**MEMO OF TRANSMITAL:**

**TO:** Dr. Rivers

**FROM:** Mark Betters

**DATE:** 25 January 2017

**SUB:** Final Report

I would like to present you with this Final Report as it was described and outlined in class; may its contents be to your liking and bear your utmost approval.

This report discusses the designs and implementations of the communication system, electrical signals and power, artificial intelligence, and exoskeleton of a wall-climbing robotic spider that is being built by UD Engineering Capstone team VENOM (“Vertically Engineered Octonary Machine“). Since this is a Capstone project, the designs and design implementations described in this report are constrained by a $200 budget, a May 2017 completion date, and the hours of labor that can be reasonably expected from VENOM’s six team members.

It is my honor to thank Dr. Chase Cotton for managing the Capstone team in Fall 2016 and Spring 2017 and for interviewing for this report. I also thank my fellow Capstone team members- Jon Garcia, Marcus Gula, Michael Matimu, Nathan Walker, and Daniel Wang- for their persistent work on this project.

Thank you to Dr. Rivers for this assignment, since it gave me the opportunity to formalize a robot’s design and implementation. This report will be put to good use for Capstone team VENOM’s project.

**SPECIFICATIONS AND PROCESSES FOR A WALL-CLIMBING ROBOT SPIDER**

Prepared by: Mark Betters

Prepared for: Dr. Rivers

Final Due Date: February 3 2017

# ABSTRACT

The design and implementation of a cheap and effective wall-climbing robot is a welcome innovation in the robotics market and community. This report is organized by first having a section on the exoskeleton and then sections ordered by the flow of information in the robot’s communication system components, which are the Cloud9 website, the Raspberry Pi micro-processor, and the Arduino micro-controller. The flow of information starts with the operator selecting and submitting a command on the VENOM website, which is then read by a Raspberry Pi micro-processor, which then passes the command to an Arduino micro-controller, which then translates the command into a looping series of polls for latest distance sensor values and output signals to the servos that control the rotation and elevation of the robot’s legs. The commands that the operator can pick range from simple (“walk forward”) to complex (“explore the room”).

# TABLE OF CONTENTS

**ABSTRACT ...........................................................................................................iii**

**TABLE OF CONTENTS ......................................................................................iii**

**LIST OF ILLUSTRATIONS ................................................................................iv**

**INTRODUCTION ..................................................................................................1**

**Purpose, Scope, & Limitations ........................................................................................1**

**Methods of Research .........................................................................................................1**

**Report Organization ........................................................................................................1**

**INTERVIEW WITH DR. CHASE COTTON .....................................................1**

**EXOSKELETON ....................................................................................................3**

**Prototyping with Balsa Wood ..........................................................................................3**

**Torso, Legs, Distance Sensors, & Servos ........................................................................3**

**3D Printing ........................................................................................................................5**

STL Models ............................................................................................................5

Materials .................................................................................................................5

Printing Processes ...................................................................................................5

**OVERALL COMMUNICATION SYSTEM .......................................................6**

**CLOUD9 WEB SERVER ......................................................................................6**

**Cost .....................................................................................................................................6**

**User Interface ....................................................................................................................6**

**Python Web Server Script ................................................................................................7**

**RASPBERRY PI .....................................................................................................8**

**Cost .....................................................................................................................................8**

**Electrical Power ................................................................................................................8**

**USB WiFi Dongle ..............................................................................................................8**

**Initialization Script ...........................................................................................................8**

**Command-Relaying Python Script .................................................................................9**

**ARDUINO ...............................................................................................................9**

**Cost ....................................................................................................................................9**

**Electrical Power ................................................................................................................9**

**Digital, Analog, & PWM Pins ........................................................................................10**

**Distance Sensor Input Signals & “Polling” ..................................................................11**

**Servo Control Signals .....................................................................................................11**

**Receiving Latest Command from Raspberry Pi ..........................................................11**

**Translating Latest Command into Sensor Polls & Servo PWM Signals ...................11**

**CONCLUSIONS & RECOMMENDATIONS ...................................................12**

**APPENDIX …........................................................................................................13**

**Appendix A: Works Cited ..............................................................................................13**

**Appendix B: Amazon Product Links .............................................................................13**

# LIST OF ILLUSTRATIONS

**Figure 1: Final Robot Spider Balsa Wood Prototype ................................................................3**

**Figure 2: HC-SR04 Distance Sensor ...........................................................................................4**

**Figure 3: Spider in Flexed Position .............................................................................................4**

**Figure 4: Example of STL ASCII File Contents ........................................................................5**

**Figure 5: Overall Communication System .................................................................................6**

**Figure 6: Cloud9 User Interface ..................................................................................................7**

**Figure 7: Arduino & Servos Power Source ..............................................................................10**

**Figure 8: Arduino Teensy 3.2 Pin Diagrams ............................................................................10**

# INTRODUCTION

## Purpose, Scope, & Limitations

This report will give UD Capstone team VENOM the means to communicate the design and implementation of a wall-climbing robot spider. The technical steps laid out in this report will serve as a rough guide for the reader to make a robot with similar capabilities and under the same constraints, which are primarily a $200 parts budget and a short time to construct the robot. The sections of this report will start with the exoskeleton and then each cover a communication component of the robot, ordered by command transmission. Each section will describe what the component is, its cost, and how it works in selecting, relaying, translating, or acting out a command. Since there are few known examples of cheap wall-climbing robots, there are few sources that cover the whole integrated design of such a system. Therefore, the information in more specifically focused sources had to be adapted to serve the designs of specific components in this report.

