



A G H

AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE
WYDZIAŁ FIZYKI I INFORMATYKI STOSOWANEJ

KATEDRA ZASTOSOWAŃ FIZYKI JĄDROWEJ

Projekt Dyplomowy

Opracowanie oprogramowania sterującego mikroskopem optycznym dla
linii badawczej Polyx na synchrotronie Solaris

Developement of the control software for the optical microscope of
Polyx beamline at Solaris synchrotron

Autor: Krzysztof Olech
Kierunek studiów: Fizyka Techniczna
Opiekun pracy: dr inż. Paweł Wróbel

Kraków, 2022

Dziękuję Panu dr inż. Pawłowi Wróblowi za cierpliwość, udostępnienie stanowiska testowego, oraz za nieocenioną pomoc podczas pisania pracy.

Dziękuję Panu Filipowi Kosiorowskiemu za pomoc z pakietem QT i za wstępne opracowanie komunikacji z kamerą i manipulatorem.

Dziękuję Rodzicom za cierpliwość i pomoc z korektą ortograficzną pracy.

Spis treści

1 Wstęp	4
1.1 Synchrotron	4
1.2 Synchrotron Solaris	5
1.2.1 Akcelerator liniowy	5
1.2.2 Pierścień akumulacyjny	5
1.2.3 Pracujące linie	5
1.2.4 Linie w budowie	6
1.3 Linia Polyx	7
1.3.1 Planowane eksperymenty	7
1.3.2 Stanowisko pomiarowe	7
2 Cel pracy	9
2.1 Opis problemu	9
2.2 Zaproponowany układ pomiarowy	9
2.3 Układ testowy	10
2.4 Układ docelowy	10
2.5 Główne założenia programu	11
3 Zastosowane technologie	12
3.1 Język Python	12
3.2 Użyte biblioteki	12
3.2.1 Pakiet PyQt5	12
3.2.2 OpenCV	12
3.2.3 Tupcam	12
3.2.4 Manipulator dll handling	12
4 Opis Kodu	13
4.1 Schemat działania	13
4.2 Obiekt główne okno	14
4.3 Obiekt Oznacz ROI	14
4.4 Podgląd z kamery	15
4.5 ROI	16
4.6 Podgląd ROI	16
4.7 Manipulator obiekt	16
4.8 Map window	17
5 Opis funkcjonalności programu	18
5.1 Main window	18
5.2 Podgląd ROI	20
5.3 Map window	21
6 Podsumowanie	22
6.1 Dalszy rozwój funkcjonalności	22
7 Literatura	23
8 Kod źródłowy	24

1 Wstęp

1.1 Synchrotron

Synchrotron to jeden z typów kołowych akceleratorów cząstek. Wiązka cząstek porusza się po pętli składającej się z prostych odcinków. Taki tor ruchu jest możliwy dzięki zakrzywianiu wiązki polem magnetycznym. Jego natężenie rośnie synchronicznie z rosnącą energią kinetyczną przyśpieszanych cząstek. Idea działania synchrotronu została wymyślona w 1944 przez Władimira Wekslera. Co ciekawe, niecały rok później, w 1945 Edwin McMilan nieświdomy pracy Wekslera, zbudował pierwszy synchrotron elektronowy. Podstawowym elementem synchrotornu jest tak zwany pierścień akumulacyjny w którym to możemy akumulować energię rozprzestrzenionych cząstek przez dłuższy czas. Jesteśmy w stanie wyróżnić kilka rodzajów synchrotronów:

1. Synchrotronowe źródła światła

Cząstki naładowane, przyśpieszane w polu elektrycznym powodują emisję fal elektromagnetycznych. Korzystając z tej zasady, możemy użyć pierścienia akumulacyjnego jako źródła fal elektromagnetycznych o wybranej długości i energii.

2. Synchrotron zderzeniowy

Są to synchrotrony składające się z dwóch pierścieni oraz stwarzyszanych z nimi preakceleratorów oraz komory, w której dochodzi do zderzenia wiązek.

Jednym z problemów związanych z synchrotronami jest to, że nie są one w stanie rozędzić cząstek od zera do wymaganej prędkości. Rodzi to potrzebę prakceleracji cząstek i następnie wstrzyknięcia ich do pierścienia głównego. W zależności od wielkości pierścienia i energii wiązki jaką chcemy osiągnąć, może być to prosty akcelerator liniowy lub nawet wcześniejszy, mniejszy synchrotron. Przykładem synchrotronowego źródła światła posiadającego dwa pierścienie jest MAX IV.

Synchrotrony i promieniowanie synchrotronowe posiadają olbrzymi zakres zastosowań:

- Krystalografia molekularna
- Litografia promieniowaniem rentgenowskim
- Badania nad fluorescencją
- Analiza strukturalna półprzewodników
- Zastosowania medyczne:
 - Obrazowanie
 - Terapia; między innymi raka
 - Opracowanie leków
- Kalibracja urządzeń radiometrycznych

Źródło:[1]

1.2 Synchrotron Solaris

Jest to pierwszy wybudowany w Polsce synchrotron. Powstał w 2015 roku na kampusie 600-lecia Uniwersytetu Jagiellońskiego w Krakowie. Wchodzi w skład Narodowego Centrum Promieniowania Synchrotronowego SOLARIS. Jego nazwa została zainspirowana powieścią Stanisława Lema.

1.2.1 Akcelerator liniowy

Akcelerator liniowy wykorzystany w synchrotronie przyspiesza elektrony do prędkości równej 99.99996% prędkości światła.

Elektrony rozpoczynają akcelerację w dziale elektronowym będącym wnęką rezonansową o częstotliwości własnej 3GHz. Do akceleratora liniowego trafiają paczki elektronów o dokładnie określonej strukturze czasowej oraz energii 2,8MeV.

Następnie elektrony trafiają do akceleratora liniowego. Możemy go podzielić na trzy jednostki przyspieszające, każda jednostka jest wyposażona w dwa pięciometrowe struktury przyspieszające zasilane przez klistron, elementy skupiające wiązkę oraz wnękę SLED. Maksymalna energia elektronów na wyjściu z akceleratora to 600MeV.

Źródło: [2]

1.2.2 Pierścień akumulacyjny

Jest to główny element synchrotronu jego obwód wynosi 96m. Składa się z 12 sekcji. Tycząca się sekcja składa się z dwóch magnesów zakrzywiających, pomiędzy którymi znajdują się magnesy kwadrupolowe i sekstupolowe ogniskujące wiązkę. Pomiędzy każdą sekcją znajdują się 3,5m odcinki proste. Można w nie wstawić urządzenia mające na celu zwiększenie intensywności promieniowania. Energia końcowa wiązki może dochodzić do 1,5GeV a czas jej życia do 13h.

Źródło: [3]

1.2.3 Pracujące linie

- PEEM/XAS

Linia pozwalająca na przeprowadzanie spektroskopii i mikroskopii w zakresie miękkiego promieniowania rentgenowskiego.

- UARPES

Linia pozwalająca przeprowadzać spektroskopię fotoemisyjną dla wysokich kątów. Pozwala ona na dokładne badanie struktury elektronowej nawet złożonych układów cząsteczkowych.

- PHELIX

Linia wykorzystuje spolaryzowane miękkie promieniowanie rentgenowskie do przeprowadzenia spektroskopii absorpcyjnej oraz spektroskopii fotoemisyjnej.

- DEMETER

Linia posiada dwie stacje końcowe, jedną z nich jest skaningowo transmisyjny mikroskopem rentgenowskim, drugą - fotoemisyjnym mikroskopem elektronowym.

1.2.4 Linie w budowie

- SOLABS

Linia będzie umożliwiać przeprowadzanie spektroskopii absorbcyjnej w szerokim zakresie energii promieniowania X.

- SOLCRYS

Linia będzie umożliwiać pomiary wiązką o wysokiej energii - do 25keV. Ponadto, samo stanowisko będzie umożliwiało kontrolę temperatury, jak i ciśnienia, w których to będzie przeprowadzane badanie.

- SOLAIR

Linia będzie posiadać dwa stanowiska: jedno z podczerwoną mikroskopią absorbcyjną, drugie stanowisko będzie umożliwiać nano-spektroskopię połączoną z mikroskopem sił atomowych.

- POLYX

Linia będzie umożliwiać spektroskopię oraz obrazowanie z wykorzystaniem twardego promieniowania rentgenowskiego.

Źródło: [4]

1.3 Linia Polyx

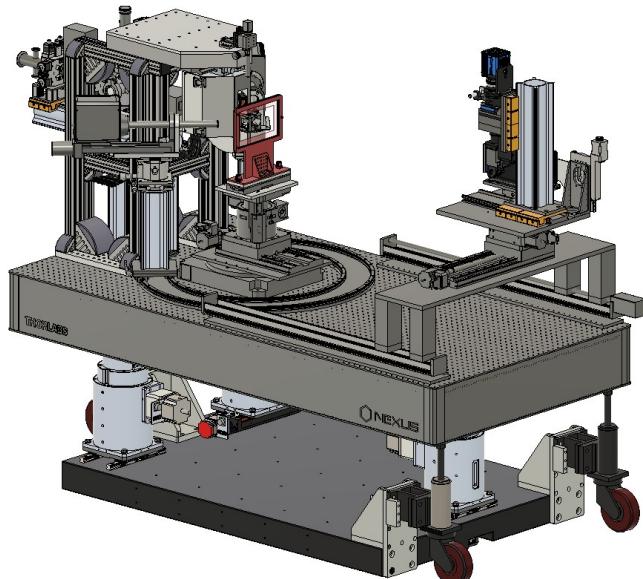
Budowana od 2019 roku linia eksperymentalna PolyX ma służyć do wielomodalnego obrazowania próbek z użyciem twardego promieniowania rentgenowskiego. Źródłem promieniowania dla linii będzie magnes zakrzywiający. Eksperymenty będą prowadzone przy pomocy monochromatycznej wiązki z zakresu 4 – 15 keV formowanej przez dwa rodzaje monochromatorów (wielowarstwowy o dużej wydajności oraz krystaliczny o dużej rozdzielczości energetycznej). Możliwa będzie również praca na polichromatycznej wiązce białej. Promieniowanie synchrotronowe będzie ogniskowane za pomocą optyki kapilarnej – możliwe będzie osiąganie wiązek o rozmiarach o średnicach od 3 μm do 100 μm.

1.3.1 Planowane eksperymenty

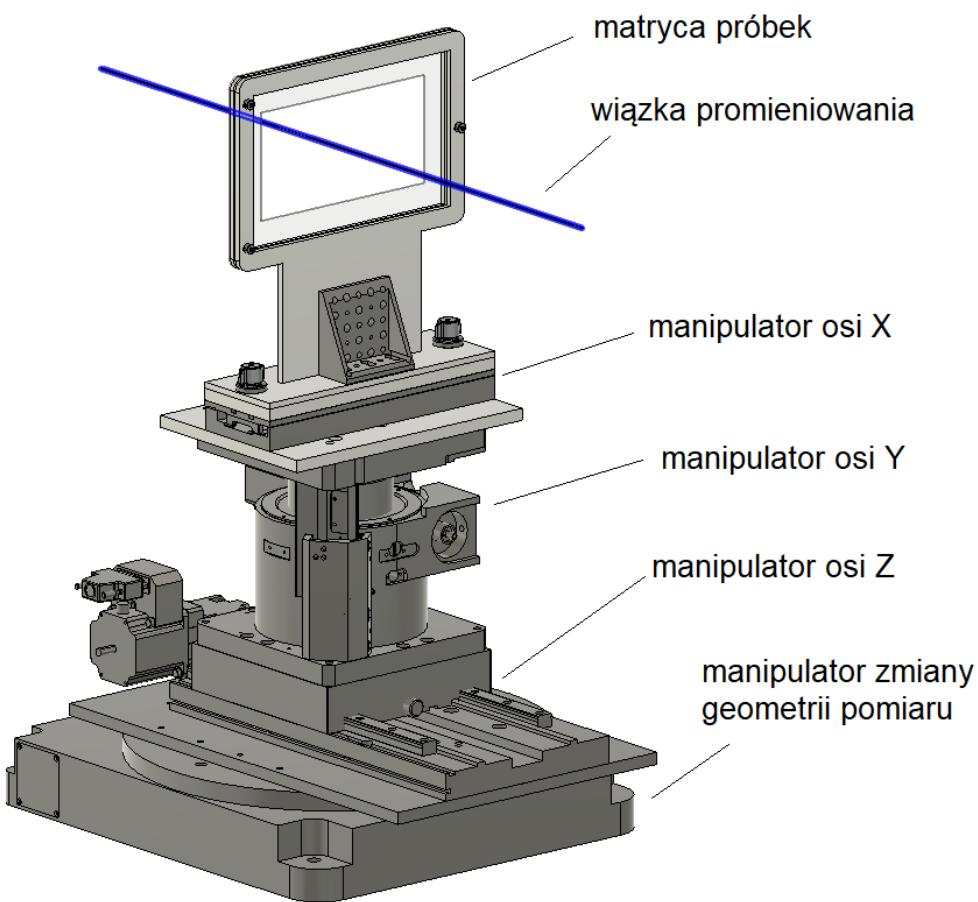
Głównymi technikami implementowanymi na linii PolyX będą:

- (a) obrazowanie rozkładów przestrzennych pierwiastków metodą skaningowej mikroanalizy fluorescencyjnej micro-XRF (X-Ray Fluorescence)
- (b) mikropunktowa analiza otoczenia chemicznego pierwiastków za pomocą rentgenowskiej spektroskopii absorpcyjnej micro-XAS (X-ray Absorption Spectroscopy)
- (c) mikrotomografia w kontraste absorpcyjnym oraz fazowym micro-CT (Computed Tomography)
- (d) rentgenowska mikroanaliza dyfrakcyjna micro-XRD (X-Ray Diffraction)
- (e) wysokorozdzielcze obrazowanie plenoptyczne

1.3.2 Stanowisko pomiarowe



Rysunek 1: Model stołu eksperymentalnego



Rysunek 2: Model manipulatora próbki

Stanowisko pomiarowe linii PolyX (Rysunek 1) zbudowane będzie na pozycjonowanym w pięciu osiach (dwie translacje i trzy rotacje) stole optycznym o wymiarach 2000 mm x 1000 mm. Stanowisko badawcze składać się będzie z czterech grup elementów: sekcji kształtuowania wiązki promieniowania (szczeliny formujące wiązkę, filtry wiązki pierwotnej, monitor natężenia wiązki, układ optyki kapilarnej), wieloosiowego manipulatora próbki, detektorów spektrometrycznych promieniowania X, detektorów do obrazowania rentgenowskiego (kamera tomograficzna, detektor planarny). Manipulator próbki w konfiguracji do pomiarów micro-XRF oraz micro-XAS pokazany jest na rysunku 2. Służyć on będzie do precyzyjnego pozycjonowania oraz przemieszczania względem skolimowanej wiązki promieniowania X zdejmowanej matrycy z próbками. Planowany zakres ruchu manipulatora to 170 mm (oś X), 90 mm (oś Y), 100 mm(oś Z), co oznacza, że maksymalny rozmiar próbki, jaki można poddać analizie, to $170 \times 90 \text{ mm}^2$.

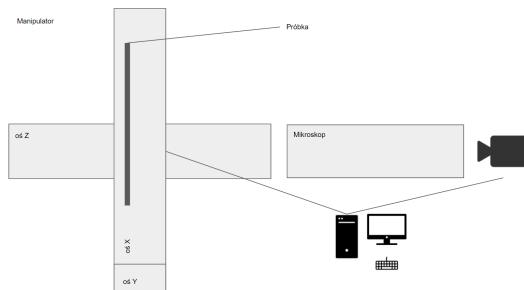
2 Cel pracy

Celem pracy jest opracowanie prototypu oprogramowania kontroli mikroskopu optycznego służącego do planowania analiz próbek metodą micro-XRF na linii PolyX.

2.1 Opis problemu

Nowa linia Polyx na Synchrotronie Solaris posiada wielkoformatowe pole robocze umożliwiające analizę dużych próbek lub wielu mniejszych próbek metodami micro-XRF oraz micro-XAS. Typowo analizom takim poddawane są jedynie wybrane obszary próbek, co wiąże się z koniecznością ich zdefiniowania. Dotychczas rozwiązywano ten problem poprzez oznaczanie obszarów zainteresowania bezpośrednio na aparaturze linii synchrotronu wyposażonej w mikroskop optyczny. Niestety, generuje to dodatkowy czas konieczny do przeprowadzenia pomiaru, ponadto czas ten nie wymaga użycia wiązki synchrotronu, prowadzi zatem do marnowania cennego czasu pomiarowego. Proponowanym na linii Polyx rozwiążaniem będzie zewnętrzne stanowisko umożliwiające inspekcję mikroskopową próbek oraz oznaczenie obszarów do analizy poza aparaturą linii badawczej. Pozwoli to efektywniej wykorzystać czas synchrotronowy umożliwiając równolegle wykonywanie analiz i ich przygotowywanie. Przeniesienie planowania pomiaru próbki poza linię nie jest problemem trywialnym, gdyż musimy nie tylko wyznaczyć same obszary z dokładnością do mikrometrów ale i również wiedzieć, gdzie znajdą się one względem matrycy z próbками tak, aby po przeniesieniu matrycy z próbками na linię pomiarową, dokonać analizy właściwych obszarów.

2.2 Zaproponowany układ pomiarowy



Rysunek 3: Schemat układu

Układ pomiarowy składa się z komputera, na którym działa oprogramowanie wykonane w ramach pracy inżynierskiej umożliwiające:

- poruszanie badanej próbki z wykorzystaniem manipulatora trzyosiowego o układzie kartezjańskim x,y,z.
- podgląd na żywo na próbkę z wykorzystaniem kamery USB patrzącej przez mikroskop optyczny.

Oprogramowanie umożliwia użytkownikowi oznaczanie obszarów zainteresowania i ich eksport w celu dalszej analizy, oraz podgląd całości matrycy z próbками, możliwość edycji nazewnictwa oznaczonych obszarów.

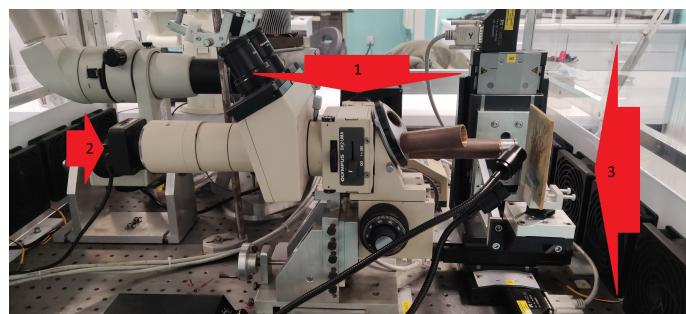
2.3 Układ testowy

Ponieważ docelowy układ mikroskopu dla linii Polyx znajdował się w fazie projektu oprogramowanie zostało opracowane z użyciem układu testowego.

Wykonane na układzie testowym testy umożliwiły również znalezienie brakujących funkcjonalności i ich zaimplementowanie.

Układ składał się z następujących elementów:

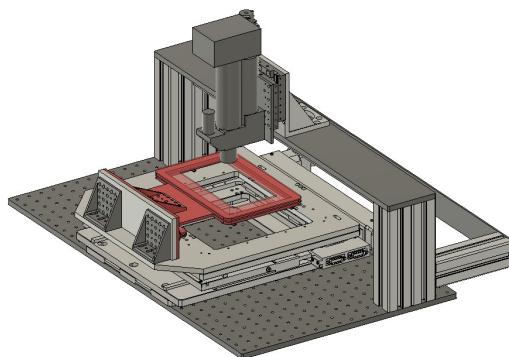
1. Mikroskop Olympus wyposażony w iluminator BH2-UMA o polu widzenia 4mm^2
2. Kamera Tuptek UCMOSO3100KPA-U-NA-N-M-AQ-NA 3.1mp
3. Manipulatora trzyosiowego składającego się z pozycjonerów Physik Instrumente M-405.CG o zakresie ruchu 50 mm sterowanych kontrolerem C-848.



Rysunek 4: Mikroskop i zamontowana do niego kamera

2.4 Układ docelowy

Docelowy układ będzie wykorzystywać specjalnie zaprojektowany manipulator i mikroskop.



Rysunek 5: Model stacji mikroskopu

Jak widzimy os Z zostanie przesunięta w porównaniu z układem testowym i będzie realizowana za pomocą przesuwu kamery. A sam manipulator będzie umożliwiać zamocowanie matryc na próbki identycznych, jak w głównym stanowisku widzianym na Rysunku 1 . Na obu rysunkach oznaczono matryce kolorem czerwonym

2.5 Główne założenia programu

W celu poprawnego funkcjonowania program musi umożliwiać sterowaniem pozycja próbki za pomocą trójosiowego manipulatora, odczytywanie obrazu z kamery podłączonej do mikroskopu.

Z uwagi na dużą różnicę rozmiarów między podglądem a próbką pożądane jest budowanie mapy próbki w trakcie nawigowania po niej.

Program powinien umożliwiać zaznaczanie prostokątnych obszarów, które mają zostać poddane analizie tak zwanych ROI z ang. *Region Of Interest*, przechowywać ich parametry, umożliwiać ich nazywanie oraz edycję oraz wyświetlać je poprawnie na aktualnym podglądzie oraz na zebranej mapie.

3 Zastosowane technologie

3.1 Język Python

Python jest wysokopoziomowym interpretowanym językiem programowania. Cechuje go elastyczność i przejrzystość z uwagi na możliwość wykorzystywania w nim wielu różnych paradygmatów programowania, obecność dynamicznego typowania, a także automatyczne zarządzanie pamięcią. Czytelność kodu zapewnia składnia języka pod postacią stosowania słów z języka naturalnego; zamiast większości symboli oraz podział bloków kodu za pomocą wcięć zamiast nawiasów.

Źródło:[5]

Program został napisany z wykorzystaniem najnowszej wersji języka wspierającej najnowszą wersję PyQt5. Został on wybrany z uwagi na jego wszechstronność, dostępność bibliotek do obsługi kamery oraz możliwość wykorzystania dynamicznej biblioteki napisanej w języku C do obsługi manipulatora.

3.2 Użyte biblioteki

Do stworzenia oprogramowania wykorzystano następujące biblioteki:

3.2.1 Pakiet PyQt5

Jest to port Pakietu QT do Pythona będący multiplatformowym środowiskiem napisanym w C++ umożliwiającym tworzenie wysokopoziomowego interfejsu aplikacji. Jego rozbudowane funkcje umożliwiły stworzenie złożonego, przystępnego interfejsu.

Źródła:[6] [7]

3.2.2 OpenCV

Pakiet OpenCV jest to wielozadaniowy pakiet do Pythona umożliwiający operacje na grafice.

Z uwagi na wszechstronność PyQt5 i bardzo wygodną bibliotekę producenta kamery, jedynie użycie OpenCv to numeracja ROI na podglądzie i skalowanie obrazów.

3.2.3 Tupcam

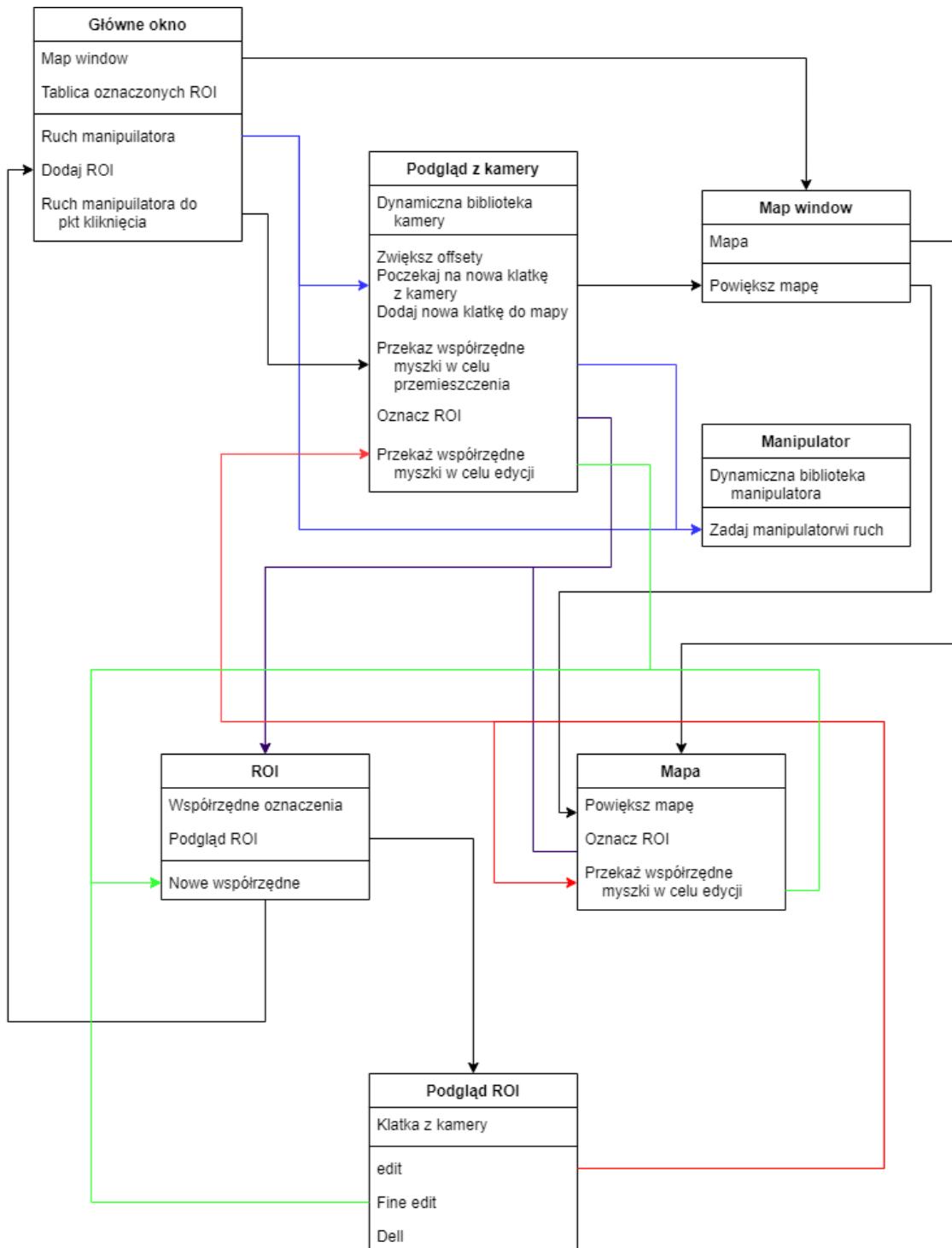
Jest to dostarczona przez producenta kamery biblioteka umożliwiająca odczyt obrazu z kamery w czasie rzeczywistym, posiadająca integrację z pakietem PyQt5.

3.2.4 Manipulator dll handling

Dynamiczna biblioteka napisana w C została dostarczona przez producenta manipulatora. Wykorzystanie jej w Pythonie wymaga konwersji zmiennych pythonowych do odpowiednich typów w celu wywołania odpowiednich funkcji.

4 Opis Kodu

4.1 Schemat działania



Rysunek 6: Schemat Programu

4.2 Obiekt główne okno

Jest to obiekt dziedziczący z QMainWindow. Jest to główne okno programu. Posiada przyciski kierunkowe umożliwiające wysłanie polecen do manipulatora oraz przyciski umożliwiające wysyłanie polecen do obiektu podgląd z kamery a także do obiektu mapy.

Parametry i obiekty przechowywane przez obiekt

Map window, Podgląd z kamery, Lista obiektów ROI

Ustawienia manipulatora:

Wyskakujące okienko umożliwia ustawienie parametrów ruchu manipulatora.

Scroll area:

Obiekt dziedziczący z Qscroll area umożliwiający przewijanie przez widgety, w tym wypadku są to obiekty klasy Podgląd ROI.

Metody klasy

Usuwanie wszystkich ROI:

Metoda usuwająca wszystkie ROI i sprzątająca wszystkie pozostałości po nich takie jak oznaczenia, Podgląd ROI.

Przesuń manipulator:

Metoda wysyłająca polecenie do manipulatora, powoduje jego ruch o zadaną odległość.

Następny/Poprzedni ROI:

Metoda wysyłająca odpowiednią flagę do obiektów Map i podgląd z kamery. Flaga ta wymusza na tych obiektach wyświetlenie jednego obiektu ROI a następnie iteracji w przód/tył po liście obiektów.

Pokaz/schowaj wszystkie obiekty oznaczone:

Metoda wysyłająca flagę do obiektów Map i podgląd z kamery. Powoduje to pokazanie/schowanie wszystkich oznaczonych obiektów.

4.3 Obiekt Oznacz ROI

Obiekt dziedziczący z QLabel. Obiekt wykorzystuje wbudowane w PyQt5 metody umożliwiające odczyt pozycji myszki. Na podstawie zebranych w ten sposób danych tworzone są obiekty ROI. Jeśli dla któregoś z obiektów ROI jest, włączony tryb edycji, to gesty użytkownika pozwalają na zmianę rozmiarów i położenia ROI. Lub w przypadku trybu centrowania na wycentrowaniu na wybranym fragmencie podglądu.

Parametry i obiekty przechowywane przez obiekt

Pozycje myszki:

Przechowywane pozycji myszki, służące do rozpoznawania gestów użytkownika.

Metody klasy

Odczyt pozycji myszki:

Przeciążenie wbudowanej metody PyQt5 odpowiedzialnej za odczyt pozycji myszki.

Paint Event:

Metoda wyświetlająca obraz na podglądzie. Oraz obsługująca otrzymane od obiektu Main window oraz od etykiety obiektów oznaczonych flagi w celu prawidłowego wyświetlenia ROI.

Stwórz ROI:

Metoda tworząca ROI na podstawie ruchu myszki i wysyłająca go do Main window.

Edit ROI:

Metoda wysyłająca do wybranego obiektu ROI pozycje myszki w celu jej edycji.

Wycentruj:

Metoda kalkulująca konieczne przemiecenie i wysyłająca je do manipulatora w celu realizacji.

4.4 Podgląd z kamery

Jest to obiekt dziedziczący z Oznacz ROI. Głównym zadaniem tego obiektu jest prezentacja użytkownikowi podglądu z kamery.

Parametry i obiekty przechowywane przez obiekt

KameraH:

Obiekt z klasy dostarczonej przez producenta kamery umożliwiający odczyt obrazu z kamery w czasie rzeczywistym oraz manipulację parametrami kamery.

Obraz_CV:

Co klatkę konwertowany obraz z kamery przekształcany na obraz OpenCV w celu nadania numeracji obrazów oznaczonych.

Obraz_Pixmap: Jest to obiekt konwertowany z obrazu OpenCv na obiekt klasy Q pixmap w celu wyświetlenia.

Mapa:

Jest to tablica Numpy. O wymiarach przeskalowanego pełnego obszaru w pikselach. Na bieżąco uzupełniana podczas ruchu manipulatora.

Metody klasy

Paint event:

Nadpisana metoda dziedziczona z klasy Obiekt Oznacz ROI, dodająca dodatkowe, obsługiwane flagi oraz odświeżającą obraz z kamery.

4.5 ROI

Jest to obiekt przechowujący wymiary obszarów oznaczonych w pikselach i mm oraz ich podglądy. Ponadto umożliwia edycje ROI.

Parametry i obiekty przechowywane przez obiekt

Podgląd ROI

Współrzędne ROI:

Obiekt przechowuje współrzędne obszaru oznaczonego w bezwzględnych współrzędnych w pikselach oraz w mm.

Nazwa:

Przechowuje automatycznie lub ręcznie nadaną nazwę ROI

Metody klasy

Edit:

Funkcja otrzymująca współrzędne myszy i przetwarzająca je na odpowiednią zmianę ROI na podstawie gestów użytkownika.

4.6 Podgląd ROI

Jest to obiekt dziedziczący z klasy QWidget. Umożliwia on interakcję użytkownika z wybranym ROI'em. Wyświetla Text_edit umożliwiający zmianę nazwy ROI oraz współrzędne oznaczonego ROI. Posiada dwa tryby pracy: standardowy, pozwalający wybrać opcje edycji, edycji z użyciem przycisków lub usunięcie obiektu a także tryb edycji z wykorzystaniem przycisków.

Parametry i obiekty przechowywane przez obiekt

Podgląd:

Zminiaturyzowany QPixmap z narysowanym obiektem oznaczonym.

Metody klasy

Edit:

Podnosi flagę wymuszającą tryb edycji. Na bieżąco, podczas tego trybu aktualizowane są wypisywane współrzędne ROI.

Fine edit:

Zamienia konfigurację obiektu, zamiast podstawowych przycisków wyświetla przyciski do edycji.

4.7 Manipulator obiekt

Jest to obiekt umożliwiający poruszanie manipulatorem. Wykonuje konwersję dynamicznych typów pythonowych na typy z C. Dodatkowo, reaguje i wykonuje wysłane polecenia przemieszczenia.

4.8 Map window

Obiekt dziedziczący z Oznacz ROI. Konwersja współrzędnych myszki w zależności od skalowania okna w celu poprawnego oznaczenia i edycji ROI, była najważniejszym zadaniem tego okna.

Parametry i obiekty przechowywane przez obiekt

Zebrana mapa:

Obiekt klasy openCv przechowujący zebraną do tej pory mapę.

Zebrana mapa pixmap:

Obiekt klasy QPixmap przechowujący zebraną do tej pory mapę.

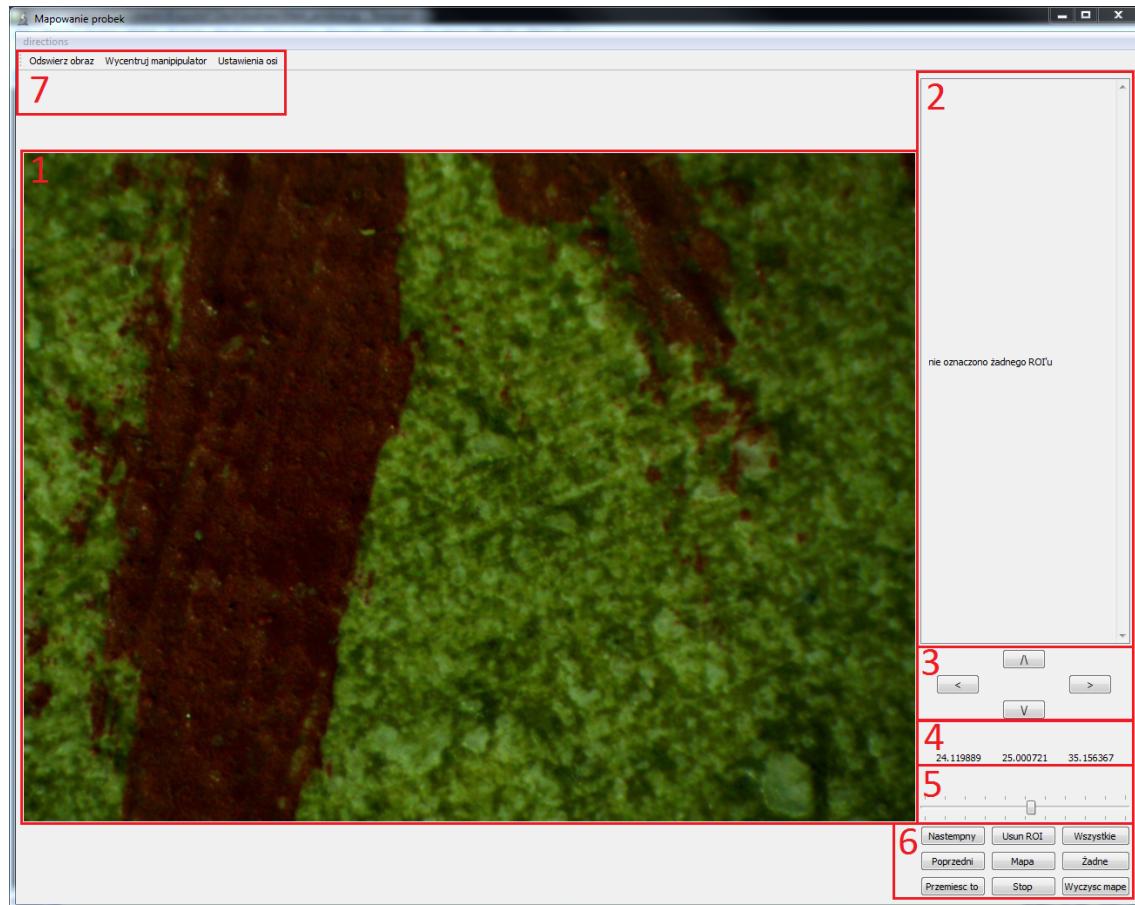
Metody klasy

Odczyt skalowania okna:

Metoda odczytująca aktualną wielkość okna i skalującą zarejestrowane pozycje myszki, tak żeby poprawnie wyświetlać oznaczone obiekty oraz umożliwić poprawne ich oznaczanie i edycje.

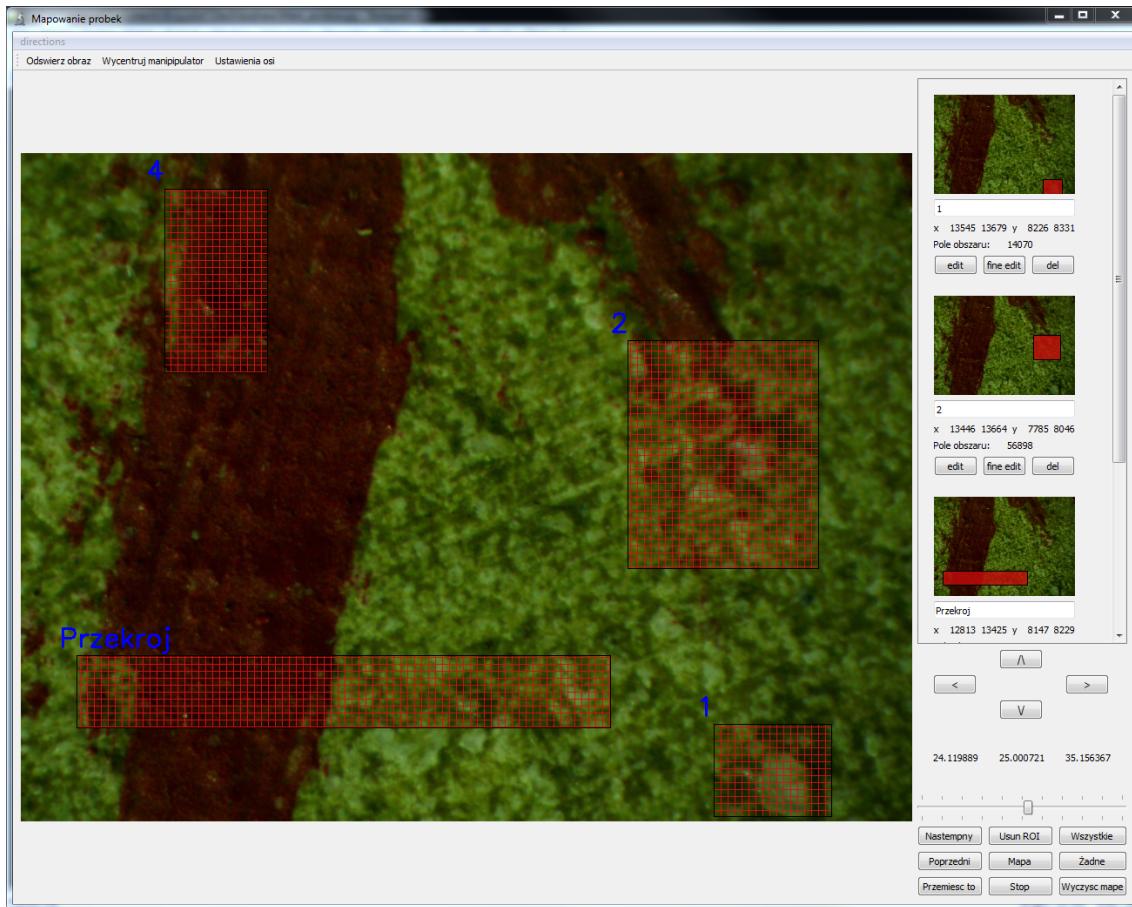
5 Opis funkcjonalności programu

5.1 Main window



Rysunek 7: Główne okno programu

1. Podgląd z kamery, Największy fragment głównego okna umożliwiający interakcje z programem za pomocą gestu 'pres and drag' - oznaczamy obszar zainteresowań.
2. Lista obszarów zaznaczonych, Obszar z możliwością przewijania, zawierający Podgląd ROI umożliwiające interakcje z oznaczonymi wcześniej obszarami
3. Klawisze kierunkowe, Klawisze stowarzyszone z klawiszami strzałek umożliwiające ruch manipulatorem w celu oglądania próbki
4. Współrzędne manipulatora, Widzmy w kolejności współrzędne manipulatora: z, x, y
5. Slider osi Z, Umożliwia on dostosowanie ostrości poprzez przesunięcie próbki bliżej lub dalej. Docelowo będzie również możliwe sterowanie ostrością mikroskopu
6. Klawisze wielozadaniowe, Klawisze umożliwiające *interakcje z oznaczonymi ROI'ami, mapą oraz manipulatorem*
7. Pasek narzędzi, Umożliwia wymuszenie odświeżenia podglądu, wycentrowanie manipulatora, oraz wywołanie okna ustawień manipulatora.

