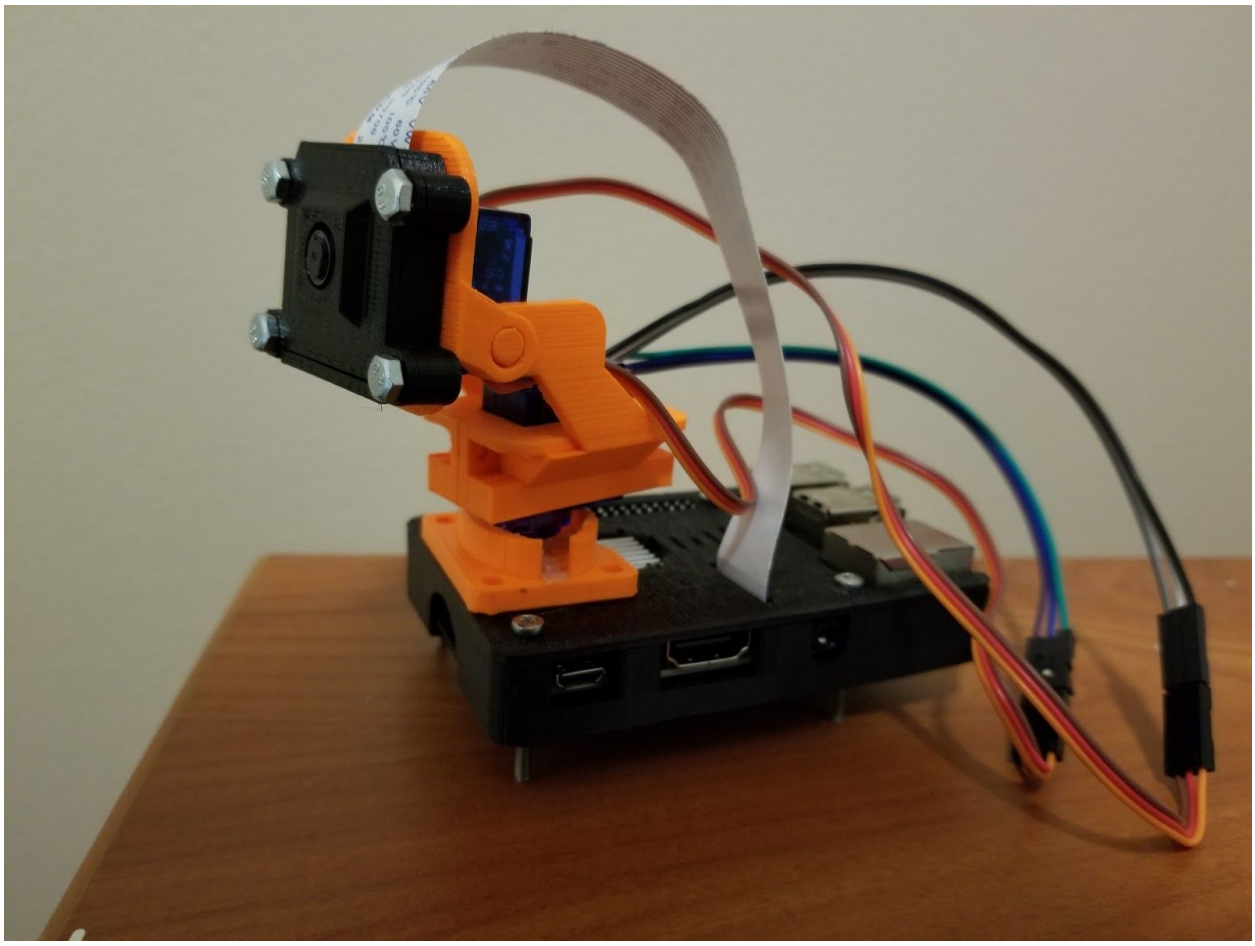
Case Assembly

This project uses a case to enclose the Raspberry Pi and a servo gimbal to connect the servos and mount the gimbal onto the case.
The parts should be 3D printed in ABS or PLA plastic at 20% infill or higher. To assemble the parts:

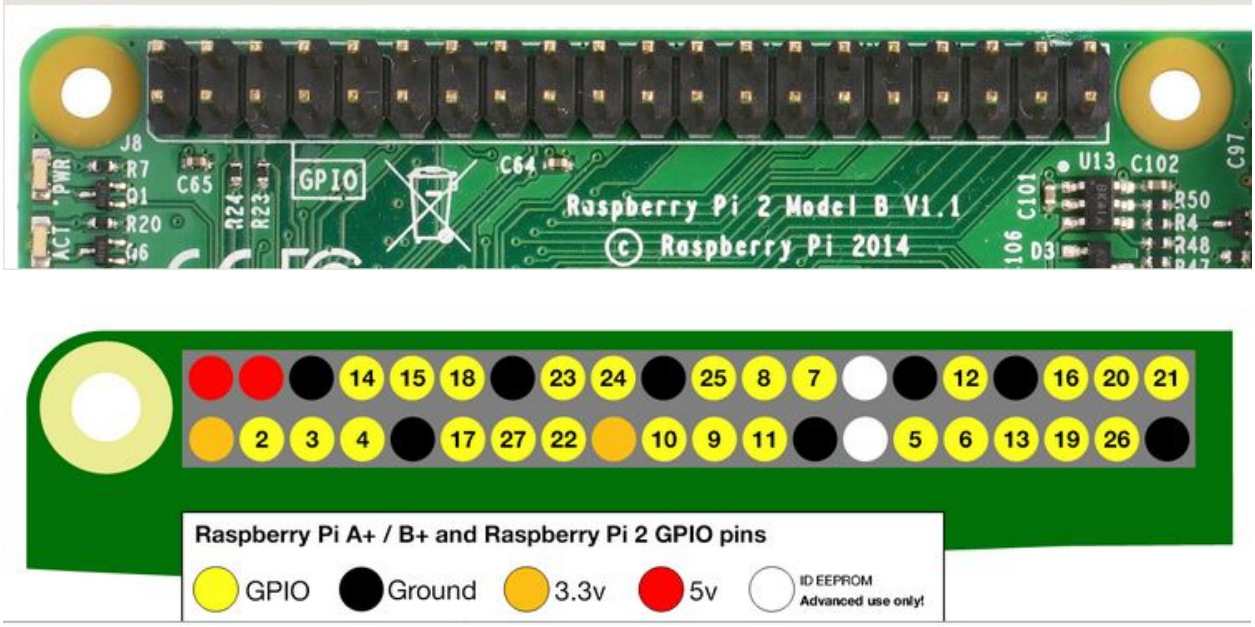
1. Hot glue the base of the gimbal mechanism onto the case lid. Place the first servo into the base and fix a screw into the servo's shaft from the underside of the case lid.
2. Slide the ribbon cable from the camera module through the slit in the lid of the case and connect the camera module into the CSI camera port on the raspberry Pi. Then close the lid on top of the base.
3. Screw four M3 screws into the corners of the case to secure the lid onto the base.
4. Enclose the servo with the parts titled HolderL.stl and HolderR.stl. Line the holes in both of these parts up so that a screw may be inserted through both holes in order to join these two parts together around the servo. Then insert two M2.5 screws through the holes.
5. Super glue the Camera_housing_part1.stl to the part in the base.stl file which has 3 holes protruding from it. The camera housing part should be superglued to the opposite side of that part.
6. Place the second servo against the side of the part with three holes protruding from it and use the two screws included with the servo to mount it to the part. Then place the single ended attachment included with the servo into the groove in the HolderR.stl file. Insert longer screw included servo through the single-ended attachment to fix it to the servo shaft. By this point, all parts of the servo gimbal should be connected together.
7. Enclose the camera module in the camera case and mount it to the front of the gimbal using 4 short M3 screws.
8. Refer to the image of the final assembly below for a visualization of the final product. For more details on the CAD files, please refer to the original designs that the parts for this project were taken from: https://www.thingiverse.com/thing:504196 and https://www.thingiverse.com/thing:2892903 (citations provided in the Final Report).



Connecting the Servos to the GPIO pins

1. Orient the newly constructed raspberry pi + gimbal assembly such that the GPIO pins are at the upper edge of the raspberry pi (i.e. the camera should be pointed to the left).
2. Find the Ground, Power, and Signal wires for each of the 2 servos. They are color-coded. The brown wire is ground, the red wire is power, and the orange wire is signal.
3. Obtain 6 female-male jumper cables. The female end will plug into the gpio pins on the raspberry pi. The male ends will plug into the 3 wires coming out of each servo.

