

Programowanie inżynierskie Matlab

Projekt stacji pogodowej

Pogoda jest kluczowym elementem w wielu dziedzinach przemysłu, np.: transporcie. Optymalizacja procesu, zminimalizowanie kosztów albo odpowiednie planowanie jest uzależnione od warunków pogodowych. Aby przedsięwziąć odpowiednie środki, przygotować się na niesprzyjające warunki lub wybrać odpowiedni termin niezbędna jest obserwacja zmian zachodzących w przyrodzie.

Podstawowymi parametrami, które słyszymy podczas prognozy pogody to ciśnienie i temperatura. Na ich podstawie możemy zaplanować nasz dzień. Kolejną rzeczą to opady, dane te możemy oprzeć zapisami wcześniejszych danych pogodowych. Możemy również mieć do czynienia z innymi danymi, kluczowymi dla konkretnych dziedzin.

Spis treści

[1 Faza kompletowania elementów 3](#_Toc9545631)

[2 Schemat połączeń 4](#_Toc9545632)

[3 Arduino IDE – opis kodu 6](#_Toc9545633)

[4 Matlab m.plik – opis kodu 7](#_Toc9545634)

[5 Pierwsza faza danych, porównanie z danymi prognozowymi 9](#_Toc9545635)

[6 Zaobserwowanie przebiegi 10](#_Toc9545636)

[7 Wnioski 10](#_Toc9545637)

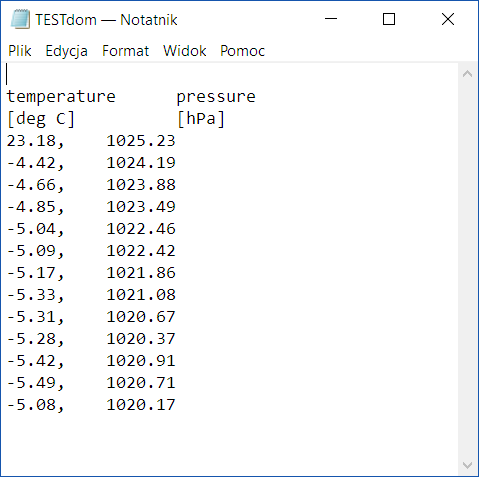
# 

# 1 Faza kompletowania elementów

Aby zacząć obserwacje skierowałem się w stronę środowiska Arduino. Zaznajomiony wcześniej z ową platformą prace nie stanowiły większych trudności.

Sercem układu jest połączenie trzech elementów: Arduino nano, czytnika kard micro sd oraz BMP180. Pobieranie danych odbywa się poprzez czujnik bmp. Odczytuje on zarówno temperaturę jak i ciśnienie. Nadzorowane jest to poprzez mikrokontroler- arduino, a następnie zapisywane na karcie sd w formacie .txt.

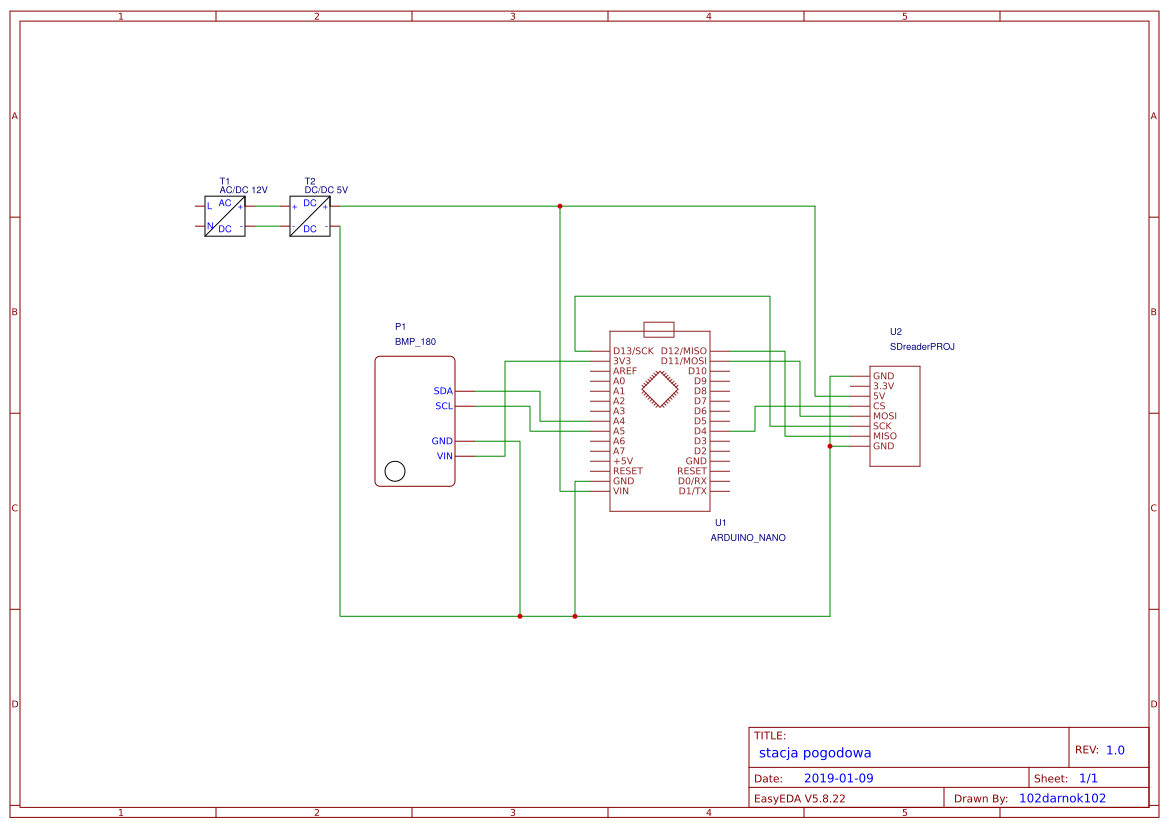
Sam czujnik mierzy ciśnienie z zakresu 300 do 1100 hPa z dokładnością 0,02 hPa.

Rysunek 1  
Przykład tworzonego pliku tekstowego

Następnie dane zgrywane są z karty sd i przetwarzane w środowisku Matlab.

Układ znajdował się na północno wschodniej stronie elewacji, dzięki czemu wykluczone zostały dane które mogłyby być zafałszowane przez nagrzaną powierzchnię przez Słońce.

# 2 Schemat połączeń

Aby układ działał poprawnie należy poprawnie połączyć ze sobą elementy. Czujnik oraz moduł karty sd to osobne urządzenia z punktu widzenia mikrokontrolera. Istotną rolę odgrywa również zasilanie.

Arduino komunikuje się z urządzeniami peryferyjnymi za pomocą interfejsów komunikacyjnych.

Pierwszy z nich (I2C) obsługuje czujnik. Jest to magistrala master- slave. Do transmisji wykorzystywane są jedynie dwie linie. SDA – linia danych oraz SCL- linia zegara. Obydwie linie są dwukierunkowe, dzięki czemu nie potrzeba więcej ścieżek do wysyłania i odbierania informacji. Sama magistrala nie gwarantuje szybkiego przesyłu danych oraz jest ograniczona zasięgiem, jednak prosta budowa i niski koszt zapewniają poprawne działanie dla mniej skomplikowanych układów.

Kolejny (SPI) odpowiada za komunikację z kartą sd. Interfejs ten jest również w układzie master-slave. Składa się z trzech linii, dwóch synchronicznie przesyłających informacje(MOSI – dane do urządzenia peryferyjnego, MISO- dane z urządzenia peryferyjnego) oraz linii zegara(SCK). Wykorzystuje się również dodatkową linię, służącą do wyboru urządzenia slave w przypadku rozleglejszych systemów.

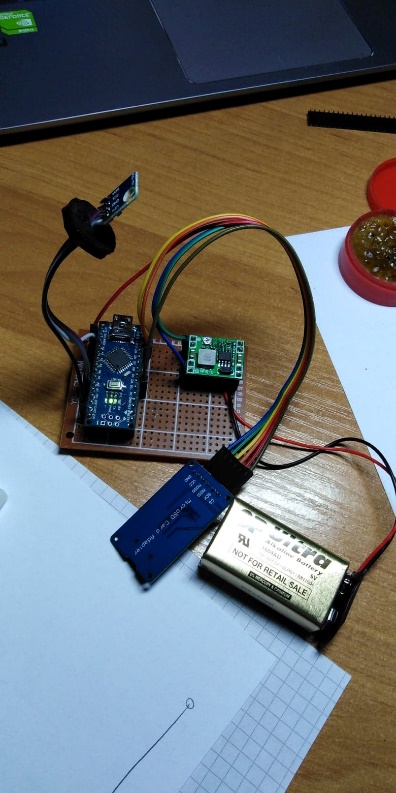
Interfejs SPI składa się z dwóch rejestrów przesuwnych połączonych w licznik pierścieniowy. Linia MISO jest wejściem danych dla urządzenia master, a wyjściem dla slave, natomiast linia MOSI jest wyjściem dla urządzenia master, a wejściem dla slave. W czasie transmisji obie linie są zawsze wykorzystywane. Jednak najczęściej dane są przesyłane tylko przez jedną linię, podczas gdy druga nadaje pustą informację.

Sam interfejs może być wykorzystywany do wgrywania programu do płytki za pomocą odpowiedniego programatora. Wykorzystywany jest również do komunikacji z układami tj.: przetworniki, RTC, EEPROM, pamięci flash.

Równie ważny jest system zasilania. Bez dostarczenia odpowiedniej ilości energii układ elektroniczny nie jest w stanie pracować. Podczas konstrukcji orientacyjnie przeprowadziłem pomiary z których wynikało że cały układ pobiera mniej niż 1W. Zwiększenie poboru następowało w chwili zapisu, jednak i tak nie przekraczało to 1W.

Niefortunnie skierowałem się do zasilania bateryjnego, jednak mogłem to zweryfikować już w fazie projektowania układu że nie był to najlepszy pomysł. Standardowe baterie 9V mają pojemność około 400 mAh, co przy ciągłej pracy układu starczyłoby energii na maksymalnie dwa tygodnie. Problem ten potęgował okres w którym rozpocząłem projekt. Pogoda w miesiącach startu osiągała wartości zerowe co wpływało na pojemność baterii przez co układ działał krócej.

Wykorzystałem jednak przetwornicę(DC/DC step-down) która przetwarzała wartość 9V na napięcie 5V zasilania. Sama przetwornica posiada regulowane napięcie wyjściowe poprzez potencjometr. Zamiana zasilania bateryjnego na zasilacz AC/DC nie stanowiła problemów.

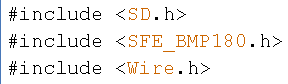
Rysunek 2  
1 skompletowanie elementów, 2 polutowana płytka, 3 zamocowanie w puszce instalacyjnej

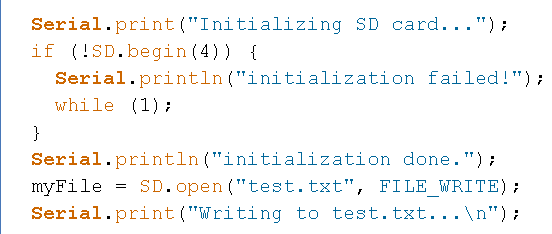
Warto zwrócić uwagę na rodzaj obudowy. Stacja była wystawiona na warunki pogodowe przez co było konieczne użycie puszki z odpowiednim współczynnikiem IP. Sam czujnik został umieszczony w kolanku w celu pobierania bardziej adekwatnych odczytów temperatury niżeli ze środka zamkniętej obudowy.

# 3 Arduino IDE – opis kodu

Arduino zapewnia szeroki zasób bibliotek oraz przykładów dołączonych do nich. Mimo że wcześniej nie miałem do czynienia ani z kartą sd podłączoną do płytki, ani obsługi czujnika, kod nie stanowił problemu. Dołączone biblioteki gwarantują dedykowane funkcje do obsługi danych urządzeń, dzięki czemu ułatwiają programowanie. Głębsza analiza dołączonych przykładów ‘basic’ sprawiły że połączyłem dwa kody w jeden, modyfikując niektóre elementy.

Opis ważniejszych części kodu:

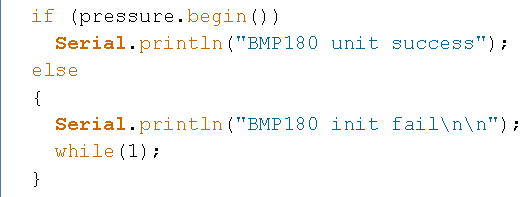
*Załączenie bibliotek obsługujących kartę, czujnik, komunikacji.*

**

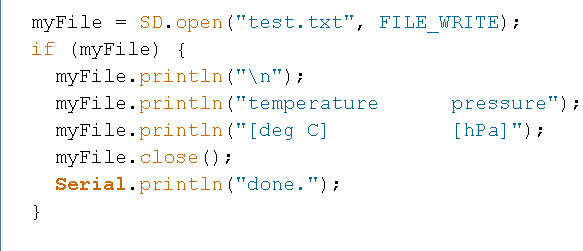
*część wykonywana jednorazowo:*

*Inicjalizacja karty sd, wysłanie komunikatu do monitora portu szeregowego.*

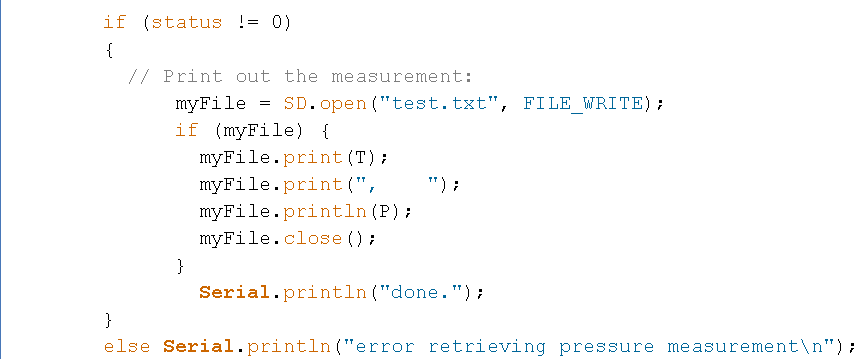
*Testowe sprawdzenie dostępności karty*

**

*Sprawdzenie dostępności czujnika bpm180. Jeżeli nastąpi problem, program zatrzyma się w tym miejscu.*

**

*Utworzenie nagłówka tabeli.*

*część wykonywana w pętli:*

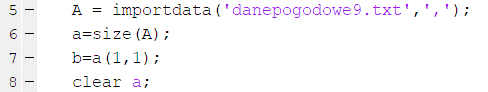
*Pobranie wartości czujnika i zapis do karty sd.*

*W razie błędu, wyświetlenie błędu.*

*Opóźnienie pętli tak aby zapis odbywał się co godzinę*

# 4 Matlab m.plik – opis kodu

Kod napisany w matlabie ma prezentować zebrane dane. Można zauważyć pomiędzy nimi różne zależności, na które zwrócę później uwagę. Podczas pisania kodu napotkałem na problemy związane z danymi. Konieczne było zastosowanie dodatkowych funkcji by zniwelować owe błędy.

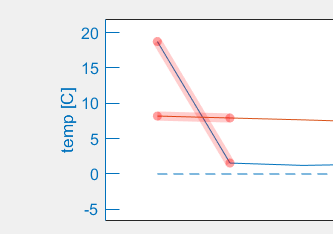
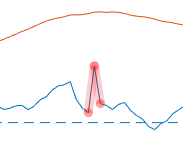


Początek kodu zawiera funkcję pobrania wartości z pliku tekstowego. Należy jednak usunąć nagłówek, gdyż wtedy nie jest tworzona tabela a struktura zawierająca dane oraz plik tekstowy.

Badana jest wielkość tabeli( ilość wierszy), następnie zostaje przypisana pod zmienną. Jest to duże ułatwienie ponieważ jest ona wykorzystywana w następnych częściach kodu. Podczas gromadzenia, wczytując coraz nowsze dane(coraz większe tabelę) liczba wierszy jest sprawdzana automatycznie.

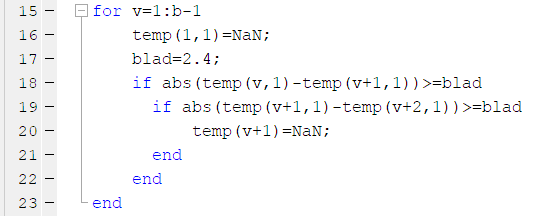


Tabela zostaje podzielona na dwie zmienne odpowiadające odpowiednio temperaturze, w pierwszej kolumnie oraz ciśnieniu w drugiej kolumnie.

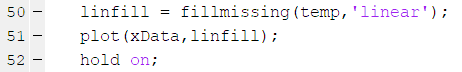
 

Rysunek 3  
przykład różnicy temperatur

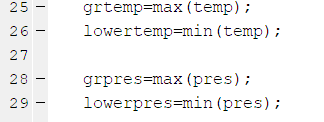
W okresie jesienno -zimowym różnica temperatur pomiędzy temperaturą w pomieszczeniu a temperaturą na dworze różni się znacząco. Prace konserwacyjne takie jak pobranie danych z karty wymagają zabrania i otworzenia puszki. Zabiera to czas, podczas którego temperatura wokół czujnika się zmienia. Po włożeniu karty sd, niezbędne jest ręczne zresetowanie arduino w celu ponownej synchronizacji z nośnikiem pamięci.



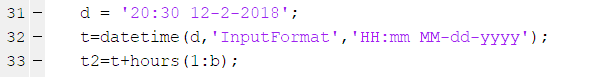
W celu eliminacji owych nieścisłości wykorzystywana jest pętla która wykonuje się tyle razy ile jest wierszy w tabeli. Jako że pierwszy pomiar odbywał się w pomieszczeniu (puszka miała temperaturę pokojową) zostaje nadpisana owa wartość jako pusta. Powtórzone jest to dla wszystkich wartości spełniających warunek. Zmienna *blad* została dobrana na podstawie obserwacji generowanego kodu i wiadomości kiedy następowały pobrania danych. Tolerancja błędu musiała zostać zmniejszona w okresie wiosennym kiedy różnica pomiędzy nocą a porankiem przekraczały ustaloną wartością.

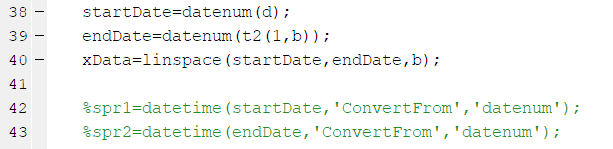


Uzupełnienie danych następuje z użyciem odpowiedniej funkcji.



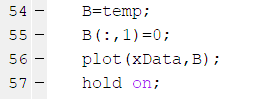
Sprawdzane są największe i najmniejsze wartości ciśnienia i temperatury.





Jako że nie ma jednoznacznego odniesienia konkretnych danych do czasu pobrania próbki, koniecznym jest utworzenie *datatime*. Pierwsza dana jest zapisana jest jako stała, ostatnia zostaje obliczona na podstawie ilości próbek.

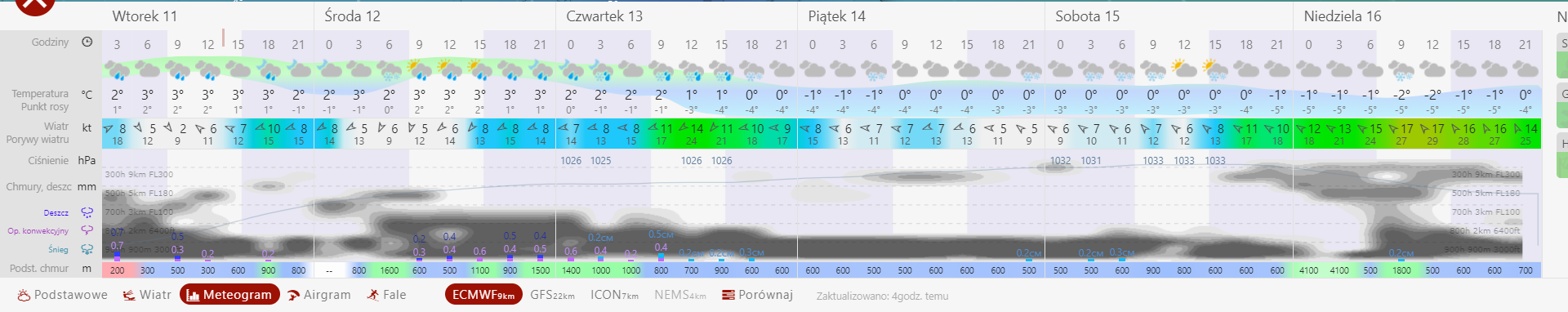
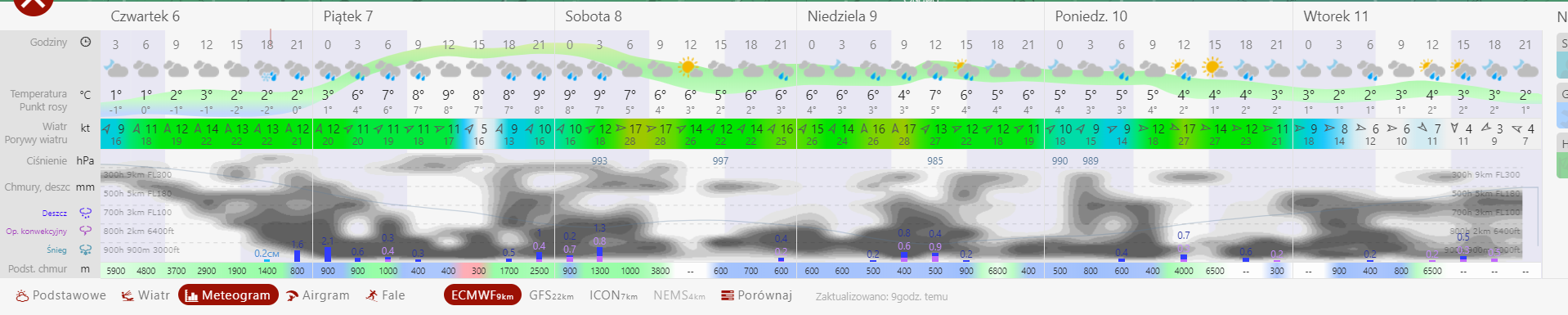
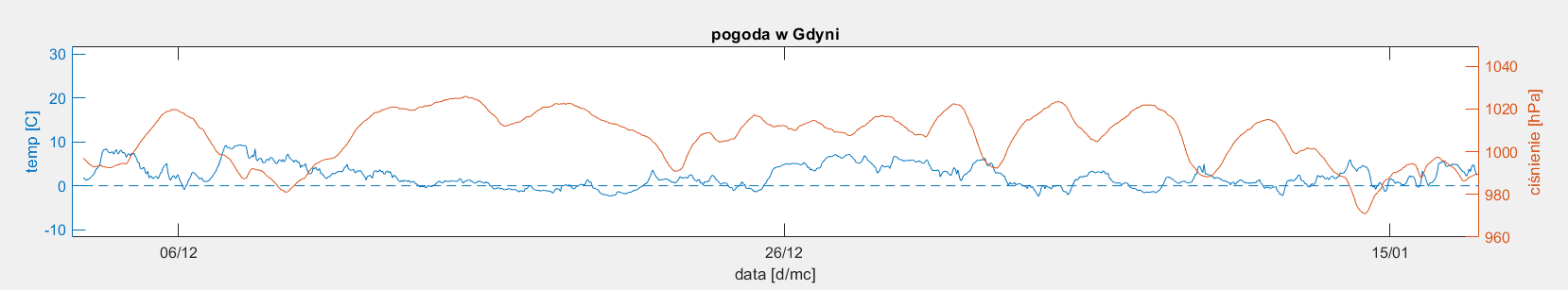
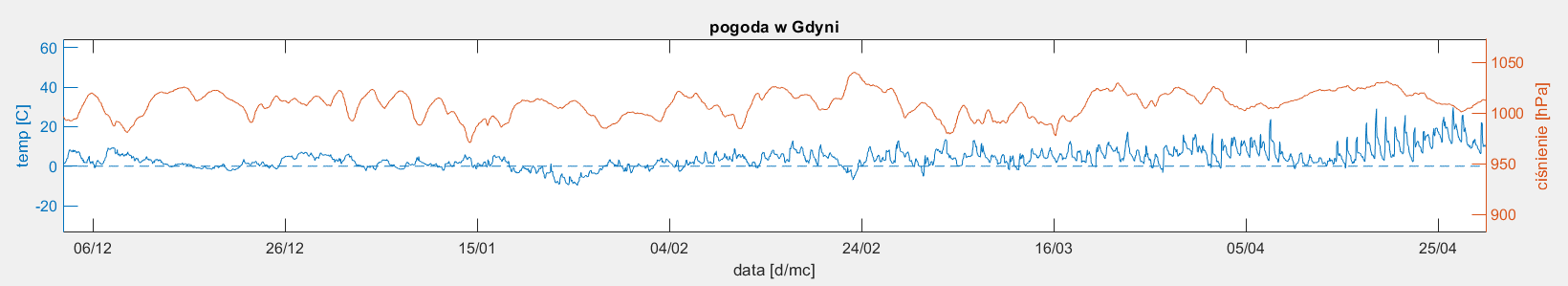
Zakomentowane zmienne służą do sprawdzenia czy czas określony przez program zgadza się z rzeczywistym czasem ostatniej próbki. Spełniło to swoje zadanie, ponieważ okazało się że gdy odstęp czasu sprawdzenia kolejnych danych był większy niż miesiąc, pojawiały się dodatkowe zapisy próbek. Spowodowane jest to funkcją w środowisku Arduino IDE delay(). Ma ona ustaloną wartość przez co pomiar odbywa się orientacyjnie co godzinę. Mimo że czas pobrania różnych zmiennych czy obsługi peryferiów zdaje się być natychmiastowy to przez kilkaset krotności powtórzenia tych samych operacji, czas napiętrzył się tak że stworzone zostały nadmiarowe pomiary.



Wyrysowanie linii zera stopni Celsjusza.

# 5 Pierwsza faza danych, porównanie z danymi prognozowymi

Rysunek 4  
Prezentacja zebranych danych z całego okresu działalności stacji

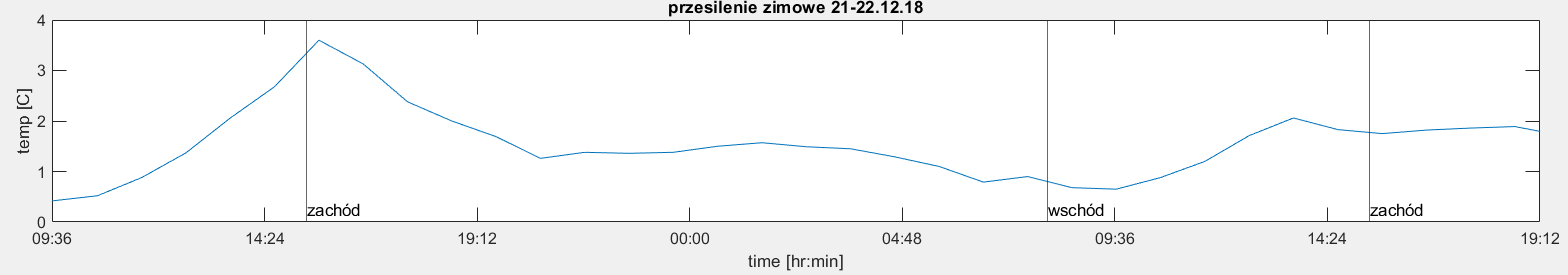
 Na przestrzeni od grudnia do kwietnia zostało pobranych ponad 3,5 tysiąca próbek. 

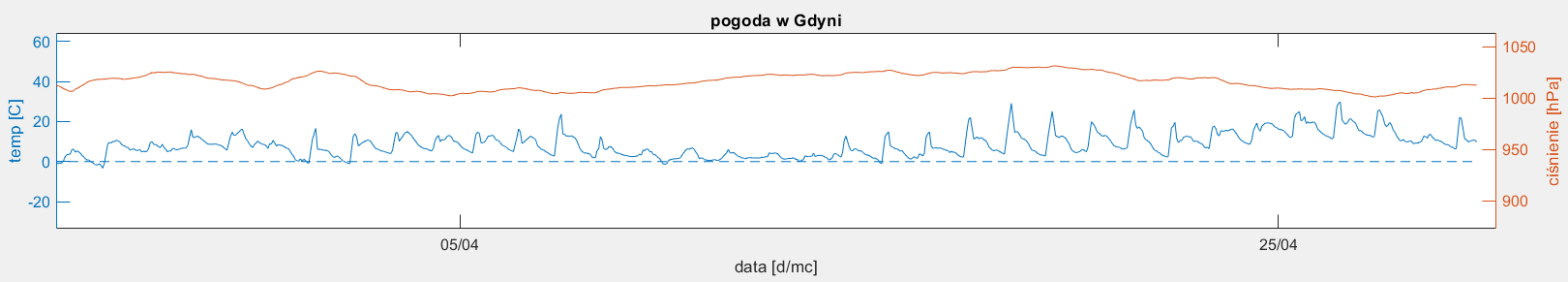
Zbierając dane gromadziłem informacje ze stacji pogodowej zawierające bardziej szczegółowy opis zjawisk atmosferycznych. Pozwoliło to np. na orientacyjne określenie opadów na podstawie wartości ciśnienia.

Przykładowo 6 grudnia panował wyż ciśnienia. Osiągnęło swoje maksimum i zaczęło spadać, skutkiem tego był niż z którym przyszły chmury opadowe.

# 6 Zaobserwowanie przebiegi

1. Przykład przesilenia zimowego.



1. Przykład powtarzalności części danych:

Na wykresie widoczny jest powtarzający się patern. Powtarzający się element oznacza zmianę temperatury w ciągu doby.

# 7 Wnioski

Na podstawie zgromadzonych informacji i ich analizie możliwa jest ocena pogody na najbliższy czas. W zależności od pory roku analiza powinna odbywać się pod innym kątem ponieważ dane inaczej na siebie oddziałują.

W celu lepszego określenia warunków atmosferycznych możliwe jest dołączenie czujnika wilgotności powietrza który w połączeniu z czujnikiem ciśnienia określałby opady deszczu czy śniegu.

Kolejnym etapem projektu mogłoby być podłączenie stacji pod sieć Ethernet w celu zdalnego odczytu danych. Rozwiązało by to problem synchronizacji z czasem rzeczywistym.

Gdyby powielić projekt i rozmieścić stacje w znacznej odległości od siebie ( co najmniej trzy stacje), możliwe jest określenie kierunku wiatru na podstawie różnicy ciśnienia. W połączeniu ze wcześniejszymi ‘ulepszeniami’, możliwe jest określenie ruchów chmur i przewidzenia pogody z większym wyprzedzeniem.