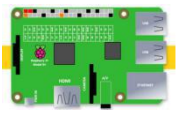
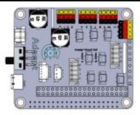
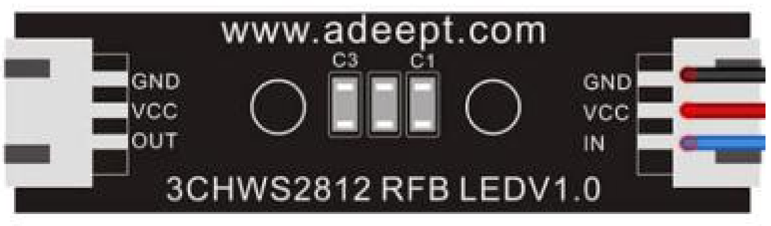
Adjusting Line Tracking Module

1. Components used in this course

Components	Quantity	Picture
Raspberry Pi	1	
Robot HAT	1	
3 pin cable	1	
WS2812 RGB LED	1	
5 pin cable	1	
Tracking Module	1	

2. Introduction of WS2812 RGB LED

WS2812 RGB module is a low-power RGB tri-color lamp with integrated current control chip. Its appearance is the same as a 5050LED lamp bead, and each element is a pixel. The pixel contains an intelligent digital interface data latch signal shaping amplification drive circuit, a high-precision internal oscillator and a 12V high-voltage programmable constant current control part, which effectively ensures the color of the pixel point light is highly consistent.



WS2812 LED is a very common module on our robot products. There are three WS2812 lights on each module. Please pay attention to the signal line when connecting. The signal line needs to be connected to WS2812 after being led out from the Raspberry Pi. The "IN" end of the LED light module.

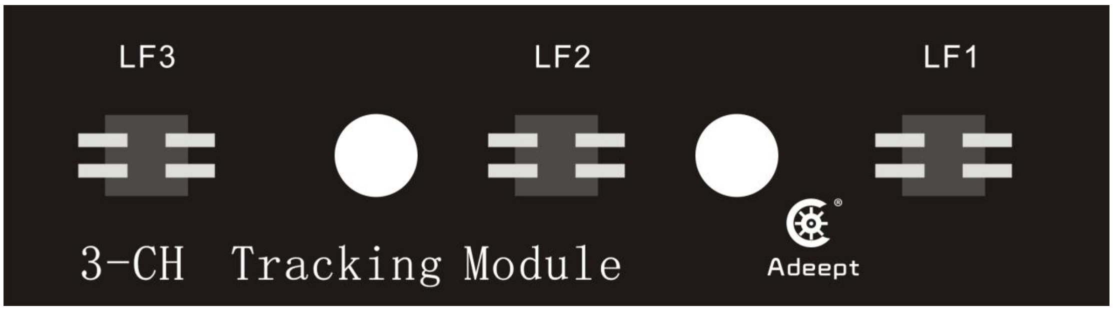
When using the Raspberry Pi to install the driver board RobotHAT, the WS2812 LED module can be connected to the WS2812 interface on the RobotHAT with a 3pin cable.

If you connect the WS2812 LED module to the WS2812 interface of RobotHAT, the signal line is equivalent to the GPIO 12 of the Raspberry Pi.

