UNIVERSIDADE DA CORUÑA

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

SEAS 4.0

# Industry 4.0 Enabling Technologies

## Information Systems and Data Management Module

Report for Smart Contract


Submitted To

Professor Paula Fraga Lamas

Professor Tiago M. Fernández Caramés

Submitted By

Abdoul-Malik Bakari

Kaung Myat Noe Aung

Musaab Hisham Hamed Ahmed

Rahmat Diko Edfi

# Summary

This assignment report focuses on the practical implementation of smart contracts using blockchain technology in industrial and shipbuilding applications. It explores the benefits of smart contracts, such as automation, efficiency, and reduced paperwork. Additionally, the report aims to identify new methods to enhance the usage of smart contracts in these domains. By leveraging blockchain and smart contracts, industries can streamline processes and improve transparency, security, and trust in industrial and shipbuilding applications. The report offers valuable insights and recommendations for implementing smart contracts effectively in industrial and shipbuilding sectors.

We, SEAS4.0 students, express our gratitude to Professor Paula Fraga Lamas for their lectures, their patient explanations, and for generously devoting their time to us. Additionally, we extend our appreciation to Professor Tiago M. Fernández Caramés for his valuable.

Report date: 05.06.2023

Reported by

ABDOUL-MALIK BAKARI

KAUNG MYAT NOE AUNG

MUSAAB HISHAM HAMED AHMED

RAHMAT DIKO EDFI

# Table of Contents



# Figures

# 1. Introduction

The rapid advancement of enabling technologies in Industrial 4.0 has revolutionized the maritime and shipbuilding sectors. Sensing and Actuation Networks, Cloud and Edge Computing, Cyber-Physical Systems, Augmented, Mixed, and Virtual Reality, Blockchain, Unmanned Vehicles, and Information Management Systems offer unprecedented opportunities for efficiency, productivity, and innovation.

This report focuses on the integration of blockchain technology within these sectors, highlighting its impact and relevance in Industrial 4.0. Blockchain brings enhanced transparency, traceability, and decentralized trust, improving operational efficiency and streamlining processes. It ensures secure and immutable recording of data, reducing the risk of fraud, errors, and disputes.

By leveraging blockchain, stakeholders in the maritime and shipbuilding sectors achieve improved transparency and traceability throughout the value chain. Additionally, it enables secure and efficient information management systems, facilitating collaboration and optimized decision-making.

The integration of blockchain technology holds immense potential for revolutionizing operations, enhancing productivity, and driving innovation. It empowers these sectors to remain competitive, agile, and future proof in the dynamic landscape of Industrial 4.0.

# 2. Objectives

## 2.1 Objective of the Study

This study aims to assess the implications of Industry 4.0 technologies on information management systems in the maritime industry. It will explore the security considerations associated with cloud and edge computing, as well as the use of blockchain, and provide guidance for the practical implementation of smart contracts using blockchain technology. The objective is to provide insights and recommendations for effectively adopting and securing Industry 4.0 technologies in maritime industry applications.

## 2.2 Motivation and Objective of the Report

The motivation of this report is to address the challenges related to maintaining accurate maintenance records for HVAC systems in yachts, specifically as implemented in the ICPS assignment. In practical situations, there is a risk of tampering with these records by manipulating the running hours of the system, which can lead to inaccurate maintenance schedules and potential equipment failures. Additionally, there is a need to balance the level of information shared, as excessive details may compromise privacy and security.

To overcome these challenges, the report proposes the implementation of a smart contract-based maintenance system specifically for the HVAC system in yachts. By leveraging blockchain technology and smart contracts, the system aims to ensure the integrity of maintenance records by securely recording the running hours of the HVAC system. This eliminates the risk of tampering and provides a transparent and verifiable method for recording and paying for maintenance services. The motivation behind this approach is to enhance the efficiency, accuracy, and trustworthiness of the maintenance process for the implemented HVAC system in yachts, ultimately improving the overall performance and reliability of these systems.

## 3. Case Scenario: Recording and Paying for Maintenance Services

**Actors**:

Yacht Owner

Maintenance Service Provider

**Goal**:

The goal of this use case is to enable the yacht owner to securely record and pay for maintenance services provided by the maintenance service provider.

