MT-DAQ 上手指南

MANGOTREE

一、 产品概述

数据采集(DAQ)是测量电压、电流、温度、压力或声音等电子或物理现象的过程。DAQ系统由传感器、DAQ测量硬件和装有可编程软件的计算机组成。

每个数据采集任务都有其一系列的要求，为了满足不同的需求，MangoTree为基于PC的系统提供了即插即用型DAQ设备，也提供了可配置的系统。我们的通用MT-DAQ驱动程序专门针对MT数据采集设备进行了扩展，并支持Python、LabVIEW、C#等编程方式，同时兼容Windows、MAC OS、Linux三大系统平台。

二、 开发环境搭建

*如果您购买的MT-DAQ产品系统为Windows版本，出厂会预装好全部开发所需的软件及驱动，用户无需自行安装即可上手开发。

*如果您购买的MT-DAQ产品系统为Linux RT或雷电版本，则需要自行在PC下搭建开发环境，具体参照如下步骤：

注意：下面所有的安装过程，确保电脑没有开启任何杀毒软件及安全助手，以免导致安装失败。

步骤一：MT-Master软件安装

MT-Master下载链接：

http://server.mangotree.cn:9000/Software/MangoTree/MT-Master/，选择对应系统的安装包进行下载安装。

步骤二：第三方软件安装

Python相关（针对使用Python编程开发的用户）

为了执行Python代码，您需要安装一个Python解释器。

*推荐用户安装Anaconda。Anaconda是一个python的开源第三方发行版，包含了python、环境及包管理器conda和大量python工具包，安装包较大。如果安装方便，推荐使用此版本。

Anaconda下载链接：

http://server.mangotree.cn:9000/Software/Python/

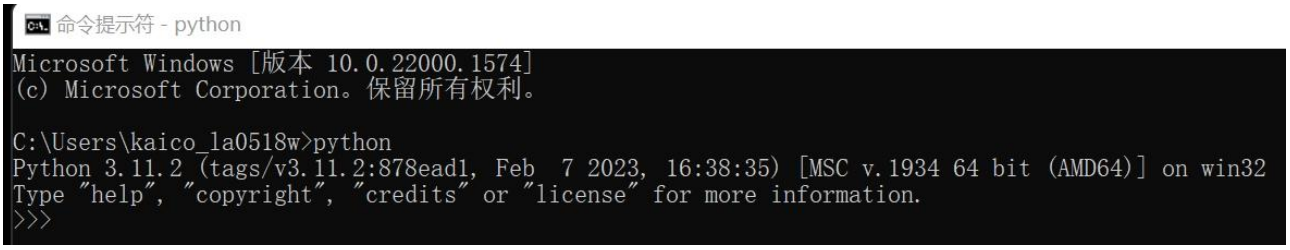
可以通过打开命令行，执行如下命令：

python #Win

或

Python3 #MacOS/Linux

来进入Python交互模式。若能成功进入，则表示已安装成功，如下图所示：



步骤三：mtdaq模块安装

方法一：

执行命令行pip install mtdaq自动下载安装

安装成功如下图所示：

方法二：

mtdaq模块安装包下载链接：

http://server.mangotree.cn:9000/Software/MangoTree/MT-DAQ/Python/mtdaq/

打开命令行并切换到whl文件所在的目录，执行如下命令：

pip install mtdaq-1.0.7-py3-none-any.whl

注：mtdaq-1.0.7对应mtdaq的版本号，根据下载的版本号修改

进行离线安装，安装成功如下图所示：

```
Anaconda Powershell Prompt (Anaconda3)
(base) PS C:\Users\kaico_la0518w> python
Python 3.9.13 (main, Aug 25 2022, 23:51:50) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> quit()
(base) PS C:\Users\kaico_la0518w> cd ./Desktop
(base) PS C:\Users\kaico_la0518w\Desktop> pip install .\mtdaq-1.0.2-py3-none-any.whl
Defaulting to user installation because normal site-packages is not writeable
Looking in indexes: https://pypi.tuna.tsinghua.edu.cn/simple
Processing c:\users\kaico_la0518w\desktop\mtdaq-1.0.2-py3-none-any.whl
Installing collected packages: mtdaq
Successfully installed mtdaq-1.0.2
(base) PS C:\Users\kaico_la0518w\Desktop>
```

注意：Python2已经停止维护，mtdaq模块不支持Python2，请使用Python3。Python3.8是支持win7的最后一个版本。如果您的系统中也安装了Python2，请将命令pip更改为pip3。

三、 开发第一个DAQ项目

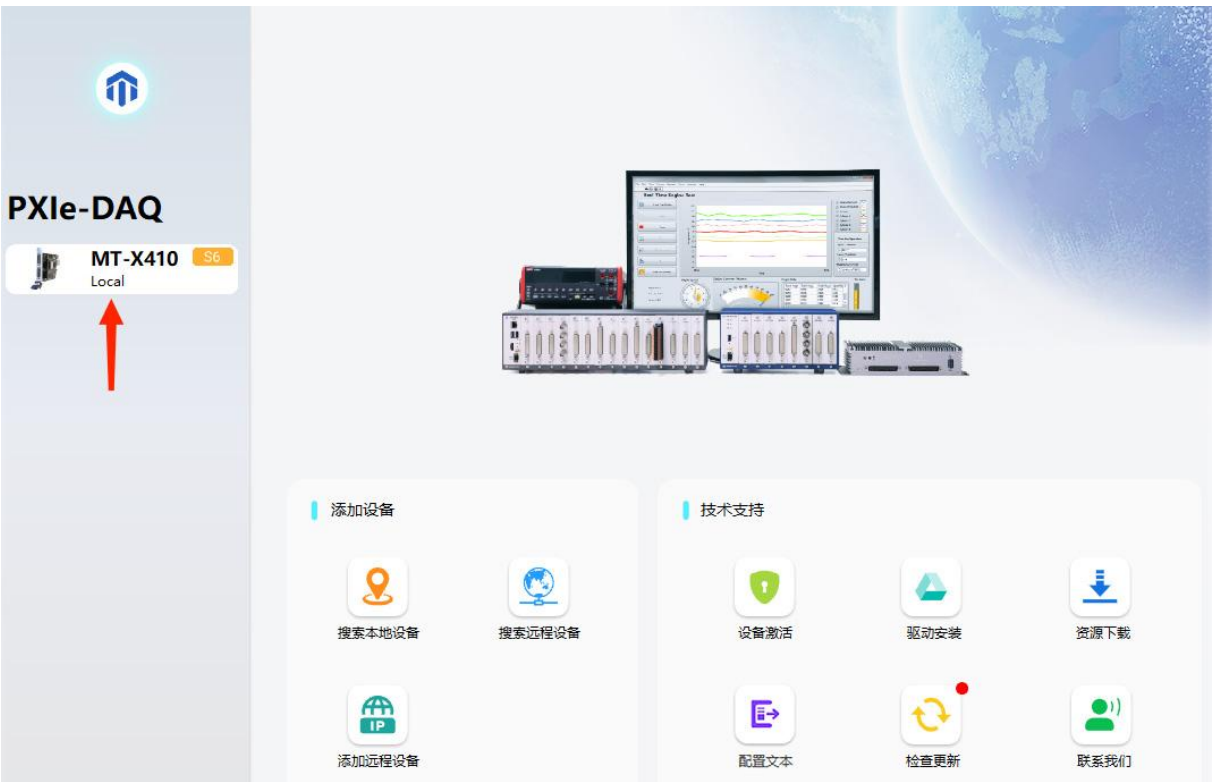
1.连接与发现设备

对于预装Windows系统的DAQ设备，在设备本地即可进行编程开发。

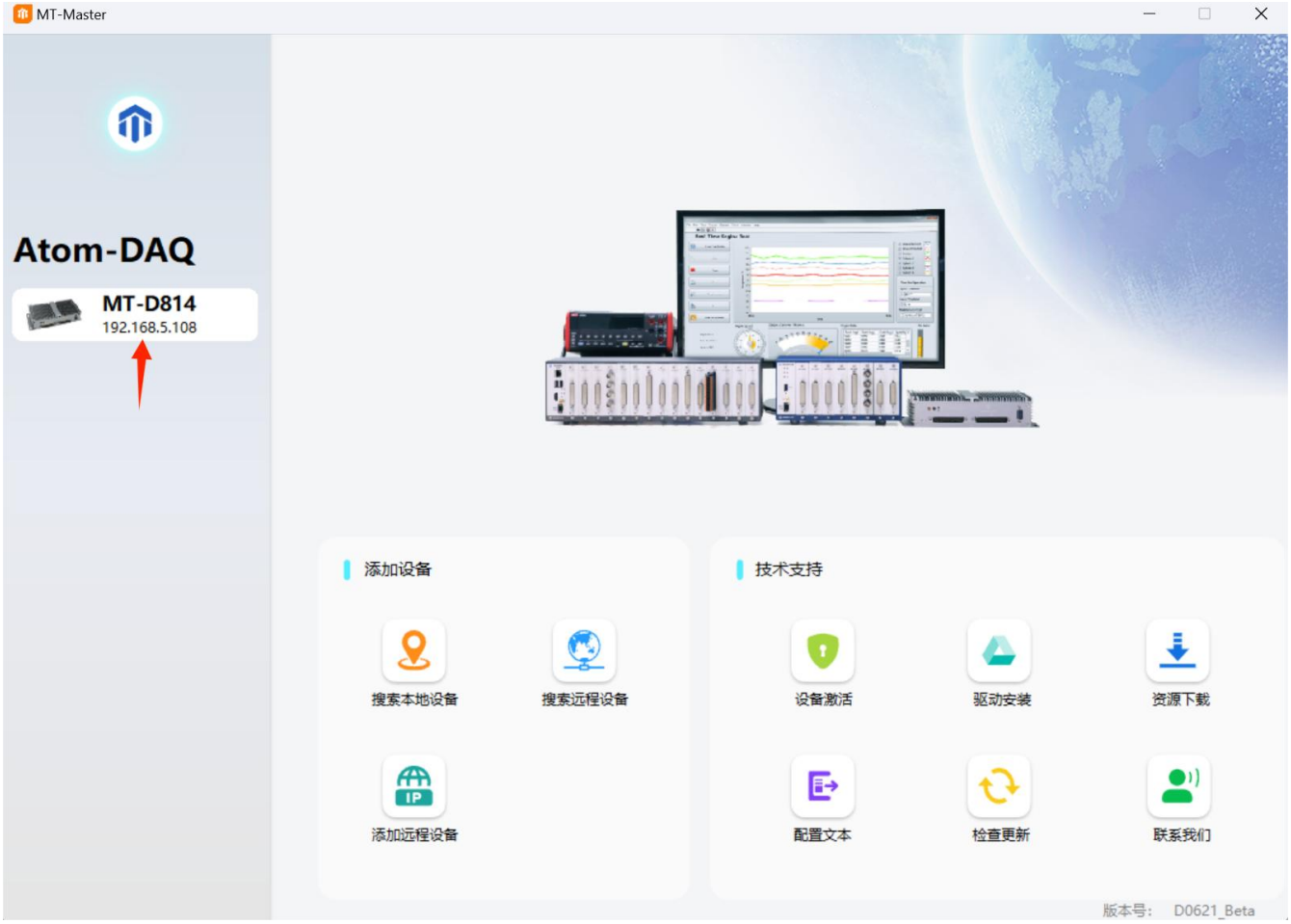
对于预装Linux RT系统的DAQ设备，用户PC通过网络发现远程设备后进行编程开发使用。

对于雷电DAQ设备，通过雷电线与PC的雷电接口连接后，在用户PC上进行编程开发。

打开MT-Master软件，Windows系统的DAQ设备及雷电DAQ设备，点击搜索本地设备，设备栏出现对应本地设备，显示为Local：



对于Linux RT系统的DAQ设备，Linux RT出厂IP设置是DHCP,用户需要确保DAQ设备的IP地址与PC主机的IP地址在同一个网段（需将PC也设置为DHCP），用户可以将DAQ设备通过网线直接连接上位机PC，或者通过交换机连接DAQ设备和PC，正确连接之后都可以在MT-Master软件中通过搜索远程设备来发现设备，设备下显示远程设备的IP地址：

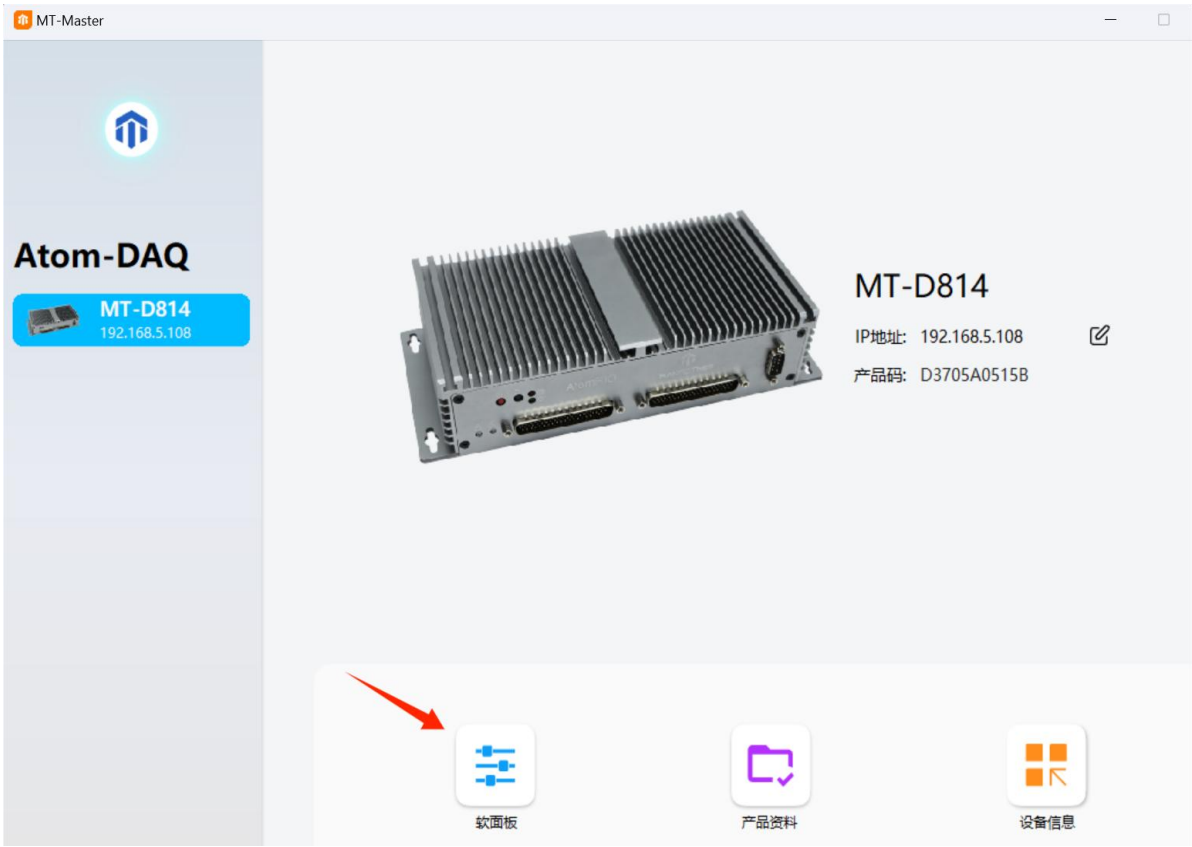


2. 激活设备

购买后初次使用MT设备，需要用MT-Master发现后进行激活，具体激活步骤参考MT-Master上手指南：http://server.mangotree.cn:9000/WebFile/Downloads/上手指南/MT-Master/；

3. 测试设备

设备激活之后用户可以自己开发使用DAQ设备，开发之前，用户也可以用设备的软面板来测试使用设备包含的基本功能，软面板在MT-Master中发现的对应设备页面下开启，如下图：



4. 使用Python开发MT-DAQ

我们假定您对Python的语法有基础的了解。如果您对Python知之甚少，可以查阅官方文档 https://docs.python.org/zh-cn/3/，或者参考网络上任意一个教程，例如 https://www.runoob.com/python3/python3-tutorial.html等。

4.1 快速开始

```
#配置参数
config="""----------MangoTreeCopyright----------
<Device>
Device-Model=D814;
Device-IP=192.168.5.77;
Device-ID=D3222A0DCDB;
Device-Slot=-1;
</Device>

<DAQMode>
DAQMode=0;
#0:General
```

```
#1:AO-Sync-AI
#2:AI-Sync-Encoder
#3:Encoder-Sync-AI
#4:DI-Trigger-AI-Sync-AO
#5:DI-Trigger-AI
#6:DI-Trigger-AO
#7:For Customize
Path=;
# Path is only for DAQMode=7
</DAQMode>

<AI>
AI-Channel=0,1;
AI-SampleRate=1000Hz;
</AI>

<AO>
AO-Channel=0,1;
AO-SampleRate=1000Hz;
</AO>

<TC>
TC-Channel=0;
TC-Type=k;                          #Thermocouple type: B/ E/ J/ K/ N/ R/ S/T
TC-CJC-Type=0;                      #Thermocouple temperature compensation mode:Internal=0/External =1
Temperature-4write/3write=0;        #Resistance Temperature Detector:4-write=0 or 3-write=1
</TC>

<RTD>
RTD-Channel=0;
RTD-4wire/3wire=0;                  #Resistance Temperature Detector:4-wire=0 or 3-wire=1
</RTD>

<DigitalWaveformInput>
DigitalWaveformInput-Channel=0;
DigitalWaveformInput-SampleRate=1000Hz;
</DigitalWaveformInput>
```

```
<DigitalWaveformOutput>
DigitalWaveformOutput-Channel=0;
DigitalWaveformOutput-SampleRate=1000Hz;
</DigitalWaveformOutput>

<DI>
DI-Channel=0;
</DI>

<DO>
DO-Channel=0;
</DO>

<Counter>
Counter-Channel=0;
Counter-Direction=0;                    # 0:RisingEdge / 1:FallingEdge
Counter-SampleRate=1000Hz;
Counter-PusleSamples=10;
</Counter>

<PWM>
PWM-Channel=0;
PWM-Frequency=1000Hz;
PWM-DutyCycle=80%;
</PWM>

<Encoder>
Encoder-Channel=0;
Encoder-SampleRate=1000Hz;
Encoder-Resolution=2000P/R;             # P/R:Plus For Every Round
</Encoder>


<DITrigger>
DITrigger-Channel=0;                    # DITrigger is only for DAQMode=4 or DAQMode=5
DITrigger-SampleClock=500000Hz;         # Clock Max is 1000000Hz(1MHz)
```

```
DITrigger-Direction=0;                    # 0:RisingEdge / 1:FallingEdge / 2:EitherEdge
DITrigger-AIOSamplePerTrigger=-1;         #-1:AIO Cotinuous Sampling When One DITrigger Comes
                                          #10:AIO Get 10 Samples For Every Single DITrigger
</DITrigger>
```

----------www.mangotree.cn----------

```
""" #生成一个频率10Hz，幅值1V的正弦波作为仿真波形用于输出
import math
freq = 10
amplitude = 1
samplerate = 1000
wave = [amplitude * math.sin(2 * i * math.pi * freq / samplerate) for i i n range(samplerate)]

#连接设备
from mtdaq import Device
dev = Device(config)
dev.start()

#AO在一个单独线程执行，输出波形10秒
def AO():
    for i in range(10):
        dev.analogWrite(wave)
import threading
threadAO = threading.Thread(target=AO)
threadAO.start()

#这些代码用于绘图
import matplotlib.pylab as pl
from IPython.display import clear_output as cls
#AI在主线程执行
for i in range(5):
    aidata = dev.analogRead(samplerate)
    #这些代码用于绘图
    pl.gca().clear()
    pl.plot(aidata)
```

```
    pl.show()
    cls(wait=True)

#确认AO线程结束后，关闭设备
threadAO.join()
dev.close()
```

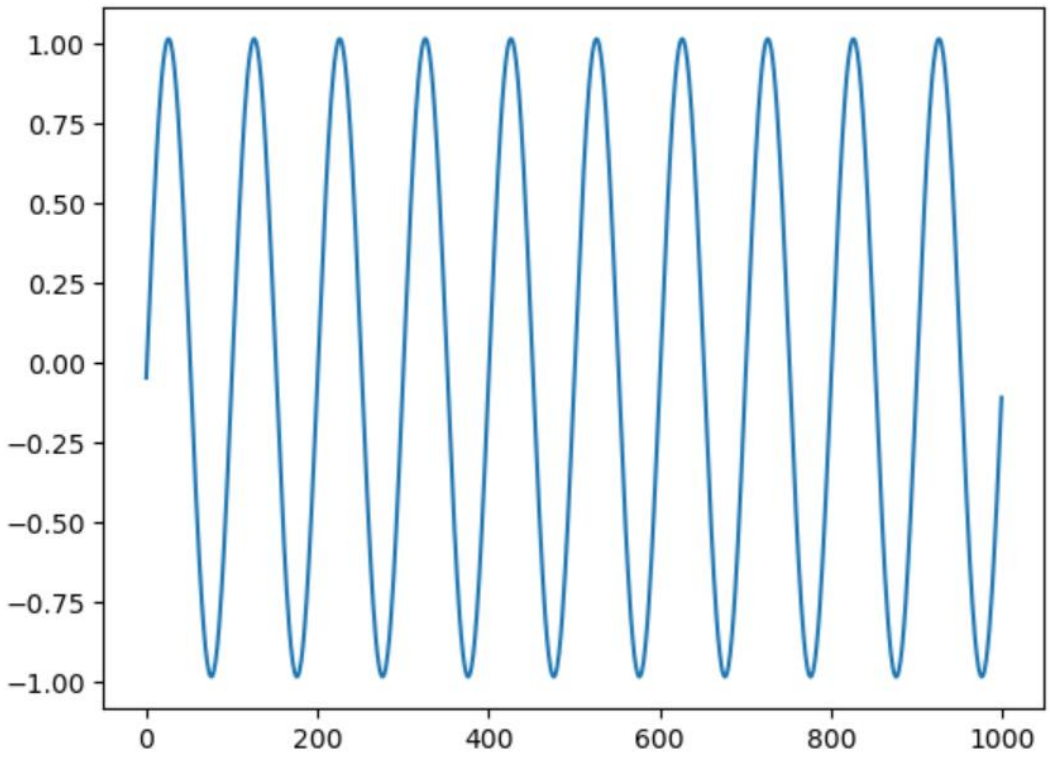
上方的示例代码推荐在Jupyter Notebook中运行以绘制图表。

如果您安装了Anaconda，Jupyter Notebook及代码依赖的模块已经包含在内了。否则，您可以执行以下命令来安装：

```
pip install jupyter
pip install matplotlib
```

*如果您想要以通常的脚本形式运行该示例，可能需要修改绘图相关代码。

这是一个通用模式下自发自收的范例。将设备的AO0与AI0进行接线，复制上方的代码到Notebook中，并修改配置参数中的IP，然后点击运行。如果设备运行正常，将绘制出如下图所示的正弦波，10秒后设备关闭。



4.2 使用mtdaq模块开发

mtdaq模块是对MT-DAQ驱动程序的一个Python封装，用意是屏蔽掉C++代码繁琐的调用细

节，直接控制MT数据采集设备，用更少的代码完成您的需求，并且能与其他python科学计算、数据分析、绘图库等无缝对接。

进入python交互模式，输入：

```
import mtdaq
```

导入正常情况下应该没有任何返回值。如果抛出ModuleNotFoundError异常，说明mtdaq模块没有安装成功。还有一种情况，即您没有安装MT-Master或安装过程出现了错误，也会抛出异常并提示。

1.Config通用配置文本

MT-DAQ设备的开发使用依赖于一个字符串配置文本，正确配置该文本才能保证设备的正常运行，不同型号的设备或板卡对应的配置参数是不同的。下文会详细介绍如何获取和修改对应设备的配置文本。

(1) Config通用配置文本模板的获取

打开MT-Master软件，切到主页中的MT-DAQ栏，点击配置文本图标，可以直接导出Config通用配置文本模板文件。点击配置文本图标，弹窗提示保存路径，用户可自定义路径保存通用配置文本文件。打开保存的文本文件，可以直接复制配置文本模板到程序框图中使用，用户根据具体设备的配置属性修改具体参数配置。

(2) 配置文本的使用及修改

MT-Master软件中导出的通用配置文本模板可以直接复制到程序中使用，并根据具体设备的配置属性修改具体参数配置。

配置文本是纯字符串类型，用户可以直接手动修改；

用户配置信息包括设备名称、设备IP地址、设备ID、PXISlot槽位号、模式；

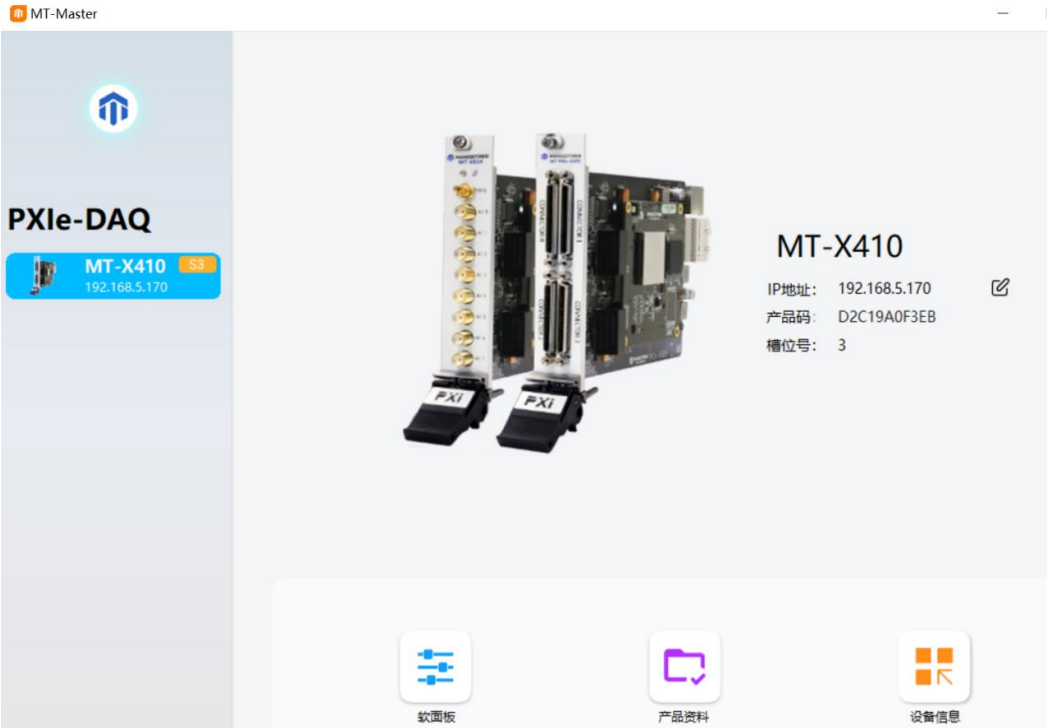
Device-Model：设备名称，用户需要填写正确的设备名称，设备名称开头都为MT-，如MT-D814，MT-X410等，MT-可以不写（如D814、X410）填写无效的设备名称将无法生成所需配置文件，会出现如下提示：

注：如果本地设备在线或者填写的IP下有远程设备，用户也可以不填设备名称，会配置对应搜索到的本地或远程设备；

Device-IP：远程DAQ设备的IP地址，如果为本地DAQ设备，则无需填写，保持为空即可；

Device-ID：DAQ设备的产品码，标识设备的代码，是保障用户权益的重要标识；

Device-Slot：DAQ设备的槽位号，在PXIe系统中，如果PXIe机箱插了多张板卡，会出现多个本地设备或同一IP下多个远程设备的情况，就需要填写Device-ID或Device-Slot来区分；



以MT-X410为例，如上图所示，在MT-Master中可以看到IP地址、设备名称、Device-ID(产品码)和Device-Slot（槽位号）。

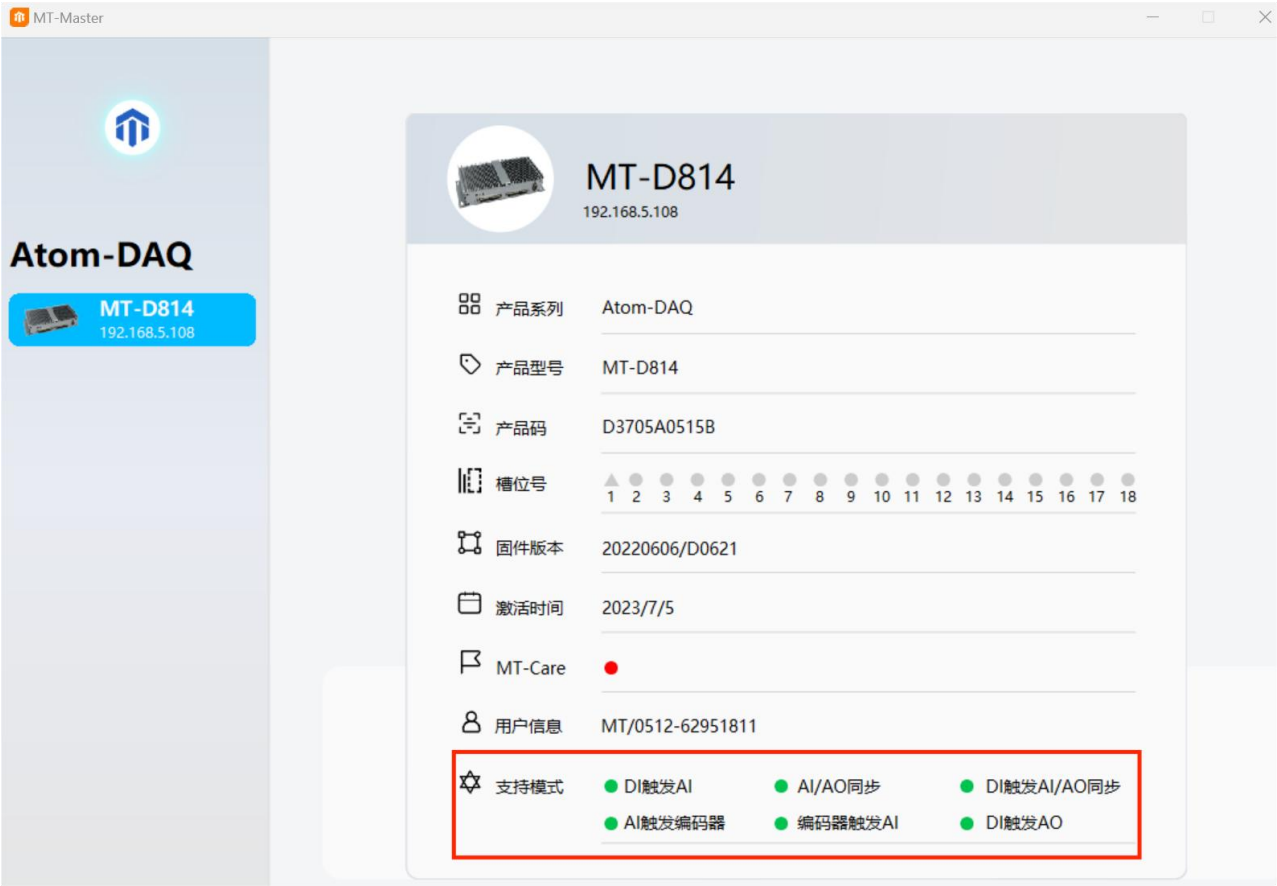
这四个参数均为字符串，可为空。如果ip不为空，会通过这四个参数对设备进行限定，成功找到设备则返回该设备的默认配置表。如果未找到设备或ip为空，则返回device_name对应设备的默认配置表。

Mode：DAQ设备模式，可以通过枚举类Mode的成员变量指定，也可以直接填写数字0-6。其中Mode=0表示为通用模式（General）。除了通用模式（General）还包括几个特殊模式：AI/AO同步（AO-Sync-AI）、AI触发编码器同步（AI-Trigger-Encoder）、编码器触发AI同步（Encoder-Trigger-AI）、DI触发AI/AO同步（DI-Trigger-AIO）、DI触发AI（DI-Trigger-AI）、DI触发AO（DI-Trigger-AO）和Customer；

```
<DAQMode>
DAQMode=0;
#0:General
#1:AO-Sync-AI
#2:AI-Trigger-Encoder
#3:Encoder-Trigger-AI
#4:DI-Trigger-AI-Sync-AO
#5:DI-Trigger-AI
#6:DI-Trigger-AO
#7:For Customize
Path=;
# Path is only for DAQMode=7
</DAQMode>
```

通用模式适用所有的MT-DAQ设备，该模式下用户可以开发使用全部设备数据手册中说明的功能，包括：AI、AO、 Counter、PWM、Encoder、Digital WaveForm Input、 Digital WaveForm Output、Digital Input、Digital Output、Temperature，不同设备支持的功能不同。

如果需要使用特殊模式下的模式，用户需要确认购买设备是否支持该模式，如下图所示，在MT-Master中的设备信息里，对应模式为绿色则支持，红色则不支持；Customer是定制DAQ模式，正常用户无法使用。如果模式配置错误，也会导致设备无法正常运行。



配置文件中<...>和</...>中间部分是用户可修改部分如：<Device>、</Device>

```
<Device>
Device-Model=X410;
Device-IP=192.168.5.170;
Device-ID=D2C19A0F3EB;
Device-Slot=3;
</Device>
```

如<AI>、</AI>是AI属性配置部分，包括通道数及采样率：

```
<AI>
AI-Channel=0,1;
AI-SampleRate=1000Hz;
</AI>
```

上图中表示开启了0，1两个通道，采样率为1000Hz。

开启多个通道有两种格式，第一种是用逗号隔开，例如：0，1，2，..；第二种是用<开始通道数字-结束通道数字>表示，例如<0-15>或<3-10>；如果读取了多个通道的数据，返回的数据是交错的，需要手动拆分。

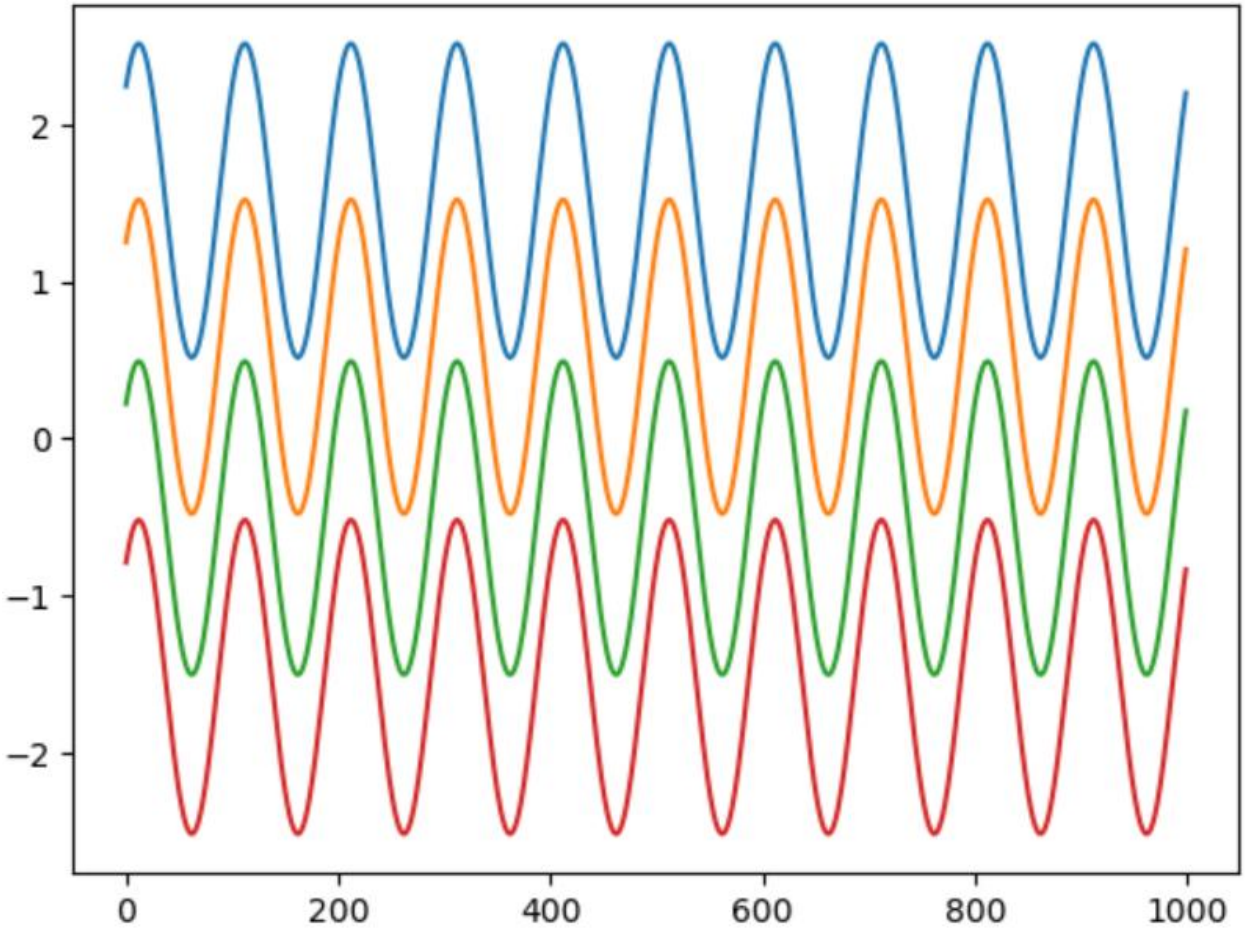
您可以执行以下命令：

e. g. ：

```
#生成一个频率10Hz，幅值1V的正弦波作为仿真波形用于输出
import math
freq = 10
amplitude = 1
samplerate = 50000
wave = [amplitude * math.sin(2 * i * math.pi * freq / samplerate) for i in range(samplerate)]
wave1 = list(map(lambda x:x+1.5, wave))
wave2 = list(map(lambda x:x+0.5, wave))
wave3 = list(map(lambda x:x-0.5, wave))
wave4 = list(map(lambda x:x-1.5, wave))
from itertools import chain
#aodata = list(chain.from_iterable(zip(*[map(lambda x: x+i, wave) for i in range(4)])))
aodata = list(chain.from_iterable(zip(wave1,wave2,wave3,wave4)))
#连接设备
from mtdaq import Device
dev = Device(config)
dev.start()


#AO在一个单独线程执行，输出波形10秒
def AO():
    for i in range(10):
        dev.analogWrite(aodata)
import threading
threadAO = threading.Thread(target=AO)
threadAO.start()

#这些代码用于绘图
import matplotlib.pylab as pl
from IPython.display import clear_output as cls
#AI在主线程执行

import numpy as np
```

```
for i in range(5):
    aidata = dev.analogRead(50000)
    aidata = np.array(aidata).reshape(samplerate, 4)

    #这些代码用于绘图
    pl.gca().clear()
    pl.plot(aidata)
    pl.show()
    cls(wait=True)

#确认AO线程结束后，关闭设备
threadAO.join()
dev.close()
```

将设备的AO0,AO1,AO2,AO3与AI0,AI1,AI2,AI3进行接线，复制上方代码到Notebook中对应位置修改，并修改配置参数中的IP，然后点击运行。若设备运行正常，将绘制出如下图所示的前四个通道的正弦波，10秒后设备关闭。



采样率是根据每张卡的硬件属性来设置的，具体信息可参考所购买产品对应的DataSheet。例如：如果是同步卡，采样率设置为200kHz，表示每个通道的采样率可最大设置为200kHz;如果是非同步卡，采样率设置为200kHz，则表示所有通道的采样率总和可最大设置为200kHz;也就是说一个通道可以设置成200kHz，但当有两个通道时，单通道设置的最大采样率就只能为100kHz。

