SmartBuoy

Design Specification Document

TABLE OF CONTENTS

EXECUTIVE SUMMARY

Background: SmartBuoy was designed as a fully self-contained water quality probe, able to be deployed in a body of water and monitored remotely via a GSM cellular network. SmartBuoy is an economical solution to water quality monitoring, meant for both amateur and professional researchers.

The device is powered by 5-volt lithium ion battery, recharged by an onboard solar panel and includes sensors to measure electrical conductivity, pH, temperature, turbidity, and total dissolved solids. Location data is provided by an Adafruit GPS module. A data reading is taken every 15 seconds and streamed live. Every hour a reading is sent to a database for storage.

Scope: This portion of the project provides a dashboard to retrieve and view the data readings, both live and historic. In addition to the dashboard, a second application was developed to simulate the operation of the SmartBuoy. This allows for testing and review of the dashboard application without requiring the deployment of the device. This is also beneficial in situations where development and testing are done without access to the SmartBuoy. In addition to the dashboard and simulator, a database has been created for data storage.

Dweet.io is used as an intermediary for the retrieval of live updates from the device. In this case, the simulator generates the reading data and sends it to Dweet.io using an Http client. Dweet.io stores the five most recent data readings. The dashboard then retrieves the most recent reading, also using an Http client. These readings are sent and retrieved every 15 seconds. Once per hour a reading is also sent to the database for storage. The database was built using Microsoft Azure and is accessed using LINQ – part of the Microsoft.NET Framework.

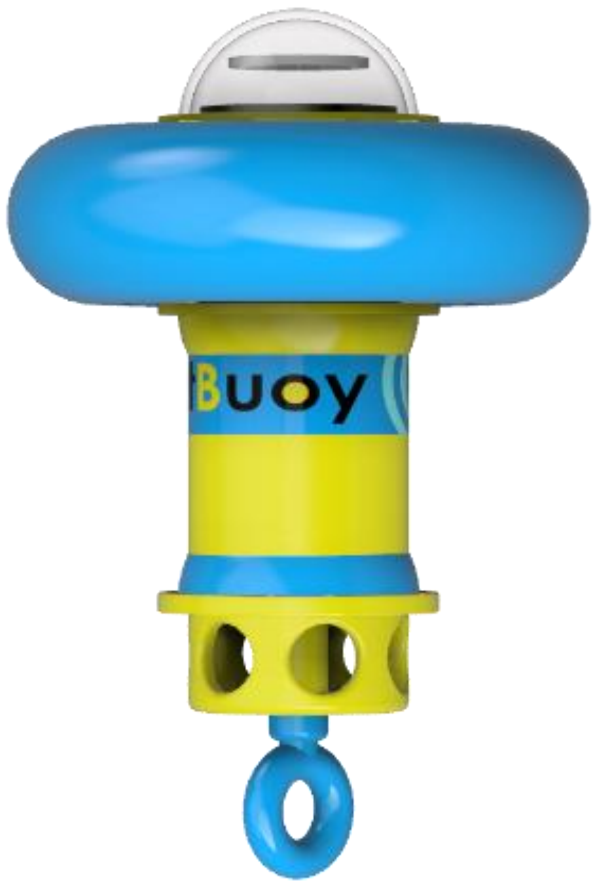
Operation: After opening the Dashboard application you will be presented with the interface. Five main areas will be visible: gauge cluster, map, data table, line chart, and controls. There are two datetime pickers in the controls section; one for the start date and one for the end date. Setting these values will provide the date range of the reading to be retrieved from the database.

Clicking the button marked “GET DATA” will execute the retrieval. You will see the data appear as rows in the data table and in the line chart. You will also see the locations of each reading marked on the map. A row of the data table will be highlighted in green; the values of this row’s cells are mapped to the gauges. Use the range slider to move through the rows of data. You will see the gauge values change to match the newly highlighted row. The cell values of the selected row will also be the final data added to the map and chart.

Clicking the button marker “GO LIVE” will retrieve the latest data reading from Dweet.io. If the simulator running, you will see data begin to fill the data table, chart, and map. The gauges will reflect the most recent data. If the simulator is not running, a single reading – the most recent in Dweet.io will be displayed – no other data will appear.

To use the simulator, first launch the application, you will then see the interface. There is a large power button. Clicking this button starts/stops data generation and transmission. As the simulator operates, data is generated and transmitted. Picture boxes containing images of LEDs are used indicate that the unit operational. The application uses LED images indicating power status, live streaming, database transmission, and generation of each data point.

SMARTBUOY DEVICE (for reference purposes only)



Measurements:

Overall Length:	450 mm
Body Diameter:	100 mm
Float Diameter:	300 mm
Weight:	1.2 kg

Materials:

Body:	PLA Thermoplastic
Dome:	Acrylic
Float:	Vinyl

Components:

Acrylic Dome

PLA Body

Vinyl Floatation Ring

Solar Panel

5-volt Lithium Battery

Mayfly Microcontroller

SD Datalogger

SIM 900 Cellular Modem

Adafruit GPS Module

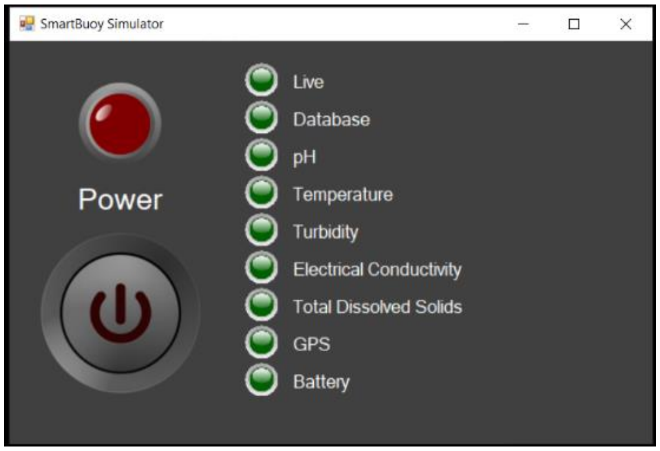
Thermometer

Turbidity Probe

pH Probe

Electrical Conductivity Probe

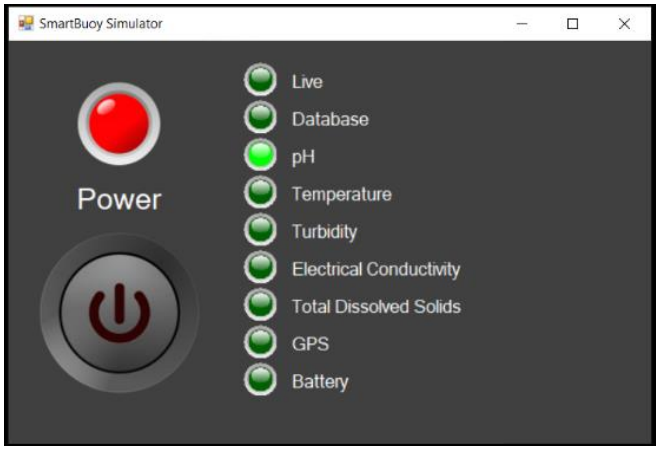
SCENARIOS SIMULATOR

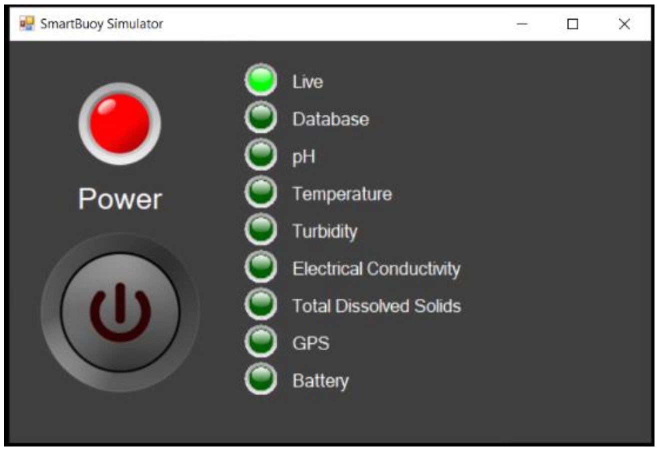


POWER OFF

This is the initial state of the simulator program after launch. Notice none of the LED indicators are illuminated.

Clicking the power button begins data generation and transmission.





POWER ON

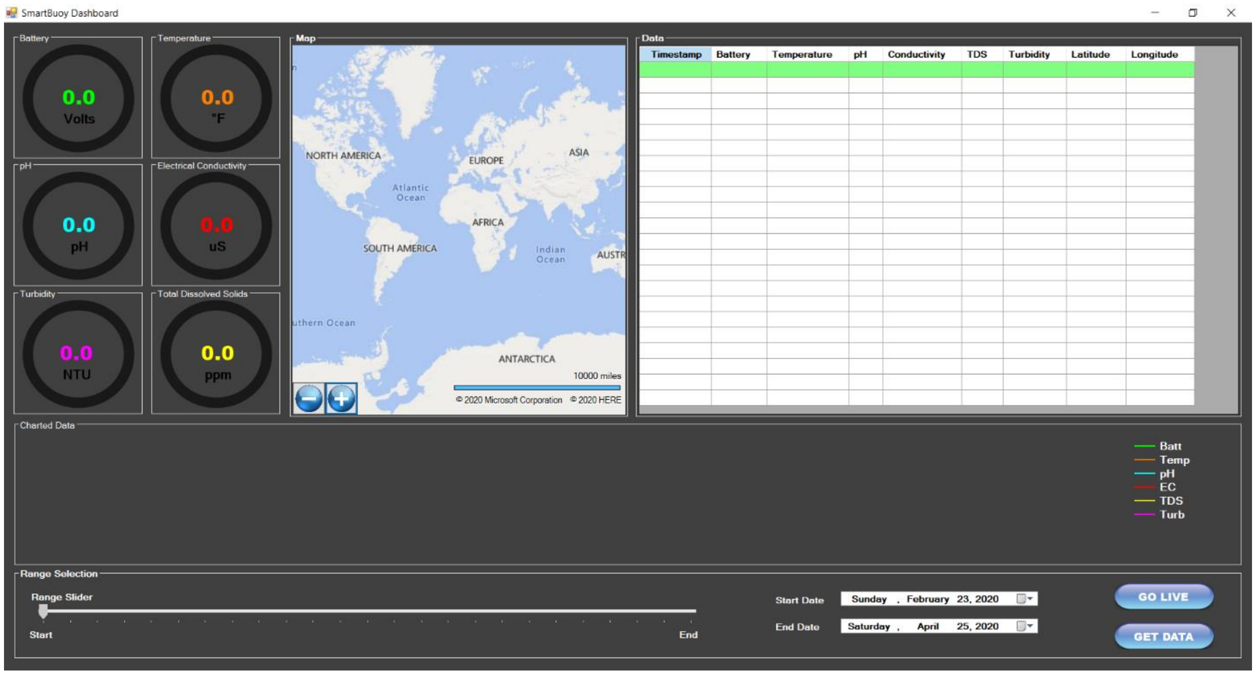
These two images show the simulator after the power button has been clicked.

The red power indicator is illuminated showing that the simulator is active.

The green indicators are illuminated as the simulator performs the operation associated with it. Notice the pH indicator illuminated indicating the pH is being measured. In the second image the Live indicator shows that the data is being transmitted to Dweet.io

The cycle of flashing LEDs will continue as data is generated and transmitted. Clicking the power button again will stop the process and reset the simulator to its inactive state.

SCENARIOS DASHBOARD



Timestamp	Battery	Temperature	pH	Conductivity	TDS	Turbidity	Latitude	Longitude

Start Date Sunday , February 23, 2020

End Date Saturday , April 25, 2020

INITIAL STATE OF APPLICATION

This is the initial state of the dashboard program after launch.

Notice the five main areas:

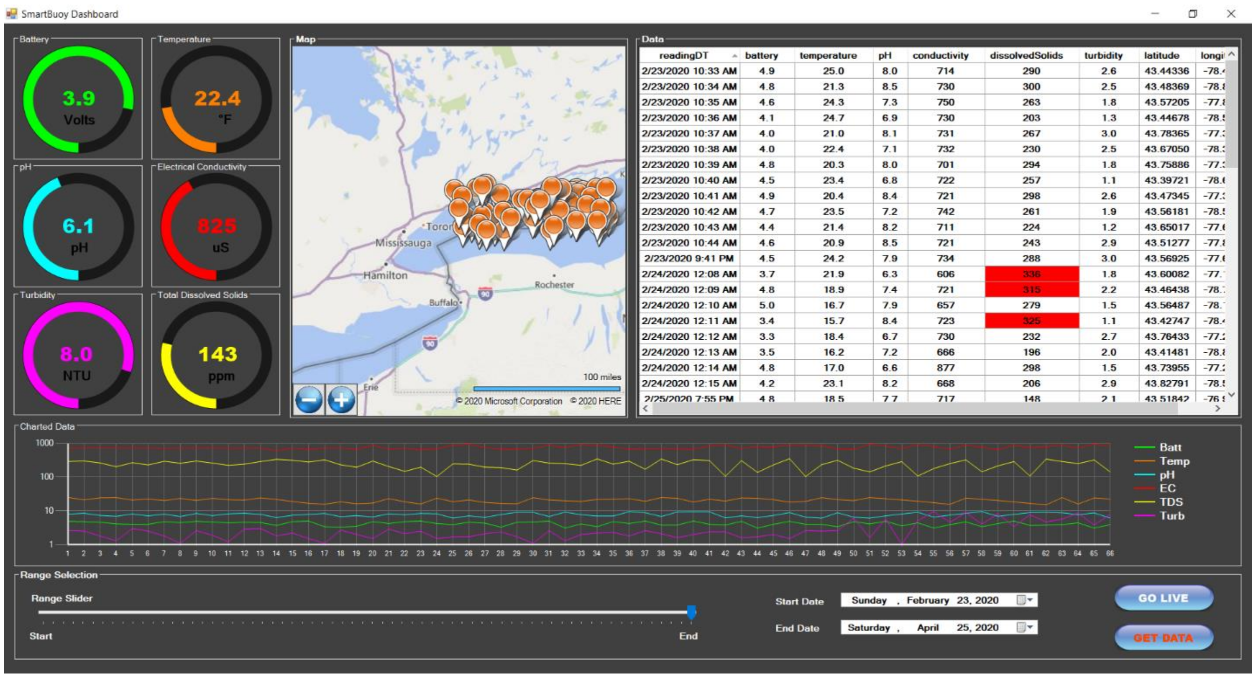
- **Gauge Cluster**
- **Map**
- **Data Table**
- **Line Chart**
- **Controls**

Begin by selecting a start and end date. Then click the GET DATA button.

or

Begin by clicking GO LIVE to begin receiving live data from the simulator.

SCENARIOS DASHBOARD



Data

readingDT	battery	temperature	pH	conductivity	dissolvedSolids	turbidity	latitude	longi
2/23/2020 10:33 AM	4.9	25.0	8.0	714	290	2.6	43.44336	-78.4
2/23/2020 10:34 AM	4.8	21.3	8.5	730	300	2.5	43.48369	-78.8
2/23/2020 10:35 AM	4.6	24.3	7.3	750	263	1.8	43.57205	-77.8
2/23/2020 10:36 AM	4.1	24.7	6.9	730	203	1.3	43.44678	-78.5
2/23/2020 10:37 AM	4.0	21.0	8.1	731	267	3.0	43.78365	-77.3
2/23/2020 10:38 AM	4.0	22.4	7.1	732	230	2.5	43.67050	-78.3
2/23/2020 10:39 AM	4.8	20.3	8.0	701	294	1.8	43.75886	-77.3
2/23/2020 10:40 AM	4.5	23.4	6.8	722	257	1.1	43.39721	-78.6
2/23/2020 10:41 AM	4.9	20.4	8.4	721	298	2.6	43.47345	-77.3
2/23/2020 10:42 AM	4.7	23.5	7.2	742	261	1.9	43.56181	-78.5
2/23/2020 10:43 AM	4.4	21.4	8.2	711	224	1.2	43.65017	-77.6
2/23/2020 10:44 AM	4.6	20.9	8.5	721	243	2.9	43.51277	-77.8
2/23/2020 9:41 PM	4.5	24.2	7.9	734	288	3.0	43.56925	-77.6
2/24/2020 12:08 AM	3.7	21.9	6.3	606	336	1.8	43.60082	-77.7
2/24/2020 12:09 AM	4.8	18.9	7.4	721	315	2.2	43.46438	-78.1
2/24/2020 12:10 AM	5.0	16.7	7.9	657	279	1.5	43.56487	-78.7
2/24/2020 12:11 AM	3.4	15.7	8.4	723	325	1.1	43.42747	-78.4
2/24/2020 12:12 AM	3.3	18.4	6.7	730	232	2.7	43.76433	-77.2
2/24/2020 12:13 AM	3.5	16.2	7.2	666	196	2.0	43.41481	-78.8
2/24/2020 12:14 AM	4.8	17.0	6.6	877	298	1.5	43.73955	-77.2
2/24/2020 12:15 AM	4.2	23.1	8.2	668	206	2.9	43.82791	-78.5
2/25/2020 7:55 PM	4.8	18.5	7.7	717	148	2.1	43.51842	-76.9

Start Date Sunday , February 23, 2020

End Date Saturday , April 25, 2020

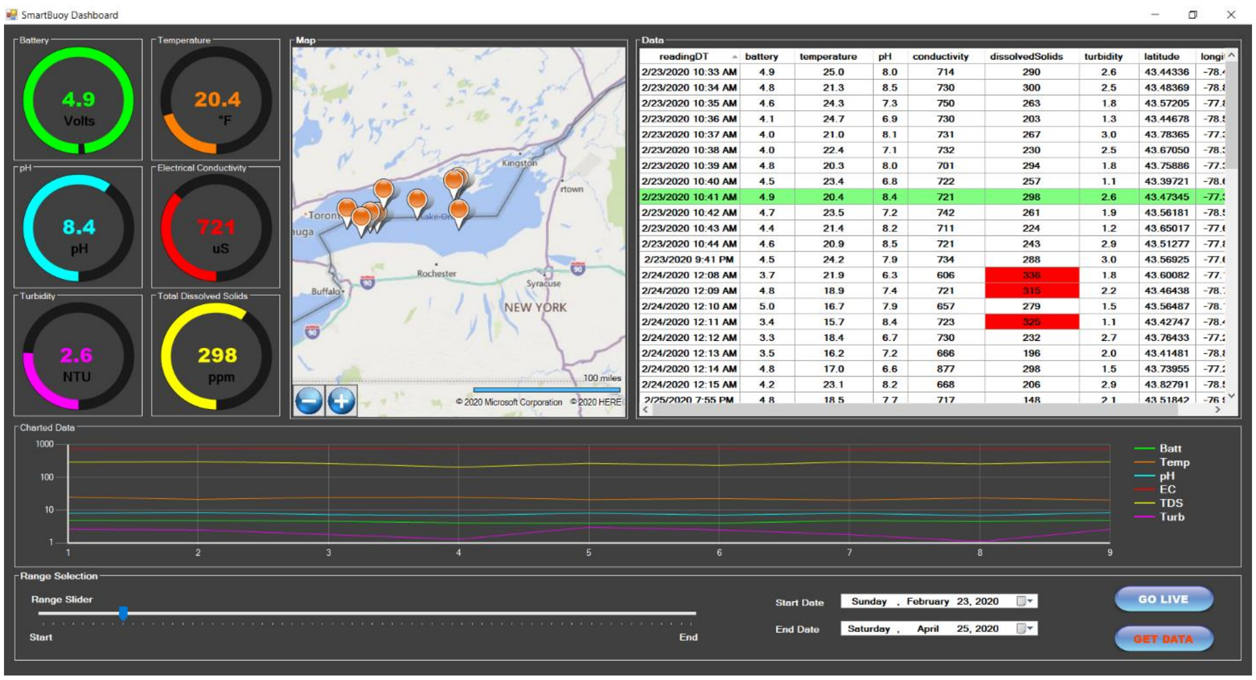
AFTER SETTING START AND END DATES AND CLICKING THE GET DATA BUTTON

This image shows the state of the application after the data is retrieved from the database.

Notice the main features now contain data.

The cells of the table highlighted in red indicated a value outside an environmentally acceptable range.

SCENARIOS DASHBOARD



readingDT	battery	temperature	pH	conductivity	dissolvedSolids	turbidity	latitude	longi
2/23/2020 10:33 AM	4.9	25.0	8.0	714	290	2.6	43.44336	-78.4
2/23/2020 10:34 AM	4.8	21.3	8.5	730	300	2.5	43.48369	-78.8
2/23/2020 10:35 AM	4.6	24.3	7.3	750	263	1.8	43.57205	-77.8
2/23/2020 10:36 AM	4.1	24.7	6.9	730	203	1.3	43.44678	-78.5
2/23/2020 10:37 AM	4.0	21.0	8.1	731	267	3.0	43.78365	-77.3
2/23/2020 10:38 AM	4.0	22.4	7.1	732	230	2.5	43.67050	-78.3
2/23/2020 10:39 AM	4.8	20.3	8.0	701	294	1.8	43.75886	-77.3
2/23/2020 10:40 AM	4.5	23.4	6.8	722	257	1.1	43.39721	-78.6
2/23/2020 10:41 AM	4.9	20.4	8.4	721	298	2.6	43.47345	-77.3
2/23/2020 10:42 AM	4.7	23.5	7.2	742	261	1.9	43.56181	-78.5
2/23/2020 10:43 AM	4.4	21.4	8.2	711	224	1.2	43.65017	-77.6
2/23/2020 10:44 AM	4.6	20.9	8.5	721	243	2.9	43.51277	-77.8
2/23/2020 9:41 PM	4.5	24.2	7.9	734	288	3.0	43.56925	-77.6
2/24/2020 12:08 AM	3.7	21.9	6.3	606	336	1.8	43.60082	-77.7
2/24/2020 12:09 AM	4.8	18.9	7.4	721	315	2.2	43.46438	-78.7
2/24/2020 12:10 AM	5.0	16.7	7.9	657	279	1.5	43.56487	-78.7
2/24/2020 12:11 AM	3.4	15.7	8.4	723	325	1.1	43.42747	-78.4
2/24/2020 12:12 AM	3.3	18.4	6.7	730	232	2.7	43.76433	-77.2
2/24/2020 12:13 AM	3.5	16.2	7.2	666	196	2.0	43.41481	-78.8
2/24/2020 12:14 AM	4.8	17.0	6.6	877	298	1.5	43.73955	-77.2
2/24/2020 12:15 AM	4.2	23.1	8.2	668	206	2.9	43.82791	-78.5
2/25/2020 7:55 PM	4.8	18.5	7.7	717	148	2.1	43.51842	-76.9

Start Date	Sunday , February 23, 2020
End Date	Saturday , April 25, 2020

AFTER MOVING THE RANGE SLIDER TO THE LEFT

This image shows the state of the application after the range slider is moved to the left.

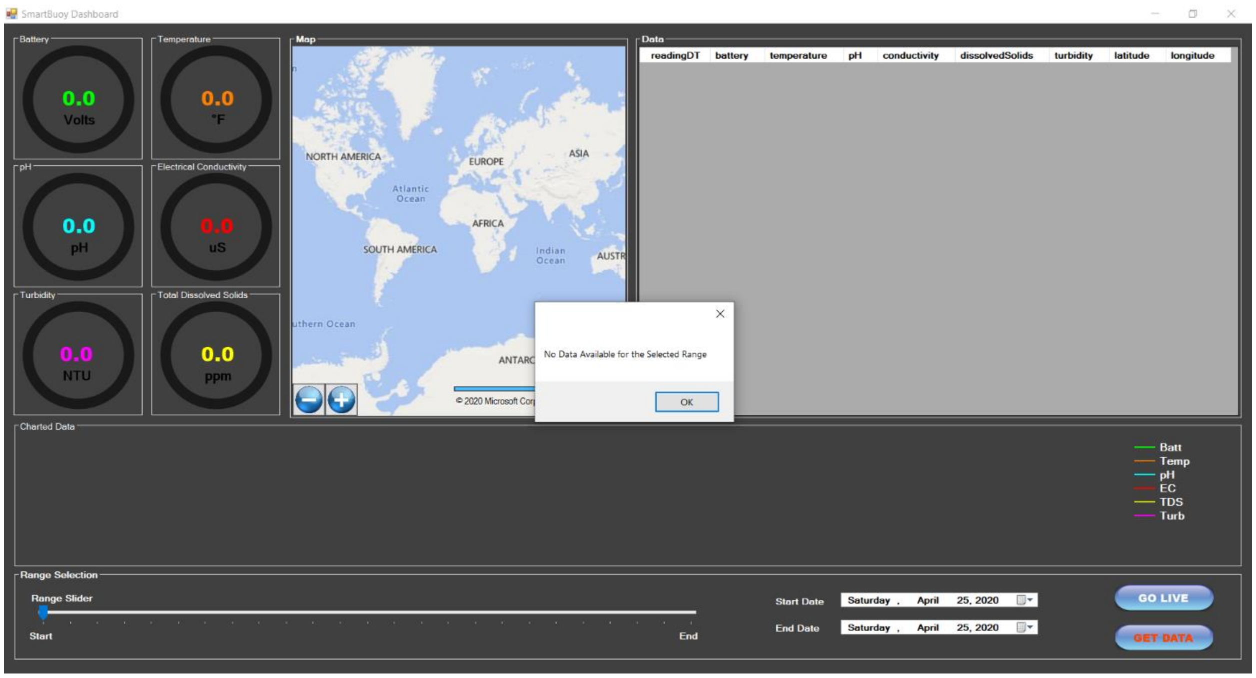
Notice there are fewer points on the map and line chart.

The range slider moves through the rows of the data table.

The values in the highlighted row appear on the gauges.

The map and chart show data from the rows up to the selected row.

SCENARIOS DASHBOARD



readingDT	battery	temperature	pH	conductivity	dissolvedSolids	turbidity	latitude	longitude

No Data Available for the Selected Range

OK

Start Date: Saturday , April 25, 2020

End Date: Saturday , April 25, 2020

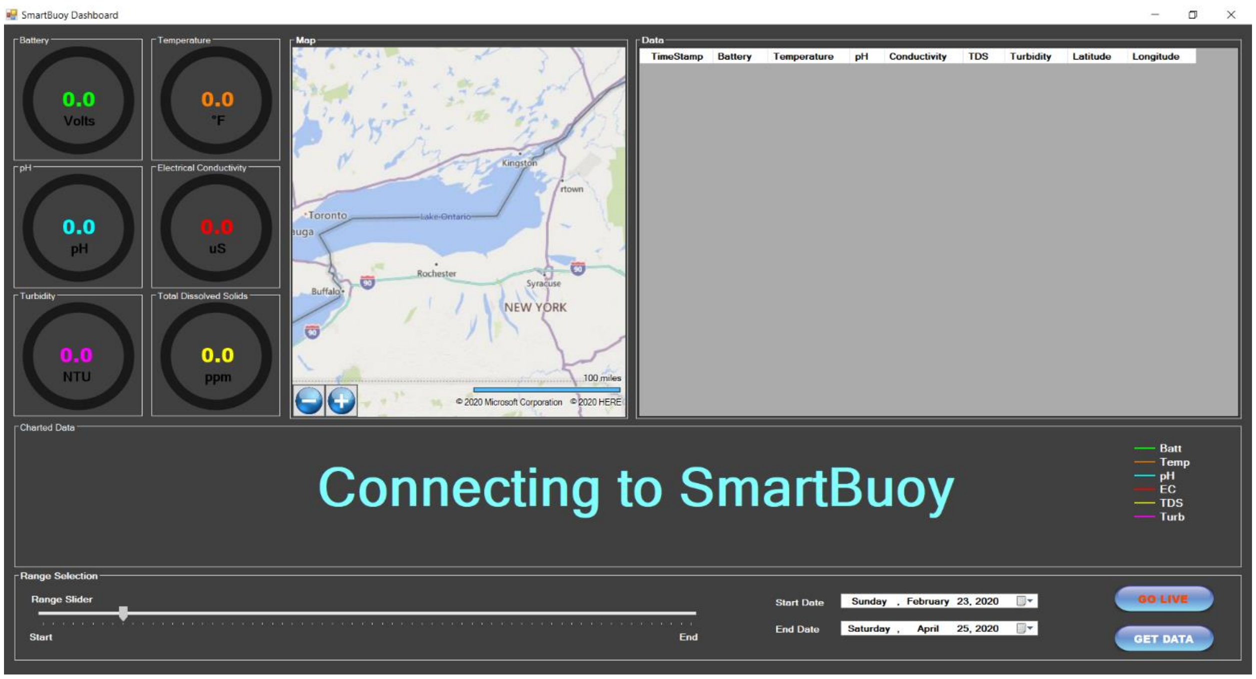
ERROR MESSAGE IF NO DATA FALLS WITHIN THE DATE RANGE

This image shows the result of selecting a date range with no data.

All components are reset

A Message Box pops up, alerting the user of the error

SCENARIOS DASHBOARD



Connecting to SmartBuoy

Start Date: Sunday , February 23, 2020

End Date: Saturday , April 25, 2020

AFTER CLICKING THE GO LIVE BUTTON

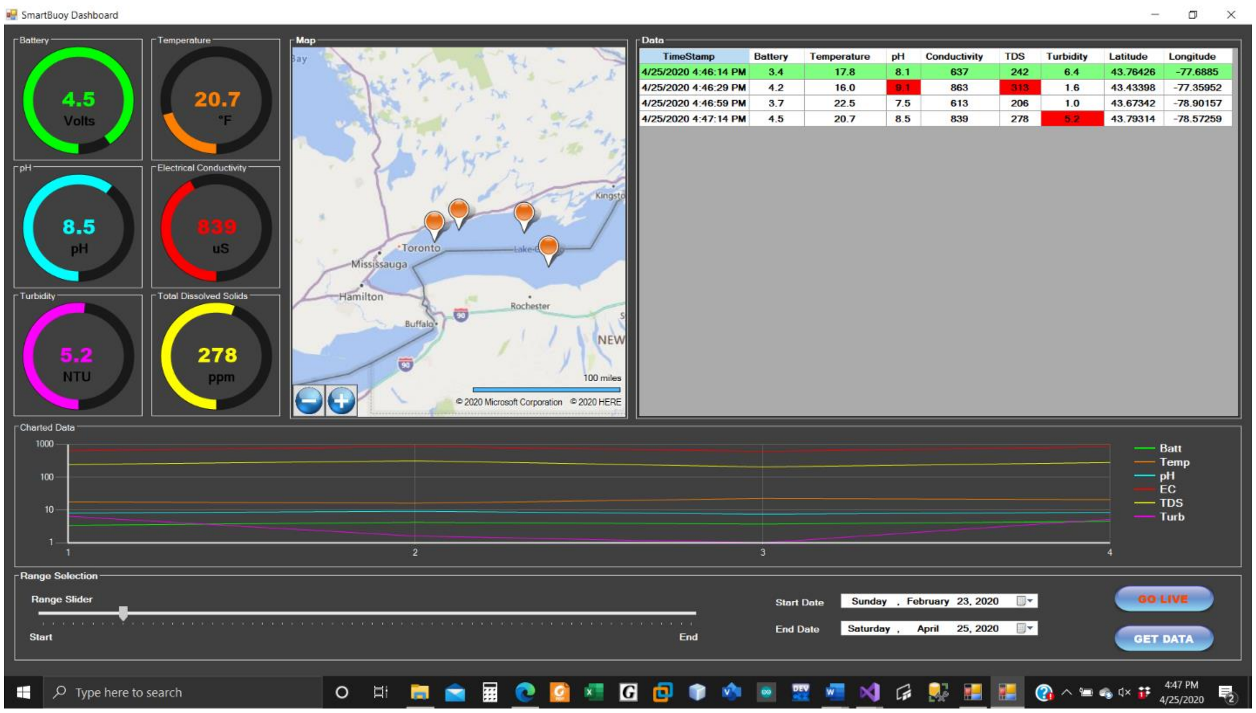
This image shows the state of the dashboard after clicking the GO LIVE button.

All components are reset

A message is displayed, informing the user that the dashboard is connecting to the SmartBuoy

In this case the data is coming from the simulator

SCENARIOS DASHBOARD



Data

TimeStamp	Battery	Temperature	pH	Conductivity	TDS	Turbidity	Latitude	Longitude
4/25/2020 4:46:14 PM	3.4	17.8	8.1	637	242	6.4	43.76426	-77.6885
4/25/2020 4:46:29 PM	4.2	16.0	9.1	863	313	1.6	43.43398	-77.35952
4/25/2020 4:46:59 PM	3.7	22.5	7.5	613	206	1.0	43.67342	-78.90157
4/25/2020 4:47:14 PM	4.5	20.7	8.5	839	278	5.2	43.79314	-78.57259

AFTER LIVE DATA IS RETREIVED

This image shows the dashboard after retrieving four readings from Dweet.io

A reading is retrieved every 15 seconds.

The reading is added to the data table, line chart, and map.

The gauges show the values of the most recent reading.

The connection remains until GET DATA is clicked or the application is closed.

DWEET.IO

Dweet.io is simple publishing and subscribing for machines, sensors, devices, robots, and gadgets. It is essentially Twitter for machines. A device will publish data to dweet.io, where it may be retrieved by those subscribed to it. The five most recent dweets are stored for up to 24 hours, although private accounts may be purchased, which store data up to 30 days. Free accounts require no set up. Simply name a device and begin dweeting. Dweets can contain multiple key/value pairs with a maximum of 2000 characters.

What makes dweet.io great is its simplicity. A device may publish data through a simple request.

For example: **https://dweet.io/dweet/for/SmartBuoy?hello=world&status=online**

Retrieve a message by simply requesting: **https://dweet.io/get/latest/dweet/for/SmartBuoy**

Which would return a JSON string containing the key/value pairs

EXECUTABLES

DASHBOARD SmartBuoyGUI.exe

Launches the dashboard

SIMULATOR SmartBuoySimulator.exe

Launches the simulator

PROGRAMMING LANGUAGE

C#.NET

This project is a Windows Form Application written using C#.NET. C# is a general object-oriented programming language. It is based on the .NET framework, containing libraries, including Language Integrated Query (LINQ), that are well suited for networking and web development.

Windows Forms

Windows Forms is a set of managed libraries in .NET Framework designed to develop rich client applications. It is a graphical API to display data and manage user interactions with event-driven architecture, meaning its applications wait for user input for its execution.

LINQ

LINQ is a Microsoft .NET Framework component that adds native data querying capabilities to .NET languages. LINQ provides the ability to interact with data stores using query expressions similar to SQL queries. LINQ is compatible with arrays, enumerable classes, XML documents, relational databases, and third-party data sources.

System.Net

The namespace provides a simple programming interface for many of the protocols used on networks today. The classes form the basis of what are called pluggable protocols, an implementation of network services that enables you to develop applications that use Internet resources without worrying about the specific details of the individual protocols.

WebResponse

Part of the System.Net namespace. Provides a response from a Uniform Resource Identifier (URI).

HttpClient

Part of the System.Net namespace. Provides a base class for sending HTTP requests and receiving HTTP responses from a resource identified by a URI

DATABASE DESIGN

Data will be stored in an MSSQL database created using Microsoft Azure.

The database only requires a single table. Each of the fields holds a measurement from the SmartBuoy and/or simulator.

Reading :	Table
readingDT :	field (datetime, null)
battery :	field (decimal(2,1), null)
temperature :	field (decimal(4,1), null)
pH :	field (decimal(2,1), null)
conductivity :	field (decimal(4,0), null)
dissolvedSolids :	field (decimal(3,0), null)
turbidity :	field (decimal(2,1), null)
longitude :	field (decimal(7,5), null)
latitude :	field (decimal(7,5), null)

SIMULATOR DESIGN Simulator

Simulator :	Implements the SimulatedReading class and creates the GUI
reading :	SimulatedReading
PowerIsOn :	bool flag to indicate the state of the simulator
btnPowerOn :	Button to start the ReadingTimer – changes the image of indicatorPower from LED_RedOff to LED_RedOn
btnPowerOff :	Button to stop the ReadingTimer – changes the image of indicatorPower from LED_RedOn to LED_RedOff
indicatorPower :	PictureBox contains an image of an LED to show the state of the simulator
indicatorSQL :	PictureBox contains an image of an LED to show activity of the SQL connection
indicatorLIVE :	PictureBox contains an image of an LED to show activity of the HTTP connection
indicatorBATT :	PictureBox contains an image of an LED to show activity of a battery reading
indicatorTEMP :	PictureBox contains an image of an LED to show activity of a temperature reading
indicatorTDS :	PictureBox contains an image of an LED to show activity of a TDS reading
indicatorEC :	PictureBox contains an image of an LED to show activity of an EC reading
indicatorPH :	PictureBox contains an image of an LED to show activity of a pH reading
indicatorGPS :	PictureBox contains an image of an LED to show activity of a longitude and latitude reading
ReadingTimer :	Timer creates a SimulatedReading, calls CycleLEDs(), uses LINQ to insert into database, calls DweetStream.BroadcastLive() to publish to dweet.io
FlashLED(PictureBox) :	Toggles the image of PictureBox between a dim and illuminated green LED
CycleLEDs() :	Repeatedly calls the FlashLED(PictureBox) method passing indicatorSQL, indicatorLIVE, indicatorBATT, indicatorTEMP, indicatorTDS, indicatorEC, indicatorPH, and indicatorGPS

SIMULATOR DESIGN SimulatedReading

SimulatedReading :	Class used to simulate and post readings
readingDT :	string variable for the current date and time
battery :	decimal variable for voltage
pH :	decimal variable for pH
conductivity :	decimal variable for electrical conductivity
temperature :	decimal variable for temperature
dissolvedSolids :	decimal variable for total dissolved solids
turbidity :	decimal variable for turbidity
longitude :	decimal variable for longitude
latitude :	decimal variable for latitude
GetReading() :	void method Generates random numbers and assigns them to battery, pH, conductivity, temperature, dissolvedSolids, turbidity, longitude, and latitude. Assigns the current DateTime to readingDT

SIMULATOR DESIGN SmartBuoyDB

SmartBuoyDB :	Inherits DataContext – An instance of class is used with LINQ to act as the connection to the database.
Login :	String variable contains the database connection string

SIMULATOR DESIGN Logger

Logger :	The Logger class contains methods used to write caught exceptions to a text file
LogError(Exception) :	Accepts an exception and writes it to a file.
PathToFile(string) :	Accepts the local path of a file in the solution directory and returns the full path as a string.

SIMULATOR DESIGN DweetStream

BroadcastLive(SimulatedReading) :	Builds a string using the SimulatedReading data members. That string becomes the uri for an HttpClient get request.

SIMULATOR DESIGN RangeValidator

RangeValidator :	Contains methods to verify the measurements are within the range of the probes. Any measurement outside of the acceptable range will raise an alert indicating the probe requires calibration.
isValidReading(SimulatedReading) :	Tests value of SimulatedReading.latitude, SimulatedReading.longitude, SimulatedReading.battery, SimulatedReading.pH, SimulatedReading.temperature, SimulatedReading.conductivity, SimulatedReading.turbidity, and SimulatedReading.dissolvedSolids by calling isVoltageInRange(decimal), isPhInRange(decimal), isTempInRange(decimal), isConductivityInRange(decimal), isTurbidityInRange(decimal), and isDissolvedSolidsInRange(decimal). Returns true if all called methods return true. or opens a MessageBox and returns false if any methods return false.
isPositionInRange(decimal, decimal) :	Tests value of SimulatedReading.latitude and SimulatedReading.longitude. Returns true if value is in range or opens a MessageBox and returns false if out of range.
isVoltageInRange(decimal) :	Tests value of SimulatedReading.battery. Returns true if value is in range or opens a MessageBox and returns false if out of range.
isPhInRange(decimal) :	Tests value of SimulatedReading.pH. Returns true if value is in range or opens a MessageBox and returns false if out of range.
isTempInRange(decimal) :	Tests value of SimulatedReading.temperature. Returns true if value is in range or opens a MessageBox and returns false if out of range.
isConductivityInRange(decimal) :	Tests value of SimulatedReading.conductivity. Returns true if value is in range or opens a MessageBox and returns false if out of range.
isTurbidityInRange(decimal) :	Tests value of SimulatedReading.turbidity. Returns true if value is in range or opens a MessageBox and returns false if out of range.
isDissolvedSolidsInRange(decimal) :	Tests value of SimulatedReading.dissolvedSolids. Returns true if value is in range or opens a MessageBox and returns false if out of range.

DASHBOARD DESIGN Reading

Reading :	Class used to retrieve and manipulate reading data
readingDT :	string variable for the current date and time
battery :	decimal variable for voltage
pH :	decimal variable for pH
conductivity :	decimal variable for electrical conductivity
temperature :	decimal variable for temperature
dissolvedSolids :	decimal variable for total dissolved solids
turbidity :	decimal variable for turbidity
longitude :	decimal variable for longitude
latitude :	decimal variable for latitude

DASHBOARD DESIGN RangeLimitAlert

RangeLimitAlert :	Contains methods to verify the measurements are within an environmentally acceptable range.
isReadingWithinLimit :	Tests values of all measurements. Returns true if all values are in range, returns false if out of range
isVoltageWithinLimit :	Tests value of battery. Returns true if value is in range, returns false if out of range
isPhWithinLimit :	Tests value of pH. Returns true if value is in range, returns false if out of range
isTempWithinLimit :	Tests value of temperature. Returns true if value is in range, returns false if out of range
isConductivityWithinLimit :	Tests value of conductivity. Returns true if value is in range, returns false if out of range
isTurbidityWithinLimit :	Tests value of turbidity. Returns true if value is in range, returns false if out of range
isDissolvedSolidsWithinLimit :	Tests value of dissolvedSolids. Returns true if value is in range, returns false if out of range

DASHBOARD DESIGN **ReadingProxy**

ReadingProxy :	Class for using data from deserialized JSON string
Rootobject :	Root class of JSON string
_this :	string – not used
by :	string – not used
the :	string – not used
with :	With[] – not used
With :	Subclass of the JSON string
thing :	string – not used
created :	DateTime – not used
content :	Content – not used
Content :	Subclass of the JSON string – contains data
DATETIME :	string variable for the current date and time
VOLTS :	double variable for voltage
TEMP :	double variable for temperature
PH :	double variable for pH
EC :	double variable for electrical conductivity
TDS :	double variable for total dissolved solids
TURB :	double variable for turbidity
LAT :	double variable for latitude
LON :	double variable for longitude
WebResponse() :	method retrieves the most recent Dweet.io post. Returns the data as a Reading object

DASHBOARD DESIGN Dashboard

Dashboard :	Implements the Reading class, ReadingProxy class, and creates the GUI
dtStart :	DateTimePickers used to set the start date for the data to be retrieved from the database.
dtEnd :	DateTimePickers used to set the end date for the data to be retrieved from the database.
btnHistoric :	Button that initiates the retrieval sequence. A connection is made to the database and a command is sent to retrieve the rows with dates in the specified range.
dataHistoric :	A DataGridView that displays the retrieved data. Each row of the grid is a row from the database, which represents one transmission from the SmartBuoy.
lineChart :	Links to dataHistory as its data source to graph all of the measurements.
rangeSlider :	A slider whose number of ticks is set to equal the number of rows retrieved from the database. The value of the slider's position corresponds to a row from dataHistoric. That value is used to select a row number, then each cell of that row is mapped to it is the appropriate gauge for display.
BuoyMap :	Uses the latitude and longitude columns of dataHistoric to plot points representing the location of the device when each reading was taken. Points are displayed beginning with the first row of dataHistoric up to the row selected by rangeSlider.
btnLive :	A button opens a connection to Dweet.io. Dweet acts as an intermediate between the SmartBuoy and the GUI. Dweet receives the data from the SmartBuoy and stores the five most recent transmissions. The dashboard uses an HTTP request to get the most recent reading.
gaugeBatt :	A custom control to view the battery voltage. The Value sets the level of the gauge. The Text displays the value as text.
gaugeTemp :	A custom control to view the temperature. The Value sets the level of the gauge. The Text displays the value as text.
gaugeEC :	A custom control to view the electrical conductivity. The Value sets the level of the gauge. The Text displays the value as text.
gaugeTDS :	A custom control to view the total dissolved solids. The Value sets the level of the gauge. The Text displays the value as text.
gaugeTurb :	A custom control to view the turbidity. The Value sets the level of the gauge. The Text displays the value as text.

DASHBOARD DESIGN Dashboard

gaugePH :	A custom control to view the pH. The Value sets the level of the gauge. The Text displays the value as text.
LiveTimer :	Timer calls methods to retrieve the live data. The interval is set to 30 seconds
btnZoomIn :	A button to zoom in on the map
btnZoomOut :	A button to zoom out on the map
MapHost	A control to host the WPF map control
btnHistoric_Click() :	Retreives data from the database. The user specifies the start and end dates. If the end date is before the start date an error is thrown. The data retrieved is shown in the dataGridview, gauges, and chart latitudes and longitudes are plotted as points on the map.
btnLive_ClickAsync() :	Enables LiveTimer and clears the map, gauges, chart, grid
LiveTimer_Tick() :	Retrieves the latest reading from dweet.io. The data is added to the chart, map, gauges, and a new row in the dataGridview. Comparing DateTime values prevents duplicate rows from being added
rangeSlider_Scroll() :	Increments or decrements rangeSlider.Value, which corresponds to a row from the dataGridView. Cell values of that row are used to set gauges and become the last values added to the chart and map
btnZoomIn_Click() :	Increments the value of BuoyMap.Map.ZoomLevel
btnZoomOut_Click() :	Decrements the value of BuoyMap.Map.ZoomLevel
dtStart_ValueChanged() :	Sets dtEnd.Min to the Value of dtStart when dtStart.Value changes
dtEnd_ValueChanged() :	Sets the Max date of dtStart to the Value of dtEnd when dtEnd.Value changes
setGauges() :	Sets the gauge Value and gauge Text to the values of the Reading object data members
clearGauges() :	Adds a pin to the map
setChart() :	Accepts an int representing the maximum dataGridview row number creates the chart from the dataGridview rows
rowToReading() :	Accepts an integer representing a row number. Assigns the values of the cells in a dataGridView row to the Reading object data members
readingToRow() :	Accepts an integer representing a row number. Assigns the value of the Reading object data members to cells of the row in the dataGridView.
checkLimits() :	Tests all grid cell values in a given row against a range of acceptable values. Any cell out of range is highlighted in red

DASHBOARD DESIGN **SmartBuoyDB**

SmartBuoyDB :	Inherits DataContext – An instance of class is used with LINQ to act as the connection to the database.
Login :	String variable contains the database connection string

DASHBOARD DESIGN **Logger**

Logger :	The Logger class contains methods used to write caught exceptions to a text file
LogError(Exception) :	Accepts an exception and writes it to a file.
PathToFile(string) :	Accepts the local path of a file in the solution directory and returns the full path as a string.

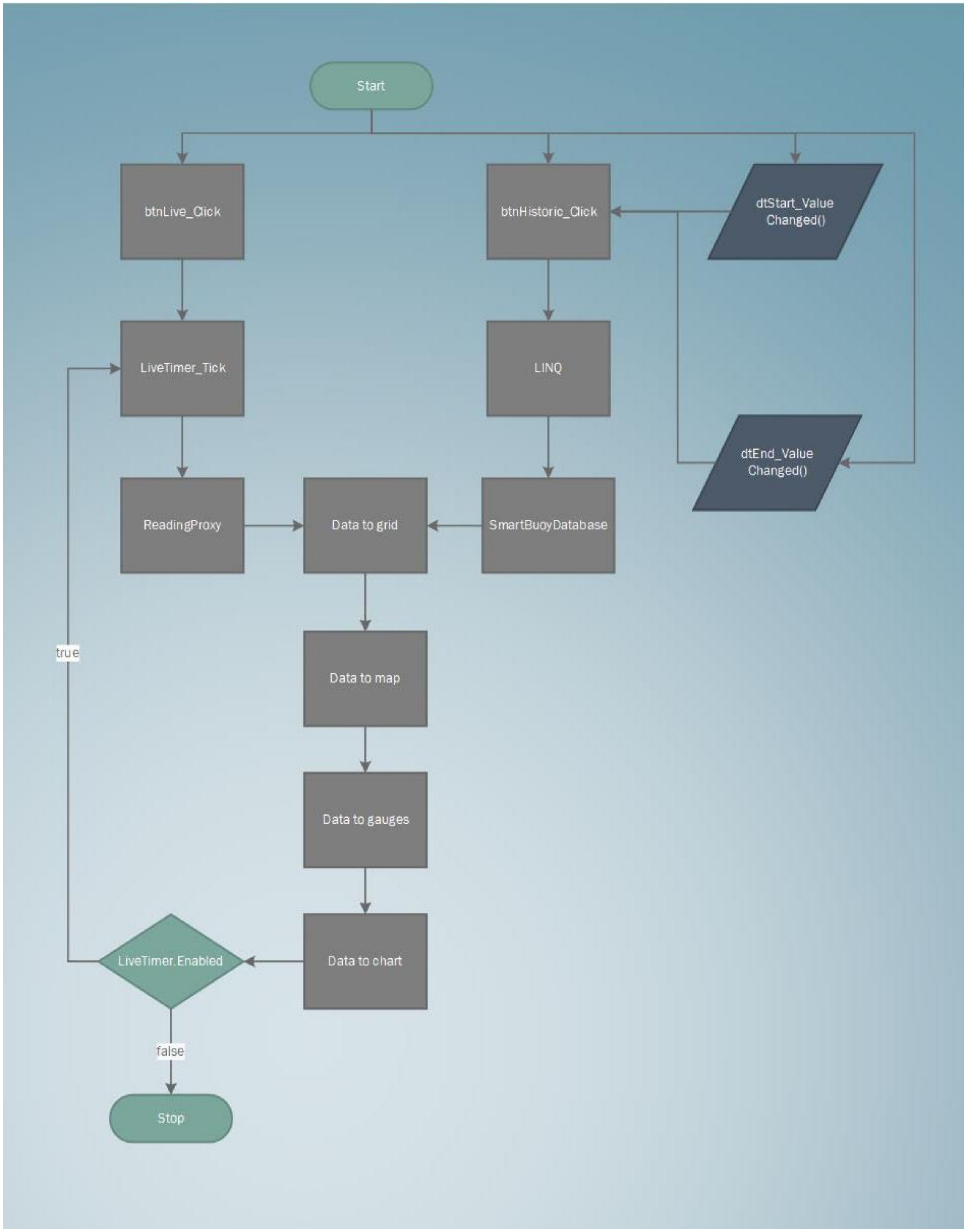
DASHBOARD DESIGN **Microsoft.Maps.MapControl.WPF**

Microsoft.Maps.MapControl.WPF :	A WPF control to implement a Bing Map

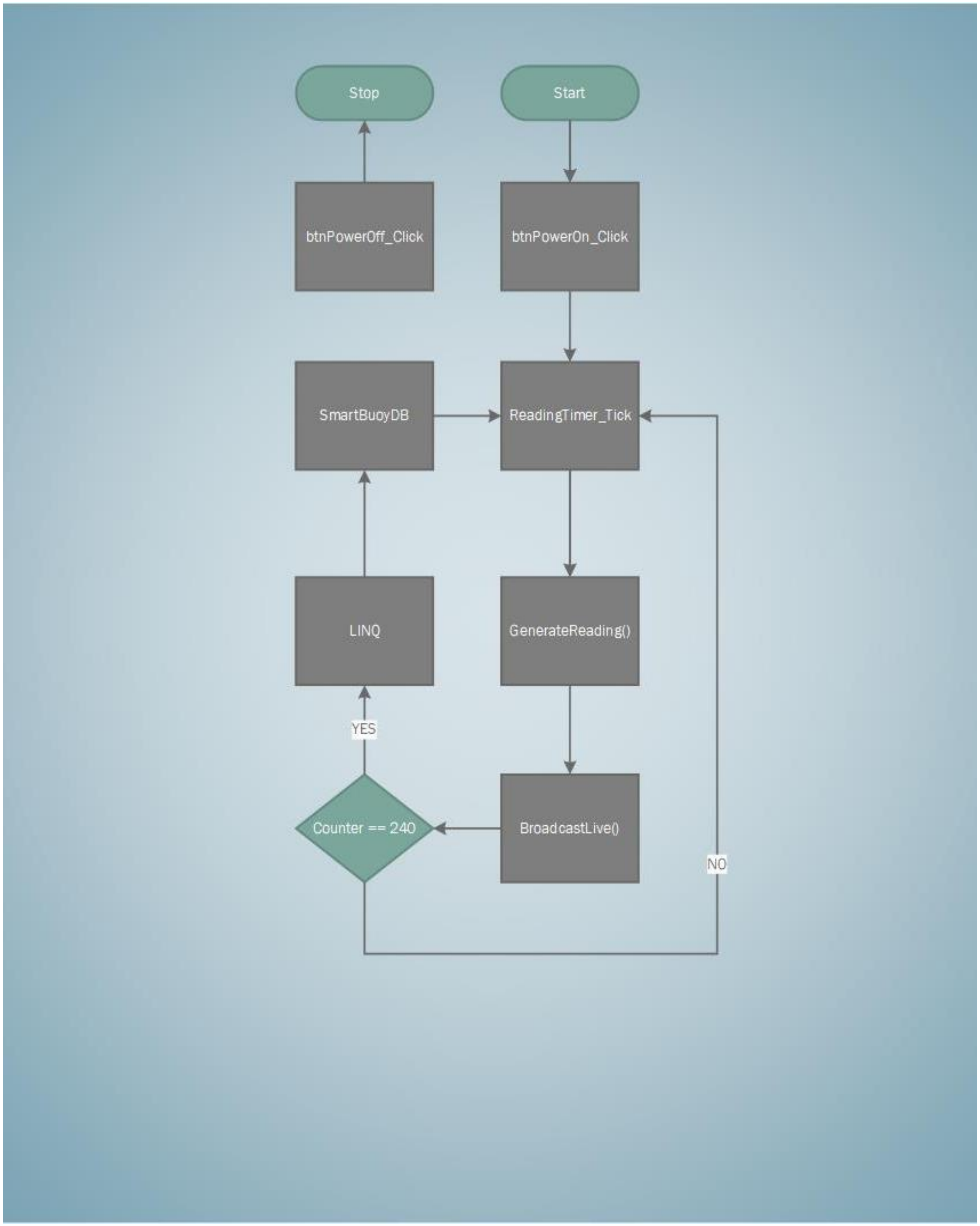
DASHBOARD DESIGN **CirclularProgressBar**

CircularProgressBar :	A custom control for WinForm with animation. It is used as a gauge in this project
Value :	Sets the arc length
Text :	sets the text in the center of the component

PROGRAM FLOW DIAGRAM DASHBOARD



PROGRAM FLOW DIAGRAM SIMULATOR



SUMMARY

Thank you for reviewing the SmartBuoy. Hopefully, this document has achieved its intended purpose and given you a thorough understanding of the SmartBuoy Dashboard and Simulator. If you are interested in viewing the project, it can be found on GitHub.

APPENDIX B Build and Release Process

The application is currently available for download and installation from GitHub.

The project will implement continuous delivery (CD) to improve the application. Development will continue in short cycles, fixing bugs as they are found and adding features to improve usability. This is an iterative approach, with each iteration delivering a reliable product.

Updates will be manually deployed on GitHub. Users should expect a feature update every 6 months, with periodic bug fixes deployed as required between updates.

APPENDIX C Client Installation Instructions

Step 1:	Download the repository to your Windows machine
Step 2:	Extract the compressed files
Step 3:	Navigate to the folder SmartBuoy Simulator Installer
Step 4:	Run the setup file
Step 5:	Follow Windows prompts. Disregard Windows security warnings
Step 6:	Repeat steps 3, 4, and 5, instead navigating to the folder SmartBuoy Dashboard Installer
Step 7:	The Simulator and Dashboard are now available in the start menu

APPENDIX D Developer Setup Instructions

This application was written using C#.NET. To continue development, you will need an IDE that supports the .NET Framework. The obvious choice is Microsoft Visual Studio – which can be downloaded here. Additionally, there are alternative compatible IDEs like Project Rider

If installing Visual Studio, be sure to select ".Net desktop development" when presented with a choice of workloads.

After you have chosen your IDE, download the repository to your machine, extract the compressed files, and open the solution