

AKADEMIA GÓRNICZO-HUTNICZA

Raport z projektu

**Kontroler silników BLDC do
dronów i pojazdów RC**

z przedmiotu

Sensory w Aplikacjach Wbudowanych

Elektronika i Telekomunikacja - Systemy Wbudowane, rok I studiów
magisterskich

Michał Nizioł

Krzysztof Sikora

Spis treści

1. Opis i założenia projektu.
 - 1.1. Założenia projektowe i wysokopoziomowy opis projektu.
 - 1.1.1. Założenia projektowe odnośnie warstwy sprzętowej.
 - 1.1.2. Założenia projektowe odnośnie oprogramowania.
 - 1.2. Podział odpowiedzialności
2. Sprzęt
 - 2.1. Wybór komponentów.
 - 2.2. Opis schematu.
 - 2.3. PCB.
3. Oprogramowanie.
4. Uruchomienie i testy.

Bibliografia.

1. Opis i założenia projektu

1.1. Założenia projektowe i wysokopoziomowy opis projektu

1.1.1. Założenia projektowe odnośnie warstwy sprzętowej

Kontroler silników BLDC przeznaczony jest do stosowania w dronach oraz pojazdach zdalnie sterowanych. Jego zadaniem jest sterowanie prędkością i momentem obrotowym silnika.

Do najważniejszych założeń projektu zaliczamy:

- możliwość sterowania prędkością obrotową i momentem silnika
- maksymalny prąd wyjściowy do 60A
- niewielkie wymiary PCB - umożliwiające montaż w dronach oraz pojazdach RC
- praca z akumulatorami LiPo do 3S
- wsparcie dla algorytmu 6-step
- pady do kondensatora filtrującego zasilanie
- pady do przewodów zasilających oraz silnika - umożliwiające bezpośrednie lutowanie do płytka bez stosowania złącz
- implementacja cyfrowego protokołu DSHOT
- kontrola poprzez interfejs PWM oraz protokół DSHOT
- kontrola poprzez UART - służąca do dobrania parametrów pracy kontrolera
- dobranie parametrów regulatora PI do sterowania prędkością silników

1.2. Podział odpowiedzialności

Tabela z przypisaniem poszczególnych kamieni milowych/funkcjonalności do członków projektu

Osoba	Podział odpowiedzialności
Michał Nizioł	Firmware
Krzysztof Sikora	Hardware

Table 1.

2. Sprzęt

2.1. Wybór komponentów

Wykorzystywane elementy:

- **STSPIN32F0A** - mikrokontroler STM32 z wbudowanym układem drivera MOSFETów i wzmacniaczami operacyjnymi. Układ ten został wybrany, ponieważ integruje w sobie większość niezbędnych elementów kontrolera ESC, które zostały tu już wymienione. Zasilany jest z napięcia do 45V, co pozwala jeszcze bardziej zredukować liczbę komponentów w BOM. Wbudowany w niego mikrokontroler STM32F0 pozwala na programowanie poprzez SWD oraz kontrolę silnikiem poprzez UART, bądź dowolny inny interfejs. Dedykowana aplikacja Motor Pilot od ST umożliwia szybki dobór parametrów silnika.
- **PSMNR55-40SSH** - MOSFET typu N o maksymalnym napięciu 40V i prądzie ciągłym 500A. Cechuje się on bardzo małym R_{DS(ON)} do 0.55mOhm oraz niskim ładunkiem bramki - około 200nC, co pozwala na jego szybkie przełączanie. Niskie napięcia progowe i stroma charakterystyka przejściowa minimalizują straty mocy.

Poza wymienionymi wyżej układami, projekt zawiera jedynie elementy pasywne: rezystory, kondensatory, cewki oraz złącza. W przypadku tych komponentów, ich wybór podyktowany był rozmiarem (możliwie mały, lecz łatwy w lutowaniu ręcznym), jak i parametrami takimi jak napięcie na elemencie, maksymalny prąd, wydzielana moc.

2.2. Opis schematu

Schematy oraz projekt PCB wykonane zostały w programie Altium Designer 25.

Na schemacie “*Figure 1. Schemat układu STSPIN32F0A.*” widoczny jest główny układ scalony - STSPIN32F0A. Cewka L1, wraz z diodą D7 oraz kondensatorami filtrującymi tworzy przetwornicę typu buck, za której sterowanie odpowiada wbudowany układ z wbudowanym tranzystorem przełączającym. Odpowiada ona za wytworzenie na płytce napięcia 3.3V zasilającego mikrokontroler. Każdy pin zasilający układu posiada ponadto dodatkowe kondensatory filtrujące zasilanie. Ich wartości są wartościami zalecanymi przez producenta, możliwe jest jednak zastosowanie kondensatorów o większej pojemności w tej samej obudowie.

W dolnej części strony widoczne są elementy odpowiadające za sprężenie zwrotne z rezystora pomiarowego.

Większość pinów układu odpowiada za sterowanie bramkami tranzystorów MOS (V_{BOOTx}, HS_{Ux}, MS_{Ux}), bądź pomiar napięcia z wyjść napięciowych (OUT_x).

Na schemacie “[*Figure 2. Schemat wyjścia silnika - MOSFETów mocy.*](#)” przedstawiono tranzystory mocy MOSFET typu N. Odpowiadają one za przełączanie poszczególnych uzwojeń silnika BLDC. Ich bramki zasilane są poprzez rezystory, połączone równolegle z diodami, które mają za zadanie przyspieszyć proces rozładowywania ładunku bramki. Konkretnie wartości elementów należy ustalić na etapie testów układu, przedstawione wartości zostały dobrane, bazując na płytce ewaluacyjnej STEVAL-SPIN3202.

Rezystory R6 i R7 odpowiadają za pomiar prądu na uzwojeniach silnika, są to rezystory o większym rozmiarze i maksymalnej wydzielanej mocy.

Schemat “[*Figure 3. Schemat wejścia zasilania.*](#)” pokazuje wejście zasilania wraz z opcjonalnymi kondensatorami filtrującymi zasilanie. Możliwe jest również dolutowanie dodatkowego kondensatora elektrolitycznego o dużej pojemności do dedykowanych padów na PCB. Pojemność wejściowa uzależniona jest od maksymalnego prądu podłączonego silnika.

Na rysunku “[*Figure 4. Schemat układu sprzężenia zwrotnego.*](#)” widzimy dzielniki napięcia wraz z diodami zabezpieczającymi, które doprowadzają napięcia z poszczególnych faz silnika do układu pomiarowego w mikrokontrolerze STSPIN32. Mierzone jest również napięcie zasilające.

Wyprowadzenia modułu przedstawia schemat “[*Figure 5. Schemat wyprowadzeń modułu.*](#)”. Wyprowadzone zostało złącze dla programatora ST-LINK (SWD), a także: UART, wejście timera (dla PWM) i dodatkowe piny GPIO. Dzięki temu możliwe jest łatwiejsze debugowanie, a także wykorzystanie oprogramowania Motor Pilot od STMicroelectronics do dobrania parametrów kontroli silnika.

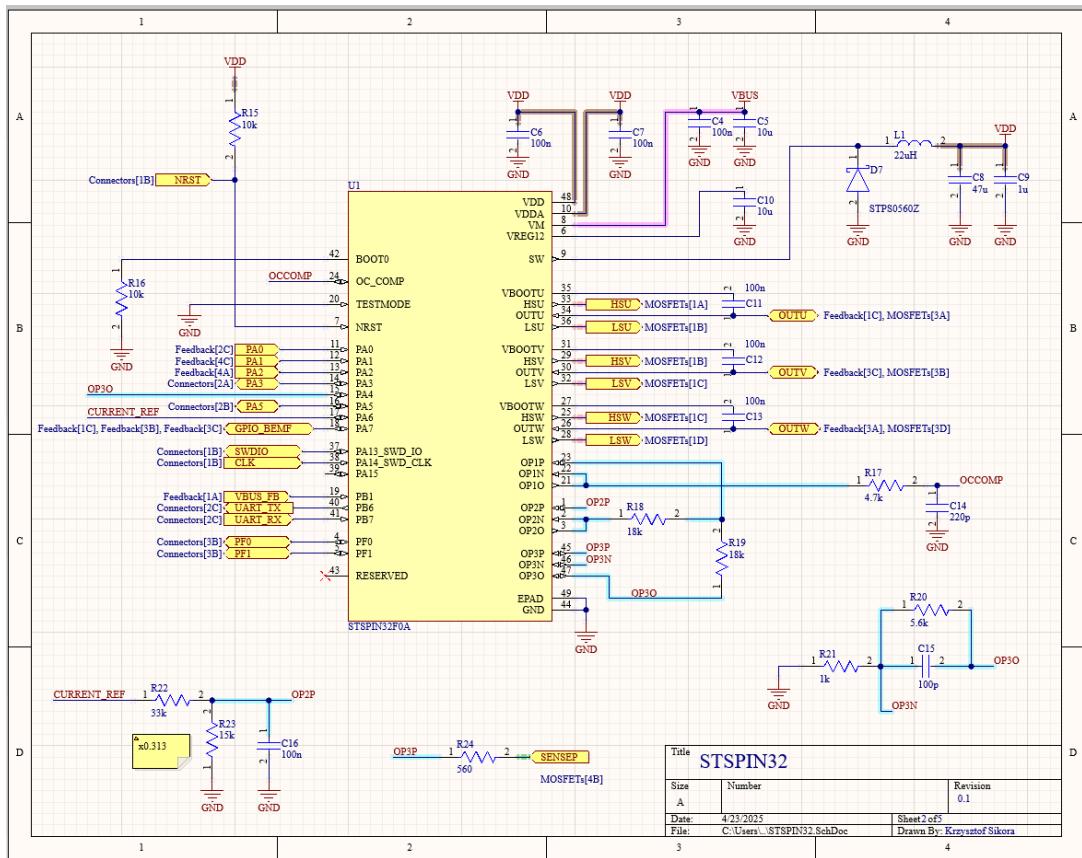


Figure 1. Schemat układu STSPIN32F0A.

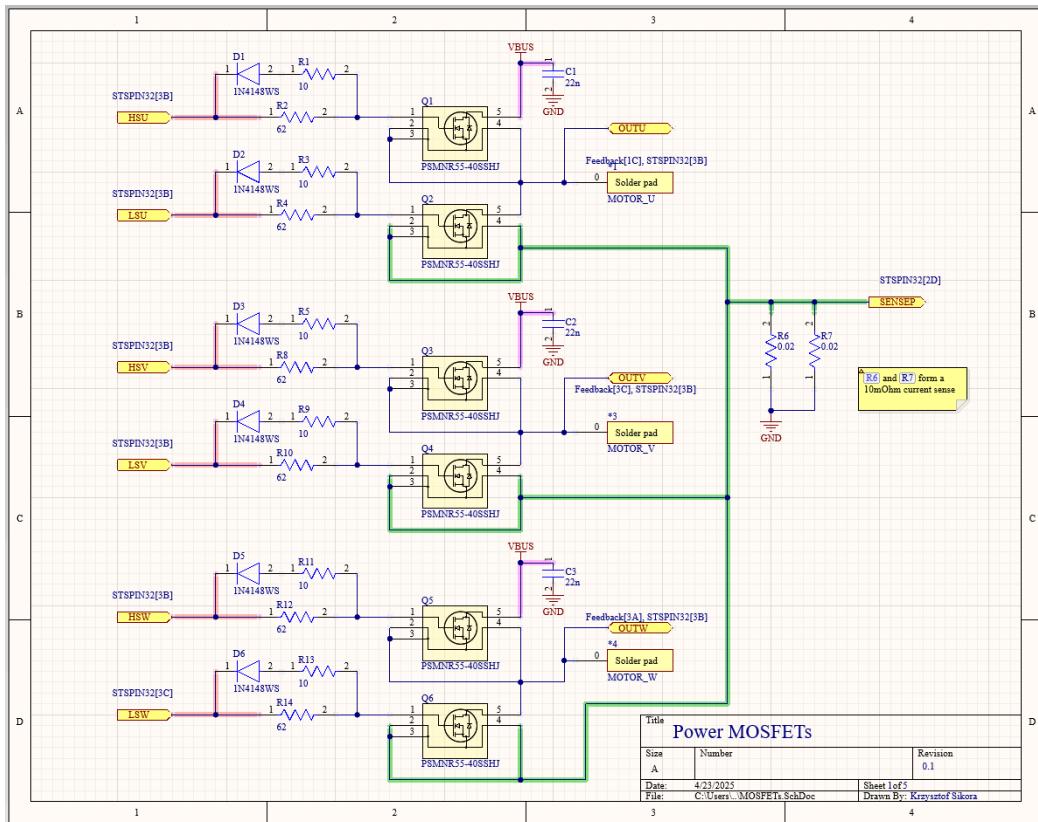


Figure 2. Schemat wyjścia silnika - MOSFETów mocy.

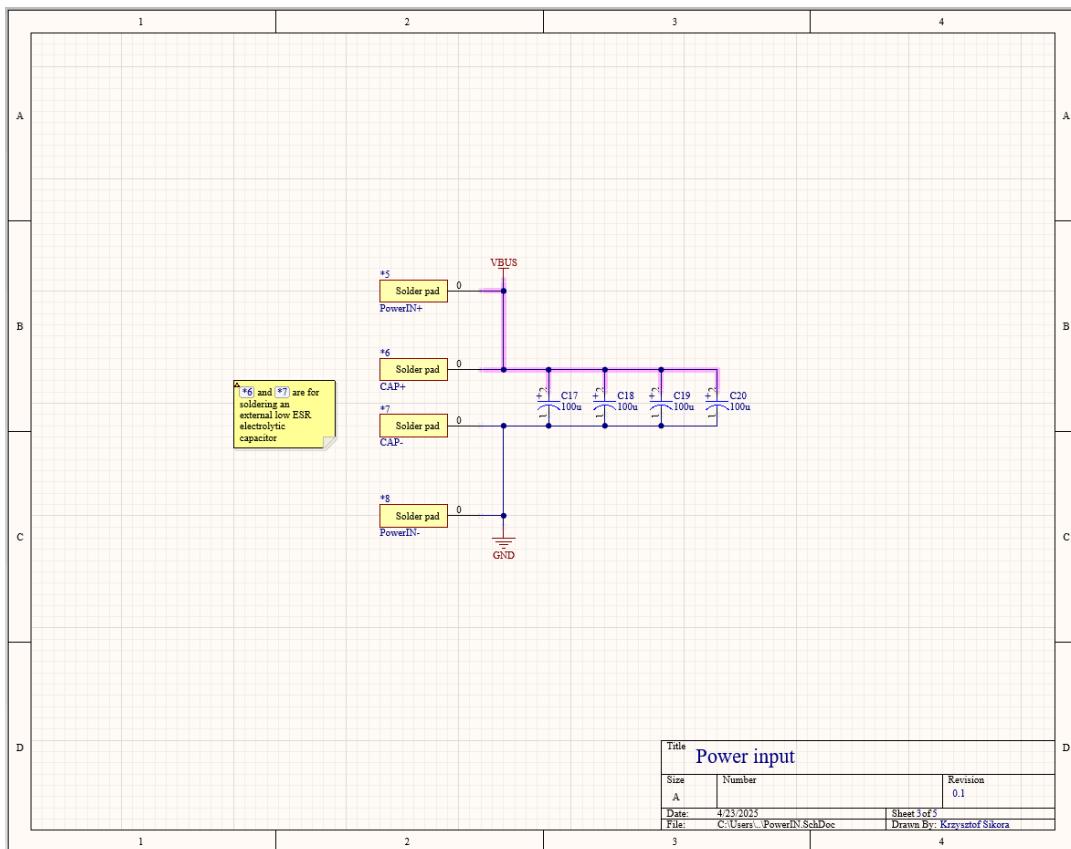


Figure 3. Schemat wejścia zasilania.

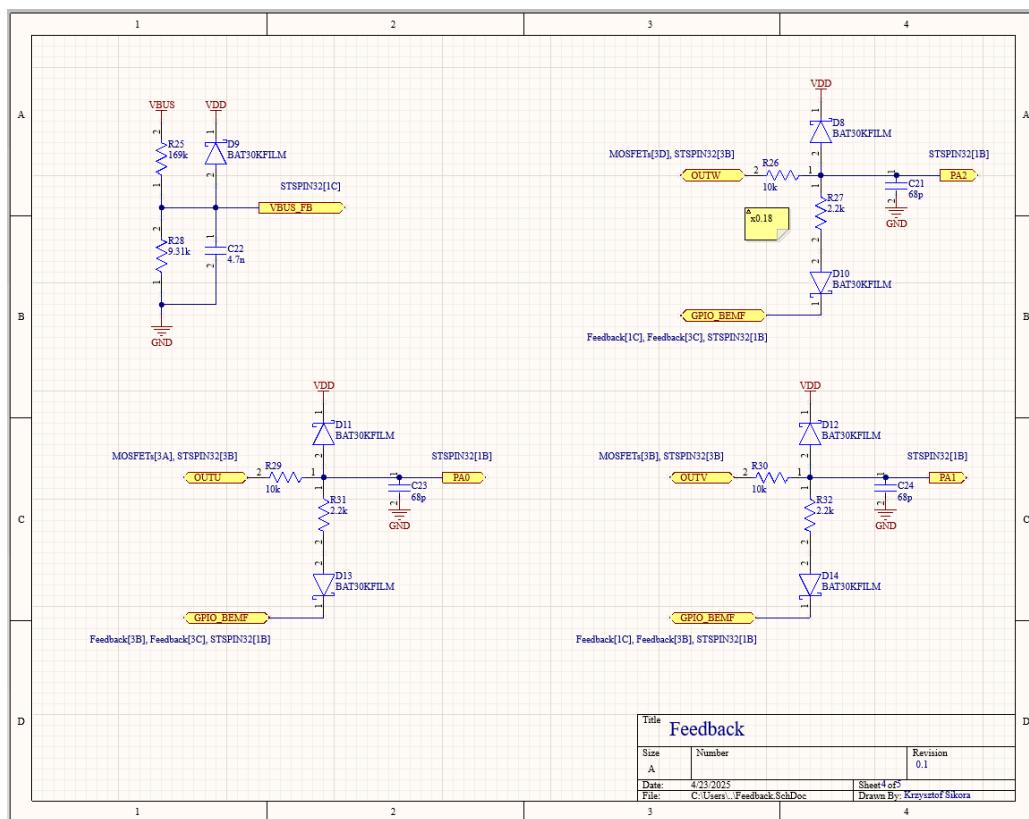


Figure 4. Schemat układu sprzężenia zwrotnego.

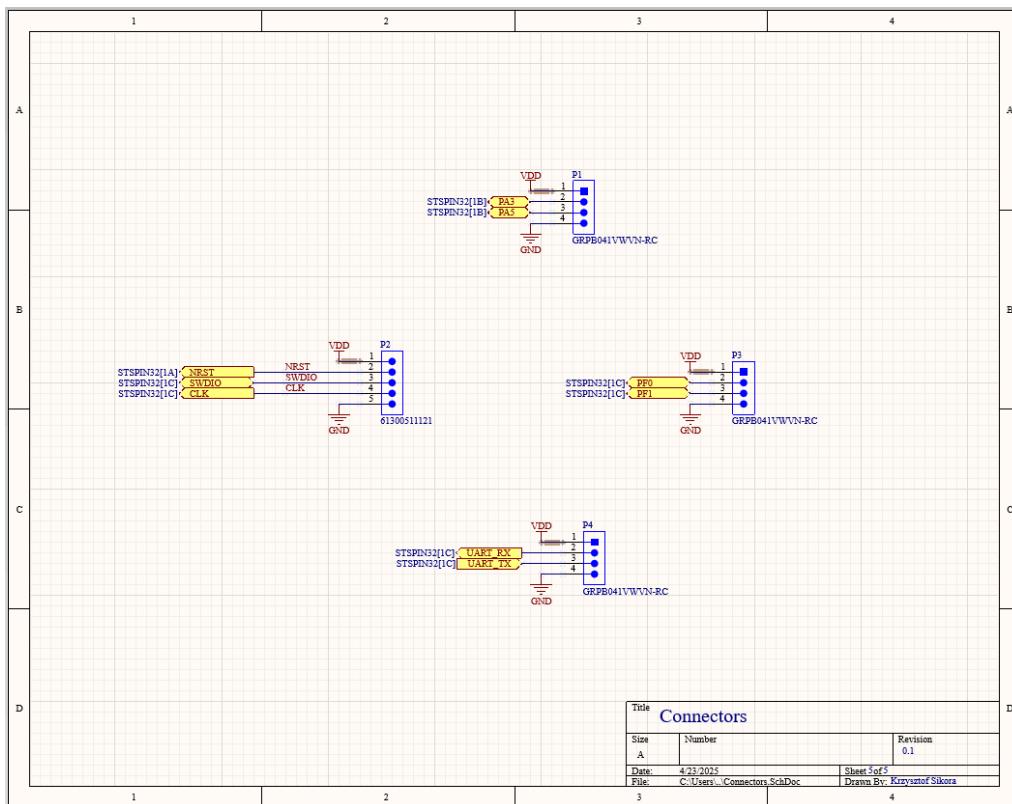


Figure 5. Schemat wyprowadzeń modułu.

2.3. PCB

PCB wykonane zostało przez JLCPCB. Jest to płytka 4-warstwowa, na laminacie FR4 TG 135-140 o grubości całkowitej 1.6 mm. Projekt oparty został o prosty stackup: SIGNAL / GND / POWER / SIGNAL. Dodatkowo, zasilanie do mosfetów poprowadzone zostało również na warstwie dolnej, ze względu na większy prąd (warstwy skrajne mają 1 oz copper, wewnętrzne 0.5 oz copper). Większość elementów znalazła się na górnej warstwie, poza kilkoma elementami, dołączonymi do bramek tranzystorów przełączających i filtracji głównego zasilania. Ma to na celu ułatwić montaż oraz przyszłe testy.

Wejścia zasilania, pady kondensatora filtrującego oraz wyjścia silnika, to pady o rozmiarach 5 mm x 5 mm. Do nich przylutowane zostaną przewody od silnika oraz zasilające z akumulatora. W kolejnej iteracji, pady te można uzupełnić o przelotki, które zapobiegają ich oderwaniu od PCB. Utrudnia to jednak ich częste lutowanie i rozlutowywanie podczas testów, co zwiększa ryzyko przegrzania elementów.

Na warstwie trzeciej (zasilania) widoczne są dwa pola: zewnętrzne - VBUS, oraz wewnętrzne, na którym znajduje się zasilanie 3.3V. Podział taki wynika z ułożenia elementów, a konkretnie układu STSPIN32, który wraz z periferiami jest jedynym układem zasilanym z tego napięcia.

Poza polami na górnej i dolnej warstwie zasilania, ścieżkami o większej szerokości od pozostałych, są ścieżki zasilania (np. 3.3V), a także odpowiadające za ładowanie i rozładowywanie bramek tranzystorów MOSFET.

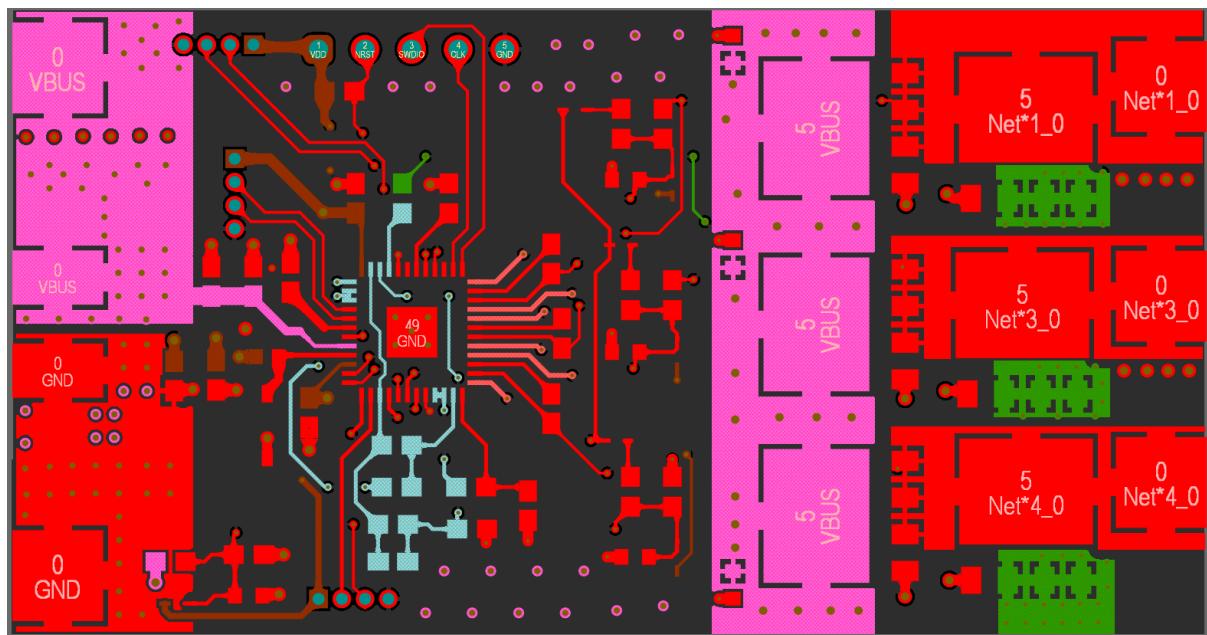


Figure 6. Warstwa góra PCB.

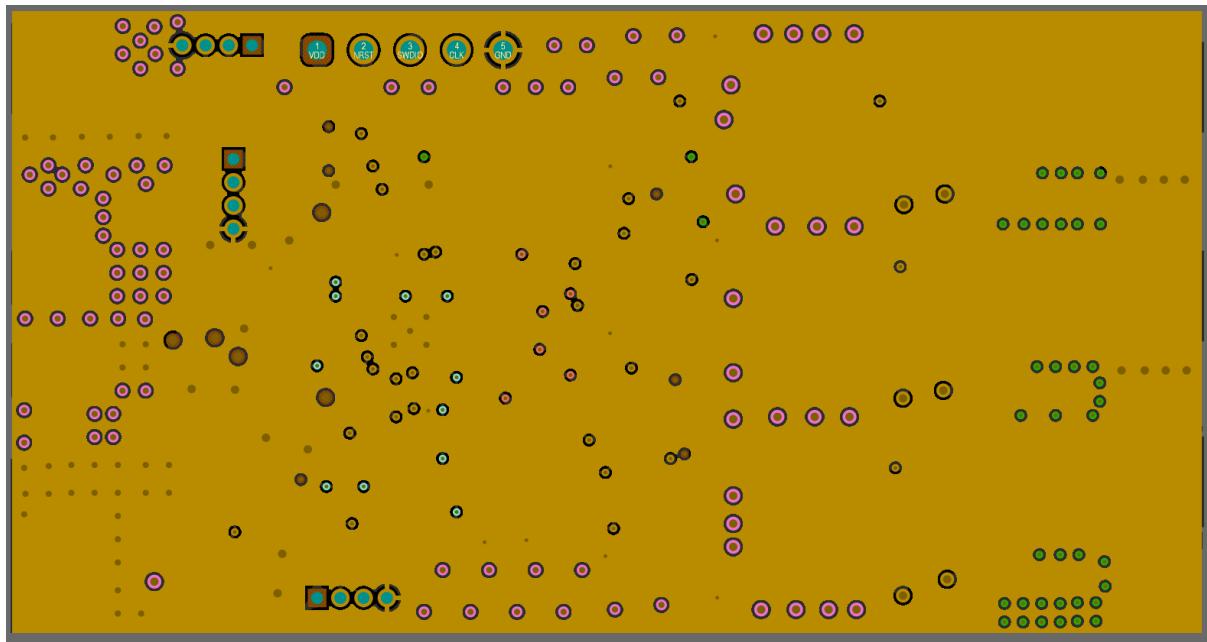


Figure 7. Warstwa druga PCB - GND.

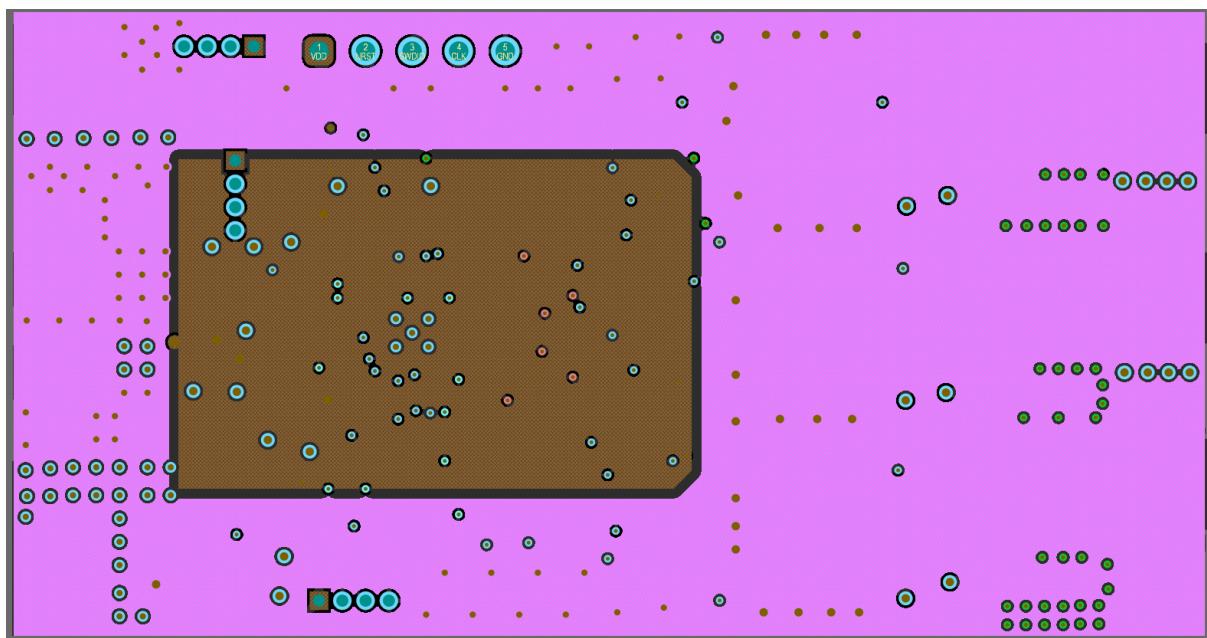


Figure 8. Warstwa trzecia PCB - pola zasilania.

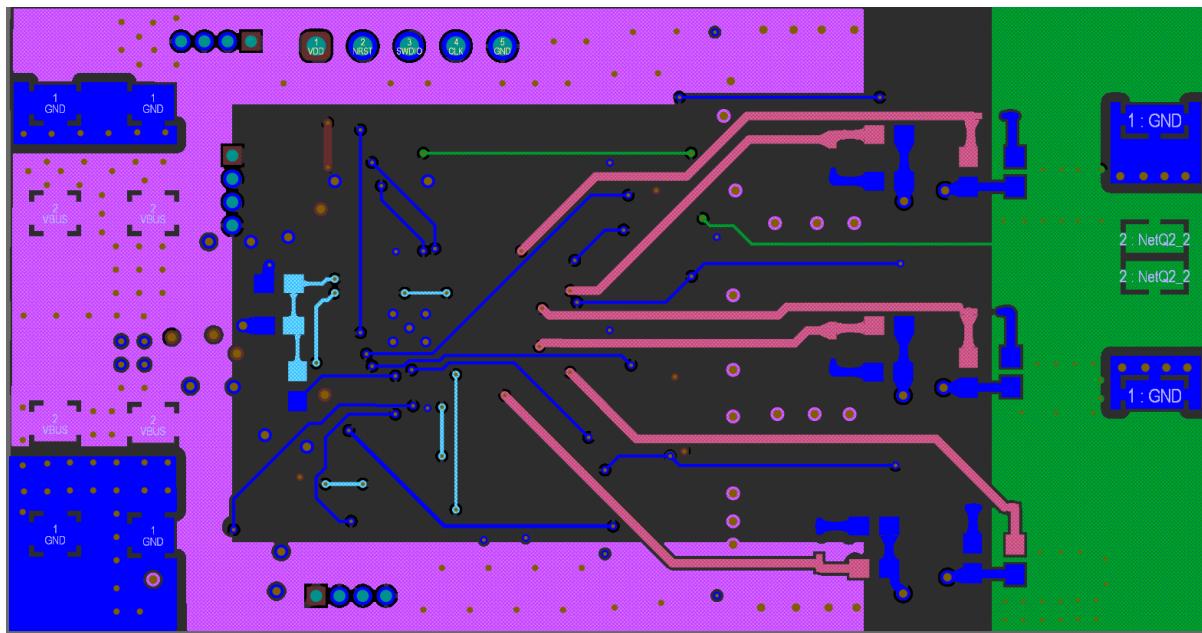


Figure 9. Warstwa dolna PCB.

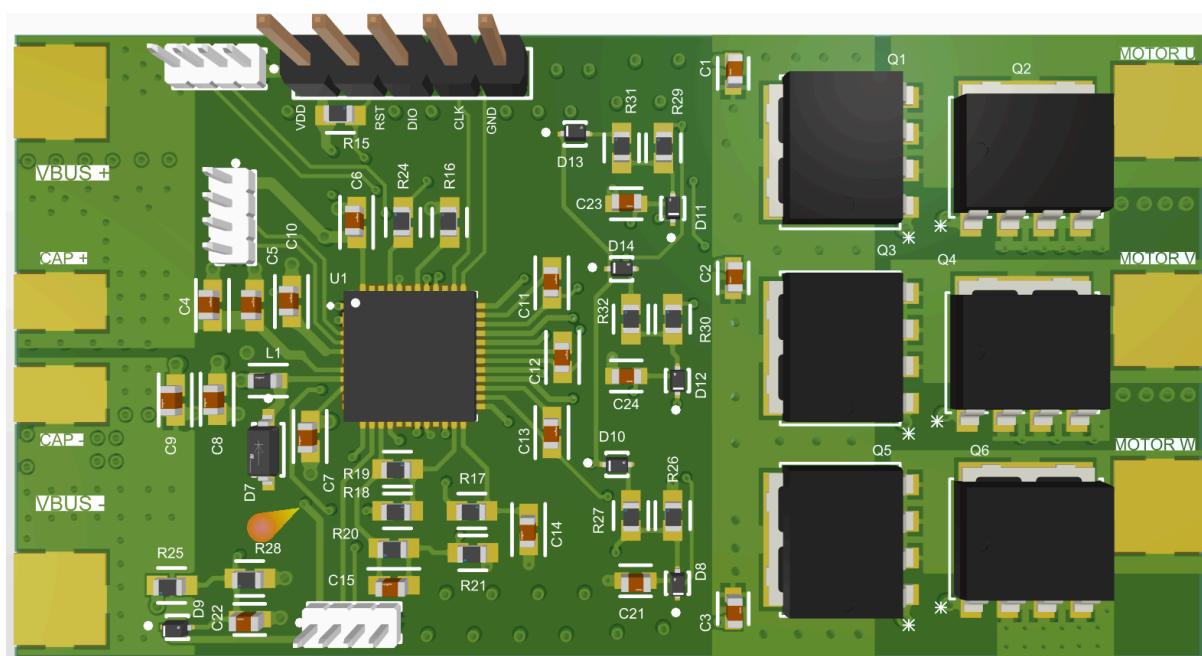


Figure 10. Widok 3D górnej strony PCB.

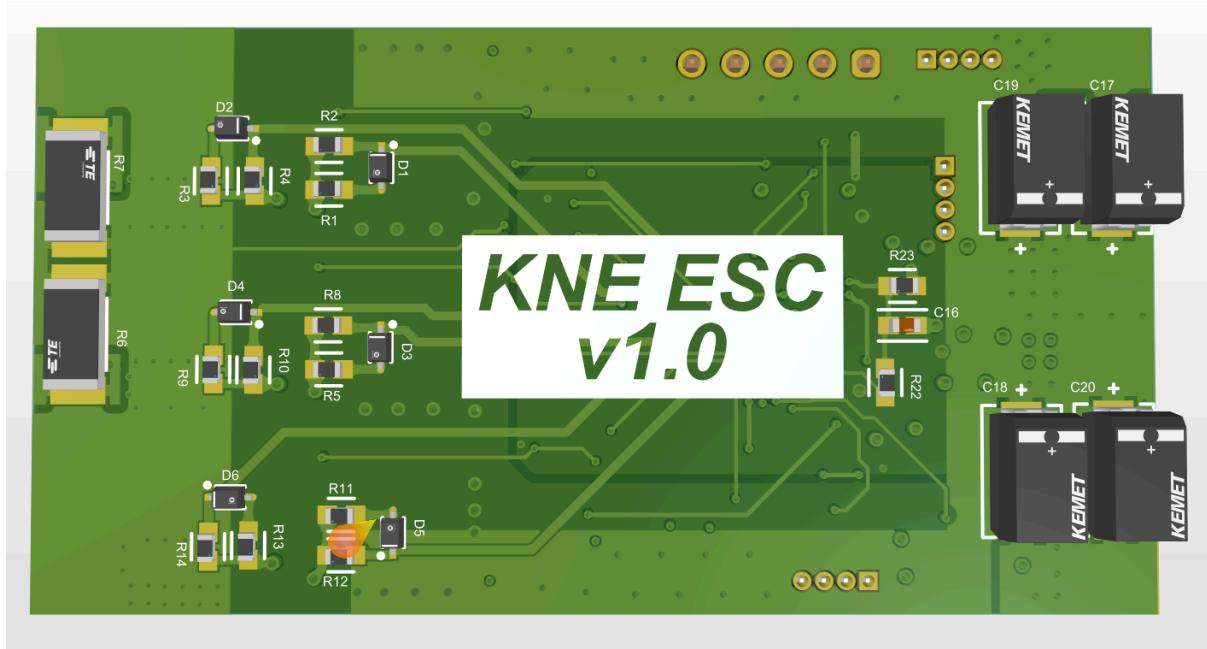


Figure 11. Widok 3D dolnej strony PCB.

3. Oprogramowanie

Przy tworzeniu oprogramowania, wykorzystano STM32 Motor Control Software Development Kit oraz STM32 Cube IDE. Do dobrania parametrów algorytmu sterowania silnikiem wykorzystano oprogramowanie Motor Pilot od STM, które umożliwia monitorowanie RPM silnika oraz modyfikację parametrów w czasie jego działania.

Sterowanie prędkością silnika następuje z wykorzystaniem algorytmu 6-step bez czujnika. Do realizacji tego algorytmu użyto Motor SDK. Jest to narzędzie, które umożliwia skonfigurowanie układu ESC oraz ustawienie parametrów algorytmu. Oprogramowanie to implementuje konieczne komponenty do sterowania silnikiem, przedstawione na rysunku: „Figure 12. Schemat blokowy sterowania silnikiem BLDC.”. Algorytm 6-step opiera się na wymuszeniu przepływu prądu w odpowiednich cewkach silnika (zależnie od jego pozycji), tak, aby uzyskać stabilny moment obrotowy wirnika. Uzyskuje się to z wykorzystaniem tranzystorów, które w każdym z 6 etapów podłączają jedną cewkę do zasilania, kolejną do masy, a ostatnia pozostaje nie podłączona, w kolejności przedstawionej na rysunku „Figure 13. Wizualizacja algorytmu 6-step.”. Do szacowania pozycji wirnika wykorzystany jest pomiar BEMF (ang. Back Electromotive Force) na nie podłączonej w danym momencie cewce oraz pomiar prądu silnika poprzez rezystor bocznikowy. Sterowanie prędkością silnika następuje poprzez zmniejszenie przepływu prądu przez cewki. Efekt ten uzyskuje się przez zastosowanie sygnału PWM do sterowania cewkami. Do ustalenia wartości wypełnienia tego sygnału zastosowany jest regulator PI, który na podstawie oszacowanej prędkości silnika, dostosowuje wypełnienie sygnału PWM, tak aby uzyskać możliwie stabilną zadaną prędkość [9].

Najważniejsze parametry algorytmu dobrano na podstawie artykułu [3] oraz poprzez badanie ich wpływu na działanie silnika z wykorzystaniem oprogramowania Motor Pilot. Działanie algorytmu dostosowano do silnika 7 biegowego, którego maksymalną wartość obrotów na minutę ustalono na 8000. Częstotliwość sygnału PWM, który steruje prądem cewek ustalono na 75 kHz. Wartości prędkości pętli regulatora PI ustalono w następujący sposób: $P = 0.13403$, $I = 0.00018$. Przy takich wartościach stwierdzono stabilną pracę silnika na zadanych prędkościach oraz płynne przechodzenie pomiędzy prędkościami silnika.

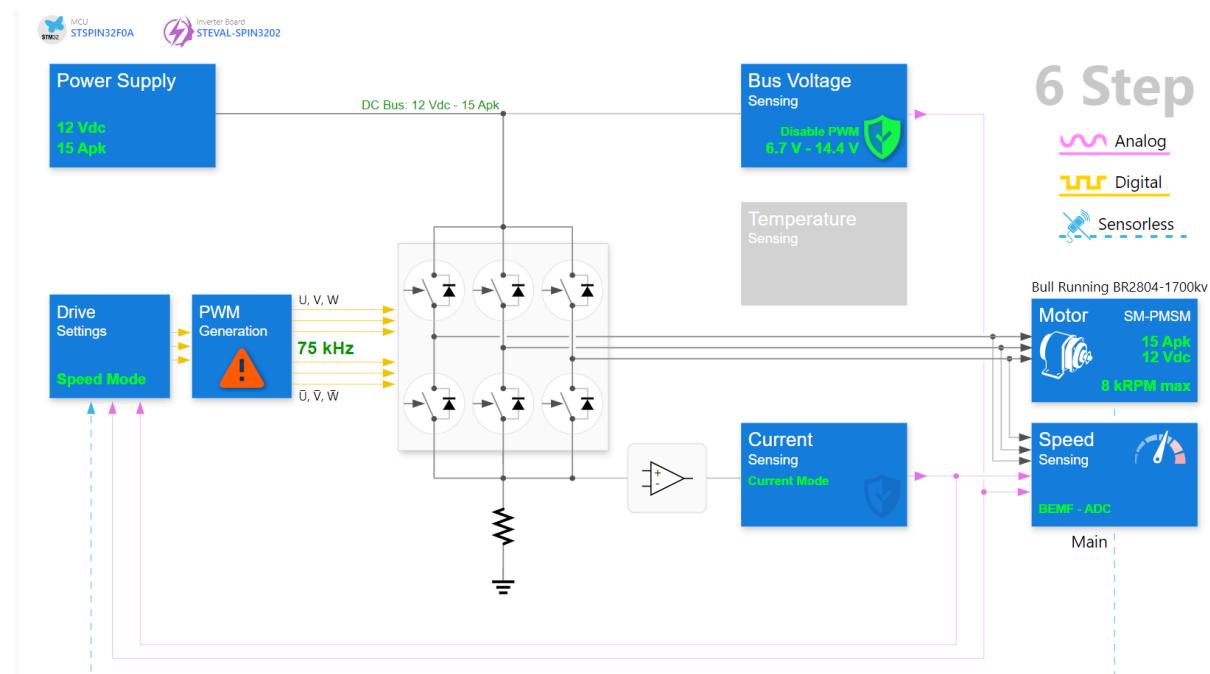


Figure 12. Schemat blokowy sterowania silnikiem BLDC.

Sterowanie prędkością silnika jest możliwe na dwa sposoby: z wykorzystaniem sygnału PWM lub protokołu cyfrowego DShot. Otrzymany PWM powinien mieć częstotliwość równą 50 Hz. Sterowanie prędkością silnika odbywa się przez sterowanie szerokością impulsu w zakresie 1.5 ms do 2 ms, gdzie 1.5 ms oznacza zerową prędkość, natomiast 2 ms oznacza maksymalną prędkość silnika. W przypadku protokołu DShot do ustawienia prędkości jest dostępny zakres wartości od 48 do 2047, gdzie 48 oznacza minimalną prędkość silnika, a 2047 maksymalną. Wartość throttle równa zero oznacza zatrzymanie silnika. Protokół opiera się na wysyłaniu 16-bitowych ramek z odstępem 2 μ s między kolejnymi ramkami. Pierwsze 11 bitów stanowi wartość zadanej prędkości silnika, kolejnym bitem jest bit telemetrii, który służy do uzyskania danych diagnostycznych silnika od ESC. Ostatnie 4 bity są wartością CRC (ang. Cyclic Redundancy Check), służącą do sprawdzenia poprawności przesyłanej ramki. Logiczne zero oraz jedynka kodowane są w tym protokole poprzez wysłanie impulsu o odpowiedniej szerokości. Każdy z bitów stanowi część sygnału PWM o odpowiedniej wartości wypełnienia. Dla jedynki jest to wypełnienie równe 75%, natomiast dla 0 jest to wypełnienie równe 37.5%. Protokół ten występuje w trybach prędkości transmisji: 150 kb/s, 300 kb/s, 600 kb/s oraz 1200 kb/s. Dla najniższej prędkości czas trwania pojedynczego bitu wynosi 6.67 μ s, natomiast całej ramki wynosi 106.72 μ s [8].

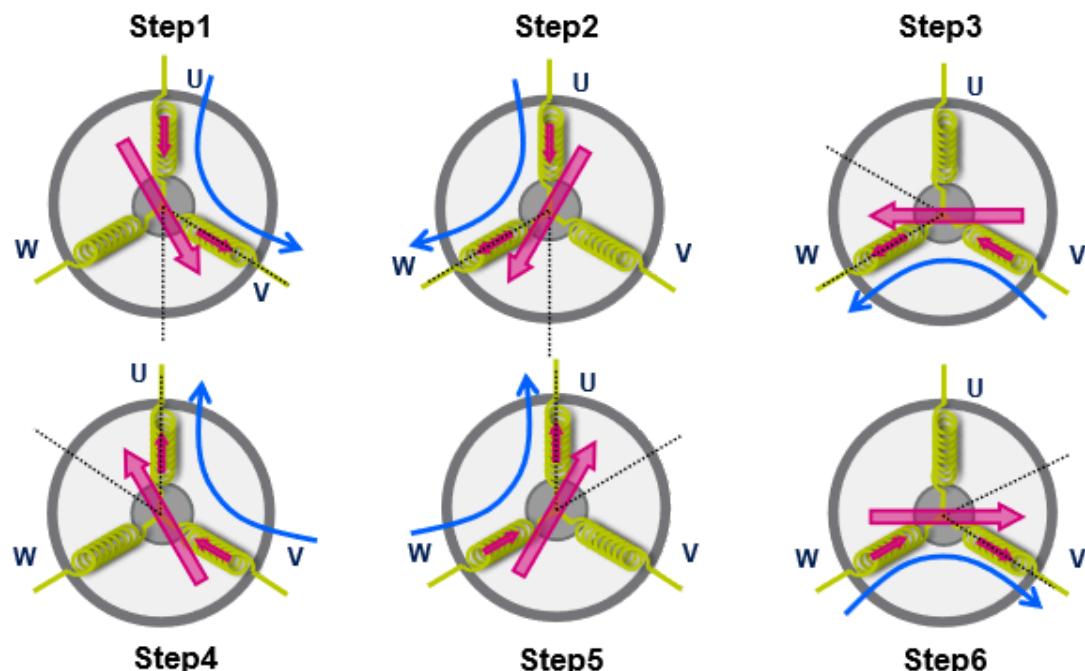


Figure 13. Wizualizacja algorytmu 6-kroku.

Odbiór obydwóch sygnałów zrealizowano poprzez pin GPIO z aktywowanym przerwaniem ustawionym na wyzwalanie zboczem narastającym i opadającym oraz licznik ze skonfigurowaną częstotliwością na tyle dużą, aby móc dokładnie zmierzyć czas trwania impulsu. W trakcie wystąpienia przerwania odczytywany jest stan licznika i zapisywany w buforze. W przypadku protokołu DShot, po odebraniu całej ramki, jest ona dekodowana do postaci pakietu z odpowiednimi wartościami. Następnie sprawdzana jest poprawność pakietu poprzez porównanie obliczonego CRC z tym w pakiecie. Po przeprowadzonej walidacji wartość throttle jest skalowana na RPM i następuje ustawienie prędkości silnika. W przypadku sygnału PWM mierzony jest każdy impuls i jeśli jego długość znajduje się w odpowiednim zakresie oraz jego wartość jest różna od poprzedniej, następuje ustawienie prędkości silnika.

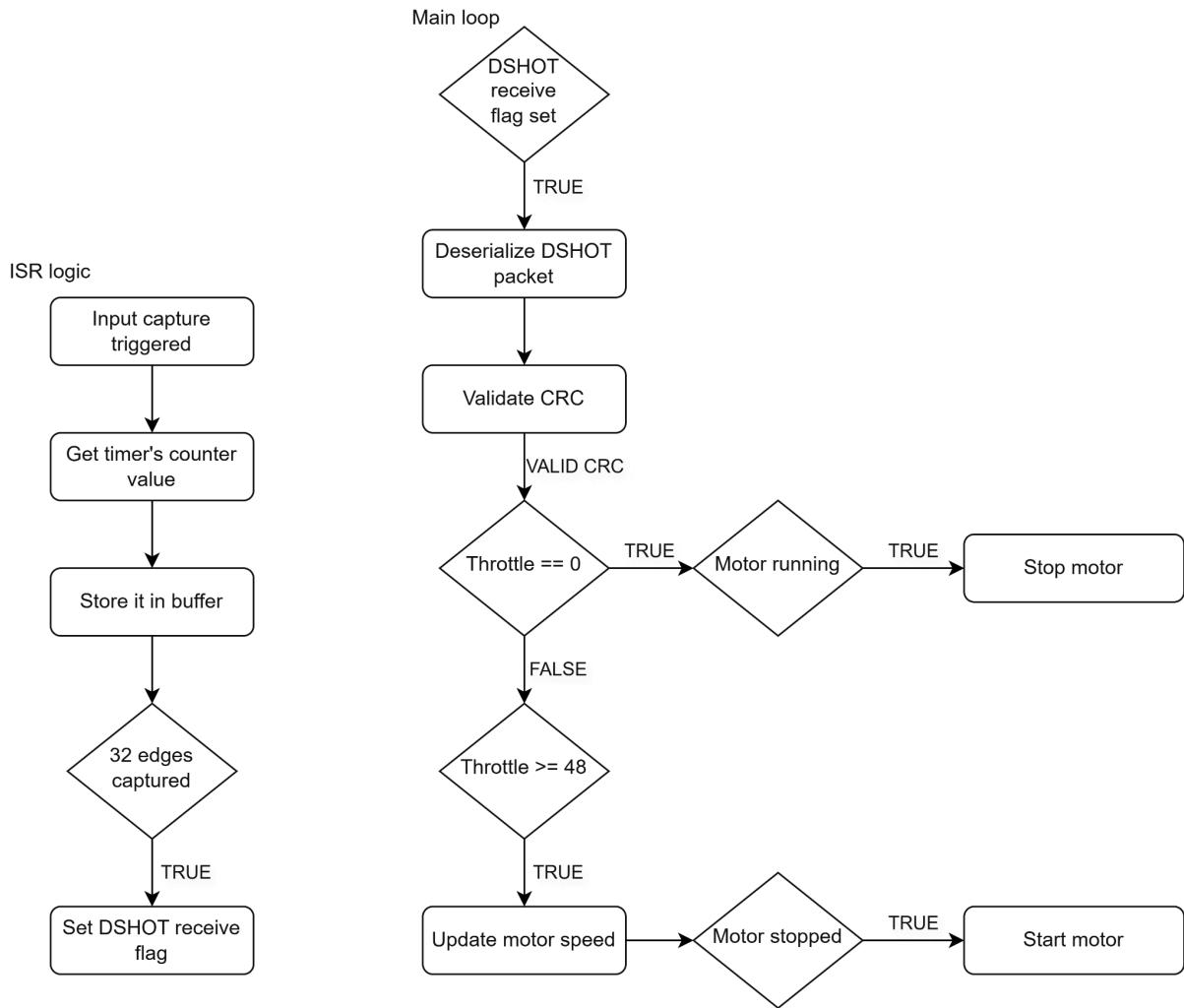


Figure 14. Schemat blokowy sterowania silnikiem przez protokół DSHOT.

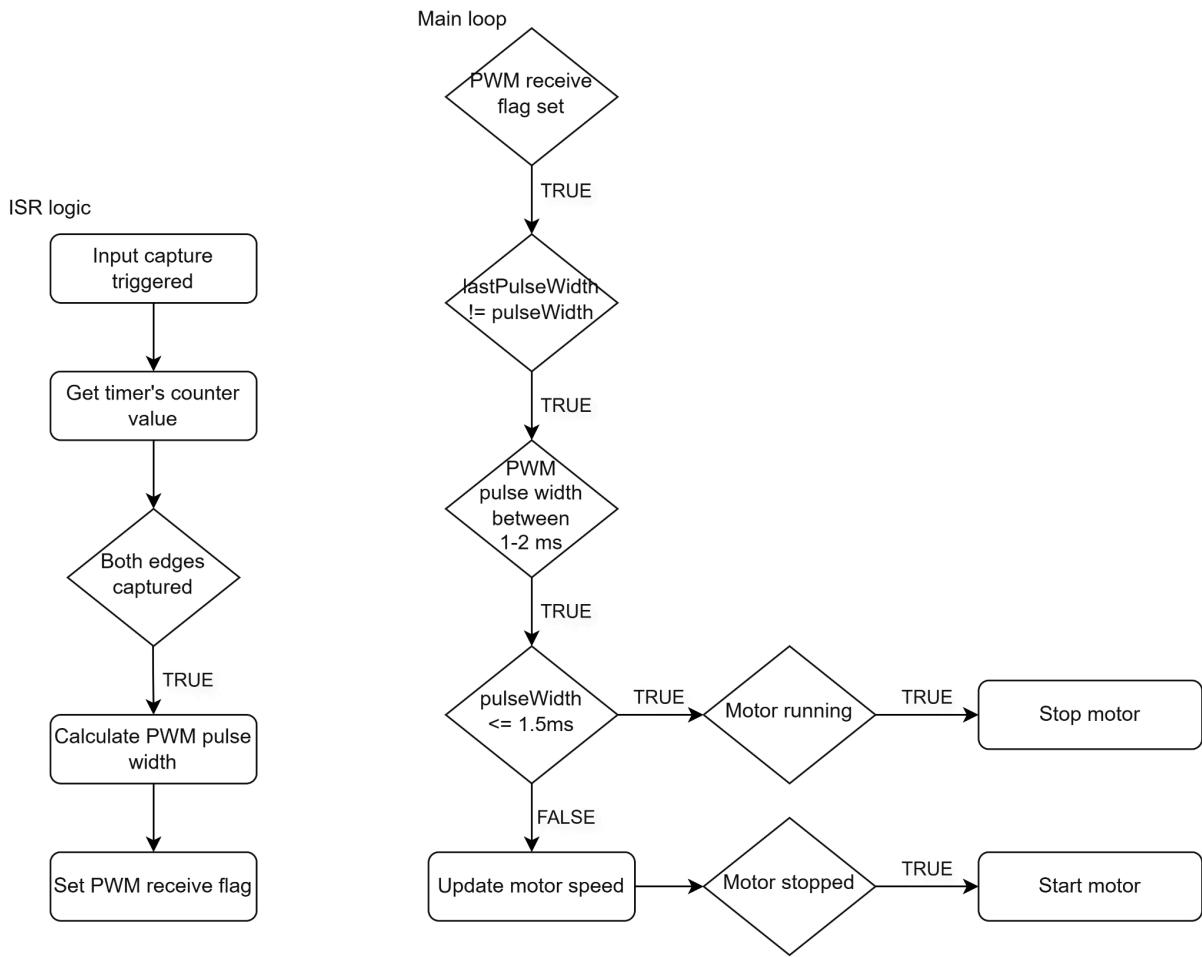


Figure 15. Schemat blokowy sterowania silnikiem przez sygnał PWM.

W celu rozszerzenia komunikacji z ESC zaimplementowano interfejs z wykorzystaniem protokołu UART, który umożliwia uzyskanie dodatkowych danych o silniku. Komunikacja opiera się na 4-bajtowych pakietach o strukturze: jeden bajt na typ pakietu, dwa bajty na dane i jeden bajt CRC. Dodatkowo zaimplementowano też pakiet telemetrii, który zawiera dane o silniku: wypełnienie sygnału PWM sterującego prądem silnika, zadaną prędkość RPM, średnią zmierzona prędkość RPM, obecny stan silnika oraz CRC w celu weryfikacji poprawności danych. Wysłanie pakietu o odpowiednim typie umożliwia otrzymanie pakietu telemetrii. Pakiet telemetrii wysyłany jest również w przypadku otrzymania pakietu DShot z ustawionym bitem telemetrii. Dodatkowo interfejs ten umożliwia ustawienie trybu komunikacji z ESC: PWM, DSHOT lub UART oraz sterowanie prędkością silnika.

4. Uruchomienie i testy

Po zlutowaniu PCB i zaprogramowaniu mikrokontrolera poprzez ST-Link v2, do płytki dołączono silnik BLDC z drona DJI Phantom 2. Jego parametry ustalone zostały już wcześniej, przy pomocy płytki deweloperskiej od ST, aby uniknąć ewentualnego uszkodzenia tranzystorów mocy, bądź układów zabezpieczających, w przypadku braku synchronizacji.

W celu zebrania dodatkowych danych telemetrycznych do dalszej analizy i poprawy pracy silnika, wyprowadzony na płytce interfejs UART podłączono do komputera za pomocą

odpowiedniego konwertera. Pozwala to na logowanie aktualnych parametrów, takich jak np. prędkość zadana silnika i rzeczywista prędkość obrotowa.

Z wykorzystaniem oprogramowania Motor Pilot od STM i interfejsu UART przeprowadzono testy działania silnika poprzez zadawanie określonych prędkości i monitorowanie zmierzonej prędkości silnika. Zweryfikowano poprawność dobranych parametrów regulatora PI. Nie stwierdzono znaczących wahań zmierzonej prędkości silnika od zadanej. Silnik pracuje płynnie w zakresie prędkości 4000 do 8000 RPM. Jak widać na zdjęciu "[Figure 18. ESC z pracującym silnikiem BLDC.](#)" silnik obraca się z zadaną prędkością. Jego start zarówno bez, jak i z obciążeniem jest płynny.

Do testowania układu wykorzystano płytę Bluepill z mikrokontrolerem STM32F103C8T6, z której zadawano sygnał PWM do sterowania prędkością silnika. Wypełnienie sygnału PWM ustawiano poprzez potencjometr w postaci drążka. Pozycję drążka odczytywano przez ADC, która była dekodowana na odpowiednią prędkość silnika, a następnie na odpowiednie wypełnienie. Cały układ testowy jest przedstawiony na zdjęciu "[Figure 19. Układ testowy sterowania silnikiem przez PWM z użyciem płytki Bluepill.](#)" Płytkę zasilana była napięciem 12V (maksymalne napięcie silnika to 3S - 12.9V). Z wykorzystaniem tego samego układu przetestowano sterowanie prędkością silnika przez protokół DShot. Przy dużych prędkościach transmisji stwierdzono problem w odbiorze pakietów, w których były gubione bity.

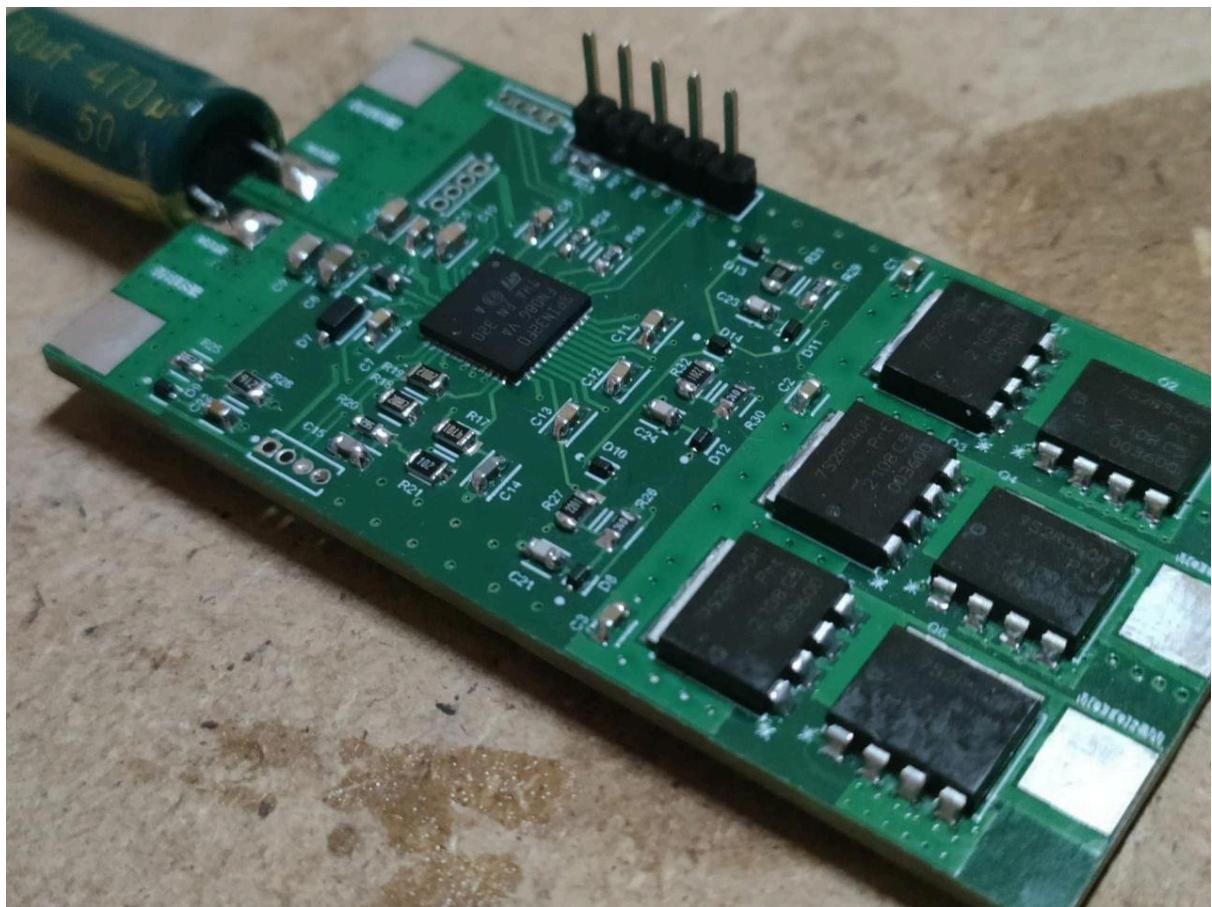


Figure 16. Górná strona zlutowanego PCB.



Figure 17. Dolna strona PCB po zlutowaniu.

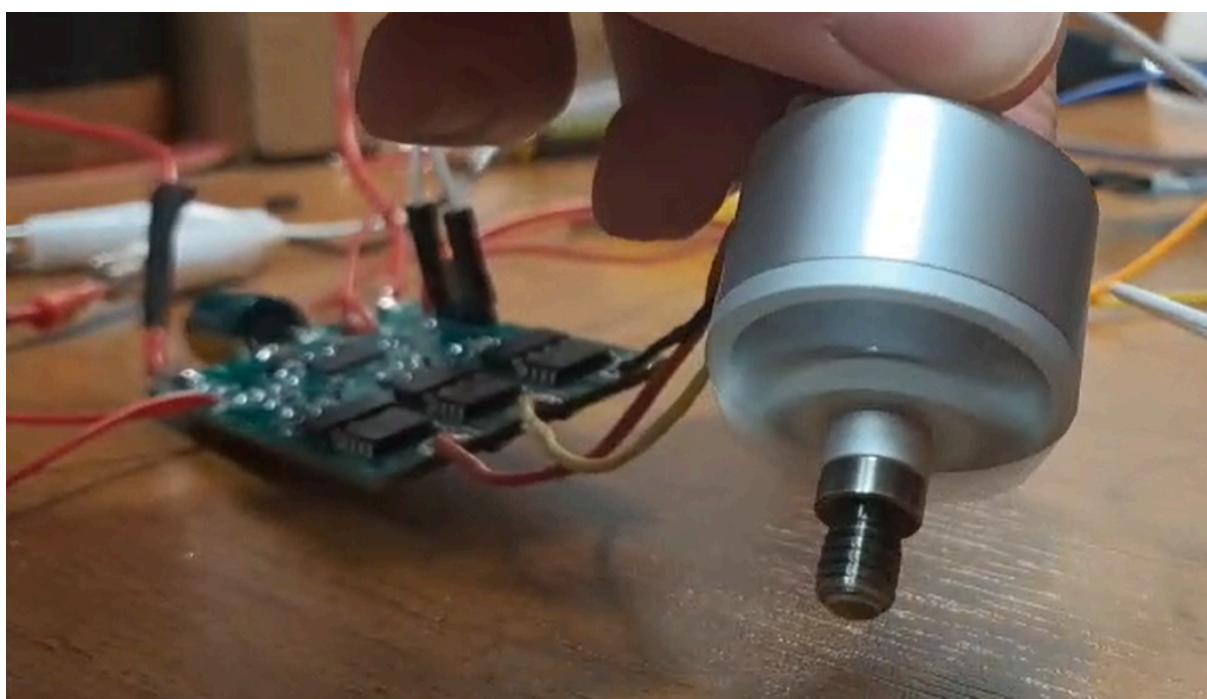


Figure 18. ESC z pracującym silnikiem BLDC.

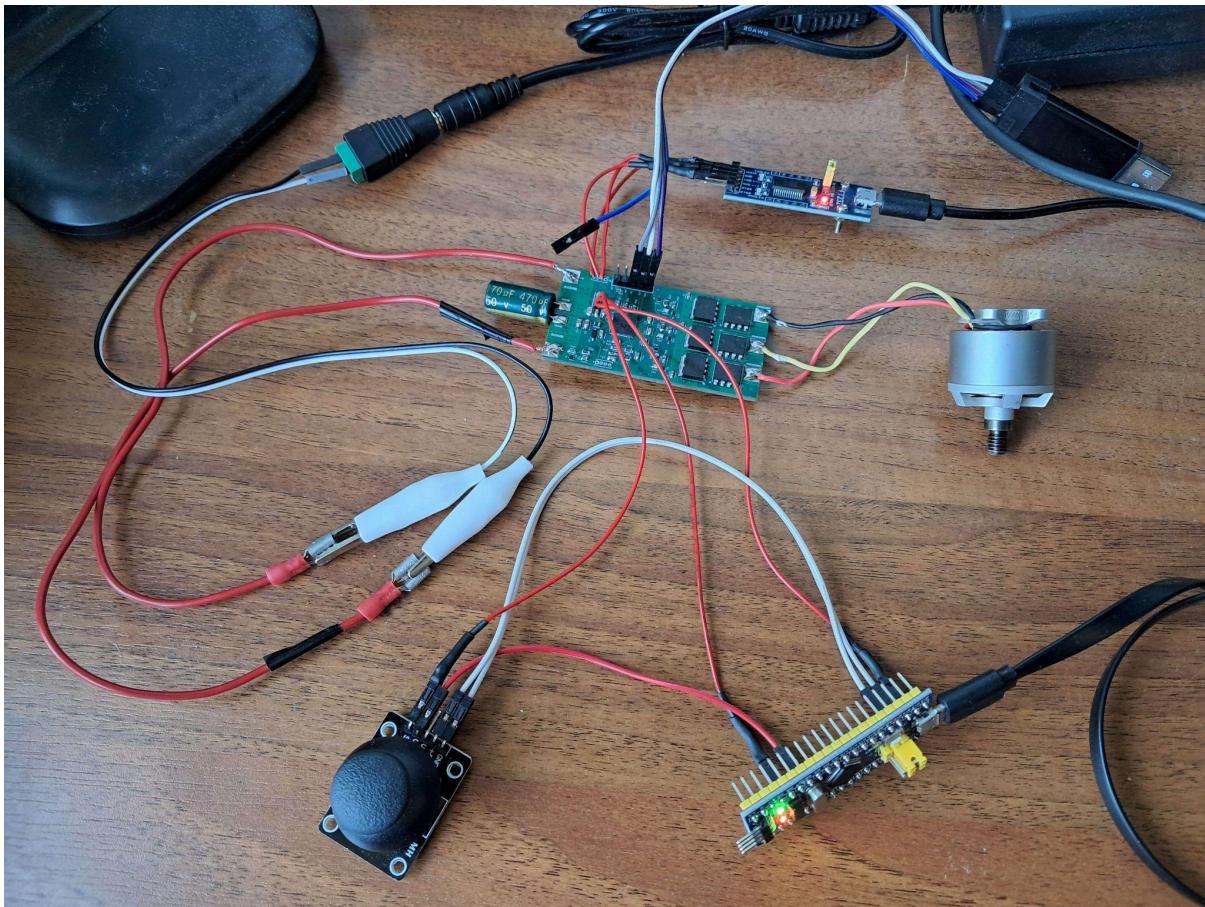


Figure 19. Układ testowy sterowania silnikiem przez PWM z użyciem płytki Bluepill.

Na chwilę obecną nie znaleziono błędów w projekcie PCB, ani oprogramowaniu, które wpływały na pracę urządzenia. Zauważono jednak ograniczenie prędkości interfejsu UART, które utrudnia pracę z oprogramowaniem Motor Pilot od ST. Wynikać może ono zarówno z ułożenia ścieżek na PCB i ewentualnych interferencji, zakłóceń, bądź z podłączenia poprzez zbyt długie przewody połączeniowe. Problem ten wymaga dalszej analizy, nie stanowi jednak przeszkody do stosowania kontrolera w urządzeniach docelowych, tj. dronach i samochodach RC.

Do testowania działania kontrolera stworzono aplikację ESC Pilot, napisaną w języku Python. Komunikuje się ona z ESC przez stworzony interfejs z wykorzystaniem protokołu UART. Pozwala ona na zadanie prędkości obrotowej silnika oraz odczyt danych telemetrycznych, takich jak wypełnienie PWM, prędkość średnia odczytana przez ESC oraz status pracy silnika (RUN, IDLE, STOP, itp.). Interfejs aplikacji przedstawiono na rysunku

Figure 20. Zrzut ekranu aplikacji ESC Pilot.

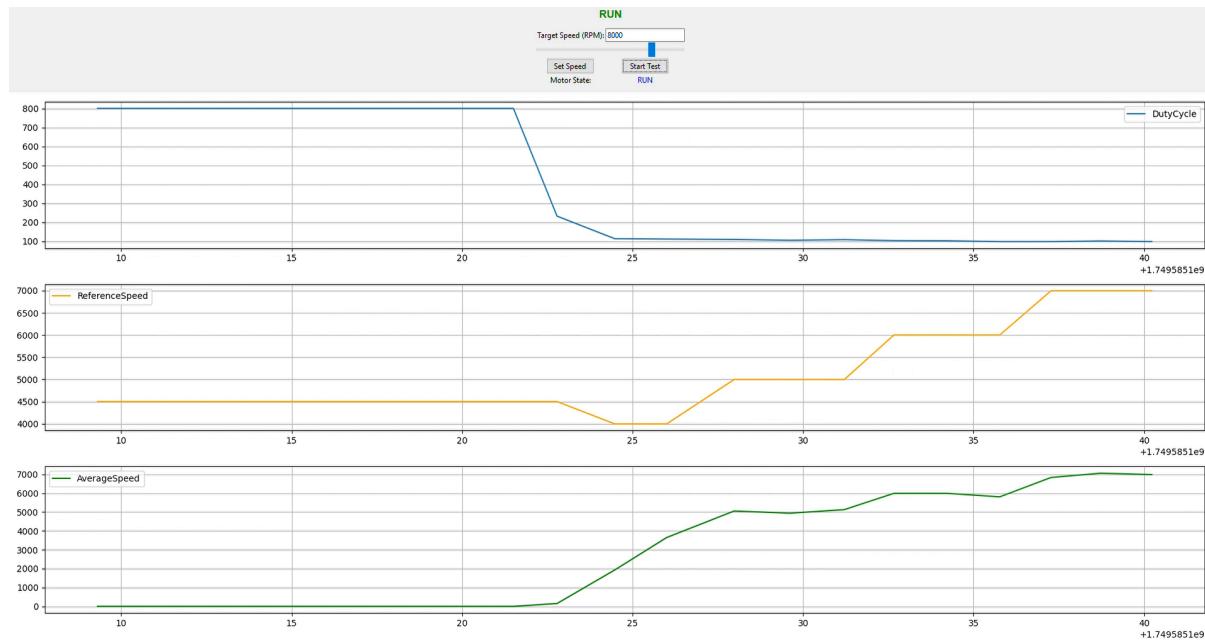


Figure 20. Zrzut ekranu aplikacji ESC Pilot.

Aplikacja ta pozwala na zapis danych telemetrycznych do pliku .csv. Fragment takiego pliku przedstawiono poniżej:

Time [since unix epoch]	DutyCycle	ReferenceSpeed [RPM]	AverageSpeed [RPM]	MotorState
1749584591.8437037	105	6000	6024	RUN
1749584592.9468708	104	6000	6024	RUN
1749584594.0984666	102	6000	5940	RUN
1749584595.1688685	105	6000	6084	RUN
1749584596.2884579	103	6000	5976	RUN
1749584597.3765178	102	6000	5928	RUN
1749584598.4796581	105	6000	5964	RUN
1749584599.599031	110	6000	6270	RUN
1749584600.718403	107	6000	6162	RUN
1749584601.9016843	106	6000	6090	RUN
1749584603.03	102	6000	5934	RUN

6604				
1749584604.14 02366	100	6000	5832	RUN
1749584605.30 79164	102	6000	5922	RUN
1749584606.45 84975	100	6000	5862	RUN
1749584607.59 45804	102	6000	5934	RUN
1749584608.72 9338	104	6000	6006	RUN

Table 2. Przykładowe dane z pliku CSV.

Na podstawie zebranych w ten sposób danych, stworzono wykresy, przedstawiające porównanie prędkości zadanej i odczytanej z ESC podczas zmiany prędkości obrotowej w zakresie 4000 do 8000 RPM w dwóch niezależnych testach.

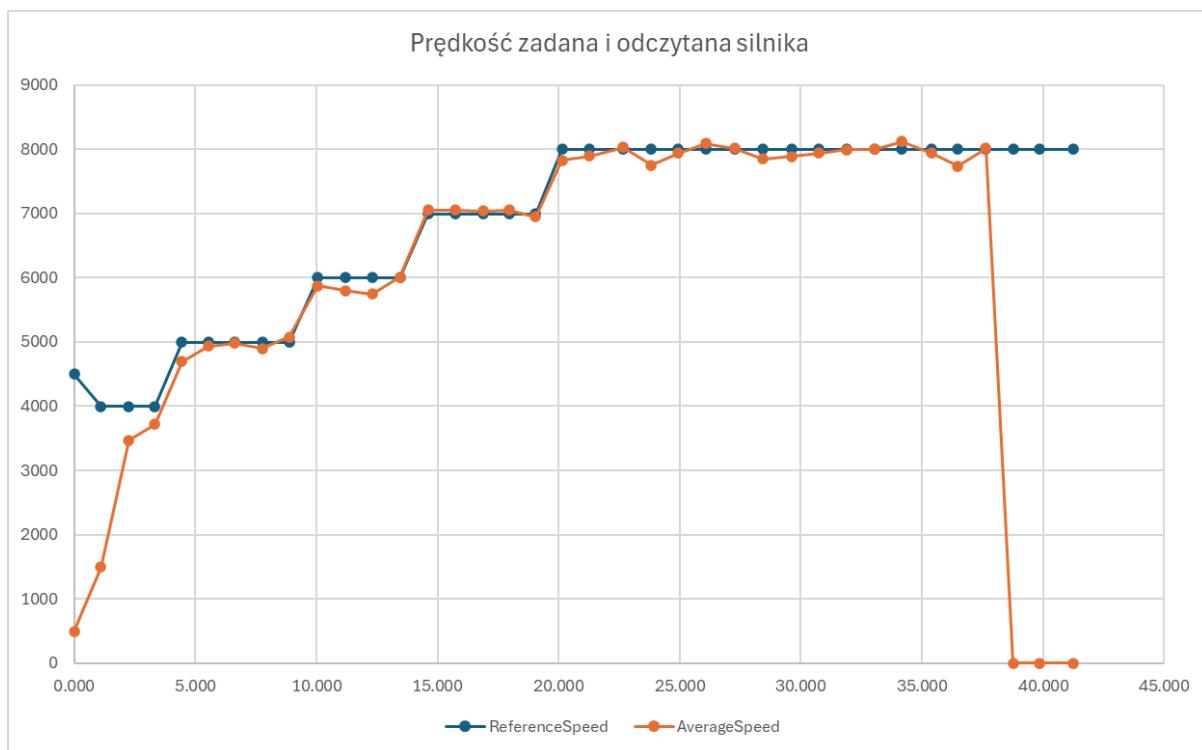


Figure 21. Wykres prędkości zadanej i odczytanej. Próba 1.

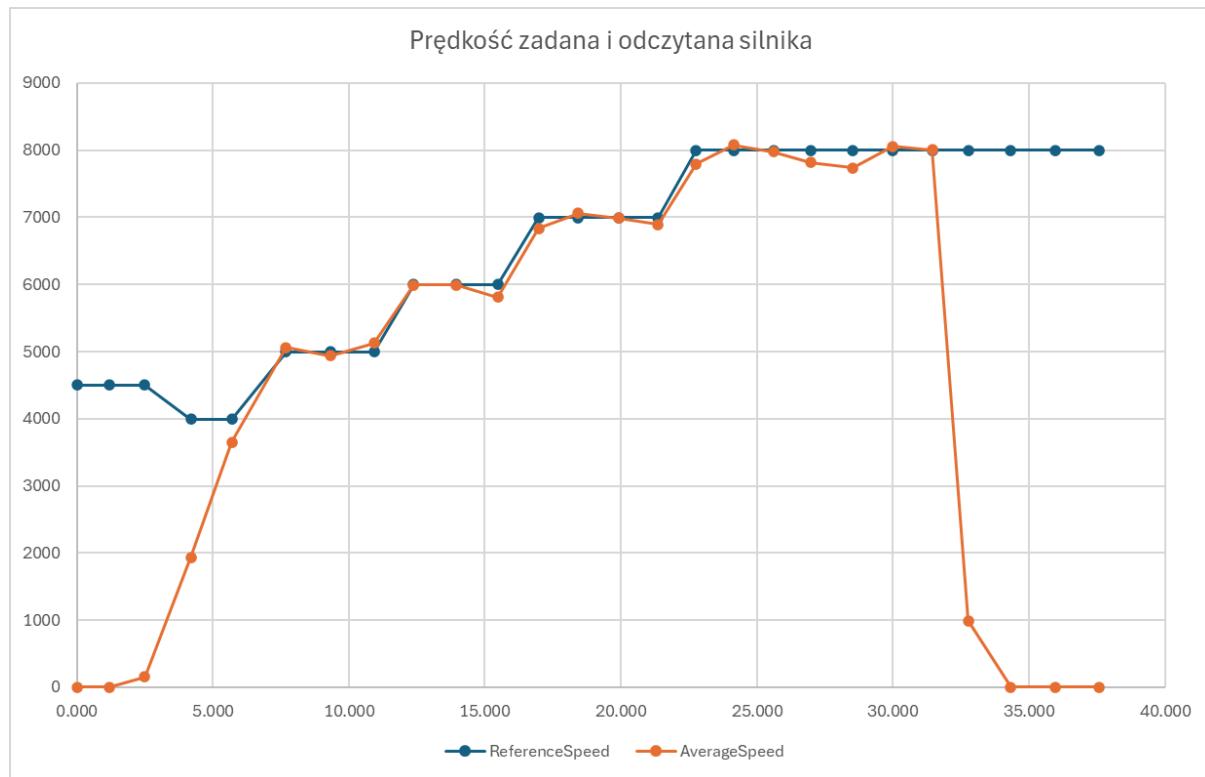


Figure 22. Wykres prędkości zadanej i odczytanej. Próba 2.

Warto nadmienić, że po zatrzymaniu silnika, prędkość zadana nie ulega zmianie na zero, tylko przechowuje ostatnią zadaną prędkość. Ustawiona wartość domyślna (po uruchomieniu ESC) jest równa 4500.

Przeprowadzono także test ze stałą prędkością obrotową - 6000 RPM. Pozwoliło to uzyskać wartości odchyлеń prędkości zmierzonej od zadanej, a także odchylenie standardowe. Dane te przedstawiono w tabeli poniżej.

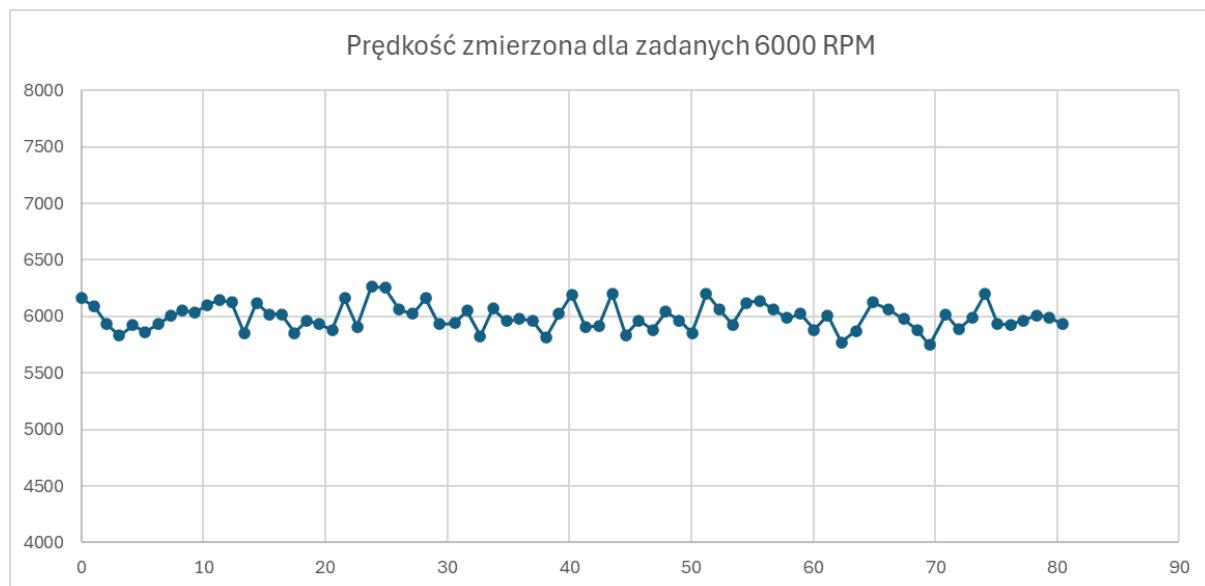


Figure 23. Wykres prędkości odczytanej dla zadanej prędkości 6000 RPM.

Wartość średnia zmierzona [RPM]	5999.494
Odchylenie od wartości zadanej [RPM]	0.506024
Odchylenie od wartości zadanej [%]	0.008434
Maksymalna odchyłka [RPM]	270
Odchylenie standardowe [RPM]	115.2589
Odchylenie standardowe [%]	1.920981

Table 3. Pomiary odchylenia prędkości od zadanej.

Dodatkowo do referencyjnego pomiaru prędkości obrotowej użyto tachometru GM8905, widocznego na rysunku: “*Figure 24. Pomiar prędkości obrotowej silnika tachometrem.*”. Średnia wartość obrotowa zmierzona dla silnika o zadanej prędkości 5000 RPM wyniosła: 4914 RPM. Wartość maksymalna wyniosła: 5004 RPM, natomiast minimalna: 4825 RPM. Dodatkowo zmierzono temperaturę silnika, pracującego z maksymalną prędkością 8000 RPM. Po upływie 5 minut wyniosła ona 34 stopnie Celsjusza. Temperatura tranzystorów w ESC nie przekroczyła 28 stopni Celsjusza.



Figure 24. Pomiar prędkości obrotowej silnika tachometrem.



Figure 25. Pomiar temperatury silnika przy prędkości obrotowej 8000 RPM.

Cała dokumentacja projektu wraz z oprogramowaniem i aplikacją ESC Pilot znajduje się w serwisie GitHub: <https://github.com/Michalek007/ESC>.

Bibliografia

1. Nexperia - PSMNR55-40SSH Datasheet.
<https://assets.nexperia.com/documents/data-sheet/PSMNR55-40SSH.pdf>
Ostatni dostęp: 23.04.2025
2. STMicroelectronics - STSPIN32F0A Datasheet.
<https://www.st.com/resource/en/datasheet/stspin32f0a.pdf>
Ostatni dostęp: 23.04.2025
3. STMicroelectronics - “6-step sensor-less parameter optimization”.
https://wiki.stmicroelectronics.cn/stm32mcu/wiki/STM32MotorControl:6-step_-_Optimization_and_troubleshooting_of_sensor-less_firmware_parameters
Ostatni dostęp: 23.04.2025
4. STMicroelectronics - “Getting started with the motor control six-step firmware example for STEVAL-PTOOL2V1.”
https://www.st.com/resource/en/user_manual/um2788-getting-started-with-the-motor-control-sixstep-firmware-example-for-stevaltool2v1-stmicroelectronics.pdf
Ostatni dostęp: 23.04.2025
5. STMicroelectronics - “Getting started with the STEVAL-SPIN3202 evaluation board, advanced BLDC controller with embedded STM32 MCU.”
https://www.st.com/resource/en/user_manual/um2278-getting-started-with-the-stevalspin_3202-evaluation-board-advanced-bldc-controller-with-embedded-stm32-mcu-stmicroelectronics.pdf
Ostatni dostęp: 23.04.2025
6. Signal integrity journal - “The Myth of Three Capacitor Values.”
<https://www.signalintegrityjournal.com/articles/1589-the-myth-of-three-capacitor-values>
Ostatni dostęp: 23.04.2025
7. Wurth Elektronik - “Effect of layout, vias and design on the blocking quality of filter capacitors”
<https://www.we-online.com/catalog/media/o695199v410%20ANP098a%20EN.pdf>
Ostatni dostęp: 23.04.2025
8. Opis protokołu DShot - <https://brushlesswhoop.com/dshot-and-bidirectional-dshot/>
Ostatni dostęp: 23.04.2025
9. STMicroelectronics - “6-step Firmware Algorithm”
https://wiki.st.com/stm32mcu/wiki/STM32MotorControl:6-step_Firmware_Algorithm
Ostatni dostęp: 23.04.2025