การสร้างรูปคลื่นสัญญาณแบบมีคาบโดยใช้ Rigol DG1022 Digital Function Generator และภาษา Python

เขียนโดย เรวัต ศิริโภคาภิรมย์
ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ ม.เทคโนโลยีพระจอมเกล้าพระนครเหนือ (KMUTNB)

เผยแพร่ครั้งแรก: วันที่ 26 พฤศจิกายน พ.ศ. 2560 (November 26, 2017)

ปรับปรุงแก้ไข: วันที่ 26 กันยายน พ.ศ. 2563 (September 26, 2020)

คำสำคัญ / Keywords: *Digital Function Generator*, *Python for Automated Instrumentation and Measurement*, *Rigol DG1022*

เครื่องกำเนิดสัญญาณแบบดิจิทัลและการเขียนโปรแกรมควบคุมผ่านพอร์ต USB

เครื่องกำเนิดสัญญาณ (Function Generator) สามารถสร้างสัญญาณไฟฟ้าแบบมีคาบ (Periodic Signals) ตามรูปแบบที่ต้องการได้ เช่น คลื่นไซน์ (Sinusoidal) คลื่นสี่เหลี่ยม (Rectangular) คลื่นสามเหลี่ยม (Triangular) สัญญาณแบบ PWM (Pulse Width Modulation) หรือ สร้างคลื่นสัญญาณแบบมีคาบจากชุดข้อมูลที่อัพโหลดไปยังหน่วยความจำภายในของเครื่องได้

ในปัจจุบันเครื่องกำเนิดสัญญาณที่ทำงานแบบดิจิทัลเริ่มมาแทนที่เครื่องกำเนิดสัญญาณแบบแอนะล็อก และสามารถเขียนโปรแกรมควบคุมจากคอมพิวเตอร์ ผ่านพอร์ตสื่อสารได้ อย่างเช่น RS232, USB และ RJ45/LXI เป็นต้น

บทความนี้แนะนำตัวอย่างการเขียนโปรแกรมด้วยภาษา Python เพื่อควบคุมสั่งงานเครื่องกำเนิดสัญญาณ Rigol Function Generator รุ่น DG1022 โดยการสร้างชุดข้อมูลของรูปคลื่นสัญญาณที่ต้องการ และโปรแกรมลงในหน่วยความจำภายในของเครื่องกำเนิดสัญญาณ และสร้างเป็นสัญญาณเอาต์พุตตามที่ต้องการ

รูปเครื่องกำเนิดสัญญาณ Rigol DG1022

หลักการสร้างชุดข้อมูลสำหรับเครื่อง Rigol DG1022 เพื่อสร้างเป็นสัญญาณแรงดันไฟฟ้า มีดังนี้ ผู้ใช้จะต้องสร้างอาร์เรย์ของข้อมูลแบบ integer ที่มีค่าอยู่ในช่วง 0 ถึง 16383 (ข้อมูลแต่ละตัวมีขนาด 14 บิต) ข้อมูลในอาร์เรย์จะเป็นข้อมูลสำหรับกำหนดรูปคลื่นสัญญาณหนึ่งคาบ (data samples per period) ขนาดสูงสุดของอาร์เรย์สำหรับข้อมูลหนึ่งคาบสัญญาณที่ใช้งานได้กับเครื่อง Rigol DG1022 คือ 4096 แต่จะมีขนาดเล็กกว่า 4096 ก็ได้ เช่น 512, 1024 เป็นต้น

