Факултет инжењерских наука Универзитета у Крагујевцу

Назив студијског програма: Машинско инжењерство

Ниво студија: Основне академске студије

Модул: Примењена механика и аутоматско управљање

Предмет: Сензори и актуатори

Број индекса: 141/2012

Андрија Д. Станојевић

Коришћење Python програмског језика у апликацијама мерења и управљања

завршни рад

Комисија за преглед и одбрану:

1. Проф. др Милан Матијевић - ментор
2. Проф. др Драган Тарановић
3. Проф. др Слободан Савић

Датум одбране: _____

Оцена:_____

Задатак рада:

Описати примену NI USB 6008 AD/DA конвертора коришћењем Python програмског језика у апликацијама мерења и управљања. Као илустративни пример користити електрично коло које симулира систем првог реда.

Препоручена литература:

[1] Портал предмета Сензори и актуатори

[2] М. Тодоровски, *Рачунарска подршка у извођењу експеримената и анализи резултата мерења, Дипломски рад,* Машински факултет у Крагујевцу, Крагујевац, 2010

[3] А. Станојевић, А. Вељовић, *Основи коришћења NI 6008 USB AD/DA картице путем програмског језика Python и програмског пакета LabVIEW, Извештај са студентске праксе,* Факлултет инжењерских наука у Крагујевцу, Крагујевац, 2015

Крагујевац, 2015

ментор:

Проф. Др Милан Матијевић

_____

Резиме

Рад објашњава могућност употребе NI6008 USB AD/DA конвертора путем програмског језика Python, кроз експеримент са RC електричним колом. У експерименту се врши снимање одзива на различите облике улазног напона. Дати су опис и карактеристике коришћеног конвертора, поступак припреме за експеримент, објашњени су програми написани у Python програмском језику, и на крају приказани добијени резултати.

Кључне речи: Python програмски језик, NI 6008, RC коло, мерење

Abstract

The paper explains the possibility of using the NI6008 USB AD/DA converter using the Python programming language, through RC electric circuit experiment. A response on different forms of input voltage is recorded during the experiment. The description and characteristics of the used converter are given, as well as the preparation procedure for the experiment and the programs written in the Python programming language. Finally, obtained results are presented.

Keywords: Python programming language, NI 6008, RC circuit, measurement

Садржај

1. Увод 1
2. NI USB 6008 АД/ДА конвертор 2
   2.1 Аналогно-дигитална (А/Д) конверзија сигнала 2
   2.2 NI USB 6008 АД/ДА картица 3
      2.2.1 Основне карактеристике 3
3. Поставка експеримента 6
   3.1 Дефинисање експеримента 6
   3.2 Поступак припреме за извођење експеримента 7
      3.2.1 Правилан одабир параметара отпорника и кондензатора. 7
      3.2.2 Формирање електричног кола на протоборд плочи и повезивање са NI USB 6008 7
   3.3 Потребни софтвери за коришћење NI USB 6008 картице 9
4. Python програмско решење 9
   4.1 Основни програм 9
   4.2 Програми за покретање екперимената 12
      4.2.1 Креирање програма за снимање одзива на одскочни улазни напон 16
      4.2.2 Креирање програма за снимање одзива са синусним улазним напоном 19
5. Закључак 22
6. Прилози 23
   6.1 Прилог 1. Основи програм 23
   6.2 Прилог 2. Код за снимање одзива на одкочни улазни напон 26
   6.3 Прилог 3. Код за снимање одзива на синусни улазни напон 27

Литература: 29

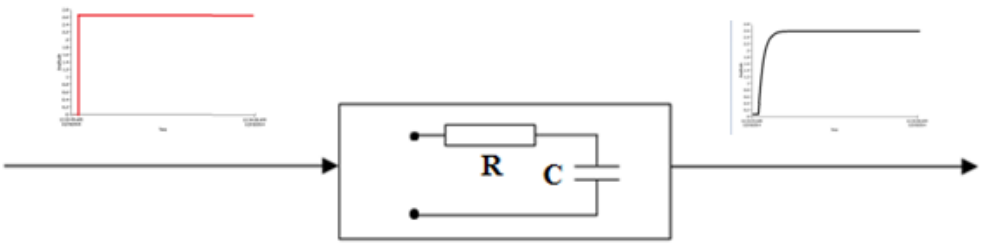
1. Увод

Циљ рада се односи на овладавање техником коришћења NI USB 6008 AD/DA картице, путем програмских решења креираних у Python програмском језику. Добијена решења, уз одговарајуће услове и предзнање, омогућују употребу поменуте картице без софтвера предвиђеног од стране произвођача, за који је потребно платити лиценцу.

Процес који се разматра је једноставно електрично коло сачињено од једног отпорника и кондензатора. Улаз у процес биће напон са излаза NI картице, чији се тип и вредност задају софтверски, а излаз из процеса биће напон на кондензатору.

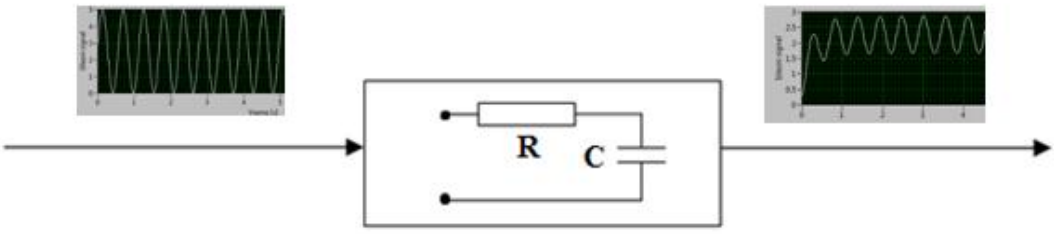
Спроводе се два експеримента у којима се NI USB 6008 користи за постављање улазног сигнала у процес, и мерење (прикупљање, чување и презентација) излазног сигнала из процеса:

1) Улазни напон RC кола је одскочни (степ) сигнал, а излаз је напон на кондензатору.



Слика 1.1. *Одскочни улаз и одзив RC кола*

2) Улазни напон RC кола је облика $Asin(\omega t)$, а излаз је напон на кондензатору.

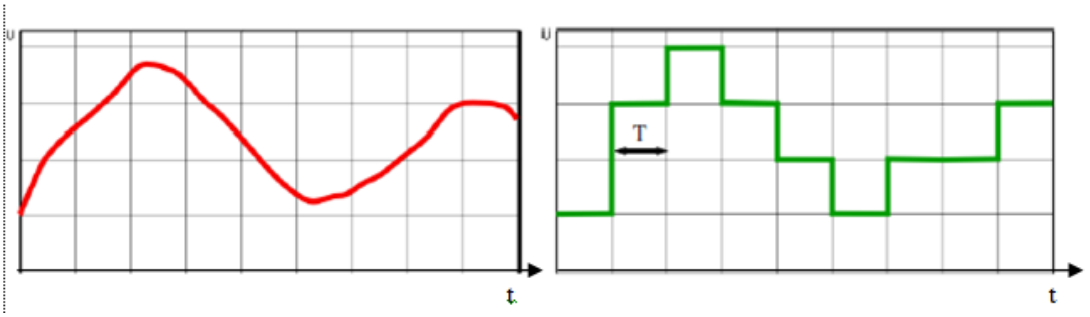


Слика 1.2. *Хармонијески улаз и одзив RC кола*

2. NI USB 6008 АД/ДА конвертор

2.1 Аналогно-дигитална (А/Д) конверзија сигнала

Аналогни сигнал је континуални сигнал. Пример произвољног аналогног сигнала може се видети на слици 2.1.1.а [3]. Струјни и напонски сигнали су један од примера аналогних електричних сигнала, који су и најчешће излазне величине са мерних претварача. Такав сигнал није погодан за обраду у дигиталним рачунарима, па се из тог разлога врши дискретизација сигнала по времену и нивоу и на тај начин се добија дигитални сигнал (слика 2.1.1.б)) [3].



а) *б)*

Слика 2.1.1. *а) Аналогни сигнал ; б) Дигитални сигнал*

Дискретизација по времену, односно, узимање вредности аналогног сигнала, врши се на сваких *T* секунди, где је *T* периода одабирања. Истовремено се врши дискретизација по нивоу. Уколико се стварна вредност аналогног сигнала у тренутку одабирања налази у опсегу између два нивоа, као меродавна биће прихваћена вредност нивоа која се мање разликује од стварне. Једна од могућности је да се вредност задржава до следећег тренутка одабирања и процес се затим понавља, као на слици 1. Што је већи број нивоа и мања периода одабирања *T*, дигитални сигнал биће веродостојнији аналогном.

