Grant Novota

Qt GUI Development Project

Assumptions/Notes:

- Using Python3.8 on an Ubuntu machine.
- Database is sqlite3 and interactions are through a python script using sqlalchemy.
- Generated graphs of temperature and humidity that update on a set time interval and show upper and lower bounds for alarms.
- Alarms are pop-ups.
- Humidity and temperature metrics are displayed in a table format in the UI.
- Made a change to show the total number of humidity/temperature samples and use the entire dataset for max, min, and average values.
- Encountered issues with the installation of Qt5 on Ubuntu and had to execute “sudo apt-get install --reinstall libxcb-xinerama0”.

Required Libraries:

- sqlalchemy
- PyQt5
- matplotlib
- numpy
- datetime

Code:

The project code is made up of 3 parts: the pseudo sensor code “pseudoSensor.py”, the database interaction code “db.py”, and the main code “main.py”

pseudoSensor.py:

```
import random

# pseudo temp and humidity sensor
class PseudoSensor:
    h_range = [0, 20, 20, 40, 40, 60, 60, 80, 80, 90, 70, 70, 50, 50, 30, 30, 10,
10]
    t_range = [-20, -10, 0, 10, 30, 50, 70, 80, 90, 80, 60, 40, 20, 10, 0, -10]

    h_range_index = 4
    t_range_index = 5
    humVal = 0
    tempVal = 0
```

```
    def __init__(self):
        self.humVal = self.h_range[self.h_range_index]
        self.tempVal = self.t_range[self.t_range_index]

    def generate_values(self):
        self.humVal = self.h_range[self.h_range_index] + random.uniform(0, 10)
        self.tempVal = self.t_range[self.t_range_index] + random.uniform(0, 10)

        self.h_range_index += 1

        if self.h_range_index > len(self.h_range) - 1:
            self.h_range_index = 0

        self.t_range_index += 1

        if self.t_range_index > len(self.t_range) - 1:
            self.t_range_index = 0

        return self.humVal, self.tempVal
```

db.py:

```
# database functions
from datetime import datetime, timezone
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy import Column, ForeignKey, Integer, String, Float, Boolean,
DateTime
from sqlalchemy import Index
from sqlalchemy.orm import relationship, backref, sessionmaker
from sqlalchemy import create_engine, select

Base = declarative_base()

# Tables
class Temperature(Base):
    __tablename__ = 'temperature'
    id = Column(Integer, primary_key = True, autoincrement=True)
    # fahrenheit
    value_f = Column(Float)
    # celsius
    value_c = Column(Float)
    time = Column(DateTime)
```

```
class Humidity(Base):
    __tablename__ = 'humidity'
    id = Column(Integer, primary_key = True, autoincrement=True)
    value = Column(Float)
    time = Column(DateTime)

# general db functions
def create(database):
    # an engine that the session will use for resources
    engine = create_engine(database)
    # create a configured session class
    Session = sessionmaker(bind=engine)
    # create a session
    session = Session()
    return engine, session

def result_dict(r):
    return dict(zip(r.keys(), r))

def result_dicts(rs):
    return list(map(result_dict, rs))

def database_dump(session):
    Database = [Temperature, Humidity]
    for table in Database:
        stmt = select('*').select_from(table)
        result = session.execute(stmt).fetchall()
        print(result_dicts(result))
    return

def create_tables(engine):
    Base.metadata.create_all(engine)
    return

def init_session():
    engine, session = create("sqlite:///db.sqlite3")
    create_tables(engine)
    return session

def close(conn):
    conn.close()
    return

def delete_obj(session, obj):
    session.delete(obj)
```

```
    session.commit()
    return

# add rows to tables
def add_temp(session, value_f, value_c, time):
    temp = Temperature(value_f=value_f, value_c=value_c, time=time)
    session.add(temp)
    session.commit()
    return

def add_humidity(session, value, time):
    humidity = Humidity(value=value, time=time)
    session.add(humidity)
    session.commit()
    return

def get_all_temps(session, type):
    temp_list = []
    temp_times = []
    temps = session.query(Temperature).all()
    for temp in temps:
        if type == "f":
            temp_list.append(temp.value_f)
        else:
            temp_list.append(temp.value_c)
        temp_times.append(temp.time)
    return temp_list, temp_times

def get_all_humids(session):
    humid_list = []
    humid_times = []
    humids = session. query(Humidity).all()
    for humid in humids:
        humid_list.append(humid.value)
        humid_times.append(humid.time)
    return humid_list, humid_times
```

main.py

```
# main code
import sys
import time
import numpy as np
```

```
from datetime import datetime

from matplotlib.backends.qt_compat import QtCore, QtWidgets, is_pyqt5
if is_pyqt5():
    from matplotlib.backends.backend_qt5agg import (
        FigureCanvas, NavigationToolbar2QT as NavigationToolbar)
    from PyQt5.QtWidgets import QApplication, QWidget, QPushButton, QTableWidget,
QTableWidgetItem, QInputDialog, QLineEdit
    from PyQt5.QtGui import QIcon
    from PyQt5.QtCore import pyqtSlot
else:
    from matplotlib.backends.backend_qt4agg import (
        FigureCanvas, NavigationToolbar2QT as NavigationToolbar)
from matplotlib.figure import Figure
from matplotlib import dates

# my libraries
import db
from psuedoSensor import PseudoSensor

# init database
session = db.init_session()

# alarm limits
temp_min_limit = 30.0
temp_max_limit = 80.0
humid_min_limit = 30.0
humid_max_limit = 70.0

class ApplicationWindow(QtWidgets.QMainWindow):
    def __init__(self):
        super().__init__()
        self.temp_min_limit = temp_min_limit
        self.temp_max_limit = temp_max_limit
        self.humid_min_limit = humid_min_limit
        self.humid_max_limit = humid_max_limit

        self._main = QtWidgets.QWidget()
        self.setCentralWidget(self._main)
        layout = QtWidgets.QVBoxLayout(self._main)

        # graphs of temperature and humidity - refresh on interval
        dynamic_canvas_temps = FigureCanvas(Figure(figsize=(5, 3)))
        layout.addWidget(dynamic_canvas_temps)
        # self.addToolBar(QtCore.Qt.BottomToolBarArea,
```

```
        #                 NavigationToolbar(dynamic_canvas_temps, self))

        self.dynamic_ax_temps = dynamic_canvas_temps.figure.subplots()
        self.temp_timer = dynamic_canvas_temps.new_timer(
            100, [(self.update_canvas_temps, (), {}),])
        self.temp_timer.start()

        dynamic_canvas_humid = FigureCanvas(Figure(figsize=(5, 3)))
        layout.addWidget(dynamic_canvas_humid)
        # self.addToolBar(QtCore.Qt.BottomToolBarArea,
        #                 NavigationToolbar(dynamic_canvas_humid, self))

        self.dynamic_ax_humid = dynamic_canvas_humid.figure.subplots()
        self.humid_timer = dynamic_canvas_humid.new_timer(
            150, [(self.update_canvas_humid, (), {})])
        self.humid_timer.start()

        # table of current temp/humidity
        self.current_table = QTableWidget()
        self.current_table.setRowCount(1)
        self.current_table.setColumnCount(2)
        self.current_table.setHorizontalHeaderLabels(["Temperature (F)",
"Humidity (%)"])
        self.current_table.setItem(0,0, QTableWidgetItem("0.0"))
        self.current_table.setItem(0,1, QTableWidgetItem("0.0"))
        layout.addWidget(self.current_table)

        # table of calculated metrics
        self.metrics_table = QTableWidget()
        self.metrics_table.setRowCount(1)
        self.metrics_table.setColumnCount(7)
        self.metrics_table.setHorizontalHeaderLabels(["Total Samples", "Min
Temperature (F)", "Min Humidity (%)", "Max Temperature (F)", "Max Humidity (%)",
"Avg Temperature (F)", "Avg Humidity (%)"])
        self.metrics_table.setItem(0,0, QTableWidgetItem("0.0"))
        self.metrics_table.setItem(0,1, QTableWidgetItem("0.0"))
        self.metrics_table.setItem(0,2, QTableWidgetItem("0.0"))
        self.metrics_table.setItem(0,3, QTableWidgetItem("0.0"))
        self.metrics_table.setItem(0,4, QTableWidgetItem("0.0"))
        self.metrics_table.setItem(0,5, QTableWidgetItem("0.0"))
        self.metrics_table.setItem(0,6, QTableWidgetItem("0.0"))
        layout.addWidget(self.metrics_table)

        single_button = QPushButton('Sample Data (single)', self)
        layout.addWidget(single_button)
```

```
        single_button.setToolTip('Samples one data point')
        single_button.move(100,70)
        single_button.clicked.connect(self.single_sample)

        multi_button = QPushButton('Sample Data (10 Times)', self)
        layout.addWidget(multi_button)
        multi_button.setToolTip('Samples 10 data points')
        multi_button.move(100,70)
        multi_button.clicked.connect(self.multi_sample)

        calc_button = QPushButton('Calculate Metrics', self)
        layout.addWidget(calc_button)
        calc_button.setToolTip('Calculates metrics')
        calc_button.move(100,70)
        calc_button.clicked.connect(self.calc_metrics)

        # alarm message box
        self.alarm_dialog = QtWidgets.QErrorMessage()

        # buttons for changing alarm limits
        maxt_button = QPushButton('Set Max Temperature', self)
        layout.addWidget(maxt_button)
        maxt_button.setToolTip('Set Max Temperature')
        maxt_button.move(100,70)
        maxt_button.clicked.connect(self.set_max_temp)

        mint_button = QPushButton('Set Min Temperature', self)
        layout.addWidget(mint_button)
        mint_button.setToolTip('Set Min Temperature')
        mint_button.move(100,70)
        mint_button.clicked.connect(self.set_min_temp)

        maxh_button = QPushButton('Set Max Humidity', self)
        layout.addWidget(maxh_button)
        maxh_button.setToolTip('Set Max Humidity')
        maxh_button.move(100,70)
        maxh_button.clicked.connect(self.set_max_humidity)

        minh_button = QPushButton('Set Min Humidity', self)
        layout.addWidget(minh_button)
        minh_button.setToolTip('Set Min Humidity')
        minh_button.move(100,70)
        minh_button.clicked.connect(self.set_min_humidity)
```

```
        # close UI button
        exit_button = QPushButton('Exit UI', self)
        layout.addWidget(exit_button)
        exit_button.setToolTip('Exit UI')
        exit_button.move(100,70)
        exit_button.clicked.connect(self.close)


    # def _update_canvas(self):
    #     self._dynamic_ax.clear()
    #     t = np.linspace(0, 10, 101)
    #     # Shift the sinusoid as a function of time.
    #     self._dynamic_ax.plot(t, np.sin(t + time.time()))
    #     self._dynamic_ax.figure.canvas.draw()
    #     return

    def update_canvas_temps(self):
        temp_list, temp_times = db.get_all_temps(session, "f")
        humid_list, humid_times = db.get_all_humids(session)
        min_list = []
        max_list = []
        for temp in temp_list:
            min_list.append(self.temp_min_limit)
            max_list.append(self.temp_max_limit)
        # t_times = dates.date2num(temp_times)
        # h_times = dates.date2num(humid_times)
        self.dynamic_ax_temps.clear()
        self.dynamic_ax_temps.set_title("Temperature vs Time")
        self.dynamic_ax_temps.set_xlabel("Time")
        self.dynamic_ax_temps.set_ylabel("Temp (F)")
        self.dynamic_ax_temps.plot(temp_times, temp_list, label = "Measured")
        self.dynamic_ax_temps.plot(temp_times, max_list, '--', label = "Max
Temp")
        self.dynamic_ax_temps.plot(temp_times, min_list, '--', label = "Min
Temp")
        self.dynamic_ax_temps.figure.canvas.draw()
        return

    def update_canvas_humid(self):
        humid_list, humid_times = db.get_all_humids(session)
        min_list = []
        max_list = []
        for humidity in humid_list:
            min_list.append(self.humid_min_limit)
```

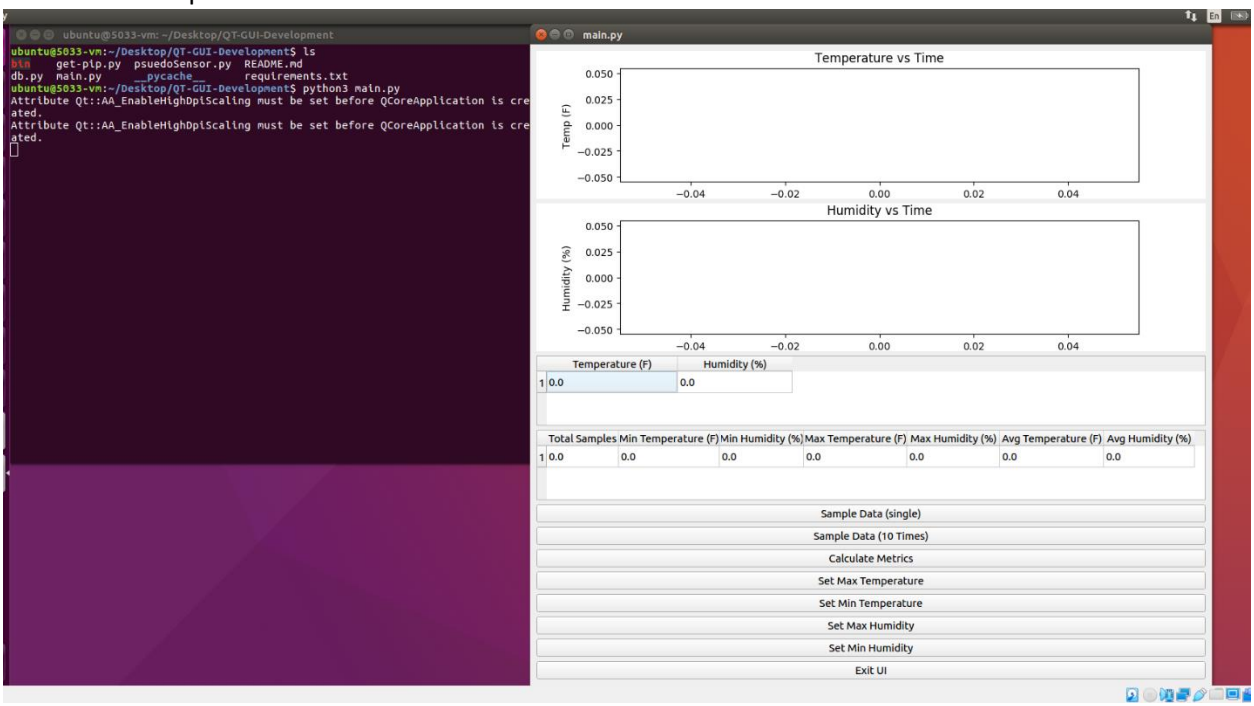
```
            max_list.append(self.humid_max_limit)
        self.dynamic_ax_humid.clear()
        self.dynamic_ax_humid.set_title("Humidity vs Time")
        self.dynamic_ax_humid.set_xlabel("Time")
        self.dynamic_ax_humid.set_ylabel("Humidity (%)")
        self.dynamic_ax_humid.plot(humid_times, humid_list, label = "Measured")
        self.dynamic_ax_humid.plot(humid_times, max_list, '--', label = "Max
Humidity")
        self.dynamic_ax_humid.plot(humid_times, min_list, '--', label = "Min
Humidity")
        self.dynamic_ax_humid.figure.canvas.draw()
        return

    @pyqtSlot()
    def single_sample(self):
        h,t = self.sample_data()
        self.current_table.setItem(0,0, QTableWidgetItem(str(t)))
        self.current_table.setItem(0,1, QTableWidgetItem(str(h)))
        print('sample', 'temp:', t, 'humidity:', h)
        return

    @pyqtSlot()
    def multi_sample(self):
        max = 10
        print("take 10 samples:")
        for i in range(max):
            h,t = self.sample_data()
            self.current_table.setItem(0,0, QTableWidgetItem(str(t)))
            self.current_table.setItem(0,1, QTableWidgetItem(str(h)))
            print('sample', i+1, 'temp:', t, 'humidity:', h)
            time.sleep(1)
        return

    @pyqtSlot()
    def calc_metrics(self):
        temp_list, temp_times = db.get_all_temps(session, "f")
        humid_list, humid_times = db.get_all_humids(session)
        # set total samples
        self.metrics_table.setItem(0,0, QTableWidgetItem(str(len(temp_list))))
        # min temp
        self.metrics_table.setItem(0,1, QTableWidgetItem(str(min(temp_list))))
        # min humidity
        self.metrics_table.setItem(0,2, QTableWidgetItem(str(min(humid_list))))
        # max temp
        self.metrics_table.setItem(0,3, QTableWidgetItem(str(max(temp_list))))
```

```
        # max humidity
        self.metrics_table.setItem(0,4, QTableWidgetItem(str(max(humid_list))))
        # avg temp
        self.metrics_table.setItem(0,5,
QTableWidgetItem(str(sum(temp_list)/len(temp_list))))
        # avg humidity
        self.metrics_table.setItem(0,6,
QTableWidgetItem(str(sum(humid_list)/len(humid_list))))
        return

    @pyqtSlot()
    def set_max_temp(self):
        i, okPressed = QInputDialog.getInt(self, "Get integer","Set Max Temp:",
28, 0, 100, 1)
        if okPressed:
            self.temp_max_limit = i
            print(self.temp_max_limit)

    @pyqtSlot()
    def set_min_temp(self):
        i, okPressed = QInputDialog.getInt(self, "Get integer","Set Min Temp
Value:", 28, 0, 100, 1)
        if okPressed:
            self.temp_min_limit = i
            print(self.temp_min_limit)

    @pyqtSlot()
    def set_max_humidity(self):
        i, okPressed = QInputDialog.getInt(self, "Get integer","Set Max Humidity
Value:", 28, 0, 100, 1)
        if okPressed:
            self.humid_max_limit = i
            print(self.humid_max_limit)

    @pyqtSlot()
    def set_min_humidity(self):
        i, okPressed = QInputDialog.getInt(self, "Get integer","Set Min Humidity
Value:", 28, 0, 100, 1)
        if okPressed:
            self.humid_min_limit = i
            print(self.humid_min_limi)

    # get sample of data from pseudo sensor
    def sample_data(self):
        ps = PseudoSensor()
```

