

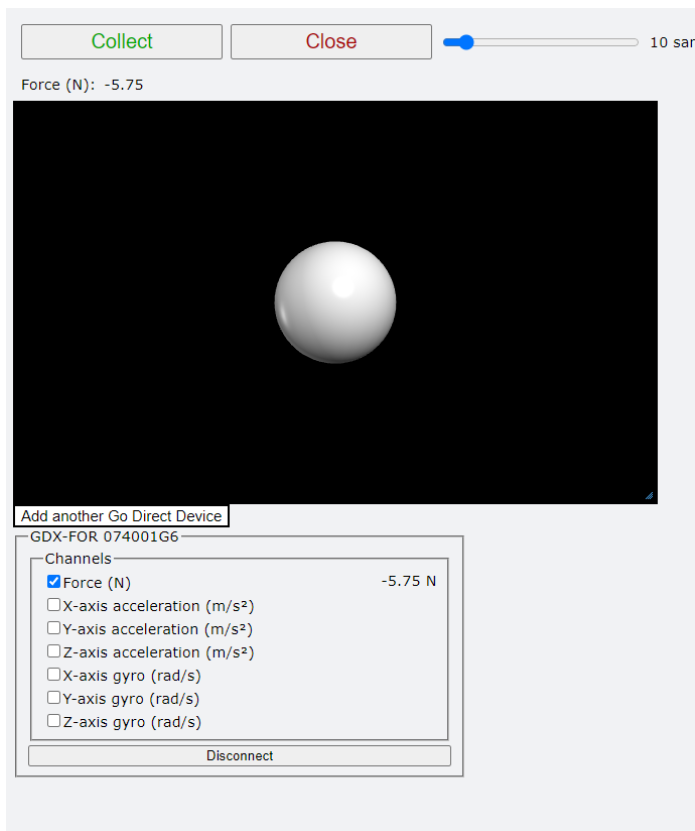
Vernier Go Direct® Sensors and Web VPython

Version 11/13//2022

This guide shows you how to get started writing Web VPython programs to communicate with most Vernier Go Direct devices. Web VPython (formerly known as Glowscript) is an easily accessible online coding platform that bundles Python and VPython functions together to allow you to easily create great graphics. It was developed to simplify the creation of physics animations and simulations in Python. It works on Windows, Macintosh, Linux computer, or Chromebooks. The great thing is that there is nothing to install. Internet access is required.

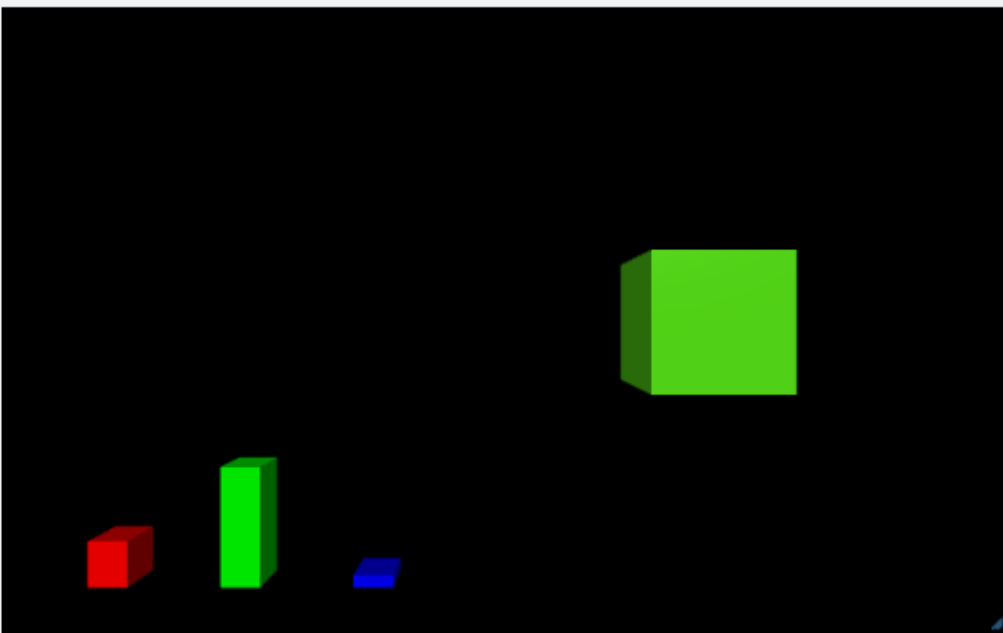
We worked with Bruce Sherwood, the lead developer of Web VPython, to allow most of our Go Direct (“GDX”) sensors to collect data with Web VPython. We would like to thank Bruce for helping us with this project, but also for creating such an amazing, free, programming environment. For more on the history of VPython, e history of VPython, see <https://brucesherwood.net/?p=136> .

Here are some screenshots of Web VPython programs collecting data from our GDX sensors:

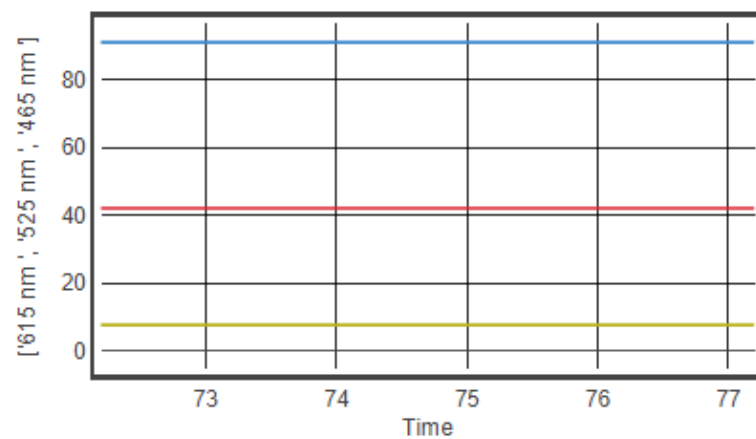


Stop Close 10 samples/second

615 nm : 42.06 525 nm : 90.89 465 nm : 7.69



Add another Go Direct Device



GDX-LC 09100449

Channels

☐ Light (lux)

