IoT Smart Home System

Cruz Baños Omar Emiliano
Università di Bologna

Abstract— **This project simulates a smart home equipped with various sensors and actuators to monitor and automate daily activities. Data from sensors is collected by an ESP32 microcontroller and sent via MQTT to an online broker, HiveMQ. The data is then stored in an InfluxDB cloud database and visualized using Grafana.**

I. Introduction

The Internet of Things (IoT) is revolutionizing smart home technology by interconnecting everyday devices to the internet, enabling enhanced automation, monitoring, and control. In smart homes, IoT devices such as sensors, actuators, and smart appliances work together to provide real-time information and automated responses, significantly improving convenience, efficiency, and security. By optimizing energy usage, enhancing security, providing convenience, and ensuring health and wellness, IoT devices transform our living environments. This project demonstrates the implementation of an IoT-based smart home system that integrates various sensors and actuators with cloud services to monitor and control home conditions, showcasing the practical applications of IoT in creating more intelligent and efficient homes.

II. Project´s Architecture

The architecture of this project includes the integration of various hardware and software components to create a smart home system.

A. Schematics

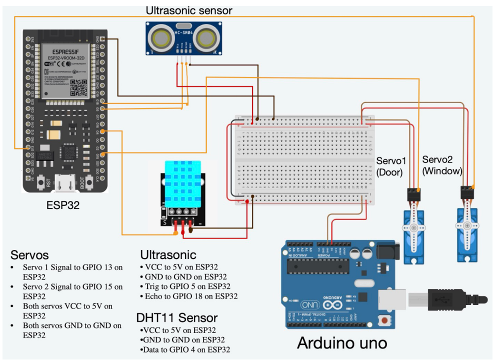


Fig. 1. Electronics schematics

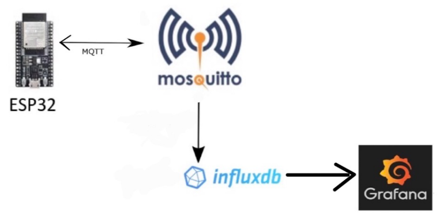


Fig. 2. Project Road

B. Data Generation

Using a DHT11 sensor to collect Temperature and Humidity values and an Ultrasonic sensor HC-SR04 to measure distance from the main entrance. Servomotors are use as actuators for opening and closing the window and the main entrance door.

C. Data Processing

With and ESP32 the data is processed.

D. Data Communication

The collected data from the ESP32 is sent to an MQTT broker in the Cloud, HiveMQ.

E. Data Storage

This data is stored in an InfluxDB database

F. Data Analytics

The data is downloaded to .csv document and then analyzed using the python library pandas or Excel

G. Application

The system also integrates with the OpenWeather API and a Telegram bot is used to send notifications

III. Project´s Implementation

The smart home system use a ESP32 microcontroller as the brain of the house. In order to get the smart house running we need to set up our microcontroller

- Hardware set up

ESP32 Microcontroller: The ESP32 serves as the main processing unit. It is selected for its Wi-Fi capabilities and easy porgramming.

Arduino Uno: It is used as a power source due to the low power source that the ESP32 can offer. Arduino gives 5V while the ESP32 3.3V, this is relevant because the ultrasonic sensor and the servomotors need 5V in order to work correctly

DHT11 Sensor: It is a digital temperature and humidity sensor. It is connected to the ESP32 to monitor environmental conditions. The data pin of the DHT11 is connected to the digital pin 4 on the ESP32.

Ultrasonic Sensor: The HC-SR04 ultrasonic sensor is used to measure distance and obstacles in its way, the trigger and echo pins are connected to the pins 5 and 18 respectively. This sensor acts as Ring tool for the door.

Servomotors: Two servomotors are used for controlling physical objects in the house, like windows or doors, They are connected to PWM-capable pins, 13 and 15, on the ESP32.

- Software Development

The software side is divided in two parts, the first one is the code used for getting and sending data from the sensors, the second part is the script used for sending the data to the cloud. Every data is sent directly to the cloud using a MQTT server, HiveMQ, and a Cloud data base, Influxdb, and that is why is very important to stablish a secure TLS conection using a CA Certificated between the ESP32 and the broker

I. ESP32 Code

The ESP32 is programmed using the Arduino IDE. The program is written in C++ and includes libraries for the use of Wi-Fi, MQTT, and sensor management.

Libraries and definitions: Two different libraries for WiFi are used WiFi.h and WiFiClientSecure.h, DHT libraries for the use of the DHT11 sensor, PubSubclient.h for the use of MQTT communication and ESP32Servo that allow us to control the servomotors. Pins are assigned aswell as the information neccesary to connect to the internet and the broker.

Setup Function: The WiFi conection is initialized and then the MQTT Client, DHT11 sensor and servomotors are also configured.

