Dawid Dopart 248895
Błażej Ułanowicz 249452
GRUPA C (CZ TN 17:15-20:15)

**Prowadzący:**
dr inż. Dominik Żelazny

# Urządzenia peryferyjne – laboratorium
# Ćwiczenie 9 – obsługa skanera płaskiego [WIA]

## 1. Cel ćwiczenia

Celem ćwiczenia było utworzenie aplikacji umożliwiającej obsługę skanera płaskiego. W ramach ćwiczenia należało zaimplementować poniższe funkcjonalności:

- skanowanie,
- podgląd gotowego skanu,
- zapis do pliku graficznego,
- modyfikacja parametrów skanowania (jasność, kontrast, tryb skanowania – kolor).

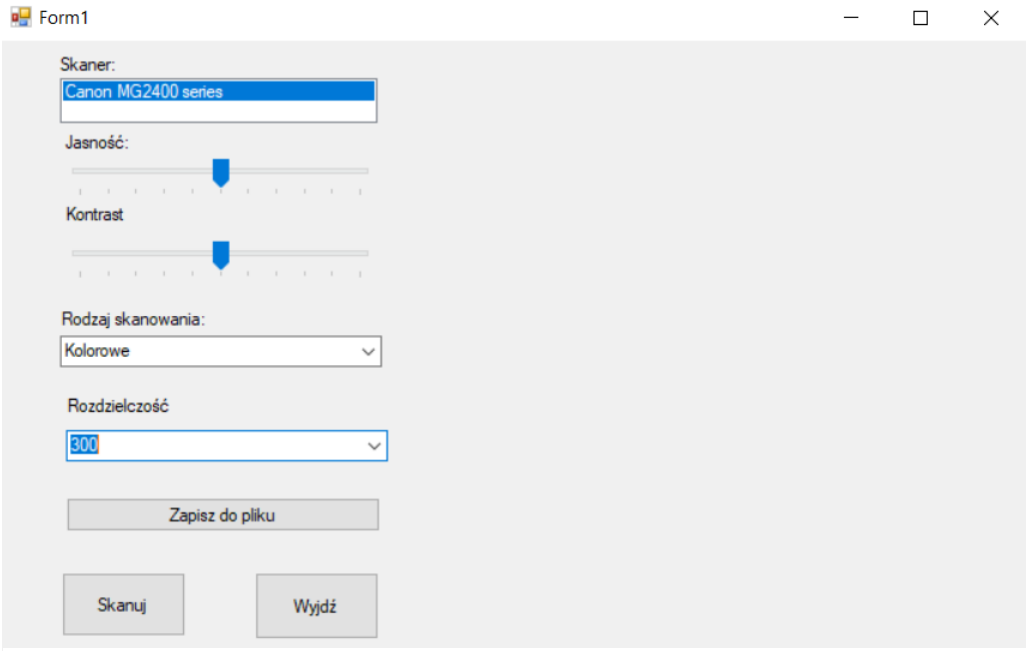
## 2. Wykorzystane technologie i środowisko pracy

Do zaimplementowania funkcjonalności użyto języka C# oraz skorzystano z biblioteki Windows Image Acquisition (WIA) do operowania skanerem.
Do utworzenia interfejsu użytkownika skorzystano z Windows Forms.

Gotowa aplikacje była testowana przy użyciu komputera z Windows 10 oraz drukarki wielofunkcyjnej Canon PIXMA MG2400.

## 3. Opis działania programu

W ramach realizacji zadania utworzony został proste graficzne UI, dające możliwość wybrania wszystkich zaimplementowanych opcji.




*Zrzut ekranu 1: Okno aplikacji po uruchomieniu*

Korzystając z obiektu klasy *DeviceManager* z frameworku WIA (służącego do operowania podłączonymi urządzeniami), filtrujemy dostępne urządzenia, których typ oznaczony jest jako *ScannerDeviceType*, czyli posiadają możliwość skanowania – w taki sposób dostajemy listę skanerów oraz drukarek wielofunkcyjnych.

```
private void Form1_Load(object sender, EventArgs e)
{
    for(int iii = 1; iii <= dm.DeviceInfos.Count; iii++)
    {
        if (dm.DeviceInfos[iii].Type != WiaDeviceType.ScannerDeviceType)
            continue;

        scannerListBox.Items.Add(dm.DeviceInfos[iii].Properties["Name"].get_Value());
        scannerDevices.Add(dm.DeviceInfos[iii]);
        scannerListBox.SelectedIndex = iii-1;
    }
}
```

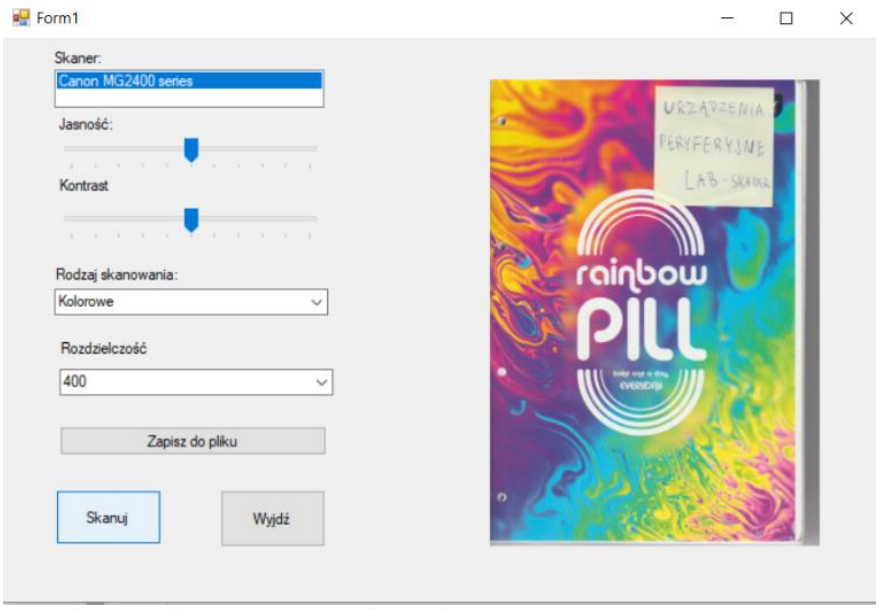
*Kod 1: Pobieranie urządzeń typu skaner*

Główna funkcjonalność programu zawarta jest w kodzie kliknięcia na przycisk „Skanuj”, który uruchamia proces skanowania. Na początku program próbuje nawiązać połączenie z wybranym urządzeniem korzystając z metody *Connect()*, która zwraca instancję urządzenia. Kolejnym krokiem jest zmiana ustawień skanera na wybrane wcześniej przez użytkownika (omówienie kodu w dalszej części sprawozdania). Następnie rozpoczyna się proces skanowania, który uruchamiany jest przy pomocy metody *Transfer* na instancji urządzenia (wybranym parametrem jest format wyjściowego obrazu – w tym przypadku *WIA.FormatID.wiaFormatJPEG*, czyli format *.jpeg*). W taki sposób utworzony zostaje obiekt typu *ImageFile*, zawierający dane zeskanowanego obrazu.
Dalsza część funkcji to kod odpowiadający za konwersję klasy *ImageFile* z frameworku WIA do akceptowalnej przez Windows Forms klasy *System.Drawing.Image*. Należało tutaj pobrać dane binarne zeskanowanego obrazu, przetworzyć je na strumień, a następnie przekazać je do konstruktora klasy *Image*. W ten sposób udało nam się utworzyć podgląd obrazu bez wcześniejszego zapisywania go.
Warto dodać, że opcja zapisywania została przeniesiona do innej funkcji, gdzie po naciśnięciu przycisku „Zapisz do pliku”, użytkownik wybiera ścieżkę, a skan zostaje pod nią zapisany. Korzystamy tutaj z gotowej klasy SaveFileDialog (służącej do wyświetlania ekranu wyboru pliku) oraz metody *SaveFile* klasy *ImageFile*.

```
private void scanBox_Click(object sender, EventArgs e)
{
    var selectedDevice = scannerDevices.ElementAt(scannerListBox.SelectedIndex).Connect();
    var scanner = selectedDevice.Items[1];
    AdjustScannerSettings(scanner, dpi, 0, 0, brightnessLevel, contrastLevel, colorSettting);
    scannedFile = (ImageFile)scanner.Transfer(WIA.FormatID.wiaFormatJPEG);
    var imageBytes = (byte[])scannedFile.FileData.get_BinaryData();
    var ms = new MemoryStream(imageBytes);
    scanImage.Image = Image.FromStream(ms);
}
```

*Kod 2: Skanowanie obrazu*

Efekt działania skanowania można zobaczyć na poniższym zrzucie ekranu.




*Zrzut ekranu 2: Okno programu po udanym skanowaniu*

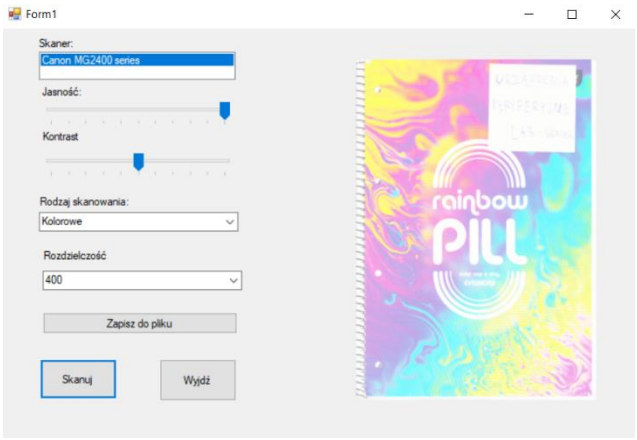
Do modyfikacji ustawień skanowania posłużyła funkcja *SetWIAProperty()* przyjmująca trzy parametry: właściwości, nazwę opcji oraz wartość jaka ma zostać ustawiona.
Nazwa właściwości to tak naprawdę odpowiednie stringi reprezentujące 4-cyfrowe kody np. „6164” – zmiana koloru skanu, „6154” – procent jasności.

```
private static void SetWIAProperty(IProperties properties, object propName, object propValue)
{
    Property prop = properties.get_Item(ref propName);
    prop.set_Value(ref propValue);
}
```