ถ้าข้อมูลมีค่าเท่ากับ 0 จะเทียบเท่ากับค่าแรงดันต่ำ Voltage LOW และ 16383 เทียบเท่ากับ Voltage HIGH ค่าของข้อมูลแบบ integer ที่ส่งไปแต่ละตัว จะถูกแปลงเป็นแรงดันเอาต์พุตในช่วงที่กำหนด โดยวงจร DAC (Digital-to-Analog Converter) ที่อยู่ภายในเครื่องกำเนิดสัญญาณ ซึ่งใช้การแปลงแบบเชิงเส้น เช่น ถ้ากำหนดแรงดัน Voltage Peak-to-Peak (Vpp) เท่ากับ 2V ซึ่งอยู่ระหว่าง -1V และ +1V ถ้าส่งข้อมูลเท่ากับ 0 จะหมายถึง แรงดัน -1V ถ้าส่งข้อมูลเท่ากับ 8129 จะหมายถึง 0V แต่ถ้าส่งค่า 16383 จะมีแรงดันประมาณ +1V เป็นต้น

บทความนี้จะไม่ขอกล่าวถึง การเตรียมพร้อมด้านฮาร์ดแวร์และซอฟต์แวร์เพื่อใช้งาน Rigol Function Generator เพราะได้กล่าวถึงไว้ในบทความอื่นที่เกี่ยวข้องแล้ว

วัตถุประสงค์การเรียนรู้

- ทดลองใช้ Python Script ที่ให้ไว้เป็นตัวอย่างในการเชื่อมต่อและควบคุมสั่งการอุปกรณ์ผ่านพอร์ต USB ด้วยภาษา Python
- เรียนรู้คำสั่งเบื้องต้นที่ใช้เพื่อโปรแกรมอุปกรณ์ Rigol Digital Function Generator (DG1000 Series)
- คำนวณค่าและสร้างอาร์เรย์ของข้อมูลสำหรับหนึ่งคาบสัญญาณ โดยใช้ภาษา Python แล้วอัพโหลดไปยังเครื่องกำเนิดสัญญาณ

รายการอุปกรณ์สำหรับการทดลอง

- เครื่องกำเนิดคลื่นสัญญาณ Rigol Digital Function Generator DG1022 พร้อมสายสัญญาณ 1 เส้น สำหรับสร้างสัญญาณทดสอบ
- เครื่องมือวัด Rigol Digital Oscilloscope พร้อมสาย Probe อย่างน้อย 1 เส้น สำหรับวัดคลื่นสัญญาณ
- สาย USB type-B สำหรับเชื่อมต่อระหว่างคอมพิวเตอร์และเครื่อง Rigol DG1022
- คอมพิวเตอร์สำหรับรันสคริปต์ Python และติดตั้งซอฟต์แวร์ที่จำเป็นในการใช้งานไว้แล้ว

ตัวอย่าง Python Script สำหรับทดสอบการเชื่อมต่อกับอุปกรณ์เครื่องมือวัด

```
#!/usr/bin/env python
################################################################################
# Author: Rawat S.
#   (Dept. of Electrical & Computer Engineering, KMUTNB, Bangkok/Thailand)
################################################################################

import visa
import time, sys
visa_driver = '' # use either 'visa64' or 'visa32' or '@py' or left empty.
resources  = visa.ResourceManager( visa_driver )
devices    = resources.list_resources()
usb_device = None

if len(devices) > 0:
    print ('Found #devices: %d' % len(devices) )
    for device in devices:
        print ('>>', device)
        if str(device).startswith('USB0'):
            usb_device = device
            print ('select:', str(device)) # select the first device found
            break

print (30*'-')

if usb_device != None:
    instr = resources.open_resource( usb_device ) # use the selected USB device
    instr.write( '*IDN?' )
    time.sleep(0.5)
    ret_str = instr.read()
    fields  = ret_str.split(',')
    dev_model = fields[2]
```

```
    print ( ', '.join(fields[0:3]) )
    print ( dev_model )
else:
    print ('No device found')
    sys.exit(-1)

print ('\nDone....')
```