WiFi Function: Due to the poor conections that sometimes houses can have, in this function is implemented that the ESP32 try to reconnect constantly if no WiFi is connected, a similar function is implemented in the MQTT reconnect function.

MQTT Reconnect Function: Reconnects to the MQTT broker if the connection is lost.

Callback Function: Handles the incoming MQTT commands received either for the Telegram app or via the MQTT topic.

Loop Function: In the loop function data is sent periodically from the sensors to their respectives topics.

II. Python Script

The Python script collects sensor data from the MQTT broker, stores it in InfluxDB, and retrieves weather data from the OpenWeather API to later compare data. It also sends alerts via Telegram to the user indicating that somebody is at his front door and can receive command straight from the app.

Libraries and definitions: Here are imported necessary libraries and are configured OpenWeather API, InfluxDB, and the MQTT client.

Get OpenWeather function: Data of temperature and humidity from the Open Weather API is collected. And Publish_openweather_temperature publish data to the corresponding topic.

Write data function: Writes data to Influxdb

MQTT Callbacks: on_connect and on_message functions handle connection to the MQTT broker and processes incoming messages.

Telegram alert: Sends alerts to the user´s telegram based on the sensors data

Command Handling: send_mqtt_command and handle_message processed commands receive via Telegram and send the corresponding MQTT meesage

Threads: these allow the program to handle multiple tasks simultaneously, ensuring the Telegram bot remains responsive while continuously fetching and publishing weather data.

With everything working fine we have a functional Smart home system that senses the temperature and humidity inside the house, and depending of the values, the windows can be open or close automatically. And while you are away from home you can still detect if someone is in front of your house by getting and alert in your phone and even open the door for them..

IV. Challenges and Solutions

The first problem that I encountered was the realization if it was best to manage the project locally or in the cloud, I opted to do everything in the cloud so it will be similar to the real use of the project. And I also could work with the project without the need of being on the same WiFi that the sensors. The other problem was to find that in order to do multiple task in a single program I needed to assigned individual threads for every task so they won´t interfiere with eachother.

V. Results

The data that is exported to the database can be better visualizaed through dashboards in Grafana (Fig.3)

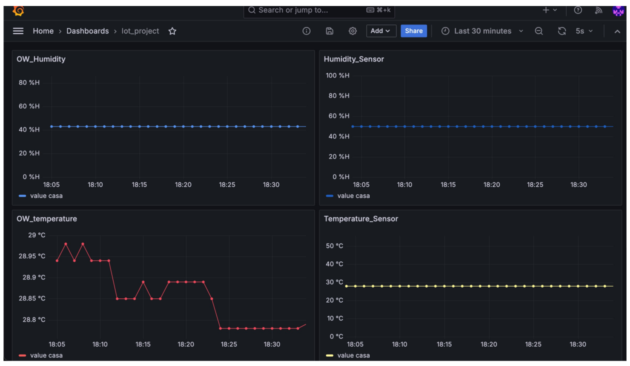


Fig. 3. Dashboard from Grafana

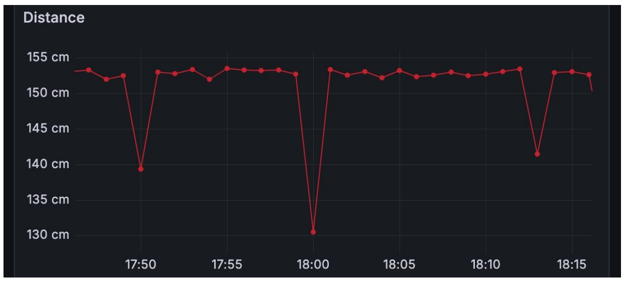


Fig. 4. Dashboard of the Ultrasonic sensor

In this Graphic (fig.4) of the ultrasonic sensor, we can see clearly that we had three detected presence infront of our main entrance and those moments were stored in our data base and can be seen in Grafana at every moment.

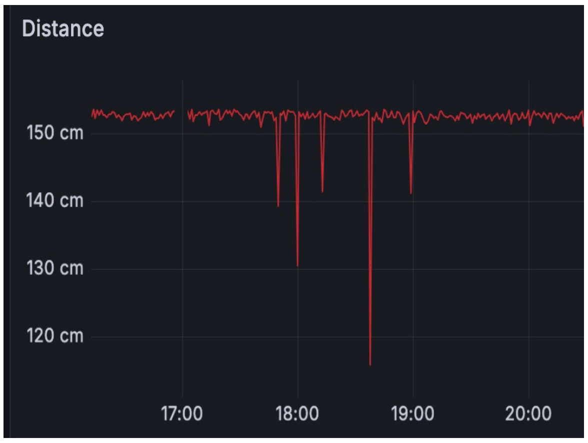
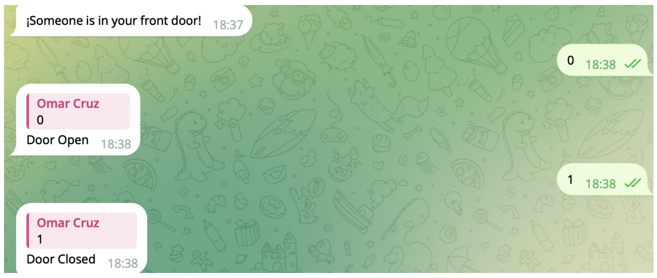


