**TRAINING THE NEXT GENERATION OF JEDI SCIENTISTS**

**1. CONTACT**

Corresponding Author: Jacob George

Email: [Jacob.George@Utah.edu](mailto:Jacob.George@Utah.edu)

Webpage: [faculty.utah.edu/~JacobGeorge](https://faculty.utah.edu/~JacobGeorge)

**2. REFERENCES**

J. A. George, C. R. Butson, “Training the Next Generation of Jedi Scientists,” Society for Neuroscience, Chicago, IL, October, 19, 2019. DOI: [10.13140/RG.2.2.16405.06888](https://doi.org/10.13140/RG.2.2.16405.06888) (Poster)

J. A. George, C. R. Butson, “Training ‘Jedi Scientists’ to Control Bionic Arms with their Minds,” Utah Biomedical Engineering Conference, Salt Lake City, UT, September 13. DOI: [10.13140/RG.2.2.14253.84964](https://doi.org/10.13140/RG.2.2.14253.84964) (Poster)

**3. EQUIPMENT**

**3.1 Overview**

The following equipment is recommended for each group of 1-4 students. Two Backyard Brains Muscle SpikerShields per Arduino Uno allows simultaneous recording of two channels. The software and learning objectives have been developed around this two-channel configuration. That being said, using a single channel (i.e., a single SpikerShield per Arduino Uno) is also possible. Furthermore, alternative SpikerShields, such as the [Heart and Brain SpikerShield](https://backyardbrains.com/products/heartandbrainspikershieldbundle) will also work.

**3.2 Hardware**

* Laptop / Tablet / Desktop
* Arduino Uno
* Backyard Brains Muscle SpikerShield (x2)
* Disposable Surface Electrodes

**3.3 Software**

* Windows 64-bit (x64)
* MATLAB
* MuJoCo HAPTIX
* Arduino IDE

**3.4 Files**

* JediScientist.ino
* JediScientist\_demo.m
* JediScientist.m

**4. PURCHASING**

**4.1 Arduino Uno & Backyard Brains Muscle SpikerShield**

You can purchase two [bundles](https://backyardbrains.com/products/muscleSpikershieldBundle) from Backyard Brains for $299.98 total. Each bundle contains one Arduino Uno and one Backyard Brains Muscle SpikerShield. Alternatively, you can purchase one [Arduino](https://store.arduino.cc/usa/arduino-uno-rev3) ($22.00) and two [DIY Muscle SpikerShields](https://backyardbrains.com/products/diyMuscleSpikerShield) ($64.99 each). This reduces the total cost to $151.98, but also requires self-assembly of the two SpikerShields using a soldering iron, some solder and wire cutters.

**4.2 Disposable Surface Electrodes**

You can purchase [100 low-cost surface electrodes](https://backyardbrains.com/products/emglargeelectrodes) from Backyard Brains for $29.99. Alternatively, you can purchase [100 medical-grade EMG/ECG/EKG electrodes](https://bio-medical.com/covidien-kendall-disposable-surface-emg-ecg-ekg-electrodes-1-24mm-50pkg.html) for $37.98.

**5. SOFTWARE INSTALLATION**

**5.1 Windows 64-bit (x64)**

Double check your system is running Windows 64-bit (x64). Go to “System Information” and then look at the item titled “System Type.” If it says ‘x64-based’ then your computer is a 64-bit operating system and you are good to go.

**5.2 MuJoCo HAPTIX**

**5.2.1 Download**

MuJoCo HAPTIX can be downloaded can be downloaded for free through [Roboti](https://www.roboti.us/index.html). In most cases, you should install the latest version ([mjhaptix150](https://www.roboti.us/download/mjhaptix150.zip)). If version 150 fails to load, or if your computer processor lacks AVX support (e.g., processors built before 2011 or lower performance tablets like the Microsoft Surface Go), try installing version 140 ([mjhaptix140](https://www.roboti.us/download/mjhaptix140.zip)) instead. Extract the zip folder to a location on your computer that you can easily access. You will need to know this location later.