## Methods of Research

An interview with Dr. Chase Cotton proved foundational to the planning of this report. The interview was conducted in his office at UD and lasted about one hour. Questions were answered with anecdotes, general design guidelines, and specific suggestions for technical specifications.

Online sources used include forums, software module documentations, electronic component data sheets, images, and tutorial articles. These sources are all cited in Appendix A. Links for products bought on Amazon are listed in Appendix B.

## Report Organization

This report is organized by first a section on the exoskeleton and then electronic components, ordered by the path a command takes from the human operator to the final resulting actuation of the robot’s exoskeleton.

# INTERVIEW WITH DR. CHASE COTTON

The interview that follows is a discussion between me and Dr. Chase Cotton about the key design and economic considerations for a wall-climbing robot. I started by asking him the central question, “How can a robot climb a wall?” There are two main methods that he gave, suction and adhesion.

For the robot to stick to a wall using suction, it needs two components, suction cups and a suction motor that sucks air out of the cups through plastic tubes. The suction of the cups can be increased by increasing their opening’s surface area, taking a longer time to suck the air out, or pumping the suction cups with special gases. Therefore, the primary advantage of suction cups is that they can be made arbitrarily strong. However, there are four drawbacks to using suction. First, the suction motor, while small enough to fit on the robot’s exoskeleton, is somewhat heavy

(several ounces), and would therefore weigh the robot down and counteract its own efforts to keep the robot on the wall. Second, suction motors and suction cups cost more than adhesives. (However, 3D-printing the cups would slightly mitigate the cost.) Third, suction cups only work well on smooth surfaces. This is because on rough surfaces there are small imperfections that prevent the suction cups from fully sealing when they are pressed against the surface. Fourth, the robot’s micro-controller would have to keep track of when to signal the suction motor to pump and suck air, which adds logical complexity to the micro-controller’s program.

Adhesion is an alternative to suction that does not have any of suction’s drawbacks. Adhesives are light, cheap, work well on smooth and rough surfaces, and require no additional logical complexity in the micro-controller’s program. However, there are two drawbacks to adhesives that suction cups do not have. First, adhesives tend to leave a potentially unwanted residue on surfaces. Second, adhesives wear out after repeated use and would have to be continually replaced by hand, unless there were an automatic adhesive dispenser built into the legs of the robots, but this would add logical complexity to the micro-controller’s program.

The next question I asked Dr. Cotton was, “Is a cheap wall-climbing robot spider a viable product?” He replied that glass building maintenance could be a niche market for such a product. Current methods of maintaining glass buildings involve robotic apparatuses that consist of cranes installed on the roofs of buildings that elevate robots to do cleaning and inspection. These systems are expensive and tend to constrain the architectural design of the building, since the roof must accommodate a crane. A cheaper and less architecturally limiting alternative is drones. However, legal drone operations require a licensed human operator, which is costly. A wall-climbing robot spider is cheap, does not constrain the building’s architectural design, and has the legal right to act autonomously.

Following this question, I asked Dr. Cotton, “What additional features would be beneficial for this robot spider to have?” He replied that the exoskeleton should include a generic mount to serve a wide range of components. Most notably, the mount should be able to accommodate a camera, so that the robot could be used to send live video footage back to the operator.

My next question was, “Which is better, controlling the robot via a game controller and Bluetooth receiver over Bluetooth or via a Cloud9 website and Raspberry Pi over WiFi?” Dr. Cotton replied that Bluetooth is far less reliable than WiFi over long distances, therefore making WiFi the superior protocol. Another consideration, he said, was that the robot should act autonomously whenever it has not received a new command in a long time.

