HTML GUI for Temperature & Humidity Sensor

ECEA5347: Rapid Prototyping of Embedded Interface Designs

Glenn Frey Olamit

November 5, 2021

1. Implementation & Assumption Notes

I used Javascript, Html, Jquery and Css to implement the front-end of the project as recommended in this course. I used Python and it's associated library like Tornado for web server, QtSql API to access database, JSON, re for regular expression, etc for the backend implementation.

I used Raspberry Pi model 3b+ for this project and installed the latest Raspbian OS for this embedded project as it fit the requirement. I installed Geany, a lightweight GUI text editor as resources must be conserved and MobaXterm to work remotely.

I used Sqlite3 for the following benefit: 1. SQLite has a relatively low overhead. 2. It is a self-contained system. No external dependencies are required to make it function. 3. No separate server process. SQLite won't chew up your Raspberry Pi's RAM and CPU when not being utilized. 4.Zero configuration is needed making it easy to use right out of the box.

After the user login and the connection is open, the user will be redirected to the main page with a uri address http:localhost:8888/sensor.

I assume the values extracted from the psudosensor in Celsius and the Values I choose to display are in Celsius.

I assume in project requirement number six the temperature and humidity reading is not recorded to the database. I used one button to read both data from the sensors and display the UI.

In project requirement seven I used a table to display the ten values being recorded. A text to display the number of recordings and a progress bar to indicate the status and completion.

In project requirement seven I change the colour of the text display for alarm to red and display alarm text warning to indicate the alarm status of temperature and humidity.

I used two html pages for this project. The websocket.html for the login page that will indicate if the user successfully login to the website using websocket and then be redirected to the main page which is the index.html.

The css and javascript are not separated to its associated html. Since the code is relatively short and for simplicity.

