ED-GWL1010

一款高性价比 LORAWAN 室内网关

上海晶珩电子科技有限公司

2023-03-23

版权声明

ED-GWL1010 及其相关知识产权为上海晶珩电子科技有限公司所有。
上海晶珩电子科技有限公司拥有本文件的版权并保留所有权利。未经上海晶珩电子科技有限公司的书面许可，不得以任何方式和形式修改、分发或复制本文件的任何部分。

免责声明

上海晶珩电子科技有限公司不保证本手册中的信息是最新的、正确的、完整的或高质量的。上海晶珩电子科技有限公司也不对这些信息的进一步使用作出保证。如果由于使用或不使用本手册中的信息，或由于使用错误或不完整的信息而造成的物质或非物质相关损失，只要没有证明是上海晶珩电子科技有限公司的故意或过失，就可以免除对上海晶珩电子科技有限公司的责任索赔。上海晶珩电子科技有限公司明确保留对本手册的内容或部分内容进行修改或补充的权利，无需特别通知。

目　录

1 产品概述

ED-GWL1010 是 EDATEC 推出的一款高性价比 LoRaWAN 室内网关产品。ED-GWL1010 基于 EDATEC 全新睿莓 1(ED-REIMEI1)单板电脑平台，采用主板 + 扩展板的形式，集成 Semtech 新一代 SX1302/SX103 基带芯片和 Microchip 安全加密芯片 ATECC608，支持 DC Jack 供电和 PoE 供电，可选钣金外壳。

1.1 目标应用

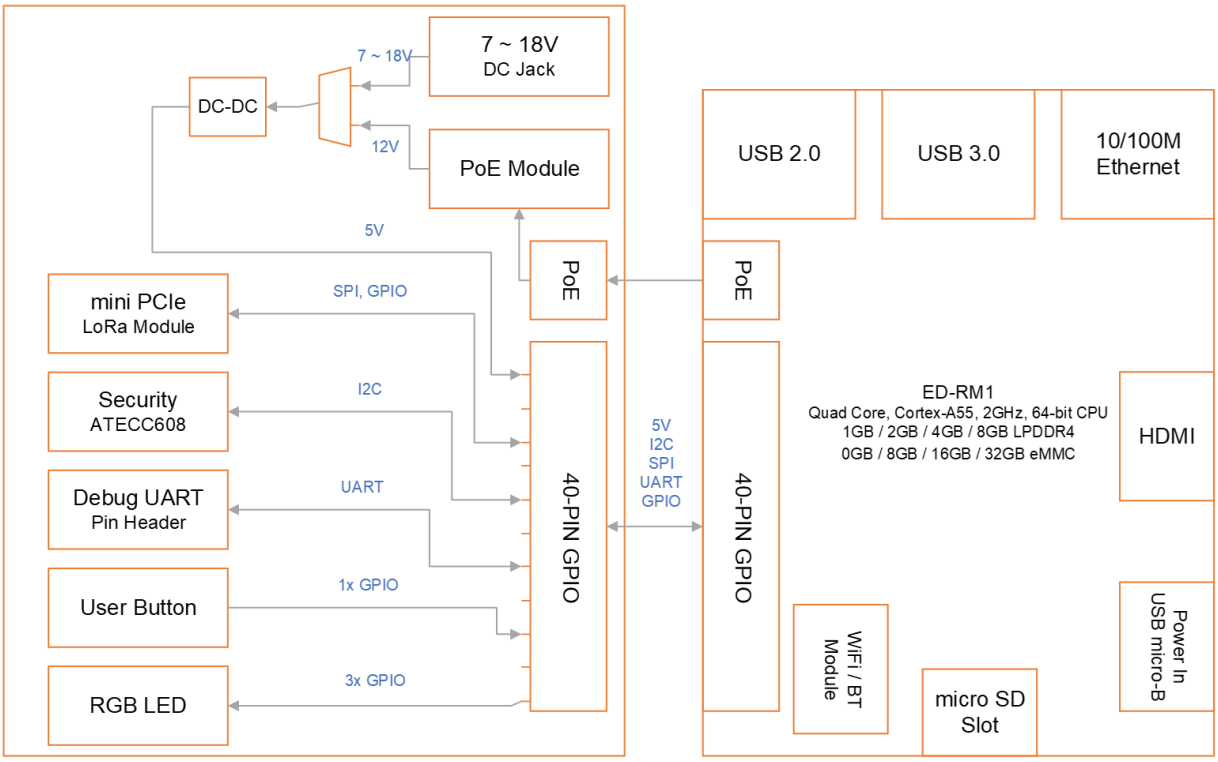
- LoRa 智能网关
- 工业控制
- 智能制造
- 智慧城市
- 智慧交通

1.2 规格参数

功能	参数
CPU	AMLogic S905X4 4 核, ARM Cortex-A55(ARM v8), 2GHz, 64 位 CPU
内存	可选 1GB / 2GB / 4GB / 8GB LPDDR-3200 SDARM
eMMC 闪存	可选 0GB / 8GB / 16GB / 32GB
SD 卡	可同时与 eMMC 使用，可从 SD 卡启动
以太网	1x 10/100M 以太网，支持 PoE
WiFi / 蓝牙	2.4G / 5.8G 双频 WiFi，蓝牙 5.0
LoRa	基于 Semtech SX1302+SX1250 的 LoRa 网关模块，通过 CE / FCC 认证，可选欧版或美版
LoRa Frequency	美版: US915, AU915, AS923
	欧版: EU868
USB Host	1x USB 3.0 Type A, 1x USB 2.0 Type A
mini PCIe	1x mini PCIe Slot，支持 SPI 总线，用于扩展 LoRa 网关模块
LED 指示灯	1x RGB LED
按键	1x User Button
电源输入	7V ~ 18V
尺寸	110 (L) x 90 (W) x 26 mm (H)

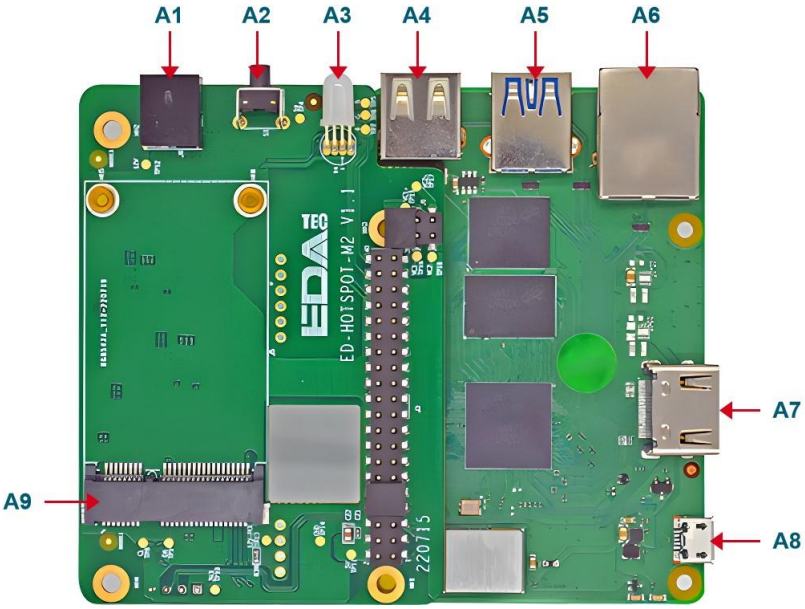
功能	参数
外壳	桌面型，钣金外壳
天线配件	1x WiFi / BT 外置天线, 1x LoRa 外置天线
工作环境温度	-25 ~ 50°C
操作系统	Debian 11, Lite, 64-bit OS
软件资源	提供 ChipStack 等 LoRaWAN 网络的示例指引

1.3 系统框图

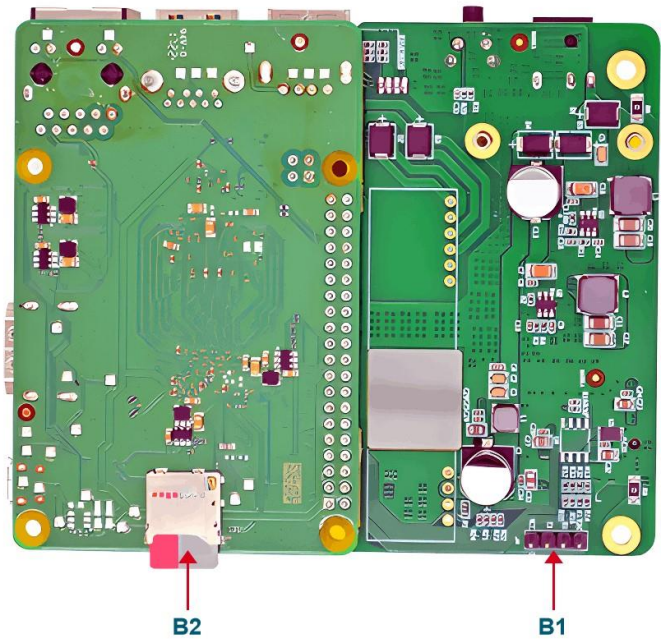


ED-GWL1010 Block Diagram

1.4 功能布局



编号	功能描述	编号	功能描述
A1	12V DC 电源插座	A6	以太网 RJ45 接口
A2	按键	A7	HDMI type A 接口
A3	RGB LED	A8	Micro USB 供电接口
A4	USB 2.0 接口	A9	LoRa mini-PCIe 接口
A5	USB 3.0 接口		

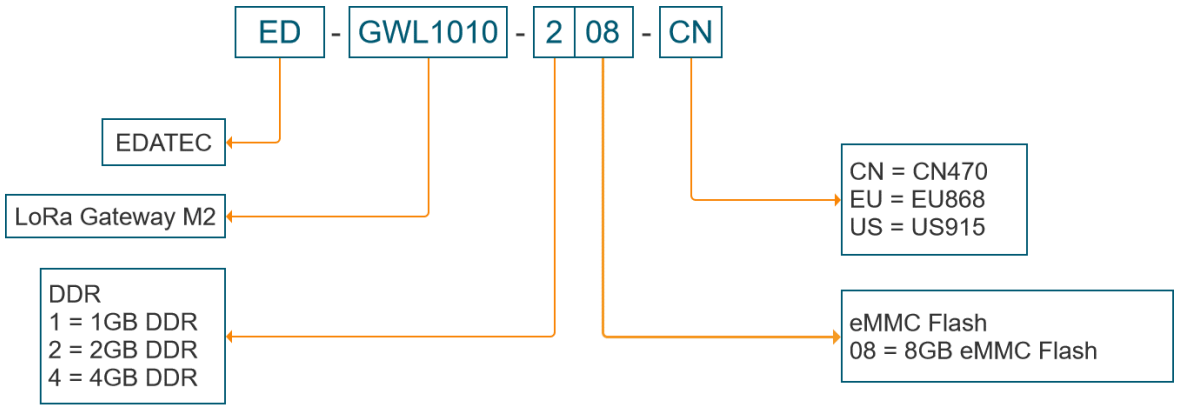


编号	功能描述
B1	调试串口
B2	Micro SD 卡槽

1.5 包装清单

- 1x ED-GWL1010 主机
- [选配]1x LoRa 天线
- [选配]1x 2.4GHz/5GHz WiFi/BT 天线

1.6 订购编码



Example

Part# : ED-GWL1010-208-CN
Configration : GWL1010 LoRa Gateway
1pcs REIMEI1 Computer
2GB DDR and 8GB eMMC Flash
CN470 LoRa Module

2 快速启动

2.1 设备清单

- 1x ED-GWL1010 主机
- 1x WiFi / BT 外置天线
- 1x LoRa 外置天线
- 1x 网线
- 1x 12V@2A 电源适配器

2.2 硬件连接

1.安装 WiFi 外部天线。
2.安装 LoRa 外部天线。
3.插入网线到以太网网口，网线连接可上网的路由器、交换机等网络设备。
4.插入 ED-GWL1010 的 DC 电源输入口（+12V DC）,并给电源适配器供电。

2.3 首次启动

ED-GWL1010 没有电源开关，插入电源线，系统将会开始启动。

2.3.1 使能 SSH

开机自动使能 SSH：
在设备启动时，开机前向 boot 分区放入一个名为 ssh 的空文件，开机后会自动使能 SSH。

命令使能 SSH：

```
sudo raspi-config
```

输入上方命令后，会出现一个命令行界面，第三个接口配置，找到 SSH，选择 yes 使能 SSH 功能。
3 Interface Options -> 2 SSH -> Yes

2.3.2 SSH 工具

Windows 端推荐使用 putty 来实现 SSH 远程连接。

- Putty 下载：Download PuTTY - a free SSH and telnet client for Windows

2.3.3 查找设备 IP

- 设备开启如果接有显示屏可以使用 ifconfig 命令查看当前设备 IP
- 如果没有显示屏，则可以通过路由器查看分配的 IP
- 如果没有显示屏，则可以下载 nmap 工具扫描当前网络下的 IP
  nmap 支持 Linux、macOS、Windows 等多个平台。如果希望使用 nmap 扫描 192.168.3.0~255 的网段，则可以使用以下命令：

  ```
  nmap -sn 192.168.3.0/24
  ```

  等待一段时间后即会输出结果，类似与下方输出：

  ```
  Starting Nmap 7.92 ( https://nmap.org ) at 2022-12-30 21:19 中国标准时间
  Nmap scan report for 192.168.3.1 (192.168.3.1)
  Host is up (0.0010s latency).
  MAC Address: XX:XX:XX:XX:XX:XX (Phicomm (Shanghai))
  Nmap scan report for DESKTOP-FGEOUUK.lan (192.168.3.33)
  Host is up (0.0029s latency).
  MAC Address: XX:XX:XX:XX:XX:XX (Dell)
  ```

```
Nmap scan report for 192.168.3.66 (192.168.3.66)
Host is up.
Nmap done: 256 IP addresses (3 hosts up) scanned in 11.36 seconds
```

2.3.4 SSH 远程登录

```
ssh phantom@<IP>
```

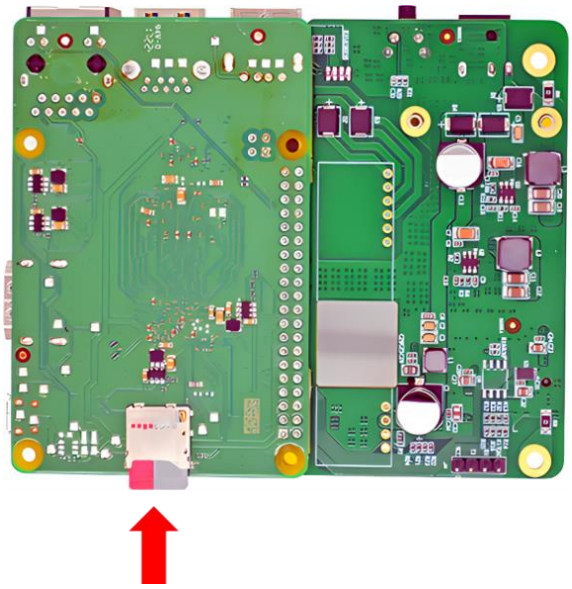
用户名：phantom
密码　：phantom
端口号：22

3 接线指南

3.1 Internal I/O

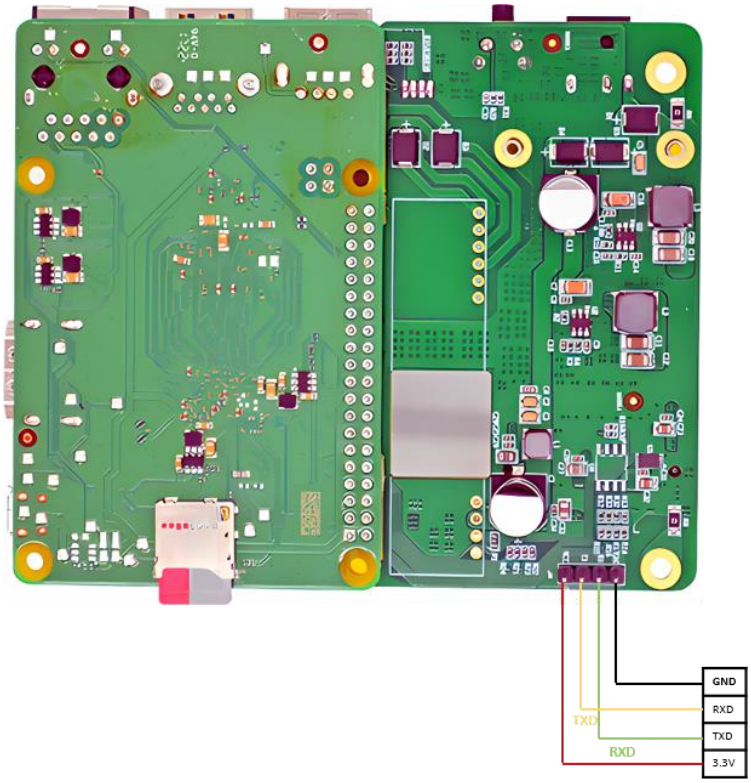
3.1.1 micro-SD Card

micro SD 卡槽位于主板背面，请将 micro SD 卡正面朝上，按平行方向插入 micro SD 卡槽。



3.1.2 TTL 串口

TTL 串口位于 LoRa 接口板的背面，请按下图所示连接串口。



4 软件操作指引

4.1 按键

ED-GWL1010 具有一个用户按键，用户可以自定义按键功能，按键与 PIN16 相连，默认状态下为高电平，当按键按下时，该管脚为低电平。

我们使用 libgpiod 库和对应命令行工具进行 GPIO 的设置和读取。

4.1.1 安装 libgpiod

安装 ligpiod

```
sudo apt-get update
#安装 libgpiod 的静态库及头文件
sudo apt-get install libgpiod-dev
#安装基于 libgpiod 的命令行工具
sudo apt-get install gpiod
```

libgpiod 支持 6 个命令行测试命令，分别是：

- gpiodetect - 列出系统上存在的所有 gpiochips，它们的名称，标和 GPIO 线数
- gpioinfo - 列出指定 gpiochips 的所有行、它们的名称、消费者、方向、活动状态和附加标志
- gpioget - 读取指定 GPIO 线的值

- gpioset - 设置指定 GPIO 线的值，可能保留这些线导出并等待超时、用户输入或信号
- gpiofind - 找到 gpiochip 名称和给定行名称的行偏移
- gpiomon - 等待 GPIO 线上的事件，指定要观看的事件，退出前要处理多少事件或事件应该报告给控制台

使用 gpioinfo 命令查看 GPIO 信息

```
phantom@phantom:~ $ gpioinfo
gpiochip0 - 87 lines:
        line   0:       "PIN27"         kernel   input  active-high [used]
        line   1:       "PIN28"         kernel   input  active-high [used]
        line   2:  "EMMC_DAT0"          kernel   input  active-high [used]
        line   3:  "EMMC_DAT1"          kernel   input  active-high [used]
        line   4:  "EMMC_DAT2"          kernel   input  active-high [used]
        line   5:  "EMMC_DAT3"          kernel   input  active-high [used]
        line   6:  "EMMC_DAT4"          kernel   input  active-high [used]
        line   7:  "EMMC_DAT5"          kernel   input  active-high [used]
        line   8:  "EMMC_DAT6"          kernel   input  active-high [used]
        line   9:  "EMMC_DAT7"          kernel   input  active-high [used]
        line  10:   "EMMC_CLK"          kernel   input  active-high [used]
        line  11:   "NAND_ALE"          unused   input  active-high
        line  12:   "EMMC_CMD"          kernel   input  active-high [used]
        line  13:           "-"         unused   input  active-high
        line  14:   "EMMC_RST"          unused   input  active-high
        line  15: "EMMC_NAND_DQS" kernel input active-high [used]
        line  16:           "-"         unused   input  active-high
        line  17:           "-"         unused   input  active-high
        line  18:    "SD_DAT0"          kernel   input  active-high [used]
        line  19:    "SD_DAT1"          kernel   input  active-high [used]
        line  20:    "SD_DAT2"          kernel   input  active-high [used]
        line  21:    "SD_DAT3"          kernel   input  active-high [used]
        line  22:     "SD_CLK"          kernel   input  active-high [used]
        line  23:     "SD_CMD"          kernel   input  active-high [used]
        line  24:      "SD_CD"            "cd"   input  active-high [used]
        line  25:    "USB_PSU" "fe03a080.usb3phy" output active-low [used]
        line  26:  "VDDEE_PWM"          unused   input  active-high
        line  27: "VDDCPU_PWM"          kernel   input  active-high [used]
        line  28:         "LED"         "act"  output  active-high [used]
        line  29:   "DEBUG_TX"          kernel   input  active-high [used]
        line  30:   "DEBUG_RX"          kernel   input  active-high [used]
        line  31:       "PIN40"         unused   input  active-high
        line  32:       "PIN31"         unused   input  active-high
        line  33:       "PIN12"         unused   input  active-high
        line  34:           "-"         unused   input  active-high
        line  35:       "PIN32"         unused   input  active-high
```

```
line  36:         "PIN29"          unused     input   active-high
line  37:          "PIN8"          kernel     input   active-high [used]
line  38:         "PIN10"          kernel     input   active-high [used]
line  39:              "-"         unused     input   active-high
line  40:         "PIN35"          unused     input   active-high
line  41:     "HDMI_SDA"           kernel     input   active-high [used]
line  42:     "HDMI_SCL"           kernel     input   active-high [used]
line  43:     "HDMI_HPD"           kernel     input   active-high [used]
line  44:     "HDMI_CEC"           kernel     input   active-high [used]
line  45:         "PIN19"          kernel     input   active-high [used]
line  46:         "PIN21"          kernel     input   active-high [used]
line  47:         "PIN24"       "spi0.0"    output   active-high [used]
line  48:         "PIN23"          kernel     input   active-high [used]
line  49: "PCIE_RESET"             unused     input   active-high
line  50: "WIFI_SD_D0"             kernel     input   active-high [used]
line  51: "WIFI_SD_D1"             kernel     input   active-high [used]
line  52: "WIFI_SD_D2"             kernel     input   active-high [used]
line  53: "WIFI_SD_D3"             kernel     input   active-high [used]
line  54: "WIFI_SD_CLK" kernel input active-high [used]
line  55: "WIFI_SD_CMD" kernel input active-high [used]
line  56:              "-"         unused     input   active-high
line  57:              "-"         unused     input   active-high
line  58:              "-"         unused     input   active-high
line  59:              "-"         unused     input   active-high
line  60:         "BT_ON"     "shutdown"     output   active-high [used]
line  61:         "WL_ON"          unused     input   active-high
line  62: "BTUART_A_TX" kernel input active-high [used]
line  63: "BTUART_A_RX" kernel input active-high [used]
line  64: "BTUART_A_CTS_N" kernel input active-high [used]
line  65: "BTUART_A_RTS_N" kernel input active-high [used]
line  66:              "-"         unused     input   active-high
line  67:              "-"         unused     input   active-high
line  68:              "-"         unused     input   active-high
line  69:              "-"         unused     input   active-high
line  70:          "PIN3"          kernel     input   active-high [used]
line  71:          "PIN5"          kernel     input   active-high [used]
line  72:         "PIN18"          unused     input   active-high
line  73:         "PIN22"          unused     input   active-high
line  74:         "PIN37"          unused     input   active-high
line  75:         "PIN13"          unused     input   active-high
line  76:         "PIN15"          unused     input   active-high
line  77:         "PIN16"          unused     input   active-high
line  78:         "PIN26"       "spi0.1"    output   active-high [used]
line  79:         "PIN11"          unused     input   active-high
```

```
        line  80:        "PIN36"         unused     input   active-high
        line  81:        "PIN38"         unused     input   active-high
        line  82:        "PIN33"         unused     input   active-high
        line  83:         "PIN7"         unused     input   active-high
        line  84:    "LAN_LEDG"          kernel     input   active-high [used]
        line  85:    "LAN_LEDY"          kernel     input   active-high [used]
        line  86:           "-"         unused     input   active-high
```

可以看到系统只有一个 gpiochip0，共有 87 个 GPIO 管脚，已被驱动或系统占用的 GPIO 会在最后一列显示为[used].

4.1.2 检测按键

根据数据手册 **ED-GWL1010_Datasheet_CN.pdf** 文档中的 GPIO 管脚名称，结合 gpioinfo 的返回结果查找到对应的 line 号。GPIO23 对应管脚名称为 PIN16，PIN16 对应的 line 号为 77。

```
phantom@phantom:~ $ gpioinfo | grep PIN16
        line  77:        "PIN16"         unused     input   active-high
```

配置 GPIO 输入

```
#读取 chip0 的 line77 管脚状态
gpioget 0 77
```

返回值为 1 时，表示 line77 即 PIN16 管脚为高电平，返回值为 0 时，表示 line77 即 PIN16 管脚为低电平。

4.2 LED 指示

ED-GWL1010 的 RGB LED 由三个 GPIO 进行控制，控制引脚分别为 GPIO16 控制蓝色，GPIO20 控制绿色，GPIO21 控制红色，GPIO 低电平有效。

根据数据手册 **ED-GWL1010_Datasheet_CN.pdf** 文档中的 GPIO 管脚名称，结合 gpioinfo 的返回结果查找到对应的 line 号。
GPIO16 对应管脚名称为 PIN36，PIN36 对应的 line 号为 80。
GPIO20 对应管脚名称为 PIN38，PIN38 对应的 line 号为 81。
GPIO21 对应管脚名称为 PIN40，PIN40 对应的 line 号为 31。

配置蓝灯亮

```
#设置 chip0 的 line80 管脚为低电平
gpioset 0 80=0
```

配置蓝灯灭

```
#设置 chip0 的 line80 管脚为高电平
gpioset 0 80=1
```

配置绿灯亮

```
#设置 chip0 的 line81 管脚为低电平
gpioset 0 81=0
```

配置绿灯灭

```
#设置 chip0 的 line81 管脚为高电平
gpioset 0 81=1
```

配置红灯亮

```
#设置 chip0 的 line31 管脚为低电平
gpioset 0 31=0
```

配置红灯灭

```
#设置 chip0 的 line31 管脚为高电平
gpioset 0 31=1
```

4.3 USB

ED-GWL1010 具有一个 USB2.0 A 型接口和一个 USB3.0 A 型接口。

4.3.1 查看 USB 设备信息

显示 USB 设备

```
lsusb
```

显示信息如下：

```
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 005: ID 1a2c:2d23 China Resource Semico Co., Ltd Keyboard
Bus 001 Device 004: ID 30fa:0300    USB OPTICAL MOUSE
Bus 001 Device 003: ID 0424:9e00 Microchip Technology, Inc. (formerly SMSC)
LAN9500A/LAN9500Ai
Bus 001 Device 002: ID 1a40:0201 Terminus Technology Inc. FE 2.1 7-port Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

4.3.2 USB 存储设备挂载

您可以将外部硬盘、SSD 或 USB 棒连接到设备上的任一 USB 端口，并挂载文件系统以访问存储在其上的数据。

对于一般情况，您可以直接使用如下命令挂载或卸载外置存储设备。

```
lsblk
```

```
NAME          MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda             8:0    1 29.1G  0 disk
└─sda1          8:1    1 29.1G  0 part
mmcblk0       179:0    0 59.5G  0 disk
├─mmcblk0p1 179:1    0   256M  0 part /boot
└─mmcblk0p2 179:2    0 59.2G  0 part /
```

使用 mount 命令来挂载 sda1 到/mnt 目录，挂载完成后用户可以直接在/mnt 目录下操作存储设备。

```
sudo mount /dev/sda1 /mnt
```

使用完成以后使用命令 umount 卸载存储设备。

```
sudo umount /mnt
```

4.3.2.1 挂载

您可以将存储设备安装在特定的文件夹位置。通常在/mnt 文件夹中进行，例如/mnt/mydisk。请注意，文件夹必须是空的。

1. 将存储设备插入设备上的 USB 端口。
2. 使用以下命令列出系统上的所有磁盘分区:
   ```
   sudo lsblk -o UUID,NAME,FSTYPE,SIZE,MOUNTPOINT,LABEL,MODEL
   ```
   文件系统使用挂载点/和/boot。您的存储设备将显示在此列表中，以及任何其他连接的存储设备。
3. 使用“大小”、“标签”和“型号”列来标识指向您的存储设备的磁盘分区的名称。例如 sda1,mmcblk0。
4. FSTYPE 列包含文件系统类型。如果您的存储设备使用 exFAT 文件系统，请安装 exFAT 驱动程序:
   ```
   sudo apt update
   sudo apt install exfat-fuse
   ```
5. 如果您的存储设备使用 NTFS 文件系统，您将对其拥有只读访问权限。如果要写入设备，可以安装 ntfs-3g 驱动程序:
   ```
   sudo apt update
   sudo apt install ntfs-3g
   ```
6. 运行以下命令获取磁盘分区的位置:
   ```
   sudo blkid
   ```
   比如显示/dev/sda1
7. 创建一个目标文件夹作为存储设备的装载点。本例中使用的挂载点名称是 mydisk。您可以指定自己选择的名称:
   ```
   sudo mkdir /mnt/mydisk
   ```
8. 在您创建的装载点装载存储设备:
   ```
   sudo mount /dev/sda1 /mnt/mydisk
   ```
9. 通过列出以下内容来验证存储设备是否已成功装载:
   ```
   ls /mnt/mydisk
   ```

4.3.2.2 卸载

当设备关闭时，系统会负责卸载存储设备，以便安全地将其拔出。如果您想要手动卸载设备，可以使用以

下命令:

```
sudo umount /mnt/mydisk
```

如果您收到“目标繁忙”的错误，这意味着存储设备未卸载。如果没有显示错误，您现在可以安全地拔出设备。

4.3.2.3 命令行中设置自动挂载

可以通过修改 fstab 设置自动挂载。

1. 首先需要获取磁盘 UUID

   ```
   sudo blkid
   ```

2. 找到挂载设备的 UUID，例如 5C24-1453
3. 打开 fstab 文件

   ```
   sudo nano /etc/fstab
   ```

4. 添加以下内容到 fstab 文件中

   ```
   UUID=5C24-1453 /mnt/mydisk fstype defaults,auto,users,rw,nofail 0 0
   ```

   将 fstype 替换为您的文件系统的类型，您可以在上面的“挂载存储设备”的步骤 2 中找到，例如:ntfs。
5. 如果文件系统类型是 FAT 或 NTFS，则在 nofail 之后立即添加 umask = 000 这将允许所有用户对存储设备上的每个文件进行完全读/写访问。

关于更多 fstab 命令的信息可以使用 man fstab 来查看。

4.4 以太网配置

系统默认使用 dhcpcd 进行网络管理。

配置静态 IP，设置 eth0 网卡的静态 IP 为 192.168.168.108，设置默认路由为 192.168.168.1，设置 DNS 为 192.168.168.1(DNS 可以省略)：

```
sudo nano /etc/dhcpcd.conf

interface eth0
static ip_address=192.168.168.108/24
static route=192.168.168.1
static domain_name_servers=192.168.168.1
```

4.5 WiFi

4.5.1.1 扫描可用 WiFi 网络