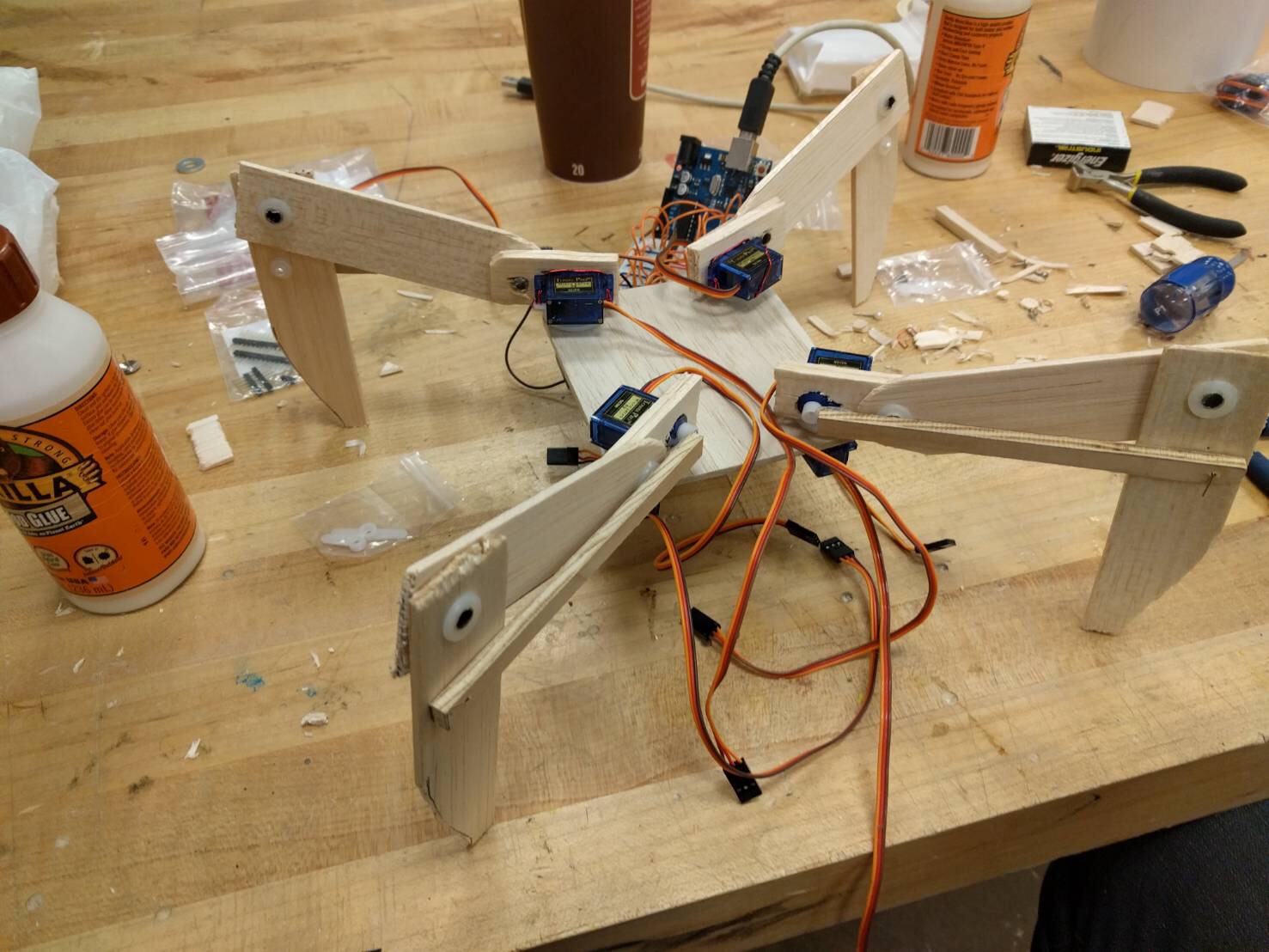
The final question I asked was, “How has robotics technology changed over your career?” Dr. Cotton answered that when he was working on one of the first industrial robots, a manufacturing arm called “Unimate,” the robot’s movements had to be programmed statically, meaning the arm did the same motion over and over. Likewise, roboticist had to program the exact motions of other robots over the next several years. Now, however, with computer vision technology, robots’ movements are programmed dynamically, meaning robots can see changes in the environment and adapt their movements to work well with those changes.

**EXOSKELETON**

**Prototyping with Balsa Wood**

To come up with a good design for a robot, team VENOM constructed several physical prototypes out of balsa wood. The final prototype is shown in Figure 1 below.

Figure 1: Final Robot Spider Balsa Wood Prototype

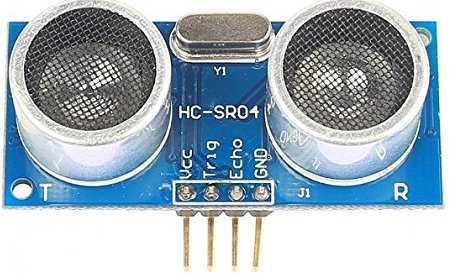


(Source: Photo taken by the author)

**Torso, Legs, Distance Sensors, & Servos**

As shown in Figure 1, the robot has a central “torso” with enough space in the middle for the Raspberry Pi, Arduino, and power sources. There are four “legs” attached to the torso. The “hip joint” connects a leg’s “thigh” to the torso and the “knee joint” connects a leg’s thigh to its “shin.” At the tip of the shins are “feet,” which have adhesives. The hip and knee joints of the balsa wood prototype are free to rotate, but the final, 3D-printed robot will have a spring between the thigh and shin that prevents the shin from extending too far.