**5.2.2 Running MuJoCo HAPTIX**

Navigate to and then open the extracted folder “mjhaptix150.” Open the folder titled “program” and run the executable titled “mjhaptix.exe.” *If a windows firewall pop-up occurs, make sure to allow access on both private and public networks!*

**5.2.3 Loading Models**

In the top left, click the  icon. In the “Open model” dialog box, navigate to and open your extracted folder titled “mjhaptix150.” Open the folder titled “model.” Open the folder titled “MPL.” If you are recording signals from your right hand, open “MPL.xml.” If you are recording signals from your left hand, open “MPL\_left.xml.” Other models, such as “MPL\_boxes.xml” can be used to evaluate performance later. Make sure to start the model using the  play button (or by pressing the spacebar).

**5.3 MATLAB**

**5.3.1 Version**

Version 2016B or later is preferred, although probably not necessary. Do NOT use a web version of MATLAB. It must be installed locally on your machine.

**5.3.2 Path**

You will need to add the MuJoCo folder (e.g., “mjhaptix150”) to the MATLAB path. At the top left of MATLAB, go to the “HOME” tab. At the top center of MATLAB press the  tab labeled “Set Path.” In the Set Path dialog box, select the option “Add with Subfolders.” Browse to and select the “mjhaptix150” folder. Then press save and close the dialog box.

**5.4 Arduino IDE**

The Arduino IDE is only needed once to upload the appropriate software onto the Arduino. This can be done from a separate computer (Mac, Windows, or Linux) and completed by the instructor ahead of time. *Students should not need to install the Arduino IDE unless they accidentally change the software on the Arduino.*

**5.4.1 Installation**

Download the appropriate version of Arduino IDE for your system. This can be done from the [Arduino homepage](https://www.arduino.cc/en/Main/Software) or from the Windows app store. Once the software is installed, use File 🡪 Open to select the “JediScientist.ino” file. If you receive a message stating that the file needs to be inside a sketch folder with the same name, press okay.

**5.4.2 Flashing the Arduino**

Next you will need to upload (flash) the “JediScientist.ino” file to the Arduino. First, go to Tools 🡪 Board and select “Arduino/Genuino Uno.” Then go to Tools 🡪 Port and select the COM port associated with your Arduino Uno. Next, upload the file to the Arduino using the  upload button or by going to Sketch 🡪 Upload. *After flashing the software, close the Arduino IDE software entirely or deselect the COM port by going to Tools 🡪 Port and selecting a COM port other than the Arduino.* You will not be able to communicate with the Arduino in MATLAB if the COM port is in use by the Arduino IDE.

**5.4.3 Debugging**

If you receive an error when trying to upload the software, first make sure you have selected the right Board and COM port. If no COM port shows up, try replacing the USB 2.0 cable or plugging into a different USB port. If you are still having issues with the correct COM port selected, try closing the software, unplugging the Arduino, plugging the Arduino back in, and then re-opening the software.

**6. HARDWARE CONFIGURATION**

You will need to stack two Backyard Brains Muscle SpikerShields on top of one another to record two channels simultaneously. Before you stack the SpikerShields, make sure the jumpers are in the correct positions. For SpikerShield that is directly connected to the Arduino Uno, move the jumper from pin A0 to pin A1 (Figure 1). For the SpikerShield that will be placed on top of the other SpikerShield, make sure the jumper is on pin A0. Next, stack the SpikerShields, making sure the pins are aligned properly and in full contact.



Figure 1: Shunt jumper on the Backyard Brains Muscle SpikerShield. The jumper (red box) needs to be moved from Pin A0 to Pin A1 for the SpikerShield connected directly to the Arduino Uno. If the jumpers for the two SpikerShields are both on the same pin (e.g., both on Pin A0) then the two channels will be shorted together and have the exact same output.

