Build an Xbox Controller Abstraction Layer in Python Using XInput API

Bartosz Konikiewicz · Follow
Published in Level Up Coding
3 min read · Dec 10, 2020

Or: how to achieve simple goal with exaggerated amount of code — Part 1

Photo by Kamil S on Unsplash

I've once thought: input — output. Pressing a gamepad button— action in Python. *Controller mapping in a nutshell*. Conceptually — yes. It is this simple. But, because you use XInput API to interact with the controller, things get complicated. Not

surprisingly. You use a system DLL, presumably via ctypes library, which means different data types than Python native ones. Sure, Microsoft API provides some convenient functions, and you will use two of them in this tutorial, but you use C pointers to interact with them. The list goes on.

On the other side of the coin, it's just ctypes code that may look nasty. Luckily you abstract it early on, then start working on the actual problem. It's pretty low level stuff, so you need to perform some tedious tasks, but it's manageable. And really fun when you see that your PC has received the input and is ready to process it. Hopefully you'd like it, too!

Implement the XInput API

The first thing you got to do is to implement classes required by XInput API. You will define three classes:

- `XInputGamepad` — the actual state of the controller.
- `XInputState` — the alleged state of the controller.
- `XInputVibration` — speed of motors.

Let's break them down a bit.

XInputGamepad

A *de facto* controller state class:

```python
import ctypes
import ctypes.wintypes


class XInputGamepad(ctypes.Structure):
    _fields_ = [
        ('wButtons', ctypes.wintypes.WORD),
        ('bLeftTrigger', ctypes.wintypes.BYTE),
        ('bRightTrigger', ctypes.wintypes.BYTE),
        ('sThumbLX', ctypes.wintypes.SHORT),
        ('sThumbLY', ctypes.wintypes.SHORT),
        ('sThumbRX', ctypes.wintypes.SHORT),
        ('sThumbRY', ctypes.wintypes.SHORT)
    ]
```

xinput_**gamepad**.py hosted with ❤️ by GitHub view raw

- `XInputGamepad` class inherits from a `Structure` class. It means that our class in Python should be treated as a `struct` .
- Define `_fields_` variable. It contains fields of the structure. As you can see, they are tuples — variable name along with their type. They are all integer.

As stated in the docs, it's the direct mapping of the following C `struct` :

```
struct _XINPUT_GAMEPAD {
  WORD  wButtons;
  BYTE  bLeftTrigger;
  BYTE  bRightTrigger;
  SHORT sThumbLX;
  SHORT sThumbLY;
  SHORT sThumbRX;
  SHORT sThumbRY;
}
```

I think the easiest way to grasp the `struct` concept is to perceive it as a class with no methods — just variables. If you were to define it as a Python class, it would look like this:

```
class XInputGamepad:
    buttons: int
    left_trigger: int
    right_trigger: int
    thumb_l_x: int
```

```
    thumb_l_y: int
    thumb_r_x: int
    thumb_r_y: int
```

It won't be this obscure all the time. Soon you will write some abstraction layer that will make interacting with XInput API more sane.

XInputState

A class that represents the controller state. It's basically `XInputGamepad` , but with an field indicating that the state has changed:

```
class XInputState(ctypes.Structure):
    _fields_ = [
        ('dwPacketNumber', ctypes.wintypes.DWORD),
        ('Gamepad', XInputGamepad),
    ]
```

- Basically the same as the previous class, the only difference being that the `Gamepad` field is is a struct you've defined before.
- `dwPacketNumber` indicates that the state may have changed.

XInputVibrations

Vibrations, as the name suggests:

```
class XInputVibration(ctypes.Structure):
    _fields_ = [
        ('wLeftMotorSpeed', ctypes.wintypes.WORD),
        ('wRightMotorSpeed', ctypes.wintypes.WORD)
    ]
```

- Xbox gamepads have two motors — left and right. They work independently.
- Each variable specify which motor to use. They represent the vibration strength.

Let's put the aforementioned classes to good use.

Fetch the Input

You'll ditch the code in a second, but it's still good to see if it works. Don't create any functions just yet. Simply check it in a `__main__` block:

```
import time
import os


if __name__ == '__main__':
    api = ctypes.windll.xinput1_4
    state = XInputState()
    gamepad_number = 0

    while True:
        api.XInputGetState(
            ctypes.wintypes.WORD(gamepad_number),
            ctypes.pointer(state)
        )
        print(state.dwPacketNumber)
        time.sleep(0.5)
        os.system('cls')
```

xinput.py hosted with ❤ by GitHub view raw

- Import the XInput `api` using `ctypes`.
- Define the controller `state`. It's just needed by a function that you will use in a moment. You won't read the state just yet.
- If several controllers are connected, specify which one to use, from range 0–3. If only one controller is connected, it should default to `0`.
- Get the state in a loop, providing `gamepad_number` and a pointer to the `state`.
- `print` a `dwPacketNumber` variable indicating if the controller state has changed.
- `sleep`, then clear (`cls`) the screen, so that it's readable.

