# SentinelSuite

**IoT Project for Server Room Surveillance**

Castorini Francesco

GitHub Repository: (https://github.com/Il-castor/IoT)

This project presents a comprehensive exploration and implementation of an Internet of Things (IoT) system designed to manage access control and environmental conditions within a critical server room environment. The project incorporates the utilization of ESP32 and ESP8266 microcontrollers for access and environmental policy enforcement, respectively, alongside a Raspberry Pi serving as a central hub for data processing and visualization.

## 1 Access and Environmental Control

### 1.1 Access Policy

It checks whether a user has credentials or not. If yes, it opens the door, otherwise it doesn't and a notification is sent. This policy is implemented on a ESP8266.

### 1.2 Environmental Policy

It measures environmental parameters such as temperature, humidity, flooding and flame and if they are above certain thresholds, countermeasures are taken. This policy is implemented on a ESP32.

## 2 Hardware

For the realization of the project I used 3 different board:

1. ESP32
2. ESP8266
3. Raspberry Pi

The boards have the following hardware specs:

| Board | Platform | CPU | RAM | Flash |
|---|---|---|---|---|
| ESP32 Dev | Espressif 32 | ESP32 | 240MHz | 4MB |
| ESP8266 | Espressif 8266 | ESP8266 | 80MHz | 4MB |
| Raspberry Pi 3B | - | 1.2GHz | 1GB | 32GB |

All of these board have WiFi capability and expandibility for connecting external device.

## 3 Software components

I use **MQTT** because it is light, efficient and ideal for transmitting data in constrained environments. In the future it will be possible to extend the sensor network.

**Mosquitto** (https://mosquitto.org/) is an open source message broker that implements the MQTT protocol.

**Node-RED** (https://nodered.org/) is a programming tool for wiring together hardware devices, APIs and online services. It is configured to process data and monitoring metrics coming from the boards.

All these software are installed on Raspberry Pi which acts as a central hub.

The complete project architecture is the following:

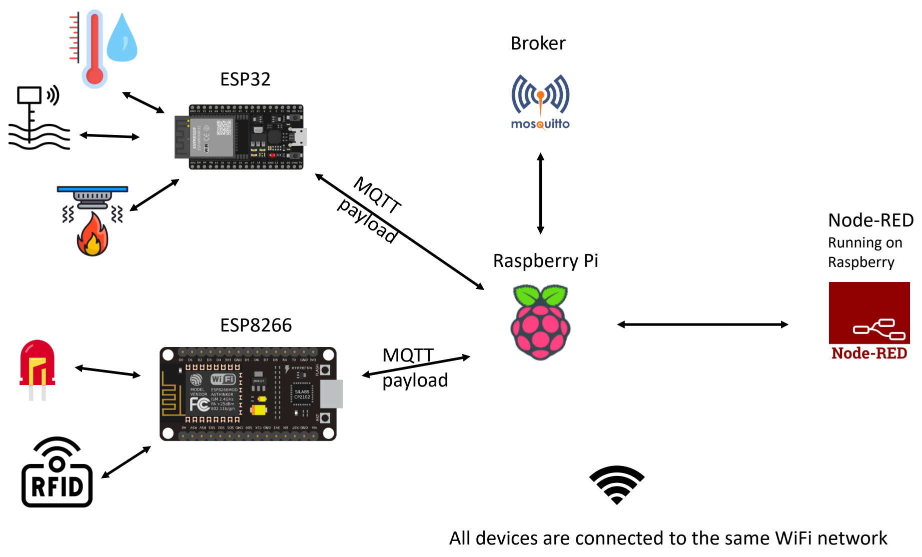


Figure 1: project architecture

I use **WiFi** because offers higher data transfer rates, ensuring the efficient transmission

of sensor data. Moreover, the infrastructure within a server room typically supports WiFi, reducing additional setup requirements. In the end, WiFi accommodates multiple simultaneous connections without substantial signal degradation, enabling both ESP devices to communicate concurrently with the Raspberry Pi.
Therefore, WiFi's balance of speed, reliability, and compatibility with the existing infrastructure makes it the preferred choice for facilitating seamless and robust communication between the ESP sensors and the central Raspberry Pi hub.

## 4 Implementation

ESP8226 implements access policy using a RFID reader. When a card is near the reader, the board sends card's values to Node-RED via MQTT and receives a certain value to enable access or not. To enable access the board turns on green led and makes a sound. If access is denied, because card is not registered, then a red led lights up. After a certain number of incorrect attempts the RFID reader is disabled and can be enable only on Node-RED dashboard.

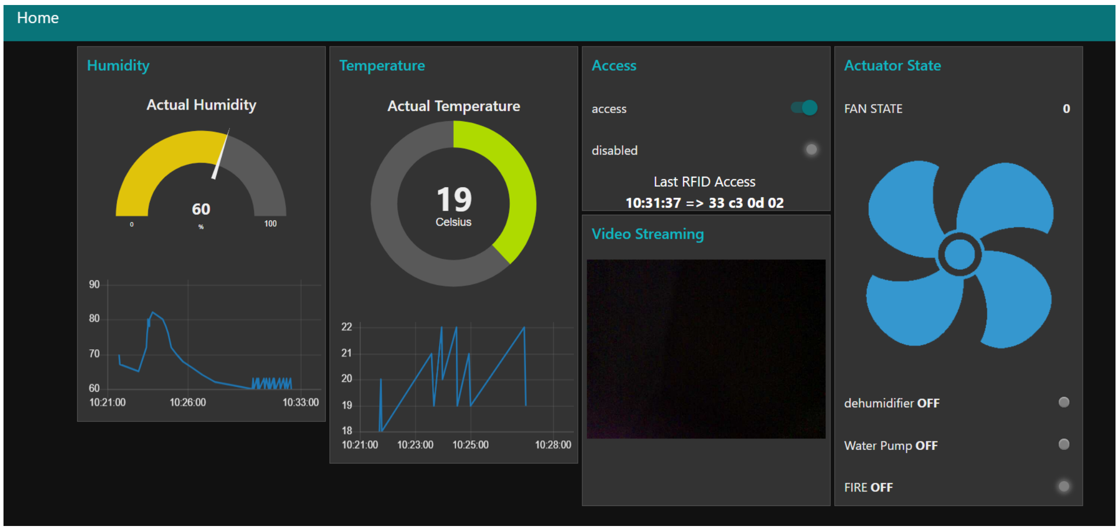
ESP32 implements environmental policy using different sensors. It uses DHT11 sensor to read humidity and temperature, flame sensor to detect if there is a fire in the room and water level sensor if there is flooding. All this data is sent to Node-RED using MQTT and the board receives certain data based on the countermeasures it needs to adapt:

- if humidity is over certain threshold it turns on dehumidifier
- if the temperature is between different thresholds then it turns on the fan at different speeds
- if water is over certain threshold it turns on a water pump to suck up the water
- if it detects flames it turns on the fire prevention system

Some LEDs are used to simulate the actuators described. The four fan speeds, which are 0, 1, 2, 3 are mapped in the following colors: white, purple, green and red; the dehumidifier is black if it is not turned on, otherwise it is orange. The turned on water pump is mapped with blue color, while the fire sprinkler system flashes red for two seconds if activated, otherwise it is turned off

For better security in access control there is a streaming video, implemented on a Raspberry Pi, which records the RFID reader.

Data collected by the 2 ESP are sent periodically to Node-RED. It examines these data and decides what to do. Node-RED sends a response to the ESP devices about which actuator to turn on or off. Additionally, these data are displayed on a dashboard so a person can monitor the server room.



access

disabled

Last RFID Access
10:31:37 => 33 c3 0d 02

FAN STATE 0

dehumidifier **OFF**

Water Pump **OFF**

FIRE **OFF**

Figure 2: Node-RED dashboard

In this dashboard there are two graphs, respectively for humidity and temperature, the state in which access to the server room is located, the video streaming and the states in which the installed actuators are found. In this dashboard there are two graphs that represent the current temperature and the history of temperatures and same for humidity. There is a section that indicates access to the server room and to watch the video streaming. Finally there is a section that indicates the states in which the actuators are.

Furthermore, via Node-RED it is possible to easily modify the parameters to make the management of the server room configurable according to needs.