Table of contents

## 1. Members:

- Jonas Caetano
- Pedro Henrique Pereira Martins
- Quenio Cesar Machado dos Santos

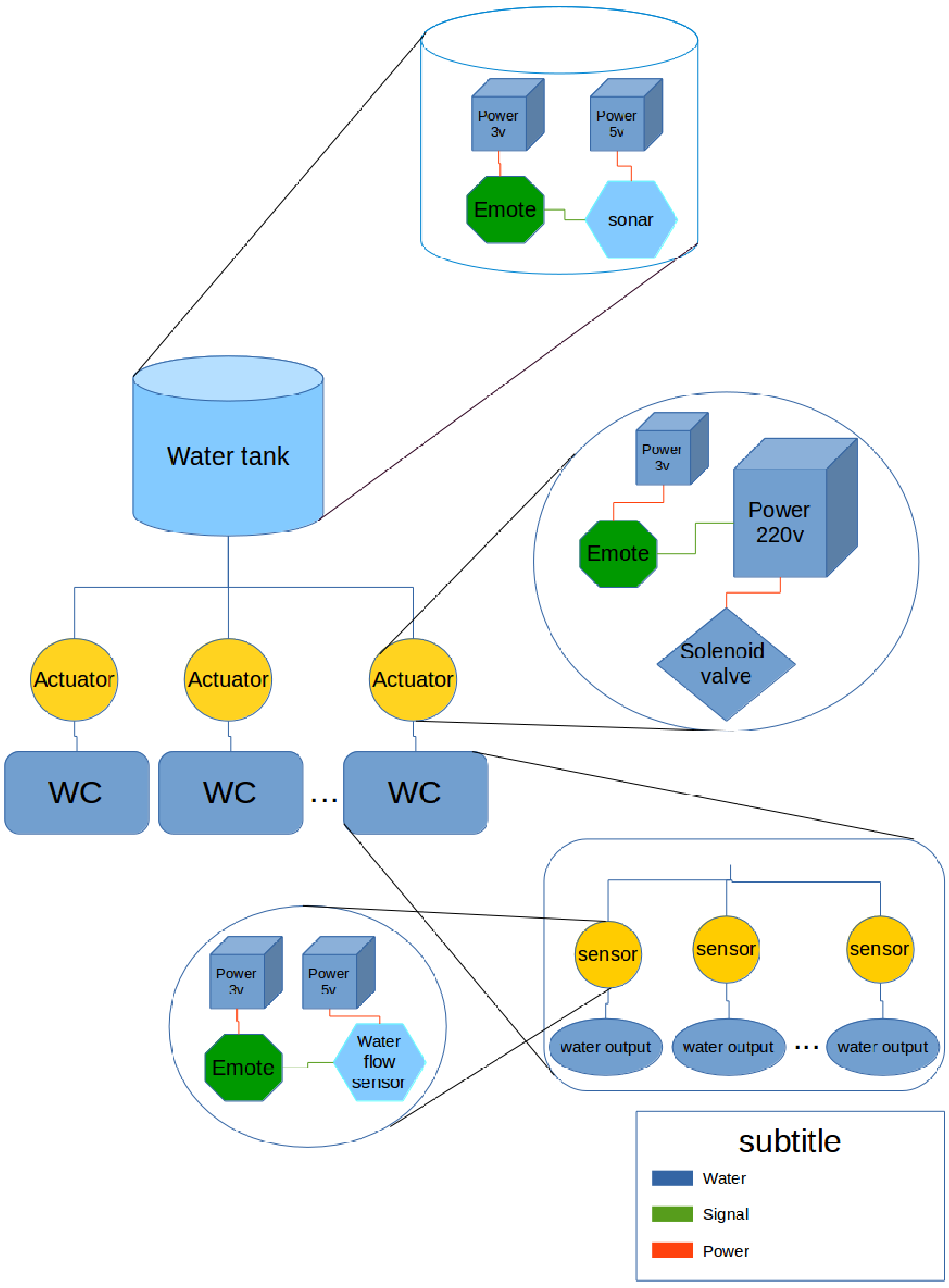
## 2. Schedule:

| Day | Time | Hours | Room |
|---|---|---|---|
| Monday | 13:30 | 2 | INE 214 |
| Wednesday | 13:30 | 2 | INE 214 |

## 3. About the project:

In our project we will try to automatize the hydraulic system of a building. To start, we will install in each faucet sensors to measure the water flow and use this flow to power the sensors. With the results of the measurement we can estimate the use

## 4. Model



Power 3v
Power 5v
Emote
sonar
Water tank
Power 3v
Power 220v
Emote
Solenoid valve
Actuator
Actuator
Actuator
WC
WC
...
WC
sensor
sensor
sensor
water output
water output
...
water output
Power 3v
Power 5v
Emote
Water flow sensor
subtitle
Water
Signal
Power

## 4.1. Consumption monitoring

The monitoring of the building water consumption will be realized with the use of water flow meters (one in each tap), coupled to these water flow meters, there will be an Emote which will pass the data collected by the sensor to a central repository.

## 4.2. Water leaking detection

To perform the leaks detection two approaches were used, fuzzy logic and neural networks.

### 4.2.1. Fuzzy logic

One of the approaches that we use was fuzzy logic, which generates results within a range, in this case, generates a value between 0 and 1, this results in a degree of certainty the occurrence of leaks where 0 is sure that is not occurring a leak and 1 is sure that is happening a leak.

For the use of fuzzy logic was extended the tool Ptolemy using a Java library for fuzzy logic representation JFuzzy, this library allows the creation of fuzzy systems using a standardized language to generate the predicates with their respective weights and margins of action that will be interpreted by the logic engine library and will generate a result within the specified range. The specification of our problem using the language adopted by JFuzzy was made as shown below:

```
FUNCTION_BLOCK hydraulics

    VAR_INPUT
        machine_count: REAL;
        person_count: REAL;
        supply_flow: REAL;
        terminal_flow: REAL;
    END_VAR

    VAR_OUTPUT
        leak_likelihood: REAL;
    END_VAR

    FUZZIFY machine_count // Interval: [0,5]
        TERM few_machines := (0, 1) (4, 0);
        TERM many_machines := (1, 0) (5, 1);
    END_FUZZIFY

    FUZZIFY person_count // Interval: [0, 40]
        TERM few_people := (0, 1) (30, 0);
        TERM many_people := (10, 0) (40, 1);
    END_FUZZIFY

    FUZZIFY terminal_flow // Interval: [0, 6]
        TERM low_flow := (0, 1) (2, 0);
        TERM med_flow := (1.5, 0) (3, 1) (4.5, 0);
        TERM high_flow := (4, 0) (6, 1);
    END_FUZZIFY

    FUZZIFY supply_flow // Interval: [0, 10]
        TERM low_flow := (0, 1) (5, 0);
        TERM med_flow := (4, 0) (5, 1) (6, 0);
        TERM high_flow := (5, 0) (10, 1);
    END_FUZZIFY

    DEFUZZIFY leak_likelihood // Interval: [0.0, 1.0]
        TERM no_leak := (0.0, 1) (0.4, 0);
        TERM possibly_leak := (0.35, 0) (0.5, 1) (0.65, 0);
        TERM actual_leak := (0.6, 0) (1.0, 1);
        METHOD : COG;           // Use 'Center Of Gravity' defuzzification method
        DEFAULT := 0.0;         // Default value is 0 (if no rule activates defuzzifier)
    END_DEFUZZIFY

    RULEBLOCK leak_detection
        AND : MIN;                      // Use 'min' for 'and' (also implicit use 'max' for 'or' to fulfill DeMorgan's Law)
        ACT : MIN;                      // Use 'min' activation method
        ACCU : MAX;                     // Use 'max' accumulation method

        RULE 1: IF person_count IS few_people AND terminal_flow is med_flow THEN leak_likelihood is possibly_leak;
        RULE 2: IF machine_count is few_machines AND terminal_flow is med_flow THEN leak_likelihood is possibly_leak;
        RULE 3: IF person_count IS few_people AND machine_count is few_machines AND terminal_flow is med_flow THEN leak_likelihood is actual_leak;
        RULE 4: IF terminal_flow IS low_flow AND supply_flow is high_flow THEN leak_likelihood is actual_leak;
    END_RULEBLOCK

END_FUNCTION_BLOCK
```

After specifying the functioning of our fuzzy logic, we extended the tool to support the using of our implementation, for this the following classes were created:

```java
class Fuzzy
{
    private static final boolean VERBOSE = false;

    private Fuzzy() {}

    static FIS loadFIS(Class<? extends TypedAtomicActor> actorClass, String fileName)
    {
        return FIS.load(actorClass.getResourceAsStream(fileName), VERBOSE);
    }

    static void receiveInput(FIS fis, TypedIOPort... ports) throws IllegalActionException
    {
        for (TypedIOPort port: ports)
        {
            final Optional<Double> currentInput = getDoubleInput(port);
            if (currentInput.isPresent())
            {
                fis.setVariable(port.getName(), currentInput.get());
            }
        }
    }

    static void sendOutput(FIS fis, TypedIOPort... ports) throws IllegalActionException
    {
        for (TypedIOPort port: ports)
        {
            setDoubleOutput(port, defuzzify(fis, port.getName()));
        }
    }

    private static double defuzzify(FIS fis, String varName)
    {
        return fis.getVariable(varName).getDefuzzifier().defuzzify();
    }
}
```

```java
class FuzzyActor extends TypedAtomicActor
{
    private String fclFileName;
    private List<TypedIOPort> inputs = new ArrayList<>(), outputs = new ArrayList<>();

    private FIS fis;

    FuzzyActor(CompositeEntity container, String name, String fclFileName)
        throws IllegalActionException, NameDuplicationException
    {
        super(container, name);
        this.fclFileName = fclFileName;
    }

    TypedIOPort newInput(String portName) throws IllegalActionException, NameDuplicationException
    {
        final TypedIOPort port = Ports.newInput(this, portName, BaseType.DOUBLE);
        inputs.add(port);
        return port;
    }

    TypedIOPort newOutput(String portName) throws IllegalActionException, NameDuplicationException
    {
        final TypedIOPort port = Ports.newOutput(this, portName, BaseType.DOUBLE);
        outputs.add(port);
        return port;
    }

    @Override
    public void initialize() throws IllegalActionException
    {
        super.initialize();
        fis = loadFIS(getClass(), fclFileName);
    }

    @Override
    public void fire() throws IllegalActionException
    {
        super.fire();

        receiveInput(fis, getInputArray());
        fis.evaluate();
        sendOutput(fis, getOutputArray());
    }

    private TypedIOPort[] getInputArray()
    {
        return inputs.toArray(new TypedIOPort[inputs.size()]);
    }

    private TypedIOPort[] getOutputArray()
    {
        return outputs.toArray(new TypedIOPort[outputs.size()]);
    }
}
```

```
public class LeakDetector extends FuzzyActor
{
    private static final String FCL_FILE_NAME = "hydraulics.fcl";

    // Ports:
    private static final String MACHINE_COUNT = "machine_count";
    private static final String PERSON_COUNT = "person_count";
    private static final String SUPPLY_FLOW = "supply_flow";
    private static final String TERMINAL_FLOW = "terminal_flow";
    private static final String LEAK_LIKELIHOOD = "leak_likelihood";

    @SuppressWarnings("WeakerAccess")
    public TypedIOPort machineCount, personCount, supplyFlow, terminalFlow, leakLikelihood;

    public LeakDetector(CompositeEntity container, String name)
        throws IllegalActionException, NameDuplicationException
    {
        super(container, name, FCL_FILE_NAME);

        machineCount = newInput(MACHINE_COUNT);
        personCount = newInput(PERSON_COUNT);
        supplyFlow = newInput(SUPPLY_FLOW);
        terminalFlow = newInput(TERMINAL_FLOW);
        leakLikelihood = newOutput(LEAK_LIKELIHOOD);
    }
}
```

The results obtained with the use of fuzzy logic were not very satisfactory, the reason is that this kind of approach requires a certain knowledge of the kind of problem that you want to solve and we have no leaks specialist, then our specification of what is a leak was not very accurate.

### 4.2.2. Neural network

Besides the use of fuzzy logic, also chose using neural networks, this approach is simpler, the knowledge that it takes on the problem domain is not defined by us, but by all of the collected data from the environment, thus, It turns the output to be more precise depending on the size and accuracy of the data collected.

The Neural networks workflow is simple, you collect a data set either with an input and the expected output, trains the network with this data set and after training your network already will be able to classify new samples of data, the result of rating will depend on the data set used to train the network and the number of intermediate neurons that were used on the network.

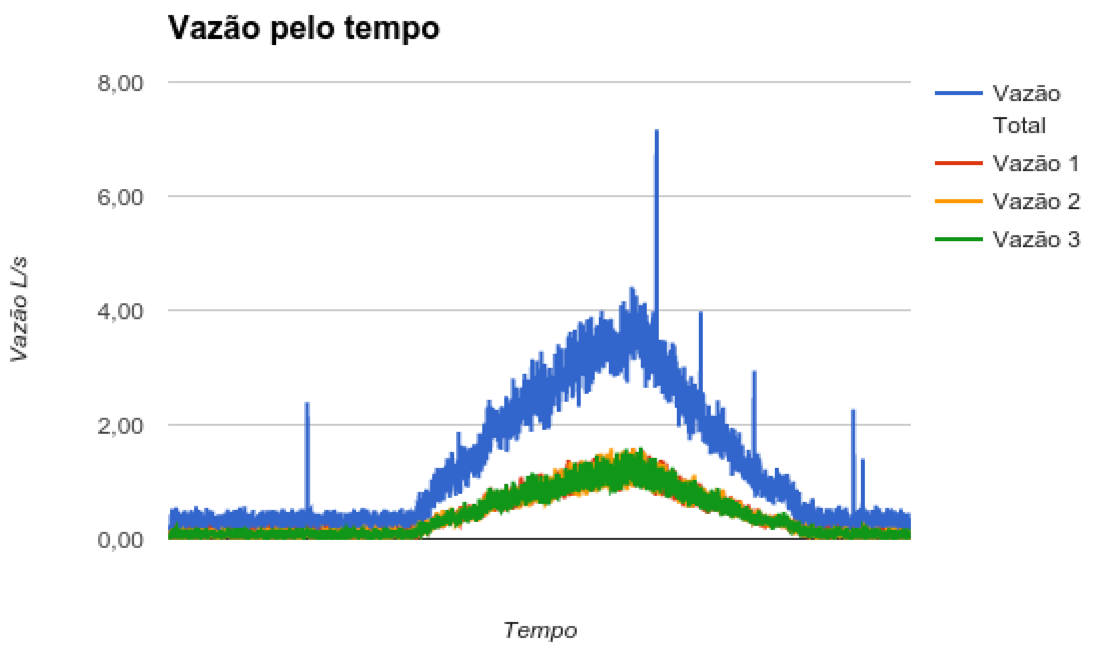
To implement our neural network and perform the tests was used MATLAB. First, we create a script in MATLAB that will load the simulation data in a matrix, after this it will separate the array into two (2), one containing the training set, and another containing the set to be classified, after the network is trained and the remaining data are classified, we did a comparison of the data classified by our network and the original data to determine the network's hit percentage. The script created in MATLAB is shown below:

```
clear
split = 900;
m = transpose(csvread('DataSet.csv'));
me = m(1:6,1:split);
mx = mapminmax(me);
ms = m(7:8, 1:split);

net_size = 8;
net = newff(mx, ms, net_size, {'tansig','tansig'}, 'trainlm');
net.trainParam.epochs = 1000;
net.trainParam.goal = 0;

te = m(1:6,split+1:end);
tx = mapminmax(te);
ts = m(7:8, split+1:end);

net = train(net,tx,ts);
mr = saturate(sim(net, mx));
plotconfusion(ms, mr);
```

```
function s = saturate(output)
    row_size = size(output, 1);
    col_size = size(output, 2);
    s = zeros(row_size, col_size);
    for i = 1:col_size
        v = output(:, i);
        if v(1) > v(2)
            s(1, i) = 1;
        else
            s(2, i) = 1;
        end
    end
end
```

The idea of adopting neural networks in our solution is to continuously expand the set of samples for training every time our network make a mistake. The sample that our network wrongly classified will be compared with other samples contained in our training set using a simple similarity technique (a simple multiplication, because both metrics have the same significance), if any sample in training set is similar with the misclassification, then the value contained in this sample response will be updated with the value contained in the sample misclassified, if not, the sample misclassified will be added to the set of samples for training.

To create the training dataset, was done a simulation equivalent to one (1) working day of a building that has 1 (one) central tank and 3 (three) water outlets, the simulation took into account the opening hours of the building between 8:00am and 10:00pm, it was estimated that each person consumes an equivalent of 2 liters of water during they staying in the building but was not taken into account a flow rate limit for each water outlet, it can be noticed a marked change in the building's water consumption during the simulation. The graph above shows the water flow pattern over the day.



The spreadsheet with the dataset used to train the neural network can be found here

## 4.3. Nonfunctional model

Were detected 4 non-functional requirements for this project:

### 4.3.1. Power source

As illustrated in our model, the EPOS mote, sensors and valves need a source power, however in some cases (in the water reservatory at the roof) where the humidity level and the temperature are high, the source power needs to be resistant to humidity and high temperature.

### 4.3.2. Sensor Consumption

The water flow sensor needs to have a low power consumption because it will be auto sufficient in power consumption, that means the sensor will be it own source of power.

### 4.3.3. Sonar sensor

The sonar sensor will be inside a water tank, it means that this sensor needs to be humidity resistant like it source power.

### 4.3.4. Solenoid valve

The valve needs to support a common water pressure, we think that a default valve supports that but some kinds of edifications have pressurizers that turn the water pressure high and some kinds of valves may get problems with this high pressure. Another requirement about the valve is the lack of energy, the solenoid valve needs to open when not energized and close when energized, its prevent the lack of water when the energy is down at the same time that reduces the energy consumption because the valve will be more time open than closed.

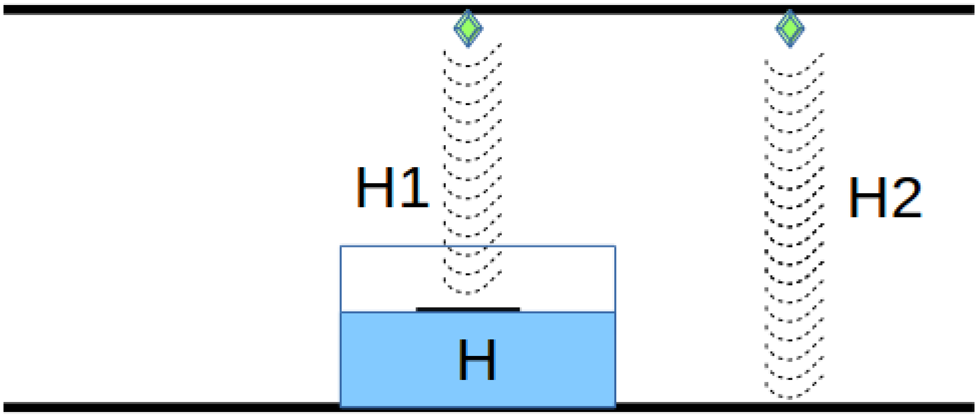
### 4.3.5. Leak detection

The response needs to have a deterministic processing time to other systems be able to have enough time to work properly.

#### 4.3.6. Sensor update time

Because wireless is used to communicate with the database, it could occur delays between each sensor update. This can be a problem if systems reading the database don't check if the data was updated in the same time.

### 4.4. Water reservatory monitoring



$Y = H/H0 * 100$

percentage of capacity in relation to H0

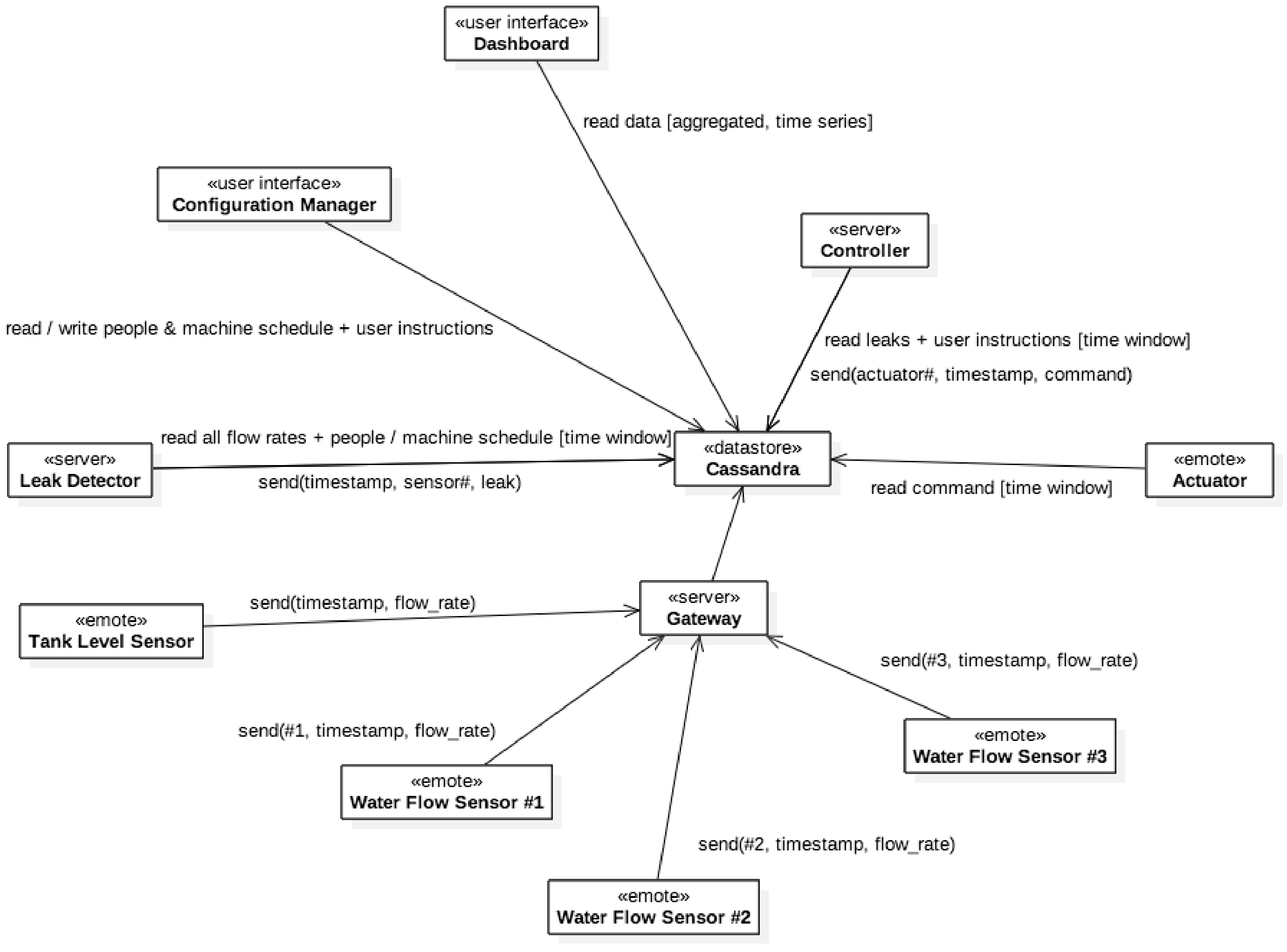
$H = H2 - H1$

$H0 = H$ obtained in the first measurement

To obtain the water level in the reservatory, we will use two sonar sensors, one to estimate the distance between the ceiling and the floor (H2) or the base that the water box is on and the other to estimate the distance between the ceiling and a board floating on the water surface (H1)...

Be continued

## 5. Architecture


«user interface» Dashboard
read data [aggregated, time series]
«user interface» Configuration Manager
read / write people & machine schedule + user instructions
«server» Controller
read leaks + user instructions [time window]
send(actuator#, timestamp, command)
«server» Leak Detector
read all flow rates + people / machine schedule [time window]
send(timestamp, sensor#, leak)
«datastore» Cassandra
«emote» Actuator
read command [time window]
«emote» Tank Level Sensor
send(timestamp, flow_rate)
«server» Gateway
send(#3, timestamp, flow_rate)
«emote» Water Flow Sensor #3
send(#1, timestamp, flow_rate)
«emote» Water Flow Sensor #1
send(#2, timestamp, flow_rate)
«emote» Water Flow Sensor #2

- Datastore : This entity is the central point of the system. It stores all the data received from the gateway, the result from the LeakDetector and, by also storing commands from the Configuration Manager and the Controller, it works as a facade for the communication between other entities.

- Gateway : Collects data from all sensors and sends to the Datastore.

- Water Flow Sensors : This sensor, connected to an EPOS Mote, calculates the water flow rate at the point where it is installed and sends this value to the Datastore via the Gateway. Combined with the water flow rate, the sensor also sends its identification and the current time. This action is executed periodically.

- Tank Level Sensor : This sensor calculates the water level of the reservoir and sends this value to the Datastore. Combined with the water level, the sensor also sends the current time. This action is executed periodically.

- Leak Detector : This entity reads a series of datas from the Datastore and, based on that, determine whether or not a leak exists in the building. After having a conclusion, it sends this result back to the Datastore. The data read from the Datastore includes the water flow rate obtained from each sensor, the number of people in the building and the number of machines actively using water. Combined with the result, the Leak Detector also sends a identification of where the leak is occurring and the current time. This action is executed periodically.

- Configuration Manager : This entity allows the users to provide the schedule of people presence and machine activity at the building. Another responsibility is to receive instructions from the users and register them at the Datastore for use of others entities.

- Controller : This entity reads the Datastore and, based on user instructions and the results of the Leak Detector, sends back to the Database commands to the Actuator. Combined with the command, the Controller also sends the identification of the target Actuator and the current time.

- Actuator : This Actuator reads the Datastore and if a command targeted to its identification exists, it opens or closes the flow based on the instruction.

- Dashboard : This entity reads the Datastore and provides to the user charts and tables with information about the current state of the system.

## 6. Computer Model

```
Class WaterFlowSensor {

Var interruptionCount = 0;
Var url = "db.lisha.ufsc.br/persistData";
Var period = 30;
Var readResult = 0;
Var timeout = 10000;
Var sensor = 1;

@interruption GPI/O pin 5
Method void GPI/OinterruptionHandler(){
        interruptionCount ++;
}

@Atomic
Method void waterflowReader(){
        readResult = interruptionCount / period;
        interruptionCount = 0;
        resultPersists();
}

Method void resultPersists(){
        Var http = getHttpConection(url);
        http.setMethod("POST");
        Var date = timer.getData();
        Var JSON = "{type:waterflow, timestamp:date, readResult:readResult, sensor:sensor}";
        http.sendRequest(JSON,timeout);
        http.close();
}

Method void main(args[]){
       Var AlarmReader = new Alarm(period*1000,waterflowReader(),'infinite');
}

}
```

```
Class LeakDetector{

Var urlRetrieve = "db.lisha.ufsc.br/retrieve";
Var urlPersist = "db.lisha.ufsc.br/persist";
Var period = 30;
Var timeout = 10000;
Var lastResult = -1;
Var sensorsNumbers = 4;
Var lastRetriviedData = -1;
Var neuralNetwork = loadNeuralNetworkFromFile();
Var trainingSet = getFile("trainingSet");

Method JSON retrieveData(JSONConsulta){
        Var http = getHttpConection(urlRetrieve);
        http.setMethod("GET");
        Var JSON = http.sendRequest(JSONConsulta,timeout);
        http.close();
        return JSON;
}

Method void retrainNeuralNetwork(){
        Var date = timer.getData();
        Var timeInterval = date - period;
        Var JSONConsult = "{type:wrong, timestamp:date}>= && >={type:wrong,
timestamp:period}"
        Var JSON = retrieveData(JSONConsult);
        if(JSON.value){
              trainingSet.append(lastRetriviedData + ! lastResult);
              neuralNetwork.train(trainingSet);
        }
}

Method void atualizeRetriviedData(JSON){
        if(JSON.elements.size == sensorsNumbers || lastRetriviedData == -1){
              lastRetriviedData = JSON;
        }else{
              JSON = lastRetriviedData;
        }
}

Method void detectLeak(){
        retrainNeuralNetwork();
        Var date = timer.getData();
        Var timeInterval = date - period;
        Var JSONConsulta = "{ type:waterflow, timestamp:date}" >= && >= "{ type:waterflow,
timestamp:date}" ;
        Var JSON = retrieveData(JSONConsulta);
        atualizeRetriviedData(JSON);
        Var result = neuralNetwork.classifie(JSON);
        Var date = timer.getDate();
        Var JSONResult = "{type:leakdetection, result:result, timestamp:date}";
        persistData(JSONResult);

}
Method void persistData(JSON){
        Var http = getHttpConection(urlPersist);
        http.setMethod("POST");
        http.sendRequest(JSON,timeout);
        http.close();
}

Method void main(args[]){
       new Alarm(period*1000,detectLeak(),'infinite');
}

}
```

```
Class SolenoidValveActor {

Var url = "db.lisha.ufsc.br/retrieveData";
Var period = 30;
Var timeout = 10000;
Var actuatorId = 1;
Var valve = new GPI/O(pin 5);

Method JSON retrieveData(JSONConsulta){
        Var http = getHttpConection(urlRetrieve);
        http.setMethod("GET");
        Var JSON = http.sendRequest(JSONConsulta,timeout);
        http.close();
        return JSON;
}

Method void actuate(){
        Var date = timer.getData();
        Var timeInterval = date - period;
        Var JSONConsult = "{ type:leakDetection, timestamp:date}" >= && >= "{
type:leakDetection, timestamp:date}"
        Var JSON = retrieveData(JSONConsult);
        if(JSON.result == 1%10 && JSON.result/10 == actuatorid){
            valve.setValue(1);
            return;
        }
        if(JSON.result == 'solved'){
            valve.setValue(0);
        }
}

Method void main(args[]){
       Var Alarm = new Alarm(period*1000,actuate(),'infinite');
}

}
```

# 7. Technologies and project overview:

All the devices used in the project were provided by the Software/Hardware Integration Laboratory (LISHA).

## 7.1. Water Flow Sensor

Will be used a simple water flow sensor that has a turbine and sends a signal each time the turbine do a complete spin. How the water flows meter consumption is low and have a turbine that rotates with the water flow (this is how it works) it could be powered using the power generated by the rotation of the turbine.

http://bazaar.seeedstudio.com/item_detail.html?p_id=635

## 7.2. Dynamo

Description

## 7.3. Eletronic valve (Registro/válvula)

Description

## 7.4. EPOSMote III

The Embedded Parallel Operational System (EPOS) will run inside of EPOSMote III. The epos mote uses a 3v power source (at least that's what I remember), to support that, we can use AA stacks of 1.5v or a 12v battery with an electrical transformer embedded can also be used.

## 7.5. Sonar

## 7.6. Humidity sensor

## 7.7. 3v power source

## 7.8. 5v power source

## 7.9. 220v power source