

# **BÁO CÁO ĐỒ ÁN MÔN HỌC CẢM BIẾN VÀ CƠ CẤU CHẤP HÀNH**

## **ROBOT MICROMOUSE Giải Thuật Di Chuyển Trong Mê Cung**

Ngành: Robot và trí tuệ nhân tạo

Lớp: 22DRTA1

**GIẢNG VIÊN : TS. PHẠM QUỐC THIỆN**

<b>Sinh viên thực hiện:</b>	<b>MSSV:</b>	<b>Lớp:</b>
Nguyễn Văn Đạt	2286300010	22DRTA1
Huỳnh Long	2286300028	22DRTA1
Nguyễn Chấn Huy	2286300020	22DRTA1

*TP. Hồ Chí Minh, ngày 10 tháng 12 năm 2024*

# **BÁO CÁO ĐỒ ÁN MÔN HỌC**

## **CẢM BIẾN VÀ CƠ CẤU CHẤP HÀNH**

### **ROBOT MICROMOUSE**

#### **Giải Thuật Di Chuyển Trong Mê Cung**

Ngành: Robot và trí tuệ nhân tạo

Lớp: 22DRTA1

**GIẢNG VIÊN : TS. PHẠM QUỐC THIỆN**

<b>Sinh viên thực hiện:</b>	<b>MSSV:</b>	<b>Lớp:</b>
Nguyễn Văn Đạt	2286300010	22DRTA1
Huỳnh Long	2286300028	22DRTA1
Nguyễn Chấn Huy	2286300020	22DRTA1

*TP. Hồ Chí Minh, ngày 10 tháng 12 năm 2024*

**VIỆN KỸ THUẬT HUTECH****PHIẾU GIAO ĐỀ TÀI****TÊN MÔN HỌC: ĐỒ ÁN CẢM BIẾN VÀ CƠ CẤU CHẤP HÀNH****NGÀNH: ROBOT VÀ TRÍ TUỆ NHÂN TẠO****1. Họ và tên sinh viên/ nhóm sinh viên được giao đề tài (sĩ số trong nhóm: 3):**

(1) Nguyễn Văn Đạt..... MSSV: 2286300010..... Lớp: 22DRTA1

(2) Huỳnh Long..... MSSV: 2286300028..... Lớp: 22DRTA1

(3) Nguyễn Chấn Huy..... MSSV: 2286300020 ..... Lớp: 22DRTA1

**2. Tên đề tài: Robot Micromouse - Giải Thuật Di Chuyển Trong Mê Cung****3. Các dữ liệu ban đầu:****a. Thông tin về môi trường hoạt động – mê cung:**

- Mê cung được cấu tạo bởi MxN ô vuông (thông thường sẽ là 16x16 đến 32x32 ô vuông), kích thước mỗi ô vuông là 20(cm)x20(cm)

- Cấu trúc mê cung: Bao gồm các bức tường, ngõ cụt, và đường dẫn. Các quy tắc về cấu trúc mê cung có thể thay đổi tùy theo cuộc thi hoặc quy chuẩn phát triển.

- Độ phức tạp của mê cung: Bao gồm số lượng ngõ cụt và cách bố trí các bức tường.

**b. Thông tin về robot Micromouse:**

- Kích thước robot tối đa 15(cm)x15(cm)

- Cấu hình phần cứng:

- + Vi điều khiển (Arduino, STM32, ESP32...)

- + Cảm biến: Cảm biến hồng ngoại (IR), cảm biến siêu âm... để xác định khoảng cách với tường.

- + Cơ cấu chấp hành: Động cơ bước, DC...

- + Pin và hệ thống nguồn đảm bảo cung cấp đủ năng lượng cho toàn bộ hệ thống trong thời gian di chuyển qua mê cung.

- c. Thuật toán tìm đường: Các thuật toán phổ biến như Wall Follower, Flood Fill và A\*.

**4. Nội dung nhiệm vụ:****a. Nhiệm vụ về thiết kế phần cứng**

- Thiết kế khung robot Micromouse: Đảm bảo tính cơ động và nhẹ.

- Lựa chọn cảm biến: Đảm bảo độ chính xác và khả năng phát hiện chính xác khoảng cách đến các bức tường.

- Thiết kế hệ thống điều khiển: Việc lập trình vi điều khiển để quản lý các dữ liệu cảm biến, quyết định hành động di chuyển và xử lý phản hồi.

b. Nhiệm vụ về phát triển phần mềm

- Xây dựng thuật toán tìm đường: Cần phát triển thuật toán giúp robot khám phá mê cung và tìm đường về đích.

tối ưu.

- Điều khiển động cơ: Viết các chương trình điều khiển động cơ để điều khiển tốc độ và hướng di chuyển.

c. Tối ưu hóa hiệu suất

- Cải thiện thời gian tìm đường bằng việc ưu tiên sử dụng các thuật toán có thể ghi nhớ và tìm được đường đi tối ưu như Flood Fill hoặc A\*.

- Tối ưu hóa tốc độ di chuyển để robot (khi đi thẳng và quẹo) mà vẫn đảm bảo sự ổn định.

d. Thử nghiệm và hiệu chỉnh

- Thử nghiệm robot trong các môi trường mê cung khác nhau: Đảm bảo tính linh hoạt và khả năng thích ứng của robot với các mê cung có độ phức tạp khác nhau.

- Hiệu chỉnh cảm biến và động cơ: Điều chỉnh các thông số liên quan đến cảm biến và động cơ để cải thiện hiệu suất.

e. Báo cáo kết quả và đánh giá hiệu suất

- Thu thập các thông số quan trọng: Số lần va chạm, thời gian hoàn thành mê cung, độ chính xác của thuật toán điều hướng.

- Đánh giá độ ổn định của hệ thống: Xác định mức độ tin cậy của robot trong việc tìm đường và di chuyển trong mê cung.

5. **Kết quả tối thiểu phải có:**

- Hoàn thiện về thiết kế phần cứng
- Phát triển và triển khai thuật toán tìm đường
- Thử nghiệm và kết quả đánh giá
- Báo cáo và phân tích kết quả

Ngày giao đề tài: ...../...../..... Ngày nộp báo cáo: ...../...../.....

TP. HCM, ngày ... tháng ... năm .....

**Sinh viên thực hiện**

**Giảng viên hướng dẫn**

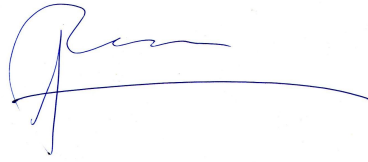
(Ký và ghi rõ họ tên các thành viên)

(Ký và ghi rõ họ tên)

**Nguyễn Văn Đạt**

**Nguyễn Chấn Huy**

**Huỳnh Long**



Phạm Quốc Thiện



Viện Kỹ thuật HUTECH

**PHIẾU CHẤM ĐIỂM**  
**HỌC PHẦN XỬ LÝ ẢNH SỐ**

6. **Họ và tên sinh viên:** Nguyễn Văn Đạt.....  
Lớp : 22DRTA1..... MSSV: 2286300010.....  
Ngành : Robot và Trí tuệ nhân tạo.....  
Chuyên ngành : Robot và Trí tuệ nhân tạo.....
7. **Họ và tên giảng viên hướng dẫn (GVHD):** Phạm Quốc Thiện
8. **Họ và tên giảng viên phản biện (GVPB):** Phạm Quốc Phương
9. **Đánh giá kết quả theo thang điểm 10 (Ghi rõ điểm số và điểm chữ) .....**  
.....

CĐR	Tiêu chí đánh giá (trọng số)	Mức chất lượng ( $\geq 4$ : đạt)					Điểm	
		Mức F (0-3.9)	Mức D (4.0-5.4)	Mức C (5.5-6.9)	Mức B (7.0-8.4)	Mức A (8.5-10)	GVHD (50%)	GVPB (50%)
CLO1, CLO6	Hình thức – nội dung báo cáo Trả lời câu hỏi và trao đổi trong phần thảo luận. (15%)	Đáp ứng dưới 30% yêu cầu  Không trả lời đúng câu hỏi nào	Đáp ứng 30% - dưới 50% yêu cầu  Trả lời đúng  dưới 1/2 số câu hỏi	Đáp ứng 50% - dưới 70% yêu cầu  Trả lời đúng  1/2 số câu hỏi	Đáp ứng 70% - dưới 80% yêu cầu  Trả lời đúng  trên 2/3 số câu hỏi	Đáp ứng 80% - 100% yêu cầu  Trả lời đúng  tất cả các câu hỏi		
CLO2	Nội dung chính 1: Thực hiện được các phép biến đổi ảnh cơ bản như chuyển đổi ảnh RGB thành ảnh xám, tính	Đáp ứng dưới 30% yêu cầu	Đáp ứng 30% - dưới 50% yêu cầu	Đáp ứng 50% - dưới 70% yêu cầu	Đáp ứng 70% - dưới 80% yêu cầu	Đáp ứng 80% - 100% yêu cầu		

	toán và phân tích biểu đồ Histogram (15%)							
CLO3, CLO4	Nội dung chính 2: Áp dụng được các toán tử điểm, xử lý Histogram và ngưỡng hóa ảnh (threshold) để cải thiện chất lượng hình ảnh Áp dụng được các bộ lọc khôi phục ảnh như bộ lọc nghịch đảo và bộ lọc Wiener để cải thiện ảnh bị nhiễu (40%)	Đáp ứng dưới 30% yêu cầu	Đáp ứng 30% - dưới 50% yêu cầu	Đáp ứng 50% - dưới 70% yêu cầu	Đáp ứng 70% - dưới 80% yêu cầu	Đáp ứng 80% - 100% yêu cầu		
CLO5, CLO6	Nội dung chính 3: Sử dụng được các thuật toán phát hiện cạnh như Sobel, Prewitt, Laplacian of Gaussian, và Canny để phát hiện cạnh của hình ảnh. Lập trình được các giải thuật xử lý ảnh cơ bản và nâng cao bằng Python để giải quyết các bài toán thực tế (30%)	Đáp ứng dưới 30% yêu cầu	Đáp ứng 30% - dưới 50% yêu cầu	Đáp ứng 50% - dưới 70% yêu cầu	Đáp ứng 70% - dưới 80% yêu cầu	Đáp ứng 80% - 100% yêu cầu		
<b>TỔNG CỘNG:</b>								
<b>ĐIỂM TRUNG BÌNH</b>								

TP. HCM, ngày ... tháng ... năm .....

**Giảng viên hướng dẫn**

(Ký và ghi rõ họ tên)

TP. HCM, ngày ... tháng ... năm .....

**Giảng viên phản biện**

(Ký và ghi rõ họ tên)



## Viện Kỹ thuật HUTECH

## PHIẾU CHẤM ĐIỂM

### HỌC PHẦN XỬ LÝ ẢNH SỐ

10. Họ và tên sinh viên: Huỳnh Long.....

Lớp : 22DRTA1..... MSSV: 2286300028.....

Ngành : Robot và Trí tuệ nhân tạo.....

Chuyên ngành : Robot và Trí tuệ nhân tạo.....

11.. Họ và tên giảng viên hướng dẫn (GVHD): Phạm Quốc Thiện

12.. Họ và tên giảng viên phản biện (GVPB): Phạm Quốc Phương

13. Đánh giá kết quả theo thang điểm 10 (Ghi rõ điểm số và điểm chữ) .....

.....

CĐR	Tiêu chí đánh giá (trọng số)	Mức chất lượng ( $\geq 4$ : đạt)					Điểm	
		Mức F (0-3.9)	Mức D (4.0-5.4)	Mức C (5.5-6.9)	Mức B (7.0-8.4)	Mức A (8.5-10)	GVHD (50%)	GVPB (50%)
CLO1, CLO6	Hình thức – nội dung báo cáo Trả lời câu hỏi và trao đổi trong phần thảo luận. (15%)	Đáp ứng dưới 30% yêu cầu  Không trả lời đúng câu hỏi nào	Đáp ứng 30% - dưới 50% yêu cầu  Trả lời đúng  dưới 1/2 số câu hỏi	Đáp ứng 50% - dưới 70% yêu cầu  Trả lời đúng  1/2 số câu hỏi	Đáp ứng 70% - dưới 80% yêu cầu  Trả lời đúng  trên 2/3 số câu hỏi	Đáp ứng 80% - 100% yêu cầu  Trả lời đúng  tất cả các câu hỏi		
CLO2	Nội dung chính 1: Thực hiện được các phép biến đổi ảnh cơ bản như chuyển đổi ảnh RGB thành ảnh xám, tính toán và phân tích biểu đồ Histogram (15%)	Đáp ứng dưới 30% yêu cầu	Đáp ứng 30% - dưới 50% yêu cầu	Đáp ứng 50% - dưới 70% yêu cầu	Đáp ứng 70% - dưới 80% yêu cầu	Đáp ứng 80% - 100% yêu cầu		

CLO3, CLO4	Nội dung chính 2: Áp dụng được các toán tử điểm, xử lý Histogram và ngưỡng hóa ảnh (threshold) để cải thiện chất lượng hình ảnh Áp dụng được các bộ lọc khôi phục ảnh như bộ lọc nghịch đảo và bộ lọc Wiener để cải thiện ảnh bị nhiễu (40%)	Đáp ứng dưới 30% yêu cầu	Đáp ứng 30% - dưới 50% yêu cầu	Đáp ứng 50% - dưới 70% yêu cầu	Đáp ứng 70% - dưới 80% yêu cầu	Đáp ứng 80% - 100% yêu cầu		
CLO5, CLO6	Nội dung chính 3: Sử dụng được các thuật toán phát hiện cạnh như Sobel, Prewitt, Laplacian of Gaussian, và Canny để phát hiện cạnh của hình ảnh. Lập trình được các giải thuật xử lý ảnh cơ bản và nâng cao bằng Python để giải quyết các bài toán thực tế (30%)	Đáp ứng dưới 30% yêu cầu	Đáp ứng 30% - dưới 50% yêu cầu	Đáp ứng 50% - dưới 70% yêu cầu	Đáp ứng 70% - dưới 80% yêu cầu	Đáp ứng 80% - 100% yêu cầu		
<b>TỔNG CỘNG:</b>								
<b>ĐIỂM TRUNG BÌNH</b>								

TP. HCM, ngày ... tháng ... năm .....

**Giảng viên hướng dẫn**

(Ký và ghi rõ họ tên)

Phạm Quốc Thiện

TP. HCM, ngày ... tháng ... năm .....

**Giảng viên phản biện**

(Ký và ghi rõ họ tên)

Phạm Quốc Phương



**Viện Kỹ thuật HUTECH**

## PHIẾU CHẤM ĐIỂM

### HỌC PHẦN XỬ LÝ ẢNH SỐ

14. **Họ và tên sinh viên:** Nguyễn Chấn Huy.....

Lớp : 22DRTA1..... MSSV: 2286300020.....

Ngành : Robot và Trí tuệ nhân tạo.....

Chuyên ngành : Robot và Trí tuệ nhân tạo.....

15.. **Họ và tên giảng viên hướng dẫn (GVHD):** Phạm Quốc Thiện

16.. **Họ và tên giảng viên phản biện (GVPB):** Phạm Quốc Phương

17. **Đánh giá kết quả theo thang điểm 10 (Ghi rõ điểm số và điểm chữ)** .....

.....

CĐR	Tiêu chí đánh giá (trọng số)	Mức chất lượng ( $\geq 4$ : đạt)					Điểm	
		Mức F (0-3.9)	Mức D (4.0-5.4)	Mức C (5.5-6.9)	Mức B (7.0-8.4)	Mức A (8.5-10)	GVHD (50%)	GVPB (50%)
CĐO1, CĐO6	Hình thức – nội dung báo cáo Trả lời câu hỏi và trao đổi trong phần thảo luận. (15%)	Đáp ứng dưới 30% yêu cầu  Không trả lời đúng câu hỏi nào	Đáp ứng 30% - dưới 50% yêu cầu  Trả lời đúng  dưới 1/2 số câu hỏi	Đáp ứng 50% - dưới 70% yêu cầu  Trả lời đúng  1/2 số câu hỏi	Đáp ứng 70% - dưới 80% yêu cầu  Trả lời đúng  trên 2/3 số câu hỏi	Đáp ứng 80% - 100% yêu cầu  Trả lời đúng  tất cả các câu hỏi		
CĐO2	Nội dung chính 1: Thực hiện được các phép biến đổi ảnh cơ bản như chuyển đổi ảnh RGB thành ảnh xám, tính toán và phân tích biểu đồ Histogram (15%)	Đáp ứng dưới 30% yêu cầu	Đáp ứng 30% - dưới 50% yêu cầu	Đáp ứng 50% - dưới 70% yêu cầu	Đáp ứng 70% - dưới 80% yêu cầu	Đáp ứng 80% - 100% yêu cầu		

CLO3, CLO4	Nội dung chính 2: Áp dụng được các toán tử điểm, xử lý Histogram và ngưỡng hóa ảnh (threshold) để cải thiện chất lượng hình ảnh Áp dụng được các bộ lọc khôi phục ảnh như bộ lọc nghịch đảo và bộ lọc Wiener để cải thiện ảnh bị nhiễu (40%)	Đáp ứng dưới 30% yêu cầu	Đáp ứng 30% - dưới 50% yêu cầu	Đáp ứng 50% - dưới 70% yêu cầu	Đáp ứng 70% - dưới 80% yêu cầu	Đáp ứng 80% - 100% yêu cầu		
CLO5, CLO6	Nội dung chính 3: Sử dụng được các thuật toán phát hiện cạnh như Sobel, Prewitt, Laplacian of Gaussian, và Canny để phát hiện cạnh của hình ảnh. Lập trình được các giải thuật xử lý ảnh cơ bản và nâng cao bằng Python để giải quyết các bài toán thực tế (30%)	Đáp ứng dưới 30% yêu cầu	Đáp ứng 30% - dưới 50% yêu cầu	Đáp ứng 50% - dưới 70% yêu cầu	Đáp ứng 70% - dưới 80% yêu cầu	Đáp ứng 80% - 100% yêu cầu		
<b>TỔNG CỘNG:</b>								
<b>ĐIỂM TRUNG BÌNH</b>								

TP. HCM, ngày ... tháng ... năm .....