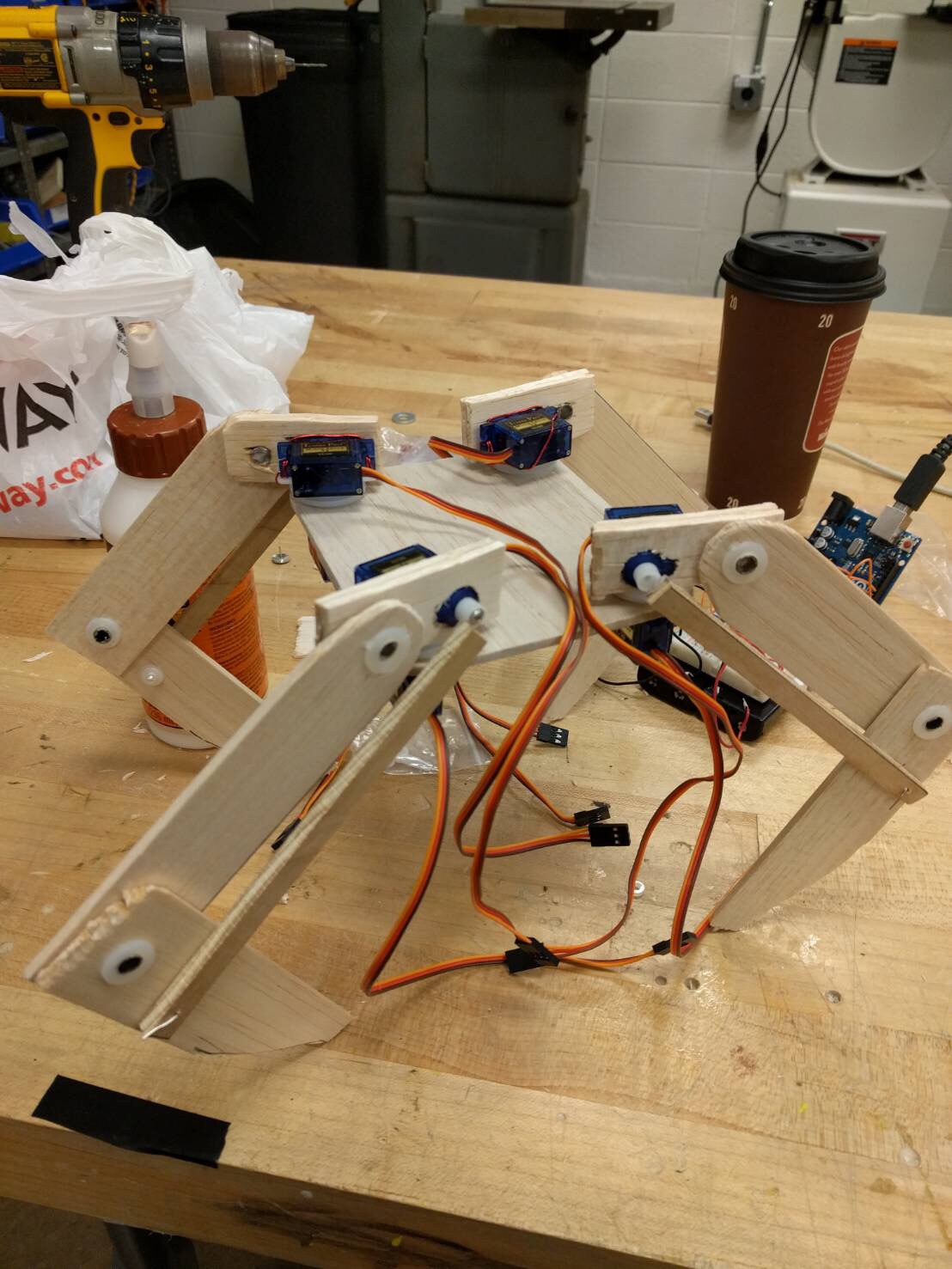
The four torso spaces between the legs are each wide enough for one of the four distance sensors. The distance sensors are not shown in Figure 1, but one is shown in Figure 2 on page 4. Since the Arduino handles inputs from the distance sensors, their signal specifications are discussed in the “Arduino” section. The distance sensor can be bought for $12.00 on Amazon.

Figure 2: HC-SR04 Distance Sensor

(Source: <https://images-na.ssl-images-amazon.com/images/I/51GZZ5EU9PL.jpg>)

As shown in Figure 1 on page 3, there are four servos (tiny plastic motors) on the four corners of the top of the torso. These are the “flexor servos,” so named because they rotate “leg shafts” that are connected to the shins, thereby elevating the whole leg. An upward elevation causes both the thigh to flex upward and the shin to extend away from the torso. A downward elevation causes the thigh to flex downward and the shin to flex toward the torso. Changing where the leg shaft is attached to the shin will change how extreme the motion is. Figure 3 shows the spider in a flexed position.

Figure 3: Spider in Flexed Position



(Source: Photo taken by the author)

There are another four servos, only one of which is visible in Figure 1 on page 3, that are underneath the four corners of the torso. These are the “abductor servos,” so named because they abduct the leg by rotating the flexor servo’s mounting.

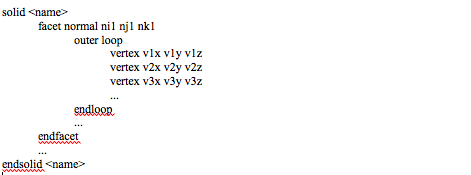
The servos are SG90 TowerPro servos, which can be bought for $6.00 on Amazon. The servos have 180 degrees of motion. Since every leg has two servos acting as a flexor and an abductor, the legs can move in two dimensions of freedom.

