I4ET

SMART CARGO HOLD ENVIRONMENTAL MONITORING & SAFETY SYSTEM

PROJECT REPORT

PROPOSED BY:

Nimra Khan
Anel Bayangaliyeva
Simon Wakpal

Table of Contents

List of Figures

1. Introduction

Global grain trade relies heavily on maritime transport. Bulk carriers move massive harvests from surplus-producing regions to import-dependent nations, maintaining food availability, stabilizing prices, and supporting global food security. These vessels drastically lower per-ton shipping costs and reduce post-harvest losses—but the unique nature of grain and the structure of cargo holds introduce significant hazards that threaten both cargo quality and crew safety.

1.1 Problem Identification

Bulk carriers operate with sealed cargo holds during transit, leaving the crew unable to monitor internal conditions in real time. This creates a **blind zone** where environmental changes like **temperature spikes**, **humidity fluctuations**, **gas accumulation**, and **dust concentration** can escalate into serious threats—including **cargo spoilage**, **toxic atmospheres**, and **explosions**—without warning. The most critical and recurring threats include:

- **Thermal Stratification & Hot Spots**: Uneven temperature distribution inside the hold can create localized "hot zones" exceeding 20 °C. These pockets accelerate spoilage, promote mold growth, and increase the risk of self-heating. Cooler zones, by contrast, lead to condensation on ceiling panels, dripping moisture onto the grain below—triggering instability, cargo shifting, and the formation of hardened wet patches.
- **Humidity Extremes**: When relative humidity drops below 50 %, grain begins to dry out and release fine particulate dust, which is highly combustible. If humidity exceeds 70 %, it encourages microbial growth and causes the grain to cake—blocking discharge equipment and slowing unloading operations.
- **Toxic Gas Build-up**: Grain naturally emits carbon dioxide and may retain fumigants like phosphine. In the absence of ventilation and active air sampling, these gases accumulate in sealed holds, displacing oxygen and creating a highly toxic atmosphere. Unmonitored entry into such environments can cause unconsciousness, poisoning, or even death.
- **Explosive Dust Risk**: Dust stirred during loading or shifting can ignite from a static discharge, smoldering ember, or mechanical spark. A primary flash may trigger a secondary explosion if more dust is lifted into the air—causing catastrophic structural failure and loss of life.
- **Post-Unloading Dangers**: Even after cargo is discharged, dangerous gas residues and unstable residual grain piles remain. Without persistent monitoring, post-entry inspections can result in sudden collapse or exposure to toxic vapors.

1.2 Vision and Approach

As ships move toward digital transformation under Industry 4.0, the integration of Industrial Cyber-Physical Systems (ICPS) becomes central to redefining how safety, efficiency, and automation are achieved onboard. Our project leverages this paradigm to close the real-time monitoring gap within sealed cargo spaces.

By embedding intelligence directly into physical environments, SCHEMSS transforms passive grain holds into responsive systems capable of sensing, interpreting, and acting autonomously. It combines onboard sensors, wireless data transmission, and a cloud-based dashboard to establish a continuous feedback loop—making the vessel environment observable and actionable.

This transition from traditional to intelligent systems marks a significant progression in maritime automation. With SCHEMSS, we aim to enhance cargo stability, reduce operational risk, and support proactive decision-making—bridging digital computation with real-world maritime safety procedures.

2. Smart Cargo Hold Environmental Monitoring & Safety System (SCHEMSS)

2.1 System Overview

The SCHEMSS system is built on a tightly integrated flow from physical sensing to cloud-based control. At its core, the ESP8266 microcontroller collects real-time environmental data from the onboard temperature, humidity, gas, and flame sensors. These inputs are continuously monitored and compared against predefined safety thresholds within the device's embedded logic. When abnormal readings are detected—such as elevated temperature, humidity deviations, gas presence, or flame—the system immediately triggers LED indicators and a buzzer, providing local visual and audible alerts.This modular configuration is designed to function with minimal human intervention and emphasizes seamless cloud integration.

SENSORS	ACTUATORS
DHT11 (Temp & Humidity Sensor)	Extraction fan, Red LED (temperature), Blue LED (humidity)
MQ-6 Gas Sensor	Extraction fan, Green LED (gas detection)
Flame Sensor	Yellow LED (flame), Passive Buzzer (audible alarm)

Table 1 - Sensors and Actuators

2.2 Objectives

The core operational goals of the system are as follows:

- Maintaining Environmental Stability and Preserving Cargo Integrity

  The integrated **temperature and humidity sensors** monitor variations in hold conditions that can compromise grain quality. If the temperature exceeds safe limits or relative humidity falls outside the optimal range, SCHEMSS activates visual and audible alerts. This early warning mechanism helps mitigate risks of mold growth, condensation, cargo caking, and spoilage—ultimately preserving structural integrity and discharge efficiency.

- Hazardous Gas Detection and Crew Safety

  Toxic or flammable gases such as carbon dioxide and residual phosphine can accumulate in sealed holds. The MQ-6 gas sensor performs continuous air sampling and issues alerts using a **blue LED** and buzzer if gas concentrations rise to hazardous levels. This functionality offers critical protection against oxygen displacement, poisoning, or explosion.

- Fire Risk Monitoring and Rapid Response

  The **flame sensor** provides real-time fire detection by identifying the visual presence of an open flame. Upon detection, SCHEMSS immediately activates the **red** LED and buzzer, enabling rapid response protocols and minimizing the risk of escalation.

- Real-Time Remote Monitoring

  Data from all sensors is transmitted via MQTT to a Node-RED dashboard on FlowFuse. This allows ship operators to monitor the cargo environment remotely and intervene proactively.

2.3 The ICPS Flow Chart

The ICPS flowchart outlines the logic structure of the SCHEMSS system, mapping how data is sensed, evaluated, and acted upon. It illustrates the full operational sequence—from sensor initialization to environmental analysis, alert generation, and system response. All decision branches and control actions can be traced visually in the diagram below.

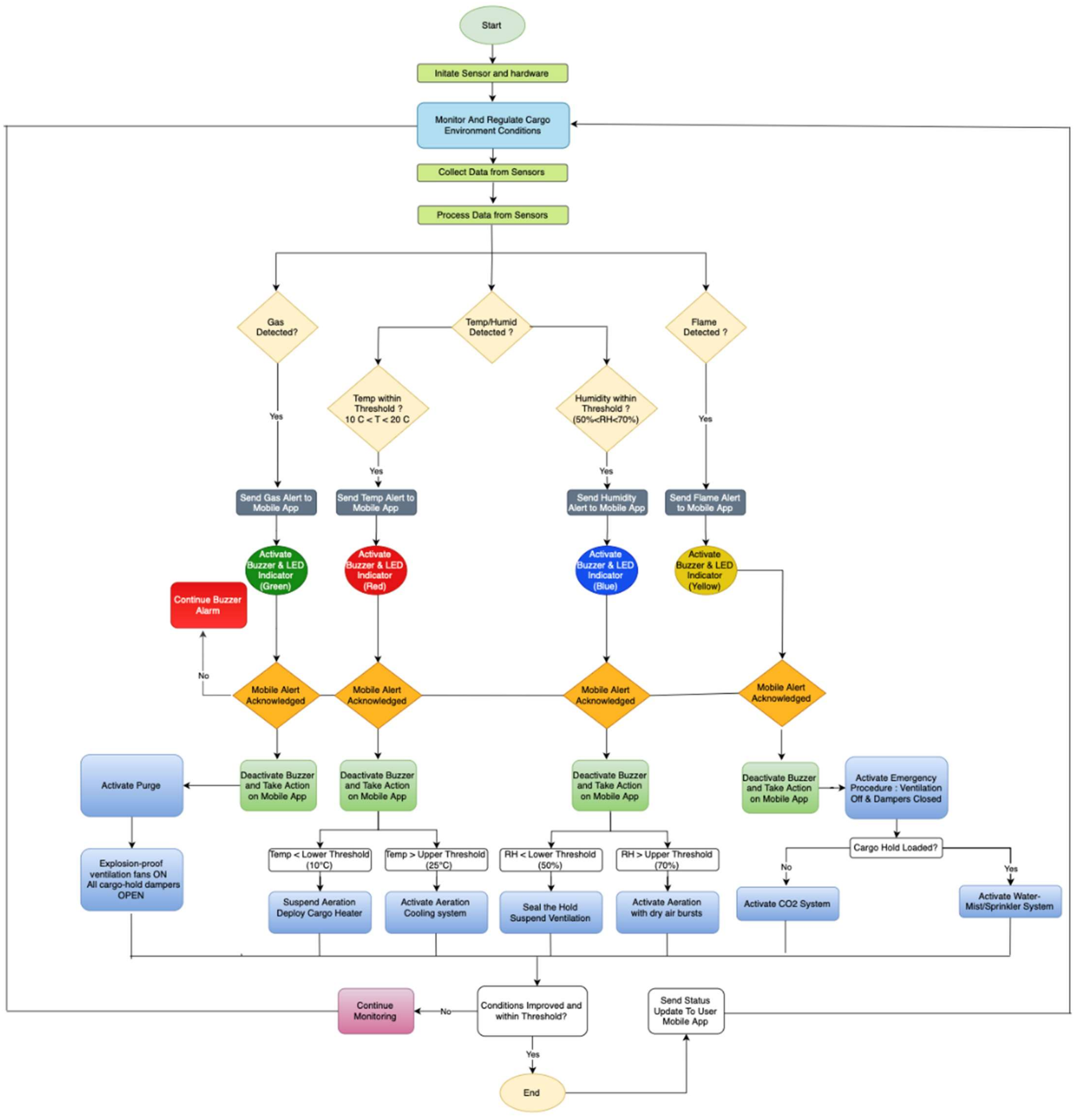

Start
Initate Sensor and hardware
Monitor And Regulate Cargo Environment Conditions
Collect Data from Sensors
Process Data from Sensors
Gas Detected?
Temp/Humid Detected ?
Flame Detected ?
Temp within Threshold ? 10 C < T < 20 C
Humidity within Threshold ? (50%<RH<70%)
Yes
Yes
Yes
Yes
Send Gas Alert to Mobile App
Send Temp Alert to Mobile App
Send Humidity Alert to Mobile App
Send Flame Alert to Mobile App
Activate Buzzer & LED Indicator (Green)
Activate Buzzer & LED Indicator (Red)
Activate Buzzer & LED Indicator (Blue)
Activate Buzzer & LED Indicator (Yellow)
Continue Buzzer Alarm
No
Mobile Alert Acknowledged
Mobile Alert Acknowledged
Mobile Alert Acknowledged
Mobile Alert Acknowledged
Activate Purge
Deactivate Buzzer and Take Action on Mobile App
Deactivate Buzzer and Take Action on Mobile App
Deactivate Buzzer and Take Action on Mobile App
Deactivate Buzzer and Take Action on Mobile App
Activate Emergency Procedure : Ventilation Off & Dampers Closed
Explosion-proof ventilation fans ON All cargo-hold dampers OPEN
Temp < Lower Threshold (10°C)
Temp > Upper Threshold (25°C)
RH < Lower Threshold (50%)
RH > Upper Threshold (70%)
Cargo Hold Loaded?
No
Yes
Suspend Aeration Deploy Cargo Heater
Activate Aeration Cooling system
Seal the Hold Suspend Ventilation
Activate Aeration with dry air bursts
Activate CO2 System
Activate Water-Mist/Sprinkler System
Continue Monitoring
No
Conditions Improved and within Threshold?
Send Status Update To User Mobile App
Yes
End

Figure 1 - ICPS Flow Chart

2.4 Communication Architecture

As shown in the diagram, the SCHEMSS communication system is structured in three clear layers. At the base, the **Device Layer** includes sensors such as the DHT11 (for temperature and humidity), MQ-6 gas sensor, flame sensor, and actuators like RGB LEDs and a buzzer—all interfaced through the ESP8266 microcontroller. These components are physically connected and interact directly onboard.

The **Routing Layer** bridges the device-level hardware with the cloud using standard internet and MQTT protocol symbols, illustrating how real-time data is transmitted wirelessly from the ship.

At the top, the **Cloud Layer** is represented by FlowFuse, which handles visualization and control logic through the Node-RED platform. Together, these three layers illustrate the end-to-end flow of data, from onboard collection to remote monitoring.

