信号分析与处理

2024/4/10

电子系统导论教学团队

实验目的

- 了解频谱分析、采样定理（混叠）和滤波的基本概念
- 掌握离散傅立叶变换的编程实现
- 掌握TLC5620的使用方法

傅立叶分析

- 傅立叶级数
  - 法国数学家傅立叶发现，任何周期函数都可以用正弦函数和余弦函数构成的无穷级数来表示。
  - 从时域到频域：

傅立叶分析

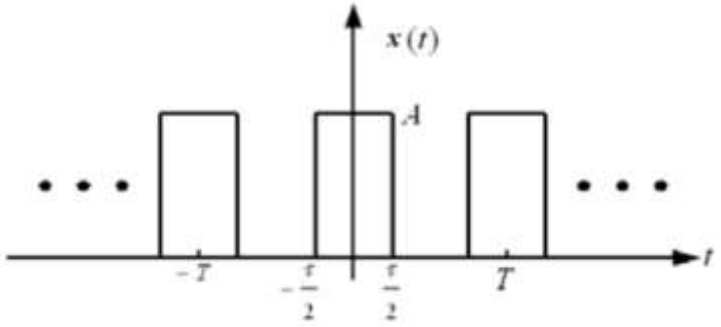
- 傅立叶级数
  - 结合欧拉公式，有

$$x(t)=\sum_{k=-\infty}^{\infty}C_k e^{jk\omega_0 t}=\sum_{k=-\infty}^{\infty}C_k e^{jk\frac{2\pi}{T}t},|t|\le T/2$$

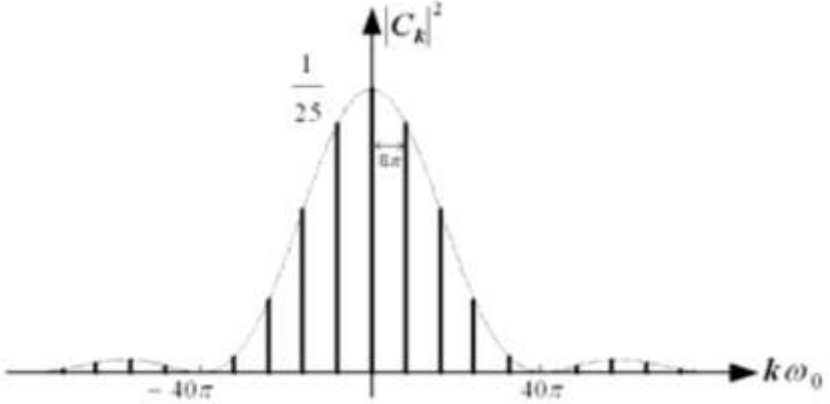
  - 其中 $C_k=\frac{1}{T}\int_{-T/2}^{T/2}x(t)e^{-jk\omega_0 t}dt$
  - C_k一般为复数，表示各频率分量的幅度和相位
  - $e^{jk\frac{2\pi}{T}t}(k=0,\pm1,\pm2,\pm3,...)$是一组正交基，它们张成了一个无穷维的空间，周期函数可以看作空间向量，傅立叶级数可以看作它向这组正交基投影。

傅立叶分析

- 傅立叶级数
  - 以方波信号为例。时域：



  - 频域：



傅立叶分析

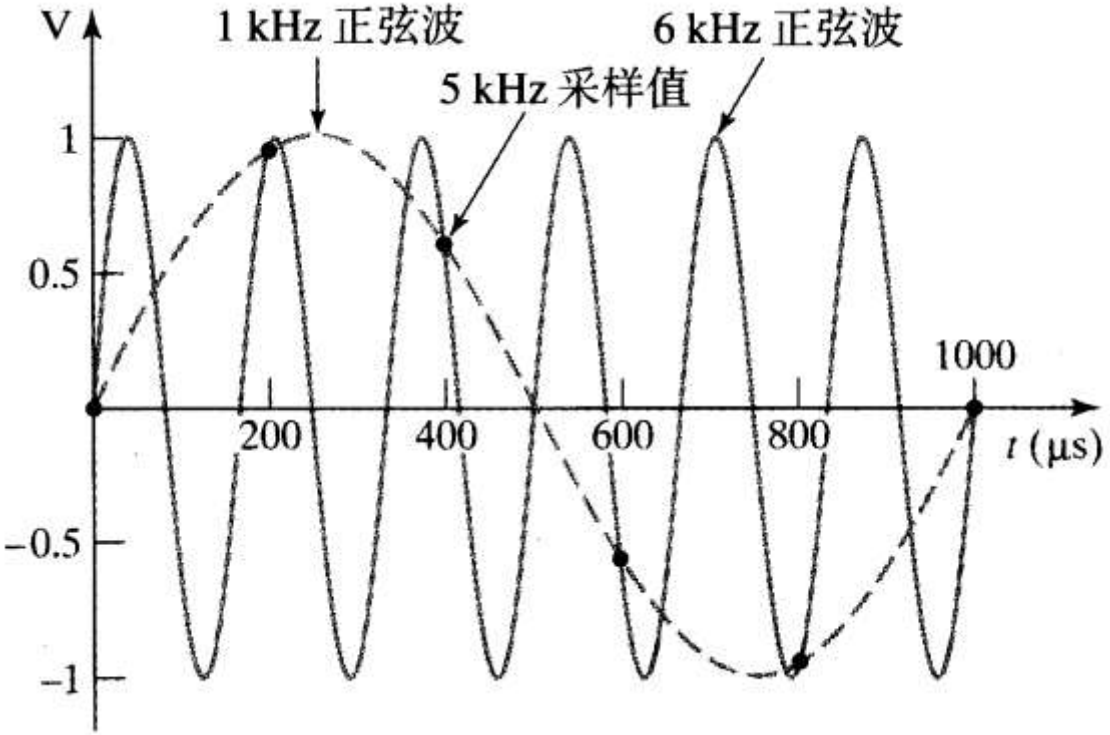
- 离散傅立叶变换
  - 经过采样的信号是离散的，不妨将有限长的离散信号进行周期延拓，即看作这一信号在时间域上不断重复，使该信号成为周期函数，这样就可以利用前面的结论。
  - 这一函数所在空间的正交基$\vec{x}_i$也要离散化。
  - 所以可以通过乘一个矩阵来完成从时间域到频率域的投影。
  - 例如
$$\begin{bmatrix} X[0] \\ X[1] \\ \cdots \\ X[N] \end{bmatrix} = \begin{bmatrix} W_N^0 & W_N^0 & \cdots & W_N^0 \\ W_N^0 & W_N^1 & \cdots & W_N^{N-1} \\ \cdots & \cdots & \cdots & \cdots \\ W_N^0 & W_N^{N-1} & \cdots & W_N^{(N-1)^2} \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ \cdots \\ x[N] \end{bmatrix}$$
  - 其中 $W_N = e^{-j\frac{2\pi}{N}}$

抽样定理

- 在进行模拟/数字信号的转换过程中，当采样频率 f_S 大于信号中最高频率 f_m 的2倍时($f_S > 2f_m$)，采样之后的数字信号完整地保留了原始信号中的信息。
- $2f_m$ 称为奈奎斯特频率。
- 任何信号都可以看作正弦波的叠加。确定一个正弦波的振幅、频率和相位信息至少需要不全为零的3个点。只要采样频率 f_S 大于最高频率 f_m 的2倍，就可以使各个频率的每个周期上至少有不全为零的3个点被采到，所以采样之后的数字信号完整地保留了原始信号中的信息。

抽样定理

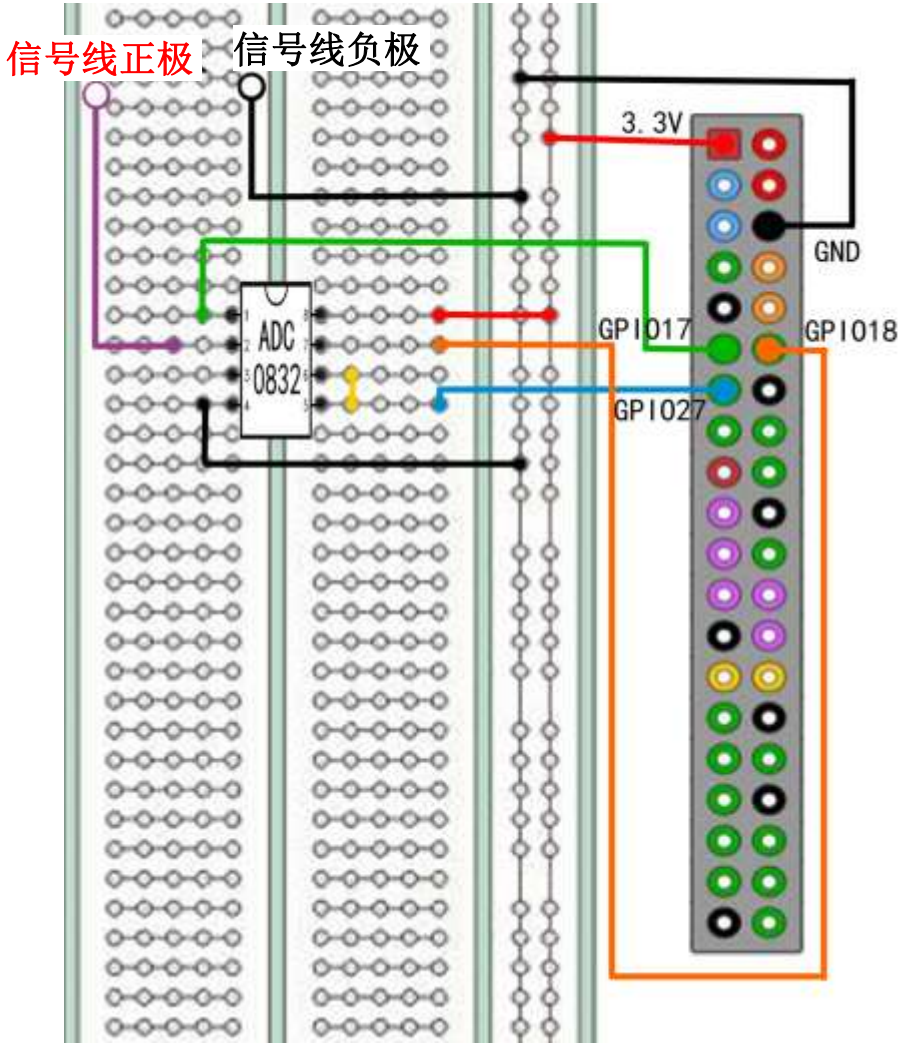
- 如果采样频率低于奈奎斯特频率，称为欠采样，将导致频谱混叠，造成信号失真。
- 如图，欠采样时恢复出来的信号不是原始信号，而是一个不真实的低频信号：



欠采样造成的混叠

- 快速旋转的车轮，螺旋桨会呈现缓慢旋转的错觉
- 原因：人眼或摄像机的采集频率远低于旋转频率，因此造成频谱混叠

树莓派实现傅立叶分析




```
import ADC0832
import time
import numpy as np
import matplotlib.pyplot as plt
```

用到前面的信号采样函数和AD转换模块ADC0832

```
fft_size=256
sampl_freq=3700
```

取256点数据做FFT（快速傅立叶变换，离散傅里叶变换的一种高效算法，数据长度为2的幂时效果最佳），实测得采样频率大约是3.7kHz。

树莓派实现傅立叶分析

```
y_fft=np.fft.rfft(y)/fft_size
```

此函数对有限长实信号做快速傅立叶变换，得到fft_size/2+1个复数，其中y_fft[0]是直流分量，y_fft[1]到y_fft[fft_size/2]是正频率分量，实信号的负频率分量是正频率分量的共轭复数，不必单独列出。

```
y_fft_ampl=np.abs(y_fft)
```

