Implementation Notes

- The “server.py” file is the python code that will run the local tornado server.
- The UI will load on the “index.html” web page, in the top there is a connect link that will connect the UI with the server.
- If the connection is successfully made the background color of the top bar will turn to green color, if the connection lost or failed the page will alert that and the background color of the connect bar will change to red.
- you can read single temperature and humidity value from the left “read button”, and with the fill button on the right you will get a 10 reading for every one second.
- In the bottom there is an alarm section, by default the alarm is set to “60%” for the humidity and “50 °C” for the temperature.
- Whenever a new value in the table exceeded the alarm value the alarm will be turned on and the background color of that input field will change to the orange.

The Code

Index.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<link rel="stylesheet" href="style.css">
<title>Temperature Dashboard</title>
<script src="https://code.jquery.com/jquery-3.6.0.min.js"
integrity="sha256-/xUj+3OJU5yExlq6GSYGSHk7tPXikynS7ogEvDej/m4="
crossorigin="anonymous"></script>
<script>
$(document).ready(function () {
var ws;
$("#hum_alarm").val(60);
$("#temp_alarm").val(50);
$("#connect_link").click(function(evt) {
evt.preventDefault();
$("#terminal").text("connecting");

ws = new WebSocket("ws://localhost:8888/ws");
// Handle incoming websocket message callback
ws.onmessage = function(evt) {
const parts = evt.data.split(":");
// console.log(parts);
if (parts[0] == "single") {
$("#hum_out").text(parts[1]);
$("#temp_out").text(parts[2]);
}
else if (parts[0] == "multiple") {
var temp_limit = parseFloat($("#temp_alarm").val());
var hum_limit = parseFloat($("#hum_alarm").val());
var hum = parseFloat(parts[1]);
var temp = parseFloat(parts[2]);
if (hum > hum_limit) {
$("#hum_alarm").css("background", "#f7b305");
$("#hum_alarm").val(hum);
};
if (temp > temp_limit) {
$("#temp_alarm").css("background", "#f7b305");
$("#temp_alarm").val(temp);
};
var table = document.getElementById("tbb");
```

```
var row = table.insertRow();
var hum_cell = row.insertCell(0);
var temp_cell = row.insertCell(1);
var time_cell = row.insertCell(2);
temp_cell.innerHTML = parts[1] + ' %';
hum_cell.innerHTML = parts[2] + ' °C';
time_cell.innerHTML = parts[3];
}
};
// Open websocket
ws.onopen = function(evt) {
$("#connect_bar").css("background", "#00ff00");
$("div#message_details").show();
$("#terminal").text("***Connection Opened***");
};

// Close Websocket callback
ws.onclose = function(evt) {
$("#terminal").text("***Connection Closed***");
alert("Connection close");
$("#connect_bar").css("background", "#ff0000");
$("div#message_details").empty();

};

// Read single value
$("#read_s").click(function(evt) {
ws.send("read single value");
});

// Read multiple values
$("#read_m").click(function(evt) {
ws.send("read multiple values");
});

// Close the UI and the tornado server
$("#close").click(function(evt) {
console.log("close tornado server");
ws.send("close");
});
});
$("#close").click(function(evt) {
// ws.send("read multiple values");
console.log("close the web page");
});

});
```

```
</script>
</head>
<body>
<h1 class="main_title">Humidity and Temperature Dashboard</h1>
<div id="connect_bar">
<a href="#" id="connect_link"> Connect to the server </a>
<div id="terminal">
</div>
<div id="status">
</div>
</div>
<div id="upper_cont">
<div id="left_cont">
<div id="single_value_container">
<div class="h_box_cont">
<div class="h_box">
<div>Temperature</div>
<div class="value_box" id="temp_out">0.00</div>
</div>
<div class="h_box">
<div>Humidity</div>
<div class="value_box" id="hum_out">0.00</div>
</div>
</div>
<div>
<input type="button" class="btn" id="read_s" value="read">
</div>
</div>
<div id="alamrs">
<h4>Alamrs</h4>
<label>Temperature</label>
<input type="text" name="Temperature" class="inp_field" id="temp_alarm" value="50">
<label>Humidity</label>
<input type="text" name="Humidity" class="inp_field" id="hum_alarm" value="60">
</div>
</div>
<!-- ========================== second section
========================== -->
<div id="tb_container">
<div id="table">
<table id="tbb">
<tr>
<th>Temperature</th>
<th>Humidity</th>
<th>Timestamp</th>
</tr>
</table>
```

