

[Build a Kickass Robot Arm: The Perfect Arduino Project for Beginners](#)

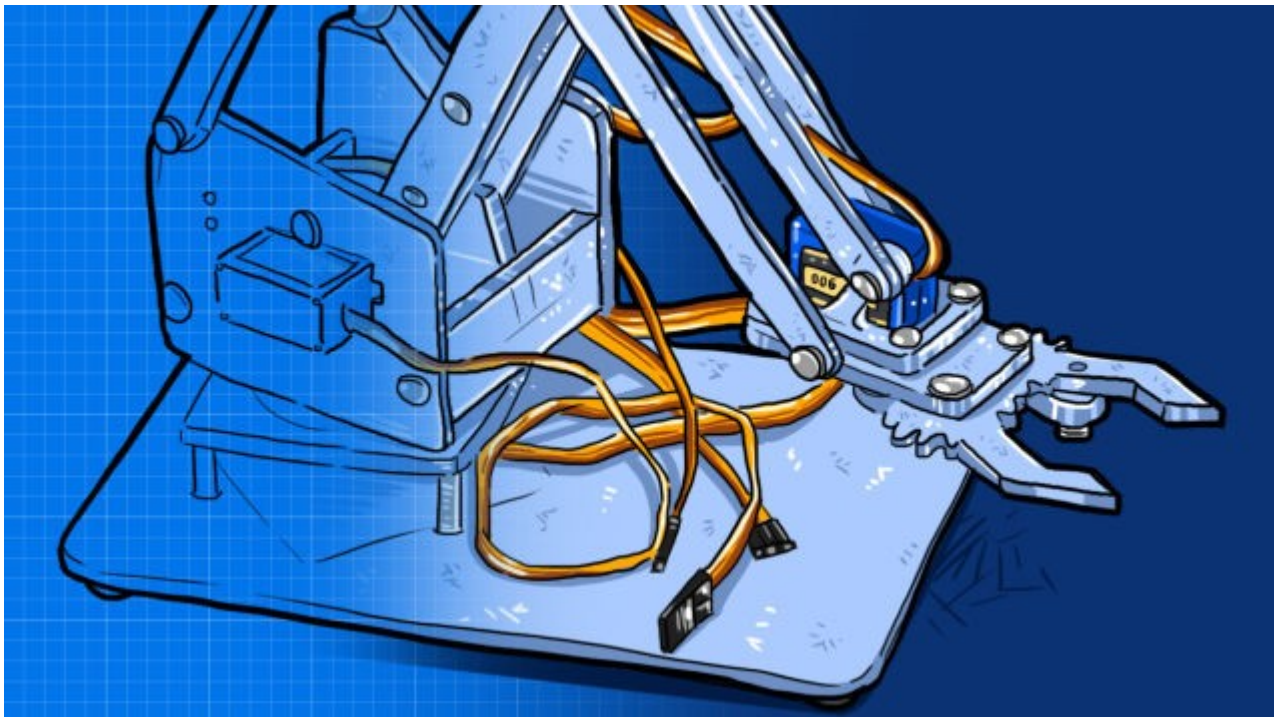
[12](#)

Eric Ravenscraft

[Profile](#)[Follow](#)[Unfollow](#)



[Eric Ravenscraft](#)



