ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Tương tác người – Robot

BÁO CÁO THỰC HÀNH BUỔI 2

LAB02 - Làm quen với điều khiển trên Webots

Sinh viên thực hiện

Nguyễn Huy Thắng – 22027545

Hà Nội, ngày 22 tháng 9 năm 2025

Tutorial 4: Thông tin thêm về Điều khiển

Mục tiêu của hướng dẫn đầu tiên này là giúp ta làm quen với những kiến thức về lập trình điều khiển cơ bản robot trong Webots. Ta sẽ thiết kế một bộ điều khiển đơn giản, tránh vật cản và dừng khẩn cấp. Cuối chương ta sẽ hiểu về các node, scene tree và bộ điều khiển API, cách khởi tạo các thiết bị robot và cách lấy giá trị cảm biến và lập trình một vòng phản hồi đơn giản.

I. Lý thuyết

1. Định nghĩa:

- E-puck là một robot di động thu nhỏ được phát triển tại EPFL với mục đích giảng dạy bởi các nhà thiết kế robot Khepera. Phần cứng và phần mềm của e-puck là mã nguồn mở, cung cấp quyền truy cập cấp thấp vào mọi thiết bị điện tử và khả năng mở rộng không giới hạn.
- Mô hình bao gồm động cơ bánh xe vi sai, cảm biến hồng ngoại để đo khoảng cách và ánh sáng, cảm biến gia tốc, camera, 8 đèn LED xung quanh, đèn LED thân xe, điều khiển Bluetooth (được mô phỏng bằng thiết bị Phát / Thu) và phần mở rộng cảm biến mặt đất.

Hình 1. Mô hình robot e-puck đang hoạt động

E-puck được thiết kế đáp ứng các yêu cầu sau:

- Thiết kế nhỏ gọn đơn giản.
- Tính linh hoạt: Được sử dụng trong nhiều hoạt động giáo dục, có nhiều cảm biến thực hiện nhiều tác vụ.
- Có phần mềm mô phỏng Webots giúp dễ dàng lập trình, mô phỏng.
- Thân thiện với người dùng: e-puck nhỏ gọn và dễ dàng lắp đặt, không cần dây cáp.

- Độ bền cao và bảo trì giá rẻ.
- Giá cả phải chăng phù hợp với sinh viên nghiên cứu.

2. Các loại cảm biến sử dụng trong e-puck

Các cảm biến và thiết bị	Tên chương trình
Motors	'left wheel motor' and 'right wheel motor'
Cảm biến vị trí	'left wheel sensor' and 'right wheel sensor'
Cảm biến tiệm cận	'ps0' to 'ps7'
Cảm biến ánh sáng	'ls0' to 'ls7'
Leds	'led0' to 'led7' (e-puck ring), 'led8' (body) and 'led9' (front)
Camera	'camera'
Cảm biến gia tốc	'accelerometer'
Cảm biến đất	'gs0', 'gs1' and 'gs2'

Bảng 1. Các loại cảm biến sử dụng trong E-puck

Ở hướng dẫn 4, chúng ta chủ yếu làm việc với cảm biến tiệm cận (cảm biến khoảng cách) ‘ps0’ đến ‘ps7’.

II. Thực hành Tutorial 4

1. Khởi tạo robot, sensor và motor

- **Khởi tạo robot**

```
wb_robot_init();
```

- **Khởi tạo 8 cảm biến khoảng cách ps0 -> ps7**

```
WbDeviceTag ps[8];
char                ps_names[8][4]                =                {
"ps0","ps1","ps2","ps3","ps4","ps5","ps6","ps7" };
for (int i=0;i<8;i++) {
  ps[i] = wb_robot_get_device(ps_names[i]);
  wb_distance_sensor_enable(ps[i], TIME_STEP);
}
```

- **Khởi tạo động cơ trái & phải**

```
WbDeviceTag left_motor = wb_robot_get_device("left wheel motor");
WbDeviceTag right_motor = wb_robot_get_device("right wheel motor");
wb_motor_set_position(left_motor, INFINITY);
wb_motor_set_position(right_motor, INFINITY);
```

- **Cài đặt tốc độ mặc định**

```
wb_motor_set_velocity(left_motor, 0.5 * MAX_SPEED);
wb_motor_set_velocity(right_motor, 0.5 * MAX_SPEED);
```

- **Khởi tạo bàn phím (để điều khiển STOP/START)**

```
wb_keyboard_enable(TIME_STEP);
```

2. **Hàm đọc giá trị cảm biến**

```
double ps_values[8];
for (int i=0;i<8;i++)
  ps_values[i] = wb_distance_sensor_get_value(ps[i]);
```

3. **Điều khiển tránh vật cản**

```
bool right_obstacle =
  ps_values[0] > 80.0 || ps_values[1] > 80.0 || ps_values[2] > 80.0;
bool left_obstacle =
  ps_values[5] > 80.0 || ps_values[6] > 80.0 || ps_values[7] > 80.0;
double left_speed = 0.5 * MAX_SPEED;
double right_speed = 0.5 * MAX_SPEED;
if (left_obstacle) {
  left_speed  = 0.5 * MAX_SPEED;
  right_speed = -0.5 * MAX_SPEED;
} else if (right_obstacle) {
  left_speed  = -0.5 * MAX_SPEED;
  right_speed = 0.5 * MAX_SPEED;
}
```

```
wb_motor_set_velocity(left_motor, left_speed);

wb_motor_set_velocity(right_motor, right_speed);
```

4. Các hàm di chuyển thẳng, quay trái, phải và dừng robot

- **Di chuyển thẳng**

```
left_speed  = 0.5 * MAX_SPEED;

right_speed = 0.5 * MAX_SPEED;
```

- **Quay trái**

```
left_speed  = -0.5 * MAX_SPEED;

right_speed =  0.5 * MAX_SPEED;
```

- **Quay phải**

```
left_speed  =  0.5 * MAX_SPEED;

right_speed = -0.5 * MAX_SPEED;
```

- **Dừng robot**

```
wb_motor_set_velocity(left_motor, 0.0);

wb_motor_set_velocity(right_motor, 0.0);
```

5. Kết quả mô phỏng

- Mô phỏng trên Webots, robot e-puck có thể thực hiện tự động di chuyển, khi gặp vật cản robot né hướng khác để tiếp tục di chuyển. Thuật toán di chuyển được xây dựng từ các giá trị cảm biến đọc được.
- Video chạy code mô phỏng trên Webots

Video chạy điều khiển né vật cản

III. Xây dựng API cho Robot

- Trong phần này, ta sẽ tiến hành xây dựng một số hàm API cơ bản để điều khiển robot: emergency_stop, watchdog. Ngoài ra robot còn được tích hợp bàn phím để dừng khẩn cấp khi cần thiết.

1. Emergency stop

- Khi ấn space (giá trị 32) sẽ kích hoạt `emergency_stop`

```
bool emergency_stop = false;
if (key == 32) {  // SPACE
  emergency_stop = true;
  printf("Emergency STOP activated!\n");
}
```

- Khi ấn S trên bàn phím (giá trị 83) robot có thể hoạt động trở lại

```
if (key == 83) {
     emergency_stop = false;
     printf("Robot START again!\n");
   }
```

- Dừng lại khi gọi hàm `emergency_stop`

```
if (emergency_stop) {
     wb_motor_set_velocity(left_motor, 0.0);
     wb_motor_set_velocity(right_motor, 0.0);
     continue; // bỏ qua xử lý tránh vật
   }
```

2. Hàm watch dog

- Khi robot không nhận được tín hiệu điều khiển trong một thời gian sẽ thực hiện dừng lại

```
#define WATCHDOG_TIMEOUT 1000 // ms
double last_update_time = wb_robot_get_time() * 1000; // ms
double current_time = wb_robot_get_time() * 1000; // ms
if (current_time - last_update_time > WATCHDOG_TIMEOUT) {
wb_motor_set_velocity(left_motor, 0.0);
wb_motor_set_velocity(right_motor, 0.0);
printf("Watchdog timeout! Robot stopped.\n");
break;
} else {
last_update_time = current_time;
}
```

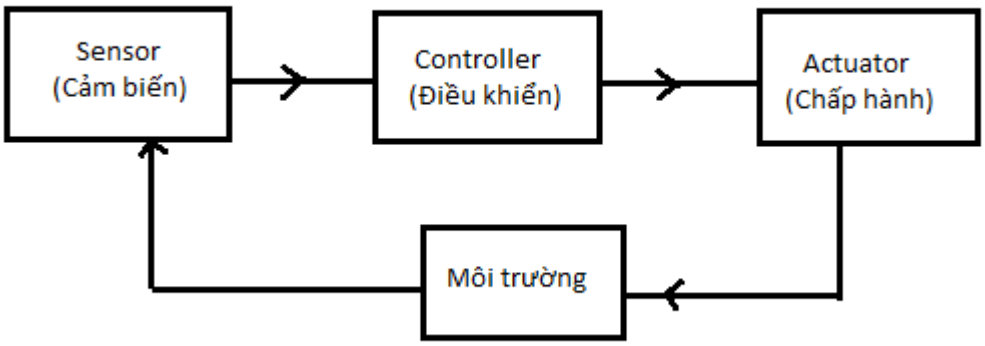
3. Kết quả

- Khi ấn nút trên bàn phím robot đã dừng lại khẩn cấp. Các hàm API đã thực hiện chức năng của nó.
- Video kết quả chạy mô phỏng trên Webots.

Video API emergency stop

IV. Sơ đồ kiến trúc kết nối

- Quan hệ giữa sensor, controller, actuator. Cảm biến sẽ nhận phản ứng môi trường đưa giá các giá trị cho bộ điều khiển và truyền hành động cho robot. Sơ đồ kiến trúc được minh hoạ dưới đây.



Hình 2. Sơ đồ kiến trúc quan hệ Sensor – Controller – Actuator.