Sensor Usage

Development Environment Setup

This tutorial applies to most of Radxa SBC products, here is a practical demonstration using the ROCK 4C+, other SBCs can be referred to here.

```
// Download the sample code
rock@rock-4c-plus:~$ sudo apt-get install cmake build-essential git python3-dev
-y
rock@rock-4c-plus:~$ git clone https://github.com/nascs/sample_code.git
rock@rock-4c-plus:~$ source sample_code/env.sh

// LibwiringX installation
rock@rock-4c-plus:~$ git clone https://github.com/wiringX/wiringX.git
rock@rock-4c-plus:~$ cd wiringX
rock@rock-4c-plus:~/wiringX/$ mkdir build
rock@rock-4c-plus:~/wiringX/$ cd build
rock@rock-4c-plus:~/wiringX/build$ cmake ..
rock@rock-4c-plus:~/wiringX/build$ make -j4
rock@rock-4c-plus:~/wiringX/build$ cpack -G DEB
rock@rock-4c-plus:~/wiringX/build$ sudo dpkg -i libwiringx*.deb
```

LCD 1602/2004

keyestudio 1602 I2C module is a 16 character by 2 line LCD display with Blue background and White backlight. keyestudio 2004 I2C Module is a 20 character by 4 line LCD display with Blue background and White backlight. Here's how they works on the radxa platform:

1. Using the rsetup utility to open the i2c7

2. Power down and switch off the power supply.

3. Wire the LCD as follows

```
LCD <--> Radxa ROCK 4
GND <--> GND
VCC <--> 5V
SDA <--> Pin 3
SCL <--> Pin 5
```

4. Reboot and check if i2c7 is on

```
radxa@rock-4c-plus:~$ ls /dev/i2c-*
/dev/i2c-0  /dev/i2c-7  /dev/i2c-9 # Enabled to detect /dev/i2c-7
```

5. Check if the LCD is recognized properly

```
radxa@rock-4c-plus:~$ sudo i2cdetect -r -y 7
         0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:                                                 -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- 27 -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

6. Running the test program

```
sudo python LCD1602.py
```

The program displays the current time of the system.

ADXL345

The ADXL345 is a small, thin, low power, 3-axis MEMS accelerometer with high resolution (13-bit) measurement at up to +-16 g. Digital output data is formatted as 16-bit twos complement and is accessible through either a SPI (3- or 4-wire) or I2C digital interface. The ADXL345 is well suited to measure the static acceleration of gravity in tilt-sensing applications, as well as dynamic acceleration resulting from motion or shock. Its high resolution (4 mg/LSB) enables measurement of inclination changes less than 1.0 degrees. Here's how it works on the radxa

platform:

1. Using the rsetup utility to open the i2c7

2. Power down and switch off the machine.

3. Wire the sensor as follows

```
ADXL345 <--> Radxa ROCK 4
GND <--> GND
VCC <--> 5V
SDA <--> Pin 3
SCL <--> Pin 5
```

4. Reboot and check if i2c7 is on

```
radxa@rock-4c-plus:~$ ls /dev/i2c-*
/dev/i2c-0  /dev/i2c-7  /dev/i2c-9
```

5. Running the test program

```
root@rock-4c-plus:/home/radxa/sample_code/modules/keyestudio# gcc adxl345.c -
lwiringx
root@rock-4c-plus:/home/radxa/sample_code/modules/keyestudio# ./a.out
```

The program displays acceleration data in the x y z directions.

Button + LED

There is a button and a light emitting diode. Both can be programmed to control the light emitting diode by pressing the button. Here's how they works on the radxa platform:

1. Wiring Method

```
led <--> Radxa ROCK 4
 s <--> Pin 3
 v <--> 3.3V/5V
```

```
 g <--> GND

button <--> Radxa ROCK 4
 s <--> Pin 5
 v <--> 3.3V/5V
 g <--> GND
```

2. Running the test program

```
radxa@rock-4c-plus:~$ gcc button_led.c -lwiringx
radxa@rock-4c-plus:~$ sudo ./a.out
```

The program shows whether the button is pressed or not, if the button is pressed, the led is off.

Ultrasonic sensor

The Keyestudio SR01 Ultrasonic sensor is a very affordable and detects the distance between the ultrasonic sensor and obstacle. It adopts CS100A chip, compatible with 3.3V and 5V. Its maximum detection distance is 3m and blind area is less than 4cm. As same as the principle of the bat, ultrasonic module sends a high frequency signals that the human body cannot hear. They will travel back if encountering the obstacle. On receiving the returned information, it will calculate the distance between the sensor and the obstacle by determining the time difference of the transmitted signal and the received signal. Here's how it works on the radxa platform:

1. Wiring Method

```
ultrasonic sensor <--> Radxa ROCK 4
trig <--> Pin 3
echo <--> Pin 5
v <--> 3.3V/5V
g <--> GND
```

2. Running the test program

```
root@rock-4c-plus:/home/radxa/sample_code/modules/keyestudio# gcc
ultrasonic_sensor.c -lwiringx
```

```
root@rock-4c-plus:/home/radxa/sample_code/modules/keyestudio# ./a.out
```

The program outputs the distance measured by the ultrasonic sensor.

4-digit 7-segment Display

The keyestudio 4-digit LED Display Module integrates a 0.36" 4-Digit 7-Segment Display Common Anode with 12 pins. It uses the driver chip TM1637. The module has 4 control pins of 2.54mm pitch, which can direct connect to microcontroller with jumper wires. So it’s very easy to control the matrix without tons of wiring. Available to make microcontroller control the 4-digit LED segment display via signal interface, greatly saving IO pin resource on microcontroller. The module has two 3mm fixed holes, convenient for mounting on other devices. If you've been eyeing matrix displays but hesitated because of the complexity, this is the solution you've been looking for! Here's how it works on the radxa platform:

1. Wiring Method

```
4-digit 7-segment Display <--> Radxa ROCK 4
CLK <--> Pin 40
DIO <--> Pin 38
VCC <--> 3.3V/5V
GND <--> GND
```

2. Running the test program

```
root@rock-4c-plus:/home/radxa/sample_code/modules/keyestudio# gcc tm1637.c -
lwiringx
root@rock-4c-plus:/home/radxa/sample_code/modules/keyestudio# ./a.out
```

The program displays the number "1024".

OLED module

OLED is short for organic light emitting diode. On the microscopic level, an OLED display is a matrix of organic LEDs that light up when they emit energy. Old LCD (Liquid Crystal Display) technology uses electronically controlled polarizers to change the way light passes or does not

pass through them. This requires an external backlight that lights up the whole display underneath. This uses a lot of energy because at the time the display is on, enough light for all pixels must be provided. The new OLED technology only uses electricity per pixel. Because each pixel creates its own light, only the pixels that are on use electricity. This makes OLED technology very efficient; also, the way these types of OLEDs are built allows them to be very thin compared to LCD. Here's how it works on the radxa platform:

1. Using the rsetup utility to open the i2c7

2. Shutdown and power down

3. Wiring Method

```
OLED <--> Radxa ROCK 4
 GND    <-->    GND
 VCC    <--> 5V
 SDA    <-->    Pin 3
 SCL    <-->    Pin 5
```

4. Reboot and check if i2c7 is on

```
radxa@rock-4c-plus:~$ ls /dev/i2c-*
/dev/i2c-0  /dev/i2c-7  /dev/i2c-9
```

5. Check if the OLED is recognized properly

```
radxa@rock-4c-plus:~$ sudo i2cdetect -r -y 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: -- -- -- -- -- -- -- --
```

6. Running the test program

```
root@rock-4c-plus:/home/radxa/sample_code/modules/keyestudio# gcc oled.c -
lwiringx
root@rock-4c-plus:/home/radxa/sample_code/modules/keyestudio# ./a.out
```

Buzzer/LED/IR transmitter

There is a buzzer, an led and an IR emitter. All three can be programmed and controlled. Here is how it works on the radxa platform:

1. Wiring Method

```
buzzer/led/ir transmitter Radxa ROCK 4
s <--> Pin 3
v <--> 3.3V/5V
g <--> GND
```

2. Running the test program

```
radxa@rock-4c-plus:~ cd sample_code/wiringX
radxa@rock-4c-plus:~/sample_code/wiringX$ gcc blink.c -lwiringx
```

The program will flash the light every second, and the buzzer, if it's on, will go off every second.

TCS34725

The keyestudio TCS34725 sensor mainly uses TCS34725 color sensor chip. It can communicate with other controllers via I2C communication interface. Keyestudio TCS34725 color sensor is a low cost, cost effective RGB full-color color recognition sensor. The sensor can recognize the surface color of an object through optical sensing. The sensor is illuminated by bright light and outputs corresponding RGB values to help to restore the color. Moreover, to avoid the surroundings interference and increase the accuracy, we particularly add an infrared light shielding plate on the sensor’s bottom, so that infrared spectrum element of incident light is minimized to make color management more accurate. On the sensor bottom you can see 4 yellow highlight LEDs, which can ensure that the sensor can use normally under low ambient light, actually realizing the fill light function. The sensor has high sensitivity, wide dynamic range, and infrared shading filter. It is an ideal color-sensitive component solution. It is widely

used for RGB LED backlight control, solid-state lighting, health products, etc. Here is how it works on the radxa platform:

1. Get test code

```
radxa@rock-4c-plus:~$ git clone https://github.com/nascs/TCS34725.git
radxa@rock-4c-plus:~$ cd TCS34725
radxa@rock-4c-plus:~/TCS34725$ git checkout -b test origin/test
```

2. Using the rsetup utility to open the i2c7

3. Shutdown and power down

4. Wiring Method

```
TCS34725 Radxa ROCK 4
 GND <--> GND
 VCC <--> 5V
 SDA <--> Pin 3
 SCL <--> Pin 5
```

5. Reboot and check if i2c7 is on

```
radxa@rock-4c-plus:~$ ls /dev/i2c-*
/dev/i2c-0  /dev/i2c-7  /dev/i2c-9
```

6. Check if TCS34725 is recognized

```
radxa@rock-4c-plus:~$ sudo i2cdetect -r -y 7
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:                                         -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- 29 -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

7. Running the test program

```
radxa@rock-4c-plus:~/TCS34725$ sudo python3 example.py
```

The program needs to be run in a dimly lit environment and will recognize the surface color of the object and output the RGB value.

DS3231

DS3231 is equipped with integrated TCXO and crystal, which make it a cost-effective I2C real time clock with high precision. The device carries a battery input, so if you disconnect the main power supply, it can still maintain accurate timing. The integrated oscillator ensures the long-term accuracy of the device and reduces the number of components. DS3231 provides both commercial and industrial temperature range and supports 16 pins small-outline package (300mil). The module itself can adapt to the system of 3.3V and 5V without level switch, which is quite convenient! Here's how it works on the radxa platform:

1. Use rsetup to open the ds3231

2. Wiring Method

```
DS3231 Radxa ROCK 4
GND <--> GND
VCC <--> 5V
SDA <--> Pin 3
SCL <--> Pin 5
```

3. Reboot and check if i2c7 is on

```
radxa@rock-4c-plus:~$ ls /dev/i2c-*
/dev/i2c-0  /dev/i2c-7  /dev/i2c-9
```

4. Check if TCS34725 is recognized

```
radxa@rock-4c-plus:~$ sudo i2cdetect -r -y 7
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:                                         -- -- -- -- -- -- -- --
```

```
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- UU -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

5. add a new rtc device

```
root@rock-4c-plus:/home/radxa# echo ds3231 0x68 | sudo tee  /sys/class/i2c-adapter/i2c-7/new_device
```

6. checkout the new rtc device

```
root@rock-4c-plus:/home/radxa# ls /dev/rtc*
/dev/rtc  /dev/rtc0 /dev/rtc1
```

7. read time from rtc module

```
root@rock-4c-plus:/home/radxa# hwclock -r -f /dev/rtc1
2000-01-01 00:01:40.083622+08:00
```

8. set the system time

```
root@rock-4c-plus:/home/radxa# apt-get install ntp -y
root@rock-4c-plus:/home/radxa# sudo service ntp start

or
date -s "2023-05-09 15:16:35"
```

9. write time to rtc module

- write time to rtc module after you make sure the time of linux is right

```
root@rock-4c-plus:/home/radxa# hwclock -w -f /dev/rtc1
```

- then read the hardware RTC at the time to see if it is correct

```
root@rock-4c-plus:/home/radxa# hwclock -r -f /dev/rtc1
2023-05-09 15:18:45.390726+08:00
```

```
root@rock-4c-plus:/home/radxa# hwclock -s -f /dev/rtc1
```

- use timedatectl to view all the time information

```
root@rock-4c-plus:/home/radxa# timedatectl
               Local time: Tue 2023-05-09 15:21:49 CST
           Universal time: Tue 2023-05-09 07:21:49 UTC
                 RTC time: Thu 1970-01-01 00:04:27
                Time zone: Asia/Shanghai (CST, +0800)
System clock synchronized: no
              NTP service: n/a
          RTC in local TZ: yes

Warning: The system is configured to read the RTC time in the local time zone.
         This mode cannot be fully supported. It will create various problems
         with time zone changes and daylight saving time adjustments. The RTC
         time is never updated, it relies on external facilities to maintain it.
         If at all possible, use RTC in UTC by calling
         'timedatectl set-local-rtc 0'.
```

10. set automatic startup

```
root@rock-4c-plus:/home/radxa# touch /etc/rc.local
root@rock-4c-plus:/home/radxa# chmod 777 /etc/rc.local
root@rock-4c-plus:/home/radxa# cat /etc/rc.local
#! /bin/bash

echo ds3231 0x68 | sudo tee  /sys/class/i2c-adapter/i2c-3/new_device
sudo hwclock -s -f /dev/rtc1
```

DS18B20

DS18B20 is a digital temperature sensor. It can be used to quantify environmental temperature testing. The temperature range is -55 ~ +125 °C, inherent temperature resolution 0.5 °C. It also support multi-point mesh networking. The DS18B20 can be deployed to achieve multi-point temperature measurement. It has a 9-12 bit serial output. Here's how it works on the radxa platform:

1. Use the rsetup utility to open the one wire

2. Shutdown and power down

3. Wiring Method

```
DS18B20 Radxa ROCK 4
s <--> Pin 37 (GPIO4_D6)
v <--> 3.3V/5V
g <--> GND
```

4. Reboot and check if the device is recognized

If Wire is working properly, there will be a device starting with 28 in the directory /sys/bus/w1/devices/.

5. Running the test program

```
radxa@rock-4c-plus:~/sample_code/modules/keyestudio$ gcc ds18b20.c
radxa@rock-4c-plus:~/sample_code/modules/keyestudio$ sudo ./a.out
```

The program outputs the current temperature.

Fan motor

keyestudio L9110 fan control module adopts L9110 motor control chip. It can control the rotation direction of the motor, hence the fan. The module is designed with mounting hole, compatible with servo motor control. The module is of high efficiency, with the high quality fan, it can easily blow out flame of a light in 20cm distance. It’s an essential part in fire fighting robot development. Here's how it works on the radxa platform:

1. Wiring Method