Fig. 5. Dashboard of the ultrasonic sensor were presence was detected

This is another example, and we can see the alert that has been sent to the user, and the commands that can be send to open and close the door from anywhere (fig.5)



¡Someone is in your front door! 18:37

0 18:38

Omar Cruz
0
Door Open 18:38

1 18:38

Omar Cruz
1
Door Closed 18:38

Fig. 6. Example of the notifications that the user gets and the commands that can be sent

It is possible to send data from the MQTT, in this occasion I sent a data of 22°C (fig.6), then the program opened the window and closed immediately and also sent and alert to the user (fig.7)

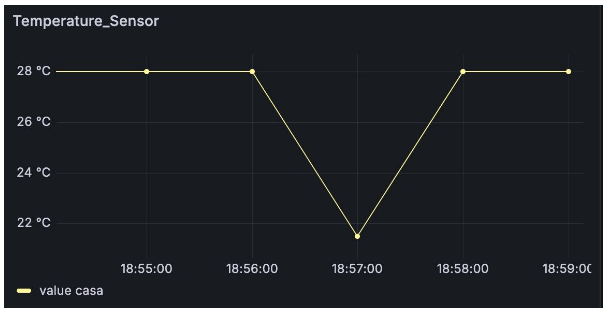
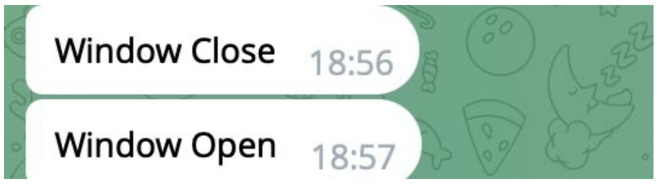


Fig. 7. Dashboard of the Temperature sensor



Window Close 18:56

Window Open 18:57

Fig. 8. Example of notifications that the user gets

You can also export your data to a .csv file and analyze it in pandas or excel so you can preddict data.

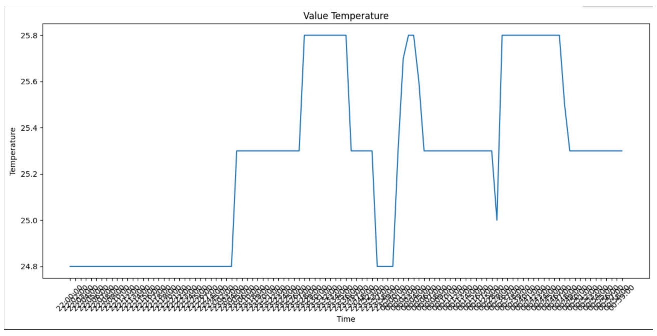
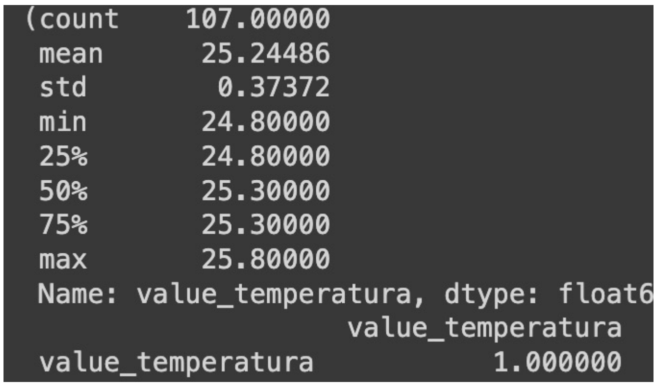


Fig. 9. Graphicated with pandas



```
(count    107.00000
 mean      25.24486
 std        0.37372
 min       24.80000
 25%       24.80000
 50%       25.30000
 75%       25.30000
 max       25.80000
 Name: value_temperatura, dtype: float6
                    value_temperatura
 value_temperatura           1.000000
```

Fig. 10. Information about the dataset

References

[1] "Eclipse Mosquitto Documentation." [Online]. Available: https://mosquitto.org/documentation/.

[2] "Getting Started with InfluxDB Cloud Dedicated." [Online]. Available: https://docs.influxdata.com/influxdb/cloud-dedicated/get-started/.

[3] "Fundamentals | Grafana documentation." [Online]. Available: https://grafana.com/docs/grafana/latest/fundamentals/.

[4] "Telegram Bot API." [Online]. Available: https://core.telegram.org/bots/api.