Jakub Hryciów 27.01.2025

Teleinformatyka 3 rok

Numer albumu: 29197

POLITECHNIKA MORSKA W SZCZECINIE
WYDZIAŁ INFORMATYKI I TELEKOMUNIKACJI

Transmisja Danych

Pittdroid Strato

System Elektroniki Balonu Stratosferycznego:

Specyfikacja i Wdrożenie

Szczecin 2025

Spis treści

Cel projektu

Celem projektu balona stratosferycznego jest stworzenie kompleksowego systemu umożliwiającego przeprowadzenie badań atmosferycznych i technologicznych na dużych wysokościach, z wykorzystaniem zaawansowanych rozwiązań elektronicznych. Projekt ma na celu zaprojektowanie i zintegrowanie innowacyjnego układu elektronicznego, który umożliwi: precyzyjne monitorowanie warunków atmosferycznych (temperatury, ciśnienia, wilgotności), rejestrację parametrów lotu (wysokości, prędkości, trajektorii), a także przesył danych telemetrycznych w czasie rzeczywistym.

W ramach projektu przewiduje się:

1. Opracowanie zaawansowanego systemu sensorów zdolnych do działania w ekstremalnych warunkach stratosferycznych, gdzie występują niskie temperatury i niskie ciśnienia.
2. Implementację niezawodnych systemów komunikacji bezprzewodowej, umożliwiających przesyłanie danych na duże odległości, w tym komunikację dwukierunkową z naziemną stacją kontrolną.
3. Zastosowanie rozwiązań pozwalających na precyzyjne pozycjonowanie GPS oraz śledzenie trajektorii lotu w czasie rzeczywistym.
4. Testowanie i weryfikację trwałości i niezawodności zastosowanej elektroniki w warunkach symulujących środowisko stratosferyczne, w celu minimalizacji ryzyka awarii podczas misji.

Dodatkowym celem projektu jest rozwój wiedzy i umiejętności zespołu w zakresie projektowania systemów embedded, komunikacji bezprzewodowej, analizy danych oraz integracji różnych elementów systemu w jedną funkcjonalną całość. Efektem końcowym będzie przygotowanie w pełni funkcjonalnego balona stratosferycznego, który posłuży jako platforma testowa dla dalszych badań atmosferycznych, technologicznych, a także edukacyjnych.

Założenia projektu

1. **Cele naukowe i badawcze**
    - Zbieranie danych meteorologicznych, takich jak temperatura, ciśnienie atmosferyczne, wilgotność powietrza oraz poziom promieniowania w stratosferze.
    - Analiza trajektorii lotu i warunków aerodynamicznych w stratosferze.
    - Testowanie nowych technologii elektronicznych w ekstremalnych warunkach atmosferycznych.
2. **Specyfikacja systemu elektronicznego**
    - **Platforma sensoryczna:**
      a) Wysokiej klasy czujniki do pomiaru temperatury, ciśnienia, wilgotności oraz poziomu promieniowania UV i kosmicznego.
      b) Moduły GPS i akcelerometry do śledzenia pozycji, prędkości i orientacji balona.
    - **Komunikacja:**
      a) System transmisji danych telemetrycznych w czasie rzeczywistym do naziemnej stacji kontrolnej.
      b) Dwukierunkowa komunikacja w celu przesyłania komend do systemu pokładowego.
    - **Zasilanie:**
      a) Wykorzystanie baterii odpornych na niskie temperatury oraz paneli słonecznych jako alternatywnego źródła zasilania.
      b) System zarządzania energią zapewniający optymalną pracę urządzeń elektronicznych.
    - **Sterowanie i rejestracja:**
      a) Mikrokontrolery lub jednostki embedded do zarządzania pracą urządzeń pokładowych i zbierania danych.
      b) System pamięci masowej do lokalnego przechowywania danych w przypadku utraty łączności.
3. **Konstrukcja i integracja systemów**
    - Zastosowanie obudów odpornych na niskie temperatury i ciśnienie atmosferyczne w stratosferze.
    - Minimalizacja masy urządzeń elektronicznych, by dostosować się do ograniczeń nośności balonu.

- Zapewnienie niezawodnej integracji komponentów elektronicznych z platformą nośną.

4. **Warunki środowiskowe i testy wytrzymałościowe**
   - Projekt systemu musi uwzględniać ekstremalne warunki panujące w stratosferze, takie jak:
     a) Temperatura do -70°C lub niższa.
     b) Niskie ciśnienie atmosferyczne (około 1% ciśnienia na poziomie morza).
     c) Silne promieniowanie UV i kosmiczne.
   - Przeprowadzenie testów w komorach symulacyjnych w celu weryfikacji odporności systemu elektronicznego na ekstremalne warunki.
5. **Ograniczenia prawne i bezpieczeństwo**
   - Projekt musi być zgodny z regulacjami dotyczącymi wypuszczania balonów w przestrzeń powietrzną, w tym uzyskanie stosownych pozwoleń od odpowiednich organów.
   - Wdrożenie procedur bezpieczeństwa w celu ograniczenia ryzyka awarii lub kolizji z samolotami.
6. **Proces realizacji projektu**
   - Podział projektu na etapy:
     a) Projektowanie systemu elektronicznego.
     b) Wybór i zakup komponentów.
     c) Integracja i testowanie.
     d) Przygotowanie do misji i przeprowadzenie startu.
   - Dokumentacja wszystkich etapów projektu w celu umożliwienia dalszego rozwoju i analiz.
7. **Zakładane rezultaty**
   - Uzyskanie sprawnego systemu elektronicznego, zdolnego do pracy w warunkach stratosferycznych.
   - Zebranie i analiza danych z przeprowadzonej misji.
   - Zwiększenie wiedzy w zakresie projektowania i testowania urządzeń w ekstremalnych warunkach.

Wybór rozwiązania projektowego

1. Struktura systemu elektronicznego

Aby zapewnić niezawodność i wielofunkcyjność, system elektroniczny został podzielony na następujące moduły:

- Moduł sensoryczny: Odpowiedzialny za pomiar kluczowych parametrów atmosferycznych (temperatura, ciśnienie, wilgotność, promieniowanie UV).
- Moduł komunikacyjny: Zapewniający łączność z naziemną stacją kontrolną.
- Moduł zasilania: Odpowiedzialny za dostarczenie stabilnego zasilania do wszystkich podsystemów.
- Moduł rejestracji danych: Zapisujący dane na lokalnych nośnikach pamięci.

2. Wybór komponentów

a) Czujniki:
   - Wybrano wysokoprecyzyjne czujniki MEMS (Micro-Electro-Mechanical Systems), które cechują się małą masą, niskim poborem energii i wysoką dokładnością.
   - Przykłady:
- Czujnik temperatury i ciśnienia: **BME280** – wielofunkcyjny, kompaktowy i odporny na niskie ciśnienia.

b) System lokalizacji:
   - Zastosowano moduł GPS z obsługą wysokości powyżej 40 km, u-blox NEO-M6, który umożliwia precyzyjne śledzenie trajektorii lotu.

c) Komunikacja:
   - Do przesyłu danych telemetrycznych wybrano moduły radiowe LoRa, które oferują duży zasięg transmisji przy niskim poborze energii.

d) Zasilanie:
   - Baterie litowo-polimerowe (Li-Po) o zwiększonej odporności na niskie temperatury.

e) Sterowanie:
   - Jako jednostkę centralną wybrano mikrokontroler Arduino Uno R3, który zapewnia wystarczającą moc obliczeniową, niski pobór energii oraz szeroką kompatybilność z peryferiami.

3. Optymalizacja energetyczna

Ograniczono pobór energii poprzez zastosowanie energooszczędnych komponentów oraz trybów oszczędzania energii w mikrokontrolerze.

4. Integracja i testy

Wszystkie podzespoły zostały zaprojektowane w taki sposób, aby umożliwić łatwą integrację i modułowość systemu. Przewidziano testy w komorze klimatycznej, które odwzorują warunki panujące w stratosferze.

5. Uzasadnienie wyboru rozwiązania

- **Niezawodność:** Wybrane komponenty zostały sprawdzone w warunkach zbliżonych do rzeczywistych w podobnych projektach.
- **Optymalizacja masy i energii:** Zastosowanie lekkich i energooszczędnych komponentów pozwala na wydłużenie czasu misji przy jednoczesnym zmniejszeniu obciążenia balonu.
- **Koszty:** Przyjęte rozwiązania uwzględniają budżet projektu, wybierając komponenty dostępne na rynku przy zachowaniu wysokiej jakości.
- **Elastyczność:** Modularna konstrukcja umożliwia łatwą wymianę lub rozbudowę systemu.

Opis projektu, opis budowy i zasady działania urządzenia.

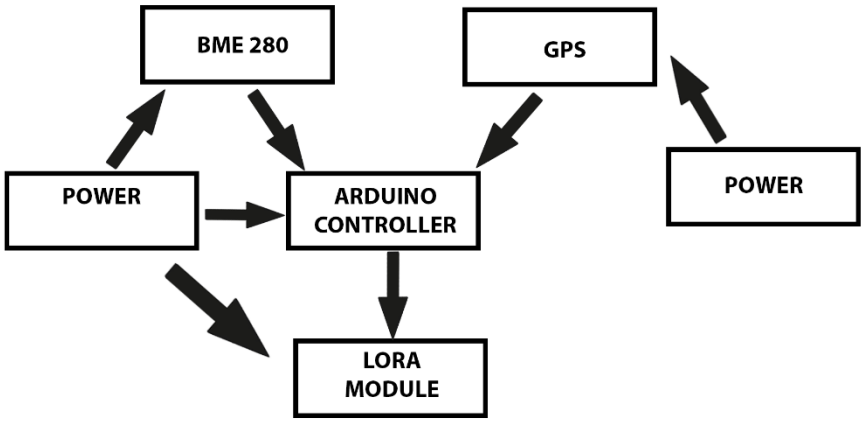
Opis:

Projekt balona stratosferycznego skupia się na stworzeniu kompleksowego systemu elektronicznego zdolnego do pracy w ekstremalnych warunkach stratosferycznych. Urządzenie składa się z modułu pomiarowego, komunikacyjnego, sterującego oraz zasilającego . Głównym celem jest monitorowanie parametrów atmosferycznych, zbieranie danych telemetrycznych i ich przesyłanie do stacji naziemnej. Elektronika została zaprojektowana z naciskiem na odporność na niskie temperatury, niskie ciśnienie oraz promieniowanie kosmiczne. Elementy systemu zostały tak dobrane by móc w pełni zoptymalizować pobór energii z akumulatora, a co za tym idzie jak najdłuższe działanie komponentów.

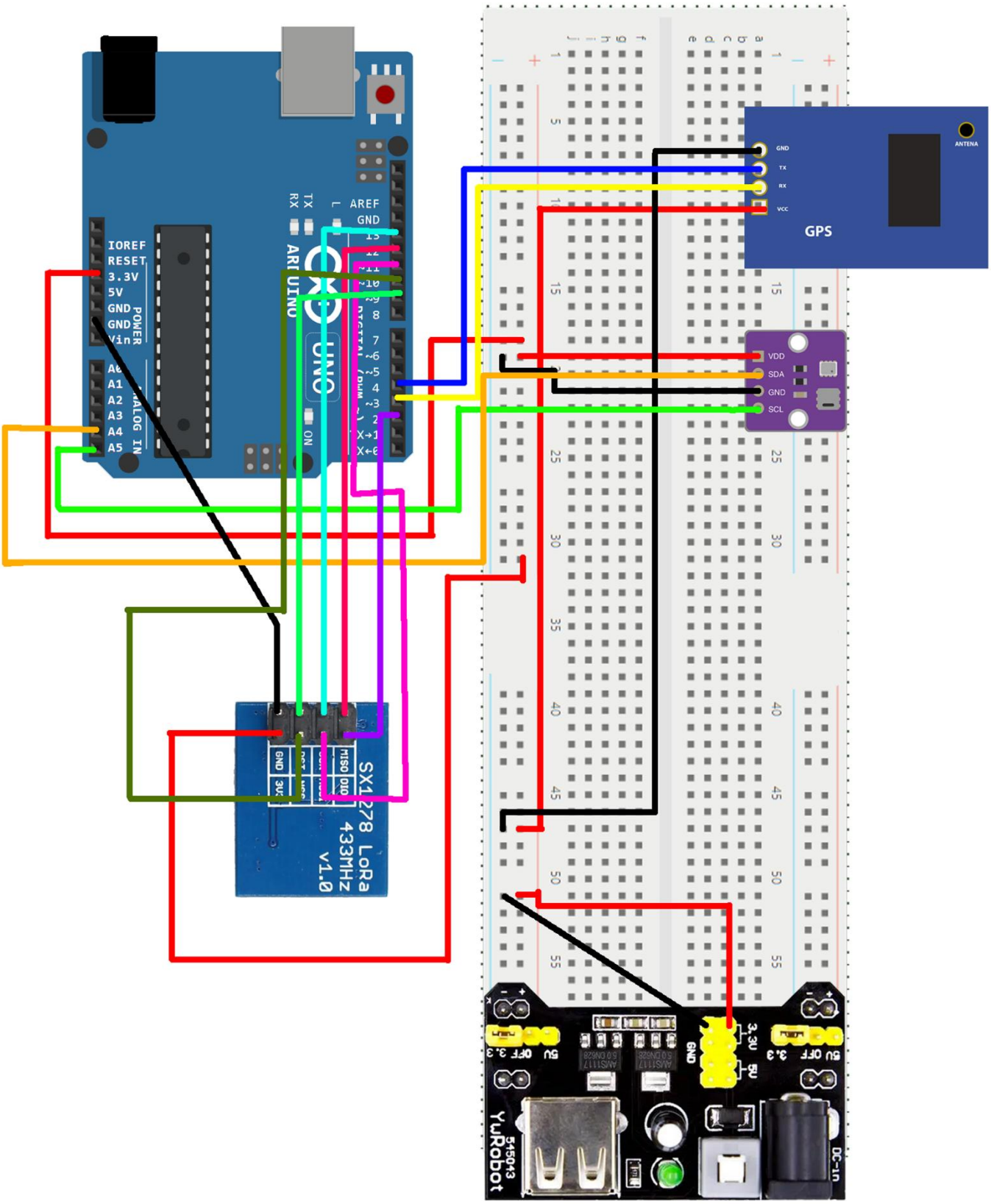
Schematy blokowe

Schemat blokowy przedstawia podział systemu na moduły:

- Moduł sensoryczny: Czujniki temperatury, ciśnienia, wilgotności, promieniowania.
- Moduł komunikacyjny: Radiowy moduł LoRa + GSM.
- Moduł sterowania: Mikrokontroler STM32.
- Nośnik: Obudowa z materiałów izolacyjnych.



Schemat podglądowy układu


GPS
ANTENA
GND
TX
RX
VCC
VDD
SDA
GND
SCL
UNO
ARDUINO
IOREF
RESET
3.3V
5V
GND
GND
Vin
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN
AREF
GND
SX1278 LoRa
433MHz
v1.0
YwRobot
515013
DC-in

Obliczenia i dobór elementów:

1. BME280 – Czujnik ciśnienia, temperatury i wilgotności

    1. **Parametry techniczne:**
        - **Zasilanie:** 1.8–3.6 V
        - **Pobór prądu:**
            - Tryb pracy: 2.0 µA (tryb niskiej mocy) do 720 µA (tryb wysokiej wydajności).
            - Tryb czuwania: ~0.1 µA.
        - **Interfejs:** I2C lub SPI.
    2. **Źródło zasilania:**
        - Napięcie 3.3 V (zgodne z poziomem logicznym mikrokontrolera).
    3. **Obliczenie poboru energii:**

    Przy założeniu pracy w trybie niskiej mocy przez 90% czasu i trybu wysokiej wydajności przez 10% czasu:

    - Średni pobór prądu:
    $$I_{\text{średni}} = (0.9 \times 2\,\mu\text{A}) + (0.1 \times 720\,\mu\text{A}) = 74.2\,\mu\text{A}.$$
    - Przy zasilaniu 3.3 V:
    $$P = 3.3\,\text{V} \times 74.2\,\mu\text{A} = 0.24486\,\text{mW}.$$

2. GPS NEO-6M-001 – Moduł lokalizacji GPS
    1. **Parametry techniczne:**
    - **Zasilanie:** 2.7–3.6 V.
    - **Pobór prądu:**
        - Tryb pracy: 45–65 mA (standard).
        - Tryb oszczędzania energii: 11 mA.
    - **Interfejs:** UART.
    2. **Obliczenie poboru energii:**

    Przy założeniu, że moduł GPS pracuje przez 100% czasu w standardowym trybie:

        - Pobór prądu: 50 mA (średnia wartość).
        - Pobór mocy:

    $$P = 3.3\,\text{V} \times 50\,\text{mA} = 165\,\text{mW}.$$

Podsumowanie poboru energii:

Sumaryczny pobór prądu przez urządzenia:

$$I_{\text{total}} = 74.2\,\mu\text{A} + 3.24\,\text{mA} + 50\,\text{mA} \approx 53.3\,\text{mA}.$$

Przy zasilaniu 3.3 V:

$$P_{\text{total}} = 3.3\,\text{V} \times 53.3\,\text{mA} \approx 176\,\text{mW}.$$

Dobór baterii:

Przy założeniu czasu pracy 24 godziny:

$$Q = I_{\text{total}} \times t = 53.3\,\text{mA} \times 24\,\text{h} = 1279.2\,\text{mAh}.$$

Wybrana bateria: Li-Po 3.7 V, 2000 mAh (zapewnia rezerwę energetyczną).

Dodatkowo, biorąc pod uwagę elastyczność projektu, można zastosować panel słoneczny zapewniający możliwość ładowania akumulatora.

Opis rozwiązań konstrukcyjnych:

Sposoby mocowania elementów:

a) Płytka stykowa

- Płytka stykowa stanowi główny element do montażu prototypowego układu elektronicznego.
- Została przyklejona do dolnej powierzchni obudowy za pomocą taśmy dwustronnej o dużej przyczepności, co zapewnia stabilność i zapobiega przesuwaniu się płytki podczas pracy.

Rozmiar płytki (np. 830 punktów kontaktowych) został dobrany tak, aby umożliwić podłączenie wszystkich modułów i czujników.

b) Arduino Uno

- Arduino Uno zostało przymocowane do dolnej części obudowy przy użyciu dystansów plastikowych (M3) o długości 10 mm oraz śrub.

Takie rozwiązanie zapewnia zarówno stabilność urządzenia, jak i łatwość demontażu w przypadku serwisowania.

c) Kable czujników i modułów

- Przewody łączące czujniki z płytką stykową zostały przymocowane do ścianek obudowy za pomocą samoprzylepnych uchwytów kablowych oraz opasek zaciskowych.
- Przewody są prowadzone w sposób zorganizowany, aby uniknąć plątania i zapewnić łatwy dostęp do poszczególnych elementów.

d) Moduły czujników

- **BME280:**

Czujnik BME280 został zamontowany w pobliżu otworów wentylacyjnych w obudowie, aby umożliwić dokładny pomiar warunków atmosferycznych. Moduł jest podłączony do płytki stykowej za pomocą przewodów Dupont, które zapewniają łatwość podłączenia i stabilność połączeń.

- **GPS NEO-6M:**

Moduł GPS jest zamocowany w górnej części obudowy, a jego antena została wyprowadzona na zewnątrz przez dedykowany otwór, aby zapewnić optymalny odbiór sygnału satelitarnego.

Mocowanie modułu odbywa się za pomocą taśmy piankowej dwustronnej.

- **Lora SX1278 V4:**

Moduł LoRa został umieszczony w pobliżu płytki stykowej, z anteną wyprowadzoną na zewnątrz przez otwór w obudowie. Antena została przytwierdzona do zewnętrznej części obudowy za pomocą złącza SMA.

Wybór obudowy:

a) Rodzaj obudowy

- Obudowa wykonana z tworzywa ABS, charakteryzującego się wysoką odpornością na uderzenia, niską masą oraz łatwością obróbki.
- Wybrano obudowę o wymiarach 200 × 150 × 100 mm, co zapewnia wystarczającą przestrzeń na płytkę stykową, Arduino Uno, okablowanie oraz moduły czujników.

b) Właściwości obudowy

- Szczelność: Obudowa została wyposażona w gumowe uszczelki na łączeniach, co zapewnia ochronę przed kurzem i wilgocią.
- Otwory wentylacyjne: W pobliżu czujnika BME280 wykonano otwory wentylacyjne zabezpieczone siatką przeciwpyłową, co pozwala na dokładny pomiar ciśnienia i temperatury.
- Wyprowadzenia: Obudowa posiada dedykowane otwory na przewody oraz anteny, które zostały uszczelnione przy pomocy gumowych przepustów.

c) Zabezpieczenia termiczne

- Dla ochrony urządzenia przed niskimi temperaturami na dużych wysokościach, wewnątrz obudowy umieszczono warstwę pianki izolacyjnej. Pianka minimalizuje wpływ ekstremalnych temperatur na elektronikę.

Organizacja wewnętrzna obudowy

- Moduły elektroniczne zostały rozmieszczone w sposób umożliwiający ich łatwe serwisowanie oraz eliminację zakłóceń.
- Kable zasilające oraz sygnałowe zostały spięte w wiązki i uporządkowane wzdłuż krawędzi obudowy za pomocą opasek zaciskowych.
- Zasilanie (bateria) zostało zamocowane na piankowej podkładce antywibracyjnej, aby zminimalizować wpływ wstrząsów.

Uwagi końcowe

- Konstrukcja obudowy i rozmieszczenie elementów umożliwia szybki demontaż oraz wymianę uszkodzonych komponentów w trakcie serwisowania.
- Całość konstrukcji została przystosowana do pracy w warunkach niskich temperatur, wibracji oraz zmiennych warunków atmosferycznych.

Lista elementów projektu:

1. Moduły i czujniki

Nazwa elementu	Model/Specyfikacja	Ilość	Uwagi
Czujnik ciśnienia, wilgotności i temperatury	BME280	1	Moduł I2C
Moduł GPS	NEO-6M-001	1	Z anteną zewnętrzną
Moduł komunikacyjny LoRa	SX1278 V4 (433 MHz)	1	Antena zewnętrzna, złącze SMA
Mikrokontroler	Arduino Uno R3	1	Oryginalny lub kompatybilny

2. Elementy pasywne i złącza

Nazwa elementu	Specyfikacja	Ilość	Uwagi
Przewody połączeniowe	Dupont, męskie/żeńskie	20	Do łączenia czujników i modułów
Płytka stykowa	830 punktów kontaktowych	1	Prototypowe połączenie elementów
Złącza SMA	Standardowe SMA	1	Do anteny LoRa

3. Zasilanie

Nazwa elementu	Specyfikacja	Ilość	Uwagi
Bateria litowo-polimerowa	7.4 V, 2200 mAh	1	Wystarczająca na kilka godzin pracy
Przetwornica DC-DC	Step-down (np. LM2596)	1	Obniża napięcie do 5V
Kabel USB	USB typu B	1	Do zasilania Arduino

4. Obudowa i akcesoria mechaniczne

Nazwa elementu	Specyfikacja	Ilość	Uwagi
Obudowa	ABS, 200 × 150 × 100 mm	1	Z otworami na anteny i czujniki
Dystanse plastikowe	M3, 10 mm	4	Mocowanie Arduino
Taśma dwustronna	Piankowa	1	Mocowanie płytki stykowej
Uchwyt na baterię	Plastikowy lub taśma rzepowa	1	Mocowanie baterii w obudowie
Opaski zaciskowe	Nylonowe	10	Organizacja kabli

5. Inne elementy pomocnicze

Nazwa elementu	Specyfikacja	Ilość	Uwagi
Antena GPS	Aktywna	1	Wyprowadzona na zewnątrz obudowy
Antena LoRa	433 MHz	1	Zewnętrzna, do złącza SMA
Uszczelki gumowe	Średnica zależna od kabli	3	Uszczelnienie otworów w obudowie

6. Narzędzia do montażu

Nazwa narzędzia	Specyfikacja	Uwagi
Lutownica	30-60 W	Do lutowania złącz Dupont
Wkrętarka lub śrubokręt	Krzyżakowy, M3	Montaż dystansów i obudowy
Multimetr	Standardowy	Sprawdzanie połączeń i zasilania

Uwagi końcowe

- Wszystkie elementy zostały dobrane z uwzględnieniem łatwej dostępności na rynku.
- Lista może być uzupełniona o dodatkowe komponenty, takie jak diody LED do sygnalizacji stanu urządzenia lub rezystory do testów prototypowych.
- Przed montażem warto przygotować zapasowe przewody i złącza Dupont.

Opis uruchomienia urządzenia

Uruchomienie urządzenia polega na kilku krokach, które obejmują podłączenie elementów, wgranie odpowiedniego oprogramowania, testowanie funkcji urządzenia oraz jego kalibrację przed finalnym montażem w obudowie. Poniżej przedstawiono szczegółowy opis poszczególnych etapów uruchamiania projektu balonu stratosferycznego.

1. Podłączenie elementów do Arduino

- **Płytka stykowa**: Elementy takie jak czujniki (BME280, GPS NEO-6M), moduł LoRa (SX1278), oraz przewody zasilające muszą być poprawnie połączone z płytką stykową. Należy upewnić się, że wszystkie przewody są prawidłowo podłączone do odpowiednich pinów Arduino.
- **Arduino Uno**: Mikrokontroler Arduino Uno musi być prawidłowo podłączony do płytki stykowej. Należy zwrócić uwagę, aby elementy były podłączone zgodnie z wymaganiami zasilania i komunikacji (np. I2C dla BME280, GPIO dla LoRa).

2. Zasilanie

- **Bateria Li-Po**: Urządzenie zasilane jest akumulatorem litowo-polimerowym o napięciu 7.4V (2200mAh). Bateria powinna być podłączona do przetwornicy DC-DC, która zapewnia obniżenie napięcia do 5V.
- **Przetwornica DC-DC**: Zasilanie do Arduino oraz pozostałych modułów (czujników, LoRa) jest dostarczane poprzez przetwornicę step-down (np. LM2596), która konwertuje napięcie z 7.4V na stabilne 5V.

3. Wgrywanie kodu na Arduino

- **Arduino IDE**: Aby wgrać kod do mikrokontrolera Arduino, należy użyć oprogramowania Arduino IDE. Po podłączeniu Arduino do komputera za pomocą kabla USB, w programie należy wybrać odpowiednią płytkę i port.
    - **Wgrywanie programu**: Program powinien obsługiwać następujące funkcje:
    - Odczyt danych z czujników (BME280 i GPS).
    - Przesyłanie danych za pomocą modułu LoRa.
    - Obsługa komunikacji I2C z BME280 oraz serialowego odbioru danych z GPS.
- **Weryfikacja działania**: Po załadowaniu programu na płytkę należy upewnić się, że urządzenie działa poprawnie, wyświetlając dane na monitorze szeregowym w Arduino IDE.

4. Testowanie działania urządzenia

- **Moduł LoRa**: Należy sprawdzić, czy moduł LoRa poprawnie komunikuje się z odbiornikiem LoRa. Test polega na przesyłaniu danych z czujników (np. temperatury, ciśnienia) na odległość, aby upewnić się, że sygnał jest prawidłowo odbierany przez drugi moduł LoRa.
- **Test GPS**: Sprawdzamy, czy moduł GPS prawidłowo odbiera sygnał satelitarny i zwraca współrzędne geograficzne. Należy także sprawdzić, czy urządzenie prawidłowo wyświetla te dane na monitorze szeregowym.
- **Czujnik BME280**: Weryfikujemy odczyty z czujnika ciśnienia, wilgotności i temperatury, porównując je z rzeczywistymi wartościami w danym środowisku. Jeśli wartości są niewłaściwe, należy przeprowadzić kalibrację czujników.

5. Montaż urządzenia w obudowie

- **Obudowa:** Po zakończeniu testów, urządzenie powinno zostać zamontowane w odpowiedniej obudowie. Obudowa powinna chronić wszystkie elementy przed uszkodzeniami mechanicznymi oraz wpływem czynników atmosferycznych. W przypadku balonu stratosferycznego, obudowa musi być szczelna, aby chronić urządzenie przed wilgocią i zanieczyszczeniami.

- **Mocowanie czujników:** Czujniki, takie jak GPS i BME280, należy zamocować w taki sposób, aby miały zapewnioną odpowiednią cyrkulację powietrza (czujnik BME280) oraz widoczność dla sygnałów GPS. Dobrze jest również zabezpieczyć antenę LoRa, aby zapewnić jej odpowiednią transmisję sygnału.

6. Kalibracja czujników

- BME280: Czujnik BME280 może wymagać kalibracji, zwłaszcza w odniesieniu do pomiarów temperatury i wilgotności. Należy sprawdzić poprawność odczytów w różnych warunkach i, jeśli to konieczne, dostosować wartości w programie.
- GPS NEO-6M: Moduł GPS wymaga czasu na złapanie sygnału GPS, zwłaszcza w pomieszczeniu. W trakcie testów należy zapewnić, by antena GPS była ustawiona w kierunku nieba, aby uzyskać jak najlepszy sygnał.

7. Testy w rzeczywistych warunkach

- **Test w terenie:** Urządzenie należy przetestować w warunkach zbliżonych do tych, które będą występować podczas lotu balonu stratosferycznego. Testy powinny obejmować pomiary temperatury, ciśnienia oraz wilgotności, a także przesyłanie danych przez LoRa.
- **Wysokość i zasięg transmisji:** W trakcie testów warto sprawdzić, jak urządzenie zachowuje się w różnych warunkach atmosferycznych oraz na większych

wysokościach. Dodatkowo należy upewnić się, że urządzenie może przesyłać dane na odpowiednią odległość.

8. Przygotowanie do wypuszczenia balonu

- Po przeprowadzeniu wszystkich testów i upewnieniu się, że urządzenie działa prawidłowo, można przystąpić do przygotowań do wypuszczenia balonu stratosferycznego. Należy sprawdzić, czy bateria zapewni odpowiednią długość pracy urządzenia oraz czy system LoRa będzie miał odpowiedni zasięg.

Oprogramowanie

Opis oprogramowania

Oprogramowanie dla projektu balonu stratosferycznego będzie odpowiedzialne za obsługę czujników, komunikację z modułem LoRa, zbieranie danych z GPS, a także zarządzanie zasilaniem urządzenia. Poniżej przedstawiono szczegółowy opis oprogramowania, które zostanie zaimplementowane na mikrokontrolerze Arduino Uno, w tym bibliotekach, funkcjach oraz sposobie działania poszczególnych komponentów.

1. Wymagane biblioteki

Do obsługi urządzeń i komunikacji w projekcie konieczne będzie zaimportowanie odpowiednich bibliotek do Arduino IDE. Oto lista wymaganych bibliotek:

- Wire.h – do komunikacji I2C z czujnikiem BME280.
- DFRobot_BME280 – do obsługi czujnika BME280.
- SoftwareSerial.h – do komunikacji z modułem GPS za pomocą portu szeregowego.
- TinyGPS++ – do obsługi danych GPS.
- LoRa.h – do obsługi komunikacji LoRa (moduł SX1278).

2. Struktura programu

Oprogramowanie zostanie podzielone na kilka głównych sekcji:

- Inicjalizacja – wczytanie bibliotek, inicjalizacja czujników, LoRa oraz GPS.
- Odczyt danych – zbieranie pomiarów z czujników (BME280 i GPS).
- Komunikacja LoRa – przesyłanie zebranych danych przez moduł LoRa.
- Monitorowanie stanu systemu – monitorowanie działania urządzenia poprzez Serial Monitor w Arduino IDE (opcjonalnie).

3. Inicjalizacja systemu

Na początku programu należy zainicjować wszystkie używane moduły i czujniki. W tym celu tworzony jest obiekt dla czujnika BME280 oraz modułu GPS.

```
void setup() {
  // Inicjalizacja Serial
  Serial.begin(115200);
  ss.begin(9600);  // Ustawienie portu szeregowego dla GPS

  // Inicjalizacja LoRa
  Serial.println("LoRa Initializing...");
  LoRa.setPins(loraSS, loraRST, loraDIO0);  // Pinów dla LoRa
  if (!LoRa.begin(loraFrequency)) {  // Inicjalizacja LoRa na częstotliwości 433 MHz
    Serial.println("LoRa initialization failed!");
    while (1);  // Jeśli inicjalizacja LoRa się nie uda, program zatrzymuje się
  }
  Serial.println("LoRa initialization succeeded.");

  // Inicjalizacja BME280
  Serial.println("bme read data test");
  while (bme.begin() != BME::eStatusOK) {  // Próba inicjalizacji czujnika BME280
    Serial.println("bme begin failed");
    printLastOperateStatus(bme.lastOperateStatus);  // Wyświetlanie statusu błędu
    delay(2000);  // Czekaj 2 sekundy i próbuj ponownie
  }
  Serial.println("bme begin success");
}
```

2. Odczyt danych z czujników

W tej sekcji program odpowiada za zbieranie danych z czujników GPS i BME280. Odczytywana jest lokalizacja GPS (szerokość, długość, wysokość) oraz dane z czujnika BME280 (temperatura, wilgotność, ciśnienie, wysokość).

```
// Funkcja zbierająca i przetwarzająca dane GPS
void processGPSData() {
  while (ss.available() > 0) {  // Jeśli dane są dostępne w buforze GPS
    char c = ss.read();  // Odczytaj znak z GPS
    gps.encode(c);  // Przetwarzaj dane GPS

    // Jeśli dane GPS zostały zaktualizowane, zapisz je do bufora
    if (gps.location.isValid() && gps.location.isUpdated()) {
      lastGPSData = "Lat: " + String(gps.location.lat(), 6) +
                    ", Lng: " + String(gps.location.lng(), 6) +
                    ", Alt: " + String(gps.altitude.meters(), 2) +
                    ", Sats: " + String(gps.satellites.value());  // Zbieranie danych GPS
      Serial.println("Updated GPS Data: " + lastGPSData);  // Wyświetlanie danych GPS na serial monitorze
    }
  }
}
```

3. Wysyłanie danych przez LoRa

Funkcja wysyłania danych przez LoRa zbiera informacje z czujników BME280 i GPS, łącząc je w jeden ciąg znaków, który następnie jest przesyłany przez moduł LoRa. Po odczytaniu danych (takich jak temperatura, wilgotność, ciśnienie, wysokość, oraz współrzędne GPS) tworzymy z nich jeden komunikat, który łączy dane z obu czujników. Następnie używamy funkcji LoRa.beginPacket() do rozpoczęcia pakowania danych, a LoRa.print() do ich przesyłania. Po zakończeniu transmisji wywołujemy LoRa.endPacket(), co finalizuje wysyłanie danych. Cały proces kończy się wyświetleniem potwierdzenia na monitorze szeregowym, co pozwala na monitorowanie poprawności transmisji. LoRa zapewnia wysyłanie danych na dużą odległość z niskim zużyciem energii, co jest idealne w przypadku balonów stratosferycznych, które działają w odległych lokalizacjach i na dużych wysokościach.

```
// Funkcja do wysyłania danych przez LoRa
void sendData() {
  // Odczyt danych z BME280
  float temp = bme.getTemperature();  // Odczyt temperatury
  uint32_t press = bme.getPressure();  // Odczyt ciśnienia
  float alti = bme.calAltitude(seaLevelPressure, press);  // Obliczanie wysokości na podstawie ciśnienia
  float humi = bme.getHumidity();  // Odczyt wilgotności
  String sensorData = "Temp: " + String(temp) + " C, Hum: " + String(humi) +
                      "%, Press: " + String(press) + " Pa, Alt: " + String(alti) + " m";  // Łączenie danych z BME280

  // Połączenie danych z GPS i BME280
  String fullData = lastGPSData + " | " + sensorData;  // Łączenie danych GPS z danymi czujników

  // Wysyłanie danych przez LoRa
  LoRa.beginPacket();  // Rozpoczęcie pakietu LoRa
  LoRa.print(fullData);  // Wysyłanie danych
  LoRa.endPacket();  // Zakończenie pakietu LoRa

  // Wyświetlanie w konsoli
  Serial.println("Data sent via LoRa: " + fullData);  // Potwierdzenie wysłania danych
}
```

Algorytm

1. Inicjalizacja systemu:

    - Uruchomienie portu szeregowego do komunikacji.
    - Inicjalizacja komunikacji z czujnikiem GPS przez port szeregowy (SoftwareSerial).
    - Inicjalizacja modułu LoRa i sprawdzenie poprawności połączenia.
    - Sprawdzenie dostępności czujnika BME280 i jego poprawna konfiguracja.

2. Odczyt danych z czujników:

- Odczyt danych GPS:
    - Sprawdzanie dostępności nowych danych GPS.
    - Jeśli dane GPS są zaktualizowane, zapisujemy szerokość, długość, wysokość oraz liczbę satelitów.
- Odczyt danych z czujnika BME280:
    - Pobranie temperatury, wilgotności, ciśnienia oraz obliczenie wysokości na podstawie ciśnienia.

3. Łączenie danych:

- Po zebraniu danych z GPS i BME280, łączymy je w jeden ciąg znaków w formacie: Lat: <szerokość>, Lng: <długość>, Alt: <wysokość>, Temp: <temperatura>, Hum: <wilgotność>, Press: <ciśnienie>.

4. Wysyłanie danych przez LoRa:

- Rozpoczęcie pakowania danych do przesyłania przez LoRa za pomocą funkcji LoRa.beginPacket().
- Wysłanie danych przez funkcję LoRa.print().
- Zakończenie pakietu i jego wysyłka z pomocą LoRa.endPacket().

5. Monitorowanie czasu:

- Program monitoruje, czy minęła określona ilość czasu od ostatniego wysłania danych (np. 10 sekund).
- Jeśli tak, zbiera nowe dane i wysyła je ponownie.

Zakończenie:

- Po zakończeniu cyklu (np. wysyłania danych) system wraca do początkowego stanu i zaczyna odczytywać dane ponownie.

Schemat działania programu:

```
START
 |
 |-> Inicjalizacja systemu (porty szeregowe, LoRa, czujnik BME280, GPS)
 |
 |-> Odczyt danych z GPS
 |
 |-> Odczyt danych z BME280
 |
 |-> Połączenie danych GPS i BME280 w jeden ciąg
 |
 |-> Wysyłanie danych przez LoRa
 |
 |-> Czekanie na upłynięcie czasu (np. 10 sekund)
 |
 |-> Powrót do odczytu danych
 |
END
```

Listing programu z komentarzami

```
#include <Wire.h>
#include <SoftwareSerial.h>
#include <TinyGPS++.h>
#include "DFRobot_BME280.h"
#include <SPI.h>
#include <LoRa.h>

// Typ czujnika BME280 (I2C)
typedef DFRobot_BME280_IIC BME;

// Ustawienia GPS
SoftwareSerial ss(4, 3); // RX, TX dla GPS
TinyGPSPlus gps;

// Inicjalizacja BME280
BME bme(&Wire, 0x76); // Adres czujnika BME280
const float seaLevelPressure = 1015.0f; // Ciśnienie na poziomie morza

// Ustawienia LoRa
const int loraSS = 10;     // Pin CS
const int loraRST = 9;     // Pin RST
const int loraDIO0 = 2;    // Pin DIO0
const long loraFrequency = 433E6; // Częstotliwość (433 MHz)
```

```
// Timer do wysyłania danych
unsigned long lastSendTime = 0; // Czas ostatniego wysłania danych
const unsigned long sendInterval = 10000; // Przerwa między wysyłaniem danych (10 sekund)
// Bufor na ostatnie poprawne dane GPS
String lastGPSData = "No GPS data available";
// Funkcja wyświetlająca status BME280
void printLastOperateStatus(BME::eStatus_t eStatus) {
  switch (eStatus) {
  case BME::eStatusOK:
    Serial.println("everything ok");
    break;
  case BME::eStatusErr:
    Serial.println("unknown error");
    break;
  case BME::eStatusErrDeviceNotDetected:
    Serial.println("device not detected");
    break;
  case BME::eStatusErrParameter:
    Serial.println("parameter error");
    break;
  default:
    Serial.println("unknown status");
    break;
  }
}
```

```
// Funkcja zbierająca i przetwarzająca dane GPS
void processGPSData() {
  while (ss.available() > 0) {
    char c = ss.read(); // Odczytaj znak z GPS
    gps.encode(c);     // Przetwarzaj dane GPS

    // Jeśli dane GPS zostały zaktualizowane, zapisz je do bufora
    if (gps.location.isValid() && gps.location.isUpdated()) {
      lastGPSData = "Lat: " + String(gps.location.lat(), 6) +
                    ", Lng: " + String(gps.location.lng(), 6) +
                    ", Alt: " + String(gps.altitude.meters(), 2) +
                    ", Sats: " + String(gps.satellites.value());
      Serial.println("Updated GPS Data: " + lastGPSData);
    }
  }
}
// Funkcja do wysyłania danych przez LoRa
void sendData() {
  // Odczyt danych z BME280
  float temp = bme.getTemperature();
  uint32_t press = bme.getPressure();
  float alti = bme.calAltitude(seaLevelPressure, press);
  float humi = bme.getHumidity();
  String sensorData = "Temp: " + String(temp) + " C, Hum: " + String(humi) +
                      "%, Press: " + String(press) + " Pa, Alt: " + String(alti) + " m";
```

```
// Połączenie danych z GPS i BME280
  String fullData = lastGPSData + " | " + sensorData;
  // Wysyłanie danych przez LoRa
  LoRa.beginPacket();
  LoRa.print(fullData);
  LoRa.endPacket();
  // Wyświetlanie w konsoli
  Serial.println("Data sent via LoRa: " + fullData);
}
void setup() {
  // Inicjalizacja Serial
  Serial.begin(115200);
  ss.begin(9600);
  // Inicjalizacja LoRa
  Serial.println("LoRa Initializing...");
  LoRa.setPins(loraSS, loraRST, loraDIO0);
  if (!LoRa.begin(loraFrequency)) {
    Serial.println("LoRa initialization failed!");
    while (1);
  }
  Serial.println("LoRa initialization succeeded.");
```

```
// Inicjalizacja BME280
  Serial.println("bme read data test");
  while (bme.begin() != BME::eStatusOK) {
    Serial.println("bme begin failed");
    printLastOperateStatus(bme.lastOperateStatus);
    delay(2000);
  }
  Serial.println("bme begin success");
}
void loop() {
  // Przetwarzanie danych GPS
  processGPSData();
// Sprawdź, czy upłynęło 10 sekund
  unsigned long currentTime = millis();
  if (currentTime - lastSendTime >= sendInterval) {
    lastSendTime = currentTime; // Zaktualizuj czas ostatniego wysłania danych
    sendData(); // Wyślij dane
  }
}
```

Uwagi i spostrzeżenia z uruchamiania projektu

Podczas uruchamiania urządzenia napotkano kilka istotnych kwestii, które należy uwzględnić przy implementacji i testowaniu systemu. Na początku wystąpiły problemy z inicjalizacją czujnika BME280, które można było rozwiązać poprzez dodanie pętli oczekującej na poprawne uruchomienie oraz komunikatów diagnostycznych. W przypadku modułu GPS, czas oczekiwania na pierwszy sygnał może być długi, szczególnie w zamkniętych przestrzeniach, co należy uwzględnić w programie, informując użytkownika o konieczności oczekiwania na stabilny sygnał. Podczas testów komunikacji LoRa zauważono zależność zasięgu od przeszkód terenowych, takich jak budynki, co wymaga przeprowadzenia testów terenowych i dobrania odpowiedniej anteny. Problemy z zarządzaniem zasilaniem, zwłaszcza przy dużym poborze prądu przez moduły GPS i LoRa, skłoniły do rozważenia zastosowania akumulatorów o większej pojemności oraz trybów oszczędzania energii. Z kolei błędy w konfiguracji pinów urządzeń, szczególnie przy komunikacji I2C, wymagały dokładnego przypisania pinów, aby uniknąć konfliktów. W celu usprawnienia działania systemu zastosowano również regularne monitorowanie danych wyjściowych z czujników i wprowadzenie komunikatów diagnostycznych, co pozwoliło na szybkie wykrywanie problemów. Testy komunikacji LoRa potwierdziły, że zasięg oraz jakość sygnału mogą się różnić w zależności od terenu, a optymalizacja kodu oraz zarządzanie pamięcią i energią były kluczowe dla stabilności działania urządzenia w długotrwałych pomiarach terenowych.

Bibliografia

1. **Arduino Documentation**
   Arduino Software. "Arduino IDE Documentation." Arduino.cc.
   Dostęp: https://www.arduino.cc/reference/en/

2. **LoRa Module**
   Semtech Corporation. "LoRa Technology." Semtech.com. Dostęp:
   https://www.semtech.com/lora

3. **TinyGPS++ Library**
   Mikal Hart. "TinyGPS++ Library Documentation." GitHub. Dostęp:
   https://github.com/mikalhart/TinyGPSPlus

4. **BME280 Sensor**
   DFRobot. "BME280 Environmental Sensor." DFRobot.com.
   Dostęp: LINK

5. **BME280 Sensor Datasheet**
   Bosch Sensortec. "BME280 Datasheet." Bosch.com.
   Dostęp:
   https://dfimg.dfrobot.com/nobody/wiki/08b27f9827b4182a692b7069958dc81f.pdf

6. **LoRaWAN Protocol**
   The LoRa Alliance. "LoRaWAN Specification." LoRa Alliance.
   Dostęp: https://lora-alliance.org/

7. **GPS Module**
   u-blox. "NEO-6M GPS Module Datasheet." u-blox.com.
   Dostęp: https://content.u-blox.com/sites/default/files/products/documents/NEO-6_DataSheet_%28GPS.G6-HW-09005%29.pdf

8. **Arduino Forum**
   Arduino Community. "Troubleshooting LoRa, GPS and BME280." Arduino Forum.
   Dostęp: https://forum.arduino.cc/

9. **LoRaWAN Network Design**
   Semtech. "Designing LoRaWAN Networks for the Internet of Things." Semtech.
   Dostęp: https://www.semtech.com/lora