Politechnika
Śląska

PROJEKT INŻYNIERSKI

Robotyczny system sortowania klocków

Jakub PAJĄK
Nr albumu: 300566

Kierunek: Automatyka i Robotyka
Specjalność: Technologie Informacyjne w Automatyce i Robotyce

PROWADZĄCY PRACĘ
dr inż. Krzysztof Jaskot
KATEDRA Automatyki i Robotyki
Wydział Automatyki, Elektroniki i Informatyki

Gliwice 2024

Tytuł pracy

Robotyczny system sortowania klocków

Streszczenie

Celem projektu jest wykonanie autonomicznej platformy jezdnej, mającej na celu bezobsługowe sortowanie klocków spadających w określone miejsce z taśmociągu lub innego podajnika. Robot korzystając z możliwości wizji komputerowej będzie rozpoznawać kolor aktualnego klocka, a następnie na podstawie określonego koloru będzie wybierać trasę do odpowiedniego pojemnika. Domyślnie rozpoznawane będą trzy kolory (czerwony, zielony, niebieski) oraz trzy pojemniki, odpowiednie dla każdego koloru. Wymagania: znajomość programowania układów SBC (Raspberry Pi), Python, OpenCV

Słowa kluczowe

Robot mobilny, analiza wizyjna, Programowanie mikrokontrolerów

Thesis title

Robotic brick sorting system

Abstract

The goal of this project is to create an autonomous mobile platform designed for unattended sorting of blocks falling into a designated area from a conveyor belt or another feeder. The robot will use computer vision to recognize the color of the current block and, based on the identified color, will choose the path to the appropriate container. By default, three colors (red, green, blue) will be recognized, and there will be three containers, each corresponding to one of the colors. Requirements: knowledge of SBC programming (Raspberry Pi), Python, OpenCV

Key words

Mobile robot, visual analysis, microcontroller programming

Spis treści

Rozdział 1

Wstęp

Niniejsza praca inżynierska koncentruje się na zaprojektowaniu i budowie autonomicznej platformy jezdnej, której zadaniem jest sortowanie klocków na podstawie ich koloru. W dobie rozwijającej się automatyzacji i robotyzacji coraz większą rolę odgrywają systemy zdolne do autonomicznego wykonywania zadań, które wcześniej wymagały udziału człowieka. Projekt ten wkomponowuje się w ten trend, mając na celu zautomatyzowanie procesu sortowania elementów w środowisku przemysłowym lub magazynowym. W tym rozdziale zostaną omówione kluczowe aspekty projektu, w tym wprowadzenie w problem, osadzenie go w odpowiednim kontekście naukowym oraz zakres i cele pracy.

1.1 Wprowadzenie w problem

Problem, który stara się rozwiązać niniejszy projekt, dotyczy automatyzacji procesów sortowania w różnych środowiskach przemysłowych, takich jak linie produkcyjne czy magazyny. W tradycyjnym podejściu sortowanie odbywa się ręcznie lub z wykorzystaniem prostych mechanicznych sorterów, które często są mało elastyczne i kosztowne w utrzymaniu. W odpowiedzi na to wyzwanie, projekt ten przewiduje stworzenie autonomicznego robota mobilnego, wykorzystującego kamerę i algorytmy wizji komputerowej do rozpoznawania kolorów klocków oraz następnie do przewiezienia ich do odpowiednich pojemników.

Takie rozwiązanie jest szczególnie przydatne w sytuacjach, gdzie istnieje potrzeba dynamicznej zmiany parametrów sortowania lub tam, gdzie wymagana jest szybka reakcja na zmiany w procesie produkcyjnym. Automatyzacja pozwala również na zmniejszenie kosztów operacyjnych, zwiększenie efektywności oraz redukcję błędów ludzkich w procesie sortowania.

1.2 Osadzenie problemu w dziedzinie

Tematyka projektu wpisuje się w kilka dynamicznie rozwijających się dziedzin, takich jak robotyka mobilna, automatyzacja przemysłowa oraz wizja komputerowa. Roboty mobilne, zdolne do samodzielnej nawigacji oraz wykonywania skomplikowanych zadań, znajdują coraz szersze zastosowanie w przemyśle, logistyce, a także w gospodarstwach domowych. W kontekście niniejszego projektu, kluczową rolę odgrywa zastosowanie odometrii oraz systemów napędowych, które pozwalają na precyzyjną kontrolę ruchu robota.

Wizja komputerowa, z kolei, umożliwia rozpoznawanie obiektów na podstawie ich cech wizualnych, takich jak kolor czy kształt. Technologia ta, oparta na bibliotekach takich jak OpenCV, stała się dostępna nawet dla małych systemów wbudowanych, takich jak Raspberry Pi. W niniejszym projekcie robot wykorzystuje kamerę do rejestrowania obrazów klocków, które następnie są analizowane w czasie rzeczywistym w celu określenia koloru, a na tej podstawie podejmowana jest decyzja gdzie analizowany klocek powinien zostać przetransportowany.

1.3 Cel pracy

Głównym celem projektu jest zaprojektowanie i zbudowanie autonomicznej platformy jezdnej, która potrafi sortować klocki na podstawie ich koloru. Robot będzie wykorzystywał kamerę oraz algorytmy wizji komputerowej do identyfikacji kolorów takich jak czerwony, zielony oraz niebieski, a następnie będzie odpowiednio przekierowywał klocki do odpowiednich pojemników. Proces ten ma odbywać się w pełni automatycznie, bez potrzeby ingerencji człowieka.

Dodatkowym celem jest stworzenie systemu kontroli, który pozwoli na bieżąco monitorować pracę robota oraz ewentualnie wprowadzać modyfikacje w jego działaniu. System powinien być elastyczny, umożliwiający łatwe rozszerzenie o dodatkowe funkcje, takie jak rozpoznawanie większej liczby kolorów lub innych cech klocków.

1.4 Zakres pracy

Projekt obejmuje szeroki zakres działań, zarówno w zakresie sprzętowym, jak i programistycznym. Poniżej przedstawiono główne etapy realizacji:

- Zbudowanie fizycznej platformy jezdnej, w tym konstrukcji mechanicznej oraz systemu napędowego z enkoderami,
- Implementacja systemu rozpoznawania kolorów za pomocą kamery oraz algorytmów wizji komputerowej z użyciem Raspberry Pi i biblioteki OpenCV,

- Opracowanie algorytmu podejmowania decyzji na podstawie rozpoznanego koloru i przekierowania klocka do odpowiedniego pojemnika,
- Testowanie i optymalizacja działania systemu, aby zapewnić jego poprawne funkcjonowanie w rzeczywistych warunkach, takich jak linie produkcyjne czy magazyny.

Zakres projektu przewiduje także możliwość dalszej rozbudowy o nowe funkcjonalności, co zwiększa elastyczność i potencjalne zastosowanie platformy w innych dziedzinach, takich jak automatyczne sortowanie elementów na podstawie kształtu czy materiału.

1.5 Zwięzła charakterystyka rozdziałów

Struktura pracy została podzielona na następujące rozdziały:

- **Rozdział 1: Wstęp** – Przedstawienie problemu, celów oraz zakresu projektu.
- **Rozdział 2: Analiza tematu** – Przegląd dostępnych rozwiązań, technik oraz technologii stosowanych w podobnych systemach, a także omówienie aktualnego stanu wiedzy w tej dziedzinie.
- **Rozdział 3: Wymagania i narzędzia** – Opis wymagań projektowych oraz narzędzi, takich jak Raspberry Pi, OpenCV oraz Python, które zostaną użyte do realizacji projektu.
- **Rozdział 4: Specyfikacja zewnętrzna** – Opis funkcjonalności systemu z perspektywy użytkownika, w tym interfejsu oraz interakcji z otoczeniem.
- **Rozdział 5: Specyfikacja wewnętrzna** – Szczegółowa analiza architektury wewnętrznej systemu, w tym struktury oprogramowania i integracji z komponentami sprzętowymi.
- **Rozdział 6: Weryfikacja i walidacja** – Opis przeprowadzonych testów, ocena efektywności systemu oraz zgodności z założeniami projektowymi.
- **Rozdział 7: Podsumowanie i wnioski** – Zakończenie projektu, omówienie osiągniętych celów, a także wskazanie możliwych kierunków rozwoju systemu.

1.6 Wkład autora

Projekt został w pełni zrealizowany przez autora. Autor odpowiada za wszystkie etapy pracy, począwszy od projektowania i budowy platformy, poprzez implementację algorytmów rozpoznawania kolorów i sterowania ruchem robota, aż po testowanie i optymalizację systemu. Ponadto, autor zaprojektował system kontrolny, który pozwala na monitorowanie pracy robota w czasie rzeczywistym oraz umożliwia wprowadzenie ewentualnych korekt.

Rozdział 2

Analiza tematu

W rozdziale tym zostanie przeanalizowany temat projektu, z uwzględnieniem aktualnego stanu wiedzy oraz znanych rozwiązań dotyczących automatycznych systemów sortowania za pomocą wizji komputerowej.

2.1 Sformułowanie problemu

Głównym problemem rozwiązywanym w tym projekcie jest automatyzacja procesu sortowania obiektów według ich cech wizualnych, w tym przypadku koloru. Tradycyjne metody sortowania wymagają interwencji człowieka lub stosowania mechanicznych sorterów, które często są mniej elastyczne i mniej precyzyjne w identyfikacji złożonych cech. Celem projektu jest wykorzystanie technologii wizji komputerowej, aby umożliwić robotowi autonomiczne rozpoznawanie i sortowanie klocków według koloru.

2.2 Osadzenie tematu w kontekście aktualnego stanu wiedzy (*state of the art*)

Temat projektu osadza się w dziedzinach robotyki mobilnej oraz wizji komputerowej, które rozwijają się bardzo dynamicznie w ostatnich latach. Rozwiązania oparte na wizji komputerowej, zwłaszcza w połączeniu z platformami SBC (Single-Board Computer), takimi jak Raspberry Pi, stają się coraz bardziej popularne w automatyzacji procesów przemysłowych i logistyce. Wykorzystanie takich narzędzi jak OpenCV w projektach robotycznych pozwala na zwiększenie elastyczności i precyzji systemów sortujących.

2.3 Studia literaturowe

Podczas prac nad poszczególnym elementami projektu istoną rolę odgrywała analiza dostępnej literatury naukowej. Poniżej zajduje się spis artykułów, książek, dokumentacji

oraz stron internetowych, których treść okazała się przydatna podczas pracy.

2.3.1 Opracowanie systemu wizyjnego

Projektowy system wizyjny oparty został na dedykowanej kamerze dla platformy Raspberry Pi oraz na popularnej bibliotece OpenCV, co pozwala na efektywną analizę obrazu i wykrywanie obiektów w czasie rzeczywistym. W tym celu kluczowa okazała się dokumentacja biblioteki Libcamera2 [4], oferująca szczegółowe informacje na temat konfiguracji kamery, procedur inicjalizacji, oraz wykorzystania łącza CSI (Camera Serial Interface) do przesyłu obrazu. Dokumentacja dostarczona przez Raspberry Pi Ltd zawiera obszerne wytyczne dotyczące ustawień obrazu, obsługi błędów oraz optymalizacji parametrów obrazu pod kątem specyficznych wymagań, co stanowiło cenną pomoc podczas implementacji i konfiguracji systemu wizyjnego.

W celu bardziej zaawansowanego przetwarzania obrazu wykorzystano bibliotekę OpenCV. Dla zrozumienia jej funkcjonalności, zwłaszcza w kontekście realizacji projektu, cennym źródłem była literatura specjalistyczna, m.in. książka [2], która stanowi kompleksowy przegląd możliwości tej biblioteki. Książka ta obejmuje szeroką gamę algorytmów i metod przetwarzania obrazu, z których jedynie wybrane fragmenty — jak detekcja krawędzi za pomocą algorytmu Canny'ego oraz analiza barw — zostały zaimplementowane w niniejszym projekcie. Wybór tych technik wynikał z ich efektywności oraz dostosowania do wymagań projektu, jakim jest klasyfikacja obiektów na podstawie koloru.

Przy opracowywaniu metody rozpoznawania kolorów wykorzystano również wyniki badań z publikacji [1], opisującej różnorodne metody i algorytmy analizy wizyjnej, stosowane w procesie klasyfikacji obiektów na podstawie ich cech kolorystycznych. Dzięki znajomości podstaw teoretycznych zaprezentowanych w tym dokumencie, wybór odpowiednich technik przetwarzania obrazu oraz implementacja algorytmu klasyfikacji kolorów przebiegały bardziej efektywnie.

Kolejnym istotnym krokiem była analiza metod detekcji krawędzi, przy czym szczególna uwaga została poświęcona algorytmom i funkcjom dostępnym w bibliotece OpenCV. Jednym z analizowanych dokumentów był artykuł [8], który szczegółowo omawia różne techniki detekcji krawędzi. Po dokonaniu analizy, jako metoda najbardziej odpowiednia dla projektu została wybrana detekcja krawędzi metodą Canny'ego, ze względu na jej skuteczność w kontekście warunków oświetleniowych oraz wysoką precyzję w identyfikacji krawędzi obiektów.

2.3.2 Opracowanie systemu kontroli silników

System kontroli silników dla platformy mobilnej oparto o napęd różnicowy, który wybrano ze względu na jego precyzję oraz stosunkowo prostą implementację algorytmów sterujących. Napęd różnicowy umożliwia manewrowanie robotem poprzez odpowiednie

sterowanie prędkością oraz kierunkiem obrotu poszczególnych kół, co jest szczególnie istotne w zastosowaniach wymagających dokładności na ograniczonej przestrzeni. Przy opracowywaniu systemu sterowania teoretyczną podstawę stanowiły wyniki badań opisane w artykule [5], który szczegółowo omawia technikę regulacji PID (proporcjonalno-całkująco-różniczkującą) w zastosowaniu do robotów mobilnych z napędem różnicowym. Artykuł ten dostarcza kompleksowej wiedzy na temat regulacji PID w robotyce, co pozwoliło na łatwiejsze zaimplementowanie programu do kontroli prędkości i kierunku działania poszczególnych silników.

Istotnym elementem systemu kontroli była również implementacja systemu enkoderów, które monitorują ruch robota i pozwalają na dokładną analizę przebytej drogi oraz prędkości. W projekcie dostępne były dwie metody odczytywania impulsów generowanych przez enkodery: poprzez przerwania generowane przy każdym impulsie oraz przy użyciu wbudowanego licznika mikrokontrolera. Wybór metody wiązał się z koniecznością dokładnego zrozumienia zarówno korzyści, jak i ograniczeń każdej z nich.

Dzięki analizie dokumentacji mikrokontrolera ATMega328 [3] oraz literatury opisującej przerwania [7], możliwe było dogłębne zrozumienie mechanizmów przerwań, co stanowiło podstawę do implementacji wybranego algorytmu. Dokumentacja mikrokontrolera dostarcza szczegółowych informacji na temat możliwości wykorzystywania przerwań i liczników, co pozwala na optymalizację algorytmu kontroli silników oraz dostosowanie jego parametrów do wymogów projektu. Analiza tych źródeł umożliwiła autorowi wybór optymalnego rozwiązania, które zwiększa precyzję pomiaru pozycji i prędkości przy jednoczesnym zachowaniu wydajności systemu.

Rozdział 3

Wymagania i narzędzia

W rozdziale tym przedstawiono szczegółowy opis wymagań funkcjonalnych i niefunkcjonalnych, które określają, jakie zadania system musi spełniać oraz jakie kryteria jakościowe są wymagane do zapewnienia prawidłowego działania platformy mobilnej do sortowania klocków według koloru. Omówiono również narzędzia i metody, które wspierają realizację projektu, w tym wykorzystywane technologie, frameworki oraz metodologię pracy nad projektem.

3.1 Wymagania funkcjonalne i niefunkcjonalne

Wymagania funkcjonalne odnoszą się do kluczowych funkcji systemu, które muszą zostać spełnione, aby system mógł realizować swoje podstawowe zadania. W przypadku autonomicznej platformy mobilnej do sortowania klocków funkcje te obejmują między innymi:

- Rozpoznawanie kolorów klocków – system musi identyfikować kolory klocków przy pomocy kamery i odpowiednich algorytmów przetwarzania obrazu (czerwony, zielony, niebieski).
- Przekierowywanie klocków – po rozpoznaniu koloru, system kieruje klocek do odpowiedniego pojemnika.
- Samodzielne poruszanie się – robot powinien nawigować w wyznaczonej przestrzeni zgodnie z zaprogramowanymi trasami.

Wymagania niefunkcjonalne obejmują kryteria jakościowe, które determinują, jak system ma działać. Są to:

- Niezawodność – system powinien pracować nieprzerwanie przez określony czas bez błędów.

- Szybkość przetwarzania obrazu – analiza obrazu i detekcja koloru muszą odbywać się na bieżąco, co jest szczególnie istotne w przypadku szybkich linii produkcyjnych.

3.2 Przypadki użycia i diagramy UML

3.3 Opis narzędzi i metod

W projekcie wykorzystano różnorodne narzędzia wspomagające rozwój systemu, między innymi:

- Raspberry Pi – służy jako główny komputer zarządzający systemem, wyposażony w odpowiednie moduły do obsługi kamery oraz komunikacji.
- Arduino UNO R3 - mikrokontroler pełniący rolę kontrolera napędów oraz enkoderów.
- OpenCV – biblioteka do przetwarzania obrazu.
- Python - główny język programowania obsługujący system kontroli robota oraz analizę wyzyjną.
- Libcamera2 – biblioteka wspomagająca obsługę kamer podłączonych przez interfejs CSI, co umożliwia komunikację między Raspberry Pi a kamerą.
- Raspbian - system operacyjny układu SBC Raspberry Pi. Został on wykorzystany ze względu na kompatybilność z komputerem.
- Visual Studio Code - środowisko rozbudowanego edytora tekstu, pełniącego rolę programu do pisania aplikacji oraz zdalnego łączenia się z komputerem poprzez protokuł SSH.
- Arduino IDE - zaintegrowane środowisko programistyczne dedykowane do programowania mikrokontrolerów rodziny ATMega.

3.4 Metodyka pracy nad projektowaniem i implementacją

Realizacja projektu przebiegała zgodnie z metodyką iteracyjno-przyrostową, co pozwalało na stopniowe dodawanie funkcjonalności i testowanie każdego etapu w rzeczywistych warunkach pracy systemu. Na każdym etapie projektowania, implementacji i testowania zbierano informacje o działaniu systemu i, jeśli zachodziła potrzeba, dostosowywano parametry systemu lub optymalizowano kod.

Rozdział 4

Specyfikacja zewnętrzna

W poniższym rozdziale szczegółowo opisano wymagania sprzętowe i programowe, sposób instalacji, aktywacji oraz wszystkie inne informacje konieczne dla użytkownika, które są niezbędne do skutecznego działania platformy mobilnej.

4.1 Wymagania sprzętowe i programowe

Podstawowym założeniem było wykorzystanie mikrokomputera Raspberry Pi, który stanowi centralny element systemu. Ze względu na jego kompaktowe rozmiary, niską wagę oraz odpowiednią moc obliczeniową, Raspberry Pi idealnie wpisuje się w potrzeby projektu.

Wymagania sprzętowe

- Mikrokomputer Raspberry Pi – wybrany model Raspberry Pi 4 Model B, który charakteryzuje się czterordzeniowym procesorem, co zapewnia wystarczającą moc obliczeniową do przetwarzania obrazu oraz zarządzania algorytmami sterowania.
- Kamera – zastosowano dedykowaną kamerę kompatybilną z Raspberry Pi, która umożliwia przechwytywanie obrazów w wysokiej rozdzielczości. Kamera jest podłączona przez interfejs CSI (ang. *Camera Serial Interface*), co zapewnia niskie opóźnienia i wysoką jakość obrazu.
- Silniki i napęd różnicowy – do napędu platformy wykorzystano silniki prądu stałego z enkoderami, które umożliwiają precyzyjne sterowanie ruchem robota oraz monitorowanie jego pozycji.
- Zasilanie – system zasilany jest z pakietu czterech ogniw litowo-jonowych, dostarczających wysatraczjącą ilość napięcia oraz umożliwiających pracę przez odpowieni czas.

- Chwytak - wykorzystano chwytak zasilany jednym serwomechanizmem, wystarczający do realizacji zadań.

Wymagania programowe

- Python - język programowania bardzo dobrze sprawdzający się przy wykorzystaniu komputera Raspberry Pi.
- OpenCV - biblioteka umożliwiająca bardzo dużą ilość operacji na obrazie cyfrowym.
- Libcamera2 - biblioteka konieczna do poprawnego wykorzystywania kamery wraz z OpenCV na komputerze Raspberry Pi.
- Raspbian OS - system operacyjny, będący specjalną implementacją systemu Linux, posiadającą wiele wbudwanych biliotek oraz innych udogodnień wspomagających rozwój oprogramowania na komputerze Raspberry Pi.
- git - system kontroli wersji, wykorzystany w celu spełnienia standardów rozwoju oprogramowania.

4.2 Sposób instalacji

4.2.1 Instalacja systemu operacyjnego

Pierwszym krokiem w procesie instalacji systemu kontroli robota jest zainstalowanie systemu operacyjnego Raspbian na jednostce centralnej mikrokomputera Raspberry Pi. W celu skutecznej realizacji tego zadania zaleca się skorzystanie z oprogramowania dostarczanego przez firmę Raspberry Pi Ltd, które umożliwia utworzenie przenośnego dysku uruchomieniowego. Dysk ten może przyjąć postać karty microSD lub przenośnego dysku USB. Istnieje również szereg innych programów, które mogą spełniać tę funkcję; jednakże najbardziej rekomendowanym rozwiązaniem jest użycie aplikacji *Raspberry Pi Imager*. Program ten, oprócz możliwości utworzenia dysku startowego, oferuje opcję pobrania odpowiedniej wersji systemu Raspbian lub innego dystrybucji systemu Linux, co czyni go bardzo elastycznym narzędziem, dopasowanym do potrzeb użytkownika.

Zaleca się korzystanie z 64-bitowej wersji systemu Raspbian, zapewnia to lepszą wydajność i większą kompatybilność z aplikacjami.

Po utworzeniu nośnika rozruchowego, należy umieścić kartę microSD lub przenośny dysk USB w odpowiednim gnieździe mikrokomputera Raspberry Pi. Następnie, po podłączeniu urządzenia do źródła zasilania, rozpocznie się proces instalacji systemu operacyjnego. W przypadku, gdy obraz nie jest wyświetlany po podłączeniu mikrokomputera do monitora za pomocą kabla microHDMI, użytkownik powinien zweryfikować ustawienia pliku konfiguracyjnego *config.txt* znajdującego się na nośniku startowym. W szczególności,

aby dostosować rozdzielczość wyświetlacza, należy dodać lub zmodyfikować odpowiednie linie w tym pliku. Przykładowy fragment kodu, który można umieścić w pliku *config.txt*, aby ustawić rozdzielczość na 1920x1080, wygląda następująco:

```
1 hdmi_group=1
2 hdmi_mode=16
```

Powyższe parametry konfiguracyjne odpowiadają za wybór standardu HDMI oraz określenie trybu rozdzielczości, gdzie wartość 16 oznacza rozdzielczość 1080p. Po wprowadzeniu zmian w pliku konfiguracyjnym, należy zapisać plik i ponownie uruchomić mikrokomputer.

W przypadku problemów z wyświetlaniem obrazu warto również przetestować inny monitor, jeśli jest on dostępny, ponieważ istnieje możliwość, że dany model monitora nie obsługuje określonego formatu sygnału HDMI.

Po instalacji systemu operacyjnego Raspbian na urządzeniu Raspberry Pi, kolejnym istotnym etapem jest instalacja wymaganych pakietów oraz bibliotek, które zapewnią funkcjonalność kluczowych elementów systemu sterowania robota. Wśród podstawowych bibliotek, niezbędnych do poprawnego działania systemu, znajdują się: Python, OpenCV, Libcamera2 oraz RPi.GPIO.

Wszystkie poniższe komendy należy wykonać w terminalu Raspberry Pi, aby pobrać i zainstalować odpowiednie pakiety. Warto zauważyć, że przed przystąpieniem do instalacji zaleca się aktualizację pakietów systemowych.

4.2.2 Aktualizacja systemu

Dobrą praktyką, umożliwiającą uniknięcie wielu błędów wynikających z braku kompatybilności systemu bazującego na jądrze Linux jest wykonywanie aktualizacji oraz uaktualnienia zainstalowanych pakietów poprzez zdalne repozytorium *apt* wykonując poniższe komendy.

```
1 sudo apt update
2 sudo apt upgrade -y
```

4.2.3 Instalacja Python

Python jest podstawowym językiem programowania, w którym napisana została aplikacja sterująca robotem. Raspbian posiada zazwyczaj zainstalowaną wersję Python, jednak można zainstalować lub zaktualizować go do nowszej lub wymaganej wersji przy użyciu następującej komendy. Niektóre biblioteki lub pakiety wymagają konkretnej wersji Python, na przykład w wersji 3.10, podczas gdy najnowsza wykracza poza wersję 3.12. W

związku z powyższym możliwe jest zaistnienie problemu kompatybilności, przez co zaleca się zweryfikowanie, która wersja będzie odpowiednia. W przypadku poniższego projektu wersja 3.10 będzie odpowiednia.

```
sudo apt install -y python3 python3-pip
```

4.2.4 Instalacja OpenCV

OpenCV (ang. *Open Source Computer Vision Library*) to biblioteka służąca do przetwarzania obrazu i analizy wizyjnej. W projekcie OpenCV zostanie wykorzystane m.in. do detekcji koloru klocków. Instalacja OpenCV jest możliwa za pomocą menedżera pakietów pip:

```
pip3 install opencv-python-headless
```

4.2.5 Instalacja Libcamera2

Biblioteka *Libcamera2* jest odpowiedzialna za obsługę kamery, która jest podłączona do złącza CSI Raspberry Pi. Aby zainstalować pakiet Libcamera2, należy wykonać następujące polecenie (więcej informacji można znaleźć w dokumentacji[[4]]):

```
    sudo apt install -y python3-picamera2
```

4.2.6 Instalacja RPi.GPIO

Biblioteka *RPi.GPIO* zapewnia interfejs programistyczny do korzystania z pinów GPIO (ang. *General Purpose Input/Output*) mikrokomputera Raspberry Pi. Zalecana jest instalacja poprzez repozytorium *apt* zamiast *pip*, przez wzgląd na możliwe probkemy z kompatybilnością.

```
    sudo apt-get update
    sudo apt-get -y install python-rpi.gpio
```

Po zakończeniu instalacji wszystkich wymienionych pakietów i bibliotek, system Raspberry Pi jest gotowy do uruchomienia aplikacji kontrolującej robota. Przed przystąpieniem do pracy zaleca się również zweryfikowanie poszczególnych bibliotek.

4.2.7 Weryfikacja instalacji pakietów

Po zakończeniu procesu instalacji pakietów i bibliotek warto przeprowadzić weryfikację, aby upewnić się, że wszystkie komponenty zostały zainstalowane poprawnie i są

gotowe do użycia. Testowanie każdego pakietu z osobna pozwala na szybkie wykrycie ewentualnych problemów i zapewnia, że środowisko pracy jest w pełni skonfigurowane.

Weryfikacja instalacji Python

Aby sprawdzić, czy Python jest poprawnie zainstalowany, w terminalu wpisz:

```
python3 --version
```

Komenda ta powinna wyświetlić zainstalowaną wersję Python, np. `Python 3.x.x`. Jeśli polecenie działa bez błędów, oznacza to, że Python jest zainstalowany poprawnie.

Weryfikacja instalacji OpenCV

Aby upewnić się, że biblioteka OpenCV jest dostępna, można przeprowadzić test bezpośrednio w Python. Uruchom interpreter Python komendą:

```
python3
```

Następnie zaimportuj bibliotekę OpenCV, wpisując:

```
import cv2
print(cv2.__version__)
```

Polecenie to wyświetli wersję OpenCV, np. `4.5.x`. Jeśli import przebiegnie bez błędów, oznacza to, że biblioteka OpenCV działa poprawnie.

Weryfikacja instalacji Libcamera2

Aby sprawdzić poprawność instalacji biblioteki Libcamera2 oraz połączenie kamery, wykonaj testowe przechwycenie obrazu za pomocą poniższej komendy:

```
libcamera-still -o test_image.jpg
```

Jeżeli kamera działa poprawnie, komenda ta zapisze zdjęcie o nazwie `test_image.jpg` w katalogu domyślnym. Otwarcie pliku i weryfikacja jego zawartości potwierdzi działanie kamery i poprawność instalacji.

4.3 Sposób aktywacji

4.4 Kategorie użytkowników

4.5 Sposób obsługi

4.6 Administracja systemem

4.7 Kwestie bezpieczeństwa

4.8 Przykład działania

4.9 Scenariusze korzystania z systemu

- wymagania sprzętowe i programowe
- sposób instalacji
- sposób aktywacji
- kategorie użytkowników
- sposób obsługi
- administracja systemem
- kwestie bezpieczeństwa
- przykład działania
- scenariusze korzystania z systemu (ilustrowane zrzutami z ekranu lub generowanymi dokumentami)

Rozdział 5

[Właściwy dla kierunku – np. Specyfikacja wewnętrzna]

Jeśli „Specyfikacja wewnętrzna”:

- przedstawienie idei
- architektura systemu
- opis struktur danych (i organizacji baz danych)
- komponenty, moduły, biblioteki, przegląd ważniejszych klas (jeśli występują)
- przegląd ważniejszych algorytmów (jeśli występują)
- szczegóły implementacji wybranych fragmentów, zastosowane wzorce projektowe
- diagramy UML

Krótka wstawka kodu w linii tekstu jest możliwa, np. `int a;` (biblioteka `listings`). Dłuższe fragmenty lepiej jest umieszczać jako rysunek, np. kod na rys 5.1, a naprawdę długie fragmenty – w załączniku.

```
1  class test : public basic
2  {
3      public:
4        test (int a);
5        friend std::ostream operator<<(std::ostream & s,
6                                       const test & t);
7      protected:
8        int _a;
9
10 };
```

Rysunek 5.1: Pseudokod w `listings`.

Rozdział 6

Weryfikacja i walidacja

- sposób testowania w ramach pracy (np. odniesienie do modelu V)
- organizacja eksperymentów
- przypadki testowe zakres testowania (pełny/niepełny)
- wykryte i usunięte błędy
- opcjonalnie wyniki badań eksperymentalnych

Tabela 6.1: Nagłówek tabeli jest nad tabelą.

	metoda						
			alg. 3			alg. 4, $\gamma = 2$	
ζ	alg. 1	alg. 2	$\alpha = 1.5$	$\alpha = 2$	$\alpha = 3$	$\beta = 0.1$	$\beta = -0.1$
0	8.3250	1.45305	7.5791	14.8517	20.0028	1.16396	1.1365
5	0.6111	2.27126	6.9952	13.8560	18.6064	1.18659	1.1630
10	11.6126	2.69218	6.2520	12.5202	16.8278	1.23180	1.2045
15	0.5665	2.95046	5.7753	11.4588	15.4837	1.25131	1.2614
20	15.8728	3.07225	5.3071	10.3935	13.8738	1.25307	1.2217
25	0.9791	3.19034	5.4575	9.9533	13.0721	1.27104	1.2640
30	2.0228	3.27474	5.7461	9.7164	12.2637	1.33404	1.3209
35	13.4210	3.36086	6.6735	10.0442	12.0270	1.35385	1.3059
40	13.2226	3.36420	7.7248	10.4495	12.0379	1.34919	1.2768
45	12.8445	3.47436	8.5539	10.8552	12.2773	1.42303	1.4362
50	12.9245	3.58228	9.2702	11.2183	12.3990	1.40922	1.3724

Rozdział 7

Podsumowanie i wnioski

- uzyskane wyniki w świetle postawionych celów i zdefiniowanych wyżej wymagań
- kierunki ewentualnych danych prac (rozbudowa funkcjonalna . . .)
- problemy napotkane w trakcie pracy

Bibliografia

[1] Aditi Bajaj i Jyotsna Sharma. „Computer Vision for Color Detection". W: *International Journal of Innovative Research in Computer Science & Technology (IJIRCST)* (2020), s. 53–59.

[2] Gary Bradski i Adrian Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. United States of America, Sebastopol: O'Reilly Media, 2008. ISBN: 978-0-596-51613-0.

[3] B. Cottenceau. *Introduction to Arduino*.

[4] Raspberry Pi Ltd. *The Picamera2 Library. A libcamera-based Python library for Raspberry Pi cameras*.

[5] Octavian Bologa Mihai Crenganis. „PID CONTROLLER FOR A DIFFERENTIAL STEERING MOBILE PLATFORM". W: *Konferencji ASTR*. 2015, s. 6.

[6] Imię Nazwisko i Imię Nazwisko. *Tytuł strony internetowej*. 2021. URL: `http://gdzies/w/internecie/internet.html` (term. wiz. 30. 09. 2021).

[7] SHRIPAD G. DESAI PRINCE KUMAR. „Interrupts in The Microcontroller". W: *IRE Journals* 4.4 (2021), s. 166–169.

[8] Djemel Ziou i Salvatore Tabbone. „Edge Detection Techniques-An Overvie". W: *&#1085 / Pattern Recognition and Image Analysis: Advances in Mathematical Theory and Applications* 8.4 (1998), s. 537–559.

Dodatki

Spis skrótów i symboli

CSI szeregowy interfejs kamery (ang. *Camera Serial Interface*)

OpenCV otwarta biblioteka wizji komputerowej(ang. *Open Source Computer Vision Library*)

N liczebność zbioru danych

μ stopnień przyleżności do zbioru

$\mathbb{E}$ zbiór krawędzi grafu

$\mathcal{L}$ transformata Laplace'a

Źródła

Jeżeli w pracy konieczne jest umieszczenie długich fragmentów kodu źródłowego, należy je przenieść w to miejsce.

```
1 if (_nClusters < 1)
2     throw std::string ("unknown number of clusters");
3 if (_nIterations < 1 and _epsilon < 0)
4     throw std::string ("You should set a maximal number of
          iteration or minimal difference — epsilon.");
5 if (_nIterations > 0 and _epsilon > 0)
6     throw std::string ("Both number of iterations and minimal
          epsilon set — you should set either number of iterations
          or minimal epsilon.");
```

Lista dodatkowych plików, uzupełniających tekst pracy

W systemie do pracy dołączono dodatkowe pliki zawierające:

- źródła programu,
- dane testowe,
- film pokazujący działanie opracowanego oprogramowania lub zaprojektowanego i wykonanego urządzenia,
- itp.

Spis rysunków

Spis tabel