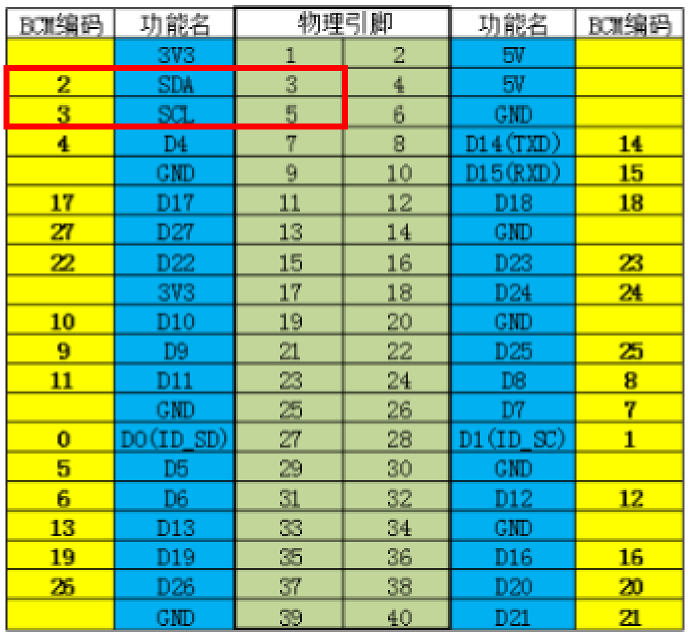
I2C communication

The I2C pins of the motherboard are shown in the figure, where physical pins 3 and 5 are I2C pins. You need to enable the I2C service before using it.

GPIO和BCM对照表



BCM编码	功能名	物理引脚		功能名	BCM编码
	3V3	1	2	5V	
2	SDA	3	4	5V	
3	SCL	5	6	GND	
4	D4	7	8	D14(TXD)	14
	GND	9	10	D15(RXD)	15
17	D17	11	12	D18	18
27	D27	13	14	GND	
22	D22	15	16	D23	23
	3V3	17	18	D24	24
10	D10	19	20	GND	
9	D9	21	22	D25	25
11	D11	23	24	D8	8
	GND	25	26	D7	7
0	D0(ID_SD)	27	28	D1(ID_SC)	1
5	D5	29	30	GND	
6	D6	31	32	D12	12
13	D13	33	34	GND	
19	D19	35	36	D16	16
26	D26	37	38	D20	20
	GND	39	40	D21	21

First install I2Ctool, enter in the terminal:

```
sudo apt-get update
sudo apt-get install -y i2c-tools
```

Check the installation status, enter in the terminal:

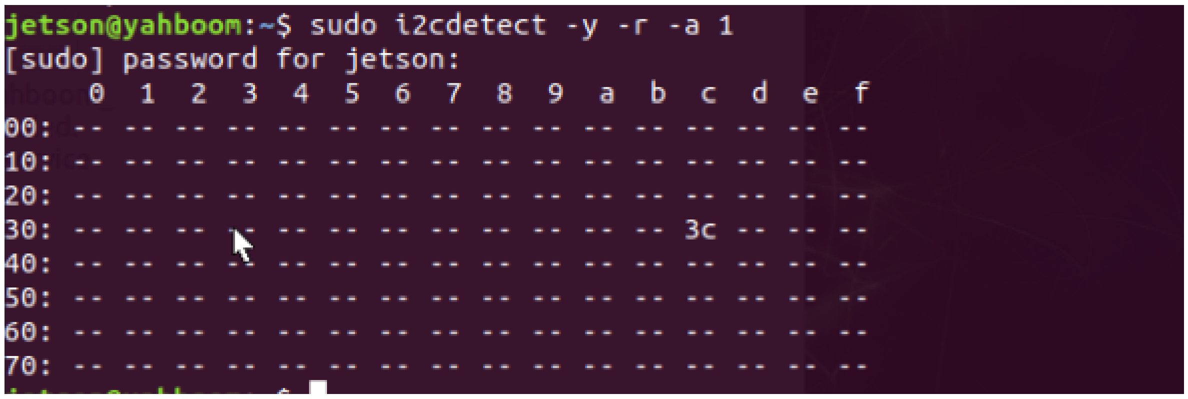
```
apt-cache policy i2c-tools
```

The output is as follows, which means the installation is successful

```
i2c-tools:
Installed: 4.0-2
Candidate: 4.0-2
Version list:
*** 4.0-2 500
500 http://ports.ubuntu.com/ubuntu-ports bionic/universe arm64 Packages
100 /var/lib/dpkg/status
```

