



Version V1.5
By Gray
2021/5/27

Pi interface Hat Guide



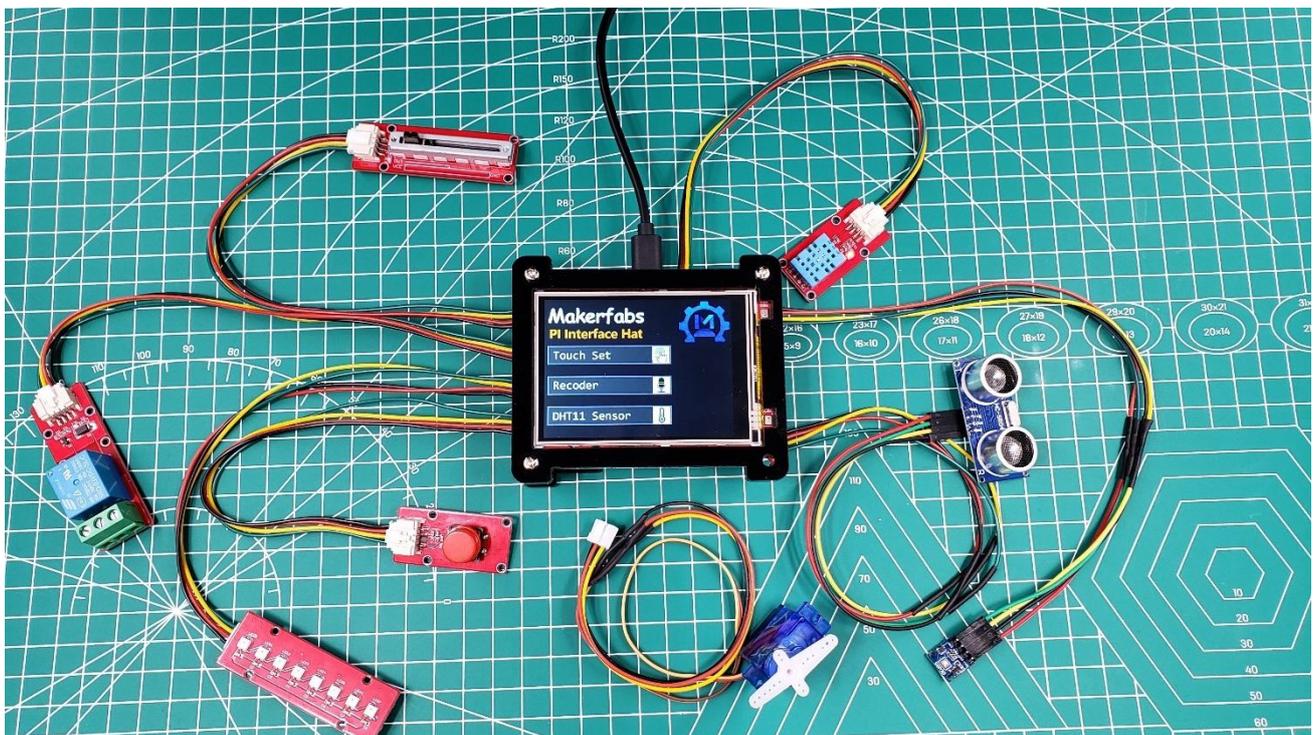
TABLE OF CONTENTS

1. Introduce	3
2. Features.....	4
3. Diagram.....	5
4. Basic usage.....	7
5. WiFi connection.....	9
6. Kill the process of default program.....	13
7. Using demo	14
7.1 Recode and Play	14
7.2 Display picture.....	16
7.3 Touch screen.....	18
7.4 Get the ADC value.....	19
7.5 Monitor the button status.....	21
7.6 Get the CO2 level.....	23
7.7 Use HC-SR04 to measure distance	24
7.8 Control the servo	26
7.9 Control the WS2812 LEDs.....	28
7.10 Connect to a HDMI Display to show the Linux desktop	30
8. Default program.....	31
9. Re-install Raspberry Pi OS.....	33

1. Introduce

Raspberry Pi Embedded System Development Kit (the below will use "Pi interface Hat" replace it) is a perfect product for learning or using raspberry. Base on raspberry pi zero, it has a 320*240 LCD with touch for display, integrates a chip to process audio playing& recording, it also has been equipped with a speaker, a 3.5mm audio jack and two mics, and some common add-on functions for Pi such as the A/D and GPIO expansions.

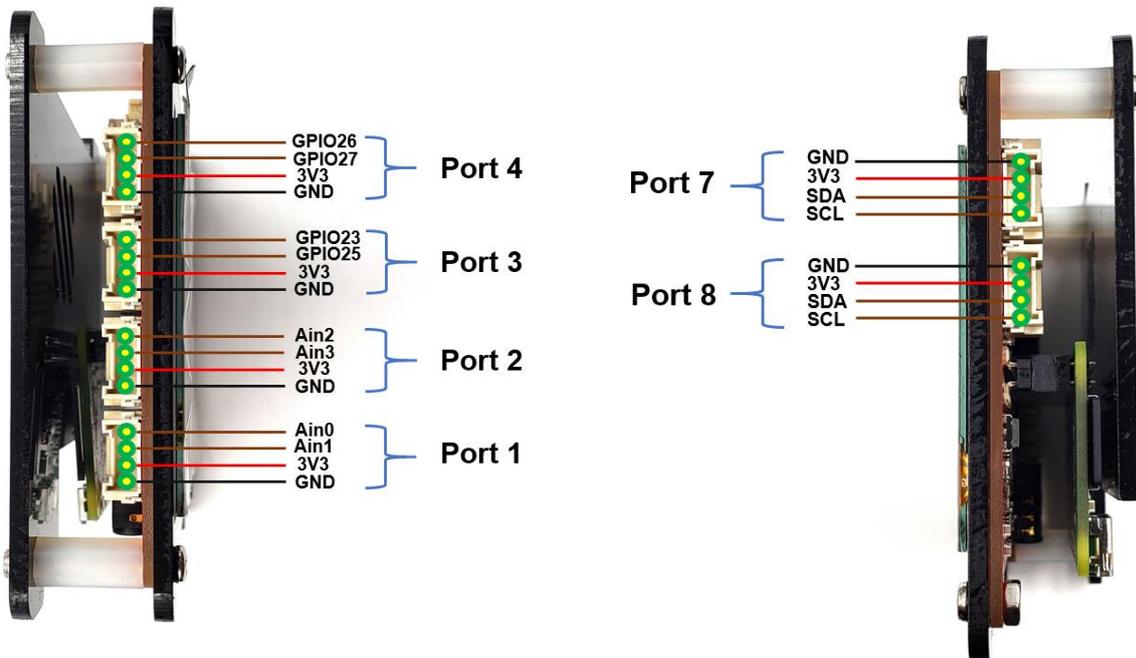
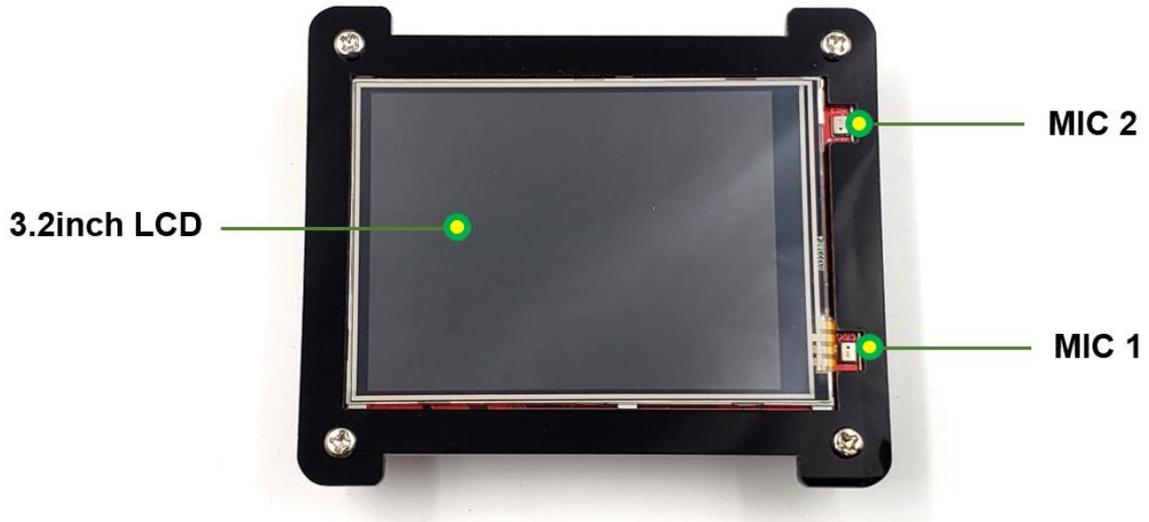
As one of the most successful open hardware platform, Raspberry Pi has been widely used in plenty of applications. This Pi interface Hat could be very suitable to learn Raspberry Pi for beginner, with Makerfabs provided many demos for raspberry to display, play, record and so on. These demos are available for you to study easily. It uses a SPI display (but not a HDMI display) and hardware expansions such as Mic/Audio, to make it more suitable for embedded projects.

