Projekt

Sterowniki robotów

Założenia projektowe

Sterownik lotu drona sterowanego wektorem ciągu „Goose”

TVCG

Skład grupy:
Eryk Możdżeń, 259375

Termin: wtTN19

Prowadzący:
dr inż. Wojciech Domski

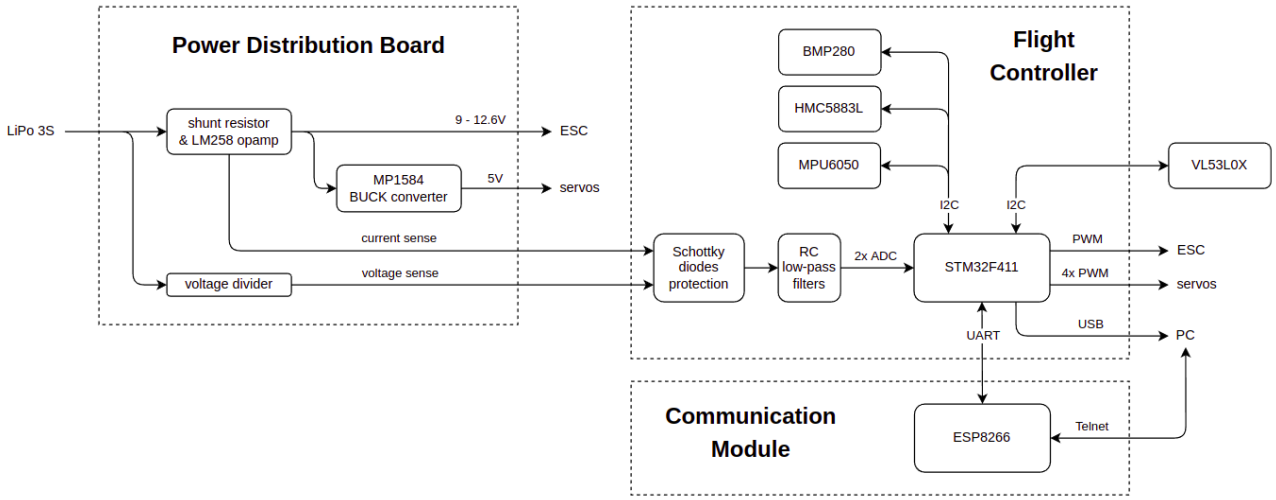
27 marca 2023

Spis treści

1 Opis projektu

Projekt zakłada rozwinięcie oprogramowania sterownika lotu, aktualnie rozwijanego bezzałogowego obiektu latającego „Goose” [5]. Założenia konstrukcyjne oraz funkcjonalność robota zostały stworzone w oparciu o pracę magisterską [4]. Przydatne okazały się być także pracę opisujące modelowanie sił tworzonych przez lotki [2] oraz modelowanie efektu żyroskopowego w kontekście projektu [3].

Do utrzymania się w powietrzu pojazd będzie używał jednego wirnika, wprawianego w ruch za pomocą silnika BLDC. Za pomocą czterech lotek umieszczonych pod śmigłem, sterowanych modelarskimi serwomechanizmami, będzie przekierowywał strumień powietrza tak, aby zachować stabilną pozycję oraz umożliwiać sterowanie lotem drona.



Rysunek 1: Aktualny system złożony z 3 płytek

Power Distribution Board (PDB) odpowiada za zasilanie całego systemu. Umożliwia pomiar aktualnie pobieranego prądu oraz napięcia baterii w postaci analogowego sygnału napięciowego w zakresie około 0 - 3 V.

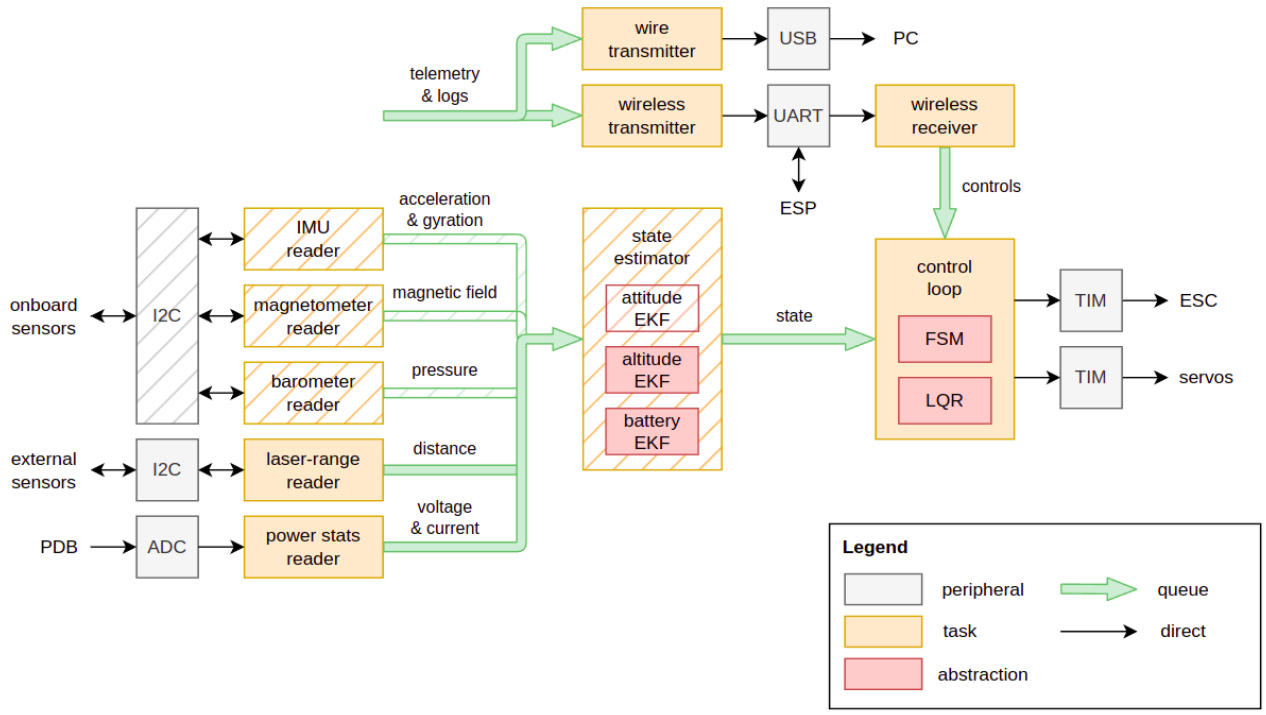
Flight Controller (FC) oparty o mikrokontroler STM32F411[8] jest odpowiedzialny za odczyt wszystkich czujników oraz zadawanie wartości na elementy wykonawcze. Dostępne czujniki:

- IMU MPU6050, zintegrowanego akcelerometru oraz zyroskopu MEMS
- barometru BMP280
- magnetometru HMC5883L
- czujnika odległości TOF VL53L0X[6]
- wartość napięcia baterii
- wartość pobieranego chwilowego prądu

Communication Module to w gruncie rzeczy dostawka do kontrolera lotu zapewniająca komunikację bezprzewodową ze komputerem PC pilota. W tym celu został wykorzystany projekt [1], umożliwiający zdalne programowanie oraz debuggowanie mikrokontrolerów STM32, tworzący z modułu ESP8266 zdalny serwer GDB oraz komunikację UART-Telnet.

2 Funkcjonalności tworzone w ramach projektu

Z uwagi na fakt, że pracę nad projektem zostały rozpoczęte jeszcze przed semestrem letnim, prowadzący zgodził się na ocenę części prac pozostałych do zakończenia projektu. Jako początek prac w ramach przedmiotu uznaje datę 4 marca 2023 (commit 1b35b248). Do tego czasu zostało stworzone oprogramowanie odczytujące dane z czujników umieszczonych na płytce po magistrali I2C oraz implementacja fuzji ich wskazań za pomocą rozszerzonego filtru Kalmana, której wynikiem jest estymacja orientacji. Dane były wysyłane w formacie tekstowym po magistrali USB. Aby zrealizować projekt w pełni, należy rozszerzyć istniejący program do architektury pokazanej niżej 2, dodając następujące funkcjonalności i elementy.



Rysunek 2: Schemat docelowej architektury programu, kreślenie oznacza elementy wykonane przed 4 marca

2.1 Estymacja wysokości nad ziemią

W celu uzyskania estymacji aktualnej wysokości nad ziemią dron skorzysta z fuzji odczytów czujników, takich jak barometr, czujnik odległości TOF, akcelerometr oraz uprzednio wyestymowanej orientacji. Do fuzji zostanie wykorzystany filtr Kalmana. W celu odczytu danych z zewnętrznego VL53L0X będzie konieczne użycie magistrali I2C. Aby odciążyć jednostkę obliczeniową zostanie użyta komunikacja z użyciem przerwań.

2.2 Estymacja poziomu baterii

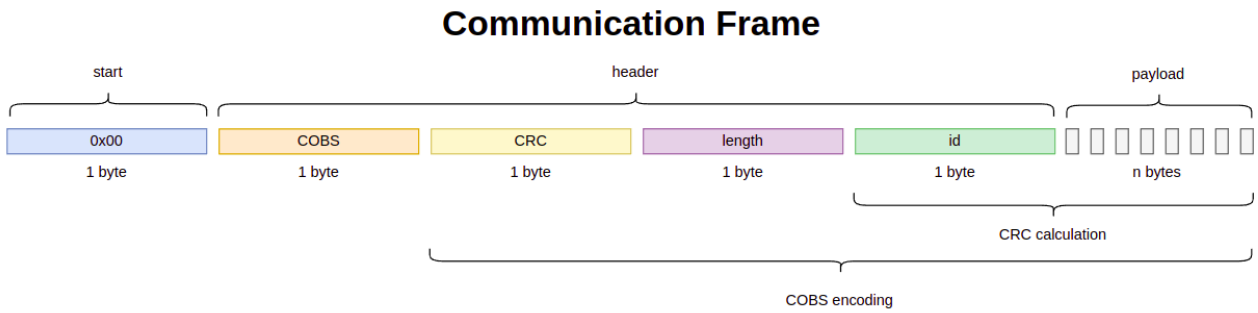
Estymacja poziomu baterii, również osiągnięta filtrem Kalmana, możliwa będzie dzięki pomiarom pobieranego chwilowego prądu oraz napięcia baterii. W celu dokonania pomiaru potrzebna będzie konfiguracja przetwornika analogowo-cyfrowego ADC. W celu odciążenia jednostki obliczeniowej zostanie zastosowany bezpośredni dostęp do pamięci DMA.

2.3 Protokół komunikacji

W celu komunikacji umożliwiającej dwustronną komunikację pomiędzy stacją pilota (PC) a dronem został opracowany binarny protokół komunikacyjny. Ramka danych3 składa się z trzech części: startu, nagłówka oraz ładunku.

Bajt startu o wartości 0 definitywnie oznacza początek nadawania ramki danych. Doświadczenie z poprzednich projektów pokazuje, że jest to mechanizm konieczny do sprawnego dekodowania informacji. Dzięki zastosowaniu algorytmu COBS[9], w modelowym przypadku, zapewnione będzie występowanie wartości 0 wyłącznie na początku ramki danych.

Nagłówek o stałym rozmiarze będzie mieścił w sobie informacje takie jak nadmiarowy bit COBS, suma kontrolna CRC-8[10], długość ładunku danych oraz jednobajtowy identyfikator ramki. Dzieki zastosowaniu sumy kontrolnej będzie możliwa prosta walidacja odebranych danych.



Rysunek 3: Zaproponowana ramka danych

2.4 Komunikacja przewodowa i bezprzewodowa

Docelowo, kontroler lotu powinien transmitować informacje liczbowe w systemie (pomiary z czujników, estymaty stanu, sterowania) oraz komunikaty tekstowe dwoma drogami bezprzewodowo i jeśli jest taka możliwość – przewodowo.

Podczas normalnej pracy, mikrokontroler powinien wysyłać i odczytywać dane z interfejsu UART z użyciem DMA, stosując protokół 3. Użytkownik powinien mieć także możliwość odczytu danych dzięki wirtualnemu portowi szeregowemu USB stworzonego przy pomocy biblioteki od ST[7].

2.5 Sterowanie w zamkniętej pętli

Głównym zadaniem w systemie (widocznym na 2 o nazwie „control loop") będzie task odpowiedzialny za analizę obecnego stanu obiektu, odczyt informacji przychodzących oraz zadawanie sterowań w zamkniętej pętli sprzężenia. Sterowanie będzie zrealizowane za pomocą regulatora LQR. Jako, że do sterowania dronem będzie potrzebne 5 sygnałów PWM (4 lotki + wirnik) konieczne będzie wykorzystanie dwóch odpowiednio skonfigurowanych układów liczników. Jednocześnie wykorzystana będzie maszyna stanów, mająca piecze nad zachowaniem systemu i pełniąca rolę zabezpieczenia. Maszyna stanów powinna gwarantować przewidywalną reakcję na zdarzenia takie jak utrata danych o obiekcie (awaria czujników), brak informacji o sterowaniu (zerwanie połączenia) itp.

3 Zakres prac

Podcele wymagane do zakończenia projektu zostały wymienione w tabelach 1 oraz 2.

Eryk Możdżeń
Opracowanie protokołu komunikacji
Testy protokołu komunikacji
Integracja nowego protokołu do systemu
Implementacja tasków wysyłających i odbierających z UART
Oprogramowanie i doczyt VL53L0X po I2C
Oprogramowanie i odczyt prądu i napięcia po ADC

Tabela 1: Prace do wykonania – etap II

Eryk Możdżeń
Opracowanie fuzji czujników do estymacji wysokości nad ziemią
Opracowanie fuzji czujników do estymacji poziomu rozładowania baterii
Implementacja abstrakcyjnej maszyny stanów
Stworzenie diagramu maszyny stanów drona
Implementacja taska „control loop”
Oprogramowanie liczników w trybie PWM

Tabela 2: Prace do wykonania – etap III

4 Opis kamieni milowych

4.1 Sprawny system komunikacji (do 6 kwietnia)

Na tym etapie powinna istnieć pełna architektura programu odpowiedzialna za komunikację z dronem. Istniejące funkcjonalności powinny pozwalać na dwustronny przesył danych za pomocą dostępnym mediów.

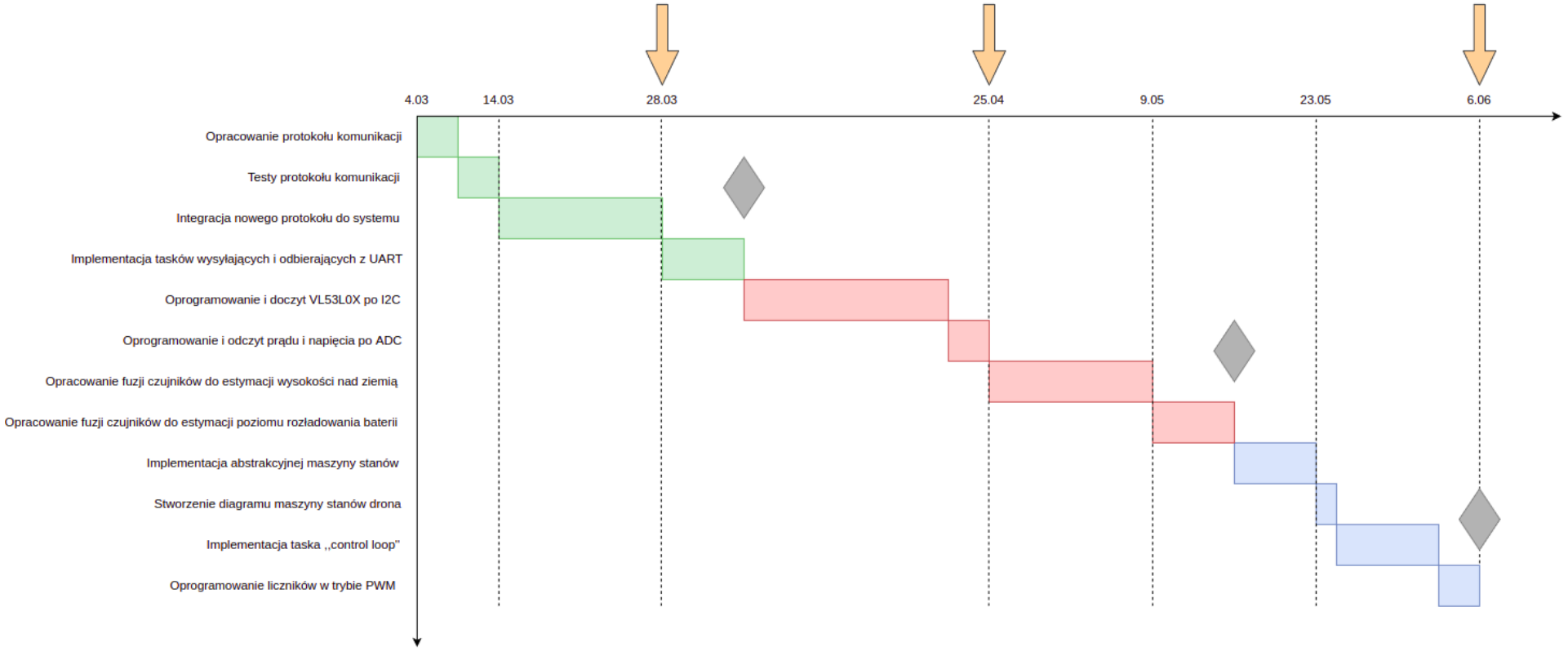
4.2 Estymacja pełnego stanu obiektu (do 16 maja)

Bezzałogowiec powinien mieć dostęp do informacji o swoim pełnym stanie tzn. orientacja (wyrażona w kwaternionie), wysokość lotu (w metrach nad poziomem podłoża) oraz poziom rozładowania baterii (w procentach).

4.3 Sterowanie w zamkniętej pętli (do 6 czerwca)

Program drona, oparty o maszynę stanu, umożliwia sterowanie obiektem w zamkniętej pętli. Reaguje na informacje wysyłane przez pilota i zmianę swojego stanu. Program ma zdefiniowane zachowanie w przypadku utraty połączenia z pilotem oraz przy uszkodzeniu czujników.

5 Wykres Gantta



Rysunek 4: Wykres Gantta projektu, umowny początek na 4 marca,
strzałki oznaczają ostateczne terminy zdawania etapów, romby – kamienie milowe

6 Konfiguracja mikrokontrolera

6.1 Zegary

Do generacji sygnałów zegarowych stała użyta pętla PLL, wykorzystująca zewnętrzny rezonator kwarcowy o częstotliwości 8 MHz.Listing 1 prezentuje kod konfigurujący pętlę PLL oraz preskalery za pomocą biblioteki HAL. HCLK rdzenia osiąga częstotliwość 96 MHz. Wszystkie linie zegarowe peryferiów również osiągają 96 MHz, z wyjątkiem linii APB1 dla której częstotliwość wynosi 48 MHz.

```
1  RCC_OscInitTypeDef oscillator;
2  oscillator.OscillatorType = RCC_OSCILLATORTYPE_HSE;
3  oscillator.HSEState = RCC_HSE_ON;
4  oscillator.PLL.PLLState = RCC_PLL_ON;
5  oscillator.PLL.PLLSource = RCC_PLLSOURCE_HSE;
6  oscillator.PLL.PLLM = 4;
7  oscillator.PLL.PLLN = 96;
8  oscillator.PLL.PLLP = RCC_PLLP_DIV2;
9  oscillator.PLL.PLLQ = 4;
10
11 RCC_ClkInitTypeDef clock;
12 clock.ClockType = RCC_CLOCKTYPE_HCLK | RCC_CLOCKTYPE_SYSCLK |
       RCC_CLOCKTYPE_PCLK1 | RCC_CLOCKTYPE_PCLK2;
13 clock.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
14 clock.AHBCLKDivider = RCC_SYSCLK_DIV1;
15 clock.APB1CLKDivider = RCC_HCLK_DIV2;
16 clock.APB2CLKDivider = RCC_HCLK_DIV1;
17
18 HAL_RCC_OscConfig(&oscillator);
19 HAL_RCC_ClockConfig(&clock, FLASH_LATENCY_3);
```

Listing 1: konfiguracja pętli PLL i preskalerów

6.2 Piny IO

Numer pinu	Pin	Tryb pracy	Funkcja
5	PH0		RCC OSC IN
6	PH1		RCC OSC OUT
20	PA4	Analog	ADC1 IN4
21	PA5	Analog	ADC1 IN5
26	PB0	Alternate, Push-Pull	TIM3 CH3
27	PB1	Alternate, Push-Pull	TIM3 CH4
28	PB2		BOOT1
29	PB10	Output, Push-Pull	GPIO
37	PC6	Alternate, Push-Pull	TIM3 CH1
38	PC7	Alternate, Push-Pull	TIM3 CH2
40	PC9	Alternate, Open Drain	I2C3 SDA
41	PA8	Alternate, Open Drain	I2C3 SCL
42	PA9	Alternate, Push-Pull	TIM1 CH2
44	PA11	Alternate, Push-Pull	USB D –
45	PA12	Alternate, Push-Pull	USB D +
46	PA13		SWDIO
49	PA14		SWCLK
50	PA15	Input	External Interrupt
57	PB5	Input	External Interrupt
58	PB6	Alternate, Push-Pull	UART1 TX
59	PB7	Alternate, Push-Pull	UART1 RX
61	PB8	Alternate, Open Drain	I2C1 SCL
62	PB9	Alternate, Open Drain	I2C1 SDA

Tabela 3: Konfiguracja pinów mikrokontrolera

6.3 UART

UART używany będzie do dwustronnej komunikacji między ESP8266 a STM32.

Baud Rate	115200
Word Length	8
Parity	None
Stop Bits	1

Tabela 4: Konfiguracja peryferium UART1

6.4 I2C

I2C stosowane jest do komunikacji z czujnikami. I2C1 5 przeznaczone jest dla czujników umieszczonych na płytce PCB z mikrokontrolerem (magnetometr, IMU oraz barometr). I22C3 6 przeznaczone jest dla czujników po za płytką (czujnik odległości).

Clock Speed	100000
Duty Cycle	2
Adressing Mode	7 bit
Dual Adressing Mode	disable
General Call Mode	disable
No Stretch Mode	disable

Tabela 5: Konfiguracja peryferium I2C1

Clock Speed	100000
Duty Cycle	2
Adressing Mode	7 bit
Dual Adressing Mode	disable
General Call Mode	disable
No Stretch Mode	disable

Tabela 6: Konfiguracja peryferium I2C3

6.5 USB

W celu przesyłu danych pomiędzy mikrokontrolerem a komputerem został wykorzystany interfejs USB (PCD skonfigurowane w tabeli 7). Gdy nie jest potrzebny debugger jest także wykorzystywany do wgrywania programu za pomocą fabrycznego bootloadera z użyciem trybu DFU.

Speed	full speed
DMA Enable	disable
PHY Interface	embedded
SOF Enable	disable
Low Power Enable	disable
LPM Enable	disable
VBUS Sensing Enable	disable
Use Dedicated EPL	disable

Tabela 7: Konfiguracja peryferium PCD do USB OTG

6.6 Układy licznikowo-czasowe

W projekcie będą używane trzy układy timerów. TIM11 11 wykorzystany jest jako podstawa czasu dla biblioteki HAL (SysTick wykorzystywany jest przez FreeRTOS). TIM1 8 oraz TIM3 9 generują przebiegi PWM o częstotliwości 50 Hz (standard modelarski) wymagane do sterowania serwomechanizmami oraz sterownika silnika BLDC (ESC). Każdy kanał PWM jest skonfigurowany tak jak w tabeli 10.

Period	1999
Prescaler	959
Clock Division	1
Counter Mode	up

Tabela 8: Konfiguracja peryferium TIM1

Period	1999
Prescaler	959
Clock Division	1
Counter Mode	up

Tabela 9: Konfiguracja peryferium TIM3

OCMode	PWM1
Pulse	0
OC Polarity	high
OC Fast Mode	disable

Tabela 10: Konfiguracja pojedynczego kanału PWM

Period	999
Prescaler	95
Clock Division	1
Counter Mode	up

Tabela 11: Konfiguracja peryferium TIM11

6.7 Przetwornik analogowo-cyfrowy

ADC 12 będzie używane do pomiaru napięcia baterii oraz (pośrednio) prądu pobieranego przez układ. Konfiguracja kanałów dostępna jest w tabelach 13 i 14. DMA skonfigurowane jest za pomocą ustawień w tabeli 15.

Clock Prescaler	div 8
Resololution	12 bitów
Scan Conversion Mode	enable
Continous Conversion Mode	enable
Discontinous Conversion Mode	disable
External Trigger Conversion Edge	none
External Trigger Conversion	software start
Data Align	right
Number of Conversions	2
DMA Continous Request	enable
EOC Selection	single

Tabela 12: Konfiguracja peryferium ADC1

Channel	IN4
Sample Time	480 cycles
Rank	1
Offset	0

Tabela 13: Konfiguracja kanału ADC1 IN4

Channel	IN5
Sample Time	480 cycles
Rank	1
Offset	0

Tabela 14: Konfiguracja kanału ADC1 IN5

Mode	Circular
Data Width	Half World

Tabela 15: Konfiguracja kanału DMA2 Stream 0

Bibliografia

[1] *Blackmagic Wireless SWD Debug probe hosted on esp-idf SDK (for ESP8266) with UART on Telnet port and HTTP using xterm.js*. URL: https://github.com/walmis/blackmagic-espidf.

[2] Christoffer Carholt i in. "Design, Modelling and Control of a Single Rotor UAV". W: czer. 2016. DOI: 10.1109/MED.2016.7536015.

[3] Victor H. Dominguez i in. "Micro Coaxial Drone: Flight Dynamics, Simulation and Ground Testing". W: *Aerospace* 9.5 (2022). ISSN: 2226-4310. DOI: 10.3390/aerospace9050245. URL: https://www.mdpi.com/2226-4310/9/5/245.

[4] Emil Bjerregaard Jacobsen. *Modelling and Control of Thrust Vectoring Mono-copter*. 2020. URL: https://projekter.aau.dk/projekter/files/421577367/Master_Thesis_Emil_Jacobsen_v5.pdf.

[5] Eryk Możdżeń. *UAV TVC Goose*. URL: https://github.com/Eryk-Mozdzen/uav-tvc-goose.git.

[6] STMicroelectronics. *DS11555 VL53L0X Time-of-Flight ranging sensor*. 2022. URL: https://www.st.com/en/imaging-and-photonics-solutions/vl53l0x.html#documentation.

[7] STMicroelectronics. *Provides the USB Device library part of the STM32Cube MCU Component "middleware"for all STM32xx series*. URL: https://github.com/STMicroelectronics/stm32_mw_usb_device.

[8] STMicroelectronics. *RM0383 Reference manual STM32F411xC/E advanced Arm® -based 32-bit MCUs*. 2018. URL: https://www.st.com/en/microcontrollers-microprocessors/stm32f411/documentation.html.

[9] Wikipedia. *Consistent Overhead Byte Stuffing*. URL: https://en.wikipedia.org/wiki/Consistent_Overhead_Byte_Stuffing.

[10] Wikipedia. *Cykliczny kod nadmiarowy*. URL: https://pl.wikipedia.org/wiki/Cykliczny_kod_nadmiarowy.