



A G H

AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

**WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI,
INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ**

Projekt dyplomowy

Sprzętowo-programowy system wizyjny wspomagający autonomiczne lądowanie drona.

Hardware-software vision system supporting the autonomous landing of a drone.

Autor: Jakub Kłosiński
Kierunek studiów: Automatyka i Robotyka
Opiekun pracy: dr inż. Tomasz Kryjak

Kraków, 2020

Serdecznie dziękuję . . .

Spis treści

1. Wprowadzenie	7
1.1. Cele pracy	7
1.2. Zawartość pracy.....	8
2. Metody przetwarzania obrazu i sposoby sterowania w procesie autonomicznego lądowania drona	9
3. Sprzęt wykorzystany w projekcie	13
3.1. Platforma obliczeniowa	13
3.2. Moduł rejestrujący obraz	14
3.3. Platforma statku powietrznego	16
3.4. Autopilot.....	16
3.5. Laserowy czujnik wysokości	18
4. Zaimplementowany system wspomagający autonomiczne lądowanie drona.....	19
4.1. Implementacja i ewaluacja modelu programowego	19
4.1.1. Akwizycja ramek na kartę SD	20
4.2. Implementacja sprzętowo-programowa.....	20
4.2.1. Integracja kamery i płytki	22
4.2.2. Podłączenie laserowego czujnika wysokości	23
4.2.3. Konwersja z przestrzeni barw RGB do YCbCr	23
4.2.4. Binaryzacja.....	23
4.2.5. Mediana.....	23
4.2.6. Erozja i dylatacja.....	25
4.2.7. Prostokąt otaczający i pole powierzchni	25
4.2.8. Indeksacja.....	26
4.2.9. Środek ciężkości	27
4.2.10. Algorytm sterowania.....	29
4.2.11. Integracja płytki z autopilotem.....	29
5. Testy i ocena wyników	31

5.1.	Wybór znacznika	31
5.1.1.	Kształt	31
5.1.2.	Kolor	31
5.2.	Test reakcji autopilota na zadawane komendy	35
5.3.	Test sterowania	35
6.	Podsumowanie i kierunki dalszych prac.....	37
A.	Dodatek A	39

1. Wprowadzenie

Ostatnie lata pokazały wszechstronność i przydatność dronów w wielu dziedzinach przemysłu. Według raportu Skyward z 2018 roku, 1 na 10 przebadanych firm w Stanach Zjednoczonych używała bezzałogowych statków powietrznych. Aż 88 procent z nich odczuła pozytywne strony rozpoczęcia korzystania z nich w przeciągu roku lub krócej. Do najczęściej wymienianych zalet dronów należy zdobywanie większej ilości informacji, bardziej efektywna praca oraz oszczędzanie czasu. Udział dronów w rynku będzie się stale powiększał, gdyż 3 na 4 przedsiębiorców planuje zwiększać wydatki przeznaczone na operacje wykonywane przez drony [1]. O planach częstszego stosowania dronów może również świadczyć fakt wprowadzania regulacji prawnych - konieczności wyposażania dronów w nadajnik numeru identyfikacyjnego [2]. Odpowiednia regulacja zwiększy bezpieczeństwo lotów i może otworzyć drogę do masowego wykorzystania dronów.

Ważnym kierunkiem rozwoju bezzałogowych statków powietrznych jest ich autonomizacja, czyli przystosowanie do lotów bez nadzoru człowieka. Oznacza to wyposażenie dronów w systemy monitorowania otoczenia i implementację algorytmów sterowania działających w oparciu o zebrane informacje. Kluczową fazą autonomicznego lotu drona jest lądowanie. Bliskość ziemi wymaga dokładnego sterowania ruchem statku powietrznego. W przypadku, gdy dostarczane dane pochodzą z systemu wizyjnego, decydujące jest szybkie przetwarzanie obrazów. Podejście sekwencyjne, oparte o mikrokontrolery i procesory sygnałowe, często okazuje się nieskuteczne. Przekroczone zostają możliwości obliczeniowe takich układów, uniemożliwiając dodawanie kolejnych funkcjonalności związanych z autonomizacją. Układy FPGA (ang. *Field Programmable Gate Arrays*) są preferowaną platformą obliczeniową do realizacji zadań przetwarzania strumienia wideo. Oferują możliwość zrównoleglenia obliczeń oraz niewielkie opóźnienie przetwarzania.

1.1. Cele pracy

Celem pracy było stworzenie sprzętowo-programowego systemu wizyjnego, będącego komponentem systemu autonomicznego lądowania drona. Pierwszym krokiem prac było przeprowadzenie analizy literatury naukowej - głównie dotyczącej detekcji i śledzenia oznaczonego lądowiska.

W drugim etapie należało wykonać model programowy algorytmu, który pozwala na wykrycie i podążanie w kierunku lądowiska. Wybór jego oznaczenia był również częścią prac. Założeniem było wyposażenie drona w kamerę o osi optycznej skierowanej prostopadle do podłoża i wykonanie lądowania

na statycznej platformie. Należało również ocenić możliwość wykonania lądowania na ruchomym celu. Ponadto należało zintegrować z układem komponent odpowiedzialny za pomiar wysokości - Garmin Li-dar Lite v3.

W trzecim etapie należało wspomniany system podzielić na część sprzętową i programową oraz zaimplementować w układzie Zynq SoC na karcie ZYBO Z7-20. Wejściem powinieć być obraz z kamery PCAM oraz informacja z wysokościomierza, a wyjściem sterowanie dla drona - regulacja położenia względem lądowiska oraz wysokości.

Ostatnim etapem prac była próba integracji wykonanego podsystemu ze sterownikiem drona i wykonania lądowania w sposób autonomiczny. Testy należało przeprowadzić przy zachowaniu zasad bezpieczeństwa - kluczowe było upewnienie się co do niezawodności działania systemu.

1.2. Zawartość pracy

W rozdziale drugim znajduje się przegląd prac naukowych dotyczących problematyki autonomicznego lądowania drona. Rozdział trzeci opisuje sprzęt wykorzystany podczas realizacji projektu. Część trzecia przedstawia opis modelu programowego systemu wizyjnego wraz z uzasadnieniem wyboru poszczególnych modułów. Następnie skoncentrowano się na opisie implementacji całego systemu, począwszy od odbioru sygnału wizyjnego, a skończywszy na wysyłaniu sygnałów sterujących dronem. Rozdział czwarty poświęcono testom: wyboru znacznika znajdującego się na lądowisku, komunikacji z autopilotem oraz ręcznego sterowania na podstawie informacji z systemu. W ostatnim rozdziale znajduje się podsumowanie prac oraz przedstawienie możliwości rozwoju projektu.

2. Metody przetwarzania obrazu i sposoby sterowania w procesie autonomicznego lądowania drona

Wykonywanie różnorodnych scenariuszy misji przez drona wymaga osiągnięcia określonego poziomu autonomii. Bezzałogowy statek powietrzny musi być w stanie wystartować, nawigować oraz lądować bez bezpośredniego udziału człowieka. Przemieszczanie drona pomiędzy różnymi punktami nawigacyjnymi przebiega bezproblemowo, kiedy dostępny jest sygnał GPS. Kłopotu nie sprawia również start wykonywany bez pomocy operatora. Największe wyzwanie stanowi autonomiczne lądowanie drona. Platforma obliczeniowa zainstalowana na statku powietrznym musi być w stanie wykryć lądowisko oraz wysłać odpowiednie sygnały sterujące. Efektem działania algorytmu sterującego musi być zajęcie dogodnej pozycji do obniżenia lotu i ostatecznie przyziemienie w określonym punkcie.

Zadania autonomicznego lądowania drona można podzielić na dwie kategorie:

- lądowanie na platformie pozostającej w bezruchu względem ziemi,
- lądowanie na poruszającej się platformie.

W przypadku lądowania na oznaczonym stacjonarnym lądowisku, głównym problemem jest skuteczna detekcja znacznika. Do realizacji tego zadania niezbędne jest wykonanie szeregu kroków. Przykładowe rozwiązanie z pracy [3] obejmuje:

- zamianę obrazu kolorowego na obraz w skali szarości,
- binaryzację ze stałym progiem,
- indeksację,
- odrzucenie obiektów o liczbie pikseli mniejszej niż zadany próg,
- identyfikację znacznika.

Wartości progów binaryzacji oraz odrzucenia małych obiektów zostały dobrane eksperymentalnie. Aby dodatkowo wyeliminować biały szum oraz zakłócenia typu „sól i pieprz” w pracy [4] wprowadzono operację mediany z oknem 3x3 na obrazie w skali szarości. Dodatkowo wprowadzono ograniczenie na maksymalny rozmiar obiektu. W algorytmie przedstawionym w pracy [5] binaryzacji dokonano w przestrzeni barw HSV (ang. *Hue Saturation Value*), na podstawie składowych S i V. Pozwoliło to na przeprowadzenie segmentacji niezależnej od koloru. Przed rozpoznawaniem kształtów, na obrazie binarnym



(a) Znacznik użyty w pracy [3]

(b) Znacznik użyty w pracy [6]

(c) Znacznik użyty w pracy [7]

Rys. 2.1. Porównanie znaczników użytych w pracach [3],[6] i [7]

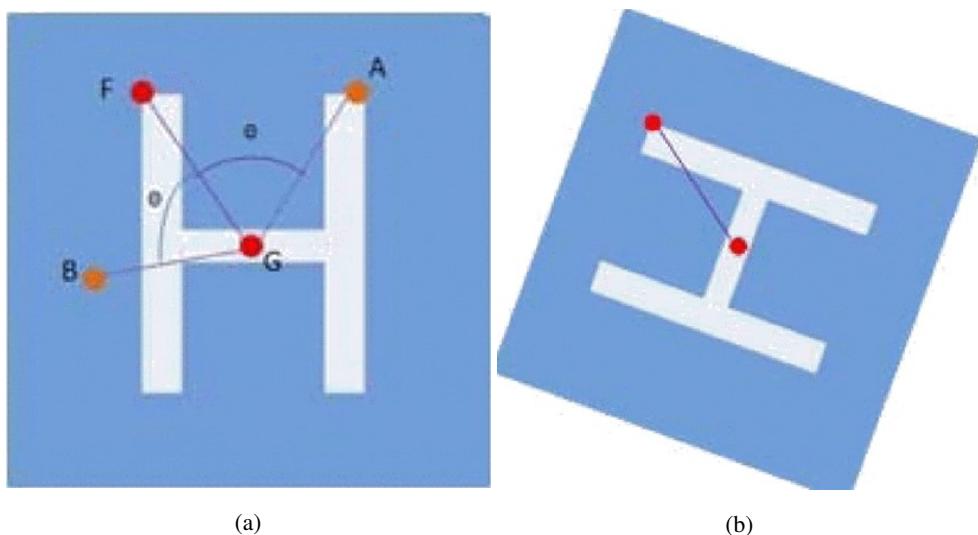
dokonano kolejno erozji, detekcji krawędzi oraz dylatacji. Wykonanie erozji pozwoliło na eliminację z obrazu małych grup pikseli.

Istotnym elementem planowania autonomicznego lądowania drona jest wybór znacznika lądowiska. W pracy [6] przedstawiono marker złożony z trzech figur: kwadratu, koła i krzyża. Znacznik pokazano na Rys. 2.1b. Zaimplementowany został algorytm detekcji lądowiska polegający na kolejnym wykrywaniu wymienionych figur, co pozwoliło na coraz lepsze określenie położenia celu.