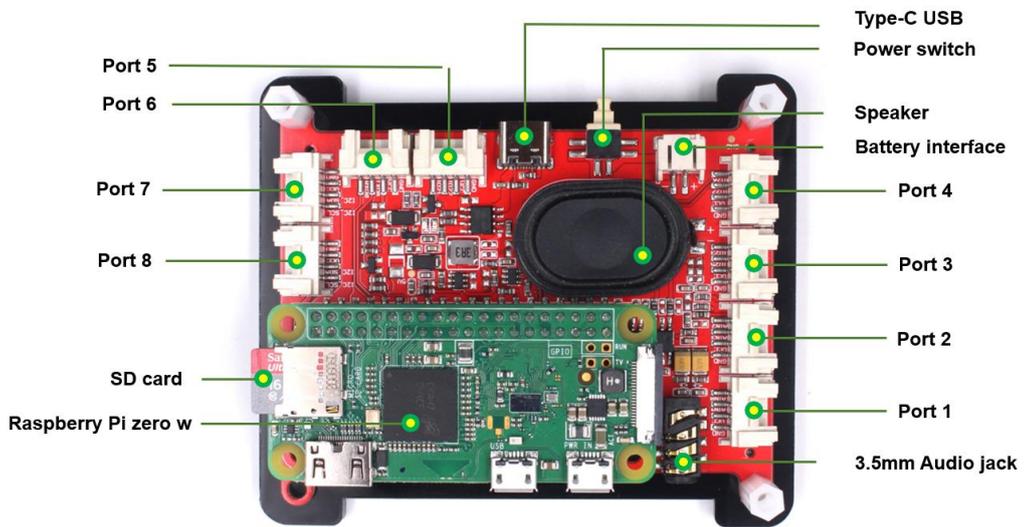
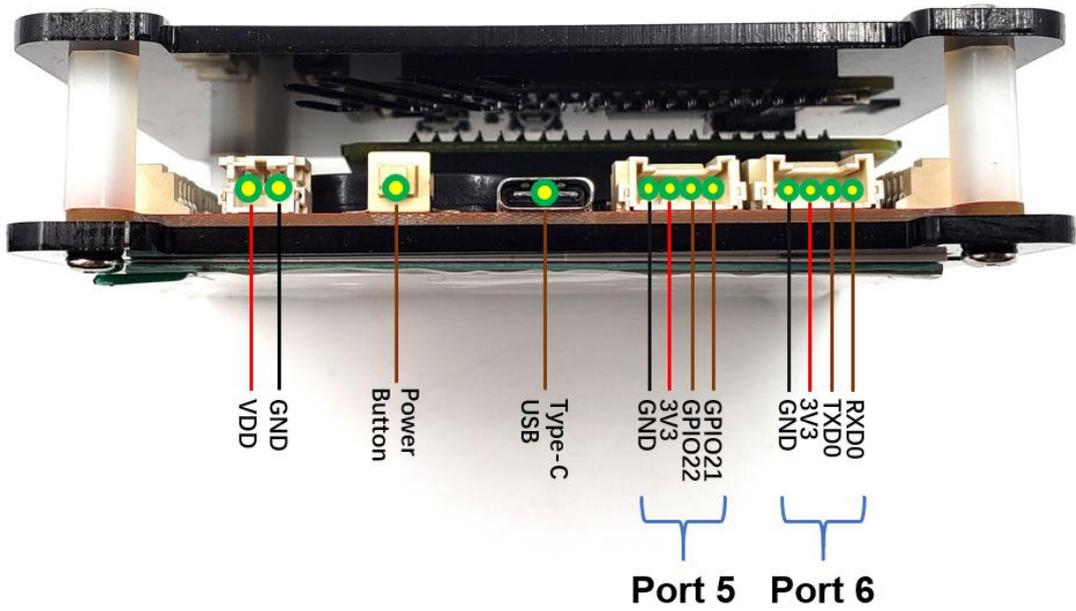


2. Features

- Raspberry pi zero W, 1GHz single-core CPU, 512MB RAM, Micro USB OTG port, Mini HDMI port, CSI camera connector, wireless LAN and Bluetooth.
- 3.2inch display, 320x240, ili9341 driver with SPI
- Resistive touch screen, XPT2046 controller
- Speaker
- 3.5mm audio jack
- Stereo Codec with Class D Speaker Driver: WM8960
- MEMS Mic*2: AOS3729A-T42-NXC
- Hardware expandable: I2C port, GPIO port, UART port
- ADC port: ADS1115
- Type-C USB power or battery power
- Support chargeable and 1A Maximum charging current
- Overcharge and over discharge protection
- Size: 94mm*80mm*24mm

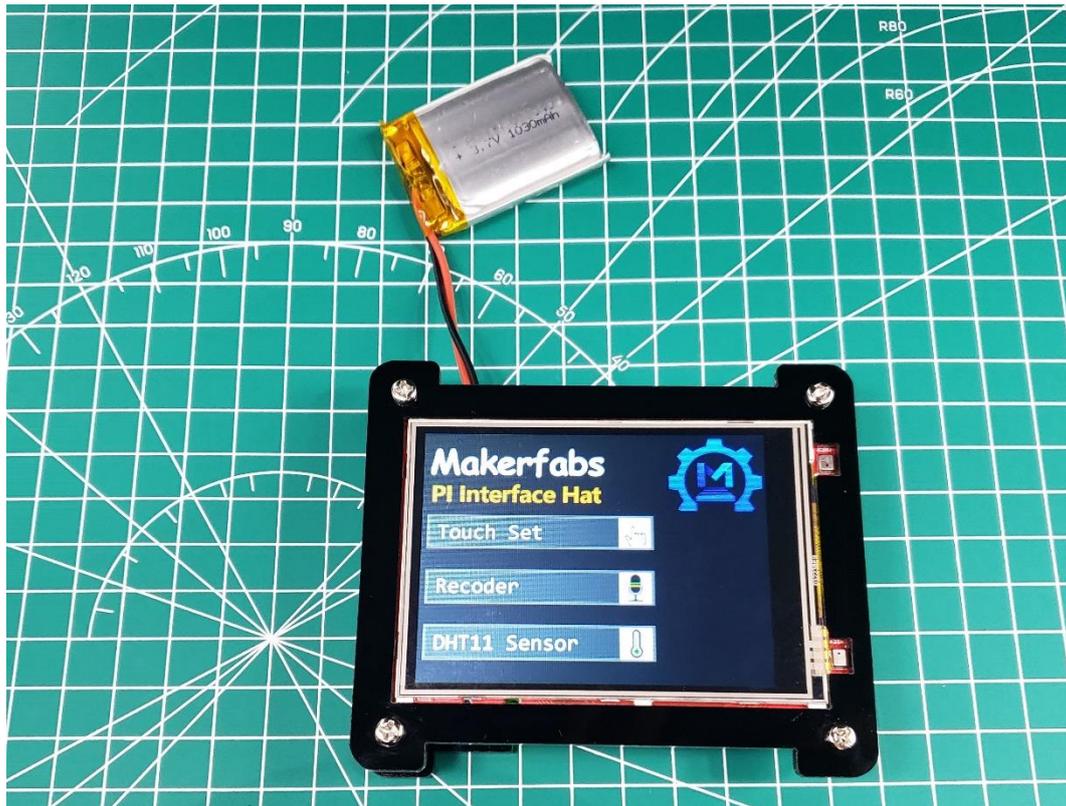
3. Diagram





4. Basic usage

Start by opening the package of the product, press the button, and Pi interface Hat can be used normally. As the system power up (Pi Zero required about one-minute to starting), it will show a short animation with audio. Next, there are three options that will be shown and can be pressed to run the different programs respectively, then you can try some of the basic demos: testing screen, recording audio, measuring temperature.



- **Touch Set**

Click the black pixel one by one on the screen with your finger or touch screen pen, it is to calibrate the touch accuracy. The screen will flash new page that you can touch or draw at will. Press the button at the top right to back to option menu.



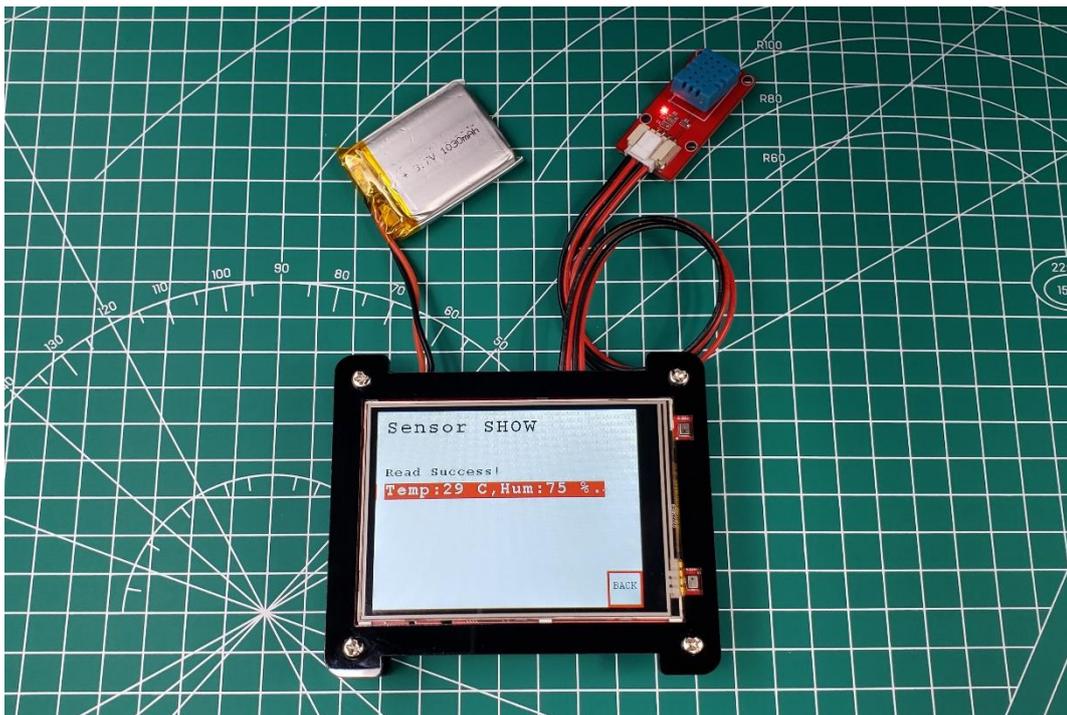
- **Record audio**

Click the recording button at the left to start record audio with a length of 5 seconds. Click the play button at the right, the speaker will play the audio recorded.