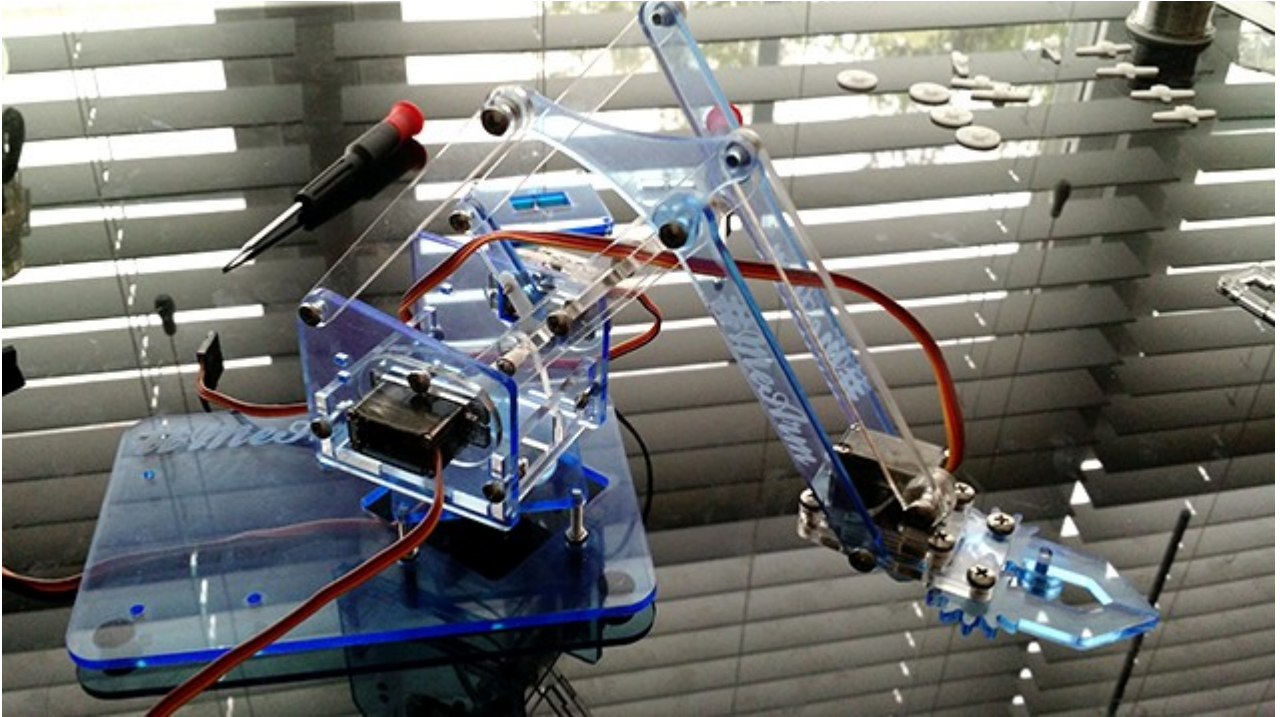
The Arduino is a [cheap, fun way to get into building your own electronics](#). It can also be daunting to get started. Here, we'll show you how to get a start-to-finish Arduino primer with a killer project: building [a sweet robot arm](#).

In this guide, we're going to introduce you to Arduino using the [meArm robot arm](#) project as a guide to a variety of skills. The meArm is an open-source kit with all the parts you need to build a small, Arduino-powered robotic arm. You can order a ready-made set from [stores like Hackaday](#), or [download the plans from Thingiverse](#) and cut them out yourself. You can use a laser cutter, [3D printer](#), or even carve the parts out of wood. The kits are relatively cheap (I got mine for about \$50), so it's pretty

<http://lifehacker.com/build-a-kickass-robot-arm-the-perfect-arduino-project-1700643747>

accessible.

Why a Robot Arm?



Learning any new skill is always a challenge. Arduino can be particularly daunting because you're essentially learning how to build entire electronic devices from scratch. It requires learning several new skills all at once: electricity, breadboarding, coding, sensors, servos, remote controls, assembly, and more.

Any of these skills can be hard to learn on their own. While there are a wealth of tutorials online, it's important to have a [single, overarching goal that you can work towards](#). We also know that your brain learns better when you [spread out your learning over time](#). Having something [you're excited about](#) couldn't hurt.

Building a robot arm is a long-term project that can cover all these needs at once. I've personally been attempting to learn Arduino projects off-and-on for the last nine months or so, and in that time, the robot arm has been the best learning experience I've had, particularly because:

- **It's comprehensive: Finding a first project is hard. Finding one that will actually teach you something is even harder. You can build an LED circuit fairly easily, but then all you have is an LED circuit. Learning to build a robot arm will teach you how to breadboard a circuit, how to program your Arduino, and how to work with moving parts. In the end, you'll have a real, physical thing that does what you program it to do. Not just a proof-of-concept light that turns on when you press a button.**

<http://lifehacker.com/build-a-kickass-robot-arm-the-perfect-arduino-project-1700643747>

