QUADROCOPTER

(sterowanie, stabilizacja, autopilot)

Projekt przejściowy 2014/15
specjalności Robotyka
na Wydziale Elektroniki

Politechnika Wrocławska 2014

Autorzy

Marcin Ciopcia
Michał Drwięga
Daniel Gut
Alicja Jurasik
Agata Leś
Kacper Nowosad
Piotr Semberecki
Hanna Sienkiewicz
Mateusz Stachowski
Paweł Urbaniak
Krzysztof Zawada

Skład raportu wykonano w systemie LaTeX

Spis treści

Część I

Wstęp

1. Wstęp, Piotrek

1.1. Quadrocopter, Krzysiek

1.1.1. Własności obiektu, Daniel

1.2. Dostępne rozwiązania, Kacper

1.3. Cel pracy, Hania

Część II

Zadania

2. Stabilizacja w punkcie - przygotowania, Daniel

2.1. Interfejs do istniejącego oprogramowania i sprzętu, Kacper

2.1.1. Flying Machine Arena, Michał

2.1.2. Rozpoznanie protokołu quadrocoptera, Daniel

Przygotowanie specyfikacji interfejsu, Piotr

Wybór interfejsu, Piotr

2.1.3. Rozpoznanie możliwości platformy, Daniel

2.2. Czujniki odległości, Alicja

2.2.1. Dobór czujników odległości, Hania

Specyfikacja potrzeb, Hania

Dostępne rozwiązania, Alicja

Dobór czujników, Hania

2.2.2. Rozmieszczenie czujników odległości, Paweł

Przegląd rozwiązań, Paweł

Propozycja rozmieszczenia, Paweł

2.3. Algorytmy rozpoznawania przesunięcia, Kacper

2.3.1. Algorytm Lucas—Kanade, Piotrek

2.3.2. Zmodyfikowany PTAM, Kacper

2.3.3. Algorytm SVO, Hania

3. Stabilizacja w punkcie - realizacja, Krzysiek

3.1. Ahsokam Marcin

3.2. Uruchomienie platformy latającej, Marcin

3.2.1. Uzbrajanie silników, Marcin

3.3. Uruchomienie kamery 3D, Michał

3.4. Uruchomienie komunikacji, Michał

Komunikacja ze sterownikiem quadrocoptera odbywa się z wykorzystaniem protokołu **MAVLink**, który został zaprojektowany do wymiany informacji między podsystemami robota oraz do komunikacji ze stacją kontroli (**GCS**). Specyfikację protokołu można znaleźć pod następującym odnośnikiem: https://pixhawk.ethz.ch/mavlink.

Dla dwóch wersji ROS'a: hydro i indigo został zaimplementowany protokół MAVLink, w postaci pakietu **mavros**. Pakiet ten umożliwia z poziomu ROS'a dostęp do parametrów sterownika, oraz pozwala wysyłać do niego komendy sterujące. Ponadto, pakiet ten umożliwia dodawanie własnych pluginów. Dokładny opis jest dostępny pod następującym odnośnikiem: http://wiki.ros.org/mavros.

Poniżej wymienione zostały wybrane funkcjonalności pakietu **mavros** w przypadku załadowania domyślnego zestawu pluginów.

- Publikowanie topic'ów zawierających dane z IMU (plugin **imu_pub**), między innymi:
  - orientacja obliczona przez sterownik (*~imu/data*),
  - wartości zmierzone przez IMU (*~imu/data_raw*),
  - odczyty magnetometru (*~imu/mag*),
  - temperatura (*~imu/temperature*),
  - ciśnienie powietrza (*~imu/FluidPressure*).
- Subskrybowanie topic'ów z komendami lotu:
  - zadane prędkości (*~setpoint/cmd_vel*),
  - zadane przyspieszenia (*~setpoint/accel*),
  - zadane prędkości kątowe (*~setpoint/att_vel*),
  - zadana poza (*~setpoint/attitude*).

Warto również wspomnieć o dwóch podobnych pakietach: **roscopter** i **mavlink_ros**. Pierwszy z nich występuje dla ROS'a w wersjach: groovy i hydro. Jego funkcjonalno-

ści są jednak ograniczone w porównaniu do pakietu **mavros**. Natomiast drugi pakiet, **mavlink_ros** jest przestarzały, a twórcy zalecają zastąpienie go pakietem **mavros**.

3.4.1. Komunikacja ROS—kamera, Mateusz

3.4.2. Komunikacja ROS—MAVLINK, Mateusz

3.4.3. Komunikacja ROS—GroundStation, Krzysiek

3.4.4. Komunikacja ROS—czujniki, Daniel

3.4.5. Komunikacja MAVLINK—PixHawk, Mateusz

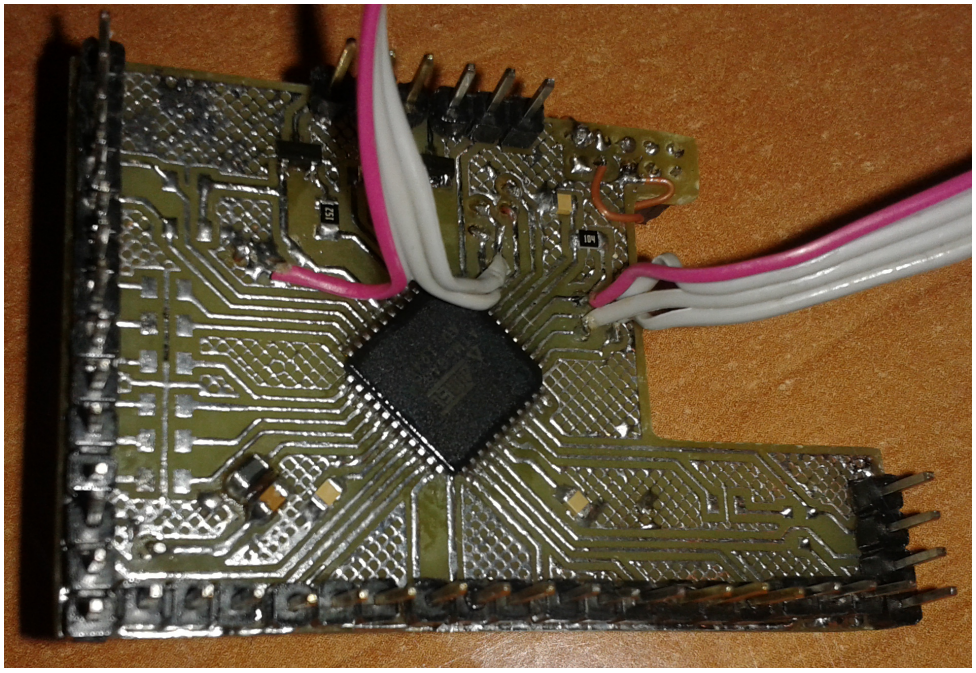
3.5. Czujniki odległości

3.5.1. Płytka PCB, Alicja

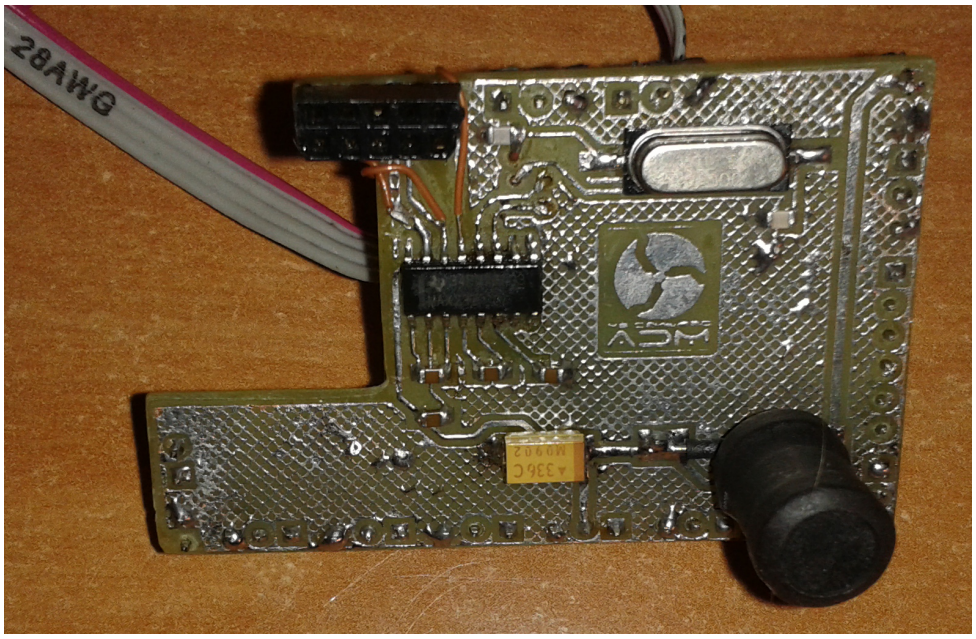
Specyfikacja potrzeb, Marcin

Wykonanie i uruchomienie płytki PCB, Michał

Na rysunku 3.1 przedstawiono wykonany interfejs czujników. Widoczne złącza umożliwiają podłączenie zasilania, czujników odległości Sharp, stopnia mocy oświetlenia, czy sygnału z aparatury RC.



(a) widok z góry



(b) widok z dołu

Rysunek 3.1. Przedstawienie wykonanego interfejsu

3.5.2. Oprogramowanie interfejsu, Michał

Urządzenie spełnia rolę interfejsu sterującego wybranymi podzespołami quadrocoptera oraz jednostki pomiarowej, mierzącej odległości robota od przeszkód. Wyposażone zostało w mikrokontroler Atmel Atmega32. Aktualnie zaimplementowane funkcjonalności są następujące:

— cykliczne wykonywanie pomiarów napięć czujników odległości (porty ADC1-ADC7),
— pomiar odległości poprzez zliczanie czasu trwania impulsów (PW) generowanych przez sonary:

  - przerwania zewnętrzne,
  - timer zliczający czas.
- wysyłanie pomiarów do komputera nadrzędnego (UART),
- sterowanie oświetleniem quadrocoptera,
- dekodowanie sygnału PPM z zadajnika RC,
- obsługa Pixhawk Failsafe.

Oprogramowanie zostało utworzone w języku C, z wykorzystaniem środowiska Atmel Studio 6, które jest nieodpłatnie udostępniane przez producenta mikrokontrolerów. Szczegółowa dokumentacja oprogramowania została utworzona z wykorzystaniem generatora Doxygen.

Protokół komunikacji z komputerem nadrzędnym

W celu umożliwienia komunikacji urządzenia z komputerem nadrzędnym konieczne było zaprojektowanie odpowiedniego protokołu komunikacyjnego. Warto zaznaczyć, że do omawianej komunikacji wykorzystuje się interfejs szeregowy UART.

Główną ideą zaprojektowanego protokołu komunikacji jednostki pomiarowej z komputerem jest wykorzystanie architektury klient-serwer. Rolę klienta pełni aplikacja uruchomiona na komputerze nadrzędnym, która wysyła odpowiednio zakodowane rozkazy. Natomiast interfejs sensorów dekoduje i realizuje otrzymywane rozkazy. W dalszej części przedstawiono szczegółowy opis protokołu.

Weryfikacja poprawności ramek

Suma kontrolna

W celu weryfikowania poprawności transmisji zaimplementowano obliczanie prostej sumy kontrolnej. Algorytm zlicza liczbę bitów o stanie wysokim w ciągu wysłanych bajtów. Metoda ta nie zapewnia dużej odporności na zakłócenia, ponieważ jest duże prawdopodobieństwo, że suma bitów o wysokim stanie niepoprawnego ciągu bajtów będzie równa sumie poprawnego. Jednak zaletą takiej sumy kontrolnej jest niewielka złożoność obliczeniowa.

Potwierdzanie rozkazów

Zwiększenie niezawodności transmisji zrealizowano przez implementację potwierdzania rozkazów. Potwierdzanie dotyczy tylko rozkazów, dla których nie jest oczekiwana transmisja zwrotna, czyli między innymi rozkazów konfiguracyjnych. W przypadku tej grupy oczekuje się potwierdzenia w postaci krótkiej ramki składającej się z kodu rozkazu oraz sumy kontrolnej.

Opis protokołu

W pierwszym odebranym bajcie danych przez jednostkę pomiarową, dwa najstarsze bity są bitami kontrolnymi o wartościach wysokiego stanu logicznego (11). Kolejne 6 bitów określa rozkaz, który należy wykonać. W ten sposób można zakodować do 64 różnych rozkazów, co wydaje się być wartością zupełnie wystarczającą. W przypadku, gdy nie ma dodatkowych danych dla jednostki pomiarowej, drugi bajt zawiera sumę kontrolną. Ogólna postać ramki danych została przedstawiona na rysunku 3.2.

Nr bajtu	1	2....n-1	n
Funkcja	2 bity startowe i kod rozkazu	Dodatkowe dane	Suma kontrolna

Rysunek 3.2. Ogólna postać ramki danych

Zaimplementowany protokół zakłada przesyłanie bardziej znaczących bajtów przed mniej znaczącymi (Big endian). Każdy dodatkowo przesyłany ciąg danych jest weryfikowany sumą kontrolną, której algorytm został przedstawiony w poprzednim podrozdziale. Rozkazy podzielono na cztery części w zależności od wartości pierwszych czterech bitów:

0xC (1100) żądania przesłania informacji zwrotnej,
0xD (1101) rozkazy konfiguracyjne,
0xE (1110) pozostałe rozkazy,
0xF (1111) pozostałe rozkazy.

W kolejnych podrozdziałach przedstawiono opisy poszczególnych grup rozkazów. Rozkazy opisane zostały przez podanie odpowiadających im wartości bajtów i krótkich opisów działania.

Żądania przesłania informacji zwrotnej - 0xC

1100 0000 - 0xC0

Weryfikacja poprawności działania transmisji. Oczekuje się odpowiedzi w postaci 0101 0101 - 0x55.

1100 1001 - 0xC1

Polecenie wysłania zmierzonych wartości z sensorów odległości oraz odczytanych wartości z sygnału PPM. Jednostką pomiarów sensorów Sharp są mV. W przypadku sonarów odległość podawana jest w cm. Wartości sygnału PPM dla kolejnych kanałów podawane jest w μs Kolejność pomiarów w ramce określona została następująco:

— 2 bajty - napięcie w mV odczytane z sensora Sharp 1,
— 2 bajty - napięcie w mV odczytane z sensora Sharp 2,
— 2 bajty - napięcie w mV odczytane z sensora Sharp 3,
— 2 bajty - napięcie w mV odczytane z sensora Sharp 4,
— 2 bajty - napięcie w mV odczytane z sensora Sharp 5,
— 2 bajty - napięcie w mV odczytane z sensora Sharp 6,
— 2 bajty - napięcie w mV odczytane z sensora Sharp 7.
— 2 bajty - odległość zmierzona sonarem nr 1,
— 2 bajty - odległość zmierzona sonarem nr 2,
— 2 bajty - szerokość impulsu w μs w sygnale PPM dla kanału nr1,
— 2 bajty - szerokość impulsu w μs w sygnale PPM dla kanału nr2,
— 2 bajty - szerokość impulsu w μs w sygnale PPM dla kanału nr3,
— 2 bajty - szerokość impulsu w μs w sygnale PPM dla kanału nr4,
— 2 bajty - szerokość impulsu w μs w sygnale PPM dla kanału nr5,
— 2 bajty - szerokość impulsu w μs w sygnale PPM dla kanału nr6,
— 2 bajty - szerokość impulsu w μs w sygnale PPM dla kanału nr7,
— 2 bajty - szerokość impulsu w μs w sygnale PPM dla kanału nr8,

Rozkazy konfiguracyjne - 0xD

1101 0000 - 0xD0
Polecenie uruchomienia oświetlenia quadrocoptera.

1101 0001 - 0xD1
Polecenie wyłączenia oświetlenia quadrocoptera.

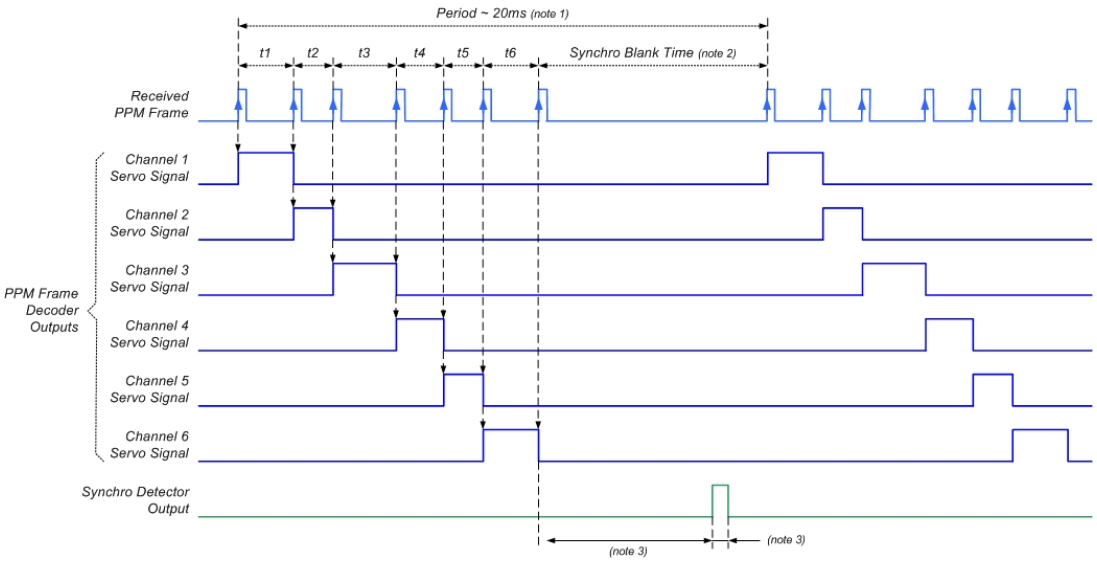
1101 0010 - 0xD2
Włączenie Pixhawk Failsafe (ustawienie stanu wysokiego).

1101 0011 - 0xD3
Wyłączenie Pixhawk Failsafe (ustawienie stanu niskiego).

3.5.3. Dekodowanie sygnału PPM, Michał

W celu umożliwienia ręcznego sterowania quadrocopterem przez operatora należało zrealizować dekodowanie sygnału PPM odbieranego przez aparaturę. Sygnał ten podłączony jest do wykonanego interfejsu i dekodowany przez umieszczony w nim mikrokontroler ATmega32.

PPM (Pulse Position Modulation) jest sygnałem, w którym informacje kodowane są z wykorzystaniem modulacji położenia impulsów. Umożliwia przesyłanie wartości w kilku kanałach (w przypadku dostępnej aparatury jest to 8 kanałów). Postać takiego sygnału przedstawiona została na rysunku 3.5.3.



Po zdekodowaniu sygnału PPM, wartości o poszczególnych kanałach wysyłane są do komputera nadrzędnego *nano6060*, gdzie przetwarzane są w środowisku *ROS* i następnie wysyłane do sterownika quadrocoptera Pixhawk'a. Takie podejście umożliwia ręczne sterowanie, ale także realizację korekcji sterowania z wykorzystaniem modułu antykolizyjnego.

Realizacja dekodowania sygnału PPM na mikrokontrolerze

Na potrzeby dekodowania sygnału PPM wykorzystano przerwanie zewnętrzne generowane w zależności od rodzaju zbocza sygnału. Rozpoznawanie początku ramki zrealizowane zostało przez badanie przekroczenia maksymalnego odstępu między impulsami, które

w przypadku zwracanych wartości kanałów wynosi 2ms. Poniżej przedstawiono funkcję realizującą obsługę przerwania generowanego w przypadku zmiany stanu pinu do którego podłączono sygnał PPM.

Wydruk 3.1. Obsługa przerwania realizującego dekodowanie PPM

```
ISR (INT2_vect)
{
  cli();
  if (MCUCR & _BV(ISC2)) // rising edge interrupt
  {
    TCNT2 = 0; // Timer2 reset
    timeCntPPM = 0;
    MCUCR &= ~_BV(ISC2); // change to falling edge detect
  }
  else // falling edge interrupt
  {
    if ((timeCntPPM + TCNT2) > MAX_PPM_WIDTH) // new frame
    {
      channelPPMcnt = 0;
    }
    else // measurement for PPM channel
    {
      channelPPM[channelPPMcnt] = timeCntPPM + TCNT2;
      if (channelPPMcnt < 7) // increase channel counter
      {
        channelPPMcnt++;
      }
    }
  }
  GIFR |= (1 << INTF1); // interrupt flag clear
  sei();
}
```

Komunikacja ROS-czujniki, Daniel

3.5.4. Montaż czujników na platformie docelowej, Mateusz

3.6. Implementacja algorytmów przesunięcia, Kacper

3.6.1. Algorytm Lucas—Kanade, Piotrek

3.6.2. Zmodyfikowany PTAM, Kacper

3.6.3. Algorytm SVO, Hania

3.7. Stabilizacja w poziomie, Alicja

3.7.1. Fuzja sensoryczna, Alicja

Czujniki wykorzystywane w fuzji, Alicja

Dostępne rozwiązania, Alicja

Realizacja, Hania

3.8. Stabilizacja w pionie, Paweł

3.8.1. Fuzja sensoryczna, Michał

Algorytm stabilizacji wysokości
Dane wejściowe:
— d_1, d_2 – pomiary z dwóch czujników odległości: górnego i dolnego,
— q – orientacja quadrocoptera względem zewnętrznego układu odniesienia, w postaci kwaternionu (z Pixhawka),
Dane wyjściowe:
— F_z – siła w osi z quadrocoptera.

Kroki algorytmu
1. Obliczenie rzeczywistych odległości od podłoża i sufitu na podstawie odczytów sensorów oraz orientacji quadrocoptera względem zewnętrznego układu odniesienia.
2. Fuzja pomiarów – wybranie do stabilizacji odległości, która wolniej się zmienia (mniejsza pochodna). Celem jest uniezależnienie wysokości od niewielkich przeszkód.
3. Regulator PID:
   — wejście: wybrana wysokość h,
   — wyjście: sterowanie w postaci siły w osi Z.

Czujniki wykorzystywane w fuzji, Paweł

Realizacja, Michał

3.9. Algorytm stabilizacji w punkcie

4. Stabilizacja w pomieszczeniu

4.1. Sterownik nadrzędny, Marcin

4.1.1. Zdarzeniowy sterownik antykolizyjny, Paweł

Analiza wykonalności, Paweł

Specyfikacja, Paweł

4.1.2. Lot wzdłuż ściany, Krzysiek

Analiza wykonalności, Mateusz

Specyfikacja, Mateusz

Część III

Testy

5. Testy stabilizacji

5.1. Testy integracyjne na platformie docelowej

5.1.1. Testy modułu ROS

5.1.2. Testy modułu czujników

Część IV

Zakończenie

6. Podsumowanie

7. Dodatki

8. Bibliografia