ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

-----📖-----

BÁO CÁO THỰC HÀNH MÔN HỌC

TƯƠNG TÁC NGƯỜI – ROBOT

BÀI THỰC HÀNH SỐ 3

Sinh viên thực hiện

Nguyễn Huy Thắng - 22027545

Hà Nội, ngày 28 tháng 04 năm 2025

Tóm tắt nội dung

Trong bài thực hành này, sinh viên lập trình robot e-puck di chuyển trong môi trường mô phỏng Webots. Robot được yêu cầu:

- Trường hợp 1: Khi phát hiện "người" (vật cản phía trước), robot dừng lại và quay mặt hoặc nháy đèn LED để chào.
- Trường hợp 2: Khi có hai người đứng gần nhau, robot tự động đi vòng ra phía sau họ, tránh cắt ngang trước mặt, rồi dừng lại chào xã giao (quay đầu hoặc nháy đèn LED).

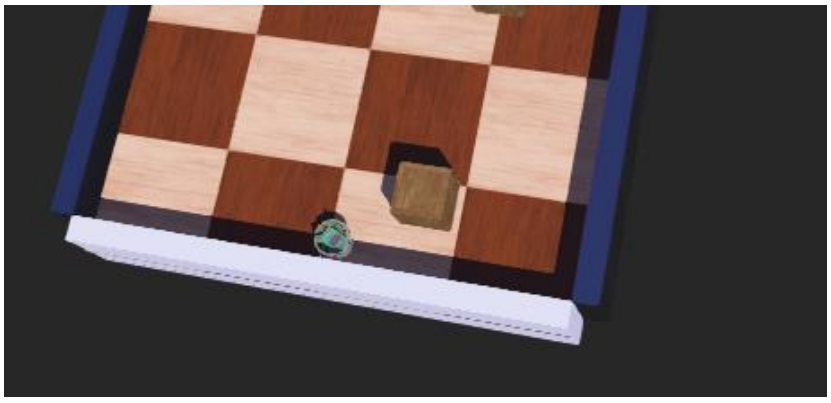
Qua bài này, robot thể hiện hành vi tương tác "xã hội" cơ bản, mô phỏng hành vi lịch sự trong môi trường có người.

1. Mục tiêu

- Làm quen với việc điều khiển chuyển động e-puck.
- Hiểu cách đọc dữ liệu từ cảm biến khoảng cách (IR) để phát hiện vật thể (người).
- Cài đặt các hành vi phản ứng tự động khi robot phát hiện người.
- Biết cách mô phỏng tương tác robot-người trong Webots.
- Xây dựng tư duy "social behavior" cơ bản cho robot phục vụ nghiên cứu HRI.

2. Thực hành

2.1. Trường hợp 1



Môi trường có vật thể "người" và robot e-puck

Ta coi môi trường không vật thể, vật thể thùng coi như người và robot e-puck coi như robot đang di chuyển và gặp người sẽ thực hiện yêu cầu: dừng lại, quay mặt phía trước và nhảy đèn để "chào".

a. Khởi tạo chương trình

```
#include <webots/robot.h>
#include <webots/distance_sensor.h>
#include <webots/motor.h>
#include <stdio.h>

#define TIME_STEP 64
#define THRESHOLD 78.0
#define MAX_SPEED 6.28

int main() {
  wb_robot_init();
```

Giải thích:

- `#include <webots/...>:` nạp các thư viện cần thiết để điều khiển robot, cảm biến và motor trong Webots.
- `robot.h`: cung cấp các hàm khởi tạo, cập nhật trạng thái robot.
- `distance_sensor.h`: dùng để đọc giá trị cảm biến khoảng cách.
- `motor.h:` điều khiển tốc độ quay của bánh xe robot.
- `#define TIME_STEP 64:` xác định bước thời gian mô phỏng (mỗi 64 ms).
- `#define THRESHOLD 78.0`: ngưỡng giá trị cảm biến để phát hiện vật cản (người).
- `#define MAX_SPEED 6.28:` tốc độ tối đa của motor e-puck (rad/s).
- `wb_robot_init():` khởi tạo mô phỏng robot, bắt buộc phải gọi trước khi dùng bất kỳ thiết bị nào.

b. Khởi tạo cảm biến khoảng cách

```
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);
 }
```

Giải thích:

- Robot e-puck có 8 cảm biến hồng ngoại (IR) đặt quanh thân để phát hiện vật cản.
- `ps_names` chứa tên của các cảm biến trong mô phỏng Webots.

- `wb_robot_get_device(ps_names[i]):` lấy thẻ định danh (tag) của cảm biến để truy cập.
- `wb_distance_sensor_enable(ps[i], TIME_STEP):` bật cảm biến và cập nhật giá trị mỗi 64 ms.

  → Ta đã có mảng `ps[0..7]` để đọc giá trị khoảng cách quanh robot.

c. Khởi tạo động cơ bánh xe

```
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);
wb_motor_set_velocity(left_motor, 0.0);
wb_motor_set_velocity(right_motor, 0.0);
```

Giải thích:

- Lấy thiết bị motor hai bánh của robot bằng tên `"left wheel motor"` và `"right wheel motor".`
- `wb_motor_set_position(..., INFINITY):` cho phép motor quay vô hạn, nghĩa là ta có thể điều khiển tốc độ (velocity control) thay vì vị trí.
- `wb_motor_set_velocity(..., 0.0):` đặt vận tốc ban đầu = 0, tức robot đứng yên khi khởi động.

d. Định nghĩa các hàm di chuyển, dừng, quay

```
void move_forward(double speed) {
  wb_motor_set_velocity(left_motor, speed);
  wb_motor_set_velocity(right_motor, speed);
}

void stop() {
  wb_motor_set_velocity(left_motor, 0.0);
  wb_motor_set_velocity(right_motor, 0.0);
}

void turn_around() {
  wb_motor_set_velocity(left_motor, 0.5 * MAX_SPEED);
  wb_motor_set_velocity(right_motor, -0.5 * MAX_SPEED);

  // Quay 180 độ (khoảng 1500 ms tương đương Python)
  int duration = 1500 / TIME_STEP;  // số bước mô phỏng cần quay
  for (int i = 0; i < duration; i++)
    wb_robot_step(TIME_STEP);
```

```
  stop();
}
```

Giải thích:

- `move_forward(speed):` hai bánh quay cùng chiều → robot đi thẳng với vận tốc speed.
- `stop():` dừng robot (vận tốc hai bánh = 0).
- `turn_around():` Bánh trái quay tiến, bánh phải quay lùi → robot quay tại chỗ.
- `duration = 1500 / TIME_STEP:` thời gian quay ~1.5 giây, tương ứng góc 180°.
- Sau khi đủ thời gian quay → robot dừng lại. → Hàm này dùng để "chào" người bằng cách quay mặt về hướng ngược lại.

e. Vòng lặp chính – Điều khiển hành vi

```
while (wb_robot_step(TIME_STEP) != -1) {
  double front_left = wb_distance_sensor_get_value(ps[7]);
  double front_right = wb_distance_sensor_get_value(ps[0]);

  printf("ps7=%.1f, ps0=%.1f\n", front_left, front_right);

  // Nếu gặp vật cản trước mặt
  if (front_left > THRESHOLD || front_right > THRESHOLD) {
    stop();
    turn_around();
    break;
  } else {
    move_forward(0.5 * MAX_SPEED);
  }
}
```

Giải thích:

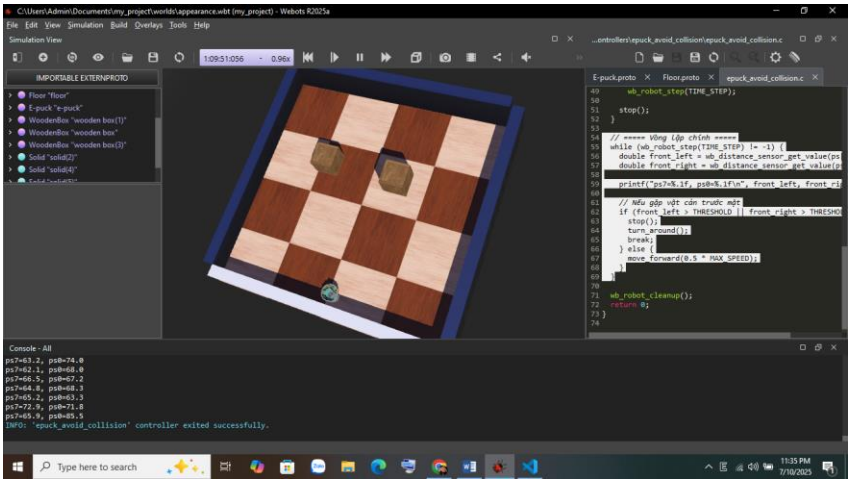
- `wb_robot_step(TIME_STEP):` cập nhật mô phỏng từng khung hình (64 ms).
- `front_left, front_right:` đọc giá trị cảm biến hai bên trước robot (ps7, ps0).
- Nếu một trong hai cảm biến vượt ngưỡng `THRESHOLD` → phát hiện người (vật cản trước mặt). Khi đó:
- `stop():` dừng robot.
- `turn_around():` quay chào người.
- `break;:` kết thúc vòng lặp → nhiệm vụ hoàn thành.
- Ngược lại (không có người): robot di chuyển thẳng với tốc độ trung bình `0.5 * MAX_SPEED.`

Video kết quả

➔ Video thấy kết quả chạy đúng chương trình theo yêu cầu bài toán.

2.2. Trường hợp 2

- Hai vật thể (hộp) tượng trưng cho hai "người" đứng gần nhau, robot e-puck đến vị trí đằng sau sau đó chỉnh quỹ đạo đi vòng ra phía sau tránh cắt ngang trước mặt sau đó dừng lại ở vị trí đích và thực hiện cử chỉ xã hội (quay mặt hoặc nháy đèn LED).



Môi trường giả lập hai vật thể như con người

a. Khởi tạo các tham số

```
#include <webots/robot.h>
#include <webots/distance_sensor.h>
#include <webots/motor.h>
#include <stdio.h>
#include <string.h>

#define TIME_STEP 64
#define THRESHOLD_FRONT 78.0
#define THRESHOLD_SIDE 78.0
#define FORWARD_SPEED 3.0
#define TURN_SPEED 2.0
#define MAX_SPEED 6.28
```

Giải thích:

- `from controller import Robot:` nhập thư viện điều khiển của Webots để truy cập robot e-puck.
- TIME_STEP = 64: chu kỳ mô phỏng (ms). Robot cập nhật dữ liệu mỗi 64 ms.
- `THRESHOLD_FRONT` và `THRESHOLD_SIDE:` ngưỡng cảm biến (giá trị lớn hơn ngưỡng → có vật cản gần).

- `FORWARD_SPEED, TURN_SPEED:` tốc độ di chuyển khi tiến hoặc quay.
- `robot = Robot():` tạo đối tượng robot để điều khiển.

b. Khởi tạo cảm biến khoảng cách

```
  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);
  }
```

Giải thích:

- Khởi tạo 8 cảm biến khoảng cách (ps0 → ps7).
- `getDevice(f"ps{i}"):` truy cập cảm biến theo tên.
- `sensor.enable(TIME_STEP):` bật cảm biến, cho phép đọc giá trị mỗi 64 ms.
- `ps.append(sensor):` lưu tất cả cảm biến vào danh sách ps.

c. Khởi tạo động cơ

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

Giải thích:

- `getDevice("left/right wheel motor"):` truy cập 2 động cơ bánh xe.
- `setPosition(float('inf')):` cho phép bánh xe quay tự do (chế độ vận tốc).

d. Các hàm di chuyển tiến, lùi, dừng

```
void move_forward(double speed) {
  wb_motor_set_velocity(left_motor, speed);
  wb_motor_set_velocity(right_motor, speed);
}

void move_backward(double speed) {
  wb_motor_set_velocity(left_motor, -speed);
  wb_motor_set_velocity(right_motor, -speed);
}

void turn_left(double speed) {
```

```
  wb_motor_set_velocity(left_motor, -speed);
  wb_motor_set_velocity(right_motor, speed);
}

void turn_right(double speed) {
  wb_motor_set_velocity(left_motor, speed);
  wb_motor_set_velocity(right_motor, -speed);
}

void stop() {
  wb_motor_set_velocity(left_motor, 0);
  wb_motor_set_velocity(right_motor, 0);
}

void turn_around() {
  // Robot quay 180 độ để chào
  wb_motor_set_velocity(left_motor, TURN_SPEED);
  wb_motor_set_velocity(right_motor, -TURN_SPEED);

  int duration = 1500 / TIME_STEP;
  for (int i = 0; i < duration; i++)
    wb_robot_step(TIME_STEP);

  stop();
}
```

Giải thích:

- `move_forward()` → cả hai bánh quay cùng chiều, robot tiến thẳng.
- `move_backward()` → quay ngược, robot lùi.
- `turn_left(), turn_right()` → hai bánh quay ngược chiều để robot quay tại chỗ.
- `stop()` → dừng cả hai bánh.
- `turn_around()` → quay 180° rồi dừng, được dùng để robot "chào" sau khi tránh vật cản.

e. Vòng lặp chính – Điều khiển

```
  while (wb_robot_step(TIME_STEP) != -1) {
    double front_left = wb_distance_sensor_get_value(ps[7]);
    double front_right = wb_distance_sensor_get_value(ps[1]);
    double side_left = wb_distance_sensor_get_value(ps[6]);
    double side_right = wb_distance_sensor_get_value(ps[2]);

    printf("[KC hai bên] Trái: %.1f | Phải: %.1f\n", side_left, side_right);
```

Giải thích:

- Vòng lặp chạy liên tục cho đến khi mô phỏng kết thúc.
- Robot đọc giá trị từ các cảm biến để xác định khoảng cách.
- Dữ liệu cảm biến được in ra để theo dõi.

f. Giai đoạn 1: Đi thẳng và phát hiện người

```
if (strcmp(stage, "move_forward") == 0) {
  move_forward(FORWARD_SPEED);

  if (side_left > THRESHOLD_SIDE || side_right > THRESHOLD_SIDE) {
    stop();
    printf("⚠ Phát hiện người gần hơn ở một bên!\n");

    move_backward(FORWARD_SPEED);
    for (int i = 0; i < 1000 / TIME_STEP; i++)
      wb_robot_step(TIME_STEP);
    stop();

    if (side_left > side_right) {
      printf("➡ Né sang phải\n");
      strcpy(stage, "avoid_right");
    } else {
      printf("⬅ Né sang trái\n");
      strcpy(stage, "avoid_left");
    }
  }
}
```

Giải thích:

- Robot di chuyển thẳng.
- Khi cảm biến bên trái hoặc phải phát hiện người gần, robot dừng lại → lùi ra một đoạn.
- Sau đó xác định bên nào gần hơn → chọn hướng tránh (trái hoặc phải).

g. Giai đoạn 2: Né sang phải hoặc trái

```
else if (strcmp(stage, "avoid_right") == 0) {
  turn_right(TURN_SPEED);
  for (int i = 0; i < 1200 / TIME_STEP; i++)
    wb_robot_step(TIME_STEP);

  move_forward(FORWARD_SPEED);
  for (int i = 0; i < 4000 / TIME_STEP; i++)
    wb_robot_step(TIME_STEP);

  turn_left(TURN_SPEED);
```

```
    for (int i = 0; i < 1200 / TIME_STEP; i++)
      wb_robot_step(TIME_STEP);

    move_forward(FORWARD_SPEED);
    for (int i = 0; i < 4000 / TIME_STEP; i++)
      wb_robot_step(TIME_STEP);

    stop();
    strcpy(stage, "go_behind");
  }

  // ===== Né sang trái =====
  else if (strcmp(stage, "avoid_left") == 0) {
    turn_left(TURN_SPEED);
    for (int i = 0; i < 1200 / TIME_STEP; i++)
      wb_robot_step(TIME_STEP);

    move_forward(FORWARD_SPEED);
    for (int i = 0; i < 4000 / TIME_STEP; i++)
      wb_robot_step(TIME_STEP);

    turn_right(TURN_SPEED);
    for (int i = 0; i < 1200 / TIME_STEP; i++)
      wb_robot_step(TIME_STEP);

    move_forward(FORWARD_SPEED);
    for (int i = 0; i < 4000 / TIME_STEP; i++)
      wb_robot_step(TIME_STEP);

    stop();
    strcpy(stage, "go_behind");
  }
```

Giải thích:

- Robot quay phải, đi vòng qua vật thể, sau đó quay lại trái để trở về hướng cũ.
- Kết thúc đường tránh, chuyển sang giai đoạn "go_behind".

h. Đến phía sau và chào

```
  else if (strcmp(stage, "go_behind") == 0) {
    if (front_left < THRESHOLD_FRONT && front_right < THRESHOLD_FRONT) {
      turn_around();
      printf("✅ Đã đến phía sau và chào!\n");
      break;
    }
  }
}
```

Giải thích:

- Khi robot không còn vật cản phía trước → xác định đã đi ra phía sau người.
- Robot quay 180° (thực hiện "chào") rồi kết thúc chương trình.

Video kết quả

➔ Video ta thấy robot thực hiện đúng hành vi như yêu cầu, bài thực hành đạt được social behavior cho robot với vật cản (thùng gỗ) là "người"