PUSR®
www.pusr.com

Open Source Controller

USR-EG828

Technical Manual

V2.0

Be Honest & Do Best

Your Trustworthy Smart Industrial IoT Partner

Content

1. Introduction ..... - 3 -
2. Hardware Interface Description ..... - 3 -
2.1. Serial port ..... - 3 -
2.2. CAN interface ..... - 6 -
2.3. Cellular network ..... - 6 -
2.4. WiFi interface ..... - 7 -
2.5. GPS ..... - 8 -
2.6. IO interface ..... - 10 -
3. Ubantu System ..... - 14 -
3.1. Downloading Linux GCC Compiler ..... - 14 -
3.2. Upgrading Ubantu version ..... - 15 -
3.3. Checking the current Linux version ..... - 15 -
4. Installing applications ..... - 15 -
4.1. Docker ..... - 15 -
5. Contact Us ..... - 16 -
6. Disclaimer ..... - 17 -

1. Introduction

This document primarily provides an explanation of the hardware and software interfaces used in the USR-EG828 product, facilitating users to quickly adapt the product after acquisition. The USR-EG828 comes with a standard Linux Ubuntu 20.04 system.

2. Hardware Interface Description

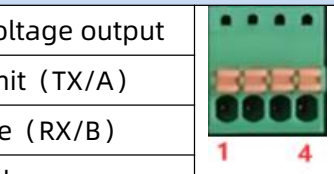
2.1. Serial port

Description of serial port driver identification:

Interface	Driver Identification
RS485-1	ttyS1
RS485-2	ttyS7
RS232-1	ttyS3
RS232-2	ttyS4

Details of the RS485/RS232 connector: the interface pin spacing is 2.0MM

NO.	Definition	Attribute	Description
1	3.3V	Output	3.3V voltage output
2	TX/A	Output	Transmit（TX/A）
3	RX/B	Input	Receive（RX/B）
4	GND	Ground	Ground



Demo function example: testing 485 bidirectional transmission

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
```

```c
#include <termios.h>
#include <stdbool.h>

int UART_INIT(char uart_num)
{
  int serial_port =0;
  if (uart_num == 0)//rs485 初始化
  {
      serial_port = open("/dev/ttyS1", O_RDWR | O_NOCTTY | O_NDELAY);
  }
  else
  {
      serial_port = open("/dev/ttyS3", O_RDWR | O_NOCTTY | O_NDELAY);
  }

  if (serial_port < 0) {
      printf("Error %i from open: %s\n", errno, strerror(errno));
      return 0;
  }

  // Create new termios struct, we call it 'tty' for convention
  struct termios tty;

  // Read in existing settings, and handle any error
  if(tcgetattr(serial_port, &tty) != 0) {
      printf("Error %i from tcgetattr: %s\n", errno, strerror(errno));
      return 0;
  }

  tty.c_cflag &= ~PARENB; // Clear parity bit, disabling parity (most common)
  tty.c_cflag &= ~CSTOPB; // Clear stop field, only one stop bit used in communication (most common)
  tty.c_cflag &= ~CSIZE; // Clear all bits that set the data size
  tty.c_cflag |= CS8; // 8 bits per byte (most common)
  tty.c_cflag &= ~CRTSCTS; // Disable RTS/CTS hardware flow control (most common)
  tty.c_cflag |= CREAD | CLOCAL; // Turn on READ & ignore ctrl lines (CLOCAL = 1)
  tty.c_lflag &= ~ICANON;
  tty.c_lflag &= ~ECHO; // Disable echo
  tty.c_lflag &= ~ECHOE; // Disable erasure
  tty.c_lflag &= ~ECHONL; // Disable new-line echo
  tty.c_lflag &= ~ISIG; // Disable interpretation of INTR, QUIT and SUSP
```