Избор вредности периоде одабирања *T* зависи од природе процеса над којим се врши мерење. Уколико су промене брзе, потребно је мање *T*.

А/Д конверзија се врши у подсистему за мерење величина од интереса. Уколико на основу резултата мерења, рачунар генерише управљање, или једноставно треба да се генерише управљање у отвореној повратној спрези, потребно је Д/А конверзијом дигитални сигнал генерисан у рачунару конвертовати у аналогни.

АД/ДА картица NI USB 6008, је картица која се повезује са рачунаром УСБ комуникацијским портом, и омогућава процесе АД и ДА конверзије.

2.2 NI USB 6008 АД/ДА картица

NI USB 6008 АД/ДА конвертор је производ корпорације “National instruments”. Изглед картице је дат на слици 2.2.1.

Слика 2.2.1. *NI USB 6008 АД/ДА картица.*

2.2.1 Основне карактеристике

За правилну употребу картице потребно је знати њене основне карактеристике. Неке од њих дате су у табели 2.2.1.1 [4].

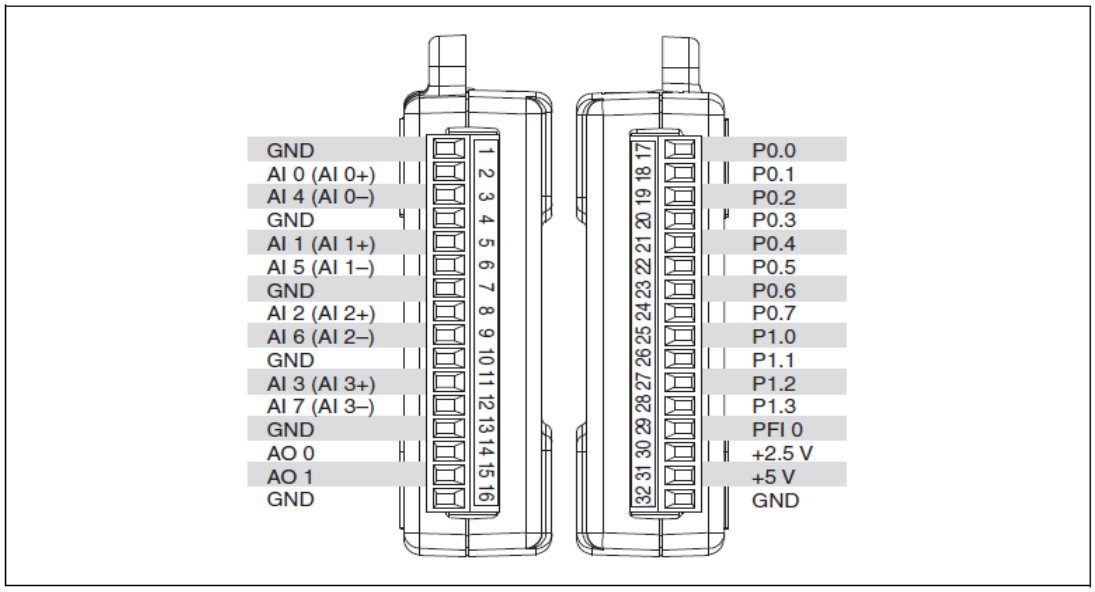
Табела 2.2.1.1 *Карактеристике NI USB 6008*

Karakteristike NI USB 6008 АД/ДА картице	
Број аналогних излаза (AO)	2
Број аналогних улаза (AI)	8 SE / 4 DI ***
Број дигиталних улаза/излаза	12
Улазна резолуција	11 bit SE / 12bit DI ***
Максимални број одбирака по секундки	10 kS/s
Излазна резолуција	12 bit
Фреквенција аналогних излаза	150 Hz
Опсег напона на излазу	0 - 5 V
Опсег напона на улазу	± 1 до ± 20 V
Напајање картице и комункација са рачунаром	Преко УСБ порта
***SE – single ended (мерење једним улазом) ; DI – differential input (мерење са два улаза)	

У наставку следи објашњење појединих карактеристика:

- *Број аналогних улаза (AI)* зависи од начина мерења. Уколико се мерење врши једним крајем (SE – single ended) у односу на заједничку масу која је на нултом потенцијалу (GND), број улазних канала је 8. Други тип мерења је диференцијални (DI), односно, мери се разлика потенцијала између два улаза, како је 8/2 = 4 у том случају је број улаза 4. Диференцијални мод (DI) важи само за тачно одређене парове улаза.
- *Улазна резолуција* дефинише број нивоа којима се може представити аналогни сигнал. Како је резолуција у овом случају 12 бита за DI мод, број нивоа који картица тада може дати је 2^{12}, односно, 4096.
- *Максимални број одбирака по секунди, или "Sampling Rate"* односи се на дискретизацију аналогног сигнала по времену. Како је број одбирака 10 kS/s, односно 10000 одбирака у секунди, теоретски, минимална периода одабирања биће 1/10000 = 0,0001 s = 0,1 ms.
- *Фреквенција аналогних излаза* представља максимални број одбирака (семплова) по секунди који могу да се генеришу на аналогним излазима. У овом случају износи 150 Hz или 150 S/s. На ову карактеристику је потребно обратити пажњу уколико се на излазу генерише периодични сигнал (нпр. синусни сигнал), где се мора правилно одабрати учестаност сигнала (може бити највише 75 Hz) како би исти задржао жељену форму.
- *Опсег напона на улазу* зависи од начина мерења. За *single ended* мод (SE, или RSE), опсег износи ± 10 V, за *differential input* мод (DI, или RDI) износи ± 20 V. Минимални опсег који се може остварити је ± 1 V.

Потребно је знати распоред улаза/излаза (I/O) картице за успешно повезивање са процесом. На слици 2.2.1.1 [4] дат је приказ свих аналогних (лево) и дигиталних (десно) улаза и излаза.



Слика 2.2.1.1. *Приказ аналогних и дигиталих улаза и излаза картице*

Тумачење ознака са слике 2.2.1.1:

- **GND** – *ground.* „Земља“, односно заједничка маса. Нулти потенцијал.
- **AO_**број излаза – *Analog output*. Аналогни излази, има их 2 (0 и 1).
- **AI_**(од 0 до 7) – *Analog input.* Аналогни улази за **SE** мод, има их 8.
- **(AI_број +) ; (AI_број -) -**Парови аналогних улаза за **DI** мод. Постоји 4 пара.
- **+5 V –** Излаз за напајање који даје напон од константних 5 V (или мање уколико је напон са УСБ порта мањи), и струју јачине до 200 mA.
- **+2,5 V –** Излаз за напајање напона 2,5 V који је погодан за тестирање картице.
- **P0.**(од 0 до 7) – *Port 0.* Дигитални улази и излази. Да ли ће се канал користити као улаз или излаз дефинише се софтверски.
- **P1.**(од 0 до 3) – *Port 1.* Дигитални улази и излази. Да ли ће се канал користити као улаз или излаз - дефинише се софтверски.

Препоручен софтвер за коришћење картице од стране произвођача је:

- *LabVIEW - Laboratory Virtual Instrument Engineering Workbench*, софтверски пакет који је производ исте компаније.

Неопходан софтвер за коришћење картице је

- *NI-DAQmx* – Драјвер за картицу, неопходан и за *LabVIEW,* али и за примену картице са другим софтверским алатима.

3. Поставка експеримента

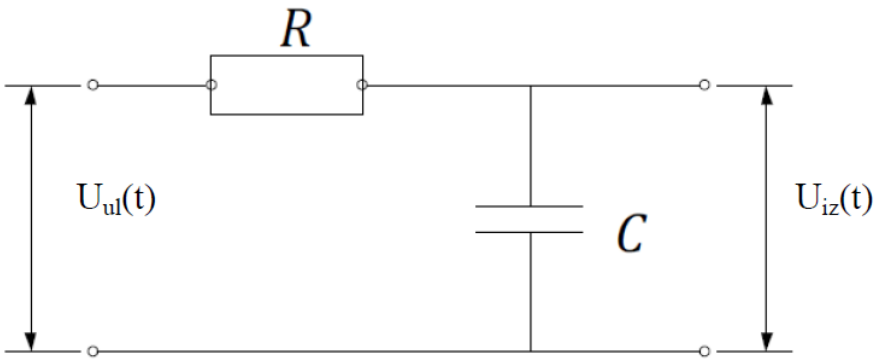
3.1 Дефинисање експеримента

Циљ експеримента је снимање одзива редног RC кола на одкочну (степ) и синусну побуду путем АД/ДА картице NI USB 6008, коришћењем програмских решења креираних у Python програмском језику.