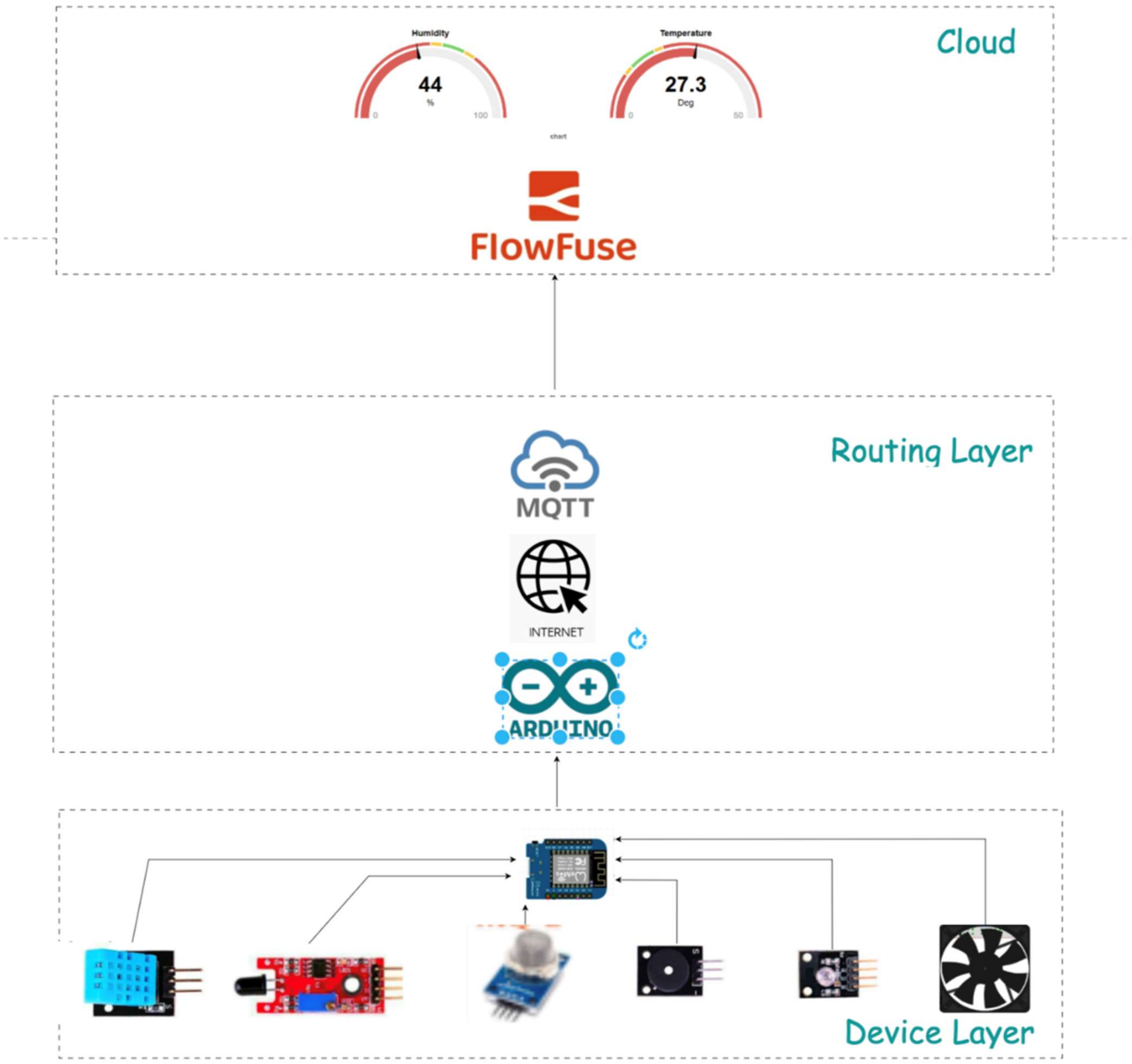


Figure 2 - Cloud Computing Architecture

2.6 The ICPS System Connection

The following diagram illustrates the physical layout and wiring of the SCHEMSS system components. Sensors and actuators are connected to the ESP8266-based microcontroller using jumper wires and mounted on a breadboard. This wiring configuration ensures real-time data acquisition and reliable communication with the cloud infrastructure.

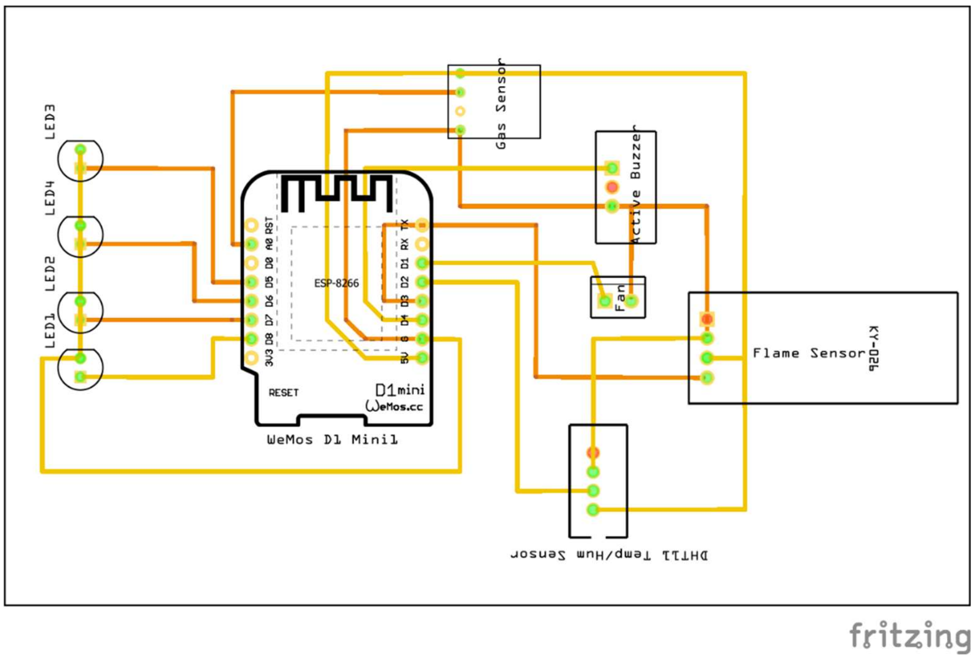


Figure 3 - Fritzing PCB

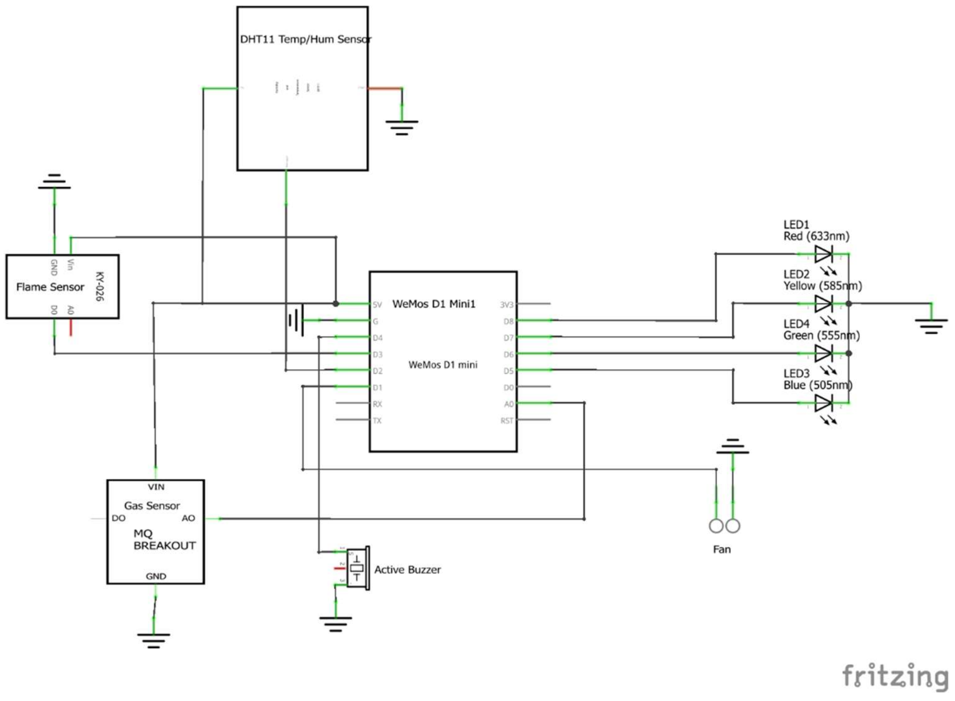


Figure 4 - Fritzing Schematic

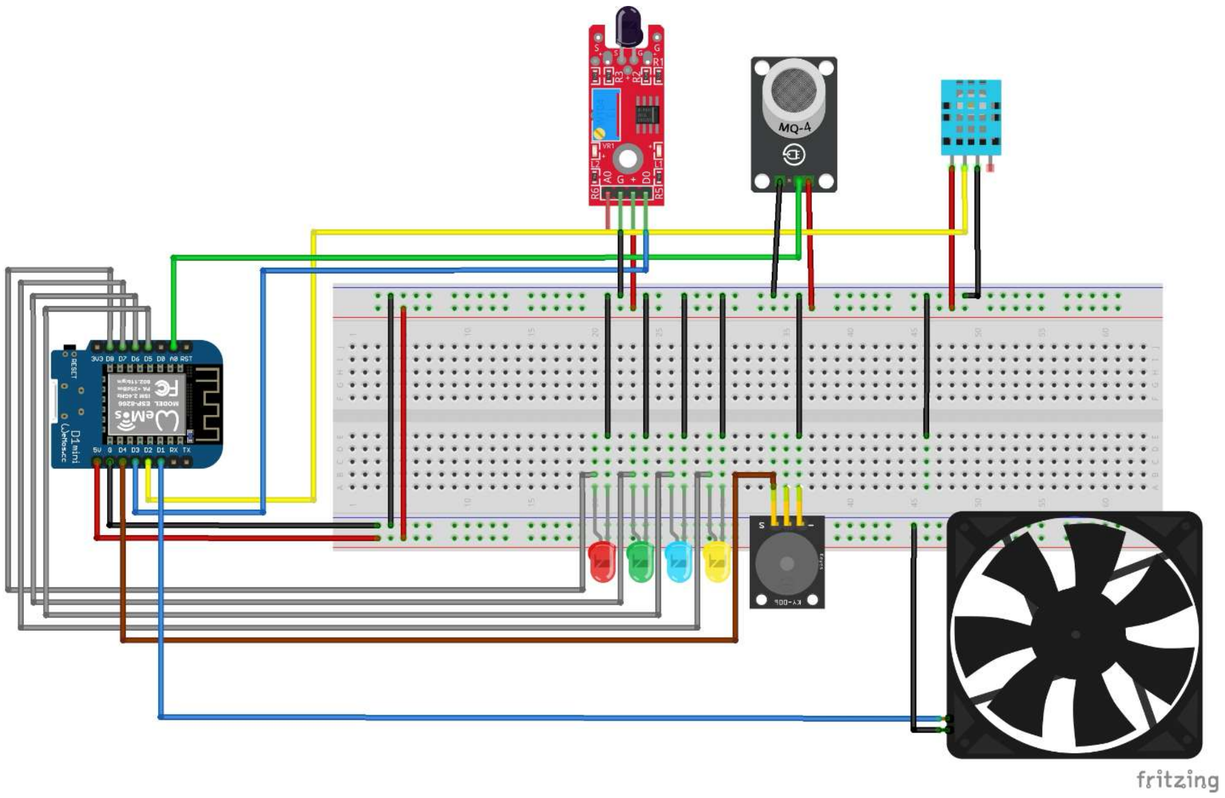
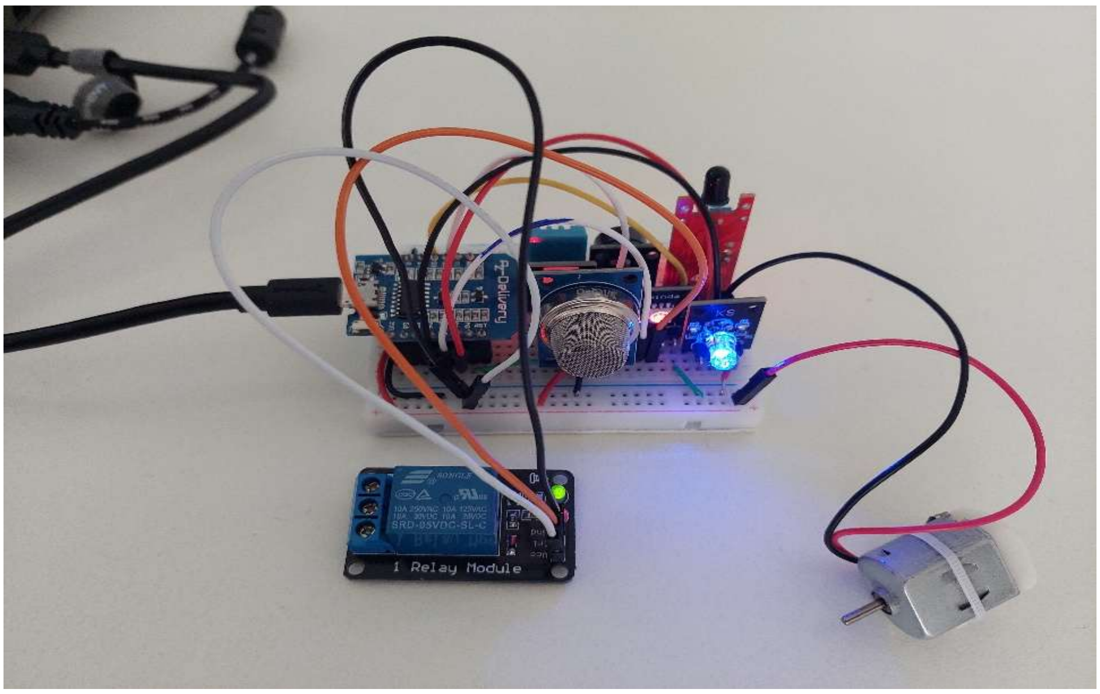