```
    tty.c_iflag &= ~(IXON | IXOFF | IXANY); // Turn off s/w flow ctrl
    tty.c_iflag &= ~(IGNBRK | BRKINT | PARMRK | ISTRIP | INLCR | IGNCR | ICRNL); //IGNBRK | PARMRK       // Disable any special handling of received bytes
    tty.c_oflag &= ~OPOST; // Prevent special interpretation of output bytes (e.g. newline chars)
    tty.c_oflag &= ~ONLCR; // Prevent conversion of newline to carriage return/line feed
    tty.c_cc[VTIME] = 10;    // Wait for up to 1s (10 deciseconds), returning as soon as any data is received.
    tty.c_cc[VMIN] = 0;

    // Set in/out baud rate to be 9600
    cfsetispeed(&tty, B9600);
    cfsetospeed(&tty, B9600);

    // Save tty settings, also checking for error
    if (tcsetattr(serial_port, TCSANOW, &tty) != 0) {
        printf("Error %i from tcsetattr: %s\n", errno, strerror(errno));
        return 0;
    }
    return serial_port;
}

int main()
{
    int port485_fd = UART_INIT(0);//uart init
    int port232_fd = UART_INIT(1);//uart init
    char write_buf[] = "Hello, Serial!";
    unsigned char read_buf[1000];

    printf("RS485 %d \n", port485_fd);
    printf("RS232 %d \n", port232_fd);
    write(port485_fd, write_buf, sizeof(write_buf));
    write(port232_fd, write_buf, sizeof(write_buf));
    bzero(read_buf,sizeof(read_buf));
    while (1)
    {

        int read_len = read(port485_fd, read_buf, sizeof(read_buf));
        // printf("read_len %d \n", read_len);
        if (read_len > 0)
        {
            printf("Read %d bytes: %s\n", read_len, read_buf);
```

```
        }
        usleep(10); // Delay for 1 second
    }
    // Close the serial port
    close(port485_fd);
    close(port232_fd);
    return 0;
}
```

2.2. CAN interface

The interface pin spacing of CAN interface is 3.81MM, interface identification is can0.

NO.	Definition	Attribute	Description
1	L	Output	CAN_L
2	H	Output	CAN_H
3	G	Ground	Reference ground

2.3. Cellular network

The EG828 has a built-in 4G cellular module for direct internet access. Network parameters can be viewed directly after powering on and logging into the desktop, or queried using Linux commands.

Command	Interface
ifconfig	Wwan or usb0
ip addr	Wwan or usb0

```
root@linux:~/test# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: can0: <NOARP,ECHO> mtu 16 qdisc noop state DOWN group default qlen 10
    link/can
3: eth0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN group default qlen 1000
    link/ether f2:31:cf:61:47:e6 brd ff:ff:ff:ff:ff:ff
4: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether ee:31:cf:61:47:e6 brd ff:ff:ff:ff:ff:ff
    inet 192.168.10.35/24 brd 192.168.10.255 scope global dynamic noprefixroute eth1
       valid_lft 61585sec preferred_lft 61585sec
    inet6 fe80::8e74:3458:ee5f:93b1/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
5: wlan0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN group default qlen 1000
    link/ether e8:51:9e:cb:1f:0b brd ff:ff:ff:ff:ff:ff
6: usb0: <BROADCAST,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether ca:2e:1c:f1:21:83 brd ff:ff:ff:ff:ff:ff
    inet 10.27.135.218/30 brd 10.27.135.219 scope global dynamic noprefixroute usb0
       valid_lft 3992sec preferred_lft 3992sec
    inet6 fe80::bbbd:4ed7:2a2c:a55c/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
root@linux:~/test#
```

```
eth0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether f2:31:cf:61:47:e6  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 34

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.10.35  netmask 255.255.255.0  broadcast 192.168.10.255
        inet6 fe80::8e74:3458:ee5f:93b1  prefixlen 64  scopeid 0x20<link>
        ether ee:31:cf:61:47:e6  txqueuelen 1000  (Ethernet)
        RX packets 131806  bytes 8723759 (8.7 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 101714  bytes 12370253 (12.3 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 46

