

RCSIM Tier 1 — Arduino/RP2350 Digital to PPM: instrukcja użytkownika

Ilustrowany przewodnik użytkownika: karty konfiguracji, akcje wdrożenia i procedury krok po kroku. Zrzuty przedstawiają angielski interfejs.

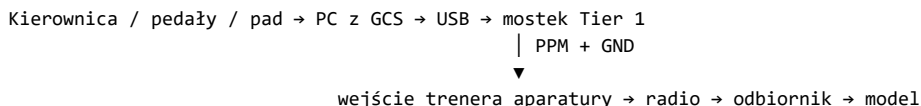
Spis treści

1. Do czego służy Tier 1	3
2. Wybierz firmware przed podłączeniem przewodów	3
2.1. Arduino Nano/Uno ATmega328P	3
2.2. Samodzielny RP2350 Tier 1	3
2.3. RP2350 z pakietu MCS — osobny wariant	4
3. Przygotowanie stanowiska i przewodów	4
3.1. Zasilanie i poziomy	4
4. Wgranie Arduino przez Arduino IDE	5
5. Wgranie XIAO RP2350	6
5.1. Gotowy plik UF2	6
5.2. Kompilacja właściwego projektu PlatformIO	6
6. Ustawienie GCS dla mostka	6
7. Kanały, kierownica i pedały — pełna kolejność	7
7.1. Definicje kanałów	7
7.2. Przypisanie wejść	8
7.3. Neutralny punkt, martwa strefa i Expo	9
8. Aparatura i odbiornik — konfiguracja trenera	10
8.1. Ogólna procedura	10
8.2. Przykład MT12 z fotografii projektu	11
9. CSV, binarny MCS i polaryzacja — czego nie mieszać	12
10. Failsafe, rozłączenie i bezpieczny powrót	13
10.1. Obowiązkowa próba bez jazdy	13
10.2. Zakończenie normalnej sesji	13
11. Aktualizacja i odzyskiwanie	13
12. Szybka diagnostyka	13
13. Źródła i granice potwierdzenia	14

Wydanie: 14 października 2026. Instrukcja opiera się na lokalnym kodzie i istniejących fotografiach projektu. Nie wykonano w jej ramach flashowania ani fizycznej próby Twojego zestawu. Zdjęcia pokazują przykłady montażu; schemat właściwego wariantu i pomiar mają pierwszeństwo przed kolorem przewodu na fotografii.

1. Do czego służy Tier 1

Tier 1 zamienia cyfrowe sterowanie z komputera na PPM dla wejścia trenera aparatury. Kierownica, pedały lub pad dostarczają wejścia do GCS; mostek Arduino albo RP2350 generuje sygnał, który aparatura przekazuje do swojego odbiornika radiowego.



Nie podłączaj zespolonego PPM bezpośrednio do zwykłego wejścia serwa lub ESC. Serwo wymaga pojedynczego kanału PWM, a tutaj sygnał zawiera wiele kanałów. PPM samo nie przesyła do GCS obrazu, GPS, IMU ani LiDAR. Obraz FPV oraz ewentualna dodatkowa telemetria są osobnymi torami.

Tier 0 generuje PPM kartą dźwiękową i nie jest opisanym tutaj mostkiem USB. Tier 2 to ESP32-CAM; Tier 3 obejmuje ELRS/mLRS, CRSF/MAVLink oraz mody EdgeTX/ExpressLRS; Tier 4 to platforma robotyczna/autonomiczna Raspberry Pi 5G. Dodatkowe tryby SBUS/iBUS/CRSF koprocatora nie zmieniają definicji Tier 1 Digital to PPM.

2. Wybierz firmware przed podłączeniem przewodów

2.1. Arduino Nano/Uno ATmega328P

Źródło: RCSIM_PC/pc_app/ESP32Arduino/RCSIM_Arduino_ESP32/Arduino Tier 1/Arduino.ino.

- USB/serial: **115200 bps**, tekstowe kanały CSV.
- Wyjście PPM: **D10** i wspólna **GND**.
- **8 kanałów**; szkic korzysta z biblioteki **PPMEncoder** i Timera1 AVR.
- Biblioteka uruchamiana jest z parametrem polaryzacji opisanym w projekcie jako dodatni. Dokładną ramkę i przebieg potwierdź dla użytej wersji biblioteki; nie kopiuj wartości 300 µs z innego firmware.
- Brak poprawnych ramek przez ponad **500 ms** albo ESTOP zatrzymuje generator i przełącza D10 w INPUT. Aparatura musi mieć sprawdzone zachowanie po utracie trenera.

2.2. Samodzielny RP2350 Tier 1

Katalog: RCSIM_PC/pc_app/ESP32Arduino/RCSIM_Arduino_ESP32/RP2350 Tier 1/. Zawiera main.cpp, platformio.ini oraz firmware.uf2 dla **Seeed Studio XIAO RP2350**.

- USB CDC: **115200 bps**, CSV, **8 kanałów**.
- PPM: **D8 / GPIO 2**, nie D7.
- Ramka **22,5 ms**, krótkie impulsy LOW **300 µs**, pomiędzy nimi HIGH przy domyślnym push-pull.
- Po ponad **500 ms** bez poprawnych danych lub ESTOP ramki ustają; domyślna wersja utrzymuje GPIO 2 w HIGH. Nie jest to to samo co przełączenie pinu w wysoką impedancję; alternatywny wariant open-drain wymaga innej kompilacji.
- Komunikat „Signal Killed” nie oznacza fizycznego odcięcia zasilania modelu.

