5.4 Use of onboard OLED screen

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
```

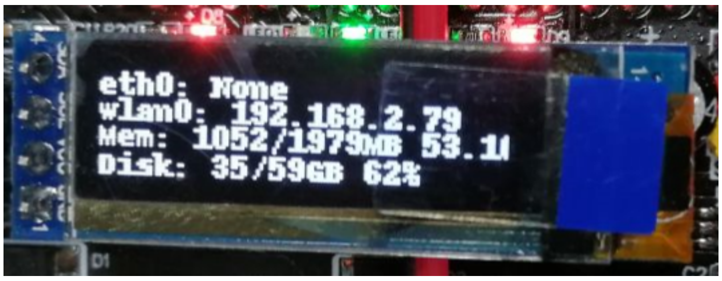
If you do not restart Jetbotmini to restart the APP function, execute the following command:

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

5.4.1 Introduction to onboard OLED screen driver

The onboard OLED screen uses the Adafruit_Python_SSD1306 driver. As the name implies, the driver name of the OLED screen is called SSD1306. This is a very common OLED driver. If you are interested, you can see [JetBot-Mini-Robot-Car] --> [Annex]- -> [JetBot-Mini Dependency package]--> [Adafruit_Python_SSD1306]--> [Adafruit_SSD1306]--> [SSD1306.py], we can see the OLED I2C peripheral address is 0x3C, which is stored Regarding all methods and register information for driving the screen, the screen we use is a model with a resolution of 128*32. The information it can display is enough to provide us with the information we need to know about the use of Jetbotmini. For the specific display content, we see the use below introduce.

5.4.2 Display IP address, system, memory usage information on the I2C screen



```
eth0: None
wlan0: 192.168.2.79
Mem: 1052/1979MB 53.1
Disk: 35/59GB 62%
```

As shown in the figure above, the Jetbotmini usage information interface displayed by OLED, the displayed contents are as follows:

1. The first line of content: the IP address information of the wired network
2. The second line of content: the IP address information of the WIFI wireless network

3. The third line of content: running memory usage and percentage of running memory used
4. The fourth line of content: Disk memory usage and the percentage of used disk memory

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

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
```

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. For the specific content, we can see the following code unit Notes in the box:

```
# 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()
```

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 then we run the following cell code to start real-time collection of Jetbotmini every one second The usage data is 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)
    # You 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)
    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 Jetbotmini will be displayed on the OLED display immediately.

The corresponding complete source code is located:

/home/jetson/Notebooks/English/5.Use_of_OLED/Use_of_OLED.ipynb