Sensor testing and validation

1. When you are testing a sensor, on each boot, you must turn off transmission and sensing:

```
sudo systemctl stop sensor && sudo systemctl stop transmit
```

This will make sure that the sensor does not try to record values and send them to server.

It is very important not to forget this step!!! Make sure to turn off the sensor and transmit processes every 10 minutes or so, as the Thing will periodically restart these services.

2. Then copy the sensor.yml for this Thing to /home/pi. You can do this using scp as follows:

```
scp 1.yml pi@IP.ADDRESS.OF.PY:~/sensor.yml
```

Where "1.yml" is the filename of a particular Thing.

3. Then, use `sudo raspi-config` to do the following:

- Interfacing Options
    - SPI
        - Enable
    - I2C
        - Enable
    - Serial
        - Do NOT enable shell access over serial
        - Enable serial port hardware
- Advanced Options
    - Expand Filesystem

4. Then reboot the sensor when asked to when exiting raspi-config.

5. On reboot, once again disable the sensor and transmit processes:

```

sudo systemctl stop sensor && sudo systemctl stop transmit
```

6. Make sure the filesystem was actually expanded:

```
pi@raspberrypi:~ $ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root       7.1G  1.4G  5.4G  21% /
devtmpfs        460M     0  460M   0% /dev
tmpfs           464M     0  464M   0% /dev/shm
tmpfs           464M   12M  452M   3% /run
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs           464M     0  464M   0% /sys/fs/cgroup
/dev/mmcblk0p1   42M   22M   20M  54% /boot
tmpfs            93M     0   93M   0% /run/user/1000
pi@raspberrypi:~ $
```

7. Now create a test version of the sensor.yml:

```
cp sensor.yml sensor-test.yml
```

Then edit sensor-test.yml to make sure that console log level is set to 'DEBUG'. The logging section should look like:

```
logging:
  logger_path: /var/log/sensor.log
  level_file: WARNING
  level_console: DEBUG
```

8. Next, you can run the test program, which just reads data from the sensors, without storing or transmitting the results. The output should look something like:

```
pi@raspberrypi:~ $ sudo sensor_raspi_test -c sensor-test.yml
2018-06-16 18:17:16,226 - sensors - DEBUG - About to start scheduler...
2018-06-16 18:17:16,227 - sensors - DEBUG - Test Sampler: scheduling observation sampling...
2018-06-16 18:17:16,227 - sensors - DEBUG - Test Sampler: Running scheduler...
2018-06-16 18:17:21,232 - sensors - DEBUG - Begin generate_observations_minute...
2018-06-16 18:17:21,234 - sensors - DEBUG - Iterating over 2 sensors...
2018-06-16 18:17:21,235 - sensors - DEBUG - <sensors.raspi.sensor.mq131.Mq131 object at 0x76370a30>
2018-06-16 18:17:21,236 - sensors - DEBUG - Calling generate_observations for sensor type mq131...
2018-06-16 18:17:21,490 - sensors - DEBUG - Enqueing observations...
2018-06-16 18:17:21,491 - sensors - DEBUG - featureOfInterestId: None, datastreamId: 7aa5a63a-168b-11e8-bddd-
07415d9f6346,
        phenomenonTime: 2018-06-16T23:17:21.237515+00:00, result: 0.016964128105500142, parameters: {'Rs':
'11413.075950821556', 'Ro': '2.501', 'Rs_Ro_Ratio': '4563.405018321294', 'adc_avg': '61.6'}; id: None
2018-06-16 18:17:21,492 - sensors - DEBUG - <sensors.raspi.sensor.dht11.Dht11 object at 0x76370990>
2018-06-16 18:17:21,493 - sensors - DEBUG - Calling generate_observations for sensor type dht11...
2018-06-16 18:17:22,577 - sensors - DEBUG - Enqueing observations...
2018-06-16 18:17:22,579 - sensors - DEBUG - featureOfInterestId: None, datastreamId: b3342d2e-168f-11e8-bde0-
97af2ace9bc4,
        phenomenonTime: 2018-06-16T23:17:21.494426+00:00, result: 25, parameters: {'RH': 15}; id: None
2018-06-16 18:17:22,580 - sensors - DEBUG - featureOfInterestId: None, datastreamId: 924ae84a-1690-11e8-b2d0-
cf67eba61b1b,
        phenomenonTime: 2018-06-16T23:17:22.577544+00:00, result: 15, parameters: {'T_air': 25}; id: None
2018-06-16 18:17:22,581 - sensors - DEBUG - Test Sampler: End of iteration.
```

Let this run for 2-3 iterations. Hit `Ctl-C` to quick the test program.

9. Now you can run the calibration program for the MQ131 sensor (ideally after letting the Thing run overnight/for 24 hours). For small Things this looks like:

```
pi@raspberrypi:~ $ sudo calibrate_mq131
Calibrating MQ131 sensor (this will take about 30 seconds)...
Ro value for MQ131 sensor is 2.8064056342785766
pi@raspberrypi:~ $
```

For large Things, this looks like:

```
pi@raspberrypi:~ $ sudo calibrate_mq131 --adc ads1015
Calibrating MQ131 sensor (this will take about 30 seconds)...
Ro value for MQ131 sensor is 2.8064056342785766
pi@raspberrypi:~ $
```

This program will attempt to set the "zero point" or Ro value for the MQ131 ozone sensor. Technically this should be done in a controlled environment that has ~0ppm of ozone, but we will do this inside the building. Also, this should be done after the Thing has been on for at least 24 hours and the MQ131 sensor has had a chance to "warm up".

Once you obtain the Ro value, it can be added to the sensor.yml configuration for the MQ131 sensor, which should end up looking something like:

```
sensors:
  - type: mq131
    observed_properties:
      - name: ozone
        datastream_id: 1f02ccc6-5735-45f3-b1a0-0d358308b698
    properties:
      - Ro: 2.81
```

Next, re-run the test and see how the ozone values have changed a bit since setting the Ro value.

10. Test the config to make sure it is still valid

```
sudo sensor_raspi_test -c sensor.yml --configtest && sudo sensor_raspi_test -c sensor-test.yml --configtest
```

11. Delete sensor log and local database and shutdown

```
sudo systemctl stop sensor && sudo systemctl stop transmit && sudo rm /var/log/sensor.log && sudo rm /var/spool/sensor.sqlite && sudo shutdown -h now
```

Wait 10-15 seconds and then unplug/power down the Thing.