Assumptions

The temperature is assumed to be from -20 and 100 Fahrenheit, I did not add support to work with Celsius.

I stored the data of the values in a global list called vals instead of using a database.

When setting to monitor 10 values every 10 seconds I chose to only display that values were loaded.

To make things more simply I did not sanitize the text input from the user, I also think that having a slider would more convenient.

The values of the limit is not set until the “Set” button is pressed.

To make things simpler for me and not having to implement threads or something similar to handle requesting data every second and having the program not hang, I used a library to get a timer. This timer reads the data every second as long as the variable “monitor” is set to True, which can be toggled by hitting the “Read10/sec” button.

SensorCode.py

```
from statistics import mean
import sys
from PyQt6 import QtWidgets, uic
from pseudoSensor import PseudoSensor
from MainWindow import Ui_MainWindow
import datetime
import multitimer

# Globals
# I decided not to use a database so we will store the values in a list with items of the form [humidity,
temperature, datetime]

monitor = False
vals = []
timer_count = 0
hmax = 80
tmax = 40
ps = PseudoSensor()



class MainWindow(QtWidgets.QMainWindow, Ui_MainWindow):
    def __init__(self, *args, obj=None, **kwargs):
        super(MainWindow, self).__init__(*args, **kwargs)
        self.setupUi(self)
        self.closeProgram.setCheckable(True)
        self.closeProgram.clicked.connect(self.closeProgram_was_clicked)
        self.read1.setCheckable(True)
        self.read1.clicked.connect(self.read1_was_clicked)
        self.Reset.setCheckable(True)
        self.Reset.clicked.connect(self.reset_was_clicked)
        self.read10.setCheckable(True)
        self.read10.clicked.connect(self.read10_was_clicked)
        self.getStats.setCheckable(True)
        self.getStats.clicked.connect(self.getStats_was_clicked)
        self.Set.setCheckable(True)
        self.Set.clicked.connect(self.Set_was_clicked)

    def Set_was_clicked(self):
        global monitor
        global hmax
        global tmax
        monitor = False
        hmax = int(self.humidityLimit.toPlainText())
        tmax = int(self.temperatureLimit.toPlainText())
        self.displayText.clear()
        self.displayText.append('hmax: ' + str(hmax))
        self.displayText.append('tmax: ' + str(tmax))
```

```
    # Close the program, we must stop the timer so the program actually finishes
    def closeProgram_was_clicked(self):
        timer.stop()
        self.close()

    # Make everything start from scratch

    def reset_was_clicked(self):
        vals = []
        monitor = False
        self.displayText.clear()

    # Read a single value

    def read1_was_clicked(self):

        h,t = ps.generate_values()
        vals.append([h, t, datetime.datetime.now()])
        print(vals)
        if(checkht(h,t)):
            return
        self.displayText.clear()
        self.displayText.append("H " + str(h))
        self.displayText.append("T " + str(t))

    # Enable the monitor flag so that we can gather data every second with the timer
    def read10_was_clicked(self):
        global monitor
        monitor = not monitor

# Function to get the Stats

    def getStats_was_clicked(self):
        global monitor
        monitor = False
        self.displayText.clear()
        h = []
        t = []
        v = []
        print(len(vals))

        # if we have no data, continue
        if len(vals) == 0:
            self.displayText.append("No data")
            return

        # Now we have data, so we must check if we have 10 or fewer
        elif len(vals) < 10:
            for i in range(len(vals)):
                h.append(vals[i][0])
                t.append(vals[i][1])
        else:
            v = vals[-10:]
```

```
            for i in range(10):
                h.append(v[i][0])
                t.append(v[i][1])
        # Now we can compute the stats

        self.displayText.append("H max: " + str(max(h)))
        self.displayText.append("H min: " + str(min(h)))
        self.displayText.append("H avg: " + str(mean(h)))
        self.displayText.append("T max: " + str(max(t)))
        self.displayText.append("T min: " + str(min(t)))
        self.displayText.append("T avg: " + str(mean(t)))



app = QtWidgets.QApplication(sys.argv)

window = MainWindow()

# A timer so that we can do a function every second
def timer_1():
        global timer_count
        timer_count = timer_count + 1
        print(timer_count)
        #global monitor
        if monitor:
            for i in range(10):
                h,t = ps.generate_values()
                vals.append([h, t, datetime.datetime.now()])
                print(vals)
                if(checkht(h,t)):
                    break

            window.displayText.append("Appended 10 values")

#CONFIGURING THE TIMER
timer = multitimer.MultiTimer(interval=1, function=timer_1, args=None, kwargs=None, count=999999,
runonstart=True)
timer.start()

def checkht(h, t):
        global monitor
        global hmax
        global tmax
        if (h < hmax) and (t < tmax):
            return False
        monitor = False
        window.displayText.clear()
        window.displayText.append("Reading out of Range")
        window.displayText.append("H: " + str(h))
        window.displayText.append("T: " + str(t))
        window.displayText.append("Hmax: " + str(hmax))
        window.displayText.append("Tmax: " + str(tmax))
        return True
```

```
window.show()
app.exec()
```