2. Code

2.1 server.py

Breakline

```
import tornado.httpserver
import tornado.websocket
import tornado.ioloop
import tornado.web
import socket
import json
import sys
import re
import sqlite3
from PyQt5.QtSql import QSqlDatabase, QSqlQueryModel, QSqlQuery, QSqlTableModel
from os.path import exists
from psuedoSensor import PseudoSensor
'''
This is a simple Websocket Echo server that uses the Tornado websocket handler.
Please run `pip install tornado` with python of version 2.7.9 or greater to install tornado.
This program will echo back the reverse of whatever it recieves.
Messages are output to the terminal for debuggin purposes.
'''



connection = sqlite3.connect("project.db")
cursor = connection.cursor()
if not exists("project.db"):
    print("File projects.db does not exist.")
    sys.exit()
db = QSqlDatabase.addDatabase("QSQLITE")
db.setDatabaseName("project.db")
db.open()

ps = PseudoSensor()
humidity,temperature = ps.generate_values()

class readDataRequestHandler(tornado.web.RequestHandler):
```

```
    def set_default_headers(self):
        print("setting headers!!!")
        self.set_header("Access-Control-Allow-Origin", "*")


    def get(self):
        T = str(temperature)
        H = str(humidity)
        self.write(json.dumps(ps.generate_values()))

class recordDataRequestHandler(tornado.web.RequestHandler):

    def set_default_headers(self):
        print("setting headers!!!")
        self.set_header("Access-Control-Allow-Origin", "*")


    def post(self):
        ps = PseudoSensor()
        humidity,temperature = ps.generate_values()
        query = "INSERT INTO weather (temperature, humidity, date) VALUES ( '%f', '%f',
datetime('now'))"
        value = (temperature, humidity)
        cursor.execute(query % value)
        connection.commit()
        print("H ",humidity)
        print("T ",temperature)
        self.write(json.dumps({"message": "Data added successfully."}))

class analyzeDataAveTempRequestHandler(tornado.web.RequestHandler):

    def set_default_headers(self):
        print("setting headers!!!")
        self.set_header("Access-Control-Allow-Origin", "*")


    def get(self):
        query = "SELECT SUM(temperature) FROM (SELECT temperature FROM weather ORDER BY
id DESC LIMIT 10)"
        cursor.execute(query)
```

```
        connection.commit()
        SUM = cursor.fetchall()
        print(SUM)
        TOTAL = re.findall(r"[-+]?\d*\.\d+|\d+",str(SUM))
        print(type(TOTAL))
        print(TOTAL)
        total = float(TOTAL[0])
        print(type(total))
        print(total)
        average = total/10
        print("average ",average)
        self.write(json.dumps(str(average)))

class analyzeDataAveHumidityRequestHandler(tornado.web.RequestHandler):

    def set_default_headers(self):
        print("setting headers!!!")
        self.set_header("Access-Control-Allow-Origin", "*")


    def get(self):
        query = "SELECT SUM(humidity) FROM (SELECT humidity FROM weather ORDER BY id
DESC LIMIT 10)"
        cursor.execute(query)
        connection.commit()
        SUM = cursor.fetchall()
        print(SUM)
        TOTAL = re.findall(r"[-+]?\d*\.\d+|\d+",str(SUM))
        print(type(TOTAL))
        print(TOTAL)
        total = float(TOTAL[0])
        print(type(total))
        print(total)
        average = total/10
        print("average ",average)
        self.write(json.dumps(str(average)))

class analyzeDataMinTempRequestHandler(tornado.web.RequestHandler):

    def set_default_headers(self):
```

```
        print("setting headers!!!")
        self.set_header("Access-Control-Allow-Origin", "*")


    def get(self):
        query = "SELECT MIN(temperature) FROM weather"
        cursor.execute(query)
        connection.commit()
        MINIMUM = cursor.fetchall()
        print(MINIMUM)
        MIN = re.findall(r"[-+]?\d*\.\d+|\d+",str(MINIMUM))
        print(type(MIN))
        print(MIN)
        MIN = float(MIN[0])
        print(type(MIN))
        print(MIN)
        print("minimum ",MIN)
