

# DOKUMENTACJA PROJEKTU

## VPlotter

---

### **Autorzy:**

Krzysztof Murawski – szef projektu, email: h2022krzmur@lo1rumia.webd.pl

Agata Wojciechowska – email: h2022agawoj@lo1rumia.webd.pl

Julia Kołodziejczyk – email: h2022julkol@lo1rumia.webd.pl

Piotr Stęchły – email: h2022pioste@lo1rumia.webd.pl

Maksym Varlamow – email: h2022makvar@lo1rumia.webd.pl

Maciej Gessler – email: h2022macges@lo1rumia.webd.pl

Kacper Gurzęda – email: h2022kacgur@lo1rumia.webd.pl

Adam Sądej – email: h2022adasad@lo1rumia.webd.pl

Filip Połubiński – email: h2022filpol@lo1rumia.webd.pl

Mateusz Witka – email: h2022matwit@lo1rumia.webd.pl

Jan Dorszyński – email: h2022jandor@lo1rumia.webd.pl

# SPIS TREŚCI

---

Wykorzystane biblioteki:

- Biblioteka os
- Biblioteka PIL
- Biblioteka math
- Biblioteka numpy

Pliki projektu:

- Plik config.py
- Plik image\_processor.py
- Plik luk.py
- Plik generator\_krokov\_silnikow.py
- Plik pixcel.py
- Plik main.py
- Symulacja.py

# WYKORZYSTANE BIBLIOTEKI

---

**Biblioteka os** - "operating system" (system operacyjny), pozwala na wykonywanie operacji związanych z zarządzaniem plikami, katalogami oraz środowiskiem operacyjnym.

*Funkcje wykorzystane w projekcie:*

**os.path.abspath()** - daje dostęp do ścieżki "absolutnej" (z ang. abs – absolute), czyli do całej ścieżki pliku, którego podamy w nawiasie.

**os.path.dirname()** - daje ścieżkę, ale ucina z niej ostatni, konkretny plik, który na tej ścieżce jest (np. Ścieżka to C:\Users\uzytkownik\Pulpit\folder\plik.txt, wynikiem działania funkcji dla argumentu "plik.txt" będzie C:\Users\uzytkownik\Pulpit\folder).

**os.path.join()** - łączy ścieżki podane w nawiasie.

**Biblioteka PIL (Python Imaging Library)** - służy do wykonywania operacji z obrazami, pozwala ona m.in. na zmianę wymiarów obrazu, przekształcanie obrazu oraz na konwersję z tablicy numpy na obraz.

*Funkcje wykorzystane w projekcie:*

**Image.open()** - wczytuje obraz o podanej w nawiasie ścieżce.

**ImageDraw.Draw()** - pozwala na rysowanie na obrazie. W nawiasie podajemy obraz, na którym będziemy chcieli wykonać operacje rysowania.

**Image.new(tryb, rozmiar, kolor)** - służy do stworzenia nowego obrazu. W nawiasie podajemy tryb obrazu (np. "RGB" - kolorowy obraz, "L" - obraz monochromatyczny), rozmiar podany jako krotka "(szerokość, wysokość)" oraz kolor (opcjonalny) - określa on kolor tła naszego obrazu.

**nazwa\_zmiennej.convert("L")** - przekształca obraz na monochromatyczny.

**image.resize((szerokość, wysokość))** - zmienia wymiary obrazu do podanych wymiarów.

**Image.fromarray()** - konwertuje tablicę numpy na obraz PIL.

**Biblioteka math** – jest to biblioteka, która ułatwia wykonywanie operacji matematycznych w pythonie.

*Funkcje wykorzystane w projekcie:*

**math.pi** - zwraca liczbę pi przybliżoną do 15 miejsc po przecinku.

**math.sqrt()** - zwraca pierwiastek kwadratowy z wartości podanej w napisie.

**math.asin()** - funkcja odwrotna do funkcji **math.sin()**, przyjmuje wartość sinusa i zwraca odpowiadający jej kąt w radianach.

**math.degrees()** - przekształca radiany na stopnie.

**Biblioteka NumPy** - jest to biblioteka, która ułatwia wykonywanie operacji matematycznych poprzez wykonywanie pracy na tablicach i macierzach.

