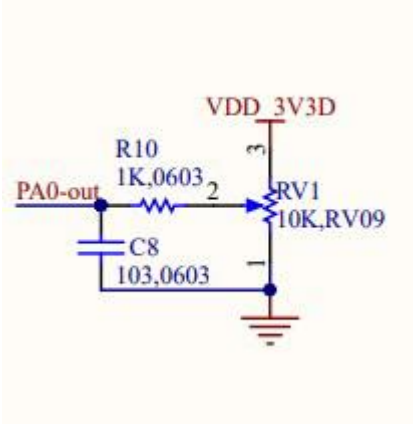
第 14 课，学会使用 ADC 进行输入模拟量采集

本课学习内容：

- 掌握 ADC 的理念
- 学会 ADC 应用用于，通过 REPL 调试进行 ADC 应用和通过 OLED 实施显示 ADC 数据
- 掌握 round（）函数用法 python 通用数据处理函数，四舍五入功能

ADC 是模拟量到数字量转换的英文简写，意思 MCU 有内置的 ADC 模块，可以通过一定引脚接入，只要是设置好对应的引脚就可以，测试外界的一定范围的电压值，STM32 测试电压范围为 0~3.3V,ADC 对应的引脚可以认为是高阻特性，可以认为电流不流入 MCU 内。我们看一下底板原理图，是哪一引脚接入的 ADC.



我们看到是 PA0 这个引脚接入了测试 ADC 的电位器，改变电位器的抽头位置，就可以改变 PA0 位置的电压值，范围是 0~3.3V。

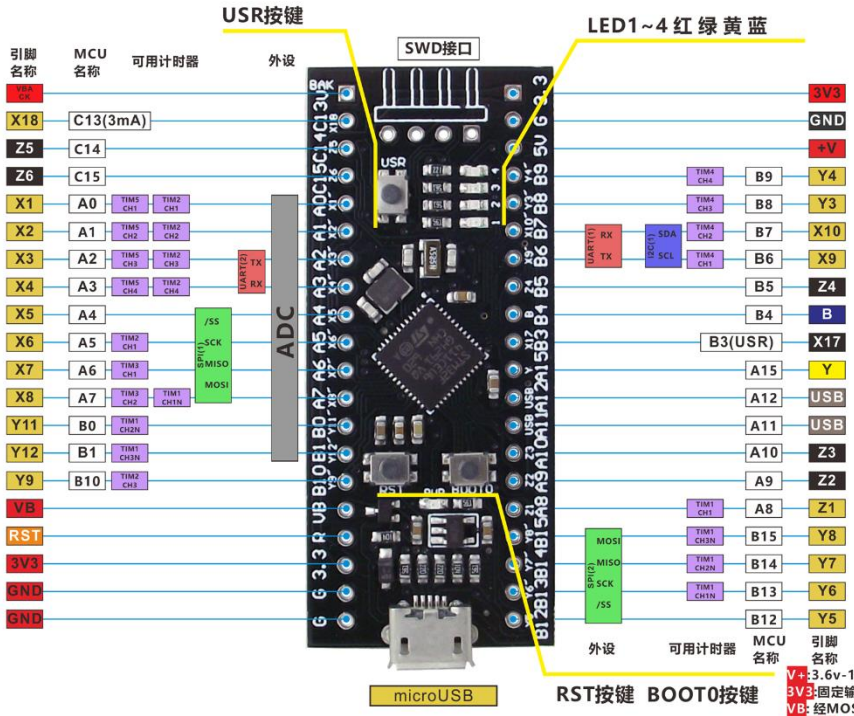
同时根据 Pyboard mini 引脚示意图，我们编写程序也可以使用“X1”名称在程序中代指该引脚。我们直接使用 Pyboard-tool 软件直接进去 REPL 交互测试。

