FileSystemPoP

Testes mqtt e mqtts

1. mqtt://192.168.1.12:1883
   {"movement":1,"door":0,"voltage_detector":1,"temperature_obj":22.830011,"temperature_env":23.029993,"unix_time":1724358055}
   [2178247][I][StatesManagerService.cpp:1019] mqttResourcedata(): [StatesService] payloadResourcedata: {"movement":1,"door":0,"voltage_detector":1,"temperature_obj":22.869989,"temperature_env":23.010004,"unix_time":1724357845}

   - Check "Metrics"
   - Check Mqtt config "Client ID"

2. mqtts://a1t0ofnrl6u1vk-ats.iot.us-east-1.amazonaws.com:8883
   Received message '{"movement":1,"door":0,"voltage_detector":1,"temperature_obj":22.809992,"temperature_env":22.990015,"unix_time":1724357015}' on topic 'sensor_site/b0a732810a30/uplink'

   - Check "Metrics"
   - Check Mqtt config "Client ID"

Write Flash

```
#!/bin/bash
# Command to flash esp32

esptool.py --chip esp32 --port "/dev/ttyUSB0" --baud 115200 \
--before default_reset --after hard_reset \
write_flash -z --flash_mode dio --flash_freq 40m --flash_size 4MB \
0x1000 bootloader.bin \
0x8000 partitions.bin \
0xe000 boot_app0.bin \
0x10000 firmware_028.bin \
0x3d0000 spiffs.bin
```

Arquivos de configuração do firmware

```
file: /config/apSettings.json
{
    "provision_mode": 1,
    "ssid": "SmartIoT-b0a732810a30",
    "password": "smartiot",
    "channel": 1,
    "ssid_hidden": false,
    "max_clients": 4,
    "local_ip": "192.168.4.1",
    "gateway_ip": "192.168.4.1",
    "subnet_mask": "255.255.255.0"
}
```

file: /config/mqttSettings.json

```
{
    "enabled": false,
    "uri": "mqtts://a1t0ofnrl6u1vk-ats.iot.us-east-1.amazonaws.com:8883",
    "username": "",
    "password": "",
    "client_id": "esp32-b0a732810a30",
    "keep_alive": 120,
    "clean_session": true
}
```

file: /config/ntpSettings.json

```
{
    "enabled": true,
    "server": "time.google.com",
    "tz_label": "Europe/Berlin",
    "tz_format": "GMT0BST,M3.5.0/1,M10.5.0"
}
```

file: /config/securitySettings.json

```
{
    "jwt_secret": "784fc466-5c053861",
    "users": [
        {
            "username": "admin",
            "password": "admin",
            "admin": true
        },
        {
            "username": "guest",
            "password": "guest",
            "admin": false
        }
    ]
}
```

file: /config/wifiSettings.json

```
{
    "hostname": "esp32-b0a732810a30",
    "priority_RSSI": true,
    "wifi_networks": [
        {
            "ssid": "LIVE TIM_3614",
            "password": "Ft52Ebce",
            "static_ip_config": false
        }
    ]
}
```

file: /config/io0Settings.json

```
{
    "enabled": false,
    "client_id": "esp32-b0a732810a30",
    "i2c_address": 161,
    "resource_key": 1,
    "simulator_enabled": false,
    "simulator_frequency": 1
}
```

file: /config/io1Settings.json

```
{
    "enabled": false,
    "client_id": "esp32-b0a732810a30",
    "i2c_address": 162,
    "resource_key": 1,
    "simulator_enabled": false,
    "simulator_frequency": 1
}
```

```
file: /config/io2Settings.json
{
    "enabled": false,
    "client_id": "esp32-b0a732810a30",
    "i2c_address": 163,
    "resource_key": 1,
    "simulator_enabled": false,
    "simulator_frequency": 1
}
```

```
file: /config/brokerSettings.json
{
    "mqtt_path": "smartiotassistant/light/b0a732810a30",
    "name": "light-b0a732810a30",
    "unique_id": "light-b0a732810a30"
}
```

Atualizar o arquivo `spiffs.bin` na partição "spiffs, data, spiffs, 0x3D0000,0x20000" dedinido em partitions.csv