3. Introduction to the line tracking module

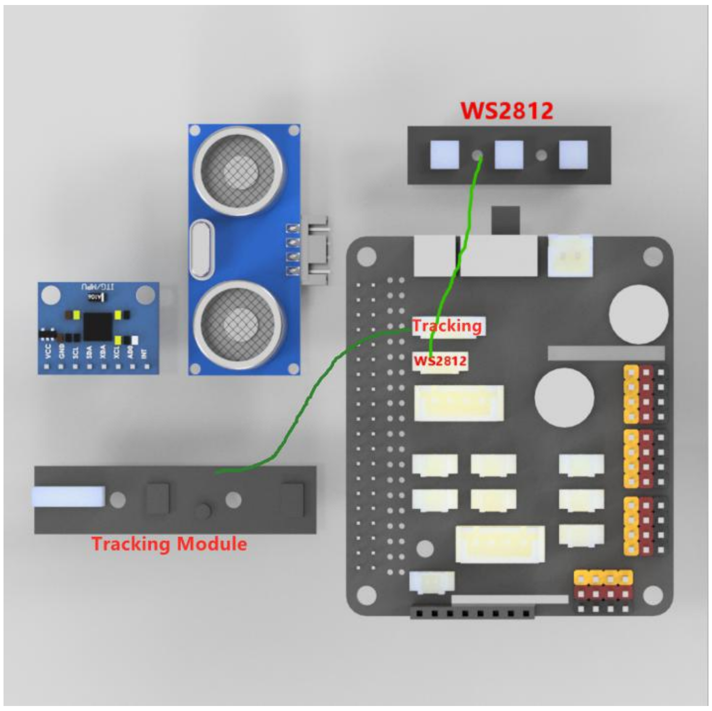
On white paper with black lines, because the black lines and white paper have different reflection coefficients to light, the black lines can be judged according to the intensity of the received reflected light. In the Tracking module, a more common detection method-infrared detection method is used.

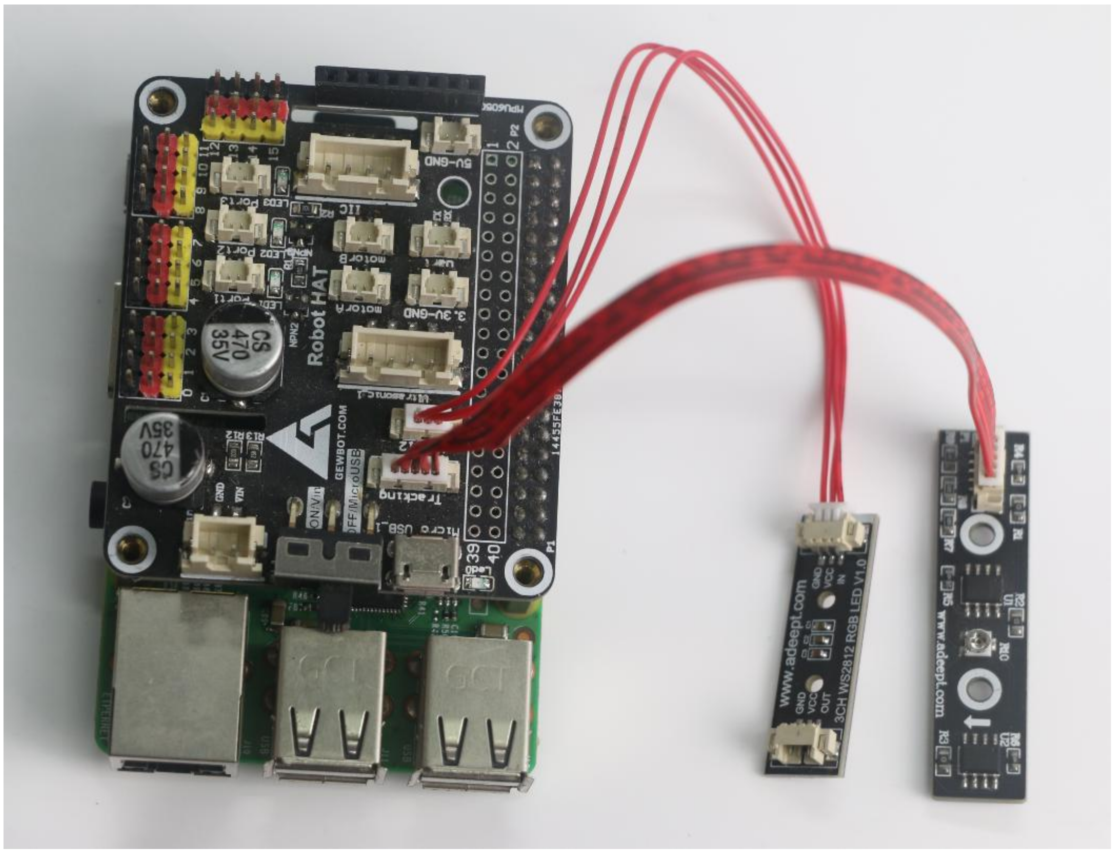
Infrared detection method uses infrared rays to have different reflection properties on different colored physical surfaces. When the program is running, the infrared light is continuously emitted to the ground. When the infrared light meets the white ground, the diffuse emission occurs, and the reflected light is received by the receiving tube; If it encounters a black line, the infrared light is absorbed, and the receiving tube on the car cannot receive the signal.