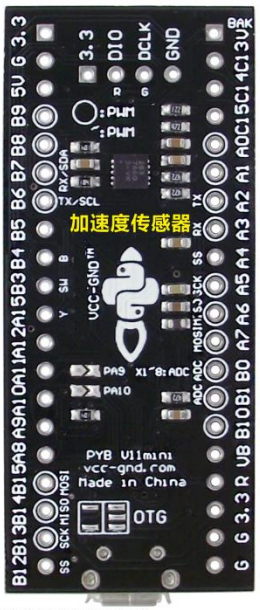
REPL 测试 ADC 语句为：

```
from pyb import Pin, ADC #引用 ADC 和 PIN 库

adc = ADC(Pin('X1'))

adc.read()
```



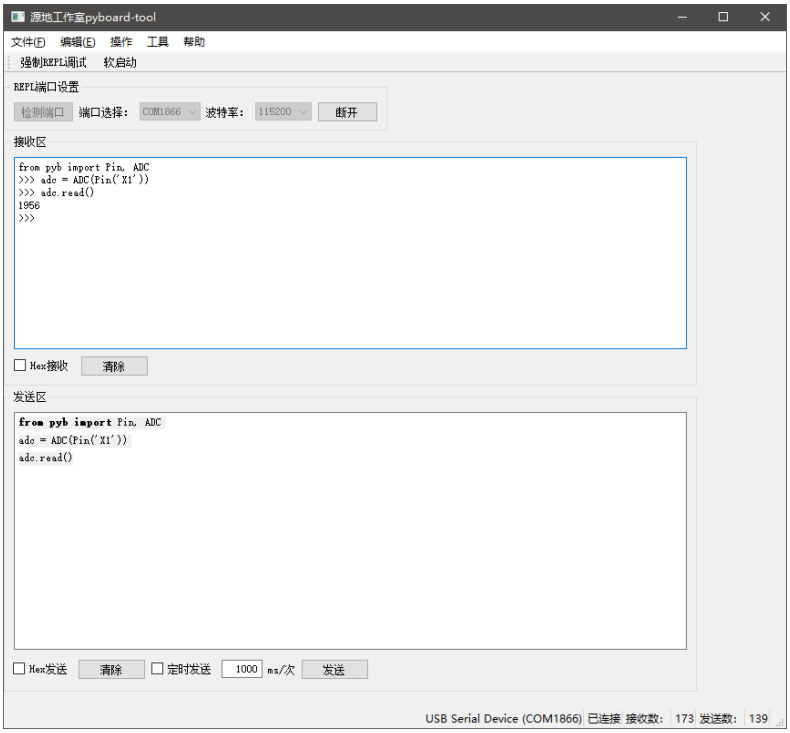


μPython™ uPython™ pyboard™
源地® VCC-GND®

源地工作室

PYBV1.1mini

V+:3.6v-12v 直流电源输入端(当USB连接时由USB供电).
3V3:固定输出3.3V电源端,最大输出电流不要超过300mA.
VB:经MOS管保护的电池电源输入端.
VBACK:备份电源输入端.B6(X9),B7(X10)连接到了加速度传感器I2C引脚上,慎用.B2连接到了加速度传感器的中断输出引脚,故没有引出.
B:B4被板载LED4蓝灯使用,慎用. Y:A15被板载LED3黄灯使用,慎用.
USB:A11,A12外接USB DM和DP,虽然引出引脚，只能在不使用USB功能时使用.
Z2:A9,被用作USB插入检查，当不用系统时，背后焊盘断开可用.
Z3:A10,被用作USB的ID检查，当不用系统时，背后焊盘断开可用.
Z4:B5为加速度传感器AVDD引脚供电使用，慎用.
Z5:C14连接到实时时钟晶振上，平时慎用.Z6:C15连接到实时时钟晶振上，平时慎用.
X17:B3被USR按键功能使用，慎用.
PWR:当板子通电后PWR电源指示灯亮起，方便在电路故障（负载短路）后指示供电情况.
BOOT0:按住BOOT0按键再按下RST按键将可以方便进入DFU状态，方便刷固件.
SWD:接口可以连接使用调试器例如J-link或ST-Link，进行常规C语言开发.



在接收框中，我们看到了 1956 这个数值，我们连解释一下，我们 MCU 中的 ADC 是 12 位的我们算一下 12 位二进制最大是多少。

最大是十进制的 4095，也就是说模拟量中的 0~3.3V 对应这个数值的 0~4095，这次我们测试 ADC 数值为 1956，我们可以反推出实际测试到点电压值=3.3/4095*1956=1.57V。可以尝试一下转动点位上的旋钮，改变一下电压值，重新测量一下，也是按照上述的方法进行计算。

还记得前几节课中的 OLED 显示吗我们可以，利用 OLED 显示，实时显示当前测试到模拟量。

脚本代码为：

```
from pyb import Pin, ADC#引用引脚库和 ADC 库

from ssd1306 import SSD1306#引用 SSD1306

import _thread,time#引用线程和时间库

from machine import Pin#引用引脚库


adc = ADC(Pin('PA0'))#ADC 赋给 abc

display = SSD1306(pinout={'sda': 'PB7','scl':
'PB6'},height=64,external_vcc=False)#OLED 初始化


def funcA(sec):#线程 A

  time.sleep(sec)#线程开始延时 sec

  display.poweron()#线程开启
```

display.init_display()#显示初始化

display.draw_text(1,1,'ADC')#显示内容 1，先写入缓存

display.draw_text(1,14,'ADCvol')#显示内容 2，先写入缓存

while True:#线程中的死循环

display.draw_text(55,1,str(adc.read()))#显示项目 1 数据

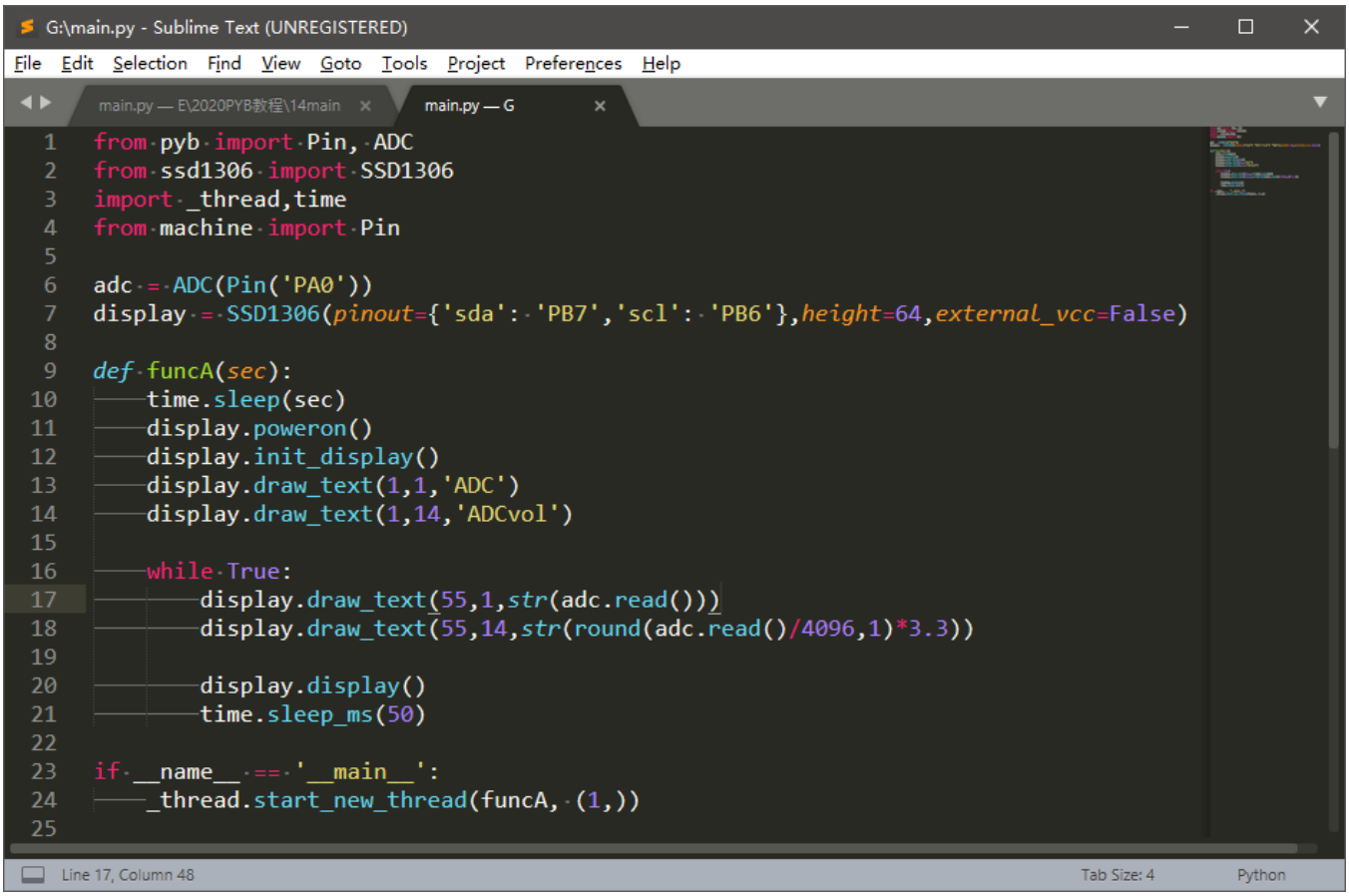
display.draw_text(55,14,str(round(adc.read()/4096,1)*3.3))#显示项目 2 数据，注意这里有个 Python 通用的 round（）用过

