

Kamerki USB jako detektor promieniowania kosmicznego

Patryk Elak, Anna Latanowicz, Dominik Jezierski-Tratkiewicz,
Tytus Puzianowski - Krzypkowski,

luty 2026

Streszczenie

Projekt studencki kontynuujący rozwój aplikacji wykorzystującej kamery USB jako niskobudżetowe i łatwo dostępne detektory promieniowania kosmicznego, wykorzystywanej podczas warsztatów przeprowadzanych w szkołach podstawowych i ponadpodstawowych. Najważniejszym celem wykonanych prac było stworzenie środowiska przystępnego dla każdego nauczyciela oraz ucznia.

W ramach projektu wykonano zadania takie jak: zintegrowanie aplikacji z zewnętrzną stroną internetową, implementacja interfejsu GUI, stworzenie algorytmów wykrywania większej ilości zdarzeń i ich klasyfikacji oraz testy programu po zmianach i porównanie z poprzednią wersją.

1 Integracja aplikacji z API CREDO

W aplikacji zaimplementowano pełną, dwukierunkową komunikację z serwerem CREDO¹ z wykorzystaniem protokołu *HTTP*. Przekształciło to aplikację w tzw. klienta *IoT*, jednocześnie pozwalając uczestniczącym w warsztatach szkołom na porównanie się z innymi placówkami i uzyskanie wglądu w podsumowanie swoich detekcji².

Program obsługuje teraz uwierzytelnianie (logowanie tokenem) i rejestrację nowych użytkowników, rejestrację detektorów (zestaw kamera i komputer) oraz cykliczne wysyłanie sygnałów podtrzymujących stan zalogowania w celu utrzymania sesji.

Najważniejszą zmianą w tej części jest implementacja systemu raportującego zdarzenia zebrane przez detektory. Klatki z każdej detekcji są teraz konwertowane w czasie rzeczywistym do formatu *JSON*, zawierającego wycinek obrazu zakodowany w *Base64*. Serwer oprócz samego obrazu odbiera także opisujące konkretne zdarzenie metadane (jasność, kształt, czas itp.), które są wymagane przez administratora portalu wyświetlającego zdarzenia.

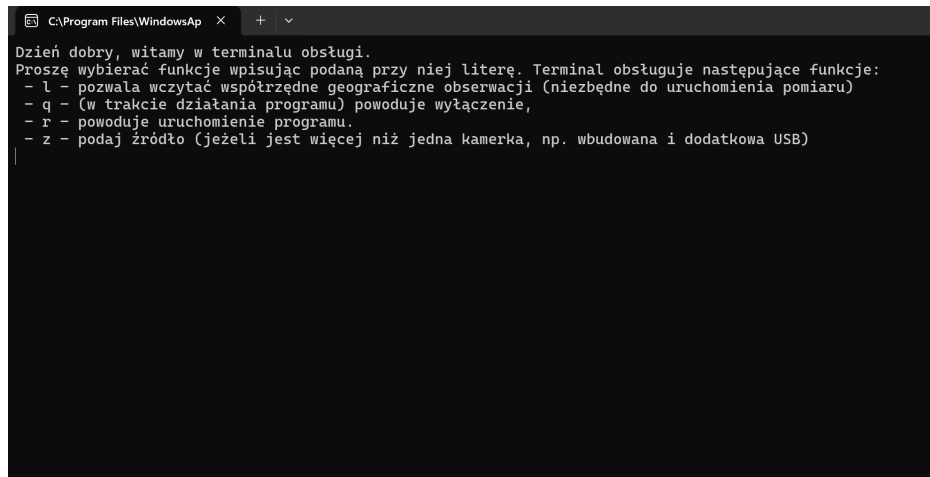
W celu zapewnienia płynności działania detekcji przy jednoczesnej komunikacji sieciowej, zmieniono architekturę aplikacji na wielowątkową. Proces analizy obrazu został odseparowany od procesu wysyłania danych za pomocą kolejki FIFO (*First In, First Out*). Dzięki temu rozwiązaniu, ewentualne opóźnienia w transmisji danych lub chwilowy brak dostępu do sieci nie blokują głównej pętli programu, a detektor nie traci klatek wideo podczas próby połączenia z serwerem.

¹Szczegółowy opis budowy serwera oraz sposobu komunikacji z nim, znajduje się w dokumentacji projektu [1].