4. Circuit diagram (wiring diagram)

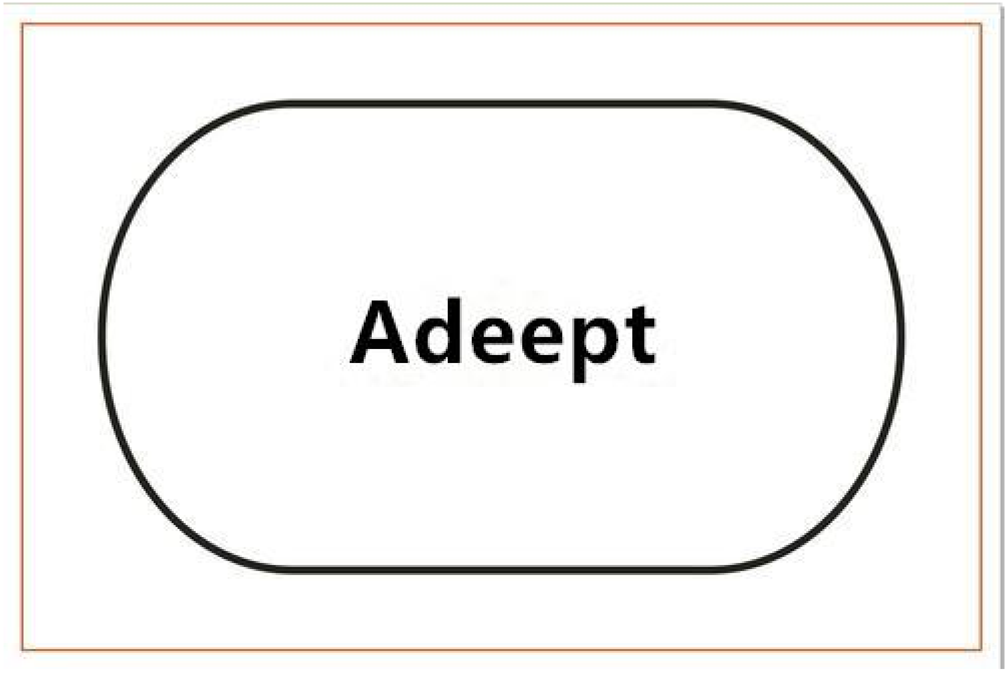
When the WS2812 LED module is in use, the IN port needs to be connected to the WS2812 port on the RobotHAT driver board, as shown in the figure below:

