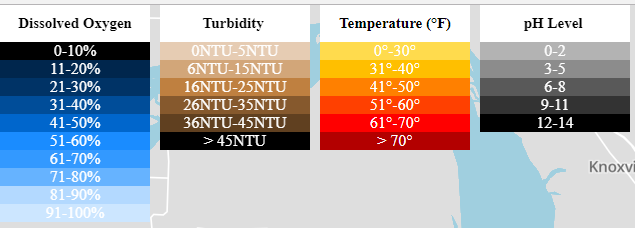
**Overview:**

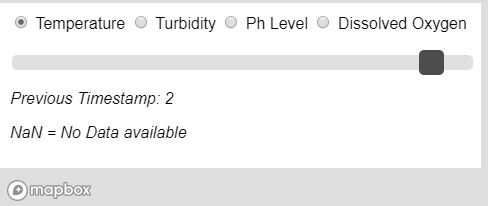
The website created for this Project consists of 3 primary files:

* Index.html
* Project.css
* sensorData.cgi

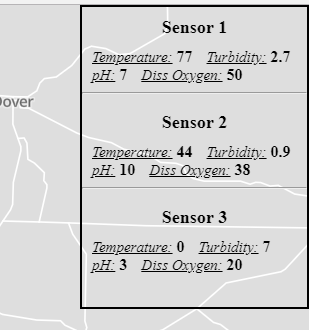
Index.html contains the HTML elements of the webpage itself, and the programmatic functionality, such as displaying the map, displaying data on the map in the form of markers, coloring polygons around the markers based on variable data, a menu to toggle between which variables are represented by the polygons, and a slider to display data from a previous timestamp. This file consists of both HTML for webpage elements and Javascript for client-side programmatic functionality. Other than the map, the HTML elements included on the webpage are the legends:



The menu containing the radio toggle and time slider:



The data display:



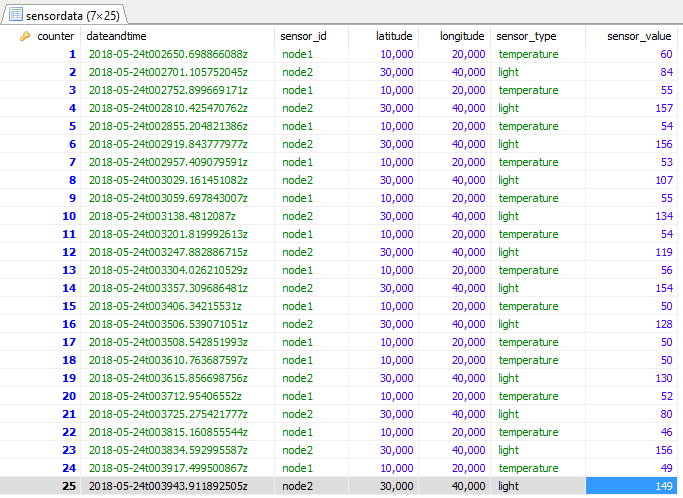
Project.css contains the styling declarations for the elements of index.html to display properly on the screen, such as the legends on the top left side of the screen, the toggle menu and slider inputs at the bottom left of the screen, and the data display menu on the right side of the screen.

sensorData.cgi is a script to be located on the server with both other files that pulls data from the database and sends that data to the webpage, where it is parsed and manipulated for the purposes of proper display of information. This file was written in Perl, and is a server-side based script.

The webpage may be modified to pull from other databases via manipulation of the sensorData.cgi file. Line 15 of this file:

my $Connection = DBI->connect("DBI:SQLite:lake.db") or die $DBI::errstr;

Must be modified in order to connect to other databases. During project development, we created a local SQLite3 database called lake.db. The statement “DBI:SQLite:lake.db” must be changed in order for the data to pulled from other sources. Similarly, the database must adhere to a specific format, similar to the following example:



Containing 7 columns in the previous example’s exact order. Order of columns matter due to the nature of the method that data is sent back to the webpage.

