

Luneta — poradnik

Dla graczy: jak zainstalować, połączyć z ariku.pl i co Luneta robi z danymi.

- [Start](#)
- [Instalacja](#)
- [Aktualizacje](#)
- [Problemy](#)
- [Prywatność i fair play](#)
- [Sumy SHA256](#)
- [Regulamin](#)
- [Mapowanie ekranów](#)
- [Pakiety gier](#)
- [Ekran startowy \(splash\)](#)

Start

“ Źródło: [repo luneta/packaging/releases-repo/wiki/Home.md](https://github.com/luneta/packaging/releases-repo/wiki/Home.md) — edytuj w repo, nie tutaj.

Luneta

Kompan do gier w trayu: czyta liczby z okna gry lokalnym OCR, sam odhacza dniówki, pilnuje walut, pity i biblioteki postaci. Gry: **Blue Archive**, **NTE**.

“ Luneta obserwuje. Luneta wyjaśnia. Człowiek działa.

Luneta niczego nie klika i nie wysyła do gry klawiszy, nie czyta pamięci gry i nie wstrzykuje kodu. Robi zrzut samego okna gry, a OCR liczy na Twoim komputerze.

Strony

- [Instalacja](#) — Windows, Appliance, .deb, Arch
- [Aktualizacje](#) — tray, wersje zwykłe i poprawkowe
- [Sumy SHA256](#) — jak sprawdzić pobrany plik
- [Prywatność i fair play](#) — co Luneta robi, a czego nigdy
- [Problemy](#) — najczęstsze kłopoty

Linki

- pobieranie: [najnowsze wydanie](#)
- konto i token: ariku.pl/luneta

Instalacja

“ Źródło: [repo luneta/packaging/releases-repo/wiki/Instalacja.md](#) — edytuj w repo, nie tutaj.

Pliki są w [najnowszym wydaniu](#).

Windows 10/11

1. Pobierz `Luneta_<wersja>_x64-setup.exe` i uruchom.
2. Windows SmartScreen może ostrzec o nieznanym wydawcy (instalator nie jest podpisany) — „Więcej informacji” → „Uruchom mimo to”. Jeśli wolisz, najpierw [sprawdź sumę](#).
3. Luneta zamieszka w trayu (obok zegara).

Linux — AppImage (każda dystrybucja)

```
chmod +x Luneta_<wersja>_amd64.AppImage
./Luneta_<wersja>_amd64.AppImage
```

AppImage aktualizuje się sam z menu traya (podmienia plik i uruchamia się ponownie). Potrzebny działający tray (np. XFCE, KDE; w GNOME rozszerzenie AppIndicator).

Debian / Ubuntu

```
sudo apt install ./luneta_<wersja>_amd64.deb
```

Aktualizacja z traya pobiera nowy `.deb` i otwiera go w instalatorze systemu.

Arch

Paczka z wydania:

```
sudo pacman -U luneta-bin-<wersja>-1-x86_64.pkg.tar.zst
```

albo z AUR: `luneta-bin`. Nowe wersje instalujesz pacmanem / helperem AUR.

Pierwsze uruchomienie

1. Zaakceptuj regulamin (ekran przy pierwszym starcie).
2. Na ariku.pl/luneta zaloguj się przez Discord → **utwórz token**.
3. W Lunecie: ustawienia → wklej token → Enter. Token trafia do systemowego sejfu haseł.
4. Uruchom grę w **1920×1080** (inne rozdzielczości są skalowane, mniej pewnie).
5. Na górze okna Lunety kliknij **auto-skan** (albo F6 w grze).

Bez tokenu Luneta też działa lokalnie — token jest tylko do synchronizacji z ariku.pl.

Aktualizacje

„ Źródło: [repo luneta/packaging/releases-repo/wiki/Aktualizacje.md](https://github.com/luneta/packaging/releases-repo/wiki/Aktualizacje.md) — edytuj w repo, nie tutaj.

Jak to działa

Luneta sprawdza [repo wydań](#) przy starcie i co 6 godzin. Gdy jest nowsza wersja:

- powiadomienie systemowe i wpis w zakładce „aktywność”,
- w menu traya pozycja „**aktualizuj do X.Y.Z**”,
- w ustawieniach przycisk **aktualizuj**.

Nic nie instaluje się samo — aktualizację zaczynasz Ty, kliknięciem.

