16路串口伺服器

NuMicro® 64 位元系列微控制器範例程式碼介紹

文件資訊

應用簡述	此程式實現了串口伺服器的基本功能，包括 TCP 透傳、ModbusTCP 與 Modbus RTU/ASCII 互轉。
BSP 版本	MA35D1 Series Buildroot 2024
開發平臺	Gateway-MA35D16F V0.2

The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.

Nuvoton is providing this document only for reference purposes of NuMicro microcontroller and microprocessor based system design. Nuvoton assumes no responsibility for errors or omissions.

All data and specifications are subject to change without notice.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

1. 概述

此程式實現了 16 路串口伺服器的主要功能，實現 TCP 透傳，和 Modbus TCP 和 ModbusRTU/ASCII 協議轉換功能。

工作模式包括：

- TCP 伺服器模式
- TCP 用戶端模式
- ModbusRTU 主機模式
- ModbusRTU 從機模式
- ModbusASCII 主機模式
- ModbusASCII 從機模式

1.1 原理

利用 MA35D1 的 16 路 UART 和 2 路 Ethernet，軟體上實現市場上串口伺服器的基本功能。

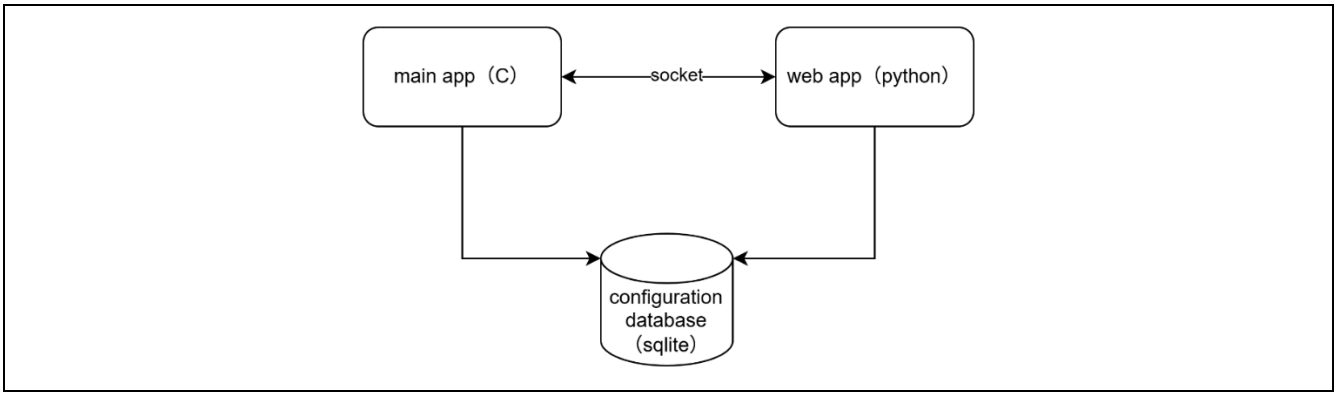

圖 1-1 軟體組成圖

軟體組成如圖 1-1：

- 主程式為 C 語言程式，實現 Modbus 協定轉換的主要功能。
- Web 應用為 Python，實現 web 伺服器功能，讓用戶可以在瀏覽器端配置串口伺服器。
- SQLite 資料庫儲存串口伺服器的配置。

1.2 編譯方法

此程式編譯用到 Python 和 SQLite 等依賴函式庫，使用 Buildroot 可以方便引用這些依賴。所以此程式也以 buildroot package 的形式編譯。本文件中若不做特殊說明都以 Buildroot 目錄為基礎目錄。

MA35D1 環境架設，生成鏡像檔可參考新唐 MA35D1 Buildroot Project User Manual。

詳細步驟：

1. 將 *serial_server* 目錄拷貝到 package/ ， 修改 *package/Config.in* 添加一行：

```
menu "Applications"
        source "package/applications/Config.in"
endmenu
#Add
source "package/serial_server/Config.in"
```

2. 在 buildroot 中 make menuconfig，增加以下配置：

    1) 選中串口伺服器範例程式

    [*] MA35 serial server demo

    2) 選中 python 和需要的 python 包

    Target packages --->Interpreter languages and scripting
        [*] python3
            ---> python3 module format to install (.py sources and .pyc compiled)
            ---> core python3 modules
                [*] sqlite module
            ---> External python modules
                [*] python-flask-login
                [*] python-netifaces
                [*] python-psutil