โค้ดตัวอย่างแรกนี้ ใช้ทดสอบการทำงานของ Python Script เพื่อเชื่อมต่อด้วย PyVISA ผ่านพอร์ต USB ไปยังเครื่องกำเนิดสัญญาณ DG1022 ในกรณีที่มีอุปกรณ์เชื่อมต่อกับพอร์ตของ USB อยู่ในขณะนั้น จะมีการส่งคำสั่ง `*IDN?` เพื่ออ่านข้อมูลจากเครื่องที่ใช้ระบุว่า เป็นเครื่องรุ่นใด มีรหัส Serial Number เป็นอย่างไร

ตัวอย่างที่ 2: Python Script

โค้ดสาธิตการสร้างอาร์เรย์ข้อมูล เพื่อนำไปสร้างรูปคลื่นสัญญาณตามที่กำหนดไว้ เช่น รูป Sine Wave, Half-Sine Wave, Sawtooth Wave

```
#!/usr/bin/env python
################################################################################
# Author: Rawat S.
#   (Dept. of Electrical & Computer Engineering, KMUTNB, Bangkok/Thailand)
################################################################################

import visa
import time, sys, re
import numpy as np

vendor_id   = None
device_id   = None
instr_model = None
resources   = None
instr       = None
################################################################################
visa_driver = ''  #  use either 'visa64' or 'visa32' or '@py' or left empty.
resources = visa.ResourceManager( visa_driver )
devices = resources.list_resources()

if len(devices) > 0:
    print ('Found #devices: %d' % len(devices) )
    for device in devices:
        print ('>>', device)
        device = device.replace('::',',')
        fields = device.split(',')
```