Po kliknięciu Luneta pobiera plik dla Twojego systemu, sprawdza jego **sumę SHA256** (z pliku `.sha256` obok w wydaniu) i dopiero wtedy:

instalacja	co się dzieje
Windows	uruchamia instalator, Luneta zamyka się na czas aktualizacji
ApplImage	podmienia plik i uruchamia się ponownie
.deb	otwiera paczkę w instalatorze systemu
inna (pacman, ręczna)	otwiera stronę wydań

Niezgodna albo brakująca suma = aktualizacja przerwana, stara wersja zostaje.

Gdy git.ariku.pl nie odpowiada, Luneta bierze wydanie z kopii na ariku.pl.

Wersje

Numer `X.Y.Z`:

- **Y / X** (np. 0.13.0, 1.0.0) — nowe funkcje.

- **Z > 0** (np. 0.13.1) — poprawki.

Od 0.15.2 każda wersja jest zwykłym wydaniem: Luneta zawsze proponuje najnowszą, a ariku.pl pokazuje tylko ją.

Powrót do starszej wersji

Każda wersja zostaje w [wydaniach](#) — pobierz starszy instalator i zainstaluj go ręcznie. Dane lokalne zostają, ale starsza wersja może nie znać zmian zapisanych przez nowszą.

Problemy

“ Źródło: [repo luneta/packaging/releases-repo/wiki/Problemy.md](https://github.com/luneta/packaging/releases-repo/wiki/Problemy.md) — edytuj w repo, nie tutaj.

Nie widzę ikony w trayu (Linux). Twoje środowisko musi obsługiwać tray (AppIndicator). W GNOME zainstaluj rozszerzenie „AppIndicator and KStatusNotifierItem Support”.

Luneta nie wykrywa gry. Gra musi być uruchomiona (Blue Archive albo NTE). Auto-skan pauzuje, gdy okno gry jest w tle — na nakładce widać „gra w tle — pauza”.

Złe odczyty liczb. Ustaw grę na 1920×1080. Zły ekran zgłoś w grze klawiszem F8 (albo F5 — zrzut z opisem); zgłoszenie zostaje lokalnie do poprawek.

„suma sha256 się nie zgadza” / „brak sumy sha256”. Pobieranie przerwane albo wydanie jeszcze się wgrywa. Spróbuj za kilka minut („sprawdź teraz” w ustawieniach). Stara wersja działa dalej.

Aktualizacja nie przychodzi. Ustawienia → „sprawdź teraz”. Instalacje z pacmana/AUR i ręczne nie aktualizują się same — instaluj nową paczkę z [wydań](#).

Token „unieważniony (401)”. Wygeneruj nowy na ariku.pl/luneta i wklej w ustawieniach. Dane czekają w lokalnej kolejce i wyślą się po zmianie tokenu.

Log. Ustawienia → na dole ścieżka do pliku logu.

Prywatność i fair play

“ Źródło: [repo luneta/packaging/releases-repo/wiki/Prywatnosc-i-fair-play.md](https://github.com/luneta/packaging/releases-repo/wiki/Prywatnosc-i-fair-play.md) — edytuj w repo, nie tutaj.

Fair play

Luneta jest narzędziem informacyjnym, nie botem ani cheatem.

- **Nie** klika w grze, nie wysyła klawiszy, ruchów myszy ani wejścia kontrolera.
- **Nie** czyta ani nie zmienia pamięci gry, nie wstrzykuje kodu, nie przechwytuje ruchu sieciowego.
- **Nie** odbiera nagród, nie wykonuje zadań, nie nawiguje po menu.
- **Nie** obchodzi anti-cheatu i niczego przed nim nie ukrywa.

Nakładka nad grą to osobne, przezroczyste okno systemowe — kliknięcia przechodzą przez nie do gry, nic nie jest rysowane w procesie gry. Można ją wyłączyć.

Ostatnią akcję w grze zawsze wykonujesz Ty.

Prywatność

- Zrzut obejmuje **tylko okno gry**, a przy auto-skaniu tylko gdy okno gry jest aktywne.
- Surowy zrzut **nie opuszcza komputera** i nie trafia na dysk (wyjątek: lokalny dziennik problemów, który można wyłączyć w ustawieniach).
- OCR jest lokalny i deterministyczny — **nic nie idzie do AI**.
- Na serwer ariku.pl (po podaniu tokenu) idą same liczby: waluty, odhaczenia zadań.
- Wyniki analizy nagrań, transkrypcje i notatki — tylko za osobną zgodą w ustawieniach, domyślnie wyłączone. Same nagrania nigdy nie wychodzą z komputera.
- Token trzymany jest w systemowym sejfie haseł (Secret Service / Menedżer poświadczeń).

Szczegóły: regulamin na ariku.pl/luneta.