Figure 5 - Breadboard



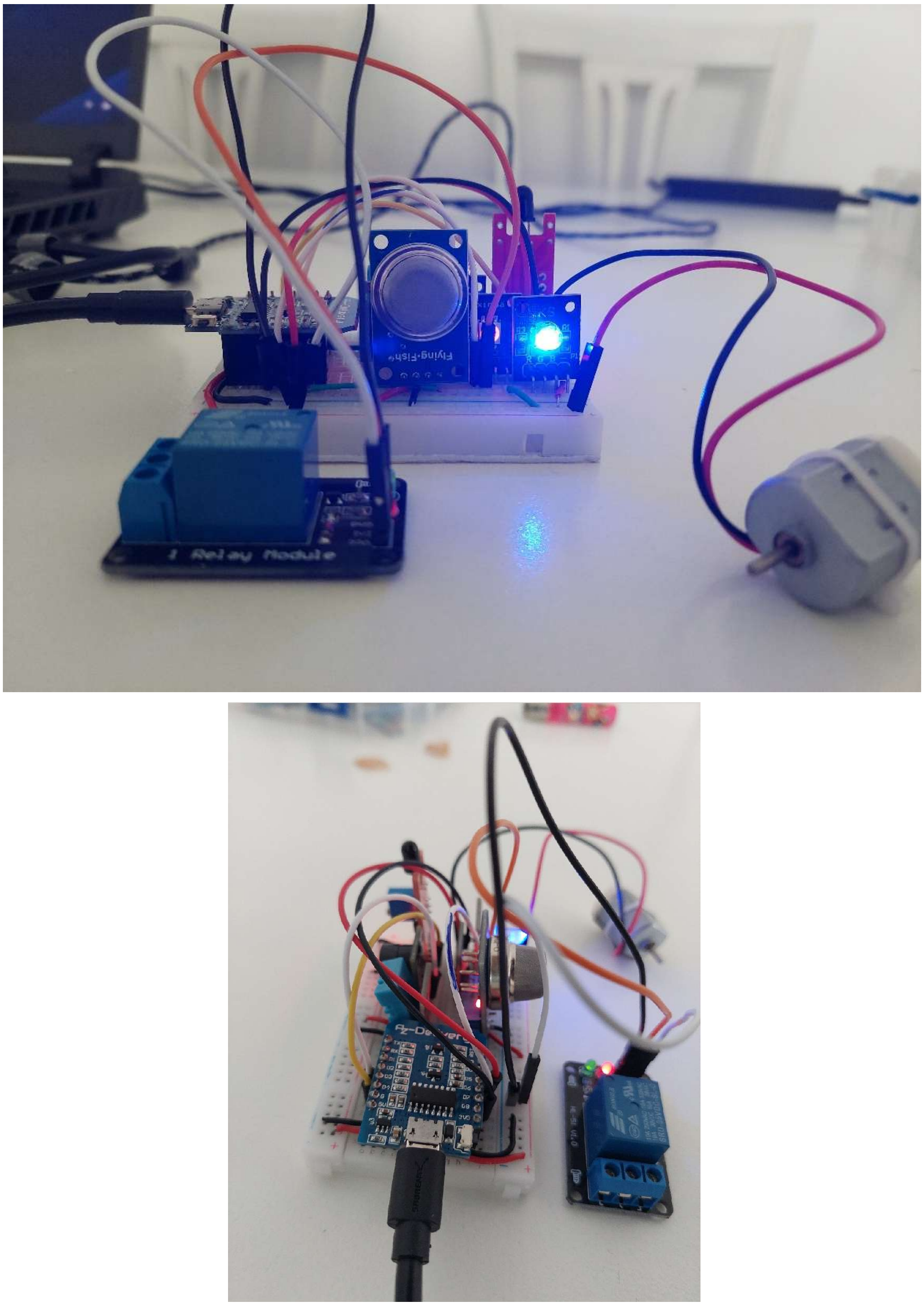
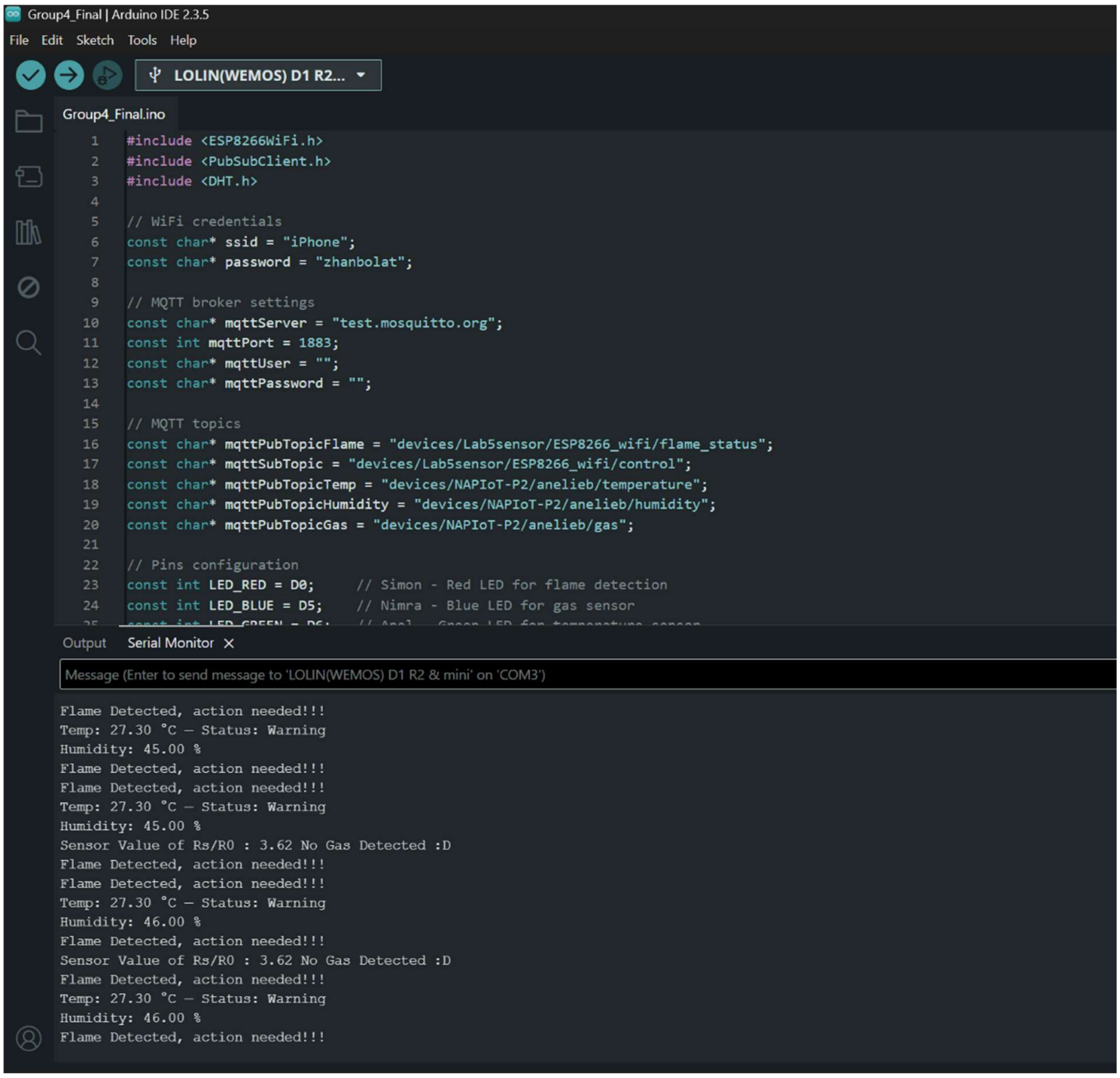


Figure 6 - Combined Setup

2.7 Test and Validation

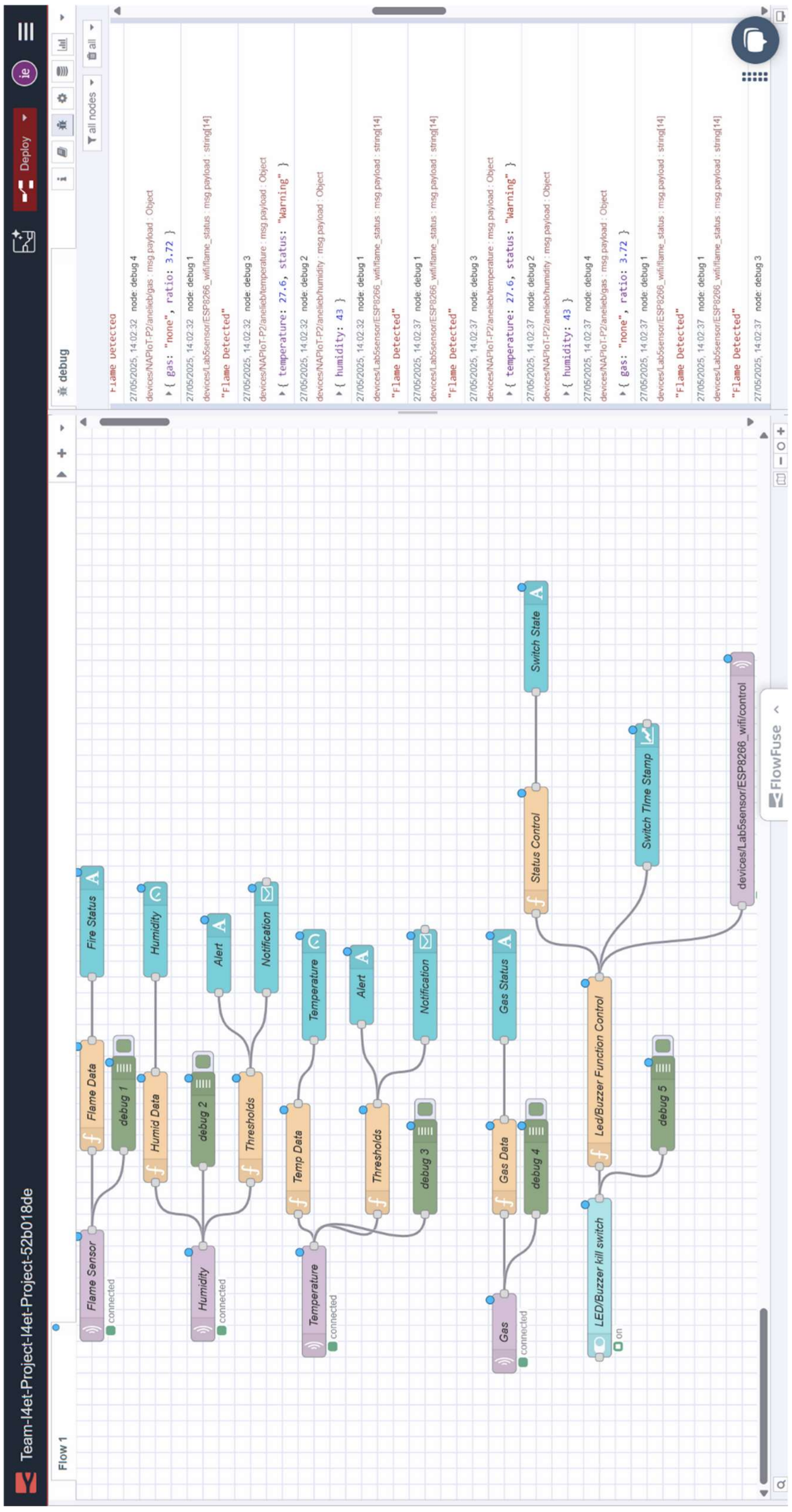
These images display the real-time execution of our SCHEMSS system, which runs on a WeMos D1 Mini (ESP8266) microcontroller. The Serial Monitor output confirms successful communication with all connected sensors and demonstrates how the system detects and responds to environmental hazards—including gas presence, flame detection, and abnormal temperature/humidity—by publishing MQTT alerts and triggering corresponding LEDs and buzzer indicators.


