

**POLITECHNIKA WROCŁAWSKA**  
**WYDZIAŁ ELEKTRONIKI**

---

KIERUNEK: Automatyka i Robotyka (AIR)  
SPECJALNOŚĆ: Technologie Informacyjne  
w Systemach Automatyki (ART)

**PRACA DYPLOMOWA**  
**INŻYNIERSKA**

System lokalizacji samolotów z wykorzystaniem  
ADS-B

Airplane tracking system using ADS-B

AUTOR:  
Karol Szpila

PROWADZĄCY PRACĘ:

dr inż. Krzysztof Halawa  
Katedra Informatyki Technicznej

OCENA PRACY:



# Spis treści

<b>1</b>	<b>Wstęp</b>	<b>2</b>
<b>2</b>	<b>Cele i założenia projektowe</b>	<b>4</b>
2.1	Środowisko sprzętowe i wykorzystane narzędzia . . . . .	4
<b>3</b>	<b>Architektura sprzętowa</b>	<b>5</b>
3.1	Schemat Urządzenia . . . . .	5
3.2	Wykonanie PCB . . . . .	10
<b>4</b>	<b>Architektura oprogramowania</b>	<b>23</b>
4.1	Wzorzec projektowy MVC . . . . .	23
4.2	Model UML . . . . .	24
4.3	Procesy w systemie operacyjnym . . . . .	26
4.4	Wykrywanie i dekodowanie wiadomości ADS-B . . . . .	28
<b>5</b>	<b>Interfejs użytkownika</b>	<b>34</b>
<b>6</b>	<b>Podsumowanie</b>	<b>36</b>
	<b>Bibliografia</b>	<b>36</b>

# Rozdział 1

## Wstęp

ADS-B (ang. Automatic Dependent Surveillance–Broadcast) to system służący do śledzenia statków powietrznych wykorzystywany w kontroli ruchu powietrznego. Powstał, aby uzupełniać pracę PSR (ang. Primary Surveillance Radar) i w przyszłości całkowicie go zastąpić. PSR to radar aktywny bazujący na wysyłaniu fal elektromagnetycznych oraz pomiarze czasu powrotu fali odbitej od przeszkody. Wadą takich systemów jest brak informacji o wykrytym obiekcie poza jego lokalizacją i rozmiarem, wrażliwość na ukształtowanie terenu oraz warunki pogodowe. ADS-B bazuje na lokalizacji przy pomocy satelit GPS. Statki powietrzne w sposób ciągły ustalają swoją pozycję oraz nadają drogą radiową swoje położenie, prędkość oraz identyfikator.

Mode-S to system SSR (ang. Secondary Surveillance Radar) wykorzystywania do odpytywania statku powietrznego o danym 24-bitowym adresie ICAO (ang International Civil Aviation Organization). SSR to wtórny radar dozorowania, uzupełniający pracę PSR o dodatkowe informacje odebrane od statku powietrznego. Zapytania wysyłane są przez ATM (ang. Air Traffic Management) na częstotliwości 1030Mhz, natomiast odpowiedzi na 1090Mhz. ADS-B wykorzystuje Mode-S jako technologię do przesyłania informacji.

Tabela 1.1: *Rodzaje pakietów Mode-S*

Rodzaj wiadomości	Downlink Format	Zawartość
Mode-S (56 bitów)	DF0	Odpowiedz Short Air to Air ACAS
	DF4	Poziom lotu
	DF5	Identyfikator (Roll-call)
	DF11	Odpowiedz All-call
Mode-S (112 bitów)	DF16	Odpowiedz Long Air to Air ACAS
	DF17 (ADS-B)	Pozycja powietrzna pozycja lądowa status ID i rodzaj samolotu prędkość powietrzna
Mode-S EHS (112 bitów)	DF20	Poziom lotu oraz (BDS 4.0/5.0/6.0)
	DF21	ID oraz (BDS 4.0/5.0/6.0)

Powyższa tabela prezentuje podział wiadomości Mode-S ze względu na rozmiar bloku danych oraz DF (ang. Downlink Format), czyli format odebranej ramki. Wyróżniamy wiadomości krótkie o długości 56 bitów i rozszerzone 112 bitowe. DF0 i DF16 wykorzystywane są w ACAS (ang. Airborne Collision Avoidance System). ACAS to system zapobiegania kolizji w kontroli ruchu powietrznego. Wyróżniamy odpowiedź Short Air to Air ACAS (DF0), odbierane przez stacje ATM, oraz Long Air to Air ACAS (DF16), stosowaną do informowania poszczególnych statków o możliwości kolizji. Uczestnicy ruchu wysyłający ramkę DF5 potwierdzają, że są wyposarzone w transponder Mode-S. Ponadto, można wyróżnić odpowiedzi Mode-S EHS (ang. Enhanced Surveillance) DF20 i DF21, które zawierają dodatkowe informacje niedostępne w ADS-B, w zależności od wysłanego przez kontrolę naziemną BDS (ang. Comm-B Data Selector). BDS to numer określający żądana w odpowiedzi informację. Tylko SSR który wysłał zapytanie Mode-S EHS jest w stanie zdekodować otrzymyany pakiet DF20 i DF21, ponieważ nie zawiera ona wysłanego BDS. Poniżej przedstawiono tabelę z parametrami odpowiadającymi poszczególnym BDS.

Tabela 1.2: *BDS dla wiadomości DF20 i DF21*

<b>BDS Register</b>	<b>Basic DAP Set (if Track Angle Rate is available)</b>	<b>Alternative DAP Set (if Track Angle Rate is <b>not</b> available)</b>
<b>BDS 4,0</b>	Selected Altitude	Selected Altitude
<b>BDS 5,0</b>	Roll Angle	Roll Angle
	Track Angle Rate	
	True Track Angle	True Track Angle
	Ground Speed	Ground Speed
<b>BDS 6,0</b>	Magnetic Heading	Magnetic Heading
	Indicated Airspeed (IAS) / Mach no. (Note: IAS and Mach no. are considered as 1 DAP (even if technically they are 2 separate ARINC labels). If the aircraft can provide both, it must do so).	Indicated Airspeed (IAS) / Mach no. (Note: IAS and Mach no. are considered as 1 DAP (even if technically they are 2 separate ARINC labels). If the aircraft can provide both, it must do so).
	Vertical Rate (Barometric rate of climb/descend or baro-inertial)	Vertical Rate (Barometric rate of climb/descend or baro-inertial)
		True Airspeed (provided if Track Angle Rate is <b>not</b> available)

Wiadomość Mode-S można odbierać przy pomocy dowolnego odbiornika dostrojonego na częstotliwości 1090MHz. Tunery z układem RTL2832U można łatwo przekształcić w SDR (ang. Software Defined Radio) SDR to system komunikacji radiowej w którym parametry odbiornika są konfigurowane poprzez program bez ingerencji w sprzęt. Wspomniany odbiornik jest znacznie tańszy od profesjonalnych rozwiązań, co spopularyzowało ADS-B w zastosowaniach amatorskich.

## Rozdział 2

# Cele i założenia projektowe

Celem niniejszej pracy było stworzenie prototypu systemu pozwalającego na lokalizację oraz zbieranie informacji o statkach powietrznych wyposażonych w transpondery ADS-B. W systemie można wyróżnić dwie zasadnicze części. Wewnętrzna odpowiedzialna za wykrywanie, zbieranie i dekodowanie wiadomości oraz wizualną, zajmującą się prezentacją danych oraz interakcją z użytkownikiem. Urządzenie można wykorzystać jako alternatywa dla drogich i profesjonalnych systemów w przypadku zastosowań amatorskich.

### 2.1 Środowisko sprzętowe i wykorzystane narzędzia

W tym podrozdziale zostaną przedstawiono wykorzystanie w systemie urządzenia oraz środowisko programistyczne.

Jako odbiornik radiowy zostanie wykorzystany tuner DVB-T z układem RTL2832U wyposażony w interfejs USB. Urządzenie można łatwo zamienić w SDR. Zdecydowano się na wykorzystanie mikrokontrolera firmy ST z serii STM32F767, ze względu na posiadanie periferii niezbędnych do obsługi urządzeń w systemie, oraz dostępność narzędzi i oprogramowania. W projekcie wykorzystano wyświetlacz LCD-TFT o przekątnej 10.1 cala. Zdecydowano się na wykorzystanie języków programowania C99 oraz C++ 2014. Poniżej wymieniono użyte biblioteki.

- librtlsdr - obsługa Tunera DVB-T jako SDR.
- STM32 USB Host Library - komunikacja mikrokontrolera z radiem poprzez interfejs USB.
- STemWin - obsługa wyświetlacza z panelem dotykowym.
- HAL - konfiguracja i interakcja na niskim poziomie z warstwą sprzętową.

Zdecydowano się na wykorzystanie systemu operacyjnego FreeRTOS, w celu zapewnienia wielozadaniowości oraz mechanizmów do zarządzania zasobami sprzętowymi. Do programowania urządzenia wykorzystano programatora ST-Link z zestawu uruchomieniowego STM32F429I-DISC1. Poniżej zaprezentowano wykorzystane środowisko programowe.

- Atollic True Studio - środowisko do programowania mikrokontrolerów.
- STM32CubeMx - generowania kodu konfiguracyjnego dla układów STM32.
- Circuit Maker - wykonanie projektu PCB.

# Rozdział 3

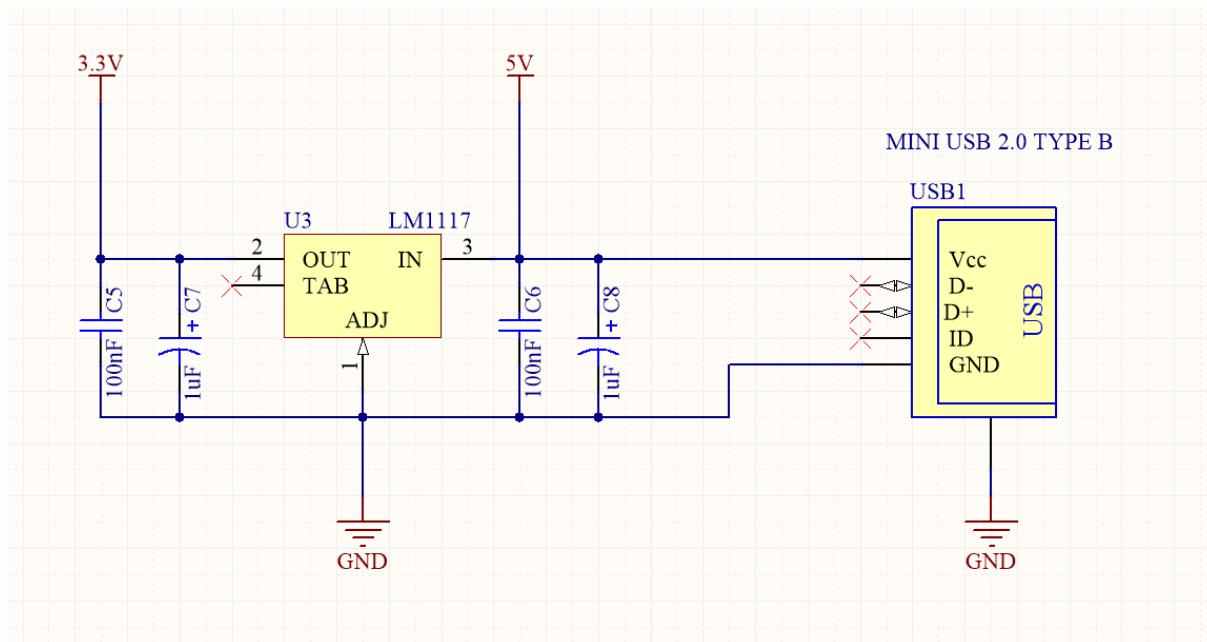
## Architektura sprzętowa

W tym rozdziale zostanie opisana część sprzętowa projektu, czyli schemat urządzenia oraz projekt PCB, wykorzystane elementy oraz zewnętrzne urządzenia.

### 3.1 Schemat Urządzenia

#### Zasilanie

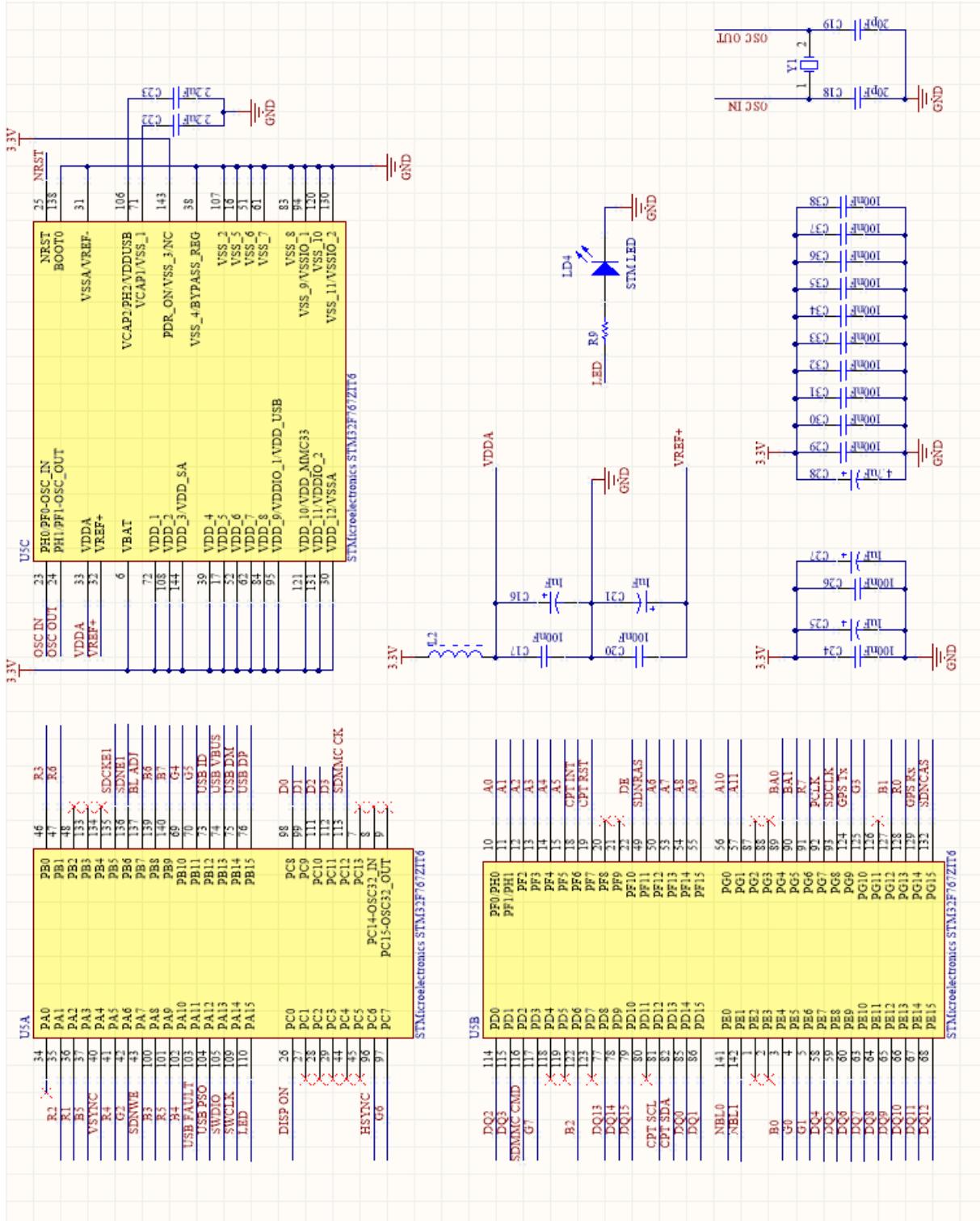
Urządzenie jest zasilane z zewnętrznego źródła 5V poprzez port USB typu mini A. Pozwala to na podłączenie PCB zarówno do sieci przy pomocy ogólnodostępnych przetwornic lub akumulatorów z wyjściem USB. Stabilizator LM1117 wykorzystano do obniżenia napięcia zasilania do wartości 3,3V przy jakiej pracują układy scalone znajdujące się na PCB. Kondensatory C5, C6, C7, C8 zajmują się filtracją zasilania 3,3V i 5V. Poniżej przedstawiono schemat podłączania.



Rysunek 3.1: Schemat części zasilania PCB

## Mikrokontroler

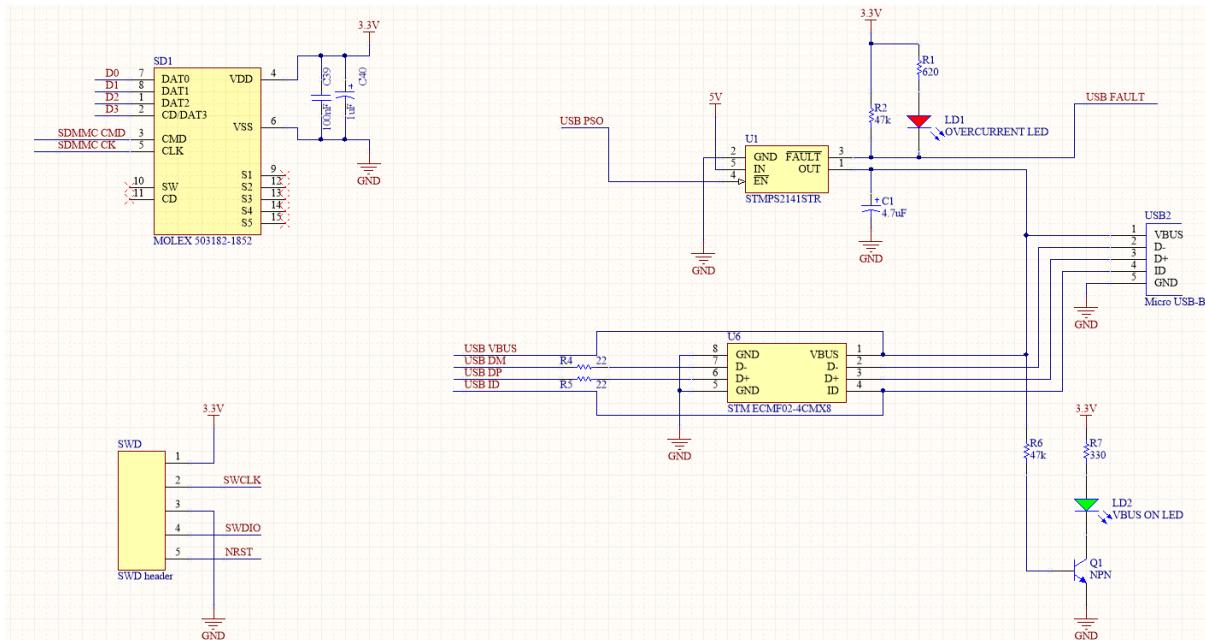
Wykorzystany mikrokontroler to STM32F767ZIT6. Układ został wybrany ze względu na zmieszczenie w najmniejszej obudowie wszystkich wymaganych układów peryferiów takich jak: układ LTDC do sterowania wyświetlaczem LCD, kontroler zewnętrznej pamięci pozwalający obsługiwać SDRAM i interfejsy komunikacyjne USB, I2C oraz UART. Poniżej przedstawiono schemat podłączenia.



Rysunek 3.2: Schemat podłączenia mikrokontrolera STM32F767ZIT6

## Interfejsy komunikacyjne

System posiada gniazdo na kartę Micro SD, która może zostać wykorzystana do przechowywania skompresowanych obrazów do tła GUI lub map. Do programowania mikrokontrolera wykorzystano interfejs SWD zgodny z ST-Link. Do komunikacji z SDR wykorzystano interfejs USB2.0 Full-Speed, który został wyprowadzony poprzez gniazdo Micro USB-B pozwalając w ten sposób zaoszczędzić miejsca na PCB. System jako Host będzie zasilać podłączony układ. Zgodnie ze standardem maksymalny prąd wyjściowy to 500mA. Z tego powodu zastosowano przełącznik STMP2141STR. W przypadku podania stanu wysokiego na pin USB PSO układ zostanie włączony. Jeżeli nie występują żadne sytuacje nieporządne takie jak przetężenie prądowe czy zwarcie, zapali się zielona dioda sygnalizująca poprawne działanie. W przypadku jakichkolwiek problemów prąd zostanie natychmiast odcięty, a układ wystawi wysoki stan na pin FAULT informując o tym mikrokontroler i zapalając czerwoną diodę. W celu zabezpieczania interfejsu przed niepożądanymi wyładowaniami elektrostatycznymi zastosowano układ ochrony ESD (ang. Electro Static Discharge) STMECMF02-4CMX8 dedykowany dla USB2.0, który pełni również funkcję filtra EMI, (ang. Electroagnecite Interferences). Poniżej przedstawiono schemat połączenia układów.

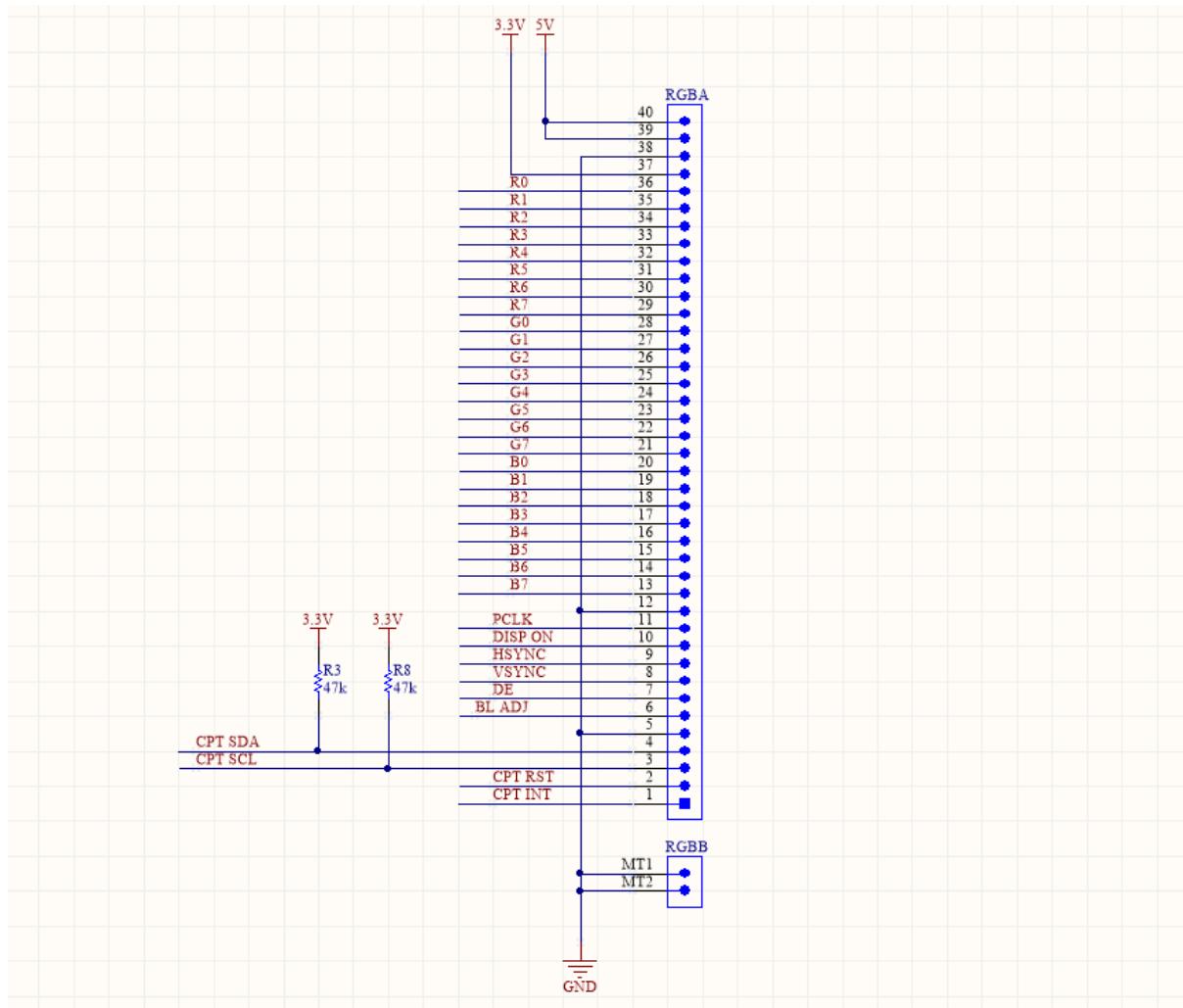


Rysunek 3.3: Schemat połączenia zewnętrznych interfejsów

## Wyświetlacz

Zgodnie z założeniem, system ma posiadać interfejs użytkownika. Do tego celu wybrano wyświetlacz HY101CTP z pojmemnościowym panelem dotykowym. Przekątna wynosi 10,1", a rozdzielcość to 1024 na 600 pikseli. Matryca jest sterowana poprzez MIPI-DPI (ang. Mobile Industry Processor Interface - Display Parell Interface). Panel dotykowy komunikuje się z mikrokontrolerem poprzez interfejs I2C. Podświetlenie jest włączane stane wysokim na pinie DISP ON a jasność zależy od wypełniania sygnału PWM o częstotliwości 10Khz na lini BL ADJ. Rezystory R3 i R8 mają za zadanie wymusić stan wysoki na liniach, ponieważ są sterownie w trybie otwartego drenu. W tej konfiguracji można

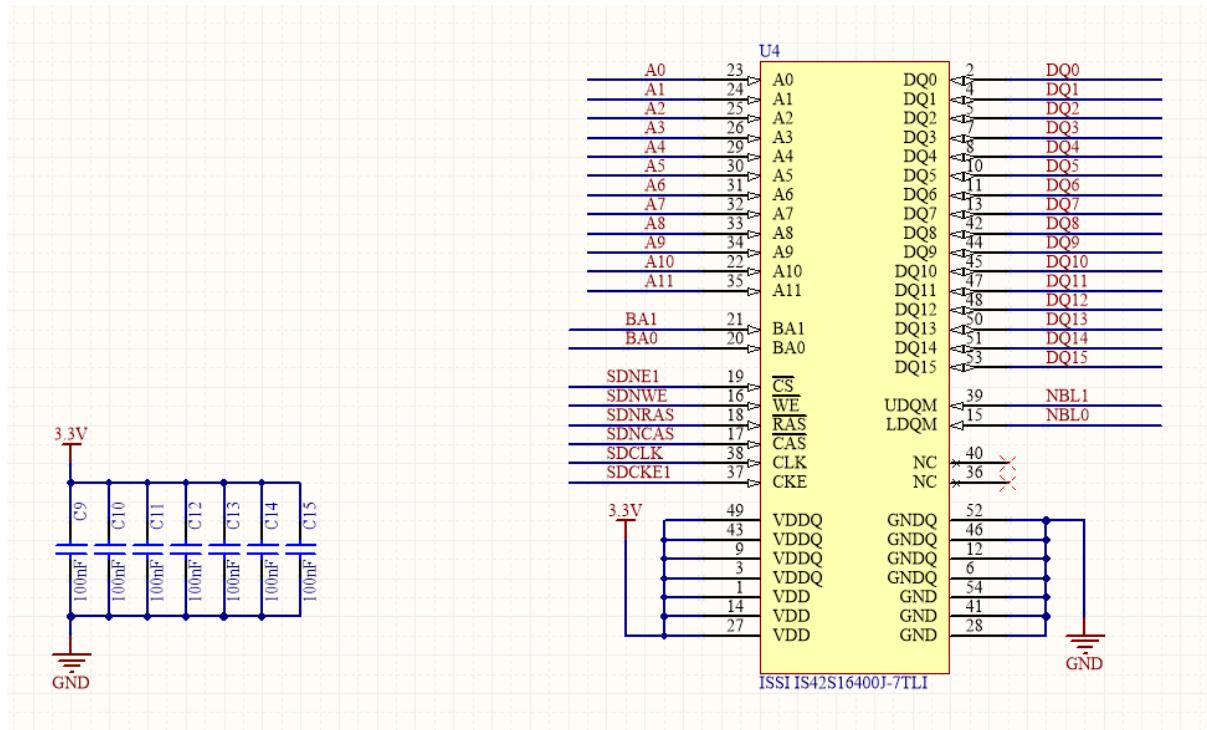
tylko zewrzeć linie do masy zmieniając stan wysoki na niski. Pozwoliło to wyeliminować sytuacje w której linia na jednym końcu jest zwarta do masy a na drugim do zasilania co doprowadziło by do zwarcia i gwałtownego wzrostu mocy mogącego uszkodzić układ.



Rysunek 3.4: Schemat podłączenia wyświetlacza

## SDRAM

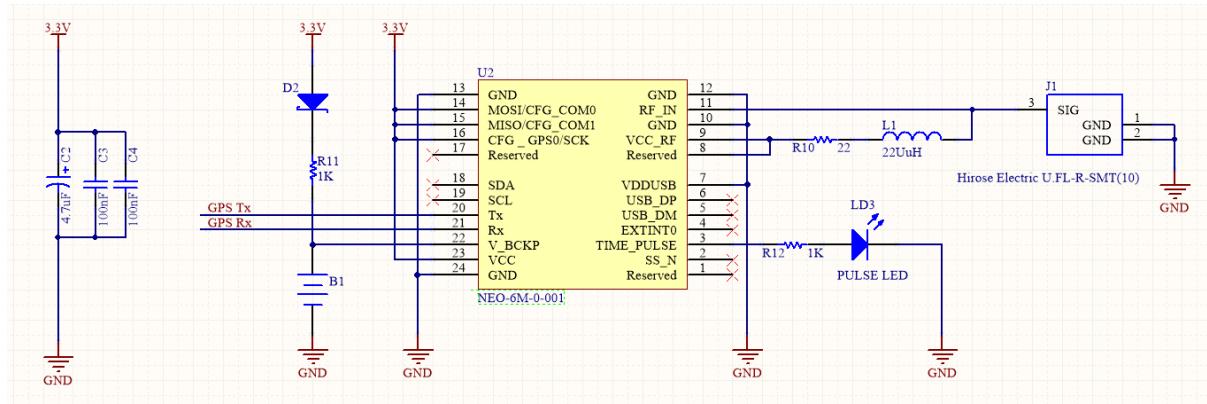
Do działania interfejsu graficznego potrzebna jest pamięć, aby przechowywać ramki wysyłane do wyświetlacza poprzez interfejs MIPI-DPI. Założono iż warstwy będą przechowywane w formacie ARGB8888 (po bajcie na każdy kolor i kanał alfa). Kontroler LTDC jest w stanie sprzętowo łączy dwie warstwy w wynikową która jest wysyłana do wyświetlacza. Zdecydowano, że potrzeba pamięci wystarczająco dużej, by zmieścić trzy bufore. Pierwsza warstwa jest przeznaczony na tło, które jest niezmienne podczas działania urządzenia. Dwie pozostałe warstwy odpowiadają do naprzemiennej prezentacji zmiennych danych takich jak położenie statku powietrznego. Będzie to implementacja mechanizmu podwójnego buforowania pozwalająca wyeliminować migotanie matrycy podczas modyfikacji bufora aktualnie wyświetlonego. Trzy warstwy o rozmiarze 1024 na 600 pikseli w formacie ARGB8888 zajmą 57600 Kb. W pracy zdecydowano się wykorzystać układ IS42S16400J-7TLI posiadający 65536 Kb pamięci. Poniżej przedstawiono schemat połączenia z mikrokontrolerem.



Rysunek 3.5: Schemat podłączenia układu SDRAM

## Moduł GPS

Aby system był w stanie poprawnie obliczyć odległość od namierzonego statku powietrznego i poprawnie zaznaczyć jego pozycję na radarze, potrzebna znać pozycję urządzenia. Do tego zadania wybrano układ NEO-6M-0-001 z zewnętrzną aktywną anteną. Do układu została podłączona zewnętrzna bateria podtrzymująca zasilanie. Dzięki temu urządzenie może uruchomić poprzez ciepły start, co pozwala zaoszczędzić czas potrzebny na znalezienie odpowiedniej liczby satelitów. Do komunikacji z mikrokontrolerem wykorzystano interfejs UART (ang. Universal Asynchronous Receiver Transmitter).



Rysunek 3.6: Schemat podłączenia modułu GPS

## 3.2 Wykonanie PCB

Poniżej szczegółowo opisano projekt oraz wykonanie PCB wraz z obliczeniami. Opisano decyzję projektowe wynikające z ograniczeń technologii oraz występowania niepożądanych zjawisk fizycznych.

### Parametry PCB

W projekcie PCB zdecydowano się na wykonanie technologią czterowarstwową. Pozwoliło to zmniejszyć rozmiary urządzenia oraz zapewnić dobre ekranowanie pomiędzy warstwami sygnałowymi. Ograniczyło to przesłuchy pomiędzy warstwami. Ponadto ciągłość warstw referencyjnych (masy i zasilania) pozwalały na prądy powrotne przepływały możliwe najkrótszą drogą o najmniejszej impedancji zmniejszając emisje EMC. Poniżej przedstawiono tabele z układem warstw PCB.

Tabela 3.1: *Układ warst PCB*

Warstwa	grubość [mm]	grubość [mil]
sygnały	0.03556	1,4
dielektryk	0,17018	6,7
masa	0,01778	0,7
rdzeń	1,1938	47
zasilanie	0,01778	0,7
dielektryk	0,17018	6,7
sygnały	0.03556	1,4

Warstwy przewodzące wykonane są z miedzi natomiast jako dielektryk wykorzystano materiał FR-408 o stałej dielektrycznej  $\epsilon_r = 3.66$ .

### GPS

Zgodnie z notą katalogową układu NEO-6M-0-001, ścieżka antenowa powinna mieć impedancję dopasowaną do  $Z_0 = 50\Omega$ , którą obliczono z następującego wzoru :

$$Z_0 = \frac{87}{\sqrt{\epsilon_r + 1.41}} \ln \left( \frac{5.98H}{0.8W + T} \right) [2] \quad (3.1)$$

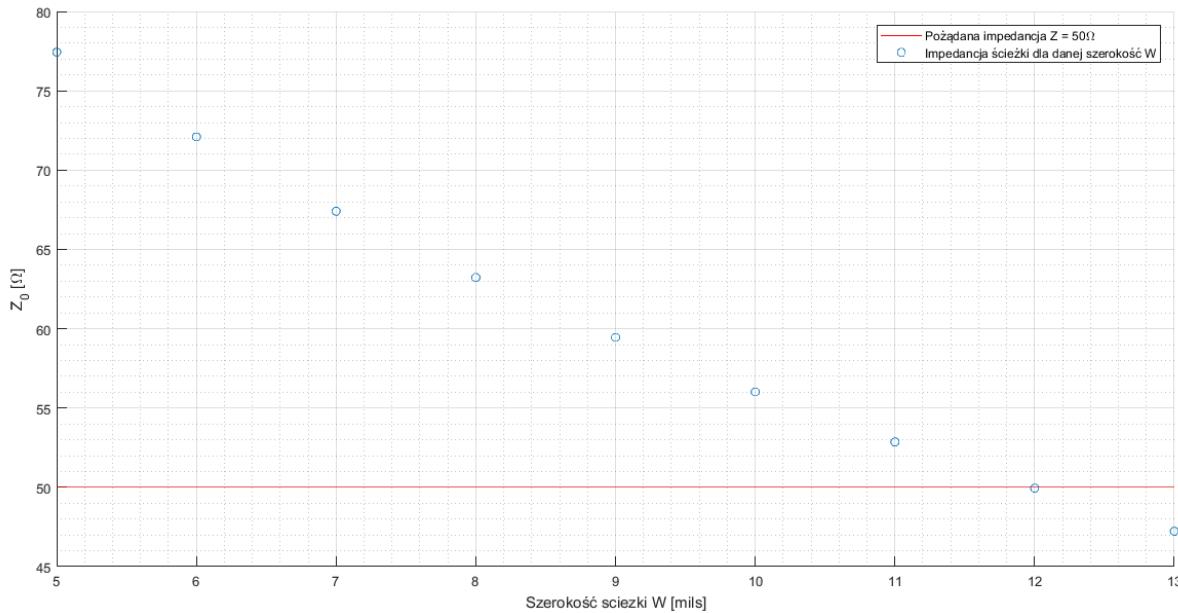
dla

$$0,1 < W/H < 2,0$$

$$1 < \epsilon_r < 15$$

gdzie

- $\epsilon_r$  - stała dielektryczna
- H - grubość dielektryka
- W - szerokość ścieżki
- T - grubość ścieżki



Rysunek 3.7: Impedancja ścieżki w zależności od szerokości ścieżki

Powyższy wykres przedstawia impedancję ścieżki antenowej modułu GPS w zależności od szerokości, obliczoną ze wzoru 3.1. Z tabeli 3.1 odczytano  $T = 1,4\text{mils}$ ,  $H = 6,7\text{mils}$  oraz  $\varepsilon_r = 3.66$ . Czerwoną linią zaznaczono pożądane dopasowanie  $Z_0 = 50\Omega$ . Założono że W zostanie obliczone z dokładnością do 1 mils. Obliczenia wykonano zaczynając od najmniejszej szerokości gwarantowanej przez producenta  $W = 5\text{mils}$ , aż do otrzymania pierwszego punktu pod prostą z optymalnym dopasowaniem. Najlepsze dopasowanie otrzymano dla  $W = 12\text{mils}$ , gdzie  $Z_0 \approx 49.95$ . Dla takiego  $Z_0$  obliczono współczynnik odbicia, który reprezentuje jaką część sygnału została stracona w wyniku niedopasowania impedancji przy przejściu pomiędzy liniami transmisyjnymi.

$$\Gamma = \frac{Z_l - Z_s}{Z_l + Z_s} [2] \quad (3.2)$$

gdzie

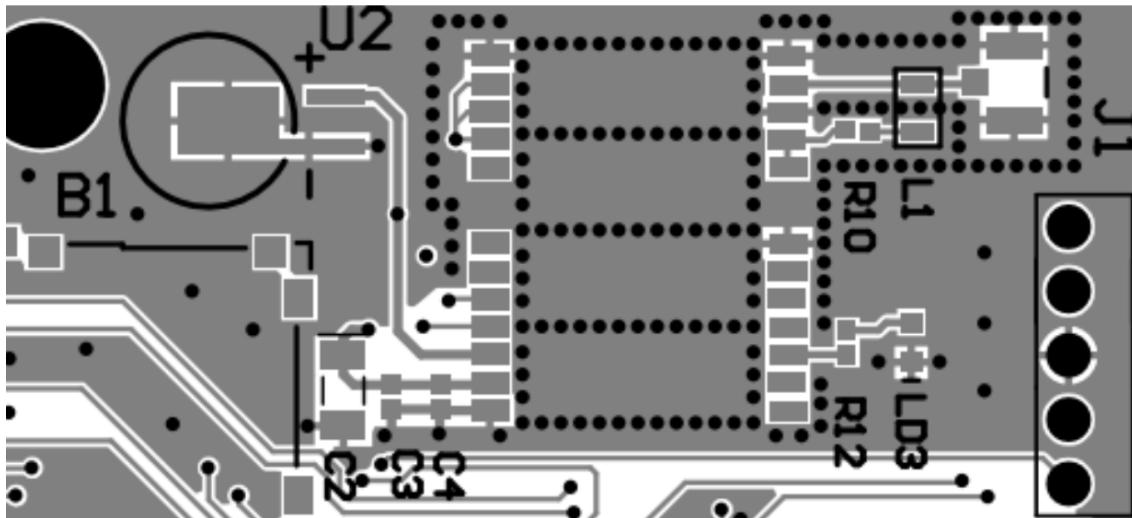
- $Z_l$  - impedancja obciążenia, w tym przypadku impedancja lini transmisyjnej  $Z_0 = 49,95\Omega$
- $Z_s$  - impedancja źródła, czyli wejścia antenowego modułu GPS, zgodnie z notą katalogową  $Z_s = 50\Omega$

Korzystając ze wzoru 3.2 obliczono:

$$\Gamma = \frac{Z_0 - Z_s}{Z_0 + Z_s} = \frac{49,59\Omega - 50\Omega}{49,59\Omega + 50\Omega} \approx 0.0005 \quad (3.3)$$

W przeliczeniu na procenty daje to  $\Gamma \cdot 100\% = 0.0005 \cdot 100\% = 0.05\%$  sygnału odbitego przez linię transmisyjną. Jest to wartość akceptowalna zatem pozostało prze szerokości  $W = 12\text{mils}$ .

Zgodnie z zaleceniami noty katalogowej wykonano serię przelotek pod i do okola układu GPS (U2). Ten sam zabieg zastosowano również dla ścieżki antenowej (pomiędzy J1 i U2). Zapewniło to lepsze odprowadzanie ciepła przez moduł, a zatem niższą temperaturę pracy i mniejszą emisję EMC. Poniżej zaprezentowano zdjecie z projektem.



Rysunek 3.8: Projekt PCB dla układu GPS

## USB

W projekcie wykorzystano wbudowany w STM32F767xx kontroler interfejsu USB2.0 Full-Speed o maksymalnej przepustowości 12Mb/s. Zgodnie ze standardem, należy dopasować impedancję różnicową pomiędzy liniami transmisyjnymi sygnału nieodwróconego i odwróconego do  $90 \Omega$ . Poniżej przedstawiono zastosowany wzór oraz obliczenia.

$$Z_0 = \frac{174}{\sqrt{\varepsilon_r + 1.41}} \ln \left( \frac{5.98H}{0.8W + T} \right) \left( 1 - 0.48e^{(-0.96\frac{D}{H})} \right) [2] \quad (3.4)$$

dla

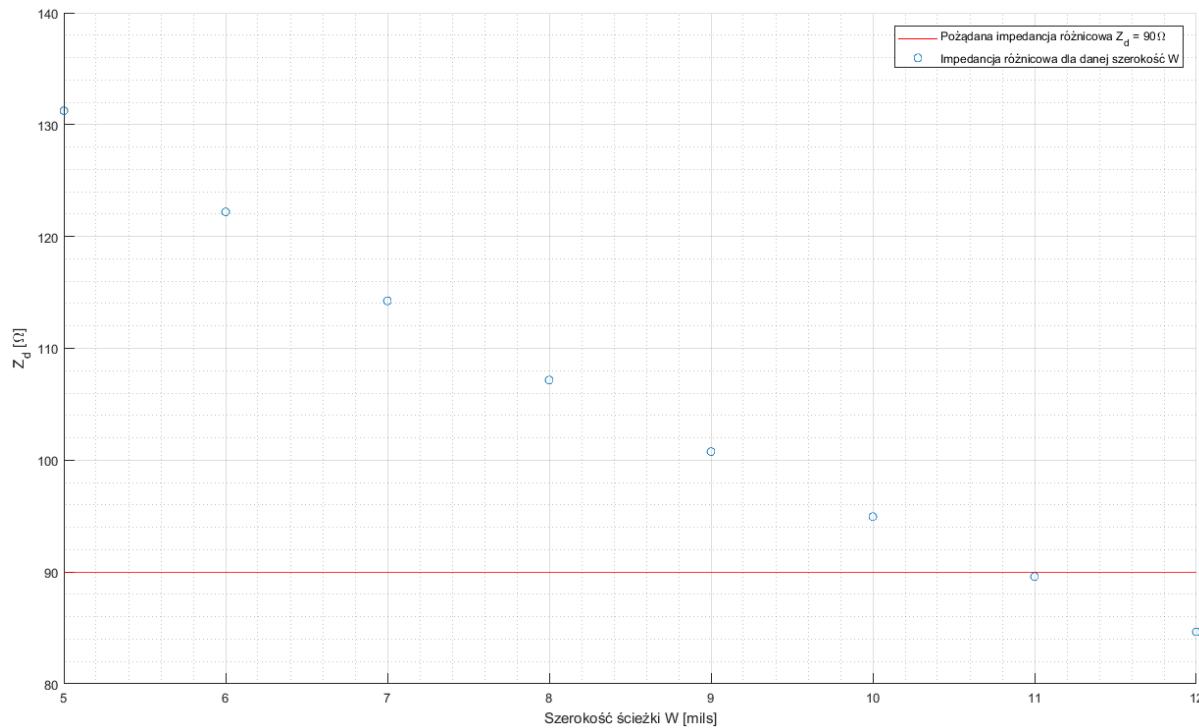
$$0,1 < W/H < 2,0$$

$$1 < \varepsilon_r < 15$$

gdzie

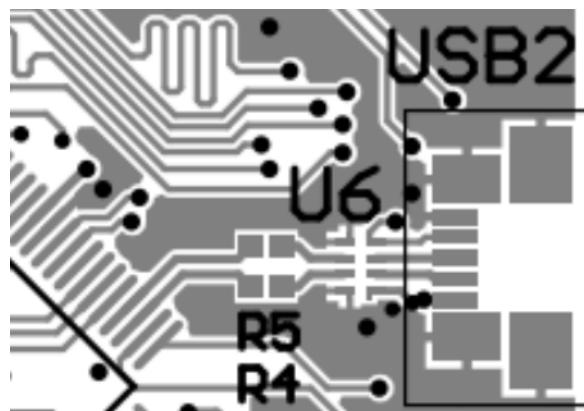
- $\varepsilon_r$  - stała dielektryczna
- H - grubość dielektryka
- W - szerokość ścieżki
- T - grubość ścieżki
- D - odległość pomiędzy ścieżkami

Poniższy wykres przedstawia impedancję różnicową lini transmisyjnych interfejsu USB w zależności od szerokości obliczoną ze wzoru 3.4 z parametrami  $T = 1,4mils$ ,  $H = 6,7mils$  i  $\varepsilon_r = 3,66$  z tabeli 3.1. Założono odległość pomiędzy ścieżkami  $D = 8mils$  równą rozstawowi pinów w mikrokontrolerze. Czerwoną linią zaznaczono pożąданie dopasowanie  $Z_d = 90\Omega$ . Założono że W zostanie obliczone z dokładnością do 1 mils. Obliczenia wykonano zaczynając od najmniejszej szerokości gwarantowanej przez producenta  $W = 5mils$ , aż do punktu pod prostą z optymalnym dopasowaniem. Najlepsze dopasowanie otrzymano dla  $W = 11mils$ , gdzie  $Z_d \approx 89.56$ . Zgodnie ze specyfikacją interfejsu USB2.0 Full-speed impedancja różnicowa musi wynosić  $Z_d = 90 \pm 15\%.[4]$  Zatem  $76,5\Omega \leq Z_d \leq 103,5\Omega$ .  $Z_d$  obliczone dla  $W = 11mils$  spełnia to wymaganie.



Rysunek 3.9: Impedancja różnicowa w zależności od szerokości ścieżki

Wszystkie elementy zostały umieszczone na górnej warstwie sygnałowej, aby nie stosować przelotek które wprowadzając niepożądane pojemności i indukcyjności linii transmisyjnym. Ścieżki są zginane po kątem nie większym niż  $45^\circ$ . Warstwą referencyjną dla sygnałów jest warstwa masy co zapewnia lepsze ekranowanie dla sygnałów szybkich. Linie różnicowe są oddzielone polem masy od innych sygnałów o co najmniej 50mils. Zastosowanie powyższych reguł pozwoliło ograniczyć zjawiska utrudniające dopasowania impedancji. Wspominanie w rozdziale schematu rezystory R5 i R4 służą jako szeregową terminację sygnału. Dzięki temu zabiegowi czasy narastania zboczy się większe co zmniejsza generowanie zakłócenia EMC. Układ U6 działa jako zabezpieczenie przeciwko wyładowaniom elektrostatycznym mogącym uszkodzić urządzenie oraz jako dodatkowy filtr EMI. Rezystory R5 i R4 pełnią rolę szeregowej terminacji sygnału. Zgodnie z notą katalogową katalogową układu STM32F767xx ich rezystancja wynosi  $22\ \Omega$ . Poniżej przedstawiono część projektu PCB z interfejsem USB.



Rysunek 3.10: Projekt PCB dla interfejsu USB

## Interfejsy szybkie

Częstotliwość sygnału zegarowego dla SDRAM wynosi  $108MHz$ . W przypadku komunikacji z wyświetlaczem szybkość interfejsu zależy od oczekiwanej liczby klatek na sekundę. Częstotliwość sygnału zegarowego dla interfejsu MIPI-DPI wyraża się wzorem:

$$CLK = W \cdot H \cdot fps \quad (3.5)$$

gdzie:

- W - szerokość matrycy w pikselach
- H - wysokość matrycy w pikselach
- fps - częstotliwość odświeżania ekranu w Hz

Dla użytego w projekcie wyświetlacza przy założeniu odświeżania matrycy  $60Hz$  korzystając z równania 3.5 otrzymujemy:

$$CLK = 1024 \cdot 600 \cdot 60Hz = 614400 \cdot 60Hz = 36864000Hz \approx 37MHz$$

Interfejsy o takich częstotliwościach zostały uznane za szybkie, co za tym idzie podjęto dodatkowe działania podczas projektowania ich linii transmisyjnych. Zadbano by warstwy referencyjne pod ścieżkami sygnałowymi były ciągłe, aby zapewnić ekranowanie i ograniczyć przesłuchy od linii po przeciwnej stronie płytka. Ponadto dopasowano długości wszystkich ścieżek w interfejsie, by sygnały przychodziły w tym możliwie podobnym czasie. Wzorując się na projekcie referencyjnym płytka ewaluacyjnej STM32F746G-DISCO zadbano, by różnica długości pomiędzy ścieżkami dla obu interfejsów nie była większa niż  $100mils$ . Opóźnienie linii transmisyjnej, czyli czas jaki sygnał potrzebuje na pokonanie określonej drogi wyraża się wzorem:

$$t = \frac{l\sqrt{\varepsilon_r}}{c} \quad (3.6)$$

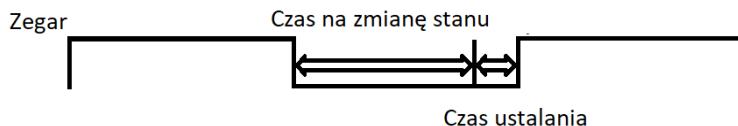
gdzie

- l - długość linii transmisyjnej
- $\varepsilon_r$  - stała dielektryczna
- c - szybkość rozchodzenia się fali w próżni

Korzystając ze wzoru 3.6 dla parametrów  $l = 0.00254m$  ( $100mils$ ) i  $c = 299\ 792\ 458\frac{m}{s}$  oraz  $\varepsilon_r = 3,66$  obliczono opóźnienie sygnału czyli czas jakim dzieli sygnały w liniach o długości różniącej się o  $100mils$ .

$$t = \frac{0.00254m\sqrt{3,66}}{299792458\frac{m}{s}} = \frac{0.00485931m}{299792458\frac{m}{s}} = 1.621 \cdot 10^{-11}s = 16,21ps$$

Wyświetlacz jest sterowny bezpośrednio poprzez interfejs LVDS (ang. Low Voltage Differential Signaling), jednak posiada również wbudowany konwerter THC63LVDM83D pozwalający na komunikację ze pomocą MIPI-DPI. Oznacza to, że wszelkie ograniczenia czasowe powinny być rozpatrywane względem wspomnianego wcześniej układu. Wyjścia mikrokontrolera zmieniają się przy zboczu opadającym sygnału zegarowego, natomiast



Rysunek 3.11: *Diagram ograniczeń czasowych dla interfejsów szybkich*

są zatrzaskiwane przez konwerter przy zboczu narastającym. Wszystkie sygnały powinny dojść do wyświetlacza od momentu wystąpieniem zbocza opadającego do narastającego z uwzględnieniem czasu na ustalenie się sygnału (ang. Setup Time). Jest to najpóźniejszy moment w którym muszą ustalić się stany na wszystkich liniach przed przyjściem sygnału taktującego. Poniżej przedstawiono diagram ilustrujący opisane ograniczenia czasowe.

Zatem maksymalna rozbieżność czasowa pomiędzy sygnałami wyraża się wzorem.

$$t_{max} = t_{low} - t_{setup} \quad (3.7)$$

gdzie

- $t_{low}$  - czas trwania stanu niskiego dla sygnału zegarowego
- $t_{setup}$  - czas na ustalenie się sygnału

Z noty katalogowej mikrokontrolera odczytano że stan niski dla sygnału zegarowego LTDC wynosi minimalnie 45% okresu sygnału zegarowego obliczonego z równania 3.5 zatem:

$$t_{low} = \frac{1}{36864000MHz} \cdot 45\% = 27127ps \cdot 45\% = 12207ps$$

W nocie THC63LVDM83D znaleziono  $t_{setup} = 2000ps$ . Korzystając z 3.7 obliczono

$$t_{max} = 4167ps - 2000ps = 2167ps$$

Ograniczenia czasowe nie zostały przekroczone, ponieważ  $t_{max} = 12207ps \geq t = 16, 21$ .

W przypadku FMC wykorzystywanego do obsługi SDRAM, komunikacja odbywa się w obie strony. Ponownie czas trwania stanu niskiego sygnału zegarowego wynosi 45% okresu. Zatem:

$$t_{low} = \frac{1}{108MHz} \cdot 45\% = 9259ps \cdot 45\% = 4167ps$$

Poniżej przedstawiono tabelę z czasami ustalania dla wszystkich sygnałów interfejsu odczytyanych z noty układu IS42S16400J-7TLI.

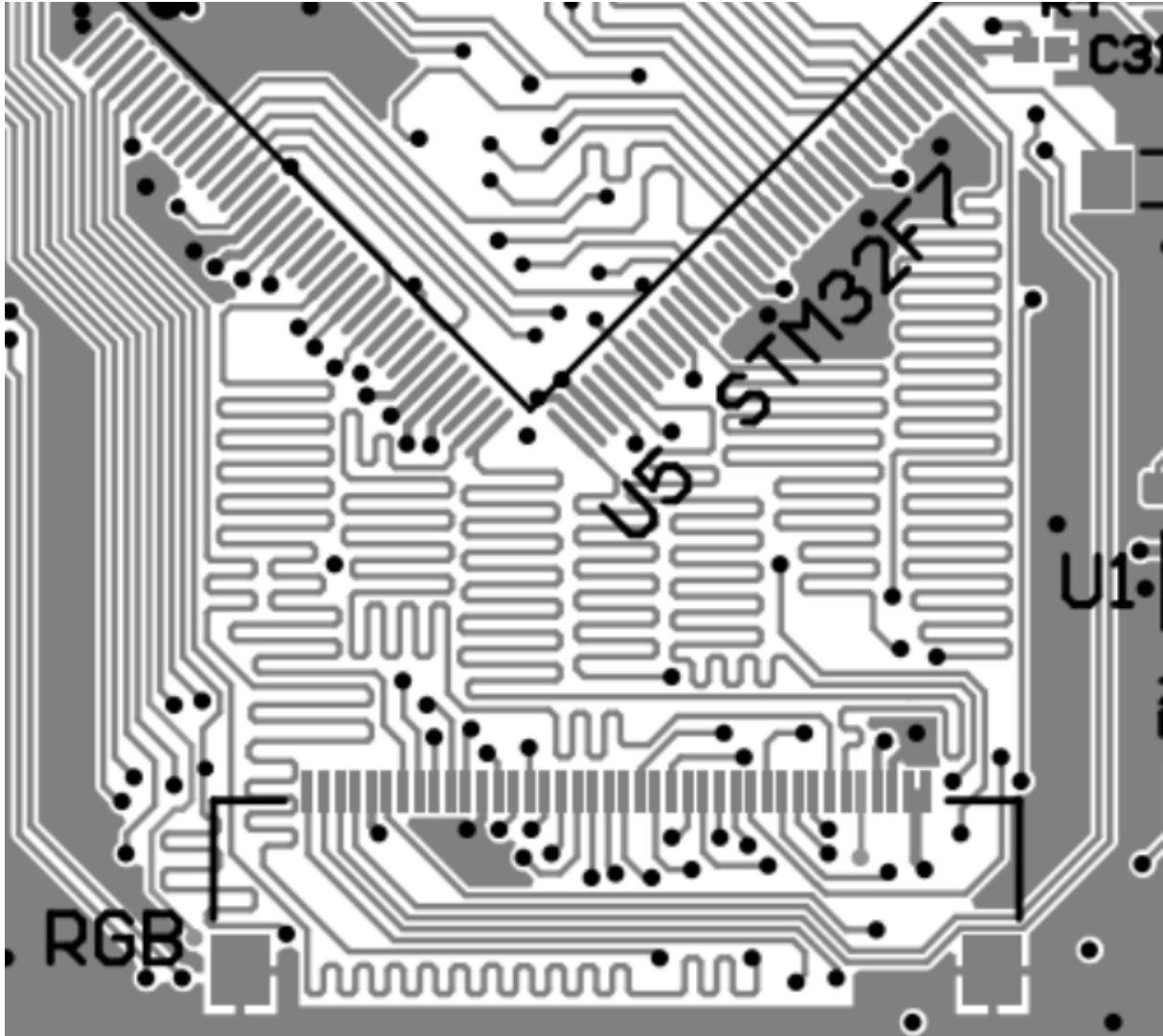
Tabela 3.2: *Czasy ustalania dla wszystkich sygnałów układu SDRAM*

Parametr	czas [ps]
Input Data Setup Time	1500
Address Setup Time	1500
CKE Setup Time	1500
Command Setup Time	1500

Oznacza to, że można przyjąć  $t_{setup} = 1500ps$ . Korzystając z równania 3.7 obliczono:

$$t_{max} = 4167ps - 1500ps = 2667ps$$

Ponownie  $t_{max} = 2667ps \geq t = 16, 21$ , zatem ograniczenia czasowe nie zostały przekroczone. Poniżej przedstawiono zbliżenie na projekt PCB, gdzie można zauważyc meandry, których celem jest wyrównanie różnicy długości pomiędzy ścieżkami sygnałowymi.



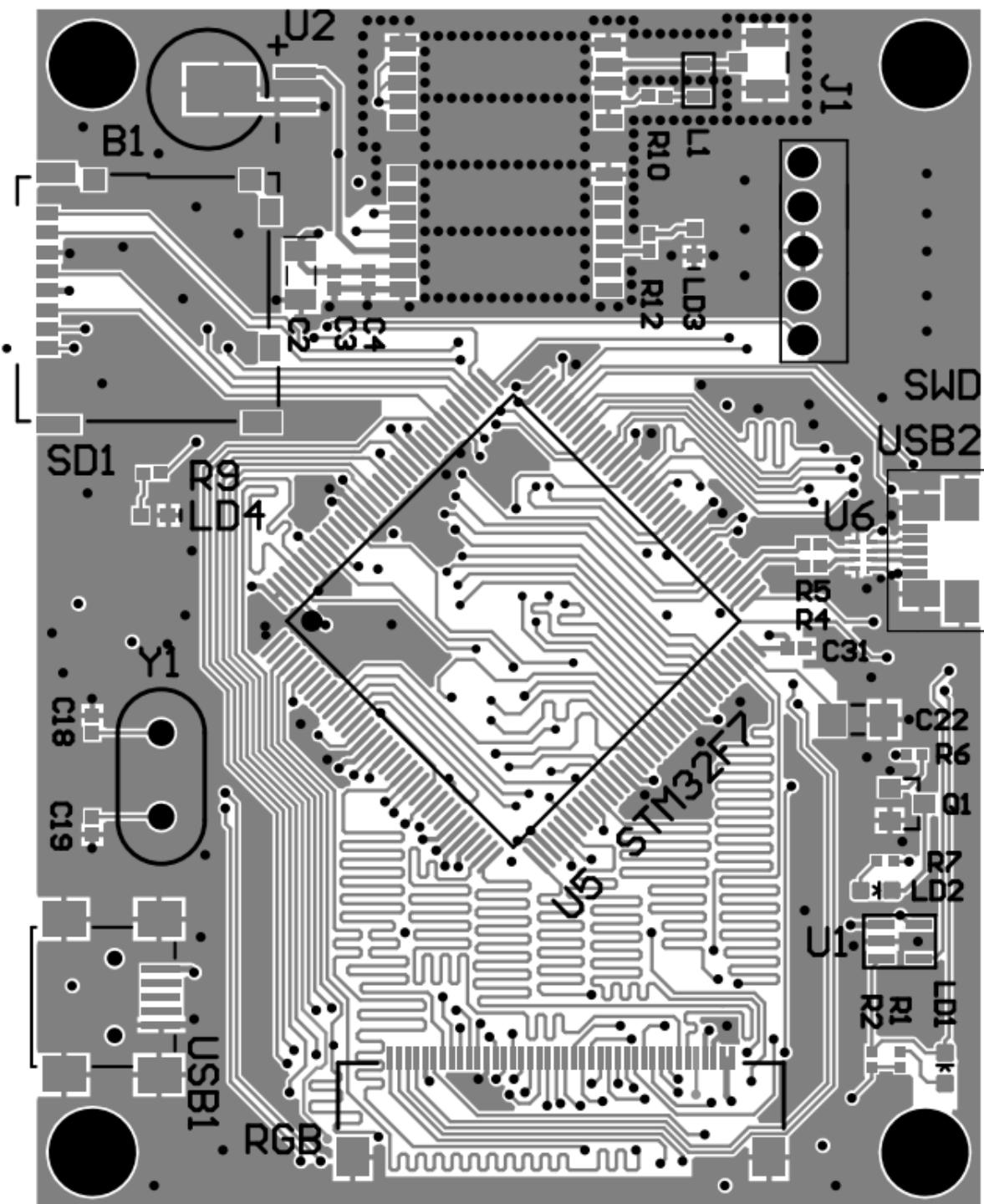
Rysunek 3.12: Przykład dopasowania długości ścieżek na PCB

## Warstwa zasilania

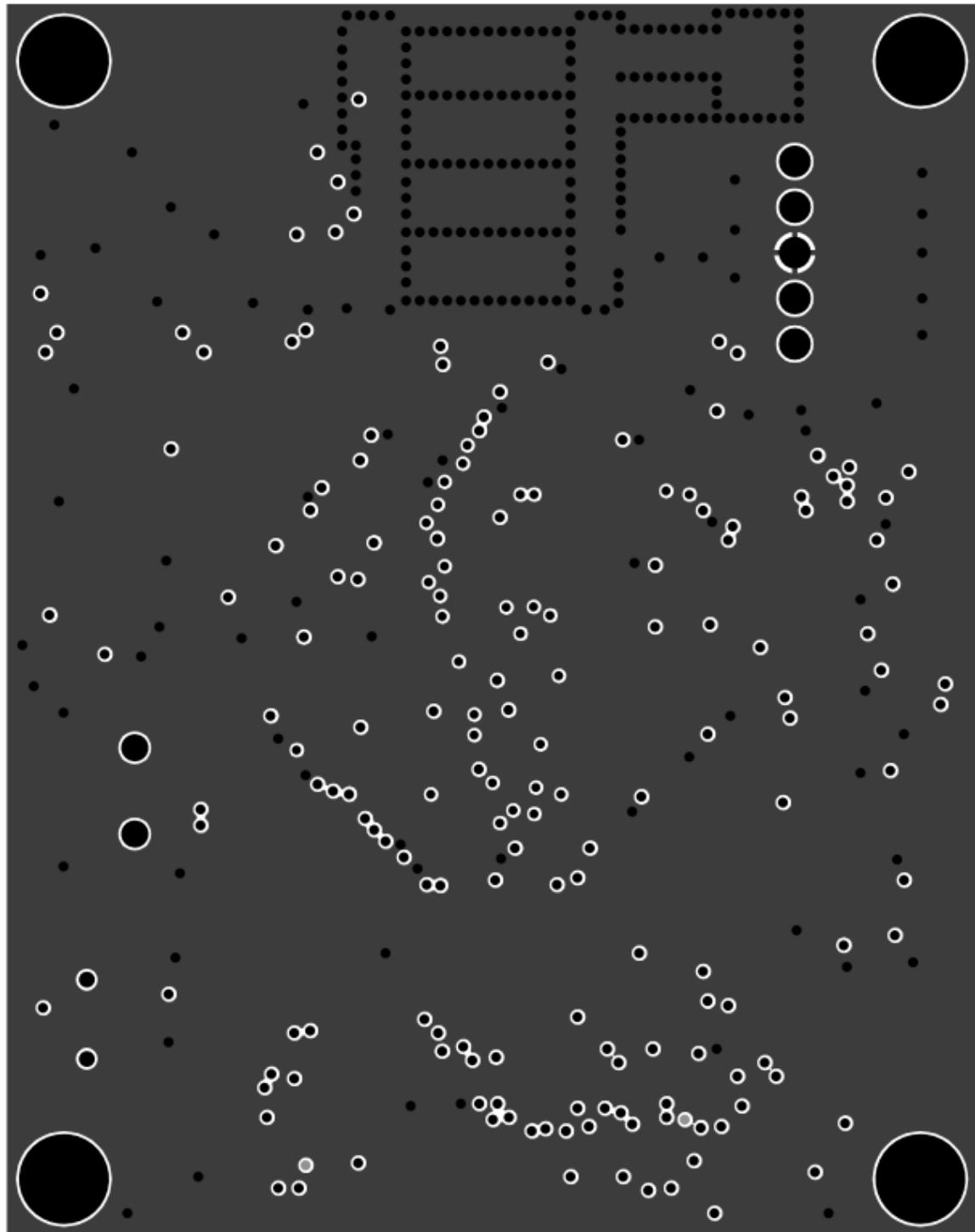
Większość układów na PCB jest zasilana napięciem 3,3V. Jednak zasilanie pokudłaczego urządzenia USB i podświetlenie wyświetlacza potrzebuje napięcia 5V. Wymagało to odpowiedniego podzielania warstwy trzeciej. Zadbano o to żeby żadne ścieżki sygnałów wrażliwych na zakłócenia nie przechodziły nad przerwą pomiędzy polem zasilanie 3,3V i 5V. Nieciągłość warstwy referencyjnej powoduje powstawanie pętli prądowych, ponieważ prąd powrotny szuka innej dostępnej ścieżki powrotu. Zjawisko to może powodować wzrost zakłóceń elektromagnetycznych emitowanych przez urządzenie. Warstwę przedstawia obraz 3.14 w podrozdziale 3.2.

## Prezentacja PCB

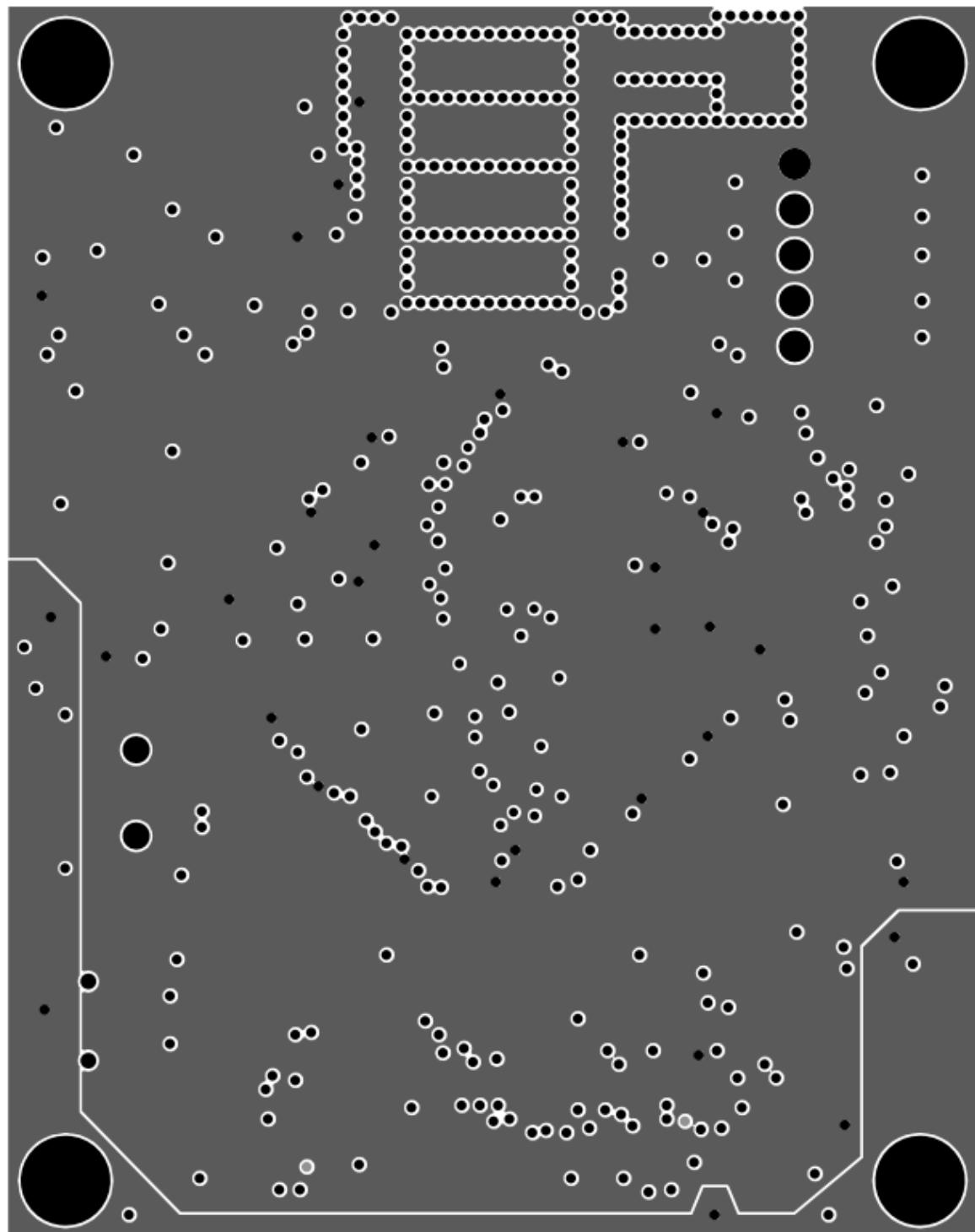
Poniżej przedstawiono obrazy wszystkich warstw z programu wykorzystanego do projektu oraz zdjęcie gotowego PCB.



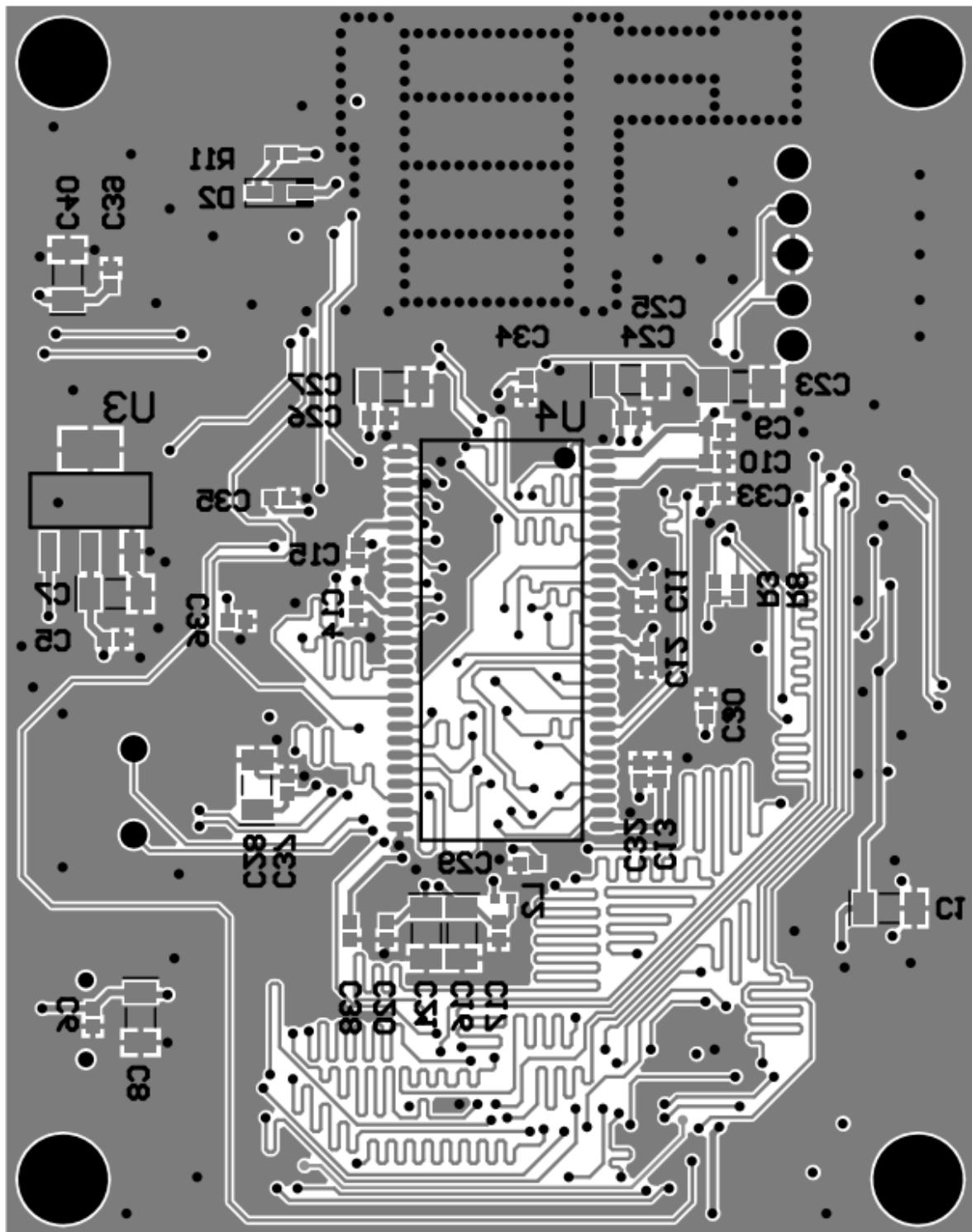
Rysunek 3.13: Układ warstwy 1.



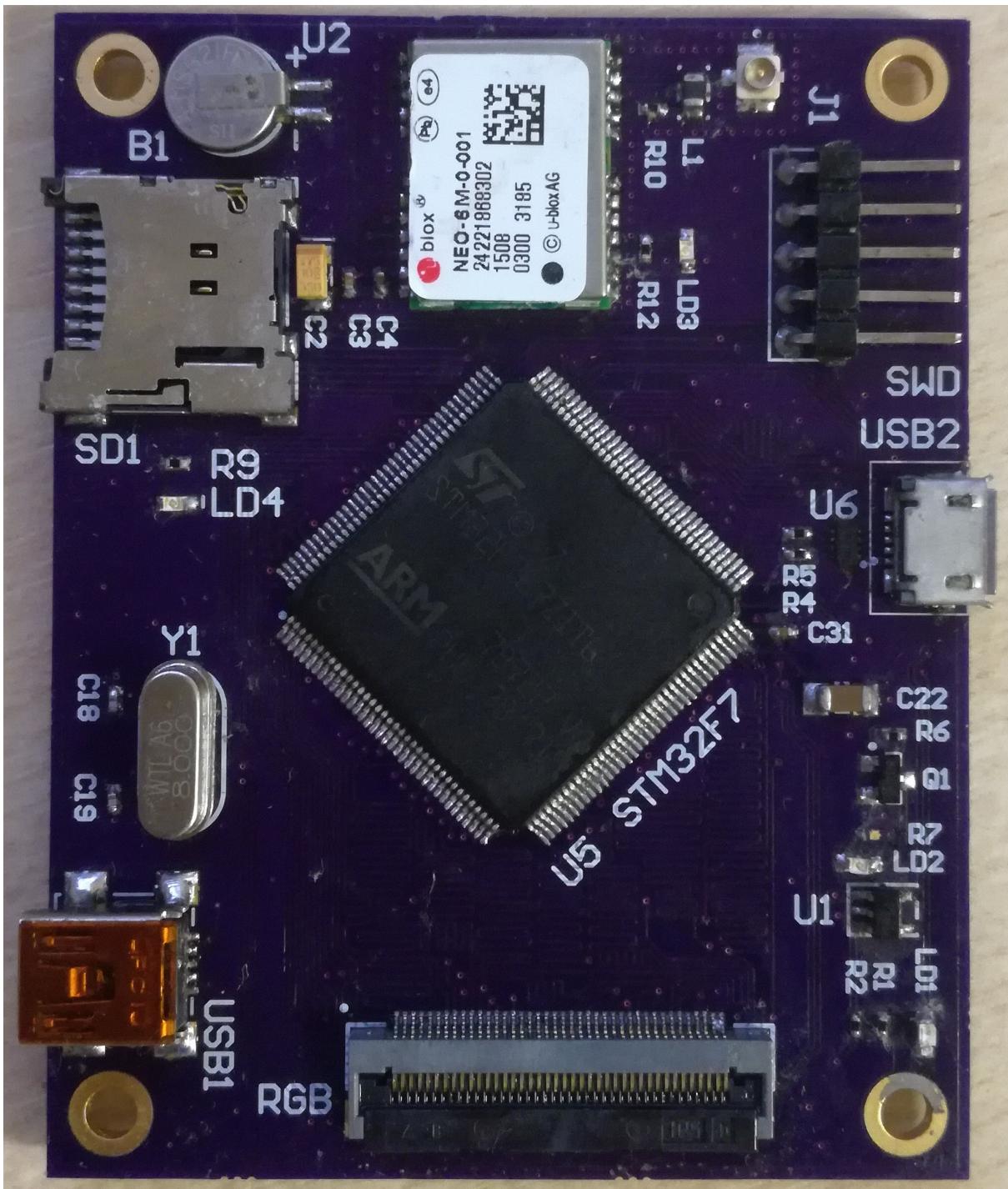
Rysunek 3.14: *Układ warstwy 2.*



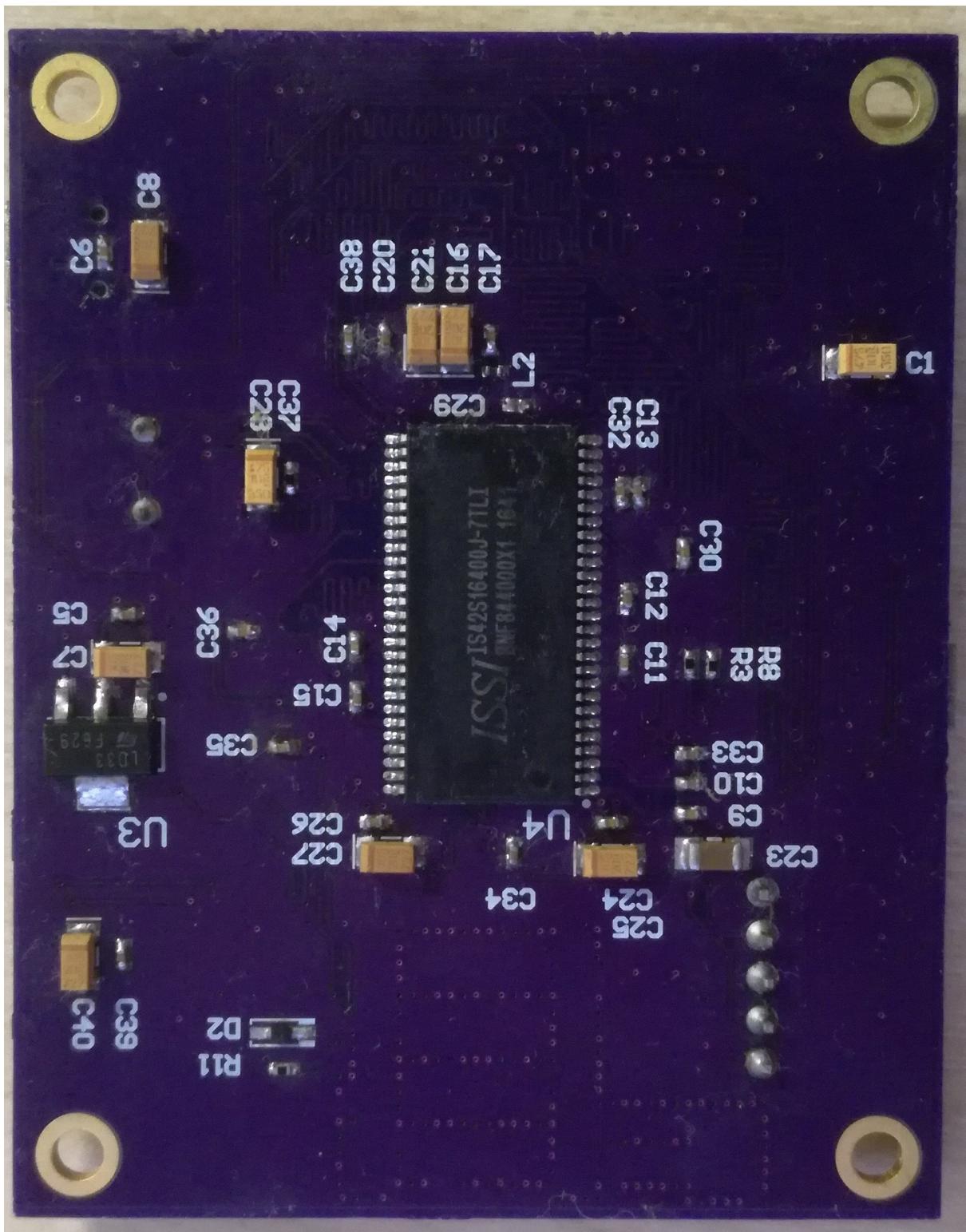
Rysunek 3.15: Układ warstwy 3.



Rysunek 3.16: Układ warstwy 4.



Rysunek 3.17: Zdjęcie wierzchu PCB.



Rysunek 3.18: Zdjęcie spodu PCB.

# Rozdział 4

## Architektura oprogramowania

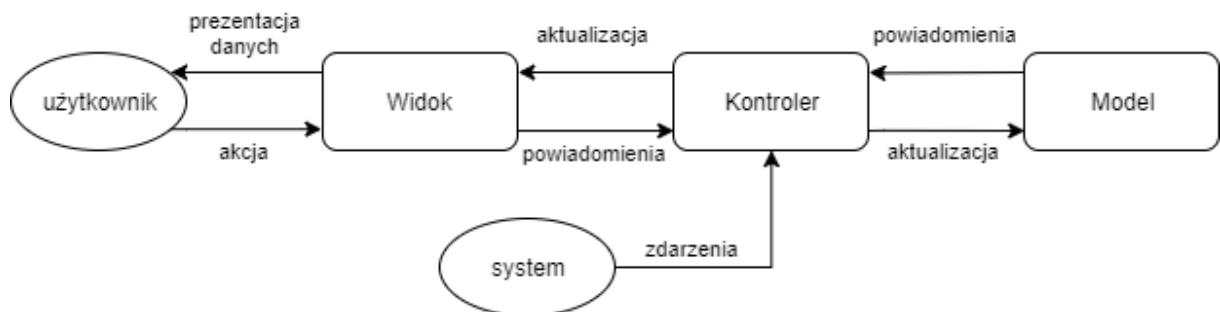
W tym rozdziale opisano strukturę programu wraz z modelem UML oraz opisem funkcjonalności poszczególnych klas. Przedstawiono wykorzystane wzorce projektowe i algorytmy.

### 4.1 Wzorzec projektowy MVC

Model-Widok-Kontroler, MVC (ang. Model-View-Controller) to wzorzec projektowy wykorzystywany przy tworzeniu aplikacji z interfejsem użytkownika. W MVC można wy różnić trzy zasadnicze części:

- Model - reprezentacja logiki biznesowej (struktura danych oraz wykonywane na nich operacje).
- Widok - warstwa prezentacji danych zawartych w modelu.
- Kontroler - wykonuje operacje na modelu. Zapewnia komunikację pomiędzy widokiem i modelem. Obsługuje żądania użytkownika i systemy dotyczące modelu.

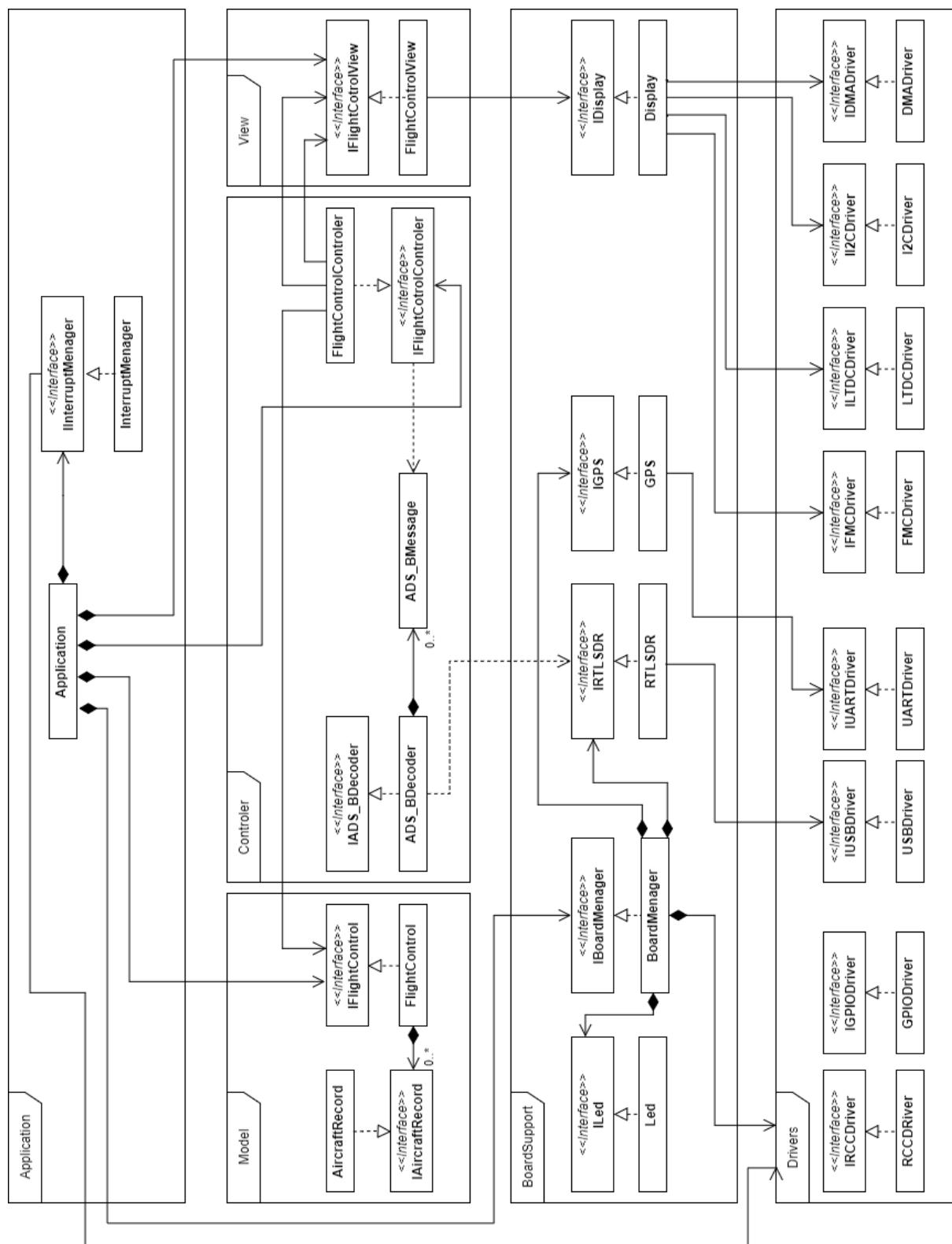
Wzorzec MVC zapewnia całkowitą niezależność modelu od widoku, co ułatwia implementację obu części. Ponadto zapewnia przejrzystość projektu grupując klasy we wcześniej wymienione warstwy abstrakcji. Pozwala to ograniczyć liczbę powiązań pomiędzy komponentami. Poniżej przedstawiono diagram interakcji dla zaimplementowanego wzorca MVC.



Rysunek 4.1: *Diagram interakcji pomiędzy komponentami MVC*

## 4.2 Model UML

Poniżej przedstawiono diagram UML reprezentując powiązania i interakcje pomiędzy poszczególnymi klasami.



Rysunek 4.2: Model klas UML

Poniżej opisano funkcjonalności klas we wszystkich przestrzeniach nazw.

**Application** - przestrzeń zawierająca klasy najwyższego poziomu odpowiedzialne na działanie aplikacji.

**FlightRadarApp** - główna klasa odpowiadająca za działanie i sterowanie aplikacją.

**InterruptMenager** - klasa implementująca wywołania zwrotne dla obsługiwanych przerwań. Zapewnia synchronizację pomiędzy systemem operacyjnym i procesami a odpowiednimi zdarzeniami sygnalizowanymi przez przerwania.

**Model** - przestrzeń która, zawiera klasy związane z realizacją modelu z MVC

**AircraftRecord** - klasa do przechowywania informacji o samolotach. Wpisy są zgromadzane w liście. Wezły są wyszukiwane według adresu ICAO. Odpowiada modelowi danych z MVC.

**FlightControl** - implementuje logikę modelu, wraz z metodami dostępu i modyfikacji poszczególnych wpisów. Kontroluje czas życia poszczególnych węzłów oraz usuwa te które uległy przeterminowaniu. Informuje kontroler o zmianach.

**Controller** - przestrzeń nazw zawierająca klasy związane z realizacją kontrolera z MVC.

**ADS\_BDecoder** - klasa odbierająca od obiektu RTLSDR ciągi próbek sygnału radiowego. Odpowiedzialna za wykrywanie ramek od dekodowanie ich do typu ADS\_BMessage.

**ADS\_BMessage** - struktura reprezentująca odebraną wiadomość. W zależności od typu zawiera inne informacje o samolocie.

**FlightControlController** - reprezentacja kontrolera w MVC. Klasa jest odpowiedzialna na dodawanie i modyfikowanie wpisów w zależności od odebranych wiadomości. Zajmuje się również przetwarzaniem akcji systemu (przerwania od zegary odliczającego czas życia wpisu) i użytkownika (interakcja z panelem dotykowym). Przekazuje dane z modelu do widoku i informuje o potrzebie odświeżenia interfejsu.

**View** - klasy realizujące funkcjonalność widoku z MVC

**FlightControlView** - realizacja widoku w MVC. Klasa jest odpowiedzialna za prezentację danych z modelu oraz przesyłania akcji użytkownika do kontrolera.

**BoardSupport** - zawiera klasy będące wysokopoziomowymi interfejsami do obsługi urządzeń wchodzących w skład PCB. Jest odpowiedzialna za inicializację wszystkich klas z przestrzeni BoardSupport i Drivers.

**BoardMenager** - klasa zapewniająca komunikację pomiędzy aplikacją a sprzętem.

**Led** - interfejs do obsługi LED znajdującej się na płytce.

**RTLSDR** - klasa implementująca obsługę tunera DVB-T jako SDR korzystając z librtlsdr. Odpowiada za strojenie urządzenia i konfigurację układu próbującego.

**GPS** - zajmuje się przetwarzaniem wiadomości przychodzących od modułu GPS poprzez UART. Zapewnia dostęp do zdekodowanych współrzędnych geograficznych.

**Display** - kontroler wyświetlacza realizujący wysokopoziomowe funkcjonalności takie jak menedżer okien i obsługa widżetów.

**Drivers** - przestrzeń nazw zawierająca wszystkie sterowniki. Zaimplementowane klasy zapewniają interfejs do biblioteki HAL odpowiedzialnej za konfigurację i obsługę sprzętu.

**RCCDriver** - konfigurację linii zegarowych oraz pętli PLL (ang. Phase Locked Loop) dla peryferiów oraz rdzenia w mikrokontrolerze.

**GPIODriver** - konfiguracja pinów dla wszystkich interfejsów wraz z obsługą wyjść i zewnętrznych przerwań.

**USBDriver** - sterownik do obsługi USB2.0 Full-Speed. Odpowiada za wykrywanie podłączonych urządzeń oraz nawiązywanie z nimi połączenia.

**UARTDriver** - zapewnia obsługę interfejsu UART.

**FMCDriver** - konfiguracja kontrolera zewnętrznej pamięci mikrokontrolera. Wykorzystywany do komunikacji z układem SDRAM.

**LTDCTDriver** - obsługa kontrolera wyświetlacza LCD-TFT wraz z konfiguracją warstw.

**I2CDriver** - klasa do obsługi interfejsu I2C.

**DMA Driver** - sterownik kontrolera DMA (ang Direct Memory Access). Wykorzystywany do przyspieszenia operacji renderowania kolejnych klatek dla wyświetlacza.

Komunikacja pomiędzy komponentami odbywa się przy pomocy interfejsów. Każda klasa dziedziczy po swoim interfejsie w którym zadeklarowane są wszystkie metody niezbędne do komunikacji z obiektem. Pozwala to na odseparowanie rzeczywistej implementacji klasy. Dzięki temu dany komponent można łatwo zmieniać bez wpływu na te które z nim oddziałują. Podejście to ułatwia testowanie oprogramowania, ponieważ wewnętrzną logikę można zastąpić spreparowanymi danymi. Za przykład może posłużyć klasa obsługująca sensor, która zawsze zwraca przygotowany wcześniej odczyt.

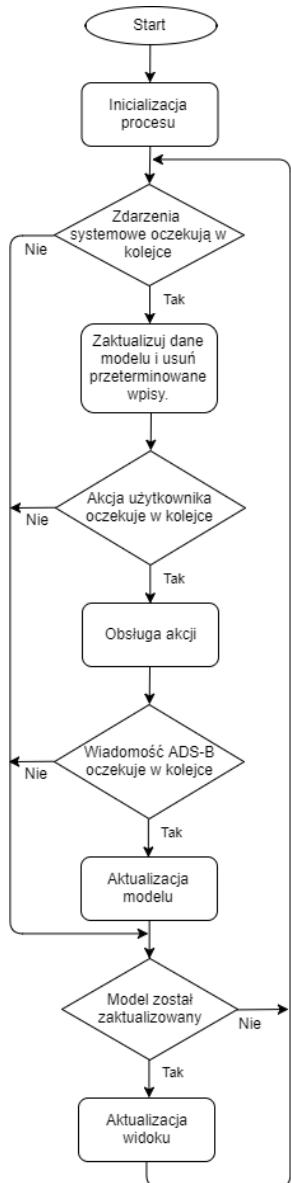
### 4.3 Procesy w systemie operacyjnym

W projekcie wykorzystano system operacyjny FreeRTOS, co pozwoliło podzielić poszczególne części aplikacji na procesy. Wielozadaniowość jest potrzebna, by jednocześnie obsługiwać interfejs użytkownika, zarządzać akwizycją danych oraz wykonywać obliczenia związane z logiką programu, tak aby użytkownik miał wrażenie płynnego działania aplikacji. Mechanizm semaforów wykorzystano do synchronizacji zadań z przerwaniami sprzętowym, natomiast kolejki pozwoliły na przesyłania pomiędzy nimi danych i zdarzeń. Ponadto użyto liczników programowych do odliczania czasu życia poszczególnych wpisów w modelu. Każdy wpis jest usuwany po 120s od otrzymania ostatniej dotyczącej go wiadomości.

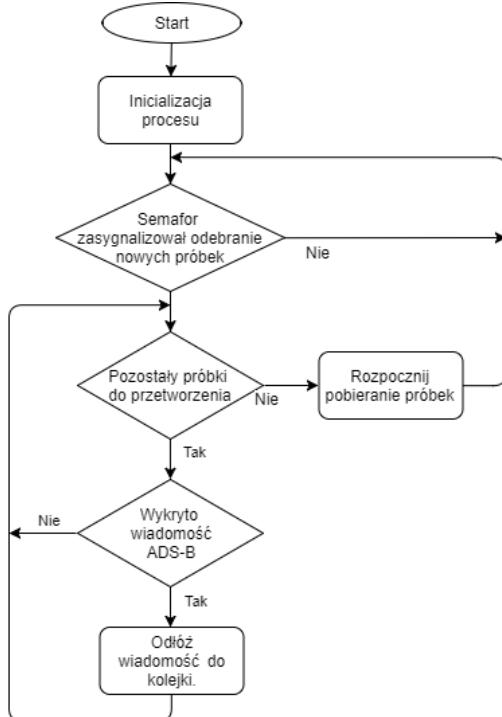
W systemie wydzielono następujące procesy:

- USBMonitorTask - odpowiedzialny za obsługę maszyny stanów USB oraz wykrywanie i konfigurację Tunera DVB-T jako SDR.
- RTLSDRDataAquisitionTask - proces zajmujący się obsługą komunikacji z SDR oraz wykrywaniem wiadomości ADS-B i przesyłaniem ich do FlightControlerTask poprzez kolejkę.
- FlightControlerTask - zadanie zajmuje się dekodowaniem wiadomości ADS-B. Przekazuje zdarzenia systemowe, takie jak przepełnienia liczników, oraz zdekodowanie ramki do kontrolera MVC FlightControlController.
- GUITask - proces obsługujący interfejs graficzny poprzez klasę FlightControlView.

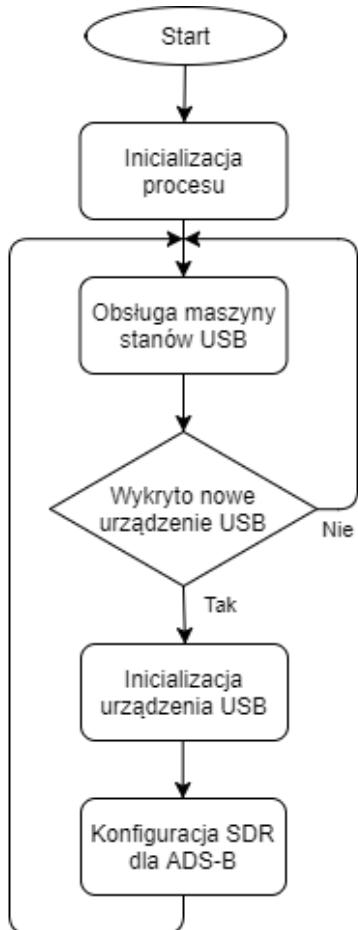
Poniżej przedstawiono schematy blokowe poszczególnych zadań.



Rysunek 4.3: Schemat blokowy zadania FlightControlerTask



Rysunek 4.4: Schemat blokowy zadania RTLSDRDataAquisitionTask



Rysunek 4.5: Schemat blokowy zadania  
USBMonitorTask



Rysunek 4.6: Schemat blokowy zadania  
GUITask

## 4.4 Wykrywanie i dekodowanie wiadomości ADS-B

### Wykrywanie Mode-S

Wiadomości Mode S mają długość 56 lub 112 bitów w zależności od formatu. Przed wysłaniem dane są poddawane modulacji położenia impulsu z częstotliwością sygnału zegarowego 1Mhz. Krótki odstęp pomiędzy stanami wysokimi oznacza logiczne 1 natomiast długi 0. Każdy bit jest kodowany co  $1\mu s$ . Na początku wiadomości dodawana jest preambuła o długości  $8\mu s$  składająca się z czterech impulsów o długości  $0.5\mu s$  w odstępach kolejno  $0.5\mu s$ ,  $2,5\mu s$ ,  $0,5\mu s$ . Tak przygotowany pakiet jest konwertowany do formatu I/Q i wysyłany na częstotliwość 1090Mhz.

SDR został skonfigurowany, by wydzielać sygnał z fali nośnej o częstotliwość 1090Mhz. Na jego wyjściu otrzymujemy przebieg o częstotliwości  $f = 2Mhz$ , ponieważ poszczególne impulsy trwają  $0,5\mu s$ . Zgodnie z twierdzeniem Kotielnikowa-Shannona do poprawnego odtworzenia sygnału potrzebujemy próbować go z częstotliwością co najmniej  $2f$ . W SDR próbki są pobierane przez dwa 8 bitowe przetworniki ADC. Przed jednym z układów następuje przesunięcie sygnału w fazie o  $-90^\circ$ . Dzięki temu dla częstotliwości próbki

2Mhz pasmo przenoszenie jest dwukrotnie szersze i również wynosi 2Mhz. Dane otrzymane z SDR są w postaci I/Q. Para próbek I/Q stanowi liczbę rzeczywistą gdzie I to chwilowa amplituda sygnału odebranego przez przetwornik, a to Q to chwilowa amplituda sygnału przesuniętego w fazie o  $-90^\circ$  względem I. Chwilowa amplituda nadanego sygnału to moduł liczby zespolonej I/Q obliczony z następującego wzoru

$$A = \sqrt{I^2 + Q^2} \quad (4.1)$$

Aby wykryć pakiet należy przeliczyć ciąg próbek I/Q na kolejne wartości amplitudy, według wzoru 4.1. Następnie znaleźć w sygnale preambułę oraz zdekodować wiadomości przy pomocy demodulacji PPM. Aby potwierdzić poprawność ramki należy obliczyć sumę CRC i porównać z wartością odebraną w wartością.

## Dekodowanie ADS-B

ADS-B to wiadomości Mode-S o parametrze Downlink format równym 17. Poniżej przedstawiono strukturę pakietu ADS-B.

Tabela 4.1: *Struktura wiadomości ADS-B*

Rozmiar [bit]	Zawartość
5	Downlink format
3	Wersja systemu Mode-S
24	Adres ICAO
5	Typ wiadomości
51	Dane
24	Suma kontrola CRC

W zależności od typu, wiadomość może zawierać następujące informacje.

Tabela 4.2: *Struktura wiadomości ADS-B*

Typ	Zawartość
1 - 4	Identyfikator samolotu i typ transpondera
5 - 8	Pozycja naziemna
9 - 18	Pozycja powietrzna i wysokość barometryczna
9	Prędkość
20-22	Pozycja powietrzna i wysokość GNSS
23-31	Zarezerwowane

Poniżej przedstawiono algorytmy dekodowania danych zawartych w wiadomości w zależności o typu.

Identyfikator samolotu (ang. callsign) jest przechowywany w wiadomości jako ciąg ośmiu 6 bitowych indeksów. Poszczególne znaki są odczytywane z poniższej tablicy

```
#ABCDEFGHIJKLMNPQRSTUVWXYZ#####
_#####0123456789#####
```

gdzie: # - znak nieużywany, \_ - spacja

Tabela 4.3: Struktura pakietu ADS-B typu 1-4

Rozmiar [bit]	3	6	6	6	6	6	6
Zawartość	Rodzaj transpondera	Znak 1	Znak 2	Znak 3	Znak 4	Znak 5	Znak 6

Pozycja statku powietrznego jest zakodowana w formacie CPR (ang. Compact Position Reporting). Wyróżniamy dwie metody dekodowania takich wiadomości. GUP (ang. Globally Unambiguous Position), która potrzebuje do dwóch kolejnych wiadomości i LUP (ang. Locally Unambiguous Position), gdzie wymagana jest pozycja referencyjna, np. lokalizacja odbiornika radiowego, która nie może być odalona od statku powietrznego o więcej niż 180Nm. Ze względu na ograniczoną możliwość anteny tunera w projekcie wykorzystano metodę LUP. Poniżej przedstawiono strukturę pakietu ADS-B dla wiadomości z pozycją.

Tabela 4.4: Zawartość wiadomości ADS-B w zależności od typu

Rozmiar [bit]	Nazwa	Zawartość
2	SS	Status lotu
1	NIC <sub>b</sub>	flaga dokładności pozycji
12	ALT	wysokość barometryczna lub GNSS
1	T	flaga synchronizacji z czasem UTC
1	F	flaga parzystości ramki
17	Lat <sub>raw</sub>	zakodowana szerokość geograficzna w formacie CPR
17	Lon <sub>raw</sub>	zakodowana długość geograficzna w formacie CPR

Pozycja w formacie CPR to dwie wartości zmiennopozycyjne w zakresie od 0 do 1. W wiadomości są zakodowane jako 17 bitowe liczby całkowite w formacie Q1.17. Konwersja z formatu Q1.17 do CPR odbywa się według następującego wzoru:

$$L_{CPR} = \frac{L_{raw}}{2^{17}} \quad (4.2)$$

gdzie:

- $L_{CPR}$  - zdekodowana długość lub szerokość geograficzna w formacie CPR.
- $L_{raw}$  - długość lub szerokość geograficzna zakodowana w formacie Q1.17.

Poniżej przedstawiono algorytm dekodowania pozycji samolotu.

Znaczenie użytych symboli:

- NZ to ustalona liczba stref szerokości geograficznych wynosząca 15.
- $\text{Lat}_{ref}$  - referencyjna szerokość geograficzna
- $\text{Lon}_{ref}$  - referencyjna długość geograficzna
- $\text{Lat}_{CPR}$  - szerokość geograficzna w formacie CPR
- $\text{Lon}_{CPR}$  - długość geograficzna w formacie CPR
- Lat - zdekodowano szerokość geograficzną
- Lon - zdekodowana długość geograficzną

Definicje użytych funkcji:

$$\text{mod}(x, y) = x - y \left\lfloor \frac{x}{y} \right\rfloor \quad (4.3)$$

$$NF(Lat) = \left\lfloor \frac{2\pi}{\arccos \left( 1 - \frac{1 - \cos \left( \frac{2\pi}{2NZ} \right)}{\cos^2 \left( \frac{\pi}{180} \cdot Lat \right)} \right)} \right\rfloor \quad (4.4)$$

Poniżej przedstawiono kolejne etapy obliczeń.

$$dLat = \begin{cases} \frac{360}{4NZ} = \frac{360}{60}, & F = 0 \\ \frac{360}{4NZ-1} = \frac{360}{59}, & F = 1 \end{cases} \quad (4.5)$$

gdzie: F to flaga parzystości wiadomości.

$$j = \left\lfloor \frac{\text{Lat}_{ref}}{dLat} \right\rfloor + \left\lfloor \frac{\text{mod}(\text{Lat}_{ref}, dLat)}{dLat} - \text{Lat}_{CPR} + \frac{1}{2} \right\rfloor \quad (4.6)$$

Obliczone parametry pozwoliły na obliczenia szerokość geograficznej według następującego wzoru:

$$\text{Lat} = dLat \cdot (j + \text{Lat}_{CPR}) \quad (4.7)$$

Następnie wyznaczono parametry:

$$dLon = \begin{cases} \frac{360}{NL(Lat)}, & \text{gdy } NL(Lat) > 0 \\ 360, & \text{gdy } NL(Lat) = 0 \end{cases} \quad (4.8)$$

$$m = \left\lfloor \frac{Lon_{ref}}{dLon} \right\rfloor + \left\lfloor \frac{mod(Lon_{ref}, dLon)}{dLon - Lon_{CPR} + \frac{1}{2}} \right\rfloor \quad (4.9)$$

Parametry  $m$  i  $dLon$  wykorzystano do obliczenia długości geograficznej według poniższego wzoru.

$$Lon = dLon \cdot (m + Lon_{CPR}) \quad (4.10)$$

Pułap lotu jest zakodowany na polu ALT w wiadomości ADS-B. Zdekodowanie wymaga usunięcia z liczby 7. bitu (flagi Q), a następnie obliczenia według następującego wzoru:

$$A = \begin{cases} 100N - 1000, & \text{gdy } Q = 0 \\ 25N - 1000, & \text{gdy } Q = 1 \end{cases} \quad (4.11)$$

gdzie:

- N - wartość pola ALT uśnięciu bitu Q.
- A - zdekodowany pułap lotu w stopach.

Wiadomości typu 9 dzielimy na dwa rodzaje. Z prędkością względem powierzchni ziemi (Subtype 1) lub powietrzna (Subtype 3). Poniżej przedstawiono format pakietu.

Tabela 4.5: *Zawartość wiadomości ADS-B z prędkością w zależności od podtypu.*

Rozmiar [bit]	Nazwa		Zawartość (Subtype 1)	Zawartość (Subtype 3)
1	IC		Flaga zmiany pułapu	
1	RES-A		Rejestr zarezerwowany	
3	NAC		Niepewność prędkości	
1	$S_{EW}$	$S_{Hdg}$	Znak prędkości Wschód-Zachód	Flaga dostępności kursu
10	$V_{EW}$	Hdg	Prędkość Wschód-Zachód	Kurs
1	$S_{NS}$	$AS_t$	Prędkości Północ-Południe	Typ prędkości powietrznej
10	$V_{NS}$	AS	Prędkość powietrzna	Prędkość powietrzna
1	$Vr_{src}$		Flaga typu wysokości Barometryczna lub GNSS	
1	$S_{Vr}$		znak zmiany pułapu	
1	Vr		prędkość zmiany pułapu	
2	RES-B		rejestr zarezerwowany	
1	$S_{Diff}$		znak różnicy wysokości barometrycznej i GNSS	
7	Diff		różnica wysokości barometrycznej i GNSS	

Poniżej przedstawiono algorytmy dekodowania kursu i szybkości w zależności od typu wiadomości.

- v - szybkość statku w węzłach
- h - kurs statku w stopniach

Dla wiadomości ADS-B Subtype 1 algorytm wygląda następująco:

$$dV_{EW} = \begin{cases} V_{EW} - 1, & \text{gdy } S_{EW} = 0 \\ -(V_{EW} - 1), & \text{gdy } S_{EW} = 1 \end{cases} \quad (4.12)$$

$$dV_{SN} = \begin{cases} V_{SN} - 1, & \text{gdy } S_{SN} = 0 \\ -(V_{SN} - 1), & \text{gdy } S_{SN} = 1 \end{cases} \quad (4.13)$$

Następnie można wyznaczyć kurs oraz szybkość statku według następujących wzorów.

$$v = \sqrt{dV_{EW}^2 + dV_{SN}^2} \quad (4.14)$$

$$h = \arctan2(dV_{EW}, dV_{SN}) \cdot \frac{180}{\pi} \quad (4.15)$$

Dla ADS-B Subtype 3 algorytm wygląda następująco:

$$h = \frac{Hdg}{1024} \cdot 360, \text{ gdy } S_{Hdg} = 1 \quad (4.16)$$

$$v = AS \quad (4.17)$$

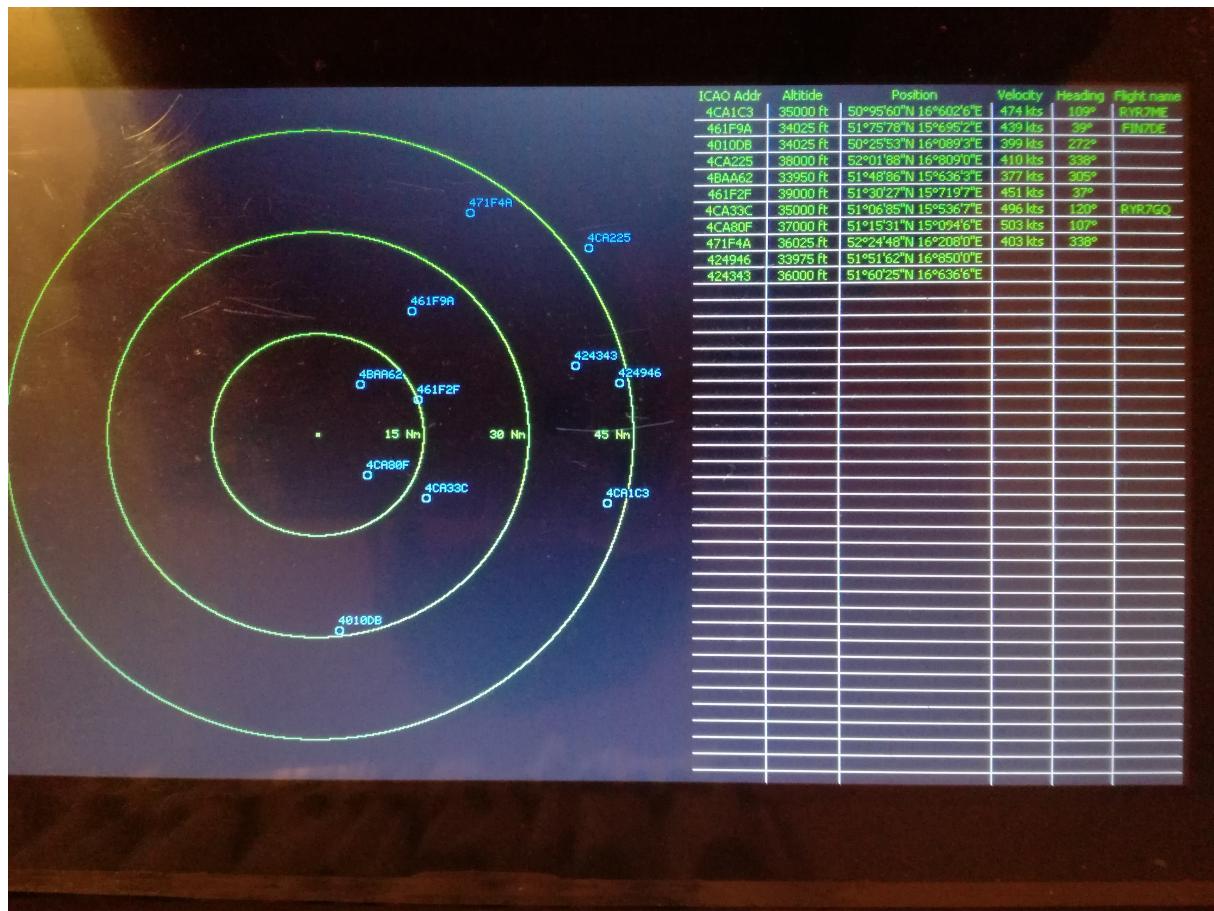
Jednak te wiadomości są wysyłane tylko przez samoloty starszego typu i większość odbieranych pakietów stanowi\_subtype 1.

# Rozdział 5

## Interfejs użytkownika

W tym rozdziale opisano poszczególne elementy interfejsu użytkownika wraz z opisem użytych widżetów z biblioteki STemWIn i formatem prezentowanych danych.

### Prezentacja GUI



Rysunek 5.1: Wygląd interfejsu użytkownika

---

## Opis Widżetów

Interfejs graficzny użytkownika GUI (ang. Graphical User Interface) podzielono na dwie części. Pierwsza warstwa pełni rolę tła i składa się z jednego widżetu IMAGE, który prezentuje obraz pustego radaru. Element jest renderowany tylko raz, po włączeniu urządzenia co pozwala oszczędzić moc obliczeniową mikrokontrolera. Druga warstwa jest podwójnie buforowana i znajduje się na niej pierwszy plan składa się z dwóch Widżetów. LIST\_VIEW prezentującego informacje o wykrytych samolotach w formie listy. Poszczególne kolumny zawierają kolejno: adres ICAO, pułap lotu w stopach, współrzędne geograficzne, prędkość w węzłach, kurs w stopniach i identyfikator samolotu. Drugi widżet typu RADAR został wykonany na potrzeby projektu. Jego zadaniem pokazywanie znaczników wykrytych samolotów na przeźroczystym tle. Gdy LTDC połączy warstwę tła i pierwszego planu otrzymamy pełny obraz z pozycją samolotów na tle radaru.

## Rozdział 6

### Podsumowanie

# Bibliografia

- [1] Stephen H. Hal, Garrett W. Hall, and James A. McCall. *High-Speed Digital System Design - A Handbook of Interconnects Theory and Design Practices*. A Wiley-Interscience Publication JOHN WILEY & SONS, INC. New York, Chichester, Weinheim, Brisbane, Singapore, Toronto, 2000.
- [2] Richard G. Lyons *Understanding Digital Signal Processing*. Eight Printing, 2001.
- [3] Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC, Philips *Universal Serial Bus Specification*. Revision 2.0, 2000
- [4] Junzi Sun. *ADS-B Decoding Guide*.  
<https://media.readthedocs.org/pdf/adsb-decode-guide/latest/adsb-decode-guide.pdf>