DOKUMENTACJA ROBOTA {NAZWA ROBOTA}

AUTORZY: PAWEŁ SOBCZYŃSKI, SZYMON ZIAJEWSKI, WIKTOR ROSÓŁ

OPIEKUNOWIE: MATEUSZ STEFANIK, MAŁGORZATA ZIAJEWSKA

1. Przeznaczenie robota:

{coś tu trzeba napisać}

1. Komponenty:
   1. Jednostka logiczna z modułem bluetooth
   2. Kamera
   3. Sterownik silnika chwytaka
   4. Sterownik silnika kół
   5. Serwo podstawy
   6. Serwo ramienia 1
   7. Serwo ramienia 2
   8. Serwo obrotu chwytaka
   9. Silnik otwierający/zamykający chwytak
   10. Silnik koła 1
   11. Silnik koła 2
   12. Moduł zasilający 5V
   13. Moduł zasilający 7.4V
2. Robot:
   1. {zdjęcie robota z podpisami}
3. Zasady działania komponentów:
   1. Serwo:

Do serwa dochodzą 3 kable: zasilanie 5V, uziemienie, sygnał PWM. PWM to inaczej modulacja szerokości impulsów, (ang. pulse-width modulation). Jest to metoda regulacji sygnału prądowego lub napięciowego, o stałej amplitudzie i częstotliwości. Wykorzystane w układzie serwa mają czas jednego pełnego cyklu 20 ms (milisekund). Długość sygnału prądowego wacha się od 0.5 ms do 2.2 ms.

* 1. Sterownik silnika:

Do sterownika wchodzą 4 kable: zasilanie 5V lub 12V (wykorzystane są dwa różne sterowniki w robocie), uziemienie, dwa kable sygnałowe. Sterownik ma za zadanie kręcić silnikiem w obie strony oraz go zatrzymać. Aby to osiągnąć potrzebne są dwa kable z danymi. Zasada działania jest następująca: jeśli na obydwu kablach jest sygnał 0 lub 1, to silnik się zatrzymuje. Jeśli na pierwszym jest 1 a na drugim 0, silnik kręci się w jedną stronę. Natomiast jeśli na pierwszym jest 0 a na drugim jest 1, to silnik kręci się w drugą stronę. Sprecyzowanie kierunku kręcenia jest niemożliwe gdyż zależy to od podłączenia silnika do sterownika.

* 1. Moduł zasilania:

Moduł zasilania składa się z akumulatorów litowo-jonowych 18650, połączonych w moduł zasilający układy robota.

1. Sterowanie robotem:
   1. Lewo – obrót podstawą w {strona}
   2. Prawo – obrót podstawą w {strona}
   3. Góra – pierwszy przegub w {strona}
   4. Dół – pierwszy przegub w {strona}
   5. Kwadrat – drugi przegub w {strona}
   6. Kółko – drugi przegub w {strona}
   7. Trójkąt – obrót chwytaka w {strona}
   8. Krzyżyk – obrót chwytaka w {strona}
   9. L1 – otwieranie chwytaka
   10. R1 – zamykanie chwytaka
   11. Lewy analog w lewo – obrót robota w lewo
   12. Lewy analog w prawo – obrót robota w prawo
   13. Prawy analog w górę – jazda do przodu
   14. Prawy analog w dół – jazda do tyłu



Nie jest zalecane klikanie wielu przycisków jednocześnie!

1. Dokumentacja kodu:
   1. Technologia:

Wykorzystany układ logiczny został zaprogramowany w języku Arduino.

* 1. Import bibliotek:

Obraz zawierający tekst, Czcionka, zrzut ekranu, linia

Opis wygenerowany automatycznie

Program korzysta z dwóch bibliotek. ESP32Servo – pozwala układowi ESP32 na sterowanie serwem. Interfejs jest identyczny jak dla zwykłych płytek Arduino.

Ps3 Controller Host – pozwala na emulację konsoli Ps3, aby można było podłączyć pada dualshock 3 do układu ESP32.

* 1. Wykorzystywane piny:

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

Wszystkie wykorzystywane piny zostały zdefiniowane jako stałe kompilacji.

* 1. Klasa MyServo:

Aby poruszać serwami robota potrzeba pamiętać obecną pozycję serwa. Dla czytelności i prostoty użytkowania została stworzona klasa MySerwo która będzie pamiętać obecną pozycję, sterować serwem oraz po starcie programu ustawi go w pozycję początkową.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Konstruktor MyServo():

Inicjalizuje obiekt, wszystkie zmienne oraz ustawa serwo w pozycję początkową.

Zmienna servo wskazuje na obiektem klasy Servo() z biblioteki ESP32Servo. Wykorzystane metody:

.attach(int pin, int min, int max) – łączy pin z serwem oraz ustawia minimalny i maksymalny czas impulsu elektrycznego sygnału PWM w mikrosekundach.

.writeMicroseconds(int value) – ustawia serwo w odpowiedniej pozycji, value jest czasem impulsu elektrycznego w mikrosekundach.

.write(int degree) – ustawia kąt w jakim serwo ma się ustawić (od 0 do 180 stopni).

* 1. Klasa Engine:

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

Klasa Engine jest implementacją działania sterownika silnika którego działanie zostało wytłumaczone w podpunkcie 3.2.

* 1. Stałe:

Obraz zawierający tekst, Czcionka, zrzut ekranu, biały

Opis wygenerowany automatycznie

Stałe oznaczające pozycję domyślną serw w ramieniu, ustawianą podczas uruchamiania robota.

Obraz zawierający tekst, Czcionka, zrzut ekranu, linia

Opis wygenerowany automatycznie

Stałe oznaczające pozycję neutralną serw w ramieniu. Jest to pozycja złożona ramienia dzięki której jazda oraz wyłączenie robota jest bezpieczniejsze.

* 1. Inicjalizacja obiektów:

Obraz zawierający tekst, Czcionka, zrzut ekranu

Opis wygenerowany automatycznie

Inicjalizacja wszystkich potrzebnych obiektów do sterowania robotem.

* 1. Callback functions:

Callback funtions można tłumaczyć jako funkcje zwrotne. Podczas gdy biblioteka wywołuje jakąś funkcję, użytkownik może chcieć dodać dodatkowe operacje do niej. Taką możliwość daję biblioteka Ps3 Controller Host. Do obiektu Ps3, który ma własną implementację zarządzania kontrolerem dualshock 3, jesteśmy w stanie przekazać callback function aby móc przypisać własne zachowanie do odpowiednich przycisków lub eventów.

* + 1. notify():

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Funkcja notify() implementuje całe sterowanie robotem.

* + 1. connected():

Obraz zawierający tekst, Czcionka, zrzut ekranu, linia

Opis wygenerowany automatycznie

Funkcja służąca głównie do debugowania, aby było wiadomo, że pad został podłączony.

* + 1. disconnected():

Obraz zawierający tekst, Czcionka, zrzut ekranu, linia

Opis wygenerowany automatycznie

Funkcja służąca głównie do debugowania, aby było wiadomo, że pad został rozłączony.

* 1. setup():

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

Jedna z dwóch głównych funkcji w języku Arduino. Wywoływana jest raz na początku programu.

* 1. loop():

Obraz zawierający tekst, Czcionka, pismo odręczne, biały

Opis wygenerowany automatycznie

Druga główna funkcja w języku Arduino. Wykonuje kod w pętli. W tym przypadku jest pusta i jest zdefiniowana po to aby uniknąć błędów kompilacji.

1. Zakończenie:

{walnąć jakieś ładne zakończenie i będzie gituwa}