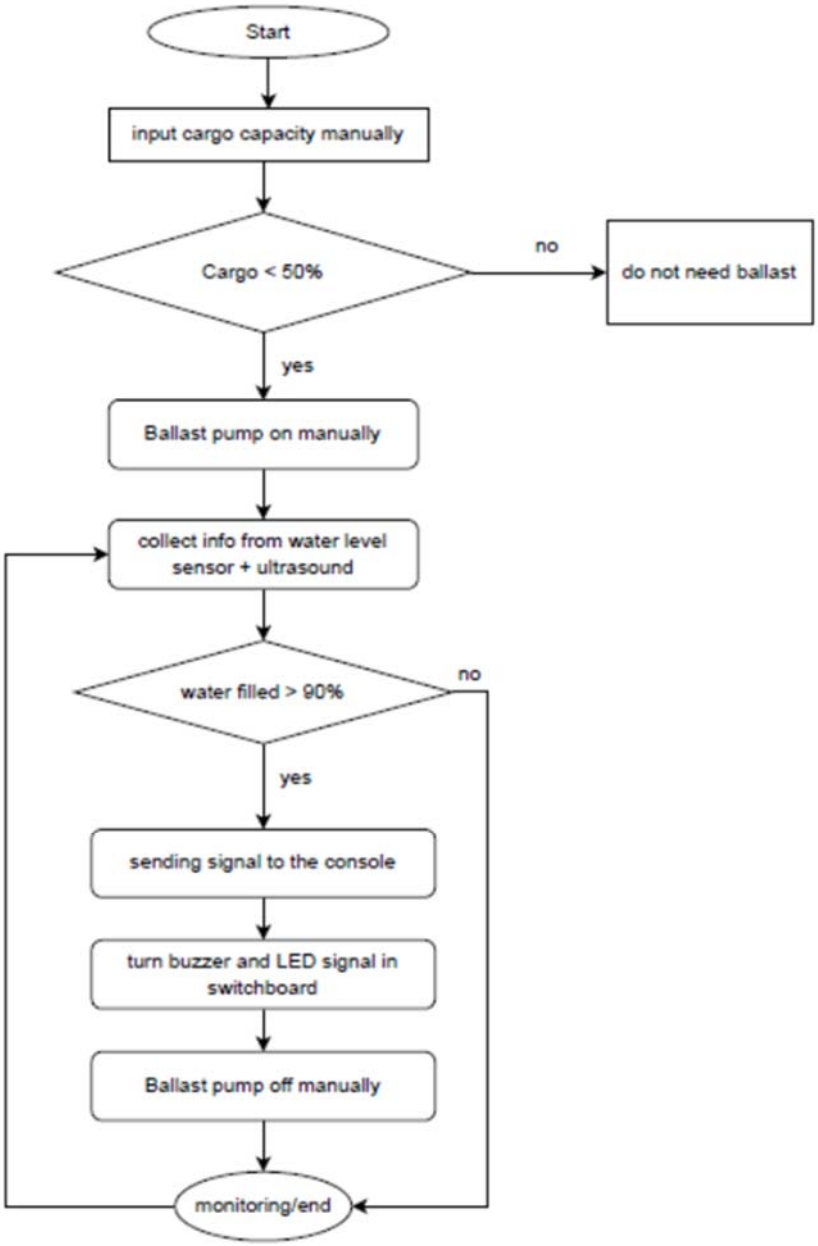
SMART CONTRACT FOR BALLASTING SYSTEM

(1) OBJECTIVE

To implement a smart contract in Solidity by applying blockchain features for a ballasting system (following ICPS flowchart) which consist of sensors and actuators.



(2) CODING PROCEDURE

a) Declaration of version of the Solidity complier

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.5.0 <0.8.0;
```

b) Creating Contract name and variable of Sensor

```
contract Ballast_System {
uint waterLevelUS;
uint waterLevelSensor;
```

c) Creating Warning functions to check Water Level of tanks and make Alarms (Buzzers & LED) by received Sensor Values

For Ultrasound Sensor

```
function warning_US (uint waterLevelUS) public view returns (string memory)
{
uint threshold = 450;
if (waterLevelUS > threshold)
{
return "Ballast tank is full! Buzzer & LED are on! Please turn the pump off.";
}
else
{
return "Ballast tank is not full! Buzzer & LED are off! Please do not turn the pump off.";
}
}
```

For Water Level Sensor

```
function warning_WLS (uint waterLevelSensor) public view returns (string memory)
{
uint threshold = 450;
if (waterLevelSensor > threshold)
{
return "Ballast tank is full! Buzzer & LED are on! Please turn the pump off.";
}
else
{
return "Ballast tank is not full! Buzzer & LED are off! Please do not turn the pump off.";
}
}
```

d) Creating Hash functions to generate hash value for received Sensor Values

For Ultrasound Sensor

```
function _sensor_US_Id(string memory _str)
private view returns (uint)
{
uint rand_US=uint(keccak256(abi.encodePacked(_str)));
return rand_US %waterLevelUS;
}
```

For Water Level Sensor

```
function _sensor_WLS_Id(string memory _str)
private view returns (uint)
{
uint rand_WLS=uint(keccak256(abi.encodePacked(_str)));
return rand_WLS %waterLevelSensor;
}
```

e) To store the keys by mapping for sensors value

```
mapping(address => uint) public sensors;
}
```