**3D Printing**

**STL Models**

STL (“STereoLithography”) is a file format that is used to define 3D solids. STL files are ASCII text files with contents that look like Figure 4.

Figure 4: Example of STL ASCII File Contents



(Source: Screenshotted by the author)

The named solid in the STL file can have one or more facets. Every facet is described by a normal vector and one or more loops. Every loop has three or more vertices. STL files do not enforce connectedness and closed-ness, so different 3D modeling programs may resolve the same STL file into different solids if there is any ambiguity.

**Materials**

While balsa wood is an adequate material for prototyping the robot, due to how easy it is to carve, it is too brittle to be used for the final iteration of the exoskeleton. Therefore, the robot parts were printed with plastic, which is less brittle.

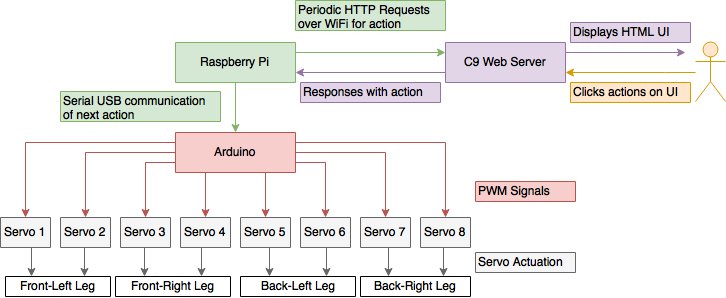
**Printing Process**

Printing an object from an STL file is as simple as going to [**https://www.shapeways.com**](https://www.shapeways.com), uploading the STL file, and paying for the materials and shipping.

**OVERALL COMMUNICATION SYSTEM**

The overall communication system is displayed in Figure 5 below. The purpose of the communication system is for the operator to be able to select the next command, for that command to be transmitted to the robot, and for the robot to translate the command into servo signals. The next three sections will discuss, in order, the Cloud9 (C9) web server, Raspberry Pi, and Arduino.

Figure 5: Overall Communication System

TODO: Show the sensors in the figure.

(Source: Made and screenshotted in draw.io by the author)

**CLOUD9 WEB SERVER**

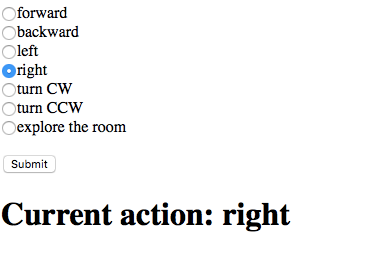
**Cost**

Cloud9 web servers are free to host. The drawback is that they are operational for a limited time and can only handle a small number of web requests.

**User Interface**

To see the user interface for selecting commands for the robot, as shown in Figure 6 on page 7, the operator must go to the web server at the Cloud9 URL <https://cpeg498-spider-mbetters.c9users.io>. When the operator makes this web request, the user interface is sent to the operator’s computer as an HTML file. The operator’s web browser will read the HTML file and display the user interface. The user interface is simply a list of radio-button commands and a “Submit” button. Only one command can be selected at a time. A new command is issued by selecting a command and then clicking the “Submit” button.

Figure 6: Cloud9 User Interface

(Source: Screenshot taken by the author)

**Python Web Server Script**

The Cloud9 website is served by a Python web server script. This script uses the web.py web server framework, as described in the “Python Web.py Module Documentation.” This framework allows programmers to construct a web server from simple data structures and two API calls, a call to construct an application from the URL mapping data structure (discussed in the next paragraph) and a call to run the application as a web server.

The URL mapping data structure is a tuple that maps URL patterns, which are strings of characters that represent a family of possible URLs, to the names of request-handler classes, which are blocks of code that respond to different kinds of web requests that are made to the given URL. Evenly indexed elements of the tuple (zeroth, second, fourth, etc.) are URL patterns and oddly indexed elements of the tuple (first, third, fifth, etc.) are the names of the request-handler classes that handle web requests sent to the URL pattern in the previous even element.

The other data structures are request-handler classes. These are named blocks of code that define sub-blocks of code for handling different kinds of web requests, primarily GET and POST requests. There are a few requests that need to be supported by the web server.

The first is “GET user interface page,” which is triggered when the client requests <https://cpeg498-spider-mbetters.c9users.io> and handled by the server reading the user interface HTML file and sending it to the client.

The second is “GET the JavaScript file for the user interface page,” which is triggered when the client requests <https://cpeg498-spider-mbetters.c9users.io>/js/index.js and handled by the server reading the user interface JavaScript file index.js and sending it to the client.

The third is “POST new command,” which is triggered by the client selecting a command on the user interface and then clicking the “Submit” button and handled by the server storing the new command in the command file.

The fourth is “GET latest command,” which is triggered when the Raspberry Pi requests <https://cpeg498-spider-mbetters.c9users.io>/info and handled by the server reading the latest command from the command file and then sending it to the Raspberry Pi.

