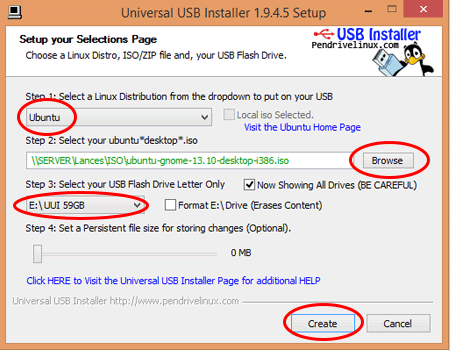
Project Ottershaw Last Updated: 02/20/15

**Computer Setup: Ubuntu**

**I. Installing the Ubuntu Partition:**

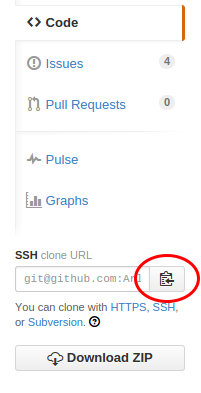
1. Materials:
   1. Ubuntu image file (\*.iso extension) for 64-bit or 32-bit, depending on your machine
   2. USB flash drive with at least 2 GB of memory or similar CD
2. Download the \*.iso file from the [Ubuntu web site](http://www.ubuntu.com/download/desktop) and follow the instructions. Save the file in your desktop or home folder.
3. Assuming a Windows-based machine, we need to format the USB to be a bootable flash drive. For doing this download Universal USB Installer from [pendrivelinux](http://www.pendrivelinux.com/universal-usb-installer-easy-as-1-2-3/" \l "button).
4. After installation, run the program and pick Ubuntu for Step 1, search for the \*.iso file using the *Browse* button and ensure that the correct letter designation is picked for your memory stick. Usually, the program auto-detects the device's letter. Check the format Drive box right next to the drive selection drop-down box and click create.



1. Once the formatting is complete, go into the drive folder and you should see many folders and files. Run the wubi.exe file and it should prompt you to try Ubuntu. Click the first button and then click the radio button that offers to help you make a bootable screen during reboot.
2. After the program installs that helper software, reboot your computer and press F9 in the boot screen. Make your computer boot from the USB and the Ubuntu splash screen should appear. Ensure that you enter the option that says Install Ubuntu.
3. Follow the setup and leave enough space for the partition during the last step. Click install and wait for the computer to partition the hard drive with the new Linux OS.
4. After a reboot, you should be able to see the option to run the computer with your Ubuntu partition.

**II. Programming Software Installation**

1. **NOTE**: Before setting anything up, ensure that your computer has Python 2.7.X since most of the intermediary software cannot run with Python 3.X
2. **NOTE 2:** Familiarize yourself with using the Terminal, Ubuntu works best when the user interfaces with the OS using this tool. From here you can make, move, and remove, files or directories, navigate through your file system, run code, launch applications, and even download software.
3. There are many ways to program in Python; however, there are two main options:
   1. Download your text editor of preference, there are many out there that you can try out, but pick the one you feel the most comfortable with. Some examples are: [gedit](https://apps.ubuntu.com/cat/applications/gedit/), [sublime text](http://www.sublimetext.com/2), notepad +, kate, Vim, [Geany](http://www.geany.org/Download/ThirdPartyPackages), etc.
   2. Another option is using an IDE such as [Eclipse](https://eclipse.org/downloads/). This has the added bonus of having a debugger; nonetheless, there will be extra setup for the environment to be able to interpret Python code.
4. Once you Python programming environment is setup, install the [Arduino IDE](http://playground.arduino.cc/Linux/Ubuntu). Follow the link and download it through the Terminal using the *sudo apt-get update && sudo apt-get install arduino arduino-core* command.
5. At this point you are ready to start developing; however, to be able to collaborate, we've set up a Git repository which you are going to need to access with rights to push and pull. If you don't have a GitHub account, please [sign up](https://github.com/).
6. You can decide whether your team wants to use the Terminal to run Git commands or if its better to use a Git Client. Again, there are many clients that you can use, but we used [SmartGit](http://www.syntevo.com/smartgit/). Before you install the program, make sure you run this quick [tutorial](http://eric.sau.pe/how-to-install-smartgit-on-ubuntu/) first to make sure Git and Mercurial are installed. Without this, SmartGit will **not** work.
7. Now, there is one more thing we need to set up in order to connect your computer to the Git Repository: an SSH key. This is just an authentication key for GitHub to know who can push to the repository. Follow this tutorial on how to generate an [SSH Key](https://help.github.com/articles/generating-ssh-keys/).
8. SmartGit Setup:
   1. Continue to open SmartGit by running the smartgit.sh script with the ./ command in the bin folder of the directory.
   2. Click *Repository → Clone* or if you are prompted at startup to choose, pick clone git repository followed by OK.
   3. Check the Remote Git, Mercurial or SVN repository on the pop-up screen and copy the link of the URL of the [ProjectOttershaw Git Repository](https://github.com/Arlo012/ProjectOttershaw) found on the right side of the page, as shown in the image, in the text box below.



* 1. Click *Next* and write the address of the SSH key created in step g) followed by it's passphrase and click *Login*. Refrain from storing the passphrase, since this is a security layer for the project.
  2. After this, run a *Pull* command found in the top left corner and you should see the directory appear in the navigation panel on the left.

1. ROS Core Setup: Since ProjectOttershaw works with the Raspberry Pi and the Arduino, you need to install both the general ROS-Indigo with Catkin and ROSserial for interfacing with the Arduino.
   1. ROS-Indigo: Follow the wiki for ROS installation provided [here](http://wiki.ros.org/indigo/Installation/Ubuntu). A couple of things to point out: run the sources.list command for your appropriate Ubuntu version (We used **Trusty** for 14.04), install **ROS-Base (Bare Bones)**, which is the minimum required to have the ROS working on your machine.
   2. Familiarize yourself with the ROS commands. Follow [ROS Tutorials](http://wiki.ros.org/ROS/Tutorials) one through eight at least from the beginner level, to get a good feel for the environment, and tutorial 12 to understand how to develop in Python using ROS.
   3. ROSserial: Follow the wiki for [Arduino IDE Setup](http://wiki.ros.org/rosserial_arduino/Tutorials/Arduino IDE Setup). Keep in mind that you **have** to follow the Catkin tutorial and **not** the rosbuild.
2. Once that tutorial is finished, your team is completely ready to add on to the Legacy of Project Ottershaw. Remember to keep updating this document as the project develops. Boris the spider will both teach you and learn from you. We expect nothing but great things from the upcoming development teams. GOOD LUCK!