# 16路串口服务器

NuMicro® 64 位系列微控制器范例代码介绍

## 文件信息

| | |
|---|---|
| 应用简述 | 此程序实现了串口服务器的基本功能，包括 TCP 透传、ModbusTCP 与 Modbus RTU/ASCII 互转。 |
| BSP 版本 | MA35D1 Series Buildroot 2024 |
| 开发平台 | 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

此程序实现了 16 路串口服务器的主要功能，实现 TCP 透传，和 Modbus TCP 和 ModbusRTU/ASCII 协议转换功能。

工作模式包括：

- TCP 服务器模式
- TCP 客户端模式
- ModbusRTU 主机模式
- ModbusRTU 从机模式
- ModbusASCII 主机模式
- ModbusASCII 从机模式

## 1.1 原理

利用 MA35D1 的 16 路 UART 和 2 路 Ethernet，软件上实现市场上串口服务器的基本功能。

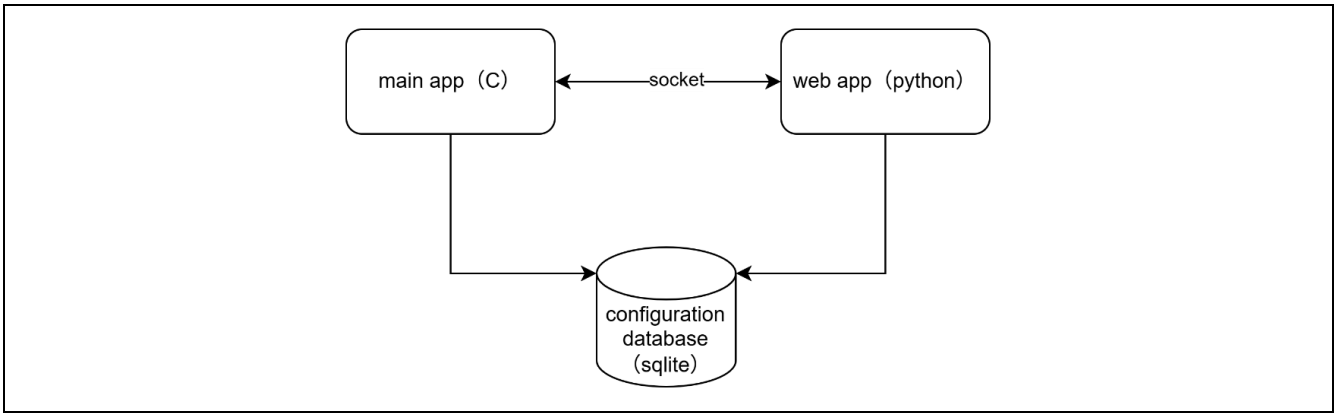


图 1-1 软件组成图

软件组成如图 1-1:

- 主程序为 C 语言程序，实现 Modbus 协议转换的主要功能。
- Web 应用为 Python，实现 web 服务器功能，让用户可以在浏览器端配置串口服务器。
- SQLite 数据库存储串口服务器的配置。

## 1.2 编译方法

此程序编译用到 Python 和 SQLite 等依赖库，使用 Buildroot 可以方便引用这些依赖。所以此程序也以 buildroot package 的形式编译。本文档中若不做特殊说明都以 Buildroot 目录为基础录。

MA35D1 环境架设，生成映像文件可参考新唐 MA35D1 Buildroot Project User Manual。

详细步骤：

1. 将 *serial_server* 目录拷贝到 *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) 选中串口服务器例程

      [*] MA35 serial server demo

   2) 选中 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) 选中 dhcpcd

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

   4) 配置内核使用的 dts

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

   5) 配置启动装置

      Bootloaders  ---> (ma35d1_sdcard0) Board defconfig

3. 将 *sserver* 目录拷贝到 buildroot 根目录。修改 *package/serial_server/ serial_server.mk,*确保 SERIAL_SERVER_SITE 路径正确。

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

4. 将 *ma35d1-gateway-512m.dts* 拷贝到 *output/build/linux-custom/arch/arm64/boot/dts/nuvoton*。

5. Uart 驱动修改，为了适配 Gateway-MA35D16F 开发板的 RS485 收发器，增加 gpio 控制 RTS 代码。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 重新编译内核，再 make 生成镜像文件。完成之后使用 nuwriter 烧录镜像到 Gateway-MA35D16F 开发板的 SD0。

# 2. 代码介绍

整个工程有两个进程，C 语言主程序和 Python 实现的 web 配置程序。配置程序只是对串口服务器进行配置并保存数据库。 主要的数据转发功能都在主程序中实现，这里着重介绍主程序的实现。

## 2.1 架构图

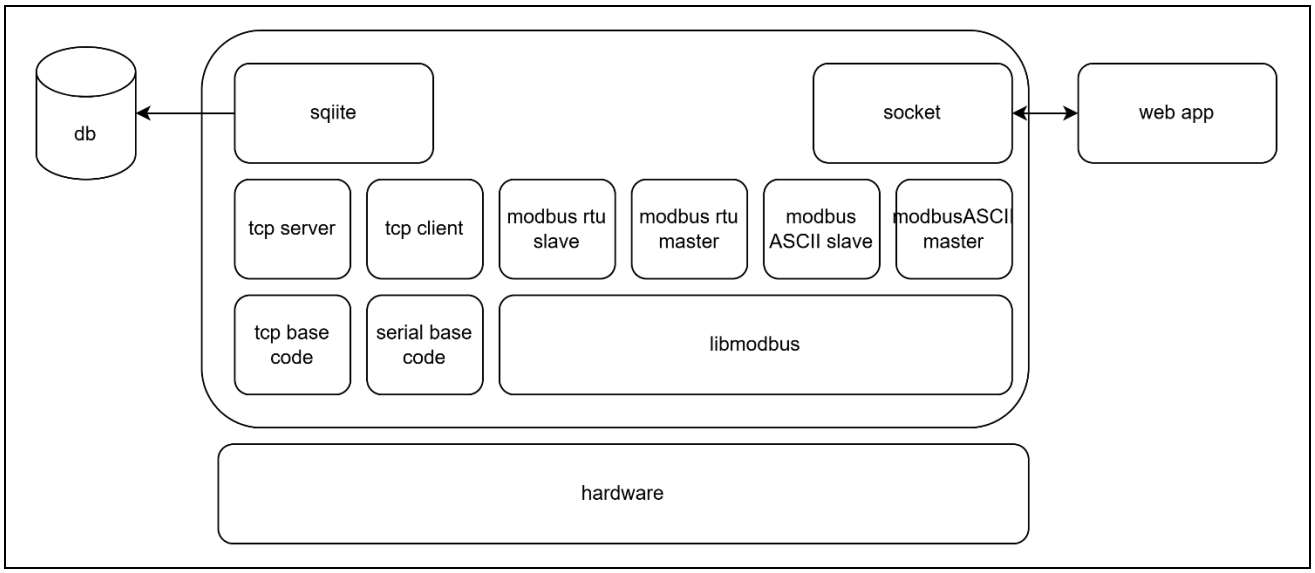


图 2-1 主程序架构

主程序架构如图 2-1，程序每一种模式都为一个独立模块，Modbus xxx 模块依赖于开源的 libmodbus 库。Tcp server 和 Tcp client 两种模式只调用底层的 tcp 和 serial 相关接口。 数据库的访问则调用 libsqlite 相关接口。 与 web app 进程通信则使用内部 socket 实现。