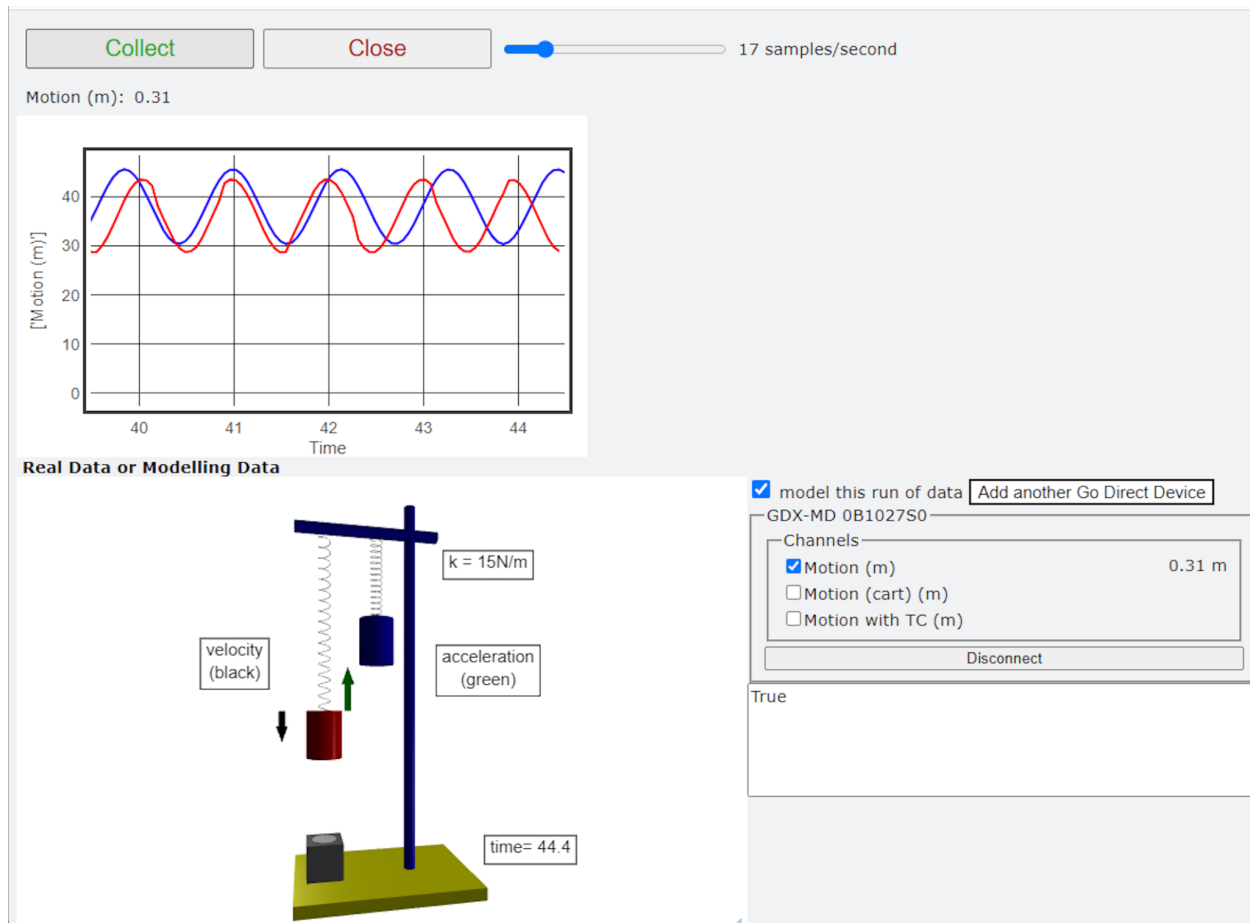
☐ UV

☒ 615 nm 42.06

☒ 525 nm 90.89

☒ 465 nm 7.69

Disconnect



Note that we have a separate guide for using the same Go Direct sensors with installed Python and VPython. After installing a few files, you can take data with Mac, Windows, and Linux computers. For information on how to use installed Python or VPython see vernier.com/engineering/python.

This guide is just about Web VPython and contains the following topics:

- Introduction
- Take a look at a very simple GDX sensor program
- About the Vernier Canvas
- Tips for using Web VPython
- About Our Sample Programs
- Support

If you are new to Python you should start with the simple program described first and then venture further as you get comfortable. If you are familiar with Python, look over the Getting Started section and then start experimenting with the sample programs. Refer to this document

if you have questions. There is also a Getting Started Guide for Web VPython posted on our web site.

Note: Working with most Vernier GDX sensors works as described here. There are a few of our GDX chemistry and biology sensors (including spectrometers, Cyclic Voltammetry System, Melt Station, Blood Pressure, Mini GC, and Polarimeter) are not supported. A complete list, with order codes is listed in the Support section of this manual.

Introduction

Requirements

All that is needed to start using a Vernier GDX sensor with Web VPython is a Windows, Mac, Linux computer, or a Chromebook, a browser, and the GDX sensor. It is probably best to use a Chrome or Edge browser. There are some browsers that do not work well.

Set up and Sign onto a Web VPython Account

Go to www.webvpythont.org and click on `Sign in`. If you already have a Google account, just sign in. If you do already have a Google account, you can create one. You do not have to give any other personal information. You just need to make up (and remember) a username and password.

Your work in Web VPython is automatically saved to your account online. You can create folders and rename and delete files.

Coding with Web VPython

Once you have created a Web VPython account and signed in, and you are ready to start programming.

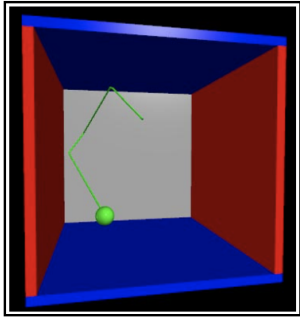
Web VPython

HelpSign in

VPython is an easy-to-use, powerful environment for creating 3D animations. Here at glowscript.org (or webvpython.org, which takes you here), you can write and run VPython programs right in your browser, store them in the cloud for free, and easily share them with others. You can also use VPython with installed Python: see vpython.org.

The [Help](#) provides full documentation.
[Welcome to VPython](#), a [Trinket](#) tutorial, is useful for anyone new to programming in VPython.

See the [Example programs](#).
 To get started writing your own programs you need to Sign in.



Version 3.2

Example programs | Forum

One of the simplest possible programs is listed below. Try it, just to make sure things are working. You do this by choosing `Run this program` (right above the code) after you have typed the short program.

box by dvernier 2022/11/14 09:50:45 (Saved)

Signed in as dvernier(Sign out)

Run this program

Share or export this program

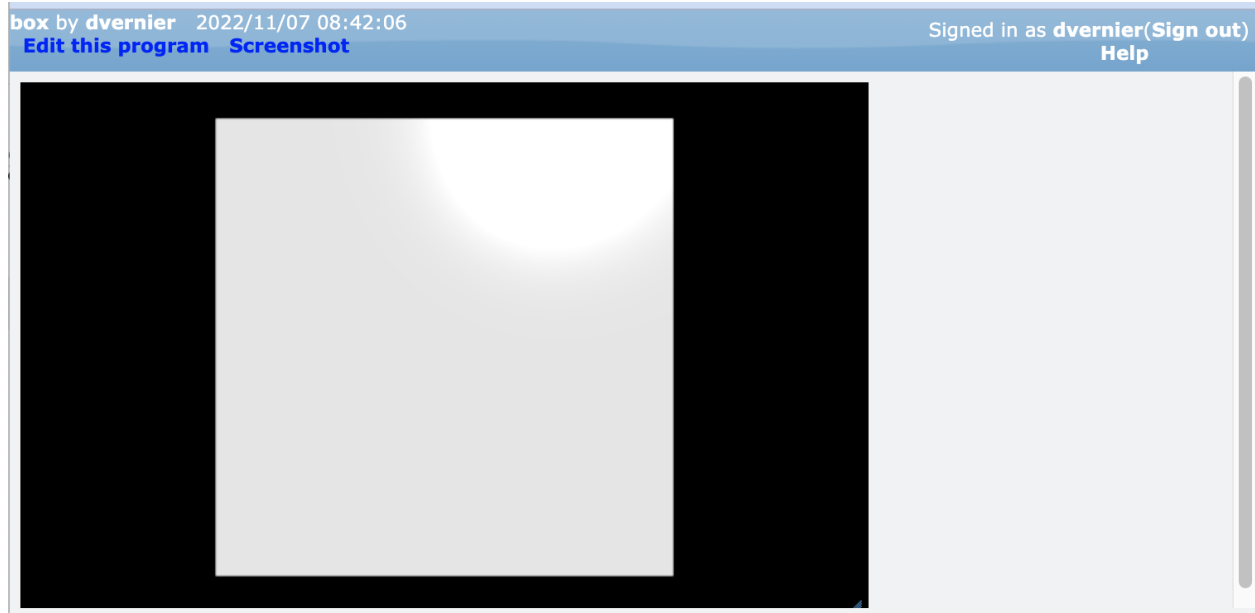
Download

Help

```

1 Web VPython 3.2
2 box()
```

This program should draw a box on a new browser window. You can control the view of the box in many ways. To rotate the view, drag with right button or Ctrl-drag.
 To zoom, drag with middle button, or Alt/Option depressed, or use scroll wheel.
 On a two-button mouse, middle is left + right. To pan left/right and up/down, Shift-drag.
 On a touch screen: pinch/extend to zoom, swipe or two-finger rotate.



If the box does not appear, click on the word Help on the right side. The Help built into Web VPython is very complete. There are many great Web VPython tutorials on the internet. Some are listed in the Support section of this Guide.

Take a look at a very simple GDX sensor program

It is easy to incorporate Go Direct sensor data into your Web VPython program! If you have a GDX sensor, give it a try. You can connect a sensor by either USB or Bluetooth. Start with USB, because it is just one step simpler. Choose a relatively simple sensor like a temperature, force, light, or pressure sensor, if you have one. Do not use a Motion Detector, Sound, or Rotary Motion for this program (they are set up a bit differently).