- **Measure temperature**

Connect the Mabee_DHT11 module to the product Port 5 (GPIO 22), touch the measuring temperature button. The temperature and humidity will be display on the screen after a moment.



5. WiFi connection

Before starting, you have to connect the raspberry to your WiFi.

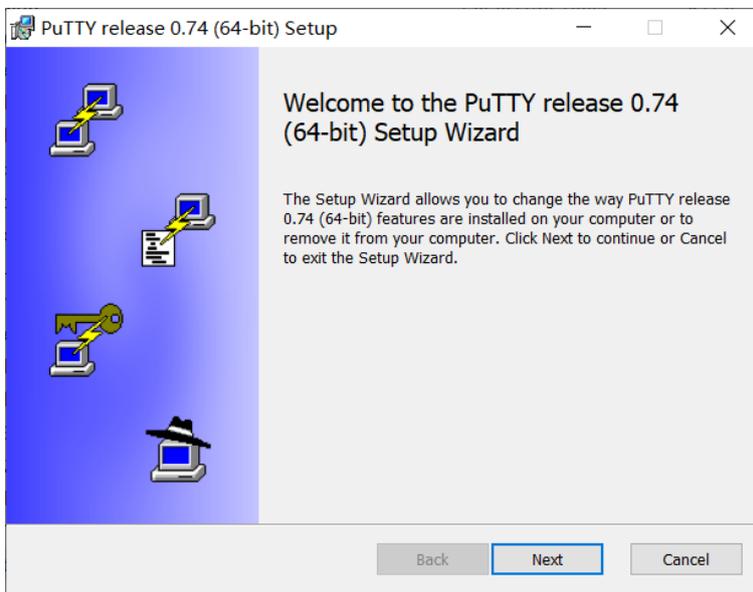
1. Prepare a Software

- SSH protocol

SSH or Secure Shell is a cryptographic network protocol for operating network services securely over an unsecured network. Typical applications include remote command-line, login, and remote command execution, but any network service can be secured with SSH.

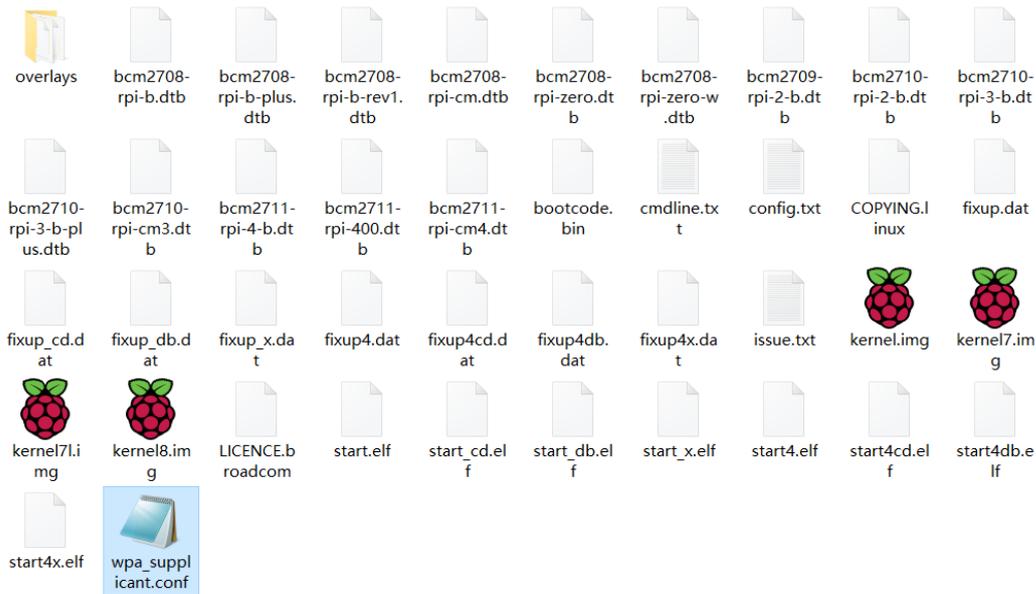
- PuTTY tool

PuTTY is an SSH client, usually used to remotely control the Raspberry Pi. You can get the install-package from here: <https://www.putty.org/>.



2. How to connect

- Remove the SD card from the raspberry pi and insert the SD card into the PC;
- Enter the SD card root, you will see some files in the root;
- Create a new file to this root that name **“wpa_supplicant.conf”**.



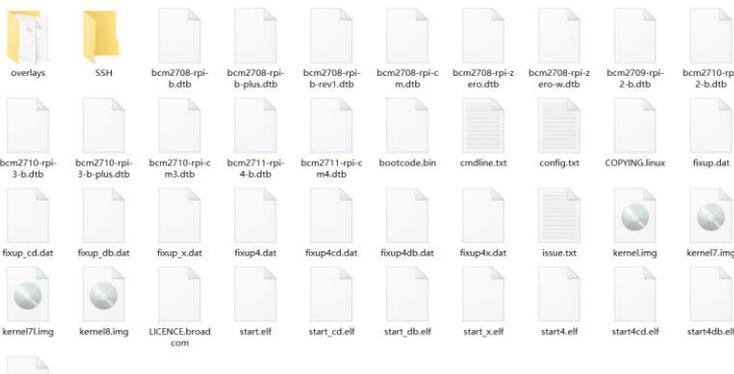
- Open the new file **wpa_supplicant.conf** with text, and copy the following code to it. Fill in your WiFi SSID and password into the code instead of “****”. Then save the file and exit.

```
Country=CN

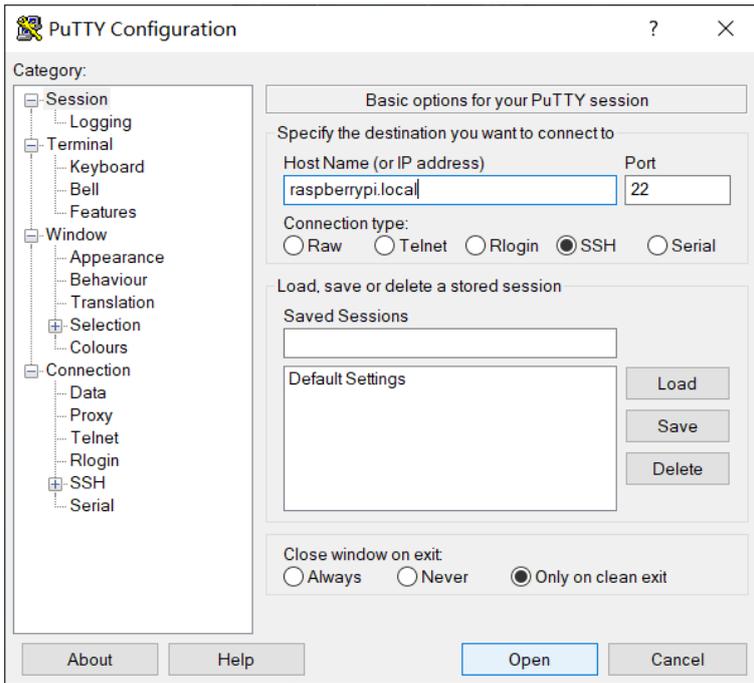
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1

network={
    ssid="*****"
    psk="*****"
    key_mgmt=WPA-PSK
    priority=1
}
```

- Set up a folder called SSH in the root, which is to turn on the Raspberry Pi SSH service.



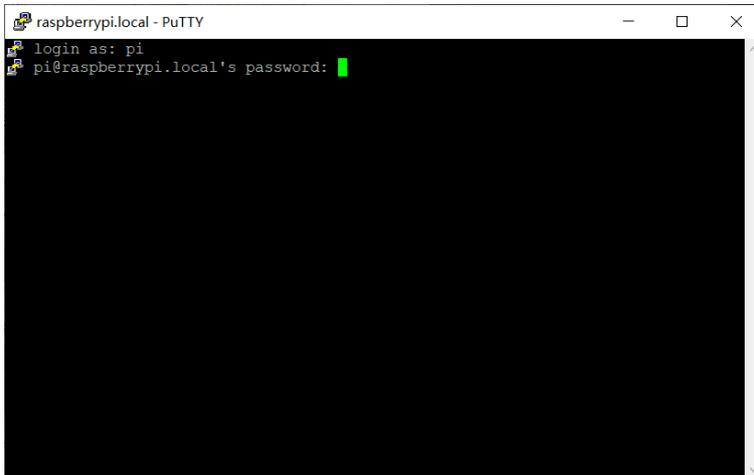
- Plug the SD card back to the Raspberry Pi, power on the **Pi Interface Hat** again. Raspberry Pi will automatically connect to WIFI.
- Make sure your computer and Raspberry Pi are on the same network. Open the Putty, enter “raspberrypi.local” in the address box and click “open”.



