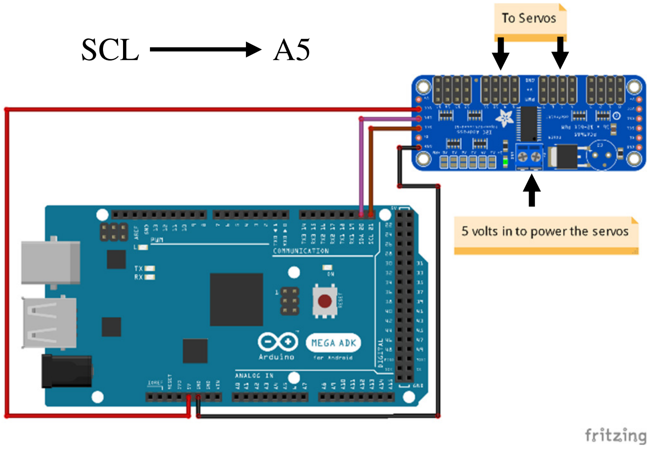
SCL → Pin 21

For an Arduino UNO or Nano

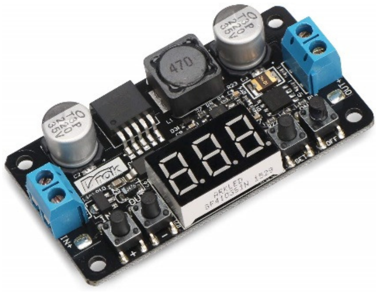
SDA → A4

SCL → A5

Here is a diagram that shows a typical connection to an Arduino Mega:

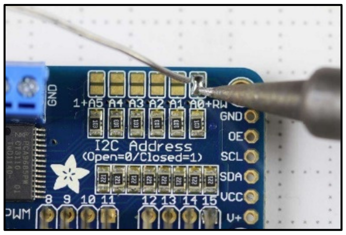


I will be running 4 wires through my Slip Ring for the I2C connection. 2 of these wires carry 12 volts and Ground and the other 2 carry the SDA and SCL lines from the Arduino Mega. In the Dome, I have a DC-DC Buck converter that steps down the voltage from the 12 volts to 5 volts to power the servo driver board as well as the servos (I do not have the 5 volts from the Mega going up through my Slip Ring as the photo above suggests). I made sure the converter could handle the power needed to drive all the servos and board at the same time. This is the converter I used in the dome.



DROK LM2596 Numerical Control Voltage Converter Board DC 5-32V 24v 32v to Adjustable 0-30V 12 v 5 v Switching Regulator Module 1.5A Volt Transformer with Red LED Voltage Tester

Note: Each I2C device must have a unique address that the code refers to in order to know which I2C device it is talking to. The address for the Servo Driver Board has already been assigned a default address of 0x40. If you use more than one of these boards in your Droid (which I am) they need to have their own unique addresses so I changed the address of this board to 0x41 because I will be using a similar board later that controls my Holo Projectors and it has an address of 0x40. The way I changed the address of the board to 0x41 is by creating a solder joint here directly on the board (bridge A0):



I had to add this address to the code otherwise the signals would not be sent to this particular board (see below for how to send the proper signals in order to drive the servos).

Step 2: Adding the Code to Control the Servos

The next step I took was to add the code to send the signals needed to open and close each servo or design a routine (function) for all the servos to move at the same time. It is important to know the limits of each of your servos (all servos are slightly different). If you send a signal to open or close a servo beyond its limits, you will end up damaging it and will also cause unwanted "stuttering" to happen. The best way to find each servo's limit is to use a good quality servo controller such as this one here: (I bought 2 of these in order to set up my Holo Projectors in the next section). When you plug a servo into this tester it will show you the max and min limits of the servo.

In the main code on the Mega, I added the following:

Before Setup:

- #include <Adafruit_PWMServoDriver.h> (I had to install this library to my Arduino IDE)
- #include <Wire.h> // This library should already be included in the 360 code
- Adafruit_PWMServoDriver pwm1 = Adafruit_PWMServoDriver(0x41); // for Dome Panel Servos

In Setup:

- Wire.begin(); // This should already be included in the 360 code

In the Loop:

- In order to make a servo move you give it the following command with 3 arguments:

  pwm.setPWM (the servo, 0, the position you want it to move to);

- For example: if you want servo #1 to move to position 300 in its range then write:

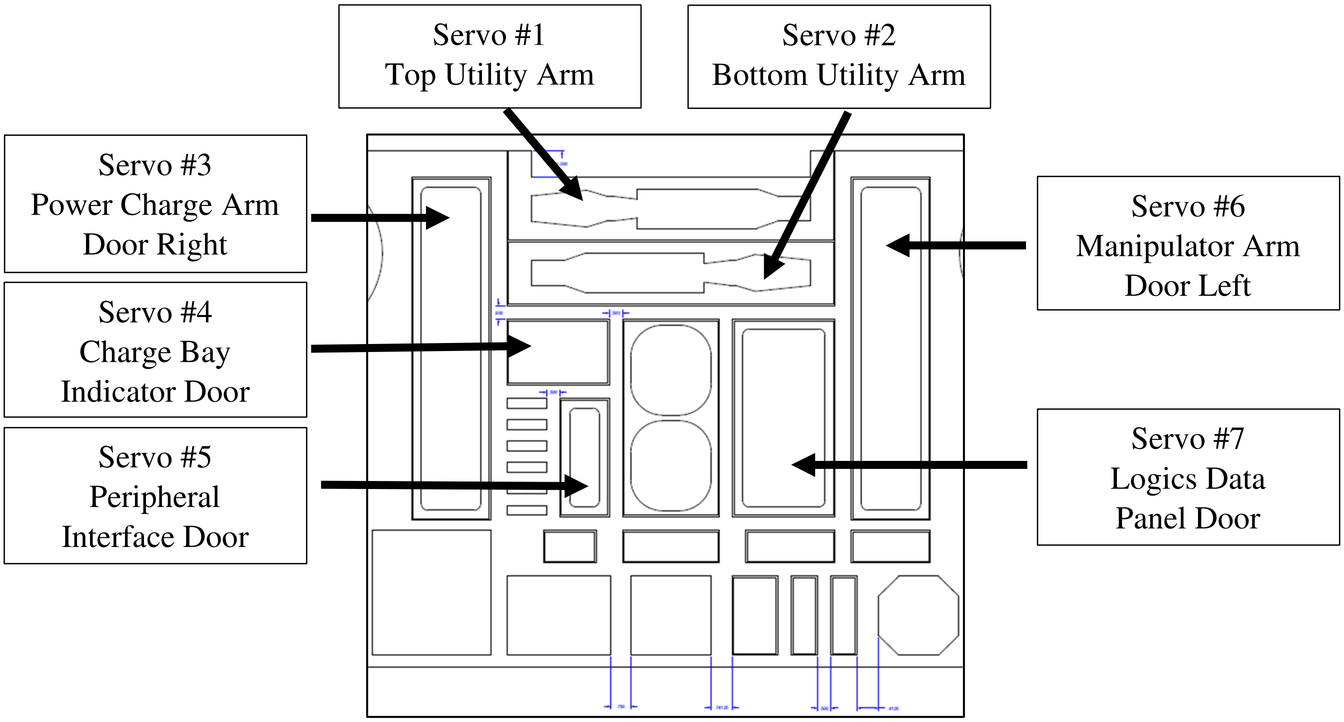
  pwm.setPWM (S1, 0, 300);

Here is an example code that uses a <u>new</u> Adafruit PWM Servo Driver Board (with and address of 0x41) **to drive a servo from its min to max settings:** (you can try this as a separate sketch on an Uno or Mega)

```
/*This sketch uses an Adafruit PWM Servo Driver board to open a servo to its Max setting and then close   it to its Minimum setting. It was created by Dan Massey on Nov. 17, 2019.*/

#include <Adafruit_PWMServoDriver.h> // install this library to your IDE

#include <Wire.h> //install this library to your IDE

Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver(0x40);

/* This sets the address of the servo driver board to 0x40.  If you were going to use a second servo driver board you would change this to:

  Adafruit_PWMServoDriver pwm1 = Adafruit_PWMServoDriver(0x41); and so on.

*/

void setup() {

 Wire.begin(); //Starts the wire library

}

void loop() {

 /*S1 stands for servo#1.  It is attached to Pie Panel #2 on the Droid.  S1=0 means that servo #1 is attached to position zero (0-15) on the servo driver board.  In order to make a servo move you give it the command:

   "pwm.setPWM (the servo, 0, the position you want it to move to);"

   For example: if you want servo #1 to move to its maximum setting then write:

   "pwm.setPWM (S1, 0, S1Max);"

 */

int delayTime = (1000);  //Delay time

uint8_t S1=0; //Declare Servo#1 on position zero of the servo board

int S1Min = 100; //Declare the Minimum travel distance for servo #1 (yours will be different based on your servo)

int S1Max = 1800; //Declare the Maximum travel distance for servo #1 (yours will be different based on your servo)

pwm.begin(); //Start up the servo board

pwm.setPWMFreq(60);  // Analog servos run at ~60 Hz updates

pwm.setPWM(S1, 0, S1Max); //Opens servo #1 to its Max setting

    delay (delayTime);

pwm.setPWM(S1, 0, S1Min); //Closes servo #1 to its Min setting

    delay (delayTime);


pwm.setPWM(S1, 0, 0); //Detaches servo #1 so there is no power going to it

}
```

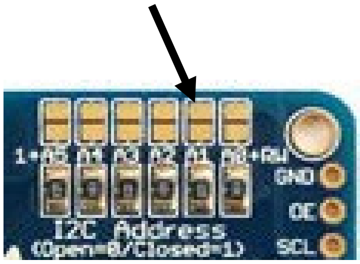
Body Panel Servos: I decided to have 5 of the body panels open and close as well as the 2 utility arms and control them using the same type of Adafruit 16 Channel Servo Driver board as the one in the dome. For test purposes, I used the same types of servos:

As with my dome panels not all of the body panels will open. Here is a diagram of the panels that will open as well as names assigned to them. Each Utility Arm Door will house a manipulator arm that can extend when the door is open. On the back of R2D2 will be a large section that can be manually removed to access the electronics inside (back body panel).



Controlling the Body Panel Servos:

I used the same I2C lines coming from the Arduino Mega to send signals to the Body servo driver. Before doing that however, I had to assign a new address to this driver board so the signals knew which driver board to activate. I gave this board the address of 0x42. That means I had to solder the two tabs together in the “A1” section of the driver board (bridge A1).



I can control these servos using the same type of commands as the dome panel servos.

PART C: Dome Lighting

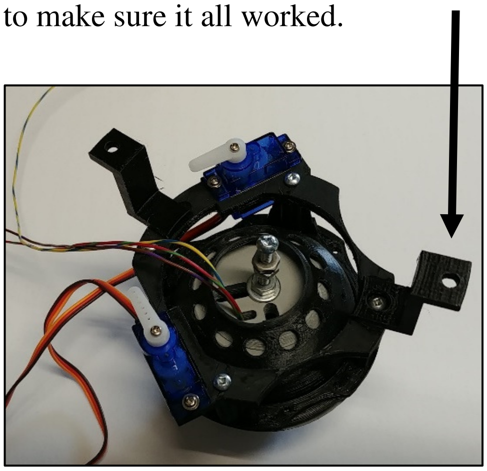
Holo Projectors

You have several choices when it comes to installing the Holo Projectors. You can design and build your own or download a file someone else has created and 3D print them. You can also purchase pre-made Holo Projectors from the Astromech site.
I chose two of these options. For the first part of my build where I was drafting everything up on my Proof of Concept Board, I chose to print off a set made by shmorgan at:
https://www.thingiverse.com/thing:2423670

shmorgan

Thank you shmorgan for providing these files for us to print.

I created another mount for these so I could screw them to my board and test out the electronics and code to make sure it all worked.



For my final set of Holo Projectors I plan on using aluminum ones created by BobC at the Astromech site (more on that later).
As far as the electronics go, I decided to use Flthymcnsty's (Ryan Sondgeroth) Holo Projector Sketch and NeoPixel LEDs. He has created a very nice setup using NeoPixels and servos that receive a command and respond accordingly. I would highly recommend this setup. You can purchase his kit off of the Astromech site. The handbook for his setup can be found here:
http://2geekswebdesign.com/FlthyHPs/...anual_v1.8.pdf

Flthymcnsty

Thank you, Ryan, for providing such an awesome way to control the Holo Projectors as well as giving us an incredibly detailed manual to follow.

Making it work with the Xbox 360 Sketch

Flthymcnsty's sketch requires a string of data to be sent in order to trigger the corresponding LEDs and servo movements. In order to make this work with the Xbox 360 sketch I had to add the following to the main code:

i. At the beginning of the code I added the following:
   - #include <SoftwareSerial.h>
   - #include <Adafruit_NeoPixel.h>
   - #include <Adafruit_PWMServoDriver.h>
   - #include <Servos.h>
   - #define I2C2ADDRESS 0x19 // 25 in Hexadecmal
   - #define SERVOI2CADDRESS 0x40 // Address of the servo board
   - #define FlthyTXPin 14 //Serial connection on Arduino Mega
   - #define FlthyRXPin 15 //Serial connection on Arduino Mega
   - const int FlthyBAUD = 9600;
   - SoftwareSerial FlthySerial(FlthyRXPin, FlthyTXPin);

ii. In the "**Setup**" section I added:
   - FlthySerial.begin(FlthyBAUD);
   - Wire.begin();
   -

iii. In the "Loop" section whenever I want to send a particular command, I send it like this: FlthySerial.print("R0063\r");
   - This example will send a command to toggle the Rear HP LEDs to Green (as explained in his manual).

I sent 2 wires (Serial Connection) through my slip ring from pins 14 and 15 on the Arduino Mega to the Tx and Rx pins on Flthymcnsty's Pro Mini board. I included a common ground wire as well going up through the slip ring (same one that ties all of the dome electronics together).

TXPin 14 goes to Rx pin on Pro Mini

RxPin 15 goes to Tx pin on Pro Mini

That’s it for now. I will continue to update this periodically. The 3 Appendices can be found below this page.

Appendix A

Button Click Functions

🟥 **Sound** 🟦 **Lights** 🟩 **Motion**

Y	Random Tracks 13-16
Y and L1	Track 8 – Annoyed Holo Projectors move back and forth
Y and L2	Track 2 - Chortle
Y and R1	Track 9 - Theme
Y and R2	Body Panel Routine
Y and D-Pad Up	
Y and D-Pad Down	
Y and D-Pad Left	
Y and D-Pad Right	

A	Random Tracks 17-24
A and L1	Track 6 – Short Circuit
A and L2	Track 1 – Scream 2
A and R1	Track 11 - Emperor
A and R2	Utility Arms Routine
A and D-Pad Up	
A and D-Pad Down	
A and D-Pad Left	
A and D-Pad Right	

X	Random Tracks 25-31
X and L1	Track 5 – Leah Front Holo Projector moves down and to the right LED Leah Sequence
X and L2	Track 4 – Wolf Whistle
X and R1	Track 12 – Chorus
X and R2	Dome Panel Wave Routine

X and D-Pad Up

X and D-Pad Down

X and D-Pad Left

X and D-Pad Right

B -------------------------------- Random Tracks 32-51

B and L1 ----------------------- Track 6 – Short Circuit
Short Circuit

B and L2 ----------------------- Track 3 – DooDoo

B and R1 ----------------------- Track 10 – Cantina

B and R2

B and D-Pad Up

B and D-Pad Down

B and D-Pad Left

B and D-Pad Right

L1 (Left Bumper)

L2 (Left Trigger)

R1 (Right Bumper)

R1 and D-Pad Up ------------- Volume Up

R1 and D-Pad Down --------- Volume Down

R1 and D-Pad Left

R1 and D-Pad Right

R2 (Right Trigger)

R2 and D-Pad Up ------------- 2-3-2 Up

R2 and D-Pad Down --------- 2-3-2 Down

R2 and D-Pad Left

R2 and D-Pad Right

D-Pad Up

D-Pad Down

D-Pad Left

D-Pad Right

Appendix B

Sound Clip Assignments

1	Scream 2	26	Ooh 2	51	Sent 20
2	Chortle	27	Ooh 3	52	Hum 19
3	DooDoo	28	Ooh 4	53	Hum 20
4	Wolf Whistle	29	Ooh 5	54	Adams Family
5	Leia	30	Ooh 6	55	Gangnam Style
6	Short Circuit	31	Ooh 7	56	Leia Snip
7	Patrol 1	32	Sent 1	57	Luke
8	Annoyed	33	Sent 2	58	Muppets
9	Theme Song	34	Sent 3	59	Bubye
10	Cantina	35	Sent 4	60	
11	Emperor	36	Sent 5	61	
12	Chorus	37	Sent 6	62	
13	Alarm 3	38	Sent 7	63	
14	Alarm 5	39	Sent 8	64	
15	Alarm 7	40	Sent 9	65	
16	Alarm 8	41	Sent 10	66	
17	Misc 3	42	Sent 11	67	
18	Misc 7	43	Sent 12	68	
19	Misc 14	44	Sent 13	69	
20	Misc 16	45	Sent 14	70	
21	Misc 17	46	Sent 15	71	
22	Misc 25	47	Sent 16	72	
23	Misc 30	48	Sent 17	73	
24	Misc 34	49	Sent 18	74	
25	Ooh 1	50	Sent 19	75	

Appendix C

Arduino Mega Pinout Assignments

A1		20	(SDA) – Dome and Body Servos via Servo Driver boards
A2		21	(SCL) – Dome and Body Servos via Servo Driver boards
A3		22	
A4		23	
A5		24	
A6		25	
A7		26	
A8		27	
A9		28	
A10		29	
A11		30	
A12		31	
A13		32	
A14		33	
A15		34	
1	(TX0) – MP3 Trigger	35	
2		36	
3		37	
4		38	
5		39	
6		40	
7		41	
8		42	
9		43	
10		44	
11		45	
12		46	
13		47	
14	(TX3) – Serial Connection to Holoprojectors	48	
15	(RX3) – Serial Connection to Holoprojectors	49	
16	(TX2) – Syren 10 Motor Controller	50	
17		51	
18	(TX1) – Sabbertooth Motor Controller	52	
19	(RX1) – Sabbertooth Motor Controller	53	