Implementation/Assumption Notes

Virtual Network Connect (VNC) was used to connect from a Windows 10 laptop to a Raspberry PI 4. No problems were encountered using the Raspberry PI in this mode. First, Qt Designer was installed on the Raspberry PI. It was installed in a different directory that the one described in class. It was installed in the /usr/lib/qt5 directory. The initial layout was done in qt designer. However, I could not find how to connect a clicked button action to a button so I ended up coding the “on button click” function in the generated python code.

The ui form was made with 4 push button across the top, a Single Read button, a Read 10 button, a AVG/MIN/MAX button and a close button. There are two spin boxes , one for setting the Humidity Alarm and one for setting the Temperature Alarm. A radio button is between the Humidity spin box and the Temperature spin box. The alarms are set by clicked the radio button. To turn off the alarm, the button is pressed again. Python code was made from the ui file with the “pyuic5 project1.ui > dialog.py” command. When the Single Read button is pressed, a time stamp is read, the humidity and temperature is read from the psuedosensor. The alarm is tested to see if it is on. If so, the humidity and temperature that have been read are tested with the humidity and temperature alarm settings. If the humidity and temperature that where read from the sensor exceed the alarm settings, alarm text is added to the timestamp, humidity and temperature text. This text is output to the display box. The text is also appended to a storage structure.

The Read 10 button operates in a very similar fashion to the Single Read button except that it is in a loop that iterates 10 times.

The AVG/Min/Max function read the humidity and temperature ten time, computes an average humidity and temperature, finds the minimum and maximum of the ten values read and outputs them to the display box.

The Close button closes the application.

Rather than repeatedly running through the qt designer application, the generated code was modified to resize the form and display box and to add code to implement the “on button clicked” functionality. Despite an extensive web search, I could not find a way to turn the alarm text red.

The application is implemented in 3 files. The psuedoSensor.py file implements the sensor simulation, the project1app.py which launches the ui and the dialog.py file which implements the ui. The dialog.py file includes the ui form and the code to implement the functionality of the form.

Project1app.py

```
import sys
from PyQt5.QtWidgets import QDialog, QApplication
from dialog import MyDialog

class AppWindow(QDialog):
    def __init__(self):
        super().__init__()
        self.ui = MyDialog()
        self.ui.setupUi(self)
        self.show()

app = QApplication(sys.argv)
w = AppWindow()
w.show()
sys.exit(app.exec_())
```

dialog.py

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'project1.ui'
#
# Created by: PyQt5 UI code generator 5.11.3
#
# WARNING! All changes made in this file will be lost!

import time;
from psuedoSensor import PseudoSensor
ps = PseudoSensor()
alarmOn = False
storeStruct = []

from PyQt5 import QtCore, QtGui, QtWidgets

class MyDialog(object):
    def setupUi(self, Form):
        Form.setObjectName("Form")
        # The following line was modifed from the UI file to resize the form
        Form.resize(650, 500)
        self.singleRead = QtWidgets.QPushButton(Form)
```

