Autorzy projektu- Szymon Łysoń i Jakub Małek

Tytuł projektu- MAD- Miniaturowy, autonomiczny dron

Skrócony opis projektu

Obecnie dostępne autonomiczne drony są kosztowne i masywne. Postawiliśmy sobie za cel wyeliminowanie powyższych problemów, tworząc pierwszy na świecie miniaturowy, samolądujący, w pełni autonomiczny dron. Naszym głównym celem jest stworzenie niewielkiego, wielofunkcyjnego urządzenia, którego głównym celem będzie pokonać trasę między dwoma punktami, autonomicznie omijając napotkane przeszkody, oddalonymi do paru kilometrów, a następnie wylądować w najbardziej optymalnym miejscu.

Inspiracja

Główną ispiracją dla naszego projektu były niedawne przełomowe próby skonstuowania dronów autonomicznych. Na przestrzeni ostatniego roku pojawiło się wiele nowych rozwiązać w dziedzinie dronów autonomicznych. Między innymi zostały spopularyzowane różne techniki skanowania 3D, które otworzyło takim dronom masę nowych możliwości.

Napotkane problemy

Głównym problemem było sprawienie, aby nasz komputer pokładowy, Compute Module 4, posiadał wszystkie funkcje, jakich od niego wymagaliśmy. Sporą trudnością było Wi-Fi, które przez długi czas, aż do teraz nie udało się w pełni opanować. Brak Wi-Fi sprawia, że nie możemy testować lotów autonomicznych.

Definicja problemu

Zakres problemu

Głównym problemem, jaki chcieliśmy rozwiązać, było stworzenie w pełni autonomicznego drona, który nie wymagałby interakcji ze strony człowieka od startu do lądowania. Jest to największa trudność, jaką muszą stawić czoła dzisiejsze drony autonomiczne.

Przegląd techniczny

Istnieje obecnie mnóstwo dronów dostępnych do sprzedaży detalicznej, m.in. marki DJI, lub Hubsan. Potrafią one latać ponad 10 minut, a także posiadają funkcje takie jak śledzenie celu lub utrzymywanie stabilnej pozycji. Jednak ich oprogramowanie nie daje żadnej możliwości w pełni autonomicznego lotu w dynamicznym otoczeniu.

Drony autonomiczne dostępne obecnie na rynku nie są używane komercyjnie na dużą skalę. Na ogół nie są przeznaczone do lotów na otwartej przestrzeni ze względu na warunki pogodowe, m.in. wiatr. Z tego powodu obecne rozwiązania nadają się do działania jedynie we wnętrzu budynków oraz do stosowania w bezwietrznych warunkach pogodowych.

Z tych względów istniejące rozwiązania nie znajdują praktycznych zastosowań w nauce, biznesie etc. Drony potrafiące latać jedynie w pomieszczeniach mogą być wykorzystywane do niewielu innych celów poza skanowaniem wnętrz. Pojazdy, które są zdolne do lotu na otwartej przestrzeni nie są najczęściej przystosowane do lotu w silnym wietrze, co uniemożliwia ich stosowa

Dużym problemem autonomicznych dronów jest ich wysoka cena, zwykle zaczynająca się w okolicy $1000. Jednak jest często jest ona sztucznie zawyżona, nie adekwatna do parametrów użytych części, a przy tym uniemożliwia ona popularyzację tego typu systemów.

Drony, które są w stanie latać autonomicznie w niesprzyjających warunkach pogodowych nie są jeszcze możliwe do kupienia dla przeciętnego człowieka. Jest niewiele firm, na przykład Percepto lub Skydio które zajmują się produkcją takich statków, ale ich produkty są drogie a do tego często dostępne tylko dla dużych firm bądź wojska.

Sporą przeszkodą dla rozwoju dronów autonomicznych jest fakt, iż tego typu pojazdy muszą nieustannie monitorować przestrzeń wokół siebie. Bardzo popularnym rozwiązaniem są skanery Lidar [1]. Są one jednak ciężkie, drogie i zajmują wiele miejsca, a dodatkowo ich efektywność zależy od wielu czynników, chociażby kolor skanowanego obiektu.

Kryteria rozwiązania

Abyśmy uznali nasz projekt za zakończony sukcesem musi on spełnić wiele kryteriów. Najważniejszą cechą naszego rozwiązania musi być zdolność drona do lotu w pełni autonomicznego, lądowanie na znalezionym miejscu do lądowania. Podczas lotu powinien być w stanie skanować przestrzeń przed sobą w celu uniknięcia kolizji a także robienie mapy 3D. Powinien mieć możliwość względnie ciągłej komunikacji na odległość co najmniej dwóch kilometrów. W celu uniknięcia ryzyka utracenia pojazdu powinien samodzielnie badać swój stan, a w przypadku wykrycia np. ryzyka rozładowania, samodzielnie wylądować i poinformować użytkownika o swoje lokacji.

Opis Projektu

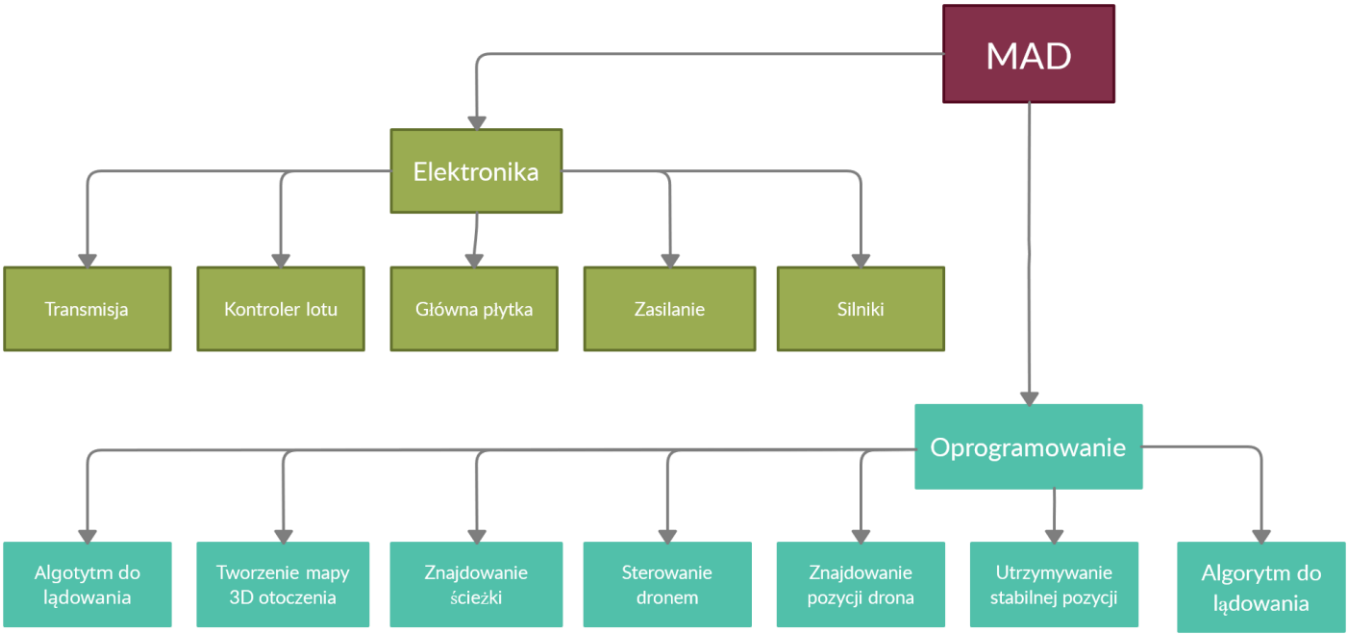
Przegląd

Naszą odpowiedzią na wcześniej przedstawione problemy jest MAD– Miniaturowany autonomiczny dron. Będzie on mógł samodzielnie wystartować, dolecieć wybranego punktu w promieniu paru kilometrów od miejsca startu, omijając napotkane po drodze przeszkody, a następnie znaleźć płaskie miejsce do lądowania i na nim samodzielnie wylądować. Całość bez konieczności ingerencji ze strony człowieka. Wystarczy jedynie zaznaczyć docelowy punkt końcowy w intuicyjnej aplikacji, a dron sam tam doleci i wyląduje.

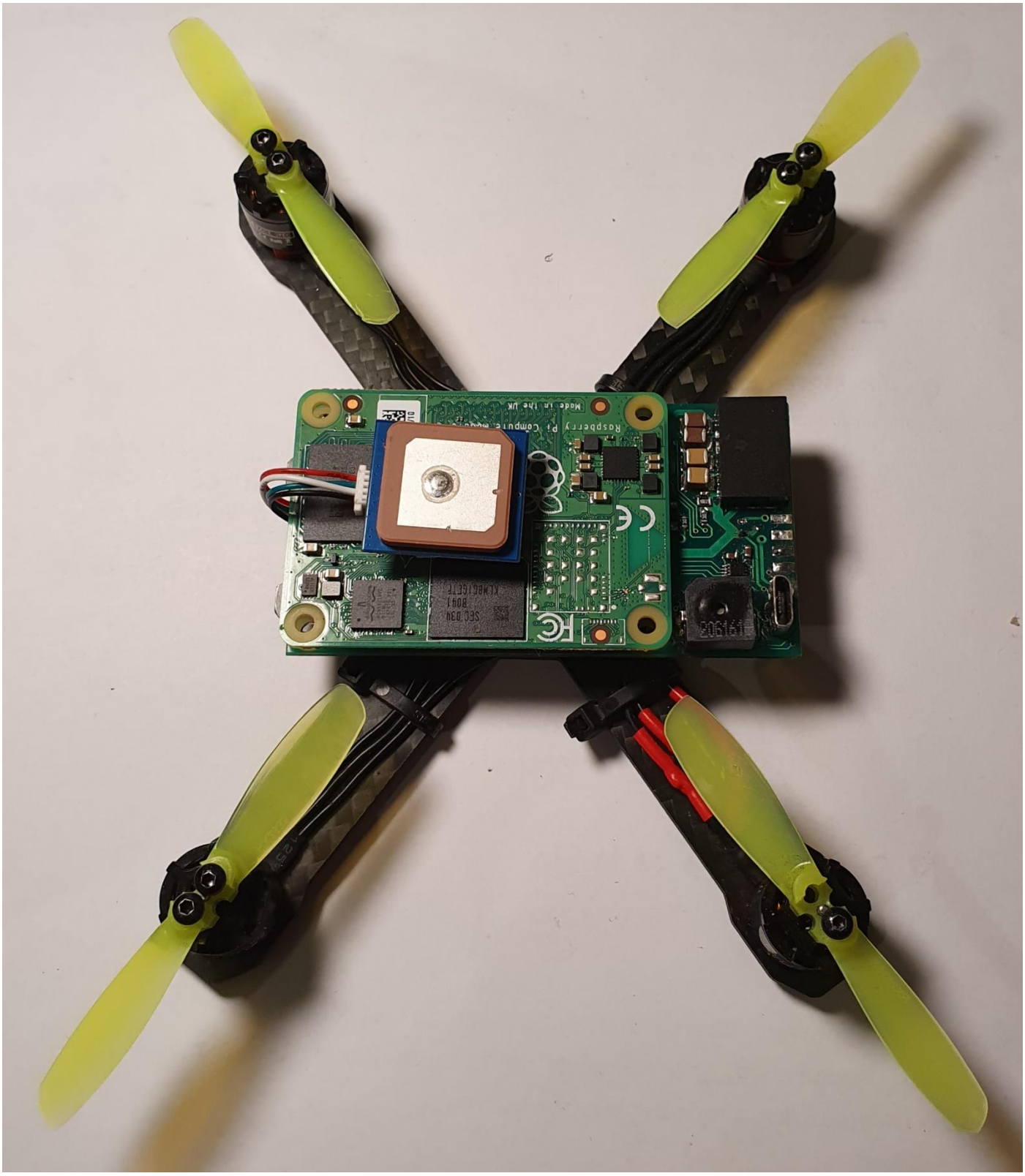
W celu zminimalizowania urządzenia stworzyliśmy własną płytkę drukowaną, zawierającą prawie całą wykorzystywaną przez nas elektronikę. Skanowanie otoczenia odbywa się za pomocą kamer stereo. Obraz z nich jest przetwarzany do mapy 3D, na podstawie której wyznaczana jest jego dalsza ścieżka. Zastosowaliśmy też dokładny system pozycjonowania 3D, dzięki czemu dron zna swoją pozycję w czasie rzeczywistym z dokładnością co do kilkunastu centymetrów.

Szczegółowy opis

Na poniższym diagramie znajduje się opis głównych składowych projektu, które to poniżej szczegółowo opisaliśmy.



Aktualnie zbudowany prototyp waży niecałe 300 gram. Dla zmniejszenia wagi i ceny użyliśmy gotowej, kupnej konstrukcji mechanicznej, którą jedynie dostosowaliśmy do naszych potrzeb. Poniżej znajduje się zdjęcie prototypu. Aktualna wersja jest chwilowo pozbawiona kamer i baterii, ze względu używanie ich do osobnych testów. Nie zawiera on też planowanej anteny, ze względu na opóźnienia producenta używanego przez nas komputera w certyfikacji modułów wi-fi. Jest też to główny powód jego niezdolności do lotu, który to jednak przetestowaliśmy z sukcesem na wcześniejszych prototypach opartych o Raspberry pi zero w.



Oprogramowanie

Technologia

Budując autonomicznego drona nie można być do końca pewnym jakie problemy będzie trzeba rozwiązać. Z tego powodu do stworzenia oprogramowania odpowiadającego za autonomiczny lot pojazdu najlepiej jest wybrać język programowania, który posiada dużą liczbę bibliotek rozwiązujących przynajmniej część problemów, jakie mogą pojawić się w przyszłości. Istotnym czynnikiem jest również prędkość języka oraz ilość pamięci, jakiej potrzebuje.

Wybranie języka, który pozwoliłby na jak najlepsze, a przy tym szybkie i łatwe pisanie kodu jest bardzo trudne. Najlepszym wyborem mógłby wydawać się Python, ze względu na możliwość bardzo szybkiego i prostego pisania kodu, a przy tym posiadający duże wsparcie bibliotek do m.in. tworzenia sieci neuronowych, uczenia maszynowego, obliczeń naukowych etc. Python jest niestety bardzo wolny, ponieważ jest językiem interpretowanym. Niektóre algorytmy, które wymagają dużej ilości obliczeń nie byłyby w stanie działać w Pythonie.

W tej sytuacji najlepszym rozwiązaniem byłoby połączenie zalet kilku języków. Python, który posiada wiele bibliotek zewnętrznych oraz automatyczne zarządzanie pamięcią, mógłby być głównym językiem, o jaki byłoby oparte oprogramowanie drona. Algorytmy, które wymagają największej ilości obliczeń mogą być napisane w C++ i dzięki temu będą oszczędzać czas. Takie podejście sprawia, że nasze oprogramowanie jest zarówno szybkie, jak i proste do zmodyfikowania i rozwijania.

Symulacja drona

W celu testowania oprogramowania odpowiedzialnego za lot autonomiczny należało stworzyć środowisko, w którym dałoby się obserwować działanie całości, oraz które umożliwiałoby symulowanie niektórych interakcji ze światem, na przykład skanowania 3D. W tym celu zaprogramowaliśmy prostą symulację fizyki, uwzględniającą siły generowane przez silnik, siłę grawitacji oraz opór powietrza, która komunikuje się z programem Unity, zapewniającym wizualizację poruszania się drona wraz z symulacją skanowania 3D.

Algorytm do lądowania

W momencie, w którym oprogramowanie podejmie decyzję o lądowaniu, dron staje w miejscu, a obydwie kamery obracają się w dół, umożliwiając zeskanowanie terenu pod dronem. Uzyskany skan zostaje podzielony na kwadraty o wymiarach 0.3m na 0.3m w osiach x i z, gdzie oś y oznacza kierunek pionowy. Dla każdego z tych kwadratów są następnie, używając regresji liniowej, obliczane średnie kąty nachylenia tych obszarów. Biorąc pod uwagę te wartości oprogramowanie wybiera najlepszy kwadrat do lądowania lub, jeśli nie znajdzie żadnego wystarczająco dobrego, leci w inne miejsce, aby rozpocząć poszukiwania od nowa.

Kiedy zostanie znaleziony optymalny kawałek terenu do lądowania, dron leci nad ten obszar, a następnie zaczyna zniżać wysokość, jednocześnie utrzymując ciągle tę samą pozycję w osiach x oraz z, używając algorytmu opisanego dalej. Gdy dron znajdzie się na tyle nisko nad ziemią, że czujnik odległości będzie w stanie wiarygodnie liczyć wysokość, dron przestanie używać skanowania za pomocą kamer, a do końca procesu zniżania będzie polegał na czujniku.

Testy określiły maksymalną odległość wiarygodnego pomiaru odległości przez czujnik na 2m, a minimalną na 5cm. W tym zakresie jego pomiary są przyjmowane za prawdziwe, niezależnie od wskazań innych czujników.

Działa on z częstotliwością 10Hz. W celu zwiększenia płynności pomiarów, umożliwiającą brak synchronizacji czujnika z innymi częściami algorytmu do lądowania, w przerwach między pomiarami czujnika odległość jest liczona używając niżej opisanego algorytmu do śledzenia drona.

Algorytm do skanowania

Skanowanie 3D odbywa się za pomocą tzw. widzenia stereoskopowego. Polega ono na wykonywaniu jednocześnie zdjęć z dwóch kamer oddalonych o niewielką odległość, a następnie porównywanie zdjęć, zrobionych przez nie, w celu określenia jak bardzo dane obszary są przesunięte. Znając przesunięcie poszczególnych pikseli, odległość między kamerami oraz ogniskową obiektywów w użytych kamerach można policzyć odległości punktów na obrazach od kamery. Istnieje wiele rozwiązań tego problemu. My zdecydowaliśmy się użyć biblioteki OpenCV ze względu na dobrą dostępność pomocniczych materiałów.

Dla zwiększenia skanowanej powierzchni wybraliśmy kamery szerokokątne OV5647 160°

Algorytm do znajdowania ścieżki na mapie 3D

Aby dron mógł być w pełni autonomiczny musi posiadać dobry algorytm znajdywania ścieżki na mapie 3D. Istnieje wiele algorytmów do wyznaczania ścieżki, ale nasz musiał spełniać kilka trudnych wymagań, przede wszystkim musiał być w stanie działać na naszym mocno ograniczonym sprzęcie, a także umieć wyznaczyć trasę do obiektu oddalonego o kilometry.

Najbardziej optymalny algorytm do tego celu jest algorytm A* [2]. Posiada wiele przewag nad m.in. Algorytmem Dijkstry [3], z czego najbardziej istotną przewagą jest prędkość działania oraz ilość zużywanej pamięci. Jednak nawet algorytm A* ma zbyt dużą złożoność pamięciową, aby efektywnie znajdować trasę lotu na dystansie kilometrów. Z tego powodu MAD używa zmodyfikowanego algorytmu A* używającego tablicy haszującej do przechowywania punktów, zamiast klasycznej tablicy trójwymiarowej. Ścieżka otrzymana z algorytmu A* jest przedstawiona w postaci listy punktów w przestrzeni 3D.

Algorytm do sterowania dronem

Aby dron był w stanie latać w pełni autonomicznie, musi być w stanie pokonywać wcześniej wyznaczoną trasę (patrz poprzedni punkt) nie uderzając w przeszkody. Aby osiągnąć taki wynik musi posiadać oprogramowanie umożliwiające mu planowanie rotacji R oraz mocy silników S w danym czasie. Znając wektor sił, którymi dron działa za pomocą silników, można z łatwością obliczyć R oraz S, używając prostej trygonometrii.

Znając ścieżkę, po której ma poruszać się dron łatwo jest zauważyć, że na dowolnych trzech punktach tworzących ją można opisać okrąg. Dron, chcąc poruszać się po tych punktach będzie poruszał się po torze zbliżonym do tego okręgu. Znając maksymalne przyspieszenie, jakie jest w stanie osiągnąć dron, wyznaczone eksperymentalnie oraz promień okręgu, po jakim się porusza, można w łatwy sposób obliczyć maksymalną

prędkość, jaką może mieć w danym punkcie, aby nie wypaść z pożądanego toru lotu. Pierwsza część algorytmu polega na obliczeniu maksymalnej prędkości, z jaką może poruszać się dron w danym punkcie ścieżki.

Następnie dla każdego punktu z wyjątkiem ostatniego jest obliczany wektor jednostkowy prędkości w danym punkcie. Jest on równy wektorowi zmiany pozycji względem następnego punktu podzielonego przez jego długość. Wektor prędkości w danym punkcie będzie równy wektorowi jednostkowemu prędkości pomnożonemu przez maksymalną prędkość w danym punkcie.

Znając zmianę prędkości oraz czas, w jakim dron pokona trasę z dwóch sąsiadujących punktów obliczane jest przyspieszenie w danym punkcie ścieżki. Wektor siły, jaką musi działać dron w danym punkcie jest równy sumie masy pomnożonej przez wcześniej uzyskany wektor przyspieszenia, wektora sił, jaki jest potrzebny do przeciwdziałania oporom powietrza w danym momencie oraz wektorowi siły potrzebnej do przeciwdziałania sile grawitacji.

Dokładność powyższego algorytmu byłaby bardzo mała, jeśli uwzględniałby punkty oddalone np. o metr. Jednak jego dokładność można bardzo mocno zwiększyć dzieląc ścieżkę na wiele punktów znajdujących się bliżej siebie. Działa to na podobnej zasadzie jak całkowanie numeryczne; im mniejszy jest podprzedział co jaki następują obliczenia, tym wynik jest dokładniejszy.

Algorytm do utrzymywania stabilnej pozycji

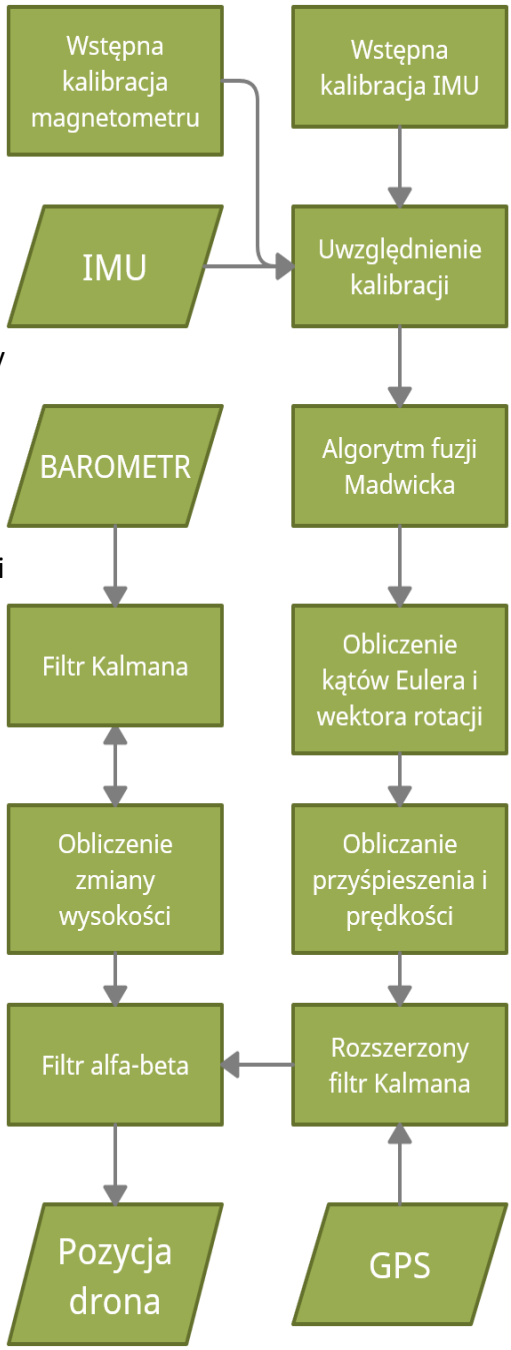
W celu wykonywania niektórych zadań dron musi być w stanie utrzymywać przez dłuższy czas stabilną pozycję oraz przeciwdziałać sile wiatru. Aby osiągnąć takie rezultaty dron musiał posiadać algorytm, który nieustannie badałby zmianę pozycji w stosunku do pożądanej pozycji oraz dostosowywał moc silników, aby wyrównać ten błąd. Istnieje wiele algorytmów, które pomogłyby otrzymać takie rezultaty, ale my zdecydowaliśmy się użyć algorytmu PID. Posiada wiele zalet, m.in. jest łatwy w konfiguracji oraz możliwy do zaimplementowania w praktycznie dowolnej sytuacji.

Algorytm do śledzenia pozycji drona

Śledzenie pozycji drona jest jednym z kluczowych elementów naszego projektu, jako że od jego dokładności zależy jakość mapy 3D. Stworzyliśmy więc autorski algorytm, który ma zadowalające rezultaty zarówno w krótkim, jak i długim okresie czasu. Uruchamiamy go w 100Hz. Po prawej znajduje się jego opis blokowy.

- IMU- Pobieranie danych z dwóch akcelerometrów i żyroskopów BMI088. Wybraliśmy je ze względu na ich wysoką precyzje, nawet pod wpływem dużych drgań i wahań temperatury.
- Wstępna kalibracja IMU- Ten etap jest niezbędny dla uzyskania odchylenia (ang. bias) wyników. Polega on na położeniu drona bez ruchu, w możliwie poziomej pozycji na 10s, podczas których są zbierane dane z IMU. Następnie algorytmu liczy ich odchylenie od spodziewanych wyników.

- Wstępna kalibracja magnetometru- W celi kalibracji magnetometru przed startem algorytm dopasowuje elipsę utworzoną z pomiarów magnetometru ustawionym pod wieloma różnymi znanymi pozycjami, znajdując się w przybliżeniu w tym samym miejscu. W celu policzenia kierunku magnetometru podczas obracania urządzeniem używamy akcelerometru, dopasowując wektor pionowo skierowanej siły przeciwdziałającej grawitacji do jego wskazań, otrzymując w ten sposób wektor rotacji. Do dopasowania elipsoidy używamy gotowego algorytmu [13]. Z którego otrzymuję jej parametry wykorzystywane do poprawienia wyników. Szczególnie istotne jest przemieszczenie jej środka, które oznaczę przez C, oraz jej spłaszczenie względem sfery, które oznaczę przez E. Są to tak zwane błędy twarde i miętki wynikające kolejno z obecności magnetycznych materiałów w dronie i otoczeniu. Powyższy proces opera się o [14]



- Uwzględnienie kalibracji- Ten etap pozwala na zmniejszenie błędów urządzeń za pomocą wcześniej otrzymanych stałych. Wpierw, jeszcze w czipach, dane trafiają do filtrów dolnoprzepustowych. Od danych z IMU odejmowane są wcześniej policzone początkowe odchylenia wyników. W celu uzyskania poprawnych wyników z magnetometru, oznaczonych tu przez B, algorytm używa wzoru

  B = (B_0 – C)*E

- Algorytm fuzji Madgwicka- Ten algorytm [6] stał się standardem, jeśli chodzi o znajdowanie pozycji za pomocą wewnętrznych czujników. Pozwala on na najdokładniejsze policzenie kwaternionu służącego do jednoznacznego opisu orientacji urządzenia za pomocą wcześniej wspomnianych czujników.
- Obliczenie kątów Eulera i wektora rotacji- Niech wcześniej wspomniany kwaternion przybierze postać Początkowo algorytm liczy kąty eulera zgodnie z powszechnie znanym wzrorem wzorem z [7] Następnie uwzględnia deklinację magnetyczną dodając ją do kątów, po czym z otrzymanych kątów liczy wektor rotacji **R** według wzoru z [8]. Obliczanie przyśpieszenia i prędkości- Algorytm liczy przyśpieszenie na podstawie wzoru A_NED * R = A_XYZ gdzie A_NED to przyśpieszenie w układnie współrzędnych North-East-Down, a A_XYZ to skalibrowane dane z akcelerometru. Następnie liczy prędkość zgodnie z powszechnie znanym wzorem V = A_NED*dt, gdzie dt to okres między dwoma kolejnymi obliczeniami prędkości.
- Rozszerzeny Filtr Kalmana- Ten filtr, działający z częstotliwością 10Hz, pozwala na zmniejszenie odchylenia w czasie liczonej pozycji od prawdziwej uwzględniając pomiary GPSu. Ze względu na różnicę częstotliwości jego działania od reszty algorytmu wynikającą z maksymalnej częstotliwości działania GPSu, przez większość czasu jest on pomijany, licząc przemieszczenie za pomocą wzoru [POPRAWIĆ RÓWNANIE W WORDZIE DELTA pos= 0.5A_NED*dt^2]. Napisaliśmy go w oparciu o [10]. Wykonuje on niezależne obliczenia dla każdego z trzech wymiarów.

- Filtr Kalman [11]a- Służy zmniejszeniu błędu barometru i wraz z wbudowanym w barometr filtrem dolnoprzepustowym praktycznie eliminuje błąd pomiaru wynikający z działania urządzenia. W przyszłości planujemy uwzględnić wpływ prędkości i wiatru na jego pomiary.
- Liczenie zmiany wysokości- Barometr liczy swoją wysokość nad punktem startu ze wzoru

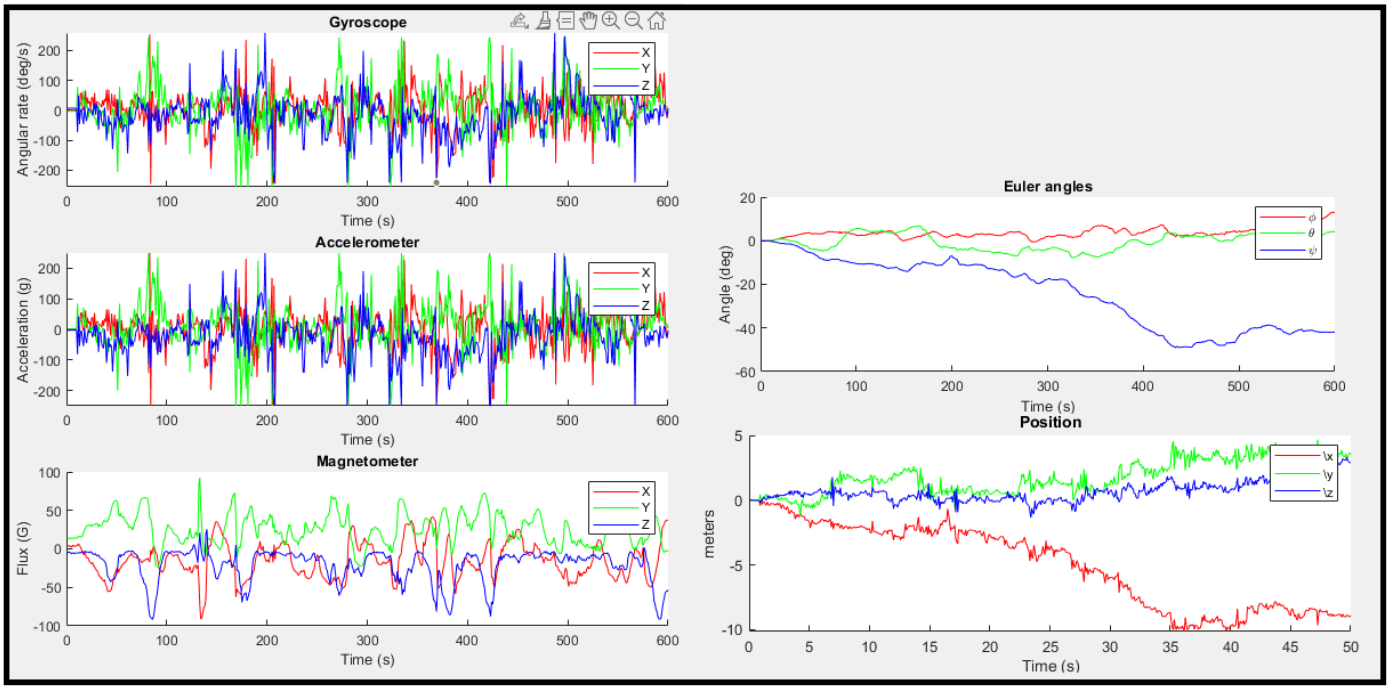
  y = 44300*(1-(pressure/pressure_0)^0.190295

- Filtr alfa-beta[12]: Pozwala on na dodatkowe uwzględnienie pomiarów barometru w pozycji drona, zmniejszając przez to błąd pomiaru w dłuższym okresie czasu. Przybiera on postać

  y = a * Pos_Y + (1 - a) * y

  Gdzie a to to tak zwane complementary gain, którego wartość dostosowaliśmy na podstawie szacunków błędów pomiarowych.

Jak dotąd przeprowadziliśmy podstawowe testy algorytmu, jeszcze bez uwzględniania GPSu i barometru. Odłączyliśmy je by móc przetestować wpierw każdą składową projektu oddzielnie. Polegały one na przejściu z urządzeniem kilku metrów jednocześnie nim zbaczając i obracając, a następnie zmierzeniu rzeczywistej końcowej pozycji i kąta. Po drobnych modyfikacjach udało nam się osiągnąć błąd końcowy na poziomie około 20%, co uwzględniając brak wyżej wymienionych urządzeń jest satysfakcjonującym rezultatem.

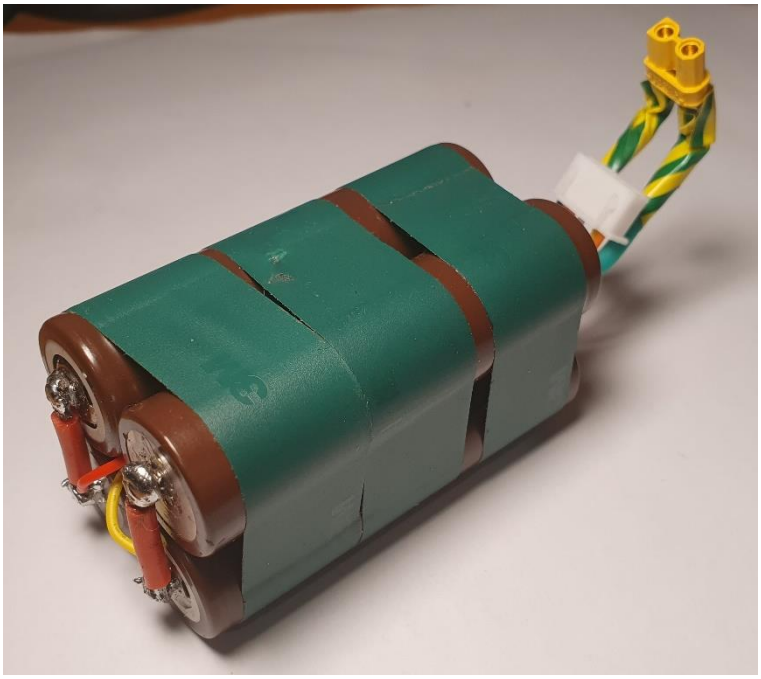


Maksymalny czas na wszystkich powyższych wykresach powinien wynosić 50 sekund, a nie 600.

Elektronika

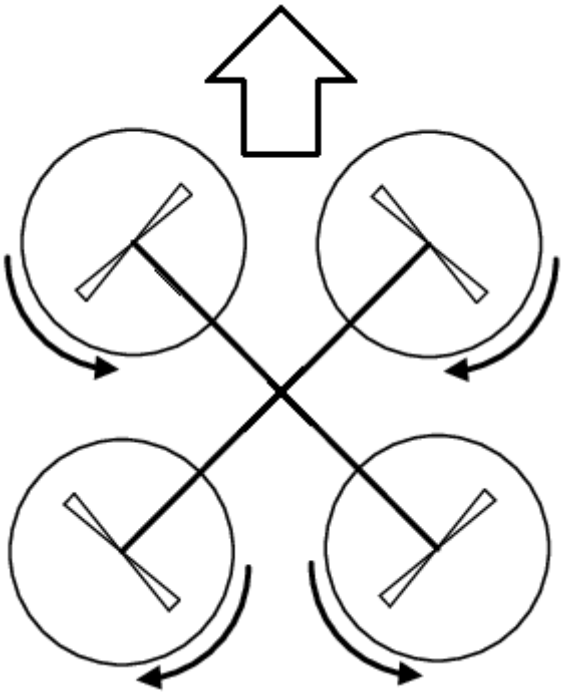
Jako główny komputer zdecydowaliśmy się na użycie najnowszego Raspberry Pi Compute Module 4, tworząc tym samym pierwszy znany nam tak rozbudowany używający go projekt. Nie nadaje się on jednak na kontroler lotu, ze względu na działanie systemu operacyjnego mogącego zakłócać często przeprowadzane kluczowe obliczenia dla stabilizacji i kontroli drona.

Zasilanie



Zdecydowaliśmy się na zasilanie drona 4 połączonymi szeregowo bateriami Li-Ion 3Ah 20A. Dzięki temu nie tylko ponad dwukrotnie zwiększył się zasięg lotu w porównaniu do częściej spotykanych w dronach baterii Li-Po o podobnej wadze, ale też poprawiło się bezpieczeństwo konstrukcji, umożliwiając jego wykorzystywanie w łatwopalnym otoczeniu, takim jak suchy las. Poniżej znajduje się zdjęcie używanej przez nas paczki baterii. Ważą one aż 2/3 całej wagi drona.

Wstępne testy pokazały, że dron może latać około 20 minut, co zgadza się naszymi z przewidywaniami.



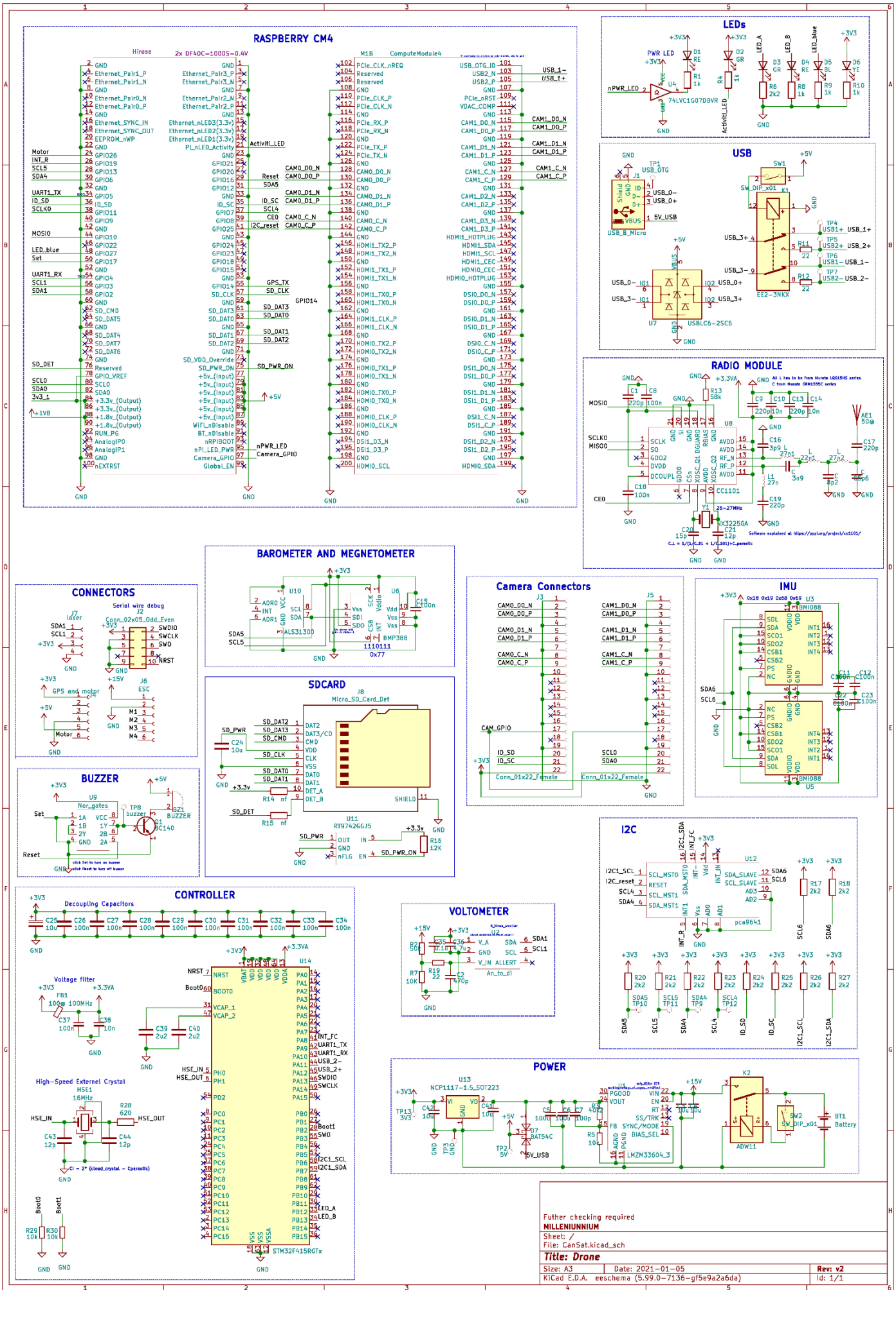
Silniki i śmigła

Zdecydowaliśmy się na użycie silników EMAX RS1106 II 4500KV ze względu na ich niewielką wagę oraz dobre osiągi. W celu zminimalizowania konstrukcji używamy śmigieł Gemfan RotorX 2535 o średnicy 6,3cm. Zdecydowaliśmy się na jedną płytkę kontrolującą szybkość wszystkich czterech silników. Aktualnie używamy modelu 15A firmy Hakrc, głównie ze względu na niską cenę i niewielkie wymiary. S

Silniki obracają się w konfiguracji X przedstawionej na rysunku obok.

Główna płytka

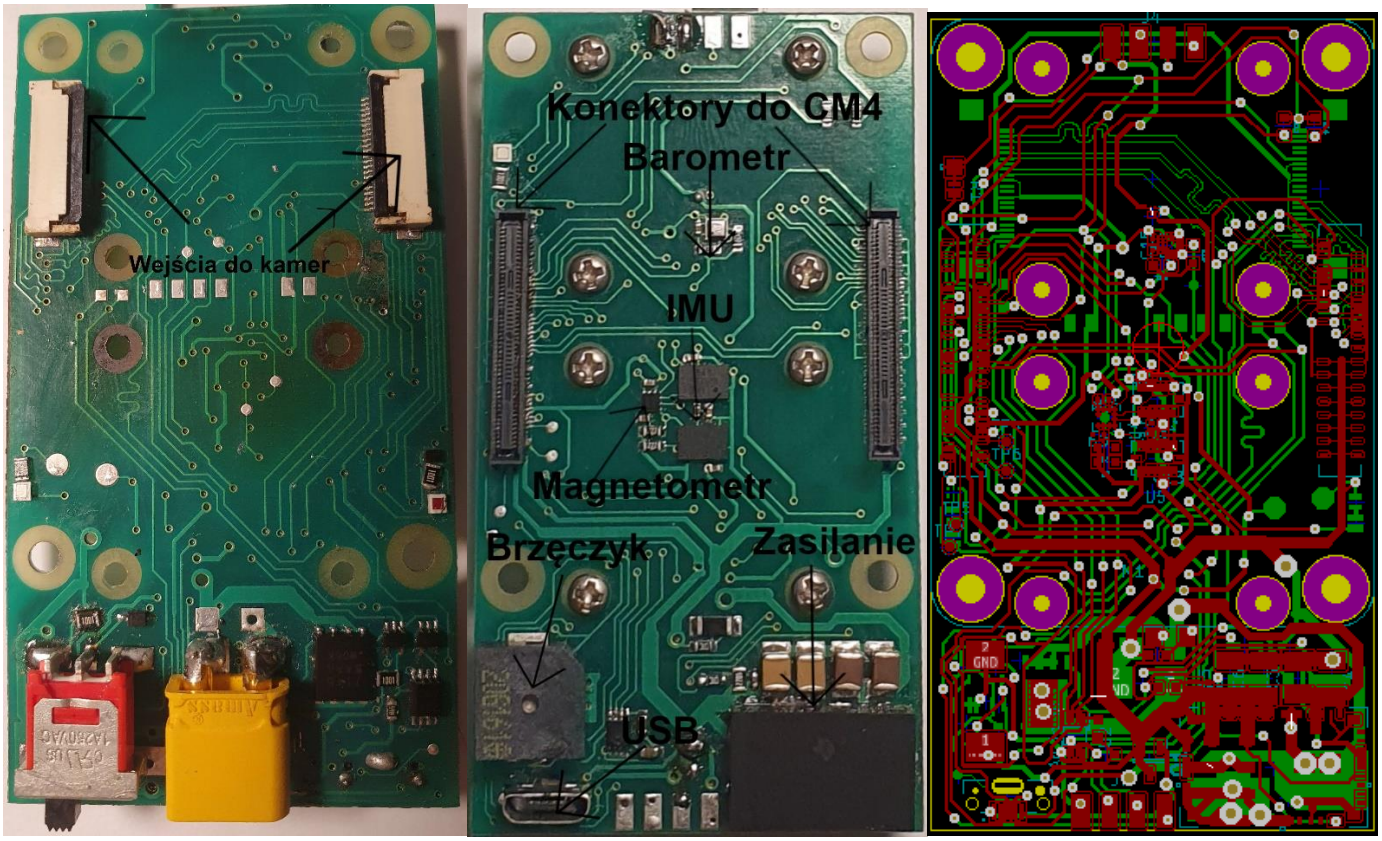
W celu zmniejszenia wymiarów i masy urządzenia, zdecydowaliśmy się na zbudowanie własnej płytki drukowanej, zawierającej większość potrzebnych elementów. Aktualnie zlutowaliśmy podstawową płytkę w celu przetestowania części funkcji, jednakże zrobiliśmy trudny do ręcznego naprawienia błąd przy projekcie uniemożliwiający pomiar baterii i wykorzystywanie brzęczyka w zamierzony przez nas sposób. Wykryliśmy je, naprawiliśmy i zrobiliśmy nowy schemat elektroniczny dodając do niej szereg nowych funkcji. W przeciągu najbliższego tygodnia skończymy jej projekt i ją zamówimy, tak by zbudować ją do końca stycznia. Jej schemat znajduje się poniżej. Pliki w wysokiej jakości znajdują się w naszym repozytorium na githubie.


RASPBERRY CM4
LEDs
USB
RADIO MODULE
BAROMETER AND MEGNETOMETER
CONNECTORS
Camera Connectors
IMU
SDCARD
BUZZER
I2C
CONTROLLER
VOLTOMETER
POWER
Futher checking required
MILLENIUNNIUM
Sheet: /
File: CanSat.kicad_sch
Title: Drone
Size: A3
Date: 2021-01-05
KiCad E.D.A. eeschema (5.99.0-7136-gf5e9a2a6da)
Rev: v2
Id: 1/1

Projektowana przez nas płytka będzie zawierać wszystkie używane części elektroniczne za wyjątkiem kontrolera szybkości silników (ang. Electronic Speed Controller), GPSu oraz czujnika odległości, który umieściliśmy na osobnej płytce ze względu na jego umiejscawianie pod bateriami. Pozwoli to na znaczne zmniejszenie konstrukcji oraz zmniejszy jego ewentualne koszty produkcji.

W celu działania brzęczyka przy jednoczesnym wyłączaniu głównych systemów, zbudowaliśmy przerzutnik typu RS, umożliwiając wielogodzinne wydawanie dźwięku nawet po stosunkowo długim locie, co nie tylko ułatwia znalezienie drona w przypadku straty sygnału, ale też pozwala na sygnalizowanie swojej obecności osobie, do której miał on dolecieć.

Poniżej znajduje się projekt aktualnie zbudowanej wadliwej płytki, wraz z jej zdjęciami. Jej wymiary to 40 na 72 mm.



Kontroler lotu

Zadaniem kontrolera jest stabilizacja drona podczas lotu oraz odchylanie go od podany przez komputer kąt.

Podczas prostych testów lotu używaliśmy kupnego, gotowego kontrolera lotu. Jednak duża część jego funkcji była dublowana przez naszą płytkę PCB, a do tego zajmował niepotrzebnie dużo miejsca. Dlatego zdecydowaliśmy się na zbudowanie własnego kontrolera, zintegrowanego z resztą podzespołów. Nie skończyliśmy jeszcze jego projektu, a jedynie jego zamieszczony wyżej schemat.

Zdecydowaliśmy się go oprzeć o 32 bitowy moduł stm32f405vgt6, ze względu na jego dobre parametry, odpowiednie interfejsy, dobrą dokumentacje techniczną oraz wsparcie producenta ułatwiające napisanie na niego niskopoziomowych programów.

Komunikacja

W komunikacji bezprzewodowej ciężko jest pogodzić wymagany przez nas zasięg i dużą przepustowość danych. Dlatego zdecydowaliśmy się na wykorzystaniem dwóch różnych częstotliwości danych, które będziemy wykorzystywać do różnych celów.

433MHz idealnie nadaje się do przesyłania niewielkich danych na odległość nawet paru kilometrów. Dlatego będziemy za pomocą tego sygnału przesyłać aktualną pozycję i stan baterii, jak i ewentualne instrukcję dotyczące chociażby powrotu. Z kolei Wi-Fi 2,4GHz umożliwia przesyłanie dużych danych, jednak jego zasięg ogranicza się zwykle do kilkunastu- kilkudziesięciu metrów. W celu umożliwienia łączenia się z dronem z urządzenia bez zewnętrznych anten, będziemy przesyłać wszystkie dane tą częstotliwością. W przypadku, w którym urządzenie otrzymuje dane równolegle, używa tych z częstotliwości 433MHz.

Tak jak wspomnieliśmy wyżej, producent używanego przez nas komputera ma problemy z certyfikacją jego wersji zawierającej wbudowane Wi-Fi, którego to zamierzamy użyć, przez co nie mieliśmy jeszcze okazji przetestować jego dokładnego zasięgu. Częściowo, na innej płytce i wyłącznie w otwartym terenie, przetestowaliśmy już moduł 433MHz. Dzięki zastosowaniu anteny typu Yaga, udało nam się osiągnąć zasięg co najmniej dwóch kilometrów.

Procedura

Lot z punktu startu do danego punktu końcowego przebiega według następującej procedury:

1. Włączenie wszystkich urządzeń wraz z ich opcjonalną kalibracją.
2. Wzniesienie się na wysokość 2 metrów
3. Pobranie informacji dotyczących docelowego miejsca lądowania
4. Obrót w kierunku celu
5. Wykonanie skanu 3D
6. Wyznaczenie ścieżki
7. Podążanie ścieżką; dron jednocześnie nieustannie kontroluje swoje otoczenie i jeśli zachodzi taka potrzeba, modyfikuje ścieżkę
8. Po dotarciu na współrzędne x oraz z celu (gdzie y oznacza kierunek pionowy) następuje procedura lądowania, opisana dalej

Przyszłe kroki

Do końca połowy stycznia skończmy przeprowadzać testy współdziałania poszczególnych części drona, za wyjątkiem płytki, która będzie gotowa do końca stycznia. Następnie, na początku lutego, zamierzamy przeprowadzić pierwsze testy całości. Jednak do tego czasu przetestujemy szczegółowo każdy algorytm oraz protokół.

Następnie do końca lutego planujemy połączyć całą naszą pracę razem i stworzyć w pełni autonomicznego drona. Planujemy, że do tego czasu będzie on w stanie wykonać niekontrolowany przez człowieka lot na dystansie paru kilometrów do wcześniej wybranego celu, a także będzie czasie lotu zbierać i wysyłać wiele informacji o terenie, w jakim się porusza. Będzie mógł być również łatwo dostosowany do innych celów, takich jak przyniesienie ładunku do rannej osoby znajdującej się w niedostępnym terenie czy nawet wykrywanie i zbieranie śmieci w lesie. Dlatego do końca marca zamierzamy dodać możliwość podnoszenia i przenoszenia drobnych ładunków, do czego już dostosowaliśmy projekt płytki.

Następnie zamierzamy usprawnić znajdowanie pozycji za pomącą orientacji wizualnej, co ułatwi korzystanie z niego w zamkniętych pomieszczeniach, bez zasięgu GPSu. Do końca lipca planujemy zbudowanie własnego kontrolera szybkości motorów, co znacznie zmniejszy koszty ewentualnej produkcji drona.

Zaczęliśmy również tworzenie aplikacji na komputer, która umożliwiłaby wygodne sterowanie dronem wraz z monitorowaniem wszystkich jego odczytów lub wyznaczanie celów. Obecnie posiada jedynie część wszystkich planowanych funkcji, dlatego w przyszłości planujemy dokończenie jej, aby umożliwić jego komfortowe i intuicyjne użytkowanie.

Podziękowania

Chcielibyśmy podziękować opiekunce naszego projektu- Annie Rzepie, która pomaga nam odpowiednio ukierunkować projekt i zawsze służy wsparciem, oraz Panu Mariuszowi Pułasowi, który wspomógł nas cenną konsultacją.

[1]https://www.researchgate.net/publication/328312350_Development_of_Autonomous_Drones_for_Adaptive_Obstacle_Avoidance_in_Real_World_Environments

[2] https://pl.wikipedia.org/wiki/Algorytm_A*

[3] https://pl.wikipedia.org/wiki/Algorytm_Dijkstry

[4] https://www.mathworks.com/matlabcentral/fileexchange/24693-ellipsoid-fit

[5] https://users.soe.ucsc.edu/~elkaim/Documents/ReimmanTAES08.pdf

[6] https://www.x-io.co.uk/res/doc/madgwick_internal_report.pdf

[7] https://en.wikipedia.org/wiki/Conversion_between_quaternions_and_Euler_angles

[8] https://en.wikipedia.org/wiki/Quaternions_and_spatial_rotation

[9] https://en.wikipedia.org/wiki/Extended_Kalman_filter

[10] https://en.wikipedia.org/wiki/Kalman_filte

[11] https://en.wikipedia.org/wiki/Alpha_beta_filter

[12] [Yury (2021). Ellipsoid fit [4], MATLAB Central File Exchange. Retrieved January 6, 2021

[13] Geometric Approach to Strapdown Magnetometer Calibration in Sensor Frame 5