Serial Device Communication for Testing and Prototyping

Sarah Bennedsen

Python at Formlabs

- Formlabs engineers from Software to Mech E tend to be fairly python-literate
- Engineers can write their own test interfaces
- Pandas, matplotlib, numpy, etc. enable everyone to do their own analysis

Systems Integration and Prototyping

- Print Process and Systems Engineers can prototype behaviors and algorithms quickly
    - Using high-level libraries like numpy makes complex tasks easier
    - R&D experts can test and iterate faster than release cycles
- Once complete, algorithms get converted to C++
    - Rewriting helps harden code and catch bugs
    - Embedded software has the opportunity to focus on architecture and performance
    - Intermediaries like systems engineers ensure that behavior is preserved

Measurement Devices

- We use a wide range of measurement devices

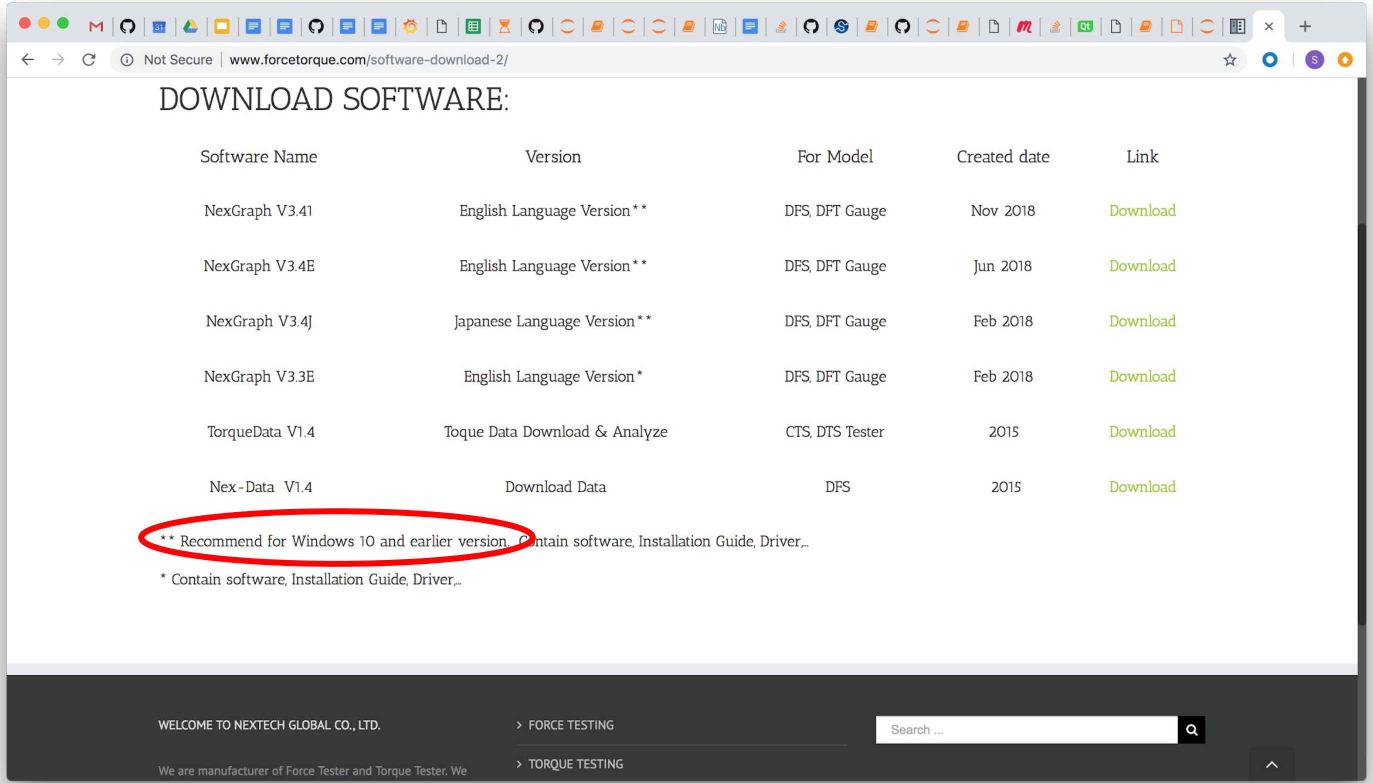
- Often we want particular functionality to perform tests
    - Custom logs
    - Synchronization with other devices or printers
    - The ability to run the device from a printer

Example Device: Nextech Force Gauge

Push/pull gauge to measure forces in compression and in tension.