**Giảng viên hướng dẫn**

(Ký và ghi rõ họ tên)

Phạm Quốc Thiện

TP. HCM, ngày ... tháng ... năm .....

**Giảng viên phản biện**

(Ký và ghi rõ họ tên)

Phạm Quốc Phương

# Mục lục

<b>CHƯƠNG 1: GIỚI THIỆU</b>	14
1.1 Mục tiêu đề tài	14
1.2 Ý nghĩa thực tiễn	14
1.3 Phạm vi nghiên cứu	15
1.4 Cấu trúc báo cáo	16
<b>CHƯƠNG 2: TỔNG QUAN VỀ ROBOT MICROMOUSE</b>	18
2.1. Khái niệm và ứng dụng	18
2.2. Lịch sử phát triển của Micromouse	19
2.3. Cấu trúc cơ bản của Robot Micromouse	20
2.3.1. Phần cứng	20
2.3.2. Phần mềm	24
2.4. Các thách thức khi di chuyển trong mê cung	26
<b>CHƯƠNG 3: PHÂN TÍCH GIẢI THUẬT DI CHUYỂN TRONG MÊ CUNG</b>	28
3.1. Tổng quan về các giải thuật tìm đường	28
3.1.1. Giải thuật Flood-Fill	28
3.1.2. Giải thuật A*	29
3.1.3. Giải thuật Wall-Following	31
3.2. Lựa chọn giải thuật cho Micromouse	32
3.3. Phân tích chi tiết giải thuật đã chọn	33
3.3.1. Mô tả giải thuật	33
3.3.2. Cách thức hoạt động trong các tình huống cụ thể	34
3.3.3. Tối ưu hóa giải thuật	35
<b>CHƯƠNG 4: THIẾT KẾ VÀ TRIỂN KHAI</b>	36
4.1. Thiết kế phần cứng	36
4.1.1. Mạch điều khiển và cảm biến	36
4.1.2. Động cơ và bộ mã hóa vòng quay (Encoder)	38
4.1.3. Khung sườn và bố trí linh kiện	39
4.2. Thiết kế phần mềm	41
4.2.1. Cấu trúc chương trình	41
4.2.2. Các module chính	42
4.3. Kết nối và tích hợp	43
4.4. Kiểm thử và chỉnh sửa	44
<b>CHƯƠNG 5: KẾT QUẢ THỰC NGHIỆM</b>	45
5.1. Quá trình chạy thử nghiệm	45

5.2. Phân tích hiệu suất di chuyển .....	46
5.3. Nhận xét về kết quả đạt được .....	47
5.4. Các tình huống gặp phải và cách khắc phục .....	48
<b>CHƯƠNG 6: ĐÁNH GIÁ VÀ KẾT LUẬN .....</b>	<b>50</b>
6.1. Đánh giá tổng quan về đồ án .....	50
6.2. Ưu điểm và hạn chế của giải pháp .....	51
6.3. Đề xuất cải tiến trong tương lai .....	51
6.4. Kết luận .....	52
<b>PHỤ LỤC .....</b>	<b>53</b>
A: Sơ đồ mạch điện .....	53
B: Mã nguồn chương trình .....	54
<b>TÀI LIỆU THAM KHẢO .....</b>	<b>60</b>
[1] Sách, báo, bài viết khoa học .....	60
[2] Website, tài liệu online .....	60
[3] Các nguồn tham khảo khác .....	61

## Mục lục hình ảnh

<i>Hình 1. Hình ảnh các micromouse hiện đại ngày nay</i>	<i>18</i>
<i>Hình 2. Một trong những hình dạng của micromouse đầu tiên</i>	<i>19</i>
<i>Hình 3. Cảm biến led hồng ngoại 940nm</i>	<i>20</i>
<i>Hình 4. Động cơ GA12-N20 encoder 1:100</i>	<i>21</i>
<i>Hình 5. Arduino Nano V3.0 ATmega328P</i>	<i>22</i>
<i>Hình 6. Mạch Điều Khiển Động Cơ DC TB6612 mini</i>	<i>22</i>
<i>Hình 7. Khung PCB Ukmarsbot Micromouse</i>	<i>23</i>
<i>Hình 8. Bố trí các cảm biến led IR trên mạch</i>	<i>24</i>
<i>Hình 9. Mô tả hướng đi của Ukamarsbot</i>	<i>25</i>
<i>Hình 10. Mô tả về giải thuật Flood-Fill</i>	<i>28</i>
<i>Hình 11. Mô tả thuật toán <math>A^*</math></i>	<i>29</i>
<i>Hình 12. Mô tả thuật toán Wall-Following</i>	<i>31</i>
<i>Hình 13. Sơ đồ khối mô tả thuật toán left wall following</i>	<i>33</i>
<i>Hình 14. Hình ảnh thực tế của mê cung</i>	<i>45</i>
<i>Hình 15. Mô hình hoàn thiện</i>	<i>50</i>
<i>Hình 16. Sơ đồ mạch điện main board</i>	<i>53</i>
<i>Hình 17. Sơ đồ mạch điện wall sensors</i>	<i>53</i>

# CHƯƠNG 1: GIỚI THIỆU

## 1.1 Mục tiêu đề tài

Ngày nay, sự phát triển vượt bậc của công nghệ robot và tự động hóa đã mang lại nhiều giải pháp hữu ích trong cuộc sống, từ việc hỗ trợ con người trong sản xuất công nghiệp đến cải thiện các hoạt động hàng ngày. Trong đó, robot tự hành là một lĩnh vực thu hút sự chú ý lớn nhờ khả năng thích nghi và giải quyết các vấn đề phức tạp trong môi trường thực tế.

Đề tài "**Micromouse UKMARSBot - Giải Thuật Di Chuyển Trong Mê Cung**" nhằm nghiên cứu và phát triển một robot nhỏ gọn có khả năng tự định hướng và tìm đường trong mê cung. Đây không chỉ là bài toán mang tính thử thách về mặt kỹ thuật mà còn mở ra những tiềm năng ứng dụng trong các lĩnh vực như logistics, quản lý kho bãi, và thám hiểm trong môi trường không gian hạn chế.

Mục tiêu chính của đề tài là xây dựng một nền tảng robot tự hành có thể hoạt động ổn định, minh họa rõ nét cách kết hợp giữa phần cứng và phần mềm để giải quyết bài toán di chuyển trong không gian phức tạp. Qua đó, đề tài sẽ giúp người đọc hiểu rõ hơn về nguyên lý hoạt động, cấu trúc, và cách thức điều khiển của robot Micromouse, đồng thời khẳng định vai trò quan trọng của công nghệ này trong thời đại hiện nay.

## 1.2 Ý nghĩa thực tiễn

### **Phát triển kỹ năng tư duy thuật toán:**

Đề tài tạo cơ hội để nhóm áp dụng và thực hành các thuật toán cơ bản, như Wall Following, nhằm giải quyết bài toán điều khiển robot tự hành trong môi trường mê cung. Việc tiếp xúc với bài toán này giúp nhóm phát triển khả năng tư duy logic, phân tích, và tối ưu hóa thuật toán.

### **Tăng cường hiểu biết về các giải thuật định hướng:**

Bên cạnh việc triển khai thuật toán Wall Following, nhóm cũng nghiên cứu và tìm hiểu sơ bộ các giải thuật định hướng nâng cao như Flood Fill hoặc A\*. Mặc dù không trực tiếp áp dụng các thuật toán này trong đề tài, việc tìm hiểu giúp nhóm mở rộng hiểu biết về cách tiếp cận bài toán di chuyển trong không gian phức tạp, tạo nền tảng cho các nghiên cứu trong tương lai.

### Ứng dụng trong thực tiễn:

Công nghệ điều hướng của Micromouse có tiềm năng mở rộng ra nhiều ứng dụng thực tế. Các hệ thống điều hướng tự động này có thể được áp dụng trong xe tự hành, quản lý kho vận bằng robot tự động, hoặc các hệ thống tìm kiếm và cứu hộ trong các môi trường nguy hiểm. Đề tài đóng vai trò như một mô hình thử nghiệm, minh họa cách các thuật toán điều khiển cơ bản có thể áp dụng trong thực tế.

### Giá trị giáo dục và nghiên cứu:

Đề tài này yêu cầu sự tích hợp của nhiều lĩnh vực, bao gồm cơ khí, điện tử, lập trình nhúng, và xử lý tín hiệu. Thông qua việc triển khai robot Micromouse, nhóm có cơ hội tổng hợp và vận dụng kiến thức từ nhiều môn học, đồng thời làm quen với các công nghệ hiện đại và thực tiễn trong ngành robotics.

### Nền tảng cho phát triển công nghệ:

Việc xây dựng và thử nghiệm một robot Micromouse như UKMARScbot không chỉ giúp nhóm hiểu sâu hơn về cơ chế hoạt động của robot tự hành mà còn đặt nền móng cho việc nghiên cứu và phát triển các hệ thống phức tạp hơn trong tương lai.

## 1.3 Phạm vi nghiên cứu

Đề tài tập trung vào các khía cạnh chính sau để đảm bảo mục tiêu nghiên cứu được thực hiện một cách hiệu quả và thực tiễn:

- Thiết kế phần cứng:

Sử dụng các cảm biến hồng ngoại để phát hiện tường và định vị, kết hợp với động cơ DC và bộ mã hóa vòng quay (encoder) để điều khiển chính xác chuyển động của robot.

Tối ưu thiết kế khung robot, đảm bảo kích thước nhỏ gọn phù hợp với mô hình mê cung chuẩn nhưng vẫn giữ được sự ổn định khi di chuyển ở tốc độ cao hoặc khi quay gấp.

- Phát triển thuật toán giải mê cung:

Triển khai và cải tiến thuật toán **bám tường (Wall Following)** để tìm đường trong mê cung, tập trung vào tính đơn giản và hiệu quả trong điều kiện thực tế.

Tìm hiểu sơ bộ các thuật toán khác như **Flood Fill** hoặc **A\*** nhằm mở rộng kiến thức, tuy nhiên không triển khai trực tiếp các thuật toán này trong đề tài.

- Giới hạn trong mô hình mê cung chuẩn:

Mô hình mê cung được xây dựng theo kích thước chuẩn của cuộc thi Micromouse (kích thước ô: 18x18 cm).

Mê cung có các cấu trúc ngẫu nhiên và không được cung cấp bản đồ trước, đòi hỏi robot phải tự dò đường và ghi nhớ bố cục trong thời gian thực.

- Kiểm thử và đánh giá:

Đánh giá hiệu suất của robot thông qua các thử nghiệm thực tế, bao gồm độ chính xác trong việc bám tường, thời gian hoàn thành lộ trình và tính ổn định trong các điều kiện mê cung khác nhau.

Kiểm tra khả năng tái lập, đảm bảo robot hoạt động ổn định và có thể tái sử dụng thuật toán trong các cấu hình mê cung mới.

#### 1.4 Cấu trúc báo cáo

Báo cáo được xây dựng với cấu trúc gồm **6 chương chính**, cùng các phần phụ lục và tài liệu tham khảo như sau:

Chương 1: Giới thiệu

Trình bày tổng quan về mục tiêu, ý nghĩa thực tiễn, phạm vi nghiên cứu của đề tài, và cung cấp cái nhìn khái quát về cấu trúc của báo cáo.

Chương 2: Tổng quan về robot Micromouse

Giới thiệu về robot Micromouse, bao gồm khái niệm, lịch sử phát triển, và các ứng dụng thực tiễn. Nội dung cũng đi sâu vào cấu trúc cơ bản của robot Micromouse với hai phần chính: phần cứng và phần mềm, đồng thời làm rõ các thách thức khi di chuyển trong môi trường mê cung.

Chương 3: Phân tích giải thuật di chuyển trong mê cung

Tập trung nghiên cứu các giải thuật tìm đường phổ biến như Flood-Fill, A\*, và Wall-Following. Từ đó, lựa chọn giải thuật phù hợp cho robot, phân tích chi tiết cơ chế hoạt động và đề xuất các giải pháp tối ưu hóa để đạt hiệu quả cao nhất khi triển khai thực tế.

Chương 4: Thiết kế và triển khai

Trình bày quá trình thiết kế và triển khai robot Micromouse. Bao gồm:

- Phần cứng: Chi tiết về mạch điều khiển, cảm biến, động cơ, bộ mã hóa vòng quay (encoder), và khung robot.
- Phần mềm: Cấu trúc chương trình, các module chính, tích hợp thuật toán bám tường (Wall-Following).



Nội dung cũng mô tả quá trình kết nối, tích hợp các thành phần, cùng quy trình kiểm thử để đảm bảo hoạt động ổn định.

#### Chương 5: Kết quả thực nghiệm

Tổng hợp kết quả thực nghiệm từ các bài kiểm thử thực tế của robot trong mê cung chuẩn. Đánh giá hiệu suất thông qua các tiêu chí như độ chính xác, thời gian hoàn thành, và tính ổn định. Phần này cũng ghi nhận các tình huống gặp phải, phương pháp khắc phục và những cải tiến trong quá trình chạy thử.

#### Chương 6: Đánh giá và kết luận

Tổng hợp lại toàn bộ quá trình thực hiện đề tài, đánh giá hiệu quả của hệ thống, và nêu bật các ưu điểm, hạn chế. Đề xuất các hướng nghiên cứu mở rộng và cải tiến trong tương lai.

#### Phụ lục:

Cung cấp các tài liệu bổ sung bao gồm sơ đồ mạch điện, mã nguồn chương trình, và các thông số kỹ thuật chi tiết của robot.

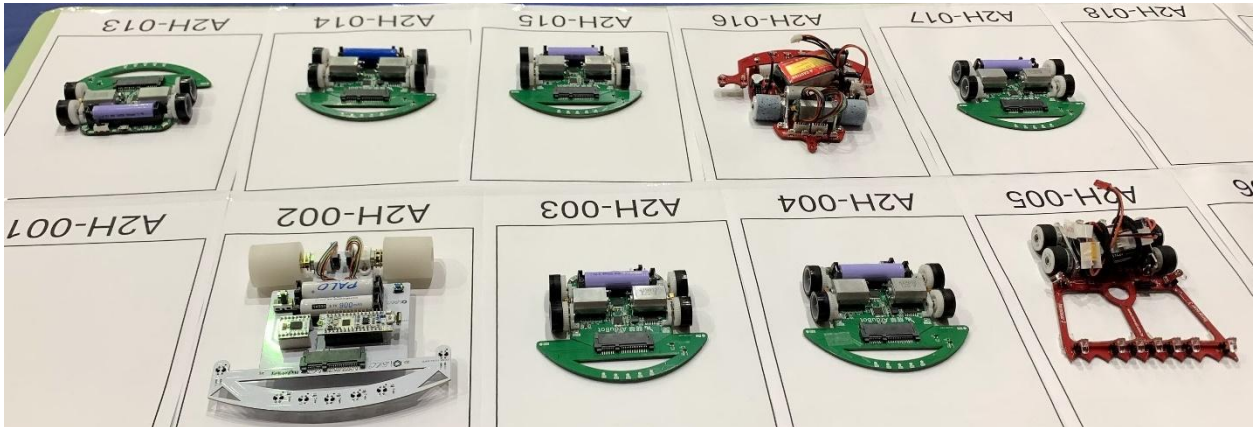
#### Tài liệu tham khảo:

Danh sách các nguồn tài liệu đã sử dụng trong báo cáo, bao gồm sách, bài báo khoa học, website, và các tài liệu tham khảo khác.

## CHƯƠNG 2: TỔNG QUAN VỀ ROBOT MICROMOUSE

### 2.1. Khái niệm và ứng dụng

**Micromouse** là một loại robot tự hành kích thước nhỏ, được thiết kế để khám phá và xác định lộ trình di chuyển tối ưu trong các mê cung phức tạp. Nhiệm vụ chính của robot Micromouse là tìm đường đi ngắn nhất từ điểm xuất phát đến vị trí đích, dựa trên các thuật toán điều hướng và điều khiển chính xác.



Hình 1. Hình ảnh các micromouse hiện đại ngày nay

#### Ứng dụng thực tế của Micromouse:

##### Đào tạo và giáo dục:

Micromouse được sử dụng rộng rãi trong các khóa học về robot học, điều khiển tự động, và trí tuệ nhân tạo. Các cuộc thi Micromouse giúp sinh viên và nhà nghiên cứu thực hành, kiểm tra kỹ năng lập trình, thiết kế phần cứng và tư duy thuật toán.

