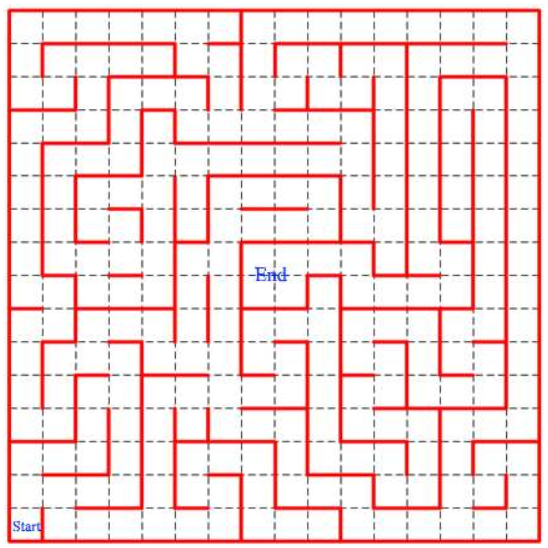
**PHƯƠNG ÁN THIẾT KẾ ROBOT CUỘC THI TMIRC 2024**

Tóm tắt luật chơi: Mê cung có kích thước 16x16 ô vuông. Robot xuất phát từ 1 trong 4 cạnh và tìm đường đến đích ở trung tâm mê cung. Mỗi robot có 5 lượt chơi và không thể can thiệp chương trình robot trong quá trình. Điểm tính theo trung bình thời gian hoàn thành 5 lượt.



Ví dụ minh họa mê cung

**Giải thuật và chiến thuật**

Dựa trên môi trường mê cung kích thước 16x16, so sánh các thuật toán phổ biến gồm DFS, BFS, Dijkstra và Flood Fill:

**DFS (Depth-First Search)**

Nguyên lý hoạt động: Duyệt sâu vào một nhánh của cây hoặc đồ thị trước khi quay lại để duyệt các nhánh khác.

Ưu điểm:

- Dễ triển khai và không yêu cầu nhiều bộ nhớ so với BFS và Dijkstra.

- Phù hợp cho việc tìm kiếm đường đi sâu nhất trong mê cung.

Nhược điểm:

- Không đảm bảo tìm được đường đi ngắn nhất.

- Có thể dẫn đến lặp vô hạn trong các mê cung có chu trình.

**BFS (Breadth-First Search)**

Nguyên lý hoạt động: Duyệt qua các đỉnh theo từng mức (level) một trước khi đi sâu vào mức tiếp theo.

Ưu điểm:

- Tìm được đường đi ngắn nhất trong mê cung với các ô có trọng số bằng nhau.

- Không bị kẹt trong chu trình của đồ thị.

Nhược điểm:

- Sử dụng nhiều bộ nhớ hơn so với DFS, đặc biệt là trong các mê cung lớn.

**Dijkstra**

Nguyên lý hoạt động: Tìm đường đi ngắn nhất từ một đỉnh xuất phát đến tất cả các đỉnh khác trong đồ thị có trọng số không âm.

Ưu điểm:

- Đảm bảo tìm được đường đi ngắn nhất.

- Hiệu quả với các đồ thị có trọng số không âm.

Nhược điểm:

- Độ phức tạp tính toán và yêu cầu bộ nhớ lớn hơn so với DFS và BFS.

- Không phù hợp với các mê cung có trọng số âm.

**Flood Fill**

Nguyên lý hoạt động: Lấp đầy một vùng liên thông trong hình ảnh bằng cách thay đổi màu sắc của các pixel.

Ưu điểm:

- Đơn giản và dễ triển khai.

- Hiệu quả cho việc biến đổi hình ảnh và xử lý các vùng liên tục trong mê cung.

Nhược điểm:

- Chỉ áp dụng cho việc lấp đầy vùng và không phải là giải pháp cho việc tìm đường đi trong mê cung.

**Tóm tắt so sánh**

- Độ phức tạp tính toán: Dijkstra có độ phức tạp lớn nhất với (O((V + E)log V)), trong khi DFS và BFS có độ phức tạp (O(V + E)) với (V) là số đỉnh và (E) là số cạnh trong mê cung.

- Độ phù hợp với bài toán: DFS và BFS thường được sử dụng khi chỉ cần tìm một đường đi từ điểm xuất phát đến đích mà không quan tâm đến đường đi ngắn nhất. Dijkstra phù hợp khi cần tìm đường đi ngắn nhất và các ô trong mê cung có trọng số không âm. Flood Fill thích hợp cho việc biến đổi hình ảnh và xử lý vùng liên tục trong mê cung.

