DESIGN SPECIFICATION

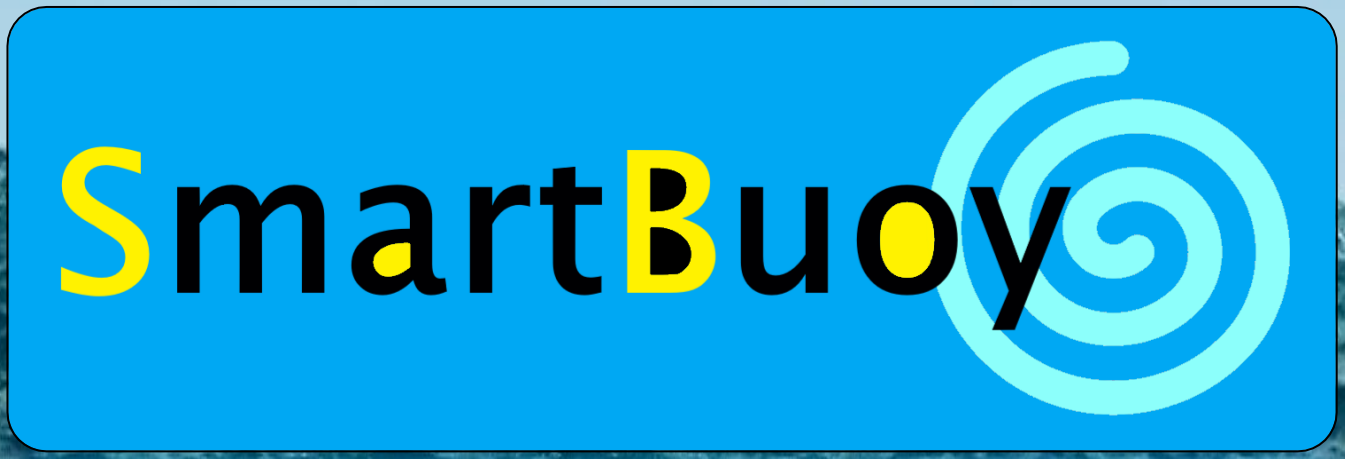

TABLE OF CONTENTS

ABSTRACT

Background: SmartBuoy is designed to be a fully self-contained water quality probe, able to be deployed into a body of water and monitored remotely via a GSM cellular network. SmartBuoy is an economical solution to water 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 30 seconds and streamed live. Every hour a reading is sent to a database for storage.

Project Scope: The project for this course is a GUI dashboard to retrieve and view the data readings. Because this is an online course, access to the device is unavailable to the class and professor. To mitigate this obstacle, I have created a second project to simulate the operations of the device. In addition to the dashboard and simulator, the database will also be created for data storage.

Current State: As of 2/28/2020

1. The simulator is fully functional. Clicking the power button begins a loop that creates artificial readings, posts them to Dweet.io and the SQL database. Every 30 seconds an artificial reading is created using random numbers. This data is sent to Dweet.io using HttpClient. Every hour a reading is sent to the SQL database.

   Dweet.io is a service for sharing IoT data.

2. The database has been created using Microsoft Azure, with a single table added. Two stored procedures have been created. One to insert data and the other to retrieve it.

3. The dashboard is partially functional. Two datetime pickers specify the date range of reading to be retrieved from the database. Clicking the GET DATA button creates a SqlConnection and retrieves all rows within the specified range. This data is displayed in a DataGrid and plotted on a line chart. The location of each reading is plotted onto a map. When a row is selected, the data of each cell is displayed by custom gauge components. Clicking the GO LIVE gets the data from Dweet.io. At this time the connection is functional. The data is retrieved and sent to the corresponding gauges

Operation:

The database has been populated using the simulator. The dashboard can retrieve that data without the simulator running. To use the live data function the simulator must be running

To Do List:

1. Live data to the map, chart, and grid
2. Clean up code / Code reuse
3. Complete commenting and documentation
4. Complete error handling
5. Encrypt connection string in config file