## 2.2 主程序流程图

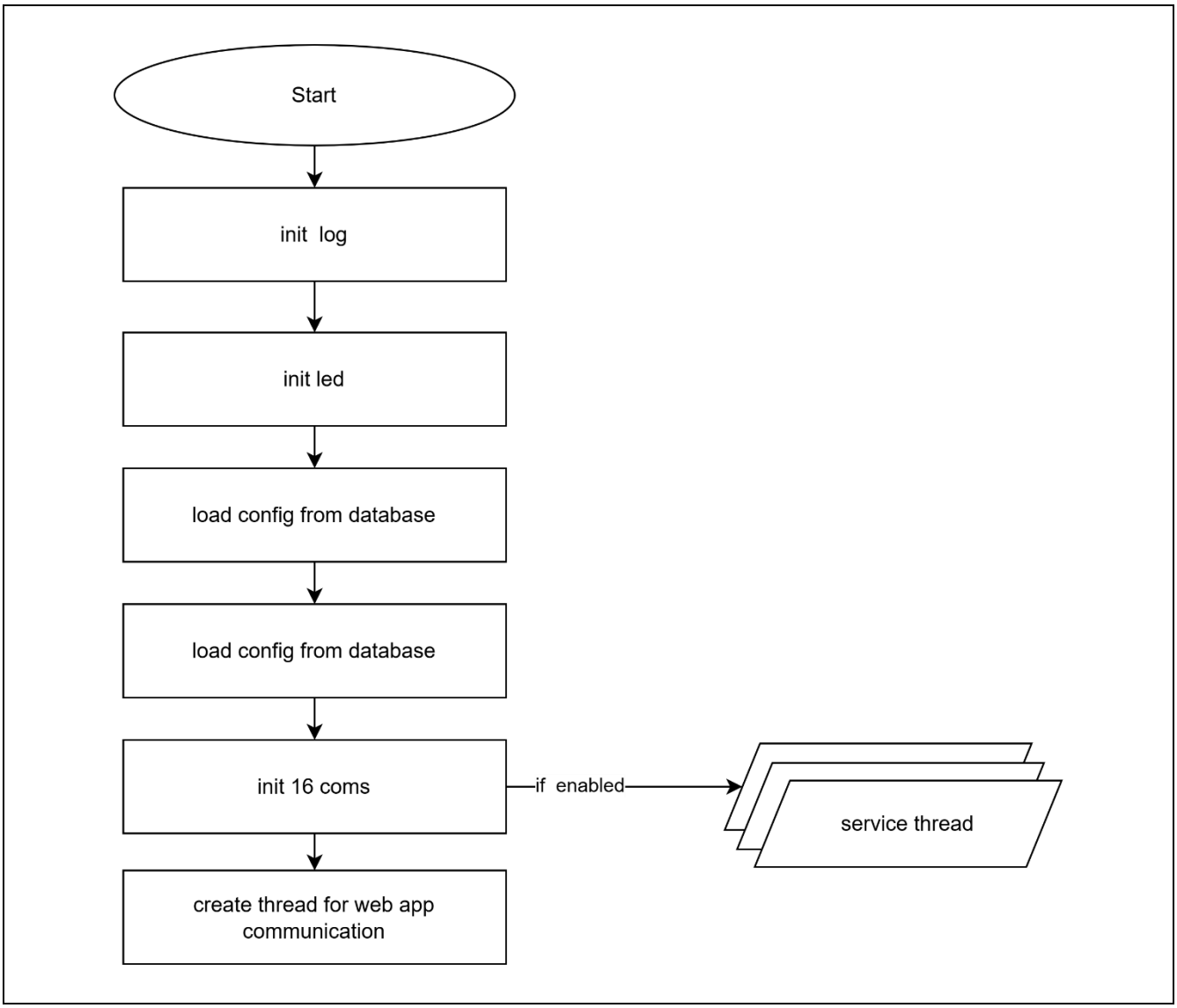


图 2-2 主程序流程图

程序流程图如图 2-2，程序启动之后先初始化日志，LED 指示灯。然后从数据库中加载 16 路串口的配置，如果串口使能则根据配置进入对应的模式，创建线程开始等待接收数据。16 路串口初始化完毕后则创建线程与 web 应用通信。

## 2.3 代码介绍

在 main 函数中，会先初始化日志接口和 led 灯接口。然后把 g_com_service 和 g_com_conf 结构体数组清零，这两个数组中保存 16 个串口对应的模式配置和通用配置。接下来 load_config() 会从 sqlite 数据库中读取配置并填充上述两个结构体数组。

```
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;
}
```

接下来进入 com_service_init()函数，会根据 g_com_service 对每一个串口初始化。 如果 com_en 等于 1 则对对应的串口进行初始化，根据 net_mode 调用对应模式的初始化函数。每种模式初始化函数里会创建服务线程完成该模式的具体功能。

```
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);
        }
    }

}
```

下面为关键结构体说明。

每一个串口对应一个 com_service 结构体，config 成员保存指向串口配置的指针，rx_base 和 tx_base 用于保存程序运行时查询到的接收发送数据字节数，用于统计， mode_private 为指向工作模式特有额外的数据，一般在模式初始化时填充。