Sumy SHA256

“ Źródło: [repo luneta/packaging/releases-repo/wiki/Sumy-SHA256.md](https://github.com/luneta/packaging/releases-repo/wiki/Sumy-SHA256.md) — edytuj w repo, nie tutaj.

Każde wydanie ma plik `SHA256SUMS` oraz `<plik>.sha256` obok każdego instalatora. Luneta przy aktualizacji z traya sprawdza sumę sama. Przy ręcznym pobraniu:

Linux

W katalogu z pobranymi plikami (pobierz też `SHA256SUMS`):

```
sha256sum -c SHA256SUMS --ignore-missing
```

Oczekiwany wynik: `<plik>: OK`.

Windows (PowerShell)

```
(Get-FileHash .\Luneta_<wersja>_x64-setup.exe -Algorithm SHA256).Hash.ToLower()
```

Porównaj z pierwszą kolumną w `Luneta_<wersja>_x64-setup.exe.sha256`.

Inna suma = plik uszkodzony albo podmieniony. Nie instaluj, pobierz ponownie z git.ariku.pl.

Regulamin

„ Źródło: [repo luneta/docs/REGULAMIN.md](#) — edytuj w repo, nie tutaj.

Luneta — regulamin używania

Wersja 3 z 2026-10-14. Instalując albo uruchamiając Lunetę, akceptujesz ten regulamin. Jeśli się z nim nie zgadzasz, nie instaluj aplikacji, a jeśli już ją masz, usuń ją i swoje konto.

1. Kto udostępnia

Lunetę udostępnia ariku (ariku.pl), nieodpłatnie i bez gwarancji. To nie jest usługa komercyjna. Nie ma umowy, obowiązku wsparcia ani zapewnionej dostępności serwera.

Kod aplikacji jest na licencji MIT (plik `LICENSE` dołączony do instalatora): możesz jej używać, kopiować i zmieniać, z zachowaniem informacji o autorze. Ten regulamin dotyczy korzystania z aplikacji razem z kontem i serwerem ariku.pl.

2. Dostęp

- Instalator jest publicznie dostępny na ariku.pl. Bez konta Luneta działa lokalnie.
- Synchronizacja z ariku.pl wymaga zalogowania kontem Discord na ariku.pl i powiązania urządzenia. Dostęp do konta i serwera może zostać cofnięty w każdej chwili, bez podania przyczyny.
- Konto i token urządzenia są osobiste. Nie przekazuj ich innym.

3. Co robi aplikacja

- Robi zrzuty okna gry (na żądanie albo w auto-skanie), czyta z nich liczby (OCR) i zapisuje wyniki na twoim koncie.
- Analizuje twoje nagrania (np. z OBS), a po włączeniu transkrypcji zamienia mowę na tekst.
- Opcjonalnie inny twój komputer z Lunetą w sieci lokalnej może liczyć mowę z nagrań (`luneta peer`). Dostaje wtedy samo audio, nic nie zapisuje na stałe i odsyła tekst tylko do ciebie.
- Po osobnym włączeniu udostępniania Luneta może wysłać na serwer kandydującą paczkę modelu rozpoznawania: manifest i wyuczone punkty ekranów. Nie zawiera ona nagrań, surowych klatek ani ścieżek do twoich plików. Drugi przełącznik może wysyłać paczkę automatycznie po nauce. Paczki candidate nie aktualizują automatycznie innych komputerów; właściciel musi jawnie zatwierdzić wersję stable.
- Domyślnie wszystko zostaje na twoim komputerze. Na serwer trafia tylko to, co sam włączysz w ustawieniach. Każdą z tych zgód możesz w każdej chwili wyłączyć.

4. Nagrania i cudze głosy — twoja odpowiedzialność

Na twoich nagraniach mogą być głosy i wypowiedzi innych osób (np. rozmówców z Discorda). Włączając transkrypcję albo wysyłkę transkrypcji lub notatek o osobach na serwer (same nagrania nigdy nie trafiają na serwer):

- oświadczasz, że masz prawo przetwarzać te nagrania, a osoby na nich wiedzą, że są nagrywane, i się na to zgodziły;
- bierzesz pełną odpowiedzialność za treść, którą przetwarzasz i wysyłasz;
- zobowiązujesz się pokryć roszczenia osób trzecich i ich koszty, jeśli wynikają one z twoich nagrań lub danych, a nie z błędu aplikacji.

Nie nagrywaj ludzi bez ich wiedzy i nie używaj Lunety do śledzenia ani nękania kogokolwiek. Złamanie tej zasady oznacza utratę dostępu i usunięcie twoich danych z serwera.

