Laboratorium Podstaw Robotyki

Politechnika Poznańska
Katedra Sterowania i Inżynierii Systemów

ĆWICZENIE 7

Budowanie lokalnej mapy otoczenia – skaner z czujnikiem podczerwieni

Celem ćwiczenia jest zbadanie wybranych właściwości pomiarowych optycznego czujnika odległości, przeprowadzenie jego kalibracji i opracowanie oprogramowania w środowisku MATLAB umożliwiającego obsługę systemu pomiarowego za pośrednictwem łącza RS-232. Końcowy rezultatem ćwiczenia jest pobranie danych ze skanera optycznego i wykreślenie na tej podstawie dwuwymiarowej lokalnej mapy otoczenia.

1 Wprowadzenie

W zadaniu sterowania i nawigacji robotów, w szczególności robotów mobilnych, duże znaczenie ma znajomość cech geometrycznych środowiska zewnętrznego, które prezentuje się w postaci map bitowych lub wektorowych [4].

Pozyskiwanie informacji o otoczeniu i tworzenie mapy otoczenia może odbywać się on-line (czyli w trakcie wykonywania zadania np. eksploracji terenu, omijania przeszkód, itd.) lub off-line. Druga metoda sprowadza się do określania mapy statycznej, która może być uwzględniona na etapie planowania bezkolizyjnej ścieżki. Następnie informacja ta może być uaktualnia w trakcie wykonywania zadania, co pozwala reagować na dynamicznie zmieniające się warunki środowiska (np. zmianę położenia innych obiektów i przeszkód).

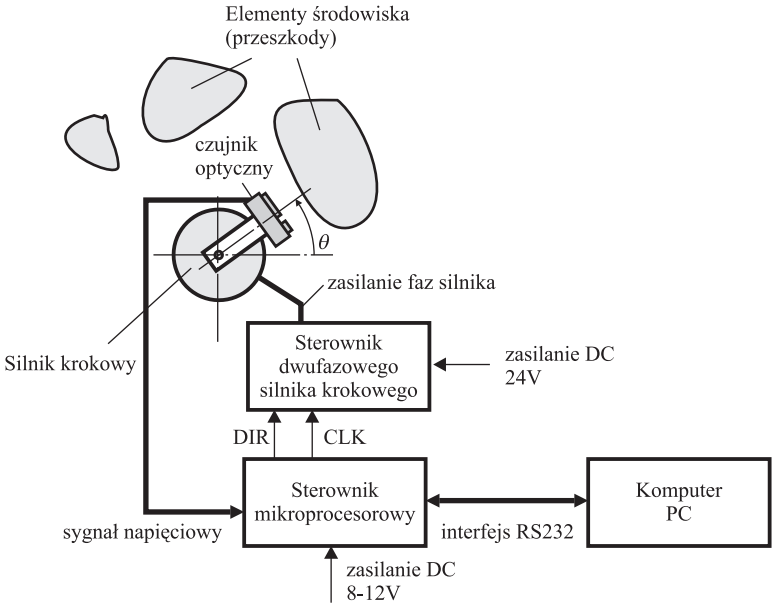
W celu akwizycji danych, w szczególności w trybie on-line, niezbędne jest wyposażenie robota w odpowiedni zestaw sensorów zewnętrznych. Spośród wielu rozwiązań wyróżnić można m. in. systemy wizyjne, czujniki optyczne oraz czujniki ultradźwiękowe [4].

W niniejszym ćwiczeniu do budowy lokalnej dwuwymiarowej mapy otoczenia wykorzystany zostanie czujnik optyczny odległości działający na zasadzie triangulacji i wykorzystujący półprzewodnikowy element PSD.

2 Stanowisko laboratoryjne

Schemat stanowiska laboratoryjnego przedstawiony został na rysunku 1. W celu kontrolowanej zmiany orientacji θ czujnika optycznego typu GP2D12 firmy Sharp [3] względem podstawowego układu współrzędnych, zastosowano dwufazowy silnik krokowy. Rozwiązanie to stanowi prosty skaner przestrzenny pozwalający określić odległość czujnika pomiarowego od przeszkody, co przy znanej orientacji czujnika umożliwia znalezienie brzegu przeszkody w przestrzeni kartezjańskiej. Za sterowanie silnikiem krokowym na poziomie napięć i prądów odpowiada sterownik PWM typu 3972 firmy Allegro pozwalający na pracę z tzw. mikrokrokiem.

