27 MAY 2024

SEAS 4.0

Co-funded by the Erasmus+ Programme of the European Union

UNIVERSIDADE DA CORUÑA

INDUSTRY 4.0 ENABLING TECHNOLOGIES

Group Project Smart Contract Integration Report

LECTURERS

- Paula Fraga Lamas
- Tiago M. Fernández Caramés

GROUP MEMBERS

- Mehmet Furkan SAPMAZ
- Ibrahim Hamid Saray ABDALLA
- Bruno OMLENYI

Contents

Enhancing Maritime Safety: A Smart Contract Approach for Ship Engine Room Monitoring

Abstract:

This report presents the integration and evaluation of a smart contract tailored for a Ship Engine Room Monitoring System . The smart contract, named "EngineRoomMonitor," is designed to bolster safety and operational efficiency in maritime environments by automating incident management and maintenance schedules based on sensor data. The report provides an in-depth analysis of the smart contract's structure, functionalities, and its potential implications for the maritime industry.

1. Introduction:

Efficient monitoring systems are paramount in ensuring the safety of personnel and assets onboard vessels. The Ship Engine Room Monitoring System addresses critical parameters such as temperature, smoke, flame detection, and light to mitigate potential hazards. Integrating a smart contract into this system streamlines incident management and maintenance coordination, marking a significant advancement in maritime safety technology.

2. Motivation:

The motivation behind integrating a smart contract into the Ship Engine Room Monitoring System stems from the necessity for seamless incident management and maintenance coordination. Conventional monitoring systems may lack automated processes for handling incidents detected by sensors. By leveraging a smart contract, the system can efficiently manage incidents in real-time and schedule maintenance or inspections promptly.

3. Smart Contract Overview:

The "EngineRoomMonitor" smart contract is developed using Solidity, a programming language for Ethereum smart contracts. It defines threshold values for temperature and smoke levels and incorporates functionalities for incident logging, maintenance requests, and inspection scheduling.

4. Structure and Functionality:

- Event Logging: The smart contract includes event logging functionalities to record incidents detected by sensors. Events such as "IncidentLogged," "MaintenanceRequested," and "InspectionScheduled" capture relevant data, including timestamps, sensor types, values, and messages.

- Threshold Values: Predefined threshold values for temperature and smoke and light levels are set within the contract. When sensor data exceeds these thresholds, the contract triggers incident logging and initiates the incident management process.

- Service Providers: The contract maintains a list of pre-approved service providers, including their names, addresses, and maintenance capabilities. Service providers offering maintenance services are notified when incidents occur, while others are scheduled for inspections.

5. Implementation and Deployment:

The "EngineRoomMonitor" smart contract is deployed on the Ethereum blockchain, providing decentralized incident management capabilities to the Ship Engine Room Monitoring System. Integration with the existing monitoring system enables seamless communication between sensors and the smart contract.

6. Evaluation:

- Efficiency: The smart contract enhances the efficiency of incident management by automating the process based on predefined thresholds.

- Transparency: By operating on a blockchain platform, the smart contract ensures transparency and immutability of incident records, fostering trust among stakeholders.

- Scalability: The contract's modular structure allows for scalability, enabling the addition of new sensors and service providers as the monitoring system expands.

7. system code

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract EngineRoomMonitor {
    // Define an event to log incidents
    event IncidentLogged(uint256 timestamp, string sensorType, uint256 value,
string message);
    event MaintenanceRequested(uint256 timestamp, string sensorType, uint256
value, address serviceProvider);
    event InspectionScheduled(uint256 timestamp, string sensorType, uint256
value, address serviceProvider);

    // Define threshold values
    uint256 constant TEMPERATURE_THRESHOLD = 50;
    uint256 constant SMOKE_THRESHOLD = 200;

    // Define structure to store service providers
    struct ServiceProvider {
        string name;
        address payable serviceAddress;
        bool providesMaintenance;
    }

    // List of pre-approved service providers
    ServiceProvider[] public serviceProviders;

    // Function to add a service provider
    function addServiceProvider(string memory name, address payable
serviceAddress, bool providesMaintenance) public {
        serviceProviders.push(ServiceProvider(name, serviceAddress,
providesMaintenance));
    }
```

```
    // Function to log sensor data and check for incidents
    function logSensorData(string memory sensorType, uint256 value) public {
        bool isIncident = false;

        if (keccak256(abi.encodePacked(sensorType)) == keccak256("temperature")
&& value > TEMPERATURE_THRESHOLD) {
            isIncident = true;
        } else if (keccak256(abi.encodePacked(sensorType)) == keccak256("smoke")
&& value > SMOKE_THRESHOLD) {
            isIncident = true;
        }

        if (isIncident) {
            // Log incident
            emit IncidentLogged(block.timestamp, sensorType, value, "Incident
detected");

            // Request maintenance or schedule inspection
            requestService(sensorType, value);
        }
    }

    // Function to request maintenance or schedule inspection
    function requestService(string memory sensorType, uint256 value) internal {
        for (uint256 i = 0; i < serviceProviders.length; i++) {
            if (serviceProviders[i].providesMaintenance) {
                // Request maintenance
                emit MaintenanceRequested(block.timestamp, sensorType, value,
serviceProviders[i].serviceAddress);
            } else {
                // Schedule inspection
                emit InspectionScheduled(block.timestamp, sensorType, value,
serviceProviders[i].serviceAddress);
            }
        }
    }
}
```

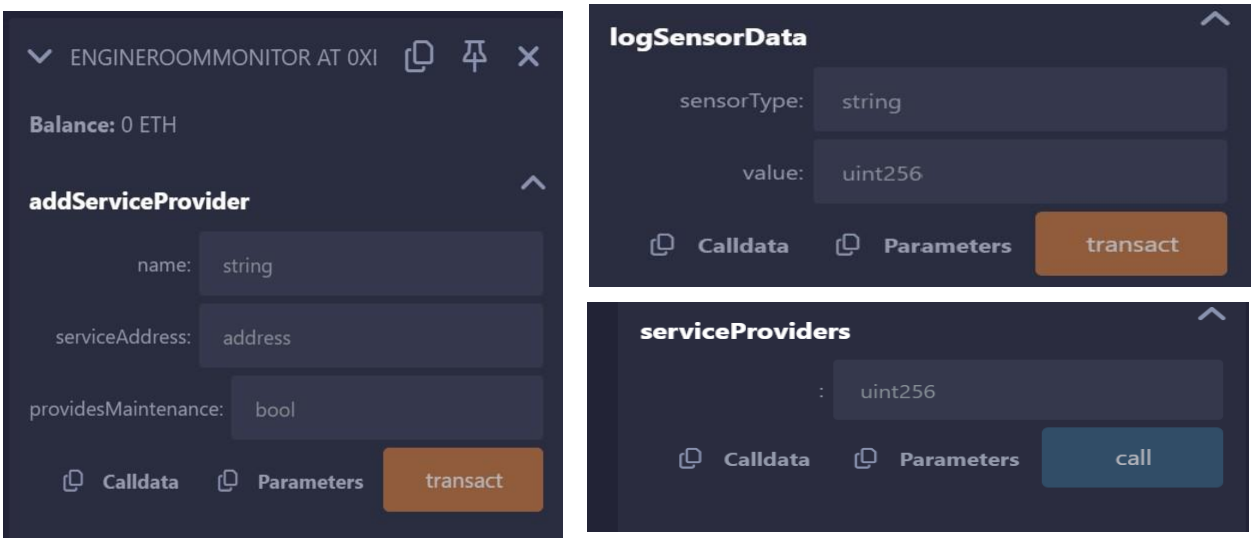

Figure 1 smart contract interaction window

8. Conclusion:

The integration of a smart contract into the Ship Engine Room Monitoring System represents a significant advancement in maritime safety technology. By automating incident management and maintenance scheduling, the smart contract enhances operational efficiency and reduces response times to potential hazards. Future research could explore additional functionalities and optimizations to further improve the system's capabilities.

References:

[1] Solidity Documentation. Retrieved from https://docs.soliditylang.org/

[2] Ethereum. Retrieved from https://ethereum.org/