4. Identify the two power (5V) GPIO pins in the gpio pinout below. They are colored red. Connect 2 of the jumper cables to those two gpio pins and connect the other ends of the jumper cables to the power wire on the 2 servos.
5. Identify two of the ground GPIO pins in the gpio pinout below. They are colored black in the diagram. Connect 2 of the jumper cables to those two gpio pins and connect the other ends of the jumper cables to the ground wires on the 2 servos.
6. Identify pins 17 and 27 in the gpio pinout below. Connect the remaining 2 jumper cables to those two gpio pins. Connect the jumper cable which is now attached to pin 17 to the signal wire on the servo closest to the raspberry pi (the azimuth servo). Connect the last jumper cable to the signal wire of the other servo.



****NOTE: This user manual assumes the reader has access to a Raspberry Pi 3B with a Raspbian image installed, and that the user has working knowledge of a terminal and Debian-based Raspbian environment. If this assumption does not hold, there are various tutorials online for configuring a raspberry pi for use as a desktop with a fresh installation of the Raspbian OS. Furthermore, This project uses a virtual environment in python to easily isolate packages used in one project from those used in another project. The user manual will assume that the reader has created a virtual environment called "capstone", or has opted to use another method to download and install packages for this specific project.****

Raspbian Packages Installation

To run the scripts used in this project, there are some prerequisite packages that must be installed first.

1. First, openCV must be installed on the raspberry pi. Follow the instructions below the text "**How to pip install OpenCV on Raspberry Pi"** in this tutorial: https://www.pyimagesearch.com/2018/09/19/pip-install-opencv/ which outlines the steps to install OpenCV with OR without the use of virtual environments. However, note that the instructions for installing opencv with the use of a virtual environment make a new environment called "cv" whereas in this project, the authors created a virtual environment titled "capstone"- a small detail.
2. If the optional method used to install opencv in a virtual environment was not used in the step above, skip this step. Otherwise, enter the virtual environment that you created in the previous step by typing "workon [virtual_environment_name]" into the terminal. You should see that any commands entered on the terminal after this command should be prefixed with the name of the virtual environment that you have entered. NOTE: This step must be completed each time a new terminal is opened.
3. Type "sudo apt-get install pigpio python-pigpio python3-pigpio" to install the pigpio library used for software control of the GPIO pins necessary for smooth operation of the servo motors.
4. Type "pip install imutils" to install the dependency.

5. Type “sudo pigpiod” to activate the daemon required to use the functionality of the pigpio library.
   NOTE: This step must be done prior to running any code associated with this project and must be done each time you login to the raspberry pi.

Running Scripts

Before running any script, check that a few things are performed. First, if working with virtual environments, make sure that you are in the correct one by typing “workon [name_of_virtual_environment]” in the terminal. This must be done each time a new terminal is opened. Next, make sure that the pigpiod daemon is active by running “sudo pigpiod”. Note that this only needs to be done once per session. Finally, if running integrated.py or hog_integrated.py, ensure that the gpio_servo.py script is in the same directory as these files. You can do that by navigating to the directory where integrated.py and hog_integrated.py are stored, and typing “cp /path/to/gpio_servo.py ./gpio_servo.py” which will copy gpio_servo.py into the working directory.

To run *integrated.py*, first make sure you are in your virtual environment, if using one, and that you have activated the pigpiod daemon. In the terminal, type “python3 integrated.py -t mosse” which will run the integrated.py python script using the mosse tracker. A window will appear with a live feed from the raspberry pi camera module. At any time, type the letter ‘s’ and the live feed will freeze. Then, using your mouse, click and drag the mouse over a person or object frozen on the window. Then click the spacebar or “Enter” to continue with the live feed. Now, the bounding box that you have drawn around the person or object should be persistent, and continue to bound the person or object as it moves around within view of the camera. If at any time the bounding box disappears or never locked onto the intended person or object, simply click the ‘s’ key again and redraw a new bounding box. The raspberry pi gimbal should move appropriately to keep the person within view of the camera provided the person moves slowly enough and does not change appearance rapidly (such as by turning around). At any time while the feed is frozen, press ‘c’ to cancel the selection and return to the live feed from the camera.

To run *hog_integrated.py,* make sure that you are in the correct virtual environment if using them and that the command “sudo pigpiod” has been run already. Next, ensure that the hog_bb.py script is in the same directory as hog_integrated.py by copying it into the directory using the same instructions as outlined above. Run the script by typing

"python3 hog_integrated.py -w $POSINT$", replacing $POSINT$ with some positive integer; 30 is the default. This integer is the number of frames to wait before recomputing HOG if the bounding box is either lost or stationary. The live feed should then display on the monitor. If at any point the screen freezes momentarily, it is because the HOG computation is loading the CPU to perform this computationally expensive operation.