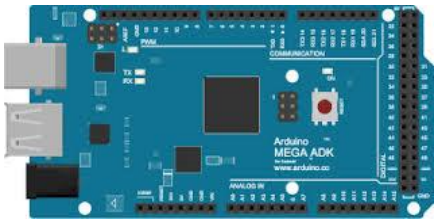
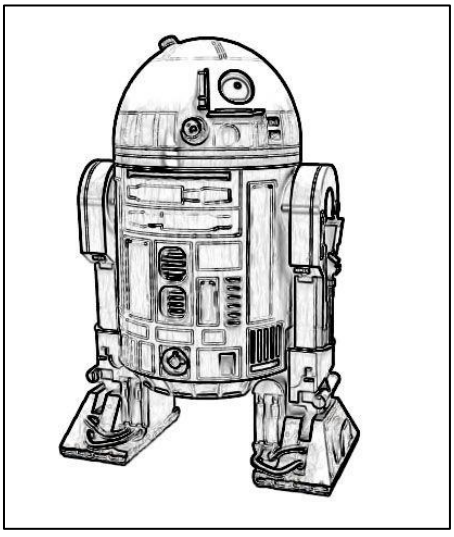
PADAWAN 360 AND THE MAESTRO CONTROL BOARD

MAKERDAN (DAN MASSEY)

VERSION 1.0

SPECIAL THANKS TO MICHAEL BADDELEY





This is a tutorial on how to interface the Padawan 360 control system with the Maestro servo driver boards for droid automation.

The Padawan code does a great job using the Xbox 360 controller. However, you have to remember that while the loop is running to accomplish these tasks, you don't want to stop the loop or slow it down in any way otherwise you may lose control of your droid for a short period of time. The Maestro boards described here will store pre-programmed sequences for servos. This allows the main Arduino loop to keep going while the servos carry out their tasks.

These boards come in 4 different varieties as shown below. Each of these microcontrollers can drive 6,12,18 and 24 servos each respectively. The smallest one is called a Micro Maestro and the others are called Mini Maestros.

Information and software for these boards can be found at Pololu.com.

Here is a link to the Maestro PDF manual:
https://www.pololu.com/docs/pdf/0J40/maestro.pdf

Pololu also provides free software that you can use (Pololu Maestro Control Center) to program the boards. There are many advantages to using these boards.

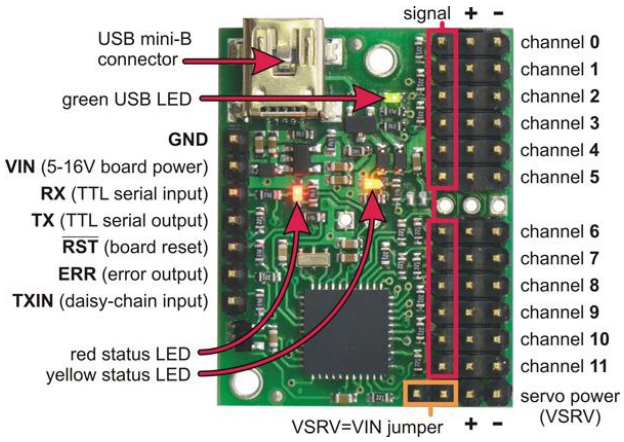
1. You can create and save servo routines (sequences) onto the board and activate them at any time in the Arduino code using a button click from the Xbox 360.
2. While the sequence is activated on the Maestro, the normal Arduino loop keeps running which allows you to have full control over the drive and dome motors.
3. The Maestro software allows you to select the minimum and maximum endpoints of each individual servo, so you don't end up going past the servo limits and accidently burning out your servos.
4. The Maestro software also allows you to vary the acceleration, and speed of each servo as well as have the script loop if needed.

Michael Baddeley

Although I had used one of these boards many years ago, I had not really understood their potential until I watched the first Zoom Tech session put on by Michael Baddeley. I would like to thank him for the tremendous amount of work he has done for the droid building community and the help he has given countless people. My explanation will follow along with each of the Zoom tech sessions he has provided. The first Zoom Tech session can be found here:

https://www.youtube.com/watch?v=SLkgoe9RS10&list=PLXkMwp_Z-ip1TCwXoLOuODb9SZYeoG-zG&index=8

Pinouts found on a Maestro board (example using the Mini Maestro 12)



It would be best to supply the board with a well-regulated 5-volt supply to VIN and a separate 5-volt supply to the servo power pins (VSRV) that can handle enough amps to drive as many servos as needed. Something like the one below will work well.

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

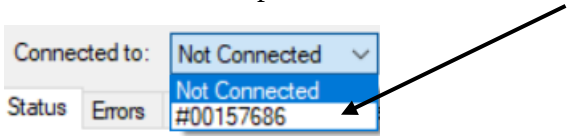
Note: it is possible to move the jumper connector at the bottom of the board to the right one position. This will allow you to power the board and the servos with one voltage line provided the line has enough amps to power the board and all the servos assuming you are using 5-volt servos.

Using the Software

These are the tabs you will find at the top of the Control Center software

Status	Errors	Channel Settings	Serial Settings	Sequence	Script

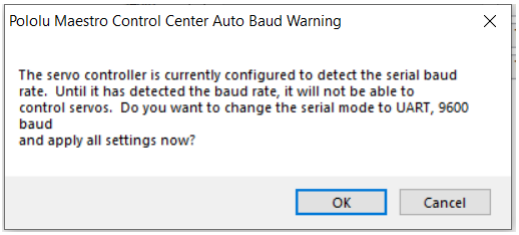
Once you have downloaded the software and plugged the USB cable into your board you should see a flashing green light appear on the board and you will most likely not get a connection right away. You will have to select your board in the drop-down box next to "connected to:"



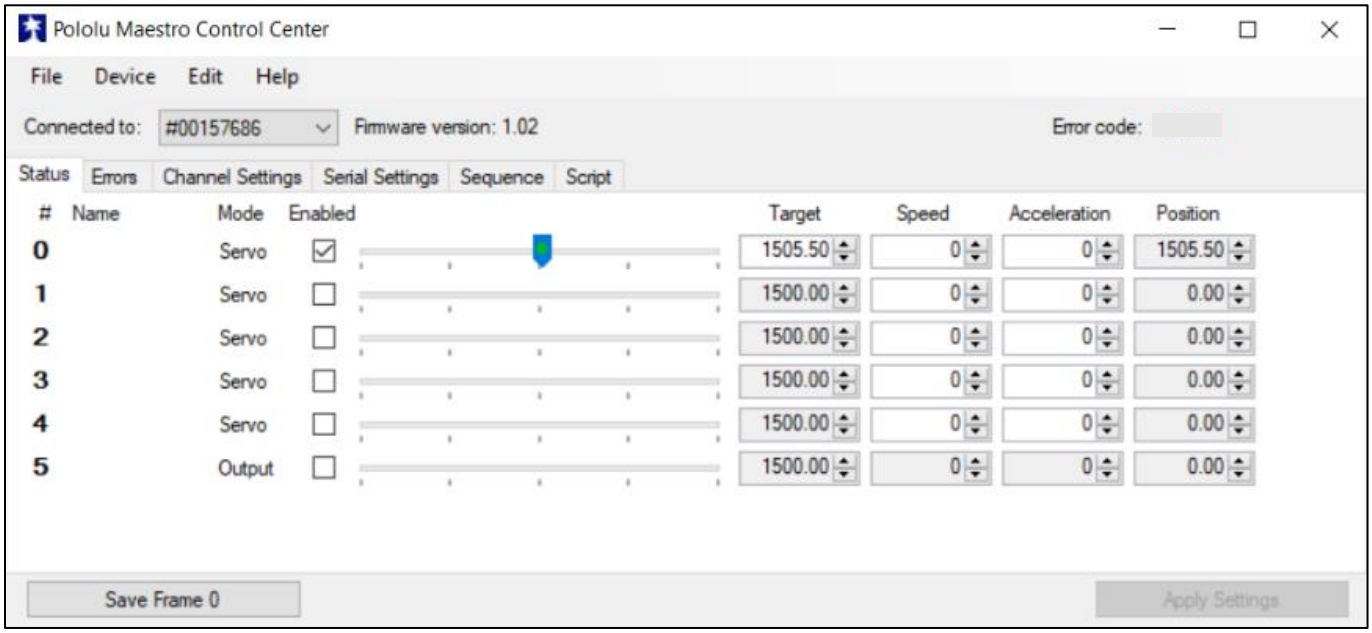
Once you select your board, you will be able to test out any servos you have connected to it.