Group4_Final | Arduino IDE 2.3.5
File Edit Sketch Tools Help
LOLIN(WEMOS) D1 R2...
Group4_Final.ino
Output
Serial Monitor
Message (Enter to send message to 'LOLIN(WEMOS) D1 R2 & mini' on 'COM3')

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <DHT.h>

// WiFi credentials
const char* ssid = "iPhone";
const char* password = "zhanbolat";

// MQTT broker settings
const char* mqttServer = "test.mosquitto.org";
const int mqttPort = 1883;
const char* mqttUser = "";
const char* mqttPassword = "";

// MQTT topics
const char* mqttPubTopicFlame = "devices/Lab5sensor/ESP8266_wifi/flame_status";
const char* mqttSubTopic = "devices/Lab5sensor/ESP8266_wifi/control";
const char* mqttPubTopicTemp = "devices/NAPIoT-P2/anelieb/temperature";
const char* mqttPubTopicHumidity = "devices/NAPIoT-P2/anelieb/humidity";
const char* mqttPubTopicGas = "devices/NAPIoT-P2/anelieb/gas";

// Pins configuration
const int LED_RED = D0;     // Simon - Red LED for flame detection
const int LED_BLUE = D5;    // Nimra - Blue LED for gas sensor
```

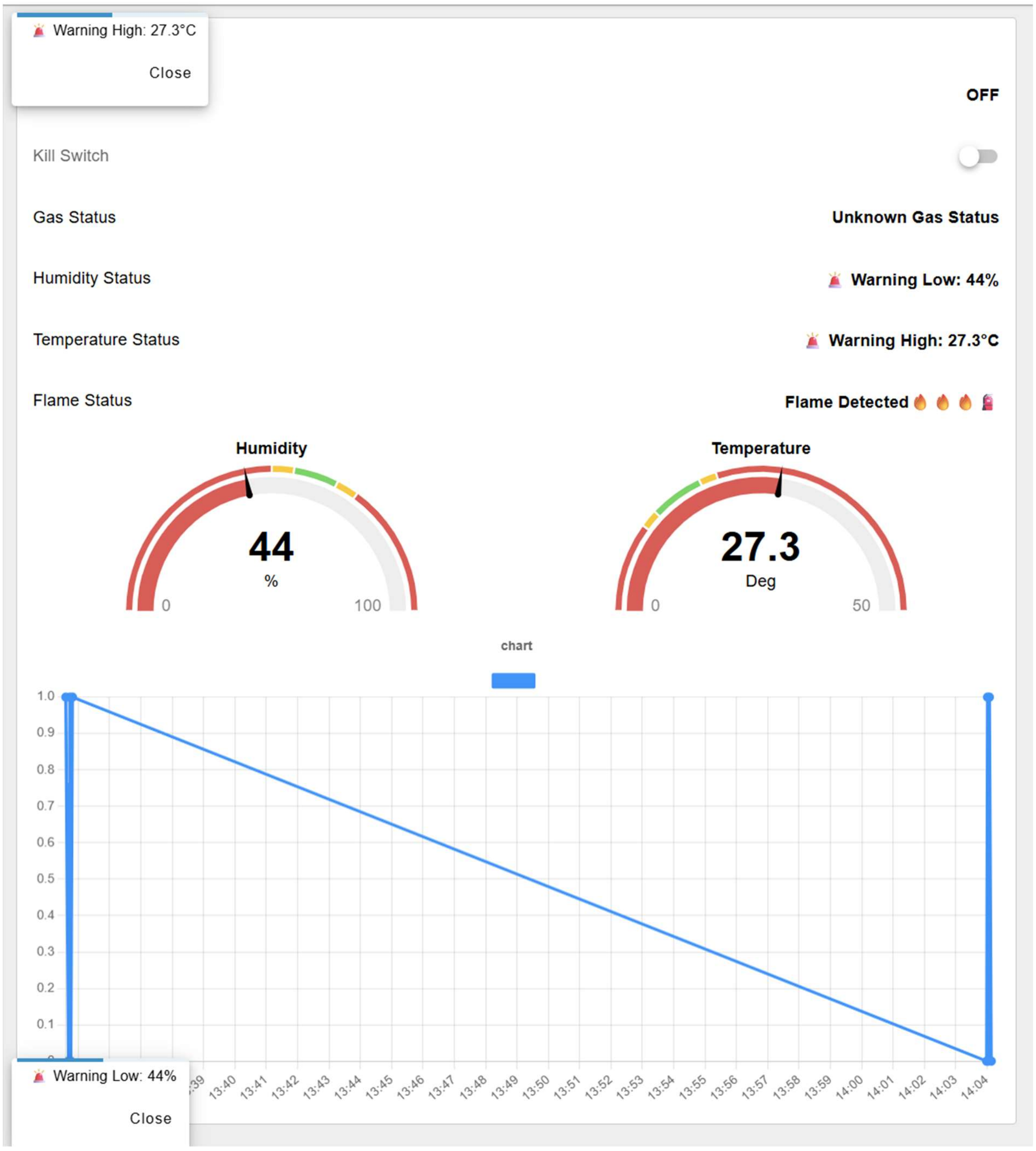
```
Flame Detected, action needed!!!
Temp: 27.30 °C — Status: Warning
Humidity: 45.00 %
Flame Detected, action needed!!!
Flame Detected, action needed!!!
Temp: 27.30 °C — Status: Warning
Humidity: 45.00 %
Sensor Value of Rs/R0 : 3.62 No Gas Detected :D
Flame Detected, action needed!!!
Flame Detected, action needed!!!
Temp: 27.30 °C — Status: Warning
Humidity: 46.00 %
Flame Detected, action needed!!!
Sensor Value of Rs/R0 : 3.62 No Gas Detected :D
Flame Detected, action needed!!!
Temp: 27.30 °C — Status: Warning
Humidity: 46.00 %
Flame Detected, action needed!!!
```

Figure 7 - Arduino Code



```
Flame Detected
27/05/2025, 14:02:32  node: debug 4
devices/NAPIoT-P2/anelieb/gas : msg.payload : Object
▸{ gas: "none", ratio: 3.72 }
27/05/2025, 14:02:32  node: debug 1
devices/Lab5sensor/ESP8266_wifi/flame_status : msg.payload : string[14]
"Flame Detected"
27/05/2025, 14:02:32  node: debug 3
devices/NAPIoT-P2/anelieb/temperature : msg.payload : Object
▸{ temperature: 27.6, status: "Warning" }
27/05/2025, 14:02:32  node: debug 2
devices/NAPIoT-P2/anelieb/humidity : msg.payload : Object
▸{ humidity: 43 }
27/05/2025, 14:02:32  node: debug 1
devices/Lab5sensor/ESP8266_wifi/flame_status : msg.payload : string[14]
"Flame Detected"
27/05/2025, 14:02:37  node: debug 1
devices/Lab5sensor/ESP8266_wifi/flame_status : msg.payload : string[14]
"Flame Detected"
27/05/2025, 14:02:37  node: debug 3
devices/NAPIoT-P2/anelieb/temperature : msg.payload : Object
▸{ temperature: 27.6, status: "Warning" }
27/05/2025, 14:02:37  node: debug 2
devices/NAPIoT-P2/anelieb/humidity : msg.payload : Object
▸{ humidity: 43 }
27/05/2025, 14:02:37  node: debug 4
devices/NAPIoT-P2/anelieb/gas : msg.payload : Object
▸{ gas: "none", ratio: 3.72 }
27/05/2025, 14:02:37  node: debug 1
devices/Lab5sensor/ESP8266_wifi/flame_status : msg.payload : string[14]
"Flame Detected"
27/05/2025, 14:02:37  node: debug 1
devices/Lab5sensor/ESP8266_wifi/flame_status : msg.payload : string[14]
"Flame Detected"
27/05/2025, 14:02:37  node: debug 3
```

Figure 8 - FlowFuse Architecture



Warning High: 27.3°C

Close

OFF

Kill Switch	
Gas Status	Unknown Gas Status
Humidity Status	Warning Low: 44%
Temperature Status	Warning High: 27.3°C
Flame Status	Flame Detected

Warning Low: 44%

Close

Figure 9 - Dashboard

3. Smart Contract — Warranty and Bonus Automation for SCHEMSS

3.1 Motivation

As the maritime industry embraces digitalization, ensuring safety and reliability in sealed cargo holds becomes increasingly vital. These environments, especially when transporting sensitive goods, demand constant monitoring to detect hazards such as fire, toxic gas accumulation, or humidity temperature related damage. The SCHEMSS system was developed in response to these needs, combining advanced sensors with automated digital agreements to maintain high safety standards. Traditional warranty and support systems—dependent on paperwork, human validation, and trust—are inadequate for such critical applications. Our motivation is to offer a robust and transparent solution that enforces accountability, simplifies support, and incentivizes performance, all while minimizing disputes and downtime.

3.2 Case Definition

We install three essential environmental monitoring sensors—TEMP_HUM (temperature and humidity), GAS (toxic gas detection), and FLAME (fire detection)—within the sealed cargo holds of a client's vessel. After payment confirmation, a smart contract named SCHEMSSWarranty is deployed on the Ethereum blockchain. This contract governs a two-year warranty period for each sensor. If no faults are reported during this time, the warranty naturally expires. Vessels operating without faults for at least 90 consecutive days are eligible for a bonus. Faults can be reported by the vessel owner, triggering a verification process by the company. Upon confirmation, a full refund is issued for the faulty sensor if it falls within the warranty period.

3.3 Objectives

The SCHEMSS smart contract introduces a fully automated framework for managing environmental sensor installations onboard cargo vessels. Designed to enforce warranties and incentivize operational excellence without human intervention, the system offers unique benefits to both the service provider and the vessel owner. By shifting all conditions and responses into blockchain-executed logic, the contract ensures transparency, fairness, and efficiency.

The table below outlines the distinct but complementary objectives for each party:

Stakeholder	Objectives
Company (Service Provider)	• Enforce warranty using on-chain timestamps. • Log installation metadata for transparency. • Automate payments, bonuses, and refunds. • Remove paperwork and reduce claim disputes. • Improve service credibility through transparent processes. • Enhance service reputation through objective, automated post-installation support.
Vessel Owner (Client)	• Access tamper-proof installation and warranty records. • Report faults via secure contract functions. • Receive refunds automatically for verified faults. • Earn bonuses for fault-free operation. • Benefit from a fully digital and frictionless support process.

Table 2 - Objectives

This shared framework creates a mutually beneficial system: the company minimizes risk and overhead, while the vessel owner enjoys responsive, secure, and transparent equipment support. SCHEMSS thereby establishes not just a technical solution, but a scalable governance model for digital maritime operations in the industry 4.0 era.

3.3 Design and Implementation

The SCHEMSSWarranty smart contract, developed in Solidity, facilitates the management of sensor installations using blockchain-based automation. It integrates lifecycle tracking, payment logic, and operational incentives into one cohesive system.

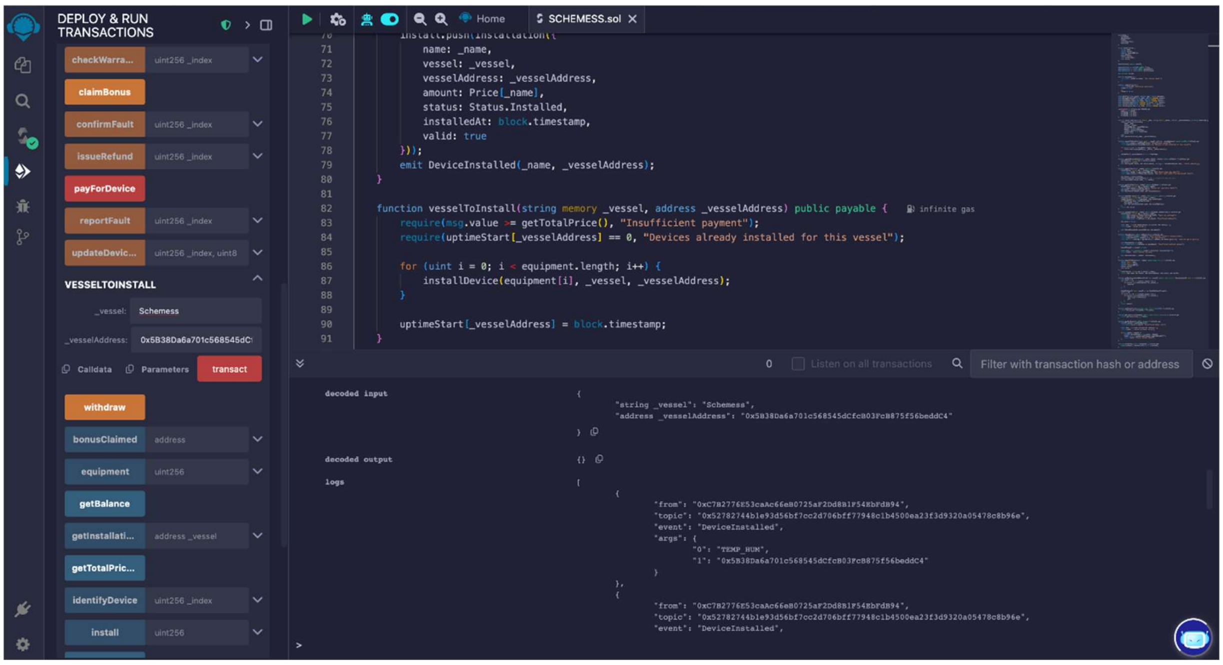
- **Automated Payment Validation:** Verifies full payment upon installation request. Logs installation details including vessel address and timestamp.
- **Sensor Registration:** Records metadata on-chain for each installed sensor, including status, price, and warranty timestamp.
- **Fault Reporting and Confirmation:** Allows vessel owners to report issues only during the Operational phase. Faults must be confirmed by the company before action is taken.
- **Warranty Enforcement:** Tracks warranty validity based on installation time. Expiration disables refund eligibility.
- **Refund Execution:** On fault confirmation within warranty, full refund is transferred to the vessel's address securely.

- **Bonus Distribution:** If no fault occurs in the first 90 days, the vessel can claim a 1 ETH performance reward.
- **Secure Fund Management:** Only the contract owner can withdraw or manage funds. Ether-based transactions are verified with noReentrancy protection.
- **Data Access Functions:** Functions like identifyDevice() and getInstallationsByVessel() provide full traceability.
- **Security Features:** Includes onlyOwner and noReentrancy modifiers. Disallows direct payments with disabled receive() and fallback().

This design ensures an end-to-end, transparent, and automated support mechanism for safety-critical maritime systems.

3.4 Tests and Validation

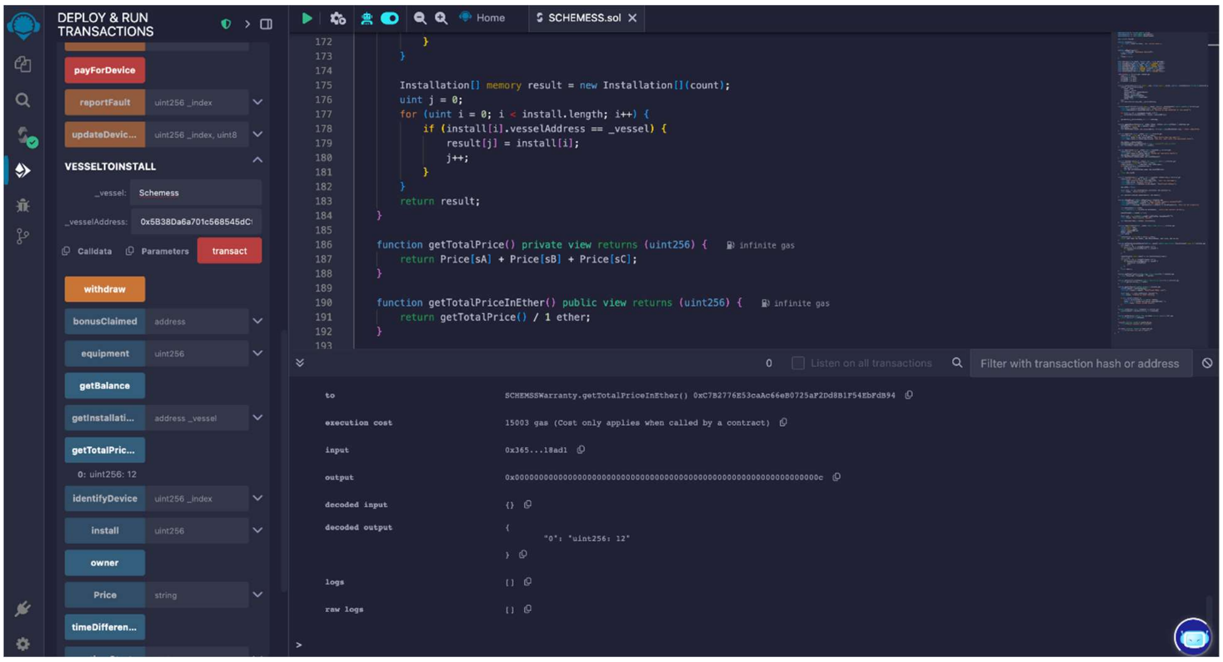
1) Demonstrated payment for the full SCHEMSS sensor package by executing vesselToInstall() with the vessel name 'Schemess' and wallet address 0x5B38...bdC4, triggering successful registration and on-chain logging of all installed units


DEPLOY & RUN TRANSACTIONS
checkWarra... uint256 _index
claimBonus
confirmFault uint256 _index
issueRefund uint256 _index
payForDevice
reportFault uint256 _index
updateDevic... uint256 _index, uint8
VESSELTOINSTALL
_vessel: Schemess
_vesselAddress: 0x5B38Da6a701c568545dC
Calldata
Parameters
transact
withdraw
bonusClaimed address
equipment uint256
getBalance
getInstallati... address _vessel
getTotalPric...
identifyDevice uint256 _index
install uint256
Home
SCHEMESS.sol
0
Listen on all transactions
Filter with transaction hash or address

