**A picture containing shape

Description automatically generated**

Industry 4.0 Enabling Technologies

Prof. Tiago Manuel Fernández Caramés

Prof. Paula Fraga Lamas

SMART CONTRACT DOCUMENTATION

**BY**

**NWAFOR OGECHUKWU**

**MEHMET ANIL GÜRLEK**

**LUIZ FELIPE BAUZER DE OLIVEIRA**

**ENES SANCAK**

**SUSTAINABLE SHIP AND SHIPPING 4.0**

**HIGHER POLYTECHNIC UNIVERSITY COLLEGE**

**UNIVERSITY OF A CORUÑA**

**A close-up of a logo

Description automatically generated with medium confidence**

Documentation

The smart contract named **HVACContract.sol** deployed is a **HVAC Verification and Validation System** such that the temperature, humidity and gas sensor data is collected and if the data recorded is within the acceptable range it accepts the data collected from the sensor, but if the data collected from the sensor is not within the acceptable range the system reverts and prints the statement “Temperature is not within the acceptable range”, “Humidity is not within the acceptable range”, “CO2 level above threshold”, “COlevel above threshold” respectively.

For our case study in validation with our Arduino project, the temperature limit should not be lower than 18 degrees Celsius nor exceed 25 degrees Celsius, the humidity should not be lower than 40% nor exceed 70%, the CO2 level should not exceed 1000ppm and CO should not exceed 9ppm, therefore the sensor data not within this level stated reverts and the system records the information that the data collected is not within the acceptable range.

Key Points

* Contract is created using Solidity version >=0.5.0 and 0.8.0
* The keccak256 hash of abi.encodePacked(\_str) is used to generate a pseudo-random hexadecimal, typecast it as a uint, and finally store the result in a uint called rand.
* Function Temperature, Humidity, CO2 and CO set to public takes a uint \_degree, \_percentage, \_PPM and \_ppm respectively which is our sensor data that will retrieved from our sensor.
* Function validationTemp , validationHum, validationCO and validationCOO set to private view takes a uint \_valueId , valueIId, valueII, and valueI respectively and checks that the data received by the system does not exceed the set limits and an IF control structure to ensure the data received by the system also does not exceed the set limits.
* Uint mapping named degree , percent , value and valuee is created to store the value of the temperature data, humidity data, CO2 and CO data respectively .
* Modifier is used in order to apply our conditions with a require that checks if the sensor data is within acceptable range otherwise it reverts the transaction if the check fails.