```
self.singleRead.setGeometry(QtCore.QRect(70, 20, 99, 30))
self.singleRead.setAutoDefault(False)
self.singleRead.setDefault(False)
self.singleRead.setObjectName("singleRead")
# The following line was added to the file made from the UI
self.singleRead.clicked.connect(self.singleReadAction)
self.timeLable = QtWidgets.QLabel(Form)
self.timeLable.setGeometry(QtCore.QRect(80, 140, 68, 22))
self.timeLable.setObjectName("timeLable")
self.humidityAlarmLabel = QtWidgets.QLabel(Form)
self.humidityAlarmLabel.setGeometry(QtCore.QRect(130, 60, 121, 22))
self.humidityAlarmLabel.setObjectName("humidityAlarmLabel")
self.TemperatureAlarmLabel = QtWidgets.QLabel(Form)
self.TemperatureAlarmLabel.setGeometry(QtCore.QRect(380, 60, 151, 22))
self.TemperatureAlarmLabel.setObjectName("TemperatureAlarmLabel")
self.read10 = QtWidgets.QPushButton(Form)
self.read10.setGeometry(QtCore.QRect(190, 20, 99, 30))
self.read10.setObjectName("read10")
# The following line was added to the file made from the UI
self.read10.clicked.connect(self.read10Action)
self.average = QtWidgets.QPushButton(Form)
self.average.setGeometry(QtCore.QRect(310, 20, 99, 30))
self.average.setObjectName("average")
# The following line was added to the file made from the UI
self.average.clicked.connect(self.averageAction)
self.close = QtWidgets.QPushButton(Form)
self.close.setGeometry(QtCore.QRect(440, 20, 99, 30))
self.close.setObjectName("close")
# The following line was added to the file made from the UI
self.close.clicked.connect(self.closeAction)
self.displayValues = QtWidgets.QTextBrowser(Form)
# The follwong line was modified from the UI to resize the display values box
self.displayValues.setGeometry(QtCore.QRect(60, 170, 550, 321))
self.displayValues.setObjectName("displayValues")
self.tempLabel = QtWidgets.QLabel(Form)
self.tempLabel.setGeometry(QtCore.QRect(270, 140, 101, 22))
self.tempLabel.setObjectName("tempLabel")
self.commentLabel = QtWidgets.QLabel(Form)
self.commentLabel.setGeometry(QtCore.QRect(380, 140, 81, 22))
self.commentLabel.setObjectName("commentLabel")
self.enterHmdAlrm = QtWidgets.QDoubleSpinBox(Form)
self.enterHmdAlrm.setGeometry(QtCore.QRect(150, 90, 71, 32))
```

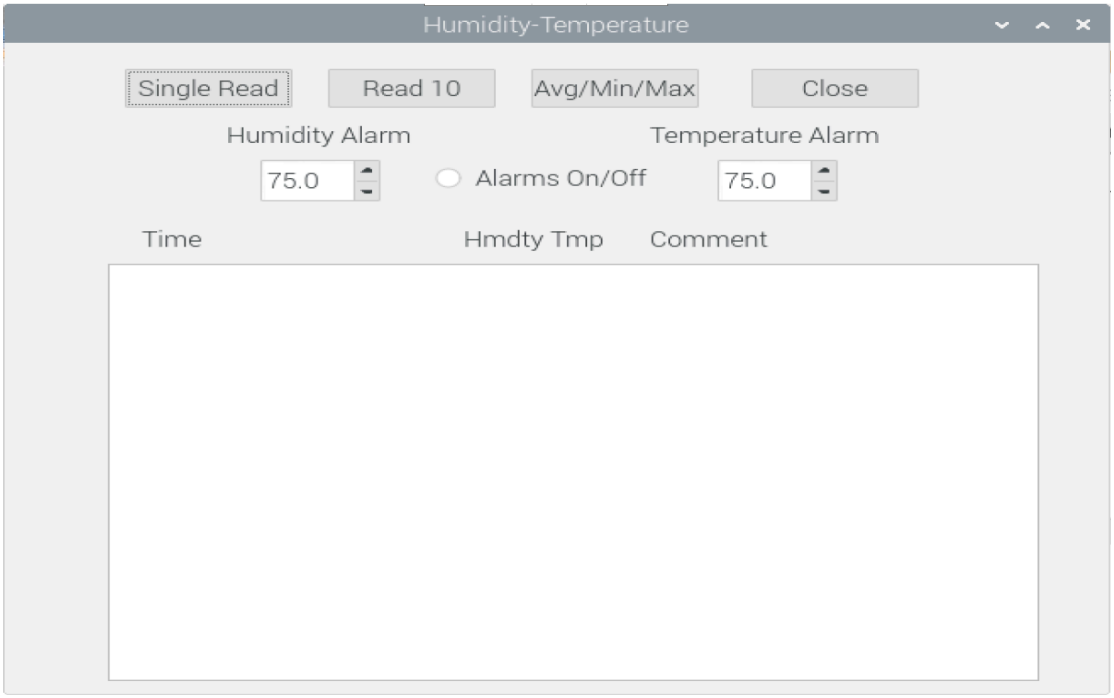
```
        self.enterHmdAlrm.setDecimals(1)
        self.enterHmdAlrm.setMaximum(100.0)
        self.enterHmdAlrm.setSingleStep(0.1)
        self.enterHmdAlrm.setProperty("value", 75.0)
        self.enterHmdAlrm.setObjectName("enterHmdAlrm")
        self.enterTmpAlrm = QtWidgets.QDoubleSpinBox(Form)
        self.enterTmpAlrm.setGeometry(QtCore.QRect(420, 90, 71, 32))
        self.enterTmpAlrm.setDecimals(1)
        self.enterTmpAlrm.setMinimum(-20.0)
        self.enterTmpAlrm.setSingleStep(0.1)
        self.enterTmpAlrm.setProperty("value", 75.0)
        self.enterTmpAlrm.setObjectName("enterTmpAlrm")
        self.AlarmOnOff = QtWidgets.QRadioButton(Form)
        self.AlarmOnOff.setGeometry(QtCore.QRect(250, 90, 141, 27))
        self.AlarmOnOff.setObjectName("AlarmOnOff")
        self.AlarmOnOff.setChecked(False)
        # The following line was added to the file made from the UI
        self.AlarmOnOff.toggled.connect(self.onClicked)

