实验 3 【模数转换】

指导教师： 王海鹏

学生姓名： 蔡亦扬 学生姓名： 周训哲 学生姓名： 孔恩燊

学号：23307130258 学号：20307110315 学号：23307130021

专业：技术科学试验班 专业：计算机 专业：技术科学试验班

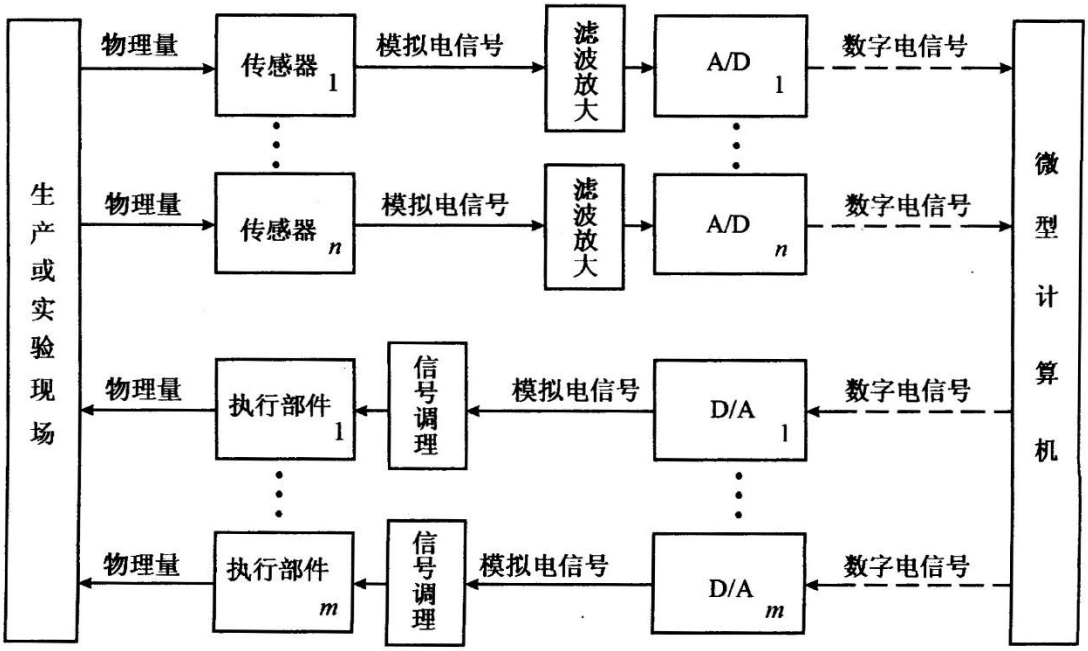
日 期： 2024.4.9

一、 实验目的：

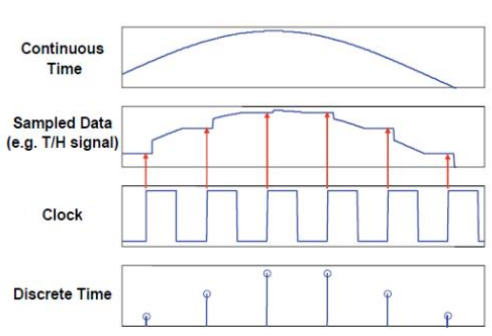
1. 了解模数转换的基本概念
2. 初步了解信号发生器、示波器的使用
3. 以ADC0832为例，掌握ADC的基本使用方法

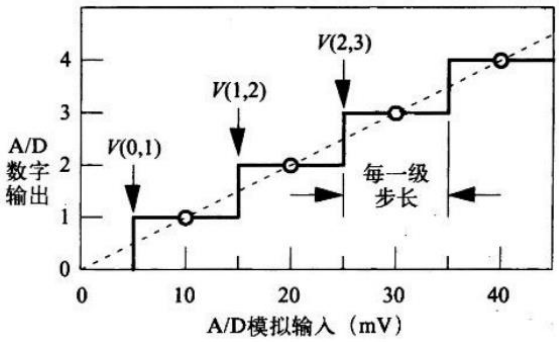
二、 实验原理：

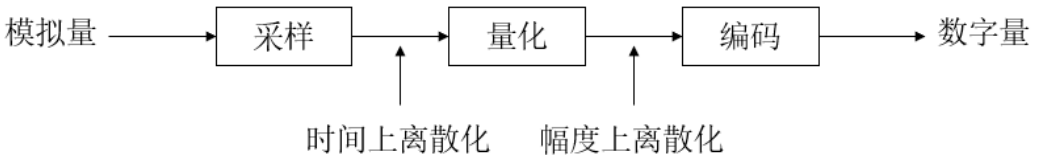
1. 模数转换：



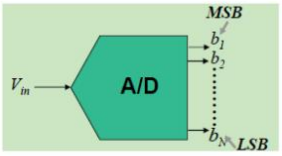
（1）模拟信号转换为数字信号,一般分为四个步骤进行,即取样、保持、量化和编码。前两个步骤在取样-保持电路中完成,后两步骤则在 ADC 中完成，量化的过程是时间上的离散化，编码的过程是幅度的离散化。







