AidoSDR Libiio_Pyiio

使用篇

Rev. 1.0

前言

libiio 是由 Analog Devices 开发的一个库，用于简化与 Linux 工业 I/O(IIO) 设备接口的软件开发。该库抽象了硬件的低级细节，并提供了一个简单而完整的编程接口，可用于高级项目。

pyiio 是 Analog Devices 提供的 Python 语言接口库，用于访问其工业 I/O 子系统(IIO)，本质上是 libiio 的 Python 绑定。它允许用户在 Python 中直接调用 libiio 的底层功能，便于控制和采集来自 ADI 硬件（如 AD9361、PlutoSDR、AD7606 等）的数据。通过 pyiio，用户可以轻松实现设备枚举、通道读写、属性配置、缓冲区管理等操作，适合快速原型开发、自动化测试与数据分析。

第 1 章 环境配置

1.1 libiio 库下载和安装

官方将 libiio 库的源码放在了 github 仓库中 https://github.com/analogdevicesinc/libiio ，这里我们通过编译的方式安装 libiio 库

1.1.1 依赖安装

sudo apt install libxml2 libxml2-dev bison flex cmake git libaio-dev libboost-all-dev

sudo apt install libusb-1.0-0-dev

sudo apt install libavahi-common-dev libavahi-client-dev

sudo apt install bison flex cmake git libgmp-dev

sudo apt install swig

sudo apt install liborc-dev

1.1.2 下载编译安装 libiio

git clone -b v0.24 https://github.com/analogdevicesinc/libiio.git

cd libiio

mkdir build

cd build

cmake ../

make

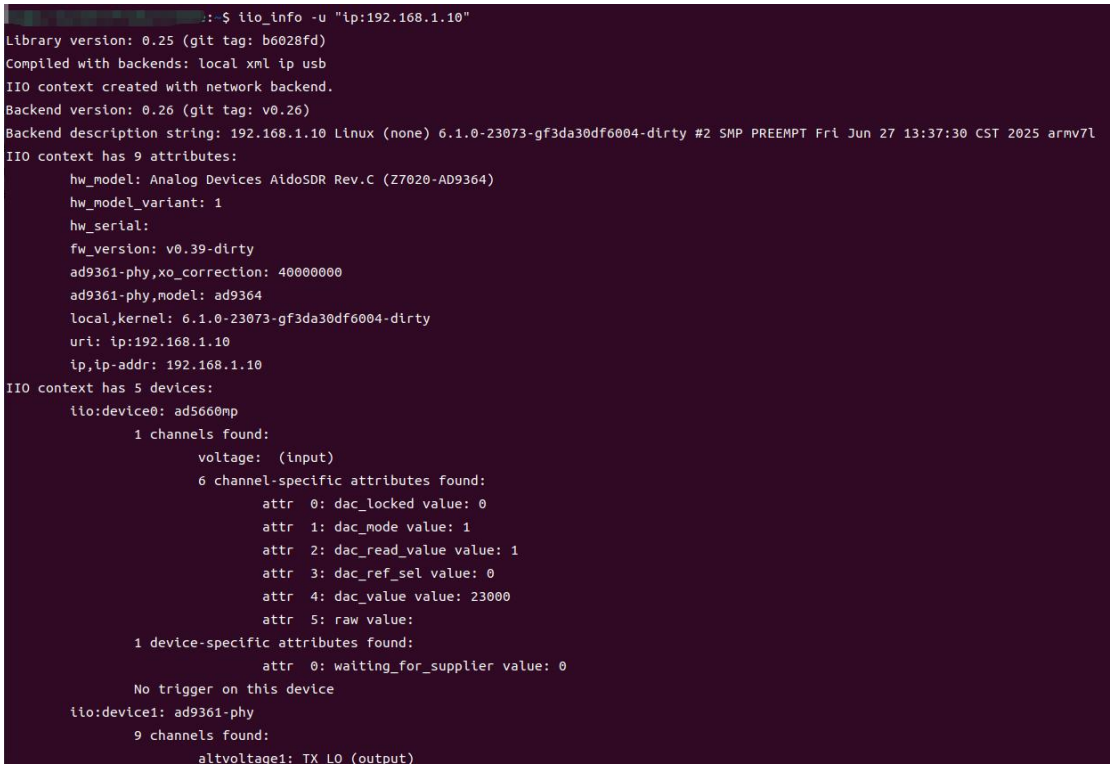
sudo make install

libiio 测试，依次执行了上面的步骤后，用户可以通过 iio 命令测试是否安装成功，首先给 AidoSDR 板子上电，并确保运行固件为 iio 固件

在主机控制台上运行下面的命令：

iio_info -u “ip:192.168.1.10”

显示如下图：（截图不全）

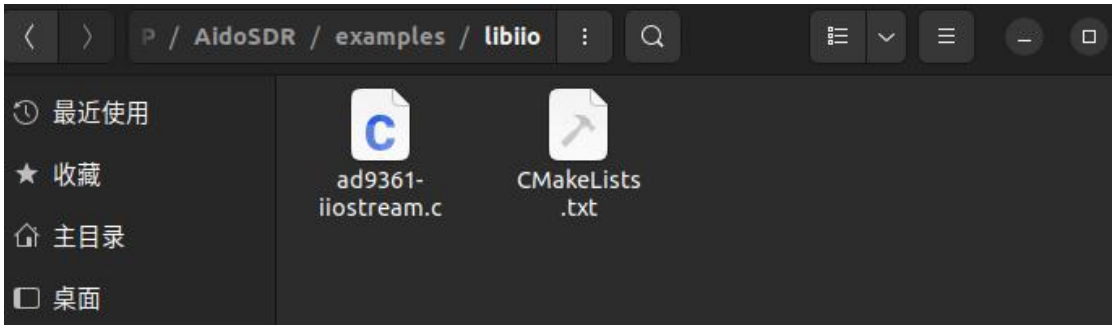


上图说明已经在主机上成功安装 libiio 库并和设备连接

第 2 章　运行例子编译运行

2.1 源码

AidoSDR 已经为用户提供了源码，这里我们需要编译 ad9361-iiostream.c 文件并运行。在该文件夹下编译源码 ad9361-iiostream.c 文件新建 build 文件夹，如图：



该例子是修改官方的例子来的，官方的例子可以在下面的文件中找到

https://github.com/analogdevicesinc/libiio/tree/main/examples

cd build

cmake ..

```
io/build$ cmake ..
-- The C compiler identification is GNU 11.4.0