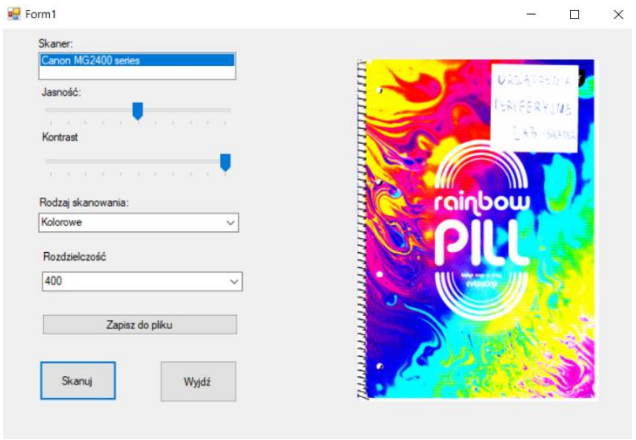
*Kod 3: Modyfikacja ustawień skanowania*

Powyższa funkcja została zagregowana do większej funkcji *AdjustScannerSettings()* która przyjmuje parametry takie jak: urządzenie skanujące, wartość dpi, współrzędne rozpoczęcia skanu, procentowe poziomy jasności i kontrastu, a także wybrany tryb skanowania.
Pobierane parametry wykorzystywane są w kolejnych wywołaniach funkcji *SetWIAProperty()* do ustawiania odpowiednich wartości właściwości. Dodatkowo w celu zwiększenia czytelności kodu zamieniono liczbowe kody na stałe znakowe co umożliwiło zrozumienie działania kodu bez potrzeby zaglądania do dokumentacji i sprawdzania znaczenia każdego kodu właściwości.

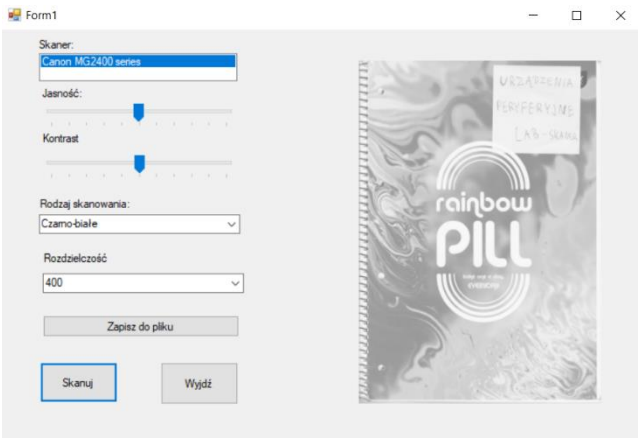
Przykładowe efekty modyfikacji opcji skanowania:



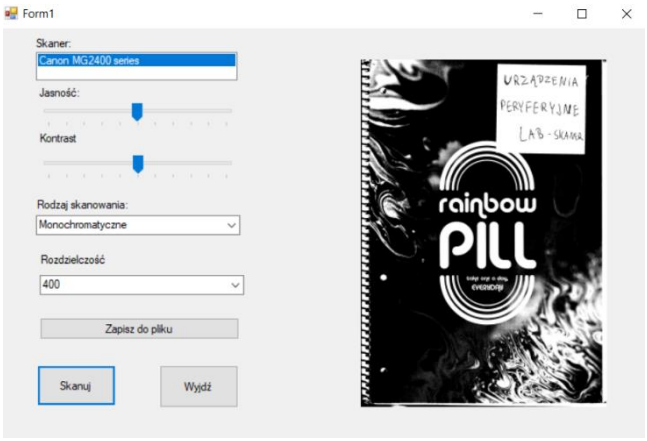

*Zrzut ekranu 3: Modyfikacja jasności*




*Zrzut ekranu 4: Modyfikacja kontrastu*




*Zrzut ekranu 5: Skan w odcieniach szarości*




*Zrzut ekranu 6: Skan czarno-biały*

## 4. Wnioski

Przygotowana aplikacja spełnia wszystkie cele laboratorium oraz działa poprawnie. Dzięki skorzystaniu z frameworku WIA oraz dokładniejszemu zapoznaniu się z jego dokumentacją, udało się podłączyć skaner oraz zmienić jego domyślne ustawienia. Problemy, z jakimi się napotkaliśmy, były głównie związane z odpowiednią obsługą *PictureBox* z Windows Forms, a dokładnie wypełnienia go obrazem bez wcześniejszego jego zapisywania. Ostatecznie wypracowane zostało rozwiązanie zaprezentowane we wcześniejszym punkcie.