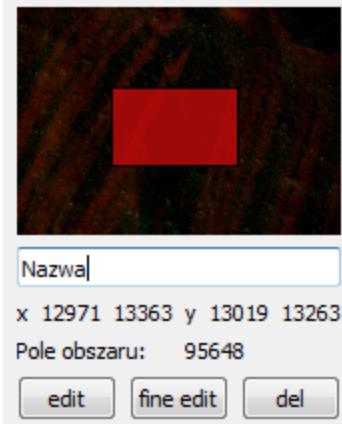


Rysunek 8: Główne okno programu z oznaczonymi obszarami zainteresowania

Widzimy ten sam fragment próbki z oznaczonymi kilkoma obszarami. Możemy zaobserwować, że wypełniała się lista z legendami oznaczonych obszarów. Korzystając z klawiszy wielozadaniowych możemy:

1. Przeglądać oznaczone ROI
2. Schować wszystkie oznaczone ROI
3. Wyświetlić wszystkie oznaczone ROI
4. Usunąć wszystkie oznaczone ROI
5. Wyświetlić okno mapy
6. Przełączyć się w tryb centrowania na wybranym punkcie, w ramach którego zamiast oznaczać ROI będziemy centrować podgląd na oznaczonym punkcie
7. Awaryjnie zatrzymać manipulator

5.2 Podgląd ROI



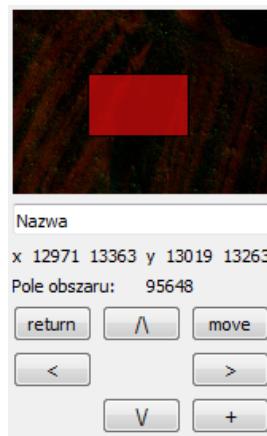
Rysunek 9: Podgląd ROI

Po oznaczeniu obszaru zainteresowania pojawia się stowarzyszona z nim graficzna reprezentacja obiektu Podgląd ROI:

Widzimy miniaturowy podgląd próbki; pole edycji tekstu, w którym to możemy wpisać nazwę dla naszego obszaru. Następnie widzimy współrzędne w pikselach, w których został oznaczony nasz ROI oraz jego pole w pikselach. Poniżej widzimy trzy przyciski. Dwa z nich umożliwiają edycję obszaru; trzeci usuwa oznaczony obszar.

Pierwszy przycisk włącza tryb edycji z wykorzystaniem myszki. Możemy naciskając i przytrzymując jedną z krawędzi ROI na podglądzie rozszerzyć lub zwęzić go w tej osi, naciskając i przytrzymując narożnik manipulujemy dwoma wymiarami ROI w tym samym czasie. Naciśkając środek ROI możemy przesunąć go bez zmian wymiarów.

Drugi z nich zamienia przyciski pod podglądem na zestaw przycisków.



Rysunek 10: Podgląd ROI Fine edit

Główne z nich umożliwiają przesuwanie lub poszerzanie obszaru w zależności od wybranej aktualnej konfiguracji.

Mogemy zamienić tryb edycji z trybem przesuwania za pomocą przełącznika w prawym górnym rogu d-pad. Pod nim znajdują się przełączniki umożliwiające zmianianie, którą oś będziemy aktualnie modyfikować.

5.3 Map window



Rysunek 11: Zebrana do testów mapa



Rysunek 12: Zdjęcie próbki.

Okno pokazujące aktualnie zebrany podgląd całej próbki umożliwia oznaczanie obszarów wielokrotnie większych od standardowego obserwowanego obszaru. Wyświetlany na ekranie obraz nie jest podglądem próbki w czasie rzeczywistym.

6 Podsumowanie

W ramach pracy zostało wykonane sprawne działające oprogramowanie sterujące układem testowym mikroskopu. Docelowy projekt nie jest jednak zamknięty. Jest na tyle dopracowany, że może funkcjonować jako narzędzie do pracy. Niestety nie posiada wszystkich funkcjonalności potrzebnych w docelowym oprogramowaniu oraz nie jest wykonywany na docelowym sprzęcie.

6.1 Dalszy rozwój funkcjonalności

Testy wykonanego oprogramowania ujawniły konieczność rozwoju funkcjonalności programu, który w pierwszej kolejności powinien objąć:

- **Automatyczne zebranie mapy.**

Obecnie zbieranie mapy całej próbki jest uciążliwą, długotrwałą, ręczną pracą użytkownika. Docelowo będzie automatyczna funkcjonalność, która samodzielnie zbierze mapę całej próbki lub wybranego jej fragmentu.

- **Wycentruj na ROI'u.**

Obecnie nie ma żadnej funkcjonalności umożliwiającej powrót do oznaczonego wcześniej ROI'u, trzeba ręcznie nawigować po próbce w celu powrotu. Docelowo zostanie wykonana funkcjonalność umożliwiająca wykonanie tego automatycznie.

- **Autofokus.**

Obsłużenie autofocusu kamery z poziomu kodu

- **Autokalibracja.**

Obecnie kalibracja umożliwiająca przeliczenie miedzy przestrzenią pikseli a przestrzenią metryczną została wykonana raz ręcznie dla jednego przybliżenia. Docelowo zostanie wykonana funkcjonalność autokalibracji.

- **Mozaiki podglądu ROI.**

Obecnie miniaturowa ROI może się składać maksymalnie z pojedynczego podglądu próbki lub całej zebranej mapy. Docelowo będzie to dostosowane do wielkości ROI'u

- **Obsługa zmiennych powiększeń.**

Obecnie powiększenie mikroskopu jest stałe. Docelowo mikroskop będzie posiadał zmienne powiększenie obsługiwane automatycznie. Konieczne będzie sterowanie tym powiększeniem z poziomu programu oraz dobór odpowiedniej kalibracji w zależności od powiększenia.

- **Eksport zebranych danych.**

Obecnie nie ma możliwości zapisania oznaczonych ROI a nie zebranej mapy. Docelowo po ustaleniu formatów plików wymiany zostaną zaimplementowane metody eksportu zebranych danych.

Literatura

- [1] Synchrotron. <https://en.wikipedia.org/wiki/Synchrotron>. Accessed on 2022-1-1.
- [2] A. liniowy. <https://synchrotron.uj.edu.pl/akcelerator-liniowy>. Accessed on 2022-1-2.
- [3] P. akumulacyjny. <https://synchrotron.uj.edu.pl/pierscien-akumulacyjny>. Accessed on 2022-1-2.
- [4] S. Solaris. [https://en.wikipedia.org/wiki/Solaris_\(synchrotron\)](https://en.wikipedia.org/wiki/Solaris_(synchrotron)). Accessed on 2022-1-1.
- [5] Python. <https://docs.python.org/3/>. Accessed on 2022-1-1.
- [6] PyQt5. <https://doc.qt.io/qtforpython/>. Accessed on 2022-1-1.
- [7] PyQt5. <https://pypi.org/project/PyQt5/>. Accessed on 2022-1-1.

8 Kod źródłowy

Kod źródłowy

1	Plik wykonawczy.	25
2	Główne okno programu.	26
3	Klasa umożliwiajaca tworzenie ROI.	36
4	Odczyt obrazu z kamery.	43
5	Wyswietlenie mapy.	51
6	Klasa przechowujaca ROI.	52
7	Klasa odpowiadajaca za etykiete ROI.	61
8	Klasa odpowiedzialna za komunikacje z manipulatorem.	72
9	Okno ustawien.	81
10	Slider osi Z.	82

Manipulator posiada nieintuicyjną kolejność osi:

Y,Z jako osie płaszczyzny równoległej do próbki,

X jako oś prostopadła.

Poza fragmentami, gdzie konieczne było wysyłanie poleceń do manipulatora, starałem się używać intuicyjnej orientacji osi w celu lepszego wyobrażenia działania programu oraz łatwiejszego, późniejszego przejścia na docelowy manipulator.

Link do Githuba: <https://github.com/KrOlech/Inzynierka.git>

Kod źródłowy 1: Plik wykonawczy.

```
1 import sys
2 from PyQt5.QtWidgets import QApplication
3 from Main_window import Glowne_okno
4
5
6 def glowna():
7     """
8         glowna funkcja programu tworzaca glowne okno programu oraz
9         nawiiazujaca polaczenie z manipulatorem i przerywajaca go
10        w przypadku napotkania krytycznego bledu
11    """
12
13 #stworzenie aplikacji
14 app = QApplication(sys.argv)
15
16 manipulaor_obeikt = None
17
18 try:
19     # stworzenie glownego okna programu
20     okno = Glowne_okno()
21
22     manipulaor_obeikt = okno.pobierz_manipulator()
23     #wyswietlenie glownego okna
24     okno.show()
25
26     # wykonanie aplikacji
27     app.exec_()
28
29 except Exception as e:
30     #wypisanie bledu
31     print(e)
32
33     if manipulaor_obeikt is not None:
34         #przerwanie komunikacji z manipulatorem
35         del manipulaor_obeikt
36
37 #wywolanie glownej funkcji programu
38 if __name__ == '__main__':
39
40     glowna()
```

Kod źródłowy 2: Główne okno programu.

```
1 from PyQt5 import QtGui
2 from PyQt5.QtCore import Qt
3 from PyQt5.QtWidgets import QMainWindow, QMessageBox, QToolBar, \
4     QAction, QHBoxLayout, QVBoxLayout, QGridLayout, QWidget, \
5     QLabel, QPushButton, QScrollArea
6 from podglad_z_kamery import Obraz_z_kamery
7 from Map import Map_window
8 from slider import Slider
9 from setingswindows import Okno_ustawien
10 from engineclass import manipulator
11
12 class Glowne_okno(QMainWindow):
13
14     #prostokaty zaznaczone
15     ROI = []
16
17     #bazowa wartosc ile ma sie przesunac manipulator (0.1mm-1)
18     krok = 10
19
20     #zmienna przechowujaca okno mapy
21     map = None
22
23     #ostatni numer nadany ROI'owi
24     ostatnia_nazwa = 0
25
26     tekst_placeholdera = "nie oznaczono zadnego ROI'u"
27
28     def __init__(self, *args, **kwargs):
29         super(Glowne_okno, self).__init__(*args, **kwargs)
30         """
31             Konstruktor glonego okna programu.
32             Tworzy glowne okno nawiazuje polaczenie z kamera
33             i manipulatorem.
34             Przechowuje parametry oznaczonych obszarow.
35         """
36
37         # ustawienie ikony
38         self.setWindowIcon(QtGui.QIcon('icon.png'))
39
40         # ustawienie tytulu okna
41         self.setWindowTitle("Mapowanie probek")
42
43         #ustawienie geometrii okna
44         self.setGeometry(5, 30, 1280, 1024)
45
46         #wylaczenie sledzenia myszki
47         self.setMouseTracking(False)
48
49         #stworzenie podgladu probki i umieszczenie go w leyaucie
50         self.obraz = Obraz_z_kamery(self)
51
52         # polaczenie z manipulatorem
53         self.manipulaor = manipulator()
54         self.manipulaor.glowne_okno(self)
55
56         #stworzenie layoutow
57         self._stworz_layouty()
58
59         #stworzenie przyciskow do przemieszczania podgladu
60         self._Przyciski_kierunkowe()
```

```

61         self._stworz_slider()
62
63     #stworzenie i dodanie przyciskow do wszelkich zastosowan
64     self._przyciski_wielozadaniowe()
65
66     #stworzenie i dodanie obszaru skrolowania
67     self._Scroll_arrea()
68
69     #polaczenie layoutow
70     self._polacz_layouty()
71
72     self._ustaw_centralny_widget()
73
74     self._toolbars()
75
76     self._upadet_position_read()
77
78     self.obraz.odczytaj_klatke()
79
80     self.obraz.odswierz_ofsets()
81
82
83     def closeEvent(self, event):
84         '''
85             Dekonstruktor glownego okna zapisuje pozycje manipulatora
86             oraz upewnia sie ze komunikacja z oknem zostala zatrzymana
87             oraz usuwa zebrana mape
88         '''
89
90         reply = QMessageBox.question(self, "mesage",
91                                     "Czy napewno chcesz zamknac program?", QMessageBox.Yes | QMessageBox.No,
92                                     QMessageBox.Yes)
93
94         if reply == QMessageBox.Yes:
95
96             del self.manipulaor
97             event.accept()
98
99             if self.map != None:
100                 del self.map
101
102         else:
103             event.ignore()
104
105
106     def pobierz_manipulator(self):
107         return self.manipulaor
108
109 ##### Toolbars #####
110 ##### Toolbars #####
111 ##### Toolbars #####
112
113     def _toolbars(self):
114
115         '''
116             metoda prywatna generujaca paski zadan do glownego okna
117         '''
118
119         #pierwszy pasek zawierajacy funkcje umozliwiajace:
120         toolbar = QToolBar("Funkcje")
121         self.addToolBar(toolbar)

```

```

122     #zapisanie podgladu
123     action_1 = self._qactiontoolbar("Odswierz obraz",
124         lambda x: self.obraz.odczytaj_klatke())
125     toolbar.addAction(action_1)
126     #wysrodkowanie manipulatora
127     action_2 = self._qactiontoolbar("Wycentruj manipulator",
128         lambda x: self.manipulator.center())
129     toolbar.addAction(action_2)
130
131     #wywolanie okna akcji
132     seting_window = self._qactiontoolbar("Ustawienia osi",
133         self._ustawienia_osi)
134     toolbar.addAction(seting_window)
135
136     def _qactiontoolbar(self, nazwa, funkcja):
137         '''
138             metoda prywatna tworzaca QAction
139             z podanej funkcji o podanej nazwie
140         '''
141
142         Button = QAction(nazwa, self)
143
144         #dodanie trigera
145         Button.triggered.connect(funkcja)
146
147
148         return Button
149
150     def _ustawienia_osi(self):
151
152         '''
153             metoda prywatna tworzaca i otwierajaca okno ustawien
154         '''
155
156         self.ustawienia_osi = Okno_ustawien(self)
157         self.ustawienia_osi.show()
158
159 #####Layout and widget#####
160 #####Layout and widget#####
161 #####Layout and widget#####
162
163     def _stworz_layouty(self):
164
165         '''
166             Prywatna metoda tworzaca layouty GUI
167         '''
168
169         #Glony layout
170         self._mainlayout = QHBoxLayout()
171
172         #layout
173         self._secoundarylayout = QVBoxLayout()
174
175         #layauty przyciskow i opisow
176         self._kierunkowelayout = QGridLayout()
177         self._przyciskilayout = QGridLayout()
178
179
180         self._sliderlegendslayout = QVBoxLayout()
181         self._sliderlayout = QHBoxLayout()
182

```

```

183     def _polacz_layouty(self):
184         '''
185             Prywatna metoda laczaca layouty GUI
186         '''
187         self._mainlayout.addWidget(self.obraz)
188
189         #laczenie layoutow
190         self._secendarylayout.addWidget(self.scroll)
191         self._secendarylayout.addLayout(self._kierunkowelayout)
192         self._secendarylayout.addLayout(self._sliderlayout)
193         self._secendarylayout.addLayout(self._przyciskilayout)
194         self._mainlayout.addLayout(self._secendarylayout)
195
196     def _ustaw_centralny_widget(self):
197         '''
198             Prywatna metoda Wstawiajacca glowny layout jako glowny widget
199         '''
200         widget = QWidget()
201         widget.setLayout(self._mainlayout)
202         self.setCentralWidget(widget)
203
204 #####
205 ##### przyciski kierunkowe#####
206 #####
207
208     def _nazwij_przyciski_kierunkowe(self):
209         '''
210             Prywatna metoda nadajaca nazwy przyciskom kierunkowym
211         '''
212         nazwy = ['/\\\', "<", ">", '\\/']
213         [switch.setText(name) for name, switch in
214          zip(nazwy, self._kierunkowe)]
215
216     def _podepnij_funkcje_do_przyciskow(self):
217         '''
218             prywatna metoda przypisujaca funkcje
219             i skroty klawiszowe do przyciskow kierunkowych
220         '''
221
222         #tablica funkcji bedaca przypisana do przyciskow
223         fun = [self._key_dwn, self._key_left,
224                self._key_right, self._key_up]
225
226         #Stworzenie tablicy QAction w celu dolaczenia
227         #skrotow klawiszowych
228         self.actions = [QAction("&dw", self),
229                         QAction("&lf", self),
230                         QAction("&ri", self),
231                         QAction("&up", self)]
232
233         #przypiecie funkcji do QAction
234         [a.triggered.connect(f) for a, f in zip(self.actions, fun)]
235
236         #tablica klawiszy kierunkowych
237         keyband = [Qt.Key_Up,
238                    Qt.Key_Left,
239                    Qt.Key_Right,
240                    Qt.Key_Down]
241
242         #przypisanie skrotu klawiszowego do akcji
243         [a.setShortcut(k) for a, k in zip(self.actions, keyband)]

```

```

244
245     #przypisanie funkcji do przyciskow
246     [switch.released.connect(f) for f, switch in zip(fun,
247                                                 self._kierunkowe)]
248
249     menu = self.menuBar()
250     test = menu.addMenu("directions")
251     [test.addAction(f) for f in self.actions]
252
253 def _dodaj_przyciski_do_layout(self):
254     '''
255     Prywatna metoda dodajaca przyciski kierunkowe do layoutu
256     '''
257     it = [3, 2, 4, 3]
258     jt = [2, 3, 3, 4]
259     [self._kierunkowelayout.addWidget(value, j, i) for j, i, value
260      in zip(jt, it, self._kierunkowe)]
261
262 def _dodaj_legende_pozycji_do_layout(self):
263     '''
264     Prywatna metoda tworzaca opisy pozycji manipulatora
265     '''
266
267     self.position_label = [QLabel(str(25.0)),
268                           QLabel(str(25.0)),
269                           QLabel(str(25.0))]
270
271     labelexyz = [QLabel("X"), QLabel("Y"), QLabel("Z")]
272
273     [self._kierunkowelayout.addWidget(value, 7, i) for i, value
274      in zip(range(2, 5, 1), self.position_label)]
275
276 def _Przyciski_kierunkowe(self):
277     '''
278     Prywatna metoda tworzaca przyciski kierunkowe
279     Nastepnie przypisujaca im funkcje
280     '''
281
282     # stworzenie przyciskow
283     self._kierunkowe = [QPushButton() for _ in range(4)]
284
285     # zablokowanie rozmiarow przyciskow
286     [button.setMaximumWidth(50) for button in self._kierunkowe]
287
288     #nadanie nazw przyciskow
289     self._nazwij_przyciski_kierunkowe()
290
291     # przypiecie funkcji do przyciskow
292     self._podepnij_funkcje_do_przyciskow()
293
294     #dodanie do layatow przyciskow kierunkowych
295     self._dodaj_przyciski_do_layout()
296
297     #dodanie i stworzenie odczytu pozycji manipulatora
298     self._dodaj_legende_pozycji_do_layout()
299
300 def _key_up(self):
301     '''
302     metoda wywolujaca metode _key_move
303     z parametrami do przemiszczenia w gore
304     '''
305     self._key_move(self.manipulator.przesun_w_gore, self.obraz.gora, 3, 0)

```

```

305     def _key_left(self):
306         """
307             metoda wywolujaca metode _key_move
308             z parametrami do przemiszczenia w lewo
309         """
310         self._key_move(self.manipulaor.przesun_w_lewo, self.obraz.lewo,
311                         2,1)
312     def _key_right(self):
313         """
314             metoda wywolujaca metode _key_move
315             z parametrami do przemiszczenia w prawo
316         """
317         self._key_move(self.manipulaor.przesun_w_prawo, self.obraz.prawo,
318                         1, 2)
319     def _key_dwn(self):
320         """
321             metoda wywolujaca metode _key_move
322             z parametrami do przemiszczenia w dol
323         """
324         self._key_move(self.manipulaor.przesun_w_dol, self.obraz.dol, 0,
325                         3)
326     def _key_move(self,fun_manipulator,fun_obraz, key_en, key_dis):
327         """
328             Prywatna metoda wykonujaca przemieszczenie.
329             :param fun_manipulator: Funkcja kierunkowa
330                 wykonywana przez manipulator
331             :param fun_obraz: Funkcja kierunkowa
332                 wykonywana przez obraz
333             :param key_en: nr. przycisku ktory moze zostac
334                 odblokowany jesli bedzie mozliwe wykonanie
335                 ruchu w tym kierunku
336             :param key_dis: nr. przycisku ktory moze zostac
337                 zablokowany jesli zostanie osiagniety limit
338         """
339         # blokada przyciskow zeby nie wyslac dwukrotnie polecen do manipulatora
340         [k.setEnabled(False) for k in self._kierunkowe]
341
342         #Zapisanie aktualnej mapy jesli nie zostala jeszcze stworzona
343         self.obraz.zapisz_aktualny_podglad()
344
345         #wykonanie kroku na manipulatorze
346         t = fun_manipulator(self.krok/10)
347
348         #zapisanie nowych pozycji manipulatora
349         self._upadet_position_read()
350
351         #wykonanie odpowiedniej funkcji kierunkowej
352         # na manipulatorze
353         fun_obraz()
354
355         #odblokowanie przyciskow
356         [k.setEnabled(True) for k in self._kierunkowe]
357         """
358             ewentualne zablokowanie przyciskow
359             jesli zostal osiagniety limit
360             lub odblokowanie przycisku jesli zostal cofniety
361         """
362         if t:
363             self._kierunkowe[key_en].setEnabled(True)
364         else:
365             self._kierunkowe[key_dis].setEnabled(False)

```

```

366
367     def _upadet_position_read(self):
368         '''
369         Prywatna metoda wpisujaca odczytanie nowej
370         pozycji manipulatora
371         '''
372         [label.setText(str(position)) for label, position
373          in zip(self.position_labele,
374                  self.manipulaor.pobierz_pozycje_osi())]
375 #####
376 ##### przyciski wielozadaniowe#####
377 #####
378     def _podlacz_functie_do_przyciskow_wielozadaniowych(self):
379         '''
380         Prywatna metoda
381         przypinajaca funkcje do przyciskow wielozadaniowych
382         '''
383         self.przyciski[0].clicked.connect(self.obraz.nastempny)
384         self.przyciski[1].clicked.connect(self.usun_ROI)
385         self.przyciski[2].clicked.connect(self.obraz.narysuj_calosc)
386         self.przyciski[3].clicked.connect(self.obraz.poprzedni)
387         self.przyciski[4].clicked.connect(self.pokaz_mape)
388         self.przyciski[5].clicked.connect(self.obraz.schowajcalosc)
389         self.przyciski[6].setCheckable(True)
390         self.przyciski[6].clicked.connect(
391             self.przelacz_tryb_move_on_pres)
392         self.przyciski[7].clicked.connect(
393             self.manipulaor.simple_stop)
394         self.przyciski[8].clicked.connect(self.obraz.reset_map)
395
396     def _nazwij_przyciski_wielozadaniowe(self):
397         '''
398         Przypisanie nazw do wielozadaniowych przyciskow
399         '''
400
401         nazwy = ["Nastempny", "Usun ROI",
402                  "Wszystkie", "Poprzedni",
403                  "Mapa", "Zadne",
404                  "Przemiesc to", "Stop",
405                  "Wyczysc mape"]
406
407         [switch.setText(name) for name, switch
408          in zip(nazwy, self.przyciski)]
409
410     def _dodaj_przyciski_wielozadaniowe_do_layout(self):
411         '''
412         Prywatna metoda dodajaca przyciski wielozadaniowe do layoutu
413         '''
414         it = [2, 3, 4, 2, 3, 4, 2, 3, 4]
415         jt = [5, 5, 5, 6, 6, 6, 7, 7, 7]
416         [self._przyciskilayout.addWidget(w, j, i) for w, i, j
417          in zip(self.przyciski, it, jt)]
418
419     def _przyciski_wielozadaniowe(self):
420         '''
421         Prywatna metoda tworzaca i organizujaca
422         przyciski wielozadaniowe
423         '''
424         #przyciski multipurpos
425         self.przyciski = [QPushButton() for _ in range(9)]
426

```

```

427     [button.setMaximumWidth(100) for button in self.przyciski]
428
429     self._nazwij_przyciski_wielozadaniowe()
430
431     self._podlacz_functie_do_przyciskow_wielozadaniowych()
432
433     #dodanie przyciskow
434     self._dodaj_przyciski_wielozadaniowe_do_layout()
435
436 def pokaz_mape(self):
437     '''
438     Wyswietlenie mapy probki
439     ewentualne jej stworzenie jesli jeszcze nie zostala stworzona
440     '''
441
442     if self.map is None:
443         self.map = Map_window(self.obraz.ponbierz_map(), self)
444         self.map.show()
445     else:
446         self.map.new_image(self.obraz.ponbierz_map())
447         self.map.show()
448
449 def przelacz_tryb_move_on_pres(self):
450
451     '''
452     Metoda obslugujaca centrowanie podgladu
453     na wybranym fragmencie
454     '''
455
456     if self.przyciski[6].isChecked():
457
458         if self.map is not None:
459             self.map.move_to_point = True
460             self.obraz.przemiesc_sie_do_pktu = True
461
462     else:
463         if self.map is not None:
464             self.map.move_to_point = False
465             self.obraz.przemiesc_sie_do_pktu = False
466
467 ##### Scroll area #####
468 ##### Scroll area #####
469 #####
470
471 def _Scroll_arrea(self):
472     '''
473     Prywatna metoda tworzaca obszar przewijany umozliwiajacy
474     podglad labeli opisujacych oznaczone obszary
475     '''
476     self.defalaut_label = QLabel(self.tekst_placeholder)
477
478     self.scroll = QScrollArea()
479     self.widget = QWidget()
480     self.vbox = QVBoxLayout()
481
482     self.widget.setLayout(self.vbox)
483
484     self.scroll.setVerticalScrollBarPolicy(Qt.ScrollBarAlwaysOn)
485     self.scroll.setHorizontalScrollBarPolicy(Qt.ScrollBarAlwaysOff)
486
487     self.scroll.setWidgetResizable(True)

```

```

488         self.scroll.setWidget(self.widget)
489
490         self.scroll.setMaximumSize(720, 650)
491
492         self.vbox.addWidget(self.defalaut_lable)
493
494 #####
495 #####
496 ##### zarzodzanie Roi'ami#####
497 ##########
498
499     def dodaj_ROI(self, ROI):
500         """
501             Metoda obslugujaca dodanie nowego obszaru zaintesowania
502             :param ROI: Obiekt ROI
503         """
504
505             #obsluzenie usuwania domyslnej etykiety
506             if self.defalaut_lable != 0:
507                 try:
508                     self.vbox.removeWidget(self.defalaut_lable)
509                 except Exception as e:
510                     print(e)
511                 self.defalaut_lable = 0
512
513             #dodanie podgladu ROI
514             self.vbox.addWidget(ROI.pobierz_podglad())
515
516             #dodanie ROIa do listy ROI
517             self.ROI.append(ROI)
518
519     def usun_ROI(self):
520         """
521             Metoda usuwajaca wszystkie oznaczone obszary
522         """
523
524             #zapytanie uzytkownika czy na pewno chce to zrobic
525             reply = QMessageBox.question(self, "mesage",
526                             "Czy na pewno chcesz usunac wszystkie oznaczone ROI?", QMessageBox.Yes | QMessageBox.No, QMessageBox.Yes)
527
528             if reply == QMessageBox.Yes:
529                 while self.ROI != []:
530                     [r._del_() for r in self.ROI]
531
532     def usun_wybrany_ROI(self, ROI):
533         """
534             metoda usuwajaca podany ROI
535             :param ROI: obiekt klasy ROI
536             :return:
537         """
538
539
540         try:
541             #usuniecie podgladu
542             self.vbox.removeWidget(ROI.pobierz_podglad())
543         except AttributeError:
544             pass
545
546             #usuniecie ROI'a z listy
547             if ROI in self.ROI:
548                 self.ROI.remove(ROI)

```

```

549     #wrzucenie placeholdera jesli nie ma ROI
550     if len(self.ROI) == 0 and self.defalaut_lable == 0:
551         self.defalaut_lable = QLabel(self.tekst_placeholdera)
552         self.vbox.addWidget(self.defalaut_lable)
553
554 ##### Zarzadzanie Sliderem #####
555 ##### Zarzadzanie Sliderem #####
556 ##### Zarzadzanie Sliderem #####
557 ##### Zarzadzanie Sliderem #####
558
559     def _stworz_slider(self):
560         '''
561         Metoda dodajaca slider osi Z
562         '''
563         x = self.manipulaor.pobierz_pozycje_osi('x')[0]
564         self.slide = Slider(self, x - 5, x + 5, x, Qt.Horizontal)
565
566         self._sliderlayout.addWidget(self.slide)
567
568     def ustaw_krok(self, value):
569         '''
570         Metoda ustawiajaca krok dla manipulatora
571         :param value: wartosc liczbowa
572         '''
573
574         self.krok = value

```

Kod źródłowy 3: Klasa umozliwiajaca tworzenie ROI.

```
1 import cv2
2 from PyQt5.QtWidgets import QLabel
3 from PyQt5.QtCore import QPoint, QRect
4 from PyQt5.QtGui import QPixmap, QImage, QPalette, QColor,\n    QPainter, QBrush
5 from obszar_oznaczony_clasa import Obszar_zaznaczony
6
7
8
9 class oznacz_ROI(QLabel):
10     '''
11     Klasa dziedzicząca z QLabel umozliwiająca wyświetlenie
12     obrazu oraz zaznaczenie na nim obszaru zainteresowania
13     i edycji tego obszaru.
14     '''
15     # obiekt Klasy Glowne_okno podany jako argument
16     # przy tworzeniu obiektu klasy Obraz_z_kamery -
17     # pozwala na komunikacje z oknem głównym
18     main_window = ''
19
20     # aktualna pozycja myszki nad widgetem
21     x = 0
22     y = 0
23
24     # ostatnie 2 pozycje kliknięte pozycja myszki nad widgetem
25     x1 = 0
26     y1 = 0
27     x2 = 0
28     y2 = 0
29
30     # punkty początkowe i końcowe prostokąta
31     poczatek = QPoint()
32     koniec = QPoint()
33
34     ilosc_klikniec = False
35
36     # zmienne obsługujące wyświetlanie ROI
37     # 'no_rectagle' nie rysuje żadnych obszarów oznaczonych
38     # 'all_rectagls' rysuje wszystkie obszary oznaczone
39     # 'One_rectagle' rysuje jeden wybrany obszar oznaczony
40     # 'viue_muve' obsługuje rysowanie podczas przemieszczania
41     # 'previu_rectagle' obsługuje rysowanie nowego obszaru
42     co_narysowac = 'all_rectagls'
43
44     # iterator do wyświetlenia poprzedniego, następnego zaznaczenia
45     ktorysty = 0
46
47     # wartości offsetu aktualnego podglądu
48     offsetx = 0
49     offsety = 0
50
51     # rozmiar obszaru
52     rozmiar = (1024, 768)
53
54     # zmienna umozliwiająca przeliczenie pikeli na mm
55     # 1 mm to 510 pikeli
56     delta_pikeli = 510
57
58     # zmienne obsługujące tryb edycji
59     edit_tryb = False
60     edited_roi = None
```

```

61
62     #zmienna określająca czy jesteśmy w trybie edycji
63     #czy centrowania na kliknięciu
64     przemiesc_sie_do_pktu = False
65
66     # maksymalna pozycja manipulatora w mm
67     manipulator_max = 50
68
69     # skala mapy
70     skala = 32
71
72
73     def __init__(self, main_window, *args, **kwargs):
74         super(oznacz_ROI, self).__init__(*args, **kwargs)
75
76         # wskaznik do głównego okna programu
77         self.main_window = main_window
78
79         # ustawienie bazowej szaty graficznej
80         palette = self.palette()
81         palette.setColor(QPalette.Window, QColor('white'))
82         self.setPalette(palette)
83
84         # wyłącza skalowanie okna
85         self.setScaledContents(False)
86
87         # włączenie śledzenia myszki
88         self.setMouseTracking(True)
89
90         # Tworzy białe tło
91         self.setAutoFillBackground(True)
92
93 #####wgrywanie obrazu#####
94
95     def zaladuj_obraz(self, nasrysuj_opisy = False,
96                         narysuj_jeden_ROI = False):
97         """
98             Metoda dodająca opisy do podglądu oraz wgrywająca
99             podgląd do etykiety
100            :param nasrysuj_opisy: wartość logiczna określająca
101            czy wypisywać opisy czy nie
102            :param narysuj_jeden_ROI: wartość logiczna określająca
103            czy wyświetlić jeden czy wszystkie ROIe
104        """
105
106        # skalowanie obrazu
107        klatka = self.image_opencv.copy()
108
109        # skalowanie kopii obrazu
110        self.klatka = self.image_opencv.copy()
111
112        # dodanie opisów w odpowiednich miejscach
113        if nasrysuj_opisy:
114            for i, rectangle in enumerate(self.main_window.ROI):
115                rx, ry = rectangle.pobierz_lokacje_tekstu(self.offsetx,
116                                                self.offsety)
117
118                cv2.putText(klatka, str(rectangle.pobierz_nazwe()),
119                            (rx, ry), cv2.FONT_HERSHEY_SIMPLEX,
120                            1, (0, 0, 255), 2)

```

```

122     #dodanie opisow pojedynczego ROI'a
123     #jesli ta opcja zostala wybrana
124     if narysuj_jeden_ROI:
125
126         rx, ry = self.main_window.ROI[self.ktory].\
127                     pobierz_lokacje_tekstu(self.ofsetx,
128                                         self.ofsety)
129
130         cv2.putText(klatka, str(self.main_window.\
131                             ROI[self.ktory].pobierz_nazwe()),
132                     (rx, ry), cv2.FONT_HERSHEY_SIMPLEX,
133                     1, (0, 0, 255), 2)
134
135     # konwersja z open Cv obrazu na QImage
136     obraz_z_klatki = QImage(klatka,
137                               klatka.shape[1],
138                               klatka.shape[0],
139                               klatka.strides[0],
140                               QImage.Format_RGB888)
141
142     #konwersja obrazu z Qimage na pixmap
143     self._obraz_z_klatki = QPixmap.fromImage(obraz_z_klatki)
144
145     # wgranie obrazu
146     self.setPixmap(self._obraz_z_klatki)
147
148     #zablokowanie rozmiaru
149     self.setMaximumSize(self._obraz_z_klatki.width(),
150                         self._obraz_z_klatki.height())
151
152
153 #####mouse tracking#####
154
155     def mousePressEvent(self, e):
156         '''
157             Metoda nadpisajaca wbudowany event pyqt
158             wykonywany w momencie przycisniecia przycisku myszki
159             :param e: odczytana pozycja myszki
160         '''
161
162         # Sprawdzenie trybu pracy
163         if self.edit_tryb:
164             #tryb edycji ROI'u
165             self.edited_roi.wspolrzedne_nacisniecia(e, self.ofsetx,
166                                                 self.ofsety)
166
167
168         elif self.przemiesc_sie_do_pktu:
169             #tryb centrowania na punkcie
170             self._wycentruj_na_pkcie_function(e)
171
172         else:
173             #podstawowa opcja umozliwiajaca oznaczenie ROI'u
174             self._zapisz_pierwsze_kliknienie(e)
175
176     def _zapisz_pierwsze_kliknienie(self, e):
177         '''
178             Prywatna metoda zapisujaca pozycje pierwszego kliknienia
179         '''
180         # zapis pozycji kliknienia
181         self.x1 = e.x()
182         self.y1 = e.y()

```

```

183
184     # zapisanie pozycji pierwszego kliknienia
185     self.poczatek = e.pos()
186
187     #Zapisanie ze juz raz doszlo do kliknienia
188     self.ilosc_klikniet = True
189
190     #podniesienie flagi ze nalezy narysowac podglad nowo
191     #zaznaczanego ROI'u
192     self.co_narysowac = 'previu_rectagle'
193
194     def _wycentruj_na_pkcie_function(self, e):
195         '''
196             Prywatna metoda konwertujaca wspolrzedne w pixelach
197             na mm i zadajaca manipulatorowi przemieszczenie
198             w celu wycetrowania na wybranym punkcie
199         '''
200
201         x1, y1 = e.x(), e.y() #zapisanie pozycji kliknienia
202
203         #ccx - polowa rozmiaru x
204         #ccy - polowa rozmiaru y
205         ccx, ccy = self.rozmiar[0] / 2, self.rozmiar[1] / 2
206
207         #odleglosci ktore nalezy przemiescic manipulator w pixelach
208         self.dxp, self.dyp = int(y1 - ccy), int(x1 - ccx)
209
210         #konwersja odleglosci w pixelach na odleglosci w mm
211         dx = (x1 - ccx) / self.delta_pixeli
212         dy = (y1 - ccy) / self.delta_pixeli
213
214         #zadanie przemieszczenia manipulatorowi
215         self.main_window.manipulator.move_axes_to_abs_woe_ofset('yz',
216                                         [dx, dy])
217
218         #odswiezienie ofsetow
219         self.ofsetx += self.dxp
220         self.ofsety += self.dyp
221
222         #odswiezienie mapy
223         self._map_update()
224
225     def mouseReleaseEvent(self, e):
226         '''
227             Metoda przeciazajaca Pqt5 event umozliwiajaca
228             obsluzenie momentu puszczenia przycisku myszki
229             :param e: pozycja myszki
230         '''
231         #wybranie odpowiedniego trybu
232         if self.edit_tryb:
233             #przekazanie pozycji w celu edycji
234             self.edited_roi.wspolrzedne_puszczenia(e, self.ofsetx,
235                                                 self.ofsety)
236
237         elif self.przemiesc_sie_do_pkta:
238             #ignorowanie eventu w ramach przesuwania manipulatora
239             pass
240         else:
241             #zapisanie pozycji puszczenia przycisku
242             self._zapisz_puszczenie_przycisku(e)

```

```

244     def _zapisz_puszczenie_przycisku(self, e):
245         '''
246             Zapisanie miejsca puszczenia przycisku myszki
247         '''
248         # zapisanie wspolrzednych klikniecia
249         self.x2 = e.x()
250         self.y2 = e.y()
251
252         # zapisanie pozycji klikniecia jako obiekt klasy qpoint
253         self.koniec = e.pos()
254
255         # dopisanie nowego prostokata do listy
256         nowy_prostokat = self.stworz_prostokat()
257
258         self.main_window.ROI.append(nowy_prostokat)
259
260         # implementacja iteratora wyswietlanego prostokata
261         self.ktory += 1
262
263         #wyczszczescenie licznika klikniec
264         self.ilosc_klikniet = False
265
266         #podniesienie flagi w celu wyrysowania wszystkich ROI
267         self.co_narysowac = 'all_rectagls'
268
269         self.update()
270
271     def mouseMoveEvent(self, e):
272         '''
273             Metoda przeciazajaca Pqt5 event
274             umozliwiajaca obsluzenie momentu przesuniecia myszki
275             :param e: pozycja myszki
276         '''
277
278         if self.edit_tryb:
279             # tryb edycji ROI'u
280             self.edited_roi.przemiesc_kordynaty(e, self.ofsetx,
281                                                 self.ofsety)
282
283         elif self.przemiesc_sie_do_pkta:
284             # ignorowanie eventu w ramach przesuwania manipulatora
285             pass
286
287         elif self.ilosc_klikniet:
288             # parametry umozliwiajace rysowanie
289             # podgladu tworzonego ROI
290             self._stworz_tymczasowy_ROI(e)
291
292     def _stworz_tymczasowy_ROI(self, e):
293
294         self.x2 = int(e.x())
295         self.y2 = int(e.y())
296
297         # zapis aktualnej pozycji myszki
298         # w celu wyswietlenia podgladu
299         self.koniec = e.pos()
300
301         #podniesienie odpowiedniej flagi
302         self.co_narysowac = 'previu_rectagle'
303
304         self.update()

```

```

305
306 #####ROI edit#####
307
308     def _map_update(self):
309         """
310             Abstrakcyjna metoda odswiezajaca mape
311         """
312         pass
313
314     def edit_roi(self, roi):
315         """
316             Metoda wywolywana przez ROI w celu samoedycji
317             :param roi: Roi wyolujacy edycje
318         """
319         #podniesienie odpowiedniej flagi
320         self.edit_tryb = True
321         self.przemiesc_sie_do_pktu = False
322
323         #zapisanie wskaznika do edytowanego ROI'u
324         self.edited_roi = roi
325
326     def zakończ_edit(self):
327         """
328             Metoda konczaca edycje ROI
329         """
330         #opuszczenie odpowiedniej flagi
331         self.edit_tryb = False
332         #usuniecie wskaznika do ROI
333         self.edited_roi = None
334         #zwrocenie aktualnego podgladu w celu aktualizacji
335         return self.klatka
336
337 #####tworzenie prostokatow#####
338     def narysuj_prostokat(self, prostokat):
339         """
340             Metoda zwracajaca Qrectagle
341             w celu wyrysowania go na podgladzie
342             :param prostokat: obiekt klasy ROI
343             :return: obiekt klasy Qrectagle
344         """
345         x = prostokat.pobierz_prostokat(self.ofsetx, self.ofsety,
346                                         1 / self.skala)
347         return x
348
349     def stworz_prostokat(self):
350         """
351             Metoda tworzaca obiekt klasy ROI
352             :return: obiekt klasy roi stworzona na podstawie zapisanych dancyh
353         """
354         #ponisienie nr domyslnej nazwy
355         self.main_window.ostatnia_nazwa += 1
356
357         ROI = Obszar_zaznaczony(self, self.x1, self.y1, self.x2, self.y2,
358                                 self.klatka, self.ofsetx, self.ofsety,
359                                 self.main_window.ostatnia_nazwa,
360                                 1 / self.skala)
361
362         #zapisanie ROI do tablicy
363         self.main_window.dodaj_ROI(ROI)
364
365         return ROI

