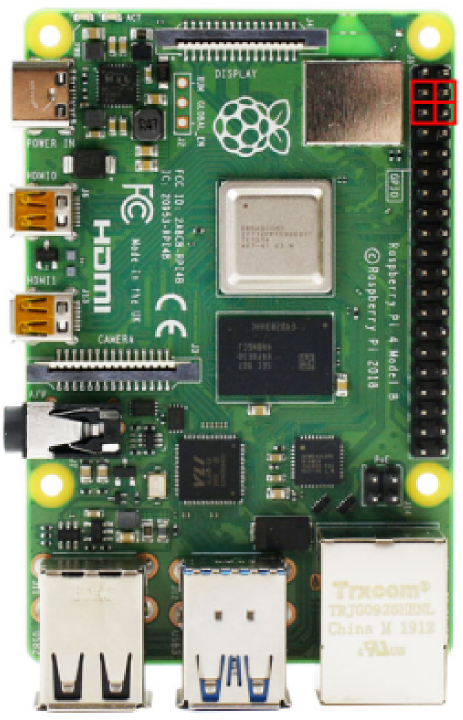
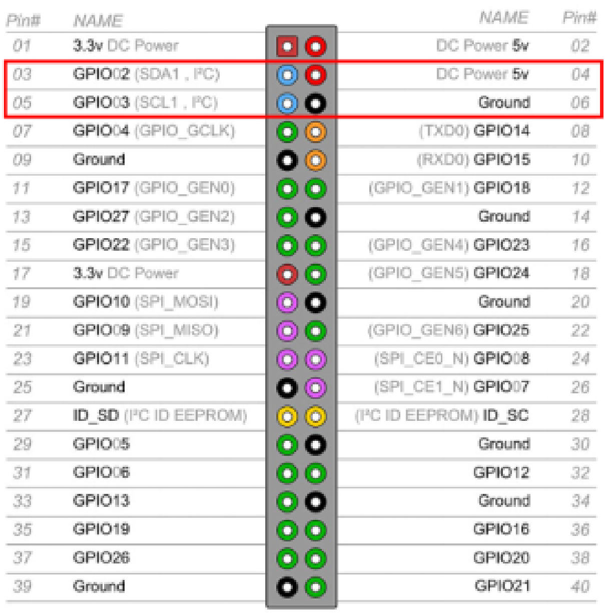
7.I2C communication

Note: This course requires the use of a 0.91-inch OLED display screen and DuPont cable as materials. Please provide them in advance

The I2C pin of Raspberry Pi is shown in the figure, and the I2C service needs to be enabled before use.





Pin#	NAME	NAME	Pin#
01	3.3v DC Power	DC Power 5v	02
03	GPIO02 (SDA1 , I^{2}C)	DC Power 5v	04
05	GPIO03 (SCL1 , I^{2}C)	Ground	06
07	GPIO04 (GPIO_GCLK)	(TXD0) GPIO14	08
09	Ground	(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)	(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)	Ground	14
15	GPIO22 (GPIO_GEN3)	(GPIO_GEN4) GPIO23	16
17	3.3v DC Power	(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)	Ground	20
21	GPIO09 (SPI_MISO)	(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)	(SPI_CE0_N) GPIO08	24
25	Ground	(SPI_CE1_N) GPIO07	26
27	ID_SD (I^{2}C ID EEPROM)	(I^{2}C ID EEPROM) ID_SC	28
29	GPIO05	Ground	30
31	GPIO06	GPIO12	32
33	GPIO13	Ground	34
35	GPIO19	GPIO16	36
37	GPIO26	GPIO20	38
39	Ground	GPIO21	40

Install I2C tool, and input the terminal as follows:

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

Check the installation status, terminal input:

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

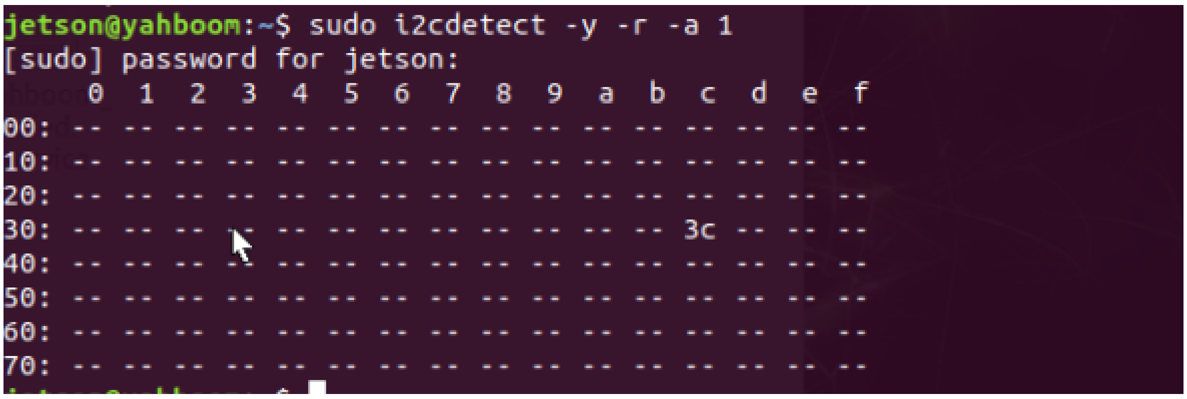
The following output indicates successful installation:

```
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 certain bus and print out the device i2c bus address.

For example, if a device with address 0x0f is mounted on the I2C pin here, the corresponding device I2C address will be displayed

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

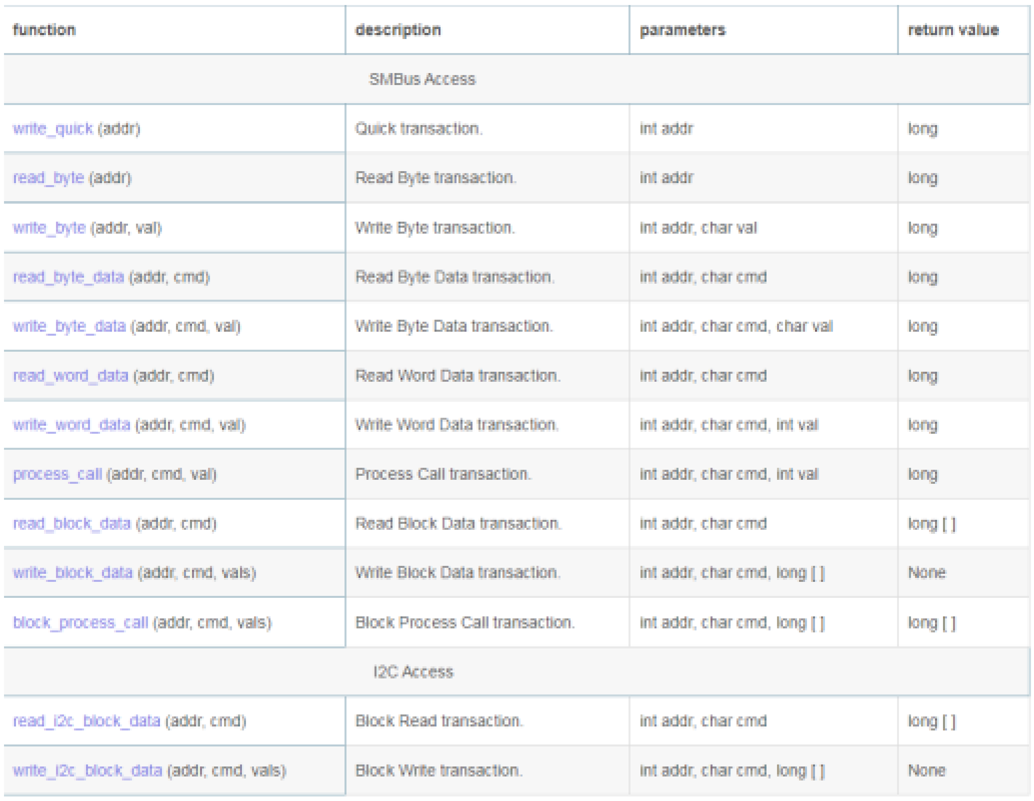


```
jetson@yahboom:~$ sudo i2cdetect -y -r -a 1
[sudo] password for jetson:
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- --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

Smbus is a Python library. If smbus is not installed, the terminal input is:

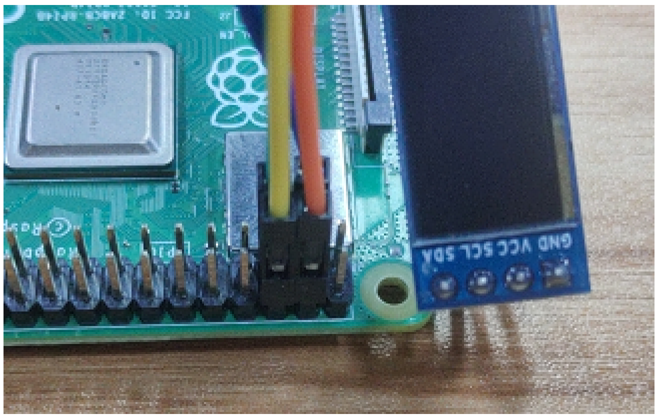
```
sudo apt-get update
sudo apt-get install -y python3-smbus
```

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:
Raspberry Pi Pin 3 (SDA) → OLED Module SDA
Raspberry Pi Pin 5 (SCL) → OLED Module SCL
Raspberry Pi Pin 4 (5V) → OLED Module VCC
Raspberry Pi Pin 6 (GND) → OLED Module GND



Import Adafruit_ SSD1306 library This is the OLED library, and you need to download this library when using your own image

```
pip3 install Adafruit_SSD1306
```

```python
#!/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:

```python
# 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
```

Input following command:

```
sudo python3 yahboom_oled.py
```

Experimental phenomenon: