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.