# Installation instructions:

1. Switch user to root, create the /opt/gps directory and grant the needed permissions:

sudo su

mkdir /opt

mkdir /opt/gps

chmod 777 /opt/gps

1. To enable communication to Azure, we need the Azure SDK for Python. We need to download it, setup the build environment, build it, then copy the iothub\_client.so file to the /opt/gps directory. This may take ~15 – 30 minutes.

sudo su

// download the Azure IoT client for python

git clone --recursive https://github.com/Azure/azure-iot-sdk-python.git

// setup the Raspberry Pi build environment

~/azure-iot-sdk-python/build\_all/linux/setup.sh

// build the client

~/azure-iot-sdk-python/build\_all/linux/build.sh

// copy the file we need to the target directory

cp ~/azure-iot-sdk-python/device/samples/iothub\_client.so /opt/gps/

1. Modify the /etc/network/interfaces file to use Ad-Hoc Wifi connections and assign a static IP. Note, each host should have a different static IP address. Delete all lines after and including the **iface wlan0**, then add the following lines 6:

iface wlan0 inet static

address 192.168.1.100

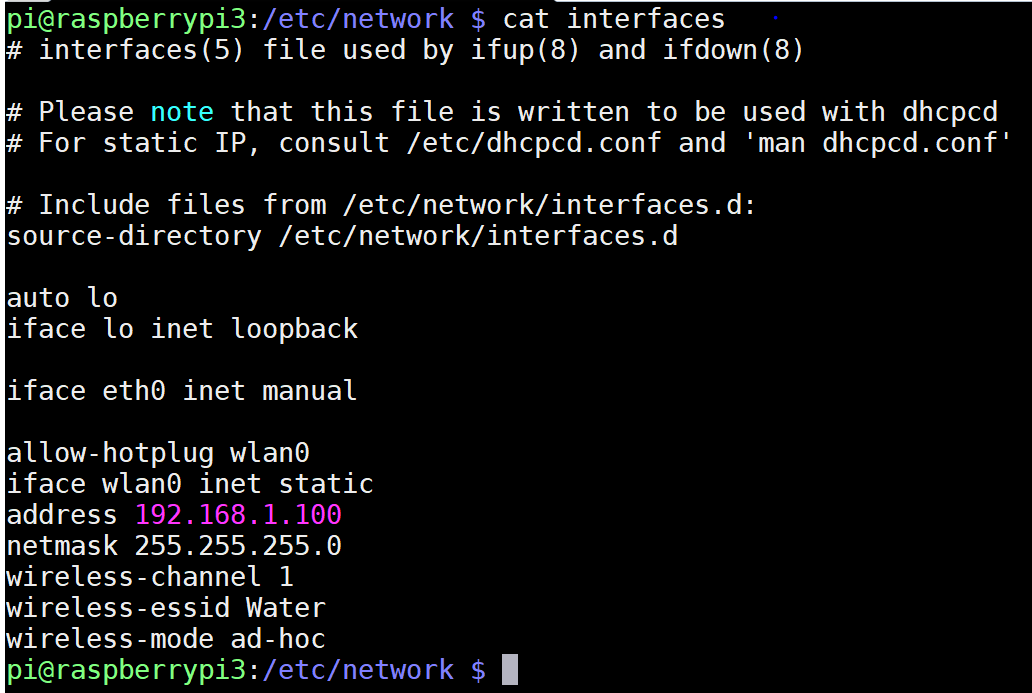
netmask 255.255.255.0

wireless-channel 1

wireless-essid Water

wireless-mode ad-hoc

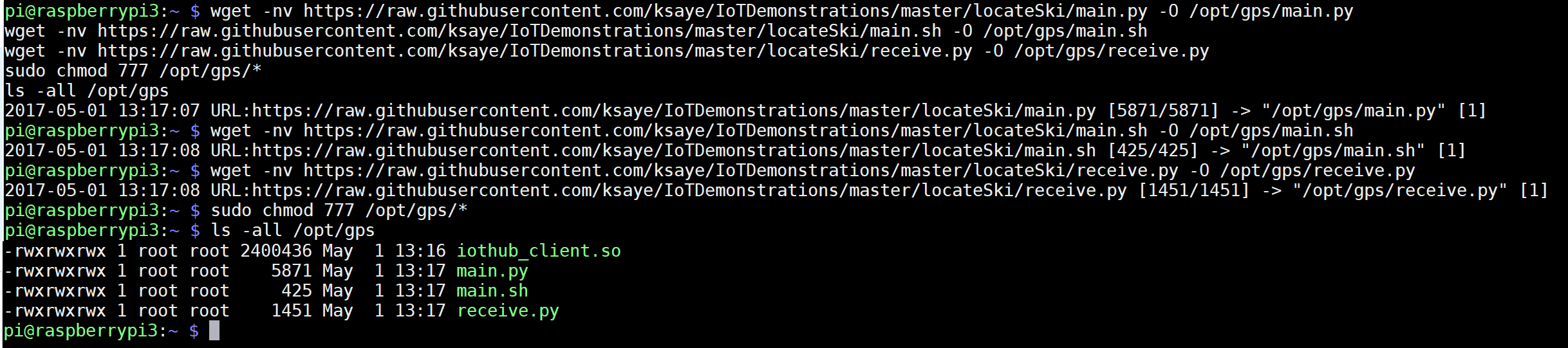
Your /etc/network/interfaces file should look like this:



1. Download the following 3 files to the /opt/gps directory using the following commands and set the permission to execute (each bullet is a complete line / command):

* wget https://raw.githubusercontent.com/ksaye/IoTDemonstrations/master/locateSki/main.py -O /opt/gps/main.py
* wget https://raw.githubusercontent.com/ksaye/IoTDemonstrations/master/locateSki/main.sh -O /opt/gps/main.sh
* wget https://raw.githubusercontent.com/ksaye/IoTDemonstrations/master/locateSki/receive.py -O /opt/gps/receive.py
* sudo chmod 777 /opt/gps/\*

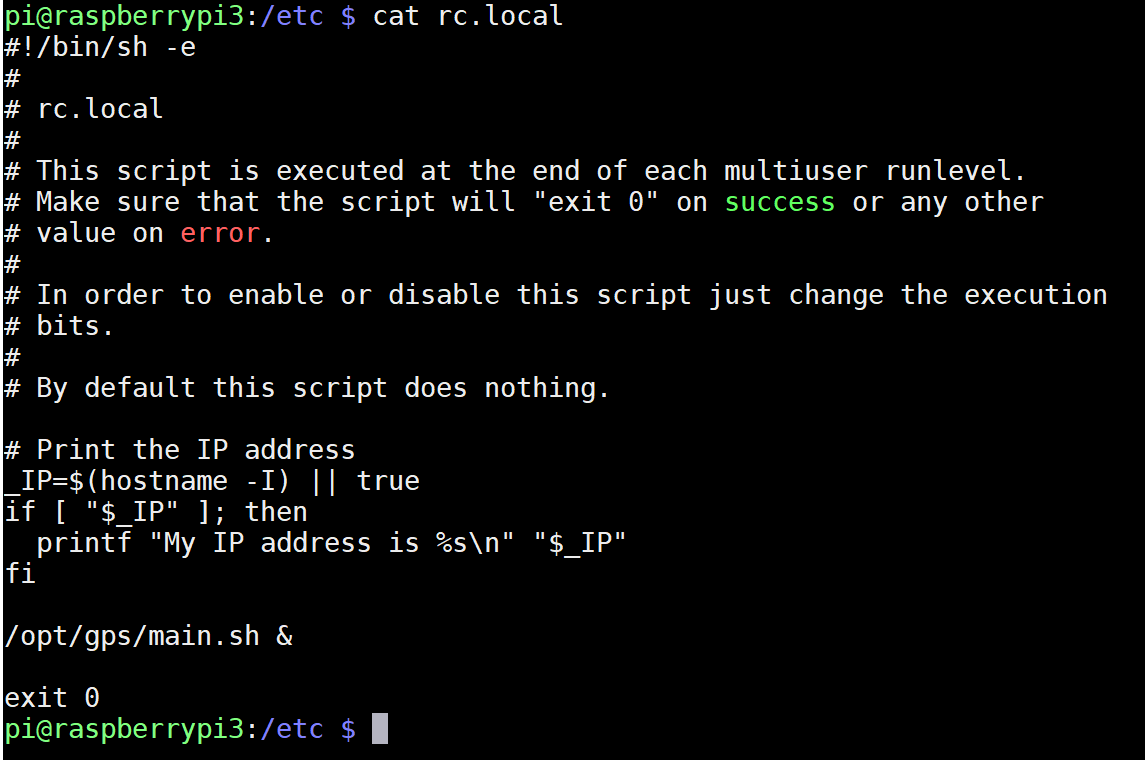
The following screen shows the output of the commands:



1. Modify the /etc/rc.local by inserting the following command before the exit 0.

/opt/gps/main.sh &

Your /etc/rc.local file should look something like this:



1. Next, modify the connection\_string section of your main.py (line 40) to reflect the unique connection string required for each device.
2. Make any needed changes to /opt/etc/receive.py which might include: different keepdistance value (line 20), different GPIO pin or different action.

# Operations:

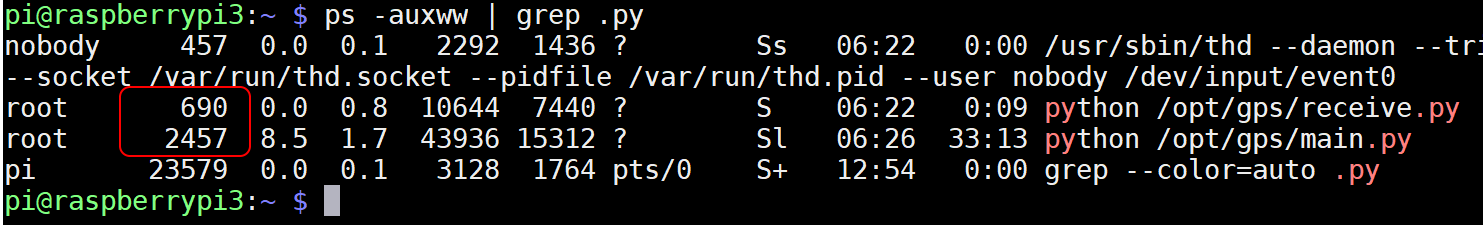
The main.sh script runs everytime the machine boots. If there is ever a problem or crash with main.py or receive.py, main.sh will restart it.

If you need to change either main or receive.py simply save the changes and kill the existing process and main.sh will restart.

To see the existing processes, type:

ps -auxww | grep .py

Note the process ids below 690 and 2457 of receive and main.py respectively.



To kill a process, type:

sudo kill -9 ####

Where ### is the PID of the process seen by the ps command.

To see interactively what the scripts are doing, kill the main.sh process, then kill the main and receive.py processes and simply execute the script natively, such as the following command:

/opt/gps/main.py

To stop the interactive command, just use Control + C.