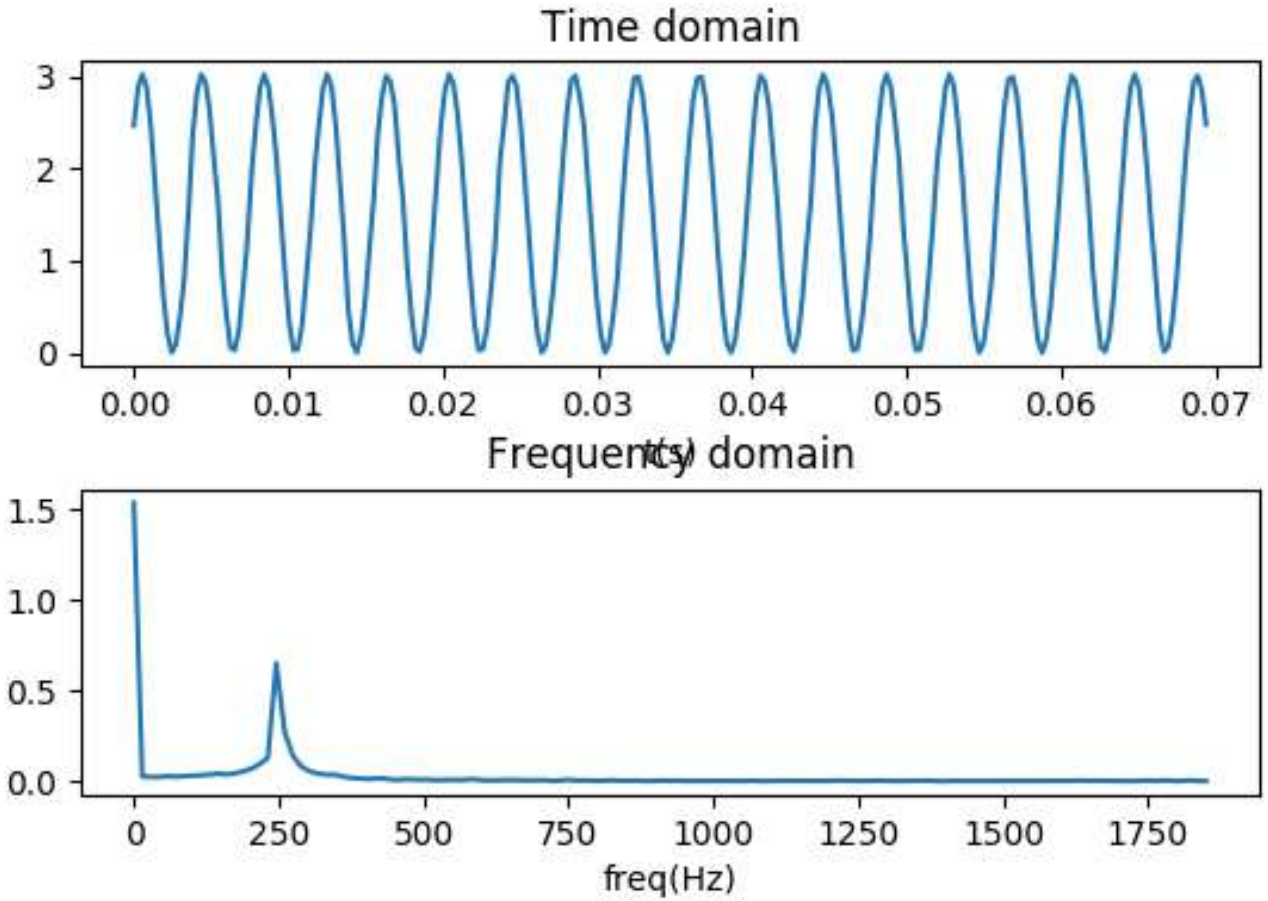
只绘制幅度谱

```
freq=np.linspace(0,sampl_freq/2,int(fft_size/2+1))
```

计算频谱上每一点的真实频率
详见fft.py

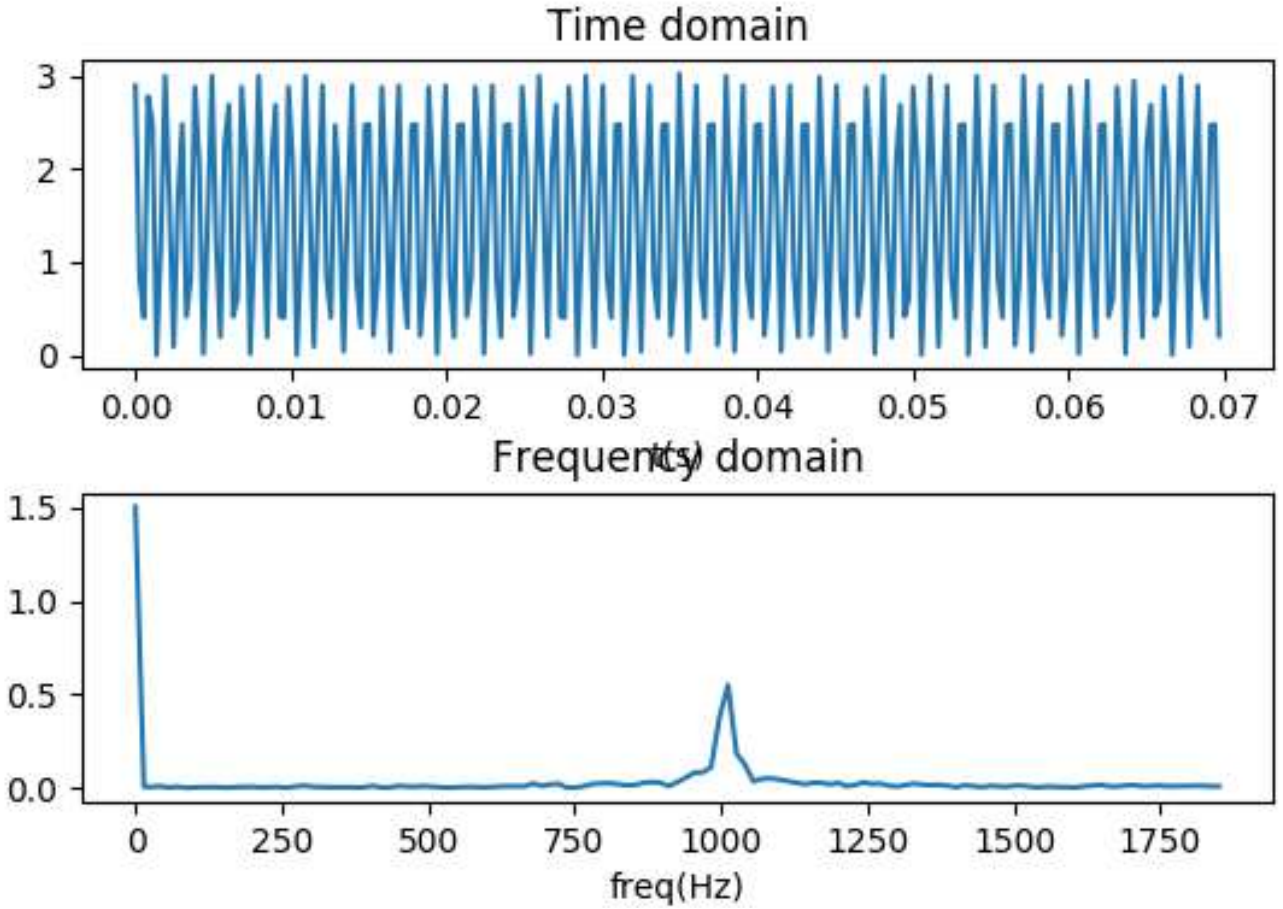
树莓派实现傅立叶分析

- 过采样（信号频率为250Hz）：



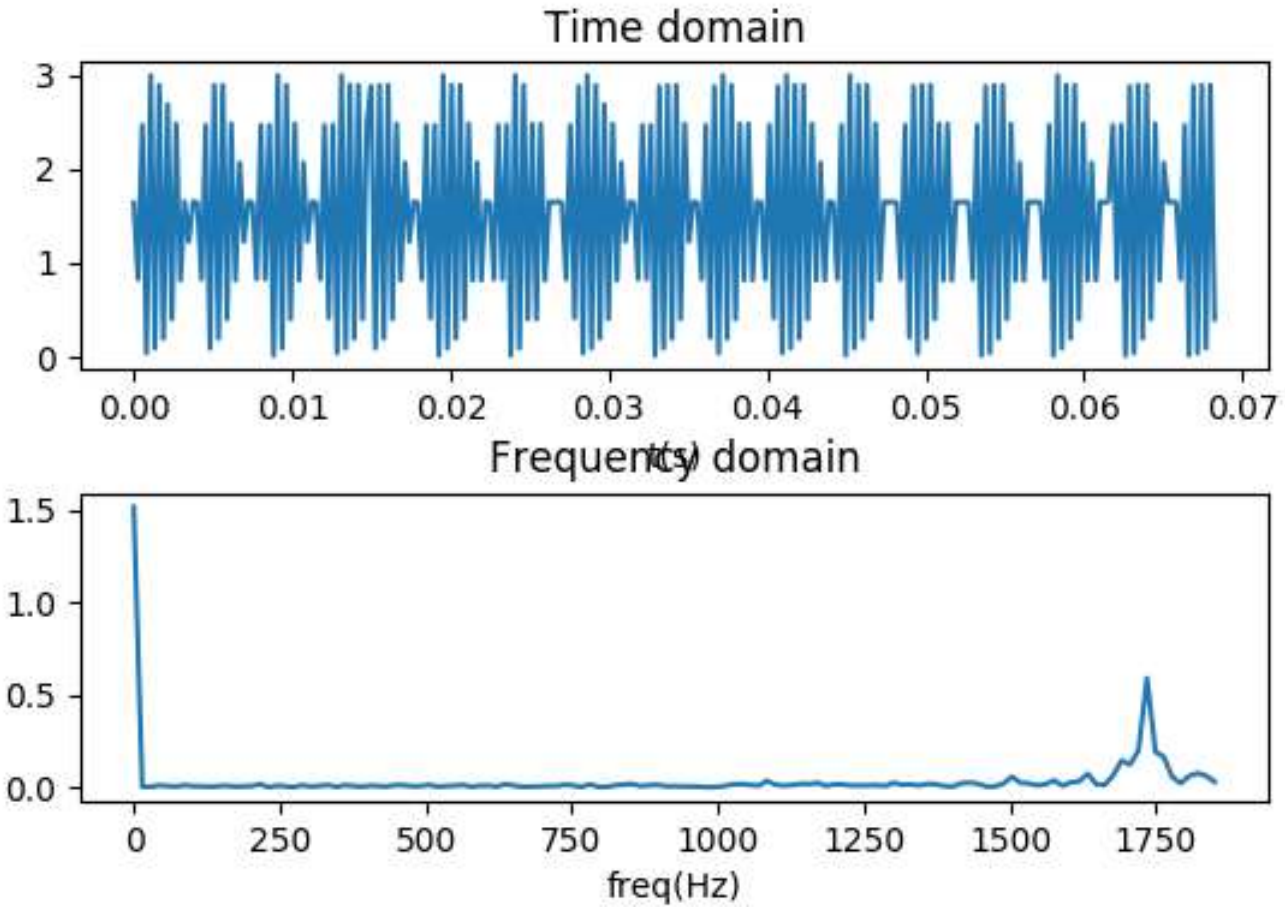
树莓派实现傅立叶分析

- 过采样（信号频率为1kHz）：



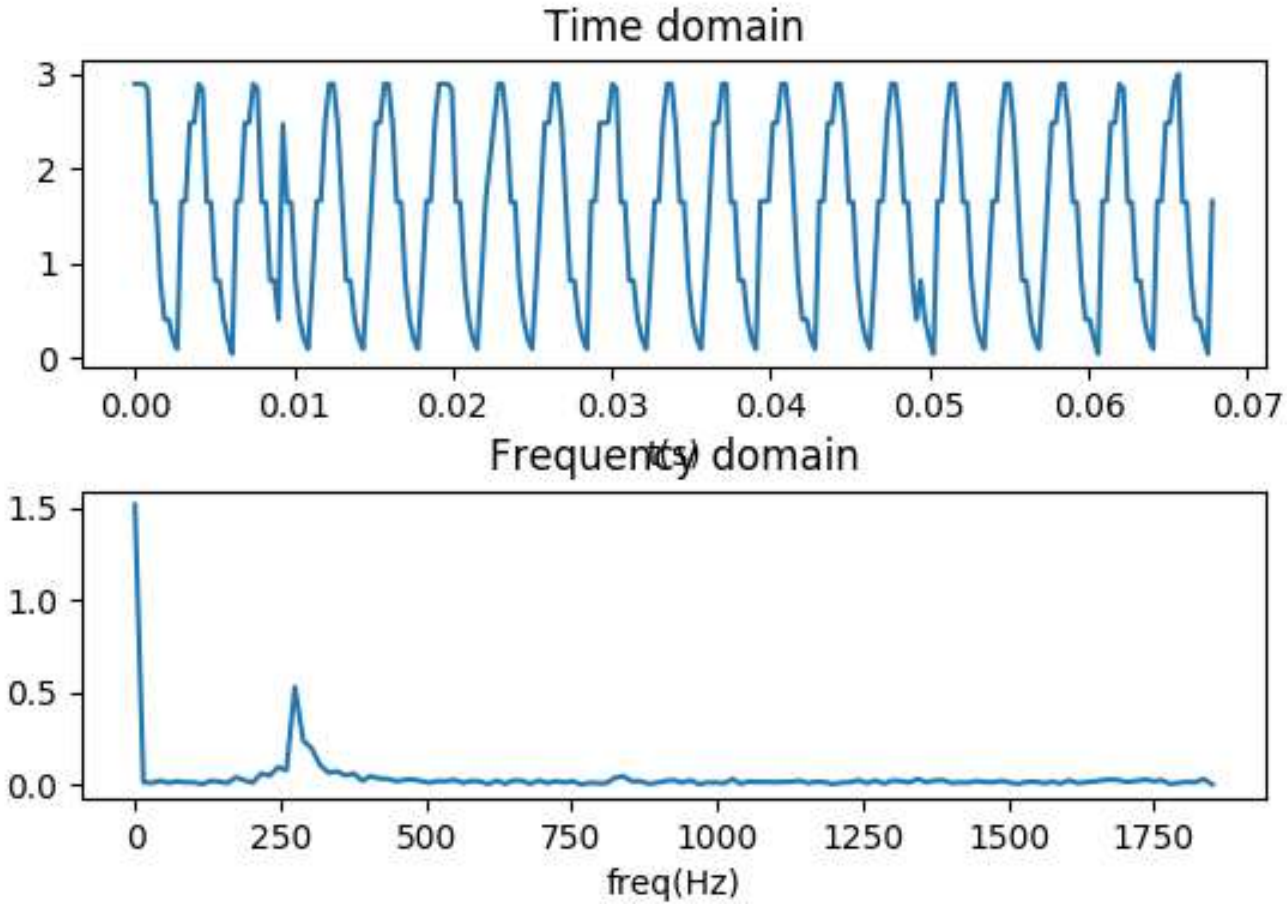
树莓派实现傅立叶分析

- 欠采样（信号频率为2kHz）：



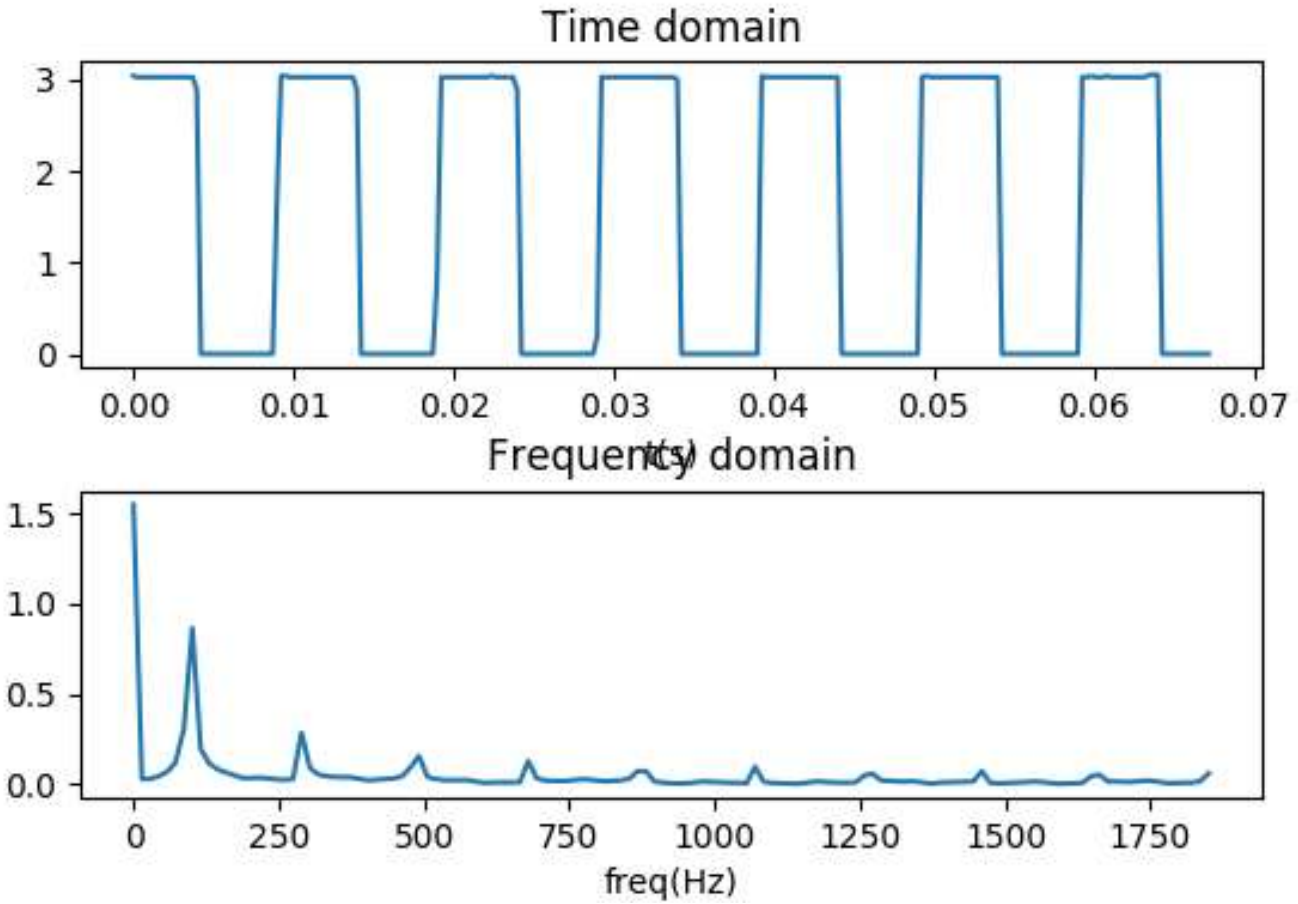
树莓派实现傅立叶分析

- 欠采样（信号频率为3.5kHz）：



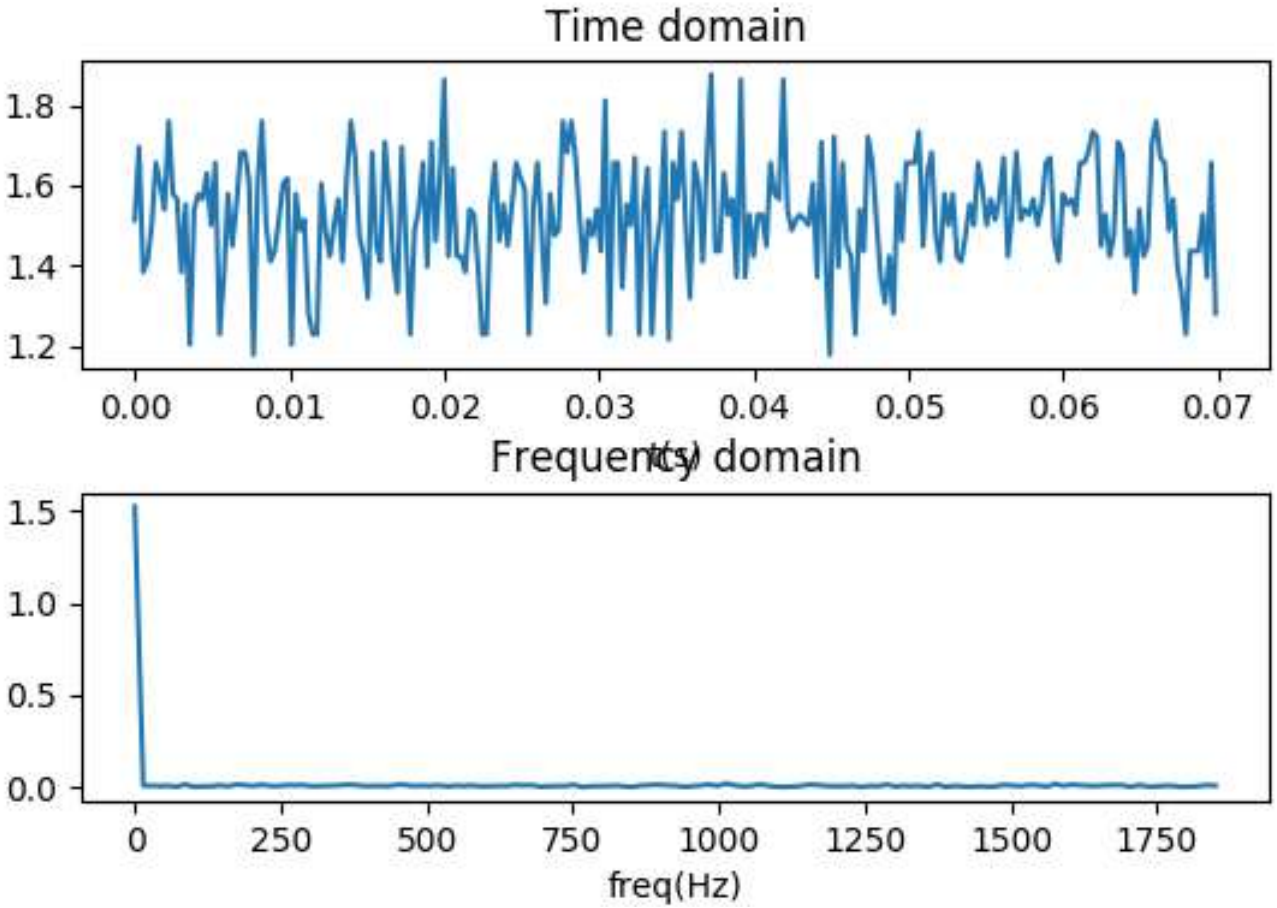
树莓派实现傅立叶分析

- 对100Hz方波采样：

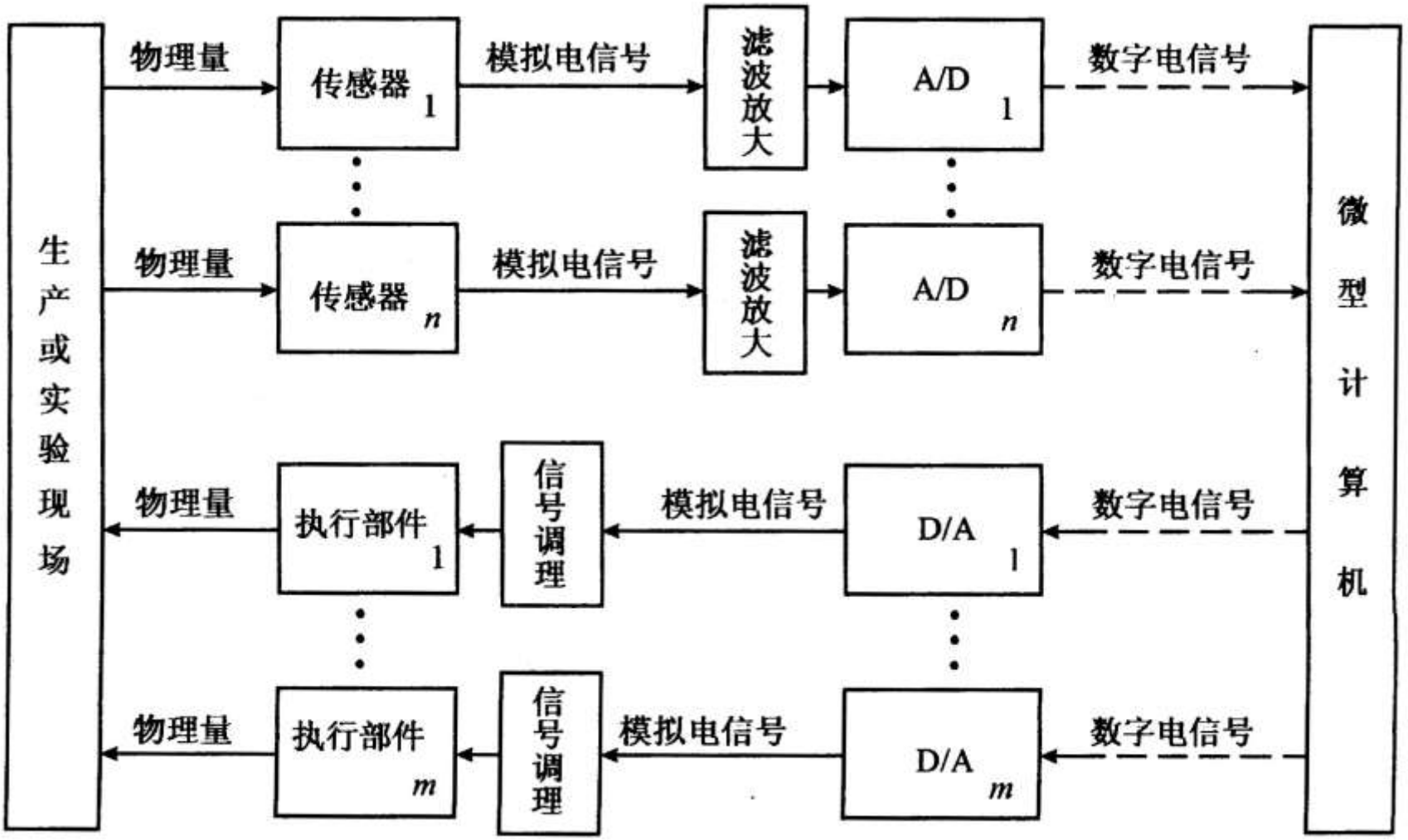


树莓派实现傅立叶分析

- 对白噪声采样：

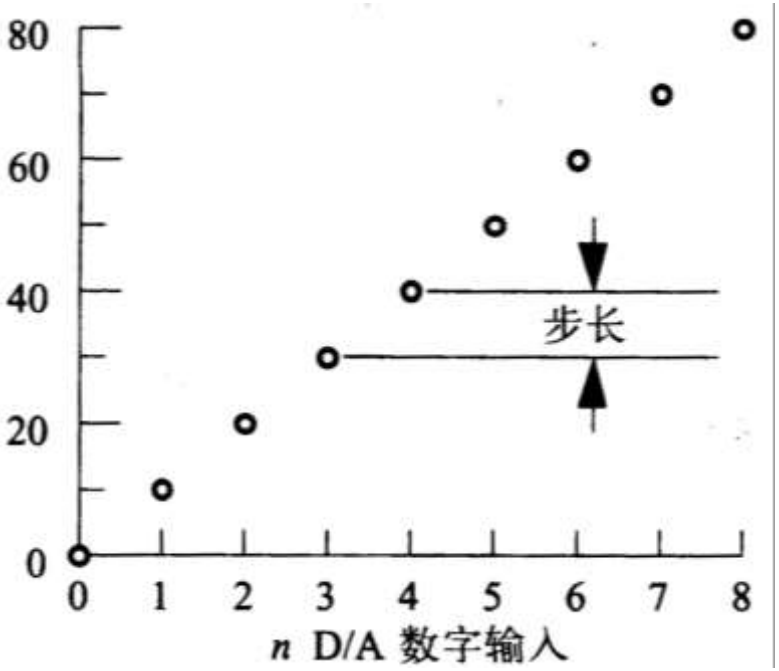


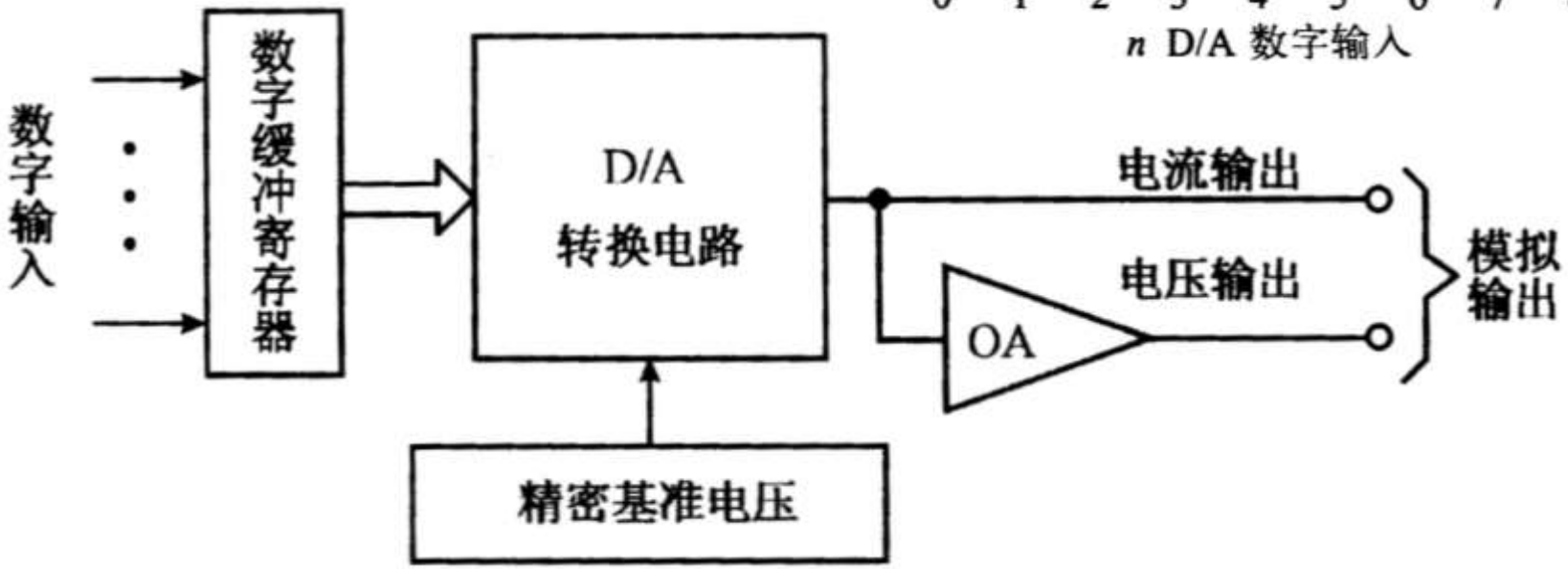
模拟量与数字量


生产或实验现场
物理量
传感器 1
传感器 n
模拟电信号
滤波放大
A/D 1
A/D n
数字电信号
微型计算机
执行部件 1
执行部件 m
信号调理
D/A 1
D/A m

数模转换器DAC

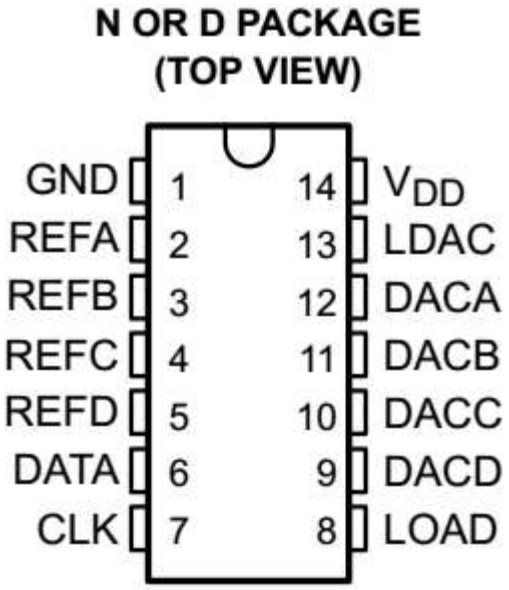
- 输入：数字量D和模拟基准电压V_R
- 输出：模拟量V_A或I_A





树莓派实现回放

- TLC5620：DAC模块
  - TLC5620是美国德州仪器（TI）公司生产的8位带有高阻抗缓冲输入的4通道D/A转换芯片。可产生单调的、1到2倍于基准电压和接地电压差值的输出。通常情况下TLC5620的供电电压为5V，器件内部集成上电复位功能。通过CLK、DATA、LOAD和LDAC四根控制线可实现对该芯片的控制。



树莓派实现回放

- TLC5620：DAC模块
  - 1.GND：输入工作电压地端。
  - 2~5.REFA~REFD：4个参考电压输入端。
  - 6.DATA：串行数据输入端，时钟下降沿数据被读入。
  - 7.CLK：时钟信号输入端。
  - 8.LOAD：数据装载控制端。当LDAC为低电平时，在LOAD信号下降沿，将输入的数据锁入输出门，并立即产生模拟电压输出。
  - 9~12.DACD~DACA：4个模拟电压输出端。
  - 13.LDAC：装载DAC控制端。当LDAC为高电平时，有数字信号输入时DAC输出不会被更新。只有当LDAC由高电平下降为低电平时才会更新模拟输出。
  - 14.VDD：输入工作电压正端。

树莓派实现回放

- TLC5620：DAC模块
  - 各通道输出电压表达式：

$$V_O(\text{DACA|B|C|D}) = \text{REF} \times \frac{\text{CODE}}{256} \times (1 + \text{RNG bit value})$$

  - 其中REF是参考电压，RNG是串行控制字内的1或0，CODE是数字信号输入，范围是0~255。
  - 串行控制字内的A1和A0用于选通通道：

A1	A0	DAC UPDATED
0	0	DACA
0	1	DACB
1	0	DACC
1	1	DACD

树莓派实现回放

- TLC5620：DAC模块
  - 工作时序：

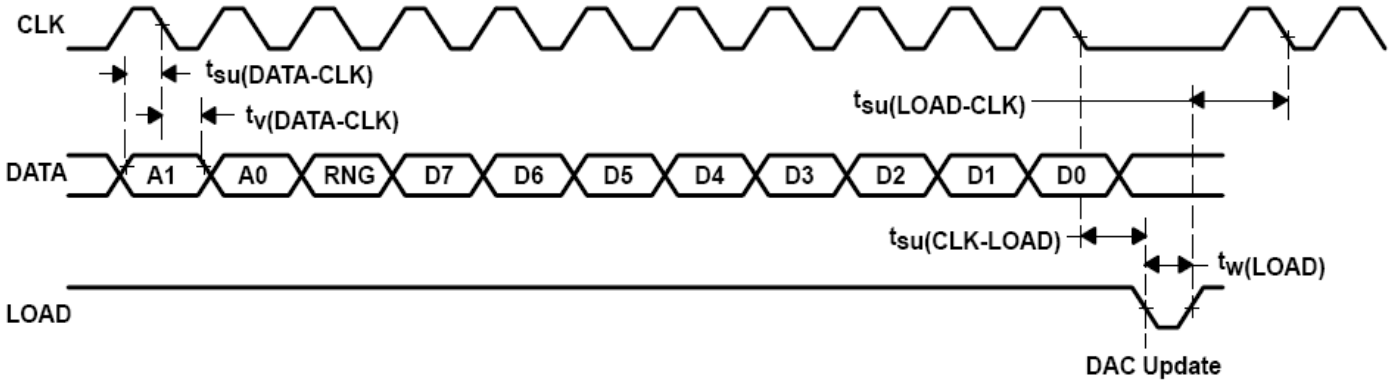
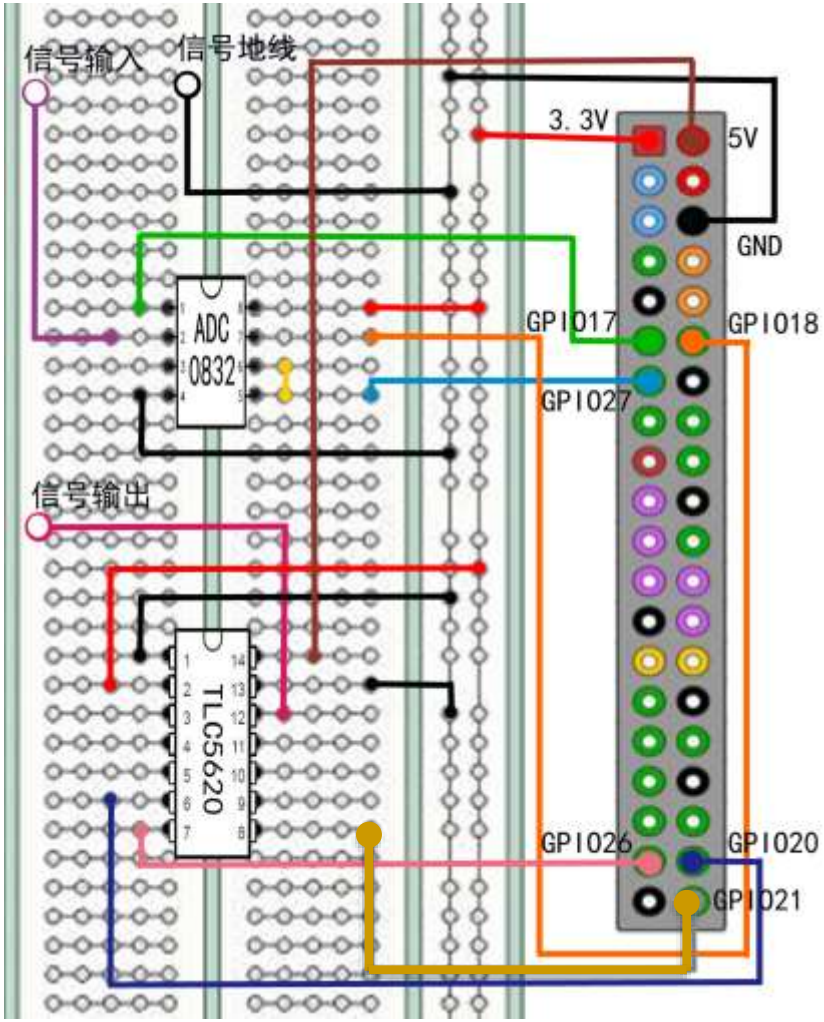


Figure 1. LOAD-Controlled Update (LDAC = Low)

  - 当LOAD为高电平、LDAC为低电平时，串行数据在CLK每一个下降沿由时钟同步送入DATA端口。一旦8位数据位都送入，LOAD变为低脉冲电平，以便把数据锁存至串行数据寄存器中。由于LDAC为低电平，锁存在串行数据寄存器中的数据自动锁存至所选择的DAC中，更新DAC输出。