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

com_config 结构体存储串口配置各成员说明如下：

- inx 为串口序号，
- com_en 为串口是否开启，
- com_mode 为工作模式，
- s_conf 存储 uart 基础配置如波特率、奇偶校验等，
- pkt_size、pkt_time、pkt_mode 则是分包相关配置，
- mode_conf 保存每种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. 软件与硬件需求

## 3.1 软件需求

- BSP 版本
  - MA35D1 Series Buildroot 2024

## 3.2 硬件需求

- 电路组件
  - Gateway-MA35D16F V0.2
- 接线示意图

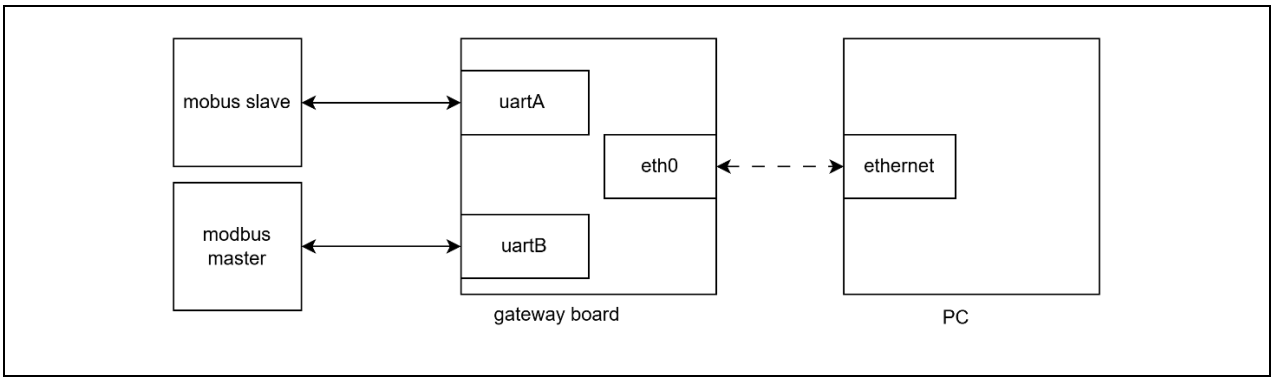


图 3-1 线路示意图

# 4. 目录信息

目录结构如下图所示。

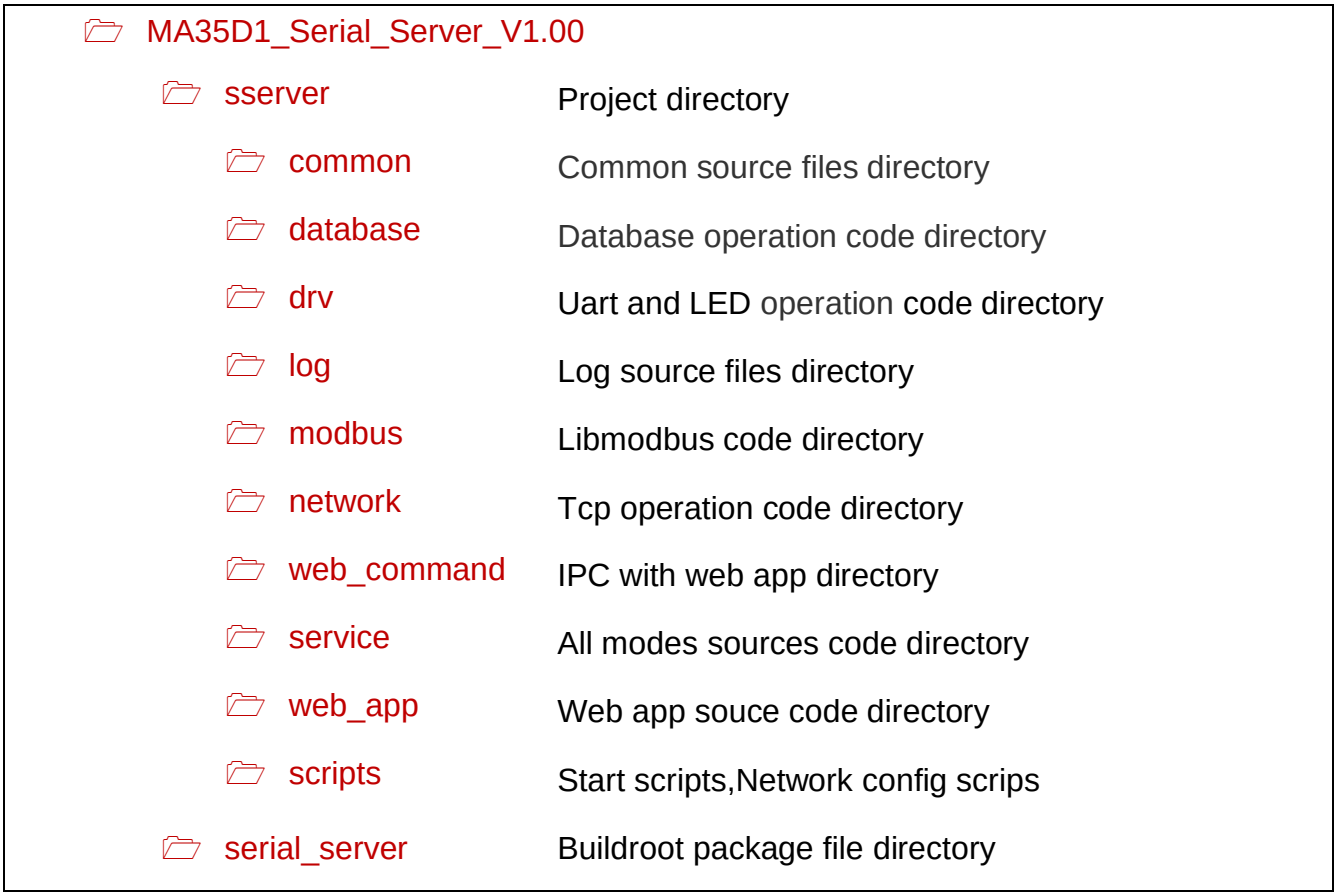


图 4-1 目录信息

