oled屏使用指导说明

canvas module是AntJS系统基于不同屏幕的专用画图子模块，其接口与HTML5的canvas API保持兼容。本文是canvas module在oled屏上具体实现的指导说明。

1. oled屏介绍

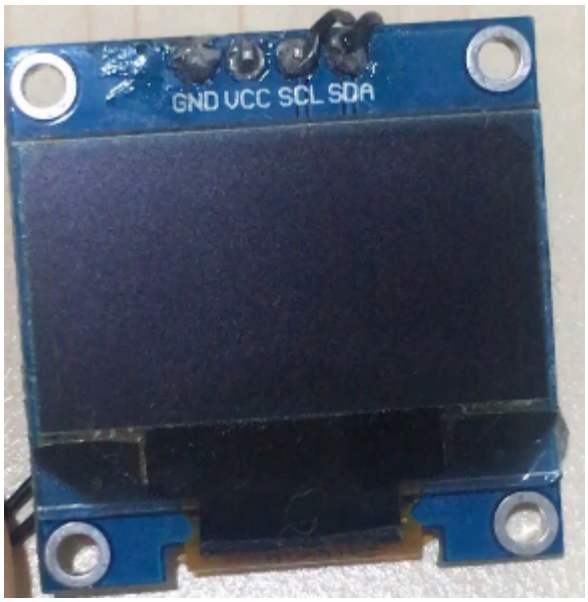
1.1. 基本信息

- 此款屏幕是 0.96inch oled屏模块，分辨率为 128x64，带有内部控制器，使用IIC接口进行通信。
- 屏幕具有功耗低，可视角度大等优点，常用于电子产品等显示应用。
- 显示尺寸： 21.7mm × 10.8mm

1.2. 硬件引脚

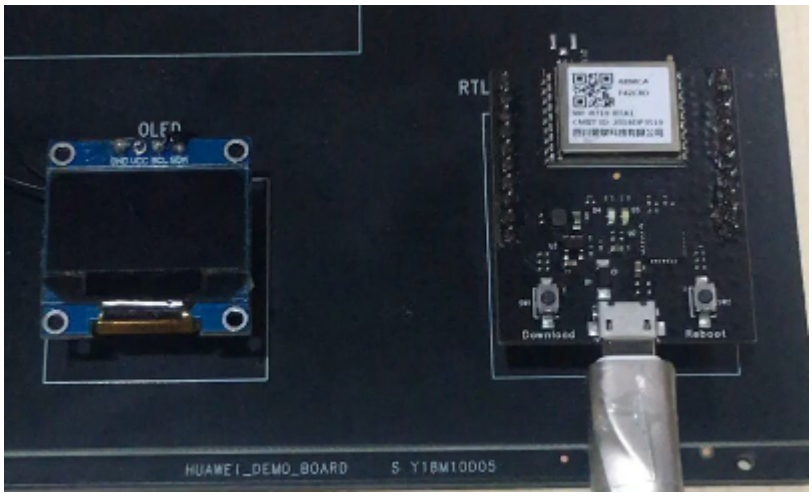
屏幕驱动板共有4个引脚，如下：

- VCC： 3.3V/5V
- GND： GND
- SCL： IIC时钟线
- SDA： IIC数据线



2. 硬件连接

oled屏与模块的硬件连接如下所示：



3. API接口

- `require("canvas")`
  获取canvas模块。
- `open(cfg)`
  打开canvas端口。其中cfg的格式如下：

```
cfg={screen_type:1,orientation:4};
```