    3) 選中 dhcpcd

    Target packages ---> Networking applications ---> [*] dhcpcd

    4) 配置內核使用的 dts

    Kernel ---> (nuvoton/ma35d1-gateway-512m) In-tree Device Tree Source file names

5) 配置啟動裝置

Bootloaders ---> (ma35d1_sdcard0) Board defconfig

3. 將 *sserver* 目錄拷貝到 buildroot 根目錄。修改 *package/serial_server/ serial_server.mk*,確保 SERIAL_SERVER_SITE 路徑正確。

```
SERIAL_SERVER_SITE = $(TOPDIR)/sserver
```

4. 將 *ma35d1-gateway-512m.dts* 拷貝到 *output/build/linux-custom/arch/arm64/boot/dts/nuvoton*。

5. Uart 驅動修改，為匹配 Gateway-MA35D16F 開發板的 RS485 收發器，增加 gpio 控制 RTS 程式碼。修改 *output/build/linux-custom/drivers/tty/serial/ma35d1_serial.c*。

```
#include <linux/io.h>
#include <linux/gpio.h> //Add for gpio RTS
#include <linux/of_gpio.h>//Add for gpio RTS
#include <asm/irq.h>
#include <asm/serial.h>

…
struct uart_ma35d1_port {

…
        int max_count;
        int rts_gpio; //Add for gpio RTS
        /* We provide a per-port pm hook. */
        void    (*pm)(struct uart_port *port,
                        unsigned int state, unsigned int old);
};

…
static void rs485_stop_rx(struct uart_ma35d1_port *port)
{

}
//Add for gpio RTS
static void set_rs485_rst(struct uart_ma35d1_port *p,int value)
{
        int i=0;
        i= value;

        if(p->rts_gpio != -EINVAL){
                gpio_direction_output(p->rts_gpio,i);
        }

}
//Add end

…
static inline void __stop_tx(struct uart_ma35d1_port *p)
```

```
{
        unsigned int ier;

        ier = serial_in(p, UART_REG_IER);
//Modify for gpio RTS
        //if (ier & THRE_IEN)
        //        serial_out(p, UART_REG_IER, ier & ~THRE_IEN);
        if (ier & THRE_IEN){
                if(p->rts_gpio!=-EINVAL){
                        if(serial_in(p,UART_REG_FSR)&TE_FLAG){
                                set_rs485_rst(p,0);
                                serial_out(p, UART_REG_IER, ier & ~THRE_IEN);
                        }
                }else{
                        serial_out(p, UART_REG_IER, ier & ~THRE_IEN);
                }
        }
//Modify end
        if (p->rs485.flags & SER_RS485_ENABLED)
                rs485_start_rx(p);
}

…
static void ma35d1serial_start_tx(struct uart_port *port)
{
…
         serial_out(up, UART_REG_IER, ier & ~THRE_IEN);
         set_rs485_rst(up,1); //Add for gpio RTS
         if (uart_circ_chars_pending(xmit) <
                 (16-((serial_in(up, UART_REG_FSR)>>16)&0x3F)))

…
}

static int ma35d1serial_probe(struct platform_device *pdev)
{
…
       up->port.rs485_config = ma35d1serial_config_rs485;
//Add for gpio RTS
       up->rts_gpio = of_get_named_gpio(pdev->dev.of_node, "rts-gpio", 0);
        if (gpio_is_valid(up->rts_gpio)) {
                pr_err("test will request gpio\n");
                ret = devm_gpio_request(up->port.dev,
                                        up->rts_gpio, "RTS");
                if (ret) {
                        dev_err(&pdev->dev,
                                "error requesting RTS GPIO\n");
                }

                ret = gpio_direction_output(up->rts_gpio,0);
                if (ret) {
                        dev_err(&pdev->dev, "error setting RTS GPIO\n");
                }
        }else{
                pr_err("test port do not have rts_port\n");
                        up->rts_gpio=-EINVAL;
```

```
        }
//Add end
        ret = uart_add_one_port(&ma35d1serial_reg, &up->port);
        set_rs485_rst(up,0); //Add for gpio RTS
        platform_set_drvdata(pdev, up);

        return 0;
}
```

6. make linux-rebuild 重新編譯內核，再 make 生成鏡像檔。完成之後使用 nuwriter 燒錄鏡像到 Gateway-MA35D16F 開發板的 SD0。

2. 程式碼介紹

整個專案有兩個程序，C 語言主程式和 Python 實現的 web 配置程式。配置程式只是對串口伺服器進行配置並保存資料庫。 主要的資料轉發功能都在主程式中實現，這裡著重介紹主程式的實現。

2.1 架構圖

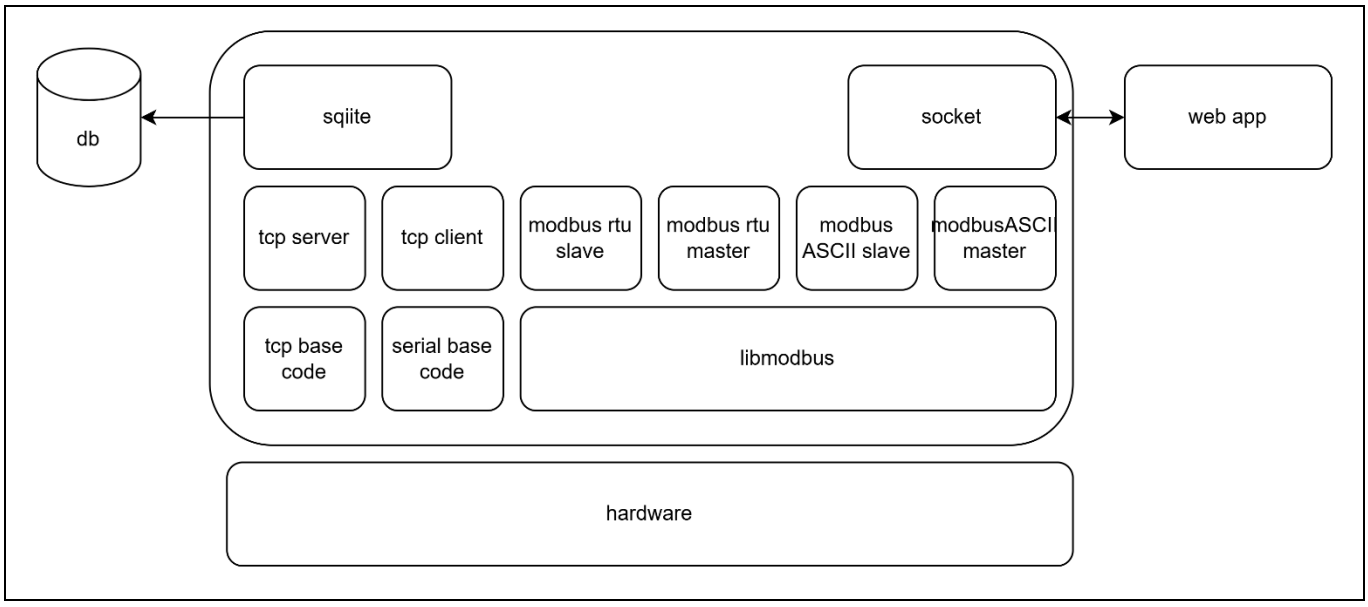

圖 2-1 主程式架構

主程式架構如圖 2-1，程式每一種模式都為一個獨立模組，Modbus xxx 模組依賴於開源的 libmodbus 函式庫。Tcp server 和 Tcp client 兩種模式只調用底層的 tcp 和 serial 相關介面。 資料庫的訪問則調用 libsqlite 相關介面。 與 web app 程序通信則使用內部 socket 實現。

2.2 主程式流程圖

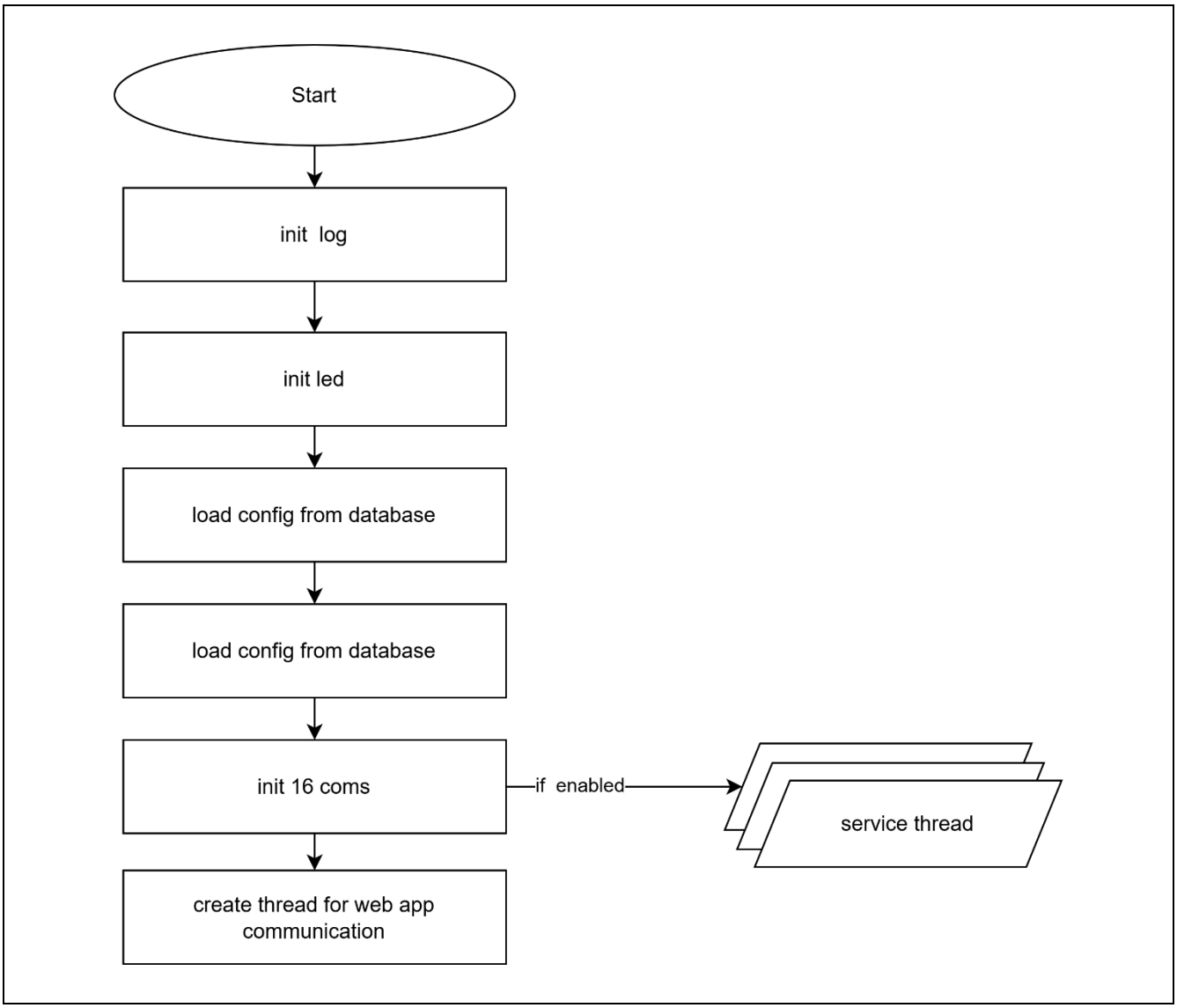

圖 2-2 主程式流程圖

主程式流程圖如圖 2-2，程式啟動之後先初始化日誌，LED 指示燈。然後從資料庫中載入 16 路串口的配置，如果串口使能則根據配置進入對應的模式，創建執行緒開始等待接收資料。16 路串口初始化完畢後則創建執行緒與 web 應用通信。

2.3 代碼介紹

在 main 函數中，會先初始化日誌介面和 LED 燈介面。然後把 g_com_service 和 g_com_conf 結構體陣列清零，這兩個陣列中保存 16 個串口對應的模式配置和通用配置。接下來 load_config() 會從 sqlite 資料庫中讀取配置並填充上述兩個結構體陣列。

```
int main(int argc,char **argv)
{
    int ret = 0;

    log_init(0, basename(argv[0]));
    led_init();
    memset( g_com_service, 0, sizeof(g_com_service));
    memset( g_com_conf, 0, sizeof(g_com_conf));

    load_config();

    com_service_init();

    init_web_command_service();

    while (1)
    {
        //LedRun();
        //usleep(500 * 1000);
        sleep_ms(5000);
    }
    return 0;
}
```

接下來進入 com_service_init()函數，會根據 g_com_service 對每一個串口初始化。 如果 com_en 等於 1 則對對應的串口進行初始化，根據 net_mode 調用對應模式的初始化函數。每種模式初始化函數裡會創建服務執行緒完成該模式的具體功能。

```
int com_service_init()
{
    int i;
    struct com_service *p_service;
    struct com_config *p_config;

    unsigned long tmp1,tmp2;

    for(i=0; i<g_sys_conf.uart_count; i++) {
        g_com_service[i].config = &g_com_conf[i];
        p_service = &g_com_service[i];
        p_config = p_service->config;

        SLOG_DEBUG("i=%d ,p_config:0x%x ,net_mode:%d
                  com_en:%d\n",i,p_config,p_config->net_mode,p_config->com_en);
        if(p_config!=NULL && p_config->com_en == 1) {
            switch(p_config->net_mode) {
            case MODE_TCP_SERVER:
                mode_tcp_server_init(p_service);
                break;
            case MODE_TCP_CLIENT:
                mode_tcp_client_init(p_service);
                break;
            case MODE_RTU_SLAVE:
                rtu_slave_init(p_service);
                break;
```

```
            case MODE_RTU_MASTER:
                rtu_master_init(p_service);
                break;
            case MODE_ASCII_MASTER:
                ascii_master_init(p_service);
                break;
            case MODE_ASCII_SLAVE:
                ascii_slave_init(p_service);
                break;
            default:
                break;
            }
            led_enable(i <= 7?i:i+8,1); //use the first row and third row to show the
status of com service.
            get_serial_count(i+1,&p_service->tx_base,&p_service->rx_base);
        }
    }

}
```

下面為關鍵結構體說明。

每一個串口對應一個 com_service 結構體，config 成員保存指向串口配置的指標，rx_base 和 tx_base 用於保存程式運行時查詢到的接收發送資料位元組數，用於統計， mode_private 為指向工作模式特有額外的資料，一般在模式初始化時填充。

```
struct com_service {
    struct com_config *config;
    unsigned long rx_base;
    unsigned long tx_base;
    void *mode_private;
};
```

com_config 結構體存儲串口配置各成員說明如下:

- inx 為串口序號,
- com_en 為串口是否開啟,
- com_mode 為工作模式，
- s_conf 儲存 uart 基礎配置如串列傳輸速率、同位等，
- pkt_size、pkt_time、pkt_mode 則是分包相關配置，
- mode_conf 保存每種工作模式特有的配置。

```
struct com_config {
    int inx;
    int com_en;
    int large_flow;//NOT USE
```

```
    int com_mode;
    struct serial_configure s_conf;
    int pkt_size;
    int pkt_time;
    int net_mode;
    void *mode_conf;
};
```