5. Gry

Luneta nie czyta pamięci gry, nie przechwytuje ruchu, nie wstrzykuje kodu i nie klika za ciebie. Mimo to za zgodność z regulaminem każdej gry odpowiadasz ty. Autor nie odpowiada za decyzje wydawców gier (np. blokady kont).

6. Dane na serwerze

- Dane są przechowywane do czasu, aż je usuniesz albo zostanie usunięte twoje konto.
- Możesz je usunąć z aplikacji albo prośbą przez Orin.
- Wysłaną paczkę nauki możesz usunąć z Lunety. Zatwierdzenie jej jako stable udostępnia model wszystkim klientom; usunięcie własnego konta usuwa także twoje wysłane paczki.
- Nie ma gwarancji kopii zapasowych. Ważne rzeczy trzymaj też u siebie.

7. Zgłoszenia od innych osób

Każdy, kto uważa, że jego głos, wypowiedź albo dane trafiły na serwer Lunety, może poprosić o usunięcie (przez Orin albo kontakt na ariku.pl). Takie dane zostaną usunięte, a konto użytkownika, który je wysłał, może zostać zablokowane.

8. Brak gwarancji

Aplikacja działa „tak jak jest”. OCR i analiza mogą się mylić. W najszerszym zakresie, na jaki pozwala prawo, autor nie odpowiada za szkody wynikłe z używania Lunety, utratę danych ani niedostępność serwera.

9. Zmiany

Regulamin może się zmienić. Nowa wersja pokaże się w aplikacji. Dalsze używanie oznacza jej akceptację. Jeśli ci nie pasuje, możesz usunąć aplikację i konto.

10. Prawo

Obowiązuje prawo polskie.

Mapowanie ekranów

“ Źródło: repo luneta/docs/MAPOWANIE.md — edytuj w repo, nie tutaj.

Mapowanie ekranów: F-klawisze i paczka sesji

Decyzja właściciela z 2026-10-06. Cel: mapować ekrany i zgłaszać problemy bez wychodzenia z gry, a po sesji oddać Claude Code jedną zwięzłą paczkę zamiast setek klatek.

F-klawisze (tylko gdy wykryta gra działa)

klawisz	akcja
F5	zgłoś : zrzut okna gry + okienko przy kursorze (opis, potem typ)
F6	auto-skan i nakładka razem (któreś wyłączone → oba wł., inaczej oba wył.)
F7	panel mapowania (debug/normal), etykiety zmiennych „tu jest X”
F8	opis do ostatniego zgłoszenia z F5
F9	okno Lunety pokaż/schowaj
F10	wolne na przyszłość

Poza grą klawisze są zwolnione (globalny skrót zabierałby np. F5 przeglądarce). Włącznik: ustawienia → „F5 zgłoś...”. Kod: `src-tauri/src/fkeys.rs`, `src-tauri/src/report.rs`.

Każda akcja daje komunikat w nakładce (jeśli włączona) i w zakładce „aktywność”. Zero wejścia do gry: okienko zgłoszenia to zwykłe okno pulpitu, chwilowo zabiera fokus.

Zgłoszenie (F5)

1. F5 — Luneta robi zrzut okna gry **w tej chwili** (zanim pojawi się okienko).
2. Przy kursorze otwiera się okienko: wpisz, o co chodzi (opcjonalnie), potem kliknij typ (albo Ctrl+1...5):
 - **nowy ekran / do mapowania** — ekran, którego Luneta nie zna;
 - **zły odczyt** — zła liczba, zły ekran, zły fakt;
 - **zadanie nie odhaczone / błędnie**;
 - **pomysł na funkcję** — np. „pokaż tu taktyki”;
 - **inne**.
3. Klik zapisuje klatkę i wpis do lokalnego dziennika problemów (`<dane>/review/`, `kind:` `manual`, `data.category`). Esc albo 60 s bez wyboru: nic nie trafia na dysk.

F8 otwiera to samo okienko z samym polem opisu dla ostatniego zgłoszenia (dziennik jest tylko dopisywany, więc opis to osobny wpis `kind: note` z `data.ref`; `luneta review` i paczka sklejają je ze zgłoszeniem). Zgłoszenia nie idą na serwer.

Paczka sesji

```
luneta session-pack <gra> [--since <godz.>|--all] [--frames N] [--learn <katalog>...]
```

albo przycisk **paczka sesji** w zakładce „aktywność” (gra wybrana w dniówkach).

- Okres: od ostatniej paczki tej gry, najdalej 24 h wstecz (`--since 3` = ostatnie 3 h, `--all` = cały dziennik).
- Zawartość `session.md`: zgłoszenia pogrupowane po typie (z opisami F8), etykiety z panelu, dziennik automatyczny zliczony po rodzaju i ekranie, wynik toru ekranów (`luneta screens`) na unikalnych klatkach sesji wg **bieżących** reguł i modelu, lista klatek w paczce, opcjonalnie wiersz „sprawdzenie” z `luneta learn` (`--learn mapping/ba-1 ...` ; model trafia do katalogu paczki, nie do `assets/`).
- Klawisze (ustawienie „klawisze”, domyślnie wył.; `CLAUDE.md`): sekcja „klawisze” — najczęstsze per ekran i sekwencja z 15 s przed każdym zgłoszeniem F5. Źródło `<dane>/review/<dzień>/keys.jsonl`, zapisywane tylko przy aktywnym oknie gry i działającym auto-skaniu.
- Klatki: najwyżej N (domyślnie 8), **tylko nierozpoznane** — zgłoszenia „nowy ekran”/„zły odczyt”, `new_screen`, `anchor_mismatch`, `unknown_screen`, którego bieżące reguły nadal nie rozpoznają. Prawie identyczne (dHash < 12 bitów) odpadają. Przeskalowane do 1280 px, JPEG 80. Logika wyboru: `luneta_lib::core::session_pack` (z testami).

- Miejsce: `<dane>/review/packs/<data>-<gra>/` (Linux: `~/.local/share/pl.ariku.luneta/review/packs/`). Zostaje 10 ostatnich paczek. Nic nie jest wysyłane.

Szacunek tokenów jest w nagłówku `session.md`: tekst $\approx$ znaki / 3, obraz $\approx$ szer.xwys./750 po skalowaniu do 1568 px. Klatka 1920×1080 $\approx$ 1840 tokenów, 1280×720 $\approx$ 1230. Pomiar 2026-10-06 (BA, 24 h, 45 klatek w dzienniku): paczka $\sim$ 11,2 tys. (tekst $\sim$ 1,4 tys. + 8 klatek $\sim$ 9,8 tys.) vs surowe klatki $\sim$ 83 tys.; czas $\sim$ 1 min 45 s (OCR 44 unikalnych klatek).

Przegląd przez Claude Code

Tylko na wyraźną prośbę właściciela, w sesji z nim (CLAUDE.md, wyjątek „dziennik problemów”). Wskaż katalog paczki: najpierw `session.md`, klatki tylko te, do których odsyła tekst.

Pakiety gier

„ Źródło: `repo luneta/docs/GAME-PACKS.md` — edytuj w repo, nie tutaj.

Pakiety gier: obsługa nowej gry bez programowania

Luneta ma dwie wbudowane gry: Blue Archive i NTE. Kolejną grę dodasz **pakiem**. Pakiet to katalog z plikiem `game.json`. Opisujesz w nim, gdzie na zrzucie okna gry są liczby i co z nich wynika. Agent wczytuje pakiety przy starcie.

Pakiet nie zawiera kodu ani skryptów i nie ma dostępu do sieci. Opisuje tylko obraz okna gry, więc zasady bezpieczeństwa Lunety obowiązują bez zmian: zero czytania pamięci gry, zero wstrzykiwania, zero klikania za gracza.

Gdzie leżą pakiety

system	katalog
Linux	<code>~/.local/share/pl.ariku.luneta/games/<klucz>/</code>
Windows	<code>%APPDATA%\pl.ariku.luneta\games\<klucz>\</code>
test / dev	katalog wskazany zmienną <code>LUNETAS_GAMES_DIR</code>

W katalogu pakietu są dwa pliki:

- `game.json`: opis gry (piszesz sam);
- `screens.json`: model punktów charakterystycznych ekranów (generuje `luneta learn`, opcjonalny).

Przykład: `games/nte_demo/game.json`. Sprawdzisz go poleceniem `LUNETAS_GAMES_DIR=games luneta screens nte_demo <klatki>`.

Krok po kroku

1. **Szkielec:** `luneta game new moja_gra --label "moja gra" --process MojaGra.exe`. Klucz musi być taki sam jak klucz gry na serwerze, bo stamtąd przychodzą waluty i zadania.
2. **Proces:** uruchom grę i wpisz `luneta detect`. Do `processes` wpisz nazwę pliku `.exe` z listy. Popraw `windowTitles`, jeśli trzeba.
3. **Klatki:** `luneta map-session --game moja_gra --minutes 20`. Chodź po ekranach, które chcesz śledzić. Klatki zapisują się lokalnie w `mapping/`.
4. **Tekst z klatek:** `luneta ocr-lines <klatka.jpg>` wypisze linie OCR z pozycjami `[x, y, szerokość × wysokość]`. Z nich wybierzesz słowa kluczowe ekranów i obszary pól.
5. **Opis ekranów i pól** w `game.json` (format niżej). Sprawdzaj na bieżąco: `luneta screens moja_gra mapping/<katalog>`, a dla całego pliku `luneta game check moja_gra`.
6. **Punkty charakterystyczne:** `luneta learn moja_gra ~/.local/share/pl.ariku.luneta/games/moja_gra/screens.json mapping/<katalog>`. Wiersz `sprawdzenie:` ma pokazać `inaczej 0`.
7. **Restart agenta.** Gra pojawi się w dniówkach, auto-skane i wykrywaniu.

Format `game.json`

Wszystkie współrzędne podajesz w pikselach klatki **1920×1080** jako `[x0, y0, x1, y1]`. Inna rozdzielczość jest skalowana do tej.

```
{
  "key": "moja_gra",
  "label": "moja gra",
  "processes": ["MojaGra.exe"],
  "windowTitles": ["Moja Gra"],
  "screens": [
    { "name": "*", "fields": [
      { "key": "gems", "kind": "currency", "rect": [1600, 20, 1760, 70] }
    ] },
    { "name": "Dungeon", "keywords": ["Dungeon"], "any": ["Enter", "Sweep"],
      "keywordArea": [0, 0, 600, 120],
      "fields": [
        { "key": "keys", "kind": "counter", "area": [1300, 900, 1900, 1000], "pattern":
"Keys\\s*(\\d+)\\s*/\\s*(\\d+)" }
      ] }
  ],
  "rules": [
```

```
{ "field": "keys", "task": "daily_dungeon", "done": "zero", "remaining": true }
]
```

Ekrany (screens)

pole	znaczenie
name	nazwa ekranu (dowolna). "*" = pola czytane na każdym ekranie, np. pasek walut
keywords	wszystkie te napisy muszą być na klatce. Wielkość liter ma znaczenie, to fragmenty tekstu z OCR
any	i co najmniej jeden z tych napisów
keywordArea	opcjonalnie: napisy muszą leżeć w tym prostokącie, np. tytuł w rogu

Ekrany sprawdzane są w kolejności z pliku i wygrywa pierwszy pasujący. Dlatego ekrany bardziej szczegółowe wpisz wyżej.

Pola (fields)

kind	wynik	jak czytać
currency	waluta, czyli snapshot na serwer (klucz waluty jak na serwerze)	rect (OCR cyfr w stałym miejscu, najpewniejsze) albo pattern z grupą 1
counter	licznik N/M (bilety, stamina, pity)	rect albo pattern z grupami 1 i 2; przy jednej grupie podaj max. Bez pattern szuka „N/M”
percent	procent	rect albo pattern z grupą 1
timer	czas hh:mm:ss (> 0 = coś w toku)	pattern z 3 grupami
label	tekst, np. nazwa banera	pattern z grupą 1

area zawęży szukanie wzorca do linii OCR, których środek leży w prostokącie. Nie przyspiesza OCR: agent robi jeden OCR całej klatki, bo model detekcji ocrs ma stałe wejście 600×800 i osobny OCR każdego pola kosztuje tyle co cała klatka (pomiar luneta perf 2026-10-05).

pattern to wyrażenie regularne (składnia Rust regex). W JSON odwrotny ukośnik piszesz podwójnie: \\d.

Reguły (rules): wnioski o zadaniach

pole	znaczenie
field	klucz pola
task	klucz zadania na serwerze (dniówki, tygodniówki)
done	zero (0/N, np. bilety zużyte), full (N/N), below_max (< N, np. była walka), nonzero (> 0, np. czas w toku), at_least:<liczba>
remaining	licznik pokazuje, ile zostało (np. „Gift 0/3”). Postęp to wtedy max – wartość

Spełniona reguła odhacza zadanie raz na okres i pokazuje „✓ zaliczone” na nakładce. Niespełniona pokazuje „postęp”.

Dobre praktyki

- **Waluty** czytaj z rect w stałym miejscu. OCR cyfr na małym wycinku jest dokładniejszy niż szukanie wzorcem w całej klatce.
- **Prostokąt** obejmuje samą liczbę, bez ikony. Ikona bywa czytana jako „1”.
- **Słowa kluczowe** wybieraj z napisów, które zawsze są na ekranie: tytuł, zakładki, przyciski. Nie bierz tekstu, który się zmienia, np. nazw banerów.
- **Nowe klatki:** gdy auto-skan czegoś nie rozumie, zapisuje klatkę w dzienniku problemów (luneta review). Te klatki dobrze nadają się do kolejnego luneta learn .
- **Serwer:** gra musi mieć na serwerze ten sam klucz, swoje waluty i zadania. Nieznane serwerowi waluty agent pomija.

Kiedy pakiet nie wystarczy

Logika, której nie da się zadeklarować, wymaga modułu wbudowanego w Ruście (src-tauri/src/modules/ , wpis w builtin()). Przykłady: kolor ikony decyduje o walucie, biblioteka postaci z dopasowaniem do bazy, kilka układów paska zależnie od ekranu. Tak są zrobione Blue Archive i NTE.

Ekran startowy (splash)

“ Źródło: repo luneta/docs/SPLASH.md — edytuj w repo, nie tutaj.

Ekran startowy (splash)

Decyzja właściciela z 2026-10-17. Przy uruchomieniu Lunety pokazuje się przezroczyste okno z jednym z sześciu wariantów, losowanym przy każdym starcie. Rogi są ścięte i półprzezroczyste (widać pulpit pod spodem). Okno jest wyłącznie informacją: nie przyjmuje kliknięć i nie dotyka gry.

Ostatnia aktualizacja: 2026-10-17. Kod: `src-tauri/src/splash.rs`, `src/views/Splash.tsx`, `src/views/splash.css`, `src/components/SplashSettings.tsx`. Grafiki: `src/assets/splash/`, źródła i skrypty: `assets/design/splash/`.

Dla użytkownika

Ustawienia

Zakładka **ustawienia** → **ekran startowy** (zapisuje od razu):

ustawienie	znaczenie
pokazuj splash przy uruchomieniu	włącznik; domyślnie włączony
wybór wariantu	losowy (bez powtórki poprzedniego), po kolei albo stały
wariant	używany w trybie „stały” i w podglądzie
czas widoczności (suwak)	1-10 s , domyślnie 7 s: minimum od pokazania okna; potem splash zamyka się, gdy wszystko jest gotowe
▶ podgląd wariantu	pokazuje wybrany wariant od razu, także przy wyłączonym splashu

Splash nigdy nie wisi dłużej niż 15 s (zabezpieczenie), nawet gdyby coś poszło nie tak.

Warianty

id	tytuł	akcent
terminal_boot	1/6 Terminal boot	wariant kanoniczny i zapasowy
world_map	2/6 Mapa świata	miasto o zmierzchu, holograficzna mapa
system_scan	3/6 Analiza systemu	panel z modułami gier
api_connect	4/6 Łączenie z API	panel połączenia i autoryzacji ariku.pl
game_modules	5/6 Ładowanie modułów gier	lista modułów i licznik
ready	6/6 Gotowe	komunikat „Luneta jest gotowa.” po zakończeniu startu

Co znaczą statusy

Wszystkie statusy są prawdziwe, nic nie jest wypalone w obrazku:

krok	skąd
baza lokalna	odczyt kolejki SQLite
silnik OCR	obecność modeli OCR (brak = ostrzeżenie, praca lokalna nadal możliwa)
moduły gier	liczba wczytanych gier (wbudowane i pakiety)
ariku.pl	test połączenia: tylko odczyt zgód (GET .../me/consents), timeout 10 s
autoryzacja	czy serwer przyjął token
nakładka	czy nakładka na ekranie jest włączona

Stany: (OK), (trwa), (ostrzeżenie), (błąd), (nie dotyczy). Zasady:

- Do czasu odpowiedzi serwera ariku.pl i autoryzacja to „trwa”. Timeout to ostrzeżenie, nigdy sukces.
- **Brak sieci** to „tryb offline” (ostrzeżenie), a **401** to „token odrzucony” (błąd autoryzacji, połączenie jest). Brak tokenu to ostrzeżenie, nie blokada.
- Pasek postępu jest nieokreślony, bo realnego procentu nie mamy. „Gotowe.” pojawia się dopiero, gdy żaden krok już nie trwa; z ostrzeżeniami splash pisze, czego dotyczą („działa lokalnie”).

Prywatność i fair play

Okno jest zwykłym oknem pulpitu: nic nie jest wstrzykiwane do gry ani rysowane w jej procesie, kliknięcia przechodzą przez nie dalej. Do sieci idzie jedno zapytanie tylko do odczytu (zgody). Szanuje `prefers-reduced-motion` systemu (wyłącza mruganie i pasek w ruchu).

Dla deweloperów

Jak to działa

1. `splash::spawn_startup` (z `setup`) czyta ustawienia, wybiera wariant (`pick`, czysta funkcja, testowana), zapisuje `splash_last` i tworzy okno `splash` (960×420, przezroczyste, bez ramki).
2. Okno startuje **ukryte**. Pierwsze `splash_status` z frontu pokazuje je (`reveal`: rozmiar, środek, `show`, zawsze na wierzchu, kliknięcia przechodzą). Równolegle idzie test API.
3. Front odpytuje `splash_status` co 350 ms. Zamyka okno (`splash_close`), gdy `ready` i minął minimalny czas widoczności (`minShowMs` z suwaka, liczony od pierwszych danych), najwcześniej 300 ms po gotowości. Watchdog w Rust zamyka okno po 15 s.
4. Stan API trzyma `API_RESULT` (`Idle`, `Loading`, `Done`); `classify_api` rozdziela połączenie i autoryzację, bez treści odpowiedzi serwera w szczegółach.

Klucze ustawień: `splash` (0/1), `splash_mode` (`random` | `fixed` | `sequential`), `splash_variant`, `splash_min_s` (1–10), `splash_last`. Komendy: `splash_status`, `splash_close`, `splash_preview`, `splash_config_get`, `splash_config_set`.

Pułapki (każda kosztowała zawieszenie lub puste okno)

- **Komendy dotyczące okna muszą być `async`**. Sync komendy Tauri chodzą na wątku głównym, a operacje na oknie wysyłają tam wiadomości i czekają.
- **Nie trzymaj `guardów mutexu` w literale struktury**. `st.store.lock()` w `Inputs { ... }` żyje do końca instrukcji, a `toast::enabled` blokuje ten sam mutex: samo-zakleszczenie, które zamraża całą aplikację. Odczyty rób w osobnych `let`.
- **Ukryte okno GTK ma 10×10**, dopóki rozmiar nie zostanie ustawiony jawnie (jak w `toast::place`).
- **Timery JS w ukrytym oknie są wstrzymane** (też `requestAnimationFrame`). Sygnał „pokaż” idzie z Rust przy pierwszym `splash_status`, nie z `setTimeout`.
- **Okno z przezroczystością tworzymy ukryte** i pokazujemy po załadowaniu treści; pokazane przed treścią nie maluje się w WebKitGTK.
- `is_visible()` nie jest wiarygodne dla okna utworzonego jako ukryte: używamy flagi `REVEALED`.

Budowanie do testów na żywo

`cargo build --release` daje binarkę, która ładuje front z `http://localhost:1420` (okno pokaże „Could not connect to localhost”). Wersję z wbudowanym frontem robi `npm run build && npx tauri build --no-bundle` (`docs/BUDOWANIE.md`). Do szybkiej iteracji nad CSS można zbudować `cargo build --release` i uruchomić `npx vite --port 1420` obok.

Grafiki

Tła to 6 plików WebP (1920 px, bez tekstu) w `src/assets/splash/`. Powstały z kolażu referencyjnego: cięcie na kafle, usunięcie wypalonego tekstu i UI (`clean_text.py`, inpainting), upscale RealESRGAN x4 anime (`upscale.py`, `spandrel` na XPU, bez serwera ComfyUI). Opis przebiegu, strefy napisów i lista rzeczy do usunięcia (m.in. logotypy gier):

`assets/design/splash/README.md`.

Podmiana grafiki: zachowaj nazwę pliku (`{0N-<id>.webp}`); proporcje ok. 2,1-2,4:1 (okno ma 2,29:1, reszta jest przycinana `cover`). Pozycje paneli przy grafice (03, 04, 05) są wyliczone pod obecne tła w `splash.css`; po zmianie kadru popraw je ręcznie. Brak pliku = ciemna siatka zamiast pustego okna, a w logu pojawia się ostrzeżenie `splash: brak grafiki ...`.

Testy

`cargo test splash`: wybór wariantu (bez powtórki, zawijanie, zapas), rozróżnienie offline / 401 / 5xx / odrzucone, statusy „trwa” bez fałszywego sukcesu, brak tokenu jako ostrzeżenie, zakres suwaka 1-10 s. Logi diagnostyczne okna: `splash: ...` (start, wariant, rozmiar widoku, błędy).

Czego nie ma

Splash przy starcie sesji gry (brief przewidywał go jako opcję) jest niezaimplementowany.