1. Verificar se a extensão `PlatformIO` esta instalada corretamente no projeto
2. Abrir o terminal `PlatformIO CLI - scripts`. No menu principal clicar em `Menu View -> Command Palette -> PlatformIO: New Terminal`
3. Atualizar os arquivos na pasta `data`
4. Gerar um novo `spiffs.bin` a partir dos arquivos atualizados na pasta `data` do PlatformIO com o comando `pio run --target buildfs`.

```
FileSystemPoP]$ pio run --target buildfs
Processing esp32-poe-iso (board: esp32-poe-iso; framework: arduino; platform: espressif32
------------------------------------------------------------
Verbose mode can be enabled via `-v, --verbose` option
CONFIGURATION: https://docs.platformio.org/page/boards/espressif32/esp32-poe-iso.html
PLATFORM: Espressif 32 (6.6.0) > OLIMEX ESP32-PoE-ISO
HARDWARE: ESP32 240MHz, 320KB RAM, 4MB Flash
DEBUG: Current (cmsis-dap) External (cmsis-dap, esp-bridge, esp-prog, iot-bus-jtag, jlink
PACKAGES:
 - framework-arduinoespressif32 @ 3.20014.231204 (2.0.14)
 - tool-esptoolpy @ 1.40501.0 (4.5.1)
 - tool-mkspiffs @ 2.230.0 (2.30)
 - toolchain-xtensa-esp32 @ 8.4.0+2021r2-patch5
LDF: Library Dependency Finder -> https://bit.ly/configure-pio-ldf
LDF Modes: Finder ~ chain, Compatibility ~ soft
Found 33 compatible libraries
Scanning dependencies...
No dependencies
Building in release mode
Building FS image from 'data' directory to build/esp32-poe-iso/spiffs.bin
/config/io2Settings.json
/config/securitySettings.json
/config/brokerSettings.json
/config/apSettings.json
/config/io0Settings.json
/config/ntpSettings.json
/config/wifiSettings.json
/config/io1Settings.json
/config/mqttSettings.json
/certs/private.pem.key
/certs/certificate.pem.crt
/certs/AmazonRootCA1.pem
========== [SUCCESS] Took 0.56 seconds ==========
```

Este comando cria o arquivo `spiffs.bin` na pasta `build/esp32-poe-iso/` contendo os arquivos da pasta `data` dentro desto projeto.

5. Gravar o arquivo spiffs.bin, gerado no passo anterior, no ESP32 e fazer com que o firmware utilize esses novos arquivos de configuração.

A pasta `FileSystemPoP/scripts` contém os scripts utilizados para as gravações no ESP32.

Gravando o `spiffs.bin`

Depois de gerar o `spiffs.bin`, é necessário gravá-lo no ESP32 utilizando o `esptool.py` através do script adequado:

```
FileSystemPoP\scripts]$ python spiffs_update_flash_script.py
esptool.py v4.5.1
Serial port /dev/ttyUSB0
Connecting....
Chip is ESP32-D0WD-V3 (revision v3.1)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
Crystal is 40MHz
MAC: b0:a7:32:81:0a:30
Uploading stub...
Running stub...
Stub running...
Configuring flash size...
Flash will be erased from 0x003d0000 to 0x003effff...
Compressed 131072 bytes to 4841...
Wrote 131072 bytes (4841 compressed) at 0x003d0000 in 1.2 seconds (effective 874.0 kbit/s
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
ESP32 successfully flashed.
```

Este comando grava apenas o arquivo `spiffs.bin` no endereço correto (`0x3D0000`), conforme definido na tabela de partições.

Comportamento do Firmware após Atualização do SPIFFS

Depois de gravar o novo `spiffs.bin`, o ESP32 será resetado (através da inclusão do parâmetro `--after hard_reset` no comando `esptool.py`) através do script de acionamento `spiffs_flash_script.py`. Após o reset o firmware começará a

execução considerandoos arquivos atualizados.

Comportamento esperado:

1. **Reset do Firmware:** Quando o ESP32 reinicia, o firmware é reinicializado e, durante a inicialização, o código do firmware deve montar o sistema de arquivos SPIFFS e acessar os arquivos de configuração que foram gravados com o novo `spiffs.bin`.
2. **Leitura dos Arquivos de Configuração:** Se o firmware foi projetado para ler arquivos de configuração do SPIFFS durante a inicialização, ele deverá carregar as novas configurações automaticamente.
3. **Manutenção do Funcionamento:** Desde que o firmware seja capaz de lidar com as novas configurações e que os arquivos de configuração tenham o formato esperado, ele deverá continuar funcionando corretamente com as novas configurações.

Considerações:

- **Validação dos Arquivos de Configuração:** É importante que o firmware tenha uma lógica para validar os arquivos de configuração. Isso garante que o sistema não falhe ao tentar carregar uma configuração inválida ou corrompida.

Atualizar do arquivo `firmware.bin` na partição "app0, app, ota_0, 0x10000, 0x1E0000," definida em partitions.csv

1. Verificar se a extensão `PlatformIO` esta instalada corretamente no projeto
2. Abrir o terminal `PlatformIO CLI - scripts`. No menu principal clicar em `Menu View -> Command Palette -> PlatformIO: New Terminal`
3. Atualizar o arquivo `firmware.bin` na pasta `"../firmware_bins/update_firmware/firmware.bin"`

```
FileSystemPoP\scripts]$ python firmware_update_flash_script.py
esptool.py v4.5.1
Serial port /dev/ttyUSB0
Connecting....
Chip is ESP32-D0WD-V3 (revision v3.1)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
Crystal is 40MHz
MAC: b0:a7:32:81:0a:30
Uploading stub...
Running stub...
Stub running...
Configuring flash size...
Flash will be erased from 0x00010000 to 0x001b1fff...
Compressed 1708672 bytes to 1156900...
Wrote 1708672 bytes (1156900 compressed) at 0x00010000 in 101.8 seconds (effective 134.2 k
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
ESP32 successfully flashed with firmware.
```

Este script grava o arquivo `firmware.bin` no ESP32.