**Steps**:

The running hours of the system are recorded on the block chain.

The yacht owner initiates a request for maintenance service based on the running hours.

The maintenance service provider performs the necessary maintenance tasks on the HVAC system.

As the maintenance service provider completes the service, the running hours of the HVAC system are automatically recorded and stored in the blockchain.

The maintenance service provider generates an invoice for the service and submits it to the yacht owner but here is fixed.

The yacht owner reviews the invoice and, if satisfied, approves the payment.

The payment is executed through the smart contract, transferring the agreed-upon amount from the yacht owner's account to the maintenance service provider's account.

Expected Result:

The use case enables the secure and transparent recording of maintenance activities and running hours, as well as the seamless payment for services rendered. It reduces the risk of tampering with maintenance records, improves accuracy in scheduling future maintenance, and fosters trust between the yacht owner and the maintenance service provider.

# 3. Design aspects

The proposed smart contract has several design aspects that can be highlighted:

**Data Protection and Controlled Access:**

The smart contract safeguards sensitive information, such as running hours, and maintenance addresses, by storing them as private variables. By keeping these variables private, the contract prevents anyone outside the contract from directly accessing or modifying them. This approach ensures that only authorized interactions and modifications occur, maintaining the security and integrity of the contract's internal information.


Fig 3. 1 REQUESTER address is only provided when the payment must be made.

**Maintenance Trigger and Thresholds:**

The smart contract calculates running hours for the heater, AC, and fan components based on running hours. Specific thresholds are set (100 hours for AC and heater, 200 hours for the fan) to trigger maintenance requests. Once a threshold is reached, the contract identifies the appropriate maintenance service provider and determines the payment amount. The simulation for the heater can be seen in the above figure 3.1.

**Payment and Approval System:**

The contract includes a payment and approval system for maintenance services.

Only the contract owner can approve the payment request from the maintenance service provider. Upon approval, the contract transfers the specified amount of ether from the contract to the maintenance service provider's address.

**Termination of the Contract:**

Either the maintenance service provider or the contract owner can terminate the contract. This flexibility is designed to handle situations where the maintenance service provider is no longer able to provide the service or if there is a change in ownership of the contract.

When the contract is destroyed, and any remaining funds are transferred to the contract owner. This ensures that the owner can recover their funds in case the maintenance service provider cannot fulfill their obligations or if there are changes in contract ownership.

