Programowanie robota mobilnego E-puck w języku Python [1]

Joanna Ratajczak, Mirela Kaczmarek

1 Zasady bezpieczeństwa

W trakcie pracy z robotem E-puck należy zachować ostrożność. Pod żadnym pozorem nie można dopuścić aby robot spadł ze stołu. Należy unikać wszelkich sytuacji mogących spowodować uszkodzenie robota.

Podczas wykonywania programu robota można zatrzymać wywołując komendę `e.stop()`. Przerwanie wykonywania programu jest także możliwe poprzez użycie kombinacji **Ctrl-C**.

2 Opis robota

Robot E-puck (Rysunek 1) stanowi platformę dydaktyczną powstałą w ramach otwartego projektu [2].

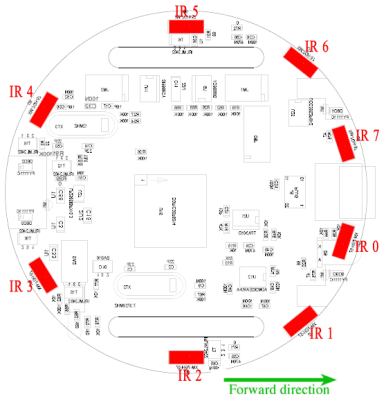
Rysunek 1: Robot E-puck [1].

E-puck jest niewielkim (średnica 7 cm, wysokość 5 cm, waga 200 g) robotem mobilnym poruszającym się na dwóch kołach sterowanych silnikami

[1] Data powstania: 24.10.2015. Ćwiczenie w ramach kursu Podstawy Robotyki realizowanego na Wydziale Elektrycznym

[2] Strona projektu: http://www.e-puck.org

krokowymi. Jest wyposażony m. in. w 8 czujników odległości IR (których rozkład przedstawiono na rysunku 2), kamerę RGB, akcelerometr 3D oraz trzy mikrofony.



Rysunek 2: Rozłożenie czujników IR [1].

3 Dostępne komendy

Dostępne są następujące komendy:

- `set_speed(lewy,prawy)` – ustawienie prędkości kół.
- `sleep(n)` – odczekanie zadanego czasu. Parametr `n` oznacza zadany czas podawany w sekundach.
- `stop()` – zatrzymanie robota.
- `read_speed()` – odczyt prędkości kół. Funkcja zwraca dwuelementową listę `[v_lewego, v_prawego]`,
- `read_proximity()` - odczyt czujników zbliżeniowych. Stan czujników zwracany jest w 8 elementowej tablicy indeksowanej od 0, zawierającej kolejno `[IR0, IR1, IR2, IR3, IR4, IR5, IR6, IR7]` zgodnie z oznaczeniami przyjętymi na rysunku 2.

- `read_light()` – odczyt czujników oświetlenia. Stan czujników zwracany jest w 8 elementowej tablicy indeksowanej od 0, zawierającej kolejno `[IR0, IR1, IR2, IR3, IR4, IR5, IR6, IR7]` zgodnie z oznaczeniami przyjętymi na rysunku 2.

- `read_proximity_avg(n)` – uśredniony odczyt czujników zbliżeniowych. Parametr `n` określa liczbę pomiarów użytych do uzyskania wyniku uśrednionego. Uśredniony stan czujników zwracany jest w 8 elementowej tablicy (analogicznie do `read_proximity()`.)

- `read_light_avg(n)` – uśredniony odczyt czujników oświetlenia. Parametr `n` określa liczbę pomiarów użytych do uzyskania wyniku uśrednionego. Uśredniony stan czujników zwracany jest w 8 elementowej tablicy (analogicznie do `read_proximity()`).

Moduł współpracy z robotem mobilnym E-Puck został napisany w wersji obiektowej, co oznacza, że wszystkie instrukcje powinny być poprzedzone odpowiednią instancją, np. `e.set_speed(100,100)`.

4 Przykład programu

Poniżej przedstawiono przykładowy program napisany w języku Python:

```
from epuck import *

# funkcja uzytkownika
def fun(e):
      e.set_speed(100,-100)
      e.sleep(1)
      e.stop()
      print "Koniec"


# uruchomienie funkcji
go(fun)
```

5 Składnia języka Python [2]

Python to przejrzysty skryptowy język programowania obiektowego, wykorzystujący przejrzystą składnię, przez co programy są łatwe do zrozumienia. Zawiera różnorodne wbudowane typy danych: liczbowych (zmiennoprzecinkowe, zespolone, liczby całkowite o nieskończonej precyzji), tekstowych, listy, słowniki, itp. Jezyk zapewnia obsługę wyjątków pozwalając efektywniej obsługiwać błędy.