```
            install.push(Installation({
                name: _name,
                vessel: _vessel,
                vesselAddress: _vesselAddress,
                amount: Price[_name],
                status: Status.Installed,
                installedAt: block.timestamp,
                valid: true
            }));
            emit DeviceInstalled(_name, _vesselAddress);
        }

        function vesselToInstall(string memory _vessel, address _vesselAddress) public payable {    infinite gas
            require(msg.value >= getTotalPrice(), "Insufficient payment");
            require(uptimeStart[_vesselAddress] == 0, "Devices already installed for this vessel");

            for (uint i = 0; i < equipment.length; i++) {
                installDevice(equipment[i], _vessel, _vesselAddress);
            }

            uptimeStart[_vesselAddress] = block.timestamp;
        }
```

```
decoded input          {
                           "string _vessel": "Schemess",
                           "address _vesselAddress": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4"
                       }

decoded output         {}

logs                   [
                           {
                               "from": "0xC7B2776E53caAc66eB0725aF2Dd8B1F54EbFdB94",
                               "topic": "0x52782744b1e93d56bf7cc2d706bff77948c1b4500ea23f3d9320a05478c8b96e",
                               "event": "DeviceInstalled",
                               "args": {
                                   "0": "TEMP_HUM",
                                   "1": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4"
                               }
                           },
                           {
                               "from": "0xC7B2776E53caAc66eB0725aF2Dd8B1F54EbFdB94",
                               "topic": "0x52782744b1e93d56bf7cc2d706bff77948c1b4500ea23f3d9320a05478c8b96e",
                               "event": "DeviceInstalled",
>
```

2) Successfully retrieved the total cost of all three SCHEMSS components (TEMP_HUM, GAS, FLAME), which equals 12 ETH, using the getTotalPriceInEther() function

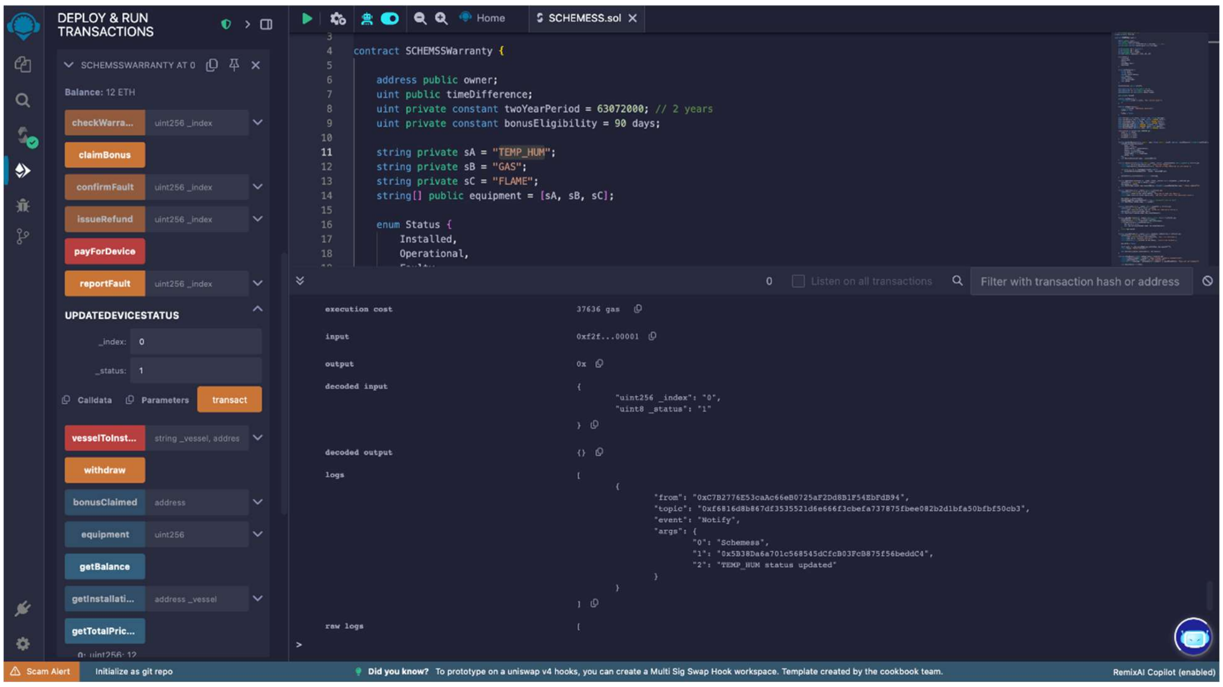


_vessel: Schemess
_vesselAddress: 0x5B38Da6a701c568545dC
0: uint256: 12

```
        }
    }

    Installation[] memory result = new Installation[](count);
    uint j = 0;
    for (uint i = 0; i < install.length; i++) {
        if (install[i].vesselAddress == _vessel) {
            result[j] = install[i];
            j++;
        }
    }
    return result;
}

function getTotalPrice() private view returns (uint256) {    infinite gas
    return Price[sA] + Price[sB] + Price[sC];
}

function getTotalPriceInEther() public view returns (uint256) {    infinite gas
    return getTotalPrice() / 1 ether;
}
```

```
to               SCHEMSSWarranty.getTotalPriceInEther() 0xC7B2776E53caAc66eB0725aF2Dd8B1F54EbFdB94
execution cost   15003 gas (Cost only applies when called by a contract)
input            0x365...18ad1
output           0x000000000000000000000000000000000000000000000000000000000000000c
decoded input    {}
decoded output   {
                     "0": "uint256: 12"
                 }
logs             []
raw logs         []
>
```

3) Executed the updateDeviceStatus() function to change the status of the first sensor (index 0) on the vessel "Schemess" from Installed to Operational, enabling it to be eligible for fault reporting as per the smart contract's logic



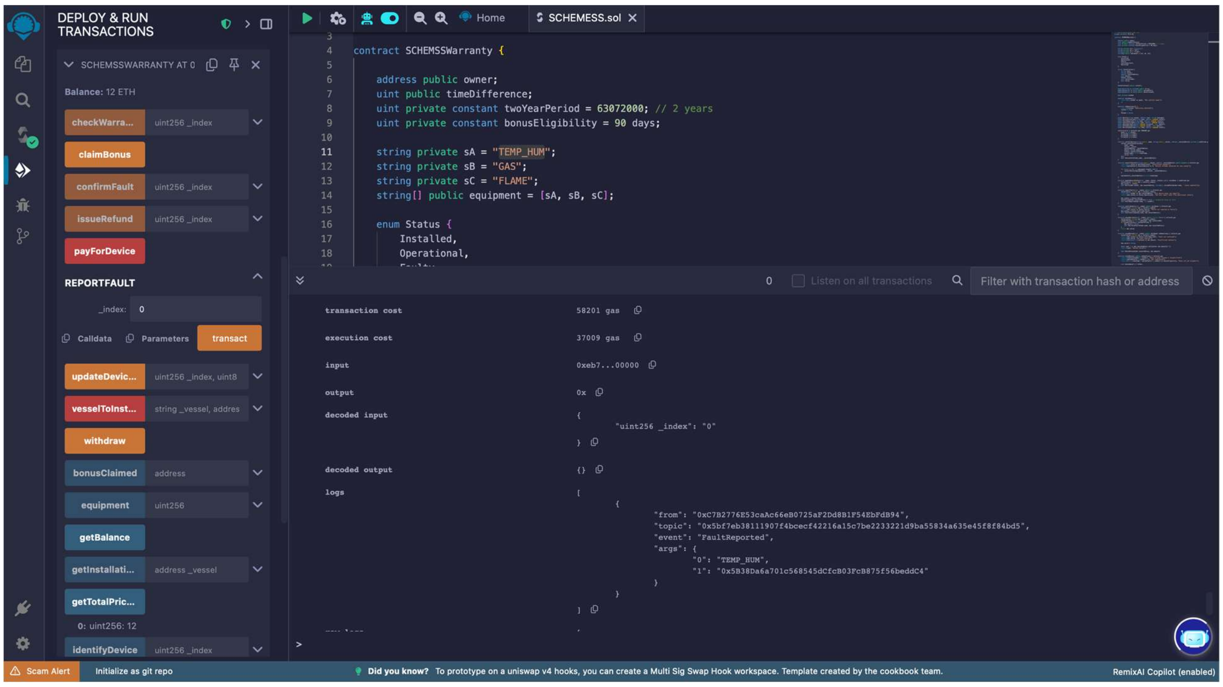
Balance: 12 ETH
_index: 0
_status: 1
0: uint256: 12

