PART 1 IMPLEMENTATION/ASSUMPTIONS

Implementation details

The project has been implemented using Python, PyQt5, Qt Widgets, QT Charts and SQLite. Sensor sampling is implemented by the pseudo code provided by the class. All acquisitions are stored in the database and is persistent on disk. Listed below are the broad details of the implemented code.

Create an SQLite Database to store humidity/temperature readings

1. The schema for the database is: id (INTEGER), timestamp (TEXT, seconds since epoch), humidity (REAL, percentage), temperature (REAL, Fahrenheit).
2. Create methods:
    a. Create database and table if it does not exist
    b. Operations to count samples, insert, query and produce avg, min, max statistics.

Create the main Dialog UI file using Qt Designer

Create a Dialog with:

1. A Qt button to read, log and show single sample.
2. A Qt button to read 10 samples.
3. A Qt button to show statistics on the last 10 or fewer readings, including minimum, maximum and average in the units set by the user.
4. Line edits to enable the user to set alarm values for humidity and temperature. Valid values are between 0-100% and -20-100 degrees Fahrenheit.
5. A text widget to show values as they are acquired, queried.
6. A Qt button to open display widget to plot curves.
7. A Qt button to shutdown the application.

Implement the user interactions

1. Implement the user interactions for the buttons created on Qt Designer.
2. Implement the SQLite functions to populate and query the database.
3. Allow for setting changes (alarm, units).
4. Implement the graphing using Qt Charts.