**7. RUNNING THE DEMO**

**7.1 Electrode Placement**

**7.1.1 Extrinsic Extensor Hand Muscles**

Locate your extrinsic extensor hand muscles by placing your hand on a flat surface and repeatedly lifting up your middle finger. While doing this, place your other hand on the dorsal portion of your forearm, distal to your elbow. You should feel your muscles contracting in your forearm. Place two electrodes along the length of your muscle (i.e., along the length of your arm where you feel the contractions) a few centimeters apart. Connect the bipolar alligator clips (colored) to these two electrodes. Next, place a third electrode proximal to your elbow and connect the ground alligator clip (black) to it. Connect the auxiliary cable to the auxiliary port on the second (top) SpikerShield. The final configuration should resemble Figure 2.

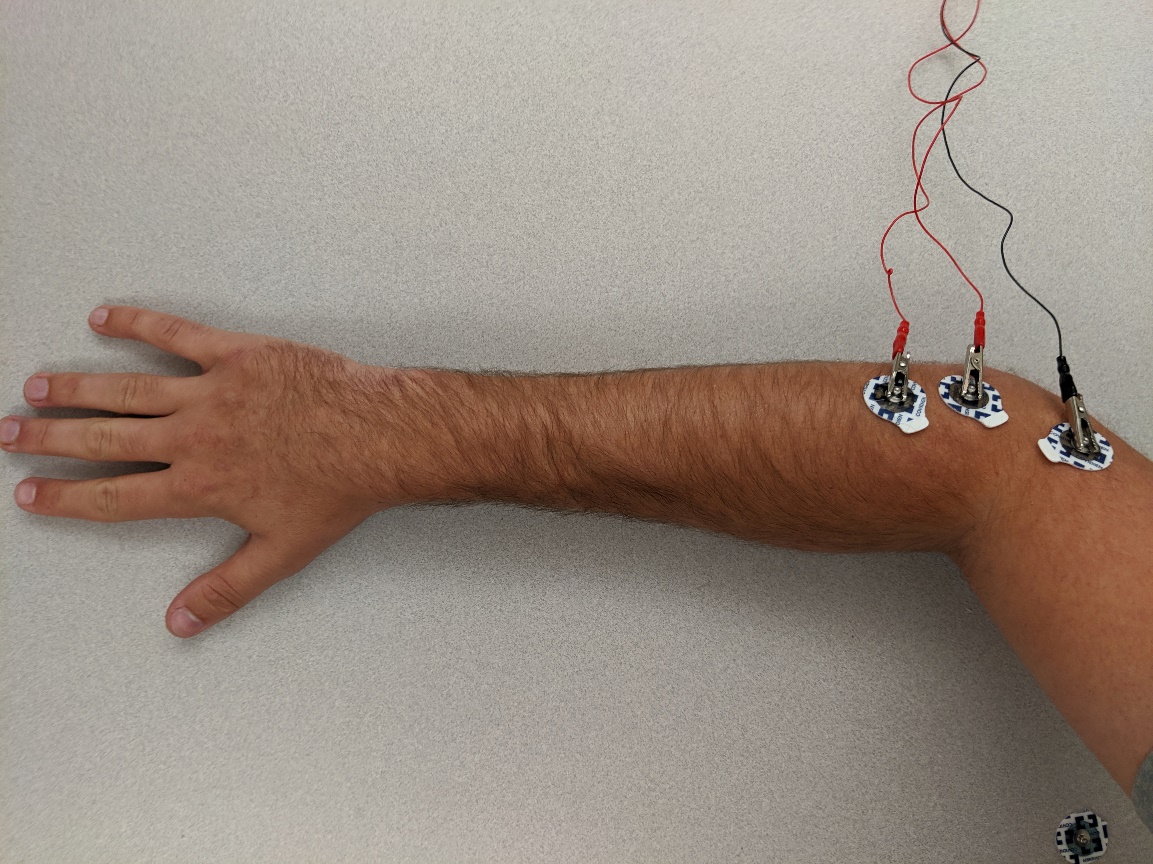


Figure 2: Extrinsic extensor hand muscles. Try to place the tabs of the electrodes such that when you pull the electrode off by the tab it goes in the direction of the hair on the forearm.