# 5. 范例程序执行

在 Buildroot 编译该工程时，编译脚本为 *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))
```

编译脚本会把主程序、web 程序、配置脚本、html 资源安装到对应的目录。*sserver/scripts/S60sserver* 安装到根目录 *etc/init.d/*。linux 启动之后会自动运行主程序和 web 程序。

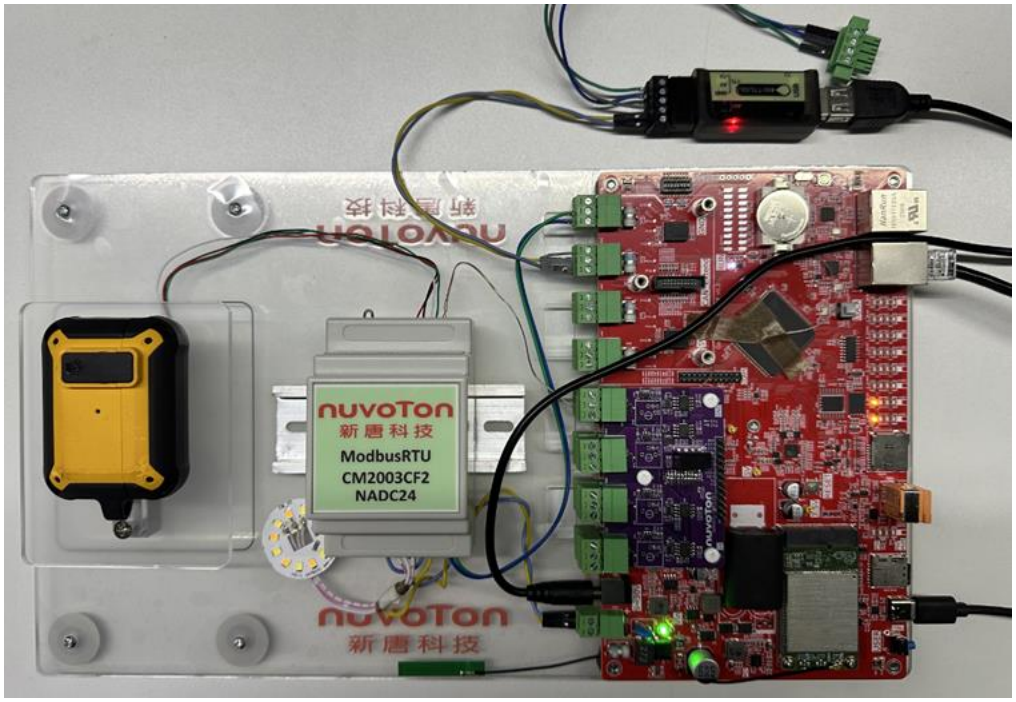


图 5-1 接线示例

如图 5-1 为串口服务器接线示例，串口服务器 com1 接 CM2003 ModbusRTU 从机，eth0 接电脑。

启动后，eth0 连上网线后可以用浏览器访问配置界面：

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

网页配置 com1 为“ModbusRTU Slave mode”，并配置 tcp 端口号（例如“10001”）。

此时即可在电脑端通过 ModbusTCP 软件读取从机数据，读取从设备从 1000 开始的 10 个输入寄存器值如图 5-2。