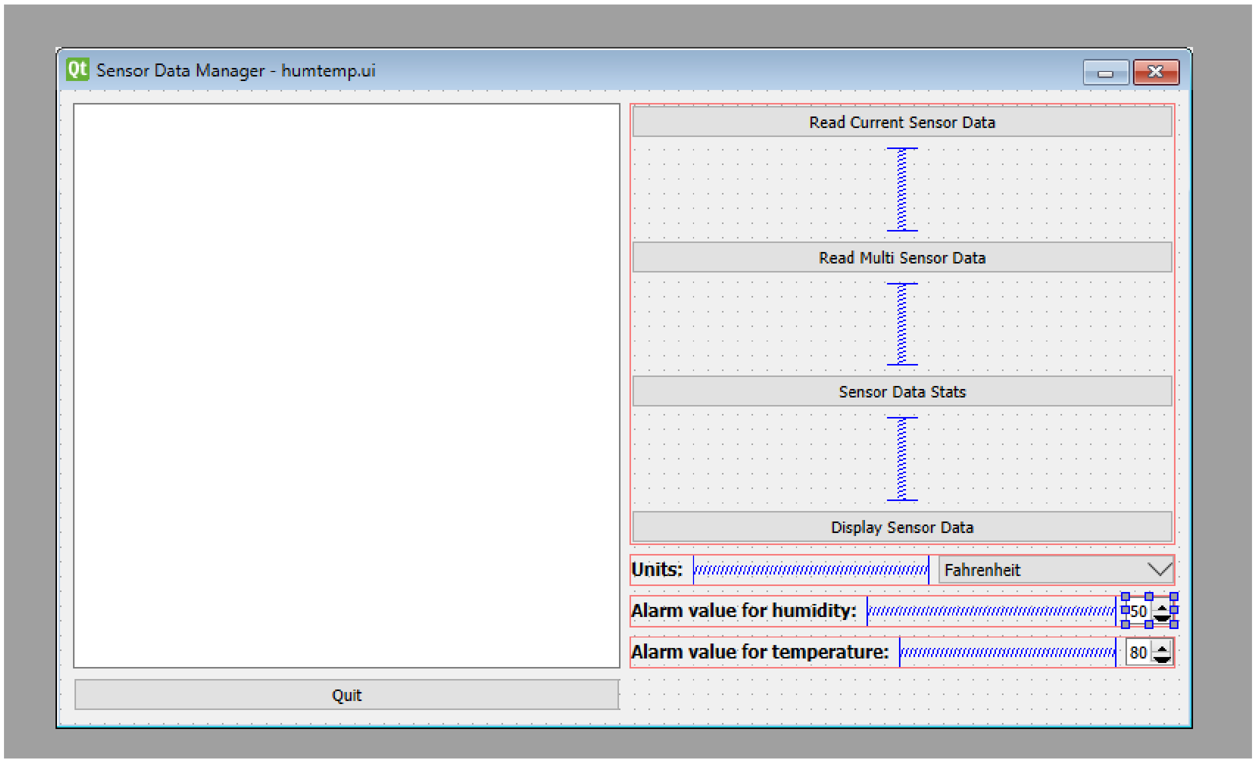
PART 2 – THE CODE

Sensor.py

```
# -*- coding: utf-8 -*-
"""
Created on Fri Jun  4 13:29:30 2021

@author: laura

"""
# Create ui file with designer and python code with : pyuic5 humtemp.ui > humtemp.py
import sys
import os
import datetime
import time
from PyQt5.QtWidgets import QDialog, QApplication, QVBoxLayout
from PyQt5.QtChart import QChart, QChartView, QLineSeries
from humtemp import Ui_sensorDialog as sensorDialog
from pseudoSensor import PseudoSensor
from humtempdb import HumTempDB

class SensorWindow(QDialog):
    """Sensor Window Dialog Class input database and sensor class"""
    def __init__(self, ps, db):
        super().__init__()
        self.ui = sensorDialog()
        self.ui.setupUi(self)

        self.db = db
        self.ps = ps

# Connections added
        self.ui.read1PB.clicked.connect(self.slot_read1)
        self.ui.read10PB.clicked.connect(self.slot_read10)
        self.ui.statsPB.clicked.connect(self.slot_stats)
        self.ui.displayPB.clicked.connect(self.slot_display)
        self.ui.quitPB.clicked.connect(self.slot_quit)
```

```
# Show it!
        self.show()


    def get_1sample(self):
        """ Acquire single sample and log it to screen,
        checking alarm values"""
        h,t = self.ps.generate_values()
        tmax = self.ui.alarmTSB.value()
        if self.ui.unitCB.currentIndex() == 1:
            t = ( 32. - t ) * 5. / 9.
            tmax = ( tmax * 9./5. ) + 32.
        hr = round(h)
        tr = round(t)
        ts = get_timestamp()
        myText = "At timestamp " + str(ts) + ":\n Humidity: " + \
        str(hr) + \
        " %; Temperature: " + \
        str(tr) + " " + \
        self.ui.unitCB.currentText() + \
        "\n"
        self.ui.textEdit.append(myText)
        self.db.insert_record(ts,hr,tr)

        if  h > self.ui.alarmHSB.value():
            myText = "<html><b>ALARM: Humidity exceeded\n<html><b>"
            self.ui.textEdit.append(myText)
        if  t > tmax:
            myText = "<html><b>ALARM: Temperature exceeded<\b><html>\n"
            self.ui.textEdit.append(myText)


    def slot_read1(self):
        """Slot to acquire a single sensor reading"""
        self.get_1sample()


    def slot_read10(self):
        """Slot to acquire 1 second space readings from sensor"""
        for i in range(10):
            self.get_1sample()
            time.sleep(1)

    def slot_stats(self):
        """Slot that produces stats (avg,min,max) for the last 10 readings"""
        print("stats")
        stats = self.db.stats_samples(10)