3. 軟體與硬體需求

3.1 軟體需求

- BSP 版本
  - MA35D1 Series Buildroot 2024

3.2 硬體需求

- 電路元件
  - Gateway-MA35D16F V0.2
- 接線示意圖

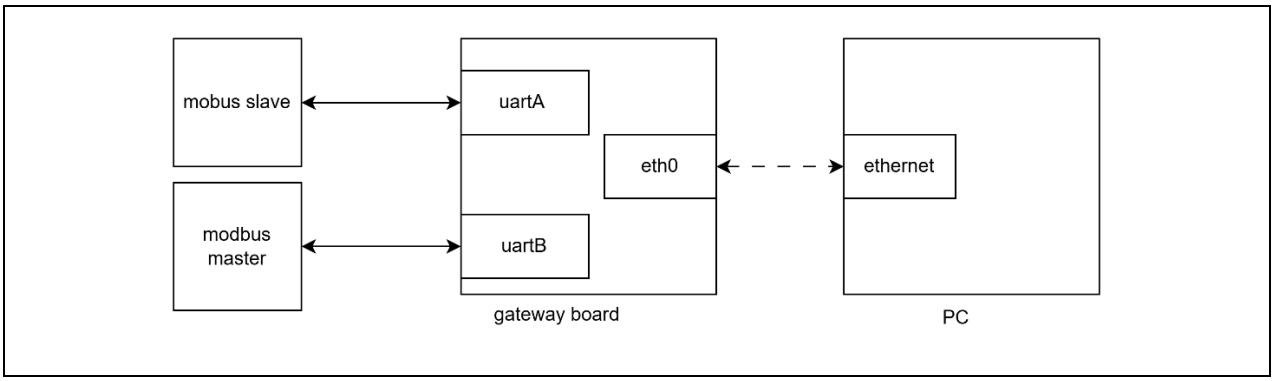

圖 3-1 線路示意圖

4. 目錄資訊

目錄結構如下圖所示。

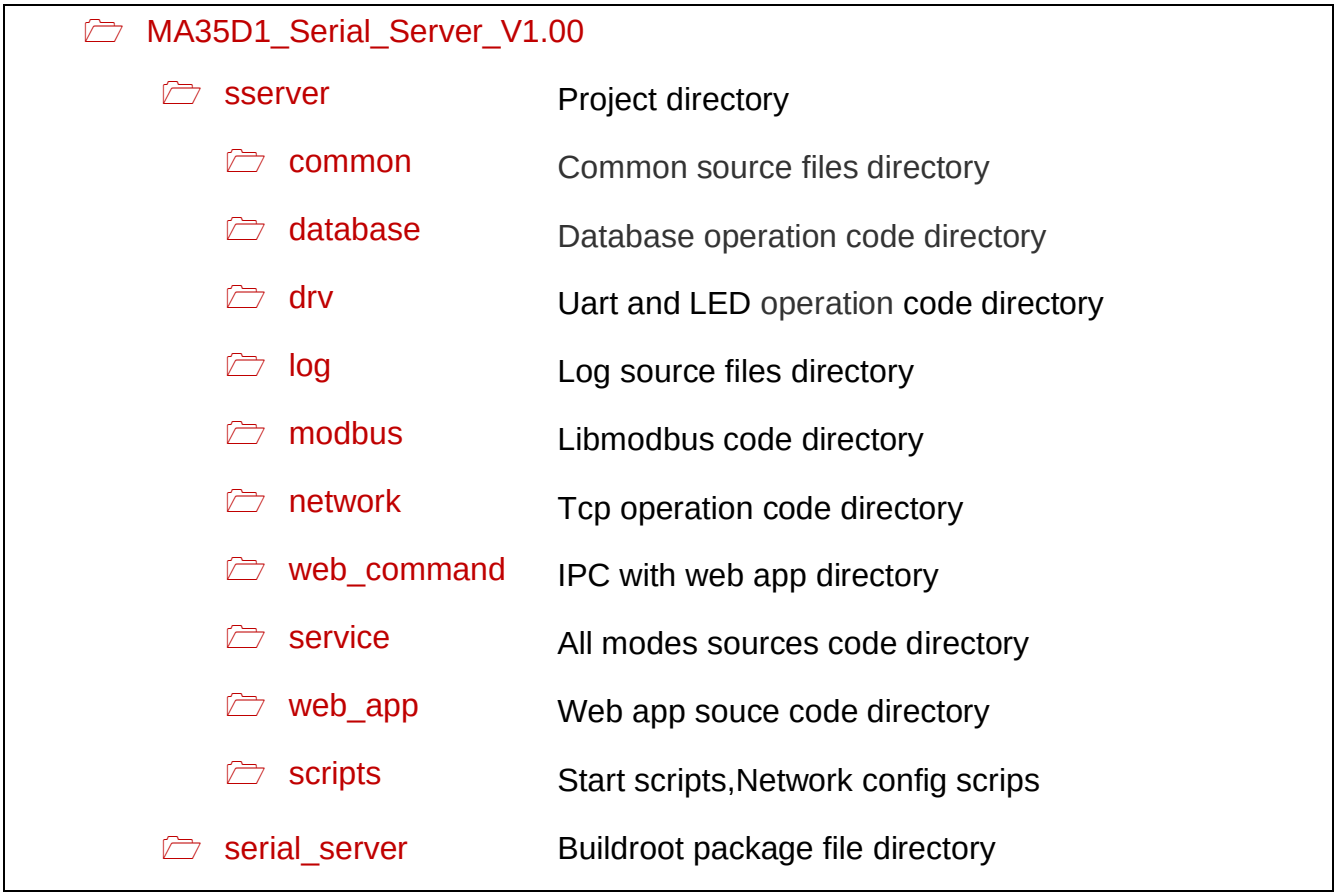

圖 4-1 目錄資訊

5. 範例程式執行

在 Buildroot 編譯該專案時，編譯腳本為 *serial_server/serial_server.mk.*

```
SERIAL_SERVER_SITE = $(TOPDIR)/sserver
SERIAL_SERVER_VERSION = 1.0.0
SERIAL_SERVER_SITE_METHOD = local
SERIAL_SERVER_INSTALL_STAGING = YES

define SERIAL_SERVER_INSTALL_TARGET_CMDS
    # install C program
    $(INSTALL) -D -m 0755 $(@D)/uart_server $(TARGET_DIR)/usr/app/
        $(INSTALL) -D -m 0755 $(@D)/scripts/S60sserver $(TARGET_DIR)/etc/init.d/
        install -m 0755 $(@D)/scripts/* $(TARGET_DIR)/usr/app/scripts/

    # install PYC
    PYTHONPATH="$(TARGET_DIR)/usr/lib/python$(PYTHON3_VERSION_MAJOR)/site-
packages:$(TARGET_DIR)/usr/lib" \
        $(HOST_DIR)/bin/python$(PYTHON3_VERSION_MAJOR) $(@D)/web_app/compile.py
        $(INSTALL) -D -m 0644 $(@D)/web_app/app.pyc $(TARGET_DIR)/usr/app/
        $(INSTALL) -D -m 0644 $(@D)/web_app/database.pyc $(TARGET_DIR)/usr/app/
        $(INSTALL) -D -m 0644 $(@D)/web_app/check_and_init_db.pyc $(TARGET_DIR)/usr/app/
        rsync -a --chmod=u=rwX,go=rX $(@D)/web_app/templates $(TARGET_DIR)/usr/app/
        $(INSTALL) -D -m 0644 $(@D)/web_app/database_schema.sql $(TARGET_DIR)/usr/app/

endef

$(eval $(cmake-package))
```

編譯腳本會把主程式、web 程式、配置腳本、html 資源安裝到對應的目錄，*sserver/scripts/S60sserver* 安裝到根目錄 *etc/init.d/*。linux 啟動之後會自動執行主程序和 web 程式。