PseusoSensor.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
```

qtui.ui

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>MainWindow</class>
 <widget class="QMainWindow" name="MainWindow">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>711</width>
    <height>349</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>MainWindow</string>
  </property>
  <widget class="QWidget" name="centralwidget">
   <widget class="QPushButton" name="read1">
    <property name="geometry">
     <rect>
      <x>190</x>
      <y>230</y>
      <width>111</width>
      <height>32</height>
     </rect>
    </property>
    <property name="text">
     <string>Read 1 Val</string>
    </property>
   </widget>
   <widget class="QPushButton" name="getStats">
    <property name="geometry">
     <rect>
      <x>420</x>
      <y>230</y>
      <width>113</width>
      <height>32</height>
     </rect>
    </property>
    <property name="text">
     <string>Get Stats</string>
    </property>
```

```
</widget>
<widget class="QLabel" name="label">
 <property name="geometry">
  <rect>
   <x>10</x>
   <y>20</y>
   <width>121</width>
   <height>16</height>
  </rect>
 </property>
 <property name="text">
  <string>Temperature Limit</string>
 </property>
</widget>
<widget class="QLabel" name="label_2">
 <property name="geometry">
  <rect>
   <x>10</x>
   <y>80</y>
   <width>101</width>
   <height>16</height>
  </rect>
 </property>
 <property name="text">
  <string>Humidity Limit %</string>
 </property>
</widget>
<widget class="QPlainTextEdit" name="temperatureLimit">
 <property name="geometry">
  <rect>
   <x>10</x>
   <y>40</y>
   <width>51</width>
   <height>31</height>
  </rect>
 </property>
 <property name="plainText">
  <string>40</string>
 </property>
</widget>
<widget class="QTextBrowser" name="displayText">
 <property name="geometry">
  <rect>
   <x>230</x>
   <y>20</y>
   <width>256</width>
   <height>192</height>
  </rect>
 </property>
</widget>
<widget class="QPushButton" name="closeProgram">
 <property name="geometry">
  <rect>
   <x>250</x>
```

```
   <y>280</y>
   <width>113</width>
   <height>32</height>
  </rect>
 </property>
 <property name="text">
  <string>Close</string>
 </property>
</widget>
<widget class="QPlainTextEdit" name="humidityLimit">
 <property name="geometry">
  <rect>
   <x>10</x>
   <y>110</y>
   <width>41</width>
   <height>31</height>
  </rect>
 </property>
 <property name="plainText">
  <string>80</string>
 </property>
</widget>
<widget class="QPushButton" name="read10">
 <property name="geometry">
  <rect>
   <x>310</x>
   <y>230</y>
   <width>100</width>
   <height>32</height>
  </rect>
 </property>
 <property name="text">
  <string>Read 10/sec</string>
 </property>
</widget>
<widget class="QPushButton" name="Reset">
 <property name="geometry">
  <rect>
   <x>380</x>
   <y>280</y>
   <width>100</width>
   <height>32</height>
  </rect>
 </property>
 <property name="text">
  <string>Reset</string>
 </property>
</widget>
<widget class="QPushButton" name="Set">
 <property name="geometry">
  <rect>
   <x>120</x>
   <y>60</y>
   <width>100</width>
```

```
     <height>32</height>
    </rect>
   </property>
   <property name="text">
    <string>Set</string>
   </property>
  </widget>
 </widget>
 <widget class="QStatusBar" name="statusbar"/>
</widget>
<resources/>
<connections/>
</ui>
```