```

```

366
367     def rmv_rectagle(self, roi):
368         '''
369         Metoda usuwajaca ROI
370         :param roi: Roi do usuniecia
371         '''
372
373         #jesli ROI jest w tablicy to go usuwamy
374         if roi in self.main_window.ROI:
375             self.main_window.ROI.remove(roi)
376
377         #wywolanie metody sprzatajacej po ROI w glownym oknie
378         self.main_window.usun_wybrany_ROI(roi)
379
380         #podniesienie odpowiedniej flagi
381         self.co_narysowac = 'all_rectagls'
382
383         self.update()
384
385 #####Paint Event#####
386     def paintEvent(self, event):
387         '''
388         Metoda przeciazajaca PyQt5 event
389         obslugujaca wyswietlanie ROI i podgladu
390         '''
391         # inicializacja paintera
392         qp = QPainter(self)
393
394         # wyrysowanie obrazu z kamery
395         qp.drawPixmap(self.rect(), self._obraz_z_klatki)
396
397         # wgranie stylu rysowania ROI'u
398         qp.setBrush(QBrush(QColor(200, 10, 10, 200)))
399
400         # zmienne decydujace o wypisywaniu opisow
401         nasrysuj_opisy = True
402         narysuj_jeden_ROI = False
403
404         if self.co_narysowac == 'all_rectagls':
405             # pokazuje wszystkie prostokaty
406             self.wszystkie_prostokaty(qp)
407
408         else:
409             # podstawowa opcja rysuje nowy prostokat
410             self.wszystkie_prostokaty(qp)
411             qp.drawRect(QRect(self.poczatek, self.koniec))
412
413         #odswiezanie podgladu
414         self.zaladuj_obraz(nasrysuj_opisy, narysuj_jeden_ROI)
415
416     def wszystkie_prostokaty(self, Painter):
417         '''
418         Metoda rysujaca wszystkie ROI'e
419         :param Painter: Qt Painter
420         '''
421         for rectangle in self.main_window.ROI:
422             Painter.drawRect(self.narysuj_prostokat(rectangle))

```

Kod źródłowy 4: Odczyt obrazu z kamery.

```
1 import numpy as np
2 import cv2
3 from PyQt5.QtWidgets import QMessageBox
4 import sys
5 import toupcam as tcam
6 from PyQt5.QtCore import pyqtSignal, pyqtSlot, Qt, QRect
7 from PyQt5.QtGui import QImage, QPainter, QBrush, QColor
8 from roi_create import oznacz_ROI
9 from Map import Map_window
10
11
12
13 class Obraz_z_kamery(oznacz_ROI):
14
15     """
16     Klasa Obraz_z_kamery dziedziczy z oznacz_ROI,
17     pozwala na odczyt obrazu z kamery oraz zbieranie mapy.
18     """
19
20     # wlasne sygnaly umozliwiajace obsluge kamery
21     nowy_obraz_z_kamery = pyqtSignal()
22     nowy_wymuszony_obraz_z_kamery = pyqtSignal()
23
24     h_cam = None      # wskaznik do kamery
25     buf = None        # bufor na video
26     w = 0             # szerokosc video
27     h = 0             # wysokosc video
28
29     # ' ', 'up', 'dawn', 'right', 'left', 'multi',
30     zmiana_kierunku = ' '
31
32     # zmienna binarna pozwalajaca na
33     # stworzenie obiektu mapa z pierwszej
34     # klatki
35     first = True
36
37     # pixmap obiekt odczytany z kamery
38     klatka = True
39
40     # numpy arrey ktora bedzie przechowywac mapy
41     map = np.zeros(100)
42     map.shape = (10, 10, 1)
43
44     # skala mapy
45     skala_mapy = 32
46
47     # skala mapy
48     skala = 1
49
50     # konstruktor
51     def __init__(self, main_window, *args, **kwargs):
52         super(Obraz_z_kamery, self).__init__(main_window, *args, **kwargs)
53
54         self._inicjalizacja_kamery()
55
56         self.h_cam.put_AutoExpoEnable(False)
57
58         self.odczytaj_klatke()
59
60
```

```

61 ######
62 ##### odczyt kamery#####
63 ##########
64
65     def odczytaj_klatke(self):
66         """
67             Metoda wysylajaca sygnal do kamery
68             wymuszajacy odczyt nowej klatki
69         """
70
71         self.h_cam.Snap(1)
72
73
74     @pyqtSlot()
75     def nowy_wymuszony_obraz_z_kamery_sygnal(self):
76         """
77             Metoda odczytujaca klatke z kamery konwertujaca
78             ja na obraz openCV
79             zapisujaca ja do freame_2 oraz image_opencv
80             nastepnie wywolujaca metode rozszerzajaca mape
81         """
82
83     # sprawdzamy komunikacje z kamera
84     if self.h_cam is not None:
85
86
87         #definiujemy rozmiar buforu na obraz
88         w, h = self.h_cam.get_Size()
89         bufsize = ((w * 24 + 31) // 32 * 4) * h
90         still_img_buf = bytes(bufsize)
91
92         #pobieramy do buforu obraz z kamery
93         self.h_cam.PullStillImageV2(still_img_buf, 24, None)
94
95         #konwertujemy obraz z buforu do obrazu opencv
96         obraz = self.obraz_bitowy_do_obrazu_opencv(still_img_buf)
97
98         #zapisujemy odczytany obraz
99         self.klatka_2 = obraz.copy()
100
101        self.image_opencv = obraz.copy()
102
103        #rozszerzamy mape
104        self.zapisz_aktualny_podglad()
105        self.zaladuj_obraz(True, False)
106
107    @staticmethod
108    def metoda_wywolwana_przez_kamere(nevent, ctx):
109        """
110            Statyczna metoda wywolywana przez kamere
111            :param nevent: event weemitowany przez kamere
112            :param ctx: self
113        """
114
115        if nevent == tcam.TOUPCAM_EVENT_IMAGE:
116            ctx.nowy_obraz_z_kamery.emit()
117
118        elif nevent == tcam.TOUPCAM_EVENT_STILLIMAGE:
119            ctx.nowy_wymuszony_obraz_z_kamery.emit()
120
121

```

```

122     @pyqtSlot()
123     def nowy_obraz_z_kamery_sygnal(self):
124         '''
125             Metoda odczytujaca obraz z kamery,
126             konwertujaca go na obraz openCV,
127             zapisujaca go
128             oraz wgrywajaca go do podgladu
129         '''
130
131         #sprawdzenie komunikacji z kamera
132         if self.h_cam is not None:
133
134             #proba odczytania klatki
135             try:
136                 self.h_cam.PullImageV2(self.buf, 24, None)
137
138             #obsluzenie bledu na wypadek bledu odczytu
139             except tcam.HRESULTException:
140
141                 print('pull obraz failed')
142                 QMessageBox.warning(self, '', 'pull obraz failed',
143                                     QMessageBox.Ok)
144
145             else:
146
147                 #konwersja i odczyt obrazu z bufora
148                 self.image_opencv = self.\
149                     obraz_bitowy_do_obrazu_opencv(self.buf)
150
151                 #zaladowanie obrazu
152                 self.zaladuj_obraz(True, False)
153
154
155     def obraz_bitowy_do_obrazu_opencv(self,
156                                         still_img_buf, dtype=np.uint8):
157         '''
158             metoda konwertujaca obraz z buforu bitow na
159             tablice numpaya obslugiwana przez openCV
160             :param still_img_buf: bufor z obrazem
161             :param dtype: typ danych
162             :return: skonwertowana tabela z obrazem
163         '''
164
165         arr_1d = np.frombuffer(still_img_buf, dtype=dtype)
166
167         return arr_1d.reshape(self.h, self.w, 3)
168
169     def _inicjalizacja_kamery(self):
170         '''
171             Metoda inicializujaca kamere
172         '''
173
174         #wywolanie instrukcji inicializujacej
175         a = tcam.Toupcam.EnumV2()
176
177         #sprawdzenie czy adres komunikacyjny nie jest bledny
178         if len(a) <= 0:
179             QMessageBox.warning(self, '',
180                                 "error during camera initialisation",
181                                 QMessageBox.Ok)

```

```

183     else:
184         #pobranie nazwy kamery
185         self.nazwa_kamery = a[0].displayname
186
187         # stworzenie i podlaczenie wlasnych pyqt5 signal
188         self.nowy_obraz_z_kamery.connect(
189             self.nowy_obraz_z_kamery_sygnal)
190         self.nowy_wymuszony_obraz_z_kamery.connect(
191             self.nowy_wymuszony_obraz_z_kamery_sygnal)
192
193         # otwarcie komunikacji z kamera
194         try:
195             self.h_cam = tcam.Toupcam.Open(a[0].id)
196
197         except tcam.HRESULTException:
198             QMessageBox.warning(self, '',
199                                 'failed to open camera', QMessageBox.Ok)
200
201     else:
202         # stworzenie buforu
203         self.w, self.h = self.h_cam.get_Size()
204         self.buf = bytes(((self.w * 24 + 31) // 32 * 4)
205                           * self.h)
206
207         try:
208             #obsluzenie starszej wersji systemu widows
209             if sys.platform == 'win32':
210                 self.h_cam.put_Option(tcam.\
211                                     TOUPCAM_OPTION_BYTEORDER, 0)
212
213             #przekazanie kamerze metody do wywolywania
214             #i obiektu na ktorym ma byc wywolana
215             self.h_cam.StartPullModeWithCallback(
216                 self.metoda_wywolwana_przez_kamere, self)
217
218         except tcam.HRESULTException:
219             QMessageBox.warning(self, '',
220                                 'failed to start camera', QMessageBox.Ok)
221
222 #####Paint Event#####
223
224     def paintEvent(self, event):
225         ''
226
227         Metoda przeciazajaca paintEvent
228         z oznacz_ROI dodajaca obsluge
229         dodatkowych trybow rysowania
230         jest wywolywana automatycznie za
231         kazdym razem kiedy obiekt jest odswiezany
232         ''
233
234         # inicializacja qpintera
235         qp = QPainter(self)
236
237         # rysowanie obrazu
238         qp.drawPixmap(self.rect(), self._obraz_z_klatki)
239
240         # wgranie ustawien koloru i wypelnienia ROI
241         qp.setBrush(QBrush(QColor(200, 20, 20, 255),
242                           Qt.CrossPattern))
243

```

```

244     # zmienna określająca czy wyświetlamy numeracje czy nie
245     nasrysuj_opisy = True
246     narysuj_jeden_ROI = False
247
248     if self.co_narysowac == 'all_rectagls':
249
250         # pokazuje wszystkie prostokaty
251         self.wszystkie_prostokaty(qp)
252
253     elif self.co_narysowac == 'no_rectagle':
254
255         # chowa wszystkie prostokaty
256         nasrysuj_opisy = False
257
258     elif self.co_narysowac == 'One_rectagle':
259
260         # rysuje wybrany prostokąt
261         self._wybrany_prostokat(qp)
262         nasrysuj_opisy = False
263         narysuj_jeden_ROI = True
264
265     elif self.co_narysowac == 'viue_muve':
266
267         # rysuje wszystkie prostokaty
268         # i obsługuje odświeżenie podgladu.
269         self.wszystkie_prostokaty(qp)
270
271         self.odczytaj_klatke()
272
273     else:
274         # podstawowa opcja rysuje nowy prostokąt
275         self.wszystkie_prostokaty(qp)
276         qp.drawRect(QRect(self.poczatek, self.koniec))
277
278         self.zaladuj_obraz(nasrysuj_opisy, narysuj_jeden_ROI)
279
280     def _wybrany_prostokat(self, painter):
281         '''
282             metoda rysująca wybrany prostokąt
283         '''
284
285         painter.drawRect(self.narysuj_prostokat(
286                         self.main_window.ROI[self.ktory]))
287
288 #####funkcje wywoływanie przez guziki z głównego okna#####
289
290     def narysuj_calosc(self):
291         '''
292             podniesienie flagi odpowiedzialnej
293             za narysowanie wszystkich prostokątów
294         '''
295         self.co_narysowac = 'all_rectagls'
296
297     def schowajcalosc(self):
298         '''
299             podniesienie flagi odpowiedzialnej
300             za to żeby nie rysować żadnych prostokątów
301         :return:
302         '''
303         self.co_narysowac = 'no_rectagle'
304

```

```

305     def nastempny(self):
306         '''
307             podniesienie flagi i ustawienie
308             odpowiedniego iteratora
309             odpowiedzialnego za narysowanie
310             nastepnego prostokata
311         '''
312
313         if len(self.main_window.ROI) > 0:
314
315             self.co_narysowac = 'One_rectagle'
316
317             if self.ktory < len(self.main_window.ROI)-1:
318                 self.ktory += 1
319             else:
320                 self.ktory = 0
321
322         else:
323             pass
324
325     def poprzedni(self):
326         '''
327             podniesienie flagi i ustawienie odpowiedniego iteratora
328             odpowiedzialnego za narysowanie poprzedniego prostokata
329         '''
330
331         if len(self.main_window.ROI) > 0:
332             self.co_narysowac = 'One_rectagle'
333             self.ilosc_klikniec = True
334
335             if self.ktory == 0:
336                 self.ktory = len(self.main_window.ROI)-1
337             else:
338                 self.ktory -= 1
339
340             self.update()
341         else:
342             pass
343 ##### przesuwanie podgladu#####
344
345     def lewo(self):
346         '''
347             metoda obslugujaca przesuniecie podgladu przez manipulator
348         '''
349         self.ofsetx -= self.delta_pixeli
350         self._flagi_przemieszczenie()
351
352     def prawo(self):
353         '''
354             metoda obslugujaca przesuniecie podgladu przez manipulator
355         '''
356         self.ofsetx += self.delta_pixeli
357         self._flagi_przemieszczenie()
358
359     def dol(self):
360         '''
361             metoda obslugujaca przesuniecie podgladu przez manipulator
362         '''
363         self.ofsety -= self.delta_pixeli
364         self._flagi_przemieszczenie()

```

```

366     def gora(self):
367         '''
368             metoda obslugujaca przesuniecie podgladu przez manipulator
369         '''
370         self.ofsety += self.delta_pixeli
371         self._flagi_przemieszczenie()
372
373     def _flagi_przemieszczenie(self):
374         '''
375             metoda podnoszaca odpowiednia flage
376             oraz wylaczajaca tryb edycji jesli jest wlaczony
377         '''
378
379         if self.edit_tryb:
380             self.edited_roi.podglad.przyciski[0].toggle()
381             self.edited_roi.zakoncz_edit()
382             self.edit_trybe = False
383
384         self.co_narysowac = 'view_muve'
385         self.update()
386
387     def odswierz_ofsets(self):
388         '''
389             metoda obslugujaca zrownowazone polaczenie
390             startowe w zaleznosci od pozycji startowej manipulatora
391         '''
392         xm, ym, zm, = self.main_window.manipulaor.pobierz_pozycje_osi()
393
394         ym, zm = int((50-ym)*510), int((50-zm)*510)
395
396         self.ofsetx = ym
397         self.ofsety = zm
398
399 ##### edycja mapy #####
400
401     def zapisz_aktualny_podglad(self):
402         '''
403             metoda zapisujaca aktualna klatke do mapy
404             jesli mapa nie zostala jeszcze zdefiniowana
405             tworzy tablice do zbierania mapy
406             oraz tworzy obiekt wyswietlajacy mape jesli
407             nie zostal on jeszcze stworzony
408         '''
409
410         if self.first:
411             self._stworz_pojemnik_na_mape()
412
413         # wykonanie funkcji wklejajacej aktualny podglad do mapy
414         self.wklejenie_klatki_do_mapy()
415
416         if self.main_window.map is None:
417             self.main_window.map = Map_window(self.map, self.main_window)
418         else:
419             self.main_window.map.new_image(self.map)
420
421     def _stworz_pojemnik_na_mape(self):
422         '''
423             prywatna metoda tworzaca pojemnik na mapie
424         '''
425         x, y, z = self.klatka_2.shape

```

```

427     # x rozmiar
428     rozmiar_mapy = int((x + self.manipulator_max * self.delta_pixeli)
429                         / self.skala_mapy)
430     # y rozmiar
431     rozmiar_mapy *= int((y + self.manipulator_max * self.delta_pixeli)
432                           / self.skala_mapy)
433     rozmiar_mapy *= 3 # RGB kolory
434     # stworzenie tablicy przechowujacej obraz mapy
435     self.map = np.zeros(rozmiar_mapy, dtype=np.uint8)
436     # okreslenie ksztaltu tej tablicy
437     self.map.shape = (int((y + self.manipulator_max
438                           * self.delta_pixeli)/ self.skala_mapy),
439                       int((x + self.manipulator_max
440                           * self.delta_pixeli)/ self.skala_mapy), 3)
441     # zapisanie ze mapa juz jest zainicjowana
442     self.first = False
443
444 def wklejenie_klatki_do_mapy(self):
445     """
446         metoda zapisujaca do mapy aktualny podglad we wskazane
447         miejsce na ktorym znajduje sie manipulator
448     """
449     x, y, z = self.klatka_2.shape
450     xm, ym, zm, = self.main_window.manipulator.pobierz_pozycje_osi()
451
452     # przeliczenie milimetrow na pixele i odwrocenie osi
453     ym = int((50-ym)*510/self.skala_mapy)
454     zm = int((50-zm)*510/self.skala_mapy)
455
456     #przeskalowanie podgladu
457     klatka = cv2.resize(self.klatka_2, (int(y / self.skala_mapy),
458                                         int(x / self.skala_mapy)))
459     #wklejenie podgladu we wlosciwe miejsce na mapie
460     try:
461         self.map[zm:zm+int(x/self.skala_mapy),
462                  ym:ym+int(y/self.skala_mapy)] = klatka
463     except Exception as e:
464         print(e)
465
466 def reset_map(self):
467     """
468         metoda czyszczaca zebiana aktualnie mape
469     """
470     # zapytanie uzytkownika czy napewno chce to zrobic
471     reply = QMessageBox.question(self, "mesage",
472                                 "Czy napewno chcesz usunac mapę?", QMessageBox.Yes | QMessageBox.No,
473                                 QMessageBox.No)
474
475     if reply == QMessageBox.Yes:
476         self.first = True
477         self.zapisz_aktualny_podglad()
478
479
480 def pobierz_map(self):
481     return self.map
482
483 def _map_update(self):
484     self.co_narysowac = 'view_muve'
485     self.update()

```

Kod źródłowy 5: Wyswietlenie mapy.

```
1 from PyQt5.QtWidgets import QWidget, QVBoxLayout
2 from PyQt5 import QtGui
3 from PyQt5.QtGui import QImage, QPixmap
4 from PyQt5.QtCore import Qt
5 from roi_create import oznacz_ROI
6
7 class Map(oznacz_ROI):
8
9     def __init__(self, obraz, main_window, *args, **kwargs):
10         super(Map, self).__init__(main_window, *args, **kwargs)
11         #skalowanie nowego obrazu
12         self.image_opencv = obraz
13
14         #konwersja obrazu z openCV na QImage
15         self._obraz_z_klatki = QImage(obraz, obraz.shape[1],
16                                       obraz.shape[0],
17                                       obraz.strides[0],
18                                       QImage.Format_RGB888)
19         self._obraz_z_klatki = QPixmap.fromImage(self._obraz_z_klatki)
20
21         #seting pixmap
22         self.setPixmap(self._obraz_z_klatki)
23
24     def new_image(self, img):
25         '''
26             Metoda przyjmujaca nowy obraz z open CV
27             konwertujaca go na QImage i zapisujaca
28             :param img: open CV obraz
29         '''
30         self._obraz_z_klatki = QImage(img, img.shape[1], img.shape[0],
31                                       img.strides[0], QImage.Format_RGB888)
32         #zapisanie obrazu
33         self._obraz_z_klatki = QPixmap.fromImage(self._obraz_z_klatki)
34
35 class Map_window(QWidget):
36
37     def __init__(self, map, main_window, *args, **kwargs):
38         super(Map_window, self).__init__(*args, **kwargs)
39         self.setWindowTitle("Mapa Probki")
40         self.setWindowIcon(QtGui.QIcon('icon.png'))
41
42         #stworzenie layoutu
43         self.layout = QVBoxLayout()
44
45         #map obiekt
46         self.map = Map(map, main_window)
47
48         #dodanie mapy do podgladu
49         self.layout.addWidget(self.map, Qt.AlignCenter)
50
51         #ustawienie layoutu
52         self.setLayout(self.layout)
53
54     def new_image(self, img):
55         '''
56             Metoda przekszukujaca nowa mape do ROI'u
57             :param img: nowa mapa klasy openCV
58         '''
59         self.map.new_image(img)
```

