Università Degli Studi di Messina

Engineering And Computer Science

Industrial IoT Project

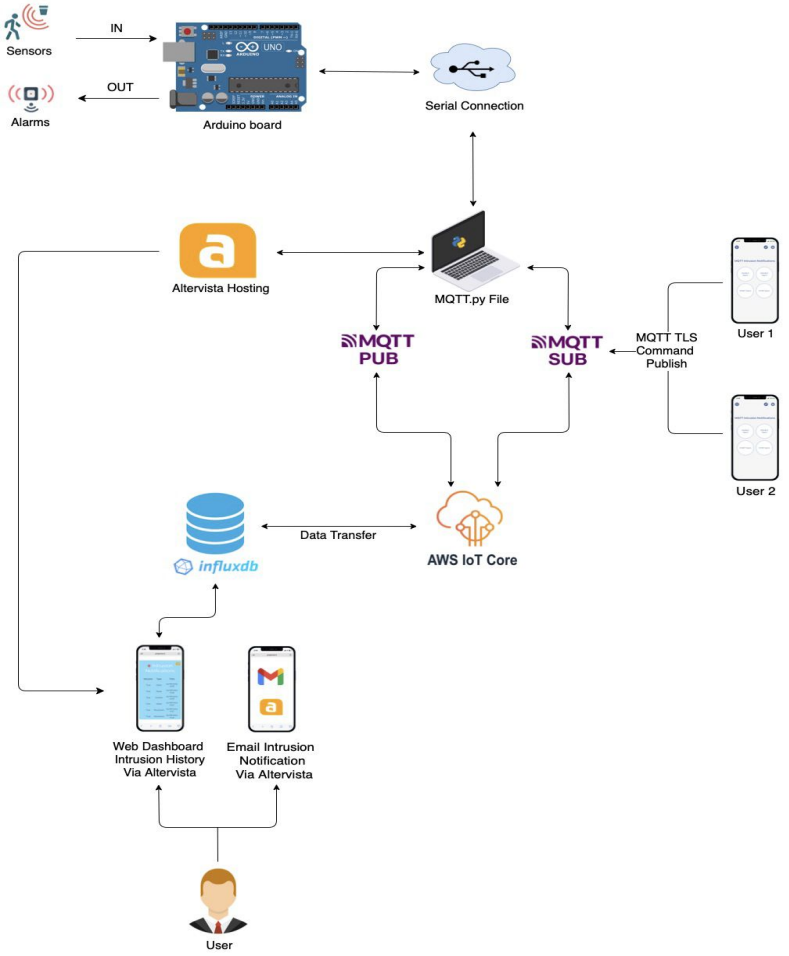
IoT-based Smart Intrusion Detection And Notification System For Industry 4.0

Davide Giuseppe Allegra
Cateno Gabriele D'Alì

Overview

Features:

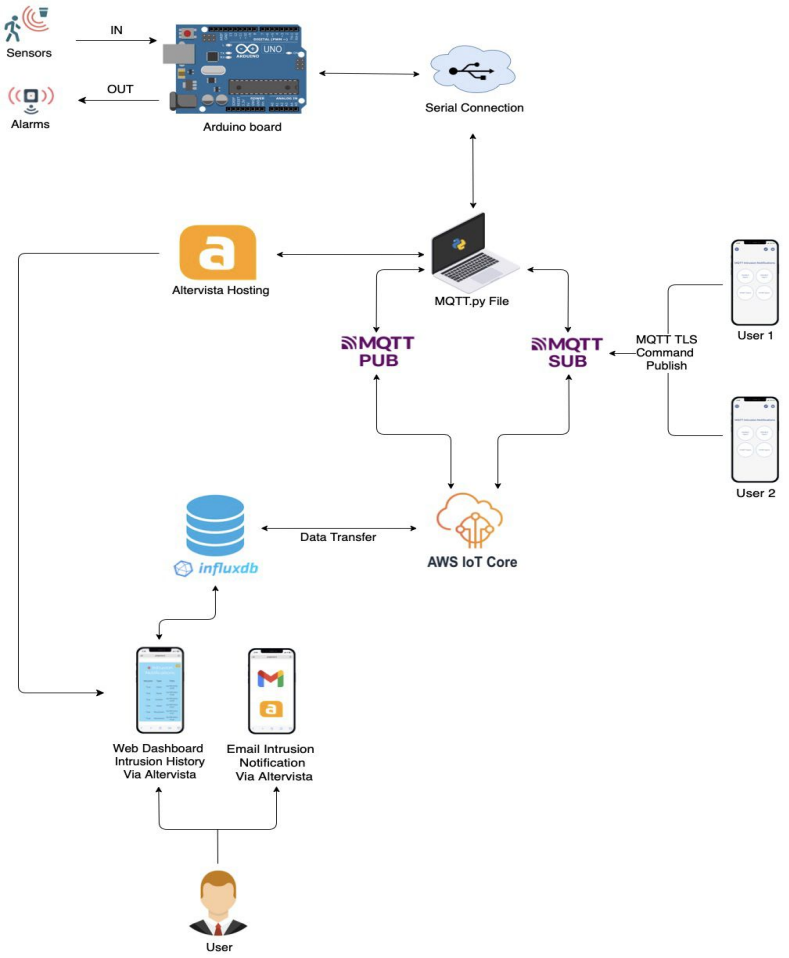
- Intrusion Detection Sensor
- Alarms and Notifications
- MQTT Protocol
- Remote Control
- Data Storage
- System Activity Monitoring



Overview

Technologies Used:

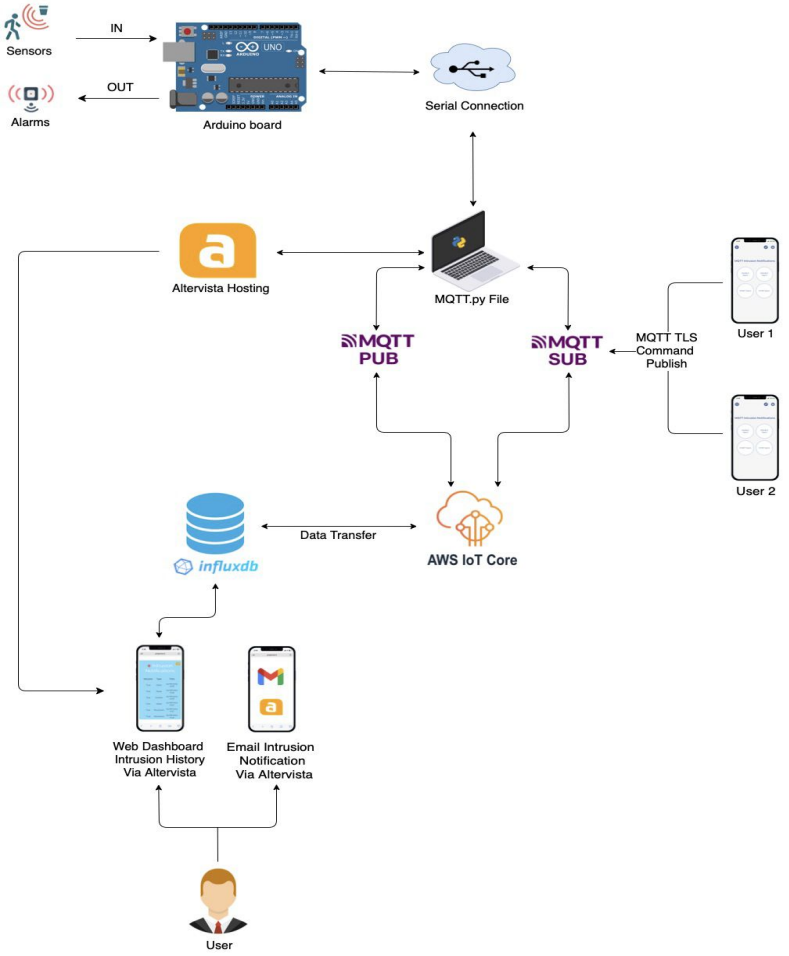
- Arduino IDE
- PyCharm IDE
- Hosting Altervista
- AWS
- MQTT
- TLS IoT Tool
- InfluxDB Cloud
- Draw.io



Overview

Programming Languages Used:

- C
- Python
- HTML
- CSS
- Java Script
- InfluxQL/SQL



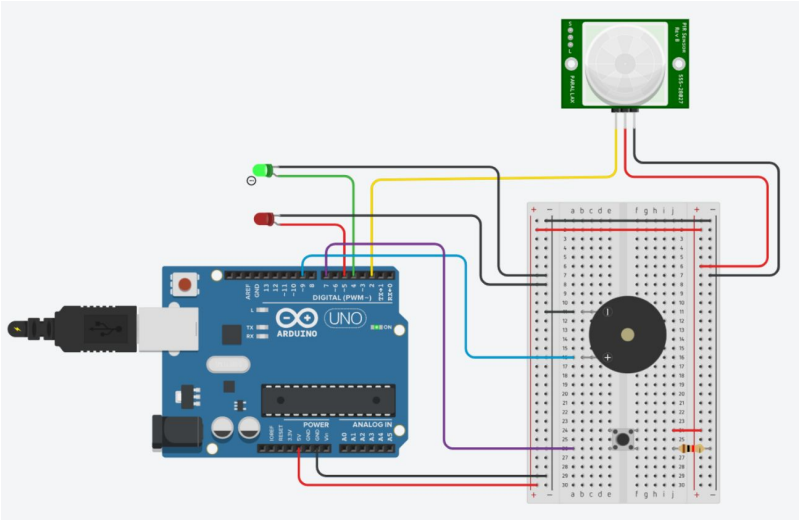
Arduino

Key Features

- Open-Source electronic platform
- Easy of use
- Versatility
- Lightweight

Weaknesses

- Limited power and memory
- Not suitable for real-time applications
- Limited connectivity



Arduino

Setup

- For the first time wait for one minute allowing PIR sensor stable itself
- Define variables
- Set pinMode and Serial

Loop

- Serial data communication
- MQTT messages checking
- Performing action based on commands received through MQTT
- Switch status and alarm LEDs on/off

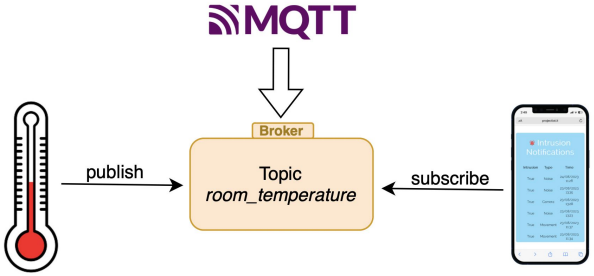
```
#include <SoftwareSerial.h>

SoftwareSerial mySerial(10, 11); // RX, TX

void setup()
{
    mySerial.begin(9600);
    Serial.begin(9600);
}

void loop()
{
    if(deviceIsReady)
    {
        if(alarmEnabled)
        {
            int motionValue = digitalRead(PIR);

            if(motionValue == HIGH)
            {
                if(alarmTriggered == false)
                {
                    mySerial.println(1);
                    Serial.println(1);
                    startAlarm();
                }
            }
            else
            {
                mySerial.println(0);
                Serial.println(0);
            }
        }

        MQTT_checkMessages();
        delay(1000);
    }
}
```

```
void MQTT_checkMessages()
{
    if(Serial.available() > 0)
    {
        String message = readSerialMessage();

        if(message.length() > 0)
        {
            // Perform some action based on the message content
            if(message.equals("ENABLE_ALARM"))
            {
                changeSystemStatus(1);
            }
            else if(message.equals("DISABLE_ALARM"))
            {
                changeSystemStatus(0);
            }
            else if(message.equals("START_ALARM"))
            {
                startAlarm();
            }
            else if(message.equals("STOP_ALARM"))
            {
                stopAlarm();
            }
        }
    }
}
```

MQTT

Key Features:

- Lightweight
- Broker
- Publish/Subscribe
- Topic
- QoS
- Retain
- Last Will
- Security
- Scalability



Example: Room Temperature

	PUB	SUB
Arduino	Write	Read
User 1	Read	Write
User 2	Read	Write

MQTT

Client Creation and Configuration

```python
# Config.py
class Config:
    # Config AWS
    endpoint = "apv9omwb522qd-ats.iot.eu-central-1.amazonaws.com"
    client_id = "ArduinoUnoR3"
    root_ca = "keys/AWS-Root-RSA.pem"
    private_key = "keys/AWS-private.pem.key"
    certificate = "keys/AWS-certificate.pem.crt"


# MQTT.py
from AWSIoTPythonSDK.MQTTLib import AWSIoTMQTTClient

# MQTT topic
sub_topic = "ArduinoSensors/sub"
pub_topic = "ArduinoSensors/pub"
QoS = 0

# Create a MQTT client
my_mqtt_client = AWSIoTMQTTClient(Config.client_id)
my_mqtt_client.configureEndpoint(Config.endpoint, 8883)
my_mqtt_client.configureCredentials(Config.root_ca, Config.private_key, Config.certificate)

# Broker AWS IoT connection
my_mqtt_client.connect()
print(">> MQTT broker connected")
```

Publish

```python
# Receiving Arduino data
if ser.in_waiting > 0:
    alarm_triggered = ser.readline().decode().strip()
    if alarm_triggered == '1':
        # Publishing sensor data
        sensor_data = {
            "message": "intrusion detected"
        }
        # Convert payload in JSON format
        my_mqtt_client.publish(pub_topic, json.dumps(sensor_data), QoS)
```

Subscribe

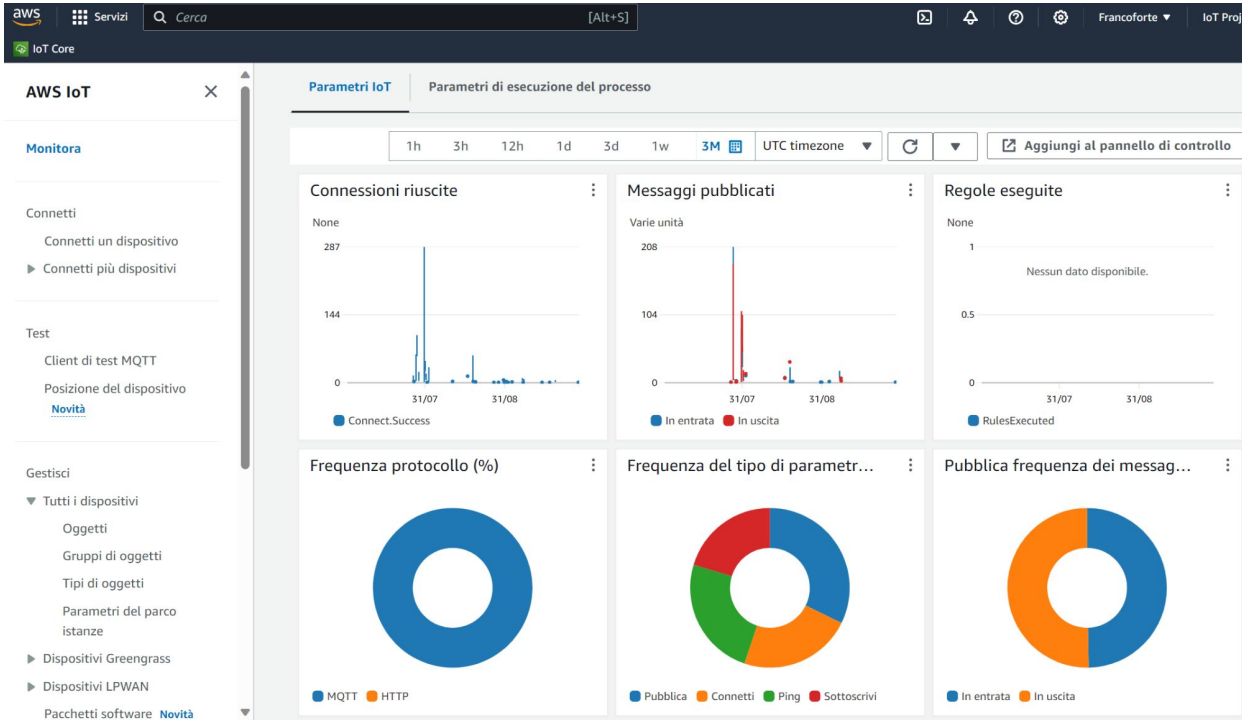
```python
# Topic subscription
my_mqtt_client.subscribe(sub_topic, QoS, readMessages)
print(">> Client subscribed to ArduinoSensors topic")

# To loop message broadcasting
while True:
    time.sleep(1)
```

AWS

Amazon Web Service

Key Features:

- Services offered
- Availability
- Scalability
- Resource manager
- Security
- Global Reach
- Ecosystem
- AI Tools



AWS

Amazon Web Service

Device Creation:

- Thing
- Certificate
- Policy

Device Connection:

- Client ID
- MQTT Endpoint
- Certificate
- AWS Root CA
- Private Key

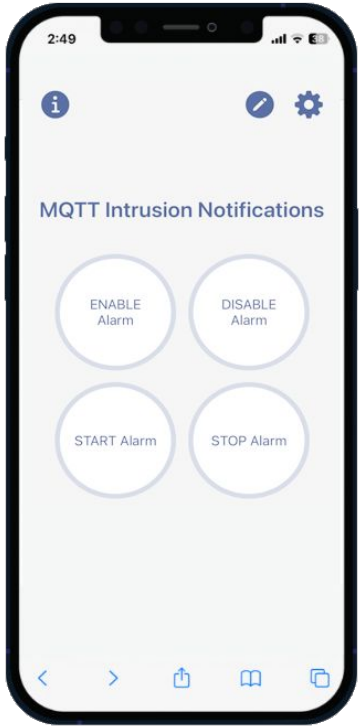
Effetto della policy	Operazione della policy	Risorsa della policy
Allow	iot:Connect	arn:aws:iot:eu-central-1:014811469566:client/ArduinoUnoR3
Allow	iot:Publish	arn:aws:iot:eu-central-1:014811469566:topic/ArduinoSensors/pub
Allow	iot:Publish	arn:aws:iot:eu-central-1:014811469566:topic/ArduinoSensors/sub
Allow	iot:Receive	arn:aws:iot:eu-central-1:014811469566:topic/ArduinoSensors/pub
Allow	iot:Receive	arn:aws:iot:eu-central-1:014811469566:topic/ArduinoSensors/sub
Allow	iot:Subscribe	arn:aws:iot:eu-central-1:014811469566:topicfilter/ArduinoSensors/pub
Allow	iot:Subscribe	arn:aws:iot:eu-central-1:014811469566:topicfilter/ArduinoSensors/sub

TLS IoT Tool

```python
def readMessages(client, userdata, message):
    # Function called when a message is received
    payload = json.loads(message.payload)
    received_message = payload["message"]
    print(f">> MQTT Received Message: {received_message}")
    if received_message == "ENABLE_ALARM" or received_message == "DISABLE_ALARM" ...
        ... or received_message == "START_ALARM" or received_message == "STOP_ALARM":
        # Sending a welcome message to the board
        ser.write(received_message.encode())
```

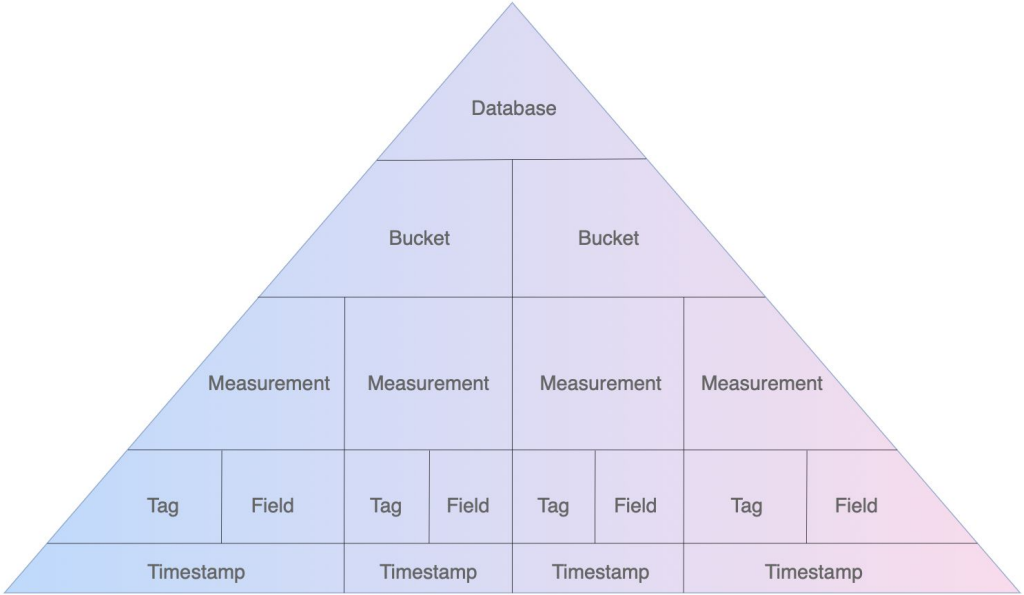
```
void stopAlarm()
{
    if(alarmTriggered)
    {
        alarmTriggered = false;
        digitalWrite(RedLED, LOW);
        noTone(buzzer);
    }
}
```

```
void startAlarm()
{
    if(alarmEnabled && alarmTriggered == false)
    {
        alarmTriggered = true;
        digitalWrite(RedLED, HIGH);
        for(int i=0; i<25; i++)
        {
            MQTT_checkMessages();
            checkButtonStatus();
            if(alarmTriggered && alarmEnabled)
            {
                tone(buzzer, 700);
                delay(200);
                noTone(buzzer);
                delay(200);
            }
            else break;
        }
        digitalWrite(RedLED, LOW);
        stopAlarm();
    }
}
```



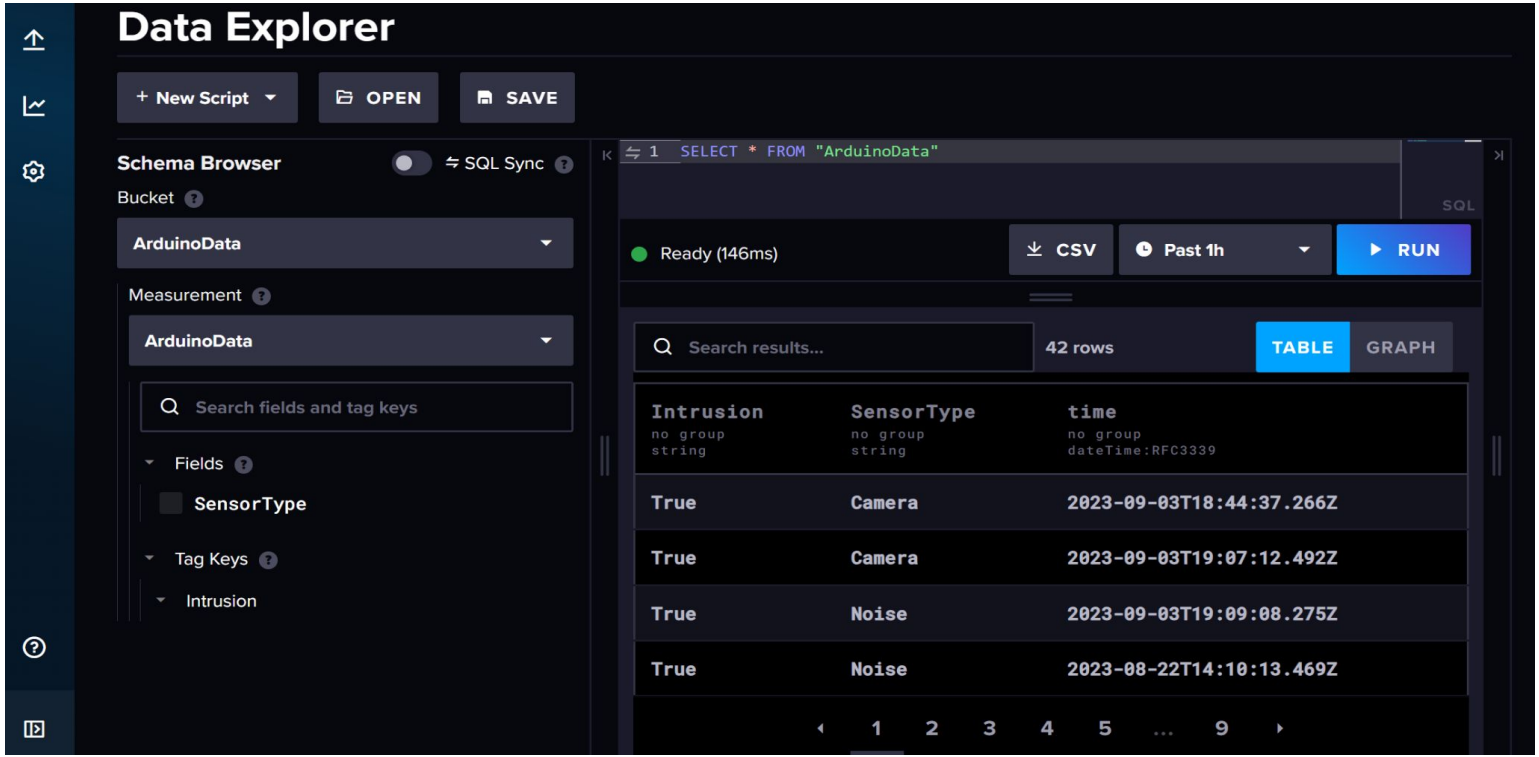
InfluxDB

InfluxDB Cloud on AWS




```python
# Config.py
class Config:
    token_influx = "H4cem9xbCj-7rT165wBxmHEDNebCg2Fq1bZ0LGY9cDlGGSP7zwKs"
                +  "TsEeb8Em7eNcPX6RD6VdgqXU9dwUCnx3-Q=="
    org_influx = "University of Messina"
    host_influx = "https://eu-central-1-1.aws.cloud2.influxdata.com"
    bucket = "ArduinoData"


# MQTT.py
# Simulating other sensors
types = ["Movement", "Noise", "Camera"]
sensortype = random.choice(types)

# Saving data into InfluxDB
data_influx = {
    "data": {
        "Intrusion": "True",
        "SensorType": sensortype
    }
}

for key in data_influx:
    point = (
        Point("ArduinoData")
        .tag("Intrusion", data_influx[key]["Intrusion"])
        .field("SensorType", data_influx[key]["SensorType"])
    )
client.write(database=Config.bucket, record=point)
```