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 674  bytes 66492 (66.4 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 674  bytes 66492 (66.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

usb0: flags=4291<UP,BROADCAST,RUNNING,NOARP,MULTICAST>  mtu 1500
        inet 10.27.135.218  netmask 255.255.255.252  broadcast 10.27.135.219
        inet6 fe80::bbbd:4ed7:2a2c:a55c  prefixlen 64  scopeid 0x20<link>
        ether ca:2e:1c:f1:21:83  txqueuelen 1000  (Ethernet)
        RX packets 514  bytes 72630 (72.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1915  bytes 302665 (302.6 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlan0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether e8:51:9e:cb:1f:0b  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@linux:~/test#
```

2.4. WiFi interface

The EG828 has a built-in WiFi module for direct WiFi connection. WiFi connection operations can be performed directly after powering on and logging into the desktop, or connected using Linux commands.

Command	Description
nmcli dev wifi list	Search for WiFi hotspots
nmcli --ask dev wifi connect <SSID> password <password>	Connect to specified WiFi hotspot
ifconfig	Query network status (WLAN0)
nmcli device disconnect wlan0	Disconnect WiFi
nmcli connection delete id <SSID>	Clear WiFi information

```
        28:9C:6E:9C:2B:BA  HN_EP05160386093PBKME-00831  Infra  1     65 Mbit/s
35      **    WPA2
        48:98:CA:63:78:E9  EZVIZ_C44642345              Infra  3     65 Mbit/s
32      **    WPA2
root@linux:~# nmcli --ask dev wifi connect USR-AP password www.usr.cn
Device 'wlan0' successfully activated with '7f72c85e-b4f3-43b6-8893-e84cddbb3d09'.
root@linux:~# ifconfig
eth0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether f2:31:cf:61:47:e6  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 34

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.10.35  netmask 255.255.255.0  broadcast 192.168.10.255
        inet6 fe80::f861:85ca:2448:2b70  prefixlen 64  scopeid 0x20<link>
        ether ee:31:cf:61:47:e6  txqueuelen 1000  (Ethernet)
        RX packets 765  bytes 68622 (68.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 871  bytes 124170 (124.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 46

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 947  bytes 84427 (84.4 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 947  bytes 84427 (84.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.11.47  netmask 255.255.254.0  broadcast 192.168.11.255
        inet6 fe80::f07f:9100:54fc:c118  prefixlen 64  scopeid 0x20<link>
        ether e8:51:9e:cb:1f:0b  txqueuelen 1000  (Ethernet)
        RX packets 108  bytes 24189 (24.1 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 38  bytes 9423 (9.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@linux:~#
```

2.5. GPS

The EG828-GL version comes with built-in GPS positioning functionality. The following commands can be used to configure and view the information:

Command	Description
apt-get install gpsd gpsd-clients	Install GPSD Client
vim /etc/default/gpsd	Modify GPS interface information
echo -ne "at+qgps=1\r\n" > /dev/ttyUSB2	Enable GPS function
cgps -s	Check positioning information

Note: The GPS function is supported by USR-EG828-GL only.

The specific steps:

1. Install GPSD Client: apt-get install gpsd gpsd-clients

2. Modify GPS interface information: enter gpsd file using vim /etc/default/gpsd

```
# Devices gpsd should collect to at boot time.
# They need to be read/writeable, either by user gpsd or the group dialout.
DEVICES="/dev/ttyUSB1"

# Other options you want to pass to gpsd
GPSD_OPTIONS=""
~
~
~
~
```

3. After making the changes, hold down Ctrl+X to save, preferably press it twice, then use the Esc key to exit insert mode, and type :wq to save the file and exit to the command interface.

```
# Devices gpsd should collect to at boot time.
# They need to be read/writeable, either by user gpsd or the group dialout.
DEVICES="/dev/ttyUSB1"

# Other options you want to pass to gpsd
GPSD_OPTIONS=""
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
:wq
```

4. Before enabling GPS, first confirm that the command channel is clear by executing the `cat /dev/ttyUSB2` command to check if the channel is open. If you receive an echo of the command, it indicates that it is functioning properly. At this point, exit to the command mode.

5. Execute the command echo `-ne "at+qgps=1\r\n" > /dev/ttyUSB2` to activate the GPS function.

6. Run `cgps -s` to enter the GPS information display interface, and wait for a while to see the GPS positioning information.

```
                                                      Seen 16/Used  2
Time:          2024-06-21T00:42:11.000Z       PRN  Elev   Azim   SNR  Use
Latitude:         36.66561207 N          GP     2  34.0   45.0  42.0   Y
Longitude:       117.09933975 E          GP    21  17.0   49.0  28.0   Y
Alt (HAE, MSL):    350.722,     363.845 ft GP   3  43.0  108.0  18.0   N
Speed:            0.00 mph               GP     6  25.0  230.0  23.0   N
Track (true, var):    0.0,  -5.4     deg GP     7   1.0  184.0  31.0   N
Climb:          1003.94 ft/min           GP     8   0.0    0.0   0.0   N
Status:         3D FIX (29 secs)         GP    14  82.0  229.0  17.0   N
Long Err  (XDOP, EPX):  n/a ,  n/a       GP    17  55.0  317.0  30.0   N
Lat Err   (YDOP, EPY):  n/a ,  n/a       GP    19  32.0  291.0  24.0   N
Alt Err   (VDOP, EPV):  0.90, +/- 67.9 ft GP   22  63.0  309.0  22.0   N
2D Err    (HDOP, CEP):  1.10, +/- 68.6 ft GP   24   1.0  322.0   0.0   N
3D Err    (PDOP, SEP):  1.50, +/- 93.5 ft GP   30  18.0  210.0   0.0   N
Time Err  (TDOP):       n/a              SB    39   0.0    0.0  34.0   N
Geo Err   (GDOP):       n/a              SB    41   0.0    0.0  34.0   N
ECEF X, VX:              n/a    n/a      SB    46   0.0    0.0  34.0   N
ECEF Y, VY:              n/a    n/a      SB    50   0.0    0.0  34.0   N
ECEF Z, VZ:              n/a    n/a
Speed Err (EPS):        n/a
Track Err (EPD):        n/a
Time offset:           0.014 sec
Grid Square:            OM86np
```

7. You can also view the raw GPS data by executing `cat /dev/ttyUSB1`, or you can obtain the data through the USB1 interface driver for processing.