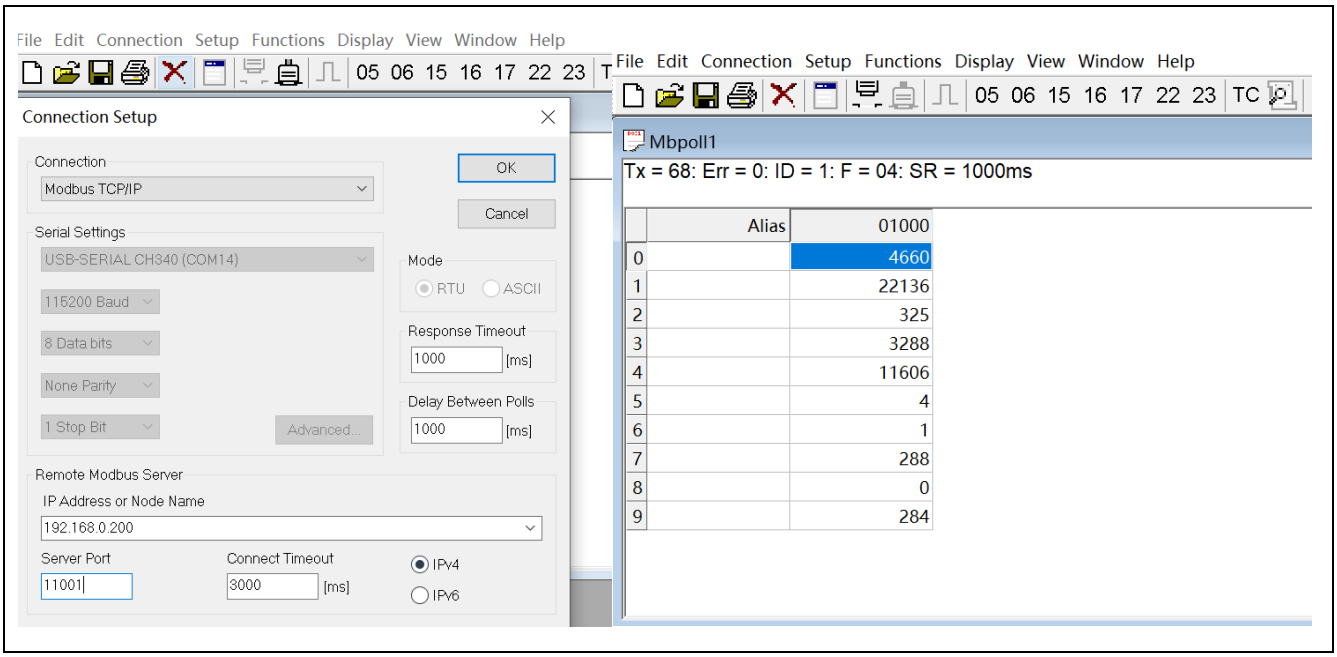
File Edit Connection Setup Functions Display View Window Help
05 06 15 16 17 22 23
Connection Setup
Connection
Modbus TCP/IP
OK
Cancel
Serial Settings
USB-SERIAL CH340 (COM14)
115200 Baud
8 Data bits
None Parity
1 Stop Bit
Advanced...
Mode
RTU
ASCII
Response Timeout
1000
[ms]
Delay Between Polls
1000
[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

图 5-2 从机数据

# 6. 修订纪录

| Date | Revision | Description |
|---|---|---|
| 2025.06.16 | 1.00 | 初始发布。 |

## 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.*