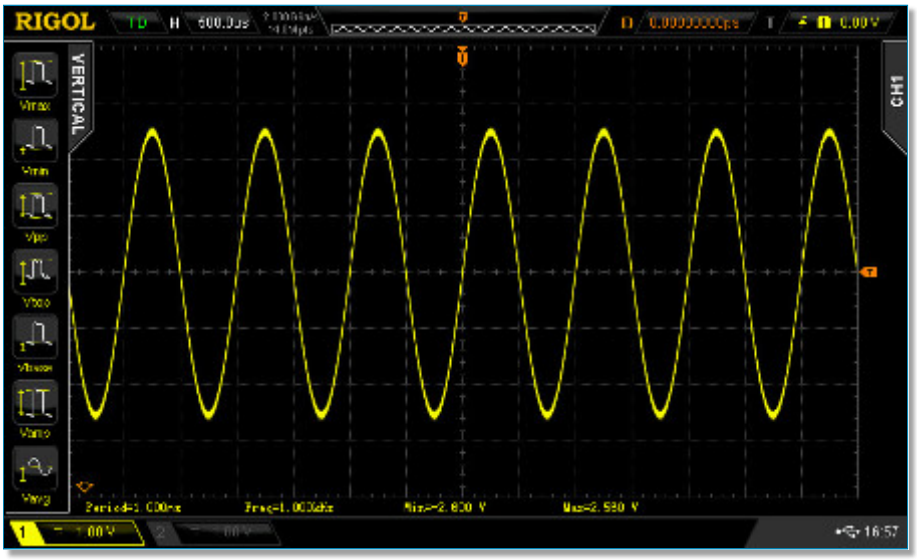
```
        if len(fields) >= 5 and fields[3].startswith('DG'):
            vendor_id = fields[1]
            device_id = fields[2]
            if not vendor_id.startswith('0x'):
                vendor_id = '0x{:04X}'.format( int(fields[1]) )
            if not device_id.startswith('0x'):
                device_id = '0x{:04X}'.format( int(fields[2]) )
            instr_model = fields[3]
            print (vendor_id, device_id, instr_model)

print (30*'-')
################################################################################
def listInstruments():
    global resources
    devices = resources.list_resources()
    print (devices)

def selectInstrument( vendor_id, device_id, instr_model ):
    cmd_str = "USB0::%s::%s::%s::INSTR" % (vendor_id,device_id,instr_model)
    instr = resources.open_resource( cmd_str, timeout=500, chunk_size=102400 )
    return instr

def cmdWrite(cmd, dly=0.1):
    global instr
    instr.write( cmd )
    time.sleep( dly )

def cmdRead(cmd, dly=0.1):
    global instr
    instr.write( cmd )
    time.sleep( dly )
    try:
        str = instr.read()
    except Exception:
        str = None
    return str

def showInstrumentInfo():
    print ( cmdRead("*IDN?",1.0) )

# select Rigol DG1022
if vendor_id == '0x1AB1' and device_id == '0x0588':
    instr = selectInstrument( vendor_id, device_id, instr_model)
else:
    print ('No DG1022 instrument found !!!')
    sys.exit(-1)

# show info about the instrument detected
showInstrumentInfo()
```

```
# connect to the remote instrument
cmdWrite( "SYSTem:REMote" )

##############################################################################
def gen_data_sin():
    N = 512                    # number of samples per period
    x = np.arange(-N/2,N/2)    # N points per period
    #y = [ int(8191 * np.sin( 2*np.pi*i/N) + 8192) for i in x ]
    y = 8191*np.sin(2*x*np.pi/N) + 8192
    y = y.astype(int)
    return ','.join(map(str,y))

def gen_data_halfwave_sin():
    N = 512                    # number of samples per period
    x = np.arange(-N/2,N/2)    # N points per period
    #y = [ int(8191 * np.abs(np.sin( np.pi*i/N)) + 8192) for i in x ]
    y = 8191*np.abs(np.sin(x*np.pi/N)) + 8192
    y = y.astype(int)
    return ','.join(map(str,y))

def gen_data_square( duty_cycle = 0.5 ):
    N = 1024               # number of samples per period
    x = np.arange(0,N)     # N points per period
    y = [ int(16383 * (i < duty_cycle*N)) for i in x ]
    return ','.join(map(str,y))

def gen_data_sawtooth( ):
    N = 2048               # number of samples per period
    x = np.arange(0,N)     # N points per period
    y = [ int( 16383.0*i/N ) for i in x ]
    return ','.join(map(str,y))

#data = gen_data_sin()
#data = gen_data_halfwave_sin()
#data = gen_data_square(0.25)
data = gen_data_sawtooth()

##############################################################################
# set some parameters for waveform generation
freq   = 1000    # in Hz
volt   = 5.0     # Volt peak-to-peak
offset = 0.0     # Volt offset

print ('turn off output CH1')
cmdWrite( "OUTP OFF" )

print ("select user-defined (arbitrary) waveform")
cmdWrite( "FUNC USER" )

print ( 'max. volatile memory depth: ' + cmdRead("DATA:ATTR:POIN? VOLATILE") )
```

