AKADEMIA MORSKA
SZCZECIN

Wydział Informatyki i Telekomunikacji

Katedra Elektroniki i Telekomunikacji

Numer ewidencyjny …….

Data pobrania tematu pracy …….

Data zdania pracy ……

PRACA DYPLOMOWA
INŻYNIERSKA

Dyplomant	Marek Jakóbiak	Nr albumu: 24135
Kierunek/Specjalność	Informatyka/Informatyka Morska	
Promotor	dr inż. Marcin Mąka	Ocena:
Recenzent		Ocena:
Egzamin dyplomowy - data		

TEMAT: Opracowanie aplikacji do odbioru informacji z radiostacji MF/HF

Dyplomant…… Promotor…… Dziekan……

AKADEMIA MORSKA W SZCZECINIE

WYDZIAŁ ELEKTRONIKI I TELEKOMUNIKACJI

Marek Jakóbiak

PRACA DYPLOMOWA INŻYNIERSKA

Temat: Opracowanie aplikacji do odbioru informacji z radiostacji MF/HF

Praca wykonana w Katedrze Elektroniki i Telekomunikacji
pod kierunkiem dr inż. Marcina Mąki

Szczecin 2021

Oświadczam, że praca pt. **„Opracowanie aplikacji do odbioru informacji z radiostacji MF/HF”** jest pracą dyplomową mojego autorstwa. Wszystkie dane i sformułowania pochodzące z literatury są opatrzone odpowiednimi odsyłaczami. Praca ta nie była w całości ani w części przez nikogo przedłożona do żadnej oceny i nie była publikowana.

…… ……

Imię i nazwisko *własnoręczny podpis*

Oświadczam, że wersja elektroniczna mojej pracy dyplomowej inżynierskiej pt. **„Opracowanie aplikacji do odbioru informacji z radiostacji MF/HF”** jest zgodna z wersją drukowaną.

…… ……

Imię i nazwisko *własnoręczny podpis*

Wyrażam zgodę na udostępnianie mojej pracy w Czytelni Biblioteki Głównej Akademii Morskiej w Szczecinie oraz na udostępnianie elektroniczne w sieci BG.

Data …… Podpis ……

SPIS TREŚCI

WYKAZ UŻYTYCH SKRÓTÓW I SYMBOLI

Skrót	Określenie	
	W języku angielskim	W języku polskim
AM	Amplitude Modulation	Modulacja amplitudy
ASK	Amplitude-Shift Keying	Kluczowanie amplitudy
COMSAR	Sub-Committee on Radiocommunciations and Search and Rescue	Podkomisja Radiokomunikacji, Poszukiwań i Ratownictwa
COSPAS-SARSAT	b.d.	Międzynarodowy satelitarny system dla poszukiwania i ratownictwa
DSC	Digital Selective Calling	Cyfrowe selektywne wywołanie
DTR	Data Terminal Ready	Terminal danych gotowy
ECC	Error Correction Coding	Kodowanie korekcyjne
EPIRB	Emergency Position-Indicating Radio Beacon	Awaryjny, Wskazujący Pozycję, Nadajnik Radiowy
FEC	Forward Error Correction	Zwrotna korekcja błędów
FM	Frequency Modulation	Modulacja częstotliwości
FSK	Frequency-Shift Keying	Kluczowanie częstotliwości
GMDSS	Global Maritime Distress and Safety System	Światowy Morski System Łączności Alarmowej i Bezpieczeństwa
HF	High Frequency	Fale krótkie
IMO	International Maritime Organization	Międzynarodowa Organizacja Morska
INMARSAT	International Maritime Satellite	Międzynarodowy System Satelitów Morskich

ITU	International Telecommunication Union	Międzynarodowy Związek Telekomunikacyjny
MF	Medium Frequency	Fale średnie
MID	Maritime Identification Digits	Morskie cyfry identyfikujące
MMSI	Maritime Mobile Service Identity	Morski numer identyfikacyjny
MRCC	Maritime Rescue Coordination Centre	Morska Służba Poszukiwania i Ratownictwa
NBDP	Narrow Band Direct Printing	Wąskopasmowa telegrafia dalekopisowa
PM	Phase Modulation	Modulacja fazy
SAR	Search And Rescue	Poszukiwanie i ratownictwo
SOLAS	International Convention for the Safety of Life at Sea	Międzynarodowa konwencja o bezpieczeństwie życia na morzu.
SSB	Single SideBand	Modulacja jednowstęgowa
UI	User Interface	Interfejs Użytkownika
VHF	Very High Frequency	Fale ultrakrótkie

Określenie	Wyjaśnienie
Application Programming Interface API	Interfejs programowania aplikacji to zbiór reguł opisujący sposób w jaki programy komunikują się ze sobą
Asterix (*)	Symbol typograficzny reprezentujący gwiazdkę, którego nazwa pochodzi ze starożytnej greki (*asteriskos*) oznaczający „małą gwiazdę”.
Binding	Nazwa sposobu na powiązanie danych interfejsu użytkownika z danymi z części logicznej. Pozwala na wymianę informacji pomiędzy dwoma warstwami aplikacji WPF.
Bitrate	Ilość transmitowanych poprzez kanał komunikacyjny informacji w danej jednostce czasu, wyrażone w bitach.
Code Behind	Wzorzec występujący w WPF, który pozwala na umieszczenie logiki aplikacji poza warstwą prezentacji, w oddzielnych plikach. Pliki są kojarzone z widokami, poprzez nazwy. Przykładowo, plik z logiką w języku C# (cs) dla widoku *MainWindow.xaml* będzie nazywał się *MainWindow.xaml.cs*.
ComboBox	Kontrolka w warstwie widoku odpowiadająca za pole wyboru. Umożliwia wybór jednej możliwości z wielu.
DirectX	API firmy Microsoft wspomagające generowanie dźwięku, grafiki oraz innych zadań związanych z aplikacjami multimedialnymi, m. in. grami komputerowymi.
Extensible Application Markup Language XAML	Język opisujący interfejs użytkownika, będący połączeniem języków HTML i XML, używany przede wszystkim w WPF (Windows Presentation Foundation)
eXtensible Markup Language XML	Prosty i uniwersalny format tekstowy pochodzący bezpośrednio z języka SGML (ISO 8879). Służy reprezentacji różnego rodzaju danych w ustrukturyzowany sposób.

Framework	Szkielet służący do budowy aplikacji. Zawiera w sobie zestaw komponentów oraz bibliotek, tym samym definiując strukturę oraz mechanizm działania aplikacji stworzonej z jego pomocą.
Garbage Collector	Programowy zarządca odpowiedzialny za odśmiecanie pamięci. Zwalnia nieużywaną już pamięć, która wcześniej została przydzielona w sposób dynamiczny.
GDI+ Graphics Device Interface+	Wraz z jądrem oraz API Windowsa, jeden z trzech podstawowych komponentów. Odpowiada za przedstawienie elementów graficznych, a następnie wysłanie do odpowiednich urządzeń wyjściowych (fax, drukarka, monitor).
Getter	Typ właściwości odpowiedzialnej za pobranie zmiennej.
HyperText Markup Language HTML	Hipertekstowy język znaczników służący do opisu struktury informacji w stronach internetowych.
Model	Abstrakt będący składową MVVM, którego zadaniem jest przechowywanie oraz walidacja prowadzanych danych.
Model–View–ViewModel MVVM	Wzorzec architektury programowania polegający na rozdzieleniu od siebie widoków (view), modeli (model) oraz dodanie pomiędzy nich abstrakcji umożliwiającej wymianę danych – model widoku (ViewModel)
Open source	Typ oprogramowania komputerowego, gdzie kod źródłowy rozpowszechniany jest na podstawie licencji pozwalającej wszystkim jego użytkownikom na swobodne uruchamianie, rozpowszechnianie i edycję.
Responsywność	Słowo utworzone od angielskiego *response* – odpowiedź. Określa czas potrzebny interfejsowi na reakcję na działanie użytkownika. Im niższa responsywność, tym opóźnienie jest większe.
Setter	Typ właściwości odpowiedzialnej za ustawienie zmiennej.
Timeout	Limit czasu przewidziany na dane zdarzenie lub operację.

User eXperience UX	Całość wrażeń składających się na doświadczenie użytkownika w trakcie korzystania z produktu.
View	Warstwa widoku w MVVM zawierająca w sobie interfejs użytkownika. Odpowiada za interakcję z użytkownikiem.
ViewModel	Abstrakcja będąca składową MVVM pośrednicząca w wymianie informacji pomiędzy widokiem (*view*), a modelem (model). Posiada dostęp do danych widoku oraz struktur i obiektów modeli.

CEL I ZAKRES PRACY

Celem niniejszej pracy jest opracowanie aplikacji, która umożliwiałaby obiór i archiwizację informacji otrzymywanych z radiostacji MF/HF z przystawką DSC, za pośrednictwem komunikacji szeregowej (typu COM). Informacje te obejmują zapis dźwiękowy rozmowy przeprowadzanej za pomocą radiostacji oraz komplet danych zawartych w wywołaniach DSC. Całość powinna zostać przedstawiona w sposób umożliwiający łatwe odczytanie zebranych informacji przez operatora radiostacji i komputera.

WSTĘP

Wykorzystanie fali elektromagnetycznej do transmisji danych, wytworzonej po raz pierwszy w 1886 roku przez Heinricha Hertza, nieodwracalnie zmieniło obraz współczesnej cywilizacji. Chociaż odkrycie samego radia w 1895 zawdzięczamy Nikoli Tesli, to jednak dopiero włoski wynalazca Guglielmo Marconi zdołał wzbudzić powszechne zainteresowanie nowym sposobem przesyłania informacji nadając – 27 lipca 1896 roku – sygnał z dachu Poczty Głównej w Londynie do oddalonego o kilometr odbiornika. Prezentacja wywołała wśród widzów ogromne zainteresowanie co zaowocowało pierwszą udaną próbną transmisją sygnału przez kanał La Manche w roku 1899, a w 1901 emisją, z Anglii do Kanady, tym razem konkretnej już informacji: poprzez Ocean Atlantycki nadano literę „S".

Wybuch I Wojny Światowej w roku 1914 przyspieszył nieco rozwój badań nad radiokomunikacją, jednakże jego apogeum przypada dopiero na lata dwudzieste XX wieku. Na przestrzeni lat 1922-1933 na terenach wszystkich państw europejskich oraz w całych Stanach Zjednoczonych powstały liczne radiostacje nadające regularne programy i audycje, natomiast przystępna cena radioodbiorników sprawiła, że urządzenie to stało się powszechne nawet w domach średniozamożnych rodzin. Stworzyło to niespotykaną wcześniej możliwość dotarcia do mas, co w krótkim czasie zostało z powodzeniem wykorzystane na przykład w celach politycznych: komunikacja radiowa stała się narzędziem propagandowym.

Równolegle do zastosowań cywilnych, radiofonia została użyta również w celach militarnych. Wraz z jej rozprzestrzenieniem się zarówno na lądzie jak i na morzu, zaczęła uwydatniać się potrzeba zabezpieczania transmitowanych informacji. Obawa przed dostaniem się w niepowołane ręce wrażliwych danych zapoczątkowała metodę kodowania nadawanych komunikatów oraz badania nad szyfrowaniem. Niemniej jednak, dopiero wprowadzenie cyfrowych nadajników operujących na danych w formie binarnej przyczyniło się do rozwoju kryptologii i nadało jej szczególnego znaczenia.

Wraz z końcem Zimnej Wojny zaczęto skupiać się na rozwoju technologii mającej na celu poprawę bezpieczeństwa na wodach morskich i śródlądowych oraz zmniejszenie szkód wyrządzanych przez błędy nawigacyjne. Opracowując nowy system przyjęto założenie, że podstawą sprawnego i bezkolizyjnego przemieszczania się wszelkiego rodzaju jednostek pływających jest zachowanie efektywnej komunikacji zarówno

pomiędzy tymi jednostkami, jak i pomiędzy poszczególnymi punktami nadzorującymi ich manewry. Jednym z kroków milowych w zakresie bezpieczeństwa ruchu okrętów było stworzenie – w roku 1992 – Światowego Morskiego Systemu Łączności Alarmowej i Bezpieczeństwa (*Global Maritime Distress and Safety System – GMDSS*). Siedmioletni okres jego wdrażania zakończył się z dniem 01.02.1999, o czym oficjalnie poinformowano podczas IV Sesji Podkomitetu ds. Radiokomunikacji oraz Poszukiwań i Ratownictwa (*Sub-Committee on Radiocommunciations and Search and Rescue – COMSAR*) i Międzynarodowej Organizacji Morskiej (*International Maritime Organization – IMO*), która odbyła się w lipcu tego samego roku w Londynie.

Tak długi okres implementacji umożliwił przeprowadzenie wielu miarodajnych doświadczeń z zakresu różnych dziedzin, które obejmowały, między innymi, eksploatację systemu, weryfikację procedur operacyjnych, a nawet konstrukcję specjalistycznych urządzeń GMDSS. Jednymi z najważniejszych spośród zrealizowanych prób były te, które dotyczyły analiz pracy systemu w warunkach rzeczywistych, a w szczególności funkcjonowania podsystemów umożliwiających prowadzenie łączności oraz alarmowanie w sytuacjach zagrożenia.

Jednym z wielu podsystemów GMDSS, służącym do komunikacji w sytuacjach alarmowych, jest cyfrowe wywołanie selektywne (*Digital Selective Calling – DSC*). Jest to przystawka wchodząca w skład radiostacji VHF czy MF/HF.

Przed pojawieniem się przystawki DSC jednym z jej największych problemów był brak możliwości adresacji korespondencji do danego radioodbiornika. Było to wyjątkowo kłopotliwe na akwenach o wysokim natężeniu ruchu, a także w portach. Często dla takich obszarów ilość dostępnych kanałów nie była wystarczająca. Sytuacja uległa zmianie w momencie wynalezienia nowego typu radiostacji, której wykorzystanie znalazło zastosowanie przede wszystkim w sytuacjach wymagających zautomatyzowanego nawiązywania łączności. Okazało się to nieocenione w radiokomunikacji morskiej. Opracowane na tej zasadzie DSC zostało przewidziane do pracy w paśmie pośredniofalowym (MF; 2Mhz), krótkofalowym (HF; 4Mhz, 6Mhz, 8Mhz, 12Mhz, 16Mhz) oraz ultrakrótkofalowym (VHF; 156 – 174 Mhz). Jego głównym zadaniem jest przesyłanie wywołań alarmowych w sytuacjach niebezpieczeństwa (*distress*), niemniej jednak odpowiedzialny jest on również za automatyczne nawiązywanie łączności w celach publicznych oraz eksploatacyjnych. Stało się to możliwe dzięki zaimplementowaniu w ramkę komunikatu pola kategorii. Szeroki wybór kategorii pozwala radiooperatorom w prosty sposób określić powód nadania informacji oraz jej charakter. W celu

zminimalizowania ryzyka zaistnienia problemów z odbiorem i przetwarzaniem komunikatów, format ramki przesyłanego komunikatu zawartego w dokumentacji technicznej jest ściśle ustalony normami Międzynarodowego Związku Telekomunikacyjnego. Ramka komunikatu posiada również pole służące do kontroli poprawności przesyłanych informacji – znak detekcji błędu o wielkości 8 bitów. Jego mechanizm opiera się na kontroli parzystości otrzymanych danych, dzięki której w wypadku wykrycia niepoprawności, radiostacja wysyła zapytanie o powtórzenie komunikatu w celu dokonania ewentualnej korekty.

1. PRZESYŁANIE INFORMACJI W ETERZE

1.1 Podstawowe informacje o GMDSS

Dzień 01.02.1992 r. jest jednym z ważniejszych w historii radiotelekomunikacji morskiej, ponieważ to właśnie wtedy rozpoczęto wdrażanie Światowego Morskiego Systemu Łączności Alarmowej i Bezpieczeństwa GMDSS. Dotychczasowy system łączności i bezpieczeństwa opierał się na definicji zawartej w V rozdziale Konwencji o bezpieczeństwie życia na morzu SOLAS, która obejmowała szereg wymagań. Według nich pewne klasy statków, przebywając na morzu, powinny prowadzić stały nasłuch radiowy na międzynarodowych częstotliwościach bezpieczeństwa, zgodnie z Regulaminem Radiokomunikacyjnym Międzynarodowego Związku Telekomunikacyjnego ITU. W wyposażeniu tych statków winny znajdować się radiowe urządzenia nadawcze pozwalające emitować sygnały na określony minimalny zasięg. Do czasu wprowadzenia GMDSS przeznaczone do tego były dwa ręcznie obsługiwane systemy [5]:

- telegraf Morse'a – stosowany na częstotliwościach 500 kHz, wymagany dla wszystkich statków pasażerskich oraz dla statków towarowych o wyporności powyżej 1600 ton,
- radiotelefon – stosująca częstotliwości 2182 kHz oraz 156,8 MHz (kanał nr 16 dla VHF), wymagany dla wszystkich statków pasażerskich, a także dla statków towarowych o wyporności powyżej 300 ton.

Dotychczasowy system posiadał szereg zasadniczych wad [4]:

- wymagany minimalny zasięg nadajników emitujących sygnały alarmowe na częstotliwości 500 kHz i znajdujących się na pokładach jednostek pływających wynosił jedynie 100-150 km. Uniemożliwiało to zaalarmowanie innych statków oraz stacji nadbrzeżnych znajdujących się w większej odległości, a w rejonach o niskim natężeniu ruchu stawało się to niemal niemożliwe;
- na morzu, na obszarach znacznie oddalonych od brzegu, alarmowanie ograniczone było jedynie do statków, które znajdowały się w pobliżu źródła emisji, co praktycznie uniemożliwiało udzielenie pomocy z wykorzystaniem, przez brzegowe ośrodki Morskiego Centrum Koordynacji Ratownictwa MRCC, akcji ratowniczych SAR;
- niedostępność zautomatyzowanego systemu odpowiedzialnego za nawiązywanie łączności fonicznej lub telegraficznej w relacji ląd-statek i statek-ląd

uniemożliwiała zorganizowanie odpowiednio szybkiej pomocy ratowniczej, a przede wszystkim włączenia do akcji poszukiwawczo-ratowniczej innych statków znajdujących się w najbliższej okolicy wypadku lub katastrofy;

- podstawa tego systemu – telegrafia Morse'a – była podatna na różnego rodzaju zakłócenia i zmiany warunków propagacji fal, obniżając tym samym jakość oraz efektywną szybkość przesyłania informacji, a zamontowanie jakiegokolwiek systemu korekcyjnego uniemożliwiała niedetekcyjność kodu telegrafii Morse'a;
- charakter czynności manualnych związanych z nadaniem komunikatu z zastosowaniem telegrafii Morse'a może – szczególnie w nagłych sytuacjach – stwarzać operatorowi trudności, które mogły spowodować błędny odbiór pozycji statku, a tym samym do nieskutecznego naprowadzania na miejsce katastrofy na miejsce katastrofy;
- brak systemu radiokomunikacyjnego, mogącego w momencie tonięcia okrętu w sposób automatyczny alarmować ratownicze centrum brzegowe, okoliczne statki brzegowe bądź samoloty; istniała jedynie możliwość ciągłego nadawania sygnałów w celu identyfikacji i naprowadzenia na miejsce katastrofy.

W świetle przedstawionych powyżej wad tradycyjnego systemu radiokomunikacyjnego, stworzenie systemu nowej generacji, który pozwoliłoby na podniesienie stopnia bezpieczeństwa i skuteczności akcji ratowniczych prowadzonych na wodach, jawi się jako nagląca konieczność.

Odpowiedzią na takie zapotrzebowanie był wdrożony, po wspomnianym wcześniej siedmioletnim okresie przejściowym, system GMDSS. Dzięki wprowadzeniu najnowszych osiągnięć techniki, wykorzystujących przede wszystkim geolokalizację, cyfrowe systemy radiotelegrafii automatycznej oraz cyfrowe selektywne wywołanie DSC, stało się możliwe opracowanie systemu, w którym proces nadawania i odbioru sygnałów alarmujących o bezpieczeństwie jest w stanie zachodzić automatycznie. Udało się osiągnąć automatyczne, a także niezależne od warunków meteorologicznych, propagacyjnych, oraz pozycji geograficznej statku, zestawianie połączeń radiokomunikacyjnych w relacji statek-ląd i ląd-statek. Tak duża niezależność od nieprzewidywalnych warunków na morzu stała się możliwa poprzez zastosowanie szeregu środków łączności wykorzystujących radiowe zakresy częstotliwości pasma średniofalowego MF, pasma krótkofalowego HF, pasma VHF, a także używających częstotliwości satelitarnych takich jak pasma L (1,5 – 1,6 GHz) oraz pasma C (4 – 6 GHz). Zostały zdefiniowane odpowiednie częstotliwości do wysyłania

sygnałów alarmowych za pomocą cyfrowego selektywnego wywołania DSC, radiopław awaryjnych EPIRB w systemie COSPAS-SARSAT oraz INMARSAT.

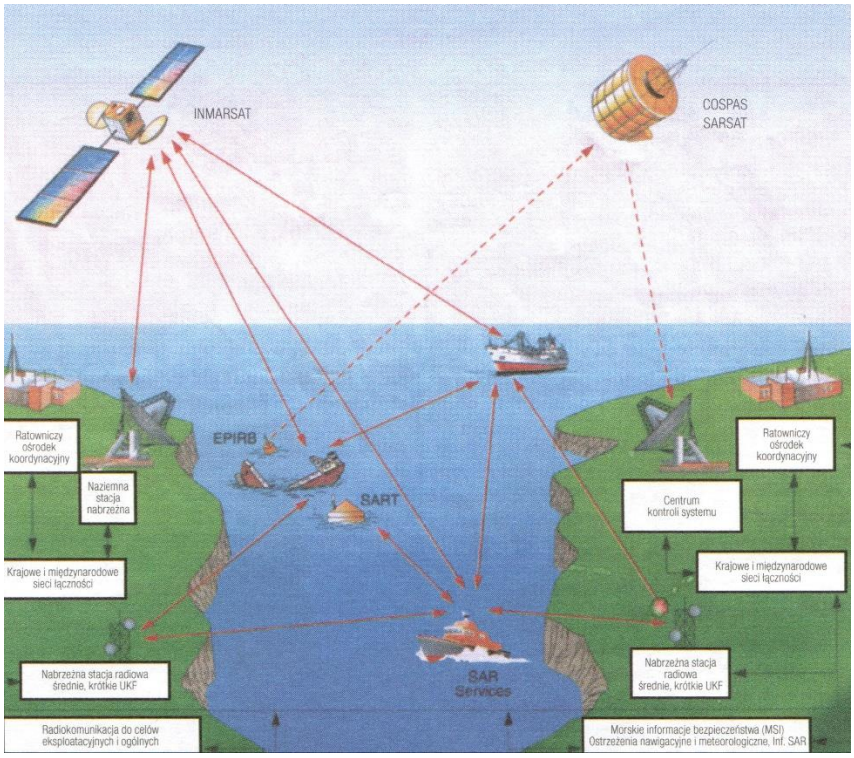
Podstawowe założenia i koncepcję systemu GMDSS przedstawiono na rysunkach 1.1 – 1.4. Według niej ośrodki koordynacji ratownictwa (MRCC i SAR) oraz statki będące w pobliżu miejsca katastrofy zostaną natychmiastowo poinformowane o zaistniałej, groźnej dla życia i mienia, sytuacji tym samym tworząc warunki do podjęcia skoordynowanej akcji ratowniczej. Warte zaznaczenia jest tutaj minimalne opóźnienie, tak ważne dla zwiększenia szans powodzenia akcji ratowniczych.

W celu minimalizacji ryzyka nieprzewidzianych zdarzeń, system, umożliwia także radiokomunikację w celach pilnych i bezpieczeństwa, oraz rozpowszechnianie morskich informacji bezpieczeństwa takich jak: ostrzeżenia nawigacyjne i meteorologiczne, prognozy pogody, a także szeregu innych pilnych informacji.

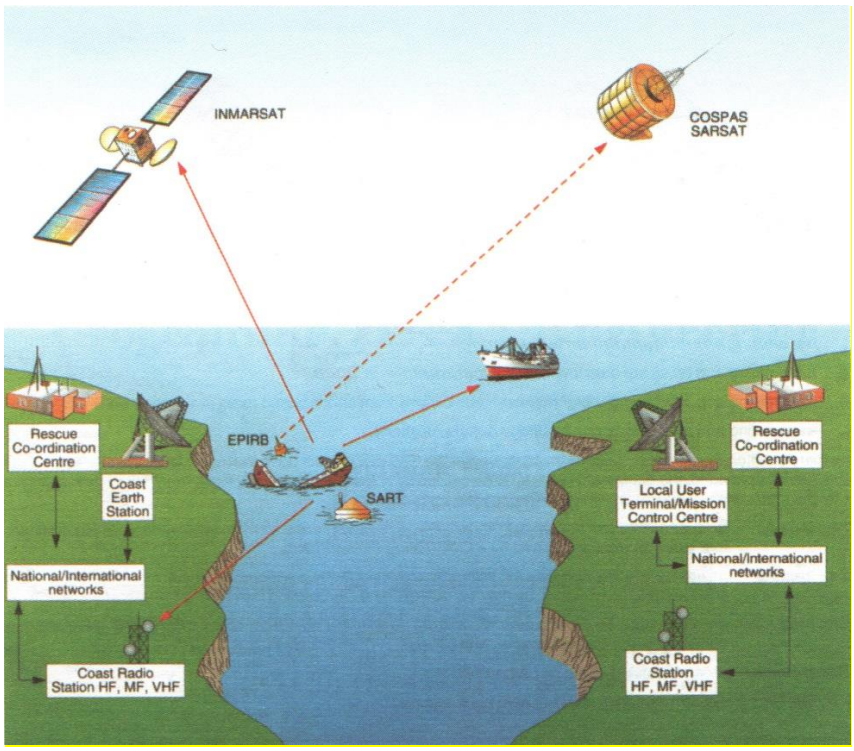
Każdy statek objęty tym systemem, niezależnie od położenia na morzu, musi spełniać wszelkie funkcje radiokomunikacyjne będące niezbędne dla zapewnienia bezpieczeństwa sobie i statkom żeglującym w tym samym rejonie. W przypadku ośrodków lądowych, wszystkie administracje morskie podpisując poprawki Konwencji SOLAS dotyczące wprowadzenia nowego systemu radiokomunikacyjnego GMDSS zobowiązały się do zainstalowania odpowiedniej aparatury radiokomunikacyjnej w celu zapewnienia realizacji łączności w relacji ląd-morze i na odwrót.

Podsumowując, wszystkie okręty objęte systemem GMDSS, w momencie wypłynięcia z portu na akwen morski, muszą być zdolne do [4]:

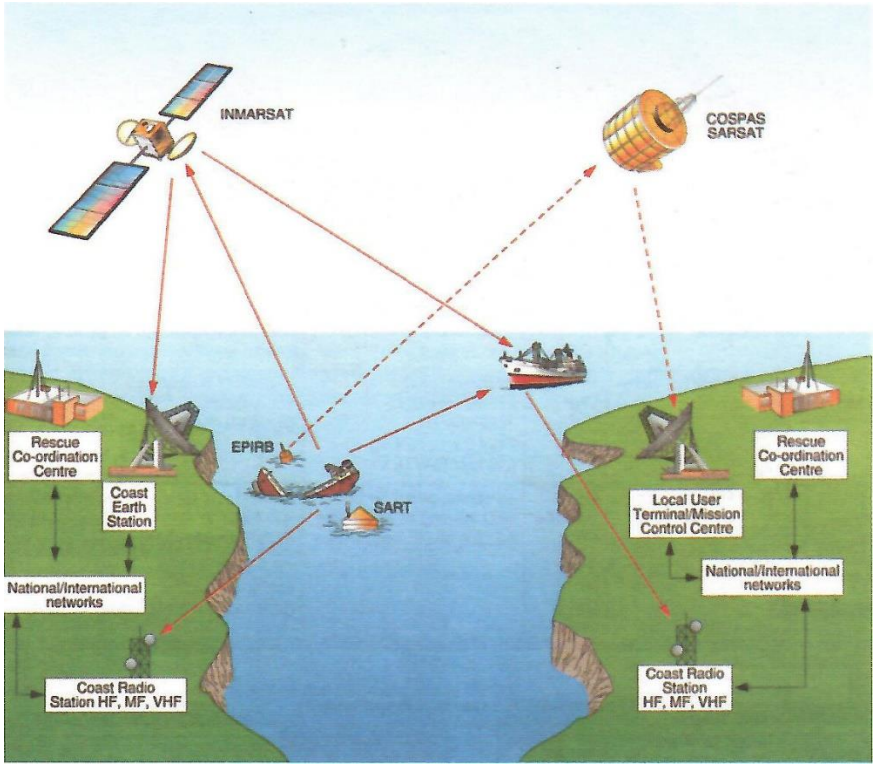
- nadawania, z użyciem dwóch niezależnych środków radiowych stosujących różne systemy radiowej, sygnałów alarmowych w relacji statek-ląd,
- obioru sygnałów alarmowych w relacji ląd-statek,
- nadawania i odbioru sygnałów alarmowych w relacji statek-statek,
- nadawania i odbioru informacji koordynacji poszukiwań i ratownictwa,
- nadawania i odbioru informacji na miejscu katastrofy,
- nadawania i odbioru sygnału lokalizacyjnego,
- nadawania i odbioru morskich informacji bezpieczeństwa,
- nadawania i odbioru informacji eksploatacyjnych i ogólnych za pośrednictwem lądowych ośrodków i sieci radiokomunikacyjnych,
- nadawania i odbioru informacji pomiędzy motkami statków.



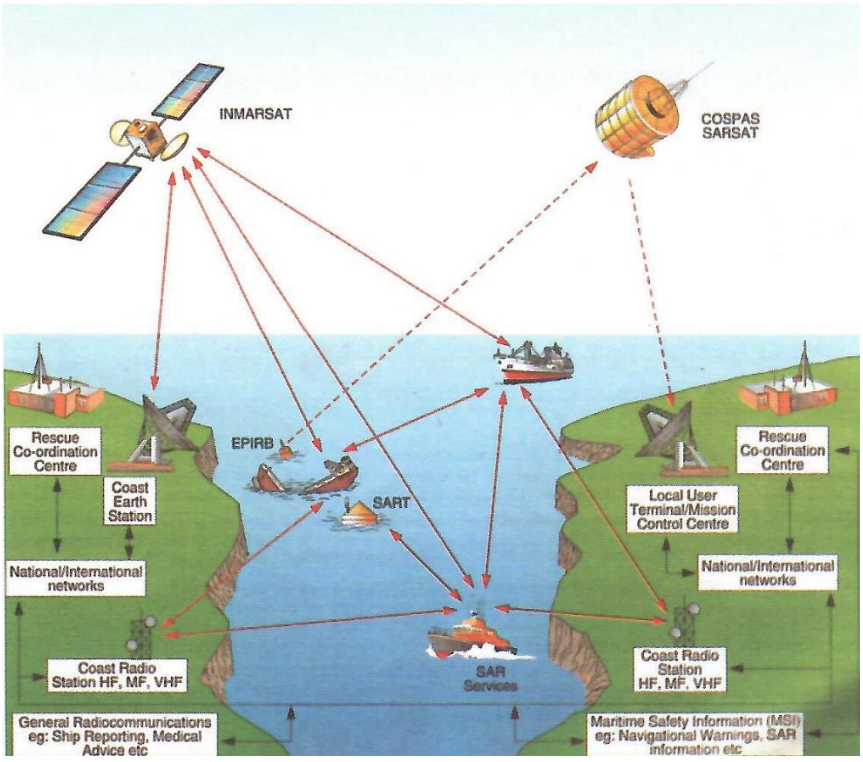
Rys. 1.1 Schemat struktury organizacyjnej systemu GMDSS. Źródło: [5]



Rys. 1.2 Wysłanie sygnałów alarmowych systemem GMDSS. Źródło: [5]



Rys. 1.3 Moment dotarcia informacji alarmowych do stacji koordynacji ratownictwa i statków w pobliżu miejsca katastrowy. Źródło: [5]



Rys. 1.4 Podjęcie akcji SAR z uwzględnieniem pełnej wymiany informacji na jaką pozwala GMDSS. Źródło: [5]

Ze względu na indywidualne właściwości i ograniczenia w zakresie zasięgu, a także rodzaju transmitowanych sygnałów, jakimi charakteryzują się podsystemy radiowe

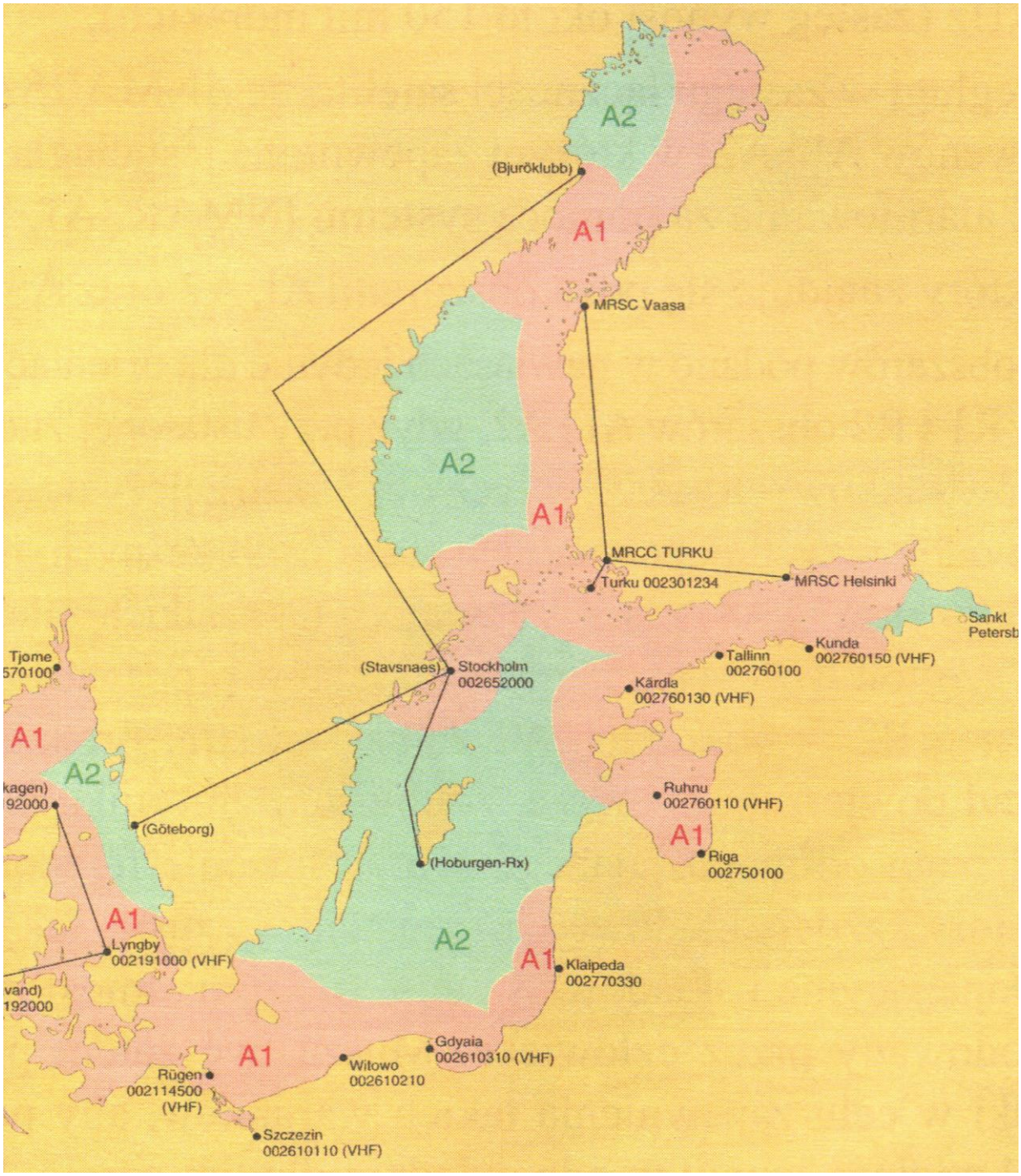
wchodzące w skład GMDSS, zostało uznane za konieczne wyposażenie statków w aparaturę do radiołączności zależnie od obszaru, w którym statek się porusza. Zmieniło to dotychczasowe podejście, jakim było uzależnienie zainstalowanych urządzeń od wyporności i przeznaczenia statku. Obszary zostały zdefiniowane w następujący sposób [4]:

A1 - obszar morski będący w zasięgu co najmniej jednej stacji nabrzeżnej zapewniającej komunikację w paśmie VHF, z którego możliwa jest realizacja ciągłej i skutecznej łączności alarmowej z użyciem cyfrowego selektywnego wywołania DSC, prowadzonej na kanale 70 (156,525 MHz) morskiego pasma VHF, obejmującego 156-174 MHz (zasięg działania jest określany indywidualnie dla każdej stacji i wynosi około 20-25 Mm),

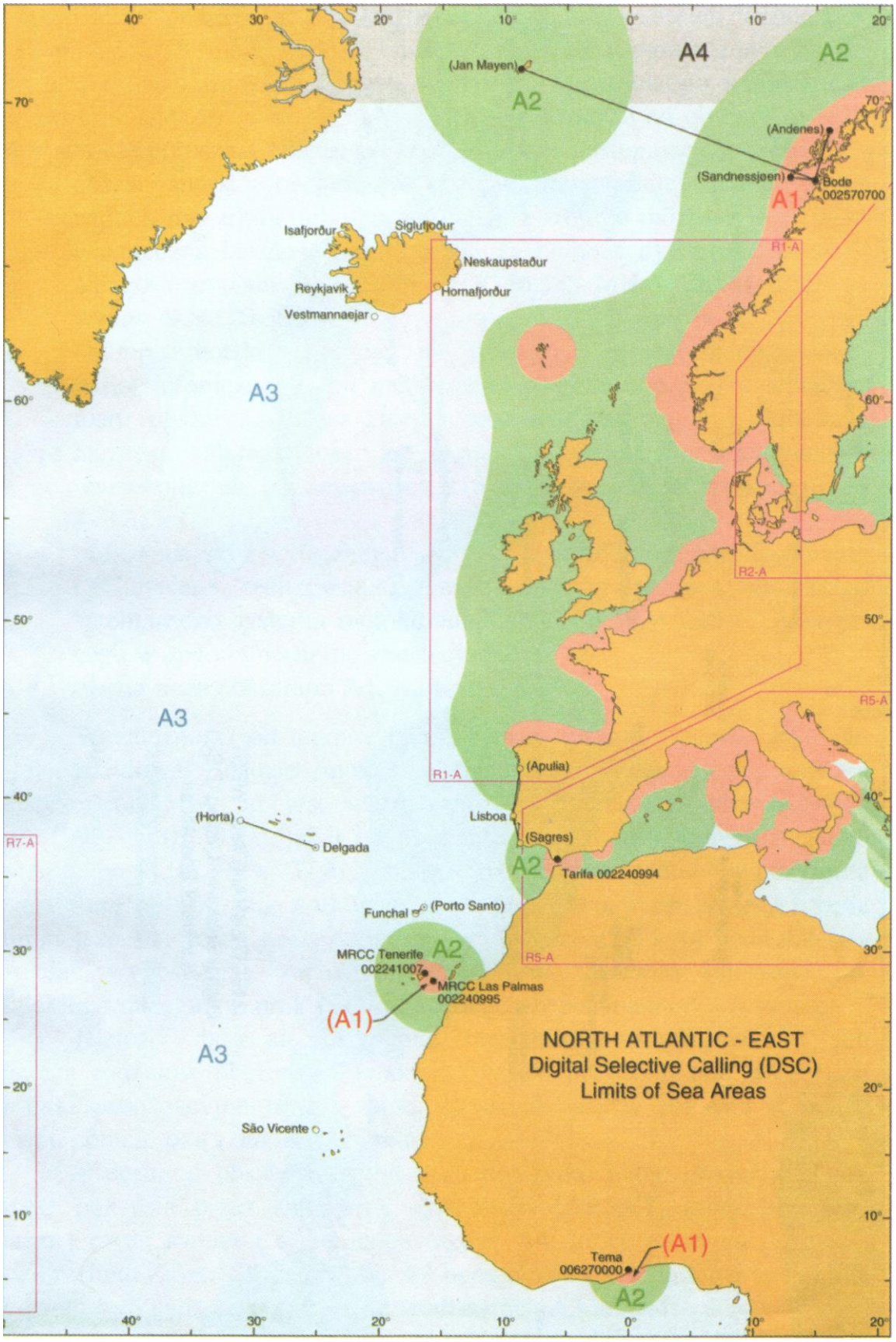
A2 - akwen morski, z wyłączeniem obszaru A1, będący w zasięgu minimum jednej radiotelefonicznej stacji nabrzeżnej średniofalowej, w którym możliwa jest realizacja ciągłej i skutecznej łączności alarmowej za pomocą cyfrowego selektywnego wywołania DSC na częstotliwości 2187,5 kHz (zasięg to około 150 Mm),

A3 - akwen, z wyłączeniem obszaru A1 i A2, w zasięgu łączności satelitarnej INMARSAT , w którym zapewniona jest ciągła i niezawodna łączność alarmowania za pomocą systemu INMARSAT,

A4 - obszar znajdujący się poza obszarami A1, A2 oraz A3.



Rys. 1.5 Mapa Bałtyku z naniesionymi obszarami alarmowania DSC. Źródło: [4]

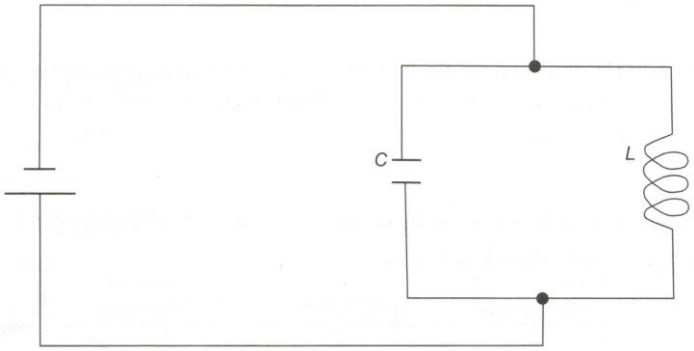


Rys. 1.6 Mapa północno-wschodniego oceanu Atlantyckiego z naniesionymi obszarami alarmowania DSC.

Źródło: [4]

1.2 Fale radiowe i kanały

Jak powszechnie wiadomo, najprostszym sposobem na wygenerowanie w obwodzie fal elektromagnetycznych jest stworzenie obwodu rezonansowego poprzez równoległe połączenie kondensatora o pojemności *C* z cewką o indukcyjności *L* (Rys. 1.7).



Rys. 1.7 Równoległy obwód rezonansowy. Źródło: [4]

Częstotliwość rezonansowa f_r powyższego obwodu wynosi:

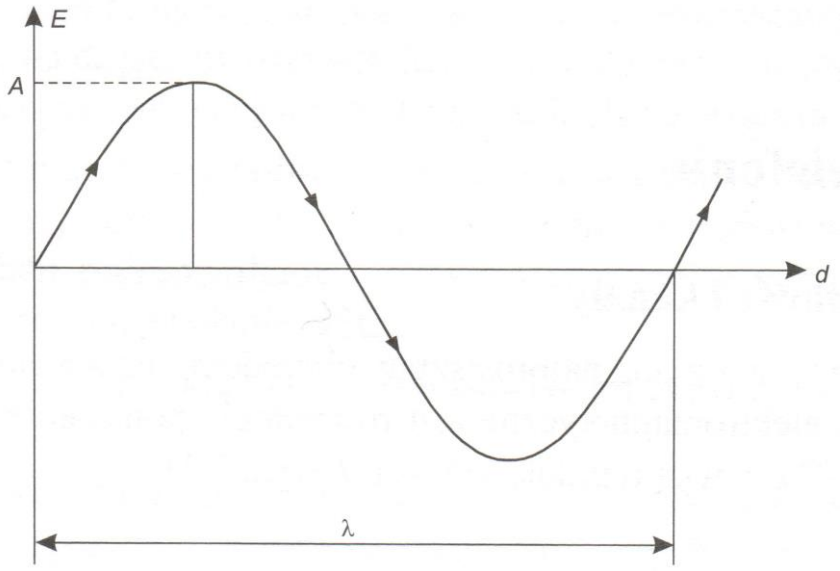
$$f_r = \frac{1}{2\pi\sqrt{LC}} \tag{1}$$

Głównym parametrem fali radiowej jest częstotliwość będąca bezpośrednio powiązana z długością fal. Jak wynika z rysunku 1.8, pełny cykl zmiany pola elektrostatycznego zawierającego się w czasie T nazywamy długością fali λ. Częstotliwością f określamy liczbę zmian pełnego cyklu elektrostatycznego, czyli liczbę fal wyemitowanych w jednostce czasu. Jednostką częstotliwości jest 1 Hz, co odpowiada jednemu pełnemu cyklowi elektrostatycznemu w ciągu sekundy. Wielokrotnościami tej jednostki są: 1 kHz = 10^3 Hz, 1 MHz = 10^6 Hz, oraz 1 Ghz = 10^9 Hz [4].

Częstotliwość fali elektromagnetycznej uzależniona jest od czasu *T* wg. zależności:

$$f = \frac{1}{T} \tag{2}$$

czyli wraz ze zmniejszeniem się okresu fali, rośnie jej częstotliwość.



Rys. 1.8 Przebieg zmian pola elektrycznego fali elektromagnetycznej: *E* – pole elektrostatyczne, λ – długość fali, *A* – amplituda, *d* – kierunek rozchodzenia się fali radiomagnetycznej. Źródło: [4]

Biorąc pod uwagę szybkość rozchodzenia się fali elektromagnetycznej, wyrażaną literą *c* (*c* = 300 000 km/s),

$$\lambda = c \cdot T \tag{3}$$

możemy podstawić wzór (1) do wzoru (2) otrzymując:

$$\lambda = \frac{c}{f} \tag{4}$$

Wyrażając λ we wzorze (4) w metrach, a częstotliwość w MHz, możemy zapisać:

$$\lambda[m] = \frac{300}{f[MHz]} \tag{5}$$

Podsumowując, długość fali o częstotliwości 150 MHz wynosi 2 m, a przeliczanie częstotliwości fali na jej długość lub odwrotnie nie nastręcza trudności.

Regulamin Radiokomunikacyjny jasno określa zakresy częstotliwości. Zgodny z nim podział został zamieszczony w poniższej tabeli [4].

Tabela 1.1 Pasma częstotliwości według Regulaminu Radiokomunikacyjnego.

Lp.	Długość fali	Częstotliwość	Pasmo częstotliwości	Przyjęty skrót angielski
1.	Fale myriametrowe	Very Low Frequencies	3 – 30 kHz	VLF
2.	Fale kilometrowe	Low Frequencies	30 – 300 kHz	LF
3.	Fale hektometrowe	Medium Frequencies	300 kHz – 3 MHz	MF
4.	Fale dekametrowe	High Frequencies	3 – 30 MHz	HF
5.	Fale metrowe	Very High Frequencies	30 – 300 MHz	VHF
6.	Fale decymetrowe	Ultra High Frequencies	300 MHz – 3GHz	UHF
7.	Fale centymetrowe	Super High Frequencies	3 – 30 GHz	SHF
8.	Fale milimetrowe	Extra High Frequencies	30 – 300 GHz	EHF

Źródło: [4]

1.3 Modulacje sygnału

W celu przeniesienia informacji pierwotnej, którą może być głos człowieka, teleks (telegrafia), facsimile lub transmisja danych, z użyciem fal radiowych z źródła informacji do obiektu przeznaczenia informacji musi zostać odpowiednio przetransformowana.

Proces do tego służący nazywamy modulacją. Polega on na przekształceniu niektórych parametrów fali radiowej, takich jak np. częstotliwość nośna (*carrier frequency*), w takt zmieniającej się informacji pierwotnej. Wyróżniamy trzy zasadnicze, analogowe, metody przeprowadzania tego procesu:

- modulacja amplitudy – AM (*Amplitude Modulation*),
- modulacja częstotliwości – FM (*Frequency Modulation*),
- modulacja fazy – PM (*Phase Modulation*).

Obecnie, w radiotelefonicznych systemach radiokomunikacji morskiej używa się modulacji SSB (pasma T i U) oraz FM w paśmie V [2].

1.3.1 Modulacja amplitudy (AM)

Modulacja amplitudy opiera się na zmianie amplitudy częstotliwości nośnej w takt zmian przebiegu modulującego stanowiącego przesyłaną przez nas informację. Najczęściej jest to sygnał akustyczny, czyli o częstotliwości zbliżonej mowie ludzkiej. W profesjonalnych urządzeniach i systemach do przenoszenia mowy ludzkiej stosuje się pasmo od 300 Hz do 3 kHz. Pozwala to uzyskać zrozumiały odbiór i dobrą jakość [4].

W czasie modulacji amplituda I_m prądu i zmienia się według poniższej zależności:

$$I_m = I_{m0} + m \sin \omega t \tag{6}$$

gdzie: I_m – chwilowa wartość amplitudy prądu modulowanego,

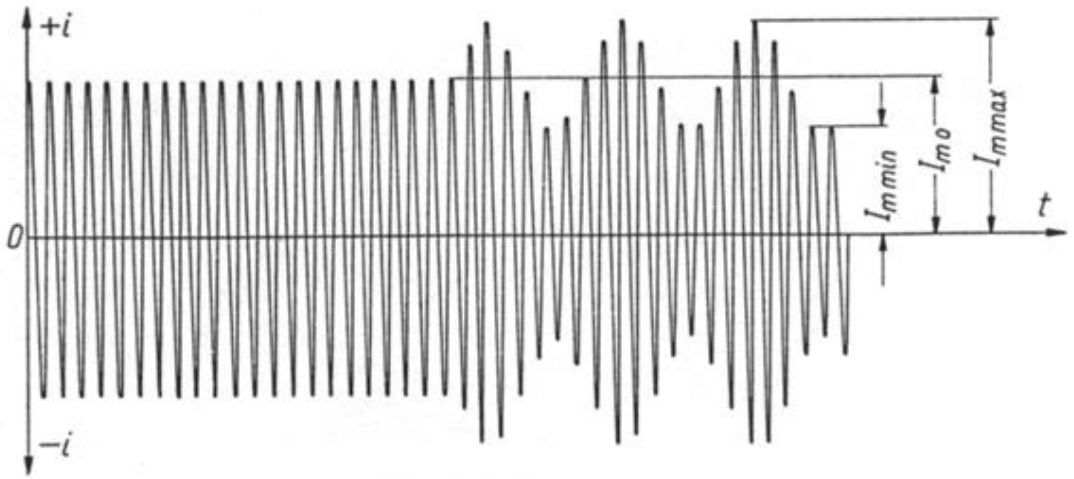
I_{m0} – amplituda fali nośnej,

m – głębokość modulacji,

ω – pulsacja częstotliwości akustycznej równa $2\pi f$ [6].

Parametrem opisującym tego typu modulację jest tzw. głębokość modulacji *m*, czyli stosunek największego przyrostu amplitudy fali nośnej $I_{m\,max}$ do amplitudy fali nośnej niemodulowanej I_{m0} [6]. Jej wartość nie powinna przekraczać jedności, ponieważ inaczej zaobserwujemy znaczny wzrost zniekształceń sygnału wyjściowego. Oczywiście istnieje możliwość określenia głębokości modulacji na podstawie rysunku lub obrazu oscyloskopowego przebiegu zmodulowanego [2][6]:

$$m = \frac{I_{m\,max} - \mathrm{I}_{m0}}{I_{m0}} = \frac{I_{m0} - \mathrm{I}_{m\,min}}{I_{m0}} = \frac{I_{m\,max} - \mathrm{I}_{m\,min}}{2I_{m0}} = \frac{I_{m\,max} - \mathrm{I}_{m\,min}}{I_{m\,max} + I_{m\,min}} \leq 1 \qquad (7)$$



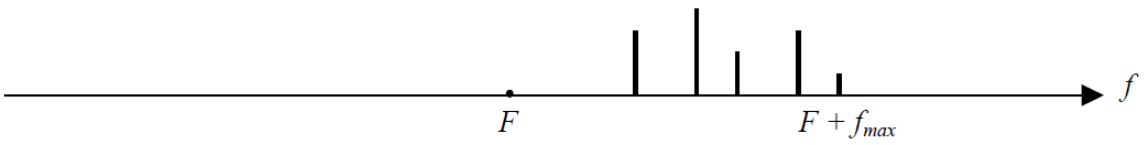
Rys. 1.9 Wykres amplitudy sygnału początkowo bez modulacji, a następnie zmodulowany amplitudowo tonem prostym sinusoidalnym o częstotliwości akustycznej f. Źródło: [6]

1.3.1.1 Modulacja jednowstęgowa SSB

Powstający w ten sposób sygnał zmodulowany składa się z trzech elementów: dolnej i górnej wstęgi bocznej zawierających pełną informację oraz składowej o częstotliwości nośnej niezwiązanej z przesyłaną informacją. Jeżeli pragniemy zaoszczędzić szerokość pasma oraz ilość energii potrzebnej do nadawania sygnału to możemy wyciąć jedną ze wstęg bocznych (rys. 1.10) otrzymując sygnał o nazwie SSB (*Single SideBand*). Brak jednego z elementów składowych zmodulowanego sygnału sprawia, że przesyłana informacja posiada gorszą jakość. Z tego powodu stosuje się ją głównie do przesyłania mowy ludzkiej, która ma większą odporność na zakłócenia przy odbiorze przez ludzki mózg. Jako, że maksymalną częstotliwością nadawaną w sygnale akustycznym jest 2800 Hz to szerokość pasma dla modulacji SSB wynosi [6]:

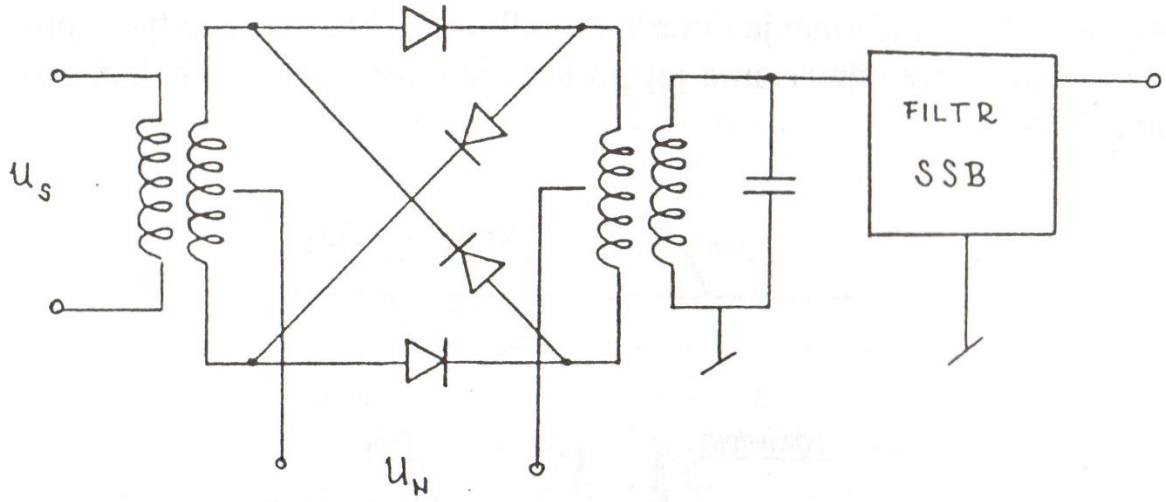
$$B = f_{max} = 2.8\,kHz \qquad (8)$$

Pozostaje jedynie rozwiązać problem techniczny odzyskania jej z tak „zubożonego” sygnału po stronie odbiorcy.



Rys. 1.10 Widmo sygnału zmodulowanego wstęgowo bez fali nośnej. Źródło: [6]

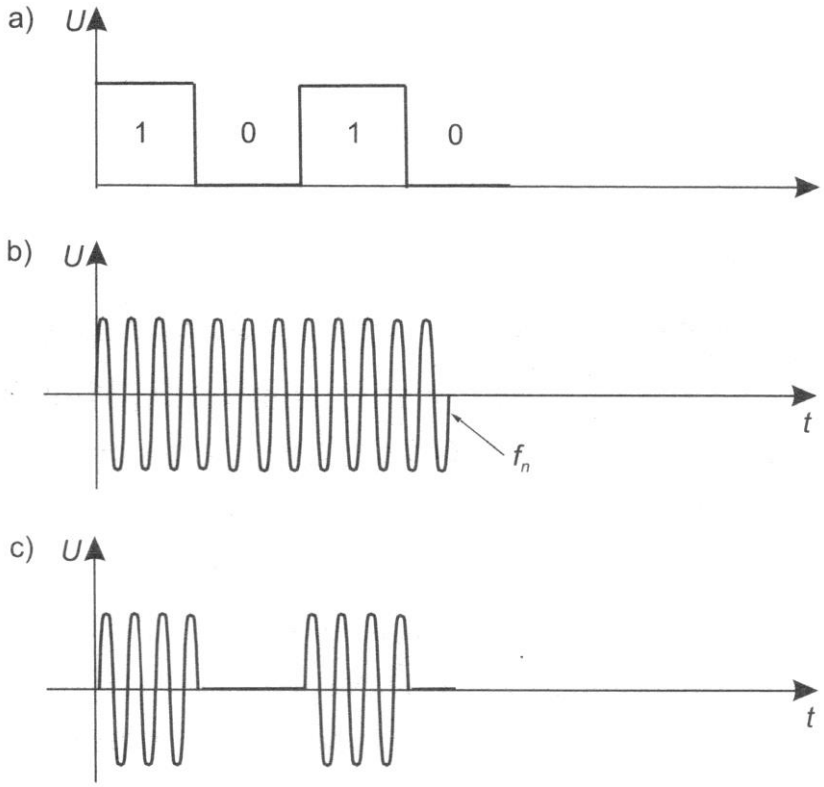
W nowoczesnych urządzeniach nadawczych, sygnał SSB wytwarza się na przykład za sprawą modulatora SSB zbudowanego z układu modulatora zrównoważonego i filtru (rys. 1.11). Zadaniem postawionym przed tym podwójnie zrównoważonym modulatorem diodowym jest wyeliminowanie z widma nośnej i większej ilości składowych niepożądanych. Innymi słowy filtr odfiltrowuje jedną ze wstęg bocznych. W celu uzyskania poziomu wymaganego do stłumienia zbędnej wstęgi bocznej w urządzeniach radiokomunikacyjnych stosuje się filtry kwarcowe [2].



Rys. 1.11 Modulator przebiegu SSB – metoda filtrowa. Źródło: [2]

1.3.1.2 Kluczowanie amplitudy ASK

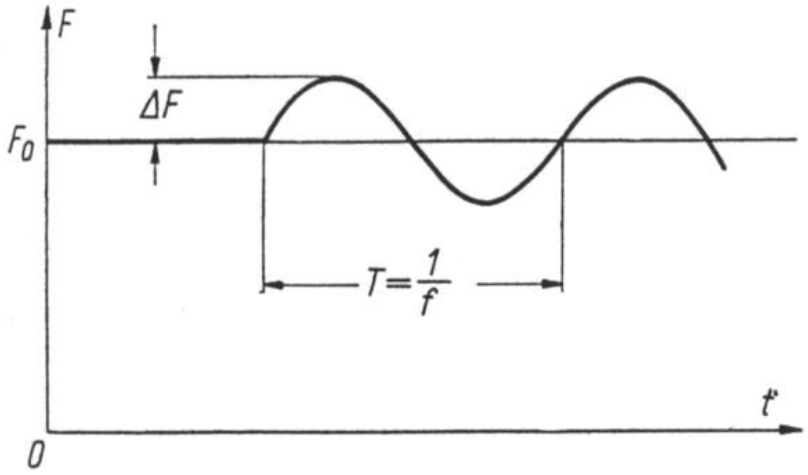
Jest jeszcze szczególny przypadek modulacji amplitudowej noszący nazwę kluczowanie amplitudy ASK (*Amplitude Shift Keying*). Różni się on tym od zwykłej modulacji AM, że sygnałem przenoszącym informację jest sygnał binarny – zerojedynkowy (wykres a na rys. 1.12). Jeżeli przebieg fali nośnej będzie taki jak na wykresie „b" rys. 1.12, to przebieg fali nośnej po modulacji będzie taki jak zobrazowano na wykresie „c" rys 1.12 [4].



Rys. 1.12 Kluczowanie amplitudy: a) przebieg binarny – zerojedynkowy, b) fala nośna f_n, c) przebieg wyjściowy. Źródło: [4]

1.3.2 Modulacja częstotliwości (FM)

Modulacja częstotliwości polega na dostosowaniu zmian w sygnale modulującym do zmian częstotliwości fali nośnej zachowując stałą wartość amplitudy. Uzależnia to częstotliwość fali nośnej od wartości sygnału modulującego. Sposób w jaki częstotliwość jest uzależniona od czasu przy modulacji tonem prostym przedstawiono na poniższym rysunku:



Rys. 1.13 Wykres częstotliwości sygnału początkowo bez modulacji, a następnie zmodulowany częstotliwościowo. Źródło: [6]

Podobnie jak w przypadku modulacji amplitudy, falę nośną o częstotliwości F_0 wytwarza stabilny generator wewnątrz nadajnika. Podczas modulacji częstotliwości wartość ta będzie się zmieniała proporcjonalnie do wartości sygnału modulującego. Dla dodatnich będą to zmiany $F_0 + \Delta F$, a dla ujemnych $F_0 - \Delta F$. Dla zachowania skuteczności modulacji FM, częstotliwość fali nośnej powinna być przynajmniej 1000 razy większa od częstotliwości sygnału modulującego. Praktyczne jej zastosowanie ogranicza się do fal ultrakrótkich VHF, powyżej 30 MHz [6].

Maksymalna wielkość zmiany częstotliwości ΔF nazywana jest dewiacją częstotliwości. Decyduje ona o szerokości pasma zajmowanego przez sygnał zmodulowany i jest wielkością charakterystyczną dla systemu komunikacyjnego wykorzystującego ten rodzaj modulacji. Szerokość pasma zajmowanego przez sygnał FM można wyrazić następująco [2]:

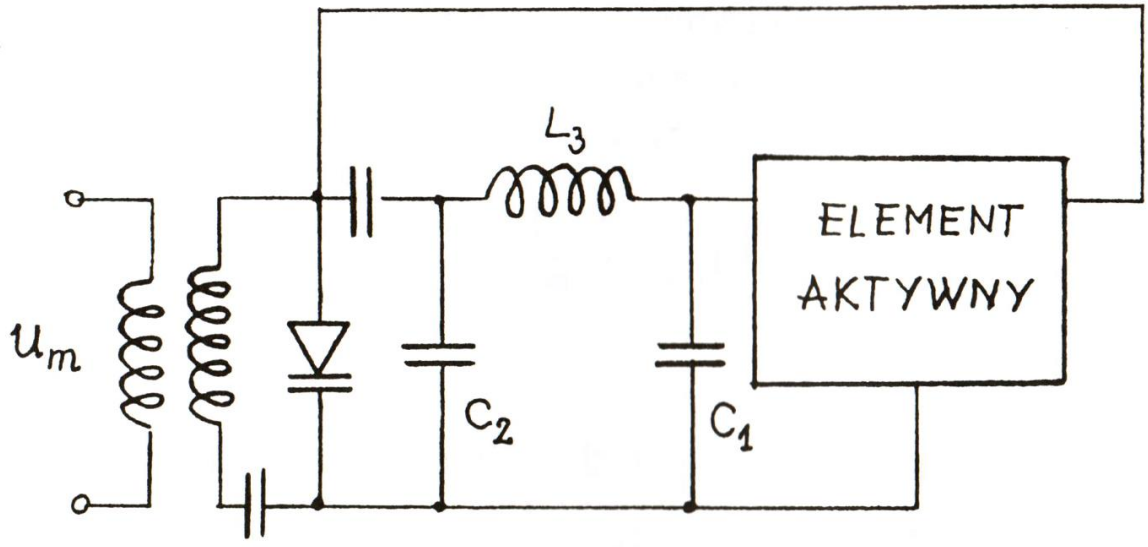
$$B_{FM} = 2(\Delta F + f_m) \tag{9}$$

gdzie:

ΔF – dewiacja częstotliwości,

f_m – maksymalna częstotliwość sygnału modulującego.

Modulacja FM o właściwych parametrach jest nieco bardziej skomplikowana od modulacji amplitudowej. Najprościej można to zrobić dołączając do obwodu generatora LC reaktancję zmienną o wartości uzależnionej od przyłożonego napięcia. Sterowanie tą reaktancją (np. wykorzystując diodę pojemnościową), a tym samym przebiegiem modulującym sprawi, że generowana częstotliwość ulegnie zmianie (rys. 1.14).



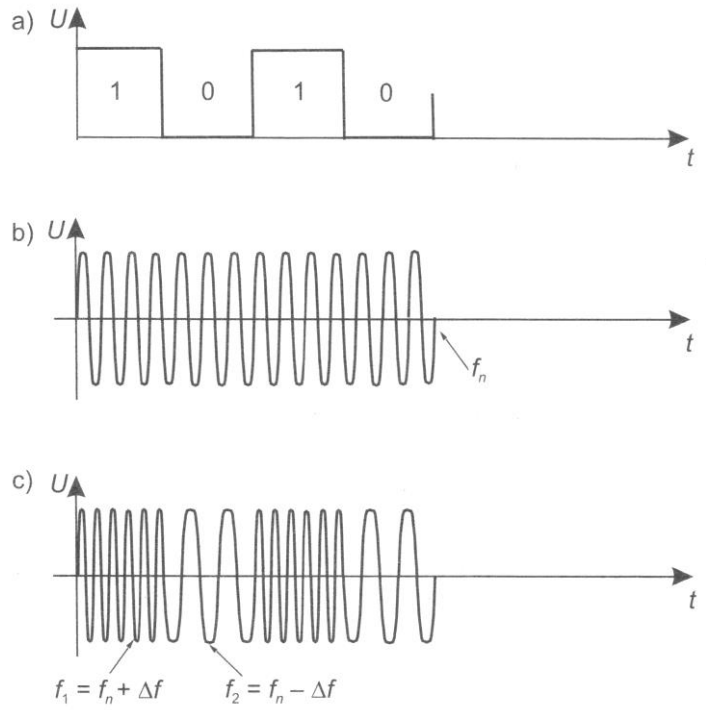
Rys. 1.14 Modulator FM z diodą pojemnościową w układzie generatora Colpittsa. Źródło: [2]

Rozwiązanie to posiada jednak wady wykluczające je z profesjonalnego zastosowania. Pierwszą z nich jest pogorszenie stabilności częstotliwości środkowej generatora wynikłe z nieliniowości charakterystyki diody pojemnościowej, przez co przebieg zmodulowany zawiera znaczny poziom zniekształceń. Dodatkowo w układzie powstaje pasożytnicza modulacja amplitudy [2].

Problemy te rozwiązano stosując w profesjonalnym wyposażeniu radiokomunikacyjnym rozbudowane układy, w których modulacja częstotliwości otrzymywana jest z wykorzystaniem modulacji AM i PM oraz wielokrotnego powielania sygnału [2].

1.3.2.1 Kluczowanie częstotliwości FSK

Podobnie jak w przypadku modulacji amplitudowej, kluczowanie częstotliwości FSK (*Frequency Shift Keying*) jest szczególnym przypadkiem modulacji częstotliwości FM znajdujący praktyczne zastosowanie w przenoszeniu sygnałów cyfrowych. Ze względu na prostokątny charakter przebiegu modulującego (rys. 1.15, wykres „a"), a amplituda i częstotliwość przebiegu nośnego nie ulega zmianie (rys. 1.15, wykres „b") to wynikiem tego typu kluczowania jest przebieg przedstawiony na wykresie „c" rysunku 1.15. Jak można zauważyć, „jedynkom" sygnału binarnego odpowiadają powiększenia częstotliwości wyjściowej sygnału o Δf, a dla „zer" binarnych zmniejszenia o Δf [4].



Rys. 1.15 Kluczowanie częstotliwości FSK a) przebieg binarny – zerojedynkowy, b) przebieg częstotliwości nośnej, c) przebieg wyjściowy. Źródło: [4]

1.3.3 Modulacja fazy (PM)

Modulacja fazy nazywana też kątową, jest procesem, którego głównym założeniem jest uzależnienie kąta fazowego fali nośnej od wartości sygnału modulującego. Przy modulacji fazy fali nośnej prostym tonem sinusoidalnym o częstotliwości f, kąt zmienia się okresowo zgodnie z zależnością [6]:

$$\varphi = \varphi_0 + \Delta\varphi \sin 2\pi f \tag{10}$$

gdzie:

$\Delta\varphi$ – maksymalna zmiana kąta fazowego, zwana też dwiacją fazy,

φ_0 – kąt fazowy fali nośnej.

Zatem, zmodulowany fazowo prąd, będzie wyrażony wzorem [6]:

$$i = I_{m0} + \sin(2\pi F_0 t + \Delta\varphi \sin 2\pi \mathrm{ft} + \varphi_0) \tag{11}$$

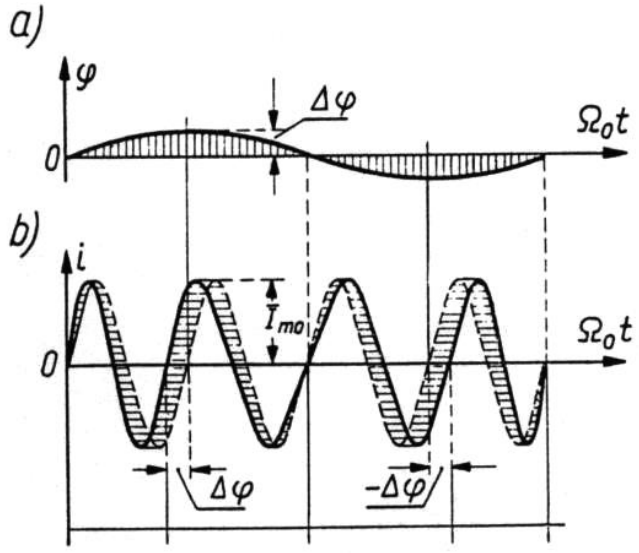
gdzie:

I_{m0} – amplituda fali nośnej,

F_0 – częstotliwość fali nośnej,

$\Delta\varphi$ – dewiacja fazy,

φ_0 – kąt fazowy fali nośnej.



Rys. 1.16 Idea modulacji fazy. a) zmiany kąta fazowego
b) przebieg prądu zmodulowanego (linia ciągła) i przed modulacją (linia przerywana)
Źródło: [6]

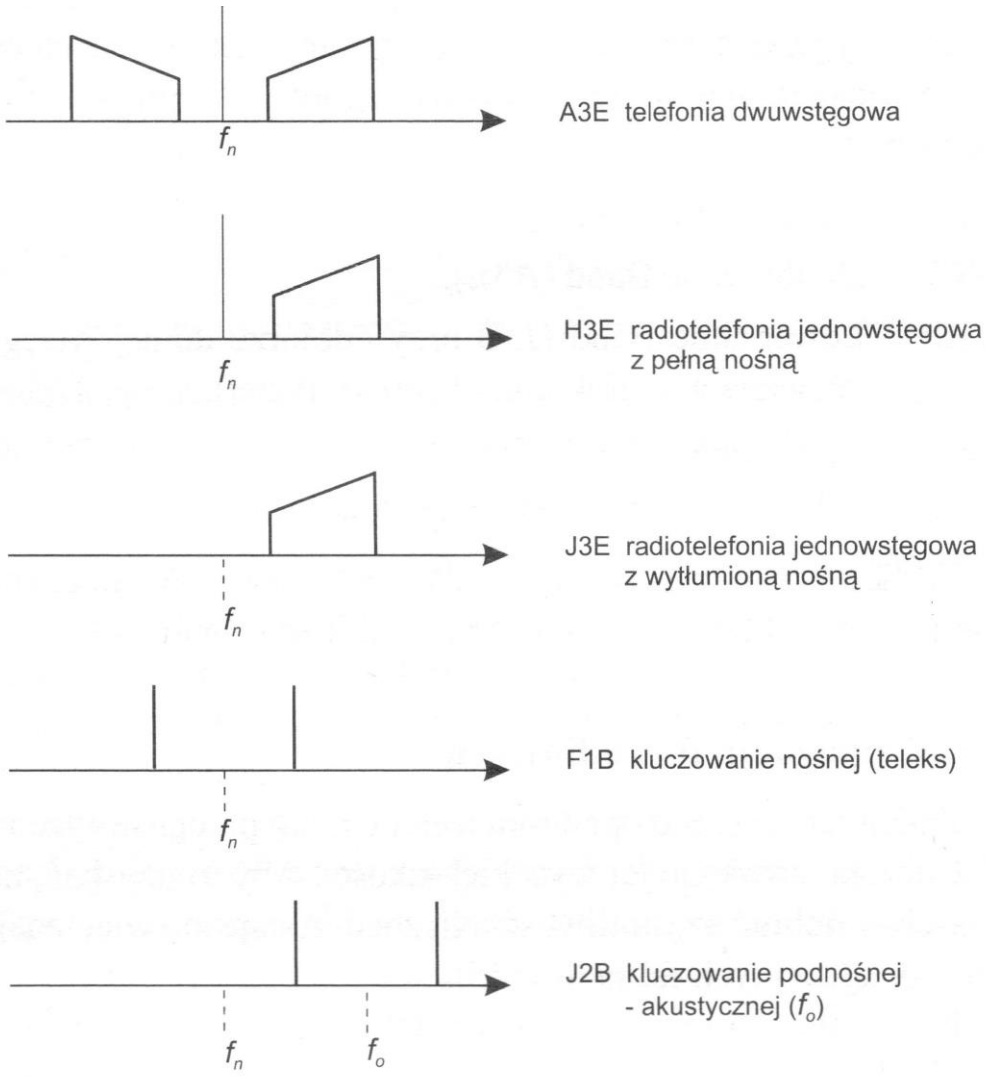
Jak wynika z powyższego rysunku, modulacja fazowa zmienia nie tylko kąt fazowy, ale również częstotliwość. Maksymalna dewiacja częstotliwości ΔF wynosi $\Delta\varphi f$. Można zauważyć, że przy odpowiednim doborze parametrów, modulacje AM i FM mogą otrzymać identyczne wyniki. Jednakże, dotyczy to jedynie sygnałów sinusoidalnych. Fakt ten

wykorzystano przy budowie modulatorów FM w profesjonalnym sprzęcie radiooperatorskim. Niestety, w przypadku modulacji innymi sygnałami (trójkątny, prostokątny itp.) podobna zależność nie zachodzi [6].

Modulacja fazowa posiada dwie podstawowe zalety nad modulacją amplitudy. Pierwszą z nich jest wierniejsze przesyłanie sygnałów ze względu na przenoszenie szerszego pasma częstotliwości w modulacji kątowej. Oznacza to przede wszystkim lepszą jakość przesyłanych sygnałów akustycznych. Drugą zaletą jest znacznie większa odporność na zakłócenia. Wynika to z faktu, że większość zakłóceń w transmisji ma charakter amplitudowy, czyli nakładają się one na amplitudę sygnału. Skoro przy modulacji amplitudy wszystkie informacje są zapisane w zmianach amplitudy to każde takie zakłócenie wprowadza niepożądaną zmianę przesyłanych danych. Te szkodliwe zmiany nie mają wpływu na wartość sygnału przy modulacji fazowej, ponieważ komplet informacji zawiera się w zmianach częstotliwości i mogą zostać usunięte z wykorzystaniem odpowiednich układów w odbiorniku [6].

1.4 Emisje

Regulamin Radiokomunikacyjny definiuje emisję jako celowe wytwarzanie, poprzez radiową stację nadawczą, energii w postaci fal radiowych. Charakteryzuje je między innymi szerokość pasma wymagana do transmisji informacji z wymaganą jakością i prędkością oraz ich klasyfikacją.



Rys. 1.17 Przykłady widm emisji radiowych. Źródło: [6]

Klasyfikacja emisji polega na podaniu jej charakterystycznych cech (rys. 1.17). W zakresie podstawowym przyjmuje format trzyznakowego ciągu alfanumerycznego. Znaczenie tych symboli wygląda następująco [4]:

1. Pierwszy symbol jest literowy i oznacza sposób zmodulowania głównej nośnej:
    a. emisja nośnej niemodulowanej N
    b. emisja, w której główna nośna jest modulowana amplitudowo:
        - dwuwstęgowa A
        - jednowstęgowa, pełna nośna H
        - jednowstęgowa, nośna zredukowana R
        - jednowstęgowa, nośna stłumiona J
        - niezależne wstęgi boczne B
        - szczątkowe wstęgi boczne C
    c. emisja, w której główna nośna jest modulowana kątowo:
        - modulacja częstotliwości F
        - modulacja fazy G
2. Drugi symbol jest cyfrowy i oznacza charakter sygnału modulującego główną nośną:
    a. brak sygnału modulującego 0
    b. pojedynczy kanał zawierający informację kwantową lub cyfrową, bez użycia modulującej podnośnej (z wyłączeniem zwielokrotnienia z podziałem czasowym) 1
    c. pojedynczy kanał zawierający informację cyfrową lub kwantową, z zastosowaniem modulującej podnośnej (tak jak powyżej) 2
    d. pojedynczy kanał zawierający informację analogową 3
3. Trzeci symbol jest literowy i oznacza rodzaj nadanej informacji:
    a. nadawanie pozbawione informacji N
    b. telegrafia do odbioru słuchowego (Morse'a) A
    c. telegrafia do obioru automatycznego, np. NBDP B
    d. faksymilografia C
    e. transmisja danych, telemetria D
    f. telefonia (wraz z rozgłaszaniem dźwiękowym) E
    g. telewizja (obraz) F

Przykładowe oznaczenia dla wybranych emisji [4]:

- A1A – telegrafia ręczna, tonowa, kodem Morse'a,
- H2A – telegrafia ręczna, tonowa, kodem Morse'a, sygnał jednowstęgowy wraz z nośną,
- A3E – dwuwstęgowa modulacja amplitudy z wykorzystaniem sygnału fonicznego,
- H3E – jednowstęgowa modulacja amplituda z pełną nośną,
- R3E – jednowstęgowa modulacja amplitudy z nośną zredukowaną,
- J3E – jednowstęgowa modulacja amplitudy z nośną szczątkową,
- F3C – symilografia (przesyłanie statycznych obrazów),
- F1B – telegrafia do odbioru automatycznego NBDP będąca odpowiednikiem FSK (*Frequency Shift Keying*), czyli kluczowania przesuwem częstotliwości nośnej,
- J2B – kluczowanie częstotliwości podnośnej (np. $1700\ Hz \pm 85\ Hz$).

Warto również zaznaczyć, że w praktyce bardzo często spotyka się skrótowe nazwy emisji używane w potocznym języku technicznym. Poniżej pokazano i rozwinięto część z nich, w nawiasach zostały umieszczone oznaczenia emisji zgodne z Regulaminem Radiokomunikacyjnym [4]:

- CW (A1A) – Carrier Wave,
- DSB (A3E) – Double Side Band,
- USB (USB) – Upper Side Band,
- TLX (F1B) – Telex, czyli łączność dalekopisowa,
- LSB (J3B przy odbiorze dolnej wstęgi bocznej) – Lower Side Band, nieużywana w radiokomunikacji morskiej.

1.5 Propagacja fal radiowych

Fale hektometrowe (pasmo MF) rozchodzą się głównie jako fale przyciemne, uginając się wzdłuż powierzchni ziemi. Oznacza to, że w obliczeniach efektywnego zasięgu, oprócz częstotliwości, należy uwzględnić parametry fizyczne gruntu takie jak przewodność i stała dielektryczna. W przypadku fal rozchodzących się nad powierzchnią morza, za skuteczny zasięg można przyjąć około 150 mil morskich dla łączności radiotelefonicznej oraz 300 mil morskich dla DSC lub teleksu. W przypadku rejonów tropikalnych zasięgi są znacznie mniejsze. Wynika to z wysokiego poziomu zakłóceń atmosferycznych w rejonach równikowych [3].

W przypadku pasma HF, fale dekametrowe, rozchodzą się przede wszystkim jako jonosferyczne. Różnią się od fal przyziemnych tym, że odbijają się od warstwy jonosfery. Wykorzystując wielokrotne odbijanie się ich od warstwy jonosfery i powierzchni Ziemi, można osiągnąć zasięg globalny. Ze względu na zależność jonizacji górnych warstw jonosfery od aktywności słońca, właściwości odbijające zależne są od pory dnia, roku oraz cyklu zmian plam słonecznych [3].

Skuteczność komunikacji odbywająca się na wyższych częstotliwościach, tj. powyżej 8 MHz, rośnie wraz ze stopniem jonizacji. Na niższych natomiast skuteczność wzrasta wraz ze spadkiem stopnia jonizacji. Zatem można przyjąć, że komunikacja z wykorzystaniem częstotliwości powyżej 8 MHz jest skuteczniejsza w dzień (większy stopień jonizacji), a na niższych częstotliwościach (poniżej 8 MHz) w nocy kiedy stopień jonizacji jest niższy. Graniczna częstotliwość, 8 MHz, może być wykorzystywana niezależnie od pory dnia i jonizacji atmosfery [3].

Ze względu na większą ilość energii potrzebnej do wygenerowania fali jonosferycznej, łączność na w paśmie HF prowadzi się głównie w rejonach najdalej położonych od brzegu (A3 i A4). W rejonie A4, komunikacja krótkofalowa, jest jedynym sposobem łączności okrętu z brzegiem.

2. RADIOSTACJA MF/HF Z PRZYSTAWKĄ DSC

2.1 Radiostacja MF/HF - SAILOR RE 2100

Konstrukcja radiostacji MF/HF pozwala jej na pracę w dwóch zakresach częstotliwości. Pierwszy z nich, zakres fal pośrednich MF (*Medium Frequencies*) zawiera się w przedziale od 1606,5 kHz do 4000 kHz, czyli hektometrowy rząd długości fali. Drugi natomiast to zakres fal krótkich HF (*High Frequencies*) zawierający się w przedziale od 4000 kHz (4 MHz) do 27500 kHz (27,5 MHz), a więc dekametrowy rząd długości fali [3].

W radiostacji MF/HF zawiera się zarówno nadajnik jak i odbiornik. Odbiornik umożliwia nasłuch na częstotliwościach z zakresu 150 – 30 000 kHz. Nadajnik natomiast zezwala nadawać jedynie na częstotliwościach przeznaczonych dla służby morskiej. Jakakolwiek próba nadawania na innych częstotliwościach niż zarezerwowane spotyka się z komunikatem „Error 11" sygnalizowanym przez radiostację [3]. Oznacza on niedozwoloną częstotliwość.

Rys. 2.1 Płyta czołowa radiostacji MF/HF SAILOR RE 2100. Źródło: [3]

Radiostację włącza się poprzez przekręcenie w prawo pokrętła oznaczonego jako „Vol." Dalszy obrót zgodny z kierunkiem obrotu zegara powoduje zwiększenie głośności głośnika. Urządzenie jest w stanie automatycznie zwiększać wzmocnienie wzmacniacza wejściowego w momencie obniżenia poziomu odbieranego sygnału oraz zmniejszać go w momencie wzrostu odbieranego sygnału. Aby włączyć tą funkcję należy wcisnąć przycisk „AGC" (automatic gain control). O działającej funkcji informuje dioda pod wyświetlaczem opisana jako „AGC". Należy pamiętać, że w skrajnych przypadkach, przy bardzo słabym sygnale, funkcja automatycznej regulacji wzmocnienia może nie działać. Wówczas należy wyłączyć „AGC" i wyregulować wzmocnienie ręcznie, z wykorzystaniem pokrętła „RF",

tak aby usłyszeć nadawcę. Blokada szumów znajduje się pod przyciskiem „SQ”, a dioda informująca o włączaniu funkcji obok diody AGC. Poziom blokady dobiera automatycznie komputer radiostacji. Należy pamiętać, iż blokady szumów nie stosuje się przy łączności alarmowej.

