Cấu trúc các thư mục

1. SDcard Folder

- device.txt: chứa thông tin của các channel của device

- config.txt: thông tin setting của gateway

- event.txt: các sự kiện xảy ra dưới gateway

- ghi lại thông tin các gói tin không thể gửi lên server.

2. Desktop_UI Folder

Source code của ứng dụng Desktop

- **tool.py** là file để build.

3. STM32_MODBUS_RTU_MASTER Folder

Source code chính của gateway.

---

1. Công cụ

- Dùng **STM32CubeIDE** làm môi trường để cấu hình vi điều khiển, tạo code.

(Khuyến khích: nên dùng **version 1.4.0**).

- Dùng **Visual Code** hoặc **Qt Creator** để biên dịch cho ứng dụng Desktop.

- Các cấu hình trên board **Olimex STM32-E407**

    + 2 port Modbus: USART2, USART3

    + 1 serial port: USART6

    + SDIO chế độ 4bit

- Để build Desktop app trong VS Code -> **python .\tool.py**

2. Hoạt động của source code

2.1 Các tác vụ chính

Nằm trong **void **StartDefaultTask**(**void const** * argument)**

- Các tác vụ chính nằm trong file **freertos.c.** Bao gồm các task:

    + ModbusRTUTask: Khởi tạo, cấu hình các thành phần cần thiết cho giao thức Modbus (chọn uart, baud)

    + ModbusTestTask: Triển khai Modbus RTU đảm nhận xử lý các gói tin Modbus Uplink

    + ModbusDownlinkTask: Nhận command để xử lý các gói tin Modbus Downlink

    + NetworkTimeTask: Lấy thời gian từ mạng, sau đó kích hoạt RTC

    + netmqttTask: Triển khai giao thức MQTT trên nên giao thức TCP đã khởi tạo trước đó.

+ ResetHandlerTask: Reset hệ thống khi gọi lệnh **save**

****Note**:**

1. Chú ý độ ưu tiên của mỗi task.

2. Đọc file **command.c** để hiểu rõ các lệnh để thao tác cài đặt cho gateway.

3. Hiện tại có 2 cơ chế lấy thời gian để đảm bảo lấy được thời gian từ mạng để kích RTC. Một là netmqttTask lấy thời gian trực tiếp từ internet. Hai là lấy từ server thông qua topic **time.**

2.2 Các công việc được bộ lập trình quản lý

Nằm trong **void StartDefaultTask(**void const** * argument)**

```
enum {
    SYS_START,SYS_MB_PROTOCOL,SYS_MB_APP,SYS_MB_DOWNLINK ,SYS_NET_TIME,SYS_CORE_DISCOV,SYS_MQTT,SYS_HTTP , SYS_RECORD, SYS_DEVICE, SYS_DEFAULT
};
```

2.3 Hoạt động

1. Khởi tạo ngoại vi cần thiết (UART, SDIO, RTC, FATFS,…)

2. Gọi hàm để khởi tạo các chức năng của FreeRTOS thông qua hàm MX_FREERTOS_Init();

3. Sao khi khởi tạo Queue, các task vụ -> Cho phép chạy scheduler của FreeRTOS thông qua hàm osKernelStart();

4. **StartDefaultTask**(**void const** * argument): Task chính mà scheduler sẽ quản lý (dựa vào trạng thái của 2 biến uiSysState và uiSysUpdate thì scheduler sẽ quyết định task nào sẽ được chạy.

5. Hoạt động của các luồng dữ liệu Modbus, MQTT -> Đã được trình bày rõ trong báo cáo luận văn.

3. Các công việc chưa hoàn thành

- Gateway

+ Porting Modbus TCP vào source code.

Có thể tham khảo link này: https://m.blog.naver.com/eziya76/220971629198

+ Cơ chế giám sát tại gateway: Phát triển thuật toán để có thể phát hiện thiết bị lỗi tại biên và có thể loại bỏ thiết bị ra khỏi network.

+ Các lệnh xóa device chưa hoàn thành.

- **Desktop application**

+ Tùy chỉnh giao diện để responsive với người dùng.

+ Các lệnh xóa device chưa hoàn thành.