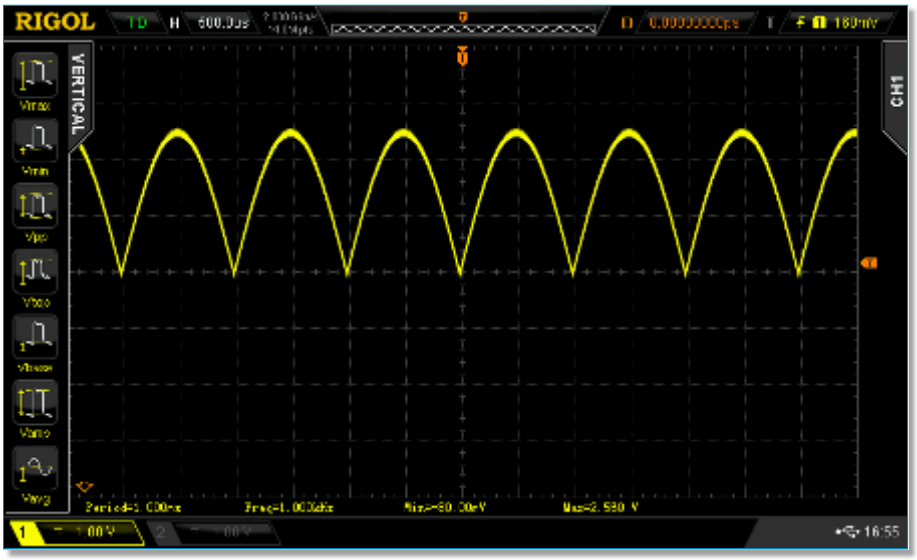
```
print ("set frequency: {:d} Hz".format(freq))
cmdWrite( "FREQ %d" % freq )

print ("select voltage output unit: voltage peak-to-peak")
cmdWrite( "VOLT:UNIT VPP" )

print ("set initial voltage value: {:.3f}".format( volt ))
cmdWrite( "VOLT %.1f" % volt )

print ("set initial voltage offset: {:.3f}".format( offset ))
cmdWrite( "VOLT:OFFS %.3f" % offset )

cmdWrite( "DATA:DELelete VOLATILE" )
time.sleep(1.0)

cmdWrite( "DATA:DAC VOLATILE,%s" % data ) # upload data to the volatile memory
time.sleep(1.5)

cmdWrite( "FUNC:USER VOLATILE" ) # apply the data in the volatile memory
time.sleep(1.0)

print ("turn on the output CH1")
cmdWrite("OUTP ON")
time.sleep(1.0)

if instr != None:
    instr.close()
    del instr
if resources != None:
    resources.close()
    del resources

print ('Done....')
sys.exit(0)
################################################################################
```

ตัวอย่างที่สองนี้ สาธิตการเขียนโค้ดและฟังก์ชันในภาษา Python เพื่อคำนวณค่าข้อมูลในอาร์เรย์ที่ใช้ในการสร้างรูปคลื่นสัญญาณแบบมีคาบ ลองศึกษาการทำงานของฟังก์ชันในโค้ดตัวอย่าง ดังนี้ `gen_data_sin()`, `gen_data_halfwave_sin()`, `gen_data_square()`, `gen_data_sawtooth()` และผลที่ได้เมื่อเรียกใช้ฟังก์ชันแต่ละฟังก์ชัน

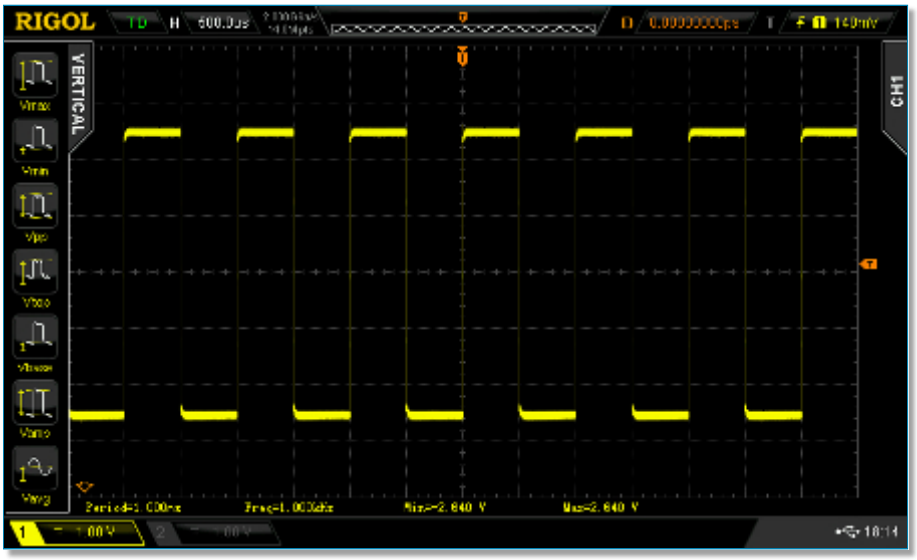
สำหรับคำสั่งที่เกี่ยวข้องกับ Rigol DG1022 สามารถศึกษาเพิ่มเติมได้จากเอกสารที่เป็นไฟล์ .PDF ของผู้ผลิต DG1000 Programming Guide (2014)