树莓派实现回放



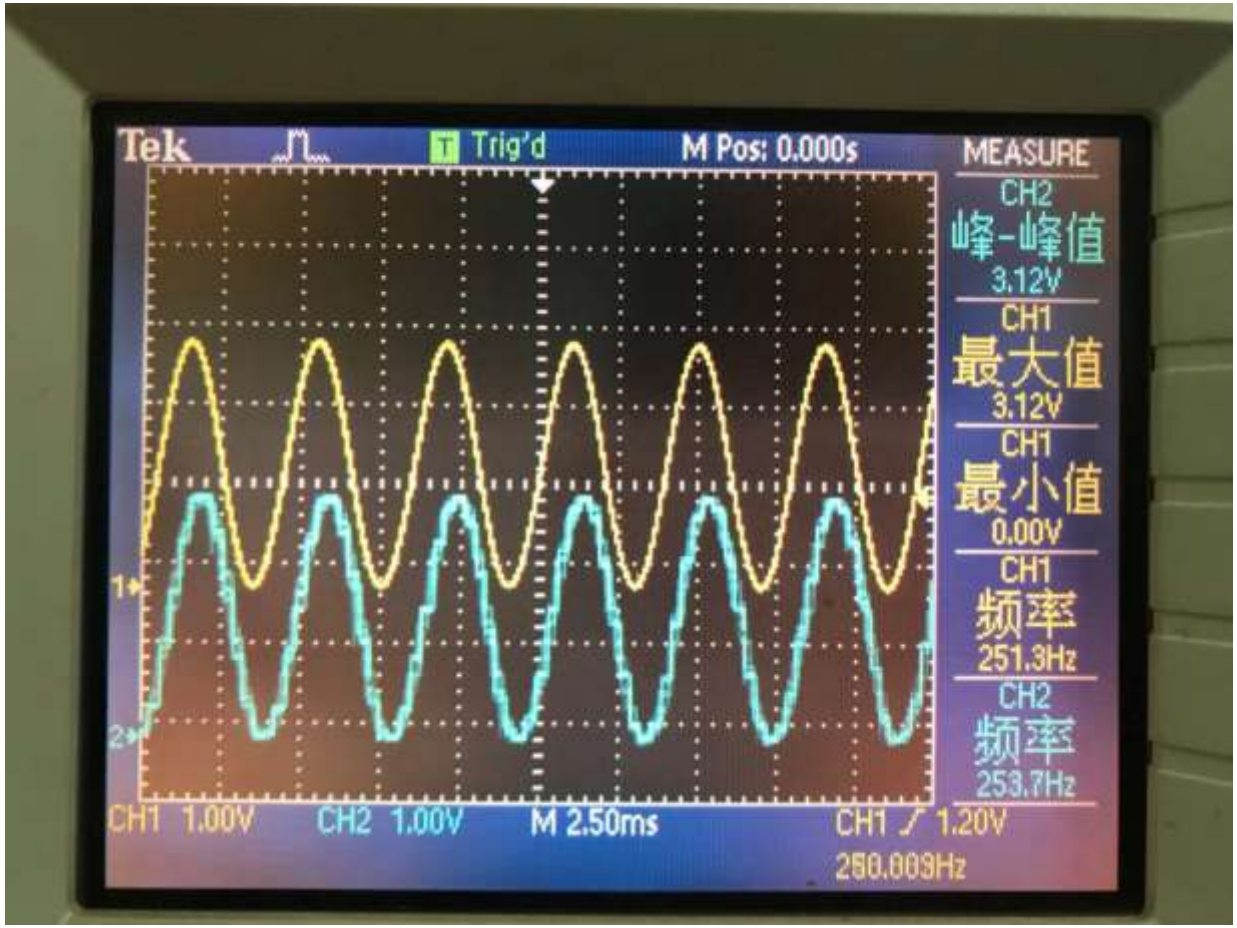
用Pin37，Pin38，Pin40分别控制CLK，DATA，LOAD引脚。
引脚LDAC接地。
电源电压VDD接5V。
令A1A0=00，选择DACA输出模拟信号。

```
import ADC0832
import DAC_TLC5620 as DAC
import time
import numpy as np
import matplotlib.pyplot as plt
```

DAC_TLC5620.py是使用TLC5620完成DA转换的函数。
详见playback.py和DAC_TLC5620.py。

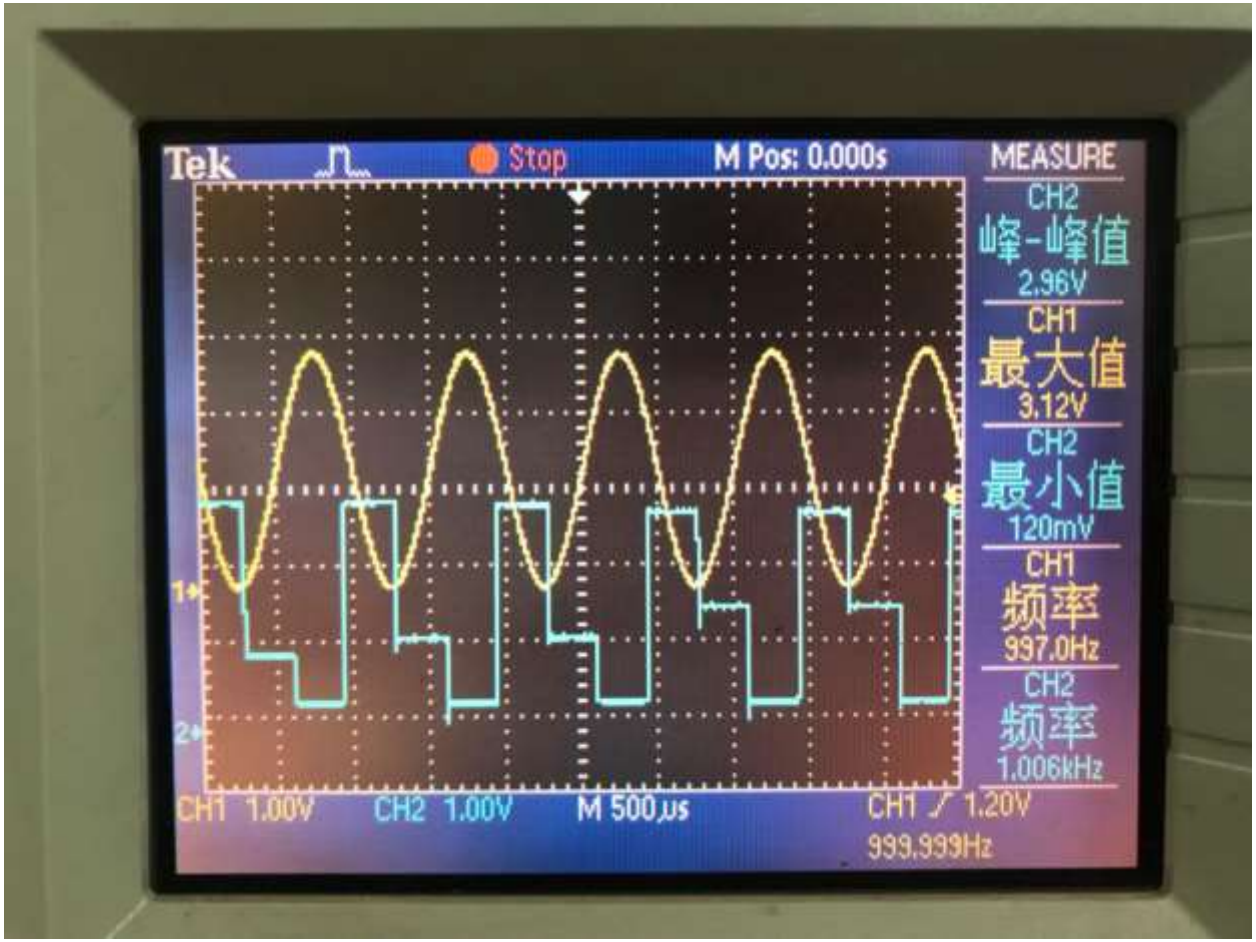
树莓派实现回放

- 对正弦波进行实时回放（过采样）：



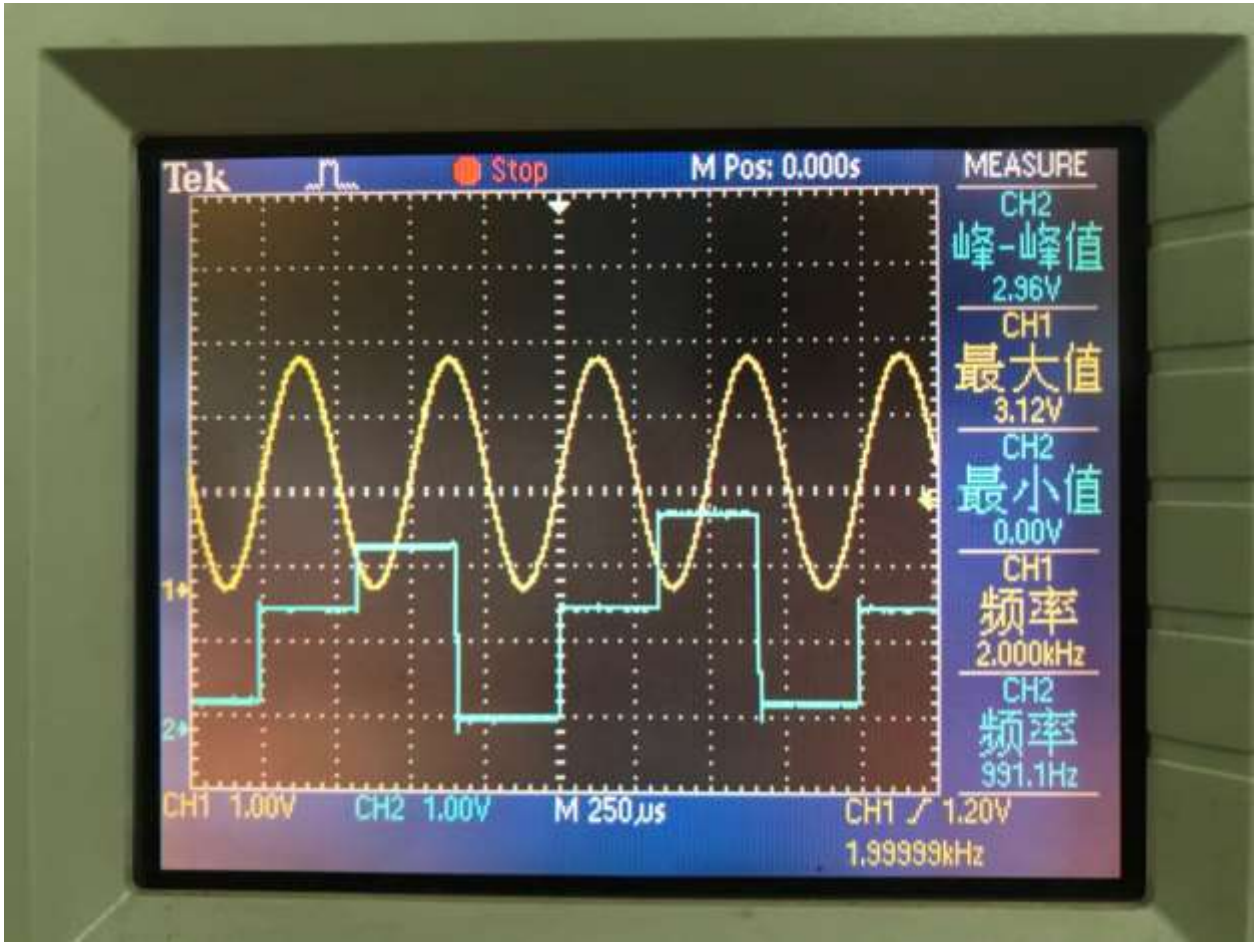
树莓派实现回放

- 对正弦波进行实时回放（过采样）：



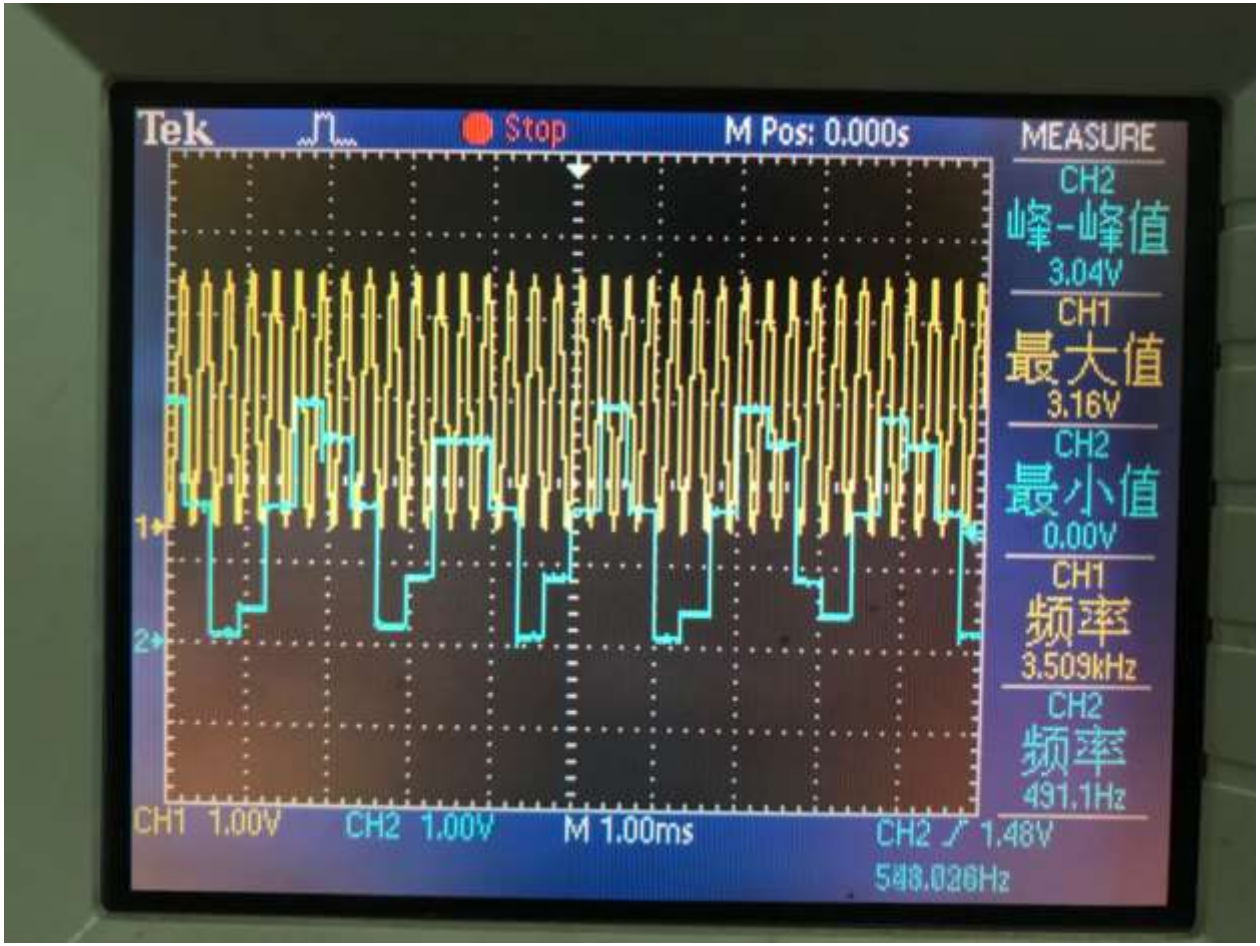
树莓派实现回放

- 对正弦波进行实时回放（欠采样）：



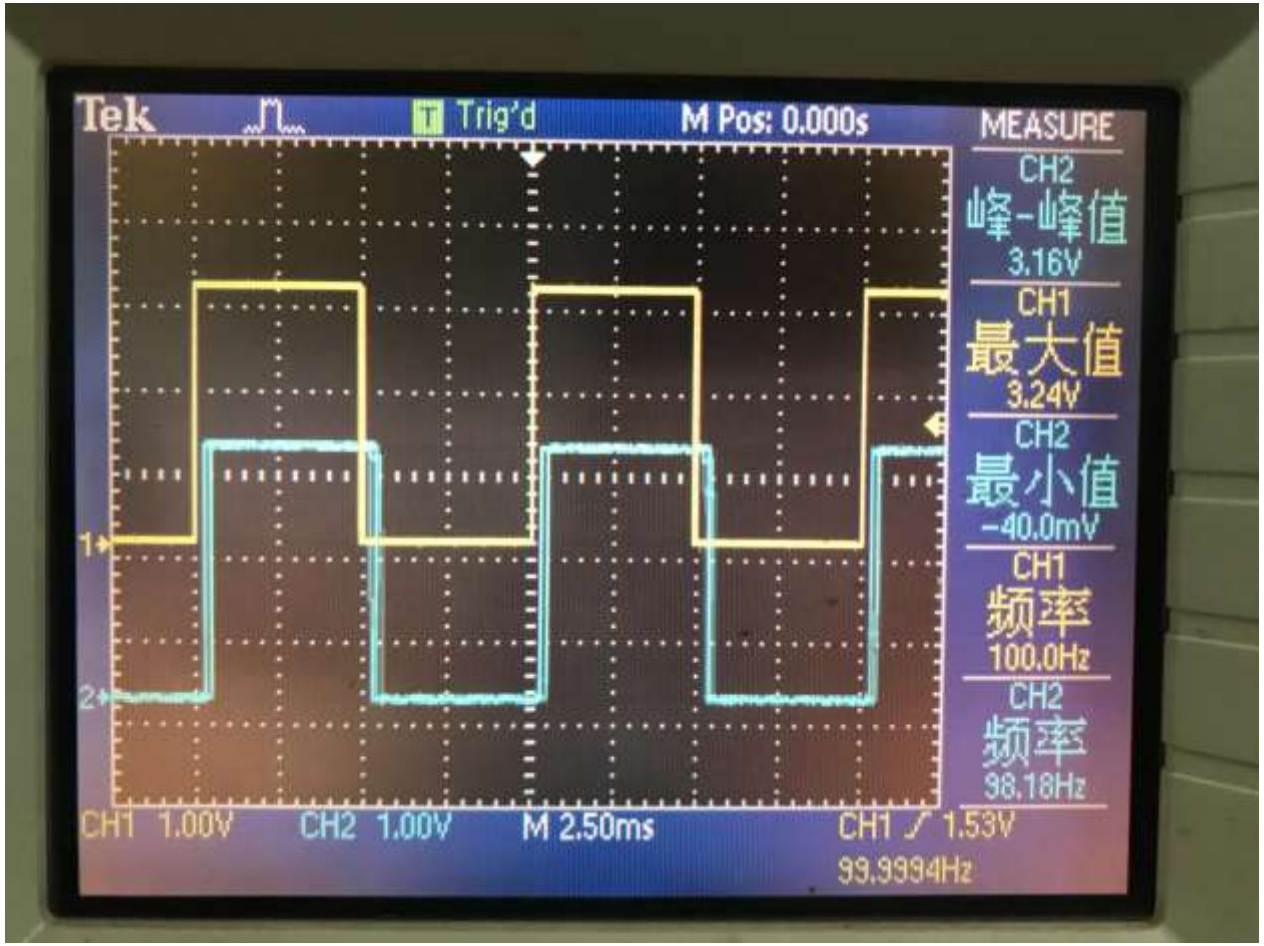
树莓派实现回放

- 对正弦波进行实时回放（欠采样）：



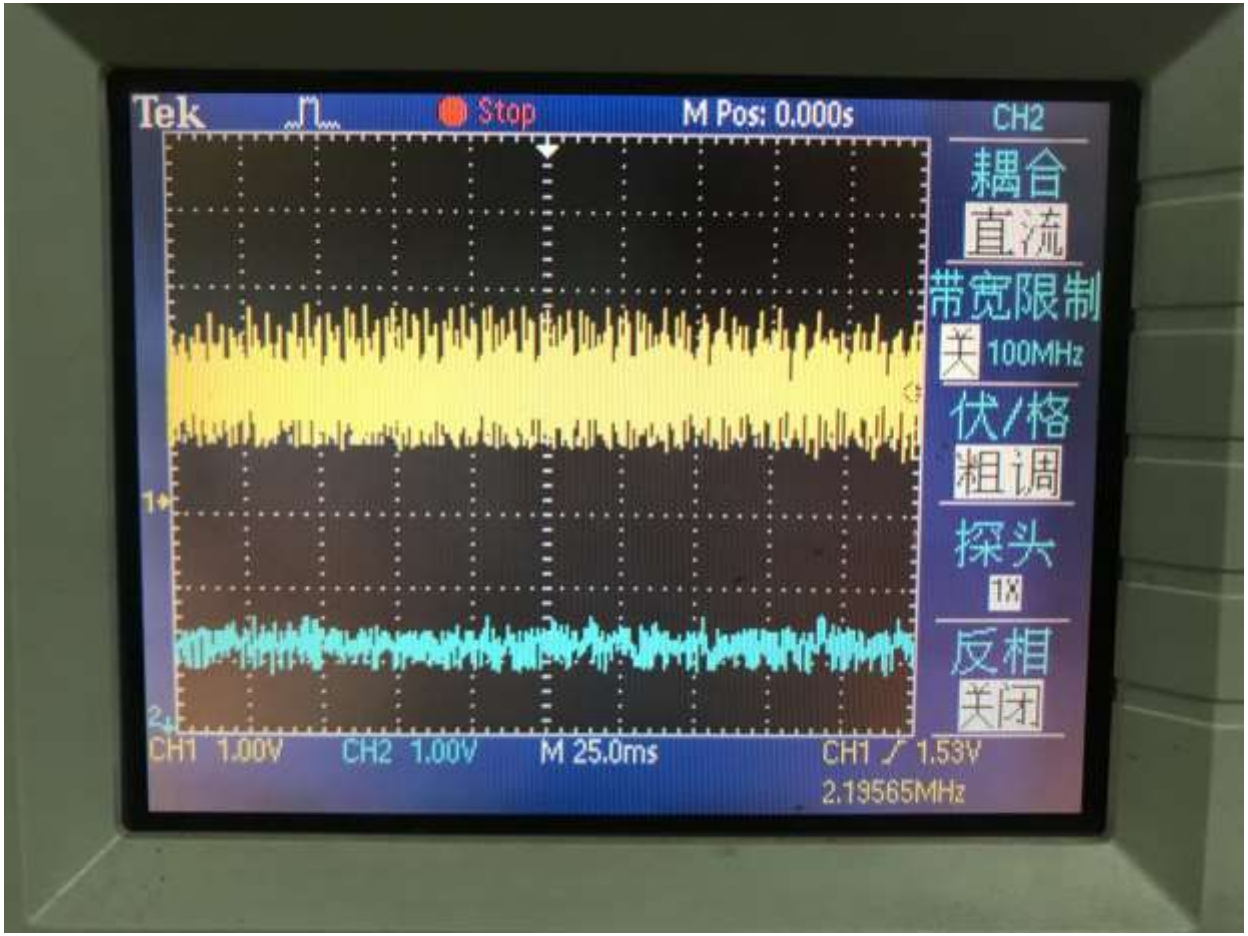
树莓派实现回放

- 对方波进行实时回放：



树莓派实现回放

- 对白噪声进行实时回放：



树莓派实现滤波

```
fft_size=256
sampl_freq=3700
freq_low=0
freq_high=500
```

不妨假设下截止频率是0Hz，上截止频率是500Hz。

```
y_fft=np.fft.fft(y)
```

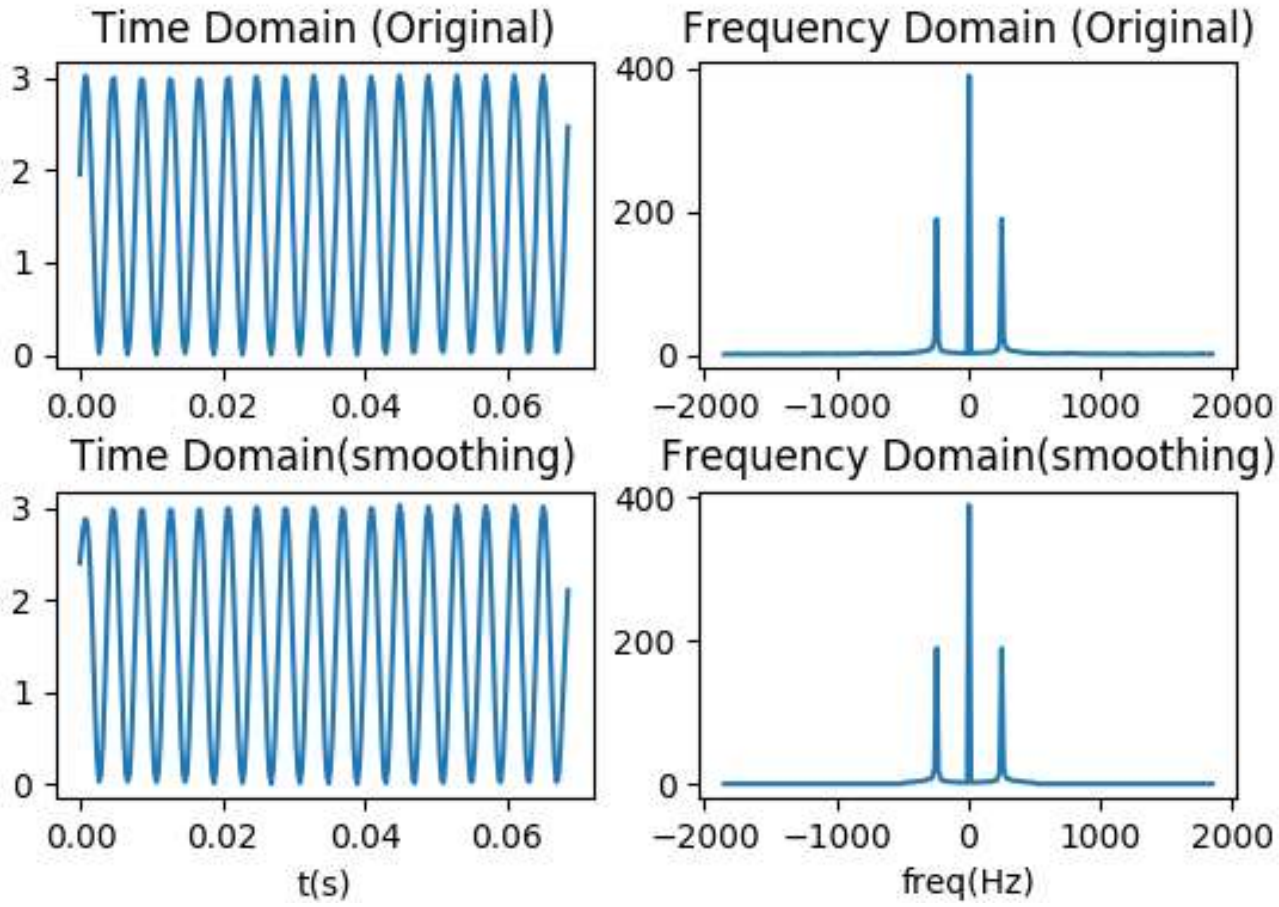
函数np.fft.fft(y)对有限长信号y做FFT（快速傅立叶变换），得到fft_size个数据，其中y_fft[0]是直流分量，y_fft[1]~y_fft[fft_size/2]是正频率分量，y_fft[fft_size/2+1]~y_fft[fft_size-1]是负频率分量。

```
smooth_org=np.fft.ifft(smooth_fft)
```

函数np.fft.ifft(smooth_fft)对经过频域滤波的信号smooth_fft做傅里叶反变换。
详见smoothing.py。

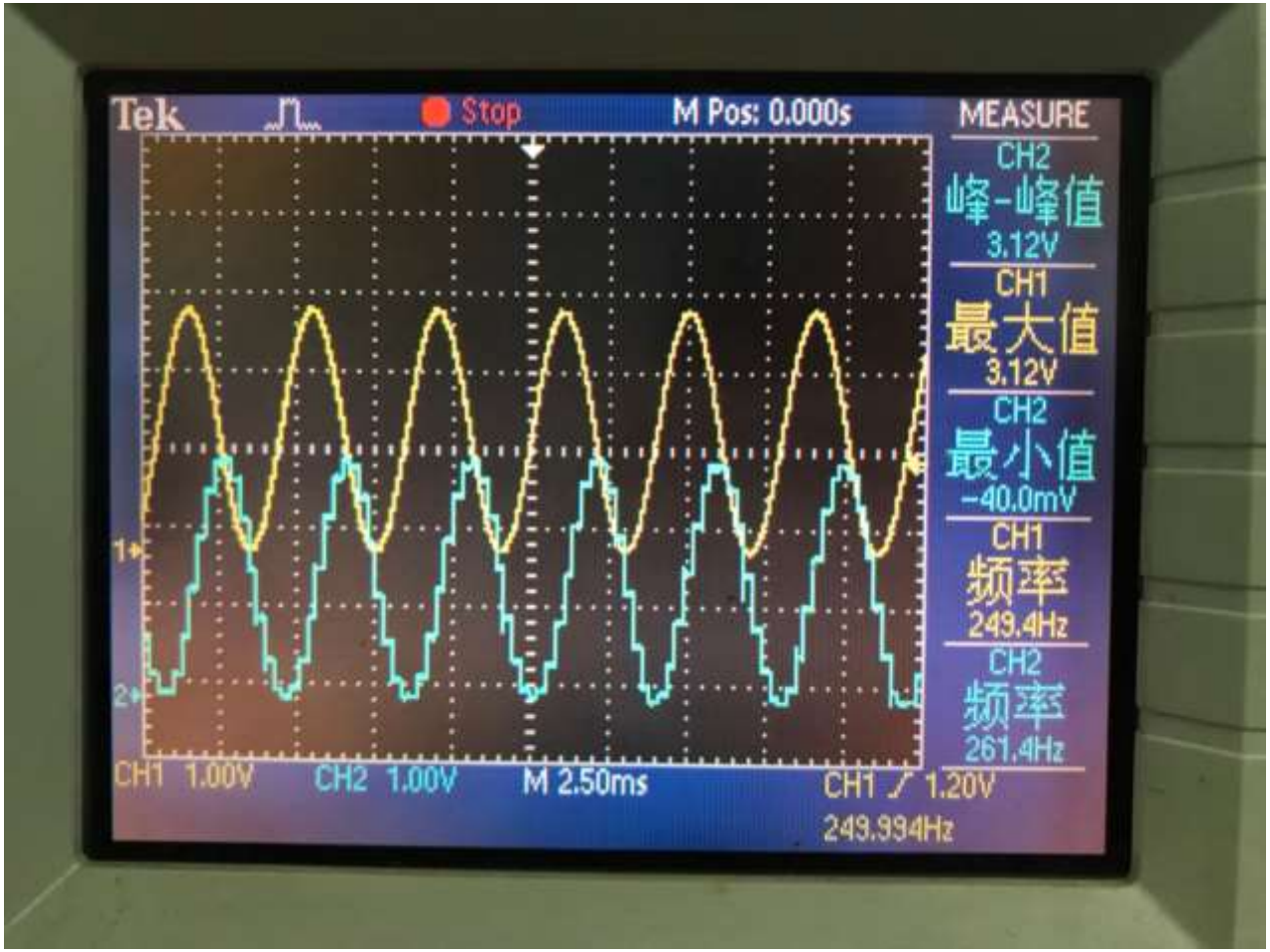
树莓派实现滤波

- 250Hz正弦波（500Hz低通滤波）：



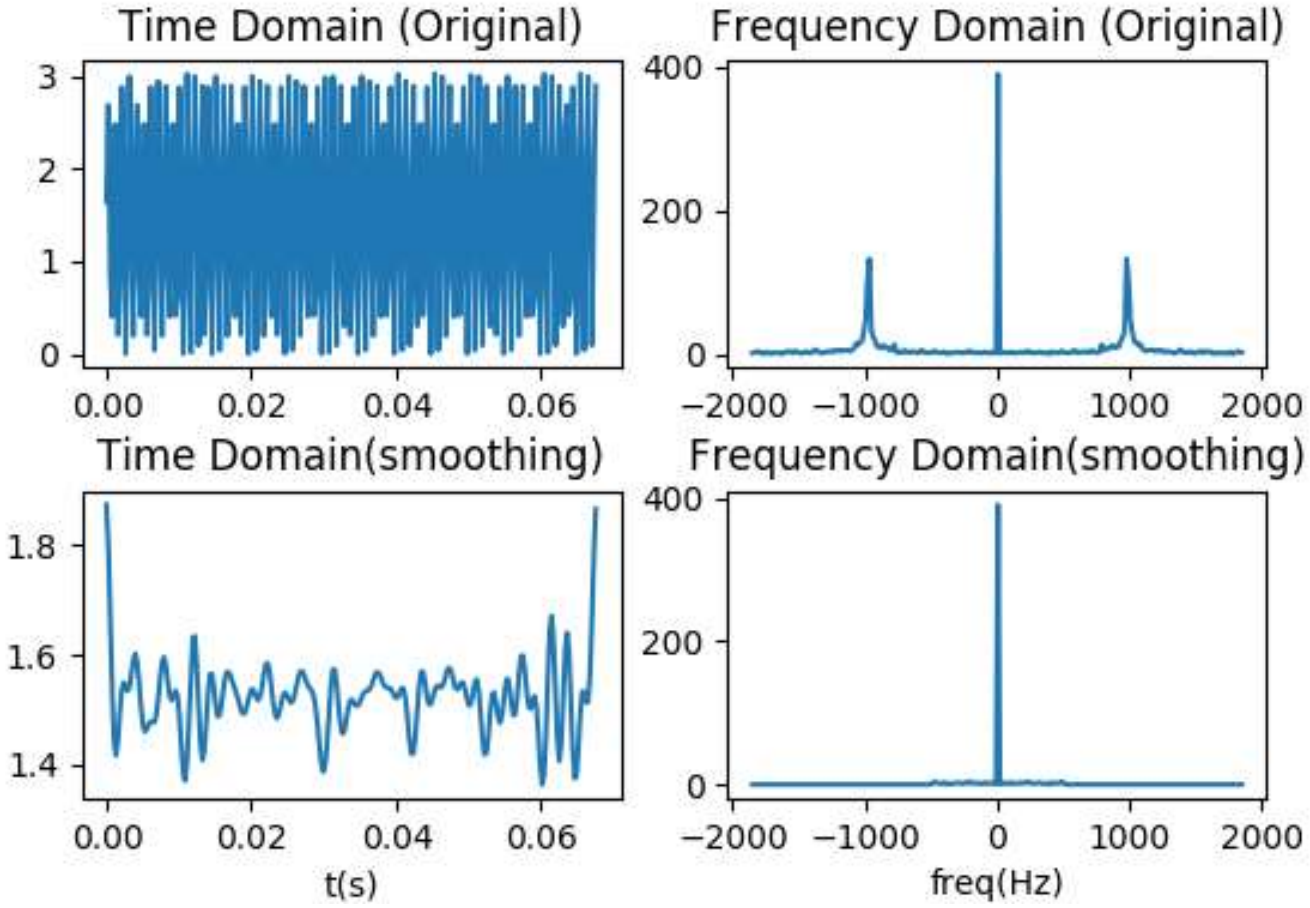
树莓派实现滤波

- 250Hz正弦波（500Hz低通滤波）：



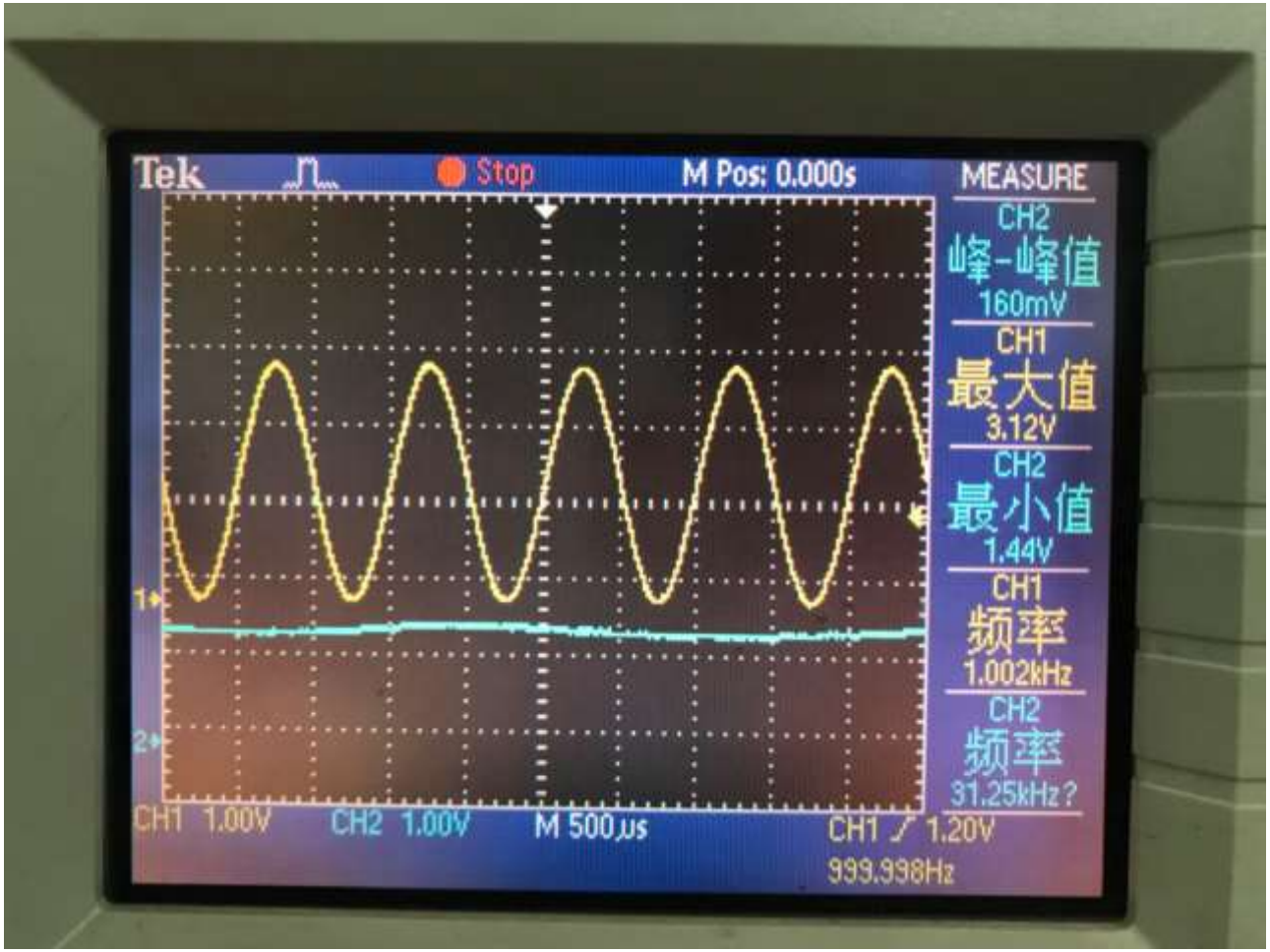
树莓派实现滤波

- 1kHz正弦波（500Hz低通滤波）：



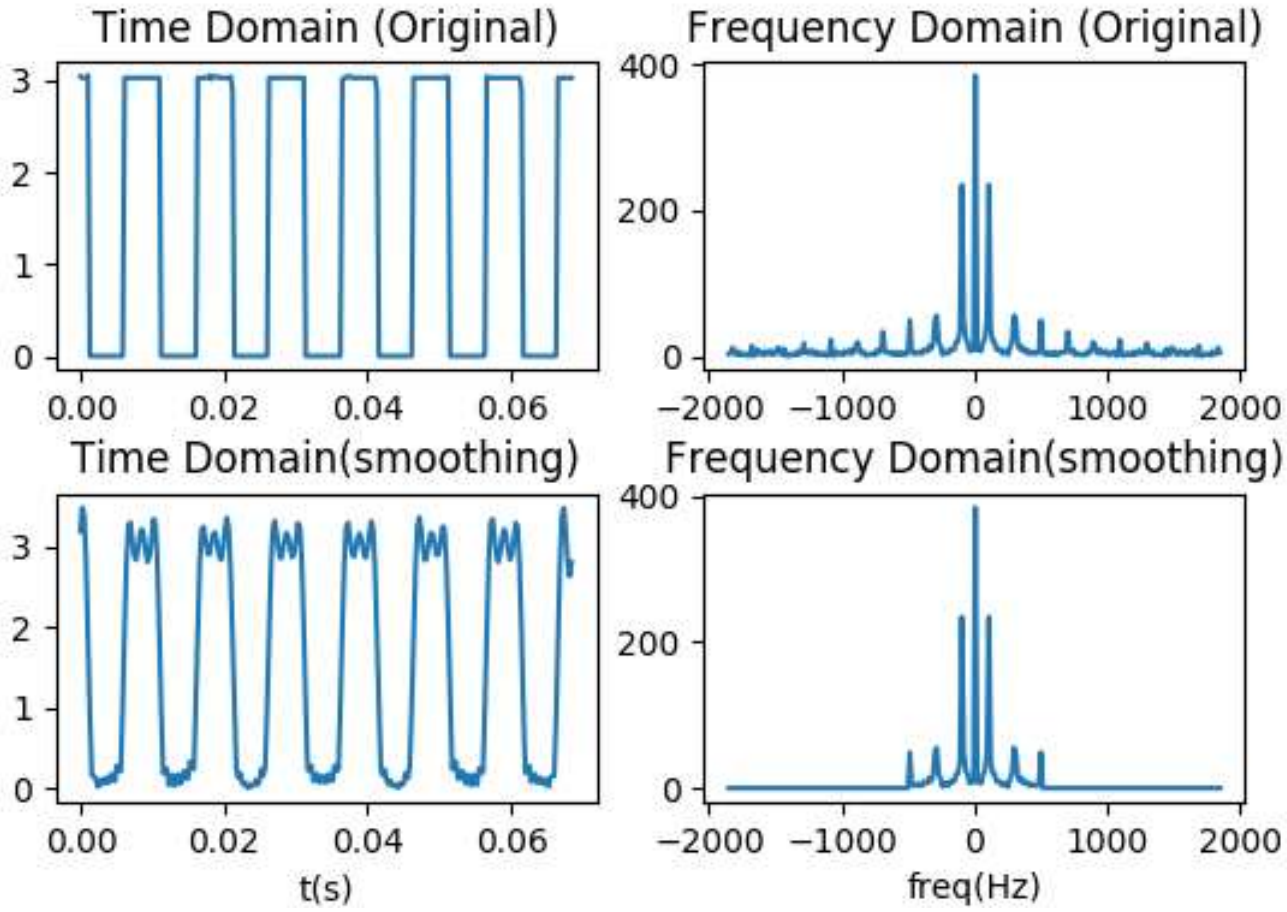
树莓派实现滤波

- 1kHz正弦波（500Hz低通滤波）：



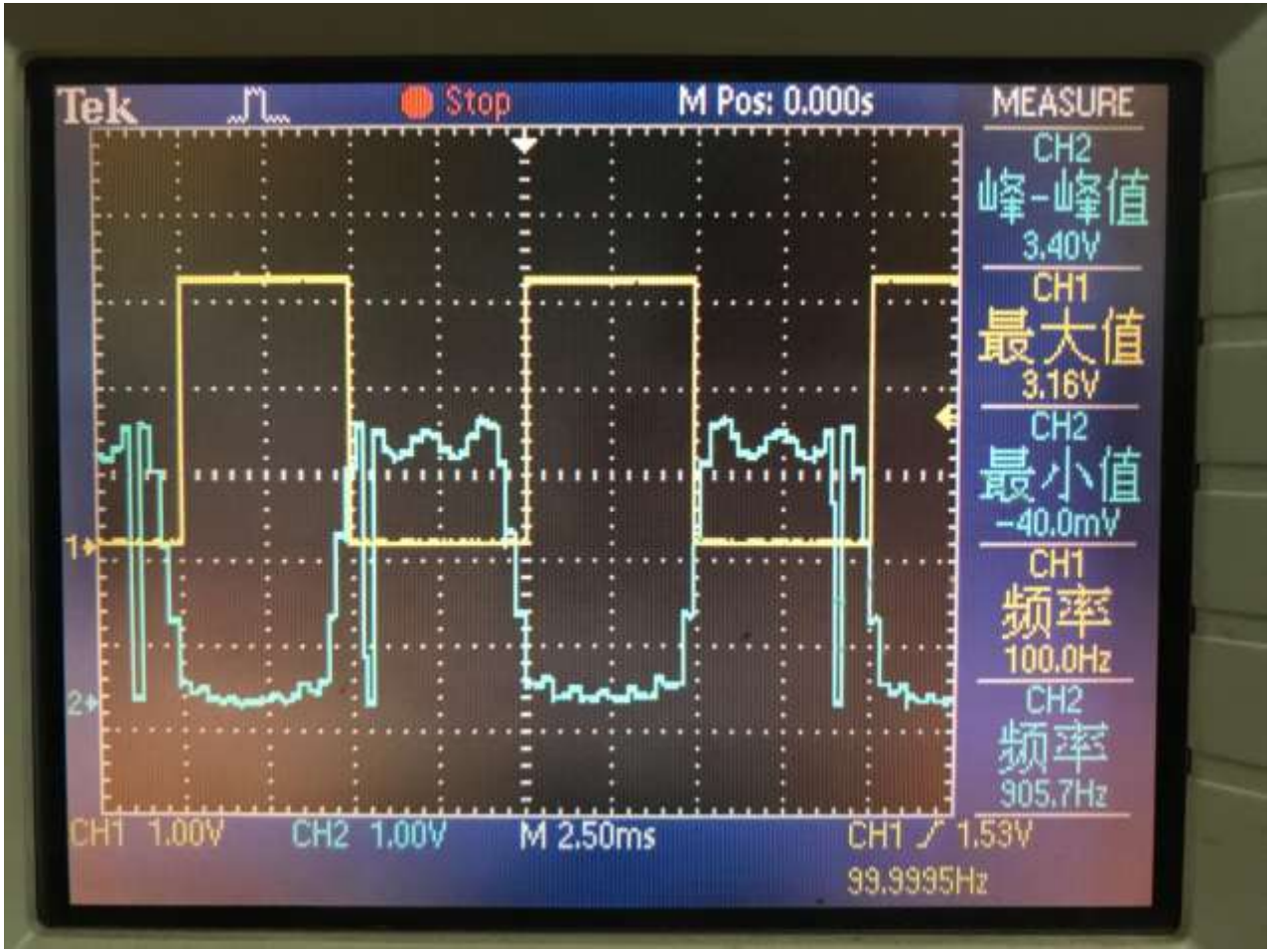
树莓派实现滤波

- 100Hz方波（500Hz低通滤波）：



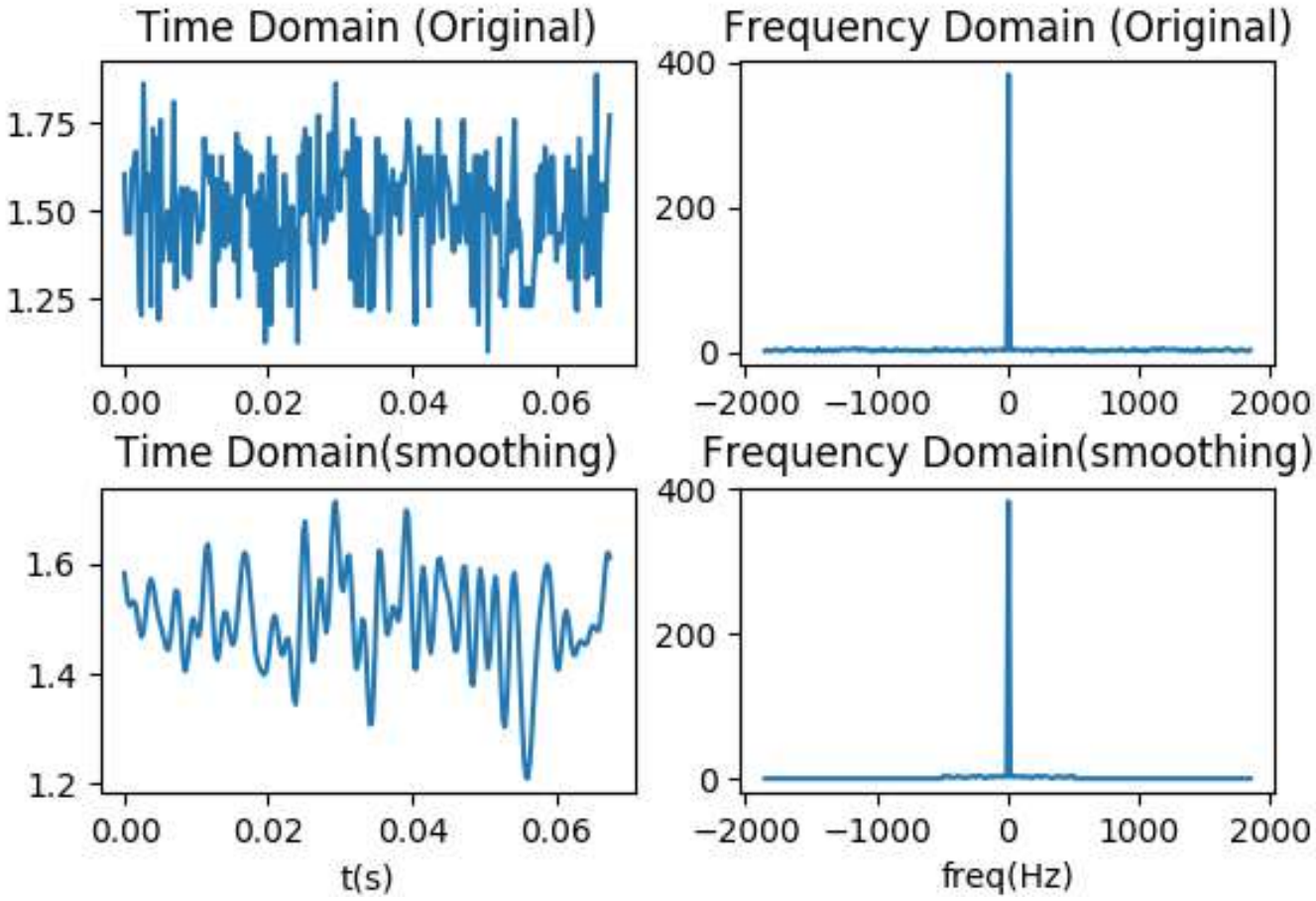
树莓派实现滤波

- 100Hz方波（500Hz低通滤波）：



树莓派实现滤波

- 白噪声（500Hz低通滤波）：



文件列表

1. ADC0832.py——AD转换
2. fft.py——对采集到的数字信号做快速傅立叶变换并显示频谱
3. DAC_TLC5620.py——DA转换
4. playback.py——信号回放
5. smoothing.py——频域滤波并回放

参考资料

1. TLC5620 Datasheet

2. 频域信号处理——用python做科学计算，
http://old.sebug.net/paper/books/scipydoc/frequency_process.html

3. 关于python插值，http://blog.sina.com.cn/s/blog_5ea41d19010127s3.html

4. File: Fourier series and transform.gif - Wikipedia，
https://en.wikipedia.org/wiki/File:Fourier_series_and_transform.gif

实验内容

- 完成对正弦波的采样和重构，验证采样定律
  - 设置信号发生器输出0-3.3V之间不同频率的正弦波，进行定时采样并通过DAC描绘出波形，验证采样定律，观察欠采样时的混叠现象。
- 完成对正弦波的滤波
  - 设置信号发生器输出0-3.3V之间不同频率的正弦波，进行定时采样，在频域上滤波后输出，通过DAC描绘出波形，观察频域滤波的效果。

实验报告中需要回答的问题

1. ADC和DAC的参考电压各起什么作用？
2. 方波的最大频率成分是什么？

致谢

- 本课件由以下同学协助编写
  - 方安然(14307130370)