

# Intelligent Stability System for Passenger Seats in Vessels

Group members:

Muhammad Omer Farooq Zahid

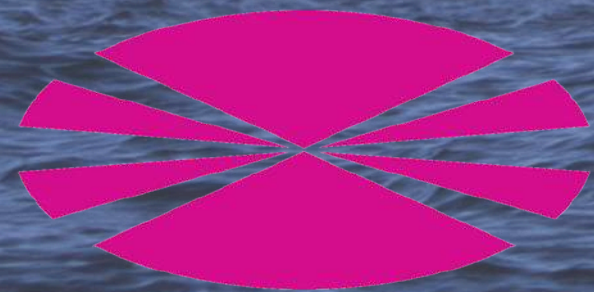
Milad Jalilzadehkhatouni

Urfan Aghayev

Professor:

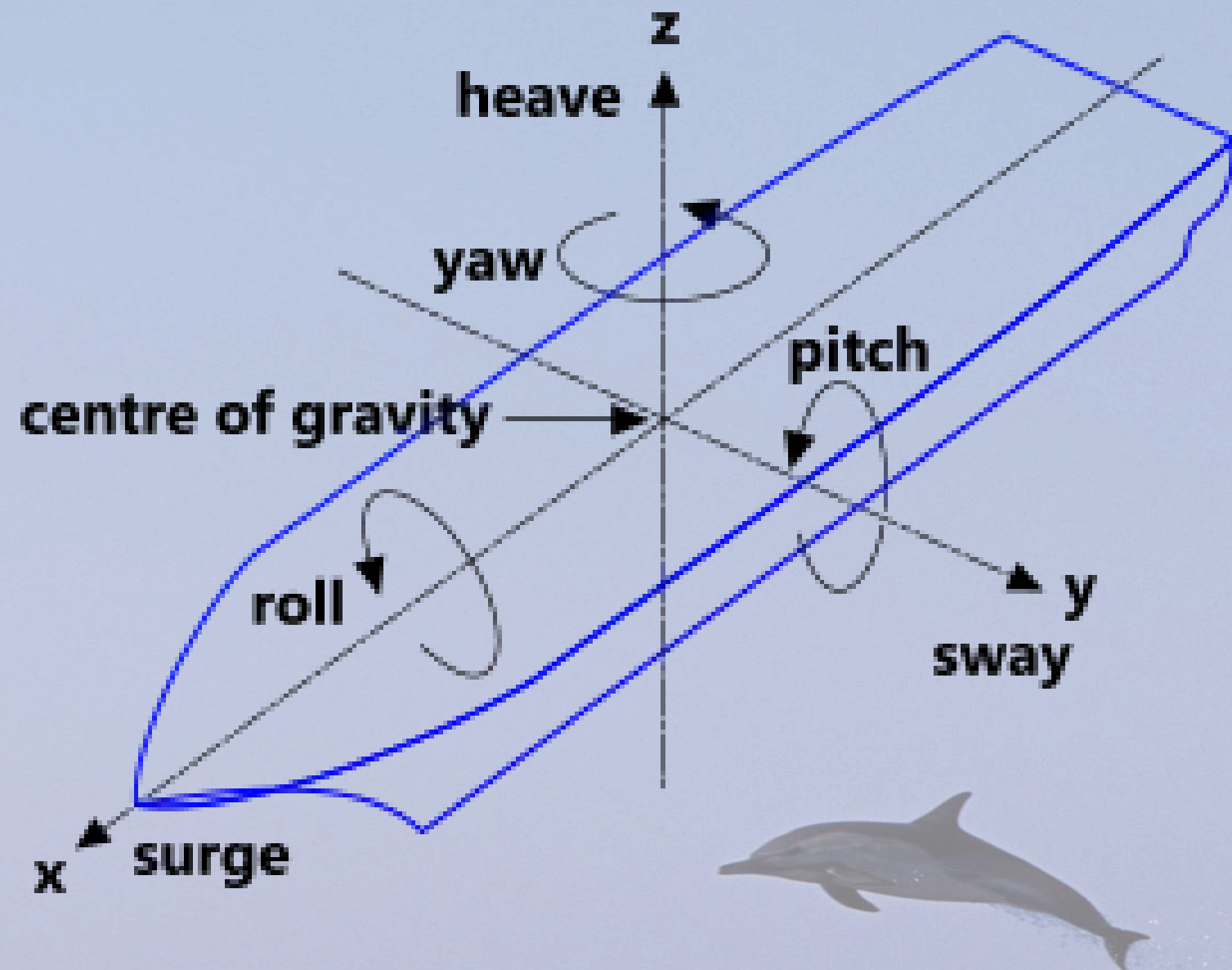
Paula Fraga Lamas

Tiago Manuel Fernández Caramés





# The negative effects of vessel motions for passengers



The motions in the ships such as rolling, pitching and yawing are known for the bad effects and discomfort on passengers. They create difficulty to move, sleep, cause seasickness and even injuries.

These unpleasant motions are the main reason that many people refuse to travel by sea





The second main issue at sea is the unfavourable weather conditions such as temperature and high humidity

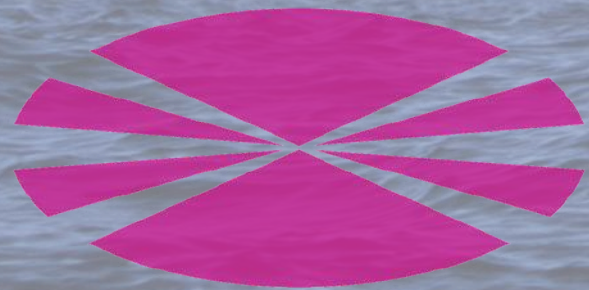
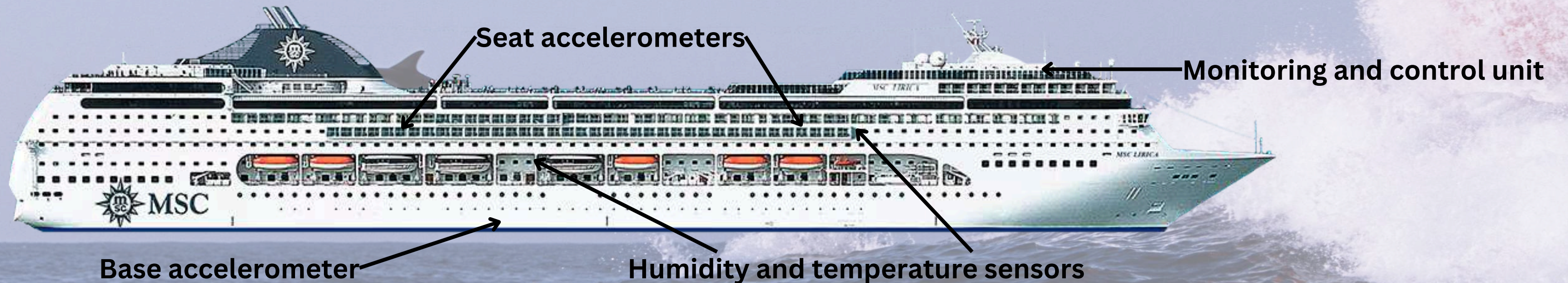
Usually the high humidity and temperature is observed at sea throughout the whole voyage which is the main reason of physical discomfort and reduced well-being on passengers as well as on crew





# The main purpose of our system

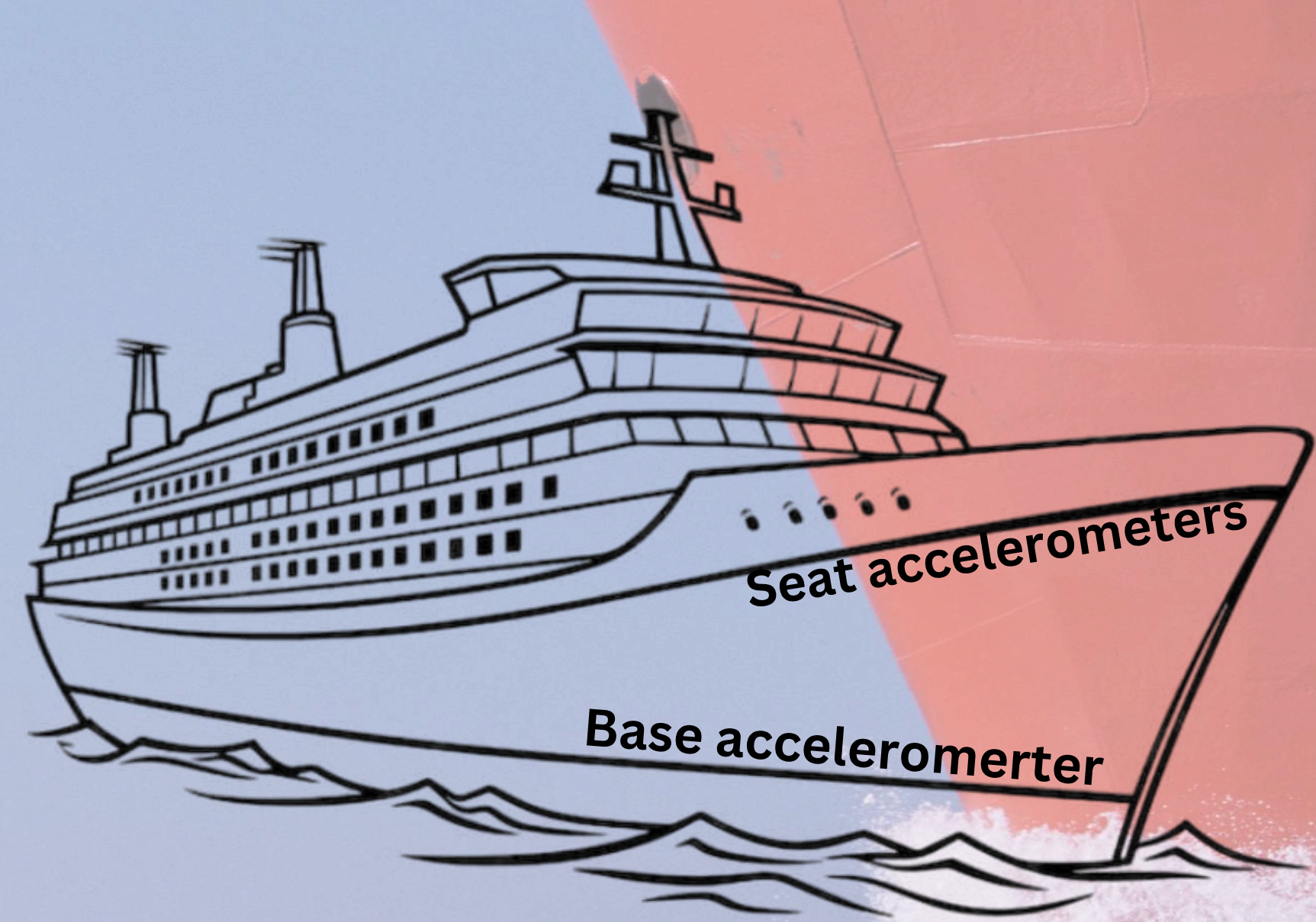
Our system aims at preventing or reducing negative effects of these phenomena by helping passengers to monitor and minimize these excessive values with the help of sensors and actuators





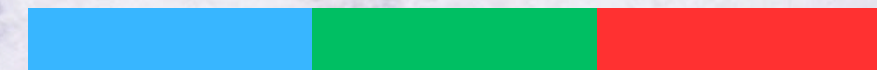
# Acceptable values

The difference between base and seat accelerometer detects how much ship is rolling or pitching. By determining threshold value the crew can monitor and prevent motions which exceed the acceptable value. We determined range between 1.5G and 3G as acceptable acceleration. Values beyond that is considered unacceptable and should be prevented. In our arduino system this values are indicated by both buzzer and RGB led.



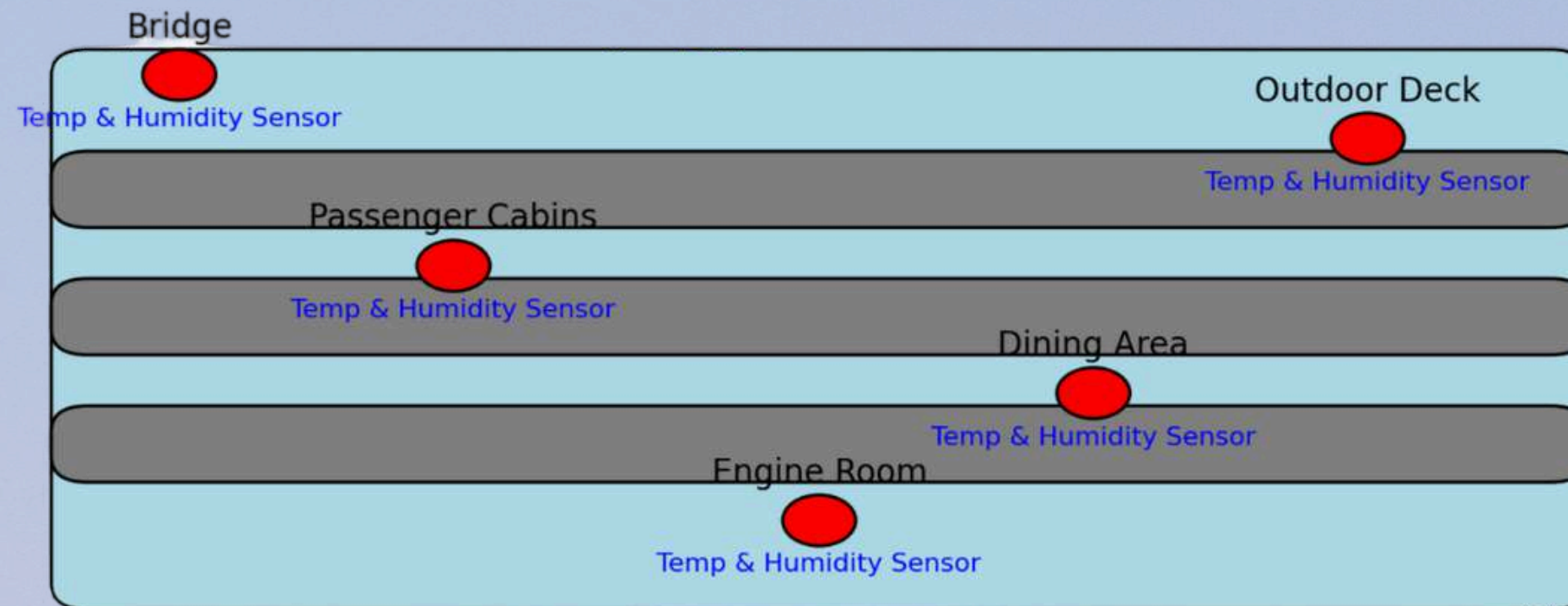
RGB light color and buzzer warning according to acceleration:

$a < 1.5G$      $1.5G - 3G$      $a > 3G$

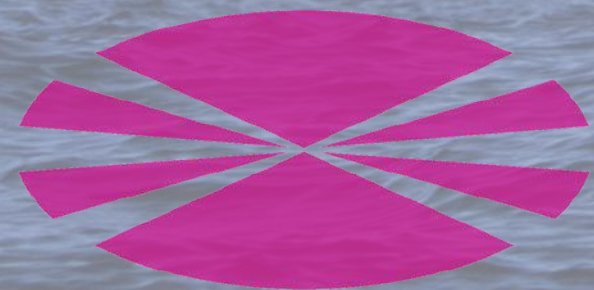
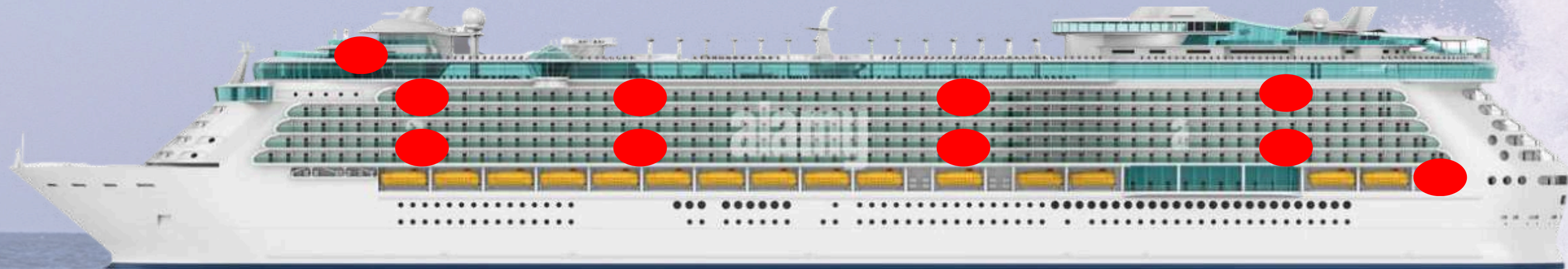




# Temperature and Humidity



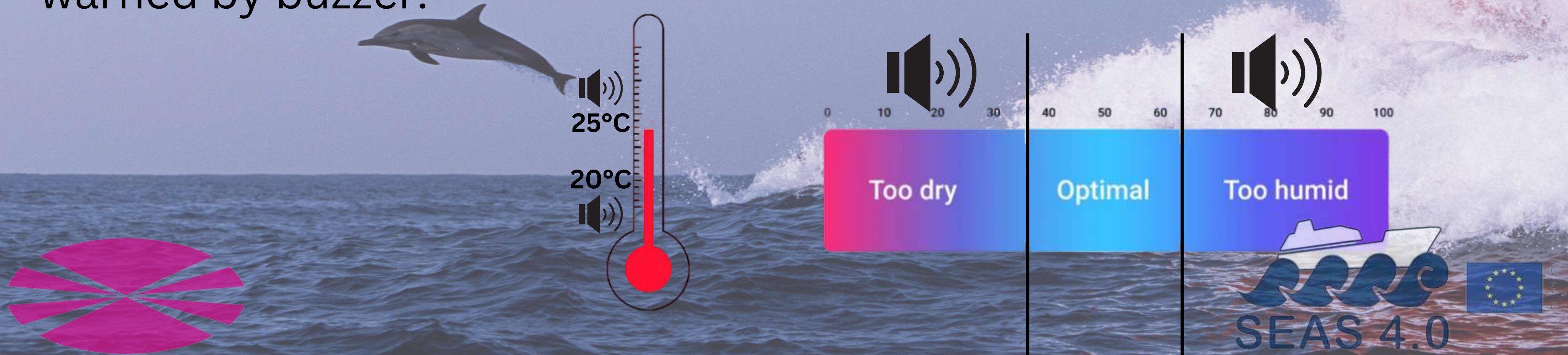
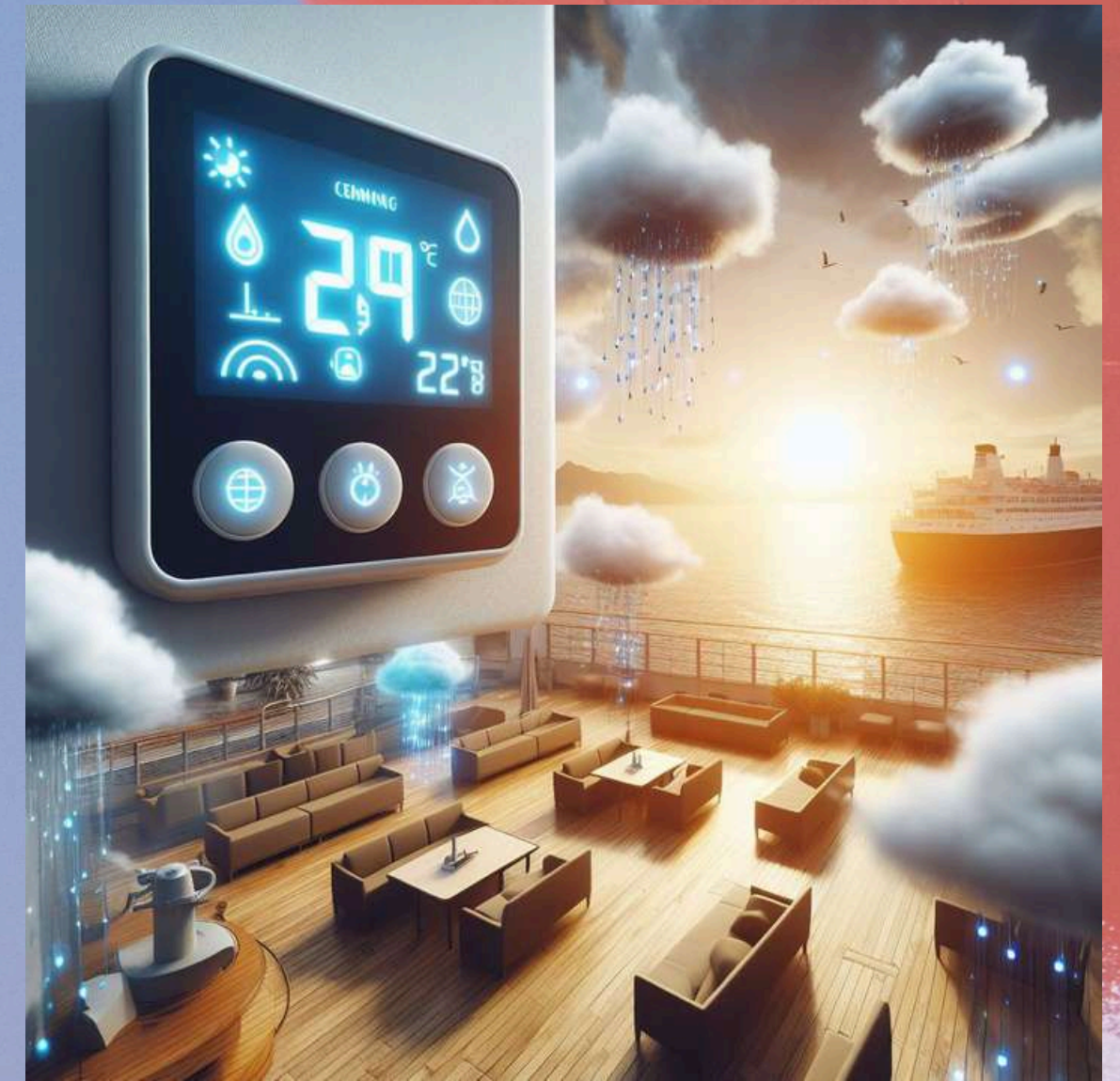
Temperature and humidity are other factors needed to be controlled in passenger ships. Our system also focuses on the favourable temperature and humidity values in all cabins of the passenger ship to provide the highest comfort.





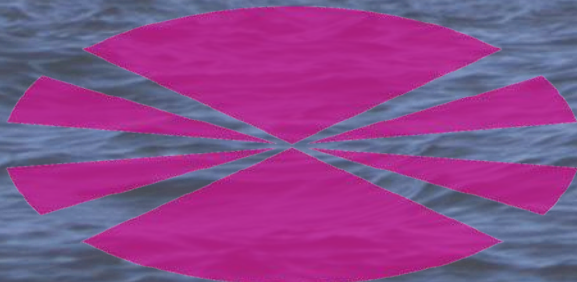
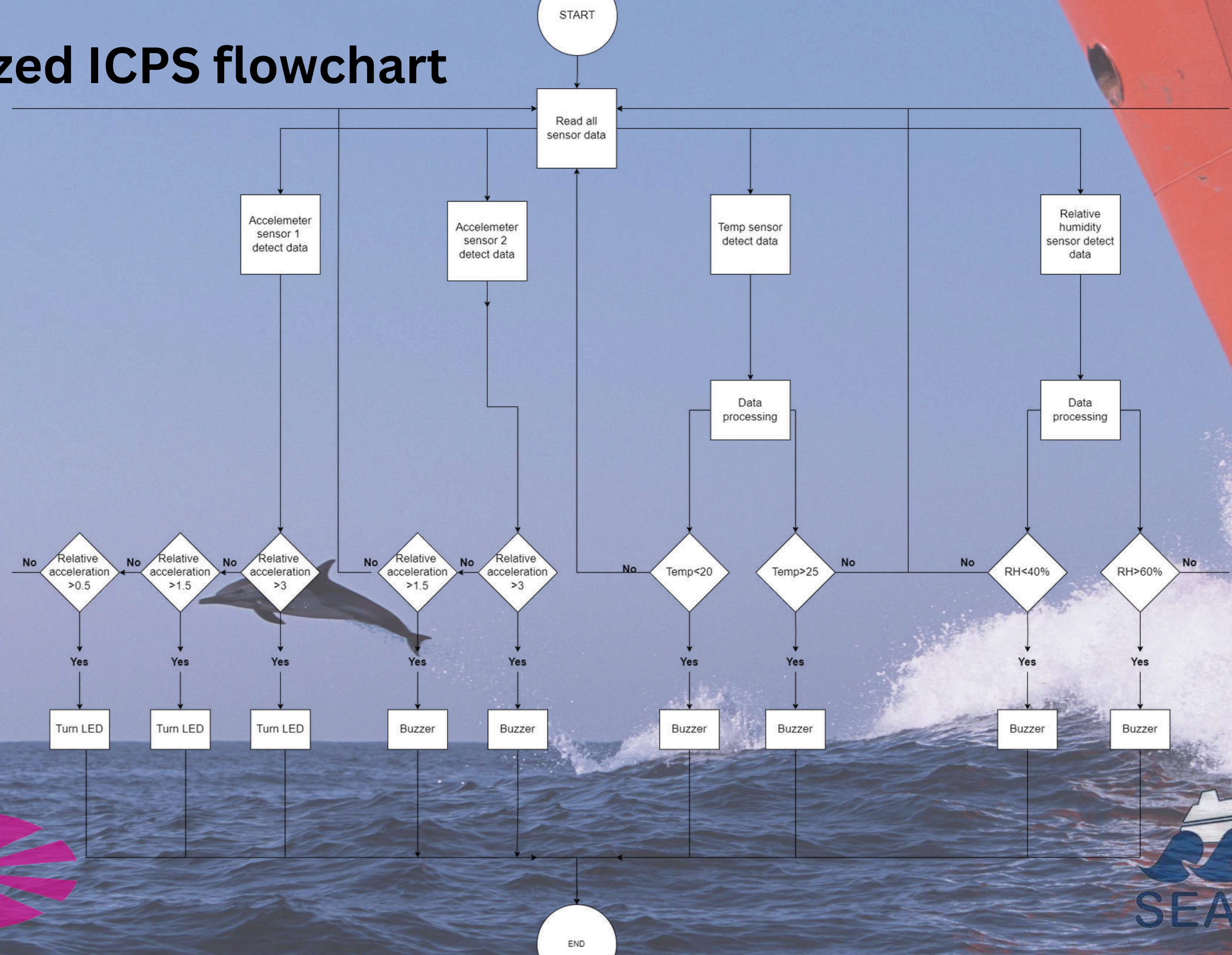
For humidity sensor we determined the range between 40% and 60% relative humidity and for temperature sensor range between 20°C and 25°C as comfortable condition for all passengers.

In our system values beyond that are warned by buzzer:



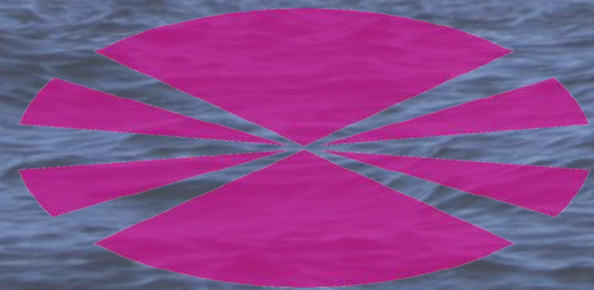
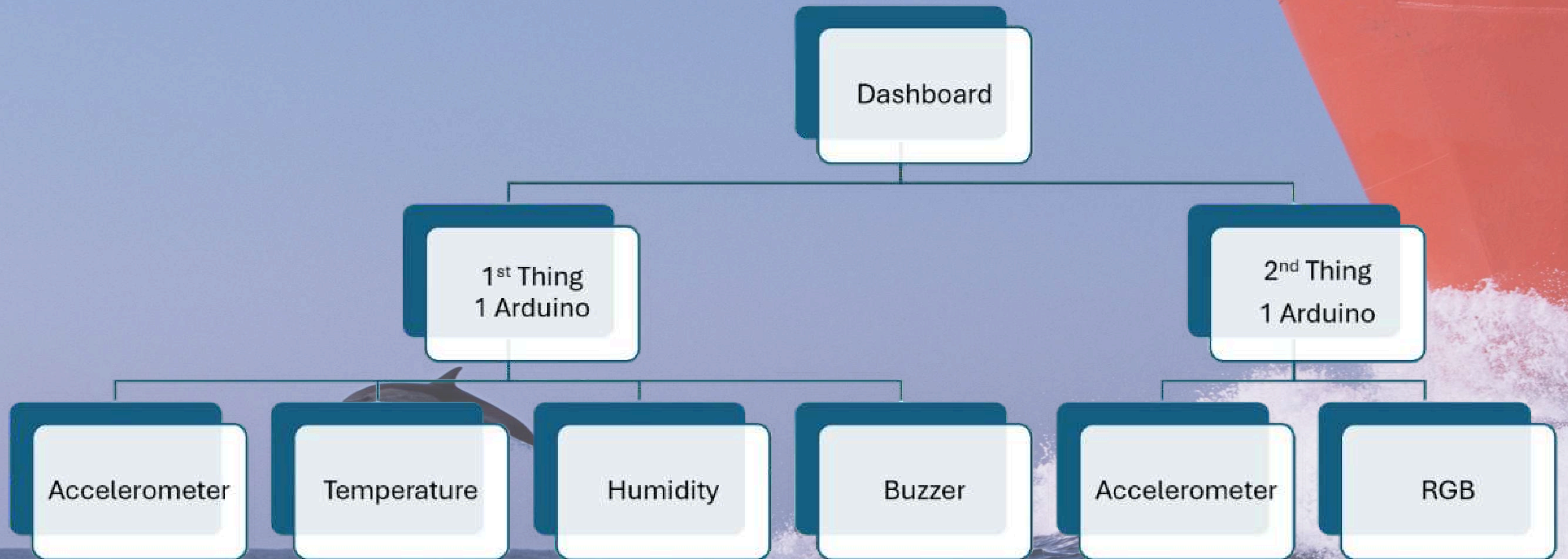


# Finalized ICPS flowchart





# Complete overview of our Ardunio system





# Arduino Cloud Dashboard

Temperature

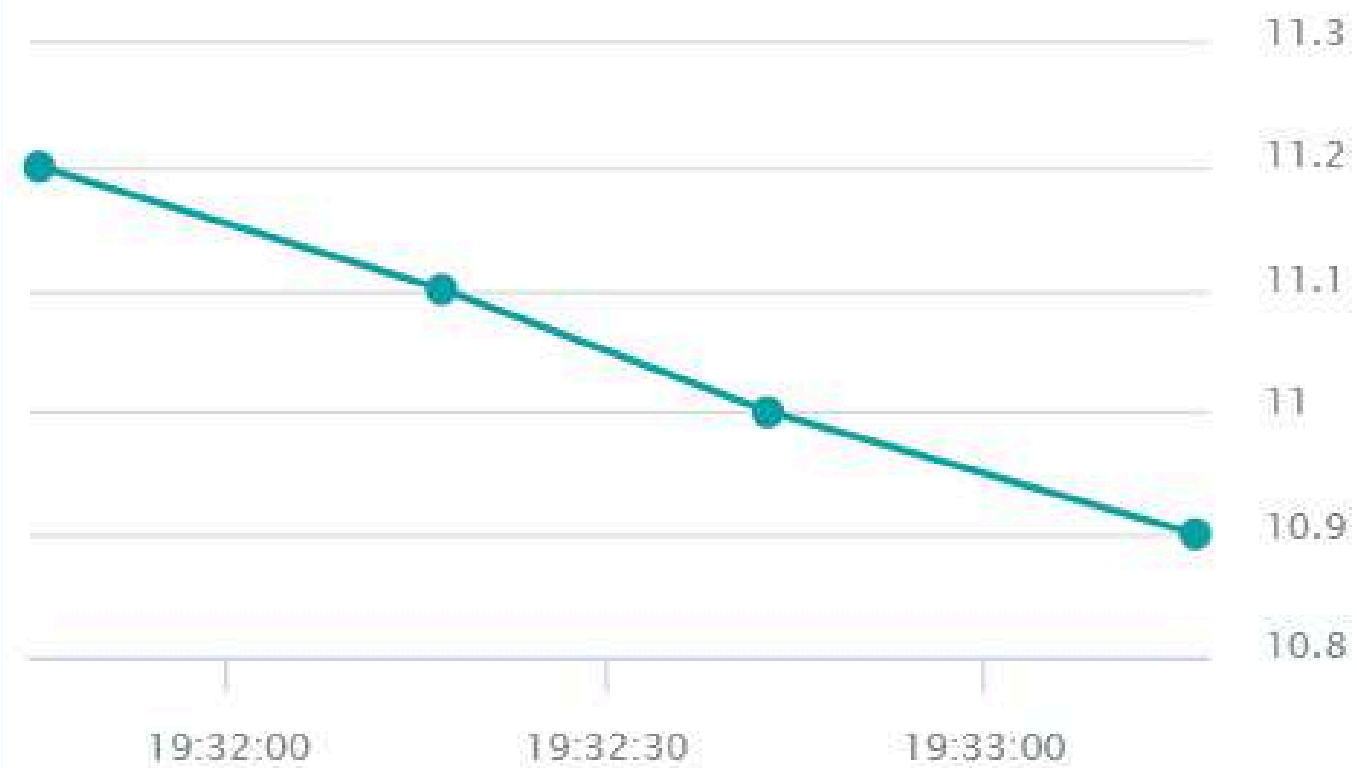
15 D

7 D

1 D

1 H

LIVE



Percentage

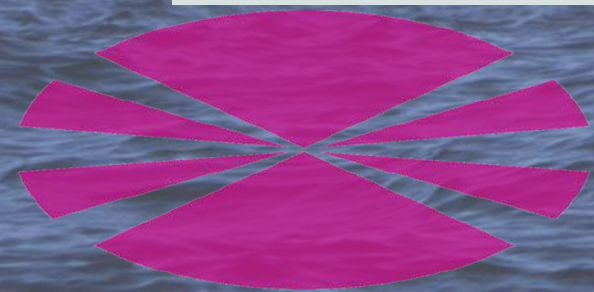


Accelerometer\_ship

0.8

Accelerometer\_seat

1.8





# Blockchain smart contract

In this system we will provide an experience similar to in flight entertainment systems found on airplanes, where passengers can control various aspects of their immediate environment. Functions of this system are



## Passenger Comfort and Environment:

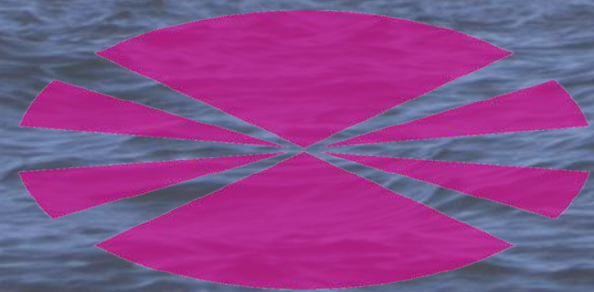
Login with Name and Travel ID  
Adjust Temperature, Humidity

## Motion Damping Feature Activation:

Passengers have option to activate the damped seat feature using tokens allocated

## Owner Management Control:

Create or delete a customer login credentials  
Manage Temperature, Humidity, Activate Damping  
Perform all the functions without needing tokens





# Blockchain smart contract

The image shows the Remix IDE interface with a Solidity smart contract named `FerryService` and the deployment transaction settings.

**Left Panel (ENVIRONMENT & DEPLOY):**

- ENVIRONMENT:** Remix VM (Cancun)
- ACCOUNT:** 0x5B3...eddC4 (84.999999999885231)
- GAS LIMIT:** Estimated Gas (3000000)
- VALUE:** 5 Ether
- CONTRACT:** FerryService - Project.sol
- DEPLOY:** CUSTOMERACCOUNT: 0xA8483f64d9C6d1EcF9b84
- Buttons:** Calldata, Parameters, **transact** (highlighted), Publish to IPFS
- Transactions recorded:** 160
- Pinned Contracts:** No pinned contracts found for selected workspace & network

**Right Panel (Code Editor):**

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
3
4 contract FerryService {
5     struct Customer {
6         string fullName;
7         bool firstTime;
8         uint256 tokenBalance;
9     }
10
11     mapping(uint256 => Customer) private customers;
12     mapping(address => uint256) private loggedInUsers;
13     uint256 private ownerTravelId;
14
15     string public freeWifiCode = "SARRAY_MIL";
16
17     // Variables for temperature and humidity
18     int private temperature;
19     uint private humidity;
20
21     bool private dampedSeatActive;
22
23     address private constant companyAccount = 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4;
24     address private customerAccount;
25
26     // array of jokes
27     string[] private jokes = [
28         "Why don't skeletons fight each other? They don't have the guts.",
29         "Why did the scarecrow become a successful neurosurgeon? He was outstanding in his field.",
30         "What did the janitor say when he jumped out of the closet? Supplies!"
31     ];
32
33     constructor(address _customerAccount) payable {
34         ownerTravelId = 637042012; // Omer Farooq's travel ID
35         customerAccount = _customerAccount;
36
37         // Pre-loading customer database
38         customers[69039] = Customer("Milad", true, 0);
39     }
40 }
```

**Bottom Panel (Footer):**

- 0 Listen on all transactions
- Filter with transaction hash or address



# Blockchain smart contract

FERRYSERVICE AT 0X78C...3E723

Balance: 3 ETH

activateDamp...

addCustomer

uint256 \_newTravelId, string

login

string \_fullName, uint256 \_

logout

rechargeTokens

uint256 \_tokenAmount

removeCusto...

uint256 \_travelId

setHumidity

uint256 \_newHumidity

setTemperature

int256 \_newTemperature

freeWifiCode

getHumidity

getTemperatu...

getTokenBala...

isDampedSea...

tellJoke

```
7      bool firstTime;
8      uint256 tokenBalance;
9  }
10
11  mapping(uint256 => Customer) private customers;
12  mapping(address => uint256) private loggedInUsers;
13  uint256 private ownerTravelId;
14
15  string public freeWifiCode = "SARRAY_MIL";
16
17  // Variables for temperature and humidity
18  int private temperature;
19  uint private humidity;
20
21  bool private dampedSeatActive;
22
23  address private constant companyAccount = 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4;
24  address private customerAccount;
25
26  // Jokes array
27  string[] private jokes = [
28      "Why don't skeletons fight each other? They don't have the guts.",
29      "Why did the scarecrow become a successful neurosurgeon? He was outstanding in his field.",
30      "What did the janitor say when he jumped out of the closet? Supplies!"
31  ];
32
33  constructor(address _customerAccount) payable {
34      ownerTravelId = 637042012; // Omer Farooq's travel ID
35      customerAccount = _customerAccount;
36
37      // Pre-loading customer database
38      customers[69039] = Customer("Milad", true, 0);
```

0

☐ Listen on all transactions

Q



# Blockchain smart contract

addCustomer

\_newTravelId: 4949

\_fullName: Paula

Calldata Parameters **transact**

login

\_fullName: Omer Farooq

\_travelId: 637042012

Calldata Parameters **transact**

logout

rechargeTokens

uint256\_tokenAmount

removeCustomer

travelId: 39069

Calldata Parameters **transact**

setHumidity

uint256\_newHumidity

setTemperature

int256\_newTemperature

freeWifiCode

getHumidity

getTemperatu...

getTokenBala...

isDampedSea...

tellJoke

Low level interactions

1 // SPDX-License-Identifier: MIT

2 pragma solidity ^0.8.0;

3

4 contract FerryService {

5 struct Customer {

6 string fullName;

7 bool firstTime;

8 uint256 tokenBalance;

9 }

10

11 mapping(uint256 => Customer) private customers;

12 mapping(address => uint256) private loggedInUsers;

13 uint256 private ownerTravelId;

14

15 public freeWifiCode = "SARRAY\_MIL";

16

17 // Variables for temperature and humidity

18 int private temperature;

19 uint private humidity;

20

21 bool private dampedSeatActive;

22

23 address private constant companyAccount = 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4;

24 address private customerAccount;

25

26 // Jokes array

27 string[] private jokes = [

28 "Why don't skeletons fight each other? They don't have the guts.",

29 "Why did the scarecrow become a successful neurosurgeon? He was outstanding in his field.",

30 "What did the janitor say when he jumped out of the closet? Supplies!"

31 ];

32

33 constructor(address \_customerAccount) payable {

34 ownerTravelId = 637042012; // Omer Farooq's travel ID

35 customerAccount = \_customerAccount;

36

37 // Pre-loading customer database

38 customers[69039] = Customer("Milad", true, 0);

0 Listen on all transactions

Filter with transaction hash or address

✓ [vm] from: 0x5B3...eddC4 to: FerryService.addCustomer(uint256,string) 0x845...Ce450 value: 0 wei data: 0x877...00000 logs: 0 hash: 0xb46...0c676

transact to FerryService.removeCustomer pending ...

✓ [vm] from: 0x5B3...eddC4 to: FerryService.removeCustomer(uint256) 0x845...Ce450 value: 0 wei data: 0x4c3...0989d logs: 0 hash: 0x7bf...abb5d



# Blockchain smart contract

**addCustomer**

newTravelId: 4949

fullName: Paula

Calldata Parameters transact

**login**

fullName: Urfan

travelId: 39069

Calldata Parameters transact

**logout**

**rechargeTokens** uint256\_tokenAmount

**removeCustomer**

travelId: 39069

Calldata Parameters transact

**setHumidity** uint256\_newHumidity

**setTemperature** int256\_newTemperature

**freeWifiCode**

**getHumidity**

**getTemperatu...**

**getTokenBala...**

**isDampedSea...**

**tellJoke**

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
3
4 contract FerryService {
5     struct Customer {
6         string fullName;
7         bool firstTime;
8         uint256 tokenBalance;
9     }
10
11     mapping(uint256 => Customer) private customers;
12     mapping(address => uint256) private loggedInUsers;
13     uint256 private ownerTravelId;
14
15     string public freeWifiCode = "SARRAY_MIL";
16
17     // Variables for temperature and humidity
18     int private temperature;
19     uint private humidity;
20
21     bool private dampedSeatActive;
22
23     address private constant companyAccount = 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4;
24     address private customerAccount;
25
26     // Jokes array
27     string[] private jokes = [
28         "Why don't skeletons fight each other? They don't have the guts.",
29         "Why did the scarecrow become a successful neurosurgeon? He was outstanding in his field.",
30         "What did the janitor say when he jumped out of the closet? Supplies!"
31     ];
32
33     constructor(address _customerAccount) payable {
34         ownerTravelId = 637042012; // Omer Farooq's travel ID
35         customerAccount = _customerAccount;
36
37         // Pre-loading customer database
38         customers[69039] = Customer("Milad", true, 0);
39     }
40 }
```

0 Listen on all transactions Filter with transaction hash or address

**revert**  
The transaction has been reverted to the initial state.  
Reason provided by the contract: "Invalid name or travel ID."



# Blockchain smart contract

The image displays a blockchain development environment with a smart contract interface on the left and its Solidity code on the right.

**Left Panel (Interface):**

- Transaction Section:** Includes a `_travelId` input field with the value `69039`, and buttons for `Calldata`, `Parameters`, and `transact`.
- Function Buttons:** `logout`, `rechargeTokens` (with a `uint256: tokenAmount` dropdown), `removeCusto...` (with a `uint256: _travelId` dropdown), `setHumidity` (with a `_newHumidity` input field set to `60`), `setTemperature` (with a `_newTemperature` input field set to `20`), `freeWifiCode`, `getHumidity` (showing `0: uint256: 60`), `getTemperatu...` (showing `0: int256: 20`), `getTokenBala...` (showing `0: uint256: 6`), `isDampedSea...`, and `tellJoke`.
- Low level interactions:** A section at the bottom labeled `CALLDATA`.

**Right Panel (Solidity Code):**

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
3
4 contract FerryService {
5     struct Customer {
6         string fullName;
7         bool firstTime;
8         uint256 tokenBalance;
9     }
10
11     mapping(uint256 => Customer) private customers;
12     mapping(address => uint256) private loggedInUsers;
13     uint256 private ownerTravelId;
14
15     string public freeWifiCode = "SARRAY_MIL";
16
17     // Variables for temperature and humidity
18     int private temperature;
19     uint private humidity;
20
21     bool private dampedSeatActive;
22
23     address private constant companyAccount = 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4;
24     address private customerAccount;
25
26     // Jokes array
27     string[] private jokes = [
28         "Why don't skeletons fight each other? They don't have the guts.",
29         "Why did the scarecrow become a successful neurosurgeon? He was outstanding in his field.",
30         "What did the janitor say when he jumped out of the closet? Supplies!"
31     ];
32
33     constructor(address _customerAccount) payable {
34         ownerTravelId = 637042012; // Omer Farooq's travel ID
35         customerAccount = _customerAccount;
36
37         // Pre-loading customer database
38         customers[69039] = Customer("Milad", true, 0);
39     }
40 }
```

**Bottom Panel (Debug Console):**

- Buttons: `Listen on all transactions`, `Filter with transaction hash or address`, `Debug`.
- Log entries:
  - `[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: FerryService.getTemperature() data: 0x642...1d04b`
  - `call to FerryService.getTokenBalance`