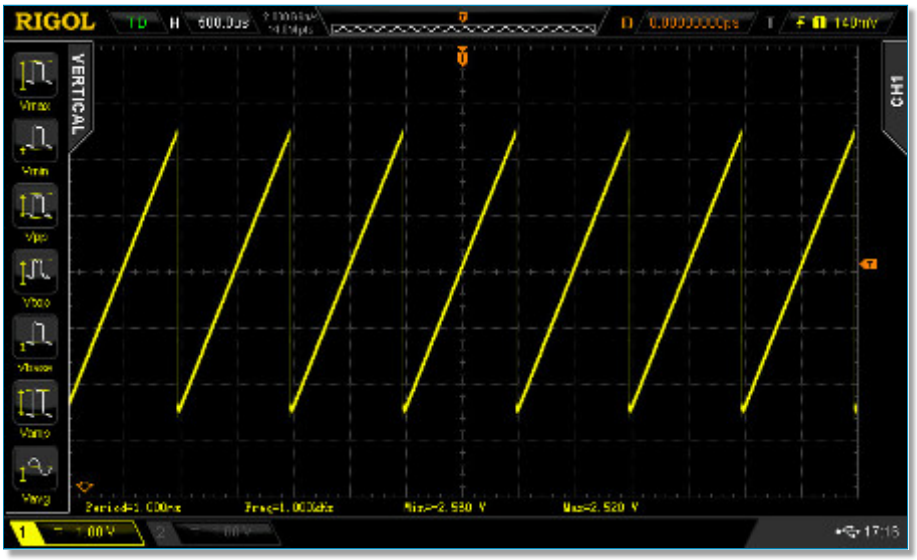
รูปสัญญาณเอาต์พุตที่เป็นคลื่นแบบ Sine Wave



รูปสัญญาณเอาต์พุตที่เป็นคลื่นแบบ Half-Sine Wave



รูปสัญญาณเอาต์พุตที่เป็นคลื่นแบบ Square Wave



รูปสัญญาณเอาต์พุตที่เป็นคลื่นแบบ Sawtooth Wave

สรุปผลการเรียนรู้ที่คาดหวัง

- ได้เรียนรู้การติดตั้งซอฟต์แวร์ที่จำเป็นสำหรับการเชื่อมต่อกับอุปกรณ์เครื่องมือวัดที่รองรับ VISA Programming API
- สามารถใช้ภาษา Python ในการติดต่อสื่อสารและเชื่อมต่อกับเครื่องกำเนิดสัญญาณ Rigol DG1022 ผ่านพอร์ต USB
- ได้เห็นตัวอย่างฟังก์ชันใน Python สำหรับสร้างอาร์เรย์ของข้อมูลที่นำไปใช้สร้างรูปคลื่นสัญญาณไฟฟ้าแบบมีคาบ โดยใช้เครื่องกำเนิดสัญญาณ

แนวทางการเรียนรู้เพิ่มเติม

- **ทดลองสร้างรูปคลื่นสัญญาณแบบมีคาบอื่น ๆ นอกเหนือจากที่ให้ไว้เป็นตัวอย่าง**
- **สร้างฟังก์ชันด้วย** Python **ที่ใช้ในการสร้างอาร์เรย์ข้อมูลของคลื่นสัญญาณ สามารถกำหนดพารามิเตอร์ต่าง ๆ ได้ เช่น ช่วงแรงดันสูงสุดต่ำสุด ความถี่ จำนวนข้อมูลหรือขนาดอาร์เรย์ เป็นต้น**
- **สร้างรูปกราฟของข้อมูลในอาร์เรย์ โดยใช้** Python Matplotlib **เปรียบเทียบกับรูปสัญญาณที่วัดได้ด้วยออสซิลโลสโคป**