Wrocław, 9 grudnia 2019r.

Mikołaj Baran, 241128
Jakub Aniszewski, 241133

prowadzący: Dominik Żelazny

Laboratorium Urządzeń Peryferyjnych

Ćwiczenie 12 - Obsługa kamery USB

1 Cel ćwiczenia

1. Korzystając z przykładowej aplikacji stwierdzić obecność i poprawność kamery podłączonej do portu USB komputera.
2. Wylistuj urządzenia typu cap (kamery) i stwórz interfejs umożliwiający wybór po nazwie urządzenia (drivera) z którym chcesz się połączyć.
3. Połącz się z wybranym urządzeniem i za pomocą odpowiednich komunikatów łączących się z driverami kamery - skonfiguruj ją.
   - za pomocą programu powinno dać się zmieniać opcje kamery (rozdzielczość obrazu, nasycenie, kontrast, ew. zoom, sterowanie kamera etc.)
   - zapisz obraz z kamery w dowolnym formacie (wskazany JPG)
   - zapisz obraz z kamery w postaci filmu AVI
4. Rozbuduj program o:
   - zmień tak program z zadania 3 aby generował stronę html z odświeżanym automatycznie obrazem z kamery
   - dodaj opcje która w przypadku gdy kamera potrzebuje swoich własnych sterowników automatycznie po włączeniu programu instaluje je; po poznaniu sterowników kamery należy znaleźć plik inf, które zostanie odpowiednio uruchomiony przez program (ShellExecute)
   - stwórz prosty detektor ruchu - poprzez analizę obrazu z kamery w czasie rzeczywistym (wystarczy sprawdzać zmiany koloru kilku punktów (pikseli), ćwiczenie można rozwinąc o najprostsze algorytmy wykrywające krawędzie etc.)

2 Wstęp

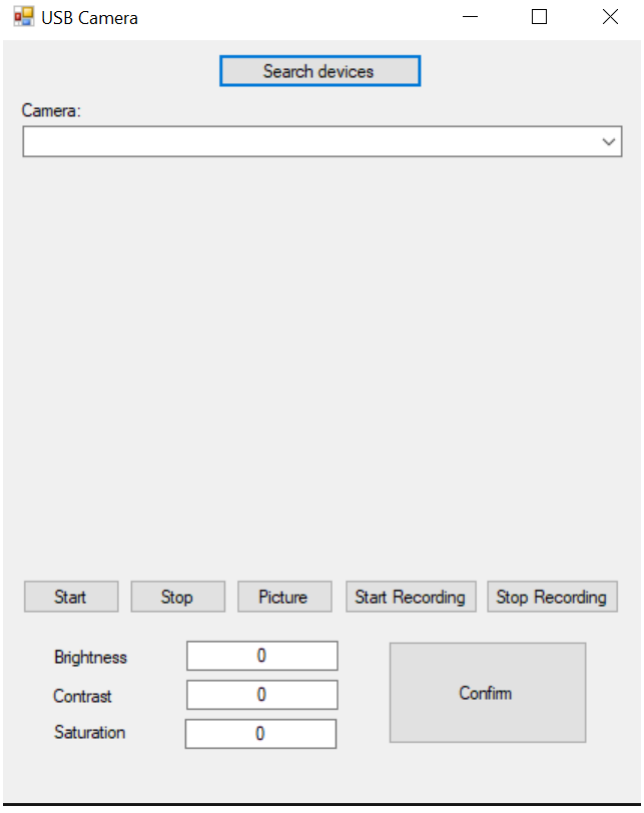
Kamera cyfrowa jest urządzeniem, które realizuje przekształcenie rejestrowanego obrazu do postaci sygnału cyfrowego. Najczęściej kamera jest podłączana przy pomocy łącza USB, jednak coraz częściej spotyka się kamery wbudowane w monitor lub obudowę laptopa. Przy pomocy obiektywu, filmowany obraz jest rzutowany na matrycę, najczęściej typu **CCD** (Charge coupled devices) lub **CMOS** (Complemetary metal-oxide semiconductor), która jest rozwiązaniem tańszym, oferującym szybszą formę odczytu (można odczytać dowolną liczbę pikseli, w dowolnej kolejności, w matrycy CCD trzeba odczytać całą zawartość), rozdzielczość obrazu 640x480 pikseli oraz 30 klatek na sekundę. Matryca posiada informacje na temat oświetlenia obrazu, nie posiada jednak informacji o kolorach. Do uzyskania kolorowego obrazu używa się filtrów w formie mozaiki w kolorach RGB (red, green, blue – kolejno czerwony, zielony i niebieski). Najczęściej stosowany w urządzeniach cyfrowych do rejestracji obrazu jest filtr Bayera. Uzyskiwany jest efekt „zielonej szachownicy, ponieważ zawiera 50% zielonych, 25% czerwonych i 25% niebieskich filtrów elementarnych , ponieważ ludzkie oko jest najwrażliwsze na zielony kolor.

3 Opis programu

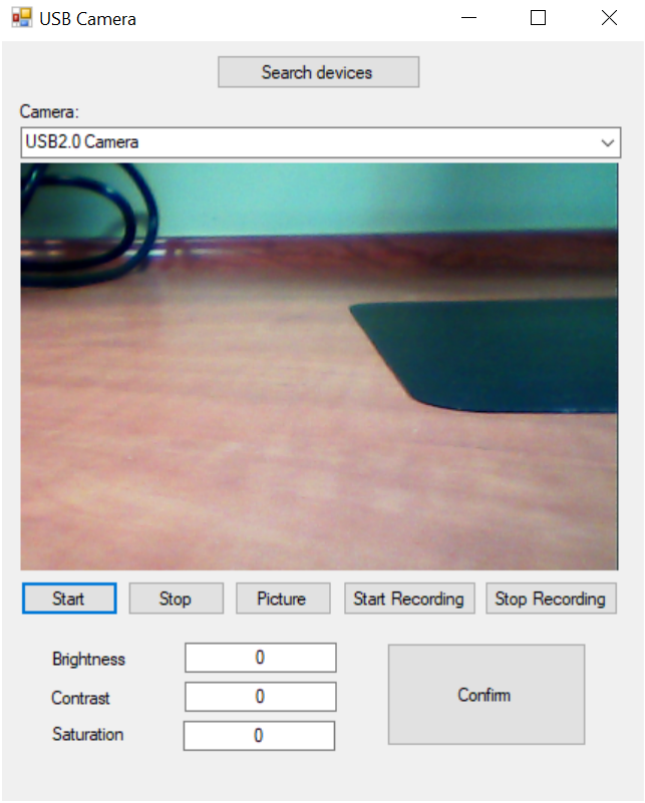
Pierwszym krokiem, przed przystąpieniem do pisania właściwego programu było sprawdzenie poprawności działania kamery USB. Jak widać na poniższym zrzucie ekranu urządzenie zostało poprawnie połączone.

Program był pisany w języku C# w środowisku Visual Studio przy użyciu biblioteki **AForge.NET** zaprojektowanej specjalnie z myślą między innymi o przetwarzaniu obrazów.

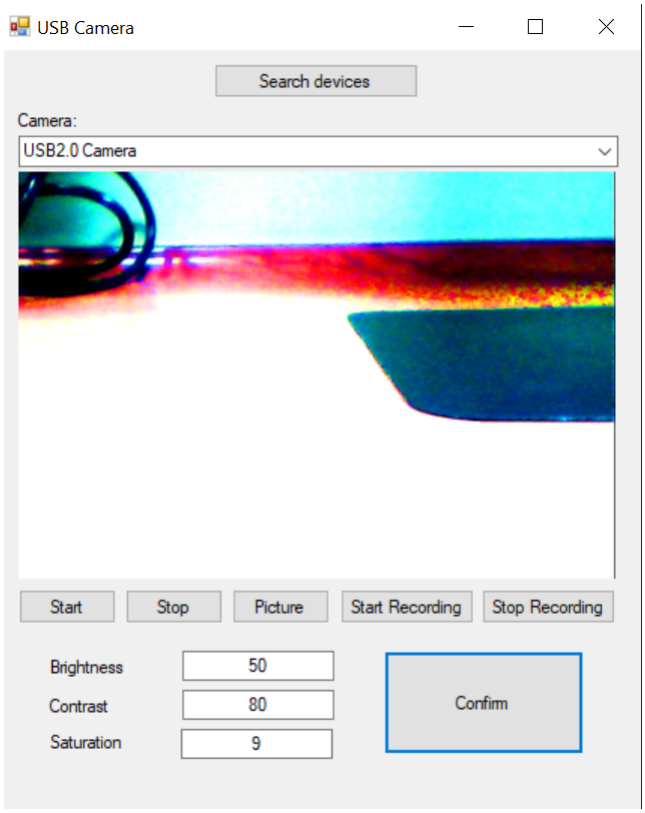
Interfejs programu jest dosyć intuicyjny.



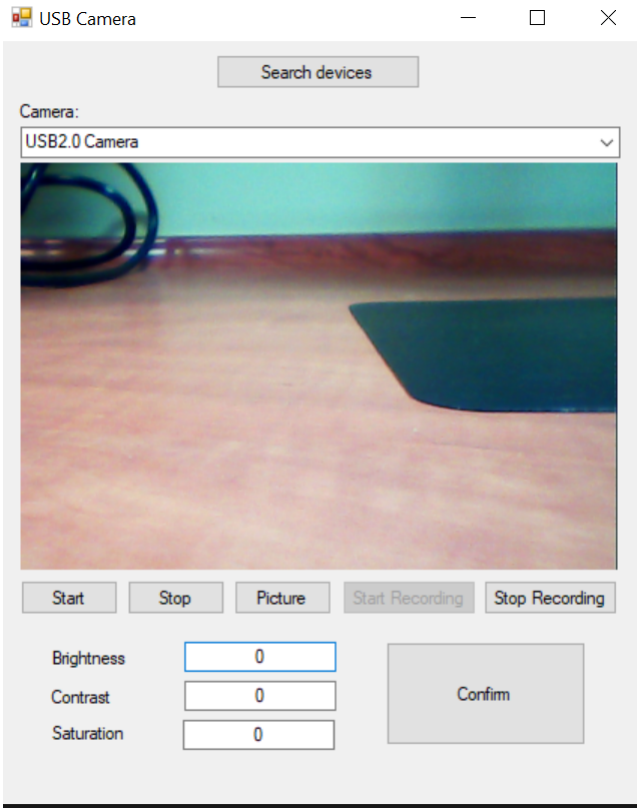
Do wykrywania urządzeń video służy przycisk *Search devices*. Następnie wykryte urządzenia pojawiają się w liście rozwijanej *Camera*.



Przyciski *Start* i *Stop* służą odpowiednio do rozpoczęcia oraz zatrzymania przechwytywania podglądu obrazu z kamery. Przycisk *Picture* zapisuje zrzut ekranu do pliku.



W dolnej sekcji programu możemy manipulować parametrami wyświetlanego obrazu. Są to jasność (*Brightness*), kontrast (*Contast*) i nasycenie (*Saturation*).



Aby rozpocząć nagrywanie obrazu do pliku video należy nacisnąć przycisk *Start Recording*. Po rozpoczęciu nagrywania przycisk ten staje się nieaktywny. Aby zakończyć nagrywanie należy nacisnąć przycisk *Stop Recording*, po czym wybrać w oknie dialogowym lokalizację do zapisania powstałego pliku video.