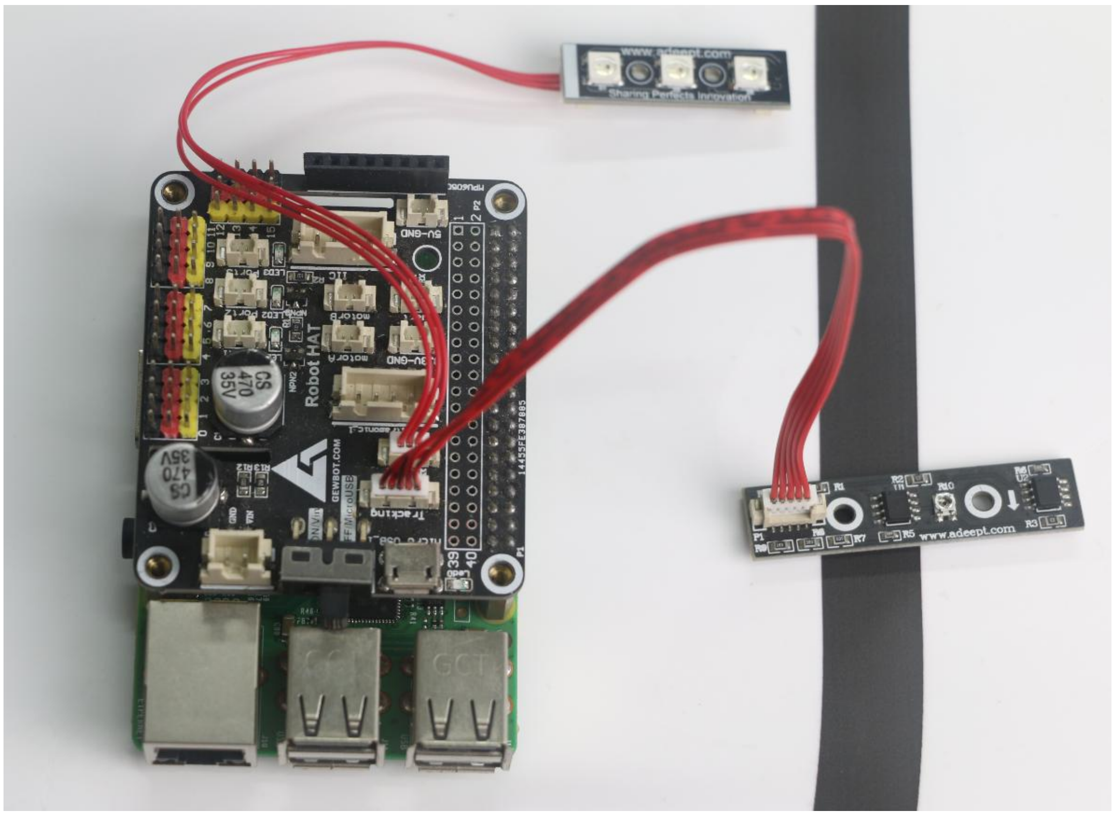




5. Make a track for debugging the line-tracking module

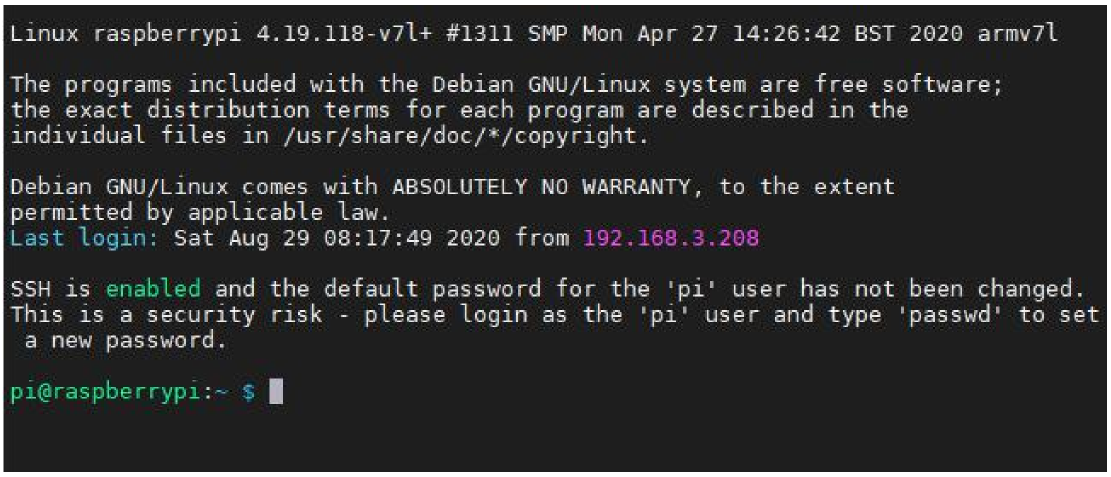
On the track of the robot car you have made, debug the line-tracking module.





6. Running the debugged program

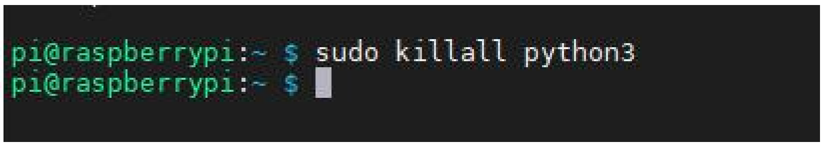
1. Log in to your Raspberry Pi via SSH:



```
Linux raspberrypi 4.19.118-v7l+ #1311 SMP Mon Apr 27 14:26:42 BST 2020 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Aug 29 08:17:49 2020 from 192.168.3.208

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set
 a new password.

pi@raspberrypi:~ $
```