```
contract SCHEMSSWarranty {

    address public owner;
    uint public timeDifference;
    uint private constant twoYearPeriod = 63072000; // 2 years
    uint private constant bonusEligibility = 90 days;

    string private sA = "TEMP_HUM";
    string private sB = "GAS";
    string private sC = "FLAME";
    string[] public equipment = [sA, sB, sC];

    enum Status {
        Installed,
        Operational,
```

```
execution cost   37636 gas
input            0xf2f...00001
output           0x
decoded input    {
                     "uint256 _index": "0",
                     "uint8 _status": "1"
                 }
decoded output   {}
logs             [
                     {
                         "from": "0xC7B2776E53caAc66eB0725aF2Dd8B1F54EbFdB94",
                         "topic": "0xf6816d8b867df3535521d6e666f3cbefa737875fbee082b2d1bfa50bfbf50cb3",
                         "event": "Notify",
                         "args": {
                             "0": "Schemess",
                             "1": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4",
                             "2": "TEMP_HUM status updated"
                         }
                     }
                 ]
raw logs         [
>
```

Did you know? To prototype on a uniswap v4 hooks, you can create a Multi Sig Swap Hook workspace. Template created by the cookbook team.

4) Executed the reportFault() function for the temperature sensor (TEMP_HUM, index 0), which was previously updated to the Operational state, successfully triggering a transition to the Faulty state and emitting the FaultReported event


DEPLOY & RUN TRANSACTIONS
SCHEMSSWARRANTY AT 0
Balance: 12 ETH
checkWarra... uint256 _index
claimBonus
confirmFault uint256 _index
issueRefund uint256 _index
payForDevice
REPORTFAULT
_index: 0
Calldata Parameters transact
updateDevic... uint256 _index, uint8
vesselToInst... string _vessel, addres
withdraw
bonusClaimed address
equipment uint256
getBalance
getInstallati... address _vessel
getTotalPric...
0: uint256: 12
identifyDevice uint256 _index
Home
SCHEMESS.sol
Listen on all transactions
Filter with transaction hash or address
Scam Alert
Initialize as git repo
RemixAI Copilot (enabled)

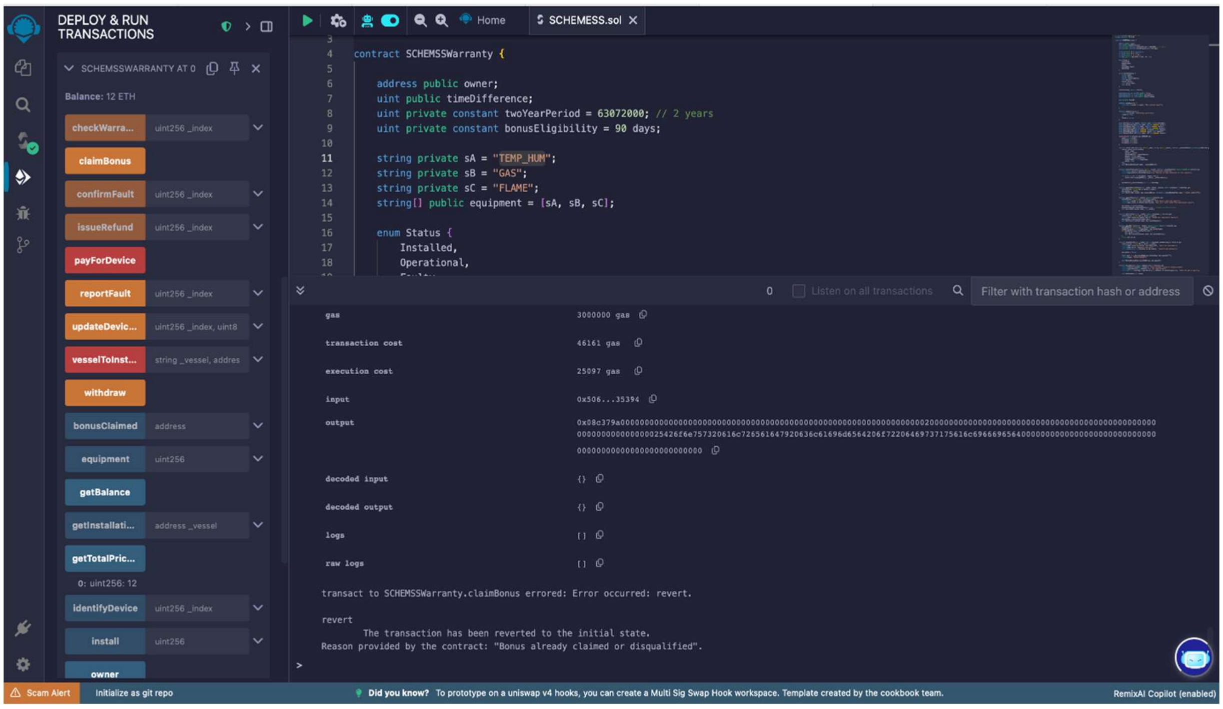
```
contract SCHEMSSWarranty {

    address public owner;
    uint public timeDifference;
    uint private constant twoYearPeriod = 63072000; // 2 years
    uint private constant bonusEligibility = 90 days;

    string private sA = "TEMP_HUM";
    string private sB = "GAS";
    string private sC = "FLAME";
    string[] public equipment = [sA, sB, sC];

    enum Status {
        Installed,
        Operational,
```

```
transaction cost        58201 gas
execution cost          37009 gas
input                   0xeb7...00000
output                  0x
decoded input           {
                                "uint256 _index": "0"
                        }
decoded output          {}
logs                    [
                                {
                                        "from": "0xC7B2776E53caAc66eB0725aF2Dd8B1F54EbFdB94",
                                        "topic": "0x5bf7eb38111907f4bcecf42216a15c7be2233221d9ba55834a635e45f8f84bd5",
                                        "event": "FaultReported",
                                        "args": {
                                                "0": "TEMP_HUM",
                                                "1": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4"
                                        }
                                }
                        ]
```

Did you know? To prototype on a uniswap v4 hooks, you can create a Multi Sig Swap Hook workspace. Template created by the cookbook team.

5) Attempted to execute the claimBonus() function, but the transaction reverted with the message "Bonus already claimed or disqualified" because a fault was reported within

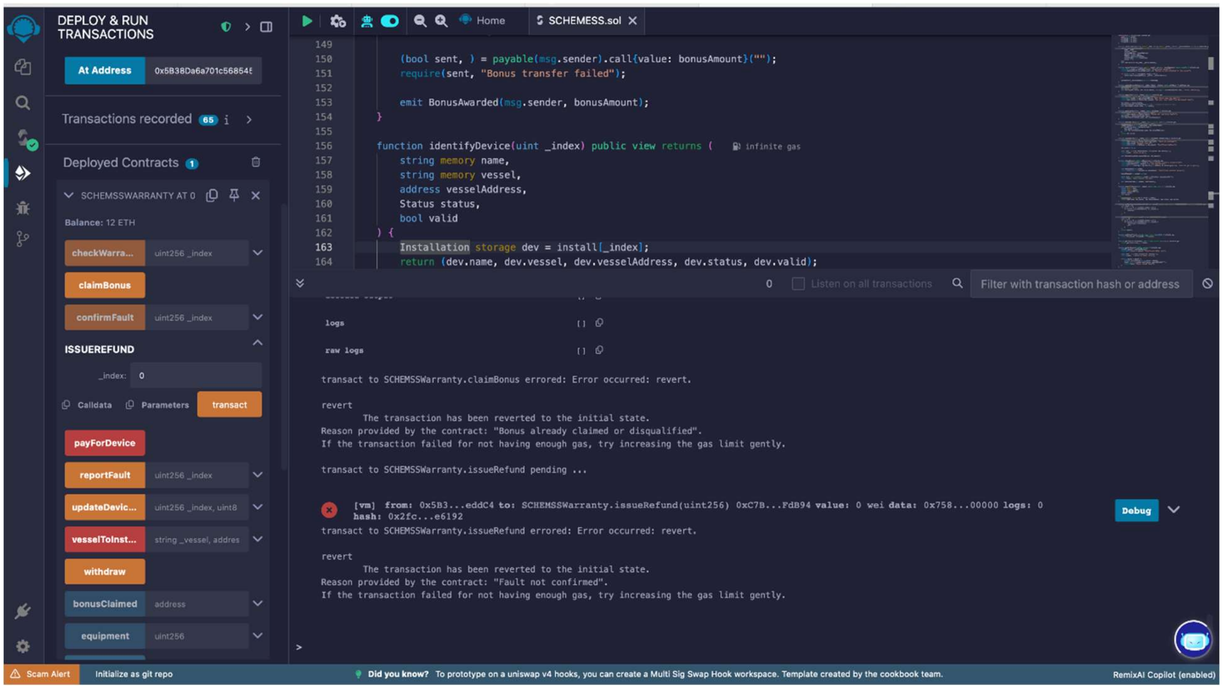
the first 90 days, which automatically disqualifies the vessel from receiving the performance bonus


DEPLOY & RUN TRANSACTIONS
SCHEMSSWARRANTY AT 0
Balance: 12 ETH
checkWarra... uint256 _index
claimBonus
confirmFault uint256 _index
issueRefund uint256 _index
payForDevice
reportFault uint256 _index
updateDevic... uint256 _index, uint8
vesselToInst... string _vessel, addres
withdraw
bonusClaimed address
equipment uint256
getBalance
getInstallati... address _vessel
getTotalPric...
0: uint256: 12
identifyDevice uint256 _index
install uint256
owner
Home
SCHEMESS.sol
Listen on all transactions
Filter with transaction hash or address
Scam Alert
Initialize as git repo
Did you know? To prototype on a uniswap v4 hooks, you can create a Multi Sig Swap Hook workspace. Template created by the cookbook team.
RemixAI Copilot (enabled)

```
contract SCHEMSSWarranty {

    address public owner;
    uint public timeDifference;
    uint private constant twoYearPeriod = 63072000; // 2 years
    uint private constant bonusEligibility = 90 days;

    string private sA = "TEMP_HUM";
    string private sB = "GAS";
    string private sC = "FLAME";
    string[] public equipment = [sA, sB, sC];

    enum Status {
        Installed,
        Operational,
```

```
gas                     3000000 gas
transaction cost        46161 gas
execution cost          25097 gas
input                   0x506...35394
output                  0x08c379a000000000000000000000000000000000000000000000000000000000000000200000000000000000000000000000000000000000000000000
                        000000000000000025426f6e757320616c726561647920636c61696d6564206f72206469737175616c69666965640000000000000000000000000000
                        0000000000000000000000000
decoded input           {}
decoded output          {}
logs                    []
raw logs                []

transact to SCHEMSSWarranty.claimBonus errored: Error occurred: revert.

