Scott Dudley

**Sprint 2 Research:**

***Discussion with TARDEC / Finding PiCar-V: (4 Hours)***

For Sprint 2, our group contacted TARDEC via conference call to discuss options for the Robot Follower project. We needed to find sufficient hardware to achieve our requirements. After a little bit of research, I found Sunfounder’s PiCar-V, a robotics kit that fit all of our needs at an affordable cost. The kit includes parts to build a robot car, using a Raspberry Pi for the controller, and a camera that we can program to track objects. We confirmed with TARDEC that they can order two of these kits for us. Our group plans on learning from the available code on Sundfounder’s Git, while configuring our own software and desktop application to interface with the cars. Sources about PiCar-V are provided below:

PiCar-V Website:

<https://www.sunfounder.com/smart-video-car-kit-v2-0.html>

PiCar-V Manual:

<https://www.sunfounder.com/learn/download/U21hcnRfVmlkZW9fQ2FyX1YyLjBfZm9yX1Jhc3BiZXJyeV9QaV9QaUNhci1WXy5wZGY=/dispi>

PiCar-V Git:

<https://github.com/sunfounder/SunFounder_PiCar-V>

<https://github.com/sunfounder/SunFounder_PiCar>

***Researching Python Components: (3 hours)***

We will not receive the hardware until Sprint 3, so I decided to research how Sunfounder’s software for the PiCar worked. Sunfounder uses primarily Python code. They use Python dependencies called I2C and OpenCV. The I2C interface uses SMBus functions, which are used in this case for Sunfounder’s messaging protocol. OpenCV is used with the ‘libjpeg8-dev’ package to capture, stream, and edit the video output from the camera on the PiCar-V. For networking, Sunfounder connects to a Django server where everything is being controlled. Django allows the creation of a server that can be accessed via html by any third-party application. Since most of my component research was focused on Django, I created its own section below my sources for I2C and OpenCV:

I2C interface:

<http://www.raspberry-projects.com/pi/programming-in-python/i2c-programming-in-python/using-the-i2c-interface-2>

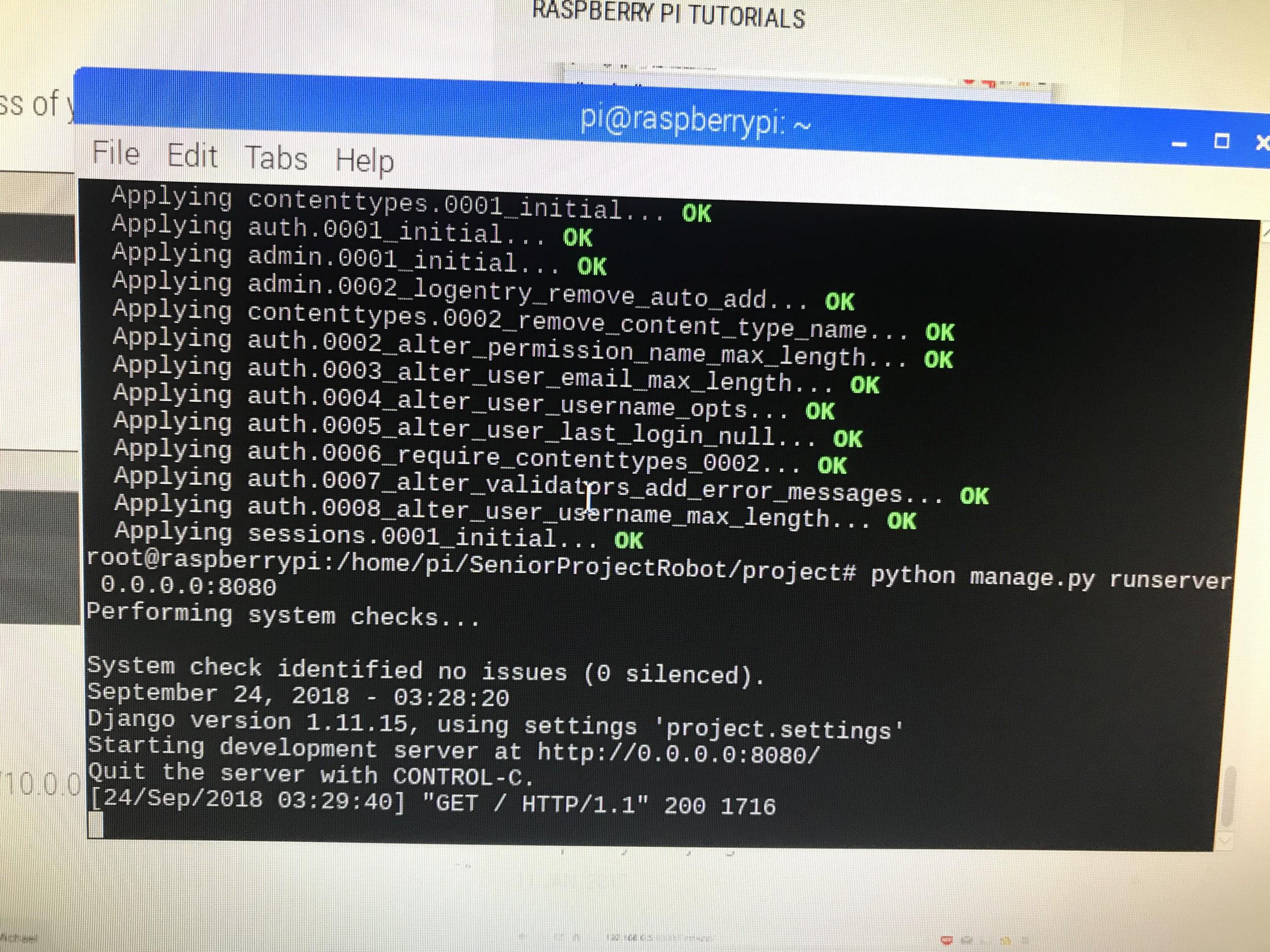
OpenCV website:

<https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_setup/py_table_of_contents_setup/py_table_of_contents_setup.html#py-table-of-content-setup>

***Django Server Creation: (8 hours)***

Since we knew that we wouldn’t be receiving hardware anytime soon, I decided to order my own Raspberry Pi to attempt the creation of my own Django server, so that we could quickly translate it over to our project hardware once we receive it. I also created a BASH script that could be quickly run on a Pi that would install all the building blocks that we would use in the main software. With a lot of trial and error, I eventually got a Django server running that I could access via web page using a specific port, 8080 in this case.

*Django Server from Raspberry Pi:*



*Accessing Server via Safari on iPhone:*

******

Although I was able to get a Django server running, we decided to omit the idea and instead use simpler socket programming for networking. This decision was made toward the end of Sprint 2. Sources for Django server creation are provided below:

Django Tutorials:

<https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django>

<http://raspberrypituts.com/raspberry-pi-django-tutorial-2017/>

<https://docs.djangoproject.com/en/1.10/intro/tutorial01/>

***Use Case Diagram: (2 hours)***

I also updated our Use Case Diagram since we now knew for sure the hardware and software that we would be using for the project. We ended up keeping a few theoretical ideas while adding and adjusting several functional requirements.

***Total Work for Sprint 2: 17 hours***