        myText = "<html style='color:blue'><i>Average humidity: </i></html>" + \
            str(round(stats[0])) + "\n"
        self.ui.textEdit.append(myText)
        myText = "<html style='color:blue'><i>Minimum humidity: </i></html>" + \
            str(round(stats[1])) + "\n"
        self.ui.textEdit.append(myText)
        myText = "<html style='color:blue'><i>Maximum humidity: </i></html>" + \
            str(round(stats[2])) + "\n"
        self.ui.textEdit.append(myText)
        myText = "<html style='color:green'><i>Average temperature: </i></html>" + \
            str(round(stats[3])) + "\n"
        self.ui.textEdit.append(myText)
        myText = "<html style='color:green'><i>Minimum temperature: </i></html>" + \
            str(round(stats[4])) + "\n"
        self.ui.textEdit.append(myText)
        myText = "<html style='color:green'><i>Maximum temperature: </i></html>" + \
            str(round(stats[5])) + "\n"
        self.ui.textEdit.append(myText)

    def slot_display(self):
        print("display")
        rows = self.db.query_samples(10)
        seriesHum = QLineSeries()
        seriesTemp = QLineSeries()
        for sample in rows:
            print(sample)
            seriesHum.append(sample[0],sample[2])
            seriesTemp.append(sample[0],sample[3])
        self.plot_series("Humidity",seriesHum)

        self.plot_series("Temperature",seriesTemp)

        print(rows)


    def plot_series(self,title,series):

        myChart = QChart()

        myChart.setTitle(title)

        myChart.addSeries(series)

        myChart.createDefaultAxes()

        myChartView = QChartView(myChart)

        myDisplayDialog = QDialog(self)

        layout = QVBoxLayout()

        layout.addWidget(myChartView)

        myDisplayDialog.setWindowTitle(title)

        myDisplayDialog.setModal(0)

        myDisplayDialog.setLayout(layout)

        myDisplayDialog.setMinimumHeight(800)

        myDisplayDialog.setMinimumWidth(800)

        myDisplayDialog.adjustSize()

        myDisplayDialog.show()


    def slot_quit(self):

        db.shutdown()

        self.done(0)


def get_timestamp():

    # ct stores current time

    ct = datetime.datetime.now()

    # ts store timestamp of current time

    ts = ct.timestamp()

    return ts


if __name__ == '__main__':

    database = r"C:\Users\laura\work\db\humtemp.db"

    db = HumTempDB(database)

    os.environ["QT_ENABLE_HIGHDPI_SCALING"] = "1"
    ps = PseudoSensor()


    app = QApplication(sys.argv)
    w = SensorWindow(ps,db)
    w.show()
    sys.exit(app.exec_())
```