2. If your robot code has been configured, the Raspberry Pi will have a chance to run the webServer.py program. Please end the webServer.py program when running this program. The Raspberry Pi will still automatically run webServer.py when it is powered on next time, the command to end the program:

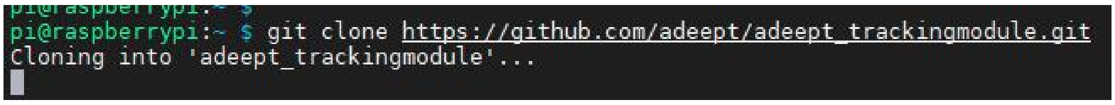
sudo killall python3



```
pi@raspberrypi:~ $ sudo killall python3
pi@raspberrypi:~ $
```

3. Download the code program, enter the following command in the console:

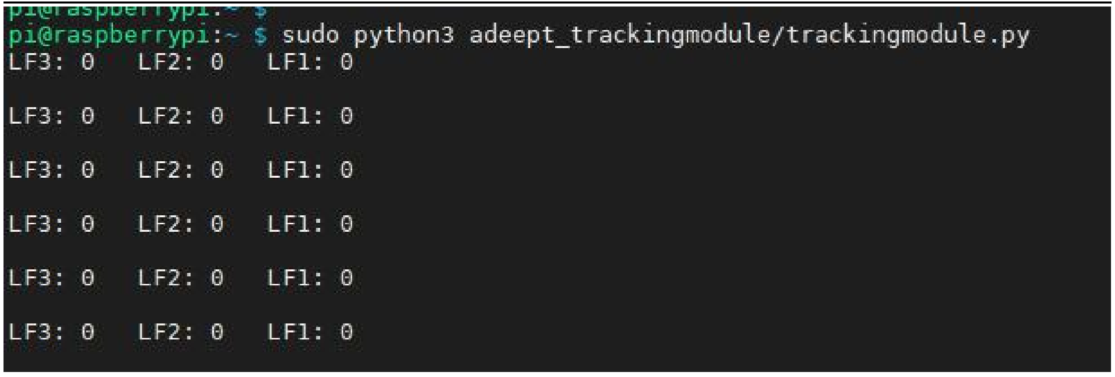
git clone https://github.com/adeept/ adeept_trackingmodule.git



```
pi@raspberrypi:~ $ git clone https://github.com/adeept/adeept_trackingmodule.git
Cloning into 'adeept_trackingmodule'...
```

4.After the download is complete, run the trackingmodule.py program and directly enter the following commands to run this program on the Raspberry Pi:

sudo python3 adeept_trackingmodule/tackingmodule.py

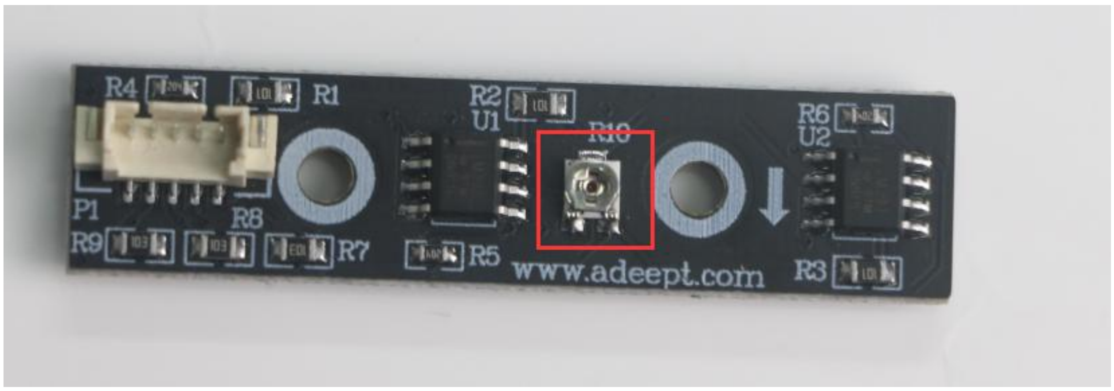


```
pi@raspberrypi:~ $
pi@raspberrypi:~ $ sudo python3 adeept_trackingmodule/trackingmodule.py
LF3: 0   LF2: 0   LF1: 0

LF3: 0   LF2: 0   LF1: 0

LF3: 0   LF2: 0   LF1: 0

LF3: 0   LF2: 0   LF1: 0

LF3: 0   LF2: 0   LF1: 0

LF3: 0   LF2: 0   LF1: 0
```

5. After running the program successfully, you will see the return values of the 3 detection points of the line tracking module, LF1, LF2, LF3 correspond to LF1, LF2, LF3 on the line tracking module. "0" means that the return value was received, and "1" means that the return value was not received. Note that when the line tracking module is not facing the ground, it will return "1" because the detection distance is too long.

