# Class 使用说明

## 项目简介

本项目包含一个 Main 类和多个相关的传感器数据处理模块，主要功能包括：

* 通过 Wi-Fi 连接到网络
* 通过 MQTT 协议与 EMQX 服务器通信
* 读取传感器数据，并进行解析和处理
* 将处理后的数据发布到 MQTT 服务器
* 设备校零与标定

本项目主要适用于支持 UART 通信的传感器，适用于环境监测、气体检测等应用场景。

## 功能概述

### 1. Main 类主要实现的功能：

* **Wi-Fi 连接**: 通过 wifi\_connect 方法连接到指定的 Wi-Fi 网络
* **时间同步**: 通过 set\_time 方法从 NTP 服务器同步时间
* **MQTT 连接与数据发布**: 通过 emqx\_connect 方法连接到 MQTT 服务器，并发布传感器数据
* **UART 数据处理**: 通过串口（UART）接收和发送数据，解析不同类型的传感器数据
* **传感器数据解析**: 结合 ReturnDataSubstring 类，将传感器数据解析成易于理解的格式

### 2. ReturnDataSubstring 类

* **气体类型映射**: 支持从 GAS\_TYPE\_MAPPING\_4 和 GAS\_TYPE\_MAPPING\_7 中获取对应的气体名称
* **数据提取与处理**: 通过 substring\_data\_4 和 substring\_data\_7 方法解析传感器返回的数据
* **校零与标定信息**: 处理模块的校零、标定状态

### 3. FlagCode 类

* **定义与传感器通信的指令码**，用于不同型号的传感器，包括读取数据、校零、标定等指令。

### 4. DataChangeUtil 类

* **数据转换工具**，支持十六进制字符串和字节数组的转换，十六进制到十进制转换等。

### 5. FactoryUtil 类

* **根据传感器类型 ID 选择合适的指令进行通信**，支持不同波特率的传感器。

## 安装要求

* **Python 3.x**
* **MicroPython 库**（适用于嵌入式设备）
* **EMQX 服务器**（MQTT 服务器）
* **支持 UART 通信的传感器**
* 依赖：ReturnDataSubstring, DataChangeUtil, FlagCode 类

## 使用方法

### 1. 连接 Wi-Fi

Main.wifi\_connect("Your\_SSID", "Your\_Password")

此方法会扫描 Wi-Fi 信号，尝试连接，并显示连接状态。

### 2. 同步时间

Main.set\_time()

print(Main.time\_string) # 输出当前同步时间

### 3. 连接 MQTT 服务器

Main.emqx\_connect(

MQTT\_BROKER="mqtt.example.com",

MQTT\_PORT=1883,

KEEPALIVE\_TIME=60,

MQTT\_USER="user",

MQTT\_PASSWORD="password",

CLIENT\_ID="sensor\_client",

MQTT\_TOPIC="sensor/data",

REPLY\_TOPIC="sensor/response",

ID=1,

BAUDRATE=115200,

TX=17,

RX=16,

BIT=8,

PARITY=None,

STOP=1

)

该方法会连接到 MQTT 服务器，订阅数据主题，发送心跳消息，并在接收到新数据时将其发布。

### 4. 解析传感器数据

response = uart.read()

hex\_string = ' '.join(f'{byte:02x}' for byte in response)

rm\_space = DataChangeUtil.clean\_string(hex\_string)

if hex\_string[:2] == 'AA':

data = ReturnDataSubstring.substring\_data\_4(rm\_space, sensor\_flag)

elif hex\_string[:2] == '3A':

data = ReturnDataSubstring.substring\_data\_7(rm\_space, sensor\_flag)

print(f"Parsed Data: {data}")

### 5. 发送 MQTT 消息

sensor\_data = {"data": data}

json\_data = json.dumps(sensor\_data).encode('utf-8')

client.publish(REPLY\_TOPIC, json\_data)

### 6. 处理 MQTT 订阅消息

def mqtt\_subscribe\_callback(topic, msg):

sensor\_flag = int(msg.decode('utf-8'))

code = FactoryUtil.by\_type\_get\_return(sensor\_flag, BAUDRATE\_COPY)

uart.write(code)

### 7. 处理错误与重连

try:

# MQTT 连接和数据处理代码...

except Exception as e:

print(f"Error: {e}")

if client:

client.disconnect()

client = None

reconnect()

## 主要类说明

### FlagCode 类

sensor\_info\_cmd = FlagCode.F\_SENSOR\_TYPE1 # 获取传感器信息

* F\_SENSOR\_TYPE1: 读取传感器类型
* F\_SENSOR\_NUM2: 读取浓度数据
* F\_SENSOR\_MODULE\_ZERO3: 校零
* F\_SENSOR\_MODULE\_CALIBRATION4: 标定

### ReturnDataSubstring 类

gas\_type = ReturnDataSubstring.switch\_type\_4(2) # CO

print(gas\_type) # 输出: "监测气体:CO"

### DataChangeUtil 类

hex\_str = "AA 0F 01 C5 80 EE"

byte\_array = DataChangeUtil.hex\_string\_to\_byte\_array(hex\_str)

print(byte\_array) # bytearray([170, 15, 1, 197, 128, 238])

### FactoryUtil 类

type\_id = 1

baud\_rate = 115200

command = FactoryUtil.by\_type\_get\_return(type\_id, baud\_rate)

print(command) # 对应的字节数组

## 常见问题

**如何选择正确的** type\_id**？**

* + 1：获取气体类型
  + 2：获取浓度数据
  + 3-4：查询校零和标定状态

**如何确保传感器数据正确解析？**

* + 确保从传感器读取的数据格式正确，AA 开头使用 substring\_data\_4，3A 开头使用 substring\_data\_7。

**如何处理 Wi-Fi 连接问题？**

* + 确保 ssid 和 password 正确。
  + 连接失败时，检查 Wi-Fi 是否可用。

**如何处理 MQTT 消息？**

* + 通过 mqtt\_subscribe\_callback 处理 MQTT 消息，并发送到传感器。

## 许可证

本项目使用 MIT 许可证，详情请见 LICENSE 文件。