（2）将模拟电压量转换成离散数值，通常用二进制或 16 进制表示。 D= Vin/Δ



$N = \#\ of\ bits$

$V_{FS} = full\ scale\ output$

$\Delta = min.\ resolvable\ input \rightarrow 1LSB$

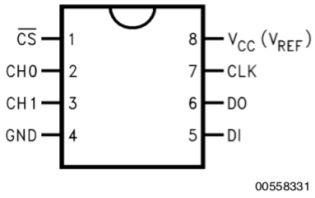
$$\Delta = \frac{V_{FS}}{2^N}$$

$$or\ N = log_2 \frac{V_{FS}}{\Delta} \rightarrow resolution$$

2. ADC0832的简介：

Top View

ADC0832 2-Channel MUX
Dual-In-Line Package (N)



00558331

COM internally connected to GND.
V_{REF} internally connected to V_{CC}.
Top View

（1）芯片接口说明：

- -CS 片选使能，低电平芯片使能。
- CH0 模拟输入通道0，或作为IN+/-使用。
- CH1 模拟输入通道1，或作为IN+/-使用。
- GND 芯片参考0 电位（地）。
- DI 数据信号输入，选择通道控制。
- DO 数据信号输出，转换数据输出。
- CLK 芯片时钟输入。
- Vcc/REF 电源输入及参考电压输入（复用）。

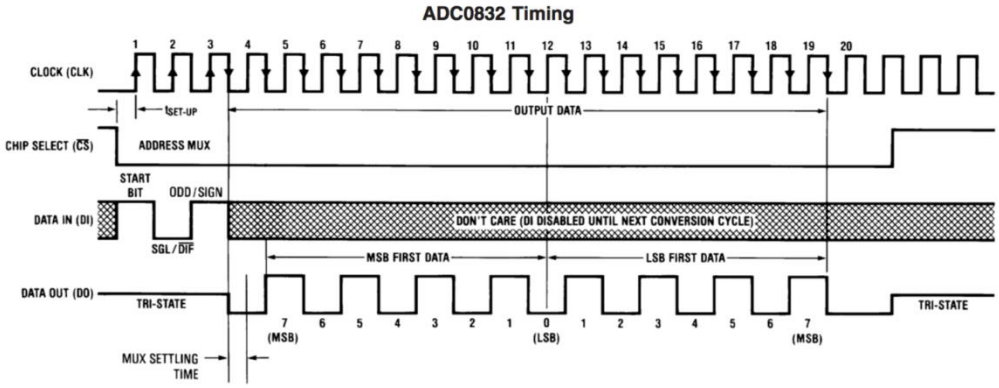
（2）参考电压：用作模拟信号输入电压的最大参考值。若输入参考电压+3V，则认为+3V为模拟信号的最高电压，将0～ +3V等分为256级，每一级对应0～255的一个数字信号从DO输出。如：输入的模拟信号大小为+1.5V ，则输出127。

（3）特点：输入输出电平与 TTL/CMOS 相兼容；5V 电源供电时输入电压在 0~5V 之间；该 ADC 的满量程参考电压直接设置为电源电压，本实验中用 GPIO3.3V 供电，输入电压必须在 0~3.3V 之间；工作频率为 250KHz，转换时间为 32μS；一般功耗为 15mW；8P、14P—DIP（双列直插）、PICC 多种封装；商用级芯片温宽为 0℃ to +70℃ ，工业级芯片温宽为-40℃ to +125℃。

（4）应用接口与工作时序：① 正常情况下 ADC0832 与单片机的接口应为 4 条数据线， 分别是-CS、CLK、DO、DI。但由于 D0 端与 DI 端在通信时并未同时有效并与树莓派的接口是双向的，所以电路设计时可以将 D0 和 DI 并联在一根数据线 上使用。当-

CS 输入端高电平时芯片禁用。当要进行 A/D 转换时，须先将-CS 置于低电平并且保持低电平直到转换完全结束。 ② 在第 1 个时钟脉冲的下降沿之前 Dl 端必须是高电平，表示启始信号。在第 2、 3 个脉冲下降沿之前 DI 端应输入 2 位数据用于选择模拟信号通道。

Timing Diagrams (Continued)



（5）工作原理：

如表 1 所示，当这两位数据为“1”、“0”时，设置 CH0 为单端输入；当 2 位数据为“1”、“1”时，设置 CH1 为单端输入；当 2 位数据为“0”、“0”时，将 CH0 作为正输入端 IN+，CH1 作为负输入端 IN-进行差分输入；当 2 位数据为“0”、“1”时，将 CH0 作为负输入端 IN-，CH1 作为正输入端 IN+进行差分输入。

