

RCSIM — Tier 2: podręcznik użytkownika ESP32-CAM

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

Spis treści

1. Przeznaczenie	2
2. Warianty obecne w repozytorium	2
3. Płytki i ograniczenia pinów	2
3.1. Połączenia podstawowego zestawu WROVER	2
4. Wymagania i konfiguracja podstawowa	4
4.1. Wi-Fi i adres — zmień przed kompilacją	5
4.2. Przygotowanie Arduino IDE	5
4.3. Wgrywanie i bezpieczny start	5
5. Protokół bazowy	6
6. Pierwsze uruchomienie na stanowisku	6
6.1. Connection Settings — dokładna konfiguracja GCS	6
6.2. Kierownica, pedały i kanały	7
6.3. Obraz MJPEG i parametry kamery	8
7. Failsafe i uzbrojenie	8
8. Rozszerzenia V2, V3 i V4	9
9. Diagnostyka i ograniczenia	9
Odzyskiwanie po nieudanym wgraniu lub restarcie	10
10. Źródła i ślady weryfikacji	10

Stan opracowania: 14 października 2026 r. Fakty pochodzą z kodu projektu. Opis nie potwierdza fizycznych testów każdej kombinacji płytki i peryferiów. W ramach przygotowania dokumentacji nie wgrywano firmware ani nie uruchamiano pojazdu.

1. Przeznaczenie

Tier 2 to platforma pojazdu oparta na ESP32, obejmująca warianty z kamerą. W podstawowym firmware ESP32 odbiera kanały przez Wi-Fi/UDP, steruje PCA9685 i udostępnia obraz MJPEG oraz telemetrię IMU. GCS działa na komputerze; ESP32 wykonuje zadania pokładowe.

Podstawowy tor sterowania: kontroler → GCS → UDP CSV → ESP32 → PCA9685 → serwa/ESC. Tor obrazu: kamera → ESP32 → HTTP MJPEG → klient obrazu GCS. Tor telemetrii: IMU → ESP32 → UDP JSON → GCS.

Rozszerzenie firmware o CRSF nie zmienia tej platformy w Tier 3. Tier 3 obejmuje odrębnie system radiowy ELRS/mLRS, CRSF/MAVLink i mody aparatury. Raspberry Pi z robotyką i autonomią należy do Tier 4.

2. Warianty obecne w repozytorium

Katalog bazowy firmware: RCSIM_PC/pc_app/ESP32Arduino/RCSIM_Arduino_ESP32/ESP32 Tier 2/ESP32/.

- ESP32/ESP32.ino: podstawowy Wi-Fi/UDP, kamera, IMU i PCA9685.
- ESP32V2/ESP32V2autocalibPCA/ESP32V2autocalibPCA.ino: dodatkowa kalibracja PCA9685.
- ESP32V3+GPS+FPV/ESP32 V3 GPS + FPV: wariant z GPS, OSD i wyborem IMU.
- ESP32V4_CRSF_MultiLink/ESP32V4_CRSF_MultiLink.ino: hub CRSF z wyborem transportu i multiplexerem dwóch łącz.
- ESP32_CRSF_Dongle_Transmitter/ESP32_CRSF_Dongle_Transmitter.ino: odrębny nadajnik pośredniczący, nie zamiennik firmware kamery.

Wersje nie są jednym programem z automatycznym wykrywaniem wszystkich funkcji. Przed kompilacją zapisz wybrany wariant i sprawdź jego definicje pinów. Poniższa procedura podstawowa dotyczy ESP32/ESP32.ino.

3. Płytki i ograniczenia pinów

Podstawowy szkic zawiera gałęzie BOARD_AI_THINKER, BOARD_WROVER_DEV i LILYGO_TSIMCAM_S3. W badanym pliku aktywna jest definicja BOARD_WROVER_DEV. Dokładna nazwa makra LilyGo różni się między wariantami firmware. Nie należy kopiować jej bez porównania warunku preprocesora.

Dla podstawowej gałęzi Wrover kod wybiera I2C SDA GPIO13 i SCL GPIO14. PCA9685 jest inicjalizowany przez tę magistralę; skaner oznacza adres 0x40. IMU MPU6050 jest oznaczane w skanerze adresem 0x68. Adres skanera nie zastępuje sprawdzenia rzeczywistego adresu modułu.

Gałąź AI Thinker wymaga rozwiązania konfliktu pinów przed użyciem kamery i I2C. Ten sam plik przypisuje GPIO21 do linii kamery Y5 i do SDA, a GPIO22 do PCLK i do SCL. Nie należy traktować tego wariantu jako gotowego schematu równoczesnej pracy. Wariant LilyGo ma osobną mapę pinów; jego obecność w kodzie nie potwierdza zgodności każdego modelu sprzedawanego pod tą nazwą.