display.display()#显示操作

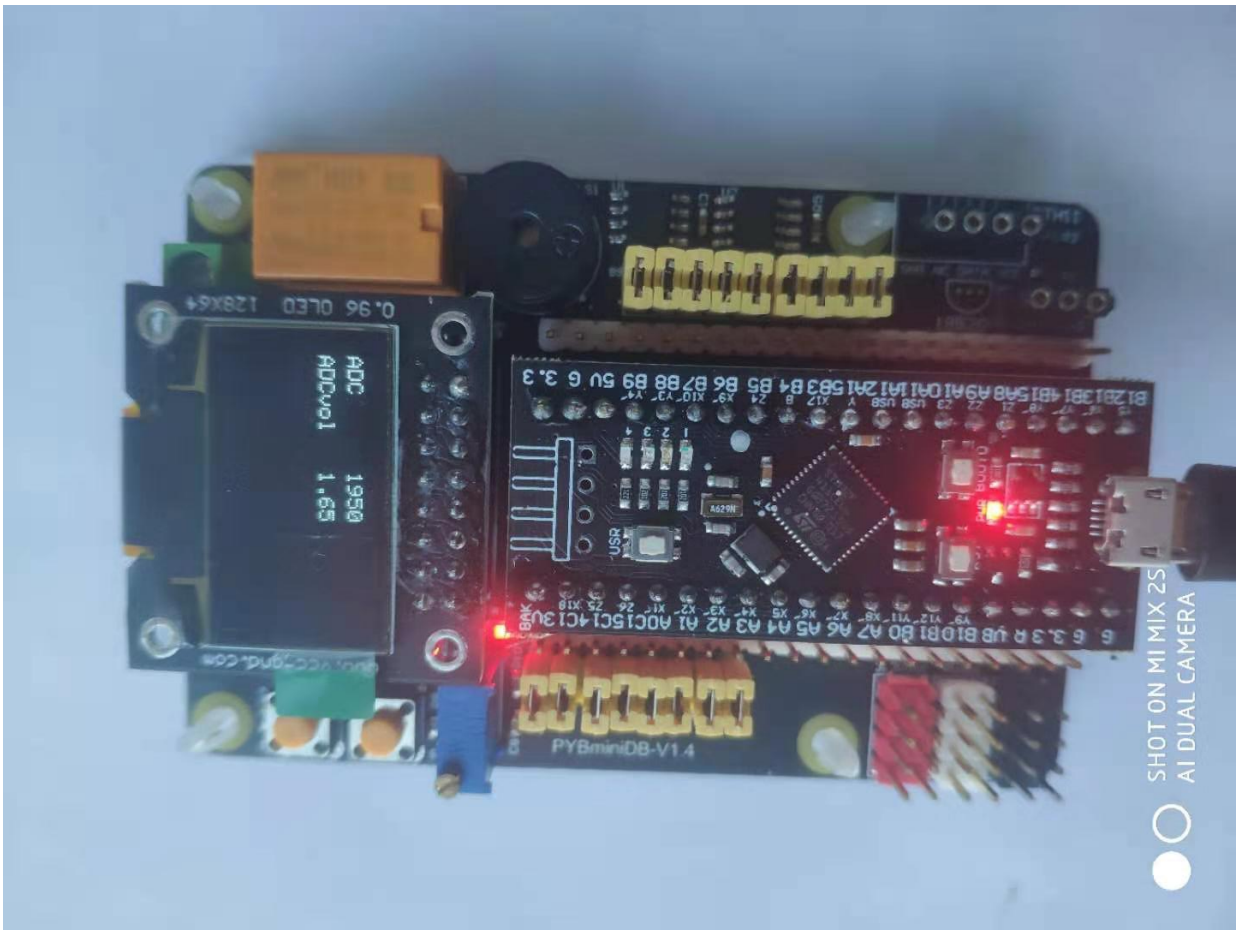
time.sleep_ms(50)#延时 50ms 刷新

if __name__ == '__main__':#程序入口

_thread.start_new_thread(funcA, (1,))#开启线程 A



在 pyboard 板子生成的可移动硬盘中保存好，还是不厌其烦的说一下，在执行保存动作时候，一定等 LED1 熄灭，才算是完成保存动作。然后我们连接 Pyboard-tool 软件，点击**“软启动”**，观察程序表现，我们看到 ADC 实时数据和计算后的数据。



这里的脚本函数用的了 Python round() 函数。

功能：round() 方法返回浮点数 x 的四舍五入值。

语法：以下是 round() 方法的语法:

round(x [, n])