- screen_type表示屏幕的类型：
  - 1 表示1.54 inch ink型屏幕，其像素规格为200*200；
  - 2 表示0.96 inch oled型屏幕，其像素规格为128*64；
  - 3 表示3.5 inch tft型屏幕，其像素规格为320*480。
- orientation表示旋转方向(即原点位置)：
  - 1 表示以屏幕左上角为原点；
  - 2 表示以屏幕右上角为原点；
  - 3 表示以屏幕左下角为原点；
  - 4 表示以屏幕右下角为原点。
- `beginPath()` 新建一条路径，路径一旦创建成功，图形绘制命令被指向到路径上生成路径。
- `closePath()` 闭合路径，之后图形绘制命令又重新指向到上下文中。
- `moveTo(x, y)` 把画笔移动到指定的坐标(x, y)。相当于设置路径的起始点坐标。
- `lineTo(x, y)` 绘制线段，端点分别为画笔当前坐标和(x, y)。
- `strokeRect(x1, y1, x2, y2)` 绘制矩形，对角顶点坐标分别为(x1, y1)、(x2, y2)。
- `fillRect(x1, y1, x2, y2)` 填充矩形，对角顶点坐标分别为(x1, y1)、(x2, y2)。
- `strokeTriangle(x1, y1, x2, y2, x3, y3)` 绘制三角形，三角形顶点坐标分别为(x1, y1)、(x2, y2)、(x3, y3)。
- `fillTriangle(x1, y1, x2, y2, x3, y3)` 填充三角形，三角形顶点坐标分别为(x1, y1)、(x2, y2)、(x3, y3)。

- `strokePolygon(verticesArray)` 绘制多边形，其中verticesArray为多边形的顶点集合，的格式如下：`var polygon = [[10,40], [20,20], [60,30], [80,10], [100,60]];`
- `fillPolygon(verticesArray)` 填充多边形，其中verticesArray为多边形的顶点集合，的格式如下：`var polygon = [[10,40], [20,20], [60,30], [80,10], [100,60]];`
- `arc(x, y, radius, start_angle, end_angle, anti_clockwise)` 绘制圆弧，圆心坐标(x, y)，半径为radius，起始角度为start_angle，终止角度为end_angle，anti_clockwise表示圆弧方向，0时表示顺时针方向，1时表示逆时针方向。
- `fillCircle(x, y, radius)` 填充圆形，圆心坐标(x, y)，半径为radius。
- `strokeEllipse(x1, x2, y1, y2)` 绘制椭圆，x1为最左侧点横坐标，x2为最右侧点横坐标，y1位最上侧点纵坐标，y2为最下侧点纵坐标。
- `fillEllipse(x1, x2, y1, y2)` 填充椭圆，x1为最左侧点横坐标，x2为最右侧点横坐标，y1位最上侧点纵坐标，y2为最下侧点纵坐标。
- `fillText(x, y, textString, fontString)` 书写字符串，字符串起始位置坐标为(x, y)，textString为字符串内容，fontString为所使用字体字符串。
- `stroke()` 显示屏幕内容。
- `clear()` 清除屏幕。

4. 范例

4.1 初始化

```
canvas = require("canvas");
cfg = {screen_type:2,orientation:1};
canvas_port = canvas.open(cfg);
```

4.2 orientation对比

以下将以绘制线段为例，对比演示cfg中orientation的作用。 代码模板如下：

```
canvas = require("canvas");
cfg = {screen_type:2,orientation:1};
canvas_port = canvas.open(cfg);
canvas_port.beginPath();
canvas_port.moveto(0,0);
canvas_port.lineto(30,30);
canvas_port.stroke();
canvas_port.closePath();
```