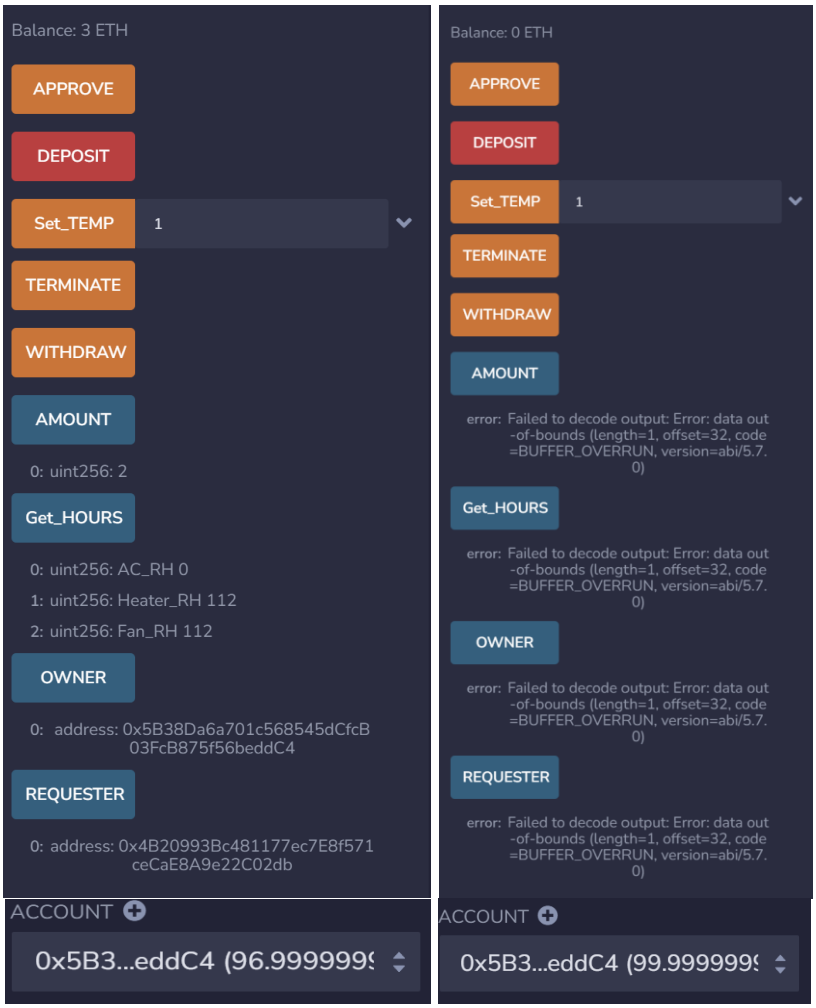


Fig 3. 2 Before and After Termination of the Contract

**Deposit and Withdrawal:**

The contract restricts the ability to deposit funds to only the contract owner, ensuring that only authorized individuals can add funds to the contract for future payments. This measure helps prevent money laundering and minimizes confusion regarding the source of payments.

Similarly, the contract owner has the exclusive right to withdraw any remaining funds from the contract. This ensures that the funds are returned to the same account from which they were originally deposited, maintaining transparency and accountability.

**Access Control:**

The contract differentiates the contract owner (yacht owner) and the maintenance service provider. Only the contract owner can approve maintenance requests and perform withdrawal operations. The require statements ensure that only the designated parties can execute specific functions. These design aspects aim to provide modularity, security, and control in managing maintenance requests, payments, and contract termination.

These design aspects collectively contribute to the modularity, security, and control of the smart contract, enabling efficient management of maintenance requests, payments, contract termination, and access to sensitive data.

# 4. Implementation of the Smart Contract

### Development Environment

The smart contract is implemented using Solidity, a programming language specifically designed for creating smart contracts on the Ethereum blockchain. The contract begins with the pragma statement, specifying the version of the Solidity compiler to be used.

### Variables

The contract includes several private variables to store and manage data related to temperature, running hours, and maintenance addresses. These variables, such as temperature, steps, acHours, heaterHours, and fanHours, are declared as uint8 and uint256 to efficiently store the required values.

Sample addresses for maintenance services, such as acMaintenanceAddress, heaterMaintenanceAddress, and fanMaintenanceAddress, are defined as private variables. These addresses represent the service providers responsible for maintenance tasks.

The contract also defines public variables for the contract owner (OWNER), the requester of maintenance services (REQUESTER), and the amount of ether to be transferred (AMOUNT). The OWNER variable is defined as a payable address, allowing the contract owner to send and receive funds.

```
contract HVAC_maintenance {
    uint8 private temperature;
    uint8 private steps;
    uint256 private acHours;
    uint256 private heaterHours;
    uint256 private fanHours;

    // Sample addresses for maintenance services
    address private acMaintenanceAddress = 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2;
    address private heaterMaintenanceAddress = 0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db;
    address private fanMaintenanceAddress = 0x78731D3Ca6b7E34aC0F824c42a7cC18A495cabaB;

    // Address of the contract OWNER
    address payable public OWNER = payable(msg.sender);
    // Address of the REQUESTER
    address payable public REQUESTER;
    // AMOUNTS of ether to transfer in Wei
    uint private AMOUNTS;
```

Fig 4. 1 Variables Usage

### Fuctions

The Set_TEMP function is a public function that accepts the temperature as an input. It validates the temperature range and calculates the number of steps based on the temperature difference from the reference value of 15. The function then updates the running hours of the respective components (heater, AC, and fan). If any of the running hour thresholds are reached (100 hours for AC and heater, 200 hours for the fan), the REQUESTER and AMOUNT variables are set accordingly for the maintenance service.

The Get_HOURS function is a public view function that returns the current running hours for the AC, heater, and fan components.

The APPROVE function allows the contract owner to approve a maintenance request. It verifies that the sender is the OWNER and there is a pending payment request (AMOUNT > 0). If the conditions are met, the function transfers the specified amount of ether to the REQUESTER address. Additionally, it resets the running hours for the respective component to 0 and clears the REQUESTER and AMOUNT variables.

