AquaLogic: Simulated IoT Edge Systems for Diver Safety and Marine Observation

Phillip Benard, Dr. Aravind Mohan
Department of Computer Science, McMurry University, Abilene, TX

Abstract

AquaLogic is a simulated IoT edge framework for underwater environments that emphasizes diver safety and marine life observation. Using virtual multi-modal sensors and the command-line logic simulator DrMoss, the system models real-time edge decision-making, including predictive alerts for diver fatigue, oxygen levels, and environmental hazards, alongside the automated collection of environmental and species data for marine research. All sensor and environmental data are represented in structured JSON format, enabling efficient parsing, simulation, and extensibility. The implementation leverages the Java and object-oriented programming principles to define reusable simulated sensor, diver, and environmental modules, while GraphViz is used to visualize logic circuits and Boolean decision pathways. AquaLogic was developed, executed, and benchmarked against other logic simulators such as Logisim and Logicly, demonstrating improved flexibility in modeling complex underwater scenarios. Furthermore, DrMoss provides a modular environment where custom Boolean logic and sensor interactions can be tested before deployment in physical systems. This platform also highlights the benefits of logic-based simulation for training and educational purposes, enabling researchers to evaluate system performance and safety protocols without incurring field costs or risks.

Introduction

Internet of Things (IoT) and edge computing technologies enable real-time collection and processing of data close to its source, reducing latency and improving responsiveness in complex environments. AquaLogic is a simulated IoT edge system designed for underwater operations that integrates virtual sensors representing diver health, environmental conditions, and marine life presence. All data inputs and outputs are maintained in JSON format, facilitating structured simulation and automated analysis. The system is implemented in Java using object-oriented programming principles within the DrMoss command-line simulator, enabling modular and reusable code for diver, sensor, and environment objects. Boolean logic circuits process the data to generate real-time alerts for safety, guidance for navigation, and automated instructions for data recording, with circuit layouts visualized using GraphViz for clarity and verification. Organizations such as NOAA and the Scripps Institution of Oceanography highlight the importance of simulated underwater IoT frameworks for training, research, and environmental monitoring, making platforms like AquaLogic particularly relevant. DrMoss allows researchers to define custom logic interactions and quickly test new IoT edge workflows, offering a controlled environment for experimentation that complements physical field tests. Comparisons with other logic simulators, such as Logisim and Logicly, indicate that DrMoss and AquaLogic provide greater flexibility in modeling multi-sensor, multi-agent underwater scenarios. This simulation enables researchers, educators, and students to explore decision-making strategies, optimize edge computation algorithms, and prepare for real-world deployments in underwater IoT systems, without requiring access to expensive or potentially risky physical hardware.

Methodology

The goal of this research project was to get a deeper understanding of logic design in computer science and to apply that understanding to a real world application where we simulate data related to diving in deep water, exploring marine life and apply logic design on the simulated data, in order to get a better understanding of aquatic life. The purpose of Workflow 1 is to predict diver fatigue and detect environmental hazards to alert the diver in real-time as seen in Tables 4, 5, and 6. For example; it will check for a strong ocean current, monitor the diver's level of fatigue, visibility within the water, oxygen levels of the tank, and whether or not the manual over-ride has been initiated. If a strong current is detected or if there is low visibility and if the diver's fatigue level is not high, then the system is programmed to alert the diver to reduce their swimming pace. This is just one of many possible programmed results from input that the system is designed to handle. DrMoss is a logic simulator developed using Java, JSON, and GraphViz technology, Aqualogic is an application developed using the DrMoss core engine. In Table 2 , we demonstrated an example Aqualogic Workflow (W1) with five inputs and four outputs, the workflow internally uses ten logical operators including AND, NOT and OR. For example, inputs F or O and not M, are combined to create output A, which is used to alert the diver to rest and/or ascend. Tables 4 and 5, provide sample input and output with descriptions to explain their purpose.