Sterowanie nadrzędne realizuje komputer klasy PC za pośrednictwem asynchronicznego łącza szeregowego RS232, który wykorzystywany jest do komunikacji ze sterownikiem mikroprocesorowym. Za jego pośrednictwem można wysłać rozkaz obrotu silnika o jeden krok (lub mikrokrok)



Rysunek 1: Układ dwuwymiarowego skanera optycznego

oraz rozkaz odczytu napięcia z czujnika odległości, które określa bieżącą odległość czujnika do przeszkody. Sterownik mikroprocesorowy zbudowany został w oparciu o 8-bitowy mikrokontroler RISC ATMega8 produkowany przez firmę Atmel. Układ wyposażony został w wejścia analogowe o zakresie dynamicznym od 0 do ok. 2.5$[V]$ oraz wejścia/wyjścia cyfrowe pozwalające na zadawanie bądź odbieranie sygnałów cyfrowych zgodnych ze standardem TTL. Rozdzielczość przetwarzania sygnału analogowego wynosi 10 bitów.

2.1 Zasada działania optycznego czujnika odległości wykorzystującego metodę triangulacji

Uproszczoną zasadę działania optycznego czujnika odległości, zastosowanego w ćwiczeniu ilustruje rysunek 2. Wiązka promieniowania podczerwonego wytwarzana przez diodę elektroluminescencyjną skupiana jest przez soczewkę S1 i wysyłana w kierunku jej osi optycznej. Wiązka pada na przeszkodę w punkcie P i ulega rozproszeniu. Część rozproszonej wiązki skupiana jest przez soczewkę S2 w punkcie P' i tworzy na powierzchni czujnika PSD obraz punktu P. W zależności od odległości d od przeszkody zmienia się kąt φ pod jakim wiązka pada na soczewkę S2 co z kolei powoduje zmianę położenia punktu P' na powierzchni czujnika PSD. Na podstawie rys. 2 możemy zapisać, że

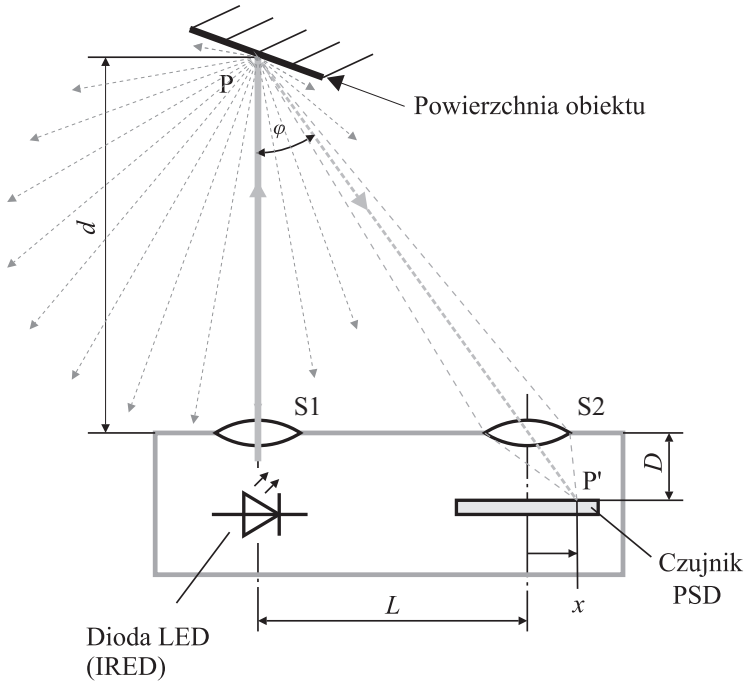
$$\tan\varphi \;=\; \frac{L}{d} \;=\; \frac{x}{D}. \tag{1}$$

Po przekształceniu otrzymujemy:

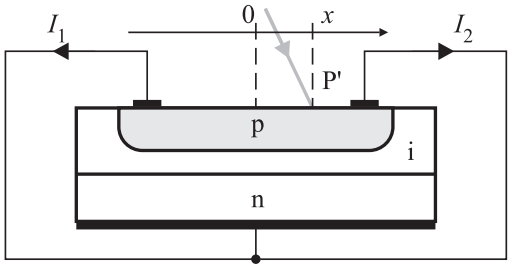
$$d \;=\; \frac{k}{x}, \tag{2}$$

gdzie $k = L \cdot D$ jest stałą geometryczną czujnika pomiarowego. Z powyższej zależności wynika, że odległość d możemy obliczyć mierząc przesunięcie x na powierzchni detektora optycznego.

W rodzinie czujników GP2Dxx [1] produkowanych przez firmę Sharp w celu określenia położenia obrazu punktu P' zastosowano półprzewodnikowy czujnik PSD (ang. *Position Sensitive Detector*) [2]. Jest to w istocie fotodioda PIN o dużej powierzchni czułej na światło, której przekrój poprzeczny przedstawia rysunek 3. Wartstwa p półprzewodnika w kształcie podłużnej linijki wystawiona jest na działanie światła. Wskutek wewnętrznego zjawiska fotoelektrycznego w punkcie na który pada



Rysunek 2: Zasada pomiaru odległości przez czujnik optyczny wykorzystujący metodę triangulacji.

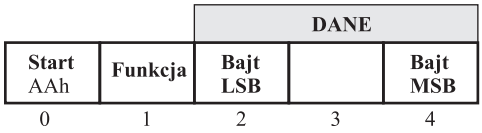


Rysunek 3: Budowa i zasada działania czujnika PSD.

światło pojawia się ładunek elektryczny, który w postaci prądów I_1 oraz I_2 przepływa do elektrody połączonej z warstwą n czujnika. W zależności od tego w którym miejscu pada światło zmieniają się wartości prądów I_1 oraz I_2 (np. w przypadku gdy promień świetlny pada dokładnie na środek linijki $I_1 = I_2$). Na podstawie wartości I_1 oraz I_2 można wyznaczyć odległość x (wyprowadzenie zależności na x można znaleźć w [2]). Okazuje się że x jest funkcją ilorazu $\frac{I_1}{I_2}$ który jest mało wrażliwy na zmiany natężenia oświetlenia (zmiana natężenia oświetlenia spowoduje proporcjonalną zmianę wartości obu prądów, ale ich iloraz pozostanie niezmieniony). Dzięki temu możliwy jest pomiar odległości od przeszkód o różnym współczynniku rozpraszania, oraz różnych lokalizacji (natężenie wiązki rozproszonej zależy między innymi od kąta rozpraszania i odległości od obiektu). Układ elektroniczny zawarty w czujniku przekształca wyznaczoną wartość x w napięcie, które jest sygnałem wyjściowym elementu.

2.2 Układ mikroprocesorowy – protokół komunikacyjny

Do komunikacji ze sterownikiem mikroprocesorowym wykorzystany został asynchroniczny interfejs szeregowy RS232C. W protokole transmisji wyróżniono urządzenie inicjujące i zarządzające (czyli Mastera, którym w tym przypadku jest komputer PC) oraz urządzenie nasłuchujące (tutaj sterownik mikroprocesorowy – Slave), które wykonuje rozkazy Mastera i zwraca informacje na żądanie. Strukturę ramki komunikacyjnej przedstawioną na rysunku 4.



Rysunek 4: Ramka komunikacyjna.

Poszczególne bajty mają następujące znaczenie:

- **Start** – znacznik rozpoczęcia ramki równy: AAh,
- **Funkcja** – określa numer rozkazu; wartości dozwolone: 01h lub 02h,
- **Dane** – trzy bajty danych; wartości dozwolone: 00 ÷ 0Fh (ważna jest tylko młodsza część bajtu[1]; starsza część bajtu nie jest interpretowana).

W systemie dostępne są dwie funkcje (rozkazy), których opis przedstawiono poniżej:

- **Funkcja = 01h** – wykonaj przesunięcie o jeden krok silnika; pierwszy bajt w polu **DANE** określa kierunek obrotu według zasady: 1 – obrót w lewo, 2 - obrót w prawo; pozostałe bajty w polu **DANE** należy wyzerować.
  Odpowiedź : jeżeli ramka nadawcza została odebrana poprawnie, urządzenie przesyła w odpowiedzi tę samą ramkę.
- **Funkcja = 02h** – zwróć aktualną wartość sygnału wyjściowego czujnika; bajty w polu **DANE** należy wyzerować.
  Odpowiedź: w polu **DANE** ramki odpowiedzi przesyłany jest 10-bitowy wynik konwersji analogowo-cyfrowej sygnału z czujnika w postaci trzech tetrad (zaczynając od najmniej znaczącej tetrady); wartość binarną wyniku konwersji d_{BIN} uzyskuje się następująco:

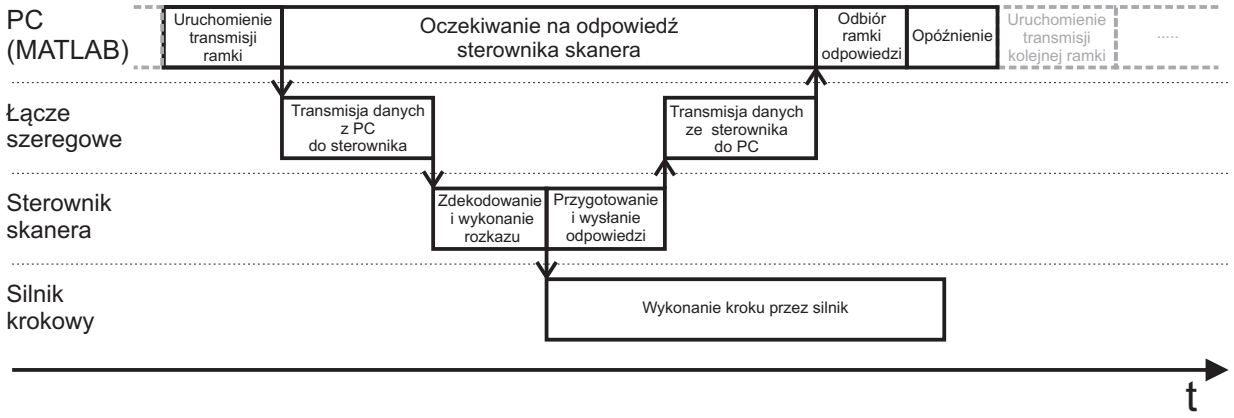
$$d_{BIN} = Bajt_2 + Bajt_3 \cdot 16 + Bajt_4 \cdot 256. \tag{3}$$

2.3 Cyklogramy funkcji sterujących

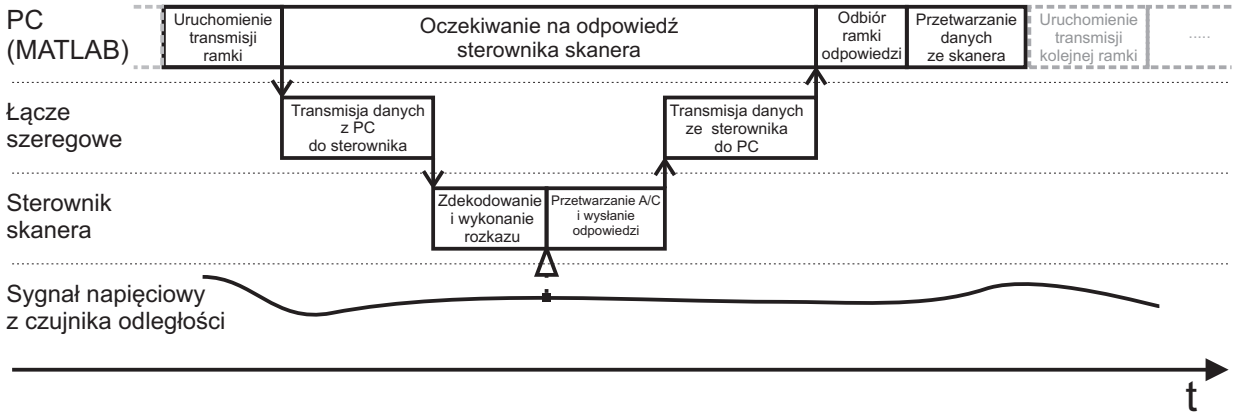
Do prawidłowego działania skanera konieczne jest zachowanie odpowiednich zależności czasowych pomiędzy rozkazami ruchu a razkazami wykonania pomiaru. W szczególności przed wykonaniem pomiaru (serii pomiarów) należy zaczekać aż silnik zatrzyma się po zakończeniu kroku (mikrokroku) i ustaną drgania. Dodatkowo należy pamiętać że dla każdego silnika krokowego istnieje pewna maksymalna częstotliwość pracy w trybie start-stop, powyżej której silnik przestaje prawidłowo działać. Na rys. 5 przedstawiono przebieg kolejnych operacji związanych z przesunięciem silnika o jeden krok. Jak widać odpowiedź sterownika dociera do komputera nadrzędnego zanim silnik zatrzyma się. Dlatego konieczne jest wprowadzenie dodatkowego opóźnienia przed wysłaniem kolejnego polecenia do sterownika.

Przebieg kolejnych operacji związanych z wykonaniem pomiaru sygnału z czujnika odległości przedstawiono na rys. 6

[1] Jest to tzw. tetrada lub inaczej kęs.



Rysunek 5: Sekwencja operacji związanych z przesunięciem silnika o jeden krok



Rysunek 6: Sekwencja operacji związanych z wykonaniem pomiaru sygnału z czujnika odległości

3 Obsługa komunikacji *komputer PC ↔ sterownik mikroprocesorowy* w środowisku MATLAB

Środowisko MATLAB od wersji 6.0 posiada wbudowane mechanizmy obsługi portu szeregowego (COM) występującego w komputerach osobistych. Poniżej podane są podstawowe informacje dotyczące wykorzystania tych mechanizmów do przesyłu danych portem szeregowym pomiędzy komputerem PC a sterownikiem mikroprocesorowym.

3.1 Obsługa portu szeregowego

Przed rozpoczęciem wymiany danych przez port szeregowy komputera w większości systemów operacyjnych i języków programowania konieczne jest jego otwarcie. Jest to operacja, która „informuje" system operacyjny, że aplikacja użytkownika chce uzyskać dostęp do danego kanału wymiany danych. W przypadku gdy kanał ten nie jest aktualnie wykorzystywany przez jakikolwiek inny proces (czyli nie został otwarty przez jakikolwiek inny program), system operacyjny zezwala na dostęp do tego medium transmisyjnego. Jako efekt operacji otwarcia portu zwracany jest zwykle identyfikator (tzw. deskryptor lub uchwyt) lub obiekt, za pośrednictwem którego realizowane są wszelkie dalsze operacje odczytu, zapisu czy konfiguracji otwartego kanału transmisji danych.

Przykładowo, w środowisku MATLAB pracującym w systemie operacyjnym Windows, otwarcie i obsługa portu szeregowego COM1 przebiega według następującego schematu:

- w pierwszej kolejności należy utworzyć obiekt, poprzez który następować będą kolejne odwołania do otwieranego portu:

```
s = serial('COM1');
```

- następnie należy skonfigurować port pod kątem parametrów transmisji wymaganych przez urządzenie zewnętrzne. Dla sterownika dalmierza odpowiednie ustawienie parametrów portu szeregowego wygląda następująco:

```
set(s,'BaudRate',115200,'DataBits',8,'Parity','None',
            'StopBits',1,'FlowControl','none');
```

- teraz można wykonać właściwe otwarcie portu:

```
fopen(s);
```

- od tej chwili możliwe jest wysyłanie i odbiór danych. Z poziomu środowiska MATLAB możliwe jest wysyłanie dwóch rodzajów danych: danych tekstowych oraz danych binarnych. Do wysyłania danych tekstowych (w postaci znaków czytelnych dla użytkownika) służy funkcja `fprintf(s,'Znaki do wyslania')`. Jednakże w przypadku komunikacji z urządzeniami takimi jak np. sterownik mikroprocesorowy konieczna jest wymiana danych w formie binarnej. W tym celu należy skorzystać z funkcji:

```
fwrite(s,dane,'precyzja','trybWysylania')
```

  W powyższym wywołaniu `s` jest obiektem reprezentującym otwarty port, przez który dane mają być wysłane, a `dane` są danymi binarnymi (np. wektorem) które mają być kolejno wysłane portem `s`. Parametr `'precyzja'` określa liczbę bitów przeznaczonych na każdy znak i jak bity te mają być interpretowane. W tym miejscu możliwe są następujące wartości:

  - `uchar, schar, char` czyli odpowiednio: wartość 8-bitowa bez znaku, 8-bitowa ze znakiem, wartość 8-bitowa ze znakiem lub bez znaku;
  - `int8, int16, int32` - wartości całkowite ze znakiem odpowiedniej długości;
  - `uint8, uint16, uint32` - wartości całkowite bez znaku odpowiedniej długości;
  - `single, float32, float` - wartości zmiennoprzecinkowe, 32-bitowe;
  - `double, float64` - wartości zmiennoprzecinkowe, 64-bitowe;

  W przypadku komunikacji z modułem sterownika skanera wykorzystujemy format `uint8`. Parametr `'trybWysylania'` (opcjonalny) może przyjmować jedną z następujących dwóch wartości:

  - `'sync'` - tryb domyślny, synchroniczny, powodujący zatrzymanie wykonywania dalszej części programu do momentu, aż wszystkie dane nie zostaną wysłane,
  - `'async'` - tryb asynchroniczny, w którym nie występuje oczekiwanie na wysłanie danych a dalsze instrukcje są natychmiast wykonywane.

  Odbiór danych otwartym wcześniej łączem szeregowym w środowisku MATLAB realizują funkcje `fscanf(...)` oraz `fread(...)`, przy czym podobnie jak to miało miejsce przy wysyłaniu danych, pierwsza służy do odczytu znaków tekstowych natomiast druga – do odbioru danych binarnych. **UWAGA: Przed odczytaniem danych warto pobrać informację o ilosci bajtów danych dostepnych do odczytu:**

```
ileDanych = s.BytesAvailable;
```

Jeżeli wiemy ile bajtów danych powinno być w buforze to możemy wstrzymać wykonanie dalszych operacji aż do nadejścia pełnej odpowiedzi z urządzenia (wykorzystując jedną z instrukcji warunkowych). Do odczytu danych pomiarowych ze sterownika skanera należy wykorzystać funkcję`fread(...)`, wywołaną w jednej z następujących postaci:

```
dane = fread(s,iloscDanychDoOdebrania)
dane = fread(s,iloscDanychDoOdebrania,'precyzja')
[dane, iloscDanychOdebranych] = fread(s,iloscDanychDoOdebrania)
[dane, iloscDanychOdebranych, komunikat] = fread(s,iloscDanychDoOdebrania)
```

Dane binarne odebrane z portu `s` zwracane są w wektorze `dane` a ich faktyczna ilość, w przypadku dwóch ostatnich postaci wywołania funkcji `fread(...)`, może być zwrócona w zmiennej `iloscDanychOdebranych`. Jako wartość argumentu `iloscDanychDoOdebrania` należy podać maksymalną liczbę danych do odebrania, przy czym liczba ta nie jest ilością bajtów, lecz ilością wartości zapisanych w postaci określonej przez argument `precyzja` dla funkcji `fwrite(...)`. Parametr `komunikat`, obecny w ostatniej postaci wywołania zawierał będzie tekstową informację o przyczynie niepowodzenia w przypadku, gdy wykonanie instrukcji `fread(...)` zakończy się niepowodzeniem.

- należy pamiętać, aby po zakończeniu wymiany danych łączem szeregowym zamknąć otwarty wcześniej port; w przeciwnym wypadku mogą wystąpić problemy z ponownym jego otwarciem (konieczne może okazać się zamknięcie programu MATLAB, a następnie ponowne jego uruchomienie). Zamknięcie portu wygląda następująco:

  ```
  fclose(s);
  delete(s);
  clear s;
  ```

Warto dodać, że ilość bajtów żądanych do oczytania w postaci argumentu `iloscDanychDoOdebrania` musi zmieścić się w buforze odbiorczym, w którym gromadzone są bajty napływające łączem szeregowym. Odczytanie rozmiaru tego bufora jest możliwe za pomocą instrukcji:

```
get(s, 'InputBufferSize')
% lub w zapisie "‘kropkowym"':
s.InputBufferSize
```

natomiast jego zmiana za pomocą instrukcji:

```
set(s, 'InputBufferSize', nowyRozmiarBuforaWbajtach);
```

Wartość domyślna rozmiaru bufora wejściowego wynosi 512 bajtów i jest wystarczająca dla większości zastosowań.

3.1 Do portu szeregowego komputera PC podłączyć (wyłączone!) urządzenie mikroprocesorowe wraz ze sterownikiem silnika krokowego i optycznym czujnikiem odległości. Następnie włączyć zasilanie urządzenia – do mikrosterownika doprowadzić napięcie stałe o wartości z zakresu $8 \div 12[V]$, do sterownika silnika krokowego ok. $12 \div 24[V]$ [a]. Przełącznik *Enable* sterownika silnika ustawić w pozycji *Off*, a pokrętło wyboru ograniczenia prądu ustawić w pozycji środkowej.

3.2 W środowisku MATLAB zainicjalizować port szeregowy (funkcje `serial, fopen`), do którego podłączono urządzenie mikroprocesorowe. Ustawić wymagane parametry transmisji podane w punkcie 3.1.

3.3 Zdefiniować w postaci pięcioelementowego wektora ramki realizujące: rozkaz obrócenia silnika krokowego o jeden krok w prawo, rozkaz obrócenia silnika krokowego o jeden krok w lewo oraz rozkaz wykonania pomiaru sygnału z czujnika odległości. Zweryfikować poprawność działania dwóch pierwszych rozkazów wysyłając je (funkcja `fwrite`) do urządzenia. Zaobserować ruch silnika. **Uwaga: przełączniki *Enable* oraz *Stop* sterownika silnika ustawić odpowiednio w pozycji *On* i *Off*.**

3.4 Zaimplementować odbiór ramki odpowiedzi (funkcja `fread`). Następnie wysłać ramki zapytań opracowanych w poprzednim punkcie i odczytać odpowiedź urządzenia. Zdekodować ramkę odpowiedzi rozkazu 02h, tak aby otrzymać wynik przetwarzania A/C w postaci znormalizowanej, czyli w zakresie $[0, 1)$ (surowy wynik przetwarzania (3) należy podzielić przez liczbę poziomów dyskretnych w wyjściowym słowie bitowym przetwornika A/C, tj. przez 2^{10}).

[a] W przypadku nieprawidłowego działania sterownika silnika krokowego należy go zresetować poprzez wyłączenie i włączenie zasilania.

4 Kalibracja czujnika odległości i tworzenie dwuwymiarowej mapy otoczenia

Czujnik odległości Sharp wykorzystujący zasadę triangulacji zwraca wartość pomiaru u^*, której odpowiada pewna wartość $d^* = f(u^*)$ rzeczywistej odległości głowicy czujnika od najbliższej przeszkody[2]. Zastosowany czujnik posiada silnie nieliniową charakterystykę statyczną $d = f(u)$, która w przybliżeniu odpowiada wycinkowi hiperboli (porównaj zależność (2)). Co więcej, w czujnikach rodziny GP2Dxx występuje dodatkowy wpływ nieliniowości związany z charakterystyką przetwarzania elementu PSD oraz z występowaniem strefy martwej dla niewielkich odległości. W efekcie końcowym, charakterystyka czujnika nie jest monotoniczna dla odległości mniejszych od około $0.1[m]$ (patrz [1]).