The DECLINE function enables either the maintenance service provider or the contract owner to terminate the contract. Once invoked, the function self-destructs the contract, transferring any remaining funds to the OWNER address.

The WITHDRAW function allows the contract owner to withdraw any remaining funds from the contract. It verifies that the sender is the OWNER and retrieves the balance of the contract. The function transfers the balance amount to the OWNER address.

The DEPOSIT function is a payable function that allows the contract owner to deposit funds into the contract. It verifies that a non-zero amount is being deposited and the sender is the OWNER.

These implementation details ensure that the smart contract operates securely and transparently. It enables temperature-based calculation of running hours, triggering maintenance requests at predefined thresholds, facilitating payment approval and transfer, and providing mechanisms for contract termination and fund management.

```
function TERMINATE() public {    34247 gas
    require(msg.sender == REQUESTER || msg.sender == OWNER, "Only REQUESTER or OWNER can TERMINATE");
    selfdestruct(OWNER);
}

function WITHDRAW() public {    infinite gas
    require(msg.sender == OWNER, "Only OWNER can WITHDRAW");
    uint balance = address(this).balance;
    OWNER.transfer(balance);
}

function DEPOSIT() public payable {    2689 gas
    // Require a non-zero AMOUNTS to be deposited
    require(msg.value > 0, "DEPOSIT AMOUNTS must be greater than zero");
    require(msg.sender == OWNER, "Only OWNER can DEPOSIT");
}
function AMOUNT() public view returns (uint) {    infinite gas
    return AMOUNTS / 1 ether;
}
```

Fig 4. 2 Functions Usage

# 5. Test and validation

The HVAC_maintenance contract was thoroughly tested to ensure its reliable and correct functionality. The following functionality tests were performed:

Set_TEMP Test: This test verified the behavior of the Set_TEMP function by setting different temperature values within the valid range. The test ensured that the running hours for the AC, heater, and fan were updated accurately based on the provided temperature. It also checked if the correct maintenance service requester and amount were assigned when the running hours met the specified threshold values.

Get_HOURS Test: The test confirmed that the returned running hours for the AC, heater, and fan matched the values stored in the contract, ensuring the accurate recording of the maintenance hours.

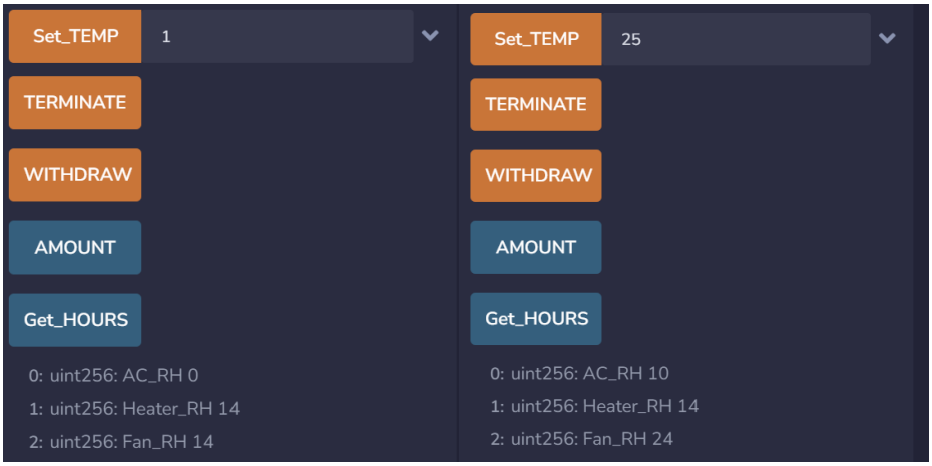


Fig 5. 1 Checking Running Hour of AC, Heater and Fan

DEPOSIT Test: The DEPOSIT function was tested to validate its behavior. The test ensured that only the contract owner could deposit funds which means the account deployed the contract can only do the deposit, preventing invalid or unauthorized deposits.


Fig 5. 2 Before and After Deposit

WITHDRAW Test: It is verified that only the contract owner was permitted to withdraw funds. The test ensured that the contract balance was accurately transferred to the contract owner, maintaining the integrity of the financial transactions.