6. Start debugging the line-tracking module. When LF1 is placed about 10 ~ 15mm above the black line, the console returns "LF3: 0 LF2: 0 LF1: 1", and it is observed that D1 of the WS2812 LED light is blue Color light, D2 and D3 are off. The height of the actual installation of the line tracking module for each robot product is different. Please refer to the tutorial or measure the height by yourself after installing the robot product.

7. If the phenomenon of "LF3: 0 LF2: 0 LF1: 1" does not appear or D1 lights up in blue but D2 and D3 are off, you need to use a screwdriver to rotate the potentiometer on the line tracking module until this phenomenon occurs. . Note: The height of the line tracking module will affect the results, please keep the height of the line-tracking module unchanged. The potentiometer of the line-tracking module is shown in the figure below.



8.After debugging, you can test LF2 and LF3. When LF2 or LF3 is placed above the black line, D2 and D3 corresponding to the WS2812 LED will also light up in blue.

9. When you want to terminate the running program after debugging, you can press the shortcut key "Ctrl + C" on the keyboard.

7. Main code program

Trackingmodule.py

```
import time
from rpi_ws281x import *
import RPi.GPIO as GPIO

# LED strip configuration:
LED_COUNT      = 3      # Number of LED pixels.
LED_PIN        = 12      # GPIO pin connected to the pixels (18 uses PWM!).
#LED_PIN        = 10      # GPIO pin connected to the pixels (10 uses SPI /dev/spidev0.0).
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 = 255     # Set to 0 for darkest and 255 for brightest
LED_INVERT     = False   # True to invert the signal (when using NPN transistor level shift)
LED_CHANNEL    = 0       # set to '1' for GPIOs 13, 19, 41, 45 or 53


line_pin_right = 19
line_pin_middle = 16
line_pin_left = 20
```

```

# Create NeoPixel object with appropriate configuration.
strip = Adafruit_NeoPixel(LED_COUNT, LED_PIN, LED_FREQ_HZ, LED_DMA, LED_INVERT, LE
D_BRIGHTNESS, LED_CHANNEL)
# Intialize the library (must be called once before other functions).
strip.begin()


def setup():
    GPIO.setwarnings(False)
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(line_pin_right,GPIO.IN)
    GPIO.setup(line_pin_middle,GPIO.IN)
    GPIO.setup(line_pin_left,GPIO.IN)
    #motor.setup()

# Define functions which animate LEDs in various ways.
def colorWipe( R, G, B):
    """Wipe color across display a pixel at a time."""
    color = Color(R,G,B)
    for i in range(strip.numPixels()):
        strip.setPixelColor(i, color)
        strip.show()

def run():
    status_right = GPIO.input(line_pin_right)
    status_middle = GPIO.input(line_pin_middle)
    status_left = GPIO.input(line_pin_left)
    print('LF3: %d   LF2: %d   LF1: %d\n'%(status_right,status_middle,status_left))

    if status_left == 1 :
        strip.setPixelColor(0, Color(0, 0, 255))
    else:
        strip.setPixelColor(0, Color(0, 0, 0))

    if status_middle == 1:
        strip.setPixelColor(1, Color(1, 0, 255))
    else:
        strip.setPixelColor(1, Color(1, 0, 0))

    if  status_right == 1:
        strip.setPixelColor(2, Color(2, 0, 255))
    else:
        strip.setPixelColor(2, Color(0, 0, 0))

    strip.show()
if __name__ == '__main__':
    try:
      setup()
      while 1:
        run()
      pass
    except KeyboardInterrupt:
      colorWipe(0, 0, 0)
```