Потребно је прво дефинисати опрему и електронске компоненте потребне за спровођење експеримента:

- Отпорник, измерене отпорности R = 38 кΩ
- Кондензатор капацитивности C = 10 μF
- Лабораторијска протоборд плоча, као основа за формирање електричног кола
- Проводници
- Мултиметар
- NI USB 6008 АД/ДА картица
- Рачунар

Електрична шема кола је дата на слици 3.1.1 [5].



Слика 3.1.1. *Електрична шема RC кола*

Улазни сигнал (напон $U_{ul}(t)$) у RC коло доводи се са аналогних излаза NI картице, и представља улаз у процес. Да ли ће $U_{ul}(t)$ бити одскочни или синусни сигнал, дефинише се софтверски. Као излаз из процеса, прати се напон на кондензатору $U_{iz}(t)$, који се са NI картицом повезује путем једног од аналогних излаза, односно, мерење се врши у *single ended* моду.

3.2 Поступак припреме за извођење експеримента

3.2.1 Правилан одабир параметара отпорника и кондензатора.

Како је излаз процеса напон на кондензатору, пожељно је да прелазни процес, односно, пуњење кондензатора у овом случају, траје адекватно дуго (да се може пратити у реалном времену, а да експеримент не траје предуго).
Трајање прелазног процеса, при одскочном напонском улазу дефинисано је временском константом RC кола:

$$\tau = RC \qquad 3.2.1.1$$

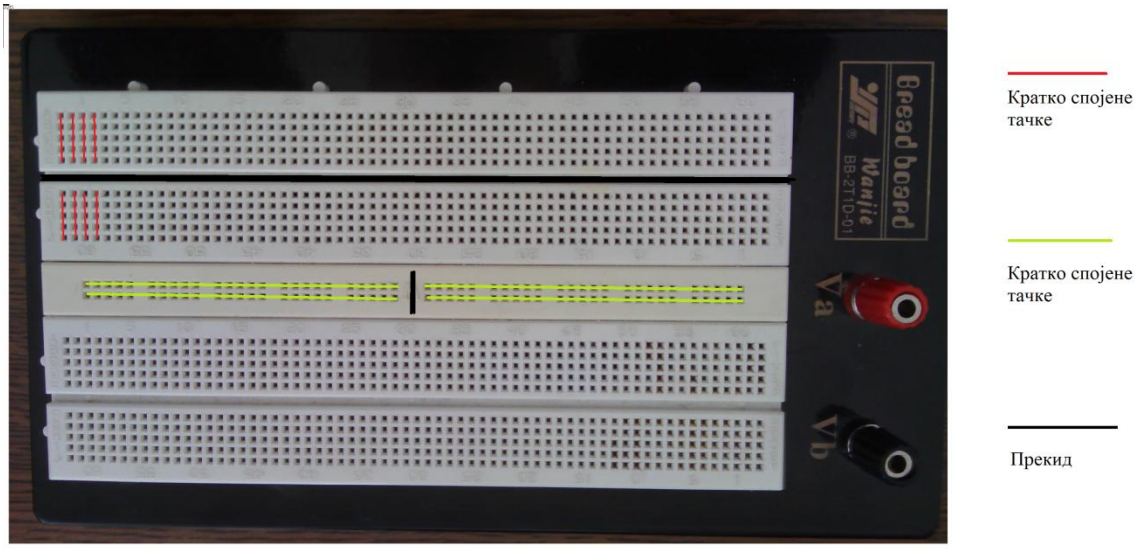
Процес траје приближно 5τ [1]. Према изабраним вредностима отпорности R и капацитивности C, прелазни процес ће приближно износити:

$$5RC = 5 \times 38k\Omega \times 10\mu F = 1900\ ms = 1.9s \qquad 3.2.1.2$$

што одговара потребама експеримента.

3.2.2 Формирање електричног кола на протоборд плочи и повезивање са NI USB 6008

Најпре, на слици 3.2.2.1 се може видети изглед протоборд плоче коришћене у експерименту, након чега ће уследити опис исте.



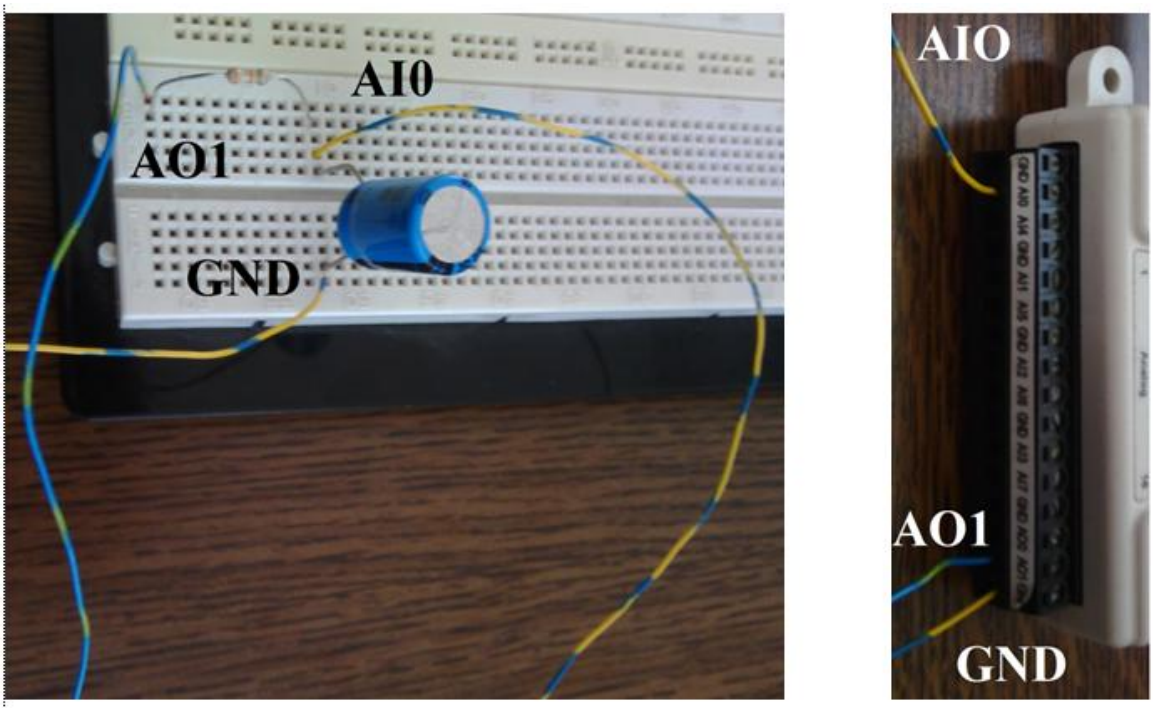
Слика 3.2.2.1. *Протоборд плоча*

- Црвеном линијом, илустрован је начин на које су спојене убодне тачке (пинови) вертикалних водова на плочи. Приказ је дат за прва четири вода, сви преостали вертикални водови повезани су на исти начин. Користе се за

формирање електричног кола прикључивањем компоненти и проводника на одговарајуће место.

- Зеленом линијом приказан је начин на који су спојени хоризонтални водови плоче. Практично се најчешће користе за довођење + и – краја са извора напајања.
- Црна линија представља прекид.

Како су параметри и шема везивања компоненти познати, следи реализација кола на протоборд плочи и повезивање са картицом (слика 3.2.2.2).