**7.1.2 Extrinsic Flexor Hand Muscles**

Locate your extrinsic flexor hand muscles by repeatedly flexing your pinky and ring fingers. While doing this, place your other hand on the ventral side of your forearm, distal to your elbow. You should feel your muscles contracting in your forearm. Place two electrodes along the length of your muscle (i.e., along the length of your arm where you feel the contractions) a few centimeters apart. Connect the bipolar alligator clips (colored) to these two electrodes. Next, place a third electrode proximal to your elbow and connect the ground alligator clip (black) to it. Connect the auxiliary cable to the auxiliary port on the first (bottom) SpikerShield. The final configuration should resemble Figure 3.

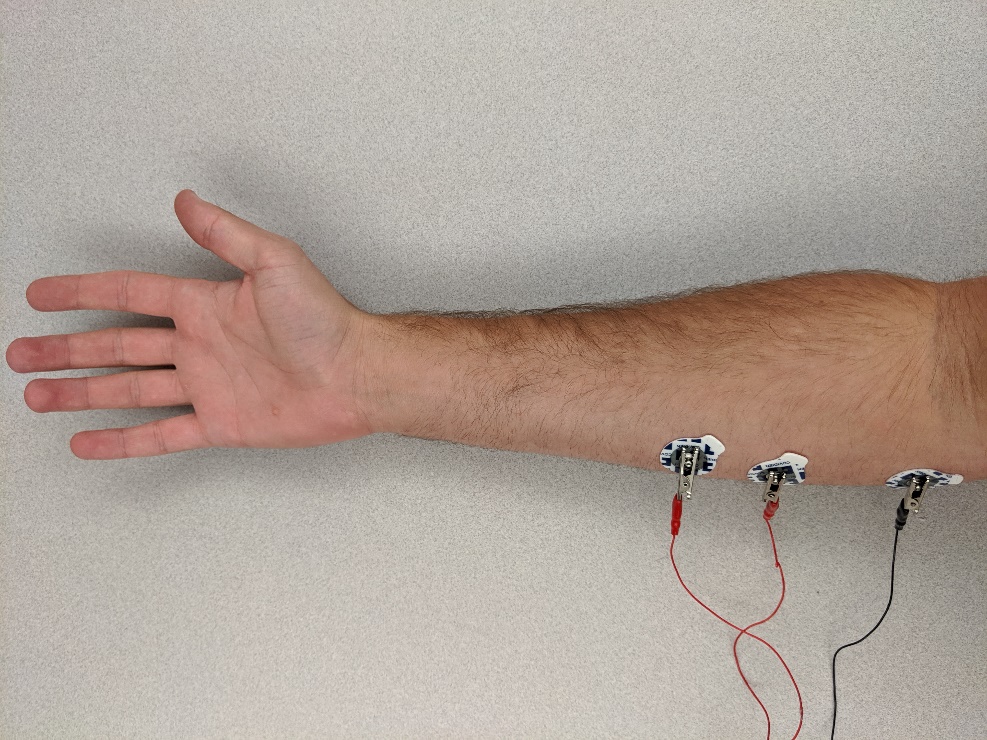


Figure 3: Extrinsic flexor hand muscles. Muscles are located by feeling for contractions in the ventral forearm when performing repeated grasps.

**7.2 Software Setup**

The JediScientist script is written in three sections of code. It’s recommended to run each section at a time using the  run-and-advance button. The first section establishes a connection to the MuJoCo virtual environment, the second section establishes a connection to the Arduino Uno, and the third section continuously acquires signals, displays them, and moves the hand in real-time.