AMOUNT Test: This test confirmed the accuracy of the AMOUNT function. It checked if the returned value correctly represented the stored amount in the contract, converted from Wei to Ether. This validation ensured the consistent and reliable representation of amounts in the contract.


Fig 5. 3 Different Amounts of Charges

APPROVE Test: This test focused on the APPROVE function to ensure its proper execution. It checked if only the contract owner was allowed to approve the maintenance service and if there was an amount to approve. The test also verified if the requested amount was transferred

correctly to the respective maintenance service provider and if the running hours were reset to zero for the corresponding service.

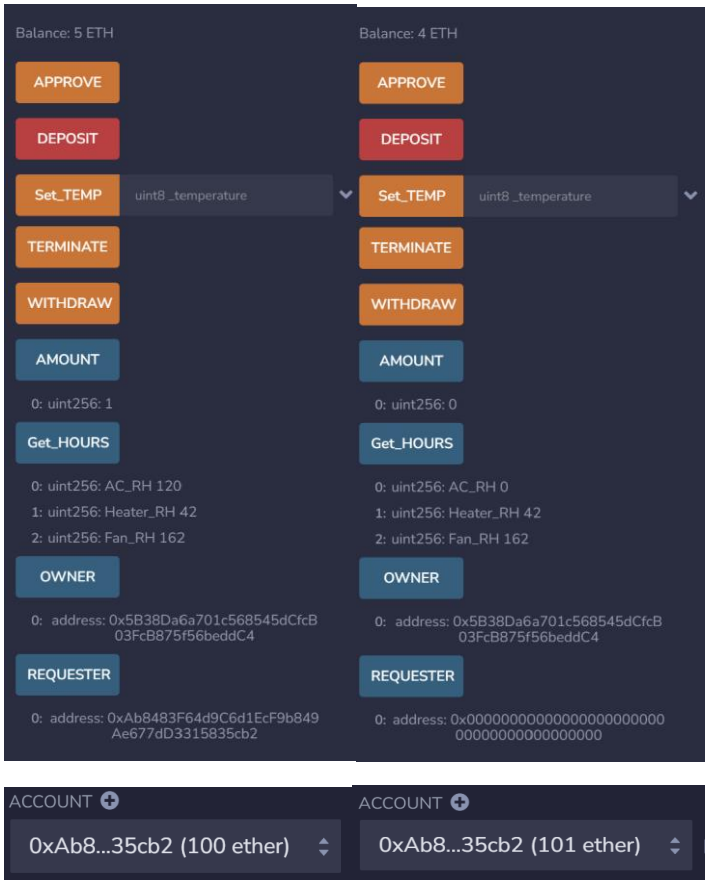


Fig 5. 4 Checking the Payment to Service Provider

TERMINATE Test: The test ensured that only the requester or the contract owner could terminate the contract. It verified if the contract self-destructed as expected and transferred the remaining funds to the contract owner, preventing any potential misuse of the contract. As same as Fig 3.2.

By conducting these comprehensive tests and validations, the HVAC_maintenance contract was thoroughly verified for its reliability, correctness, and adherence to the expected behavior. These measures ensure that the contract functions as intended, providing secure and transparent maintenance services and payments for the yacht owner and the maintenance service provider.

# 6. Conclusion

To conclude, the development and testing of this smart contract have yielded valuable insights and outcomes, showcasing its reliability in accurately tracking running hours and triggering maintenance requests. This ensures timely and efficient servicing of the HVAC system, resulting in improved performance and durability.

Overall, the utilization of blockchain technology in this smart contract represents a significant advancement in the realm of Industrial 4.0 technologies. It brings numerous benefits, including the ability to track and trace maintenance activities, facilitate data sharing among stakeholders, and streamline payment processes. Moreover, the immutable recording of maintenance activities and payments on the blockchain has greatly reduced the likelihood of disputes and fostered a transparent and accountable relationship between the yacht owner and maintenance service provider.

By leveraging the power of blockchain, this smart contract sets a precedent for the maritime industry and demonstrates how emerging technologies can drive advancements in data management and collaboration. Its implementation showcases the potential for increased efficiency, transparency, and reliability in maintenance operations. As the industry continues to evolve, the use of smart contracts and blockchain technology will likely become increasingly prevalent, revolutionizing how information is managed and shared within the maritime sector.