Grant Novota

HTML GUI Development Project

Assumptions/Notes:

- Using Python3.8 on an Ubuntu machine. Viewed in firefox.
- 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 handling static files with tornado, so decided to not include graphs.

Required Libraries:

- sqlalchemy
- tornado
- numpy

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:

```python
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)
```

```python
    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
```

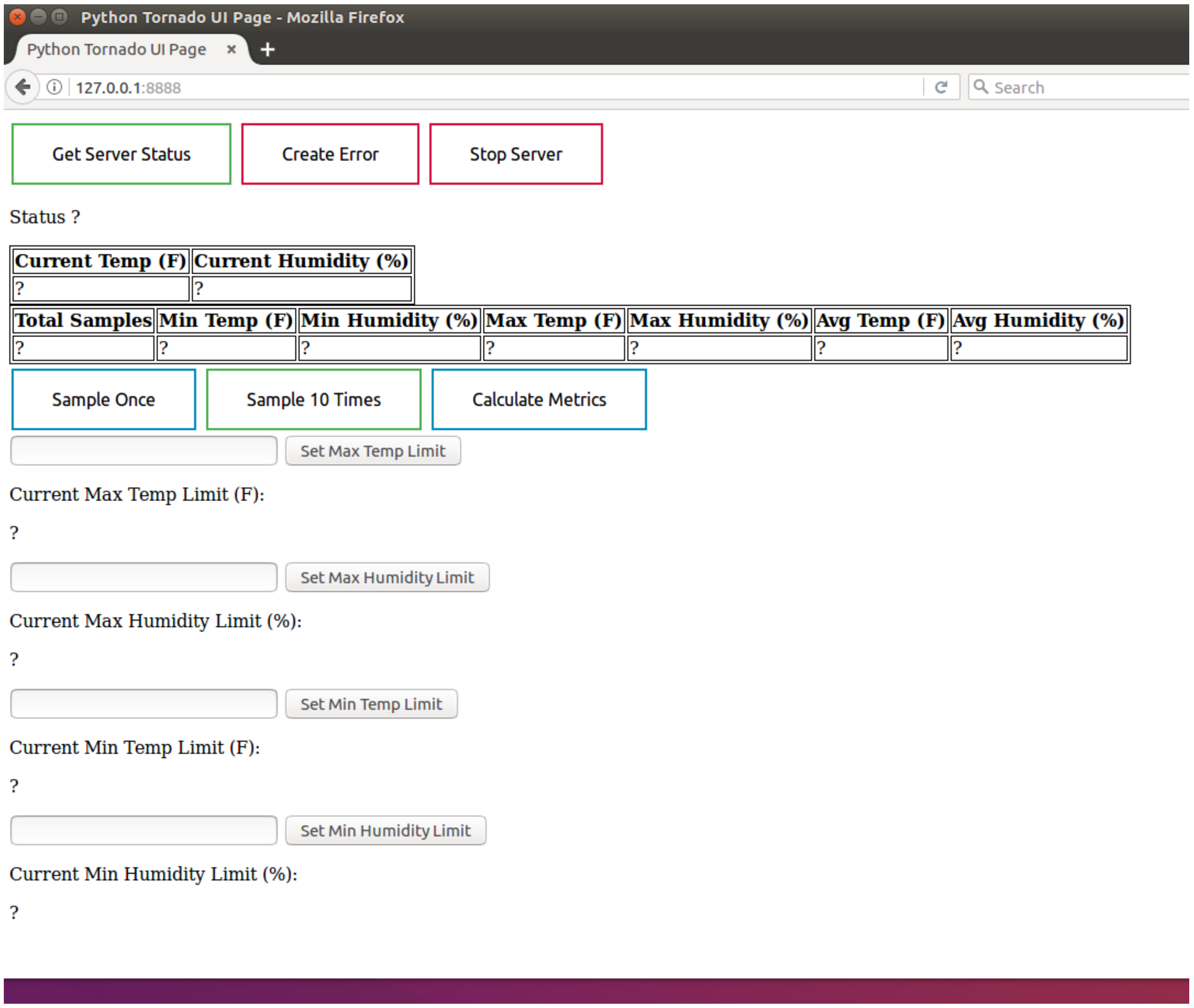
```python
# add rows to tables
def add_temp(session, value_f, value_c, time):
    temp = Temperature(value_f=value_f, value_c=value_c, time=time)
    session.add(temp)
    session.commit()
    return

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

def get_latest_temp(session):
    return session.query(Temperature).order_by(Temperature.id.desc()).first()

def get_latest_humidity(session):
    return session.query(Humidity).order_by(Humidity.id.desc()).first()
```