- **It's expandable:** If Iron Man's [45 different suits](#) taught us anything, it's that you can always improve a robot. This robot arm kit starts with some basic fundamental skills, but you can build on it with a wide variety of expansions. You can add remote controls (like [Infrared](#) or [Bluetooth](#)), and even learn how to expand your Arduino's capabilities with [extra shields](#). Just ask "What else can I make this do?" and you can find all sorts of new skills to learn without starting a new project over from scratch.
- **It's friggin' cool: Chances are, if you're even reading this far, it's because the thought of having your very own robot excites you. Robots are cool. They can also feel futuristic and inaccessible. If learning is better when you're excited about what you're learning, then it's hard to beat a robot arm to break into the Arduino world.**

All that being said, this doesn't necessarily mean that this should be your *first ever* project. It can be! But if you've never touched a circuit board, it's okay to take it slow. Don't think of the robot arm as your first step. Think of it as your final exam. As soon as you get a good [Arduino starter kit](#), you should try a couple basic things like plugging an LED into a breadboard or controlling it with a button, just to get the hang of it. You can probably skip [the Love-O-Meter project](#), though.

Most importantly, **Google everything**. Remember, this is a long-term project. We're not going to walk you through every step, but we will show you the building blocks you need to get there. Don't expect to start with no experience on Friday and finish with a remote-controlled, sentient robot by Sunday. We'll have plenty of links to guides throughout this article, and we fully expect that you'll leave here, follow those guides for a few hours, and come back. Think of this less like a step-by-step manual and more like a map. If you get a little lost along the way, don't be afraid to stop and ask for directions.

What You'll Need



This guide will be divided into two main sections. The first will be getting the basic robot arm built and operational. The second will show you some optional projects that you can use to expand its capabilities. To get through the first section, here's what you'll need:

- **An Arduino starter kit:** Most Arduino starter kits will include the basic components you'll need for this project (as well as many others). You'll need an Arduino (we'll use an [Uno R3](#)), various lengths of wire, a USB cable to connect to your computer, and a [breadboard](#), and a [potentiometer](#), which can be used as a knob for controlling your robot later on. Adafruit has a selection of a few [starter kits here](#) for varying price ranges. [This kit in particular](#) includes everything listed above for \$65. You may also need a 470uf capacitor later on, which you can pick up at RadioShack for [dirt cheap](#).
- **A meArm kit:** For simplicity's sake, you can buy an entire kit [here](#). This includes everything you need to build the arm itself. Optionally, you can download the plans [here](#) and make them yourself. The plans do require very precise sizes, so only use this option if you have access to tools that can cut (or 3D print) the pieces correctly.
- **An Arduino IDE:** An IDE (or [Integrated Development Environment](#)) is the program you'll use to write and upload software—called “sketches”—to your Arduino. You can download the official Arduino IDE [here](#). In my personal experience, I found that [previously-covered CodeBender](#) is an excellent, browser-based alternative that stores your sketches online for easy access.

<http://lifehacker.com/build-a-kickass-robot-arm-the-perfect-arduino-project-1700643747>

These will get you started and cover the basics. It's also a lot to buy all at once, so don't feel bad if you don't want to go any further than this. Over time, you can add more tools and equipment to your arsenal.

What This Project Entails

We'll assume that you've acquired everything in the first bulleted list in the section above and you're ready to put your robot together. We won't detail every single step when other, more official guides already do, but we'll guide you through the different phases of the project. You can take this at whatever pace you're comfortable with, but we'll break it up into chunks you can tackle across multiple weekends.