```
fan Radxa ROCK 4
INA <--> Pin 13
INB <--> Pin 14
VCC <--> 3.3V/5V
```

```
GND <--> GND
```

2. Running the test program

```
radxa@rock-4c-plus:~/sample_code/modules/keyestudio$ gcc fan_motor.c -lwiringx
radxa@rock-4c-plus:~/sample_code/modules/keyestudio$ sudo ./a.out
```

IR Receiver

IR is widely used in remote control. With this IR receiver, your project is able to receive command from any IR remoter controllers if you have right decoder. On our radxa platform, we have adapted the Car Mp3 remote control. Here's how it works on the radxa platform:

1. Use the rsetup utility to open the IR receiver

2. Wiring Method

```
IR Receiver Radxa ROCK 4
s <--> Pin 13
v <--> 3.3V/5V
g <--> GND
```

3. Install the test software and perform the test

```
radxa@rock-4c-plus:~$ sudo apt-get install evtest -y
radxa@rock-4c-plus:~$ sudo evtest
```

We have adapted the "Car MP3" remote control. When you press the buttons of the remote control, you can see the corresponding information on the terminal.

MG90s

A servo is an actuator for position (angle) servoing for those control systems where the angle needs to be constantly changing and can be maintained. Here's how it is used on the radxa

platform:

1. Wiring Method

```
MG90s Radxa ROCK 4
s <--> Pin 13
v <--> 3.3V/5V
g <--> GND
```

2. Running the test program

```
radxa@rock-4c-plus:~/sample_code/modules/keyestudio$ gcc servo.c -lwiringx
radxa@rock-4c-plus:~/sample_code/modules/keyestudio$ sudo ./a.out 90
//90 is angle
```

This program causes the MG90s to rotate at the angle you set.

WS28b20

WS2812B is a digital programmable LED lamp bead, also known as Neopixel, which is based on a single bus control, multiple WS2812B lamp beads can be connected in series on a single data line, and each lamp bead has a built-in control IC and RGB LEDs. by sending color data to one of the lamp beads, synchronous control of all lamp beads can be realized, thus forming a variety of colors and dynamic effects of LED lighting system. With small size, low power consumption and strong programmability, WS2812B light beads are widely used in lighting decoration, visual arts, smart home and other fields. Here's how it is used on the radxa platform:

1. Use the rsetup utility to open spi1

2. Shutdown and powe down

3. Wiring Method

```
ws2812b Radxa ROCK 4
IN <--> Pin 19
GDN <--> GND
```

```
VCC <--> VCC
```

5. Reboot and check if spi1 opens properly

6. Running the test program

```
radxa@rock-4c-plus:~/sample_code/modules/keyestudio$ sudo python3 ws2812b.py
```

The program will light up the WS2812B lamps one by one and then turn them all off again.

DHT11/DHT22

DHT11/DHT22 is a digital temperature and humidity sensor. It adopts high-precision temperature and humidity sensor and special analog-to-digital converter, which can digitize the temperature and humidity information and then transmit it through a single bus.DHT11 is suitable for indoor environment measurement, with a measuring range of temperature 0-50°C, humidity 20-90RH%, and an accuracy of ±2°C and ±5%RH, and DHT22 is suitable for more demanding environment measurement, with a measuring range of -40~~80°C, humidity 0-100RH%, and an accuracy of ±0.5°C and ±2%RH.These two sensors have a simple structure, easy to use, and low price. The DHT22 is suitable for more demanding environment measurement, with a measuring range of -40~~80°C, humidity 0-100RH%, and accuracy of ±0.5°C and ±2%RH. These two sensors have the advantages of simple structure, easy to use, and low price, which are widely used in the fields of Internet of Things, smart home, and environment monitoring. Here's how it is used on the radxa platform:

1. Wiring Method

```
DHT11/DHT22 Radxa ROCK 4
S <--> Pin 3
GDN <--> GND
VCC <--> VCC
```

2. Running the test program

This program outputs the temperature and humidity detected by the DHT module