Atualizar todos os arquivo nas partições definidas em partitions.csv

1. Verificar se a extensão `PlatformIO` esta instalada corretamente no projeto
2. Abrir o terminal `PlatformIO CLI - scripts`. No menu principal clicar em `Menu View -> Command Palette -> PlatformIO: New Terminal`
3. Atualizar o arquivo `firmware.bin` na pasta
   `"../firmware_bins/update_full_firmware/boot_app0.bin"`
   `"../firmware_bins/update_full_firmware/bootloader.bin"`
   `"../firmware_bins/update_full_firmware/firmware.bin"`
   `"../firmware_bins/update_full_firmware/partitions.bin"`
   `"../firmware_bins/update_full_firmware/spiffs"`

```
FileSystemPoP\scripts]$ python full_firmware_update_flash_script.py
esptool.py v4.5.1
Serial port /dev/ttyUSB0
Connecting.....
Chip is ESP32-D0WD-V3 (revision v3.1)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
Crystal is 40MHz
MAC: b0:a7:32:81:0a:30
Uploading stub...
Running stub...
Stub running...
Configuring flash size...
Flash will be erased from 0x00001000 to 0x00005fff...
Flash will be erased from 0x00008000 to 0x00008fff...
Flash will be erased from 0x0000e000 to 0x0000ffff...
Flash will be erased from 0x00010000 to 0x001b1fff...
Flash will be erased from 0x003d0000 to 0x003effff...
Compressed 17536 bytes to 12202...
Wrote 17536 bytes (12202 compressed) at 0x00001000 in 1.5 seconds (effective 95.4 kbit/s)
Hash of data verified.
Compressed 3072 bytes to 146...
Wrote 3072 bytes (146 compressed) at 0x00008000 in 0.1 seconds (effective 261.7 kbit/s)..
Hash of data verified.
Compressed 8192 bytes to 47...
Wrote 8192 bytes (47 compressed) at 0x0000e000 in 0.2 seconds (effective 408.2 kbit/s)...
Hash of data verified.
Compressed 1708672 bytes to 1156887...
Wrote 1708672 bytes (1156887 compressed) at 0x00010000 in 101.8 seconds (effective 134.2 
Hash of data verified.
Compressed 131072 bytes to 4841...
Wrote 131072 bytes (4841 compressed) at 0x003d0000 in 1.2 seconds (effective 875.8 kbit/s
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
ESP32 successfully flashed.
```

Este script grava os arquivos `boot_app0.bin, bootloader.bin, firmware.bin, partitions.bin, spiffs` no ESP32.

Aqui está uma descrição dos parâmetros usados no script `full_firmware_update_flash_script.py`:

1. **`esptool_path`**:
    - **Descrição**: Esta variável armazena o caminho para o utilitário `esptool.py`, que é usado para gravar firmware em dispositivos ESP32.
    - **Detalhes**: O caminho é expandido usando `os.path.expanduser()` para converter o `~` (diretório home) no caminho completo.
2. **`command`**:
    - **Descrição**: Esta lista armazena todo o comando a ser executado para gravar o ESP32.
    - **Parâmetros**:
        - `"python3"`: Especifica o interpretador Python para executar o `esptool.py`.
        - `esptool_path`: O caminho para o `esptool.py`.
        - `--chip "esp32"`: Indica que o chip alvo é um ESP32.
        - `--port "/dev/ttyUSB0"`: Especifica a porta serial usada para comunicação com o ESP32. Ajuste isso com base na sua porta real.
        - `--baud "115200"`: Define a taxa de transmissão para comunicação. `115200` é um padrão comum.
        - `--before "default_reset"`: Especifica o método de reset antes da gravação. `"default_reset"` reinicia o chip antes da operação de gravação.
        - `--after "hard_reset"`: Especifica o método de reset após a gravação. `"hard_reset"` reinicia o chip após a conclusão da gravação.
        - `write_flash`: O comando para gravar dados na memória flash do ESP32.
        - `"-z"`: Habilita a compressão para reduzir o tamanho dos dados a serem gravados.
        - `--flash_mode "dio"`: Define o modo de gravação. `"dio"` é um padrão comum.

    - `--flash_freq "40m"`: Define a frequência de gravação. `40m` indica 40 MHz.
    - `--flash_size "4MB"`: Especifica o tamanho da memória flash do chip ESP32. `4MB` é um tamanho comum para chips ESP32.
3. **Endereços de Memória e Caminhos dos Arquivos**:
    - `"0x1000", bootloader_path`: Grava o binário do bootloader no endereço `0x1000`.
    - `"0x8000", partitions_path`: Grava o binário da tabela de partições no endereço `0x8000`.
    - `"0xe000", boot_app0_path`: Grava o binário da aplicação de boot no endereço `0xe000`.
    - `"0x10000", firmware_path`: Grava o binário principal do firmware no endereço `0x10000`.
    - `"0x3d0000", spiffs_path`: Grava o binário do SPIFFS no endereço `0x3d0000`.

Este script é projetado para gravar vários componentes (bootloader, tabela de partições, firmware da aplicação, SPIFFS) em um chip ESP32 usando o utilitário `esptool.py`.