This happens to be a pretty simple device to talk to. It supports a USB connection, but uses the RS-232 protocol, which is old but not uncommon.

Trying Out Their Software



Not Secure | www.forcetorque.com/software-download-2/

DOWNLOAD SOFTWARE:

Software Name	Version	For Model	Created date	Link
NexGraph V3.41	English Language Version**	DFS, DFT Gauge	Nov 2018	Download
NexGraph V3.4E	English Language Version**	DFS, DFT Gauge	Jun 2018	Download
NexGraph V3.4J	Japanese Language Version**	DFS, DFT Gauge	Feb 2018	Download
NexGraph V3.3E	English Language Version*	DFS, DFT Gauge	Feb 2018	Download
TorqueData V1.4	Toque Data Download & Analyze	CTS, DTS Tester	2015	Download
Nex-Data V1.4	Download Data	DFS	2015	Download

** Recommend for Windows 10 and earlier version, Contain software, Installation Guide, Driver,...

* Contain software, Installation Guide, Driver,...

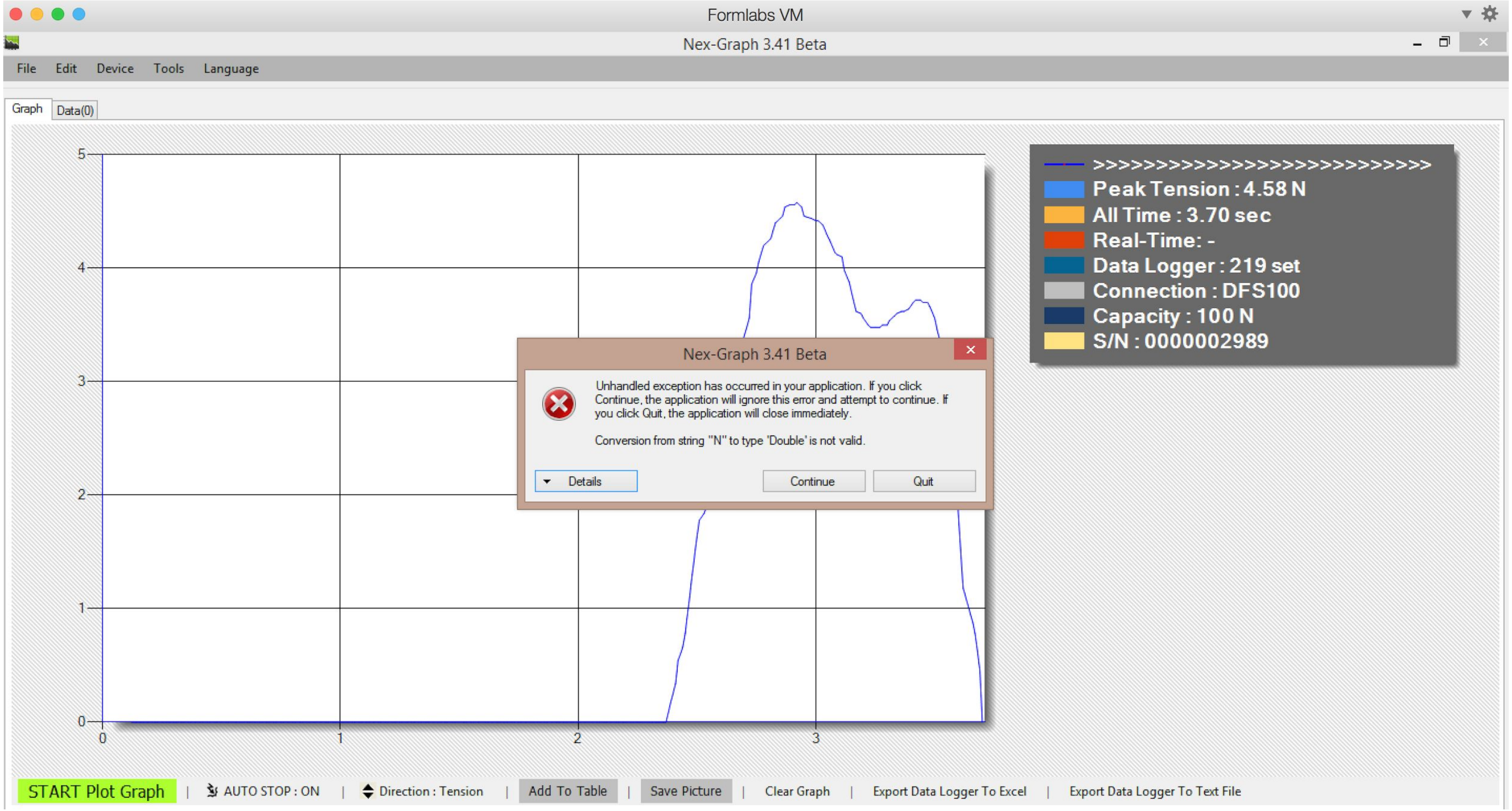
WELCOME TO NEXTECH GLOBAL CO., LTD.

We are manufacturer of Force Tester and Torque Tester. We

› FORCE TESTING

› TORQUE TESTING

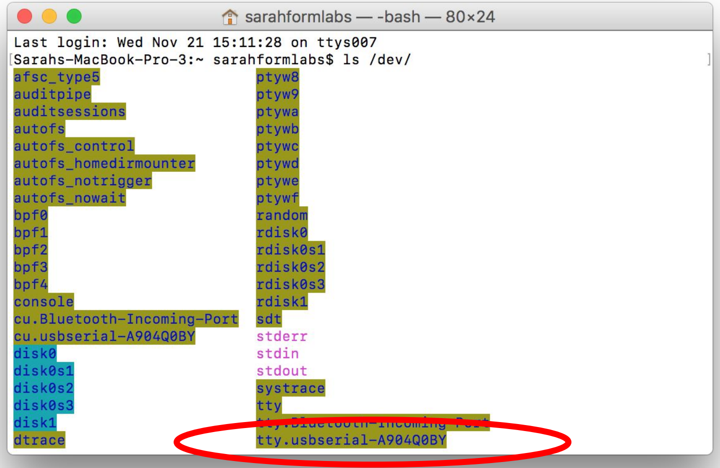
Search ...


Formlabs VM
Nex-Graph 3.41 Beta
File Edit Device Tools Language
Graph Data(0)
>>>>>>>>>>>>>>>>>>>>>>>>>>>
Peak Tension : 4.58 N
All Time : 3.70 sec
Real-Time: -
Data Logger : 219 set
Connection : DFS100
Capacity : 100 N
S/N : 0000002989
Nex-Graph 3.41 Beta
Unhandled exception has occurred in your application. If you click Continue, the application will ignore this error and attempt to continue. If you click Quit, the application will close immediately.
Conversion from string "N" to type 'Double' is not valid.
Details
Continue
Quit
START Plot Graph | AUTO STOP : ON | Direction : Tension | Add To Table | Save Picture | Clear Graph | Export Data Logger To Excel | Export Data Logger To Text File

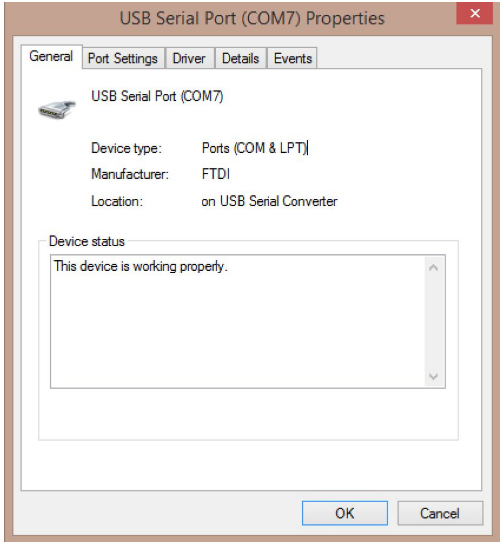
How about some alternatives…

First, how do we find our device?

