

A G H

AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

WYDZIAŁ INŻYNIERII METALI I INFORMATYKI PRZEMYSŁOWEJ

Katedra Informatyki Stosowanej i Modelowania

Projekt dyplomowy

*Opracowanie systemu omijania przeszkód dla dronów FPV
Development of an Obstacle Avoidance System for FPV Drones*

Autor: *Kacper Farion*

Kierunek studiów: Informatyka Techniczna

Opiekun projektu: *prof. dr hab. inż. Andrij Milenin*

Kraków, 2025

Z głębokim szacunkiem i wdzięcznością, niniejszą pracę dedykuję Panu Profesorowi Andrijowi Mileninowi – za jego nieocenioną pomoc, wsparcie oraz cierpliwość w trakcie jej powstawania; moim najbliższym, a szczególnie mamie oraz tacie, za wsparcie w dążeniu do życiowych celów oraz niezłomne powtarzanie, żeby nigdy się nie poddawać; Kuzynowi, za cierpliwość w trakcie druku obudowy, która wielokrotnie musiała zostać przerobiona i wydrukowana ponownie; oraz wszystkim, którzy przez setki godzin znosili moje opowieści o pasji do dronów, zaszczepionej we mnie przez Pana Profesora.

Spis treści

1.	Wstęp i cele pracy	5
2.	Przegląd możliwych rozwiązań	8
2.1.	Interfejs UART	8
2.2.	Interfejs I ² C	8
2.3.	Technika PWM	9
2.4.	Protokół MSP	9
3.	Wybór i charakterystyka komponentów	10
3.1.	Wybór i charakterystyka komponentów jednostki latającej - drona	10
3.1.1.	Kontroler lotu	10
3.1.2.	Rama – główny element nośny	11
3.1.3.	Silniki bezszczotkowe	12
3.1.4.	Odbiornik RC	12
3.1.5.	Kamera FPV	13
3.1.6.	Nadajnik video oraz antena	14
3.1.7.	Bateria	15
3.1.8.	GPS	15
3.1.9.	Śmigła	16
3.1.10.	Elementy dodatkowe	17
3.2.	Wybór i charakterystyka kontrolera RC – radia	17
3.3.	Wybór i charakterystyka gogli FPV	18
3.4.	Wybór i charakterystyka platformy Arduino	19
3.5.	Wybór i charakterystyka czujników ultradźwiękowych	22
4.	Wybór, charakterystyka oraz konfiguracja oprogramowania	24
4.1.	Wybór, charakterystyka i konfiguracja oprogramowania jednostki latającej - drona	24
4.2.	Charakterystyka i konfiguracja oprogramowania kontrolera RC	53
5.	Implementacja systemu omijania przeszkód	65

5.1.	Schemat połączeń i montaż komponentów jednostki latającej – drona	65
5.2.	Schemat połączeń i montaż komponentów platformy Arduino.....	68
5.3.	Implementacja systemu omijania przeszkód w programie Arduino IDE	71
5.4.	Integracja i montaż systemu.....	81
6.	Testy systemu.....	91
6.1.	Testy laboratoryjne w kontrolowanych warunkach	91
6.2.	Eksperymentalne loty testowe w rzeczywistym środowisku.....	93
6.3.	Analiza skuteczności unikania przeszkód w zależności od parametrów lotu	94
7.	Podsumowanie, wnioski końcowe oraz przyszłe kierunki badań	95
7.1.	Ocena efektywności zastosowanego systemu.....	95
7.2.	Możliwości dalszego rozwoju.....	95
7.3.	Propozycje optymalizacji algorytmów unikania przeszkód.....	95
8.	Bibliografia	96
	Spis rysункów	100
	Spis tabel	104
	Spis zdjęć	104
	Spis fragmentów kodu	106
	Spis schematów	107
	Spis równań.....	107

1. Wstęp i cele pracy

W dobie rozwijających się technologii Bezzałogowych Statków Powietrznych (BSP) oraz bardzo szerokiego spektrum ich wykorzystywania, a także wobec rozwijającej się podkategorii dronów FPV (z ang. *first person view*), niewspółmiernie rośnie również zapotrzebowanie na technologie związane z automatyzacją procesów wspomagających pilotów niniejszych statków, w procesie uczenia pilota. Rosnące zapotrzebowanie okazuje się występować nie tylko w celach wyspecjalizowanych jednostek terytorialnych, służb, ale również w przypadku osób fizycznych, chcących dokonać nagrań w celach własnych. W czasie rosnącej liczby wypuszczanych przez pilotów maszyn, można wysnuć pytanie, w jaki sposób poprawnie i bez szkód nauczyć się latać bezzałogowymi statkami powietrznymi, gdyż sam proces wymaga poświęcenia mu sporej ilości czasu, a także poczynienia odpowiednich kroków przed samym lataniem. Samo w sobie sterowanie dronem jest charakterystyczne, gdyż użytkownik porusza się w trzech płaszczyznach gałkami kontrolera, który płaszczyzn ma dwie. Współczesnym głównym problemem niszy dronów FPV, jest wybór pomiędzy gotowym oprogramowaniem zapewnionym przez producentów, a koniecznością budowy BSP od podstaw, co z kolei daje bardzo duże możliwości personalizacyjne, jednakże wiąże się z potrzebą poświęcenia niemałej ilości czasu na proces zaznajamiania się z technologią działania poszczególnych urządzeń niezbędnych do lotu.

Wobec powyższego, zdecydowano się opracować system, który proces uczenia się latania BSP jest w stanie ułatwić, co może przyczynić się do zmniejszonej w trakcie tego procesu ilości szkód, gdyż maszyny same w sobie charakteryzują się małą podatnością na uszkodzenia mechaniczne. Podkategorię dronów FPV charakteryzuje również zwiększoną precyzja w stosunku do zwykłych dronów oraz możliwość szczegółowej personalizacji ich oprogramowania oraz używanych komponentów – w zasadzie cały proces konstrukcji oraz wgrywania oprogramowania może zależeć od przyszłego pilota. W dobie panujących wszędzie mikrokontrolerów oraz obsługiwanych przez nie czujników, stosunkowo prostym wydaje się być sformułowanie tezy, iż nawet jeżeli jeden ze sprawdzonych systemów operacyjnych drona FPV, nie jest kompatybilny z danym typem czujników, to można to właśnie za pomocą mikrokontrolera w profesjonalny sposób obejść. Dowiedzenie powyższej tezy jest o tyle skomplikowane, o ile wymaga od pilota znajomości kompatybilnych z danym kontrolerem lotu obecnym na dronie – odpowiedniakiem procesora dla komputerów PC – protokołów komunikacji. Analogicznie dla wybranej platformy i modelu mikrokontrolera. Jako że jedną z najbardziej powszechnych platform mikrokontrolerów jest platforma Arduino, zdecydowano, że wobec faktu, iż ww. platforma obsługuje ogromną liczbę czujników, w tym różnych czujników do pomiaru odległości, będzie ona dobrą podstawą do

zrealizowania niniejszego systemu. Należy jednak stwierdzić, iż w czasach ciągle rozwijających się technologii dronów FPV, które już w samej ilości użytkowników należą do mniejszości, takie rozwiązania nie są rozwiązaniami standardowymi, mają one bardziej wymiar eksperymentalny, wobec czego, chcąc wykonać część zasadniczą niniejszej pracy, należy wziąć pod uwagę, iż jest to tematyka bardzo specyficzna, mogąca bardzo, lub przeciwnie niewiele, różnić się w stosunku do poszczególnych modeli komponentów montowanych na dronie FPV podczas budowy.

Warto również zaznaczyć, że choć w typowych dronach, posiadających zaimplementowaną domyślnie kontrolę wizualną oraz stabilizację stosuje się już podobne systemy wykrywania przeszkód, to w przypadku dronów FPV pojawia się wyzwanie, wynikające *stricto* ze specyfiki tejże podgrupy – pilot nie ma sposobu na dostrzeżenie tego co znajduje się za dronem, zakładając, iż używa gogli FPV. W dronach FPV nie tyle nie ma sensu wprowadzać automatycznego systemu omijania przeszkód, co byłoby to rozwiązanie niepraktyczne z uwagi na często wymagany poziom precyzji podczas wykonywanych operacji. Istotnym aspektem natomiast, byłoby w sytuacji awaryjnej ostrzeżenie pilota, że to za nim znajduje się przeszkoda – na wzór pierwotnych kamer cofania, wyposażonych jedynie w czujniki reagujące dźwiękowo w zależności od przeszkody, lub wypisując prosty komunikat w postaci symbolu na desce rozdzielczej samochodu z tą różnicą, że pilot drona FPV nie ma żadnej możliwości obrócenia się i dostrzeżenia przeszkody. Niniejsza praca skupia się na wprowadzeniu nowego podejścia dedykowanego dronom FPV, które zazwyczaj są kontrolowane przez pilota w trybie surowym (acro) a szczególnie w przypadku mniej doświadczonych pilotów wymagających takiego wsparcia.

Celem niniejszej pracy zostało nie tylko dowiedzenie tezy wskazanej we wstępie, ale również usprawnienie tudzież ułatwienie procesu uczenia się latania pilotów w warunkach prawdziwych. O ile mając do dyspozycji narzędzia takie jak symulatory dla dronów FPV proces ten można bardzo mocno usprawnić oraz zminimalizować ryzyko nieprawidłowych manewrów, za pomocą spędzenia odpowiedniej ilości godzin w symulatorze^[1], o tyle proces ten można dodatkowo usprawnić i pomóc początkującym pilotom.

Niniejsze opracowanie, może również stanowić podstawę dla przyszłych pilotów dronów FPV, chcących zgłębić tajniki funkcjonowania poszczególnych komponentów obecnych na pokładzie drona, w tym nadajnika wideo niejako przekazującego obraz z kamery do gogli FPV jednym z dwóch możliwych do wyboru sposobów komunikacji, modułu GPS udzielającego informacji o położeniu drona w przestrzeni i innych aspektów wiążących się z pilotowaniem drona. Wobec tego, w niniejszej

pracy znaleźć można nie tylko suchy opis poszczególnych komponentów, ale również uwagi i wskazówki, jak postępować oraz co może przytrafić się podczas budowy, wgrywania i testowania BSP należącego do niszy FPV.

Przygotowane opracowanie zawiera również porównania, które nadają szerszy kontekst możliwym wyborom komponentów. Wynika to z faktu, że drona FPV można zbudować i oprogramować na co najmniej kilka sposobów, a sam wybór systemu wgrywanego na kontroler lotu nie ma charakteru deterministycznego.

W niniejszym tekście posłużono się terminologią zapożyczoną z języka angielskiego, która w wielu przypadkach nie ma precyzyjnych odpowiedników w języku polskim. Do takich pojęć należą m. in.:

- *stack* – zestaw składający się z kontrolera lotu oraz kontrolera silników (ESC), dostarczany najczęściej w formie gotowego modułu,
- *frame* – zależnie od kontekstu, oznacza ramę drona (główną konstrukcję nośną) lub ramkę danych przesyłaną pomiędzy mikrokontrolerem Arduino a kontrolerem lotu,
- *freestyle, long range* – określenie stylów latania dronami FPV; pierwszy związany jest z wykonywaniem dynamicznych manewrów przy użyciu odpowiednio dobranej maszyny, drugi natomiast odnosi się do lotów dalekiego zasięgu i stabilnych ujęć realizowanych przez BSP.

2. Przegląd możliwych rozwiązań

Mówiąc o komunikacji pomiędzy kontrolerem lotu oraz platformą Arduino, należało na samym początku rozważyć sposób komunikacji oraz połączeń pomiędzy wymienionymi urządzeniami. W tym przypadku, można było posłużyć się przynajmniej trzema podstawowymi interfejsami sprzętowej komunikacji i są to kolejno:

- UART (z ang. *Universal asynchronous receiver-transmitter*),
- I²C (z ang. *Inter-Integrated Circuit*),
- PWM (z ang. *Pulse Width Modulation*) .

2.1. Interfejs UART

UART, czyli uniwersalny asynchroniczny nadajnik-odbiornik, jest sprzętowym interfejsem komunikacyjnym, który konwertuje dane równolegle na strumień szeregowy i odwrotnie, umożliwiając wymianę informacji pomiędzy urządzeniami bez wspólnej linii zegarowej^[2]. Jest to rozwiązanie sprzętowe o tyle interesujące w kontekście dronów, co umożliwiające zastosowanie wielu protokołów komunikacji pomiędzy Arduino a kontrolerem lotu. Do najczęściej wykorzystywanych protokołów komunikacji w dronach – w tym dronach FPV – zaliczyć można: MSP (*MultiWii Serial Protocol*), CRSF (*Crossfire Serial Protocol*), SmartPort (*FrSky*), MAVLink czy iBUS (FlySky).

2.2. Interfejs I²C

Jest to powszechnie stosowana, dwuprzewodowa magistrala szeregowa, zaprojektowana przez firmę Philips (obecnie NXP), służąca do komunikacji między układami scalonymi. I²C wykorzystuje w komunikacji dwie linie: SDA – jako linię danych oraz SCL – jako linię zegara i umożliwia komunikację typu master-slave a kierunek transakcji wynika z bitu R/W w adresie. Magistrala obsługuje wiele urządzeń z unikalnymi adresami, zapewniając mechanizmy arbitrażu oraz synchronizacji, co czyni ją elastycznym i wydajnym rozwiązaniem w integracji czujników i peryferiów^[3]. W przypadku rozważania komunikacji pomiędzy platformą Arduino a kontrolerem lotu drona, ważnym aspektem okazuje się natomiast być elastyczność wysyłanej ramki danych – a ta w przypadku I²C jest z góry ściśle zdefiniowana. Jej użycie wyklucza również wiele kontrolerów lotu, gdyż ich warstwa sprzętowa nie jest dostosowana do obsługi wielu urządzeń peryferyjnych przy pomocy magistrali I²C – w tym przypadku, jest to interfejs komunikacyjny wykorzystywany do

połączenia do kontrolera lotu ścisłe określonych urządzeń takich jak GPS, barometr czy magnetometr.

2.3. Technika PWM

Jest techniką modulacji szerokości impulsu prostokątnego, stosowana zwłaszcza w systemach dronów do regulacji prędkości silników i pozycjonowania serw. Impuls o zmiennej szerokości steruje w tym przypadku średnią dostarczaną mocą, co pozwala na precyzyjne i energetycznie efektywne sterowanie mechaniczne. O ile jest to technika odporna na zakłócenia i pozwalająca na cyfrowe sterowanie analogowymi komponentami, o tyle nie pełni funkcji pełnoprawnego protokołu komunikacyjnego a w konsekwencji nie można za pomocą tego sposobu przesyłać parametrów, telemetrii czy odpowiedzi zwrotnych. Wobec specyfiki projektu, jasne staje się, że w komunikacji pomiędzy platformą Arduino a kontrolerem lotu drona, znacznie lepszym rozwiązaniem jest UART w połączeniu z MSP.

2.4. Protokół MSP

Spośród podanych powyżej sposobów komunikacji sprzętowej, po krótkiej analizie warstwy sprzętowej wybranego do projektu kontrolera lotu, zdecydowano się wybrać interfejs UART jako najbardziej pasujący i najprostszy w implementacji. Z dostępnych protokołów obsługiwanych przez UART oraz kompatybilnych z platformą Arduino, wybrano MSP jako domyślny protokół dla najpowszechniejszego stosowanego oprogramowania kontrolerów lotu – Betaflight.

MSP (*MultiWii Serial Protocol*) to binarny protokół komunikacyjny wywodzący się z projektu MultiWii, obecnie szeroko wykorzystywany w kontrolerach lotu opartych na systemach Betaflight oraz iNav, wspierany także w ArduPilot. Jest to protokół przenoszony poprzez UART, ale może również być użyty za pośrednictwem interfejsu USB^[4]. Struktura protokołu opiera się na ramkach binarnych, które składają się z nagłówka, identyfikatora komendy, ładunku danych oraz sumy kontrolnej. MSP obsługuje zarówno komendy odczytu, jak i komendy zapisu. Istotną zaletą opisywanego protokołu komunikacji jest elastyczność w definiowaniu i rozszerzaniu zestawu ramek, co pozwala na personalizację komunikacji i dostosowanie jej do potrzeb danego systemu. Dzięki temu, MSP jest znacznie bardziej uniwersalny od prostych metod takich jak PWM oraz mniej ograniczony sprzętowo niż I²C. W kontekście niniejszego projektu umożliwia to nie tylko przesyłanie komend sterujących z Arduino do kontrolera lotu, ale również odbieranie danych telemetrycznych i diagnostycznych, co jest kluczowe w przypadku opracowywanego systemu opartego na platformie Arduino, niewspieranego natywnie przez oprogramowanie kontrolera lotu.

3. Wybór i charakterystyka komponentów

3.1. Wybór i charakterystyka komponentów jednostki latającej - drona

W celu zaimplementowania systemu omijania przeszkód, w niniejszej pracy posłużono się własnoręcznym od podstaw modelem drona FPV. Zdecydowano o takim podejściu – wybierania komponentów osobno – do niniejszego tematu, z uwagi na możliwości jakie daje, a także z uwagi na ramy budżetowe. Warto w tym miejscu nadmienić, że takie podejście wymaga również własnoręcznego dobrania dwóch pozostałych niezbędnych do lotu elementów, jakimi są radio oraz gogle FPV. Kupno gotowego drona FPV również jest możliwe, jednakże wiąże się najczęściej z ograniczeniami (brak możliwości wyboru oprogramowania, brak możliwości zamówienia części zamiennych na wypadek kraksy etc.).

3.1.1. Kontroler lotu

Jako główny element całości układu, wybrano kontroler lotu – z ang. *FC (Flight Controller)* – *Skystars F7 22 HD Pro 4* wraz z podłączonym już w zestawie kontrolerem silników typu 4 w 1 – z ang. *ESC (Electronic Speed Controller)* – *Skystars KO 60A BL32*. Całość została zakupiona jako tzw. „*Stack*”. Zdecydowano się na taki zakup, z uwagi na możliwości które daje obecny w środku układu kontrolera lotu procesor F7 – STM32F722RG6. Jest to powszechnie wykorzystywany w kontrolerach lotu procesor – Matek F722-HD, Matek F722-Mini SE, SpeedyBee F7 V2, Foxeer F722 V3 mini – charakteryzujący się pamięcią flash 512KB, rdzeniem Arm Cortex-M7 z FPU, zasilaniem 1,7-3,6V oraz pracą w zakresie temperatur od -40 do 85°C. Przeglądając pozostałe do wyboru liczne warianty kontrolerów lotu, najczęściej obecnych w zestawie wraz z kontrolerami silników – przykładowo: F4, H7 – zdecydowano, że kontroler lotu firmy Skystars z procesorem F7 będzie najdogodniejszą opcją na potrzeby projektowe, porównując zakres możliwości do ceny, albowiem kontrolery lotu H7 są najdroższymi możliwymi do wyboru kontrolerami (jednocześnie z największymi możliwościami), natomiast F4 są opcjami budżetowymi.



Rysunek 1 - Wybrany kontroler lotu (*Skystars F7 22 HD Pro 4*) wraz z kontrolerem silników połączone w stack
Źródło: oficjalna strona producenta^[5]

3.1.2. Rama – główny element nośny

Po wyborze kontrolera lotu, zdecydowano o rozmiarze i przeznaczeniu całości projektu dobierając ramę drona, bowiem to właśnie ten element decyduje o tym, do czego użytkownik będzie wykorzystywał cały, złożony już układ. W ten sposób, wyróżnić można co najmniej trzy kategorie latania dronem i są to kolejno: *freestyle*, *racing* oraz *long-range*, mówiąc o dronach składanych od podstaw. Egzemplarze gotowe – przykładowo DJI Avata 2, iFlight Nazgul Evoque F6X V2 HD 6S – łączą w sobie częściowo wszystkie z wyżej wymienionych stylów latania, gdyż przeznaczone są do szerszego grona odbiorców. I tak przechodząc kolejno pomiędzy stylami latania oraz przeznaczenia dronów FPV, drony z kategorii *freestyle* charakteryzują się średnimi rozmiarami – jest to najczęściej 5'' rama – napędem w zakresie 6S ~ 1700-2000 KV lub 4S ~ 2300-2500 KV, umiarkowaną masą oraz stosunkowo krótkim czasem lotu w zakresie 3-6 minut z kamerą. Loty oraz drony *freestyle* nastawione są przede wszystkim na wykonywanie manewrów oraz kreatywne latacie z płynnymi ujęciami z kamery akcji zamontowanej na dronie. Loty oraz drony typu *racing*, są, jak nazwa wskazuje, nastawione na prędkość i responsywność, wobec czego często niejednokrotnie są budowane na mniejszych niż 5'' ramy, co przepłacają jeszcze krótszym niż drony typu *freestyle* czasem lotu, oraz zmniejszoną trwałość. Budując drona do wyścigów, użytkownik powinien starać się maksymalnie obniżyć masę całości projektu. Drony z tej kategorii rozwijają prędkości rzędu 120-200+ $\frac{km}{h}$. Ostatnią z głównych „szkół” dronów FPV, jest kategoria *long-range*. W tym przypadku, najczęściej stosuje się większe – 7'' ramy oraz większe zasilacze. Napęd jednostek *long-range* mieści się w zakresie 6S ~ 1200-1500 KV. Drony oraz loty *long-range* są nastawione na długość oraz na długie, stabilne ujęcia z kamery, a także na bezpieczeństwo. W przypadku opisywanej konfiguracji, zdecydowano o przeznaczeniu *freestyle* - *long-range* będącym połączeniem dwóch głównych nurtów jako optymalnej, z uwagi na założenia projektu. Kierując się ww. wyborem przeznaczenia drona, zdecydowano o zakupie ramy GEPRC Mark4 – 7''. Rama Mark4 charakteryzuje się możliwością zmieszczenia na niej odpowiednio wybranego wcześniej zestawu kontrolera lotu oraz kontrolera silników, możliwością zamontowania silników odpowiednich dla 7'' śmigieł oraz wagą wynoszącą 121 gramów. Całość ramy wykonana jest z włókna węglowego, a jej ramiona mają grubość 5mm, co daje odpowiednią sztywność oraz niską wagę.



Rysunek 2 - Wybrana rama GEPRC Mark 4 – 7''
Źródło: oficjalna strona producenta^[6]

Warto w tym miejscu zaznaczyć, że już na tym etapie, częściowo dokonano wyboru rodzaju nadajnika wideo, gdyż wybrana rama jest szczególnie dobra dla analogowych odpowiedników. Wobec powyższego, następnymi elementami, które niezbędne są do obsługi drona FPV a które wybrano, to silniki, kamera oraz nadajnik wideo.

3.1.3. Silniki bezszczotkowe

Jako że silniki w głównej mierze decydują o przeznaczeniu drona, chcąc połączyć koncepcję latania *freestyle* oraz *long-range* zdecydowano o zakupie silników GEPRC SpeedX2 2407E. Są to silniki typu 2407, generujące 1750 obrotów na minutę na 1V bez obciążenia. Oznaczenie 2407 należy czytać tak, że stojan silnika ma 24mm średnicy i wysokość 7mm. Podsumowując wybór silników, są to motory duże, nie grzejące się przesadnie przy stałym ciągu (w tym wypadku należałyby ograniczyć limit „gazu” w oprogramowaniu na kontrolerze lotu – co też zostanie uczynione w późniejszym etapie).



Rysunek 3 - Wybrany silnik GEPRC SpeedX2 2407 1750 KV
Źródło: oficjalna strona producenta^[7]

3.1.4. Odbiornik RC

W tym momencie wybierania komponentów, zdecydowano o wyborze odbiornika RC. Jest to jeden z głównych elementów całości zestawu, z uwagi na fakt, iż bez niego kontrolowanie drona za pomocą radia nie jest możliwe. W tym przypadku, posłużono się akceptowaną i również

powszechnie stosowaną w środowisku FPV marką Radiomaster. Wybranym odbiornikiem został Radiomaster RP3 ELRS Nano. Jest to odbiornik posiadający dwie anteny, wobec czego umożliwiający większe pokrycie pasm w trakcie lotu (jedna antena ułożona pionowo, druga poziomo), wykorzystujący interfejs CRSF a co za tym idzie pasujący do możliwości kontrolera lotu oraz m. in. oscylator temperaturowo kompensowany (TCXO) poprawiający stabilność częstotliwości przy zmianach temperatury. Niniejszy odbiornik zawiera również LNA – *Low Noise Amplifier* – celem zwiększenia sygnału odbierania.^[8] Z uwagi na obostrzenia związane z członkostwem Polski w Unii Europejskiej, koniecznym okazał się wybór odbiornika w wersji LBT – *Listen Before Talk*. O różnicach pomiędzy poszczególnymi rodzajami odbiorników oraz nadajników opowiedziane zostanie w dalszym punkcie opisującym wybór nadajnika – radia.



Rysunek 4 - Wybrany odbiornik Radiomaster RP3 V2 ELRS 2.4GHz
Źródło: oficjalna strona producenta^[8]

3.1.5. Kamera FPV

Kolejnym elementem dobranym do zestawu niniejszej pracy była kamera. W tym przypadku, wybór stanął na Caddx Ratel 2 z uwagi na przystępna cenę oraz pozytywne w większości opinie o produkcie. Kamera ta bowiem jest powszechnie stosowana w środowisku FPV, z uwagi na bardzo niską masę oraz dobrą rozdzielczość dla zestawów analogowych, do których należy niniejszy zestaw.



Rysunek 5 - Wybrana kamera FPV Caddx Ratel 2
Źródło: oficjalna strona producenta^[9]

Wybierając kamerę FPV, od razu rozważono odpowiedni nadajnik wideo – z ang. *VTX (Video Transmitter)* – gdyż elementy te są ze sobą bezpośrednio powiązane. Mając do wyboru dwie

technologię: analogową oraz cyfrową, zdecydowano się na moduł bardziej przystępny cenowo, jednakże przepłacający jakością obrazu wobec swojego cyfrowego odpowiednika – nadajnik analogowy.

3.1.6. Nadajnik video oraz antena

Zadaniem nadajnika wideo, jest transmitowanie obrazu z kamery do gogli tak, aby pilot miał podgląd na to co dzieje się przed/pod dronem – jest to zależne od ustawienia kamery. Mając na uwadze istotę ww. komponentu drona, wybrano nadajnik marki RushFPV – Rush Tank Max Solo VTX 2.5 W. Wyżej wymieniony nadajnik, charakteryzuje się nadawaniem obrazu na pasmie 5.8 GHz, możliwością ręcznego konfigurowania poziomów mocy za pomocą przycisku (są to kolejno: PIT/25mW/500mW/1000mW/MAX – do 2.5 W w zależności od zapewnionego chłodzenia), wejściem wideo w formacie CVBS PAL/NTSC oraz sterowaniem za pomocą SmartAudio. Dodatkowo, z uwagi na możliwości nadajnika, jest on zaopatrzony w zestawie w radiator oraz wentylator, które dodatkowo chłodzą układ, niezależnie od warunków pogodowych. W konfiguracji projektowej, został on umieszczony z tyłu ramy, dla ułatwienia montażu anteny – bez której *notabene* nie byłoby możliwe przesłanie obrazu do gogli – nadajnik by się zwyczajnie spalił.

Mając na uwadze powyższe oraz fakt, iż wyjście antenowe w wybranym modelu VTX to MMCX, antenę dobrano według identycznego producenta – RushFPV Cherry II MMCX RHCP. Zdecydowano się na polaryzację anteny RHCP (Right-Hand Circular Polarization) i jest to element, który identycznie musi zgadzać się z odbiornikiem w goglach. W przeciwnym wypadku – jeżeli nadajnik będzie RHCP, a odbiornik LHCP – mamy do czynienia z ogromnym spadkiem sygnału – rzędu 20-30dB, wobec czego sygnał praktycznie znika. Daje to również możliwość wybrania polaryzacji podczas latania w grupie – bowiem jeżeli jedna osoba posiada zestaw RHCP, druga powinna wybrać przeciwnie – LHCP, aby uniknąć wzajemnych zakłóceń^[10]. Rysunek 6 prezentuje wybrany nadajnik obrazu – VTX, rysunek 7 natomiast prezentuje wybraną do niego antenę.



Rysunek 6 - Wybrany nadajnik wideo - RushFPV Rush Tank Max Solo
Źródło: oficjalna strona producenta^[11]



Rysunek 7 - Wybrana antena do nadajnika wideo - RushFPV Cherry II MMCX
Źródło: oficjalna strona producenta^[12]

3.1.7. Bateria

Kolejnym elementem, który należało dopasować do istniejącej już konfiguracji, była bateria. W przypadku tego elementu zestawu, istnieje wiele możliwości: od baterii 1S do 6S. Podział baterii charakteryzuje liczba ogniw Li-Po (litowo-polimerowe) lub Li-ion (litowo-jonowe) połączonych ze sobą szeregowo. Z uwagi na fakt, iż nadajnik wideo opisany powyżej nie daje możliwości wyboru innego niż 6S rodzaju baterii, zdecydowano się właśnie na zakup pakietu z tej kategorii. Nie chcąc przesadnie przeciągać finalnie złożonego drona oraz mając na uwadze charakter projektu, zdecydowano się na zakup baterii Tattu Funfly 1550 mAh 6S 100C. Jest to pakiet litowo-polimerowy, małej wielkości, małej pojemności a co za tym idzie – niskiej wagi. Podjęto decyzję, że na potrzeby nauki latania dronem z wykorzystaniem systemu omijania przeszkód, będzie to wystarczający pakiet. Napięcie nominalne ww. baterii to 22.2V, jest ona zaopatrzona w balancer – przeznaczony do bezpiecznego, zbalansowanego ładowania – a jej waga wynosi zaledwie 260 gramów. Rysunek 8 prezentuje wybrany model zasilacza typu 6S.



Rysunek 8 - Wybrana bateria Tattu Funfly 1550 mAh 6S 100C
Źródło: strona z częściami do dronów FPV – rotorama.com^[13]

3.1.8. GPS

Do całości zaprojektowanego drona dobrano również GPS – celem pomiaru prędkości podczas lotu, możliwości śledzenia trasy danego przelotu a także na wypadek zgubienia drona w przypadku awarii jakiegokolwiek z komponentów na pokładzie. W tym przypadku, komponentem

dobranym do projektu był iFlight Blitz M10 GPS V2 z kompasem. Jest to moduł wyposażony w dość dużą jak na tak małe urządzenie antenę – 25x25mm – oraz filtr LNA (z ang. Low Noise Amplifier) – którego zadaniem jest wzmacnianie bardzo słabego sygnału satelitarnego bez dodawania zbędnego szumu. Dobrany do zestawy GPS posługuje się powszechnym protokołem UBLOX z częstotliwością odświeżania wynoszącą 10Hz. Rysunek 9 prezentuje wybrany model GPS.



*Rysunek 9 - Wybrany GPS iFlight Blitz M10 V2 z kompasem
Źródło: oficjalny sklep producenta^[14]*

3.1.9. Śmigła

W tym momencie konstruowania zestawu latającego, dobrano jako ostatnie śmigła oraz zapasowe przewody. Śmigła dobrano według parametrów silników oraz rozmiaru jednostki i zdecydowano, że wyborem odpowiednim będą śmigła Gemfan 7040-3 wykonane z poliwęglanu. Jest to materiał wykorzystywany powszechnie do wykonania śmigieł z uwagi na wysoką udarność i dobrą sztywność. Śmigła w całości dobieranego zestawu mają pomijalną cenę wobec powyższego tym kryterium nie sugerowano się podczas zakupu. Śmigła dobrano według schematu działania silników – 2 śmigła lewoskrętne oraz 2 prawoskrętne. Rysunek 10 prezentuje wybrany model śmigieł.



*Rysunek 10 - Wybrane śmigła Gemfan 7040-3
Źródło: sklep z częściami do dronów rcmaniak.pl^[15]*

3.1.10. Elementy dodatkowe

Dodatkowo, w ramach realizowanego projektu niezbędnym zakupem okazał się brzeczyk. Jest to o tyle ważny element w budowie drona FPV, o ile pilot planuje dokonywać lotów w terenach z gęstymi zarościami, gdyż umożliwia łatwe znalezienie drona w przypadku awarii lub przewrócenia się podczas nauki latania. W trakcie wyboru pomiędzy dwoma wariantami brzeczyków – z baterią lub bez baterii – kierowano się ceną oraz założeniem, że pilot przystępujący do nauki latania dronem według niniejszej pracy, będzie posiadał ograniczenia drona w postaci limitów ustalanych w oprogramowaniu. Wobec powyższego, zakupiony został model brzeczyka bez baterii z przewodem JST pobierającym 5V zasilania – czyli dokładnie tyle, ile jest w stanie zapewnić kontroler lotu w dedykowanym dla brzeczyka miejscu na płycie.

Po otrzymaniu wszystkich wyżej wymienionych komponentów, umieszczono je w wybranej ramie, a następnie przylutowano do kontrolera lotu te, które wymagały takiego sposobu instalacji – były to: kamera, nadajnik wideo, GPS, odbiornik RC, brzeczyk oraz silniki. Dodatkowo, przylutowano również złącze zasilania z baterii, a razem z nim kondensator, zgodnie z zalecaniami producenta kontrolera lotu. W ostatnim kroku, w celu perspektywicznego podłączenia Arduino do kontrolera lotu, przylutowano cztery przewody umożliwiające połączenie kontrolera lotu z płytą stykową.

3.2. Wybór i charakterystyka kontrolera RC – radia

Mając zaprojektowaną jednostkę latającą, w kolejnym kroku wybrano radio – element niezbędny do kontrolowania drona z ziemi. Jako, że odbiornik zamówiony został w technologii ExpressLRS oraz wersji na region Listen Before Talk (LBT), wymogiem koniecznym było wybranie odpowiadającego mu nadajnika. Postanowiono wybrać model również odpowiadającej firmy Radiomaster – Zorro.

ExpressLRS (ELRS) jest protokołem typu open-source łączą radioowego, powstały z połączenia projektów ExpressLRS i mLRS. Zapewnia on minimalne opóźnienia i bardzo dużą czułość odbiornika.^[16] Listen Before Talk (LBT) natomiast, odnośnie się bezpośrednio do sposoby komunikacji pomiędzy odbiornikiem a nadajnikiem – w procedurze LBT, zanim sygnał zostanie wyemitowany, nadajnik sprawdza czy kanał jest wolny. Chroni to innych użytkowników pasma ISM^[17]. Jest to sposób komunikacji wymagany w krajach członkowskich Unii Europejskiej przy pracy w narzędziach/urządzeniach pracujących na częstotliwości 2,4GHz.

Zdecydowano się na nadajnik Radiomaster Zorro, mając na względzie również inne nadajniki RC, głównie z uwagi na kształt urządzenia oraz jego cenę - na rysunku 11 pokazano wygląd nadajnika w jego rzucie z góry. Jak już wspomniano, jest to model wyposażony w protokół komunikacji ELRS, ekran LCD, częstotliwość pracy 2,4GHz oraz m.in. możliwość ładowania baterii zasilających kontroler przez USB-C, czy chociażby bardzo przydatną funkcję tworzenia własnych modeli (profili) wewnętrz urządzenia.^[18] O tym jak skonfigurować nadajnik oraz stworzyć dostosowany do własnych potrzeb profil, opowiedziano w rozdziale [4.2. Charakterystyka i konfiguracja oprogramowania kontrolera RC niniejszego opracowania.](#)



Rysunek 11 - Wybrany nadajnik Radiomaster Zorro w rzucie z góry
 Źródło: Oficjalna strona Radiomaster^[18]

[3.3. Wybór i charakterystyka gogli FPV](#)

Po wybraniu odpowiedniego nadajnika RC (kontrolera), ostatnim elementem niezbędnym do kontroli drona FPV są gogle FPV. W tym przypadku, pod uwagę należało wziąć typ zamówionego nadajnika wideo który znajdować się będzie docelowo na dronie – VTX. W niniejszym przykładzie, posłużyono się analogowym nadajnikiem marki RushFPV, wobec czego gogle należało zakupić odpowiednie dla ww. technologii. Gogle, są w całości zestawu najdroższym elementem, jednakże można je wykorzystywać w dowolnej ilości dronów, dopóki obsługiwane technologie nadawania i odbierania obrazu z kamery się pokrywają (analogowo lub cyfrowo). Wobec powyższego, zdecydowano o kupnie produktu marki Skyzone – modelu SKY04O Pro. Są to gogle przeznaczone *stricto* do analogowych nadajników obrazu, posiadają parę wyjść obrazu typu OLED oraz oferują m.in. dobrą, jak na analogowe odbiorniki obrazu, rozdzielcość rzędu 1280x720 pikseli. Dodatkowo, wybrane gogle mogą być zasilane baterią od typu 2S do 6S, wobec czego wraz z nimi, została również zamówiona bateria typu 4S, co jednakże nie jest niezbędne, gdyż zostały one również wyposażone

w wejście zasilające typu USB-C, którego drugi koniec można dla przykładu wpiąć do urządzenia typu powerbank i rezultat będzie jednakowy. Gogle posiadają również wejście HDMI, wobec czego mogą zostać wykorzystane chociażby w symulatorze latania uruchamianym na komputerze PC, przed dokonaniem faktycznych lotów w terenie. Jest to pomocna praktyka, z uwagi na przyzwyczajanie pilota do tego, co będzie miało miejsce w rzeczywistości i z pewnością wpływa na kształtowanie umiejętności pilota oraz zmniejsza ryzyko późniejszych błędów. Z uwagi na fakt, iż gogle posiadają DVR – z ang. *Digital Video Recorder*, urządzenie do nagrywania obrazu – obsługują również karty SD do pojemności 128 GB^[19]. Karta SD o maksymalnej pojemności została zakupiona, celem możliwości nagrania, a następnie odtworzenia materiałów testowych w niniejszym projekcie. Rysunek 12 prezentuje wybrany model gogli FPV.