```
$GPVTG,0.0,T,5.4,M,0.0,N,0.0,K,A*22
$GPRMC,004110.00,A,3639.954385,N,11705.959783,E,0.0,0.0,210624,5.4,W,A,V*59
$GPGSA,A,3,02,21,,,,,,,,,,,9.7,9.6,0.9,1*26
$GPGSV,5,1,20,02,34,045,41,03,43,108,19,06,25,230,23,07,01,184,28,1*66
$GPGSV,5,2,20,14,82,229,15,17,55,317,27,21,17,049,29,22,63,309,27,1*6C
$GPGSV,5,3,20,30,18,210,24,08,00,092,,19,32,291,,24,01,322,,1*6B
$GPGSV,5,4,20,33,,,34,38,,,34,40,,,34,42,,,34,1*6E
$GPGSV,5,5,20,46,,,34,48,,,34,49,,,34,50,,,34,1*60
$GPGGA,004111.00,3639.953607,N,11705.959742,E,1,02,9.6,105.9,M,-4.0,M,,*77
$GPVTG,0.0,T,5.4,M,0.0,N,0.0,K,A*22
$GPRMC,004111.00,A,3639.953607,N,11705.959742,E,0.0,0.0,210624,5.4,W,A,V*5D
$GPGSA,A,3,02,21,,,,,,,,,,,9.7,9.6,0.9,1*26
$GPGSV,5,1,20,02,34,045,41,03,43,108,19,06,25,230,23,07,01,184,27,1*69
$GPGSV,5,2,20,14,82,229,15,17,55,317,28,21,17,049,29,22,63,309,28,1*6C
$GPGSV,5,3,20,30,18,210,23,08,00,092,,19,32,291,,24,01,322,,1*6C
$GPGSV,5,4,20,33,,,34,38,,,34,40,,,34,42,,,34,1*6E
$GPGSV,5,5,20,46,,,34,48,,,34,49,,,34,50,,,34,1*60
$GPGGA,004112.00,3639.952966,N,11705.959730,E,1,02,9.6,105.9,M,-4.0,M,,*78
$GPVTG,0.0,T,5.4,M,0.0,N,0.0,K,A*22
$GPRMC,004112.00,A,3639.952966,N,11705.959730,E,0.0,0.0,210624,5.4,W,A,V*52
$GPGSA,A,3,02,21,,,,,,,,,,,9.7,9.6,0.9,1*26
```

2.6. IO interface

The EG828 has built-in IO interfaces, supporting DI, DO, and AI interfaces. The default configuration is 2 DO, 4 DI, and 4 AI, with each AI supporting voltage and current detection. The IO interface is controlled through the serial port, identified as ttyS8.

Parameter	Default Value(Unmodifiable)
Baud rate	921600
Parity	None
Data bits	8
Stop bits	1

The IO interface can be controlled by standard Modbus RTU protocol via serial port.

Function code	Description	Data Access
01 H	Read Coil(s)	Bit access
02 H	Read Single or Multiple Holding Register(s)	Bit access
03 H	Read Single or Multiple Holding Register(s)	32-bit access
04 H	Read input status	32-bit access
05 H	Write Single Coil	Bit access
06 H	Write single holding register	32-bit access
0F H	Write multiple coils	Bit access
10 H	Write Single or multiple holding registers	32-bit access

The Modbus slave address is 0x01, and the register address is as follows:

IO Type	Register type	Register address (Hex)	Function code	Bit(s)	Description

			(Hex)		
DI	1x	0000 ~ 0003	02	Bit	
DO	0x	0000 ~ 0001	01/05/0F	Bit	ON：0xFF00 OFF：0x0000
AI	3x	0000 ~ 0006	04	32 Bit Unsigned （AB CD）	
AI	3x	0010 ~ 0016	04	32 Bit Unsigned （AB CD）	

The test demo code is as the follows, mainly implementing the 1-second toggling function of the DO (Digital Output) interface.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#include <termios.h>
#include <stdbool.h>

#define SAVLE_ADDR 0X01
#define DO1_ADDR 0X0000
#define DO2_ADDR 0X0001
#define DO3_ADDR 0X0002
#define DO4_ADDR 0X0003
#define DO_ON   0XFF00
#define DO_OFF  0X0000

unsigned char IO_ID = 0;
unsigned char IO_sta = 0;

unsigned short crc16(const unsigned char* data, unsigned short length) // CRC16
{
    unsigned short crc = 0xFFFF;
    for (int pos = 0; pos < length; pos++)
    {
        crc ^= (unsigned short)data[pos]; // XOR byte into least sig. byte of crc
        for (int i = 8; i != 0; i--)
        { // Loop over each bit
```

```
            if ((crc & 0x0001) != 0) // If the LSB is set
            {
                crc >>= 1; // Shift right and XOR 0xA001
                crc ^= 0xA001;
            } else          // Else LSB is not set