Kod źródłowy 6: Klasa przechowująca ROI.

```
1 from PyQt5.QtCore import QPoint, QRect
2 from PyQt5.QtGui import QImage, QPixmap
3 from kwadrat_label import Podglad_ROI
4
5
6 class Obszar_zaznaczony:
7     '''
8         klasa przechowujaca wspolrzedne probki
9         w pixelach i mm oraz podglad
10        '''
11
12     # wspolrzedne bezwzgledne w geometrii probki w pixelach
13     x0 = 0
14     y0 = 0
15     x1 = 1
16     y1 = 1
17
18     # wspolrzedne bezwzgledne w geometrii probki w mm
19     xm0 = 0
20     ym0 = 0
21     xm1 = 0
22     ym1 = 0
23
24     #wartosci binarne przechowujace flagi do edycji
25     pierwsze_klikniecie = False
26
27     kanta_gora = False
28     kanta_dol = False
29     kanta_lewa = False
30     kanta_prawy = False
31
32     lewa_gora = False
33     prawa_gora = False
34     lewy_dol = False
35     prawy_dol = False
36
37     przemiesc_wszystkie = False
38
39     szerokosc_obszaru_klikniecia = 10
40
41     stala_przemieszczenia = 10
42
43     # konstruktor tworzacy obiekt ze wzglednych
44     # wspolrzednych probki w pixelach
45     def __init__(self, obraz_obiekt, xp0, yp0, xp1, yp1,
46                  image, px0=0, py0=0,
47                  nazwa="defalaut", s=1):
48
49         #Nadrzedny obiekt (map lub kamera) w ktojrym strworzono
50         #obiekt przekazanoy w celu komunikacji
51         self.obraz_obiekt = obraz_obiekt
52
53         #nazwa obiektu
54         self.nazwa = nazwa
55
56         #metoda tworzaca niezalezne wspolrzedne probki w pixelach
57         self.x0, self.x1, self.y0, self.y1 = self.\
58                                         _zalezne_to_niezalezne(xp0, yp0,
59                                         xp1, yp1,
60                                         px0, py0, s)
```

```

61
62     #metoda tworzaca widget umozliwiajacy interakcje z obiektem
63     self._stworz_podglad_ROI(image)
64
65     #metoda tworzaca niezalezne wspolrzedne probki w mm
66     self._ustaw_niezalezne_probki()
67
68     def __del__(self):
69         """
70             dekonstruktor klasy obsluguje usuniecie podgladu
71             i wyloczenie edycji
72         """
73
74         self.obraz_objekt.rmv_rectagle(self)
75         try:
76             self.zakoncz_edit()
77             del self.podglad
78         except AttributeError:
79             pass
80
81     def usun(self):
82
83         self.obraz_objekt.rmv_rectagle(self)
84
85         try:
86             self.zakoncz_edit()
87             del self.podglad
88         except AttributeError:
89             pass
90
91
92     def pobierz_wzgledny_rectagle(self):
93         """
94             metoda zwracajaca aktualne wzgledne wspolrzedne
95             roiu w pixelach
96             :return: x0, y0, x1, y1
97         """
98         x0 = (self.x0 - self.obraz_objekt.ofsetx)
99         y0 = (self.y0 - self.obraz_objekt.ofsety)
100
101        x1 = (self.x1 - self.obraz_objekt.ofsetx)
102        y1 = (self.y1 - self.obraz_objekt.ofsety)
103
104        return x0, y0, x1, y1
105
106    def pobierz_niezalezne_pixele(self):
107        """
108            metoda zwracajaca niezalezne wspolrzedne w pixelach
109            :return: x0, x1, y0, y1
110        """
111        return self.x0, self.x1, self.y0, self.y1
112
113    def _ustaw_niezalezne_probki(self):
114        """
115            metoda konwertujaca wspolrzedne bezwzgledne w pixelach
116            na wspolrzedne probkki
117        """
118        self.xm0 = self.x0/510
119        self.ym0 = self.y0/510
120        self.xm1 = self.x1/510
121        self.ym1 = self.y1/510

```

```

122
123     def _niezalezne_to_zalezne(self, px00, py00, s):
124         """
125             metoda konwertujaca wspolrzedne niezalezne
126             na zalezne
127             w pixelach na zalezne
128             :param px00: ofset x
129             :param py00: ofset y
130             :param s: skala
131             :return: x0, x1, y0, y1
132         """
133         return self._konwersja(self.x0, self.y0, self.x1, self.y1,
134                               -px00, -py00, s)
135
136     def _zalezne_to_niezalezne(self, xp0, yp0, xp1, yp1,
137                               px00, py00, s):
138         """
139             metoda konwertujaca wspolrzedne zalezne
140             na niezalezne
141             :param xp0:
142             :param yp0:
143             :param xp1:
144             :param yp1:
145             :param px00: ofset x
146             :param py00: ofset y
147             :param s: skala
148             :return: x0, x1, y0, y1
149         """
150         return self._konwersja(xp0,yp0,xp1,yp1,px00,py00,s)
151
152     def _konwersja(self, xp0, yp0, xp1, yp1, px00, py00, s):
153         """
154             metoda konwertujaca wspolrzedne
155             :param xp0:
156             :param yp0:
157             :param xp1:
158             :param yp1:
159             :param px00: ofset x
160             :param py00: ofset y
161             :param s: skala
162             :return: x0, x1, y0, y1
163         """
164         x0 = int((xp0 + px00)/s)
165         y0 = int((yp0 + py00)/s)
166         x1 = int((xp1 + px00)/s)
167         y1 = int((yp1 + py00)/s)
168
169         return x0, x1, y0, y1
170
171     def pobierz_prostokat(self, px00, py00, s):
172         """
173             metoda zwracajaca prostokat w ukladzie
174             aktualnie wyswietlonym
175             :param px00: ofsetx
176             :param py00: ofsety
177             :param s: skala
178             :return: QRectagle
179         """
180         xp0, xp1, yp0, yp1 = self._niezalezne_to_zalezne(px00,py00,1/s)
181
182         return QRect(QPoint(xp0, yp0), QPoint(xp1, yp1))

```

```

183
184     def ustaw_nazwe(self, nazwa):
185         """
186             metoda umozliwiajaca nazwanie obiektu
187             :param nazwa: string
188         """
189         self.nazwa = nazwa
190
191     def pobierz_nazwe(self):
192         """
193             metoda zwracajaca nazwe obiektu
194             :return: string
195         """
196         return self.nazwa
197
198     def _pobierz_gorny_naroznik(self, ox, oy):
199         """
200             metoda zwracajaca najwyszy prawy naroznik obiektu
201             :param ox: ofset x
202             :param oy: ofset y
203             :return: x, y
204         """
205
206         if self.x0 < self.x1 and self.y0 < self.y1:
207             xp0 = (self.x0 - ox)
208             yp0 = (self.y0 - oy)
209         elif self.x0 < self.x1 and self.y0 > self.y1:
210             xp0 = (self.x0 - ox)
211             yp0 = (self.y1 - oy)
212         elif self.x0 > self.x1 and self.y0 < self.y1:
213             xp0 = (self.x1 - ox)
214             yp0 = (self.y0 - oy)
215         else:
216             xp0 = (self.x1 - ox)
217             yp0 = (self.y1 - oy)
218
219         return xp0, yp0
220
221     def pobierz_lokacje_tekstu(self, ox, oy):
222         """
223             metoda zwracajaca lokacje nazwy wyswietlanej na podgladzie
224             :param ox: ofset x
225             :param oy: ofset y
226             :return: x, y
227         """
228         xp0, yp0 = self._pobierz_gorny_naroznik(ox, oy)
229
230         return xp0 - 20, yp0 - 10
231
232     def _stworz_podglad_ROI(self, obraz):
233         """
234             Prywatna metoda tworzaca etykiete ROI'u
235             wyswietlana na podgladzie
236             :param obraz: klatka z podgladu
237         """
238
239         obraz_qimage = QImage(obraz,
240                               obraz.shape[1],
241                               obraz.shape[0],
242                               obraz.strides[0],
243                               QImage.Format_RGB888)

```

```

244     self.obraz = QPixmap.fromImage(obraz_qimage)
245
246     self.podglad = Podglad_ROI(str(self.nazwa),
247                                 self.obraz, self)
248
249
250     def pobierz_obraz(self):
251         """
252             metoda zwracajaca obraz
253             :return: obraz zawierajacy podglad probki
254             zrobiony w momencie tworzenia
255         """
256
257         return self.obraz
258
259     def pobierz_podglad(self):
260         """
261             metoda zwracajaca widget podgladu ROI
262             :return: obiekt klasy ROI stowarzyszony z ROI,em
263         """
264
265         return self.podglad
266
267 #####edycja obiektu za pomoca strzalek#####
268
269     def przekstalc_gorna_linie(self, mode):
270         """
271             metoda wykonujaca edycje jednej z krawedzi o stala wartosc
272             :param mode: jestli True to zwiększymy wymiar jeśli false
273             to zmniejszymy
274         """
275
276         if mode:
277             self.y0 += self.stala_przemieszczenia
278         else:
279             self.y0 -= self.stala_przemieszczenia
280
281     def przekstalc_dolna_linie(self, mode):
282         """
283             metoda wykonujaca edycje jednej z krawedzi o stala wartosc
284             :param mode: jestli True to zwiększymy wymiar jeśli
285             false to zmniejszymy
286         """
287
288         if mode:
289             self.y1 -= self.stala_przemieszczenia
290         else:
291             self.y1 += self.stala_przemieszczenia
292
293     def przekstalc_lewa_linie(self, mode):
294         """
295             metoda wykonujaca edycje jednej z krawedzi o stala wartosc
296             :param mode: jestli True to zwiększymy wymiar jeśli
297             false to zmniejszymy
298         """
299
300         if mode:
301             self.x0 += self.stala_przemieszczenia
302         else:
303             self.x0 -= self.stala_przemieszczenia

```

```

305     def przekstalc_prawa_linie(self, mode):
306         """
307             metoda wykonujaca edycje jednej z krawedzi o stala wartosc
308             :param mode: jestli True to zwiększymy wymiar jeśli
309             False to zmniejszamy
310         """
311         if mode:
312             self.x1 -= self.stala_przemieszczenia
313         else:
314             self.x1 += self.stala_przemieszczenia
315
316     def przesun_w_gore(self):
317         """
318             metoda przesuwajaca podglad w jednym z kierunkow
319         """
320         self.y0 -= self.stala_przemieszczenia
321         self.y1 -= self.stala_przemieszczenia
322
323     def przesun_w_dol(self):
324         """
325             metoda przesuwajaca podglad w jednym z kierunkow
326         """
327         self.y0 += self.stala_przemieszczenia
328         self.y1 += self.stala_przemieszczenia
329
330     def przesun_w_lewo(self):
331         """
332             metoda przesuwajaca podglad w jednym z kierunkow
333         """
334         self.x0 -= self.stala_przemieszczenia
335         self.x1 -= self.stala_przemieszczenia
336
337     def przesun_w_prawo(self):
338         """
339             metoda przesuwajaca podglad w jednym z kierunkow
340         """
341         self.x0 += self.stala_przemieszczenia
342         self.x1 += self.stala_przemieszczenia
343
344 #####odbieranie i wysylanie flagi edycji#####
345
346     def edit(self):
347         """
348             metoda wlaczajaca tryb edycji
349         """
350         self.obraz_objekt.edit_roi(self)
351
352     def zakoncz_edit(self):
353         """
354             metoda konczaca tryb edycji
355         """
356
357             #zakonczenie trybu edycji w obrazie i pobranie
358             #z niego nowej klatki podgladu
359             klatka = self.obraz_objekt.zakoncz_edit()
360
361             #konwersja klatki na QImage
362             klatka_qimage = QImage(klatka, klatka.shape[1],
363                                     klatka.shape[0],
364                                     klatka.strides[0],
365                                     QImage.Format_RGB888)

```

```

366
367     #zapisanie klatki
368     self.obraz = QPixmap.fromImage(klatka_qimage)
369
370     #odswiezenei klatki w podgladzie
371     self.podglad.nowy_obraz(self.obraz)
372
373 ##### samo edycha#####
374
375     def wspolrzedne_nacisniecia(self, e, ofsetx, ofsety):
376         ''
377         Metoda wywolywana w trybie edycji przy nacisnieciu
378         przycisku myszki
379         :param e: wspolrzedne w ktorych myszka zostala nacisneta
380         :param ofsetx:
381         :param ofsety:
382         ''
383
384         # restart paramterow edycji
385         self.pierwsze_kliknienie = True
386
387         self.kanta_gora = False
388         self.kanta_dol = False
389         self.kanta_lewa = False
390         self.kanta_prawy = False
391
392         self.przemiesc_wszystkie = False
393
394         # odczyt i konwersja wspolrzednych kliknienia
395         self.px0, self.py0 = e.x() + ofsetx, e.y() + ofsety
396
397         # rozpoznanie kant koro ktorych doszlo do kliknienia
398         if self.x0 - self.szerokosc_obszaru_kliknienia < self.px0 < \
399             self.x0 + self.szerokosc_obszaru_kliknienia and \
400             self.y0 - self.szerokosc_obszaru_kliknienia < self.py0 < \
401             self.y0 + self.szerokosc_obszaru_kliknienia:
402
403             self.kanta_prawy = True
404
405         if self.x1 - self.szerokosc_obszaru_kliknienia < self.px0 < \
406             self.x1 + self.szerokosc_obszaru_kliknienia and \
407             self.y0 - self.szerokosc_obszaru_kliknienia < self.py0 < \
408             self.y1 + self.szerokosc_obszaru_kliknienia:
409
410             self.kanta_lewa = True
411
412         if self.y0 - self.szerokosc_obszaru_kliknienia < self.py0 < \
413             self.y0 + self.szerokosc_obszaru_kliknienia and \
414             self.x0 - self.szerokosc_obszaru_kliknienia < self.px0 < \
415             self.x1 + self.szerokosc_obszaru_kliknienia:
416
417             self.kanta_gora = True
418
419         if self.y1 - self.szerokosc_obszaru_kliknienia < self.py0 < \
420             self.y1 + self.szerokosc_obszaru_kliknienia and \
421             self.x0 - self.szerokosc_obszaru_kliknienia < self.px0 < \
422             self.x1 + self.szerokosc_obszaru_kliknienia:
423
424             self.kanta_dol = True
425
426

```

```

427     #sprawdzanie czy nie spełniamy warunków które goś z rogo
428     self.lewa_gora = self.kanta_lewa and self.kanta_gora
429     self.prawa_gora = self.kanta_prawy and self.kanta_gora
430
431     self.lewy_dol = self.kanta_lewa and self.kanta_dol
432     self.prawy_dol = self.kanta_prawy and self.kanta_dol
433
434     #sprawdzenie czy nie kliknęliśmy w środek obszaru
435     if self.y0+self.szerokosc_obszaru_klikniecia < self.py0 < \
436         self.y1-self.szerokosc_obszaru_klikniecia and \
437         self.x0+self.szerokosc_obszaru_klikniecia < self.px0 < \
438         self.x1-self.szerokosc_obszaru_klikniecia:
439
440         self.przemiesc_wszystkie = True
441
442     # odświeżenie kordynatów na podglądzie
443     self.podglad.odświerz_kordynaty()
444
445 def wspolrzedne_puszczenia(self, e, ofsetx, ofsety):
446     '''
447     metoda wykonywana po puszczeniu przycisku myszki
448     zapisuje wykonana edycje ROI'u
449     :param e: pozycja myszki
450     :param ofsetx:
451     :param ofsety:
452     '''
453
454     # reset licznika klikniec
455     self.pierwsze_klikniecie = False
456
457     # odczyt i konwersja współrzędnych kliknięcia
458     self.px1, self.py1 = e.x() + ofsetx, e.y() + ofsety
459
460     # sprawdzenie trybu pracy i adekwatna do niego edycja ROI'u
461     if self.przemiesc_wszystkie:
462         dx, dy = self.px1 - self.px0, self.py1 - self.py0
463         self.x0 += dx
464         self.x1 += dx
465         self.y0 += dy
466         self.y1 += dy
467
468     elif self.lewa_gora:
469         self.x1 = self.px1
470         self.y0 = self.py1
471
472     elif self.lewy_dol:
473         self.x1 = self.px1
474         self.y1 = self.py1
475
476     elif self.prawy_dol:
477         self.x0 = self.px1
478         self.y1 = self.py1
479
480     elif self.prawa_gora:
481         self.x0 = self.px1
482         self.y0 = self.py1
483
484     elif self.kanta_dol:
485         self.y1 = self.py1

```

```

488     elif self.kanta_lewa:
489         self.x1 = self.px1
490
491     elif self.kanta_prawy:
492         self.x0 = self.px1
493
494     elif self.kanta_gora:
495         self.y0 = self.py1
496
497     # odswierzenie kordynatow na podgladzie
498     self.podglad.odswierz_kordynaty()
499
500 def przemiesc_kordynaty(self, e, ofsetx, ofsety):
501
502     # sprawdzenie licznika klikniec
503     if self.pierwsze_kliknienie:
504
505         # odczyt i konwersja wspolrzednych kliknienia
506         self.px1, self.py1 = e.x() + ofsetx, e.y() + ofsety
507
508         # sprawdzenie trybu pracy i adekwatne obsluzenie
509         # edycji dla niego
510         if self.przemiesc_wszystkie:
511             dx, dy = self.px1 - self.px0, self.py1 - self.py0
512             self.x0 += dx
513             self.x1 += dx
514             self.y0 += dy
515             self.y1 += dy
516             self.px0, self.py0 = e.x() + ofsetx, e.y() + ofsety
517
518         elif self.lewa_gora:
519             self.x1 = self.px1
520             self.y0 = self.py1
521
522         elif self.lewy_dol:
523             self.x1 = self.px1
524             self.y1 = self.py1
525
526         elif self.prawy_dol:
527             self.x0 = self.px1
528             self.y1 = self.py1
529
530         elif self.prawa_gora:
531             self.x0 = self.px1
532             self.y0 = self.py1
533
534         elif self.kanta_dol:
535             self.y1 = self.py1
536         elif self.kanta_lewa:
537             self.x1 = self.px1
538         elif self.kanta_prawy:
539             self.x0 = self.px1
540         elif self.kanta_gora:
541             self.y0 = self.py1
542
543     # odswierzenie kordynatow na podgladzie
544     self.podglad.odswierz_kordynaty()