A simple program using one of these GDX sensors looks like this (This is the example program `gdx_getting_started_usb`. This program is the VPythonGettingStarted example in our collection of programs at <https://www.glowscript.org/#/user/vernier-web-vpython/folder/MyPrograms/>

```
VPythonGDXGettingStartedUSB by dvernier 2022/11/08 15:10:14 Signed in as dvernier(Sign out)
Run this program Share or export this program Download Help

1 Web VPython 3.2
2 get_library("https://unpkg.com/@vernier/godirect/dist/webVPython.js")
3 gdx.open(connection='usb')
4 gdx.select_sensors([1])
5 gdx.vp_vernier_canvas()
6 ball = sphere(pos=vector(0,1,0),radius=1, color=color.red)
7 gdx.start(100)
8 while gdx.vp_close_is_pressed() == False:
9     rate(500)
10     while gdx.vp_collect_is_pressed() == True:
11         rate(500)
12         measurements = gdx.read()
13         if measurements == None:
14             break
15         ball.radius=measurements[0]
16
```

Let's go through this program line-by-line to explain what each line is doing:

Web VPython 3.2

The first line is required for all Web VPython programs.

```
get_library("https://unpkg.com/@vernier/godirect/dist/webVPython.js")
```

This line provides the link to JavaScript libraries that allow Web VPython to read our GDX sensors. You must have internet connectivity to use these libraries.

```
The.gdx.open(connection='usb')
```

This line tells the program how you would like to connect to a Vernier GDX sensor. You can change the 'usb' to 'ble', if you want to connect using Bluetooth.

```
gdx.select_sensors([1])
```

This line specifies which of the channels of your GDX sensor you want to read. You can add as many channels as the sensor has, separated by commas. Information about available channels is included in this manual, in the Support section.

```
gdx.vp_vernier_canvas()
```

This line is not actually required, but it adds a lot of features to make data collection with GDX sensors easy. The simplest version, which we used here, will add Collect/Stop and Close buttons a slider for controlling data collection rate. After you connect to a GDX sensor, it will add a live "meter" display of the active channels, and a channel selection box, listing what channels you are reading. You can also customize the statement to add a graph, or to leave off any of these features. This is all explained later in this manual.

```
ball = sphere(pos=vector(0,1,0),radius=1,color=color.red)
```

This is standard VPython to create an object we can work with. It is easy to change the ball's position, size, or color (or other properties).

```
gdx.start()
```

This line starts data collection. By default, it will set the data collection period to 100 ms. If you put a number in the parentheses, it controls the time between sensor readings in milliseconds. Note that you can also control this sample period by using the slider as the program runs.

```
while gdx.vp_close_is_pressed() == False:
    rate(500)
    while gdx.vp_collect_is_pressed() == True:
        rate(500)
```

These four lines are used to set up the data collection loop. Use this configuration most times you use the `gdx.vp_vernier_canvas()` line. It is really pretty standard VPython code. The loop will start when the Collect button is pressed and run until the Close button is pressed. The rate

statements are needed to prevent the program from monopolizing the browser when nothing is happening in the loop.

```
measurements = gdx.read()
```

This line reads all the channels on the GDX sensor. They will be returned as a 1-dimensional list.

```
if measurements == None:
```

```
    Break
```

These lines handle the situation when there is no data to read. They stop the data collection loop. They are not really required for Web VPython, but we include them here because they make our Web VPython programs more compatible with installed VPython.

```
ball.radius=measurements[0]
```

This line reads the first channel value (the first item in the list) and changes the size of the ball object accordingly.

Connect a GDX sensor using a USB cable and try out the program. The program listed above is at

<https://www.glowscript.org/#/user/vernier-web-vpython/folder/MyPrograms/program/VPythonGettingStartedUSB> . Open a browser and go to that link. Copy the whole program, log onto your Web VPython account, choose Create a new Program and paste in this code. This version of the program is yours to work with and modify as you want.

Note that if you have this program open and click on Run this program, you will get an error message. You can simply run most Web VPython programs, but when the program involves hardware devices, for security reasons, this is not allowed. Instead you have to go through this procedure:

Rather than choosing “Run this program”, choose “Share or export this program”

PythonGDXGettingStartedUSB by dvernier 2022/11/08 15:10:14 Signed

Run this program Share or export this program Download

```
1 Web VPython 3.2
2 get_library("https://unpkg.com/@vernier/godirect/dist/webVPython.js")
3 gdx.open(connection='usb')
4 gdx.select_sensors([1])
5 gdx.vp_vernier_canvas()
6 ball = sphere(pos=vector(0,1,0),radius=1, color=color.red)
7 gdx.start(100)
8 while gdx.vp_close_is_pressed() == False:
9     rate(500)
10    while gdx.vp_collect_is_pressed() == True:
11        rate(500)
12        measurements = gdx.read()
13        if measurements == None:
14            break
15        ball.radius=measurements[0]
```

Sharing VPythonGDXGettingStartedUSB by dvernier
Signed in as dvernier (Sign out) Help

There are several ways to share or export a program:

- You can publish a link to <https://www.glowscript.org/#/user/dvernier/folder/AAPT/program/VPythonGDXGettingStartedUSB>. Anyone can run the program here at glowscript.org, or view its source, without having to sign in. **The program must be in a public folder, not a private folder.**
- You can publish a link to an entire public folder: <https://www.glowscript.org/#/user/dvernier/folder/AAPT/>.
- Export: you can embed the program, including from a private folder, directly in your own web page, using the HTML code provided below. If your page already uses jquery, you should delete that <script> tag.
- Another option is to save the HTML code shown below in a file ending in ".html", then doubleclick the html file and it will run in your browser.
- Or simply click **Download as HTML** and then double-click the downloaded file to run it. Some browsers display the file name at the bottom of the page, and a single click on the name will run the program.
- You can also place the html file in an iframe portion of a web page by placing in the html of the web page a statement like this :

```
<iframe src="test.html" width="320" height="340"></iframe>
```

 where "test.html" is the location of the program relative to the location of the iframe statement, and where the program has a canvas somewhat smaller than 320 by 340.
- You may have a need to allow anyone to run the program but to keep the program's original source code private. It is not possible to hide JavaScript code from a user of the program (rightclick the web page and choose "View page source" or use the browser debugger), but you can "minify" the program to make it so difficult to read as to make it effectively private. To do this, in the HTML code above, copy the code from "// START JAVASCRIPT" through "// END JAVASCRIPT" and paste it into a "minifier" such as the one at javascript-minifier.com, and then replace the START-END code with the output of the minifier. You will see that it is very difficult to understand the minified program, even if you pass it through a "beautifier" program, because many of the variable names have been replaced by single letters.

**If the area below is blank, the program has errors.
Fix the errors and then try again.**

```
<div id="glowscrip" class="glowscrip">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<link type="text/css" href="https://www.glowscript.org/cas/redmond/2.1/jquery-ui.custom.css" rel="stylesheet" />
<link type="text/css" href="https://www.glowscript.org/cas/ide.css" rel="stylesheet" />
<script type="text/javascript" src="https://www.glowscript.org/lib/jquery/2.1/jquery.min.js"></script>
<script type="text/javascript" src="https://www.glowscript.org/lib/jquery/2.1/jquery-ui.custom.min.js"></script>
<script type="text/javascript" src="https://www.glowscript.org/package/glow.3.2.min.js"></script>
<script type="text/javascript" src="https://www.glowscript.org/package/R$run.3.2.min.js"></script>
<script type="text/javascript"><!--//--><![CDATA]//><!--

// START JAVASCRIPT
;(function() {
var g0_modules = {};
g0_modules.pythonize = {};

(function(){
    function strings() {
        var string_funcs, exclude, name;
        string_funcs = set(["capitalize lstrip lstrip rstrip islower issuper isspace lower upper swapcase center count endswith startswith find rfindex
index rindex format join ljust rjust partition rpartition replace split splitlines zfill".split(" ");
        if (arguments.length) {
            exclude = (function){
                var s = g0.set();
                s.jssset.add("split");
                s.jssset.add("replace");
                return s;
            }();
        }
    }
})());
```

If you are using a Chrome browser, an icon for the downloaded file will usually appear at the bottom left of the browser and you can just click on that to run it. The file created will actually be in your downloads folder.

VPythonGDXGettingStartedUSB by dvernier 2022/11/08 15:10:14 Signed in as dvernier(Sign out)
Run this program Share or export this program Download Help

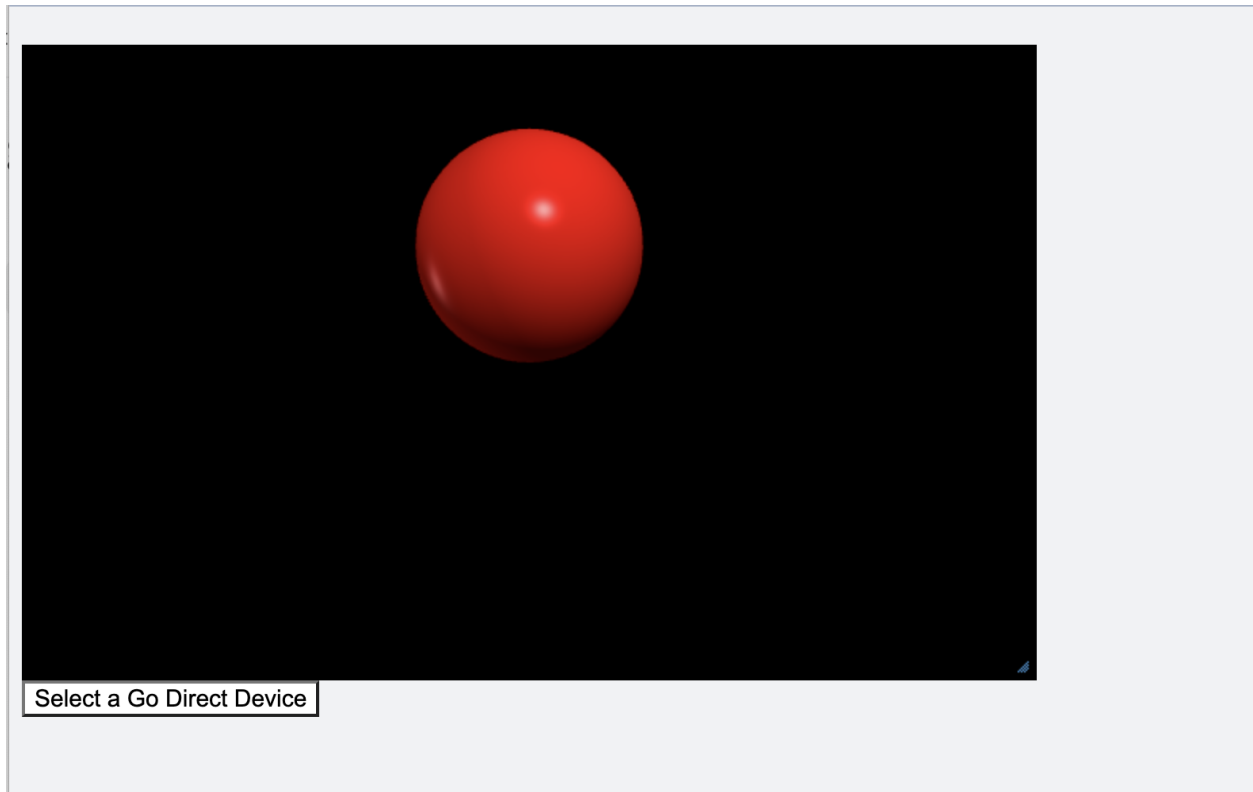
```
1 Web VPython 3.2
2 get_library("https://unpkg.com/@vernier/godirect/dist/webVPython.js")
3 gdx.open(connection='usb')
4 gdx.select_sensors([1])
5 gdx.vp_vernier_canvas()
6 ball = sphere(pos=vector(0,1,0),radius=1, color=color.red)
7 gdx.start(100)
8 while gdx.vp_close_is_pressed() == False:
9     rate(500)
10    while gdx.vp_collect_is_pressed() == True:
11        rate(500)
12        measurements = gdx.read()
13        if measurements == None:
14            break
15        ball.radius=measurements[0]
16
17
18
```

VPythonGDXGe....html Show All X

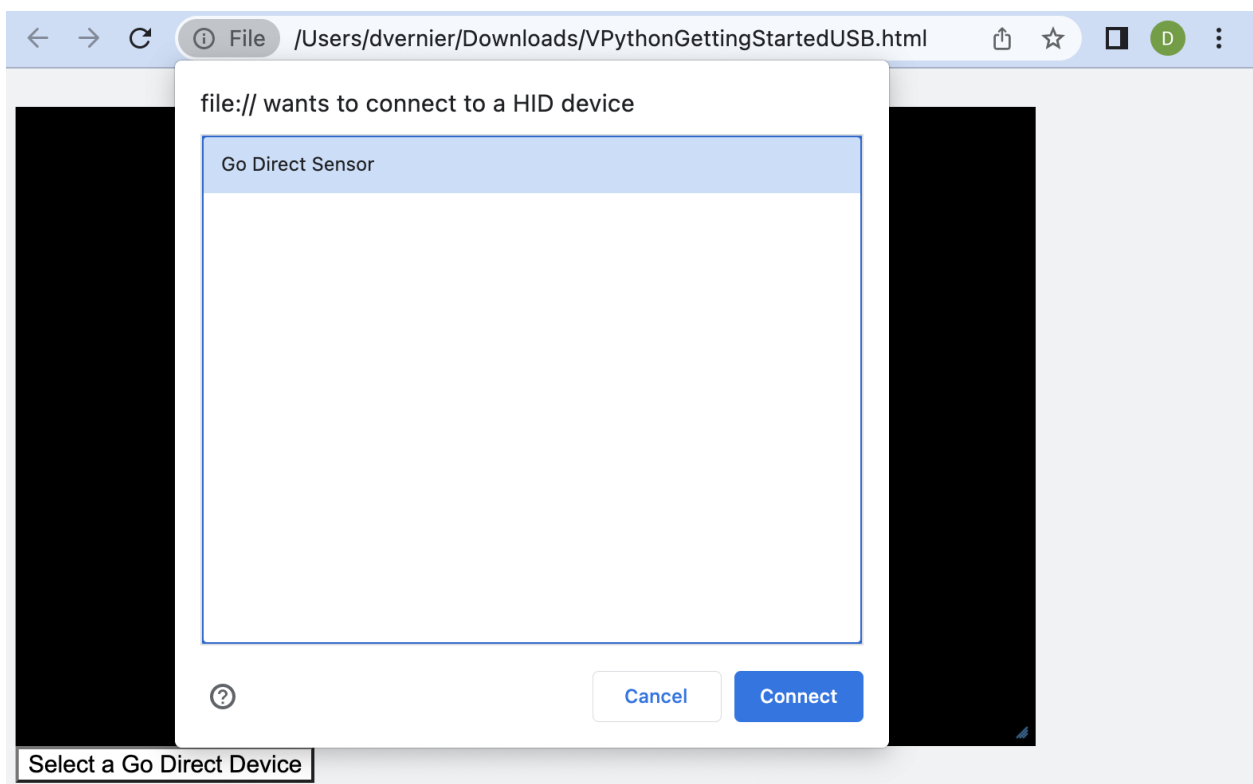
Now, you will probably think this is a clumsy way to run a program, and it is, but it works well and you get used to it. One advantage of this, is that you are creating executable HTML files that can be used by anyone, even if they have never heard of Web VPython or even Python.

The downsides of this system are that you end up with lots of downloads and lots of open tabs in your browser. You will need to do periodic cleanup operations.

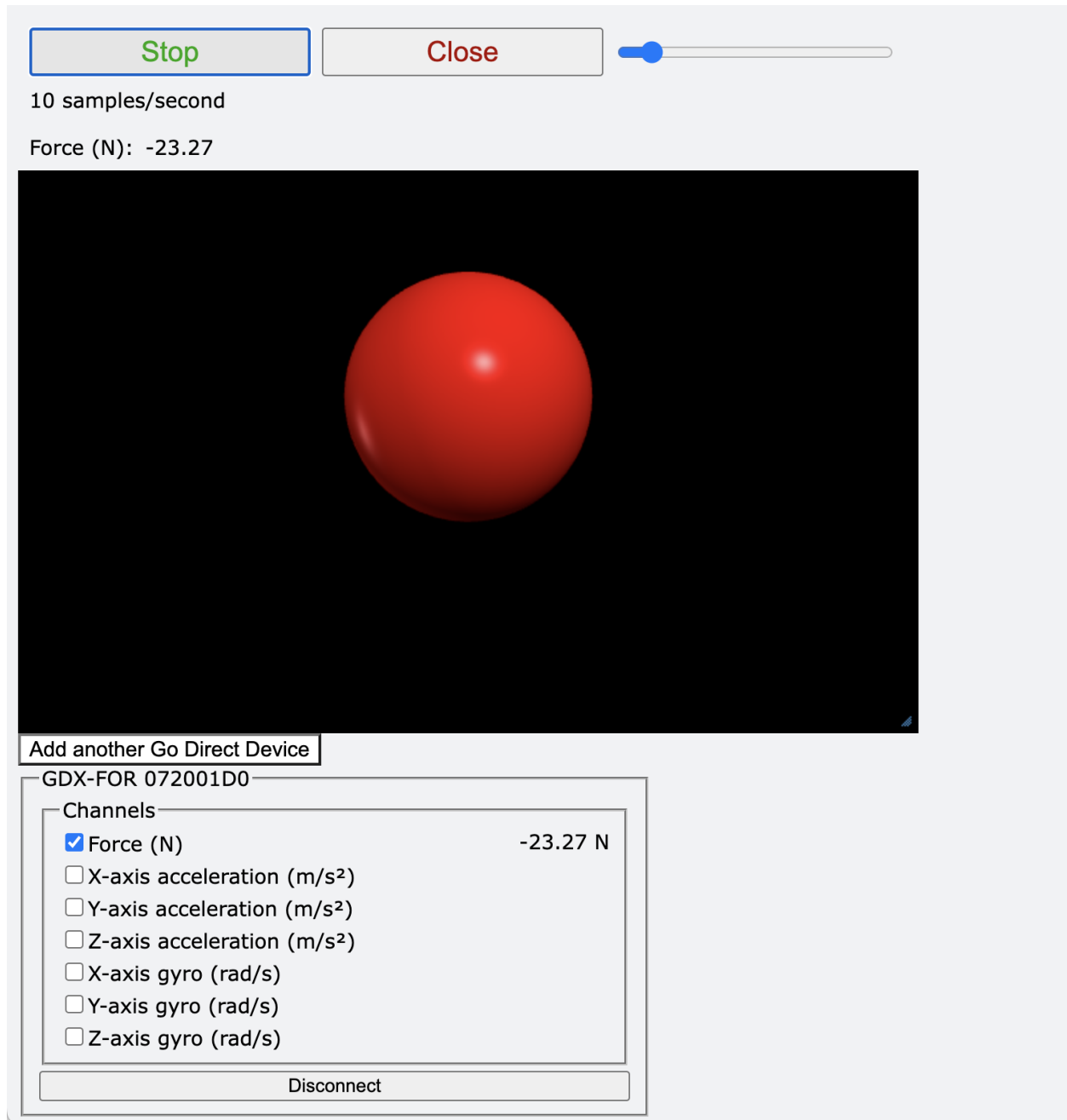
When the program runs, you will see the features created with the `gdx.vp_vernier_canvas()` statement and the ball object. There is a button labeled `Select a Go Direct Device`.



Click on that button and you should see a list of available devices. There will probably be only one on the list. Click on it to connect.



When the sensor is connected, you will see a screen like the one below. Note that there is a live, “meter” display of the sensor reading. There is a channel selection box showing which channel of the GDX device is being used.



Click on the Collect button to start data collection. Your sensor reading will control the size of the box.

Click on Stop when you want to stop data collection.

Click on Close to gracefully shut down the connection between the GDX sensor and the computer.

More information on the.gdx Functions

Here is some more information about the.gdx functions, including how you might add arguments to a few of the functions:

gdx.open(connection=USB) or gdx.open(connection=USB)

There are two parameters that can be used, `connection=` and `device_to_open=`

- The `connection=` parameter can be set to connect Go Direct devices by USB 'usb' or Bluetooth 'ble'

```
gdx.open(connection='usb')
```

```
gdx.open(connection='ble')
```

Also, you can shorten the command to `gdx.open('usb')` or `gdx.open('ble')`. You can also use the upper case letters "USB" and "BLE".

- If you are using two GDX devices, it is useful to add the `device_to_open=` parameter with your device names. This will allow you to specify the channels for the particular GDX devices. Here is one example:

In the Color Match example program, we use a GDX-FOR, force sensor and a GDX-LC at the same time. We use a statement like this:

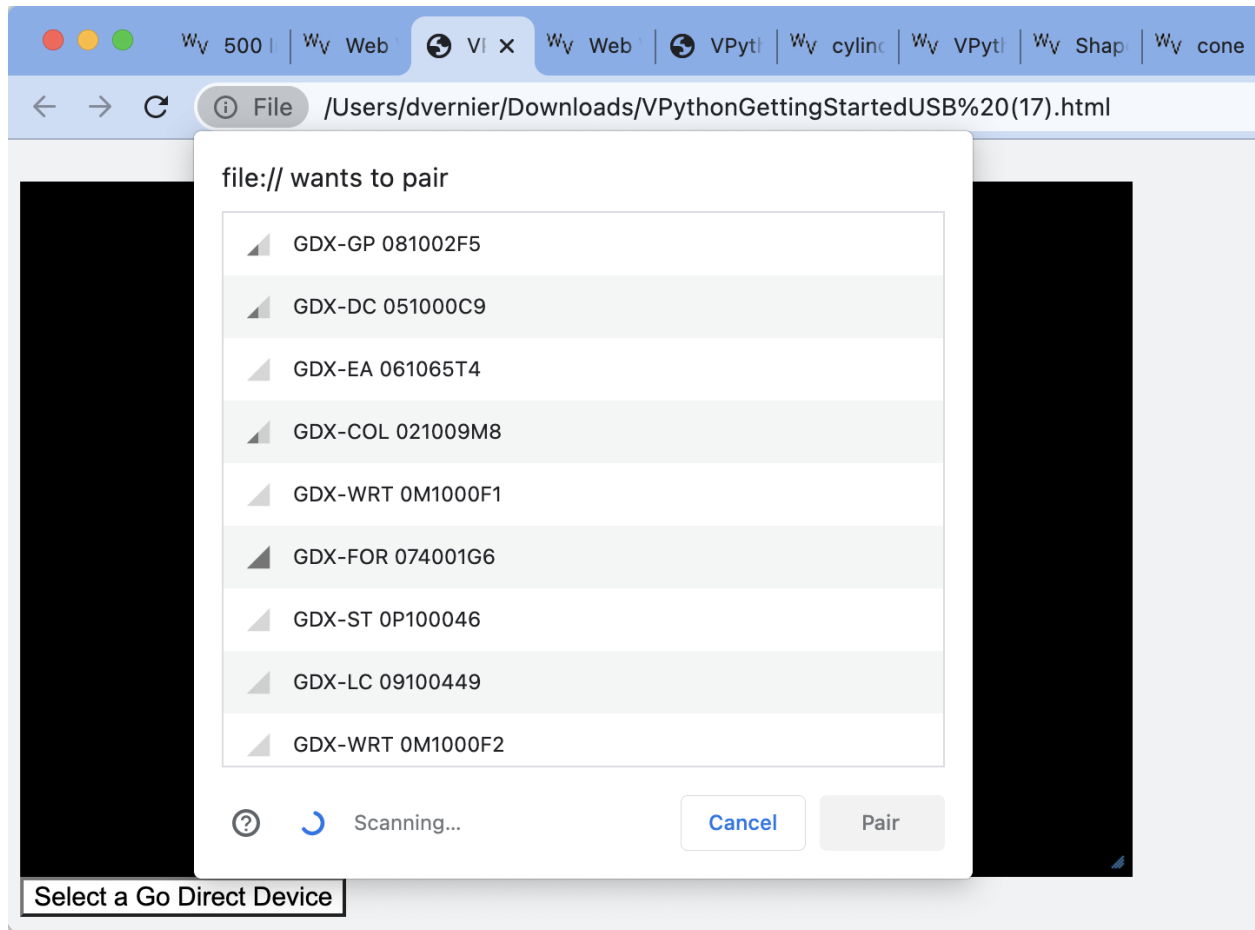
```
gdx.open(connection='usb', device_to_open="GDX-FOR 071000U9, GDX-LC  
151000C1")
```

The reason this is important, is that we need to also specify which channels of each sensor we want to use. After this statement, we are can now use:

```
gdx.select_sensors([[1],[1,5,6,7]])
```

And the program will know that the first list of sensor channels is for the first GDX device listed in the open statement and the second list is for the second one.

Note that if you are using a Bluetooth connection, when you see the list of available sensors, there is a wedge-shaped icon displayed. This is an indication of the Bluetooth radio signal strength. The darker it is, the stronger the Bluetooth signal is. Here is an example made in a room with lots of GDX devices turned on.



gdx.select_sensors()

Note that the format is a list of sensor channels inside square brackets like this

```
gdx.select_sensors([[1]])
```

Or, if you want to use multiple channels on one GDX device, separate the channel numbers with commas::

```
gdx.select_sensors([[1,2]])
```

Or, if you have two GDX devices:

```
gdx.select_sensors([[1],[1,5,6,7]])
```

Note that the channels available on most GDX devices are listed later in this manual. You can use channel 1 for almost any sensor. Two exceptions are: GDX-MD and GDX-RMS. Use channel 5 for them. Also do not use channel 1 with GDX-SND (because it need to sample very quickly on that channel).

If you leave the parameters out, the user can specify the channels by clicking in the Channel Selection Box.

gdx.start()

This command sets the sampling rate, that is, the time between samples in milliseconds. The same rate is used for all selected sensors.

Sampling at a period that is less than 10 milliseconds (in other words, greater than 100 samples/second) may be problematic.

If this function's argument is left blank, a default sample time of 10 milliseconds (100 readings per second) will be chosen. The user can always change the sample time if the slider is on the screen.

measurements = gdx.read()

The `gdx.read()` function will take single point readings from the selected sensors at the desired period and return the readings as a 1D list. The first item in the list is `measurements[0]`, the second is `measurements[1]`, etc.

gdx.vp_get_slider_period()

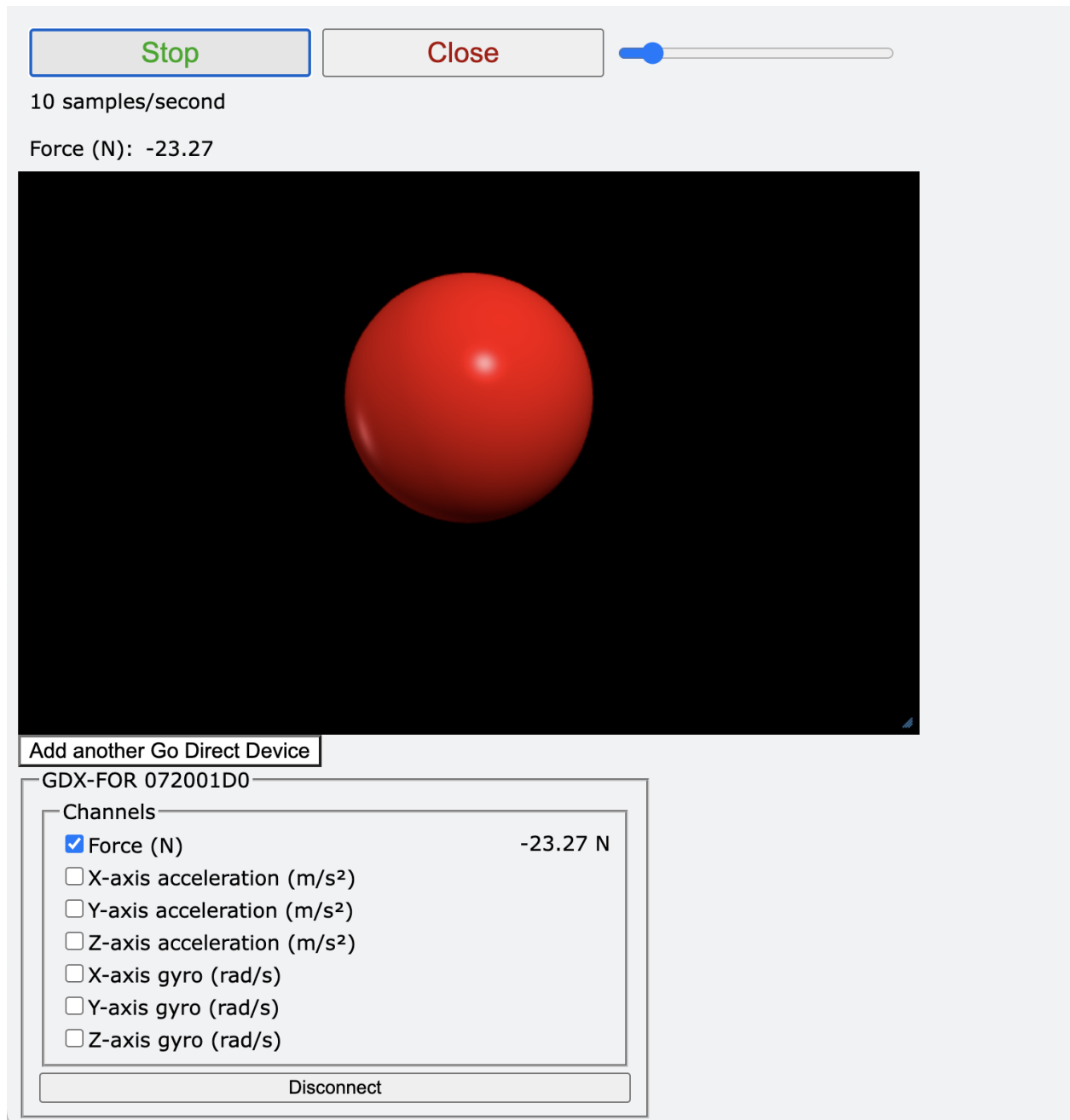
This is not used in our simple sample program above. It returns the period of data collection. It is sometimes needed because the slider can change the data collection rate and other parts of the program may need to know about that change.

About the Vernier Canvas

The function `gdx.vp_vernier_canvas()` adds some features that are often nice for programs involving data collection. The default form:

```
gdx.vp_vernier_canvas()
```

will set up a display like this:



Any GDX data collection program can then be set up so that the COLLECT button starts data collection. It will change to STOP and then you can stop data collection. These operations can be repeated as you like. The CLOSE button will nicely shut down the GDX connection with the computer.