DEVICE

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

SIMULATOR : SimulatedReading class

SimulatedReading :	Class used to simulate and post readings
readingDT :	string variable for the current date and time
battery :	double variable for voltage
pH :	double variable for pH
conductivity :	double variable for electrical conductivity
temperature :	double variable for temperature
dissolvedSolids :	double variable for total dissolved solids
turbidity :	double variable for turbidity
longitude :	double variable for longitude
latitude :	double 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
BroadcastLive() :	Async Task method creates a string from the values of readingDT, battery, pH, conductivity, temperature, dissolvedSolids, turbidity, longitude, and latitude. Then uses HttpClient to post the values to Dweet.io
SendToDatabase() :	void method creates a SqlConnection and sends the values of readingDT, battery, pH, conductivity, temperature, dissolvedSolids, turbidity, longitude, and latitude to the database in a SqlCommand containing the PostReadings stored procedure

SIMULATOR : Resources

Resources :	Directory
LED_RedOn :	Image of red LED on
LED_RedOff :	Image of red LED off
LED_GreenOn :	Image of green LED on
LED_GreenOff :	Image of green LED off

SIMULATOR : Simulator class

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 calls the FlashLEDs() method and the GetReading(), BroadcastLive(), ToDatabase() methods of the SimulatedReading class
FlashLEDs() :	void method to consecutively change the image of indicatorSQL, indicatorLIVE, indicatorBATT, indicatorTEMP, indicatorTDS, indicatorEC, indicatorPH, and indicatorGPS PictureBoxes from LED_GreenOff to LED_GreenOn. Sleep for 200ms, then change back to LED_GreenOff.

DATABASE

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)
GetHistoricReadings :	Stored Procedure to select rows from Reading where the value of readingDT is greater than @START and less than @END
PostReadings :	Stored Procedure to insert the values @readingDT, @battery, @temperature, @pH, @conductivity, @dissolvedSolids, @turbidity, @latitude, @longitude into the fields readingDT, battery, temperature, pH, conductivity, dissolvedSolids, turbidity, latitude, longitude in the Reading table

DASHBOARD : Reading class

Reading :	Class used to retrieve and manipulate reading 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

DASHBOARD : ReadingProxy class

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 : Dashboard class

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's 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 : Dashboard class

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

IMAGES

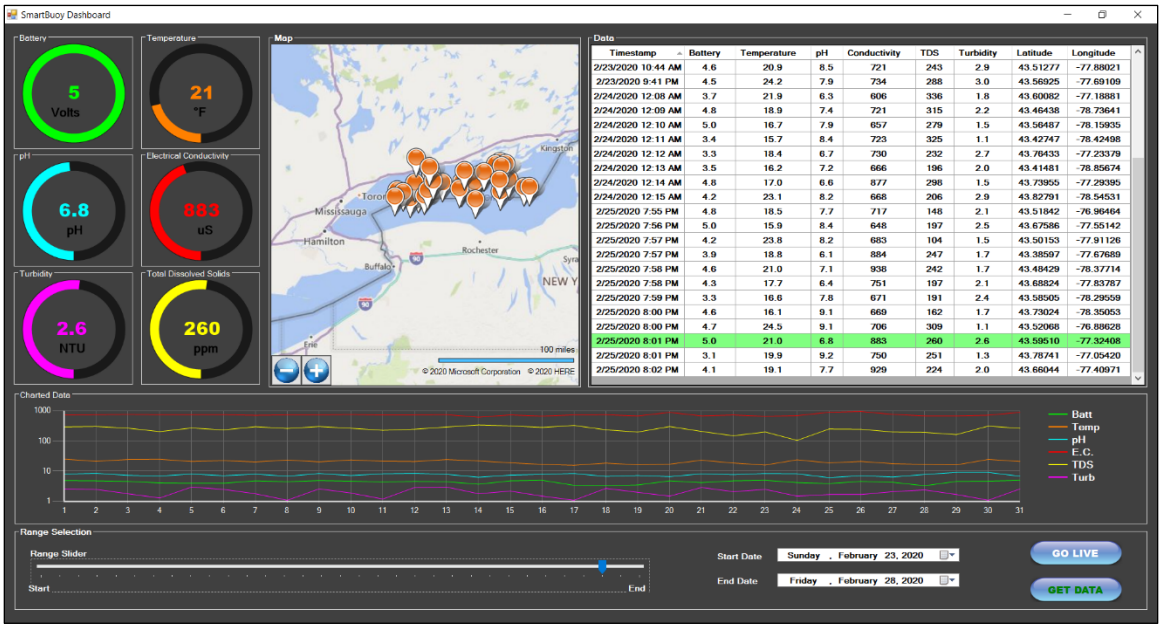

Image 1.1 : Dashboard

Image 1.2 : Simulator