

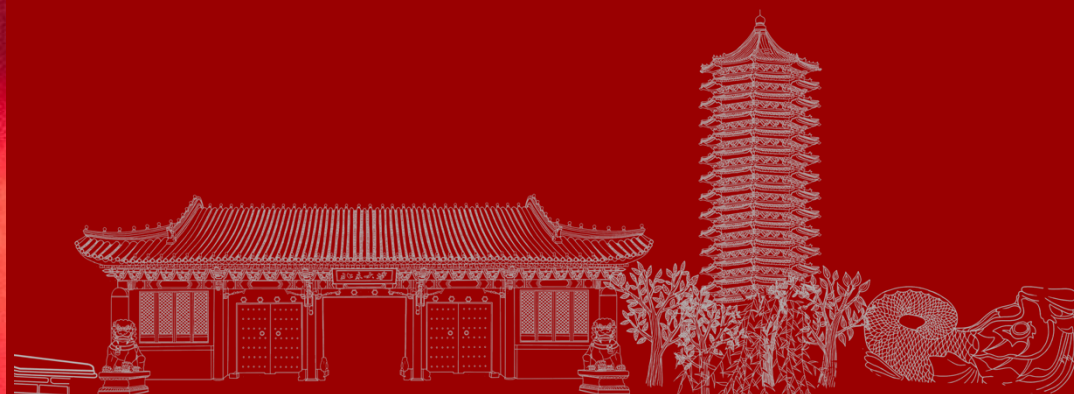
UART与ADC 编程

K210 与 K230 的 UART API
K230 的 ADC API 与运行示例

汇报人

由昕妍

2026年10月14日



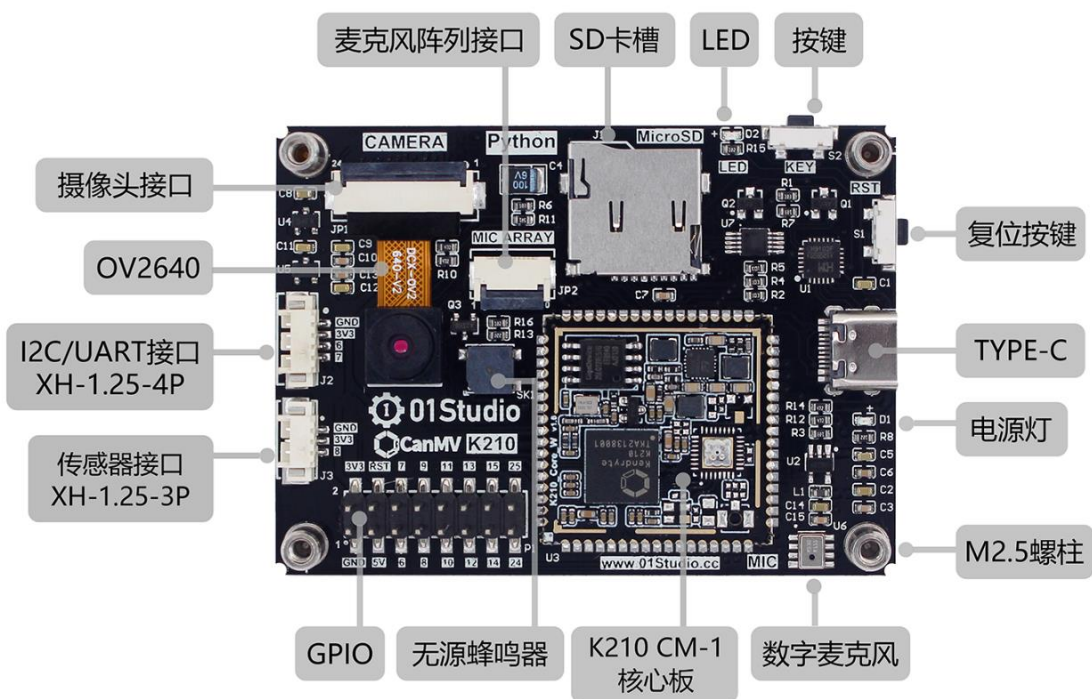
1. UART : K210 与 K230

对比项目	CanMV K210	CanMV K230
芯片 UART 数量	UART1、UART2、UART3, 共 3 个	UART0~UART4, 共 5 个
系统终端占用	参考教程中未特别说明终端占用	UART0 被小核终端占用, UART3 被大核终端占用
用户程序可使用的 UART	UART1、UART2、UART3	UART1、UART2、UART4
板上接线资源	有扩展排针, 也有一个 XH-1.25mm-4P 的 UART/I2C 接口; UART 实验使用扩展排针	通过 40Pin 排针引出 UART1、UART2; UART2 也可通过板背面的四针接口接线

1. UART : K210 与 K230

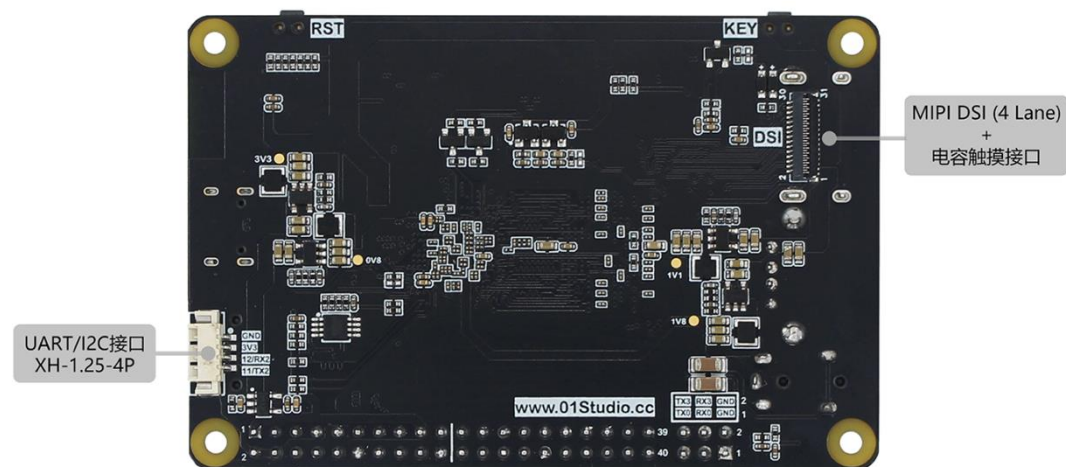
(板上接口)

CanMV K210



扩展排针与 UART/I2C 四针接口

CanMV K230



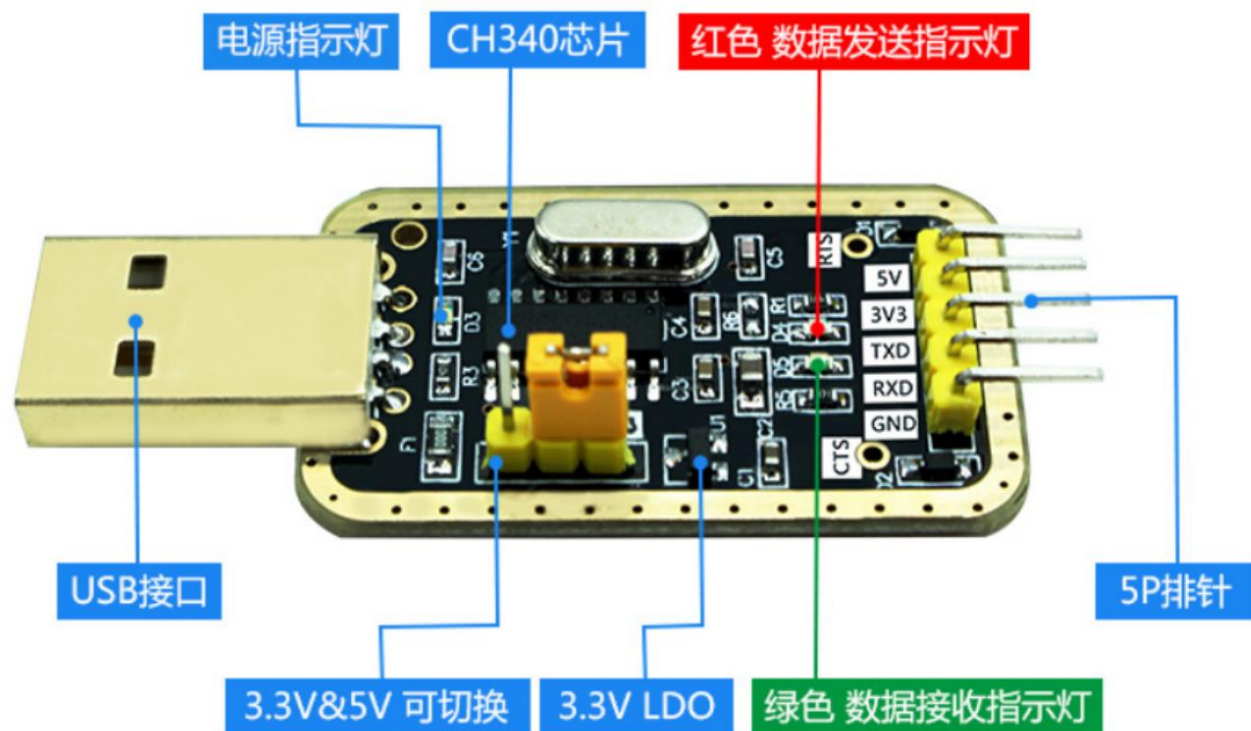
40Pin 排针引出 UART1、UART2
板背面另有 UART/I2C 四针接口

2. UART API：创建、收发与释放

通信双方的波特率、数据位、校验位和停止位应匹配。

API	作用
UART(...)	选择串口并创建对象；常用波特率为 115200 或 9600。
uart.read(num)	从接收缓冲区读取数据；num 表示读取的字节数量。
uart.readline(num)	按两板 01Studio 教程的写法，读取一行数据；下面的收发例程不依赖此方法。
uart.write(buf)	发送数据。
uart.deinit()	用完后注销串口对象。

2. UART API：创建、收发与释放 (接线)



USB 转 TTL 模块

选择 3.3 V 通信电平

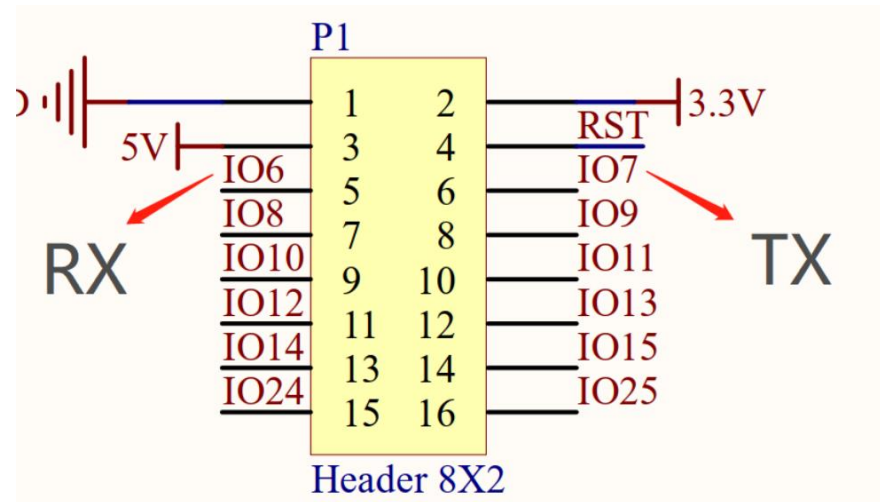
开发板 TX 接模块 RXD
开发板 RX 接模块 TXD
开发板 GND 接模块 GND

TX/RX 交叉连接，双方共地。

2. UART API：创建、收发与释放

(接线对照)

K210: IO7/TX, IO6/RX



K230: GPIO3/TX, GPIO4/RX

GPIO2	W15	GPIO2/JTAG_TCK/PULSE_CNTR0
GPIO3	V15	GPIO3/JTAG_TDI/PULSE_CNTR1/UART1_TXD
GPIO4	Y15	GPIO4/JTAG_TDO/PULSE_CNTR2/UART1_RXD
GPIO5	U14	GPIO5/JTAG_TMS/PULSE_CNTR3/UART2_TXD
GPIO6	V14	GPIO6/JTAG_RST/PULSE_CNTR4/UART2_RXD
CAM2 I2C4 SDA	W14	GPIO7/PWM2/IIC4_SCL
CAM2 I2C4 SCL	U12	GPIO8/PWM3/IIC4_SDA
CAM0 GPIO0	T13	



3. 两板 API 的主要差异 (1/2)

对比项目	CanMV K210	CanMV K230	区别说明
导入 UART	<code>from machine import UART</code>	<code>from machine import UART</code>	基本相同
导入引脚映射模块	<code>from fpioa_manager import fm</code>	<code>from machine import FPIOA</code>	引脚映射 API 完全不同
创建引脚映射对象	不需要	<code>fpioa = FPIOA()</code>	K230 需要先创建 FPIOA 对象
TX 引脚映射	<code>fm.register(7, fm.fpioa.UART1_TX, force=True)</code>	<code>fpioa.set_function(3, FPIOA.UART1_TXD)</code>	K210 使用 UART1_TX, K230 使用 UART1_TXD
RX 引脚映射	<code>fm.register(6, fm.fpioa.UART1_RX, force=True)</code>	<code>fpioa.set_function(4, FPIOA.UART1_RXD)</code>	K210 示例为 IO6/IO7, K230 示例为 GPIO3/GPIO4

3. 两板 API 的主要差异 (2/2)

对比项目	CanMV K210	CanMV K230	区别说明
构造函数	UART(uart, baudrate, bits, parity, stop, timeout, read_buf_len)	UART(id, baudrate=115200, bits=UART.EIGHTBITS, parity=UART.PARITY_NONE, stop=UART.STOPBITS_ONE)	K210 教程明确列出 timeout、read_buf_len; K230 教程使用带默认值的枚举常量
串口编号参数	UART.UART1、UART.UART2、UART.UART3	板上主要使用 UART.UART1、UART.UART2	K230 芯片还有 UART4 可用, 但 01Studio 开发板教程明确引出的用户串口是 UART1、UART2
数据位 bits	通常直接使用数值, 默认 8	默认 UART.EIGHTBITS	含义相同, K230 文档更强调常量写法
校验位 parity	None、UART.PARITY_EVEN、UART.PARITY_ODD	UART.PARITY_NONE、UART.PARITY_EVEN、UART.PARITY_ODD	无校验的推荐写法不同
停止位 stop	教程写作 1、1.5、2	构造函数示例使用 UART.STOPBITS_ONE	K230 建议优先使用文档给出的常量
接收缓冲区	构造函数支持 read_buf_len, 示例为 4096	01Studio 构造函数说明中未列出	不应把 K210 的 read_buf_len 直接迁移到 K230
uart.read([num])	支持; 示例使用 uart.read()	支持; 示例使用 uart.read(128)	不传参数读取当前可用数据; 传参数限制读取量
uart.write(buf)	发送数据	发送数据	用法基本相同
uart.deinit()	注销串口	注销串口	用法基本相同

4. K230 的 ADC

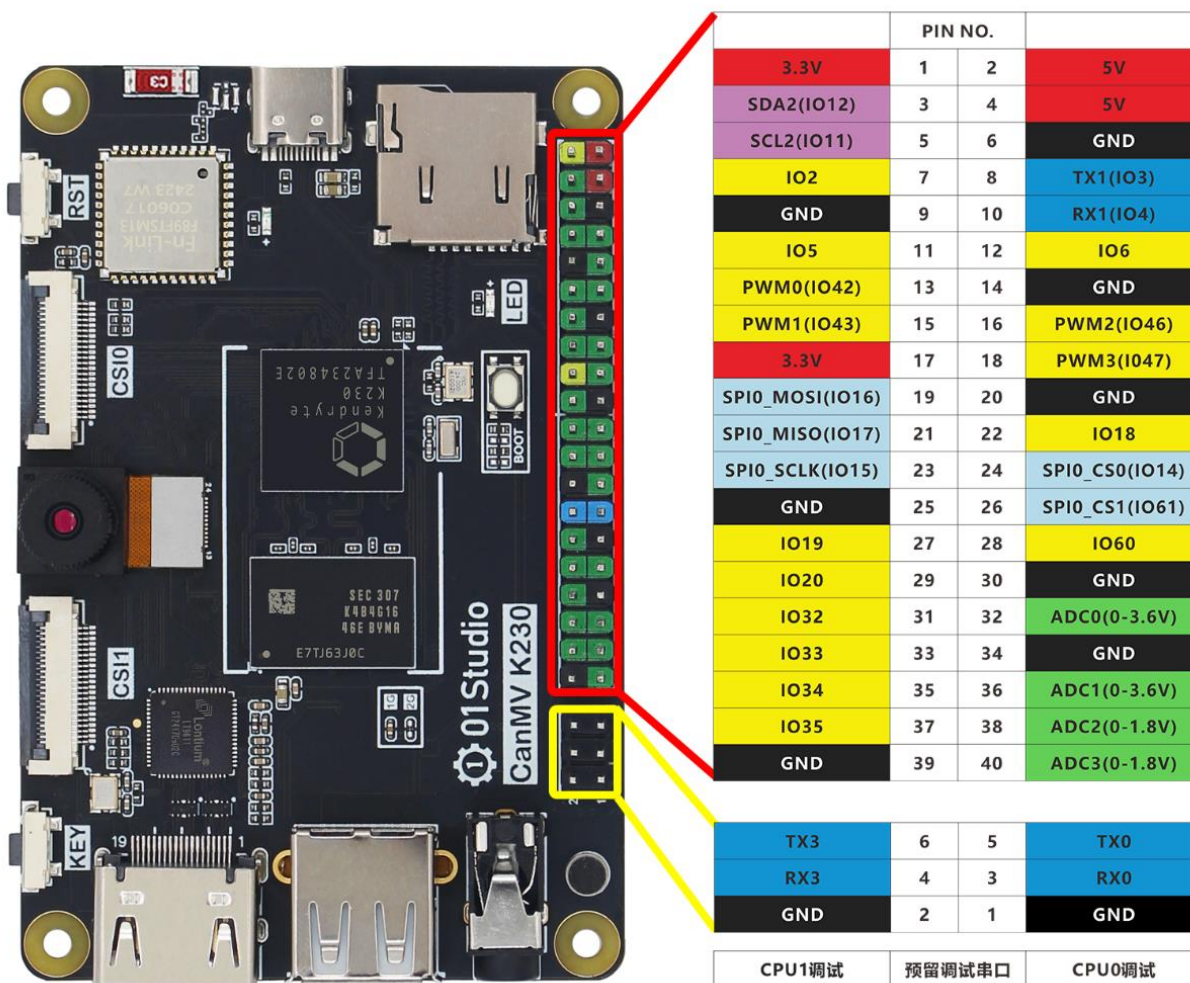
ADC 将模拟电压转换为数字读数，可读取电位器、模拟传感器等设备输出。

项目	K230 ADC 说明
分辨率	12 位，采样值范围为 0 ~ 4095
标准 CanMV K230 板引出通道	ADC0 ~ ADC3
选择通道	使用 ADC(channel) 创建对应通道的 ADC 对象
读取采样值	adc.read_u16() 返回 0 ~ 4095 的采样值
读取电压	adc.read_uv() 返回以微伏为单位的读数
ADC0、ADC1 输入量程	0 ~ 3.6 V；按 01Studio 标准板教程换算板上输入电压时，需将 read_uv() 换算成伏特后再乘 2
ADC2、ADC3 输入量程	0 ~ 1.8 V；换算电压时不需要乘 2
接线注意事项	信号源与开发板共地，输入电压不得超过所选通道的量程

量程及“乘 2”规则适用于 01Studio 标准 CanMV K230 开发板。

4. K230 的 ADC

(引脚位置)



ADC 区域

IO32	31	32	ADC0(0-3.6V)
IO33	33	34	GND
IO34	35	36	ADC1(0-3.6V)
IO35	37	38	ADC2(0-1.8V)
GND	39	40	ADC3(0-1.8V)

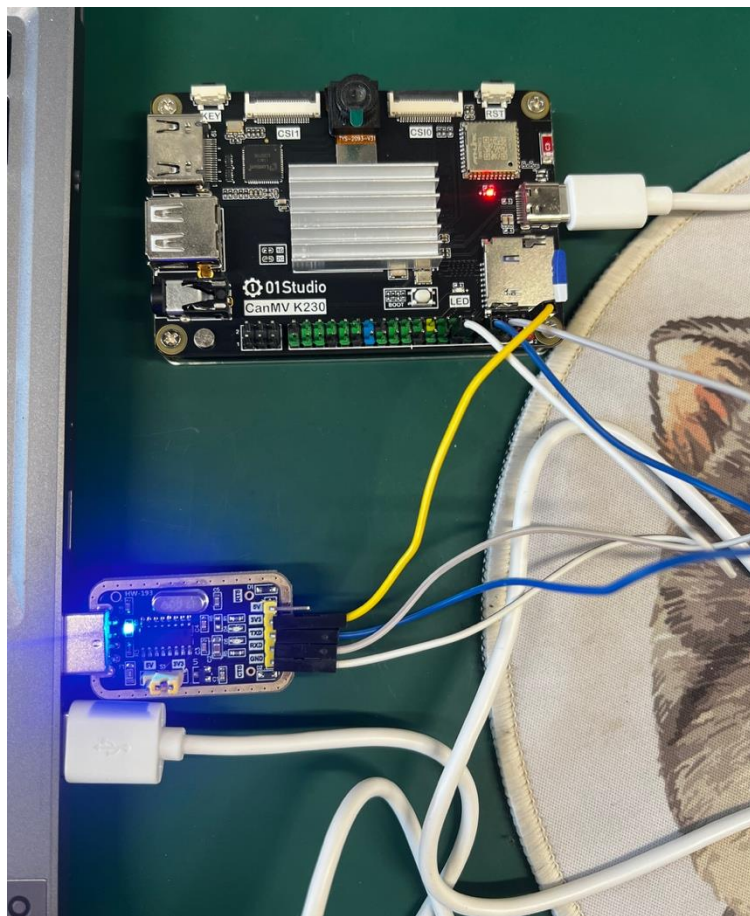
ADC0 / ADC1: 32、36 脚
输入量程 0 ~ 3.6 V

ADC2 / ADC3: 38、40 脚
输入量程 0 ~ 1.8 V

5. UART 运行示例

(接线与启动)

K230 与 USB 转 TTL 实际接线



CanMV 开发板已进入运行状态



终端提示脚本正在运行

```
>>>  
脚本正在运行; 按 Ctrl-C 停止
```

5. UART 运行示例

(例程代码)



```
1 '''
2 实验名称: UART (串口通信)
3 作者: 01Studio
4 实验平台: 01Studio CanMV K230
5 说明: 通过编程实现串口通信, 跟电脑串口助手实现数据收发。
6 '''
7
8 #导入串口模块
9 from machine import UART
10 from machine import FPIOA
11 import time
12
13 fpioa = FPIOA()
14
15 # UART1代码
16 fpioa.set_function(3, FPIOA.UART1_TXD)
17 fpioa.set_function(4, FPIOA.UART1_RXD)
18
19 uart=UART(UART.UART1,115200) #设置串口号1和波特率
20
21 '''
22 # UART2代码
23 fpioa.set_function(11, FPIOA.UART2_TXD)
24 fpioa.set_function(12, FPIOA.UART2_RXD)
25
26 uart=UART(UART.UART2,115200) #设置串口号2和波特率
27 '''
28
29 uart.write('Hello 01Studio!')#发送一条数据
30
31 while True:
32
33     text=uart.read(128) #接收128个字符
34     if text != None:
35         print(text) #通过REPL打印串口3接收的数据
36
37     time.sleep(0.1) #100ms
```

5. UART 运行示例

(开发板发送)



串口监视器波特率：115200

查看模式 文本 端口 COM7 - USB-SERIAL CH340 (COM7) 波特率 115200 行尾 无

串口监视器收到开发板发送的数据

```
Hello 01Studio!
```

5. UART 运行示例

(接收 hello)



串口监视器发送 hello

```
Hello 01Studio!  
---- 已发送 utf8 编码消息: "hello" ----  
|
```

CanMV 终端收到 b'hello'

```
///  
b'hello'  
脚本正在运行; 按 Ctrl-C 停止
```

发送内容与接收内容一致

终端以 bytes 形式显示接收到的数据。

5. UART 运行示例

(接收 123 与 ESP26)

串口监视器发送

123

```
----- 已发送 utf8 编码消息: "123" -----
```

ESP26

```
----- 已发送 utf8 编码消息: "ESP26" -----
```

CanMV 终端接收

b'123'

```
b'123'  
脚本正在运行; 按 Ctrl-C 停止
```

b'ESP26'

```
b'ESP26'  
脚本正在运行; 按 Ctrl-C 停止
```

两组记录均显示: 接收到的数据与发送内容一致。



5. UART 运行示例

(例程代码)

```
1 '''
2 实验名称: ADC (电压测量)
3 版本: v1.0
4 作者: 01Studio
5 实验平台: 01Studio CanMV K230
6 说明: ADC共4个通道, 其中通道0、1实际量程为0-3.6V, 通道2、3量程为0-1.8V。
7       (请勿超出测量量程, 可能导致主控芯片烧坏!)
8 '''
9
10 from machine import ADC
11 import time
12
13 '''
14 构建ADC对象:
15 ADC0 (排针32引脚, 量程0-3.6V), ADC1 (排针36引脚, 量程0-3.6V),
16 ADC2 (排针38引脚, 量程0-1.8V), ADC3 (排针40引脚, 量程0-1.8V)。
17 '''
18 adc = ADC(0) #通道0
19
20 while True:
21
22     print(adc.read_u16()) # 获取ADC通道采样值
23
24     # 获取ADC通道电压值, 保留2为小数。通道0、1实际量程为0-3.6V, 返回值x2。
25     print('%.2f'%(adc.read_uv()/1000000*2), "V")
26
27     time.sleep(1) #延时1秒
28
```

6. ADC 运行示例

(启动)



北京大学
PEKING UNIVERSITY

ADC 例程持续输出采样值与电压

```
>>>
```

```
3
```

```
0.00 V
```

```
3
```

```
0.00 V
```

```
3
```

```
0.00 V
```

```
脚本正在运行; 按 Ctrl-C 停止
```

终端逐次输出:

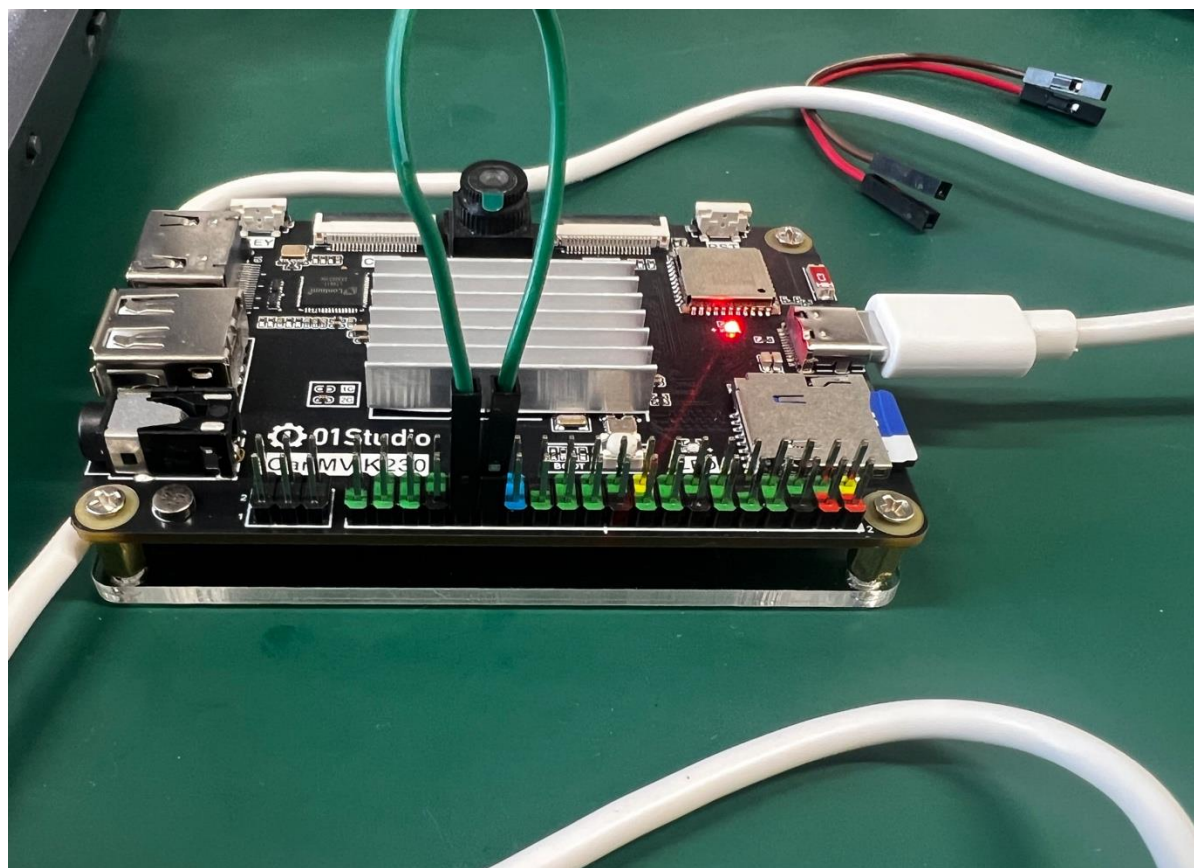
原始采样值
换算后的电压

图中持续显示
3 与 0.00 V

6. ADC 运行示例

(接 GND)

ADC 输入接 GND



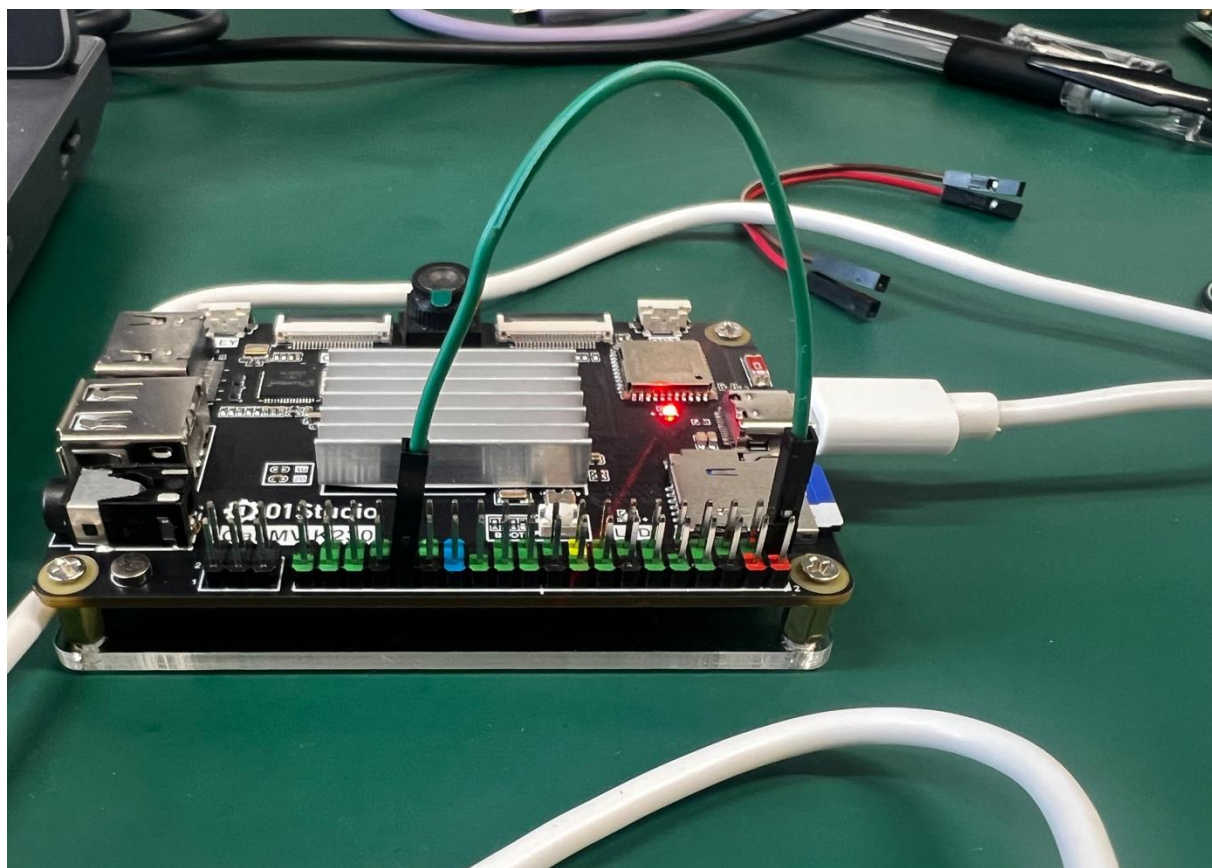
采样值约 3 ~ 4
显示电压 0.00 V

```
3
0.00 V
3
0.00 V
3
0.00 V
4
0.00 V
3
0.00 V
```

6. ADC 运行示例

(接 3.3 V)

ADC 输入接 3.3 V



采样值约 3769 ~ 3780
显示电压约 3.31 ~ 3.32 V

3769
3.32 V
3774
3.32 V
3780
3.32 V
3779
3.32 V
3779
3.31 V
3775
3.32 V