```

Kod źródłowy 7: Klasa odpowiadająca za etykiety ROI.

```
1 from PyQt5.QtWidgets import QLabel, QWidget, QLineEdit, \
2     QHBoxLayout, QVBoxLayout, QGridLayout, QPushButton
3 from PyQt5.QtCore import QRect, QPoint
4 from PyQt5.QtGui import QPainter, QBrush, QColor, QPixmap
5
6
7 class Prosty_Podglad(QLabel):
8
9     """
10     Obiekt dziedziczący z QLabel specjalnie przystosowany
11     do wyświetlania
12     miniaturowej podglądu obszaru oznaczonego
13     """
14
15     def __init__(self, obraz, wspolrzedne, *args, **kwargs):
16         super(Prosty_Podglad, self).__init__(*args, **kwargs)
17
18         # Zapisanie otrzymanych nie skalibrowanych współrzędnych
19         self.wspolrzedne = wspolrzedne
20
21         # Metoda kalibrująca prostokąt
22         self._prostkat = self._stworz_prostokat(wspolrzedne)
23
24         # Metoda wgrzewająca podgląd
25         self._wgraj(obraz)
26
27     def _stworz_prostokat(self, wspolrzedne):
28
29         """
30         Prywatna metoda kalibrująca otrzymane współrzędne
31         i zwracająca stworzony na ich podstawie prostokąt
32         :param wspolrzedne: współrzędne ROI'u (x0,y0,x1,y1)
33         :return: QRectagle stworzony z przekazanych współrzędnych
34         """
35
36         skalibrowane_wspolrzedne = self._kalibracja_wspolrzednych(
37                                         wspolrzedne)
38
39         poczatkowy_naroznik = QPoint(skalibrowane_wspolrzedne[0],
40                                       skalibrowane_wspolrzedne[1])
41
42         koncowy_naroznik = QPoint(skalibrowane_wspolrzedne[2],
43                                     skalibrowane_wspolrzedne[3])
44
45         return QRect(poczatkowy_naroznik, koncowy_naroznik)
46
47     def _kalibracja_wspolrzednych(self, rect):
48
49         """
50         Prywatna metoda kalibrująca przechwycone współrzędne
51         ROI do przeskalowanego obrazu
52         :param rect: lista zawierająca współrzędne (x0,y0,x1,y1)
53         :return: skalibrowana lista współrzędnych ROI
54         """
55
56         # Wielkości umożliwiające skalowanie ROI'u
57         kalibracja = [6.36, 6.72, 6.33, 6.53]
58
59         # Kalibracja przekazanego ROI
60         return [r / a for r, a in zip(rect, kalibracja)]
```

```

61
62     def _wgraj(self, obraz):
63         """
64             Prywatna metoda wgrywajaca przekazaony obraz
65             :param obraz: obiekt klasy Qpixmap
66             :return:
67             """
68
69         #zapisanie obrazu
70         self._obraz = obraz
71
72         #wgranie obrazu
73         self.setPixmap(self._obraz)
74
75     def update_rectagle(self, kalibracja):
76         """
77             Metoda umozliwiajaca wykonanie kalibracji wielkosci podgladu
78             :param kalibracja: tablica kalibracji
79             """
80
81         #skalibrowanie podgladu
82         Rect = [r / a for r, a in zip(self.wspolrzedne, kalibracja)]
83
84         #nadpisanie prostokata oznaczonego nowo skalibrowanym
85         self.rectangle = QRect(QPoint(Rect[0], Rect[1]),
86                               QPoint(Rect[2], Rect[3]))
87
88     def paintEvent(self, QPaintEvent):
89         """
90             Metoda nadpisujaca wbudowana metode QPinter z PyQt5
91             wgrywajaca obraz oraz rysujaca na nim oznaczony ROI
92             :param QPaintEvent: obiekt klasy event
93             :return:
94             """
95
96         #inicjalizacja pintera
97         qp = QPainter(self)
98
99         #rysowanie obrazu
100        qp.drawPixmap(self.rect(), self._obraz)
101
102        #stworzenie i wgranie stylu prostokata
103        qp.setBrush(QBrush(QColor(200, 10, 10, 200)))
104
105        #rysowanie prostokatu
106        qp.drawRect(self._prostkat)
107
108    def nowy_podglad(self, obraz, wspolrzedne):
109        """
110            metoda odswierzajaca podglad
111            :param obraz: nowy podglad
112            :param wspolrzedne: nowe wspolrzedne ROI
113            """
114
115        self._wgraj(obraz)
116
117        self.wspolrzedne = wspolrzedne
118
119        self._prostkat = self._stworz_prostokat(wspolrzedne)
120
121

```

```

122 class Podglad_ROI(QWidget):
123     """
124     Obiekt przeciazajacy obiekt klasy QWidget
125     Przedstawia on oznaczony ROI oraz umozliwia interakcje z nim
126     nadanie mu nazwy edycje lub usuniecie
127     """
128
129
130     def __init__(self, tekst, obraz, obiekt_oznaczony, *args, **kwargs):
131         super(QWidget, self).__init__(*args, **kwargs)
132
133         # obiekt klasy Qpiuxmap zawierajacy aktualny podglad
134         # na ktorym oznaczono ROI
135         self.obraz = obraz
136
137         # wskaznik do obiekt klasy obszar oznaczony
138         self.obiekt_oznaczony = obiekt_oznaczony
139
140 #####
141 ##### Przyciski i labele#####
142 #####
143
144         # widget zawierajacy podglad na oznaczony ROI
145         self.podglad = Prosty_Podglad(self.obraz, obiekt_oznaczony.\
146                                         pobierz_wzgledny_rectagle())
147
148         #widget zawierajacy nazwe ROI oraz umozliwiajacy jego edycje
149         self.name_lable = QLineEdit(tekst)
150         self.name_lable.textChanged.connect(self.nowa_nazwa)
151
152         # stworzenie opisu wspolrzednych ROI
153         self._stworz_legende()
154
155 #####
156 ##### Layout#####
157 #####
158
159         self._stworzenie_layoutow()
160
161         self._wstawienie_widgetow_do_layoutow()
162
163         self._polacz_layouty()
164
165         # ustawienie glownego layoutu
166         self.setLayout(self._glowny_layout)
167
168         # stworzenie przyciskow
169         self.glowne_przyciski()
170
171         # zablokowanie rozmiaru
172         self.zablokuj_rozmiar()
173
174     def __del__(self):
175         """
176             dekonstruktor klasy usuwa on oznaczony obiekt
177         """
178         try:
179             self.obiekt_oznaczony.usun()
180             self.obiekt_oznaczony = 0
181         except AttributeError:
182             pass

```

```

183
184     def _usun(self):
185         """
186             Metoda umozliwiajaca usuniecie obiektu uzywajac przycisku
187             """
188
189         try:
190             self.obiekt_oznaczony.usun()
191             self.obiekt_oznaczony = 0
192         except AttributeError:
193             pass
194
195     def _stworzenie_layoutow(self):
196         """
197             Prywatna metoda tworzaca layaouty
198             """
199
200         self._przyciski_layout = QBoxLayout()
201
202         self._legenda_layout = QBoxLayout()
203
204         self._pole_layout = QBoxLayout()
205
206         self._kierunkowe_layout = QGridLayout()
207
208         self._glowny_layout = QVBoxLayout()
209
210         self._drugorzedy_layout = QVBoxLayout()
211
212     def _wstawienie_widgetow_do_layoutow(self):
213         """
214             Prywatna metoda wstawiajaca widgety do layoutow
215             """
216
217         self._legenda_layout.addWidget(self.x_label)
218         self._legenda_layout.addWidget(self.x0_label)
219         self._legenda_layout.addWidget(self.x1_label)
220
221         self._legenda_layout.addWidget(self.y_label)
222         self._legenda_layout.addWidget(self.y0_label)
223         self._legenda_layout.addWidget(self.y1_label)
224
225         self._pole_layout.addWidget(self.pole_label)
226         self._pole_layout.addWidget(self.poleL)
227
228         self._drugorzedy_layout.addWidget(self.name_label)
229
230         self._glowny_layout.addWidget(self.podglad)
231
232     def _polacz_layouty(self):
233         """
234             Prywatna metoda laczaca layouty
235             """
236
237         self._drugorzedy_layout.setLayout(self._legenda_layout)
238         self._drugorzedy_layout.setLayout(self._pole_layout)
239         self._drugorzedy_layout.setLayout(self._przyciski_layout)
240
241         self._glowny_layout.setLayout(self._drugorzedy_layout)
242
243         self._glowny_layout.setLayout(self._kierunkowe_layout)

```

```

244
245
246     def zablokuj_rozmiar(self, x=180, y=225):
247         """
248             metoda zablokowujaca rozmiar widgetu
249             w celu zachowania poprawnosci rozmiaru podgladu
250             :param x: szerokosc widgetu
251             :param y: wysokosc widgetu
252         """
253
254         self.setMaximumSize(x, y)
255         self.setMinimumSize(x, y)
256
257
258     def _stworz_legende(self):
259         """
260             prywatna metoda tworzoaca legende zawierajaca wspolrzedne
261             ROI'u oraz jego pole
262         """
263
264         # odczytanie wspolrzednych bezwzglednych
265         # ROI w celu wykonaniu opisu
266         x, x1, y, y1 = self.obiekt_oznaczony.pobierz_niezalezne_pixele()
267
268         #obliczenie pola ROI'u w pixelach
269         p_pixele = self.pole(x, x1, y, y1)
270
271         self.pole_label = QLabel("Pole obszaru:")
272
273         self.poleL = QLabel(str(p_pixele))
274
275         self.x_label = QLabel("x")
276
277         self.x0_label = QLabel(str(x))
278         self.x1_label = QLabel(str(x1))
279
280         self.y_label = QLabel("y")
281
282         self.y0_label = QLabel(str(y))
283         self.y1_label = QLabel(str(y1))
284
285
286     @staticmethod
287     def przycisk(fun, text, clicable=False):
288         """
289             Statyczna metoda tworzaca przycisk o zadanych parametrach
290             :param fun: funkcja wywolywana przy nacisnieiu przycisku
291             :param text: napis na przycisku
292             :param clicable: opcja umozliwiajaca zatrzymania stanu
293             przycisku
294             :return:
295         """
296
297         przycisk = QPushButton()
298         przycisk.setMaximumWidth(50)
299         przycisk.setText(text)
300         przycisk.setCheckable(clicable)
301         przycisk.clicked.connect(fun)
302
303         return przycisk
304

```

```

305 ##### konfiguracja przyciskow #####
306 ##### konfiguracja przyciskow #####
307 ##### konfiguracja przyciskow #####
308
309     def glowne_przyciski(self):
310         '''
311             Metoda tworzaca glowne przyciski
312         '''
313
314         self.przyciski = [QPushButton() for _ in range(3)]
315
316         [self._przyciski_layout.addWidget(wartosc) for wartosc
317          in self.przyciski]
318
319         nazwy = ["edit", "fine edit", "del"]
320
321         [przycisk.setText(nazwa) for nazwa, przycisk
322          in zip(nazwy, self.przyciski)]
323
324         funkcje = [self.edit, self.fine_edit, self._usun]
325
326         [b.clicked.connect(f) for b, f in zip(self.przyciski, funkcje)]
327
328         # ustawienie przycisku edycji jako przelacznika
329         # miedzy trybem normalnym a trybem edycji
330         self.przyciski[0].setCheckable(True)
331
332     def przyciski_kalibracyjne(self):
333         '''
334             Metoda umieszcza jaca przyciski
335             umoziwi jace kalibracje podgladu
336         '''
337
338         self.przyciski = [QPushButton() for _ in range(8)]
339
340         [self._przyciski_layout.addWidget(wartosc) for wartosc
341          in self.przyciski]
342
343         nazwykalibracionmode = ['xp', 'yp', 'zp', 'sp',
344                                'xm', 'ym', 'zm', 'sm']
345
346         [switch.setText(name) for name, switch in
347          zip(nazwykalibracionmode, self.przyciski)]
348
349         self.x, self.y, self.z, self.s = 6.36, 6.72, 6.33, 6.53
350
351         funkikalibracionmode = [self.xp, self.yp, self.zp, self.sp,
352                                self.xm, self.ym, self.zm, self.sm]
353
354         [b.clicked.connect(f) for b, f in
355          zip(self.przyciski, funkikalibracionmode)]
356
357     def _fine_edit_buttons(self):
358         '''
359             metoda tworzaca przyciski do edycji
360         '''
361
362         self.kierunkowe = [QPushButton() for _ in range(4)]
363
364         [button.setMaximumWidth(50) for button in self.kierunkowe]

```

```

366
367     # nazwy dla przyciskow
368     nazwy = ['/\\\', "<", ">", '\\/']
369     [switch.setText(name) for name, switch
370      in zip(nazwy, self.kierunkowe)]
371
372     # przypiecie funkcji do przyciskow
373     funkcje = [self.gorny_przycisk, self.lewy_przycisk,
374                 self.prawy_przycisk, self.dolny_przycisk]
375
376     [przycisk.clicked.connect(funkcja) for funkcja, przycisk
377      in zip(funkcje, self.kierunkowe)]
378
379     # dodanie do layoutow przyciskow kierunkowych
380     it = [3, 2, 4, 3]
381     jt = [2, 3, 3, 4]
382     [self._kierunkowe_layout.addWidget(value, j, i)
383      for j, i, value in zip(jt, it, self.kierunkowe)]
384
385     # Przywrocenie standardowych przyciskow
386     self.powrot = self.przycisk(self.retyurn_to_normalbuttons,
387                                  "return")
388     self._kierunkowe_layout.addWidget(self.powrot, 2, 2)
389     self.kierunkowe.append(self.powrot)
390
391     #przelaczanie pomiedzy ruszaniem ROI'em a jego rozszerzaniem
392     self.przemiesc = self.przycisk(self.przelacz_przemiesc,
393                                     "przemiesc", True)
394
395     self._kierunkowe_layout.addWidget(self.przemiesc, 2, 4)
396     self.kierunkowe.append(self.przemiesc)
397
398     #przelaczanie pomiedzy kierunkiem zmian
399     self.zwieksz = self.przycisk(self.przelacz_zwieksz, "+", True)
400
401     self._kierunkowe_layout.addWidget(self.zwieksz, 4, 4)
402     self.kierunkowe.append(self.zwieksz)
403
404
405     def usun_fine_edit(self):
406         '''
407             metoda usuwajaca przyciski do edycji
408         '''
409
410         [self._kierunkowe_layout.removeWidget(b) for b in self.kierunkowe]
411         self.kierunkowe = 0
412         self.przemiesc = 0
413         self.powrot = 0
414         self.zwieksz = 0
415
416     def usun_glowne_przyciski(self):
417         '''
418             metoda usuwajaca glowne przyciski
419         :return:
420         '''
421
422         [self._przyciski_layout.removeWidget(b) for b in self.przyciski]
423         self.przyciski = 0
424
425
426

```

```

427 ##### Oznaczony komunicacia #####
428 ##### Obiekt oznaczony #####
429 #####
430     def nowa_nazwa(self):
431         """
432             metoda nadajaca nowa nazwe na glownym widoku roi'u
433         """
434         self.obiekt_oznaczony.ustaw_nazwe(self.name_label.text())
435
436
437     def odswierz_kordynaty(self):
438         """
439             metoda odswiezajaca na biezaco
440             koordynaty wyswietlane na obiekcie
441         """
442
443         self.podglad._wgraj(self.obraz)
444
445         x, x1, y, y1 = self.obiekt_oznaczony.pobierz_niezalezne_pixele()
446
447
448         self.x0_label.setText(str(x))
449         self.x1_label.setText(str(x1))
450
451         self.y0_label.setText(str(y))
452         self.y1_label.setText(str(y1))
453
454         self.poleL.setText(str(self.pole(x, x1, y, y1)))
455
456     def pole(self, x, x1, y, y1):
457         """
458             funkcja liczaca pole obszaru oznaczonego
459             :param x: pierwsza wspolrzedna x
460             :param x1: druga wspolrzedna x
461             :param y: pierwsza wspolrzedna y
462             :param y1: druga wspolrzedna y
463             :return: pole
464         """
465
466         xm = min(x, x1)
467         xM = max(x, x1)
468
469         ym = min(y, y1)
470         yM = max(y, y1)
471
472         return abs(xM-xm)*abs(yM-ym)
473
474 #####
475 ##### funkcje przyciskow #####
476 #####
477     def edit(self):
478         """
479             Metoda przypisana do przycisku edycji wlaczajaca
480             i wylaczajaca tryb edycji
481         """
482         if self.przyciski[0].isChecked():
483             self.obiekt_oznaczony.edit()
484
485         else:
486             self.obiekt_oznaczony.zakoncz_edit()
487

```

```

488     def nowy_obraz(self, obraz):
489         """
490             Metoda wgrywajaca nowy obraz i prostokat do podgladu
491             :param obraz: obraz odczytany po koncu edycji
492         """
493         if type(obraz) == QPixmap:
494             self.obraz = obraz
495
496             self.podglad.nowy_podglad(self.obraz, self.objekt_oznaczony. \
497                                         pobierz_wzgledny_rectagle())
498
499     def fine_edit(self):
500         """
501             metoda wywolywana po nacisnieciu przycisku
502             fine edit wlaczajaca tryb edycji
503             z wykorzystaniem przyciskow
504             :return:
505         """
506
507             self.usun_glowne_przyciski()
508
509             self._fine_edit_buttons()
510
511             #poprawienie ksztaltu widgetu
512             self.zablokuj_rozmiar(180, 270 + 15)
513
514     def retyurn_to_normalbuttons(self):
515         """
516             metoda usuwajaca przyciski do edycji i
517             ustawiajaca przyciski standardowe
518         """
519
520             self.usun_fine_edit()
521
522             self.glowne_przyciski()
523             self.zablokuj_rozmiar()
524
525
526 #####dwutrybowe przyciski _kierunkowe#####
527
528     def gorny_przycisk(self):
529         """
530             metoda wykonywana podczas nacisniecia jednego z przyciskow
531             posiada 2 tryby
532             w zaleznosci od ustawienia przelacznika przemiesc
533             wywoluje odpowiednia metode obiektu oznaczonego w celu
534             edycji/przemieszczenia ROI'u
535         """
536
537
538         if self.przemiesc.isChecked():
539
540             self.objekt_oznaczony.przekształc_gorna_linie(
541                                         self.zwieksz.isChecked())
542
543         else:
544
545             self.objekt_oznaczony.przesun_w_gore()
546
547             self.odswierz_kordynaty()
548

```

```

549     def dolny_przycisk(self):
550         """
551             metoda wykonywana podczas nacisniecia jednego z przyciskow
552             posiada 2 tryby
553             w zaleznosci od ustawienia przelacznika przemiesc
554             wywoluje odpowiednia metode obiektu oznaczonego w celu
555             edycji/przemieszczenia ROI'u
556         """
557         if self.przemiesc.isChecked():
558             self.obiekt_oznaczony.przekstalc_dolna_linie(
559                 self.zwieksz.isChecked())
560         else:
561             self.obiekt_oznaczony.przesun_w_dol()
562
563         self.odswierz_kordynaty()
564
565     def lewy_przycisk(self):
566         """
567             metoda wykonywana podczas nacisniecia jednego z przyciskow
568             posiada 2 tryby
569             w zaleznosci od ustawienia przelacznika przemiesc
570             wywoluje odpowiednia metode obiektu oznaczonego w celu
571             edycji/przemieszczenia ROI'u
572         """
573         if self.przemiesc.isChecked():
574             self.obiekt_oznaczony.przekstalc_lewa_linie(
575                 self.zwieksz.isChecked())
576         else:
577             self.obiekt_oznaczony.przesun_w_lewo()
578
579         self.odswierz_kordynaty()
580
581     def prawy_przycisk(self):
582         """
583             metoda wykonywana podczas nacisniecia jednego z przyciskow
584             posiada 2 tryby
585             w zaleznosci od ustawienia przelacznika przemiesc
586             wywoluje odpowiednia metode obiektu oznaczonego w celu
587             edycji/przemieszczenia ROI'u
588         """
589         if self.przemiesc.isChecked():
590             self.obiekt_oznaczony.przekstalc_prawa_linie(
591                 self.zwieksz.isChecked())
592         else:
593             self.obiekt_oznaczony.przesun_w_prawo()
594
595         self.odswierz_kordynaty()
596
597 #####
598     def przelacz_przemiesc(self):
599         """
600             metoda wywolywana przy przelaczeniu przycisku
601             przemiesc zmienia napis na nim
602         """
603         if self.przemiesc.isChecked():
604             self.przemiesc.setText("ksztalt")
605
606         else:
607             self.przemiesc.setText("przemiesc")
608
609

```

```

610     def przelacz_zwieksz(self):
611         """
612             metoda wywolywana przy przelaczeniu
613             przycisku zwieksz zmienia napis na nim
614         """
615
616         if self.zwieksz.isChecked():
617             self.zwieksz.setText("-")
618
619         else:
620             self.zwieksz.setText("+")
621
622 ##### metody uzyte do kalibracji podgladu #####
623 #####
624 #####
625
626     def xp(self):
627         self.x += 0.001
628         self.zmien()
629
630     def yp(self):
631         self.y += 0.001
632         self.zmien()
633
634     def zp(self):
635         self.z += 0.001
636         self.zmien()
637
638     def sp(self):
639         self.s += 0.001
640         self.zmien()
641
642     def xm(self):
643         self.x -= 0.001
644         self.zmien()
645
646     def ym(self):
647         self.y -= 0.001
648         self.zmien()
649
650     def zm(self):
651         self.z -= 0.001
652         self.zmien()
653
654     def sm(self):
655         self.s -= 0.001
656         self.zmien()
657
658     def zmien(self):
659         self.podglad.update_rectagle((self.x, self.y,
660                                         self.z, self.s))
661         print(self.x, self.y, self.z, self.s)
662         self.update()