                crc >>= 1;    // Just shift right
        }
    }
    return crc;   // Note, this CRC calculation is reversed endian to some implementations
}

// IO uart init --ttyS8
int UART_INIT(void)
{
    int serial_port =0;
    serial_port = open("/dev/ttyS8", O_RDWR | O_NOCTTY | O_NDELAY);
    if (serial_port < 0)
    {
        printf("Error %i from open: %s\n", errno, strerror(errno));
        return 0;
    }

    struct termios tty;   // Create new termios struct, we call it 'tty' for convention
    if(tcgetattr(serial_port, &tty) != 0)   // Read in existing settings, and handle any error
    {
        printf("Error %i from tcgetattr: %s\n", errno, strerror(errno));
        return 0;
    }

    tty.c_cflag &= ~PARENB; // Clear parity bit, disabling parity (most common)
    tty.c_cflag &= ~CSTOPB; // Clear stop field, only one stop bit used in communication (most common)
    tty.c_cflag &= ~CSIZE; // Clear all bits that set the data size
    tty.c_cflag |= CS8; // 8 bits per byte (most common)
    tty.c_cflag &= ~CRTSCTS; // Disable RTS/CTS hardware flow control (most common)
    tty.c_cflag |= CREAD | CLOCAL; // Turn on READ & ignore ctrl lines (CLOCAL = 1)
    tty.c_lflag &= ~ICANON;
    tty.c_lflag &= ~ECHO; // Disable echo
    tty.c_lflag &= ~ECHOE; // Disable erasure
    tty.c_lflag &= ~ECHONL; // Disable new-line echo
    tty.c_lflag &= ~ISIG; // Disable interpretation of INTR, QUIT and SUSP
```

```
    tty.c_iflag &= ~(IXON | IXOFF | IXANY); // Turn off s/w flow ctrl
    tty.c_iflag &= ~(IGNBRK | BRKINT | PARMRK | ISTRIP | INLCR | IGNCR | ICRNL); // Disable any special handling of
received bytes
    tty.c_oflag &= ~OPOST; // Prevent special interpretation of output bytes (e.g. newline chars)
    tty.c_oflag &= ~ONLCR; // Prevent conversion of newline to carriage return/line feed
    tty.c_cc[VTIME] = 10;    // Wait for up to 1s (10 deciseconds), returning as soon as any data is received.
    tty.c_cc[VMIN] = 0;

    // Set in/out baud rate to be 9600
    cfsetispeed(&tty, B921600);
    cfsetospeed(&tty, B921600);

    // Save tty settings, also checking for error
    if (tcsetattr(serial_port, TCSANOW, &tty) != 0)
    {
        printf("Error %i from tcsetattr: %s\n", errno, strerror(errno));
        return 0;
    }
    return serial_port;
}

int main()
{
    int serial_ttyS8 = UART_INIT();//uart init
    int res = 0;
    unsigned char IO_sta = 0;

    while (1)
    {
        unsigned char msg[8] = {0x00};
        msg[0] = SAVLE_ADDR;
        msg[1] = 0x0f;//cmd id
        msg[2] = (DO1_ADDR >> 8) & 0xff;
        msg[3] = DO1_ADDR & 0xff;
        msg[4] = 0x00;
        msg[5] = 0x02;
        IO_sta = ~IO_sta;
        printf("IO_sta is %d\n", IO_sta);

        if(IO_sta == 0)
```

```
        {
            //DO OPEN
            msg[6] = 0x01;
            msg[7] = 0x00;
        }
        else
        {
            //DO CLOSE
            msg[6] = 0x01;
            msg[7] = 0x03;
        }
        // crc16
        unsigned short crc = crc16(msg, sizeof(msg));
        unsigned char fullMessage[10];
        for (int i = 0; i < sizeof(msg); ++i)
        {
            fullMessage[i] = msg[i];
        }
        fullMessage[8] = crc & 0xFF; // CRC high
        fullMessage[9] = (crc >> 8) & 0xFF; // CRC  low

        write(serial_ttyS8, fullMessage, sizeof(fullMessage));
        sleep(1); // Delay for 1 second
    }
    // Close the serial port
    close(serial_ttyS8);
    return 0;
}
```