InfluxDB


Data Explorer
+ New Script
OPEN
SAVE
Schema Browser
SQL Sync
Bucket
ArduinoData
Measurement
ArduinoData
Search fields and tag keys
Fields
SensorType
Tag Keys
Intrusion
1 SELECT * FROM "ArduinoData"
SQL
Ready (146ms)
CSV
Past 1h
RUN
Search results...
42 rows
TABLE
GRAPH
Intrusion
no group
string
SensorType
no group
string
time
no group
dateTime:RFC3339
True Camera 2023-09-03T18:44:37.266Z
True Camera 2023-09-03T19:07:12.492Z
True Noise 2023-09-03T19:09:08.275Z
True Noise 2023-08-22T14:10:13.469Z
1 2 3 4 5 ... 9

Web Page

Intrusion History

Languages:

- HTML
- CSS
- JavaScript

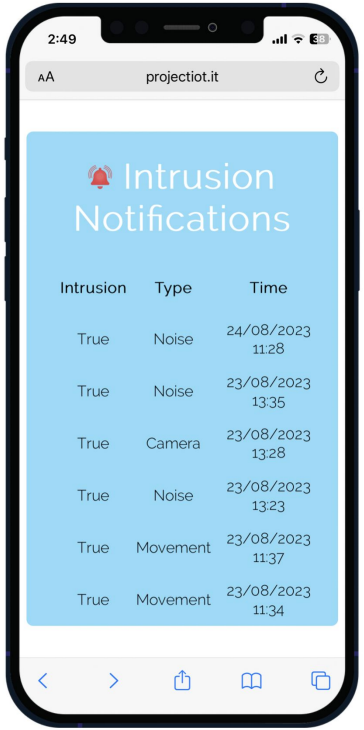
Hosting:

- Altervista

Features:

- Async
- Autorefresh
- Remote access

```
var returnJSON = [];
const token = "pEw624Urc5zDDETiGu5gbJcnsVElszwsD16p6bIIC31i4zs0rxX3bGT9cZkz-Mcxc04CcC-GfVDIVEPbKKp0AA==";
const org = "University of Messina";
const bucket = "ArduinoData";

const url = `https://eu-central-1-1.aws.cloud2.influxdata.com/api/v2/query?org=${encodeURIComponent(org)}`;
const query = `from(bucket:"${bucket}") |> range(start: -30d) |> filter(fn: (r) => r["_measurement"] == "ArduinoData")`
            + `|> sort(columns: ["_time"], desc: true)`;

const requestBody = JSON.stringify({
  query: query
});

fetch(url, {
  method: "POST",
  headers: {
    "Authorization": `Token ${token}`,
    "Content-Type": "application/json"
  },
  body: requestBody
})
.then(response => response.text())
.then(data => {});
```



Intrusion Emails

Python Side:

```python
if alarm_triggered == '1':
    # Sending email to the admins
    current_time = datetime.datetime.now()
    today = current_time.strftime('%d/%m/%Y %H:%M')
    emailBody = f"An intrusion was detected by the {sensortype} sensor at {today}"
    sendEmail(emailBody)
```

```python
def sendEmail(body):
    receivers = ",".join(Config.emails)
    url = f'{Config.altervista_host}/iot/sendEmail.php'
         + '?receivers={receivers}&subject={Config.subject}&body={body}<br><br>'
    response = requests.get(url)
```

PHP Side (Altervista):

```php
$emails = htmlentities($_GET['receivers'], ENT_QUOTES);
$subject = htmlentities($_GET['subject'], ENT_QUOTES);
$body = htmlentities($_GET['body'], ENT_QUOTES);

$decodedSubject = html_entity_decode($subject, ENT_QUOTES);
$decodedBody = html_entity_decode($body, ENT_QUOTES);

$headers = "From: IoT-Notification \r\n";

$receivers = array();
$emailList = explode(',', $emails);

foreach($emailList as $email)
{
    $receivers[] = trim($email);
}

$n = 0;
foreach($receivers as $receiver)
{
    if(mail($receiver, $decodedSubject, $decodedBody, $headers))
    {
        $n++;
    }
    sleep(1);
}

http_response_code(200);
```