# RASPBERRY PI

**Cost**

At the time of this report’s writing, the cost of a Raspberry Pi 3 Model B Motherboard is $59.99, as shown on Amazon. This is the most expensive component of the spider, and was chosen simply because it has a built-in network stack, therefore making web requests over WiFi easy to program.

**Electrical Power**

The Raspberry Pi requires that its power port is connected to a 5 volt, 2.5 ampere power supply. This power supply can be given by a battery pack. Since an AA battery provides 1.5V, four of them connected in series (in a row with positive terminals connected to negative terminals) provides 6V, which is close enough to 5V that it will power the Raspberry Pi. In order to regulate the current provided by the four batteries, a UBEC (Universal Battery Elimination Circuit) component can be purchased for $26.45 from. The UBEC component is connected to the negative terminal of the first battery in the series and the positive terminal of the last battery in the series. Then, the terminals of the UBEC component are connected to the Raspberry Pi’s power port.

**USB WiFi Dongle**

The Raspberry Pi is connected to WiFi via a USB WiFi dongle. A USB WiFi dongle can be purchased for $7.95 from Amazon. Upon powering on and booting up, the Raspberry Pi will attempt to connect to any WiFi networks listed in its network interface file, as described in an article on the Adafruit website called “Setting Up WiFi with Occidentalis.”

**Initialization Script**

When the Raspberry Pi boots up, it executes an initialization script, called “init.d.” This script is discussed in more detail on a topic on the Raspberry Pi forum called “Run Script at Boot.”

The script is written in Bash, a computer language that is explored in greater depth on the “Advanced Bash Scripting Guide.” The initialization script changes the directory to the Desktop, then it executes another script on the Desktop. This other script is a command-relaying script, which is written in Python.

**Command-Relaying Python Script**

The command-relaying Python script is an infinite loop that continually makes web GET requests, using the Requests module (documented in the “Python Requests Module Documentation”) to the URL <https://cpeg498-spider-mbetters.c9users.io>/info to get the latest command from the website. Once it gets the latest command, it uses the Pyserial module (documented in the “Python PySerial Module Documentation”) to write the command to the USB interface, which sends the command as a series of signals over a USB wire to the Arduino micro-controller. (Since the Raspberry Pi has a USB 2.0 port and the Arduino micro-controller model used for this robot has a Micro USB port, The USB wire mentioned here is a USB 2.0 to Micro USB cable, which can be purchased on Amazon for $4.99.)

There are four USB interfaces on the Raspberry Pi. One of the interfaces is used for the USB WiFi dongle and another one is used for the USB 2.0 to Micro USB cable. The interface that Pyserial must write to depends on which port the USB 2.0 to Micro USB cable is plugged into.