- Input the Raspberry Pi Username and password step by step.

Raspberry Pi Username: pi

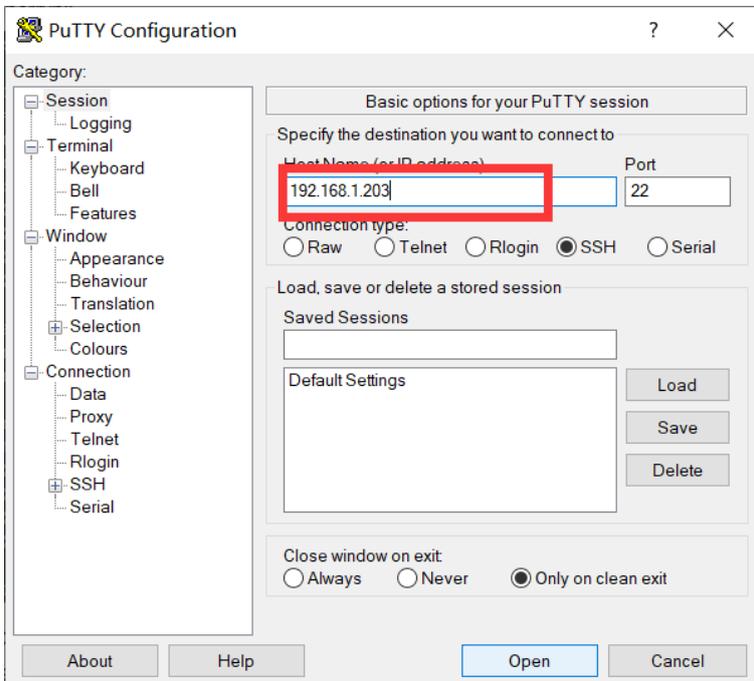
The **initial password** for Raspberry Pi is: raspberry. Note it will not be displayed when entering password and you have to complete it in one time.



- Now input the command "ifconfig" in the terminal to check the Raspberry Pi IP.

```
pi@raspberrypi: ~  
a new password.  
pi@raspberrypi:~$ ifconfig  
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536  
    inet 127.0.0.1 netmask 255.0.0.0  
    inet6 ::1 prefixlen 128 scopeid 0x10<host>  
    loop txqueuelen 1000 (Local Loopback)  
    RX packets 33 bytes 3164 (3.0 KiB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 33 bytes 3164 (3.0 KiB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 192.168.1.203 netmask 255.255.255.0 broadcast 192.168.1.255  
    inet6 ...  
<global>  
    inet6 ... prefixlen 64 scopeid 0x20<link>  
    ether ... txqueuelen 1000 (Ethernet)  
    RX packets 130 bytes 15320 (14.9 KiB)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 288 bytes 43256 (42.2 KiB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
pi@raspberrypi:~$
```

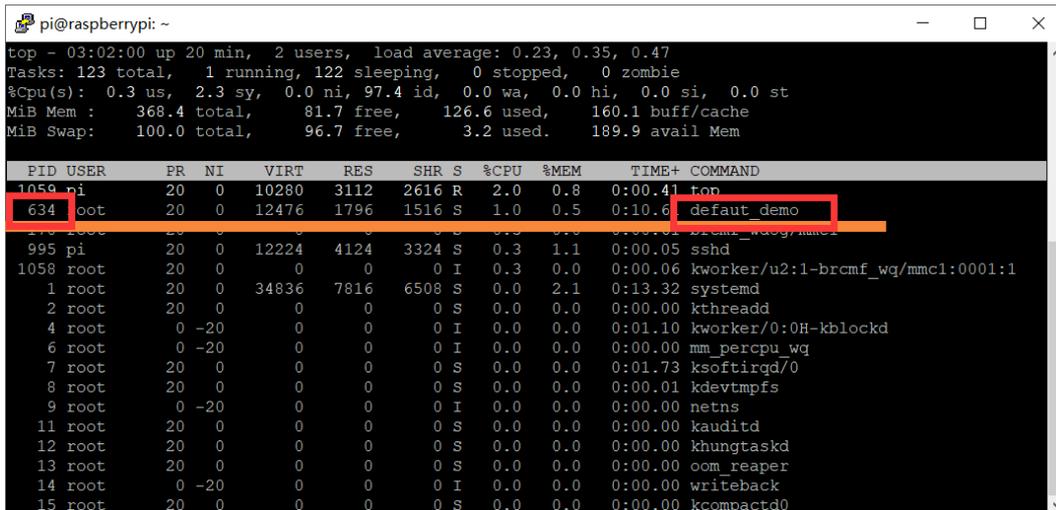
- The WiFi connection is complete.
- You can use the IP to enter Putty terminal next time, as follow.



6. Kill the process of default program.

Because of the default program (the 3 demos) started automatically, you have to kill the process to ensure the Pi not in busy status, follow the steps to kill:

- Enter the “top” command in the terminal, find the id of the default program. In the top table, check the PID number which the COMMAND is “default_demo”, and the number is the process ID.



```
pi@raspberrypi: ~  
top - 03:02:00 up 20 min, 2 users, load average: 0.23, 0.35, 0.47  
Tasks: 123 total, 1 running, 122 sleeping, 0 stopped, 0 zombie  
%Cpu(s): 0.3 us, 2.3 sy, 0.0 ni, 97.4 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st  
MiB Mem : 368.4 total, 81.7 free, 126.6 used, 160.1 buff/cache  
MiB Swap: 100.0 total, 96.7 free, 3.2 used, 189.9 avail Mem  


| PID  | USER | PR | NI  | VIRT  | RES  | SHR  | S | %CPU | %MEM | TIME+   | COMMAND                           |
|------|------|----|-----|-------|------|------|---|------|------|---------|-----------------------------------|
| 1059 | pi   | 20 | 0   | 10280 | 3112 | 2616 | R | 2.0  | 0.8  | 0:00.41 | top                               |
| 634  | root | 20 | 0   | 12476 | 1796 | 1516 | S | 1.0  | 0.5  | 0:10.6  | default_demo                      |
| 995  | pi   | 20 | 0   | 12224 | 4124 | 3324 | S | 0.3  | 1.1  | 0:00.05 | sshd                              |
| 1058 | root | 20 | 0   | 0     | 0    | 0    | I | 0.3  | 0.0  | 0:00.06 | kworker/u2:1-brcmf_wq/mmc1:0001:1 |
| 1    | root | 20 | 0   | 34836 | 7816 | 6508 | S | 0.0  | 2.1  | 0:13.32 | systemd                           |
| 2    | root | 20 | 0   | 0     | 0    | 0    | S | 0.0  | 0.0  | 0:00.00 | kthreadd                          |
| 4    | root | 0  | -20 | 0     | 0    | 0    | I | 0.0  | 0.0  | 0:01.10 | kworker/0:0H-kblockd              |
| 6    | root | 0  | -20 | 0     | 0    | 0    | I | 0.0  | 0.0  | 0:00.00 | mm_percpu_wq                      |
| 7    | root | 20 | 0   | 0     | 0    | 0    | S | 0.0  | 0.0  | 0:01.73 | ksoftirqd/0                       |
| 8    | root | 20 | 0   | 0     | 0    | 0    | S | 0.0  | 0.0  | 0:00.01 | kdevtmpfs                         |
| 9    | root | 0  | -20 | 0     | 0    | 0    | I | 0.0  | 0.0  | 0:00.00 | netns                             |
| 11   | root | 20 | 0   | 0     | 0    | 0    | S | 0.0  | 0.0  | 0:00.00 | kauditd                           |
| 12   | root | 20 | 0   | 0     | 0    | 0    | S | 0.0  | 0.0  | 0:00.00 | khungtaskd                        |
| 13   | root | 20 | 0   | 0     | 0    | 0    | S | 0.0  | 0.0  | 0:00.00 | oom_reaper                        |
| 14   | root | 0  | -20 | 0     | 0    | 0    | I | 0.0  | 0.0  | 0:00.00 | writeback                         |
| 15   | root | 20 | 0   | 0     | 0    | 0    | S | 0.0  | 0.0  | 0:00.00 | kcompactd0                        |


```

- Click “Ctrl+C” to exit the top table.
- Enter the below command to kill the process.

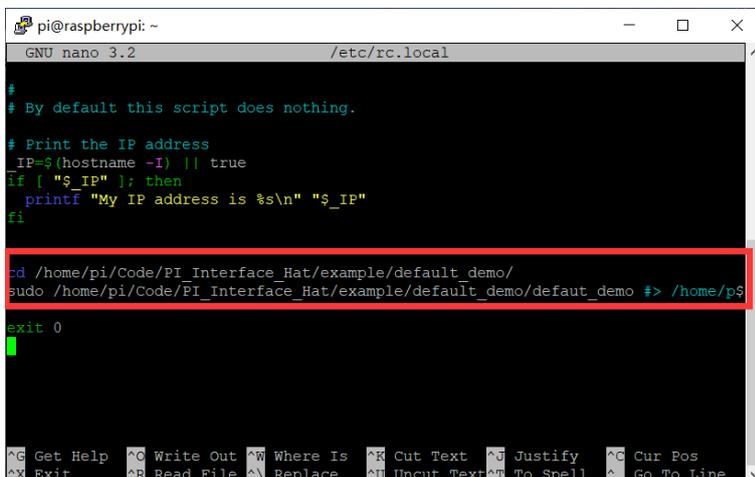
```
Sudo kill 634
```

- *(Optional)* If you do not want the default program started automatically, follow the steps to cancel the auto-run of the default program:

- Open the file executed when Raspberry pi is powered on.

```
Sudo nano /etc/rc.local
```

- Comment out the follow code by adding “#” before it.



```
pi@raspberrypi: ~  
GNU nano 3.2 /etc/rc.local  
#  
# By default this script does nothing.  
  
# Print the IP address  
_IP=$(hostname -I) || true  
if [ "$_IP" ]; then  
  printf "My IP address is %s\n" "$_IP"  
fi  
  
cd /home/pi/Code/PI_Interface_Hat/example/default_demo/  
sudo /home/pi/Code/PI_Interface_Hat/example/default_demo/default_demo &#gt; /home/p$  
  
exit 0  
#
```

```
GNU nano 3.2 /etc/rc.local
#
# By default this script does nothing.
# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
  printf "My IP address is %s\n" "$_IP"
fi

cd /home/pi/Code/PI_Interface_Hat/example/default_demo/
sudo /home/pi/Code/PI_Interface_Hat/example/default_demo/default_demo #> /home/$
exit 0

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To Line
```

- Click “Ctrl+o” and Enter to save, then click “Ctrl+x” to exit, the default program will not run when the Raspberry is powered on next time.

Note: if you see some steps with the red “Optional” and in italics, it means these steps can be ignore, if you use the Makerfabs Pi Hat Default system in the SD card.

7. Using demo

For learning further, Makerfabs provided multiple examples to show how to use. In many examples, additional programs need to be used instead of just Raspberry commands. The program that used to drive various components is available in Github. The Github link is:

https://github.com/Makerfabs/PI_Interface_Hat.git.

For convenience, we have loaded the project to the Raspberry, and the next chapter will show how to use them. The project covers all the programs for the demo, and it is necessary to follow the step to use the program for success.

(Optional) If you want to obtain the last version of the project, please use the following command to load it.

```
Git clone https://github.com/Makerfabs/PI_Interface_Hat.git
```

7.1 Recode and Play

1. Follow the steps to install the Linux kernel drivers for recording and playing (NOTE: If the Raspberry Pi OS is provided by Makerfabs, please ignore this step that the driver has been installed):
 - *(Optional) Enter the following command in the Raspberry Pi terminal window to install the driver:*

```
git clone https://github.com/respeaker/seeed-voicecard
cd seeed-voicecard
sudo ./install.sh
sudo reboot
```


“-Dhw:1,0” means that sound card 1 is used for recording, “3” means that the recording time is 3s, and “temp.wav” is the file name which saved for recording.

4. Use the following command to play audio.

```
aplay -Dhw:1,0 temp.wav
```

“-Dhw:1,0” means that sound card 1 will be used for output, and “temp.wav” is the name of the output file.

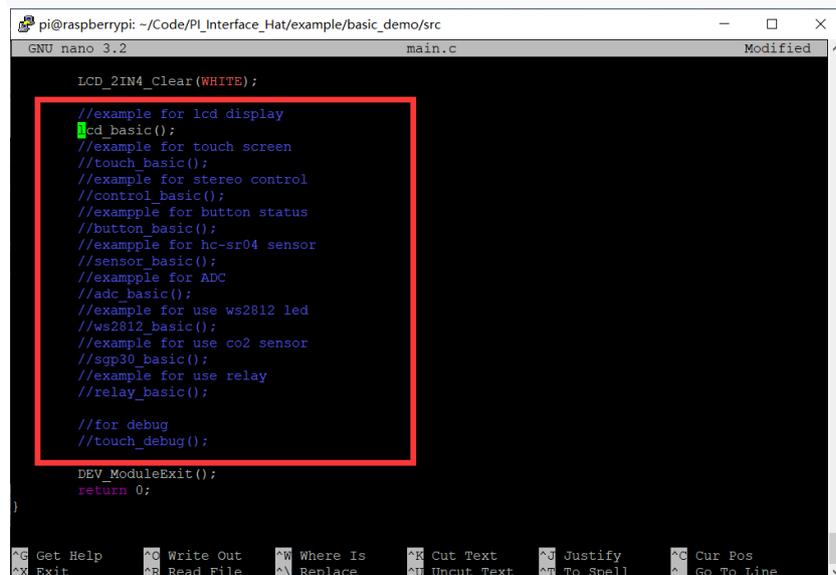
Note1: the sound card can only support audio WAV-formatted and two-channel.

Note2: the volume is recommended to adjust to 80%, too strong is bad for playing. When it power by battery, sometime the unsteady voltage will cause the speaker working not well.

7.2 Display picture.

1. 3.2 inch LCD with driver ili9341 is integrated on the board, and it is connected to Raspberry Pi zero with SPI, but not using the HDMI interface.
2. Modify the code main.c. uncomment the “lcd_basic()” function by deleting “//” before it, and comment out the other function by adding “//” before it,so there will be only the *LCD_basic()*; runs:

```
sudo nano main.c
```



```
pi@raspberrypi: ~/Code/PI_Interface_Hat/example/basic_demo/src
GNU nano 3.2 main.c Modified
LCD_2IN4_clear(WHITE);

//example for lcd display
lcd_basic();
//example for touch screen
//touch_basic();
//example for stereo control
//control_basic();
//example for button status
//button_basic();
//example for hc-sr04 sensor
//sensor_basic();
//example for ADC
//adc_basic();
//example for use ws2812 led
//ws2812_basic();
//example for use co2 sensor
//spp30_basic();
//example for use relay
//relay_basic();

//for debug
//touch_debug();

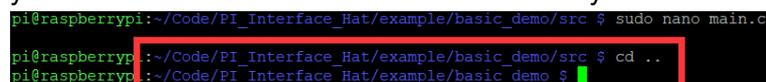
DEV_ModuleExit();
return 0;
```

Click “Ctrl+o” and Enter to save, then click “Ctrl+x” to exit.

3. Return directory to “/PI_Interface_Hat/example/basic_demo”.

```
Cd ..
```

you can see the current location of the directory in the terminal.



```
pi@raspberrypi:~/Code/PI_Interface_Hat/example/basic_demo/src $ sudo nano main.c
pi@raspberrypi:~/Code/PI_Interface_Hat/example/basic_demo/src $ cd ..
pi@raspberrypi:~/Code/PI_Interface_Hat/example/basic_demo $
```

4. Run the command to compile program.

```
make
```

- Print a list of all files in the file directory, check whether the executable file is added. Then execute it.

```
Ls  
sudo ./basic_demo
```

```
pi@raspberrypi:~/Downloads/PI_Interface_Hat/example/basic_demo $ ls  
basic_demo bin lib Makefile pic src  
pi@raspberrypi:~/Downloads/PI_Interface_Hat/example/basic_demo $ sudo ./basic_demo
```

- The picture will be displayed on the screen.



- (Optional)** If you want to show some picture yours, follow the steps to modify the code and execute the code main.c.

a. Copy the picture to the expected file directory “/Code/PI_Interface_Hat/example/basic_demo/pic/”, it is notice that the size of the picture is suitable for 240*320 LCD and the format is BMP.

b. Modify the code “PI_Interface_Hat/example/basic_demo/src/lcd_basic.c”, that replace ./pic/LCD_2inch4.bmp (the address of the previous picture) with the Absolute address of displayed picture. The LCD_2inch4.bmp is the testing picture pre-set, it is a bird picture you saw.

```
Cd ~/Code/PI_Interface_Hat/example/basic_demo/src  
sudo nano lcd_basic.c
```

```
pi@raspberrypi: ~/Downloads/PI_Interface_Hat/example/basic_demo/src  
GNU nano 3.2 lcd_basic.c  
LCD_2IN4_Display((UBYTE *)BlackImage);  
DEV_Delay_ms(1000);  
Paint_SetRotate(ROT90);  
GUI_ReadBmp("./pic/LCD_2inch4.bmp");  
LCD_2IN4_Display((UBYTE *)BlackImage);  
DEV_Delay_ms(1000);  
Paint_SetRotate(ROT270);  
GUI_ReadBmp("./pic/LCD_2inch4.bmp");  
LCD_2IN4_Display((UBYTE *)BlackImage);  
DEV_Delay_ms(1000);  
/* Module Exit */  
free(BlackImage);  
BlackImage = NULL;
```

```
pi@raspberrypi: ~/Code/PI_Interface_Hat/example/basic_demo/src
GNU nano 3.2 lcd_basic.c

printf("show bmp\r\n");

Paint_SetRotate(ROTATE_90);
GUI_ReadBmp("./pic/test1.bmp");
LCD_2IN4_Display((UBYTE *)BlackImage);

DEV_Delay_ms(1000);

/*
Paint_SetRotate(ROTATE_90);
GUI_ReadBmp("./pic/LCD_2inch4.bmp");
LCD_2IN4_Display((UBYTE *)BlackImage);

DEV_Delay_ms(1000);
*/

Paint_SetRotate(ROTATE_90);
GUI_ReadBmp("./pic/test1.bmp");
LCD_2IN4_Display((UBYTE *)BlackImage);

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To Line
```

Click "Ctrl+o" to save and "Ctrl+x" to exit.

c. Follow the previous steps to modify the main.c and execute it.

