Implementation Notes

- My main Python program start the Qt UI
- You can read single sensor data using the read button, the data is shown in the LCD panel.
- I used MySQL database to store incoming humidity/temperature value pairs along with a timestamp
- On the right side there is a Qt button that will read 10 values from the pseudo sensor with a one second delay between each reading.
- After the right table is field you can use the analyse button to calculate and display on the UI minimum, maximum, and average values for each reading type
- The two fields under the table are Alarm fields, if the coming data exceeded the value in there they will indicate that.
- The most right buttom Qt button close the UI and end the Python main program

humidity_reader.py

```
from PyQt5 import QtWidgets as qtw
from PyQt5 import QtCore as qtc
from PyQt5 import uic
from psuedoSensor import PseudoSensor
from database_f import Data_base
from datetime import datetime
from matplotlib import pyplot as plt


class Main_UI(qtw.QWidget):
def __init__(self):
super(Main_UI, self).__init__()
#Load the UI file
uic.loadUi('humidity_reader.ui', self)

#Storage vars
self.hum_list = []
self.temp_list = []
self.time_list = []

#Define UI widgets
self.read_btn = self.findChild(qtw.QPushButton, 'read_btn')
self.read_many = self.findChild(qtw.QPushButton, 'pushButton')
self.tmp_lcd = self.findChild(qtw.QLCDNumber, 'temp_out')
self.hum_lcd = self.findChild(qtw.QLCDNumber, 'hum_out')
self.table = self.findChild(qtw.QTableWidget, 'table')
self.table_2 = self.findChild(qtw.QTableWidget, 'table_2')
self.close_btn = self.findChild(qtw.QPushButton, 'close_btn')
self.analyse_btn = self.findChild(qtw.QPushButton, 'analyse_btn')
self.hum_alarm = self.findChild(qtw.QLineEdit, "hum_alarm")
self.temp_alarm = self.findChild(qtw.QLineEdit, "temp_alarm")
self.plot_btn = self.findChild(qtw.QPushButton, 'plot_btn')
# Initialize my sensor reader
self.ps = PseudoSensor()

#Initialize a timer
self.timer = qtc.QTimer()
self.timer.timeout.connect(self.read_multi_values)

#Action
self.read_btn.clicked.connect(self.read_values)
self.read_many.clicked.connect(self.start_reading)
self.close_btn.clicked.connect(self.close)
self.analyse_btn.clicked.connect(self.analyse_function)