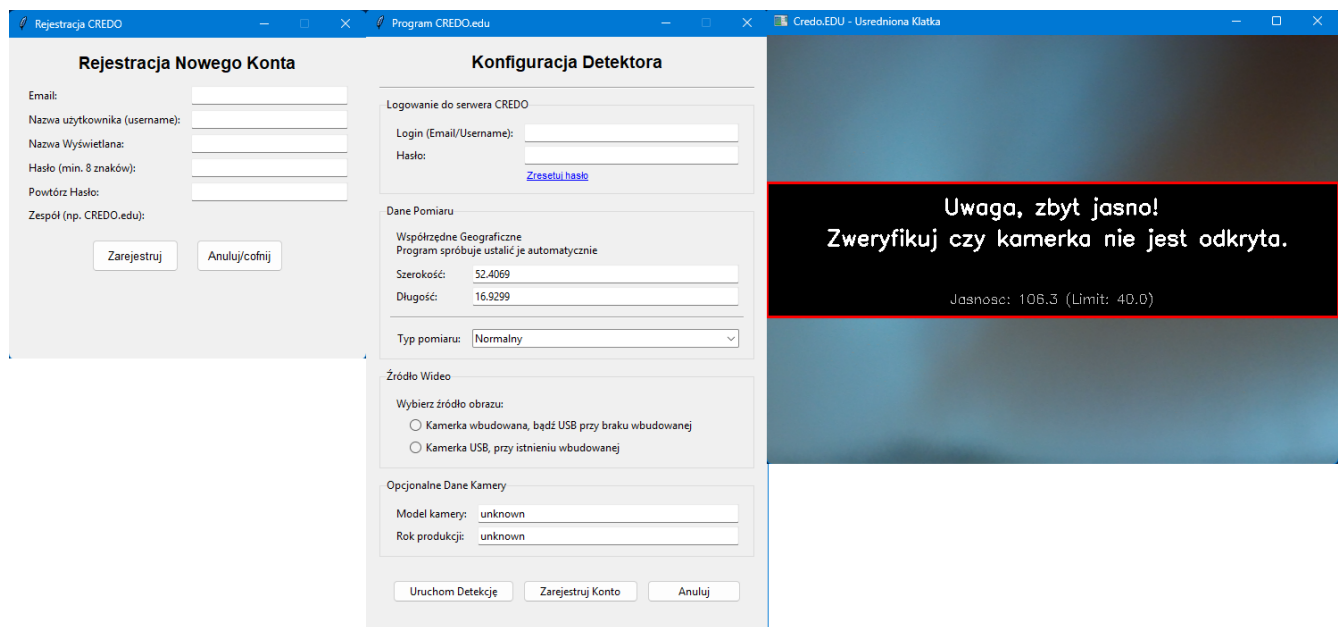
²Lista uczestniczących placówek oraz zebrane zdarzenia każdej z nich dostępne są publicznie [2].

2 Implementacja interfejsu graficznego (GUI)

Dotychczasowy interfejs tekstowy oparty na komunikacji z terminalem (Rys. 1) zastąpiono oknem graficznym zbudowanym przy użyciu pakietu `tkinter`. Nowe GUI (Rys. 2) pozwoliło na zwiększenie przystępności użytkowania aplikacji bez konieczności znajomości obsługi terminala/wiersza poleceń. Za pomocą formularzy i list rozwijanych umożliwia rejestrację, logowanie oraz konfigurację istotnych parametrów takich jak: współrzędne geograficzne, typ pomiaru, wybór i rodzaj kamery.



Rysunek 1: Wygląd interfejsu użytkownika przed zmianami.



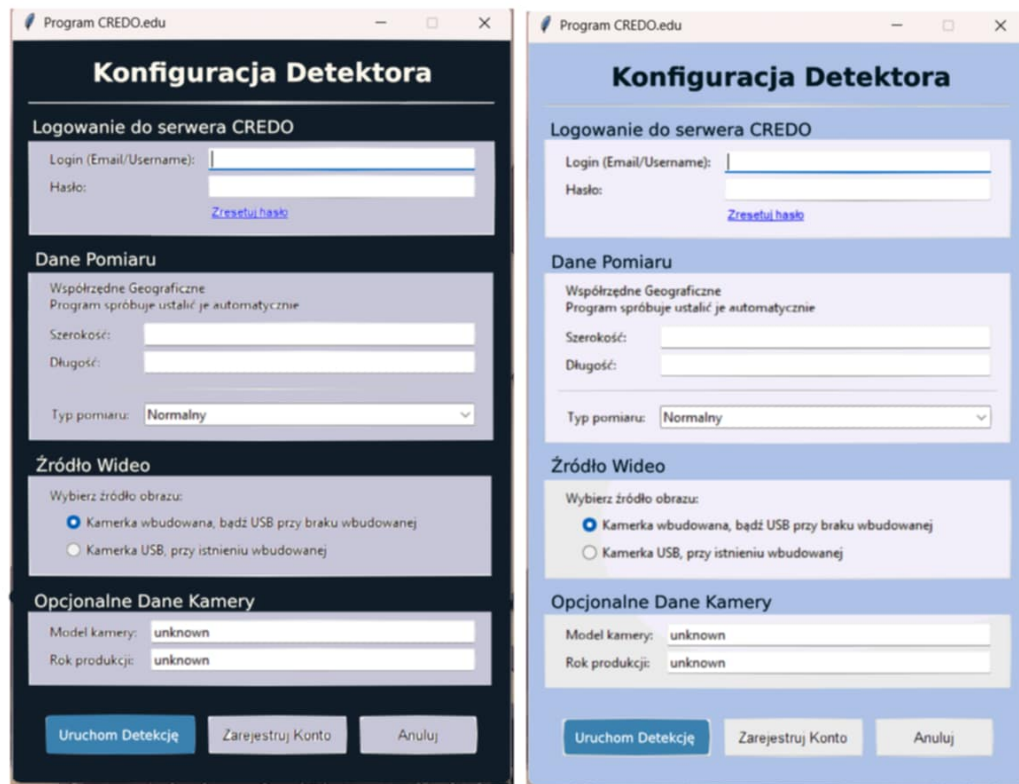
Rysunek 2: Wygląd interfejsu użytkownika po zmianach. Po lewej: okienko rejestracji nowego konta; pośrodku: konfiguracja przeprowadzanych pomiarów; po prawej: komunikat informujący o możliwej odsłoniętej kamerce (poprawiający bezpieczeństwo wysłanych danych).

Na tym etapie wprowadzono automatyczne odczytywanie szerokości oraz długości geograficznej na podstawie adresu IP użytkownika. W oparciu o te dane, aplikacja automatycznie określa wysokość nad poziomem morza, wykorzystując API serwisu *Open-Meteo*. Jednocześnie, aby usprawnić proces konfiguracji

przy kolejnych uruchomieniach programu, wprowadzono trwały zapis ustawień użytkownika oraz wizualną obsługę błędów poprzez okna dialogowe.

3 Projekt strony wizualnej interfejsu użytkownika

Wykonano dwa projekty części wizualnej okna dialogowego. Głównym celem było zwiększenie czytelności oraz uatrakcyjnienie doświadczenia użytkownika. Projekt uwzględnił możliwość wyboru dla użytkownika między jasnym i ciemnym motywem. Niestety ze względów technicznych projektu na razie nie udało się zaimplementować.



Rysunek 3: Projekt kolorystyczny dwóch wariantów interfejsu programu: ciemny i jasny

4 Implementacja algorytmu wyszukującego ponad jedno zdarzenie

Pierwotny algorytm, zaimplementowany we wczesnym okresie pracy nad programem na każdej klatce filmu analizował, ile każdy piksel mający większą niż ustalona jasność ma sąsiadów o tej samej własności, a następnie wybierał najjaśniejszy piksel mający przynajmniej minimalną (2 lub 3) liczbę sąsiadów. Pozwalało to na szybkie rozpoznanie, czy na danej klatce występuje interesujące zdarzenie kosmiczne.

W celu umożliwienia wykrywania obecności więcej niż jednego zdarzenia na pojedynczej klatce, co jest rzadkim i interesującym wydarzeniem, postanowiono zmienić algorytm wykrywający zdarzenia. Nowy algorytm w pierwszej kolejności grupuje piksele ponad danym poziomem jasności w grupy w których każde dwa piksele łączy ścieżka przechodząca tylko przez piksele z danej grupy, a następnie znajduje najjaśniejszy

piksel w każdej grupie o wielkości większej lub równej wcześniej wspomnianej minimalnej liczby sąsiadów plus jeden (plus jeden pochodzi od piksela od którego liczyliśmy sąsiadów).

5 Przebudowa algorytmu wyszukiwania interesujących zdarzeń

Pierwotny algorytm klasyfikacji zdarzeń, zaimplementowany we wczesnym okresie pracy nad programem, wykorzystywał współrzędne skrajnych pikseli do przypisania zdarzeniu kształtu - uderzenia prostopadłego, pionowego, poziomego lub skośnego. Celem było przedstawienie koncepcji i zainteresowanie młodzieży zaangażowanej w projekt CREDO EDU.

W ramach rozwijania programu i analizy pierwszych wyników okazało się, że taka klasyfikacja jest niewystarczająca. Zdarzenia prostopadłe i liniowe są bowiem bardzo częste (a przez to niezbyt interesujące) natomiast zakręcające lub „chmurowe” (tak nazwaliśmy zdarzenia o dużej powierzchni i średniej jasności) są znacznie rzadsze. Naturalną konsekwencją była potrzeba stworzenia algorytmu, który byłby w stanie je sklasyfikować. Rozważano dwie grupy algorytmów:

1. algorytm NEWS - algorytm mający oszacować kształt zdarzenie, podróżując po jego krawędzi,
2. algorytm cięciwowy - wyszukujący dwa skrajne piksele i rozciągający między nimi prostą, a następnie porównujący ilość pikseli nad nią i pod nią,

Po rozważeniu różnych przypadków zdecydowaliśmy się na zastosowanie algorytmu cięciwowego, który dzieli zdarzenia na: punktowe, chmurowe, proste i zakrzywione.

6 Testy programu po zaimplementowanych zmianach

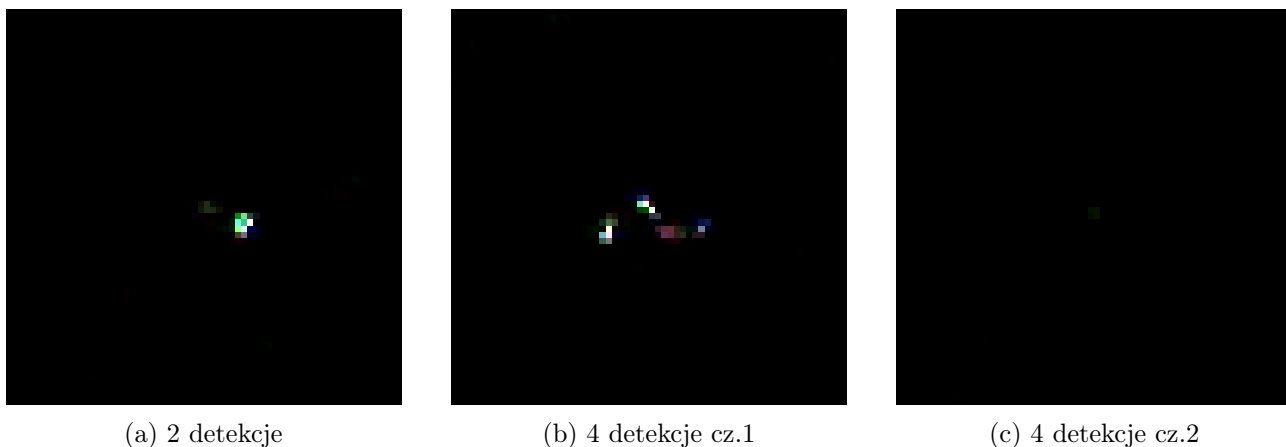
Ostateczna wersja programu została przetestowana. Głównym celem testu było sprawdzenie poprawnego wykrywania wielu zdarzeń na jednej klatce oraz klasyfikowania detekcji na podstawie ich kształtu. Do testów wykorzystano źródła radioaktywne, aby zwiększyć ilość zdarzeń i prawdopodobieństwo wielokrotnych detekcji. Dodatkowo źródła były ustawione pod kątem do kamery, aby zwiększyć liczbę zdarzeń innych niż typu perpendicular (padających prostopadle do kamery).

Podczas wszystkich testów kamera była ustawiona na stojaku, a jej obiektyw był skierowany pionowo do góry. Do pierwszego testu użyto Cs-137. Źródło znajdowało się w odległości około 3,5 cm w pionie od kamery i było przesunięte o około 3 cm w poziomie względem jej środka. Czas pomiaru wynosił 20 minut. Liczba detekcji wynosiła 4, z czego 2 były na jednej klatce. Wszystkie były typu perpendicular.

