ANALOG DEVICES
超越一切可能™

虚拟电子实验室：如何使用Python编程语言和ADALM2000创建示波器

Christian Jason Garcia，软件系统工程师，和
Arnie Mae Baes，软件系统工程师

摘要

虚拟电子实验室是一系列基于软件的仪器。这是作为软件应用实现的仿真电子实验室环境，用户可以在该环境中开展大量电子实验。功能齐全的物理实验室可能造价昂贵且难以管理。想象一下，如果能够构建一个可放入口袋的电子实验室；将带来无限可能！

本文旨在演示用户如何使用ADALM2000开发自己的虚拟实验室仪器。本文将使用Python这种简单的开源编程语言。将Python与ADALM2000相结合，可以开发多个虚拟实验室仪器，如示波器、信号发生器、数字万用表等。但本文将重点讨论一种仪器——示波器。从该仪器入手是个不错的选择，这是我们在实际电子实验室中常用的基本仪器之一。

简介

仪器仪表行业正稳定迅速地朝着虚拟化方向发展。基于软件的仪器托管在PC上，PC使用尽可能少的专业硬件将其连接到必须测量/控制的设备。该硬件通常包括用于直接将信号数字化或控制独立仪器的接插板。

虚拟仪器仪表因其灵活性、模块化和可移植性而闻名。ADI公司为客户提供适合几乎所有用例的电子模块，其中ADALM2000就是一个很好的例子。

通过ADALM2000，工程师或开发人员可根据具体需求创建自己的虚拟电子实验室。通过libm2k库，用户可以使用C++、C#或Python开发用于控制ADALM2000的软件应用。后面部分将详细讨论ADALM2000和libm2k。

什么是示波器?

示波器可用于常见电路和复杂电路的信号分析，因而是电子工程的重要组成部分。除此之外，如今的示波器能够与计算机连接，因此在示波器中捕获的信号能够以数字形式存储，供日后分析。

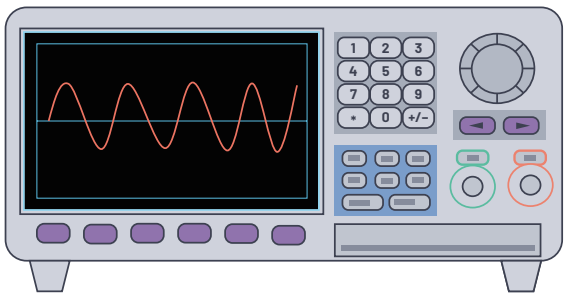

图1. 示波器示意图。

示波器用于直观呈现模拟或数字波形的电压和时间特性。前面板控件（放大器触发、扫描时间和显示屏）用于调整显示内容，以更好地直观呈现信号。

示波器向我们展示信号输入在特定时间段内的行为，这对于分析常见电路至关重要。此外，它有助于验证这些电路的功能。这也是示波器成为不可或缺的电子实验设备的主要原因。此外，我们允许工程师定制自己的示波器来满足需求，从而可以改进特定电子电路的分析。

ADALM2000是什么?

ADALM2000是主动学习模块，具有数字示波器、函数发生器、逻辑分析仪、电压表、频谱和数字总线分析仪，以及两个可编程的电源。对于基础用户或学生，可以将Scopy与ADALM2000连接。对于应用开发人员，可使用libm2k库开发应用接口。对于固件开发人员，还可以选择开发能够直接在ADALM2000上运行的定制软件或HDL。

开始使用

安装Python和PyCharm

Python是功能强大、简单易学的开源编程语言。Python可从Python官方网站下载。如果您不确定要使用哪个版本，请选择Python 3.7。

Python可在没有集成开发环境(IDE)的情况下使用，但为了更轻松地下载库和进行调试，可以使用PyCharm。PyCharm是一个IDE，为开发人员提供多个必需的工具，因而是用于Python开发的热门IDE。在JetBrains官方网站下载最新版PyCharm Community。

安装库

Python库包含可用于特定应用的方法或函数。在本文中，我们将使用libm2k、matplotlib和NumPy。

Libm2k

若要使用Python与ADALM2000交互，您需要安装libm2k库。这是C++库，带有可用于Python、C#、MATLAB®和LabVIEW®的绑定，具备以下功能：

- **AnalogIn**用于示波器或电压表。我们将重点介绍该功能。
- **AnalogOut**用于信号发生器。
- **Digital**用于逻辑分析仪或模式发生器。
- **PowerSupply**用于恒压发电机。
- **DMM**用于数字万用表。

有关该库的详细信息，请访问libm2k维基百科页面。

安装Libm2k

安装该库的一种方法是按照以下步骤操作：

- 转到发布页面。
  - 下载该库的最新可执行版本。
    示例：Libm2k-0.4.0-Windows-Setup.exe
- 运行可执行文件。当“设置”窗口提示您选择其他任务时，请务必选择**安装libm2k Python绑定**。

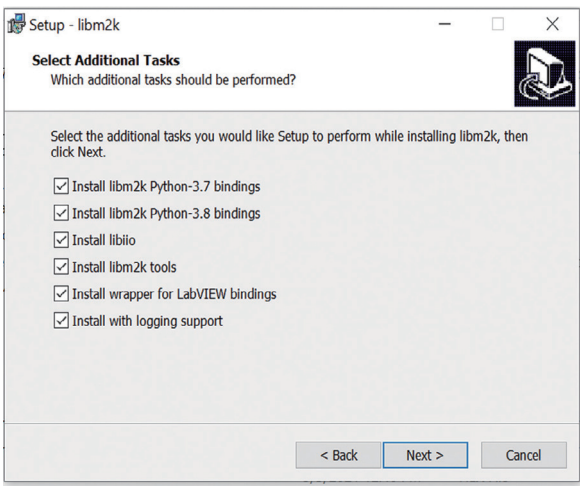

图2. Libm2k安装窗口。

- 安装结束。Libm2k将安装在Python的默认环境中。

Matplotlib

若要创建示波器显示，您需要使用matplotlib库。该库备受欢迎且易于使用，用于在Python中定制和显示可视化内容。有关该库的详细信息，请访问matplotlib网站。

NumPy

简单的示波器仍将需要大量数学计算。NumPy库可以为复杂的计算提供简单的函数。有关该库的详细信息，请访问NumPy网站。

安装Matplotlib和NumPy

若要安装matplotlib和NumPy，请在PyCharm中按照以下步骤操作：

- 转到“文件”>“设置”>“项目解释器”。
- 点击“设置”窗口右侧的+图标。
- 将出现“可用软件包”窗口。在搜索框中，搜索matplotlib和NumPy。
- 指定要安装的版本（选择最新版本）。
- 点击**安装软件包**按钮。

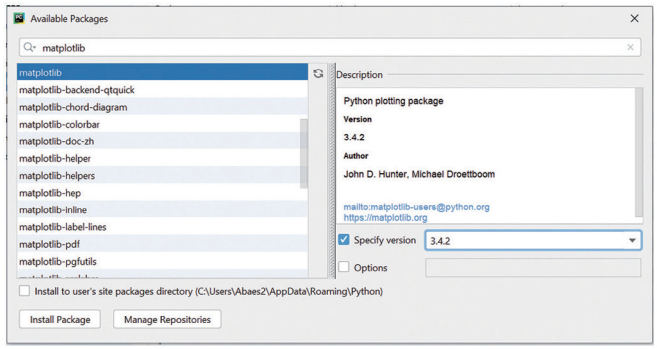

图3. 在PyCharm中安装库包。

硬件设置

在开始编码前，我们先设置硬件组件。需要使用以下硬件组件：

- 信号源（或信号发生器，如适用）
- ADALM2000
- 探头和限幅器

如果信号发生器可用，请按照图4中显示的配置，使用探头和/或限幅器将ADALM2000设备连接到通道1和通道2。

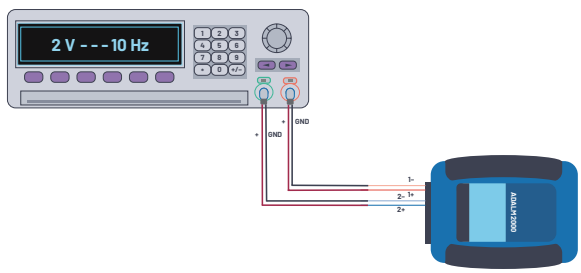

图4. 使用信号发生器和ADALM2000的实际设置。

表1. 引脚配置

信号发生器	ADALM2000
Ch1 正极引线 (+)	1+
Ch1 地	1–
Ch2 正极引线 (+)	2+
Ch2地	2–

对于其他可用的信号源，您也可以遵循相同配置。最后，通过USB端口将ADALM2000设备连接到PC。

简单的虚拟示波器

在这一部分，我们将逐个代码块介绍程序。我们还将讨论代码的作用，并说明以这些方式编写代码的原因。我们将在随后的部分中演示其他示例，在这些示例中，我们会修改基础代码，以添加更多功能，从而满足开发人员用例要求。

首先，导入将用于开发虚拟示波器的三个库（libm2k、matplotlib和NumPy）。

```
import libm2k
import matplotlib.pyplot as plt
import numpy as np
```

统一资源标识符(URI)是连接到PC的每个ADALM2000的唯一标识符。该代码块确保ADALM2000连接到PC。如果没有ADALM2000设备插入PC，代码将自动退出。

```
# Detect ADALM2000 device connected to PC
uri = libm2k.getAllContexts()
if uri == ():
  print("No ADALM2000 found. Please replug ADALM2000 device.")
  exit(1)
else:
  print("Successfully connected to ADALM2000.")
```

通过检测到的URI连接到ADALM2000。“**uri[0]**”是在连接了多个设备的情况下检测到的第一个ADALM2000设备的URI。

```
# Connect to ADALM2000 with the detected uri
adalm2000_dev = libm2k.m2kOpen(uri[0])
```

对ADC和DAC运行校准。这是确保我们将获得准确测量的重要步骤。

```
# Run the calibration for ADC and DAC
adalm2000_dev.calibrateADC()
adalm2000_dev.calibrateDAC()
```

设置采样速率和时长。可用采样速率有1 kHz、10 kHz、100 kHz、1 MHz、10 MHz和100 MHz。采样速率是在1秒内获得样本的次数，时长是获得这些样本的持续采样时间。例如，如果将采样速率设为1000，时长设为3，那么每秒将获得1000个样本，并持续采样3秒。因此，共有3000个样本。

```
# Set the sample rate and time duration
sample_rate = 1000  # Hz or sample/sec
duration = 3  # seconds (time duration of our data)
```

启用并将通道1设置为示波器的模拟输入。

```
# Enable and setup channel 1 as analog input for our oscilloscope
ocsi = adalm2000_dev.getAnalogIn()
ocsi.setSampleRate(sample_rate)
# Channel 1
ocsi.enableChannel(libm2k.ANALOG_IN_CHANNEL_1, True)
ocsi.setRange(libm2k.ANALOG_IN_CHANNEL_1, -10, 10) # range of voltage (from -10 to 10)
```

Linspace用于创建等间距的样本阵列。我们将使用该NumPy函数创建时间x轴数据阵列。该函数的第一和第二个参数分别表示阵列的起始和结束值。最后一个参数是希望在起始和结束值范围内生成的样本数。

在该示例中，起始值是0，结束值是设置的时长，也就是3。对于样本数，将**duration**与**sample_rate**相乘，即可获得所需的总样本数，也就是3000个样本。这3000个样本将均匀放置在0和3之间。该数组将存储在**time_x**中。

data_y存储我们使用ADALM2000设备收集的波形样本。通道1的样本存储在**data_y[0]**中，通道2的样本存储在**data_y[1]**中。为了显示精确的波形频率，必须使用与**time_x**相同的样本数量。

```
# x-axis data
time_x = np.linspace(0, duration, (duration * sample_rate))

data_y = ocsi.getSamples(duration * sample_rate)
```

创建我们将处理的图形。**plt.subplots**函数将返回图形对象（存储在**fig**中）和轴对象（存储在**ax**中），这些对象将用于自定义整个图形。

我们可以添加网格线，作为波形的参考坐标。添加轴标签和y限制，以添加有关图形的更多细节。

```
# Create the figure that we will manipulate
fig, ax = plt.subplots()
plt.plot(time_x, data_y[0])
ax.grid()
ax.set_xlabel("Time (s)")
ax.set_ylim([-4, 4])
ax.set_ylabel("Voltage")
```

显示图形。

```
plt.show()
```

在代码末尾销毁上下文。

```
libm2k.contextClose(adalm2000_dev)
```

运行代码，将会看到类似图5的图形。

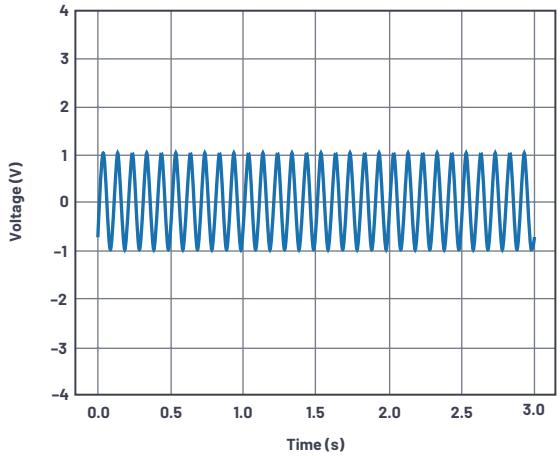

图5. 单通道正弦波输出；一个信号发生器输出：10 Hz，2 V p-p。

双通道虚拟示波器

在这一部分，我们将使用上一部分中的代码，并添加更多代码块，以创建双通道虚拟示波器。

若要添加另一个通道，请复制**ocsi.enableChannel**和**ocsi.setRange**行代码，并将第一个参数从**libm2k.ANALOG_IN_CHANNEL_1**更改为**libm2k.ANALOG_IN_CHANNEL_2**。

```
# Enable and setup channel 1 and 2 as analog input for our oscilloscope
ocsi = adalm2000_dev.getAnalogIn()
ocsi.setSampleRate(sample_rate)
# Channel 1
ocsi.enableChannel(libm2k.ANALOG_IN_CHANNEL_1, True)
ocsi.setRange(libm2k.ANALOG_IN_CHANNEL_1, -10, 10) # range of voltage (from -10 to 10)
# Channel 2
ocsi.enableChannel(libm2k.ANALOG_IN_CHANNEL_2, True)
ocsi.setRange(libm2k.ANALOG_IN_CHANNEL_2, -10, 10) # range of voltage (from -10 to 10)
```

在创建图形时，为通道2添加另一个图形。通道2的数据在**data_y[1]**阵列中。我们也可以自定义两个图形的颜色，以便轻松区分二者。在该示例中，通道1使用浅珊瑚色，通道2使用钢蓝色。

```
# Create the figure that we will manipulate
fig, ax = plt.subplots()
plt.plot(time_x, data_y[0], color='lightcoral') # channel 1 plot
plt.plot(time_x, data_y[1], color='steelblue') # channel 2 plot
ax.grid()
ax.set_xlabel("Time (s)")
ax.set_ylim([-4, 4])
ax.set_ylabel("Voltage")
```

运行代码，应该会得到类似图6的结果。

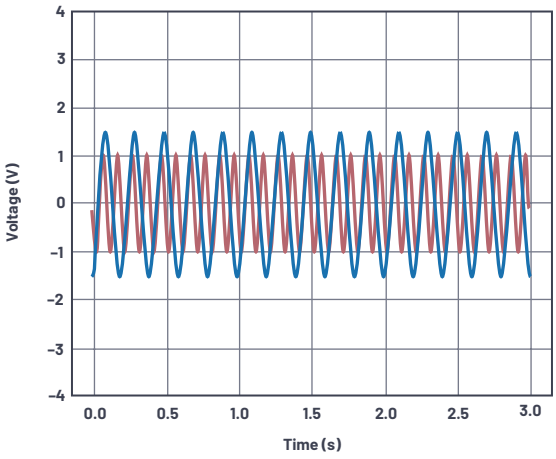

图6. 双通道正弦波输出。通道1信号发生器输出：10 Hz，2 V p-p；通道2信号发生器输出：5 Hz，3 V p-p。

虚拟示波器的其他功能

在这一部分，我们将为虚拟示波器添加其他功能，以提升交互性。Matplotlib提供我们可以使用的多个小部件。在该示例中，我们将使用文本标签和滑块小部件。我们还将继续使用上一部分中的代码。

为matplotlib滑块添加另一次导入。

```
import libm2k
import matplotlib.pyplot as plt
import numpy as np
from matplotlib.widgets import Slider
```

将时间和数据阵列转换为NumPy阵列。在下一个代码块进行的计算中，将使用这些阵列。

```
# x-axis data
time_x = np.linspace(0, duration, (duration * sample_rate))
# y-axis data
data_y = ocsi.getSamples(duration * sample_rate)

# Convert to numpy arrays
data_y_np1 = np.array(data_y[0]) # data from ch1
data_y_np2 = np.array(data_y[1]) # data from ch2
time_x_np = np.array(time_x) # time data for x axis
```

获取所有波形数据后，提取这些波形的特性将不在话下。在以下代码块中，我们从获取的两个通道的数据中提取了V_{pp}、V_{ave}和V_{rms}。要计算V_{pp}，将data_y numpy阵列中找到的最大值和最小值的绝对值相加。要计算V_{ave}，只需用V_{pp}除以pi。要计算V_{rms}，用V_{pp}除以2乘以$\sqrt{2}$。

```
# Compute for Vpp, Vave, and Vrms
v_pp_1 = abs(np.min(data_y_np1)) + abs(np.max(data_y_np1))
v_ave_1 = v_pp_1 / np.pi
v_rms_1 = v_pp_1 / (2 * np.sqrt(2))

v_pp_2 = abs(np.min(data_y_np2)) + abs(np.max(data_y_np2))
v_ave_2 = v_pp_2 / np.pi
v_rms_2 = v_pp_2 / (2 * np.sqrt(2))
```

该代码块与前面部分类似。唯一的区别是，我们为图形使用NumPy阵列，而不是使用原始阵列。我们还根据图形创建了波形对象。稍后我们将使用这些对象。

```
# Create the figure and the waveforms that we will manipulate
fig, ax = plt.subplots()
wave1, = plt.plot(time_x_np, data_y_np1, color='lightcoral') # channel 1 plot
wave2, = plt.plot(time_x_np, data_y_np2, color='steelblue') # channel 2 plot
ax.grid()
ax.set_xlabel("Time (s)")
ax.set_ylim([-4, 4])
ax.set_ylabel("Voltage")
```

为了在图形中显示计算的V_{pp}、V_{ave}和V_{rms}，我们将利用matplotlib库中的文本标签小部件。创建字符串标签**label_ch1**和**label_ch2**，然后连接这两个字符串，以创建最终标签**fin_label**。我们将使用**plt.text**创建文本标签。第一和第二个参数**(0.2, 3)**是文本的x和y位置。第三个参数是要显示的字符串。第四和第五个参数分别是文本和框的样式。

```
# Make a text label at the top of graph to show the computed Vpp, Vave and Vrms
label_ch1 = "Channel 1 : Vpp = {:.2f} Vave = {:.2f} Vrms = {:.2f}".format(v_pp_1, v_ave_1, v_rms_1)
label_ch2 = "\nChannel 2 : Vpp = {:.2f} Vave = {:.2f} Vrms = {:.2f}".format(v_pp_2, v_ave_2, v_rms_2)
fin_label = label_ch1 + label_ch2
plt.text(0.2, 3, fin_label, style='italic', bbox={'facecolor': 'paleturquoise', 'alpha': 0.5, 'pad': 5})
```

接下来，我们创建偏移滑块。该滑块用于调整波形的参考电平。将主图形向左调整，为滑块留出空间。**plt.axes**定义滑块的尺寸、位置和表面颜色。**Slider**函数用于为偏移滑块创建具有特定特性的对象。

```
# Adjust plot position so we can place the slider
plt.subplots_adjust(left=0.2)

# Make a vertically oriented slider to control the offset
ax_offset = plt.axes([0.05, 0.2, 0.0225, 0.63], facecolor='lemonchiffon')
offset_slider = Slider(ax=ax_offset, label="Offset", valmin=-2, valmax=2,
                       valinit=0, orientation="vertical")
```

创建**update_offset**函数，并将其注册到**offset_slider**对象。每次更改滑块的值时，该函数都会向波形添加偏移量。

```
# The function to be called anytime a slider's value changes
def update_offset(val):
    wave1.set_ydata(data_y_np1 + offset_slider.val)
    wave2.set_ydata(data_y_np2 + offset_slider.val)
    fig.canvas.draw_idle()

# Register the update_offset function with each slider
offset_slider.on_changed(update_offset)
```

运行代码，将会看到类似图7的图形。

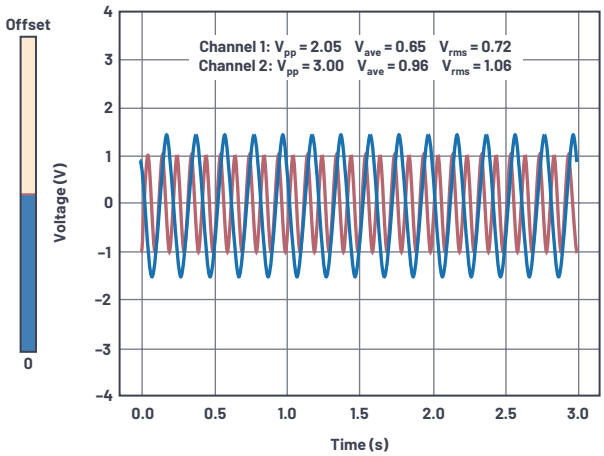

图7. 带偏移滑块的默认双通道正弦波输出。

尝试使用滑块调整偏移量。将会看到波形实时上下移动。

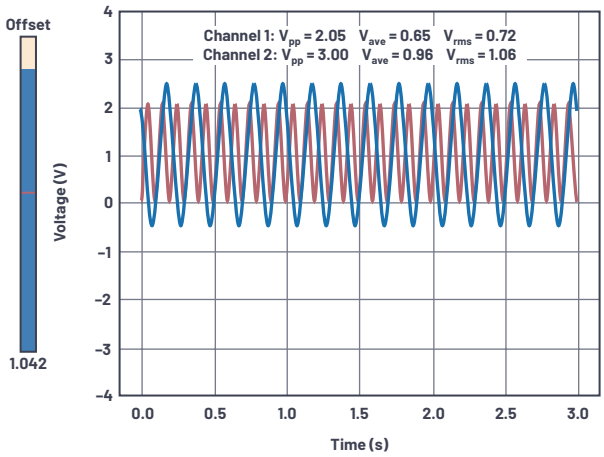

图8. 调整偏移量滑块（向左滑动），用于调整两个通道输出的偏移量。

总结

本文解释了拥有虚拟电子实验室的重要性和便利性。文中还演示了如何使用ADALM2000和Python开发虚拟示波器。讨论了软件要求和硬件设置，并提供了3个示例。

参考资料

“ADALM2000概述。”ADI公司，2021年3月。

Bhunia、Chandan、Saikat Giri、Samrat Kar和Sudarshan Haldar。“基于PC的低成本虚拟示波器。”*IEEE教育论文集*，第47卷第2期，2004年5月。

“Limb2k示例：analog.py。”ADI公司

Tegen, Amelia和Wright, Jeremy。“示波器：数字替代版：与模拟示波器相比，数字示波器的测量、瞬态捕捉和数据存储能力显著改进。”*IEEE Potentials*，第2卷，1983年。

“什么是libm2k？”ADI公司，维基百科，2021年4月。

作者简介

Arnie Mae Baes于2019年12月加入ADI公司，担任固件工程师。在进入公司的第一年，她重点负责GUI和固件开发。2020年12月，她加入消费电子软件工程部门，现在重点负责固件测试开发。她毕业于菲律宾八打雁国立大学，获电子工程学士学位。联系方式：arniemae.baes@analog.com。

Christian Jason Garcia是ADI公司的一名固件验证工程师，工作地点在菲律宾垂亚斯将军城。他拥有圣托马斯大学电子和通信工程学士学位，于2018年11月加入ADI公司。他在电动交通部门专门负责SmartMesh网络的软件测试和系统验证。联系方式：christian.garcia@analog.com。

在线支持社区

ADI EngineerZone™ 中文技术论坛

访问ADI在线支持社区，与ADI技术专家互动。提出您的棘手设计问题、浏览常见问题解答，或参与讨论。

请访问ez.analog.com/cn

TA23166sc-3/22(A)