4 Najważniejsze funkcje w programie

4.1 Funkcja, która wykrywa urządzenia

```
private void searchButton_Click(object sender, EventArgs e)
        {
            videoDevicesList = new FilterInfoCollection(FilterCategory.VideoInputDevice);
            foreach (FilterInfo videoDevice in videoDevicesList)
                camListCmb.Items.Add(videoDevice.Name);
        }
```

Tworzymy nowy obiekt typu **FilterInfoCollection** i przypisujemy go do atrybutu *videoDevicesList*, który jest listą wykrytych urządzeń typu video. Następnie przekazujemy nazwy wykrytych urządzeń do obiektu comboBox w interfejsie graficznym aplikacji.

4.2 Funkcja zapisująca zrzut ekranu

```
private void screenshot(object sender, EventArgs e)
        {
            stop(sender, e);
            Bitmap picture = (Bitmap)view.Image;
            saveFileDialog.Filter = "Bitmap Image|*.bmp";
            saveFileDialog.Title = "Save an Image File";
```

```
            saveFileDialog.ShowDialog();
            System.IO.FileStream fs = (System.IO.FileStream)saveFileDialog.OpenFile();
            picture.Save(fs, System.Drawing.Imaging.ImageFormat.Bmp);
            fs.Close();
        }
```

Na początku wywołujemy metodę *VideoCaptureDevice.Stop* odpowiadającą za zatrzymanie źródła video. Następnie pobieramy aktualny obraz wyświetlany w obiekcie *PictureBox* w interfejsie graficznym, po czym wyświtlamy okno dialogowe służące do zapisu zrzutu ekranu na dysku.

4.3 Funkcja odświeżająca wyświetlany obraz

```
private void recordEventHandler(object sender, NewFrameEventArgs eventArgs)
        {
            Bitmap bitmap = (Bitmap)eventArgs.Frame.Clone();
            BrightnessCorrection br = new BrightnessCorrection(brightess);
            ContrastCorrection cr = new ContrastCorrection(contrast);
            SaturationCorrection sr = new SaturationCorrection(saturation);
            bitmap = br.Apply((Bitmap)bitmap.Clone());
            bitmap = cr.Apply((Bitmap)bitmap.Clone());
            bitmap = sr.Apply((Bitmap)bitmap.Clone());

            if (isRecording) writer.WriteVideoFrame(bitmap);

            else
            {
                oldBitmap = bitmap;
                view.Image = bitmap;
            }
        }
```

Tworzymy nową bitmapę, czyli nową klatkę wyświetlanego obrazu. Następnie pobieramy parametry (jasność, kontrast, saturacja), które możemy ustawić w odpowiedniej sekcji naszego programu. Przypisujemy parametry do nowej bitmapy i przypisujemy nową klatkę obrazu to obiektu *PictureBox* w interfejsie graficznym. Jeśli nagrywanie jest włączone nadpisujemy aktualnie wyświetlany obraz.

5 Wnioski

Dzięki przeprowadzonym zajęciom laboratoryjnym zapoznaliśmy się z obsługą kamery przy pomocy języka C# oraz biblioteki iAForge.NET. Wynikiem realizacji zadania jest program który potrafi wykryć podłączone do komputera aktywne kamery USB a następnie umożliwia wybranie i połączenie się z wybranym urządzeniem. Obraz wyświetlany jest w czasie rzeczywistym. Poza tym program udostępnia funkcję nakładanie na obraz filtrów m.in. jasność, kontrast czy saturacja. Jednak najważniejszą i zarazem najbardziej przydatną w życiu powszechnym funkcjonalnością jest nagrywanie filmów oraz zapisywanie poszczególnych klatek w postaci obrazka. Jedynym punktem którego nie udało się zrealizować przez skomplikowanie odczytu i przetworzenia oddzielnie każdego piksela to napisanie procedury umożliwiającej wykrycie ruchu.