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

    def retranslateUi(self, Form):
        _translate = QtCore.QCoreApplication.translate
        Form.setWindowTitle(_translate("Form", "Humidity-Temperature"))
        self.singleRead.setText(_translate("Form", "Single Read"))
        self.timeLable.setText(_translate("Form", "Time"))
        self.humidityAlarmLabel.setText(_translate("Form", "Humidity Alarm"))
        self.TemperatureAlarmLabel.setText(_translate("Form", "Temperature Alarm"))
        self.read10.setText(_translate("Form", "Read 10"))
        # The following line was modified from the file made from the UI
        self.average.setText(_translate("Form", "Avg/Min/Max"))
        self.close.setText(_translate("Form", "Close"))
        self.tempLabel.setText(_translate("Form", "<html><head/><body><p>Hmdty
Tmp</p></body></html>"))
        self.commentLabel.setText(_translate("Form",
"<html><head/><body><p>Comment</p></body></html>"))
        self.AlarmOnOff.setText(_translate("Form", "Alarms On/Off"))

# When the Alarms On/Off radio button is clicked this codes checks to see
# if the radio button is on or off. It sets the alarm on/off flag accordingly
# and sends an alarms on/off message to the output window with a timestamp
```

```
    def onClicked(MyDialog):
        ts = time.gmtime()
        displayStr = time.strftime("%Y-%m-%d %H:%M:%S   ",ts)
        global alarmOn
        if MyDialog.AlarmOnOff.isChecked():
            alarmOn = True
            displayStr = displayStr + "                    Alarms On"
        else:
            alarmOn = False
            displayStr = displayStr + "                    Alarms Off"
        MyDialog.displayValues.append(displayStr)

        #store the event in the storage structure
        global storeStruct
        storeStruct.append(displayStr)

# Output a single reading of a Time Stamp, Humidity and Temperature

    def singleReadAction(MyDialog):
        ts = time.gmtime()
        h,t = ps.generate_values()
        displayStr = time.strftime("%Y-%m-%d %H:%M:%S   ",ts) + str(round(h,1)) + "%
" + str(round(t,1)) \
                    + " deg  Single Read"

        # Test to set if the alarms are set. If so, test to see if the humidity and temperature
        # are above the alarm threshold and if so, output an alarm message
        if alarmOn:
            humidityAlarm = MyDialog.enterHmdAlrm.value()
            tempAlarm = MyDialog.enterTmpAlrm.value()
            if (t >= tempAlarm) and (h >= humidityAlarm):
                alarmTxt = " Humidity/Temp Alarm"
            elif (t >= tempAlarm):
                alarmTxt = " Temp Alarm"
            elif (h >= humidityAlarm):
                alarmTxt = " Humidity Alarm"
            else:
                alarmTxt = " "
            displayStr = displayStr + alarmTxt
        MyDialog.displayValues.append(displayStr)
```

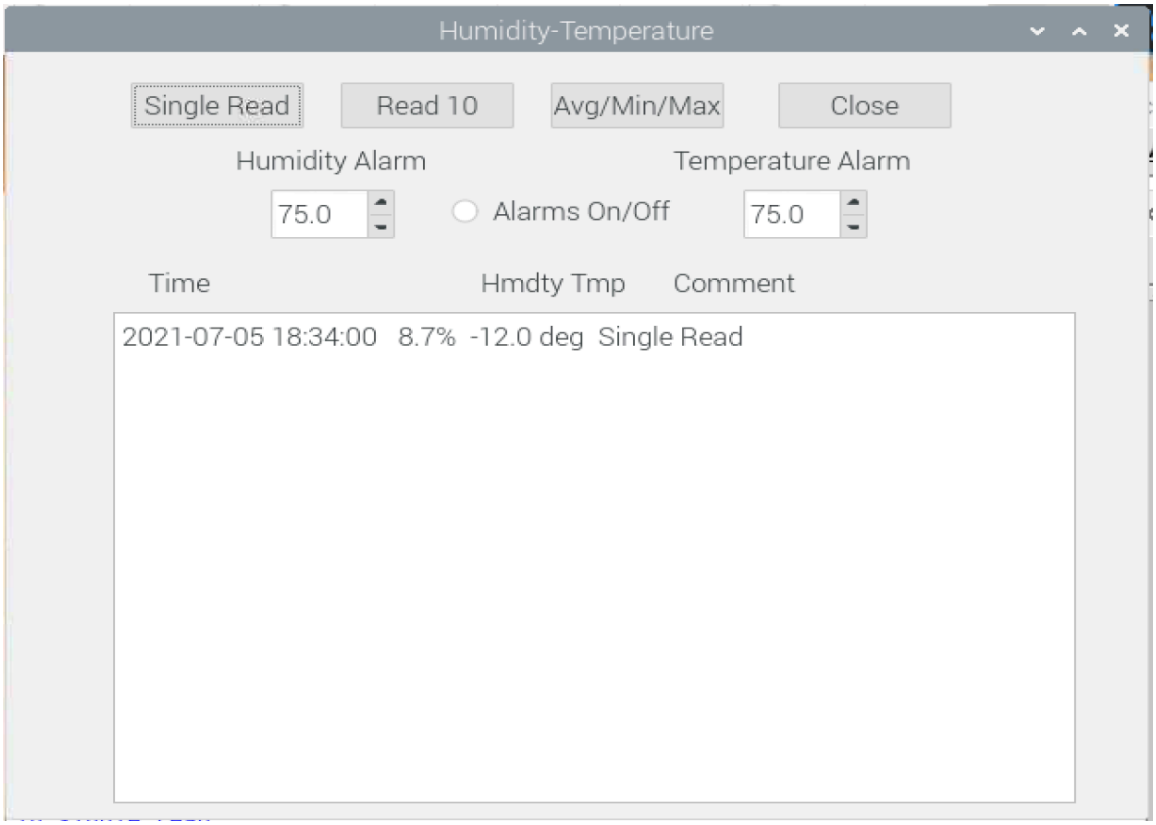
```
        #store the values in the storage structure
        global storeStruct
        storeStruct.append(displayStr)

# Read 10 Time Stamps - Humidity/Temperature values with 1 second between each
read.
    def read10Action(MyDialog):
        global storeStruct
        for i in range(1,11):
            ts = time.gmtime()
            h,t = ps.generate_values()
            displayStr = time.strftime("%Y-%m-%d %H:%M:%S   ",ts) + str(round(h,1)) +
"%  " + str(round(t,1)) \
                        + " deg " + str(i)

            # Test to set if the alarms are set. If so, test to see if the humidity and temperature
            # are above the alarm threshold and if so, output an alarm message
            if alarmOn:
                humidityAlarm = MyDialog.enterHmdAlrm.value()
                tempAlarm = MyDialog.enterTmpAlrm.value()
                if (t >= tempAlarm) and (h >= humidityAlarm):
                    alarmTxt = " Humidity/Temp Alarm"
                elif (t >= tempAlarm):
                    alarmTxt = " Temp Alarm"
                elif (h >= humidityAlarm):
                    alarmTxt = " Humidity Alarm"
                else:
                    alarmTxt = " "
            displayStr = displayStr + alarmTxt
            MyDialog.displayValues.append(displayStr)

            #store the event in the storage structure
            storeStruct.append(displayStr)

            time.sleep(1)

        displayStr = time.strftime("%Y-%m-%d %H:%M:%S   ",ts) + "
Done"
        MyDialog.displayValues.append(displayStr)

# Calculate the average, the minimum and maximum of 10 readings
    def averageAction(MyDialog):
```

```
        hSum = 0
        hMin = 101.0
        hMax = -.1
        tSum = 0
        tMin = 101.0
        tMax = -21.0
        for ii in range(1,10):
            tss = time.gmtime()
            hh,tt = ps.generate_values()

            # Calulate the sum of the 10 readings
            hSum = hh + hSum
            tSum = tt + tSum

            # Test to see if the current reading is greater than the current maximum. If so, set
the
            # maximum to the current maximum. Do likewise for the minimum.
            if tt > tMax:
                tMax = tt
            if hh > hMax:
                hMax = hh
            if hh < hMin:
                hMin = hh
            if tt < tMin:
                tMin = tt

            # sleep for 1 second
            time.sleep(1)

        #calculate the average of the ten readings
        hAvg = hSum/10
        tAvg = tSum/10

        #Display the average humidity and temperature
        displayStr = time.strftime("%Y-%m-%d %H:%M:%S   ",tss) + str(round(hAvg,1))
+ \
                    "%  Avg  " + str(round(tAvg,1)) + " deg  Avg"
        MyDialog.displayValues.append(displayStr)


        #store the event in the storage structure
        global storeStruct
```

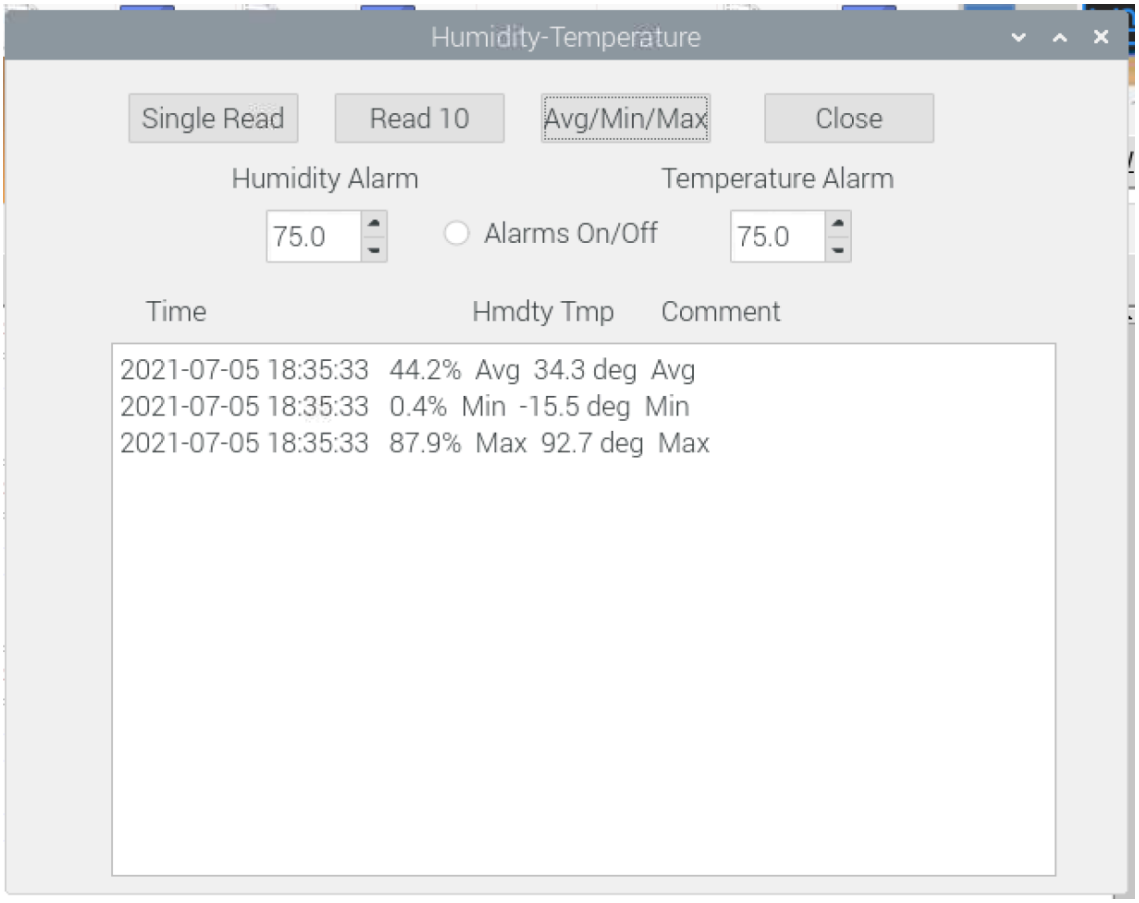
```
        storeStruct.append(displayStr)

        #Display the minimum humidity and temperature
        displayStr = time.strftime("%Y-%m-%d %H:%M:%S   ",tss) + str(round(hMin,1))
+ \
                "%  Min  " + str(round(tMin,1)) + " deg  Min"
        MyDialog.displayValues.append(displayStr)

        #store the event in the storage structure
        storeStruct.append(displayStr)

        #Display the maximum humidity and temperature
        displayStr = time.strftime("%Y-%m-%d %H:%M:%S   ",tss) + str(round(hMax,1))
+ \
                "%  Max  " + str(round(tMax,1)) + " deg  Max"
        MyDialog.displayValues.append(displayStr)

        #store the event in the storage structure
        storeStruct.append(displayStr)

#Close the application
    def closeAction(MyDialog):
        sys.exit(app.exec_())
```



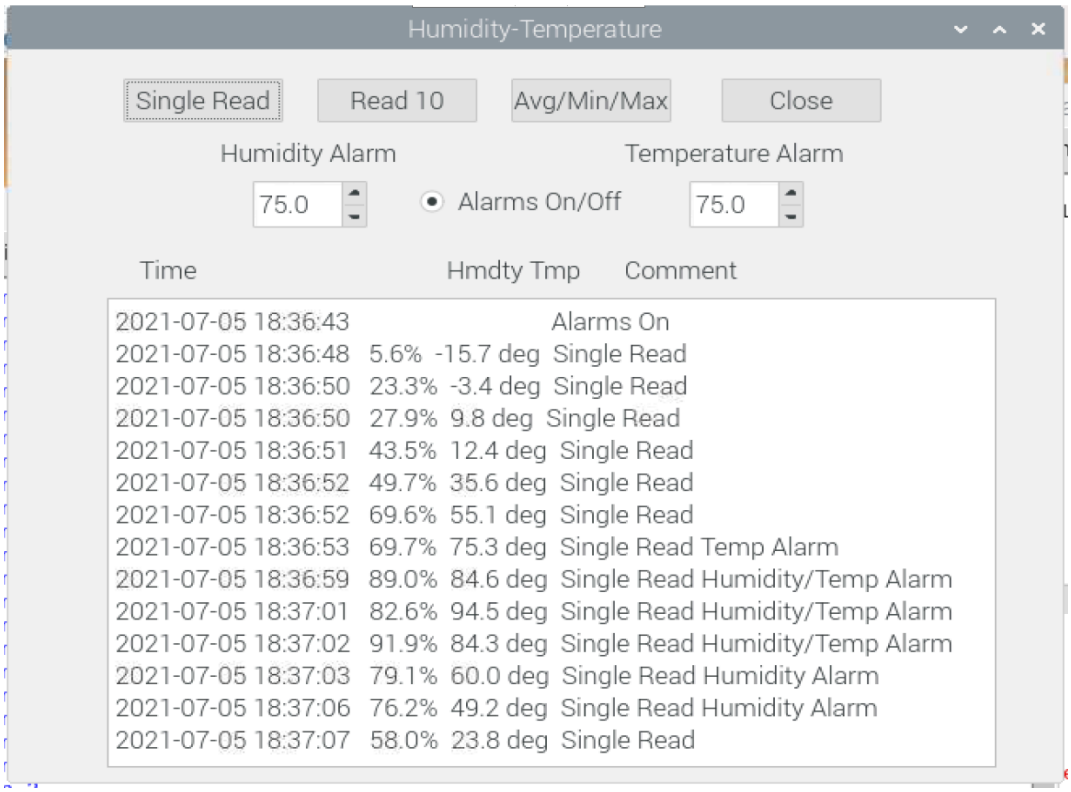
At Startup



A Single Read



Doing the Average



Showing the Alarms