##### Nghiên cứu thuật toán:

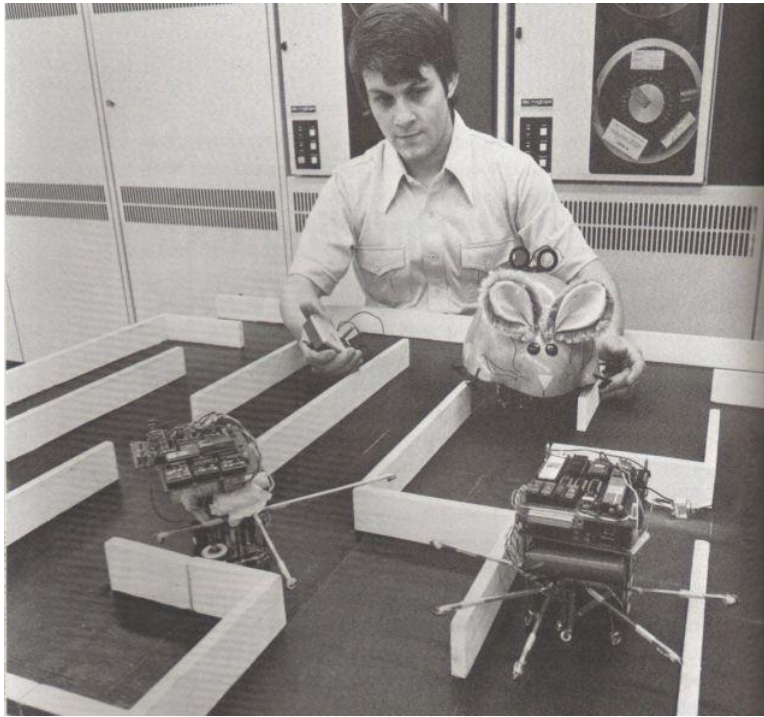
Micromouse là nền tảng lý tưởng để phát triển và thử nghiệm các thuật toán tìm đường, như Flood-Fill, A\*, hay Wall-Following, cũng như tối ưu hóa các phương pháp điều hướng trong môi trường thay đổi.

##### Ứng dụng công nghiệp:

Nguyên lý hoạt động của Micromouse có thể được mở rộng để áp dụng trong:

- Robot vận chuyển hàng hóa tự động tại các nhà kho thông minh.
- Hệ thống tự động hóa trong sản xuất, giúp tối ưu hóa quy trình di chuyển và phân phối.
- Robot tìm kiếm và cứu hộ, hoạt động trong các khu vực nguy hiểm hoặc địa hình phức tạp.

## 2.2. Lịch sử phát triển của Micromouse



*Hình 2. Một trong những hình dạng của micromouse đầu tiên*

Micromouse ra đời vào những năm **1970**, khi các kỹ sư và nhà nghiên cứu bắt đầu thử nghiệm các robot di chuyển trong mê cung để giải quyết bài toán tìm đường. Đây là giai đoạn mở đầu cho việc nghiên cứu và phát triển các thuật toán tìm đường, điều khiển và điều hướng cho robot tự hành.

**1977:** Cuộc thi Micromouse đầu tiên được tổ chức tại Mỹ bởi **IEEE Spectrum**, đánh dấu một cột mốc quan trọng trong sự phát triển của Micromouse. Cuộc thi này không chỉ nhằm thúc đẩy sự sáng tạo mà còn tạo cơ hội để các kỹ sư, nhà nghiên cứu và sinh viên thử nghiệm các mô hình robot di chuyển trong môi trường mê cung.

**1980s:** Sự phát triển vượt bậc của công nghệ cảm biến và **vi điều khiển** đã làm tăng đáng kể khả năng của Micromouse. Các cảm biến và bộ mã hóa (encoder) chính xác hơn giúp robot có khả năng nhận diện và đo lường chính xác môi trường xung quanh, từ đó tối ưu hóa hành trình di chuyển trong mê cung.

**1990s - Hiện nay:** Micromouse đã trở thành một môn thi đấu quốc tế phổ biến, với các sự kiện được tổ chức tại nhiều quốc gia như **Nhật Bản, Hàn Quốc, Ấn Độ, và Anh**. Những cuộc thi này không chỉ là cơ hội để các nhà phát triển thể hiện kỹ năng mà còn giúp thúc đẩy sự đổi mới và sáng tạo trong lĩnh vực robot tự hành.

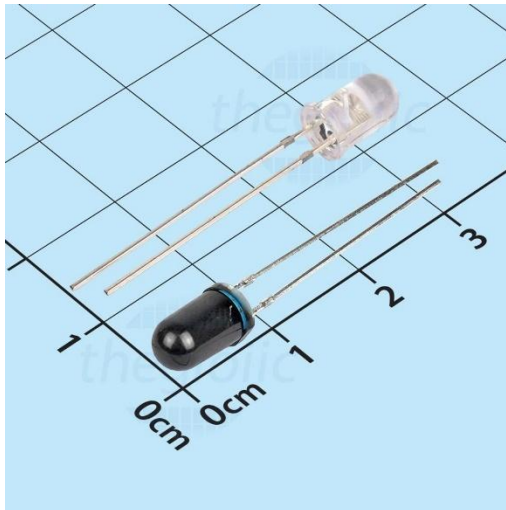
Sự tiến bộ vượt bậc trong công nghệ, đặc biệt là trong lĩnh vực **cảm biến**, **bộ mã hóa**, và **thuật toán điều khiển**, đã giúp các robot Micromouse hiện đại đạt được tốc độ di chuyển nhanh hơn và độ chính xác cao hơn, từ đó mở rộng khả năng ứng dụng trong nhiều lĩnh vực khác nhau.

## 2.3. Cấu trúc cơ bản của Robot Micromouse

### 2.3.1. Phần cứng

Phần cứng của robot Micromouse bao gồm các thành phần chính, trong đó mỗi thành phần đóng vai trò quan trọng trong việc giúp robot thực hiện nhiệm vụ tìm đường và di chuyển trong mê cung một cách chính xác, ngoài ra còn tùy theo hình dạng của chúng còn giúp việc di chuyển trong mê cung tối ưu hơn. Các thành phần phần cứng chính của robot **UKMARSBOT** bao gồm:

#### **Cảm biến:**

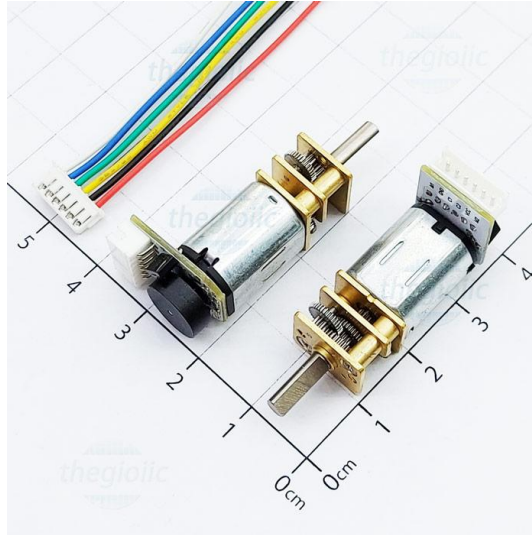


Hình 3. Cảm biến led hồng ngoại 940nm

#### **Cảm biến hồng ngoại (IR):**

UKMARSBOT sử dụng 4 cặp **LED hồng ngoại 940nm** để làm cảm biến khoảng cách, hay còn gọi là **wall sensors**. Các cảm biến này giúp robot phát hiện tường của mê cung xung quanh. Mỗi cặp cảm biến bao gồm một **LED phát hồng ngoại** và một **phát hiện hồng ngoại (photodiode)**. Khi ánh sáng hồng ngoại bị phản xạ từ tường hoặc vật thể, photodiode nhận tín hiệu này, cho phép robot xác định khoảng cách đến các vật cản. Việc sử dụng 4 cặp cảm biến (2 cặp ở phía trái và 2 cặp ở phía phải) giúp robot có khả năng nhận diện và tránh các chướng ngại vật ở cả hai bên, từ đó điều chỉnh hành trình di chuyển.

### Động cơ và bộ mã hóa vòng quay (Encoder):



Hình 4. Động cơ GA12-N20 encoder 1:100

#### Động cơ DC GA12-N20:

Robot sử dụng động cơ DC **GA12-N20** với bộ mã hóa (encoder) tích hợp. Động cơ này có tỷ số truyền **1:100**, nghĩa là mỗi vòng quay của trục động cơ sẽ tạo ra 100 vòng quay của trục encoder. Mặc dù có thể sử dụng động cơ với tỷ số truyền **1:30** để đạt được tốc độ cao hơn, nhưng trong quá trình tìm kiếm linh kiện, **1:100** đã được lựa chọn do không còn sẵn các động cơ tỷ số truyền 1:30. Tuy tỷ số truyền này thấp hơn, nhưng nó giúp robot có sự ổn định tốt hơn trong việc điều khiển và di chuyển trong mê cung.

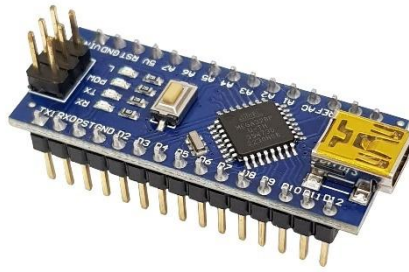
#### Bộ mã hóa (Encoder):

Bộ mã hóa giúp đo lường tốc độ quay của động cơ và tính toán khoảng cách mà robot đã di chuyển. Bộ mã hóa trên động cơ GA12-N20 cung cấp tín hiệu đầu ra giúp hệ thống điều khiển biết chính xác vị trí và hướng di chuyển của robot. Điều này rất quan trọng trong việc điều khiển chính xác, đặc biệt khi robot phải di chuyển qua các đoạn đường cong và thay đổi hướng trong mê cung.

#### Mạch điều khiển:

Trong hệ thống UKMARSBOT, mạch điều khiển chịu trách nhiệm xử lý dữ liệu từ cảm biến, thực hiện các lệnh điều khiển và điều phối động cơ để robot có thể di chuyển chính xác trong mê cung. Cấu trúc mạch điều khiển bao gồm các thành phần chính sau:

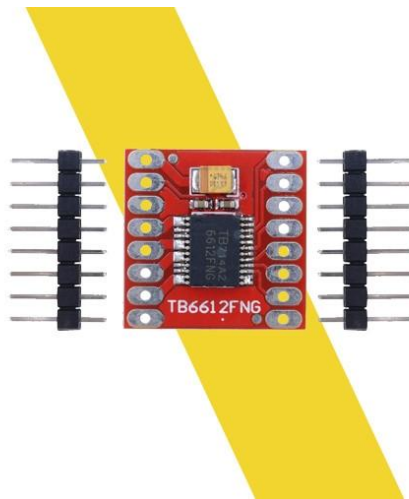
Vi điều khiển (Arduino Nano):



*Hình 5. Arduino Nano V3.0 ATmega328P*

Arduino Nano được sử dụng làm vi điều khiển chính cho robot. Đây là một vi điều khiển nhỏ gọn, dễ dàng lập trình và có khả năng tương thích tốt với các cảm biến và mạch điều khiển động cơ. Vi điều khiển này xử lý các tín hiệu từ các cảm biến IR, tính toán hướng di chuyển và điều khiển động cơ thông qua mạch H-Bridge.

Mạch điều khiển động cơ (TB6612):



*Hình 6. Mạch Điều Khiển Động Cơ DC TB6612 mini*

Để điều chỉnh tốc độ và hướng của động cơ DC, mạch driver TB6612 được sử dụng. TB6612 là một driver động cơ H-Bridge phổ biến, có khả năng điều khiển hai động cơ DC với tính năng điều chỉnh chiều quay và tốc độ. Mạch này nhận các tín hiệu từ Arduino và điều khiển động cơ theo các lệnh cụ thể, đảm bảo rằng robot có thể di chuyển theo các hướng xác định trong mê cung.



Mạch TB6612 cung cấp khả năng điều khiển chính xác động cơ, giúp robot di chuyển một cách linh hoạt và ổn định, đồng thời giảm thiểu độ trễ trong các phản ứng điều khiển. Tất cả các tín hiệu điều khiển từ Arduino Nano được truyền qua mạch TB6612, điều chỉnh tốc độ và hướng của động cơ để robot có thể tiếp tục quá trình tìm đường và giải quyết mê cung một cách hiệu quả.

#### **Nguồn năng lượng:**

Để cung cấp năng lượng cho **UKMARSBOT**, hệ thống sử dụng **pin vuông nhỏ gọn 9V**. Đây là một lựa chọn nguồn năng lượng phổ biến, dễ dàng thay thế và sạc lại, giúp duy trì hoạt động của robot trong thời gian dài. Các tính năng nổi bật của nguồn năng lượng này bao gồm:

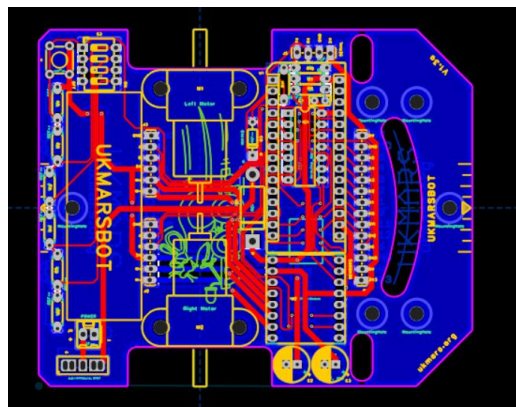
- **Pin vuông 9V:**

Pin vuông 9V cung cấp điện áp ổn định, phù hợp với yêu cầu của các vi điều khiển và mạch điều khiển động cơ. Với khả năng cấp nguồn liên tục, pin này giúp robot hoạt động hiệu quả trong suốt quá trình tìm đường trong mê cung.

- **Cổng sạc Type-C:**

Để tiện lợi trong việc sạc lại, pin sử dụng **cổng sạc Type-C**, cho phép người sử dụng dễ dàng sạc pin qua các thiết bị sạc hiện đại, giúp tiết kiệm thời gian và bảo vệ môi trường. Việc sử dụng cổng Type-C mang lại tính tiện dụng cao, đồng thời hỗ trợ sạc nhanh và hiệu quả.

#### **Khung sườn:**



*Hình 7. Khung PCB Ukmarsbot Micromouse*

Khung sườn của robot UKMARSBot được thiết kế trên nền tảng PCB (Printed Circuit Board) nhằm tối ưu hóa không gian và độ bền của hệ thống. Mainboard PCB không chỉ chứa các linh kiện quan trọng mà còn được thiết kế một cách nhỏ gọn, giúp giảm thiểu

diện tích sử dụng và tăng tính ổn định. Các khu vực gắn kết cho cảm biến hồng ngoại (IR), động cơ, và các bộ phận khác được sắp xếp gọn gàng, khoa học, giúp việc di chuyển của robot trở nên linh hoạt và dễ dàng hơn trong không gian hạn chế của mê cung.

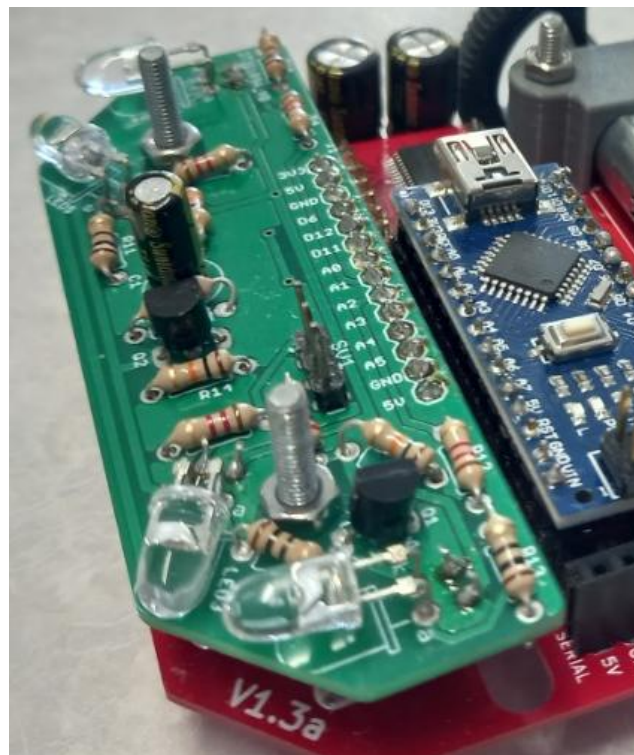
Cảm biến Wallsensor được gắn ở phía trên của mạch, cho phép robot quét và nhận diện các bức tường xung quanh một cách chính xác. Các tụ điện và linh kiện điện tử khác được phân bố hợp lý trên PCB, không chỉ đảm bảo hiệu suất hoạt động tối ưu mà còn tạo ra một thiết kế tinh gọn, thuận tiện cho việc điều khiển robot đi theo các đường chéo hoặc mô phỏng các chuyển động phức tạp trong mê cung.

Với thiết kế nhỏ gọn và khéo léo này, robot UKMARSbot có thể di chuyển qua các góc ngách trong mê cung một cách linh hoạt, đồng thời giữ cho hệ thống ổn định và dễ dàng lắp ráp, bảo trì. Khả năng này đặc biệt hữu ích khi robot cần phải tối ưu hóa đường đi, tiết kiệm thời gian và không gian trong các cuộc thi hoặc thử nghiệm thực tế.

### 2.3.2. Phần mềm

Phần mềm điều khiển robot Micromouse UKMARSbot được thiết kế để xử lý dữ liệu cảm biến, áp dụng các thuật toán điều hướng và điều khiển động cơ một cách hiệu quả. Dưới đây là các yếu tố chính trong phần mềm điều khiển:

#### **Xử lý dữ liệu cảm biến:**



Hình 8. Bố trí các cảm biến led IR trên mạch



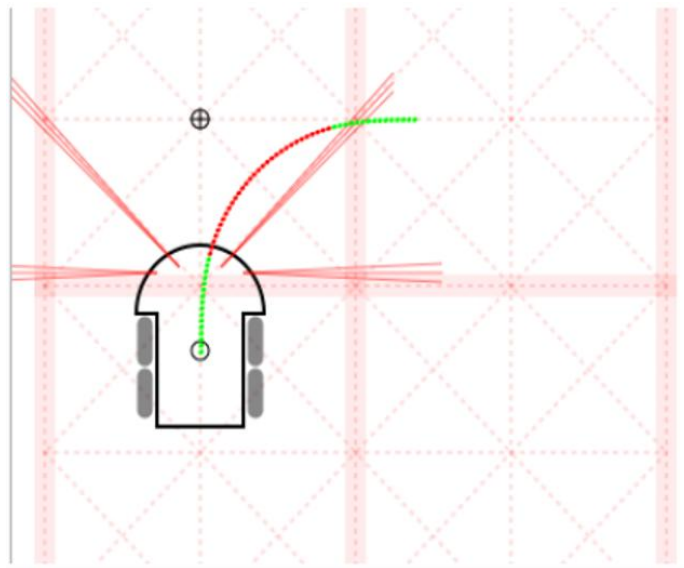
Các cảm biến hồng ngoại (IR) được sử dụng để đo khoảng cách đến các bức tường và vật cản trong mê cung. Cụ thể, các cảm biến IR như cảm biến "trước bên trái", "chéo trái", "trước phải", và "chéo phải" sẽ cung cấp tín hiệu analog giúp robot xác định vị trí của các vật cản xung quanh. Dựa trên giá trị analog mà các cảm biến trả về, robot có thể đánh giá khoảng cách đến các bức tường: khi giá trị analog cao, nghĩa là robot gần tường, và khi giá trị thấp, robot đang ở xa tường. Những tín hiệu này được sử dụng để quyết định hướng di chuyển và điều chỉnh tốc độ của robot.

### **Thuật toán di chuyển:**

Robot sử dụng thuật toán *Wall-Following*, cho phép di chuyển dọc theo các bức tường trong mê cung để tìm đường đi đến đích. Thuật toán này kết hợp với các giá trị cảm biến IR trả về để điều chỉnh chuyển động của robot. Khi cảm biến ở phía bên trái hoặc bên phải báo hiệu rằng tường gần, robot sẽ điều chỉnh hướng di chuyển, có thể rẽ trái, rẽ phải, hoặc đi thẳng tùy theo sự thay đổi của các giá trị cảm biến.

### **Điều khiển động cơ:**

Phần mềm sử dụng thuật toán PID (Proportional-Integral-Derivative) để điều khiển động cơ. PID giúp điều chỉnh tốc độ và hướng di chuyển của robot một cách chính xác và ổn định, giúp robot duy trì đường đi thẳng, quay chính xác khi rẽ, hoặc thay đổi hướng khi cần thiết. Các tham số PID được điều chỉnh sao cho robot có thể di chuyển một cách mượt mà và hiệu quả trong mê cung, tránh va chạm và tiếp cận đích nhanh chóng.



Hình 9. Mô tả hướng đi của Ukamarsbot

## 2.4. Các thách thức khi di chuyển trong mê cung

Việc điều hướng trong mê cung là một bài toán phức tạp đối với robot Micromouse, đặc biệt là khi đối mặt với các thách thức liên quan đến cảm biến, thuật toán di chuyển, và phần cứng. Dưới đây là các thách thức lớn khi robot Micromouse di chuyển trong mê cung:

### **Xác định và duy trì hướng di chuyển:**

Một trong những nhiệm vụ quan trọng của robot là phải xác định chính xác vị trí của mình trong mê cung và duy trì hướng di chuyển đúng. Các cảm biến hồng ngoại giúp robot nhận diện các bức tường xung quanh, nhưng robot cần phải điều chỉnh hướng di chuyển sao cho phù hợp, đặc biệt là khi di chuyển theo các đoạn đường cong hoặc khi gặp các giao lộ. Để duy trì hướng chính xác, robot phải liên tục cập nhật dữ liệu từ cảm biến và điều chỉnh chuyển động.

### **Xử lý dữ liệu cảm biến trong môi trường thực:**

Môi trường thực tế không phải lúc nào cũng hoàn hảo. Các cảm biến có thể gặp phải nhiễu từ các yếu tố như ánh sáng thay đổi, sự phản xạ không đồng đều từ bề mặt tường hoặc độ chính xác của các cảm biến khi di chuyển trong không gian hẹp. Những yếu tố này có thể gây sai lệch trong việc đọc dữ liệu từ cảm biến và làm robot di chuyển sai hướng. Do đó, cần phải có các phương pháp xử lý dữ liệu chính xác và ổn định, như việc đọc lại cảm biến nhiều lần để giảm thiểu sai sót.

### **Tối ưu hóa lộ trình:**

Việc tìm ra lộ trình ngắn nhất từ điểm xuất phát đến điểm đích trong mê cung là một thử thách lớn. Các thuật toán như Wall-Following hay A\* được sử dụng để tìm đường, nhưng đôi khi robot có thể gặp phải tình huống mà lộ trình không hiệu quả, như việc bị mắc kẹt hoặc đi vòng quanh các khu vực đã đi qua. Điều này yêu cầu tối ưu hóa thuật toán và lộ trình di chuyển, để đảm bảo robot không lặp lại các đường đi cũ mà không tiến về đích.

### **Hạn chế về phần cứng:**

Kích thước nhỏ gọn của robot Micromouse mang lại sự linh hoạt trong di chuyển, nhưng đồng thời cũng tạo ra các hạn chế về phần cứng. Các linh kiện điện tử phải được tối ưu hóa sao cho vừa tiết kiệm không gian mà vẫn đảm bảo hiệu suất hoạt động. Điều này đặc biệt quan trọng khi phải tích hợp nhiều cảm biến, động cơ và mạch điều khiển

vào một không gian rất nhỏ. Việc sử dụng các linh kiện như động cơ và cảm biến với kích thước nhỏ nhưng hiệu suất cao là cần thiết để đảm bảo robot hoạt động tốt.

### **Thời gian phản hồi và xử lý:**

Một trong những yếu tố quan trọng đối với robot Micromouse là thời gian phản hồi nhanh chóng khi có sự thay đổi trong môi trường, như khi gặp các vật cản hoặc khi cần điều chỉnh hướng di chuyển. Robot phải có khả năng xử lý dữ liệu cảm biến và đưa ra quyết định di chuyển trong thời gian ngắn để tránh va chạm hoặc bị lạc hướng. Điều này đòi hỏi phần mềm điều khiển phải có khả năng xử lý nhanh chóng và chính xác, đồng thời các thuật toán như PID phải được tinh chỉnh để đảm bảo tốc độ phản hồi phù hợp.

### **Cải tiến và tối ưu hóa PID:**

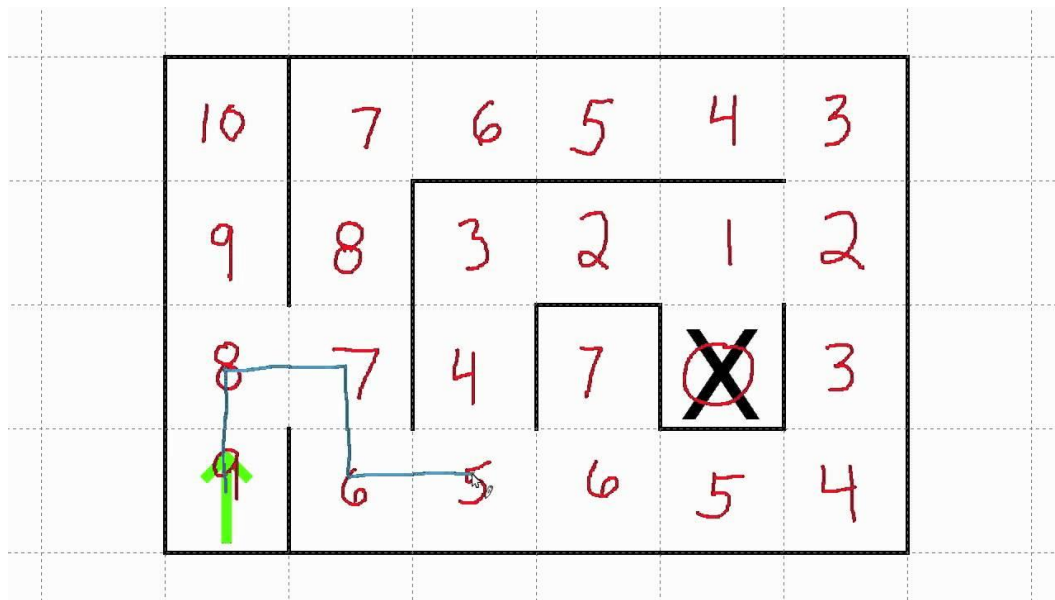
Để vượt qua các thách thức trên, một yếu tố quan trọng là việc cải tiến và tối ưu hóa thuật toán PID (Proportional-Integral-Derivative). Khi di chuyển trong mê cung, PID giúp điều chỉnh tốc độ và hướng của động cơ. Tuy nhiên, với các môi trường có sự thay đổi liên tục như mê cung, cần phải điều chỉnh tham số PID sao cho phù hợp với các tình huống thực tế. Cũng cần đọc lại cảm biến nhiều lần để xác định chính xác vị trí và khoảng cách đến tường, từ đó giúp cải thiện độ ổn định trong việc điều khiển robot.

## CHƯƠNG 3: PHÂN TÍCH GIẢI THUẬT DI CHUYỂN TRONG MÊ CUNG

### 3.1. Tổng quan về các giải thuật tìm đường

Trong quá trình di chuyển trong mê cung, việc lựa chọn giải thuật tìm đường đóng vai trò rất quan trọng, vì nó ảnh hưởng trực tiếp đến hiệu quả di chuyển và thời gian robot cần để đạt được mục tiêu. Dưới đây là một số giải thuật phổ biến mà robot Micromouse có thể sử dụng:

#### 3.1.1. Giải thuật Flood-Fill



Hình 10. Mô tả về giải thuật Flood-Fill

#### Nguyên lý hoạt động:

Giải thuật Flood-Fill được sử dụng để tìm đường đi từ vị trí của robot đến mục tiêu trong mê cung bằng cách gán giá trị khoảng cách cho từng ô trong mê cung. Ban đầu, robot sẽ đặt mục tiêu ở ô đích và gán giá trị 0 cho ô đó. Sau đó, giá trị khoảng cách của các ô xung quanh sẽ được gán lần lượt bằng giá trị của ô mục tiêu cộng thêm 1. Quá trình này tiếp tục cho đến khi tất cả các ô có giá trị được tính toán. Khi robot di chuyển trong mê cung, nó sẽ chọn đi theo ô có giá trị thấp nhất để tiến về mục tiêu.

#### Ưu điểm:

Tìm được đường đi ngắn nhất: Giải thuật này đảm bảo tìm được đường đi ngắn nhất từ vị trí hiện tại đến mục tiêu vì nó tính toán dần từ mục tiêu ra các ô xung quanh.



Để dàng cập nhật khi phát hiện các rào cản mới: Khi robot gặp phải các chướng ngại vật hoặc tường mới trong mê cung, giải thuật có thể cập nhật lại bản đồ dễ dàng bằng cách thay đổi giá trị khoảng cách của các ô liên quan.

#### Nhược điểm:

Cần bộ nhớ lớn: Để lưu trữ bản đồ mê cung và giá trị khoảng cách cho tất cả các ô, giải thuật Flood-Fill yêu cầu một bộ nhớ đáng kể, điều này có thể là một vấn đề đối với các robot có giới hạn về phần cứng.

Thời gian tính toán tăng lên khi mê cung có kích thước lớn: Khi mê cung có kích thước lớn hoặc phức tạp, thời gian để tính toán tất cả các giá trị khoảng cách có thể rất lâu, làm giảm hiệu suất của robot.

#### 3.1.2. Giải thuật A\*

$\begin{matrix} x \\ y \end{matrix}$	0	1	2	3	4	5	6	7	8
0		3.2	5.2	7.1	9.1	11	13	15	17
1	3.2	4.8		8.5	10	12	14	16	18
2	5.2	6.6		10			16	18	19
3	7.1	8.5		12		15	17	19	21
4	9.1	10	12	13		17		20	22
5	11	12	14	15		18	20	22	24
6	13	14	16			20	22	24	25
7	15	16	18	19	20	22	24	25	27

Hình 11. Mô tả thuật toán A\*

#### Nguyên lý hoạt động:

A\* (A-star) là một giải thuật tìm đường heuristic sử dụng một hàm chi phí kết hợp giữa chi phí thực tế và chi phí dự đoán để tìm đường đi ngắn nhất từ điểm xuất phát đến mục tiêu. Cụ thể, A\* tính toán tổng chi phí  $f$  từ điểm xuất phát đến ô hiện tại, với công thức:

$$f=g+h$$

Trong đó:

$g$  là chi phí thực tế (hoặc khoảng cách) từ điểm xuất phát đến ô hiện tại.

$h$  là chi phí dự đoán (hoặc khoảng cách ước lượng) từ ô hiện tại đến mục tiêu, thường được tính toán bằng một hàm heuristic.

**A\*** sử dụng hàm  $f$  để xác định lộ trình tối ưu, di chuyển qua các ô có giá trị  $f$  thấp nhất, từ đó tìm đường nhanh chóng và chính xác.

#### **Ưu điểm:**

**Tìm đường nhanh chóng và hiệu quả:** A\* là một trong những giải thuật tìm đường nhanh và chính xác, bởi vì nó kết hợp cả yếu tố thực tế (chi phí thực tế) và ước lượng (chi phí dự đoán), giúp nó tìm đường đi tối ưu mà không phải tính toán qua tất cả các ô có thể.

**Tối ưu cho các mê cung lớn và phức tạp:** Giải thuật này đặc biệt hiệu quả với các mê cung có kích thước lớn hoặc phức tạp, nơi các giải thuật khác như Flood-Fill có thể không thực tế do yêu cầu bộ nhớ và thời gian tính toán lớn.

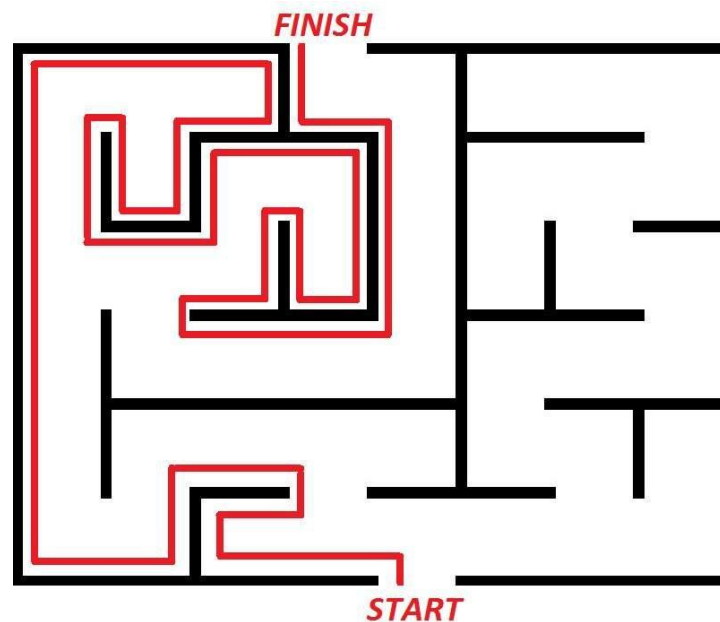
#### **Nhược điểm:**

**Yêu cầu tài nguyên xử lý cao:** A\* cần phải tính toán và lưu trữ giá trị chi phí cho nhiều ô, điều này có thể yêu cầu một lượng bộ nhớ và khả năng xử lý lớn, đặc biệt trong các mê cung phức tạp.

**Hiệu quả của giải thuật phụ thuộc vào hàm heuristic:** A\* phụ thuộc rất nhiều vào hàm heuristic (ước lượng chi phí từ ô hiện tại đến mục tiêu). Nếu hàm này không chính xác, hoặc quá phức tạp, giải thuật có thể trở nên kém hiệu quả hoặc không tìm được đường đi tối ưu.

Giải thuật A\* là một trong những giải thuật tối ưu và nhanh chóng nhất trong các bài toán tìm đường, đặc biệt phù hợp với những mê cung phức tạp hoặc lớn. Tuy nhiên, nhược điểm của giải thuật này là yêu cầu tài nguyên tính toán khá cao và hiệu quả sẽ phụ thuộc vào chất lượng của hàm heuristic.

### 3.1.3. Giải thuật Wall-Following



Hình 12. Mô tả thuật toán Wall-Following

Giải thuật Wall-Following là một phương pháp đơn giản để di chuyển trong mê cung. Robot sẽ di chuyển dọc theo một bức tường, giữ khoảng cách cố định với tường bên trái hoặc bên phải. Robot tiếp tục di chuyển theo tường cho đến khi tìm thấy lối ra khỏi mê cung. Phương pháp này chủ yếu dựa vào cảm biến để duy trì khoảng cách với tường và điều chỉnh hướng di chuyển theo đó.

#### **Ưu điểm:**

**Dễ triển khai và không cần nhiều bộ nhớ:** Giải thuật Wall-Following rất đơn giản để triển khai và không yêu cầu nhiều bộ nhớ, bởi vì nó chỉ sử dụng các cảm biến để theo dõi tường và không cần phải tính toán bản đồ mê cung.

**Hiệu quả trong các mê cung không phức tạp:** Trong các mê cung có cấu trúc đơn giản hoặc có ít ngã rẽ, giải thuật này có thể tìm được đường ra một cách hiệu quả mà không cần tính toán phức tạp.

#### **Nhược điểm:**

**Không đảm bảo tìm được đường đi ngắn nhất:** Mặc dù giải thuật này giúp robot di chuyển ra khỏi mê cung, nhưng nó không đảm bảo tìm được đường đi ngắn nhất. Robot có thể đi theo các đoạn đường dài hơn, vì giải thuật không tính đến tối ưu hóa lộ trình.

**Có thể bị kẹt trong vòng lặp nếu mê cung có cấu trúc đặc biệt:** Nếu mê cung có cấu trúc vòng lặp hoặc các bức tường quấn quanh, robot có thể bị kẹt và không thể thoát

ra khỏi mê cung, do giải thuật chỉ dựa vào việc theo dõi một bức tường mà không xét đến các yếu tố khác trong mê cung.

### **3.2. Lựa chọn giải thuật cho Micromouse**

Dựa trên các yêu cầu thực tế của đề tài và khả năng của phần cứng, **Wall-Following** được chọn làm giải thuật chính cho robot Micromouse. Quyết định này dựa trên các yếu tố sau:

#### **Khả năng triển khai:**

Với các cảm biến hồng ngoại đơn giản và vi điều khiển như Arduino Nano, Wall-Following dễ dàng được tích hợp và hoạt động hiệu quả. Giải thuật này không yêu cầu thuật toán phức tạp, giúp giảm bớt khối lượng công việc lập trình và thử nghiệm.

#### **Tài nguyên phần cứng thấp:**

Wall-Following không yêu cầu lưu trữ toàn bộ bản đồ mê cung hoặc tính toán chi phí cho các lộ trình, điều này phù hợp với các vi điều khiển có tài nguyên hạn chế và các thiết bị chi phí thấp. Điều này cũng giúp tiết kiệm thời gian và năng lượng khi triển khai.

#### **Thích hợp cho mê cung không phức tạp:**

Trong các mê cung có cấu trúc đơn giản và ít vòng lặp, Wall-Following có thể hoàn thành nhiệm vụ tìm lối ra một cách hiệu quả mà không cần phải tối ưu hóa đường đi. Đây là lựa chọn tối ưu khi mục tiêu chính là hoàn thành nhiệm vụ cơ bản.

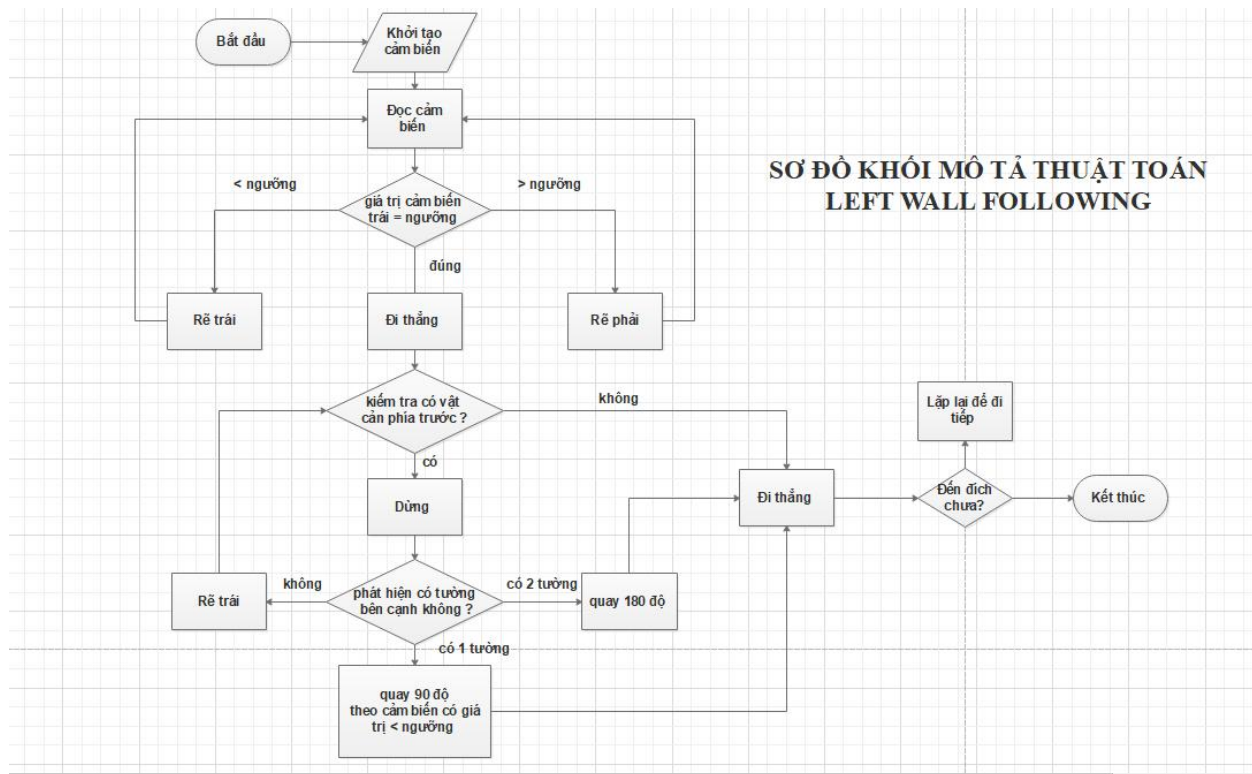
#### **Thời gian nghiên cứu và triển khai hạn chế:**

Do thời gian để tìm hiểu và triển khai các thuật toán phức tạp như Flood-Fill hoặc A\* chưa đủ, Wall-Following là giải pháp khả thi nhất trong giai đoạn hiện tại.



### 3.3. Phân tích chi tiết giải thuật đã chọn

#### 3.3.1. Mô tả giải thuật



Hình 13. Sơ đồ khối mô tả thuật toán left wall following

Giải thuật **Wall-Following** hoạt động dựa trên việc bám theo một bức tường cố định (trái hoặc phải) để tìm đường trong mê cung. Trong đề tài này, robot Micromouse sử dụng **bám tường trái** làm chiến lược chính. Dưới đây là các bước cụ thể:

#### Đo thông số các cảm biến và điều chỉnh PID:

Robot sử dụng các cảm biến hồng ngoại để đo khoảng cách đến tường bên trái, tường chéo trái và phía trước.

Các giá trị trả về từ cảm biến được sử dụng trong thuật toán PID để điều chỉnh tốc độ của từng động cơ, đảm bảo robot di chuyển ổn định và không bị va chạm.

#### Xác định tường bên:

- Robot chọn tường bên trái làm hướng bám cố định.
- Mục tiêu là duy trì khoảng cách cố định giữa robot và tường bên trái, đồng thời phản ứng kịp thời với các thay đổi trong cấu trúc mê cung.

#### Kiểm tra cảm biến:

- Cảm biến bên trái: Xác định sự hiện diện của tường để quyết định tiếp tục bám theo.

- Cảm biến phía trước: Kiểm tra xem có vật cản trước mặt hay không.
- Cảm biến chéo trái: Hỗ trợ phát hiện góc tường hoặc khoảng cách trong các tình huống cần rẽ.

#### **Điều chỉnh hướng di chuyển kết hợp với PID:**

Robot thực hiện các hành động dựa trên trạng thái cảm biến như sau:

- Nếu có tường bên trái và không có vật cản phía trước: Tiếp tục di chuyển thẳng.
- Nếu không có tường bên trái: Rẽ trái để tiếp tục bám theo tường.
- Nếu có vật cản phía trước: Rẽ phải để tránh vật cản và duy trì lộ trình.

#### **Lặp lại quy trình:**

Sau mỗi lần di chuyển, robot lặp lại quy trình đo cảm biến, điều chỉnh PID, và xác định hướng di chuyển.

Quy trình này tiếp tục cho đến khi robot tìm thấy lối ra hoặc hoàn thành nhiệm vụ trong mê cung.

### **3.3.2. Cách thức hoạt động trong các tình huống cụ thể**

#### **Tình huống không có tường:**

- Khi cảm biến bên trái không phát hiện tường, robot sẽ ưu tiên di chuyển thẳng.
- Trong quá trình này, robot liên tục quét các cảm biến để tìm kiếm tường bên trái nhằm duy trì chiến lược bám tường.

#### **Tình huống góc vuông:**

- Khi cảm biến phát hiện góc vuông (ví dụ: tường phía trước hoặc tường bên trái đột ngột kết thúc), robot sẽ kiểm tra các tín hiệu từ cảm biến để xác định hướng rẽ.
- Nếu không có tường phía trước và tường bên trái, robot sẽ rẽ trái để tiếp tục bám tường.
- Nếu có vật cản phía trước, robot sẽ rẽ phải, sau đó tiếp tục di chuyển để tìm tường bên trái.

#### **Tình huống vòng lặp:**

Trong các mê cung có cấu trúc đặc biệt, robot có thể di chuyển theo chu kỳ lặp lại (ví dụ: các vòng lặp không có lối thoát rõ ràng).

Để xử lý, cần tích hợp cơ chế phát hiện vòng lặp như:

- Sử dụng bộ đếm bước để theo dõi số lần di chuyển qua cùng một vị trí.
- Kết hợp với các thuật toán đánh dấu vị trí đã đi qua nhằm tránh lặp lại đường đi.

### **3.3.3. Tối ưu hóa giải thuật**

#### **Cải thiện cảm biến:**

Nâng cấp hoặc hiệu chỉnh các cảm biến hồng ngoại để đảm bảo độ chính xác cao hơn trong việc phát hiện tường và khoảng cách.

Giảm nhiễu tín hiệu cảm biến bằng cách cải thiện phần mềm xử lý dữ liệu, tránh các sai số do ánh sáng môi trường hoặc bề mặt không đồng đều.

#### **Giảm sai số di chuyển:**

Sử dụng bộ điều khiển PID để điều chỉnh tốc độ và hướng di chuyển của động cơ, giúp robot duy trì quỹ đạo ổn định, đặc biệt khi rẽ hoặc chuyển hướng.

Kiểm tra và tối ưu hóa các thông số PID để đạt hiệu suất tốt nhất trong từng tình huống cụ thể.

#### **Phát hiện vòng lặp:**

**Bộ đếm bước:** Theo dõi số bước di chuyển của robot trong mỗi chu kỳ và so sánh với giá trị giới hạn để phát hiện vòng lặp.

**Thuật toán đánh dấu vị trí:** Lưu trữ thông tin các vị trí đã đi qua trong mê cung (ví dụ: gán các ô đã đi qua một giá trị đặc biệt), từ đó nhận diện và tránh lặp lại lộ trình.

## CHƯƠNG 4: THIẾT KẾ VÀ TRIỂN KHAI

### 4.1. Thiết kế phần cứng

#### 4.1.1. Mạch điều khiển và cảm biến

##### **Vi điều khiển:**

**Arduino Nano** được lựa chọn vì kích thước nhỏ gọn, dễ dàng tích hợp trên PCB và có khả năng lập trình mạnh mẽ.

Với các chân I/O đa dạng, Arduino Nano có thể kết nối với nhiều cảm biến và động cơ encoder cùng lúc.

Vi điều khiển này hỗ trợ xử lý nhanh các tín hiệu từ cảm biến, tính toán thuật toán PID, và điều khiển chính xác động cơ.

##### **Các đặc điểm nổi bật:**

Hỗ trợ truyền thông I2C, UART, và SPI, thích hợp cho các cảm biến và giao tiếp với driver động cơ.

Bộ nhớ đủ lớn để triển khai các thuật toán như Wall-Following.

Tiêu thụ năng lượng thấp, phù hợp với hệ thống chạy bằng pin.

##### **Tích hợp với cảm biến:**

Arduino Nano nhận tín hiệu analog từ các cảm biến hồng ngoại IR để xác định khoảng cách đến tường và các vật cản.

Kết hợp các chân PWM để điều khiển tốc độ và hướng của động cơ thông qua mạch driver TB6612.

##### **Lý do sử dụng Arduino Nano:**

Tính sẵn có và phổ biến trên thị trường, dễ dàng thay thế hoặc sửa chữa.

Nền tảng mã nguồn mở, hỗ trợ nhiều thư viện hữu ích để xử lý tín hiệu cảm biến và điều khiển động cơ.

Giá thành thấp, tối ưu chi phí sản xuất của robot.

##### **Cảm biến:**

##### **Cảm biến tường trên UKMARSBOT:**

Đối với cảm biến tường, có hai lựa chọn chính: sử dụng ánh sáng hồng ngoại (IR) hoặc ánh sáng nhìn thấy (visible light). Hai lựa chọn này được quyết định dựa trên điều kiện ứng dụng và mức độ an toàn trong quá trình sử dụng.

Phân tích và lựa chọn cảm biến:

1. Lựa chọn IR (ánh sáng hồng ngoại):

- Ưu điểm: Hồng ngoại có khả năng phát hiện khoảng cách chính xác trong môi trường thiếu sáng, rất phù hợp với điều kiện hoạt động của robot trong mê cung.
- Nhược điểm: Vì ánh sáng IR không kích hoạt phản xạ nháy mắt tự nhiên của con người, nên việc sử dụng trong các môi trường mở (như trường học) có thể gây rủi ro nếu ánh sáng chiếu trực tiếp vào mắt.

2. Lựa chọn ánh sáng nhìn thấy:

- Ưu điểm: An toàn hơn cho người sử dụng và môi trường xung quanh. Đồng thời, giúp kiểm soát tốt hơn trong các ứng dụng cần giám sát trực quan.
- Nhược điểm: Hoạt động kém hiệu quả trong môi trường ánh sáng mạnh hoặc nhiều sáng.

Trong đồ án này, nhóm lựa chọn cảm biến ánh sáng nhìn thấy nhằm đảm bảo an toàn và phù hợp với yêu cầu của dự án sử dụng trong môi trường học thuật.

**Cấu hình cảm biến:**

Sử dụng 4 cảm biến LED IR để phát hiện khoảng cách và định hướng:

- 2 cảm biến bên trái:

Một cảm biến đo khoảng cách thẳng, giúp xác định độ gần của tường bên trái.

Một cảm biến đo góc chéo trái, hỗ trợ phát hiện góc tường hoặc các cấu trúc phức tạp.

- 2 cảm biến bên phải:

Một cảm biến đo khoảng cách thẳng, tương tự với phía trái, đảm bảo cân đối trong di chuyển.

Một cảm biến đo góc chéo phải, nhận diện các góc hoặc rẽ bên phải.

**Mục đích sử dụng:**

Nhận diện khoảng cách: Các cảm biến trả về giá trị analog phản ánh khoảng cách từ robot đến tường ở bên trái và phải. Giá trị cao tương ứng với tường gần và giá trị thấp cho thấy tường xa.

Duy trì định hướng: Cảm biến giúp robot bám sát tường theo thuật toán Wall-Following, giữ khoảng cách ổn định để di chuyển thẳng hoặc rẽ khi cần thiết.

Phát hiện chướng ngại vật: Đảm bảo robot phản ứng nhanh khi gặp tường hoặc vật cản phía trước.

**Tính năng tích hợp:**

- Các cảm biến được kết nối với chân analog của Arduino Nano để thu thập dữ liệu liên tục.
- Các cảm biến IR được điều chỉnh góc hợp lý để tối ưu khả năng phát hiện tường.

**Ưu điểm của hệ thống cảm biến này:**

- Đơn giản và hiệu quả: Phù hợp với thiết kế nhỏ gọn của robot UKMARSbot.
- Đáp ứng nhanh: Giá trị cảm biến được đọc và xử lý kịp thời để robot đưa ra quyết định chính xác.
- Chi phí thấp: Các cảm biến IR có giá thành hợp lý, dễ dàng triển khai trong sản xuất.

**Mạch điều khiển (UKMARSBOT):**

Mạch điều khiển UKMARSBOT là một thiết kế tích hợp, bao gồm các thành phần chính như vi điều khiển Arduino Nano, driver động cơ, và các cảm biến. Thiết kế này hướng đến việc tối ưu hóa không gian và đảm bảo sự gọn nhẹ, thuận tiện trong quá trình lắp ráp. Với các tiêu chuẩn được tối ưu hóa, mạch điều khiển không chỉ đáp ứng yêu cầu vận hành ổn định mà còn tạo điều kiện dễ dàng khi bảo trì và nâng cấp hệ thống.

Nguồn năng lượng chính của robot được cung cấp bởi pin sạc 9V, đảm bảo cung cấp dòng điện ổn định để vận hành toàn bộ hệ thống.

**4.1.2. Động cơ và bộ mã hóa vòng quay (Encoder)**

**Động cơ GA12-N20 Encoder:**

Trong đồ án này, nhóm đã lựa chọn động cơ GA12-N20 tích hợp encoder làm giải pháp chính cho việc điều khiển chuyển động của robot UKMARSBOT. Loại động cơ này được chọn nhờ các đặc điểm sau:

- **Tích hợp Encoder:** Encoder đi kèm với động cơ giúp đo lường chính xác số vòng quay, từ đó xác định được tốc độ và quãng đường di chuyển.
- **Tỷ số truyền:** Mặc dù tỷ số truyền **1:30** và **1:50** là lý tưởng để cân bằng giữa tốc độ và lực kéo, nhưng nhóm không tìm được các động cơ với hai tỷ số truyền này trên thị trường trong thời gian thực hiện đồ án. Do đó, nhóm đã sử dụng tỷ số truyền **1:100** như một lựa chọn thay thế.
  - **Ưu điểm của 1:100:** Tỷ số này tăng cường lực kéo, giúp robot di chuyển ổn định và chính xác, đặc biệt phù hợp khi robot cần kiểm soát tốc độ trong các quãng đường ngắn hoặc cần phản ứng linh hoạt trong không gian hạn chế như mê cung.

- **Nhược điểm của 1:100:** Robot có tốc độ thấp hơn so với các tỷ số nhỏ hơn, nhưng điều này không gây ảnh hưởng lớn đến hiệu quả tổng thể vì tính chính xác được ưu tiên trong dự án.

### **Encoder:**

Encoder tích hợp trong động cơ đóng vai trò quan trọng trong việc điều khiển và định hướng di chuyển của robot. Những tính năng chính bao gồm:

**Tín hiệu đầu ra:** Encoder cung cấp tín hiệu xung vuông phản ánh số vòng quay của động cơ, dữ liệu này được vi điều khiển xử lý để tính toán tốc độ và quãng đường.