-- The CXX compiler identification is GNU 11.4.0
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: /usr/bin/cc - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++ - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /media/wu/新加卷/work/BaiDuWangP/AidoSDR/ex
amples/libiio/build
```

make

```
io/build$ make
[ 50%] Building C object CMakeFiles/ad9361-iiostream.dir/ad9361-iiostream.c.o
[100%] Linking C executable ad9361-iiostream
[100%] Built target ad9361-iiostream
```

执行结果如下所示

```
/libiio/examples/build$ ./ad9361-iiostream
Acquiring IIO context
Acquiring AD9361 streaming devices
Configuring AD9361 for streaming
Acquiring AD9361 phy channel 0
Acquiring AD9361 RX lo channel
Acquiring AD9361 phy channel 0
Acquiring AD9361 TX lo channel
Initializing AD9361 IIO streaming channels
Enabling IIO streaming channels
Creating non-cyclic IIO buffers with 1 MiS
Starting IO streaming (press CTRL+C to cancel)
      RX     1.05 MSmp, TX     1.05 MSmp
      RX     2.10 MSmp, TX     2.10 MSmp
      RX     3.15 MSmp, TX     3.15 MSmp
      RX     4.19 MSmp, TX     4.19 MSmp
      RX     5.24 MSmp, TX     5.24 MSmp
      RX     6.29 MSmp, TX     6.29 MSmp
      RX     7.34 MSmp, TX     7.34 MSmp
      RX     8.39 MSmp, TX     8.39 MSmp
      RX     9.44 MSmp, TX     9.44 MSmp
      RX    10.49 MSmp, TX    10.49 MSmp
      RX    11.53 MSmp, TX    11.53 MSmp
      RX    12.58 MSmp, TX    12.58 MSmp
      RX    13.63 MSmp, TX    13.63 MSmp
      RX    14.68 MSmp, TX    14.68 MSmp
      RX    15.73 MSmp, TX    15.73 MSmp
```