In Table 1, we have an example of the input and output mapping for the workflow, a small truth table sample. Additionally we have the specification of the workflow W1 using the JSON format in Table 2. The specification is formulated based on the design pattern and criteria of the DrMoss platform. Furthermore, in Figure 1 we have shown the summary of the execution results where the logical circuit is first created using a series of programs built into DrMoss, it is visualized using GraphViz which take descriptions of graphs in a simple text language, and makes diagrams in useful formats, such as images and SVG for web pages; PDF or Postscript for inclusion in other documents; or display in an interactive graph browser. Finally we have a sample of DrMoss' terminal-interface, showing step-by-step how to run the program, utilizing the commands pre-programmed into the system, which can be viewed via the 'help' command for Table 3.

Table 1: Truth Table

C	F	V	O	M	A	R	S	I
0	0	0	0	0	0	0	0	1
0	1	0	1	0	1	0	1	0
1	0	0	1	0	1	1	0	0
1	1	1	0	0	1	0	0	0
1	1	1	1	1	0	0	0	0

Table 2: JSON Specs

```
{
 "inputs" : [ "C", "F", "V", "O", "M" ],
 "outputs" : [ "A", "R","S" ,"I" ],
 "gates" : [
  { "type" : "OR", "name" : "G1", "inputs" : [ "F", "O" ], "output" : "O1" },
  { "type" : "NOT", "name" : "G2", "inputs" : [ "M" ], "output" : "O2" },
  { "type" : "AND", "name" : "G3", "inputs" : [ "O1", "O2" ], "output" : "A" },
  { "type" : "OR", "name" : "G4", "inputs" : [ "C", "V" ], "output" : "O3" },
  { "type" : "NOT", "name" : "G5", "inputs" : [ "F" ], "output" : "O4" },
  { "type" : "AND", "name" : "G6", "inputs" : [ "O3", "O4" ], "output" : "R" },
  { "type" : "AND", "name" : "G7", "inputs" : [ "F", "O", "O2" ], "output" : "S" },
  { "type" : "NOT", "name" : "G8", "inputs" : [ "O" ], "output" : "O5" },
  { "type" : "NOT", "name" : "G9", "inputs" : [ "C" ], "output" : "O6" },
  { "type" : "AND", "name" : "G10", "inputs" : [ "O4", "O5", "O6" ], "output" : "I" }
}
```

Table 3: DrMoss Terminal Sample

```
Welcome to Dr. Mo's Logic System CLI! Type 'help' for commands.
SLF4J: No SLF4J providers were found.
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See https://www.slf4j.org/codes.html#noProviders for further details.
> help
Available commands:
  create          - Load and validate circuit.json and inputs.csv
  visualize       - Generate circuit diagram image (PNG)
  visualize -dark   - Generate circuit diagram image in darkmode (PNG)
  execute         - Run the circuit with inputs and print outputs
  execute -f single  - Run single-bit execution
  execute -f multi   - Run multi-bit execution
  execute -f seq     - Run sequential (clocked) execution
  save-subckt <name> - Save current circuit.json as a subcircuit module
  list-subckts      - List saved subcircuits
  exit            - Exit the CLI
> create
Circuit created successfully (10 gates, 15 nodes).
> visualize
Circuit image generated: circuit.png
Visualization complete -> circuit.png
> execute
Inputs: [0, 0, 0, 0, 0]   Inputs: [0, 0, 0, 0, 1]   Inputs: [0, 0, 1, 0, 1]
A = 0                     A = 0                     A = 0
R = 0                     R = 0                     R = 1
S = 0                     S = 0                     S = 0
I = 1                     I = 1                     I = 1
```

Table 4: Input Descriptions

VARIABLE	DESCRIPTION
C	Strong Current Detected
F	Fatigue Level High?
V	Visibility Low?
O	Oxygen Low?
M	Manual Override Enabled

Table 5: Output Descriptions

VARIABLE	DESCRIPTION
A	Alert Diver for Rest/Ascent
R	Reduce Swimming Pace
S	Stop Dive/Emergency Ascent
I	Ignore/Continue

