Realizacja

Ta część ma wpływ naukowy i twórczy, a dowiesz się o tematach, które są również pomijane w wielu publikacjach, związanych z podstawowymi szczegółami implementacji, takimi jak przetwarzanie sygnałów dla czujników i siłowników, transmisja danych, sposoby programowania drona oraz słowniczek z główne polecenia używane z najpopularniejszymi interfejsami programistycznymi. Dzięki temu będziesz w stanie zwięźle zrozumieć, jak zaprogramować drona lub inny pojazd zrobotyzowany.

Zadania drona

Istnieje sześć podstawowych zadań drona:

• Silnik: To jest najbardziej podstawowe i najważniejsze zadanie. Ogólnie składa się z automatycznego sterowania w pętli otwartej, które jest funkcją innych sterowników w pętli zamkniętej pojazdu. Prędkość każdego silnika jest modyfikowana, co ma wpływ na zachowanie drona. To zadanie jest tym, które przedstawia najszybszą zmienność i w konsekwencji wymaga aktualizacji z wysoką częstotliwością. Silniki muszą otrzymywać macierz napędu lub alokacji w sposób automatyczny.

• Postawa lub orientacja: Jest to drugie najważniejsze zadanie (również pod względem częstotliwości wykonywania). Jest to ważniejsze niż zadanie wysokościowe, ponieważ oznacza, że samolot jest odpowiednio wyważony. Gdyby tak nie było, dron mógłby losowo poruszać się po samolocie i w najgorszym wypadku spaść. Ze względu na znaczenie w locie pojazdu jest to zwykle w pełni automatyczna pętla zamknięta.

• Wysokość lub wzniesienie: jest to trzecia istotna kwestia, ponieważ każdy samolot oznacza zdolność do unoszenia się na wodzie. Z tego powodu osoby z wiedzą wstępną uważają, że to zadanie jest najważniejsze, ale jak już wspomnieliśmy, brak równowagi ma tendencję do destabilizacji drona. Zwykle jest to automatyczna pętla zamknięta, ale możesz nadać jej zachowanie półautomatyczne za pomocą pilota.

• Planarny: jest to czwarty pod względem znaczenia. Polega ona na kontrolowaniu pozycji drona, która jest prostopadła do osi wzniesienia (pozycja, którą dron utrzymuje przy ziemi). To zadanie może być również automatyczne lub półautomatyczne.

• Planowanie trajektorii: w tym przypadku jest to jednoczesna zmiana położenia, wysokości i zadań planarnych. Ta zmiana odbywa się w zależności od czasu lub określonej ścieżki od punktu do punktu. Może być automatyczny lub półautomatyczny. Zasadniczo jest to nawigacja po zbiorze punktów.

• Zdalne sterowanie: Jest to ręczny element zadań lotu. Jest to kontrola człowieka w zamkniętej pętli, ponieważ to pilot reguluje błąd zgodnie z pomiarem zapewnianym przez jego wzrok i działanie przycisków sterujących. Ponieważ akcja jest zależna od człowieka, zadanie to jest tak powolne i nieprecyzyjne, jak możliwości pilota. Jego użycie z innymi zadaniami daje im tryb półautomatyczny. W niektórych przypadkach pilot zmienia tylko żądane wartości, aw innych modyfikuje bezpośrednio efekt sterowania. Nie zaleca się, aby pilot bezpośrednio modyfikował zadania orientacyjne (przyciski i drążki poruszają się wolniej i gwałtowniej niż wymagane automatyczne ustawianie pozycji).

Teraz, gdy już mówisz o zadaniach drona, porozmawiajmy o rodzaju kontrolerów, których używają.

Pętle i rodzaje kontrolerów dla drona

Quadkopter to system, który służy jako dobry przykład do opisu różnych typów kontrolerów i pętli sterowania. Zostały one omówione w kolejnych sekcjach.

Pętle kontrolne

Otwarte i zamknięte pętle

Ogólnie wysokość, położenie planarne i orientacja drona są zamkniętymi pętlami, co oznacza, że ich korekcja błędów opiera się na sprzężeniu zwrotnym. Jest jednak zadanie, które zależy od otwartej pętli. Jest to kontrola prędkości silników. Otwarta pętla to taka, w której zadanie jest wykonywane bardziej w wyniku proporcjonalności niż w wyniku korekcji błędów. W przypadku silników wysyłane jest napięcie i obserwujemy proporcjonalną prędkość silnika bez zamkniętej pętli, aby zweryfikować i skorygować tę prędkość.

Pętle wewnętrzne i zewnętrzne

Quadkopter ma cztery silniki, a dzięki swojej geometrycznej konfiguracji może bezpośrednio sterować czterema zmiennymi, którymi są wysokość i położenie (trzy orientacje). Nazywa się to pętlą wewnętrzną. Ponieważ jednak wymagane jest, aby pojazd mógł poruszać się również w płaszczyźnie XY, odbywa się to w sposób zależny od orientacji przechyłu i pochylenia w zewnętrznej pętli sterowania.

Pętle zależne i niezależne

Niezależne pętle występują we wszystkich zmiennych, których sterowanie nie wymaga monitorowania dodatkowej zmiennej; przykładami są kontrola wysokości i kontrola odchylenia. Z drugiej strony pętle zależne wymagają informacji o jednej lub więcej dodatkowych zmiennych; przykładami są elementy sterujące X i Y, które zależą od orientacji pojazdu.

Pętle pozycji i prędkości

Quadkopter to pojazd, którego pętle pozycji (orientacja i pozycja) zależą od otwartej pętli prędkości każdego z jego silników. Serwomotory są siłownikami pozycji, a bezszczotkowe silniki prądu stałego są zwykle siłownikami prędkości.

Rodzaje kontrolerów

Kontrolery wolne od modelu i inne niż wolne od modelu

Niektóre zmienne, takie jak przechylenie, pochylenie i orientacja odchylenia, można regulować za pomocą „prostego” kontrolera PD. Oznacza to, że nie jest wymagana kompensacja ich częściowej lub pełnej dynamiki. Z drugiej strony przy kontroli wysokości konieczne jest wprowadzenie elementu kompensacji masy pojazdu (częściowy sterownik bezmodelowy).

Solidne i adaptacyjne kontrolery

W przypadku zmiennych regulowanych przy użyciu PD bez jakiejkolwiek kompensacji zakłada się, że PD są wystarczająco duże w odniesieniu do dynamiki pojazdu. W tym przypadku sterowanie nazywa się dominującym lub solidnym (o ile silniki pozwalają na tę odporność). Innym przykładem solidnych sterowników jest rodzina trybów przesuwnych. W zasadzie nie wymagają modelu systemu, tylko pętli błędów. W przypadku wysokości, gdy waga jest kompensowana, jest to bardzo szczątkowy przykład kontroli adaptacyjnej. Jeśli jednak wzmocnienia PD są wystarczająco duże, a silniki mogą same przeciwdziałać temu efektowi, człon kompensacyjny można pominąć. Ogólnie rzecz biorąc, kontrolery adaptacyjne są zaprojektowane do kompensacji częściowej lub pełnej dynamiki lub niemodelowanych perturbacji.

Nieograniczona i ograniczona kontrola

Nieograniczona kontrola to taka, która może używać nieograniczonego zakresu wartości, na przykład wszystkich wartości dodatnich i ujemnych. Ten rodzaj kontroli nie istnieje, ale możesz pomyśleć, że jest to opłacalna opcja ze względu na symulatory. Z drugiej strony, ograniczone kontrolery wprowadzają ograniczenia operacyjne. Pierwsze ograniczone sterowanie znalezione w quadkopterze jest podawane przez silniki, ponieważ mają one maksymalne i minimalne wartości prędkości i momentu obrotowego. Druga ograniczona kontrola jest bardzo zauważalna przy użyciu silników bezszczotkowych (zakładając, że nie są one dwukierunkowe). Jest to obecność wyłącznie pozytywnych wartości operacyjnych. A ostatnia ograniczona kontrola, nie tak widoczna w dronach, jest determinowana przez efekty fizyczne, takie jak czas reakcji. W pierwszym ograniczeniu stosuje się nasycenia. W drugim wymagane są współczynniki skalowania. W trzecim wykorzystywane są techniki kontroli opóźnień. Ten ostatni efekt nie jest tak widoczny na zwykłych dronach ze względu na szybką reakcję autopilotów i komputerów i zwykle jest pomijany, ale w samolotach, które używają śmigieł dwukierunkowych, należy wziąć pod uwagę te opóźnienia, ponieważ śmigła nie zmieniają kierunku obrotu momentalnie.

Sterowanie liniowe i nieliniowe

Ten rodzaj kontroli zależy od warunków pracy. Widzieliście kilka trybów lotu, w których ruch jest płynny, a kąty są ograniczone do małego obszaru wokół zera, więc słuszną decyzją było zlinearyzowanie zarówno modelu, jak i kontrolera. Opisaliśmy jednak również niektóre tryby lotu, w których kąty nie są ograniczone, a pojazd może wykonywać agresywne ruchy; dlatego sterowanie i model mają nieliniowości.

Praca ciągła i oparta na zdarzeniach

W tym przypadku kontrolerem, który jest zawsze stały i który musi być obecny podczas pełnego działania statku powietrznego, jest kontroler położenia. Tym, który zależy od zdarzeń, takich jak sygnały czasowe lub interakcja z pilotem, jest sterowanie trajektorią.

Sterowniki ciągłe i dyskretne

Tu pojawia się druga definicja ciągłości, która jest związana z tym, jak mały jest czas próbkowania. Zależy to od dwóch czynników: pierwszy to komputer, a drugi siłowniki i czujniki. W przypadku komputerów, dziś prawie wszystkie są na tyle szybkie, aby traktować ich działanie jako modele ciągłe lub układy oparte na równaniach różniczkowych. Jednak zmieniając technologię siłowników od elektrycznych silników bezszczotkowych do silników spalinowych, zauważysz, że ten drugi rodzaj silników ma wolniejsze czasy reakcji w porównaniu z silnikami elektrycznymi (mają dużo elementów, tarcie i cykle pracy) i wymagają do analizy równań różnicowych (ich modelowanie za pomocą równań różniczkowych nie jest już możliwe).

Sterowanie elektryczne i mechaniczne

Nadużywamy niektórych technologii, ponieważ podzespoły elektryczne i elektroniczne są gwarantowane przez producenta do prawidłowego działania, więc na przykład nie martwimy się o działanie ESC. Polegamy również na tym, że urządzenia elektryczne są szybsze w stosunku do elementów mechanicznych. Z tych powodów nasz projekt ignoruje komponenty elektryczne i elektroniczne i traktujemy drona jako system z czysto mechanicznym modelowaniem i sterowaniem. Jednak nowoczesne projekty, które wprowadzają sterowanie w pętli zamkniętej prędkości silników, mogą być postrzegane jako sterowanie pośrednie lub elektromechaniczne, gdzie prędkość mechaniczna silników zależy od zmiennych elektrycznych, takich jak fazy elektryczne ESC. Patrz Załącznik, aby uzyskać informacje na temat sterowania silnikiem w pętli zamkniętej. Poznałeś już

zadania drona i większość kontrolerów, z których może korzystać ten pojazd/robot. Kontynuujmy podstawowe przetwarzanie sygnału, którego wymaga dron i przykład zastosowania.

Przetwarzanie sygnału dronów

Może kusi Cię wdrożenie poprzednich kontrolerów. Jednak przed tą implementacją wymagane są pewne kroki. Najczęstsze to:

• Filtrowanie sygnału, które jest stosowane w czujnikach

• Mapowanie sygnału, które jest używane w czujnikach, drążkach zdalnego sterowania, podczas transmisji danych itp.

• Odlewanie sygnału lub konwersja danych, która jest wykorzystywana do wprowadzania wartości sterującej do silników lub podczas szeregowej transmisji danych

• Nasycenie sygnału, które służy do ograniczania wartości wysyłanych do silników lub do ograniczania wartości odczytywanych z drążka zdalnego sterowania

• Normalizacja sygnału, która służy do narzucenia zakresu pracy w czujnikach

Filtrowanie sygnału

Jak widać w poprzednich rozdziałach, głównym składnikiem regulatorów z zamkniętą pętlą jest błąd, który również zależy od mierzonych zmiennych i jest to osiągane za pomocą czujników. Jednak te czujniki zwykle mają szum elektryczny z powodu własnego zachowania lub z powodu ich interakcji z otoczeniem. Dlatego wymagany jest etap filtrowania w celu zmniejszenia lub stłumienia szumu elektrycznego. Ponieważ implikuje się przetwarzanie obliczeniowe lub elektroniczne, tłumienie szumu jest zgodnością między wygładzeniem sygnału szumu a opóźnieniem wygładzonego sygnału. Można to osiągnąć na kilka sposobów. Oto kilka przykładów:

• Pasywny: Tutaj stosowane są dodatkowe urządzenia, które z natury pochłaniają część hałasu otoczenia. Przykładami są gąbki rozpraszające lub płyny do ruchu mechanicznego, soczewki pochłaniające częstotliwość światła lub pasywne urządzenia elektroniczne, które mogą być używane jako filtry.

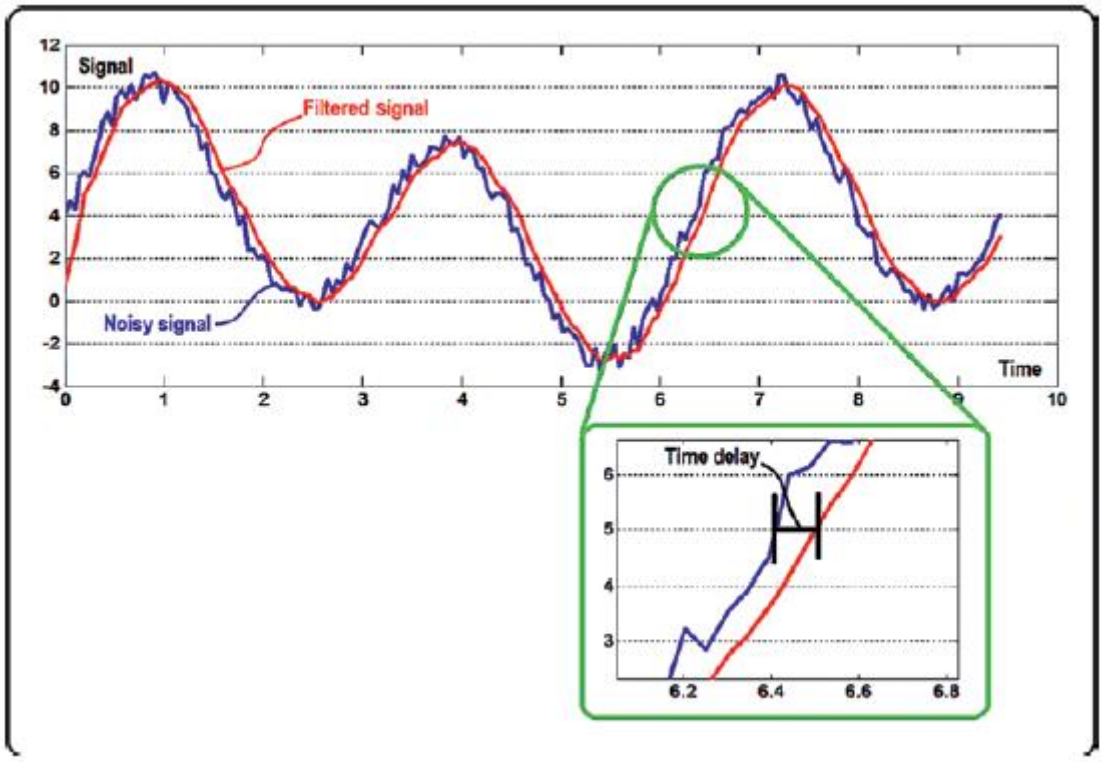
• Aktywne przez filtrowanie elektroniczne: dobrze znane przykłady to obwody dolnoprzepustowe, górnoprzepustowe, pasmowoprzepustowe i pasmowo-odrzucające.

• Aktywny przez filtrowanie matematyczne: sygnał jest wysyłany do filtru matematycznego. Wśród najbardziej znanych filtrów znajdują się filtr Kalmana i obserwator Luenbenger.

• Aktywny przez transformację przestrzeni: sygnał jest konwertowany do innej przestrzeni w celu wykonania prostego filtrowania w nowej przestrzeni, takiego jak filtrowanie na poziomie częstotliwości zamiast na poziomie czasu. Ta metoda może być również uważana za filtr matematyczny, ale ma swój własny kierunek badań. Przykładami są transformacje Fouriera i Wavelets.

• Aktywny dzięki inteligentnemu filtrowaniu: odbywa się to za pomocą dowolnego algorytmu wykorzystującego sztuczną inteligencję

Kompromis między redukcją szumów a opóźnieniem sygnału można zobaczyć na rysunku (użyto kodu ogólnego filtra dolnoprzepustowego).



Na listingu można zobaczyć ogólny kod wysokiego poziomu do używania tego typu filtrów (w tym przypadku MATLAB).

filteredSignal=lowpassFilter(noisySignal)

Nasycenie

W takim przypadku konieczne jest nałożenie wartości granicznych na sterowniki, aby maksymalne i minimalne wartości, które mają być wstrzykiwane, nie przekraczały granic silnika. Innym zastosowaniem jest ograniczenie wartości, które użytkownik może osiągnąć w drążku pilota. Nasycenie to można wykonać skokowo (na przykład za pomocą funkcji znaku, która jest równoważna czynności włączania i wyłączania) lub w sposób ciągły (za pomocą funkcji zwanej nasyceniem). Nawet funkcja ciągła może być gładka lub ostra (można użyć wielu funkcji z rodziny saturatorów zwanych funkcjami sigmoidalnymi). Zazwyczaj nieograniczona część saturatora kopiuje oryginalny sygnał jako standardową funkcję liniową, jak pokazano w kodzie z Listingu .

Mini=0

Maxi=7

if(signal<=Mini)

satsignal=Mini;

elseif(signal>=Maxi)

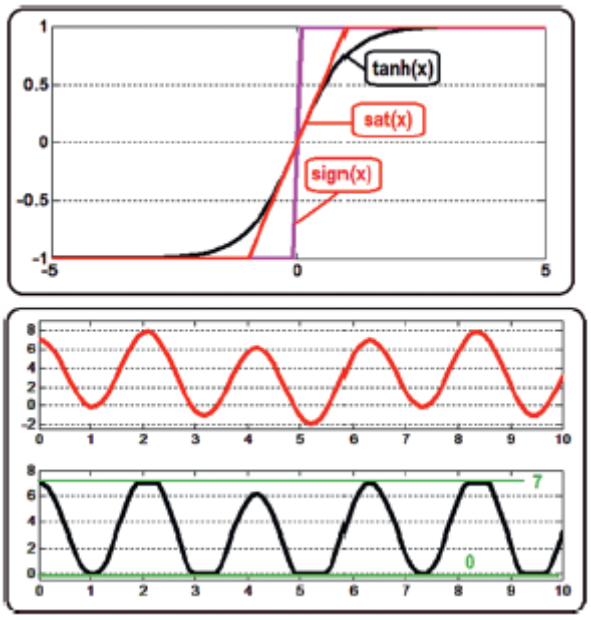
satsignal=Maxi;

else

satsignal=signal;

end

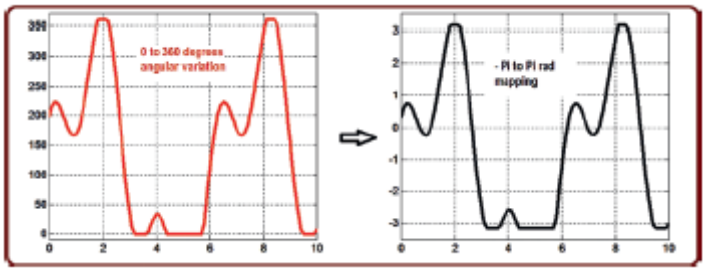
Jednak, aby przeciwdziałać niepożądanemu działaniu (na przykład nagłemu zachowaniu drążka zdalnego sterowania), czasami stosuje się inne typy sigmoidów. Rysunek



ilustruje dwie rzeczy. Górna część to rodzina saturatorów, w tym przypadku funkcja znaku, funkcja saturacji i funkcja tangensa hiperbolicznego. Oczywiście projektant musi je tak zmodyfikować, aby osiągnęły wartości minimalne i maksymalne zgodnie ze swoimi zastosowaniami, a nie tylko -1 i 1. Dolna część to zastosowanie funkcji nasycenia przy danym sygnale, w tym przypadku dopuszczalne wartości minimalne i maksymalne przejdź od 0 do 7. Kod użyty na rysunku jest naprawdę prosty i jest wyświetlany na listingu .

Stronniczość i mapowanie

Kolejnym ważnym zadaniem jest przekształcenie sygnału sterującego (sygnał o wartościach dodatnich i ujemnych) w jeden sygnał kompatybilny z silnikami (np. w zakresie wartości 1000-2000 w bezszczotkowych silnikach RC). Innym przykładem jest zasięg czujników, który może być dowolny i musi zostać przekonwertowany na standardowy zakres roboczy (0 do 360 stopni, -180 do 180 stopni, -1 do 1 metra itd.). Zobacz rysunek .



W tym przypadku wykorzystywane są funkcje skalowania i tłumaczenia. Jedną z najczęściej używanych technik odchylania i mapowania jest równanie linii między dwoma punktami znane jako mapowanie liniowe, które dostosowuje wartość wejściową do równania linii prostej (wartość wyjściowa). W tym celu sygnał wejściowy musi być wcześniej ograniczony w taki sposób, aby nie przekraczał

maksymalnych i minimalnych limitów wejściowych. Kod jest również prosty i jest pokazany na listingu (zauważ, że ten kod mapowania opisuje linię między dwoma punktami).

```
mininput=0;

minoutput=-pi;

maxinput=360;

maxoutput=pi;

mapsignal=(signal - mininput) * (maxoutput - minoutput) /

(maxinput - mininput) + minoutput;
```

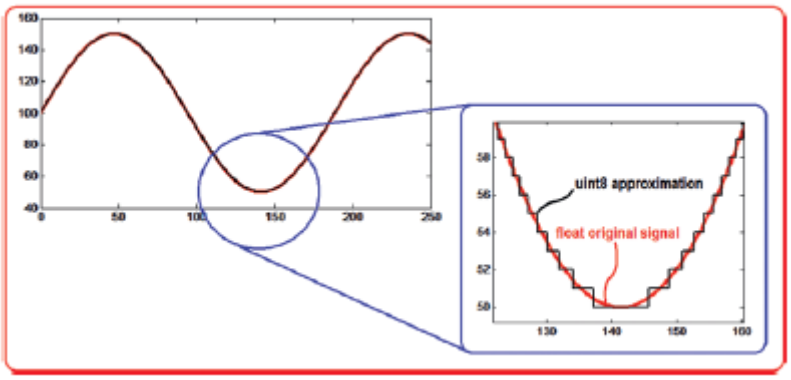
Przesyłanie danych

Jest to kolejna ważna adaptacja sygnału, którą należy przeprowadzić, gdy polecenie dopuszcza tylko jeden typ danych i otrzymuje dane innego typu. Na przykład polecenia zapisu do silnika RC typu PWM zwykle wymagają danych typu integer, a podane wartości są liczbami zmiennoprzecinkowymi lub ułamkami. W tym przypadku używamy tego, co w programowaniu nazywa się rzutowaniem, co jest transformacją z jednego typu danych na inny. Może to obejmować dwie rzeczy:

• Casting nie istnieje i użytkownik musi zaprojektować ten program.

• Odlewanie generuje aliasing, który wpływa na jakość pomiarów lub wykonania (np. sterownik może przejść z płynnego i naturalnego ruchu do gwałtownego i zrobotyzowanego lub z obrazu o wysokiej rozdzielczości do obrazu o niskiej jakości).

Rysunek przedstawia typowy przykład.



Sygnał czerwony reprezentuje wyliczoną wartość sterującą (z dużą liczbą miejsc po przecinku) i w momencie podania tego sygnału do silników, ponieważ polecenie zapisu dopuszcza inny typ danych (np. uint8), dokładność wartości pierwotnej jest zmniejszona. Z wyglądu wyglądają podobnie, ale jeśli przyjrzymy się szczegółowo, zaprezentowany zostanie efekt schodkowy zwany aliasingiem. Ten efekt może sugerować mniej płynne zachowanie w ruchu drona. W zależności od języka programowania polecenia rzutowania mogą, ale nie muszą istnieć (musi je utworzyć użytkownik), ale generalnie przyjmują postać wyświetlaną na listingu

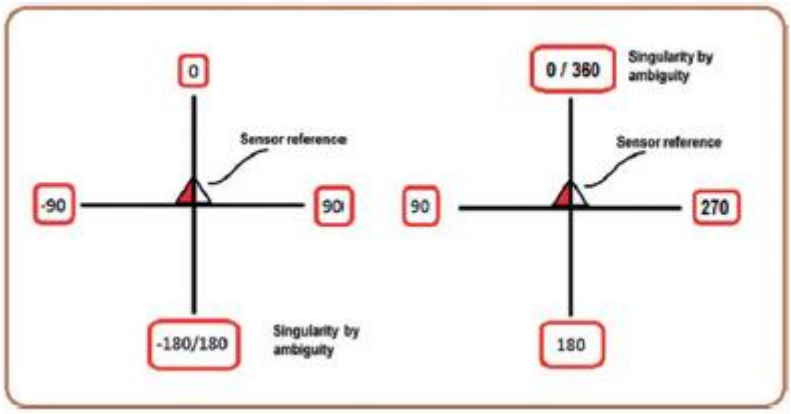
```
signal=100+50*sin(t/30);

castSignal = uint8(signal);
```

Zauważ, że jeśli rzutowanie istnieje, oznacza to, że istnieją funkcje, które wymuszają zmianę typu danych sygnału lub danych wejściowych (w tym przykładzie uint8).

Normalizacja redundancji i osobliwości

Jest to częsty problem spotykany w czujnikach kątowych i polega on na tym, że mają one przeskok od wartości maksymalnej do wartości minimalnej np. z 359 stopni na 0 stopni lub nadmiarowe wartości znane jako osobliwości takie jak 0 stopni i 360 stopni (tak dzieje się również przy -180 stopniach i 180 stopniach, jeśli czujnik działa w ten sposób).



Aby poradzić sobie z takimi warunkami osobliwości, niejednoznaczności lub nadmiarowości, istnieje wiele metod normalizacji. Jeden z nich (który dotyczy przypadku z lewej strony rysunku) opiera się na symetrycznym określeniu błędu kątowego względem wartości mierzonej.

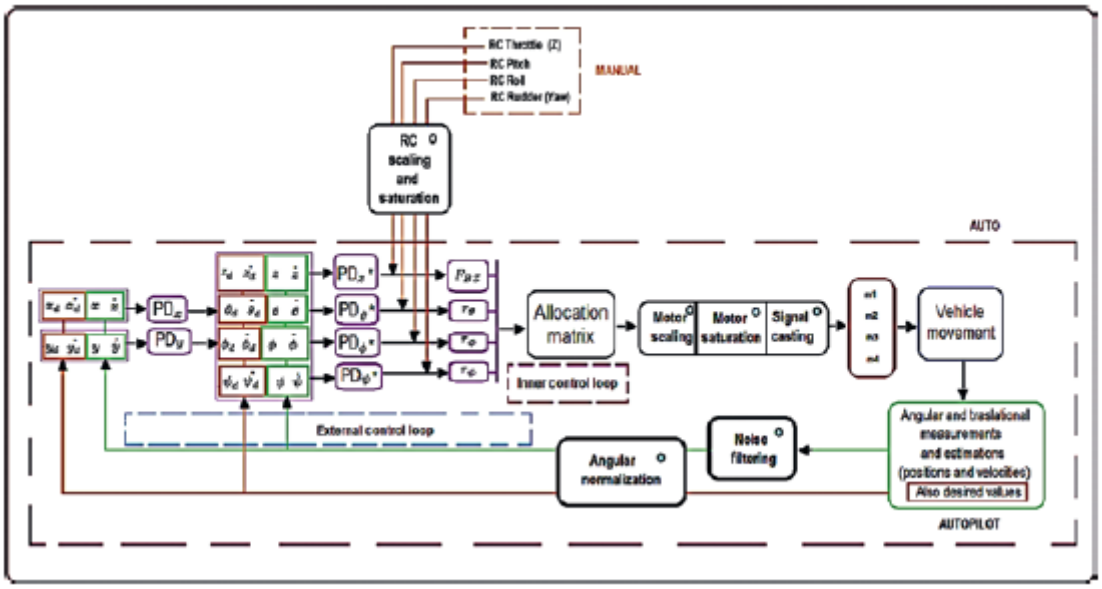
$$e_\psi = \psi_d - \psi$$

$$e_{\psi\,\mathrm{Norm}} = \begin{cases} e_\psi - 360 \text{ if } e_\psi > 180 \\ e_\psi + 360 \text{ if } e_\psi < -180 \\ e_\psi \text{ in another case} \end{cases}$$

Jest to normalizacja dynamiczna, której dodatkowym celem jest zmniejszenie obrotu pojazdu w sposób implikujący krótszą odległość. Aby osiągnąć pożądaną orientację, pojazd może to zrobić, obracając się w lewo lub w prawo. Ten algorytm, oprócz zapobiegania osobliwościom, wybiera, czy lepiej poruszać się zgodnie z ruchem wskazówek zegara, czy przeciwnie do ruchu wskazówek zegara. Należy również opracować lub zbadać alternatywne metody normalizacji, które są oparte na wartości pożądanej, a nie zmierzonej, lub oparte na bezwzględnym odniesieniu zamiast odniesienia pojazdu, lub metody oparte na wielu obrotach i metodach pomiarów kątowych od 0 do 360 stopni itp.

Przykład użycia

Rysunek



przedstawia modyfikację sterownika półautomatycznego bez ruchu odchylającego (lub bardzo małego), jak opisano wcześniej. Ta odmiana obejmuje w ten sposób pięć podstawowych metod przetwarzania sygnału: mapowanie i nasycenie pilota zdalnego sterowania jest stosowane w celu dostosowania jego wartości w celu uzyskania akceptowalnych poleceń kątowych. Filtrowanie jest stosowane do czujników pozycji w celu zmniejszenia ich poziomu hałasu. Błąd kątowy jest znormalizowany. Błąd kątowy jest znormalizowany. Przed wstrzyknięciem macierzy alokacji do silników stosuje się skalowanie, aby ich sygnały były dodatnie. Ogólnie rzecz biorąc, silniki dronów mają stały kierunek obrotu, a ich prędkość może być zmieniana od zera lub wartości początkowej do maksymalnej prędkości, dlatego w sygnałach silnika stosuje się nasycenie, aby nie przekroczyć ich wartości początkowej i maksymalnej. Wreszcie, rzutowanie jest wykonywane, ponieważ generalnie polecenia zapisu do silników żądają typu liczby całkowitej, a sygnały sterujące są wartościami zmiennoprzecinkowymi. Poznałeś podstawy przetwarzania sygnałów dla dronów. Kontynuujmy podstawy teorii transmisji danych, która jest wykorzystywana w większości środowisk programowania dronów, które zwykle bazują na transmisji szeregowej typu UART.

Teoria transmisji danych

Teoria transmisji danych jest uważana za temat o dużym znaczeniu, ponieważ umożliwia komunikację między zespołem dronów lub między pojedynczym dronem a odległą bazą. Wśród dostępnych prezentacji opisujemy tutaj typ UART (Universal Asynchronous Receiver-Transmitter), ponieważ jest to jeden z najbardziej podstawowych i wykorzystywanych protokołów z dziesięcioleciami użytkowania i jest naprawdę powszechny w SDK dronów.

Typy i podtypy danych

Zanim zaczniemy mówić o tym standardzie, wygodnie jest porozmawiać o typach i podtypach danych. Ten temat jest związany z wprowadzoną wcześniej konwersją danych. W poniższych akapitach będziemy używać pojęć C/C++, ponieważ jest to najpopularniejszy język programowania występujący w różnych pakietach SDK do tworzenia aplikacji dronów. Jest to jednak bardzo częsty temat wśród innych języków programowania i każdy z nich ma następujące podstawowe typy (lub bardzo podobne).

Typ: opis

int : Używany do liczb całkowitych

float : Używany do liczb z wartościami dziesiętnymi

double : zwiększa dokładność pływaków i jest przydatny do obliczeń naukowych lub inżynierskich

char : Pozwala na użycie znaków

bool : Pozwala na użycie wartości logicznych

Jednak w zależności od przeprowadzanych operacji, takich jak zarządzanie czasem, zarządzanie pamięcią i protokoły komunikacyjne, często można znaleźć warianty tych podstawowych typów danych. Warianty te są identyfikowane przez włączenie przedrostków i przyrostków. W programowaniu dronów bardzo często stosuje się warianty typu integer. Na przykład uint8 ma przedrostek u i przyrostek 8. Ogólnie rzecz biorąc, przedrostek wskazuje ograniczenie znaku, a przyrostek ograniczenie zakresu na podstawie bitów. Istnieje 8, 16, 32 i 64-bitowe liczby całkowite, które mogą przechowywać do 255 (255 = 2 podniesione do ósmej potęgi minus 1, aby zacząć od elementu 0), odpowiednio 65535, 4294967295 i 18446744073709551615. Prefiks u wskazuje, że możliwe do użycia wartości mają tylko znak dodatni i 0. Czyli w przypadku uint8 jego użyteczny zakres wynosi od 0 do 255. Zamiast tego, jeśli użyjemy opcji int8, będziemy mieli również 255 wartości, ale w zakresie od -128 do 127. Przydatność przedrostków i przyrostków jest opisana na przykładach. Wyobraźmy sobie proces, który musi zostać wykonany w mikrosekundach. Jeśli użyjemy podtypu danych uint8, będziemy mieli tylko 255 mikrosekund, co nie jest nawet milisekundą czasu. Jeśli użyjemy uint16, będziemy mieli tylko 65 milisekund. Wreszcie, wybierając podtyp uint32, odpowiednik w sekundach wynosi 4294 lub około 71 minut. Przejdźmy nieco dalej do wykorzystania UART, aby wskazać, że jest to komunikacja, która obsługuje tylko 8-bitowe bloki bez znaku. W ten sposób użycie podtypów 16 i 32 nie jest możliwe, a tym bardziej podtypu z wartościami ujemnymi. Z tego powodu tutaj znajduje się użyteczność podtypów uint8. Wreszcie, w wielu środowiskach programistycznych można znaleźć agregat t w przyrostku liczbowym (na przykład uint16t). Ma to tylko właściwość bycia kompatybilnymi danymi między różnymi platformami sprzętowymi. Kiedy opracowano typy danych 8, 16 i 32, zużycie pamięci różniło się między różnymi platformami obliczeniowymi. W ten sposób producenci doszli do porozumienia w sprawie przenoszenia kodu, tworząc standard oznaczony literą t.

Wprowadzenie do UART

Zauważ, że zakłada się zestaw SDK z poleceniami UART wysokiego poziomu, takimi jak read(), write(), begin() lub available(), jak ten prezentowany w Arduino lub Ardupilot. Użycie UART niskiego poziomu jest zarezerwowane dla książek o języku maszynowym, języku asemblera lub mikrokontrolerach. Transmisja UART jest opisana, ponieważ jest jedną z najczęściej stosowanych w obecnych autopilotach i ich odpowiednich pakietach SDK. Jest dwustronna, ale asynchroniczna (bez udziału timera, a co za tym idzie bez zapewnienia czasu transmisji i odbioru). Wskazaliśmy również, że standard ten opiera się na zarządzaniu 8-bitowymi pakietami danych dodatnich i całkowitych. Aby kontynuować, zadajemy następujące pytania. Biorąc pod uwagę wskazane ograniczenia,

• Jak pracujesz z danymi negatywnymi?

• Jak pracujesz z danymi dziesiętnymi (liczba z miejscami dziesiętnymi)?

• W jaki sposób gwarantowane są czynności czytania i pisania, jeśli nie ma licznika czasu do czytania lub pisania?

• A jak pracujesz z danymi większymi niż wartość 255?

Aby odpowiedzieć na te pytania, zaczynamy od koncepcji licznika cyklicznego. Łatwo to zrozumieć, jeśli odniesiemy się do ruchu obrotowego. W ruchu obrotowym ciało może być zorientowane od 0 do 360 stopni lub od 0 do 2 pi radianów, w zależności od zastosowanego standardu. Wartości wyższe niż ten zakres oznaczają wykonanie wielokrotnych obrotów lub umieszczenie w operacji modulo wskazanej wartości z 360 lub 2pi. Zatem 800 stopni odpowiada dwóm obrotom o 360 stopni i nadwyżce 80 stopni. Zauważ, że dowolny kąt jest funkcją modulo lub reszty o podstawie 360 lub 2pi oraz ilorazu tego kąta o tej samej podstawie (zauważ, że interesują nas tylko części całkowite modulo i ilorazu).

$$N = N \bmod(360) + 360(quotient(N/360))$$

W poprzednim przykładzie

$$800 = 800 \bmod(360) + 360(quotient(800/360))$$
$$800 = 80 + 360(2)$$

Korzystając z tej logiki transmisja szeregowa implikuje, że naszą bazą jest liczba 256 (od 0 do 255 jest 256 wartości), a dowolną liczbę do wysłania można rozłożyć na modulo o tej podstawie i iloraz o tej samej podstawie:

$$N = N \bmod(256) + 256(quotient(N/256))$$

Na przykład,

$$7515 = 7515 \bmod(256) + 256(quotient(7515/256))$$
$$7515 = 91 + 256(29)$$

Krótko mówiąc, aby pracować z wartościami większymi niż 255, liczba musi zostać rozłożona na dwie części, jedną związaną z modulo tej liczby, a drugą z jej ilorazem:

$$7515 = \boxed{7515 \bmod(256)} + 256(\boxed{quotient(7515/256)})$$

Kolejnym pytaniem do rozwiązania jest sposób wysyłania liczb ujemnych. Odpowiedź brzmi, że istnieją dwie opcje, a każda z nich będzie zależeć od zakresu danych, który ma być użyty (na przykład zakres czujnika). Pierwszym z nich jest w zasadzie zmarnowanie jednego z 8 bitów do wysłania na wskazanie, że wspomniany bit ma wartość 0, jeśli liczba do wysłania jest dodatnia lub 0, a ma wartość 1, jeśli liczba jest ujemna. Jednak naszą bazą komunikacyjną nie będzie już 256, ale 128 (z pozostałych 7 bitów możemy utworzyć tylko 128 różnych liczb). Drugim jest wykonanie nasycenia i mapowania (tłumaczenia i skalowania) wartości (operacje opisane w poprzedniej sekcji). Podczas wykonywania nasycenia ustalamy limity dla naszego sygnału i te limity są wymagane do zdefiniowania minimalnych i maksymalnych wartości wejściowych w naszej funkcji mapowania. Tłumaczenie umożliwia nam przejście w kierunku pozytywnego odniesienia, na przykład przejście z zakresu wartości od [-100 100] do jednej z [0 200]. Skalowanie jest opcjonalne, jeśli wejście ma duże wartości w stosunku do wyjścia, na przykład przechodząc od [-1000 1000] do [0 200]. Zauważ, że funkcja mapowania ma już oba efekty. Efektem nadmiernego skalowania (przy dużych liczbach staje się to bardziej zauważalne) jest obecność liczb z częścią ułamkową. Teraz odpowiemy, jak z nimi pracować. Pierwszą rzeczą do zrobienia jest ustalenie pożądanego stopnia precyzji. Musimy ocenić, czy interesuje nas jedno, dwa lub więcej miejsc

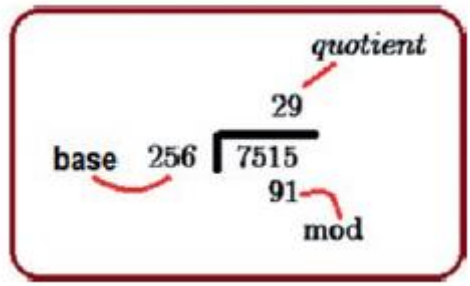
po przecinku. Na przykład, jeśli w naszym zakresie [0 200] interesuje nas praca z liczbami takimi jak 199,3 lub liczbami takimi jak 199,15. Aby komunikować się z tymi numerami, mamy też dwie podstawowe opcje. Pierwszym z nich jest oddzielenie części całkowitej od części ułamkowej i przesłanie ich jako dwóch różnych liczb. W przykładzie numeru 199.15 osobno zostanie wysłana 199, a następnie 15. Problem w tym podejściu polega na tym, że jeden z dwóch numerów może zgubić się po drodze. Inną opcją jest zastosowanie skalowania. W ten sposób, jeśli interesuje nas tylko jedno miejsce po przecinku, wystarczy pomnożyć nasz początkowy zakres przez 10, więc korzystając z podanych przykładów, 199,3 stanie się 1993, a 199,15 stanie się 1991. Zamiast tego, jeśli chcemy mieć dwa miejsca po przecinku miejsc, współczynnik wynosiłby 100, wysyłając odpowiednio 1930 i 19915. W ten sposób wysyłany jest przeskalowany numer. W tym celu wymagane jest, aby zarówno odbiornik, jak i nadajnik znali współczynnik skali. Uzyskanie wysokiego poziomu precyzji przy takim podejściu jest problemem, ponieważ całkiem możliwe, że liczba do wysłania musi zostać podzielona na dwie lub więcej części. Aplikacja jest prosta. Na przykład, jeśli chcemy mieć dwa miejsca po przecinku, zamiast mapować od [-1000 1000] do [0 200], powinniśmy to zrobić od [-1000 1000] do [0 20000]. Zwróć uwagę, że w nadawcy i odbiorniku można wykonać prawie każdą operację, w tym operacje zmiennoprzecinkowe lub podwójne. Ale transmisja danych dotyczy wyłącznie dodatnich wartości całkowitych. Z tego powodu musimy wprowadzić wyżej wymienione poprawki (istnieją interfejsy programistyczne, takie jak Arduino, które zawierają funkcje, które automatycznie wykonują te konwersje; z tego powodu wydaje się, że obsługują nie tylko liczby całkowite dodatnie). Na koniec, aby odpowiedzieć na pozostałe pytanie, jak gwarantowana jest komunikacja, wskażemy, w jaki sposób zaimplementowane są komponenty komunikacji UART. Podstawą tej procedury jest zrozumienie operacji binarnych, które są równoważne z tymi wcześniej przedstawionymi (modulo i iloraz) oraz niektórych innych binarnych operacji sprawdzania. Dzieje się tak, ponieważ wygodnie jest uprościć zadania przetwarzania danych na komputerze w języku, który zna procesor. Te operacje to wysyłanie, odbieranie, sprawdzanie i komunikowanie się.

Wysyłanie UART

Punktem wyjścia jest to równanie:

$$N = N \bmod(256) + 256(quotient(N/256))$$

W formacie dziesiętnym, operacje, aby uzyskać modulo i iloraz są uzyskiwane z podziału, jak pokazano na rysunku



W formacie binarnym iloraz można otrzymać z niecyklicznej operacji przesunięcia w prawo (zastępując liczby przesuwane zerami). Jest to stosowane do ostatnich 8 bitów binarnego odpowiednika liczby, która ma zostać podzielona. Ta operacja jest reprezentowana jako >> 8 w C ++ jako odniesienie dla innych języków programowania. Z drugiej strony, moduł uzyskuje się z operacją AND zaaplikowaną z

255 lub 1111 1111 w bazie binarnej. Jest to reprezentowane jako & w C ++ jako odniesienie dla innych języków programowania.

Na tym samym przykładzie:

7515=111001011011011 (bin)

11101 0101 1011(bin) >>8 = 0000 0000 11101(bin) = 29(dec)

Z drugiej strony:

11101 0101 1011 (bin)

& 00000 1111 1111 (bin)

= 00000 0101 1011 (bin) = 91 (dec)

W ten sposób numer (w tym przypadku 7515) jest wysyłany jako dwie grupy po 8 bitów:

0101 1011 (bin) = 91 (dec)

0001 1101(bin) = 29(dec)

Oczywiście przed wysłaniem tych numerów zostały one uzyskane za pomocą opisanych wcześniej operacji (>>8 i & 1111 1111).

Odbieranie UART

Odbiór polega na zastosowaniu następującego równania:

$$N = N \bmod(256) + 256(quotient(N/256))$$

Bezpośrednio w niektórych językach programowania lub za pomocą następującego odpowiednika binarnego: 8-bitowe niecykliczne przesunięcie w lewo i operacje OR (<< 8 i | w C ++):

Wartość binarna = Część binarna ilorazu << 8 | Część modułu binarnego

Korzystając z poprzedniego przykładu:

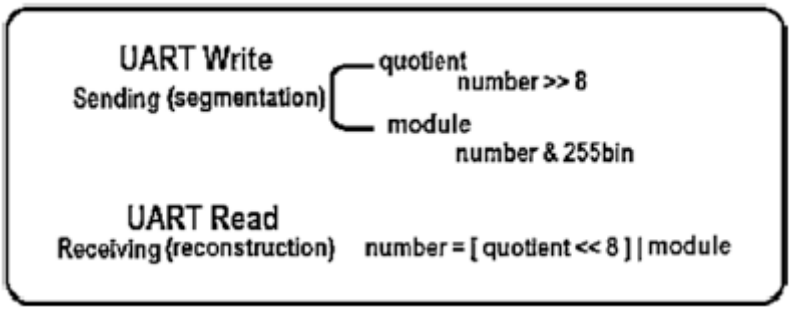
0001 1101(bin) = 29 (dec) << 8 = 0000 11101 0000 0000 = 7424 (dec)

Mając taki wynik, kontynuujemy operację OR. Zauważ, że jest więcej niż 8 bitów, ale pamiętaj, że jest to przetwarzane przez komputer odbierający po odebraniu danych. Nie dotyczy to kanału komunikacyjnego, który wymaga wiadomości 8-bitowych.

11101 0000 0000| 00000 0101 1011 (bin) = 91 (dec)

= 11101 0101 1011(bin) = 7515(dec)

Procesy czytania i pisania podsumowano na rysunku



Teraz wiesz, jak wysyłać i odbierać informacje. Jak jednak weryfikujesz dane? A skąd wiesz, czy potrafisz się komunikować? Odpowiedzią na to są następujące operacje.

Sprawdzanie UART

W takim przypadku błędna informacja jest odrzucana, najprostszym algorytmem jest suma kontrolna, a jego ulepszeniem jest suma kontrolna XOR. Proste wyjaśnienie odbywa się z podstawy dziesiętnej.

Załóżmy, że wysyłane są następujące dane:

val1Send=50

val2Sendj=30

Trzecią wartością do wysłania może być suma poprzednich ilości (suma części algorytmu sumy kontrolnej):

wartSend=wart1+wart2=80

Te same dane powinny dotrzeć do odbiorcy:

Val1Rec=50

Val2Rec=30

valSumRec=80

Następnie budowana jest część kontrolna algorytmu:

Check=Val1Rec+Val2Rec=80

W ten sposób Check i valSumRec są takie same i jest to prawidłowy zbiór danych. Załóżmy teraz, że odbiornik odczytuje błędną wartość:

Val1Rec=20

Val2Rec=30

valSumRec=80

Ponownie budujemy część kontrolną:

Check=Val1Rec+Val2Rec=50

W ten sposób Check różni się od valSumRec i gromadzenie danych jest odrzucane lub informacje są ponownie żądane.

Jak widać, algorytm sumy kontrolnej ma kilka ważnych problemów:

1. Suma może przekroczyć wartość 255 i wymagane będzie przesłanie liczby w postaci dwóch lub więcej grup danych.

2. Suma błędnych danych może skutkować poprawną kontrolą. Na przykład nadawca pisze

val1Send=50

val2Send=30

valsumSend=val1+val2=80

Odbiorca czyta

Val1Rec=60

Val2Rec=20

valSumRec=80

A ponieważ kontrola pozostaje równa 80, ten niepoprawny pakiet danych nie zostanie odrzucony.

3. Dane docierają poprawnie, ale suma jest błędna.

Nadawca pisze

val1Send=50

val2Send=30

valsumSend=val1+val2=80

A odbiorca czyta

Val1Rec=50

Val2Rec=30

valSumRec=10

Ponieważ wartość check=80 różni się od wartości valSumRec, informacje są odrzucane, mimo że dotarły poprawnie.

Aby zmniejszyć te problemy (przynajmniej problem przepełnienia), stosuje się binarne sumy kontrolne. Jednym z nich jest suma kontrolna XOR (operator ^ w C++). Zauważ, że pozostałe problemy są nadal obecne. Jeśli chcesz wysłać wartość sumy przekraczającą 255, na przykład 520, pierwszą rzeczą do zrobienia jest przekształcenie binarne:

1000001000 (bin) = 520 (dec)

Następnie dodaje się XOR element po elemencie:

(1)xor(0)xor(0)xor(0)xor(0)xor(0)xor(1)xor(0)xor(0)xor(0)=0

Tak więc dodatkowe dane do wysłania to po prostu 0.

Na przykład:

Val1Send=250

Val2Send=250

Val3Send=20

ValSum=520

Nie można wysłać wartości Valsum, ale zamiast tego wysyłana jest suma XOR jej składników binarnych:

XORSend=(1)xor(0)xor(0)xor(0)xor(0)xor(1)xor(0)xor(0)xor(0)=0

Odbiorca czyta

Val1Rec=250

Val2Rec=10

Val3Rec=20

XORRec=0

Następnie buduje się kotrolę:

Check=Val1Rec+Val2Rec+Val3Rec=280

Suma XOR składa się z binarnych składników czeku:

100011000 (bin) = 280 (dec)

XORCheck=(1)xor(0)xor(0)xor(0)xor(1)xor(1)xor(0)xor(0)xor(0)=1

Ponieważ wynik różni się od XORRec, dane są odrzucane lub ponownie żądane.

Zgoda na obrót

W takim przypadku przesyłane są dodatkowe informacje wskazujące, które urządzenia są odbiornikami, a które nadajnikami podczas procesu komunikacji. Algorytm polega na wysłaniu i ocenie wartości logicznej. Jest to posiadanie dwóch lub więcej dronów, które zawierają urządzenia komunikacyjne; wszystkie mają zmienną, która działa jak sygnalizacja świetlna lub flaga. Jedna z nich ma tę zmienną w stanie ON, a pozostałe w stanie OFF. Tylko ten, który zaczyna się od stanu ON, może pisać, podczas gdy inne czytają. Jak tylko skończy się ten, który zaczął się w stanie ON, zmienia swoją flagę lub semafor na OFF. Kiedy inni członkowie zespołu przeczytają, że flaga początkowego nadajnika zmienia się na WYŁ., inicjują sekwencję, aby określić, który z nich jako następny zmieni flagę na WŁ. Powtarza się to cyklicznie, na przemian lub tylko raz. Zasadniczo jest to uprzejma rozmowa, w której każdy uczestnik prosi lub obraca swoją kolejkę, aby się porozumieć, czekając na zakończenie rozmowy.

Ogólny algorytm UART

Ogólny algorytm komunikacji UART z wykorzystaniem wyżej wymienionych operacji jest następujący. (Podkreślono specyficzny wygląd każdej operacji; zauważ, że suma kontrolna jest operacją współdzieloną. Aby ulepszyć tę sekwencję, każdy proces może zależeć od timera, który przyspiesza lub zapewnia jego wykonanie lub po prostu w celu uniknięcia opóźnień.)

1. Sprawdzana jest flaga lub obrót urządzenia komunikacyjnego.

2. Jeśli bieżąca tura polega na pisaniu,

• Informacje do wysłania są podzielone na 8-bitowe segmenty. Informacje te są podzielone na dwie części: modulo i iloraz.

• Przeprowadzana jest suma wszystkich informacji do wysłania, az wyniku otrzymuje się sumę XOR jej składowych binarnych.

• Dostępność kanału jest weryfikowana.

• Informacja zostaje wysłana.

• Zmienne używane do pisania są czyszczone do późniejszego wykorzystania.

• Stan flagi zostaje zmieniony na odczyt.

3. Jeśli kolejka oznacza czytanie,

• Odebrane dane są odczytywane.

• Przeprowadzana jest kontrola XOR, a następnie weryfikowana z otrzymaną sumą.

• Jeśli się nie pokrywają, projektant określi procedurę do wykonania, która może obejmować od odrzucenia informacji do ponownego zażądania informacji.

• Jeśli pokrywają się, informacja jest rekonstruowana ze wskazanym przesunięciem bitowym i operacją OR.

• Zmienne wykorzystywane do odczytu są czyszczone do późniejszego wykorzystania.

• Stan flagi zostaje zmieniony na zapis.

4. Proces trwa w nieskończoność do momentu przerwania komunikacji przez użytkownika, kod lub zasilanie.

Teraz, gdy wiesz już o sterowaniu, przetwarzaniu sygnału i komunikacji UART , kolejne sekcje wskażą Ci sposoby kodowania drona. Najpierw pokażemy Ci dostępne środowiska do programowania Twojego samolotu.

Dostępne sposoby programowania drona

Obecnie istnieją dwa sposoby kodowania dronów: GUI i SDK.

GUI

GUI to uproszczony i wysoce wizualny interfejs zaprojektowany dla użytkownika z ograniczonymi lub zerowymi umiejętnościami programowania. W tym przypadku jest to graficzny interfejs użytkownika. Wiele zamkniętych architektur do programowania dronów ma jedną z nich. Te zamknięte architektury charakteryzują się tym, że umożliwiają użytkownikowi modyfikowanie podstawowych parametrów, takich jak ścieżka do przebycia, wzmocnienia wstępnie załadowanych kontrolerów oraz podstawowa kalibracja czujników i elementów wykonawczych. Bardziej specjalistyczne zadania są zamknięte dla użytkownika. Niektóre z najpopularniejszych GUI to Mission Planner i LibrePilot.

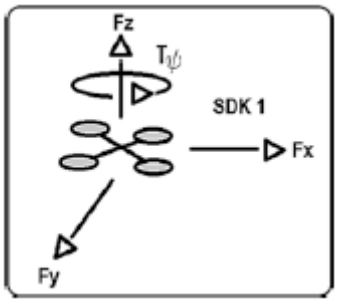
SDK

SDK to zestaw programistyczny. We wszystkich przypadkach użytkownik musi znać język programowania.

Poziom 1 pakietu SDK

W przypadku SDK Level 1 użytkownicy niewyspecjalizowani w sterowaniu dronem lub robotyce mogą zaprogramować drona do celów innych niż rekreacyjne. Architektura nie jest całkowicie otwarta i

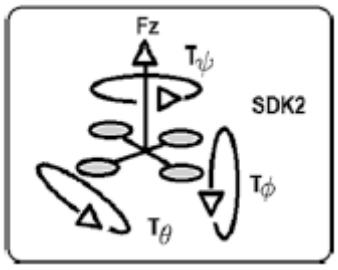
pozwala jedynie użytkownikowi zaprojektować niektóre kontrolery wysokiego poziomu w taki sposób, jakby pojazd był cząsteczką zdolną do poruszania się w osi X, Y, Z i obracania się wokół własnej osi Z. Zobacz rysunek



Jednym z najbardziej znanych przykładów jest SDK Dronekit oparty na Pythonie. W tego typu SDK użytkownik może modyfikować tylko żądane odniesienia X, Y, Z i odchylenia lub ich prędkości. Nic nie da się zrobić z siłą ciągu i momentami obrotowymi drona i oczywiście z macierzą alokacji.

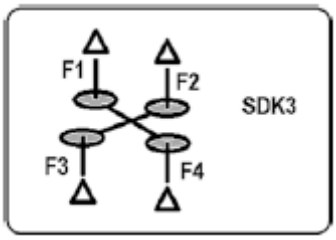
Poziom 2 pakietu SDK

Architektura w SDK Level 2 jest nieco bardziej otwarta i umożliwia sterowanie pojazdem na poziomie siły i momentu. Oznacza to, że pozwala użytkownikom, którzy są zaznajomieni ze sterowaniem dronami i robotyką, na włączenie własnych algorytmów i kontrolerów do wstępnie zaprojektowanych pojazdów, które są kompatybilne z SDK. Jednym z najbardziej znanych przykładów jest Parrot Bebop-autonomy SDK oparty na C++, a także tryb lotu bibliotek PX4. W tego typu SDK użytkownik może jedynie modyfikować siłę ciągu i momenty obrotowe drona. Nic nie można zrobić z macierzą alokacji. Ponieważ siła ciągu i momenty obrotowe drona są funkcją odniesienia przestrzennego (X, Y, Z i odchylenia), ten pakiet SDK ma wyższy poziom niż pakiety SDK typu 1.



SDK poziom 3

SDK Level 3 jest całkowicie otwarty i pozwala użytkownikowi sterować pojazdem na poziomie silnika. Umożliwia także programowanie własnych protokołów komunikacyjnych, włączanie własnych czujników, monitorowanie interesujących ich parametrów i nie tylko. Krótko mówiąc, pozwala użytkownikowi zaprojektować własne pojazdy, nawet jeśli pojazdy te nie mają precedensu. Najbardziej znanymi przykładami są biblioteki Ardupilot oraz biblioteki PX4 oparte na C++. W tego typu SDK użytkownik musi podać macierz alokacji, która jest funkcją momentów ciągu i sił drona, które z kolei są funkcją żądanej pozycji i wartości orientacji. Dlatego ten rodzaj SDK ma wyższy poziom niż zestawy SDK typu 2 i 1.



Nauczyłeś się, gdzie kodować, ale co z kodowaniem? To jest wprowadzone w następnej sekcji.

Niektóre przydatne polecenia dostępne w większości zestawów SDK

Poniżej znajdują się najczęściej używane polecenia do programowania dronów powietrznych i ogólnie każdego innego rodzaju pojazdu. Można je znaleźć w trzech najczęściej używanych pakietach SDK: Ardupilot, PX4 i Dronekit. Polecenia te są przedstawione w sposób tematyczny i są kompatybilne odpowiednio z C++, C++ dla ROS i Pythonem (chociaż te SDK mają warianty dla innych języków programowania, systemów operacyjnych i architektur obliczeniowych). W odniesieniu do wyżej wymienionych bibliotek należy wskazać, że jedyną, która pozwala użytkownikowi na włączanie zadań czasu rzeczywistego jest Ardupilot, najprostszym w obsłudze jest Dronekit, a ta, która ma wiele kompatybilności, ponieważ bazuje na ROS to PX4, co jako wadę ma ograniczoną i mało zrozumiałą dokumentację (przynajmniej do tej pory).

Streszczenie

Poznałeś zadania drona, a także podstawowe rodzaje przetwarzania sygnału, które są wymagane do jego użycia. Wyjaśnione przetwarzanie sygnału obejmowało filtrowanie sygnału, nasycenie sygnału, mapowanie sygnału, rzutowanie danych oraz normalizację nadmiarowości i osobliwości. Wszystkie mogą być używane z czujnikami, siłownikami oraz ze stopniami komunikacji lub sterowania. W przypadku zadania komunikacyjnego poznałeś zwykłe części UART wysokiego poziomu. Na koniec dowiedziałeś się o czterech sposobach programowania drona, w tym o tematycznej liście poleceń drona z podejściem opartym na trzech najczęściej używanych pakietach SDK.