main.py:

```
# main code
import sys
import time
import numpy as np
from datetime import datetime
import tornado.ioloop
import tornado.web
import os
import json

# my libraries
import db
from psuedoSensor import PseudoSensor

# init database
session = db.init_session()

tornadoPort = 8888
cwd = os.getcwd() # used by static file server

current_temp = 0.0
current_humidity = 0.0
# alarm limits
temp_min_limit = 30.0
temp_max_limit = 80.0
humid_min_limit = 30.0
humid_max_limit = 70.0

# alarms
temp_min_alarm = False
temp_max_alarm = False
humid_min_alarm = False
humid_max_alarm = False

# allow cross-origin requests
class BaseHandler(tornado.web.RequestHandler):
    def set_default_headers(self):
        print("setting headers!!!")
        self.set_header("Access-Control-Allow-Origin", "*")
        self.set_header("Access-Control-Allow-Headers", "x-requested-with")
        self.set_header('Access-Control-Allow-Methods', 'GET, POST, PUT, DELETE, 
OPTIONS')
        # HEADERS!
        self.set_header("Access-Control-Allow-Headers", "access-control-allow-
origin,authorization,content-type")

    def options(self):
        # no body
        self.set_status(204)
        self.finish()

# send the index file
class IndexHandler(BaseHandler):
    def get(self, url = '/'):
        self.render('index.html')
    def post(self, url ='/'):
        self.render('index.html')

# handle commands sent from the web browser
class CommandHandler(BaseHandler):
    #both GET and POST requests have the same responses
    def get(self, url = '/'):
        print("get")
        self.handleRequest()

    def post(self, url = '/'):
        print("post")
        self.handleRequest()

    # handle both GET and POST requests with the same function
    def handleRequest(self):
        # is op to decide what kind of command is being sent
        op = self.get_argument('op', None)

        global temp_min_limit, temp_max_limit, humid_min_limit, humid_max_limit
        global temp_min_alarm, temp_max_alarm, humid_min_alarm, humid_max_alarm

        #received a "checkup" operation command from the browser:
        if op == "checkup":
            print("checkup called")
            #make a dictionary
            status = {"server": True }
            #turn it to JSON and send it to the browser
            self.write( json.dumps(status) )

        elif op == "sample once":
            print("sample once called")
            single_sample()
            #make a dictionary
            global current_temp, current_humidity
            status = {"server": True, "current_temp": current_temp,
"current_humidity": current_humidity,
                "temp_max_limit": temp_max_limit, "humid_max_limit":
humid_max_limit,
                "temp_min_limit": temp_min_limit, "humid_min_limit":
humid_min_limit,
                "temp_max_alarm": temp_max_alarm, "humid_max_alarm":
humid_max_alarm,
                "temp_min_alarm": temp_min_alarm, "humid_min_alarm":
humid_min_alarm}
            #turn it to JSON and send it to the browser
            self.write( json.dumps(status) )

        elif op == "sample multi":
            print("multi sample called")
            max = 10
            print("take 10 samples:")
            for i in range(max):
                print('sample', i+1)
                single_sample()
                time.sleep(1)
            global current_temp, current_humidity
            status = {"server": True, "current_temp": current_temp,
"current_humidity": current_humidity,
                "temp_max_limit": temp_max_limit, "humid_max_limit":
humid_max_limit,
                "temp_min_limit": temp_min_limit, "humid_min_limit":
humid_min_limit,
                "temp_max_alarm": temp_max_alarm, "humid_max_alarm":
humid_max_alarm,
                "temp_min_alarm": temp_min_alarm, "humid_min_alarm":
humid_min_alarm}
            #turn it to JSON and send it to the browser
            self.write( json.dumps(status) )

        elif op == "calc metrics":
            print("calc metrics called")
            metrics = calc_metrics()
            metrics["server"] = True