**7.2.1 Section 1: MuJoCo**

Before running this section, double check that MuJoCo is open and that a model is open and running. Use the  icon to open a model and the  play button to run the model. After running the section of MATLAB code, you should receive a print out stating “VRE connected!” If you are seeing “VRE failed to connect” double check that the “mjhaptix150” folder is in the MATLAB path and that Windows firewall is not blocking communication to MuJoCo.

**7.2.2 Section 2: Arduino**

Before running this section, double check that the Arduino is plugged in and recognized by the computer. When plugged in, the Arduino should light up and your computer should give the Arduino Uno a unique COM port. You will need to identify the COM port associated with your Arduino Uno. Open “Device Manager” on Windows, scroll down to “Ports” and identify the COM number associated with your Arduino Uno. If you Arduino Uno does not show up in Device Manager, try a different USB port or USB cable. Once you’ve identified your COM port, modify the following line in MATLAB to specify your COM port:

uno = serial('COM16','BaudRate',115200); %setup connection to Arduino

By example, if your Arduino is connected on COM5, you would modify the text ‘COM16’ to ‘COM5’. If your Arduino is failing to connect, first double check that the COM port is not in use by another program (e.g., Arduino IDE). Second, check that the COM port is not listed twice in your device manager. If you see multiple devices with the same COM number, the simplest solution is to just use a different Arduino. Other issues connecting to the Arduino may be resolved by reconnecting the Arduino and then running MATLAB in Administrative Mode. To do this, right click on the MATLAB shortcut and select “Run as Administrator.”

**7.2.3 Section 3: Real-Time Plots and Control**

Running this section should plot EMG signals in real-time. If you are receiving an error stating the specified format “%d %d” could not be matched, you have the wrong software loaded onto the Arduino. To fix this, see section 5.4.2 “Flashing the Arduino”. Occasionally, you may see some warnings indicating the data was not received in the proper timeframe – these are generally okay as long as long as the EMG data is still valid (see section 7.3 “Verifying EMG”).

**7.3 Verifying EMG**

**7.3.1 Overview**

There are a few different issues that can arise when trying to display the EMG signals in real-time. Broadly these include: low sampling rate, eternal noise, and poor electrode placement. The following steps should help eliminate these issues.

**7.3.2 Validate Sampling Rate**

EMG has a frequency band from roughly 15-375 Hz. Per the Nyquist theorem, to capture the full EMG signal you will need to sample at least double the maximum frequency of the signal (e.g., sample at greater than 750 Hz). The Arduino samples the EMG at 1000 Hz and sends it to MATLAB in a serial buffer. MATLAB will attempt to read the read EMG as fast as possible, up to 1000 Hz. However, this is often difficult for many lower performance laptops and tablets to achieve.

First, validate that the signals are coming in real-time. Connect one channel (see XXX) and verify that the EMG signals are updating in real-time. Alternatively, you can quickly check this by repeatedly tapping the bipolar leads to the ground lead; the signal sound rail on contact in real-time. If the signals are excessively noisy or lagged, follow the steps in the next sections.

**7.3.3 Increasing Sampling Rate**

First, make sure your computer is running at maximum performance. Typically, this involves changing the Windows “Power Plan” to “Best Performance” or “High Performance” instead of “Power Saving.” It’s also recommended to close any additional background programs that may be using CPU resources.

Second, reduce the graphics on the MuJoCo. Press the  settings button, and then click the “Render” tab at the top, and uncheck the following: Shadow, Reflection, Skybox, Texture.