Phase One: Construction



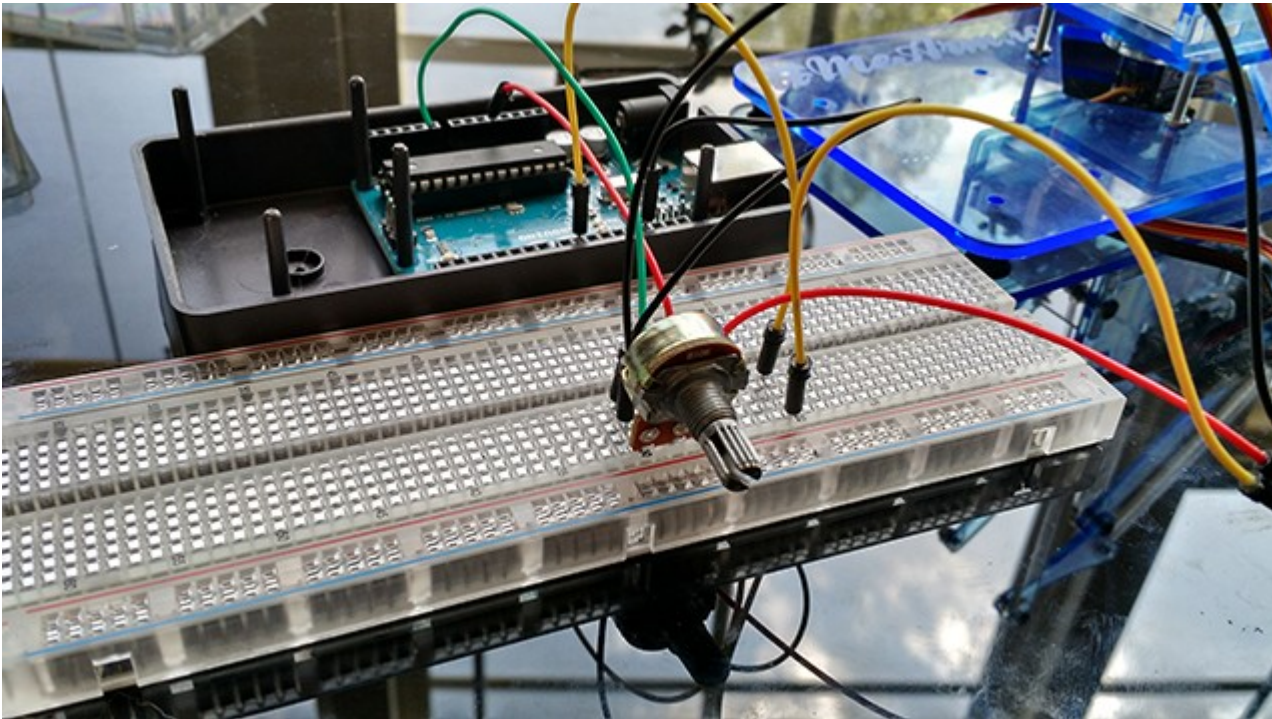
What It Entails: At this stage, you're going to assemble your robot arm. It won't do anything yet, but it will look cool. The company behind the kit has [detailed instructions here](#). Your kit should come with a set of various plastic pieces, some screws, and four servos. If you've never used a servo before, it's a small, low-power motor that will power your robot's movement. This kit uses one in the base, two on either side of the arm, and one in the gripper. If you've ever built a piece of IKEA furniture, this shouldn't be too complicated. Just follow the instructions **precisely**. The robot is more delicate than your coffee table, and over tightening a screw here, or using the wrong piece there can cause you headaches. Fortunately, the instructions above are very detailed and will warn you repeatedly before you can do something that will mess you up, so you're in good hands.

<http://lifehacker.com/build-a-kickass-robot-arm-the-perfect-arduino-project-1700643747>

What You'll Learn: Personally, this is my favorite part because you learn something that most project tutorials neglect: *how to build the thing*. Many projects show you a concept while attached to a breadboard, but never move it to something real. Here, you'll learn how to attach servos to working parts on a finished project. You'll also learn the delicate art of working with tiny moving parts.

Time Required: The construction portion here can be done in just a few hours. However, I advise letting your work sink in for a bit. If you've never worked with robotics before, this is a good time to examine how your bot was put together, how it can move, and start thinking about how you might mount your Arduino later on. The next step can start to get complicated, so don't rush into it. You can manually move the parts of your robot arm **gently** to play around with it. Don't force it too much, though, as you could damage the servos.

Phase Two: Breadboarding



What It Entails: The next step is to connect one of your servos to your Arduino. You'll accomplish this with the use of [a breadboard](#). A breadboard is a simple tool that allows to prototype electronic circuits before assembling them completely, no soldering required. Adafruit has [an excellent tutorial here](#) that will walk you through the steps of connecting your servo directly to the Arduino, as well as adding a potentiometer in later steps, which you can use as a knob to manually control movement.