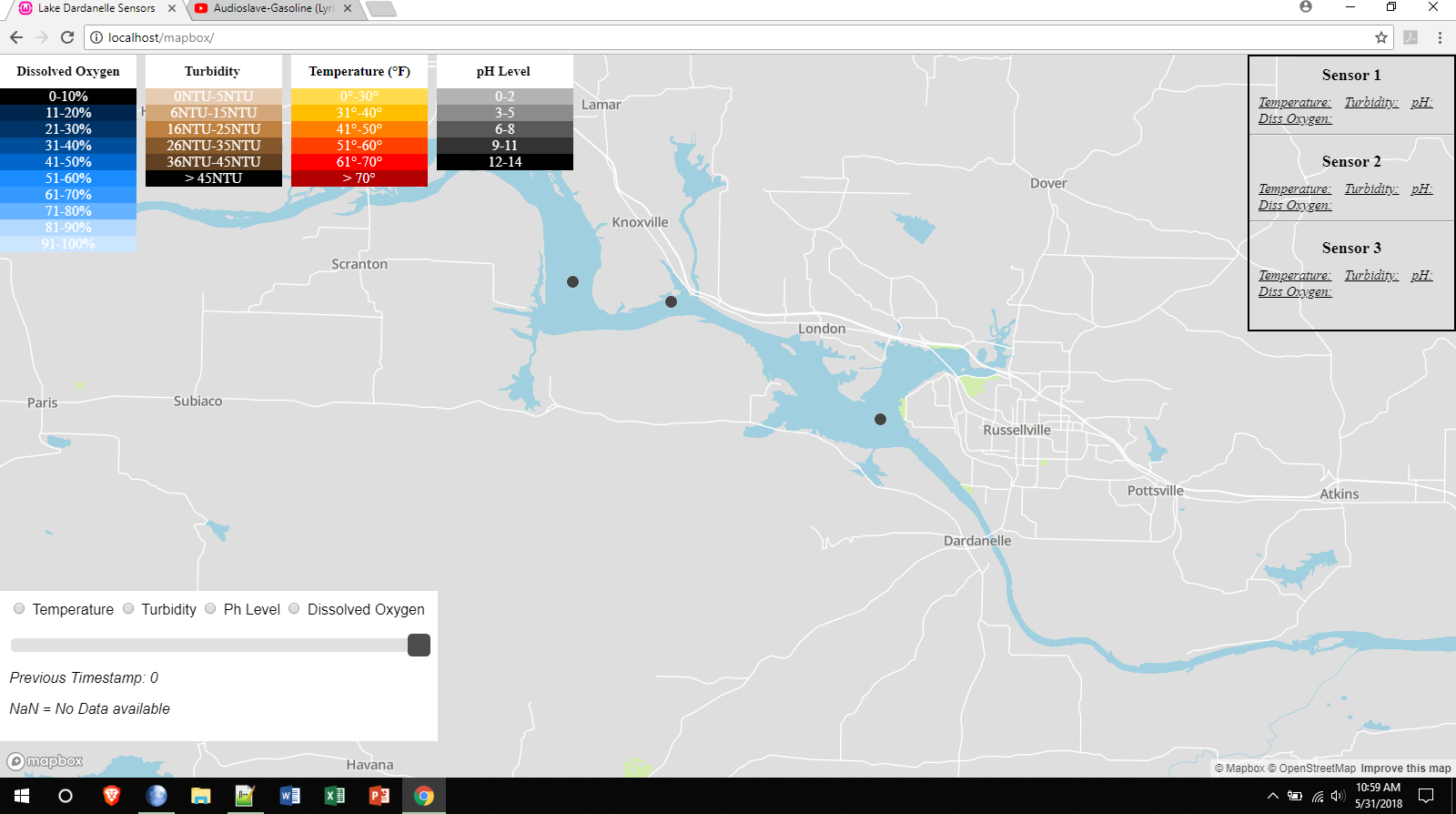
Enough data for three groups of four sensors was entered into the database for the purpose of displaying the markers to represent sensor groups. For the addition of more markers, additional code will need to be added to the web page in geoJSON format, <http://geojson.org/>, in the addLayer function from lines 637-684 of index.html, placing that sensor groups latitude and longitude in the “coordinates” feature of the geoJSON object. Because more sensor data would be pulled back from the database, the getData function, lines 55-97, would need to be added to allow for more variables as well.

Description of Functionality:

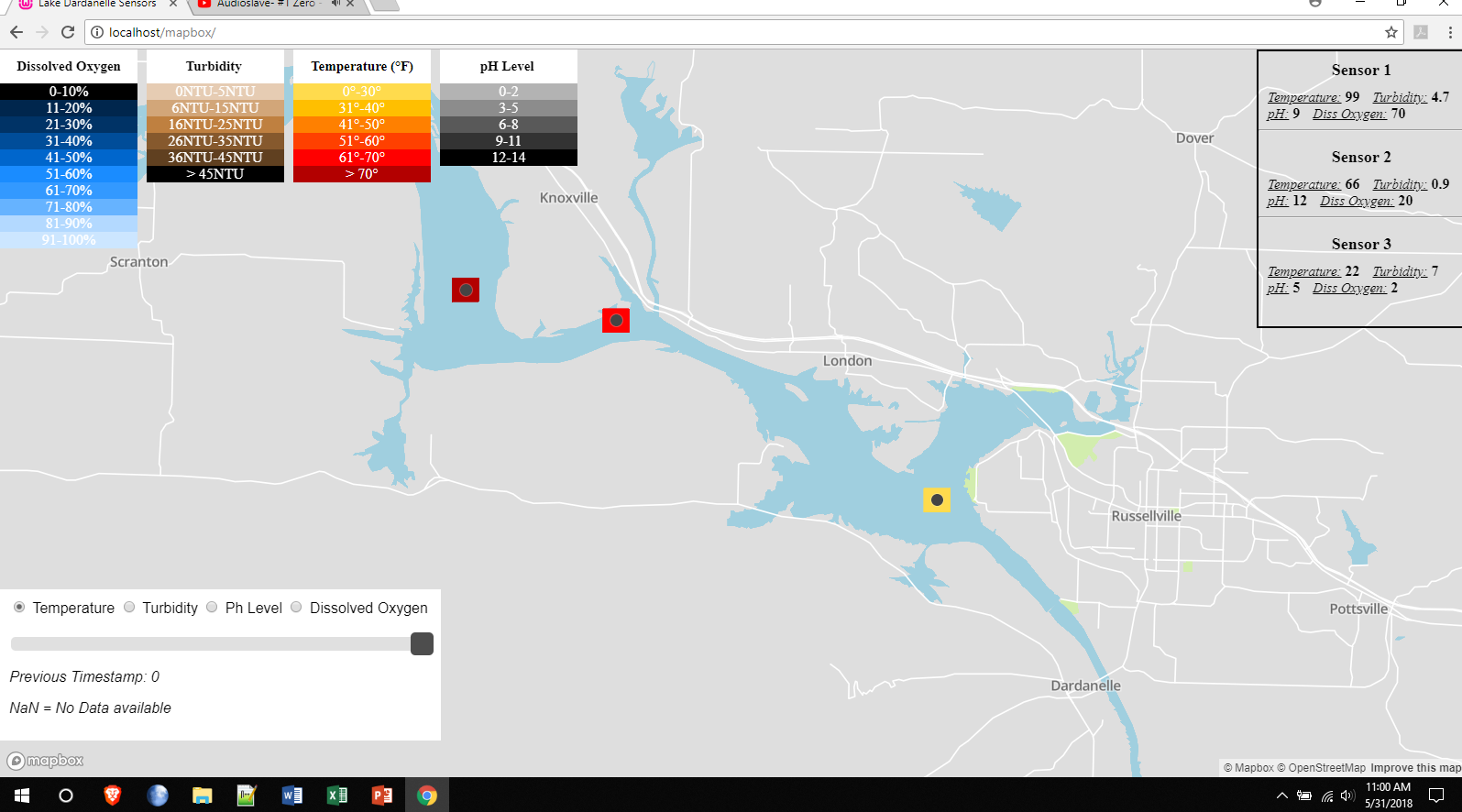
The webpage functions differently according to several different events:

* When the webpage loads
* When a radio button is toggled
* When the time slider is moved
* When a sensor marker is clicked

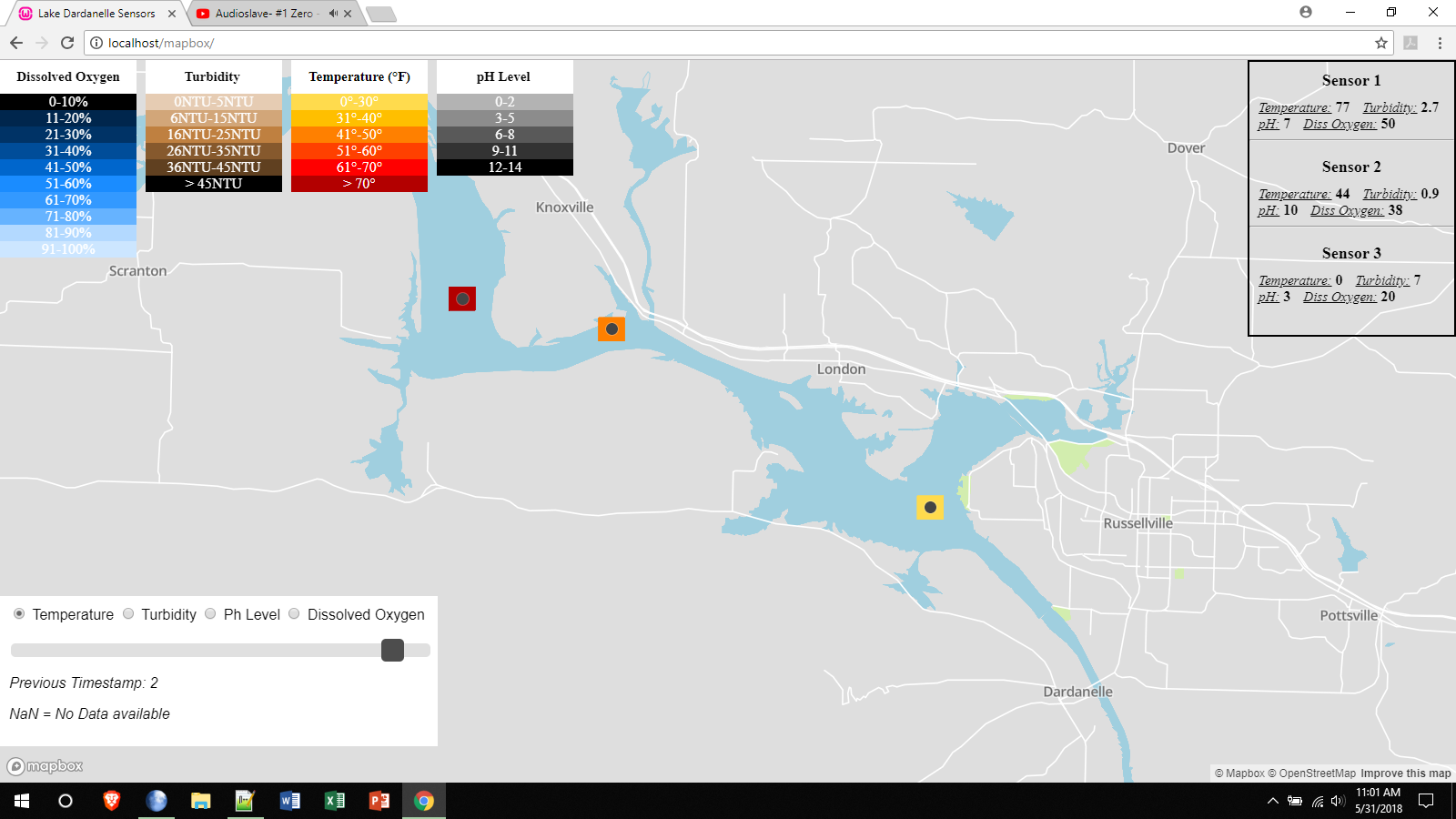
When the webpage is first loaded, the map and all of the HTML elements (legend, menu, sensor display) are displayed, and the webpage sends a request to sensorData.cgi for data. When that data is received, it comes back as a long, contiguous string, delimited by a “+”. The string is parsed into an array, and the values of the array are assigned to their corresponding variables. For example, the temperature value from sensor group 1 is assigned to a float variable called temp1. After all the data from the database is assigned to its corresponding variable, markers are displayed on the map according to the latitude and longitude of the sensor groups that were pulled from the database. These markers may be clicked to display a popup that contains its latitude and longitude, and the popup is displayed dynamically around the marker to make it readable according to where the user moves around the map. After the markers are displayed, polygons are drawn around the markers and are made invisible, so that they may be colored based which variable the user chooses in the toggle menu. The sensor values will appear in a data display on the webpage according to which sensor group to which it belongs. The following is an example of the webpage after it initially loads using Google Chrome as a browser:



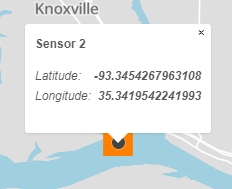
The radio button menu consists of 4 buttons, to switch between sensor variables: temperature, turbidity, pH, and dissolved oxygen. When a radio button is toggled, the polygon surrounding each of the sensor markers will become visible and be colored according to which one of the variables were selected. Legends are included on the webpage to give the user a way to visually determine the value of that variable at that sensor. For example, if temperature is clicked, a polygon around each marker is drawn and colored on a scale from light yellow to dark red to represent the heat of the water. The hotter the water is, the darker red the color around the marker will be. Each time the user clicks a different button in the toggle menu, the polygon surrounding the marker is re-colored according to the variable chosen. 4 different color scales were chosen for each variable so that the user could distinguish from one variable to another. The following is an example of the colored polygons that are drawn after a radio button is pressed:



When the time slider is moved, the webpage will send the value of the slider to the sensorData.cgi script, which will send back data from a particular timestamp. The value of the slider is subtracted from the maximum counter in the database in order to bring back data from that number of timestamps in the past. The data is sent back, and the variables are reassigned the new values, the marker polygons are re-drawn, and recolored, and the data display menu displays the updated values. The user can slide back on forth to watch the marker polygons re-color dynamically and see gradual changes in a certain variable over time. If the slider is moved to a point where no data is present, the data display shows NaN to represent an undefined value, and the marker polygons will become the most extreme color according to their legend. The following is an example of how the webpage will change after the time slider is moved:



When a sensor marker is clicked, a popup is displayed that gives the sensor number, the latitude and the longitude. The popup will disappear when another part of the map is clicked, another marker is clicked, or when the “x” in the top right corner of the marker is clicked. The popup has been styled via the project.css file to be a white color to allow for good visual acuity. The value name is displayed on the left hand side of the display and the value itself is displayed to its right. The following is an example of the popup that will display after a sensor marker is clicked.