revert
        The transaction has been reverted to the initial state.
Reason provided by the contract: "Bonus already claimed or disqualified".
>
```

6) Attempted to execute issueRefund() without the fault being confirmed by the contract owner, which caused the transaction to revert with the error: "Fault not confirmed". This

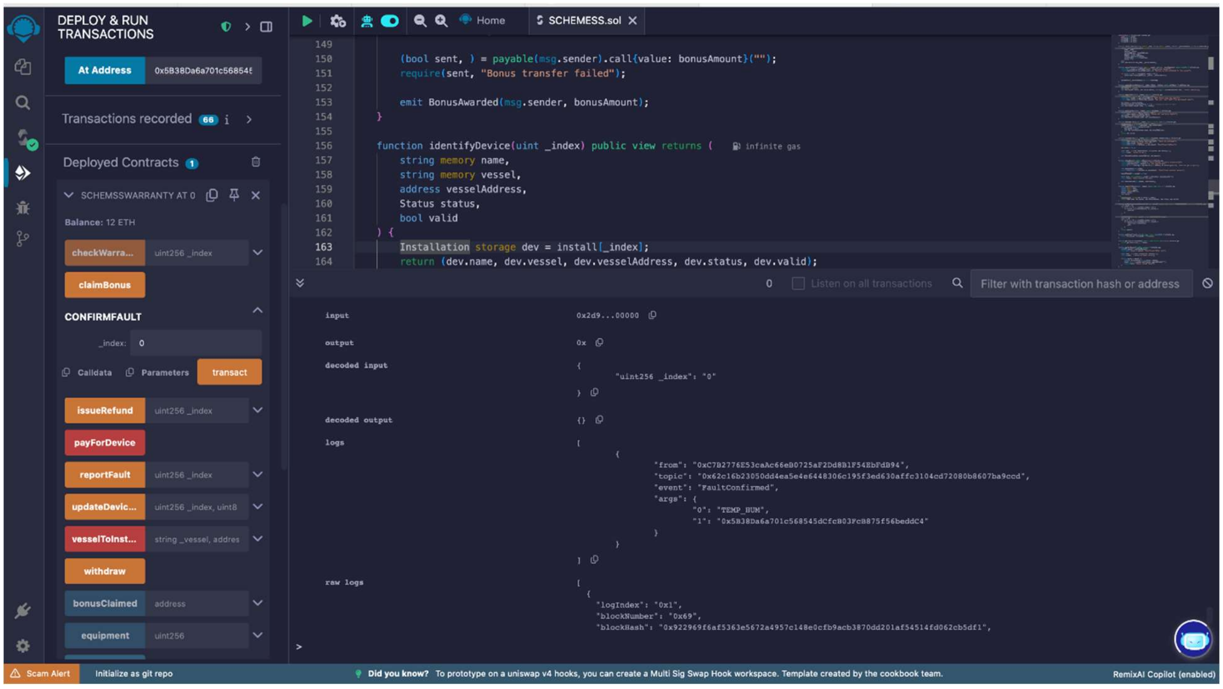

DEPLOY & RUN TRANSACTIONS
At Address 0x5B38Da6a701c56854
Transactions recorded 65
Deployed Contracts 1
SCHEMSSWARRANTY AT 0
Balance: 12 ETH
checkWarra... uint256 _index
claimBonus
confirmFault uint256 _index
ISSUEREFUND
_index: 0
Calldata
Parameters
transact
payForDevice
reportFault uint256 _index
updateDevic... uint256 _index, uint8
vesselToInst... string _vessel, addres
withdraw
bonusClaimed address
equipment uint256
Home
SCHEMESS.sol
Listen on all transactions
Filter with transaction hash or address
Debug
Scam Alert
Initialize as git repo
Did you know? To prototype on a uniswap v4 hooks, you can create a Multi Sig Swap Hook workspace. Template created by the cookbook team.
RemixAI Copilot (enabled)

```

        (bool sent, ) = payable(msg.sender).call{value: bonusAmount}("");
        require(sent, "Bonus transfer failed");

        emit BonusAwarded(msg.sender, bonusAmount);
    }

    function identifyDevice(uint _index) public view returns (    infinite gas
        string memory name,
        string memory vessel,
        address vesselAddress,
        Status status,
        bool valid
    ) {
        Installation storage dev = install[_index];
        return (dev.name, dev.vessel, dev.vesselAddress, dev.status, dev.valid);
```

```
logs                    []
raw logs                []

transact to SCHEMSSWarranty.claimBonus errored: Error occurred: revert.

revert
        The transaction has been reverted to the initial state.
Reason provided by the contract: "Bonus already claimed or disqualified".
If the transaction failed for not having enough gas, try increasing the gas limit gently.

transact to SCHEMSSWarranty.issueRefund pending ...

[vm] from: 0x5B3...eddC4 to: SCHEMSSWarranty.issueRefund(uint256) 0xC7B...FdB94 value: 0 wei data: 0x758...00000 logs: 0
hash: 0x2fc...e6192
transact to SCHEMSSWarranty.issueRefund errored: Error occurred: revert.

revert
        The transaction has been reverted to the initial state.
Reason provided by the contract: "Fault not confirmed".
If the transaction failed for not having enough gas, try increasing the gas limit gently.
>
```

safeguards the refund logic and ensures only validated fault cases are eligible for reimbursement.

7) Executed the confirmFault() function for index 0, officially verifying the reported fault for the TEMP_HUM sensor. The log confirms event "FaultConfirmed" was emitted, meaning the sensor’s status has transitioned to Confirmed_Fault, making it eligible for refund processing



At Address 0x5B38Da6a701c56854€

Balance: 12 ETH

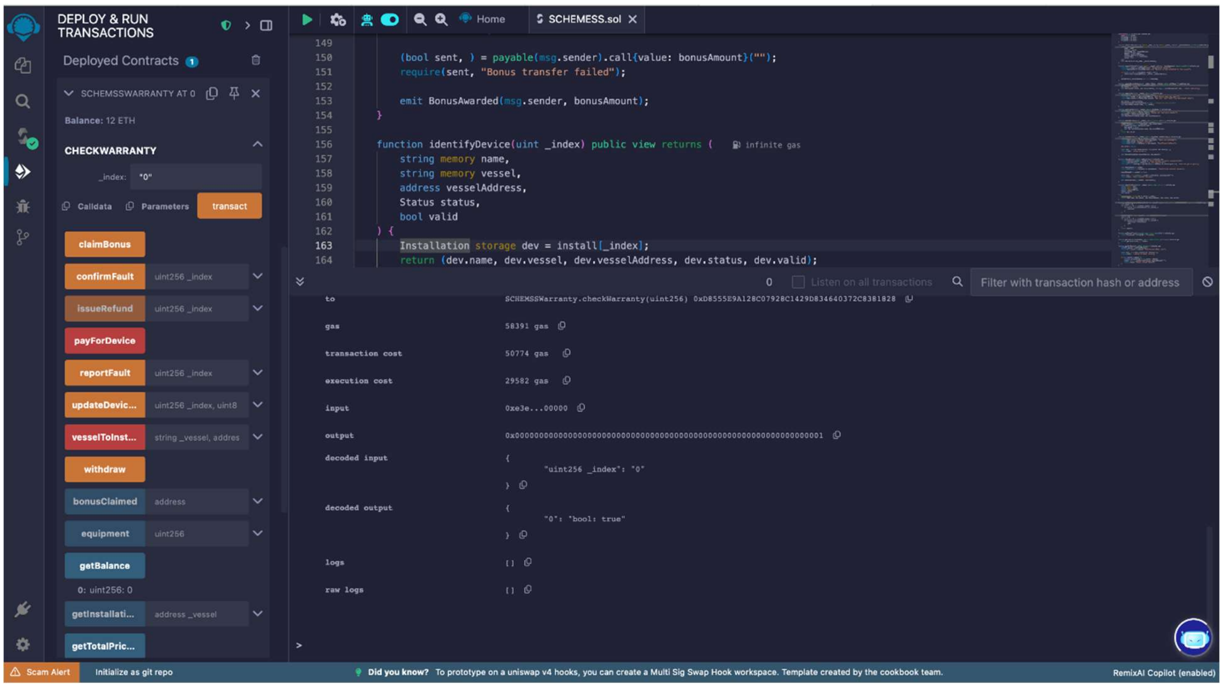
checkWarra... uint256 _index
_index: 0
issueRefund uint256 _index
reportFault uint256 _index
updateDevic... uint256 _index, uint8
vesselToInst... string _vessel, addres
bonusClaimed address
equipment uint256

```
        (bool sent, ) = payable(msg.sender).call{value: bonusAmount}("");
        require(sent, "Bonus transfer failed");

        emit BonusAwarded(msg.sender, bonusAmount);
    }

    function identifyDevice(uint _index) public view returns (    infinite gas
        string memory name,
        string memory vessel,
        address vesselAddress,
        Status status,
        bool valid
    ) {
        Installation storage dev = install[_index];
        return (dev.name, dev.vessel, dev.vesselAddress, dev.status, dev.valid);
```

```
input                          0x2d9...00000

output                         0x

decoded input                  {
                                       "uint256 _index": "0"
                               }

decoded output                 {}

logs                           [
                                       {
                                               "from": "0xC7B2776E53caAc66eB0725aF2Dd8B1F54EbFdB94",
                                               "topic": "0x62c16b23050dd4ea5e4e6448306c195f3ed630affc3104cd72080b8607ba9ccd",
                                               "event": "FaultConfirmed",
                                               "args": {
                                                       "0": "TEMP_HUM",
                                                       "1": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4"
                                               }
                                       }
                               ]

raw logs                       [
                                 {
                                   "logIndex": "0x1",
                                   "blockNumber": "0x69",
                                   "blockHash": "0x922969f6af5363e5672a4957c148e0cfb9acb3870dd201af54514fd062cb5df1",
>
```

Did you know? To prototype on a uniswap v4 hooks, you can create a Multi Sig Swap Hook workspace. Template created by the cookbook team.

8) Successfully called checkWarranty(0) to verify the warranty status of the TEMP_HUM sensor before issuing a refund. The returned value true confirms the warranty is still active (within the 2-year period).

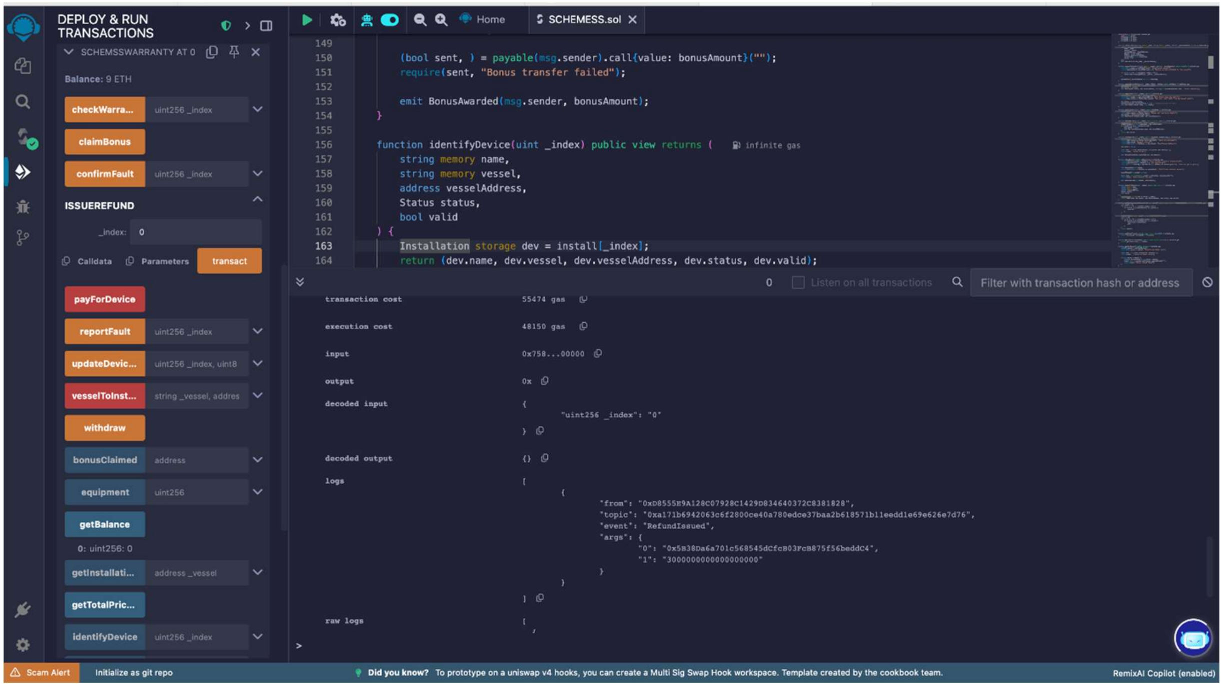


```
        (bool sent, ) = payable(msg.sender).call{value: bonusAmount}("");
        require(sent, "Bonus transfer failed");

        emit BonusAwarded(msg.sender, bonusAmount);
    }

    function identifyDevice(uint _index) public view returns (    infinite gas
        string memory name,
        string memory vessel,
        address vesselAddress,
        Status status,
        bool valid
    ) {
        Installation storage dev = install[_index];
        return (dev.name, dev.vessel, dev.vesselAddress, dev.status, dev.valid);
```

