DOI: 10.17814/mechanik.2015.10.506

mgr inż. Paweł Ksieniewicz
Wydział Elektroniki
Politechnika Wrocławska, Wrocław

SYSTEM KOLEKCJONUJĄCY DANE POGODOWE NA BAZIE METEOROGRAMÓW

STRESZCZENIE

Artykuł skupia się na problemie pozyskiwania i kolekcjonowania danych meteorologiczny dotyczących obszaru Morza Bałtyckiego, które są niezbędnym elementem do zaprojektowania oraz zaimplementowania efektywnego systemu ostrzegania o ryzyku wystąpienia szturmu w akwenie tego morza. Porusza on również niezwykle istotny problem, który utrudnia lub wręcz uniemożliwia podjęcie prac w tym zakresie. Problem ten dotyczy ograniczonego przez instytucje państwowe dostępu do przydatnych w badaniach naukowych danych meteorologicznych, które są traktowane głównie jako produkt komercyjny.

Niniejszy artykuł skupia się studium przypadku dotyczącym obszaru Morza Bałtyckiego. Celem omawianych prac była konstrukcja niezależnej od Instytutu Meteorologii i Gospodarki Wodnej bazy danych meteorologicznych wyżej wymienionego akwenu z okresu najwyższej częstotliwości występowania sztormów. W tym celu posłużono się meteorogramami uzyskanymi z Numerycznej Prognozy Pogody Uniwersytetu Warszawskiego, oprogramowaniem dekodującym dane numeryczne z nieskompresowanych bitmap oraz bazą danych, która pozwala na ich akumulację i udostępnienie. Wykorzystanie wcześniej wspomnianej metody pozwoliło na skonstruowanie, złożonej z 600600 obiektów bazy danych z okresu czterech miesięcy, na dostatecznie dokładnej siatce punktów pomiarowych rozpiętej na akwenie morza, przedstawiając tym samym propozycję alternatywnego rozwiązania nakreślonego na początku problemu.

Słowa kluczowe: *bazy danych, meteorologia, programowanie, dostęp do danych.*

A SYSTEM COLLECTING WEATHER DATA BY MEANS OF METEOGRAMS

ABSTRACT

The paper concentrates on the problem of limited by the state institutions access to meteorological data, relevant to scientific research, that are regarded only as commercial product. The search for the new source of access was based on the case study of the Baltic Sea area. The aim of the project was to create a meteorological database, independent of the Institute of Meteorology and Water Management, concerning the Baltic Sea in the period with the highest frequency of storms. For this purpose, meteograms from the Numerical Weather Forecast from the Warsaw University, the software decoding numerical data from uncompressed bitmap and the database allowing accumulation of data and making them available were used. The application of mentioned method allowed us to construct database consisted of 600600 items, from four months, on precise grid of measuring points stretched on the Baltic Sea basin; thereby presenting an alternative solution of the problem.

Keywords: *databases, meteorology, programming, data access.*

1. WSTĘP

Wstępnym celem omawianego przedsięwzięcia było utworzenie systemu ostrzegania o ryzyku wystąpienia sztormu na akwenie *Morza Bałtyckiego*. W trakcie pracy nad nim okazało się jednak, że w polskich warunkach największym wyzwaniem nie jest właściwa praca nad doskonaleniem systemu, ale samo zdobycie danych niezbędnych do badań. Toteż głównym celem omawianego w niemniejszym artykule przedsięwzięcia okazało się opracowanie efektywnego metody kolekcjonowania niezbędnych danych meteorologicznych.

Sztorm to zjawisko meteorologiczne występujące nad obszarami mórz i oceanów. Charakteryzuje się długotrwałym, porywistym i silnym wiatrem o sile nie mniejszej niż 8 stopni w skali Beauforta [9]. Wyrażając to w jednostkach metrycznych, mowa jest o wietrze o sile minimum 17,2 *m/s* utrzymującym się co najmniej kilka godzin [8].

Za parametry istotne dla powstania zjawiska sztormu uznano:

- temperaturę powietrza, z dokładnością do $1^{\circ}C$,
- ciśnienie atmosferyczne zredukowane do poziomu morza, z dokładnością do $1hPa$,
- średnią prędkość wiatru, z dokładnością do $1m/s$,
- prędkość wiatru w porywach, z dokładnością do $1m/s$.

Do poprawnej nauki i bieżącej predykcji ryzyka sztormu niezbędne były odpowiednio wyselekcjonowane dane historyczne (nauka) i aktualne (pomiar, odczyt). Zdefiniowano więc kryteria oceny źródeł danych wejściowych:

- **częstość** — dane dostępne w regularnych i możliwie niewielkich interwałach czasu,
- **gęstość** — siatka punktów meteorologicznych możliwie dokładnie odwzorowująca obszar akwenu,
- **serializowalność** — dane w powszechnym i łatwym do przetwarzania formacie.

2. ŹRÓDŁO DANYCH

2.1. Trudności z IMGW

Wstępną fazą projektu była lokalizacja źródła danych spełniających zdefiniowane kryteria. Ze względu na *ustawę o dostępie do informacji publicznej* [3], wydawało się to trywialnym zadaniem, ograniczającym się do odwiedzenia witryny internetowej finansowanego z budżetu państwa *Instytutu Meteorologii i Gospodarki Wodnej*. Instytut ten od kilkudziesięciu lat prowadzi pomiary na terytorium Polski. Dzięki temu dysponuje historycznymi i aktualnymi danymi meteorologicznymi dla całego obszaru interesującego nas z punktu widzenia projektu. Jednak rzeczywistość sprawiła, że to zadanie po początkowym rozpoznaniu, stało się odrębnym projektem.

Po analizie dostępnych informacji, okazało się, że IMGW nie traktuje się historycznych i bieżących danych meteorologicznych, jako informacji publicznej, a jako produkt komercyjny, żądając tym samym opłat za ich udostępnianie. Dodatkowo, odmawia się udostępnienia danych mówiących o wysokości opłat za jakie sprzedaje się te dane uczelniom i firmom. Podsumowując, wykonana dzięki państwowym pieniądzom baza danych jest dostępna wyłącznie za nieujawnioną i wynegocjowaną opłatą, indywidualną dla każdego podmiotu.

Inną drogą do uzyskania danych potrzebnych do badań jest umowa stowarzyszeniowa pomiędzy IMGW, a uczelnią. Niestety, podczas realizacji projektu, Politechnika Wrocławska okazała się jedyną wrocławską państwową uczelnią, nie będącą sygnatariuszem podobnego dokumentu. O dziwo, nie było to problemem, ponieważ umowa taka wymaga zdefiniowania i zaakceptowania przez IMGW przed rozpoczęciem semestru wszystkich projektów realizowanych w jego przeciągu przez studentów i pracowników uczelni. W świetle tej konstrukcji umowy, projekt realizowany w ramach kursu uczelnianego, nadal nie miałby dostępu do pożądanych informacji.

Problem dostępności danych dotyczących Morza Bałtyckiego dotknął również dr Marka Chabiora piszącego prace habilitacyjną na *Zachodniopomorskim Uniwersytecie Technologicznym* w Szczecinie. Wniósł on skargę w sprawie IMGW do wiceministra środowiska. Następnie wniósł kolejną skargę dotyczącą IMGW i wiceministra środowiska do premiera [2]. Dzięki temu udało mu się doprowadzić do zmian w *Prawie wodnym*, ale niestety IMGW nie zareagował na jego nowelizację. Przypadki takie, jak ten można mnożyć, co ciekawe niektóre podmioty wygrały nawet z IMGW sprawy sądowe dotyczące tego problemu. [5]

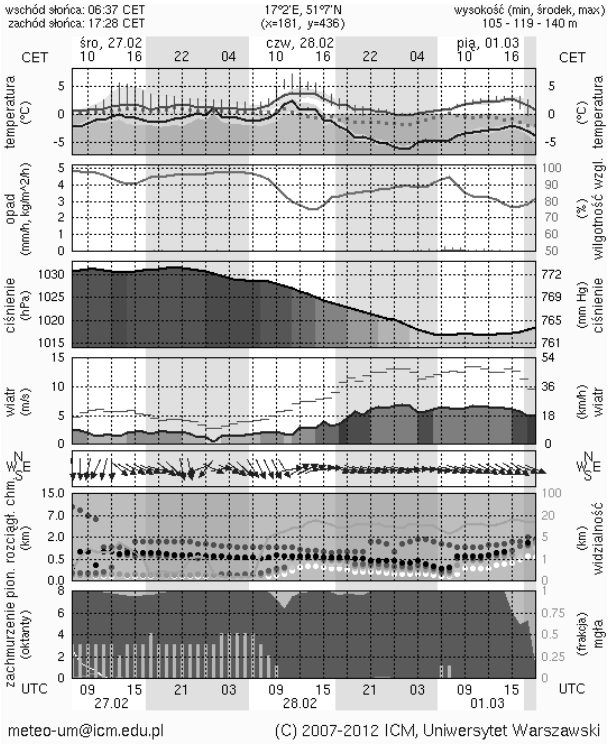
Interesujące jest to, że za dane z 3 stacji z interwału 20 lat IMGW, od osoby fizycznej, życzy sobie 3 miliony złotych [4]. Wyżej wymienione i wiele podobnych przypadków zostało opisanych na stronach Pozarządowego Centrum Dostępu do Informacji Publicznej [6].

2.2. Wybrane źródło danych wejściowych

Ostatecznie wybranym źródłem danych okazały się, udostępniane w formie plików png meteorogramy modelu um, generowane przez *Interdyscyplinarne Centrum Modelowania Matematycznego i Komputerowego, Uniwersytetu Warszawskiego* [1].

Meteorogramem nazywamy diagram lub wykres reprezentujący wzorzec wy-stępowania dwóch lub więcej zjawisk meteorologicznych, zaobserwowanych na przestrzeni czasu w danej stacji meteorologicznej, lub ich grupie. Rysunek 1 prezentuje przykładowy meteorogram.

Nie prezentują one danych historycznych, ani aktualnych odczytów, a jedynie bieżącą prognozę pogody. Charakteryzuje się ona jednak wysokim współczynnikiem sprawdzalności. Kolekcjonując jej odczyty przez dłuższy okres można skonstruować bazę danych pogodowych wystarczającą do przeprowadzenia procesu uczenia sieci neuronowej.



Rysunek 1: Meteorogram

2.3. Zakreślenie geograficznego obszaru badań

Meteorogramy po stronie ICM generowane są przy użyciu skryptu mgram_pict.php, wywoływanego przykładowym adresem url:

http://www.meteo.pl/um/metco/mgram_pict.php?ntype=0u&fdate=2013030106 &row=436&col=181&lang=pl

Akcją get przekazywane jest mu pięć parametrów:

- ntype — nieokreślony parametr nie wydający się mieć wpływu na kształt diagramu
- fdate — data, w formacie +%Y%m%d00, określająca początek meteorogramu
- row — określenie długości geograficznej
- col — określenie szerokości geograficznej
- lang — język pól meteorogramu

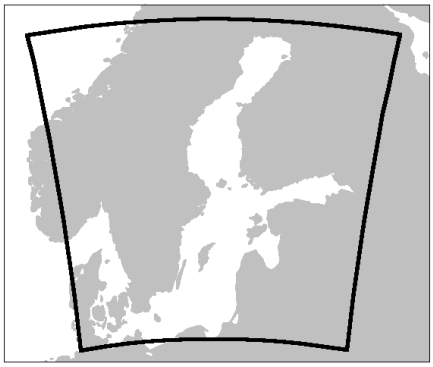
Parametry row i col do określenia koordynat geograficznych nie używają stopni, a własnej skali, wynikającej z zastosowanej metody kartograficznej. Parametr row przyjmuje argumenty od 10 do 430, a parametr col od 10 do 598. Za poprawną, skrypt przyjmuje co siódmą liczbę całkowitą mieszczącą się w tych przedziałach, co efektywnie ogranicza siatkę pozornych punktów pomiarowych do 60 kolumn na 84 wiersze. Mapę całego dostępnego obszaru prezentuje rysunek 2.

Pełna, dostępna sieć pozornych punktów pomiarowych składa się więc z 5040 elementów. Średni czas generowania diagramu na serwerze wynosi około 5 sekund. Sprawia to, że sam proces kolekcjonowania materiału do przygotowania danych numerycznych, dla pełnej mapy, przyjmując czas 5 sekund na diagram, trwałby równo 7 godzin. Dla optymalizacji czasu tego procesu, zdecydowano się na ograniczenie siatki.

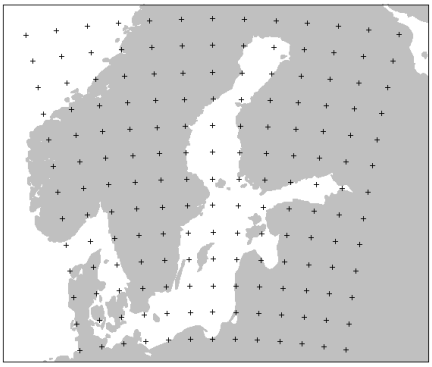
Rozpoczęto od określania przestrzeni, na której miała opierać się siatka badanych punktów. Za graniczne punkty przyjęto 10 i 346 parametru row oraz 52 i 402 parametru col. Zakres został przedstawiony na rysunku 3.

Pełna siatka dla ograniczonego zakresu zawierała 2400 punktów. Czas kolekcjonowania dla niej wynosił w przybliżeniu 3 godziny 20 minut, co nadal nie było akceptowalnym rezultatem. W skrypcie ściągającym dane z serwera wprowadzono więc dodatkowy parametr w postaci mnożnika skoku iteracji, ustalając go na 4. Prowadziło to do skoku nie o wartość 7 a 28. Ograniczyło to siatkę do 169 punktów, pozwalając na zebranie danych w przeciągu, akceptowalnych, 15 minut.

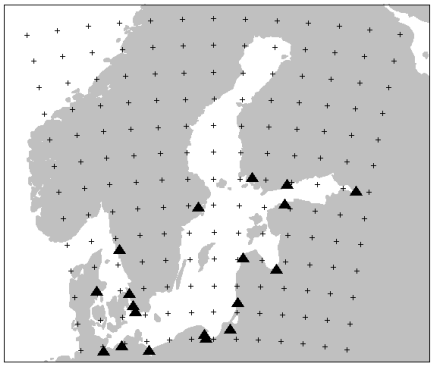
Po analizie statystyk przeładunku i rocznej liczby przewożonych pasażerów, wyselekcjonowano listę dziewiętnastu najistotniejszych portów Morza Bałtyckiego [7] i naniesiono ich lokalizacje na mapę punktów pomiarowych (Rysunek 5).



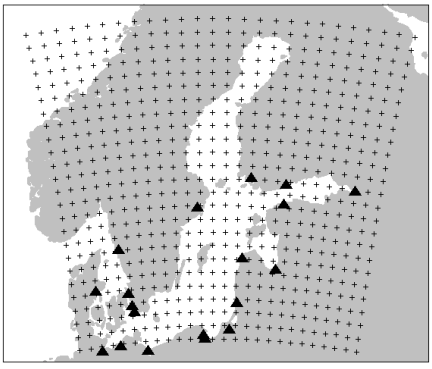
Rysunek 3: Geograficzna przestrzeń badań



Rysunek 4: Lokalizacja pierwszego zestawu punktów pomiarowych



Rysunek 5: Porty Morza Bałtyckiego (trójkąty) naniesione na pierwszy zestaw punktów pomiarowych

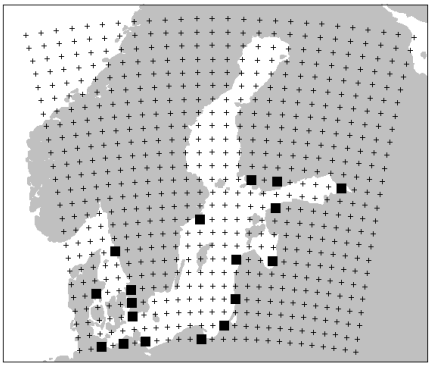


Rysunek 6: Lokalizacja drugiego zestawu punktów pomiarowych

Jak widać na rysunku 5, przygotowana siatka jest niedokładna i w niektórych przypadkach, np. w okolicy *Cieśnin Duńskich* do jednego punktu pomiarowego może przynależeć kilka portów, odległych nawet o kilkadziesiąt kilometrów.

Podjęto więc decyzję o zwiększeniu dokładności siatki punktów, przez zmniejszenie współczynnika skoku do 2. Doprowadziło to do siatki 600 punktów i około 50 minut potrzebnych do ściągnięcia niezbędnych plików png.

Na potrzeby bazy danych dokonano również przypisania wybranych wcześniej portów morskich do punktów meteorologicznych. Gotową siatkę, na której zrealizowano projekt, z zaznaczonymi punktami przypisanymi do portów, prezentuje Rysunek 7.



Rysunek 7: Przypisanie portów do punktów pomiarowych

Tabela 1: Wybrane porty Morza Bałtyckiego

Miasto portowe	Kod portu	**Miasto portowe**	Kod portu
G¨oteborg	*SEGOT*	Sankt Petersburg	*RULEN*
L¨ubeck	*DELBC*	Klaipeda	*LT KLJ*
Riga	*LV RIX*	Tallinn	*EET LL*
Ventspils	*LV V NT*	Rostock	*DERSK*
Świnoujście	*P LSW I*	Gdańsk	*P LGDA*
Malmø	*SEMMA*	Gdynia	*P LGDY*
Kalinigrad	*RUKGD*	Aarhus	*DKAAR*
Helsinki	*F IHEL*	Trelleborg	*SET RG*
Stockholm	*SEST O*	Helsingborg	*SEHEL*
Turku	*F IT KU*		

3. DEKODER DANYCH WEJŚCIOWYCH

Wbrew pozorom, dekoder meteorogramów, który powstał na potrzeby projektu nie był oprogramowaniem ocr. Analizowane pliki png generowane na serwerze, prawdopodobnie przy użyciu programu *gnuplot* są nieskompresowanymi plikami rastrowymi o indeksowanym kolorze. Dekoder nie przybliża więc odczytu na bazie materiału optycznego, jak ma to miejsce w przypadku ocr, ale dysponując niezakłamanym ciągiem bitów z zakodowanymi danymi, dostępnymi w nim z dokładnością wynikającą ze skali wykresów i rozdzielczości pliku png.

ICM dodatkowo ułatwiło sprawę rezygnując z wygładzania tekstu antyaliasingiem. Pozwoliło to na szybkie wykrycie obecności określonego znaku na zadanym obszarze pliku, zgodnie z metodą opisaną później. Z kolei utrudnieniem okazało się indeksowanie kolorów na grafikach. Na tym etapie, każdy plik po ściągnięciu z serwera jest konwertowany do formatu png32.

Powstały dekoder został zaprogramowany przy użyciu języka *C++* [10], używając standardu *C99*. Składa się on z trzech klas — modułów i niewielkiego zbioru pomocniczych struktur.

Dla optymalizacji czasu dekodowania, dane konfiguracyjne, wskazujące obszary, wzorce i klucze poszukiwań zostały umieszczone na stałe w kodzie. W typowych lub udokumentowanych w opisach używanych bibliotek fragmentach kodu, pominięto szczegóły implementacyjne, skupiając się jedynie na interfejsach klas.

Klasa ***pngmatrix*** korzysta z biblioteki *libpng* i biblioteki standardowej (*lib-std*). W obecnej wersji nie implementowano obsługi błędów.

Obiekt klasy *pngmatrix* pozwala traktować wczytany do niego plik graficzny, jak dwuwymiarową, adresowaną tablicę pikseli. Metody klasy pozwalają na wczytanie pliku o zadanej nazwie (listing 1), zwrócenie wymiarów obrazka i zwrócenie wartości koloru czerwonego lub zielonego, modelu rgb (listing 2).

Listing 1: Metoda konstruktora klasy *pngmatrix*

```
pngmatrix::pngmatrix(char* file_name)
{

 ...

 width =
  png_get_image_width(png_ptr, info_ptr);
 height =
  png_get_image_height(png_ptr, info_ptr);

 /* read file */
 row_pointers =
  (png_bytep*)malloc(sizeof(png_bytep)*height);

 for (y=0; y<height; y++)
  row_pointers[y] =
   (png_byte*)malloc(png_get_rowbytes
    (png_ptr,info_ptr));

 png_read_image(png_ptr, row_pointers);

 fclose(file);
}
```

Listing 2: Metoda *red_pixel* klasy *pngmatrix*

```
int pngmatrix::red_pixel(int _x, int _y)
{
    png_byte* ptr = &(row_pointers[_y][_x*4]);
    return ptr[0];
}
```

patto Klasa *patto* korzysta z obiektu klasy pngmatrix. Pozwala to na wygenerowanie 64-bitowego ciągu na podstawie zakreślonego, nie większego od 64 pikseli, fragmentu pliku graficznego. Wygenerowany ciąg, nazywany *wzorcem* jest przechowywany w zmiennej unsigned long long int.

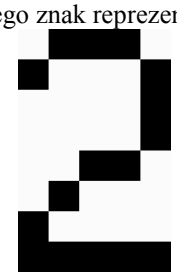
Pierwszy bit wzorca, zawsze jest równy 0. Iterując, wierszami od lewej do prawej, kolejne piksele określonego strukturą frame obszaru obrazka, porównywana jest wartość kanału czerwonego aktualnego i poprzedniego piksela. W wypadku, gdy obie porównywane wartości są różne, do wzorca dodawana jest jedynka. Po każdym porównaniu wzorzec jest przesuwany w rejestrze o jedną pozycję.

Listing 3: Generator wzorca

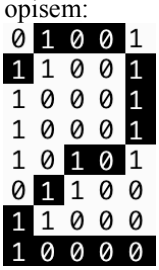
```
unsigned long long int mask = 0;
int comparator =
   png -> red_pixel(frame.y, frame.x);
int value = 0;

for ( int x = frame.x ;
      x < frame.x + frame.height ;
      x++ )
for ( int y = frame.y ;
      y < frame.y + frame.width ;
      y++ )
{
   value = png -> red_pixel(y, x);
   mask += comparator != value;
   mask = mask << 1;
   comparator = value;
}
return mask;
```

Przykładowo, dla pliku graficznego zawierającego znak reprezentujący w wy-świetlanym kroju pisma cyfrę 2:



Plik ma wymiary 5 na 8 pikseli, co po przemnożeniu daje wartość mniejszą od 64.
Wzorzec tworzony jest zgodnie z wcześniejszym opisem:



Utworzony wzorzec można zapisać w postaci liczby całkowitej 673340286496.

analizer Większość logiki programu zrealizowana została w klasie analizer. Korzysta ona z obiektów klas *pngmatrix* i *patto* oraz ze struktur pomocniczych. Metody prywatne klasy (listing 4) pozwalają skalibrować zmienne pomocnicze i uzyskać podstawowe informacje z meteorogramu.

Listing 4: Metody prywatne klasy *analizer*

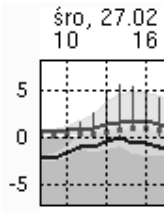
```
// metody prywatne
float temperature_with_column(int);
int pressure_with_column(int);
int wind_max_with_column(int);
int wind_avg_with_column(int);

void set_scales();
void set_date();
void set_station_id();


void first_line(frame _frame, int &v, int &r);
void last_line(frame _frame, int &v, int &r);

int column_for_hour_interval(int);
time_t timestamp_for_hour_interval(int);
```

Przykładowo, próbując odczytać z wykresu wartości temperatury:



Konstruktor klasy (listing 5) zapełnia pola konfiguracji (tablica *chars* i *pat-terns*), a następnie inicjalizuje zerami parametry *v1, v3, r1* i *r3*. Następnie wskazujemy programowi *ramkę* (*temperature scale frame*), w której ma poszukiwać skali diagramu i *zakres* (*temperature range*) w którym ma poszukiwać na nim odczytów. Na koniec wywoływane są metody wyliczające parametry skali, określający na podstawie nazwy pliku graficznego datę pomiarów i lokalizację stacji.

Listing 5: Skrót ciała konstruktora klasy *analizer*

```
chars = (char *) "1-023456789";

patterns[0] = 434207338892;
...
patterns[10] = 672786844274;

temperature_v1 = temperature_v3 =
temperature_r1 = temperature_r3 = 0;

temperature_scale_frame.x = 51;
temperature_scale_frame.y = 30;
temperature_scale_frame.width = 33
temperature_scale_frame.height = 86;

temperature_range.loc = 58;
temperature_range.len = 76;

filename = _filename;
pattern_generator = new patto(filename);

set_scales();
set_date();
set_station_id();
```

Metoda ustalająca skale (listing 6) przypisuje zmiennej v1 wartość pierwsze-go elementu skali (w tym wypadku 5) a zmiennej r1 numer linii zawierającej ten element. Zmiennej v3 przypisywana jest z kolei wartość ostatniego elementu skali (w tym wypadku -5) a zmiennej r3 numer linii zawierającej ten element.

Poszukiwanie wzorca przedstawia listing 7.

Listing 6: Skrót ciała metody *set scales()*

```
void analizer::set_scales(){
    // Temperatura
   first_line(
      temperature_scale_frame,
      temperature_v1,
      temperature_r1);
   last_line(
      temperature_scale_frame,
      temperature_v3,
      temperature_r3);
}
```

Listing 7: Poszukiwanie wzorca na fragmencie wykresu

```
line = -1;
counter = 0;
buffer = new char[5];

unsigned long long int pattern = 0;

for (
   int x = _frame.x ;
   x < _frame.x + _frame.height ;
   x++ )
for (
   int y = _frame.y ;
   y < _frame.y + _frame.width ;
   y++ )
{
   frame _patframe = {x,y,5,8};
   pattern =
      pattern_generator ->
         pattern_for_frame(_patframe);
   for ( int i = 0 ; i < PATTERNS_COUNT ; i++ )
   if (patterns[i] == pattern)
   {
      if (line != x && line != -1) break;
      buffer[counter++] = chars[i];
      line = x;
      break;
   }
}
buffer[counter] = '\0';

v = atoi(buffer);
r = line + 4;
```

Wartość odczytu z wykresu (v_2) określana jest z wykorzystaniem wzoru 1.

$$v_2 = v_1 + \frac{(r_2 - r_1) \cdot (v_3 - v_1)}{(r_3 - r_1)} \qquad (1)$$

Jedyna metoda publiczna klasy, generuje linię zgodną z formatem CSV, prezentującą odczytane dane meteorologiczne z zadanego punktu w czasie (listing 8).

Listing 8: Ciało metody *println_for_hour_interval()*

```
time_t timestamp =
   timestamp_for_hour_interval(interval);

int column =
   column_for_hour_interval(interval);

float temperature =
   temperature_with_column(column);

int pressure = pressure_with_column(column);
int wind_max = wind_max_with_column(column);
int wind_avg = wind_avg_with_column(column);

// station_id / timestamp / temperature /
// pressure / wind_max / wind_avg
printf(
   "%i,%li,%.1f,%i,%i,%i\n",
   station_id,
   timestamp,
   temperature,
   pressure,
   wind_max,
   wind_avg);
}
```

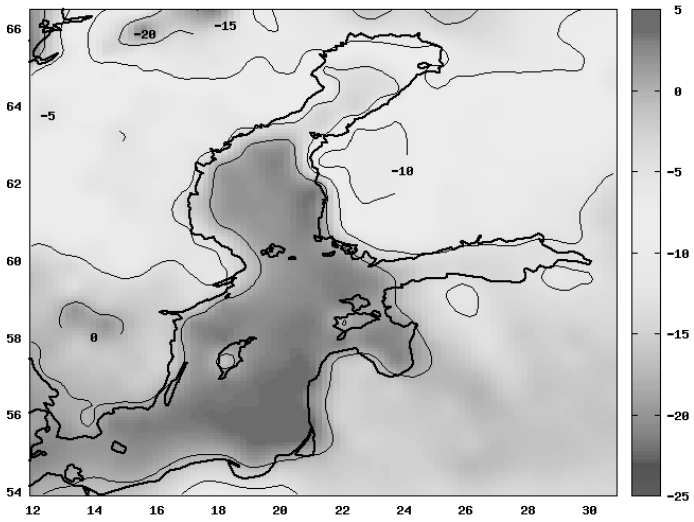
4. PODSUMOWANIE

Niemniejszy artykuł skupia się na problemie pozyskiwania danych meteorologicznych, niezbędnych do stworzenia systemu ostrzegania o ryzyku wystąpienia szturmu w obszarze Morza Bałtyckiego. Podjęte w tym celu prace ukazały, iż nie jest to zadanie trywialne i wymaga wielu regulacji państwowych, w celu umożliwienia naukowcom pozyskania niezbędnych danych oraz ich przetwarzania.

Bezpośrednim efektem prac omawianych w tym artykule jest baza danych meteorologicznych. Opisują one zmiany pogody w akwenie Morza Bałtyckiego w przeciągu kilku miesięcy największej aktywności wiatru.

Stworzona baza zawiera 600600 rekordów danych meteorologicznych dla 650 punktów w przestrzeni. Przykładowo, dla polskiego portu morskiego w Świnoujściu zarejestrowano 266 godzin sztormów (dane zbierane w okresie jesień-zima 2012-2013).

Niestety, od czasu wykonania pomiarów do niemniejszego artykułu, właściciele serwerów ICM dodali do meteorogramów znak wodny. Zmiana formatu meteorogramów uniemożliwia bezpośrednie wykorzystanie stworzonych narzędzi do dalszego zbierania danych. Przed kolejnym procesem zbierania danych metody wymagania modyfikacji. Ujawnia to kolejny problem napotkany przy pracach nad systemem ostrzegania przed sztormami – brak standaryzacji danych pomiarowych oraz sposób ich modyfikacji.



Rysunek 8: Mapa temperatury

LITERATURA

[1] Interdyscyplinarne centrum modelowania matematycznego i komputerowego, www.icm.edu.pl,
[2] Skarga w sprawie opłat za informację o pogodzie i wniosek do zmian w ustawie o dip - Sieć Obywatelska - Watchdog Polska Pozarządowe Centrum Dostępu do Informacji Publicznej,
[3] Ustawa z dnia 6 września 2001 r. o dostępie do informacji publicznej. *Dz.U. 2001 nr 112*, (poz. 1198),
[4] 3 miliony złotych za informację publiczną liczy sobie instytucja utrzymywana przez podatników - Sieć Obywatelska - Watchdog Polska Pozarządowe Centrum Dostępu do Informacji Publicznej, July 2011,
[5] Dostęp do informacji publicznej, October 2011,
[6] Instytut Meteorologii i Gospodarki Wodnej a dostęp do informacji publicznej - Sieć Obywatelska - Watchdog Polska Pozarządowe Centrum Dostępu do Informacji Publicznej, July 2012,
[7] Porty nad Morzem Bałtyckim, December 2014. Page Version ID: 41319832,
[8] Franciszek Haber. *Vademecum żeglarza i jachtowego sternika morskiego*. Aqua fit, 2012,
[9] Krzysztof Kożuchowski. Meteorologia i klimatologia - Księgarnia PWN, 2013,
[10] Bjarne Stroustrup. *The C++ Programming Language, 4th Edition*. Addison-Wesley Professional, Upper Saddle River, NJ, 4 edition edition, May 2013.