**ARDUINO**

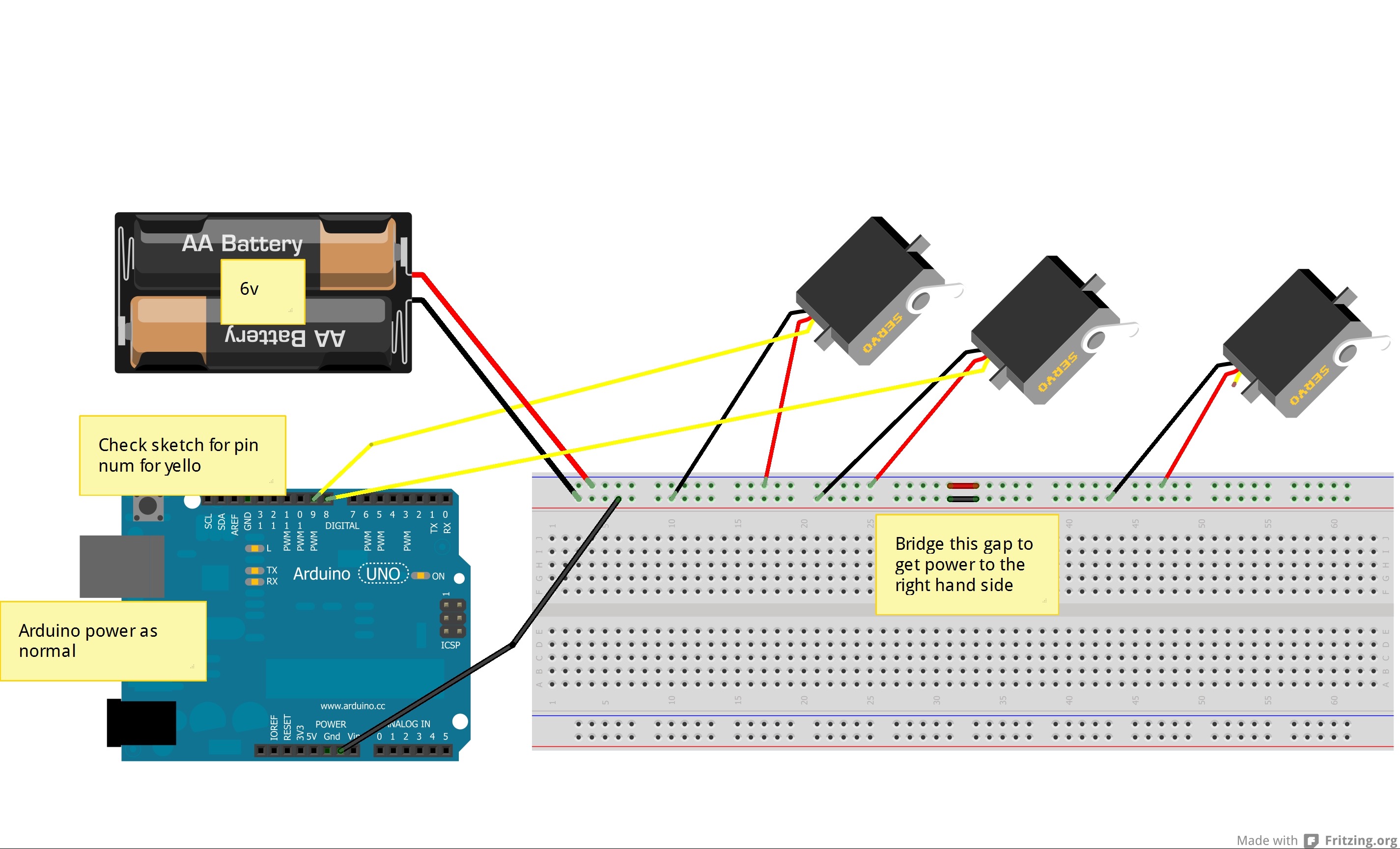
**Cost**

At the time of this report’s writing, the cost of an Arduino Teensy micro-controller is $24.45, as shown on Amazon.

**Electrical Power**

There are two key considerations for powering the Arduino. First, it must receive 3.6 to 6.0 volts. Second, the electrical ground for the Arduino must be the same as the electrical ground for the servos, which must receive 5 volts. Since 5 volts is between 3.6 volts and 6.0 volts, it would save money and weight to have the Arduino and servos share a power source, as illustrated in Figure 7 on page 10.

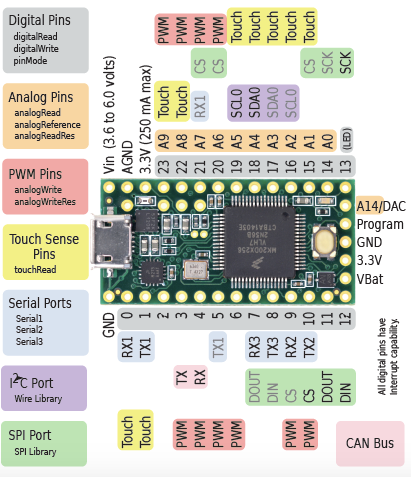
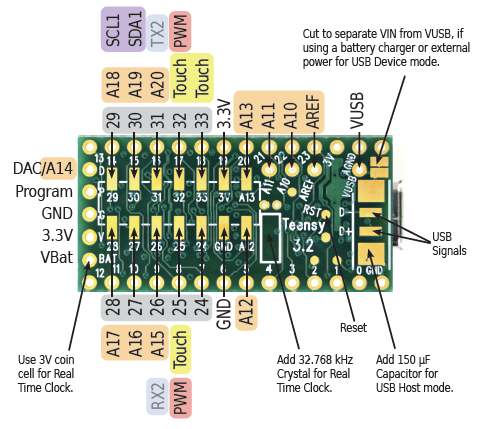
Figure 7: Arduino & Servos Power Source

(Source: <https://forum.arduino.cc/index.php?topic=146230.0)>

**Digital, Analog, & PWM Pins**

The Arduino Teensy has many 21 analog input pins, 34 digital I/O pins, and 12 PWM (pulse width modulation) output pins. The analog input pins and digital I/O pins will be used for the robot’s sensors and 8 of the PWM pins will be used to control the servos. The arrangement of the pins on the micro-controller is shown in the pin diagram in Figure 8 below.

Figure 8: Arduino Teensy 3.2 Pin Diagrams



(Source: <https://www.pjrc.com/teensy/)>

**Distance Sensor Input Signals & “Polling”**

Sensor polls are moments when the Arduino’s code checks and stores one of the four distance sensor’s detected distances. The sensor has a speaker that makes clicks whose echoes are then picked up by a mic. The sensor continuously times the amount of time between clicks and echoes and transmits this amount of time as a series of signals on its middle pins. When the Arduino polls the sensor, it receives these signals and decodes them as the amount of time between the clicks and echoes. Then, the Arduino multiplies this decoded time by the speed of sound to get the distance that the sensor detects.

**Servo Control Signals**

The PWM signals sent to the servos are the servos’ “control signals,” since they control which position the servos’ gears rotate to. Namely, the percentage of “high time” of the PWM signals, ranging from 5% to 10%, determines which degree the servo will rotate to in the range from 0 degrees to 180 degrees. By setting PWM signal high times that are between 5% and 10% in the same proportion that the desired servo position’s degree is between 0 and 180, any degree

position from 0 to 180 can be achieved. According to the “SG90 TowerPro Servo Datasheet,” the PWM signals must have a period of 20 milliseconds, which is 50 Hertz. Therefore, the range of acceptable high times is 5% of 20 milliseconds (1 millisecond) to 10% of 20 milliseconds (2 milliseconds).