If that paragraph was a little overwhelming, then this is a good time to back up and learn how breadboarding works. Sparkfun has [an excellent guide here](#) that explains how to use a breadboard and what you can do with it. Tutsplus has [a great tutorial](#) on how to connect an LED to a power source and add a button. Take time to assemble this and understand the circuit you just assembled. Once you've

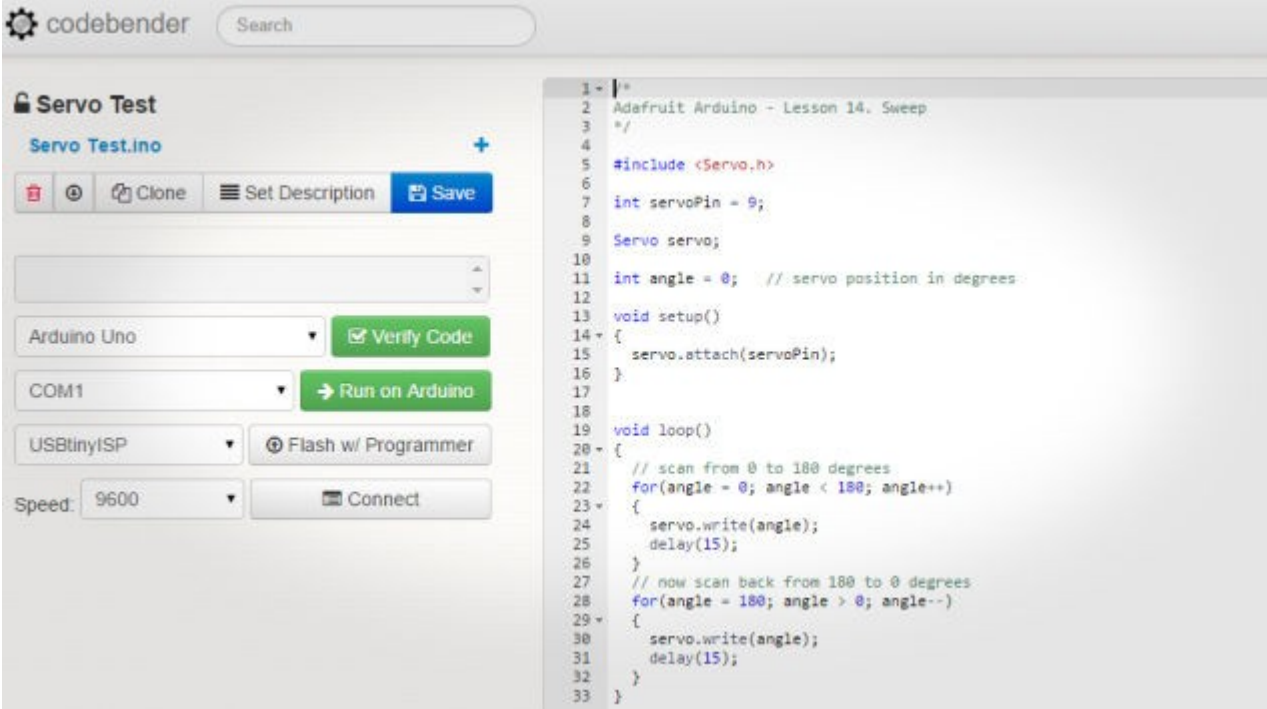
<http://lifehacker.com/build-a-kickass-robot-arm-the-perfect-arduino-project-1700643747>

got a grasp on how electricity flows through a simple circuit, you should be ready to connect one of your robot's servos. From personal experience, this part can seem daunting. Assembling the parts and following instructions is easy, though. Understanding how it all works is the hard part, but it just takes time.

What You'll Learn: Breadboarding is the foundation of most hobbyist electronics prototyping. If you followed all of the guides above, you'll learn how to connect LEDs, buttons, resistors, potentiometers, and servos to either a power source or Arduino. With just those components, you can already make a lot of fun stuff. Once you've got the basics down, it's easier to build on it by learning about different types of components, how they work, and how to integrated them into your projects (more on some ways to do that in the expansion section below).

Time Required: If you're already familiar with breadboarding, connecting the servo should take about five minutes. *However*, if you've never touched electronics before, give yourself a day or two to read the guides above, fiddle around with various circuits, and get a feel for how they work. I'd even advise taking a week to process the lessons you've learned. Breadboarding circuits is simple, but it can be a difficult concept to grasp. And it's not something you want to rush through, especially given how complicated the next section can get.

