Getting started with GVS

A manual on setting up a GVS experiment in the Sensorimotor lab  
*Nynke Niehof*

# Required hardware

National Instruments data acquisition unit (NIDAQ)

* Looks like this:  
  
* Can be found in the Robot Lab
* The NIDAQ is connected to a pc USB port, where it receives a digital signal, and generates an analog signal that is sent to the stimulator

Biopac STMISOLA Linear isolated stimulator

* Looks like this:  
  
* Can be found in the Robot Lab
* Connect the NIDAQ output to the stimulator’s input RCA analog connector (black cord coming out of the back)

Electrodes

* Can be found in the Robot Lab

Oscilloscope (optional)

* Hook it up to the NIDAQ to check the generated analog signal
* Connect it to the STMISOLA with a resistor in between the leads to measure the voltage/current output
* Can be borrowed at TSG (Gerard)

# Required software

NI-DAQmx drivers

* The device type is cDAQ-9171
* Download the drivers from [here](http://search.ni.com/nisearch/app/main/p/bot/no/ap/tech/lang/nl/pg/1/sn/catnav:du,n8:3478.41.181.5495,ssnav:ndr/)
* Note: the driver software package is huge (~2 GB) and requires registration
* Note: you may have to install Microsoft .NET before you install NIDAQ, but you will get a notification of this during installation, and it should be able to install .NET automatically.
* Unzip, install

NIDAQmx API for Python

* This is an interface between the NIDAQmx driver and Python. It provides a way to communicate with the NIDAQ through Python in an object-oriented way.
* Download and install from [source](https://github.com/ni/nidaqmx-python/blob/master/docs/index.rst), or use pip install nidaqmx
* Usage examples found [here](https://nidaqmx-python.readthedocs.io/en/latest/#usage)
* Note: nidaqmx is only tested for Windows. In Linux, you may have to use [PyDAQmx](https://pythonhosted.org/PyDAQmx/usage.html) instead.

# Generating stimuli

A brief manual on how to use the NIDAQ (in C) can be found [here](http://www.ni.com/tutorial/5409/en/#toc3). Below are the same steps, but in Python.

1. Import the nidaqmx package

import nidaqmx

1. Create a task and virtual channel

A *task* is a collection of virtual channels, timing, triggering and other properties. A *virtual channel* is a collection of settings that include a physical (I/O) channel, a name, the type of stimulus to measure or generate, et cetera.

Create task object:

task = nidaqmx.Task()

Physical channel name: choose any analog output channel from ao0 to ao13 (the first 14 COM-ports in the NIDAQ’s black connector block)

physicalChannelName = "cDAQ1Mod1/ao0"

In the collection of analog output channels (ao\_channels), create a voltage channel

task.ao\_channels.add\_ao\_voltage\_chan(  
 physicalChannelName,  
 name\_to\_assign\_to\_channel="GVSoutput",  
 min\_val=-3.0,  
 max\_val=3.0,  
 units=nidaqmx.constants.VoltageUnits.VOLTS)

1. Write a signal to the output channel

Create a writer

writer = nidaqmx.stream\_writers.AnalogSingleChannelWriter(

task.out\_stream, auto\_start=True)

Create my\_signal as a Numpy array, then write it to the channel:

writer.write\_many\_sample(my\_signal)

This will generate the analog signal, which will be sent to the STMISOLA, and will be turned into a GVS signal.

1. When finished, close the task

task.stop()  
task.close()

Example code for a GVS project can be found on [GitLab](https://gitlab.socsci.ru.nl/nniehof/GVSnoise)