            status = metrics
            #turn it to JSON and send it to the browser
            self.write( json.dumps(status) )
        elif op == "set max temp":
            value = self.get_argument('value', None)
            temp_max_limit = float(value)
            print("max temp value:", value)

        elif op == "set max humidity":
            value = self.get_argument('value', None)
            humid_max_limit = float(value)
            print("max humidity value:", value)

        elif op == "set min temp":
            value = self.get_argument('value', None)
            temp_min_limit = float(value)
            print("min temp value:", value)

        elif op == "set min humidity":
            value = self.get_argument('value', None)
            humid_min_limit = float(value)
            print("min humidity value:", value)

        elif op == "create error":
            status = {}
            self.write( json.dumps(status) )

        elif op == "stop server":
            stop_tornado()


        #operation was not one of the ones that we know how to handle
        else:
            print(op)
            print(self.request)
            raise tornado.web.HTTPError(404, "Missing argument 'op' or not
recognized")

    def send_update(self):
        global current_temp, current_humidity
        status = {"current_temp": current_humidity, "current_humidity":
current_humidity }
        self.write( json.dumps(status) )

# adds event handlers for commands and file requests
application = tornado.web.Application([
    #all commands are sent to http://*:port/com
    #each command is differentiated by the "op" (operation) JSON parameter
    (r"/(com.*)", CommandHandler ),
    (r"/", IndexHandler),
    (r"/(index\.html)", tornado.web.StaticFileHandler,{"path": cwd}),
    (r"/(.*\.png)", tornado.web.StaticFileHandler,{"path": cwd }),
    (r"/(.*\.jpg)", tornado.web.StaticFileHandler,{"path": cwd }),
    (r"/(.*\.js)", tornado.web.StaticFileHandler,{"path": cwd }),
    (r"/(.*\.css)", tornado.web.StaticFileHandler,{"path": cwd }),
])

# END OF WEB APP FUNCTIONS


# get sample of data from pseudo sensor
def sample_data():
    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
    global temp_min_limit, temp_max_limit, humid_min_limit, humid_max_limit
    global temp_min_alarm, temp_max_alarm, humid_min_alarm, humid_max_alarm

    # reset alarms
    temp_min_alarm = False
    temp_max_alarm = False
    humid_min_alarm = False
    humid_max_alarm = False

    # set alarms
    if temp_f > temp_max_limit:
        temp_max_alarm = True
    elif temp_f < temp_min_limit:
        temp_min_alarm = True
    elif h > humid_max_limit:
        humid_max_alarm = True
    elif h < humid_min_limit:
        humid_min_alarm = True

    return h, temp_f

def single_sample():
    h,t = sample_data()
    global current_temp, current_humidity
    current_temp = t
    current_humidity = h
    print('sample', 'temp:', t, 'humidity:', h)
    return

def calc_metrics():
        temp_list, temp_times = db.get_all_temps(session, "f")
        humid_list, humid_times = db.get_all_humids(session)
        metrics = {}
        # set total samples
        metrics["total_samples"] = str(len(temp_list))
        # min temp
        metrics["min_temp"] = str(min(temp_list))
        # min humidity
        metrics["min_humidity"] = str(min(humid_list))
        # max temp
        metrics["max_temp"] = str(max(temp_list))
        # max humidity
        metrics["max_humidity"] = str(max(humid_list))
        # avg temp
        metrics["avg_temp"] = str(sum(temp_list)/len(temp_list))
        # avg humidity
        metrics["avg_humidity"] = str(sum(humid_list)/len(humid_list))
        return metrics

def start_tornado():
    application.listen(tornadoPort)
    tornado.ioloop.IOLoop.instance().start()

def stop_tornado():
    tornado.ioloop.IOLoop.instance().stop()

if __name__ == "__main__":
    #start tornado
    print("Starting server on port number %i..." % tornadoPort )
    print("Open at http://127.0.0.1:%i/index.html" % tornadoPort )
    start_tornado()
```