7.3 Touch screen.

The LCD also has a touch panel with XPT2046. As the previous steps, change the directory to "~/Code/PI_Interface_Hat/example/basic_demo/src", and modify the main.c to correct the function of the program. Uncomment "touch_basic()" as the figure shows.

```
cd ~/Code/PI_Interface_Hat/example/basic_demo/src
sudo nano main.c
```

```
pi@raspberrypi: ~/Code/PI_Interface_Hat/example/basic_demo/src
GNU nano 3.2 main.c Modified

LCD_2IN4_Clear(WHITE);

//example for lcd display
//lcd_basic();
//example for touch screen
touch_basic();
//example for stereo control
//control_basic();
//example for button status
//button_basic();
//example for hc-sr04 sensor
//sensor_basic();
//example for ADC
//adc_basic();
//example for use ws2812 led
//ws2812_basic();
//example for use co2 sensor
//sgp30_basic();
//example for use relay
//relay_basic();

//for debug
//touch_debug();

DEV_ModuleExit();
return 0;
}
```

1. Change directory to "~/Code/PI_Interface_Hat/example/basic_demo", Run the command to compile program.

```
cd ..
make
```

- Print a list of all files in the file directory, check whether the executable file is added and execute it.

```
ls
sudo ./basic_demo
```

- The touch position will be display in the terminal of SSH when touch screen.



```
pi@raspberrypi: ~/Code/PI_Interface_Hat/example/ba
Debug : 2inch4 LCD Init...
Debug : Draw Touch Setting...
Debug : Init xpt2046...
Debug : xpt_x_unit:113
Debug : xpt_y_unit:84
Debug : xpt_x_start:2254
Debug : xpt_y_start:2104
Debug : Please press...
Debug : Get press...1
Debug : x,y:153,320
Debug : pos1,pos2:0,32760
Debug : Get press...2
Debug : x,y:129,165
Debug : pos1,pos2:16640,15904
Debug : Get press...3
Debug : x,y:65,230
Debug : pos1,pos2:9640,21760
Debug : Get press...4
Debug : x,y:155,242
Debug : pos1,pos2:19872,22192
Debug : Get press...5
```

7.4 Get the ADC value.

As you know, Raspberry Pi did not provide the ADC ability. Due to the Pi Interface Hat that had integrated ADS1115 chip, it has four channels ADC which the interfaces are Port 1 and Port 2. Blows we will have a try to A/D the input voltage signal .

- Connect the Potentiometer to the port1.



- As the above steps to modify and execute the code "main.c". Uncomment the specified function in the main.c.

```
//example for ADC  
adc_basic();
```

```
pi@raspberrypi: ~/Code/PI_Interface_Hat/example/basic_demo/src  
GNU nano 3.2 main.c Modified  
LCD_2IN4_Clear(WHITE);  
  
//example for lcd display  
//lcd_basic();  
//example for touch screen  
//touch_basic();  
//example for stereo control  
//control_basic();  
//example for button status  
//button_basic();  
//example for hc-sr04 sensor  
//sensor_basic();  
//example for ADC  
adc_basic();  
//example for use ws2812 led  
//ws2812_basic();  
//example for use co2 sensor  
//sgp30_basic();  
//example for use relay  
//relay_basic();  
  
//for debug  
//touch_debug();  
  
DEV_ModuleExit();  
return 0;
```

3. Follow the below step to execute the “make” command for verifying the program and run it.

```
cd ~/Code/PI_Interface_Hat/example/basic_demo  
make  
sudo ./basic_demo
```

4. The ADC value would be obtained and shown on the LCD.



- The detail of the code for getting the value from the ADC in the /PI_Interface_Hat/example/basic_demo/src/adc_basic.c.

```
int handle = wiringPiI2CSetup(ADS1115_ADDRESS);  
while (1)
```

```

{
    float volts[4];
    char value_str[80];
    for (int i = 0; i < 4; ++i)
    {
        float v = readVoltage(handle, i, 0);
        if (v > 6)
        {
            v = 0;
        }
        volts[i] = v;
    }
    sprintf(value_str, "%.2lf %.2lf %.2lf %.2lf", volts[0], volts[1], volts[2], volts[3]);
    Paint_Clear(WHITE);
    Paint_DrawString_EN(10, 110, value_str, &Font16, WHITE, BLACK);
    LCD_2IN4_img(10, 110, 240, 130, (UBYTE *)BlackImage);
}

```

- Display the value on the LCD.

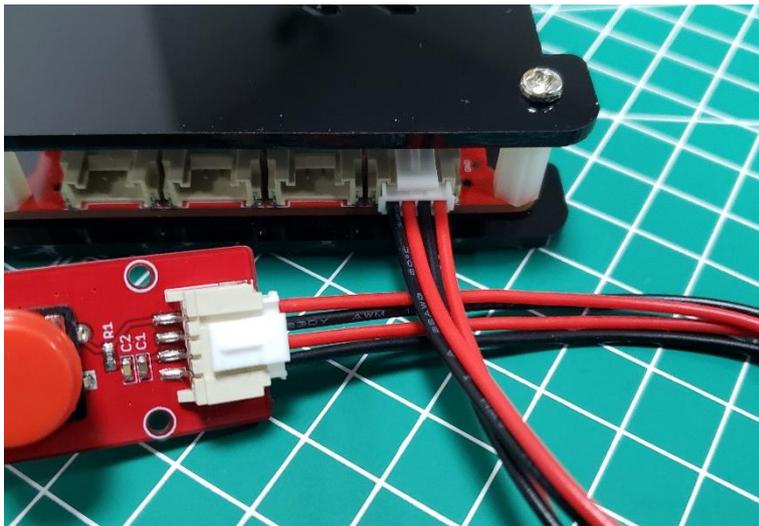
```

Paint_Clear(WHITE);
Paint_DrawString_EN(10, 110, value_str, &Font16, WHITE, BLACK);

```

7.5 Monitor the button status.

1. Connect the BOTTON to the Port 4, that the signal wire is connected to GPIO26.



2. As the above steps to modify and execute "main.c", uncomment the content in the main.c

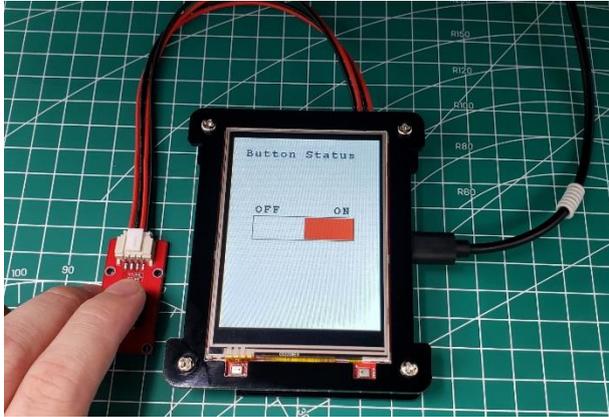
```

//example for button status
button_basic();

```

3. Follow the below commands to run “make” step by step.

```
cd ~/Code/PI_Interface_Hat/example/basic_demo
make
sudo ./basic_demo
```



4. The demo code which read the value of button is `/PI_Interface_Hat/example/basic_demo/src/button_basic.c`.

- Get the value of the button:

```
int button_pin = 12;    //GPIO 26
DEV_GPIO_Mode(button_pin, INPUT);
```

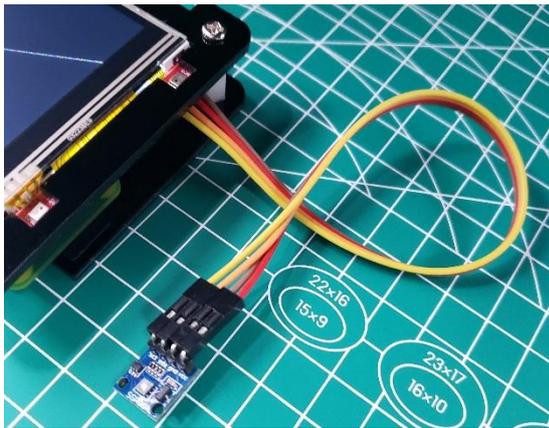
- Display the button status on the LCD.

```
if (DEV_Digital_Read(button_pin) == LOW)
{
    LCD_2IN4_FillRect(121, 141, 199, 178, WHITE);
    LCD_2IN4_FillRect(41, 141, 120, 178, BLACK);
}
else
{
    LCD_2IN4_FillRect(41, 141, 120, 178, WHITE);
    LCD_2IN4_FillRect(121, 141, 199, 178, RED);
}
DEV_Delay_ms(100);
```

7.6 Get the CO2 level.

1. Pi Interface Hat provides two I2C interfaces which are Port 7 and Port 8. Connect the SGP30 sensor to the Port 7, the connection of the sensor is shown in following figure.

SGP30 sensor	Raspberry
VCC	VCC
GND	GND
SCL	SCL
SDA	SDA



2. *(Optional)* You have to run the command to install the SGP30 library for driving. Again, if you use Makerfabs default OS in the Hat SD card, you do not need to do this as it already installed.

```
sudo pip3 install pimoroni-sgp30
```

3. *(Optional)* Install the package to support SGP30 library.

```
sudo pip3 install smbus2
```

4. As the above steps to modify the code "main.c". Uncomment the content of the SGP30 in the main.c.

```
//example for use co2 sensor  
sgp30_basic();
```

5. Follow the below steps to run the "make" command.

```
cd ..  
make  
sudo ./basic_demo
```

6. The value would be shown in the raspberry terminal.

```
Air Quality:
Equivalent CO2: 484 (ppm)
Total VOC: 9 (ppb)

Air Quality:
Equivalent CO2: 492 (ppm)
Total VOC: 12 (ppb)

Air Quality:
Equivalent CO2: 501 (ppm)
Total VOC: 13 (ppb)

Air Quality:
Equivalent CO2: 504 (ppm)
Total VOC: 11 (ppb)

Air Quality:
Equivalent CO2: 452 (ppm)
Total VOC: 0 (ppb)
```

7. The demo code for SGP30 to monitor is /PI_Interface_Hat/example/basic_demo/py/sgp30_test.py and different from others. It was done in Python and would be run by Linux command.

- Run the Linux command in /PI_Interface_Hat/example/basic_demo/src/sgp30_basic.c.

```
system("sudo python3 ./py/sgp30_test.py");
```

- Initialize the SGP30 sensor in the python file /PI_Interface_Hat/example/basic_demo/py/sgp30_test.py.

```
sgp30 = SGP30()
sgp30.start_measurement(crude_progress_bar)
```

- Get the CO2 level.

```
result = sgp30.get_air_quality()
print(result)
time.sleep(1.0)
```

7.7 Use HC-SR04 to measure distance

1. Connect the HC-SR04 module to Port 4, the connection of the module is shown in following figure.

HC-SR04	Raspberry
VCC	VCC
GND	GND
ECHO	GPIO27
TRIG	GPIO26



2. As the above steps to modify and execute the code. Modify the code that uncomments special content in the main.c.

```
//example for hc-sr04 sensor
sensor_basic();
```

3. Follow the above tutorial to run the “make” command and execute the program.



4. The detail of measuring the distance in the `/PI_Interface_Hat/example/basic_demo/src/sensor_basic.c`

```
unsigned int begin_time = micros();
// Wait for ping response, or timeout.
while (digitalRead(ECHO) == LOW && micros() - begin_time < timeout)
{
}
// Cancel on timeout.
if (micros() - begin_time > timeout)
{
    printf("1.Out of range.\n");
    return 0;
}
ping = micros();
// Wait for pong response, or timeout.
while (digitalRead(ECHO) == HIGH && micros() - ping < timeout)
```

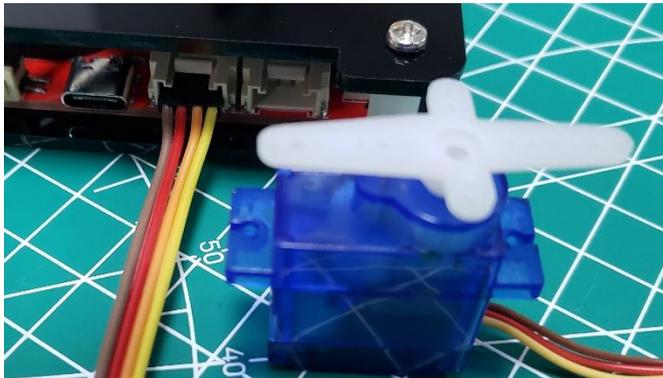
```

{
    //printf("ECHO HIGH\n");
}
// Cancel on timeout.
if (micros() - ping > timeout)
{
    printf("2.Out of range.\n");
    return 0;
}
pong = micros();
// Convert ping duration to distance.
distance = (int)((pong - ping) * 0.017150);
printf("Distance: %.d cm.\n", distance);

```

7.8 Control the servo

1. The mini servo has three pins, the red one is VCC, the brown is GND, and the yellow is signal. The servo can be controlled by PWM signal. Connect the signal pin of the servo to Ports 5(GPIO21 in hardware).



2. Uncomment the content in the main.c of the project as above instruction.

```

//example for stereo control
control_basic();

```

3. In order to drive the servo successfully, you have to check the signal pin used in the code “/src/control_basic.c”. Use the command to open the code “control_basic.c”.

```

sudo nano control_basic.c

```

Note to change the port code to 5, as it use the BCM2835 library which is a C library for Raspberry to provide access to GPIO and other IO functions on the Broadcom BCM 2835 chip, and the GPIO21 is coded to “5” in the BCM:

```

pi@raspberrypi:~/Code/PI_Interface_Hat/example/basic_demo $ gpio readall
-----Pi ZeroW-----
| BCM | wPi | Name | Mode | V | Physical | V | Mode | Name | wPi | BCM | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 8 | 3.3v | | | 1 | 2 | | | 5v | | |
| 3 | 8 | SDA.1 | ALTO | 1 | 3 | 4 | | | 5v | | |
| 4 | 9 | SCL.1 | ALTO | 1 | 5 | 6 | | | 0v | | |
| 4 | 7 | GPIO. 7 | IN | 1 | 7 | 8 | 0 | IN | TxD | 15 | 14 |
| | | 0v | | | 9 | 10 | 1 | IN | RxD | 16 | 15 |
| 17 | 0 | GPIO. 0 | IN | 0 | 11 | 12 | 0 | ALTO | GPIO. 1 | 1 | 18 |
| 27 | 2 | GPIO. 2 | OUT | 1 | 13 | 14 | | | 0v | | |
| 22 | 3 | GPIO. 3 | IN | 1 | 15 | 16 | 1 | OUT | GPIO. 4 | 4 | 23 |
| | | 3.3v | | | 17 | 18 | 1 | IN | GPIO. 5 | 5 | 24 |
| 10 | 12 | MOSI | ALTO | 0 | 19 | 20 | | | 0v | | |
| 9 | 13 | MISO | ALTO | 0 | 21 | 22 | 1 | OUT | GPIO. 6 | 6 | 25 |
| 11 | 14 | SCLK | ALTO | 0 | 23 | 24 | 1 | IN | CE0 | 10 | 8 |
| | | 0v | | | 25 | 26 | 1 | OUT | CE1 | 11 | 7 |
| 0 | 30 | SDA.0 | IN | 1 | 27 | 28 | 1 | IN | SCL.0 | 31 | 1 |
| 5 | 21 | GPIO.21 | IN | 1 | 29 | 30 | | | 0v | | |
| 6 | 22 | GPIO.22 | IN | 1 | 31 | 32 | 0 | IN | GPIO.26 | 26 | 12 |
| 13 | 23 | GPIO.23 | IN | 0 | 33 | 34 | | | 0v | | |
| 19 | 24 | GPIO.24 | ALTO | 0 | 35 | 36 | 0 | IN | GPIO.27 | 27 | 16 |
| 26 | 25 | GPIO.25 | IN | 0 | 37 | 38 | 0 | ALTO | GPIO.28 | 28 | 20 |
| | | 0v | | | 39 | 40 | 0 | ALTO | GPIO.29 | 29 | 21 |
-----Pi ZeroW-----
| BCM | wPi | Name | Mode | V | Physical | V | Mode | Name | wPi | BCM |

```

```

.....

servo_init(5);

servo_angle(5, 0);

.....

servo_angle(5, angle);

```

Click “Ctrl+o” and Enter to save then click “Ctrl+x” to exit.

- Follow the below steps to run the “make” command for verify the program.

```

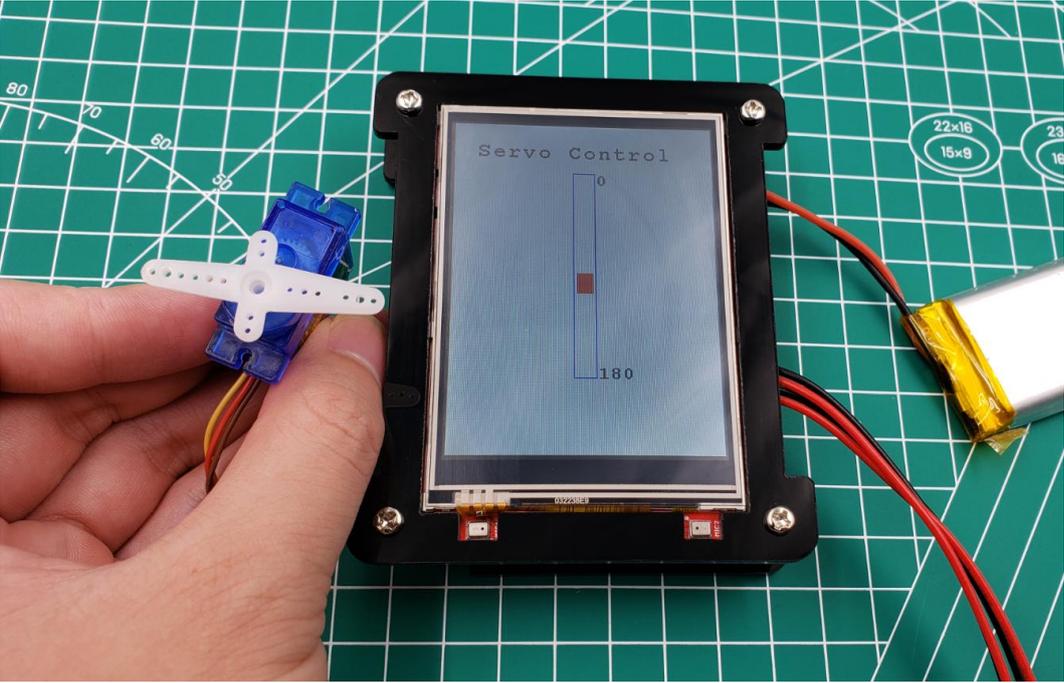
cd ~/Code/PI_Interface_Hat/example/basic_demo

make

sudo ./basic_demo

```

- The control button will be displayed on the LCD. Touch the screen to drive the servo running.