а) *б)*

Слика 3.2.2.2: *а) Формирање кола на протоборд плочи; б) Повезивање са картицом*

Поред проводника на слици 3.2.2.2, стоје ознаке канала NI картице на које се проводници воде:

- **A01 и GND –** *Analog output 1 (Аналогни излаз број 1 - картице) и Ground (Уземљење или заједничка маса).* Са ових канала се доводи аналогни напонски улаз у процес.
- **AI0 –** *Analog input 0 (Аналогни улаз број 0 - картице).* Улазни канал картице се користи за мерење излаза процеса, односно, напона на кондензатору. Мерење се врши у *single ended* режиму што је уочљиво са слике, јер се практично само један проводник доводи на улаз картице.

3.3 Потребни софтвери за коришћење NI USB 6008 картице

Пре почетка креирања било којих програма за употребу NI картице, потребно је на рачунару имати инсталиран драјвер за исту. Уколико нема инсталационог CD-а, драјвер под називом **NI-DAQmx** може се наћи на званичном сајту „National instruments“ корпорације.

За потребе рада, конкретно је коришћен **NI-DAQmx 9.9**, који се може бесплатно преузети са следеће адресе:

- http://www.ni.com/download/ni-daqmx-9.9/4707/en/

Постоји доста новијих верзија драјвера, али је верзија 9.9 коришћења због програмског решења у Python-у, које проверено ради са поменутом верзијом драјвера.

Оно што је битно јесте да драјвер долази са интерфејсом за програмирање апликација (API), што значи да се са њиме добија динамичка библиотека (nicaiu.dll) која садржи решене функције за комуникацију са NI USB 6008. Ова библиотека је предвиђена да се користи са програмским језиком C [8]. Због лакше синтаксе и ширих могућности, не користи се C, већ Python 2.7.

Програмски језик Python је бесплатан и инсталација истог и потребних библиотека се може скинути са следеће адресе:

- http://sicp-s3.mit.edu/tutor/6.01/reference/software/install

Након инсталације наведених софтвера, могуће је приступити креирању програмских решења, која су описана у даљем тексту.

4. Python програмско решење

4.1 Основни програм

Комуникација између NI6008 и рачунара је сложена. Конвертор захтева од рачунара серију инструкција везану за његово подешавање за одређени задатак (врста мерења, брзина мерења, који канали су активни, величина бафера ако се користи итд.) а рачунар треба да интерпретира добијене сигнале са USB порта. Такође, потребно је да конвертор комуницира само са једном апликацијом у датом тренутку. Ово све решава поменути драјвер NI-DAQmx 9. Опис основног програма који следи је урађен по узору на вежбе са портала предмета Сензори и Актуатори [8].

Да би се користила библиотека која долази уз NI-DAQmx9.9, предвиђена за C, у Python-у, користи се модул Python-a ctypes. Овај модул импортује се командом :

```
import ctypes
```

Потребне су дефиниције константи које се налазе у NIDAQmx.h хедер фајлу који се налази у …место инсталације драјвера \National Instruments\NI-DAQ\DAQmx ANSI C Dev\Include. Постоји више начина да се преузму дефиниције константи из овог фајла. Реформатирањем оваг фајла добијен је Python фајл NIDAQmxConstans.py. За потребе аквиционе апликације, прво се импортују све константе:

from NIDAQmxConstants import *

Када се учита ctypes, учитава се dll фајл:

```
nidaq = ctypes.windll.nicaiu
```

Пошто се користе типови података из програмског језика C, уводе се једноставније ознаке за функције које их обезбеђују:

```
int32 = ctypes.c_long

uInt32 = ctypes.c_ulong

uInt64 = ctypes.c_ulonglong

float64 = ctypes.c_double
```

Најједноставније објашњење формулисања аквизиције помоћу DAQmx драјвера је следеће. Прво је потребно дефинисати задатак (task) коме се придружују улазни или излазни канали. Сваки задатак може имати само један тип канала (улазни или излазни) и може имати већи број канала исте врсте.

Функција за формирање задатка је:

```
nidaq.DAQmxCreateTask("",ctypes.byref(taskHandle))
```

где је први параметар назив задатка, а други параметар је целобројна идентификација задатка која се прослеђује по референци.

Када је формиран задатак, излазни канал се може поставити преко:

```
nidaq.DAQmxCreateAOVoltageChan(taskHandle,chanal,"",float64(0.0),float64(5.0),DAQmx_Val_Volts, "")
```

Овде је први параметар идентификација задатка коме се канал придружује, други параметар је број физичког канала и уређаја (на пример „Dev0/ao1“). Трећи параметар је назив канала. Четврти и пети параметар су доња и горња граница излаза у V. Шести параметар је јединица мерења и последњи параметар је назив јединице (ако се не користе V).

Када је задатак оформљен и канал му је додат, потребно је покренути задатак помоћу команде:

```
nidaq.DAQmxStartTask(taskHandle)
```

Када је задатак покренут, аналогни излаз се шаље помоћу команде:

```
nidaq.DAQmxWriteAnalogScalarF64(taskHandle, 1, float64(10.0), value64, None));
```

Овде је први параметар идентификација задатка, други је логичка вредност која одређује да ли се задатак сам покреће са овом командом, трећа је временско ограничење за извршавање (ако време истекне, а команда се не изврши, команда избацује грешку). Четврти параметар је вредност која се шаље и последњи параметар је празан, односно резервисан је за будуће верзије драјвера и сада нема никакву намену.

Када је све што је планирано за овај задатак урађено, задатак се зауставља командом:

```
nidaq.DAQmxStopTask(taskHandle)
```

И брише из меморије командом:

```
nidaq.DAQmxClearTask(taskHandle)
```

Што се тиче команди везаних за аналогни улаз, задатак се поставља на идентичан начин као за аналогни излаз. Канал аналогног улаза се додаје задатку командом:

```
nidaq.DAQmxCreateAIVoltageChan(taskHandle, chanal, "",DAQmx_Val_RSE, lo, hi,
DAQmx_Val_Volts, None)
```

Први, други и трећи параметар су идентификација задатка, ознака физичког канала и назив канала. Четврти параметар је вредност која означава да ли се врши једнострано или диференцијално мерење. Пети и шести параметар су доња и горња граница мерења. Седми параметар је јединица мерења. Последњи параметар је празан.

Ако се врши мерење које користи бафер, потребно је подесити клок за одабирање командом:

```
nidaq.DAQmxCfgSampClkTiming(taskHandle, "",float64(10000.0), DAQmx_Val_Rising,
DAQmx_Val_FiniteSamps, uInt64(max_num_samples))
```

Параметри овде су : идентификација задатка, избор терминала за клок одабирања (ако се користи екстерни), број одбирака у секунди, ивица клока која се посматра, врста мерења (коначни или континуални), број одбирака који се мери по каналу.

Исто као за излазе, прво се покреће задатак, а затим се може очитати вредност командом:

```
nidaq.DAQmxReadAnalogScalarF64(taskHandle, ctypes.c_double(-1), ctypes.byref(data), None)
```

Овде су параметри: идентификација задатка, временско ограничење за извршење, променљива у коју се мерење бележи, празан параметар. После завршетка, потребно је исто зауставити и избрисати задатак.

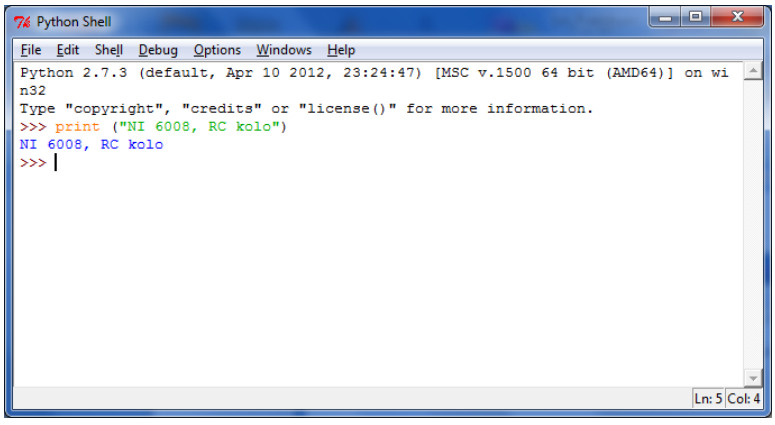
У скрипти која се користи за експеримент, формиране су функције SetAO() и ReadAIChanal() које обједињују ове команде. Скрипта у целости дата је у прологу 1 (поглавље 6.1).

4.2 Програми за покретање екперимената

За потребе експеримента, креирана су два *Python* фајла базирана на постојећој скрипти, описаној у претходном поглављу, сачуваној под именом Ni6008DAQ.

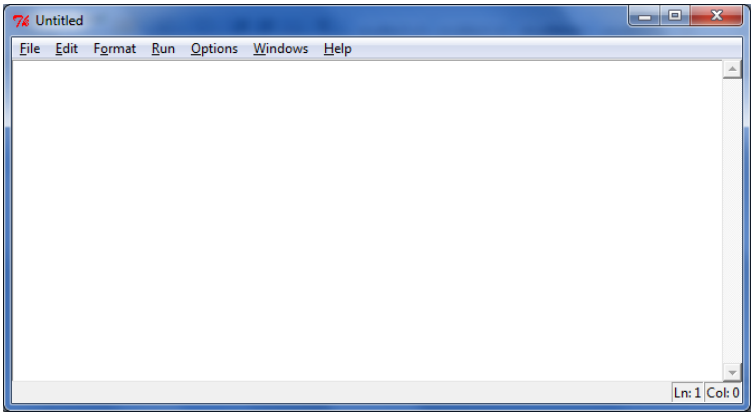
- Ради функционисања написаних кодова поменута скрипта и креирани фајлови морају бити у истом фолдеру.
- Код се пише у Python IDLE**.**
- Уколико пречица није на екрану, покретање се врши на следећи начин: Start/All Programs/python 27/IDLE(Python GUI)

После покретања отвориће се интерактивни интерпретер, python shell. Команде унете у овом прозору извршавају се одмах након уношења, без накнадног покретања програма (Слика 4.2.1).



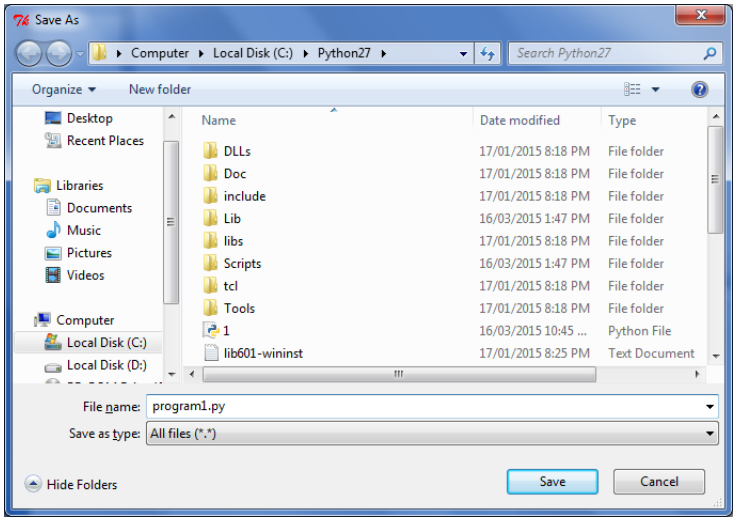
Слика 4.2.1. *Изглед првог прозора после покретања IDLE*

Да би се написао програм у целини, уз могућност едитовања, потребно је покренути едитор. Из File менија изабрати New Window. Отвориће се прозор као на слици 4.2.2.



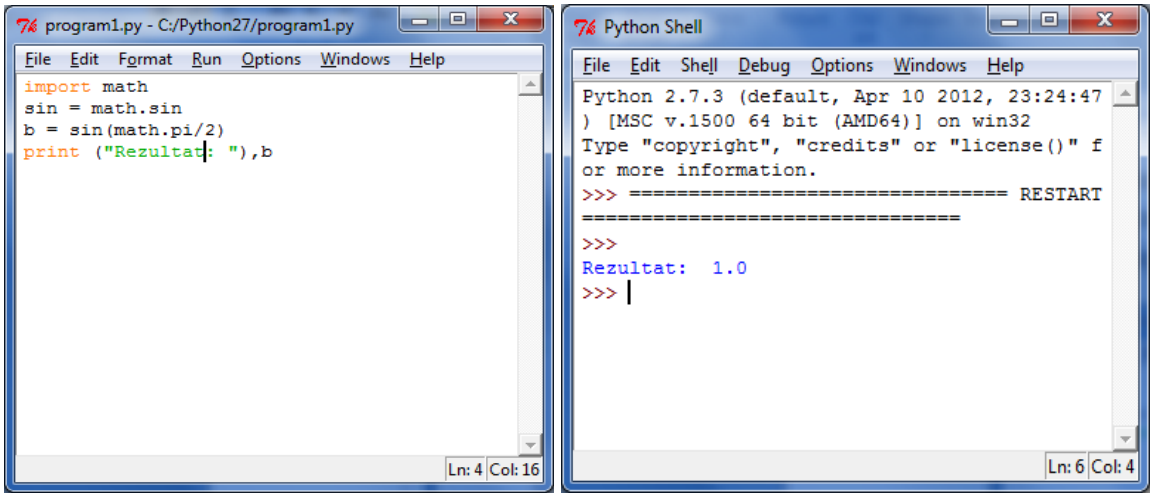
Слика 4.2.2. *Изглед едитора*

Из File менија изабрати Save As, одабрати локацију чувања фајла и на жељено име додати екстензију .py, нпр. program1.py. (слика 4.2.3)



Слика 4.2.3. *Чување фајла*

Након што је фајл сачуван, отвора се прозор едитора са именом фајла у горњем левом углу. Тада се може започети са писањем кода. На слици 4.2.4.а) дат је пример кода за штампање синуса угла π/2. Покретање програма врши се притиском на тастер F5 или кликом на Run па затим Run Module. Пре извршења програма изаћи ће дијалог који каже да се програм мора сачувати, кликнути на ОК. После снимања фајла отвориће се python shell, где ће бити приказани резултати (Слика 4.2.4.б)).



а) б)

Слика 4.2.4. *а) Едитор за куцање кода, б) python shell*

После уводног, уопштеног дела, могуће је приступити прављењу програмског решења за коришћење NI USB 6008 картице за управљање и аквизицију података, сходно задатку експерименталног рада.

Први корак представља увожење свих потребних библотека, командом import.

```
import Ni6008DAQ as daq
```

омогућава комуникацију са картицом. (*as daq* значи да се библиотека увози под тим називом).

```
import time
```

омогућава приступ системском времену.

```
import matplotlib
from matplotlib import pyplot as plt
```

омогућава цртање графика.

```
import math
```

Бибиотека која омогућава израчунавање сложенијих математичких операција.

У наставку кода:

```
T = 0.02
```

Дефинисање периоде одабирања у секундама. Вредност периоде одабирања се може мењати сходно потребама, али због перформанси програма није препоручљиво узимати вредности мање од 0.02 секунде.

```
x = [ ]
```

Празан низ, где ће бити уписане вредности променљиве x.

```
y = [ ]
```

Празан низ, где ће бити уписане вредности променљиве y.

```
daq.SetAO(1,1,0)
```

у формату daq.SetAO(број уређаја, број канала, вредност напона у волтима), поставља вредност од 0 V**,** на AO1 излазни канал NI USB 6008 картице. Напајање се гаси, да би се кондензатор, чији излаз се снима, испразнио.

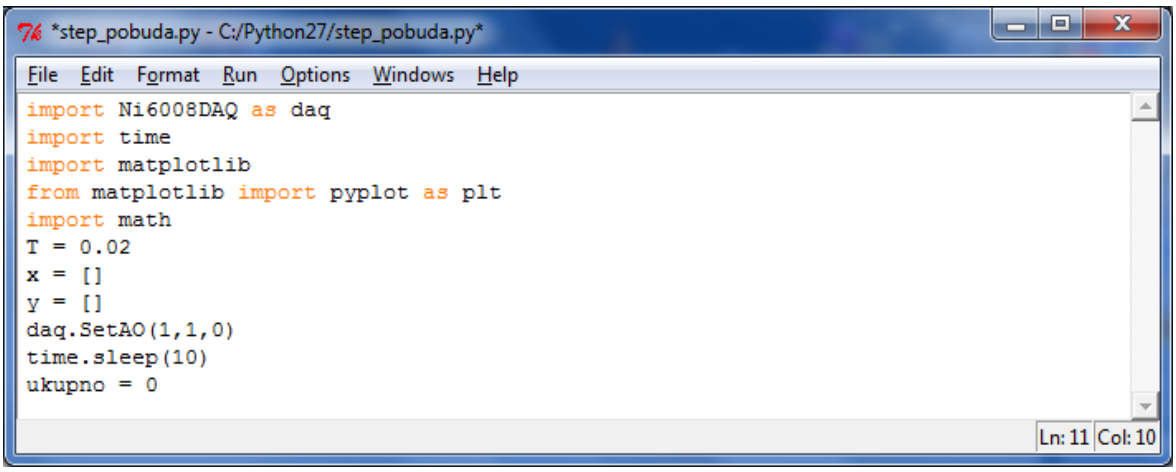
```
time.sleep(10)
```

У формату time.sleep (време у секундама) – чека 10 секунди да би се кондензатор испразнио.

```
ukupno = 0
```

Почетна вредност променљиве ***ukupno.***

Претходно описан део кода је заједнички за снимање одзива на степ и синусни улаз. (Слика 4.2.5).



Слика 4.2.5. *Први, заједнички део кода*

4.2.1 Креирање програма за снимање одзива на одскочни улазни напон

После претходног, заједничког дела кода, излазу АО додељује се вредност напона од 5V, на следећи начин:

```
daq.SetAO(broj uredjaja, broj izlaznog kanala, vrednost napona u  V)
```

Променљивој y се додељује вредност са улазног канала AI0, променљивој x се додели 0, и дефинише се променљива „vreme“ која представља системско време пре уласка у while петљу.

```
y.append((daq.ReadAIChanel(broj uredjaja, broj ulaznog kanala)))

x.append(0)

vreme = time.clock()
```

Да би се очитале вредности са излаза у тачно одређеним временским тренуцима (са периодом одабирања *T*), и дефинисало време трајања екеперимента потребно је формирати while петљу на следећи начин:

```
while ukupno < 5:
        while T+ukupno > time.clock() - vreme:
            pass
        ukupno += T
        y.append((daq.ReadAIChanal(1,0)))
        x.append(ukupno)
```

Вредност 5 је време трајања експеримента и може се мењати по потреби. Унутрашња петља пореди време које је протекло од почетка екперимента (***time.clock() – vreme)*** са временом када треба да се деси одабирање (***T+ukupno).***

- Чим ***time.clock() – vreme*** достигне вредност од ***T+ukupno*** врши се снимање сигнала са излаза (одабирање) у променљиву ***y*** функцијом ***y.append((daq.ReadAIChanal(1,0))).***
- Променљивој ***x*** (која ће бити штампана на временској оси) додељује се тренутна вредност променљиве ***ukupno.***
- Променљива ***ukupno*** увећева се за вредност периоде одабирања ***T.***
- Процес се понавља док је ***ukupno < 5.***

Вредност напона на излазу ***AO1*** из картице поставља се на нулу.

```
daq.SetAO(1,1,0)
```

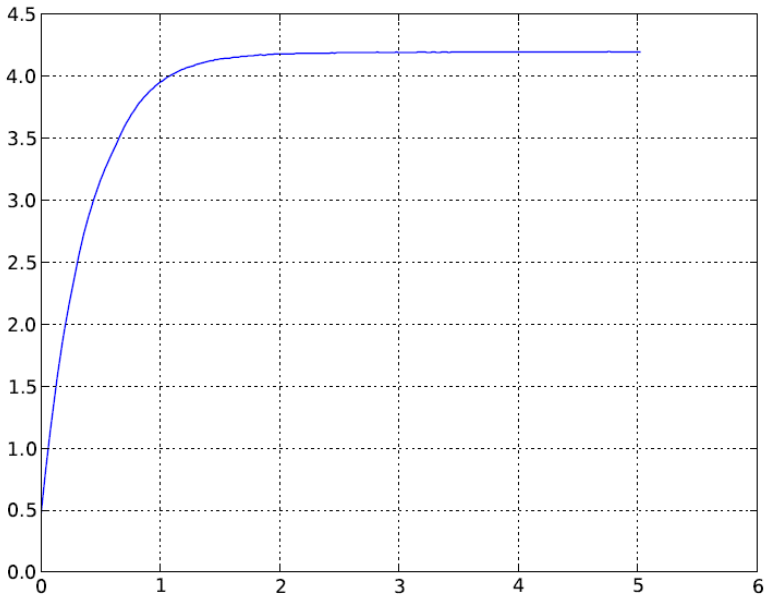
Штампање дијаграма:

```
plt.plot(x,y)
plt.draw()
plt.grid()
plt.show()
```

- Приликом чувања фајла, на име се додаје екстензија .py (на пример: step.py). Фајл се чува у фолдеру где се налази и скрипта Ni6008DAQ.
- Отварање фајла се врши на следећи начин: Десни клик на иконицу→Edit with IDLE.
- После отварања, покренути програм тастером F5 или кликом на Run→Run module.

Целокупни код дат је у прилогу 2 (поглавље 6.2).

После извршења програма аутоматски се добија дијаграм одзива система на одскочни напон на улазу (Слика 4.2.1.1).



Слика 4.2.1.1. Одзив на одскочни улазни напон.

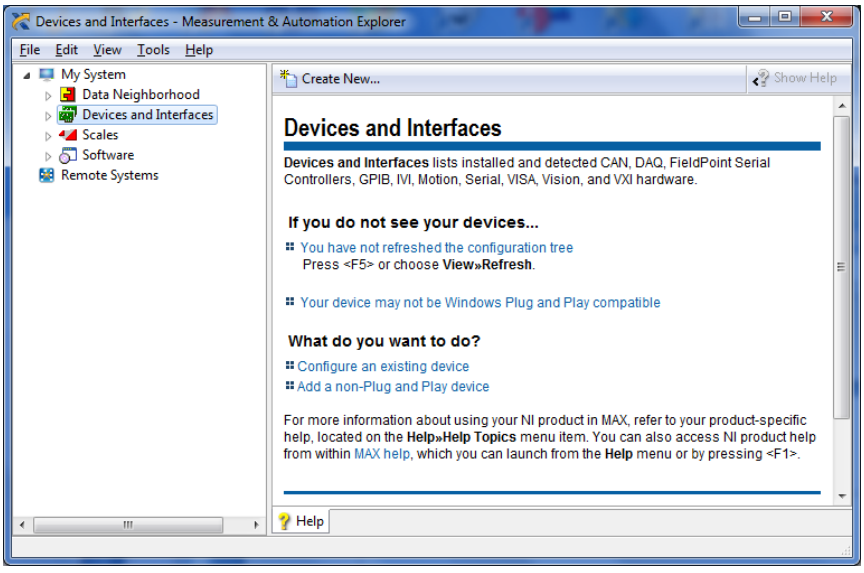
Напомена: Променљива time.clock***()*** је стварно системско време, променљива time.clock()-vreme је реално време од почетка експеримента па до текућег тренутка времена, а променљива ukupno броји (временским кораком ***T*** (периода одабирања)) циклусе у којима је издата наредба за читање са картице. Код система који раде у реалном времену – на сваких T (периода одабирања) секунди се прецизно генерише тзв. прекидна рутина, односно, у том тренутку се зауставе сви процеси рачунара док се не изврши прекидна рутина у којој се сигнал узима са АД конвертора. Овде то није

случај, и одабирање није сасвим прецизно. У којој мери је успело програмирање у реалном времену се може проверити на крају експеримента (да ли је ukupno = time.clock()-vreme) Ако се ова времена разликују, то значи да се одабирање није дешавало тачно на сваких *T* секунди, а та кашњења су се десила јер је рачунар био запослен извршењем и других програма чије је започето извршење ишло до краја и није се прекидало на сваких *T* секунди). Ако оперативни систем рачунара није ”real time” ова времена ће се мање или више разликовати. Ако рачунар има мање ”послова” блискост ових времена је већа. Постоји могућност да се активирањем програма за аквизицију искључе сви остали процеси рачунара док се програм не изврши, али се опет у програму могу појавити процеси који одузимају време и праве непрецизно одабирање.

Напомена: За идентификацију броја уређаја који је потребно унети у функцији за задавање вредности излаза [SetAO(broj uređaja, broj izlatnog kanala, napon u voltima)], и функцији за читање са улаза [ReadAIChanel(broj uredjaja, broj ulaznog kanala)], потребно је покренути претходно инсталиран софтвер NI MAX (инсталира се са NIDAQmx, о коме је претходно било речи).

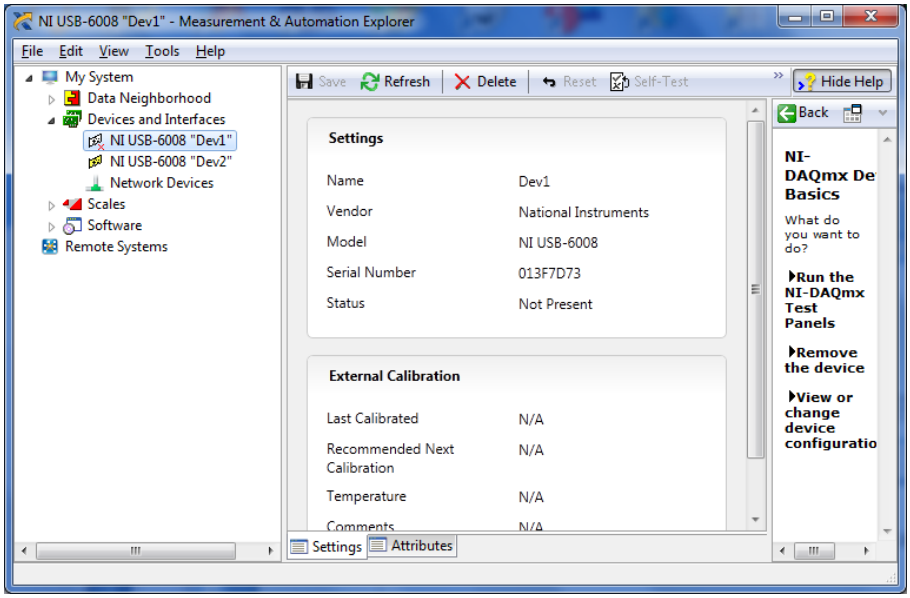
Уколико се пречица не налази на десктопу, покретање се врши на следећи начин: Start/All Programs/National Instrments/MAX/NI MAX.

После покретања појавиће се прозор као на слици 4.2.1.2.



Слика 4.2.1.2. *NI MAX након покретања*

У стаблу са леве стране, кликнути на стрелицу поред Devices and Interfaces, после чега ће се појавити листа уређаја са ознаком у формату “Dev1” (Слика 4.2.1.3). Број поред ознаке *Dev*, користи се за идентификацију броја уређаја при задавању поменутих наредби у пајтон програмском решењу. Уколико је у одељку Settings статус уређаја “Not Present” (није присутан) проверити статус следећег уређаја са листе.



Слика 4.2.1.3. *NI MAX, идентификација броја уређаја*

Напомена важи за било који случај коришћења NI USB 6008, из програмског језика Python.

4.2.2 Креирање програма за снимање одзива са синусним улазним напоном

Програм за снимање одзива на синусну побуду, разликује се само у две линије кода унутар while петље. После промене, петља изгледа овако:

```
while ukupno < 5:
            while T+ukupno > time.clock() - vreme:
            pass
      ukupno += T
      u = 2.5+2.5*math.sin(ukupno*2*math.pi/0.5)
      daq.SetAO(1,1,u)
      y.append((daq.ReadAIChanal(1,0)))
      x.append(ukupno)
```

```
u = 2.5+2.5*math.sin(ukupno*2*math.pi/0.5)
```

израчунава синус у дискретним временским тренуцима, на свако T (променљива укупно се повећава за T, сходно условима из петље), са периодом од 0.5. Како је $\omega = 2\pi/T_0$, периодом T_0 дефинисана је кружна учестаност синуса ω.

Облик синусне функције је следећи: $\boldsymbol{u = A+Asin(\omega t)}$, сабирање са $\boldsymbol{A}$, је отуда што картица не може да генерише негативан напон (опсег излаза је од 0 – 5 V), па се подешава да минимална вредност амплитуде добијеног сигнала не буде -2,5 већ 0.

Алтернативно, функција за израчунавање синуса може се записати и на други начин:
u = 2.5+2.5*math.sin(ukupno*w), али је онда претходно потребно дефинисати вредност кружне учестаности w = жељена вредност.

```
daq.SetAO(1,1,u)
```

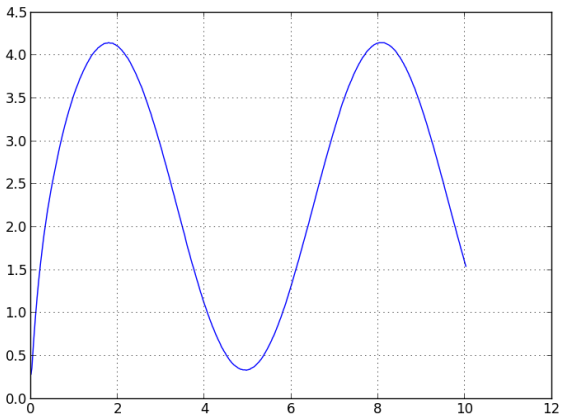
прослеђује израчунат синус на ***AO 1*** излаз картице.

Као што је поменуто, остатак кода је исти као у претходном случају. Потребно је изменити две линије кода на описан начин, сачувати фајл под новим именом и покренути га.

Целокупни код дат је у прилогу 3 (поглавље 6.3).

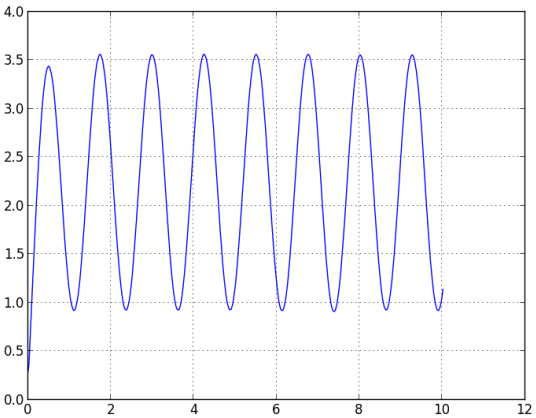
Примери одзива на синусни улазни напон различитих учестаности могу се видети видети на следећим сликама:

Одзив система на синусни улаз за $\boldsymbol{\omega}$ = 1 и **A** = 2.5 приказан је на слици 4.2.2.1.



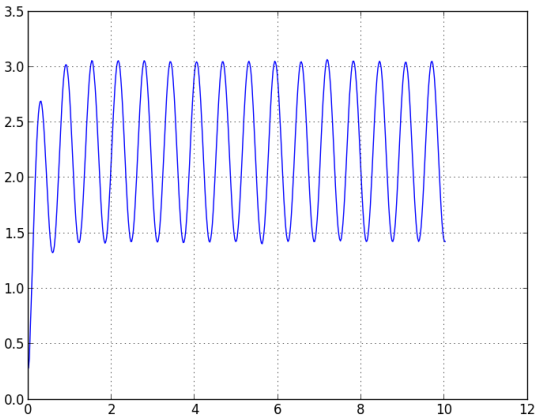
Слика 4.2.2.1. *Одзив на синусни улаз 1*

Одзив система на синусни улаз за $\boldsymbol{\omega}$ = 5 и **A** = 2.5 приказан је на слици 4.2.2.2.



Слика 4.2.2.2. *Одзив на синусни улаз 2*

Одзив система на синусни улаз за $\boldsymbol{\omega}$ = 10 и **A** = 2.5 приказан је на слици 4.2.2.3.



Слика 4.2.2.3. *Одзив на синусни улаз 3*

5. Закључак

Основни циљ рада био је да се прикаже примена NI USB 6008 AD/DA конвертора коришћењем Python програмског језика у апликацијама мерења и управљања. Ради комплетније слике, укратко је описан сам процес анаогно-дигиталне (и супротно) конверзије, дате су карактеристике NI 6008 AD/DA конвертора, детаљно је описан процес припреме за екперимент и поступак писања програма у Python-у.

За екперимент, коришћено је једноставно редно електрично коло, које се састоји из једног отпорника и кондензатора (RC коло), као систем првог реда, због једноставности и познатих теоријских основа. На крајеве кола је са аналогних излаза NI 6008 довођен напон раличитих карактеристика, и са крајева кондензатора преко аналогног улаза у картицу сниман је напон, као одзив на различите улазе у процес. У раду су коришћени само аналогни улази и излази картице, с тим да је такође, решењима у Python-у, могуће користити и дигиталне улазе и излазе.