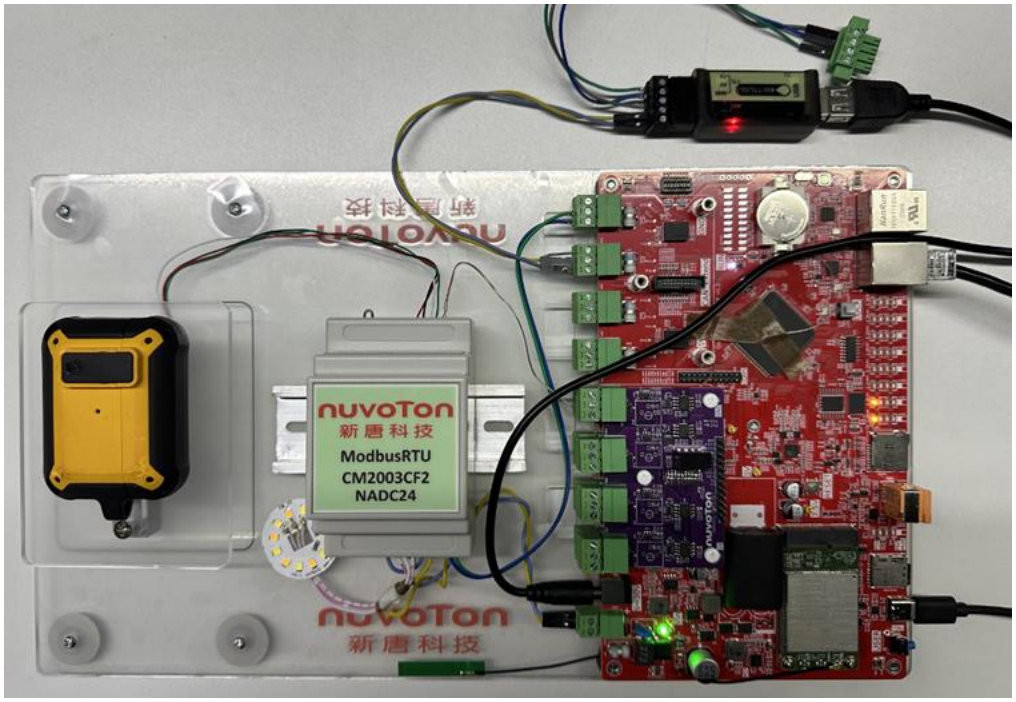

圖 5-1 接線範例

如圖 5-1 為串口伺服器接線範例，串口伺服器 com1 接 CM2003 ModbusRTU 從機，eth0 接電腦。

啟動後，eth0 連上網線後可以用瀏覽器訪問配置介面：

http://192.168.0.200:5000
用戶名：**admin**
密　碼：**admin**

網頁配置 com1 為“ModbusRTU Slave mode”，並配置 tcp 埠號（例如“10001”）。

此時即可在電腦端通過 ModbusTCP 軟體讀取從機資料，讀取從設備從 1000 開始的 10 個輸入暫存器值如圖 5-2。

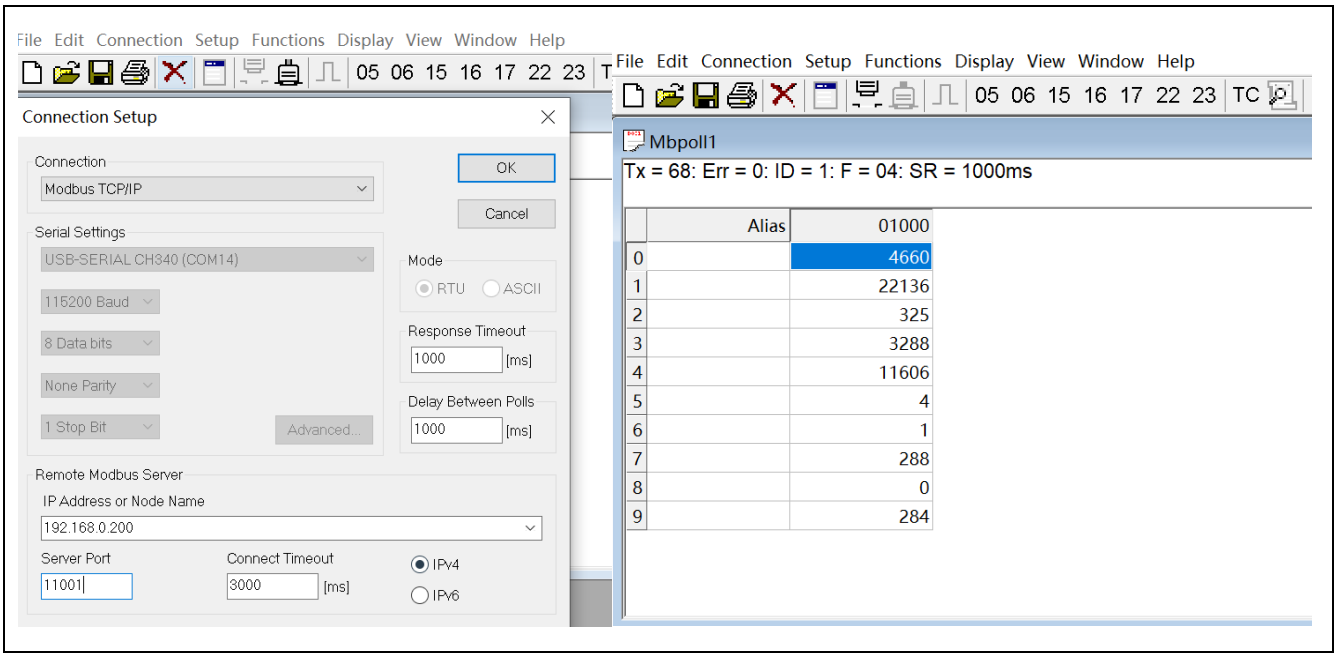
File Edit Connection Setup Functions Display View Window Help
05 06 15 16 17 22 23
Connection Setup
Connection
Modbus TCP/IP
OK
Cancel
Serial Settings
USB-SERIAL CH340 (COM14)
115200 Baud
8 Data bits
None Parity
1 Stop Bit
Advanced...
Mode
RTU
ASCII
Response Timeout
1000
[ms]
Delay Between Polls
1000
[ms]
Remote Modbus Server
IP Address or Node Name
192.168.0.200
Server Port
11001
Connect Timeout
3000
[ms]
IPv4
IPv6
File Edit Connection Setup Functions Display View Window Help
05 06 15 16 17 22 23 TC
Mbpoll1
Tx = 68: Err = 0: ID = 1: F = 04: SR = 1000ms
Alias
01000
0 4660
1 22136
2 325
3 3288
4 11606
5 4
6 1
7 288
8 0
9 284

圖 5-2 從機數據

6. 修訂紀錄

Date	Revision	Description
2025.06.16	1.00	初始發佈。

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.

Please note that all data and specifications are subject to change without notice.
All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.