Mac/Linux: The [/dev](#) directory contains the special [device files](#) for all the devices.



Windows: You can look at your [COM ports](#) in the Device Manager



Okay, now how do we talk to it?

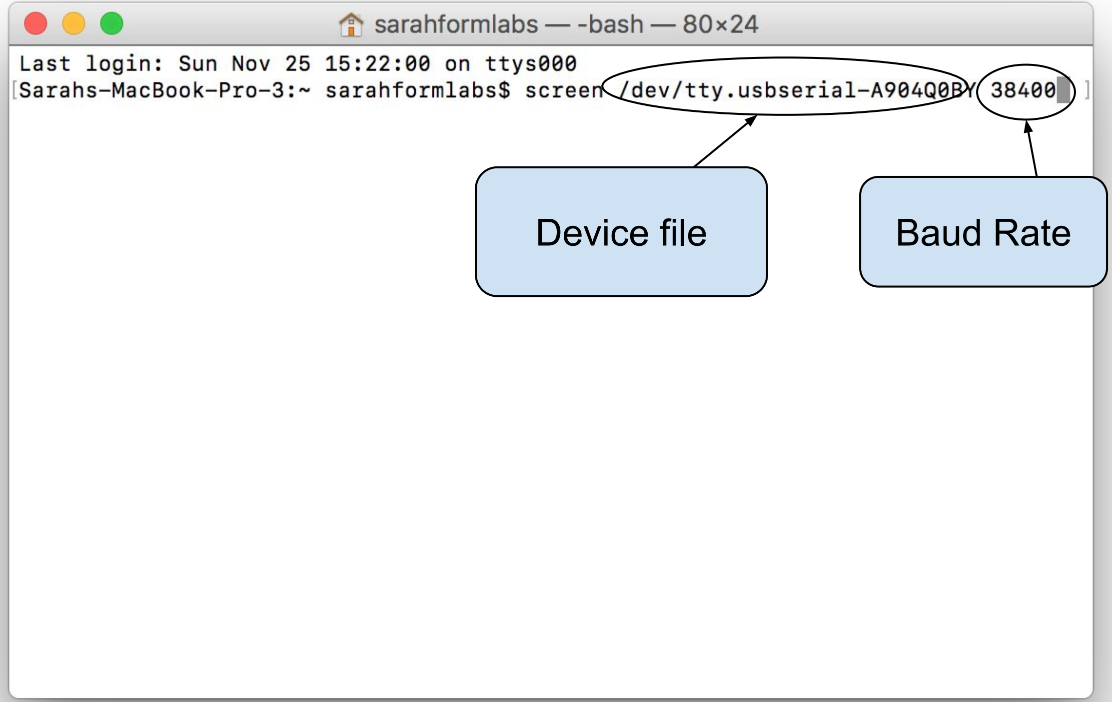
Computer Control of Force Gauge A computer can control the force gauge by sending commands through either USB or RS232 port.

Command	Action
"m"	Cycle modes of measurement.
"u"	Cycle units of measurement.
"z"	Zero the gauge.
"r"	Reset to previously set "Zero".

Output signal The displayed reading may be transmitted to a PC by pressing the PRINT key or sending a request command from a PC to the gauge through either the USB or RS232 port.

Command	Action
"l"	Send live reading value from unit.
"p"	Send peak tension value from unit.
"c"	Send peak compression value from unit.
"x" or pressing PRINT key	Send live reading value from unit, if current mode is track mode. Send peak tension value from unit, if current mode is peak tension mode. Send peak compression value from unit, if current mode is peak compression mode.
"d"	Send memory
"!"	Send information of gauge (model, capacity, serial number, firmware revision, original offset, current offset, overload count).

Screen (or PuTTY)



Screen (or PuTTY)

```
sarahformlabs — screen /dev/tty.usbserial-A904Q0BY 38400 ▸ SCREEN — 80...
0.00     N       Live Compression
0.00     N       Live Tension
0.00     N       Live Compression
0.00     N       Live Compression
0.00     N       Live Compression
0.00     N       Live Compression
DFS 100N 0000002989 3.08 +0.00 +0.07 05
T: + 4.58 N
T: + 4.58 N
T: + 4.58 N
T: + 0.00 N
T: + 0.00 N
T: + 3.16 N
```

Is there an easy way to get more functionality?

PySerial

Finding and connecting to your device

```
In [1]: from serial.tools import list_ports

In [2]: import serial

In [3]: ports = list(serial.tools.list_ports.grep('usb'))

In [4]: serial_connection = serial.Serial(ports[0].device, 38400, timeout=1)
```

Discovering device info

```
In [25]:  ports = list(serial.tools.list_ports.comports())

In [26]: ports[0]
Out[26]: <serial.tools.list_ports_common.ListPortInfo at 0x10b789780>

In [27]: ports[0].device
Out[27]: '/dev/cu.Bluetooth-Incoming-Port'

In [28]: ports[1].device
Out[28]: '/dev/cu.usbserial-A904Q0BY'

In [29]: ports[1].hwid
Out[29]: 'USB VID:PID=0403:6001 SER=A904Q0BY LOCATION=20-2'
```

Write to the device

```
In [44]: serial_connection.write(b'm')
Out[44]: 1

In [45]: serial_connection.write(str.encode('m'))
Out[45]: 1
```

Using read()

```
In [81]: serial_connection.write(str.encode('x'))
Out[81]: 1

# Reads one byte
In [82]: serial_connection.read()
Out[82]: b'3'

# Waits for 50 bytes, but times out and returns
In [83]: serial_connection.read(50)
Out[83]: b'.16\tN\tPeak Tension\t\r\n\r'
```

Using readline()

```
In [46]: serial_connection.write(str.encode('x'))
Out[46]: 1

# Reads until a newline or a timeout
In [47]: reading = serial_connection.readline()

In [48]: reading
Out[48]: b'\r3.16\tN\tPeak Tension\t\r\n'

In [49]: reading.decode().split()
Out[49]: ['3.16', 'N', 'Peak', 'Tension']
```

Don’t forget

```
In [84]: serial_connection.close()
```

Now what?

- Build out connection logic so that for the next user this is plug-and-play
- Make functions for reading and writing all desired messages
- Add logging functionality
- Connect to a device of choice! For example,
  - Attach to a data mule laptop and run lifetime testing for springs or flexures
  - Plug into a printer and make custom logs to examine force during a print
- Make a live plotter...

```python
import serial
import serial.tools.list_ports


class NextechDFS:
    SEND_LIVE_READING_CMD = b'l'
    PRODUCT_ID = '0403:6001'

    def __init__(self):
        detected_ports = list(serial.tools.list_ports.grep(self.PRODUCT_ID))
        self.serial = serial.Serial(detected_ports[0].device, baudrate=38400, timeout=1)

    def get_reading(self):
        self.serial.write(self.SEND_LIVE_READING_CMD)
        return self.serial.readline().decode().split()[2]

    def __enter__(self):
        return self

    def __exit__(self, type, value, traceback):
        self.serial.close()
```

```python
import matplotlib.pyplot as plt
import matplotlib.animation as animation
import force_gauge_demo


def animate(i, xs, ys):
    xs.append(i)
    ys.append(gauge.get_reading())

    # format this next frame of your plot
    ax.clear()
    ax.plot(xs[-30:], ys[-30:], marker='.')
    plt.xlabel('Sample Number')
    plt.ylabel('Force (N)')


with force_gauge_demo.NextechDFS() as gauge:
    fig, ax = plt.subplots()
    ani = animation.FuncAnimation(fig, animate, fargs=([], []), interval=200)
    plt.show()
```