**Receiving Latest Command from Raspberry Pi**

When the Raspberry Pi transmits the latest command over the USB cable to the Arduino, the Arduino will receive and read it as a serial input. In the program for the Arduino, there is a fresh check for new serial input values every iteration of the Arduino “loop()” function, which is a block of code that executes infinitely.

**Translating Latest Command into Sensor Polls & Servo PWM Signals**

When the Arduino receives a command, it will translate the command into a series of sensor polls and servo PWM signals that repeat in an infinite loop. The timing of adduction and tension for each leg must be precisely controlled for movement to mimic that of a spider. For this project, the team’s current solution is to send alternating left/right signals to different legs to achieve rotation in the basic repeated pattern of extension, abduction, flexion, adduction. Leg rotation in this manner is roughly equivalent, but not exactly equivalent, to organic leg movement.

**CONCLUSIONS & RECOMMENDATIONS**

The robotic spider described in this report is a worthwhile project for any robot hobbyist. Wall-climbing can be achieved using adhesives, which is a better method than suction, as discussed in the interview with Dr. Chase Cotton. It is recommended that readers interested in producing more organic leg movement consult sources on spider anatomy and movement, as well as sources on machine learning. The leg apparatus used is not fully conducive to organic movement. An alternative apparatus might use the “Klann Linkage,” which produces leg scooping and swinging motions concurrent with rotation.

In conclusion, this robot is a small start to a problem for which there is a large market and scientific interest. At the time of writing, the consumer technology is cheap enough for anyone on a $200 budget to buy the parts for this robot. The only challenge is to design the robot carefully enough that its movements are organic. This is an area of ongoing research.

**APPENDIX**

**Appendix A: Works Cited**

Python Web.py Module Documentation: <http://webpy.org>

Setting Up WiFi with Occidentalis: [https://learn.adafruit.com/adafruits-raspberry-pi-lesson-3-network-setup/setting-up-wifi-with-occidentalis](https://learn.adafruit.com/adafruits-raspberry-pi-lesson-3-network-setup/setting-up-wifi-with-occidentali)

Run Script at Boot: <https://www.raspberrypi.org/forums/viewtopic.php?f=48&t=70520>

Advanced Bash-Scripting Guide: [http://tldp.org/LDP/abs/html/](%22http:)

Python Requests Module Documentation: <http://docs.python-requests.org/en/master/>

Python PySerial Module Documentation: https://pythonhosted.org/pyserial/

SG90 TowerPro Servo Datasheet: <http://www.micropik.com/PDF/SG90Servo.pdf>

Klann Linkage: https://en.wikipedia.org/wiki/Klann\_linkage

**Appendix B: Amazon Product Links**

HC-SR04 Distance Sensor: https://www.amazon.com/SainSmart-HC-SR04-Ranging-Detector-Distance/dp/B004U8TOE6/ref=sr\_1\_2?ie=UTF8&qid=1485120517&sr=8-2&keywords=distance+sensors

SG90 TowerPro Servos: <https://www.amazon.com/Angelelec-Steering-High-Strength-Transparent-Precision/dp/B01EDQVPAA/ref=sr_1_3?s=pc&ie=UTF8&qid=1485059003&sr=1-3&keywords=sg90>

Raspberry Pi 3 Microprocessor: [https://www.amazon.com/Raspberry-Pi-896-8660-Model-Motherboard/dp/B01CD5VC92/ref=sr\_1\_2?s=pc&ie=UTF8&qid=1485044056&sr=1-2&keywords=raspberry+pi](%22)

Universal Battery Elimination Circuit: [https://www.amazon.com/Novak-5465-5-Amp-Universal-Bec/dp/B000WZ4PW2/ref=sr\_1\_2?ie=UTF8&qid=1485048013&sr=8-2&keywords=universal+battery+eliminator+circuit](%22ht)

USB WiFi Dongle: [https://www.amazon.com/Rosewill-Wireless-adapter-Dongle-150Mbps/dp/B00ZWPPD0K/ref=sr\_1\_2?s=pc&ie=UTF8&qid=1485048617&sr=1-2-spons&keywords=wifi+dongle&psc=1](https://www.amazon.com/Rosewill-Wireless-adapter-Dongle-150Mbps/dp/B00ZW)

Micro USB Cable: [https://www.amazon.com/AmazonBasics-USB-Male-Micro-Cable/dp/B01EK87A82/ref=sr\_1\_6?ie=UTF8&qid=1485054204&sr=8-6&keywords=usb%2Bto%2Bmicro%2Busb%2Bcable&th=1](https://www.amazon.com/AmazonBasics-USB-Male-Micro-Cable/)

Arduino Teensy Microcontroller: [https://www.amazon.com/PJRC-6485230-Teensy-3-2/dp/B01FSTHUPA/ref=sr\_1\_3?s=pc&ie=UTF8&qid=1485055184&sr=1-3&keywords=arduino+teensy](https://www.amazon.com/PJRC-6485230-Teensy-3-2/dp/B01FSTHUPA/ref=sr_1_3?s=pc&ie=UTF8&)