Jako że do radiostacji mogą zostać podłączone nadajniki o różnej mocy, SAILOR RE 2100 umożliwia ustawienie mocy nadawania na pięciu różnych poziomach: $\frac{1}{16}$, $\frac{1}{8}$, $\frac{1}{4}$, $\frac{1}{2}$, pełna. W przypadku radiostacji okrętowych, w paśmie MF, istnieje możliwość zastosowania nadajników o mocy maksymalnej nieprzekraczającej 400W, a w paśmie HF 1,5 kW. Zmiany poziomu mocy można dokonać wykorzystując przyciski „TX”, a następnie „POWER”. Po wciśnięciu przycisku „TX”, operator ma kilka sekund na regulację poziomu mocy wielokrotnie naciskając przycisk „POWER”. Każde wciśnięcie tego przycisku powinno zwiększyć lub zmniejszyć moc o połowę. Poziom mocy sygnalizuje pojedyncza lub podwójna kreska na wyświetlaczu – im wyżej nad trójkątem mocy znajduje się kreska, tym większa moc.

Radiostacja MF/HF SAILOR obsługuje cztery rodzaje emisji:

- J3E – emisja jednowstęgowa bez fali nośnej, stosowana przy łączności fonicznej,
- H3E – emisja jednowstęgowa z falą nośną, aktualnie bez zastosowania, poprzednio używana przy łączności alarmowej,
- R3E – emisja jednowstęgowa ze zredukowaną falą nośną, stosowana przy współpracy z radiostacjami starszego typu, bez syntezy częstotliwości (w niektórych radiostacjach wymieniona na CW – emisję z falą ciągłą stosowaną do pracy na kluczu alfabetem Morse’a),
- TLX – emisja teleksowa, stosowana do pracy na teleksie.

2.2 Przystawka DSC

Cyfrowe selektywne wywołanie DSC (*Digital Selective Calling*) jest jednym z elementów wykorzystywanych w GMDSS. Jego głównymi funkcjami jest nadawanie cyfrowych, zunifikowanych sygnałów alarmowych w przypadku niebezpieczeństwa oraz automatyczne, inicjowanie, a potem ustanawianie połączeń radiowych w relacji: statek-ląd, ląd-statek, statek-statek [4].

System został zaprojektowany z myślą o pracy w paśmie pośredniofalowym (2 MHz), krótkofalowym (4 MHz, 6 MHz, 8 MHz, 12 MHz i 16 MHz) oraz w morskim zakresie pasma VHF (156-174 MHz) [4].

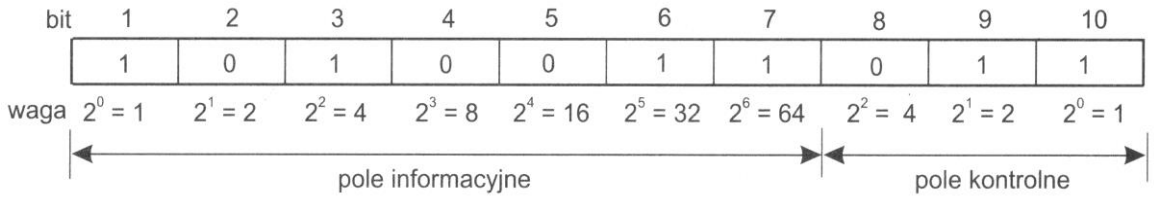
DSC to wielodostępowy system przesyłania informacji cyfrowych o częściowo kontrolowanym dostępie do kanału. Reguła, na podstawie której ustalany jest dostęp do kanału, polega na śledzeniu fali nośnej i wymuszeniu przesyłania pakietu (bloku) informacji, w momencie, gdy kanał jest wolny [4].

Blokom informacji przypisuje się jeden z czterech priorytetów, zależnie od typu oraz stopnia pilności:

- alarmowanie (*distress*),
- pilność (*urgent*),
- bezpieczeństwo (*safety*),
- wywołanie rutynowe (*routine*).

Pakiety, po uprzednim wyborze operatora, mogą zostać skierowane do wszystkich stacji, do pojedynczej stacji lub grupy stacji przy zachowaniu kodu identyfikującego MMSI (*Maritime Mobile Service Identity*).

Na rysunku poniżej przedstawiono postać ogólną 10-bitowego ciągu kodowego będącego podstawą sekwencji wywoławczej:



Rys. 2.2 Ciąg kodowy stosowany w DSC o długości N = 10. Źródło: [4]

Polu informacyjnemu, na które przeznaczono 7 bitów (przy czym waga pierwszego bitu wynosi 1, a siódmego 64), możemy przypisywać rozwinięcia binarne symboli z tabeli 2.1. Pole kontrolne służy natomiast ochronie przesyłania informacji przed błędami w kanale. Polega to na uzupełnieniu go 3-pozycyjnym ciągiem kontrolnym wyrażającym liczbę zer

występujących w polu informacyjnym. Jak można zauważyć, liczba bitów pola kontrolnego k odpowiada logarytmowi o podstawie 2 z liczby bitów pola informacyjnego n powiększonej o 1 [4].

$$k = [log_2(n+1)] \quad \text{(N)}$$

Tak więc, połączenie ciągu informacji z ciągiem kontrolnym tworzy ciąg kodu Bergera, który z resztą idealnie sprawdza się dla kanałów binarnych całkowicie asymetrycznych jakie wykorzystuje DSC.

Tabela 2.1 Kod 10-elementowy z 3-bitową kontrolą błędów.

Symbol nr	Emitowany sygnał i pozycja bitu 1 2 3 4 5 6 7 8 9 10	Symbol nr	Emitowany sygnał i pozycja bitu 1 2 3 4 5 6 7 8 9 10	Symbol nr	Emitowany sygnał i pozycja bitu 1 2 3 4 5 6 7 8 9 10
00	B B B B B B B Y Y Y	43	Y Y B Y B Y B B Y Y	86	B Y Y B Y B Y B Y Y
01	Y B B B B B B Y Y B	44	B B Y Y B Y B Y B B	87	Y Y Y B Y B Y B Y B
02	B Y B B B B B Y Y B	45	Y B Y Y B Y B B Y Y	88	B B B Y Y B Y Y B B
03	Y Y B B B B B Y B Y	46	B Y Y Y B Y B B Y Y	89	Y B B Y Y B Y B Y Y
04	B B Y B B B B Y Y B	47	Y Y Y Y B Y B B Y B	90	B Y B Y Y B Y B Y Y
05	Y B Y B B B B Y B Y	48	B B B B Y Y B Y B Y	91	Y Y B Y Y B Y B Y B
06	B Y Y B B B B Y B Y	49	Y B B B Y Y B Y B B	92	B B Y Y Y B Y B Y Y
07	Y Y Y B B B B Y B B	50	B Y B B Y Y B Y B B	93	Y B Y Y Y B Y B Y B
08	B B B Y B B B Y Y B	51	Y Y B B Y Y B B Y Y	94	B Y Y Y Y B Y B Y B
09	Y B B Y B B B Y B Y	52	B B Y B Y Y B Y B B	95	Y Y Y Y Y B Y B B Y
10	B Y B Y B B B Y B Y	53	Y B Y B Y Y B B Y Y	96	B B B B B Y Y Y B Y
11	Y Y B Y B B B Y B B	54	B Y Y B Y Y B B Y Y	97	Y B B B B Y Y Y B B
12	B B Y Y B B B Y B Y	55	Y Y Y B Y Y B B Y B	98	B Y B B B Y Y Y B B
13	Y B Y Y B B B Y B B	56	B B B Y Y Y B Y B B	99	Y Y B B B Y Y B Y Y
14	B Y Y Y B B B Y B B	57	Y B B Y Y Y B B Y Y	100	B B Y B B Y Y Y B B
15	Y Y Y Y B B B B Y Y	58	B Y B Y Y Y B B Y Y	101	Y B Y B B Y Y B Y Y
16	B B B B Y B B Y Y B	59	Y Y B Y Y Y B B Y B	102	B Y Y B B Y Y B Y Y
17	Y B B B Y B B Y B Y	60	B B Y Y Y Y B B Y Y	103	Y Y Y B B Y Y B Y B
18	B Y B B Y B B Y B Y	61	Y B Y Y Y Y B B Y B	104	B B B Y B Y Y Y B B
19	Y Y B B Y B B Y B B	62	B Y Y Y Y Y B B Y B	105	Y B B Y B Y Y B Y Y
20	B B Y B Y B B Y B Y	63	Y Y Y Y Y Y B B B Y	106	B Y B Y B Y Y B Y Y
21	Y B Y B Y B B Y B B	64	B B B B B B Y Y Y B	107	Y Y B Y B Y Y B Y B
22	B Y Y B Y B B Y B B	65	Y B B B B B Y Y B Y	108	B B Y Y B Y Y B Y Y
23	Y Y Y B Y B B B Y Y	66	B Y B B B B Y Y B Y	109	Y B Y Y B Y Y B Y B
24	B B B Y Y B B Y B Y	67	Y Y B B B B Y Y B B	110	B Y Y Y B Y Y B Y B
25	Y B B Y Y B B Y B B	68	B B Y B B B Y Y B Y	111	Y Y Y Y B Y Y B B Y
26	B Y B Y Y B B Y B B	69	Y B Y B B B Y Y B B	112	B B B B Y Y Y Y B B
27	Y Y B Y Y B B B Y Y	70	B Y Y B B B Y Y B B	113	Y B B B Y Y Y B Y Y
28	B B Y Y Y B B Y B B	71	Y Y Y B B B Y B Y Y	114	B Y B B Y Y Y B Y Y
29	Y B Y Y Y B B B Y Y	72	B B B Y B B Y Y B Y	115	Y Y B B Y Y Y B Y B
30	B Y Y Y Y B B B Y Y	73	Y B B Y B B Y Y B B	116	B B Y B Y Y Y B Y Y
31	Y Y Y Y Y B B B Y B	74	B Y B Y B B Y Y B B	117	Y B Y B Y Y Y B Y B
32	B B B B B Y B Y Y B	75	Y Y B Y B B Y B Y Y	118	B Y Y B Y Y Y B Y B
33	Y B B B B Y B Y B Y	76	B B Y Y B B Y Y B B	119	Y Y Y B Y Y Y B B Y
34	B Y B B B Y B Y B Y	77	Y B Y Y B B Y B Y Y	120	B B B Y Y Y Y B Y Y
35	Y Y B B B Y B Y B B	78	B Y Y Y B B Y B Y Y	121	Y B B Y Y Y Y B Y B
36	B B Y B B Y B Y B Y	79	Y Y Y Y B B Y B Y B	122	B Y B Y Y Y Y B Y B
37	Y B Y B B Y B Y B B	80	B B B B Y B Y Y B Y	123	Y Y B Y Y Y Y B B Y
38	B Y Y B B Y B Y B B	81	Y B B B Y B Y Y B B	124	B B Y Y Y Y Y B Y B
39	Y Y Y B B Y B B Y Y	82	B Y B B Y B Y Y B B	125	Y B Y Y Y Y Y B B Y
40	B B B Y B Y B Y B Y	83	Y Y B B Y B Y B Y Y	126	B Y Y Y Y Y Y B B Y
41	Y B B Y B Y B Y B B	84	B B Y B Y B Y Y B B	127	Y Y Y Y Y Y Y B B B
42	B Y B Y B Y B Y B B	85	Y B Y B Y B Y B Y Y		

Objaśnienie: Elementowi B odpowiada zero binarne nadawane wyższą częstotliwością akustyczną (1785 Hz), a elementowi Y odpowiada jedynka binarna nadawana niższą częstotliwością (1615 Hz)

Źródło: [4]

Przyjęty przez Rec. ITU-RM.493-9 kod Bergera jest kodem χ z parametrem charakterystycznym $K = 3$ [4]. Dzięki wykorzystaniu go, system DSC, jest w stanie spełnić założenia rekomendacji ITU (*International Telecommunication Union*) mówiące, że system powinien charakteryzować się prawdopodobieństwem błędu decyzyjnego rzędu 10^{-6}, przy prawdopodobieństwie błędu binarnego 10^{-2}. Poniższa tabela przedstawia zastosowany system synchroniczny z 10-elementowym kodem detekcyjnym wykorzystującym Międzynarodowy Alfabet Telegraficzny nr 5 (ITA-5) składający się z 128 (2^7) znaków.

Ciągi kodowe, stworzone wpierw według ustalonych reguł, tworzą kolejno pakiet będący przesyłany pomiędzy stacjami systemu. Ogólna postać takiego pakietu została przedstawiona na rysunku poniżej.

<table>
<tr><td rowspan="2">ciąg zerojedynkowy</td><td colspan="2">sekwencja fazująca</td><td rowspan="2">specyfikator formatu</td><td rowspan="2">adres</td><td rowspan="2">kategoria</td><td rowspan="2">samo-identyfikacja</td></tr>
<tr><td>synchronizacja bitowa</td><td>synchronizacja blokowa</td></tr>
</table>

<table>
<tr><td>wiadomość 1</td><td>wiadomość 2</td><td>wiadomość 3</td><td>...</td><td>znak końca sekwencji</td><td>znak detekcji błędu</td></tr>
</table>

Rys. 2.3 Ogólny schemat sekwencji wywoławczej. Źródło: [4]

Stacje wysyłająca i odbierająca pakiet tworzą prosty model systemu telekomunikacyjnego. Stacja nadawcza, dokonuje dwuetapowego przekształcenia sygnałów niosących informacje elementarne. Polega to na zmianę ciągów symboli na ciągi kodowe, które w postaci ciągów sygnałów elementarnych sterują układami modulacji sygnału nośnego. Aby zachować zgodność z Rec. ITU-RM.493-9 każdy sygnał transmitowany jest dwukrotnie w systemie czasowego odbioru zbiorczego (*time diversity*), gdzie przesunięcie czasowe pomiędzy transmisją DX, a retransmisją RX wynosi [4]:

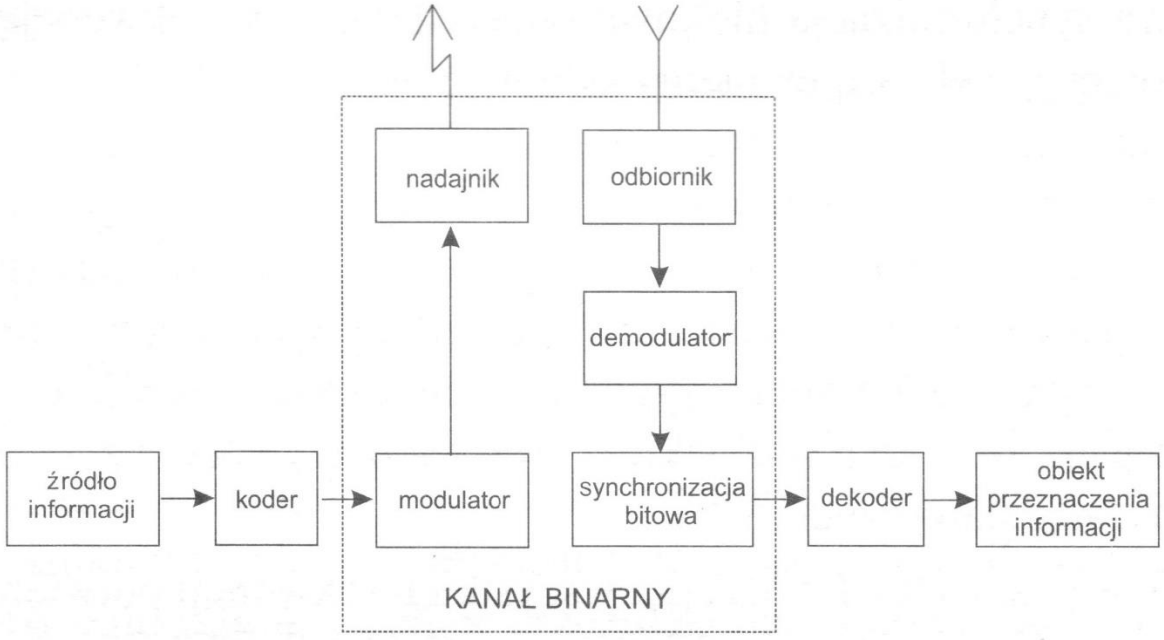
- 400 ms w paśmie MF/HF,
- $33\frac{1}{3}$ ms w radiotelefonicznych kanałach pasma VHF.

Tego typu transmisja nazywana jest również metodą rozgłoszeniową FEC (*Foward Error Correction*) i znalazła zastosowanie w innym podsystemie wchodzącym w skład GMDSS, a mianowicie w systemie telegrafii wysokopasmowej o wydruku bezpośrednim NBDP (*Narrow Band Direct Printing*) [4].

Klasy transmisji, przesunięcie częstotliwości i szybkość transmisji w DSC wynoszą [4]:

a) dla kanałów w pasmach MF i HF: F1B lub J2B, 170 Hz, 100 Bd,
b) dla kanałów w pasmach VHF: modulacja częstotliwości z preemafazą 6dB/oktawę, przesuw częstotliwości pomiędzy częstotliwościami akustycznymi 1300 Hz i 2100 Hz, podnośna 1700 Hz, a za szybkość modulacji 1200 Bd [4].

W odbiorniku zachodzi dwuetapowy proces, odwrotny do tego w nadajniku. Oznacza to, że w jednostce odbiorczej wpierw zostają podjęte niezbędne decyzje o każdym z sygnałów elementarnych, a następnie decyzje co do postaci poszczególnych symboli oraz całego pakietu. Na poniższym rysunku można zauważyć, że w tym wypadku kanał binarny jest stworzony z szeregu urządzeń służących do przekształcania ciągów kodowych w sygnały radiowe i sygnałów radiowych w ciągi kodowe [4].



Rys. 2.4 Metoda transmisji sygnałów w systemie DSC. Źródło: [4]

2.2.1 Format sekwencji wywoławczej

Tak jak to przedstawiono na rys. 2.3, każdą sekwencję fazującą poprzedza ciąg sygnałów zerojedynkowych. Wynosi on 200 bitów dla sygnałów alarmowych oraz 20 bitów dla wywołań innego typu. Wykorzystywany jest przed odbiorniki z przeszukiwaniem częstotliwości (*scanning receiver*) do zatrzymania dalszego przeszukiwania.

Z racji tego, że w systemie DSC pakiety danych są przesyłane bez synchronizacji momentów wprowadzenia zakodowanych wiadomości do kanału radiowego, dekodowanie pakietu musi zostać poprzedzone dwuetapową synchronizacją: na początku, w ciągu

odebranych elementarnych danych, zostają umiejscawiane ciągi kodowe (następuje synchronizacja bajtowa), a ostatecznie zostaje dokonana identyfikacja usytuowania części funkcjonalnych pakietu (synchronizacja blokowa).

2.2.1.1 Sekwencja fazująca

Każda sekwencja fazująca składa się z dwóch części [4]:

1. informacji dla odbiornika (*scanning receiver*), której głównym zadaniem jest umożliwienie zatrzymania dalszego przeszukiwania; odbiornik ten, stosownie do założeń GMDSS, prowadzi nasłuch na częstotliwościach zarezerwowanych dla wywołań DSC,
2. informacji służących dokładnemu odtworzeniu pozycji danych bitów oraz jednoznacznej lokalizacji pozycji ciągów kodowych wchodzących w skład całej sekwencji wywoławczej.

Synchronizacja bitowa występuje przed blokową, zapewniając tym samym odpowiednie warunki dla wczesnej synchronizacji bitowej. Sekwencja synchronizacji bitowej zbudowana jest z ciągu występujących naprzemiennie bitów B-Y (0-1). Jej długość powinna wynosić [4]:

- 200 bitów w pasmach HF i MF dla wywołań „pośrednictwo w niebezpieczeństwie", „potwierdzenie w niebezpieczeństwie", „niebezpieczeństwo" i „potwierdzenie odebranego sygnału o niebezpieczeństwie od stacji pośredniczącej" oraz dla każdej jednej sekwencji wywoławczej, której adresatami są stacje statkowe,
- 20 bitów w pasmach HF i MF dla wszystkich pozostałych sekwencji potwierdzających oraz dla wszystkich sekwencji wywoławczych, poza wywołaniami „niebezpieczeństwo", adresowanych do stacji nabrzeżnych,
- 20 bitów dla wszystkich wywołań w paśmie VHF.

Jednym z powszechniejszych błędów jest niewłaściwa synchronizacja będąca wynikiem błędu dozwolonego bitu w synchronizacji bitowej. Aby go uniknąć synchronizację osiąga się poprzez rozpoznanie poszczególnych symboli, a nie zmian w sekwencji synchronizacji bitowej. Do tego właśnie stosuje się synchronizację blokową. W sekwencji fazującej zwarte są symbole nadawane przemiennie na pozycjach DX i RX. Na pierwszej z nich nadawanych jest sześć symboli odzwierciedlających sześciokrotne nadanie symbolu 125. Natomiast na pozycji RX podawane są symbole synchronizujące. Określają one początek sekwencji informacyjnej i są to kolejno: 111, 110, 109, 108, 107, 106, 105, 104 [4].

Synchronizacja zostaje osiągnięta w momencie odebrania dwóch symboli RX i jednego DX, trzech RX lub dwóch symboli DX i jednego RX na podpowiadających im pozycjach [4].

2.2.1.2 Specyfikator formatu

Specyfikator formatu sekwencji wywoławczej definiuje postać danej sekwencji, zależnie od rodzaju wywołania. Ma postać numerów i jest on nadawany dwukrotnie, na pozycjach RX i DX. Najczęściej używanymi symbolami specyfikatora formatu są następujące numery [4]:

- 112, dla wywołań w niebezpieczeństwie,
- 116, dla wywołań „do wszystkich statków",
- 114, dla selektywnego wywołania statków o wspólnej cesze,
- 120, dla selektywnego wywołania pojedynczej stacji,
- 102, dla wywołania selektywnego do grupy statków przebywających w konkretnym obszarze geograficznym,
- 123, dla wywołania selektywnego do pojedynczej stacji mającej na wyposażeniu półautomatyczny serwis.

Specyfikator formatu mówi, jaki rodzaj adresu będzie zamieszczony w dalszej części sekwencji. W przypadku odbioru specyfikatorów formatu o numerach 112 (w niebezpieczeństwie) i 116 (do wszystkich statków), zalecane jest, aby odbiór został dokonany poprzez dekodery odbiorników obu znaków w celu efektywnej eliminacji niebezpieczeństwa błędu. Dla pozostałych rodzajów wywołań symbole adresu pozwalają na uniknięcie błędów, stąd też pojedyncza detekcja symbolu specyfikatora formatu jest dostateczna.

2.2.1.3 Adres

Po specyfikatorze formatu, w sekwencji wywoławczej, następuje część adresowa zawierająca informację jednoznacznie określającą adresata lub grupę adresatów danej sekwencji. Przy wywołaniu selektywnym lub danej grupy statków, adres numeryczny lub alfanumeryczny jest umieszczany w polu adresowym, gdzie typ adresu jest określany przez specyfikator formatu. Grupy statków w danym regionie geograficznym są definiowane za pomocą współrzędnych geograficznych kodowanych według siatki Merkatora. Adres nie jest podawany przy wywołaniach w zagrożeniu lub do wszystkich statków [4].

Pole adresowa składa się z szeregu symboli alfanumerycznych kodowanych ciągami według kodu ITA-5. Duże litery alfabetu łacińskiego odpowiadają ciągom o numerach 65-90, a cyfry 0-9 numerom 48-57. Należy pamiętać, że pierwszym symbolem adresu powinna być litera [4].

Dziesięciocyfrowy identyfikator, nazywany adresem numerycznym, składa się z dziewięciocyfrowej liczby dziesiętnej z zerem na dziesiątej pozycji. Każde kolejne dwie cyfry dziesiętne zostają zakodowane w formie symbolu o wartości z zakresu 00-99 zajmującego jeden dziesięciobitowy ciąg kodowy. Razem, adres numeryczny, zbudowany jest z pięciu symboli (ciągów kodowych) [4].

Poniższa tabela przedstawia metodę zapisu licz dziesiętnych z użyciem kodu dziesięciobitowego. Sekwencja bitowa D2-D1 ulega zmianie w zakresie od 00 do 99 w każdym z symboli. Symbole oznaczające liczby dwucyfrowe są przesyłane w formie dziesięciobitowego ciągu kodowego. Jeżeli liczba posiada nieparzystą ilość cyfr, zostaje poprzedzona zerem w celu zachowania całkowitej liczby symboli [4].

Tabela 2.2 Przedstawianie liczb wyrażonych w formie dziesiętnej za pomocą 10-bitowych ciągów kodowych (symboli).

Sposób wyrażenia									
Tysiące milionów D2	Setki milionów D1	Dziesiątki milionów D2	Miliony D1	Setki tysięcy D2	Dziesiątki tysięcy D1	Tysiące D2	Setki D1	Dziesiątki D2	Jedności D1
Symbol 5		Symbol 4		Symbol 3		Symbol 2		Symbol 1	

Źródło: [4]

Zgodnie z Regulaminem Radiokomunikacyjnym identyfikatory morskich ruchomych stacji tworzone są z serii dziewięciu cyfr, gdzie pierwsze trzy są tzw. Morskimi Cyframi Identyfikacyjnymi MID (*Maritime Identification Digits*) i określają do jakiego państwa należy dana radiostacja. Identyfikator, w przypadku nadawania, znajduje się w części samo identyfikującej wywołania, a przy odbiorze w adresie. Jest on transmitowany w postaci pięciu symboli, zawierających w sumie 10 cyfr [4]:

$$(X_1, X_2)\ (X_3, X_4)\ (X_5, X_6)\ (X_7, X_8)\ (X_9, X_{10}).$$

Ostatnia cyfra (X_{10}) zawsze jest zerem, ponieważ w praktyce dziewięciopozycyjne identyfikatory w zupełności wystarczają.

Dla polepszenia wywołań grup radiostacji statkowych dzielących wspólną cechę (interes), wdrożono tzw. wywołania grupowe. Sprowadza się to do przypisania określonej grupie radiostacji tego samego adresu wywołań grupowych. Z racji braku formalnych

regulacji w kontekście ilości tworzonych grup radiostacji, każda radiostacja posiada swój identyfikator indywidualny oraz przynajmniej jeden adres wywołań grupowych [4].

Adresy wywołań grupowych budowane są na podstawie identyfikatorów poszczególnych stacji wykorzystywanych do kreacji numerów radiostacji. Numery radiostacji to adresy widziane z lądowych sieci telekomunikacyjnych i są używane przy zautomatyzowanych połączeniach DSC w łączu sieć publiczna - statek. Takie połączenie realizowane jest dzięki automatycznej konwersji przez stację nabrzeżną numerów radiostacji statkowych na odpowiadające im identyfikatory [4].

Postać identyfikatora wygląda następująco: X_1 X_2 X_3 X_4 X_5 X_6 X_7 X_8 X_9. Pierwsze trzy cyfry określające MID oznaczają: X_1 – rejon geograficzny, a (X_2, X_3) – kraj. Pierwsza z nich (X_1) przyjmuje następujące wartości [4]:

0 – wywołanie grupowe,

1 – zarezerwowane dla przyszłych zastosowań,

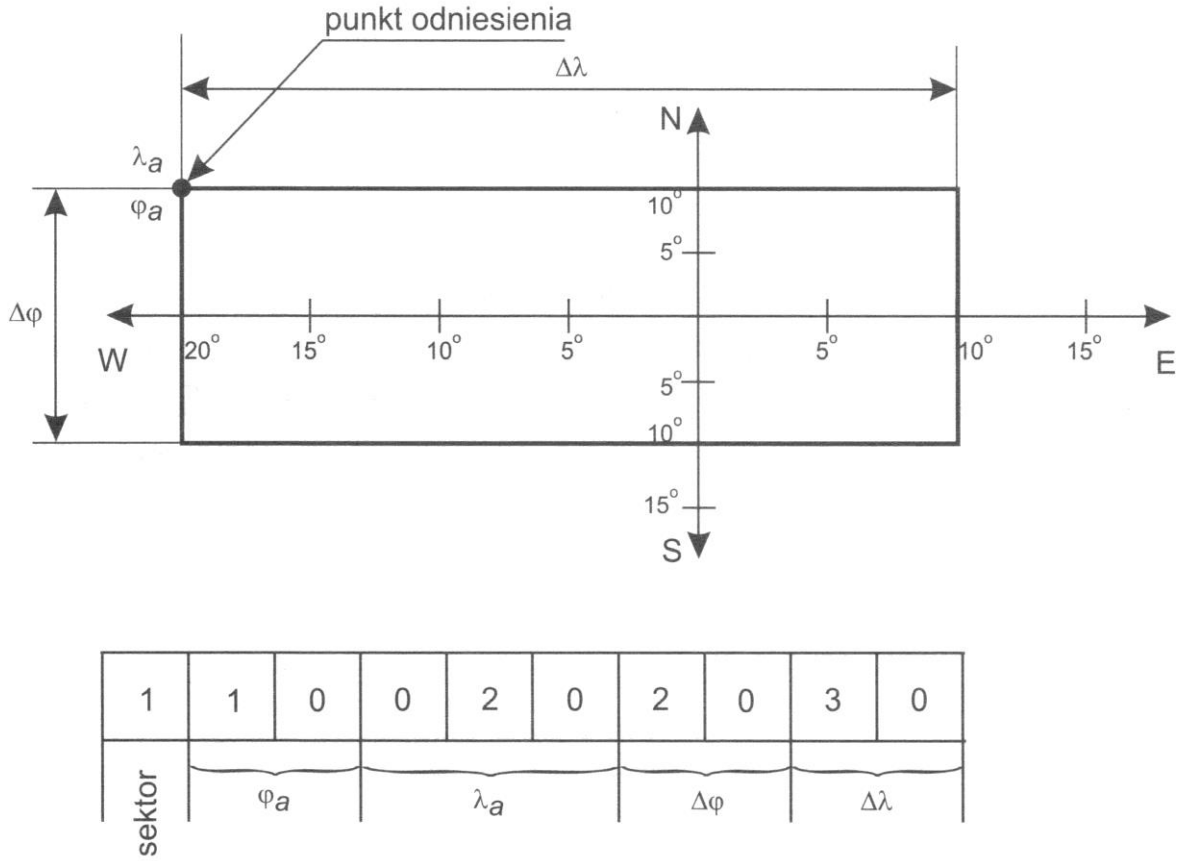
2 – Europa,

3 – Ameryka Północna,

4 – Azja (bez części południowo-wschodniej).

Zadaniem radiostacji nabrzeżnej jest zdekodowanie skrótu 8 Y, otrzymując w ten sposób MID i wstawienie w pozycji X_8 i X_9 wartości 0. Proces ten dokonuje się automatycznie, bez udziału radiooperatora. Jedną z niewątpliwych wad przedstawionego rozwiązania jest prawdopodobieństwo wystąpienia do 10 MID-ów w jednej stacji nabrzeżnej, przez co korzystanie z tej metody do zwiększenia liczby radiostacji, ze zautomatyzowanym dostępem do sieci lądowych, wymaga wprowadzania odpowiednich uzgodnień międzynarodowych. Gdy powstanie międzynarodowa publiczna sieć komutowana z możliwością transmisji siedmiocyfrowych numerów radiostacji, będą mieć postać MID X_4 X_5 X_6 X_7 00. Stacja nabrzeżna będzie musiała jedynie automatycznie wstawić 0 w pozycje X_8 i X_9. Siedmiocyfrowe numery identyfikacyjne radiostacji statkowych będą mogły bez przeszkód egzystować równocześnie z numerami wykorzystującymi skrót 8 Y [4].

W wywołaniu do statków znajdujących się w określonym obszarze geograficznym adres geograficzny ma postać dziesięciu cyfr, czyli pięciu symboli przekładających się na współrzędne geograficzne zadanego obszaru będącym prostokątem w siatce Merkatora z punktem odniesienia w lewym górnym rogu. Podawana jest najpierw szerokość, a potem długość geograficzna [4].



Rys. 2.5 Sposób opisu obszaru geograficznego w sekwencji wywoławczej. Źródło: [4].

Kolejne cyfry adresu oznaczają [4]:

- pierwsza, sektor kuli ziemskiej, w którym leży punkt odniesienia: sektor NE – 0, NW – 1, SE – 2, SW – 3,
- druga i trzecia, szerokość geograficzną punktu odniesienia w dziesiątkach i jednościach stopni,
- czwarta, piąta i szósta, długość geograficzną punktu odniesienia w setkach, dziesiątkach i jednościach stopni,
- siódma i ósma, współrzędną pionową (północ-południe) w danym prostokącie siatki Merkatora (Δφ) w dziesiątkach i jednościach stopni,
- dziewiąta i dziesiąta, współrzędną poziomą (zachód-wschód) w danym prostokącie Merkatora (Δλ), w dziesiątkach i jednościach.

2.2.1.4 Kategoria

Informacja zawarta w polu kategorii jest kodowana w sposób opisany w poniższej tabeli i określa priorytet całej sekwencji wywoławczej. Pole to jest puste w przy wywołaniach w niebezpieczeństwie, ponieważ w tym przypadku priorytet został już wcześniej zdefiniowany przez specyfikator formatu [4].

Dla wywołań związanych z bezpieczeństwem, w polu kategorii, przewidziano wystąpienie następujących informacji [4]:

- alarmowanie,
- nagła potrzeba, pilność,
- bezpieczeństwo.

Reszta wywołań zawiera poniższe informacje [4]:

- interesy statku: obsługa komunikacji brzeg-statek i statek-brzeg (6. priorytet wg. Regulaminu Radiokomunikacyjnego),
- wywołania rutynowe

Tabela 2.3 Sposób kodowania kategorii.

Numer symbolu	Kategorie wywołań
112	Niebezpieczeństwo (*distress*)
110	Pilność (*urgency*)
108	Bezpieczeństwo (s*afety*)
106	Interesy statku (*interest*)
100	Wywołanie rutynowe (*routine*)

Źródło: [4]

2.2.1.5 Samoidentyfikacja

W polu samoidentyfikacja umieszczony zostaje dziewięciocyfrowy identyfikator stacji nadawczej, kodowany w taki sam sposób, jak w przypadku adresu znajdującego się w polu adres sekwencji wywoławczej. W przypadku wywołań będącymi odpowiedziami potwierdzającymi (a*cknowledge respond*), samoidentyfikacja zostaje przypisana automatycznie z pola adresowego odebranego wywołania [4].

2.2.1.6 Blok wiadomości

Najważniejszą rolą selektywnego cyfrowania wywołania DSC jest przesyłanie wiadomości pomiędzy stacjami. Każda wiadomość w sekwencji wywoławczej składa się z szeregu elementu zależnych od rodzaju wywołania. Na potrzeby niniejszej pracy przedstawię format w postaci wywołania w niebezpieczeństwie, które wykorzystuje wszystkie cztery pola wiadomości.

2.2.1.7 Wiadomość 1

W tym polu opisywany jest rodzaj niebezpieczeństwa zagrażającego statkowi i zostaje zakodowany w sposób podany w poniższej tabeli.

Tabela 2.4 Rodzaje niebezpieczeństwa.

Numer symbolu	Rodzaj niebezpieczeństwa
100	Ogień, eksplozja
101	Przeciek kadłuba
102	Kolizja
103	Wejście na mieliznę
104	Przechył, niebezpieczeństwo przewrotki
105	Tonięcie
106	Utrata mocy i sterowności
107	Inne niebezpieczeństwo
108	Opuszczenie statku
112	Emisja radiopławy awaryjnej EPIRB, która służy do określenia pozycji

Źródło: [4]

2.2.1.8 Wiadomość 2

Wiadomość ta zawiera pozycję statku znajdującego się w niebezpieczeństwie. Jest zapisywana zgodnie z poniższą tabelą, z wykorzystaniem dziesięciu cyfr przekładających się na pięć symboli. Wartości sektorów kuli ziemskiej to: 0 – NE, 1 – NW, 2 – SE, 3 – SW.

Tabela 2.5 Metoda zapisu danych lokalizacyjnych pozycji statku w niebezpieczeństwie.

<table>
<tr><td rowspan="2">Znak 5</td><td>X</td><td colspan="2">Sektor kuli ziemskiej</td></tr>
<tr><td>X</td><td>Dziesiątki stopni</td><td rowspan="4">Szerokość geograficzna</td></tr>
<tr><td rowspan="2">Znak 4</td><td>X</td><td>Jedności stopni</td></tr>
<tr><td>X</td><td>Dziesiątki minut</td></tr>
<tr><td rowspan="2">Znak 3</td><td>X</td><td>Jedności minut</td></tr>
<tr><td>X</td><td>Setki stopni</td><td rowspan="5">Długość geograficzna</td></tr>
<tr><td rowspan="2">Znak 2</td><td>X</td><td>Dziesiątki stopni</td></tr>
<tr><td>X</td><td>Jedności stopni</td></tr>
<tr><td rowspan="2">Znak 1</td><td>X</td><td>Dziesiątki minut</td></tr>
<tr><td>X</td><td>Jedności minut</td></tr>
</table>

Źródło: [4]

W przypadku niemożliwości określenia pozycji statku, dziesięć następujących po sygnale „rodzaj niebezpieczeństwa” cyfr powinno zostać zamienione w dziesięciokrotne powtórzenie liczby 9 (5 symboli nr 99), w celu zachowania ciągłości sekwencji [4].

2.2.1.9 Wiadomość 3

Tutaj przesyłane są informacje mówiące o czasie (UTC), w którym została określona pozycja statku w niebezpieczeństwie. Pierwsze dwie cyfry mówią o godzinie, trzecia i czwarta określa minuty. W przypadku niemożności podania czasu (np. pozycja nie mogła zostać określona) dwa, wymienione uprzednio symbole, powinny zostać zastąpione symbolami o numerze 88, czyli trzecia wiadomość powinna składać się z samych ósemek [4].

2.2.1.10 Wiadomość 4

Czwarta wiadomość składa się z jednego znaku określającego rodzaj późniejszej komunikacji (wydruk bezpośredni lub telefonia), najkorzystniejszej dla stacji znajdującej się w niebezpieczeństwie. Przykładowe wartości mogące się tutaj znaleźć: 109 – J3E, 113 – F1B/J2B, 100 – F3E/G3E simplex.

2.2.1.11 Koniec sekwencji wywoławczej

Każdą sekwencję wywoławczą należy zakończyć znakiem końca sekwencji nadawanym trzy razy na pozycji DX i raz na RX. Może nim być jeden z trzech symboli oznaczających:

- 117, wywołanie wymagające potwierdzenia RQ (*RQ Acknowledge required*),
- 112, wywołanie będące odpowiedzią na wywołanie wymagające potwierdzenia BQ (*BQ Acknowledge respond*),
- 127, reszta wywołań (*other calls*).

Tabela 2.6 Sekwencja wywoławcza na przykładzie wywołań niebezpieczeństwo i do wszystkich statków.

Specyfikator formatu (2)	Adres (5)	Kategoria (1)	Samoidentyfikacja (5)	Wiadomość				EOS (1)	ECC (1)
				1	2	3	4		
Wywołanie w niebezpieczeństwie 112	–	–	00–99	(1) Rodzaj niebezpieczeństwa 100–124	(5) Pozycja statku w niebezpieczeństwie 00–99	(2) Czas	(1)*	127	ECC
Do wszystkich statków 116	–	Niebezpieczeństwo 112 Pilność 110 Bezpieczeństwo 108	00–99	(2) Telekomenda 100–126 oprócz 117, 122, 125	(6) Częstotliwość lub kanał 00–99	Nie użyta	Nie użyta	127	ECC

Objaśnienie: W nawiasach znajduje się ilość wysyłanych znaków w sekwencji wywoławczej, gwiazdka oznacza rodzaj dalszej komunikacji, ECC (*Error check character*) – znak detekcji błędu, EOS (*End of sequence)* – sygnał końca sekwencji.

Źródło: [4]

2.2.1.12 Znak detekcji błędów

Ostatnim znakiem emitowany w sekwencji wywoławczej jest znak detekcji błędów. Jego rolą jest wykrycie błędów przeoczonych podczas analizy, przez urządzenie odbiorcze, dziesięciobitowego kodu detekcyjnego oraz rozdziału czasowego DX i RX. Jest to ostateczne zabezpieczenie wywołania przed błędami występującymi w odbieranych sekwencjach. Znak detekcji błędu zostaje utworzony poprzez przypisanie jego siedmiu bitom informacyjnym najmniej znaczącego bitu sum modulo-2 wszystkich odpowiadających im bitów w znakach informacyjnych (z wyjątkiem ciągu zerojedynkowego, sekwencji fazującej, specyfikatora formatu i znaku detekcji błędu). Automatyczna emisja potwierdzenia otrzymanego wywołania przez stację odbiorczą powinna nastąpić dopiero po odebraniu znaku detekcji błędu i dokonania nim walidacji otrzymanego komunikatu.

3. PROJEKT APLIKACJI DO ODBIORU INFORMACJI Z RADIOSTACJI MF/HF

3.1 Założenia i cel aplikacji

Celem pracy było stworzenie aplikacji umożliwiającej obiór i archiwizację danych otrzymywanych przez radiostację, z wykorzystaniem komputera klasy PC. Otrzymywane informacje obejmowały zarówno metadane każdego zarejestrowanego połączenia, jak i pełny dźwiękowy zapis komunikacji głosowej. Całość miała tworzyć swojego rodzaju czarną skrzynkę będącą wsparciem dla oficera łączności, a także narzędziem kontroli dla przełożonego.

Wzorcowym modelem radiostacji, obsługiwanej przez aplikację, był SAILOR RE 2100. Ze względu na swoją prostą konstrukcję umożliwiającą naprawę w większości światowych portów oraz uniwersalność, urządzenie to znalazło sobie miejsce na wielu statkach świata, będąc tym samym idealnym modelem dla tego typu aplikacji.

Komunikacja z radiostacją odbywa się w sposób cyfrowy z wykorzystaniem portu COM w standardzie RS-232. Do realizacji części praktycznej wykorzystano również adapter z portu szeregowego na trójpolowego minijacka 3,5 mm. Tym samym stało się możliwe przechwycenie komunikatów dźwiękowych przez komputer klasy PC. Nieobecność portu COM we współczesnych komputerach dedykowanych rynkowi konsumenckiemu wymagała zastosowania adaptera z portu USB na port szeregowy.

W toku nauki na Akademii Morskiej, najwięcej doświadczenia nabyłem w programowaniu z wykorzystaniem platformy .NET i języka C#. Wymagania dotyczące projektu nie precyzowały docelowego systemu operacyjnego. Z tego względu, wziąłem pod uwagę popularność poszczególnych systemów operacyjnych na platformie PC. Na dzień 01.06.2019, najpopularniejszy był system Windows z wynikiem 79,45% [8]. Tym samym, platforma .NET będzie kompatybilna z większością aktywnych systemów komputerowych. W przyszłości, niskim nakładem pracy, możliwe będzie dostosowanie stworzonej aplikacji do świeżo udostępnionego .NET Core, którego integracja do platformy .NET nastąpi wraz z premierą frameworku .NET 5.0. Dzięki .NET Core możliwe jest kompilowanie aplikacji napisanej w C# do plików binarnych dla każdego z trzech najpopularniejszych systemów operacyjnych, tj. Windows, MacOS i Linux.

3.2 Zastosowane technologie i biblioteki

Środowisko .NET Framework posiada wsparcie dla szeregu technologii, których wybór ma bezpośredni wpływ na kształt przyszłej aplikacji. Niniejszy projekt został wykonany w technologii WPF, będącej swoistym połączeniem języków XAML i C#. Technologia ta jest nowsza od wykorzystywanego podczas zajęć laboratoryjnych Windows Forms. Dzięki zastosowaniu nowszego wzorca przy projektowaniu aplikacji (MVVM), możliwym stało się stworzenie dynamicznych interfejsów użytkownika.

Dodatkowym atutem .NET Framework jest wsparcie dla pakietów (*packages*) będących odpowiednikiem bibliotek DLL. Dzięki dedykowanemu menadżerowi (*NuGet*), jesteśmy w stanie zainstalować odpowiednie pakiety wraz z ich zależnościami [10]. Łatwość instalacji nowych paczek, zarządzania ich zależnościami, oraz wsparcie dla jednego z największych hostingów dla projektów programistycznych (GitHub) zapewnia NuGetowi rosnącą popularność, a tym samym różnorodność udostępnianych przez użytkowników pakietów. Wszystkie wykorzystane w projekcie biblioteki (od 3.2.4 do 3.2.7) zostały zainstalowane z wykorzystaniem NuGeta.

3.2.1 Silnik graficzny WPF

Wraz z rozwojem sprzętu graficznego, a tym samym oczekiwań konsumentów w kontekście wyglądu aplikacji, „gigant z Redmont" opracował nowy produkt pozwalający uwolnić się od ograniczeń GDI+ (podsystem Windows XP; API udostępnione poprzez zestaw klas C++ służący do wyświetlania informacji) oraz Windows USER (komponent systemów operacyjnych Microsoft Windows służący do tworzenia prostych interfejsów użytkownika). Udało się tego dokonać w 2006 roku, gdy premierę miała pierwsza wersja Windows Presentation Foundation (WPF). Powstała w ten sposób biblioteka łączyła najlepsze funkcje takich technologii jak Windows Forms (wydajność programisty), HTML (deklaratywne znaczniki), Adobe Flash (zaawansowane wsparcie dla animacji), oraz DirectX (3D i akceleracja sprzętowa) [7].

3.2.2 Deklaratywny język znaczników XAML

XAML (*eXtensible Application Markup Language*) to względnie prosty, deklaratywny, język znaczników zainspirowany językami XML (*eXtensible Markup Language*) i HTML (*HyperText Markup Language*). Jest mechanizmem wykorzystywania API frameworka .NET pozwalającym oddzielić część odpowiedzialną za interakcję z użytkownikiem od części logicznej programu [9]. Dzięki temu projektanci UI/UX (*User*

Interface/User eXperience) mogą pracować niezależnie od programistów C#. Potrzeba takiej rozdzielności wywodzi się ze skuteczności nowoczesnych metodyk tworzenia aplikacji internetowych (np. SCRUM) oraz potrzeby wdrożenia ich do procesu projektowania i implementacji aplikacji desktopowych. Część widoczna dla użytkownika, (frontend), komunikuje się z częścią logiczną programu (backend) przede wszystkim za pomocą dwóch mechanizmów.

Pierwszym z nich jest tzw. code behind, czyli plik .cs, w którym można pisać kod w języku C# oraz plik .xaml z kodem XAML. Podejście to pozwala powiązać logikę aplikacji bezpośrednio z widokiem, dzięki czemu interfejs użytkownika ma bezpośredni dostęp do zmiennych i może wywoływać fragmenty kodu (metody). Taki sposób pisania aplikacji nie jest zalecany, ponieważ widoku od logiki nie oddziela żadna warstwa abstrakcji. Ułatwia to użytkownikowi ingerencję w logikę, zmniejszając bezpieczeństwo programu. Dlatego też, posługiwanie się code behind jest uzasadnione jedynie przy walidacji wprowadzanych danych przez użytkownika. Wyjątkiem od reguły są naprawdę małe projekty, gdzie model MVVM wydłużyłby proces tworzenia aplikacji w sposób nieproporcjonalny.

Drugim mechanizmem komunikacji, jest Binding (wiązanie). W dowolnej właściwości kontrolki można użyć instrukcji „Binding [nazwa]" (np. `ComboBox SelectedValue="{Binding StopBitsValue}"`), aby móc w danej przestrzeni nazw odnieść się do uprzednio zdefiniowanego zasobu. Następnie, po ustawieniu nowej wartości w kodzie C#, należy poinformować interfejs użytkownika o zmianie. Tym samym, wykonana zostaje metoda rozszerzająca interfejs `INotifyPropertyChanged`, dzięki czemu UI zostaje odświeżony. Zrozumienie tego mechanizmu jest niezbędne do sprawnego posługiwania się WPF [7].

3.2.3 Język programowania C#

Język wydany przez firmę Microsoft wraz z premierą Visual Studio .NET 2002. Jest on bogatą implementacją paradygmatu obiektowego, a więc spełnia takie założenia jak enkapsulacja, dziedziczenie, czy polimorfizm. Cechami charakterystycznymi C#, z punktu widzenia obiektowości, są [1]:

- system typów – fundamentem języka jest zamknięta jednostka danych nazywana typem. W systemie typów wszystkie typy współdzielą wspólny typ bazowy. Dla przykładu, każdy typ może być przekonwertowany na string poprzez wywołanie na nim metody `ToString`,

- klasy i interfejsy – w tradycyjnym paradygmacie obiektowym jedynym typem jest klasa. W C# istnieje wiele rodzajów typów. Jednym z nich jest interfejs, który wygląda podobnie do klasy, ale różni się tym, że jedynie opisuje obiekty i metody, bez zawierania w sobie implementacji. Jest to szczególnie przydatne w sytuacjach wymagających wielokrotnego dziedziczenia, którego C# (w odróżnieniu od np. C++ lub Eiffel) nie wspiera,
- właściwości, metody i wydarzenia – w czystej obiektowości, wszystkie funkcje są metodami. W C# natomiast, metody wraz z właściwościami i zdarzeniami, są jednym z wielu składników funkcji. Tak więc możliwe jest przekazanie danej metodzie argumentu w postaci funkcji.

Pomimo tego, że język C# jest przede wszystkim obiektowy, zapożycza również cechy charakterystyczne dla programowania funkcyjnego. Jedną z najważniejszych jest możliwość traktowania funkcji jak wartości. Jest to możliwe dzięki wykorzystaniu delegatów umożliwiających np. programowanie wielowątkowe [1].

3.2.4 Biblioteka NAudio

Jest to dźwiękowa biblioteka open source napisana przez Marka Heatha dla frameworku .NET. Pozwala na odtwarzanie i zapis dźwięku z wykorzystaniem API dostępnych w systemach Microsoft. Kluczową funkcją tej biblioteki jest możliwość nagrania dźwięku bezpośrednio z wejść mikrofonowego oraz głośnikowego z wykorzystaniem WASAPI. Dodatkowym atutem był zapis dźwięku do kontenera WAV oraz możliwość odtworzenia go.

Dostępna jest także dokumentacji biblioteki w formie krótkich poradników. Pomocne okazały się również demonstracyjne programy, wraz z kodem źródłowym, udostępnione przez twórcę w celu zaprezentowania funkcjonalności biblioteki [11].

3.2.5 Biblioteka Extended WPF Toolkit

Niestety, w podstawowej wersji WPF brakuje wielu niezbędnych kontrolek, które znajdowały się chociażby w Windows Forms [7]. Między innymi brak jest kontrolki odpowiadającej `NumericUpDown`. Jest ona niezbędna w przypadku precyzyjnego wyboru wartości całkowitych w takich miejscach jak szybkość transmisji (*bitrate*), bity danych (*data bits*), czy też czasu końca (*timeout*).

Gotowe rozwiązanie przygotowała firma Xceed w postaci biblioteki open source o nazwie Extended WPF Toolkit. Zawiera ona w sobie kolekcję kontrolek, komponentów

i narzędzi umożliwiających stworzenie łatwych w użyciu aplikacji WPF na miarę współczesnych czasów [12].

3.2.6 Biblioteka GMap.NET

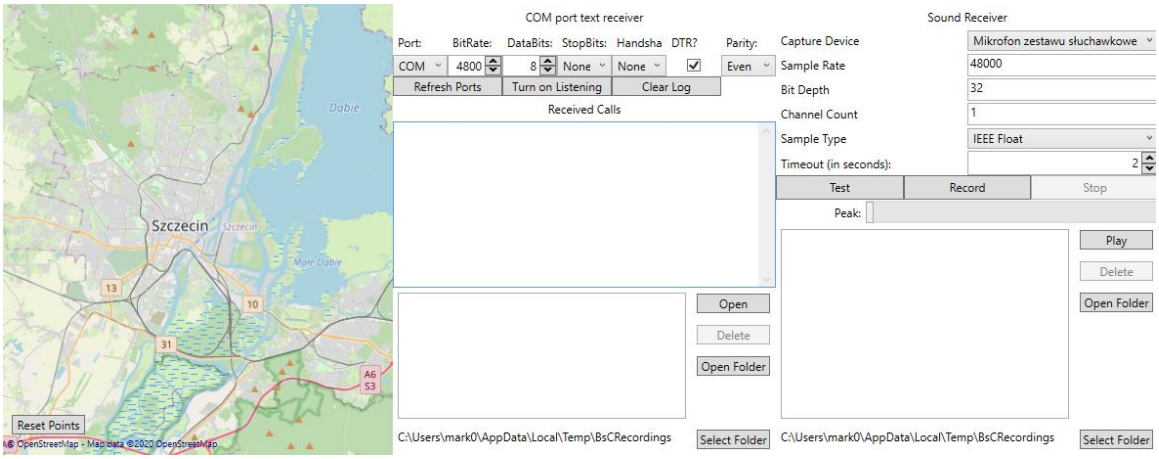
Radiostacja SAILOR RE 2100 jest w stanie przekazać, za pośrednictwem portu COM odebrane współrzędne geograficzne nadawcy, jeżeli tylko zostały przez niego dodane do wiadomości DSC. Każdy nawigator potwierdzi, że pozycja bez mapy nie jest wystarczająca. Rozwiązaniem tego problemu okazała się biblioteka GMap.NET. Jest to potężna open source'owa kontrolka umożliwiająca wyświetlenie dowolnej pozycji geograficznej na mapie od jednego z ponad szesnastu obsługiwanych przez nią dostawców map. Jej API okazało się proste w użyciu, jednakże wymagała przesyłania informacji pomiędzy kontrolerami i ViewModelami, co w WPFie jest dosyć kłopotliwe. Wynika to z braku obsługi wielokrotnego dziedziczenia przez C#. Najprostszym sposobem, niewymagającym przebudowy całej aplikacji, było użycie biblioteki TinyMessenger [13].

3.2.7 Biblioteka TinyMessenger

Jest to agregator zdarzeń (eventów) pozwalający komunikować się niezwiązanym ze sobą obiektom. Dzięki zastosowaniu modelu Publish/Subscribe (Opublikuj/Zasubskrybuj) pozbawionego implementacji kolejki, umożliwia zasubskrybowanie wiadomości przed jej opublikowaniem. Podmiotem komunikacji jest sama wiadomość. Tym samym, aby odebrać wiadomość, odbiorca musi jedynie wiedzieć co chce otrzymać. Wykorzystanie tej biblioteki umożliwiło przesłanie, w sposób asynchroniczny, danych odebranych przez kontroler portu COM do kontrolera odpowiadającego za wyświetlanie mapy.

3.3 Zarys ogólny opracowanej aplikacji

Zadaniem postawionym przed aplikacją było umożliwienie archiwizacji w czasie rzeczywistym informacji otrzymanych poprzez port COM z radiostacji MF/HF. Dodatkowo, aplikacja powinna mieć możliwość rejestracji przychodzących i wychodzących komunikatów dźwiękowych. Została również zaimplementowana mapa ukazująca pozycję nadawcy. Założeniem zapisanych danych było umożliwienie radiooperatorowi, bądź oficerowi przeprowadzającemu kontrolę, odtworzenia w dowolnym momencie informacji otrzymanych drogą radiową w celu ich weryfikacji. Językiem aplikacji został język angielski, ponieważ jest on podstawowym językiem wykorzystywanym w radiokomunikacji morskiej.



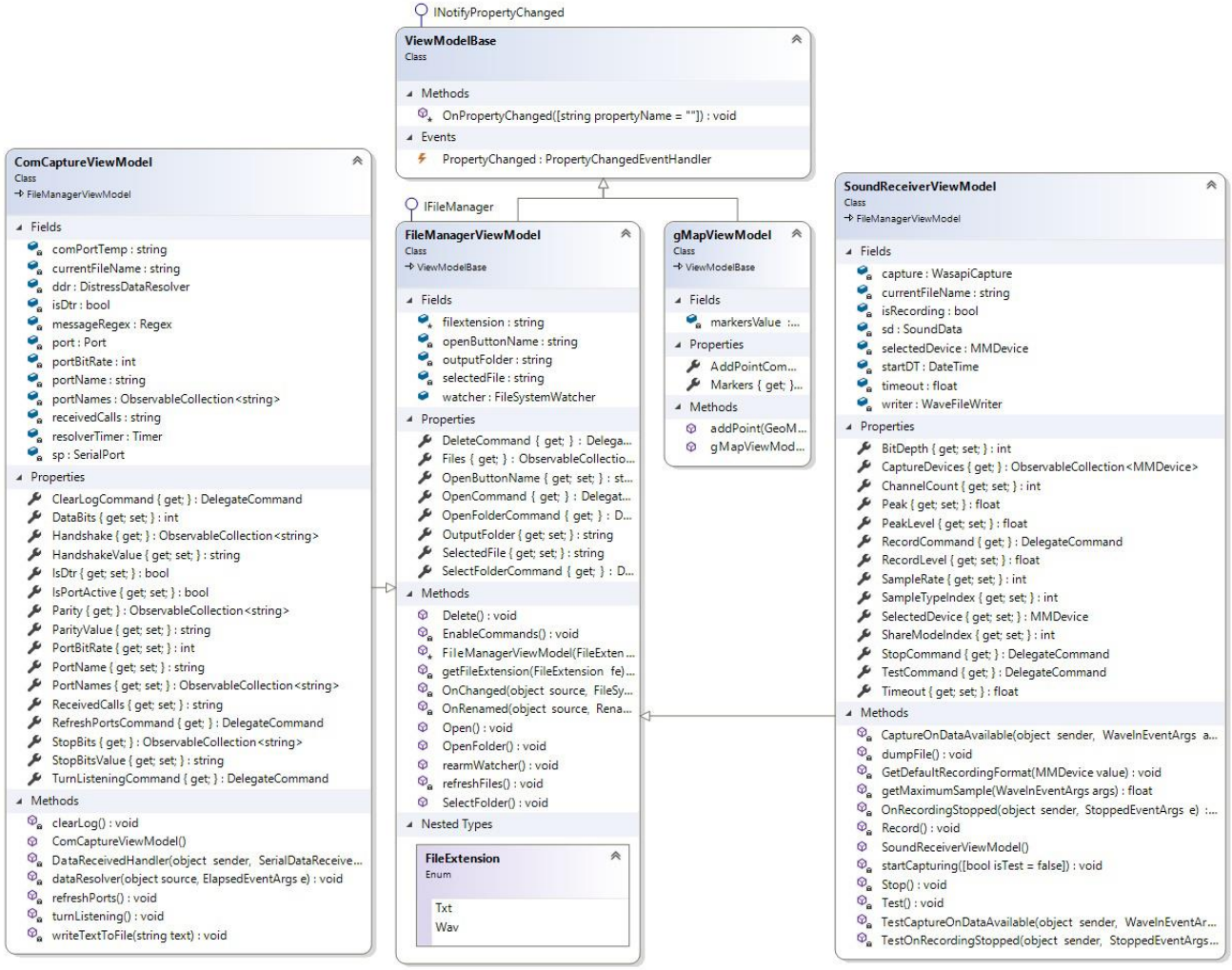
Rys. 3.1 Okno aplikacji tuż po uruchomieniu. Źródło: Opracowanie własne

Okno główne aplikacji składa się z siatki (Grid) mającej 3 kolumny. Pierwsza posiada 400 px szerokości, a następne dwie dzielą pozostałe miejsce pomiędzy siebie (`1*`).

Listing 3.1. Styl okna głównego.

```
<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="400"/>
        <ColumnDefinition Width="1*"/>
        <ColumnDefinition Width="1*"/>
    </Grid.ColumnDefinitions>
    <local:gMapView Grid.Column="0"/>
    <local:ComCaptureView Grid.Column="1"/>
    <local:SoundReceiverView Grid.Column="2"/>
</Grid>
```

Każda z trzech kolumn odpowiada innej funkcjonalności systemu. Pierwsza od lewej wyświetla pozycję nadawcy na mapie, druga służy do obsługi portu COM, a ostatnia ma na celu obsługę rejestracji dźwięku. Struktura stworzonego programu prezentuje się następująco:



Rys. 3.2 Diagram klas przedstawiający modele widoków (ViewModels). Źródło: Opracowanie własne

Rys. 3.3 Diagram klas przedstawiający modele (kolor czerwony), interfejs (kolor zielony) i widoki (kolor niebieski). Źródło: Opracowanie własne

Interfejs składa się z szeregu deklaracji i służy do opisywania możliwie najmniejszych funkcjonalności. Projektując interfejsy warto kierować się zasadą segregacji interfejsów będącą częścią SOLIDu zaproponowanego przez Roberta C. Martina. Polega ona na tworzeniu takich interfejsów, aby zawierały minimalną liczbę deklaracji metod, zachowując przy tym ich ścisłe powiązanie w obszarze funkcjonalności. Pozwala to uniknąć implementowania interfejsów z niepotrzebnymi metodami. Ze względu na stosunkowo niewielki rozmiar projektu, świadomie porzuciłem segregację interfejsów. W języku C# przyjęto, że nazwa każdego interfejsu rozpoczyna się wielką literą „I".

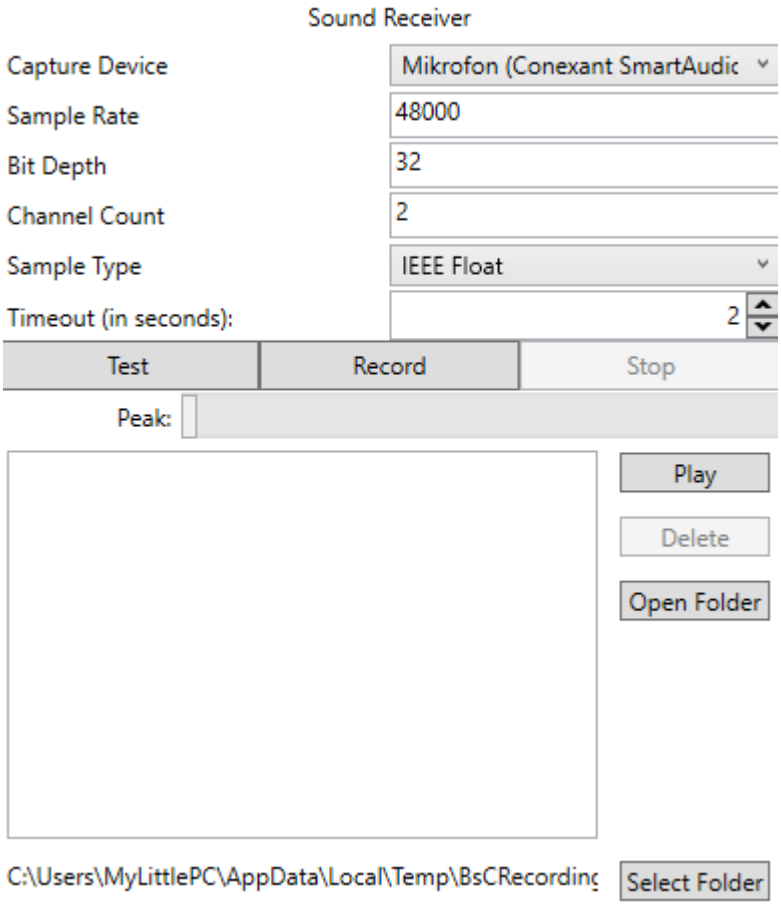
Modele składają się z pól (w przypadku tworzenia getterów i seterów) oraz właściwości, które określają typy danych, jakie będą wykorzystywać klasy dziedziczące po modelach. Modele opisują obiekty świata rzeczywistego.

Widoki dzielą się na dwie części. Pierwszą z nich jest code behind, czyli prosta klasa składająca się głównie z konstruktora. We wzorcu MVVM dopuszczalne jest jedynie umieszczenie tam walidacji danych wprowadzanych w pola interfejsu użytkownika oraz ustawienie pól kontrolek zaimportowanych z wcześniejszych implementacji frameworku .NET, takich jak Windows Forms. W aplikacji wykorzystano code behind do walidacji danych w kontrolce `myUpDownControl` w widoku `ComCaptureView`. Służy do tego regex `"[^0-9.-]+"` filtrujący wszystkie znaki nie będące cyframi. Drugą częścią widoku jest plik XAML definiujący ułożenie elementów interfejsu użytkownika. Język do tego użyty jest skrzyżowaniem HTML z XML, a tym samym relatywnie łatwy do czytania przez człowieka. Ważną częścią relacji pomiędzy C# i XML są bindingi. Binding to etykieta dowolnego typu, pozwalająca przypisać dwukierunkowo dane z kodu C# do interfejsu użytkownika. W celu wyświetlenia zaktualizowanych danych, widok musi zostać poinformowany o ich zmianie poprzez wywołanie eventu `PropertyChanged` z argumentem w postaci nazwy bindingu. Implementacja wygląda w sposób następujący:

```
PropertyChanged?.Invoke(this, new PropertyChangedEventArgs( propertyName));
```

W następnych rozdziałach dokładniej opisano działanie każdej z części aplikacji.

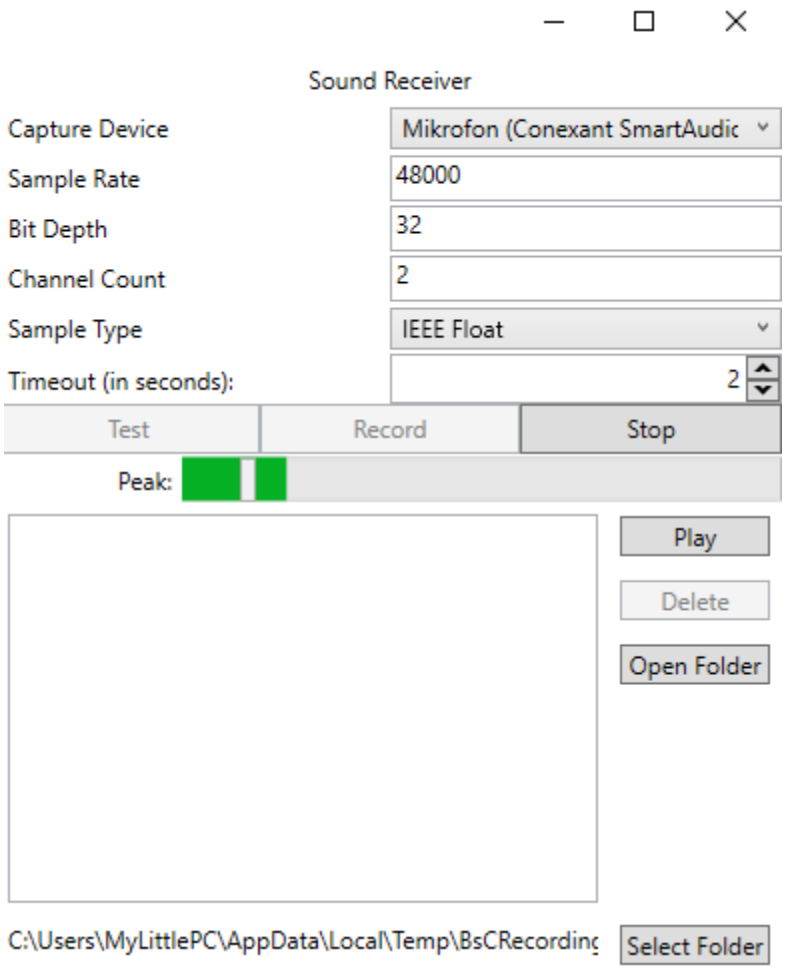
3.4 Przetwarzanie dźwięku z radiostacji



Rys. 3.4 Część interfejsu użytkownika w aplikacji odpowiadająca za przetwarzanie dźwięku w radiostacji. Źródło: Opracowanie własne

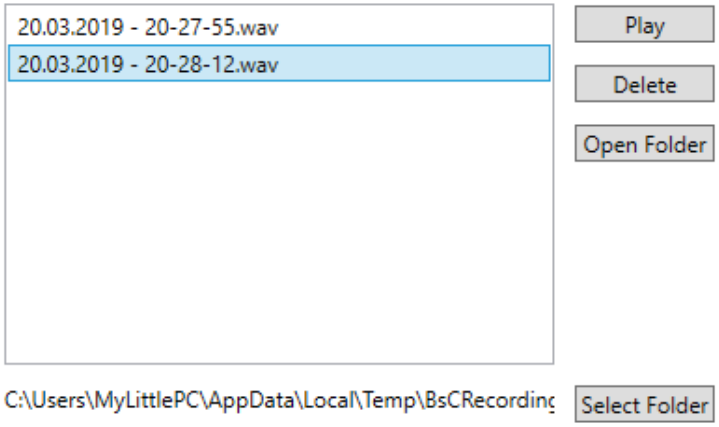
Przed próbą zapisania ścieżki dźwiękowej, użytkownik powinien upewnić się, że posiada zainstalowane najnowsze wersje sterowników do swojej karty dźwiękowej. W systemie Windows 7, 8 i 10 wszelkie niezbędne sterowniki są instalowane w sposób automatyczny wraz z aktualizacjami. Kluczowym jest również posiadanie wejścia mikrofonowego typu minijack. W przypadku radiostacji SAILOR RE 2100, niezbędny okazał się rozgałęźnik sygnału wpinany w port COM stacji. Dzięki temu możliwe stał się nasłuch odbieranego przez radiostację sygnału. Po tym wszystkim, użytkownik jest gotowy uruchomić aplikację. W sekcji, po prawej stronie na górze, znajduje się wybór urządzenia, z którego program będzie prowadził nasłuch. Częstotliwość próbkowania (wyrażona w Hz), ilość bitów głębi oraz liczba kanałów zostanie pobrana w sposób automatyczny, z aktualnych ustawień sterownika. Pola celowo nie zostały zablokowane, aby użytkownik był w stanie dokonać ewentualnej korekty. Należy jednak pamiętać, aby przetestować każde ustawienie przyciskiem „Test", ponieważ nieodpowiednia konfiguracja spowoduje

pojawienie się błędu w czasie nagrywania. Typ próbkowania, a dokładniej wykorzystywany do tego kodek systemowy jest jeden – „IEEE Float" oparty na standardzie reprezentacji binarnej IEEE 754. Liczbę binarną zapisuje się tak, że pierwsze 23 bity od prawej (pola od 0 do 22) to mantysa M, czyli binarna reprezentacja liczby ułamkowej. Następne 8 bitów (pola od 23 do 30) to wykładnik E, a ostatni 31. bit to znak S. Dla liczb dodatnich znak przyjmuje wartość zero, a dla ujemnych jest to jeden. Wartość pola limitu czasu określa po ilu sekundach nieaktywności, mierzonych od ostatniego przekroczenia suwaka szczytu (*peak*) przez kolumnę głośności (zielona), program powinien zapisać nagranie do pliku. Aplikacja rozpocznie nagrywanie kolejnego nagrania, niezwłocznie, po następnym przekroczeniu przez kolumnę głośności suwaka szczytu. W celu określenia właściwego położenia suwaka szczytu dla naszej konfiguracji sprzętowej, w sukurs przychodzi tryb testu, który pozwala na uruchomienie przechwytywania z pominięciem zapisu do pliku.

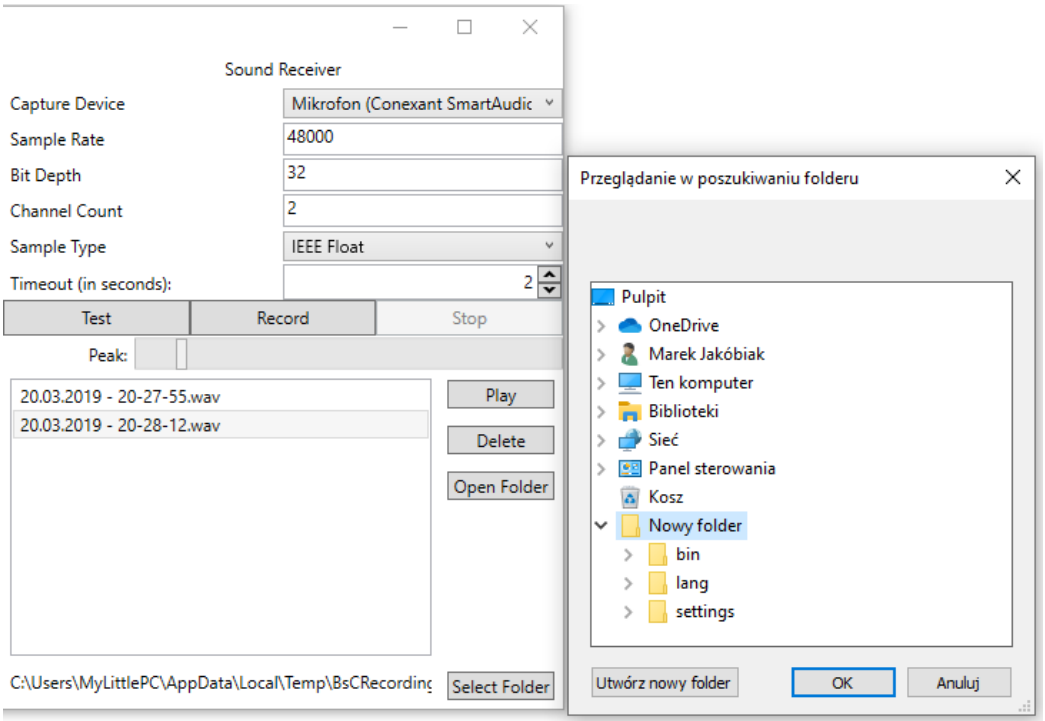


Rys. 3.5 Moment przekroczenia suwaka szczytu przez kolumnę głośności w trybie testowym. Źródło: Opracowanie własne

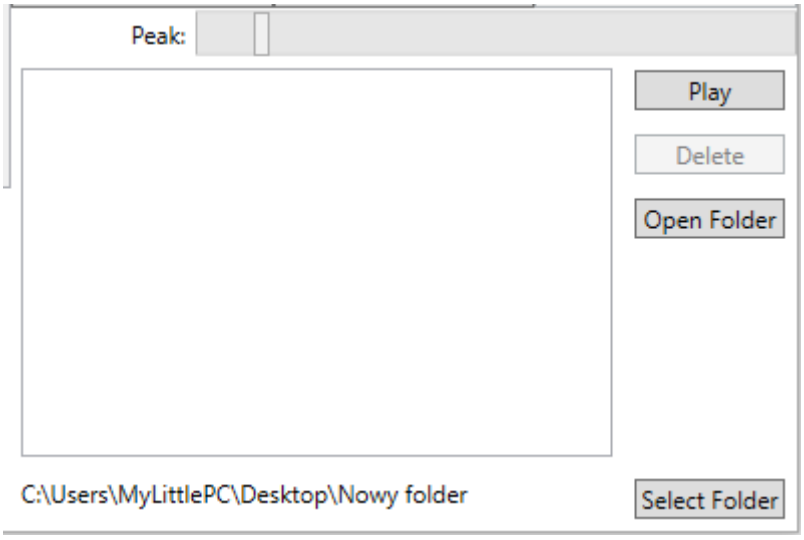
Pod kolumną głośności znajduje się pole wyboru plików, które jest odświeżane na bieżąco. Folder zapisu, użytkownik, może wybrać używając dedykowanego przycisku *Select Folder* umieszczonego w prawym, dolnym, rogu okna. Wybrana ścieżka wyświetlana jest za pomocą etykiety (*label*) obok przycisku. Domyślny katalog znajduje się w *%Temp%\BsCRecording*. Został on wybrany ze względu na brak wymogu uprawnień do zapisu tam plików w systemach Windows. Wybrany katalog można otworzyć przyciskiem *Open Folder* po prawej stronie. Wybrany plik z nagraniem użytkownik jest w stanie usunąć przyciskiem *Delete*. Umożliwione zostało odtworzenie zapisanego nagrania. W tym celu należy wcisnąć *Play* - otworzy się domyślny, dla danego systemu, odtwarzacz.



Rys. 3.6 Pole wyboru plików ścieżki dźwiękowej, wraz z dwoma ścieżkami. Źródło: Opracowanie własne



Rys. 3.7 Okno wyboru folderu do zapisu nagrań z aplikacji. Źródło: Opracowanie własne



Rys. 3.8 Po zmianie katalogu docelowego, wyświetlana zawartość katalogów uległa odświeżeniu. Źródło: Opracowanie własne

3.4.1 Implementacja kontrolki przetwarzania dźwięku

Jako szkielet kontrolki wykorzystano `Grid` 10x2. Ma on postać dziewięciu wierszy o wysokości 25px i jednego o wysokości dynamicznej oraz dwie kolumny wypełniające sobą całą dostępną szerokość:

Listing 3.2. Układ siatki w kontrolce SoundReceiverView

```
<Grid.RowDefinitions>
     <RowDefinition Height="25"/>
     <RowDefinition Height="25"/>
     <RowDefinition Height="25"/>
     <RowDefinition Height="25"/>
     <RowDefinition Height="25"/>
     <RowDefinition Height="25"/>
     <RowDefinition Height="25"/>
     <RowDefinition Height="25"/>
     <RowDefinition Height="25"/>
     <RowDefinition Height="*"/>
</Grid.RowDefinitions>
<Grid.ColumnDefinitions>
     <ColumnDefinition Width="*"/>
     <ColumnDefinition Width="*"/>
</Grid.ColumnDefinitions>
```

W każdym z wierszy znajduje się jedna z trzech kontrolek:

- ComboBox:

```
<ComboBox Grid.Row="1" ItemsSource="{Binding CaptureDevices}"
SelectedItem="{Binding SelectedDevice, Mode=TwoWay}"
VerticalAlignment="Center"/>
```

- TextBlock:

```
<TextBlock HorizontalAlignment="Left" Grid.Row="2" TextWrapping="Wrap"
Text="Sample Rate" VerticalAlignment="Center" Margin="5,0,0,0"/>
```

- TextBox:

```
<TextBox Height="23" Grid.Row="3" Grid.Column="1" TextWrapping="Wrap"
Text="{Binding BitDepth, Mode=TwoWay}" IsEnabled="{Binding
IsBitDepthConfigurable}" VerticalAlignment="Center"/>
```

Właściwości `Row` i `Column` z klasy `Grid` określają numer rzędu oraz kolumny w `Gridzie` (iterowane od 0). `ItemsSource` odpowiada za źródło danych do wyświetlenia (najczęściej jest to `Binding`), a `SelectedItem` pozwala na dwustronne (`TwoWay`) powiązanie danych z kontrolki z danymi w kodzie. `Horizontal` i `Vertical Alignment` określają w jaki sposób ułożyć wnętrze komórek. Działanie właściwości `Margin` jest analogiczne do jej odpowiednika w języku HTML, a więc odpowiada za szerokość marginesu. W praktyce oznacza to odstęp w pikselach (kolejno: lewa, góra, prawo, dół), jaki powinna mieć zawartość od krawędzi komórki. Ostatnimi trzema kontrolkami zastosowanymi tutaj są `Button` i `FileManagerView`.

- Button:

```
<Button Grid.Column="0" Content="Test" Width="Auto" Command="{Binding
TestCommand}"/>
```

- FileManagerView:

```
<local:FileManagerView Grid.Row="9" Grid.ColumnSpan="2"
DataContext="{StaticResource ViewModel}"/>
```

`SingleUpDown` z przestrzeni nazw `xctk` to kontrolka z darmowej, dla użytku niekomercyjnego, biblioteki o nazwie Extended WPF Toolkit. Właściwość `Buttona` o nazwie Content pozwala na ustawienie tekstu przycisku. Warto zauważyć, że `FileManagerView` posiada `StaticResource`. Jest to statyczny zasób, do którego odwołujemy się poprzez lokalną (`local`) przestrzeń nazw. Dzięki czemu jesteśmy w stanie użyć jednego widoku w wielu miejscach. `FileManagerView` jest kontrolką będącą w rzeczywistości uproszczonym menadżerem plików i znajduje się również w kolumnie odpowiedzialnej za odbiór danych z radiostacji.

3.4.2 Implementacja logiki funkcjonalności

Zgodnie ze wzorcem MVVM, logika kontrolki została umieszczona w klasie `SoundReceiverViewModel`, która komunikuje się z widokiem za pomocą bindingów. Na samym początku klasy znajdują się instrukcje określające używane przez klasę przestrzenie nazw.

Listing 3.3. Instrukcje wykorzystujące przestrzeń nazw

```
using NAudio.CoreAudioApi;
using NAudio.Wave;
using System;
using System.Collections.ObjectModel;
using System.IO;
using System.Linq;
using System.Windows;
using BSc_Thesis.Models;
```

Oprócz składników systemowych, znajduje się tutaj również referencja do biblioteki NAudio, oraz do modeli (przestrzeń nazw `BSc_Thesis.Models`). Zgodnie z rys. 3.2, klasa dziedziczy po `FileManagerViewModel`. Dzięki temu jest w stanie udostępniać dane managerowi plików. Po tym następują pola klasy (`#region Fields`). Została przyjęta zasada o umieszczaniu pól klasy przed ich właściwościami. Innymi słowy, są to deklaracje zmiennych użytych w kodzie:

Listing 3.4. Region Fields zawierający pola klasy SoundReceiverViewModel

```
#region Fields
private WasapiCapture capture;
private WaveFileWriter writer;
private DateTime startDT;
private bool isRecording = false;
private string currentFileName;
private float timeout = 2.0f;
private MMDevice selectedDevice;
private SoundData sd = new SoundData();
#endregion
```

Obiekt klasy `WasapiCapture` służy do przechwytywania dźwięku z użyciem Windows Audio Session API (WASAPI). Jest to interfejs programowania aplikacji udostępniony przez Microsoft wraz z premierą systemu operacyjnego Windows Vista. Tutaj też przypisywany jest czas (domyślnie w sekundach), po którym zostanie przerwane nagrywanie do pliku .wmv.

Właściwości w ViewModelu służą obsłudze Bindingów. Region publicznych obiektów otwierają cztery gettery, z czego wyróżniają się dwa:

```
public DelegateCommand RecordCommand { get; }
public ObservableCollection<MMDevice> CaptureDevices { get; }
```

`DelegateCommand` to klasa, której zadaniem jest przypisanie metody do zdarzenia naciśnięcia przycisku w widoku (właściwość `Command="{Binding RecordCommand}"`) przycisku. Dzięki temu, przypisana metoda zostanie wykonana asynchronicznie w osobnym wątku, nie zamrażając tym samym interfejsu użytkownika.

`ObservableCollection` jest klasą wbudowaną w podstawową przestrzeń nazw .NET o nazwie System. Różni się od zwykłej kolekcji tym, że może być obserwowana. Oznacza to, że jakakolwiek zmiana w jej obrębie informuje o tym wszystkich obserwatorów. Bindując obiekt tej klasy do np. właściwości `ItemsSource` w ComboBoxie

(`ItemsSource="{Binding CaptureDevices}"`), otrzymamy listę obiektów aktualizowaną na bieżąco, bez strat na responsywności interfejsu użytkownika.

Reszta właściwości jest bliźniaczo podobna do siebie i wygląda w następujący sposób:

Listing 3.5. Przykładowa publiczna właściwość typu get/set

```
public float Timeout {
  get => timeout;
  set {
    if (timeout != value) {
      timeout = value;
      OnPropertyChanged();
    }
  }
}
```

Get zwraca identyczne, ale prywatne pole, a set przypisuje nową wartość do tego pola. Przypisanie następuje tylko wtedy, gdy nowa dana różni się od obecnie przechowywanej. Po zmianie przechowywanej zmiennej, trzeba poinformować o tym interfejs. Robi się to poprzez wykonanie metody `OnPropertyChanged` informującej interfejs użytkownika o dokonanej zmianie. Metoda ta została odziedziczona z klasy `ViewModelBase` poprzez `FileManagerViewModel` i wygląda następująco:

Listing 3.6. Implementacja metody INotifyPropertyChanged

```
class ViewModelBase : INotifyPropertyChanged
{
  public event PropertyChangedEventHandler PropertyChanged;
  protected virtual void OnPropertyChanged([CallerMemberName]string propertyName = "")
  {
    PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
  }
}
```

Pole zdarzenia `PropertyChanged` z interfejsu `INotifyPropertyChanged`, zostało przykryte w taki sposób, aby mogło przyjąć nazwę właściwości, która wywołała metodę `OnPropertyChanged`.

Po właściwościach typu get/set następuje konstruktor klasy:

Listing 3.7. Implementacja konstruktora klasy SoundReceiverViewModel

```
public SoundReceiverViewModel() : base(FileExtension.Wav)
{
  var enumerator = new MMDeviceEnumerator();
  var     defaultDevice     =     enumerator.GetDefaultAudioEndpoint(DataFlow.Capture,
Role.Console);
  CaptureDevices                                  =                                    new
ObservableCollection<MMDevice>(enumerator.EnumerateAudioEndPoints(DataFlow.All,
DeviceState.Active).AsEnumerable());
  SelectedDevice = CaptureDevices.FirstOrDefault(c => c.ID == defaultDevice.ID);
  RecordCommand = new DelegateCommand(Record);
  StopCommand = new DelegateCommand(Stop) { IsEnabled = false };
  TestCommand = new DelegateCommand(Test);
  startDT = DateTime.Now;
}
```

Na początku zostaje przypisane konstruktorowi klasy bazowej (`FileManagerViewModel`) odpowiednie rozszerzenie plików. Klasa odpowiadająca za zarządzanie plikami wymaga tego, aby wiedzieć na jakim formacie będzie pracować. Przekazywany typ wyliczeniowy enum `FileExtension` składa się z dwóch wartości – „Txt" i „Wav", odpowiadającym swoim rozszerzeniom. Powyższy konstruktor służy przede wszystkim do wylistowania wszystkich dostępnych urządzeń audio zdolnych do przechwycenia dźwięku. Używany jest do tego Enumerator dostępny w bibliotece NAudio. Dla ułatwienia konfiguracji aplikacji, jako urządzenie startowe, wybieram zawsze te domyślne w systemie. Na końcu następuje przypisanie funkcji do zdarzeń zbindowanych z widokiem (pole `Command` w Buttonie). Na końcu konstruktora do zmiennej `startDT` ustawiany jest aktualny czas systemowy.

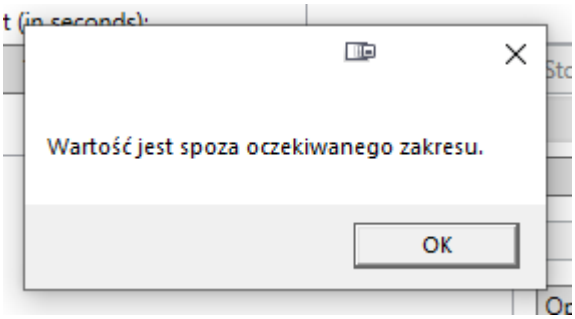
Po konstruktorze występują funkcje prywatne zawierające logikę omawianej funkcjonalności. Dla zwiększenia przejrzystości niniejszej pracy pominięto opis funkcji pomocniczych, które jej nie zawierają. Sercem przechwytywania dźwięku z radiostacji są dwie metody, `startCapturing` i `CaptureOnDataAvailable`:

Listing 3.8. Implementacja metody startCapturing

```
private void startCapturing(bool isTest = false)
{
  try {
    if (SelectedDevice.DataFlow == DataFlow.Capture) {
      capture = new WasapiCapture(SelectedDevice);
      capture.WaveFormat      =      WaveFormat.CreateIeeeFloatWaveFormat(SampleRate,
ChannelCount);
    } else
      capture = new WasapiLoopbackCapture(SelectedDevice);
    capture.ShareMode   =   ShareModeIndex   ==   0   ?   AudioClientShareMode.Shared   :
AudioClientShareMode.Exclusive;
    RecordLevel = SelectedDevice.AudioEndpointVolume.MasterVolumeLevelScalar;
    capture.StartRecording();
    capture.RecordingStopped += OnRecordingStopped;
    if (isTest) {
      capture.DataAvailable += TestCaptureOnDataAvailable;
    } else {
      capture.DataAvailable += CaptureOnDataAvailable;
    }
    RecordCommand.IsEnabled = false;
    TestCommand.IsEnabled = false;
    StopCommand.IsEnabled = true;
  } catch (Exception e) {
    MessageBox.Show(e.Message);
  }
}
```

Metoda `startCapturing` jest wywoływana przez `Record()` i `Test()`, przypisane, z wykorzystaniem bindingów, do odpowiednich przycisków na interfejsie użytkownika. Proces rozpoczyna się od stworzenia i przypisania właściwego obiektu zmiennej `capture`, przez co możliwa staje się również obsługa zapisu dźwięku z linii głośnika. Mikrofon wymaga użycia klasy `WasapiCapture`, a głośnik `WasapiLoopbackCapture`. Kolejno

następuje sczytanie głośności urządzenia, włączenie nagrywania i przypisanie odpowiednich metod do właściwych wydarzeń. Odpowiadają one za przepełnienie bufora oraz zakończenie nagrywania. Wykorzystano tutaj instrukcję `try-catch` w celu zmniejszenia wpływu ewentualnych błędów procesu nagrywania na działanie programu. Użytkownik zostaje poinformowany o zaistniałych błędach informacją w MessageBoxie.



Rys. 3.9 Przykładowy MessageBox z błędem w systemie Windows 10, wynikły z nieprawidłowej konfiguracji nagrywania. Źródło: Opracowanie własne.

W momencie zapełnienia bufora zostaje wywołane wydarzenie `DataAvailable` z obiektu `capture`. Przypisana wydarzeniu metoda zależy od tego, czy użytkownik nacisnął przycisk „Test", czy może jednak „Play". W pierwszym przypadku do zmiennej `Peak`, będącej źródłem danych dla paska „Peak" z interfejsu użytkownika, przypisujemy maksimum z próbki z bufora. Odbywa się to w następujący sposób:

Listing 3.9. Implementacja metody GetMaximumSample

```
private float getMaximumSample(WaveInEventArgs args)
{
  WaveBuffer buffer = new WaveBuffer(args.Buffer);
  float max = 0;
  for (int index = 0; index < args.BytesRecorded / 4; index++) {
    var sample = buffer.FloatBuffer[index];
    if (sample < 0)
      sample = -sample;
    if (sample > max)
      max = sample;
  }
  return max;
}
```

Bufor wyrażany jest w bajtach i można go podzielić na 4-bajtowe części. W języku C#, taką wielkość posiadają liczby zmiennoprzecinkowe typu (`float`). Z tego powodu każda część zostaje niejawnie rzutowana na float. Stąd też iterator pętli nie może przekroczyć ilości nagranych bajtów (`BytesRecorded`) podzielonych przez 4. Otrzymany zapis dźwięku ma charakter sinusoidalny. W celu znalezienia jego maksimum, należy najpierw wyznaczyć wartości bezwzględne ze wszystkich próbek sygnału. Następnie, poprzez porównanie kolejnych próbek, wyznacza się maksymalną wartość sinusoidy.

Złożoność operacji ulega zwiększeniu, gdy nie testujemy źródła dźwięku:

Listing 3.10. Implementacja metody CaptureOnDataAvailable

```
private void CaptureOnDataAvailable(object sender, WaveInEventArgs args)
{
  if ((DateTime.Now - startDT).Seconds > Timeout && isRecording == true) {
    dumpFile();
    isRecording = false;
  }
  float max = getMaximumSample(args);
  if (max >= PeakLevel) {
    isRecording = true;
    startDT = DateTime.Now;
    if (writer == null) {
      currentFileName = String.Format("{0:dd.MM.yyy - HH-mm-ss}.wav", DateTime.Now);
      writer = new WaveFileWriter(Path.Combine(OutputFolder, currentFileName),
capture.WaveFormat);
    }
  }
  if (writer != null)
    writer.Write(args.Buffer, 0, args.BytesRecorded);
  Peak = max;
}
```

Aby tego dokonać, należy wpierw sprawdzić, czy nie upłynął ustalony przez użytkownika czas od ostatniego zarejestrowanego dźwięku. Jeżeli upłynął, plik zostaje zapisany, a pisarz usunięty (metoda `dumpFile()` przykrywa `writer.Dispose()`). Po przeanalizowaniu próbki w celu otrzymania jej maksimum, porównujemy najwyższą wartość z ustalonym przez użytkownika aplikacji limitem. Gdy jest większy lub równy to włączamy nagrywanie i odświeżamy czas startu. Bufor na bieżąco zapisywany jest do pliku, w nazwie którego zakodowane są data i czas z dokładnością co do sekundy. Na samym końcu uaktualniana jest zmienna będąca źródłem danych dla interaktywnego paska „Peak” z interfejsu użytkownika.

Po wciśnięciu przycisku *Stop*, nagrywanie jest zatrzymywane, a interfejs zerowany. Zarezerwowana pamięć zostaje zwolniona przez garbage collectora – przycisk *Stop* wyzwala, poprzez odpowiednie zdarzenie, metodę `Stop()`, a `capture?.StopRecording()` wyzwala zdarzenie `RecordingStopped`, które wcześniej przypisano w `startCapturing(bool isTest = false)`:

Listing 3.11. Implementacja metody OnRecordingStopped

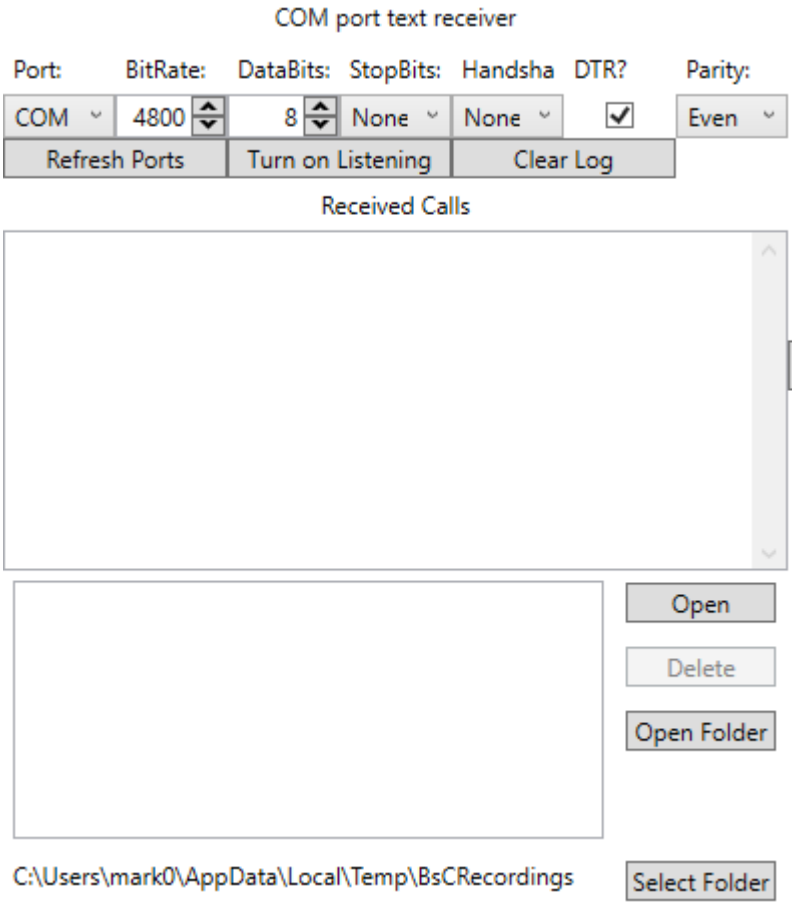
```
void OnRecordingStopped(object sender, StoppedEventArgs e)
{
  if (writer != null) {
    writer.Dispose();
    writer = null;
  }
  capture.Dispose();
  capture = null;
}
private void Stop()
{
  capture?.StopRecording();
  RecordCommand.IsEnabled = true;
  StopCommand.IsEnabled = false;
  TestCommand.IsEnabled = true;
  Peak = 0.0F;
}
```

3.4.3 Podsumowanie

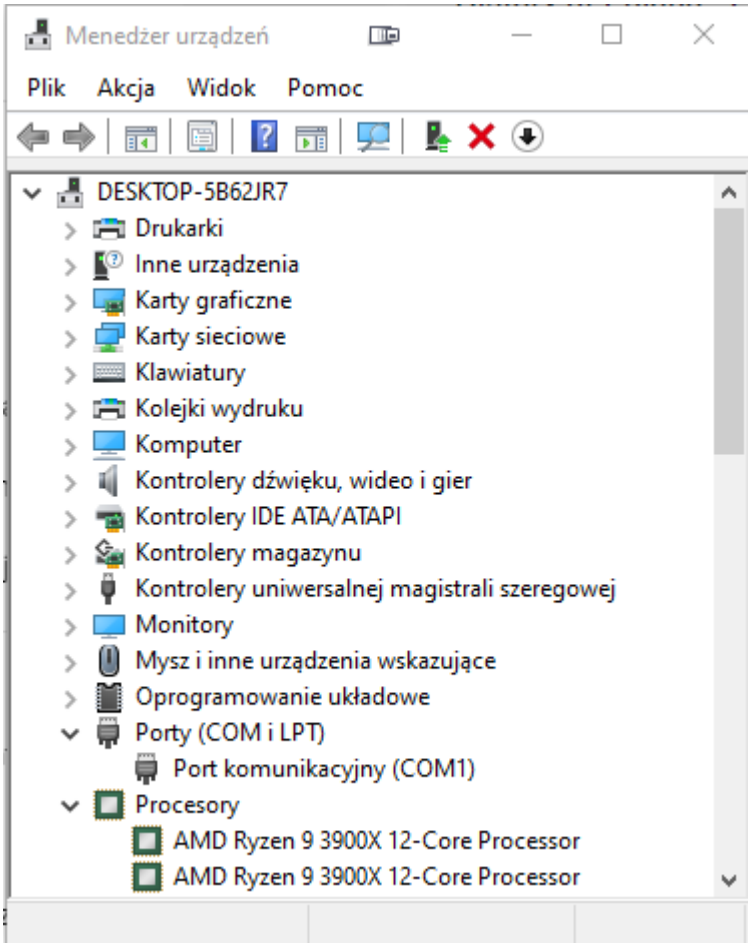
Udało się w pełni zaprojektować oraz zaimplementować funkcjonalność umożliwiającą rejestrację i zapis dźwięków odbieranych z radiostacji MF/HF. Wybór C# jako języka implementacji okazał się w pełni uzasadniony. Język ten ma dostępny szereg bibliotek, rozszerzający go o funkcjonalności umożlwiające łatwe przechwycenie wymaganych danych. Największą trudność sprawiła odpowiednia interpretacja i zapis otrzymywanych danych. Pomocna okazała się wiedza z teorii fal, wyniesiona z zajęć elektrotechniki, oraz właściwość plików *.wav umożliwiająca niemal nieskończone dopisywanie kolejnych danych do końca pliku z wykorzystaniem metody `append()`.

3.5 Odbiór danych z radiostacji z wykorzystaniem portu COM

Do rozpoczęcia odbierania wiadomości z radiostacji SAILOR RE 2100 wymagany jest port COM. Niestety, postęp technologiczny spowodował odrzucenie portu COM na rzecz mniejszego i wydajniejszego USB. W związku z powyższym, do symulacji rzeczywistych warunków pracy radiostacji oraz odbieranych komunikatów, połączono ze sobą dwa komputery z wykorzystaniem dwóch adapterów z portu USB 2.0 AM do portu COM M. Po podłączeniu adaptera w Menadżerze urządzeń w systemie Windows pojawia się port COMn, gdzie n to numer portu (rys. 3.10). Należy go zapamiętać, ponieważ będzie niezbędny przy konfiguracji aplikacji, w celu prowadzenia nasłuchu informacji przesyłanych z radiostacji za pomocą portu COM.



Rys. 3.10 Część interfejsu użytkownika w aplikacji odpowiadająca za odbiór danych z radiostacji z wykorzystaniem portu COM. Źródło: Opracowanie własne



Rys. 3.11 Menadżer urządzeń systemu Windows 10 z widocznym, zainstalowanym, portem COM1. Źródło: opracowanie własne

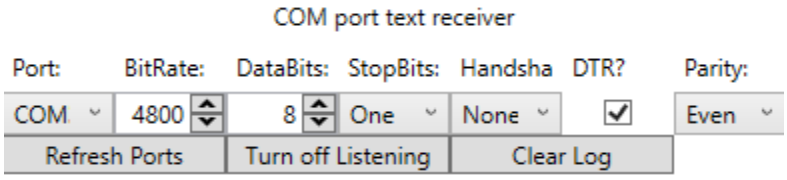
W celu odbioru danych z radiostacji należy, po włączeniu aplikacji i połączeniu radiostacji z komputerem używając przewodu COM, skonfigurować odpowiednio podstawowe parametry transmisji szeregowej. Parametrami niezbędnymi do odbioru są:

- Port – numer portu COMn, na którym aplikacja ma prowadzić nasłuch,
- BitRate - szybkość transmisji danych w bitach na sekundę,
- DataBits - ilość bitów na bajt,
- StopBits - ilość bitów stopu, przyjmuje wartości typu enum: None, One, OnePointFive i Two,
- Handshake - sposób sterowania transmisją używany do nawiązywania połączenia po porcie COM. Jest to enum przyjmujący wartości:
  - None - wyłącza sterowanie transmisją,
  - RequestToSend - gdy kontrola znajduje się po stronie sprzętu. RTS sygnalizuje dostępność danych do transmisji. W momencie, gdy bufor wejściowy po stronie klienta ulegnie wypełnieniu, flaga RTS zostanie

ustawiona na false, aż do momentu, gdy w buforze zostanie zrobione miejsce,

- RequestToSendXOnXOff - wykorzystuje jednocześnie kontrolę sprzętową (RTS) oraz programową (XON/XOFF),
- XOnXOff - umożliwia zastosowanie protokołu XON/XOFF polegającym na wysyłaniu bitu kontroli XOFF w celu zatrzymania transmisji i XON, aby wznowić transmisję,

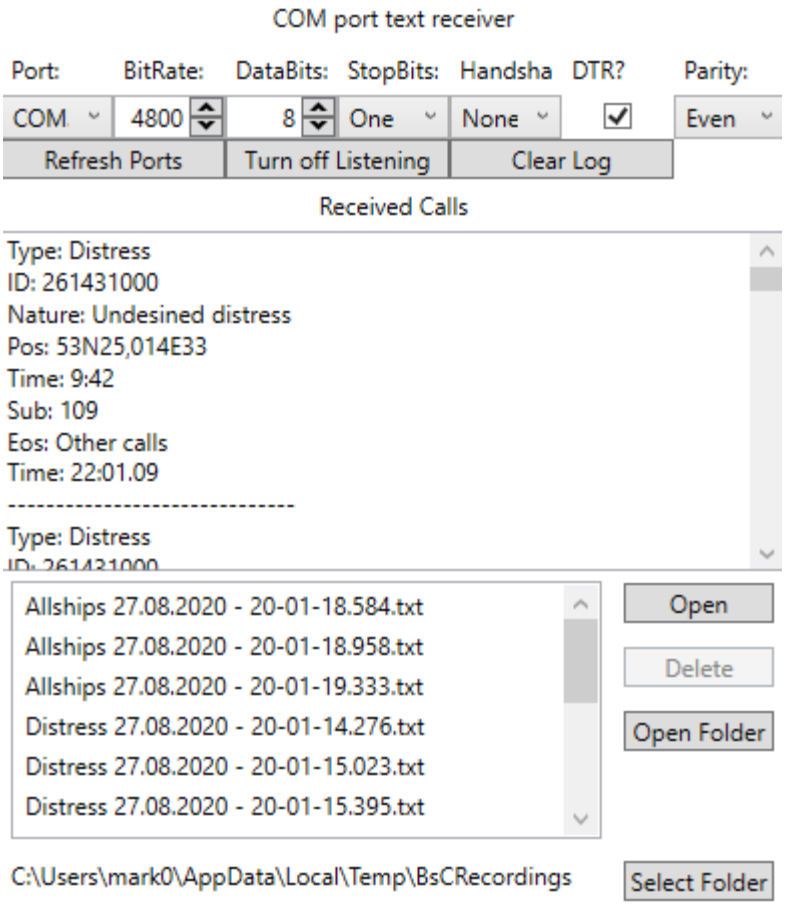
- DTR - sygnał gotowości Data Terminal Ready wysyłany w czasie transmisji szeregowej,
- Parity - włączający kontrolę parzystości oraz określający bit parzystości. Jest enumem przyjmującym wartości:
  - Even - ustawiający bit parzystości tak, że licznik zestawów bitów jest liczbą parzystą,
  - Mark - bit parzystości odpowiadający logicznej jedynce,
  - None - wyłączający kontrolę parzystości,
  - Odd - ustawiający bit parzystości tak, że licznik zestawów bitów jest liczbą nieparzystą,
  - Space - bit parzystości odpowiadający logicznemu zeru.



Rys. 3.12 Przykładowa konfiguracja odbioru transmisji na porcie COM, zgodna z radiostacją SAILOR RE 2100. Źródło: opracowanie własne

Przyciski pod konfiguracją służą kolejno (od lewej) do odświeżenia listy dostępnych portów, włączania/wyłączania nasłuchu na porcie oraz czyszczenia logu transmisji.

Po włączeniu nasłuchu odpowiedniego portu COM, aplikacja zaczyna odbierać dane, na których dokonuje analizy składniowej, tym samym nadając im format czytelny dla człowieka. Wszelkie otrzymane sygnały są wyświetlane w przewijanym logu aplikacji oraz zapisywane w wybranym folderze, w formacie .txt. Folder zapisu można wybrać przyciskiem *Select Folder* w prawym, dolnym, rogu kontrolki. Domyślną ścieżką jest *%temp%/BsCRecordings*, ponieważ do zapisu w tym folderze w systemie Windows nie są potrzebne żadne dodatkowe uprawnienia.



Rys. 3.13 Odbiór informacji z radiostacji SAILOR RE 2100. Źródło: opracowanie własne

3.5.1 Implementacja kontrolki odbioru danych

Kontrolka składa się z trzech wierszy, gdzie wysokość dwóch pierwszych jest ustawiona na `Auto`, a trzeciej na `*`:

Listing 3.12. Układ wierszy w siatce kontrolki ComCaptureView

```
<Grid.RowDefinitions>
     <RowDefinition Height="Auto"/>
     <RowDefinition Height="Auto"/>
     <RowDefinition Height="*"/>
</Grid.RowDefinitions>
```

Różnica pomiędzy `Auto`, a `*` jest taka, że pierwsze z nich dostosowuje wysokość do zawartości wiersza, a asterix zajmuje pozostałe miejsce. Kolejno następuje definicja etykiety znajdującej się w pierwszym wierszu:

```
<Label HorizontalAlignment="Center" Grid.Row="0" Content="COM port text
receiver"/>
```

Właściwość `HorizontalAlignment` określa poziome rozmieszczenie etykiety względem zawartości wiersza. Ustawienie jej na `Center` powoduje wyśrodkowanie etykiety. `Grid.Row` precyzuje kolumnę (iterowane od zera), a `Content` zawartość tekstową.

Następna jest definicja wiersza numer jeden, czyli drugiego od góry. Odwołanie do kolumny następuje poprzez otwarcie etykiety `<Grid Grid.Row="1">`. Wiersze zostały zdefiniowane w sposób następujący:

Listing 3.13. Układ siatki pierwszego wiersza kontrolki ComCaptureView

```
<Grid.RowDefinitions>
      <RowDefinition Height="Auto"/>
      <RowDefinition Height="Auto"/>
      <RowDefinition Height="Auto"/>
</Grid.RowDefinitions>
<Grid.ColumnDefinitions>
      <ColumnDefinition Width="*"/>
      <ColumnDefinition Width="*"/>
      <ColumnDefinition Width="*"/>
      <ColumnDefinition Width="*"/>
      <ColumnDefinition Width="*"/>
      <ColumnDefinition Width="*"/>
      <ColumnDefinition Width="*"/>
</Grid.ColumnDefinitions>
```

Każdy z trzech rzędów dostosowuje swoją wysokość do zawartości, dzięki czemu powiększenie jakiejkolwiek kontrolki spowoduje automatyczne pomniejszenie kontrolek w pozostałych rzędach. Jest również siedem równych kolumn, tyle ile właściwości posiada klasa `SerialPort` z systemowej przestrzeni nazw `System.IO.Ports`.

Po definicji wierszy i kolumn następuje deklaracja etykiet z pierwszego rzędu. Odbywa się to analogicznie do definicji etykiety z początku niniejszej kontrolki. Dodatkowo użytą właściwością jest `Grid.Column`, która wskazuje na kolumnę, w której znajduje się dana etykieta. Tak jak rzędy, kolumny również są numerowane od zera. Następna jest deklaracja ComboBoxa:

```
<ComboBox SelectedIndex="0" SelectedValue="{Binding PortName}"
ItemsSource="{Binding PortNames}" Grid.Row="1"/>
```

Użyte zostały tutaj dwie, wcześniej nieomawiane, właściwości. `SelectedIndex` służy do zdefiniowania wybranego elementu listy (iterowane od zera) w momencie inicjalizacji ComboBoxa. Dzięki temu, przy uruchomieniu aplikacji zostanie wybrany pierwszy element listy. Drugą właściwością jest `SelectedValue` zawierający `Binding PortName`. Służy on dwukierunkowej wymianie informacji pomiędzy interfejsem użytkownika, a aplikacją. Dzięki temu możliwe jest odczytanie przez aplikację aktualnie wybranego elementu listy oraz nadpisanie tegoż wyboru w momencie odświeżenia listy.

Następnym elementem wiersza jest `IntegerUpDown` z przestrzeni nazw `xctk` (Extended WPF Toolkit):

```
<xctk:IntegerUpDown Value="{Binding PortBitRate}" Grid.Column="1"
Grid.Row="1" PreviewTextInput="myUpDownControl_PreviewTextInput"/>
```

Zastosowana została właściwość `PreviewTextInput` filtrująca wprowadzany do kontrolki tekst. Zabieg ten nie dopuszcza do wprowadzenia do kontrolki innych wartości niż te numeryczne. Walidator został zdefiniowany w code behind, a jego działanie oparte jest na wyrażeniach regularnych (`Regex`):

Listing 3.14. Walidator pola BitRate

```
static readonly Regex _regex = new Regex("[^0-9.-]+");

private void myUpDownControl_PreviewTextInput(object sender, TextCompositionEventArgs
e)
{
  e.Handled = _regex.IsMatch(e.Text);
}
```

Każde ręczne wprowadzenie znaku do kontrolki `IntegerUpDown` wywołuje zdarzenie (*event*), które zawiera w sobie wprowadzany tekst. Następnie wprowadzony tekst jest dopasowywany do obiektu klasy `Regex`. Wynik jest przypisywany właściwości `Handled`, dzięki czemu interfejs zostaje powiadomiony o obsłużeniu zdarzenia i jego wyniku.

```
<CheckBox HorizontalAlignment="Center" VerticalAlignment="Center"
Grid.Column="5" Grid.Row="1" IsChecked="{Binding IsDtr}"/>
```

Zastosowany został również `CheckBox` z właściwością `IsChecked` zawierającą `Binding IsDtr` do zmiennej typu `bool`. Właściwość ta przyjmuje wartości `true` lub `false` odzwierciedlające aktualny stan pola wyboru.

Listing 3.15. Układ siatki drugiego wiersza kontrolki ComCaptureView

```
<Grid Grid.Row="2">
  <Grid.RowDefinitions>
    <RowDefinition Height="Auto"/>
    <RowDefinition Height="2*"/>
    <RowDefinition Height="2*"/>
  </Grid.RowDefinitions>
  <Label Grid.Row="0" Content="Received Calls" HorizontalContentAlignment="Center"
HorizontalAlignment="Center"/>
  <TextBox ScrollViewer.VerticalScrollBarVisibility="Visible" Grid.Row="1"
HorizontalAlignment="Stretch" Text="{Binding ReceivedCalls}"/>
  <local:FileManagerView Grid.Row="2" DataContext="{StaticResource ViewModel}"/>
</Grid>
```

Następny wiersz składa się z trzech pomniejszych wierszy. Wysokość pierwszego z nich została ustawiona na `Auto`, a dwóch pozostałych wynosi `2*`. W pierwszym z nich znajduje się wyśrodkowana (właściwości `HorizontalContentAlignment` i `HorizontalAlignment`) etykieta. Pod nią występuje duży `TextBox` z właściwością

uwidaczniającą pasek służący do przewijania tekstu o nazwie `ScrollViewer.VerticalScrollBarVisibility`. `HorizontalAlignment` ustawiony na `Stretch` rozciąga `TextBox` na całą dostępną szerokość wiersza. Ostatnim elementem jest kontrolka `FileManagerView`, która została omówiona w poprzednim rozdziale.

3.5.2 Implementacja logiki funkcjonalności

Odbiór danych z radiostacji obsługiwany jest z wykorzystaniem klasy `ComCaptureViewModel`. Dziedziczy ona po modelu `FileManagerViewModel` (rys. 3.2) odpowiedzialnym za menadżer plików w dolnej części kontrolki. Podobnie jak w przypadku `SoundReceiverViewModel`, ta klasa również rozpoczyna się od deklaracji pól, właściwości oraz „getterów' i „seterów".