Phase Three: Programming



The screenshot shows the CodeBender web interface. On the left, there's a sidebar with the title "Servo Test" and a file named "Servo Test.ino". Below the title are several controls: a "Clone" button, a "Set Description" button, and a "Save" button. There are also dropdown menus for "Arduino Uno", "COM1", and "USBtinyISP", along with a "Speed" dropdown set to "9600". A "Connect" button is at the bottom of these controls. On the right, the main area displays the Arduino sketch code. The code is as follows:

```
1 - /*
2 Adafruit Arduino - Lesson 14. Sweep
3 */
4 #include <Servo.h>
5
6 int servoPin = 9;
7
8 Servo servo;
9
10 int angle = 0; // servo position in degrees
11
12 void setup()
13 {
14   servo.attach(servoPin);
15 }
16
17
18 void loop()
19 {
20   // scan from 0 to 180 degrees
21   for(angle = 0; angle < 180; angle++)
22   {
23     servo.write(angle);
24     delay(15);
25   }
26   // now scan back from 180 to 0 degrees
27   for(angle = 180; angle > 0; angle--)
28   {
29     servo.write(angle);
30     delay(15);
31   }
32 }
33 }
```

What It Entails: Once you've got everything hooked up, it's time to turn it on. For that, you'll need to have your Arduino IDE setup and plugged into your board. If you want to use [CodeBender](#) like I do, you can follow the [Getting Started guide here](#). Alternatively, you can follow Adafruit's [guide to the](#)

<http://lifelife.com/build-a-kickass-robot-arm-the-perfect-arduino-project-1700643747>

[official IDE here](#).

Once your environment is set up, you'll also start programming the thing. Adafruit's guide has [a simple servo sketch](#) you can use to make your robot move. I'd advise using the base servo (the one on the bottom) for this, as it's the only one on your robot that has the full 180 degree motion. You could damage some of the other servos by forcing them to beyond their physical limits if you try this sketch with the other servos. However, once you understand how this sketch works, you can try modifying it to work with the others!

What You'll Learn: This is the phase where it all comes together. You'll learn a bit about how servo movement works, and *a lot* about how to program an Arduino. If you've never dabbled in programming before, you can drop the sweep sketch into the IDE and it will work, but I'd advise checking out some of our previous guides on [how to learn to code](#). The Arduino language shares a lot of syntax with C/C++ and Java, so if you have any experience with those, you should feel comfortable. You can also check out the [Arduino reference library here](#).

Time Required: Even if you have some programming experience, I'd advise taking another weekend to learn how to set up the Arduino IDE. Learning to code is a lifetime skill, so don't be afraid to work on this phase for a few weeks. You can build on this with the [knob sketch Adafruit provided](#), which will allow you to manually control your robot. Don't be afraid to mess up. You can also experiment with some basic logic structures at this phase.

Congratulations! You Just Made a Robot

If you made it through all of that, then you just learned a bunch of skills in one, long project. When I first put together this robot, I found it was surprisingly simple, despite being an introduction to a lot of complex topics. Once you make it to the end, though, most electronics projects—like [the kind we regularly feature](#)—don't look so scary anymore.