Scan all i2c devices on a bus and print out the device i2c bus address, for example here If a device with address 0x0f is mounted on the I2C pin, the corresponding device I2C address will be displayed

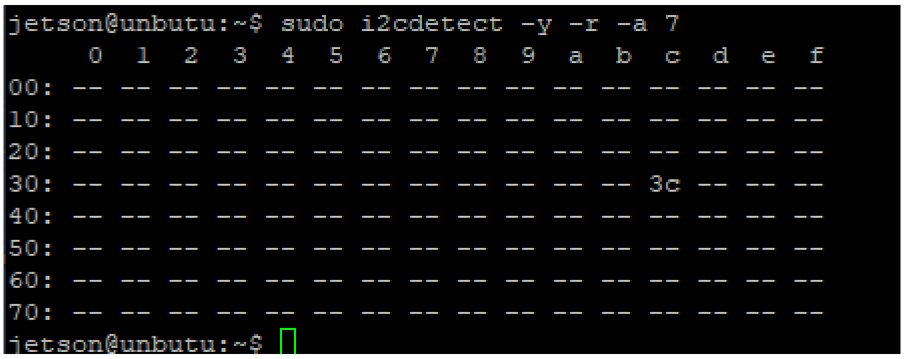
```
sudo i2cdetect -y -r -a 1
```



```
jetson@yahboom:~$ sudo i2cdetect -y -r -a 1
[sudo] password for jetson:
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- 3c -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
```

If you are using the Jetson Orin NX or Jetson Orin Nano motherboard, you need to change the device number from 1 to 7.

```
sudo i2cdetect -y -r -a 7
```

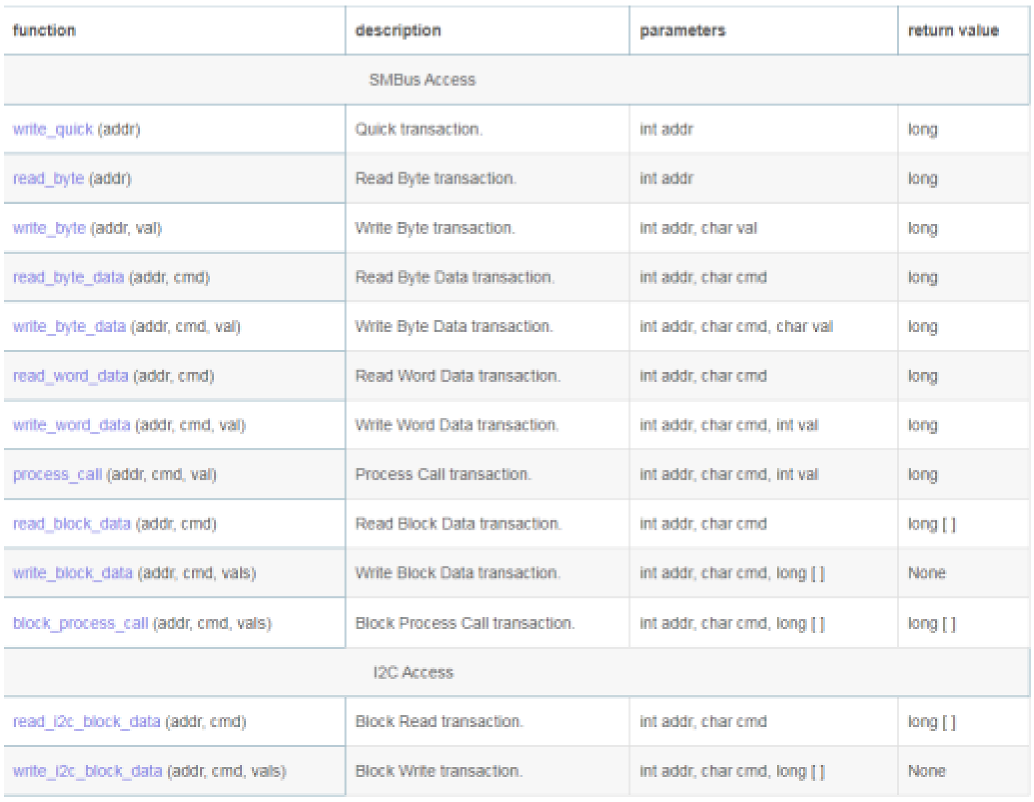


```
jetson@unbutu:~$ sudo i2cdetect -y -r -a 7
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- 3c -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
jetson@unbutu:~$
```