Кроз добијене резултате показано је да је могуће користити NI картицу путем Python програмског језика, и дат је простор за даље усавршавање и развитак самог процеса.

6. Прилози

6.1 Прилог 1. Основи програм

```
def CHK(err):
    if err < 0:
        buf_size = 1
        buf = ctypes.create_string_buffer('\000' * buf_size)
        nidaq.DAQmxGetErrorString(err,ctypes.byref(buf), buf_size)
        raise RuntimeError('nidaq call failed with error %d: %s'%(err,repr(buf.value)))



def SetAO(DevID = 1, AOchanalID = 0, value = 5):
    chanal = "Dev{0}/ao{1}".format(DevID, AOchanalID)
    data = float64(1.0)
    # opseg vrednosti između 0 and 5V
    value = min(max(value,0),5)
    value64 = float64(value)
    print chanal, value, value64
    taskHandle = TaskHandle(0)
    CHK(nidaq.DAQmxCreateTask("",ctypes.byref(taskHandle)))
    CHK(nidaq.DAQmxCreateAOVoltageChan(taskHandle,
                                       chanal,
                                       "",
                                       float64(0.0),
                                       float64(5.0),
                                       DAQmx_Val_Volts,
                                       ""));
```

```
    CHK(nidaq.DAQmxStartTask(taskHandle));
    CHK(nidaq.DAQmxWriteAnalogScalarF64(taskHandle,
                                1,
                                float64(10.0),
                                value64,
                                None));
    if taskHandle.value != 0:
        nidaq.DAQmxStopTask(taskHandle)
        nidaq.DAQmxClearTask(taskHandle)

def ReadAIChanal(DevID = 1, AIchanalID = 0):
    chanal = "Dev{0}/ai{1}".format(DevID,AIchanalID)
    hi=float64(10.0)
    lo=float64(-10.0)
    #hi=float64(5.0)
    #lo=float64(0.0)

    # initialize variables
    taskHandle = TaskHandle(0)
    max_num_samples = 1000
    data = float64(0.0)
    #data = numpy.zeros((max_num_samples,),dtype=numpy.float64)
    CHK(nidaq.DAQmxCreateTask("",ctypes.byref(taskHandle)))
    CHK(nidaq.DAQmxCreateAIVoltageChan(taskHandle,
                                chanal,
                                "",
                                DAQmx_Val_Diff,
```

```
                    lo,
                    hi,
                    DAQmx_Val_Volts,
                    None))
    CHK(nidaq.DAQmxCfgSampClkTiming(taskHandle,
                    "",
                    float64(10000.0),
                    DAQmx_Val_Rising,
                    DAQmx_Val_FiniteSamps,
                    uInt64(max_num_samples)));
    CHK(nidaq.DAQmxStartTask(taskHandle))
    read = int32()
    CHK(nidaq.DAQmxReadAnalogScalarF64(taskHandle,
                    ctypes.c_double(-1),
                    ctypes.byref(data),
                    None))

    #print "Acquired %d points"%(read.value)
    if taskHandle.value != 0:
        nidaq.DAQmxStopTask(taskHandle)
        nidaq.DAQmxClearTask(taskHandle)
    avgData = data.value
return avgData
```

6.2 Прилог 2. Код за снимање одзива на одкочни улазни напон

```
#Zadavanje izlaza sa kartice: daq.SetAO(broj uredjaja, broj kanala, vrednost napona)
#Citanje sa ulaza: daq.ReadAIChanal(broj uredjaja, broj kanala)

import Ni6008DAQ as daq    #uvozi biblioteku Ni6008DAQ, kao daq
import time    #uvozi vreme
import matplotlib
from matplotlib import pyplot as plt    #uvozi biblioteku za crtanje dijagrama
import math     #uvozi biblioteku za matematicke operacije
T = 0.02     #Definisanje periode odabiranja u sekundama
x = []       #definisanje praznog niza za y
y = []       #definisanje praznog niza za x
daq.SetAO(1,1,0) #daq.SetAO(broj uredjaja, broj izlaznog kanala, vrednost napona)
time.sleep(10)   #ceka 10 sekundi
ukupno = 0
daq.SetAO(1,1,5)    #izlazni napona sa kartice (broj uredjaja, broj kanala, vrednost napona
y.append((daq.ReadAIChanal(1,0)))   #zapisivanje u y sa daq.ReadAIChanal(broj
uredjaja,broj kanala)
x.append(0)
vreme = time.clock()    #sistemsko vreme pre while petlje
while ukupno < 5:
    while T+ukupno > time.clock() - vreme:
        pass
    ukupno += T
    y.append((daq.ReadAIChanal(1,0)))
    x.append(ukupno)
daq.SetAO(1,1,0) # vracanje izlaza sa kartice na 0 V
```

```
plt.plot(x,y)   #Stampanje
plt.draw()
plt.grid()
plt.show()
```

6.3 Прилог 3. Код за снимање одзива на синусни улазни напон

```
#Zadavanje izlaza sa kartice: daq.SetAO(broj uredjaja, broj kanala, vrednost napona)
#Citanje sa ulaza: daq.ReadAIChanal(broj uredjaja, broj kanala)
import Ni6008DAQ as daq   #uvozi biblioteku Ni6008DAQ, kao daq
import time  #uvozi vreme
import matplotlib
from matplotlib import pyplot as plt  #uvozi biblioteku za crtanje dijagrama
import math   #uvozi biblioteku za matematicke operacije
T = 0.02   #Definisanje periode odabiranja u sekundama
x = []
y = []
daq.SetAO(1,1,0)  #daq.SetAO(broj uredjaja, broj izlaznog kanala, vrednost napona)
time.sleep(10)   #ceka 10 sekundi
ukupno = 0
daq.SetAO(1,1,0) #izlazni napona sa kartice (broj uredjaja, broj kanala, vrednost napona)
y.append((daq.ReadAIChanal(1,0))) #zapisivanje u y sa daq.ReadAIChanal(broj uredjaja,broj kanala)
x.append(0)
vreme = time.clock() #sistemsko vreme


while ukupno < 5:
while T+ukupno > time.clock() - vreme:
```

```
        pass
    ukupno += T
    u = 2.5+2.5*math.sin(ukupno*2*math.pi/0.5) #Definisanje sinusnog ulaza
    daq.SetAO(1,1,u) #zadavanje sinusnog napona na izlazu iz kartice
    y.append((daq.ReadAIChanal(1,0)))  # zapisivanje u y sa daq.ReadAIChanal(broj uredjaja,broj kanala)
    x.append(ukupno) #zapisivanje vremena u x
daq.SetAO(1,1,0) # vracanje izlaza sa kartice na 0 V
plt.plot(x,y) #Stampanje
plt.draw()
plt.grid()
plt.show()
```

Литература:

[1] М. Матијевић, Г. Јакупановић, Ј. Цар, *Рачунарски подржано мерење и управљање, 2.издање,* Крагујевац, септембар 2008

[2] Т. Шекара, *Практикум за лабораторијске вјежбе из аутоматског управљања,* Електротехнички факултет Универзитета у Београду, (шесто издање), Академска мисао, Београд, 2003.

[3] П. Живковић, *Рачунарски подржано мерење и управљање путем програмског пакета MATLAB/Simulik и NI 6008/6009, Seminarski rad,* Машински факултет у Крагујевцу, Kragujevac, 2008

[4] National instruments corporation, *NI USB-6008/6009 User Guide and Specification,* http://www.ni.com/pdf/manuals/371303m.pdf *(15.02.2015)*

[5] М. Тодоровски, *Рачунарска подршка у извођењу експеримената и анализи резултата мерења, Дипломски рад,* Машински факултет у Крагујевцу, Крагујевац, 2010

[6] John V. Guttag, *Introduction to Computation and Programming Using Python,* MIT, 2013

[7] *Labview окружење: Практикум из софтверских алата 2013,* http://automatika.etf.bg.ac.rs/images/FAJLOVI_srpski/predmeti/izborni_kursevi_os/upravljanje_procesima/OS2PSA/materijali/psa_labview_2013.pdf *(21.03.2015)*

[8] Матијевић et all, *Лабораторијске везбе (Л1, Л2, Л3, Л4), предмет: Мерење и управљање,* ФИН, Крагујевац, шк. 2015/16, 2016/17