4.6 battery power inquiry

1. Introduction to battery power inquiry driver

The battery power query driver is also customized by Yahboom officially for Jetbot. We provide the matching Battery_Vol_Lib.py driver. Put the driver file in the same folder as the program we need to use it.

The hardware that implements this feature is designed on the hardware of the RGB programming light strip.

Path of package : [Jetbot-AI Car] --> [Annex] --> [Driver file]---> [Battery_Vol_Lib.py]

The initialization of the driver is done when we first create the RGB instance.

```
# Create a battery charge query object
BatteryLevel = BatteryLevel()
```

The driver is as follows：

```
  def Update(self):
      AD_value = self._device.readList(0x00,2)
      Battery_vol = ((AD_value[0] << 8) + AD_value[1]) * 13.3 / 1023.0
      print(Battery_vol)
      ##Charge power judgment
      if Battery_vol >= 12:
          return 'Battery_High'
      elif Battery_vol >= 11.1:
          return 'Battery_Medium'
      elif Battery_vol >= 10.05:
          return 'Battery_Low'
      ##Discharge power judgment
      if Battery_vol <= 9.9:
          return 'Battery_Empty'
      elif Battery_vol <= 10.95:
          return 'Battery_Low'
      elif Battery_vol <= 11.85:
          return 'Battery_Medium'
```

We can get the current battery status by simply calling the example method shown above.

- Battery_High
- Battery_Medium
- Battery_Low
- Battery_Empty

2. Battery power drive

In this course, we refreshed the displayed battery level in real time in the lower right corner of the OLED screen.

Before this, in order to avoid Jetbot's real-time refresh display, the stats.py program running by jetbot_stats.service and the program we are testing now use an I2C peripheral OLED screen at the same time and generate an exception conflict.
We need to stop refresh the operation of the OLED service at the Jetbot command console.
sudo systemctl stop jetbot_start

After executing the above command, we can observe that there is no value refresh display on the screen.
Next, run the code we want to test:
First, import the modules we need to use：

Import related driver libraries

```
import time
import Adafruit_SSD1306
from PIL import Image
from PIL import ImageDraw
from PIL import ImageFont
from jetbot.utils.utils import get_ip_address
import subprocess
from Battery_Vol_Lib import BatteryLevel
```


Next, we create the SSD1306 object instance that we need to use and create the variables we need to use. When creating an OLED object instance, the driver initialization function is automatically called to initialize the OLED.

Create the BatteryLevel object instance we need to use and create the variables we need to use. When we create the object instance, we will automatically call the driver initialization function to initialize the battery query driver.

The specific code is shown below:

Initialization of SSD1306

```
# 128x32 Display and hardwareI2C:
disp = Adafruit_SSD1306.SSD1306_128_32(rst=None, i2c_bus=0, gpio=1) # Set gpio to hack to 1 to avoid platform detection
# Initialization of library.
disp.begin()
# clear display
disp.clear()
disp.display()
# Create a blank image for the drawing
# Make sure to create an image with a mode of '1' or a 1-bit color
width = disp.width
height = disp.height
image = Image.new('1', (width, height))
# Get the drawing object to be drawn on the image
draw = ImageDraw.Draw(image)
# Draw a black filled box to clear the image
draw.rectangle((0,0,width,height), outline=0, fill=0)
# Draw some shapes
# First,we need to define some constants to adjust the size of the shape
padding = -2
top = padding
bottom = height-padding
# Move from left to right to track the current x position of the drawing graph.
x = 0
# Load default font
font = ImageFont.load_default()
# Create a battery charge query object
BatteryLevel = BatteryLevel()
```


After running the above code, we can see that the display on the OLED screen is cleared, which means that our code is initialized for both the OLED screen and the battery query driver.

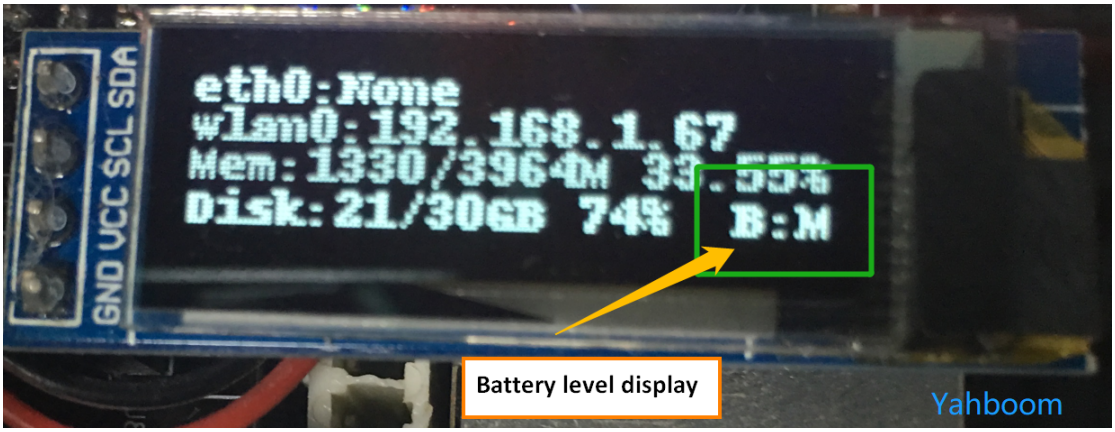
Next, we run the following code to start collecting Jetbot usage data in real time every second and refresh the display to the onboard OLED display.

Cycle refresh information display to OLED screen every second

Here, based on the OLED routine, the captured power information is displayed on the screen at the bottom right corner of the OLED screen. among them: B: H stands for high battery B: M stands for medium power B: L stands for low battery B: E stands for empty battery

```
while True:
    #  Draw a black filled box to clear the image.
    draw.rectangle((0,0,width,height), outline=0, fill=0)

    # From this link you can get the shell script for system monitoring:
    # https://unix.stackexchange.com/questions/119126/command-to-display-memory-usage-disk-usage-and-cpu-load
    cmd = "top -bn1 | grep load | awk '{printf \"CPU Load: %.2f\", $(NF-2)}'"
    CPU = subprocess.check_output(cmd, shell = True )
    cmd = "free -m | awk 'NR==2{printf \"Mem:%s/%sM %.2f%%\", $3,$2,$3*100/$2 }'"
    MemUsage = subprocess.check_output(cmd, shell = True )
    cmd = "df -h | awk '$NF==\"/\"{printf \"Disk:%d/%dGB %s\", $3,$2,$5}'"
    Disk = subprocess.check_output(cmd, shell = True )

    draw.text((x, top),       "eth0:" + str(get_ip_address('eth0')),  font=font, fill=255)
    draw.text((x, top+8),     "wlan0:" + str(get_ip_address('wlan0')), font=font, fill=255)
    draw.text((x, top + 16), str(MemUsage.decode('utf-8')), font=font, fill=255)

    #Get power by calling the battery power drive library
    temp = BatteryLevel.Update()
    if temp == 'Battery_High':
        draw.text((x, top+25),    str(Disk.decode('utf-8')) + "  B:H",  font=font, fill=255)
        print('Battery_High')
    elif temp == 'Battery_Medium':
        draw.text((x, top+25),    str(Disk.decode('utf-8')) + "  B:M",  font=font, fill=255)
        print('Battery_Medium')
    elif temp =='Battery_Low':
        draw.text((x, top+25),    str(Disk.decode('utf-8')) + "  B:L",  font=font, fill=255)
        print('Battery_Low')
    elif temp == 'Battery_Empty':
        draw.text((x, top+25),    str(Disk.decode('utf-8')) + "  B:E",  font=font, fill=255)
        print('Battery_Empty')
    # draw.text((x, top+25),    str(Disk.decode('utf-8')),  font=font, fill=255)

    # Display image
    disp.image(image)
    disp.display()
    time.sleep(1)
```


After running the above code, the current usage information of Jetbot robot car will be displayed on the OLED display, and the battery power information will be displayed in the lower right corner of the screen, as shown below.



- Battery_High ----Display B:H
- Battery_Medium ----Display B:M
- Battery_Low ----Display B:L
- Battery_Empty ----Display B:E

The corresponding complete source code is located at:
/home/jetbot/Notebook/7.Battery power inquiry/Battery power inquiry.ipynb