

Tier 4: Raspberry Pi Robotic/Autonomous 5G Platform

Instrukcja oparta na researchu kodu źródłowego. Procedury, warianty i ograniczenia opisano oddzielnie od deklaracji wymagających testu fizycznego zestawu.

Spis treści

1. Definicja i granice Tier 4	2
2. Dwa rozróżnienia przed montażem	2
3. Elementy platformy	2
4. Montaż i konfiguracja wyjść	3
5. Wdrożenie przez Deployment Tool	4
6. Plik konfiguracji	6
7. Sieć IP/5G i porty aplikacji	6
8. Pierwsze uruchomienie bez autonomii	7
9. Nawigacja, SLAM i AI jako rozszerzenia	7
10. Zabezpieczenia i ograniczenia	7
11. Diagnostyka po wdrożeniu	8
12. Źródła researchu	8

Instrukcja operatora i integratora platformy pokładowej. Stan researchu: 14 października 2026. Zakres potwierdzenia: lokalny kod aplikacji RPi, konfiguracja, skrypty wdrożenia i komponenty AI. Nie wykonano wdrożenia na fizycznym pojeździe ani testu operatora 5G.

1. Definicja i granice Tier 4

Tier 4 to platforma robotyczna/autonomiczna z komputerem pokładowym Raspberry Pi, sensorami, wykonawczym sterowaniem modelu oraz komunikacją IP/5G. Raspberry Pi wykonuje zadania na pojeździe; GCS na PC jest stacją operatora. MCS jest osobnym stosem usług i interfejsem opisanym w odrębnym dokumencie.

SLAM, AI, Hailo, RTK, dodatkowe radio i watchdog są rozszerzeniami lub wariantami tej platformy. Nie tworzą Tier 5 ani Tier 6. Samo dołożenie Hailo nie potwierdza gotowości autonomii.

ELRS/mLRS z CRSF/MAVLink bez komputera pokładowego należy do Tier 3. Integrację radia z Raspberry Pi opisujemy tutaj jako opcjonalny tor komunikacyjny Tier 4, korzystający z procedur właściwych dla danego radia.

2. Dwa rozróżnienia przed montażem

Podział procesów: `core/supervisor.py` nadzoruje proces aplikacji, `core/main_service.py` koordynuje usługi, sensory i komendy. Kontener aplikacji i hostowy MediaMTX są osobnymi elementami sprawdzanego wariantu uruchomienia.

Podział usług: pokładowy endpoint WebRTC /offer na porcie 8080 nie jest tym samym serwerem co WebUI MCS na porcie 8080. Nie uruchamiaj obu usług na tym samym adresie i porcie bez rozdzielenia konfiguracji. Sprawdzany `docker-compose.yml` używa `network_mode: host`, więc sam podział na kontenery nie usuwa kolizji portów. Obecność pliku `web_service.py` z panelem HTTP nie dowodzi uruchomienia tego panelu w danym entypointcie.