Screenshots:

1. The HTML UI at start up



Status ?

| Current Temp (F) | Current Humidity (%) |
|---|---|
| ? | ? |

| Total Samples | Min Temp (F) | Min Humidity (%) | Max Temp (F) | Max Humidity (%) | Avg Temp (F) | Avg Humidity (%) |
|---|---|---|---|---|---|---|
| ? | ? | ? | ? | ? | ? | ? |

Current Max Temp Limit (F):

?

Current Max Humidity Limit (%):

?

Current Min Temp Limit (F):

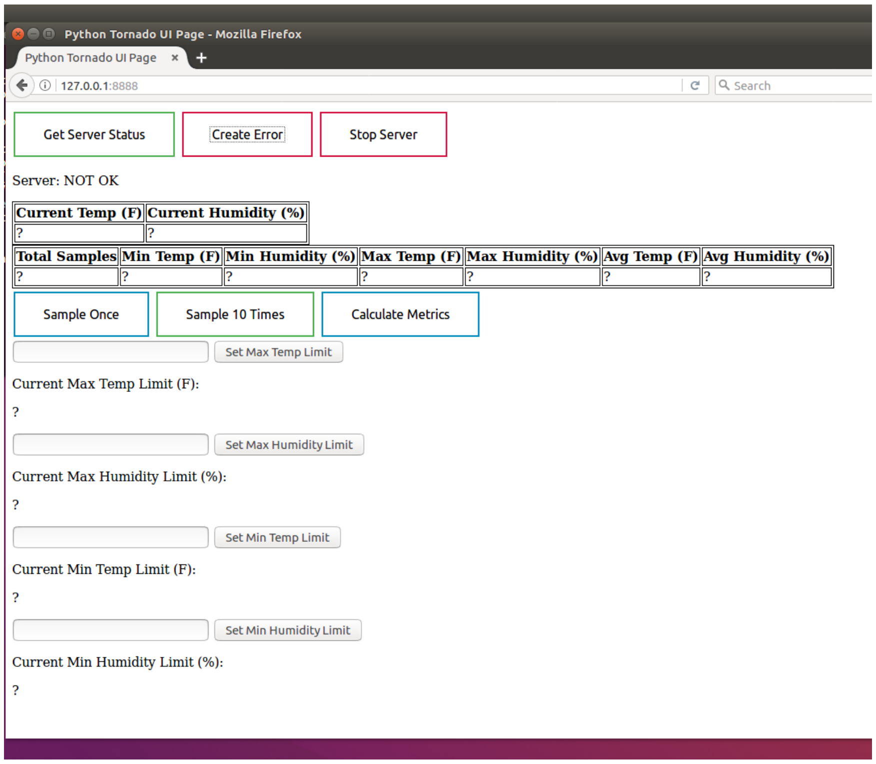
?

Current Min Humidity Limit (%):

?

2. Error conditions

Error in code:



Python Tornado UI Page - Mozilla Firefox

127.0.0.1:8888

Server: NOT OK

| Current Temp (F) | Current Humidity (%) |
|---|---|
| ? | ? |

| Total Samples | Min Temp (F) | Min Humidity (%) | Max Temp (F) | Max Humidity (%) | Avg Temp (F) | Avg Humidity (%) |
|---|---|---|---|---|---|---|
| ? | ? | ? | ? | ? | ? | ? |

Current Max Temp Limit (F):

?