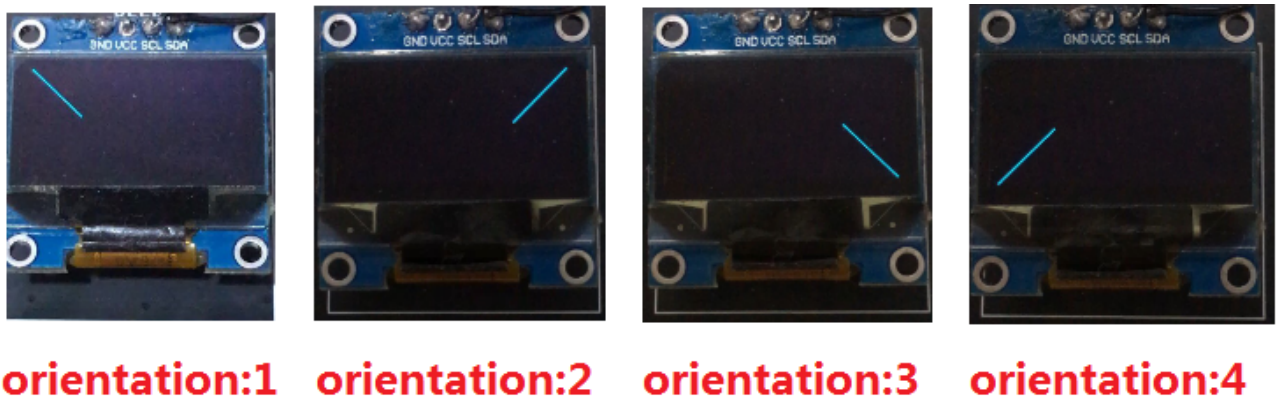
依次将代码模板中的cfg修改为：

```
cfg = {screen_type:2,orientation:2};
```

```
cfg = {screen_type:2,orientation:3};
```

```
cfg = {screen_type:2,orientation:4};
```

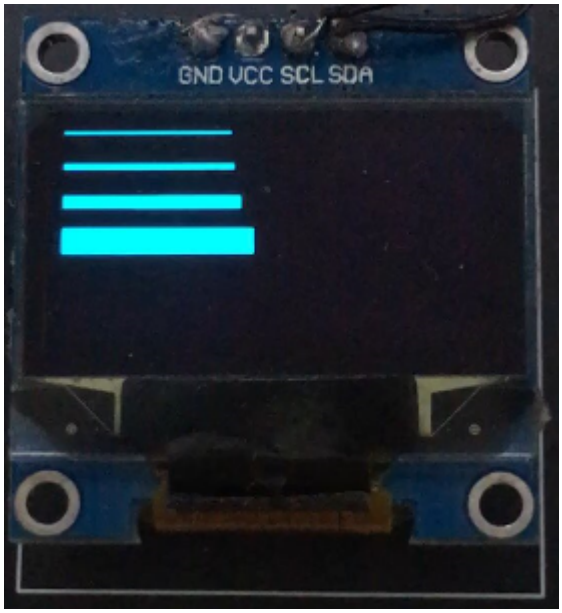
实际显示效果如下：



4.3 绘制线段

```
canvas = require("canvas");
cfg = {screen_type:2,orientation:1};
canvas_port = canvas.open(cfg);
canvas_port.beginPath();

canvas_port.lineWidth = 1;
canvas_port.moveto(0,0);
canvas_port.lineto(50,0);

canvas_port.lineWidth = 2;
canvas_port.moveto(0,10);
canvas_port.lineto(50,10);

canvas_port.lineWidth = 4;
canvas_port.moveto(0,20);
canvas_port.lineto(50,20);

canvas_port.lineWidth = 8;
canvas_port.moveto(0,30);
canvas_port.lineto(50,30);

canvas_port.stroke();
canvas_port.closePath();
```

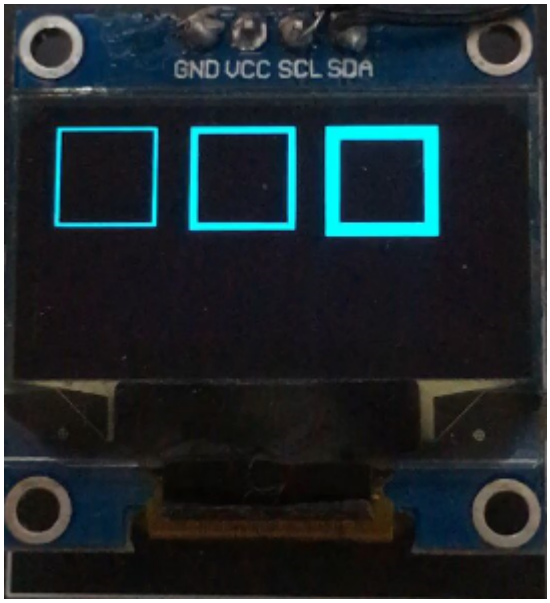
代码中可通过设置canvas_port.lineWidth来调整线宽，线宽的取值范围为[1, 8]。实际显示效果如下：