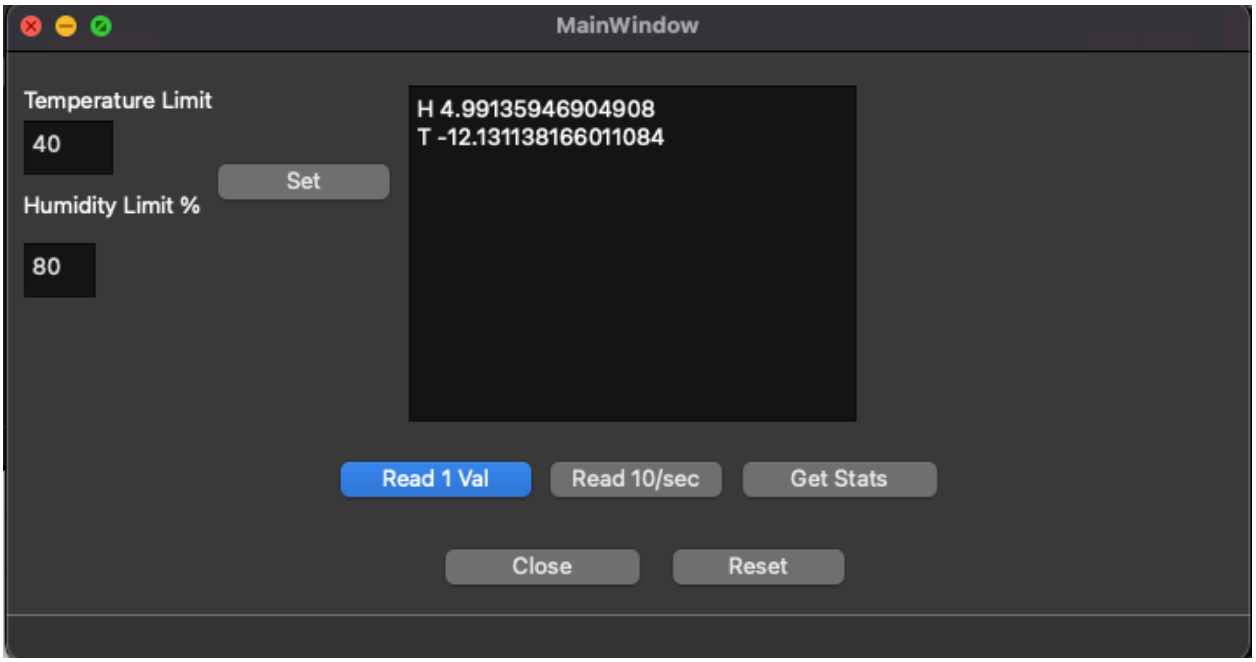
QTUI - Python Output

```
# Form implementation generated from reading ui file 'qtui.ui'
#
# Created by: PyQt6 UI code generator 6.2.0
#
# WARNING: Any manual changes made to this file will be lost when pyuic6 is
# run again.  Do not edit this file unless you know what you are doing.


from PyQt6 import QtCore, QtGui, QtWidgets


class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(711, 349)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.read1 = QtWidgets.QPushButton(self.centralwidget)
        self.read1.setGeometry(QtCore.QRect(190, 230, 111, 32))
        self.read1.setObjectName("read1")
        self.getStats = QtWidgets.QPushButton(self.centralwidget)
        self.getStats.setGeometry(QtCore.QRect(420, 230, 113, 32))
        self.getStats.setObjectName("getStats")
        self.label = QtWidgets.QLabel(self.centralwidget)
        self.label.setGeometry(QtCore.QRect(10, 20, 121, 16))
        self.label.setObjectName("label")
        self.label_2 = QtWidgets.QLabel(self.centralwidget)
        self.label_2.setGeometry(QtCore.QRect(10, 80, 101, 16))
        self.label_2.setObjectName("label_2")
        self.temperatureLimit = QtWidgets.QPlainTextEdit(self.centralwidget)
        self.temperatureLimit.setGeometry(QtCore.QRect(10, 40, 51, 31))
        self.temperatureLimit.setObjectName("temperatureLimit")
```