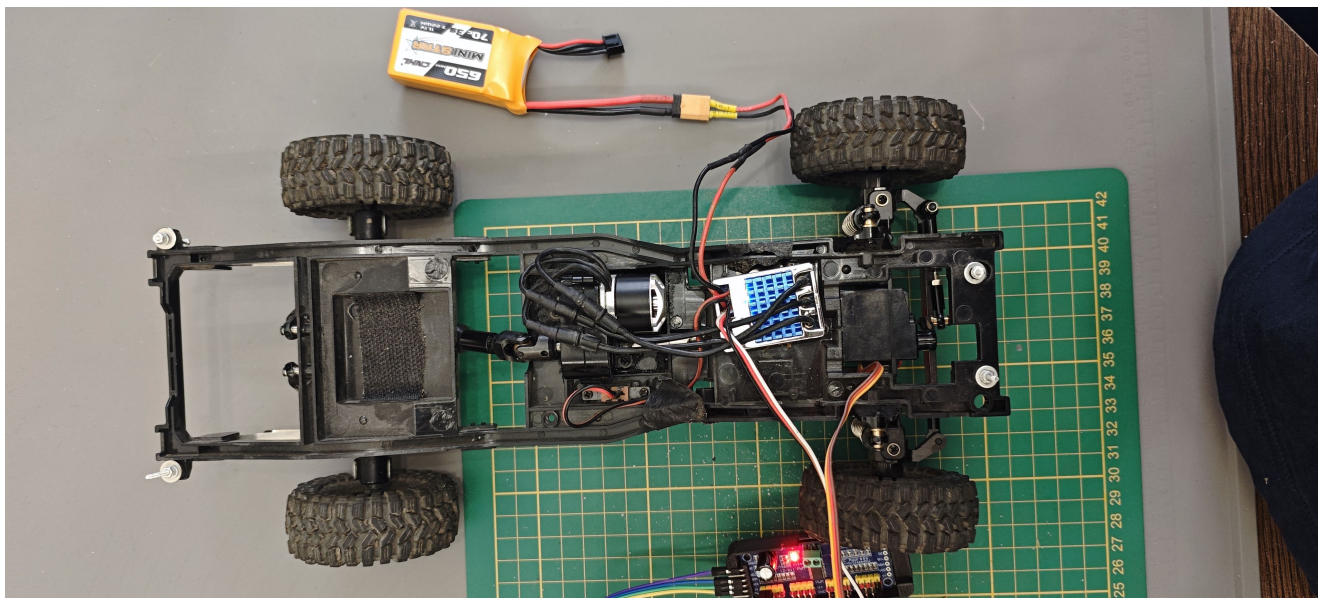
3.1. Połączenia podstawowego zestawu WROVER

Poniższy schemat dotyczy aktywnej gałęzi BOARD_WROVER_DEV, nie AI Thinker ani V4:

ESP32 GPIO13 (SDA) — SDA PCA9685 — SDA MPU6050

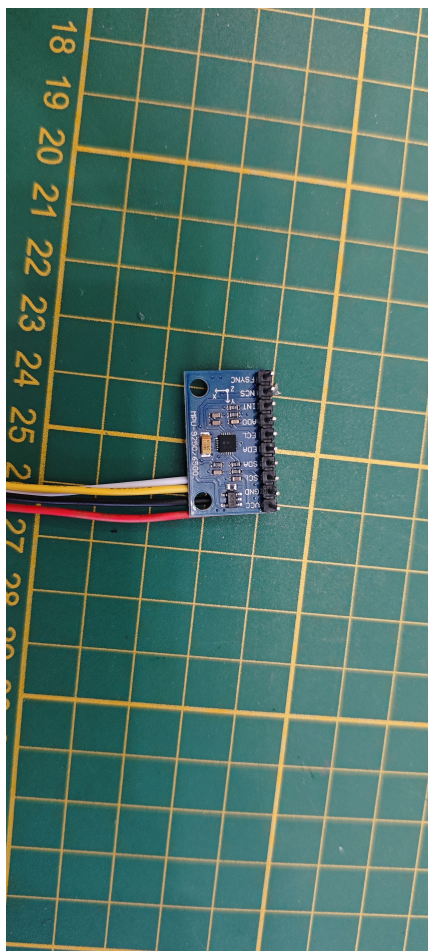
ESP32 GPIO14 (SCL) — SCL PCA9685 — SCL MPU6050
Wspólna masa — GND modułów oraz toru sygnałowego
PCA9685 CH0 — sygnał odbiorczego kanału CH1, np. skręt
PCA9685 CH1 — sygnał kanału CH2, np. ESC
Oddzielne zasilanie wykonawcze — V+ serw/ESC według wymagań modelu

Numery kanałów PCA są liczone od zera: CH0 odpowiada pierwszej liczbie w datagramie. Przykładowe przypisanie skrętu i gazu nie jest wymuszone przez bazowy firmware — dobierz je do mapowania GCS. Nie utożsamiaj portu sygnałowego ESC z jego przewodami mocy.



Moduł ESP32-CAM z kamerą OV2640 przygotowany do montażu

VCC logiki PCA i V+ szyny serw mają inne role. Dopasuj napięcia zasilania, poziomy I2C i podciąganie magistrali do ESP32 i posiadanych modułów. Nie prowadź napięcia pakietu bezpośrednio do GPIO ani nie zasilaj serw z pinu logiki. Dokładny regulator/BEC i rozdział mocy zależą od sprzętu; repozytorium nie dostarcza kompletnego, zatwierzonego schematu dla wszystkich zestawów.



Płytki ESP32 oraz kontroler PCA9685 zamontowane na podwoziu modelu

Przy wyłączonym zasilaniu sprawdź masę, polaryzację, taśmę kamery i orientację przewodów sygnałowych. Kamera korzysta z wielu GPIO, więc piny zajęte przez jej magistralę nie są wolnymi wyprowadzeniami na dodatkowe I2C/UART. Na pierwsze uruchomienie pozostaw odłączony napęd.

4. Wymagania i konfiguracja podstawowa

Potrzebne są odpowiednia płytka z kamerą, programator/USB właściwy dla płytki, PCA9685, zgodne poziomy elektryczne i prawidłowo zaprojektowane zasilanie. Zasilanie serw i ESC należy dobrać do obciążenia konkretnego modelu. Ten kod nie określa kompletnego schematu rozdziału mocy. Na stanowisku odłącz napęd lub uniemożliw ruch pojazdu.

W szkicu ustaw dane swojej sieci w `ssid` i `password`. Nie publikuj rzeczywistego hasła w repozytorium ani w zrzutach diagnostycznych. Sprawdź, że aktywna jest dokładnie właściwa gałąź płytki. Kod bazowy ma włączone `ENABLE_CAMERA` i `ENABLE_IMU`. Brak któregoś urządzenia wymaga dostosowania konfiguracji do rzeczywistego zestawu.

Ustawienia sieciowe bazowego szkicu:

- port sterowania ESP32: UDP 12345;
- docelowy port telemetry komputer: UDP 12347;
- serwer obrazu: TCP/HTTP 81;
- adres strumienia deklarowany przez firmware: `http://IP_ESP32:81/stream`;
- monitor szeregowy: 115200 bps.

GCS używa strategii ESP32 lub aliasu ESP32_UDP. Ustaw `esp_ip` na rzeczywisty adres ESP32, `esp_port` na port sterowania, a port nasłuchu telemetryi uzgodnij z firmware. Adres sieciowy nie jest stałym adresem każdego urządzenia.

4.1. Wi-Fi i adres — zmień przed kompilacją

Bazowy szkic ma `useStaticIP = true` i przykładowy adres **192.168.31.111**, bramę **192.168.31.1** oraz maskę **255.255.255.0**. To nie jest wykryty adres Twojej sieci. Ustaw niekolidujący adres i bramę zgodne z routerem albo wybierz `useStaticIP = false`, aby pobrać adres przez DHCP. W obu przypadkach odczytaj faktyczny adres w monitorze szeregowym.

Firmware łączy się z wpisaną siecią jako klient. W bazowej wersji nie ma formularza WWW do wpisywania SSID ani automatycznego AP konfiguracyjnego. Nazwa hotspotu telefonu w przykładzie innego wariantu nie instaluje tunelu VPN i nie daje automatycznie dostępu z Internetu.

4.2. Przygotowanie Arduino IDE

1. Zainstaluj Arduino IDE i pakiet **esp32 by Espressif Systems** przez Boards Manager. Korzystaj z pakietu właściwego dla ESP32, nie z samego pakietu płytek Arduino AVR.
2. Otwórz bazowy ESP32/ESP32.ino. Dla V2 otwórz jego własny szkic; dla V4 zachowaj wszystkie pliki .h w katalogu szkicu. V3 wymaga przygotowania lokalnego folderu szkicu i nazwy pliku .ino, ponieważ plik źródłowy nie ma rozszerzenia.
3. W Library Manager zainstaluj **Adafruit PWM Servo Driver Library**, jej zależności, w tym **Adafruit BusIO**, oraz **MPU6050_light** dla bazowej IMU. `WiFi`, `WiFiUDP`, `Wire` i `esp_camera` pochodzą z pakietu ESP32.
4. Wybierz rzeczywistą płytkę w **Tools** → **Board**, np. **ESP32 Wrover Module** dla odpowiadającego modułu lub **AI Thinker ESP32-CAM** dla odpowiadającej płytki. Niezależnie ustaw zgodną gałąź makr w szkicu. Sam wybór płytki w IDE nie zmienia `BOARD_WROVER_DEV` w źródle.
5. Dopasuj Flash Size, PSRAM i Partition Scheme do płytki. README podaje przykładowo **Huge APP (3MB No OTA/1MB SPIFFS)**, **DIO, 40 MHz**; nie są to uniwersalne ustawienia dla ESP32-S3. Wybór partycji „No OTA” nie zapewnia aktualizacji bezprzewodowej.
6. Wybierz właściwy **Port**, początkowo **Upload Speed 115200**, i uruchom **Verify**. Zapisz wersję pakietu ESP32 i bibliotek, z którymi szkic faktycznie się kompiluje. Repozytorium nie przypina jednego zweryfikowanego zestawu wersji dla wszystkich wariantów.
7. Jeżeli brakuje nagłówka biblioteki, zainstaluj odpowiadającą bibliotekę. Przy błędzie API kamery, `watchdog` lub ESP-NOW sprawdź zgodność wersji pakietu i wariantu; nie traktuj kompilacji innego szkicu jako potwierdzenia zgodności.

Nie wymagaj bibliotek `AsyncTCP`/`ESPAsyncWebServer` tylko dlatego, że starsze README wymienia je jako opcjonalne: bazowy szkic korzysta z `WiFiServer`. V3 wymaga dodatkowo **TinyGPSPlus** oraz biblioteki wybranej IMU: `MPU6050_light`, `MPU9250_WE`, `Adafruit BNO08x` albo `DFRobot_BMX160`. Nie instaluj kilku czujników w sprzęcie tylko po to, by odpowiadały wszystkim opcjom kompilacji.

4.3. Wgrywanie i bezpieczny start

1. Zabezpiecz napęd i odłącz urządzenia mogące zakłócać piny programowania. Podłącz płytkę przez jej USB albo zgodny programator UART. Poziomy logiczne programatora muszą odpowiadać płytce.
2. Dla płytki z automatycznym bootloaderem wybierz port i **Upload**. Dla klasycznego AI Thinker bez USB repozytorium opisuje zwarcie **GPIO0 do GND**, reset i dopiero rozpoczęcie Upload; po sukcesie zdejmij zworę i ponownie zresetuj. Nie przenoś tej procedury bez sprawdzenia na każdy model S3.
3. Zamknij inne monitory zajmujące port. Przy problemach z transmisją wróć do 115200 i sprawdź kabel danych oraz zasilanie zamiast wielokrotnie kasować Flash.

4. Po komunikacji zakończenia otwórz monitor **115200 bps**, zresetuj i odczytaj Wi-Fi, I2C, IMU oraz kamerę.
5. Podczas wyznaczania offsetów IMU pozostaw moduł nieruchomy. Brak komunikatu błędu nie zastępuje sprawdzenia odczytów.

Nie wskazano jednej gotowej paczki BIN właściwej dla wszystkich płytek. Nie wgrywaj pliku o przypadkowej nazwie ani obrazu MCS/Raspberry Pi. Zachowaj sprawdzone źródła i ustawienia kompilacji do ponownego odtworzenia firmware.

5. Protokół bazowy

ESP32UDPStrategy wysyła kanały jako ASCII CSV zakończone nową linią. Przykład zawartości datagramu dla ośmiu kanałów:

```
1500,1500,1500,1500,1500,1500,1500,1500
```

Firmware bazowy przetwarza maksymalnie 16 kanałów. Ogranicza odebrane wartości do zakresu 1000–2000 μ s i mapuje je na wartości rejestrów PWM 205–410 przy zadanych 50 Hz. To implementacja przeliczenia, nie wynik kalibracji konkretnego oscylatora PCA9685. Nieprzesłane kanały nie są w tej pętli automatycznie uzupełniane nową wartością. Nie należy zakładać walidacji kompletności wszystkich 16 kanałów.

Telemetria IMU jest składana jako JSON z obiektem imu i polami ax, ay, az, gx, gy, gz. Pętla próbuje wysłać ją w odstępach większych niż 50 ms. Adres odbiorcy pobiera z ostatniego pakietu sterowania. Przed pierwszym pakietem sterowania brak telemetrii nie przesądza o awarii IMU.

6. Pierwsze uruchomienie na stanowisku

1. Wybierz szkic, płytkę i zgodne biblioteki wynikające z jego `include`.
2. Sprawdź mapę kamery i I2C; rozwiąż konflikty przed podłączeniem peryferiów.
3. Ustaw sieć, wgraj program i otwórz monitor szeregowy 115200 bps.
4. Odczytaj stan połączenia Wi-Fi, adres IP i wynik skanowania I2C.
5. Potwierdź wykrycie PCA9685 i używanej IMU.
6. Sprawdź strumień obrazu pod adresem ESP32 na porcie 81.
7. W GCS ustaw transport ESP32 oraz zgodne porty.
8. Wyślij kanały, sprawdź powrót JSON i ruch pojedynczego serwa testowego.
9. Zweryfikuj kierunki i zakresy bez czynnego napędu.
10. Przerwij sterowanie na ponad 500 ms i sprawdź wartości wszystkich wyjść.
11. Dopiero po sprawdzeniu zachowania ESC przejdź do prób ruchowych.

Brak obrazu, brak sterowania i brak telemetrii diagnozuj osobno. Są to różne przepływy danych i mogą ulegać niezależnym awariom.

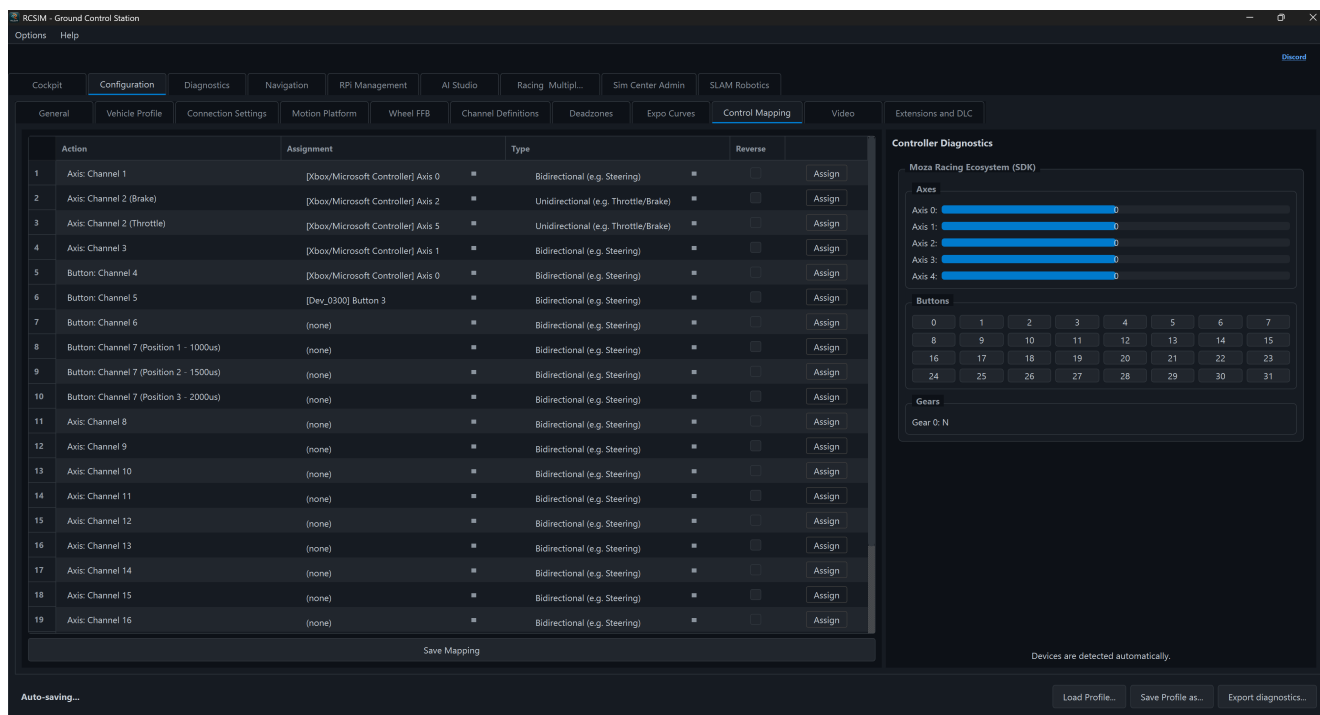
6.1. Connection Settings — dokładna konfiguracja GCS

1. W **Configuration** → **Connection Settings** wybierz w górnym **GCS Mode** pozycję **ESP32 (WiFi Lite Mode)**. Samo otwarcie karty **Tier 2 (WiFi ESP32)** nie wyłącza Mock i nie zmienia transportu.
2. Dla procedury bazowej wybierz **Tier 2 Classic (Wi-Fi UDP / CSV Telemetry)** oraz **Wi-Fi UDP**. Lista zawiera też Pro V4, MicroLink VPN, ESP-NOW i USB, ale obecna strategia ESP32 w GCS nadal wysyła CSV i odbiera JSON. Te wybory nie tworzą automatycznie zgodnego toru CRSF/ESP-NOW/USB.

3. W **ESP32 / VPN IP Address** wpisz adres odczytany po uruchomieniu ESP32. W **UDP Control Port** wpisz **12345**, chyba że świadomie zmieniono go w firmware.
4. W **FPV Camera Port** ustaw **81**. Sprawdź podgląd **MJPEG Stream Address**: http://IP_ESP32:81/stream. To adres obrazu, nie port sterowania.
5. W **General** → **Advanced** → **GCS Listen Port** ustaw **12347**, zgodnie z firmware. Zezwól na odbiór tego UDP w odpowiedniej sieci komputera.
6. Zakończ edycję pól, poczekaj na automatyczny zapis i zachowaj profil przez **Save Profile as....** Wróć do **Cockpit**, wybierz **Manual (Driver)** i **Connect**.
7. Najpierw sprawdź pakiety telemetry i sygnał na jednym kanale testowym. Otwarte gniazdo UDP nie potwierdza obecności ESP32 ani sprawnego sterowania.

Gdy kamera działa, a nie ma telemetry: wyślij najpierw sterowanie, ponieważ firmware uczy się adresu komputera z nadawcy datagramu. Następnie sprawdź IMU i port nasłuchu. **Gdy jest telemetry, a brak reakcji:** sprawdź mapowanie, PCA, zasilanie wykonawcze i kanały; nie zmieniaj portu wideo.

6.2. Kierownica, pedały i kanały

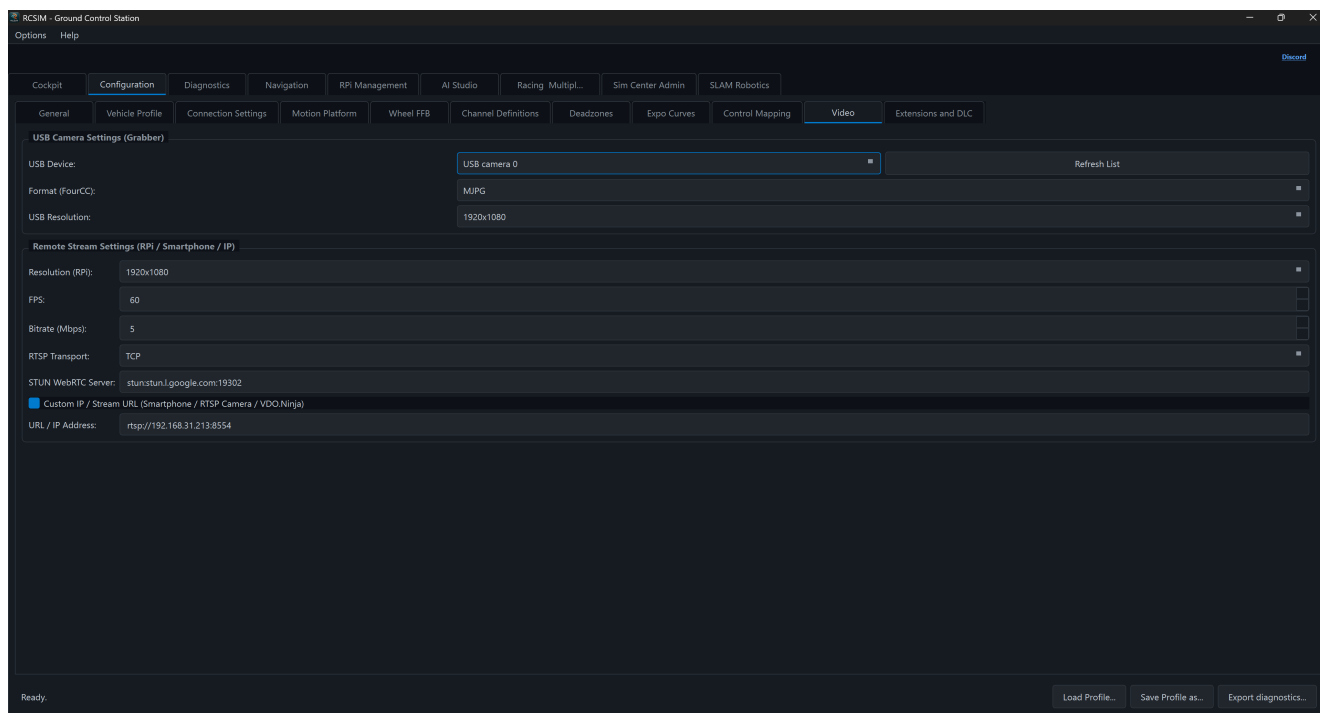


Przykładowy ekran Control Mapping GCS — przypisania nie są gotowym profilem Tier 2

W **Channel Definitions** przygotuj typ kanału skrętu i napędu. Dla skrętu użyj osi dwukierunkowej; dla dwóch pedałów wybierz odpowiednią funkcję gaz/hamulec zgodną z ESC. W **Control Mapping** wybierz **Assign**, porusz tylko właściwą osią i sprawdź **Assignment**, **Type** oraz **Reverse**. Zapisz przez **Save Mapping**.

Sprawdź odczyty w **Controller Diagnostics**, ustaw neutralny gaz i dopiero potem dopasuj martwą strefę oraz Expo. Nie kopiuj numerów osi z ilustracji: pokazuje przykładowy pad, nie Twoją kierownicę. Pierwsza liczba wychodzącego CSV steruje CH0 PCA, druga CH1 itd.; kanał AUX uzbrojenia z V4 nie jest automatycznie obecny w bazowym firmware.

6.3. Obraz MJPEG i parametry kamery



Przykładowe ustawienia Video GCS — adres RTSP ze zrzutu nie jest adresem ESP32-CAM

Podstawowy ESP32 udostępnia **HTTP MJPEG**, nie RTSP i nie WebRTC. Sprawdź obraz pod jego własnym adresem, a w kokpicie wybierz źródło zgodne z tym strumieniem. Jeżeli potrzebujesz pola własnego źródła w **Video**, użyj **Custom IP / Stream URL** i pełnego `http://IP_ESP32:81/stream`. Adres RTSP/8554 widoczny na ilustracji dotyczy innego przykładu.

Pola **Resolution (RPI)**, **FPS**, **Bitrate**, **RTSP Transport** oraz **STUN WebRTC Server** nie przeprogramowują tego szkicu. Przy PSRAM kod wybiera **VGA 640×480**, jakość JPEG 12 i dwa bufor; bez PSRAM **CIF 400×296**, jakość 15 i jeden bufor. Nie ma panelu zdalnej zmiany tych nastaw w bazowym firmware.

Serwer obsługuje aktywne połączenie strumienia w pętli; na pierwszą próbę użyj jednego klienta. Zamknij okno przeglądarki z podglądem przed oceną obrazu w GCS. Rozdzielczość i płynność zależą od sensora, pamięci i Wi-Fi; nie od samej liczby FPS wpisanej w GCS.

7. Failsafe i uzbrojenie

W bazowym firmware timeout wynosi 500 ms. Po jego przekroczeniu `triggerFailsafe()` ustawia wszystkie 16 wyjść na wartość odpowiadającą 1500 μ s. To neutralizacja PWM, a nie elektryczne odcięcie sygnału lub zasilania. Kod nie dowodzi, że każdy regulator zatrzyma silnik przy tej wartości.

Odbiór danych aktualizuje czas ostatniego pakietu. Parser bazowy nie wymaga pełnej ramki przed odświeżeniem tego czasu. Nie należy deklarować odporności na dowolnie błędny ruch sieciowy. W tej strategii GCS nie ma dedykowanej implementacji komend ARM/ESTOP porównywalnej z tekstowymi komendami mostka Arduino. Próba rozbrojenia musi zostać zweryfikowana dla rzeczywistego toru aplikacji.

8. Rozszerzenia V2, V3 i V4

V2 ma włączoną autokalibrację PCA9685 i definiuje sprzężenie z kanału 15 PCA9685 do GPIO12 ESP32. Funkcja kalibracji używa `pulseIn` i zmienia częstotliwość oscylatora w sterowniku. Nie uruchamiaj jej jako gotowej kalibracji bez wymaganego przewodu zwrotnego. Kanał przeznaczony na sprzężenie trzeba uwzględnić w okablowaniu wyjść. W badanym V2 kalibracja generuje około 1500 μ s, mierzy impuls i wykonuje maksymalnie 15 prób z kryterium błędu do 3 μ s. Jest to kryterium programu, nie certyfikat dokładności pomiaru. GPIO12 ma funkcje startowe układu ESP32; oceń jego użycie z konkretną płytką i poziomami sygnału. Nie podłączaj kanału pomiarowego do napędu.

V3 deklaruje GPS NMEA, Walksnail OSD, pomiar baterii i wybór MPU6050, MPU9250, BNO08X lub BMX160 przez makra kompilacji. Wybór w źródle nie oznacza równoczesnego podłączenia wszystkich czujników. Plik programu w repozytorium nie ma rozszerzenia `.ino`; przy przygotowaniu lokalnego szkicu zachowaj jego treść i wymagania Arduino IDE. Aktywny jest MPU6050 i WROVER. W pliku zdefiniowano mapę kamery tylko dla gałęzi WROVER; samo odkomentowanie nazwy innej płytki nie dostarcza kompletnego pinoutu. GPS odbiera NMEA na **GPIO32/9600 bps**; Walksnail ma **TX GPIO2, RX GPIO15/115200 bps**, a bateria wejście **GPIO33** przez prawidłowo dobrany dzielnik. Nie podłączaj pakietu bezpośrednio do ADC. GPS, OSD i pomiar baterii są domyślnie włączone, podobnie jak autokalibracja CH15–GPIO12. Wyłącz nieobecne elementy przed przygotowaniem własnego wariantu. Zwykły GPS NMEA w tym szkicu nie jest sam w sobie odbiornikiem RTK.

V4 jest osobnym hubem CRSF. Domyślnie wybiera `MODE_DUAL_LINK_HYBRID`. Definiuje RF UART RX15/TX16 przy 420000 bps oraz I2C GPIO8/9. Ta mapa dotyczy wariantu ESP32-S3, nie klasycznej płytki AI Thinker. Dodatkowo GPS używa RX18/TX17 przy 9600 bps, a ADC baterii GPIO1 i współczynnika dzielnika 5,545. Domyślnym sensorem w `SensorsManager.h` jest **LSM6DS3**; MPU6050 jest alternatywą kompilacji. PCA szuka adresów 0x40–0x47, także przez hub TCA9548A 0x70. Kalibracja zwrotna jest domyślnie wyłączona pinem –1. Timeout zadeklarowano jako 150 ms, CH5 jest kanałem uzbrojenia, a CH6 kanałem wyboru źródła RF/AUTO/sieć. Nie wolno przepisywać do V4 timeoutu i mapy pinów z bazowego szkicu kamery. CH1 oznacza skręt, CH2 gaz/hamulec. Wymuszenie RF lub sieci przez CH6 ogranicza wybór źródła; nie zakładaj automatycznego ratowania drugim linkiem w każdym ustawieniu przełącznika. W AUTO kod preferuje RF i wymaga stabilnych ramek przed powrotem. Sprawdź każdy zanik i powrót osobno na stanowisku.

V4 nie zawiera implementacji kamery/MJPEG obecnej w szkicu bazowym. Nie oczekuj obrazu pod portem 81 po zastąpieniu nim ESP32-CAM. W aktualnej inicjalizacji PCA wywołanie neutralizacji poprzedza ustawienie flagi gotowości sterownika; nie potwierdza więc ustawienia neutralnych wyjść natychmiast po starcie. Przygotuj niezależnie bezpieczny start i sprawdź fizyczne sygnały przed dopuszczeniem napędu.

Zgodność formatu musi być sprawdzona przed połączeniem V4 z GCS. Bazowa ESP32UDPStrategy wysyła tekst CSV. V4 przekazuje dane do parsera ramek binarnych CRSF. Te transporty nie są wymienne tylko dlatego, że używają tych samych numerów portów. Konfiguracja VPN/telefonu również wymaga faktycznego pośrednika sieciowego; nazwa trybu nie potwierdza samodzielnego działania usługi VPN na mikrokontrolerze. Warstwa MicroLink bez włączonej integracji biblioteki zgłasza tryb emulacji/zewnętrznego klienta. W trybie domyślnym Hybrid używany jest zwykły UDP przez Wi-Fi. Do prób internetowych potrzebna jest rzeczywista trasa sieciowa i zgodny nadawca CRSF; przełącznik firmware w GCS tego nie buduje. Odrębny szkic Dongle służy pośredniczeniu z innym torem, nie dodaje brakującej kamery do V4.

9. Diagnostyka i ograniczenia

- Brak sterowania: porównaj IP, port, format CSV/CRSF i aktywny wariant firmware.
- Brak telemetrii: sprawdź pakiet sterujący, IMU, port 12347 i zaporę komputera.
- Brak kamery: sprawdź mapę pinów, zasilanie i wynik inicjalizacji w logu.
- Błędne pozycje serw: sprawdź przeliczenie PWM, kalibrację i mechaniczne ograniczenia.

- Nieudana kalibracja V2: sprawdź połączenie CH15–GPIO12.
- AI Thinker z kamerą i I2C: najpierw rozwiąż opisany konflikt GPIO21/22.

GCS emituje dla podstawowego ESP32 statystyki `ping_ms = 5.0` i `loss_percent = 0.0` po odebraniu JSON. To wartości wpisane w kod, nie pomiar opóźnienia i strat łącza. Nie należy na ich podstawie oceniać jakości Wi-Fi lub dostępu przez Internet. Strategia bazowa zgłasza brak obsługi binarnych poleceń AI.

Odzyskiwanie po nieudanym wgraniu lub restarcie

1. Zatrzymaj próby ruchowe i odłącz napęd. Zanotuj wybrany szkic, płytkę, pakiet ESP32 i komunikat błędu.
2. **Brak portu:** sprawdź kabel danych/programator i sterownik urządzenia, a następnie listę portów systemu. Numer COM może zmienić się po zmianie gniazda.
3. **Upload zatrzymuje się na łączeniu:** sprawdź właściwy port, zwolnij go z innych aplikacji i zastosuj procedurę BOOT/RESET właściwą dla płytki. Dla klasycznego AI Thinker zdejmij zworę GPIO0–GND przed normalnym startem.
4. **Pętla resetów lub błąd kamery:** sprawdź zasilanie, taśmę, mapę GPIO i konfigurację PSRAM. Nie zamieniaj dowolnie pinów ani płytki IDE tylko po to, by zniknął błąd.
5. **Program działa, Wi-Fi nie:** sprawdź SSID/hasło, statyczną podsieć lub DHCP i rzeczywisty adres w logu. Stary adres zapisany w GCS nie jest dowodem adresu po ponownym wgraniu.
6. Przywróć sprawdzony wariant źródeł i ponownie wykonaj Verify/Upload. Kasowanie całej pamięci traktuj jako osobną operację wymagającą zachowania potrzebnych ustawień, nie uniwersalną pierwszą naprawę.
7. Po odzyskaniu programu powtórz pełną próbę neutralnego gazu i failsafe. Kamera działająca po flashowaniu nie potwierdza poprawnych wyjść PWM.

10. Źródła i ślady weryfikacji

Ścieżki są względne wobec projektu. Długie ścieżki firmware używają prefiksu `RCSIM_PC/pc_app/ESP32Arduino/RCSIM_Arduino_ESP32/ESP32 Tier 2/ESP32/`.

- `RCSIM_PC/pc_app/core/comm/esp32_strategy.py`, linie 23–30, 51–74, 83–128: transport, JSON, statystyki i CSV.
- `RCSIM_PC/pc_app/core/comm_manager.py`, linie 660–668: tworzenie strategii ESP32.
- `ESP32/ESP32.ino`, linie 21–99: funkcje, wybór płytki i pinout.
- `ESP32/ESP32.ino`, linie 116–123, 156–162, 215–299, 301–355: porty, inicjalizacja, sterowanie, failsafe i IMU.
- `ESP32V2/ESP32V2autocalibPCA/ESP32V2autocalibPCA.ino`, linie 12–14, 164–202, 296: kalibracja i sprzężenie.
- `ESP32V3+GPS+FPV/ESP32 V3 GPS + FPV`, linie 18–39: funkcje i wybór IMU/płytki.
- `ESP32V4_CRSF_MultiLink/ESP32V4_CRSF_MultiLink.ino`, linie 31–87, 103–122, 246–265: tryby, pinout, kanały i parser CRSF.