Run a Python code at Raspberry Pi startup with SYSTEMD

The Systemd can control which programs should start when the Raspberry Pi boots up. [1]

Step 1– Create A Unit File
Open a unit file called “sensors” using the command as shown below:

sudo nano /lib/systemd/system/sensors.service

Add the following command inside the Unit file:

```
[Unit]
Description=My Sensors Service
After=multi-user.target

[Service]
Type=idle
ExecStart=/usr/bin/python /Your/python/scriptPath.py

[Install]
WantedBy=multi-user.target
```

You should save and exit the nano editor.
Your Unit file defines a new service called “sensors Service” and we are requesting that it should be launched when the multi-user environment running. The “ExecStart” parameter is used to declare the command that we want to run at boot up. The “Type” is set to “idle” to ensure that the ExecStart command will run only when everything else has loaded during boot up. Note that the paths are absolute and you should declare the complete location of Python and your script.
You can change the ExecStart line as follows to store script’s output in a log file:
ExecStart=/usr/bin/python /home/maghsoud/sensor_reader.py > /var/log/sensors.log 2>&1

The you need to change the permission of your Unit file so that it can be executable:
sudo chmod 644 /lib/systemd/system/sensors.service

Step 2 – Finally, Configure systemd
Finally, you can reload the daemon and configure system to start your service during boot up:

```
sudo systemctl daemon-reload
sudo systemctl enable sensors.service
```

Reboot the Raspberry pi:

```
sudo reboot
```

References

[1] MATT, "How To Autorun A Python Script On Boot Using systemd," 12 October 2015. [Online]. Available: https://www.raspberrypi-spy.co.uk/2015/10/how-to-autorun-a-python-script-on-boot-using-systemd/. [Accessed 22 Novemebr 2018].