Uniwersytet Zielonogórski
Instytut Sterowania i Systemów Informatycznych

Laboratorium Sterowania Robotów

Korzystanie z czujników w sterowaniu robotem Lego NXT

Uwagi wstępne

1. Wszystkie przykłady i zadania wykonujemy w środowisku MATLAB z użyciem skrzynki narzędziowej `RWTHMindstormsNXT`.
2. Komunikację z robotem rozpoczynamy od podłączenia poprzez kabel `USB`. Następnie wykorzystujemy połączenie poprzez `Bluetooth`, zgodnie z zaleceniami prowadzącego.
3. Poniższy opis został przygotowany na podstawie informacji umieszczonych na stronie: `http://www.mindstorms.rwth-aachen.de/`.

Wstęp teoretyczny

1. Do komunikacji z sensorami robota będziemy korzystali z następujących funkcji:
   - `OpenSwitch` (otwarcie dostępu do przełącznika)
   - `GetSwitch` (odczytanie stanu przełącznika)
   - `OpenSound` (otwarcie dostępu do mikrofonu, tj. czujnika poziomu dźwięku)
   - `GetSound` (odczytanie aktualnego poziomu dźwięku)
   - `OpenLight` (otwarcie dostępu do czujnika poziomu oświetlenia)
   - `GetLight` (odczytanie aktualnego poziomu oświetlenia)
   - `OpenUltrasonic` (otwarcie dostępu do czujnika ultradźwiękowego)
   - `GetUltrasonic` (odczytanie aktualnej odległości do przeszkody)
   - `CloseSensor` (zamkniecie dostępu do wskazanego sensora)

   Wymienione powyżej funkcje pozwalają nam na łatwy dostęp do standardowych sensorów robota: Sensor ultradźwiękowy korzysta z cyfrowego interfejsu dlatego jest on wewnętrznie obsługiwany inaczej niż pozostałe sensory. Korzystając jednak z funkcji wysokiego poziomu (czyli tych powyżej) nie zauważamy żadnej różnicy. Jedyną ważną informacją godną zapamiętania jest to iż funkcja `GetUltrasonic` jest około 2 razy wolniejsza niż pozostałe funkcje `Get*`. Wywołanie funkcji `CloseSensor` jest konieczne do wyłączenia danego czujnika (sensora) - to pozwala na oszczędzanie baterii. Numery portów do których podłączone są sensory oznaczone są liczbami całkowitymi od 0 do 3. Korzystając ze skrzynki narzędziowej `RWTHMindstormsNXT` używamy zdefiniowanych stałych o nazwach od `SENSOR_1` do `SENSOR_4` dla poprawienia czytelności utworzonego przez nas kodu. Przykłady zastosowań:

```
OpenSwitch(SENSOR_1);          % otwarcie dostępu do przełącznika podłączonego do portu nr 1
OpenSound(SENSOR_2, 'DB');     % musimy wybrać tryb, DB lub DBA
                                % DB - pomiar w decybelach
                                % DBA - pomiar w skali akustycznej
OpenLight(SENSOR_3, 'ACTIVE'); % musimy wybrać tryb, ACTIVE lub INACTIVE
                                %ACTIVE  - aktywna iluminacja: zapalona czerwona dioda
                                %INACTIVE - pasywna iluminacja: wyłączona czerwona dioda
OpenUltrasonic(SENSOR_4);
```

2. Teraz możemy już korzystać z podłączonych sensorów. Oto przykłady :

```
if GetSwitch(SENSOR_1) % Pamiętaj aby zawsze określić numer portu!
    % jeśli czujnik dotyku jest wciśnięty to zrób coś
end

if GetSound(SENSOR_2) < 100     % pamiętaj że wartości zmieniają się od 0 do 1023
    % czyli całkiem cicho w tym przypadku
end
```

```
if GetLight(SENSOR_3) > 1000  % pamiętaj że wartości zmieniają się od 0 do 1023
    % bardzo duża jasność
end

if GetUltrasonic(SENSOR_4) < 30 % jednostkami są [cm] odległości
    % czyli jesteśmy całkiem blisko przeszkody
end
```

Musimy również pamiętać o wyłączeniu czujników jeśli z nich nie korzystamy. Dzięki temu oszczędzamy baterię robota.

```
% nie zapomnij o tym :-)
CloseSensor(SENSOR_1);  % zamykamy dostęp na porcie nr 1
CloseSensor(SENSOR_2);   % ... i na porcie nr 2
CloseSensor(SENSOR_3);   % ... i na porcie nr 3
CloseSensor(SENSOR_4);   % ... i na porcie nr 4 też
```

3. Korzystanie z dodatkowych sensorów firmy `HiTechnic` (`www.hitechic.com`)

   Do budowy robotów LEGO NXT możemy użyć wielu dodatkowych sensorów dostarczanych przez zewnętrznych producentów. Wśród najbardziej cenionych i przydatnych sensorów są te produkowane przez firmę `HiTechnic`. W naszym przypadku będziemy korzystać z 3-ech dodatkowych sensorów:

   - akcelerometru (czujnik przyspieszenia),
   - żyroskopu,
   - kompasu.

   Pierwszym opisywanym sensorem jest akcelerometr. Aby uzyskać do niego dostęp wykorzystujemy funkcję `OpenAccelerator` wskazując numer portu do którego podłączony jest ten sensor. Dane odczytujemy przy użyciu funkcji `GetAccelerator` która zwraca wektor o rozmiarze 1x3. Poszczególne kolumny tego wektora zawierają wartości przyspieszenia odpowiednio w kierunku x, y, oraz z. Odczytana wartość 200 odnosi się to przyspieszenia 1g. Maksymalny zakres odczytywanych wartości to od -2g do +2g.

   Przykład użycia:

   ```
   OpenAccelerator(SENSOR_4); % otwieramy dostęp do sensora na pocie nr 4
   acc_Vector = GetAccelerator(SENSOR_4); % pobieramy wartości
   CloseSensor(SENSOR_4); % i zamykamy dostęp
   ```

   Kolejnym użytecznym sensorem jest żyroskop do którego dostęp uzyskujemy wywołując funkcję `OpenGyro`. Aby móc używać żyroskopu musimy go skalibrować, gdyż w innym przypadku zostanie wywołane ostrzeżenie. kalibracja może być przeprowadzona w trybie `AUTO` poprzez wywołanie funkcji `CalibrateGyro` , np.:

   ```
   offset = CalibrateGyro(port, 'AUTO')
   ```

   lub trybie ręcznym (manualnym)

   ```
   offset = CalibrateGyro(port, 'MANUAL', manualOffset)
   ```

   Po przeprowadzonej kalibracji możemy odczytywać prędkość kątową obrotu wywołując funkcję `GetGyro`.

   W większości podstawowych zastosowań (a tak to jest podczas laboratorium) wykonujemy kalibrację w trybie automatycznym, czyli wywołujemy np.:

   ```
   offset = CalibrateGyro(port, 'AUTO')
   ```

   i `offset` jest wyznaczany automatycznie. Podczas tej kalibracji dane z czujnika są odczytywane automatycznie przynajmniej 5 razy przez ok. 1 sekundy. Podczas tej operacji sensor musi być **nieruchomy**.

   W przypadku bardziej zaawansowanych zastosowań, kiedy kalibracja jest naprawdę ważna i konieczna do wykonania wielokrotnie, możemy zaoszczędzić trochę czasu poprzez użycie odczytanej już w trybie automatycznym wartości zmiennej `offset`. Aby wykonać kalibrację w trybie ręcznym należy wywołać

```
CalibrateGyro(port, 'MANUAL', manualOffset)
```

co nie wymaga utrzymywania żyroskopu w stanie nieruchomym. Należy używać wartości całkowitych dla zmiennej `manualOffset`, gdyż żyroskop ma dokładność +/- 1 stopień/sekundę.

Należy pamiętać aby dokonać kalibracji po każdorazowej zmianie środowiska pracy (temperatura, ciśnienie, itp.). Dodatkowo kalibracja jest tylko aktualna dla danego portu do którego dołączony jest żyroskop. Po zmianie portu, kalibracja musi być wykonana ponownie.

Przykłady użycia i kalibracji żyroskopu

```
% Przykład 1
% przykład kalibracji żyroskopu w trybie automatycznym

port = SENSOR_2;     % żyroskop na porcie nr 2
OpenGyro(port);      % otwieramy dostęp do żyroskopu
CalibrateGyro(port, 'AUTO'); % kalibracja w trybie auto

% teraz żyroskop jest gotowy do użycia
% przykładowo odczytujemy prędkość kątową

speed = GetGyro(port);

% dalsza część programu tutaj
% na koniec zamykamy dostęp do żyroskopu

CloseSensor(port);

% Przykład 2
% Wykorzystanie kalibracji manualnej

h = COM_OpenNXT();     % uchwyt do kostki
COM_SetDefaulNXT(h);
OpenGyro(SENSOR_1);    % otwarcie dostępu do żyroskopu

...

% jeden raz wykonujemy kalibrację automatyczną

offset = CalibrateGyro(SENSOR_1, 'AUTO');

% i przechowujemy wartość 'offset'

...

% Program główny wygląda teraz tak:
% otwieramy dostęp do żyroskopu

OpenGyro(SENSOR_1);

% i używamy wartości 'offset' określonej wcześniej

CalibrateGyro(SENSOR_1, 'MANUAL', offset);

% i możemy korzystać z żyroskopu ...
```

Ostatnim omawianym sensorem jest kompas służący do odczytu położenia względem kierunków geograficznych. Standardowo dostęp do czujnika uzyskujemy poprzez wywołanie funkcji `OpenCompass` podając jako argument numer portu do którego podłączony jest kompas. Następnie, podobnie jak to było w przypadku żyroskopu, przeprowadzamy kalibrację wywołując (jedną z poniższych)

```
CalibrateCompass(port, f_start)
CalibrateCompass(port, f_start, handle)
```

Przed rozpoczęciem kalibracji kompasu należy usunąć wszelkie obiekty metalowe znajdujące się w pobliżu czujnika. W szczególności należy wziąć pod uwagę bliskie położenie kostki NXT i silników robota. Następnie ustawiamy wartość parametru `f_start = true` (wywołując funkcję `CalibrateCompass`) aby formalnie rozpocząć procedurę kalibracji, a w celu jej zakończenia `f_start = false`. Pomiędzy zmianami tego parametru dokonujemy kalibracji czyli wykonujemy 2 pełne obroty kompasem.

Przykład kalibracji kompasu:

```
% dostęp do kompasu musi aktywny przed kalibracją
OpenCompass(SENSOR_2); % np.: kompas podłączony do poru nr 2

% startujemy procedure kalibracji kompasu
CalibrateCompass(SENSOR_2, true);

% Kompas jest przymocowany do sinika A, którym wykonujemy 2 pełne obroty
m = NXTMotor('A', 'Power', 5, 'TachoLimit', 720)
m.SendToNXT();

m.WaitFor();

% kalibracja powinna teraz się zakończyć
CalibrateCompass(SENSOR_2, false);
% i możemy teraz korzystać z kompasu
```

Wartość obrotu (w stopniach) odczytujemy wywołując funkcję `GetCompass`, gdzie argumentem jest numer portu. Odczytana wartość jest w zakresie od 0 do 360 gdzie 0 stopni oznacza północ, 90 - zachód, itd. Po zakończeniu korzystania z kompasu wywołujemy funkcję `CloseCompass`.

Zadania do wykonania

1. Zbudować robota według wskazówek prowadzącego. Robot powinien zawierać dwa silniki i 4 sensory (dotyku, ultradźwiękowy, światła, dźwięku).
2. Przetestować wszystkie programy umieszczone w powyższym wprowadzeniu teoretycznym.
3. Zaimplementować funkcję wykreślającą:
   (a) charakterystyki czułości sensora dźwięku (tj. odczytanych wartości z sensora w stosunku do zadanych dźwięków o różnej głośności) w skali dB i dBA.
   (b) charakterystyki czułości sensora światła w obu trybach pracy sensora. Pomiarów dokonujemy przy użyciu kolorowej tablicy znajdującej się w laboratorium.
4. Zaimplementuj program do poruszania robotem w przód i w tył. Zmiana kierunku jazdy ma odbywać się za każdorazowym klaśnięciem w dłonie.
5. Zbadać możliwości rozpoznania obiektów (znajdujących się w sali laboratorium) i odległości do nich przy użyciu sensora ultradźwiękowego. Zaimplementuj program pozwalający omijać robotowi obiekty znajdujące się przed robotem. Użyj sensora dotyku do określenia czy robot uderzył obiekt znajdujący się na trajektorii robota.
6. Zaimplementować program do odczytu wartości z jednego otrzymanych sensorów (akcelerometr, żyroskop, kompas).
7. Zaimplementować program do losowego ruchu robota (każdorazowo kierunek i pokonywana odległość ma być losowana). Na podstawie odczytów z akcelerometru, żyroskopu i/lub kompas należy narysować trajektorię przemieszczania się robota. Sprawdzić wyznaczoną pozycję z rzeczywistymi współrzędnymi robota.

Sprawozdanie

Sprawozdanie z przeprowadzonego laboratorium powinno zawierać:

- Kody źródłowe wszystkich utworzonych podczas laboratorium skryptów(Nie zamieszczać kodów programów przykładowych !) wraz z komentarzem.
- Wyniki pomiarów i odpowiednie wykresy zależności.
- Uwagi i wnioski.