3. Ubantu System

3.1. Downloading Linux GCC Compiler

Command	Function
apt-get update	Update Linux software sources
apt-get install gcc	Download and install GCC compiler
gcc hello.c -o hello	Compile the hello.c file

For cross-compilation, download from specific websites.

3.2. Upgrading Ubantu version

The product comes with Ubuntu 20.04. To upgrade to the latest version of Ubuntu, use the following commands:

Command	Function
apt-get update	Update sources
apt-get upgrade	Update installed packages
apt-get dist-upgrade	Handle dependency relationships
apt-get install update-manager-core	Install upgrade tool
do-release-upgrade	Upgrade
lsb_release -a	Version query

3.3. Checking the current Linux version

Command	Function
lsb_release -a	Display Ubuntu's distribution ID, description, version number, codename, etc.
uname -a	Display all system information, including kernel version and system architecture
hostnamectl	Display the static, dynamic, and transparent hostname settings of the system

4. Installing applications

4.1. Docker

The EG828-GL can install Docker containers with the following commands:

Command	Function
apt-get update	Update sources
apt-get upgrade	
apt install apt-transport-https ca-certificates curl software-properties-common	Install necessary packages to allow apt to use HTTPS repositories
curl -fsSL https://mirrors.aliyun.com/docker-ce/linux/ubuntu/gpg | apt-key add -	Add Docker's domestic repository to your system, using Aliyun as an example
apt-key list	List all GPG keys added to the system
add-apt-	Add the Docker repository to

repository "deb [arch=arm64] https://mirrors.aliyun.com/docker-ce/linux/ubuntu $(lsb_release -cs) stable"	APT sources
apt-get update	Update apt package index again
apt-get install docker-ce docker-ce-cli containerd.io	Install the latest version of Docker CE (Community Edition)

mkdir -p /etc/docker
tee /etc/docker/daemon.json <<-'EOF'
{
"registry-mirrors": ["https://dnbf7xuh.mirror.aliyuncs.com"]
}
EOF
systemctl daemon-reload
sudo systemctl restart docker
	Configure Docker to use Aliyun mirror accelerator

docker -- version	Verify Docker installation and operation
systemctl stop docker.socket systemctl disable docker.socket systemctl stop docker systemctl disable docker	Stop/disable Docker
systemctl status docker	Check Docker status

5. Contact Us

Jinan USR IOT Technology Limited

Address : Floor 12 and 13, CEIBS Alumni Industrial Building, No. 3 Road of Maolingshan, Lixia District, Jinan, Shandong, China

Official website: https://www.pusr.com

Official shop: https://shop.usriot.com

Technical support: http://h.usriot.com/

Email : sales@usriot.com

Tel : +86-531-88826739

Fax : +86-531-88826739-808

6. Disclaimer

The information in this document provided in connection with Jinan USR IoT technology ltd. and/or its affiliates' products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of USR IoT products. EXCEPT AS SET FORTH IN THE TERMS AND CONDITIONS AS SPECIFIED IN THE LICENSE AGREEMENT FOR THIS PRODUCT, USR IoT AND/OR ITS AFFILIATES ASSUME NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL USR IoT AND/OR ITS AFFILIATES BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF USR IoT AND/OR ITS AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. USR IoT and/or its affiliates make no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. USR IoT and/or its affiliates do not make any commitment to update the information contained in this document.

Your Trustworthy Smart IOT Partner

PUSR®
www.pusr.com

Official Website: www.pusr.com

Official Shop: shop.usriot.com

Technical Support: h.usriot.com

Inquiry Email: inquiry@usriot.com

Skype & WhatsApp: +86 13405313834

关注有人微信公众号　登录商城

Click to view more: Product Catalog & Facebook & Youtube