Current Max Humidity Limit (%):

?

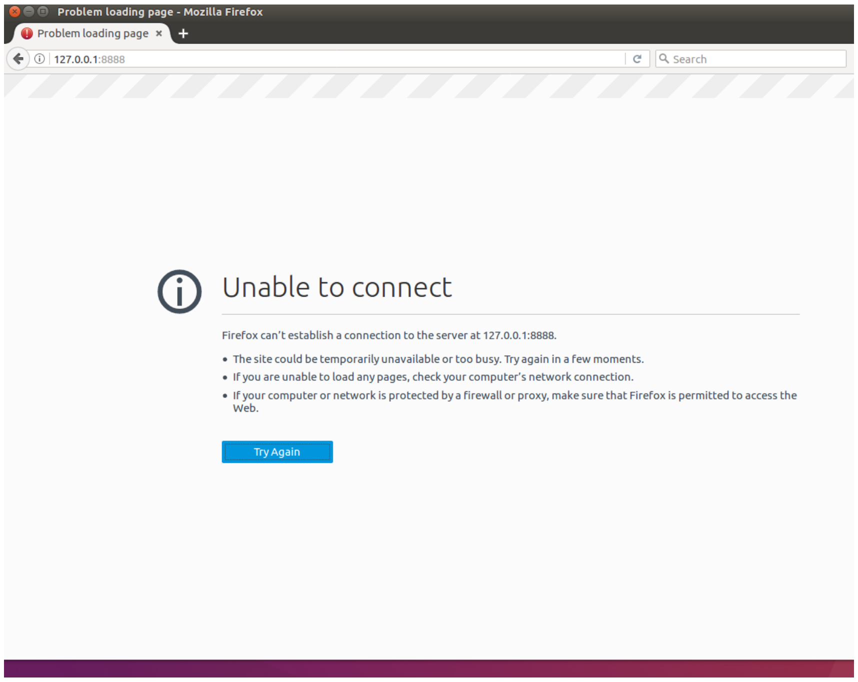
Current Min Temp Limit (F):

?

Current Min Humidity Limit (%):

?

Disconnected from server:



Problem loading page - Mozilla Firefox

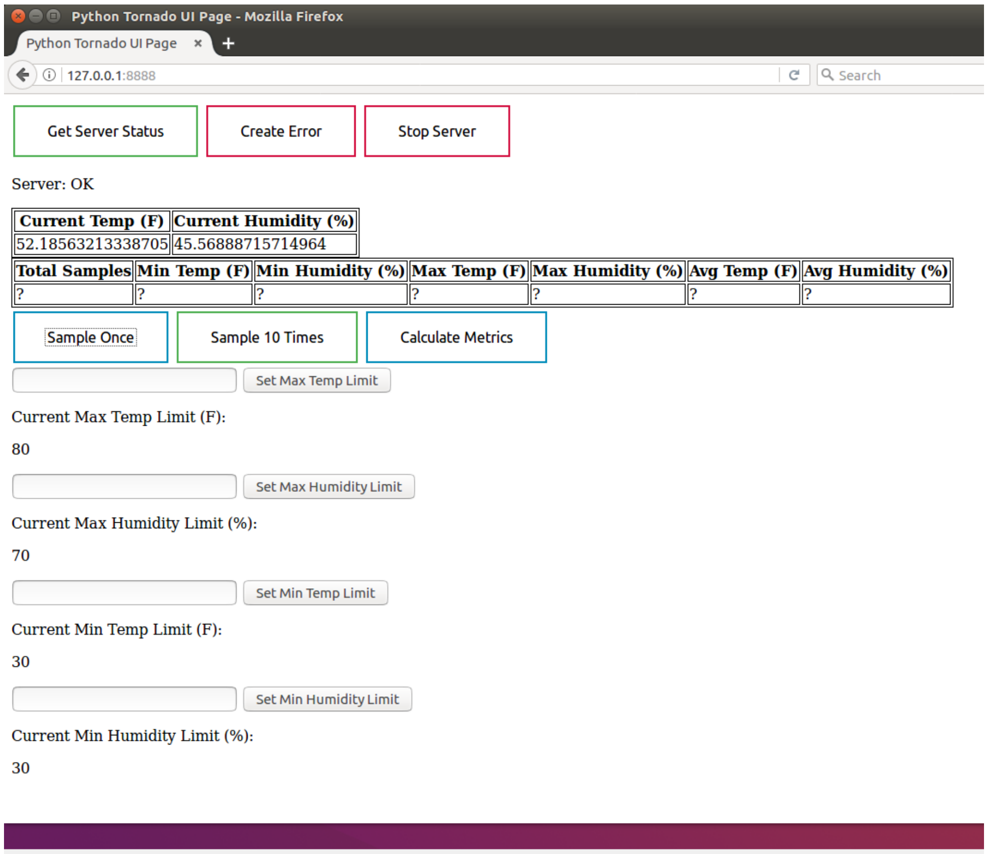
127.0.0.1:8888

Unable to connect

Firefox can't establish a connection to the server at 127.0.0.1:8888.

- The site could be temporarily unavailable or too busy. Try again in a few moments.
- If you are unable to load any pages, check your computer's network connection.
- If your computer or network is protected by a firewall or proxy, make sure that Firefox is permitted to access the Web.

3. The UI after its first single data point reading



Server: OK

| Current Temp (F) | Current Humidity (%) |
|---|---|
| 52.18563213338705 | 45.56888715714964 |

| Total Samples | Min Temp (F) | Min Humidity (%) | Max Temp (F) | Max Humidity (%) | Avg Temp (F) | Avg Humidity (%) |
|---|---|---|---|---|---|---|
| ? | ? | ? | ? | ? | ? | ? |

Current Max Temp Limit (F):

80

Current Max Humidity Limit (%):

70

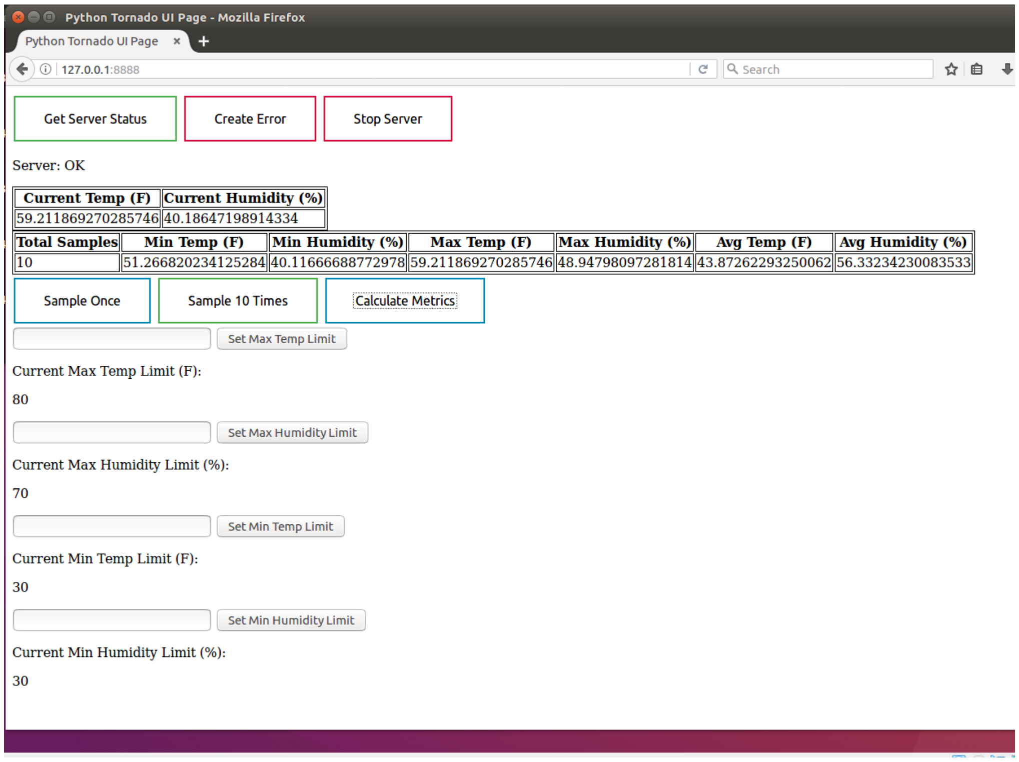
Current Min Temp Limit (F):

30

Current Min Humidity Limit (%):

30

4. The UI after it has calculated a 10 point average



Server: OK

| Current Temp (F) | Current Humidity (%) |
|---|---|
| 59.211869270285746 | 40.18647198914334 |

| Total Samples | Min Temp (F) | Min Humidity (%) | Max Temp (F) | Max Humidity (%) | Avg Temp (F) | Avg Humidity (%) |
|---|---|---|---|---|---|---|
| 10 | 51.266820234125284 | 40.11666688772978 | 59.211869270285746 | 48.94798097281814 | 43.87262293250062 | 56.33234230083533 |

Current Max Temp Limit (F):

80

Current Max Humidity Limit (%):

70

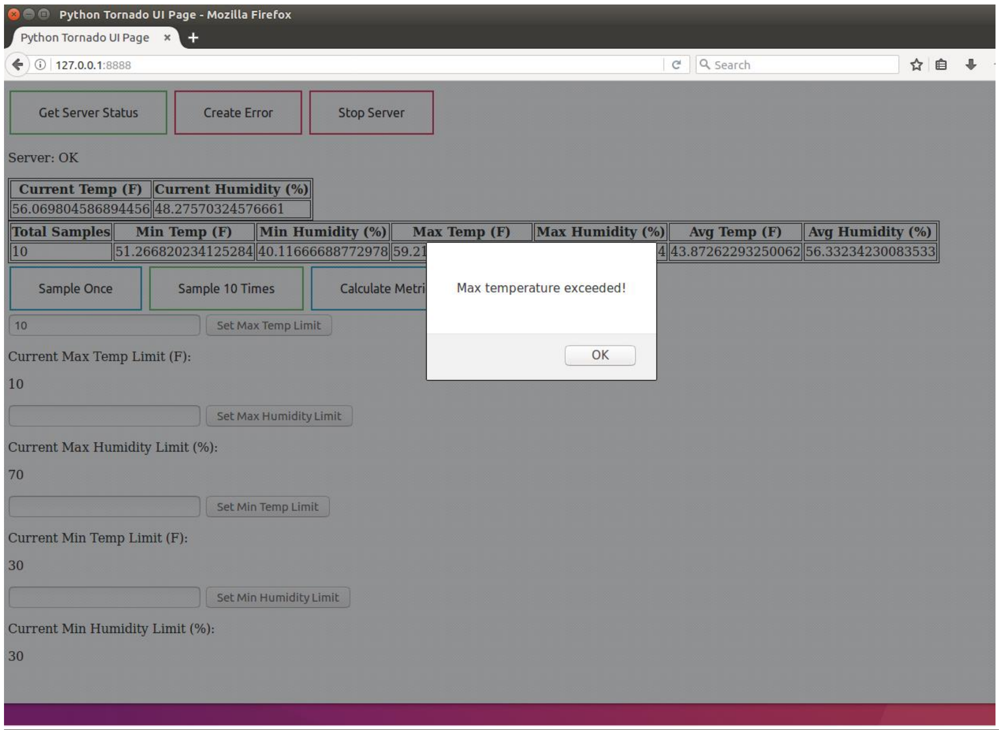
Current Min Temp Limit (F):

30

Current Min Humidity Limit (%):

30

5. The UI after it has seen either a temperature or humidity alarm



Server: OK

| Current Temp (F) | Current Humidity (%) |
|---|---|
| 56.069804586894456 | 48.27570324576661 |

| Total Samples | Min Temp (F) | Min Humidity (%) | Max Temp (F) | Max Humidity (%) | Avg Temp (F) | Avg Humidity (%) |
|---|---|---|---|---|---|---|
| 10 | 51.266820234125284 | 40.11666688772978 | 59.21 | 4 | 43.87262293250062 | 56.33234230083533 |

Max temperature exceeded!

Current Max Temp Limit (F):

10

Current Max Humidity Limit (%):

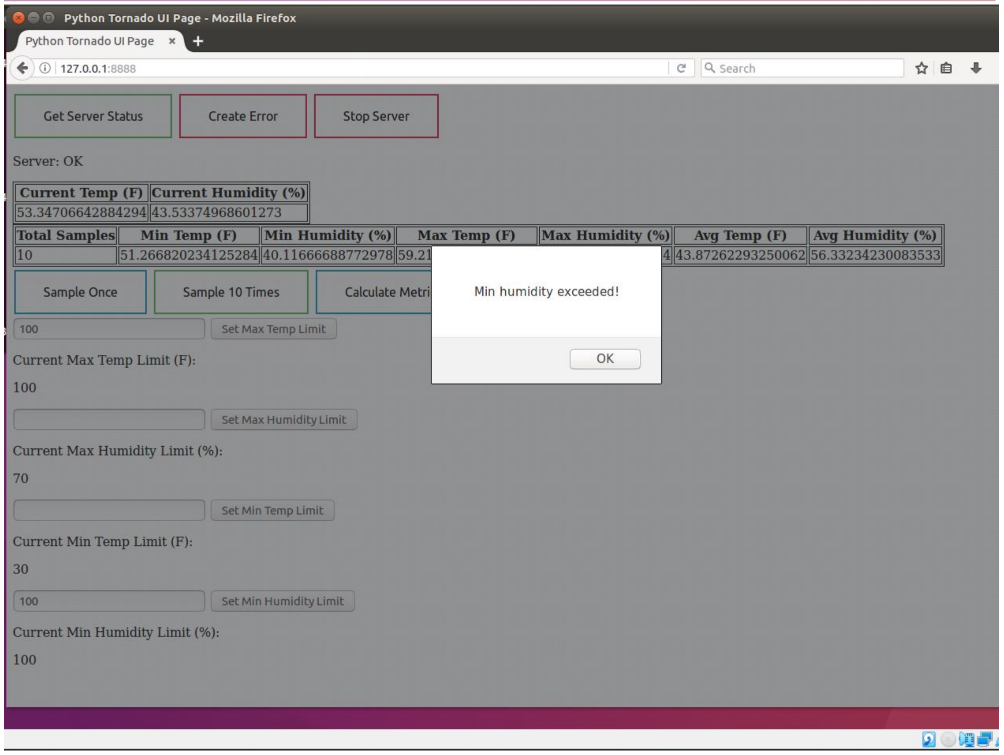
70

Current Min Temp Limit (F):

30

Current Min Humidity Limit (%):

30



Server: OK

| Current Temp (F) | Current Humidity (%) |
|---|---|
| 53.34706642884294 | 43.53374968601273 |

| Total Samples | Min Temp (F) | Min Humidity (%) | Max Temp (F) | Max Humidity (%) | Avg Temp (F) | Avg Humidity (%) |
|---|---|---|---|---|---|---|
| 10 | 51.266820234125284 | 40.11666688772978 | 59.21 | 4 | 43.87262293250062 | 56.33234230083533 |

Min humidity exceeded!

Current Max Temp Limit (F):

100

Current Max Humidity Limit (%):

70

Current Min Temp Limit (F):

30

Current Min Humidity Limit (%):

100