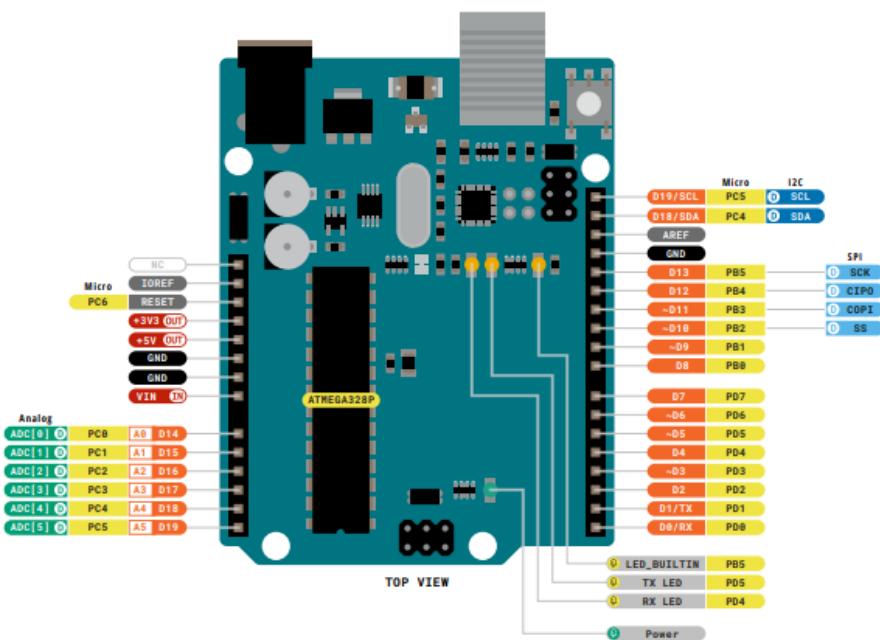


Rysunek 12 - Wybrany model gogli FPV - Skyzone 04O Pro
Źródło: Oficjalna strona producenta^[19]

3.4. Wybór i charakterystyka platformy Arduino

Mając na uwadze charakter projektu – wykrywanie przeszkód podczas lotu dla początkujących pilotów – zdecydowano się na wybór najpowszechniejszego wśród również początkujących elektroników modelu Arduino, będącego jednocześnie najlepiej udokumentowaną i stosunkowo tanią wersją ww. platformy – *Arduino Uno* w wersji *Rev 3*. Jest to model umożliwiający wykonanie systemu omijania przeszkód w wersji podstawowej – wyzwalającej sygnał dźwiękowy za pomocą *beepera* na dronie po zebraniu danych z czujnika ultradźwiękowego, znajdującego się na płytce stykowej podłączonej do platformy, oraz wypisującej pojedynczy znak na ekranie gogli FPV. Arduino Uno R3, zgodnie z tym co mówi oficjalna dokumentacja – jest to mikrokontroler wyposażony w procesor ATmega328P, 32 kB pamięci flash, 2 kB pamięci RAM oraz m.in. 14 cyfrowych wejść/wyjść – z których 6 może zostać użytych do sterowania sygnałem PWM – czy chociażby port USB umożliwiający łatwe programowanie, w zasadzie od momentu zakupu płytki^[20].

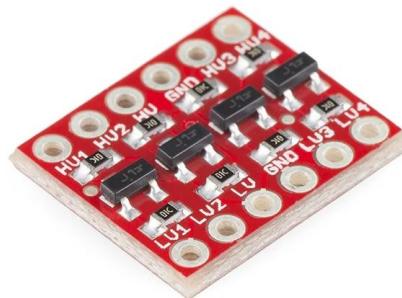
Zdecydowano również o ww. modelu z uwagi na łatwość wymiany procesora, na wypadek jego spalenia w trakcie testów połączenia pomiędzy nim, a kontrolerem lotu. Arduino Uno R3 obsługuje również ogromną ilość czujników – na stronie botland.com.pl jest ich przykładowo ponad 130 – w tym czujniki takie jak: ultradźwiękowy czujnik odległości HC-SR04, czujnik ruchu PIR HC-SR501, czujniki temperatury i wilgotności DHT11 i inne. Całościowy Pinout (wejścia/wyjścia) opisywanego modelu platformy Arduino zaprezentowano na rysunku 13. Dodatkowym atutem wybranego producenta/modelu platformy mikrokontrolera, jest jego uprzednia znajomość z toku studiów w ramach zajęć laboratoryjnych oraz interfejs programowania który oparty jest na poznanym w trakcie studiów języku programowania C.



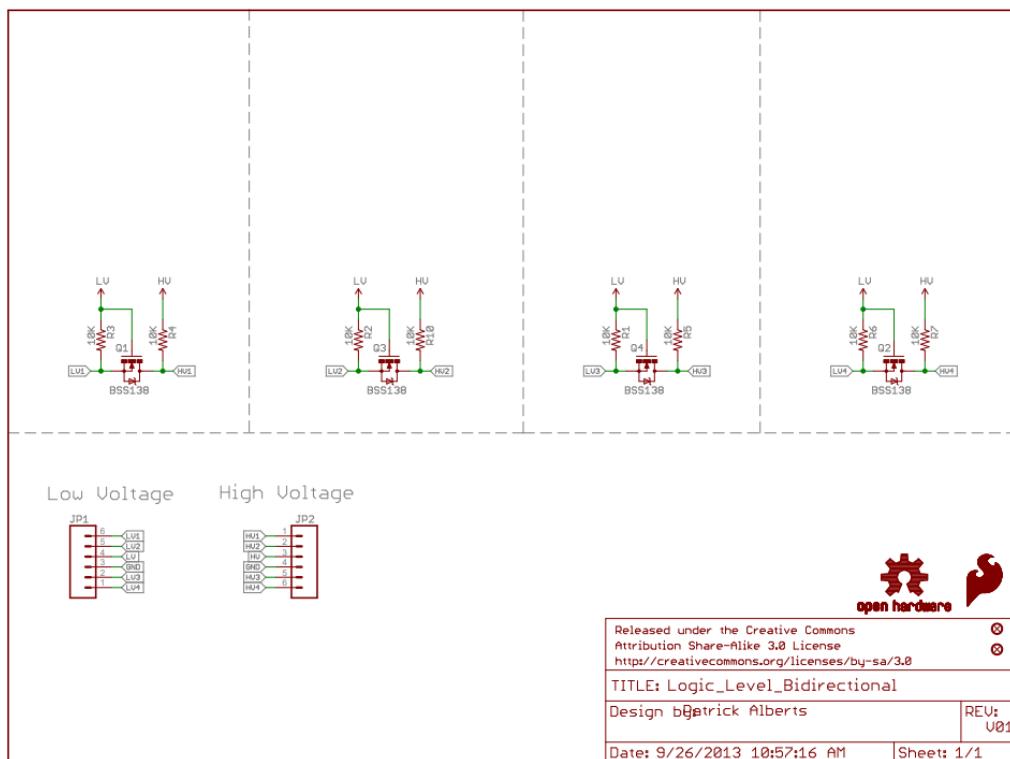
Rysunek 13 - Całościowy model wejść/wyjść Arduino Uno R3
Źródło: Oficjalna dokumentacja Arduino^[21]

Dodatkowo, poza zakupionym i wybranym modelem platformy Arduino, świadomie zakupiono konwerter poziomów logicznych SparkFun BOB-12009. Jest to element niezbędny w opisywanej konfiguracji, gdyż charakterystyka wybranego modelu Arduino pokazuje, że pracuje on domyślnie na poziomie 5V. Niestety, chcąc podłączyć jakikolwiek przewód przesyłowy pomiędzy mikrokontrolerem a kontrolerem lotu drona pracującym domyślnie na poziomie 3.3V, użytkownik w innej sytuacji ryzykowałby spaleniem danego portu szeregowego na kontrolerze, lub w gorszym przypadku spaleniem całości kontrolera. Wobec powyższego, wszelkie połączenia muszą odbywać się za pośrednictwem konwertera poziomów logicznych. Wybrany konwerter, jest modułem

dwukierunkowym, posiadającym 4 kanały przesyłowe. Umożliwia komunikację pomiędzy urządzeniami z poziomami napięć 5V, 3.3V, 2,8V oraz 1,8V. Na rysunku 14 zaprezentowano wygląd konwertera, natomiast na rysunku 15 pokazano jego schemat.



Rysunek 14 - Zdjęcie konwertera poziomów logicznych marki SparkFun modelu BOB-12009
Źródło: Oficjalna strona producenta^[22]



Rysunek 15 - Schemat konwertera poziomów logicznych marki Sparkfun modelu BOB-12009
Źródło: Oficjalna strona producenta^[23]

3.5. Wybór i charakterystyka czujników ultradźwiękowych

Czujnikiem ultradźwiękowym wykorzystanym w niniejszym projekcie, jest czujnik ultradźwiękowy HC-SR04 marki justPi, będący najpowszechniej dostępnym czujnikiem wykorzystującym technologię ultradźwięków dla platformy Arduino. Jest to moduł, który pracuje z częstotliwością 40Hz, natomiast jego zakres pomiarowy wynosi od 2 do 400 cm zasięgu. Czujnik w celu poprawnej pracy, zasilany jest napięciem 5V, czyli domyślnym poziomem logicznym dla mikrokontrolera Arduino Uno R3. Jest to również czujnik kompatybilny z mikrokontrolerem Raspberry Pi. W celu rozpoczęcia pomiaru ww. czujnikiem, konieczne jest podłączenie go do Arduino lub płytki stykowej - na której również bazować będzie niniejszy projekt – oraz podanie mu na pin o nazwie TRIG impulsu napięciowego przez $10\mu s$. Wówczas, impuls zostaje wysłany z jednej strony czujnika (nadajnik), a po odbiciu od obiektu wraca do drugiej (odbiornika). Wynikiem takiej operacji, jest podawany przez czujnik stan wysoki proporcjonalny czasem do mierzonej odległości. Najprościej z tego schematu działania czujnika obliczyć odległość od przeszkody w sposób następujący:

$$V = \frac{s}{t} \left[\frac{m}{s} \right] \rightarrow s = Vt [m]$$

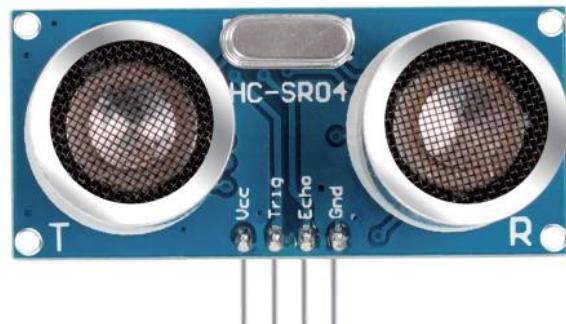
gdzie v = prędkość rozchodzenia się fali dźwiękowej w powietrzu – $340 \frac{m}{s}$, więc dla opisywanego przykładu, w którym $s = 1 \text{ cm}$, $V = 340 \frac{m}{s}$ i $t = 1 \mu s$ otrzymano:

$$s = \frac{t \cdot 34}{1000} \cdot \frac{1}{2} [cm]$$

Jednakże, w praktyce, z uwagi na łatwość zarządzania bibliotekami do czujników podłączanych do Arduino, korzysta się z nich. Jest to rozwiązanie łatwiejsze, eliminujące w większości przypadków popełnienie błędów i również w niniejszym projekcie zastosowano takie podejście – wykorzystano bibliotekę *NewPing* oraz jej podstawowy element - *sonar*.

Czujniki ultradźwiękowe są jednym z dostępnych sposobów pomiaru odległości od obiektu, dostępnych dla platformy Arduino. Do reszty zaliczyć można: cyfrowe czujniki odległości, analogowe czujniki odległości, laserowe czujniki odległości (ToF) oraz chociażby skanery LiDAR. Czujniki ultradźwiękowe zostały wybrane do niniejszego projektu jako opcja budżetowa nie tylko z uwagi na swoją cenę i łatwość wymiany w razie awarii, ale również z uwagi na dostępność i łatwość w przygotowaniu systemu. Nie rozpatrywano w przypadku rozważania czujnika kompatybilności

z kontrolerem lotu, gdyż całość logiki przetwarzania odebranego z czujnika sygnału miała leżeć po stronie Arduino.



Rysunek 16 - Wybrany czujnik ultradźwiękowy HC-SR04
Źródło: Nieoficjalna dokumentacja Sunfounder^[24]

4. Wybór, charakterystyka oraz konfiguracja oprogramowania

4.1. Wybór, charakterystyka i konfiguracja oprogramowania jednostki latającej - drona

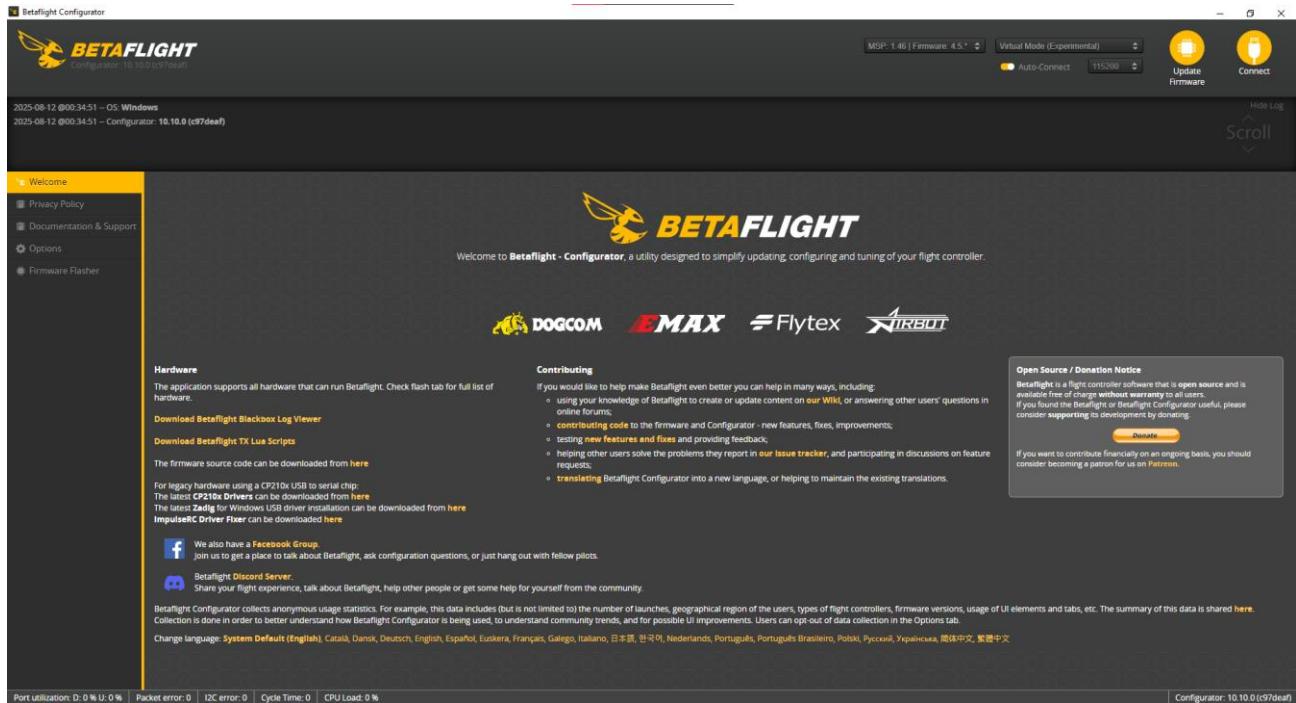
Podobnie jak w przypadku tradycyjnych komputerów PC, tak w przypadku oprogramowania dla kontrolera lotu, należy wiedzieć, iż istnieje wiele – przynajmniej 3 popularne – systemów, w których może pracować pilot, jak chodzi o kontroler lotu jednostki latającej (drona). Dokonując wyboru spośród trzech systemów z największą ilością użytkowników, wzięto pod uwagę kilka czynników takich jak:

- 1) intuicyjność systemu, a co za tym idzie łatwość jego obsługi,
- 2) obciążenie dla procesora, a co za tym idzie „wielkość” systemu w kontekście zajmowanej pamięci,
- 3) kompatybilność z zamysłem projektu.

Zakupiony model kontrolera lotu, tak jak ma to miejsce najczęściej, został dostarczony z wgranaą wersją oprogramowania Betaflight, aktualną na moment wyprodukowania kontrolera, tak więc z przestarzałą względem rozwoju oprogramowania. Postanowiono nie zmieniać oprogramowania na inne z pozostałych dwóch – iNav lub ArduPilot – z uwagi na ww. czynniki. Oprogramowanie Betaflight jest bowiem dobrze udokumentowane i w zakresie, w którym zamierzono, jest możliwe wykonanie na nim systemu omijania przeszkód informującym użytkownika o przeszkodach. Jest to również oprogramowanie, które oferując dużą liczbę możliwości dla pilota, zajmuje stosunkowo mało miejsca wewnątrz pamięci kontrolera, a co za tym idzie, umożliwia jego płynne funkcjonowanie. Systemy iNav oraz ArduPilot mają co prawda większe możliwości w kwestii instalacji niestandardowych rozwiązań – do których zalicza się chociażby podłączenie Arduino do kontrolera lotu czy wykorzystanie sensorów – natomiast są przystosowane do innych, dedykowanych kontrolerów. W kwestii oprogramowania ArduPilot, warto nadmienić, iż jego instalacja w pełnym zakresie na kontrolerze lotu jest możliwa dla wysokobudżetowych jednostek z chipem H7 – dla jednostek posiadających chip F7, jego instalacja jest możliwa jedynie w bardzo ograniczonym zakresie z uwagi na małą ilość pamięci na płytce.

W tym miejscu, pobrano Betaflight Configurator, będący konfiguratorem oprogramowania wewnątrz płytki, z oficjalnego repozytorium producenta^[25]. Betaflight Configurator nie różni się w ogóle w kwestii interfejsu użytkownika na systemach Windows, Linux czy macOS, jednakże w niniejszej pracy rozważany został przypadek korzystania z aplikacji dla użytkowników systemu

Windows. Mając zainstalowaną aplikację konfiguratora w wersji angielskiej, otworzono ją oraz sprawdzono jej funkcjonalności. Interfejs użytkownika prezentuje rysunek 15.

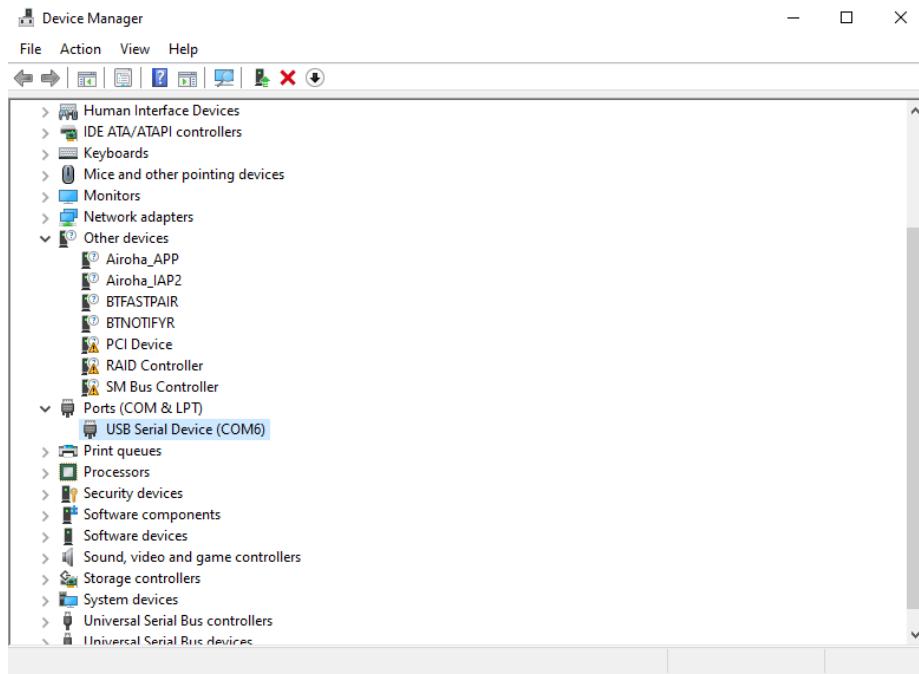


Rysunek 17 - Aplikacja Betaflight Configurator po pierwszym uruchomieniu
 Źródło: opracowanie własne

Warto w tym miejscu nadmienić, że każdorazowo konfigurowanie oprogramowania kontrolera lotu, powinno odbywać się w bardzo dobrze wentylowanym pomieszczeniu, lub przez krótki okres czasu, z uwagi na przegrzewanie się płytki i znajdujących się na niej m. in. regulatorowi liniowemu, żyroskopowi (długa ekspozycja na ciepło powoduje jego dryf), barometrowi czy chociażby lutom i złączom, co prowadzi do zmęczenia termicznego.

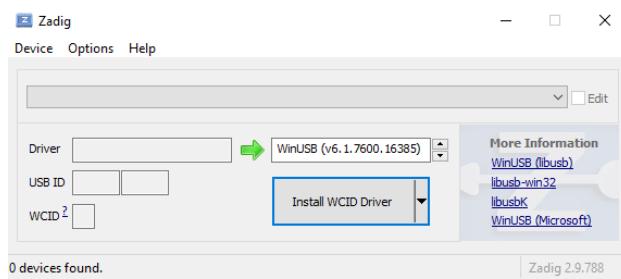
W następnej kolejności podłączono za pomocą przewodu USB-B – USB-C kontroler do komputera oraz przystąpiono do aktualizacji oprogramowania znajdującego się wewnątrz kontrolera lotu. W tym miejscu warto nadmienić, że kontroler lotu, w zależności od dostawcy, może nie posiadać sterownika portu szeregowego dla systemu Windows. Aby naprawić taki stan rzeczy i poprawnie przystąpić do aktualizacji kontrolera, konieczne jest doinstalowanie sterownika ręcznie. W tym celu, należy sprawdzić stan urządzeń podłączonych do komputera za pomocą narzędzia „Menadżer Urządzeń” („Device Manager”) oraz wykryć wśród podłączonych urządzeń o nazwie *USB Serial Device (COMx)* gdzie zamiast x widnieje cyfra odpowiadająca numerowi portu szeregowego. W pokazanym poniżej przykładzie – rys. 15 – pokazano w jaki sposób sprawdzić można poprawność

zainstalowanego sterownika – jeżeli urządzenie o wskazanej nazwie nie widnieje w menadżerze, należy zainstalować sterownik ręcznie.

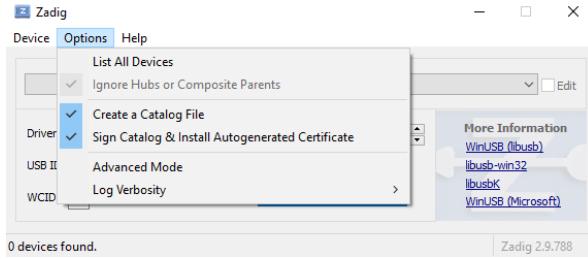


Rysunek 18 - Okno menadżera urządzeń, ukazujące poprawnie rozpoznany port szeregowy kontrolera lotu
 Źródło: opracowanie własne

W celu instalacji sterownika, najprościej posłużyć się programem Zadig – jest to oprogramowanie służące do instalacji uniwersalnych sterowników USB w celu ułatwienia dostępu do urządzeń zewnętrznych^[26]. Wówczas, po instalacji programu oraz uruchomieniu go, mając podłączony do komputera kontroler lotu, z listy „Options” należy wybrać polecenie „List All Devices” i wybrać urządzenie STM32, a następnie przyciskiem „Install WCID Driver” zainstalować sterownik na płytce.

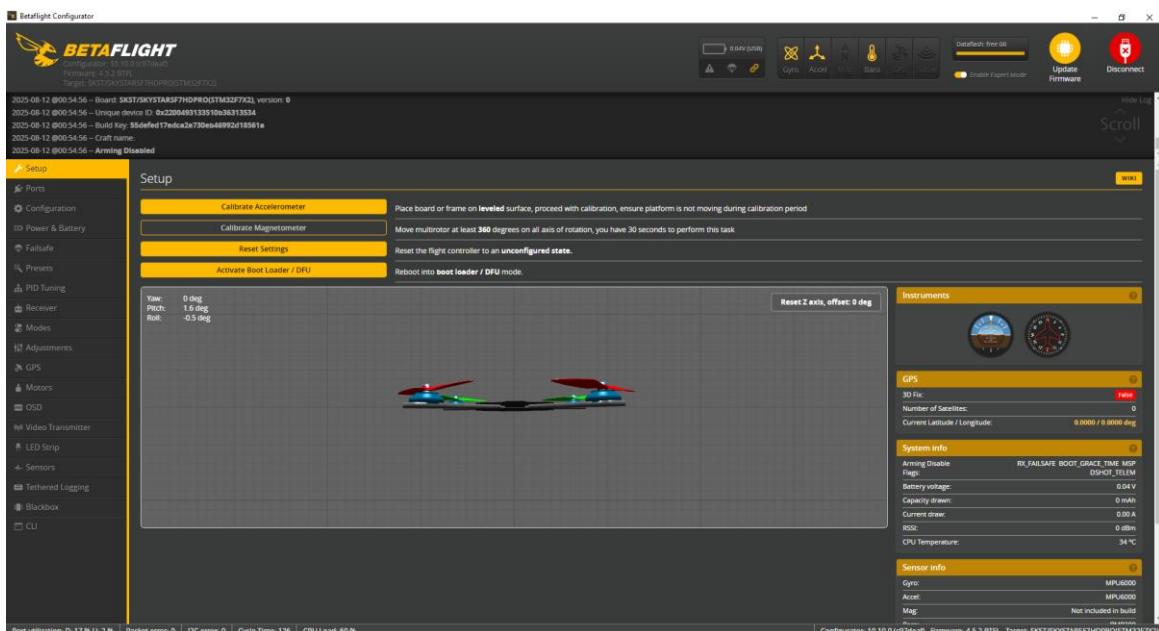


Rysunek 19 - Interfejs aplikacji Zadig służącej do ręcznej instalacji sterownika
 Źródło: opracowanie własne

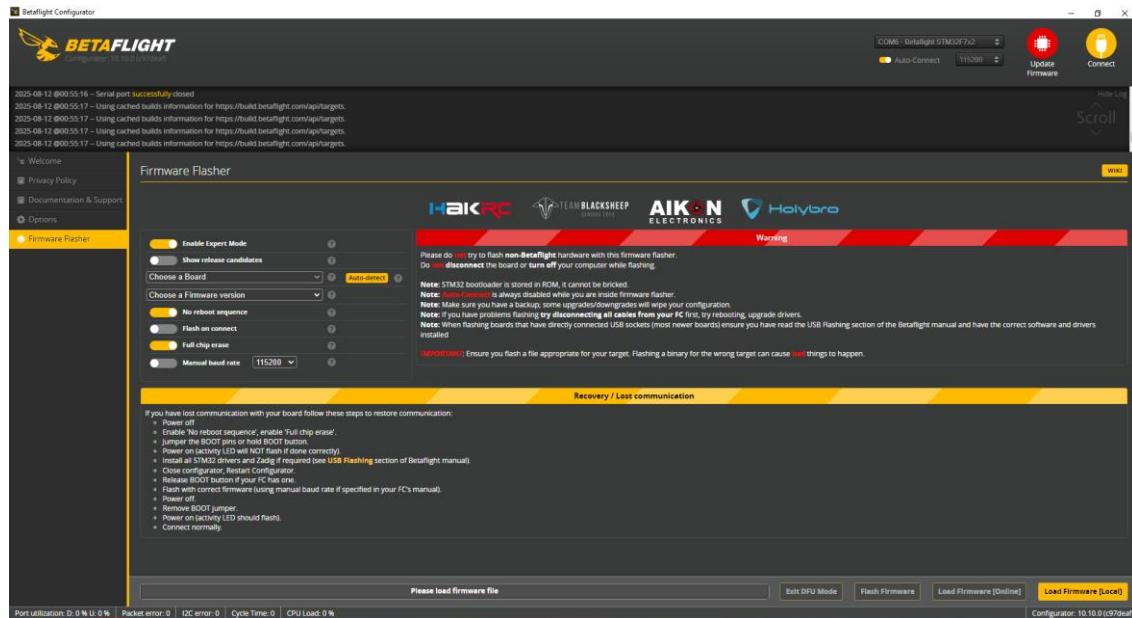


Rysunek 20 - Lista opcji dostępnych w programie Zadig z możliwością wyświetlenia wszystkich urządzeń
Źródło: opracowanie własne

W tym momencie, mając zainstalowany sterownik dla systemu Windows, można przystąpić do aktualizacji oprogramowania, gdyż znajdujące się wewnątrz świeżo zamontowanego kontrolera lotu jest najczęściej przestarzałe, a instalacja stabilnego, zaktualizowanego oprogramowania jest dobrą praktyką, nie tylko dla kontrolera lotu, ale także dla użytkownika – zaktualizowane oprogramowanie bowiem najczęściej niesie za sobą większą ilość funkcji które spełniać może system. W tym celu, wewnątrz aplikacji Betaflight Configurator, należy wybrać z menu zakładkę dostępnego po lewej stronie pozycję *Firmware Flasher*, poprzedzoną ikoną mikroukładu. Jeżeli kontroler lotu połączył się z komputerem i aplikacja Betaflight Configurator otworzyła automatycznie port szeregowy kontrolera – co prezentuje rysunek 19 – należy w oknie prezentującym konfigurację zestawu, w prawym górnym rogu aplikacji wybrać opcję „*Update Firmware*”, również z ikoną mikroukładu, aby przejść do ww. zakładki. Zamknięcie portu szeregowego oraz poprawne wejście do zakładki aktualizacji oprogramowania prezentuje rysunek 20.



Rysunek 21 - Okno aplikacji Betaflight Configurator po otwarzeniu portu szeregowego kontrolera lotu
Źródło: opracowanie własne

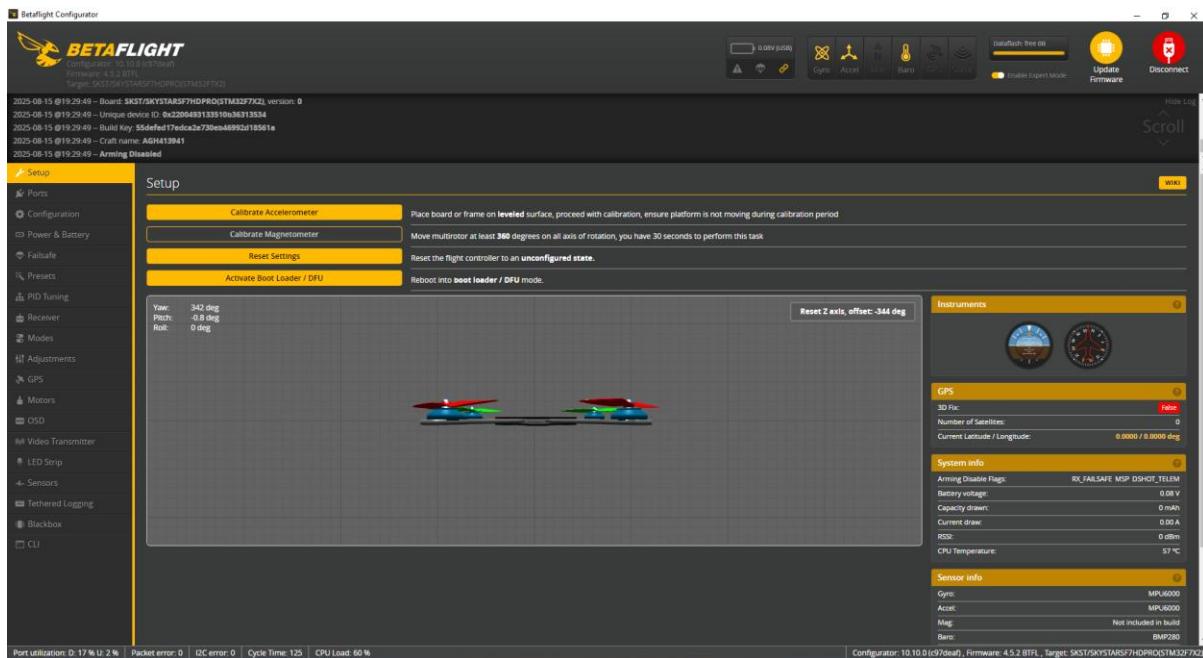


Rysunek 22 - Okno aplikacji Betaflight Configurator po poprawnym zamknięciu portu szeregowego kontrolera lotu oraz przejściu do zakładki aktualizacji oprogramowania
źródło: opracowanie własne

W tym miejscu, wpisano poprawnie dane płytki znajdującej się na jednostce latającej – z pierwszej dostępnej rozwijanej listy wybrano SKYSTARSF7HDPRO, natomiast z następnej wybrano najnowszą, **stabilną** wersję oprogramowania – 4.5.2. Wybieranie wersji systemu w trakcie rozwoju jako niezalecane pominięto, z uwagi na fakt, iż nie jest to wersja dobrze wytestowana oraz jak nazwa wskazuje, niestabilna, co prowadzić może w najgorszych przypadkach do utraty kontroli nad dronem lub jego rozbicia. W kolejnym kroku, odznaczono opcję uaktualniania „*No reboot sequence*” oraz zaznaczono „*Full chip erase*”. Pierwsza z ww. opcji, odpowiedzialna jest za brak restartowania płytki po aktualizacji, natomiast druga, odpowiada za pełne wyczyszczenie pamięci płytki. Protokół radia jaki wybrano poniżej tych opcji to CRSF, ponieważ takim właśnie protokołem posługuje się wybrany nadajnik RC, natomiast z opcji dodatkowych wybrano opcję „*LED Strip*”, zostawiając domyślne wartości „*Acro Trainer*”, „*GPS*”, „*OSD (Analog)*”, „*OSD (Digital)*”, „*Pin IO*” oraz „*VTX*” niezmienione (zaznaczone jako dodane). Protokół telemetrii pozostawiono bez zmian na wartości automatycznie zawartej, natomiast jako protokół silników wybrano „*DSHOT*”, zgodnie z zalecaniami producenta kontrolera silników^[27]. W kolejnym kroku załadowano oprogramowanie online i wgrano je do kontrolera. W tym kroku zważano jedynie na ciągłe połączenie kontrolera lotu z komputerem. Całość procesu, widoczna również była w konsoli programu znajdującej się w górnej jego części. Po zakończonej konfiguracji, przystąpiono do konfiguracji

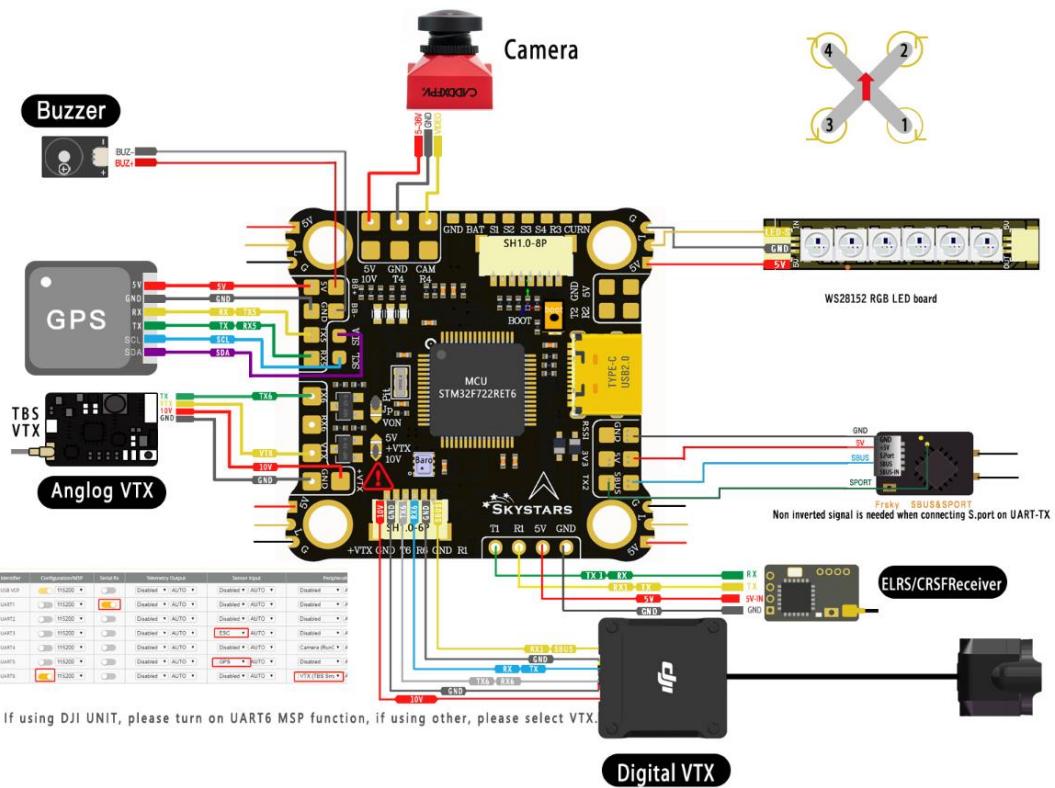
jednostki latającej. W tym miejscu, proces konfiguracji drona zaczął przeplatać się z konfiguracją kontrolera RC, co prowadziło do naprzemiennego konfigurowania jednego i drugiego urządzenia – zarówno kontrolera lotu, jak i nadajnika. W związku z zalecankami odnośnie unikania nagrzewania kontrolera lotu, całość podzielona była na etapy, aby nie dopuścić do jego przegrzania. Wobec powyższego, kolejno otwierano następujące po sobie zakładki konfiguratora, w następującej kolejności:

- 1) Zakładka „*Setup*” – pierwsza z dostępnych w konfiguratorze zakładek oferuje pilotowi podgląd parametrów drona, stan żyroskopu oraz jego aktualne położenie w lokalnym układzie odniesienia – wartości *yaw*, *roll* i *pitch* odpowiadają kolejno za wartości: odchylenia w obrocie wokół osi pionowej (zmiana kierunku lewo/prawo), przechylenia w obrocie wokół osi podłużnej (zmiana kierunku na przód/tył kadłuba) oraz pochylenia w obrocie wokół osi poprzecznej (zmiana kierunku w górę/w dół). Zakładka również daje podgląd na stan aktualnych komponentów podłączonych do kontrolera lotu, takich jak: temperatura procesora kontrolera lotu, stan naładowania akumulatora podawany w Voltach, aktualne zużycie akumulatora podawane w Amperach, model żyroskopu, model barometru i inne. W zakładce *Setup*, użytkownik skalibrować może akcelerometr oraz magnetometr i w przypadku niniejszej pracy, skalibrowany został pierwszy z wymienionych komponentów. Aby wykonać tą konfigurację poprawnie, położono drona na równej, całkowicie poziomej powierzchni oraz naciśnięto przycisk „*Calibrate Accelerometer*”. Ta operacja, z perspektywy bezawaryjnego lotu jest niezwykle istotna, gdyż właśnie to położenie skonfigurowane w niniejszym kroku, będzie w przyszłości odniesieniem dla oprogramowania, do poziomowania drona w powietrzu. Wynikiem powyższego działania, było otrzymanie poprawnie skonfigurowanego akcelerometru co pokazuje rysunek 21.

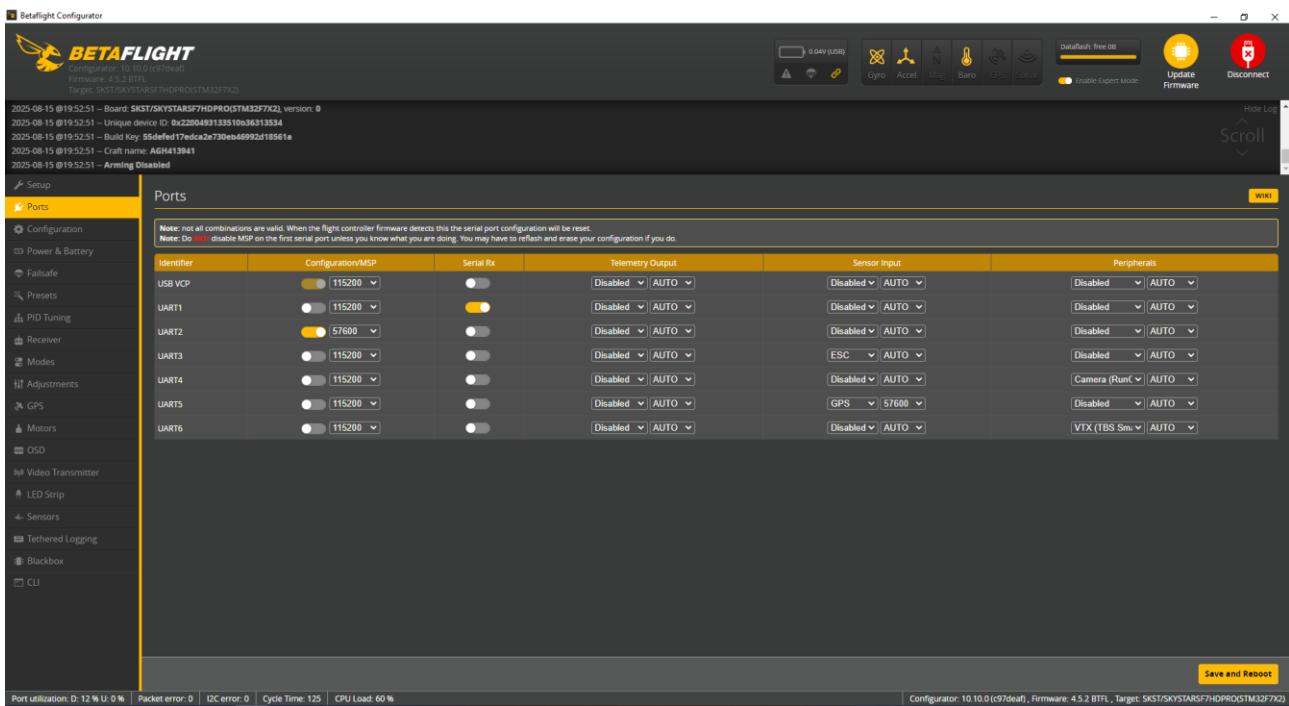


Rysunek 23 - Zakładka Setup w programie Betaflight Configurator
źródło: Opracowanie własne

- 2) Zakładka „Ports” – druga z dostępnych w oprogramowaniu zakładek, umożliwia użytkownikowi powiązanie fizycznie podłączonych do kontrolera lotu komponentów, z odpowiadającymi im własnościami w oprogramowaniu. W tej zakładce, pilot chcąc wykorzystać urządzenia takie jak GPS, czy nadajnik video, **musi** wybrać na który port szeregowy fizycznie je podłączył oraz mówiąc krótko „powiedzieć” oprogramowaniu jak czytać dane odebrane z tych komponentów i co z nimi robić^[28]. Wobec fizycznego podłączenia komponentów do drona opisywanego w niniejszej pracy, wybrano odpowiadające im porty szeregowe na kontrolerze lotu zgodnie z zaleceniami producenta kontrolera, co powinno zawsze być priorytetem dla dronów budowanych własnoręcznie, gdyż nie zawsze – tak jak w przypadku kontrolera lotu *Skystars F7 22 HD Pro 4* – wyjścia portów szeregowych widnieją jasno opisane na kontrolerze lotu. Wobec powyższego, mają na uwadze rysunek 22 prezentujący rysunek producenta kontrolera lotu pokazujący wyprowadzenia portów szeregowych i dedykowanych im komponentów, zaznaczono właściwe opcje w oprogramowaniu Betaflight Configurator – wynik tej operacji prezentuje rysunek 22.



Rysunek 24 - Fragment schematu połączeń kontrolera lotu Skystars F7 22 HD Pro 4

Źródło: Strona z częściami do dronów FPV – rotorama.cz – na podstawie dokumentacji producenta kontrolera lotu^[29]

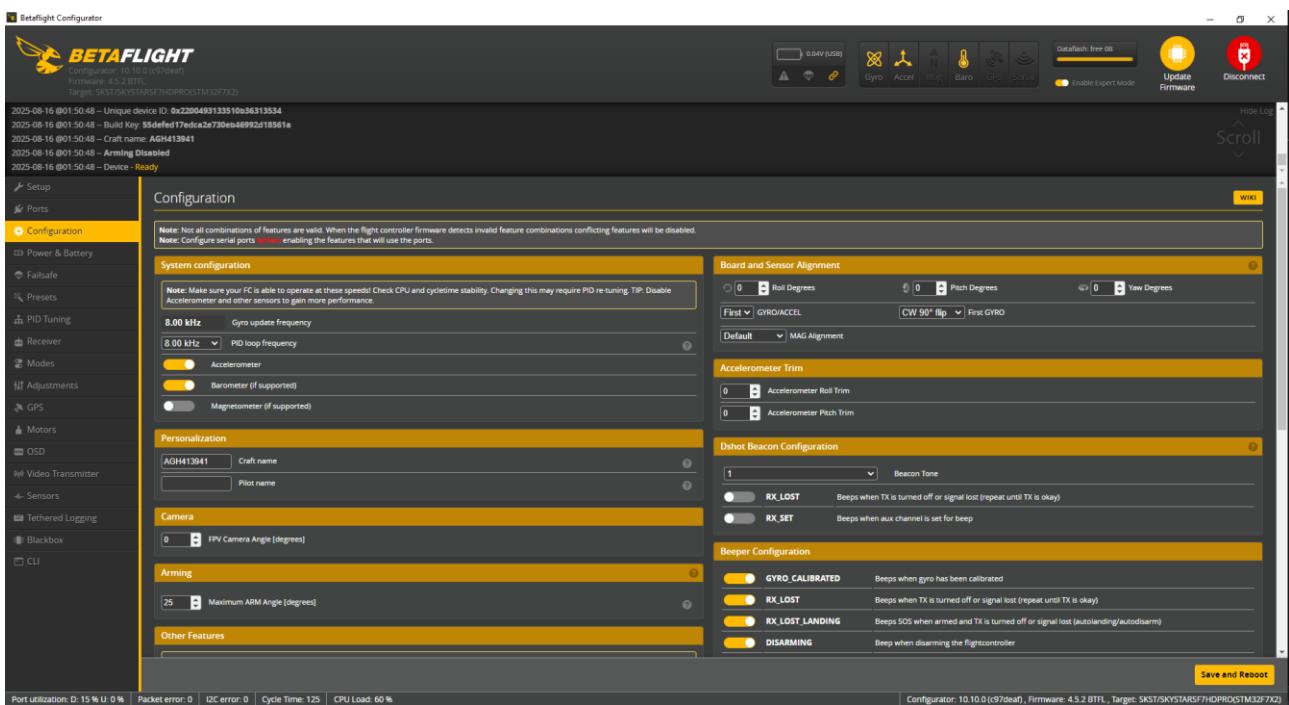
Rysunek 25 - Zrzut ekranu ukazujący poprawną konfigurację portów szeregowych w oprogramowaniu Betaflight Configurator

Źródło: opracowanie własne

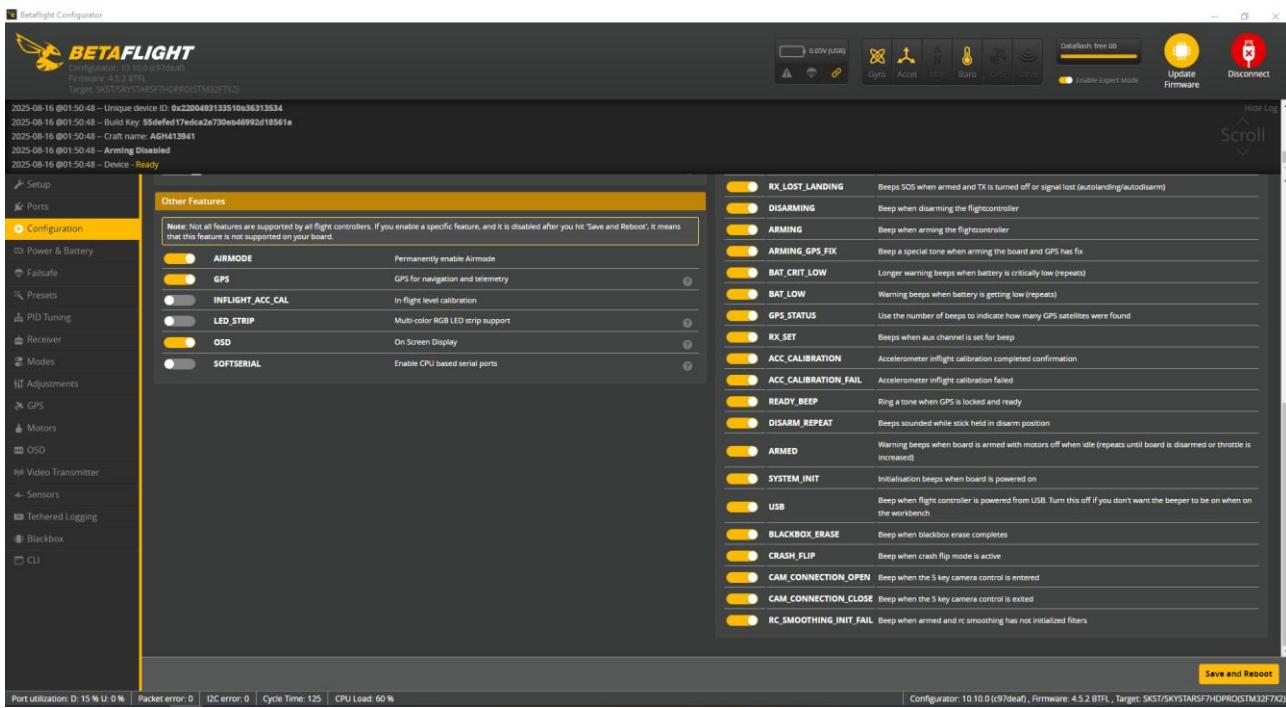
Rysunek 23, ukazujący poprawnie dokonaną konfigurację portów szeregowych należy rozumieć w ten sposób, że fizycznie przylutowane zostały komponenty takie jak: na porcie szeregowym UART1, jak to zwykle ma miejsce, został podłączony odbiornik RC, odbierający sygnał z nadajnika będącym kontrolerem naziemnym – dla tego komponentu, zaznaczono również opcję „*Serial RX*” gdyż jest to port szeregowy, do którego podłączony jest priorytetowy odbiornik sterowania dronem; na porcie UART2 zostały przylutowane przewody przesyłowe (RX oraz TX) które w późniejszym czasie zostały podłączone do jednostki Arduino Uno – w tym przypadku, suwakiem zaznaczono również opcję „*MSP*” z uwagi na planowane przesyłanie danych z Arduino do kontrolera lotu ww. protokołem; na porcie UART3 wedle schematu producenta, podłączony został kontroler silników – ten element nie był lutowany, gdyż podłącza się go najczęściej za pomocą wtyczki; na porcie UART4 została przylutowana kamera FPV służąca do podglądu obrazu z drona w czasie rzeczywistym; na porcie UART5 został przylutowany GPS oraz ustawiona została zalecana przez producenta tego komponentu częstotliwość komunikacji wynosząca 57600 Bd; na ostatnim z dostępnych portów szeregowych – UART6, przylutowany został nadajnik Video, odpowiadający za przesyłanie obrazu z kamery do gogli FPV zakładanych przez pilota i dla tego komponentu została wybrana opcja VTX (TBS SmartAudio) zgodnie z zaleceniami producenta nadajnika. Do każdego z powyższych połączeń należało wybrać odpowiednią częstotliwość komunikacji z kontrolerem lotu. We wszystkich, z wyjątkiem dwóch komponentów – GPS oraz przewodów przesyłowych do Arduino Uno – wynosiła ona 115200 Bd. W opisanych dwóch wyjątkach, częstotliwość wyniosła 57600 Bd. Do żadnego z prezentowanych urządzeń nie została skonfigurowana pod-zakładka „*Telemetry Output*”, gdyż jest ona przeznaczona dla starszych systemów wymagających telemetrii.^[28]

- 3) Zakładka „*Configuration*” – ta zakładka przeznaczona jest do zmiany ustawień systemowych, personalizacji nazwy drona, nazwy pilota, ustawień pochylenia kamery, włączania/wyłączania opcji peryferii takich jak GPS, nakładka OSD (*On screen display*) na gogle i inne. Każda z opcji możliwych do wyboru została jasno opisana przez dostawcę oprogramowania – dalej: Betaflight – wobec czego opisanie wszystkich z możliwości zostało pominięte z uwagi na ich ilość. Najważniejszymi w niniejszej pracy – te wartości zostały ustawione/zmienione – okazały się być: w pod-zakładce „*System configuration*” włączony został akcelerometr oraz barometr; w pod-zakładce „*Personalization*” ustawiona została nazwa drona; w pod-zakładce „*Arming*” ustawiona została wartość maksymalnego pochylenia drona w której można go uzbroić jako 25°; w pod-zakładce „*Other features*” włączono takie

opcje jak *Airmode*, *GPS* oraz *OSD* jako pomocne do realizacji niniejszego projektu. *Airmode* będący opcją włączającą pełną korekcję PID przy zerowym gazie, umożliwia płynne akrobacje z gazem ustawionym na zero^[30], *GPS* natomiast został włączony z uwagi na fakt wykorzystywania tego modułu w projekcie. *OSD* jako nakładka na ekran w goglach umożliwia wyświetlenie przydatnych w trakcie lotu parametrów także tą opcję również włączono w niniejszej zakładce. Pozostałe parametry – „*Beeper configuration*” pozostawiono bez zmian, gdyż domyślnie wszystkie z nich są włączone. Wynik zakończonej konfiguracji drona w opisywanej zakładce prezentują rysunki 26 oraz 27.



Rysunek 26 - Zrzut ekranu prezentujący pierwszą część zakładki "Configuration" w programie Betaflight Configurator
 Źródło: Opracowanie własne



Rysunek 27 - Zrzut ekranu prezentujący drugą część zakładki "Configuration" w programie Betaflight Configurator
Zródło: opracowanie własne

- 4) Zakładka „Power & Battery” – w niniejszej zakładce powinno edytować się maksymalne, minimalne oraz ostrzegawcze naładowanie akumulatora zainstalowanego na maszynie. Jest to o tyle ważne, gdyż system wówczas po dotarciu do pułpu ostrzegawczego, da informację pilotowi w postaci, którą uprzednio wybierze w konfiguracji. Etap ten jest również ważny z tego względu, że akumulator, który straci całkowicie swoje napięcie (osiągnie napięcie naładowania) uniemożliwi pilotowi bezpieczne lądowanie. W dalszych etapach, wybrano sposób ostrzegania pilota, jednakże, aby było to możliwe, konieczne jest uprzednie wprowadzenie poprawnych wartości i tak w niniejszej konfiguracji zostało uczynione. Wartości, które wpisano w niniejszej zakładce prezentuje tabela 1.

Tabela 1 - Opis wartości wstawionych w zakładce „Power & Battery” programu Betaflight Configurator

Kryterium	Wartość wpisana	Jednostka
Minimalny poziom naładowania baterii (średni)	3.3	V
Maksymalny poziom naładowania baterii (średni)	4.2	V
Ostrzegawczy poziom naładowania baterii (średni)	3.5	V

Pojemność baterii	1550	mAh
-------------------	------	-----

Źródło: opracowanie własne

Niniejsze wartości zostały wpisane następująco, gdyż odpowiadają one parametrom pojedynczego ogniw w zastosowanym w projekcie akumulatorze. Zastosowana została bateria typu 6S, wobec czego zgodnie z tym co podaje producent, nominalna wartość naładowania pakietu wynosi 22.2V, a co za tym idzie, nominalna wartość:

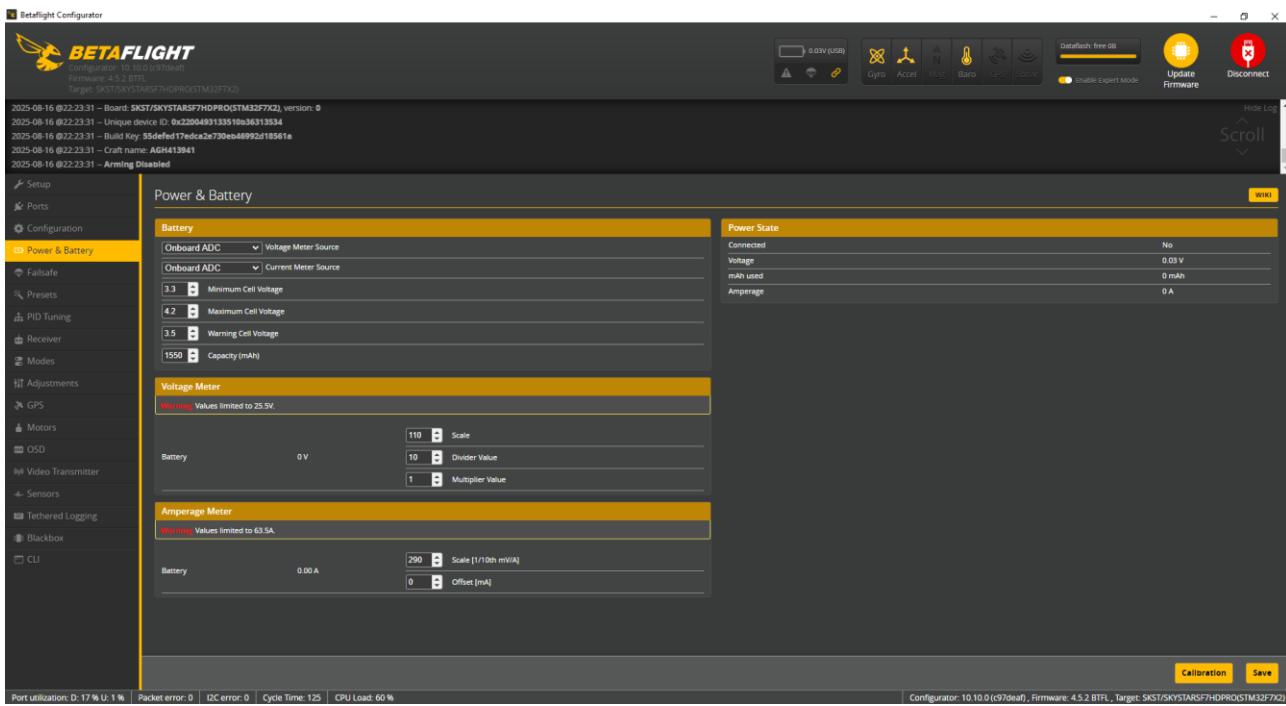
$$nom_cell_vol [V] = \frac{nom_vol [V]}{num_cells}^{(1)} \text{ gdzie:}$$

Równanie 1 – wzór obliczania nominalnego naładowania dla pakietu 2-6S

Źródło: opracowanie własne

nom_vol - nominalne całkowite naładowanie akumulatora - w opisywanym przypadku 22.2V
num_cells - liczba ogniw zastosowanych w akumulatorze - w opisywanym przypadku 6 sztuk

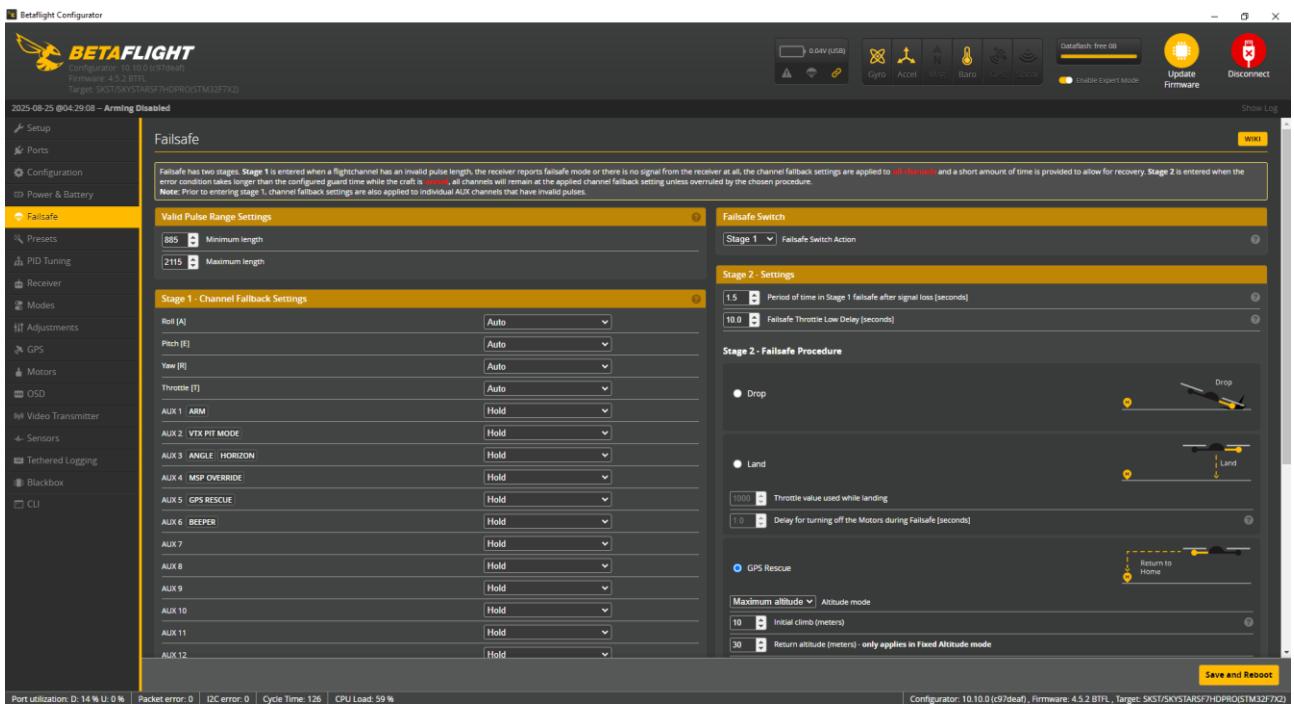
Wobec powyższego wzoru, ustalono, że nominalny poziom naładowania pojedynczego ogniw (a zarazem średni dla całej baterii) wynosi 3.7V. Typową cechą charakterystyczną ogniw litowo-polimerowych jest również maksymalny stan naładowania wynoszący 4.2V, wobec powyższego taka wartość została wpisana. Wartości ostrzegawcze, są wartościami o których decyduje użytkownik, jednakże nie zaleca się schodzenia poniżej 3.3V na ogniwie, z uwagi na fakt, iż utrzymywanie minimalnego progu naładowania powyżej 3V wydłuża żywotność ogniw^[31]. Wartość ostrzegawcza została ustawiona na 3.5V, aby dać pilotowi wystarczającą ilość czasu do bezpiecznego wylądowania. Finalną zawartość opisywanej zakładki ukazano na rysunku 28.



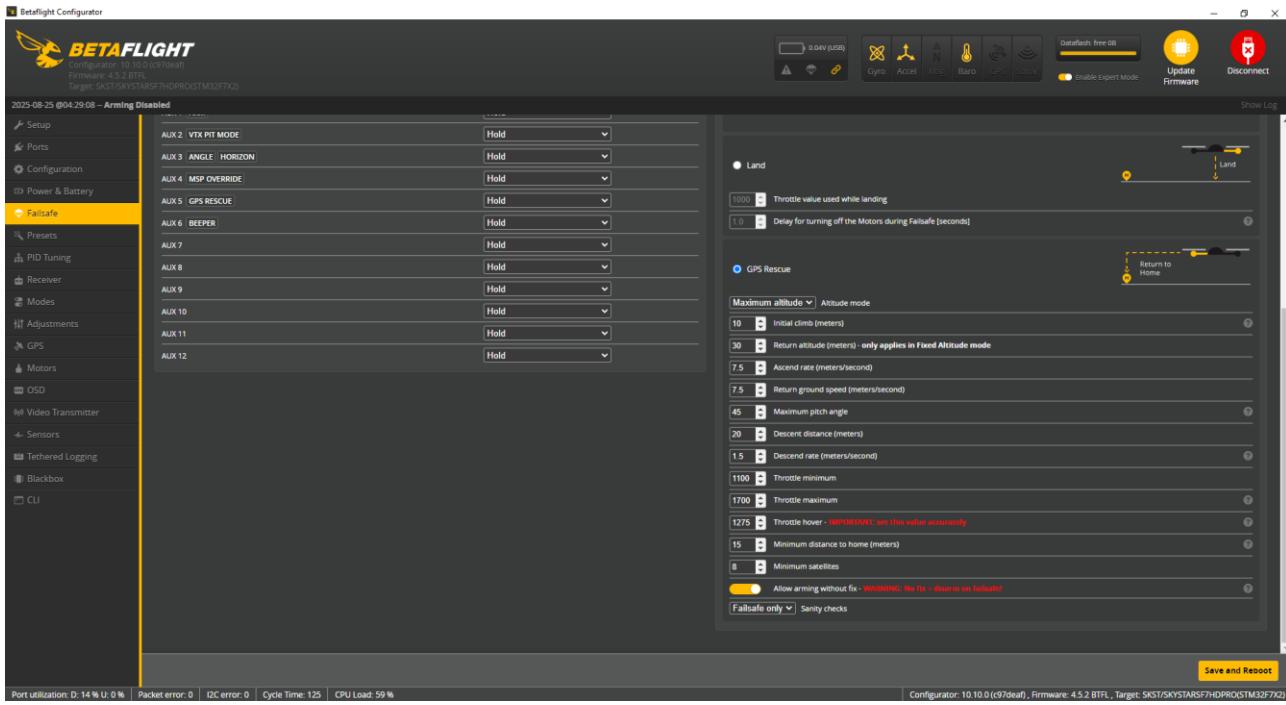
Rysunek 28 - Zrzut ekranu z zawartością zakładki "Power & Battery" w programie Betaflight Configurator
źródło: opracowanie własne

- 5) Zakładka „Failsafe” – jest to zakładka wykorzystana w niniejszym projekcie, umożliwiająca dostosowanie zachowania drona w sytuacji awaryjnej, przykładowo: utraty sygnału sterowania. W opisywanym przypadku, została wykorzystana opcja powrotu drona do miejsca startu lotu – *RTH (Return To Home)* lub *GPS Rescue*. Wybierając tą opcję jako domyślną w razie sytuacji awaryjnej, użytkownik powinien wprowadzić parametry powrotu drona do miejsca startu, takie jak: wznos w lokalnym układzie odniesienia (od miejsca uruchomienia trybu), prędkość wznosu, ewentualny opad drona oraz jego prędkość, agresywność tego procesu (maksymalną wartość odchylenia w osi poprzecznej), minimalną liczbę satelit które dron musi pierwotnie złapać zanim dojdzie do jego użbrojenia i inne. Wartością, która została zmieniona i różniła się od domyślnych, była jedynie liczba satelit – wartość tą ustawiiono na 8. Wobec zastosowania funkcji powiązanej z modułem GPS, gdyż cała opisywana zakładka jest dedykowana dla dronów wykorzystujących ten komponent, na czas testów w pomieszczeniu zaznaczono również opcję uzbijania drona bez tzw. „fix'a”. Fix, jest to sytuacja, gdy GPS na dronie jest powiązany z minimalną liczbą satelit, co w przypadku testów sprzętowych wewnętrz pomieszczenia nie jest możliwe. Aby moduł GPS został powiązany z satelitami, musi otrzymywać zasilanie oraz znajdować się na zewnątrz. Optymalnymi warunkami, które zdecydowanie przyspieszają proces znajdywania satelit, jest wyjście z dronem na powierzchnię w żaden sposób nie zasłoniętą – bez drzew, dachów – oraz

odczekanie czasu około 5 minut. Sama w sobie maszyna latająca, zawsze komunikuje znalezienie odpowiedniej liczby satelit w sposób opisany przez producenta. W opisywanym przypadku, po podłączeniu brzęczyka, dron również komunikuje brak opisywanego „fix'a”. Skonfigurowaną zawartość zakładki Failsafe zaprezentowano na rysunkach 29 oraz 30.



Rysunek 29 - Zrzut ekranu prezentujący zawartość zakładki Failsafe w programie Betaflight Configurator - część 1
 Źródło: opracowanie własne



Rysunek 30 - Zrzut ekranu prezentujący zawartość zakładki Failsafe w programie Betaflight Configurator - część 2
źródło: opracowanie własne

- 6) Zakładka „Presets” – jest to zakładka przeznaczona do stosowania łatwych konfiguracji maszyny, przygotowanych przez innych użytkowników. Każdy element w opisywanej zakładce, to praktycznie rzecz biorąc zestaw komend konsoli, które zostały zastosowane przez autora. W niniejszym przykładzie, wykorzystany został jedynie jeden preset, przygotowujący tabelę dla nadajnika video o nazwie „Rush Tiny Tank, Tank Racing, Tank Ultimate

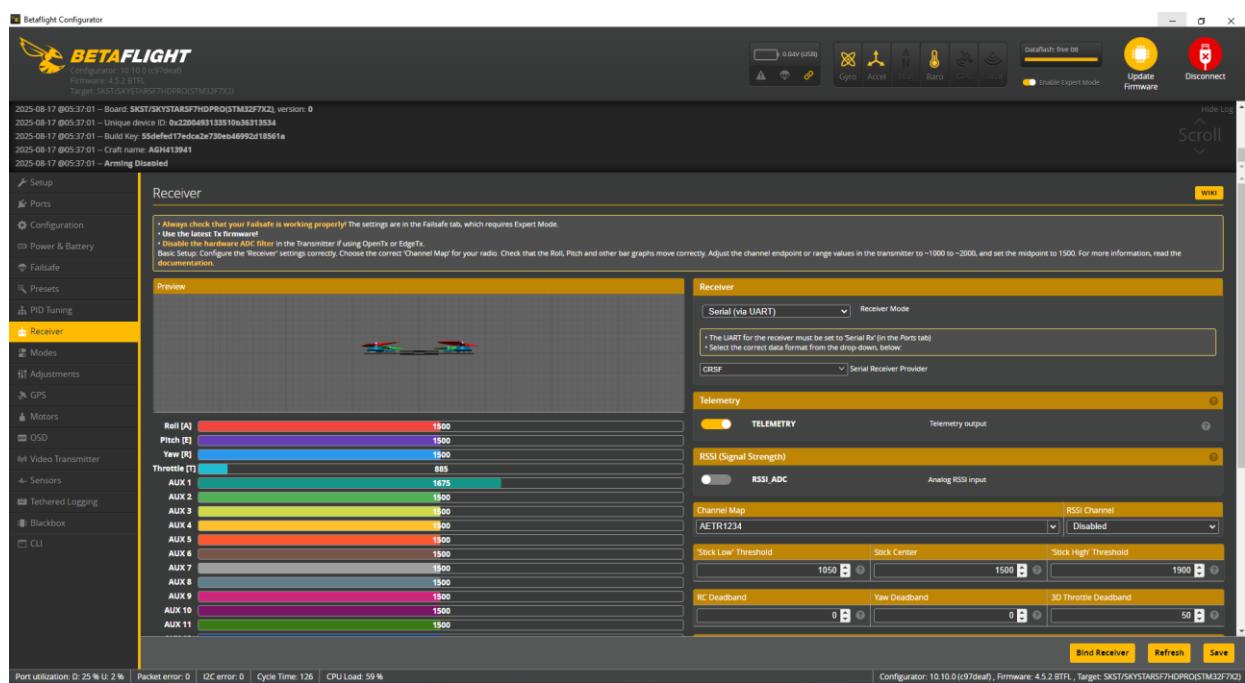
```
# reset VTXtable
vtxtable bands 0
vtxtable channels 0
vtxtable powerlevels 0
vtxtable powervalues
vtxtable powerlabels
vtxtable bands 5
vtxtable channels 8
vtxtable band 1 BAND_A A CUSTOM 5865 5845 5825 5805 5785 5765 5745 5725
vtxtable band 2 BAND_B B CUSTOM 5733 5752 5771 5790 5809 5828 5847 5866
vtxtable band 3 BAND_E E CUSTOM 5705 5685 5665 5665 5885 5905 5905 5905
vtxtable band 4 AIRWAVE F CUSTOM 5740 5760 5780 5800 5820 5840 5860 5880
vtxtable band 5 RACEBAND R CUSTOM 5658 5695 5732 5769 5806 5843 5880 5917
vtxtable powerlevels 4
vtxtable powervalues 14 26 29 32
vtxtable powerlabels 25 400 800 MAX
```

*Fragment kodu 1 - zawartość zestawu dla nadajnika video wykorzystanego podczas konfiguracji
 Źródło: zestaw konfiguracyjny użytkownika „Directory” w programie Betaflight Configurator*

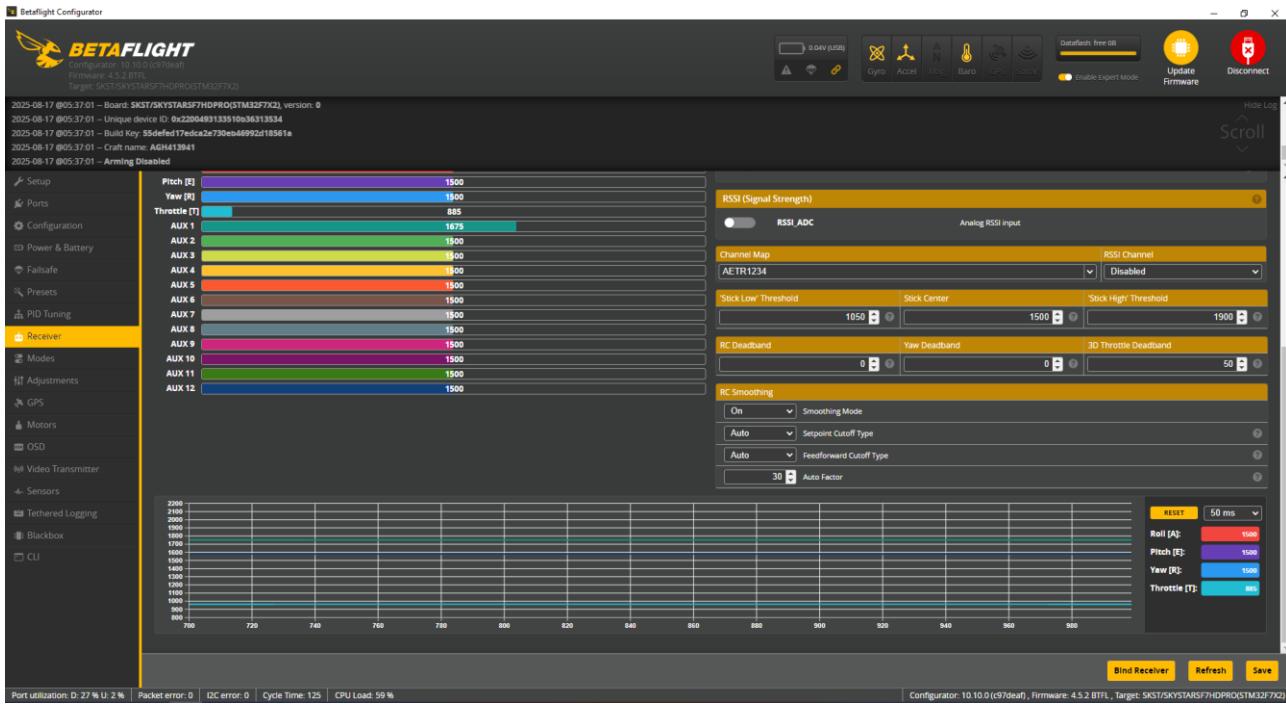
Plus/Mini/II, and Tank Solo VTX tables” autora Directory. Zestaw komend zawierających się w niniejszym zestawie, wykorzystanym w trakcie konfiguracji, prezentuje fragment kodu 1. Wpisanie wartości zaprezentowanych we fragmencie kodu 1 do konsoli systemowej w programie Betaflight Configurator, skutkowałoby działaniem identycznym do zastosowania wyżej opisanego, gotowego zestawu, wobec powyższego zdecydowano się na użycie zakładki presets, gdyż jest to proces szybszy – następuje niemal natychmiast.

- 7) Zakładka „PID Tuning” – jest to zakładka przeznaczona dla zaawansowanych pilotów oraz konstruktorów, gdyż umożliwia dostosowanie kontrolera PID. Kontroler PID, jest elementem, przez który niejako sygnał sterujący przechodzi, zanim trafi do silników drona. Kontroler PID kontroluje każdy aspekt dynamiki lotu drona, a więc reaguje na ruchy drążków, stabilizuje model oraz porównuje żądaną i rzeczywistą prędkość kątową. W rozwinięciu skrótu PID, przez poszczególne litery rozumie się: P – siła korekty do celu, I – korekcja długotrwałych błędów, D – zmniejszenie przeregulowania, patrząc na tempo zmian błędu^[32]. Jest to zakładka stricte przeznaczona dla osób, które chcą dostosować sposób w jaki układ sterujący reaguje na ruch drążków a w konsekwencji tego, w jaki sposób lata dron. Wobec powyższego, nie chcąc przesadnie ingerować w ww. system, zmieniono jedynie główny mnożnik, obniżając jego wartość do poziomu 0.3 wartości podstawowej, gdyż taki stan pozwala na powolne obracanie maszyną w powietrzu, w każdej z trzech osi obrotu. Finalny efekt modyfikacji ukazano na rysunku 31.

- 8) Zakładka „*Receiver*” – jak nazwa wskazuje, niniejsza zakładka przeznaczona jest do modyfikacji oraz weryfikacji działania odbiornika RC. Wobec ustawień fabrycznych, zmodyfikowany został jedynie parametr „*RC Smoothing*” i został on włączony. Jest to opcja poprawiająca jakość połączenia pomiędzy odbiornikiem oraz nadajnikiem RC, z uwagi na możliwość występowania zakłóceń wejścia po stronie pilota RC. Pozostawiono wartość średkową gałki pilota jako 1500 oraz pozostawiono wartość minimalną gałki jak 1000. Są to wartości domyślne. Końcowy efekt konfiguracji wyżej opisanej zakładki, pokazano na rysunkach 31 oraz 32.



Rysunek 31 - zakończona konfiguracja drona pod zakładką "Receiver" - część 1
Źródło: opracowanie własne



Rysunek 32 - zakończona konfiguracja drona pod zakładką "Receiver" - część 2
 Źródło: opracowanie własne

- 9) Zakładka „*Modes*” – jedna z najważniejszych zakładek programu konfiguracyjnego, będąca odpowiedzialna za przypisanie zasięgów kanałów AUX do odpowiadających funkcji oprogramowania. Jest to zakładka, w której użytkownik definiuje, w jakim zasięgu działania przełączników fizycznych na pilocie RC będą działać funkcje oprogramowania takie jak użbrajanie maszyny, przełączanie trybów latania, włączanie brzęczyka, zmniejszanie mocy nadajnika video i inne. Funkcje wykorzystane w niniejszej pracy oraz zasięgi kanałów im przypisanych prezentuje tabela 2.

Tabela 2 - Wykorzystane funkcje zakładki „*Modes*” programu Betaflight Configurator wraz z przypisanymi wartościami kanałów AUX

Funkcja	Definicja	Kanał	Zasięg
ARM	Uzbrojenie drona	AUX1	1700-2100
ANGLE	Tryb latania z włączonym pełnym wspomaganiem każdej osi obrotu – brak możliwości	AUX3	900-1300

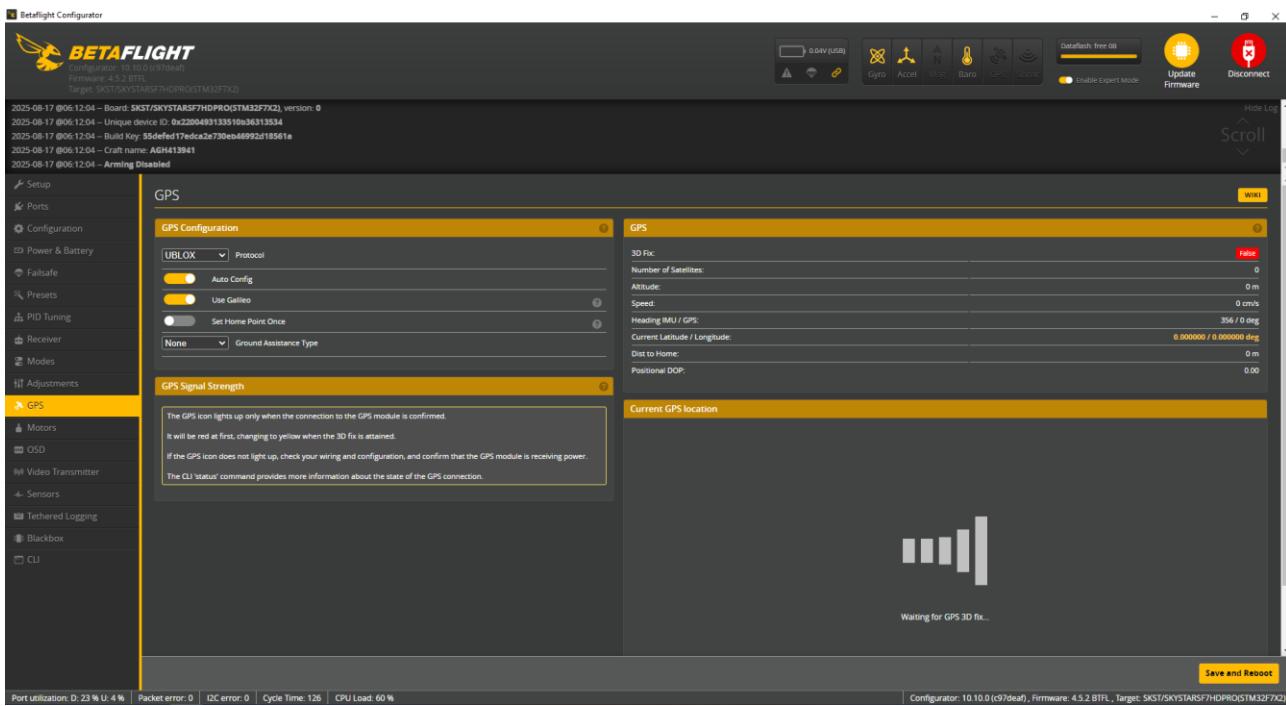
	wykonywania manewrów		
HORIZON	Tryb latania złączonym wspomaganiem każdej z osi obrotu – możliwość wykonywania manewrów	AUX3	1300-1700
VTX PIT MODE	Przełączenie nadajnika video w stan niskiej emisji sygnału (niskiego poboru prądu)	AUX2	900-1300
GPS RESCUE	Uruchomienie trybu awaryjnego wybranego przez użytkownika w zakładce <i>Failsafe</i> – w tym przypadku powrotu do miejsca startu	AUX5	1700-2100
BEEPER	Włączenie brzęczyka	AUX6	1700-2100
MSP OVERRIDE	Włączenie trybu nadpisywania danych kontrolera lotu poprzez protokół MSP	AUX4	1300-2100

Źródło: opracowanie własne

Niniejsze wartości zostały dobrane po uprzednim sprawdzeniu, jakie wartości generuje naciśnięcie lub przełączenie poszczególnych guzików/przełączników na pilocie RC. Do tego

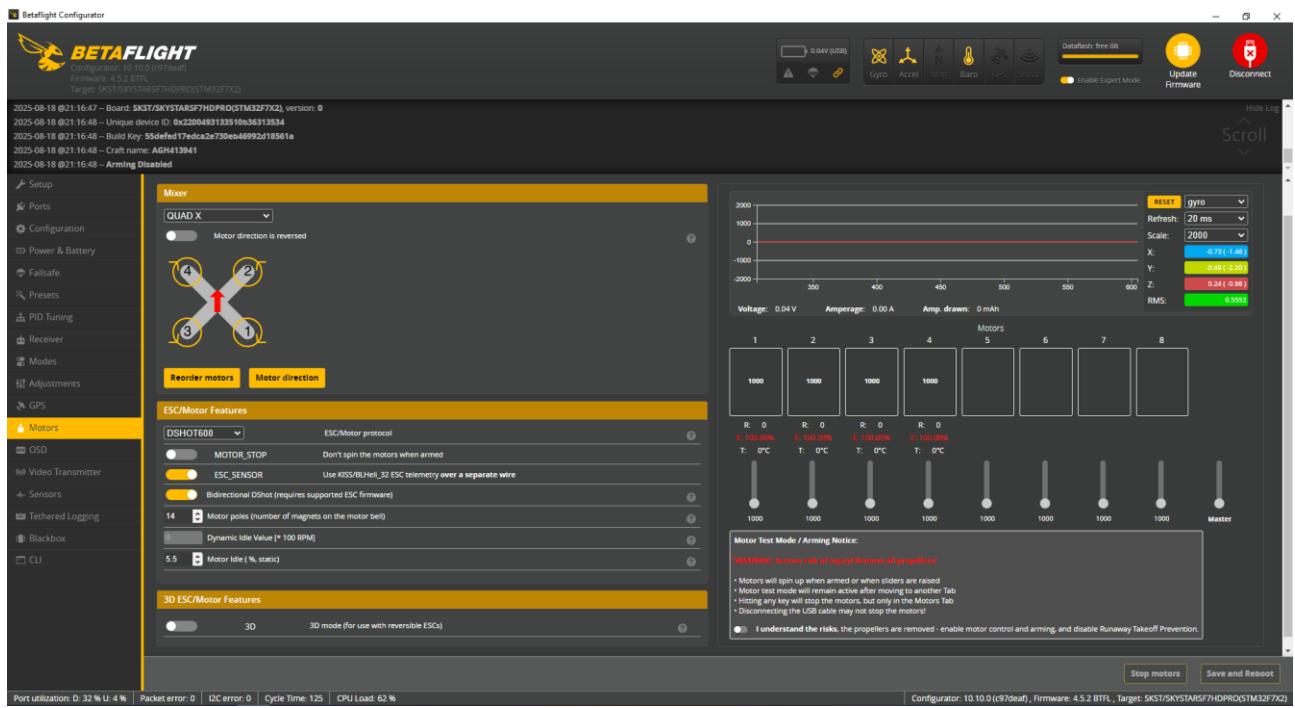
celu, wykorzystano zakładkę *Receiver*, gdyż daje ona podgląd kanałów AUX w czasie rzeczywistym. Jednocześnie, mając otwarty podgląd, zanotowano, który kanał odpowiada któremu guzikowi/przełącznikowi na pilocie. Aby przypisać niniejsze wartości prawidłowo, należy uprzednio poprawnie skonfigurować wartości wysyłane do odbiornika z pilota, a co najważniejsze, prawidłowo sparować ze sobą nadajnik oraz odbiornik RC, co zostało szczegółowo omówione w rozdziale 3.2. Charakterystyka i konfiguracja oprogramowania kontrolera RC. Szczególną uwagę należy zwrócić na funkcję „*VTX PIT MODE*” gdyż niewłaściwe jej skonfigurowanie może w konsekwencji prowadzić do nieustannej pracy nadajnika wideo po dostarczeniu zasilania z baterii LiPo (bądź innej zasilającej drona), co może prowadzić do nagrzewania się ww. komponentu a w ostateczniie nawet do jego spalenia bądź trwałego uszkodzenia. Jest to ważne w dowolnej konfiguracji drona FPV, gdyż nadajnik obrazu wideo jest elementem kluczowym do bezpiecznego użytkowania drona.

- 10) Zakładka „*Adjustments*” – niniejsza zakładka nie została w żaden sposób wykorzystana w niniejszej pracy.
- 11) Zakładka „*GPS*” – w podanej zakładce programu konfiguracyjnego, użytkownik ma możliwość dostosowania działania wskazanego w nazwie zakładki modułu obecnego na dronie. W przypadku opisywanej konfiguracji, zgodnie z tym co wskazał producent zamontowanego modułu lokalizacyjnego, została włączona opcja wykorzystywania systemu GALILEO, będącego europejskim odpowiednikiem niezależnego systemu satelitarnego, a także wybrany został protokół komunikacji – UBLOX, gdyż takim protokołem posługuje się wybrany GPS^[14].



Rysunek 33 - Zrzut ekranu prezentujący zakończoną konfigurację w zakładce "GPS" programu Betaflight Configurator
źródło: opracowanie własne

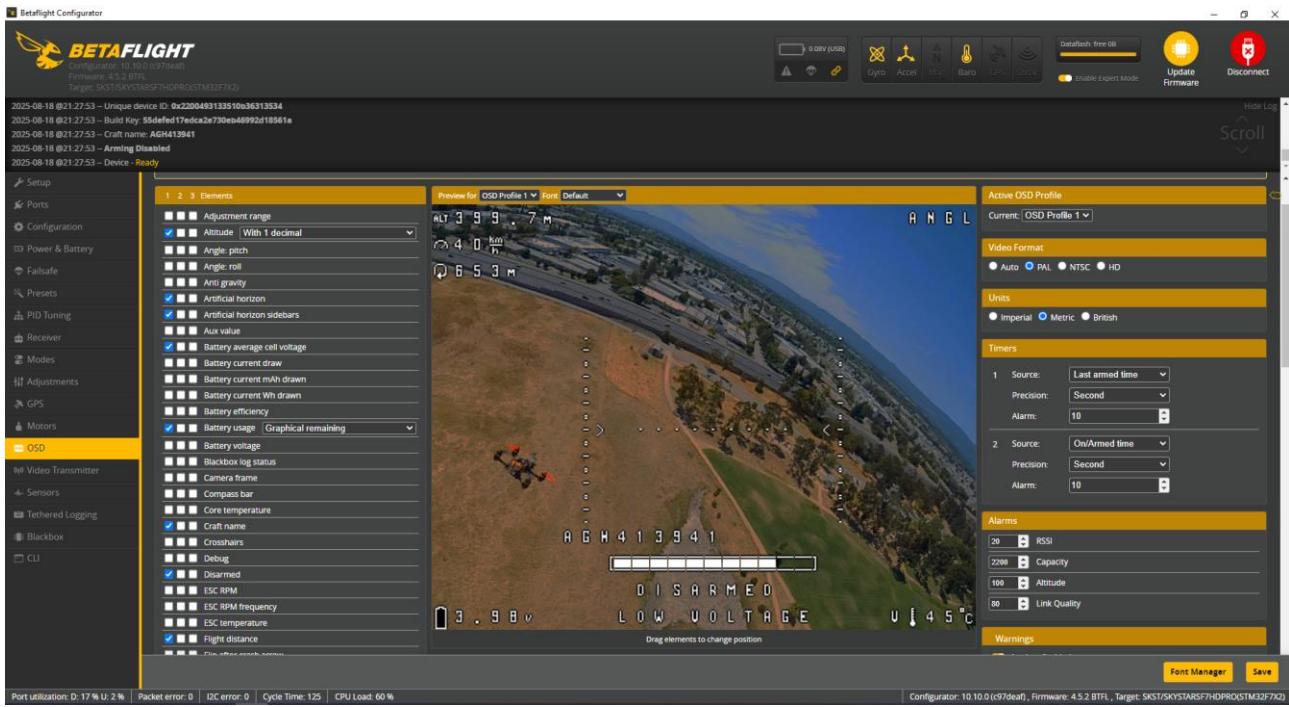
12) Zakładka „*Motors*” – niniejsza zakładka daje użytkownikowi dostęp do schematu działania silników zamontowanych na maszynie. Można na jej podstawie zweryfikować oraz edytować kierunek obrotu silników, a także dostosować protokół komunikacji kontrolera lotu z kontrolerem silników. W opisywanym przypadku, zgodnie ze specyfikacją techniczną kontrolera silników, wybrano protokół kontrolera silników jako DSHOT600. Wśród dostępnych możliwości, w przypadku kontrolera silników zamontowanego w niniejszym przykładzie, znalazły się również DSHOT300, DSHOT150, ONESHOT125, ONESHOT42, MULTISHOT czy PWM^[27]. Opisywany kontroler, zgodnie z tym co podaje producent, wspiera również technologię przesyłania telemetrii z uwagi na wgrane oprogramowanie BLHeli_32 a także obustronny DSHOT, wobec czego opcje te zostały zaznaczone jako włączone w oprogramowaniu konfiguracyjnym. Zakładka *Motors* umożliwia również sprawdzenie w czasie rzeczywistym, „na surowo” czy silniki w ogóle działają po podłączeniu baterii LiPo, wobec czego oprogramowanie daje możliwość weryfikacji połączenia fizycznego: silników do kontrolera silników oraz kontrolera silników do kontrolera lotu. W opisywanym przypadku, wszelkie testy zakończone zostały z powodzeniem i wszystkie połączenia były sprawne na każdym etapie testowania. Zakończoną konfigurację silników w zakładce zaprezentowano na rysunku 34.



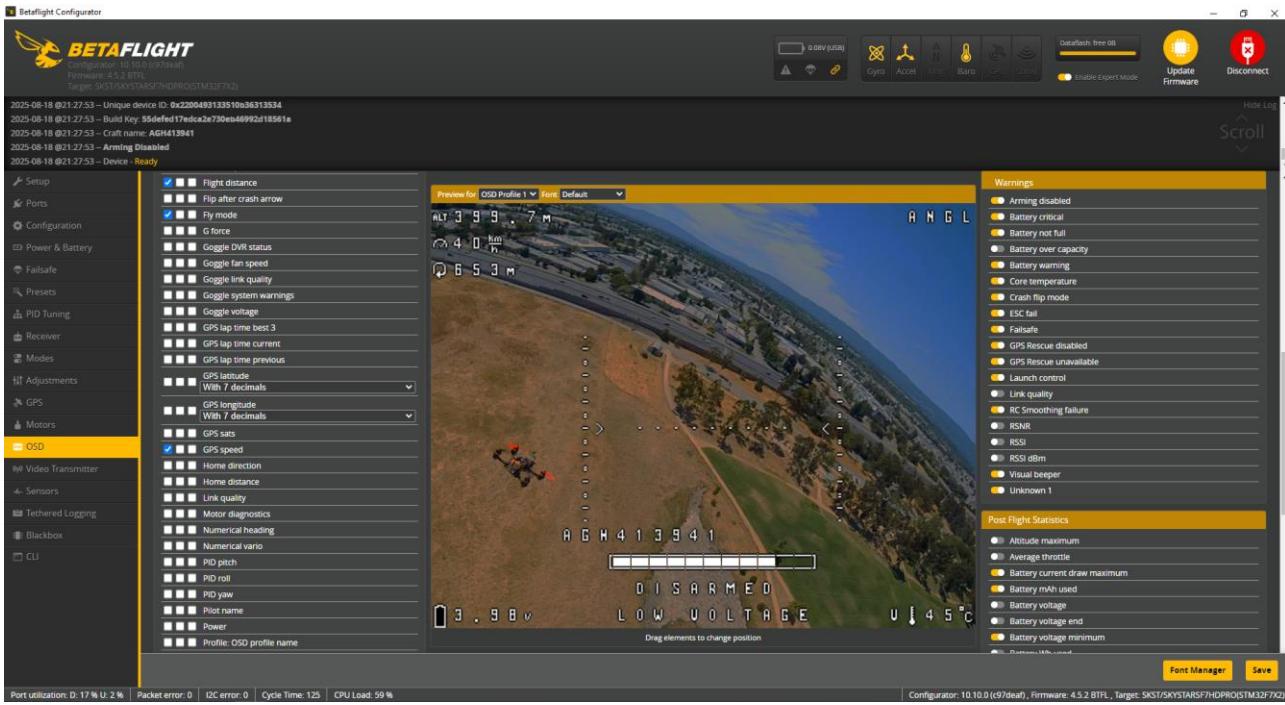
Rysunek 34 - Zrzut ekranu prezentujący zakończoną konfigurację w zakładce "Motors" programu Betaflight Configurator
 Źródło: opracowanie własne

13) Zakładka „OSD” – jest zakładką skoncentrowaną na personalizacji nakładki oprogramowania Betaflight w goglach FPV – z ang. *on screen display*. Wobec powyższego, program konfiguracyjny udostępnia użytkownikowi łatwy i intuicyjną siatkę przeznaczoną do umieszczania na niej elementów, które pilot chce widzieć w trakcie lotu dronem, nakładając je niejako na obraz kamery. Wobec powyższego, użytkownik ma możliwość własnoręcznego ustawienia i rozmieszczenia elementów, które widzieć będzie zarówno w trakcie, jak i po wykonaniu lotu na panelu podsumowania. Zakładka również wymaga dostosowania formatu obrazu video przesyłanego z nadajnika VTX do gogli FPV. W związku z wybranym modelem gogli i ich kompatybilnością, a także analogowym sposobem przesyłania obrazu, wybrano w tym przypadku format obrazu PAL. Jak chodzi o format HD, to jego wybór w przypadku gogli odbierających sygnał z nadajnika w sposób analogowy, jest niemożliwy. Przechodząc do siatki elementów widocznych przez pilota, skoncentrowano się na elementach niezbędnych oraz przydatnych w trakcie lotu dla każdego pilota. W niniejszym przypadku, takimi elementami okazały się być: wysokość maszyny, sztuczny horyzont oraz jego pasy boczne, średnia naładowania ogniw akumulatora, zużycie baterii, nazwa maszyny, prędkość GPS, tryb lotu, status uzbrojenia, przebyty przez maszynę dystans, temperatura nadajnika video oraz ostrzeżenia systemowe. Wszystkie wymienione wyżej elementy zostały rozmieszczone na siatce według własnego uznania i są jedynie jednym z wielu przykładów,

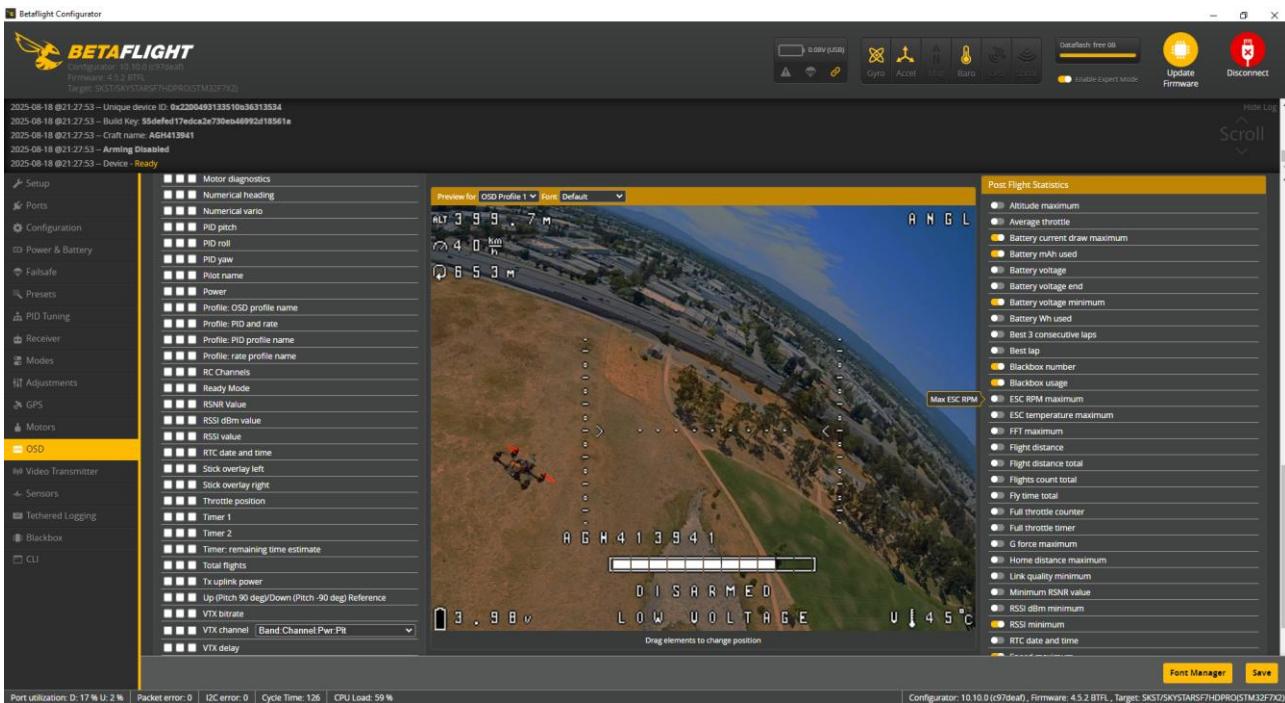
w jaki może zostać skonfigurowana nakładka na obraz w goglach FPV. Zakładka OSD umożliwia także dostosowanie cech ostrzeżeń systemowych a także wybranie parametrów alarmu, jednakże w tej materii nie dokonano żadnych zmian względem domyślnych wartości, gdyż są one wystarczające dla niniejszego projektu. Zakończoną konfigurację oraz ostateczny wygląd siatki, zaprezentowano na rysunkach 35-38.



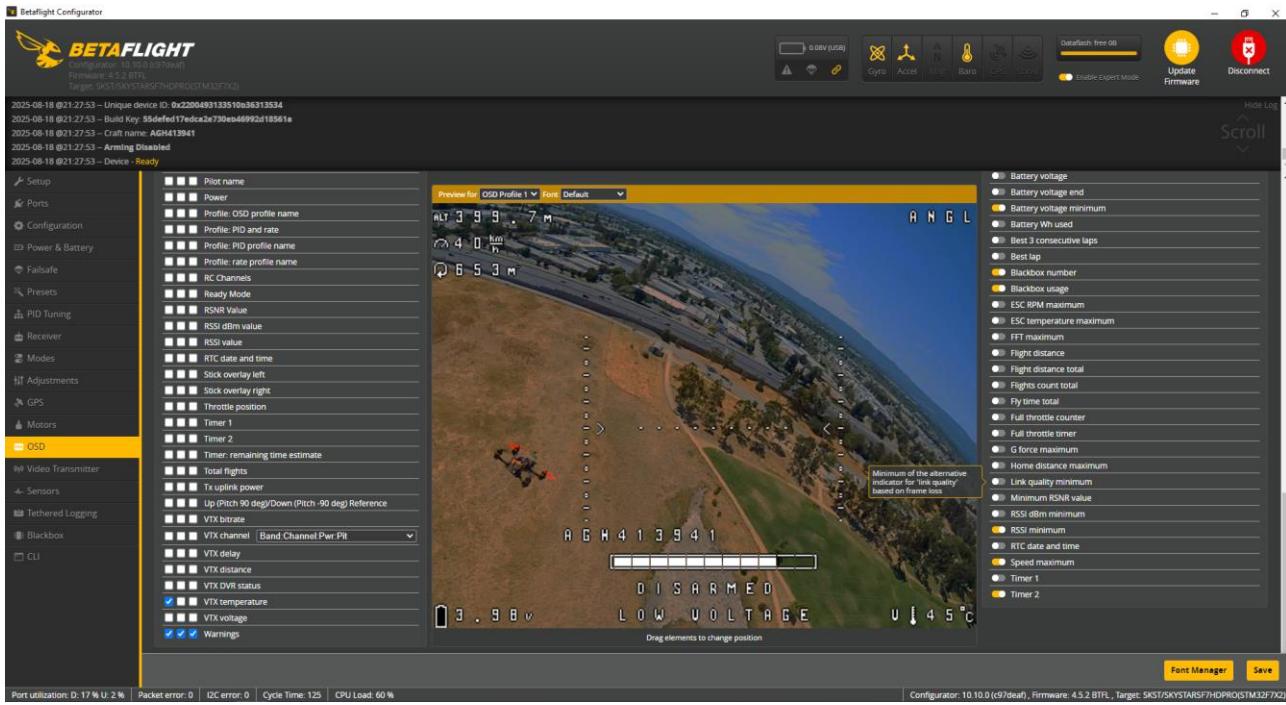
Rysunek 35 – Zrzut ekranu przedstawiający zakończoną konfigurację w zakładce „OSD” programu Betaflight Configurator – część 1
źródło: opracowanie własne



Rysunek 36 – Zrzut ekranu przedstawiający zakończoną konfigurację w zakładce „OSD” programu Betaflight Configurator – część 2
źródło: opracowanie własne



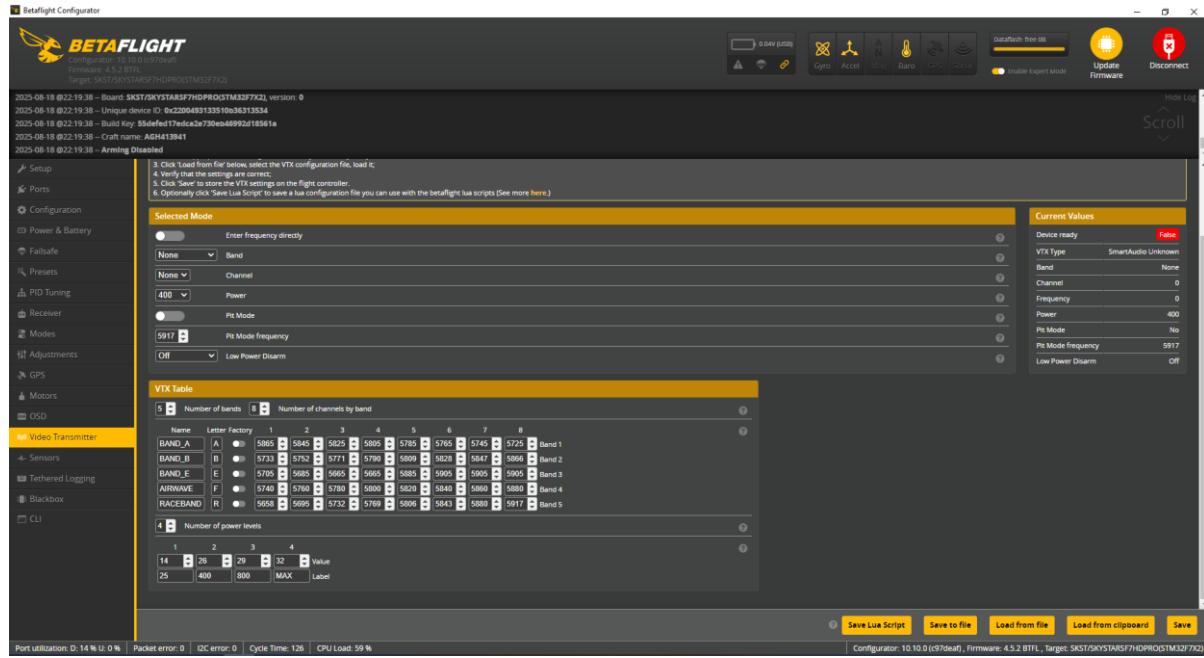
Rysunek 37 – Zrzut ekranu przedstawiający zakończoną konfigurację w zakładce „OSD” programu Betaflight Configurator – część 3
źródło: opracowanie własne



Rysunek 38 – Zrzut ekranu przedstawiający zakończoną konfigurację w zakładce „OSD” programu Betaflight Configurator – część 4
 Źródło: opracowanie własne

14) Zakładka „*Video Transmitter*” – niniejsza zakładka daje użytkownikowi możliwość dostosowania pracy nadajnika obrazu wideo znajdującego się na dronie. Zakładka pozwala dostosować takie opcje jak moc nadajnika wyrażaną w mW, częstotliwość nadawania w trybie niskiego poboru prądu (z ang. *pit mode*) oraz tabeli VTX. Wobec faktu, iż do skonstruowania pierwotnej tabeli VTX opisującej pasma i częstotliwości nadawania nadajnika, użyto skonstruowanego przez użytkownika „*Directory*” zestawu – będącego notabene przepisaniem dokładnych wartości wskazanych przez producenta w tabeli VTX podanej w instrukcji użytkownika do nadajnika^[11], a po testach w terenie okazały się one być w pełni działające i bezbłędne, wartości tych nie zmieniano. Jako moc nadajnika, zdecydowano się wybrać wartość 400mW, jako że testy wykonywane w terenie miały miejsce w niewielkim oraz średnim dystansie pomiędzy pilotem oraz trzymanym przez niego nadajnikiem RC (kontrolerem naziemnym) a dronem (odbiornikiem RC). Warto w tym miejscu ponownie nadmienić, że oprogramowanie Betaflight daje łatwą możliwość ustawienia mocy nadawania na tryb „niski” – z ang. *pit mode* – z poziomu radia naziemnego, za pomocą funkcji „*VTX PIT MODE*” w zakładce „*Modes*” i pilot powinien o tym pamiętać na każdym etapie lotu, szczególnie przed podłączeniem zasilania do maszyny przed startem z uwagi na łatwość spalenia nadajnika lub jego uszkodzenia w przypadku zbyt wysokiej temperatury ww. komponentu wynikającej z niedostatecznego wentylowania – nawet w warunkach

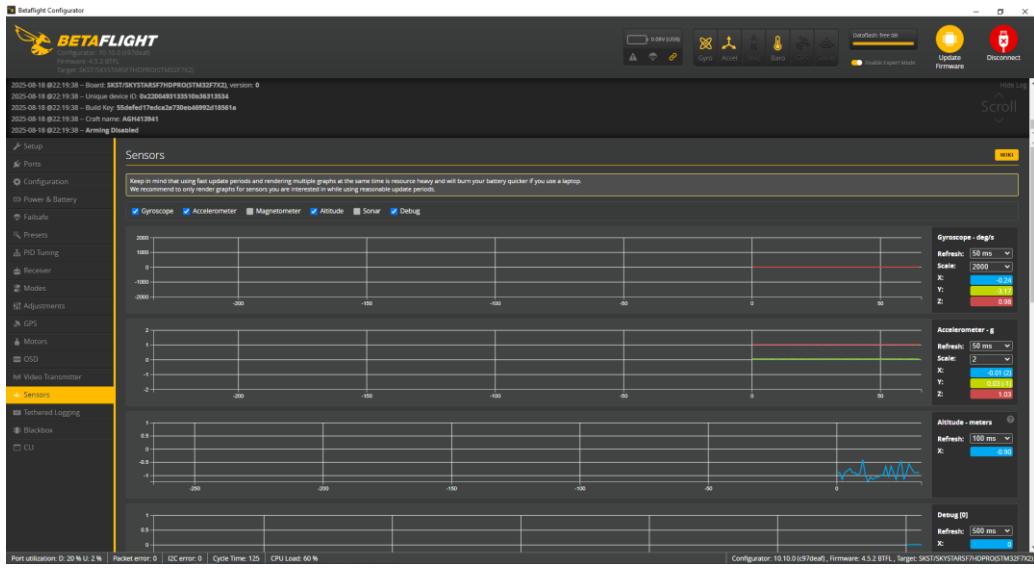
zewnętrznych, nadajnik wideo bardzo szybko i bardzo mocno się nagrzewa. Zakończoną konfigurację zaprezentowano na rysunku 39 – w przypadku tego elementu budowy drona, należy dokładnie zapoznać się z instrukcją zapewnioną do wybranego komponentu oraz z jego specyfikacją, gdyż komponent nadajnika obrazu jest najbardziej „kruchym” elementem peryferyjnym układu, a jednocześnie jednym z najważniejszych.



Rysunek 39 - Zrzut ekranu przedstawiający zakończoną konfigurację w zakładce "Video Transmitter" programu Betaflight Configurator

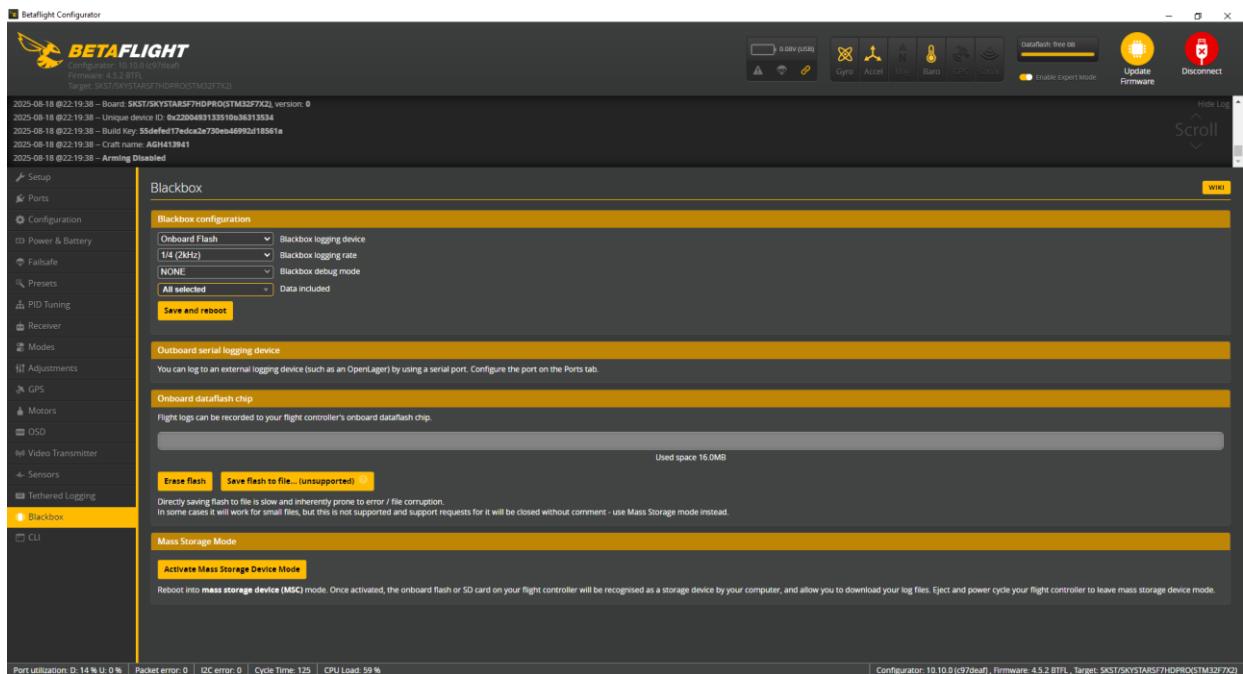
Źródło: opracowanie własne na podstawie^[33]

- 15) Zakładka „Sensors” – zakładka do podglądu na żywo wybranych sensorów obecnych w układzie maszyny. Niniejsza zakładka nie była wykorzystywana na żadnym etapie konstruowania drona wyposażonego w system wykrywania przeszkód. Fragment podglądu sensorów zaprezentowano na rysunku 40.



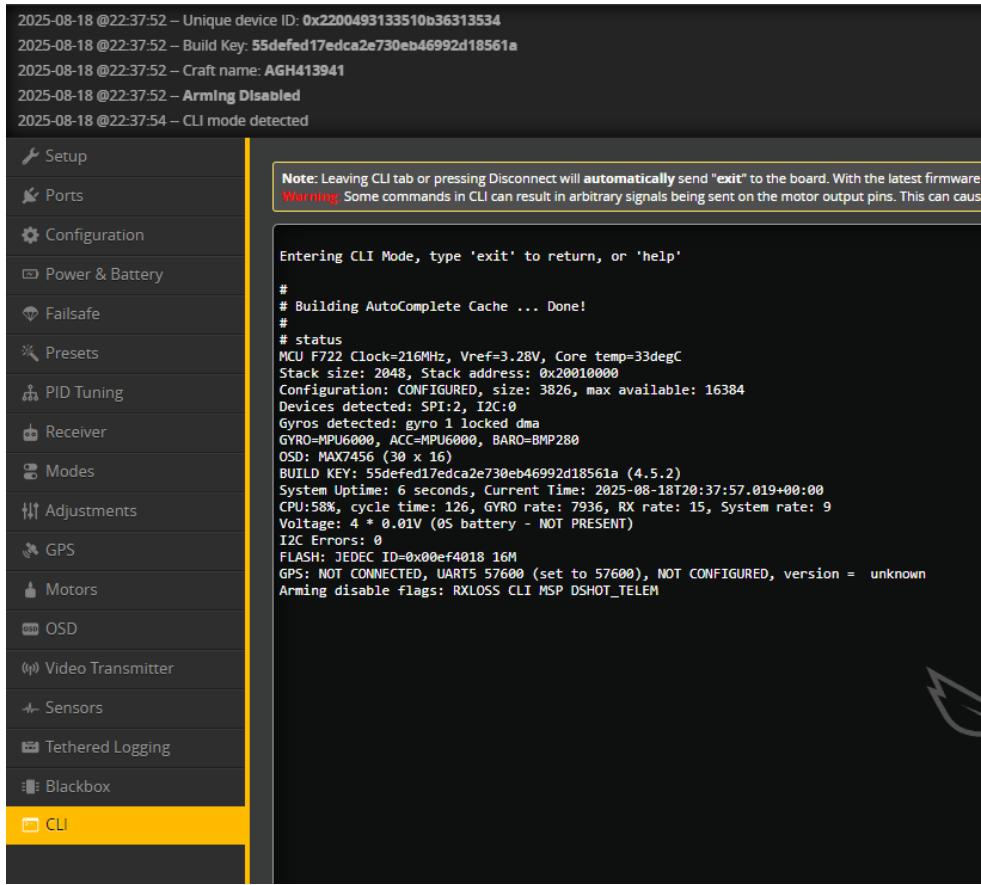
Rysunek 40 - Zrzut ekranu ukazujący fragment zakładki "Sensors" programu Betaflight Configurator
 Źródło: opracowanie własne

- 16) Zakładka „*Tethered Logging*” – jest zakładką przeznaczoną do ustawień opcji logów kontrolera lotu – jw. Również nie była wykorzystywana w opisywanym przypadku, wobec czego żadna z dostępnych w niej wartości nie została zaznaczona.
- 17) Zakładka „*Blackbox*” – daje użytkownikowi możliwość ustawienia parametrów czarnej skrzynki obecnej wewnątrz kontrolera lotu lub połączonej po porcie szeregowym karty pamięci. W tym przypadku, nie zmieniano ustawień domyślnych, z uwagi na fakt, iż zewnętrzne urządzenie z kartą pamięci do przechwytywania logów nie zostało podłączone do kontrolera lotu. Opisywana zakładka daje również możliwość podglądu aktualnego stanu pamięci kontrolera a także jej wyczyszczenie. Niniejszą zakładkę zaprezentowano na rysunku 42.



Rysunek 41 - Zrzut ekranu ukazujący możliwości zakładki "Blackbox" programu Betaflight Configurator
 Źródło: opracowanie własne

18) Zakładka „*CLI*” – jest zakładką dającą użytkownikowi dostęp do konsoli systemowej kontrolera lotu. Jest zakładką wielokrotnie wykorzystywaną w niniejszym przypadku systemu, gdyż daje szybką możliwość podglądu ewentualnych błędów oraz parametrów poszczególnych urządzeń peryferyjnych. Komendą głównie wykorzystywaną w tym przypadku było polecenie „status” zwracające na ekran konsoli podstawowe parametry konfiguracji drona, takie jak m. in. obciążenie procesora, protokół OSD, stan konfiguracji, flagi uzbrojenia czy liczba urządzeń podłączonych za pomocą protokołu SPI. Konsola w opisywanej zakładce, daje również zaawansowanym użytkownikom możliwość konfiguracji każdego z elementów drona za pomocą wiersza poleceń, bez użycia interfejsu graficznego. Jest to opcja przydatna, w przypadku dokonania kopii zapasowej istniejącej konfiguracji przed wprowadzeniem istotnych zmian, a następnie jej przywróceniem. Przykładowe wyjście polecenia „status” zaprezentowano na rysunku 42. Przykładowe flagi uzbrojenia, które są ważnym czynnikiem w przypadku debuggowania całości konfiguracji oraz ich opisy zaprezentowano w tabeli 3.



Rysunek 42 - Zrzut ekranu przedstawiający wycinek konsoli systemowej programu Betaflight Configurator z wynikiem polecenia "status" dla opisywanego systemu
 Źródło: opracowanie własne

Tabela 3 - Wybrane flagi uzbrojenia wraz z odpowiadającymi im opisami

Flaga uzbrojenia	Opis
RXLOSS	Flaga oznaczająca, iż system nie odnalazł właściwego kontrolera RC.
CLI	Flaga uniemożliwiająca uzbrojenie drona z uwagi nałączoną konsolę systemową.
MSP	Flaga uniemożliwiająca uzbrojenie drona z uwagi na aktywne połączenie MSP (w opisywanym przypadku na zasadzie komputer-kontroler lotu)

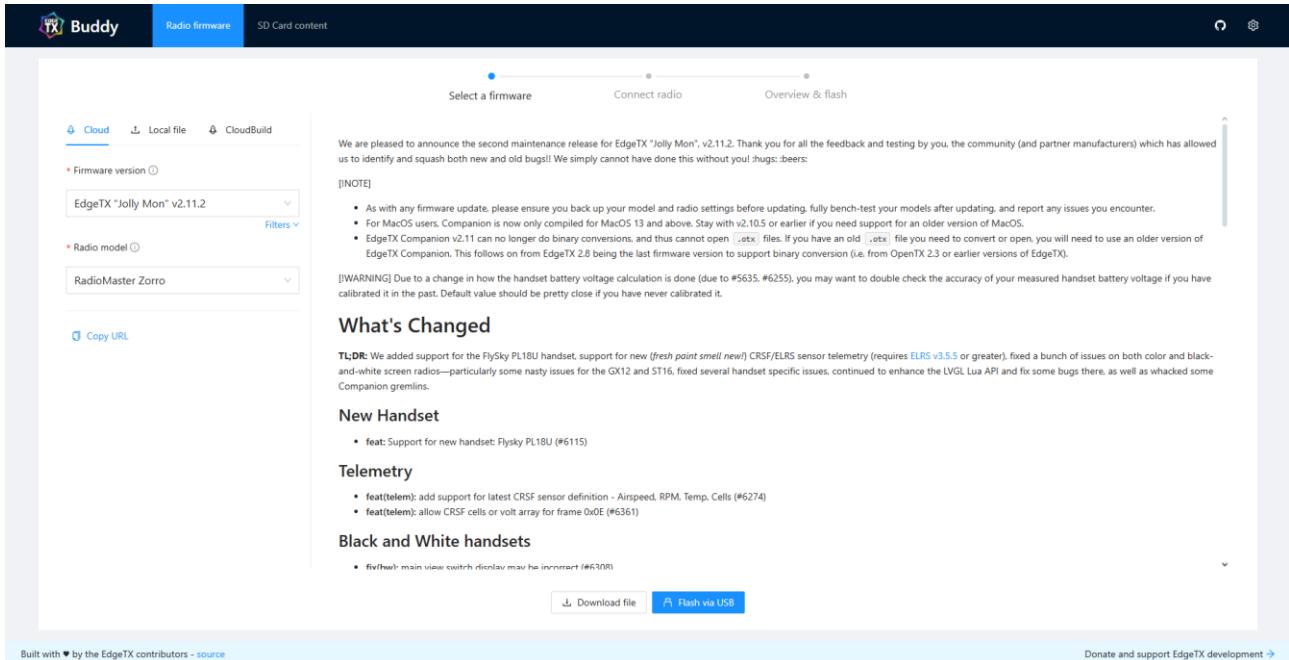
DSHOT_TELEM	Flaga uniemożliwiająca uzbrojenie z uwagi na nieaktywną funkcję filtrowania RPM z kontrolera silników
GPS	Flaga uniemożliwiająca uzbrojenie z uwagi na niespełnienie warunków odpowiedniej liczby satelit skonfigurowanej w zakładce „GPS” programu Betaflight Configurator
FAILSAFE	Flaga uniemożliwiająca uzbrojenie drona z uwagi na aktywną procedurę <i>Failsafe</i> skonfigurowaną w zakładce „ <i>Failsafe</i> ” programu Betaflight Configurator

Źródło: opracowanie własne na podstawie^[34]

Po zakończeniu konfiguracji kontrolera lotu maszyny latającej, przystąpiono do konfiguracji kontrolera RC, choć warto nadmienić, że niejednokrotnie obydwa z wymienionych procesów konfiguracyjnych się przeplatały. Było to spowodowane wymogiem wykonania takich czynności jak: parowanie nadajnika kontrolera RC z odbiornikiem obecnym na dronie, testowanie nadawania kanałów AUX, poprawne odczytywanie wartości oraz zmiana trybów przypisanych kontrolerowi lotu w zakładce „*Modes*” oraz sprawdzenie poprawności działania silników w zakładce „*Motors*” po uzbrojeniu drona.

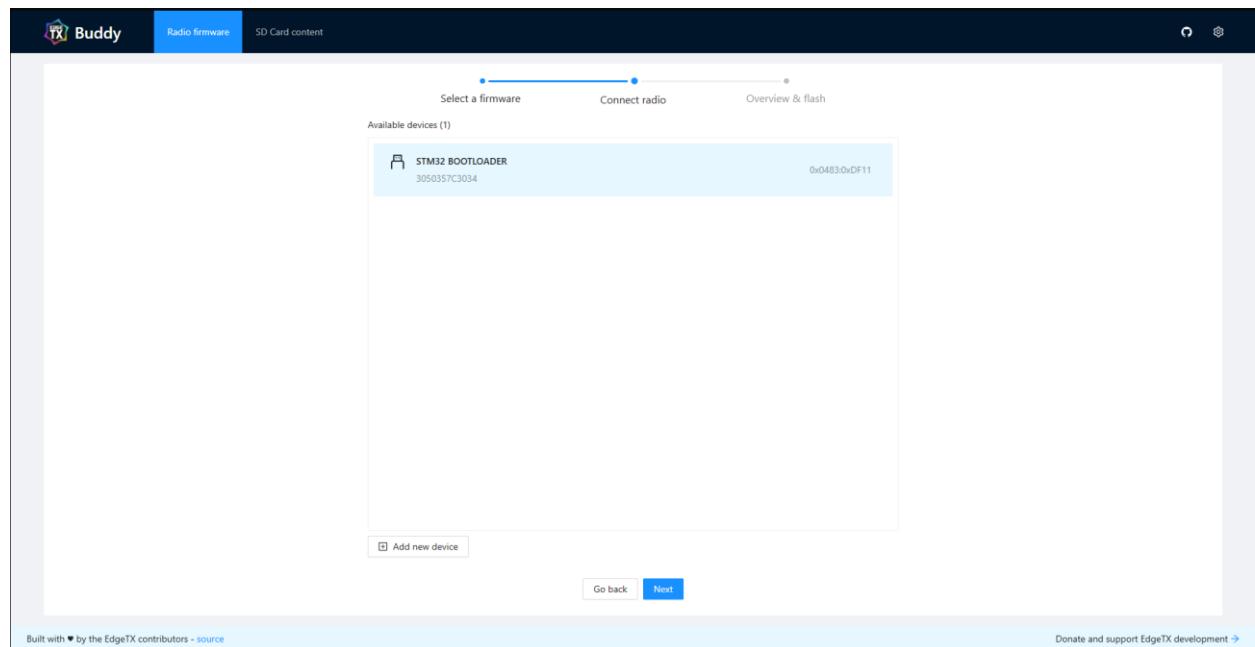
4.2. Charakterystyka i konfiguracja oprogramowania kontrolera RC

Mając przygotowany kontroler RC, włączono go sprawdzono poprawność jego działania. Z uwagi na to, że kontroler działał prawidłowo, pierwszą czynnością, którą wykonano, podobnie jak w przypadku kontrolera lotu obecnego na pokładzie maszyny latającej, było zaktualizowanie oprogramowania kontrolera oraz zaktualizowanie modułu ELRS, po uprzednim podłączeniu kontrolera do komputera PC. Jest to krok zalecany zawsze w przypadku kupna nowego kontrolera, z uwagi na nieustanne aktualizowanie przez producenta oprogramowania, co skutkuje jego poprawnym działaniem. Wobec powyższego, udało się na stronę EdgeTX Buddy po adresem buddy.edgetx.org^[35] oraz w formularzu wybrano wersję systemu, a także odpowiadający model kontrolera RC – w tym przypadku wersja oprogramowania to EdgeTX „Jolly Mon” v.2.11.2 (pozycja najwyższej dostępna na liście - najnowsza), natomiast wybrany kontroler to RadioMaster Zorro. Formularz wraz z przyciskiem pobierania ukazano na rysunku 43.



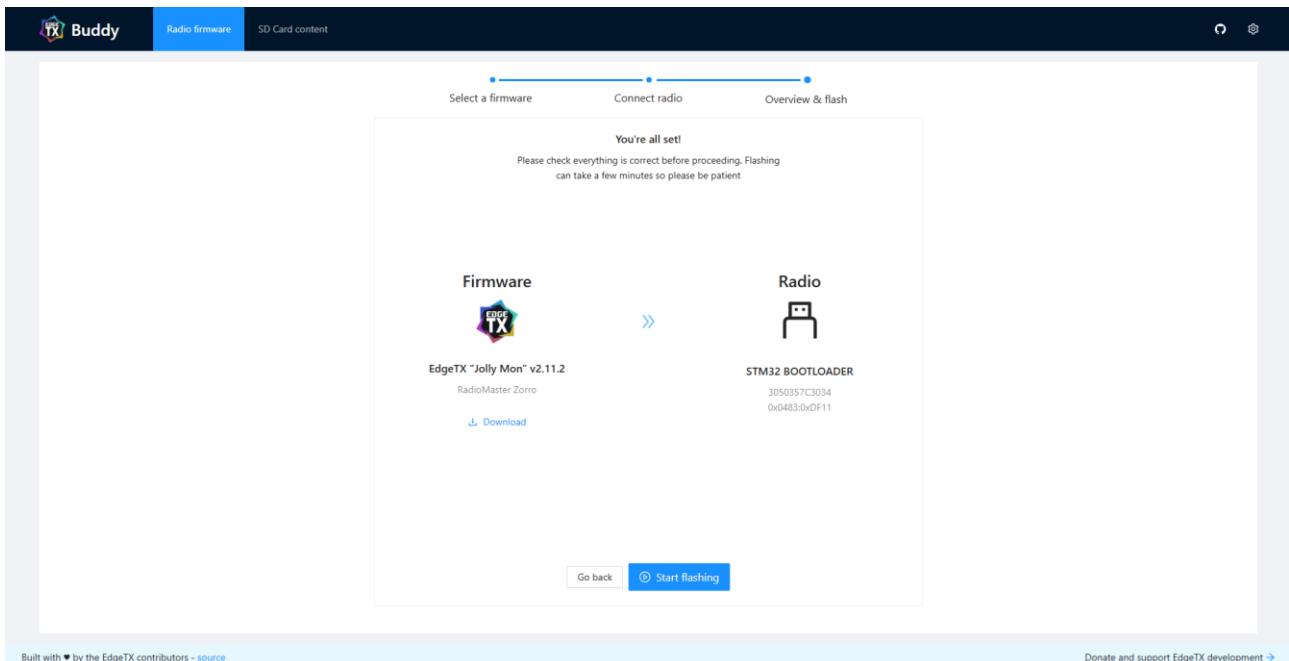
Rysunek 43 - Zrzut ekranu przedstawiający stronę aktualizacyjną oprogramowania EdgeTX z wybranym modelem kontrolera RC oraz wersją oprogramowania
 Źródło: opracowanie własne

W kolejnym kroku, należało nacisnąć przycisk „*Flash via USB*”, oraz w następnym oknie dialogowym dodać nowe urządzenie, wybierając z listy urządzenie o nazwie „STM 32 BOOTLOADER”. Podczas wykonywania niniejszego kroku, kontroler powinien być wyłączony. Proces poprawnego dodania kontrolera w oknie aktualizatora zaprezentowano na rysunku 44.



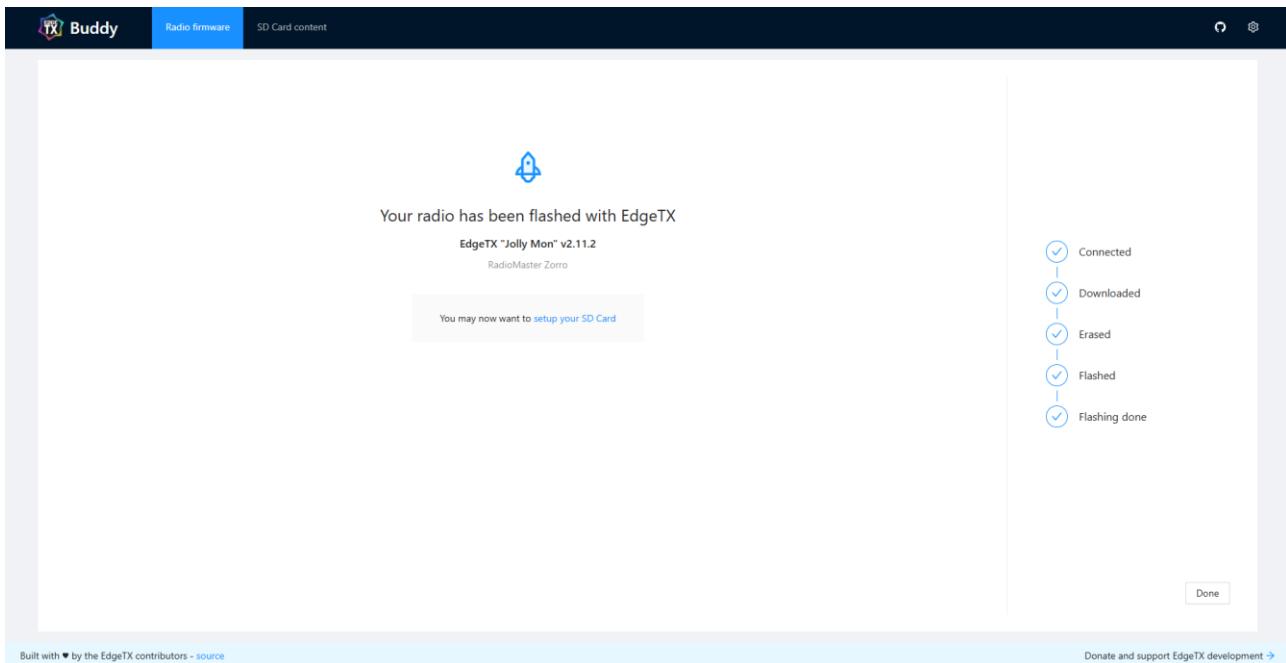
Rysunek 44 - Zrzut ekranu prezentujący poprawnie dodany kontroler RC do aktualizatora EdgeTX
 Źródło: opracowanie własne

Po naciśnięciu pokazanego w oknie przycisku „*Next*” aktualizator przenosi użytkownika do okna podsumowującego operację, która zostanie na kontrolerze przeprowadzona – wypisane są w nim wersja systemu która zostanie wgrana oraz wybrany model kontrolera, a także urządzenie, do którego oprogramowanie zostanie wgrane – niniejsze okno zaprezentowano na rysunku 45.



Rysunek 45 - Zrzut ekranu przedstawiający ostatnie okno podsumowujące przyszłą aktualizację oprogramowania EdgeTX na wybranym kontrolerze RC
 Źródło: opracowanie własne

W tym miejscu aktualizacji, należało nacisnąć przycisk „*Start flashing*” aby rozpocząć proces wgrywania oprogramowania na kontroler RC. Proces aktualizacji oprogramowania, zajmując około minuty, w tym czasie łączy się z programem rozruchowym kontrolera – z ang. *bootloader*'em, pobiera wersję oprogramowania z chmury producenta, czyści obecną na kontrolerze pamięć oraz wgrywa nowe oprogramowanie. Zakończony sukcesem proces wgrywania oprogramowania do kontrolera RC zaprezentowano na rysunku 46.



Rysunek 46 - Zrzut ekranu prezentujący poprawnie wykonany proces aktualizacji oprogramowania EdgeTX będącego systemem operacyjnym kontrolera RC
Zródło: opracowanie własne

Po naciśnięciu przycisku „Done”, aktualizator wraca do pierwotnego okna z wyborem wersji systemowej. Wobec powyższego, wyżej opisany aktualizator można zamknąć uznając, że proces aktualizacji systemu operacyjnego kontrolera lotu przebiegł pomyślnie.

W kolejnym kroku, dokonano aktualizacji wewnętrznego modułu ExpressLRS obecnego wewnętrz kontrolera RC. W tym celu, udało się do oficjalnego repozytorium ELRS dostępnego pod adresem <https://github.com/ExpressLRS/ExpressLRS-Configurator/releases>^[36] oraz pobrano najnowszą wersję konfiguratora ELRS – 1.7.9. Po zainstalowaniu konfiguratora na komputerze PC oraz jego uruchomieniu, dokonano aktualizacji w następujących po sobie krokach, na podstawie^[37]:

- 1) Zaznaczono wersję ELRS jako najnowszą, odpowiadającą 3.5.6;
- 2) Zaznaczono kategorię urządzenia jako RadioMaster 2.4 GHz;
- 3) Zaznaczono urządzenie jako RadioMaster Zorro Internal 2.4GHz TX;
- 4) Jako metodę wgrywania oprogramowania zaznaczono EdgeTX Passthrough, gdyż opisywany kontroler jest urządzeniem z wgranym systemem EdgeTX – w przeciwnym wypadku, w razie braku możliwości wgrania oprogramowania za pomocą ww. funkcji, należy postępować zgodnie z instrukcją użytkownika danego kontrolera RC, najczęściej za pomocą portu szeregowego lub WiFi;

- 5) Jako domenę regulacyjną wybrano, zgodnie z odpowiadającą kontrolerowi domenę „*2.4GHz LBT (Listen Before Talk)*”;
- 6) Włączono kontroler RC oraz podłączono go do komputera, wybierając tryb połączenia jako „*USB Serial*”;
- 7) Zainstalowano poprawny sterownik celem obsłużenia połączenia pomiędzy kontrolerem a systemem Windows za pomocą programu *Zadig*;
- 8) Naciśnięto przycisk „*Flash*” czym rozpoczęto proces wgrywania aktualnego oprogramowania do modułu ExpressLRS wewnętrz kontrolera RC.

Warto w wyżej wymienionych krokach zaznaczyć, iż ponownie posłużono się programem Zadig, gdyż system Windows, tak jak w tym przypadku, może nie widzieć podłączonego po porcie szeregowym urządzenia. W takim przypadku, odpowiedni sterownik, który podobnie jak w przypadku instalacji sterownika celem zaktualizowania kontrolera lotu, w tym przypadku jest to nie WinUSB, lecz **USB Serial (CDC)**. Bez instalacji powyższego sterownika, program ExpressLRS Configurator zwróci użytkownikowi błąd w postaci:

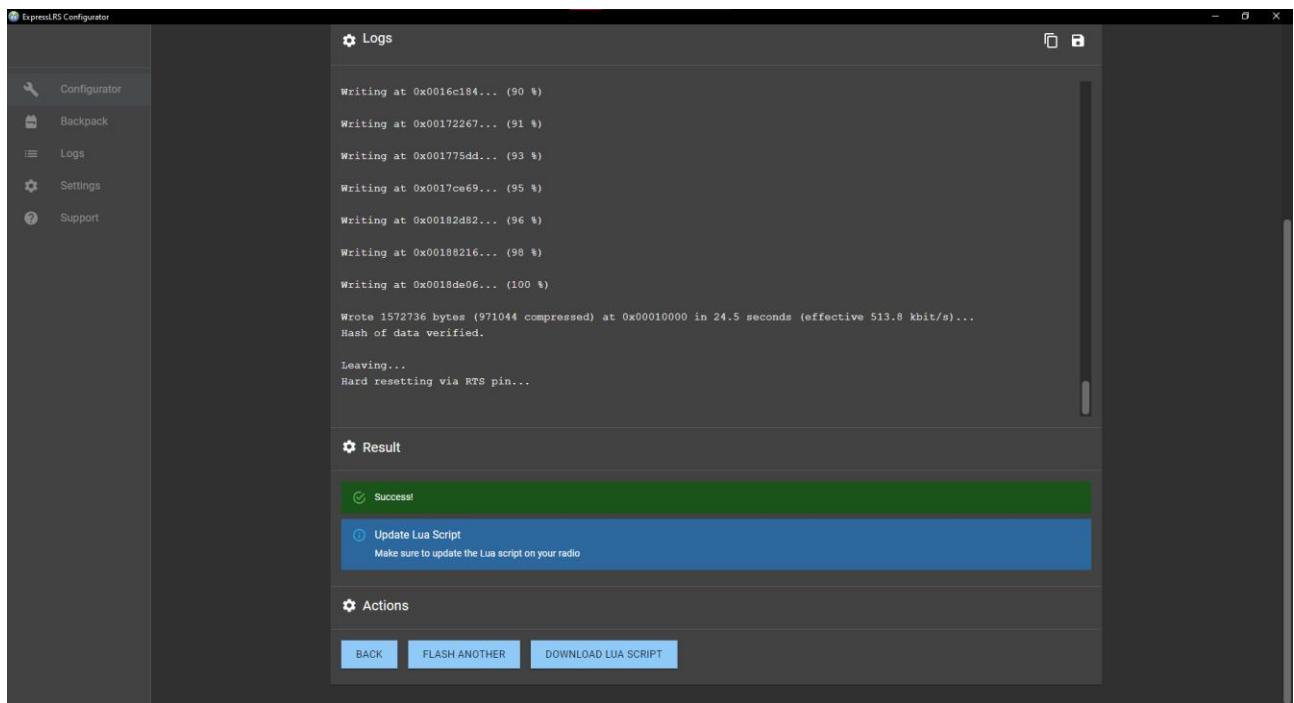
```
„DETECTED THE FOLLOWING SERIAL PORTS ON THIS SYSTEM:  
RAISE EXCEPTION('NO VALID SERIAL PORT DETECTED OR PORT ALREADY OPEN')  
EXCEPTION: NO VALID SERIAL PORT DETECTED OR PORT ALREADY OPEN”
```

lub

```
“RAISE SERIALEXCEPTION("COULD NOT OPEN PORT {!R}:  
{!R}").FORMAT(SELF.PORTSTR, CTYPES.WINERROR())  
SERIAL.SERIALUTIL.SERIALEXCEPTION: COULD NOT OPEN PORT 'COM4':  
FILENOTFOUNDERROR(2, 'THE SYSTEM CANNOT FIND THE FILE SPECIFIED.', NONE, 2)”
```

Jest to jednoznaczny komunikat informujący użytkownika o fakcie braku widoczności portu szeregowego lub braku możliwości jego otwarcia przez program.

Zakończony sukcesem proces aktualizacji modułu ELRS zaprezentowano na rysunku 47.



Rysunek 47 - Zrzut ekranu przedstawiający zakończony sukcesem proces aktualizacji modułu Express LRS wewnętrz kontrolera RC
 Źródło: opracowanie własne

W tym momencie, rozpoczęto konfigurację profilu na w pełni zaktualizowanym kontrolerze RadioMaster Zorro.

Aby zachować pełną kontrolę nad profilem użytym do celów niniejszego projektu, za pomocą obecnego na kontrolerze przycisku „MDL” stworzono całkowicie nowy profil. W pierwszej z dostępnych zakładek opracowywanego modelu, nadano mu nazwę „PRACA INZ” a następnie aktywowano protokół komunikacji CRSF. Wynik wpisanej nazwy oraz aktywacji protokołu CRSF prezentują zdjęcia 1 i 2.

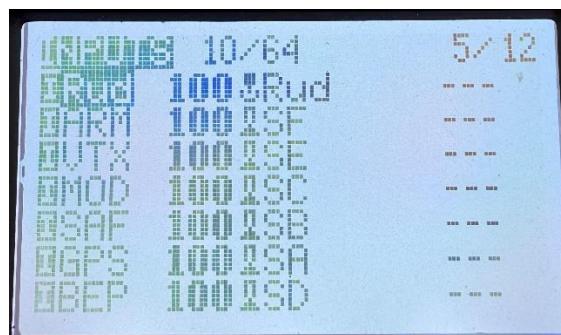


Zdjęcie 1 – zdjęcie ukazujące ekran kontrolera podczas konfiguracji modelu w zakładce „SETUP”, po wpisaniu nazwy modelu
 Źródło: opracowanie własne

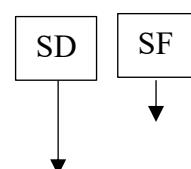


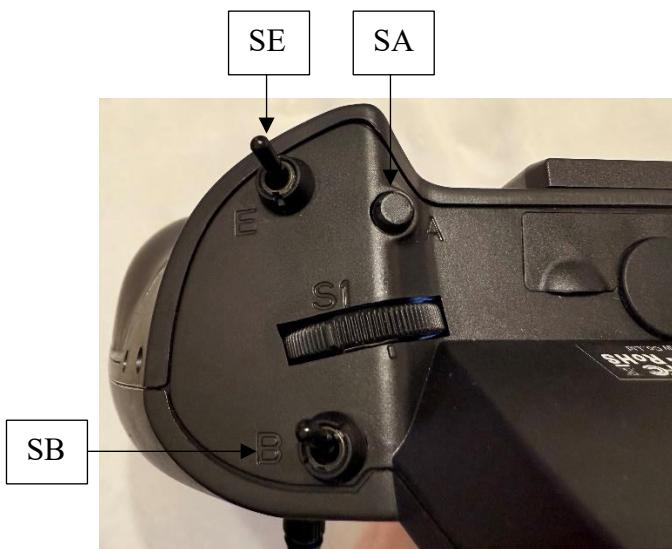
Zdjęcie 2 – zdjęcie ukazujące ekran kontrolera podczas konfiguracji modelu w zakładce „SETUP”, po aktywacji protokołu CRSF
 Źródło: opracowanie własne

Następnie, rozpoczęto konfigurację piątej z dwunastu dostępnych zakładek o nazwie „INPUTS”. Jest to zakładka przeznaczona do definiowania przycisków przesyłanych kanałami AUX, a następnie obsługiwanych przez Betaflight na kontrolerze lotu. Niejako w niej użytkownik definiuje których przycisków i/lub przełączników będzie używał w swoim zestawie. Wobec zamierzeń projektu, dodano następujące przyciski wejściowe – w nawiasach podano oznaczenia fizycznych przycisków na kontrolerze odpowiadające poszczególnym utworzonym wejściom w oprogramowaniu: ARM (SF – dwupoziomowy przełącznik), VTX (SE – dwupoziomowy przełącznik), MOD (SC – trójpoziomowy przełącznik), SAF (SB – trójpoziomowy przełącznik), GPS (SA – przycisk) oraz BEP (SD – przycisk). Na zdjęciu 3 zaprezentowano końcowy efekt konfiguracji niniejszej zakładki, natomiast na zdjęciach 4 i 5 zaprezentowano wygląd poszczególnych przełączników i przycisków obecnych fizycznie na kontrolerze, wraz z ich oznaczeniami.



Zdjęcie 3 – zdjęcie ukazujące ekran kontrolera podczas konfiguracji modelu w zakładce „INPUTS”, po poprawnym dodaniu wirtualnych wejść
 Źródło: opracowanie własne



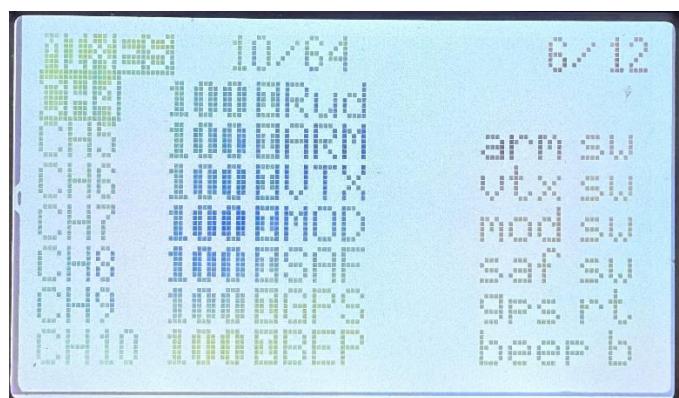


Zdjęcie 4 – zdjęcie ukazujące guzik oraz przełączniki obecne fizycznie z lewej strony kontrolera w rzucie od góry
 Źródło: opracowanie własne



Zdjęcie 5 – zdjęcie ukazujące guzik oraz przełączniki obecne fizycznie z prawej strony kontrolera w rzucie od góry
 Źródło: opracowanie własne

W następnej zakładce o nazwie „MIXES” dostępnej na kontrolerze, skonfigurowano kanały AUX które będą przesyłały stan przycisków do kontrolera lotu. Aby dodać/edytować poszczególny kanał, należy wejść w nowy (nieprzypisany), a następnie jedynie wedle uznania ustawić jego nazwę, gdyż powiązanie kanałów ze stworzonymi w uprzednim kroku wejściami następuje w przypadku wybranego kontrolera automatyczne, w kolejności wejść. Kolejno kanałom od CH5 do CH10 przypisano nazwy: CH5 – arm sw, CH6 – vtx sw, CH7 – mod sw, CH8 – saf sw, CH9 – gps rt oraz CH10 – beep b. Niestety, niemożliwym jest w niniejszym oprogramowaniu kontrolera umieszczenie pełnych nazw, z uwagi na ograniczoną ilość znaków na jeden kanał (limit wynosi 6), wobec czego posłużyono się skrótami. Niniejsze skróty zostały bardziej szczegółowo opisane w tabeli 4. Wobec powyższego kroku, otrzymano konfigurację kanałów AUX którą prezentuje zdjęcie 6.



Zdjęcie 6 – zdjęcie ukazujące ekran kontrolera podczas konfiguracji modelu w zakładce „MIXES” po poprawnym spisaniu nazw funkcji wysyłanych poprzez kanały AUX do odbiornika obecnego po stronie kontrolera lotu na dronie
 Źródło: opracowanie własne

Tabela 4 – Zestawienie poszczególnych kanałów wejściowych oraz odpowiadających im funkcji w zakładkach INPUTS oraz MIXES oprogramowania kontrolera RC

Nazwa wejścia w kanale INPUTS	Symbol odpowiadający przełącznikowi	Mögliche Werte des Schalters/Tastens	Numer kanału w zakładce MIXES	Kurzbeschreibung der Funktion in der MIXES-Karte	Pełna nazwa funkcji
ARM	SF	wysoki niski	CH5	arm sw	Arming switch
VTX	SE	wysoki niski	CH6	vtx sw	VTX switch
MOD	SC	wysoki środkowy niski	CH7	mod sw	Mode switch
SAF	SB	wysoki środkowy niski	CH8	saf sw	Safe Mode switch
GPS	SA	wysoki (domyślny) niski (wciśnięty)	CH9	gps rt	GPS Return To Home button
BEP	SD	wysoki (domyślny) niski (wciśnięty)	CH10	beep b	Beep button

Źródło: opracowanie własne

Odczytując powyższą tabelę, należy zaznaczyć, iż przełączniki odpowiadają kolejno za mechanizmy takie jak: ARM – jest przełącznikiem stanu uzbrojenia drona, a zatem jest również przełącznikiem stanu włączenia trybu „ARMED” na kontrolerze lotu, VTX – jest przełącznikiem stanu włączenia trybu „VTX PIT MODE” na kontrolerze lotu, MOD – jest przełącznikiem trybu latania, a zatem przełącznikiem stanu włączenia/wyłączenia trybów „ANGLE” oraz „HORIZON” na kontrolerze lotu, SAF – jest przełącznikiem częściowo lub całkowicie włączającym, bądź wyłączającym system kontroli przeszkód, a zatem również przełącznikiem stanu włączenia trybu „MSP OVERRIDE” na kontrolerze lotu, GPS – jest przyciskiem uruchamiającym procedurę failsafe, a zatem jest przełącznikiem stanu włączenia trybu „GPS RESCUE” na kontrolerze lotu, BEP – jest przełącznikiem stanu włączenia trybu „BEEPER” na kontrolerze lotu.

W ostatnim kroku konfiguracyjnym kontrolera, dokonano personalizacji funkcji specjalnych w zakładce 10 – „SPECIAL FUNCTIONS”. Jest to zakładka umożliwiająca użytkownikowi personalizację kontrolera pod kątem – tak jak zostało to użyte w tym przypadku – przykładowo wypowiadanych przez kontroler kwestii, po zmianie stanu przełącznika lub naciśnięciu przycisku. Producent w tym przypadku, nie ograniczał się jedynie do kwestii mówionych – za pomocą ww. zakładki, można bowiem wykonać takie czynności jak: nadpisanie poszczególnego kanału AUX, odegranie dźwięku, przeczytanie wartości, zmiana trybu wyścigowego, restart minutnika, dokonanie zrzutu ekranu, rozpoczęcie zapisywania logów, ustawienie mechanizmu failsafe, ustawienie głośności i inne. W opisywanym przypadku, użyto jedynie funkcji „Ply Trk” która jest skrótem kwestii „Play Track”, a więc w wolnym tłumaczeniu – odgrywania utworu. W przypadku kontrolera RadioMaster Zorro, takimi utworami, są tryby oraz funkcje, wobec czego ustawiono dla każdego stanu, w każdym z dodanych przełączników/guzików odgrywanie utworów za które odpowiedzialne są powyższe fizyczne przełączniki. W taki sposób, otrzymano kompletny zestaw kwestii mówionych, które w czasie rzeczywistym podczas zmiany stanu przełącznika, informują użytkownika o tym, co właśnie zostało wysłane do kontrolera lotu. Wobec powyższego, jeżeli przełącznik fizyczny o oznaczeniu SF odpowiedzialny za uzbrajanie drona znajduje się w pozycji dolnej, kontroler informuje użytkownika podając z głośnika kwestię „disarmed” – analogicznie, jeżeli przełącznik znajduje się w pozycji górnej lub zostaje do niej przełączony, kontroler odgrywa utwór „armed”, dając jasno do zrozumienia, że dron został w tym momencie uzbrojony (nawet, jeżeli po stronie kontrolera lotu przesłany sygnał na tym kanale nie został odebrany) i tak dla wszystkich stanów we wszystkich przełącznikach. Jest to praktyka ułatwiająca pilotowi latacie, dając jednocześnie informację zwrotną, co zostało naciśnięte i jaki będzie tego skutek. Zakończoną konfigurację funkcji specjalnych zaprezentowano na zdjęciach 7 i 8, natomiast szczegółowy opis odgrywanych przez kontroler kwestii prezentuje tabela 5.



Zdjęcie 7 – zdjęcie ukazujące ekran kontrolera podczas konfiguracji modelu w zakładce „SPECIAL FUNCTIONS” po poprawnym skonfigurowaniu dźwięków odgrywanych przez kontroler po naciśnięciu guzików/zmianie stanu przełączników – część 1

Źródło: opracowanie własne



Zdjęcie 8 – zdjęcie ukazujące ekran kontrolera podczas konfiguracji modelu w zakładce „SPECIAL FUNCTIONS” po poprawnym skonfigurowaniu dźwięków odgrywanych przez kontroler po naciśnięciu guzików/zmianie stanu przełączników – część 2

Źródło: opracowanie własne

Tabela 5 - zestawienie funkcji specjalnych oraz odczytywanych wartości po zmianie stanu przełącznika na kontrolerze RC - ↑ oznacza stan przełącznika w dół (od kontrolera), ↓ oznacza stan przełącznika w góre (do kontrolera)

Oznaczenie fizyczne przełącznika	Przypisana funkcja specjalna	Pełna nazwa funkcji specjalnej	Prezentowana (odgrywana) wartość
SF ↑	Ply Trk	Play Track	„Disarmed”
SF ↓			„Armed”
SE ↑			„VTX off”
SE ↓			„VTX on”
SC ↑			„Flight mode angle”
SC -			„Flight mode horizon”
SC ↓			„Flight mode acro”
SB ↑			„Buzzer off”

SB -			„Buzzer on”
SB ↓			„OSD on”
SD ↓			„Active”
SA ↓			„Return to home”
SA ↑			„Return to home stopped”

Źródło: opracowanie własne

Na tym etapie zakończono proces konfiguracyjny kontrolera RC – radia oraz sparowano go z kontrolerem lotu.

W opisywanym przykładzie, celem sparowania dwóch ww. urządzeń, podano 3-krotnie zasilanie przez bardzo krótki okres czasu (mniej niż sekundę) na kontroler lotu, co doprowadziło do przejście odbiornika RC w stan łączenia z nadajnikiem, oraz w ustawieniach kontrolera – w zakładce ExpressLRS naciśnięto na opcję „bind”. Jest to schemat działania różniący się w przypadku różnych kontrolerów RC oraz odbiorników RC, jednakże rzeczywiście bardzo podobny (element 3-krotnego podawania zasilania do kontrolera lotu, celem przejścia odbiornika w tryb parowania jest obecny w większości modeli).

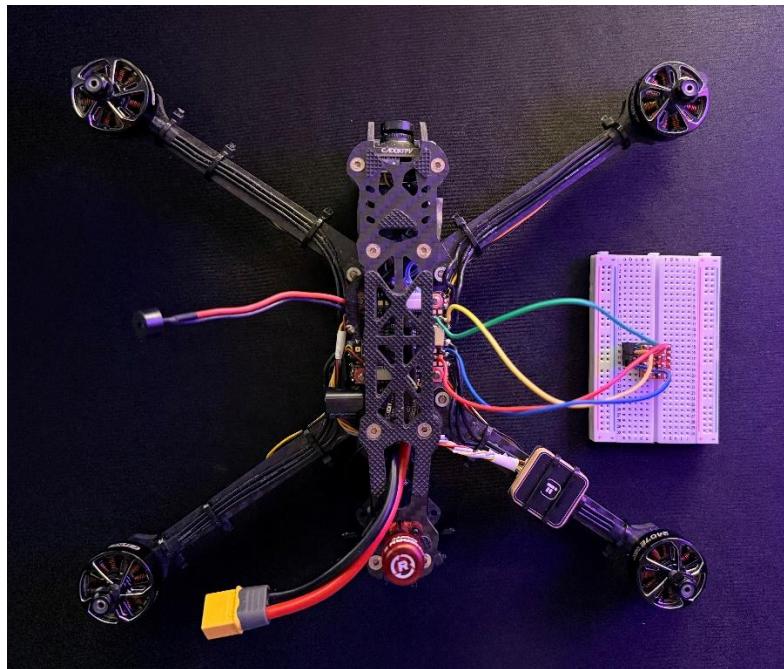
5. Implementacja systemu omijania przeszkód

5.1. Schemat połączeń i montaż komponentów jednostki latającej – drona

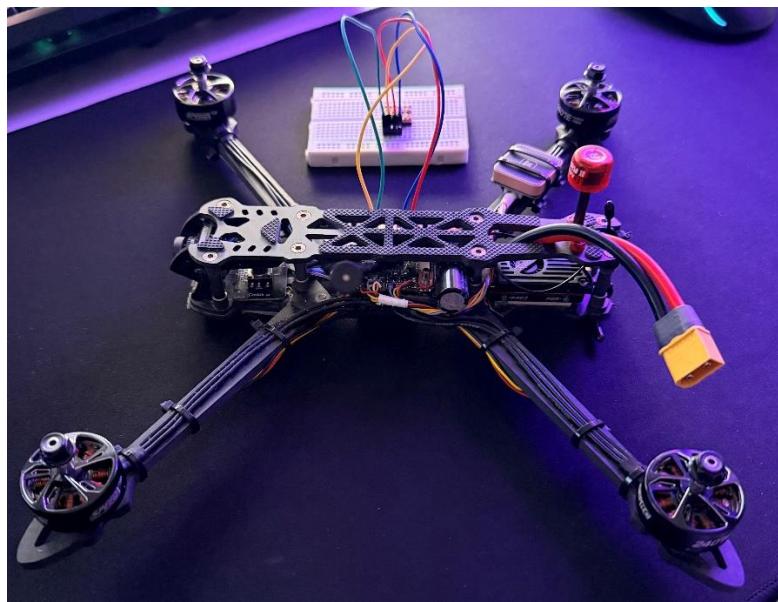
Na podstawie schematu połączeń zapewnionego przez producenta kontrolera lotu Skystars F7 22 HD Pro 4, wszystkie kolejne komponenty takie jak:

- 1) Nadajnik obrazu wideo;
- 2) GPS;
- 3) Kamera;
- 4) Odbiornik RC

zostały kolejno przymontowane do kontrolera lotu według zapewnionego przez producenta pinout'u^[29]. Silniki, stanowiące oddzielne komponenty od reszty peryferii, zostały przymontowane do kontrolera silników, również na bazie tego samego schematu dostarczanego przez producenta. Aby zrealizować ten punkt projektu, posłużyono się lutownicą oporową, cyną lutowniczą grubości 0,38 mm o składzie Sn60Pb40, topnikiem lutowniczym oraz taśmą izolacyjną. Następnie, zgodnie ze schematem 1, połączono wszystkie komponenty drona z kontrolerem lotu i rozmieszczono je w ramie. Ostateczny efekt połączeń oraz rozmieszczonych elementów na testowanym dronie, zaprezentowano na zdjęciu 9.

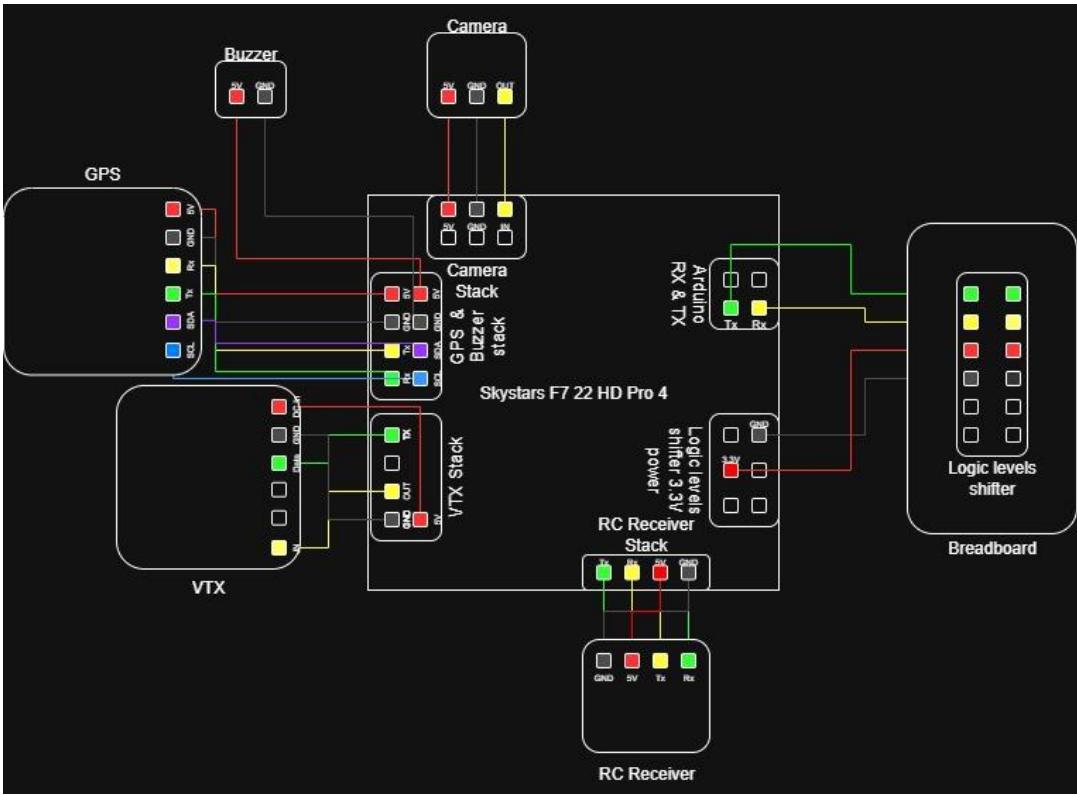


Zdjęcie 9 – zdjęcie ukazujące drona po poprawnym fizycznym podłączeniu wszystkich urządzeń peryferyjnych z kontrolerem lotu oraz płytka stykowa, w jego rzucie od góry
 Źródło: opracowanie własne



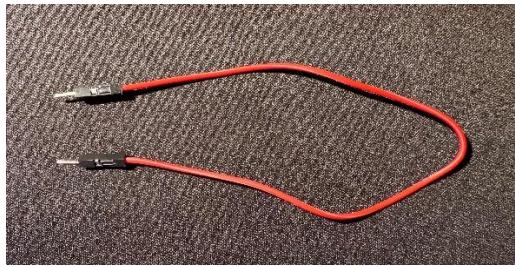
Zdjęcie 10 - zdjęcie ukazujące drona po poprawnym fizycznym podłączeniu wszystkich urządzeń periferyjnych z kontrolerem lotu oraz płytą stykową, w jego rzucie od boku z góry
 Źródło: opracowanie własne

Niniejsze zdjęcie prezentuje drona, po wykonaniu połączeń ukazanych na schemacie, jednakże nie jest to testowana wersja, gdyż jak widać na ukazanej fotografii, nie zawiera ona śmigieł. Jest to zabieg celowy, gdyż podczas testowania połączeń silników oraz innych komponentów w warunkach domowych, zalecane jest ich zdanie bądź niezakładanie na silniki. Konsekwencją założenia śmigieł w trakcie budowania systemów eksperymentalnych, może być poważne uszczerbek na zdrowiu pilota, który na danym etapie nie zweryfikował, że całość systemu działa sprawnie. Wobec powyższego, testy które miały miejsce w warunkach domowych w części dalszej niniejszego opracowania, odbywały się bez śmigieł obecnych na dronie. Zostały one założone bezpośrednio przed testami wykonywanymi na zewnątrz, co również zostało odpowiednio opisane w następnym rozdziale.



Schemat 1 – faktyczny schemat dokonanych połączeń w budowie drona
Źródło: opracowanie własne na podstawie^[29]

Schemat 1 pokazuje jasno, że w trakcie połączeń interfejsów UART zawierających przewody Rx oraz Tx, podłącza się je naprzemiennie: jeżeli jeden koniec został podłączony do portu Rx pierwszego urządzenia, drugi koniec został podłączony do portu Tx drugiego urządzenia. Jest to wymagane, gdyż linia Tx domyślnie nadaje dane, natomiast linia Rx te dane odbiera. W przypadku podłączenia linii odpowiadających sobie, otrzymano by sytuację, w której dwa urządzenia albo próbują w jednakowym czasie wysłać do siebie dane na tej samej linii (połączenia Tx-Tx), albo bez przerwy nasłuchując oczekując na dane (połączenie Rx-Rx). Jest to element charakterystyczny protokołu UART^[38]. Jedyną różnicą pomiędzy uniwersalnymi przewodami przesyłowymi w przypadku platformy kontrolera lotu, są przewody przesyłowe oraz zasilające odprowadzone do płytki stykowej (z and. *breadboard*). W tym przypadku, należało obciąć końcówkę, zdjąć końcówkę ich osłony oraz w ten sposób połączyć kontroler lotu do płytki stykowej. Poczytano powyższą operację, aby zabezpieczyć przewody przed złamaniem podczas intensywnego ich zginania i wykręcania. Wynik powyższej operacji, prezentuje zdjęcie 11a) oraz 11b), na których pokazano przewód przed dostosowaniem go do połączenia pomiędzy płytą stykową a kontrolerem lotu, oraz po.



Zdjęcie 11a) - zdjęcie przedstawiające domyślny przewód przesyłowy wykorzystywany przy systemach platformy Arduino

Źródło: opracowanie własne

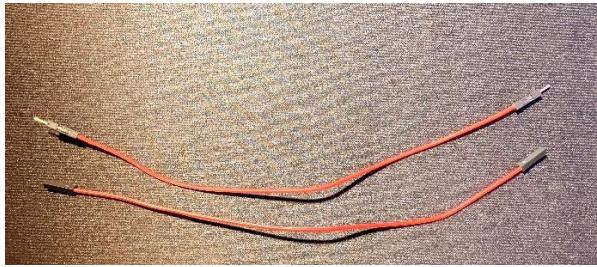


Zdjęcie 1b) - zdjęcie przedstawiające domyślny przewód przesyłowy wykorzystywany przy systemach Arduino po dostosowaniu go do możliwości podłączenia do kontrolera lotu

Źródło: opracowanie własne

5.2. Schemat połączeń i montaż komponentów platformy Arduino

Wobec faktu, iż wybrana platforma posiada wyprowadzenia z mikrokontrolera w postaci otworów przystosowanych do gotowych przewodów, przystąpiono do połączenia go z czujnikiem ultradźwiękowym oraz dronem – w tym przypadku za pośrednictwem płytki stykowej (*breadboard*). Schemat prezentujący podłączony do mikrokontrolera czujnik oraz sam kontroler do płytki stykowej zaprezentowano w schemacie 2. W tym przypadku modyfikacja dokonana na przewodach zaprezentowanych na rysunku 11a) wynikała z faktu, iż wyprowadzenia z Arduino są w wersji żeńskiej, natomiast wyprowadzenia pinów z czujnika ultradźwiękowego – w postaci męskiej. Wobec ww. okoliczności, należało dostosować przewody, aby umożliwić połączenie czujnika do platformy bezpośrednio. Aby zrealizować powyższe, połączono przewód męski oraz żeński, dokładnie w połowie. Schemat postępowania w niniejszym przykładzie zaprezentowano na zdjęciach 12a) oraz 12b). Niniejszym sposobem, wykonano przewody oznaczone na schemacie 2 kolorami : fioletowym, pomarańczowym oraz szarym – przewód GND<→GND pomiędzy czujnikiem ultradźwiękowym a platformą Arduino.



Zdjęcie 12a) - zdjęcie przedstawiające przewody połączeniowe dedykowane dla platformy Arduino - przewód męski (górny) oraz żeński (dolny)

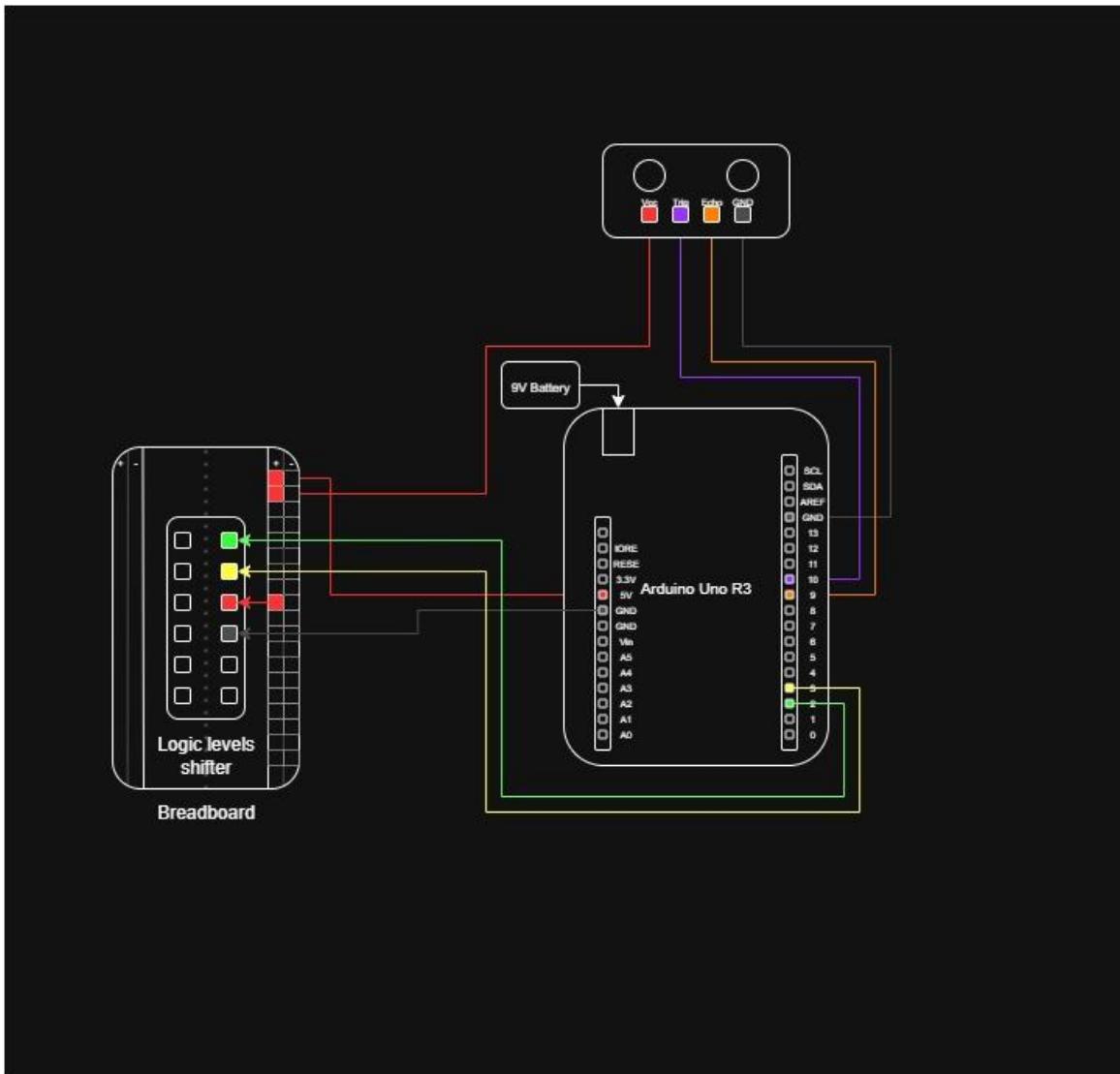
Źródło: opracowanie własne



Zdjęcie 12b) – zdjęcie przedstawiające przewód połączeniowy wykonany na potrzeby projektu, umożliwiający połączenie wyprowadzeń: męskiego (lewy koniec) z żeńskim (prawy koniec)

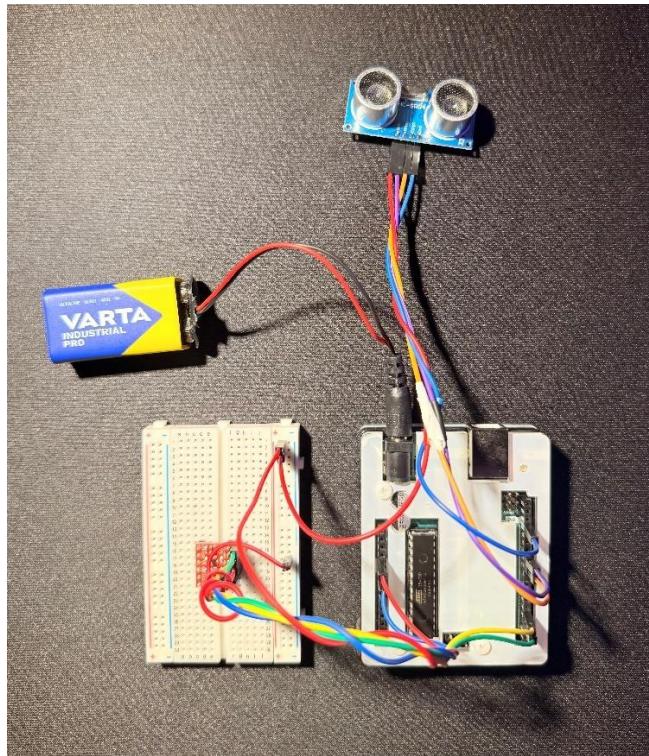
Źródło: opracowanie własne

Warto w tym miejscu zaznaczyć, iż płytka stykowa będąca komponentem pierwotnie przystosowanym dla mikrokontrolera, posiada linie zasilania (+ oraz -) po obydwu stronach płytki. Linie te, przez całą swoją długość, zachowują ciągłość ładunku elektrycznego, wobec czego, dostarczając zasilanie o napięciu 5V na jednym pinie „+”, skutkuje to możliwością zasilenia wielu komponentów, wyprowadzając przewody z pozostałych pinów należących do ww. linii. Taki schemat działania został zastosowany w przypadku wyprowadzenia złącza 5V z Arduino, poprowadzeniu go do linii „+” na płytce stykowej, a następnie wyprowadzeniu dwóch przewodów: pierwszego do czujnika ultradźwiękowego oraz drugiego do konwertera poziomów logicznych. Identycznie można zrobić w przypadku mas (GND, z ang. *ground*), jednakże zdecydowano, że nie zostanie to uczynione z uwagi na większą liczbę wyprowadzeń typu GND na mikrokontrolerze.



Schemat 2 - schemat prezentujący fizyczne połączenia pomiędzy mikrokontrolerem Arduino, a płytą stykową oraz czujnikiem ultradźwiękowym
 Źródło: opracowanie własne

Efekt wyżej opisanych operacji zaprezentowano na zdjęciu 13.



Zdjęcie 13 – zdjęcie przedstawiające połączoną platformę Arduino z czujnikiem ultradźwiękowym, płytka stykową oraz dostarczonym zasilaniem

Źródło: opracowanie własne

5.3. Implementacja systemu omijania przeszkód w programie Arduino IDE

W niniejszym miejscu pracy możliwym było napisanie kodu odpowiedzialnego za mechanizm zbierania informacji z otoczenia za pomocą zaprezentowanego w poprzednim podrozdziale systemu. Aby zrealizować niniejszy krok, należało napisać kod odpowiedzialny za: 1) wykonanie impulsu wysyłającego impuls ultradźwiękowy z czujnika HC-SR04 oraz 2) przesłanie informacji do kontrolera lotu, w przypadku, gdy przeszkoda znajduje się poniżej odległości minimalnej, wyznaczonej przez użytkownika. Wobec powyższego, napisano kod zaprezentowany i opisany w zdjęciach 14a)-14g) oraz opisany w niniejszym podrozdziale opracowania.

```

1 #include <SoftwareSerial.h>
2 #include <NewPing.h>
3 #include <stdio.h> // dla sprintf()
4
5 // Wykorzystane piny
6 #define TRIGGER_PIN      10
7 #define ECHO_PIN         9
8 #define MIN_DISTANCE_CM 200
9 #define MAX_DISTANCE_CM 300
10
11 NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE_CM);
12
13 // Częstotliwości i porty szeregowe
14 SoftwareSerial FC(2, 3);           // D2 = RX, D3 = TX
15 constexpr uint32_t MSP_BAUD = 57600;
16 constexpr uint32_t USB_BAUD = 115200;
17
18 // MSP v1
19 constexpr uint8_t MSP_API_VERSION = 1;
20 constexpr uint8_t MSP_SET_RAW_RC = 200;
21 constexpr uint8_t MSP_RC        = 105;
22 constexpr uint8_t MSP_NAME     = 10;
23 constexpr uint8_t MSP_SET_NAME = 11;
24
25 // Indeksy kanałów AUX
26 constexpr uint8_t CH_IDX_BEEPER = 9; // Brzęczyk
27 constexpr uint8_t CH_IDX_SWITCH = 7; // Przełącznik trybu bezpiecznego
28
29 // Stan
30 uint16_t rcNow[16] = {1500};
31 volatile uint16_t switchRaw = 1500;
32 enum SwitchPos { SW_DOWN, SW_MID, SW_UP };
33 volatile SwitchPos switchPos = SW_MID;
34
35 char craftName[33] = {0};
36 char origName[33] = {0};
37 bool haveOrigName = false;
38
39 // Stan OSD
40 bool osdShown     = false; // Aktywny napis "Obstacle"
41 bool osdBlankShown = false; // Aktywna pusta nazwa
42
43 // Jądro MSP

```

*Fragment kodu 2 - fragment programu zarządzającego logiką systemu wykrywania przeszkód po stronie Arduino – zmienne globalne
Zródło: opracowanie własne*

W tym fragmencie kodu w pierwszym kroku dołączono niezbędne biblioteki: *SoftwareSerial*, *NewPing* oraz *stdio* – pierwsza z wymienionych odpowiedzialna jest za programowy UART, druga do obsługi sonaru, trzecia natomiast do formatowania tekstu podczas składania komunikatu wyświetlanego w goglach FPV. Następnie zdefiniowano na stałe wartości takie jak piny do których fizycznie wpięto czujnik ultradźwiękowy oraz badane dystanse – w niniejszym przykładzie, badaną odlegością od obiektu były 2m. Maksymalny dystans ustalono natomiast po to, aby czujnik nie pracował ze zbyt małą częstotliwością pingując, a następnie oczekując na powrót fali. Następnie, w linijce 11 utworzono obiektu sonaru – jest to instancja klasy sonar, biblioteki NewPing, do której w późniejszym etapie programista może przypisać funkcję przeliczania – w tym przypadku, na centymetry.

Następnie, w linijce 14 utworzono programowy port szeregowy na pinach Arduino, które poprowadzone są do konwertera poziomów logicznych. W kolejnym etapie, zdefiniowano zmienne prędkości protokołu MSP – do komunikacji z kontrolerem lotu – oraz USB – w celu debbugowania dzięki monitorze portu szeregowego w Arduino IDE. Kolejne zmienne, zaczynając od

MSP_API_VERSION, na *MSP_SET_NAME* kończąc, odpowiadają za żądania wysyłane i odbierane do/z kontrolera lotu. *MSP_API_VERSION* mówi o tym, w jakiej wersji ramkę MSP wysyłano, *MSP_SET_RAW_RC* odpowiada za wartość nadpisywania kanałów RC, *MSP_RC* to zmienna odpowiedzialna za kod żądania bieżących kanałów RC z kontrolera lotu, *MSP_NAME* to zmienna zawierająca kod odczytu nazwy modelu z kontrolera lotu (*craftname* ustawiony w Betaflight) natomiast *MSP_SET_NAME* to przeciwnie – kod ustawienia nazwy modelu.

Kolejne zmienne zdefiniowane na tym etapie, to zmienne przechowujące numery kanałów AUX istotnych dla niniejszego projektu: kanału AUX na którym działa brzęczek oraz kanału, do którego przypisany jest przełącznik odpowiedzialny za włączenie/wyłączenie funkcji *MSP_OVERRIDE* w kontrolerze lotu. Następnie utworzono bufor na kanały RC (liczba wynosi 16 kanałów) przypisując każdemu z nich wartość 1500 (dokładnie środkowa wartość). Utworzono także zmienne *switchRaw* oraz *switchPos*, z których pierwsza odpowiada za przechowywanie ostatniej wartości kanału wyżej opisanego przełącznika, natomiast druga – za pozycję przełącznika wyliczoną progami. Ostatnimi zmiennymi utworzonymi w ArduinoIDE były bufory typu znakowego odpowiedzialne za przechowywanie nazw modelu drona, a także zmienne typu bool będące flagami stanu odczytu nazwy bazowej (*haveOrigName*) oraz napisu wyświetlanego w OSD gogli (*osdShown* i *osdBlankShown*). Po utworzeniu niezbędnych zmiennych, przystąpiono do programowania niezbędnych funkcji.

```

43 // Jądro MSP
44 void sendMspV1(uint8_t cmd, const uint8_t *payload, uint8_t len){
45     uint8_t csum = 0;
46     FC.write('$');
47     FC.write('M');
48     FC.write('<');
49     FC.write(len);
50     csum ^= len;
51     FC.write(cmd);
52     csum ^= cmd;
53     for(uint8_t i=0;i<len;i++){
54         uint8_t b = payload ? payload[i] : 0;
55         if(payload) {
56             FC.write(b);
57             csum ^= b;
58         }
59     }
60     FC.write(csum);
61 }
```

Fragment kodu 3 - fragment programu zarządzającego logiką systemu wykrywania przeszkode po stronie Arduino – funkcja
sendMspV1
 Źródło: opracowanie własne

Funkcja *sendMspV1* zaprezentowana we fragmencie kodu 3 odpowiedzialna jest za wysyłanie jednej ramki MSP w wersji v1. W niniejszym projekcie, nie było koniecznym stosowanie ramki MSP v2 z uwagi na funkcjonalność poprzednika. Funkcja przyjmuje 3 argumenty: pierwszy będący kodem

wywoływanego polecenia (czyli jedną ze stałych symbolicznych: *MSP_SET_RAW_RC*, *MSP_RC*, *MSP_NAME* lub *MSP_SET_NAME*), drugi będący wskaźnikiem bufora ładunku a trzeci będący rozmiarem bufora w bajtach. Następnie, wewnątrz funkcji tworzona jest suma kontrolna i rozpoczęte jest wysyłanie ramki MSP składającej się kolejno z: nagłówka, określającego wybrany protokół i kierunek przesyłania danych, jednobajtowego pola *len*, będącego długością ładunku, jednobajtowego pola *cmd* będącego kodem polecenia, opcjonalnego *payloadu* o długości *len* i końcowego bajtu sumy kontrolnej, liczonej jako operacja XOR wszystkich bajtów.

```

63 inline void mspRequest(uint8_t cmd){
64     sendMspV1(cmd, nullptr, 0);
65 }
66

```

Fragment kodu 4 - fragment programu zarządzającego logiką systemu wykrywania przeszkode po stronie Arduino – funkcja mspRequest

Źródło: opracowanie własne

Niniejsza funkcja, ukazana we fragmencie kodu 4 upraszcza następuje funkcję *sendMspV1* w przypadku wysyłania żądania bez danych – dla poleceń *MSP_RC* oraz *MSP_NAME*.

```

67 static const uint8_t MAX_MSP_PAYLOAD = 64;
68
69 bool readOneMspFrame(uint8_t &cmd, uint8_t *payload, uint8_t &len){
70     while (FC.available()){
71         if (FC.peek() != '$'){
72             FC.read();
73             continue;
74         }
75         if (FC.available() < 3)
76             return false;
77         FC.read();
78         if (FC.read()!='M')
79             continue;
80         char dir = FC.read();
81         if (dir != '>' && dir != '!') {
82             continue;
83         }
84         while (FC.available() < 2)
85             return false;
86         uint8_t frameLen = FC.read();
87         uint8_t frameCmd = FC.read();
88         while (FC.available() < (int)frameLen + 1)
89             return false;
90         uint8_t csum = 0;
91         csum ^= frameLen;
92         csum ^= frameCmd;
93         uint8_t bToCopy = 0;
94         if(payload != nullptr) {
95             if(frameLen < MAX_MSP_PAYLOAD)
96                 bToCopy = frameLen;
97             else
98                 bToCopy = MAX_MSP_PAYLOAD;
99         }
100        for(uint8_t i=0;i<bToCopy;i++){
101            uint8_t payloadB = FC.read();
102            payload[i] = payloadB;
103            csum ^= payloadB;
104        }
105        for(uint8_t i=bToCopy;i<frameLen;i++){
106            uint8_t skippedB = FC.read();
107            csum ^= skippedB;
108        }
109        uint8_t recvCsum = (uint8_t)FC.read();
110        if(recvCsum != csum)
111            continue;

```

Fragment kodu 5 - fragment programu zarządzającego logiką systemu wykrywania przeszkode po stronie Arduino – funkcja readOneMspFrame – część I

Źródło: opracowanie własne

```

110     |     continue;
111     |     if(dir == '!')
112     |     continue;
113     |     cmd = frameCmd;
114     |     len = bToCopy;
115     |     return true;
116   }
117   |     return false;
118 }
119

```

Fragment kodu 6 - fragment programu zarządzającego logiką systemu wykrywania przeszkode po stronie Arduino – funkcja
readOneMspFrame – część 2
 Źródło: opracowanie własne

Fragmenty kodu 5 oraz 6, dotyczą funkcji *readOneMspFrame*. Funkcja *readOneMspFrame* została stworzona jako parser, na potrzeby odbioru jednej ramki z łącza kontroler lotu→Arduino. Rozpoczyna swoje działanie i jednocześnie działa tak długo, jak długo są bajty na porcie z kontrolerem lotu. Jeżeli bajtów nie ma – kontroler lotu nie zwraca żadnych danych – funkcja zwraca wartość *falsz*. W przeciwnym wypadku, po sprawdzeniu wszystkich własności ramki MSP v1 oraz tego, czy ramka jest kompletna, funkcja zwraca odpowiedź *prawda*. W niniejszej funkcji wejściem są: identyfikator komendy, bufor na dane oraz liczba skopiowanych znaków. Funkcja została wyposażona w mechanizm liczenia sumy kontrolnej i porównywania jej z otrzymaną i na tej bazie również zwraca odpowiedź – nie tylko w przypadku, gdy ramka jest pusta.

```

120 void pollMsp(){
121     static uint8_t payload[64]; uint8_t cmd,len;
122     while (readOneMspFrame(cmd,payload,len)){
123         if (cmd == MSP_RC){
124             uint8_t n = len/2;
125             if (n>16)
126                 n=16;
127             for(uint8_t i=0;i<n;i++){
128                 rcNow[i] = (uint16_t)payload[2*i] | ((uint16_t)payload[2*i+1] << 8);
129             }
130             switchRaw = rcNow[CH_IDX_SWITCH];
131             if (switchRaw < 1300)
132                 switchPos = SW_DOWN;
133             else if (switchRaw > 1700)
134                 switchPos = SW_UP;
135             else
136                 switchPos = SW_MID;
137         }
138         else if (cmd == MSP_NAME){
139             uint8_t n = len;
140             if (n>32)
141                 n=32;
142             memcpy(craftName, payload, n);
143             craftName[n]='\0';
144             if (!haveOrigName){
145                 strcpy(origName, craftName);
146                 haveOrigName=true;
147             }
148             Serial.print(F("[MSP] Craft name: "));
149             Serial.print(craftName);
150         }
151     }
152 }
153

```

Fragment kodu 7 - fragment programu zarządzającego logiką systemu wykrywania przeszkode po stronie Arduino – funkcja *pollMsp*
 Źródło: opracowanie własne

Funkcja *pollMsp* została zaprojektowana w celu przetwarzania dostępnych ramek MSP v1 odbieranych z kontrolera lotu. Funkcja nie zawiera parametrów wejściowych, gdyż aktualizuje stan globalny zmiennych *rcNow*, *switchRaw*, *switchPos*, *craftName*, *origName* oraz *haveOrigName*. Opisywana funkcja w pętli wywołuje *readOneMspFrame* i obsługuje kolejne ramki. Funkcja dla komendy *MSP_RC* dekoduje payload do *rcNow* a następnie na podstawie *rcNow[CH_IDX_SWITCH]* wyznacza pozycję przełącznika (*switchPos*) progami 1300/1700. Dla komendy *MSP_NAME* natomiast, kopiuje do *craftName* nazwę modelu wpisaną w konfiguratorze Betaflight (mając na względzie maksymalną liczbę znaków wynoszącą 32), sprawdzając czy nastąpił pierwszy odczyt. W tym przypadku, funkcja zapisuje kopię do *origName* i ustawia flagę zapisania pierwotnej nazwy *haveOrigName* jako prawdziwą, logując nazwę na port szeregowy USB celem umożliwienia jej sprawdzenia.

```

154 // Funkcje pomocnicze
155 void sendBeeperAUX(uint16_t val){           // val 1000..2000
156     uint8_t buf[32];
157     uint16_t ch[16];
158     for(uint8_t i=0;i<16;i++){
159         ch[i]=1500;
160         ch[CH_IDX_BEEPER] = val;
161         // pack LE
162         for(uint8_t i=0;i<16;i++){
163             buf[2*i] = ch[i] & 0xFF;
164             buf[2*i+1] = ch[i]>>8;
165         }
166         sendMspV1(MSP_SET_RAW_RC, buf, sizeof(buf));
167     }
168
169 void setCraftName(const char* s){
170     // max 32 znaki
171     uint8_t len = 0;
172     while (s[len] && len<32)
173         len++;
174     uint8_t csum = 0;
175     FC.write('$');
176     FC.write('M');
177     FC.write('<');
178     FC.write(len);
179     csum ^= len;
180     FC.write(MSP_SET_NAME);
181     csum ^= MSP_SET_NAME;
182     for(uint8_t i=0;i<len;i++){
183         uint8_t b=(uint8_t)s[i];
184         FC.write(b);
185         csum ^= b;
186     }
187     FC.write(csum);
188 }
189

```

Fragment kodu 8 - fragment programu zarządzającego logiką systemu wykrywania przeszkode po stronie Arduino – funkcje pomocnicze część 1 – funkcje: sendBeeperAUX oraz setCraftName
Źródło: opracowanie własne

Funkcje zaprezentowane we fragmencie kodu 8, rozpoczynają tzw. funkcje pomocnicze, bez których napisanie programu byłoby możliwe, jednakże mniej czytelne. Pierwsza z nich, *sendBeeperAUX* służy do wstrzyknięcia do kontrolera lotu wartości kanałów RC poleceniem

MSP_SET_RAW_RC. Funkcja ustawia wszystkie kanały na 1500 µs (wszystkich kanałów jest 16), z wyjątkiem kanału, dla którego przypisany jest brzęczyk. W tym przypadku, funkcja przyjmuje wartość wejściową *val* która w dalszej części kodu pochodzi z czujnika. Ostatecznie funkcja pakuje dane do 32-bajtowego bufora i wysyła je ramką MSP za pomocą komendy *MSP_SET_RAW*.

Funkcja *setCraftName* natomiast, przyjmuje jako wejście ciąg znaków, będący domyślnie: komunikatem „OBSTACLE(<MIN_DISTANCE_CMcm)” gdzie w miejsce *MIN_DISTANCE_CM* wstawia wartość zdefiniowaną na samym początku programu (minimalna odległość od przeszkody), lub pustym wejściem. W kolejnym kroku, funkcja oblicza długość komunikatu po czym buduje ramkę MSP v1 ze wszystkimi jej elementami.

```

190 ✓ void setCraftNameBlank(){           // „pusta” nazwa - bezpiecznie jedna spacja
191     setCraftName(" ");
192     osdBlankShown = true;
193     osdShown = false;
194 }
195
196 ✓ void setCraftNameObstacle(){        // dynamicznie: „OBSTACLE (<XXcm)!”
197     char msg[33];
198     snprintf(msg, sizeof(msg), "OBSTACLE (<%ucm)!", (unsigned)MIN_DISTANCE_CM);
199     setCraftName(msg);
200     osdShown = true;
201     osdBlankShown = false;
202 }
203
204 ✓ void tickRequestRC(){              // pytaj o RC ~20 Hz
205     static uint32_t last=0; uint32_t now=millis();
206     if (now-last >= 50){
207         mspRequest(MSP_RC);
208         last=now;
209     }
210 }
211 ✓ void requestOrigNameOnce(){
212     static bool asked=false;
213     if (!asked){
214         mspRequest(MSP_NAME);
215         asked=true;
216     }
217 }
218
219 // SETUP
220 ✓ void setup(){
221     Serial.begin(USB_BAUD);
222     delay(200);
223     FC.begin(MSP_BAUD);
224     delay(300);
225     requestOrigNameOnce();
226     Serial.println(F("Ready."));
227 }
228

```

Fragment kodu 9 - fragment programu zarządzającego logiką systemu wykrywania przeszkód po stronie Arduino – funkcje pomocnicze część 2 – funkcje: *setCraftNameBlank*, *setCraftNameObstacle*, *tickRequestRC*, *requestOrigNameOnce* oraz funkcja specjalna *setup*

Źródło: opracowanie własne

Funkcje *setCraftNameBlank* oraz *setCraftNameObstacle* implementują wywołania funkcji *setCraftName* w zależności od sytuacji: pierwsza z opisywanych jako ciąg znaków podaje jedynie spację (dla bezpieczeństwa), druga natomiast sformatowany komunikat o przeszkozie uwzględniając jej odległość. Obydwie funkcje w zależności od swojego działania, zmieniają flagi stanu OSD.

Następna z opisywanych funkcji, *tickRequestRC* realizuje cykliczność odpytywania kontrolera lotu o wartości kanałów RC za pomocą funkcji *mspRequest* z komendą *MSP_RC*. Częstotliwość niniejszej cykliczności została ustalona na 20 Hz (50 ms przerwy). Funkcja posiada również mechanizm aktualizowania wewnętrznego znacznika czasu za pomocą funkcji *millis()*.

Ostatnia z funkcji pomocniczych – *requestOrigNameOnce* – jednorazowo wysyła żądanie *mspRequest* z komendą *MSP_NAME* celem pobrania oryginalnej nazwy modelu. Wspomniana funkcja została wyposażona w wewnętrzną flagę celem jednokrotnego zapytania, natomiast odczytana wartość i jej zapamiętanie następuje później w funkcji *pollMsp*.

Ostatnią zaprezentowaną we fragmencie kodu 9 jest funkcja *setup()*. Jest to funkcja specjalna, jednorazowo wywoływana podczas uruchamiania programu napisanego w Arduino IDE. Może być użyta do inicjalizacji zmiennych, rozpoczęcia pracy przy bibliotekach, rozpoczęcia pracy z trybami pinów i inne^[39]. W niniejszej pracy, w funkcji inicjalizującej wywołano zarówno otwarcie portów szeregowych: zarówno portu wykorzystującego komunikację pośrednio z kontrolerem lotu, jak i port szeregowy USB w momencie, w którym jest podłączone, jak i również funkcję pobrania oryginalnej nazwy modelu drona. Otwieranie portów w niniejszym przykładzie następuje z opóźnieniem zapewniając enumerację interfejsu USB i gotowość kontrolera lotu do przyjęcia pierwszych ramek MSP.

```

229 void loop(){
230     // 1) Odbieranie MSP (RC + nazwa)
231     tickRequestRC();
232     pollMsp();
233
234     // 2) Pomiar odległości
235     static uint32_t lastPing=0;
236     uint32_t now=millis();
237     static bool obstacle=false, obstaclePrev=false;
238     if (now - lastPing >= 50){           // ~20 Hz
239         unsigned int us = sonar.ping_median(5);
240         unsigned int dist = sonar.convert_cm(us);
241         obstaclePrev = obstacle;
242         obstacle = (dist>0 && dist < MIN_DISTANCE_CM);
243         lastPing = now;
244     }
245
246     // 3) Reakcja na ZMIANĘ pozycji przełącznika (edge)
247     static SwitchPos prevPos = SW_MID;
248     if (switchPos != prevPos) {
249         // wchodzimy z UP -> przywróć oryginalną nazwę
250         if (prevPos == SW_UP) {
251             if (haveOrigName)
252                 setCraftName(origName);
253             osdShown = false;
254             osdBlankShown = false;
255         }
256         // wchodzimy w UP -> natychmiast „wyczyść” OSD
257         if (switchPos == SW_UP) {
258             if (!haveOrigName)
259                 requestOrigNameOnce();
260             setCraftNameBlank(); // pusta nazwa od razu po przełączeniu na UP
261         }
262         prevPos = switchPos;
263     }
264
265     // 4) Logika w zależności od pozycji przełącznika
266     static uint32_t lastMSP=0;
267     if (switchPos == SW_DOWN){
268         // Override wyłączony
269         if ((osdShown || osdBlankShown) && haveOrigName){
270             setCraftName(origName);
271             osdShown=false;
272             osdBlankShown=false;
}

```

Fragment kodu 10 - fragment programu zarządzającego logiką systemu wykrywania przeszkód po stronie Arduino – główna pętla programu – funkcja loop – część 1

Zródło: opracowanie własne

```

273     }
274
275 } else {
276     // MID/UP: override w BF jest aktywny – podtrzymuj AUX Beepera wg obstacle (~25 Hz)
277     if (now - lastMSP >= 40){
278         sendBeeperAUX(obstacle ? 2000 : 1000);
279         lastMSP = now;
280     }
281
282     if (switchPos == SW_UP){
283         // w UP: przy przeszkodzie -> OBSTACLE; bez przeszkody -> pusta
284         if (obstacle && !osdshown){
285             setCraftNameObstacle();
286         } else if (!obstacle && !osdBlankShown){
287             setCraftNameBlank();
288         }
289     } else { // SW_MID
290         // w MID chcemy oryginalną nazwę (brak czyszczenia)
291         if ((osdshown || osdBlankShown) && haveOrigName){
292             setCraftName(origName);
293             osdShown=false;
294             osdBlankShown=false;
295         }
296     }
297 }
298
299
}

```

Fragment kodu 11 - fragment programu zarządzającego logiką systemu wykrywania przeszkód po stronie Arduino – główna pętla programu – funkcja loop – część 2

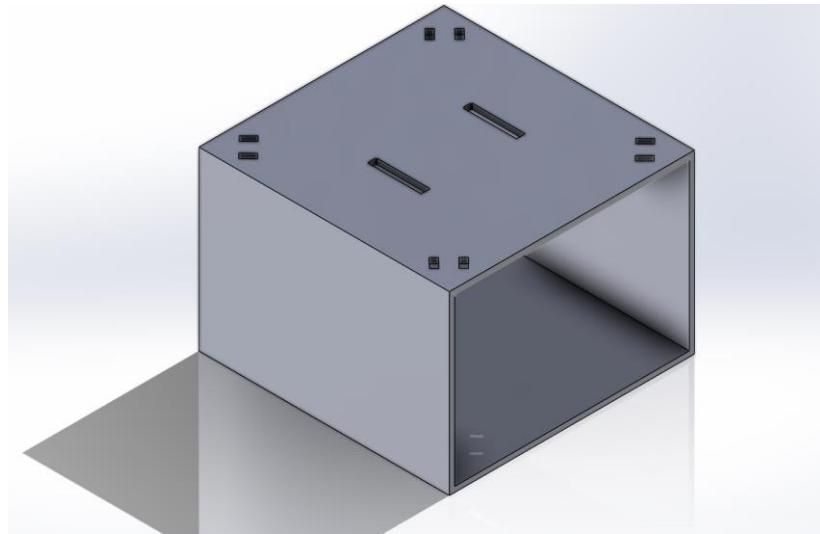
Zródło: opracowanie własne

Funkcja *loop* to odpowiednik powszechnie stosowanej nazwy pętli głównej programu – *main*. W zaprezentowanym systemie, w pętli realizowana jest na początku komunikacja MSP, co ok. 50 ms odczytując dostępne ramki i aktualizując stan wejść RC, surową wartość przełącznika odpowiadającego za pracę systemu oraz jego pozycję. W przypadku ustawienia przełącznika do pozycji „w góre” – na wartość $\approx 2000 \mu\text{s}$ – również odczytywana jest nazwa modelu drona zapisana w konfiguracji oprogramowania Betaflight. Równolegle, z podanym odstępem czasowym realizowany jest pomiar odległości sonarem, z ograniczeniem błędu za pomocą zastosowania mediany z pięciu pomiarów – *ping_median(5)* – oraz konwersją do centymetrów. Na podstawie opisanego pomiaru, ustawiana jest flaga przeskody. Kod reaguje na zmianę pozycji przełącznika wspomagania w sposób następujący: przy wejściu do pozycji górnej (zakładamy pozycję wszystkich przełączników jako bezpieczną – „w dół”), jeśli nie pobrano jeszcze nazwy modelu, wysyłane jest jednorazowe żądanie za pomocą funkcji *requestOrigNameOnce* i komendy *MSP_NAME* a następnie, natychmiast ustawiana jest pusta zawartość (jedynie spacja) nazwy modelu widoczna w OSD gogli FPV. W przypadku zmiany pozycji przełącznika z górnej na środkową/dolną, przywracana jest pierwotna nazwa drona a flagi stanu zmiany OSD – *osdShown* i *osdBlankShown* – przywracane są do wartości domyślnej. Całość logiki programu uzależniona jest nadto, o odczyt pozycji przełącznika wspomagania: jeżeli przełącznik jest w pozycji *SW_DOWN* ($\approx 1000 \mu\text{s}$ w Betaflight, „w dół” fizycznie na kontrolerze RC), nadpisywanie kanałów RC jest nieaktywne (funkcja *MSP_OVERRIDE* w Betaflight jest wyłączona) a jeżeli OSD było wcześniej modyfikowane i znana jest nazwa oryginalna (*origName* ma przypisaną wartość) przywracana jest nazwa pierwotna. Jeżeli przełącznik jest w pozycji środkowej ($\approx 1500 \mu\text{s}$ w Betaflight, „na środku” fizycznie na kontrolerze RC) lub górnej ($2000 \mu\text{s}$ w Betaflight, „do góry” fizycznie na kontrolerze RC), co ok. 40 ms wysyłane jest nadpisanie wszystkich kanałów RC i wyrównanie ich wartości do $\approx 1500 \mu\text{s}$, oraz $\approx 2000 \mu\text{s}$ (brzęczyk aktywny zgodnie z [uprzednią konfiguracją ukazaną w Betaflight Configurator](#)) w przypadku kanału do którego przypisany jest brzęczyk, jeżeli wykryto przeskodę. W przypadku niewykrycia przeskody, wartość kanału, do którego przypisany jest brzęczyk, ustawiana jest na $\approx 1000 \mu\text{s}$ (brzęczyk wyłączony). Dodatkowo, dla pozycji górnej przełącznika obecnego fizycznie na kontrolerze RC pod oznaczeniem ‘B’, natychmiast działa nadpisanie nakładki obrazu na ekranie gogli FPV – w przypadku wykrycia przeskody, wyświetlana jest stosowna informacja, natomiast w przypadku jej braku, pusta lub po wyłączeniu tryb i zmianie pozycji przełącznika – przywrócenie pierwotnej nazwy obecnej w konfiguracji.

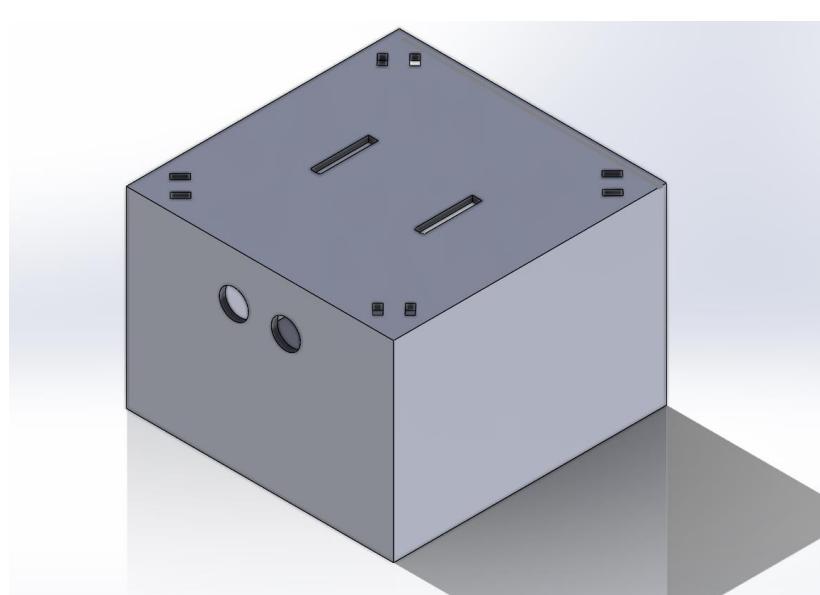
5.4. Integracja i montaż systemu

Celem zintegrowania całości opracowanego systemu, połączono w całość [schemat 1](#) i [schemat 2](#).

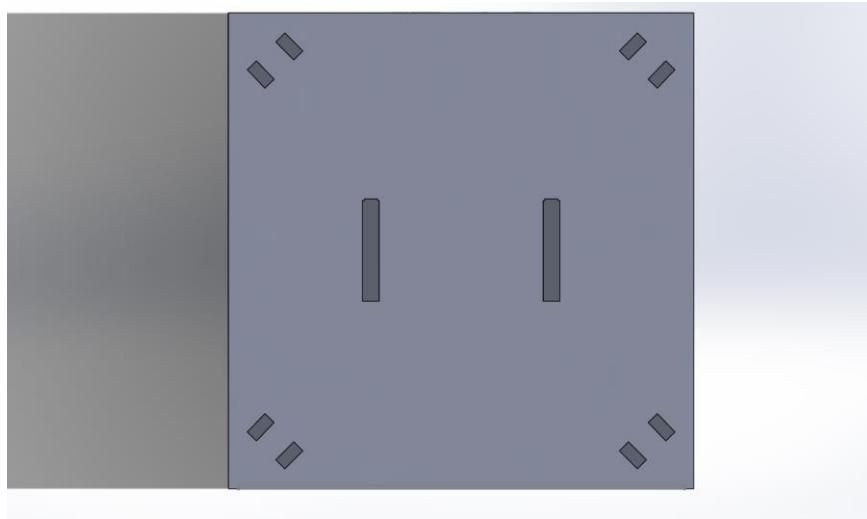
Otrzymano w ten sposób modułowy, przenośny oraz bardzo kompaktowy system, który następnie należało umieścić w obudowie. W tym celu, w programie *SolidWorks 2023*, posługując się techniką CAD zaprojektowano dwie części: pudełko oraz panel tylni, zaprezentowane kolejno na rysunkach 48-56 i 57-60.



Rysunek 48 - rzut na górną oraz tylną ściankę części-obudowy zaprojektowanej techniką CAD dla systemu wykrywania przeszkode w dronach FPV
 Źródło: opracowanie własne

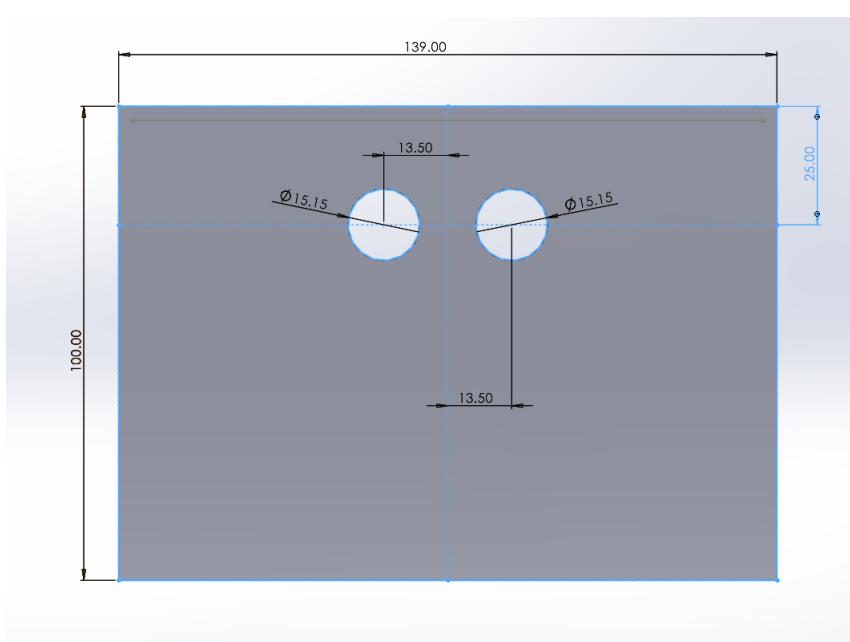


Rysunek 49 - rzut na górną oraz przednią ściankę części-obudowy zaprojektowanej techniką CAD dla systemu wykrywania przeszkode w dronach FPV
 Źródło: opracowanie własne



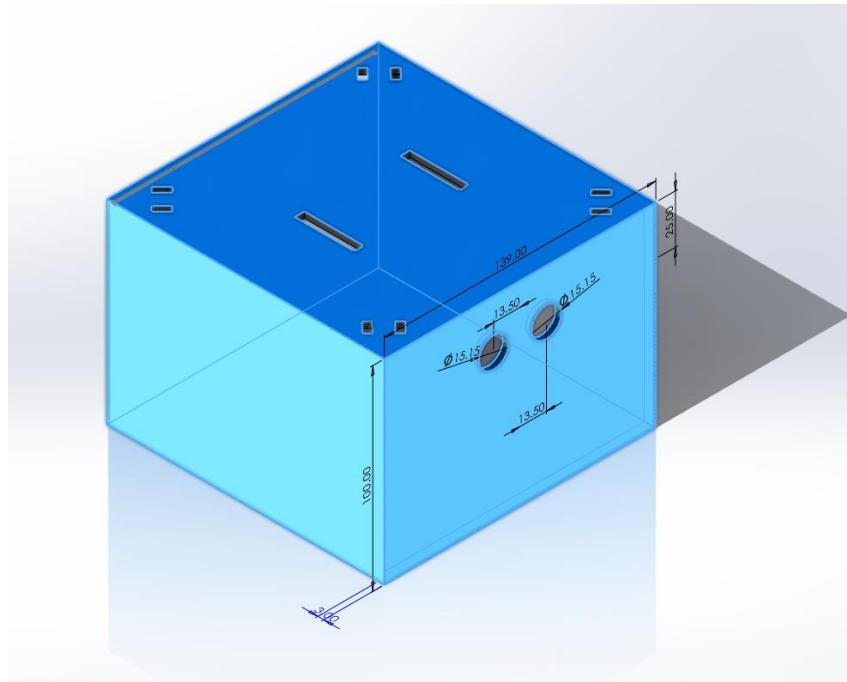
Rysunek 50 - rzut na górną ściankę części-obudowy zaprojektowanej techniką CAD dla systemu wykrywania przeszkode w dronach FPV

Źródło: opracowanie własne

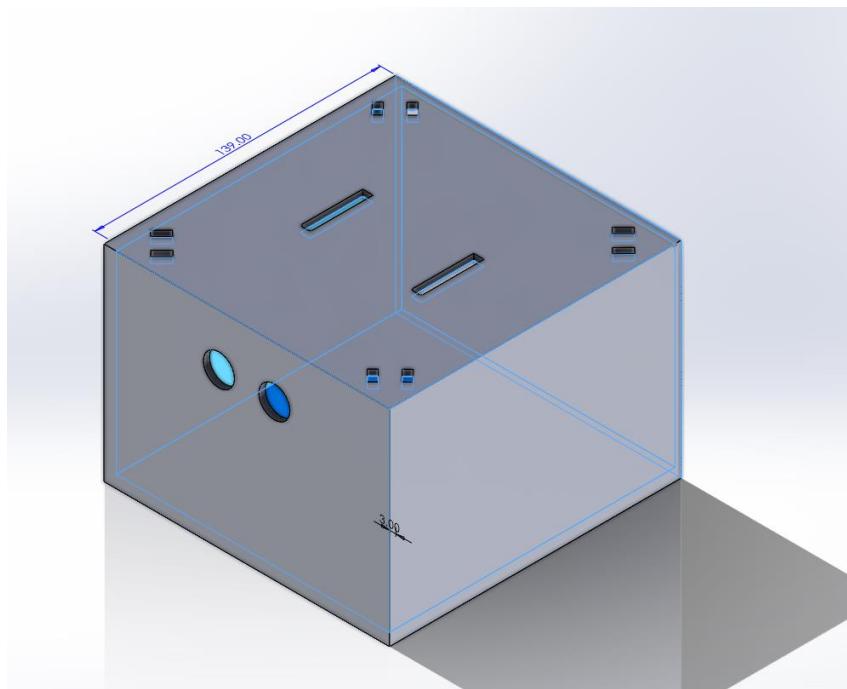


Rysunek 51 - rzut na przednią ściankę części-obudowy zaprojektowanej techniką CAD dla systemu wykrywania przeszkode w dronach FPV – wraz z wymiarami w milimetrach

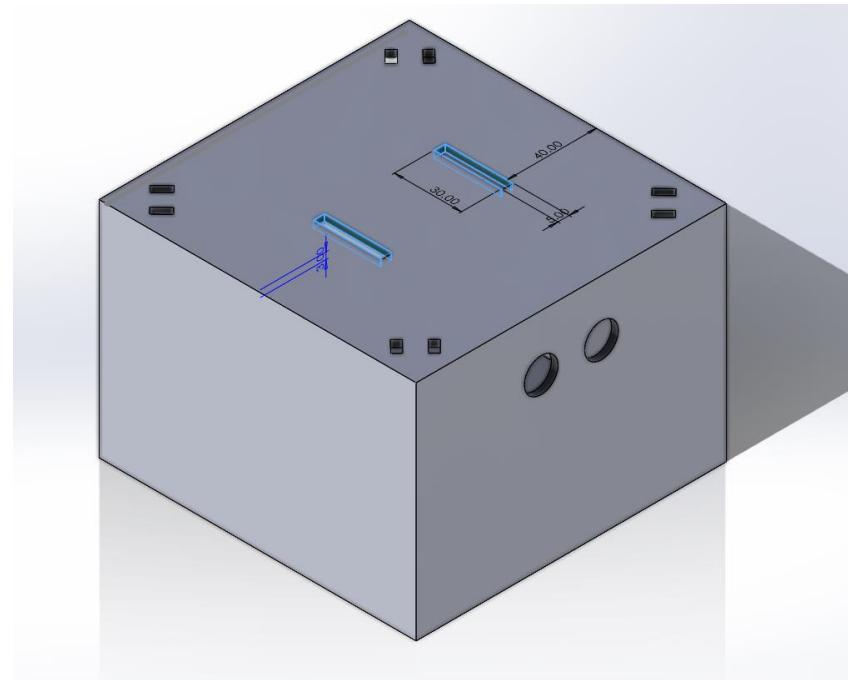
Źródło: opracowanie własne



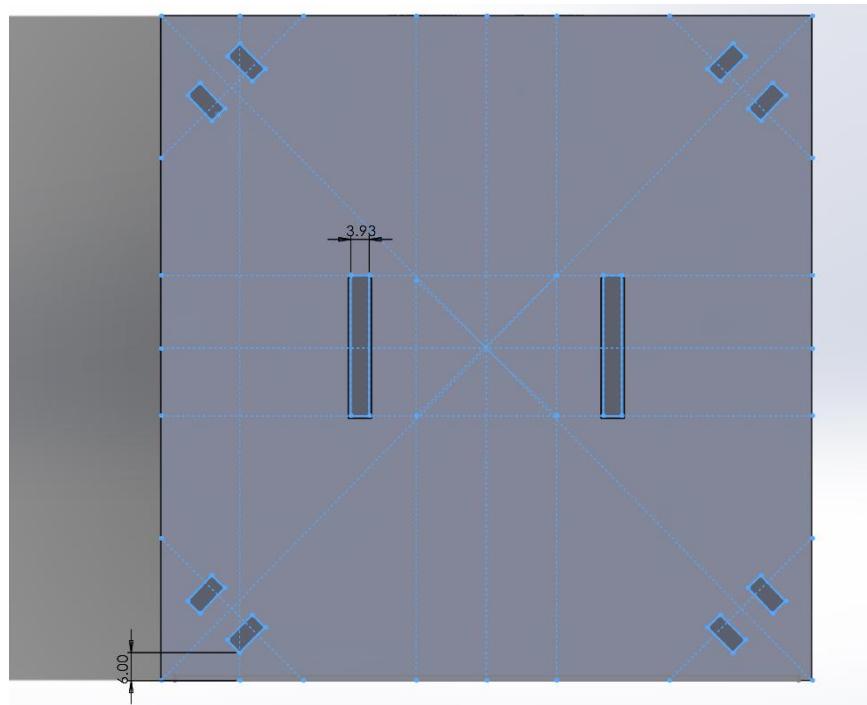
Rysunek 52 - rzut na górną oraz przednią ściankę części-obudowy zaprojektowanej techniką CAD dla systemu wykrywania przeszkode w dronach FPV – wraz z wymiarami po wykonaniu pierwszej operacji „wyciągnięcie dodania bazy”
 Źródło: opracowanie własne



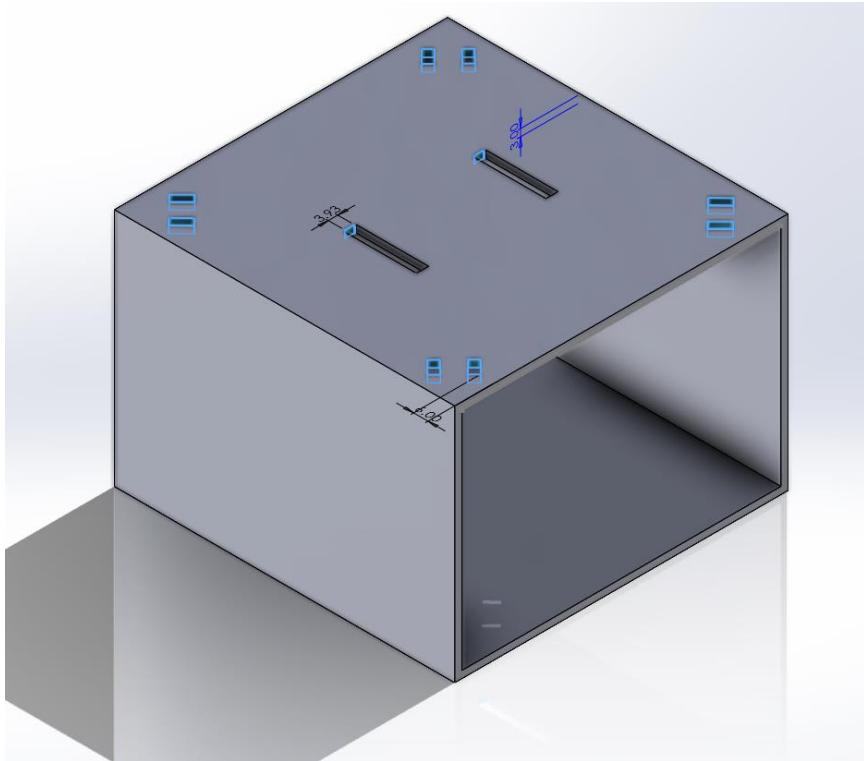
Rysunek 53 - rzut na górną oraz przednią ściankę części-obudowy zaprojektowanej techniką CAD dla systemu wykrywania przeszkode w dronach FPV – wraz z wymiarami po wykonaniu drugiej operacji „wyciągnięcie dodania bazy”
 Źródło: opracowanie własne



Rysunek 54 - rzut na górną oraz przednią ściankę części-obudowy zaprojektowanej techniką CAD dla systemu wykrywania przeszkode w dronach FPV – wraz z wymiarami po wykonaniu pierwszej operacji „wyciągnięcie wycięcia”
 Źródło: opracowanie własne

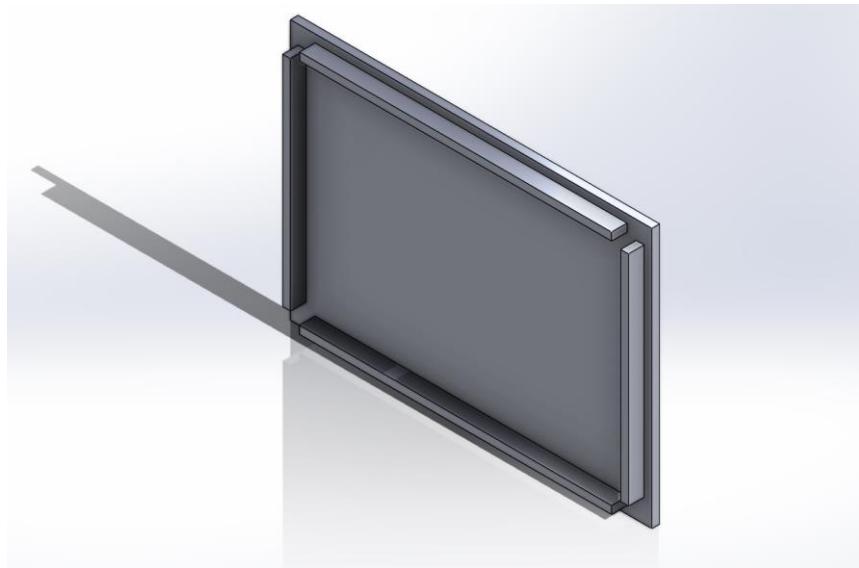


Rysunek 55 - rzut na górną ściankę części-obudowy zaprojektowanej techniką CAD dla systemu wykrywania przeszkode w dronach FPV – wraz z wymiarami po wykonaniu drugiej operacji „wyciągnięcie wycięcia”
 Źródło: opracowanie własne



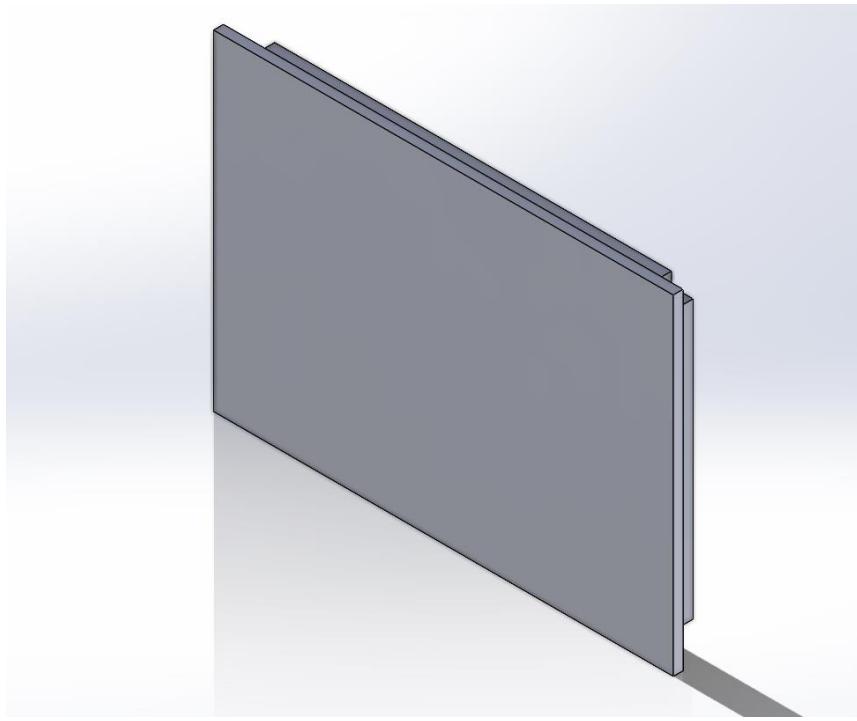
Rysunek 56 - rzut na tylną oraz górną ściankę części-obudowy zaprojektowanej techniką CAD dla systemu wykrywania przeszkode w dronach FPV – wraz z wymiarami po wykonaniu drugiej operacji „wyciągnięcie wycięcia”

Źródło: opracowanie własne

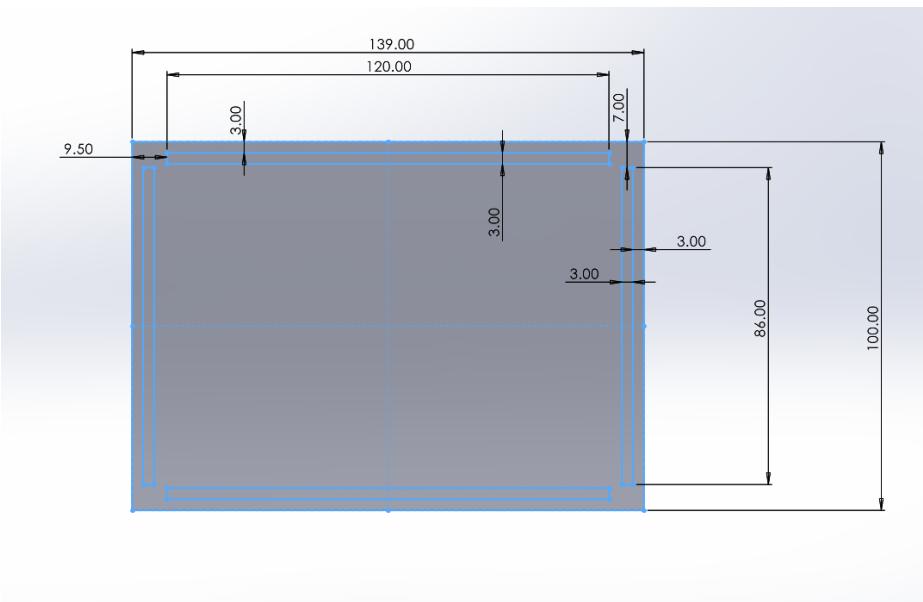


Rysunek 57 - rzut na górną oraz tylną ściankę części-panelu tylnego obudowy zaprojektowanej techniką CAD dla systemu wykrywania przeszkode w dronach FPV

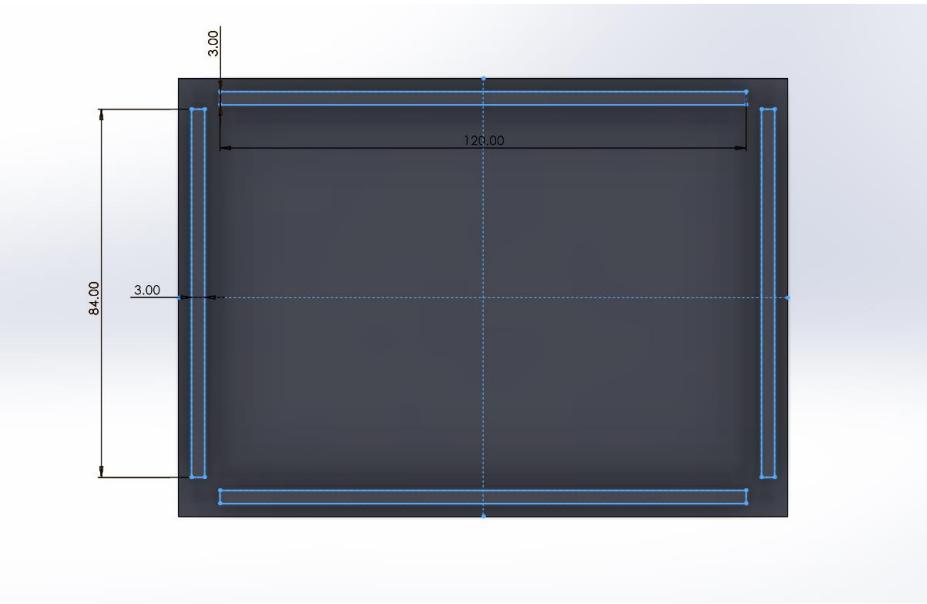
Źródło: opracowanie własne



Rysunek 58 - rzut na górną oraz przednią ściankę części-panelu tylnego obudowy zaprojektowanej techniką CAD dla systemu wykrywania przeszkode w dronach FPV
 Źródło: opracowanie własne



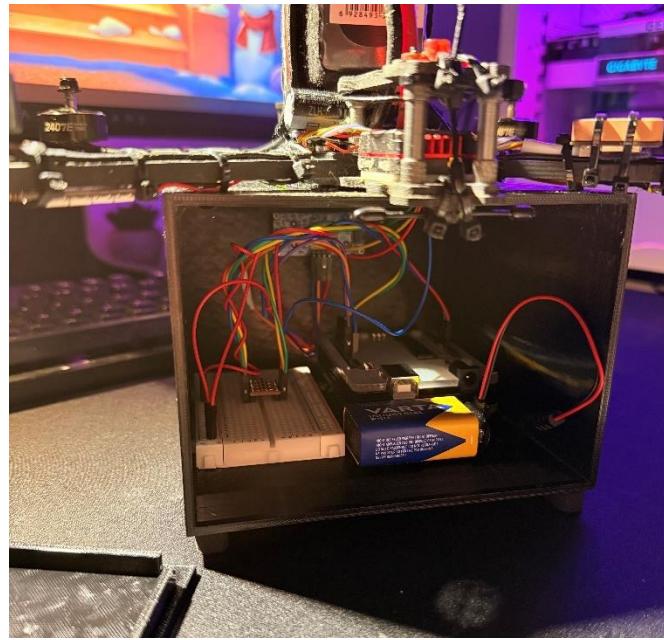
Rysunek 59 - rzut na przednią ściankę części-panelu tylnego obudowy zaprojektowanej techniką CAD dla systemu wykrywania przeszkode w dronach FPV – wraz z wymiarami
 Źródło: opracowanie własne



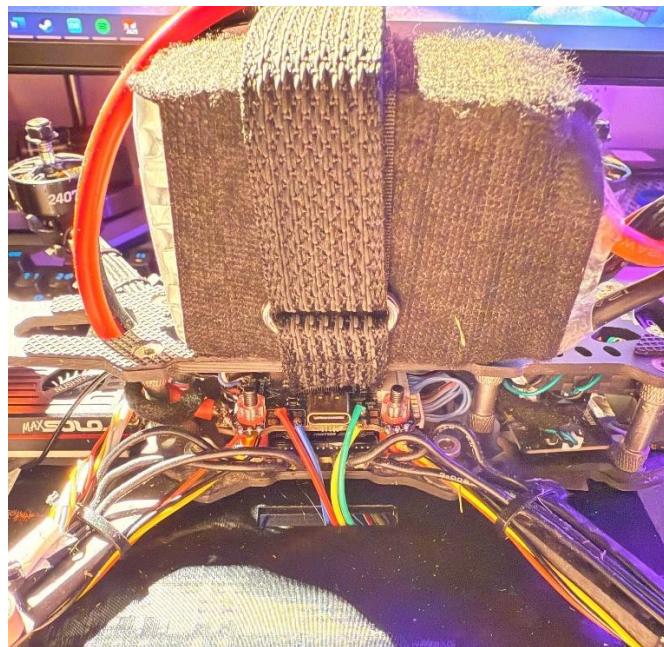
Rysunek 60 - rzut tylną ścianką części-panelu tylnego obudowy zaprojektowanej techniką CAD dla systemu wykrywania przeszkode w dronach FPV - wraz z wymiarami
 Źródło: opracowanie własne

Skonstruowaną w technice CAD obudowę na zaprojektowany system wraz z otworami na przewody oraz przewidywane mocowanie na ramie drona następnie wyeksportowano do formatu STL i wydrukowano metodą FDM, podobnie jak jej panel tylni, za pomocą mocno spersonalizowanej drukarki 3D Voron 2.4 300mm w kinematyce CoreXY z zabudowaną komorą. Do druku niniejszej obudowy, wykorzystano filament JAYO Antistring PLA, wykonany z polilaktydu – materiału o wysokiej sztywności i łatwości przetwarzania, kompostowalnego w warunkach przemysłowych, lecz cechującego się relatywnie większą kruchością (niższą udarnością)^[40]. Warto w tym miejscu zaznaczyć, iż tak jak w opisywanym przypadku, może zajść konieczność minimalnego dostosowania otworów wyjściowych czujnika w obudowie – w tym celu posłużyono się papierem ściernym.

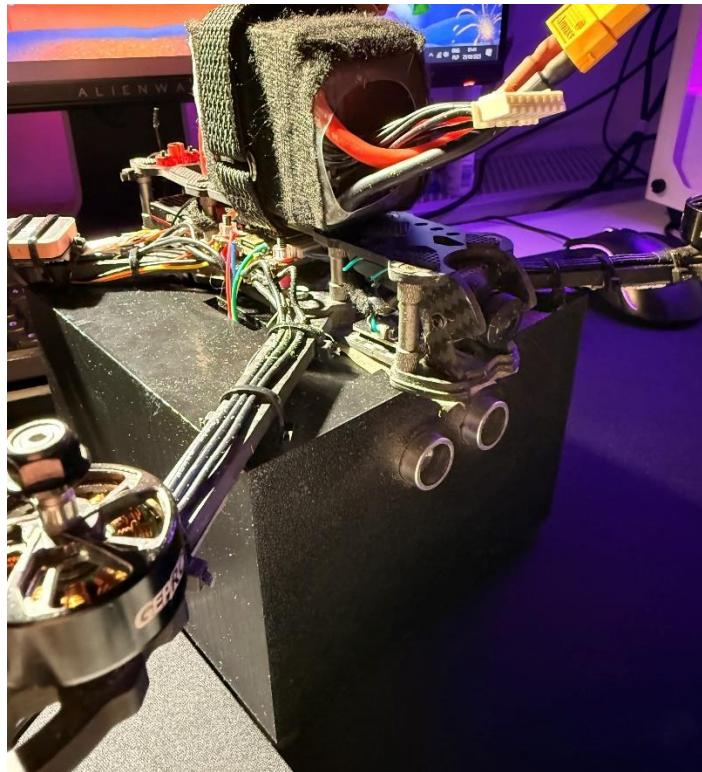
Następnie, w wydrukowanej obudowie bez założonego tylnego panelu, umieszczono płytke stykową połączoną z platformą Arduino wraz z czujnikiem oraz źródłem zasilania – baterią 9V – a obudowę przymocowano do ramy drona. Ostateczny rezultat montażu prezentują zdjęcia 14-16.



Zdjęcie 14 - zintegrowany system wykrywania przeszkoł dla dronów FPV w wersji przed testami laboratoryjnymi – zdjęcie pierwsze
Źródło: opracowanie własne



Zdjęcie 15 - zintegrowany system wykrywania przeszkoł dla dronów FPV w wersji przed testami laboratoryjnymi – zdjęcie drugie
Źródło: opracowanie własne



Zdjęcie 16 - zintegrowany system wykrywania przeszkode dla dronów FPV w wersji przed testami laboratoryjnymi – zdjęcie trzecie
 Źródło: opracowanie własne

Wszystkie elementy systemu – płytka stykowa, mikrokontroler Arduino Uno R3 oraz bateria zostały zamocowane wewnątrz obudowy przy pomocy taśmy dwustronnej, co zapobiega ich przesuwaniu w trakcie lotu. Obudowa została przymocowana do ramy drona za pomocą pasek zaciskowych w przewidzianych do tego miejscach, tj. przy narożach górnej płyty obudowy.

Ostatnim krokiem, który wykonano realizując integrację systemu wykrywania przeszkode, było zapobiegnięcie nadpisywaniu przez Arduino kanałów RC w trakcie lotu drona, gdyż prowadziłoby to do niebezpiecznej sytuacji w postaci utraty sygnału sterującego maszyną w powietrzu. W tym celu, należało zastosować maskę dla nadpisywanego kanału RC, aby umożliwić swobodne nadpisywanie go przez Arduino. Aby zrealizować powyższe, kontroler lotu ponownie podłączono do komputera PC z oprogramowaniem Betaflight Configurator, a następnie w zakładce konsoli CLI wpisano wartości:

```
SET MSP_OVERRIDE_CHANNELS_MASK = 512
SAVE
```

Wpisanie oraz zatwierdzenie kolejno po sobie powyższych dwóch poleceń, automatycznie zamyka okno konsoli, resetując kontroler lotu, wobec czego niemożliwym jest ukazanie ich wyniku

w niniejszym opracowaniu. Wartość 512 jest bezpośrednim zastosowaniem wzoru ustawienia maski MSP dla poszczególnych, pojedynczych kanałów AUX/CH, gdzie:

$$x = 2^y \text{ (2) } \text{gdzie:}$$

*Równanie 2 - wzór wartości maski na podstawie indeksu kanału AUX/CH
Źródło: opracowanie własne na podstawie^[41]*

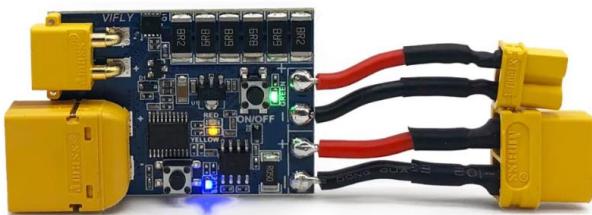
x – ustawiana wartość maski dla pojedynczego kanału

y – numer indeksu kanału (w opisywanym przykładzie CH10 → indeks = 9)

6. Testy systemu

6.1. Testy laboratoryjne w kontrolowanych warunkach

Testy w warunkach kontrolowanych odbywały się bez zamocowanych śmiegiel, z uwagi na bezpieczeństwo. Początkowo, sprawdzono jakość połączeń lutowniczych za pomocą miernika uniwersalnego, celem uniknięcia spalenia komponentów w przypadku zasilenia drona z baterii Li-Po. Następnie, po zweryfikowaniu połączeń miernikiem, pomiędzy baterią Li-Po a wtyczką drona umieszczono elektroniczny bezpiecznik wstępnego uruchamiania – VIFLY ShortSaver 2. Jest to urządzenie odcinające zasilanie przy zwarciu lub nadprądzie z przełącznikiem 1A/2A, działające z pakietami baterii 2-6S oraz posiadające złącza XT30/XT60 z czasem zadziałania kilku milisekund. Użycie takiego urządzenia, bądź innego podobnego aktywnego zabezpieczenia przeciwzwarcioowego jest zalecane podczas testów pierwszych konfiguracji, z uwagi na łatwość spalenia całości układu w przypadku niepoprawnego/niedokładnego podłączenia komponentów. Urządzenie zaprezentowano na rysunku 61.



Rysunek 61 - rysunek przedstawiający wybrane urządzenie odcinające zasilanie przy zwarciu - VIFLY ShortSaver V2
 Źródło: Strona z częściami do dronów FPV^[42]

W niniejszym przykładzie nie stwierdzono nieprawidłowości/niedokładności połączeń, wobec czego przystąpiono do testów. Uruchomiono kontroler RC oraz gogle FPV. Jednostka Arduino została podłączona do komputera PC z otwartym Arduino IDE oraz monitorem portu szeregowego, natomiast drona podłączono do obecnego na górnej części ramy pakietu 6S. Testy przeprowadzono dla trzech scenariuszy:

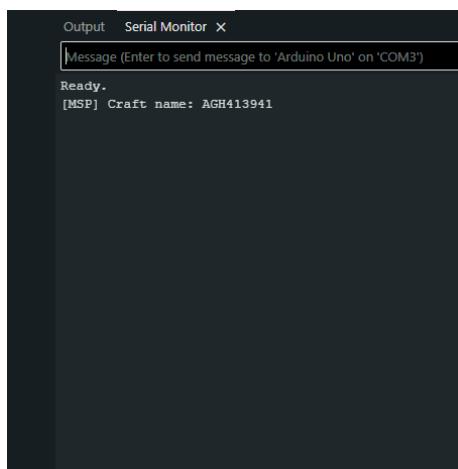
- 1) przełącznik trybu wspomagania w pozycji dolnej;
- 2) przełącznik trybu wspomagania w pozycji środkowej;
- 3) przełącznik trybu wspomagania w pozycji górnej.

Dla każdego ze scenariuszy zastosowano metodykę oddalenie od drona na odległość 3 metrów, a następnie zbliżenie się na odległość mniejszą niż 2 metry od czujnika ultradźwiękowego wyprowadzonego przez otwory w obudowie systemu. Wyniki prezentują rysunek 62 oraz tabela 6.

Tabela 6 - zestawienie działania systemu w symulowanych warunkach oraz poszczególnie aktywowane elementy systemu wykrywania przeszkód i ostrzegania pilota

Pozycja przełącznika	Odległość od przeszkodej	Skutek dźwiękowy	OSD w goglach	MSP_OVERRIDE
Pozycja dolna	> 2m	Brak	Nazwa wpisana w konfiguratorze	Wyłączone
Pozycja dolna	< 2m	Brak	Nazwa wpisana w konfiguratorze	Wyłączone
Pozycja środkowa	> 2m	Brak	Nazwa wpisana w konfiguratorze	Włączone
Pozycja środkowa	< 2m	Brzęczyk działa	Nazwa wpisana w konfiguratorze	Włączone
Pozycja górnna	> 2m	Brak	Nazwa wyczyszczona (brak na OSD)	Włączone
Pozycja górnna	< 2m	Brzęczyk działa	Wyświetlany komunikat	Włączone

Źródło: opracowanie własne



Rysunek 62 - zrzut ekranu konsoli prezentujący poprawnie odczytaną przez Arduino ramkę MSP od kontrolera lotu z nazwą modelu drona ustawioną w zakładce konfiguracyjnej oprogramowania Betaflight Configurator
Źródło: opracowanie własne

6.2. Eksperymentalne loty testowe w rzeczywistym środowisku

Po poprawnie przeprowadzonych testach w symulowanych warunkach, założono tylną ściankę obudowy całości systemu, zakręcono śmigła oraz udało się na otwartą przestrzeń umożliwiającą wykonywanie lotów BSP zgodnie ze strefami wyznaczonymi w aplikacji obecnej na smartfonie *DroneTower*. Kolejno uzbrojono drona oraz dokonano startu. W powietrzu, zawieszono kolejną maszynę BSP typu dron FPV i zbliżano do siebie dwa obiekty. W wyniku przeprowadzonego testu, potwierdzono działanie systemu dla wolnych prędkości lotu (mniejszych niż 10 km/h) – zarówno sygnał audio dźwiękowy jak i wypisanie komunikatu na ekranie gogli FPV zadziałało poprawnie w obydwu lotach testowych. Wobec powyższego dowiedziono, że system działa poprawnie, dla symulacji uczenia się latania przez młodego pilota. Fragment nagrani z lotów eksperymentalnych zaprezentowano na zdjęciach 17 i 18.



Zdjęcie 17 - Zdjęcie prezentujące pierwszy lot eksperymentalny z opracowanym systemem wykrywania przeszkode dla dronów FPV
 Źródło: opracowanie własne



Zdjęcie 18 - Zdjęcie prezentujące drugi lot eksperymentalny z opracowanym systemem wykrywania przeszkód dla dronów FPV
 Źródło: opracowanie własne

6.3. Analiza skuteczności unikania przeszkód w zależności od parametrów lotu

Jak zaznaczono końcem poprzedniego podrozdziału, system spełnia swoje funkcje, jednakże z pewnymi ograniczeniami. Założyć bowiem należy, iż system wykorzystywany będzie przez początkujących pilotów, uczących się latać i rozpoczynających pasję związaną z dronami FPV. Dowiedzono także, że czujnik ultradźwiękowy HC-SR04, z zastosowaną medianą z 5 pomiarów nadaje się do realizacji niniejszego projektu i jedynie czasami łapie szum z otoczenia, pomimo bardzo szybko kręcących się śmigieł drona FPV, które takowy szum generują. Idąc dalej, pewnym jest fakt, iż niemożliwym byłoby rozpoczęcie drona do jego maksymalnej prędkości (która w tym przypadku nie jest bliżej określona z uwagi na wpływ masy systemu) i zatrzymanie go w odpowiednim momencie zdając się na powyższy system i czas reakcji pilota. Wychodziłoby to poza ramy projektu kierowanego do początkujących pilotów. Niniejszy system, ma bowiem jedynie charakter ostrzegawczy – nie ma na celu automatyzacji lotu i autonomicznego omijania przeszkód – ten koncept został wykluczony na etapie doboru oprogramowania kontrolera lotu, które w przypadku Betaflight takie działanie uniemożliwia z uwagi na brak wsparcia. Jednocześnie, projektując niniejszy system, starano się zachować w jak największym stopniu bezpieczeństwo zarówno pilota, jak i jego najbliższego otoczenia, co byłoby niemożliwe w przypadku sterowania wartościami kanałów RC odpowiadających za lot *sensu stricte*. Wobec powyższego, możliwe jest w tym miejscu obrócenie obudowy i zastosowanie jej jako odpowiednik czujnika cofania dla samochodów osobowych.

7. Podsumowanie, wnioski końcowe oraz przyszłe kierunki badań

7.1. Ocena efektywności zastosowanego systemu

Opracowany system wykrywania przeszkód dla dronów FPV w integracji z platformą Arduino i interfejsem MSP realizuje założenia funkcjonalne. System jest sprawny, stosunkowo dobrze odporny na błędy, jak na system zaimplementowany w dronie FPV, które już same w sobie nie są odporne na uszkodzenia mechaniczne – z uwagi na grubość ścianek obudowy oraz fakt, że w razie upadku czujnik cofa się do wnętrza obudowy. Pomiar odległości na poziomie 2 metrów daje wyniki w postaci sygnału akustycznego oraz komunikatu OSD, jedynie momentami łapiąc szum z otoczenia i odbijając się od innych obiektów. Logika przełącznika dająca pilotowi możliwość aktywacji/częściowej aktywacji/dezaktywacji systemu działa zgodnie z wymogami. W przeprowadzonych próbach zarówno laboratoryjnych jak i lotach eksperymentalnych sonar wykrywał obiekty w zadanym progu. W testach akustycznych wykonywanych w trakcie lotów eksperymentalnych, sygnał akustyczny był słyszalny na poziomie kilkudziesięciu metrów, co umożliwia reakcję pilota także bez analizy OSD.

7.2. Możliwości dalszego rozwoju

Rozważając projekt implementujący czujniki mierzące odległość w konfiguracji sprzętowej z platformą Arduino, należy przede wszystkim rozważyć możliwość wykonania pełnoprawnego sonaru – czujnika zamontowanego na serwomechanizmie, które również są obsługiwane w większości oprogramowań kontrolerów lotu. Mówiąc o czujnikach mierzących odległość od obiektu, można również rozważyć opcje czujników mniej budżetowych, ale za to dokładniejszych i zwracających odpowiedź szybciej niż HC-SR04, np. czujników wyposażonych w LiDAR/ToF. Takie rozwiązanie, mogłyby zostać zastosowane dla pilotów bardziej doświadczonych, potrafiących już latać szybciej i dokładniej od początkujących.

7.3. Propozycje optymalizacji algorytmów unikania przeszkód

Wśród możliwych usprawnień niniejszego systemu, znajdą się: dynamiczny próg odległości od przeszkody zależący od prędkości lotu drona, ustawienie alarmu jedynie w przypadku gdy liczba próbek N z ostatnich pomiarów spełni założenie wykrycia przeszkody – co zmniejszy liczbę fałszywych alarmów, kompensacja temperatury otoczenia a co za tym idzie – korekta fali dźwiękowej czy połączenie większej ilości czujników celem umożliwienia wykrywania przeszkód nie tylko od frontu drona.

8. Bibliografia

- [1] Somerville, A., Wild, G., Lynam, T., & Joiner, K. (2024). "Use of Simulation for Pre-Training of Drone Pilots".
- [2] Shilpa K C, Mohan Gowda H, Nidish Shetty, Niranjan Reddy, Prafull K S (2024) "Design and Implementation of UART Transmitter and Receiver Using FPGA".
- [3] NXP Documentation - <https://www.nxp.com/docs/en/user-guide/UM10204.pdf> [data dostępu: 22.08.2025]
- [4] Nourdine Aliane (2024) „A Survey of Open-Source UAV Autopilots” – MDPI - <https://www.mdpi.com/2079-9292/13/23/4785> [data dostępu: 22.08.2025]
- [5] Skystars Manufacturer Official Site - Stack Specification. - https://www.skystars-rc.com/product/skystars-f722-app-wifi-hd3-flight-controller-blheli_32-ko60-60a-4in1-esc-3-6s-dshot600-fly-stack [data dostępu: 15.08.2025]
- [6] GEPRC Mark4 7inch product site - <https://geprc.com/product/gep-mark4-7-frame> [data dostępu: 15.08.2025]
- [7] GEPRC SpeedX2 2407 motor product site - <https://geprc.com/product/geprc-speedx2-2407e-motor> [data dostępu: 15.08.2025]
- [8] Radiomaster RP3 Receiver Documentation - <https://radiomasterrc.com/products/rp3-expresslrs-2-4ghz-nano-receiver?variant=46486353707200> [data dostępu: 15.08.2025]
- [9] Caddx Ratel 2 Product Site - <https://caddxfpv.com/products/ratel-2-1-1-8inch-starlight-sensor-freestyle-fpv-camera?variant=33301500526691> [data dostępu: 15.08.2025]
- [10] Best FPV Antenna Guide - <https://oscarliang.com/best-fpv-antenna> [data dostępu: 4.08.2025]
- [11] RushFPV Rush Tank Max Solo VTX Product Site - <https://rushfpv.net/products/rushfpv-max-solo-vtx> [data dostępu: 15.08.2025]
- [12] RushFPV Cherry 2 Antenna 5.8G Product Site - <https://rushfpv.net/products/rushfpv-cherry2-cherry-antenna-ii-5-8g> [data dostępu: 15.08.2025]
- [13] Rotorama Drones Part Shop - Tattu Funfly 6S Battery - <https://www.rotorama.com/product/tattu-funfly-1550mah-6s-100c> [data dostępu: 15.08.2025]

- [14] *iFlight GPS Product Site.* - <https://iflight-rc.eu/en/products/blitz-m10-gps-v2?srsltid=AfmBOorIYUV0BE4UjFiTyeZGy9f51gWLrZdsWDjYXtoO4F-OU-w4g7u3> [data dostępu: 15.08.2025]
- [15] *RCManiak Part Shop - Gemfan Flash 7040 Wings.* - <https://rcmaniak.pl/pl/p/Gemfan-Flash-7040-3-dwie-pary-CW-CCW/4706> [data dostępu: 15.08.2025]
- [16] *ELRS Official Site* - <https://www.expresslrs.org> [data dostępu: 17.08.2025]
- [17] *LBT Definition* - <https://en.telecomabc.nl/l/lbt.html> [data dostępu: 17.08.2025]
- [18] *Radiomaster Zorro Documentation* - <https://radiomasterrc.com/collections/zorro-radio-control-series/products/zorro-radio-controller?variant=46486367404224> [data dostępu: 15.08.2025]
- [19] *Skyzone Sky04o FPV Goggles Product Site* - <https://www.skyzonefpv.com/products/skyzone-sky04o-pro-oled-screen-fpv-goggle-with-1280-720-resolution-and-5-8g-receiver> [data dostępu: 15.08.2025]
- [20] *Arduino Documentation* - <https://docs.arduino.cc/hardware/uno-rev3> [data dostępu: 9.08.2025]
- [21] *Arduino Documentation - Arduino R3 Pinout.* - <https://docs.arduino.cc/resources/pinouts/A000066-full-pinout.pdf> [data dostępu: 9.08.2025]
- [22] *SparkFun Manufacturer Site* - <https://www.sparkfun.com/sparkfun-logic-level-converter-bi-directional.html> [data dostępu: 11.08.2025]
- [23] *SparkFun Manufacturer Site – Logic Level Converter Sheet*
https://cdn.sparkfun.com/datasheets/BreakoutBoards/Logic_Level_Bidirectional.pdf [data dostępu: 11.08.2025]
- [24] *Sunfounder - Unofficial Arduino Parts Documentation* - https://docs.sunfounder.com/projects/ultimate-sensor-kit/en/latest/components_basic/01-component_ultrasonic_module.html [data dostępu: 12.08.2025]
- [25] *Betaflight Configurator Repo* - <https://github.com/betaflight/betaflight-configurator/releases/tag/10.10.0> [data dostępu: 20.08.2025]
- [26] *Zadig Site* - <https://zadig.akeo.ie> [data dostępu: 20.08.2025]
- [27] *Skystars Manufacturer Official Site - ESC Specification* - <https://www.skystars-rc.com/product/ko60-blhel32-60a-3-6s-esc-support-128k-dshot1200-4-in-1-esc> [data dostępu: 21.08.2025]

- [28] *Betaflight Documentation - Ports Tab* - <https://betaflight.com/docs/wiki/configurator/ports-tab> [data dostępu: 20.08.2025]
- [29] *Rotorama Drones Part Shop - Skystars F7 22 HD Pro 4 Stack Pinout* - <https://www.rotorama.cz/cms/assets/docs/0aec10bb48e49a9755c67abc44e7f1db/14987-0/mmexport1692009673698.jpg> [data dostępu: 15.08.2025]
- [30] *Betaflight Documentation – Modes Tab – Airmode*. - <https://betaflight.com/docs/development/Modes#airmode> [data dostępu: 20.08.2025]
- [31] Zhu, J., Xu, W., Knapp, M., Ehrenberg, H., Wei, X., & Dai, H. (2023, July 19). „A method to prolong lithium-ion battery life during the full life cycle”. *Cell Reports Physical Science* 4.
- [32] *Betaflight Documentation - PID Tuning* - <https://betaflight.com/docs/development/pid-tuning> [data dostępu: 20.08.2025]
- [33] *Betaflight Documentation - Video Transmitter Tab* - <https://betaflight.com/docs/wiki/configurator/vtxtab> [data dostępu: 20.08.2025]
- [34] *Betaflight Documentation - Arming Flags* - <https://betaflight.com/docs/wiki/guides/current/Arming-Sequence-And-Safety#description-of-arming-prevention-flags> [data dostępu: 20.08.2025]
- [35] *EdgeTX Buddy* - buddy.edgetx.org [data dostępu: 20.08.2025]
- [36] *ELRS Configurator Repo* - <https://github.com/ExpressLRS/ExpressLRS-Configurator/releases> [data dostępu: 20.08.2025]
- [37] *ELRS Software Update Documentation & Guide* - <https://www.expresslrs.org/quick-start/transmitters/updating/#via-etx-passthrough> [data dostępu: 20.08.2025]
- [38] *Arduino Documentation - UART* - <https://docs.arduino.cc/learn/communication/uart> [data dostępu: 22.08.2025]
- [39] *Arduino Documentation - Setup Function* - <https://docs.arduino.cc/language-reference/en/structure/sketch/setup> [data dostępu: 23.08.2025]
- [40] *PLA Specs* - <https://pmc.ncbi.nlm.nih.gov/articles/PMC9822698> [data dostępu: 23.08.2025]
- [41] *Autonomous FPV Drone Operation Guide - MSP Masks* - <https://medium.com/illumination/fpv-autonomous-operation-with-betaflight-and-raspberry-pi-0caeb4b3ca69> [data dostępu: 23.08.2025]

[42] *Rotorama Drones Part Shop - Smoke Stopper* - <https://www.rotorama.com/product/vifly-shortsaver-v2> [data dostępu: 24.08.2025]

Spis rysunków

Rysunek 1 - Wybrany kontroler lotu (Skystars F7 22 HD Pro 4) wraz z kontrolerem silników połączone w stack	10
Rysunek 2 - Wybrana rama GEPRC Mark 4 – 7”	12
Rysunek 3 - Wybrany silnik GEPRC SpeedX2 2407 1750 KV	12
Rysunek 4 - Wybrany odbiornik Radiomaster RP3 V2 ELRS 2.4GHz.....	13
Rysunek 5 - Wybrana kamera FPV Caddx Ratel 2.....	13
Rysunek 6 - Wybrany nadajnik wideo - RushFPV Rush Tank Max Solo.....	14
Rysunek 7 - Wybrana antena do nadajnika wideo - RushFPV Cherry II MMCX.....	15
Rysunek 8 - Wybrana bateria Tattu Funfly 1550 mAh 6S 100C.....	15
Rysunek 9 - Wybrany GPS iFlight Blitz M10 V2 z kompasem	16
Rysunek 10 - Wybrane śmigła Gemfan 7040-3.....	16
Rysunek 11 - Wybrany nadajnik Radiomaster Zorro w rzucie z góry.....	18
Rysunek 12 - Wybrany model gogli FPV - Skyzone 04O Pro	19
Rysunek 13 - Całościowy model wejść/wyjść Arduino Uno R3	20
Rysunek 14 - Zdjęcie konwertera poziomów logicznych marki SparkFun modelu BOB-12009	21
Rysunek 15 - Schemat konwertera poziomów logicznych marki Sparkfun modelu BOB-12009....	21
Rysunek 16 - Wybrany czujnik ultradźwiękowy HC-SR04	23
Rysunek 17 - Aplikacja Betaflight Configurator po pierwszym uruchomieniu	25
Rysunek 18 - Okno menadżera urządzeń, ukazujące poprawnie rozpoznany port szeregowy kontrolera lotu	26
Rysunek 19 - Interfejs aplikacji Zadig służącej do ręcznej instalacji sterownika	26
Rysunek 20 - Lista opcji dostępnych w programie Zadig z możliwością wyświetlenia wszystkich urządzeń	27
Rysunek 21 - Okno aplikacji Betaflight Configurator po otworzeniu portu szeregowego kontrolera lotu	27

Rysunek 22 - Okno aplikacji Betaflight Configurator po poprawnym zamknięciu portu szeregowego kontrolera lotu oraz przejściu do zakładki aktualizacji oprogramowania.....	28
Rysunek 23 - Zakładka Setup w programie Betaflight Configurator	30
Rysunek 24 - Fragment schematu połączeń kontrolera lotu Skystars F7 22 HD Pro 4	31
Rysunek 25 - Zrzut ekranu ukazujący poprawną konfigurację portów szeregowych w oprogramowaniu Betaflight Configurator.....	31
Rysunek 26 - Zrzut ekranu prezentujący pierwszą część zakładki "Configuration" w programie Betaflight Configurator	33
Rysunek 27 - Zrzut ekranu prezentujący drugą część zakładki "Configuration" w programie Betaflight Configurator	34
Rysunek 28 - Zrzut ekranu z zawartością zakładki "Power & Battery" w programie Betaflight Configurator	36
Rysunek 29 - Zrzut ekranu prezentujący zawartość zakładki Failsafe w programie Betaflight Configurator - część 1	37
Rysunek 30 - Zrzut ekranu prezentujący zawartość zakładki Failsafe w programie Betaflight Configurator - część 2	38
Rysunek 31 - zakończona konfiguracja drona pod zakładką "Receiver" - część 1	40
Rysunek 32 - zakończona konfiguracja drona pod zakładką "Receiver" - część 2	41
Rysunek 33 - Zrzut ekranu prezentujący zakończoną konfigurację w zakładce "GPS" programu Betaflight Configurator	44
Rysunek 34 - Zrzut ekranu prezentujący zakończoną konfigurację w zakładce "Motors" programu Betaflight Configurator	45
Rysunek 35 – Zrzut ekranu przedstawiający zakończoną konfigurację w zakładce „OSD” programu Betaflight Configurator – część 1	46
Rysunek 36 – Zrzut ekranu przedstawiający zakończoną konfigurację w zakładce „OSD” programu Betaflight Configurator – część 2	47
Rysunek 37 – Zrzut ekranu przedstawiający zakończoną konfigurację w zakładce „OSD” programu Betaflight Configurator – część 3	47

Rysunek 38 – Zrzut ekranu przedstawiający zakończoną konfigurację w zakładce „OSD” programu Betaflight Configurator – część 4	48
Rysunek 39 - Zrzut ekranu przedstawiający zakończoną konfigurację w zakładce "Video Transmitter" programu Betaflight Configurator	49
Rysunek 40 - Zrzut ekranu ukazujący fragment zakładki "Sensors" programu Betaflight Configurator	50
Rysunek 41 - Zrzut ekranu ukazujący możliwości zakładki "Blackbox" programu Betaflight Configurator	51
Rysunek 42 - Zrzut ekranu przedstawiający wycinek konsoli systemowej programu Betaflight Configurator z wynikiem polecenia "status" dla opisywanego systemu	52
Rysunek 43 - Zrzut ekranu przedstawiający stronę aktualizacyjną oprogramowania EdgeTX z wybranym modelem kontrolera RC oraz wersją oprogramowania	54
Rysunek 44 - Zrzut ekranu prezentujący poprawnie dodany kontroler RC do aktualizatora EdgeTX	54
Rysunek 45 - Zrzut ekranu przedstawiający ostatnie okno podsumowujące przyszłą aktualizację oprogramowania EdgeTX na wybranym kontrolerze RC.....	55
Rysunek 46 - Zrzut ekranu prezentujący poprawnie wykonany proces aktualizacji oprogramowania EdgeTX będącego systemem operacyjnym kontrolera RC	56
Rysunek 47 - Zrzut ekranu przedstawiający zakończony sukcesem proces aktualizacji modułu Express LRS wewnętrz kontrolera RC	58
Rysunek 48 - rzut na górną oraz tylną ściankę części-obudowy zaprojektowanej techniką CAD dla systemu wykrywania przeszkód w dronach FPV	81
Rysunek 49 - rzut na górną oraz przednią ściankę części-obudowy zaprojektowanej techniką CAD dla systemu wykrywania przeszkód w dronach FPV.....	81
Rysunek 50 - rzut na górną ściankę części-obudowy zaprojektowanej techniką CAD dla systemu wykrywania przeszkód w dronach FPV.....	82
Rysunek 51 - rzut na przednią ściankę części-obudowy zaprojektowanej techniką CAD dla systemu wykrywania przeszkód w dronach FPV – wraz z wymiarami w milimetrach.....	82

Rysunek 52 - rzut na górną oraz przednią ściankę części-obudowy zaprojektowanej techniką CAD dla systemu wykrywania przeszkód w dronach FPV – wraz z wymiarami po wykonaniu pierwszej operacji „wyciągnięcie dodania bazy”	83
Rysunek 53 - rzut na górną oraz przednią ściankę części-obudowy zaprojektowanej techniką CAD dla systemu wykrywania przeszkód w dronach FPV – wraz z wymiarami po wykonaniu drugiej operacji „wyciągnięcie dodania bazy”	83
Rysunek 54 - rzut na górną oraz przednią ściankę części-obudowy zaprojektowanej techniką CAD dla systemu wykrywania przeszkód w dronach FPV – wraz z wymiarami po wykonaniu pierwszej operacji „wyciągnięcie wycięcia”	84
Rysunek 55 - rzut na górną ściankę części-obudowy zaprojektowanej techniką CAD dla systemu wykrywania przeszkód w dronach FPV – wraz z wymiarami po wykonaniu drugiej operacji „wyciągnięcie wycięcia”	84
Rysunek 56 - rzut na tylną oraz górną ściankę części-obudowy zaprojektowanej techniką CAD dla systemu wykrywania przeszkód w dronach FPV – wraz z wymiarami po wykonaniu drugiej operacji „wyciągnięcie wycięcia”	85
Rysunek 57 - rzut na górną oraz tylną ściankę części-panelu tylnego obudowy zaprojektowanej techniką CAD dla systemu wykrywania przeszkód w dronach FPV	85
Rysunek 58 - rzut na górną oraz przednią ściankę części-panelu tylnego obudowy zaprojektowanej techniką CAD dla systemu wykrywania przeszkód w dronach FPV	86
Rysunek 59 - rzut na przednią ściankę części-panelu tylnego obudowy zaprojektowanej techniką CAD dla systemu wykrywania przeszkód w dronach FPV – wraz z wymiarami	86
Rysunek 60 - rzut tylną ściankę części-panelu tylnego obudowy zaprojektowanej techniką CAD dla systemu wykrywania przeszkód w dronach FPV - wraz z wymiarami	87
Rysunek 61 - rysunek przedstawiający wybrane urządzenie odcinające zasilanie przy zwarciu - VIFLY ShortSaver V2	91
Rysunek 62 - zrzut ekranu konsoli prezentujący poprawnie odczytaną przez Arduino ramkę MSP od kontrolera lotu z nazwą modelu drona ustawioną w zakładce konfiguracyjnej oprogramowania Betaflight Configurator	92

Spis tabel

Tabela 1 - Opis wartości wstawionych w zakładce „Power & Battery” programu Betaflight Configurator	34
Tabela 2 - Wykorzystane funkcje zakładki „Modes” programu Betaflight Configurator wraz z przypisanymi wartościami kanałów AUX	41
Tabela 3 - Wybrane flagi uzbrojenia wraz z odpowiadającymi im opisami.....	52
Tabela 4 – Zestawienie poszczególnych kanałów wejściowych oraz odpowiadających im funkcji w zakładkach INPUTS oraz MIXES oprogramowania kontrolera RC	61
Tabela 5 - zestawienie funkcji specjalnych oraz odczytywanych wartości po zmianie stanu przełącznika na kontrolerze RC - ↑ oznacza stan przełącznika w dół (od kontrolera), ↓ oznacza stan przełącznika w górę (do kontrolera)	63
Tabela 6 - zestawienie działania systemu w symulowanych warunkach oraz poszczególnie aktywowane elementy systemu wykrywania przeszkód i ostrzegania pilota	92

Spis zdjęć

Zdjęcie 1 – zdjęcie ukazujące ekran kontrolera podczas konfiguracji modelu w zakładce „SETUP”, po wpisaniu nazwy modelu.....	58
Zdjęcie 2 – zdjęcie ukazujące ekran kontrolera podczas konfiguracji modelu w zakładce „SETUP”, po aktywacji protokołu CRSF.....	59
Zdjęcie 3 – zdjęcie ukazujące ekran kontrolera podczas konfiguracji modelu w zakładce „INPUTS”, po poprawnym dodaniu wirtualnych wejść	59
Zdjęcie 4 – zdjęcie ukazujące guzik oraz przełączniki obecne fizycznie z lewej strony kontrolera w rzucie od góry.....	60
Zdjęcie 5 – zdjęcie ukazujące guzik oraz przełączniki obecne fizycznie z prawej strony kontrolera w rzucie od góry.....	60

Zdjęcie 6 – zdjęcie ukazujące ekran kontrolera podczas konfiguracji modelu w zakładce „MIXES” po poprawnym spisaniu nazw funkcji wysyłanych poprzez kanały AUX do odbiornika obecnego po stronie kontrolera lotu na dronie	60
Zdjęcie 7 – zdjęcie ukazujące ekran kontrolera podczas konfiguracji modelu w zakładce „SPECIAL FUNCTIONS” po poprawnym skonfigurowaniu dźwięków odgrywanych przez kontroler po naciśnięciu guzików/zmianie stanu przełączników – część 1	63
Zdjęcie 8 – zdjęcie ukazujące ekran kontrolera podczas konfiguracji modelu w zakładce „SPECIAL FUNCTIONS” po poprawnym skonfigurowaniu dźwięków odgrywanych przez kontroler po naciśnięciu guzików/zmianie stanu przełączników – część 2	63
Zdjęcie 9 – zdjęcie ukazujące drona po poprawnym fizycznym podłączeniu wszystkich urządzeń peryferyjnych z kontrolerem lotu oraz płytka stykową, w jego rzucie od góry	65
Zdjęcie 10 - zdjęcie ukazujące drona po poprawnym fizycznym podłączeniu wszystkich urządzeń peryferyjnych z kontrolerem lotu oraz płytka stykową, w jego rzucie od boku z góry	66
Zdjęcie 11a) - zdjęcie przedstawiające domyślny przewód przesyłowy wykorzystywany przy systemach platformy Arduino	68
Zdjęcie 12a) - zdjęcie przedstawiające przewody połączeniowe dedykowane dla platformy Arduino - przewód męski (górnny) oraz żeński (dolny)	69
Zdjęcie 13 – zdjęcie przedstawiające połączoną platformę Arduino z czujnikiem ultradźwiękowym, płytka stykową oraz dostarczonym zasilaniem	71
Zdjęcie 14 - zintegrowany system wykrywania przeszkoł dla dronów FPV w wersji przed testami laboratoryjnymi – zdjęcie pierwsze	88
Zdjęcie 15 - zintegrowany system wykrywania przeszkoł dla dronów FPV w wersji przed testami laboratoryjnymi – zdjęcie drugie	88
Zdjęcie 16 - zintegrowany system wykrywania przeszkoł dla dronów FPV w wersji przed testami laboratoryjnymi – zdjęcie trzecie	89
Zdjęcie 17 - Zdjęcie prezentujące pierwszy lot eksperymentalny z opracowanym systemem wykrywania przeszkoł dla dronów FPV	93
Zdjęcie 18 - Zdjęcie prezentujące drugi lot eksperymentalny z opracowanym systemem wykrywania przeszkoł dla dronów FPV	94

Spis fragmentów kodu

Fragment kodu 1 - zawartość zestawu dla nadajnika video wykorzystanego podczas konfiguracji Źródło: zestaw konfiguracyjny użytkownika „Directory” w programie Betaflight Configurator.....	39
Fragment kodu 2 - fragment programu zarządzającego logiką systemu wykrywania przeskód po stronie Arduino – zmienne globalne	72
Fragment kodu 3 - fragment programu zarządzającego logiką systemu wykrywania przeskód po stronie Arduino – funkcja sendMspV1	73
Fragment kodu 4 - fragment programu zarządzającego logiką systemu wykrywania przeskód po stronie Arduino – funkcja mspRequest.....	74
Fragment kodu 5 - fragment programu zarządzającego logiką systemu wykrywania przeskód po stronie Arduino – funkcja readOneMspFrame – część 1	74
Fragment kodu 6 - fragment programu zarządzającego logiką systemu wykrywania przeskód po stronie Arduino – funkcja readOneMspFrame – część 2	75
Fragment kodu 7 - fragment programu zarządzającego logiką systemu wykrywania przeskód po stronie Arduino – funkcja pollMsp	75
Fragment kodu 8 - fragment programu zarządzającego logiką systemu wykrywania przeskód po stronie Arduino – funkcje pomocnicze część 1 – funkcje: sendBeeperAUX oraz setCraftName	76
Fragment kodu 9 - fragment programu zarządzającego logiką systemu wykrywania przeskód po stronie Arduino – funkcje pomocnicze część 2 – funkcje: setCraftNameBlank, setCraftNameObstacle, tickRequestRC, requestOrigNameOnce oraz funkcja specjalna setup	77
Fragment kodu 10 - fragment programu zarządzającego logiką systemu wykrywania przeskód po stronie Arduino – główna pętla programu – funkcja loop – część 1	79
Fragment kodu 11 - fragment programu zarządzającego logiką systemu wykrywania przeskód po stronie Arduino – główna pętla programu – funkcja loop – część 2	79

Spis schematów

Schemat 1 – faktyczny schemat dokonanych połączeń w budowie drona	67
Schemat 2 - schemat prezentujący fizyczne połączenia pomiędzy mikrokontrolerem Arduino, a płytka stykową oraz czujnikiem ultradźwiękowym.....	70

Spis równań

Równanie 1 – wzór obliczania nominalnego naładowania dla pakietu 2-6S.....	35
Równanie 2 - wzór wartości maski na podstawie indeksu kanału AUX/CH.....	90