**CASANOVA IoT Project**

**CASANOVA** is a project that aims to design and implement a smart and secure home system by using modern microcontrollers,sensors and different communication protocols.This system will ensure to access and monitor house in real-time and if something is not in right parameters it will generate proactive alerts by enhancing both safety and convenience for homeowners.

### 

### **I-Project Description:**

1.**Authorization for opening the door.** A RFID card reader will be used in order to control access to the home.The idea is that id card ID matches a pre-stored ID in database the system will unlock the door if not an email will be sent to admin to inform that somebody was trying to open the door.An additional **ultrasonic sensor** will verify the presence of an individual in front of the door.If the system receives an **HTTP request** to unlock the door but the ultrasonic sensor detects no one present, it will assume a **hacking attempt** and trigger an alert by also sending a mail.

2.**Environmental Monitoring.** For monitoring the house or creating a prototype to monitor the house we will use one main sensors that calculate the temperature and the humidity which will provide real-time environmental data.We will inform the house owner for it’s house conditions in real time by using some leds. If the temperature exceeds a predefined threshold, the system will turn on RED LED , and if the temperature is below a certain threshold, the system will light a LED in blue.Same idea for RGB LED that is used for humidity too.

3.**Intrusion security:** By usingthe accelerometer of the Arduino Nano on the door, the system will detect if the door is opening without authorization, if it detect, an email will be send to the home owner.

**Microcontrollers:**

* + **ESP32 x2**
  + One will be used only for doing communication with BLE with Arduino Nano as the sketch is big and there will be not enough memory if we use only one ESP32
  + Communication between 2 ESP32 will be done by UART is serial communication ->Serial2 (pins RX2 and TX2).
  + **Arduino Nano 33 IoT**

**Server:**

* + **Raspberry Pi 4**

### **II-Communication Protocols:**

**1.MQTT(Message Queuing Telemetry Transport):**

MQTT is a lightweight, efficient protocol for communication between microcontrollers and the Raspberry PI server.

ESP32 ↔ Raspberry Pi (Server)

* **ESP32**: By using DTH sensor we will receive DHT data which are temperature and humidity and publish them to a topic by using MQTT protocol.
* **Raspberry Pi**: Subscribes to the topic and processes the received data to do data visualization.
* **MQTT Broker**: The MQTT broker (Mosquitto) runs on the Raspberry Pi, managing message delivery between publisher (ESP32) and subscriber(Raspberry Pi).

**2.HTTP (Hypertext Transfer Protocol):**

Used for interaction between devices and the server for specific actions.

The ESP32 sends **HTTP POST** requests to the Raspberry Pi for:

* Logging RFID scans.
* Triggering critical alerts (e.g unauthorized access attempts).
* The Raspberry Pi processes the requests and forwards the data to an influxDB measurement and sends an email if it returns unauthorized.

**3. BLE (Bluetooth Low Energy):** Short-range communication between microcontrollers

Arduino Nano ↔ ESP32 (Direct Communication)

* Arduino Nano will be used to connect the accelerometer data from it’s sensor that it has.
* It will communicate with **ESP32** by using **BLE** to send the accelerometer data to ESP32.

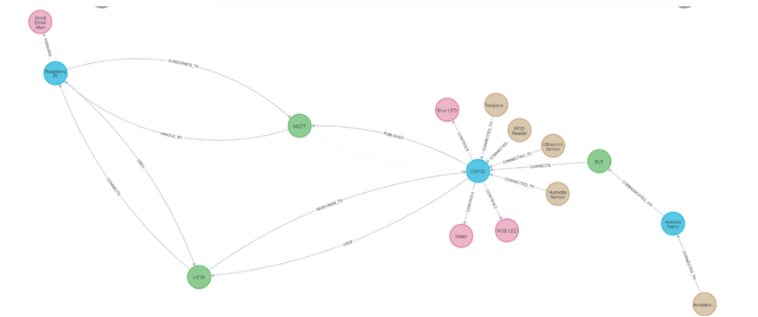
### 

### **III-Knowledge Graph**

The **knowledge graph** represents the interaction between various components of your IoT system, including devices, sensors, protocols, and actions. Here's a breakdown of each element and its relationships.The graph is created with Neo4j software.

1.Nodes:

1. Blue Circles (Devices): Microcontrollers or servers (ESP32, Arduino Nano, Raspberry Pi).
2. Green Circles (Protocols): Communication methods (MQTT, HTTP, BLE).
3. Beige Circles (Sensors): Physical sensors collecting data (Temperature, Humidity, Motion, Ultrasonic,Accelerometer data).
4. Pink Circles (Actions): Operations triggered by devices (Send Email Alert, …)

****

### **IV-Data Collection:**Strategies for gathering data

Data collection involves collecting real-time information from various sensors and devices,processing it and ensuring reliable communication between the components.

* **DHT Sensor:** Used to get temperature and humidity.
* **Accelerometer (Arduino NANO):** Used to detect when the door is opening to check if there was an authorization just before or not.
* **RFID Reader:**Read RFID tag for secure access control.
* **Ultrasonic Sensor:**Will be used to measure the distance of a person from the door to check if there is someone in front of the door or not.

After this microcontrollers collect data locally from it’s connected sensors.By processing data locally, you reduce the amount of raw data sent over the network, optimizing bandwidth and power consumption

**ESP32:**

* Reads temperature and humidity data from the sensors .
* Processes door opening events from the Arduino NANO accelerometer.
* Reads RFID data when a tag is scanned.
* Collects presence in front of the door data from the ultrasonic sensor .

**Arduino Nano:**

* Collects accelerometer data.

To transmit data to the central server (Raspberry Pi), use a combination of communication protocols:MQTT,HTTP. MQTT is the communication protocol that will be used to send data to server Raspberry PI.

**MQTT:**Publish sensor readings from ESP32 to specific topics. Raspberry Pi subscribes to these topics to receive those data. Also the RaspberryPi will publish on some topics to tell the ESP32 if the temperature is too hot, for example to turn on the motor. So the ESP32 will be a subscriber too.

**HTTP:**Send request when we read RFID card id to check if it matches with card id stored in MySQL database.

**BLE:** Used for communication between two microcontrollers Arduino and ESP32 (Accelerometer).

### **V-Data Processing and Storage:**Methods for processing and storing data.

1**.Data Processing** in this project is divided into **local processing** which is performed in **server-side processing** on the Raspberry Pi.

On the **Raspberry Pi** server, the collected data is further processed after being received via MQTT or HTTP. The server enriches this data with additional metadata like timestamps and device identifiers, enabling more detailed logging and analysis. It then evaluates incoming data for specific conditions; for example, if the temperature exceeds 30°C, the server triggers an alert by performing predefined actions. This multi-level processing ensures efficient communication, minimizes network overhead, and enables real-time responsiveness to critical events.

2.**Data Storage** is centralized on the Raspberry Pi,which acts as a data hub by using **InfluxDB** for sensor data and MySQL for storing the ids. **Grafana** is usedfor efficient storage and visualization.

**InfluxDB (Time-Series Database):**

* Purpose: Store sensor data (e.g., temperature, motion events) with timestamps for easy querying.
  + Each data point includes:
    - Sensor type (e.g., temperature, humidity, attempts).
    - Timestamp.
    - Value.

**Grafana (Visualization Tool):**

* Connected to InfluxDB for real-time data visualization.
* Dashboards show:
  + Temperature and humidity trends over time.
  + RFID authentication logs.

In addition to InfluxDB, the project utilizes **MySQL** to store and manage RFID card IDs for authentication purposes. MySQL serves as a relational database to securely store the unique identifiers of authorized RFID cards. When an RFID card is scanned, the system retrieves the card's ID and compares it with the entries stored in the MySQL database to validate access. If the scanned ID matches an entry in the database, the system grants access; otherwise, it triggers an alert and denies the request.

### **VI-Automation Tasks:**Automation tasks implemented in NODE-RED

We will use MQTT to publish alerts and the esp32 will subscribe to it to make some actions (turn on/off a motor, a led,...)

**1. Sending email**

We send an email if there is an unauthorized attempt or a suspicion of hacking.

#### **2. Motor**

We turn on a motor to simulate the wind, if the temperature exceeds a certain threshold.

#### **3. Blue LED**

We turn on a blue LED to say if the temperature is under a certain threshold.

#### **4. RGB LED**

We will use a RGB LED to describe the humidity of the room, from blue to red.

#### **5. Storing in DB**

* We separate received data to store them in the right field and measurement.
* With the logic put in NodeRed we add into db the attempts for opening the door, like when they are authorized and unauthorized.

### VII-Challenges and Solutions

#### **1. Reliable Communication Across Protocols**

* Problem: Managing data flow across MQTT, HTTP, and BLE protocols can result in synchronization issues or message loss.
* Solution: Implement QoS (Quality of Service) in MQTT to ensure reliable delivery, and include retry logic in HTTP requests to handle network failures. For BLE, establish secure pairing to prevent unauthorized interference.

**2. Integration of Databases (MySQL and InfluxDB)**

* **Problem:** Managing two databases for different data types (time-series data in InfluxDB and RFID IDs in MySQL) can complicate operations.
* **Solution:** Use Node-RED flows to query and update each database as required. Ensure clear separation of tasks: InfluxDB for real-time sensor data and MySQL for static RFID authentication data.

#### **3. Real-Time Alerts**

* **Problem:** Delays in detecting and responding to critical events (e.g., motion detection or high temperature) could reduce system effectiveness.
* **Solution:** Use Node-RED’s event-driven flow to immediately trigger actions (e.g., sending an email or activating an alarm) upon receiving critical data.

**4.ESP32 Memory**

* The sketch of esp32 was so loaded so we need to use 2 ESP32 to fix the problem with memory.We used only one for BLE connection with Arduino Nano and then did a connection by using UART2 to connect with other ESP32.