2.3. RP2350 z pakietu MCS — osobny wariant

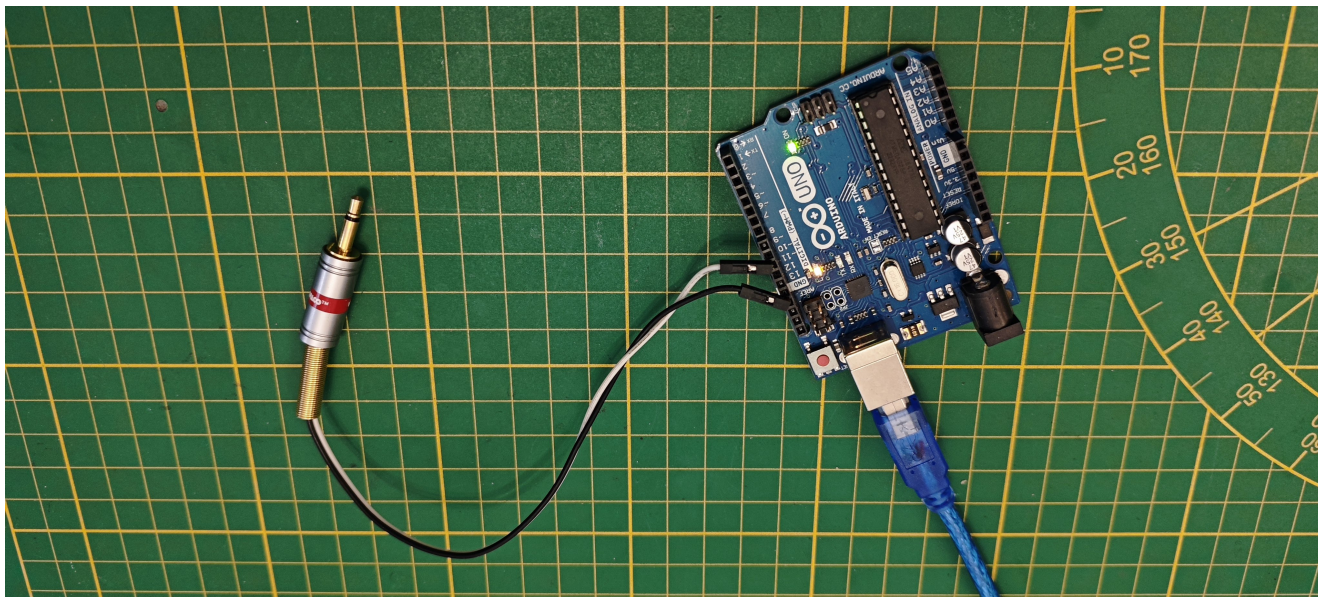
Katalog: `RCSIM_MCS/firmware/xiao_rp2350/`. Pliki `src/main.cpp`, `platformio.ini` oraz znajdujący się tutaj `firmware.uf2` nie są tym samym firmware co samodzielny Tier 1.

- Obsługuje CSV z PC oraz binarny mostek MCS. Poprawny CSV wybiera wyjście PPM; binarny strumień MCS wybiera PPM/iBUS/SBUS/CRSF.
- **D7 / GPIO 1** steruje torem MiniJack przewidzianym z odwracającym układem MOSFET; **D8 / GPIO 2** jest wyjściem szeregowym w trybach iBUS/SBUS/CRSF.
- W trybie PPM bieżące źródło wystawia przebieg także na D8. Na GPIO oba przebiegi mają krótkie LOW, natomiast tor MiniJack po MOSFET odwraca polaryzację. Nie zamieniaj D7 z D8 bez rozpoznania całego układu.
- PPM ma **8 kanałów**, ramkę **22,5 ms**, impuls **300 µs** i timeout **500 ms**. W trybie PPM po awarii impulsy ustają, a GPIO utrzymywane są w HIGH w domyślnym wariancie.
- Zatrzymanie innych protokołów ma odmienną semantykę: nie stosuj instrukcji PPM jako opisu flag failsafe SBUS.

Najważniejsze: Arduino = D10; samodzielny RP2350 = D8/GPIO 2; koprocesor MCS = D7/GPIO 1 dla przewidzianego toru MiniJack z MOSFET, z dodatkową funkcją D8 zależną od protokołu. Zapisz na obudowie płytki, który plik UF2 lub szkic został wgrany. Dwa pliki o nazwie `firmware.uf2` z różnych katalogów nie są zamienne.

3. Przygotowanie stanowiska i przewodów

Potrzebujesz zgodnej płytki, kabla USB do danych, przygotowanego firmware, GCS, kontrolera, aparatury z odpowiednim wejściem trenera, właściwego przewodu sygnałowego i odbiornika przypisanego do aparatury. Przy pierwszej próbie użyj monitora kanałów radia; pomiar oscyloskopem lub analizatorem pomaga odróżnić brak PPM od błędnego menu trenera.



Przykład z projektu: Arduino Uno, USB i przewód portu trenera

3.1. Zasilanie i poziomy

1. Odłącz napęd modelu lub unieś koła tak, aby nie mógł ruszyć. Nie pozostawiaj śmigła na napędzie lotniczym podczas konfiguracji.

2. Zasil mostek przez USB zgodnie z wymaganiami płytki. Nie łącz przypadkowo dodatkiego zasilania aparatury z USB 5 V.
3. Połącz masę mostka z masą wejścia trenera. Brak wspólnej masy może uniemożliwić odbiór lub powodować niestabilność.
4. Połącz właściwe wyjście sygnału przez interfejs zgodny z aparaturą. D10 klasycznego Uno/Nano i GPIO XIAO nie mają automatycznie takich samych poziomów elektrycznych.
5. Sprawdź dopuszczalny poziom wejścia aparatury i układ dopasowania. Komentarz „5V-tolerant” w źródle nie oznacza, że wyjście XIAO generuje 5 V, ani nie jest pozwoleniem na dowolne podanie 5 V na płytkę.
6. Zidentyfikuj styki wtyku trenera pomiarem i dokumentacją radia. Nie istnieje tu uniwersalna zasada „czubek = sygnał” dla każdego jacka i radia.

Arduino Uno/Nano: D10 — zgodny interfejs — wejście PPM trenera
 GND ————— masa trenera

RP2350 samodzielny: D8 / GPIO 2 — zgodny interfejs — wejście PPM
 GND ————— masa trenera

RP2350 firmware MCS: D7 / GPIO 1 — przewidziany MOSFET — MiniJack PPM
 GND ————— masa trenera

Schemat pokazuje funkcje przewodów, nie pełny projekt elektroniki dopasowującej. Dla wersji MCS nie pomijaj MOSFET tylko dlatego, że widzisz sygnał na GPIO; zmieni to polaryzację widzianą przez radio.

4. Wgranie Arduino przez Arduino IDE

Poniższa procedura dotyczy Uno/Nano z ATmega328P i szkicu Arduino Tier 1. Nie wybieraj ESP32 ani RP2350 jako płytki dla tego szkicu tylko dlatego, że również mają USB.

1. Zainstaluj i uruchom Arduino IDE, a następnie otwórz podany plik `Arduino.ino` z katalogu Arduino Tier 1.
2. Jeśli IDE wymaga zgodnej nazwy katalogu szkicu, zaakceptuj utworzenie jego roboczej kopii w katalogu szkiców. Nie otwieraj omyłkowo `RCSIM_MCS/firmware/co_processor/Arduino.ino`, który należy do innego mostka.
3. W menedżerze bibliotek znajdź **PPMEncoder**, wymaganą przez `#include "PPMEncoder.h"`. Sprawdź, czy wybrana biblioteka udostępnia używane `ppmEncoder.begin` i `setChannel`. Repozytorium nie zawiera ani nie przypina jej konkretnej wersji.
4. Wybierz rzeczywistą płytkę Uno albo Nano, właściwy procesor i port USB. Dla Nano użyj wariantu bootloadera zgodnego z własną płytką; nie zakładaj, że każdy klon wymaga tej samej opcji.
5. Uruchom **Verify / Weryfikuj**. Przy braku `PPMEncoder.h` wróć do instalacji biblioteki; nie usuwaj tej zależności ze szkicu.
6. Zamknij połączenie GCS i każdy monitor zajmujący port, następnie wybierz **Upload / Wgraj**.
7. Po potwierdzonym wgraniu sprawdź, czy port nadal odpowiada tej płytce. Możesz krótko otworzyć monitor 115200 bps, aby odczytać komunikat gotowości, po czym obowiązkowo go zamknij przed GCS.
8. Dopiero teraz przejdź do próby trenera. Komunikat IDE o sukcesie potwierdza zapis szkicu, nie poprawny poziom PPM ani failsafe radia.

Problemy z wgrywaniem: brak COM → sprawdź kabel danych i sterownik adaptera; port zajęty → zamknij GCS/monitor; błąd synchronizacji Nano → sprawdź płytkę, procesor i bootloader. Nie zmieniaj połączeń D10, aby naprawić błąd USB.

5. Wgranie XIAO RP2350

5.1. Gotowy plik UF2

1. Wybierz jedną implementację z rozdziału 2 i jej dokładny `firmware.uf2`. Zanotuj katalog pochodzenia; nie wybieraj przypadkowego pliku o tej nazwie z wyszukiwania systemu.
2. Rozłącz GCS, zamknij monitory i odłącz przewód sygnałowy od radia na czas programowania.
3. Podłącz XIAO RP2350 kablem USB do danych. Przytrzymaj **BOOT**, naciśnij i zwolnij **RESET**, następnie zwolnij BOOT. Jeśli korzystasz z wariantu wchodzenia do bootloadera przy podłączeniu USB, postępuj według przycisków własnej płytki.
4. Poczekaj na pojawienie się pamięci bootloadera RP. Jej nazwa może zależeć od płytki/bootloadera; identyfikuj nowo pojawiający się nośnik, nie przypadkowy dysk komputera.
5. Skopiuj wybrany plik UF2 na ten nośnik. Poczekaj na zakończenie zapisu i restart płytki; nośnik zwykle znika, a pojawia się urządzenie USB CDC.
6. Zidentyfikuj nowy port COM/USB. Przywróć przewód zgodnie z firmware: D8 dla samodzielnego mostka lub przewidziany tor D7/MOSFET dla MCS.
7. Sprawdź działanie na stanowisku od początku. Obecność artefaktu w repozytorium nie jest dowodem zgodności jego zawartości z każdą późniejszą zmianą źródeł.

5.2. Kompilacja właściwego projektu PlatformIO

Dla samodzielnego Tier 1 otwórz istniejący katalog **RP2350 Tier 1** jako projekt PlatformIO. Jego plik konfiguracyjny ustawia źródła w katalogu projektu; nie przenoś `main.cpp` do `src` na podstawie starszego opisu. Wybierz środowisko **seeed_xiao_rp2350** i wykonaj **Build**. Po poprawnej kompilacji użyj wygenerowanego UF2 z tego projektu do procedury bootloadera.

Dla koprocatora MCS otwórz oddzielnie **RCSIM_MCS/firmware/xiao_rp2350**. Źródło jest tutaj w `src/main.cpp`, wymagane jest także Adafruit NeoPixel, a konfiguracja używa `picotool` dla wgrywania. Wybierz właściwe środowisko, wykonaj Build, przygotuj BOOT/RESET i dopiero potem Upload. Nie łącz obu `platformio.ini` w jeden profil.

Oba projekty wskazują płytkę **Seeed Studio XIAO RP2350** i rdzeń Earle Philhower. Nie przedstawia to wsparcia każdego RP2040/RP2350 o podobnym kształcie. Jeśli narzędzie nie widzi płytki w bootloaderze, sprawdź kabel, tryb BOOT i dostęp do USB przed kolejną próbą.

Nie uruchamiaj do zwykłego programowania niezweryfikowanego skryptu automatycznego SSH z katalogu firmware. Do aktualizacji koprocatora podłączonego do Pi użyj udokumentowanego panelu MCS i własnych parametrów połączenia; ten podręcznik opisuje bezpośrednio USB.

6. Ustawienie GCS dla mostka

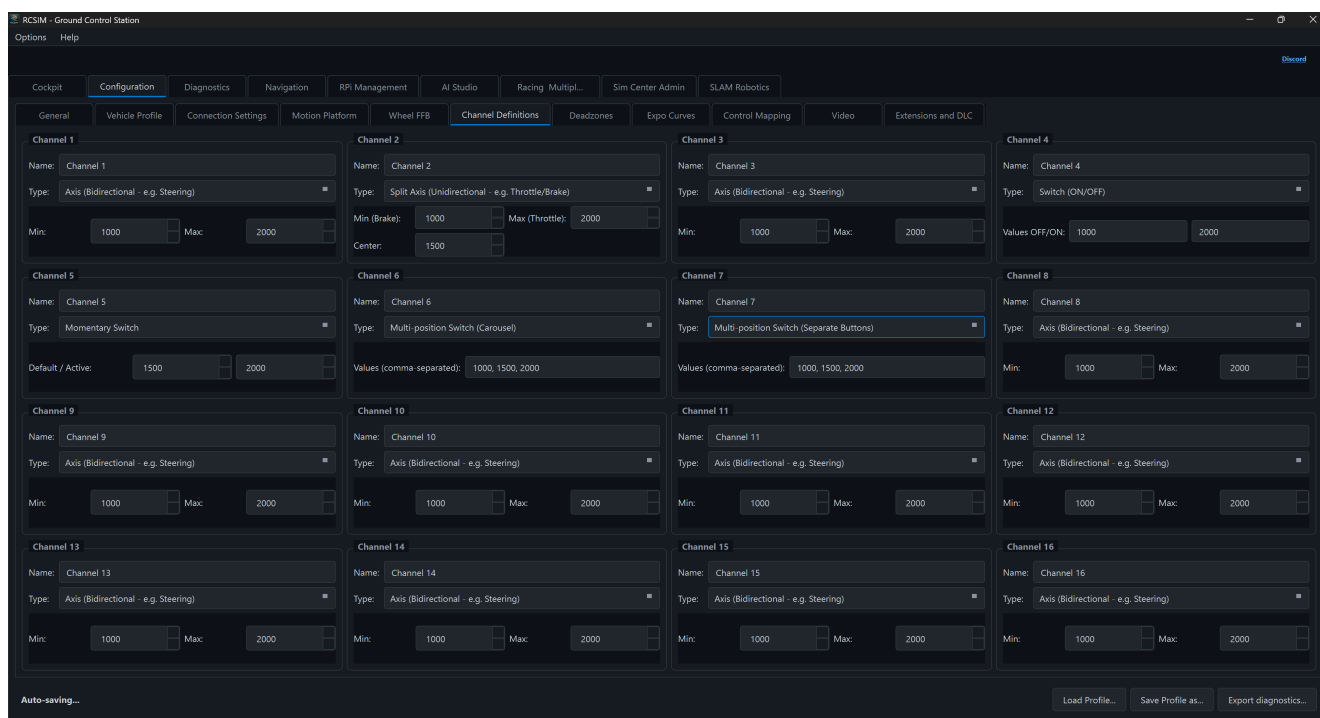
1. Podłącz kierownicę/pad oraz mostek. Uruchom GCS i otwórz **Configuration** → **Connection Settings**.
2. W **GCS Mode** wybierz **Arduino (Port Trenera)** dla Arduino albo **RP2350 (Port Trenera)** dla XIAO. Otwarcie karty Tier 1 bez zmiany GCS Mode nie aktywuje tego połączenia.
3. Na **Tier 1 (Serial / USB)** użyj **Refresh Ports**, wskaż właściwy **Port** w sekcji danego urządzenia i ustaw **Baud = 115200**.
4. Zakończ edycję i zaczekaj na automatyczny zapis. **Save Profile as...** pozwala zachować kopię profilu stanowiska; nie wgrywa firmware.

5. W **Cockpit** wybierz **Manual (Driver)**, potwierdź **Connection Mode**, następnie **Connect**. Pomyślne otwarcie COM nie jest jeszcze potwierdzeniem odbioru PPM w radiu.
6. Najpierw sprawdź kanały w GCS. Dopiero przy zabezpieczonym napędzie użyj **ARM SYSTEM**, aby zezwolić na przesyłanie sterowania i zdjąć blokadę ESTOP mostka.
7. Sprawdź monitor kanałów aparatury. Bez uzbrojenia GCS zwykły strumień sterowania jest blokowany; brak ruchu w radiu nie musi oznaczać uszkodzonego przewodu.

DTR/RTS: bieżąca strategia GCS dla RP2350 ustawia te linie na True i czeka około **0,3 s** na USB CDC. Arduino używa zwykłego otwarcia portu; płytkę może się resetować. Nie przenoś porad o wyłączonych DTR/RTS z nadajnika Nomad lub MCS na ten tor. Po ponownym otwarciu portu odczekaj na gotowość i powtórz neutralną próbę; nie zmieniaj przewodów zasilania, aby ominąć reset.

