

Baraniecki Karol Byczko Maciej	Prowadzący: Dr inż. Dominik Żelazny	Numer ćwiczenia laboratoria 12
PT 16:30 TP	Temat ćwiczenia: Obsługa kamery USB	Ocena:
Grupa: D	Data wykonania: 10 grudnia 2021	

1 Zagadnienia do opracowania

1. Zasada działania skanera płaskiego - rodzaj skanera obrazu.

Pokrywa skanera jest ruchoma, pod nią znajduje się szyba. Aby dokonać skanowania, kładzie się skanowaną kartkę na szybę skanowaną stroną do spodu. W trakcie pracy skanera pod szybą przesuwa się zespół lampa-lustro.

2. parametry skanerów płaskich:

- Rozdzielcość optyczna - (ang. optical resolution), wyrażona w dpi, podstawowy i najważniejszy parametr skanera, ponieważ im większa jest rozdzielcość układu optycznego, tym lepsza jest jakość cyfrowego odpowiednika zeskanowanego obrazu. Rozdzielcość optyczna określa rzeczywistą, sprzętową zdolność skanera do odzwierciedlenia obrazu, w przeciwieństwie do rozdzielcości interpolowanej.
 - Rozdzielcość interpolowana - (ang. optical resolution), wyrażona w dpi, często podawana przez producentów skanerów w celu przyciągnięcia uwagi kupującego. Jest to zazwyczaj niebotycznie wysoka rozdzielcość obrazu, nie ma jednak nic wspólnego z jakością układu optycznego skanera, bo uzyskiwana jest matematycznie (na gotowym obrazie) poprzez porównywanie kolorów leżących obok siebie pikseli, obliczeniu średniej wartości koloru i wstawieniu dodatkowych, wirtualnych pikseli.
 - Głębia kolorów - (ang. color depth), standardem we współczesnych skanerach jest możliwość odzwierciedlenia co najmniej 24-bitowej palety barw, co oznacza iż pojedynczy piksel obrazu może przyjąć jeden kolor z 16.8 miliona odcieni (tyle rozróżnia ludzkie oko).
 - Gęstość optyczna - (ang. optical density) - wyrażona w jednostce D (Density) określa zdolność skanera do rozróżniania odcieni barw. Skanery profesjonalne powinny się charakteryzować gęstością powyżej 3D, zaś skanery domowe i półprofesjonalne nie mniejszą niż 2.5D.
3. Twain - nazwa standardu komunikacji między urządzeniami przetwarzającymi obrazy - skanerami, aparatami cyfrowymi a programami graficznymi, opracowany dla Windows i systemów Apple Macintosh, a także Linux/Unix.
 4. Sane - Scanner Access Now Easy (SANE), wieloplatformowy interfejs programowania aplikacji API, który umożliwia dostęp do większości skanerów optycznych. Interfejs Sane jest rozwijany na zasadach FLOSS, co oznacza, że każdy może pomóc w jego tworzeniu.

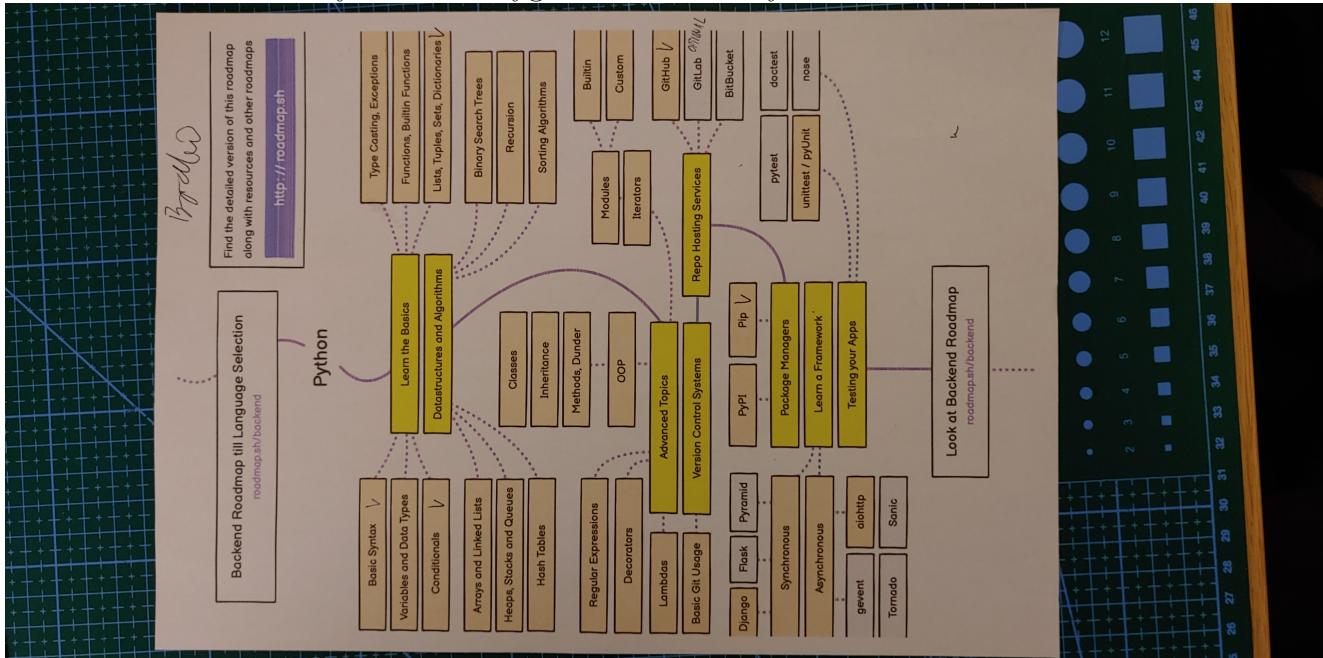
SANE różni się od TWAIN tym, że rozdzielone w nim zostały interfejs użytkownika (front-end) i sterowniki sprzętowe (back-end). TWAIN pełni obie te funkcje, podczas gdy Sane oprócz komunikacji ze sprzętem, zapewnia jedynie graficzny interfejs ustawień skanowania (np. rozdzielcość, obszar, ustawienia kolorów).

Taki podział ułatwia sieciową obsługę skanowania, na komputerze wyposażonym w skaner, demon Sane jedynie obsługuje zapytanie wysyłane z innych komputerów, ułatwia to pisanie aplikacji oraz zmniejsza ryzyko wystąpienia konfliktów.

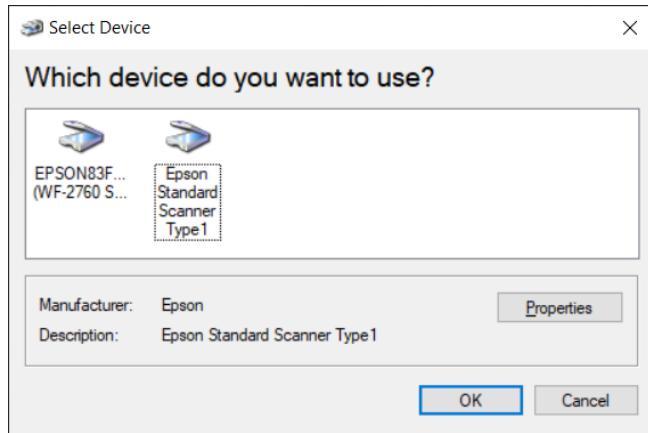
5. Biblioteka WIA lub WIA 2 - Windows Image Acquisition (WIA) to platforma do akwizycji obrazów w rodzinie systemów operacyjnych Windows, począwszy od Windows Millennium Edition (Windows Me) i Windows XP. Platforma WIA umożliwia aplikacjom do przetwarzania obrazu/grafiki interakcję ze sprzętem do przetwarzania obrazu i standaryzuje interakcję między różnymi aplikacjami i skanerami. Dzięki temu różne aplikacje mogą rozmawiać i współpracować z różnymi skanerami bez konieczności dostosowywania przez autorów aplikacji i producentów skanerów swoich aplikacji lub sterowników do każdej kombinacji aplikacji-urządzenie.