*Funkcje wykorzystane w projekcie:*

**np.array()** - to funkcja, która konwertuje dane na tablicę NumPy.

**np.array().shape** - zwraca krotkę, która zawiera wymiary tablicy - wysokość i szerokość.

## PLIK CONFIG.PY

---

Plik config.py wykorzystuje bibliotekę os i bibliotekę math oraz zawiera informacje typu integer, float, string oraz dictionary na temat:

- nazwy pliku z obrazem do narysowania,
- ilości pikseli pionowo,
- maksymalnego natężenia barw,
- odległości między łukami (w milimetrach)
- odległości lewego silnika od lewego górnego rogu obrazu w poziomie (współrzędne x) oraz w pionie (współrzędne y) - podane w milimetrach.
- odległości między silnikami,
- wartości kąta przy jednym kroku silnika – w stopniach,
- średnicy silnika – w milimetrach,
- słownika wartości liczbowych przypisanych do danego obrotu konkretnego silnika,
- szerokości obrazu (wartość niezmienna)
- długości sznurka na krok (wartość niezmienna)

## PLIK IMAGE\_PROCESSOR.PY

---

Plik `image_processor.py` wykorzystuje bibliotekę `numpy`, `math` oraz `PIL`. Wczytujemy w nim również klasę `Pixel` z pliku `pixel.py` oraz wszystkie dane z pliku `config.py` za pomocą `from config import*`.

Plik składa się z klasy `ImageProcessor`, w której tworzymy poszczególne elementy:

- **Funkcję `__init__`**, która zwraca wartość `None`, służy ona do inicjalizacji klasy.
- **Funkcję `read_image`**, która przyjmuje ścieżkę do obrazu jako argument `img_path`. Za pomocą linijki `"Image.open(img_path)"` otwiera się grafika znajdująca się pod podaną ścieżką, a następnie pobiera wymiary obrazu i przypisuje ją do `szerokosc_piksele` i `wysokosc_piksele`.
- **Funkcję `zmiana_obrazu`**, która dopasowuje obraz do nowego wymiaru zachowując proporcje, na podstawie podanych wartości - aktualnej wysokości i szerokości oraz oczekiwanej wysokości. Wykorzystano w niej funkcję biblioteki `PIL` – `resize`.
- **Funkcję `mono_image`**, która służy do konwersji obrazu na monochromatyczny (czarno-biały). Używamy do tego metody `convert("L")`. `"L"` przekształca grafikę na grafikę w skali szarości.
- **Funkcję `process_intensity_scale`**, która służy do modyfikacji jasności obrazu. Funkcja ta pobiera wartość maksymalnego natężenie piksela, szerokość i wysokość obrazu. Konwertuje obraz na tablicę `numpy` (`np.array(self.image)`), a następnie przechodzi po każdym pikselu za pomocą dwóch pętli `for`, aby pobrać jasność piksela i obliczyć nową, a następnie odwrócić tą wartość. Na koniec funkcja konwertuje tablicę pikseli z powrotem na obraz `PIL` (`Image.fromarray(piksele)`).

## PLIK LUK.PY

---

Plik składa się z klasy Luk w której tworzymy poszczególne elementy:

- **Funkcję `__init__`**, która zwraca wartość None, służy ona do inicjalizacji klasy.
- **Funkcję `oblicz_kat_luku`**, która służy do obliczenia kąta łuku za pomocą sinusów oraz twierdzenia pitagorasa. Zmienna c przyjmuje wynik obliczenia odległości między punktami początkowymi a końcowymi łuku. Następnie obliczany jest sinus połowy kąta, a kąt łuku liczymy jako dwukrotność `sinus_polowy` (zaokrąglony do dwóch miejsc po przecinku).
- **Funkcję `oblicz_dlugosc_luku`**, która oblicza długość łuku za pomocą wzoru `“(self.angle * math.pi * self.radius) / 180”`
- **Funkcję `odwroc`**, która odwraca kolejność pikselioraz zamienia punkty początkowe z końcowymi.

## PLIK GENERATOR\_KROKOW\_SILNIKA.PY

---

Plik `generator_krokow_silnika.py` tworzy dane do silnika. Wykorzystuje do tego informacje zawarte w pliku `config.py`, klasę `Pixel` z pliku `pixel.py` oraz bibliotekę `os`.

W pliku znajduje się funkcja `odleglosc_na_krok`, która zwraca ilość kroków potrzebna do długości sznurka. Oprócz tego jest też funkcja `oblicz_dlugosc_prawego_sznurka`, która służy do obliczenia długości prawego sznurka na podstawie odległości między silnikami oraz współrzędnych `x` i `y`.

`generator_krokow_silnika.py` składa się również z klasy `GeneratorKrokowSilnika`, w skład której wchodzi poszczególne elementy:

- **Funkcję `__init__`**, która zwraca wartość `None`, służy ona do inicjalizacji klasy.
- **Funkcję `generuj`**, która...
- **Funkcję `zapisz_do_pliku`**, która zapisuje kroki z listy `self.kroki` do pliku tekstowego `kroki.txt` w folderze `"res"`.



## PLIK PIXEL.PY

---

W pliku pixel.py wczytujemy wszystkie dane z pliku config.py za pomocą `from config import*`.

Plik składa się z klasy Pixel, w której tworzymy poszczególne elementy:

- **Funkcję `__init__`**, która zwraca wartość None, służy ona do inicjalizacji klasy.
- **Funkcję `get_xmm`**, która oblicza pozycję piksela na osi X w milimetrach.
- **Funkcję `get_ymm`**, która oblicza pozycję piksela na osi Y w milimetrach.

## PLIK MAIN.PY

---

Główny plik programu. Wykorzystuje bibliotekę os oraz wczytuje następujące elementy plików - z pliku config.py wczytuje wszystko, z pliku image\_processor.py wczytuje klasę ImageProcessor, a z pliku simulation.py wczytuje klasę Simulation.

Plik main.py zawiera funkcję main, która uruchamia poszczególne elementy ze wczytanych plików - np. zmiana rozmiaru obrazu, generowanie kroków, zapisywanie kroków do pliku tekstowego, tworzenie symulacji.

## SYMULACJA.PY

---

Plik symulacja.py służy do przetestowania działania programu i zwizualizowania jego efektu przed użyciem Vplottera. Wykorzystuje do tego bibliotekę math, PIL, os, a także wczytuje wszystko z pliku config.py

Plik składa się z klasy Simulation w której tworzymy poszczególne elementy:

- **Funkcję `__init__`**, która zwraca wartość None, służy ona do inicjalizacji klasy. Występuje tu zamiana wartości z milimetrów na piksele oraz stworzenie obrazu RGB z białym tłem do symulacji za pomocą funkcji: `"Image.new("RGB", (int(self.motor_spacing), int(self.motor_spacing)), "white")"`.
- **Funkcję `wart_y`**, która służy do obliczenia za pomocą wzoru herona koordynatu y, znając długości dwóch sznurków - `left_string_length` oraz `right_string_length`.
- **Funkcję `wart_x`**, która służy do obliczenia za pomocą twierdzenia pitagorasa koordynatu x, znając długości dwóch sznurków - `left_string_length` oraz `right_string_length`.
- **Funkcję `rysuj`**, która oblicza x i y na podstawie długości sznurków (`l_str_length` oraz `r_str_length`) i zmiany długości sznurka (określonego w kodzie jako `zmiana_lewego_sznurka` i `zmiana_prawego_sznurka`). Funkcja ta rysuje linię na obrazie symulacji między punktem A, a punktem B jeśli marker jest aktywny, a na końcu aktualizuje wartości koordynatów i długości sznurków.
- **Funkcję `stworz_symulacje`**, która otwiera plik `kroki.txt` i iteruje przez każdą linię, usuwając białe znaki, następnie sprawdza ona różne przypadki – np. `"10"`, `"1"` itd. - jeśli linia zawiera `"10"` to wywoływane jest `self.rysuj` z argumentami (`self.dlugosc_sznurka_na_krok, 0`), jeśli linia zawiera `"1"` to `self.marker` zostanie ustawiony jako False. Po przejściu całego pliku, funkcja pokazuje efekt końcowy symulacji.