Z uwagi na rozrzut produkcyjny, wyskalowanie czujnika odległości wymaga empirycznego określenia jego charakterystyki statycznej. Na podstawie ciągu sygnałów wyjściowych czujnika u i odpowiadającym mu ciągu pomierzonych wartości odległości d można stworzyć tablicę opisującą odwzorowanie dyskretne lub ciągłą funkcję analityczną aproksymującą rzeczywistą charakterystykę czujnika $d = f(u)$. Pierwsza wymieniona technika, znana jako *look-up table* jest szczególnie skuteczna gdy liczba wyróżnionych stanów dyskretnych jest niewielka oraz krytyczny jest czas obliczeń. W drugiej metodzie można uzyskać lepszą dokładność, choć wiąże się to zwykle z wiekszą złożonością obliczeniową. Wybór struktury funkcji aproksymującej z reguły nie jest jednoznaczny i wymaga choćby przybliżonej znajomości estymowanej charakterystyki. Spośród wielu postaci funkcji aproksymujących w robotyce szczególnie często wykorzystuje się funkcje wielomianowe, które – jak wynika z twierdzenia Taylora – pozwalają w określonym przedziale aproksymować dowolną analityczną funkcję skalarną. Przyjmijmy zatem funkcję aproksymującą charakterystykę $d = f(u)$

[2] Umieszczonej przed czołem głowicy pomiarowej.

w postaci wielomianu n-tego stopnia:

$$d \triangleq a_n u^n + a_{n-1} u^{n-1} + \ldots + a_1 u + a_0 = f(u), \tag{4}$$

gdzie u jest sygnałem wyjściowym czujnika (wynikiem pomiaru), d oznacza rzeczywistą odległość między czujnikiem a przeszkodą[3], natomiast $a_n, a_{n-1}, \ldots, a_1, a_0$ są poszukiwanymi współczynnikami wielomianowej funkcji aproksymującej.

4.1 Napisać skrypt umożliwiający wykonanie serii k pomiarów (jeden po drugim) i zwracający znormalizowane dane z przetwornika A/C czujnika Sharp w postaci k-elementowego wektora (opis protokołu komunikacyjnego sterownika mikroprocesorowego - punkt 2.2) Serie pomiarów zapisywać pod różnymi nazwami w celu dalszego przetwarzania. Obliczyć wartość średnią i wariancję σ^2 sygnału u dla każdej serii korzystając ze wzoru:

$$\sigma^2 = \frac{1}{k-1}\sum_{i=1}^{k}(u_i - \overline{u})^2, \qquad \text{gdzie} \qquad \overline{u} = \frac{1}{k}\sum_{i=1}^{k} u_i. \tag{5}$$

4.2 Przed czujnikiem odległości ustawić nieprzezroczysty ekran w odległości $d = 0.15[m]$ od punktu obrotu O (rys. 7). Następnie wykorzystując skrypt z poprzedniego punktu (dla k=100) wykonać serię pomiarów i określić wartość średnią $\overline{u}$ sygnału z przetwornika A/C czujnika Sharp. Pomiary powtarzać zwiększając za każdym razem odległość o $0.05[m]$ (aż do osiągnięcia $l = 0.6[m]$) Wyniki zapisać w postaci wektorów $\boldsymbol{d} = [d_1\ d_2\ \ldots\ d_k]^T$ oraz $\boldsymbol{u} = [u_1\ u_2\ \ldots\ u_k]^T$, gdzie k oznacza liczbę serii pomiarów.

4.3 Przyjmując, że charakterystykę czujnika można przybliżyć wielomianem piątego stopnia:

$$d \triangleq a_5 u^5 + a_4 u^4 + a_3 u^3 + a_2 u^2 + a_1 u + a_0 = f(u), \tag{6}$$

wyznaczyć współczynniki $a_5 \ldots a_0$ wykorzystując funkcję `polyfit` programu matlab.

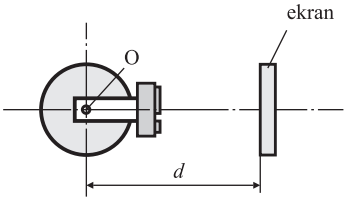
4.4 Napisać funkcję `d = LiczOdleglosc(u)`, której zadaniem będzie zwracanie wartości odległości d zgodnie z równaniem (6) na podstawie wyznaczonych w poprzednim punkcie wartości współczynników $a_5 \ldots a_0$ i sygnału u z czujnika odległości. Można wykorzystać funkcję `polyval` programu matlab.

4.5 Na jednym wykresie zaznaczyć zbiór punktów (u_i, d_i), wynikający z odebranych danych pomiarowych u_i i zadanych odległości d_i oraz wykreślić funkcję aproksymującą (6) dla $u \in [0, 1]$. Poprosić prowadzącego o sprawdzenie wyników. Ocenić czy aproksymacja charakterystyki statycznej czujnika jest zadowalająca dla założonego przedziału wartości sygnału u.

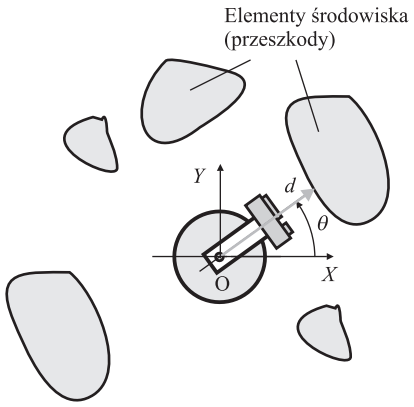
5 Tworzenie dwuwymiarowej mapy otoczenia

Przeprowadzona w poprzednim punkcie kalibracja czujnika pozwala na wykorzystanie danych pomiarowych u_i bezpośrednio do obliczania wartości odległości d_i jako funkcji $f(u_i)$. Umieszczony na platformie obrotowej czujnik Sharp daje możliwość radialnego pomiaru odległości do przeszkód znajdujacych się w lokalnym otoczeniu platformy. Akwizycja danych pomiarowych uzyskanych po pełnym obrocie platformy wraz z czujnikiem pozwala na utworzenie mapy otoczenia w postaci zbioru punktów, których aktualna odległość od osi obrotu platformy jest znana. Taką mapę można wykorzystywać do nawigacji i lokalizacji robota w środowisku rzeczywistym.

[3] Odległość d musi zostać zmierzona niezależnym narzędziem (patrz rys. 7).



Rysunek 7: Umieszczenie ekranu i pomiar odległości podczas kalibracji czujnika



Rysunek 8: Umieszczenie czujnika w środowisku z przeszkodami

5.1 Ustawić silnik krokowy z czujnikiem optycznym w otoczeniu z przeszkodami (tak aby były w polu widzenia czujnika - w praktyce w zakresie od $0.1[m]$ do $0.6[m]$) – rys. 8. Ustalić (ręcznie) początkową orientację czujnika. W środowisku MATLAB napisać skrypt zmieniający położenie wału silnika i dokonujący pomiaru odległości (wykorzystać funkcję przeliczającą wynik przetwarzania A/C na odległość). Ustawić przełączniki sterownika silnika określające tryb pracy półkrokowej. Przełączniki *Stop* i *Enable* ustawić odpowiednio w pozycji *Off* i *On*. Wykonać obrót silnika o 360° w lewo, a następnie w prawo przepisując jednocześnie wyniki pomiarów odległości i kąta obrotu do wektorów w przestrzeni roboczej MATLAB-a. Przyjąć, że jednemu pełnemu krokowi odpowiada zmiana orientacji czujnika o 1.8°.

5.2 Na podstawie otrzymanych wyników przeliczyć współrzędne biegunowe punktów na współrzędne kartezjańskie w układzie współrzędnych XOY przywiązanym do podstawy płyty mocującej silnik krokowy (porównaj rys. 8). Pomiary w których wyznaczona odległość jest większa od 0,7 [m] należy odrzucić. Wyniki zobrazować wykreślając zbiór punktów pomiarowych na wykresie dwuwymiarowym. Które wyniki można uznać za wiarygodne? Porównać otrzymaną mapę z rzeczywistą geometrią środowiska.

5.3 Powtórzyć pomiary zwiększając dokładnośc obrotu silnika poprzez ustawienie pracy mikrokrokowej – wybrać tryb 1/16 lub 1/32 kroku. Zobrazować mapę dwuwymiarową.

5.4 Zmodyfikować skrypt tak aby wizualizacja przeprowadzana była w trybie on-line i sprawdzić jego działanie. Wykorzystać polecenie `drawnow` uaktualniające wykres.

Literatura

[1] Acroname. Zasada działania czujników serii gp2dxx. `http://acroname.com/articles`.

[2] Hamamatsu. Strona internetowa. `http://www.hamamatsu.com/resources/pdf/ssd/psd_techinfo_e.pdf`.

[3] Sharp. *Strona internetowa*. `http://www.sharp-world.com/products/device/lineup/selection/opto/haca/diagram.html`.

[4] A. Woźniak, D. Dobroczyński, T. Drapikowski, T. Jedwabny, J. Majchrzak, G. Niwczyk, P. Skrzypczyński. *Autonomiczne roboty mobilne: laboratorium*. Wydawnictwo Politechniki Poznańskiej, Poznań, 1994.