Linefollower

Maja Jaszowska
177088 EFDI

Część 1

1. Wprowadzenie

Linefollower to robot, którego zadaniem jest jazda po wyznaczonej linii. Posiada algorytm, który samodzielnie dostosowuje tor jazdy za pomocą danych popranych z czujników. Sensory analizują ilość odbitego światła, dzięki czemu algorytm jest w stanie odpowiednio zadziałać.

2. Zawody linefollowerów

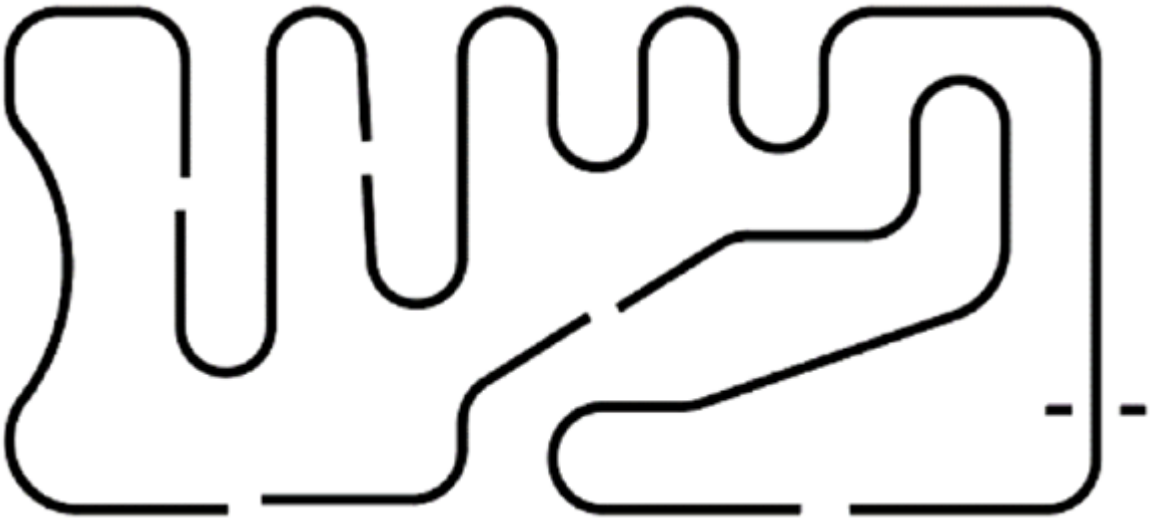
W Polsce przez lata odbywały się zawody, w których jedną z kategorii stanowiły konkurencje dedykowane tego typu robotom. Trasa zazwyczaj jest wyznaczona czarnymi liniami na białym lub bardzo jasnym tle, choć zdarzają się też trasy z białymi liniami na czarnym tle. Zarówno trasę, jak i kategorię, dostosowuje się do różnych parametrów robotów, które różnią się wielkością i rodzajem użytych materiałów. Na zawodach można spotkać osobne konkurencje dla robotów elektronicznych oraz tych zbudowanych z klocków Lego. Linefollower, który pokona trasę najszybciej i z największą precyzją, zostaje zwycięzcą.

Założenia, które musi spełniać linefollower:

- robot jeździ po trasie autonomicznie, bez pomocy zewnętrznej,
- za pomocą algorytmu dobiera wartości prędkości silników na określonych punktach trasy,
- w przypadku zgubienia czarnej linii potrafi ją samodzielnie odnaleźć,
- poprawnie analizuje dane z czujników, dzięki czemu przejeżdża trasę z dużą prędkością z jak najmniejszą ilością błędów,
- jest dobrze wyważony,
- ma możliwie najmniejszą wagę,

3. Trasa

Trasa na zawodach jest z góry określona przez organizatorów. W Polsce zazwyczaj można spotkać białą powierzchnię z czarnymi linami o szerokości od 15 do 20 mm.



Rysunek 1. Przykładowa trasa dla linefollowera
Źródło:
http://2gym-samou.sam.sch.gr/autosch/joomla15/images/stories/robotics/samianator_report.pdf
(dostęp: 25.05.2024).

Na tak wyznaczonej drodze można spotkać:

- linie proste,
- zakręty prostopadle i łuki,
- skrzyżowania i utrudnienia w postaci kilku linii wychodzących,
- inie przerywane.

Część 2

4. Wybrany sprzęt

Podczas doboru odpowiednich komponentów kierowano się ich skutecznością, napięciem zasilania, wielkością, wagą oraz trwałością. Czujniki są jednymi z najważniejszych części robota. Wybrano listwę cyfrową z czujnikami odbiciowymi Pololu QTR-8A.

Specyfikacja:

- wymiary 75 x 127 x 4 mm,
- napięcie zasilania 5 V lub 3,3 V,
- prąd zasilania 100 mA,
- sygnał na wyjściu: cyfrowy (kompatybilny z pinami I/O),
- maksymalna odległość pomiaru: 9,5 mm.[1]

Mikrokontroler jest kluczową częścią dla robota. ESP32 jest kompatybilny z wybranymi czujnikami oraz zapewnia odpowiednią moc obliczeniową do wykonywania algorytmu śledzenia linii i sterowania silnikami. Wybrano model ESP32-S3-WROOM-1-N16R8.

Specyfikacja:

- układ: ESP32-S3-WROOM-1/1U
- procesor: Dual-Core 32-bit Xtensa LX7
- taktowanie: do 240 MHz
- komunikacja bezprzewodowa: WiFi i Bluetooth
- standard WiFi: IEEE 802.11 b/g/n (802.11n do 150 Mbps)
- pasmo: od 2,412 GHz do 2,484 GHz
- standard Bluetooth: LE 5 Mesh
- pamięć SRAM: 512 kB (w tym 16 kB w pamięci RTC), Pamięć ROM: 384 kB, Pamięć Flash: 8 MB[2]

Sterownik powinien być dopasowany do dobranych silników. Konieczne jest, aby obsługiwał wymagane napięcie oraz prąd potrzebny do poprawnego działania

układu. Wybrano więc moduł sterownika Mini MX1508 przeznaczony do silników DC o zakresie napięcia od 2V do 10V i prądzie do 1.5A.