humtemp.ui



```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>sensorDialog</class>
 <widget class="QDialog" name="sensorDialog">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>767</width>
    <height>431</height>
   </rect>
  </property>
  <property name="sizePolicy">
   <sizepolicy hsizetype="Expanding" vsizetype="Expanding">
    <horstretch>0</horstretch>
    <verstretch>0</verstretch>
   </sizepolicy>
  </property>
  <property name="minimumSize">
   <size>
    <width>767</width>
    <height>431</height>
   </size>
  </property>
  <property name="windowTitle">
   <string>Sensor Data Manager</string>
  </property>
  <layout class="QGridLayout" name="gridLayout">
   <item row="0" column="0" rowspan="4">
    <widget class="QTextEdit" name="textEdit">
     <property name="readOnly">
      <bool>true</bool>
     </property>
    </widget>
   </item>
   <item row="0" column="1">
    <layout class="QVBoxLayout" name="verticalLayout">
     <item>
      <widget class="QPushButton" name="read1PB">
       <property name="sizePolicy">
        <sizepolicy hsizetype="Expanding" vsizetype="Fixed">
         <horstretch>0</horstretch>
         <verstretch>0</verstretch>
        </sizepolicy>
       </property>
       <property name="text">
        <string>Read Current Sensor Data</string>
       </property>
      </widget>
     </item>
     <item>
      <spacer name="verticalSpacer">
       <property name="orientation">
        <enum>Qt::Vertical</enum>
       </property>
       <property name="sizeHint" stdset="0">
        <size>
         <width>20</width>
         <height>17</height>
        </size>
       </property>
      </spacer>
     </item>
     <item>
      <widget class="QPushButton" name="read10PB">
       <property name="sizePolicy">
        <sizepolicy hsizetype="Expanding" vsizetype="Fixed">
         <horstretch>0</horstretch>
         <verstretch>0</verstretch>
        </sizepolicy>
       </property>
       <property name="text">
        <string>Read Multi Sensor Data</string>
       </property>
      </widget>
     </item>
     <item>
      <spacer name="verticalSpacer_2">
       <property name="orientation">
        <enum>Qt::Vertical</enum>
       </property>
       <property name="sizeHint" stdset="0">
        <size>
         <width>20</width>
         <height>17</height>
        </size>
       </property>
      </spacer>
     </item>
     <item>
      <widget class="QPushButton" name="statsPB">
       <property name="sizePolicy">
        <sizepolicy hsizetype="Expanding" vsizetype="Fixed">
         <horstretch>0</horstretch>
         <verstretch>0</verstretch>
        </sizepolicy>
       </property>
       <property name="text">
        <string>Sensor Data Stats</string>
       </property>
      </widget>
     </item>
     <item>
      <spacer name="verticalSpacer_3">
       <property name="orientation">
        <enum>Qt::Vertical</enum>
       </property>
       <property name="sizeHint" stdset="0">
        <size>
         <width>20</width>
         <height>17</height>
        </size>
       </property>
      </spacer>
     </item>
     <item>
      <widget class="QPushButton" name="displayPB">
```

```
       <property name="sizePolicy">

        <sizepolicy hsizetype="Expanding" vsizetype="Fixed">

         <horstretch>0</horstretch>

         <verstretch>0</verstretch>

        </sizepolicy>

       </property>

       <property name="text">

        <string>Display Sensor Data</string>

       </property>

      </widget>

     </item>

    </layout>

   </item>

   <item row="1" column="1">

    <layout class="QHBoxLayout" name="horizontalLayout">

     <item>

      <widget class="QLabel" name="unitLB">

       <property name="text">

        <string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p&gt;&lt;span style=&quot; font-
size:9pt; font-
weight:600;&quot;&gt;Units:&lt;/span&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>

       </property>

      </widget>

     </item>

     <item>

      <spacer name="horizontalSpacer">

       <property name="orientation">

        <enum>Qt::Horizontal</enum>

       </property>

       <property name="sizeHint" stdset="0">

        <size>

         <width>13</width>

         <height>20</height>

        </size>

       </property>

      </spacer>

     </item>
```

```xml
 <item>
  <widget class="QComboBox" name="unitCB">
   <property name="sizePolicy">
    <sizepolicy hsizetype="Expanding" vsizetype="Fixed">
     <horstretch>0</horstretch>
     <verstretch>0</verstretch>
    </sizepolicy>
   </property>
   <property name="editable">
    <bool>false</bool>
   </property>
   <property name="currentText">
    <string>Fahrenheit</string>
   </property>
   <property name="currentIndex">
    <number>0</number>
   </property>
   <property name="maxVisibleItems">
    <number>2</number>
   </property>
   <property name="insertPolicy">
    <enum>QComboBox::InsertBeforeCurrent</enum>
   </property>
   <item>
    <property name="text">
     <string>Fahrenheit</string>
    </property>
   </item>
   <item>
    <property name="text">
     <string>Celsius</string>
    </property>
   </item>
  </widget>
 </item>
</layout>
```

```
   </item>
   <item row="2" column="1">
    <layout class="QHBoxLayout" name="horizontalLayout_2">
     <item>
      <widget class="QLabel" name="alarmHLB">
       <property name="text">
        <string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p&gt;&lt;span style=&quot; font-
size:9pt; font-weight:600;&quot;&gt;Alarm value for
humidity:&lt;/span&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
       </property>
      </widget>
     </item>
     <item>
      <spacer name="horizontalSpacer_2">
       <property name="orientation">
        <enum>Qt::Horizontal</enum>
       </property>
       <property name="sizeHint" stdset="0">
        <size>
         <width>13</width>
         <height>20</height>
        </size>
       </property>
      </spacer>
     </item>
     <item>
      <widget class="QSpinBox" name="alarmHSB">
       <property name="value">
        <number>50</number>
       </property>
      </widget>
     </item>
    </layout>
   </item>
   <item row="3" column="1">
    <layout class="QHBoxLayout" name="horizontalLayout_3">
     <item>
```

```
      <widget class="QLabel" name="alarmTLB">
       <property name="text">
        <string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p&gt;&lt;span style=&quot; font-
size:9pt; font-weight:600;&quot;&gt;Alarm value for
temperature:&lt;/span&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
       </property>
      </widget>
     </item>
     <item>
      <spacer name="horizontalSpacer_3">
       <property name="orientation">
        <enum>Qt::Horizontal</enum>
       </property>
       <property name="sizeHint" stdset="0">
        <size>
         <width>13</width>
         <height>20</height>
        </size>
       </property>
      </spacer>
     </item>
     <item>
      <widget class="QSpinBox" name="alarmTSB">
       <property name="value">
        <number>80</number>
       </property>
      </widget>
     </item>
    </layout>
   </item>
   <item row="4" column="0">
    <widget class="QPushButton" name="quitPB">
     <property name="text">
      <string>Quit</string>
     </property>
    </widget>
   </item>
```

```
  </layout>
 </widget>
 <resources/>
 <connections/>
</ui>
```