Đối với yêu cầu về kích thước mê cung và luật của bài toán này, phương án phù hợp là giải thuật kết hợp flood fill và DFS, BFS. Chiến thuật thi đấu:

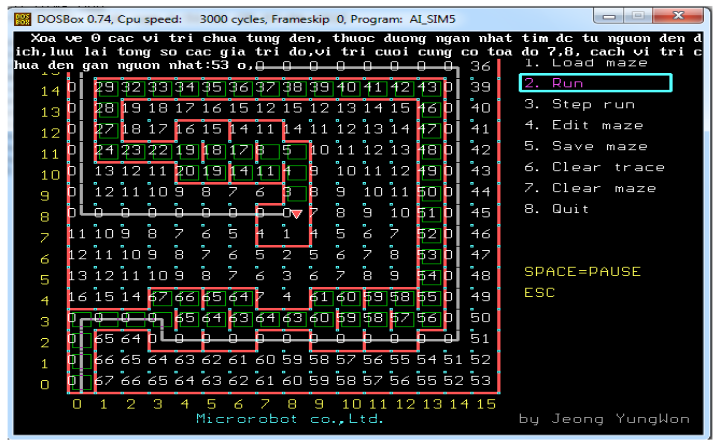
- Lượt thứ nhất: giải mê cung bằng giải thuật DFS và lưu đường đi

- Lượt thứ hai: giải mê cung bằng giải thuật BFS và lưu đường đi

- Lượt thứ ba: giải mê cung bằng giải thuật Flood Fill và lưu đường đi

- Lượt thứ tư: tối ưu đường đi từ lượt 3 bằng thông tin từ 3 lượt đi trước, giải mê cung và lưu đường đi

- Lượt thứ năm: Sử dụng đường đi lượt thứ 4 (nếu lượt 4 thành công) hoặc đường đi của lượt có thời gian ngắn nhất trước đó.



Thuật toán Flood Fill trên chương trình mô phỏng



Thiết kế mô hình robot V1

Thiết kế phần cứng V1:

- Động cơ DC giảm tốc, có phản hồi

- Bộ chống sóng hài

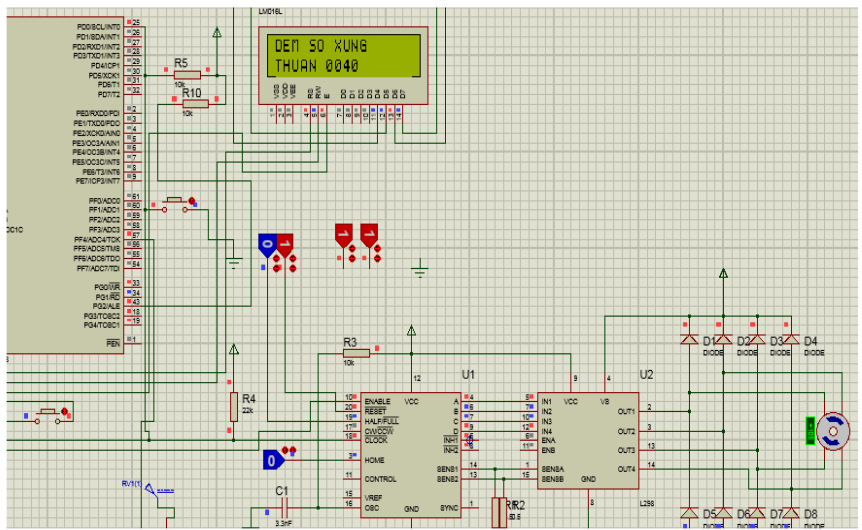
- Mạch công xuất điều khiển động cơ

- Bộ ổn áp

- Pin Lipo dạng mô đun tháo rời

- LCD và nút phục vụ thiết lập cấu hình

- Bộ nạp chương trình qua Wifi



Thiết kế mô phỏng mạch điều khiển

Kích thước robot khoảng 12x12cm, đường kính bánh 34mm, bộ điều khiển PID và chương trình điều khiển cho 5 lượt chạy. Có thể thay đổi cấu hình như thông số PID, tốc độ động cơ, độ nhạy cảm biến... trực tiếp trên LCD. Thông tin từ EEPROM có thể xuất ngược hoặc nạp chương trình mới qua wifi. Robot dò đường bằng cảm biến quang ở 4 mặt.



Cảm biến quang

Dung lượng chương trình dự kiến bao gồm bộ PID ~ 400B, chương trình điều khiển ~ 200B, thông tin từ các lượt chạy ~ 400B. Vì vậy để đảm bảo tốc độ xử lý và khả năng lưu trữ, thiết kế Robot V1 bao gồm 3 phương án cho vi điều khiển:

- Arduino Nano: dung lượng EEPROM 1024B, đảm bảo khả năng lưu trữ thông tin các lượt đi. Tốc độ xung 12 MHz

- Arduino Mega: EEPROM 4KB, tốc độ xung 16 MHz nhưng chỉ có cấu hình chip dán (khó gia công mạch)

- ESP 32: Tốc độ xử lý 160 MHz, dung lượng EEPROM 560B.

Phương án cuối cùng phụ thuộc vào kết quả thử nghiệm.

**ĐỀ CƯƠNG**

|  |  |  |  |
| --- | --- | --- | --- |
| **Công việc** | **Kết quả** | **Người thực hiện** | **Thời gian** |
| Hoàn thành phương án thiết kế  Xây dựng chiến thuật và các phương án |  |  | Trước tháng 7 |
| Thi công sân đấu  Thiết kế mô hình robot  Mô phỏng các giải thuật |  |  | 10/7 |
| Mô hình robot V1  Hoàn thành chương trình điều khiển  Thử nghiệm V1 |  |  | 18/7 |
| Cải tiến chương trình hoặc mô hình V1 dựa trên kết quả thử nghiệm |  |  | 27/7 |
| Hoàn thiện mô hình  Bổ sung các phương án dự phòng |  |  | 7/7 |