TABLE 6. MUX Addressing: ADC0832
Single-Ended MUX Mode

MUX Address		Channel #	
SGL/$\overline{\text{DIF}}$	ODD/SIGN	0	1
1	0	+	
1	1		+

COM is internally tied to A GND

TABLE 7. MUX Addressing: ADC0832
Differential MUX Mode

MUX Address		Channel #	
SGL/$\overline{\text{DIF}}$	ODD/SIGN	0	1
0	0	+	−
0	1	−	+

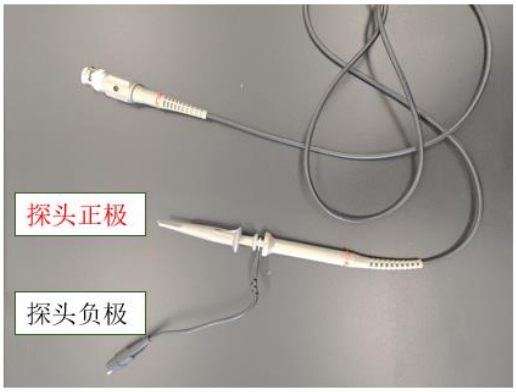
到第 3 个脉冲的下降沿之后输出 DO 进行转换数据的读取。从第 4 个脉冲下降沿开始由 DO 端输出转换数据最高位 DATA7，随后在每一个脉冲的下降沿 DO 端输出下一位数据，直到第 11 个脉冲时发出最低位数据 DATA0。

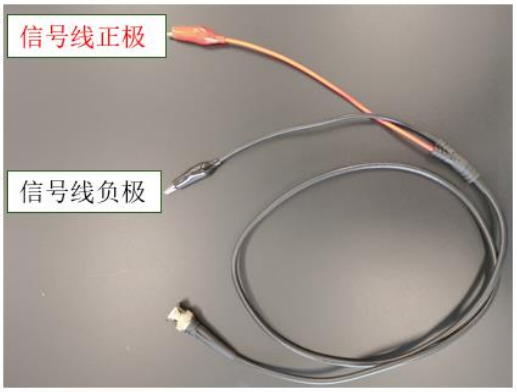
紧接着，从此位开始输出一个相反顺序的数据，即从第 11 个字节的下降沿输出 DATD0，随后输出 8 位数据，到第 19 个脉冲时数据输出完成。

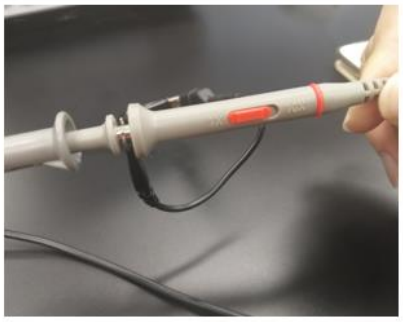
三、 实验内容：

实验 1.： 树莓派控制 ADC0832 测量电位器电压

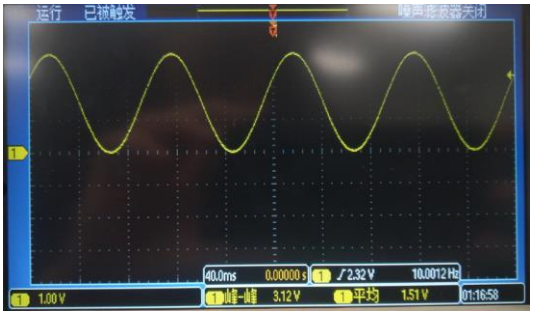
1. 仪器使用：连接并配置任意波形信号发生器。用示波器采样，根据探头实际参数设置，调节信号发生器电压值与直流偏置，使电压值在 0~3.3V 之间，示例中取频率为 10Hz。注意示波器采样时要在非自动模式下，手动设置直流耦合。