Information for Deployment or Continued Development:

For the deployment of this webpage, several considerations must be made. The webserver hosting the webpage must support CGI scripting and it must contain a Perl compiler so that the sensorData.cgi script can run properly to pull data from the database. The sensorData.cgi file must be located within the “**cgi-bin**” directory of the server, or else the index.html page will need to be modified. Line 58 of index.html gives the directory of the sensorData.cgi file:

"/cgi-bin/sensorData.cgi",

If this directory is changed, this line must be changed accordingly. Similarly, because this webpage prototype was built using a local database, the location and name of the database matters as well. If deploying using a local database hosted on the webserver, this database file must be named lake.db and must be hosted within the “**cgi-bin**” directory of the server as well. If database location or name changes, the sensorData.cgi must be modified as previously stated, on line 15:

my $Connection = DBI->connect("DBI:SQLite:lake.db") or die $DBI::errstr;

In addition, the webpage was built and designed for desktop and laptop viewing. Smaller screens will not display properly. During development, a multitude of web browsers were used to make sure that the web page worked and displayed properly over a multitude of different browsers, such as Chrome, Firefox, Edge, Pale Moon, and Brave. Microsoft’s Internet Explorer will not display the webpage properly.

Continued development of this webpage may difficult without an understanding of the functions written in the program. For that purpose, the following is a list of the functions for the index.html, along with a description of their purpose, input, output, and the context of when they should be used:

* getData(click) – lines 55 - 97
  + Sends a POST request to sensorData.cgi file, and assigns the feedback from the request to their corresponding variables. This function is called when the web document is ready, and when the time slider value changes, and it must be sent the value of the time slider.
* $(document).ready(function() – lines 99 – 184
  + Called as soon as webpage fully loads. Contains an event listener for the time slider. Calls getData, sending it a value of 0. Displays the time slider value next to its location in the menu. Calls the functions temp, turb, ph, and oxy to draw the polygons around the markers and then runs through a loop to make them initially invisible. Displays sensor data in the data display menu.
* Functions top\_lat(a), bot\_lat(a), left\_lon(a), and right\_lon(a) – lines 188 – 200
  + Called when temp, turb, ph, and oxy functions run. Used to draw polygons around sensor markers.
* Functions temp\_color(a), turb\_color(a), ph\_color(a), and oxy\_color(a) – lines 203 – 306
  + Called whenever the polygons surrounding the markers are either drawn, or re-colored. Uses the value of the sensor variable to determine the color the polygon should be
* Display(ly1, ly2, ly3)
  + Called whenever a button on the radio toggle is clicked. Used to ensure that only the polygons corresponding to the variable selected on the menu are visible, so that there are no overlapping polygons displayed of different color over the same marker.
* Functions temp(), turb(), ph(), and oxy() – lines 356-625
  + Called whenever the webpage first loads. Calls functions to draw polygons around markers, and colors those polygons by sending its corresponding sensor variable to one of the color functions. Adds the drawn polygons as a layer to the map and makes them visible by default.
* Map.on(“load”, function())
  + Called whenever the map finishes loading on the webpage. Displays markers on the webpage according to its corresponding latitude and longitude, and stores both variables in the properties of the marker object so that they may displayed whenever the marker is clicked on.
* Map.on(‘click’, ‘markers’, function (e)
  + Called whenever one of the markers on the map is clicked. Displays a popup adjacent to the marker with description information of the markers.
* repaintPolygons()
  + Called whenever the time slider value is changed. Used to recolor the polygons based on the new sensor variables values that were pulled from the database based on the new time slider value.
* Slider\_1change(val1)
  + Called whenever the time slider value is changed. Calls getData function so that new data can be taken from the database. Calls repaintPolygons function so that the polygons can be redrawn based on those values. Calls info function so that the data display menu will display the newly received sensor variables.
* Info()
  + Called whenever the webpage first loads and whenever the time slider value changes. Used to update the information displayed in the data display menu on the screen with the newest values.