Prostsze rozwiązanie zostało pokazane w pracy[7], gdzie znacznik ma kształt litery H (rysunek 2.1c) Odległość drona od lądowiska obliczana jest na podstawie liczby pikseli dzielących środek ciężkości znacznika od środka obrazu. Opisano tam również metodę wyznaczania orientacji drona względem takiego markera. Polega ona na znalezieniu piksela najbardziej odległego od środka ciężkości, a następnie wykreśleniu linii przechodzącej przez te dwa punkty. Otrzymana linia nie określa jednak pozycji znacznika w sposób jednoznaczny (Rys. 2.2). Do określenia, czy zachodzi przypadek przedstawiony na Rys. 2.2a, czy też 2.2b, obliczono kąt θ pomiędzy dwoma narożnymi punktami markera. Kąt obliczany jest na podstawie zapisanego obrazu znacznika. Następnie wykorzystując znalezioną wcześniej linię, obliczane są współrzędne tych dwóch punktów (punkty A i B na Rys. 2.2a). Kolejną czynnością jest sprawdzenie, który z punktów leży bliżej obszaru markera. Jeśli jest to punkt A - zachodzi przypadek z Rys. 2.2a, jeśli B - Rys. 2.2b.

Inny znacznik został użyty w pracy [3]. Składa się on z czterech pierścieni na czarnym tle (Rys. 2.1a) Każdy pierścień posiada unikalny stosunek promienia wewnętrznego do zewnętrznego i stanowi oddzielny obiekt dla systemu wizyjnego. Obiekty, które nie mają dokładnie jednej dziury wewnętrznej, są pomijane. Każdy z pozostałych jest ostatecznie identyfikowany za pomocą współczynnika kształtu. Zaletą takiego znacznika jest możliwość dołożenia kolejnych, większych pierścieni, jeśli lądowisko ma być widoczne z większej wysokości.

W procesie autonomicznego lądowania drona, na podstawie wyznaczonej odległości od celu, do bezzałogowego statku powietrznego wysyłane są sygnały sterujące. W pracy [8] wykorzystano trzy regulatory PID, do kontroli prędkości drona. Dzięki nim równocześnie minimalizowana jest każda z trzech



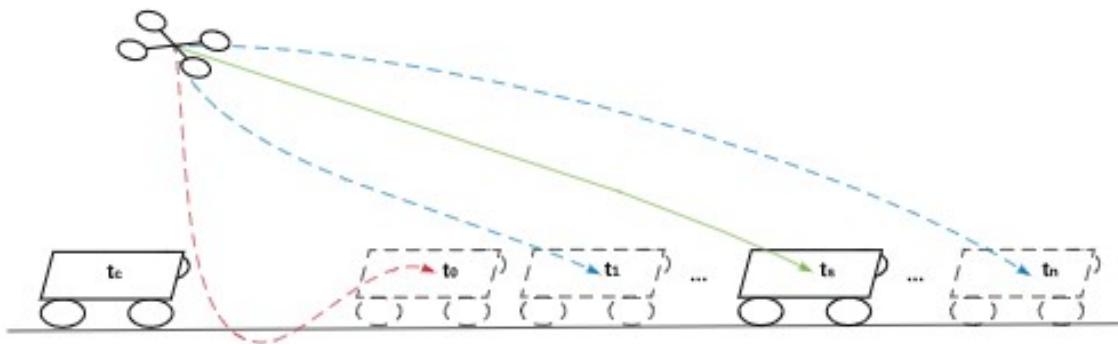
Rys. 2.2. Ilustracja określania pozycji znacznika w pracy [7]

składowych wektora położenia. W pracy [3] zwrócono uwagę na błędy spowodowane pochyleniem i przekrętem drona. Przed wysłaniem sygnału do regulatora wykonywana jest korekta uchybu na podstawie pomiaru kąta pochylenia i przekrętu. Krok ten był potrzebny z uwagi na sztywne umieszczenie kamery na ramie drona.

W przypadku wykonywania lądowania na poruszającej się platformie, do opisanego wcześniej problemu detekcji znacznika dochodzi również konieczność implementacji bardziej skomplikowanego algorytmu sterowania. Wysypane sygnały sterujące muszą uwzględniać ruch lądowiska. Przemieszczanie drona powinno nadążać za zmianą położenia platformy. W takim przypadku układ regulacji jest układem śledzącym.

W [6] użyto algorytmu pozwalającego na predykcję zachowania celu. Wykorzystano model dynamiczny celu oraz rozszerzony filtr Kalmana. Na Rys. 2.3 przedstawiono schemat wybierania trajektorii. System planuje kilka możliwych dróg dotarcia do lądowiska. Każda z nich ma początek w aktualnej pozycji drona, a kończy się w przewidzianym położeniu platformy. Przyszły stan poruszającego się celu jest określany na podstawie jego modelu dynamicznego, począwszy od ostatniej dostępnej estymaty z filtra Kalmana. Trajektorie wymagające sterowania spoza dostępnego zakresu lub kolidujące z przeszkodami, są odrzucane (trajektoria oznaczona czerwoną przerywaną linią). Spośród wszystkich możliwych dróg (niebieskie przerywane linie) wybierana jest ta wymagająca użycia najmniejszej ilości energii. Zadanie optymalizacji jest wykonywane przy wykorzystaniu szybkiej wielomianowej generacji trajektorii minimalizującej trzecią pochodną położenia.

Podsumowując, wykonanie lądowania na statycznym lądowisku wymaga skutecznej detekcji znacznika znajdującego się na platformie, który określa jego lokalizację. Znalezienie względnej pozycji markera umożliwia przekazanie informacji o uchybie do regulatora sterującego dronem. Do urządzenia wykonawczego, czyli silników, dociera ostatecznie informacja o pożądanej wartości ciągu, która spowoduje



Rys. 2.3. Przykład wybierania trajektorii z pracy [6].

zbliżenie do celu. Po ustabilizowaniu unoszenia drona nad lądowiskiem następuje polecenie stopniowego obniżania lotu, aż do zetknięcia z ziemią. W przypadku lądowania na ruchomym celu nie ma możliwości doprowadzenia do zawisu nad platformą. Należy przeprowadzić predykcję zachowania lądowiska, wygenerować trajektorie i użyć jednej z metod optymalizacji do wyboru najlepszej z nich.

3. Sprzęt wykorzystany w projekcie

3.1. Platforma obliczeniowa

W pracy wykorzystano platformę Digilent Zybo Z7-20 z układem Zynq SoC (ang. *System on Chip*) XC7Z020-1CLG400C. Układ dostępny na karcie określa się jako heterogeniczny tj. stanowi połączenie części rekonfigurowalnej (PL – ang. *programmable logic*) oraz systemu procesorowego z dwurdzeniowym procesorem ARM Cortex-A9 taktowanym z częstotliwością 667 MHz (PS – ang. *processing system*).

Na Rys. 3.2 przedstawiono architekturę układu Zynq SoC. System procesorowy zaznaczony został na zielono, natomiast część rekonfigurowalna na żółto. Część PS zawiera wiele komponentów, między innymi:

- wydajne kontrolery 1Gb Ethernet, USB 2.0, SDIO (ang. Secure Digital Input Output),
- kontrolery SPI, UART, CAN, I2C,
- kontroler pamięci DDR3,
- magistralę *Advanced Microcontroller Bus Architecture Interconnect* (AMBA),
- inne kontrolery peryferiów z wejściami/wyjściami multipleksowanymi do 54 dedykowanych pinów MIO (ang. Multiplexed Input Output),
- piny EMIO (ang. Extended MIO) pozwalające na podłączenie komponentów poprzez część PL.

Kontrolery peryferiów są podłączone do części PS poprzez magistralę AMBA w trybie *slave*. W ten sposób uzyskują dostęp do rejestrów odczytu/zapisu, adresowalnych w pamięci procesora. Również część rekonfigurowalna jest połączona z magistralą AMBA poprzez porty AXI jako *slave*. Daje to możliwość szybkiej komunikacji między układem FPGA, a procesorem. Ponadto, moduły zaimplementowane w części PL mogą wywoływać przerwania w procesorze i otrzymywać dostęp DMA (ang. *Direct Memory Access*) do pamięci DDR3.

Poniżej przedstawiono komponenty części rekonfigurowalnej:

- 53 200 tablic LUT (ang. *Look-up Table*),
- 106 400 przerzutników *flip-flop*,



Rys. 3.1. Układ ZYBO Z7-20 wraz z kamerą PCAM 5C.

- 630 KB pamięci blokowej RAM,
- 6 obszarów zarządzania zegarami CMT (ang. *Clock Management Tiles*),
- konwerter analogowo-cyfrowy.

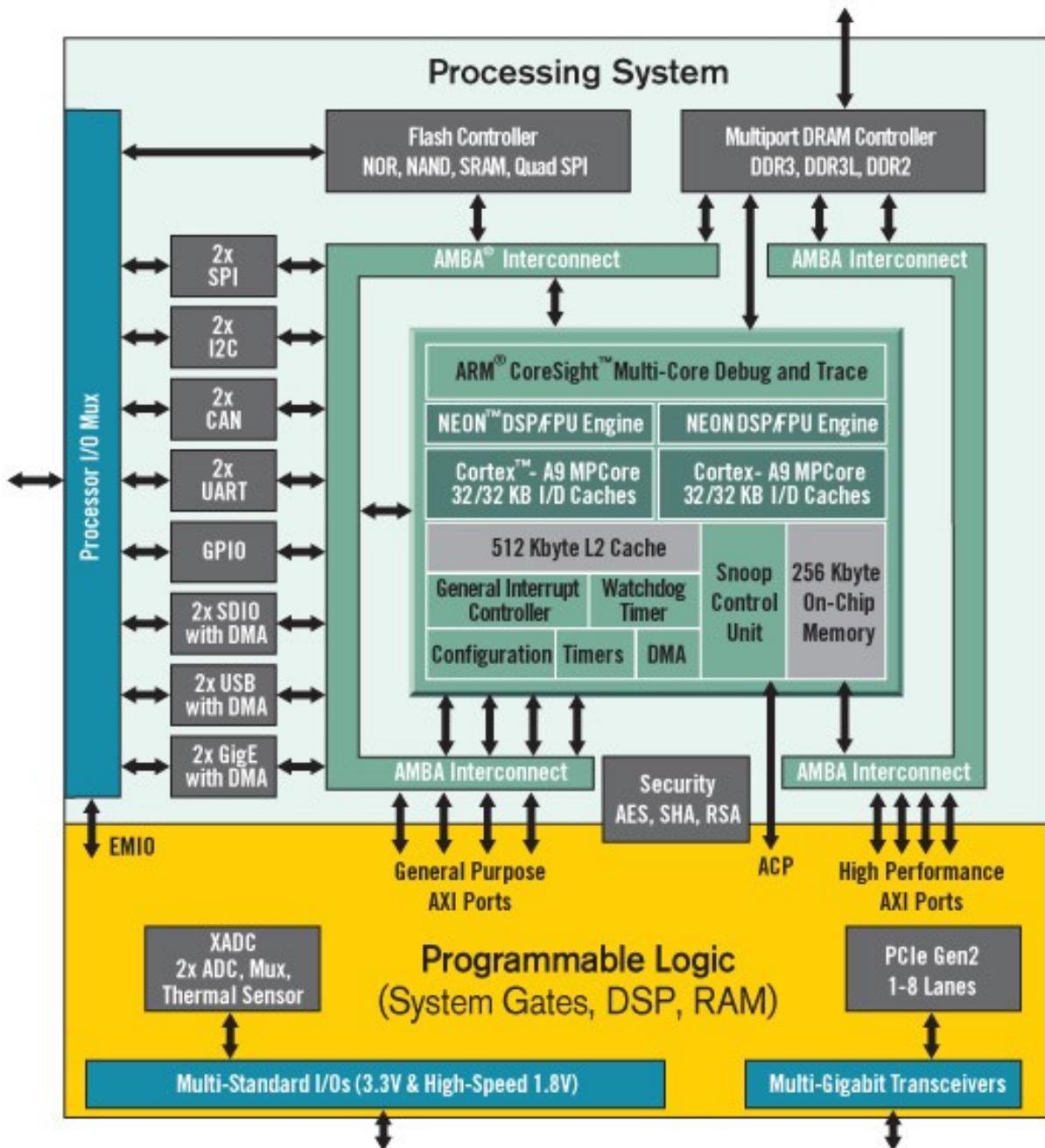
Do pozostałych części układu ZYBO Z7-20 należą między innymi:

- łącznik kamery PCAM ze wsparciem MIPI CSI-2
- wejściowy oraz wyjściowy port HDMI
- slot na kartę SD
- 6 portów PMOD
- 4 przełączniki
- 5 diod LED

3.2. Moduł rejestrujący obraz

W projekcie wykorzystano kolorową kamerę Digilent PCAM 5C. Jest to moduł przeznaczony do użycia z dedykowanymi płytami rozwojowymi. Jednym z kompatybilnych układów jest ZYBO Z7-20. Na Rys. 3.1 przedstawiono kamerę podłączoną do układu. Właściwości modułu przedstawiono poniżej:

- Rozdzielcość: 5 MPx,
- Matryca: OV5640,
- Interfejs danych: MIPI CSI-2



Rys. 3.2. Architektura układu Zynq SoC [9].

- Złącze: 15-pinowe FFC

Sposób podłączenia kamery i układu ZYBO Z7-20 przedstawione zostały w sekcji 4.2.1.

3.3. Platforma statku powietrznego

Zdecydowano się na dron sześciowirnikowy, znajdujący się na wyposażeniu studenckiego koła naukowego AVADER. Zbudowanie drona było częścią innego projektu [10]. Elementy użytego bezzałogowego statku powietrznego to:

- rama DJI F550,
- śmigła o średnicy równej 22,86 cm oraz skoku 12,7 cm,
- silniki DJI 2312/960KV z kontrolerami 420 LITE,
- czterokomorowa bateria LiPo o nominalnym napięciu 14,8 V i pojemności 6450mAh,
- nadajnik radiowy FrSky Taranis X9D Plus, odbiornik FrSky X8D,
- sterownik 3DR Pixhawk.

3.4. Autopilot

Urządzeniem bezpośrednio komunikującym się z kontrolerami silników drona jest autopilot. W pracy wykorzystano sterownik Pixhawk, będący popularnym kontrolerem lotu ogólnego przeznaczenia dostępnym na otwartej licencji. Jego główne parametry to:

- procesor Cortex-M4F z zegarem 168 MHz,
- czujniki: akcelerometr, żyroskop, kompas magnetyczny, barometr i zewnętrzny moduł GPS,
- interfejsy: UART, CAN, I2C, SPI,
- wejście karty SD,
- zewnętrzny przełącznik bezpieczeństwa uzbrajania,
- wielokolorowa dioda pokazująca stan pracy

Konfiguracja sterownika jest możliwa przy użyciu programu Mission Planner. Jest to aplikacja przeznaczona dla stacji naziemnej umożliwiająca:

- strojenie czujników autopilota,
- monitorowanie stanu sterownika (uzbrojenie silników, orientacja statku powietrznego),
- planowanie, zapisywanie i wgrywanie planów autonomicznych misji statku powietrznego,
- pobieranie i analizowanie dzienników misji.

Po instalacji oprogramowania ArduPilot sterownik umożliwia pracę w różnych trybach. Dostępne są 23 wbudowane tryby, z czego 5 jest rekomendowanych [11]. Istnieją tryby wspierające różne poziomy i typy stabilizacji, a zadania wykonywane przez autopilota w każdym z nich nieco się różnią. Poniżej przedstawiono podsumowanie najważniejszych cech rekomendowanych trybów:

- Stablize – pozwala operatorowi na manualne sterowanie dronem, stabilizując jego pochylenie i przechylenie,
- Alt Hold – kontroluje ustawienie mocy silników w celu utrzymania wysokości
- Loiter – utrzymuje aktualną pozycję drona, jeśli operator nie wydaje żadnych poleceń,
- Return-To-Launch – dron unosi się na domyślną wysokość 15 metrów, a następnie powraca do miejsca, w którym został uzbrojony. Tryb ten wymaga sygnału GPS,
- Auto – dron realizuje zaprogramowaną wcześniej misję, poruszając się po zdefiniowanych punktach. Wymagany jest sygnał GPS.

Z punktu widzenia autonomicznego lądowania, odpowiednim trybem do sterowania dronem jest tryb Guided. Umożliwia on bowiem wysyłanie dronowi komend dotyczących zmiany szybkości i lotu w określonym kierunku. Zazwyczaj wykorzystywana jest komunikacja ze stacją naziemną drogą radiową (przy użyciu programu Mission Planner), lecz wysyłanie komend przez urządzenie umieszczone na platformie drona również jest możliwe.

Komunikacja taka przebiega z wykorzystaniem protokołu MAVLink (ang. *Micro Air Vehicle Link*). Umożliwia on przesyłanie danych i komend z wykorzystaniem prawie każdej transmisji szeregowej [12]. W tabeli 3.1 przedstawiono opis ramki protokołu.

Kwestię połączenia autopilota i układu ZYBO Z7-20 opisano w sekcji 4.2.11.



Rys. 3.3. Laserowy czujnik odległości Garmin Lidar Lite v3 [13]

Tabela 3.1. Opis ramki protokołu MAVLink

Indeks bajtu	Zawartość	Wartość	Uwagi
0	Znak początku pakietu	0xFE	Wskazuje na początek nowego pakietu.
1	Długość danych	n (0-255)	Informuje o ilości przesyłanych danych.
2	Sekwencja pakietu	0-255	Wartość zwiększana przy każdej transmisji. Umożliwia detekcję utraty pakietów.
3	ID systemu	1-255	ID systemu wysyłającego. Umożliwia odróżnienie różnych sieci.
4	ID komponentu	0-255	ID komponentu wysyłającego. Umożliwia odróżnienie różnych komponentów w jednej sieci.
5	ID wiadomości	0-255	Oznacza typ wiadomości i umożliwia jej zdekodowanie.
6-n+6	Dane	(0-255) bajtów	Zawartość wiadomości. Zależy od jej ID.
n+7-n+8	Suma kontrolna (młodszy i starszy bajt)		Umożliwia wykrycie błędów transmisji

3.5. Laserowy czujnik wysokości

W projekcie do pomiaru wysokości wykorzystano urządzenie Garmin Lidar Lite v3. Jego główne parametry to:

- Napięcie zasilania – 5 V,
- Zasięg pomiaru – 40 m,
- Dokładność – 2,5 cm (pomiar do 5 m),
- Wykorzystywana długość fali – 905 nm,
- Interfejsy – I2C, PWM z zewnętrznym wyzwalaniem pomiaru.

Na Rys. 3.3 przedstawiono moduł wraz z dołączonymi przewodami.

4. Zaimplementowany system wspomagający autonomiczne lądowanie drona

W rozdziale omówiono zagadnienia związane ze szczegółami implementacji systemu. Ważną rolę w tworzeniu kolejnych modułów odegrał model programowy, czyli program napisany w dowolnym języku programowania i wykonywany na komputerze PC, którego działanie oddaje funkcje modułu twozonego w sprzęcie. Porównanie wyników działania modelu programowego i symulacji modułu daje informację o poprawności implementacji. Model programowy wraz z uzasadnieniem użycia poszczególnych modułów przedstawiono na początku rozdziału.

Następnie opisano wszystkie części przetwarzania sygnału, od przesłania obrazu przez kamerę, do wysyłania komend do autopilota.

4.1. Implementacja i ewaluacja modelu programowego

Model programowy został napisany w pakiecie Matlab przy wykorzystaniu funkcji dostępnych w bibliotece *Image Processing Toolbox*. Pozwoliło to na szybkie prototypownie systemu wizyjnego, składającego się z:

- konwersji z przestrzeni RGB do YCbCr,
- binaryzacji ze stałymi progami,
- mediany,
- otwarcia,
- indeksacji jednoprzebiegowej wyznaczającej pole powierzchni i prostokąt otaczający obiektów.

Piksel w przestrzeni barw YCbCr opisują trzy składowe: Y (luminancja), Cb (chrominancja, która wyraża różnicę między luminancją, a kolorem niebieskim) oraz Cr (chrominancja, która wyraża różnicę między luminancją, a kolorem czerwonym).

Zaletą stosowania tej przestrzeni barw jest oddzielenie sygnału luminancji od sygnałów chrominancji, pozwalające na dokonywanie przetwarzania sygnału przy mniejszej zależności od oświetlenia.

Zastosowanie binaryzacji pozwala na oddzielenie znacznika od tła. Na wyjściu modułu znajdują się tylko

dwa rodzaje pikseli: należące i nienależące do obiektu (Rys. 4.1a). Na obrazie często znajdują się jednak inne niewielkie obszary (zakłócenia), które zostały wykryte. Pozostawienie ich na wejściu modułu indeksacji zwiększyłoby niepotrzebnie liczbę nadawanych etykiet, która jest ograniczona (sekcja 4.2.8). Częściowym rozwiązańiem problemu jest zastosowanie mediany. Jest to operacja kontekstowa, w której pikselowi wyjściowemu przypisuje się wartość środkową uporządkowanego zbioru wartości pikseli z otoczenia piksela wejściowego.

Wykorzystanie mediany powoduje znaczne zmniejszenie błędnych obszarów, jednakże niekiedy na obrazie obecne są nadal małe grupy białych pikseli nienależących do obiektu (Rys. 4.1b).

Z tego powodu zdecydowano się na wykorzystanie erozji. Erozja i dylatacja to operacje morfologiczne, w których pikselowi wyjściowemu przypisuje się wartość odpowiednio najmniejszego i największego piksela w sąsiedztwie piksela wejściowego. Sąsiedztwo piksela określa kształt i rozmiar elementu strukturalnego. Zastosowanie morfologicznego otwarcia, składającego się z erozji i dylatacji, pozwala niekiedy na całkowite wyeliminowanie błędnych białych pikseli (Rys. 4.1c). Indeksacja jednoprzebiegowa to operacja pozwalająca na wyodrębnienie z obrazu cech poszczególnych obiektów. Obiekty rozumiane są jako grupy połączonych ze sobą pikseli. Ze względu na wykorzystywany współczynnik kształtu oznaczano prostokąt otaczający i pole obiektów.

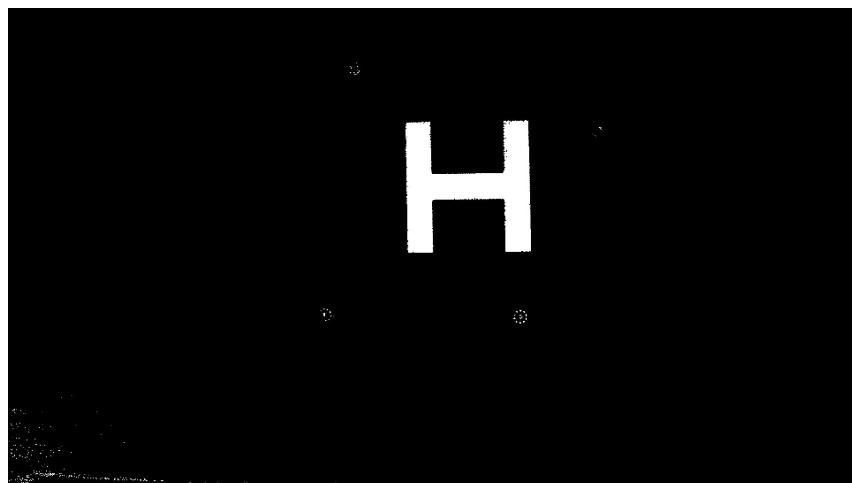
4.1.1. Akwizycja ramek na kartę SD

Aby możliwe było wprowadzanie ramek obrazu do modelu, należało umożliwić akwizycję obrazów na kartę SD. W pobranym projekcie zastosowano połączenie toru wizyjnego AXI z pamięcią procesora. Połączenie realizuje moduł AXI Video Direct Memory Access. Informacja o miejscu w pamięci, gdzie zapisywane są ramki podawana jest przy inicjalizacji połączenia na terminal.

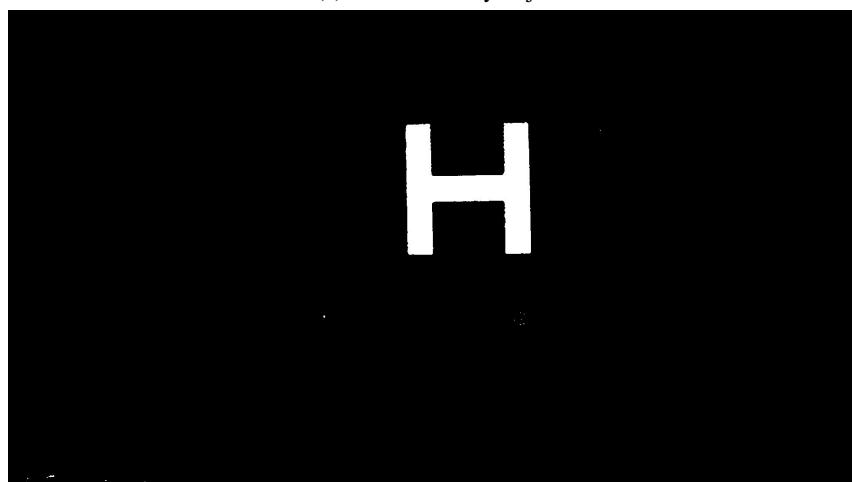
W celu umożliwienia komunikacji z kartą uaktywniono interfejs procesora SD0. Następnie do projektu w SDK dodano bibliotekę „xilffs” i, korzystając z funkcji systemu plików opisanych w [14], napisano zapis ramek do pliku. Ze względu na prostotę pliku zdecydowano się na format ppm. W formacie nie występuje żadna kompresja i jest on nieefektywny pod względem zapotrzebowania na pamięć. Niemniej jednak, dysponując odpowiednio dużą ilością miejsca na karcie i chcąc w łatwy sposób zapisać dane, wybrano właśnie ten format. Zapis danych w formacie ppm jest prosty, gdyż plik składa się jedynie z nagłówka (typ pliku, rozmiary obrazka, maksymalna wartość składowych) oraz kolejnych wartości w formie pojedynczych bajtów (jeśli maksymalna wartość jest mniejsza niż 256).

4.2. Implementacja sprzętowo-programowa

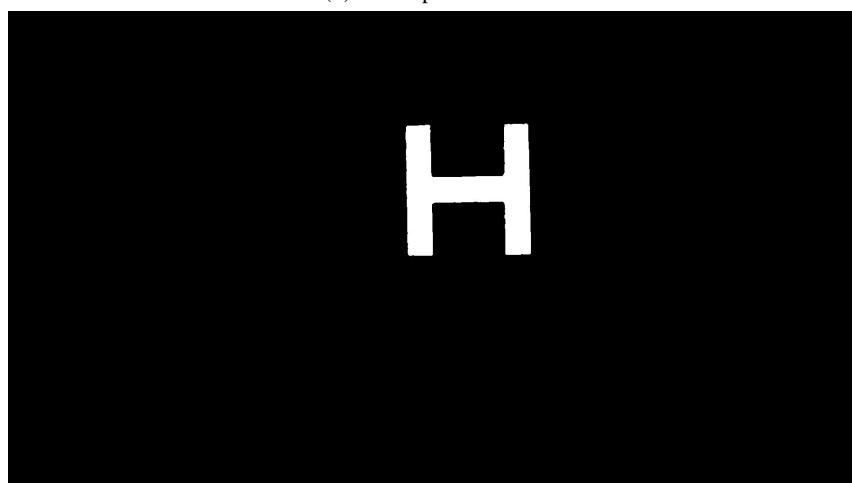
Na Rys. 4.2 przedstawiono ogólny schemat implementacji systemu. Na niebiesko zaznaczone zostały komponenty sprzętowe, na żółto oznaczono moduły zaimplementowane w części rekonfigurowalnej, natomiast na zielono zaznaczono system procesorowy. Z kamery trafia do układu sygnał wizyjny, który następnie jest przetwarzany przez poszczególne moduły. Po wyborzeznacznika informacja o jego pozycji



(a) Rezultat binaryzacji.



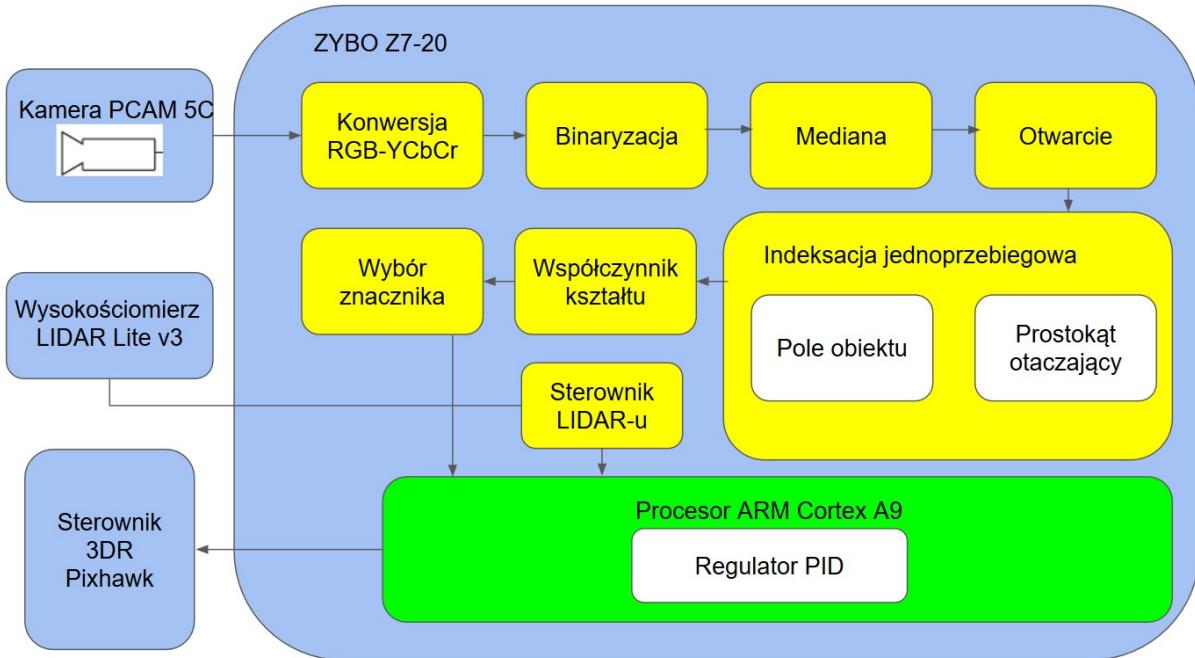
(b) Obraz po medianie



(c) Wynik otwarcia

Rys. 4.1. Przykładowe rezultaty binaryzacji, mediany i otwarcia

i uchybie regulacji trafia do procesora. System procesorowy wysyła komendy sterujące do autopilota. Zadaniem układu regulacji jest przesunięcie drona nad lądowisko, umożliwiając wykonanie lądowania. W dalszej części rozdziału omówiono poszczególne komponenty systemu.



Rys. 4.2. Schemat przedstawiający zaimplementowany system.

4.2.1. Integracja kamery i płytka

Producent na swojej stronie internetowej zapewnia projekt demonstracyjny połączenia kamery i płytka [15]. Przez port szeregowy możliwa jest zmiana rozdzielczości, szybkości akwizycji ramek, współczynnika korekcji gamma, ustawień balansu bieli. Możliwe są następujące opcje dotyczące dwóch pierwszych parametrów:

- 1280 x 720, 60 fps,
- 1920 x 1080, 15 fps,
- 1920 x 1080, 30 fps.

Okazało się również, że dla rozdzielczości 1280 x 720 większy jest kąt widzenia kamery. Ze względu na powyższy fakt oraz przyspieszenie obliczeń, zdecydowano się na najmniejszą dostępną rozdzielczość. Przeprowadzanie syntezы i implementacji projektu możliwe było przy użyciu darmowego oprogramowania Vivado oraz SDK w wersji WebPack – używano wersji 2018.2.

Pobrany projekt stanowił bazę do dalszych prac.

4.2.2. Podłączenie laserowego czujnika wysokości

Zdecydowano się na wykorzystanie wyjścia PWM. W tym trybie czujnik ustawia stan wysoki na wyjściu przez czas proporcjonalny do zmierzonej odległości. Stan wysoki trwa 10 µs na każdy zmierzony centymetr. Pin wyzwalania zatrzymuje się na stałe do masy, powodując ciągłe wykonywanie pomiaru. Wyjście PWM podłączeno do trzeciego pinu portu Pmod JB układu ZYBO. Okazało się, że możliwe jest działanie modułu przy zasilaniu bezpośrednio z płytki, podającej 3,3 V.

Moduł odczytujący wysokość zaimplementowany został jako maszyna stanów. Stan pierwszy to oczekiwanie na pojawienie się stanu wysokiego. Po wykryciu zbocza narastającego następuje zerowanie rejestrów oraz przejście do stanu drugiego. W tym stanie liczony jest czas trwania stanu wysokiego. Wywoływanie procesu z częstotliwością 1 MHz, czyli co 1 µs pozwala na łatwą konwersję zmierzonego czasu na odległość. Odczytana wysokość jest następnie przesyłana do procesora z użyciem rejestrów AXI.