self.plot_btn.clicked.connect(self.plot)
```

```
#Show
self.show()

def read_values(self):
hum, temp = self.ps.generate_values()
# Display the readings
self.tmp_lcd.display(temp)
self.hum_lcd.display(hum)
# Stor readings to the DB
my_DB = Data_base()
my_DB.store([hum, temp])
def read_multi_values(self):
# Add the readings to the table
hum, temp = self.ps.generate_values()
time = datetime.now().time().strftime('%H:%M:%S')
# Setting Alarms
if self.hum_alarm.text() != "":
h_limit = float(self.hum_alarm.text())
else:
h_limit = 60
if self.temp_alarm.text() != "":
t_limit = float(self.temp_alarm.text())
else:
t_limit = 80
if hum > h_limit:
self.hum_alarm.setText(str(hum))
self.hum_alarm.setStyleSheet("QLineEdit { background: yellow; color: red;}")
self.hum_alarm.setAlignment(qtc.Qt.AlignCenter)
if temp > t_limit:
self.temp_alarm.setText(str(temp))
self.temp_alarm.setStyleSheet("QLineEdit { background: yellow; color: red;}")
self.temp_alarm.setAlignment(qtc.Qt.AlignCenter)
self.time_list.append(time)
self.hum_list.append(hum)
self.temp_list.append(temp)
row = self.table.rowCount()
self.table.setRowCount(row+1)
self.table.setColumnCount(3)
self.table.setItem(row, 0, qtw.QTableWidgetItem(str(hum)))
self.table.setItem(row, 1, qtw.QTableWidgetItem(str(temp)))
self.table.setItem(row, 2, qtw.QTableWidgetItem(str(time)))
# Stor readings to the DB
my_DB = Data_base()
my_DB.store([hum, temp])
# Stop the timer for 10 readings
if self.table.rowCount() >= 10:
self.timer.stop()
```

```
self.plot_btn.setEnabled(True)
def start_reading(self):
while (self.table.rowCount() > 0):
self.table.removeRow(0)
self.hum_list = []
self.temp_list = []
self.timer.start(1000)
def analyse_function(self):
if len(self.hum_list ) != 0 and len(self.temp_list) !=0:
self.table_2.setItem(0, 0, qtw.QTableWidgetItem(str(max(self.hum_list))))
self.table_2.setItem(1, 0, qtw.QTableWidgetItem(str(max(self.temp_list))))
self.table_2.setItem(0, 1, qtw.QTableWidgetItem(str(min(self.hum_list))))
self.table_2.setItem(1, 1, qtw.QTableWidgetItem(str(min(self.temp_list))))
self.table_2.setItem(0, 2,
qtw.QTableWidgetItem(str(round(sum(self.hum_list)/len(self.hum_list), 4))))
self.table_2.setItem(1, 2,