参数：

x -- 数值表达式。

n -- 数值表达式，表示从小数点位数。

返回值：

返回浮点数 x 的四舍五入值。

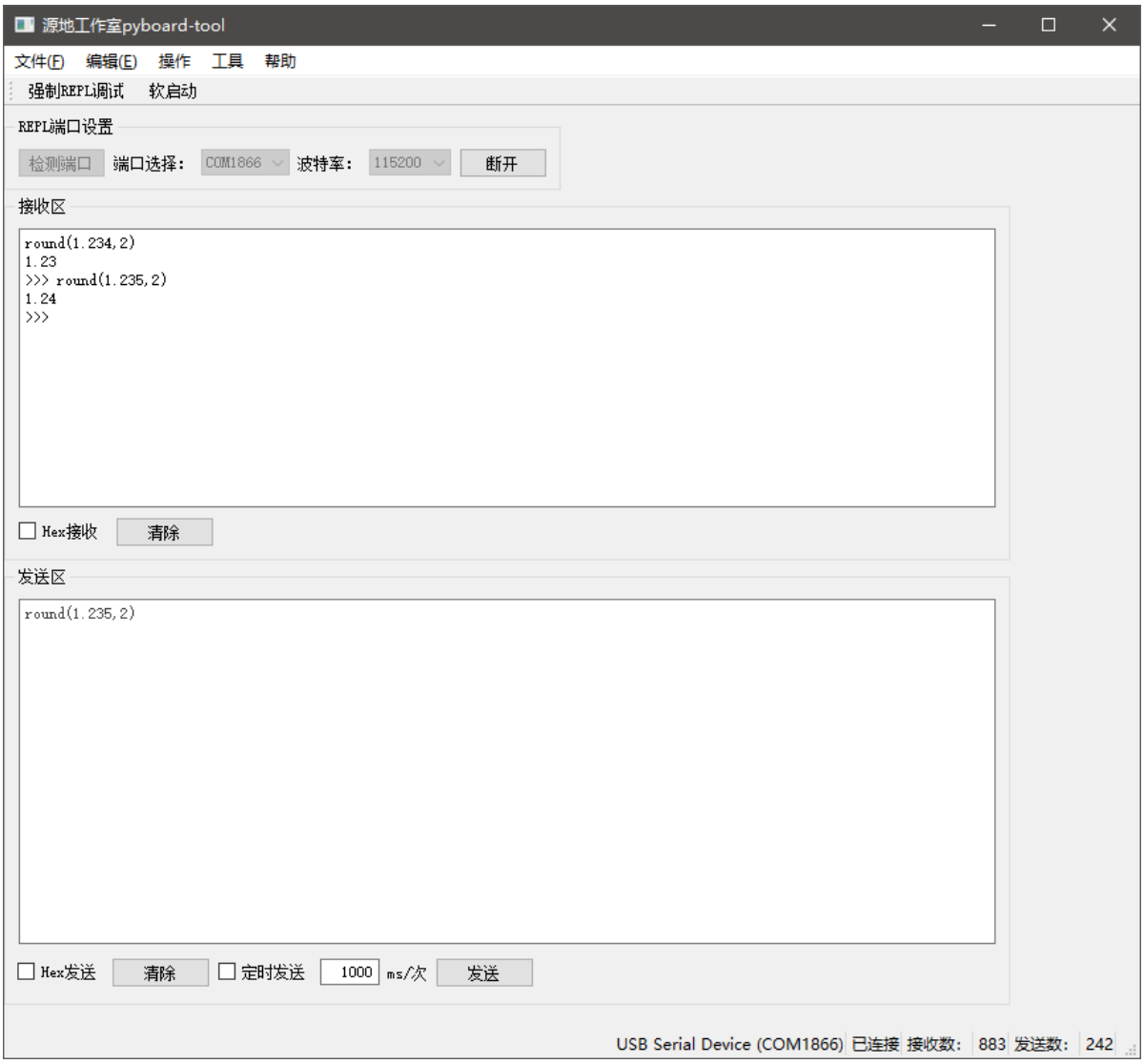
以下展示了使用 round() 方法的实例：

round(80.23456, 2) : 80.23

round(100.000056, 3) : 100.0

round(-100.000056, 3) : -100.0

其实验证这些功能函数的方法也很简单，直接可以使用 REPL 进行就可以验证。



在接收区我们可以看到测试结果。

其实 ADC 使用非常广泛，ADC 也可以测试 MCU 内部电压，和 MCU 内部的温度同时我们也是总结一下 ADC 用法：

```
import pyb


adc = pyb.ADC(Pin("Y12"))
```

```
    adc.read()


    adc = pyb.ADCAll(16)        # create an ADCAll object
    adc = pyb.ADCAll(resolution, mask)  # create an ADCAll object for selected
analog channels
    val = adc.read_channel(channel)     # read the given channel
    val = adc.read_core_temp()          # read MCU temperature
    val = adc.read_core_vbat()          # read MCU VBAT
    val = adc.read_core_vref()          # read MCU VREF
    val = adc.read_vref()               # read MCU supply voltage
    #读一下核心芯片的温度
    import pyb
    adc = pyb.ADCAll(12)
    adc.read_core_temp()
    #读取后备电池的电压
    import pyb
    adc = pyb.ADCAll(12)
    adc.read_core_vbat()
    #读取参考核心电压数值
    import pyb
    adc = pyb.ADCAll(12)
    adc.read_core_vref()
    #读取供电电压
    import pyb
    adc = pyb.ADCAll(12)
    adc.read_vref()
```

#定时读取一个 ADC 值然后存储到一个列表中

```
adc = pyb.ADC(pyb.Pin.board.X19)     # create an ADC on pin X19
tim = pyb.Timer(6, freq=10)          # create a timer running at 10Hz
buf = bytearray(100)                 # creat a buffer to store the samples
adc.read_timed(buf, tim)             # sample 100 values, taking 10s
```