Third, try modifying the setting in MATLAB. MATLAB attempts to sample the EMG as fast as possible and only updates the plots and MuJoCo at a lower frequency specified by the “delay” parameter. For example, a “delay” of 0.05 seconds results in a display rate of 20 Hz. Try decreasing this down to 0.2 seconds (5 Hz).

displayPlots = true; %visualize EMG and control values in real-time

displayVRE = true; %see hand move in real-time

delay = 0.05; % SEE NOTES BELOW

% This delay setting is necessary for computers with limited processing

% power. Use the general guidelines to start:

% Delay = 0.2; FOR VIRTUAL REALITY AND PLOTTING IN REAL-TIME

% Delay = 0.1; FOR VIRTUAL REALITY ONLY

% Delay = 0.05; FOR PLOTTING ONLY

You can also try disabling the real-time plotting or updates to MuJoCo by setting the “displayPlots” or “displayVRE” flags to “false.”

**7.3.4 Reducing Noise**

First, if you’re using a laptop of tablet, try unplugging your computer from the charger to eliminate 60 Hz noise. Second, make sure the pins of the second (top) SpikerShield are fully in contact with the first (bottom) SpikerShield. Third, make sure the alligator clips are fully connected to the electrodes and that the electrodes are in full contact with the skin. If the electrodes are old or were left out, the conductive gel may have dried out and the electrode may need to be replaced. If you still are having trouble getting a strong EMG signal, double check your electrode placement (see section 7.1 Electrode Placement).

**7.4 Controlling the Hand**

**7.4.1 Overview**

Several methods exist to convert raw EMG signals into an interpretable control signal. One approach is to use the mean absolute value of the EMG over the last ~300 milliseconds. This is calculated by zero-centering the data (i.e., subtracting off the average), taking the absolute value, and then taking the mean value of the last 300 samples (300 samples / 1000 samples/second = 0.3 seconds).

With the recommended electrode placement, the mean absolute value of the top SpikerShield (extensor muscles) will control the virtual wrist and the mean absolute value of the bottom SpikerShield (flexor muscles) will control the virtual hand. This means that extending your wrist should lead to extension of the virtual wrist and making a fist should lead to flexion of the virtual fingers.

**7.4.2 Adjusting the Gain**

Your maximum voluntary contraction should result in a maximum control value and fully flex the fingers or fully extend the wrist. If it’s too difficult to max out the control value, try increasing the gain parameter. If the control value seems to be always maxed out, try decreasing the gain parameter.

GAIN = [4,2]; % INCREASE VALUES TO INCREASE GAIN ON THE CHANNELS

In the example above, the gain on first (bottom) SpikerShield, which controls finger flexion, is set to 4, and the gain on second (top) SpikerShield, which controls wrist extension, is set to 2. In general, you want that smallest gain parameter that still allows you to consistently max out the control value with a maximum voluntary contraction.

**7.4.3 Removing Baseline Noise**

When your arm is at rest, the control values should read zero. If the values are greater than zero, increase the noise parameter to subtract off a larger amount of noise.

NOISE = [0.35,0.35]; % INCREASE VALUES TO ACCOMODATE NOISEY CHANNELS

Similar to the gain parameter, the first value controls finger flexion and the second value controls wrist extension. You can also use the noise parameter to help reduce cross-talk (e.g., from co-contractions). For example, increase the noise parameter for finger flexion until actively extending your wrist does not cause finger flexion.

**7.4.4 Adjusting the Threshold**

Most prostheses employ simple binary control such that the prosthesis only closes when the control value exceeds a predetermined threshold. This demo also uses a threshold that you can adjust. Increasing the threshold will make movement initiation harder.

THRESH = 0.5; % MAKES CONTROL BINARY

A limitation of current prostheses is that simple binary control does not allow for fine force regulation that would be useful when manipulating fragile objects. Setting the threshold to zero will enable proportional control of the prosthesis, such that the user directly controls the degree to which the fingers flex and wrist extends. Stronger muscle contractions will result in greater movement.