该方法是直接在主机上运行，也可以在板子上运行，如果需要在板子上运行，需要用交叉编译器编译该文件，然后复制到板子上就可以运行了。

第 3 章 pyiio

pyiio 是 Analog Devices 提供的 Python 语言接口库，用于访问其工业 I/O 子系统（IIO），本质上是 libiio 的 Python 绑定。它允许用户在 Python 中直接调用 libiio 的底层功能，便于控制和采集来自 ADI 硬件（如 AD9361、PlutoSDR、AD7606 等）的数据。通过 pyiio，用户可以轻松实现设备枚举、通道读写、属性配置、缓冲区管理等操作，适合快速原型开发、自动化测试与数据分析。

libiio 是用 C 编写的核心库，是所有平台（Linux、Windows、嵌入式）下 IIO 接口的基础层，提供高性能和跨平台的数据通信能力。相比之下，pyiio 更易于使用但性能略低，适用于不要求实时性的场景。

总结来说，libiio 是底层的 C API，功能全面、效率高，而 pyiio 是其 Python 封装，更适合上层开发和脚本控制。两者功能一致，选择取决于开发语言和性能需求

pyiio 源码可以在 github 上找到

https://github.com/analogdevicesinc/pyadi-iio

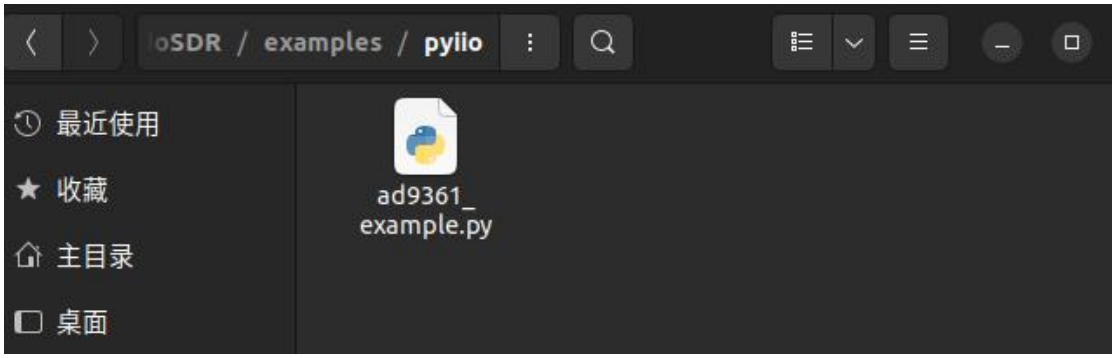
可以使用源码的方式安装 pyiio

```
tcollins@jeeves:~$ git clone https://github.com/analogdevicesinc/py
tcollins@jeeves:~$ cd pyadi-iio
tcollins@jeeves:~$ (sudo) pip install .
```

安装完成后，进入 examples 文件夹，

使用下面的代码对 AidoSDR 进行测试

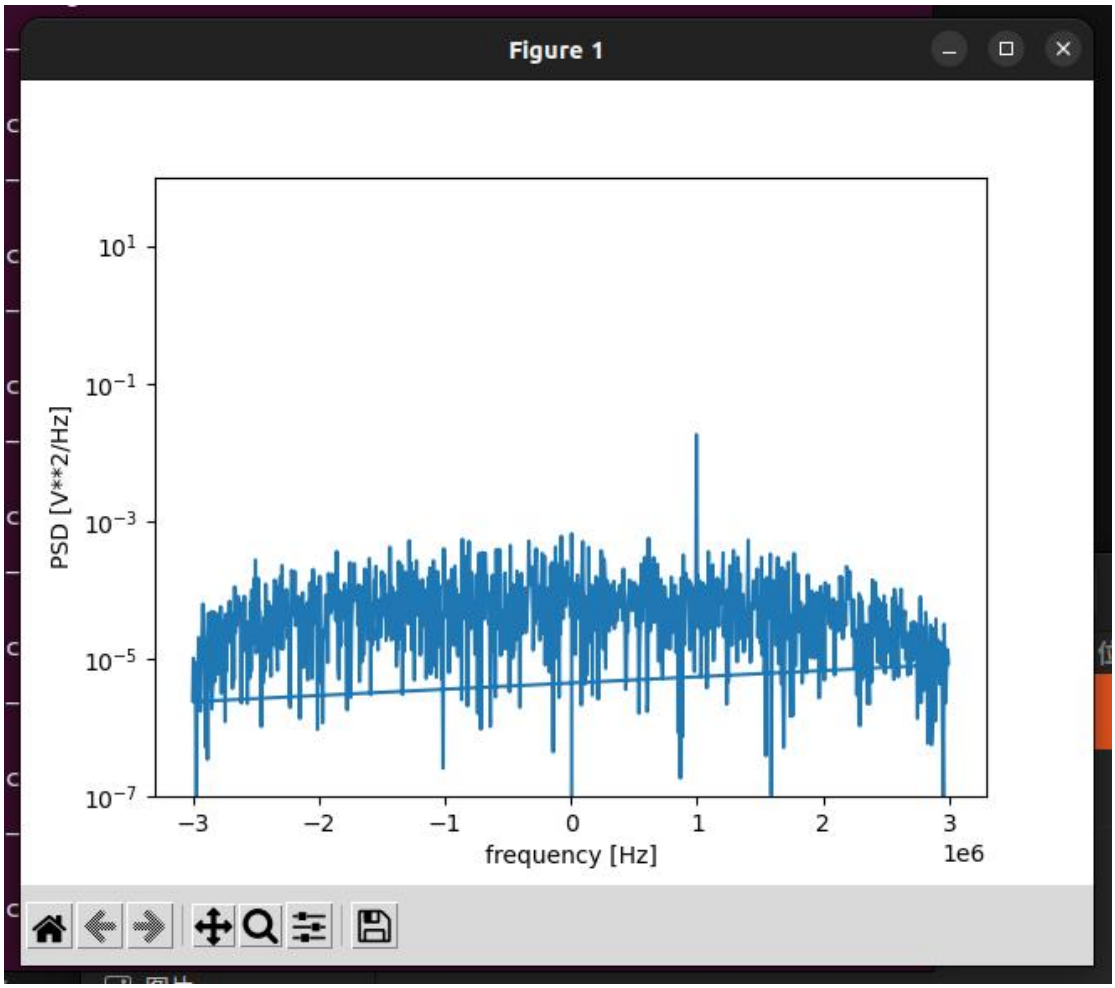
代码在资料中已经提供，如下图所示：



```
import time

import matplotlib.pyplot as plt
import numpy as np
from scipy import signal

import adi

# Create radio
sdr = adi.ad9364(uri="ip:192.168.1.10")

# Configure properties
sdr.rx_rf_bandwidth = 4000000
sdr.sample_rate = 6000000
sdr.rx_lo = 2000000000
sdr.tx_lo = 2000000000
sdr.tx_cyclic_buffer = True
sdr.tx_hardwaregain_chan0 = -30
sdr.gain_control_mode_chan0 = "slow_attack"

# Configuration data channels
sdr.rx_enabled_channels = [0]
sdr.tx_enabled_channels = [0]

# Read properties
print("RX LO %s" % (sdr.rx_lo))
```

