MEC 2016

Rogue QD

Ian Prins
Mikhail Andrenkov
Ori Almog

1 Project Overview

The RogueQD project aims to supply a simple, clear, and efficient way for naval entities to communicate between one another. By utilizing ITU bands 4 through 6, and covering a frequency domain of 3-3000 KHz, the RogueQD projects sends and receives messages between boats and naval weather buoys. Utilizing both satellite and VLF/LF/MF radio communication, a wide variety of signals can be sent. The RogueQD project also simulates error in message relay between naval entities.

Boats navigating the oceans will be able to use the RogueQD project to:

- Get accurate weather data.
- Send and respond to SOS distress calls.
- Determine its exact location via communication with GPS-equipped buoys.

2 Assumptions

1. Boats have AM radio transceivers that can transmit and listen along the 3-3000 KHz band.
2. Weather buoys in the sea are equipped with a transceiver at least as capable as the one the boats are equipped with.
3. Weather buoys are also equipped with a SCU (Satellite Communications Unit), enabling a GPS unit and centralized relay.
4. Along the selected frequency domain band, a signal sent should always be received correctly if sent from within 250 km.
5. Buoys are equipped with weather tracking sensors.

3 Technical Details

The RogueQD project is written in Java - a cross-platform and system-compatible programming language. Java is used heavily in computer systems around the world, and is arguably the world's most used programming language. Java can run on Windows, MacOS, Linux, and even *BSD systems!

The RogueQD project has taken on an OO (Object Oriented) approach, in order to simplify the architectural structure of the software system. This kind of methodology is particularly useful when modeling real-world entities, as the RogueQD project aims to do.

The project used a private repository on GitHub to keep track of the source code. The technology used here is Git - a world-renown system that sees much use in industry.

For compilation and testing, the Java 1.8.0_91 compiler was used, along with a custom makefile for automating compilation process.

4 Internal States