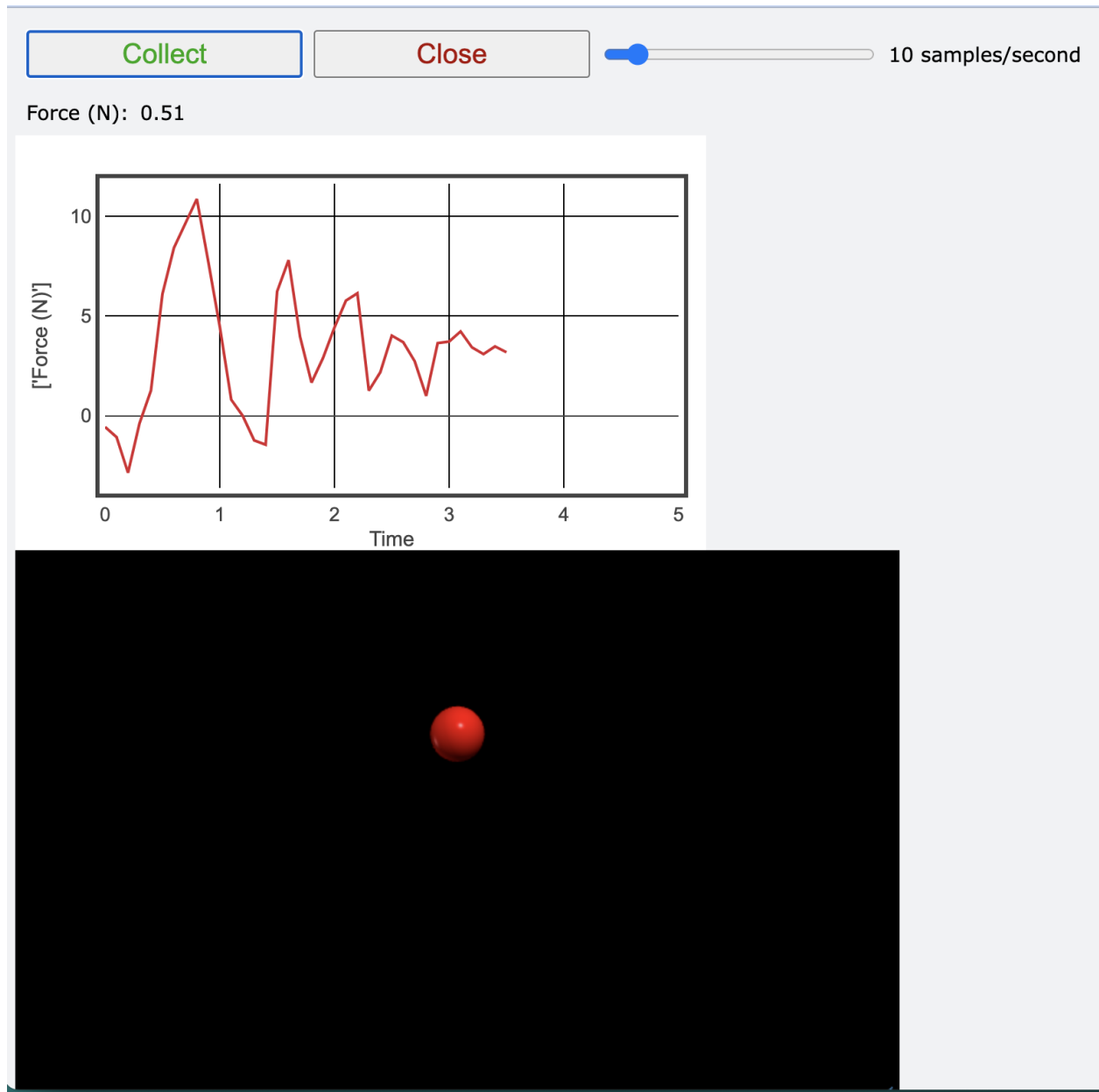
Notice on the sample screen above there is a slider to the right of the buttons that can be used to control the data collection rate of the GDX sensor.

After you are connected to a sensor, by default, below the main canvas, you will see a Channel Selection box. This shows the channels that are active on the connected GDX sensors. If you

do not want the channel selection box, you can eliminate it by modifying your `gdx.vp_vernier_canvas()` statement to:

```
gdx.vp_vernier_canvas(Channel_selection=False)
```

You can also add a graph by adding "chart=True" to the `gdx.vp_vernier_canvas()` function. The function `gdx.vp_vernier_canvas(sensor_setup=False, chart=True)` used with the simple program we have been using yields:



Note that by default, a graph is not included on the Vernier canvas. You have to specify that you want it. The complete format for the `gdx.vp_vernier_canvas` function is:

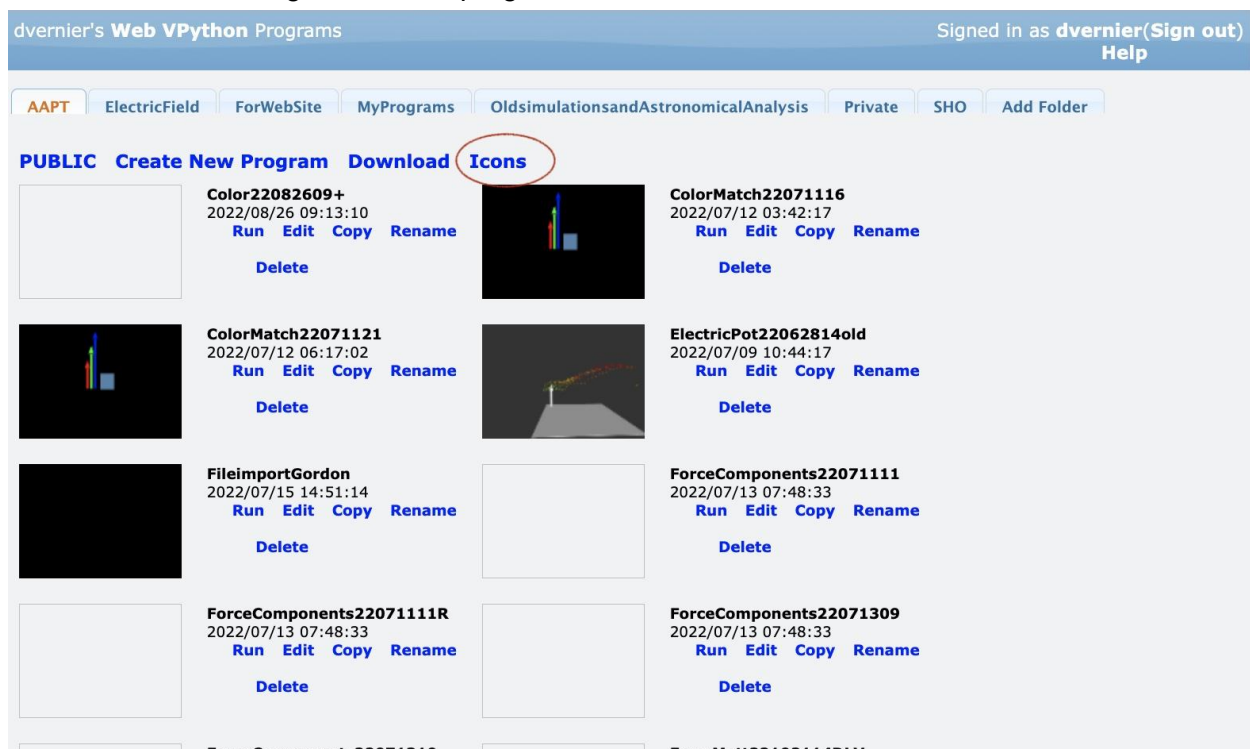
```
gdx.vp_vernier_canvas(buttons=True, meters=True, slider=True,  
channel_setup=True, chart=False)
```

The statement above has the default settings. In short, you will get the buttons, slider, meters, and the channel selection box by default, and you have to specifically turn on the graph feature, if you want it.

Tips for using Web VPython

Lists are Better than Icons

By default Web VPython displays programs as icons. If you click on the word “Icons”, shown below, the Icons change to a list of programs.



Like this:

dvernier's Web VPython Programs
Signed in as dvernier(Sign out)
Help

AAPT
ElectricField
ForWebSite
MyPrograms
OldsimulationsandAstronomicalAnalysis
Private
SHO
Add Folder

PUBLIC
Create New Program
Download
List

	Names ^	Dates			
Run	Color22082609+	2022/08/26 09:13	Copy	Rename	Delete
Run	ColorMatch22071116	2022/07/12 03:42	Copy	Rename	Delete
Run	ColorMatch22071121	2022/07/12 06:17	Copy	Rename	Delete
Run	ElectricPot22062814old	2022/07/09 10:44	Copy	Rename	Delete
Run	FileimportGordon	2022/07/15 14:51	Copy	Rename	Delete
Run	ForceComponents22071111	2022/07/13 07:48	Copy	Rename	Delete
Run	ForceComponents22071111R	2022/07/13 07:48	Copy	Rename	Delete
Run	ForceComponents22071309	2022/07/13 07:48	Copy	Rename	Delete
Run	ForceComponents22071310	2022/09/21 14:25	Copy	Rename	Delete
Run	FromMatt22102114DLV	2022/10/21 13:44	Copy	Rename	Delete
Run	FromMatt22102115DLV	2022/10/21 14:06	Copy	Rename	Delete
Run	FromMatt221025DLV	2022/10/27 09:15	Copy	Rename	Delete
Run	FromMatt221804DLV	2022/10/21 14:15	Copy	Rename	Delete
Run	GDXGettingStartedUSB	2022/11/06 11:56	Copy	Rename	Delete
Run	GettingStartedBruce	2022/09/20 07:47	Copy	Rename	Delete
Run	SHO220628old	2022/07/09 10:47	Copy	Rename	Delete
Run	SHOMat22102113	2022/10/25 12:25	Copy	Rename	Delete
Run	TestingLag	2022/09/07 10:22	Copy	Rename	Delete

The big advantages of this view are that you can sort the programs by name or date, by clicking on the column heading. You can also create a folder and rename or delete programs.

Channels Available on GDX Sensors

For most GDX sensors, you can just use channel 1. This includes temperature, force, light, acceleration, voltage, current, energy, magnetic field, CO2, O2, , colorimeters, conductivity, and carts. Here is a list of available channels on some more complex GDX sensors. You can always check the channels by running this Python program `gdx_getting_started_device_info.py`. This program is available as part of our GDX Python collection at:

<https://vernierst.github.io/godirect-examples/python/>

GDX-3MG

- 1 X magnetic field
- 2 Y magnetic field
- 3 Z magnetic field
- 4 X magnetic field 130mT
- 5 Y magnetic field 130mT
- 6 Z magnetic field 130mT

GDX- ACC

- 1 through 3 acceleration X,Y,Z (m/s^2)
- 4 through 6 acceleration X,Y,Z (high range)(m/s^2)
- 7 through 9 gyro (radians/s)

10 altitude(m)
11 angle (degrees)

GDX-CART

1 Position
2 Force
3 X Acceleration
4 Y Acceleration
5 Z Acceleration

GDX-CCS

1 Current

GDX-CO2

1 CO2 Gas
2 Temperature
3 Relative Humidity

GDX-COL

1 Transmittance

GDX-CON

1 Conductivity 0% Temperature Compensated
2 Conductivity
3 Temperature

GDX-DC

1 Volume

GDX-EA (used for pH)

1 Potential
2 pH

GDX-EKG

1 EKG
2 Heart Rate
3 EMG
4 EMG Rectified
5 Voltage

GDX-ETOH

1 Ethanol Vapor

GDX-ISEA

- 1 Potential
- 2 Nitrate
- 3 Ammonium
- 4 Calcium
- 5 Chloride
- 6 Potassium

GDX-HD

- 1 Force
- 2 X-axis acceleration
- 3 Y-axis acceleration
- 4 Z-axis acceleration
- 5 X-axis gyro
- 6 Y-axis gyro
- 7 Z-axis gyro

GDX-NRG

- 1 Potential
- 2 Current
- 3 Power
- 4 Resistance
- 4 Energy

GDX-LC

- 1 Light (lux)
- 2 UV
- 5 615 nm
- 6 525 nm
- 7 465 nm

GDX-MD

- 5 Motion
 - 6 Motion (cart)
 - 7 Motion with TC
- (all incompatible with one another)

GDX-O2

- 1 O2 Gas
- 2 O2 Gas Temperature Compensated
- 3 Temperature

GDX-ODO

- 1 DO Concentration
- 2 DO Saturation

- 3 Temperature
- 4 Pressure
- 5 DO Salinity

GDX-Q

- 1 Charge
- 2 Potential

GDX-RB

- 1 Force
- 2 Respiration Rate
- 4 Steps
- 5 Step Rate

GDX-RMS

- 5 Angle
- 6 Angle (high resolution)

GDX-SND

- 1 Sound Pressure (needs to be sampled at a high speed)
- 2 Sound Level (A weighted) (can be as slow as sampled a few times a second)
- 3 Sound Level (C weighted) (can be as slow as sampled a few times a second)
- 4 Wave Amplitude

GDX-ST

- 1 Temperature

GDX-TMP

- 1 Temperature

GDX-WRT

- 1 Temperature

GDX-WTHR

- 1 wind speed (m/s)
- 2 wind direction (degrees)
- 3 wind chill (degrees C)
- 4 temperature (degrees C)
- 5 heat index (degrees C)
- 6 dew point (degrees C)
- 7 relative humidity (%)
- 8 absolute humidity (g/m^3)
- 9 station pressure (mbar)
- 10 barometric pressure (mbar)

11 altitude (m)

Finding Errors

Web VPython does not have the greatest error reporting. If you try to “Share or export this program” and you get a blank white area below, you have an error. Unfortunately, there is no error message. Here is something that might help, in some cases. Go back to the program and click on “Run this program”. Of course, the program cannot really function, but sometimes it will report an error that you can fix and then you can really run the program in the normal way.

If you click on Run this program and get this error:

```
TypeError: Cannot read properties of undefined (read'__argnames__')
```

That is a good thing. You probably just need to use the standard Share or export this program and the Download as HTML procedure to use the program.

If you get this error message: No Web HID Support

If you get a message like `No Web HID Support` when running a program, you may be using the wrong browser. Mozilla often will not work. Use Chrome or Edge or an equivalent browser.

About Our Sample Programs

Here is list of the sample programs at

<https://www.glowscript.org/#/user/vernier-web-vpython/folder/MyPrograms/>

- VPythonGettingStartedUSB
- ex1-vpython-object
- ex2-vpython-modify-attributes
- ex3-vpython-box-length
- ex4-vpython-sphere-pos
- ex5-vpython-gdx-graph
- gdxfor-live-freebody-diagram
- gdxhd-control-tilt
- gdxlc-color-match
- Gdxmd-simple-harmonic-oscillator
- MappingElectricPotential

Support

Contact us at Vernier Science Education by email at: support@vernier.com or go to our web site and send us a chat message, or call us at 503 277 2299.

Our example programs were designed for use with analog sensors will not perform well for digital sensors like photogates and projectile launchers, and drop counters. Here is a list of GDX Sensors that we would not recommend using with this Python code:

Go Direct Photogate (GDX-VPG)
Go Direct Radiation Sensor (GDX-RAD)
Go Direct Spectrovis Plus (GDX-SVISPL)
Go Direct Visible Spectrophotometer (GDX-SPEC-VIS)
Go Direct Mini GC (GDX-GC)
Go Direct Projectile Launcher (GDX-PL)
Go Direct Cyclic Voltammetry System (GDX-CVS)
Go Direct Polarimeter (GDX-POL)
Go Direct UV-VIS Spectrophotometer (GDX-SPEC-UV)

The Help Built into Web VPython

There is great information on VPython on the internet, starting with the Help built into Web VPython. Just click on the Help button at the top right of the screen. You can then get information on any of the objects you can add to your program, like boxes, spheres, arrows, helices, etc. You can also learn about canvases.

The Web VPython examples at <https://www.glowscript.org/#/user/GlowScriptDemos/folder/Examples/> are just amazing. You can run them, you can view and copy the code. Studying them is a great way to see what is possible and to learn programming tricks.

Other Web Site for Information on Python and VPython

If you are totally new to Python, here are some generally helpful links for getting started with Python.

[Python for Beginners](#)

[Official Python FAQs](#)

Here are some websites with great information about Web VPython:

https://matter-interactions.trinket.io/00_welcome_to_vpython#/welcome-to-vpython/getting-started

<https://rhattallain.com/category/python/>

<https://www.youtube.com/watch?v=8M9q0tydzMA>