2 Zadania do wykonania

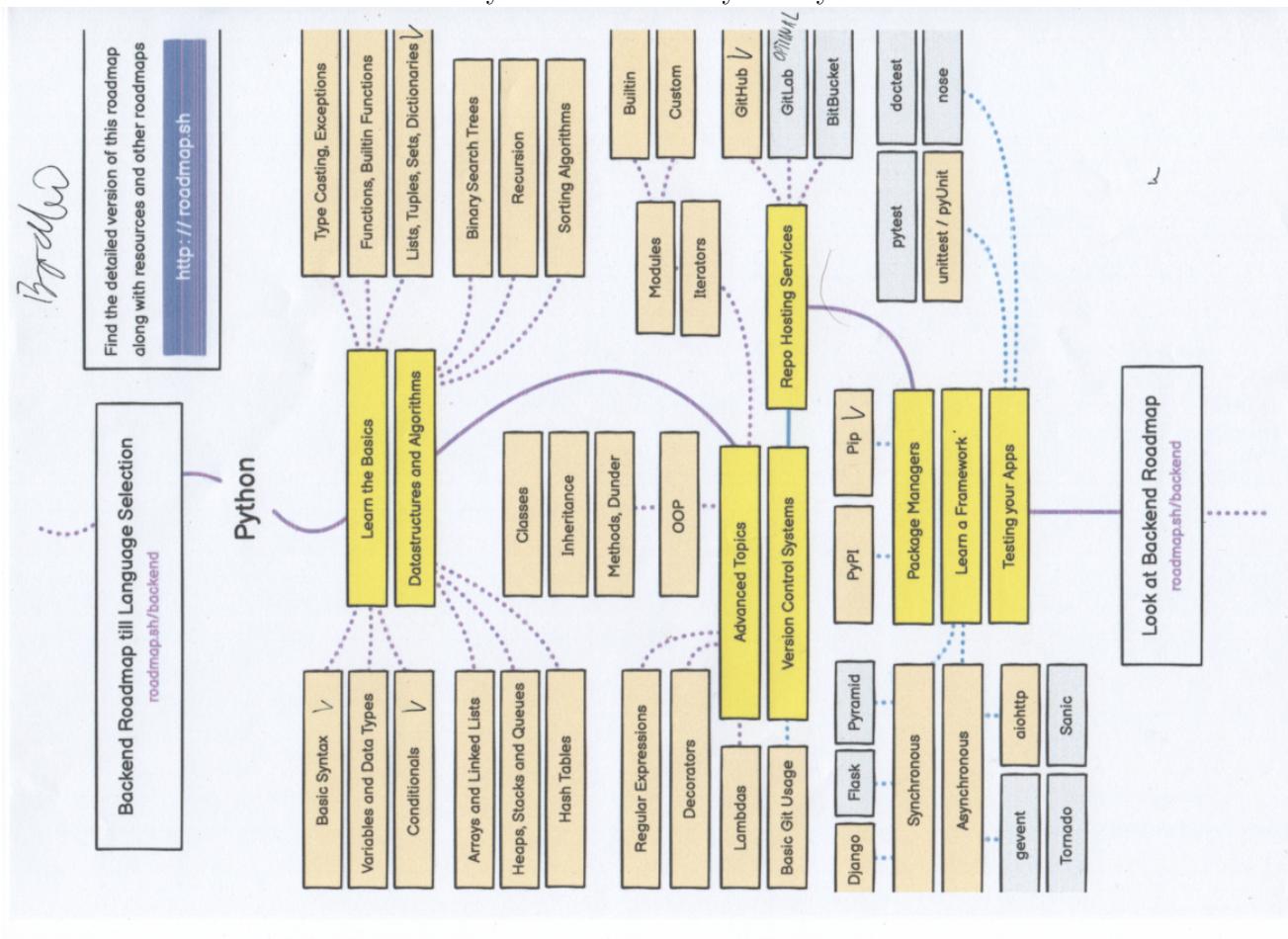
Rysunek 1: Oryginalna kartka używana w skanerze