Status Tab: This tab provides a real-time view of what your servos are doing

Plug a servo into the "Zero" slot (first servo slot) on your Maestro board. When you click "Enabled" on your first servo, you may get a dialogue box that looks like this:



This is telling you that the control settings are going to automatically change the baud rate to 9600. Click "OK". Now try testing the board with a single servo. You should be able to control the first servo by dragging the slider left and right.



Here are what the titles on in this window mean:

 # Name – the number of the servo and its name (the name can be changed in the channel settings tab)

Enabled – lets you know if the servo is activated or not

Target/Position – gives you the lower and upper limits of the servo you are using (1500 being the middle position of the servo)

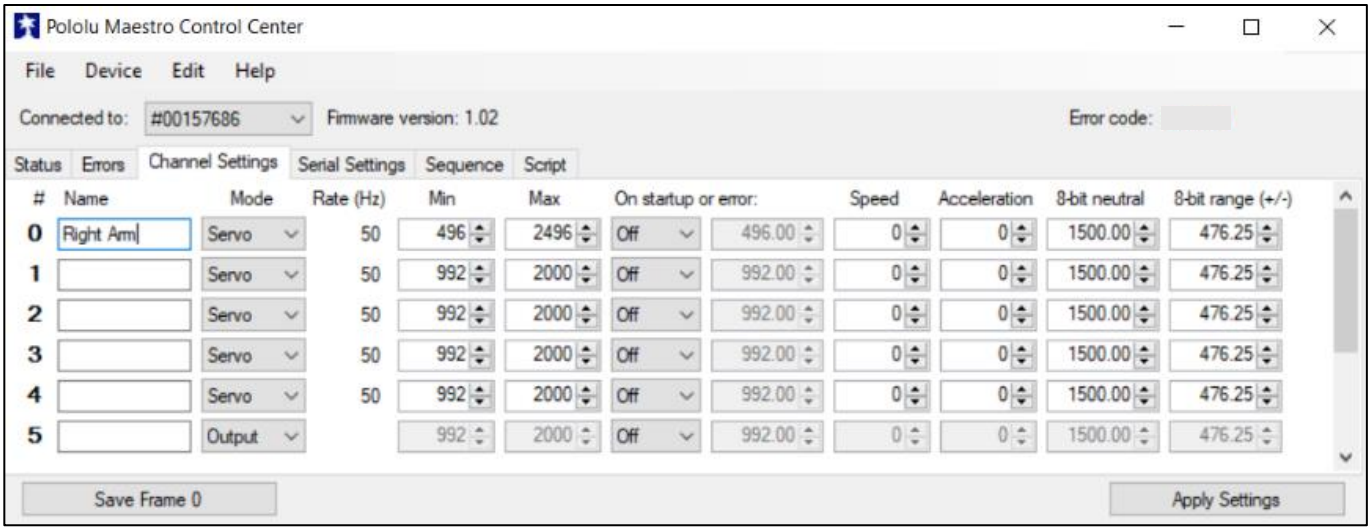
Speed – controls the speed of the servo (1 being the slowest speed and zero is a default for the fastest)

Acceleration – how fast the servo will accelerate and de-accelerate to the next position

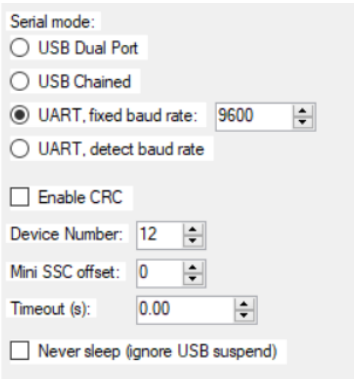
Try playing around with the settings and see how it affects the servo that is attached to your board.

Errors Tab: If something is wrong with the board or the script is incorrect or not working, you may get a red LED appear on your board and an error message show up in this window. Here, you can find out what kind of an error you have and clear it off the board if needed (more on this later).

Channel Settings Tab: This tab shows you the default settings for each servo and allows you to name each servo. Once you have made changes in this window, click "Apply Settings" at the bottom of the screen. If you power off your board and restart it, then these settings will be saved. If you navigate back to the "Status" tab, the new names will show up. You can also pre-set the Min and Max positions for each servo in this window. As well, you can pre-set a startup position for each servo in case you want to have a servo move to a preset position when the board is powered up.



Serial Settings Tab: Here is where you can select the type of serial connection you would like for your board. For an Arduino Mega, you need to have it set to "UART, Fixed Baud Rate 9600".



Before going any further, I want to explain some of the terminology involved with the Maestro system. **Sequences** are routines you can create for servos. They are numbered by default but can be called anything you like. Sequences can also go by the name "**subroutines**". You can create many sequences on each board based on the amount of memory the board holds. Sequences should not be confused with a "**Script**". A Maestro board can only have 1 script on it at any given time. All the sequences you create are part of the script and can be called to run based on their names.

Sequence Tab: In this section you can see and control a set of sequences that you created in the Status tab. Here are the steps to program a sequence (servo zero to move back and forth).

1. Click on the Status tab and enable the first servo.

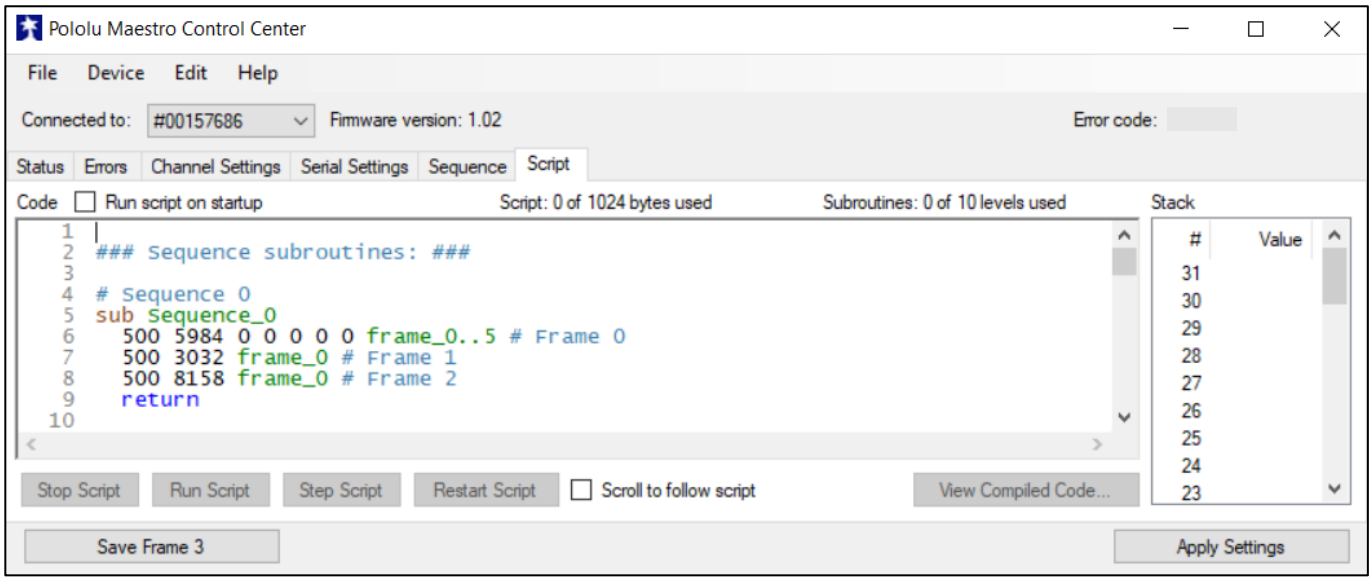
2. Click “Save Frame 0” at the bottom left of the window. (this saves the starting position of the servo)
3. Move the servo control to another position by either dragging it or typing in a value into the “position” or “target” boxes.
4. Click “Save Frame 1”
5. Move the servo again and then click “Save Frame 2.
6. Click on the “Sequence” tab and you should see all 3 frames on the left side of the window
7. Click on “Play in Loop” and then “Play Sequence”. Your servo should now move through the sequence and then repeat. You can watch this in real time if you click on the “Status” tab.
    a. Note: if you slowed down the speed of your servo and notice that your servo doesn’t get to the endpoints of the target or position values, it means that you need to give the sequence more time for the servo to move to the next position. You can do this by double clicking the “Duration” amount in the “Sequence” tab and changing the default duration from 500 ms to something larger such as 1200 ms.

Try creating a sequence with 2 or 3 servos. Also, click the “New Sequence” tab and create a second and third sequence so you have a few sequences to experiment with (the software will work if you have created at least 2 sequences). Try playing the sequences here to see how they work.

Note: you can change the name of the sequence by clicking on the “Rename” button in the “Sequence” window.

So far, these sequences are only stored in the software. In order to export it to the board where it will be held permanently, you need to click the button that says, “Copy All Sequences to Script”. When you do this, you may be given a message that warns you that your sequences will write over any other sequences previously stored. Go ahead and click “OK”. The software will now jump to the “Script” tab.

Script Tab:



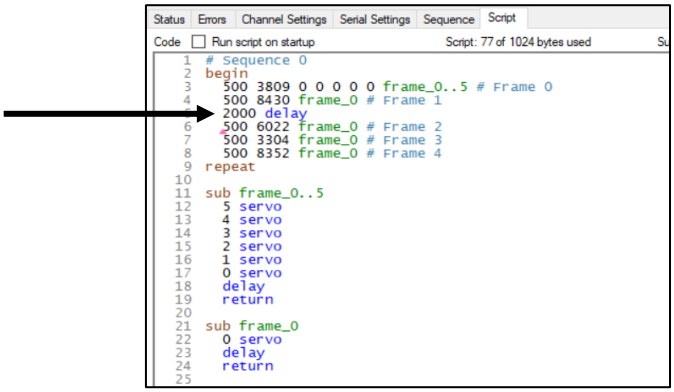
The Script window will show you what the actual code looks like. A Script is synonymous to a Sketch in the Arduino programming language. Here you should be able to see all the sequences you created earlier. If you now click on "Apply Settings", your script will be saved onto the Maestro board permanently.

If you click in the box that says, "Run script on startup" and then "Apply Settings", the script you just created will run every time you power up the Maestro board.

If you want to create a new sequence just click on the "New Sequence" button in the Sequence tab.

The Maestro software has its own set of commands that you can type into the Script and create more versatile movements with your servos. Once you become familiar with writing code for the Maestro, you will be able to type code into this window to edit your script (you can find these commands in the Pololu Maestro PDF). For example, below, I will show you how to add a delay in the script if needed.

For instance, if you want a door to open on your droid and have it stay open for a certain length of time, you can alter the code by adding a line such as "2000 delay". This syntax is different from Arduino syntax. The 2000 refers to 2000 milliseconds (2 seconds) and the delay is the function that happens. All you must do is hit enter where you want the delay to go and type it in.



Part 2: The Padawan Code and Triggering Animations

The Padawan code uses button clicks and combination of button clicks from the Xbox 360 controller to activate animations in the droid. You can use these same buttons and combinations of buttons to trigger the animations (scripts) you created on the Maestro control board.

Serial Connection: For these examples I will be assuming you are using an Arduino Mega. The way I will explain how to connect the Maestro to your Arduino is through a Serial Connection. The Arduino Mega has 3 Serial ports. 2 of them are most likely already being used to control the Sabertooth and Syren motor controllers. We could use the last serial port on the Mega but, in this case, we will create a virtual serial port called a "Software Serial Port" that will control our first Maestro board. You can make your own software serial port by assigning certain pins on the Arduino to now be a software serial connection. When you do this, you must add a few lines to the main code in order to activate the software serial connection.

Necessary Code Summary: See below for a detailed description

```
//Add these lines into the Padawan code//

# include <SoftwareSerial.h>
# include <PololuMaestro.h>
SoftwareSerial MaestroSerial_1 (10,11);
MiniMaestro MaestroDome (MaestroSerial_1);


Void Setup

MaestroSerial_1.begin (9600);

Void Loop

    MaestroDome.restartScript (0);
```

Include the library

In order to use the Maestro boards, you will need to add some specific lines of code to the Padawan sketch.

Near the beginning of the sketch (**before** the Void Setup section) there are some lines that start with "# include". This command brings in certain libraries needed for various functions to work. You will add new libraries called PololuMaestro as well as SoftwareSerial. It should look like this once installed:

```
#include <Sabertooth.h>
#include <MP3Trigger.h>
#include <Wire.h>
#include <XBOXRECV.h>
#include <Adafruit_PWMServoDriver.h>
#include <Servos.h>
#include <SoftwareSerial.h>
#include <PololuMaestro.h
```

Create a Virtual Serial Port

You now want to tell the Arduino to create the virtual serial port for your Maestro board using the software serial library. You do this by adding the following line.

```
#include <SoftwareSerial.h>
#include <PololuMaestro.h>
SoftwareSerial MaestroSerial_1 (10,11); //Receive is pin 10 and transmit is pin 11
```

This means that you have created a virtual serial port on pins 10 and 11 of the Arduino Mega called MaestroSerial_1(this name can be anything you choose). I gave it an underscore 1 in case you need to make a second one in the future. The Rx (receive pin) is pin 10 and the Tx (transmit pin) is pin 11. You will most likely only be using pin 11 since you will probably only be sending signals out and not receiving signals. If this is the case, you then only need to attach a wire from pin 11 on the Arduino to the Rx pin on the Maestro.

Assign a Maestro Board to the Serial Port

The next thing to do is assign a Maestro board to this serial port and give the board a specific name (based on what you want it to control). In this example we will call it MaestroDome (assuming you want it to control servos in the dome).

```
SoftwareSerial MaestroSerial_1 (10,11); //Receive is pin 10 and transmit is pin 11

MiniMaestro MaestroDome (MaestroSerial_1);
```

Type of Maestro Board	What the Board will Control	The Serial it is Assigned to

The following line needs to be inserted **after** the **Void Setup** command. This is where you tell the Arduino to activate (begin) the serial port you just created. You do this by adding this line of code:

```
MaestroSerial_1.begin (9600); //begins the software serial connection
```

It is important to remember that you now need to physically connect your Maestro board to the Arduino Mega by connecting the ground pins together and also a wire (transmit) from pin 11 on the Arduino board to the Rx (receive) pin on the Maestro board.

Assign buttons on the Xbox Controller to Activate Scripts

Once you have the following lines of code added to the sketch, you can have various buttons on the Xbox controller trigger animations that you have saved onto the Maestro board ahead of time. This happens in the **Void Loop** section of the code. For example, scroll down in your sketch until you get to this section:

```
// GENERAL SOUND PLAYBACK AND DISPLAY CHANGING

 // Y Button and Y combo buttons // ******
 if (Xbox.getButtonClick(Y, 0)) {
   if (Xbox.getButtonPress(L1, 0)) {
     mp3Trigger.play(8);
```

This section in the code is describing what happens if you hold down the "Y" button and the Left Bumper on the controller at the same time (see appendix A for button locations). It should activate your MP3 Trigger to play audio clip #8. You have the option here to replace the audio clip line with a line of code that triggers your Maestro script or just add the line of code below the audio clip line so these buttons will trigger both functions. Here is what it would look like if you replaced the audio clip line with a line of code that triggers script zero from your Maestro board:

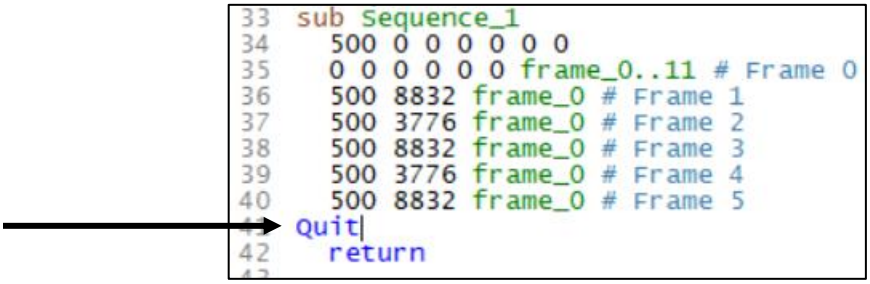
```
// Y Button and Y combo buttons // ******
if (Xbox.getButtonClick(Y, 0)) {
  if (Xbox.getButtonPress(L1, 0)) {
    MaestroDome.restartScript (0);
```

What this new line of code does is it restarts script zero that you have saved onto the Maestro board.

Upload the new code to your Arduino Mega.

When you hold down the "Y" button and the Left Bumper on your controller, you should have triggered script zero. Remember that you could have created a name for the script in the Pololu Control Center so you don't have to call it script zero there. You can call it something specific such as "Arm Routine" but if it is associated with a particular number such as sequence #0, then you have to use the number in the Arduino sketch such as restartScript (0) and not restartScript (Arm Routine).

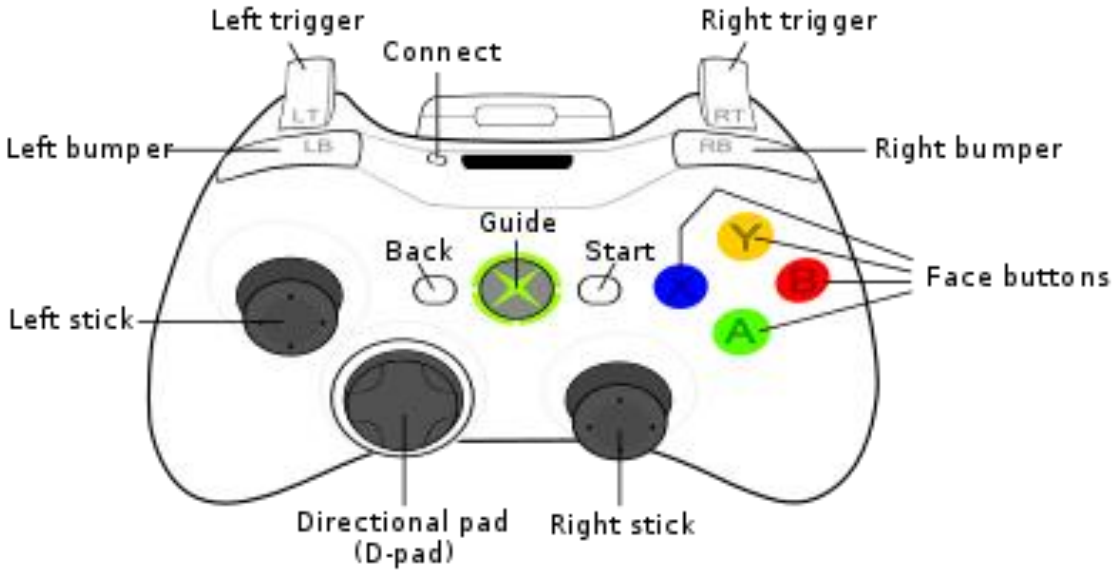
Note: As I mentioned earlier, if the red LED on the Maestro turns on, it is letting you know there is a problem with the software commands. If you look at the Error tab it will let you know the type of error that is occurring so you will have some information about how to fix it. Sometimes the error will prevent you from triggering the animation and sometimes it will just show up but still allow the routine to function. With some of my sequences the error led came on and wouldn't go away so the way I fixed it was to add the word "Quit" at the end of my sequence and just before "repeat". This command stops the sequence and prevents the error. (Remember to "Apply Settings" once you make this addition)



This concludes my tutorial on using the Maestro board with the Arduino Padawan 360 sketch. This just covers the basics. If you spend time reading through the PDF manual from Pololu you will discover many amazing things you can do to program your sequences.

Happy Building!

Appendix A – Button Layout



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

LL: Logic Lights
HP: Holo Projector

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

Start: Enable/Disable Movement (Right stick)

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.