Having that, you can connect your Xbox Controller and start the script. First, let's save it as `xinput.py`, so it can be run with the following command:

```
python xinput.py
```

You will see a random number. If it changes when you manipulate the controller, everything's sound. You can remove the `__main__` block content and move to the next part of the tutorial.

Final words (for now)

Thanks for reading!

You can find the full code on my GitHub.

Python Xbox Xinput Gamepad Mapper

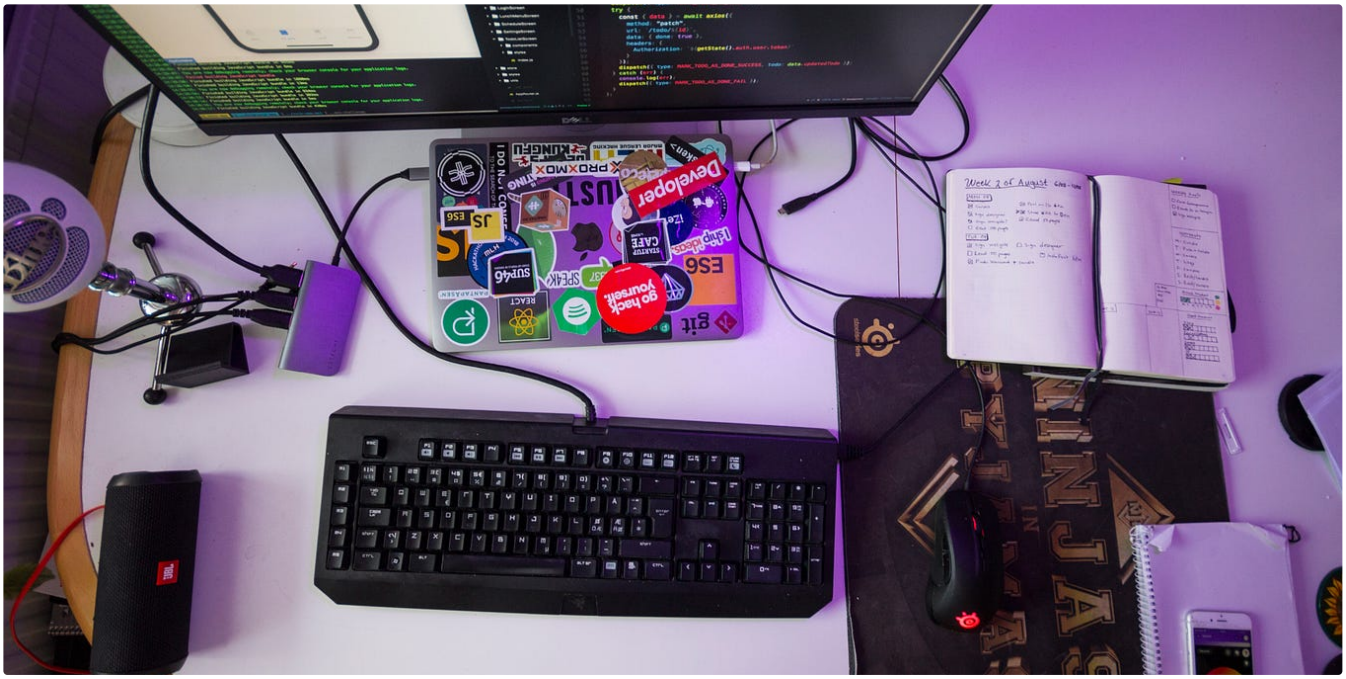
Follow

Written by Bartosz Konikiewicz

24 Followers · Writer for Level Up Coding

This is not the greatest bio in the world. This is just a tribute.

More from Bartosz Konikiewicz and Level Up Coding



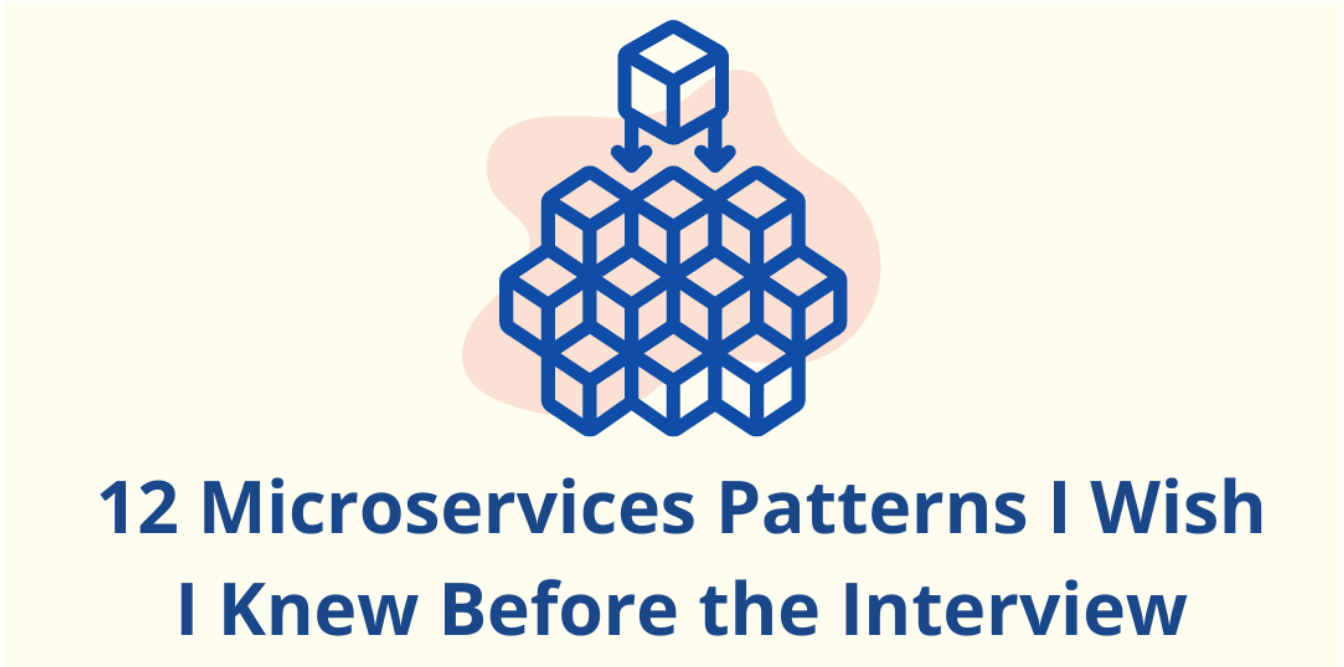
Bartosz Konikiewicz in Level Up Coding

Create a Controlled Radio Group in React, Formik, Material UI, and TypeScript

Or: how not to lose five hours at your 9 to 5

4 min read · Apr 25, 2021

113 1



Arslan Ahmad in Level Up Coding

12 Microservices Patterns I Wish I Knew Before the System Design Interview

Mastering the Art of Scalable and Resilient Systems with Essential Microservices Design Patterns

· 13 min read · May 16

1K 9

Attila Vágó in Level Up Coding

Martin Fowler Was Right: Microservices Suck*

A pragmatic view on Amazon's unexpected stance on serverless microservice architecture.