```
to                  SCHEMSSWarranty.checkWarranty(uint256) 0xD8555E9A128C07928C1429D834640372C8381828
gas                 58391 gas
transaction cost    50774 gas
execution cost      29582 gas
input               0xe3e...00000
output              0x0000000000000000000000000000000000000000000000000000000000000001
decoded input       {
                            "uint256 _index": "0"
                    }
decoded output      {
                            "0": "bool: true"
                    }
logs                []
raw logs            []
>
```

Did you know? To prototype on a uniswap v4 hooks, you can create a Multi Sig Swap Hook workspace. Template created by the cookbook team.

9) Successfully issued a refund for the confirmed faulty sensor (index 0) using the issueRefund() function

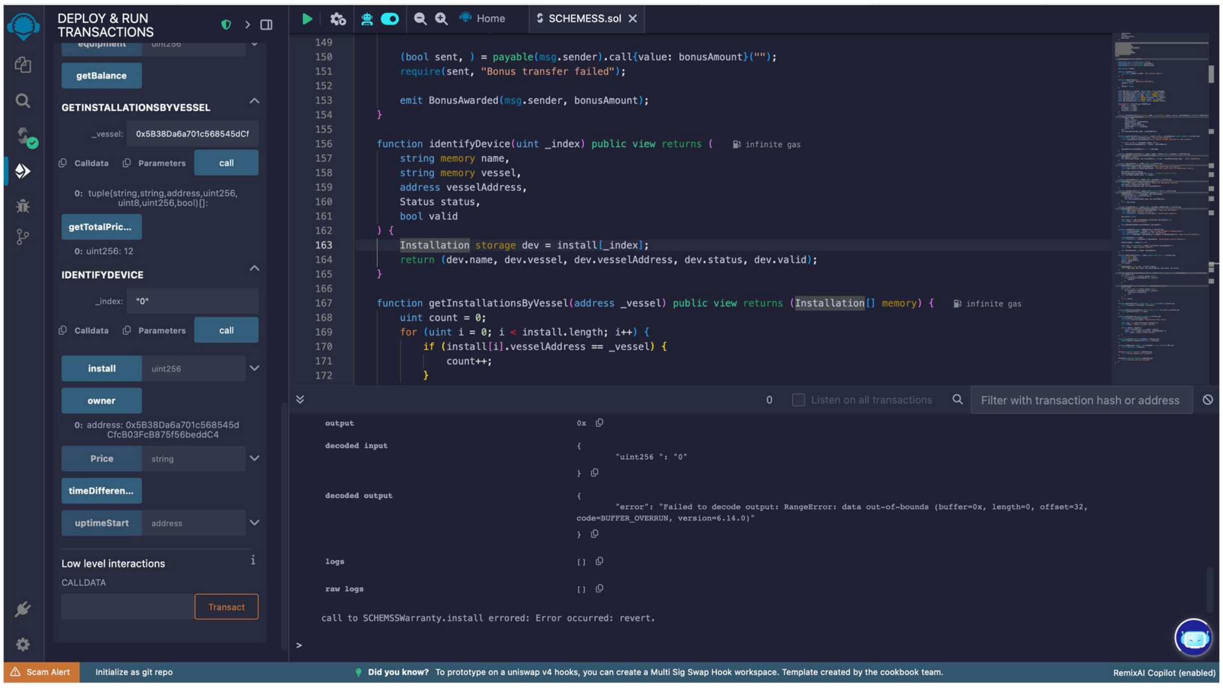


```
        (bool sent, ) = payable(msg.sender).call{value: bonusAmount}("");
        require(sent, "Bonus transfer failed");

        emit BonusAwarded(msg.sender, bonusAmount);
    }

    function identifyDevice(uint _index) public view returns (    infinite gas
        string memory name,
        string memory vessel,
        address vesselAddress,
        Status status,
        bool valid
    ) {
        Installation storage dev = install[_index];
        return (dev.name, dev.vessel, dev.vesselAddress, dev.status, dev.valid);
```

```
transaction cost    55474 gas
execution cost      48150 gas
input               0x758...00000
output              0x
decoded input       {
                            "uint256 _index": "0"
                    }
decoded output      {}
logs                [
                            {
                                    "from": "0xD8555E9A128C07928C1429D834640372C8381828",
                                    "topic": "0xa171b6942063c6f2800ce40a780edce37baa2b618571b11eedd1e69e626e7d76",
                                    "event": "RefundIssued",
                                    "args": {
                                            "0": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4",
                                            "1": "3000000000000000000"
                                    }
                            }
                    ]
raw logs            [
                      ,
>
```

Did you know? To prototype on a uniswap v4 hooks, you can create a Multi Sig Swap Hook workspace. Template created by the cookbook team.

10) Attempted to call the getInstallationsByVessel() function using an address that had no sensors installed



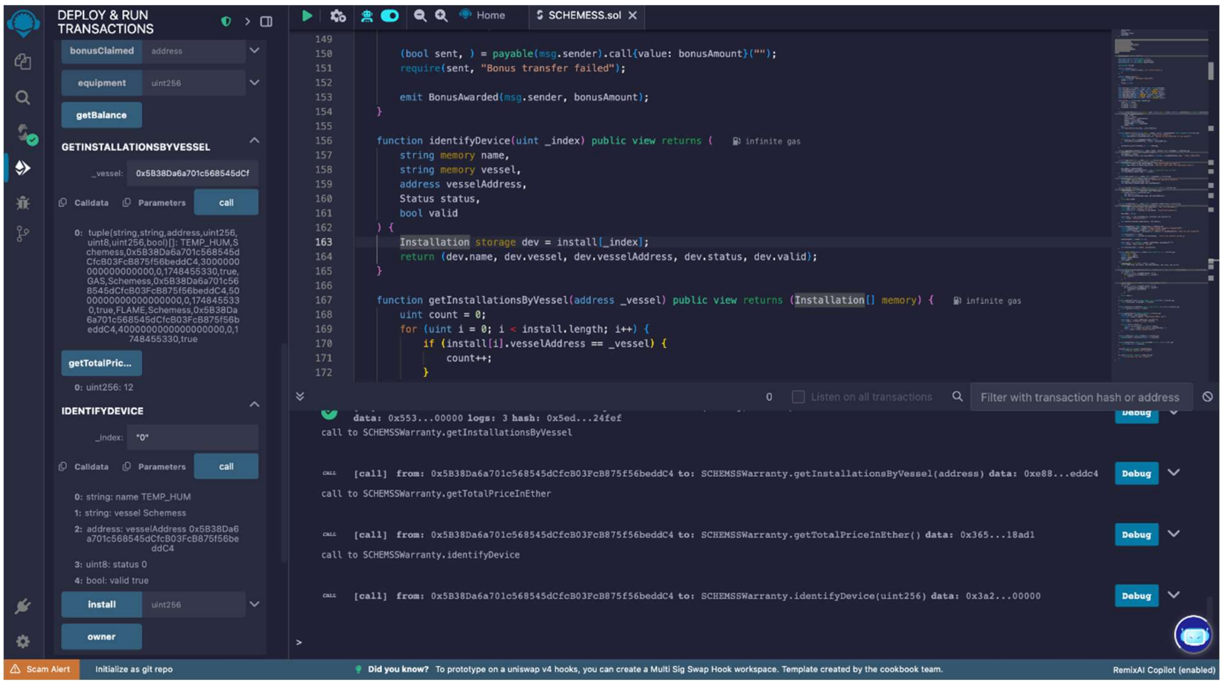
```
_vessel: 0x5B38Da6a701c568545dCf
0: tuple(string,string,address,uint256,uint8,uint256,bool)[]:
0: uint256: 12
_index: "0"
0: address: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
```

```
        (bool sent, ) = payable(msg.sender).call{value: bonusAmount}("");
        require(sent, "Bonus transfer failed");

        emit BonusAwarded(msg.sender, bonusAmount);
    }

    function identifyDevice(uint _index) public view returns (    infinite gas
        string memory name,
        string memory vessel,
        address vesselAddress,
        Status status,
        bool valid
    ) {
        Installation storage dev = install[_index];
        return (dev.name, dev.vessel, dev.vesselAddress, dev.status, dev.valid);
    }

    function getInstallationsByVessel(address _vessel) public view returns (Installation[] memory) {    infinite gas
        uint count = 0;
        for (uint i = 0; i < install.length; i++) {
            if (install[i].vesselAddress == _vessel) {
                count++;
            }
```

```
output                    0x
decoded input             {
                                  "uint256 ": "0"
                          }
decoded output            {
                                  "error": "Failed to decode output: RangeError: data out-of-bounds (buffer=0x, length=0, offset=32,
                          code=BUFFER_OVERRUN, version=6.14.0)"
                          }
logs                      []
raw logs                  []
call to SCHEMSSWarranty.install errored: Error occurred: revert.
>
```

Did you know? To prototype on a uniswap v4 hooks, you can create a Multi Sig Swap Hook workspace. Template created by the cookbook team.

11) Accessed all relevant sensor records stored on-chain



```
_vessel: 0x5B38Da6a701c568545dCf
0: tuple(string,string,address,uint256,uint8,uint256,bool)[]: TEMP_HUM,Schemess,0x5B38Da6a701c568545dCfcB03FcB875f56beddC4,3000000000000000000,0,1748455330,true,GAS,Schemess,0x5B38Da6a701c568545dCfcB03FcB875f56beddC4,5000000000000000000,0,1748455330,true,FLAME,Schemess,0x5B38Da6a701c568545dCfcB03FcB875f56beddC4,4000000000000000000,0,1748455330,true
0: uint256: 12
_index: "0"
0: string: name TEMP_HUM
1: string: vessel Schemess
2: address: vesselAddress 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
3: uint8: status 0
4: bool: valid true
```

```
        (bool sent, ) = payable(msg.sender).call{value: bonusAmount}("");
        require(sent, "Bonus transfer failed");

        emit BonusAwarded(msg.sender, bonusAmount);
    }

    function identifyDevice(uint _index) public view returns (    infinite gas
        string memory name,
        string memory vessel,
        address vesselAddress,
        Status status,
        bool valid
    ) {
        Installation storage dev = install[_index];
        return (dev.name, dev.vessel, dev.vesselAddress, dev.status, dev.valid);
    }

    function getInstallationsByVessel(address _vessel) public view returns (Installation[] memory) {    infinite gas
        uint count = 0;
        for (uint i = 0; i < install.length; i++) {
            if (install[i].vesselAddress == _vessel) {
                count++;
            }
```

```
    data: 0x553...00000 logs: 3 hash: 0x5ed...24fef
call to SCHEMSSWarranty.getInstallationsByVessel

CALL  [call]  from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: SCHEMSSWarranty.getInstallationsByVessel(address) data: 0xe88...eddc4
call to SCHEMSSWarranty.getTotalPriceInEther

CALL  [call]  from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: SCHEMSSWarranty.getTotalPriceInEther() data: 0x365...18ad1
call to SCHEMSSWarranty.identifyDevice

CALL  [call]  from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: SCHEMSSWarranty.identifyDevice(uint256) data: 0x3a2...00000

>
```

Did you know? To prototype on a uniswap v4 hooks, you can create a Multi Sig Swap Hook workspace. Template created by the cookbook team.

4. Conclusion

4.1 Conclusion and Lessons Learnt

The SCHEMSS project delivered an autonomous cargo hold safety system that merges real-time environmental sensing, cloud-based monitoring, and smart contract automation to address critical maritime hazards. Through precise integration of embedded hardware and cloud communication, we enabled early detection of temperature, humidity spikes, toxic gases, and fire—without human intervention. The smart contract, developed in Solidity, enforces transparent warranty claims and performance incentives, reducing disputes and ensuring operational accountability. Testing under real conditions revealed edge cases that transformed our system logic and proved the importance of validating beyond simulation.

We learned that building reliable Industrial Cyber-Physical Systems demands coordination at every layer—hardware, software, network, and logic. Solidity development taught us to think in deterministic rules, where trust is coded and every state transition must be secure and auditable. Real-time safety systems must be proactive and redundant, especially in unmanned, high-risk spaces. This project has sparked our interest and serves as a clear stepping stone into the world of blockchain and automation. Most importantly, we realized that safety is not just a technical objective—it is an ethical obligation. With SCHEMSS, we designed not just a system, but a responsibility.

4.2 Future Developments

Building on the success of SCHEMSS, future enhancements will focus on expanding both the functionality and scalability of the system. One of the primary directions is improving communication robustness and system modularity. We recommend implementing UART (Universal Asynchronous Receiver/Transmitter) communication to establish a reliable serial link between the Raspberry Pi and ESP32 microcontroller. This two-stage architecture will allow the Raspberry Pi to act as a local decision-making hub, collecting, processing, and logging sensor data from the ESP32 in real time, while also managing data relays to the cloud or blockchain.

The UART interface offers several benefits for industrial environments, including asynchronous bi-directional data transfer, ease of debugging, and low hardware overhead. It also allows future firmware upgrades or sensor commands to be issued dynamically from a user dashboard running on the Raspberry Pi, further improving maintainability and remote control. This separation of duties between data acquisition (ESP32) and supervisory control (Raspberry Pi) would enhance the flexibility, error handling, and resilience of the system.

Additional improvements under consideration include:

Predictive Maintenance Integration: Leveraging machine learning algorithms to analyze historical sensor data for anomaly detection and early failure prediction.

Scalability for Fleet Deployment: Enabling support for multiple vessels and bulk sensor installations from a unified dashboard, managed via cloud or decentralized interfaces.