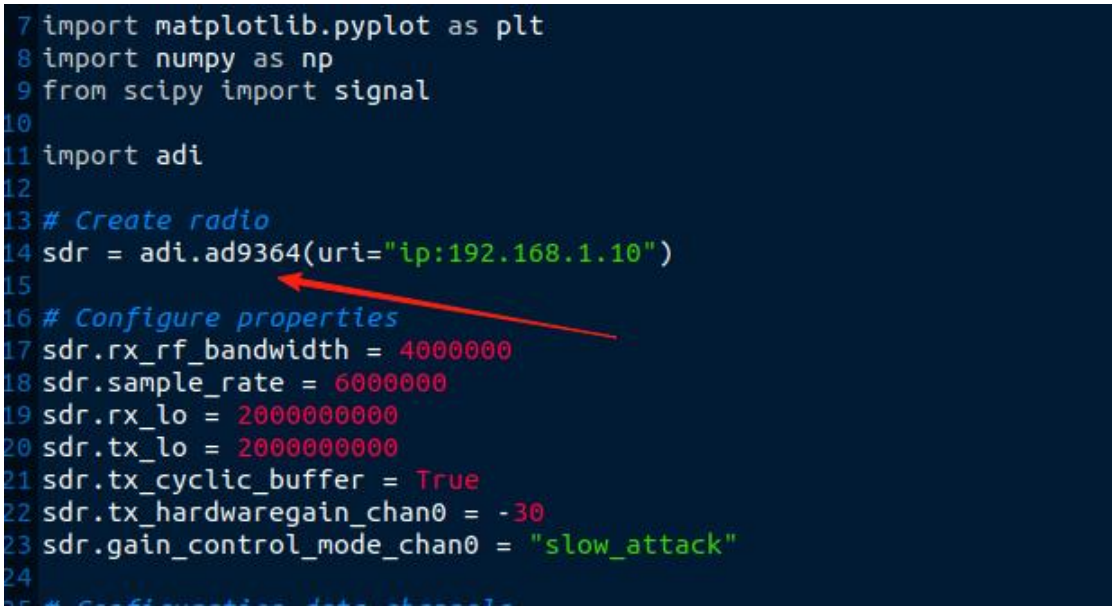
```
# Create a sinewave waveform
fs = int(sdr.sample_rate)
N = 1024
fc = int(1000000 / (fs / N)) * (fs / N)
ts = 1 / float(fs)
t = np.arange(0, N * ts, ts)
i = np.cos(2 * np.pi * t * fc) * 2 ** 14
q = np.sin(2 * np.pi * t * fc) * 2 ** 14
iq = i + 1j * q

# Send data
sdr.tx(iq)

# Collect data
for r in range(20):
    x = sdr.rx()
    f, Pxx_den = signal.periodogram(x, fs)
    plt.clf()
    plt.semilogy(f, Pxx_den)
    plt.ylim([1e-7, 1e2])
    plt.xlabel("frequency [Hz]")
    plt.ylabel("PSD [V**2/Hz]")
    plt.draw()
    plt.pause(0.05)
    time.sleep(0.1)

plt.show()
```

执行结果如下所示：



在使用 pyiio 时，可以直接使用官方 pyiio 的例子

需要对代码进行简单的修改，将此处 ad9361 修改为 ad9394,就可以编译运行了



```
import matplotlib.pyplot as plt
import numpy as np
from scipy import signal

import adi

# Create radio
sdr = adi.ad9364(uri="ip:192.168.1.10")

# Configure properties
sdr.rx_rf_bandwidth = 4000000
sdr.sample_rate = 6000000
sdr.rx_lo = 2000000000
sdr.tx_lo = 2000000000
sdr.tx_cyclic_buffer = True
sdr.tx_hardwaregain_chan0 = -30
sdr.gain_control_mode_chan0 = "slow_attack"
```