```
sudo iwlist wlan0 scan
```

4.5.1.2 连接到 WiFi

方法 1：

```
sudo raspi-config
```

选择 1 System Options，找到 S1 Wireless LAN，第一次使用需要选择国家，中国是 CN，然后会要求输入 WiFi 名称，然后再输入 WiFi 密码，保存退出即可。如果设置了国家代码则需要重启一次。

方法 2：

```
sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

在文件中添加以下内容

```
country=CN
network={
      ssid="WiFi_SSID"
      psk="Password"
}
```

Ctrl+X 退出，回车保存。

4.6 蓝牙

系统默认使能蓝牙功能，如果需要对蓝牙进行设置，则可以使用 bluetoothctl 命令设置蓝牙。

扫描

```
bluetoothctl scan on/off
```

发现设备

```
bluetoothctl discoverable on/off
```

信任设备

```
bluetoothctl trust [MAC]
```

配对

```
bluetoothctl pair [MAC]
```

连接

```
bluetoothctl connect [MAC]
```

断开连接

```
bluetoothctl disconnect [MAC]
```

更多关于蓝牙命令配置

```
bluetoothctl
help
```

4.7 串口通信

ED-GWL1010 具有 1 路 TTL 电平串口，接口名称为 J4，默认作为调试串口。

4.7.1 安装 picocom 工具

picocom 串口终端可以在 Linux 环境下十分方便的进行串口调试。
首先安装 picocom

```
sudo apt-get install picocom
```

使用 picocom 打开对应串口后，您可以键入 Ctrl+a，然后键入 Ctrl+h 以查看可用命令。

```
*** Picocom commands (all prefixed by [C-a])

*** [C-x] : Exit picocom
*** [C-q] : Exit without reseting serial port
*** [C-b] : Set baudrate
*** [C-u] : Increase baudrate (baud-up)
*** [C-d] : Decrease baudrate (baud-down)
*** [C-i] : Change number of databits
*** [C-j] : Change number of stopbits
*** [C-f] : Change flow-control mode
*** [C-y] : Change parity mode
*** [C-p] : Pulse DTR
*** [C-t] : Toggle DTR
*** [C-g] : Toggle RTS
*** [C-|] : Send break
*** [C-c] : Toggle local echo
*** [C-w] : Write hex
*** [C-s] : Send file
*** [C-r] : Receive file
*** [C-v] : Show port settings
*** [C-h] : Show this message
```

先键入 Ctrl+a，然后键入 Ctrl+c 以切换本地回显模式。

先键入 Ctrl+a，然后键入 Ctrl+q 即可退出 picocom。

4.7.2 Debug UART

要启用调试串口，需要修改 config.txt 配置文件。

```
sudo nano /boot/config.txt
```

在最后面添加

```
[all]
enable_uart=1
```

调试串口默认波特率为 115200，您可以通过 cmdline.txt 文件查看当前调试串口波特率

```
sudo nano /boot/cmdline.tx
```

4.8 LoRaWAN

ED-GWL1010 支持 LoRaWAN 开源服务平台 ChipStack，请参考如下步骤进行安装和配置。

4.8.1 安装 LoRa 服务和 ChirpStack 客户端

我们通过 APT 的方式进行安装。

- 添加 edatec APT 仓库

```
$ curl -sS https://apt.edatec.cn/pubkey.gpg | sudo apt-key add -
$ echo "deb https://apt.edatec.cn/raspbian stable main" | sudo tee /etc/apt/sources.list.d/edatec.list
$ sudo apt update
$ sudo apt install -y ed-gwl-pktfwd
```

- 安装 ChirpStack

```
$ sudo apt install -y apt-transport-https dirmngr
$ sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 1CE2AFD36DBCCA00
$ echo "deb https://artifacts.chirpstack.io/packages/4.x/deb stable main" | sudo tee
/etc/apt/sources.list.d/chirpstack.list
$ sudo apt update

$ sudo apt install -y chirpstack-gateway-bridge
```

ED-GWL1010 使用 i2c-1 和 spidev0.0。

4.8.2 配置 LoRa 服务

4.8.2.1 pktfwd 配置

```
# update region
$ cat /etc/ed_gwl/region
EU868     # EU868 / US915
```

pktfwd 使用 1700 作为 UDP 端口

```
$ sudo systemctl restart ed-pktfwd.service
```

4.8.2.2 chirpstack-gateway-bridge 的配置

您可以使用 nano 进行配置文件 chirpstack-gateway-bridge.toml 的编辑

```
$ sudo nano /etc/chirpstack-gateway-bridge/chirpstack-gateway-bridge.toml
```

```
# This configuration provides a Semtech UDP packet-forwarder backend and
 # integrates with a MQTT broker. Many options and defaults have been omitted
 # for simplicity.
 #
 # See https://www.chirpstack.io/gateway-bridge/install/config/ for a full
 # configuration example and documentation.


 # Gateway backend configuration.
 [backend]
 # Backend type.
 type="semtech_udp"

    # Semtech UDP packet-forwarder backend.
    [backend.semtech_udp]

    # ip:port to bind the UDP listener to
    #
    # Example: 0.0.0.0:1700 to listen on port 1700 for all network interfaces.
    # This is the listener to which the packet-forwarder forwards its data
    # so make sure the 'serv_port_up' and 'serv_port_down' from your
    # packet-forwarder matches this port.
    udp_bind = "0.0.0.0:1700"

 # Integration configuration.
 [integration]
 # Payload marshaler.
 #
 # This defines how the MQTT payloads are encoded. Valid options are:
 # * protobuf:    Protobuf encoding
 # * json:        JSON encoding (easier for debugging, but less compact than 'protobuf')
 marshaler="protobuf"

    # MQTT integration configuration.
    [integration.mqtt]
    # Event topic template.
    event_topic_template="eu868/gateway/{{ .GatewayID }}/event/{{ .EventType }}"

    # Command topic template.
```

```
command_topic_template="eu868/gateway/{{ .GatewayID }}/command/#"

# MQTT authentication.
[integration.mqtt.auth]
# Type defines the MQTT authentication type to use.
#
# Set this to the name of one of the sections below.
type="generic"

  # Generic MQTT authentication.
  [integration.mqtt.auth.generic]
  # MQTT server (e.g. scheme://host:port where scheme is tcp, ssl or ws)
  server="tcp://127.0.0.1:1883"

  # Connect with the given username (optional)
  username=""

  # Connect with the given password (optional)
  password=""
```

- 'event_topic_template / command_topic_template' 需要修改带有网关区域的前缀。

  Example:

  ```
  event_topic_template="eu868/gateway/{{ .GatewayID }}/event/{{ .EventType }}"
  ```

  如果您使用 US915 或 CN470 模块请将前缀 eu868 修改为 us915_0 / cn470_10

  ```
  event_topic_template="us915_0/gateway/{{ .GatewayID }}/event/{{ .EventType }}"
  ```

- integration.mqtt 服务器地址需要是您的 chirpstack 服务器

  ```
  $ sudo systemctl restart chirpstack-gateway-bridge.service
  ```

修改 chirpstack-gateway-bridge.toml 配置后，重启 chirpstack-gateway-bridge 服务生效。

4.8.2.3 Reboot

```
$ sudo reboot
```

4.8.3 安装 ChirpStack 服务端

配置云端服务器，配置之前首先需要在服务器安装 docker。

安装 docker：https://docs.docker.com/get-docker/

安装 docker-compose

```
sudo apt install docker-compose
```

4.8.3.1 配置 chirpstack-docker

我们采用 docker 容器的方式部署 ChirpStack 服务端。

```
$ git clone https://github.com/chirpstack/chirpstack-docker.git
```

需要对 chirpstack-docker 的 docker-compose.yml 进行配置。

```
$ cd chirpstack-docker
$ nano docker-compose.yml
# Remove the chirpstack-gateway-bridge, because we run the bridge on gateway.
```

删除红色字体部分。

```
$ nano docker-compose.yml

version: "3"

services:
  chirpstack:
    image: chirpstack/chirpstack:4
    command: -c /etc/chirpstack
    restart: unless-stopped
    volumes:
      - ./configuration/chirpstack:/etc/chirpstack
      - ./lorawan-devices:/opt/lorawan-devices
    depends_on:
      - postgres
      - mosquitto
      - redis
    environment:
      - MQTT_BROKER_HOST=mosquitto
      - REDIS_HOST=redis
      - POSTGRESQL_HOST=postgres
    ports:
      - 8080:8080

  chirpstack-gateway-bridge-eu868:
    image: chirpstack/chirpstack-gateway-bridge:4
    restart: unless-stopped
    ports:
      - 1700:1700/udp
    volumes:
      - ./configuration/chirpstack-gateway-bridge:/etc/chirpstack-gateway-bridge
    depends_on:
      - mosquitto
```

```
  chirpstack-rest-api:
    image: chirpstack/chirpstack-rest-api:4
    restart: unless-stopped
    command: --server chirpstack:8080 --bind 0.0.0.0:8090 --insecure
    ports:
      - 8090:8090
    depends_on:
      - chirpstack

  postgres:
    image: postgres:14-alpine
    restart: unless-stopped
    volumes:
      - ./configuration/postgresql/initdb:/docker-entrypoint-initdb.d
      - postgresqldata:/var/lib/postgresql/data
    environment:
      - POSTGRES_PASSWORD=root

  redis:
    image: redis:7-alpine
    restart: unless-stopped
    volumes:
      - redisdata:/data

  mosquitto:
    image: eclipse-mosquitto:2
    restart: unless-stopped
    ports:
      - 1883:1883
    volumes:
      - ./configuration/mosquitto/mosquitto.conf:/mosquitto/config/mosquitto.conf

volumes:
  postgresqldata:
  redisdata:
```

启动 chirpstack 服务端

```
$ docker-compose up -d
```

4.8.3.2 登录 chirpstack 服务管理界面

在 PC 端浏览器输入服务器的 IP 地址和端口号 8080，在网络正常的情况下会出现登录界面
默认管理员用户名和密码如下：

```
user: admin
psw : admin
```

4.8.4 添加 LoRa 网关和终端

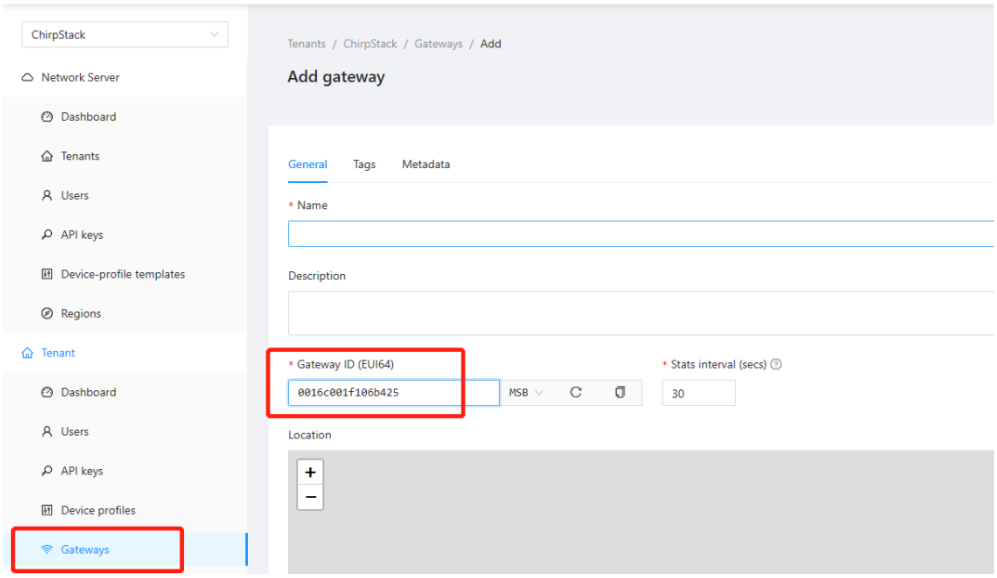
4.8.4.1 获取 LoRa 网关 ID

执行如下指令即可获得 LoRa 网关的 ID，在向 chirpstack 服务端添加 LoRa 网关时，需要填加对应的网关 ID。

```
$ /opt/ed-gwl-pktfwd/ed-gateway_id
```

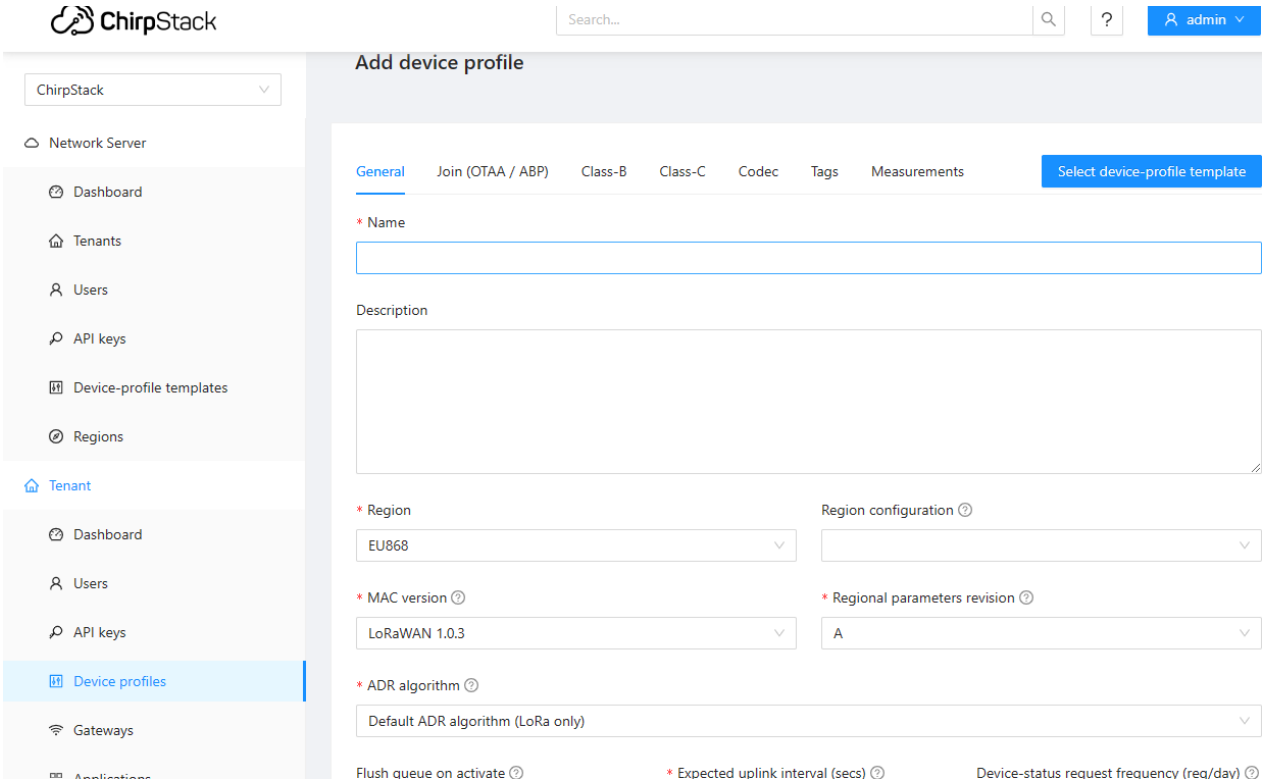
4.8.4.2 添加 LoRa 网关

在 PC 端浏览器打开 chirpstack 管理界面，点击 Gateway -> Add gateway，填入设备对应的 Gateway ID，并设置 Name，然后点击 Submit，如果网络连接正确，稍等片刻即可看到添加的网关变为 Online 状态。



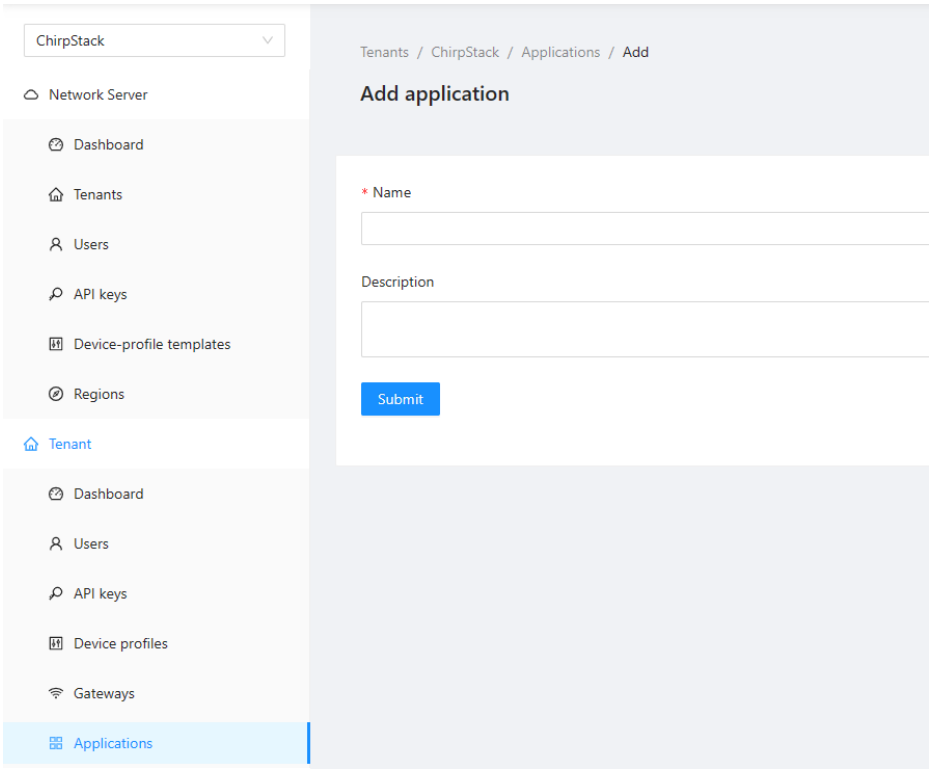
4.8.4.3 Add Device Profile

点击 Device Profile -> Add device profile 可以进一步完善设备信息。



4.8.4.4 Add Application

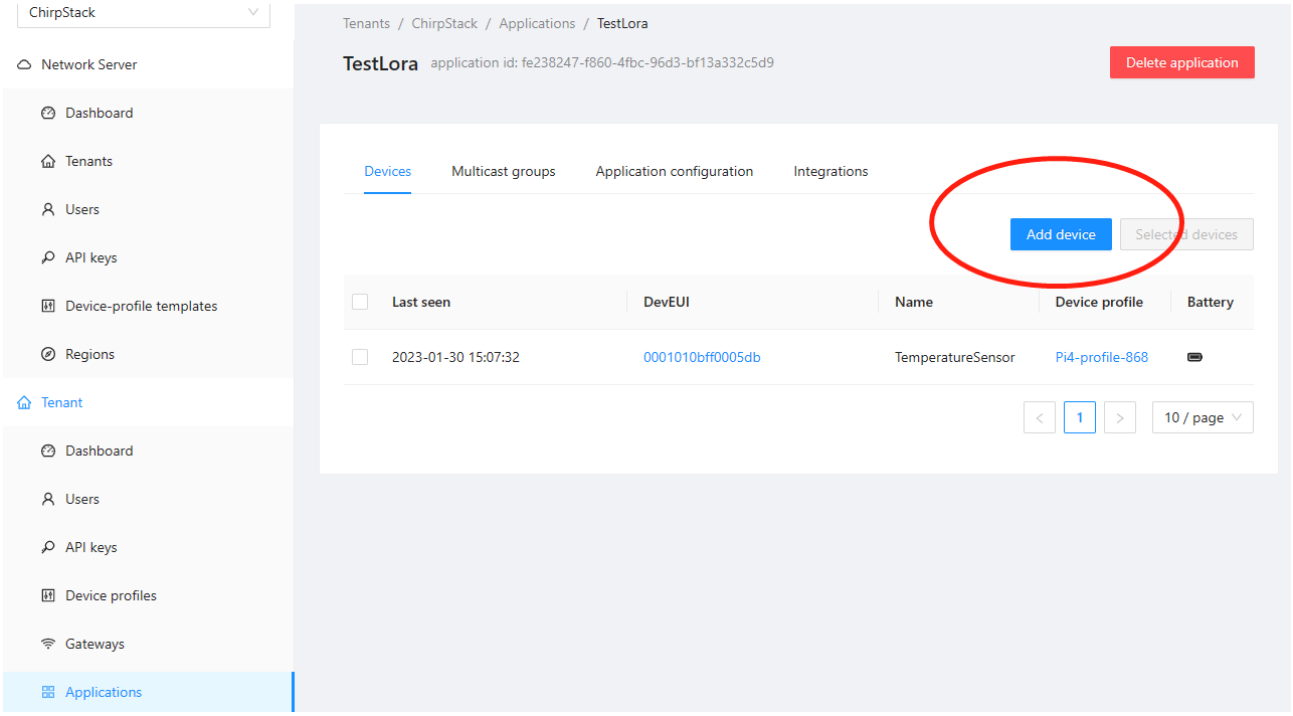
点击 Applications -> Add application

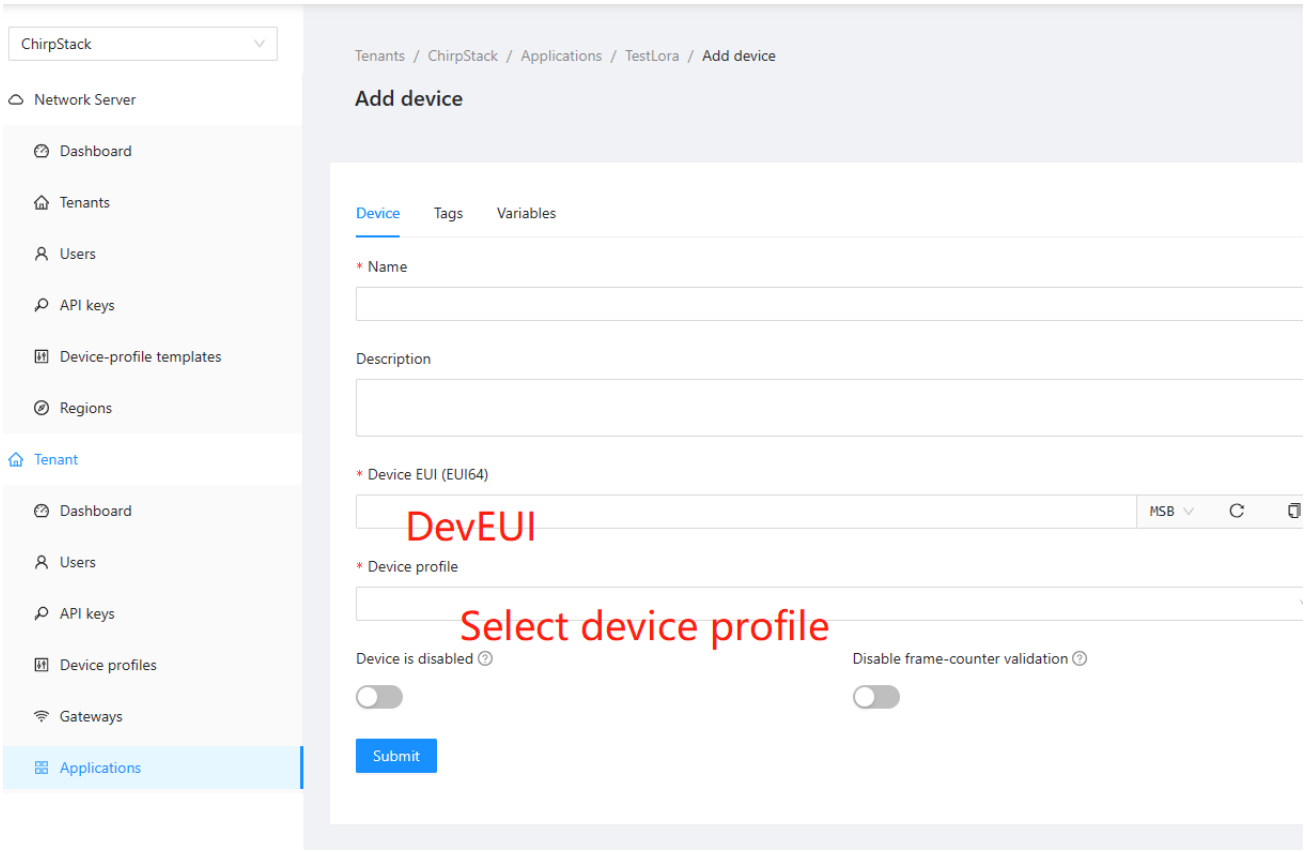


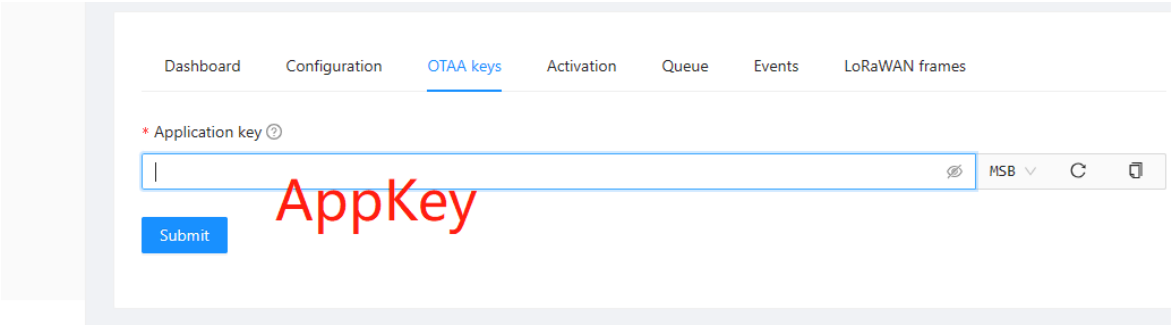
4.8.4.5 Add Device

应该知道 LoRa 终端产品的 DevEUI 和 AppKey，它们由 LoRa 终端设备制作商提供。

点击 Application -> your application -> Add device 进行 LoRa 终端设备的添加。







等待几分钟即可看到设备变成 online 状态。

5 操作系统安装

5.1 镜像下载

在出厂时我们已经在 eMMC 中烧录了系统，用户可以直接使用。
我们提供了出厂镜像，如果系统恢复出厂设置，请点击以下链接下载出厂镜像。

下载地址：https://www.123pan.com/s/PNDSVv-hboxA.html 提取码:mmrZ

5.2 系统烧录

ED-GWL1010 支持 SD 卡和 eMMC 系统双启动，SD 卡启动的优先级更高。

如果希望烧写系统到 eMMC，需要系统已通过 SD 卡启动，然后通过 dd 命令间接烧录 eMMC。

5.2.1 烧录 SD 卡

安装烧录工具，推荐 balenaEtcher：

- balenaEtcher：https://www.balena.io/etcher/
- SD 卡：至少使用 8GB 以上容量的 SD 卡(如果打算使用 SD 卡烧录 eMMC，SD 卡容量最低要求 16GB 以上)

烧录步骤：

1. 打开 balenaEtcher，选择要烧录的文件
2. 选择要烧录的 SD 卡
3. 等待烧录完成

启用 SSH：

镜像默认禁能 SSH 功能，如果希望开机后使用 SSH 远程连接到设备，则需要在开机前在 boot 分区下创建一个名为 ssh 的空文件，以保证设备开机后自动使能 SSH 功能。

5.2.2 烧录 eMMC

eMMC 目前仅支持从 SD 卡烧录，出厂时默认已经在 eMMC 中烧录好镜像，用户可以直接使用，如果发现设备无法启动，绿色指示灯不闪烁，则说明此时系统无法启动，需要使用 SD 卡将镜像烧入 eMMC 中。

```
lsblk

NAME             MAJ:MIN RM    SIZE RO TYPE MOUNTPOINT
mmcblk0          179:0       0 14.8G    0 disk
 ├─mmcblk0p1    179:1       0    256M    0 part /boot
 └─mmcblk0p2    179:2       0 14.6G    0 part /
mmcblk1          179:32     0    7.3G    0 disk
 └─mmcblk1p1    179:33     0    7.3G    0 part
mmcblk1boot0 179:64     0       4M    0 disk
mmcblk1boot1 179:96     0       4M    0 disk
```

SD 卡的分区名为 mmcblk0，可以看见 SD 卡有两个分区，一个是 mmcblk0p1，另一个是 mmcblk0p2。第二个为 eMMC 的分区，因为默认没有烧录系统所以只有一个分区 mmcblk1p1。
如果第二个分区已经烧录系统，那么使用 lsblk 命令后会显示以下内容

```
lsblk

NAME             MAJ:MIN RM    SIZE RO TYPE MOUNTPOINT
mmcblk0          179:0       0 14.8G    0 disk
 ├─mmcblk0p1    179:1       0    256M    0 part /boot
 └─mmcblk0p2    179:2       0 14.6G    0 part /
mmcblk1           179:32     0    7.3G    0 disk
 ├─mmcblk1p1    179:33     0    256M    0 part
 └─mmcblk1p2    179:34     0    5.9G    0 part
mmcblk1boot0     179:64     0       4M    0 disk
mmcblk1boot1     179:96     0       4M    0 disk
```

烧录准备

eMMC 烧录只能通过 SD 卡写入，所以首先需要一张已经烧录好系统的 SD 卡，启动系统，将即将烧录的系统放入 SD 卡中，示例中将镜像直接放在默认用户 phantom 的文件夹下，文件夹绝对路径为 /home/phantom。

烧录系统到 eMMC：

```
sudo dd if=<镜像路径> of=/dev/mmcblk1 bs=4MiB
#示例
sudo dd if=/home/phantom/phantom_2022-12-03.img of=/dev/mmcblk1 bs=4MiB
sync
```

耐心等待命令执行完毕。
执行完毕后会显示以下内容：

```
1483+1 records in
1483+1 records out
```

使用 lsblk 可以看到 mmcblk1 拥有 p1，p2 两个分区：

```
lsblk

NAME            MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
mmcblk0         179:0    0  14.8G  0 disk
 ├─mmcblk0p1    179:1    0   256M  0 part /boot
 └─mmcblk0p2    179:2    0  14.6G  0 part /
mmcblk1         179:32   0   7.3G  0 disk
 ├─mmcblk1p1    179:33   0   256M  0 part
 └─mmcblk1p2    179:34   0   5.9G  0 part
mmcblk1boot0    179:64   0    4M   0 disk
mmcblk1boot1    179:96   0    4M   0 disk
```

启用 SSH：
默认镜像没有使能 SSH 服务，如果希望开机即可使用 SSH 远程连接到设备，则按照以下步骤操作：

```
sudo mount /dev/mmcblk1p1 /mnt
sudo touch /mnt/ssh
sudo umount /mnt
```

6 FAQ

6.1.1 默认用户名密码

我们出厂镜像默认用户名是 phantom，默认密码是 phantom。

7 关于我们

7.1 关于 EDATEC

EDATEC 位于上海，是 Raspberry Pi 的全球设计合作伙伴之一。我们的愿景是提供基于 Raspberry Pi 技术平台的物联网、工业控制、自动化、绿色能源和人工智能的硬件解决方案。

我们提供标准的硬件解决方案，定制设计和制造服务，以加快电子产品的开发和上市时间。

7.2 联系方式

邮箱 - sales@edatec.cn / support@edatec.cn
手机 - +86-18621560183
网站 - https://www.edatec.cn
地址 - 上海市嘉定区嘉罗公路 1661 号 24 栋 301 室