4.4 绘制矩形

```
canvas = require("canvas");
cfg = {screen_type:2,orientation:1};
canvas_port = canvas.open(cfg);
canvas_port.beginPath();

canvas_port.lineWidth = 1;
canvas_port.strokeRect(0,0,29,29);

canvas_port.lineWidth = 2;
canvas_port.strokeRect(40,0,69,29);

canvas_port.lineWidth = 4;
canvas_port.strokeRect(80,0,109,29);

canvas_port.stroke();
canvas_port.closePath();
```

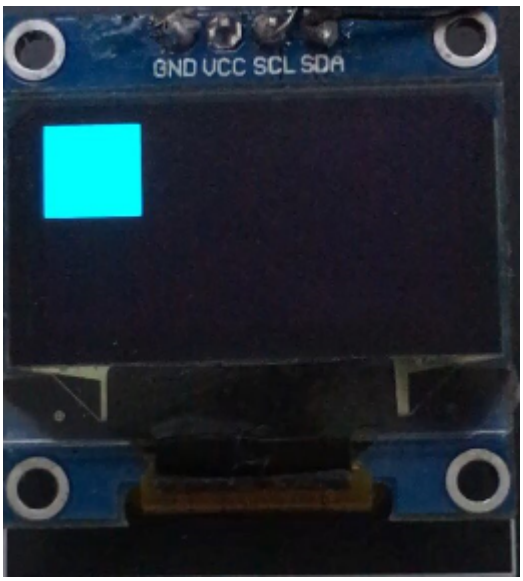
代码中可通过设置canvas_port.lineWidth来调整线宽，线宽的取值范围为[1, 8]。实际显示效果如下：



4.5 填充矩形

```
canvas = require("canvas");
cfg = {screen_type:2,orientation:1};
canvas_port = canvas.open(cfg);
canvas_port.beginPath();

canvas_port.fillRect(0,0,29,29);

canvas_port.stroke();
canvas_port.closePath();
```

实际显示效果如下：

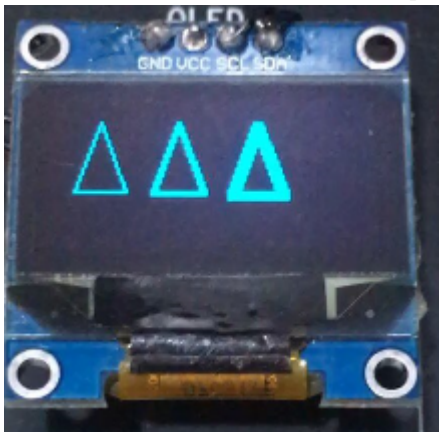


4.6 绘制三角形

```
canvas = require("canvas");
cfg = {screen_type:2,orientation:1};
canvas_port = canvas.open(cfg);
```

```
canvas_port.beginPath();

canvas_port.lineWidth = 1;
canvas_port.strokeTriangle(10,40,20,10,30,40);

canvas_port.lineWidth = 2;
canvas_port.strokeTriangle(40,40,50,10,60,40);

canvas_port.lineWidth = 4;
canvas_port.strokeTriangle(70,40,80,10,90,40);

canvas_port.stroke();
canvas_port.closePath();
```

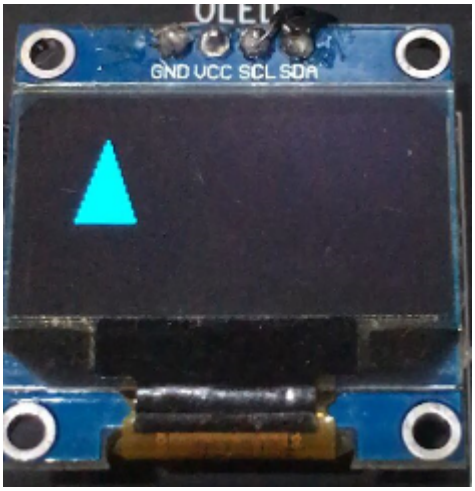
代码中可通过设置canvas_port.lineWidth来调整线宽，线宽的取值范围为[1, 8]。实际显示效果如下：



4.7 填充三角形

```
canvas = require("canvas");
cfg = {screen_type:2,orientation:1};
canvas_port = canvas.open(cfg);
canvas_port.beginPath();

canvas_port.fillTriangle(10,40,20,10,30,40);

canvas_port.stroke();
canvas_port.closePath();
```

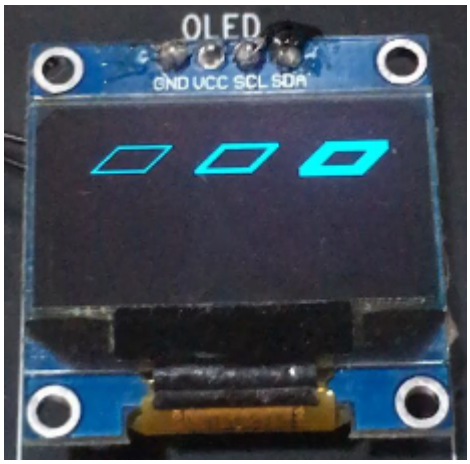
实际显示效果如下：



4.8 绘制多边形

```
canvas = require("canvas");
cfg = {screen_type:2,orientation:1};
canvas_port = canvas.open(cfg);
canvas_port.beginPath();

var verticesArray1 = [[10,20], [20,10], [40,10], [30,20]];
var verticesArray2 = [[50,20], [60,10], [80,10], [70,20]];
var verticesArray3 = [[90,20], [100,10], [120,10], [110,20]];

canvas_port.lineWidth = 1;
canvas_port.strokePolygon(verticesArray1);

canvas_port.lineWidth = 2;
canvas_port.strokePolygon(verticesArray2);

canvas_port.lineWidth = 4;
canvas_port.strokePolygon(verticesArray3);

canvas_port.stroke();
canvas_port.closePath();
```

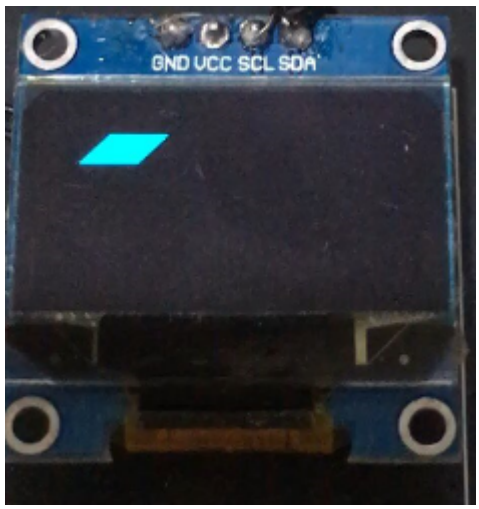
代码中可通过设置canvas_port.lineWidth来调整线宽，线宽的取值范围为[1, 8]。实际显示效果如下：



4.9 填充多边形

```
canvas = require("canvas");
cfg = {screen_type:2,orientation:1};
canvas_port = canvas.open(cfg);
canvas_port.beginPath();

var verticesArray1 = [[10,20], [20,10], [40,10], [30,20]];

canvas_port.fillPolygon(verticesArray1);

canvas_port.stroke();
canvas_port.closePath();
```

实际显示效果如下：



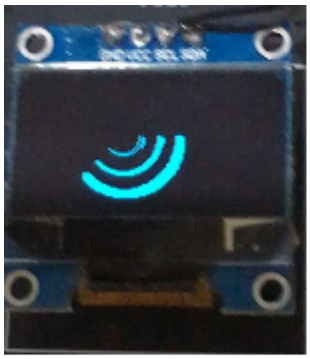
4.10 绘制圆弧

```
canvas = require("canvas");
cfg = {screen_type:2,orientation:1};
canvas_port = canvas.open(cfg);
canvas_port.beginPath();

canvas_port.lineWidth = 1;
canvas_port.arc(50, 30, 10, 0, 135, 0);

canvas_port.lineWidth = 2;
canvas_port.arc(50, 30, 20, 0, 135, 0);

canvas_port.lineWidth = 4;
canvas_port.arc(50, 30, 30, 0, 135, 0);


canvas_port.stroke();
canvas_port.closePath();
```

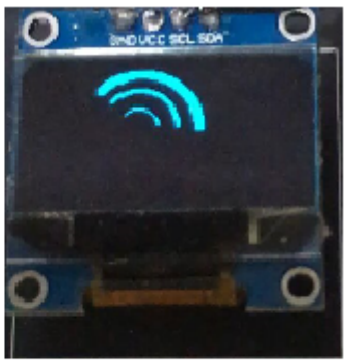
```
canvas = require("canvas");
```

```
cfg = {screen_type:2,orientation:1};
canvas_port = canvas.open(cfg);
canvas_port.beginPath();

canvas_port.lineWidth = 1;
canvas_port.arc(50, 30, 10, 0, 135, 1);

canvas_port.lineWidth = 2;
canvas_port.arc(50, 30, 20, 0, 135, 1);

canvas_port.lineWidth = 4;
canvas_port.arc(50, 30, 30, 0, 135, 1);


canvas_port.stroke();
canvas_port.closePath();
```

以上两段代码分别以顺时针方向和逆时针方向绘制圆弧。代码中可通过设置canvas_port.lineWidth来调整线宽，线宽的取值范围为[1, 8]。实际显示效果如下：



顺时针

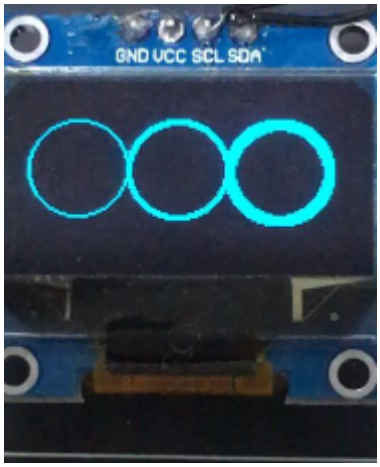


逆时针

4.11 绘制圆形

```
canvas = require("canvas");
cfg = {screen_type:2,orientation:1};
canvas_port = canvas.open(cfg);
canvas_port.beginPath();

canvas_port.lineWidth = 1;
canvas_port.arc(20, 30, 20, 0, 360, 0);

canvas_port.lineWidth = 2;
canvas_port.arc(60, 30, 20, 0, 360, 0);

canvas_port.lineWidth = 4;
canvas_port.arc(100, 30, 20, 0, 360, 0);

canvas_port.stroke();
canvas_port.closePath();
```

代码中可通过设置canvas_port.lineWidth来调整线宽，线宽的取值范围为[1, 8]。实际显示效果如下：



4.12 填充圆形

```
canvas = require("canvas");
cfg = {screen_type:2,orientation:1};
canvas_port = canvas.open(cfg);
canvas_port.beginPath();

canvas_port.fillCircle(60,30,25);

canvas_port.stroke();
canvas_port.closePath();
```

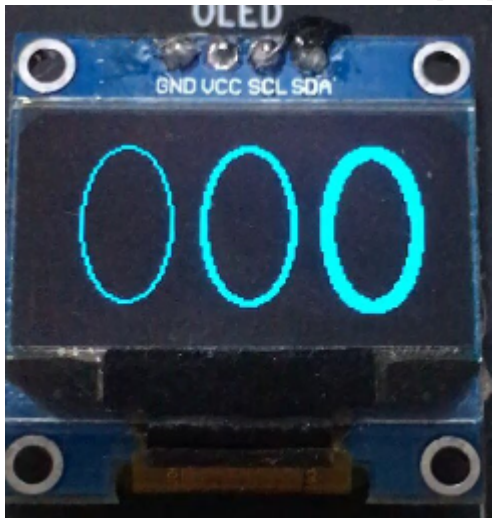
实际显示效果如下：

4.13 绘制椭圆

```
canvas = require("canvas");
cfg = {screen_type:2,orientation:1};
canvas_port = canvas.open(cfg);
canvas_port.beginPath();

canvas_port.lineWidth = 1;
canvas_port.strokeEllipse(10,40,5,60);
```

```
canvas_port.lineWidth = 2;
canvas_port.strokeEllipse(50,80,5,60);

canvas_port.lineWidth = 4;
canvas_port.strokeEllipse(90,120,5,60);

canvas_port.stroke();
canvas_port.closePath();
```

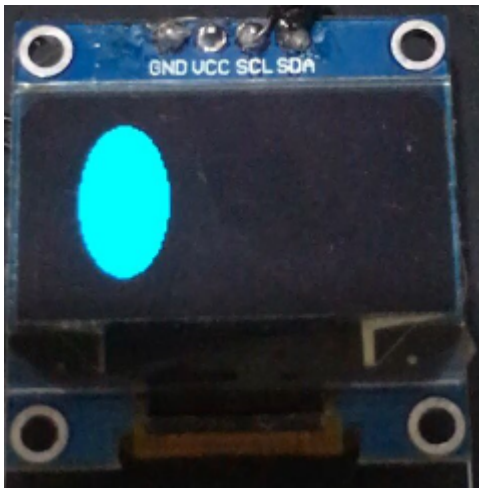
代码中可通过设置canvas_port.lineWidth来调整线宽，线宽的取值范围为[1, 8]。实际显示效果如下：



4.14 填充椭圆

```
canvas = require("canvas");
cfg = {screen_type:2,orientation:1};
canvas_port = canvas.open(cfg);
canvas_port.beginPath();

canvas_port.fillEllipse(10,40,5,60);

canvas_port.stroke();
canvas_port.closePath();
```

实际显示效果如下：



4.15 书写字符串

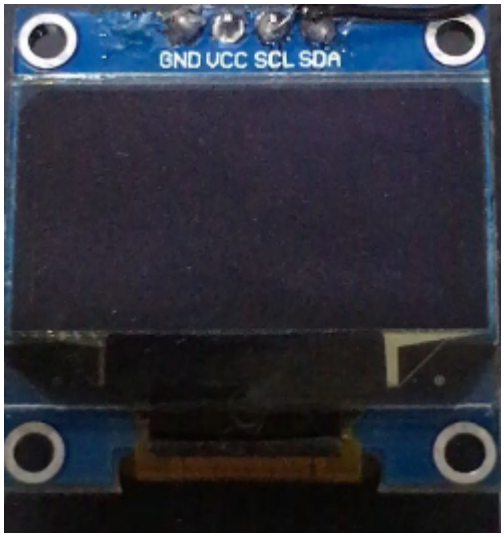
```
canvas = require("canvas");
cfg = {screen_type:2,orientation:1};
canvas_port = canvas.open(cfg);
canvas_port.beginPath();

canvas_port.fillText(0, 0, "Hello World!", "0816");

canvas_port.stroke();
canvas_port.closePath();
```

实际显示效果如下：

4.16 清除屏幕

```
canvas = require("canvas");
cfg = {screen_type:2,orientation:1};
canvas_port = canvas.open(cfg);
canvas_port.beginPath();

canvas_port.arc(30, 30, 20, 0, 360, 0);
canvas_port.clear();

canvas_port.stroke();
canvas_port.closePath();
```

以上代码中，由于在绘制圆形之后又执行了清除屏幕操作，因此屏幕不显示任何图案。实际显示效果如下：


GND VCC SCL SDA