4.2.3. Konwersja z przestrzeni barw RGB do YCbCr

Konwersję z przestrzeni barw RGB do YCbCr wykonano zgodnie ze wzorem 4.1. Przy implementacji wykorzystano sprzętowe mnożarki oraz sumatory. Na Rys. 4.3 przedstawiono schemat operacji arytmetycznych dla jednej ze składowych. Aby możliwe było prowadzenie obliczeń, wszystkie stałe należało przedstawić w postaci liczb stałoprzecinkowych. Do reprezentacji stałoprzecinkowej użyto rejestrów o szerokości 18 bitów.

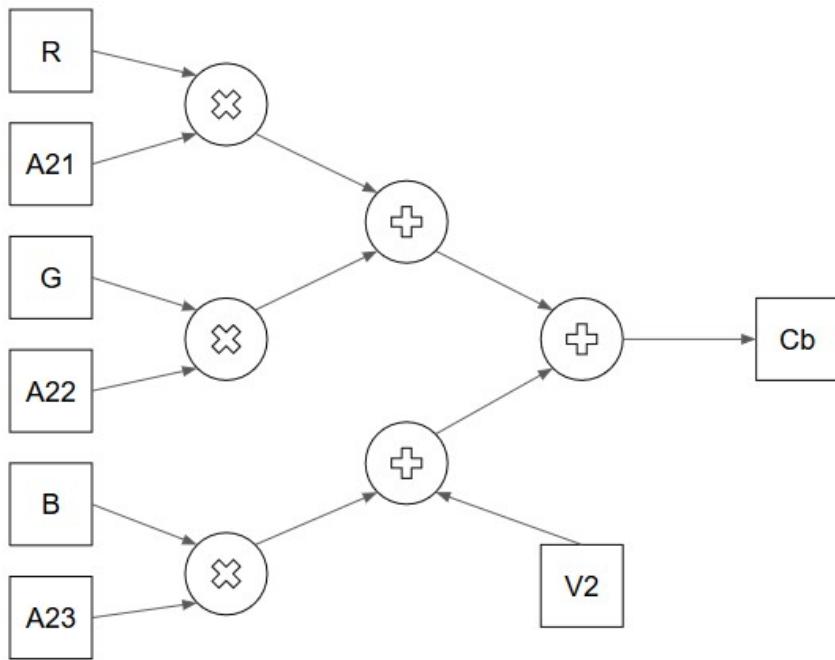
$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0,299 & 0,587 & 0,114 \\ -0,168736 & -0,331264 & 0,5 \\ 0,5 & -0,418688 & 0,081312 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix} \quad (4.1)$$

4.2.4. Binaryzacja

Piksel wyjściowy otrzymywał wartość maksymalną (kolor biały), jeśli wartości Cb i Cr mieściły się pomiędzy wyznaczonymi eksperymentalnie progami. W innym przypadku pikselowi przypisywana była wartość 0 (kolor czarny). Do szybkiej zmiany progów wykorzystano komunikację PS-PL z wykorzystaniem rejestrów AXI - progi mogły być zmieniane z poziomu terminala i testy nie wymagały ponownych implementacji.

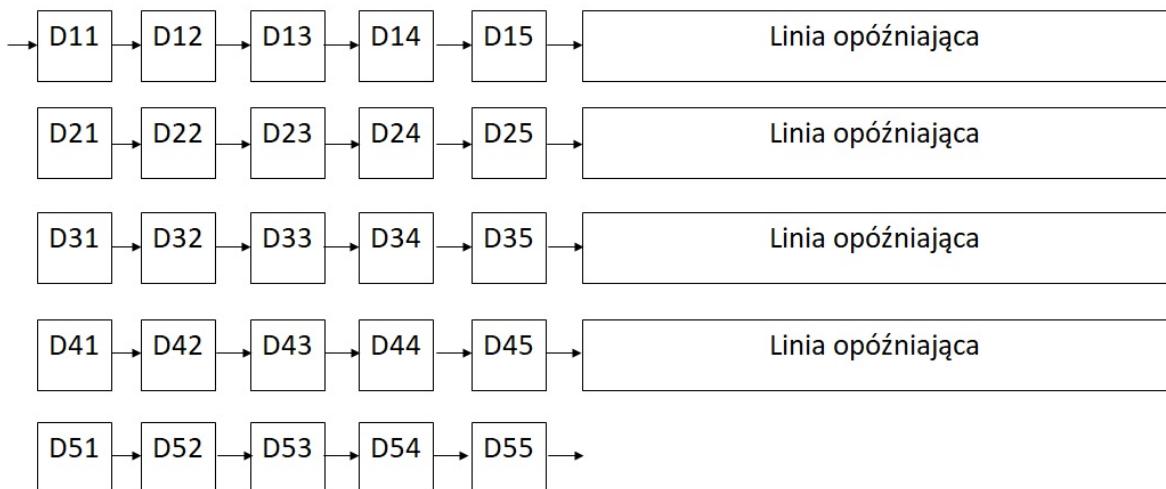
4.2.5. Mediana

W przypadku działania na obrazie binarnym operacja mediany może być przeprowadzona przez obliczenie sumy wartości pikseli wewnątrz kontekstu, a następnie porównanie jej z połową maksymalnej wartości tej sumy. Ze względu na łatwość implementacji oraz zadowalające wyniki filtracji, zdecydowano się na rozpatrywanie kontekstu w kształcie kwadratu o boku 5 pikseli. Spowodowało to konieczność zapamiętywania kontekstu piksela w 25 rejestrach oraz 4 linii obrazu w długich liniach opóźniających zbudowanych w oparciu o pamięć BRAM. Schemat wyznaczania kontekstu przedstawiono na

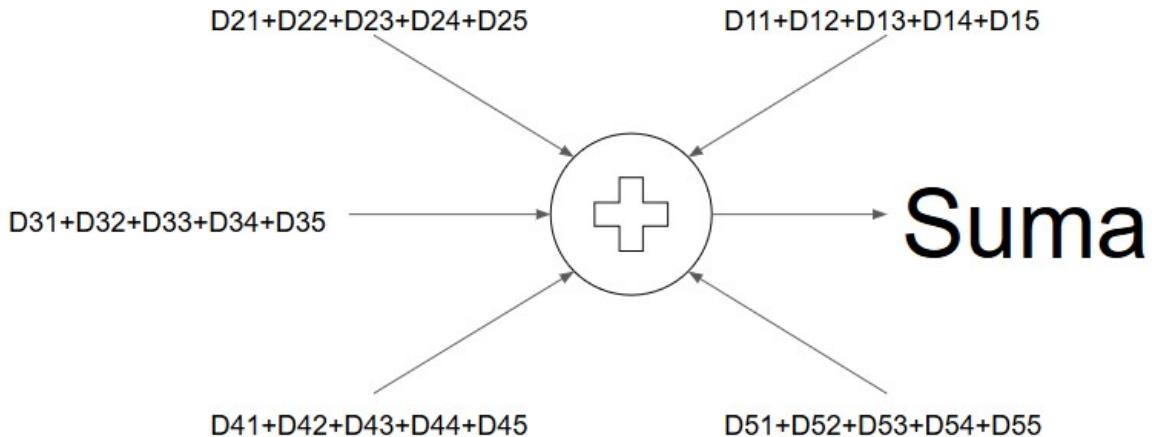


Rys. 4.3. Schemat wyznaczania składowej C_b . $A_{21} = -0,168736, A_{22} = -0,331264, A_{23} = 0,5, V_2 = 128$.

rysunku 4.4. Sygnały synchronizacji zostały doklejone do wartości piksela i w przedstawionej strukturze przesuwają się razem z nim. Sumę wyliczano w 2 etapach, dodając najpierw elementy w wierszach, potem sumując wyniki (Rys. 4.5).



Rys. 4.4. Schemat wyznaczania kontekstu dla mediany, erozji i dylatacji.



Rys. 4.5. Schemat wyznaczania sumy otoczenia piksela.

4.2.6. Erozja i dylatacja

Podobnie jak w przypadku mediany, jako element strukturalny zdecydowano się na kwadrat o boku 5 pikseli. Moduł erozji ustawia wartość piksela wyjściowego na maksymalną wartość, gdy w sąsiedztwie znajdują się same białe piksele. W module dylatacji zwracana jest wartość maksymalna, gdy przynajmniej jeden piksel w sąsiedztwie ma wartość maksymalną.

4.2.7. Prostokąt otaczający i pole powierzchni

Znalezienie prostokąta otaczającego sprowadza się do wyznaczenia skrajnych jego punktów na górze, dole, po lewej oraz prawej stronie. Na podstawie sygnałów synchronizacji oraz wymiarów obrazka wyznaczano współrzędne aktualnie przetwarzanego piksela. Jeśli jest to piksel należący do obiektu, następuje inkrementacja jego pola powierzchni. Do odpowiednich rejestrów trafiają wówczas również wartości współrzędnych piksela, jeśli wykraczają poza aktualną zawartość rejestrów:

- do rejestru zawierającego górnego bok prostokąta trafi współrzędna wierszowa, jeśli będzie ona mniejsza od aktualnej,
- do rejestru zawierającego dolny bok prostokąta trafi współrzędna wierszowa, jeśli będzie ona większa od aktualnej,
- do rejestru zawierającego lewy bok prostokąta trafi współrzędna kolumnowa, jeśli będzie ona mniejsza od aktualnej,
- do rejestru zawierającego prawy bok prostokąta trafi współrzędna kolumnowa, jeśli będzie ona większa od aktualnej,

Obliczeń pola powierzchni i współrzędnych prostokąta otaczającego nie wykonywano w oddzielnym module, lecz stanowiły one część modułu indeksacji.

4.2.8. Indeksacja

Zazwyczaj na wejście modułu indeksacji podawany jest obraz zbinaryzowany, natomiast na wyjściu pojawia się obraz, na którym wartość pikseli odpowiada przypisanej do danego obiektu etykiecie. Rozważane jest otoczenie każdego piksela, składające się z trzech pikseli nad nim oraz jednego po lewej stronie, tak jak zostało to przedstawione na rysunku 4.6. Indeksacji dokonuje się bezpośrednio na obrazie wejściowym. Podczas iteracji po wszystkich pikselach, w przypadku znalezienia piksela należącego do któregoś z obiektów, może zajść jeden z trzech przypadków:

- (a) w otoczeniu piksela znajdują się tylko piksele należące do tła,
- (b) otoczenie zawiera jeden lub więcej pikseli, którym została wcześniej przypisana taka sama etykieta L ,
- (c) w otoczeniu znajdują się piksele posiadające różne etykiety.

D5	D4	D3
D2	D1	

Rys. 4.6. Sąsiedztwo piksela brane pod uwagę przy indeksacji

W pierwszym przypadku pikselowi zostaje przypisana nowa etykieta. Gdy spełniony jest warunek b , punkt otrzymuje etykietę L , natomiast jeśli zachodzi przypadek c , przypisywana jest mniejsza z etykiet. W ten sposób otrzymuje się obraz wstępnie poetykietowany. Najczęściej posiada on więcej przypisanych etykiet, niż obiektów. Dlatego istnieje konieczność złączenia ze sobą pewnych etykiet przy użyciu tablicy sklejeń. Tablica ta zawiera informację, które etykiety powinny zostać złączone. W przypadku a do tablicy sklejeń na pozycji odpowiadającej etykiecie zapisywana jest etykieta, natomiast gdy zachodzi opcja c etykietę mniejszą zapisuje się pod indeksem większym. Do sklejenia etykiet potrzebna jest druga iteracja, tym razem po obrazie wstępnie poetykietowanym.

Powyższy fakt jest główną przeszkodą w łatwym wykonaniu takiego algorytmu w systemie potokowym. Bez zapamiętywania całej ramki, w przypadku sklejania etykiet, niemożliwy jest powrót do wcześniej przetwarzanych pikseli. Pomimo tego, możliwe jest obliczenie pewnych cech obiektów, takich jak: pole, współrzędne środka ciężkości, prostokąt otaczający. W pracy [16] podano sposób, w jaki można tego dokonać. Opiera się on na scaleniu nie samych wartości pikseli, ale obliczanych na bieżąco parametrów obiektu. Implementacja indeksacji w języku Verilog rodzi trudności związane z określeniem przypadku istnienia tej samej lub różnych etykiet w otoczeniu piksela. O ile wykrycie

przypadku *a* jest łatwe, to przypadki *b* i *c* wymagały rozważenia kilku możliwości. Zdecydowano się na wykrywanie ich za pomocą flag bitowych. Na ich podstawie wnioskowano o zachodzącym aktualnie przypadku oraz wskazywano, od którego piksela z otoczenia powinna zostać przepisana etykieta. Zgodnie z opisanymi wcześniej zasadami uzupełniano również tablicę sklejeń. Oprócz tego, na bieżąco obliczano prostokąt otaczający oraz liczbę pikseli należących do każdego ze znalezionych obiektów.

Po poetykietowaniu całej ramki obrazu wykorzystano tablicę sklejeń do złączenia obliczanych na bieżąco cech. Ten etap algorytmu zaimplementowano jako maszynę stanów:

- Stan 0 – Oczekiwanie na sygnał końca ramki wyznaczany na podstawie synchronizacji pionowej. W momencie wykrycia sygnału następuje rejestrowanie tablicy sklejeń i obliczonych parametrów (w następnym taktie zostaną one zresetowane), zerowanie tablic wypełnianych w kolejnym etapie oraz przejście do stanu 1.
- Stan 1 – Iteracja po tablicy sklejeń i uzupełnianie rzeczywistych wartości cech obiektów.
- Stan 2 – Uporządkowanie tablic z wyznaczonymi parametrami.
- Stan 3 – Obliczenie pola prostokąta otaczającego dla każdego znalezionego obiektu. Wykorzystywana jest mnożarka o latencji 3.
- Stan 4 – Szukanie obiektu spełniającego warunki minimalnej wielkości pola powierzchni oraz stosunku pola prostokąta otaczającego do pola obiektu.

Schemat przetwarzania danych po każdej ramce pokazano na rysunku 4.7.

Główną trudnością w implementacji opisanego algorytmu w układzie FPGA jest stosunkowo duże zapotrzebowanie na zasoby sprzętowe. Należy zarezerwować miejsce na cechy każdego potencjalnego obiektu. Ogranicza to liczbę możliwych etykiet. Zasoby części rekonfigurowalnej układu ZYBO Z7-20 pozwoliły na zarezerwowanie miejsca dla 30 etykiet. Przy pewnej orientacji znacznika nie jest to niestety wystarczająca liczba etykiet.

Aby możliwa była sprawna detekcja miejsca lądowania niezależnie od orientacji markera, należało zmieścić nieco koncepcję systemu. Innym podejściem było obliczanie środka ciężkości wszystkich białych pikseli. Zabiera to możliwość klasyfikacji obiektów ze względu na kształt (pozostaje klasyfikacja po kolorze) i tym samym czyni system podatnym na zakłócenia. Pojawienie się innych grup pikseli o tym samym kolorze spowoduje przesunięcie środka ciężkości. Jednak w warunkach testowych możliwe jest wypełnienie powierzchni kolorem silnie kontrastującym z barwą znacznika. Z tego powodu zaimplementowano moduł wyznaczający środek ciężkości i podczas testów korzystano z tego rozwiązania.

4.2.9. Środek ciężkości

Do wyznaczania środka ciężkości pikseli należących do obiektu wykorzystano wzory (4.2), (4.3) i (4.4).

Tablica sklejeń:

1	1
2	1
3	1
4	4
5	5
6	5
7	4

Najmniejsze wartości dla każdego obiektu:

Wartości przygotowane do wyboru znalezionego obiektu:

1	10
2	0
3	0
4	8
5	17
6	0
7	0

1	10
2	8
3	17
4	0
5	0
6	0
7	0

Znalezione położenia lewego boku prostokąta:

1	15
2	10
3	20
4	25
5	22
6	17
7	8

Rys. 4.7. Schemat procesu przetwarzania informacji po każdej ramce obrazu przy zastosowaniu indeksacji z przykładowymi danymi. Tablica sklejeń daje informację o konieczności sklejenia etykiet 1, 2, 3. Ponieważ przykład dotyczy lewego boku prostokąta, spośród liczb 15, 10, 20 (odpowiadającym etykietom 1, 2, 3) wybierana jest najmniejsza. Ostatni krok przetwarzania to usunięcie środkowych zer z wektora.

$$m_{00} = \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} x_{ij} \quad (4.2)$$

$$m_{10} = \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} i * x_{ij} \quad (4.3)$$

$$m_{01} = \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} j * x_{ij} \quad (4.4)$$

gdzie:

 N – szerokość obrazu w pikselach, M – wysokość obrazu w pikselach, x_{ij} – wartość piksela o współrzędnych i, j obrazu zbinaryzowanego.

Na ich podstawie obliczono środek ciężkości przy zastosowaniu wzorów (4.5) i (4.6).

$$X_{sc} = \frac{m_{10}}{m_{00}} \quad (4.5)$$

$$Y_{sc} = \frac{m_{01}}{m_{00}} \quad (4.6)$$

gdzie:

- X_{sc} – współrzędna pozioma środka ciężkości,
 Y_{sc} – współrzędna pionowa środka ciężkości.

W module na podstawie sygnałów synchronizacji oraz wymiarów obrazka wyznaczono współrzędne aktualnie przetwarzanego piksela. Jeśli jest to piksel należący do obiektu, następuje zwiększenie wartości odpowiednich rejestrów zgodnie ze wzorami 4.2, 4.3 i 4.4. Po przejściu przez całą ramkę obrazu wykonywane jest dzielenie na podstawie wzorów 4.5 i 4.6.

4.2.10. Algorytm sterowania

Wyznaczanymi informacjami o detekcji, koniecznymi do realizacji algorytmu sterowania, są:

- uchyb regulacji w obu osiach, rozumiany jako odległość środka obrazu od środka wykrytego obiektu (centrum prostokąta otaczającego lub środka ciężkości),
- wiadomość o znalezieniu znacznika (w przypadku stosowania indeksacji).

Dodatkowo, w celu śledzenia działania systemu, istnieje możliwość podglądu przetworzonego obrazu na monitorze, podłączonym poprzez port HDMI. Możliwy jest podgląd kolejnych etapów przetwarzania; wybór etapu następuje przez zmianę ustawień przełączników na płytce.

Algorytm sterowania bazuje na dyskretnym regulatorze PID, zrealizowanym w systemie procesorowym układu. Regulacji podlegają składowa x i y wektora położenia drona względem znacznika. Jeśli znacznik zostanie utracony z pola widzenia kamery na czas 1 sekundy, dron przechodzi w tryb unoszenia się nad ziemią.

4.2.11. Integracja płytki z autopilotem

W układzie ZYBO Z7-20 wykorzystano interfejs UART0. Wyprowadzono piny TX i RX z systemu procesorowego i podłączono je do portu Pmod JB. Po stronie sterownika Pixhawk użyto portu TELE 2. Szybkość transmisji to 115200 bodów.

Wysyłanie komend opiera się na wykorzystaniu gotowych funkcji dostępnych w sieci [17]. Przygotowują one daną wiadomość zgodnie z wymaganiami protokołu MAVLink. Następnie wywoływane są funkcje wysyłające przygotowaną tablicę bajtów przez odpowiedni UART. W projekcie wykorzystano stworzone przy wykonywaniu pracy [10] funkcje opakowujące kolejne etapy przetwarzania komend.

5. Testy i ocena wyników

5.1. Wybór znacznika

Autonomiczne lądowanie drona w oparciu o system wizyjny wymaga wyposażenia lądowiska w marker. Jego wygląd musi umożliwiać łatwą detekcję miejsca lądowania. Z powodu wykonywania operacji na zewnątrz, znacznik powinien również ułatwiać znalezienie go w zmiennych warunkach (zachmurzenie, cieśń).

5.1.1. Kształt

Detekcja kształtów może być realizowana na różne sposoby. Najprostszym jest progowanie współczynników kształtu, do bardziej zaawansowanych należy wyspecjalizowana deskrypcja cech (SIFT, SURF, HOG) i użycie klasyfikatorów (kNN, SVM, sieci neuronowe). W implementowanej pierwszej wersji systemu zdecydowano się na klasyfikację przy użyciu współczynnika kształtu.

Cechami obiektu łatwymi do wyliczenia w systemie potokowym są pole figury i najmniejszy prostokąt, w którym figura się mieści. Postanowiono zatem wykorzystać współczynnik kształtu przedstawiony we wzorze 5.1.

$$W = \frac{P_p}{P_o} \quad (5.1)$$

Gdzie:

W – współczynnik kształtu,

P_p – pole prostokąta otaczającego,

P_o – pole obiektu.

Aby taki współczynnik umożliwiał detekcję należało dobrać odpowiedni kształt znacznika. Zdecydowano się na krzyż, gdyż przy każdej jego orientacji pole prostokąta jest znacznie większe od pola obiektu.

5.1.2. Kolor

Kolor znacznika powinien umożliwiać łatwe wykrycie jego kształtu. Z tego powodu pożąданie jest silne skontrastowanie figury i tła. Najbardziej naturalnym rozwiązaniem jest rozważenie kontrastu

w przestrzeni barw RGB, gdyż taki sygnał jest dostarczany przez kamerę.

W przestrzeni RGB każdy piksel opisywany jest przez trzy składowe: czerwoną, zieloną i niebieską.



(a) Przedstawienie przestrzeni barw RGB w postaci sześcianu [18] (b) Przekroje sześciadanu przestrzeni YCbCr [19]

Rys. 5.1. Modele przestrzeni RGB i YCbCr

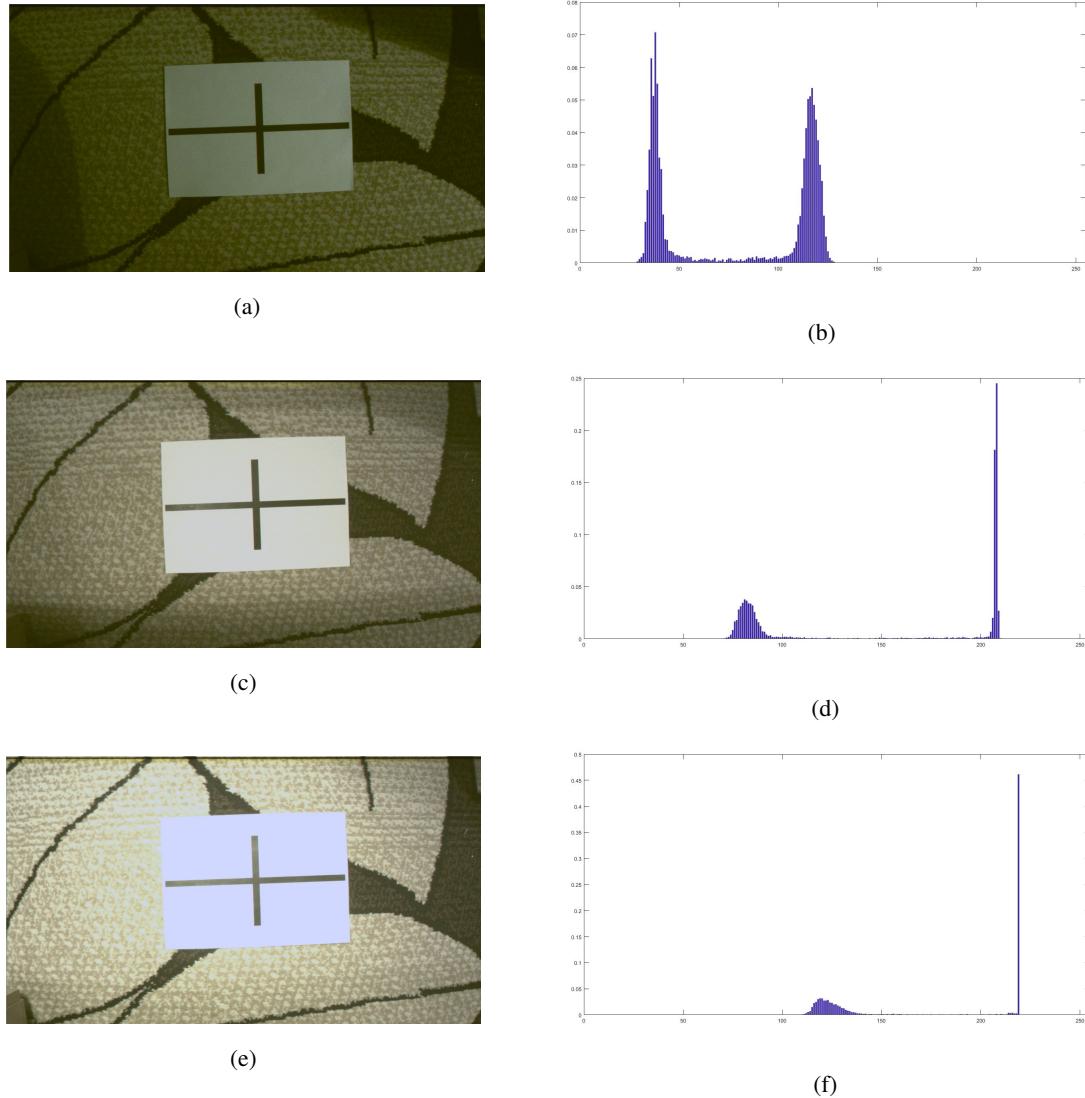
Możliwe jest przedstawienie tego systemu w formie sześcianu (Rys. 5.1a). Można wyznaczyć przekątną łączącą punkty, dla których wszystkie współrzędne są identyczne. Rozciąga się ona od czarnego w początku układu współrzędnych, do białego dla maksymalnych wartości składowych, przechodząc przez różne stopnie szarości. Wykorzystanie dużej odległości między kolorami i użycie czarno-białego znacznika wydaje się najprostszym pomysłem.

Do testów przygotowano czarny znacznik na białym tle. Kamerą PCAM 5C wykonano trzy zdjęcia przy różnym poziomie oświetlenia (Rys. 5.2a, 5.2c, 5.2e). Po przejęciu na obraz w skali szarości, posługując się narzędziem roipoly w programie Matlab, obliczono histogramy obszaru znacznika (Rys. 5.2b, 5.2d, 5.2f).

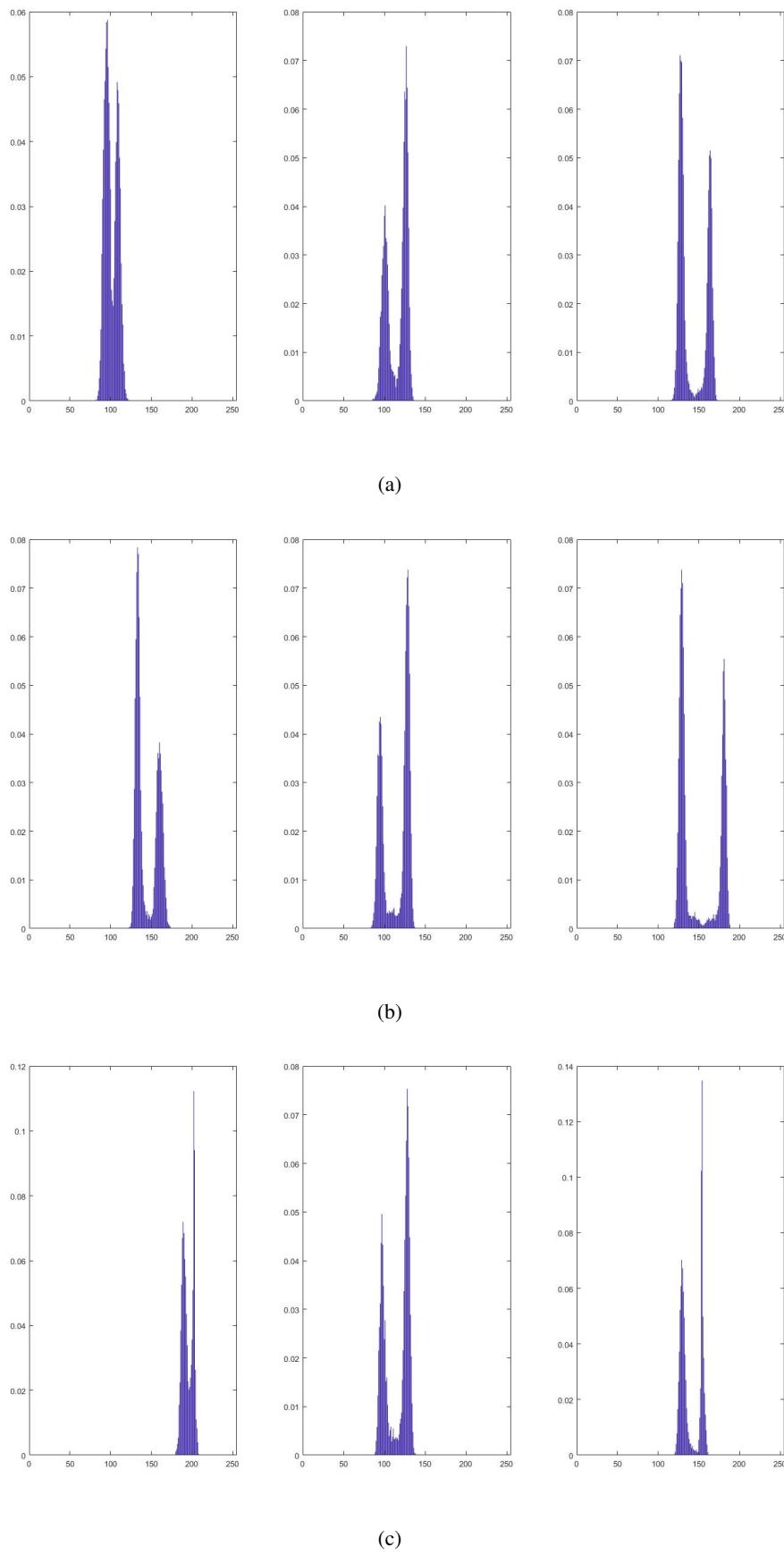
Analiza histogramów wskazuje na silne uzależnienie wartości koloru od oświetlenia. Dla kolejnych obrazów, wartości progów binaryzacji mogłyby zawierać się w zakresach: 50-100, 100-200, 150-210. Poziom białego tła dla obrazka naj słabiej oświetlonego jest mniejszy niż wartość czarnego koloru znacznika najlepiej oświetlonego. W takiej sytuacji niemożliwy jest dobór stałego progu umożliwiającego skuteczną binaryzację.

Z tego powodu zdecydowano się na powtórzenie eksperymentu w przestrzeni YCbCr. Tak jak opisano w sekcji 4.1, piksel w tej przestrzeni opisuje współrzędna luminancji i dwie składowe chrominacji. Podobnie jak w przypadku RGB, przestrzeń YCbCr również da się przedstawić w postaci sześcianu. Tym razem skala szarości przebiega przez środek płaszczyzn Cr-Cb i za zmianę poziomu szarości odpowiada współrzędna luminancji (Rys. 5.1b). Dla każdej wartości piksela informacja o jasności oddzielona jest od barwy.

Kolor znacznika określono jako czerwony, natomiast tło niebieskie, gdyż te barwy znajdują się na przeciwnych stronach płaszczyzny Cr-Cb. Zdjęcia wykonano przy takich samych poziomach oświetlenia jak poprzednio. Histogramy kolejnych obrazów przedstawiono na Rys. 5.3a, 5.3b, 5.3c. Zgodnie z oczekiwaniemi, zmiana poziomu oświetlenia przełożyła się na zmianę wartości luminancji, natomiast składowe chrominacji nie zmieniały się znacząco. Każdy z obrazów może być skutecznie zbinaryzowany przy

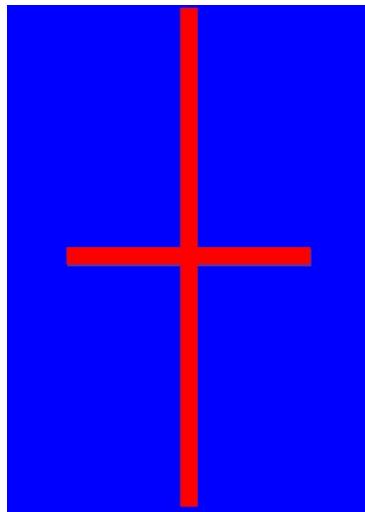


Rys. 5.2. Zdjęcia pierwszej wersji znacznika wraz z histogramami obszaru znacznika



Rys. 5.3. Histogramy obszarów znacznika w przestrzeni YCbCr.

użyciu stałych progów: górnego 120 dla składowej Cb i dolnego 150 dla składowej Cr. Różnice pomiędzy wartościami składowych dla znacznika i tła nie są jednak duże – wynoszą około 30. Z tego powodu postarano się o możliwość szybkiego dostrojenia progów bez konieczności ponownej generacji pliku konfiguracyjnego układu (sekcja 4.2.4). Ostatecznie zdecydowano się na znacznik przedstawiony na Rys. 5.4.



Rys. 5.4. Wybrany wygląd znacznika

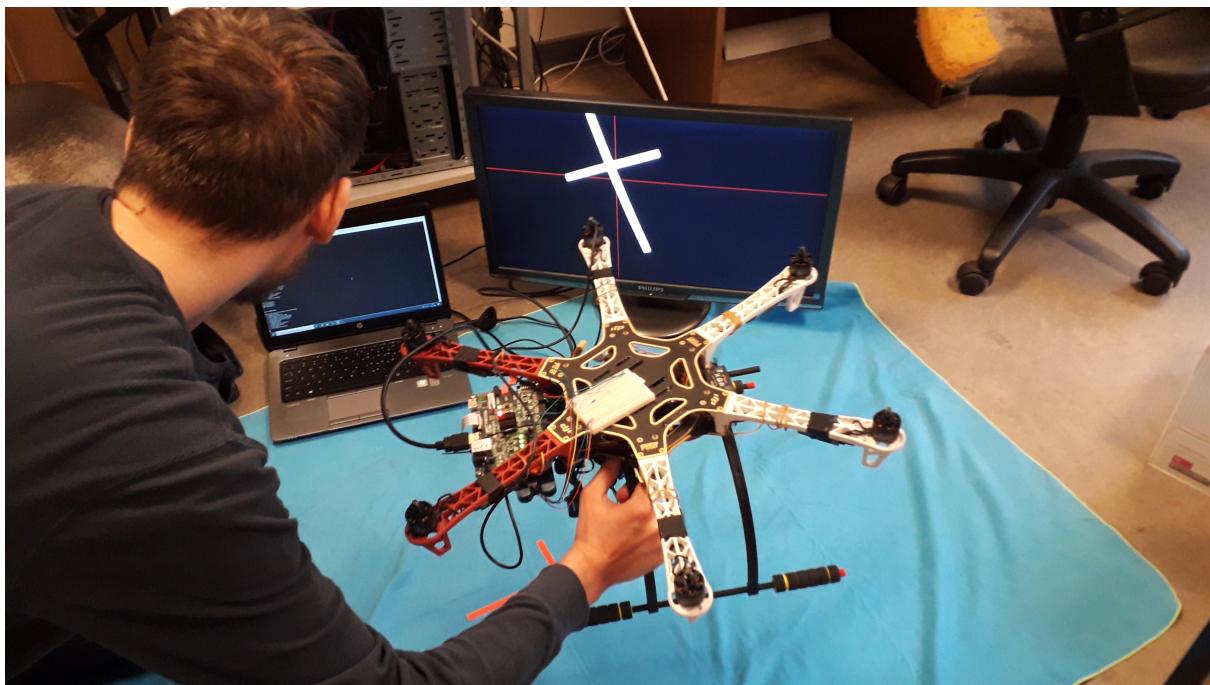
5.2. Test reakcji autopilota na zadawane komendy

Test polegał na wysyłaniu do autopilota komend z układu ZYBO Z7-20. Podczas eksperymentu śmięta drona były zdjęte. Przebywając w budynku, możliwe było wysłanie komendy uzbrojenia silników w trybie Stabilize. Jak opisano w sekcji 3.4, nie jest to jednak tryb pozwalający na wysyłanie komend ruchu. Aby możliwe było kierowanie dronem, należało przeprowadzić eksperiment na zewnątrz, w celu utrzymania dostatecznie silnego sygnału GPS. Po uaktywnieniu trybu Guided wysłano kolejno komendy uzbrojenia silników, startu i lotu z określona prędkością. Silniki reagowały na wysyłane rozkazy. Po eksperymencie można było wysnuć wniosek o poprawnej konfiguracji komunikacji pomiędzy układem ZYBO, a sterownikiem Pixhawk.

5.3. Test sterowania

Test ten polegał na ręcznym wykonywaniu komend pochodzących z systemu procesorowego. Zaprojektowano 3 fazy lotu: wznoszenie na określoną wysokość, kierowanie drona w stronę znacznika, lądowanie. Z powodu niedokładności ręcznego sterowania dronem zrezygnowano z zadawania prędkości wynikającej z regulacji PID. Ograniczono się do wysyłania komend: „w górę”, „w lewo”, „do przodu” itp. Postarano się, aby podłoże miało kolor niebieski, kontrastujący z czerwoną barwą znacznika. Zdjęcie z eksperymentu przedstawiono na Rys. 5.5.

Test wykazał prawidłowe działanie laserowego czujnika odległości. Poprawnie podawane były również



Rys. 5.5. Zdjęcie z eksperymentu. Układ ZYBO Z7-20 zamontowano z przodu plat-formy. Poniżej umieszczono kamerę i Lidar. Na górze znajduje się płytka prototypowa do łatwego łączenia komponentów.

komendy dotyczące zmiany pozycji drona. Na uwagę zasługuje jednak dość mały kąt widzenia kamery PCAM 5C, zwłaszcza w osi pionowej. Dokładniej zbadano tę kwestię i uzyskano wynik 38° dla osi pionowej oraz 76° dla poziomej. Oznacza to, że z wysokości 1 m dron był w stanie wykryć obiekty w odległości 34 cm z przodu i z tyłu, oraz 78 cm po bokach.

6. Podsumowanie i kierunki dalszych prac

W zrealizowanym projekcie przedstawiono sprzętowo-programową realizację systemu wspomagającego autonomiczne lądowanie drona. System zrealizowano w układzie ZYBO Z7-20, którego część rekonfigurowalna pozwoliła na szybkie przetwarzanie obrazu, natomiast system procesorowy umożliwił sprawną realizację algorytmu sterowania i wysyłanie komend do sterownika drona. Implementacja systemu wizyjnego w części rekonfigurowalnej przyspieszyła przetwarzanie obrazu, jednak utrudniła wdrażanie niektórych modułów z powodu braku zasobów.

Z tego względu zaimplementowano dwie wersje systemu wizyjnego: z indeksacją jednoprzebiegową i identyfikacją znacznika na podstawie koloru i współczynnika kształtu, oraz z wyznaczaniem środka ciężkości i identyfikacją markera przy użyciu jego barwy. Identyfikację na podstawie koloru przy niewielkiej zależności od oświetlenia umożliwiła binaryzacja w przestrzeni barw YCbCr, w oparciu o składowe Cb i Cr. Zaprojektowano również taki znacznik, aby ułatwić jego wykrycie. Zaimplementowanie modułów mediany i otwarcia pozwoliło na filtrację obrazu.

Wprowadzenie do układu sygnału z Lidaru umożliwiałło zbieranie dokładnej informacji o wysokości.

Zaimplementowano regulację PID, której celem jest ustawienie drona nad znacznikiem. Eksperymenty pokazały możliwość wykonania lotu testowego składającego się ze startu, regulacji położenia i lądowania w wyznaczonym miejscu. Przeprowadzone testy dowiodły również możliwości wysyłania do autopilota komend, między innymi dotyczących zmiany kierunku i szybkości lotu. W tej sytuacji celem powinno być wykonanie lotu testowego (start - regulacja położenia - lądowanie) przy sterowaniu silnikami z autopilota. Etapem pośrednim powinno być przeprowadzenie testu pozwalającego ustalić, jaką najmniejszą prędkość akceptuje autopilot, jaka największa prędkość powoduje ruch drona bez wyraźnego przechylenia i pochylenia, oraz z jaką częstotliwością mogą być wysyłane komendy. Te informacje pozwoliłyby na dobór nastaw regulatora. W przypadku niemożliwości realizacji komend z odpowiednią częstotliwością rozważyć można inną koncepcję: na podstawie wysokości i aktualnej pozycji znacznika na obrazie można byłoby wyznaczać przesunięcie, które doprowadziłoby dron ponad znacznik.

Powyższe działania możliwe są przy zastosowaniu komponentów użytych w projekcie. Użycie kamery o większym kącie widzenia pozwoliłoby jednak na detekcję lądowiska z większej odległości.

Innym kierunkiem w dalszych pracach może być próba wykonania lądowania na poruszającej się platformie. Analiza literatury naukowej dostarczyła informacji co do niezbędnych działań, jakie musi wykonywać taki system. Są to: predykcja ruchu lądowiska, generacja trajektorii pozwalających dotrzeć do celu i wybór jednej z dopuszczalnych.

A. Spis zawartości płyty CD

Na dołączonej do pracy płycie CD znajdują się następujące foldery:

- Matlab – zawiera pliki modeli programowych, testów, obróbki obrazów oraz zdjęcia niezbędne do ich uruchomienia,
- Vivado – zawiera projekt główny oraz testowy, wykorzystywany do komunikacji z autopilotem.

Bibliografia

- [1] <http://go.skyward.io/rs/902-SIU-382/images/2018StateofDrones.pdf>. Dostęp: 2020-01-18.
- [2] <https://spectrum.ieee.org/aerospace/aviation/us-commercial-drone-deliveries-will-finally-be-a-thing-in-2020>. Dostęp: 2020-01-18.
- [3] S. Lange, N. Sunderhauf i P. Protzel. „A vision based onboard approach for landing and position control of an autonomous multirotor UAV in GPS-denied environments”. W: *2009 International Conference on Advanced Robotics*. 2009, s. 1–6.
- [4] Y. Fan, S. Haiqing i W. Hong. „A Vision-Based Algorithm for Landing Unmanned Aerial Vehicles”. W: *2008 International Conference on Computer Science and Software Engineering*. T. 1. 2008, s. 993–996. DOI: [10.1109/CSSE.2008.309](https://doi.org/10.1109/CSSE.2008.309).
- [5] A. Price i in. „Real time object detection for an unmanned aerial vehicle using an FPGA based vision system”. W: *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006*. 2006, s. 2854–2859. DOI: [10.1109/ROBOT.2006.1642134](https://doi.org/10.1109/ROBOT.2006.1642134).
- [6] D. Falanga i in. „Vision-based autonomous quadrotor landing on a moving platform”. W: *2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. 2017, s. 200–207. DOI: [10.1109/SSRR.2017.8088164](https://doi.org/10.1109/SSRR.2017.8088164).
- [7] Y. Fan, S. Haiqing i W. Hong. „A Vision-Based Algorithm for Landing Unmanned Aerial Vehicles”. W: *2008 International Conference on Computer Science and Software Engineering*. T. 1. 2008, s. 993–996. DOI: [10.1109/CSSE.2008.309](https://doi.org/10.1109/CSSE.2008.309).
- [8] V. Sudevan, A. Shukla i H. Karki. „Vision based autonomous landing of an Unmanned Aerial Vehicle on a stationary target”. W: *2017 17th International Conference on Control, Automation and Systems (ICCAS)*. 2017, s. 362–367. DOI: [10.23919/ICCAS.2017.8204466](https://doi.org/10.23919/ICCAS.2017.8204466).
- [9] <https://reference.digilentinc.com/reference/programmable-logic/zybo-z7/reference-manual>. Dostęp: 2020-01-18.
- [10] M. Mach. „Wbudowany system wizyjny do śledzenia obiektów dla potrzeb nawigacji bezzałogowego statku powietrznego (UAV)”. Praca magisterska. AGH, 2017.
- [11] <https://ardupilot.org/copter/docs/flight-modes.html>. Dostęp: 2020-01-18.
- [12] <https://ardupilot.org/dev/docs/mavlink-basics.html>. Dostęp: 2020-01-21.

- [13] <https://www.sparkfun.com>. Dostęp: 2020-01-31.
- [14] http://elm-chan.org/fsw/ff/00index_e.html. Dostęp: 2019-11-03.
- [15] <https://reference.digilentinc.com/learn/programmable-logic/tutorials/zynq-z7-pcam-5c-demo/start>. Dostęp: 2019-11-03.
- [16] Abdul Malik i in. „Real-time Component Labelling with Centre of Gravity Calculation on FPGA”. W: (sty. 2011).
- [17] https://github.com/mavlink/c_library_v1/tree/master/common. Dostęp: 2020-01-22.
- [18] <http://www.algorytm.org/modele-barw/model-rgb.html>. Dostęp: 2020-01-20.
- [19] <https://www.youtube.com/watch?v=3dET-EoIMM8>. Dostęp: 2020-01-20.