        self.write(json.dumps(str(MIN)))

class analyzeDataMinHumidityRequestHandler(tornado.web.RequestHandler):

    def set_default_headers(self):
        print("setting headers!!!")
        self.set_header("Access-Control-Allow-Origin", "*")


    def get(self):
        query = "SELECT MIN(humidity) FROM weather"
        cursor.execute(query)
        connection.commit()
        MINIMUM = cursor.fetchall()
        print(MINIMUM)
        MIN = re.findall(r"[-+]?\d*\.\d+|\d+",str(MINIMUM))
        print(type(MIN))
        print(MIN)
        MIN = float(MIN[0])
        print(type(MIN))
        print(MIN)
        print("minimum ",MIN)
        self.write(json.dumps(str(MIN)))
```

```
class analyzeDataMaxTempRequestHandler(tornado.web.RequestHandler):

    def set_default_headers(self):
        print("setting headers!!!")
        self.set_header("Access-Control-Allow-Origin", "*")


    def get(self):
        query = "SELECT MAX(temperature) FROM weather"
        cursor.execute(query)
        connection.commit()
        MAXIMUM = cursor.fetchall()
        print(MAXIMUM)
        MAX = re.findall(r"[-+]?\d*\.\d+|\d+",str(MAXIMUM))
        print(type(MAX))
        print(MAX)
        MAX = float(MAX[0])
        print(type(MAX))
        print(MAX)
        print("maximum ",MAX)
        self.write(json.dumps(str(MAX)))

class analyzeDataMaxHumidityRequestHandler(tornado.web.RequestHandler):

    def set_default_headers(self):
        print("setting headers!!!")
        self.set_header("Access-Control-Allow-Origin", "*")


    def get(self):
        query = "SELECT MAX(humidity) FROM weather"
        cursor.execute(query)
        connection.commit()
        MAXIMUM = cursor.fetchall()
        print(MAXIMUM)
        MAX = re.findall(r"[-+]?\d*\.\d+|\d+",str(MAXIMUM))
        print(type(MAX))
        print(MAX)
        MAX = float(MAX[0])
```

```
        print(type(MAX))
        print(MAX)
        print("maximum ",MAX)
        self.write(json.dumps(str(MAX)))

class tableDisplayRequestHandler(tornado.web.RequestHandler):

    def set_default_headers(self):
        print("setting headers!!!")
        self.set_header("Access-Control-Allow-Origin", "*")


    def get(self):
        query = "SELECT * FROM (SELECT * FROM weather ORDER BY id DESC LIMIT 10)
ORDER BY id ASC"
        cursor.execute(query)
        result = cursor.fetchall()
        print(result)
        self.write(json.dumps(result))

class exitRequestHandler(tornado.web.RequestHandler):

    def set_default_headers(self):
        print("setting headers!!!")
        self.set_header("Access-Control-Allow-Origin", "*")


    def get(self):
        sys.exit()

class mainRequestHandler(tornado.websocket.WebSocketHandler):
    def get(self):
        self.render('index.html')

class WSHandler(tornado.websocket.WebSocketHandler):
    def open(self):
        print ('new connection')

    def on_message(self, message):
        print ('message received:  %s' % message)
```

```
        # Reverse Message and send it back
        print ('sending back message: %s' % message[::-1])
        self.write_message(message[::-1])

    def on_close(self):
        print ('connection closed')

    def check_origin(self, origin):
        return True

application = tornado.web.Application([
    (r'/ws', WSHandler),
    (r'/sensor', mainRequestHandler),
    (r'/readData', readDataRequestHandler),
    (r'/recordData', recordDataRequestHandler),
    (r'/analyzeDataAveTemp', analyzeDataAveTempRequestHandler),
    (r'/analyzeDataAveHumidity', analyzeDataAveHumidityRequestHandler),
    (r'/analyzeDataMinTemp', analyzeDataMinTempRequestHandler),
    (r'/analyzeDataMinHumidity', analyzeDataMinHumidityRequestHandler),
    (r'/analyzeDataMaxTemp', analyzeDataMaxTempRequestHandler),
    (r'/analyzeDataMaxHumidity', analyzeDataMaxHumidityRequestHandler),
    (r'/tableDisplay', tableDisplayRequestHandler),
    (r'/exit', exitRequestHandler)
])


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()
    app.run(debug=True)
```