1. Sprawdź czy skaner działa poprawnie - Uruchomił się i poprawnie wykonał skan.
2. Napisać program wykonujący skanowanie przy pomocy skanera płaskiego (WIA lub Twain - po ustaleniu z prowadzącym). Możliwości programu obejmują:
 - skanowanie z wykorzystaniem UI
UI występowało tylko podczas pracy z Windowsem okienko wyglądało następująco:



Rysunek 2: Skan wynikowy



Kod za pomocą którego został wykonany skan.

```

1 import win32com.client
2 import os
3
4 WIA_COM = "WIA.CommonDialog"
5
6 WIA_IMG_FORMAT_PNG = "{B96B3CAF-0728-11D3-9D7B-0000F81EF32E}"
7
8 WIA_COMMAND_TAKE_PICTURE = "{AF933CAC-ACAD-11D2-A093-00C04F72DC3C}"
9
10
11 def acquire_image_wia():

```

```

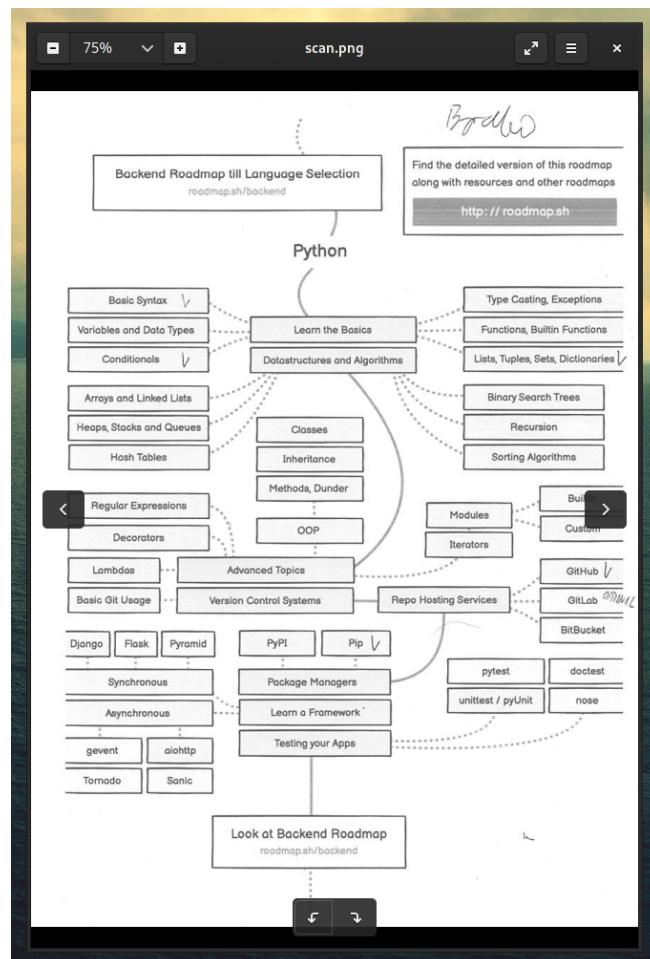
12     wia = win32com.client.Dispatch(WIA_COM) # wia is a CommonDialog
13     dev = wia.ShowSelectDevice() # display connected scanners
14
15     for i, item in enumerate(dev.Items):
16         if i + 1 == dev.Items.Count:
17             image = item.Transfer(WIA_IMG_FORMAT_PNG)
18             break
19
20     fname = 'wia-test.png'
21     if os.path.exists(fname):
22         os.remove(fname)
23     image.SaveFile(fname)
24     if fname.split('.')[ -1] == "png":
25         os.startfile(fname)
26
27
28 if __name__ == "__main__":
29     acquire_image_wia()

