Piotr Borowy

Klara Popławska

SPRAWOZDANIE
LABORATORIUM z WR

1. Typ algorytmu podążania za linią oraz krótka jego charakterystyka

Nasz program składa się de facto z dwóch różnych algorytmów: do podążania za linią oraz omijania przeszkody. Oba są praktycznie niezależne, program przełącza się miedzy tymi trybami na podstawie jednej z dwóch informacji: albo w trybie podążania za linią zobaczy przeszkodę, albo gdy w trybie omijania znajdzie się z powrotem na właściwym torze.

Algorytm podążania za linią jest naszym autorskim, acz z racji prostoty zakładam że nie jesteśmy raczej pionierami. Używamy w sumie trzech prostych zmiennych przechowujących pamięć o poprzednich przebiegach pętli głównej. Wykorzystując je oraz dane dostępne z czujników koloru, dokonujemy decyzji o poruszaniu się.

Algorytm do omijania przeszkody polega na odczytywaniu danych z dwóch czujników odległości, czyli odczytaniu w jakim położeniu znajduje się robot względem przeszkody i decyzji co zrobić dalej. Jest to algorytm podzielony na kilka stanów działania.

2. Opis implementacji algorytmu

Kod głównej funkcji:

```
def main_loop():
    running = False #flaga oznaczająca tutaj stan spoczynku
    setup(); #funkcja kalibracji
    initLast() #inicjalizacja wartości pamięci
    while True:
      read_input() #czytanie i przetwarzanie wejść
      if bUpState or running == True: #włączanie górnym
                                      #przyciskiem
            running = True
            if should_bypass():#podjęcie decyzji w którym
                               #trybie działa
                  bypass() #funkcja główna trybu omijania
            else:
                  follow() #funkcja główna trybu podążania za linią
      if bDownState: #wejście w tryb spoczynku po wciśnieciu
                     #dolnego przycisku
            initLast() #zerowanie zmiennych
            running = False
            stop() #zatrzymanie silników
```

Opis podążania za linią:

Wykorzystujemy trzy zmienne przechowujące informacje o poprzednim przebiegu pętli głównej. W naszym programie nazywają się one: “**lastWasRight**”, “**lastWasLeft**” oraz “**iterator**”.

O tyle o ile znaczenie pierwszych dwóch nasuwa się samo, są to oznaczenia czy w ostatni obiegu nie skręcaliśmy w prawo bądź lewo, o tyle zastosowanie iteratora jest mniej oczywiste. Służy on do tego, byśmy wiedzieli jak “długo” już skręcamy w lewo lub prawo. Jeżeli dopiero co “zaczynamy” zakręcanie w danym kierunku: skręt jest lekki, jeżeli trwa to już dłużej niż jeden przebieg: robot skręca nieco szybciej, gdy pewna wartości zostanie przekroczona a robot nadal nie znalazł się z powrotem na dobrej ścieżce: zaczyna bardzo ekstremalnie zakręcać w tym kierunku w którym powinien.

Skoro już wiemy na jakiej podstawie robot wie z jaką prędkością skręcać, jaka jest więc logika decyzji by jechać do przodu bądź zakręcać?

Odpowiedź jest banalnie prosta. Korzystając z dwóch czujników koloru na przodzie robota, oddalonych od siebie na odległość nieco większą od szerokości linii podejmujemy decyzje w następujący sposób:

- Jeśli mamy 2 odczyty koloru białego i ostatnio nie skręcaliśmy: jedź do przodu.
- Jeśli tylko lewy czujnik widzi kolor czarny - skręć w lewo
- Jeśli tylko prawy czujnik widzi kolor czarny - skręć w prawo
- Jeśli oba czujniki dają kolor czarny - jedź do przodu (dzięki temu przejeżdżamy przez skrzyżowania)
- Jeśli oba czujniki widzą kolor biały a ostatnio skręcaliśmy - skręcaj dalej w tym samym kierunku

Opis omijania przeszkody:

Algorytm omijania przeszkody jest podzielony na 4 stany. Wchodzimy w pierwszy z nich w momencie zauważenia przeszkody w odległości około 10 cm przed robotem. Od tego momentu program będzie wchodził poprzez pętlę główną do funkcji **“bypass()”**, aż do chwili zakończenia manewru. W pierwszym stanie robot cofa się na odległość koło 17,5 cm. Gdy osiągnie pożądaną odległość, wchodzimy w stan drugi - skręt w na lewo od przeszkody. Następnie wchodzimy w stan trzeci, czyli jazda wzdłuż przeszkody. Zrealizowane jest to na zasadzie wyrównywania odległości od przeszkody, korzystając z sonaru po prawej stronie robota. Gdy odległość od przeszkody jest mniejsza od 9

centymetrów, wtedy robot odbija w lekko lewo, gdy jest większa to skręca lekko w prawo. Gdy natomiast ta odległość jest większa od 15, uznajemy, że robot ominął przeszkodę, i wchodzimy w ostatni stan gdzie robot skręca w prawo i poszukuj czarnej linii do skutku. W momencie jej znalezienia, program przełącza się z powrotem do trybu śledzenia linii.

3. Sposób dobierania parametrów.

By dobrać parametry koloru, dla których czujniki uważają, że widzą kolor biały, lub czarny przed uruchomieniem robota do jazdy celuje się podane czujniki na pola białe i czarne. Następnie dla danego czujnika bierzemy średni wynik z koloru czarnego i białego, powstały wynik jest minimalną wartością dla którego robot odczytuje, że widziany kolor to biały.

Parametry prędkości zostały ustalone doświadczalnie, by z jednej strony robot jechał szybko, a z drugiej, by poprawnie odczytywał dane i odpowiednio na nie reagował.

4. Zalety i wady zaproponowanego rozwiązania

Zaletą tego typu rozwiązania jest możliwa duża prędkość robota. Wadą to, że w pewnych “pechowych” układach zakrętów, zdarza mu się niestety wypaść z trasy. Dodatkowym potencjalnym problemem jest ustawienie parametrów skrętu w lewo, przy omijaniu przeszkody, na sztywno. Gdyby przeszkoda byłaby odpowiednio szersza niestety robot by sobie z nią nie poradził.

5. Opis budowy robota.

Robot został zbudowany z klocków LEGO Mindstorms, z użyciem głównej kostki z najnowszej wersji EV3. Wykorzystuje on: 2 sensory koloru, 2 sensory odległości, 2 silniki i przyciski wbudowane w kostkę. Czujniki koloru znajdują się na przodzie robota, w odległości odrobinę większej od szerokości linii od siebie. Czujnik podczerwieni skierowany jest na wprost robota, zaś sonar znajduje się na jego prawym boku. Robot sterowany jest na zasadzie napędu różnicowego, dodatkowo w celu utrzymania stosownej równowagi, zastosowaliśmy pasywne koło wleczone.