7. Kanały, kierownica i pedały — pełna kolejność

7.1. Definicje kanałów



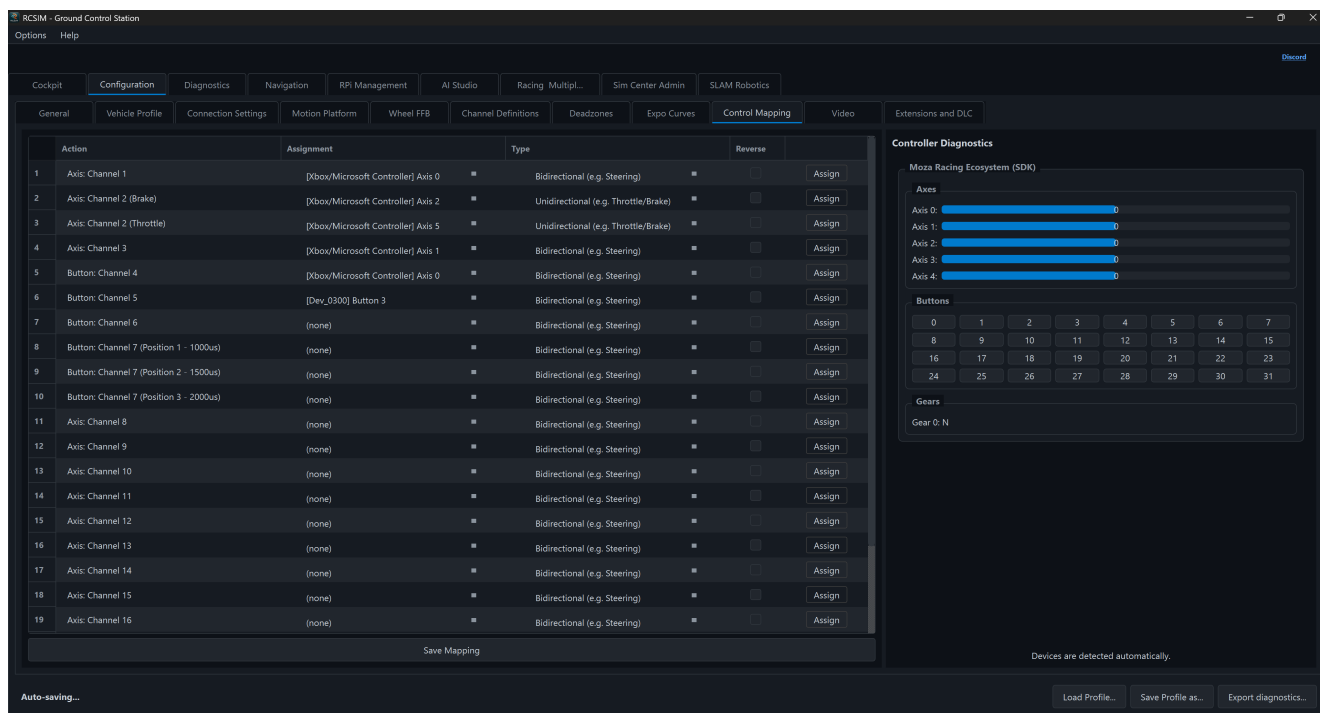
GCS — definicje osi i przełączników

Otwórz **Configuration** → **Channel Definitions**. Nazwij CH1 jako skręt i wybierz **Axis (Bidirectional)**. Dla samochodowego ESC z gazem/hamulcem/rewersem na jednym kanale możesz ustawić CH2 jako **Split Axis (Unidirectional)**, aby osobno przypisać pedały. To przykład układu, nie uniwersalna kolejność radia.

Ustaw Min, Max i Center zgodnie z ESC/serwem oraz aparaturą. Zaczynij od ograniczonego zakresu, jeśli nie znasz skrajnych położenia mechanizmu. Nie ustawiaj ekstremów tylko po to, aby pasek doszedł do końca.

Mostki PPM opisane tutaj przenoszą **pierwsze 8 kanałów**. Nie przypisuj ważnej funkcji wyłącznie do CH9–CH16. GCS dla RP2350 obcina listę do 8; firmware Arduino również przetwarza 8 wartości. Zakres akceptacji parsera **800–2200 μs** nie jest zalecanym zakresem serwa; zwykły start to zgodne z własnym sprzętem **1000–2000 μs** i właściwy punkt neutralny.

7.2. Przypisanie wejść

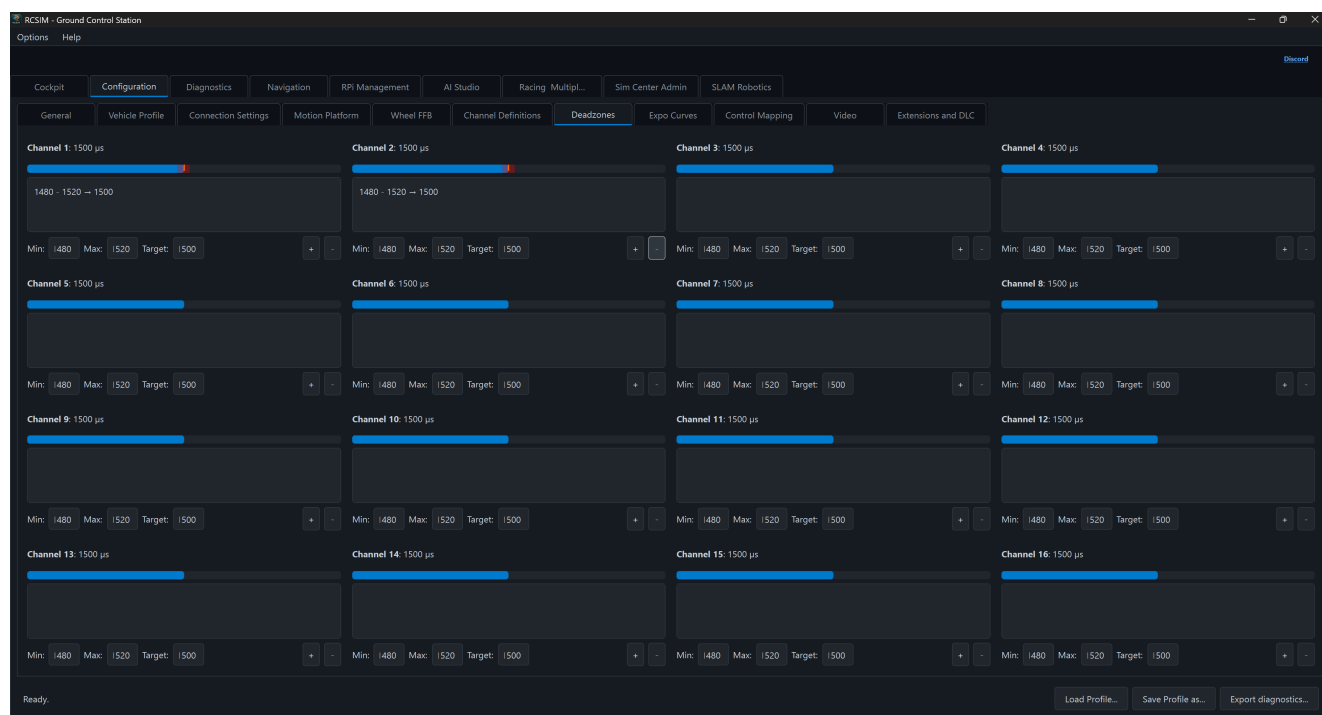


GCS — przypisywanie osi, pedałów i przycisków

1. Przejdź do **Control Mapping**, naciśnij **Assign** przy skřęcie i porusz wyłącznie kierownicą.
2. Przypisz osobno **Brake** i **Throttle** dla kanału dzielonego. Zwolnij pozostałe pedały podczas wykrywania.
3. Dobierz **Type**: skřęt jest dwukierunkowy, pedał jednokierunkowy. Numery Axis ze zrzutu są przykładem jednego kontrolera, nie instrukcją wyboru osi każdego urządzenia.
4. Użyj **Reverse**, jeśli faktyczny kierunek jest błędny. Po zmianie sprawdź pełny zakres i stan zwolnionego pedału.
5. Przypisz potrzebne przełączniki mieszczące się w 8 kanałach, następnie naciśnij **Save Mapping**.
6. W podglądzie kanałów porusz każdą osią osobno. Dwa pedały nie powinny przypadkiem sterować tą samą akcją z przeciwnych źródeł.

Wynik: kierownica wraca do właściwego środka, zwolniony gaz nie żąda ruchu, hamulec działa w zamierzonym kierunku. Potwierdź ten sam rezultat w aparaturze i odbiorniku, nie tylko w GCS.

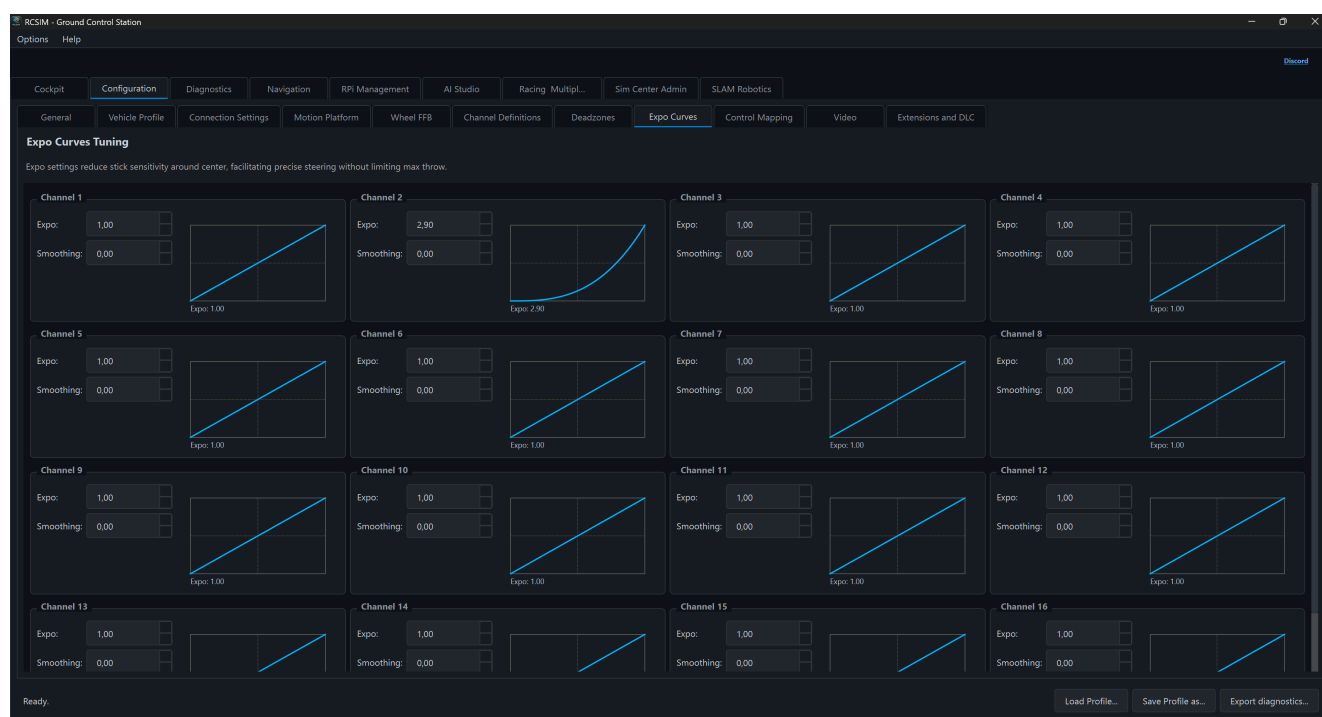
7.3. Neutralny punkt, martwa strefa i Expo



GCS — pomiar kanałów i reguły martwych stref

W **Deadzones** obserwuj spoczywający kanał. Reguła Min–Max → Target usuwa drobne drgania, np. **1480–1520 → 1500** tylko dla kanału, którego prawdziwy neutralny punkt wynosi 1500. Nie stosuj tej reguły do gazu lotniczego z minimum na 1000 ani do innego neutralnego punktu ESC.

Najpierw ustal działający punkt neutralny i kierunek, potem dodaj najmniejszą potrzebną martwą strefę. Szeroka strefa maskująca połowę ruchu pedału zwykle oznacza błędny typ osi lub zakres, a nie poprawną kalibrację.



GCS — krzywe Expo i wygładzanie

W **Expo Curves** wartość **1,00** jest liniowa; większa łagodzi początek odpowiedzi. **Smoothing = 0,00** nie wygładza. Zwiększaj je małymi krokami dopiero po poprawnym mapowaniu: mocne wygładzanie opóźnia reakcję.

Kalibracja osi wejściowej i kalibracja ESC to różne zadania. Procedurę uczenia ESC uruchamiaj według instrukcji konkretnego regulatora, dopiero gdy zakres i neutralny sygnał odbiornika są rozpoznane. Nie wykonuj przypadkowych pełnych wychyleń na uzbrojonym napędzie.

8. Aparatura i odbiornik — konfiguracja trenera

8.1. Ogólna procedura

1. Wybierz właściwy model w aparaturze. Zachowaj jego konfigurację przed zmianami.
2. Skonfiguruj wejście trenera jako odpowiednie **PPM/CPPM**, z właściwym portem, polaryzacją i liczbą kanałów. USB joystick, USB-VCP CRSF i wejście SBUS nie są tym samym wejściem co PPM MiniJack.
3. Włącz rolę przyjmowania zewnętrznego sterowania dla właściwego radia. W EdgeTX nazwy zależą od modelu, np. odpowiedni wariant Master/Jack dla wejścia trenera; nie wybieraj Master/CRSF tylko dlatego, że radio nadaje po ELRS.
4. W menu trenera dopasuj kolejność wejść do funkcji modelu. Sprawdź, czy skręt/gaz nie zamieniły się z kanałami Aileron/Elevator/Throttle/Rudder.
5. Ustaw aktywację trenera świadomym przełącznikiem lub właściwą funkcją specjalną. Sprawdź, czy po jego zwolnieniu sterowanie wraca do zamierzonego źródła.
6. Najpierw obserwuj wejścia i monitor kanałów bez możliwości ruchu modelu. Dopiero potem sprawdź odbiornik RF i ESC.

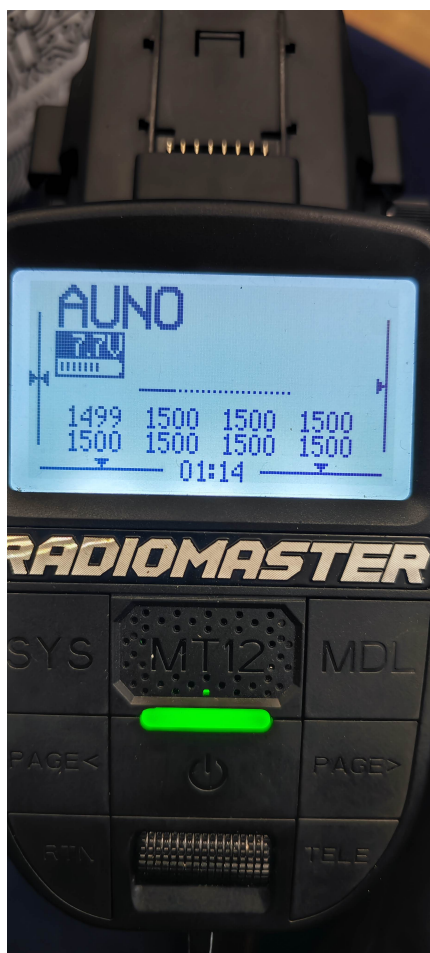
Nie istnieje jedna sekwencja menu dla wszystkich aparatów EdgeTX i wszystkich wejść. Użyj instrukcji własnego modelu dla styku złącza, poziomu sygnału i wyboru portu. Zwykły odbiornik radiowy ma być przypisany do aparatury; USB mostek nie zastępuje tego parowania.

8.2. Przykład MT12 z fotografii projektu



MT12 — przykład aktywowania funkcji specjalnej Trainer przełącznikiem

Zdjęcie pokazuje w **SPECIAL FUNCTIONS** funkcję **Trainer**, aktywowaną przełącznikiem oraz zaznaczone włączenie. To przykład konfiguracji z projektu, nie wymóg użycia dokładnie tego przełącznika ani dowód zgodności każdej wersji EdgeTX.



MT12 — podgląd wartości 8 kanałów

Podczas próby sprawdź, czy ruch jednej osi GCS powoduje zmianę jednej zamierzonej funkcji w monitorze. Bliskie środka wartości na zdjęciu nie są gotową konfiguracją gazu Twojego ESC. Jeśli radio pokazuje brak trenera mimo aktywnego USB mostka, najpierw sprawdź PPM, port i polaryzację.

9. CSV, binarny MCS i polaryzacja — czego nie mieszać

GCS wysyła do opisanych mostków tekstową linię wartości w μs , oddzielonych przecinkami i zakończonych nową linią. Komendy ARM/ESTOP również są tekstowe. Nie potrzebujesz ręcznego terminala do normalnego sterowania; robi to GCS.

Samodzielny Arduino i samodzielny RP2350 nie są odbiornikami binarnej ramki MCS. Koprocesor MCS obsługuje dodatkowo **19 bajtów** z nagłówkiem **0xAA**, wyborem protokołu i **8** wartości kanałów. Omyłkowe wskazanie jednego firmware jako drugiego może dać poprawny port USB bez poprawnego sterowania.

W koprocesorze MCS poprawna ramka binarna może zmienić protokół i wyczyścić blokadę ESTOP; nie traktuj go jako tego samego zatrzasku co samodzielny szkic Arduino. Tylko jeden host powinien używać mostka naraz. Nie otwieraj jednocześnie portu przez PC GCS i usługę MCS.

Polaryzację ocenia się na wejściu aparatury, po wszystkich tranzystorach i interfejsach. Napis Positive/Negative PPM w bibliotece, krótkie LOW na GPIO i odwrócony MOSFET nie są trzema opisami tego samego punktu pomiaru. Po zmianie firmware albo pominięciu interfejsu ponownie sprawdź przebieg na rzeczywistym złączu.

10. Failsafe, rozłączenie i bezpieczny powrót

10.1. Obowiązkowa próba bez jazdy

1. Przygotuj sprawny ręczny sposób zatrzymania i zabezpiecz napęd.
2. Uzbrój sterowanie na stanowisku, sprawdź skręt/gaz i zanotuj neutralny punkt odbiornika.
3. Rozbrój w GCS. Sprawdź rzeczywisty brak impulsów PPM lub oczekiwaną reakcję odbiornika; sam napis ESTOP nie potwierdza zatrzymania modelu.
4. Ponownie przygotuj połączenie i odłącz USB lub przerwij strumień sterowania. Timeout firmware wynosi ponad **500 ms** od ostatniej poprawnej ramki; nie jest to gwarancja całkowitego czasu zatrzymania radia i ESC.
5. Sprawdź utratę sygnału trenera w aparaturze, potem reakcję odbiornika i ESC. Radio może wrócić do własnych drążków albo utrzymać inną zaprogramowaną logikę; brak PPM nie oznacza automatycznie odcięcia gazu.
6. Osobno przetestuj utratę RF. To inne zdarzenie niż utrata USB/PPM.
7. Przy przywróceniu USB, restarcie GCS lub ARM utrzymuj gaz neutralnie i model zabezpieczony. Sprawdź, czy nic nie rusza bez świadomego działania.

Arduino: ESTOP zatrzymuje blokadę do ARM; przy zwykłym timeoutcie poprawne nowe dane mogą przywrócić sygnał. **Samodzielny RP2350:** ESTOP również blokuje CSV do ARM, generator w awarii nie nadaje ramek. **MCS RP2350:** w PPM generator zatrzymuje impulsy, ale dalsze zachowanie zależy także od tego, czy host używa CSV czy binarnego mostka.

10.2. Zakończenie normalnej sesji

Zatrzymaj model ręcznie, ustaw neutralny gaz, rozbrój i potwierdź stan odbiornika. Rozłącz GCS, zabezpiecz napęd i dopiero odłączaj interfejs. Nie traktuj zamknięcia okna, restartu komputera ani odłączenia kabla jako normalnego hamulca.

11. Aktualizacja i odzyskiwanie

Przed aktualizacją zapisz profil GCS, ustawienia modelu aparatury, oznaczenie płytki i katalog użytego firmware. Odłącz napęd oraz przewód PPM. Zamknij GCS i monitory portu.

Dla Arduino powtórz procedurę IDE z właściwą płytką i biblioteką. Dla XIAO wejdź fizycznie do BOOT/RESET i wgraj UF2 z wybranego projektu. Po powrocie portu sprawdź jego numer: system może przypisać inny COM.

Jeśli po wgraniu nie ma sterowania, nie przełączaj od razu na inną polaryzację i inny pin. Najpierw ustal, czy wgrano wariant samodzielny czy MCS, czy GCS używa właściwej strategii i czy port jest wolny. Przywróć poprzedni znany plik tylko wtedy, gdy masz potwierdzone jego pochodzenie i właściwe okablowanie.

Po każdej zmianie firmware, biblioteki, płytki albo aparatury powtórz próbę kanałów, neutralnego gazu, rozbrojenia, zaniku USB, zaniku RF oraz powrotu połączenia. Dawny test innego wsadu nie potwierdza nowego zestawu.

12. Szybka diagnostyka

- **Nie ma COM:** sprawdź kabel danych, zasilanie USB, bootloader i sterownik. Pamięć UF2 i działający port CDC są różnymi stanami urządzenia.
- **Port zajęty:** zamknij Serial Monitor, konfigurator, drugi GCS lub usługę MCS używającą tego portu.

- **GCS połączony, radio nie widzi trenera:** sprawdź ARM SYSTEM, wariant firmware, właściwy GPIO, masę, poziom elektryczny i rolę wejścia PPM w radiu.
- **Radio widzi kanały, model nie reaguje:** sprawdź aktywację trenera/miksery, właściwy model aparatury, przypisanie odbiornika i uzbrojenie ESC.
- **Gaz po puszczeniu nie wraca:** sprawdź typ osi, Reverse, Center i realną neutralną wartość ESC. Nie naprawiaj tego tylko Expo.
- **Kanały skaczą:** sprawdź wspólną masę, przebieg PPM, obciążenie złącza i spójność biblioteki/firmware. Nie obiecuj określonego jitteru bez pomiaru.
- **Po awarii mostek wraca, model rusza:** zatrzymaj próby ruchu i popraw logikę trenera, neutralne wejścia i failsafe odbiornika przed kolejnym uzbrojeniem.
- **Brak GPS/wideo/FFB IMU przez ten port:** to ograniczenie toru PPM, a nie brakujący parametr Baud. Potrzebny jest osobny zgodny tor danych.

13. Źródła i granice potwierdzenia

Procedury odpowiadają kodowi lokalnemu; zgodność każdego artefaktu binarnego ze źródłem, poziomy na Twojej płytce i reakcja konkretnego radia/ESC wymagają osobnej weryfikacji. Nie ma tutaj gwarancji zasięgu RF, jitteru ani czasu zatrzymania całego modelu.

- RCSIM_PC/pc_app/ESP32Arduino/RCSIM_Arduino_ESP32/Arduino Tier 1/Arduino.ino:16–22,36–43,65–76,91–113,154–172 — pin, biblioteka, watchdog i ARM/ESTOP.
- RCSIM_PC/pc_app/ESP32Arduino/RCSIM_Arduino_ESP32/RP2350 Tier 1/main.cpp:20–34,101–126,213–238,263–276 — przebieg PPM, CSV i zatrzymanie generatora.
- RCSIM_PC/pc_app/ESP32Arduino/RCSIM_Arduino_ESP32/RP2350 Tier 1/platformio.ini:1–10 — samodzielny projekt i właściwa płytka.
- RCSIM_MCS/firmware/xiao_rp2350/src/main.cpp:20–42,173–196,243–275,382–449,455–510,558–587 — piny, dwa wejściowe formaty i zachowanie PPM.
- RCSIM_MCS/firmware/xiao_rp2350/platformio.ini:1–28 — konfiguracja MCS, picotool i NeoPixel.
- RCSIM_MCS/src/output/bridge.py:15–56 — binarny mostek MCS i limity.
- RCSIM_PC/pc_app/core/comm/serial_strategy.py:48–51,84–91,108–131 — otwarcie Arduino, CSV i komendy.
- RCSIM_PC/pc_app/core/comm/rp2350_strategy.py:47–55,101–113,124–147 — DTR/RTS, 8 kanałów i komendy RP2350.
- RCSIM_PC/pc_app/core/comm_manager.py:232–258,483–498 — uzbrojenie i blokowanie normalnego strumienia.
- RCSIM_PC/pc_app/gui/tabs/connection_ui.py:94–101,195–205 — wybór mostka i prędkości.