如果是声音振动这类特殊的卡，采样率就只能设置固定的几种：192kHz、96kHz、48kHz。详细信息可以查看所购买产品对应的DataSheet。

注意：每个参数配置的结束都是以分号为结束符号的。

配置表中#开头的是注释，如：

```
#0:General
#1:AO-Sync-AI
#2:AI-Trigger-Encoder
#3:Encoder-Trigger-AI
#4:DI-Trigger-AI-Sync-AO
#5:DI-Trigger-AI
#6:DI-Trigger-AO
#7:For Customize
Path=;
# Path is only for DAQMode=7
```

如果配置文本中<...>和</...>中间没有任何参数，说明该设备没有此功能，忽略该项内容，如下图表示该设备没有AO输出：

```
<AO>
</AO>
```

AO属性配置是和AI属性配置是一样的，需要配置通道数和采样率。

数字波形输入属性配置和数字波形输出属性配置中同样需要配置通道数和采样率。如下图所示：

```
<DigitalWaveformInput>
DigitalWaveformInput-Channel=0;
DigitalWaveformInput-SampleRate=1000Hz;
</DigitalWaveformInput>

<DigitalWaveformOutput>
DigitalWaveformOutput-Channel=0;
DigitalWaveformOutput-SampleRate=1000Hz;
</DigitalWaveformOutput>
```

DI属性配置和DO属性配置中只需要配置通道个数，没有采样率配置。

<TC>、</TC>是热电偶属性配置部分，包括通道数、热电偶类型、热电偶温度补偿方式和电阻温度检测器类型：

```
<TC>
TC-Channel=0;
TC-Type=k;                        #Thermocouple type: B/ E/ J/ K/ N/ R/ S/T
TC-CJC-Type=0;                    #Thermocouple temperature compensation mode:Internal=0/External =1
Temperature-4write/3write=0;      #Resistance Temperature Detector:4-write=0 or 3-write=1
</TC>
```

上图中表示开启了0这一个通道，热电偶类型为K型，热电偶温度补偿方式为内部补偿，电阻温度检测器类型为四线制。如果需要修改，按#后面的注释手动修改对应参数。

<Counter>、</Counter>是计数器属性配置部分，包括通道数、触发方式、采样率和脉冲属性的样本数量：

```
<Counter>
Counter-Channel=0;
Counter-Direction=0;              # 0:RisingEdge / 1:FallingEdge
Counter-SampleRate=1000Hz;
Counter-PusleSamples=10;
</Counter>
```

上图中表示开启了0这一个通道；触发方式为上升沿触发；采样率设置为1000Hz，表示每一秒钟上传1000个样本，通过上传的样本可以看到脉冲计数在一段时间内的分布情况；样本数量设置为10，表示每采10个脉冲计算下占空比。如果需要修改，按#后面的注释手动修改对应参数。

<DITrigger>、</DITrigger是触发属性配置部分，包括通道数、触发采样时钟、触发方式、触发后采集的数据量和触发脉冲宽度：

```
<DITrigger>
DITrigger-Channel=0;              # DITrigger is only for DAQMode=4 or DAQMode=5
DITrigger-SampleClock=500000Hz;   # Clock Max is 1000000Hz(1MHz)
DITrigger-Direction=0;            # 0:RisingEdge / 1:FallingEdge / 2:EitherEdge
DITrigger-AIOSamplePerTrigger=-1; #-1:AIO Cotinuous Sampling When One DITrigger Comes
                                  #10:AIO Get 10 Samples For Every Single DITrigger
</DITrigger>
```

上图中表示开启了0这一个通道；触发采样时钟为500000Hz，最大为1MHz；触发方式为上升沿触发；触发后采集的数据量设置为-1，表示Trigger触发之后 AIO会连续采样； 如果填写100，则是 Trigger 每触发一次，AIO采样的个数为100。如果需要修改，按#后面的注释手动修改对应参数。

配置文件结构组成：

<Device> </Device>	设备基本信息
<DAQMode> </DAQMode>	设备模式配置
<AI> </AI>	AI属性配置
<AO> </AO>	AO属性配置
<TC> </TC>	热电偶属性配置
<RTD> </RTD>	热电阻属性配置
<DigitalWaveformInput> </DigitalWaveformInput>	数字波形输入属性配置
<DigitalWaveformOutput> </DigitalWaveformOutput>	数字波形输出属性配置
<DI> </DI>	DI属性配置
<DO> </DO>	DO属性配置
<Counter> </Counter>	计数器属性配置
<PWM> </PWM>	脉冲输出属性配置
<Encoder> </Encoder>	编码器属性配置
<DITrigger> </DITrigger>	DI触发属性配置

2. Device

```
class Device(config) #指定config实例化设备。
```

Device是mtdaq模块的主要组成。实例化一个Device对象需要的唯一参数是一个字符串类型的配置表。Device对象包含了下列方法：

```
Device.start()#启动设备
Device.close()#关闭设备

Device.analogRead(samples, timeout=2500)#读取模拟输入
Device.analogWrite(aodata, timeout=2500)#写入模拟输出
Device.digitalRead()#读取数字输入
Device.digitalWrite(dodata)#读取数字输出
Device.digitalWaveformRead(samples, timeout=2500)#读取数字波形输入
Device.digitalWaveformWrite(dodata, timeout=2500)#写入数字波形输出
Device.counterRead(samples, timeout=2500)#读取计数器输入
Device.pwmWrite(count)#写入脉冲输出
Device.encoderRead(samples, timeout=2500)#读取编码器输入
Device.temperatureRead()#读取温度输入

#以上都是读写方法，必须先启动设备才能调用。
#为了性能考虑，所有读方法的返回对象都是ctypes的数据类型，在内存中连续。
#因此如果读取了多个通道的数据，返回的数据是交错的，需要手动拆分。
#samples：需要读取的采样数量。
#timeout：超时时间默认2500ms。
#dio的数据均为uint32类型，从低到高按位对应不同通道。
```