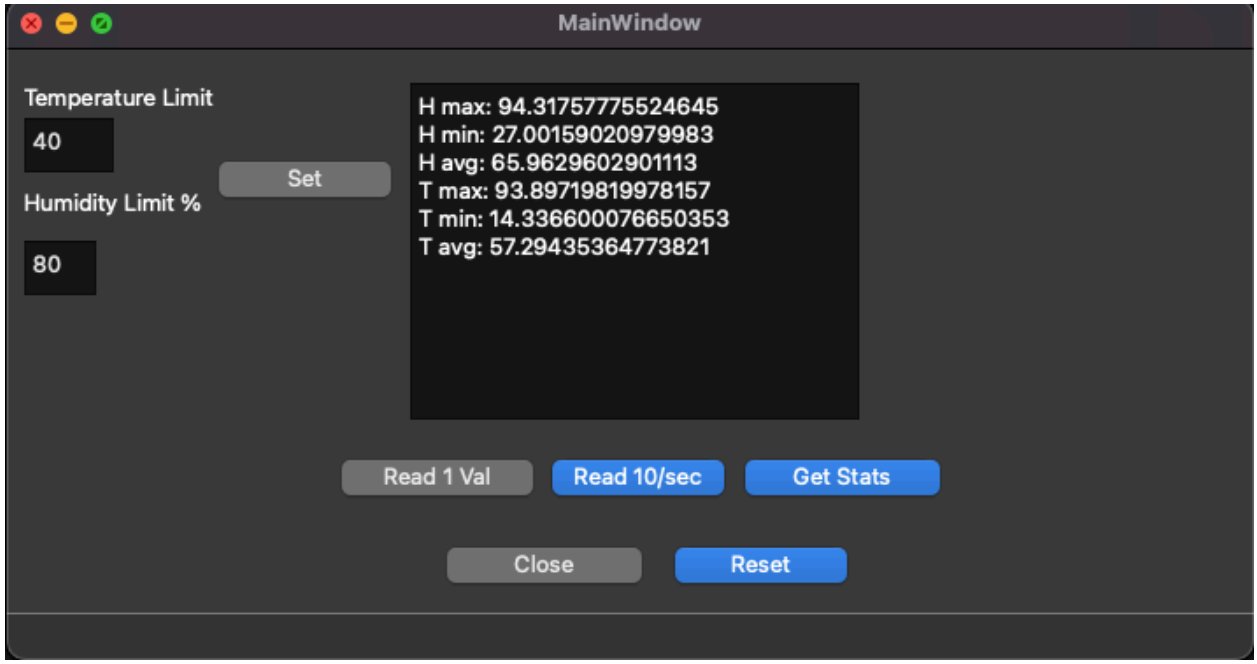
```
        self.displayText = QtWidgets.QTextBrowser(self.centralwidget)
        self.displayText.setGeometry(QtCore.QRect(230, 20, 256, 192))
        self.displayText.setObjectName("displayText")
        self.closeProgram = QtWidgets.QPushButton(self.centralwidget)
        self.closeProgram.setGeometry(QtCore.QRect(250, 280, 113, 32))
        self.closeProgram.setObjectName("closeProgram")
        self.humidityLimit = QtWidgets.QPlainTextEdit(self.centralwidget)
        self.humidityLimit.setGeometry(QtCore.QRect(10, 110, 41, 31))
        self.humidityLimit.setObjectName("humidityLimit")
        self.read10 = QtWidgets.QPushButton(self.centralwidget)
        self.read10.setGeometry(QtCore.QRect(310, 230, 100, 32))
        self.read10.setObjectName("read10")
        self.Reset = QtWidgets.QPushButton(self.centralwidget)
        self.Reset.setGeometry(QtCore.QRect(380, 280, 100, 32))
        self.Reset.setObjectName("Reset")
        self.Set = QtWidgets.QPushButton(self.centralwidget)
        self.Set.setGeometry(QtCore.QRect(120, 60, 100, 32))
        self.Set.setObjectName("Set")
        MainWindow.setCentralWidget(self.centralwidget)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "MainWindow"))
        self.read1.setText(_translate("MainWindow", "Read 1 Val"))
        self.getStats.setText(_translate("MainWindow", "Get Stats"))
        self.label.setText(_translate("MainWindow", "Temperature Limit"))
        self.label_2.setText(_translate("MainWindow", "Humidity Limit %"))
        self.temperatureLimit.setPlainText(_translate("MainWindow", "40"))
        self.closeProgram.setText(_translate("MainWindow", "Close"))
        self.humidityLimit.setPlainText(_translate("MainWindow", "80"))
        self.read10.setText(_translate("MainWindow", "Read 10/sec"))
        self.Reset.setText(_translate("MainWindow", "Reset"))
        self.Set.setText(_translate("MainWindow", "Set"))
```

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 (or both)