7.9 Control the WS2812 LEDs

1. Connect WS2812 LEDs to the Port 3. The signal pin which controlled the WS2812 LEDs is connected to IO23.



2. *(Optional)* There no library that support WS2812 and you have to run the command to install the WS2812 library.

```
sudo pip install rpi_ws281x
```

3. Uncomment the content in the main.c of the project.

```
//example for use ws2812 led  
ws2812_basic();
```

4. Open "ws2812_basic.c" by:

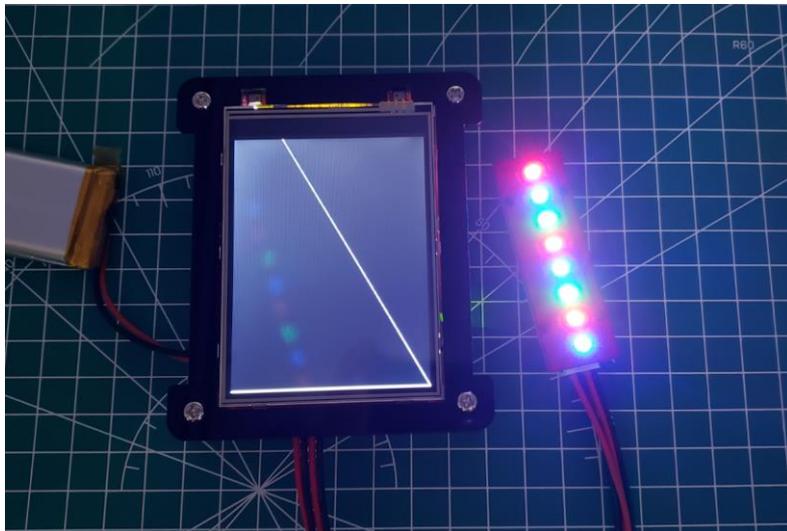
```
sudo nano ws2812_basic.c
```

Modify the code to "sudo python ./py/ws2812_test.py", the final such as the picture.

```
pi@raspberrypi: ~/Code/PI_Interface_Hat/example/basic_demo/src  
GNU nano 3.2 ws2812_basic.c  
#include "demo.h"  
#include <stdio.h> //printf()  
#include <stdlib.h> //exit()  
#include <signal.h> //signal()  
  
void ws2812_basic(void)  
{  
    //!!!First need run sudo pip2 install rpi_ws281x  
    system("sudo python ./py/ws2812_test.py");  
}
```

5. Follow the steps to verify the program and execute it.

```
cd ~/Code/ PI_Interface_Hat/example/basic_demo
make
sudo ./basic_demo
```



6. The demo code for WS2812 is `/basic_demo/py/ws2812_test.py`.

- The LED configuration:

```
# LED strip configuration:
LED_COUNT = 12      # Number of LED pixels.
LED_FREQ_HZ = 800000 # LED signal frequency in hertz (usually 800khz)
LED_DMA = 10       # DMA channel to use for generating signal (try 10)
LED_BRIGHTNESS = 50 # Set to 0 for darkest and 255 for brightest
# True to invert the signal (when using NPN transistor level shift)
LED_INVERT = False
LED_CHANNEL = 1     # set to '1' for GPIOs 13, 19, 41, 45 or 53
LED_PIN = 13       # BCM13 / GPIO23
```

- Initialize the ws2812:

```
strip = Adafruit_NeoPixel(LED_COUNT, LED_PIN, LED_FREQ_HZ,
                          LED_DMA, LED_INVERT, LED_BRIGHTNESS, LED_CHANNEL)
strip.begin()
```

- Turn the LED on:

```
for x in range(0, LED_COUNT):
    strip.setPixelColor(x, Color(255, 0, 0))
strip.show()
```

7.10 Connect to a HDMI Display to show the Linux desktop

Pi interface Hat base on Raspberry Pi development, also can be used as Raspberry Pi. Camera interface, HDMI interface, Micro USB interface and so on can be use at will. The 3.2inch LCD is communicated with Raspberry Pi by SPI protocol, it does not prevent the Raspberry Pi working with the HDMI display, and thus there 2 displays can be used simultaneously.

Connect the Pi interface Hat with Makerfabs 7 inch

display(<https://www.makerfabs.com/7-inch-hdmi-lcd-with-touch-for-raspberry-pi.html>) with HDMI cable:



8. Default program

All effects and functions displayed after booting are implemented by the default program. Follow the below steps and it will show how to get and execute the default program.

1. The default program had been downloaded to the Pi, the address in the Pi is “~/Code/PI_Interface_Hat”.

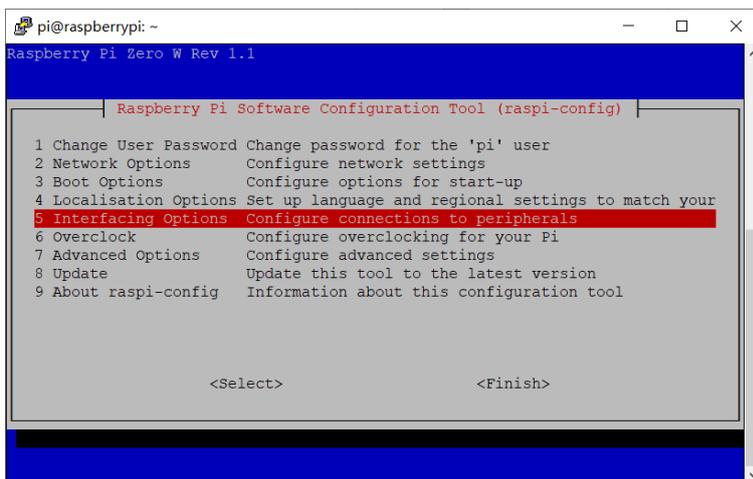
```
cd ~/Code/PI_Interface_Hat/example/default_demo/
```

2. **(Optional)** Make sure your Pi supports the SPI and I2C, and set the SPI and I2C interfaces available.

- Enter the following command in Raspberry terminal.

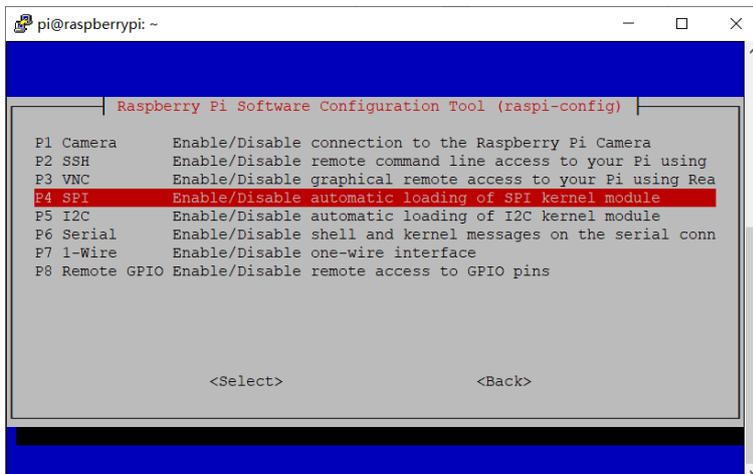
```
sudo raspi-config
```

- Click the interface options.



```
pi@raspberrypi: ~  
Raspberry Pi Zero W Rev 1.1  
Raspberry Pi Software Configuration Tool (raspi-config)  
1 Change User Password Change password for the 'pi' user  
2 Network Options Configure network settings  
3 Boot Options Configure options for start-up  
4 Localisation Options Set up language and regional settings to match your  
5 Interfacing Options Configure connections to peripherals  
6 Overclock Configure overclocking for your Pi  
7 Advanced Options Configure advanced settings  
8 Update Update this tool to the latest version  
9 About raspi-config Information about this configuration tool  
  
<Select> <Finish>
```

- Set the SPI and I2C available.



```
pi@raspberrypi: ~  
Raspberry Pi Software Configuration Tool (raspi-config)  
P1 Camera Enable/Disable connection to the Raspberry Pi Camera  
P2 SSH Enable/Disable remote command line access to your Pi using  
P3 VNC Enable/Disable graphical remote access to your Pi using Rea  
P4 SPI Enable/Disable automatic loading of SPI kernel module  
P5 I2C Enable/Disable automatic loading of I2C kernel module  
P6 Serial Enable/Disable shell and kernel messages on the serial conn  
P7 1-Wire Enable/Disable one-wire interface  
P8 Remote GPIO Enable/Disable remote access to GPIO pins  
  
<Select> <Back>
```

3. **(Optional)** Install the driver for the chip that support recording and playing. Please go to 7.1 to view the installation details.

4. Run the following command to execute the code.

```
make  
sudo ./default_demo
```

5. After the operation, the product will be got the same functions.

9. Re-install Raspberry Pi OS

Please check the official website of the Raspberry Pi <https://www.raspberrypi.org/software/> to install the raspberry pi OS.