

Interactive USB measurement device controlling with Python

Benedikt Bieringer

DPG-Frühjahrstagung 2023

Online version with code: https://2xb.github.io/usb4research

living.knowledge

Institut für Kernphysik





Outline

- Introduction
- How to write Graphical User Interfaces
- How to connect to devices how manufacturers intend it
- How to connect to devices writing own drivers

living.knowledge

Interactive USB measurement device controlling with Python

Mi.

40



Interactive USB measurement device controlling with Python



(Source: Volker Hannen)

Benedikt Bieringer



Example: iMCAr, the interactive MCA recorder software



- Self-written GUI & drivers
- Supports:
 - CAEN N957 8K MCA
 - ORTEC EasyMCA 8K
 - ORTEC ADCAM 926
- Supports automatically repeated measurements
- Uses faultguard to avoid data loss
 (https://github.com/2xb/faultguard)

Benedikt Bieringer

DPG-Frühjahrstagung 2023 https://2xb.github.io/usb4research







There's lots of software already

Don't re-invent the wheel (except for a good reason)!

- Look for already existing solutions
- For data storage & UI, Grafana + InfluxDB for time series enables users to combine data across devices
- Many USB devices already have generic drivers as part of multiple operating systems
 - Some libraries for such drivers exist (e.g. gphoto2 for cameras)



Outline

- Introduction
- How to write Graphical User Interfaces
- How to connect to devices how manufacturers intend it
- How to connect to devices writing own drivers

living.knowledge

Interactive USB measurement device controlling with Python

Mi.

0



Graphical User Interface | It's about users

Things to know about UI design: Generally speaking:

- Users combine features in unexpected ways (that may break your code)
 ⇒ Work together with other users early
- Users ignore unintuitive features
 ⇒ Make everything as simple as possible

General rule: You want users to think about their physics, not your software.



Graphical User Interface | Concept

1) Concept (sketch)

- Grab a paper, start sketching
- Show the sketch to colleagues/users, ask them how they'd intuitively use this software – be prepared for surprises!





Graphical User Interface | Rapid prototype

2) Rapid prototype

- First implementation is always dirty
- Might as well acknowledge that, get it running with compromises as quickly as possible
- Use e.g. LabView or libraries here that help you get results quickly, even if they have a lot of papercuts for end users
- Discuss the prototype with users!



(Source: Volker Hannen)



Graphical User Interface | Implement

3) Implement – My go-to approach:

- Design UI in Qt Designer, save .ui file
- Use e.g. PyQt or PySide to load .ui file (PyQt & PySide differ in their license)
- Use PyQtGraph for interavtive plotting widgets Note: Never trust any plotting library without testing!

Embed pyqtgraph plots using Qt Designer: https://pyqtgraph.readthedocs.io/en/latest/getting_ started/how_to_use.html





Graphical User Interface

- 1) Concept (sketch)
- 2) Rapid prototype
- 3) Implement

Consider these steps for other projects as well!



Help, my GUI freezes! | How GUIs handle events



Everything is running one line after the other!

If no second thread is used, every wait time freezes the GUI

Second thread can communicate with main thread event loop

In Qt:

Events = Signals Event processors = slots

https://realpython.com/python-pyqt-qthread/



Demo

living.knowledge





A GUI is nice – but how does it interact with my device?



Outline

- Introduction
- How to write Graphical User Interfaces
- How to connect to devices how manufacturers intend it
- How to connect to devices writing own drivers

living.knowledge

Interactive USB measurement device controlling with Python

Mi.

40



USB communication under the hood





USB communication under the hood





USB communication using existing library

1) Find documentation for the library

Usually provided by manufacturer

2) Interface with that library in whatever language it supports (e.g. C)

Manufacturer sometimes provides example code

3) If required: Write a wrapper for the language you want to use (e.g. Python)

For sending large amounts of data between Python & C/C++, use NumPy: https://pythonextensionpatterns.readthedocs.io/en/latest/cpp_and_numpy. html





But if the manufacturer's library doesn't work for me?

- Requires old software (remember ActiveX?) [Yudong Sun, deRSE23]
- Requires old/wrong operating system
- Difficult to install

Potential solution: Use generic library (such libraries exist for scanners/printers/keyboards/mice/cameras/storage/...)



But if the manufacturer's library doesn't work for me?

- Requires old software (remember ActiveX?) [Yudong Sun, deRSE23]
- Requires old/wrong operating system
- Difficult to install

Potential solution: Use generic library (such libraries exist for scanners/printers/keyboards/mice/cameras/storage/...)

...else: Write your own drivers & library!



Outline

- Introduction
- How to write Graphical User Interfaces
- How to connect to devices how manufacturers intend it
- How to connect to devices writing own drivers

Interactive USB measurement device controlling with Python

Mi.

40

intend it

living.knowledge



USB communication under the hood, part II





USB communication under the hood, part II



Benedikt Bieringer

DPG-Frühjahrstagung 2023 https://2xb.github.io/usb4research



Understanding existing drivers: Reverse engineering





Understanding existing drivers: Wireshark

*	Edit View G	o Capture Apa	lyze Statistics Telenh	ony Wireless	*usbmon1			~ ^
				⊷ K ≫				
!(fra	ame.len == 64)							× 🗗 🔹
	^ Time	Source	Destination	Protocol	Lengtl Leftover Capture Data	Endpoint	Info	
1	127 3.917332	host	1.34.2	USB	80 1000000010002100000000000000000000000	0x02	URB_BULK out	
1	130 3.919364	1.34.1	host	USB	76 0c00000030001200000000	0x81	URB_BULK in	
1	131 3.919572	host	1.34.2	USB	76 0c000000100011001000000	0x02	URB_BULK out	
1	134 3.920103	1.34.1	host	USB	345 190100000200011001000000640006000006400136d006900630072006f007	3006f0066 0x81	URB_BULK in	
1	136 3.920363	1.34.1	host	USB	76 0c000000300012001000000	0x81	URB_BULK in	
1	137 3.920715	host	1.34.2	USB	76 0c000000100041002000000	0x02	URB_BULK out	
1	140 3.921072	1.34.1	host	USB	84 1400000020004100200000010000001000100	0x81	URB_BULK in	
1	142 3.921237	1.34.1	host	USB	76 0c0000000300012002000000	0x81	URB_BULK in	
1	143 3.921293	host	1.34.2	USB	88 180000000100071003000000fffffff00000000ffffffff	0x02	URB_BULK out	
1	146 3.992063	1.34.1	host	USB	84 1400000020007100300000010000000400011	0x81	URB_BULK in	
1	148 3.992407	1.34.1	host	USB	76 0c000000300012003000000	0×81	URB_BULK in	
1	149 3.992600	host	1.34.2	USB	88 180000001000710040000000100010000000000ffffffff	0×02	URB_BULK out	
1	152 4.091476	1.34.1	host	USB	84 140000002000710040000000100000000400011	0×81	URB_BULK in	
1	154 4.091828	1.34.1	host	USB	76 0c000000300012004000000	0×81	URB_BULK in	
1	155 / 002022	host	1 2/ 2	IISR	8/ 1/000000010000100500000000000000000000	0402	HRR RILLK out	
ram S I E E E E E E F F	ne 127: 80 byte iection number: interface id: 0 incapsulation ty vrrival Time: Ma Time shift for ipoch Time: 1679 Time delta fror Time delta fror Time since reff irame Length: 80 iapture Length: 80 isometric marker terme is marker	s on wire (640 bi 1 (usbmon1) ype: USB packets tr 23, 2023 00:42 this packet: 0.0 528579.419040000 a previous captur previous displa prence or first f 27 b) bytes (640 bits 80 bytes (640 bit (5a)ea	ts), 80 bytes captured with Linux header and p :59.419040000 CET 000000000 seconds] seconds ed frame: 0.002654000 s yed frame: 0.002781000 rame: 3.917332000 secor) ts)	(640 bits) of badding (115) seconds] seconds] ids]	h interface usbmon1, id 0 0 000 00 86 fd d5 90 a0 ff 1 0 000 83 92 1b 64 00 00 00 0 022 10 00 00 00 00 00 00 0 023 00 00 00 00 00 00 00 00 00 0 0040 10 00 00 00 00 00 00 00 00 0 040 10 00 00 00 01 00 02	F 53 83 62 22 01 06 24 00 00 64 64 66 08 64 ff ff 67 00 </td <td>S . "</td> <td></td>	S . "	
L								

Benedikt Bieringer

DPG-Frühjahrstagung 2023 https://2xb.github.io/usb4research



Writing a new driver: PyUSB

PyUSB allows to write user-space USB drivers

Procedure:

- 1) Look at binary data going over Wireshark
- 2) Try sending and receiving the same data
- 3) Look at Wireshark to understand if one is really doing the same

Beware: PyUSB doesn't integrate your driver into the operating system! Scanner/Printer/Keyboard/Mouse/Camera/Storage... won't be usable with existing applications when writing PyUSB-based drivers

=> For Operating System integration, develop kernel-space driver (in C)





Generic drivers

Our driver should be operating system independent

=> Have the least interaction with OS possible

=> Passthrough USB driver just forwards our 0s and 1s to device





Generic drivers

How to activate/install them:

Linux:

Add a file /etc/udev/rules.d/somename.rules including the following content (vendor and product ids are listed in the table below). SUBSYSTEM=="usb", ATTRS{idVendor}=="...", ATTRS{idProduct}=="...", MODE="0666"

Then, run the following two commands: sudo udevadm control – reload sudo udevadm trigger

Windows:

Use Zadig (https://zadig.akeo.ie/) to install "libusb-win32"





Demo

Notes: Byte-ordering (Endianness) Coloring rules in Wireshark

living.knowledge





General

• USB devices can be used more freely than manufacturers anicipate

Graphical User Interfaces

- Writing GUIs for scientific contexts quite easy nowadays
- Discuss with other users, keep everything simple

Device interaction

- Using existing device libraries with Python possible
- Writing custom drivers requires some puzzle solving, but also works
- If you're interested in preventing memory loss, faultguard may be useful (https://github.com/2xB/faultguard)

https://2xb.github.io/usb4research

living.knowledge

Benedikt Bieringer – DPG-Frühjahrstagung 2023 – 23.03.2023