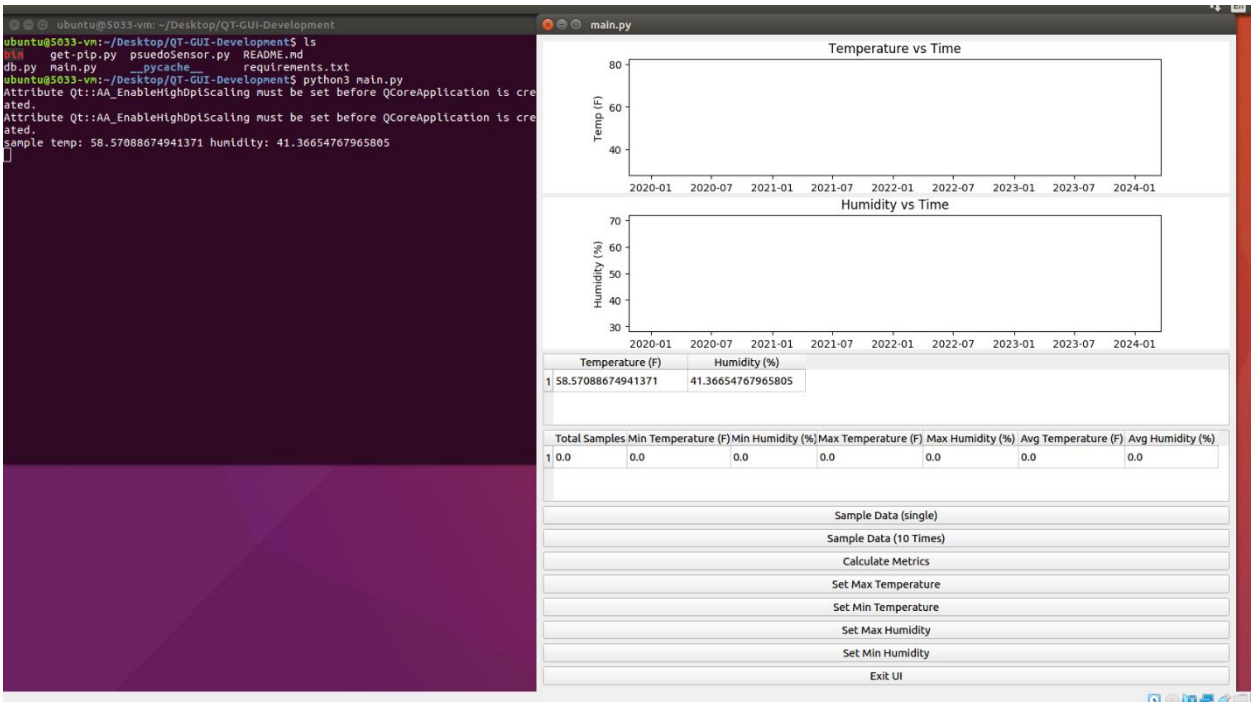
```
        h,temp_f = ps.generate_values()
        temp_c = (temp_f - 32) * 5.0/9.0
        now = datetime.now()
        db.add_temp(session, temp_f, temp_c, now)
        db.add_humidity(session, h, now)

        # check if we hit an alarm
        message = "None"
        if temp_f > self.temp_max_limit:
            message = "Oh no, Max temp of " + str(temp_max_limit) + "F exceeded!
Current: " + str(temp_f)
        elif temp_f < self.temp_min_limit:
            message = "Oh no, Min temp of " + str(temp_min_limit) + "F exceeded!
Current: " + str(temp_f)
        elif h > self.humid_max_limit:
            message = "Oh no, Max Humidity of " + str(humid_max_limit) + "%" + "
exceeded! Current: " + str(h)
        elif h < self.humid_min_limit:
            message = "Oh no, Min Humidity of " + str(humid_min_limit) + "%" + "
exceeded! Current: " + str(h)
        if message != "None":
            self.alarm_dialog.showMessage(message)

        return h, temp_f


if __name__ == "__main__":
    qapp = QtWidgets.QApplication(sys.argv)
    app = ApplicationWindow()
    app.show()
    qapp.exec_()
    # close db connection
    db.close(session)
```

Screen Captures:

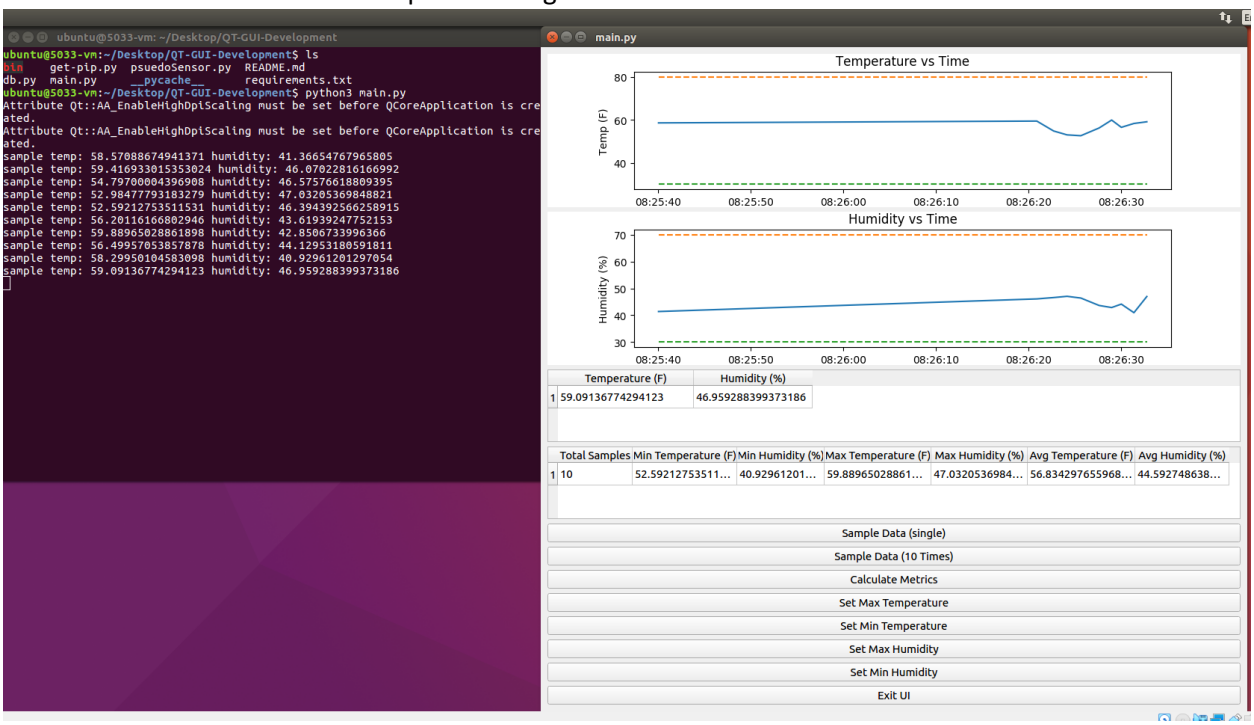
1. The UI at startup.



2. The UI after its first single data point reading.



3. The UI after it has calculated a 10 point average.



4. The UI after it has seen either a temperature or humidity alarm