humtempdb.py

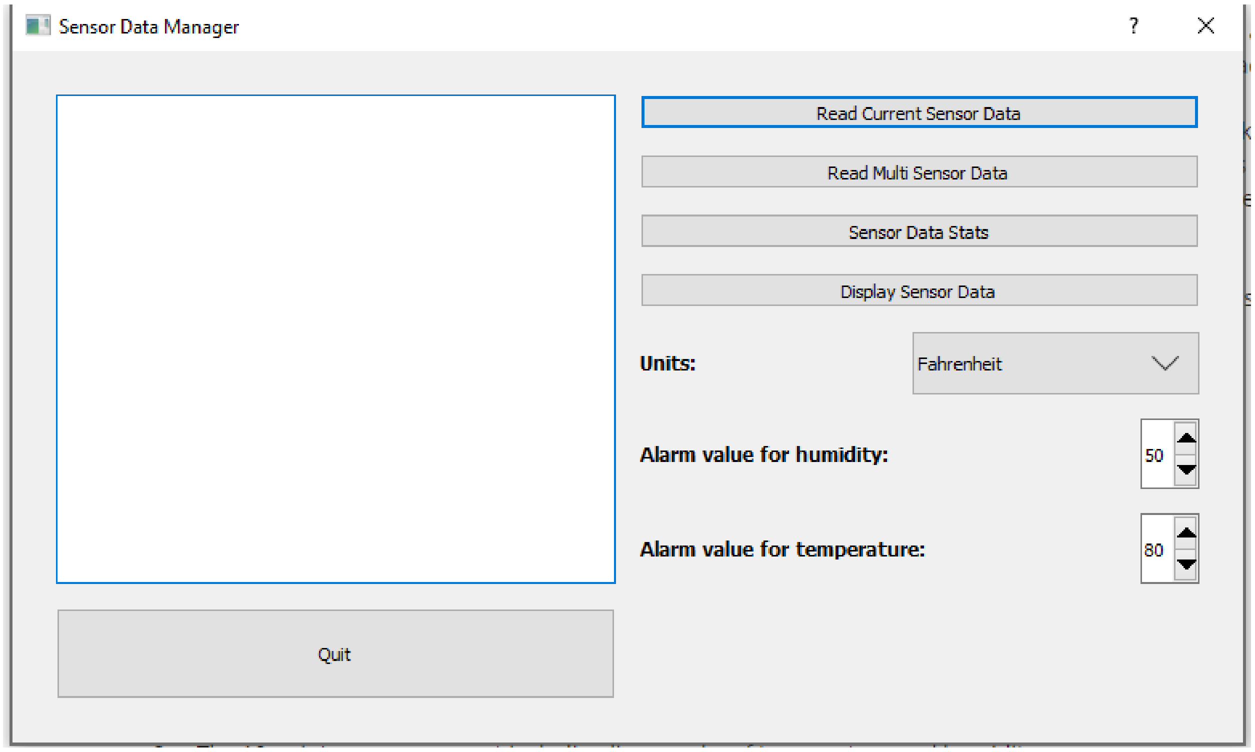
```
import sqlite3
from sqlite3 import Error

class HumTempDB:
    def __init__(self,database):
        self.database = database
        self.conn = None
        self.lastId = 0
        self.setup_table()

    def setup_table(self):
        """Setup the database/table"""
        sql_create_samples_table = """CREATE TABLE IF NOT EXISTS samples (
                                        id integer PRIMARY KEY,
                                        timestamp text,
                                        humidity integer,
                                        temperature integer); """
        # create a database connection
        self.conn = self.create_connection()
        # create tables
        if self.conn is not None:
            # create samples table
            self.create_table(sql_create_samples_table)
            self.lastId = self.total_samples()
        else:
            print("Error! cannot create the database connection.")

    def create_connection(self):
        """ create a database connection to a SQLite database """
        conn = None
        try:
            conn = sqlite3.connect(self.database)
            print(sqlite3.version)
        except Error as e:
            print(e)
        return conn

    def create_table(self,create_table_sql):
        """ create a table from the create_table_sql statement
        :param conn: Connection object
        :param create_table_sql: a CREATE TABLE statement
        :return:
        """
        try:
            c = self.conn.cursor()
            c.execute(create_table_sql)
        except Error as e:
            print(e)



    def insert_record(self,timestamp, humidity, temperature):
        """ Insert an id, timestamp, humidity, temperature record"""
        self.lastId  = self.lastId + 1

        sql = """INSERT INTO samples(id,timestamp,humidity,temperature)
                VALUES(?, ?, ?, ?) """
        cur = self.conn.cursor()
        cur.execute(sql,(self.lastId,timestamp,humidity,temperature))
        self.conn.commit()
        return cur.lastrowid

    def total_samples(self):
        """Count the # of samples stored in the database"""
        cur = self.conn.cursor()
        cur.execute("SELECT * FROM samples")
        results = cur.fetchall()

        return len(results)


    def query_samples(self,lastn):

        """Retrieve the last n samples"""

        cur = self.conn.cursor()

        idmin = self.lastId - lastn

        cur.execute("SELECT * FROM samples WHERE id > ? ", (idmin,))

        rows = cur.fetchall()

        return rows


    def stats_samples(self,lastn):

        """Retrieve stats for the last n samples"""

        cur = self.conn.cursor()

        sql = \

        """SELECT avg(humidity),min(humidity), max(humidity), avg(temperature),

        min(temperature), max(temperature)

        FROM samples

        WHERE id > ? """


        idmin = self.lastId - lastn

        findAll = cur.execute(sql,(idmin,))

        all = findAll.fetchone()

        print(all)


        return all


    def shutdown(self):

        """Shutdown database"""

        self.conn.commit()

        self.conn.close()


if __name__ == '__main__':

    database = r"C:\Users\laura\work\db\humtemp.db"

    db = HumTempDB(database)
```

pseudoSensor.py

```
import random

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 = 0

    t_range_index = 0

    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
```

PART 3 -SCREEN CAPTURES

1.The UI at startup

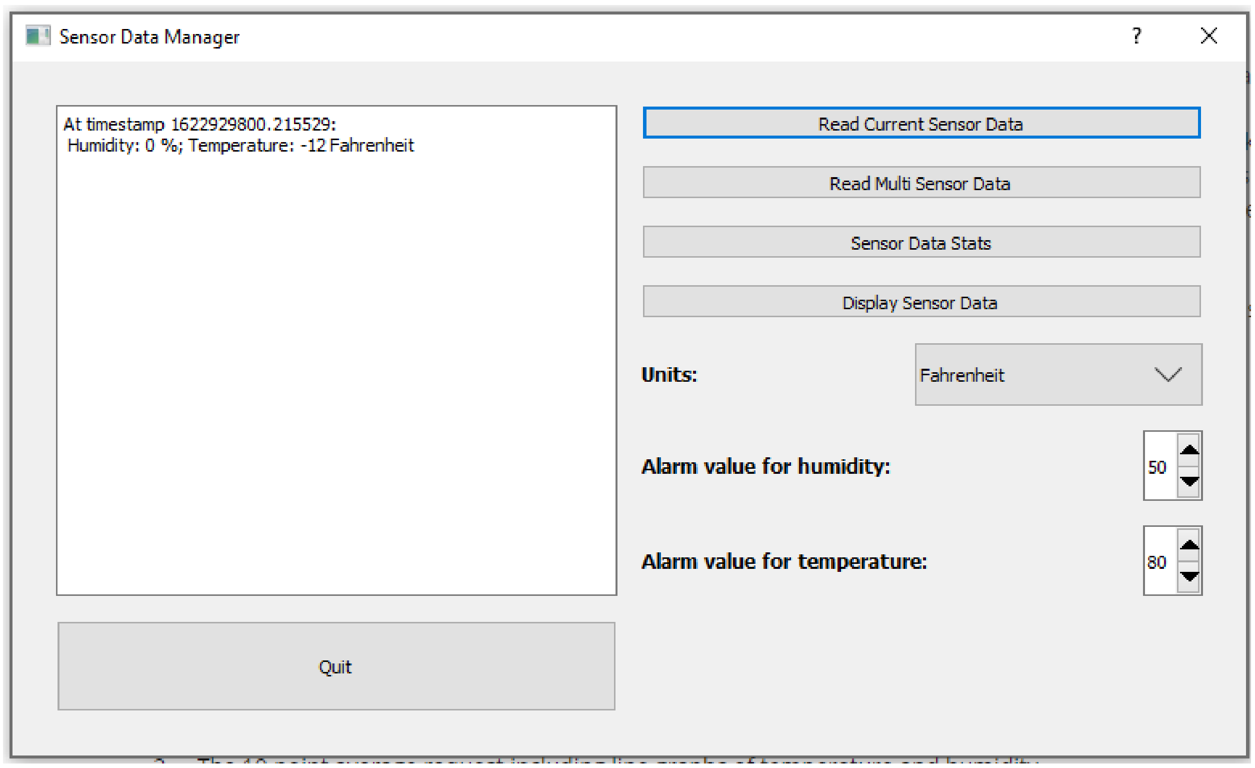


Units: Fahrenheit

Alarm value for humidity: 50

Alarm value for temperature: 80

2.The UI after is first single point reading



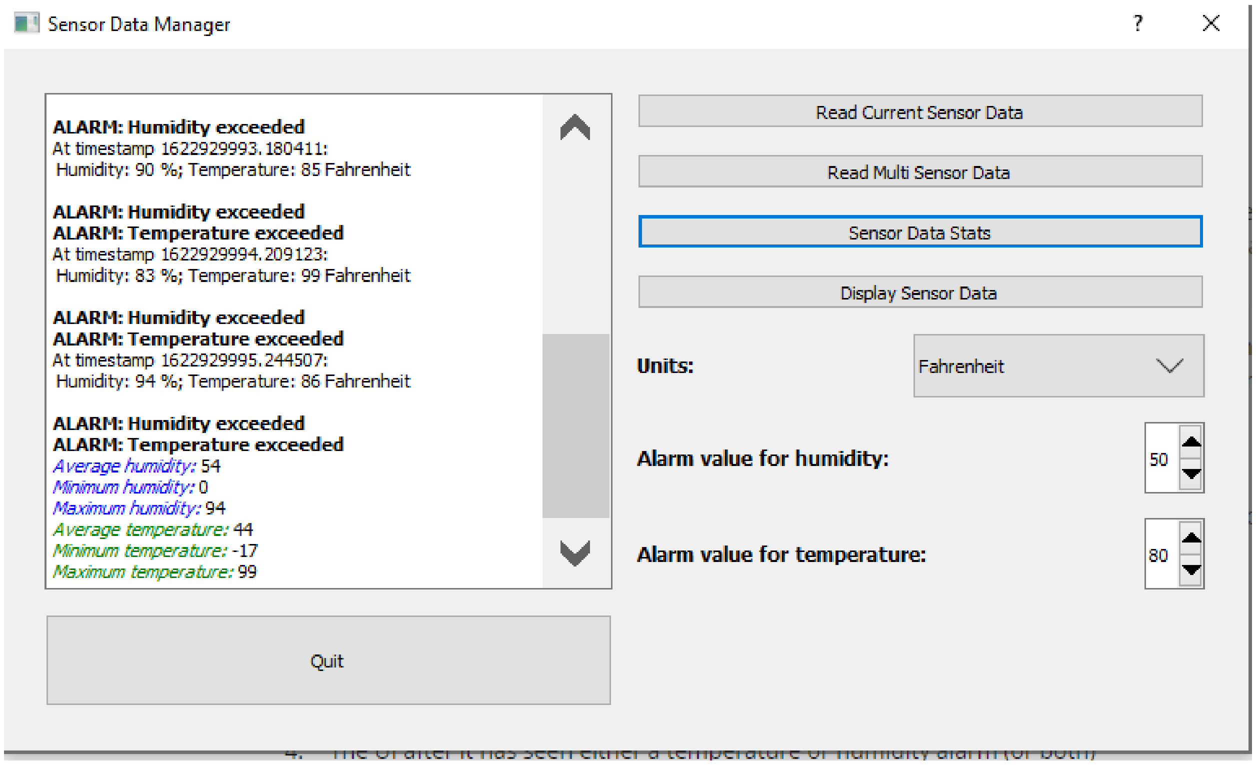
```
At timestamp 1622929800.215529:
 Humidity: 0 %; Temperature: -12 Fahrenheit
```

Units: Fahrenheit

Alarm value for humidity: 50

Alarm value for temperature: 80

3.The UI after the calculated point 10 average



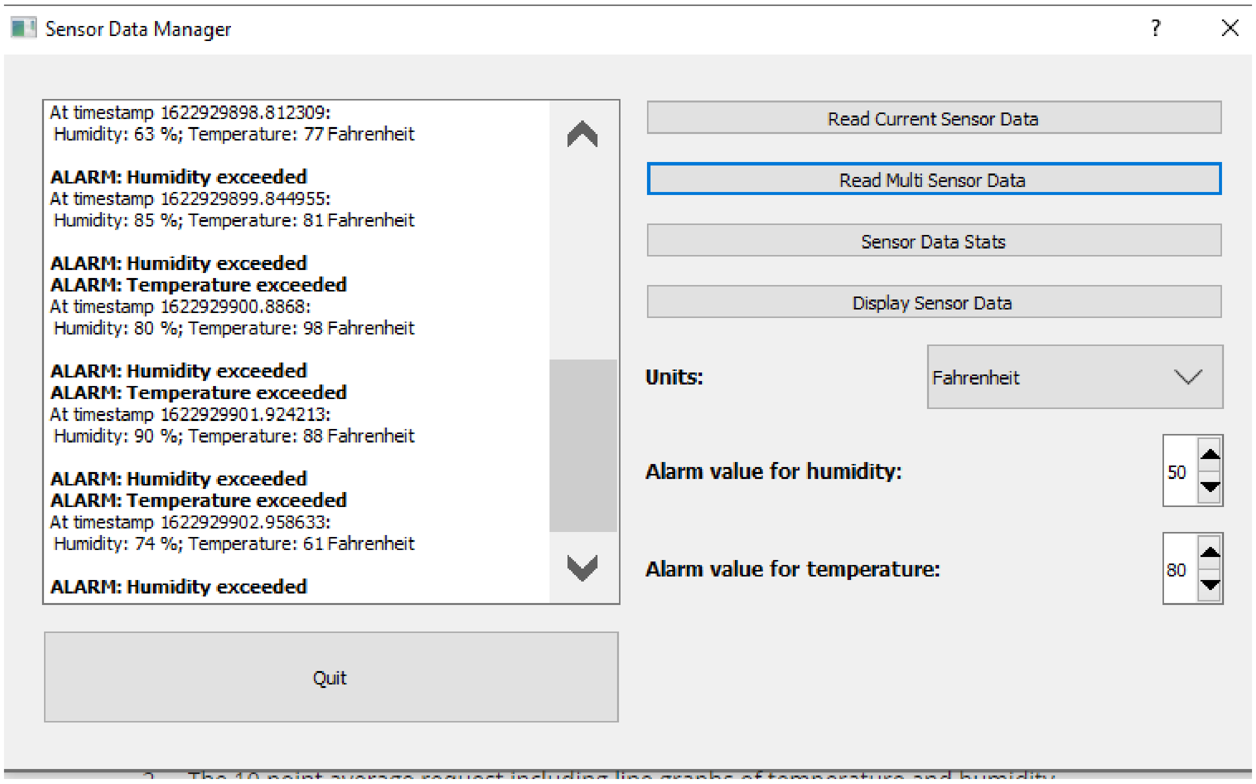
```
ALARM: Humidity exceeded
At timestamp 1622929993.180411:
 Humidity: 90 %; Temperature: 85 Fahrenheit

ALARM: Humidity exceeded
ALARM: Temperature exceeded
At timestamp 1622929994.209123:
 Humidity: 83 %; Temperature: 99 Fahrenheit

ALARM: Humidity exceeded
ALARM: Temperature exceeded
At timestamp 1622929995.244507:
 Humidity: 94 %; Temperature: 86 Fahrenheit

ALARM: Humidity exceeded
ALARM: Temperature exceeded
Average humidity: 54
Minimum humidity: 0
Maximum humidity: 94
Average temperature: 44
Minimum temperature: -17
Maximum temperature: 99
```

Units: Fahrenheit

Alarm value for humidity: 50

Alarm value for temperature: 80

4.The UI after an alarm



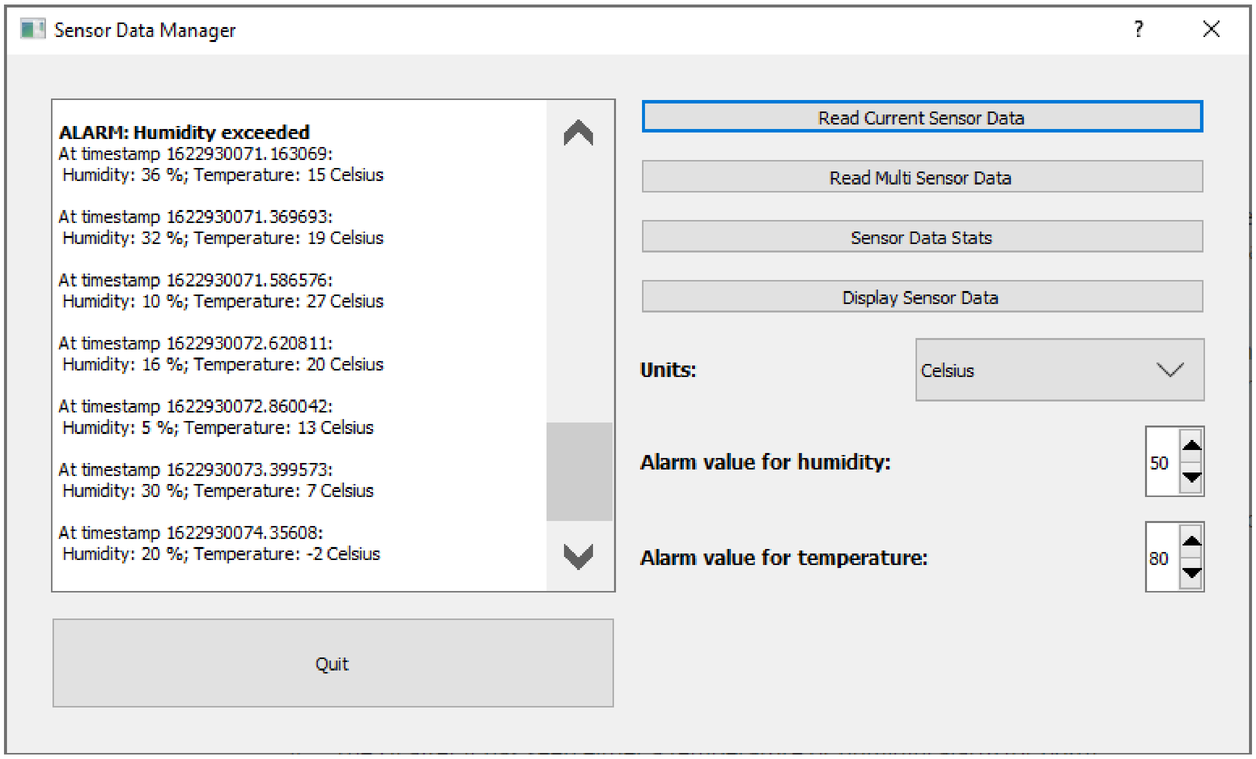
```
At timestamp 1622929898.812309:
 Humidity: 63 %; Temperature: 77 Fahrenheit

ALARM: Humidity exceeded
At timestamp 1622929899.844955:
 Humidity: 85 %; Temperature: 81 Fahrenheit

ALARM: Humidity exceeded
ALARM: Temperature exceeded
At timestamp 1622929900.8868:
 Humidity: 80 %; Temperature: 98 Fahrenheit

ALARM: Humidity exceeded
ALARM: Temperature exceeded
At timestamp 1622929901.924213:
 Humidity: 90 %; Temperature: 88 Fahrenheit

ALARM: Humidity exceeded
ALARM: Temperature exceeded
At timestamp 1622929902.958633:
 Humidity: 74 %; Temperature: 61 Fahrenheit

ALARM: Humidity exceeded
```

Units: Fahrenheit

Alarm value for humidity: 50

Alarm value for temperature: 80

Opt 1. The UI after changing units



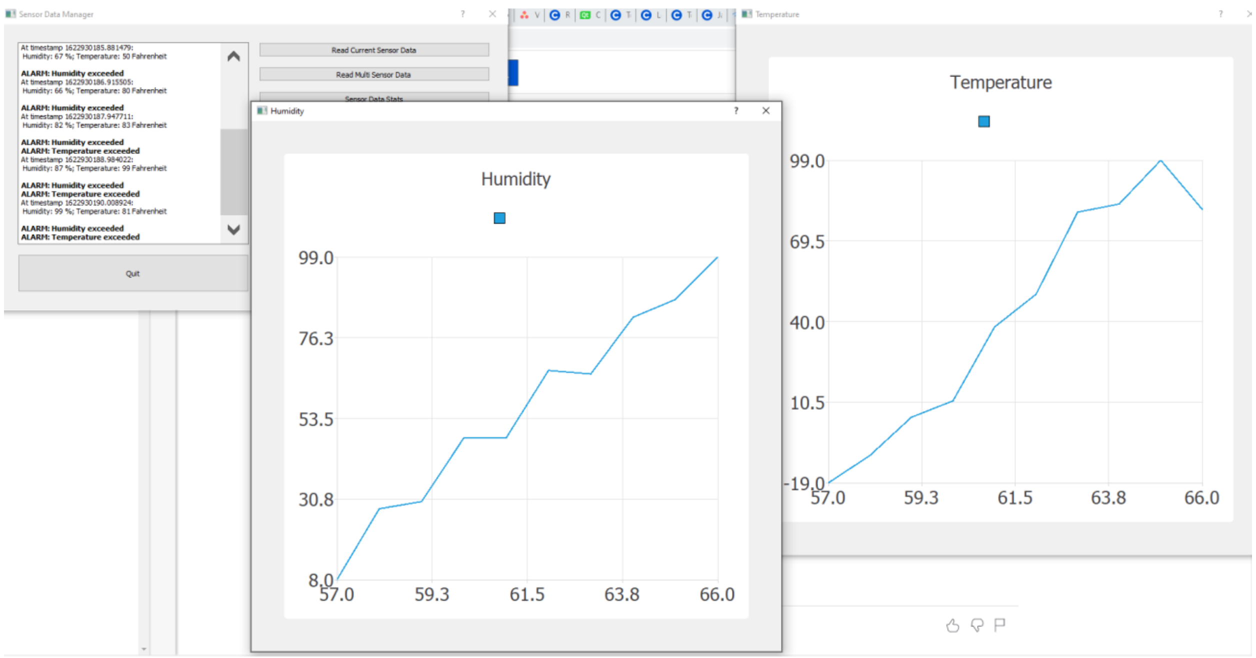
```
ALARM: Humidity exceeded
At timestamp 1622930071.163069:
 Humidity: 36 %; Temperature: 15 Celsius

At timestamp 1622930071.369693:
 Humidity: 32 %; Temperature: 19 Celsius

At timestamp 1622930071.586576:
 Humidity: 10 %; Temperature: 27 Celsius

At timestamp 1622930072.620811:
 Humidity: 16 %; Temperature: 20 Celsius

At timestamp 1622930072.860042:
 Humidity: 5 %; Temperature: 13 Celsius

At timestamp 1622930073.399573:
 Humidity: 30 %; Temperature: 7 Celsius

At timestamp 1622930074.35608:
 Humidity: 20 %; Temperature: -2 Celsius
```

Units: Celsius

Alarm value for humidity: 50

Alarm value for temperature: 80

Opt 2. Line graphs



```
At timestamp 1622930185.881479:
 Humidity: 67 %; Temperature: 50 Fahrenheit

ALARM: Humidity exceeded
At timestamp 1622930186.915505:
 Humidity: 66 %; Temperature: 80 Fahrenheit

ALARM: Humidity exceeded
At timestamp 1622930187.947711:
 Humidity: 82 %; Temperature: 83 Fahrenheit

ALARM: Humidity exceeded
ALARM: Temperature exceeded
At timestamp 1622930188.984022:
 Humidity: 87 %; Temperature: 99 Fahrenheit

ALARM: Humidity exceeded
ALARM: Temperature exceeded
At timestamp 1622930190.008924:
 Humidity: 99 %; Temperature: 81 Fahrenheit

ALARM: Humidity exceeded
ALARM: Temperature exceeded
```