2.2 index.html

Breaklines

```
<!DOCTYPE html>

<html>
<head>

    <title>Sensor</title>
    <style>
    .progress {
    position: relative;
    width: 100%;
    height: 60px;
    background: #9cbab4;
    border-radius: 5px;
    overflow: hidden;
    }

    .progress__fill {
    width: 0%;
    height: 100%;
    background: #009579;
    transition: all 0.2s;
    }

    .progress__text {
    position: absolute;
    top: 50%;
    right: 5px;
    transform: translateY(-50%);
    font: bold 14px "Quicksand", sans-serif;
    color: #ffffff;
    }
    table {
        font-family: arial, sans-serif;
        border-collapse: collapse;
        width:100%;
        padding:4px
    }
    td, th {
        border: 1px solid #CCCCCC;
        padding: 8px;
```

```
    }
    th {
        font-weight: bold;
        text-transform: uppercase;
    }
    .wrapper {
      display:grid;
      grid-template-columns: 70% 30%;
      grid-gap:1em;
    }
    #readData {
      background:#079992;
      text-align:center;
      padding:4px;
    }
    #analyzeData {
      background:#78e08f;
      text-align:center;
      padding:4px;
    }

    #progressBar {
      background:#6a89cc;
      text-align:center;
      padding:4px;
    }

    </style>
</head>

<body>
    <div class = "wrapper">
    <div>

    <button style="background-color:#333333;color:#00FF00;border-radius:5px; font-size:1em; padding:4px" id =
tableDisplay >Display</button>

    <table >
        <thead>
        <tr >
            <th>ID</th>
            <th>Temperature</th>
            <th>Humidity</th>
            <th>Time</th>
```

```
        </tr>
        </thead>
        <tbody id = "tableData"></tbody>
    </table>

    </div>
    <div id = "analyzeData" >
    <button style="background-color:#333333;color:#00FF00;border-radius:5px; font-size:1em" id = analyzeData
>Analyze</button>
    <p>Average Temperature</p>
    <p id="avetempdisplay">data</p>
    <p>Average Humidity</p>
    <p id="avehumiditydisplay">data</p>
    <p>Minimum Temperature</p>
    <p id="mintempdisplay">data</p>
    <p>Minimum Humidity</p>
    <p id="minhumiditydisplay">data</p>
    <p>Maximum Temperature</p>
    <p id="maxtempdisplay">data</p>
    <p>Maximum Humidity</p>
    <p id="maxhumiditydisplay">data</p>

     </div>

    <div id = "progressBar" >
    <button style="background-color:#333333;color:#00FF00;border-radius:5px; font-size:1em; padding:4px
text-align:center" id = "recordData" >Record 10 Values</button>
    <p id="recordingData">data</p>
    <div class="progress">
    <div class="progress__fill"></div>
    <span class="progress__text">0%</span>
    </div>
    </div>

    <div id = "readData" >
    <button style="background-color:#333333;color:#00FF00;border-radius:5px; padding:4px; font-size:1em" id =
"readData" >Read Data</button>
    <p id="tempdisplay">data</p>
    <p id="humiditydisplay">data</p>

    <p id = "tempRange">Temperature in normal range</p>
    <p id = "humidityRange">Humidity in normal range</p>
```

```
   <button onclick="return window_close_onclick();"
style="background-color:#333333;color:#00FF00;border-radius:5px; padding:4px; font-size:1em"
>EXIT</button>
   </div>
   </div>

   <script>

   readData = document.getElementById("readData")
   tempRangeColor = document.getElementById("tempRange")
   humidityRangeColor = document.getElementById("humidityRange")


   readData.addEventListener("click", e => {
            fetch('http://localhost:8888/readData')
            .then(response => response.json())
            .then(jsonResponse => { console.log(jsonResponse)
            document.getElementById("tempdisplay").innerHTML = jsonResponse[1].toString();
            document.getElementById("humiditydisplay").innerHTML = jsonResponse[0].toString();
            if (jsonResponse[1]>30 && jsonResponse[1]<40){
            document.getElementById("tempRange").innerHTML = "Temperature in normal range";
            tempRangeColor.style.color = 'black';
}            else {
            document.getElementById("tempRange").innerHTML = "Warning!!! temperature beyond normal
condition";
            tempRangeColor.style.color = 'red';
}
            if (jsonResponse[0]>20 && jsonResponse[0]<50){
            document.getElementById("humidityRange").innerHTML = "Humidity in normal range";
            humidityRangeColor.style.color = 'black';
}            else {
            document.getElementById("humidityRange").innerHTML = "Warning!!! humidity beyond normal
condition";
            humidityRangeColor.style.color = 'red';
}
})
})

   function updateProgressBar(progressBar, value) {
            value = Math.round(value);
            progressBar.querySelector(".progress__fill").style.width = `${value}%`;
            progressBar.querySelector(".progress__text").textContent = `${value}%`;
}
   myProgressBar = document.querySelector(".progress")
```

```
    bar = 0

    recordData = document.getElementById("recordData")

    recordData.addEventListener("click", e => {
            for (let i = 0; i < 10; i++) {

            fetch('http://localhost:8888/recordData', {"method" : "POST"})
            .then(response => response.json()
            .then(jsonResponse => {
            document.getElementById("recordingData").innerHTML = bar + " " + jsonResponse.message;
            setTimeout( console.log(bar + " " + jsonResponse.message+ new Date()), 3000 * i)
            updateProgressBar(myProgressBar, bar*10)
            bar =+ i + 1;
}))
}})
    analyzeData = document.getElementById("analyzeData")

    analyzeData.addEventListener("click", e => {
            fetch('http://localhost:8888/analyzeDataAveTemp')
            .then(response => response.json())
            .then(jsonResponse => { console.log(jsonResponse)
            document.getElementById("avetempdisplay").innerHTML = jsonResponse.toString();
})
            fetch('http://localhost:8888/analyzeDataAveHumidity')
            .then(response => response.json())
            .then(jsonResponse => { console.log(jsonResponse)
            document.getElementById("avehumiditydisplay").innerHTML = jsonResponse.toString();
})
            fetch('http://localhost:8888/analyzeDataMinTemp')
            .then(response => response.json())
            .then(jsonResponse => { console.log(jsonResponse)
            document.getElementById("mintempdisplay").innerHTML = jsonResponse.toString();
})
            fetch('http://localhost:8888/analyzeDataMinHumidity')
            .then(response => response.json())
            .then(jsonResponse => { console.log(jsonResponse)
            document.getElementById("minhumiditydisplay").innerHTML = jsonResponse.toString();
})
            fetch('http://localhost:8888/analyzeDataMaxTemp')
            .then(response => response.json())
            .then(jsonResponse => { console.log(jsonResponse)
            document.getElementById("maxtempdisplay").innerHTML = jsonResponse.toString();
})
```

```
        fetch('http://localhost:8888/analyzeDataMaxHumidity')
        .then(response => response.json())
        .then(jsonResponse => { console.log(jsonResponse)
        document.getElementById("maxhumiditydisplay").innerHTML = jsonResponse.toString();
})
})

   tableDisplay = document.getElementById("tableDisplay")

   tableDisplay.addEventListener("click", e => {
        fetch('http://localhost:8888/tableDisplay')
        .then(response => response.json())
        .then(jsonResponse => { console.log(jsonResponse)
        console.log(typeof jsonResponse)
        console.log(jsonResponse[0][1])
        console.log(jsonResponse[0])
        loadTableData(jsonResponse);
})
})
        function loadTableData(jsonResponse){
        tableBody = document.getElementById('tableData')
        dataHtml = " ";
        for (let i = 0; i < 10; i++) {
           dataHtml +=
`<tr><td>${jsonResponse[i][0]}</td><td>${jsonResponse[i][1]}</td><td>${jsonResponse[i][2]}</td><td>${jso
nResponse[i][3]}</td></tr>`;
        }
        console.log(dataHtml)
        tableBody.innerHTML = dataHtml;
        }


   function window_close_onclick(){
        if(confirm("Do you want to exit?")){
        fetch('http://localhost:8888/exit')

        let new_window =
        open(location, '_self');

        new_window.close();

        return false;
        }
}
```

```
  </script>
</body>
</html>
```

2.3 websocket.html

Breaklines

```
<!doctype html>
<html>
 <head>
  <title>WebSockets check connection</title>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <meta http-equiv="refresh" content="10; url=http://localhost:8888/sensor">
  <style type="text/css">
   body {
    text-align: center;
    min-width: 500px;
   }
  </style>
  <script src="http://code.jquery.com/jquery.min.js"></script>
  <script>

   // log function
   log = function(data){
    $("div#terminal").prepend("</br>" +data);
    console.log(data);
   };

   $(document).ready(function () {
    $("div#message_details").hide()

    var ws;

    $("#open").click(function(evt) {
     evt.preventDefault();

     var host = $("#host").val();
     var port = $("#port").val();
     var uri = $("#uri").val();
```

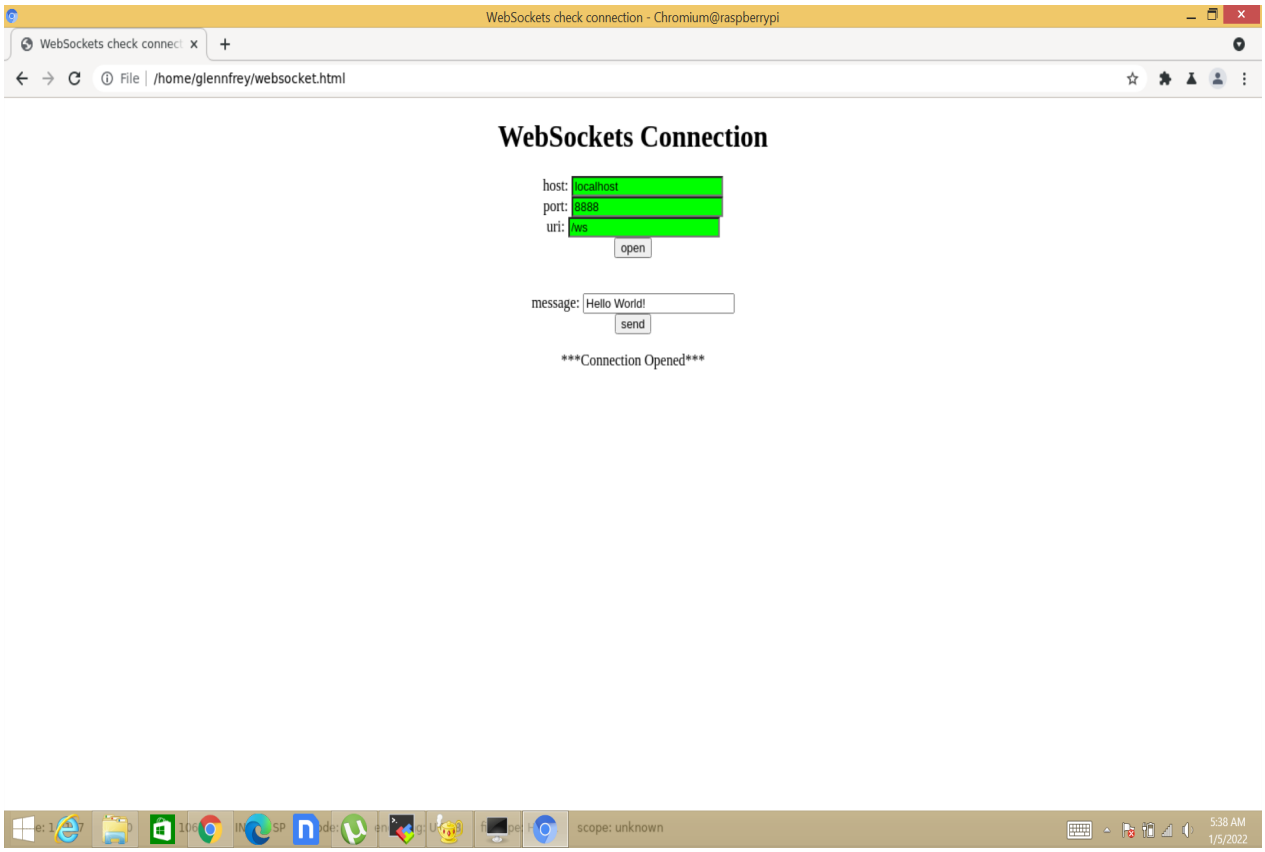
```
    // create websocket instance
    ws = new WebSocket("ws://" + host + ":" + port + uri);

    // Handle incoming websocket message callback
    ws.onmessage = function(evt) {
      log("Message Received: " + evt.data)
      alert("message received: " + evt.data);
      };

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

      };

    // Open Websocket callback
    ws.onopen = function(evt) {
      $("#host").css("background", "#00ff00");
      $("#port").css("background", "#00ff00");
      $("#uri").css("background", "#00ff00");
      $("div#message_details").show();
      log("***Connection Opened***");
    };
  });

  // Send websocket message function
  $("#send").click(function(evt) {
      log("Sending Message: "+$("#message").val());
      ws.send($("#message").val());
  });

});
let counter = document.querySelector('h1');
        let count = 1;
        setInterval(()=>{
counter.innerText = count;
count++
```

```
    if(count > 10 && host == "localhost" && port == "8888" && uri == "/ws" )
location.replace('http://localhost:8888/sensor')

},1000)
   </script>
  </head>

  <body>
   <h1>WebSockets Connection</h1>
   <div id="connection_details">
    <label for="host">host:</label>
    <input type="text" id="host" value="localhost" style="background:#ff0000;"/><br />
    <label for="port">port:</label>
    <input type="text" id="port" value="8888" style="background:#ff0000;"/><br />
    <label for="uri">uri:</label>
    <input type="text" id="uri" value="/ws" style="background:#ff0000;"/><br />
    <input type="submit" id="open" value="open" />
   </div>
   <div id="message_details">
      </br></br>
      <label for="message">message:</label>
      <input type="text" id="message" value="Hello World!"/><br />
      <input type="submit" id="send" value="send" />
   </div>
   <div id="terminal">

   </div>
  </body>
</html>
```

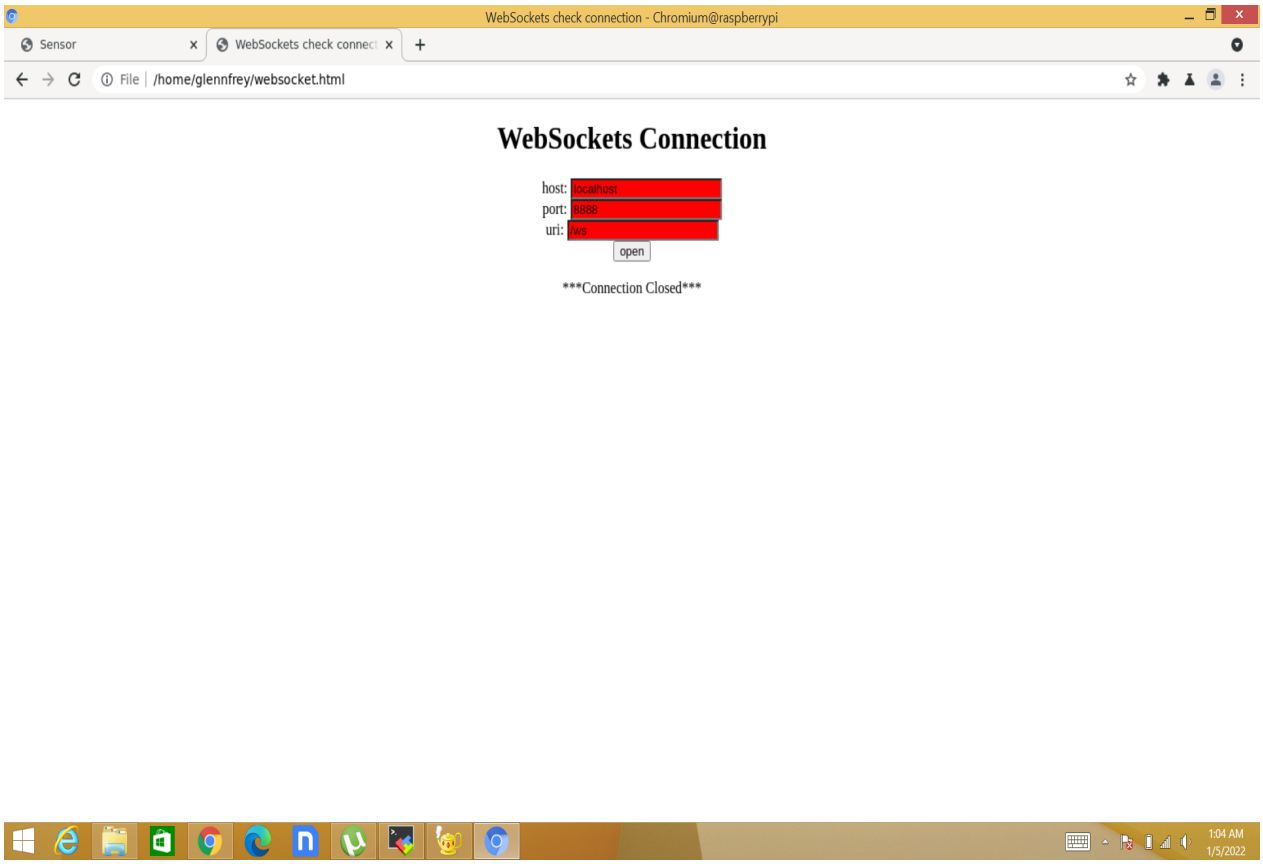
---

3. UI Images

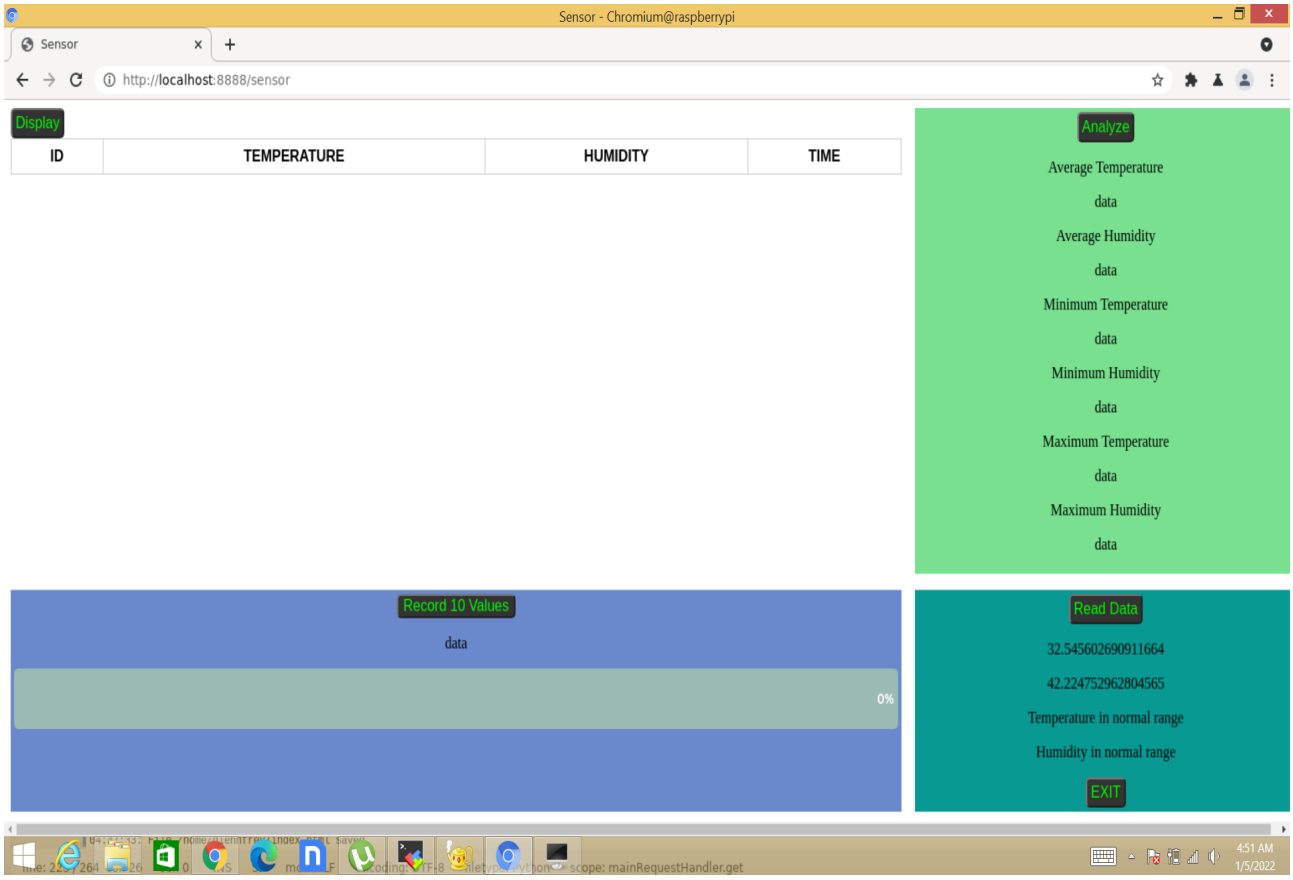
3.1 Html UI at startup



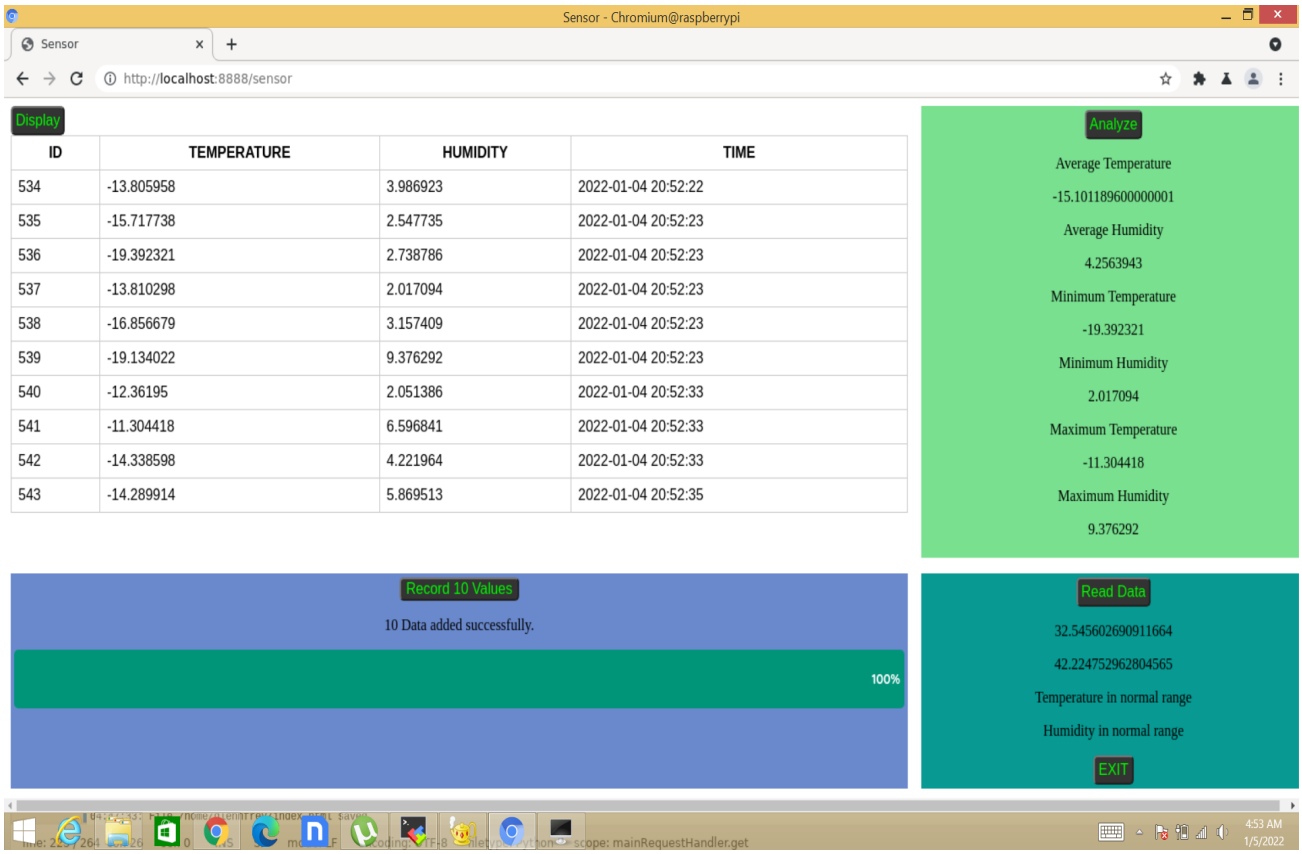
3.2 An error condition – loss of connection to the webserver or to the python code



3.3 The UI after its first single data point reading



3.4 The UI after it has calculated a 10 point average



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