**7.4.5 Moving the Hand in MuJoCo**

Some MuJoCo models have objects that you can interact with. To move the MuJoCo arm in space, first select the arm by double-clicking the square joint at the end of the arm (Figure 4). The joint should highlight once you select it. You can translate the arm about a vertical plane by holding CRTL + Right Click. You can translate the arm about a horizontal plane by holding CTRL + Right Click + Left Click. And lastly, you can rotate the arm about the joint using CTRL + Left Click.

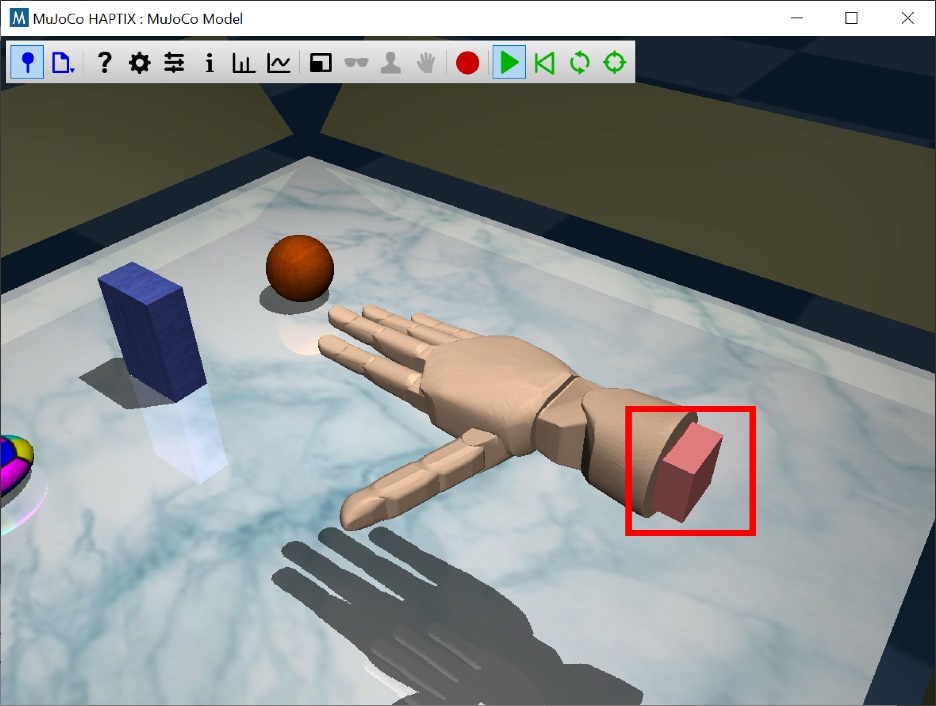


Figure 4: Moving the MuJoCo virtual arm. Double-clicking the pink square at the base of the arm (red box) will select the hand and enable movement commands.

**8. OUTREACH ACTIVITY**

**8.1 Overview**

Best introduced by movie-clips from Starwars, this activity is an exciting way to get K-12 students engaged in STEM. Potential learning objectives include both basic science principles (e.g., human anatomy, neuroscience) and engineering applications (e.g., brain-computer interfaces, prosthetic design). The instructor can quiz the students on a variety of topics while wearing electrodes and running the demo. Afterwards, groups of 1-4 students can try it out on their own.

**8.2 Example Learning Objectives**

* Motor intent is a series of electrical signals from the nerves and muscles
* Both imagined and commanded movements are detectable in the brain; only commanded movements are detected in the nerves
* Action potentials in the periphery are deterministic; a single action potential will result in a single muscle twitch
* Electromyography (EMG) is the recording of electrical activity from the muscles
* The muscles that control the hand are located in the forearm
* Extensor muscles are dorsal, flexor muscles are ventral
* EMG records the potential difference across the muscle with respect to the ground
* EMG is not directional; switching the polarity (i.e., swapping the bipolar alligator clips) has no functional impact
* EMG activity increases with muscle contractions
* EMG correlates highly with muscle force
* EMG activity can be used to measure muscle fatigue
* Agonist muscles cause movement through their own activation
* Antagonist muscles produce an opposing joint torque to the agonist muscles
* A co-contraction is the contraction of agonist and antagonist muscles around a joint to hold a position
* The mean absolute value is a common metric for making EMG signals more interpretable
* Myoelectric prosthetics use muscle activity as a control signal
* Muscle activity is still present even if your hand does not move, and muscle activity still exists after a transradial amputation