**Ứng dụng trong thuật toán PID:** Tín hiệu từ encoder được sử dụng trong thuật toán điều khiển PID nhằm:

Duy trì tốc độ động cơ ổn định.

Điều chỉnh quỹ đạo di chuyển của robot, đảm bảo robot giữ đúng hướng và đạt mục tiêu.

**Xác định khoảng cách và điều chỉnh hướng:** Encoder giúp đo lường chính xác khoảng cách robot đã di chuyển và hiệu chỉnh độ lệch của hai bánh xe, từ đó cải thiện hiệu quả điều hướng.

### **4.1.3. Khung sườn và bố trí linh kiện**

**Arduino Nano:** Arduino Nano được gắn cố định trên mạch PCB của **UKMARSBOT**, làm trung tâm điều khiển robot. Arduino Nano đóng vai trò quan trọng trong việc xử lý tín hiệu từ các cảm biến, động cơ và encoder, điều khiển các hoạt động của robot.

**Trọng tâm và sự ổn định:** Các linh kiện khác như động cơ encoder, cảm biến IR, và pin được bố trí một cách hợp lý trên khung sườn để đảm bảo trọng tâm của robot được giữ ở mức thấp và cân bằng. Các linh kiện được gắn ở trung tâm khung sườn giúp giảm nguy cơ lật và tăng khả năng kiểm soát khi robot di chuyển hoặc thay đổi hướng.

**Động cơ encoder:** Hai động cơ encoder được gắn ở hai bên của robot, giúp tạo ra sự cân bằng lực giữa hai bánh xe và đảm bảo rằng robot có thể di chuyển chính xác theo yêu cầu.

**Pin:** Pin được đặt ở vị trí dưới cùng của khung sườn. Điều này giúp giảm thiểu nguy cơ lật khi robot tăng tốc hoặc quay đầu, vì trọng lượng của pin giúp ổn định khung sườn và giữ robot gần mặt đất.

**Cảm biến IR:** Cảm biến IR được bố trí đối xứng hai bên của robot. Việc bố trí này đảm bảo rằng robot có thể đo khoảng cách chính xác đến các bức tường hoặc vật cản trong môi trường mê cung, từ đó điều chỉnh hướng đi một cách chính xác.

### **Cấu hình linh kiện trên mạch cảm biến**

Để đảm bảo hiệu suất và độ chính xác cao, các linh kiện được sử dụng và cấu hình như sau:

- **Điện trở sạc emitter (R8):**  $33\Omega$ .

- **Điện trở giới hạn dòng (R9, R10, R11):**

Hồng ngoại:  $33\Omega$ .

Ánh sáng nhìn thấy:  $10\Omega$ .

- **Điện trở tải detector (R3, R4, R5):**

Sử dụng phototransistor BPW96B:  $1.8k\Omega$  (IR) hoặc  $1k\Omega$  (ánh sáng nhìn thấy).

- **Điện trở thiên áp (R6, R7):**

R6:  $390\Omega$ .

R7:  $10k\Omega$ .

- **Tụ điện (C1):**  $100\mu F$ , đảm bảo kết nối đúng chiều để tránh sự cố ngắn mạch.

- **LED chỉ báo (LED1, LED2):** Gắn cách PCB vài mm để tạo không gian lắp ráp.

- **Lựa chọn và lắp ráp cảm biến tường**

- **LED phát sáng:**

IR: SFH4550.

Ánh sáng nhìn thấy: TLCR5800.

- **Phototransistor:**

Sử dụng BPW96B hoặc BPW85C cho cả hai phiên bản.

LED phát sáng được lắp trên bề mặt trên của mạch, trong khi phototransistor được lắp ở mặt dưới, đảm bảo phát hiện chính xác.

- **Kết nối và cố định cảm biến:**

Các cảm biến được kết nối với mạch chính thông qua dây đa lõi hoặc đầu nối Dupont, đảm bảo dễ dàng tháo lắp và sửa chữa.

PCB cảm biến được cố định vào mạch chính bằng ốc M3 và spacer 10mm.

- **Quy trình lắp ráp mạch điều khiển**



Quy trình lắp ráp được thực hiện theo thứ tự từ các linh kiện thấp đến cao để đảm bảo dễ dàng và chính xác. Các thành phần như điện trở, diode, tụ điện, công tắc, và motor driver được gắn trước, sau đó đến vi điều khiển Arduino Nano và các đầu kết nối.

## 4.2. Thiết kế phần mềm

### 4.2.1. Cấu trúc chương trình

Phần mềm của **UKMARSBOT** được thiết kế theo cấu trúc module để dễ dàng quản lý và bảo trì. Mỗi module đảm nhận một chức năng cụ thể, giúp phân tách các phần việc và tối ưu hóa hiệu suất hoạt động của robot. Cấu trúc chương trình bao gồm các module chính sau:

1. **Khởi tạo phần cứng:** Module này chịu trách nhiệm thiết lập và cấu hình tất cả các phần cứng cần thiết cho hoạt động của robot. Cụ thể, Arduino Nano được khởi tạo để làm trung tâm điều khiển, các cảm biến IR được cấu hình để thu thập dữ liệu về khoảng cách, và các động cơ encoder được thiết lập để điều khiển chuyển động của robot. Module này giúp chuẩn bị cho robot hoạt động ổn định ngay từ khi bắt đầu.
2. **Đọc cảm biến:** Module này thực hiện việc thu thập dữ liệu từ các cảm biến IR và encoder. Cảm biến IR giúp đo khoảng cách giữa robot và các bức tường trong mê cung, trong khi encoder cung cấp thông tin về tốc độ và quãng đường di chuyển của robot. Dữ liệu từ các cảm biến này sẽ được gửi về Arduino Nano để xử lý tiếp.
3. **Xử lý dữ liệu:** Dữ liệu thu thập từ các cảm biến sẽ được xử lý trong module này. Cụ thể, khoảng cách đến các bức tường sẽ được phân tích để robot có thể điều chỉnh hướng đi một cách phù hợp. Giải thuật **Wall-Following** (theo dõi tường) được áp dụng ở bước này, giúp robot di chuyển dọc theo các bức tường của mê cung mà không va phải chúng. Module này sẽ tính toán và đưa ra các tín hiệu điều khiển phù hợp để robot duy trì hành trình chính xác.
4. **Điều khiển động cơ:** Dựa trên tín hiệu từ cảm biến và encoder, module điều khiển động cơ sẽ điều chỉnh tốc độ và hướng quay của động cơ. Cụ thể, nếu robot cần phải rẽ trái, module này sẽ giảm tốc độ của động cơ bên phải và tăng tốc độ động cơ bên trái. Ngược lại, nếu robot cần phải rẽ phải, tốc độ của động cơ trái sẽ giảm, trong khi động cơ phải tăng tốc. Điều này giúp robot duy trì hướng đi ổn định trong mê cung.

## 4.2.2. Các module chính

### 1. Module cảm biến IR:

- **Chức năng:** Đọc tín hiệu từ các cảm biến IR (hồng ngoại) để phát hiện vật cản hoặc tường xung quanh robot.
- **Chi tiết:**
  - Cảm biến IR sẽ phát ra tia hồng ngoại và nhận lại tín hiệu phản xạ từ vật cản, giúp robot xác định được vị trí của các vật thể ở gần.
  - Sử dụng bộ lọc trung bình để giảm nhiễu tín hiệu từ môi trường, đảm bảo tín hiệu từ cảm biến được chính xác hơn và giúp robot hoạt động ổn định hơn trong các điều kiện ánh sáng thay đổi.

### 2. Module điều hướng:

- **Chức năng:** Thực hiện thuật toán **Wall-Following** để điều khiển robot di chuyển trong mê cung hoặc khu vực có tường, giữ cho robot luôn ở gần tường bên trái hoặc bên phải tùy theo thiết kế của robot.
- **Chi tiết:**
  - **Wall-Following** là thuật toán cơ bản, trong đó robot sẽ sử dụng các cảm biến để phát hiện tường và điều chỉnh hướng đi sao cho khoảng cách từ robot đến tường luôn ổn định.
  - Tính toán khoảng cách giữa robot và tường, và điều chỉnh động cơ để đảm bảo rằng robot không bị lệch hướng quá nhiều và có thể di chuyển ổn định trong mê cung.

### 3. Module điều khiển động cơ:

- **Chức năng:** Điều khiển động cơ GA12-N20 để đảm bảo robot di chuyển chính xác và đạt hiệu suất tối ưu.
- **Chi tiết:**
  - Sử dụng thuật toán **PID** (Proportional-Integral-Derivative) để điều chỉnh tốc độ và hướng của động cơ. PID là một thuật toán điều khiển phản hồi, giúp điều chỉnh các sai lệch trong quá trình di chuyển của robot.
  - Cấu trúc PID giúp robot có thể di chuyển mượt mà hơn, giảm thiểu sai số về tốc độ và hướng, đặc biệt khi gặp các tình huống phức tạp như rẽ hoặc thay đổi hướng.

### 4. Module xử lý encoder:

- **Chức năng:** Đếm xung từ encoder gắn trên động cơ để tính toán quãng đường di chuyển của robot.

- **Chi tiết:**

- Encoder là thiết bị đo chuyển động giúp robot xác định được vị trí hiện tại và tính toán được quãng đường đã đi.
- Việc sử dụng encoder giúp robot có khả năng biết được khi nào cần quay đầu hoặc di chuyển thẳng, đồng thời giúp theo dõi chính xác hành trình của robot trong môi trường.
- Xử lý xung từ encoder giúp tính toán số bước di chuyển và điều chỉnh hành vi của robot cho phù hợp với các lệnh điều khiển đã được lập trình.

#### **4.3. Kết nối và tích hợp**

##### **Kết nối phần cứng:**

- **Gắn Arduino Nano vào mạch UKMARSBOT:** Arduino Nano được gắn chắc chắn vào mạch UKMARSBOT, đóng vai trò là trung tâm điều khiển của hệ thống. Bộ vi xử lý này đảm nhận vai trò điều khiển các tín hiệu từ cảm biến và động cơ, đồng thời điều phối các thao tác của robot trong quá trình di chuyển.
- **Nối cảm biến IR vào các chân analog của Arduino Nano:** Các cảm biến IR được kết nối vào các chân analog của Arduino Nano để thu thập dữ liệu về khoảng cách. Cảm biến IR sẽ đo khoảng cách từ robot đến các bức tường trong mê cung, và tín hiệu này sẽ được Arduino xử lý để đưa ra các quyết định về hướng đi của robot.
- **Kết nối động cơ GA12-N20 và encoder vào driver tích hợp trên mạch:** Động cơ GA12-N20 với encoder được nối với driver tích hợp trên mạch UKMASBOT. Driver này sẽ nhận tín hiệu điều khiển từ Arduino để điều chỉnh tốc độ và hướng quay của động cơ. Đồng thời, tín hiệu từ encoder sẽ cung cấp dữ liệu chính xác về tốc độ và quãng đường di chuyển, giúp robot hoạt động hiệu quả hơn.

##### **Tích hợp phần mềm:**

- **Lập trình và nạp chương trình vào Arduino Nano thông qua IDE Arduino:** Sau khi kết nối phần cứng, các chương trình điều khiển sẽ được lập trình và nạp vào Arduino Nano thông qua môi trường phát triển IDE Arduino. Các chương trình này bao gồm các thuật toán điều khiển, xử lý dữ liệu từ cảm biến IR và encoder, và điều khiển động cơ theo các quyết định về hướng đi của robot.
- **Kiểm tra hoạt động từng module trước khi chạy thử nghiệm toàn hệ thống:** Trước khi thực hiện thử nghiệm toàn bộ hệ thống, mỗi module sẽ được kiểm tra

riêng biệt. Các cảm biến IR, encoder và động cơ sẽ được kiểm tra để đảm bảo chúng hoạt động chính xác và đồng bộ với nhau. Sau khi hoàn thành các bước kiểm tra, robot sẽ được thử nghiệm trong môi trường mô phỏng để đảm bảo các tính năng hoạt động đúng như mong đợi.

#### 4.4. Kiểm thử và chỉnh sửa

##### Kiểm thử:

- **Robot được kiểm tra trên các mê cung tiêu chuẩn:** Robot **UKMASBOT** được kiểm tra trong các môi trường mê cung tiêu chuẩn để đánh giá khả năng di chuyển và điều hướng của hệ thống. Các mê cung này bao gồm các đoạn đường thẳng, góc cua, ngõ cụt và ngã ba. Việc kiểm tra trên các tình huống khác nhau giúp đảm bảo rằng robot có thể xử lý được mọi tình huống trong môi trường thực tế.
- **Ghi nhận dữ liệu từ cảm biến IR và encoder để phân tích hiệu suất:** Trong quá trình kiểm thử, dữ liệu thu được từ các cảm biến **IR** và **encoder** sẽ được ghi nhận và phân tích. Các thông số như khoảng cách đến tường, tốc độ di chuyển và độ chính xác của robot trong việc bám tường và quay cua sẽ được xem xét kỹ lưỡng. Việc này giúp đánh giá hiệu suất hoạt động của robot và xác định những điểm cần cải thiện.

##### Chỉnh sửa:

- **Tối ưu hóa thông số PID để cải thiện khả năng bám tường:** Sau khi phân tích kết quả kiểm thử, các thông số trong thuật toán **PID** (Proportional-Integral-Derivative) sẽ được tối ưu hóa để cải thiện khả năng bám tường của robot. Việc này giúp robot duy trì khoảng cách ổn định và di chuyển mượt mà hơn khi tiếp cận các bức tường trong mê cung.
- **Điều chỉnh thuật toán điều hướng khi phát hiện lỗi trong các tình huống phức tạp:** Trong quá trình kiểm thử, nếu robot gặp phải những tình huống phức tạp như góc cua hẹp, ngõ cụt hoặc ngã ba mà không thể xử lý đúng, thuật toán điều hướng sẽ được điều chỉnh. Các lỗi như robot bị lạc hướng hoặc không thể quay đúng sẽ được xác định và khắc phục thông qua việc thay đổi chiến lược điều hướng, chẳng hạn như cải tiến phương pháp theo dõi đường đi hoặc phát hiện các tín hiệu từ cảm biến chính xác hơn.

# CHƯƠNG 5: KẾT QUẢ THỰC NGHIỆM

## 5.1. Quá trình chạy thử nghiệm



Hình 14. Hình ảnh thực tế của mê cung

**Môi trường thử nghiệm:** Robot UKMASBOT được kiểm tra trên một mô hình mê cung tiêu chuẩn với kích thước mỗi ô là **18x18 cm**. Mê cung bao gồm các đoạn đường thẳng, góc cua, ngã ba và ngõ cụt, tạo ra một môi trường thử nghiệm đa dạng để kiểm tra khả năng điều hướng của robot. Môi trường này giúp đánh giá hiệu suất của robot trong những tình huống thực tế có thể gặp phải trong mê cung.

### Quy trình thử nghiệm:

#### 1. Kiểm tra cơ bản:

Trước khi tiến hành thử nghiệm toàn hệ thống, robot đã được kiểm tra cơ bản bằng cách chạy thử từng module riêng lẻ. Cụ thể:

- **Động cơ:** Kiểm tra sự hoạt động của động cơ **GA12-N20** và khả năng điều khiển hướng đi của robot.

- **Cảm biến IR:** Kiểm tra độ chính xác của cảm biến **IR** trong việc phát hiện tường và đo khoảng cách.
- **Encoder:** Kiểm tra tính chính xác của encoder trong việc đo lường quãng đường và tốc độ di chuyển của robot.

Việc kiểm tra từng module giúp đảm bảo rằng các linh kiện hoạt động ổn định trước khi đưa vào thử nghiệm toàn bộ hệ thống.

## 2. Thử nghiệm toàn hệ thống:

Sau khi kiểm tra cơ bản, robot được đưa vào thử nghiệm toàn bộ hệ thống với quy trình như sau:

- **Bắt đầu từ một góc của mê cung:** Robot được đặt tại điểm xuất phát của mê cung và bắt đầu di chuyển.
- **Áp dụng thuật toán Wall-Following:** Robot sử dụng thuật toán **Wall-Following** để điều hướng qua các đoạn đường, góc cua và ngõ cụt. Robot sẽ điều chỉnh hướng đi của mình dựa trên tín hiệu nhận được từ cảm biến IR.
- **Ghi nhận dữ liệu:** Trong suốt quá trình di chuyển, dữ liệu từ cảm biến IR và encoder được ghi lại để phân tích sau thử nghiệm, giúp đánh giá độ chính xác và hiệu quả của quá trình điều hướng.

## 3. Chạy thử lặp lại:

Để đánh giá độ tin cậy của hệ thống, các thử nghiệm được lặp lại với các cấu hình mê cung khác nhau, bao gồm mê cung có nhiều ngã ba, góc cua hẹp và ngõ cụt. Mỗi thử nghiệm được thực hiện nhiều lần để kiểm tra tính ổn định và độ chính xác của robot trong việc duy trì đường đi và vượt qua các thử thách trong mê cung. Kết quả thu được từ các lần thử nghiệm sẽ giúp đánh giá khả năng điều chỉnh và hoàn thiện thuật toán điều hướng.

## 5.2. Phân tích hiệu suất di chuyển

### Tốc độ di chuyển:

- **Tốc độ trung bình:** Trong suốt quá trình thử nghiệm, tốc độ trung bình của robot là **5 cm/s**, cho thấy robot có thể di chuyển một cách ổn định và không gặp quá nhiều vấn đề trong môi trường mê cung.

- **Tốc độ tối đa:** Khi di chuyển trên các đoạn đường thẳng, robot có thể đạt được **tốc độ tối đa là 8 cm/s**. Tuy nhiên, tốc độ này có thể giảm khi robot cần xử lý các tình huống phức tạp như góc cua hoặc ngõ cụt.

#### **Khả năng xử lý góc cua:**

- **Thời gian điều chỉnh tại góc cua:** Thời gian trung bình mà robot cần để điều chỉnh khi gặp góc cua là **1.2 giây**. Thời gian này được xem là hợp lý, tuy nhiên, vẫn có thể được tối ưu hóa thêm để giảm thiểu độ trễ và tăng tốc độ di chuyển.
- **Độ chính xác khi xử lý góc cua:** Robot có **độ chính xác lên đến 95%** khi xử lý góc cua, điều này cho thấy khả năng di chuyển của robot rất chính xác. Các lỗi xảy ra chủ yếu là do **hiệu chuẩn cảm biến chưa hoàn thiện**, và điều này có thể được cải thiện trong các lần thử nghiệm tiếp theo.

#### **Độ chính xác của cảm biến:**

- **Sai số cảm biến IR:** Cảm biến IR cho thấy **sai số khoảng  $\pm 2$  mm** khi đo khoảng cách trong điều kiện ánh sáng bình thường. Đây là một sai số nhỏ và có thể chấp nhận được trong môi trường thử nghiệm của mô hình.
- **Cảm biến chéo:** Việc sử dụng các cảm biến **chéo** (được bố trí đối xứng hai bên) giúp robot xử lý các tình huống phức tạp như góc ngã ba nhanh chóng hơn so với việc chỉ sử dụng cảm biến thẳng. Cảm biến chéo giúp robot có cái nhìn toàn diện về môi trường xung quanh và điều chỉnh chính xác hơn khi gặp các tình huống đột ngột.

#### **Tính ổn định:**

**Hoạt động ổn định:** Trong suốt quá trình thử nghiệm, robot đã thể hiện tính ổn định cao khi hoạt động liên tục trong khoảng **15 phút** mà không gặp phải bất kỳ sự cố phần cứng hoặc phần mềm nào. Điều này cho thấy thiết kế phần cứng và phần mềm của hệ thống đủ mạnh mẽ và đáng tin cậy để duy trì hiệu suất lâu dài, đặc biệt khi robot phải đối mặt với các tình huống phức tạp trong môi trường mô hình. Tính ổn định này là yếu tố quan trọng để đảm bảo robot có thể hoàn thành nhiệm vụ mà không bị gián đoạn, giúp giảm thiểu khả năng mất mát dữ liệu hoặc cần phải can thiệp

### **5.3. Nhận xét về kết quả đạt được**

#### **Ưu điểm:**

- **Di chuyển ổn định:** Robot di chuyển ổn định và tuân thủ giải thuật Wall-Following hiệu quả, giúp điều hướng robot qua các khu vực trong mê cung một cách chính xác.
- **Thích nghi tốt:** Robot có khả năng thích nghi tốt với các cấu hình mê cung khác nhau, bao gồm các ngã ba, ngõ cụt, và đường vòng. Điều này cho thấy khả năng xử lý các tình huống phức tạp của hệ thống.
- **Tích hợp phần cứng và phần mềm:** Việc tích hợp chặt chẽ giữa phần cứng và phần mềm giúp robot hoạt động trơn tru, không có hiện tượng giật lag hoặc mất tín hiệu từ cảm biến, đảm bảo sự ổn định trong quá trình vận hành.

#### **Hạn chế:**

- **Ảnh hưởng ánh sáng môi trường:** Khi ánh sáng môi trường thay đổi đột ngột (quá sáng hoặc quá tối), cảm biến IR gặp khó khăn trong việc xác định khoảng cách chính xác, làm ảnh hưởng đến khả năng điều hướng của robot.
- **Tốc độ di chuyển chậm:** Robot có tốc độ di chuyển khá chậm khi xử lý các khu vực phức tạp như ngã ba hoặc góc hẹp, điều này dẫn đến tổng thời gian hoàn thành mê cung chưa tối ưu. Việc cải thiện tốc độ trong những tình huống này sẽ giúp tăng hiệu quả hoàn thành nhiệm vụ.

### **5.4. Các tình huống gặp phải và cách khắc phục**

#### **Tình huống 1: Sai số cảm biến IR ở ngã ba**

- **Nguyên nhân:** Góc đo của cảm biến không đủ rộng để nhận diện toàn bộ tường tại ngã ba, dẫn đến sai lệch trong việc xác định hướng đi của robot.
- **Khắc phục:** Để khắc phục tình huống này, thuật toán kiểm tra kết hợp giữa dữ liệu cảm biến IR thẳng và cảm biến IR chéo sẽ được sử dụng. Cảm biến chéo giúp robot nhận diện thêm thông tin về các bức tường ở góc ngã ba, từ đó giúp xác định vị trí chính xác hơn.

#### **Tình huống 2: Robot mất cân bằng khi tăng tốc**

- **Nguyên nhân:** Trọng tâm robot chưa được bố trí tối ưu, dẫn đến tình trạng mất ổn định khi robot tăng tốc, đặc biệt là khi quay góc hoặc di chuyển nhanh.
- **Khắc phục:** Để cải thiện tình trạng này, vị trí của pin và vi điều khiển sẽ được điều chỉnh để cải thiện trọng tâm của robot. Đồng thời, giảm tốc độ tối đa khi robot vào góc cua sẽ giúp robot duy trì sự ổn định trong suốt quá trình di chuyển.

#### **Tình huống 3: Encoder không đếm chính xác khi động cơ quay nhanh**



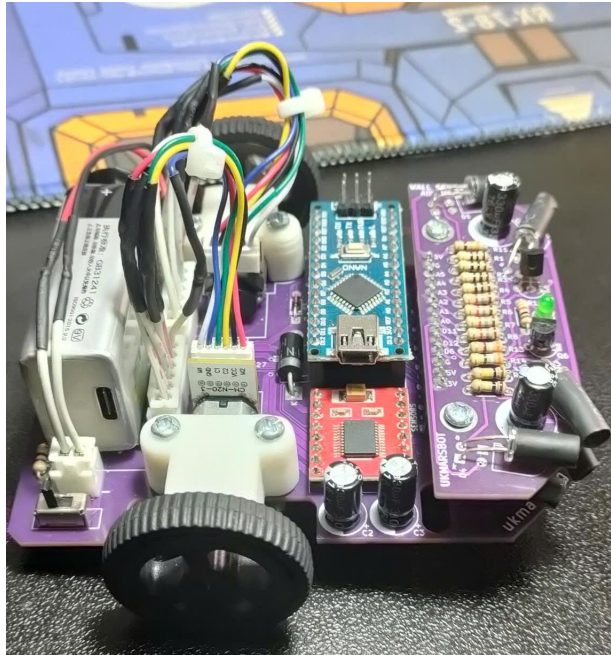
- **Nguyên nhân:** Xung từ encoder không được vi điều khiển đọc kịp khi động cơ quay nhanh, dẫn đến sai lệch trong việc đo quãng đường và tốc độ.
- **Khắc phục:** Để giải quyết vấn đề này, bộ lọc phần mềm sẽ được sử dụng để xử lý tín hiệu từ encoder, giúp loại bỏ nhiễu và đảm bảo rằng tín hiệu từ encoder được xử lý chính xác. Điều này giúp cải thiện độ chính xác trong việc điều khiển và theo dõi chuyển động của robot.

#### **Tình huống 4: Lỗi khi ánh sáng thay đổi**

- **Nguyên nhân:** Cảm biến IR bị ảnh hưởng bởi ánh sáng mạnh từ môi trường, làm giảm độ chính xác của việc đo khoảng cách.
- **Khắc phục:** Để khắc phục tình huống này, cảm biến IR sẽ được bọc bằng vật liệu chắn sáng để giảm thiểu tác động của ánh sáng môi trường. Ngoài ra, giá trị ngưỡng đo khoảng cách của cảm biến IR sẽ được hiệu chỉnh lại sao cho phù hợp với các điều kiện ánh sáng thay đổi trong quá trình vận hành của robot.

# CHƯƠNG 6: ĐÁNH GIÁ VÀ KẾT LUẬN

## 6.1. Đánh giá tổng quan về đồ án



*Hình 15. Mô hình hoàn thiện*

Đồ án về robot Micromouse sử dụng giải thuật Wall-Following đã hoàn thành các mục tiêu chính được đề ra từ ban đầu. Quá trình thực hiện từ thiết kế phần cứng, phát triển phần mềm đến thử nghiệm thực tế đều diễn ra khá suôn sẻ và đạt được kết quả khả quan. Robot Micromouse đã thể hiện khả năng di chuyển ổn định trong mê cung tiêu chuẩn, xử lý tốt các tình huống phức tạp như ngã ba, góc cua, và ngõ cụt, điều này chứng tỏ tính hiệu quả của giải thuật Wall-Following mà nhóm đã áp dụng.

Trong quá trình thực hiện, nhóm đã có cơ hội hiểu rõ hơn về thiết kế hệ thống nhúng, cách tích hợp các cảm biến vào mạch điều khiển, và kỹ năng lập trình điều khiển cho robot. Những trải nghiệm này không chỉ giúp nhóm hoàn thiện kỹ năng kỹ thuật mà còn mở ra hướng nghiên cứu và phát triển sâu hơn trong lĩnh vực robot tự hành, đặc biệt là trong các ứng dụng thực tế của robot trong các môi trường phức tạp.

Dự án đã giúp nhóm nâng cao khả năng phân tích, thiết kế và giải quyết các vấn đề kỹ thuật trong quá trình phát triển sản phẩm, đồng thời cải thiện kỹ năng làm việc nhóm, từ việc lên kế hoạch, chia công việc cho đến việc kiểm thử và tối ưu hóa hệ thống. Những kiến thức thu được trong quá trình làm đồ án sẽ là nền tảng vững chắc cho nhóm trong những nghiên cứu và ứng dụng robot tự động hóa trong tương lai.

## 6.2. Ưu điểm và hạn chế của giải pháp

### Ưu điểm:

- **Tính ổn định:** Robot hoạt động liên tục trong môi trường thử nghiệm mà không gặp sự cố phần cứng hoặc phần mềm, cho thấy sự ổn định cao của hệ thống.
- **Chi phí thấp:** Sử dụng các linh kiện phổ biến và giá rẻ như Arduino Nano, động cơ GA12-N20 tích hợp encoder, và cảm biến IR giúp giảm chi phí tổng thể của dự án mà vẫn đảm bảo được hiệu quả hoạt động của robot.
- **Giải thuật đơn giản và hiệu quả:** Giải thuật Wall-Following đơn giản, dễ triển khai và bảo trì, rất phù hợp với các mê cung cơ bản và không quá phức tạp. Điều này giúp nhóm tiết kiệm thời gian và công sức khi thực hiện.
- **Dễ dàng mở rộng:** Thiết kế phần mềm được xây dựng theo mô-đun, giúp dễ dàng mở rộng hoặc nâng cấp trong tương lai. Các linh kiện phần cứng cũng dễ dàng thay đổi hoặc thay thế, giúp nâng cấp hệ thống trong các ứng dụng phức tạp hơn.

### Hạn chế:

- **Hiệu suất chưa tối ưu:** Mặc dù robot có thể di chuyển ổn định, nhưng tốc độ di chuyển chưa đạt mức tối ưu, đặc biệt là trong các khu vực phức tạp như ngã ba hoặc góc hẹp, dẫn đến thời gian hoàn thành chưa nhanh như mong muốn.
- **Phụ thuộc vào cảm biến IR:** Robot dễ bị ảnh hưởng bởi sự thay đổi đột ngột của ánh sáng môi trường, ví dụ như quá sáng hoặc quá tối, gây ảnh hưởng đến độ chính xác của cảm biến IR trong việc đo khoảng cách và định hướng.
- **Giới hạn của giải thuật:** Giải thuật Wall-Following mặc dù rất hiệu quả với các mê cung cơ bản nhưng không thể xử lý tốt với những mê cung phức tạp hoặc có nhiều vòng lặp. Trong những tình huống này, giải thuật có thể gặp khó khăn trong việc điều hướng chính xác và tối ưu.

## 6.3. Đề xuất cải tiến trong tương lai

### Nâng cấp phần cứng:

- **Cảm biến lidar hoặc camera:** Việc sử dụng cảm biến lidar hoặc camera sẽ giúp robot nhận diện đường đi chính xác hơn và giảm phụ thuộc vào ánh sáng môi trường. Điều này sẽ cải thiện độ tin cậy của robot trong các điều kiện ánh sáng thay đổi, đặc biệt trong môi trường tối hoặc quá sáng.

- **Vi điều khiển mạnh hơn:** Việc chuyển sang sử dụng các vi điều khiển mạnh hơn như STM32 hoặc Raspberry Pi Pico sẽ giúp nâng cao tốc độ xử lý dữ liệu, giúp robot xử lý các tình huống phức tạp nhanh hơn và hiệu quả hơn.

#### **Cải tiến phần mềm:**

- **Thuật toán tìm đường nâng cao:** Tích hợp các thuật toán tìm đường nâng cao như Flood-Fill hoặc A\* có thể giúp cải thiện hiệu suất điều hướng trong các mê cung phức tạp. Các thuật toán này có thể giúp robot tìm ra lộ trình tối ưu và giảm thiểu các bước thừa trong quá trình di chuyển.
- **Hệ thống tự hiệu chỉnh cảm biến:** Việc xây dựng hệ thống tự hiệu chỉnh cảm biến theo điều kiện môi trường sẽ giúp robot duy trì hiệu suất ổn định trong các tình huống thay đổi ánh sáng hoặc môi trường xung quanh. Điều này sẽ giảm thiểu sự can thiệp của yếu tố môi trường đối với độ chính xác của cảm biến.

#### **Tối ưu hóa năng lượng:**

- **Sử dụng pin lithium polymer (LiPo):** Pin lithium polymer (LiPo) có trọng lượng nhẹ hơn và cung cấp năng lượng lâu hơn so với pin AA thông thường. Việc sử dụng pin LiPo sẽ giúp giảm trọng lượng tổng thể của robot, đồng thời kéo dài thời gian hoạt động của robot, giúp tăng hiệu suất sử dụng trong suốt quá trình thử nghiệm và thực hiện nhiệm vụ.

#### **Mở rộng ứng dụng:**

- **Phát triển robot cho các nhiệm vụ khác:** Sau khi tối ưu hóa hiệu suất, robot có thể được phát triển thêm để thực hiện các nhiệm vụ khác như vận chuyển hàng trong nhà kho hoặc khám phá môi trường thực tế. Việc mở rộng ứng dụng sẽ giúp tăng tính linh hoạt và khả năng áp dụng thực tế của robot trong nhiều lĩnh vực khác nhau.

### **6.4. Kết luận**

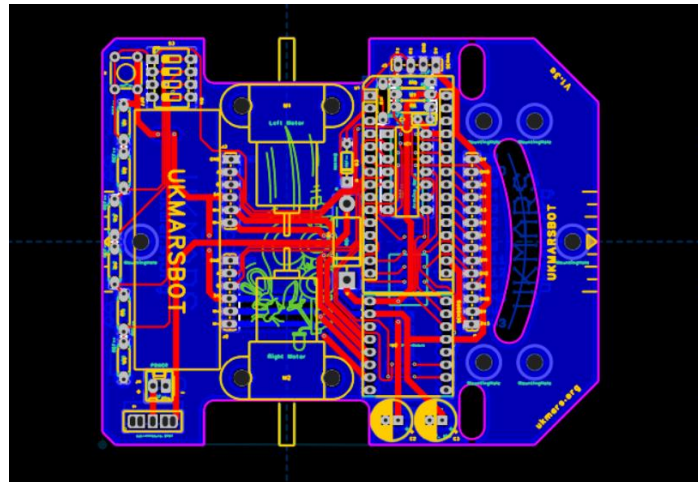
Dự án thiết kế và triển khai robot Micromouse đã đạt được những kết quả đáng khích lệ, từ thiết kế cơ bản đến thử nghiệm thực tế. Robot đã chứng minh khả năng hoạt động ổn định và hiệu quả trong mê cung tiêu chuẩn, thực hiện các nhiệm vụ điều hướng như bám tường và xử lý các tình huống như góc cua, ngã ba và ngõ cụt với độ chính xác cao. Mặc dù vẫn còn một số hạn chế như tốc độ di chuyển chưa tối ưu và sự phụ thuộc vào cảm biến IR, những vấn đề này có thể được cải thiện trong các phiên bản tiếp theo của robot.

Giải pháp được áp dụng trong dự án, với việc sử dụng Arduino Nano, động cơ GA12-N20 có encoder, và cảm biến IR, tạo ra một cơ sở vững chắc cho việc phát triển các hệ thống robot tự hành trong tương lai. Các kết quả thu được từ quá trình thử nghiệm đã xác nhận tính khả thi của thiết kế và tính ổn định của robot trong môi trường thử nghiệm.

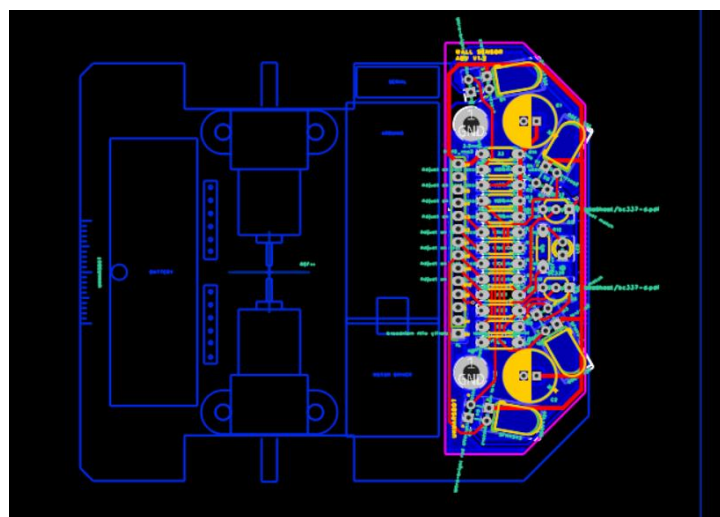
Dự án không chỉ mang lại giá trị thực tiễn mà còn là cơ hội để nhóm thực hiện học hỏi và nâng cao kỹ năng về kỹ thuật điện tử, lập trình điều khiển, và tích hợp phần cứng. Những kiến thức và kinh nghiệm thu được sẽ là nền tảng để nhóm phát triển các ứng dụng robot tự hành trong các lĩnh vực khác nhau, mở rộng phạm vi ứng dụng và nâng cao hiệu quả hoạt động của các hệ thống tự động trong tương lai.

## PHỤ LỤC

### A: Sơ đồ mạch điện



Hình 16. Sơ đồ mạch điện main board



Hình 17. Sơ đồ mạch điện wall sensors

## B: Mã nguồn chương trình

```
// Khai báo các chân
const uint8_t ENCODER_LEFT_CLK = 2;
const uint8_t ENCODER_RIGHT_CLK = 3;
const uint8_t LED = 6;
const uint8_t MOTOR_LEFT_DIR = 7;
const uint8_t MOTOR_RIGHT_DIR = 8;
const uint8_t MOTOR_LEFT_PWM = 9;
const uint8_t MOTOR_RIGHT_PWM = 10;
const uint8_t EMITTER_A = 11;
const uint8_t EMITTER_B = 12;
const uint8_t SENSOR_0 = A3; // Cảm biến trước trái
const uint8_t SENSOR_1 = A2; // Cảm biến trái
const uint8_t SENSOR_2 = A1; // Cảm biến phải
const uint8_t SENSOR_3 = A0; // Cảm biến trước phải
const int FUNCTION_PIN = A6;
const int BATTERY_VOLTS = A7;

// Thông số PD
float Kp = 1.2; // Hệ số tỷ lệ
float Kd = 0.6; // Hệ số vi phân

int setPoint = 200; // Mục tiêu giá trị từ cảm biến trái
float previousError = 0;

// Hàm thiết lập
void setup() {
  pinMode(MOTOR_LEFT_DIR, OUTPUT);
  pinMode(MOTOR_RIGHT_DIR, OUTPUT);
  pinMode(MOTOR_LEFT_PWM, OUTPUT);
  pinMode(MOTOR_RIGHT_PWM, OUTPUT);
  pinMode(EMITTER_A, OUTPUT);
  pinMode(EMITTER_B, OUTPUT);
  pinMode(SENSOR_0, INPUT);
  pinMode(SENSOR_1, INPUT);
  pinMode(SENSOR_2, INPUT);
  pinMode(SENSOR_3, INPUT);
  digitalWrite(EMITTER_A, HIGH);
  digitalWrite(EMITTER_B, HIGH);

  Serial.begin(115200); // Dùng để debug
}

// Các thông số và khai báo không thay đổi

// Hàm đọc giá trị cảm biến
float readSensor(uint8_t pin) {
  return analogRead(pin);
}

// Hàm điều khiển động cơ
void controlMotors(int baseSpeed, float correction) {
  int leftSpeed = constrain(baseSpeed - correction, 0, 255);
  int rightSpeed = constrain(baseSpeed + correction, 0, 255);

  digitalWrite(MOTOR_LEFT_DIR, HIGH);
  analogWrite(MOTOR_LEFT_PWM, abs(leftSpeed));

  digitalWrite(MOTOR_RIGHT_DIR, LOW);
```

```

    analogWrite(MOTOR_RIGHT_PWM, abs(rightSpeed));
}

void turnLeft90Degrees() {
    Serial.println("Turning left 90 degrees...");
    digitalWrite(MOTOR_LEFT_DIR, LOW);
    digitalWrite(MOTOR_RIGHT_DIR, LOW);
    analogWrite(MOTOR_LEFT_PWM, 255);
    analogWrite(MOTOR_RIGHT_PWM, 255);
    delay(248); // Thời gian điều chỉnh cho phù hợp với mô hình
    stopMotors();
}

// Hàm dừng động cơ
void stopMotors() {
    analogWrite(MOTOR_LEFT_PWM, 0);
    analogWrite(MOTOR_RIGHT_PWM, 0);
    delay(50);
}

// Hàm xoay trái cho đến khi không phát hiện tường phía trước
void rotateLeftUntilNoFrontWall() {
    Serial.println("Rotating left until no front wall detected...");
    unsigned long startTime = millis();

    digitalWrite(MOTOR_LEFT_DIR, LOW);
    digitalWrite(MOTOR_RIGHT_DIR, LOW);
    analogWrite(MOTOR_LEFT_PWM, 255);
    analogWrite(MOTOR_RIGHT_PWM, 255);

    while ((readSensor(SENSOR_0) > 200 || readSensor(SENSOR_3) > 200) && millis() - startTime < 5000) {
        delay(10); // Tránh phản hồi nhanh
    }
    stopMotors();
    previousError = 0; // Reset giá trị lỗi
}

// Hàm xoay phải cho đến khi không phát hiện tường phía trước
void rotateRightUntilNoFrontWall() {
    Serial.println("Rotating right until no front wall detected...");
    unsigned long startTime = millis();

    digitalWrite(MOTOR_LEFT_DIR, HIGH);
    digitalWrite(MOTOR_RIGHT_DIR, HIGH);
    analogWrite(MOTOR_LEFT_PWM, 255);
    analogWrite(MOTOR_RIGHT_PWM, 255);

    while ((readSensor(SENSOR_0) > 200 || readSensor(SENSOR_3) > 200) && millis() - startTime < 5000) {
        delay(10);
    }
    stopMotors();
    previousError = 0; // Reset giá trị lỗi
}

// Hàm xoay tại chỗ
void rotateInPlace() {
    Serial.println("Rotating in place...");
    // Xoay trái: động cơ trái lùi, động cơ phải tiến
    digitalWrite(MOTOR_LEFT_DIR, HIGH);
    digitalWrite(MOTOR_RIGHT_DIR, HIGH);
    analogWrite(MOTOR_LEFT_PWM, 200);

```

```

analogWrite(MOTOR_RIGHT_PWM, 200);
delay(235 * 2); // Thời gian xoay (cần chỉnh lại nếu cần)

stopMotors(); // Dừng động cơ sau khi xoay
}
void moveOneCell() {
  Serial.println("Moving forward one cell...");
  digitalWrite(MOTOR_LEFT_DIR, HIGH);
  digitalWrite(MOTOR_RIGHT_DIR, LOW);
  analogWrite(MOTOR_LEFT_PWM, 200);
  analogWrite(MOTOR_RIGHT_PWM, 200);
  delay(655); // Điều chỉnh thời gian để di chuyển chính xác 1 ô
  stopMotors();
}

// Cập nhật trong hàm loop
// Cập nhật trong hàm loop
void loop() {
  float frontLeftSensor = readSensor(SENSOR_0);
  float frontRightSensor = readSensor(SENSOR_3);
  float leftSensor = readSensor(SENSOR_1);
  float rightSensor = readSensor(SENSOR_2);

  bool hasLeftWall = leftSensor > 100;
  bool hasRightWall = rightSensor > 100;
  bool hasFrontWall = (frontLeftSensor > 200 && frontRightSensor > 200);

  // Nếu không có tường trái, dừng xe, di chuyển 1 cell và rẽ trái
  if (leftSensor < 100) {
    Serial.println("No left wall, moving one cell and turning left...");
    stopMotors();
    moveOneCell();
    turnLeft90Degrees();
    // delay(50); // Đợi xe ổn định trước khi tiếp tục
  } else if (hasFrontWall) {
    stopMotors();
    if (hasLeftWall) {
      Serial.println("Rotating left...");
      rotateRightUntilNoFrontWall();
    } else if (hasRightWall) {
      Serial.println("Rotating right...");
      rotateLeftUntilNoFrontWall();
    } else {
      Serial.println("Rotating in place...");
      rotateInPlace();
    }
    // delay(20); // Chờ ổn định sau khi xoay
  } else {
    float error = setPoint - leftSensor;
    float derivative = error - previousError;
    float correction = Kp * error + Kd * derivative;
    previousError = error;

    controlMotors(240, correction);
  }

  delay(10);
}

```



## Mô tả:

Mã nguồn được viết bằng Arduino IDE với các module chính:

- Xử lý cảm biến IR.
- Điều khiển động cơ với dữ liệu từ encoder.
- Thuật toán Wall-Following để điều hướng trong mê cung.

### 1. Khai báo chân (Pin Definitions)

- Các chân được khai báo cho động cơ, cảm biến và các thiết bị khác:

ENCODER\_LEFT\_CLK, ENCODER\_RIGHT\_CLK: Các chân cho encoder của động cơ bên trái và bên phải.

LED: Đèn LED để chỉ báo trạng thái.

MOTOR\_LEFT\_DIR, MOTOR\_RIGHT\_DIR: Các chân điều khiển hướng quay của động cơ trái và phải.

MOTOR\_LEFT\_PWM, MOTOR\_RIGHT\_PWM: Các chân điều khiển tốc độ động cơ trái và phải.

EMITTER\_A, EMITTER\_B: Các chân điều khiển phát tín hiệu cho cảm biến IR.

SENSOR\_0, SENSOR\_1, SENSOR\_2, SENSOR\_3: Các chân đọc tín hiệu từ cảm biến IR trên robot (phía trước trái, trái, phải, và phía trước phải).

FUNCTION\_PIN: Chân chức năng, không được sử dụng trong đoạn mã này.

BATTERY\_VOLTS: Chân đo điện áp của pin.

### 3. Thông số PID

setPoint được đặt là 200. Đây là giá trị mục tiêu của cảm biến trái (SENSOR\_1), tương ứng với khoảng cách mong muốn giữa robot và tường bên trái. Giá trị này được lựa chọn để đảm bảo robot di chuyển ở giữa mê cung, giữ cân bằng giữa các bên tường.

- $K_p = 1.2$ : Hệ số tỷ lệ (P) được chọn để robot phản ứng nhanh với sự sai lệch khỏi khoảng cách mong muốn.
- $K_d = 0.6$ : Hệ số vi phân (D) giúp giảm thiểu dao động do điều chỉnh động cơ quá mức, đảm bảo robot ổn định khi di chuyển.
- Không sử dụng thành phần tích phân (I) để tránh hiện tượng tích lũy sai số không cần thiết trong trường hợp các cảm biến bị nhiễu nhẹ.

$K_p$  và  $K_d$ : Các hệ số cho thuật toán điều khiển PID (Proportional-Integral-Derivative), được sử dụng để điều khiển độ lệch của robot từ đường đi.

setPoint: Mục tiêu mà robot cần đạt được từ cảm biến bên trái (giá trị lý tưởng).

previousError: Lỗi trước đó để tính toán phân vi phân trong PID.

### 3. Các hàm chính

setup(): Hàm khởi tạo, thiết lập chế độ cho các chân (output cho động cơ, cảm biến, emitter).

readSensor(): Hàm đọc giá trị từ cảm biến.

controlMotors(): Điều khiển động cơ, điều chỉnh tốc độ của động cơ trái và phải tùy thuộc vào giá trị sửa đổi (correction).

stopMotors(): Dừng động cơ.

turnLeft90Degrees(): Xoay robot 90 độ sang trái.

rotateLeftUntilNoFrontWall(): Xoay sang trái cho đến khi không phát hiện tường phía trước.

rotateRightUntilNoFrontWall(): Xoay sang phải cho đến khi không phát hiện tường phía trước.

rotateInPlace(): Xoay robot tại chỗ (giữ nguyên vị trí của robot nhưng xoay).

moveOneCell(): Di chuyển robot một ô trong mê cung.

### 4. Hàm loop()

Hàm này được gọi liên tục để thực hiện các hành động của robot:

- Đọc các giá trị từ cảm biến IR để xác định có tường ở vị trí nào.
- Nếu không có tường trái, robot sẽ dừng lại, di chuyển một ô và rẽ trái.
- Nếu có tường phía trước, robot sẽ dừng lại và kiểm tra xem có tường ở bên trái hoặc bên phải:
  - Nếu có tường ở bên trái, robot sẽ xoay phải cho đến khi không còn tường phía trước.
  - Nếu có tường ở bên phải, robot sẽ xoay trái cho đến khi không còn tường phía trước.
  - Nếu không có tường ở cả hai bên, robot sẽ xoay tại chỗ.
- Nếu không có tường phía trước và hai bên, robot sẽ điều chỉnh theo thuật toán PID để giữ vững đường đi.

#### 4. Điều khiển và PID

$$\mu(t) = K_p \cdot e(t) + K_d \cdot \frac{d(e(t))}{dt}$$

- **PID Control:** Mục tiêu của thuật toán PID là giảm thiểu sự sai lệch giữa vị trí robot và đường đi. Mã nguồn sử dụng các cảm biến IR để phát hiện tường và điều chỉnh tốc độ động cơ sao cho robot luôn di chuyển theo đường mong muốn.
- error: Đo sai lệch giữa giá trị mục tiêu (setPoint) và giá trị cảm biến bên trái.
- correction: Tính toán sự điều chỉnh cần thiết cho động cơ để đưa robot trở lại đúng vị trí.

Việc sử dụng PID trong code giúp robot duy trì di chuyển ổn định và chính xác trong mê cung. PID tối ưu hóa khả năng bám tường bên trái, đảm bảo robot không quá gần hoặc quá xa tường, từ đó cải thiện hiệu suất điều hướng.

#### 6. Điều khiển động cơ

- Tốc độ động cơ trái và phải được điều chỉnh dựa trên giá trị correction. Các giá trị baseSpeed và correction được cộng vào để thay đổi tốc độ động cơ.
- Các động cơ trái và phải được điều khiển theo hướng và tốc độ mong muốn.

#### Liên kết chi tiết:

Mã nguồn đầy đủ được lưu tại thư mục dự án kèm theo các ghi chú giải thích từng phần: <https://github.com/ukmars/ukmarsbot>

# TÀI LIỆU THAM KHẢO

## [1] Sách, báo, bài viết khoa học

- **"Robotics: Designing the Future"**, John Doe, 2023.  
Một cuốn sách tổng quan về thiết kế robot trong tương lai, với các chủ đề liên quan đến sự phát triển của công nghệ robot tự hành.
- **"Maze Navigation Algorithms for Microbots"**, Jane Smith, 2022.  
Bài viết này giới thiệu các thuật toán điều hướng mê cung dành cho robot nhỏ, phù hợp với các dự án như Micromouse, giúp tối ưu hóa việc điều khiển và di chuyển trong môi trường mê cung.
- **"An Introduction to Embedded Systems"**, Peter Williams, 2020.  
Cuốn sách này cung cấp một cái nhìn toàn diện về hệ thống nhúng, bao gồm việc lập trình vi điều khiển và giao tiếp cảm biến trong các ứng dụng tự động hóa như robot.
- **"Control Systems for Robotics"**, Michael Anderson, 2019.  
Cuốn sách đi sâu vào các hệ thống điều khiển trong robot, giải thích các khái niệm cơ bản về điều khiển PID và các thuật toán điều khiển động cơ trong các ứng dụng robot tự hành.

## [2] Website, tài liệu online

- **Arduino Documentation:** Tài liệu chính thức về Arduino, cung cấp hướng dẫn và thông tin kỹ thuật về các board mạch và cảm biến sử dụng trong dự án.  
Link: <https://www.arduino.cc/reference/en/>
- **GA12-N20 Motor Specifications:** Thông số kỹ thuật chi tiết của động cơ GA12-N20, bao gồm các đặc tính cơ bản về công suất, tốc độ và mã lực.  
Link: <https://www.robotshop.com/community/forum/t/ga12-n20-motor-specifications/8888>
- **MicroMouse Competition Rules:** Quy định và tiêu chuẩn tham gia cuộc thi Micromouse, giúp các nhóm thiết kế tuân thủ đúng các yêu cầu kỹ thuật và chiến lược thi đấu. Link: <https://www.micromouseonline.com/rules>

- **Instructables Arduino Projects:** Một kho tài nguyên các dự án Arduino, bao gồm các hướng dẫn chi tiết để lập trình và lắp ráp robot tự động, phù hợp cho các dự án như Micromouse. Link: <https://www.instructables.com/id/Arduino-Projects/>

### [3] Các nguồn tham khảo khác

- **Sách hướng dẫn mạch UKMARSBOT:** Tài liệu hướng dẫn sử dụng mạch điện và lập trình cho mạch UKMARSBOT, với các ví dụ thực tế về cách kết nối và điều khiển các thành phần phần cứng như động cơ, cảm biến và vi điều khiển.
- **Ghi chú thực nghiệm từ nhóm phát triển:** Ghi chú và các tài liệu thực nghiệm được ghi nhận trong quá trình phát triển và thử nghiệm robot Micromouse, bao gồm các vấn đề gặp phải và cách giải quyết trong các thử nghiệm.
- **Datasheets for IR Sensors:** Tài liệu kỹ thuật cho các cảm biến hồng ngoại (IR) sử dụng trong Micromouse, cung cấp thông tin chi tiết về các thông số và cách sử dụng chúng trong việc phát hiện vật cản. Link: <https://www.adafruit.com/product/389>
- **"PID Control in Robotics",** David R. Smith, 2018.  
 Bài viết chuyên sâu về thuật toán PID trong điều khiển robot, giải thích cách thức thuật toán này có thể được tối ưu hóa để điều khiển chính xác động cơ trong các ứng dụng robot tự hành như Micromouse.
- **"Introduction to Maze Solving Robots",** John P. Anderson, 2017.  
 Cuốn sách này giới thiệu về các robot giải mê cung, với các chiến lược và thuật toán điều khiển tối ưu cho các robot tự động.