Figure 1: Circuit Output

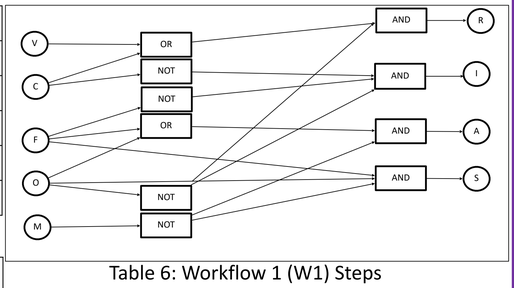


Table 6: Workflow 1 (W1) Steps

Step	Input	Output	Description
1	F OR O	01	Fatigue level is high or oxygen level is low.
2	NOT M	02	Manual over-ride is not enabled.
3	01 AND 02	A	Alert diver to rest and begin ascent to surface.
4	C OR V	03	Strong current detected or low visibility detected.
5	NOT F	04	Fatigue level is not high.
6	03 AND 04	R	Alert diver to reduce swimming pace.
7	F AND O AND 02	S	Alert diver to stop their dive and begin emergency ascent to surface.
8	NOT O	05	Oxygen level is not low.
9	NOT C	06	Strong current is not detected.
10	04 AND 05 AND 06	I	Ignore this alert, continue diving.

Experiments

On an average, Logisim had a faster design time then Logicly when the complexity of the circuit was minimal. However as the complexity of the circuit's design begins to increase around W3, the design time for Logisim increases. This happens due to the Logisim system being very sensitive to overlapping circuit wires and gates, which in turn necessitates additional redesigns. However, verification times for both Logisim and Logicly remain similar for all workflows. The design and verification times for DrMoss is faster compared to both Logicly and Logisim for all workflows that are both simple and complex in nature. As seen in our Figures 2 and 3, we illustrate that this is applicable for all of our workflows, going from W1 to W7. Additionally in Table 7, we show that our approach is accurate and reliable for all the workflows.

Table 7: List of Workflow Accuracy

List of Workflow Accuracy						
Name	Steps	Inputs	Outputs	Logisim	Logicly	DrMoss
W1	10	5	4	100%	100%	100%
W2	5	5	4	100%	100%	100%
W3	7	5	4	100%	100%	100%
W4	5	5	4	100%	100%	100%
W5	12	5	4	100%	100%	100%
W6	12	5	4	100%	100%	100%
W7	13	6	4	100%	100%	100%

Figure 2: Design Times

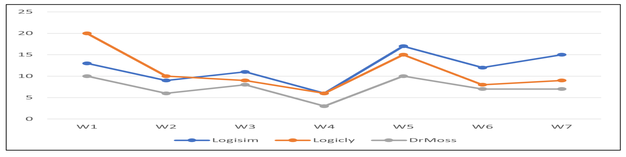
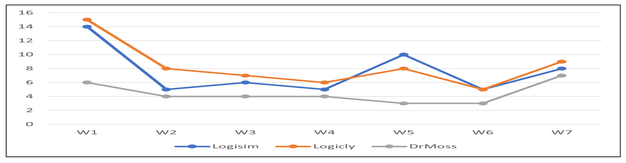


Figure 3: Verification Times



Conclusions and Future Work

The experimental evaluation demonstrates that AquaLogic, supported by the DrMoss simulator, achieves accurate simulation, robust performance in complex underwater scenarios, and efficient execution within a modular edge-computing framework. The system reliably models diver safety conditions, generates predictive alerts for fatigue, oxygen levels, and environmental hazards, and supports structured environmental data collection through flexible logic design. These results validate AquaLogic and DrMoss as an effective, simulation-driven IoT platform suitable for training, research, and safe evaluation of underwater edge systems. Future work includes improving predictive modeling and implementation of real-world data.

Contact Us

Phillip Benard Email:benard.phillip@mcm.edu

Dr. Aravind Mohan Email:mohan.aravind@mcm.edu
Phone: (325) 793 -3845