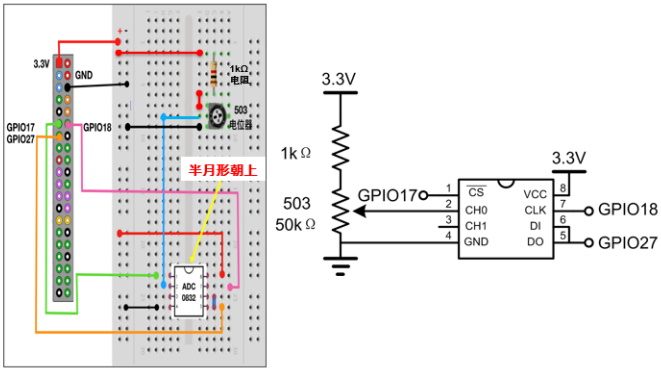


确认衰减开关为1x 还是 10x



2. 电路连接：

示例中使用 503 电位器，也可用滑动变阻器代替。电位器一端连接到 3.3V，另一端接地，这就允许电位器分压输出 0 到 3.3V 之间的电压。

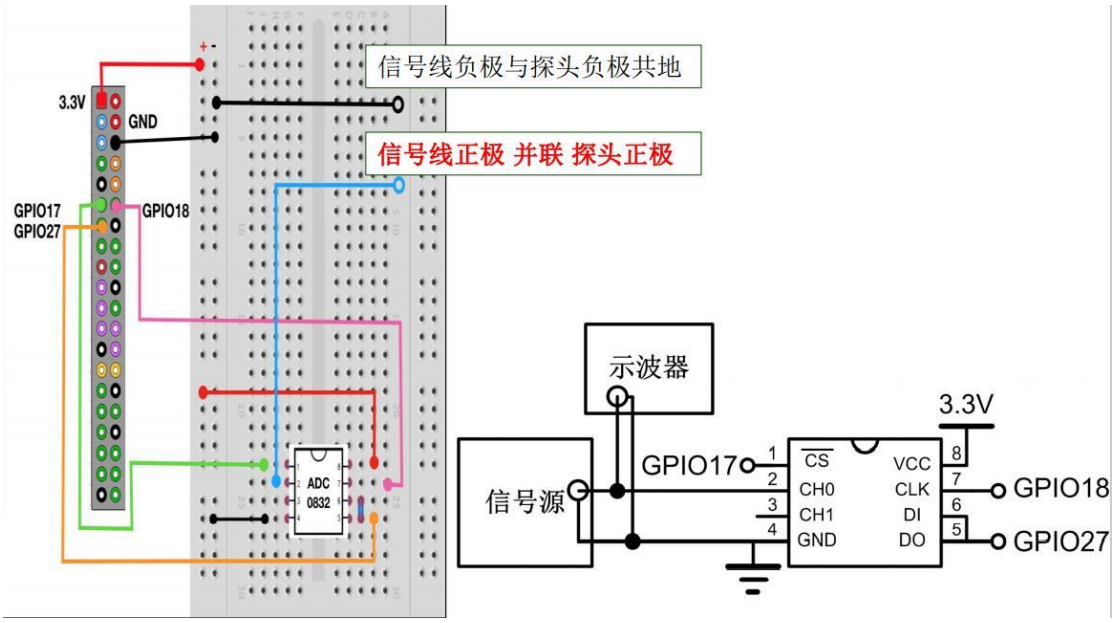


3. 写入代码：

见 measur_V.py，在终端中运行，改变电位器电阻值，即可测得相应的电压。

实验 2：树莓派控制 ADC0832 测量正弦电压

1. 电路连接

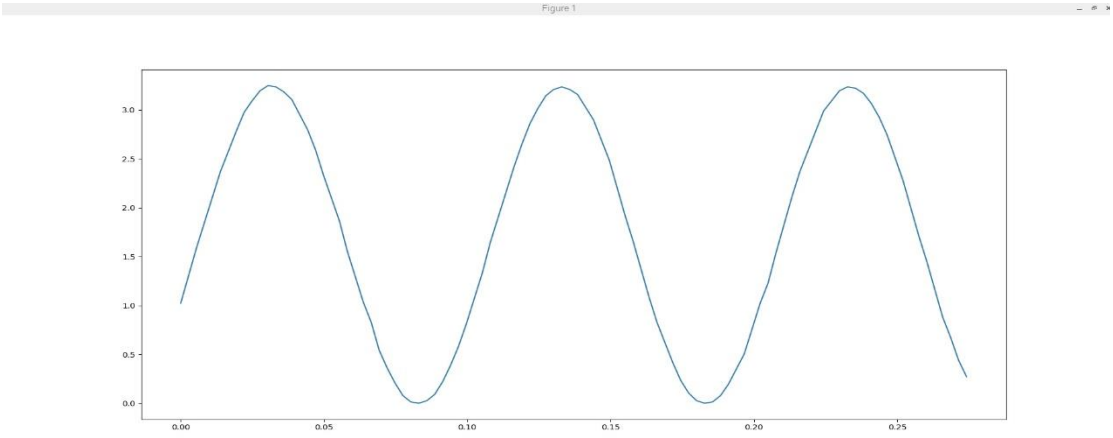


2. 写入代码：

见 measur_sin.py，在终端中运行。可根据需要在循环中加入 time.sleep（）函数或调整取点个数注意每次运行程序后要及时关闭图像窗口

注意：sleep 函数只有毫秒级精度（Linux OS 进程调度所致），因此无法保证精确定时采样。可用 wiringpi 中的 delayMicroSeconds 函数提高精度(10us 级)

3. 基于 matplotlib 的简易图像处理



见 measure_plt.py，在终端中运行。

控制坐标轴范围：通过 plt.axis()、plt.ylim()、plt.xlim()函数可以控制坐标轴的范围从而局部放大函数图象；

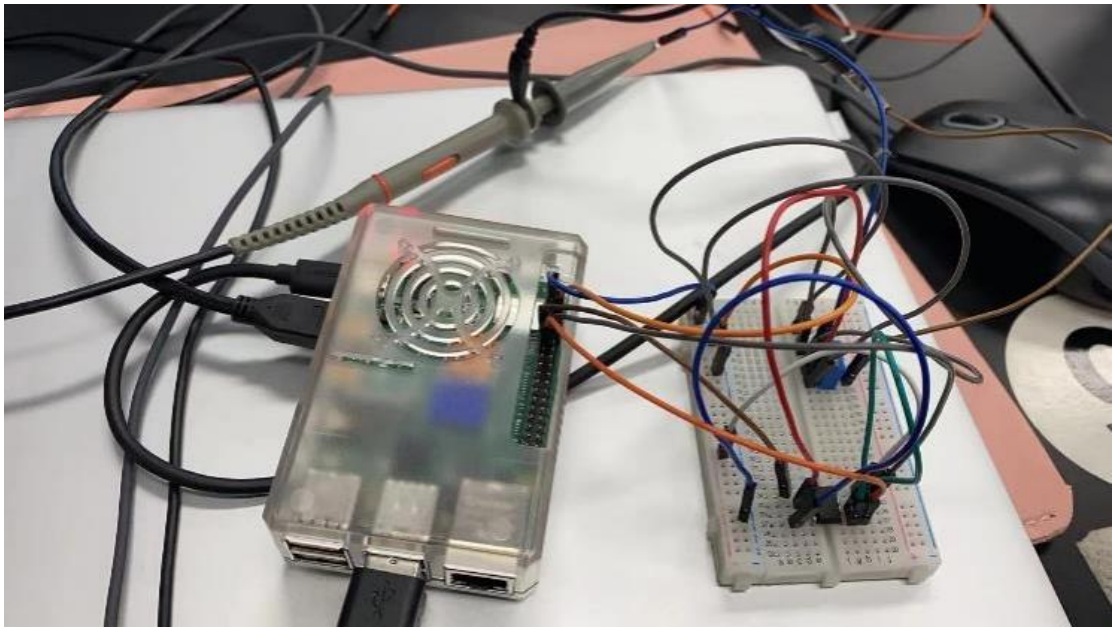
显示坐标点：plt.plot(x,y,’o’)可显示数据散点，plt.plot(x,y,’-o’)可将获得带数据点的折线

图（也可将‘o’换为‘*’等）；
注释点的坐标：通过 plt.text()函数为指定位置加上文本注释，示例中为十个点添加注释；
更多的图像处理方法参考 matplotlib.org

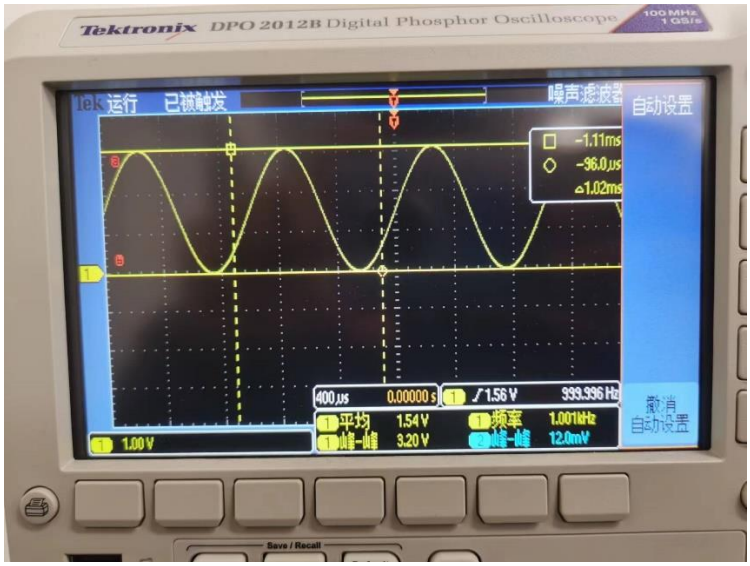
四、 实验分析：

实验 1：树莓派控制 ADC0832 测量电位器电压：

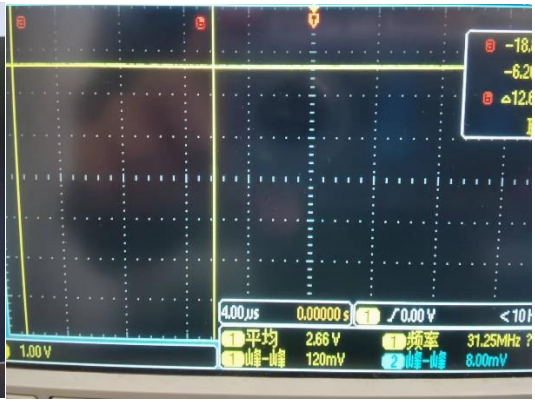
实验前已经安装好了 Matplotlib、NumPy 和 wiringpi。在连接电路时，第一次只是按图上所示图连接，虽然可以通过螺丝刀拧电位器调整阻值，在树莓派上面读出电压值，却无法在示波器上读出电压值。经过询问助教后，在原来的基础上将探头的正负极也分别与树莓派上的杜邦线相连。如图：



（该实验中，电位器相当于滑动变阻器，而 1k 欧姆的电阻作用是保护电路，防止在电位器的阻值过小导致回路中的电流过大而损坏电路的情况。在 ADC0832 中，我们将 DI DO 并联在一起，并且使用 CH0 通道。）
可以在示波器上读出电压均值。



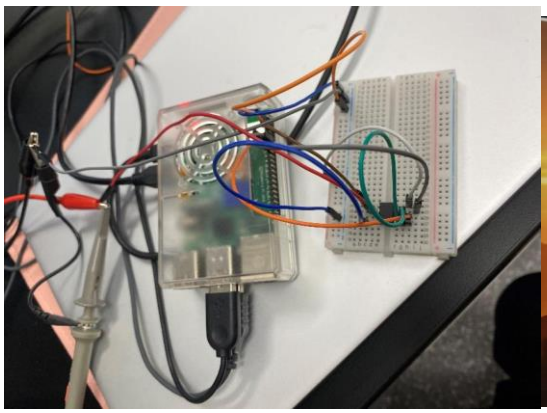
改变电位器阻值读出树莓派和示波器上的电压值，本实验进行了多组并进行误差分析。在测量过程中树莓派给出的电压精度最高的是小数点后 16 位，读数时保留了两位小数。如右上图所示：示波器所测的值一般大于等于树莓派上显示的值，且示波器的示数为真实值，更为准确。两者的误差最大达到了 0.02V.

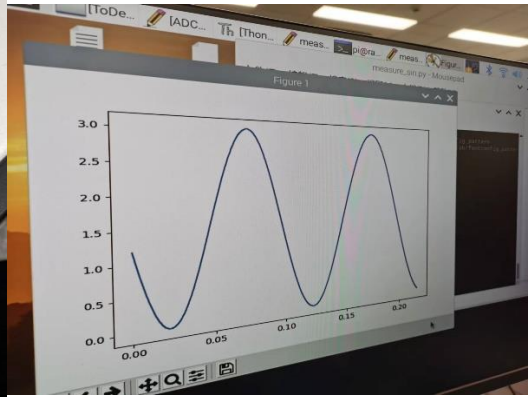


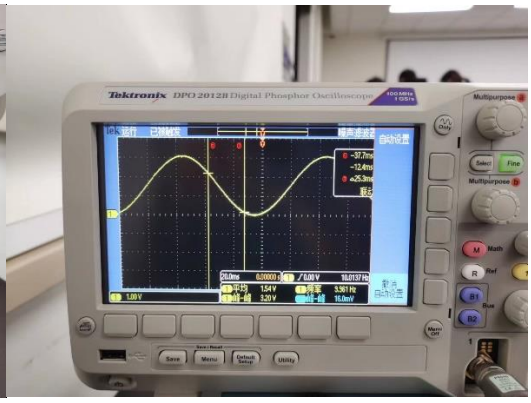
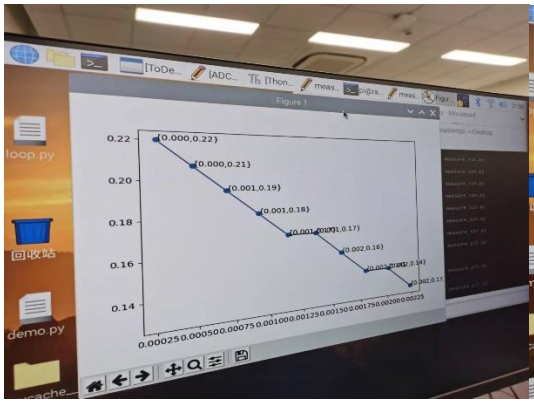
树莓派	示波器
2.16	2.17
2.2	2.21
2.25	2.27
2.34	2.36
2.42	2.44
2.47	2.48
2.52	2.54
2.59	2.6
2.64	2.66
2.7	2.71
2.77	2.78
2.82	2.83
2.87	2.88
2.91	2.93
3.04	3.04
3.08	3.09
3.13	3.15