Specyfikacja:

- moduł umożliwia sterownia dwoma silnikami prądu stałego
- napięcie zasilania: 2 - 10V DC
- napięcie zasilania części logicznej: 5V
- wbudowany regulator napięcia 5V
- maksymalny prąd wyjściowy: 1.5 A na kanał, 0.8A ciągły
- wymiary płytki: 21x25x17 mm
- podając sygnał PWM na którykolwiek z tych pinów wejściowych steruje się kierunkiem i prędkością obrotową silników.[3]

Przetwornica jest niezbędna do odpowiedniego zasilania mikrokontrolera oraz czujników w linefollowerze. Moduł zasilacza obniżającego napięcie DAOKI MINI DC-DC 12-24V TO 5V 3A obsługuje wymagane napięcie wejściowe i wyjściowe.

Specyfikacja:

- napięcie wejściowe: 4,5-24V,
- napięcie wyjściowe: 0,8V do 21V,
- napięcie wyjściowe regulowane z pomocą wbudowanego potencjometru,
- możliwość ustawienia na stałe 6 różnych napięć wyjściowych (1,8V;2,5V;3,3V;5V;8V;12V),
- za pomocą zwarcia pól znajdujących się na płytce,
- max. prąd wyjściowy: 3A,
- wymiary: 11x20,5x6mm.[4]

Silniki powinny mieć odpowiednie wymiary oraz być lekkie. Mogą odegrać istotną rolę w zapewnieniu dobrego wyważenia robota. Zbyt duże silniki mogą utrudniać pracę linefollowera. Ich sterowanie powinno być proste, a jednocześnie posiadają wysoki moment obrotowy. Wybrano silnik DC GA12-N20 6V z przekładnią o prędkości 2000 obr/min.

Specyfikacja:

- napięcie zasilania: od 1V do 6V,
- prąd na biegu jałowym: pon. 0,05A,
- moment obrotowy: 30kg*cm (3Nm),
- wymiary: 25mm dł. x 12mm szer. x 10mm. wys.,
- średnica wału: 3mm,
- długość wału: 9mm.[5]

Akumulatory Li-Po cechują się jedną z najlepszych wydajności energetycznych w porównaniu do innych typów akumulatorów. Mają wysoki stosunek mocy do wagi, co czyni je atrakcyjnym wyborem dla robotów linefollower, które często muszą być lekkie i zwinne. Akumulator Li-Pol Dualsky 520mAh 25C 2S 7,4V dostarcza wysoki prąd w krótkim czasie, co dobrze współgra z całym układem.

Specyfikacja:

- dwa ogniwa (2S) Li-Po,
- napięcie nominalne: 7,4 V,
- pojemność: 520 mAh,
- wymiary: 58 x 18 x 19 mm.[7]

Informacje o specyfikacji:

[1]https://botland.com.pl/czujniki-odbiciowe/20-listwa-z-czujnikami-odbiciowymi-qtr-8rc-cyfrowa-pololu-961-5903351249287.html

[2]https://botland.com.pl/moduly-wifi-i-bt-esp32/20739-esp32-s3-devkitc-1-n8-wifi-bluetooth-plytka-rozwojowa-z-ukladem-esp32-s3-wroom-11u-5904422382353.html?cd=1050025856&ad=51004438223&kd=&gad_source=1&gclid=Cj0KCQjwjLGyBhCYARIsAPqTz19bkTQ6sop-6Y3ugUdsIVEvCBs1hrGb6KdkcGNrm8p2TkwKad-JCNUaAmWWEALw_wcB

[3]https://sklep.avt.pl/pl/products/modul-sterownika-mini-mx1508-do-silnikow-dc-podwojny-arduino-187397.html

[4]https://rif.sklep.pl/az/regulatory-napiecia/655-mini-przetwornica-step-down-08-21-3a-975.html

[5]https://abc-rc.pl/product-pol-9257-Mini-silnik-szczotkowy-GA12-N20-50RPM-wal-9mm-3-6V-z-przekladnic cva.html

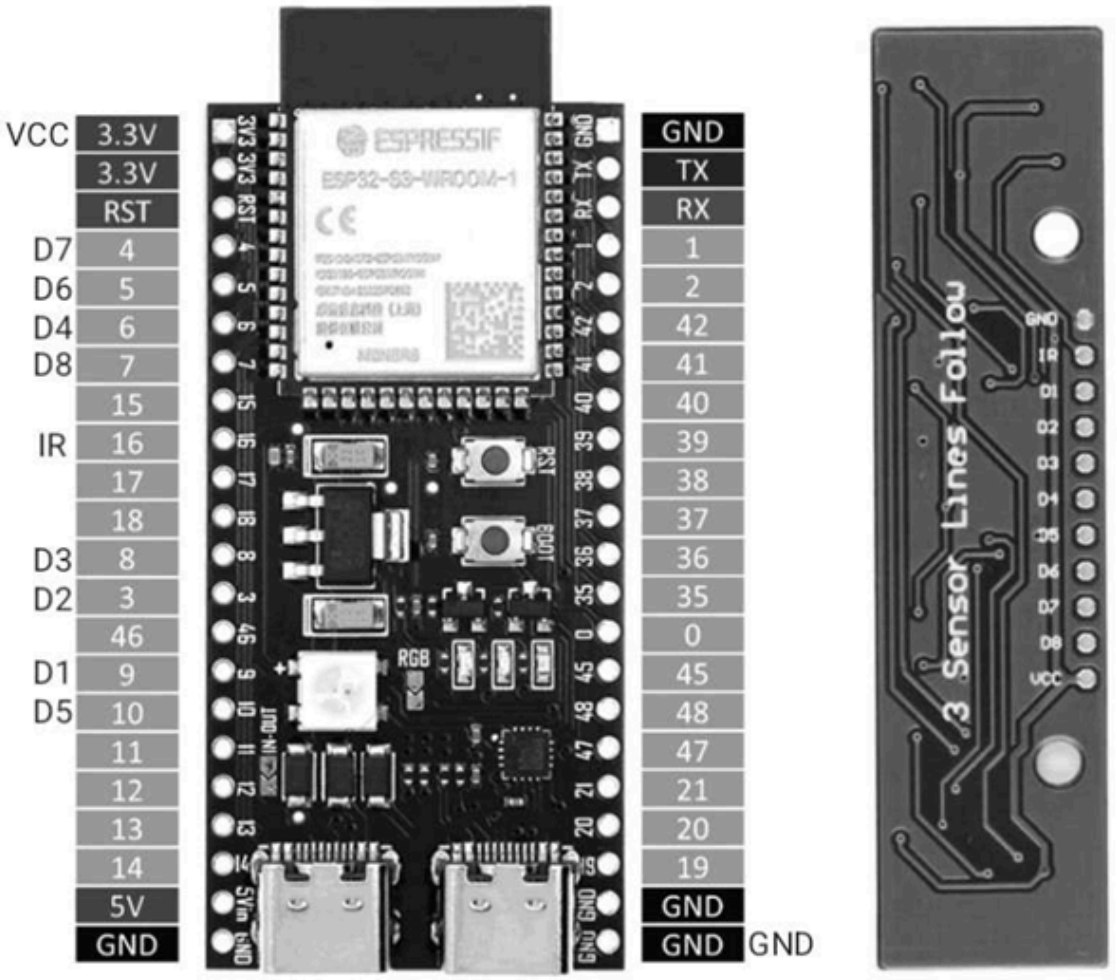
[6]https://botland.com.pl/akumulatory-li-pol-2s-74v-/570-pakiet-li-pol-dualsky-520mah-25c-2s-74v-6941047107427.html?cd=19993067448&ad=&kd=&gad_source=1&gclid=Cj0KCQjwjLGyBhCYARIsAPqTz18ktWWt5vxUDfN3VAo-B0eI9fq4_dvstwlujnqk-5hzmR1qZpwnHggaAn-CEALw_wcB

5. Budowa

Głównym celem robota jest poruszanie się po wyznaczonej linii. Podczas konstruowania linefollowera niezbędne jest zaimplementowanie mechanizmu umożliwiającego pobieranie danych na temat trasy. Dokładne zbieranie informacji o trasie jest kluczowe dla skutecznego poruszania się robota po linii, umożliwiając mu precyzyjne reakcje na zmiany w otoczeniu.

Płytka obsługuje napięcie 5V lub 3,3V, jednak w projekcie wykorzystuje się napięcie 3,3V. Dzięki swoim niewielkim wymiarom, które wynoszą 75 x 127 x 4 mm, linefollower jest dość lekki i kompaktowy, co ułatwia manewry.

Podczas montażu bardzo ważne jest, aby diody były w pewnej niewielkiej odległości od powierzchni. W przeciwnym wypadku czujniki będą wykrywać linie w sposób niepoprawny, co prowadzi do problemów z jazdą robota. Przed użyciem listwy należało pamiętać, by zalutować część oznaczoną 3,3V. Jest to wbudowany rezystor, bez którego nie jest możliwe bezpieczne korzystanie z czujników. Na internecie można znaleźć bardzo podobny model, bez wbudowanego rezystora. W takim wypadku trzeba dokupić i dolutować odpowiedni rezystor w wyznaczonym przez producenta miejscu. Moduł QTR8-A ma piny, które należało podłączyć do mikrokontrolera. Zarówno piny o numerach D1-D8 jak i te VCC, GND i IR. W przypadku jakiegokolwiek błędu podczas montażu ESP32 nie łączy się z wifi. Rysunek 2. przedstawia schemat łączenia czujników z ESP32.



Rysunek 2. Łączenie mikrokontrolera z czujnikami Pololu QTR-8RC
Źródło: https://ifuturetech.org/product/qtr-8rc-reflectance-sensor-array/ (dostęp: 25.05.2024), https://99tech.com.au/product/esp32-s3-yd-n8r8/ (dostęp: 25.05.2024). Opracowanie własne.

Mikrokontroler należało przylutować do płytki prototypowej za pomocą jednorzędowej listwy kołkowej goldpinów męskich po obu stronach ESP32. Do płytki wymagane było dolutowanie kolejnego zestawu 11 goldpinów męskich jednorzędowych. Tak jak na schemacie wyżej, konieczne jest połączenie 11 goldpinów z odpowiednimi pinami mikrokontrolera posiadającymi funkcję ADC. Podczas przygotowywania czujników wskazane jest zalutowanie oznaczenia 3.3V oraz przylutowanie goldpinów męskich. Łączenie czujników z mikrokontrolerem odbywa się za pomocą zestawu 11 przewodów połączeniowych JustPi

żeńsko-żeńskich, gdzie po jednej stronie są wpięte do pinów ESP a po drugiej do pinów czujnika.

Takie rozwiązanie wymaga utworzenia podpory, która zostanie przykręcona za pomocą śrub do płytki prototypowej, oraz czujników, które posiadają odpowiednie wypusty do tego celu w swojej konstrukcji. Istotne jest, aby podpora była odpowiednio długa, co pomaga robotowi w zwiększeniu czasu reakcji na nagłe zmiany kierunku trasy. W przypadku budowy robota z myślą o zawodach, istotne jest sprawdzenie, jakie wymiary są wymagane przez organizatora. Zazwyczaj są to 210 x 297 mm, czyli rozmiar kartki A4.

Ważne jest pamiętać, że przed rozpoczęciem trasy należy przeprowadzić kalibrację robota. Proces ten zależy od programu i trwa zazwyczaj od 10 do 25 sekund. Kalibracja jest niezbędna, aby czujniki mogły dostosować się do warunków oświetleniowych panujących na powierzchni. Polega ona na powolnym przesuwaniu czujników wzdłuż fragmentu toru, najlepiej pomiędzy czarną linią a białym podłożem.

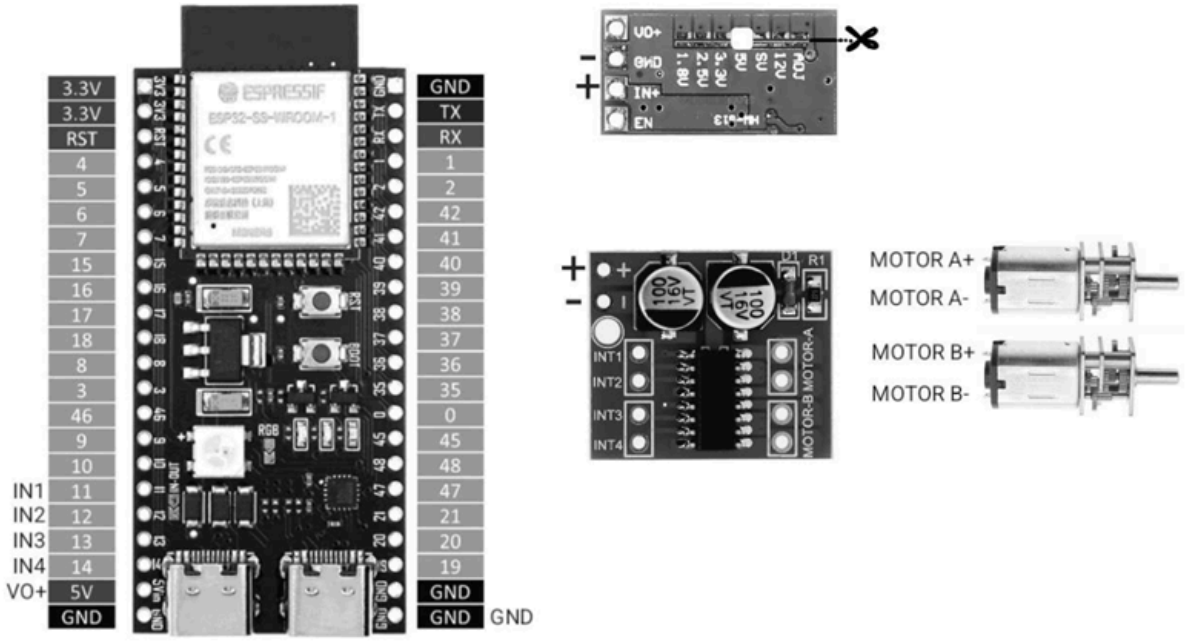
Moduł sterownika Mini MX1508 to sterownik silników umożliwia sterowanie dwoma silnikami DC. Układem można sterować przy pomocy Arduino, podpinając jego piny pod następujące wejścia układu: IN1, IN2, IN3, IN4. Podając sygnał PWM na którykolwiek z tych pinów wejściowych steruje się kierunkiem i prędkością obrotową silników. [1] Dodatkowym elementem potrzebnym do prawidłowego funkcjonowania robota będzie tzw. mostek H, który pozwala m.in. na zmianę kierunku obrotów silnika oraz sterowanie ich prędkością.[2] Pod pojęciem PWM kryje się anglojęzyczny skrót „Pulse Width Modulation”, który oznacza modulację szerokości impulsów. Jest to nic innego jak metoda sterowania układami elektronicznymi przy pomocy manipulacji samym sygnałem sterującym, a konkretniej jego parametrami technicznymi. Wynika to z samej charakterystyki fizycznej sygnału PWM, który składa się ze wspomnianych wcześniej impulsów. Ich podstawowe parametry dotyczą czasu pozostania w stanie wysokim (wartość logiczna 1) oraz częstotliwości przełączania sygnału PWM.[3]

Zasilanie należy podłączyć do pinów GND i VCC. Na sterowniku oznaczono piny dla silnika A i silnika B. Moduł posiada piny o następujących oznaczeniach: IN1, IN2, IN3, IN4. Dla silnika A są dedykowane piny sterujące IN1, IN2, a dla silnika B IN3,

IN4. Ta informacja jest istotna, aby zrozumieć sposób sterowania kierunkiem obrotu silników. Sygnał PWM jest wykorzystywany do regulacji prędkości.

Oznacza to, że gdy na obu pinach IN1 i IN2 jest stan wysoki lub niski, silniki zostaną zatrzymane. Natomiast jeśli na IN1 jest stan wysoki, a na IN2 stan niski (lub odwrotnie), silnik będzie obracał się w określonym kierunku (lub w przeciwnym). Te możliwości pozwalają na bardzo precyzyjną kontrolę prędkości silników, co jest istotne przy sterowaniu robotem liniowym.