· 5 min read · May 9

2K 36

Bartosz Konikiewicz in Level Up Coding

Handle Registration in FastAPI and Tortoise ORM

And shed a tear or two in the process—Part 3

4 min read · Dec 9, 2020

 11  1 

See all from Bartosz Konikiewicz

See all from Level Up Coding

Recommended from Medium

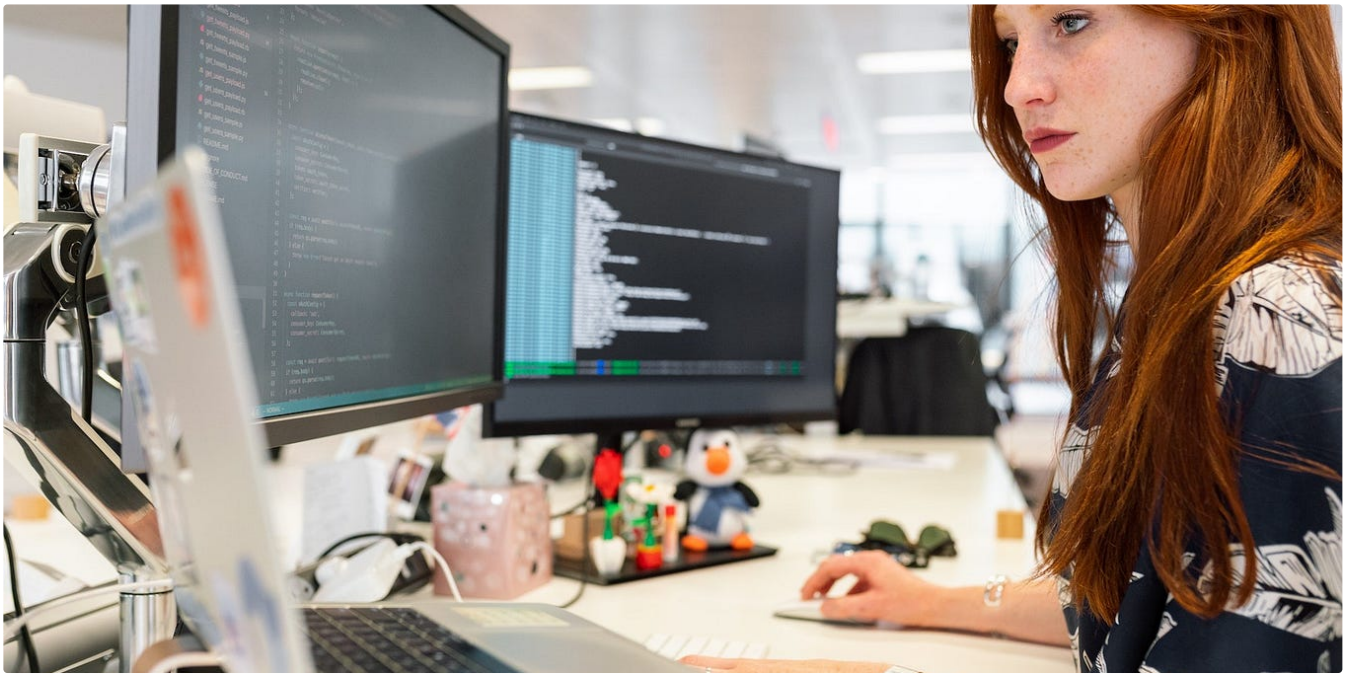
Aysel Aydin

An Introduction to Python FastAPI & Swagger UI

In this Python tutorial, you will learn about FastAPI that a Web framework for developing RESTful APIs in Python.

3 min read · Jan 7

373



The Coding Diaries in The Coding Diaries

Why Experienced Programmers Fail Coding Interviews

A friend of mine recently joined a FAANG company as an engineering manager, and found themselves in the position of recruiting for...

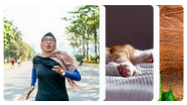
✦ · 5 min read · Nov 2, 2022

2.9K 64

Lists

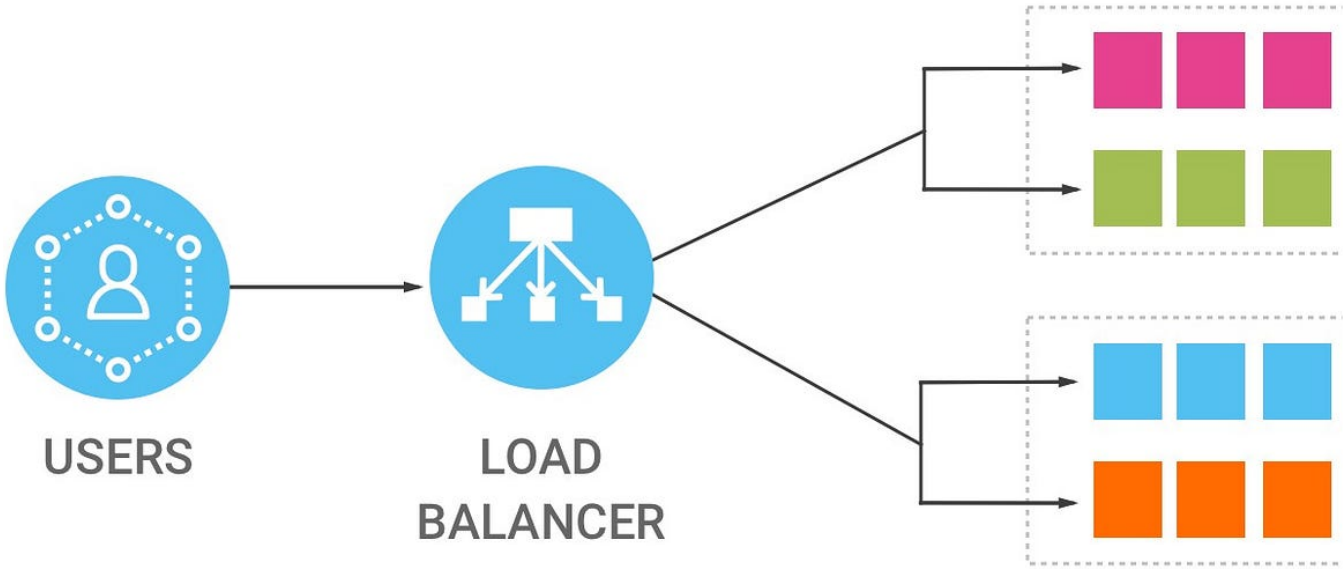
Staff Picks
320 stories · 81 saves

Stories to Help You Level-Up at Work
19 stories · 40 saves



Self-Improvement 101
20 stories · 87 saves

Productivity 101
20 stories · 88 saves



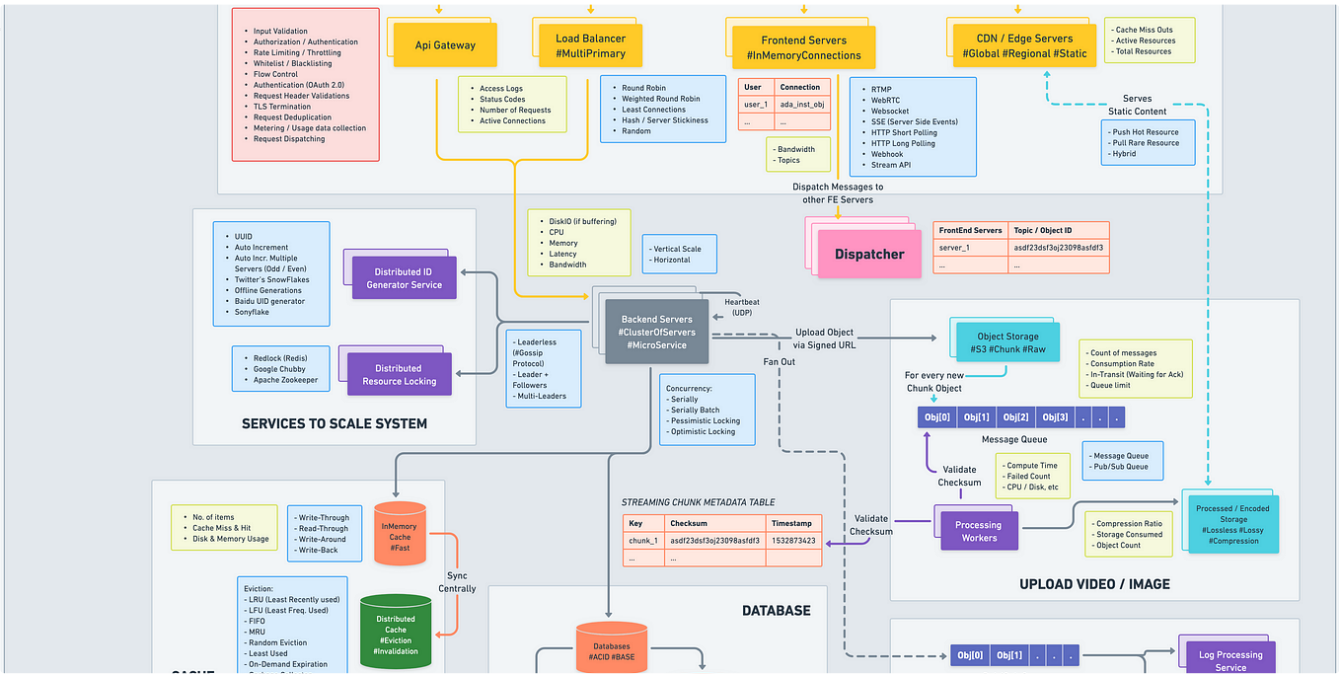
Daniel Pollak in anecdotes engineering

FastAPI Scale with non-asyncio requests handling

FastAPI is a web server framework that is particularly well-suited for use with asyncio-compatible flows. However, there are still many...

6 min read · Dec 25, 2022

186 3



Love Sharma in Dev Genius

System Design Blueprint: The Ultimate Guide

Developing a robust, scalable, and efficient system can be daunting. However, understanding the key concepts and components can make the...

· 9 min read · Apr 20

1.1K 11

Jacob Marks, Ph.D. in Towards Data Science

How I Turned My Company's Docs into a Searchable Database with OpenAI

And how you can do the same with your docs

15 min read · Apr 25

2.9K 39

Hussein Nasser

How to Become a Good Backend Engineer (Fundamentals)

I have been a backend engineer for over 18 years and I witnessed technologies come and go but one thing always remain constant; The first...

✦ · 11 min read · Dec 3, 2022

5.3K 40

See more recommendations