5.5 Battery power inquiry

Note: In the image operating system provided by Yahboom, the APP remote control process is enabled by default, in order to avoid multiple occupations of internal resources, causing some functions to fail to operate normally.

Before you running the code of this course, please follow the following to close the APP remote control process.

If you want to permanently close the function of the APP control process that starts automatically after booting, execute the following command:

```
sudo systemctl disable jetbotmini_start
```

If you want to permanently open the function of the APP control process that starts automatically after booting, execute the following command:

```
sudo systemctl enable jetbotmini_start
```

If you do not restart Jetbotmini to restart the APP control process function, execute the following command:

```
sudo systemctl stop jetbotmini_start
sudo systemctl start jetbotmini_start
```

5.5.1 Introduction to battery power drive

The battery level query driver is also customized by Yahboom for Jetbotmini. The Battery_Vol_Lib.py driver for it is also very convenient to use. No installation steps are required. Place the driver file in the same directory as the program we need. Ready to use.

Initialization of the driver:

```
from Battery_Vol_Lib import BatteryLevel
```

We see that 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)
        #Charging 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'
```

This means that we can immediately get the current battery power status as long as we call the example method shown above

- Battery_High
- Battery_Medium
- Battery_Low
- Battery_Empty

Next, we will try to get the real-time battery status and display it on the OLED screen in the next section of the course.

5.5.2 Battery power drive use

The Jetbotmini system has enabled the process of OLED real-time refresh display by default. In this course, we need to display the battery voltage in real time through OLED. The same IIC device used by the above two processes.

In order to avoid conflicts between them, we need to close the process of OLED real-time refresh display before running the code of this course.

```
sudo systemctl stop jetbotmini_stats
```

After running above command. We can observe that there is no value refresh display on the screen, then let's run the code we want to test:

The operation is similar to the previous section. First, we import the modules we need to use

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

Then we create the SSD1306 object instance we need to use and create the variables that we need to use. When creating the OLED object instance, we will automatically call the driver initialization function to initialize the OLED screen, create the BatteryLevel object instance we need to use and Create the variables that need to be used. When creating the object instance, it will automatically call the driver initialization function to initialize the battery query driver. For the specific content, we can see the comments in the code cell below:

```
# 128x32 display and hardware I2C:
disp = Adafruit_SSD1306.SSD1306_128_32(rst=None, i2c_bus=0, gpio=1)# Set GPIO
hack to 1 to avoid platform detection
# Initialize the library.
disp.begin()
# Clear display
disp.clear()
disp.display()
```

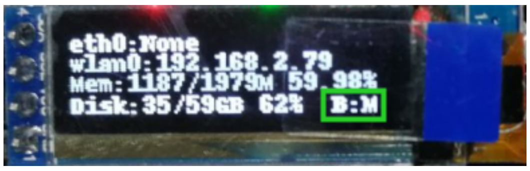
```
# Create a blank image for the drawing
# Make sure to create an image with the mode '1', i.e. 1-bit color
width = disp.width
height = disp.height
image = Image.new('1', (width, height))
# Gets the drawing object to draw on the image
draw = ImageDraw.Draw(image)
# Draw a black filled box to clear the image
draw.rectangle((0,0,width,height), outline=0, fill=0)
# Draw some shapes
# First, define some constants to make it easy to resize the shape
padding = -2
top = padding
bottom = height-padding
# Move left to right to track the current x position of the drawing.
x = 0
# Load default font
font = ImageFont.load_default()
# Create battery level query object
BatteryLevel = BatteryLevel()
```

After running the above code, we find that the display content on the OLED screen is now cleared, which means that our code has initialized the OLED screen and battery query drive, and then we run the following cell code to start each The usage data of Jetbotmini is collected in one second in real time and refreshed to the onboard OLED display.

```
while True:
    # Draw a black filled box to clear the image.
    draw.rectangle((0,0,width,height), outline=0, fill=0)
    # ou can get the shell script for system monitoring from this link :
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 the power by calling the battery power driver
    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 OLED display will immediately display the current usage information of Jetbotmini as shown below, and the battery power information will be displayed in the lower right corner of the screen. I use a green box to indicate where it is.



```
eth0:None
wlan0:192.168.2.79
Mem:1187/1979M 59.98%
Disk:35/59GB 62%  B:M
```

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

When the battery is too low, the blue LED (STATE) on the board will also change to fast flashing

The corresponding complete source code is located:

/home/jetson/Notebooks/English/6.Battery_power_query/Battery_power_query.ipynb