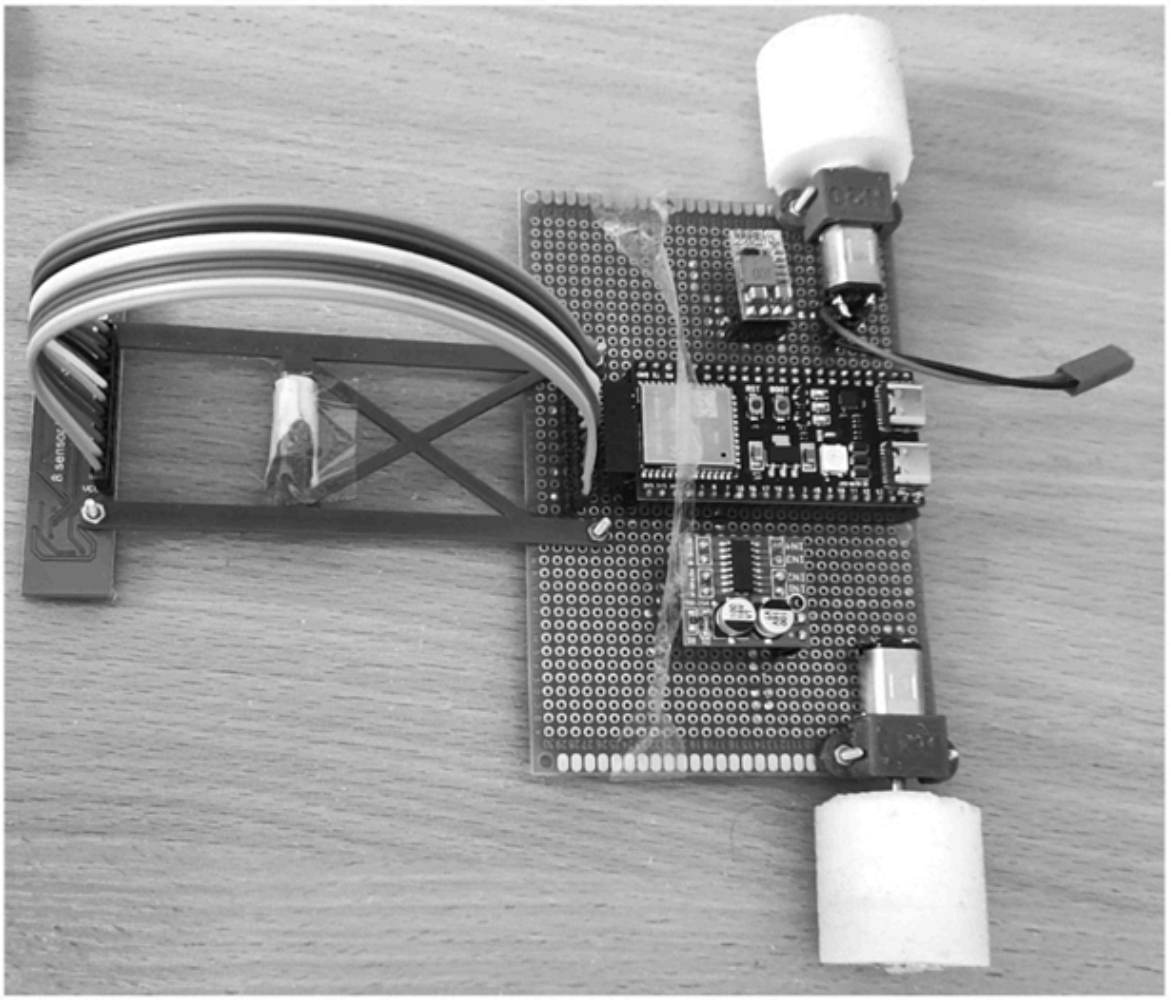
Przetwornica DAOKI służy do konwersji napięcia wejściowego z zasilacza na niższe napięcie wyjściowe, odpowiednie dla ESP32. Moduł może obsługiwać napięcia wejściowe od 4,5V do 24V, a użytkownik ma możliwość wyboru pożądanego napięcia wyjściowego. Piny IN+ i GND służą do połączenia z akumulatorem, natomiast pin VO+ umożliwia bezpośrednie zasilanie mikrokontrolera ESP32. Rysunek 3. przedstawia schemat łączenia ESP32, przetwornicy i modułu sterowania.



Rysunek 3. Łączenie mikrokontrolera z czujnikami Pololu QTR-8RC
Źródła: https://quartzcomponents.com/products/mini-l298 (dostęp: 25.05.2024), https://99tech.com.au/product/esp32-s3-yd-n8r8/ (dostęp: 25.05.2024), https://www.amazon.ca/DAOKI-Supply-Voltage-Converter-Adjustable/dp/B07X86VKCL (dostęp: 25.05.2024), https://www.amazon.pl/elektryczny-Motoreduktor-wolnoobrotowy-skrzynia-obrotowym/dp/B0C4PH7HS8 (dostęp: 25.05.2024). Opracowanie własne.

Przed zamontowaniem regulatora w układzie konieczne jest wlutowanie na płytce obok zaznaczenia 5V oraz przecięcie linii zaznaczonej na schemacie za pomocą ostrego narzędzia. Tylko wtedy regulator napięcia będzie działał poprawnie. Zarówno przetwornica, jak i sterownik silników należy przylutować do płytki prototypowej przy użyciu goldpinów. Ważne jest, aby połączyć pin przetwornicy VO+ z pinem oznaczonym jako 5V oraz pin GND z odpowiednim pinem mikrokontrolera. Niezbędne jest podłączenie zasilania do obu części w sposób, który wykluczy możliwość zwarcia. Piny IN1-IN4 muszą zostać połączone z ESP, aby wgrany na nią program mógł sterować silnikami podłączonymi do modułu za pomocą MOTOR-A i MOTOR-B.

Połączenie silników do sterownika to tylko pierwszy krok, by zapewnić stabilność całego systemu. Dla zapewnienia bezpieczeństwa silniki powinny być dodatkowo przymocowane do płytki prototypowej w taki sposób, aby solidnie trzymały się całego linefollowera.



Rysunek 4 przedstawia poglądowy wygląd zbudowanego robota.

Rysunek 4. Zbudowany linefollower.

Źródło: Opracowanie własne.

---

[1]https://abc-rc.pl/product-pol-17960-Modul-sterownika-Mini-MX1508-do-silnikow-DC-podwojny-Arduino.html).

[2]https://zpe.gov.pl/a/przeczytaj/DUpr2MS6R

[3]https://botland.com.pl/blog/sygnal-pwm-czym-jest/

6. Kod

Podczas tworzenia programu dla linefollowera, najważniejsze jest poprawna implementacja logiki i zasad, które pozwolą robotowi płynnie i bezbłędnie przejeżdżać trasę. Program wgrywany na mikrokontroler został napisany w języku C w środowisku Arduino IDE 2.3.2. Listing 1. Przedstawia implementacje programu dla robota.

```
void loop()
{
  // Odczyt pozycji czarnej linii
  int position = qtr.readLineBlack(sensorValues);


  // Zapamiętywanie ostatniej pozycji linii


if (sensorValues[0] >= 800 && sensorValues[NUM_SENSORS-1] < 800)
{
     if (last_sighted != 1 && millis() - last_detection_time >=
100) {
      last_sighted = 1;
      last_detection_time = millis();
      pixels.setPixelColor(0, pixels.Color(0, 255, 0));
// Zielony dla lewej
      pixels.show();
    }
         }    else    if    (sensorValues[0]    <    800    &&
sensorValues[NUM_SENSORS-1] >= 800) {
     if (last_sighted != 2 && millis() - last_detection_time >=
100) {
      last_sighted = 2;
      last_detection_time = millis();
      pixels.setPixelColor(0, pixels.Color(255, 0, 0));
// Czerwony dla prawej
      pixels.show();
    }
  }


    // Kontrola orientacji robota, ocena czy linia została
zgubiona


  lost_sensors = 0;
  lost = 0;
  for(int i = 0; i < NUM_SENSORS; i++){
    if(sensorValues[i] <= lost_threshold){
      lost_sensors += 1;
    }
  }
  if(lost_sensors >= (NUM_SENSORS)){
    lost = 1;
  }
```

```
//PID


int error = position - 3500;


int motorSpeed = Kp * error + Kd * (error - lastError);
lastError = error;


rightMotorSpeed = BaseSpeed + motorSpeed;
leftMotorSpeed = BaseSpeed - motorSpeed;

if (rightMotorSpeed > MaxSpeed ) rightMotorSpeed = MaxSpeed;
if (leftMotorSpeed > MaxSpeed ) leftMotorSpeed = MaxSpeed;
if (rightMotorSpeed < 0) rightMotorSpeed = 0;
if (leftMotorSpeed < 0) leftMotorSpeed = 0;



// Algorytm jazdy


if(ready == 1){
  if(lost == 1 && last_sighted == 1){
    analogWrite(RIGHT_MOTOR_FORWARD, TurnSpeed);
    analogWrite(LEFT_MOTOR_BACKWARD, TurnSpeed);
    analogWrite(LEFT_MOTOR_FORWARD, 0);
    analogWrite(RIGHT_MOTOR_BACKWARD, 0);
  }
  else if(lost == 1 && last_sighted == 2){
    analogWrite(RIGHT_MOTOR_FORWARD, 0);
    analogWrite(LEFT_MOTOR_BACKWARD, 0);
    delay(10);
    analogWrite(LEFT_MOTOR_FORWARD, TurnSpeed);
    analogWrite(RIGHT_MOTOR_BACKWARD, TurnSpeed);
  }
  else{
    analogWrite(RIGHT_MOTOR_BACKWARD, 0);
    analogWrite(LEFT_MOTOR_BACKWARD, 0);
    delay(10);
    analogWrite(LEFT_MOTOR_FORWARD, rightMotorSpeed);
    analogWrite(RIGHT_MOTOR_FORWARD, leftMotorSpeed);
  }
}
else{
  analogWrite(LEFT_MOTOR_FORWARD, 0);
  analogWrite(RIGHT_MOTOR_FORWARD, 0);
```

```
    analogWrite(LEFT_MOTOR_BACKWARD, 0);
    analogWrite(RIGHT_MOTOR_BACKWARD, 0);
  }

}
```

Listing1. Algorytm poruszania się linefollowera.

Program w pierwszej kolejności odczytuje wartości czujników i zapisuje je w tablicy sensorValues. W zmiennej position zapisano liczbę odpowiadającą położeniu, gdzie robot wykrył linię. Każdy czujnik odczytuje inne wartości, które zależą od położenia toru. Im bliżej czujnika jest czarna linia, tym większą wartość pokaże. Zależność zmiennej position od ułożenia robota jest następująca:

- jeśli wartość wynosi 0, linia jest całkowicie po lewej stronie,
- jeśli wartość wynosi 7000, linia jest po prawej stronie,
- jeśli wartość wynosi 3500, linia jest na środku pod robotem.

Zapamiętywanie ostatniej linii pozwala robotowi na ponowne odnalezienie drogi w przypadku zgubienia się. Program porównuje wartość pierwszego i ostatniego czujnika, i za pomocą diody LED na mikrokontrolerze wyświetla kolor czerwony lub zielony w zależności od tego, po której stronie listwy pojawi się linia. Jeśli wartość sensorValues[0] jest większa lub równa 800, oznacza to, że z lewej strony wykryto linię. Gdy zmienna last_sighted nie jest ustawiona na wartość odpowiadającą pojawieniu się linii z lewej, zmienia się jej wartość na 1. To samo sprawdza się w przypadku, kiedy linia zostanie wykryta przez czujnik z prawej strony.

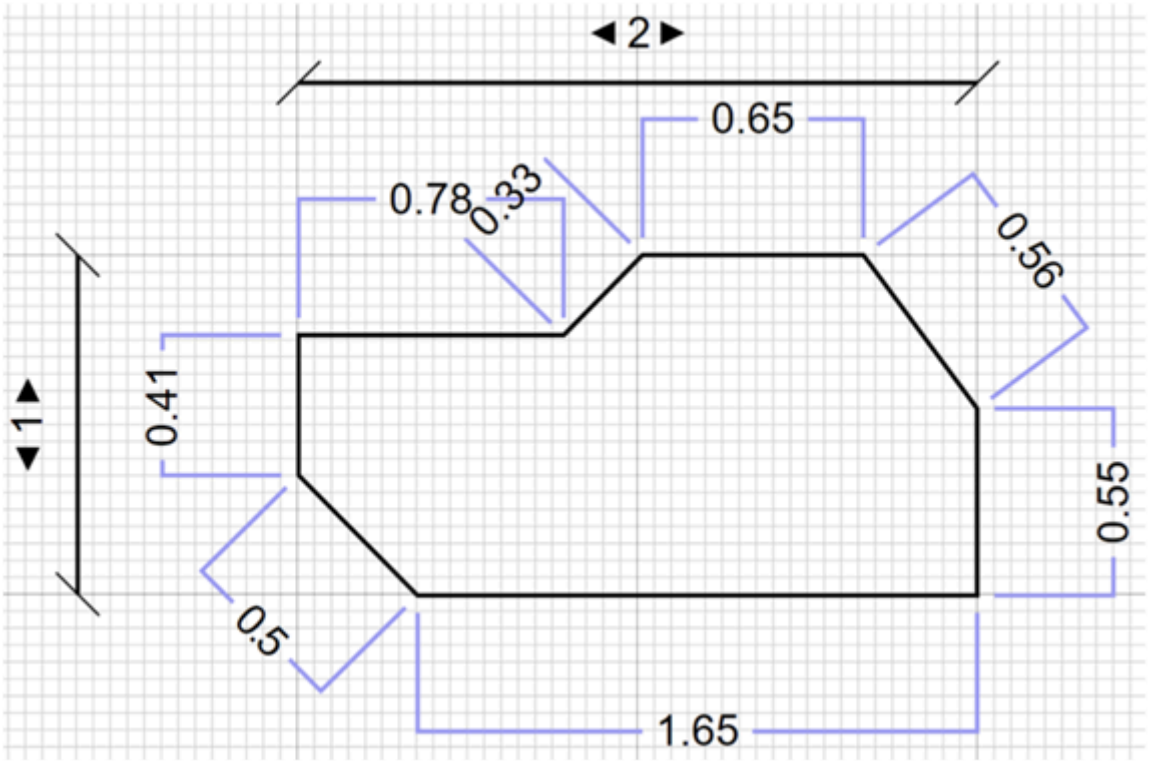
Zgubienie trasy może prowadzić do całkowitej dezorientacji robota. Zmienna lost przechowuje informacje, czy czujnik zgubił linię (wartość 1) lub jej nie zgubił (wartość 0). lost_sensors zlicza liczbę czujników, które nie wykrywają linii. Pętla przechodzi przez każdy czujnik, aby sprawdzić, który z nich wykrywa białe tło. Jedynie w przypadku, kiedy żaden czujnik nie może określić położenia linii, można stwierdzić, że robot się zgubił. Celem tego fragmentu kodu jest monitorowanie stanu linefollowera. Jeżeli każdy czujnik wskazuje wartość poniżej progu lost_threshold, oznacza to, że zgubiono trasę, i do flagi lost jest przypisana wartość 1. Robot następnie podejmuje działanie, czy należy poszukiwać linii, czy się zatrzymać.

Regulator PID (Proporcjonalno-Różniczkujący) kontroluje prędkość silników, wspomaga poprawne skręcanie i jest kluczowy przy utrzymaniu robota na ścieżce. Error to różnica pomiędzy pozycją środkową (3500) a aktualną pozycją, określa, jakie jest odchylenie robota. Jeśli position jest mniejsze niż 3500, robot jest po lewej stronie linii, a jeśli większe, to po prawej. Inicjalizacja algorytmu PD na zmiennej motorSpeed polega na zsumowaniu iloczynu błędu i wartości Kp oraz iloczynu Kd i różnicy błędu i zmiennej lastError. Cztery linie kodu ograniczają prędkość silników do wartości MaxSpeed, która określa, z jaką prędkością może maksymalnie poruszać się robot na prostej linii.

Ostatni fragment kodu kontroluje ruchem robota na podstawie stanu gotowości i widoczności linii. W przypadku zgubienia robot stara się odnaleźć linię, skręcając w stronę, w której ostatnio była widoczna. Kiedy czujniki wykrywają linię, linefollower jedzie prosto z prędkością obliczoną wcześniej. W przypadku braku gotowości robota, silniki się zatrzymują.

7. Testy

Robot został przetestowany na trasie o szerokości jednego metra i długości dwóch metrów. Testy były przeprowadzane w dobrze oświetlonym pomieszczeniu z użyciem światła sztucznego.



Rysunek 9. Trasa.
Źródło: Opracowanie własne przy użyciu: https://www.smartdraw.com/floor-plan/architecture-software.htm (dostęp: 25.05.2024).

Robot, po przejechaniu trasy dziewięciokrotnie, był wyłączany w celu naładowania baterii. Napięcie na baterii nigdy nie spadło poniżej 6,1 V. Na początku wartość Kd została ustawiona na 0, aby dobrać odpowiednie wartości Kp. Dla każdej wartości Kp linefollower był trzykrotnie testowany na trasie, zaczynając z tego samego punktu.

Kp	Kd	Wartość Prędkości Bazowej	**1**	**2**	**3**		
10	0	80	x	31,350	x		31,350
5	0	80	14,842	15,193	13,589		14,541
3	0	80	12,618	11,441	25,024		16,361
2	0	80	12,942	21,024	13,701		15,889
1,7	0	80	12,811	11,608	11,739		12,053
1,5	0	80	12,513	52,935	11,049		25,499
1,3	0	80	18,115	22,105	x		20,110
1	0	80	23,121	12,182	10,882		15,395
0,9	0	80	15,285	11,005	13,924		13,405
0,7	0	80	11,643	12,261	10,070		11,325
0,5	0	80	9,339	11,332	12,683		11,118
0,4	0	80	13,725	15,632	14,952		14,770
0,3	0	80	14,035	13,770	14,275		14,027
0,2	0	80	14,919	15,753	15,924		15,532
0,1	0	80	14,731	14,942	14,325		14,666
						średnia wszystkich wyników	15,335

Rysunek 1. Tabela czasów przejazdu dla wyzerowanego Kd.
Źródło: Opracowanie własne.

Podczas testów zdarzało się, że robot całkowicie gubił trasę i nie był w stanie jej odnaleźć; najczęściej miało to miejsce przy Kp = 10. Niejednokrotnie robot wyjeżdżał poza trasę, ale po pewnym czasie wracał na nią, choć zaczynał z innego punktu. W takich przypadkach czas był mierzony do momentu, aż robot przejechał pełną trasę jednokrotnie. Bateria była ładowana czterokrotnie w trakcie testów. Obliczono

wartości średniego czasu dla każdej wartości Kp oraz średnią dla wszystkich czasów. Średnie większe od średniej ogólnej oznaczono czerwonym kolorem, a wartości mniejsze - zielonym. Linefollower najszybciej pokonywał trasę przy Kp 0,5 oraz 0,7. Dla tych wartości testowano następnie wartości Kd. Tabela 2 przedstawia tabelę wyników dla wartości Kp równej 0,7.

Kp	Kd	Wartość Prędkości Bazowej	**1**	**2**	**3**		Średnia
0,7	20	80	26.482	17,937	11,917		14,927
0,7	15	80	14,390	15,131	11,229		13,583
0,7	13	80	11,392	8,630	13,830		11,284
0,7	10	80	11,782	11,482	11,294		11,519
0,7	9	80	11,592	10,362	8,821		10,258
0,7	8	80	8,590	9,241	9,268		9,033
0,7	7	80	10,103	9,759	8,659		9,507
0,7	6	80	8,934	9,370	8,260		8,855
0,7	5	80	10,092	8,927	9,238		9,419
0,7	4	80	12,482	12,012	13,118		12,537
0,7	3	80	14,928	14,382	13,482		14,264
0,7	1	80	17,482	15,392	15,382		16,085
0,7	0.7	80	14,281	15,382	13,382		14,348
0,7	0.5	80	13,927	14,827	18,398		15,717
0,7	0.1	80	14,291	13,193	15,382		14,289
						średnia wszystkich czasów	12,375

Tabela 2. Tabela czasów przejazdu dla pierwszej wartości Kp.

Źródło: Opracowanie własne.

Po opracowaniu wyników wyliczono średnią dla każdego czasu, aby wyznaczyć które czasy dla danej wartości Kd są średnio najniższe. Czerwonym kolorem oznaczono wartości większe od średniej ogólnej, a wartości mniejsze oznaczono zielonym. Tabela 3 przedstawia tabelę wyników dla wartości Kp równej 0,5.

Kp	Kd	Wartość Prędkości Bazowej	**1**	**2**	**3**		Średnia
0,5	20	80	16,382	14,382	14,927		15,230
0,5	15	80	14,284	21,482	14,827		16,864
0,5	13	80	14,284	15,382	16,372		15,346
0,5	10	80	29,382	14,382	11,382		18,382
0,5	9	80	13,124	14,582	13,978		13,895
0,5	8	80	14,204	13,283	13,07		13,519
0,5	7	80	14,25	11,238	14,084		13,191
0,5	6	80	13,12	12,149	11,733		12,334
0,5	5	80	10,128	9,535	12,581		10,748
0,5	4	80	9,572	8,489	11,183		9,748
0,5	3	80	9,382	10,285	11,829		10,499
0,5	1	80	12,468	11,482	14,724		12,891
0,5	0.7	80	16,391	17,382	13,294		15,689
0,5	0.5	80	12,482	13,183	21,252		15,639
0,5	0.1	80	12,642	11,525	15,25		13,139
						średnia wszystkich czasów	13,808

Tabela 3. Tabela czasów przejazdu dla drugiej wartości Kp.
Źródło: Opracowanie własne.

Wybór wartości Kp i Kd zależy od prędkości robota. Zaprezentowane wyniki mogą się różnić dla innych wartości prędkości bazowej. Określenie idealnych wartości dla robota jest niemożliwe, można jedynie stworzyć zestaw najkorzystniejszych Kp i Kd dla danej trasy. Dla prędkości 80 i Kp 0,7 najkorzystniejsze są wartości od 5 do 8 dla Kd. Natomiast dla prędkości 80 i Kp 0,5 są to wartości od 3 do 5 dla Kd.

Można więc stwierdzić, że istnieje pewna zależność pomiędzy Kp a Kd. Kd powinno być od 7 do 10 razy większe od wartości Kp, aby robot przebywał trasę w najkorzystniejszym czasie.

Część 3

8. Podsumowanie

Linefollower to bardzo popularny typ robota, chętnie budowany przez osoby o różnym stopniu zaawansowania i doświadczenia. Jest on obecny w wielu kategoriach na ogólnopolskich i międzynarodowych zawodach, zarówno w Polsce, jak i na całym świecie. Prostota tego robota tkwi w niewielkiej ilości wymaganych komponentów oraz przejrzystych zasadach jego działania.

Omówiono kompleksowo konstrukcję robota, w tym wybór użytego sprzętu, takiego jak mikrokontroler, czujniki, silniki, koła, moduł sterownika oraz moduł zasilacza obniżającego napięcie. Budowa wymagała jedynie podstawowych umiejętności manualnych, a program jest logiczny i nie wymaga wielu zabezpieczeń przed błędami jazdy. Kluczowym elementem było omówienie teorii oraz algorytmu PID, które są zaimplementowane w kodzie w sposób klarowny. Dzięki temu robot dobiera wartości prędkości dla silników w zależności od położenia na trasie.

Przeprowadzono testy w celu oceny wydajności robota. Wyniki wykazały, że robot działa zgodnie z oczekiwaniami i jest w stanie samodzielnie przejechać trasę bez pomocy z zewnątrz, zachowując precyzję i dobrą prędkość. Wyprowadzono wniosek z testów, który stwierdził, że wartość Kd powinna być 7-10 razy większa od wartości Kp. Linefollower stanowi skuteczne i edukacyjne narzędzie, które może być wykorzystywane nie tylko w zawodach, ale również w celach edukacyjnych. Zaprezentowane wyniki mogą posłużyć jako podstawa do dalszych badań i ulepszania robota.