5.1 Lista

Podstawowym złożonym typem danych w języku Python jest lista. Poszczególne elementy listy są oddzielone przecinkami, a całość znajduje się w nawiasach kwadratowych, np. `a=['pies', 'kot', 100, 1234]`. Elementy listy nie muszą być tego samego typu.

Dostęp do elementu listy uzyskuje się stosując operator indeksowania, np. `a[2]`. Lista jest indeksowana od 0. Użycie indeksów ujemnych powoduje zwrócenie odpowiedniej wartości elementu listy licząc od końca. Indywidualne elementy listy można dowolnie zmieniać, np: `a[2]=a[2]-3`.

5.2 Operatory

Standardowe operatory porównań sa identyczne jak w C: < (mniejszy), > (większy), == (równy), <= (mniejszy lub równy), >= (większy lub równy) i != (rożny). W języku zdefiniowane sa dwa wyrażenia logiczne: `True` oraz `False` posiadające logiczną wartość odpowiednio prawdy i fałszu. Łączenie warunków w złożone wyrażenia następuje przy pomocy operatorów `and`, `or` i `not`.

5.3 While

Poniżej przedstawiono prosty sposób użycia instrukcji *while*:

```
# liczby Fibonnaci'ego
a, b = 0, 1
while b < 10:
      print b
      a, b = b, a+b
```

Pętla `while` wykonuje się tak długo jak długo warunek (tutaj `b < 10`) jest prawdziwy. W Pythonie, podobnie jak w C, każda niezerowa wartość całkowita ma wartość logicznej prawdy, każda niepusta lista lub ciąg znaków także.

5.4 For

Instrukcja `for` w Pythonie różni się od tych stosowanych w C lub Pascalu. Instrukcja `for` w Pythonie powoduje iterację po elementach jakiejkolwiek sekwencji (np. listy lub łańcucha znaków), w takim porządku, w jakim są one umieszczone w danej sekwencji. Na przykład:

```
# Mierzy pewne napisy:
a = ['kot', 'okno', 'wypróżnić']
      for x in a:
            print x, len(x)
```

Instrukcja ta zwróci:

```
kot 3
okno 4
wypróżnić 9
```

Jeżeli zachodzi potrzeba iterowania po sekwencji liczb należy użyć funkcji `range()`, generującej listę zawierająca ciąg arytmetyczny, np:

```
range(10)
```

zwraca ciąg `[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]`. Punkt końcowy nie jest częścią wygenerowanej listy, `range(10)` generuje listę 10 wartości, dopuszczalnych indeksów listy o długości 10. Można także określić początek: `range(5,10)` oraz także krok: `range(0, 10, 3)`. Wyrażenie `break` podobnie jak w C, powoduje opuszczenie najbardziej zagnieżdżonej pętli `for`for lub `while`. Wyrażenie `continue` powoduje wykonanie pętli od początku.

5.5 If

Wyrażenie warunkowe uzyskuje się prawie identycznie jak w innych językach:

```
if x < 0:
      print 'ujemny'
elif x == 0:
      print 'zero'
else:
      print 'dodatni'
```

5.6 Grupowanie wyrażeń

W języku Python grupowanie wyrażeń odbywa się poprzez wcięcia. Każda linia tego samej grupy musi być wcięta o tę samą liczbę znaków. Koniec grupy oznacza się pustą linią. Zagnieżdżenie grup uzyskuje się przez zagnieżdżenie wcięć.

5.7 Funkcje wbudowane

Poniżej podano niektóre przydatne funkcje wbudowane:

- `len(a)` — podaje długość (liczbę elementów) `a`,
- `max(a)` — podaje wartość maksymalnego elementu listy `a`,
- `min(a)` — podaje wartość minimalnego elementu listy `a`,
- `print ...` — drukowanie wartości, podanie przecinka na końcu nie powoduje przejścia do następnej linii.

Dla listy `l` dostępne sa następujące funkcje: `l.index(w)` — zwraca indeks pierwszego elementu o wartości `w`, `w in l` — test, czy lista `l` zawiera element o wartości `w`.

Literatura

[1] http://www.e-puck.org

[2] Paweł Ludwików, *Programowanie robota mobilnego Khepera w języku Python*, 2011