```
radxa@rock-4c-plus:~/sample_code/modules/keyestudio$ gcc dht11.c -lwiringx
radxa@rock-4c-plus:~/sample_code/modules/keyestudio$ ./a.out ROCK 4 8
```

This Bluetooth module can easily achieve serial wireless data transmission. Its operating frequency is among the most popular 2.4GHz ISM frequency band (i.e. Industrial, scientific and medical). It adopts Bluetooth 2.1+EDR standard. In Bluetooth 2.1, signal transmit time of different devices stands at a 0.5 seconds interval, so that the workload of Bluetooth chip can be reduced substantially and more sleeping time can be saved for Bluetooth. This module is set with serial interface, which is easy-to-use and simplifying overall design/development cycle. Here's how it is used on the radxa platform:

1. Use the rsetup utility to open the uart4

2. Shutdown and power down

3. Wiring Method

```
HC-06 Radxa ROCK 4
RXD <--> Pin 19
TXD <--> Pin 21
VCC <--> 3.3/5V
GND <--> GND
```

4. Check if uart4 is turned on properly

```
radxa@rock-4c-plus:~$ ls
/dev/ttyS4
```

5. Install minicom or other serial debugging tools.

```
radxa@rock-4c-plus:~$ sudo apt-get install minicom -y
```

6. Open the serial port debugging tool

```
sudo minicom -D /dev/ttyS4 -b 9600
```

7. Install the Bluetooth debugging tool on your cell phone or computer and connect the Bluetooth module.

8. Test data sending and receiving

Use the Bluetooth debugging tool of your cell phone or computer to send data, and the minicom will receive the data sent from the cell phone or computer side.

Rotation Sensor

The voltage of the analog rotary sensor can be subdivided into 1024. Here's how it is used on the radxa platform:

1. Wiring Method

```
rotation sensor Radxa ROCK 4
s <--> Pin 26
v <--> 3.3/5V
g <--> GND
```

2. Running the test program

```
radxa@rock-4c-plus:~/sample_code/modules/keyestudio$  gcc rotation_sensor.c
radxa@rock-4c-plus:~/sample_code/modules/keyestudio$  sudo ./a.out
IIO device value: 954
```

RfID RC522

MF522-AN module adopts Philips MFRC522 original reader circuit chip design, easy to use, low cost, suitable for equipment development, development of advanced applications such reader users, the need for RF card terminal design / production of the user. This module can be loaded directly into a variety of readers molds. Module uses voltage of 3.3V, and can be directly connected to the user any CPU board communication module through the SPI interface using simple few lines,which can guarantee stable and reliable reader distance. Here's how it is used on the radxa platform:

1. Use the rsetup utility to open spi1

2. Shutdown and power down

3. Wiring Method

```
RfID RC522 Radxa ROCK 4
vcc <--> 3.3V/5V
rst <--> Pin 36
gnd <--> GND
miso <--> Pin 21
mosi <--> Pin 19
sck <--> Pin 23
nss <--> Pin 38
irq <--> Pin 40
```

2. Reboot and check if spi1 opens properly

```
radxa@rock-4c-plus:~$ ls /dev/spidev*
/dev/spidev1.0
```

2. Installation of necessary libraries

```
sudo apt update
sudo apt install python-dev python3-dev
sudo pip3 install spidev
sudo pip3 install mfrc522
```

3. Modifying library files

Change RPI.GPIO to OPi.GPIO in the python file under /usr/local/lib/python3.9/dist-packages/mfrc522.

4. Write card test

Place a blank access card on the reader and run the program, this will write a string "Hello, world"

5. Card reading test

Put the blank access card on the reader, run the program, if you read the card id and the string "Hello, world", it means the program is working properly.

GPIO Shield

The PCF8591 features four-8-bit analog to digital converters and a single 8-bit digital to analog converter. This will operate through the I2C interface on the Raspberry Pi. And it also works on the radxa platform, Here's how it is used on the radxa platform:

1. Using the rsetup utility to open the i2c7

2. Shutdown and power down

3. Wiring Method

Connect the gpio shield to Pin 40 of ROCK 4.

4. Reboot and check if the gpio shield is recognized properly

5. Running the test program

```
sudo python3 gpio_shield_pcf8591.py
```

This program will read the analog signal on the gpio shield.