Projekt

Sterowniki robotów

Założenia projektowe

Robot balansujący piłką K.U.L.A.

Skład grupy:
Michał Trela, 259312
Jan Masłowski, 258962

Termin: wtTN19

Prowadzący:
dr inż. Wojciech Domski

25 kwietnia 2023

Spis treści

1 Opis projektu

Projekt ma na celu stworzenie robota balansującego piłką w jednej osi. Lokalizacja piłki na platformie będzie odbywać się za pomocą systemu wizyjnego a sterowanie wychyleniem platformy poprzez serwomechanizm. Całość projektu oparta będzie na mikrokomputerze Raspberry Pi.

2 Założenia projektowe

2.1 Konfiguracja pinów

Tabela 1: Konfiguracja pinów mikrokontrolera

Numer pinu	PIN	Tryb pracy	Funkcja/etykieta
32	GPIO12	PWM0	SERVO_PWM0
33	GPIO13	PWM1	SERVO_PWM1

2.2 CSI

Do obsługi kamery zostanie wykorzystany zostanie wbudowany interfejs komunikacyjny CSI (Camera Serial Interface). Konfiguracja kamery będzie odbywać się za pomocą API - PiCamera.

Tabela 2: Konfiguracja pinów portu CSI

PIN	Nazwa	Opis
1	GND	Ground
2	CAM_D0_N	MIPI Data Lane 0 Negative
3	CAM_D0_P	MIPI Data Lane 0 Positive
4	GND	Ground
5	CAM_D1_N	MIPI Data Lane 1 Negative
6	CAM_D1_P	MIPI Data Lane 1 Positive
7	GND	Ground
8	CAM_CK_N	MIPI Clock Lane Negative
9	CAM_CK_P	MIPI Clock Lane Positive
10	GND	Ground
11	CAM_IO0	Power Enable
12	CAM_IO1	LED Indicator
13	CAM_SCL	I2C SCL
14	CAM_SDA	I2C SDA
15	CAM_3V3	3.3V Power Input

2.3 Opis konstrukcji

Komputerowy model zostanie wykonany korzystając z technologii CAD. Całość konstrukcji zostanie wykonana za pomocą druku 3D. Konstrukcja będzie składać się z:

- Podstawy w której zostanie umieszczony mikrokomputer Raspberry Pi
- Platformy w postaci płaszczyzny swobodnie poruszającej się w dwóch osiach.
- Uchwytu na kamerę

2.4 Lokalizacja piłki

Lokalizacja piłki będzie odbywać się poprzez analizę obrazu otrzymanego z kamery korzystając z biblioteki OpenCV. Kamera ma widok z góry na platformę. Pozycja piłki będzie określana względem markerów Aruco zaznaczających oba końce platformy. Również za pomocą markerów będzie odbywać się obliczanie aktualnego pochylenia platformy. Przez zmianę pozycji piłki w zadanym czasie będzie wyznaczany kierunek oraz wartość przyśpieszenia.

2.5 Balansowanie piłką

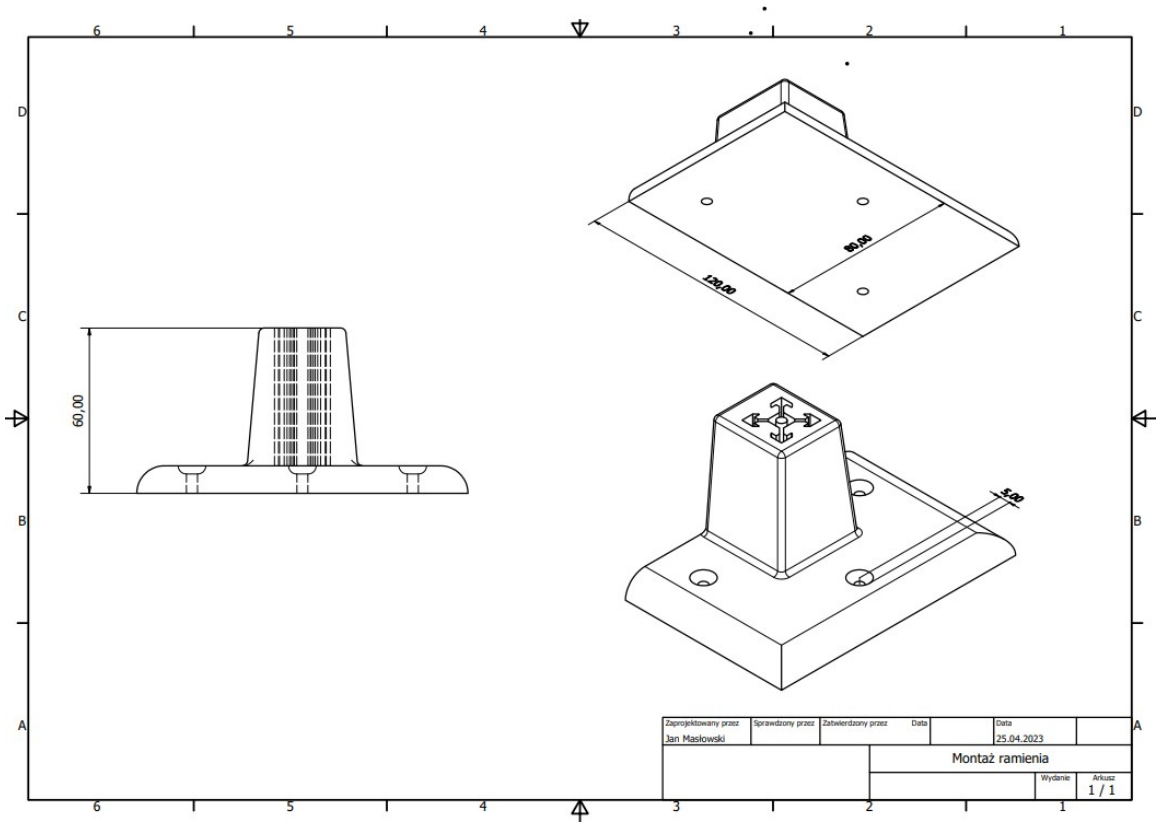
Po wyznaczeniu umiejscowienia piłki na platformie, kierunku i wartości przyśpieszenia, za pomocą regulatora PID na serwomechanizm podawana jest pozycja oraz prędkość. Serwomechanizm sterowany jest metodą PWM.

3 Konstrukcja mechaniczna

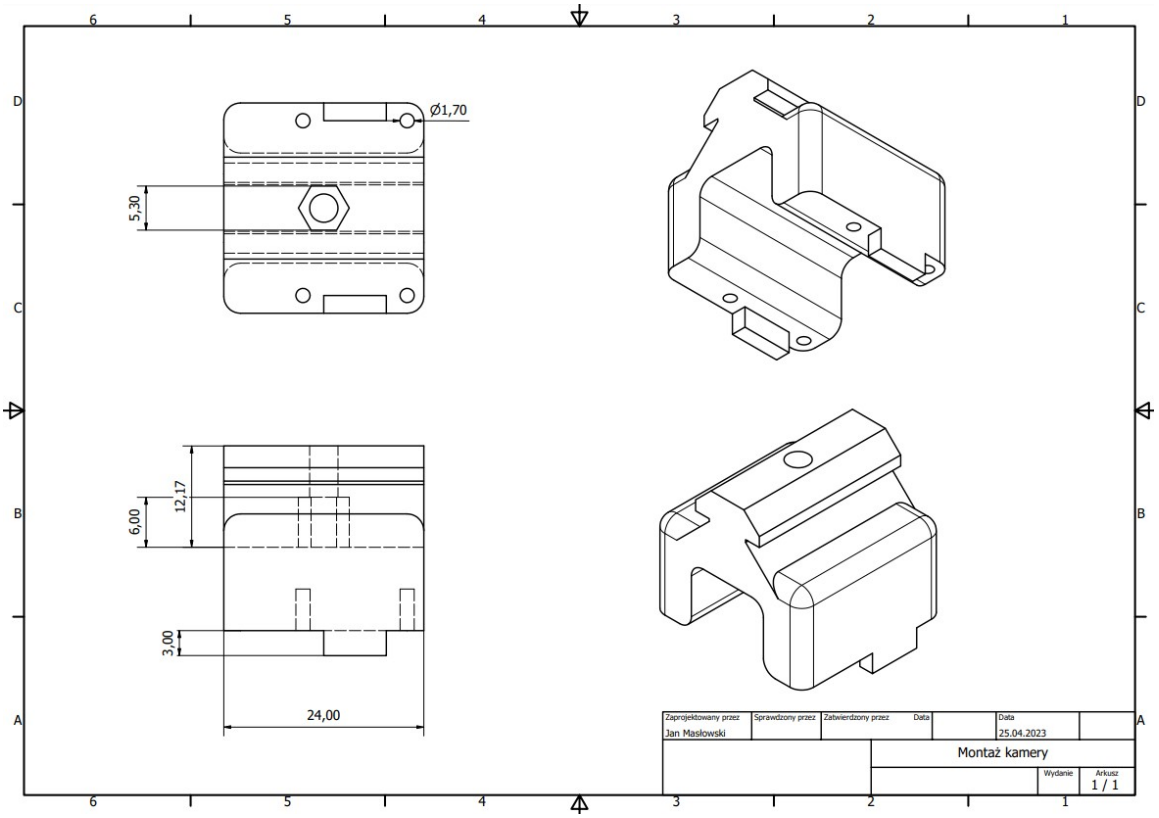
Platforma balansująca składa się z następujących elementów:

- Podstawa ze sklejki o wymiarach 29 x 36.5 x 1.8 cm
- Dwóch profili aluminiowych V2020 o wymiarach 50 i 25 cm
- Łącznika profili aluminiowych
- 11 elementów stworzonych w technologii druku 3D
- Plexi o wymiarach 25 x 25 cm
- Śrub, wkrętów i nakrętek

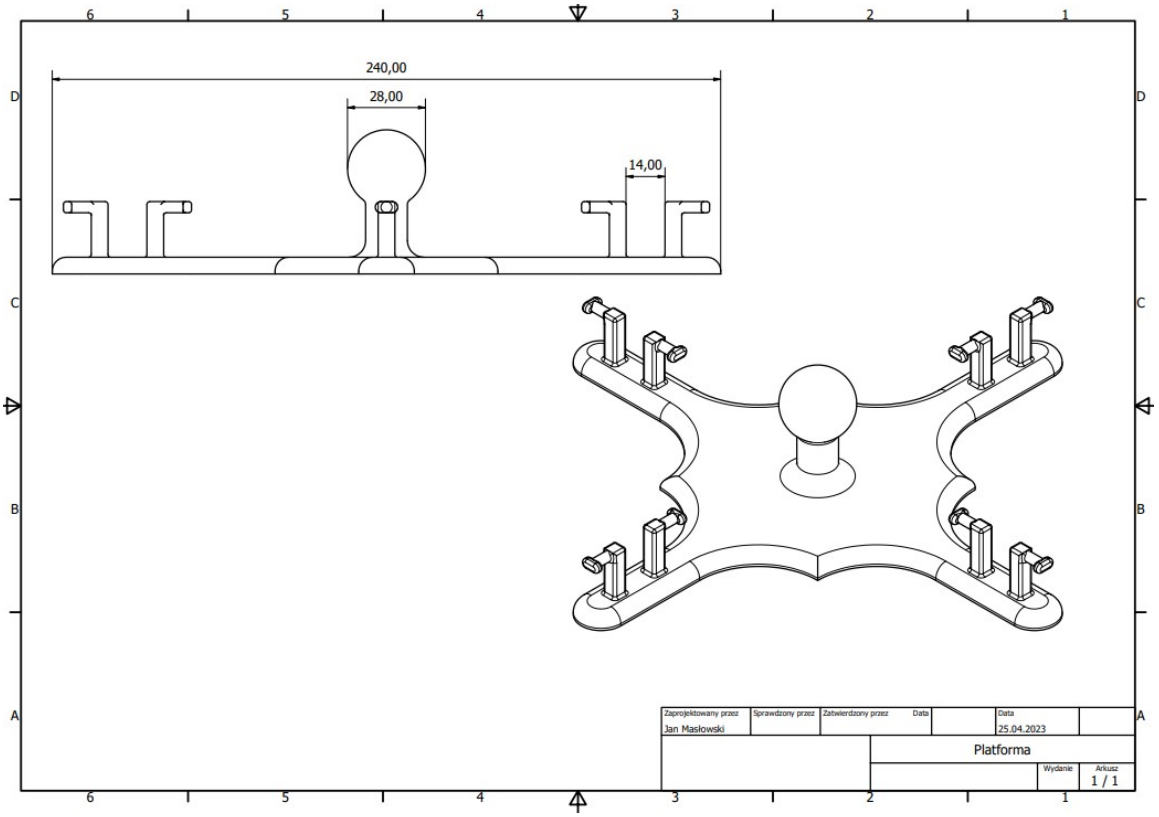
Wybrane rysunki techniczne elementów drukowanych:



Rysunek 1: Podstawa ramienia kamery.



Rysunek 2: Uchwyt kamery.

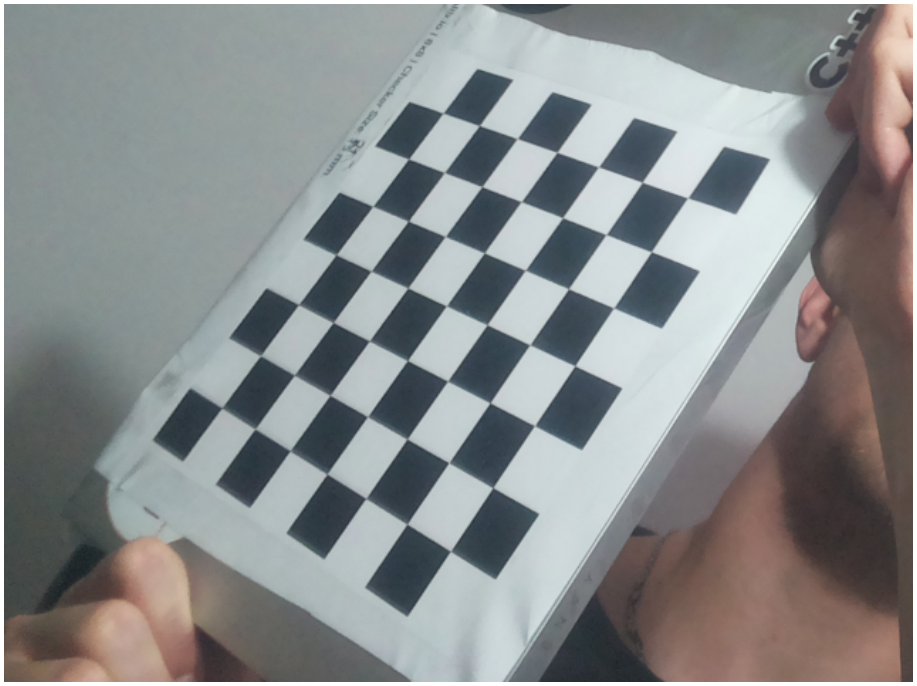


Rysunek 3: Platforma balansująca.

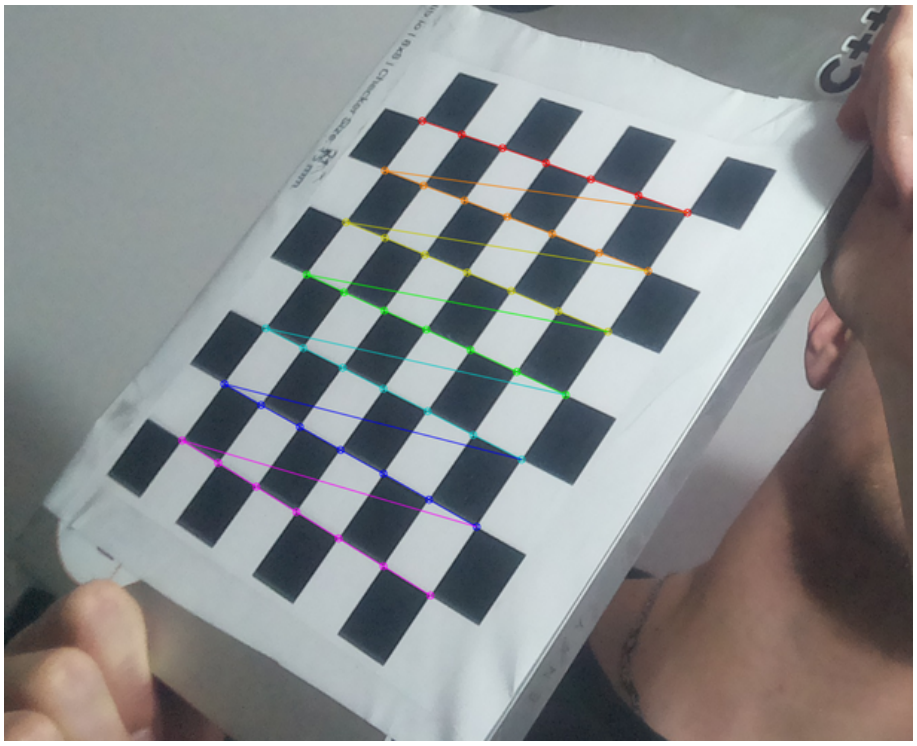
4 Opis działania programu

4.1 Kalibracja kamery

Do kalibracji wykorzystane zostały funkcje dostępne w bibliotece OpenCV. Kalibracja odbyła się poprzez analizę zdjęć szachownicy. Każde zdjęcie wejściowe jest analizowane pod kątem wyznaczenia wewnętrznych krawędzi szachownicy. Po przeanalizowaniu wszystkich zdjęć odbywa się kalibracja, która wyznacza parametry kamery.



Rysunek 4: Obraz wejściowy



Rysunek 5: Obraz wyjściowy

4.2 Detekcja markerów

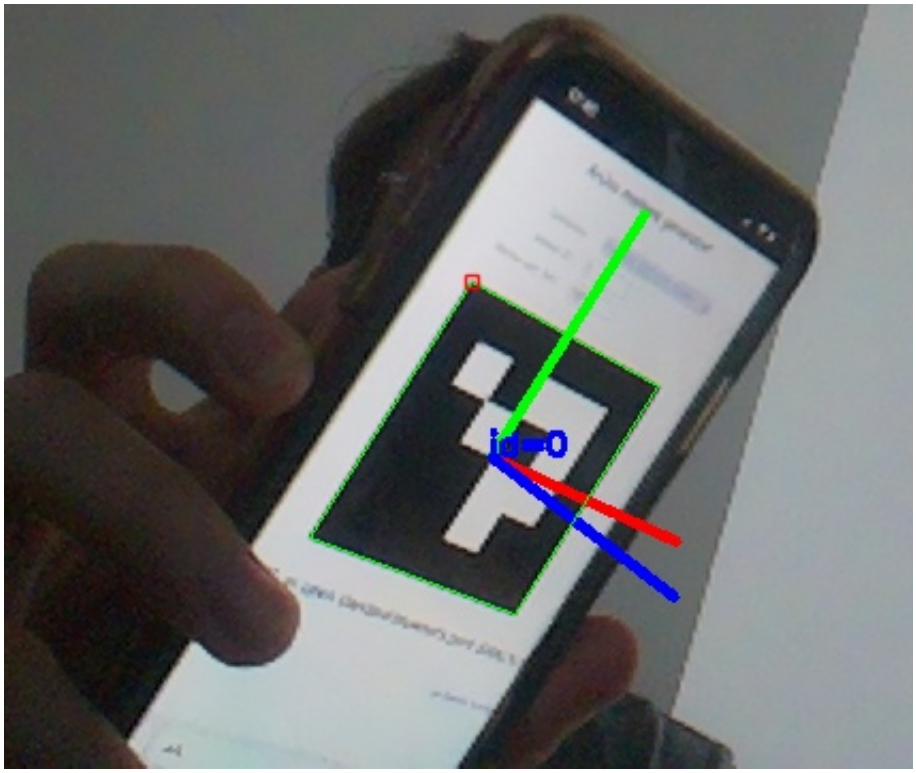
Korzystając z biblioteki OpenCV została zaimplementowana detekcja markerów Aruco oraz estymacja ich pozycji w przestrzeni.

Każda klatka jest wczytywana i analizowana w poszukiwaniu markerów Aruco.

```
def poseEstimationAruco(frame, matrixCoeff, distortionCoeff, arucoDetector):

    gray = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
    markerCorners, markerIDs, rejectedCandidates = arucoDetector.detectMarkers(
        gray)
```

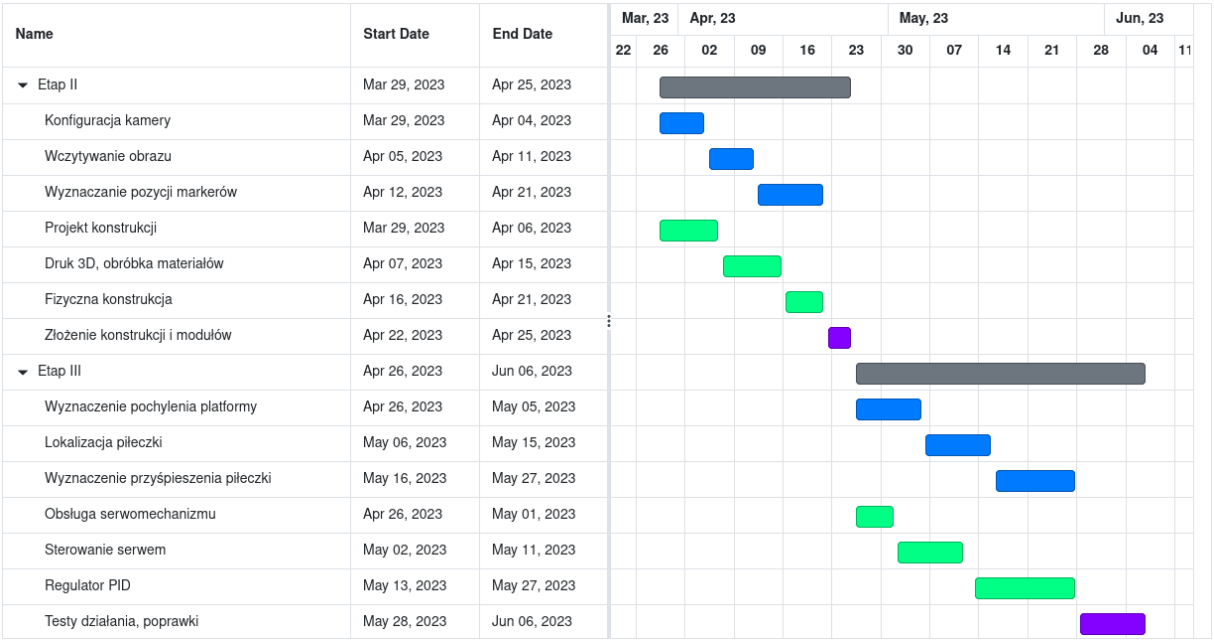
Następnie estymowana i rysowana jest pozycja w przestrzeni każdego wykrytego markera. Do estymacji wykorzystywane są paremtry wyznaczone podczas kalibracji kamery.

```
    if len(markerCorners) > 0:
        for i in range(0, len(markerIDs)):

            rvec, tvec, markerPoints = cv.aruco.estimatePoseSingleMarkers(
                markerCorners[i], 0.02, matrixCoeff, distortionCoeff)
            cv.drawFrameAxes(frame, matrixCoeff, distortionCoeff, rvec, tvec,
                0.02)
        markerIDs = markerIDs.flatten()
        cv.aruco.drawDetectedMarkers(frame, markerCorners, markerIDs)

    return frame
```



Rysunek 6: Przykład detekcji i estymacji pozycji markera

5 Harmonogram pracy



Rysunek 7: Diagram Gantta uzyskany za pomocą serwisu Online Gantt.

Legenda: Kolor niebieski: Michał Trela, Kolor zielony: Jan Masłowski, Kolor fioletowy: cele wspólne

5.1 Podział pracy

Michał Trela	**Jan Masłowski**
Konfiguracja interfejsu kamery	Stworzenie projektu konstrukcji mechanicznej
Wczytywanie obrazu z kamery	Druk 3D elementów, obróbka materiałów
Wyznaczenie pozycji markerów Aruco na podstawie wczytanego obrazu	Wykonanie fizycznej konstrukcji
Złożenie konstrukcji i modułów elektronicznych w całość	

Tabela 3: Podział pracy – Etap II

Michał Trela	**Jan Masłowski**
Wyznaczenie pochylenia platformy	Obsługa serwomechanizmu
Lokalizacja piłeczki na platformie	Sterowanie serwomechanizmem z różnymi prędkościami
Wyznaczanie wartości i kierunku przyśpieszenia piłeczki	Implementacja regulatora PID
Testy działania oraz końcowe poprawki	

Tabela 4: Podział pracy – Etap III

5.2 Kamienie milowe

- Obsługa kamery - Transmisja obrazu w czasie rzeczywistym oraz wczytywanie obrazu jako tablica do programu [1] [2] [3]
- Stworzenie fizycznej konstrukcji - Wydrukowanie i złożenie elementów w całość

- Obsługa serwomechanizmu - Możliwość sterowania serwomechanizmem z różnymi prędkościami
- Lokalizacja piłki - Wczytany obraz jest analizowany, określana jest dokładna pozycja piłki oraz pozycja markerów
- Wyznaczanie prędkości piłki - Analiza pozycji piłki w zadanym czasie i wyznaczanie wartości
- Implementacja regulatora PID - Implementacja regulatora w oprogramowaniu, wyznaczone wszystkie nastawy [4] [5]
- Balansowanie piłką - Robot balansuje piłką na platformie bez udziału człowieka

6 Dokumentacja projektu

Kod źródłowy oraz dokumentacja zostały zamieszczone w serwisie GitHub:
`https://github.com/Mmichal1/SR-Projekt.git`

7 Zadania niezrealizowane

Złożenie konstrukcji i modułów elektronicznych w całość

Tego zadania nie udało się zrealizować na czas z powodu problemów zdrowotnych jednej z osób.

Literatura

[1] G. Bradski, A. Kaehler, i in. Opencv. *Dr. Dobb's journal of software tools*, 3(2), 2000.

[2] F. Jalled, I. Voronkov. Object detection using image processing. *CoRR*, abs/1611.07791, 2016.

[3] M. A. Pagnutti, R. E. Ryan, G. J. C. V, M. J. Gold, R. Harlan, E. Leggett, J. F. Pagnutti. Laying the foundation to use Raspberry Pi 3 V2 camera module imagery for scientific and engineering purposes. *Journal of Electronic Imaging*, 26(1):013014, 2017.

[4] E. Sariyildiz, H. Yu, K. Ohnishi. A practical tuning method for the robust pid controller with velocity feed-back. *Machines*, 3(3):208–222, 2015.

[5] K. Yaovaja. Ball balancing on a stewart platform using fuzzy supervisory pid visual servo control. *2018 5th International Conference on Advanced Informatics: Concept Theory and Applications (ICAICTA)*, strony 170–175, 2018.