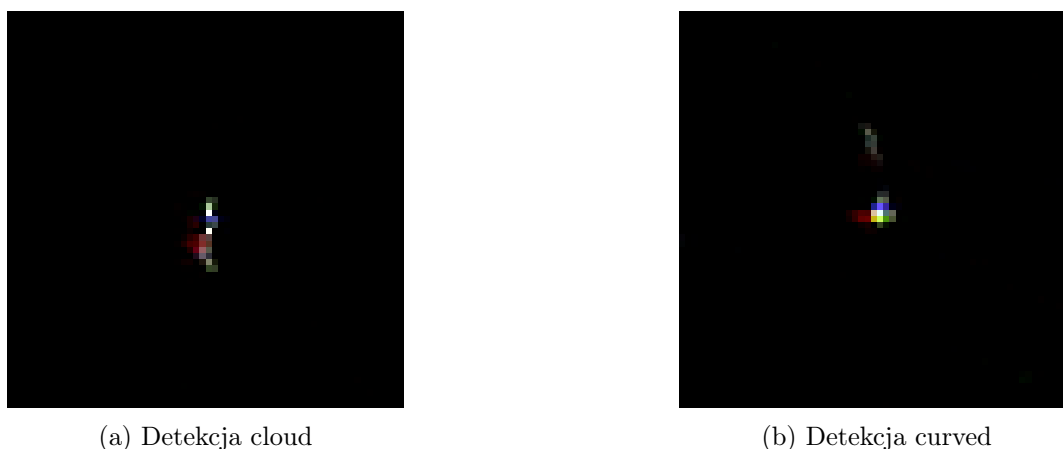
Do drugiego testu użyto Co-57. Źródło znajdowało się w odległości około 3 cm w pionie od kamery i było przesunięte o około 2,5 cm w poziomie względem jej środka. Czas pomiaru wynosił 20 minut. W tym czasie nastąpiła tylko jedna detekcja, która była typu perpendicular.

Do trzeciego testu użyto Co-60. Źródło znajdowało się obok kamery, to znaczy na jej wysokości i było przesunięte o około 3 cm w poziomie względem jej środka. Czas pomiaru wynosił 50 minut. Liczba detekcji wynosiła 74, w tym 10 klatek miało po 2 detekcje w tym jedna podwójna detekcja została przedstawiona na Rysunku 4.a, a 1 klatka miała 4, przedstawiona na Rysunku 4.b i 4.c.

W detekcjach Co-60 zostały zaklasyfikowane 66 detekcji typu perpendicular, 2 detekcje typu cloud w tym przedstawione na Rysunku 5.a oraz 6 detekcji typu curver w tym przedstawiona na Rysunku 5.b.



Rysunek 4: Wybrane detekcje wielu zdarzeń na jednej klatce



Rysunek 5: Wybrane detekcje o nietypowych kształtach

Po zakończeniu pomiarów przejrzano wszystkie klatki w celu oceny klasyfikacji zarejestrowanych zdarzeń. Program poprawnie klasyfikuje zdarzenia oraz rozróżnia wiele zdarzeń na jednej klatce.

7 Podsumowanie

W ramach projektu udało się zrealizować jego główne założenia, przekształcając aplikację w stabilne, wielowątkowe narzędzie edukacyjne i pełnoprawnego klienta IoT zintegrowanego z systemem CREDO. Wprowadzenie interfejsu graficznego znacząco uprościło konfigurację i obsługę programu, eliminując barierę wejścia związaną z pracą w terminalu. Pod kątem analitycznym z powodzeniem zaimplementowano nowe algorytmy: system bezbłędnie izoluje i wykrywa wiele zdarzeń na pojedynczej klatce obrazu oraz klasyfikuje je na podstawie kształtu z wykorzystaniem algorytmu cięciwowego. Poprawność działania zmodernizowanej aplikacji i jej algorytmów została pomyślnie zweryfikowana w warunkach laboratoryjnych podczas testów z użyciem źródeł promieniotwórczych (Cs-137, Co-57, Co-60). Oprogramowanie jest w pełni funkcjonalne i gotowe do wykorzystania w warsztatach szkolnych.

Bibliografia

- [1] CREDO Science, "CREDO Web App API v2", GitHub. Dostępne na: <https://github.com/credo-science/credo-webapp/tree/master/credoapiv2> (dostęp: 1 października 2025).
- [2] CREDO Science, "CREDO Edu Team", API CREDO. Dostępne na: https://api.credo.science/web/team/credo_edu/ (dostęp: 1 października 2025).