```
</div>
<div>
<input type="button" value="Fill" id="read_m" class="btn">
</div>
</div>
</div>
<input type="button" value="Close" class="btn" id="close">
</body>
</html>
```

style.css:

```
body{
background-color: rgb(245, 245, 245);
}

.main_title {
margin: 10px 30px;
}

#connect_bar {
margin: 20px;
padding: 10px 20px;
background-color: rgba(181, 201, 219, 0.63);
text-align: center;
font-size: 18px;
}

#upper_cont {
width: 100%;
margin: auto;
/* border: 1px yellow solid; */
display: flex;
flex-direction: row;
justify-content: space-around;
align-items: flex-start;
}

#left_cont {
margin-top: 50px;
display: flex;
flex-direction: column;
justify-content: space-between;
align-items: center;
}

#single_value_container {
```

```
margin-top: 20px;
/* border: 1px black dotted; */
display: flex;
flex-direction: column;
/* justify-content: center; */
align-items: center;
}

.h_box_cont {
/* border: 1px red dotted; */
width: 40%;
display: flex;
flex-direction: row;
justify-content: space-around;
text-align: center;
}

.value_box {
border: 2px rgb(24, 24, 56) solid;
margin: 20px;
padding: 10px 20px;
text-align: center;
border-radius: 4px;
}

.btn {
padding: 10px 20px;
margin: 20px;
border-radius: 10px;
/* border: 0.5px gray solid; */
}

#alamrs {
/* border: 1px red dotted; */
margin: 40px 10px;
margin-top: 60px;
}

.inp_field {
width: 60px;
height: 20px;
border: 2px rgb(24, 24, 56) solid;
border-radius: 4px;
}

/* ================================================== */
```

```
#table {
margin-left: auto;
margin-right: auto;
font-family: Arial, Helvetica, sans-serif;
border-collapse: collapse;
width: 100%;
}
#table td, #table th {
border: 1px solid rgb(24, 24, 56);
padding: 8px;
}
#table tr:nth-child(even){background-color: #f2f2f2;}
#table tr:hover {background-color: #ddd;}
#table th {
text-align: left;
background-color: #19525c;
color: white;
}
```

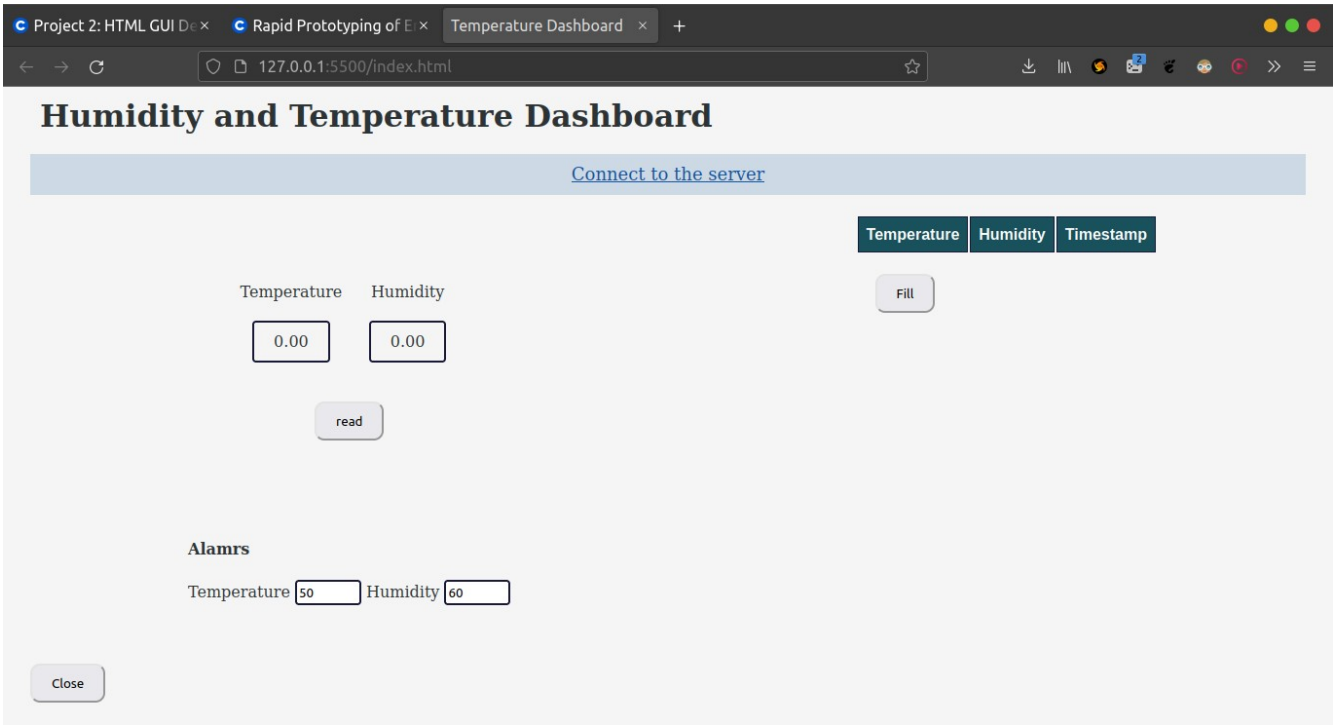
server.py:

```
import tornado.httpserver
import tornado.websocket
import tornado.ioloop
import tornado.web
import socket
from psuedoSensor import PseudoSensor as ps
from time import sleep
from datetime import datetime
class WSHandler(tornado.websocket.WebSocketHandler):
def open(self):
print('new connection')
self.ps = ps()
self.stop_index = False
def on_message(self, message):
print('request received')
if message == "read single value":
id = "single"
hum, temp = self.ps.generate_values()
self.write_message(id + ':' + str(hum) + ":" + str(temp))
elif message == "read multiple values":
for i in range(0, 10):
id = "multiple"
hum, temp = self.ps.generate_values()
timestampe = str(datetime.now().time().strftime('%H-%M-%S'))
msg = id + ':' + str(hum) + ":" + str(temp) + ":" + timestampe
```

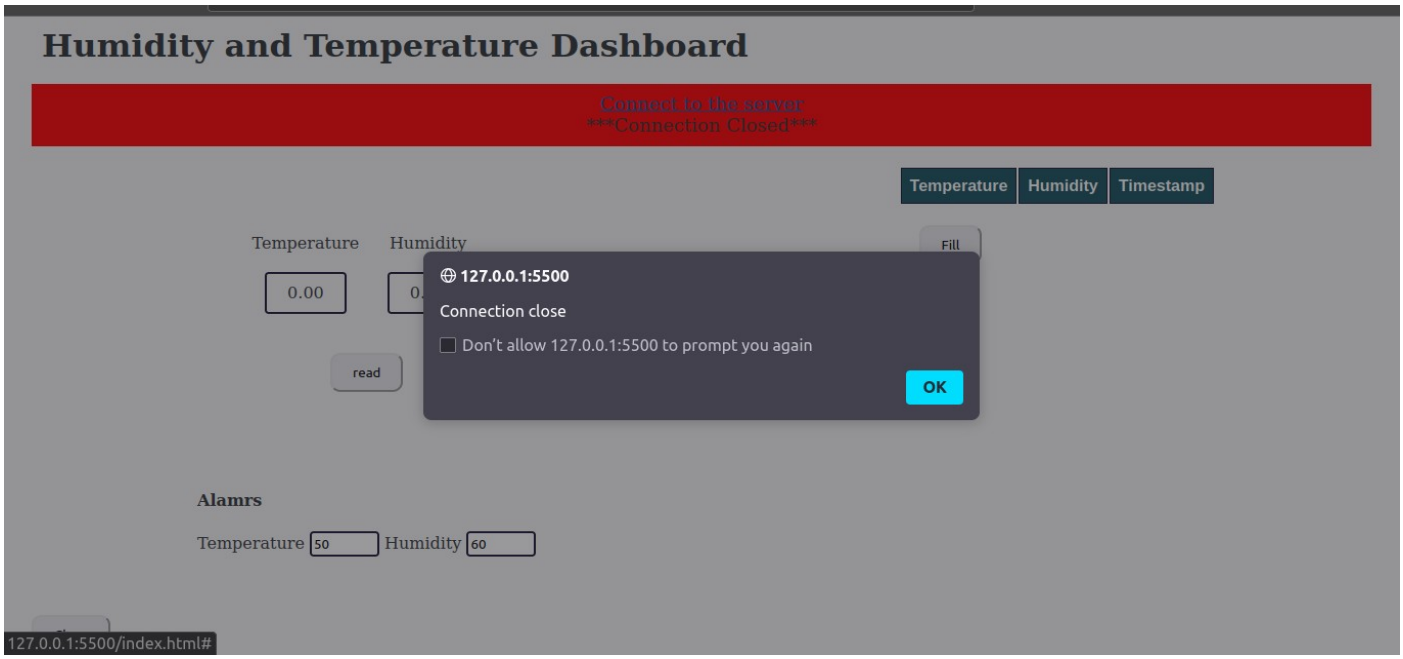
```
self.write_message(msg)
sleep(1)
if message == "close":
self.write_message("Closeing the server")
tornado.ioloop.IOLoop.instance().stop()
def on_close(self):
print ('connection closed')

def stop_it(self):
tornado.ioloop.stop()
def check_origin(self, origin):
return True
application = tornado.web.Application([
(r'/ws', WSHandler),
])
if __name__ == "__main__":
http_server = tornado.httpserver.HTTPServer(application)
http_server.listen(8888)
myIP = socket.gethostbyname(socket.gethostname())
print ('*** Websocket Server Started at %s***' % myIP)
tornado.ioloop.IOLoop.instance().start()
```

Screen capture of the project:

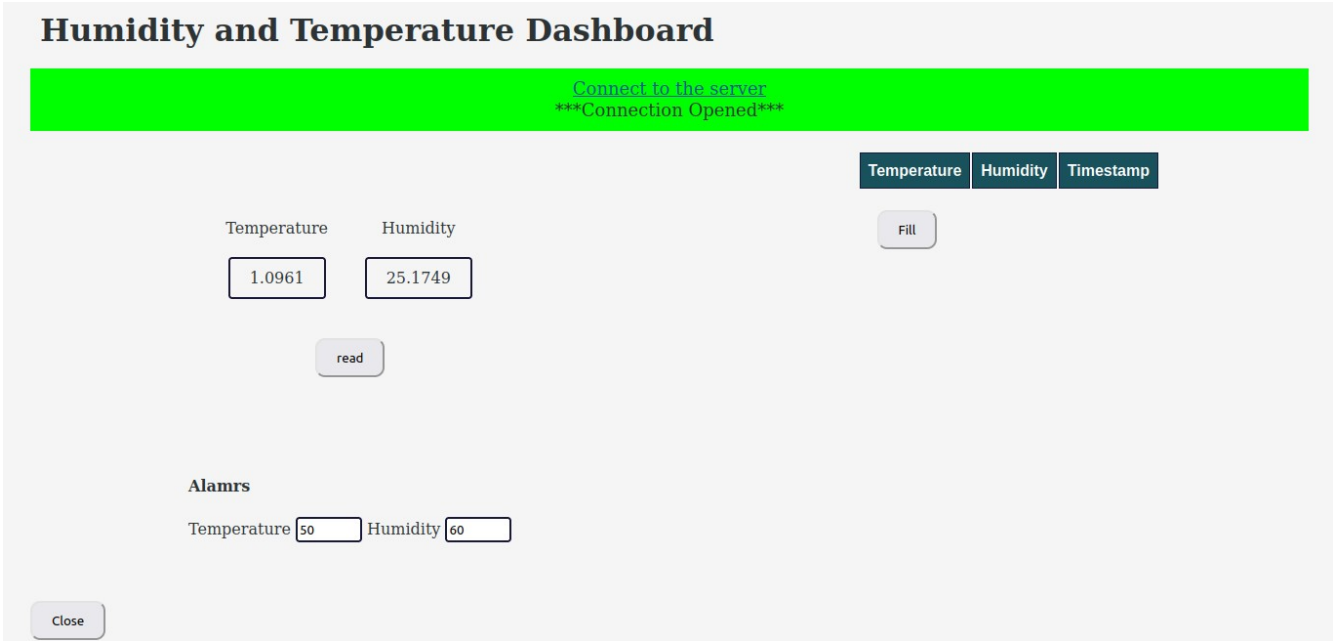
The HTML UI at startup :



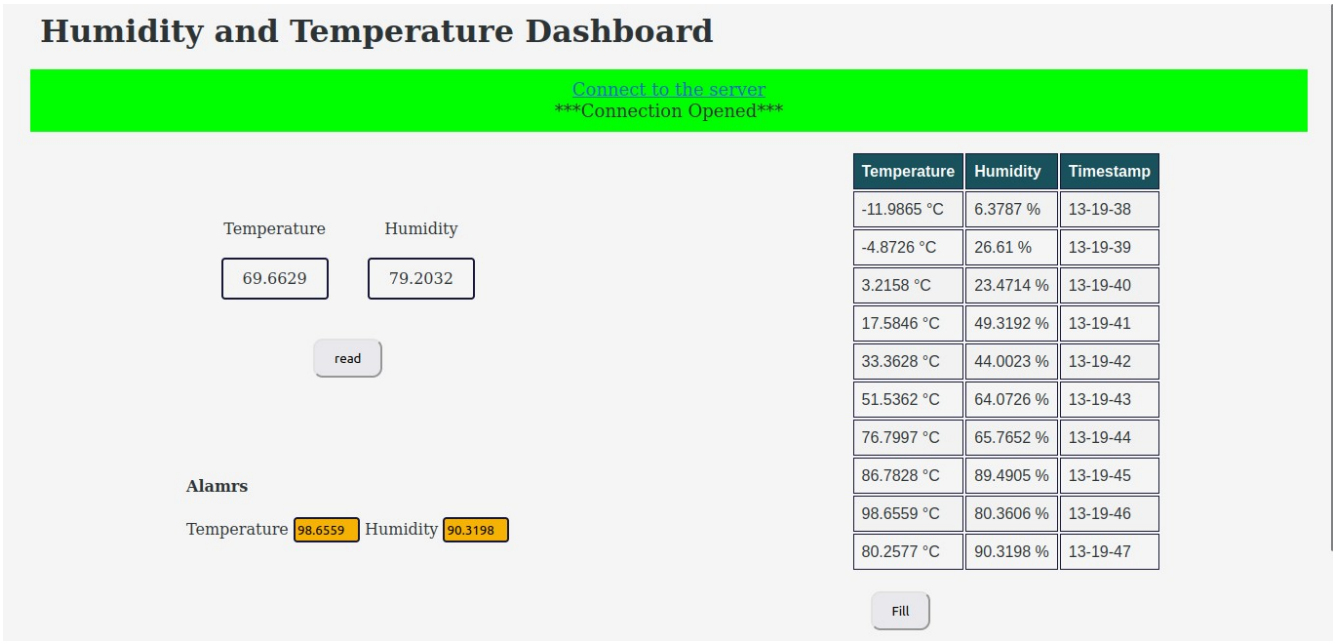
An error condition – loss of connection to the webserver or to the python code:



The UI after its first single data point reading



The UI after it has calculated a 10 point average:



The UI after it has seen either a temperature or humidity alarm (or both):