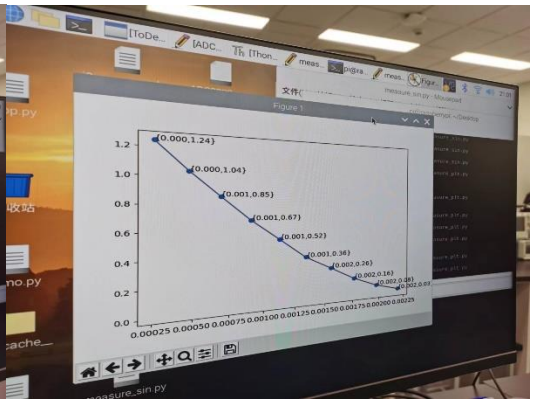
实验二：树莓派控制 ADC0832 测量正弦电压：

本实验中的 3.3V 接口的作用与实验一中的有所不同，共同发挥着驱动 ADC0832 的作用，但是实验一中多了承担闭合回路（含有电阻以及电位器）的电源的作用。同时在实验中发现，改变频率后会使正弦图像的周期发生变化。频率越高，周期越短，看起来波形就更密集。







代码运行后出现正弦波图像，接着利用 matplotlib 绘图得，如图上所示。

五、 总结与思考：

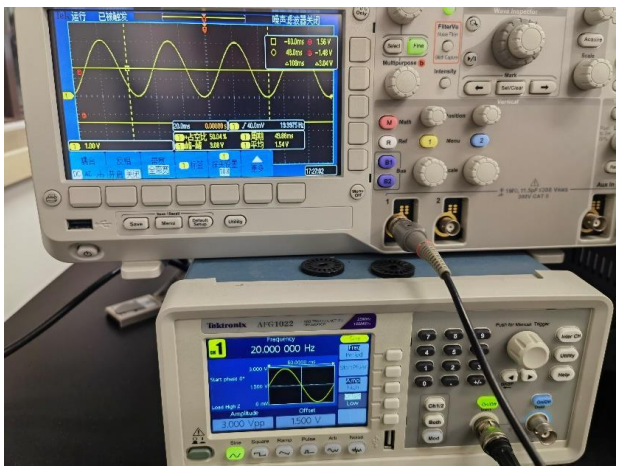
总结：

本次实验实现了模拟信号转换为数字信号。我们学习了器件 ADC0832 的工作原理并且利用 ADC0832 通过模数转换完成对电位器电压的采集，用树莓派控制 ADC0832 完成对正弦波的采集。此外，这次实验接触到了两个新仪器——示波器和信号发生器，简单学习了仪器的使用方法，同时信号发生器在实验二中充当电源，示波器与电路相连可以准确读出电压，与树莓派所测相比分析误差，引起我们的思考分析。

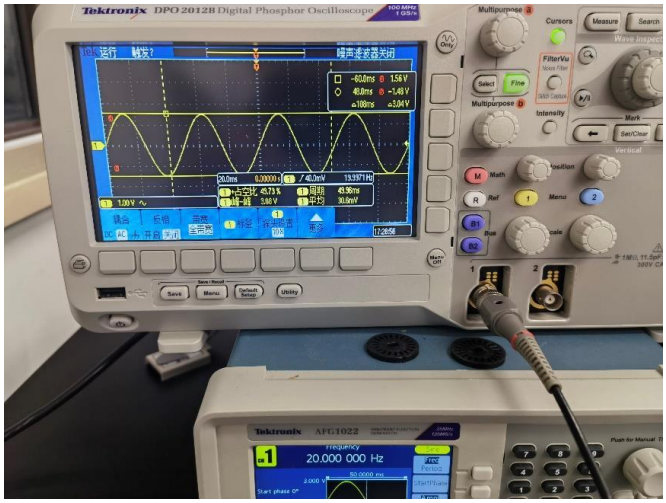
思考：

1．示波器探头的直流耦合和交流耦合有什么区别？

手动设置中调为是直流耦合，我们按照直流耦合在示波器上面得到的图像为：



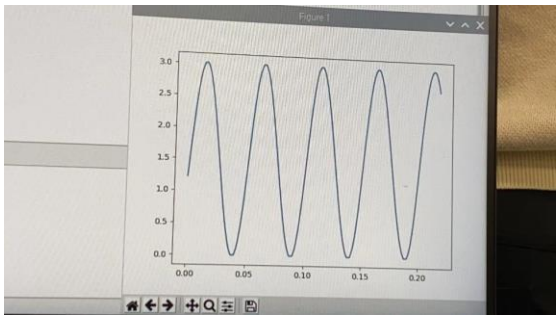
如果设置为交流耦合则是：

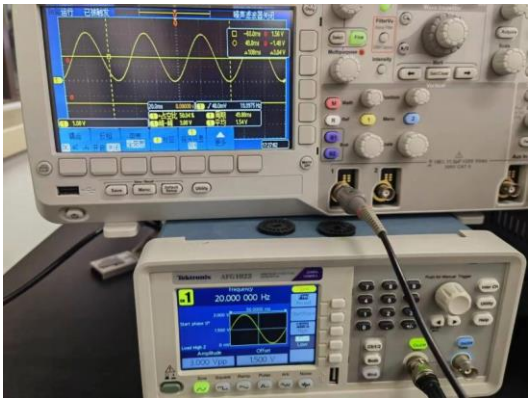


查阅资料可知：交流耦合通过隔直电容耦合，去掉了直流分量，只能过交流分量；直流耦合直通，交流直流一起过，并没有去掉交流分量（所以在图像中，直流耦合有向上偏置的趋势，看起来位置高一点）。交流耦合的各级电路是用电容或者是电感隔离开的；直流耦合的各级电路是直接的导线连接，包括通过像电阻之类的线性元件的连接。交流耦合各级电路的静态工作点是独立的；直流耦合各级电路的静态工作点是互相影响的。

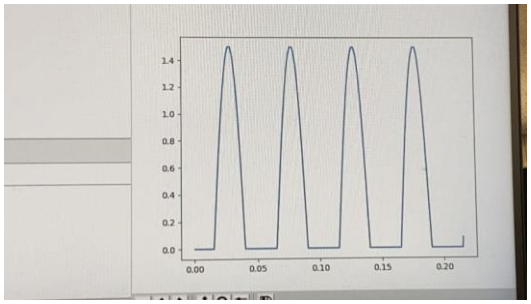
2．为什么信号发生器的输出要加直流偏置？

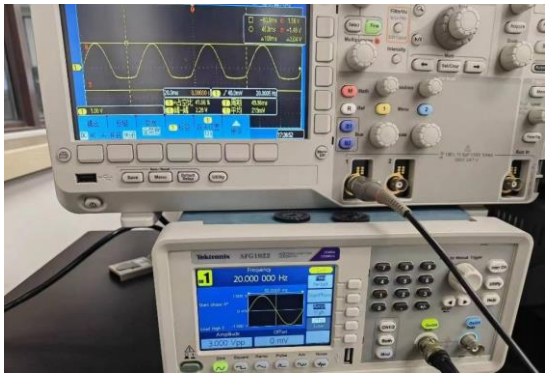
在加了直流偏置后，图像为：





如果没有加直流偏置，图像为：





ADC 是用来模数转换的，转的是电位器的电压。电位器电压范围是 0-3.3V，实际过程中不可能有负的电压值。如果没有加直流偏置，会有负的电压值出现，加了以后可以让正弦波向上偏移避免负值的出现，完整测出来电位器的电压值。

3. ADC0832.getResult() 函数内部做了什么事情？

getResult()函数的主要目的是从 ADC0832 模块的指定通道（0 或 1）读取数字化后的模拟信号值。

1. 设置数据线为输出：由于通信的初始阶段需要向 ADC 模块发送数据（或指令），因此首先将数据线（ADC_DIO）配置为输出模式

2. 启动 ADC 转换：通过一系列的时钟信号和数据线的高低电平变化，向 ADC 模块发送启动转换的命令。具体步骤包括：将片选（ADC_CS）线拉低，表示开始通信。发送初始序列：将时钟（ADC_CLK）拉低后，数据线（ADC_DIO）置高，然后时钟置高，再将数据线置高，时钟拉低，依次类推，形成特定的启动转换序列。发送通道选择：通过设置数据线的高低电平来选择读取的通道（通道 0 或通道 1）。

3. 设置数据线为输入：在发送完启动和通道选择命令后，将数据线配置为输入模式，准备接收 ADC 模块转换后的数字信号。

4. 读取数据：读取两次 8 位的数据（总共 16 位），这是 ADC0832 模块特有的数据读取方式。首先读取的 8 位是 MSB（最高位开始），接着再读取一遍作为确认（从最低位开始）。在每次时钟信号的上升沿，读取数据线上的电平，并将其作为结果的一部分。通过移位操作组装出 8 位的数字。确认读取：以相反的顺序再次读取并组装第二个 8 位数字。这一步是为了校验数据的准确性。

5. 结束通信：在数据读取完成后，将片选线（ADC_CS）拉高，结束本次 ADC 转换和数据读取过程。

6. 返回结果：如果两次读取的 8 位数据相同，说明数据有效，函数返回该 8 位数据作为结果。如果不同，则返回 0，表示读取过程中可能发生了错误。

通过这个过程，getResult()函数能够从 ADC0832 模块获取指定通道上的模拟信号的数字化值，用于后续的处理或显示。