```

Kod źródłowy 8: Klasa odpowiedzialna za komunikacje z manipulatorem.

```
1 import ctypes
2 from time import sleep
3 from PyQt5.QtWidgets import QMessageBox
4
5
6 class manipulator:
7     '''
8     Klasa obslugujaca komunikacje z manipulatorem
9     '''
10
11    # wskaznik do glownego okna
12    main = None
13
14
15    def __init__(self):
16
17        self.c848 = self.zaladuj_sterowniki()
18
19        self.controller_id = self.podloczenie_kontroler()
20
21        self.sprawdzenie_polozenia()
22
23        #proba odczytania pozycji z pliku
24        try:
25            self.ustaw_abs_positions_z_file()
26
27        except Exception:
28            # jesli nie uda sie odczytac pozycji wykonujemy centrowanie
29            self.center()
30
31            #odczytanie pozycji manipulatora i zapisanie go
32            self.x, self.y, self.z = self.pobierz_pozycje_osi('xyz')
33
34            self.wypisz_aktualna_pozycje_manipulatora()
35
36    def __del__(self):
37
38        try:
39            self.zaopisz_pozycje()
40
41            print('is connected:', self._sprawdz_polozenie_prywatne())
42
43            self.przerwij_polozenie()
44
45            print('is connected:', self._sprawdz_polozenie_prywatne())
46
47        except AttributeError as e:
48            print(e)
49
50
51    def glowne_okno(self, main):
52
53        '''
54        Metoda wykonywana po stworzeniu glownego okna
55        przekazujaca je w celu komunikacji
56        :param main: wskaznik do glownego okna
57        '''
58
59        self.main = main
```

```

61     def center(self):
62
63         """
64             Metoda wlaczajaca referecing mode i
65             nastempnie centrujaca manipulator
66         """
67
68         self.ustaw_referencing_mode()
69         self.pobierz_pozycje_osi()
70         self.reference_axes()
71         self.poczekaj_na_osiagniecie_celu()
72
73         #odczytanie pozycji manipulatora i zapisanie go
74         self.x, self.y, self.z = self.pobierz_pozycje_osi('xyz')
75
76     def _pobierz_skonwertowane_pozycje(self, osie='XYZ'):
77
78         """
79             metoda konwertuje pozycje zadanych osi na
80             tablice znakow zrozumiala przez manipulator
81             :param osie: string zawierajacy nazwy osi
82             :return: wskaznik do tablicy znakow
83         """
84
85         axes_abs = self._convert_axes(osie)
86         return ctypes.c_char_p(axes_abs.encode('utf-8'))
87
88     def ustaw_referencing_mode(self, osie='xyz', tryb=True):
89
90         """
91             metoda wlacza lub wylacza mod referecyjny
92             dla osi manipulatora
93             :param osie: osie dla ktorych wlaczamy tryb referecji
94             :param tryb: wartosc bitowa wlaczajaca
95             lub wylaczajaca mod
96             :return: potwierdzenie powodzenia
97         """
98
99         powodzenie = False
100        c_id = self._convert_id(self.controller_id)
101        for os_c in osie:
102            os = self._pobierz_skonwertowane_pozycje(os_c)
103            bool_array = self._stworz_tablice_booli(size=1, values=tryb)
104            powodzenie = self.c848.C848_R0N(c_id, os, bool_array)
105
106        return bool(powodzenie)
107
108    @staticmethod
109    def odczyt_pozycji():
110
111        statyczna metoda odczytujaca pozycje z pliku
112        :return: slownik zawierajacy pozycje
113        """
114
115        slownik_pozycji = {}
116        with open('positions.txt', 'r') as file:
117            for line in file:
118                ax, position = line.split(':')
119                slownik_pozycji[ax.strip()] = float(position.strip())
120
121        return slownik_pozycji

```

```

122
123     @staticmethod
124     def split_axes_positions(position_dict):
125         """
126             Statyczna metoda konwertujaca odczytany
127             slownik na dwie niezalezne tablice
128             :param position_dict: slownik z osiami
129             :return: tablica osi, tablica pozycji
130         """
131         axes = []
132         positions = []
133         for ax, position in position_dict.items():
134             axes.append(ax)
135             positions.append(position)
136
137         axes = '\n'.join(axes)
138         return axes, positions
139
140     def ustaw_abs_positions_z_file(self):
141         """
142             metoda odczytuje pozycje z pliku,
143             natempnie wylacza tryb referecyjny i
144             ustawia ze manipulator znajduje sie
145             na zadanych kordynatach
146             :return: status czy udalo sie
147                 ustawic odczytane pozycje czy nie
148         """
149
150         self.ustaw_referencing_mode(tryb=False)
151         position_dict = self.odczyt_pozycji()
152
153         axes, positions = self.split_axes_positions(position_dict)
154
155         return self.ustaw_absolutne_pozycje(axes=axes,
156                                              positions=positions)
157
158     def zaopisz_pozycje(self):
159
160         """
161             metoda zapisuje pozycje do pliku
162         """
163
164         axes = 'xyz'
165         positions = self.pobierz_pozycje_osi(axes)
166         with open('pozycje.txt', 'w') as file:
167             for ax, position in zip(axes, positions):
168                 file.write('{}: {}\\n'.format(ax, position))
169
170     def ustaw_absolutne_pozycje(self, axes='xyz', positions = None):
171         """
172             Metoda wysyla do manipulatora zadane pozycje
173             w celu przesuniecia manipulatora na nie
174             :param axes:
175             :param positions:
176             :return: status czy sie udalo wykonac przemieszczenie
177         """
178
179         #sprawdzenie czy zadano jakiekolwiek osie do przemieszczenia
180         if positions == None:
181             return None

```

```

183     # sprawdzenie czy zadano odpowiednia ilosc osi i pozycji
184     if len(axes) != len(positions):
185         print('number of axes and positions must be the same!')
186         return None
187
188     # wykonanie przemieszczenia
189     c_id = self._convert_id(self.controller_id)
190     sz_axes = self._pobierz_skonwertowane_pozycje(axes)
191     c_double_array = self._stworz_tablice_doublu(len(axes), positions)
192     success = self.c848.C848_POS(c_id, sz_axes, c_double_array)
193
194     return bool(success)
195
196 @staticmethod
197 def zaladuj_sterowniki(filename='C848_DLL.dll'):
198     """
199     Statyczna metoda odczytujaca i ladujaca
200     dynamiczna bibliotekę do komunikacji z silnikiem
201     :param filename: nazwa pliku biblioteki
202     :return: status
203     """
204
205     return ctypes.CDLL(r"C848_DLL.dll")
206
207
208 def podloczenie_kontroler(self):
209     """
210     podloczenie do kontrolera za pomoca zaladowanego
211     sterownika
212     return kontroler id albo
213     None jesli nie udalo sie podloczyc
214     """
215
216
217     connect_fun = self.c848.C848_ConnectRS232
218     connect_fun.argtypes = [ctypes.c_int, ctypes.c_long]
219     #connect_fun(port_szeregowy, predkosc bitow)
220     controller_id = connect_fun(4, 57600)
221
222     if controller_id != -1:
223
224         return controller_id
225
226     else:
227
228         QMessageBox.warning(self.main, '',
229                             "erore unable to conect", QMessageBox.Ok)
229         return None
230
231
232 def sprawdzenie_poloczenia(self):
233     """
234     metoda sprawdzajaca stan poloczenia z kontrolerem
235     jesli nie to wyskakuje okienko informuje o tym
236     """
237
238
239     if not self._sprawdz_poloczenie_prywatne():
240
241         QMessageBox.warning(self.main, '',
242                             "erore conection feiled", QMessageBox.Ok)

```

```

244     def _sprawdz_poloczenie_prywatne(self):
245         """
246             Prywatna metoda sprawdzajaca poloczenie
247             :return: status poloczenia
248         """
249
250         return bool(self.c848.C848_IsConnected(
251             ctypes.c_int(self.controller_id)))
252
253     def przerwij_poloczenie(self):
254         """
255             Przeywa poloczenie z kontrolerem silnikow
256             :return:
257         """
258
259         self.c848.C848_CloseConnection(ctypes.c_int(self.controller_id))
260
261     def reference_axes(self, axes='xyz'):
262         """
263             Wykonuje ruch przekazanych osi do pkt referencyjnego
264             :param axes:
265             :return: status
266         """
267
268         c_id = self._convert_id(self.controller_id)
269         sz_axes = self._pobierz_skonwertowane_axes(axes)
270         success = self.c848.C848_REF(c_id, sz_axes)
271         return bool(success)
272
273     @staticmethod
274     def _convert_axes(axes='XYZ'):
275
276         """
277             konwertuje "xyz" string do "ABC"
278             string zrozumialego dla kontrolera
279             #A = x
280             #B = y
281             #C = z
282         """
283
284         axes_map = {'x': 'A', 'y': 'B', 'z': 'C'}
285         abc_list = [axes_map[ax] for ax in list(axes.lower())]
286         return ''.join(abc_list)
287
288     def _pobierz_skonwertowane_axes(self, axes='XYZ'):
289         """
290             Przyjmuje osie w postaci stringu
291             return wskaznik na tablice znakow zrozumiala przez kontroler
292         """
293
294         axes_abc = self._convert_axes(axes)
295         return ctypes.c_char_p(axes_abc.encode('utf-8'))
296
297     @staticmethod
298     def _convert_id(controller_id):
299         """
300             converts controller_id to c_int
301         """
302
303         return ctypes.c_int(controller_id)
304

```

```

305     @staticmethod
306     def _stworz_tablice_booli(size=1, values=0):
307
308         """
309         tworzy tablice booleanow o zadanej wielkosci
310         i zwraca cpointer do niej
311         """
312
313         b = (ctypes.c_bool * size)(*values] * size)
314         return ctypes.cast(b, ctypes.POINTER(ctypes.c_bool))
315
316     @staticmethod
317     def _stworz_tablice_doublji(size=1, positions=None):
318
319         """
320         tworzy tablice doublji o zadanej wielkosci
321         i zwraca cpointer do niej
322         """
323
324         if positions == None:
325             positions = [25.0] * size
326
327         arr = (ctypes.c_double * size)(*positions)
328         return ctypes.cast(arr, ctypes.POINTER(ctypes.c_double))
329
330     def przsun_manipulator_do_zadanej_pozycji(self, axes='xyz',
331                                              positions=[25.0, 25.0, 25.0]):
332
333         """
334         Metoda zadajaca przesuniecie zadanych osi na zadana pozycje
335         :param axes: znak osi
336         :param positions: pozycje dla odpowiednich osi
337         :return: status powodzenia
338         """
339
340         return self._przsun_manipulator(axes, positions)
341
342     def _przsun_manipulator(self, axes='xyz', positions=[25.0, 25.0, 25.0]):
343
344         """
345         prywatna metoda zadajaca przesuniecie
346         zadanych osi na zadana pozycje
347         :param axes: znak osi
348         :param positions: pozycje dla odpowiednich osi
349         :return: status powodzenia
350         """
351
352         if len(axes) != len(positions):
353             QMessageBox.warning(self.main, '',
354                                 'number of axes and positions must be the same!', )
355             QMessageBox.Ok)
356             return None
357
358         c_id = self._convert_id(self.controller_id)
359         sz_axes = self._pobierz_skonwertowane_axes(axes)
360         c_double_array = self._stworz_tablice_doublji(len(axes), positions)
361         success = self.c848.C848_MOV(c_id, sz_axes, c_double_array)
362
363         return bool(success)
364
365

```

```

366     def pobierz_pozycje_osi(self, axes='xyz'):
367         '''
368             Metoda zwraca pozycje manipulatora
369         '''
370         c_id = self._convert_id(self.controller_id)
371         sz_axes = self._pobierz_skonwertowane_axes(axes)
372         c_double_array = self._stworz_tablice_doublu(size=len(axes))
373
374         if self.c848.C848_qPOS(c_id, sz_axes, c_double_array):
375             return c_double_array[:len(axes)]
376         else:
377             print('something went wrong while reading position')
378             return False
379
380     def wypisz_aktualna_pozycje_manipulatora(self):
381         '''
382             metoda czeka az manipulator przestanie sie poruszac
383             po osiognieciu pozycji wypisze pozycje
384         :return:
385         '''
386
387         while not self.pobierz_pozycje_osi('xyz'):
388             sleep(1)
389
390         self.x, self.y, self.z = self.pobierz_pozycje_osi('xyz')
391         print(self.x, self.y, self.z)
392
393     def sprawdz_czy_u_celu(self, axes='xyz'):
394         '''
395             metoda sprawdza czy manipulator osiagnal zadana pozycje
396         '''
397
398         c_id = self._convert_id(self.controller_id)
399         status = {}
400         for c in axes:
401             axis = self._pobierz_skonwertowane_axes(c)
402             bool_array = self._stworz_tablice_booli(size=1)
403             check = self.c848.C848_qONT(c_id, axis, bool_array)
404
405             if check != 1:
406                 return {c:False}
407
408             status[c] = bool_array[0]
409         return status
410
411     def poczekaj_na_osiagniecie_celu(self):
412         '''
413             metoda oczekujca az zostanie osiagnieta zadana pozycja.
414             Jednoczesnie updatujaca napisy na glownym oknie
415         '''
416
417         while not all(self.sprawdz_czy_u_celu().values()):
418             pass
419
420         self.x, self.y, self.z = self.pobierz_pozycje_osi('xyz')
421
422         if not self.main is None:
423
424             self.main._upadet_position_read()

```

```

427 ##### metody odbierajace wyslane poleca ruchu#####
428
429     def przesun_w_gore(self , krok):
430         """
431             metoda przsuwajaca pozycje manipulatora o zadany krok
432             i oczekujaca az zostanie osiagniety cel
433             :param krok:
434             :return: powodzenie przesuniecia
435         """
436
437         self.z -= krok
438         t = self.przsun_manipulator_do_zadanej_pozycji('z' , [self.z])
439         self.poczekaj_na_osiagniecie_celu()
440         return t
441
442     def przesun_w_dol(self , krok):
443         """
444             metoda przsuwajaca pozycje manipulatora o zadany krok
445             i oczekujaca az zostanie osiagniety cel
446             :param krok:
447             :return: powodzenie przesuniecia
448         """
449
450         self.z+=krok
451         t = self.przsun_manipulator_do_zadanej_pozycji('z' , [self.z])
452
453         self.poczekaj_na_osiagniecie_celu()
454
455         return t
456
457     def przesun_w_prawo(self , krok):
458         """
459             metoda przsuwajaca pozycje manipulatora o zadany krok
460             i oczekujaca az zostanie osiagniety cel
461             :param krok:
462             :return: powodzenie przesuniecia
463         """
464
465         self.y-=krok
466         t = self.przsun_manipulator_do_zadanej_pozycji('y' , [self.y])
467
468         self.poczekaj_na_osiagniecie_celu()
469
470         return t
471
472     def przesun_w_lewo(self , krok):
473         """
474             metoda przsuwajaca pozycje manipulatora o zadany krok
475             i oczekujaca az zostanie osiagniety cel
476             :param krok:
477             :return: powodzenie przesuniecia
478         """
479
480         self.y+=krok
481         t = self.przsun_manipulator_do_zadanej_pozycji('y' , [self.y])
482
483         self.poczekaj_na_osiagniecie_celu()
484
485         return t
486
487

```

```

488     def przesun_x(self, wartosc):
489         """
490             przesuwa os x na zadana wartosc
491             :param wartosc:
492             :return: status
493         """
494
495         self.x = wartosc
496         t = self.przsun_manipulator_do_zadanej_pozycji('x', [self.x])
497         self.poczekaj_na_osiagniecie_celu()
498         return t
499
500     def simple_stop(self):
501         """
502             zatrzymuje ruch manipulatora
503         """
504
505         c_id = self._convert_id(self.controller_id)
506         return self.c848.C848_STP(c_id)
507
508     def move_axes_to_abs_woe_ofset(self, axes, tab):
509         """
510             metoda umozliwiajaca przesuniecie wiecej niz 1 osi na raz
511             umozliwia implementacje centrowania na zadanym pkcie
512             :param axes: osie ktore przesuwamy
513             :param tab: tablica wartosci o ktore przesuwamy manipulator
514         """
515
516         ntab = []
517         for a,v in zip(axes,tab):
518             if a == 'x':
519                 self.x -= v
520                 ntab.append(self.x)
521             if a == 'y':
522                 self.y -= v
523                 ntab.append(self.y)
524             if a == 'z':
525                 self.z-=v
526                 ntab.append(self.z)
527
528         self.przsun_manipulator_do_zadanej_pozycji(axes, ntab)
529         self.poczekaj_na_osiagniecie_celu()

```

Kod źródłowy 9: Okno ustawien.

```
1 from PyQt5.QtWidgets import QWidget, QLabel, QGridLayout, \
2 QLineEdit, QVBoxLayout, QPushButton
3 from PyQt5.QtGui import QIcon, QIntValidator
4
5 class Okno_ustawien(QWidget):
6     """
7     Obiekt dziedziczący z QWidget umożliwiający
8     konfigurację parametrów przesyłanych do manipulatora
9     """
10    def __init__(self, glowne_okno, *args, **kwargs):
11        super(Okno_ustawien, self).__init__(*args, **kwargs)
12        self.glowne_okno = glowne_okno
13
14        self.setWindowIcon(QIcon('icon.png'))
15        self.setWindowTitle("Ustawienia osi")
16
17        self.opcje = ()
18
19        self.layout = QGridLayout()
20        self._stworz_opcje("Minimum slidera osi X", 0, 20)
21        self._stworz_opcje("maximum slidera osi X", 1, 30)
22        self._stworz_opcje("Krok dla y i z w 0.1 mm", 2, 10)
23
24        self.przycisk = QPushButton("Zapisz i zastosuj ustawienia")
25        self.przycisk.clicked.connect(self._zwroc_odpowiedzi)
26        self.przycisklayout = QVBoxLayout()
27
28        self.przycisklayout.addLayout(self.layout)
29        self.przycisklayout.addWidget(self.przycisk)
30        self.setLayout(self.przycisklayout)
31
32    def _zwroc_odpowiedzi(self):
33        """
34        metoda przekazująca odpowiedzi do odpowiednich obiektów
35        """
36        self.glowne_okno.slide.ustaw_min_max(int(self.opcje[0].text()),
37                                              int(self.opcje[1].text()))
38        self.glowne_okno.ustaw_krok(int(self.opcje[2].text()))
39        self.hide()
40
41    def _stworz_opcje(self, tekst, pozycja, wartosc):
42        """
43            :param tekst: treść opcji
44            :param pozycja: pozycja opcji
45            :param wartosc: wartość startowa
46        """
47        self.layout.addWidget(QLabel(tekst), pozycja, 0)
48
49        self.opcje += (self._stworz_lineedit(),)
50        self.opcje[-1].setText(str(wartosc))
51        self.layout.addWidget(self.opcje[-1], pozycja, 1)
52
53    def _stworz_lineedit(self):
54        """
55            metoda tworząca pole edycji z kontrolą wpisywania
56        """
57       lineEdit = QLineEdit()
58        validator = QIntValidator()
59        lineEdit.setValidator(validator)
60        return lineEdit
```

Kod źródłowy 10: Slider osi Z.

```
1 from PyQt5.QtWidgets import QSlider
2 from PyQt5.QtCore import Qt
3
4 class Slider(QSlider):
5     """
6     Obiekt dziedzicacy z Qslidera umozliwiajacy obsluge osi x
7     manipulatora odpowiadajacego za przyblizenie kamery
8     """
9     max_slidera = 1000
10
11    def __init__(self, mainwinow, min, max, value=25, *args, **kwargs):
12        super(Slider, self).__init__(*args, **kwargs)
13        self.mainwindow = mainwinow
14
15        #wartosci na slajderze
16        self.max, self.min, self.value = max, min, value
17
18        #przepisanie funkcji do zmiany wartosci na sliderze
19        self.valueChanged[int].connect(self.zmiana)
20
21        #ustawienia wygladu i parametrow slidera
22        self.setFocusPolicy(Qt.StrongFocus)
23        self.setTickPosition(QSlider.TicksBothSides)
24        self.setTickInterval(100)
25        self.setSingleStep(10)
26        self.setMaximum(self.max_slidera)
27        self.setMinimum(0)
28        self.setValue(self.rekonwersja(self.value))
29
30    def konwersja(self, wartosc):
31        """
32        konwersja wartosci z wartosci w przestrzeni
33        slidera na wartosc rzeczywista
34        :param wartosc: wartosc w przestrzeni slidera
35        :return: wartosc w przestrzeni rzeczywistej
36        """
37        wynik = wartosc / self.max_slidera
38        wynik *= (self.max - self.min)
39        wynik += self.min
40        return wynik
41
42    def rekonwersja(self, wartosc):
43        """
44        konwersja wartosci z przestrzeni rzeczywistej
45        na przestrzen slidera
46        :param wartosc: wartosc w przestrzeni rzeczywistej
47        :return: wartosc w przestrzeni slidera
48        """
49        wynik = (wartosc - self.min)
50        wynik /= (self.max - self.min)
51        wynik *= self.max_slidera
52        return wynik
53
54    def zmiana(self, wartosc):
55        self.value = self.konwersja(wartosc)
56        self.mainwindow.manipulaor.przesun_x(self.konwersja(wartosc))
57
58    def ustaw_min_max(self, min, max):
59        self.max, self.min = int(max), int(min)
60        self.setValue(self.rekonwersja(self.value))
```