Listing 3.16. Implementacja konstruktora klasy ComCaptureViewModel

```
public ComCaptureViewModel() : base(FileExtension.Txt)
{
  port = new Port();
  OutputFolder = Path.Combine(Path.GetTempPath(), "BsC_Recordings");

  if (!Directory.Exists(OutputFolder))
  {
    Directory.CreateDirectory(OutputFolder);
  }

  RefreshPortsCommand = new DelegateCommand(refreshPorts);
  TurnListeningCommand = new DelegateCommand(turnListening);
  ClearLogCommand = new DelegateCommand(clearLog);
  resolverTimer = new Timer(50);
  resolverTimer.Elapsed += dataResolver;
  resolverTimer.AutoReset = true;
  refreshPorts();
}
```

Konstruktor `ComCaptureViewModel` rozpoczyna swoje działanie od przypisania nowego obiektu klasy `Port`. Model ten został utworzony w celu umożliwienia przechowywania niezbędnych danych opisujących port COM. Kolejno przypisywana jest ścieżka do folderu wyjściowego potrzebnego do zapisania przychodzących wezwań DSC. W przypadku, gdy folder nie istnieje zostaje on utworzony. W następnej kolejności bindingom zostają przypisane odpowiednie metody wewnątrz klasy. Stworzony ulega również nowy zegar obsługujący metodę `dataResolver`. Jest on wymagany do obsługi portu COM, ponieważ w momencie włączenia nasłuchu bufor zaczyna się powoli wypełniać napływającymi danymi. W celu niedopuszczenia do jego przepełnienia, potrzeba go regularnie obsługiwać. Wykonuje to stworzony uprzednio zegar, wykonujący metodę `dataResolver` co 50 ms. Konstruktor kończy pracę odświeżając listę portów. Metoda odpowiadająca za ich odświeżenie wygląda następująco:

Listing 3.17. Implementacja metody refreshPorts

```
private void refreshPorts()
{
  using (var searcher = new ManagementObjectSearcher("SELECT * FROM
WIN32_SerialPort")) {
    string[] portnames = SerialPort.GetPortNames();
    var x = searcher.Get().Cast<ManagementBaseObject>().ToList();
    PortNames = new ObservableCollection<string>((
      from n in portnames join p in x on n equals p["DeviceID"].ToString() into np
select n
    ));
  }
}
```

Cały proces rozpoczyna zapytanie SQL wybierające wszystkie elementy z tablicy `WIN32_SerialPort`. Przechowuje ona rekordy reprezentujące porty szeregowe w komputerze używającym systemu Windows [14]. Następnie pobrana zostaje tablica nazw portów z klasy `SerialPort`. Wcześniej uzyskany wynik zapytania jest rzutowana na obiekt klasy `ManagementBaseObject`, który w prosty sposób można przekonwertować w listę metodą `ToList()`. Otrzymana lista oraz tablica stringów reprezentująca nazwy portów jest porównywana, a zgodne pola dodawane do siebie. Otrzymujemy w ten sposób obiekt klasy `ObservableCollection`, w której znajdują się nazwy portów powiązane z ich `DeviceID`. Identyfikator ten jest unikalny w skali systemu, tym samym umożliwiając bezbłędną identyfikację portu. Tak więc, znajomość tej informacji jest niezbędna do uruchomienia nasłuchu portu:

Listing 3.18. Implementacja metody turnListening

```
private void turnListening()
{
  if (!IsPortActive) {
    try {
      if (PortName == null)
        throw new Exception("Please choose valid Port");
      sp.PortName = PortName;
      sp.BaudRate = PortBitRate;
      sp.Parity = (Parity) Enum.Parse(typeof(Parity), ParityValue);
      sp.DataBits = DataBits;
      sp.StopBits = (StopBits) Enum.Parse(typeof(StopBits), StopBitsValue);
      sp.Handshake = (Handshake) Enum.Parse(typeof(Handshake), HandshakeValue);
      sp.DtrEnable = IsDtr;
      sp.DataReceived += new SerialDataReceivedEventHandler(DataReceivedHandler);
      sp.Open();
      resolverTimer.Enabled = true;

    } catch (Exception e) {
      MessageBox.Show(e.Message);
    }
  } else if (sp.IsOpen) {
    sp.Close();
    resolverTimer.Enabled = false;
  }
  IsPortActive = sp.IsOpen;
}
```

Wykonanie metody `turnListening` zależne jest od stanu flagi `IsPortActive`. Określa ona status nasłuchu. W momencie, gdy został już uruchomiony (flaga

IsPortActive ustawiona na true) następuje zamknięcie portu, wyłączenie zegara resolverTimer i przypisanie stanu portu fladze. Napis na przycisku odpowiadającym za włączenie lub wyłączenie nasłuchu jest zależny od stanu flagi IsPortActive, dzięki czemu przycisk działa jak przełącznik dwubiegunowy. Przy próbie włączenia nasłuchu dokonana zostaje walidacja wprowadzonych danych. W przypadku niepoprawnych wartości, try-catch wyświetli MessageBox z wiadomością dołączoną do błędu. Tutaj też niezbędne okazało się zastosowanie rzutowania ciągów alfanumerycznych (string) na typ wyliczeniowy „enum". Na samym końcu następuje przypisanie metody obsługującej odbiór danych do wydarzenia DataReceived, otwarcie portu oraz włączenie zegara resolverTimer.

Listing 3.19. Implementacja metody DataReceivedHandler

```
private void DataReceivedHandler(object sender, SerialDataReceivedEventArgs e)
{
  SerialPort sp = (SerialPort) sender;
  string indata = sp.ReadExisting();
  comPortTemp += indata;
}
```

Metoda DataReceivedHandler jest prostą funkcją odbierającą otrzymane dane. Otrzymany obiekt o nazwie sender zostaje rzutowany na typ SerialPort, po czym odczytaniu ulega jego zawartość i przypisana do pola comPortTemp typu string. Pole to jest używane przez metodę dataResolver wykonywaną cyklicznie, gdy pole comPortTemp typu string posiada wartość.

Każda wiadomość otrzymana ze stacji SAILOR RE 2100 transmisją szeregową, rozpoczyna się słowem Incoming. Stąd też zawartość zmiennej comPortTemp zostaje przetestowana na obecność tegoż ciągu znaków. Do operacji został wykorzystany następujący Regex:

```
private Regex messageRegex = new Regex(@"Incoming[\w\W]+?> \?");
```

Listing 3.20. Implementacja metody dataResolver

```
private void dataResolver(Object source, ElapsedEventArgs e)
{
  while (comPortTemp != string.Empty) {
    var regexResult = messageRegex.Match(comPortTemp);
    if (!regexResult.Success)
      break;
    comPortTemp = comPortTemp.Substring(regexResult.Index + regexResult.Length + 1);
    string[] m = regexResult.Value.Replace("\r", "").Split('\n');
    string result = string.Empty;
    foreach (string s in m) {
      if (s != string.Empty && !s.Contains('>')) {
        if (s.Contains('=')) {
          var s2 = s.Replace(" ", "").Split('=');
          if (s2[0] == "Nature") {
            s2[1] = ddr.ResolveDistressCode(s2[1]);
          }
          if (s2[0] == "Eos") {
            s2[1] = ddr.ResolveEndOfSequence(s2[1]);
          }
          if (s2[0] == "Cat") {
            s2[1] = ddr.ResolveCategory(s2[1]);
          }
          if (s2[0] == "Pos") {
            s2[1] = ddr.ResolveCategory(s2[1]);
            string[] s3 = s2[1].Split(',');
            Services.MessengerHub.PublishAsync<GeoMessage>(new       GeoMessage(this,
s3[0], s3[1]));
          }
             result += s2[0] + ": " + s2[1] + '\n';
           } else {
            var s2 = s.Split(' ');
            if (s.Contains("Incoming")) {
              result += "Type: " + s2[1] + '\n';
              currentFileName += s2[1];
            } else {
              result += s2[0] + ": " + s2[1] + '\n';
            }
           }
         }
       }
     writeTextToFile(result);
     ReceivedCalls += result + "------------------------------\n";
   }
}
```

Pętla ulega przerwaniu natychmiast po stwierdzeniu niezgodności z powyższym regexem (`messageRegex.Match(comPortTemp)`). Oznaczone zostają wszystkie znaki od słowa `Incoming` aż do znaku końca transmisji `> \`. Następnie, wykryty ciąg znaków zostaje wycięty ze zmiennej przechowującej bufor portu COM, a wszystkie znaki cofnięcia `\r` usunięte. Symbole następnej linii `\n` posłużyły do podzielenia ciągu na tablicę złożoną z poszczególnych linii komunikatu. Kolejne indeksy tablicy zostają poddane analizie na obecność znaku `>`. Linia z takim znakiem jest wysyłana w celu wybudzenia adresata i uprzedzeniu go o rozpoczęciu transmisji szeregowej. Zostaje zignorowana, jeżeli nie zawiera żadnych ważnych informacji. Istotne są jedynie linijki zawierające znak równości `=`, słowo `Incoming` oraz `Type:`. Pierwsze poddane są analizie linie ze znakiem równości. Spacje ulegają usunięciu, a sama linia jest dzielona po znaku `=`. Dzięki temu możliwym staje się porównanie zawartości pierwszej części linii oraz dopasowanie odpowiedniej

metody z obiektu `ddr` klasy `DistressDataResolver`. Każda z nich przyjmuje kod i zwraca przyjazną człowiekowi nazwę. Przykładowa metoda tej klasy wygląda następująco:

Listing 3.21. Implementacja metody ResolveEndOfSequence

```
public string ResolveEndOfSequence(string code)
{
  if (code == "117")
    return "RQ Acknowledge required";
  else if (code == "122")
    return "BQ Acknowledge respond";
  else if (code == "127")
    return "Other calls";
  return code;
}
```

W przypadku pozycji geograficznej, zostaje dodatkowo wysłana informacja do części aplikacji odpowiadającej za wyświetlania pozycji zgłaszającego na mapie. Po dokonaniu dekodowania, nowe wartości zapisywane są na koniec ciągu wynikowego. Ostatnią operacją jest obsłużenie przypadku, w którym linia zaczyna się od słowa `Incoming`. Ciąg znaków zostaje podzielony spacją, a słowo `Incoming` zostaje zamienione na `Type:`. Poniżej zostaje przedstawiona wiadomość nadana przez radiostację:

Listing 3.22. Przykładowa wiadomość nadana przez radiostację

```
?
?
?
K?


Incoming Distress

ID = 261431000
Nature = 107
Pos = 53N25,014E33
Time =  9:42
Sub = 109
Eos = 127

Time 22:01.09
```

oraz ta sama wiadomość po zdekodowaniu:

Listing 3.23. Przykładowa wiadomość po zdekodowaniu

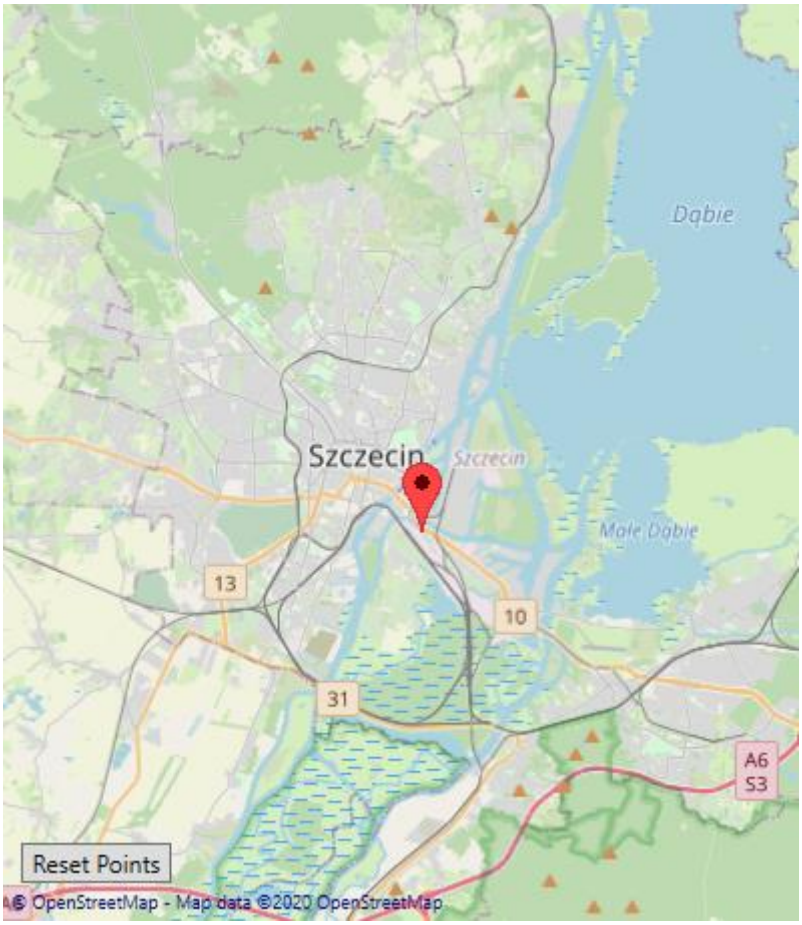
```
Type: Distress
ID: 261431000
Nature: Undesined distress
Pos: 53N25,014E33
Time: 9:42
Sub: 109
Eos: Other calls
Time: 22:01.09
```

3.5.3 Podsumowanie

Implementowanie funkcjonalności, umożliwiającej odbiór danych z radiostacji wykorzystując port COM, zakończyło się pełnym sukcesem. Aplikacja odbiera wywołania alarmowe, przekształca je do postaci czytelnej przez operatora oraz zapisuje na dysku komputera. Ograniczenia w archiwizacji komunikatów wynikłych z limitów pamięci urządzenia SAILOR RE 2100 przestają mieć znaczenie, a dane wpisywane wcześniej ręcznie do komputera zostały na nim zapisane w sposób automatyczny.

3.6 Wyświetlanie pozycji nadawcy komunikatu na mapie Google

Wyznaczenie na mapie poprawnej pozycji nadawcy komunikatu może być dla nawigatora kłopotliwe. Szczególnie, gdy jednostka nadająca jest w niebezpieczeństwie i każda sekunda jest kluczowa w przypadku sytuacji zagrożenia życia. Stąd też uznano, że bardzo przydatne może być automatyczne wyświetlenie pozycji nadawcy na ogólnodostępnych, elektronicznych mapach. Niewątpliwą zaletą tego rozwiązania jest możliwość przechowania map w pamięci podręcznej systemu, dzięki czemu nie wymagają stałego dostępu do Internetu. W przypadku niewielkich akwenów, jak morze Bałtyckie, wszystkie wymagane dane można pobrać do cache'u aplikacji w porcie i wyruszyć zaopatrzonym w komplet map elektronicznych. Oszczędne gospodarowanie transmisją danych na morzu jest kluczowe ze względu na wysokie koszty oraz niską przepustowość łącza satelitarnego.



Rys. 3.14 Część interfejsu użytkownika wyświetlająca pozycję nadawcy. Źródło: Opracowanie własne.

3.6.1 Implementacja kontrolki mapy

Do wykonania funkcjonalności została użyta biblioteka GMap.NET. Zawiera w sobie wszystkie niezbędne funkcje oraz posiada wsparcie dla wielu dostawców map, w tym tych najpopularniejszych jak Bing, Google, Here, czy StreetView. Implementacja funkcjonalności została tak przemyślana, aby umożliwić zmianę dostawcy niewielkim kosztem. Widok składa się z przycisku oraz kontrolki udostępnionej przez bibliotekę GMap.NET:

Listing 3.24. Zawartość kontrolki GMapControl

```
<gmaps:GMapControl x:Name="mapView" Loaded="MapView_Loaded" ItemsSource="{Binding Markers}"/>
<Button Command="{Binding ResetPointCommand}" Content="{Binding ResetPointName}" HorizontalAlignment="Left" Margin="10,420,0,0" VerticalAlignment="Top" Width="75"/>
```

Implementację rozpoczyna kontrolka `GMapControl` nazwana `mapView` ze zdefiniowanym zdarzeniem typu `Loaded` odpowiadającym za wywołanie metody `MapView_Loaded` zadeklarowanej w code behind oraz źródłem prawdy o rozmieszczeniu pinezek na mapie w postaci Bindingu `Markers`. Umieszczenie funkcji w tym miejscu zostało podyktowane wymaganiami kontrolki, która potrzebowała kompletu danych do poprawnego uruchomienia. Tak więc metoda zawiera przede wszystkim przypisywania wartości do odpowiednich właściwości kontrolki `mapView`:

Listing 3.25. Implementacja metody MapView_Loaded

```
private void MapView_Loaded(object sender, RoutedEventArgs e)
{
  mapView.MapProvider = GMap.NET.MapProviders.OpenSeaMapHybridProvider.Instance;
  GMaps.Instance.Mode = AccessMode.ServerAndCache;
  mapView.IgnoreMarkerOnMouseWheel = true;
  mapView.ShowCenter = false;
  mapView.MinZoom = 2;
  mapView.MaxZoom = 24;
  mapView.Zoom = 11;
  mapView.Position = new PointLatLng(53.4285, 14.5637);
  mapView.MouseWheelZoomType = MouseWheelZoomType.MousePositionAndCenter;
  mapView.CanDragMap = true;
  mapView.DragButton = MouseButton.Left;
}
```

Jako dostawca map, został wybrany darmowy OpenSeaMap. Domyślną lokalizacją jest punkt, z którego nadaje radiostacja Akademii Morskiej w Szczecinie. Konieczna była konfiguracja obsługi podstawowych zdarzeń znanych z ogólnodostępnych map internetowych, takich jak przycisk odpowiedzialny za przesuwanie mapy czy zachowanie kółka myszy. Świadczy to o szerokiej gamie możliwości konfiguracyjnych kontrolki, umożliwiającej dostosowanie jej w niemal dowolny sposób.

Ważnym elementem elektronicznej mapy są również pinezki. W celu uzyskania wyglądu mapy zbliżonego do popularnego Google Maps, użyto grafiki możliwie zgodnej

z oryginałem. Każda pojedyncza pinezka jest osobnym obiektem rysowanym na bazie kontrolki `PinControl`, której implementacja wygląda następująco:

Listing 3.26. Implementacja kontrolki PinControl

```
<Image Width="20">
<Image.Source>
  <BitmapImage UriSource="/Resources/Images/gMapsPin.png"/>
</Image.Source>
</Image>
```

Całość obsługuje systemowa kontrolka `Image`, której źródło (`Image.Source`) zostaje ustawione na obraz bitmapowy (`BitmapImage`) `gMapsPin.png` znajdujący się (`UriSource`) w katalogu `/Resources/Images` w folderze rozwiązania.

3.6.2 Implementacja logiki funkcjonalności

Funkcjonalność korzysta z dwóch klas. Pierwsza jest ViewModelem `gMapViewModel`, a druga modelem `GeoMessage`. Ostatnia z nich została użyta do zapewnienia komunikacji pomiędzy obiektami klas `ComCaptureViewModel` oraz `gMapViewModel`. Została użyta biblioteka TinyMessenger, aby zmniejszyć rozmiar implementacji. Komunikacja odbywa się w modelu subskrypcyjnym, a sama subskrypcja następuje jeszcze w konstruktorze klasy:

Listing 3.27. Implementacja konstruktora klasy gMapViewModel

```
public gMapViewModel() {
  Services.MessengerHub.Subscribe<GeoMessage>(addPoint);
  ResetPointName = "Reset Points";
  ResetPointCommand = new DelegateCommand(ResetPoints);
}
```

Dodatkowo, zostaje tutaj przypisana bindingowi `ResetPointCommand` metoda `ResetPoints`. Binding ów jest przypisany do właściwości `Command` przycisku odpowiedzialnym za czyszczenie oznaczeń na mapie. Model `GeoMessage` zawiera w sobie informacje o długości i szerokości geograficznej oraz konstruktor umożliwiający przypisanie właściwościom odpowiednich wartości w trakcie inicjalizacji obiektu:

Listing 3.28. Implementacja klasy GeoMessage

```
class GeoMessage : TinyMessageBase
{
  public string Lat { get; set; }

  public string Long { get; set; }

  public GeoMessage(object sender, string Lat, string Long) : base(sender)
  {
    this.Lat = Lat;
    this.Long = Long;
  }
}
```

Subskrypcja polega na tym, że po każdej publikacji obiektu zadanej klasy zostają, w sposób asynchroniczny, wywołane zasubskrybowane do nich metody. W poprzednim podrozdziale (3.5.2), podczas analizy składni komunikatu, w warunku odpowiedzialnym za pozycję nadawcy wykonana została następująca instrukcja:

```
Services.MessengerHub.PublishAsync<GeoMessage>(new GeoMessage(this, s3[0],
s3[1]));
```

Pod wartością `s3[0]` kryła się szerokość geograficzna, a pod `s3[1]` wysokość. Spowodowało to inwokację metody `addPoint` zasubskrybowanej w konstruktorze `gMapViewModel`. Funkcja składa się z dwóch, analogicznych, bloków `if-else` odpowiedzialnych za konwersję szerokości oraz wysokości geograficznej. Spowodowane jest to używaniem przez kontrolkę `gMap` notacji dziesiętnej z założeniem, że punktem zerowym (0,0) układu współrzędnych jest punkt przecięcia południka Greenwich z równikiem. Tak więc kierunek północny i wschodni przyjmuje wartości dodatnie, a południowy z zachodnim ujemne. Dodatkowo, wymagane było przekształcenie minut w wartość dziesiętną, co zostało rozwiązane podzieleniem ilości minut przez sześćdziesiąt. Obsługa współrzędnych geograficznych wygląda następująco:

Listing 3.29. Implementacja przekształcenia współrzędnych geograficznych

```
double la;
double lo;

if (gm.Lat.Split('N').Length == 2) {
  // N => +
  string[] lat = gm.Lat.Split('N');
  la = Double.Parse(lat[0]) + (Double.Parse(lat[1]) / 60.0);
} else {
  // S => -
  string[] lat = gm.Lat.Split('S');
  la = (-1.0) * (Double.Parse(lat[0]) + (Double.Parse(lat[1]) / 60.0));
}
if (gm.Long.Split('E').Length == 2) {
  // E => +
  string[] lon = gm.Long.Split('E');
  lo = Double.Parse(lon[0]) + (Double.Parse(lon[1]) / 60.0);
} else {
  // W => -
  string[] lon = gm.Long.Split('W');
  lo = (-1.0) * (Double.Parse(lon[0]) + (Double.Parse(lon[1]) / 60.0));
}
```

Po przekształceniu formatu zapisu współrzędnych następuje wyświetlenie pinezek na mapie. Z racji tego, iż dane mogły zostać uprzednio wyczyszczone poprzez kliknięcie przycisku na interfejsie użytkownika, następuje ponowna deklaracja zmiennej:

Listing 3.30. Odświeżenie zmiennej w przypadku wyczyszczenia danych

```
if (markersValue.Count == 0) {
  markersValue = new ObservableCollection<GMapMarker>();
}
```

Po upewnieniu się, że zmienna `markersValue` istnieje, następuje dodanie znacznika na mapę:

Listing 3.31. Implementacja dodawania znacznika na mapę

```
Application.Current.Dispatcher.Invoke((Action) delegate {
 GMapMarker gmm = new GMapMarker(new PointLatLng(la, lo));
 gmm.Shape = new PinControl();
 Markers.Add(gmm);
 OnPropertyChanged("Markers");
});
```

Należy pamiętać, że metoda wykonywana jest asynchronicznie. Oznacza to potrzebę stworzenia delegata w celu wykonania operacji na synchronicznym, głównym, wątku aplikacji. W nim utworzony zostaje obiekt klasy `GMapMarker` przyjmujący współrzędne geograficzne w formie obiektu modelu `PointLatLng`. Jako kształt zostaje ustawiony nowy obiekt kontrolki `PinControl`, który to dodawany jest do kolekcji `GMapMarkerów` o nazwie `Markers`, znanej z poprzednio omówionej właściwości `ItemsSource` w kontrolce `GMapControl`. Po poinformowaniu interfejsu o zmianie w bindingu `Markers` (`OnPropertyChanged("Markers")`) nowy znacznik ukazuje się użytkownikowi aplikacji.

3.6.3 Podsumowanie

Pomimo pierwotnych trudności, wynikłych z braku komunikacji pomiędzy obiektami aplikacji, implementacja wyświetlania pozycji nadawcy komunikatu została przeprowadzona pomyślnie. W efekcie zniwelowany został czas potrzebny operatorowi na wyznaczeniu współrzędnych geograficznych nadawcy na mapie. Rozwiązanie zostało napisane na tyle elastycznie, że w przyszłości może zostać rozszerzone o dodatkowe funkcjonalności. Sugerowany jest dynamiczny wybór dostawcy map, większy wybór pinezek oraz możliwość usuwania pojedynczych znaczników.

PODSUMOWANIE

W ramach niniejszej pracy opracowano oprogramowanie służące do rejestracji i archiwizacji wiadomości odebranych za pośrednictwem radiostacji MF/HF z przystawką DSC. Aplikacja umożliwia dwukierunkowe nagrywanie komunikacji dźwiękowej dokonywanej poprzez radiostację, dzięki czemu możliwe stało się ponowne odsłuchanie przeprowadzonej wcześniej rozmowy. Dodatkowo, całość komunikatu tekstowego wywołania zostaje zapisana w formie elektronicznej na dysku komputera w formatach (*.wmv, *.txt) obsługiwanym przez domyślne aplikacje systemu Windows. Dzięki temu zbędne staje się ręczne wprowadzanie potrzebnych informacji do sytemu. Pozyskane dane są wykorzystywane przez aplikację do zaznaczenia pozycji nadawcy komunikatu na mapie elektronicznej udostępnianej nieodpłatnie przez „OpenSeaMap”. Pozwala to operatorowi szybciej zareagować w przypadku wywołań alarmowych i skrócić czas potrzebny do udzielenia pomocy jednostce w niebezpieczeństwie.

Sama aplikacja, po wcześniejszym skonfigurowaniu, jest w stanie przepracować długie tygodnie bez potrzeby jej ponownego uruchamiania. Tym samym może służyć za swego rodzaju czarną skrzynkę oraz jako narzędzie nadzoru.

SPIS RYSUNKÓW

SPIS TABEL

SPIS LISTINGÓW

BIBLIOGRAFIA

1. Albahari B., Albahari J., C# 7.0 in a Nutshell, O'Reilly Media, Sebastopol 2017
2. Bojarski P., Korcz K., Radiotelefoniczna łączność statków morskich, Fundacja Rozwoju Wyższej Szkoły Morskiej w Gdyni, Gdynia 1998
3. Chomski J., Bober R., Instrukcja: Radiostacja MF/HF Sailor RE 2100, Akademia Morska w Szczecinie, Szczecin 2014
4. Czajkowski J., Korcz K., GMDSS dla łączności bliskiego zasięgu, Skryba, Gdańsk 2006
5. Praca zbiorowa pod redakcją Jerzego Czajkowskiego, System GMDSS regulaminy, procedury i obsługa, Skryba, Gdańsk 2002
6. Mąka M., Majzner P., Instrukcja: elektrotechnika i elektronika. Ćwiczenie nr 6: Modulacja i detekcja, Akademia Morska w Szczecinie, Szczecin 2012
7. Nathan A., WPF 4.5 Księga Eksperta, Helion, Gliwice 2015

Strony WWW:

8. http://gs.statcounter.com/os-market-share/desktop/worldwide, dostęp 09.06.2019r.
9. https://docs.microsoft.com/en-gb/dotnet/framework/wpf/advanced/xaml-overview-wpf, dostęp 29.12.2020r.
10. https://docs.microsoft.com/pl-pl/nuget/what-is-nuget, dostęp 29.12.2020r.
11. https://github.com/naudio/NAudio/tree/master/Docs, dostęp 29.12.2020r.
12. https://github.com/xceedsoftware/wpftoolkit, dostęp 29.12.2020r.
13. https://github.com/judero01col/GMap.NET/blob/master/README.md#releas e-notes, dostęp 29.12.2020r.
14. https://docs.microsoft.com/en-us/windows/win32/cimwin32prov/win32-serialport, dostęp 29.12.2020r.