```

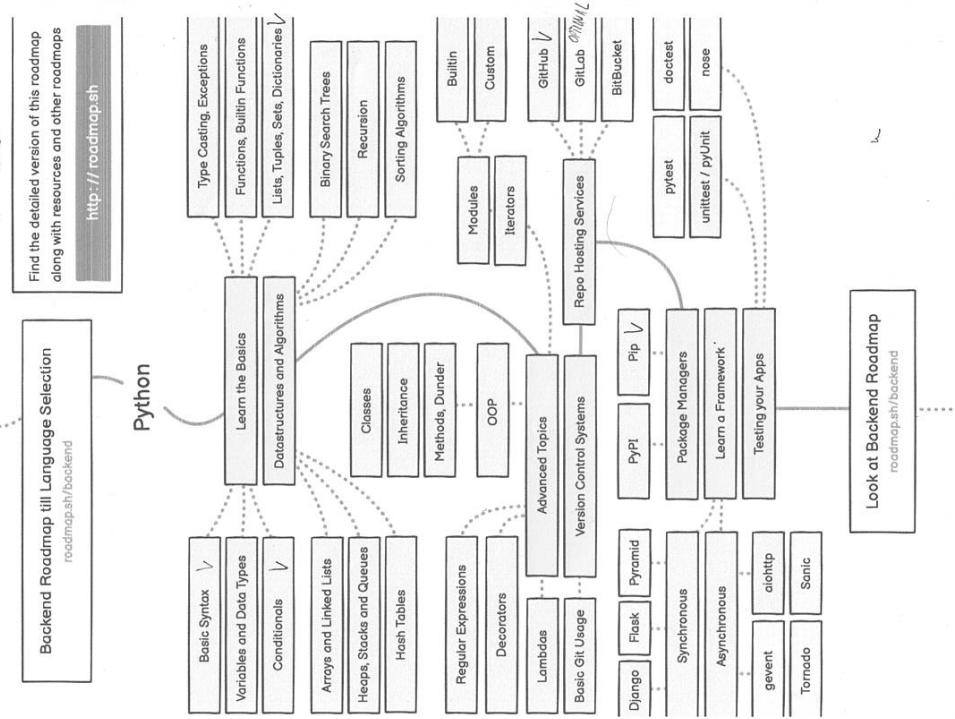
- skanowanie bez wykorzystania UI

Tutaj pozostałe części zadania wykonywaliśmy na systemie operacyjnym Linux gdyż API było bardziej przejrzyste niż to z którego próbowaliśmy korzystać

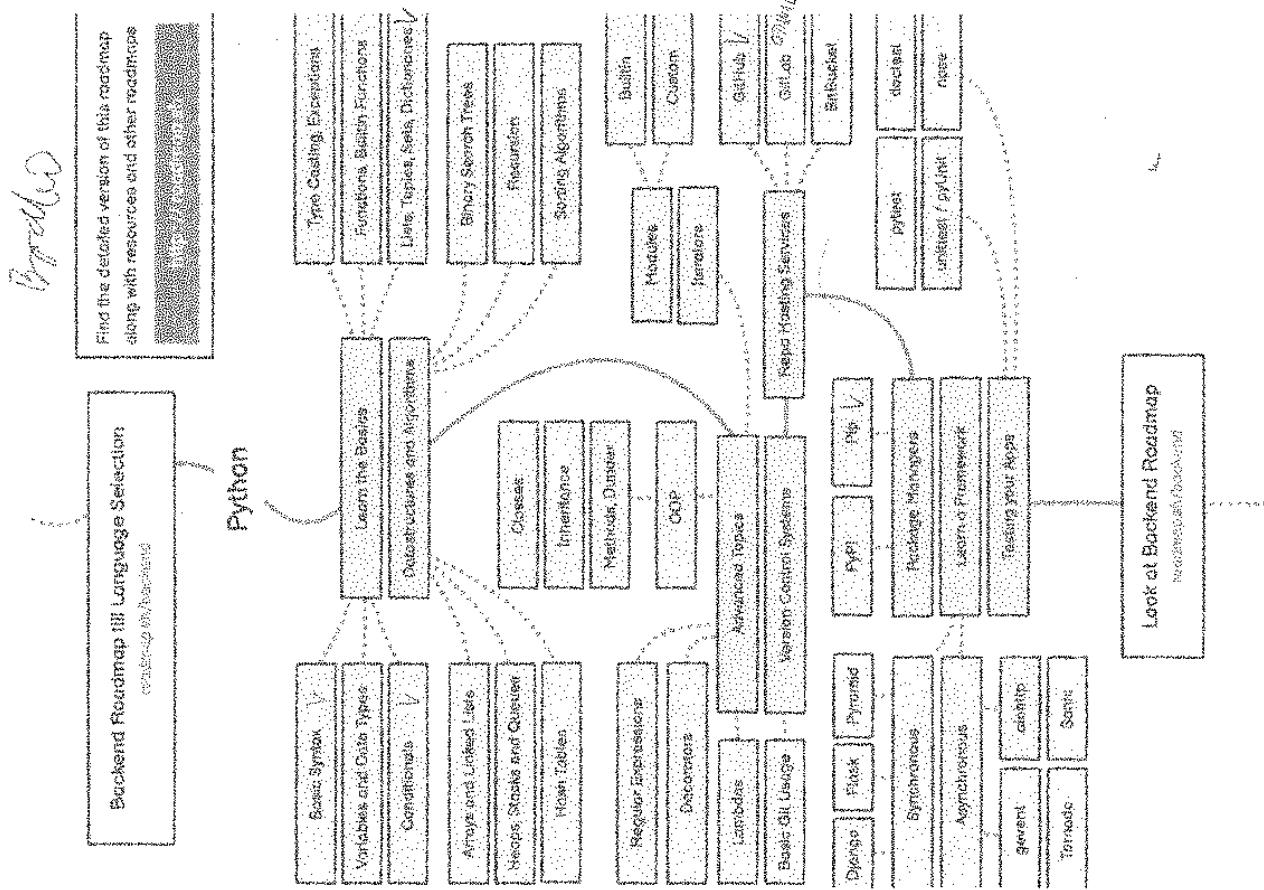
- wyświetlanie uzyskanego obrazu



- zmiana trybu skanowania (1-bitowy, skala szarości, RGB)



- zmiana rozdzielczości skanera



3. Rozszerzyć działanie programu:

- zapis skanowanych obrazów do plików graficznych - pliki są automatyczne zapisywane wraz z datą oraz godziną wykonania.

4. kod programu wykorzystanym w systemie Linux

```
1 #!/usr/bin/env python3
2 import sane
3 import sys
4 import os
5 import datetime
6
7 device_address = 'escl:http://192.168.1.6:443'
8 sane.init()
9 device = sane.open(device_address)
10
11 supported_resolutions = next(filter(lambda opt: opt[1] == 'resolution',
12                                     device.get_options()))[8]
12 supported_modes = next(filter(lambda opt: opt[1] == 'mode', device.
13                               get_options()))[8]
13
14 def usage():
15     print(f"Usage: {sys.argv[0]} [mode] [resolution]")
16     print(f"    supported modes: {', '.join(supported_modes)}")
17     print(f"    supported resolutions: {', '.join(map(str,
18                                                    supported_resolutions))}")
18     sys.exit(1)
19
20 if len(sys.argv) != 3:
21     usage()
22
23 mode = sys.argv[1]
24 resolution = int(sys.argv[2])
25
26 if mode not in supported_modes or resolution not in supported_resolutions:
27     usage()
28
29 print("All scanner options:")
30 for opt in device.get_options():
31     print(opt)
32
33 device.mode = mode
34 device.resolution = resolution
35
36 device.start()
37 image = device.snap()
38
39 filename = f"Skan {datetime.datetime.now()}.png"
40 image.save(filename)
41
42 # Eye of Gnome, śdomylna aplikacja obrazów w Gnomie
43 os.execv("/usr/bin/eog", ["/usr/bin/eog", filename])
```

3 Wnioski

Windows ma bardzo okropne API dlatego w połowie wykonywania zadania przenieśliśmy się na system operacyjny Linux. Znaleźliśmy w nim bibliotekę "Sane" która łączy się bezprzewodowo ze skanerem i może wykonywać wszystkie operacje wbudowane w skaner (wyświetlamy je po wpisaniu wybranych parametrów)