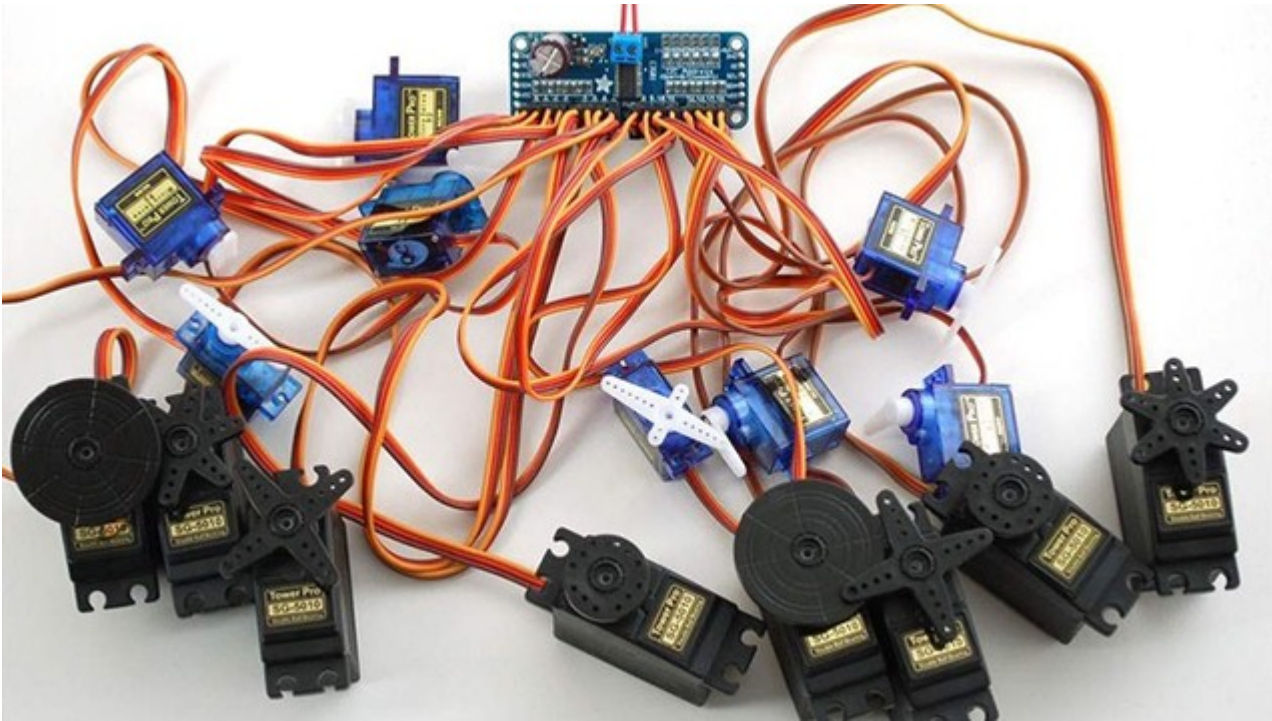
From here, you can start expanding on what you have. If you feel like you just barely got through this, try simple additions like [adding an LED](#) to indicate when the motor is turning, or [a button](#) to turn the movement on and off. Play around with the software a bit and see how it responds. If you screw up some software and overturn a servo, you can order [super cheap replacements online](#).

Build On Your Knowledge with These Expansion Projects

You've built a robot. Now what? Well, assuming it hasn't [turned sentient and tried to kill humanity](#), there are a number of projects you can pursue that can build on your existing project one piece at a time. We won't go over every detail, but we'll give you some links to get you started:

<http://lifehacker.com/build-a-kickass-robot-arm-the-perfect-arduino-project-1700643747>

