TÀI LIỆU HỌC TẬP

MINDSTORM LEGO EV3 ROBOT KIT PROGRAMMING

MỤC LỤC

Tài liệu học tập

1. Bộ công cụ phát triển

LEGO® challenge: Powerful LEGO

Hình 1: LEGO® challenge: Powerful LEGO

Bộ công cụ này cung cấp

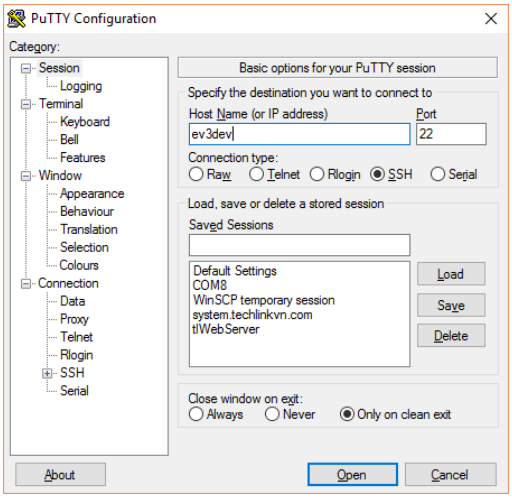
- các cảm biến cơ bản để robot thu nhận thông tin từ thế giới bên ngoài,
- bộ xử lý trung tâm với hệ điều hành Linux để xử lý và thực hiện các thuật toán thông minh do lập trình viên tự phát triển
- các cơ cấu chấp hành để robot tác động trở lại vào thế giới bên ngoài.

Tham khảo: https://www.lego.com/en-us/mindstorms/products/mindstorms-ev3-31313

2. Phần mềm Putty

Phần mềm hỗ trợ giao thức SSH cho phép kết nối máy tính với robot, để có thể đưa các đoạn mã lập trình C vào trong robot.

Download tại : http://www.putty.org/

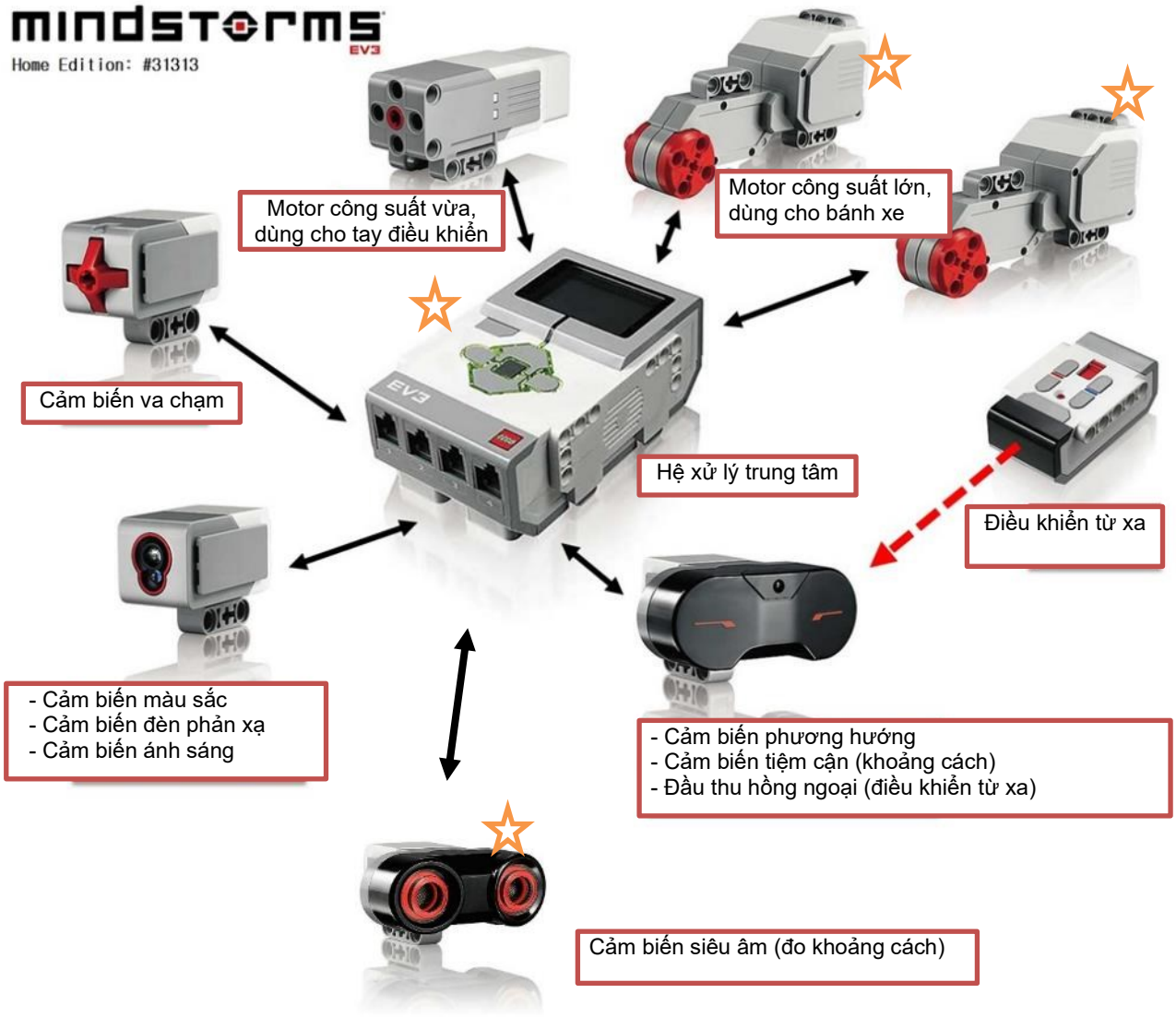


Hình 2: Phần mềm Putty

Hướng dẫn lập trình với các linh kiện trong Mindstorm EV3

Mindstorm EV3 là một bộ kit lắp ghép do hãng Lego sản xuất với đầy đủ các linh kiện cơ khí và điện tử, để lắp ráp thành nhiều loại robot khác nhau. Các ro bot này đều có thể lập trình để điều khiển robot làm việc như ý muốn.

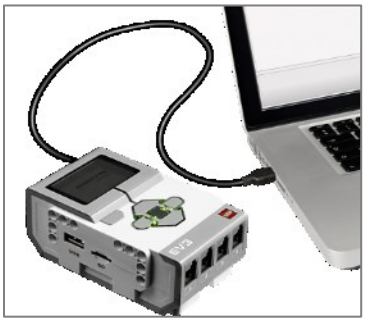
Để có thể lập trình được, Mindstorm EV3 cung cấp một nền tảng phần cứng mạnh mẽ như sau:



Hình 3: Hệ xử lý, cảm biến, cơ cấu chấp hành trong EV3

Lưu ý: các bài lab thực hành chỉ sử dụng linh kiện có dấu sao

1. P-Brick: Hệ xử lý trung tâm



Mindstorm EV3 bao gồm hệ bộ xử lý trung tâm, gọi là P-Brick với cấu hình:

- Bộ xử lý ARM 9 (tương tự như bộ xử lý trong các điện thoại di động)
- Bộ nhớ RAM 64 Megabyte
- Bộ nhớ chương trình Flash 16 Megabyte
- Hệ điều hành Linux
- Có 4 ngõ vào IN1,IN2, IN3, IN4 để nối với các cảm biến
- Có 4 ngõ ra OUTA, OUTB, OUTC, OUTD để nối với các cơ cấu chấp hành như động cơ, role…

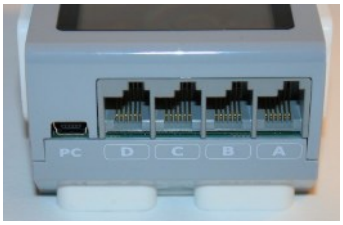
2. Cảm biến siêu âm

- Đo khoảng cách: 1 - 250 cm
- Độ chính xác: +/- 1 cm
- Hàm lấy giá trị từ cảm biến siêu âm

```
int sensor_get_value( uint8_t index,
     uint8_t input_port,
     int default_value
   );
```

Trong đó

- index: là số nguyên, chỉ số thuộc tính. Luôn bằng 0
- input_port: là số nguyên, là số hiệu cổng trên P-Brick dùng để nối với cảm biến.
  SOCKET__NONE_ = 0,
  IN1 = 0x1L,
  IN2 = 0x2L,
  IN3 = 0x4L,
  IN4 = 0x8L,
  OUTA = 0x10L,
  OUTB = 0x20L,
  OUTC = 0x40L,
  OUTD = 0x80L



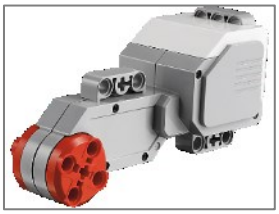
- default_value: là giá trị mặc định trả về
- Giá trị return của hàm là khoảng cách theo cm (mặc định).

- Ví dụ:

```
// Lấy giá trị của cảm biến siêu âm ở cổng kết nối IN4
val = sensor_get_value( 0, IN4, 0 );
```

3. Motor công suất lớn

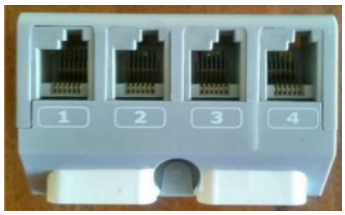
- Phản hồi với độ chính xác tới 1°
- Tốc độ tối đa: 160 - 170 rpm
- Mô men chạy 20 N.cm
- Mô men xoắn 40 N.cm
- Hàm thiết lập tốc độ cho động cơ

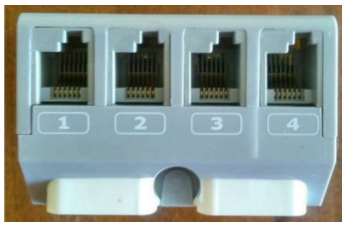


```
tacho_set_speed_sp(uint8_t output_port, int power);
```

Trong đó

- output_port: là số nguyên, là số hiệu cổng trên P-Brick nối với mortor
  SOCKET__NONE_ = 0,
  IN1 = 0x1L,
  IN2 = 0x2L,
  IN3 = 0x4L,
  IN4 = 0x8L,
  OUTA = 0x10L,
  OUTB = 0x20L,
  OUTC = 0x40L,
  OUTD = 0x80L



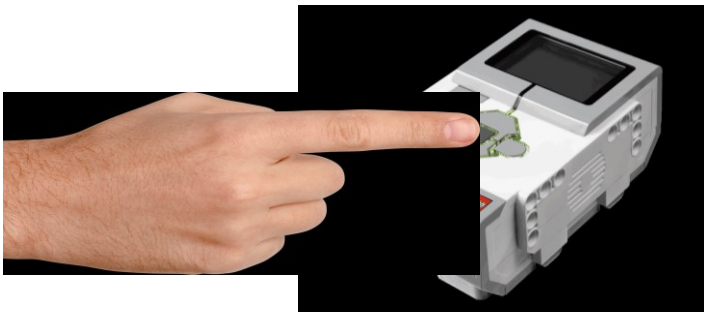


- power: công suất hoạt động.
  - power = 0 thì động cơ ngừng hoạt động.
  - power > 0 thì động cơ chuyển động tiến về phía trước.
  - power < 0 thì động cơ chuyển động lùi về phía sau.
- Ví dụ:

```
tacho_set_speed_sp(OUTB, tacho_get_max_speed( OUTB, 0 ) * 0.75 );  // 75%
tacho_run_forever(OUTB);
```

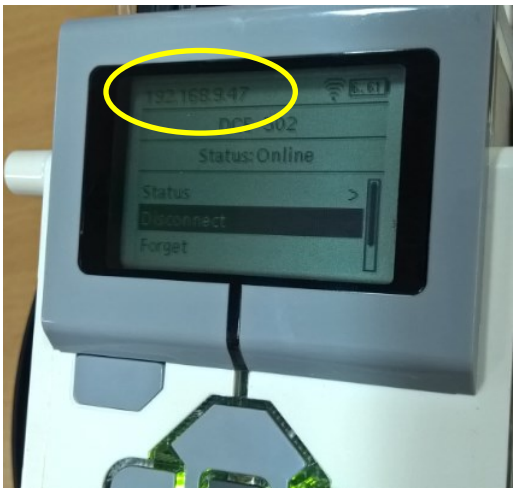
Kết nối Mindstorm EV3 với máy tính

- Khởi động P-Brick

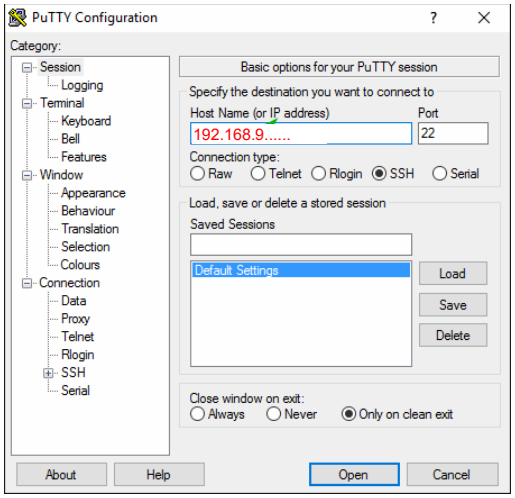


Truy cập vào robot bằng Putty

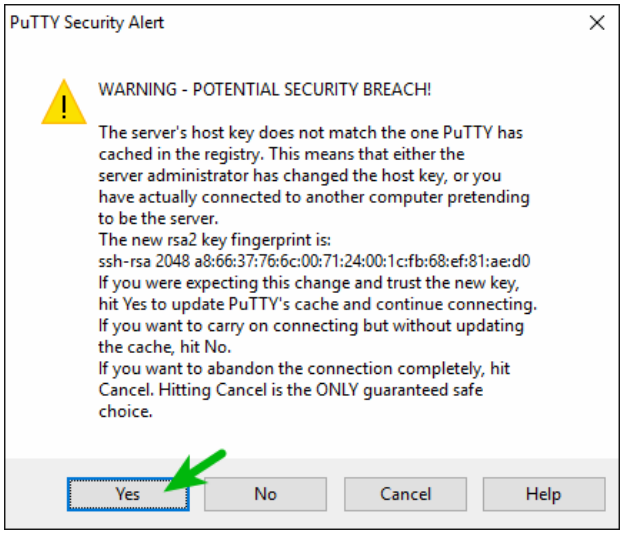
- Trên mặt màn hình của robot, xem địa chỉ IP của robot



- Trên máy tính, chạy phần mềm Putty
  Trong cửa sổ cấu hình của PuTTy, điền **địa chỉ IP** của robot vào ô "**Host Name (or IP address)**".
  Sau đó nhấn vào nút "**Open**" để kết nối.



- Nếu là lần kết nối đầu tiên, sẽ có một cảnh báo như hình bên dưới. Chỉ cần chọn “Yes” để tiếp tục. Chúng ta sẽ không nhìn thấy thông báo này vào những lần kết nối tiếp theo.



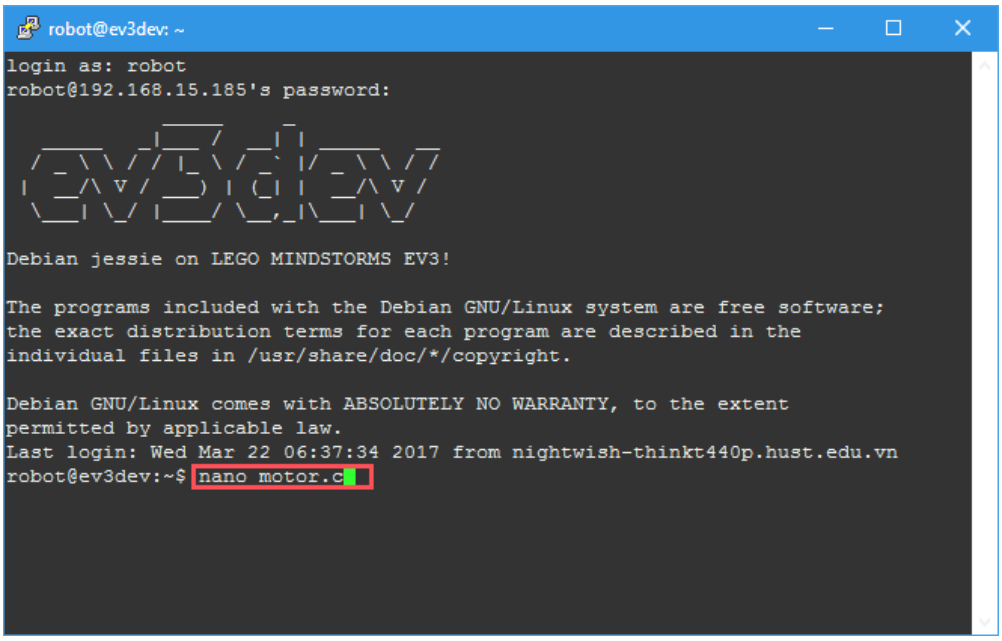
- Khi đã kết nối thành công, hệ điều hành sẽ yêu cầu chúng ta nhập username (**robot**) và password (**maker**) để đăng nhập. Sau khi đăng nhập thì cửa sổ giao diện như sau:

Sửa và dịch file chương trình điều khiển trong robot

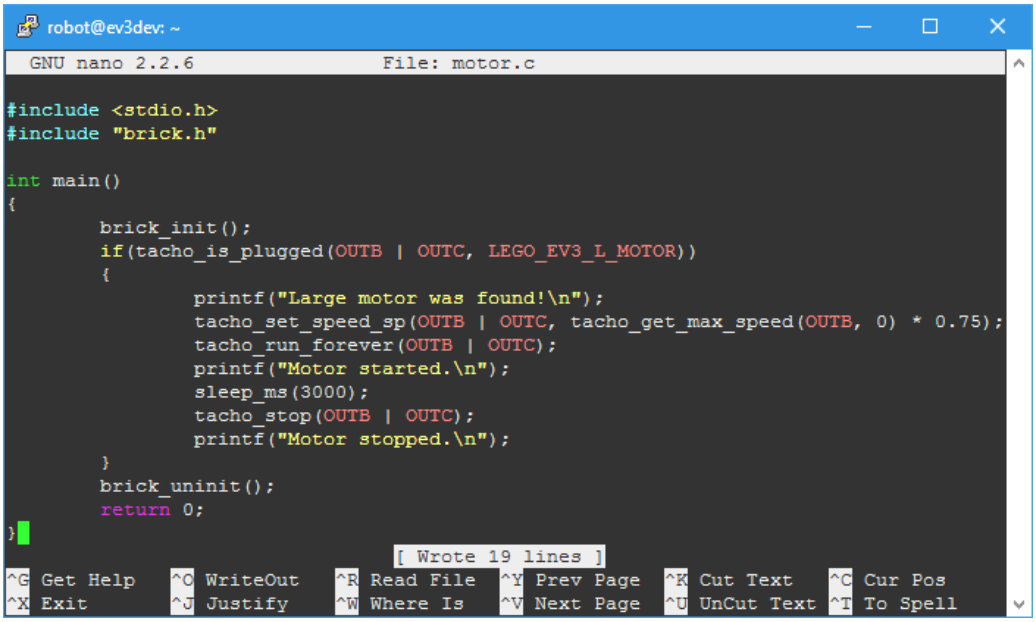
Sử dụng công cụ soạn thảo **nano** để tạo và viết mã nguồn chương trình.

Cú pháp: **nano <tên file>**

Ví dụ: **nano motor.c**



Sau khi trình soạn thảo **nano** được mở, chúng ta có thể viết mã chương trình cho EV3.

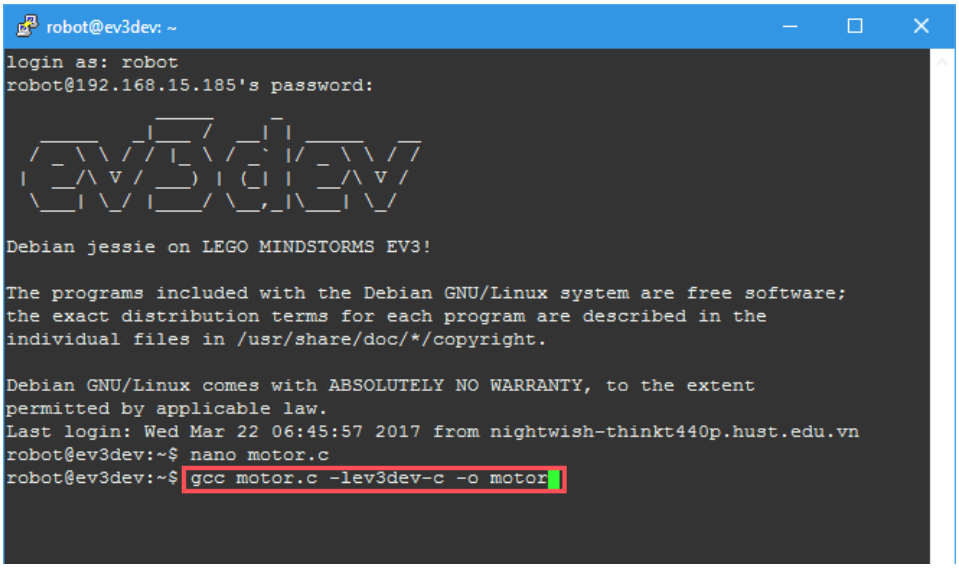


Khi đã soạn thảo xong hãy nhấn tổ hợp phím **Ctrl+O** để lưu file, nhấn tổ hợp **Ctrl+X** để thoát khỏi trình soạn thảo **nano**.

Sử dụng cú pháp sau để tiến hành biên dịch file mã nguồn thành file thực thi:

gcc <tên file mã nguồn>.c –lev3dev-c –o <tên file thực thi>

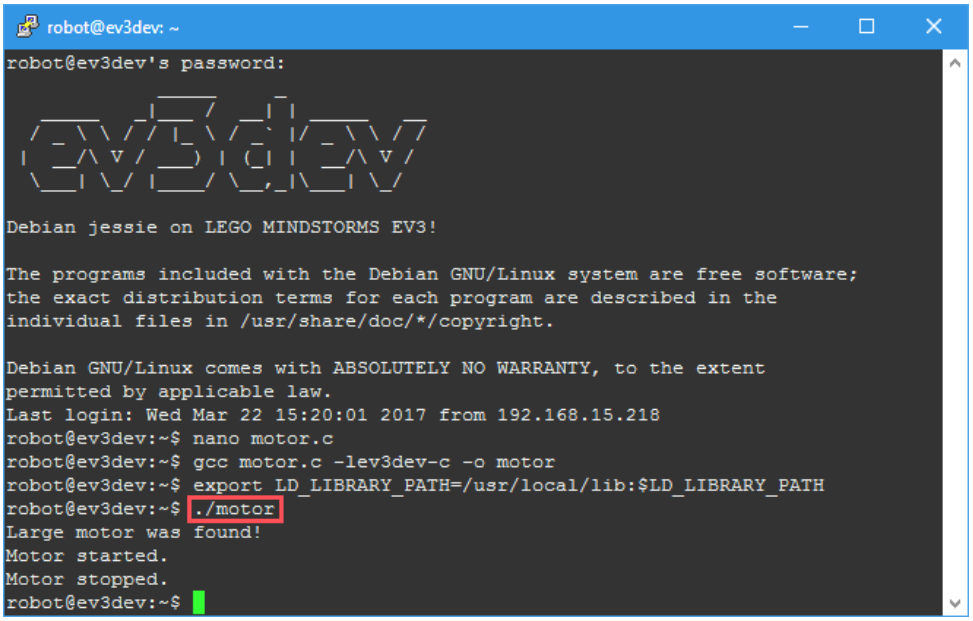
Ví dụ: **gcc motor.c –lev3dev-c –o motor**



Trong suốt qúa trình biên dịch nếu không có thông báo lỗi thì tức là mã nguồn đã được biên dịch thành công và file thực thi "**motor**" sẽ được tạo ra. Cuối cùng, để chạy chương trình ta thực hiện cú pháp như sau:

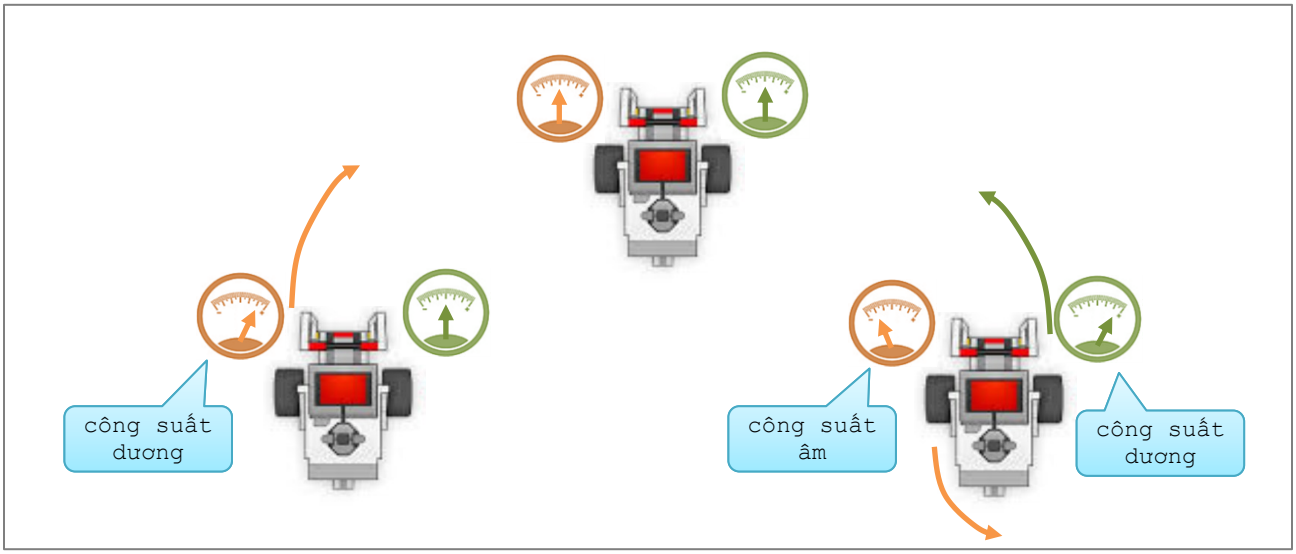
./<tên file thực thi>

Ví dụ: **./motor**



Bài thực hành 1: điều khiển hướng chuyển động

Nguyên lý



Công suất âm là công suất động cơ đã đảo chiều quay

Yêu cầu cần thực hiện

- Đọc, hiểu, và thự thi đoạn chương trình LAB 01 trên robot EV3 với ý nghĩa:

  Robot chuyển động thẳng về phía trước bằng cách:

  - **Hai động cơ trái, phải chạy cùng công suất**
  - **Công suất 2 động cơ là số nguyên dương**
- Nhóm sinh viên cần chỉnh sửa mã nguồn theo các yêu cầu trong Checklist kèm tài liệu

Chương trình điều khiển

Lưu ý: Chương trình điều khiển đã có sẵn trên robot với tên file lab01.c. Sinh viên có thể chạy thử ngay và hiệu chỉnh lệnh.

```
//-------------------------------------------------------------------------------
// Bo mon Ky thuat May tinh, Vien CNTT-TT, Dai hoc Bach Khoa Ha Noi
//                                LAB 01
//-------------------------------------------------------------------------------
#include <stdio.h>
#include "brick.h"
int main()
{
      printf( "Bai lab01.\n" );
      printf( "EV3 dang khoi dong...\n" );
      brick_init();

      // Kiem tra ket noi cua 2 dong co tai cong B va C
      if(tacho_is_plugged(OUTB | OUTC, LEGO_EV3_L_MOTOR))
      {
```

```
            // Thiet lap che do dung cho dong co
            tacho_set_stop_action(OUTB | OUTC, TACHO_BRAKE);

            // Thiet lap toc do cho dong co
            tacho_set_duty_cycle_sp(OUTB | OUTC, 50);

            // Ra lenh cho dong co hoat dong
            tacho_run_direct(OUTB | OUTC);
            printf( "Dong co dang chay.\n" );

            printf( "Doi 3 giay...\n" );
            // Doi 3 (s)
            sleep_ms(3000);

            // Dung dong co
            tacho_stop(OUTB | OUTC);
            printf( "Dong co dung.\n" );
        }

        brick_uninit();
        printf( "Ket thuc...\n" );
        return 0;
}
```

Các bước thực hiện

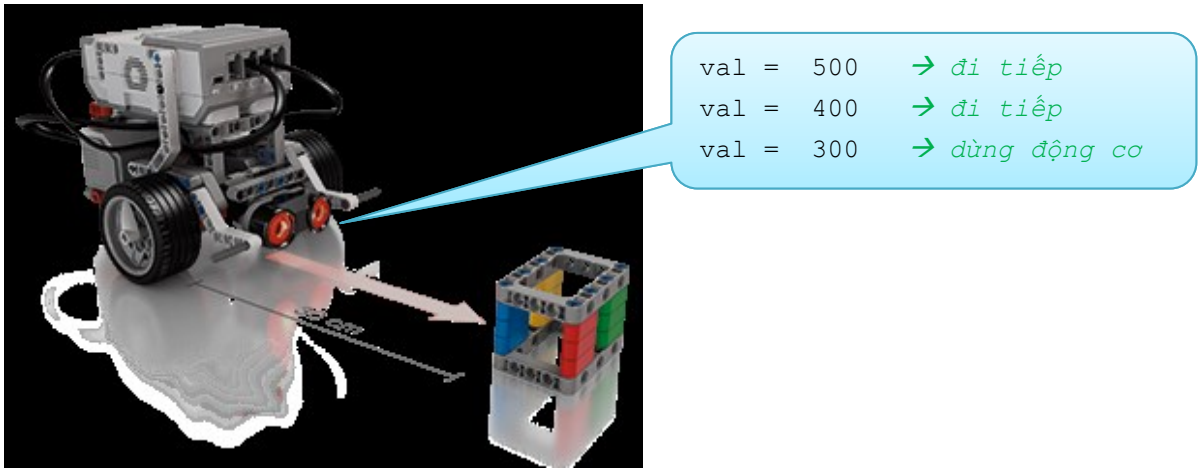
- Kết nối máy tính với P-Brick như đã hướng dẫn ở trên
- Sửa mã nguồn chương trình điều khiển ở file lab01.c
- Biên dịch lại mã nguồn điều khiển.

Cách kiểm tra

- Cho robot hoạt động bằng cách bấm vào nút nguồn.
- Đặt robot xuống sàn nhà để robot di chuyển
- Kiểm tra xem robot có chạy thẳng hay không?

Bài thực hành 2: robot di chuyển và phát hiện vật cản

Nguyên lý



Yêu cầu cần thực hiện

- Đọc, hiểu, và thự thi đoạn chương trình LAB 02 trên robot EV3 với ý nghĩa:
  Robot đi thẳng mãi mãi. Trong khi đi, nếu robot gặp vật cản ở một cự li định trước thì robot sẽ dừng lại
- Nhóm sinh viên cần chỉnh sửa mã nguồn theo các yêu cầu trong Checklist kèm tài liệu

Chương trình điều khiển

Lưu ý: Chương trình điều khiển đã có sẵn trên robot với tên file lab02.c. Sinh viên có thể chạy thử ngay và hiệu chỉnh lệnh.

```
//-------------------------------------------------------------------------------
// Bo mon Ky thuat May tinh, Vien CNTT-TT, Dai hoc Bach Khoa Ha Noi
//                                LAB 02
//-------------------------------------------------------------------------------
#include <stdio.h>
#include "brick.h"
int main()
{
	int val;

	printf( "Bai lab02.\n" );
	printf( "EV3 dang khoi dong...\n" );
	brick_init();

	// Thiet lap che do dung cho dong co
	tacho_set_stop_action(OUTB | OUTC, TACHO_BRAKE);
	// Thiet lap toc do cho dong co
	tacho_set_duty_cycle_sp(OUTB | OUTC, 50);
	// Ra lenh cho dong co hoat dong
	tacho_run_direct(OUTB | OUTC);
```

```
        while(1)
        {
                // Doi 100 (ms)
                sleep_ms(100);
                // Lay gia tri khoang cach toi vat can tu cam bien sieu am
                val = sensor_get_value (0, IN4, 0);
                // In gia tri ra man hinh
                printf("val = %d\n", val);

                if( val <= 300)      // Neu khoang cach nho hon hoac bang 30 (cm)
                {
                        // Dung dong co
                        tacho_stop(OUTB | OUTC);
                        break;
                }
        }

        brick_uninit();
        printf( "Ket thuc...\n" );
        return 0;
}
```

Các bước thực hiện

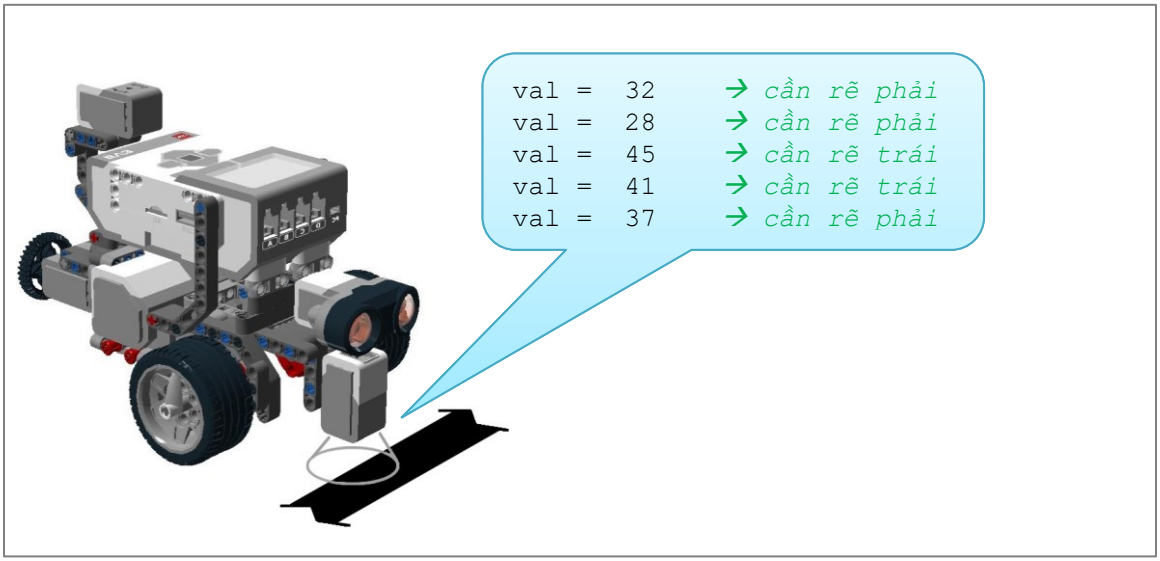
- Kết nối máy tính với P-Brick như đã hướng dẫn ở trên
- Sửa mã nguồn chương trình điều khiển ở file lab02.c
- Biên dịch lại mã nguồn điều khiển.

Cách kiểm tra

- Cho robot hoạt động bằng cách bấm vào nút nguồn.
- Đặt robot xuống sàn nhà để robot di chuyển
- Kiểm tra xem robot có chạy thẳng hay không?
- Đặt vật cản nào đó trước robot, xem robot có dừng lại hay đi lùi không?
- Khi có vật cản, khoảng cách mà robot phát hiện ra rồi dừng lại có đúng không?

Bài thực hành 3: robot dò đường

Nguyên lý



Yêu cầu cần thực hiện

- Đọc, hiểu, và thự thi đoạn chương trình LAB 02 trên robot EV3 với ý nghĩa:

  Đọc giá trị màu từ cảm biến giúp robot phân biệt vạch kẻ chỉ dẫn màu đen, trên nền đường màu trắng. Từ đó, robot điều chỉnh công suất động cơ để khi chạy, giá trị màu đó không thay đổi quá nhiều, tức là robot sẽ bám theo vạch kẻ chỉ dẫn.
- Nhóm sinh viên cần chỉnh sửa mã nguồn theo các yêu cầu trong Checklist kèm tài liệu

Chương trình điều khiển

```
//-------------------------------------------------------------------------------
// Bo mon Ky thuat May tinh, Vien CNTT-TT, Dai hoc Bach Khoa Ha Noi
//                                  LAB 02
//-------------------------------------------------------------------------------
#include <stdio.h>
#include "brick.h"
int main()
{
    int val;

    printf( "Bai lab03.\n" );
    printf( "EV3 dang khoi dong...\n" );
    brick_init();

    if(sensor_is_plugged(IN3, LEGO_EV3_COLOR))
    {
        printf("Found color sensor!\n");
        // Thiet lap che do do anh sang phan chieu cho cam bien mau
        // Trong che do nay gia tri tra ve cua cam bien tu 0 - 100
```

```
        color_set_mode_col_reflect(IN3);
    }

    // Thiet lap che do dung cho dong co
    tacho_set_stop_action(OUTB | OUTC, TACHO_BRAKE);
    // Thiet lap toc do cho dong co
    tacho_set_duty_cycle_sp(OUTB | OUTC, 50);
    // Ra lenh cho dong co hoat dong
    tacho_run_direct(OUTB | OUTC);

    while(1)
    {
        // Lay gia tri tu cam bien mau
        val = sensor_get_value(0, IN3, 0);
        // In gia tri ra man hinh
        printf("val = %d\n", val);

        // Neu gia tri
        if(val > 40)
        {
               // Giam toc do cua dong co ben trai (OUTB) de quay sang ben trai
               tacho_set_duty_cycle_sp(OUTB, 10);
               tacho_set_duty_cycle_sp(OUTC, 50);
        }
        else
        {
               // Giam toc do cua dong co ben phai (OUTC) de quay sang ben phai
               tacho_set_duty_cycle_sp(OUTC, 0);
               tacho_set_duty_cycle_sp(OUTB, 50);
        }
    }

    brick_uninit();
    printf( "Ket thuc...\n" );
    return 0;
}
```

Các bước thực hiện

- Kết nối máy tính với P-Brick như đã hướng dẫn ở trên
- Sửa mã nguồn chương trình điều khiển ở file lab03.c
- Biên dịch lại mã nguồn điều khiển.

Cách kiểm tra

- Đưa giấy màu đen/trắng vào gần cảm biến và theo dõi sự thay đổi của giá trị màu trên màn hình.
- Đặt robot lên các loại sa bàn khác nhau, xem khả năng bám đường của robot.
- Thay đổi các tham số để robot bám đường tốt hơn.