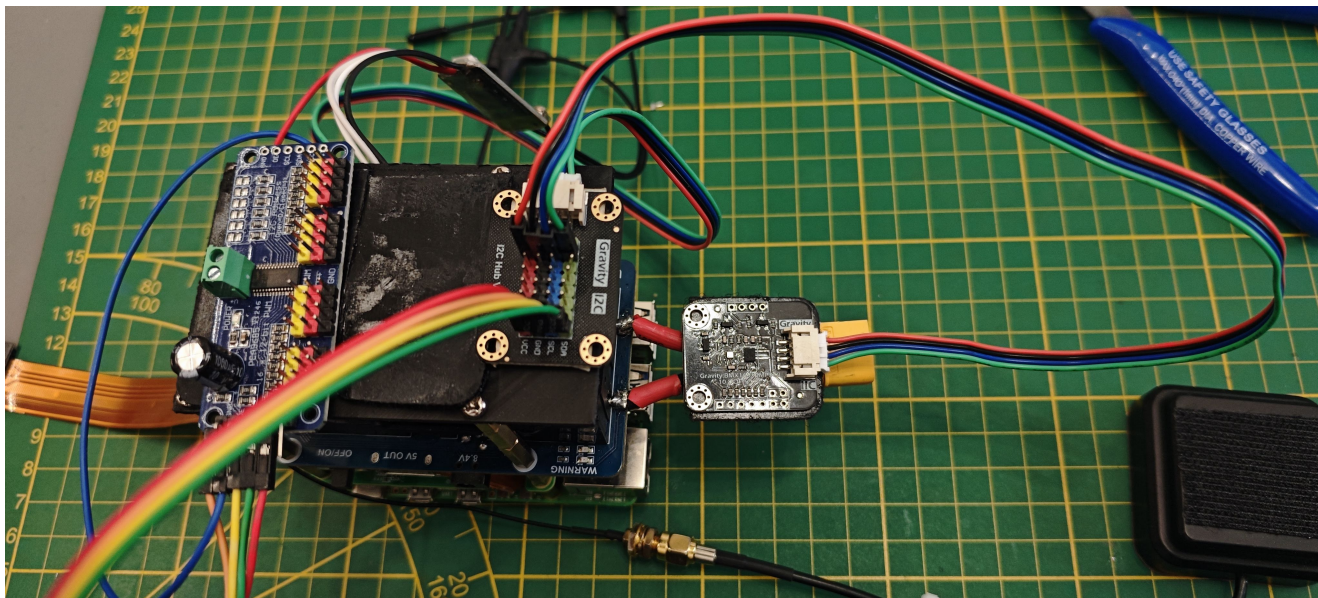
3. Elementy platformy

Element	Rola	Warunek użycia
Raspberry Pi i system ARM64	Procesy pokładowe	Zgodność obrazu systemu, bibliotek i dostępu do urządzeń.
PCA9685 i tor zasilania serw	PWM skrętu/gazu	Zgodne napięcia logiki i zasilania wykonawczego, osobno sprawdzone limity.
IMU/GPS/LiDAR	Pomiary i lokalizacja	Właściwy sterownik, port i rzeczywiste dane; nie każdy sensor jest obowiązkowy.
Kamera i MediaMTX	Obraz dla operatora/modeli	Dostępny sensor, hostowa usługa i zgodna ścieżka RTSP/WebRTC.
Modem/router 5G	Transport IP	Zasilanie, obsługiwany interfejs, karta/usługa sieciowa i działający routing.
Hailo	Akceleracja zgodnego modelu	Runtime, właściwa architektura HEF, wejścia/wyjścia i rzeczywista inferencja.
Radio RC	Dodatkowy tor sterowania/telemetrii	Zgodne firmware i protokoły; osobna weryfikacja arbitrażu oraz failsafe.

Nazwa platformy nie oznacza, że w repozytorium znajduje się jeden uniwersalny sterownik wszystkich modemów 5G. Najpierw uruchom połączenie IP na hoście. Dopiero potem sprawdzaj zdalny transport RCSIM.

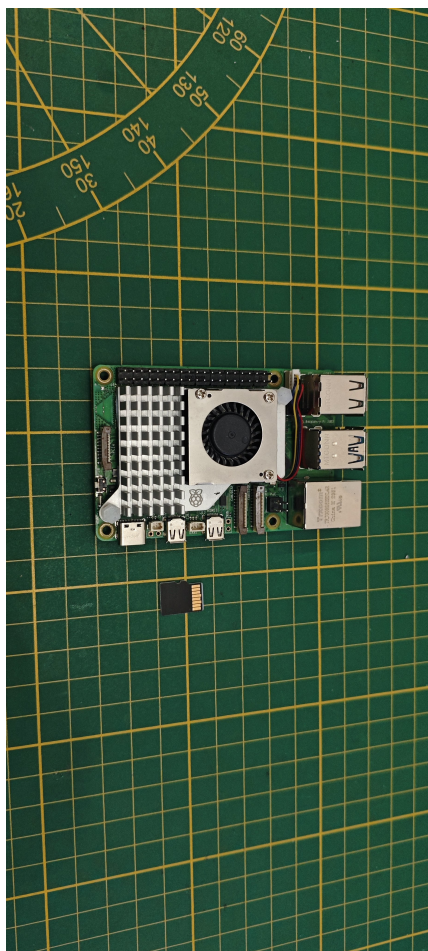
4. Montaż i konfiguracja wyjść

Pracę rozpocznij z odłączonym napędem lub na stanowisku uniemożliwiającym ruch pojazdu. Instaluj i przepinaj GPIO/I2C po wyłączeniu zasilania.



Stos sprzętowy: Raspberry Pi 5 z modulem akceleratora AI Hailo-8

Oddziel zasilanie logiki od szyny zasilania serw/ESC zgodnie z dokumentacją modułów. GPIO Raspberry Pi nie jest zasilaniem dla napędu. Złącze VCC układu logicznego i V+ serw w PCA9685 mają różne role.



Integracja komputera Raspberry Pi 5 na podwoziu modelu z okablowaniem i czujnikami

W przykładzie config.json ustawiono:

- `hardware.pca9685.calibration_channel: 15;`
- `hardware.pca9685.calibration_gpio: 17;`
- `hardware.pca9685.oe_pin: 27;`
- `hardware.pca9685.auto_calibrate: true;`
- limity skrętu i gazu w sekcji hardware.

To wartości pliku przykładowego. Sprawdź aktywnie wczytaną konfigurację i faktyczne okablowanie. Przy włączonej autokalibracji CH15 PCA9685 jest wyjściem pomiarowym, a GPIO17 wejściem pętli zwrotnej. Nie myl kanału CH15 z numerem nóżki układu scalonego.

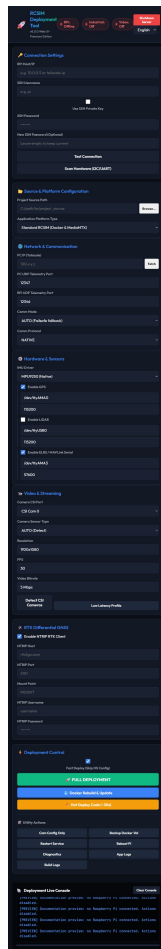
`modules/pca9685.py` mierzy szerokość impulsu przez odpytywanie GPIO i przelicza częstotliwość referencyjną. Nie jest to gwarancja precyzji systemu czasu rzeczywistego. Po kalibracji zmierz lub zweryfikuj rzeczywisty zakres wyjść.

OE: LOW zezwala na wyjścia, HIGH je blokuje. Wyłączenie PWM nie odłącza zasilania silnika ani serwa. ESC musi mieć zweryfikowaną reakcję na brak sygnału.

5. Wdrożenie przez Deployment Tool

GUI narzędzia znajduje się w

`RCSIM_deployment_tool/RCSIM_deployment_tool/RCSimRPi5deploymentapp.py`. To osobna aplikacja od GCS i MCS.



Deployment Tool: konfiguracja i wdrożenie kontenera pokładowego na Raspberry Pi

1. Przygotuj system ARM64 dla używanego Raspberry Pi oraz działające połączenie SSH.
2. Sprawdź tożsamość hosta, konto użytkownika i dostęp do urządzeń. Nie kopiuj demonstracyjnych haseł ani adresów z pomocniczych skryptów.
3. W Deployment Tool wybierz właściwy katalog projektu pokładowego RCSIM_RPI/RCSIM_rpi_embedded/rpi_project_source oraz docelowy host.
4. Zaznacz faktycznie zainstalowane urządzenia i sprawdź przypisane porty. Wygenerowany config.json porównaj z wyjściowym plikiem i własnym okablowaniem.
5. Wykonaj pełne wdrożenie przed próbą aktualizacji przyrostowej. Ścieżka hot-deploy zakłada istnienie kontenera rcsim_industrial.
6. Po wdrożeniu sprawdź hostowy MediaMTX, kontener, log aplikacji i aktywne wolumeny.
7. Skopiuj konfigurację i wersje oprogramowania do własnej kopii roboczej przed kolejną zmianą.

Dockerfile używa bazy ARM64 Debian Trixie, ustawia katalog /app i uruchamia entypoint.sh. Ten skrypt oczekuje hostowego MediaMTX na 127.0.0.1:8554, a następnie startuje supervisor. Brak usługi video może więc przerwać start kontenera, zanim sprawdzisz kanały sterujące.

Nie stosuj ogólnego polecenia docker run z przypadkowymi urządzeniami i wolumenami. Sprawdź konfigurację utworzoną przez konkretną wersję narzędzia wdrożenia.

6. Plik konfiguracji

Głównym plikiem aplikacji pokładowej jest `config.json` w katalogu projektu/aplikacji. Nie zakładaj istnienia `config/rc_config.json`. Konfiguracja GCS na PC i konfiguracja MCS nie są tym samym plikiem.

Sekcja/pole	Znaczenie
<code>pc_ip</code>	Docelowy adres GCS dla skonfigurowanego UDP; przy pustym polu usługa czeka na inicjalizację połączenia.
<code>comm_mode</code> , <code>comm_protocol</code>	Wybór/stan strategii komunikacji w aplikacji pokładowej.
<code>hardware.gps</code> , <code>hardware.lidar</code>	Włączenie, port i baudrate urządzenia.
<code>hardware.pca9685</code>	Kalibracja i OE kontrolera PWM.
<code>camera</code> , <code>video</code>	Źródło oraz parametry obrazu; wymagają zgodności z usługą kamery.
<code>ai</code>	Ścieżka i kontrakt modelu AI.
<code>autonomous_navigation</code>	Parametry nawigacji, siatki i wybranych zabezpieczeń.
<code>ntrip</code>	Opcjonalne połączenie z usługą poprawek RTK.

Nie włączaj wszystkich sensorów dlatego, że są w pliku przykładowym. Ustaw stan zgodny z wyposażeniem i potwierdź dane po starcie. Dane logowania NTRIP i sieciowe przechowuj poza publikowanym podręcznikiem i zrzutami ekranu.

7. Sieć IP/5G i porty aplikacji

Port/usługa	Rola potwierdzona w źródłach
UDP 12346	Przyjmowanie komend sterujących przez aplikację RPi.
UDP 12347	Skonfigurowany port docelowy telemetry do GCS.
HTTP 8080, /offer	Sygnalizacja pokładowej strategii WebRTC.
UDP 12348	Zarządzanie/healthcheck supervisor.
RTSP 8554	Usługa MediaMTX i źródło obrazu.
HTTP 8889	Endpoint WebRTC/WHEP MediaMTX z przykładowej konfiguracji.

Sygnalizacja HTTP i strumień mediów WebRTC nie są jednym stałym portem transmisji. Dostępność endpointu nie dowodzi zestawienia mediów ani DataChannel.

Procedura dla 5G: sprawdź adresy i routing hosta, osiągalność GCS z RPi i usług RPi ze stacji, a dopiero potem tryb UDP/WebRTC. NAT operatora, zaporę i reguły używanej sieci VPN mogą zmienić osiągalność. Nie ma w kodzie gwarancji stałej latencji ani publicznego adresu IP operatora.

Przy łączeniu się przez VPN (np. Tailscale), usługa MediaMTX bez wpisanego publicznego serwera STUN w `webrtcICEServers` (`stun:stun.l.google.com:19302`) w `mediamtx.yml` wysyła do klienta niekompatybilne adresy lokalne/Docker. W efekcie DTLS przechodzi pomyślnie, ale ruch wideo UDP jest gubiony. Wpisanie publicznego serwera STUN w konfiguracji MediaMTX jest wymagane dla stabilnego FPV przez VPN.

Nie udostępniaj portów sterowania, konfiguracji i supervisorowi otwartemu internetowi. Użyj kontrolowanej sieci i reguł dostępu właściwych dla wdrożenia.

8. Pierwsze uruchomienie bez autonomii

1. Sprawdź stabilne zasilanie i fizyczne wyłączenie napędu.
2. Potwierdź działanie supervisor'a oraz procesu `main_service.py`.
3. Odczytaj logi wykrywania urządzeń. Porównaj je z `config.json`.
4. Połącz GCS wybranym transportem, pozostawiając sterowanie w trybie ręcznym i model rozbrojony.
5. Sprawdź wiek i jednostki telemetry: ruch rzeczywistego sensora powinien zmieniać odpowiedni odczyt.
6. Zweryfikuj osobno kamerę, kanały kierunku/gazu i uzbrojenie.
7. Przetestuj zanik IP, dodatkowego RF, zatrzymanie aplikacji i reakcję ESC. Powrót połączenia nie może wywołać nieoczekiwanego ruchu.
8. Dopiero po tych testach włączaj kolejne funkcje nawigacji lub AI.

Odpowiedź PONG supervisor'a potwierdza jego odpowiedź na zapytanie, nie sprawność PWM, aktualność sensorów ani gotowość autonomii.

9. Nawigacja, SLAM i AI jako rozszerzenia

Pokładowy `SlamManager` korzysta z BreezySLAM/RMHC w osobnym wątku. Kod definiuje mapę 800 × 800 pikseli o boku 40 m i skany 360 próbek. Parametry te nie są potwierdzeniem centymetrowej dokładności lokalizacji. Przy braku biblioteki kod przechodzi na ścieżkę bez rzeczywistego silnika.

Pakiety mapy SLAM oraz obszernej telemetry przesyłane z RPi do PC podlegają limitowi MTU (1100 bajtów). Moduł `chunking.py` dzieli większe ramki mapy na fragmenty ze znacznikiem `__c: 1`, co zapobiega fragmentacji IP i utracie połączenia.

Silnik ICP/GraphSLAM w GCS jest odrębną implementacją. Parametr i plik mapy jednego silnika nie muszą być kontraktem drugiego. Sprawdź, który proces aktualnie wyznacza pozycję i ścieżkę.

AIManager rozpoznaje m.in. `model_path/hef_path` oraz runtime Hailo. Brak runtime lub pliku modelu może prowadzić do Mock, a nie automatycznej, równoważnej inferencji na CPU. Modele kompilowane pod NPU Hailo-8 wymagają statycznej rozdzielczości klatki wejściowej; przed inferencją obraz jest skalowany (`cv2.resize`) i uzupełniany o wymiar batch (`np.expand_dims`). Flaga inicjalizacji nie wystarcza do dopuszczenia jazdy AI.

Wdrożenie modelu wymaga zgodności wariantu Hailo-8/Hailo-8L, nazw i kształtów wejść, normalizacji, wyjść i interpretacji wyników. Alforge ma wybór architektury `hailo8/hailo8l`; sam eksport ONNX nie daje pliku HEF zgodnego z każdym modelem.

Przed aktywacją AI potwierdź rzeczywiste wejście z kamery/sensorów, zmieniające się wyniki inferencji oraz ręczne przejęcie sterowania. Testuj kolejno funkcje i rejestruj ich warunki; nie łącz pierwszego startu AI z pierwszym testem 5G i nowej instalacji PWM.

10. Zabezpieczenia i ograniczenia

PCA9685 ma w implementacji domyślny timeout 0,5 s liczony od ostatniego wywołania `set_servo_pulse`, a nie bezpośrednio od pakietu sieciowego. Każdy zapis wyjścia odświeża licznik; przed pierwszym zapisem kontrola jest pomijana. Faktyczna reakcja zależy od ustawień i wywoływania kontroli. Jest to inny mechanizm niż supervisor, watchdog systemowy, stan linku i logika nawigacji.

Brak dostępnego I2C może pozostawić sterownik PCA9685 w ścieżce symulacyjnej bez fizycznego wyjścia (driver_type pozostaje pusty). Działające GUI, telemetria i przyjęte komendy nie dowodzą generowania PWM na pinach. Potwierdź rzeczywisty sterownik i działanie wyjścia.

SafetySupervisor ocenia m.in. dane sensorów, wydajność AI oraz stan RF/IP. Dla utraty obu linków kod rozróżnia STOP i RTH zależnie od gotowości AI. Reakcje napięciowe też zależą od konfiguracji. Nie zakładaj jednej akcji niezależnie od stanu systemu.

Geofence narysowany w GCS nie jest sam w sobie poleceniem odłączenia napędu. Politykę zatrzymania, RTH, przejęcia i reakcji ESC trzeba sprawdzić na całym torze.

Pro Shield pozostaje koncepcją w dokumentacji projektu; w sprawdzonym drzewie nie znaleziono kompletnej paczki produkcyjnej hardware/pcb/. Nie traktuj opisu koncepcji jako zweryfikowanego schematu zasilania gotowego HAT.

11. Diagnostyka po wdrożeniu

Polecenia wykonuj na hoście RPi, chyba że wskazano kontener:

```
sudo docker ps --filter name=rctsim_industrial
sudo docker logs --tail 100 rctsim_industrial
sudo journalctl -u mediactx -n 50 --no-pager
sudo docker inspect --format '{{.json .Mounts}}' rctsim_industrial
sudo docker exec rctsim_industrial python3 tools/diagnostics/healthcheck.py
```

Nazwy kontenera/usługi odnoszą się do sprawdzanych skryptów. Jeśli wdrożenie je zmieniło, najpierw ustal własne nazwy.

Objaw	Pierwsza kontrola
Kontener nie startuje, brak telemetrii	Log entrypoint i osiągalność hostowego MediaMTX na 8554.
Supervisor odpowiada, pojazd nie reaguje	Proces aplikacji, uzbrojenie, konfiguracja wyjść, OE i logi błędów.
Lokalnie działa, przez 5G nie	Routing, adresy, NAT/VPN/zapora; osobno sterowanie, telemetria i media.
Model jest w Mock	Runtime Hailo, rzeczywista ścieżka modelu, architektura oraz log inicjalizacji.
SLAM nie aktualizuje pozycji	Dostępność BreezySLAM i rzeczywiste skany; nie tylko widoczna mapa w GUI.
I2C wykazuje błąd	Zasilanie, adres, właściwy bus, sterownik i okablowanie przy wyłączonym zasilaniu.

12. Źródła researchu

Ścieżki względem katalogu projektu; odnośniki do linii dotyczą stanu opracowania.

- RCSIM_RPI/RCSIM_rpi_embedded/rpi_project_source/Dockerfile:1 — baza ARM64 i zależności kontenera.
- RCSIM_RPI/RCSIM_rpi_embedded/rpi_project_source/entrypoint.sh:8 — oczekiwanie na MediaMTX i start supervisor.
- RCSIM_RPI/RCSIM_rpi_embedded/rpi_project_source/core/supervisor.py:42 — port, konfiguracja i nadzorowanie aplikacji.
- RCSIM_RPI/RCSIM_rpi_embedded/rpi_project_source/core/main_service.py:707 — uruchomienie WebRTC i UDP.

- `RCSIM_RPI/RCSIM_rpi_embedded/rpi_project_source/core/webrtc_manager.py:172` — endpoint /offer.
- `RCSIM_RPI/RCSIM_rpi_embedded/rpi_project_source/core/telemetry_builder.py:32` — rzeczywisty pakiet telemetryczny.
- `RCSIM_RPI/RCSIM_rpi_embedded/rpi_project_source/config.json:1` — przykładowe ustawienia, nie automatyczna autodetekcja wyposażenia.
- `RCSIM_RPI/RCSIM_rpi_embedded/rpi_project_source/modules/pca9685.py:61` — parametry kontrolera; dalsze metody kalibracji i failsafe.
- `RCSIM_RPI/RCSIM_rpi_embedded/rpi_project_source/modules/managers/slam_manager.py:19` — BreezySLAM/RMHC i stałe mapy.
- `RCSIM_RPI/RCSIM_rpi_embedded/rpi_project_source/modules/ai_manager.py:69` — wybór modelu i ścieżka Mock.
- `RCSIM_RPI/RCSIM_rpi_embedded/rpi_project_source/core/safety_supervisor.py:123` — ocena wydajności AI i utraty linków.
- `RCSIM_RPI/RCSIM_rpi_embedded/rpi_project_source/tools/diagnostics/healthcheck.py:1` — PING/PONG supervisor.
- `RCSIM_deployment_tool/RCSIM_deployment_tool/core/deployment_logic.py:887` — archiwizacja i generowanie konfiguracji; :1426 — warunki hot-deploy.
- `RCSIM_AIforge/RCSIM_AIforge/src/hailo_converter.py:237` — wybór architektury Hailo.

Procedury obsługi aplikacji PC i MCS znajdują się w osobnych dokumentach. Zasilanie konkretnego HAT, modemu i ESC zawsze dobierz także do dokumentacji ich producentów.