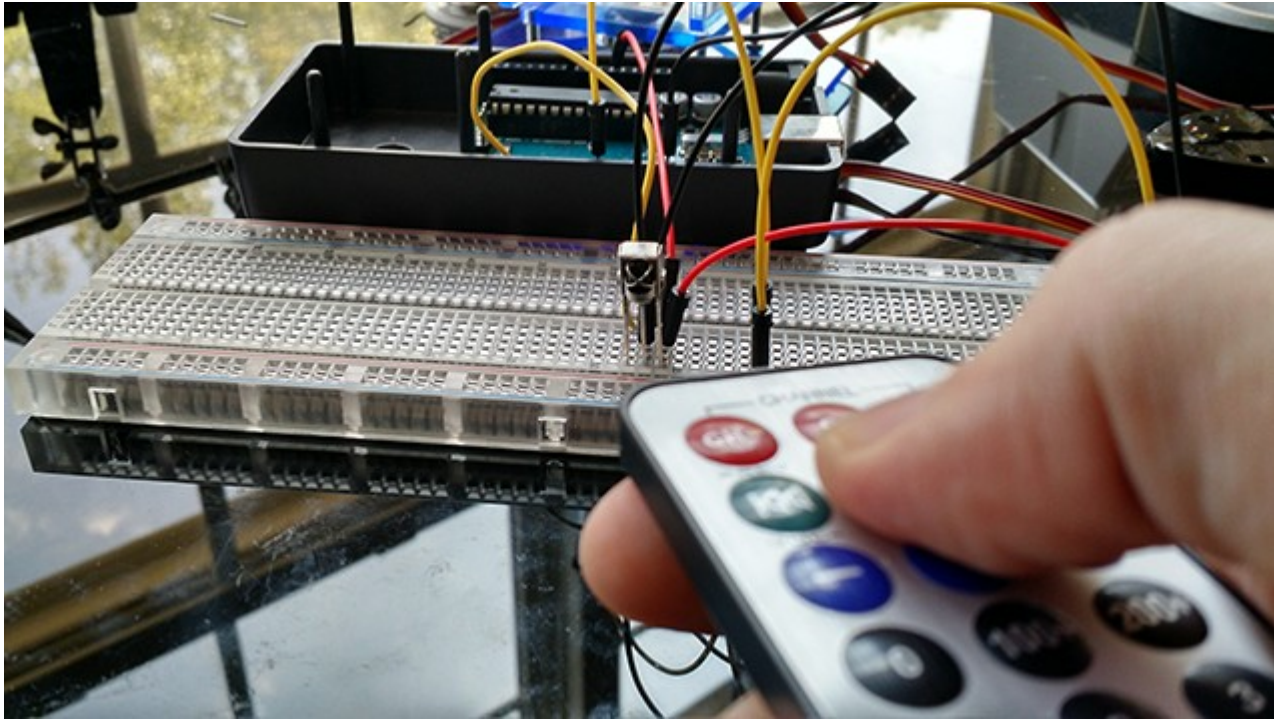
Control Multiple Servos at Once



For my build, I ordered [this microcontroller](#) which can control up to sixteen servos on its own (which, for those counting at home, would add up to four meArm robots...sweet). The kit is **not** pre-assembled, which means it would require some soldering work. You can get other controllers [like this one](#) that are pre-assembled, but many cost more and do less. Personally, I think that a \$15 controller is a decent way to practice soldering without risking too much if you ruin it, but if you don't want to take that chance, practice by soldering a couple wires together first. Here are some guides that can walk you through the process:

- [Adafruit 16-Channel Servo Driver with Arduino](#)
- [Adafruit Guide To Excellent Soldering](#)
- [How Do Servos Work?](#)

Add an Infrared Remote Control



Infrared (IR) remote controls are deceptively simple (and can be a handy addition to nearly any Arduino project). All you need is [a sensor](#) and [a remote](#). The remote will send codes to your Arduino, which you can then use to trigger commands. In this case, you could program your robot to start moving, stop moving, or to go to a certain pre-programmed position. There's already a bunch of pretty great code out there you can use in your projects. For fun, you can even read the codes in your TV remote, if you want to, say, make your robot come alive any time someone changes the channel. Here are some resources you'll need to get started:

- [How to Use IR Remotes with Arduino](#)
- [Arduino Infrared Remote tutorial](#)
- [shirriff/Arduino-IRremote library](#)

<http://lifehacker.com/build-a-kickass-robot-arm-the-perfect-arduino-project-1700643747>

Use a Wii Nunchuk to Control Your Death Machine



Alright, you want to get really crazy? Check out the above video showing a the robot arm—the same model you built!—being controlled by a Wii Nunchuk. It uses [this breakout adapter](#) (which you can plug a nunchuk directly into) and provides full joystick control, which means that you can make it move in any direction you want, like a futuristic puppet. If you've reached the point where you're ready to undertake this project using this guide, you're probably back for the dozenth time, so welcome back. This is the most advanced add-on we've included in this article, so don't feel bad if it's a bit over your head. It is, however, really cool. Here are some resources for further reading:

- [Joystick control of Phenoptix meArm with Inverse Kinematics](#)
- [Wii Nunchuk Breakout Adapter](#)
- [Inverse kinematics control library for Phenoptix meArm](#)

As you can tell, the robot arm project covers a ton of concepts and skills in the Arduino hacking scene. If you can make it through this project without getting overwhelmed or giving up, you can probably tackle most of the Arduino projects [we feature on a regular basis](#). Getting started can seem intimidating, but if you add to your knowledge and experience piece by piece, you can build something pretty awesome.