**9. EXAMPLE COLLEGE-LEVEL EXERCISE**

**9.1 Overview**

The next few lectures will show examples of how neural interfaces can be used for control and sensing from prosthetic hands. In this lab you will use neural recordings of your choice to control a virtual prosthetic hand. You are free to choose any type of surface recording such as electroencephalogram (EEG), electromyogram (EMG), electrooculogram (EOG), etc. Recordings will be made from two channels using a Backyard Brains recording system. You will be provided with MATLAB code that will display the recordings in real-time. *It will be up to you to design decoders to interpret the recordings* into functional control signals for a virtual hand*.* A rubric for this assignment is included below.

**9.2 Objective**

The objective of this lab is to perform a task with a virtual prosthetic hand using two-channel neural recordings, a decoder and a controller. The task is to pick up and put down an object in virtual space.

**9.3 Equipment**

This lab uses the following software and equipment. At least one student per table will need to have a Windows laptop setup ahead of time to perform the recordings and control the virtual hand. Instructions for configuring the software have been provided by the instructor.

* Laptop / Tablet running Windows 64-bit
* MATLAB
* MuJoCo HAPTIX Virtual Environment
* Arduino Uno with two Backyard Brains Muscle SpikerShields

**9.4 Approaches and Exercises**

* Test at least two different types of recordings (e.g., EEG, EMG, EOG, etc.) and in your lab report explain your rationale for the type of recording you chose.
* Test at least two different decoders and explain which one performed better using criteria that you define in your lab report.
* Record and upload short movies of the virtual hand that contrast control strategies with poor versus good performance. In your lab report explain why you think that the combination of recording type and decoder contributed to performance.

**9.5 Respective Contributions**

This is a group exercise at each table (about 2-4 students per group). It will probably be best to divide up the tasks during the lab session among pairs of students, and then work together on the overall goals. For this lab report only it will be permissible to share images/movies. However, every student must write his/her own lab report. At the end of your report provide a short paragraph that explains the relative contributions of each person in the lab group. You can model this after the Author Contributions sections that are now required in most journal articles. For example: “A.D.D. and W.M.G. conception and design of research; A.D.D. performed experiments; A.D.D. analyzed data; A.D.D. and W.M.G. interpreted results of experiments; A.D.D. prepared figures; A.D.D. drafted manuscript; A.D.D. and W.M.G. edited and revised manuscript; A.D.D. and W.M.G. approved final version of manuscript.”

**9.6 Lab Report and Rubric**

**Introduction (22.5%):**

* + Problem statement and objective
  + Describe the patient population your device will help
  + Reference appropriate literature (there is no need for more than 5 references)
  + Provide an overview of your approach

**Methods (22.5%):**

* + Describe how the signals are acquired and processed
  + Use flow diagrams if appropriate & attach relevant code as appendix
  + Define how performance was assessed

**Results (22.5%):**

* + Describe/demonstrate the setup and capabilities of your system
  + Design a test which demonstrates how the hand performed the task. This test could show accuracy, speed, functionality, etc.  There are also several models available for the virtual environment that contain objects you can manipulate.
  + Include still images if appropriate. Upload at least two movies contrasting good/poor performance to Box and include a link to the file.

**Discussion & Conclusions (22.5%):**

* + Discuss the strengths and limitations of your approach
  + Discuss different ways that performance could be measured
  + Suggest improvements and future work.

**Respective Contributions & Appendix (10%)**

* + Author contributions
  + Code or scripts (this portion should be all of your MATLAB code)