qtw.QTableWidgetItem(str(round(sum(self.temp_list)/len(self.temp_list), 4))))
def plot(self):
time_axis = self.time_list
hum_axis = self.hum_list
temp_axis = self.temp_list
plt.plot(time_axis, hum_axis, 'g')
plt.plot(time_axis, temp_axis, 'r')
plt.legend(['Humidity','Temperature'])
plt.ylabel('Humidity and Temperature')
plt.xlabel('Time')
plt.title("Humidity & Temperature Graph")
plt.grid(True)
plt.tight_layout()
plt.show()


if __name__ == '__main__':
app = qtw.QApplication([])
ui = Main_UI()
app.exec_()
```

database_f.py:

```
import mysql.connector
from datetime import datetime

class Data_base():
def __init__(self):
self.db_name = 'sensor_readings'
self.tb_name = 'humidity_and_temperature'

self.mydb = mysql.connector.connect(
host= 'localhost',
user= 'zoro',
password= 'okey',
auth_plugin='mysql_native_password'
)
self.mycursor = self.mydb.cursor()
# Create a DB if not exitsts
self.mycursor.execute("CREATE DATABASE IF NOT EXISTS {}".format(self.db_name))
self.mycursor.execute("USE {}".format(self.db_name))
# Create a Table if not exitsts
self.mycursor.execute("CREATE TABLE IF NOT EXISTS {} (id int NOT NULL AUTO_INCREMENT,
Humidity double(255,4), Temperature double(255,4), Time varchar(100),PRIMARY
KEY(id))".format(self.tb_name))

def store(self, records):
values = ','.join("{}".format(e) for e in records)
time = datetime.now().time().strftime('%H:%M:%S')
values = values + ', "' +str(time) + '"'
insert_formula ="INSERT INTO {} (Humidity, Temperature, Time) VALUES
({})".format(self.tb_name,values)
self.mycursor.execute(insert_formula)
self.mydb.commit()

def show_columns(self):
self.mycursor.execute("show columns from {}".format(self.tb_name))
return [c for c in self.mycursor]
def fetch_column_values(self, col):
show_col = "select {} from {}".format(col, self.tb_name)
self.mycursor.execute(show_col)
return [t for v in self.mycursor for t in v]
```

humidity_reader.ui

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
<class>Form</class>
<widget class="QWidget" name="Form">
<property name="geometry">
<rect>
<x>0</x>
<y>0</y>
<width>854</width>
<height>573</height>
</rect>
</property>
<property name="windowTitle">
<string>Form</string>
</property>
<property name="styleSheet">
<string notr="true"/>
</property>
<widget class="QPushButton" name="close_btn">
<property name="geometry">
<rect>
<x>740</x>
<y>520</y>
<width>101</width>
<height>41</height>
</rect>
</property>
<property name="text">
<string>Close</string>
</property>
</widget>
<widget class="QTableWidget" name="table_2">
<property name="geometry">
<rect>
<x>20</x>
<y>200</y>
<width>401</width>
<height>101</height>
</rect>
</property>
<row>
<property name="text">
<string>Humidity</string>
</property>
</row>
```

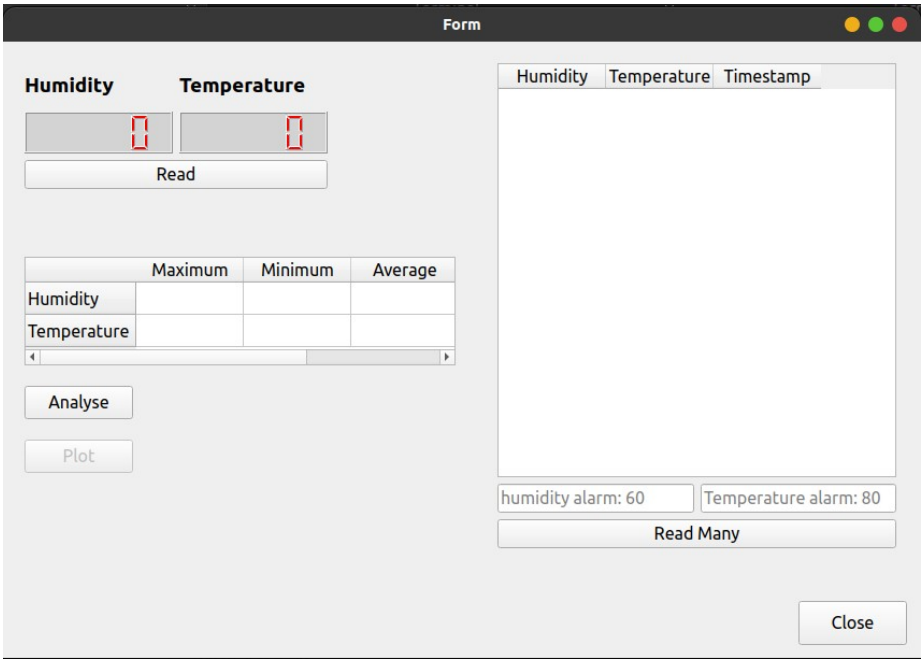
```
<row>
<property name="text">
<string>Temperature</string>
</property>
</row>
<column>
<property name="text">
<string>Maximum</string>
</property>
</column>
<column>
<property name="text">
<string>Minimum</string>
</property>
</column>
<column>
<property name="text">
<string>Average</string>
</property>
</column>
</widget>
<widget class="QPushButton" name="analyse_btn">
<property name="geometry">
<rect>
<x>20</x>
<y>320</y>
<width>101</width>
<height>31</height>
</rect>
</property>
<property name="text">
<string>Analyse</string>
</property>
</widget>
<widget class="QPushButton" name="plot_btn">
<property name="enabled">
<bool>false</bool>
</property>
<property name="geometry">
<rect>
<x>20</x>
<y>370</y>
<width>101</width>
<height>31</height>
</rect>
</property>
<property name="text">
```

```
<string>Plot</string>
</property>
</widget>
<widget class="QWidget" name="">
<property name="geometry">
<rect>
<x>20</x>
<y>20</y>
<width>282</width>
<height>117</height>
</rect>
</property>
<layout class="QGridLayout" name="gridLayout_2">
<item row="1" column="0">
<widget class="QPushButton" name="read_btn">
<property name="text">
<string>Read</string>
</property>
</widget>
</item>
<item row="0" column="0">
<layout class="QGridLayout" name="gridLayout">
<item row="0" column="0">
<widget class="QLabel" name="label">
<property name="font">
<font>
<pointsize>14</pointsize>
<weight>75</weight>
<bold>true</bold>
</font>
</property>
<property name="text">
<string>Humidity</string>
</property>
</widget>
</item>
<item row="1" column="1">
<widget class="QLCDNumber" name="temp_out">
<property name="styleSheet">
<string notr="true">color: red;
background-color: lightgrey;</string>
</property>
</widget>
</item>
<item row="0" column="1">
<widget class="QLabel" name="label_2">
<property name="font">
```

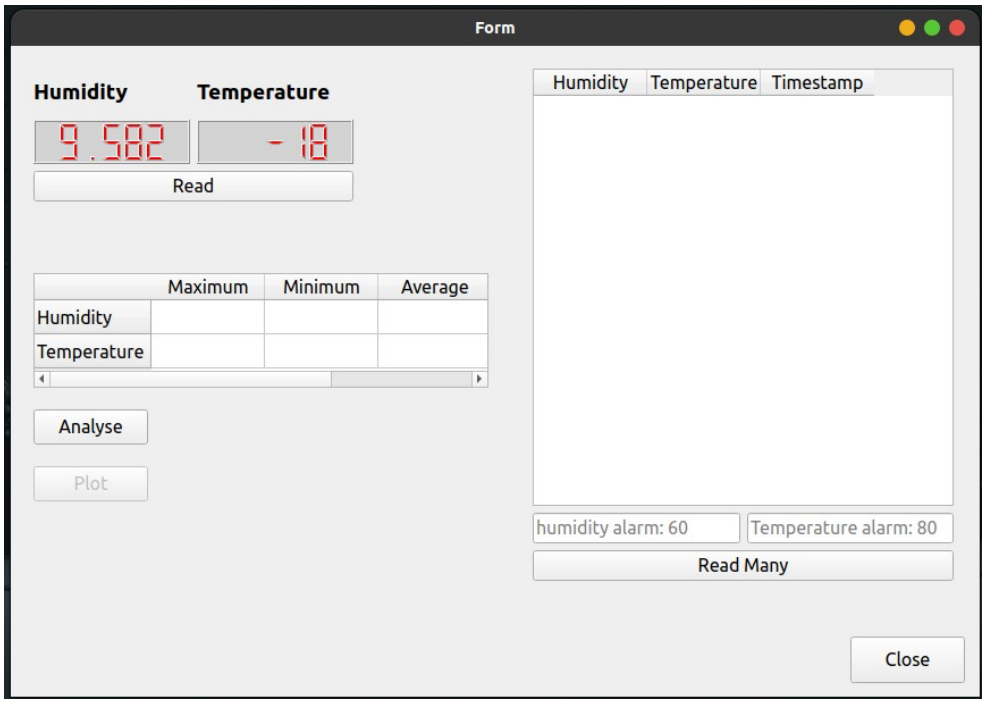
```
<font>
<pointsize>14</pointsize>
<weight>75</weight>
<bold>true</bold>
</font>
</property>
<property name="text">
<string>Temperature</string>
</property>
</widget>
</item>
<item row="1" column="0">
<widget class="QLCDNumber" name="hum_out">
<property name="styleSheet">
<string notr="true">color: red;
background-color: lightgrey;</string>
</property>
</widget>
</item>
</layout>
</item>
</layout>
</widget>
<widget class="QWidget" name="">
<property name="geometry">
<rect>
<x>460</x>
<y>20</y>
<width>371</width>
<height>451</height>
</rect>
</property>
<layout class="QGridLayout" name="gridLayout_3">
<item row="2" column="0">
<widget class="QPushButton" name="pushButton">
<property name="text">
<string>Read Many</string>
</property>
</widget>
</item>
<item row="1" column="0">
<layout class="QHBoxLayout" name="horizontalLayout">
<item>
<widget class="QLineEdit" name="hum_alarm">
<property name="text">
<string/>
</property>
```

```
<property name="placeholderText">
<string>humidity alarm: 60</string>
</property>
</widget>
</item>
<item>
<widget class="QLineEdit" name="temp_alarm">
<property name="placeholderText">
<string>Temperature alarm: 80</string>
</property>
</widget>
</item>
</layout>
</item>
<item row="0" column="0">
<widget class="QTableWidget" name="table">
<column>
<property name="text">
<string>Humidity</string>
</property>
</column>
<column>
<property name="text">
<string>Temperature</string>
</property>
</column>
<column>
<property name="text">
<string>Timestamp</string>
</property>
</column>
</widget>
</item>
</layout>
</widget>
</widget>
<resources/>
<connections/>
</ui>
```

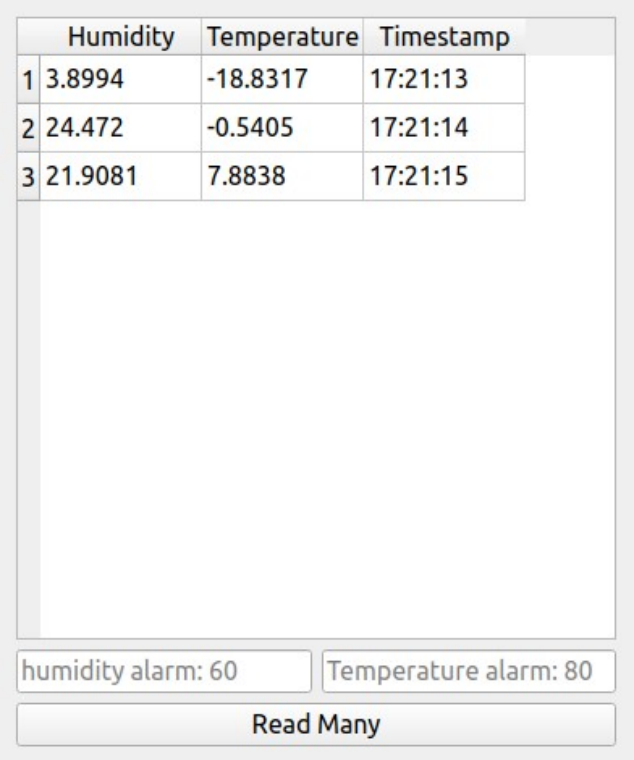
The Main Window :



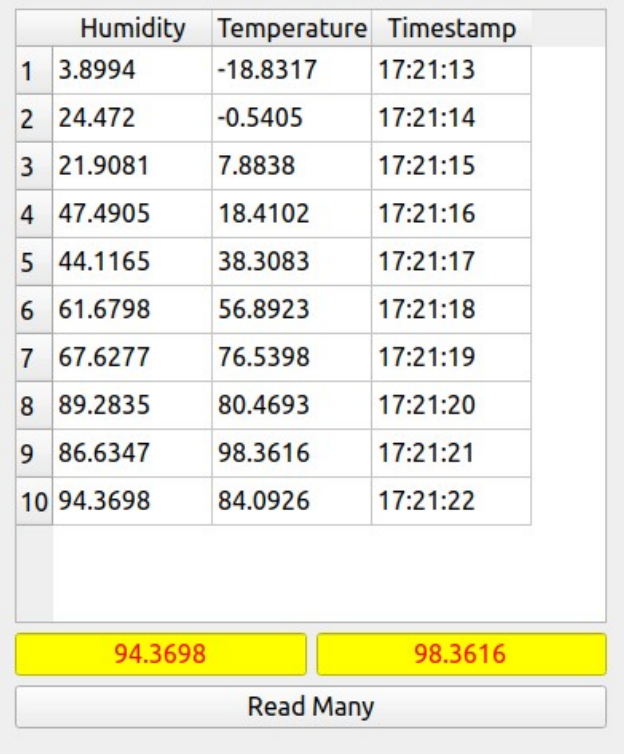
Reading Single Value:



Read Many Values: (one value each second)



The Alarm Detects Values That Exceeds The Limits:

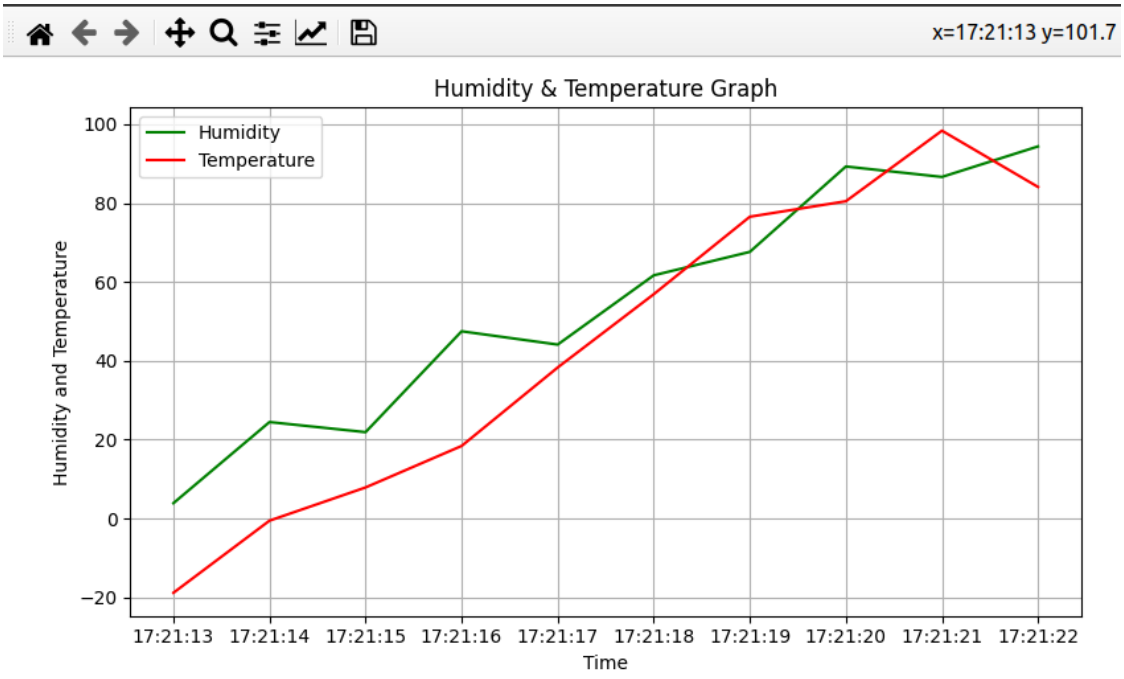


The Analyse Button:

	Maximum	Minimum	Average
Humidity	94.3698	3.8994	54.1482
Temperature	98.3616	-18.8317	44.1586

Analyse

Plot The Last 10 Values:



Store the sensor data are stored in MySQL Database: