# Sprawozdanie z projektu zaliczeniowego
# MAZE SOLVER

| Techniki Mikroprocesorowe I | Łukasz Hajec, Mateusz Kaleta |
|---|---|

## 1. Wstęp.

Celem naszego projektu było skonstruowanie i zaprogramowanie robota potrafiącego poruszać się po labiryncie i rozpoznawać poszczególne rodzaje trasy. Podstawowym założeniem jeśli chodzi o algorytm było wykorzystanie zasady lewej ręki. Mówi ona, że jeżeli przyłożymy lewą rękę do ściany labiryntu i zaczniemy poruszać się wzdłuż ściany (lub w naszym przypadku: wzdłuż linii) - znajdziemy wyjście. Do budowy projektu wykorzystaliśmy mikroprocesor ATmega 328P na płytce Arduino UNO, dwa silniki DC wraz ze sterownikiem L298N, pięć czujników odbiciowych podczerwieni z komparatorem TCRT5000. Przy programowaniu mikroprocesora korzystaliśmy ze środowiska Atmel Studio.

## 2. Realizacja.

Sterowanie silnikami odbywa się przy wykorzystaniu 6 pinów, wysokie napięcie na jednym z dwóch (PORTD) decyduje o kierunku obrotu danego silnika, przy wykorzystaniu PWM (PORTB) dostosowujemy prędkość silników. Do działania sygnału PWM niezbędne są funkcje *setPWM* (ustawia potrzebne rejestry) i obsługa przerwań *ISR(TIMER1_OVF_vect),* możemy także dowolnie modyfikować *dutyCycle* (współczynnik wypełnienia) w trakcie działania programu.

Odczyt z wyjść analogowych czujników TCRT5000 jest możliwy dzięki skorzystaniu z przetwornika ADC. Do jego konfiguracji niezbędna jest funkcja *setupADC*. W *ISR(ADC_vect)* przypisujemy odczyty do zmiennej tablicowej *pomiary[ ].* Następnie dzięki poziomom napięcia uzyskanym w funkcji *kalibracja*, odpowiadającym odczytowi podłoża i czarnej linii odczytane wcześniej pomiary konwertujemy na 0(podłoże) lub 1 i zapisujemy do *pomiary_[ ]*. Ma to zastosowanie praktyczne, ponieważ nasz robot może działać w różnych warunkach, natomiast wartości 0 i 1 wprowadzają większą przejrzystość kodu.

Najważniejszym elementem naszego programu jest wewnętrzna pętla, w której korzystamy z instrukcji warunkowej *switch*, w zależności od parametru *mode* sygnalizującego czy robot znajduje się na normalnej trasie, skrzyżowaniu, skręcie w prawo itd. W zależności od danej sytuacji mikrokontroler wykonuje odpowiednie instrukcje z zachowaniem zasady lewej ręki, tzn. np. jeżeli robot znajduje się na skrzyżowaniu w kształcie T, powinien skręcić w lewo, jeżeli na skrzyżowaniu prosto-prawo, powinien pojechać prosto.

Sprawdzanie parametru *mode* jest wykonywane przy pomocy funkcji *decyduj*. Gdy czujniki wskażą sytuację inną niż poruszanie się po trasie śledząc linię (mode = 0), funkcja ta jest wywoływana. Dokonuje ona 11 pomiarów na odległości kilkunastu centymetrów na podstawie, których wybiera odpowiedni *mode.* W kolejnej iteracji głównej pętli programu wykona się odpowiedni manewr.

## 3. Podsumowanie, wnioski.

Głównym problemem podczas realizacji naszego projektu była bardzo niedokładna praca silników DC, prawdopodobnie spowodowana zastosowaniem nieoptymalnych rozwiązań do zasilania całego układu. Razem z wszelkimi niedokładnością pracy czujników i ich ustawienia jest to też główna przyczyna tego, że osiągnęliśmy niższą skuteczność w pokonywaniu trasy labiryntu, niż zakładaliśmy na początku. Wiele zastosowanych rozwiązań programowych służy do minimalizacji tego problemu, co jednak nie jest w pełni możliwe z poziomu samego zaprogramowania mikrokontrolera.

Udało się jednak w pełni zaimplementować potrzebny algorytm. Robot dzięki kilku rozwiązaniom zabezpieczającym prawie zawsze utrzymuje się na trasie i mimo problemów z rozpoznawaniem poszczególnych rodzajów trasy, problemów z odczytami czujników, problemów przy wykonywaniu manewrów jest w stanie prędzej czy później dotrzeć do końca labiryntu i zasygnalizować to migającą diodą.  Uzyskany kod pozwala na wiele modyfikacji i zmian poszczególnych segmentów programu, co może w przyszłości pozwolić nam na poprawienie pracy robota, a także dodania funkcji optymalnego poruszania się po wcześniej pokonanej trasie.