smbus is a python library. If smbus is not installed, enter the terminal:

```
sudo apt-get update
sudo apt-get install -y python3-smbus
```

The Smbus protocol has many related library functions that can be used for I2C communication



function	description	parameters	return value
SMBus Access			
write_quick (addr)	Quick transaction.	int addr	long
read_byte (addr)	Read Byte transaction.	int addr	long
write_byte (addr, val)	Write Byte transaction.	int addr, char val	long
read_byte_data (addr, cmd)	Read Byte Data transaction.	int addr, char cmd	long
write_byte_data (addr, cmd, val)	Write Byte Data transaction.	int addr, char cmd, char val	long
read_word_data (addr, cmd)	Read Word Data transaction.	int addr, char cmd	long
write_word_data (addr, cmd, val)	Write Word Data transaction.	int addr, char cmd, int val	long
process_call (addr, cmd, val)	Process Call transaction.	int addr, char cmd, int val	long
read_block_data (addr, cmd)	Read Block Data transaction.	int addr, char cmd	long [ ]
write_block_data (addr, cmd, vals)	Write Block Data transaction.	int addr, char cmd, long [ ]	None
block_process_call (addr, cmd, vals)	Block Process Call transaction.	int addr, char cmd, long [ ]	long [ ]
I2C Access			
read_i2c_block_data (addr, cmd)	Block Read transaction.	int addr, char cmd	long [ ]
write_i2c_block_data (addr, cmd, vals)	Block Write transaction.	int addr, char cmd, long [ ]	None

Wiring situation:

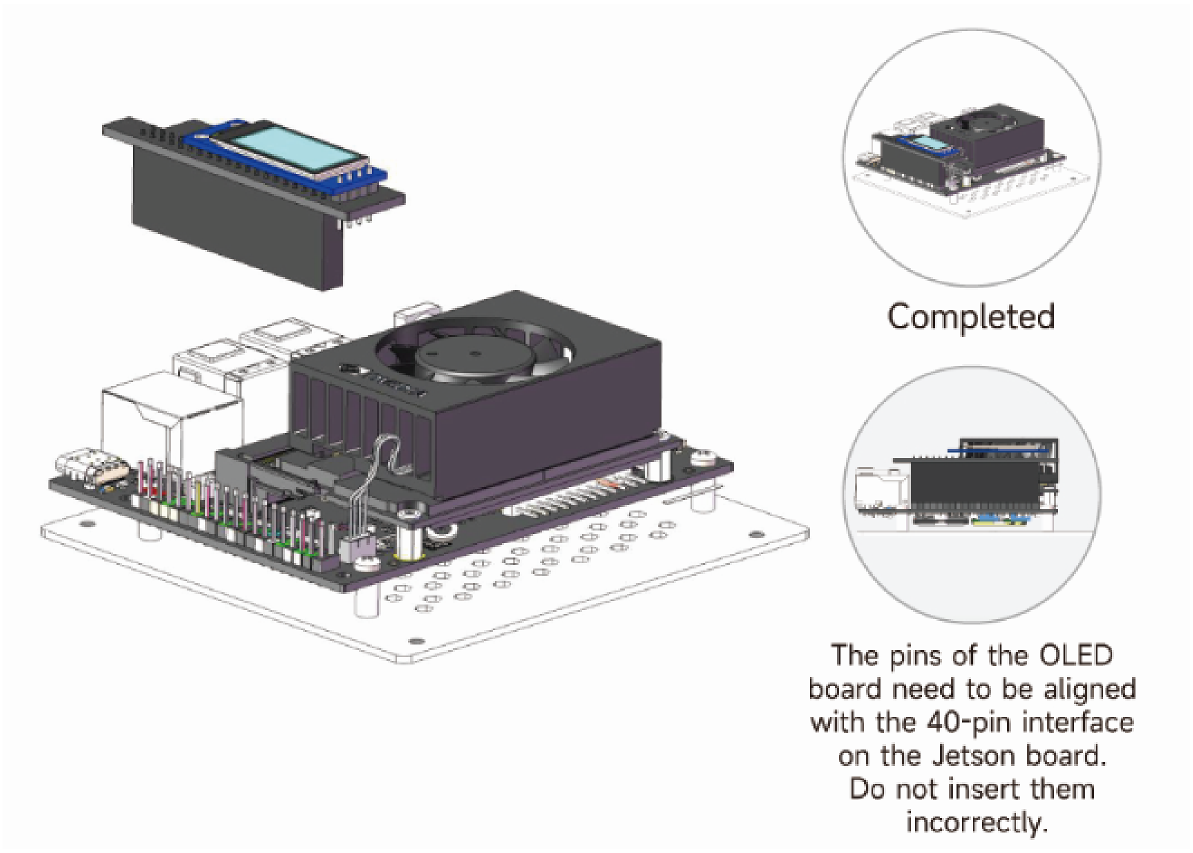
Pin 3 (SDA) → oled module SDA

Pin 5 (SCL) → oled module SCL

Pin 1 (3.3V) → oled module VCC

Pin 6 (GND) → oled module GND

The OLED module has been connected to the wire sequence, just install it in the GPIO direction.


Completed
The pins of the OLED board need to be aligned with the 40-pin interface on the Jetson board. Do not insert them incorrectly.

Import Adafruit_SSD1306 library This is the oled library. If you use your own image, you need to download this library

```
pip install Adafruit_SSD1306
```

```
#!/usr/bin/env python3
# coding=utf-8
import time
import os

import Adafruit_SSD1306 as SSD

from PIL import Image
from PIL import ImageDraw
from PIL import ImageFont

import subprocess

# V1.0.1
class Yahboom_OLED:
    def __init__(self, i2c_bus=1, debug=False):
        self.__debug = debug
        self.__i2c_bus = i2c_bus
        self.__top = -2
        self.__x = 0

        self.__total_last = 0
        self.__idle_last = 0
        self.__str_CPU = "CPU:0%"

    def __del__(self):
        if self.__debug:
            print("---OLED-DEL---")

    # 初始化OLED，成功返回:True，失败返回:False
    # Initialize OLED, return True on success, False on failure
```

Initialize oled:

```
# 初始化OLED，成功返回:True，失败返回:False
# Initialize OLED, return True on success, False on failure
def begin(self):
    try:
        self.__oled = SSD.SSD1306_128_32(
            rst=None, i2c_bus=self.__i2c_bus, gpio=1)
        self.__oled.begin()
        self.__oled.clear()
        self.__oled.display()
        self.__width = self.__oled.width
        self.__height = self.__oled.height
        self.__image = Image.new('1', (self.__width, self.__height))
        self.__draw = ImageDraw.Draw(self.__image)
        self.__font = ImageFont.load_default()
        if self.__debug:
            print("---OLED begin ok!---")
        return True
    except:
        if self.__debug:
            print("---OLED no found!---")
        return False
```

Then read some basic information functions of nano. If you are interested, you can go to this py file to learn more about it, which is to obtain local IP, TF card space occupancy, memory occupancy, system time and other information.

Terminal input:

```
sudo python3 ~/software/oled_yahboom/yahboom_oled.py
```

Note: The system defaults to enabling the OLED screen service. Running the program again will cause repeated screen refreshes. Please run the following command to stop the OLED screen service.

```
sudo systemctl stop yahboom_oled.service
```

Experimental phenomenon: