

**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN – ĐIỆN TỬ
BỘ MÔN VIỄN THÔNG**



LUẬN VĂN TỐT NGHIỆP

**ROBOT TỰ HÀNH TRÁNH VẬT CẢN SỬ DỤNG
THIẾT BỊ KINECT**

**GVHD: PGS. TS. Hoàng Đình Chiến
SVTH : Nguyễn Hồng Đức 40700566
 Nguyễn Văn Đức 40700577**

- Tp. Hồ Chí Minh, Tháng 1-2012 -

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐH BÁCH KHOA
Thành phố Hồ Chí Minh
ς ★ δ

Số: _____/BKĐT
Khoa: Điện – Điện tử
Bộ Môn: Viễn Thông

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc Lập – Tự Do – Hạnh Phúc
ς ★ δ

NHIỆM VỤ LUẬN VĂN TỐT NGHIỆP

Họ và tên:

NGUYỄN HỒNG ĐỨC

MSSV: **40700566**

Họ và tên:

NGUYỄN VĂN ĐỨC

MSSV: **40700577**

Ngành:

VIỄN THÔNG

LỚP: **DD07DV4**

1. Đầu đề luận văn: “**Robot tự hành tránh vật cản sử dụng thiết bị Kinect**”

2. Nhiệm vụ (Yêu cầu về nội dung và số liệu ban đầu):

.....
.....
.....
.....

3. Ngày giao nhiệm vụ luận văn:

4. Ngày hoàn thành nhiệm vụ:

5. Họ và tên người hướng dẫn:

Phản hướng dẫn

Nội dung và yêu cầu LVTN đã được thông qua Bộ Môn.

Ngày tháng năm 2012

CHỦ NHIỆM BỘ MÔN
(Ký và ghi rõ họ tên)

NGƯỜI HƯỚNG DẪN CHÍNH
(Ký và ghi rõ họ tên)

PHẦN DÀNH CHO KHOA, BỘ MÔN:

Người duyệt (chấm sơ bộ):

Đơn vị:

Ngày bảo vệ:

Điểm tổng kết:

Nơi lưu trữ luận văn:

PHIẾU CHẤM BẢO VỆ LVTN
(Dành cho người hướng dẫn)

Họ và tên: **NGUYỄN HỒNG ĐỨC** MSSV: **40700566**
Họ và tên: **NGUYỄN VĂN ĐỨC** MSSV: **40700577**
Ngành: **VIỄN THÔNG** LỚP: **DD07DV4**

1. Đề tài: **“Robot tự hành tránh vật cản sử dụng thiết bị Kinect”**
2. Họ tên người hướng dẫn: **PGS. TS. HOÀNG ĐÌNH CHIỀN**
3. Tổng quát về bản thuyết minh:

Số trang	Số chương
Số bảng số liệu	Số hình vẽ
Số tài liệu tham khảo	Phần mềm tính toán
4. Tổng quát về các bản vẽ:

- Số bản vẽ:	bản A1	bản A2	khô khắc
- Số bản vẽ tay			số bản vẽ trên máy tính
5. Những ưu điểm chính của LVTN:

6. Những thiếu sót chính của LVTN:

7. Đề nghị: Được bảo vệ Bổ sung thêm để bảo vệ
Không được bảo vệ
8. 3 câu hỏi sinh viên trả lời trước Hội Đồng:
 - a)
 - b)
 - c)
9. Đánh giá chung (*bằng chữ: giỏi, khá, TB*): Điểm

Ký tên (*ghi rõ họ tên*)

PHIẾU CHẤM BẢO VỆ LVTN
(Dành cho người phản biện)

Họ và tên: **NGUYỄN HỒNG ĐỨC** MSSV: **40700566**
Họ và tên: **NGUYỄN VĂN ĐỨC** MSSV: **40700577**
Ngành: **VIỄN THÔNG** LÓP: **DD07DV4**

10. Đề tài: “Robot tự hành tránh vật cản sử dụng thiết bị Kinect”

11. Họ tên người phản biện:

12. Tổng quát về bản thuyết minh:

Số trang	Số chương
Số bảng số liệu	Số hình vẽ
Số tài liệu tham khảo	Phần mềm tính toán

13. Tổng quát về các bản vẽ:

- Số bản vẽ:	bản A1	bản A2	khô khắc
- Số bản vẽ tay			số bản vẽ trên máy tính

14. Những ưu điểm chính của LVTN:

15. Những thiếu sót chính của LVTN:

16. Đề nghị: Được bảo vệ , Bổ sung thêm để bảo vệ
Không được bảo vệ .

17. 3 câu hỏi sinh viên trả lời trước Hội Đồng:

a)

b)

c)

18. Đánh giá chung (bằng chữ: giỏi, khá, TB): Điểm

Ký tên (*ghi rõ họ tên*)

LỜI CẢM ƠN

Lời đầu tiên, chúng em xin gửi đến Thầy, PGS. TS. Hoàng Đình Chiến lời cảm ơn chân thành và sâu sắc nhất. Nhờ có sự hướng dẫn và giúp đỡ tận tình của Thầy trong suốt thời gian qua, chúng em đã có thể thực hiện và hoàn thành Đồ Án Môn Học 2, Thực Tập Tốt Nghiệp và Luận Văn Tốt Nghiệp. Những lời nhận xét, góp ý và hướng dẫn tận tình của Thầy đã giúp chúng em có một định hướng đúng đắn trong suốt quá trình thực hiện Đề tài, giúp chúng em nhìn ra được những ưu khuyết điểm của Đề tài và từng bước hoàn thiện hơn.

Đồng thời, chúng em xin trân trọng cảm ơn các Thầy Cô của Trường Đại Học Bách Khoa nói chung và của khoa Điện - Điện Tử nói riêng đã dạy dỗ chúng em suốt quãng thời gian ngồi trên ghế giảng đường Đại học. Những lời giảng của Thầy Cô trên bục giảng đã trang bị cho chúng em những kiến thức và giúp chúng em tích lũy thêm những kinh nghiệm.

Chúng em cũng xin gửi lời cảm ơn tới hai cựu sinh viên Bách Khoa: anh Nguyễn Quốc Thịnh và anh Nguyễn Thanh Tâm đã tận tình hướng dẫn chúng em định hướng đúng trong những ngày bắt đầu nhận đề tài luận văn.

Bên cạnh đó, chúng tôi xin cảm ơn sự hỗ trợ và giúp đỡ của bạn bè trong thời gian học tập tại Trường Đại Học Bách Khoa và trong quá trình hoàn thành Luận Văn Tốt Nghiệp này.

Cuối cùng, chúng con cũng chân thành cảm ơn sự động viên và sự hỗ trợ của gia đình và cha mẹ trong suốt thời gian học tập. Đặc biệt, chúng con xin gửi lời cảm ơn trân trọng nhất đến cha mẹ, người đã sinh ra và nuôi dưỡng chúng con nên người. Sự quan tâm, lo lắng và hy sinh lớn lao của cha mẹ luôn là động lực cho chúng con cố gắng phấn đấu trên con đường học tập của mình. Một lần nữa, chúng con xin gửi đến cha mẹ sự biết ơn sâu sắc nhất.

Hồ Chí Minh, ngày 8 tháng 1 năm 2012

NGUYỄN HỒNG ĐỨC

NGUYỄN VĂN ĐỨC

TÓM TẮT LUẬN VĂN

Theo dự đoán trong tương lai, robot sẽ là tâm điểm của một cuộc cách mạng lớn sau Internet. Con người sẽ có nhu cầu sở hữu một robot cá nhân như nhu cầu một máy tính PC bây giờ. Với xu hướng này, cùng các ứng dụng truyền thống khác của robot trong công nghiệp, y tế, giáo dục đào tạo, giải trí và đặc biệt là trong an ninh quốc phòng thì thị trường robot sẽ vô cùng to lớn. Để tài luận văn hướng tới việc ứng dụng công nghệ xử lý ảnh mới cho robot tự hành, tạo tiền đề cho việc xây dựng một robot dịch vụ hoàn chỉnh, có khả năng phục vụ cho đời sống con người.

Trong khuôn khổ của luận văn, nhóm sẽ tập trung xây dựng một mô hình mobile robot hoàn chỉnh có khả năng tìm đường đến đích và tránh chướng ngại vật trên quãng đường di chuyển. Một điểm mới được nhấn mạnh là khôi phục thị giác máy tính cho mobile robot, với sự hỗ trợ của thiết bị chơi game Kinect có khả năng khôi phục môi trường phía trước robot dưới dạng 3D, đáp ứng được sự chính xác cần thiết khi phối hợp với các giải thuật điều khiển truyền thống cho robot.

Nhóm sinh viên thực hiện

NGUYỄN HỒNG ĐỨC

NGUYỄN VĂN ĐỨC

Đề mục	Trang
Lời cảm ơn.....	i
Tóm tắt luận văn.....	ii
Mục lục	iii
Danh mục từ viết tắt	v
Danh mục hình	viii
Danh mục bảng.....	xi

Mục lục

Chương 1: Giới thiệu	1
1.1 Xu hướng phát triển của robot hiện đại	2
1.2 Những vấn đề của robot di động.....	2
1.3 Mục tiêu luận văn và phương pháp thực hiện.....	3
1.4 Sơ lược về nội dung luận văn	4
Chương 2: Tìm hiểu về Kinect	5
2.1 Giới thiệu chung.....	6
2.2 Những thành phần chính của Kinect.....	7
2.3 Tính toán độ sâu.....	8
2.4 Một số đặc tính khác	12
Chương 3: Thư viện xử lý ảnh	15
3.1 Thư viện hỗ trợ Kinect.....	16
3.2 So sánh Kinect SDK beta và OpenNI	17
3.3 Point Cloud Library	20
Chương 4: Phát hiện vật cản	22
4.1 Các phương pháp phát hiện vật cản không sử dụng camera.....	23
4.1.1 Dùng công tắc hành trình.....	23
4.1.2 Dùng cảm biến siêu âm [13].....	23
4.2 Các phương pháp phát hiện vật cản sử dụng camera.....	26

4.2.1	Xử lý ảnh với một camera (Monocular vision).....	26
4.2.2	Xử lý ảnh với hai camera (Stereo vision)	29
4.3	Phát hiện vật cản sử dụng Kinect.....	31
Chương 5: Module điều khiển động cơ	39	
5.1	PIC 18F4550	40
5.1.1	Giới thiệu chung	40
5.1.2	Những module chính sử dụng trong luận văn	45
5.2	Mạch công suất (mạch cầu H)	55
Chương 6: Động cơ và giải thuật PID vị trí.....	56	
6.1	Động cơ Servo DC	57
6.1.1	Động cơ DC	57
6.1.2	Encoder	59
6.2	Giải thuật PID vị trí [19]	63
Chương 7: Tính toán tọa độ Robot và Kinect	68	
7.1	Các phép chuyển đổi hệ trực tọa độ cơ bản	69
7.2	Tính toán tọa độ robot.....	70
7.3	Tính toán tọa độ Kinect.....	73
Chương 8: Chương trình điều khiển	76	
8.1	Nội dung chương trình điều khiển	77
8.2	Giải thuật chương trình do máy tính xử lý.....	77
8.3	Giải thuật chương trình do vi điều khiển xử lý.....	88
KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	90	
TÀI LIỆU THAM KHẢO	92	
PHỤ LỤC 1: Kết hợp thư viện OpenNI và Code Laboratories Kinect (CL) để sử dụng chức năng điều khiển động cơ Kinect.....	93	
PHỤ LỤC 2: Cách đấu dây dùng pin 12V thay adapter và tạo đế gắn lên robot cho Kinect.....	96	
PHỤ LỤC 3: Kích thước robot	99	

Danh mục từ viết tắt

A

ADC	Analog Digital Converter
AGV	Autonomous Guided Vehicles
API	Application Programming Interface
AUV	Autonomous Underwater Vehicles
AUX	AUXiliary

C

CCP	Capture/Compare/PWM
CL	Code Laboratories
CMOS	Complementary Metal – Oxide – Semiconductor
CNC	Computerized Numerical Control
CPU	Central Processing Unit

E

ECCP	Enhanced Capture/Compare/PWM
EEPROM	Electrically Erasable Programmable Read – Only Memory
EUSART	Enhanced Universal Synchronous Asynchronous Receiver Transmitter

F

FLANN	Fast Library for Approximate Nearest Neighbors
--------------	--

G

GPIO	General Purpose Input Output
-------------	------------------------------

H

HDD	Hard Disk Drive
------------	-----------------

I

ICD	In – Circuit Debugger
ICSP	In – Circuit Serial Programming
IR	Infrared

J

JNA	Java Native Access
JNI	Java Native Interface

M

MFC	Microsoft Foundation Class Library
MSSP	Master Synchronous Serial Port
MUX	Multiplexer

N

NI	Natural Interaction
NUI	Natural User Interface

Q

QVGA	Quarter Video Graphics Array
-------------	------------------------------

R

RAM	Random Access Memory
RANSAC	RANDom SAMple Consensus
RGB	Red, Green, Blue

S

SDK	Software Development Kit
------------	--------------------------

SSP Synchronous Serial Port
SXGA Super eXtended Graphics Array

T

TOF Time Of Flight
TTL Transistor – Transistor Logic

U

UAV Unmanned Aerial Vehicles
USART Universal Synchronous Asynchronous Receiver Transmitter
USB Universal Serial Bus

V

VGA Video Graphics Array
VTK Visualization Toolkit

Danh mục hình

Hình 2.1: Thiết bị Kinect.....	6
Hình 2.2: Những thành phần chính của Kinect.....	7
Hình 2.3: Động cơ điều khiển góc ngang Kinect	8
Hình 2.4: Bên trong Kinect: RGB, IR camera và IR projector	8
Hình 2.5: Quá trình thu về bản đồ độ sâu ảnh.....	9
Hình 2.6: Mẫu hình được chiếu bởi projector và chụp lại bằng IR camera	10
Hình 2.7: Tính toán khoảng cách tới một điểm chiếu từ Projector	11
Hình 2.8: Kinect adapter	14
Hình 3.1: Thư viện OpenNI phối hợp giữa phần cứng và ứng dụng đầu cuối.....	19
Hình 3.2: Point cloud library logo.....	20
Hình 4.1: Mô hình robot dùng công tắc hành trình.....	23
Hình 4.2: Cảm biến siêu âm	24
Hình 4.3: Hiện tượng Forecasting	25
Hình 4.4: Hiện tượng Crosstalk.....	25
Hình 4.5: Thị trường của robot với optical flow	26
Hình 4.6: Ảnh gốc và ảnh sau khi tách biên.....	27
Hình 4.7: Hạn chế của phương pháp dò biên	28
Hình 4.8: Phương pháp dò nền.....	28
Hình 4.9: Phương pháp Stereo Vision.....	29
Hình 4.10: Sơ đồ xử lý: phát hiện và tách vật cản	31
Hình 4.11: Ảnh RGB và bản đồ độ sâu.....	32
Hình 4.12: RGB point cloud.....	33
Hình 4.13: Voxel grid.....	34
Hình 4.14: Pass through không có và có voxel grid.....	35
Hình 4.15: Tìm mô hình đường thẳng bằng thuật toán RANSAC.....	36
Hình 4.16: Point cloud sau khi thực hiện xong bước lọc và phân đoạn.....	37

Hình 4.17: Object cluster.....	38
Hình 5.1: PICLAB-V2 và board mạch cầu H	40
Hình 5.2: PIC 18F4550	40
Hình 5.3: Sơ đồ chân PIC18F4550	41
Hình 5.4: Sơ đồ khói của PIC 18F4550	44
Hình 5.5: Sơ đồ khói bộ dao động của PIC 18F4550	45
Hình 5.6: Cấu tạo của chân GPIO	48
Hình 5.7: Sơ đồ khói logic của hệ thống ngắn trong PIC18F4550	49
Hình 5.8: Sơ đồ khói của bộ Timer 1	50
Hình 5.9: Sơ đồ khói của bộ Timer 2	51
Hình 5.10: Sơ đồ khói bộ PWM	52
Hình 5.11: Giản đồ thời gian của sóng điều xung tại chân CCPx	52
Hình 5.12: Sơ đồ khói bộ truyền của module EUSART	54
Hình 5.13: Sơ đồ khói bộ nhận của module EUSART	55
Hình 6.1: Cặp động cơ Servo DC.....	57
Hình 6.2: Điều chỉnh độ rộng xung PWM	58
Hình 6.3: Dạng sóng áp và dòng trên động cơ.....	59
Hình 6.4: Optical Encoder.....	60
Hình 6.5: Hai kênh A và B lệch pha trong encoder	61
Hình 6.6: Encoder đi kèm động cơ Servo DC.....	62
Hình 6.7: PID vòng kín	63
Hình 6.8: Đáp ứng của các hệ thống điều khiển	65
Hình 6.9: Quá trình tính toán PID	66
Hình 7.1: Phép tịnh tiến.....	69
Hình 7.2: Phép quay	70
Hình 7.3: Mô hình robot.....	71
Hình 7.4: Tọa độ robot (quan sát từ trên xuống).....	72
Hình 7.5: Hệ trực tọa độ Kinect	73
Hình 7.6: Đồng nhất hệ trực tọa độ Kinect và Robot	74

Hình 7.7: Tọa độ Kinect trước (trái) và sau (phải) khi chuyển trực	75
Hình 8.1: Nội dung chương trình điều khiển	77
Hình 8.2: Xử lý đa tiến trình	78
Hình 8.3: Giao diện chương trình điều khiển	78
Hình 8.4: Sơ đồ giải thuật điều khiển robot do máy tính xử lý	80
Hình 8.5: Tính góc quay về đích	81
Hình 8.6: Vật cản bên trái robot	83
Hình 8.7: Vật cản bên phải robot	84
Hình 8.8: Vật cản nằm ở giữa đường di chuyển của robot	85
Hình 8.9: Đi một đoạn an toàn về phía phải vật cản	86
Hình 8.10: Cờ báo có vật cản và đường trống	87
Hình 8.11: Không gian cho robot lách qua	88
Hình 8.12: Sơ đồ giải thuật trên vi điều khiển	89

Danh mục bảng

Bảng 2.1: Góc mở và tiêu cự RGB và IR camera	13
Bảng 2.2: Công suất tiêu thụ trên Kinect	14
Bảng 5.1: Bảng mô tả các chức năng từng chân của PIC18F4550	42
Bảng 6.1: Ảnh hưởng của các thành phần Kp, Ki, Kd đối với hệ kín.....	64

Chương 1: Giới thiệu

Nội dung chính

1.1 Xu hướng phát triển của robot hiện đại

1.2 Những vấn đề của robot di động

1.3 Mục tiêu luận văn và phương pháp thực hiện

1.4 Sơ lược về nội dung luận văn

1.1 Xu hướng phát triển của robot hiện đại

Theo dự đoán trong vòng 20 năm nữa mỗi người sẽ có nhu cầu sử dụng một robot cá nhân như nhu cầu một máy tính PC hiện nay và robot sẽ là tâm điểm của một cuộc cách mạng lớn sau Internet. Với xu hướng này, cùng các ứng dụng truyền thống khác của robot trong công nghiệp, y tế, giáo dục đào tạo, giải trí và đặc biệt là trong an ninh quốc phòng thì thị trường robot sẽ vô cùng to lớn.

Robot đã có những bước tiến đáng kể trong hơn nửa thế kỷ qua. Robot đầu tiên được ứng dụng trong công nghiệp vào những năm 60 để thay thế con người làm những công việc nặng nhọc, nguy hiểm trong môi trường độc hại. Do nhu cầu sử dụng ngày càng nhiều trong quá trình sản xuất phức tạp nên robot công nghiệp cần có những khả năng thích ứng linh hoạt và thông minh hơn. Ngày nay, ngoài ứng dụng sơ khai ban đầu của robot trong chế tạo máy thì các ứng dụng khác như trong y tế, chăm sóc sức khỏe, nông nghiệp, đóng tàu, xây dựng, an ninh quốc phòng đang là động lực cho sự phát triển của ngành công nghiệp robot.

Có thể kể đến những loại robot được quan tâm nhiều trong thời gian qua là: tay máy robot (Robot Manipulators), robot di động (Mobile Robots), robot phòng sinh học (Bio Inspired Robots) và robot cá nhân (Personal Robots). Robot di động được nghiên cứu nhiều như xe tự hành trên mặt đất AGV (Autonomous Guided Vehicles), robot tự hành dưới nước AUV (Autonomous Underwater Vehicles), robot tự hành trên không UAV (Unmanned Arial Vehicles) và robot vũ trụ (Space robots). Với robot phòng sinh học, các nghiên cứu trong thời gian qua tập trung vào hai loại chính là robot đi bộ (Walking robot) và robot dáng người (Humanoid robot). Bên cạnh đó các loại robot phòng sinh học như cá dưới nước, các cấu trúc chuyển động phỏng theo sinh vật biển cũng được nhiều nhóm nghiên cứu, phát triển.

1.2 Những vấn đề của robot di động

Robot di động được định nghĩa là một loại xe robot có khả năng tự di chuyển, tự vận động (có thể lập trình lại được) dưới sự điều khiển tự động để thực hiện tốt những công việc được giao. Môi trường hoạt động của robot có thể là đất, nước, không khí,

không gian vũ trụ hay là sự tổ hợp của các môi trường trên. Bề mặt địa hình robot di chuyển có thể là bằng phẳng hoặc thay đổi lồi lõm.

Những ứng dụng thực tế đòi hỏi những robot di động có tính tự động cao và những kỹ thuật hiện đại, bao gồm sự đa dạng của những cảm biến rẻ mà đáng tin cậy và tính toán điện tử công suất làm tăng tính tự động hóa của robot di động. Tính tự động hóa có nghĩa là robot phải dựa vào chính khả năng của nó để xuất ra những dữ liệu vận hành có ích từ bộ phận cảm biến và tự nó đưa ra quyết định thích hợp.

Một trong các yêu cầu cơ bản của robot tự động thực thụ là khả năng định hướng tốt trong phạm vi môi trường chưa xác định và hình dung ra một bản đồ định hướng. Bằng cách sử dụng những quan sát thích hợp từ môi trường, kết hợp với bản đồ cùng lúc để định hướng cho robot đang là một yêu cầu cần nghiên cứu cho robot di động. Việc đồng thời định vị và vẽ bản đồ cùng lúc là một phương pháp chung có liên quan đến việc triển khai một hệ thống di động trong môi trường chưa xác định. Đối với một robot di động tự động, định hướng là một công việc để di chuyển một cách an toàn từ nơi này đến nơi khác.

Việc định hướng gặp nhiều khó khăn do nhiều vấn đề khá phức tạp. Vấn đề gây trở ngại chính là những hạn chế của việc ước tính năng lượng, những khó khăn trong việc phát hiện và nhận biết đối tượng, những khó khăn trong việc tránh xung đột với các đối tượng khác nhau, và những khó khăn liên quan tới việc sử dụng thông tin cung cấp từ môi trường.

1.3 Mục tiêu luận văn và phương pháp thực hiện

- **Mục tiêu luận văn:**

Xây dựng mô hình mobile robot có khả năng tự định hướng về đích với tọa độ đích cho trước lúc bắt đầu khởi động. Ngoài khả năng định hướng, robot phải tránh được các vật cản có trên quãng đường di chuyển. Mục đích sâu xa của đề tài là phát triển một robot tự động thông minh, có thể được sử dụng trong lĩnh vực phục vụ con người, giúp việc như robot hướng dẫn du khách hoặc làm một số công việc đơn giản trong văn phòng hoặc tại gia như mang café, hút bụi, lau nhà, ...

▪ **Phương pháp thực hiện:**

Thiết kế mô hình một robot di động tự động, trong đó bao gồm phần gia công cơ khí, mạch công suất, mạch vi điều khiển, đồng thời kết hợp kỹ thuật xử lý ảnh thông qua máy tính (hay thị giác máy tính). Khối thi giác được chọn là thiết bị chơi game Kinect, thông qua xử lý từ máy tính nó sẽ kết hợp với mạch vi điều khiển giúp robot có khả năng định hướng về đích và tránh vật cản hoàn toàn tự động.

1.4 Sơ lược về nội dung luận văn

Nội dung luận văn bao gồm 8 chương và 3 phụ lục:

- **Chương 1: Giới thiệu.** Sơ lược về nội dung đề tài.
- **Chương 2: Tìm hiểu về Kinect.** Nội dung chính đề cập tới phần cứng Kinect, các sensor tích hợp (RGB camera, depth sensor, microphone) và cách thức hoạt động của Kinect.
- **Chương 3: Thư viện xử lý ảnh.** Giới thiệu sơ lược các thư viện hiện hành cho Kinect, so sánh tổng quan các thư viện và đi sâu vào thư viện sử dụng trong luận văn.
- **Chương 4: Phát hiện vật cản.** Giới thiệu các phương pháp tránh vật cản phổ biến, tập trung vào phương pháp xử lý ảnh trong không gian 3D nhằm đưa ra các thông số chính xác nhất về vật cản cho kế hoạch di chuyển của robot.
- **Chương 5: Module điều khiển động cơ.** Bao gồm: mạch vi điều khiển sử dụng PIC 18F4550 và mạch công suất.
- **Chương 6: Động cơ và giải thuật PID vị trí.** Giới thiệu sơ lược động cơ Servo DC và chi tiết giải thuật PID vị trí.
- **Chương 7: Tính toán tọa độ Robot và Kinect.** Sơ lược về các phép chuyển đổi hệ trực cơ bản, dời hệ trực tọa độ Kinect về hệ trực tọa độ robot để đồng nhất cho quá trình xử lý.
- **Chương 8: Chương trình điều khiển.** Tập trung vào giải thuật điều khiển cho máy tính và module điều khiển động cơ.

Chương 2: Tìm hiểu về Kinect

Nội dung chính

2.1 Giới thiệu chung

2.2 Những thành phần chính của Kinect

2.3 Tính toán độ sâu

2.4 Một số đặc tính khác

2.1 Giới thiệu chung

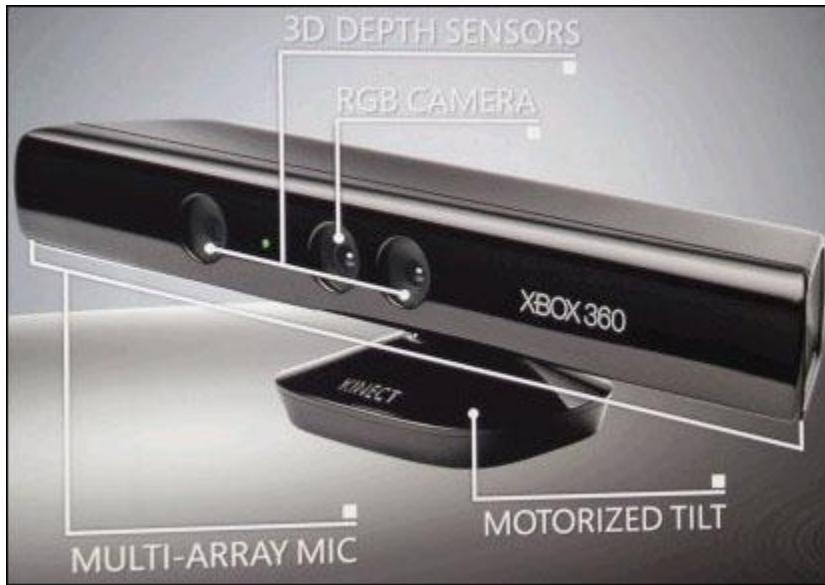


Hình 2.1: Thiết bị Kinect

Kinect là sản phẩm của Microsoft dựa trên công nghệ camera được phát triển bởi PrimeSense, những sản phẩm đầu tiên được bán tại Bắc Mỹ vào ngày 4 tháng 11 năm 2010 [1]. Kinect được coi như là một thiết bị ngoại vi cho Xbox 360, cho phép giao tiếp với con người thông qua các cử chỉ, đem lại những cảm giác thú vị cho người chơi game trên Xbox. Khả năng hiểu được cử chỉ con người của Kinect dựa trên hai đặc tính chính sau: thông tin về độ sâu ảnh (depth map), khả năng phát hiện và bám theo đặc tính cơ thể người (body skeleton tracking).

Kinect đang giữ kỷ lục Guinness thế giới về “Thiết bị điện tử được tiêu thụ nhanh nhất” với 8 triệu sản phẩm trong 60 ngày. Mười triệu sản phẩm Kinect đã được phân phối trên thế giới vào ngày 9 tháng 3 năm 2011. Bên cạnh phục vụ cho mục đích chơi game, sản phẩm Kinect còn được dùng vào mục đích nghiên cứu xử lý ảnh 3D, phát hiện cử chỉ (gesture recognition), bám theo người (body tracking) và nhiều mục đích khác. Lý do chính cho sự thành công của sản phẩm Kinect là giá cả khá rẻ (khoảng 140\$ trên 1 sản phẩm) cho thiết bị có khả năng cung cấp các thông tin 3D với chất lượng chấp nhận được.

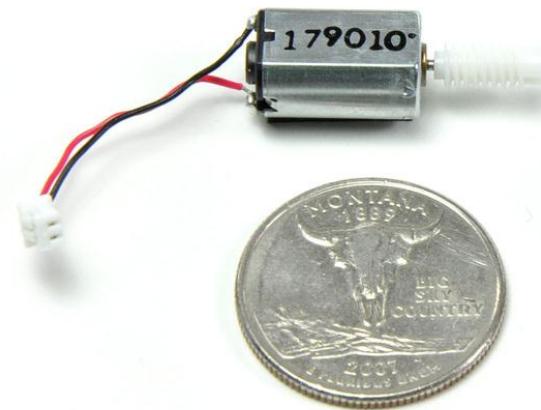
2.2 Những thành phần chính của Kinect



Hình 2.2: Những thành phần chính của Kinect

Kinect gồm có: RGB camera, cảm biến độ sâu (3D Depth Sensors), dãy microphone (Multi-array Mic) và động cơ điều khiển góc ngẩng (Motorized Tilt).

- **RGB Camera:** như một camera thông thường, có độ phân giải 640×480 với tốc độ 30 fps.
- **Cảm biến độ sâu:** độ sâu được thu về nhờ sự kết hợp của hai cảm biến: đèn chiếu hồng ngoại (IR Projector) và camera hồng ngoại (IR camera).
- **Dãy đa microphone:** gồm bốn microphone được bố trí dọc Kinect như trên hình 2.2, được dùng vào các ứng dụng điều khiển bằng giọng nói.
- **Động cơ điều khiển góc ngẩng:** là loại động cơ DC khá nhỏ, cho phép ta điều chỉnh camera lên xuống để bảo đảm camera có được góc nhìn tốt nhất.



Hình 2.3: Động cơ điều khiển góc ngǎng Kinect

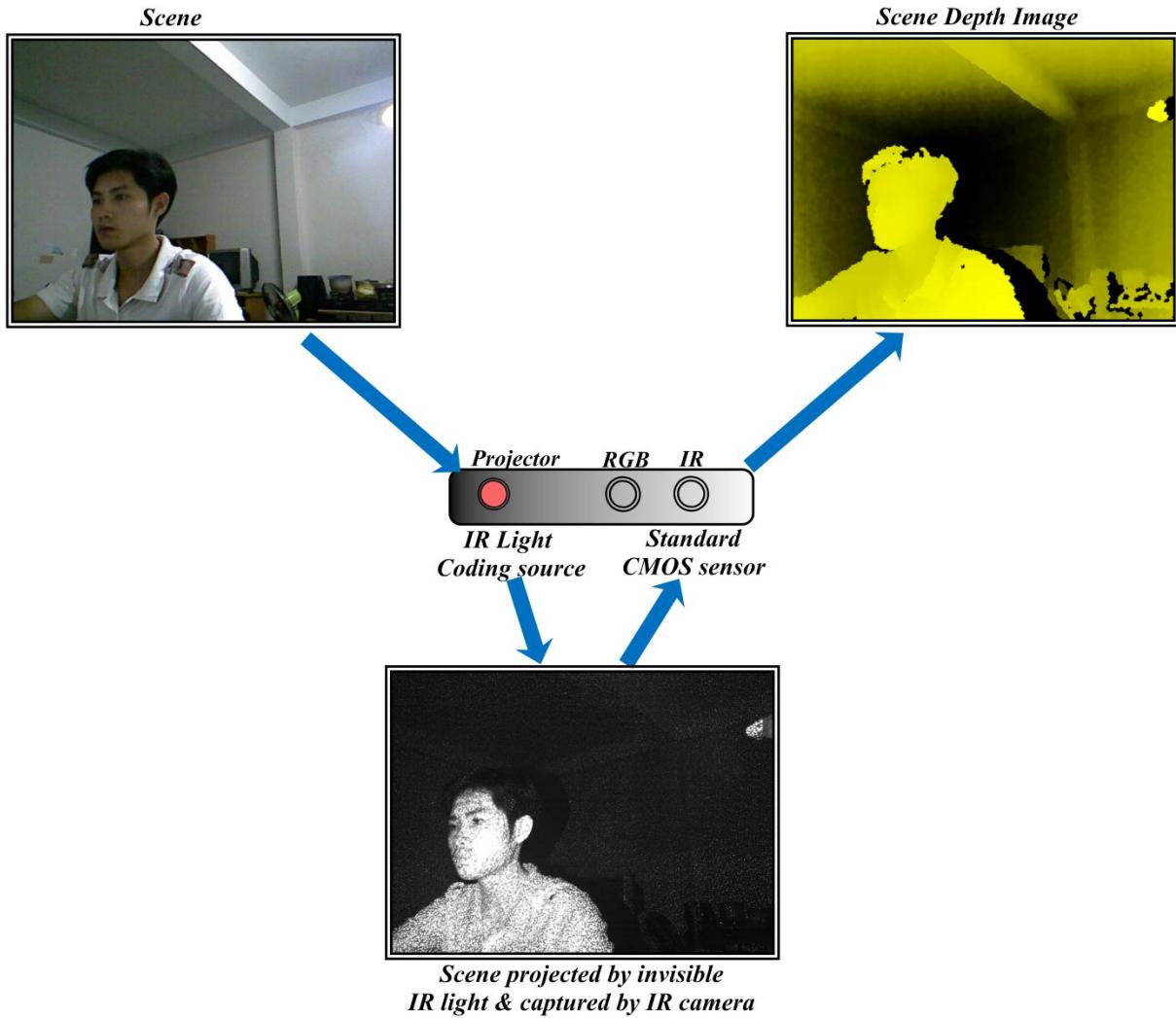
Một trong những đặc tính quan trọng nhất của Kinect đó là thu về giá trị độ sâu hay giá trị khoảng cách tới vật thể trong thế giới thực. Phần tiếp theo sẽ nói về nguyên lý hoạt động của Kinect trong việc tính toán giá trị này.

2.3 Tính toán độ sâu



Hình 2.4: Bên trong Kinect: RGB, IR camera và IR projector

Cặp cảm biến IR camera và IR projector sẽ phối hợp với nhau để cho ra giá trị độ sâu ảnh bằng công nghệ Light Coding của PrimeSense [2].

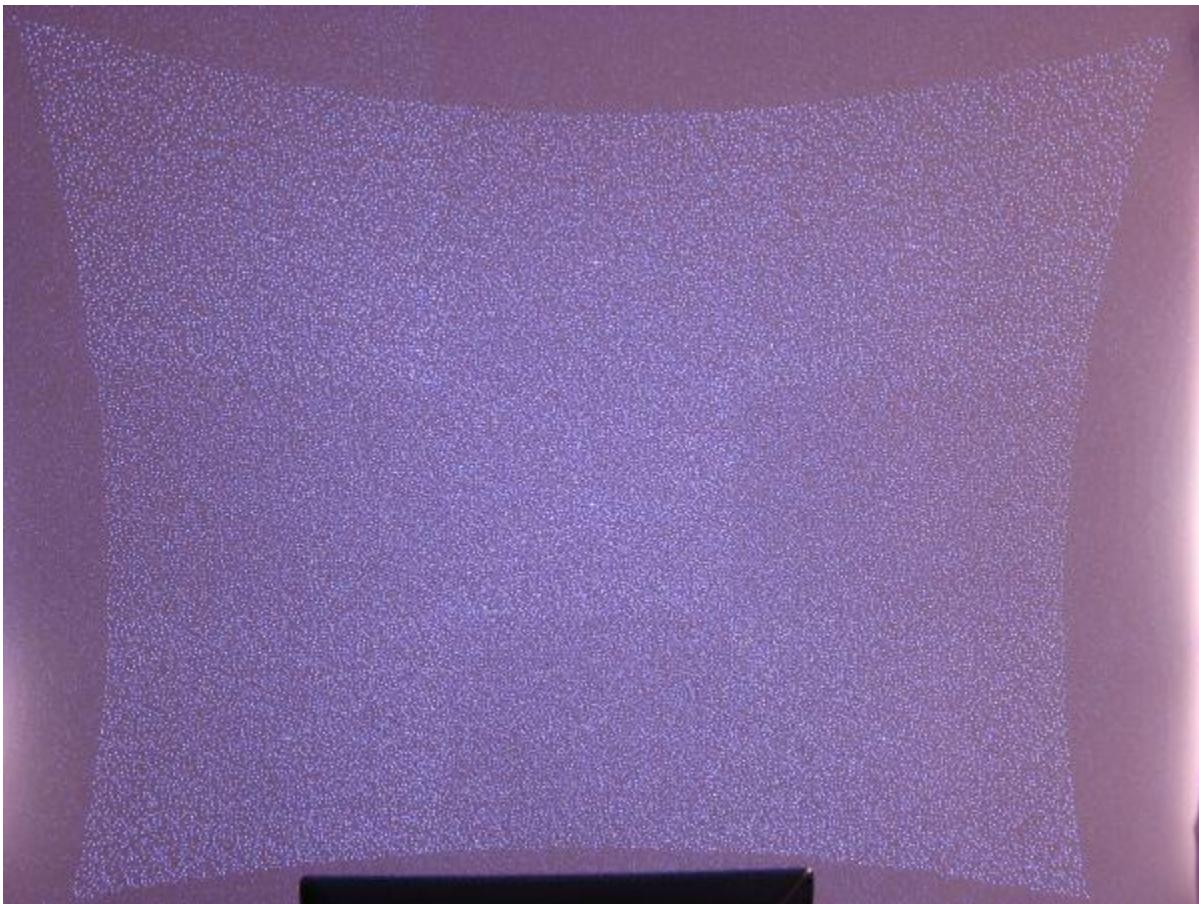


Hình 2.5: Quá trình thu về bản đồ độ sâu ảnh

Khác với kỹ thuật Stereo Camera với việc dùng cặp camera giống nhau để xây dựng nên bản đồ độ sâu, hay kỹ thuật Time-Of-Flight (TOF) định nghĩa khoảng cách bằng ước lượng thời gian di chuyển của tia sáng đi và về trong không gian; kỹ thuật Light Coding dùng một nguồn sáng hồng ngoại chiếu liên tục kết hợp với một camera hồng ngoại để tính toán khoảng cách [3]. Công việc tính toán này được thực hiện bên trong Kinect bằng chip **PS1080 SoC** của PrimeSense. Công nghệ mới này được cho là đáp ứng chính xác hơn, giá cả rẻ hơn cho việc sử dụng ở môi trường trong nhà.

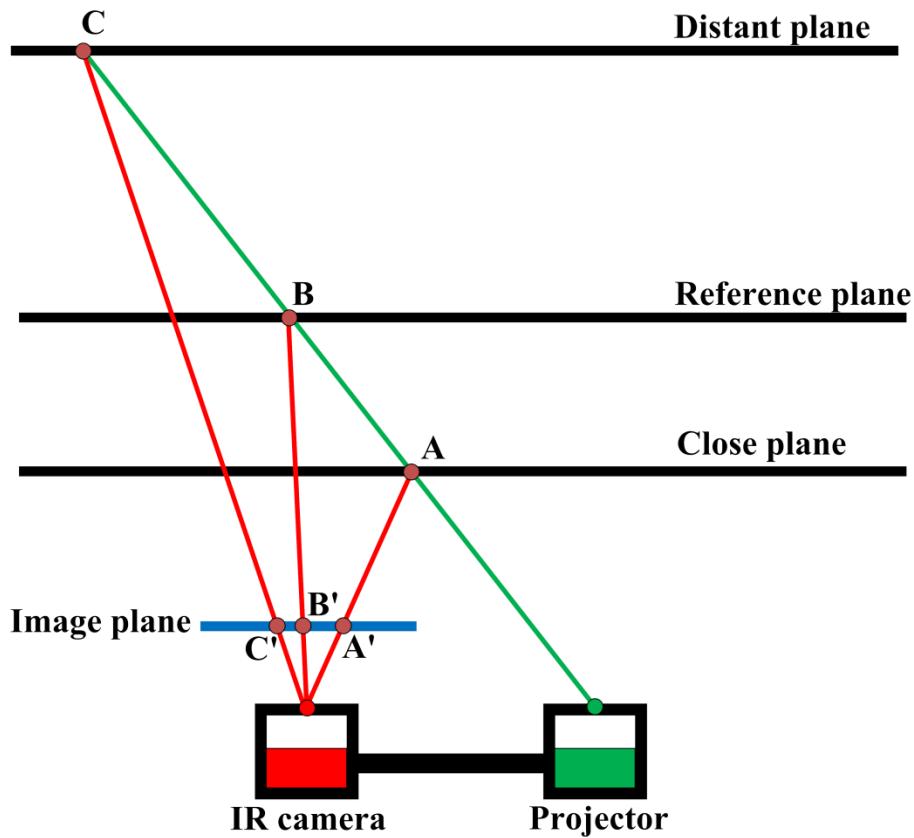
Projector sẽ chiếu một chùm sáng hồng ngoại, tạo nên những đốm sáng ở không gian phía trước Kinect, tập hợp đốm sáng được phát ra này là cố định. Những đốm sáng này được tạo ra nhờ một nguồn sáng truyền qua lưới nhiễu xạ (diffraction

gratings). Tập hợp các đốm sáng này được IR camera chụp lại, thông qua giải thuật đặc biệt được tích hợp trong **PS1080 SoC** [4] cho ra bản đồ độ sâu. Bản chất của giải thuật này là các phép toán hình học dựa trên quan hệ giữa hai cảm biến IR camera và Projector mà ta sẽ đề cập sau. Hình 2.6 cho ta thấy rõ mẫu hình tập hợp các đốm sáng từ Projector và được chụp lại bởi IR camera.



Hình 2.6: Mẫu hình được chiếu bởi projector và chụp lại bằng IR camera

Để hiểu cách thức Kinect ước lượng khoảng cách tới vật thể trong môi trường như thế nào, ta quan sát hình 2.7 trong trường hợp phân tích với một điểm đơn giản.



Hình 2.7: Tính toán khoảng cách tới một điểm chiếu từ Projector [5]

Ta giả sử Projector phát đi một tia sáng dọc đường màu xanh lá, nó sẽ được chụp lại dưới dạng một đốm sáng bởi IR camera khi chạm vào bề mặt vật thể trong không gian. Ta xét ba mặt phẳng ở ba khoảng cách khác nhau: mặt phẳng gần Kinect (close plane), mặt phẳng ở xa Kinect (distant plane) và mặt phẳng tham chiếu (reference plane) ở giữa hai mặt phẳng trên. Trong đó, mặt phẳng tham chiếu ngầm được biết trước bên trong Kinect với đầy đủ thông tin về khoảng cách. Ngoài ra, ta cũng đề cập thêm mặt phẳng ảnh (image plane) của IR camera, là mặt phẳng hình chiếu của các điểm trong không gian thu về bởi IR camera. Ta xét trong ba trường hợp khi tia sáng màu xanh lá chạm vào ba điểm trên ba mặt phẳng lần lượt là A, B, C; ba điểm này được chiếu lên mặt phẳng ảnh tương ứng là A', B', C'. Quan sát vị trí A', B' và C', ta có nhận xét: điểm A càng gần Kinect (hay close plane càng gần Kinect) thì A' càng xa B' về phía bên phải; ngược lại, điểm C càng xa Kinect (hay distant plane càng xa Kinect) thì C' càng xa B' về phía bên trái. Từ đó: khi ta biết trước hướng, điểm xuất

phát của tia sáng từ Projector và vị trí B' là hình chiếu của điểm B trên mặt phẳng tham chiếu lên mặt phẳng ảnh, ta hoàn toàn có thể tính toán được độ sâu ảnh hay khoảng cách tới vật thể.

Kinect làm điều tương tự với tập hợp các đốm sáng còn lại phát đi từ projector, với mặt phẳng tham chiếu biết trước. Nó tìm điểm là tâm của đốm sáng mà IR camera chụp lại được và điểm tương đồng của đốm sáng đó trên mặt phẳng tham chiếu (ví dụ: hình 2.7 ta có A và B, C và B là các cặp điểm tương đồng), để tìm khoảng chênh lệch giữa hai điểm này theo chiều ngang khi chiếu về trên mặt phẳng ảnh; và lưu ý là giá trị chênh lệch này được tính bằng đơn vị pixel. Tập hợp của tất cả các giá trị chênh lệch từ tập hợp đốm sáng, sẽ tạo nên bản đồ độ chênh lệch (disparity map), giá trị này càng lớn thì khoảng cách hay giá trị độ sâu ảnh (depth) càng lớn, từ đó mà ta xây dựng được bản đồ độ sâu (depth map) với giá trị tính bằng mét thực sự. Tuy nhiên, do tập hợp số lượng đốm sáng phát đi từ projector nhỏ hơn so với tổng số pixel trên mặt phẳng ảnh của IR camera nên một phần giá trị độ sâu ảnh còn lại sẽ được nội suy.

Theo tính toán của Nicolas Burrus [6], một trong những người mở đường cho việc tìm hiểu về Kinect qua các thí nghiệm của ông. Ông đã công thức hóa được quan hệ giữa giá trị khoảng cách thật z tính bằng mét và giá trị độ chênh lệch d :

$$z = \frac{1}{-0.0030711016 \times d + 3.3309495161}$$

Trong đó d là con số nguyên biểu diễn dưới dạng 11 bit, tức khoảng thay đổi từ $0 \div 2047$. Với kết quả đo đạc thực nghiệm trên thư viện OpenNI, giá trị z biến thiên trong khoảng $0.5 \div 6.0$ mét và bản đồ độ sâu ổn định trong khoảng $0.5 \div 5.0$ mét. Do đó, giá trị d thực sự biến thiên trong khoảng từ $434 \div 1030$. Như vậy, trong không gian từ $0 \div 0.5$ mét phía trước Kinect, Kinect không thể đưa về bản đồ độ sâu, đây là một nhược điểm sẽ được khắc phục và đề cập đến trong chương 7.

2.4 Một số đặc tính khác

Một số đặc tính khác của Kinect đáng quan tâm: tiêu cự và góc mờ camera (field of view), nguồn cung cấp và công suất tiêu thụ, môi trường hoạt động. Kinect là sản

phẩm thương mại của Microsoft nên các thông số kỹ thuật chi tiết không được công bố. Các thông số được trình bày dưới đây là kết quả đo đạc thực nghiệm:

- **Tiêu cự, góc mở IR camera và RGB camera:**

Hai camera RGB và IR được đặt cách nhau 2.5 cm nên có chút khác nhau ở khung hình thu về từ hai camera. Để đảm bảo khung hình RGB có thể chứa được khung hình IR, người ta thiết kế góc mở của RGB camera lớn hơn. Điều này cũng dẫn đến tiêu cự của RGB camera nhỏ hơn. Các thông số trong bảng 2.1 được đo đạc bằng thực nghiệm:

Feature		RGB camera	IR camera
Field of View (degrees)	Horizontal	~62°	~58°
	Vertical	~48°	~44°
	Diagonal	~72°	~69°
Focal length (pixels)		525	580

Bảng 2.1: Góc mở và tiêu cự của RGB và IR camera [3]

- **Nguồn cung cấp và công suất tiêu thụ:**

Vì Kinect cần nhiều điện năng để hoạt động nên cổng USB của Xbox-360 không thể đáp ứng mà phải qua một cổng chia để chia thành 2 kết nối riêng là USB và kết nối nguồn, giúp cho thiết bị kết nối với Xbox-360 bằng cổng USB trong khi nguồn điện cần cho Kinect là 12VDC được lấy từ adapter. Phiên bản Xbox-360 mới sẽ không cần adapter vì nó có các AUX port đặc biệt để cung cấp cho cổng kết nối. Với kết nối USB ta hoàn toàn có thể cho Kinect giao tiếp với máy tính. Cách thay adapter bằng nguồn pin 12V dùng trên mobile robot được nói thêm ở phần phụ lục 2.



Hình 2.8: Kinect adapter

Công suất tiêu thụ đo bằng thực nghiệm:

Power consumption (idle)	~3.3W
Power consumption (active)	~4.7W

Bảng 2.2: Công suất tiêu thụ trên Kinect [3]

▪ **Môi trường hoạt động:**

Kinect là thiết bị được thiết kế cho việc sử dụng ở môi trường trong nhà (indoor). Ở môi trường ngoài trời, kết quả thử nghiệm cho bản đồ độ sâu không chính xác vào thời điểm ánh sáng mạnh, nhưng cho kết quả chấp nhận được khi ánh sáng yếu (vào thời điểm buổi chiều tối).

Chương 3: Thư viện xử lý ảnh

Nội dung chính

3.1 Thư viện hỗ trợ Kinect

3.2 So sánh Kinect SDK beta và OpenNI

3.3 Point Cloud Library

3.1 Thư viện hỗ trợ Kinect

Ngay khi mới ra đời, Kinect đã được quan tâm bởi rất nhiều nhà phát triển phần mềm, không chỉ trên mảng phát triển game cho Xbox mà còn trên mảng xử lý ảnh ứng dụng trong y học, robot, mapping ... Do đó mà nhiều thư viện được viết cho Kinect ra đời. Cho đến thời điểm hiện tại, các thư viện đáng chú ý là **Libfreenect**, **Code Laboratories Kinect**, **OpenNI** và **Kinect SDK beta**.

- ***Libfreenect:***

Libfreenect [7] là thư viện được phát triển bởi OpenKinect, do một cộng đồng những người quan tâm đến phần cứng Kinect viết ra và chia sẻ. Cộng đồng OpenKinect làm việc hoàn toàn tự nguyện và không vì mục đích lợi nhuận, họ phát triển Libfreenect thành một mã nguồn mở cho các hệ điều hành khác nhau Windows, Linux và OS X. Hiện tại, Libfreenect được đóng gói cho việc sử dụng trên Python, C, C++, C#, Java JNI, Java JNA, Javascript.

- ***Code Laboratories Kinect:***

Code Laboratories (CL) [8] là một công ty về phần mềm chuyên hỗ trợ các nhà phát triển, lập trình viên khai thác các tính năng của các thiết bị xử lý ảnh. Trong số đó Kinect không phải là ngoại lệ, CL cung cấp cho người sử dụng những tính năng cơ bản nhất của Kinect về camera, audio và motor.

- ***OpenNI:***

Thư viện OpenNI [9] được xem là thư viện mạnh nhất trước sự có mặt của Kinect SDK beta, thư viện này hỗ trợ đa ngôn ngữ trên nhiều platform khác nhau, giúp cho các lập trình viên có thể viết các ứng dụng trên Kinect rất dễ dàng với tương tác tự nhiên Natural Interaction (NI). Mục đích chính của OpenNI là xây dựng các hàm API chuẩn, cho phép thư viện có khả năng kết hợp với các middleware nhằm làm tăng sức mạnh cho Kinect.

- ***Kinect SDK beta:***

Kinect SDK [10] beta được Microsoft đưa ra vào ngày 16 tháng 6 năm 2011, là một công cụ lập trình mạnh cho các nhà phát triển. Nó cho phép lập trình viên truy xuất toàn bộ tính năng của thiết bị Kinect. Một điều bất tiện là thư viện này chỉ hỗ trợ

trên công cụ lập trình của Microsoft là Visual Studio 2010 với các ngôn ngữ là C++, C# và Visual Basic. Các tính năng nổi bật như: thu ảnh từ các sensor, skeleton tracking và điều khiển bằng giọng nói thông qua công cụ nhận biết giọng nói, Windows Speech Recognition API. Phiên bản beta hiện tại chỉ cho phép sử dụng vào những mục đích phi lợi nhuận, phiên bản thương mại hứa hẹn sẽ sớm ra mắt trong năm 2012. Phiên bản beta mới nhất được cập nhật vào ngày 1 tháng 11 năm 2011 với nhiều lỗi được sửa và chạy tốt trên Windows 8 Developer Preview.

Tới thời điểm hiện tại, hai thư viện OpenNI và Kinect SDK beta là lựa chọn sáng suốt cho việc lập trình trên Kinect bởi tính năng hỗ trợ mạnh mẽ của hai thư viện này. Mục 3.2 sẽ phân tích và so sánh hai thư viện này và chọn lựa thư viện phù hợp cho đề tài luận văn.

3.2 So sánh Kinect SDK beta và OpenNI

Sau đây là nhận định về ưu, khuyết điểm của hai thư viện trên [11] (lưu ý là Kinect SDK đang trong giai đoạn beta nên sẽ có vài điểm thay đổi tới bản chính thức cuối cùng)

- **Kinect SDK beta:**

- ✓ Ưu điểm:

- + Hỗ trợ xử lý âm thanh.
 - + Hỗ trợ động cơ điều khiển góc ngẳng.
 - + Skeleton tracking (bám đặc tính cơ thể người): không cần hiệu chỉnh trước khi bám, vẫn bám tốt trong trường hợp cơ thể người quay theo nhiều hướng.
 - + Hỗ trợ truy xuất các sensor của Kinect đồng thời.
 - + Việc cài đặt đơn giản.

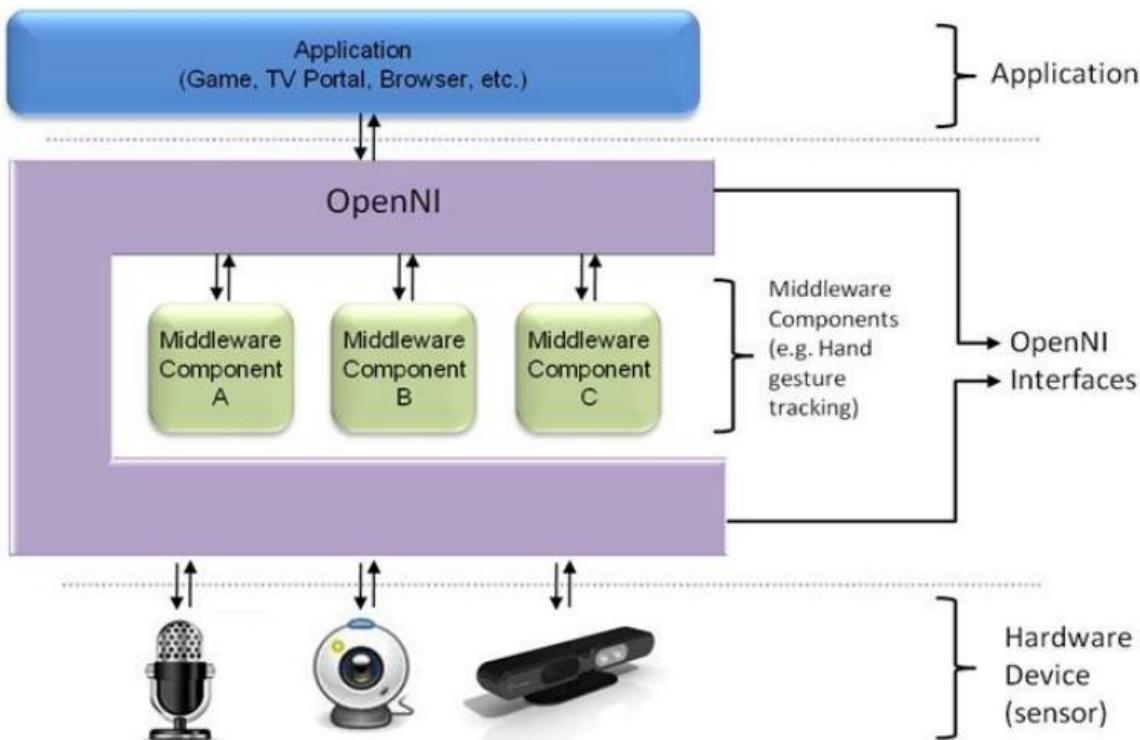
- ✓ Khuyết điểm:

- + Chỉ dùng cho mục đích phi thương mại.
 - + Chỉ hỗ trợ bám toàn thể đặc tính cơ thể người (không có chế độ hand tracking hay bám từng phần cơ thể như trên OpenNI).

- + Không hỗ trợ nhận biết cử chỉ.
 - + Chỉ hỗ trợ trên Win7 (x86 và x64) với đòi hỏi khá cao:
 - Máy tính dual-core, tốc độ 2.66 GHz hoặc nhanh hơn.
 - Windows 7 với hỗ trợ của DirectX 9.0 trở lên.
 - Ram tối thiểu 2 GB.
 - Lập trình trên Visual Studio 2010.
 - + Không hỗ trợ việc thu ảnh trực tiếp từ IR camera.
 - + Vùng Kinect không nhìn thấy trong khoảng $0 \div 0.8$ mét trước Kinect.
- ***OpenNI:***
 - ✓ *Ưu điểm:*
 - + Cho phép xây dựng các ứng dụng thương mại hóa.
 - + Hỗ trợ phát hiện cử chỉ.
 - + Skeleton tracking: hỗ trợ bám từng phần cơ thể người và hand tracking thông qua liên kết với các middleware. Hơn nữa, tiêu thụ công suất của CPU ít hơn so với khi sử dụng Kinect SDK.
 - + Hỗ trợ truy xuất các sensor của Kinect đồng thời.
 - + Hỗ trợ cho Windows, Linux và Mac OSX.
 - + Cho phép truy xuất hình ảnh thu về từ IR camera.
 - + Vùng Kinect không nhìn thấy trong khoảng chấp nhận được là 0.5 mét trước Kinect.
 - ✓ *Khuyết điểm:*
 - + Không hỗ trợ phần xử lý âm thanh cho dãy microphone.
 - + Skeleton tracking: bám đặc tính cơ thể còn nhiều lỗi và chưa được ổn định như trên Kinect SDK.
 - + Không hỗ trợ động cơ điều khiển góc ngang (xử lý vấn đề này bằng chút thủ thuật kết hợp với Code Laboratories Kinect, xem thêm phần phụ lục 1).
 - + Việc cài đặt có phần rắc rối hơn Kinect SDK.

Kết luận: Sinh viên cũng đã có thời gian làm việc trên cả hai thư viện, nên có những kết luận sau:

Kinect SDK mạnh hơn OpenNI ở đặc tính bám cơ thể người ổn định hơn, do đó sẽ đáp ứng chính xác với các cử chỉ cơ thể người; tuy nhiên để phát hiện cử chỉ ta phải tự viết giải thuật trong khi trên OpenNI đã hỗ trợ sẵn. Ngoài ra, Kinect SDK hơn OpenNI ở phần hỗ trợ cho xử lý âm thanh, cho phép xây dựng các ứng dụng điều khiển bằng giọng nói dễ dàng hơn.



Hình 3.1: Thư viện OpenNI phối hợp giữa phần cứng và ứng dụng đầu cuối

OpenNI cho phép thu bản đồ độ sâu trong giới hạn từ 0.5 mét trở về phía trước Kinect trong khi Kinect SDK thì giới hạn này là 0.8 mét về phía trước, chất lượng bản đồ độ sâu thu về trên hai thư viện là như nhau. Kinect SDK là thư viện mới ra đời đang trong giai đoạn beta và cũng chưa có sự hỗ trợ của các thư viện xử lý ảnh khác như Point Cloud. Trong khi đó OpenNI ra đời trước và gần như được tích hợp với thư viện xử lý ảnh Point Cloud. Do đó, với ứng dụng Kinect trong đề tài robot tự hành tránh vật

cần thì OpenNI là lựa chọn tối ưu hơn. Đó cũng chính là lý do sinh viên chọn thư viện OpenNI kết hợp với Point Cloud cho đề tài luận văn này.

Nói đến tầm nhìn xa hơn thì ta thấy: về tính phổ biến, OpenNI có thể phát triển trên các nền tảng hệ điều hành khác nhau và nhận được sự đóng góp của cộng đồng xây dựng mã nguồn mở rộng lớn; về chuyên kinh doanh, ta có thể phát triển thành các sản phẩm thương mại hóa trên thị trường trong rất nhiều lĩnh vực như robot, y tế, giáo dục, giải trí...

Thư viện hỗ trợ Kinect giúp ta lấy về chiều sâu ảnh và một số đặc tính đặc biệt khác (như skeleton tracking, hand tracking, gesture recognition trên OpenNI); để tận dụng tối ưu sức mạnh của nó ta kết hợp cùng thư viện xử lý ảnh trong không gian 3D là Point Cloud mà sẽ được đề cập trong mục 3.3.

3.3 Point Cloud Library



Hình 3.2: Point cloud library logo

PCL [12] là thư viện hỗ trợ cho n-D Point Cloud và cho việc xử lý ảnh trong không gian 3D. Thư viện được xây dựng với nhiều giải thuật như lọc (filtering), khôi phục bề mặt (surface reconstruction), phân vùng (segmentation), ước lượng đặc tính vật (feature estimation), ... PCL có thể dùng trên nhiều platform như Linux, MacOS, Windows và Android. Để đơn giản cho việc phát triển, PCL được chia ra thành nhiều thư viện nhỏ và có thể biên dịch một cách riêng lẻ. Phiên bản mới nhất là PCL 1.3 đưa ra vào ngày 31 tháng 10 năm 2011. PCL hoàn toàn miễn phí cho việc nghiên cứu hay phát triển các sản phẩm thương mại hóa.

Có thể nói PCL là sự kết hợp của nhiều module nhỏ. Những module này thực chất cũng là các thư viện thực hiện các chức năng riêng lẻ trước khi được PCL đóng gói. Các thư viện cơ bản này là:

- **Eigen:** một thư viện mở hỗ trợ cho các phép toán tuyến tính, được dùng trong hầu hết các tính toán toán học của PCL.
- **FLANN:** (Fast Library for Approximate Nearest Neighbors) hỗ trợ cho việc tìm kiếm nhanh các điểm lân cận trong không gian 3D.
- **Boost:** giúp cho việc chia sẻ con trỏ trên tất cả các module và thuật toán trong PCL để tránh việc sao chép trùng lặp dữ liệu đã được lấy về trong hệ thống.
- **VTK:** (Visualization Toolkit) hỗ trợ cho nhiều platform trong việc thu về dữ liệu 3D, hỗ trợ việc hiển thị, ước lượng thể tích vật thể.
- **CMinPack:** một thư viện mở giúp cho việc giải quyết các phép toán tuyến tính và không tuyến tính.

Chương 4: Phát hiện vật cản

Nội dung chính

4.1 Các phương pháp phát hiện vật cản không sử dụng camera

4.1.1 Dùng công tắc hành trình

4.1.2 Dùng cảm biến siêu âm

4.2 Các phương pháp phát hiện vật cản sử dụng camera

4.2.1 Xử lý ảnh với một camera (Monocular vision)

4.2.2 Xử lý ảnh với hai camera (Stereo vision)

4.3 Phát hiện vật cản sử dụng Kinect

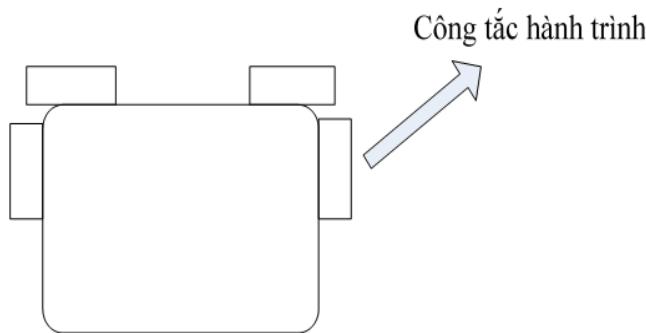
Vấn đề phát hiện và tránh vật cản là điều luôn được đề cập tới cho robot tự hành. Cùng với sự phát triển của công nghệ, ngày càng có nhiều cảm biến tích hợp giúp robot làm được điều này đơn giản và chính xác hơn. Trước đây, các loại cảm biến siêu âm được dùng rộng rãi; gần đây, người ta quan tâm đến áp dụng công nghệ xử lý ảnh nhiều hơn. Sau đây, ta điểm qua một số phương pháp phát hiện và tránh vật cản; nhận định ưu, khuyết điểm và lựa chọn phương pháp tối ưu nhất.

4.1 Các phương pháp phát hiện vật cản không sử dụng camera

4.1.1 Dùng công tắc hành trình

Đây là phương pháp đơn giản nhất và ít tốn kém.

- **Hoạt động:** công tắc hành trình được gắn cho robot như hình 4.1, khi robot di chuyển, nếu chạm chướng ngại vật, tác động công tắc, robot sẽ nhận ra và rẽ hướng khác.



Hình 4.1: Mô hình robot dùng công tắc hành trình

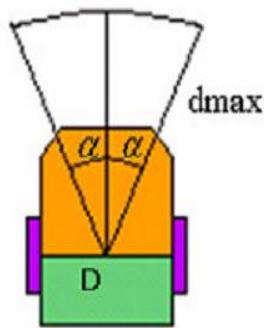
- **Ưu điểm:** đơn giản, dễ thực hiện, chi phí thấp.
- **Khuyết điểm:** robot phải va chạm mới phát hiện được chướng ngại vật. Va chạm nhiều lần sẽ dễ gây hư hỏng và ảnh hưởng đến những gì robot đang tải.

4.1.2 Dùng cảm biến siêu âm [13]

Đây là một phương pháp khá hiệu quả và thông dụng.

- **Hoạt động:** Cảm biến sẽ phát ra sóng siêu âm với góc mở nhất định. Khi đó, nếu trong tầm quét của nó phát hiện chướng ngại vật thì sóng siêu âm sẽ phản hồi lại. Ta có thể đo khoảng cách bằng cách tính thời gian từ lúc sóng siêu âm phát ra đến

lúc thu sóng về, sau đó kết hợp với vận tốc sóng siêu âm (khoảng 343 m/s) để biết được quãng đường mà sóng đã đi.

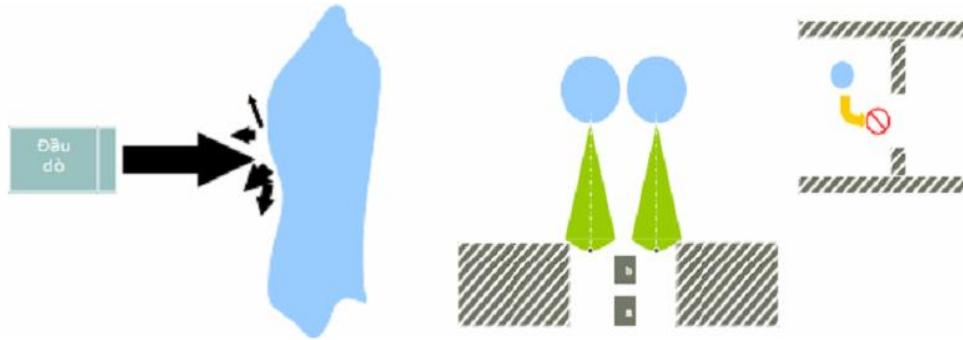


Hình 4.2: Cảm biến siêu âm

Các cảm biến được đặt lệch một góc α , khoảng cách lớn nhất (tính từ D) mà các cảm biến có thể nhận diện được là d_{max} ; d_{max} và α phải đảm bảo cho cảm biến có vùng kiểm tra đủ rộng để khi tiến thẳng robot có thể nhận diện được vật cản.

- **Ưu điểm:** xử lý nhanh, kết quả tương đối chính xác.
- **Khuyết điểm:**
 - + Cảm biến siêu âm chỉ nhận biết được vật cản khi mặt phẳng quét của cảm biến cắt ngang vật cản, do đó nó sẽ không phát hiện ra những vật cản nhỏ, thấp và nằm sát mặt đất.
 - + Do sử dụng sóng siêu âm và sự phản xạ của nó để tính khoảng cách và phát hiện vật cản nên giải thuật điều khiển khá phức tạp và phát sinh một số trường hợp sai số khó khắc phục: sai số lặp, hiện tượng Forecasting, hiện tượng đọc chéo (Crosstalk).
 - **Sai số lặp:** sai số lặp là sai số luôn xảy ra với tất cả các thiết bị đo lường, trong đó có cả cảm biến siêu âm.
 - **Hiện tượng forecasting:** hiện tượng Forecasting là hiện tượng phản xạ góc sai lệch của cảm biến. Theo nguyên lý TOF, để có khoảng cách đúng, cảm biến siêu âm phải hướng vuông góc với bề mặt chướng ngại vật cần đo. Tuy nhiên, các chướng ngại vật không bao giờ là phẳng, mịn nên tia phản

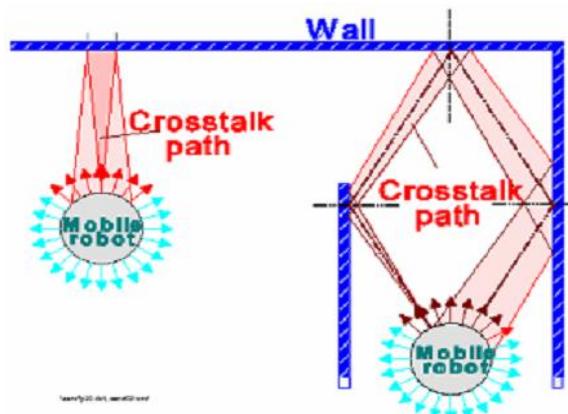
xạ có thể không tương ứng với góc tới. Các chùm tia phản xạ này có năng lượng phản xạ thấp hơn. Tuy vậy, ở một khoảng cách nào đó, cảm biến siêu âm vẫn có thể ghi nhận được những tín hiệu phản xạ này. Kết quả là thông số đọc về từ cảm biến siêu âm bị lệch do góc mở của cảm biến siêu âm lớn.



Hình 4.3: Hiện tượng Forecasting

Ngoài ra, vì góc mở rộng nên không chỉ sai về nhận dạng vị trí chướng ngại vật mà khoảng cách ghi nhận cũng bị sai lệch. Tuy nhiên sai số này không đáng kể như sai số do hiện tượng đọc chéo gây ra.

- **Hiện tượng crosstalk:** hiện tượng đọc chéo (Crosstalk) là hiện tượng mà cảm biến siêu âm này ghi nhận tín hiệu phản xạ hoặc trực tiếp từ cảm biến siêu âm khác; hoặc sau quá trình sóng siêu âm truyền đi và phản xạ qua các bề mặt nó quay lại cảm biến theo một cách không mong muốn.



Hình 4.4: Hiện tượng Crosstalk

4.2 Các phương pháp phát hiện vật cản sử dụng camera

4.2.1 Xử lý ảnh với một camera (Monocular vision)

Phương pháp này chủ yếu dựa trên các phân tích màu sắc hay sự thay đổi do chuyển động của các khung hình liên tiếp nhau; kỹ thuật xử lý ảnh đơn giản nhưng chỉ hiệu quả trong một số môi trường nhất định. Sau đây là một số phương pháp thường được sử dụng: *optical flow, edge detection, floor finder technique*.

- *Optical Flow:*

- + *Phương pháp:* là phương pháp tránh chướng ngại vật nhờ vào quan sát sự di chuyển của phần tử ảnh. Giải thuật sẽ tìm những phần tử ảnh đặc biệt trong ảnh tại một frame nào đó và quan sát độ dịch chuyển của nó ở frame tiếp theo. Vật càng gần, độ dịch chuyển càng lớn.

Phương pháp này chia thị trường của robot ra làm 2 phần trái, phải. Dựa vào giá trị optical flow tính được từ 2 vùng, ta sẽ biết được chướng ngại vật đang ở phía nào của robot và ra lệnh điều khiển robot tránh về hướng làm giảm giá trị optical flow.



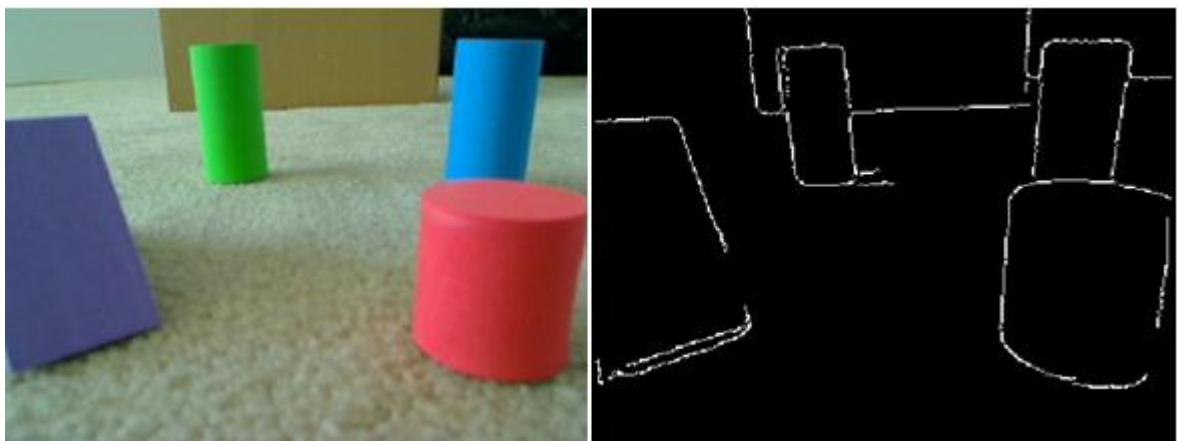
Hình 4.5: Thị trường của robot với optical flow

- + *Ưu điểm:* Phát hiện chướng ngại vật không phụ thuộc vào hình dạng vật.
- + *Khuyết điểm:*

- Chỉ tối ưu đối với những vật có góc cạnh, có nhiều điểm đặc biệt. Giả sử nếu gặp một bức tường trắng, robot không thể phân biệt được phần tử nào là phần tử tương ứng khi xét từ frame này sang frame khác.
- Dễ bị nhiễu: nếu nền có hoa văn hay đường viền sẽ gây nhiễu do camera sẽ bám theo các phần tử đặc biệt trên nền. Ngoài ra, những vật trong thị trường của robot phải đứng yên, nếu có vật chuyển động ở xa nhưng vẫn trong thị trường của robot sẽ làm tăng optical flow khiến robot nhầm lẫn đang có vật cản ở trước mặt.
- Tốc độ xử lý chậm do yêu cầu tính toán nặng.

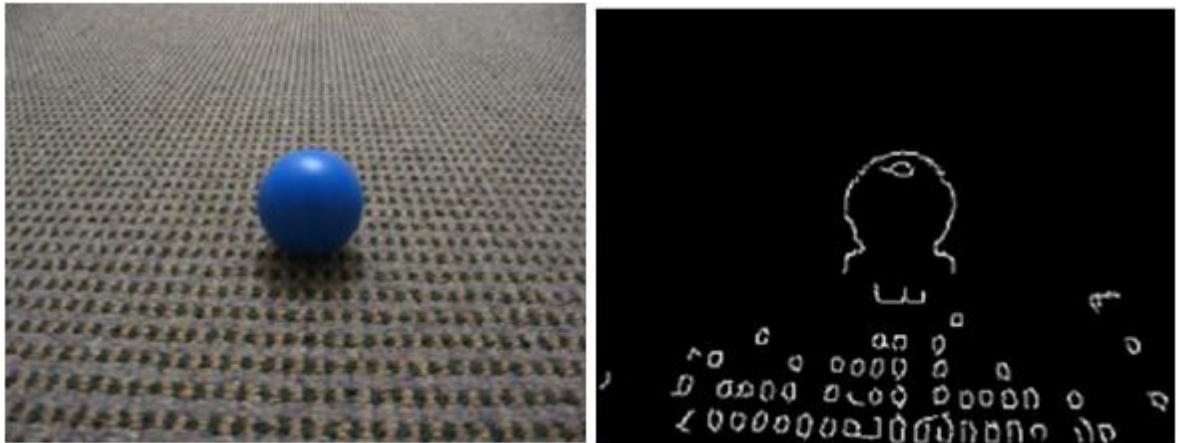
▪ **Edge Detection [14]:**

- + Phương pháp: phương pháp này dùng những kỹ thuật tách biên (như Canny) cho ta ảnh chỉ hiển thị đường biên của vật thể như hình 4.6. Từ đó giúp ta phân biệt được nền và các vật cản. Vật cản sẽ là những vật có viền bao quanh, còn nền là vùng không gian còn lại.



Hình 4.6: Ảnh gốc và ảnh sau khi tách biên

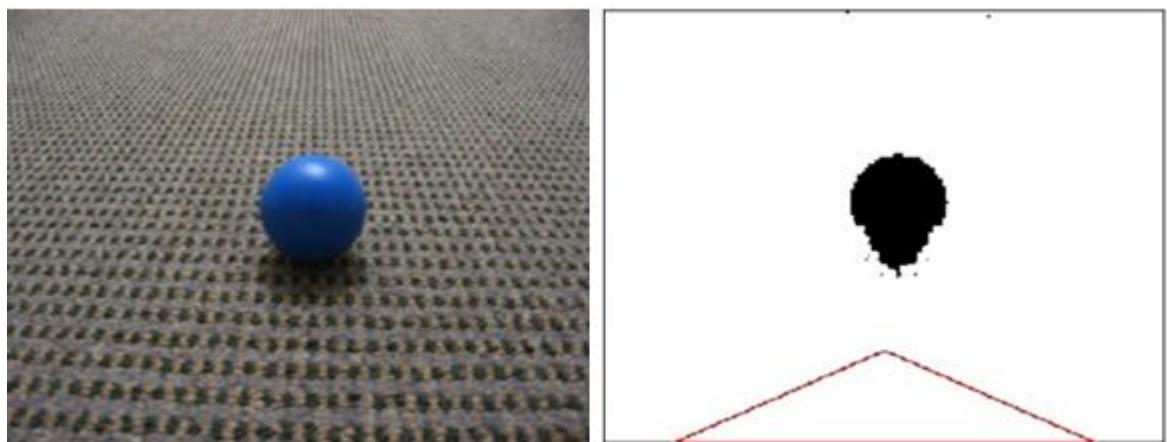
- + *Ưu điểm:* xử lý nhanh, dò tìm ra vật cản tốt, chính xác.
- + *Khuyết điểm:* chỉ hoạt động tốt trong điều kiện nền đơn sắc và không có hoa văn hay họa tiết.



Hình 4.7: Hạn chế của phương pháp dò biên

▪ **Floor Finder Technique [14]:**

- + *Phương pháp:* phương pháp này dựa trên màu sắc của các điểm ảnh, những điểm ảnh không trùng màu với màu nền thì được xem là vật cản. Ta giả định vùng không gian nhỏ trước mặt robot là không có vật cản. Bằng cách lấy giá trị màu sắc các điểm ảnh trong vùng này, ta được một tập mẫu màu sắc của nền. So sánh giá trị màu của từng điểm ảnh còn lại trong hình với tập mẫu này ta sẽ xác định được điểm ảnh nào thuộc về nền, điểm ảnh nào thuộc về vật cản.



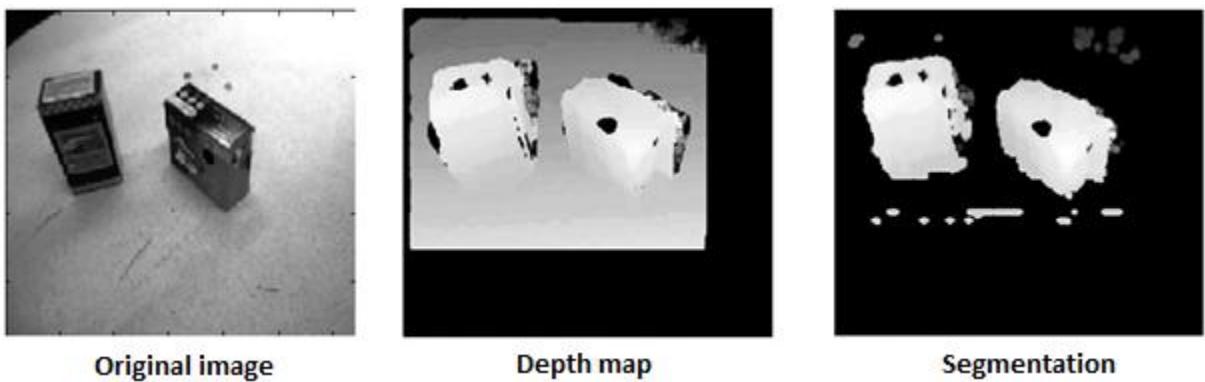
Hình 4.8: Phương pháp dò nền

- + *Ưu điểm:* đơn giản, hiệu quả, phát hiện vật cản chính xác, không phụ thuộc vào hình dạng và kích thước của vật cản. Đồng thời, tốc độ xử lý nhanh do yêu cầu tính toán không nhiều.

- + *Khuyết điểm:* như các phương pháp dùng xử lý ảnh khác, do sử dụng màu sắc để nhận biết nên dễ nhầm lẫn giữa bóng đèn trên sàn và vật cản do bóng có màu sắc khác với nền. Bên cạnh đó, nếu vật có cùng màu với nền thì phương pháp này không đạt hiệu quả cao.

4.2.2 Xử lý ảnh với hai camera (Stereo vision)

Hạn chế của phương pháp sử dụng một camera đến từ sự ảnh hưởng của màu sắc hay họa tiết môi trường phức tạp. Đó cũng là hạn chế chung của xử lý ảnh trong không gian 2D cho robot tự hành tránh vật cản. Bằng việc sử dụng từ hai camera trở lên ta thu được đầy đủ hơn thông tin từ môi trường, nhờ vậy mà những hạn chế trên được khắc phục. Thông tin thêm ở đây là giá trị khoảng cách tới vật, lúc này công việc xử lý ảnh phức tạp hơn do phải làm việc trong không gian 3D.



Hình 4.9: Phương pháp Stereo Vision

- + *Phương pháp:* ảnh từ thế giới thực được thu về thông qua hai camera, sau đó qua khâu hiệu chỉnh, khắc phục méo trước khi cho ra bản đồ độ sâu. Công việc này được thực hiện bằng các giải thuật đặc biệt với sự hỗ trợ của thư viện OpenCV. Ta thu được đầy đủ thông tin của vật cản trong môi trường như chiều cao, bê rộng hay khoảng cách từ camera tới vật sau khâu khôi phục 3D (3D reconstruction).
- + *Ưu điểm :* phát hiện vật cản chính xác, hiệu quả, không phụ thuộc hình dạng, kích thước hay màu sắc vật, có thể phát hiện các vật thể trong không trung.
- + *Khuyết điểm:* xử lý phức tạp, đòi hỏi sự chính xác cao trong khâu hiệu chỉnh. Tính toán khá nặng nên yêu cầu bộ vi xử lý cao.

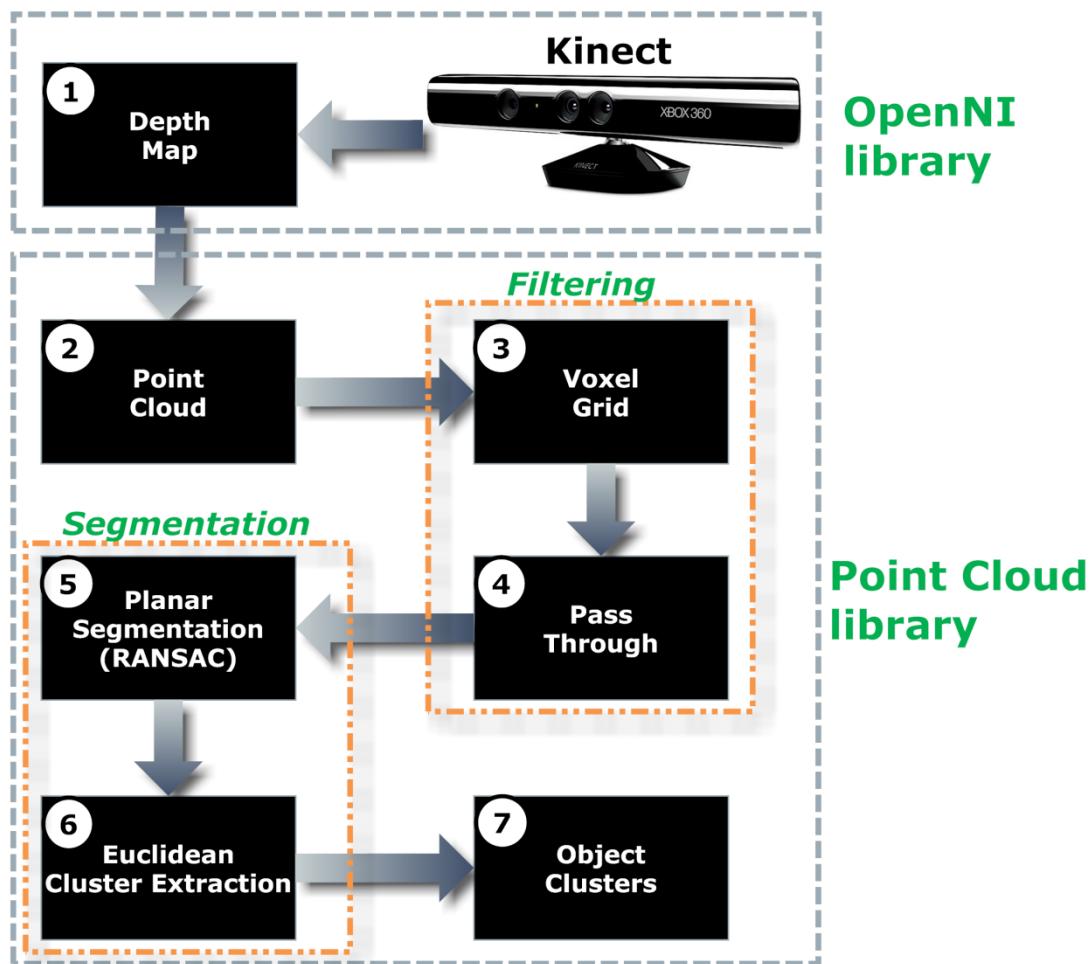
Nhận xét:

Với sự phát triển của công nghệ hiện nay thì tốc độ xử lý không còn là vấn đề khó khăn nữa; với mục đích tránh vật cản một cách tốt nhất thì phương pháp cuối cùng, “Stereo Vision” nổi trội hơn cả.

Sinh viên đã có thời gian thực hiện và hoàn thành việc phát hiện vật cản bằng phương pháp stereo vision trên hai camera. Kết quả phụ thuộc ở khâu hiệu chỉnh ban đầu rất nhiều. Vì thế, sinh viên tìm đến thiết bị có khả năng hỗ trợ việc lấy bản đồ độ sâu trực tiếp, mà không bận tâm nhiều đến việc hiệu chỉnh cho camera, đó là thiết bị chơi game Kinect.

Kinect cùng mục đích với phương pháp stereo vision là thu về bản đồ độ sâu nhưng cách thực hiện lại có chút khác biệt, dù sử dụng giải thuật tính toán tương tự nhau. Điều này đã được trình bày ở mục 2.3. Công nghệ mà Kinect sử dụng được xem như sự giao thoa giữa stereo vision và range finder (phương pháp đo đặc khoảng cách bằng sóng như laser, hồng ngoại hay sóng siêu âm). Kết quả thu về trên Kinect chính xác hơn, ổn định hơn, tiêu tốn tài nguyên máy tính ít hơn nhiều so với việc sử dụng hai camera trong phương pháp stereo vision. Do đây là một công nghệ mới nên sinh viên gọi là phương pháp xử lý ảnh trên Kinect và được trình bày chi tiết trong mục 4.3.

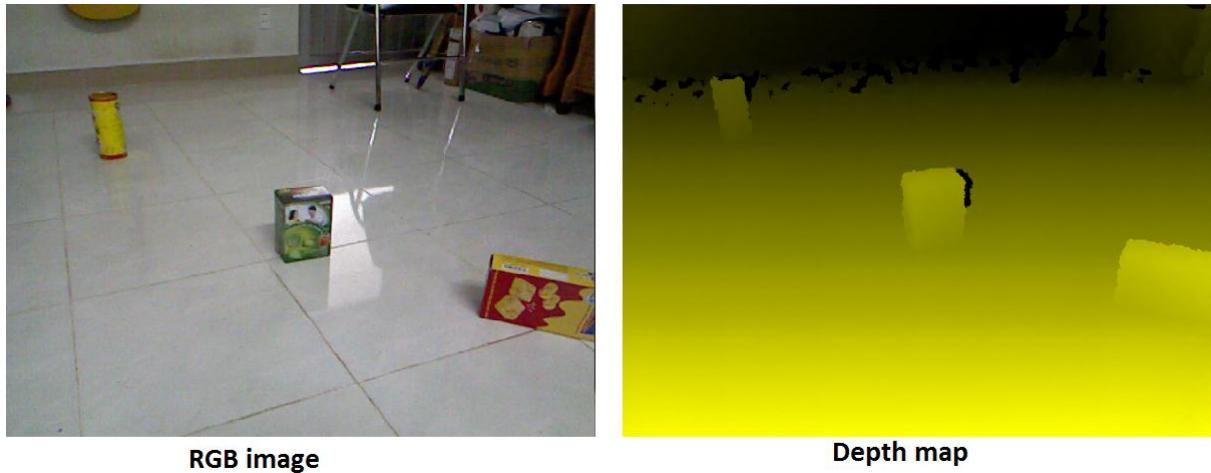
4.3 Phát hiện vật cản sử dụng Kinect



Hình 4.10: Sơ đồ xử lý: phát hiện và tách vật cản

Hình 4.10 liệt kê tất cả các khối chính trong sơ đồ phát hiện và tách vật cản sử dụng Kinect. Khối số 1 làm nhiệm vụ thu về bản đồ độ sâu, công việc này cần đến thư viện OpenNI. Các khối còn lại từ 2÷7 làm nhiệm vụ xử lý bản đồ độ sâu và cho ta đầy đủ thông tin về vật cản, công việc này cần đến thư viện Point Cloud (PCL). OpenNI và PCL được cộng đồng mã nguồn mở xây dựng mối liên hệ khá mật thiết với nhau, nên hạn chế được các xung đột có thể xảy ra như làm việc PCL cùng với các thư viện khác. Bây giờ ta tìm hiểu cách thức hoạt động từng khối trong sơ đồ trên.

▪ **Depth map:**



Hình 4.11: Ảnh RGB và bản đồ độ sâu

Depth map hay bản đồ độ sâu chứa thông tin vị trí vật trong không gian phía trước Kinect. Bản đồ độ sâu được OpenNI hỗ trợ ở các độ phân giải và tốc độ như sau:

- SXGA_15Hz: độ phân giải 1280×1024 , tốc độ 15 fps.
- VGA_30Hz: độ phân giải 640×480 , tốc độ 30 fps.
- QVGA_30Hz: độ phân giải 320×240 , tốc độ 30 fps.

Việc lựa chọn độ phân giải càng nhỏ thì tốc độ xử lý khi qua thư viện Point Cloud càng được tăng lên.

- **Point cloud:**



Hình 4.12: RGB point cloud

Point cloud \mathbf{P} là một tập hợp các phân tử \mathbf{p}_i , mỗi phân tử này sẽ chứa tập các giá trị biểu diễn không gian nD (thường thì $n = 3$) [15]:

$$\mathbf{P} = \{p_1, p_2, \dots, p_i, \dots, p_n\}, p_i = \{x_i, y_i, z_i\}$$

Bên cạnh thông tin dữ liệu là XYZ, mỗi điểm \mathbf{p}_i còn có thể chứa thêm các thông tin khác như: RGB colors, intensity values, ...

Mỗi giá trị x_i , y_i , z_i được lưu trữ bằng kiểu **float32**. Nếu ta chọn độ phân giải VGA thì ta có: $n = 640 \times 480 = 307200$ phân tử \mathbf{p} , đây là một con số khá lớn. Vì vậy:

- Xử lý chậm (dữ liệu lớn dẫn đến việc tính toán sẽ lâu hơn).
- Dữ liệu kiểu float 32 bit nên cần không gian lưu trữ lớn (cần RAM và HDD lớn).

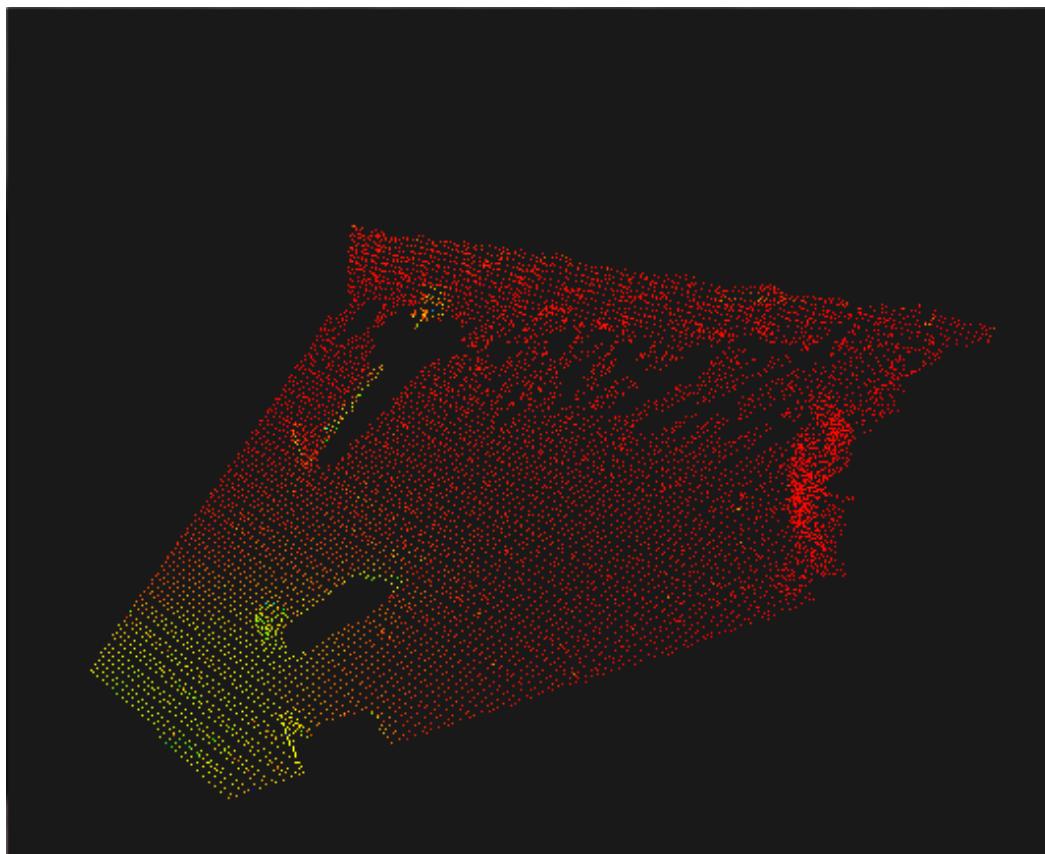
Máy tính sinh viên sử dụng có cấu hình như sau:

- **CPU:** Core i5-2430, 2.4 GHz
- **RAM:** 4 GB
- **VGA:** GeForce GT540 1GB

Tốc độ thu về point cloud đo được ở bước này vào khoảng $13 \div 15$ fps.

Chính vì hạn chế này mà ta cần phải thực hiện các bước lọc tiền xử lý để đáp ứng được yêu cầu xử lý thời gian thực. Công việc này được PCL định nghĩa là “*downsampling*” và “*removing points*”. Hai bước *downsampling* và *removing points* được sử dụng là **Voxel Grid** và **Pass Through**.

- **Voxel grid:**

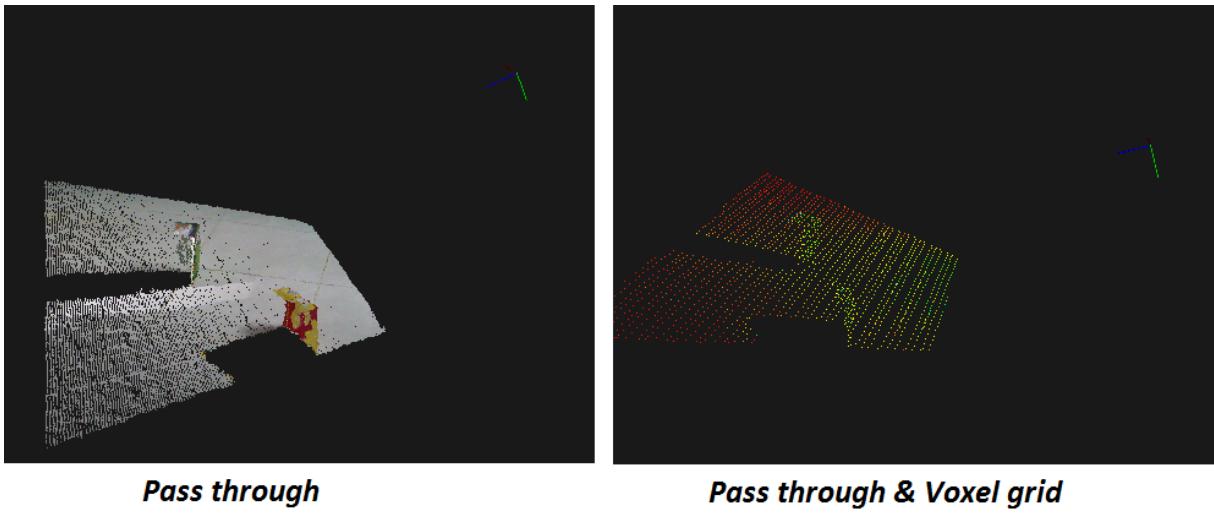


Hình 4.13: Voxel grid

Voxel grid làm giảm mật độ số điểm xuống; tập hợp các điểm quá gần nhau sẽ chỉ cần một điểm đại diện. Ta chọn giá trị mật độ phù hợp mà vẫn đảm bảo sát rõ hình dạng vật thể. Mật độ ta chọn ở đây là 3 centimet theo ba chiều X, Y và Z.

Hàm hỗ trợ: **setLeafSize(0.03f, 0.03f, 0.03f);**

- ***Pass through:***



Hình 4.14: Pass through không có (trái) và có voxel grid (phải)

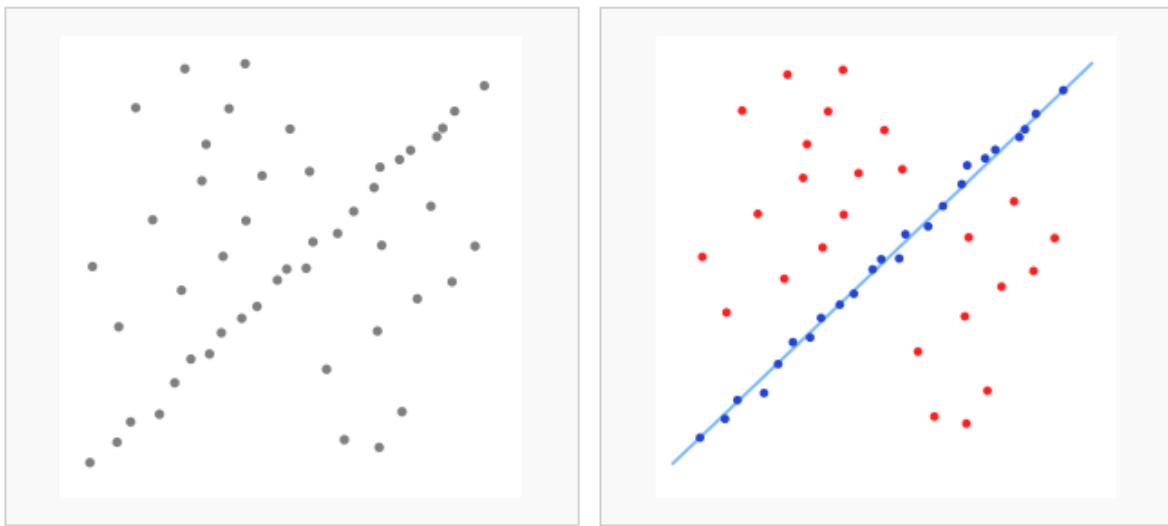
Pass through sẽ giới hạn không gian của point cloud theo các chiều X, Y và Z. Ở đây ta chỉ giới hạn theo chiều Z. Như đã phân tích trong mục 2.3, giá trị khoảng cách theo chiều Z mà Kinect có thể nhìn thấy khoảng $0.5 \div 5.0$ mét, đây là tầm nhìn không cần thiết cho mobile robot, việc giới hạn lại sẽ giúp cho PCL xử lý nhanh hơn. Vì thế, ta chỉ cần cho Kinect nhìn trong khoảng $0.5 \div 1.4$ mét, là tầm quan sát đủ cho ứng dụng robot tránh vật cản. Tốc độ thu hình sau khi kết hợp pass through và voxel grid (Hình 4.14 (phải)) đạt giá trị 30 fps, tức đạt tốc độ tối đa.

```
Hàm hỗ trợ: setFilterFieldName ("z");  
setFilterLimits(0.5, 1.4);
```

- ***Plannar segmentation:***

Plannar segmentation sẽ tách các point cloud có cấu trúc phẳng, sau đó tách ra point cloud có tổng số điểm lớn nhất, bằng giải thuật RANSAC (RANdom SAmple Consensus). Đây là giải thuật ước lượng một mô hình toán học từ một tập hợp các điểm có chứa nhiễu (outliers). Giải thuật này lần đầu được công bố vào năm 1981 bởi Fischler và Bolles [16].

Hình 4.15 cho ta thấy ứng dụng thuật toán RANSAC tìm mô hình đường thẳng (có dạng $ax + by + c = 0$) trong một tập hợp các điểm có chứa nhiễu.



A data set with many outliers for which a line has to be fitted.

Fitted line with RANSAC, outliers have no influence on the result.

Hình 4.15: Tìm mô hình đường thẳng bằng thuật toán RANSAC

Giải thuật tìm đường thẳng bằng thuật toán RANSAC được mô tả như sau:

- **Đầu vào:**
 - `data` - tập hợp các điểm
 - `k` - số lần lặp
 - `t` - ngưỡng (threshold) sai số để xác định điểm nào đó có khớp mô hình không
 - **Đầu ra:**

<code>best_model</code>	- mô hình tốt nhất
<code>best_consensus_set</code>	- tập hợp các điểm khớp với <code>best_model</code>
<code>best_model</code>	= null
<code>best_consensus_set</code>	= null
<code>best_num_points</code>	= 0
 - **Lặp k lần:**

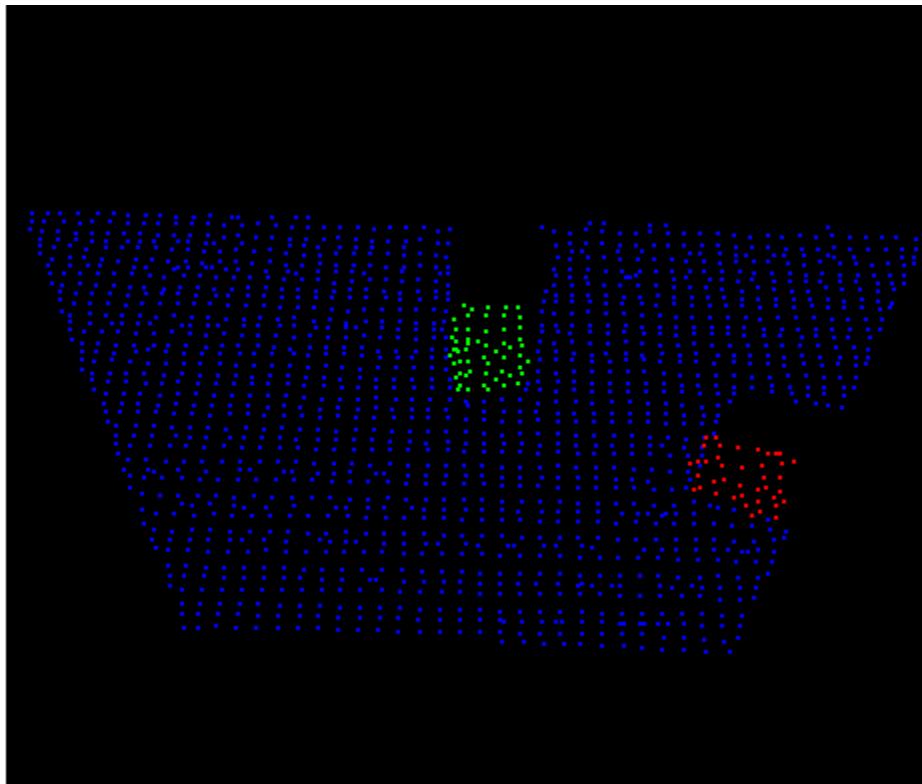
<code>consensus_set</code>	= tập hợp 2 điểm ngẫu nhiên thuộc <code>data</code>
<code>model</code>	= mô hình đường thẳng suy ra từ 2 điểm trên
- Với mỗi điểm thuộc `data` nhưng không thuộc `consensus_set`, ta xét:
- `distance` = khoảng cách từ điểm đến đường thẳng;
if `distance` < `t` (điểm thuộc mô hình nếu sai số nhỏ hơn mức ngưỡng đặt trước)
thêm điểm đó vào `consensus_set`
- `num_points` = số lượng phần tử trong `consensus_set`
if `num_points` > `best_num_points`

<code>best_model</code>	= <code>model</code>
<code>best_consensus_set</code>	= <code>consensus_set</code>
<code>best_num_points</code>	= <code>num_points</code>
- **Trả về giá trị:** `best_model` và `best_consensus_set`

Với mô hình mặt phẳng α có dạng $ax + by + cz + d = 0$, thuật toán RANSAC cũng làm được điều tương tự như đối với mô hình đường thẳng, thay vì chọn hai điểm bất kỳ để tìm mô hình thì mặt phẳng cần ba điểm. Quan sát hình 4.16, point cloud có màu xanh dương là mặt phẳng nền nhà, được tìm ra nhờ thuật toán RANSAC.

Hàm hỗ trợ: `setModelType;`
`setMethodType;`
`setDistanceThreshold(0.02); //threshold = 2cm`

- *Euclidean cluster extraction:*



Hình 4.16: Point cloud sau khi thực hiện xong bước lọc (filtering) và phân đoạn (segmentation)

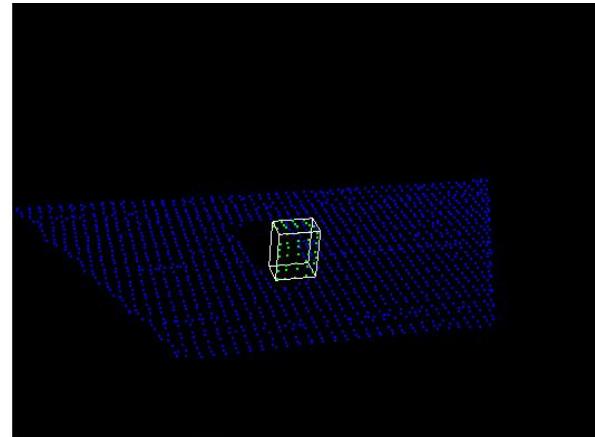
Euclidean cluster extraction làm công việc tách các point cloud có mặt trên nền nhà, tập hợp các điểm gần nhau sẽ được nhóm lại thành một point cloud hay cluster, mỗi cluster đại diện một vật thể. Hình 4.16 cho ta hai cluster: cluster màu đỏ (cluster gần Kinect) và cluster màu xanh lá (cluster xa Kinect) trên mặt phẳng nền nhà (point cloud màu xanh dương).

```
Hàm hỗ trợ: setInputCloud(cloud_filtered);  
setIndices(inliers);  
filter(*cloud_cluster);
```

- *Object clusters:*



RGB image



Object cluster

Hình 4.17: Object cluster

Object clusters là các vật cản mà ta có được sau bước Euclidean cluster extraction. Bước này làm nhiệm vụ phân tích đặc tính vật cản về kích thước cũng như vị trí vật trong không gian phía trước Kinect. Đây là những thông tin cần thiết cho kế hoạch tránh vật cản của mobile robot. Công việc này được thực hiện trên từng cluster, và đã được hỗ trợ hàm cài đặt bởi PCL.

```
Hàm hỗ trợ: getMinMax3D(&cloud_cluster, min_point, max_point);
```

Chương 5: Module điều khiển động cơ

Nội dung chính

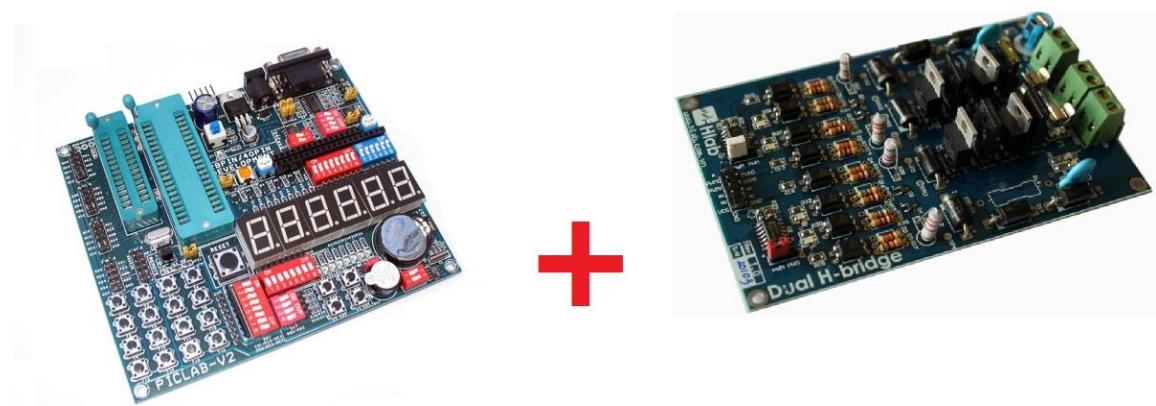
5.1 PIC 18F4550

5.1.1 Giới thiệu chung

5.1.2 Những module chính sử dụng trong luận văn

5.2 Mạch công suất (mạch cầu H)

Module điều khiển động cơ bao gồm: board mạch phát triển PIC microcontroller PICLAB-V2 [17] của Thiên Minh và board mạch cầu H kéo đồng thời hai động cơ của HLAB. Vi điều khiển chính được chọn là PIC18F4550.

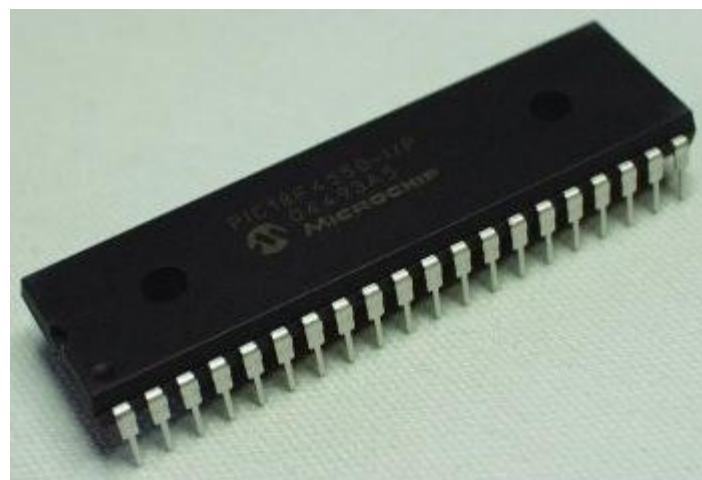


Hình 5.1: PICLAB-V2 (trái) và board mạch cầu H (phải)

5.1 PIC 18F4550

5.1.1 Giới thiệu chung

PIC18F4550 là một vi xử lý cơ bản đa chức năng và rẻ. Nó là sản phẩm của họ vi xử lý PIC thông dụng của công ty Microchip của Mỹ có trụ sở đặt tại Chandler, Arizona (Mỹ).

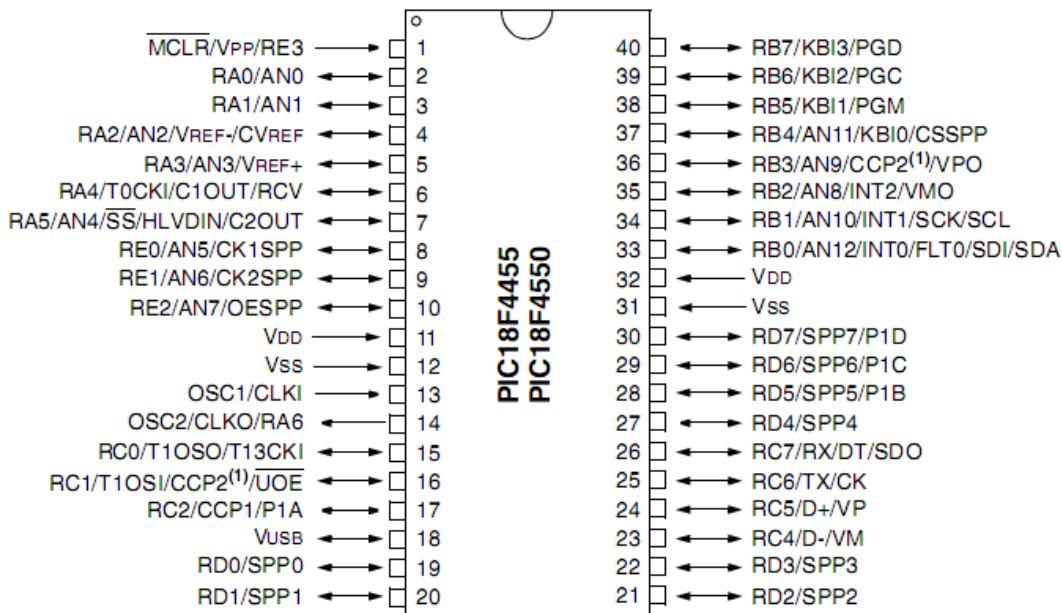


Hình 5.2: PIC 18F4550

PIC 18F4550 dùng bộ nhớ Flash, có thể ghi nhiều lần, dung lượng lớn đáp ứng được hầu hết các ứng dụng trong thực tế:

- Điện áp hoạt động rộng từ 2V đến 5.5V.
- Bộ nhớ chương trình Flash 32K ô nhớ cho phép ghi 100,000 lần. Bộ nhớ dữ liệu RAM có 2048 Bytes gồm các thanh ghi chức năng đặc biệt và các thanh ghi đa mục đích. Ngoài ra, PIC18F4550 còn được tích hợp 256 Bytes EEPROM cho phép ghi đến 1,000,000 lần.
- Có 5 Port I/O với 34 chân (Port A, Port B, Port C, Port D, Port E).
- Có 13 kênh đọc ADC 10 bit.
- Có 1 kênh CCP (Capture/Compare/PWM) và 1 kênh ECCP (Enhanced Capture/Compare/PWM).
- Giao tiếp SSP (Synchronous Serial Port) và MSSP (Master Synchronous Serial Port).
- Module Enhanced USART hỗ trợ RS-485, RS-232.
- Có 1 timer 8 bit, 3 timer 16 bit.
- Có 3 ngắt ngoài.

Sau đây là sơ đồ chân của PIC18F4550 trong hộp DIP-40:

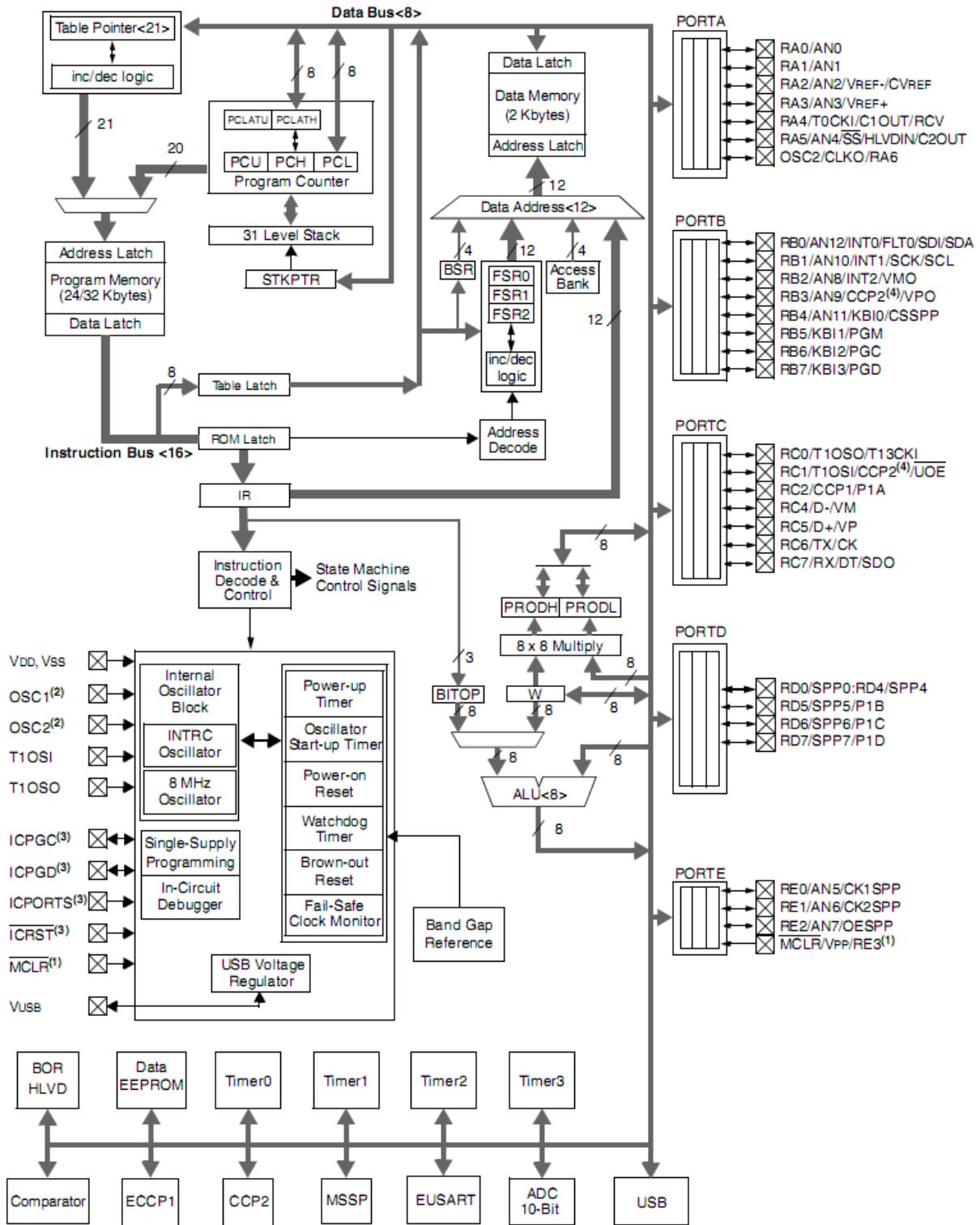


Sau đây là bảng hệ thống chức năng các chân và sơ đồ khối của PIC18F4550:

Chân	Hướng	Mô tả chức năng và các đặc tính
AN0-AN12	I	13 kênh Input có tác dụng là cổng biến đổi ADC
Avdd		Nguồn dương cho module ADC
Avss		GND cho module ADC
CLKI	I	Lối vào của xung Clock ngoài, luôn kết hợp với chân OSC1
CLKO	O	Lối ra của bộ dao động tinh thể, nối với tinh thể hoặc bộ cộng hưởng trong chế độ dao động thạch anh. Luôn kết hợp với chân chức năng OSC2
CN0 - CN7	I	Khai báo thay đổi lối vào
CN17 - CN18		
COFS	I/O	Cổng giao tiếp chuyển đổi dữ liệu đồng bộ khung
CSCK	I/O	Cổng giao tiếp chuyển đổi dữ liệu Clock vào ra nối tiếp
CSDI	I	Lối vào dữ liệu nối tiếp
CSDO	O	Lối ra dữ liệu nối tiếp
C1RX	I	Cổng nhận bus CAN1
C1TX	O	Cổng phát bus CAN1
EMUD	I/O	Cổng vào ra dữ liệu kênh truyền thông sơ cấp của ICD
EMUC	I/O	Vào ra xung nhịp kênh sơ cấp
EMUD1	I/O	Vào ra dữ liệu kênh thứ cấp
EMUC1	I/O	Vào ra dữ liệu kênh thứ cấp
EMUD2	I/O	Vào ra dữ liệu kênh thứ cấp
EMUC2	I/O	Vào ra dữ liệu kênh thứ cấp
EMUD3	I/O	Vào ra dữ liệu kênh thứ cấp
EMUC3	I/O	Vào ra dữ liệu kênh thứ cấp
IC1 - IC8	I	Các cổng vào của module Capture
INT0 - INT2	I	Các ngắt ngoài
LVDIN	I	Cổng vào phát hiện sụt thế
/MCLR	I	Chân Reset, mức tích cực thấp

OSC1	I	Lối vào bộ giao động tinh thể. Bộ đệm Trigger Schmitt được sử dụng khi cấu hình trong chế độ RC
OSC2	O	Lối ra bộ dao động tinh thể
PGD	I/O	Vào ra dữ liệu của ICSP
PGC	I	Lối vào Clock của ICSP
RA0 - RA6	I/O	Port A
RB0 - RB7	I/O	Port B
RC0 - RC7	I/O	Port C
RD0 - RD7	I/O	Port D
RE0 - RE3	I/O	Port E
SCK1	I/O	Vào ra Clock đồng bộ của khối SPI1
SDI1	I	Lối vào dữ liệu của khối SPI1
SDO1	O	Lối ra dữ liệu của SPI1
SS1	I	Slaver đồng bộ
SCL	I/O	Vào ra Clock nối tiếp của I2C
SDA	I/O	Vào ra Data nối tiếp đồng bộ của I2C
SOSCO	O	Lối ra bộ dao động tinh thể công suất thấp 32Khz
SOSCI	I	Lối vào bộ dao động 32Khz
T1CK	I	Lối vào xung Clock ngoài của Timer1
T2CK	I	Lối vào xung Clock ngoài của Timer2
U1RX	I	Cổng nhận khối UART1
U1TX	O	Cổng phát khối UART1
U1ARX	I	Cổng nhận mở rộng khối UART1
U1ATX	O	Cổng phát mở rộng khối UART1
VDD		Chân nguồn dương của PIC
VSS		Chân GND
Vref+	I	Lối vào Vref+ (cao) chuẩn của ADC
Vref-	I	Lối vào Vref- (thấp) chuẩn của ADC

Bảng 5.1: Bảng mô tả các chức năng từng chân của PIC18F4550

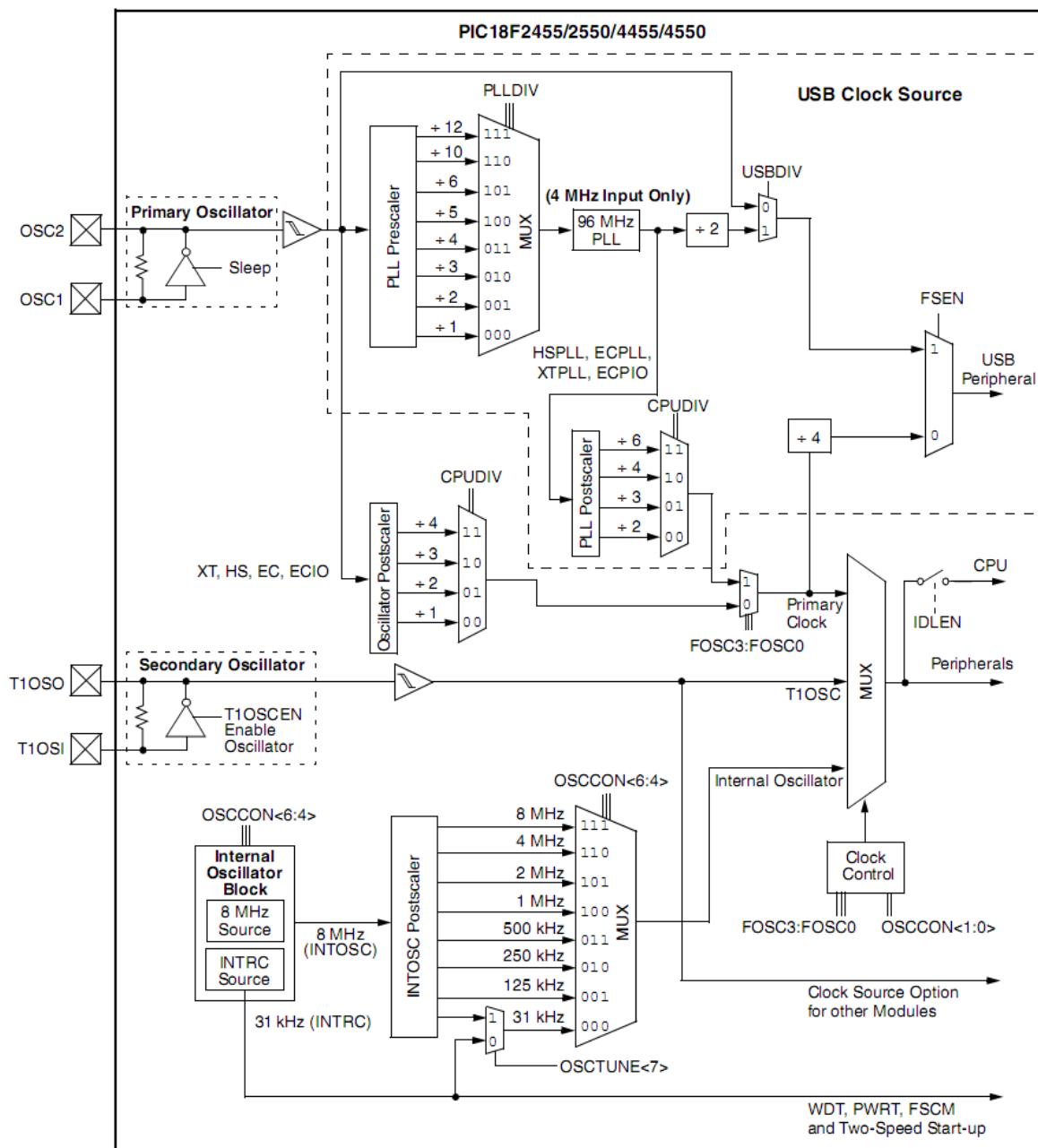


Hình 5.4: Sơ đồ khối của PIC 18F4550

5.1.2 Những module chính sử dụng trong luận văn

Với yêu cầu điều khiển động cơ có giao tiếp với máy tính, những module chính của PIC18F4550 được sử dụng là: bộ dao động, các port I/O, ngắt ngoài, các bộ Timer, khói điều xung PWM và khói giao tiếp EUSART.

- **Bộ dao động của PIC 18F4550:**



Hình 5.5: Sơ đồ khái niệm bộ dao động của PIC 18F4550

Hoạt động của bộ dao động trong PIC 18F4550 được điều khiển thông qua hai thanh ghi Configuration và hai thanh ghi control. Các thanh ghi CONFIG1L và CONFIG1H lựa chọn chế độ dao động và các options cho prescaler/postcaler của USB. Vì là các bits Configuration nên chúng được set khi thiết bị được lập trình và giữ nguyên cấu hình này cho đến khi thiết bị được lập trình mới lại. Thanh ghi OSCCON lựa chọn chế độ Active Clock, nó được dùng chủ yếu trong việc điều khiển clock switching trong chế độ quản lý năng lượng. Thanh ghi OSCTUNE dùng cho việc loại bỏ nguồn tần số INTRC, cũng như lựa chọn nguồn clock tần số thấp.

PIC 18F4550 có thể thực hiện 12 chế độ dao động khác nhau. Người lập trình có thể điều chỉnh các bit Configuration FOSC3:FOSC0 để lựa chọn chế độ dao động thích hợp:

- **XT:** Crystal/Resonator
- **XTPPLL:** Crystal/Resonator with PLL enabled
- **HS:** High-Speed Crystal/Resonator
- **HSPPLL:** High-Speed Crystal/Resonator with PLL enabled
- **EC:** External Clock with FOSC/4 output
- **ECIO:** External Clock with I/O on RA6
- **ECPLL:** External Clock with PLL enabled and FOSC/4 output on RA6
- **ECPIO:** External Clock with PLL enabled, I/O on RA6
- **INTHS:** Internal Oscillator used as microcontroller clock source, HS Oscillator used as USB clock source
- **INTXT:** Internal Oscillator used as microcontroller clock source, XT Oscillator used as USB clock source
- **INTIO:** Internal Oscillator used as microcontroller clock source, EC Oscillator used as USB clock source, digital I/O on RA6
- **INTCKO:** Internal Oscillator used as microcontroller clock source, EC Oscillator used as USB clock source, FOSC/4 output on RA6

Clock hệ thống của PIC18F4550 có thể được chọn từ hai nguồn dao động nội (Internal Oscillator) hoặc dao động ngoại (Primary Oscillator và Secondary Oscillator)

nhờ bộ chọn kênh MUX. Bộ MUX được điều khiển bởi các bit FOSC<3:0> (bit 3, bit 2, bit 1, bit 0 của thanh ghi CONFIG1 16-bit định vị tại địa chỉ 2007H và 2008H trong bộ nhớ chương trình) và các bit SCS<1:0> (bit 1, bit 0 của thanh ghi OSCCON). Các bits SCS dùng để lựa chọn chế độ Primary Clock, T1OSC hay Internal Oscillator. Các bit FOSC<3:0> được sử dụng để cấu hình bộ dao động Primary Clock.

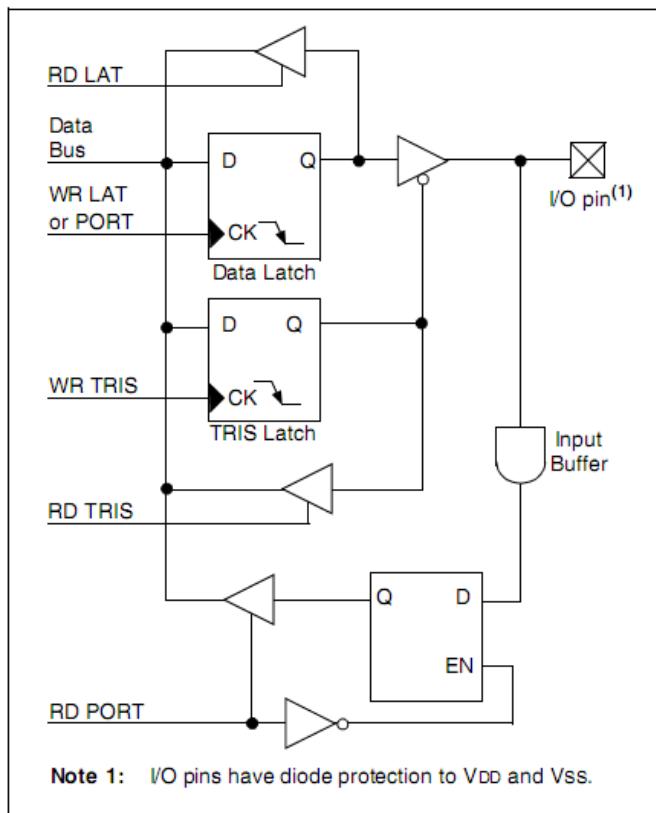
Bộ dao động nội gồm 2 bộ dao động HFINTOSC 8MHz và LFINTOSC 31kHz. Clock 8MHz của bộ HFINTOSC được chia thành các tần số 8MHz, 4MHz, 2MHz, 1MHz, 500kHz, 250kHz, 125kHz nhờ bộ chia tần số postscaler. Tần số nguồn clock (INTOSC direct, INTRC direct hoặc INTOSC postscaler) được lựa chọn bằng việc điều chỉnh các bits IRCF trong thanh ghi OSCCON.

Bộ dao động ngoại (được tích hợp bên trong PIC) cần được kết nối với các bộ lọc tại các chân OSC1, OSC2. Trong đề tài này, ta sử dụng thạch anh 12MHz để tạo thành xung clock 12MHz cung cấp cho clock hệ thống.

▪ Các port I/O:

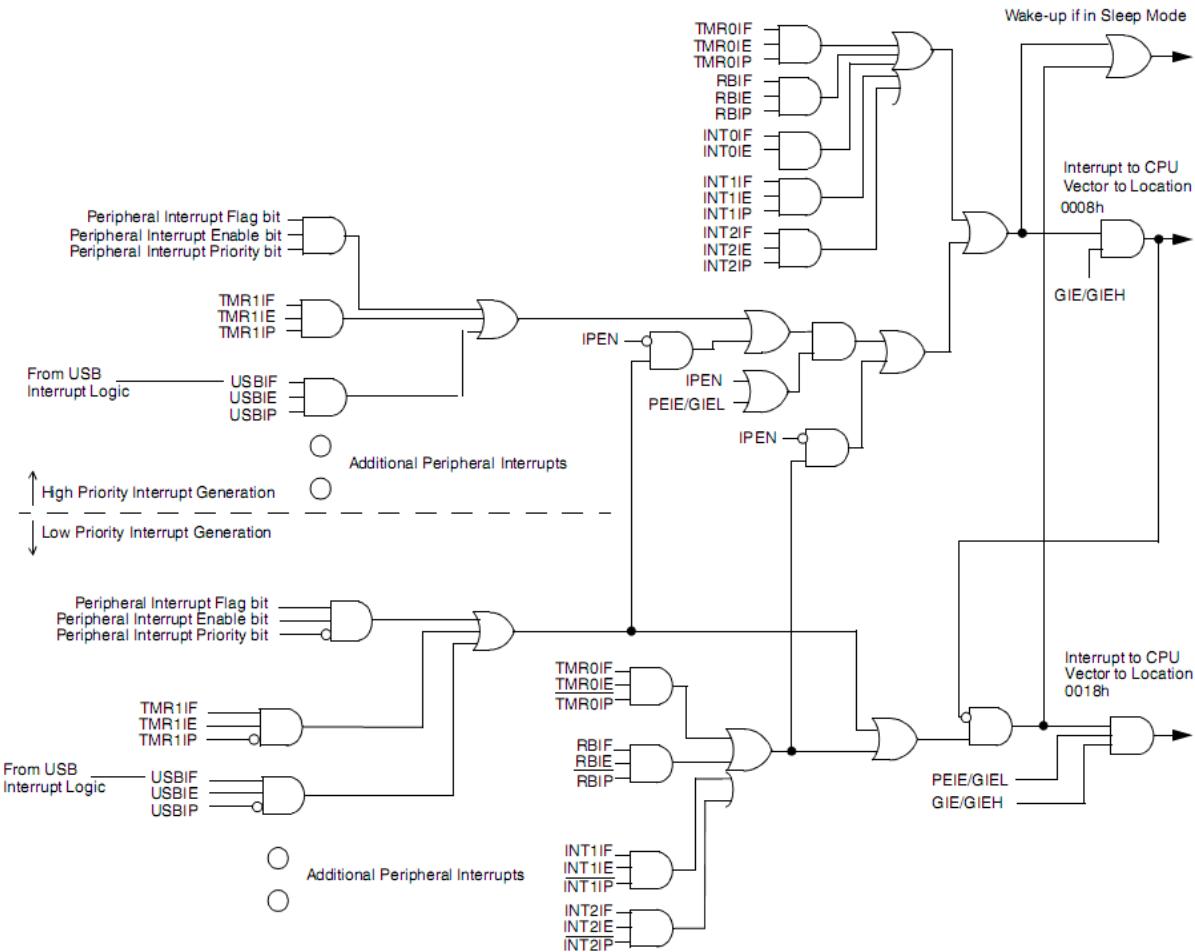
PIC18F4550 tất cả 34 chân I/O mục đích thông thường (GPIO: General Purpose Input Output) có thể được sử dụng. Tùy theo những thiết bị ngoại vi được chọn mà một vài chân có thể không được sử dụng ở chức năng GPIO. Thông thường, khi một thiết bị ngoại vi được chọn, những chân liên quan của thiết bị ngoại vi có thể không được sử dụng ở chức năng GPIO. 34 chân GPIO được chia cho 5 Port: Port A gồm 7 chân, Port B gồm 8 chân, Port C gồm 8 chân, Port D gồm 8 chân và Port E gồm 3 chân.

Mỗi port được điều khiển bởi 2 thanh ghi 8-bit, thanh ghi Port và thanh ghi Tris. Thanh ghi Tris được sử dụng để điều khiển port là nhập hay xuất. Mỗi bit của Tris sẽ điều khiển mỗi chân của port đó, nếu giá trị của bit là 1 thì chân liên quan là nhập, ngược lại nếu giá trị của bit là 0 thì chân liên quan là xuất. Thanh ghi Port được sử dụng để chứa giá trị của port liên quan. Mỗi bit của thanh ghi Port sẽ chứa giá trị của chân liên quan.



Hình 5.6: Cấu tạo của chân GPIO

- Ngắt ngoài trên các chân RB:



Hình 5.7: Sơ đồ khối logic của hệ thống ngắt trong PIC18F4550

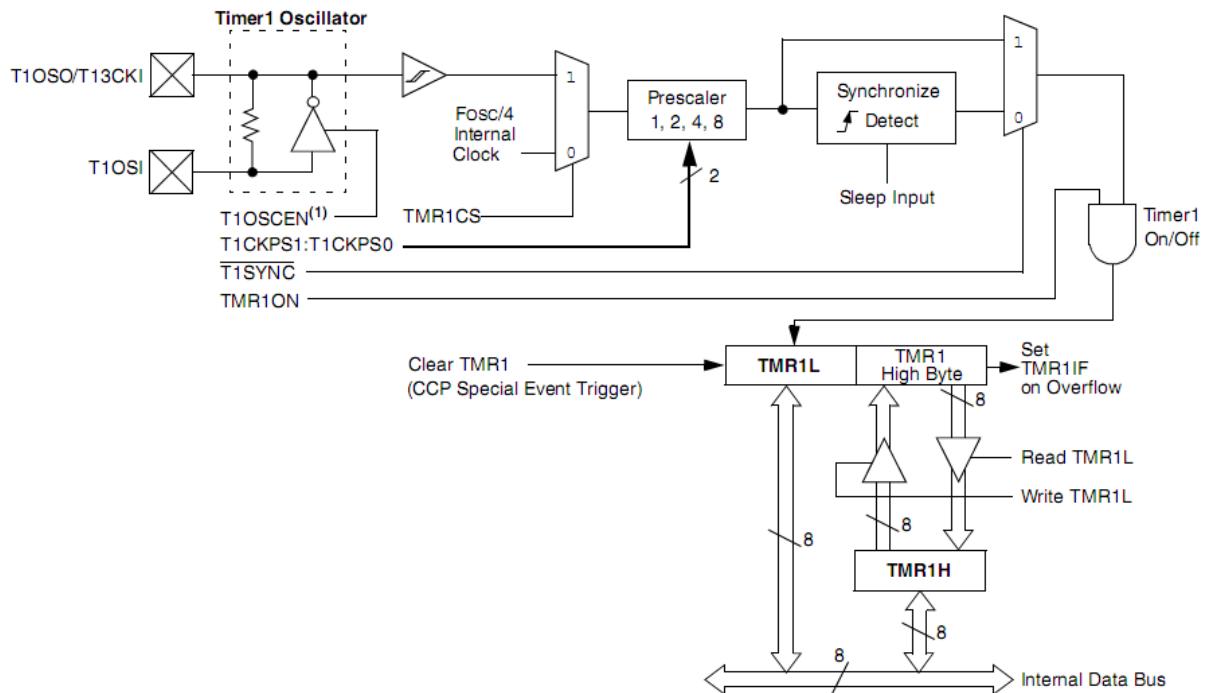
Các ngắt ngoài INT0 ứng với chân RB0, INT1 ứng với chân RB1, INT2 ứng với chân RB2.

Các ngắt ngoài trên các chân RB được kích khởi theo sùm. Sùm lên nếu như bit INTEDG = 1 (bit 6 của thanh ghi OPTION_REG), sùm xuống nếu INTEDG = 0. Khi một sùm thích hợp xuất hiện trên chân RB, cờ INTF được bật lên 1. Ngắt này có thể được cho phép nếu bit INTE=1, không cho phép nếu INTE=0. Cờ INTF cần được xóa bằng phần mềm trong trình phục vụ ngắt trước khi cho phép ngắt trở lại.

Trong đề tài này, ta dùng ngắt ngoài INT1 (trên chân RB1) và INT2 (trên chân RB2) để đọc encoder về phục vụ cho việc tính toán PID.

▪ Cấu tạo và hoạt động của bộ Timer 1 và Timer 2:

Bộ Timer 1 là bộ định thời 16-bit có cấu tạo như hình sau:

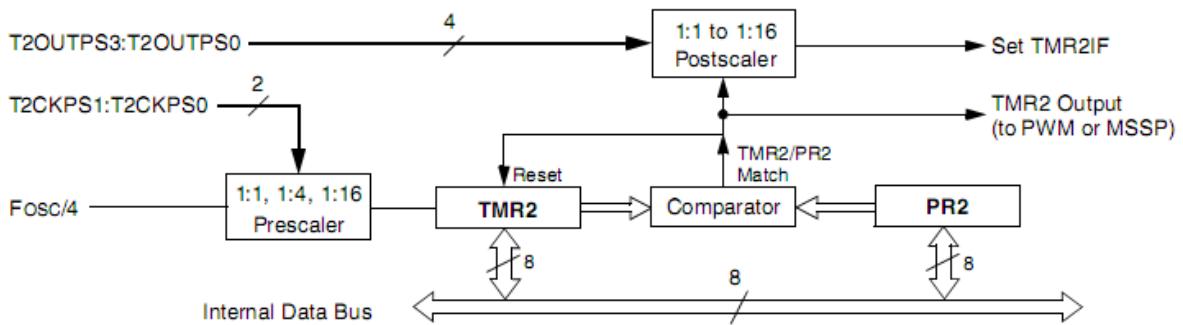


Hình 5.8: Sơ đồ khối của bộ Timer 1

Bộ Timer 1 là bộ đếm lên 16-bit được truy xuất gián tiếp thông qua cặp thanh ghi TMR1H, TMR1L. Khi được đọc hoặc ghi các thanh ghi này sẽ cập nhật trực tiếp giá trị cho bộ Timer. Khi được sử dụng với nguồn clock nội, bộ Timer 1 sẽ có vai trò là bộ định thời. Khi được sử dụng với nguồn clock ngoại, nó sẽ có vai trò là định thời hoặc bộ đếm. Sử dụng bit TMR1CS để chọn nguồn clock.

Các bit T1CKPS<1:0> định giá trị cho bộ chia tần số Prescaler. Khi bộ TMR1 tràn (từ FFFFh đến 0000h) cờ ngắt TMR1IF sẽ được thiết lập lên 1. Nếu lúc này cờ TMR1IE = 1, cờ PEIE=1 và GIE=1 thì ngắt Timer1 sẽ xảy ra. Cờ TMR1IF cần được xóa trong trình phục vụ ngắt Timer 1. Trong đề tài này, ta dùng Timer 1 cho việc lấy mẫu của bộ PID số.

Sau đây là sơ đồ khối bộ Timer 2:



Hình 5.9: Sơ đồ khái niệm của bộ Timer 2

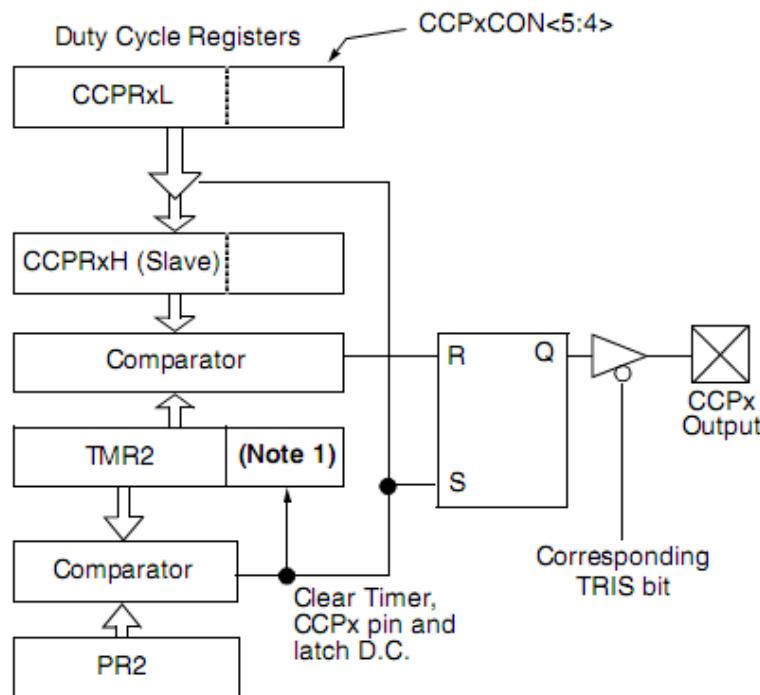
TMR2 tăng từ 00H với mỗi clock (FOSC/4), hai bit counter/prescaler trên ngõ vào clock cho ngõ vào trực tiếp với lựa chọn divide-by-4 hoặc divide-by-16 prescale. Chúng được lựa chọn bằng bit prescaler control, T2CKPS1:T2CKPS0 (T2CON<1:0>). Giá trị của TMR2 được so sánh với giá trị của thanh ghi chu kì, PR2, trong mỗi chu kì clock. Khi hai giá trị này tương ứng nhau, bộ so sánh tạo ra một tín hiệu điều khiển ở ngõ ra của bộ Timer, tín hiệu này cũng reset lại giá trị của TMR2 về 00H ở chu kì kế tiếp và lái ngõ ra bộ counter/postscaler. Các thanh ghi TMR2 và PR2 đều có thể đọc và ghi. Thanh ghi TMR2 bị xóa khi có một thiết bị nào đó reset, trong khi đó thanh ghi PR2 khởi tạo ở FFH. Ngõ ra không chia tỉ lệ của TMR2 chủ yếu được dùng cho module CCP, cơ sở thời gian cho các phép toán trong chế độ PWM.

Trong đề tài này, ta dùng Timer 2 cho việc điều xung PWM.

▪ Cấu tạo và hoạt động của khối điều xung PWM:

PIC18F4550 có hai bộ điều xung, hai bộ này sẽ tạo ra các tín hiệu điều xung trên các chân CCP1 và CCP2. Độ rộng, chu kỳ và độ phân giải của hai bộ điều xung được xác định bởi các thanh ghi PR2, T2CON, CCPR1L, CCPR2L, CCP1CON, CCP2CON. Ở chế độ điều xung (PWM), chân CCPx có thể tạo ra đầu ra PWM với độ phân giải lên đến 10 bit.

Vì chân CCPx được nhập chung với đường dữ liệu ở port B hoặc port C nên để các chân CCPx (CCP1 và CCP2) hoạt động ở chế độ PWM, cần xóa bit TRIS tương ứng của các chân đó. Sơ đồ khái niệm của các bộ điều xung được mô tả trong hình dưới đây:

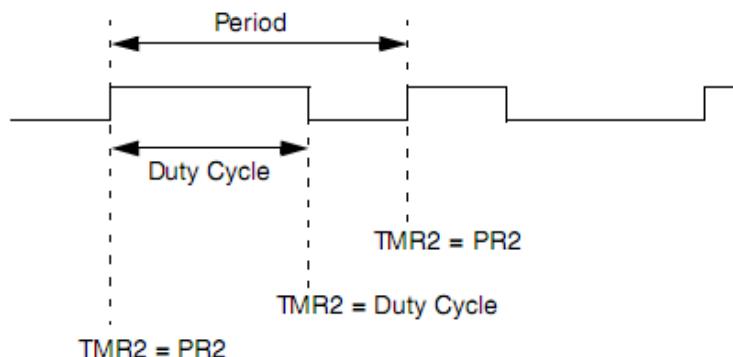


Note 1: The 8-bit TMR2 value is concatenated with the 2-bit internal Q clock, or 2 bits of the prescaler, to create the 10-bit time base.

Hình 5.10: Sơ đồ khối bộ PWM

Chú thích (Note 1) trong hình trên biểu thị rằng thanh ghi 8-bit TMR2 được kết hợp với 2-bit prescaler của bộ dao động nội để tạo ra bộ định thời 10-bit. Các thanh ghi CCPRxH là các thanh ghi chỉ đọc, kết hợp với 2 bit 5 và 4 của các thanh ghi CCPxCON có vai trò định độ rộng của xung; các thanh ghi này được ghi gián tiếp thông qua các thanh ghi CCPRxL. Thanh ghi 8-bit PR2 định chu kỳ cho xung ra.

Sóng điều xung tại các chân CCPx có giản đồ thời gian như sau:



Hình 5.11: Giản đồ thời gian của sóng điều xung tại chân CCPx

Thanh ghi TMR2 kết hợp với 2 bit prescaler sẽ đếm lên nhờ xung clock của hệ thống. Khi giá trị của TMR2 nhỏ hơn giá trị của CCPRxL:CCPxCON<5:4>, chân CCPx ở mức cao. Khi giá trị của TMR2 bằng với giá trị này, bộ so sánh sẽ đảo chân CCPx xuống mức 0. Khi giá trị của TMR2 bằng với PR2, TMR2 sẽ được xóa về 0 đồng thời kết thúc chu kỳ xung, chân CCPx lại được thiết lập ở mức cao.

Chu kỳ của xung được tính theo công thức sau:

$$\text{Chu kỳ PWM} = [(PR2)+1] \times 4 \times T_{OSC} \times (\text{giá trị Pre-scale của TMR2})$$

Trong đó:

T_{OSC} là chu kỳ của clock hệ thống. $T_{OSC} = 1/F_{OSC}$.

Trong đề tài, ta cài đặt: `setup_timer_2(T2_DIV_BY_4, 255, 1)`, do đó

$$T_{PWM} = (PR2+1) \times 4 \times T_{OSC} \times \text{Pre-scale} = 256 \times 4 \times (12 \times 10^6)^{-1} \times 4.$$

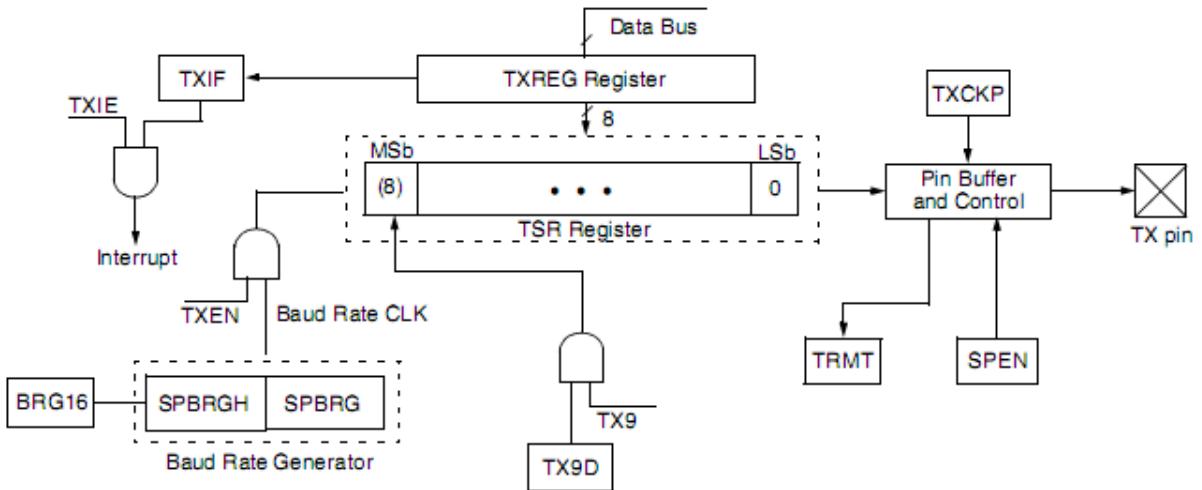
$$f_{PWM} = 1/T_{PWM} \approx 2,93 \text{ (kHz)}.$$

▪ **Hoạt động của khối giao tiếp EUSART:**

Khối giao tiếp nối tiếp EUSART (Enhanced Universal Synchronous Asynchronous Receiver Transmitter) cho phép cấu hình hoạt động ở chế độ giao tiếp nối tiếp đồng bộ và không đồng bộ. Trong đề tài này, chế độ giao tiếp không đồng bộ được sử dụng. Do đó ở đây ta sẽ tập trung mô tả hoạt động của module EUSART ở chế độ không đồng bộ.

Hoạt động truyền:

Sơ đồ khói bộ truyền được thể hiện trong hình dưới:

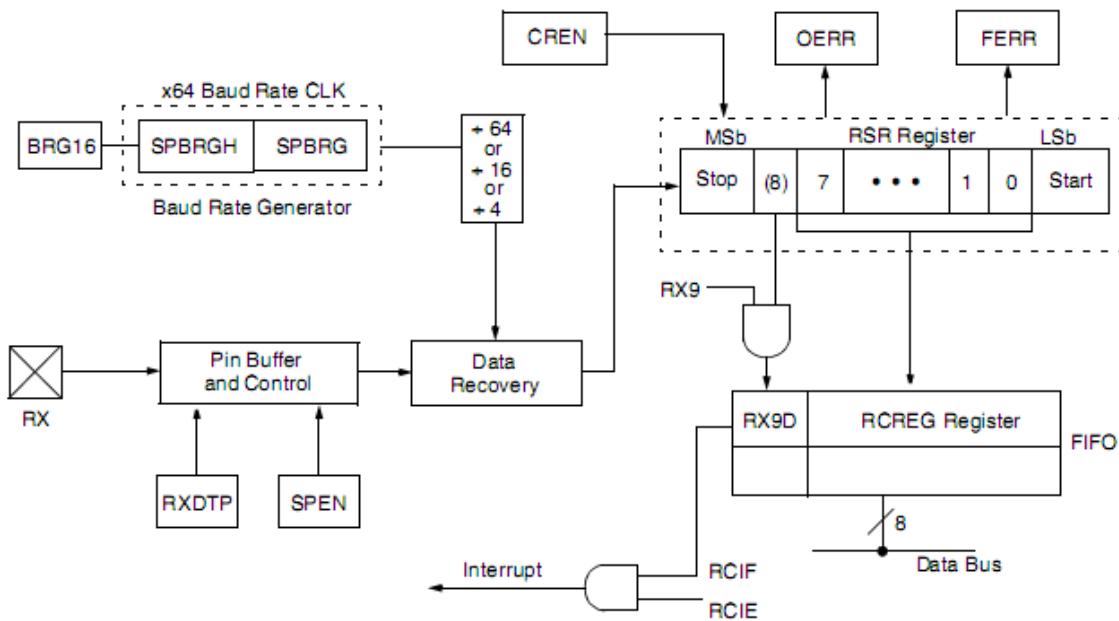


Hình 5.12: Sơ đồ khái niệm của module EUSART

Bộ phận chính của khái niệm truyền là thanh ghi truyền TSR. Thanh ghi này không thể truy cập bằng phần mềm mà phải truy cập gián tiếp qua thanh ghi đệm truyền TXREG. Quá trình truyền được khởi tạo bằng cách ghi dữ liệu truyền vào thanh ghi TXREG. Nếu đây là dữ liệu truyền lần đầu tiên hoặc dữ liệu truyền trước đã truyền hoàn tất thì dữ liệu trong TXREG ngay lập tức sẽ được truyền vào thanh ghi TSR. Nếu thanh ghi TSR vẫn còn chứa dữ liệu của ký tự truyền trước thì dữ liệu mới trong TXREG sẽ được giữ cho đến khi bit Stop của ký tự đang truyền hoàn tất. Sau đó, dữ liệu chờ trong TXREG sẽ được truyền vào TSR.

Hoạt động nhận:

Sơ đồ khái niệm bộ nhận được thể hiện trong hình dưới:



Hình 5.13: Sơ đồ khối bộ nhận của module EUSART

Dữ liệu được nhận trên chân RX/DT và đẩy vào khối Data Recovery. Khi tất cả 8 hoặc 9 bit của ký tự nhận đã được dịch vào, chúng sẽ ngay lập tức được chuyển vào bộ đếm 2 ký tự FIFO. Bộ đếm FIFO và thanh ghi RSR không thể truy cập trực tiếp bằng phần mềm mà phải truy cập gián tiếp thông qua thanh ghi RCREG. Dữ liệu trong bộ đếm nhận được đọc bằng cách đọc thanh ghi RCREG.

5.2 Mạch công suất (mạch cầu H)

Dùng Module Dual H-Bridge của HLAB (xem thêm ở hình 5.1), có một số đặc điểm sau:

- Module điều khiển hai động cơ.
- Điện áp 12V – 24V, dòng tải tối đa 5A.
- Điện áp điều khiển 5V TTL.
- Hỗ trợ chọn PWM tích cực mức cao hoặc thấp bằng jumper.
- Diode bảo vệ cầu H.
- Led chỉ thị chiều quay của động cơ.

Chương 6: Động cơ và giải thuật PID vị trí

Nội dung chính

6.1 Động cơ Servo DC

6.1.1 Động cơ DC

6.1.2 Encoder

6.2 Giải thuật PID vị trí

6.1 Động cơ Servo DC

Động cơ Servo DC [18] là động cơ không đồng bộ. Động cơ Servo DC bao gồm hai thành phần chính: động cơ DC và encoder. Ngoài ra, động cơ có thể được gắn thêm hộp số (Gear box) có tác dụng tăng momen quay và giảm tốc độ động cơ. Động cơ Servo DC được ứng dụng cho việc điều khiển chính xác: góc quay, tốc độ, momen. Trong đề tài, cặp động cơ được chọn là động cơ Servo DC hiệu DS48DE25 của Nhật với công suất 60W, sử dụng nguồn DC 24V, encoder 60 xung và được trang bị hộp số 1:5.



Hình 6.1: Cặp động cơ Servo DC

6.1.1 Động cơ DC

Động cơ DC là động cơ điện hoạt động với dòng điện một chiều. Động cơ điện một chiều ứng dụng rộng rãi trong các ứng dụng dân dụng cũng như công nghiệp. Cấu tạo của động cơ gồm có hai phần: statô đứng yên và roto quay so với statô. Phần cảm

(phản kích từ – thường đặt trên stato) tạo ra từ trường đi trong mạch từ, xuyên qua các vòng dây quấn của phần ứng (thường đặt trên roto). Khi có dòng điện chạy trong mạch phản ứng, các thanh dẫn phản ứng sẽ chịu tác động bởi các lực điện từ theo phương tiếp tuyến với mặt trụ roto, làm cho roto quay.

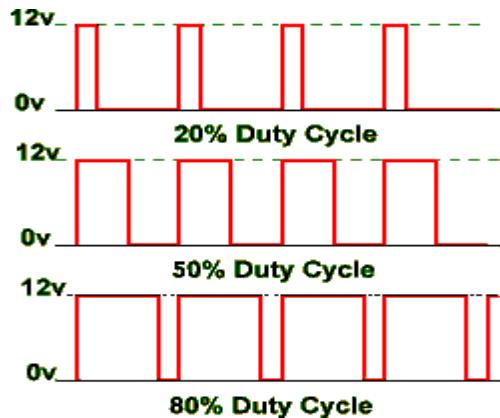
Tùy theo cách mắc cuộn dây roto và stato mà người ta có các loại động cơ sau:

Động cơ kích từ độc lập: Cuộn dây kích từ (cuộn dây stato) và cuộn dây phản ứng (roto) mắc riêng rẽ nhau, có thể cấp nguồn riêng biệt.

Động cơ kích từ nối tiếp: Cuộn dây kích từ mắc nối tiếp với cuộn dây phản ứng.

Đối với loại động cơ kích từ độc lập, người ta có thể thay thế cuộn dây kích từ bởi nam châm vĩnh cửu, khi đó ta có loại động cơ điện một chiều dùng nam châm vĩnh cửu.

Đối với loại động cơ kích từ độc lập dùng nam châm vĩnh cửu, để thay đổi tốc độ, ta thay đổi điện áp cung cấp cho roto. Việc cấp áp một chiều thay đổi thường khó khăn, do vậy người ta dùng phương pháp điều xung (PWM):

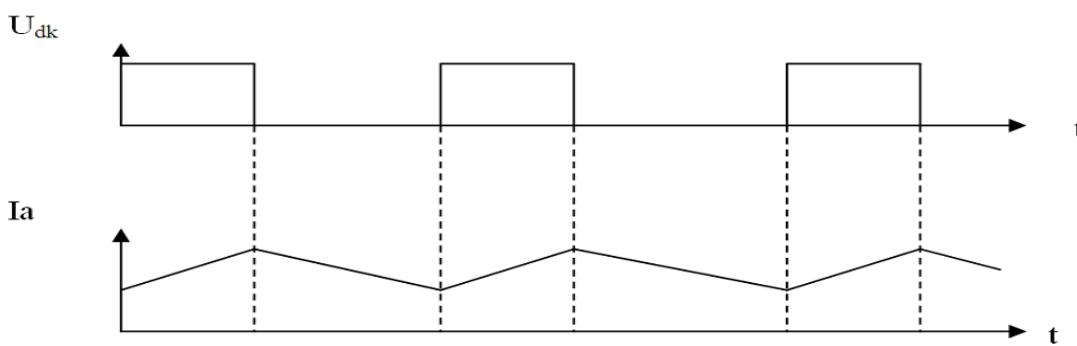


Hình 6.2: Điều chỉnh độ rộng xung PWM

Mạch điều khiển động cơ bằng phương pháp PWM hoạt động dựa theo nguyên tắc cấp nguồn cho động cơ bằng chuỗi xung đóng mở với tốc độ nhanh. Nguồn DC được chuyển đổi thành tín hiệu xung vuông (chỉ gồm hai mức 0 volt và xấp xỉ điện áp hoạt động). Tín hiệu xung vuông này được cấp cho động cơ. Nếu tần số chuyển mạch đủ lớn động cơ sẽ chạy với một tốc độ đều đặn phụ thuộc vào momen của trực quay.

Với phương pháp PWM này, ta điều chỉnh tốc độ của động cơ thông qua việc điều chế độ rộng của xung, tức là thời gian mức cao của chuỗi xung vuông cấp cho động cơ. Việc điều chỉnh này sẽ tác động đến công suất trung bình cấp cho động cơ và do đó sẽ làm thay đổi tốc độ của động cơ cần điều khiển. Như trên hình 6.2, với dãy xung điều khiển trên cùng, xung có độ rộng nhỏ nên động cơ chạy chậm. Nếu độ rộng xung càng lớn (như dãy xung thứ 2 và thứ 3) động cơ DC chạy càng nhanh.

Do đặc tính cảm kháng của động cơ, dòng qua động cơ là dòng liên tục, gợn sóng như sau:



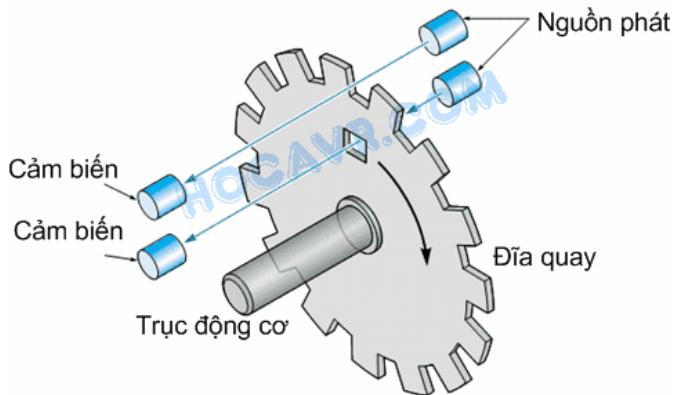
Hình 6.3: Dạng sóng áp và dòng trên động cơ

6.1.2 Encoder

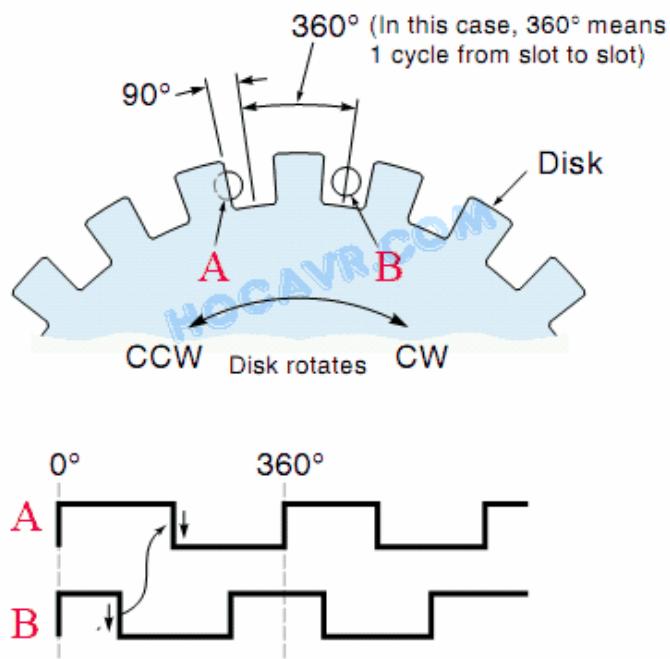
Encoder là dụng cụ dùng để xác định vị trí góc của đĩa quay hoặc xác định quãng đường di chuyển cho bánh xe, trực động cơ hay bất kỳ thiết bị nào có cơ cấu quay và cần xác định độ dịch chuyển.

Có hai loại encoder là encoder tuyệt đối (absolute encoder) và encoder tương đối (incremental encoder). Encoder tuyệt đối cung cấp cho ta chính xác vị trí encoder mà không cần qua xử lý, trong khi encoder tương đối thì đòi hỏi thêm khâu xử lý từ tín hiệu encoder trước khi xác định được vị trí quay của động cơ. Với giá thành rẻ hơn và việc xử lý cũng không quá phức tạp nên encoder tương đối được chọn trong đề tài này.

Ở đây, ta tập trung nói về *incremental optical encoder* là loại encoder tương đối sử dụng cảm biến quang. Hệ thống optical encoder bao gồm một nguồn phát quang (thường là hồng ngoại – infrared), một cảm biến quang và một đĩa có chia rãnh.

**Hình 6.4: Optical Encoder**

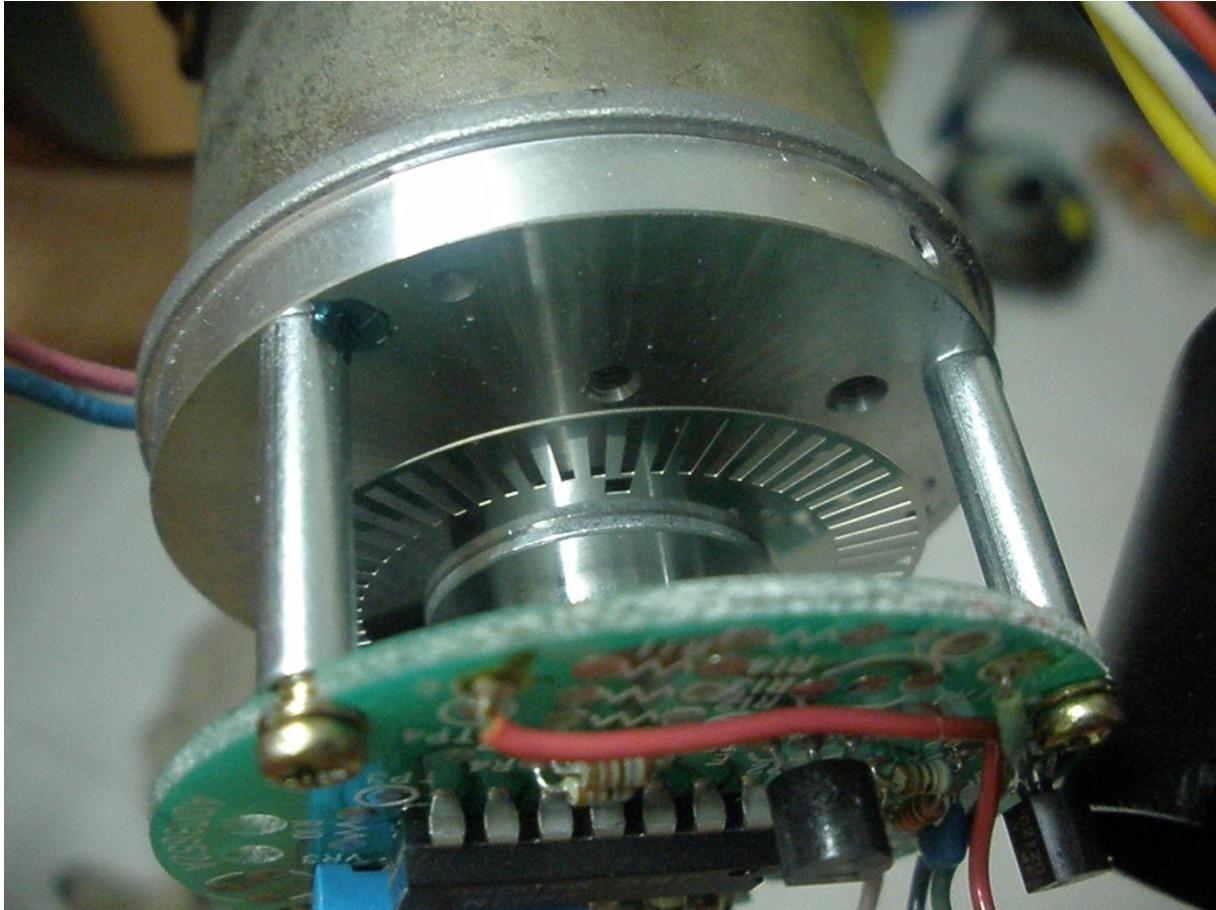
Encoder thường có ba kênh (ba ngõ ra) bao gồm kênh A, kênh B và kênh Z (Index). Trên hình 6.4, ta thấy một lỗ nhỏ trên đĩa quay và một cặp phát – thu dành riêng cho lỗ nhỏ này. Đó là kênh Z của encoder. Cứ mỗi lần động cơ quay được một vòng, hòng ngoại từ nguồn phát sẽ xuyên qua lỗ nhỏ đến cảm biến quang và cảm biến lại thu được một tín hiệu. Như thế cứ mỗi vòng quay của động cơ, lại xuất hiện một xung ở kênh Z. Bên ngoài đĩa quay được chia thành các rãnh nhỏ và có một cặp thu – phát khác dành cho các rãnh này. Đây chính là kênh A của encoder, hoạt động của kênh A cũng tương tự kênh Z, điểm khác nhau là trong một vòng quay của động cơ, sẽ có N xung xuất hiện trên kênh A. N là số rãnh trên đĩa và còn được gọi là độ phân giải (resolution) của encoder. Mỗi loại encoder có một độ phân giải khác nhau, có khi trên mỗi đĩa chỉ có vài rãnh nhưng cũng có trường hợp có đến hàng nghìn rãnh được chia. Để điều khiển động cơ, ta phải biết được độ phân giải của encoder đang dùng. Độ phân giải ảnh hưởng đến độ chính xác điều khiển và cả phương pháp điều khiển. Ngoài ra, trên encoder còn có một cặp thu phát khác được đặt trên cùng đường tròn với kênh A nhưng lệch một chút (lệch $M+0.5$ rãnh), đây là kênh B của encoder. Tín hiệu xung từ kênh B có cùng tần số với kênh A nhưng lệch pha 90° . Bằng cách phối hợp kênh A và kênh B ta có thể biết chiều quay của động cơ.



Hình 6.5: Hai kênh A và B lệch pha trong encoder

Khi cảm biến A bắt đầu bị che thì cảm biến B vẫn nhận được hồng ngoại xuyên qua và ngược lại. Hình trên là dạng xung ngõ ra trên 2 kênh. Xét trường hợp động cơ quay cùng chiều kim đồng hồ, tín hiệu di chuyển từ trái sang phải. Lúc tín hiệu kênh A chuyển từ mức cao xuống thấp (cạnh xuống) thì kênh B đang ở mức thấp. Ngược lại, nếu động cơ quay ngược chiều kim đồng hồ, tín hiệu di chuyển từ phải qua trái. Lúc này, tại cạnh xuống của kênh A thì kênh B đang ở mức cao. Như vậy, bằng cách phối hợp hai kênh A và B, ta không những xác định được góc quay (thông qua số xung) mà còn biết được chiều quay của động cơ (thông qua mức của kênh B ở cạnh xuống của kênh A).

Encoder được sử dụng trong đè tài là loại 60 xung trên một vòng quay với đủ ba kênh A, B và Z, tuy nhiên chỉ cần sử dụng 2 kênh A, B là đủ cho ứng dụng điều khiển vị trí.



Hình 6.6: Encoder đi kèm động cơ Servo DC

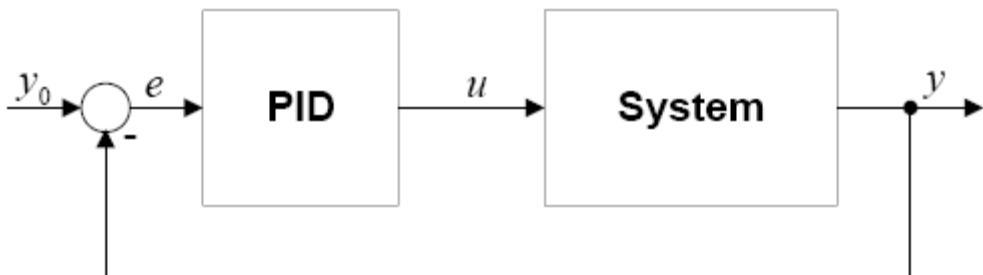
Cách đọc encoder bằng PIC:

Ta sử dụng ngắt ngoài của PIC 18F4550 để đọc encoder về, đây là phương pháp đơn giản nhưng chính xác. Ý tưởng của phương pháp này rất đơn giản, ta nối kênh A của encoder của động cơ bên phải với ngắt ngoài (chân RB1) và kênh B với một chân nào đó bất kỳ (ở đây ta dùng chân RD1). Cứ mỗi lần ngắt ngoài xảy ra, tức có 1 xung xuất hiện ở kênh A thì trình phục vụ ngắt ngoài tự động được gọi. Trong trình phục vụ ngắt này ta kiểm tra mức của kênh B, tùy theo mức của kênh B ta sẽ tăng biến đếm xung lên 1 hoặc giảm đi 1. Làm tương tự với encoder của động cơ bên trái, nối kênh A của encoder với ngắt ngoài (chân RB2) và kênh B với một chân nào đó bất kỳ (chân RD2).

6.2 Giải thuật PID vị trí [19]

Trong các lĩnh vực điều khiển mà ở đó ta cần điều khiển đầu ra theo mong muốn hoặc cần sự ổn định, chúng ta cần một thuật toán điều khiển ví dụ như điều khiển động cơ, nhiệt độ, áp suất, tốc độ hay các biến khác trong các ứng dụng khác nhau. Thuật toán PID là một trong những thuật toán kinh điển được giới thiệu để giải quyết các bài toán về điều khiển. Bộ điều khiển PID có thể sử dụng để điều khiển bất kỳ biến nào miễn sao giá trị của chúng bị tác động thông qua thuật toán PID.

Thuật toán PID có thể mô tả như hình 6.7. Bộ điều khiển PID sẽ giám sát đầu ra của hệ thống, sau đó so sánh với giá trị mong muốn. Nếu hệ thống chưa đạt được giá trị mong muốn, nghĩa là sai số e khác không, bộ PID sẽ tính toán ra giá trị tác động u mới (tác động vào hệ thống) để hiệu chỉnh hệ thống. Quá trình này sẽ được thực hiện liên tục cho đến khi nào $e = 0$, lúc đó hệ thống sẽ đạt trạng thái ổn định. Hệ thống như thế còn được gọi là hệ thống điều khiển vòng kín (Closed Loop Control) vì nó có hồi tiếp trạng thái của hệ thống về bộ điều khiển.



Hình 6.7: PID vòng kín

PID là chữ viết tắt của tỷ lệ - tích phân - vi phân. Hàm ý rằng bộ điều khiển sẽ tính giá trị tỷ lệ, tính tích phân và vi phân của sai số để hiệu chỉnh hệ thống về trạng thái cân bằng động.

Đặc tính bộ điều khiển P,I,D:

- Thành phần tỉ lệ (K_P) có tác dụng làm tăng tốc độ đáp ứng của hệ và làm giảm (chứ không triệt tiêu) sai số xác lập của hệ (steady – state error).
- Thành phần tích phân (K_I) có tác dụng triệt tiêu sai số xác lập nhưng có thể làm giảm tốc độ đáp ứng của hệ.

- Thành phần vi phân (K_D) làm tăng độ ổn định của hệ thống, giảm độ vọt lố và cải thiện tốc độ đáp ứng của hệ.

Ảnh hưởng của các thành phần K_P , K_I , K_D đối với hệ kín được tóm tắt trong bảng sau:

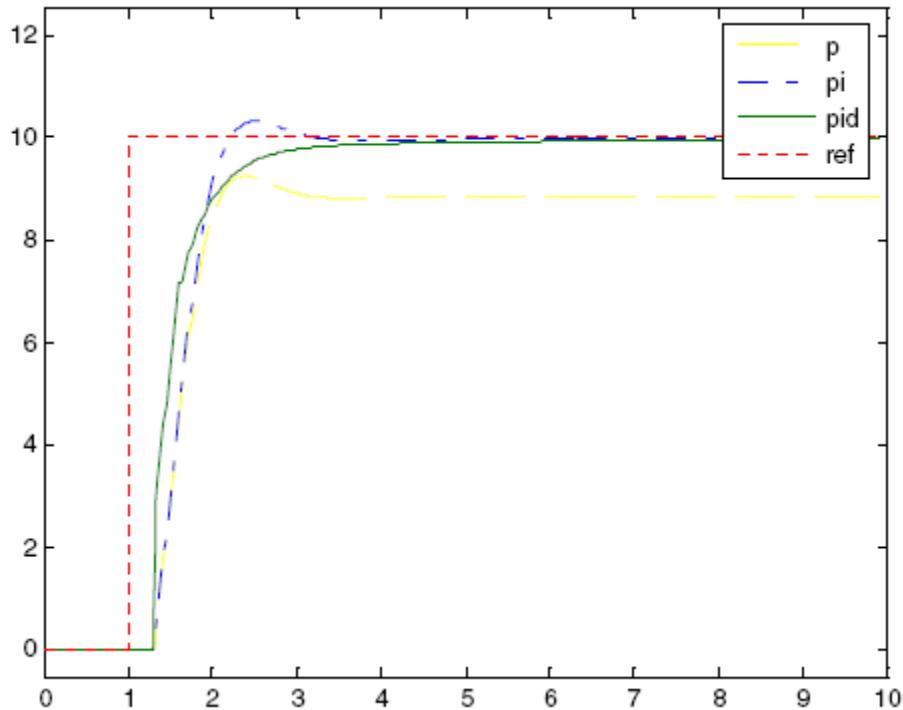
Đáp ứng của hệ thống	Thời gian tăng	Vọt lố	Thời gian ổn định	Sai lệch so với trạng thái bền
K_P	Giảm	Tăng	Ít thay đổi	Giảm
K_I	Giảm	Tăng	Tăng	Triệt tiêu
K_D	Ít thay đổi	Tăng	Tăng	Ít thay đổi

Bảng 6.1: Ảnh hưởng của các thành phần K_P , K_I , K_D đối với hệ kín

Hệ PID có thể sử dụng ở chế độ P, PI, PD tùy vào đặc tính của hệ thống.

- Hiệu chỉnh tỉ lệ P:* K_P càng lớn thì sai số xác lập càng nhỏ, vọt lố cao, tuy nhiên ta không thể tăng K_P đến vô cùng khi e_{xl} không bằng không, $K_P \leq K_{gh}$.
- Hiệu chỉnh tỉ lệ PD:* Hiệu chỉnh PD làm nhanh đáp ứng của hệ thống, giảm thời gian quá độ nhưng lại nhạy với nhiễu tần số cao.
- Hiệu chỉnh tỉ lệ PI:* Hiệu chỉnh tỉ lệ PI làm chậm đáp ứng quá độ, tăng độ vọt lố, giảm sai số xác lập.
- Hiệu chỉnh tỉ lệ PID:* Cải thiện đáp ứng quá độ (giảm vọt lố, giảm thời gian quá độ), giảm sai số xác lập.

Kết quả của bộ điều khiển thường được đánh giá thông qua việc đo đặc đáp ứng của bộ điều khiển trong một hệ thống nhất định như hình sau:



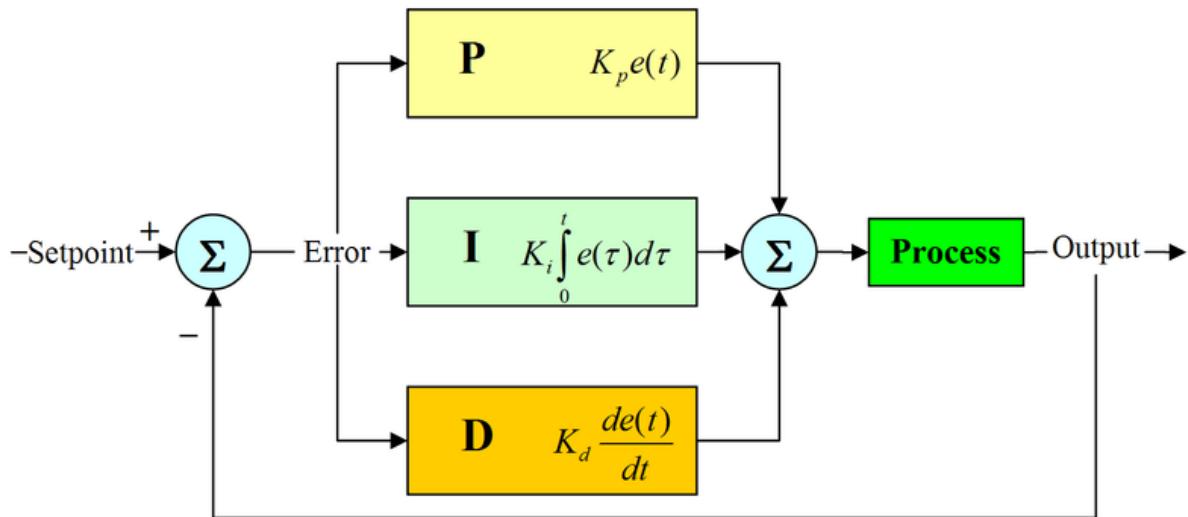
Hình 6.8 Đáp ứng của các hệ thống điều khiển

Hình 6.8 mô tả đáp ứng của hệ thống điều khiển theo sự thay đổi và đáp ứng của ba phương pháp điều khiển P, PI, PID. Trong đó đáp ứng của bộ PID là tốt nhất với độ vọt lồ ít và thời gian ổn định nhanh.

Bộ PID có thể làm việc trên dữ liệu liên tục hoặc dạng rời rạc. Hiện nay, PID rời rạc được ứng dụng rộng rãi trong các hệ điều khiển do hầu hết các hệ thống hiện nay đều là hệ thống số. Công thức bộ PID có thể được mô tả như sau:

$$u(n) = K_p e(n) + K_i \sum_{k=0}^n e(k) + K_d (y(n) - y(n-1))$$

Quá trình tính toán của bộ PID:



Hình 6.9: Quá trình tính toán PID

Bộ điều khiển PID có hàm truyền dạng liên tục như sau:

$$G(s) = K_p + \frac{K_I}{s} + K_D \times s$$

Biến đổi Z của nó như sau:

$$G(z) = K_p + \frac{K_I \times T}{2} \left(\frac{z+1}{z-1} \right) + \frac{K_D}{T} \left(\frac{z-1}{z} \right)$$

Viết lại G(z) như sau:

$$G(z) = \frac{\left(K_p + K_I \times \frac{T}{2} + \frac{K_D}{T} \right) + \left(-K_p + K_I \times \frac{T}{2} - 2 \times \frac{K_D}{T} \right) \times z^{-1} + \frac{K_D}{T} \times z^{-2}}{1 - z^{-1}}$$

Đặt:

$$a_0 = K_p + K_I \times \frac{T}{2} + \frac{K_D}{T}$$

$$a_1 = -K_p + K_I \times \frac{T}{2} - 2 \times \frac{K_D}{T}$$

$$a_2 = \frac{K_D}{T}$$

Suy ra:

$$G(z) = \frac{a_0 + a_1 \times z^{-1} + a_2 \times z^{-2}}{1 - z^{-1}}$$

Từ đó, ta tính được tín hiệu điều khiển $\mathbf{u}(k)$ khi tín hiệu vào $\mathbf{e}(k)$ như sau:

$$u(k) = G(z) \times e(k) = \frac{a_0 + a_1 \times z^{-1} + a_2 \times z^{-2}}{1 - z^{-1}} \times e(k)$$

Suy ra:

$$u(k) = u(k-1) + a_0 \times e(k) + a_1 \times e(k-1) + a_2 \times e(k-2)$$

Trong đó: **T** là thời gian lấy mẫu

e(k) là sai số hiện tại: **e(k) = giá trị đặt – giá ngõ ra**

e(k-1) và **e(k-2)** là các sai số tích lũy

u(k) là ngõ ra bộ điều khiển

K_P, K_I, K_D là các thông số P, I, D của bộ điều khiển

Cách điều chỉnh các hệ số K_P, K_I, K_D:

Đầu tiên, ta chỉnh cho K_P = 1 và K_D, K_I = 0 nghĩa là chỉ điều khiển P. Sau đó, tăng K_P lên dần dần và quan sát động cơ. Khi thấy động cơ dao động (nghĩa là nó lúc nhanh lúc chậm), ta lấy K_P tại đó nhân với 0.6 để tính toán: K_{P_t} = 0.6 × K_{P_dao_dong}. Sau đó, ta tăng K_D lên dần dần (giữ nguyên K_P), nếu K_D quá lớn sẽ làm cho động cơ bị dao động mạnh. Giảm K_D lại cho đến khi động cơ hết dao động. Điều chỉnh K_I là khó nhất, bắt đầu từ giá trị rất nhỏ (ví dụ lấy K_I = 1/K_D chẳng hạn). Hệ số K_I không cần lớn vì động cơ tự nó đã chứa thành phần K_I, thể hiện ở moment quán tính hay sức ì của động cơ. Do đó, thường với đối tượng điều khiển là nhiệt độ hay động cơ (các đối tượng có quán tính) thì chỉ cần bộ điều khiển PD là đủ.

Chương 7: Tính toán tọa độ Robot và Kinect

Nội dung chính

7.1 Các phép chuyển đổi hệ trực tọa độ cơ bản

7.2 Tính toán tọa độ robot

7.3 Tính toán tọa độ Kinect

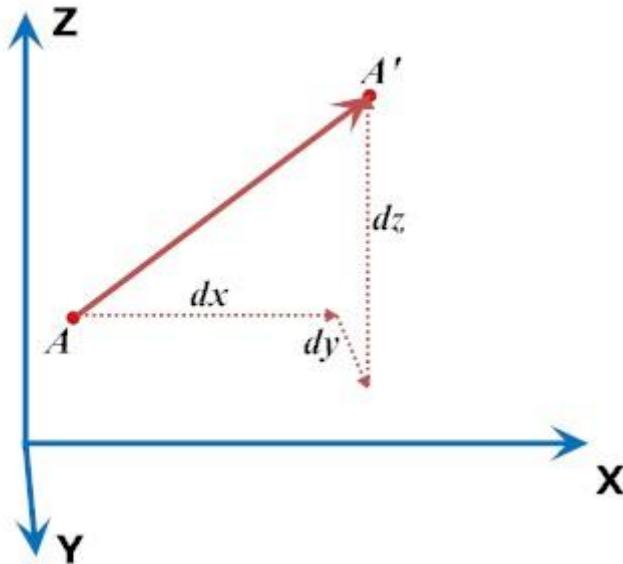
7.1 Các phép chuyển đổi hệ tọa độ cơ bản

Trước khi đi vào các phép biến đổi, ta làm quen với khái niệm “hệ tọa độ thuần nhất”. Trong hệ tọa độ thuần nhất, mỗi điểm (x, y, z) trong hệ tọa độ Descartes được biểu diễn bởi một bộ bốn giá trị trong không gian bốn chiều thu gọn (h_x, h_y, h_z, h) . Để tiện lợi cho tính toán, người ta thường chọn $h = 1$. Như vậy, một điểm (x, y, z) trong hệ tọa độ Descartes sẽ biến thành điểm $(x, y, z, 1)$ trong hệ tọa độ thuần nhất; còn điểm (x, y, z, w) trong hệ tọa độ thuần nhất (với $w \neq 0$) sẽ tương ứng với điểm $(x/w, y/w, z/w)$ trong hệ tọa độ Descartes [20].

Phép biến đổi ba chiều biến điểm P thành điểm Q có dạng: $Q = P.M$. Trong đó: $Q = (x', y', z', 1)$, $P = (x, y, z, 1)$, $d = (dx, dy, dz)$ là vector tịnh tiến và M là ma trận biến đổi 4×4 trong hệ tọa độ thuần nhất:

$$M = \begin{bmatrix} a & b & c & 0 \\ d & e & f & 0 \\ g & h & i & 0 \\ dx & dy & dz & 1 \end{bmatrix}$$

- **Phép tịnh tiến:**



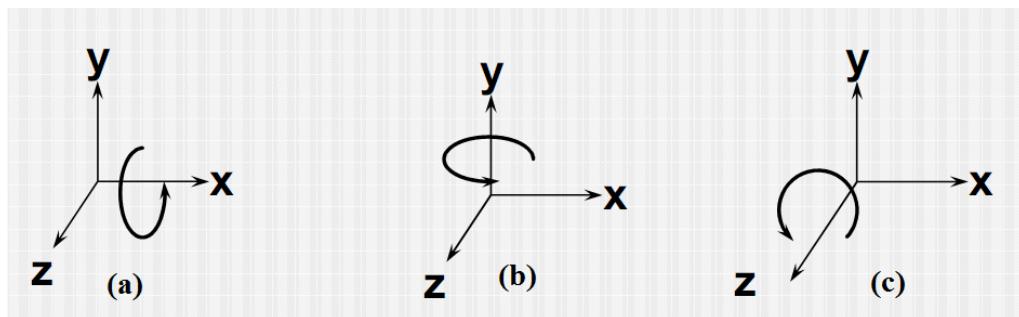
Hình 7.1: Phép tịnh tiến

Vector tịnh tiến trong phép biến đổi ba chiều có một tác động rất trực quan: mỗi điểm được dịch đi một khoảng là dx, dy, dz theo ba trục. Ma trận M cho phép tịnh tiến có dạng như sau:

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ dx & dy & dz & 1 \end{bmatrix}, Q = [x' \ y' \ z' \ 1] = P.M = [x+dx \ y+dy \ z+dz \ 1].$$

▪ Phép quay:

Ta khảo sát phép quay quanh một trục tọa độ. Khác với phép quay trong hai chiều quanh một điểm bất kì, trong ba chiều ta có phép quay quanh một trục tọa độ. Ta có các ma trận biểu diễn các phép quay quanh trục x, y, z ngược chiều kim đồng hồ với góc tương ứng β , φ , θ lần lượt là $R_x(\beta)$, $R_y(\varphi)$, $R_z(\theta)$:



Hình 7.2: Phép quay

$$R_x(\beta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \beta & \sin \beta \\ 0 & -\sin \beta & \cos \beta \end{bmatrix}$$

$$R_y(\varphi) = \begin{bmatrix} \cos \varphi & 0 & -\sin \varphi \\ 0 & 1 & 0 \\ \sin \varphi & 0 & \cos \varphi \end{bmatrix}$$

$$R_z(\theta) = \begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

7.2 Tính toán tọa độ robot

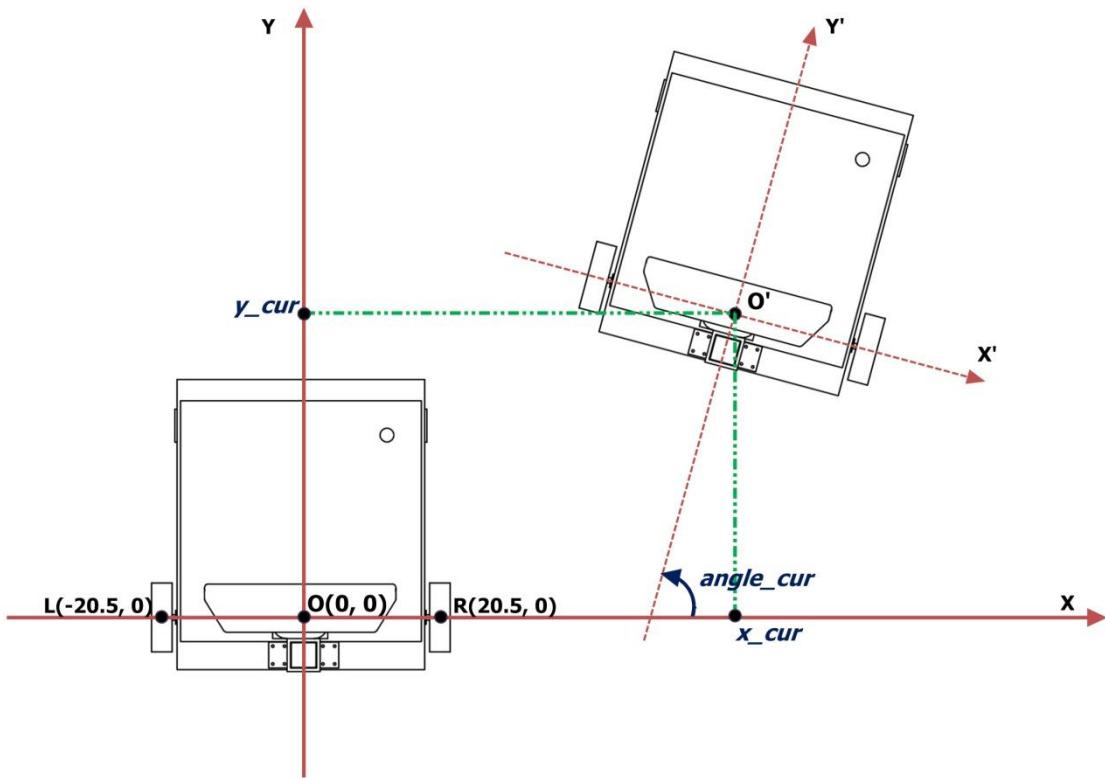
Để dễ hình dung, ta quan sát mô hình robot được vẽ trên Autodesk Inventor 2012. Với thiết kế với bốn bánh giúp robot giữ thăng bằng tốt, hai bánh sau làm nhiệm vụ

truyền động, ta đặt gốc tọa độ robot tại vị trí chính giữa hai tâm các bánh này (gọi là tâm robot).



Hình 7.3: Mô hình robot

Quan sát hình chiếu bằng của mô hình robot ở hình 7.4, tọa độ gốc của robot được chọn là tâm $O(0, 0)$. Vị trí tương ứng của tâm hai bánh xe trái và phải là $R(20.5, 0)$, $L(-20.5, 0)$ (đơn vị là centimet). Trong quá trình di chuyển, tọa độ của robot sẽ được cập nhật so với vị trí lúc khởi động khi robot đến một vị trí mới. Các giá trị cập nhật bao gồm: tọa độ x hiện tại (x_{cur}), tọa độ y hiện tại (y_{cur}) và góc hiện tại (trục $O'Y'$ so với trục OX , angle_{cur}). Như vậy, tại thời điểm bắt đầu khởi động: $x_{cur} = 0$, $y_{cur} = 0$ và $\text{angle}_{cur} = 90^\circ$.



Hình 7.4: Tọa độ robot (quan sát từ trên xuống)

Khi di chuyển robot sẽ có hai động tác chính: đi thẳng (tiến hoặc lùi) một đoạn L và xoay một góc α quanh tâm robot. Bài toán chỉ đơn thuần áp dụng các phép tính tiến và xoay cơ bản trong không gian 2D. Phương trình tính toán tọa độ mới của robot cho từng động tác:

- **Đi thẳng một đoạn L :**

$$\begin{aligned} x_{\text{new}} &= x_{\text{cur}} \pm L \cdot \cos(\text{angle}_{\text{cur}}) \\ y_{\text{new}} &= y_{\text{cur}} \pm L \cdot \sin(\text{angle}_{\text{cur}}) \\ \text{angle}_{\text{new}} &= \text{angle}_{\text{cur}} \end{aligned}$$

Dấu “+” cho trường hợp tiến và “-” cho trường hợp lùi.

- **Xoay một góc α :**

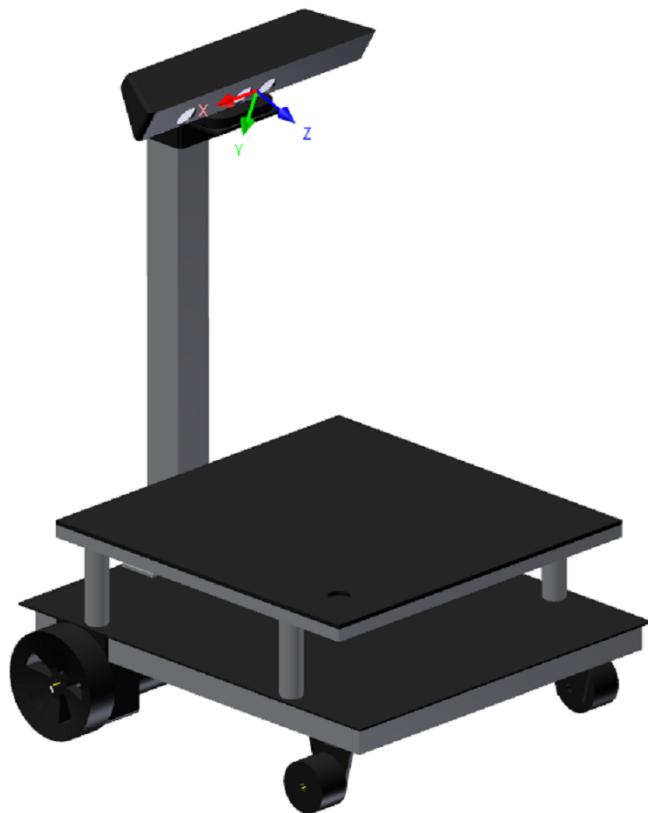
$$\begin{aligned} x_{\text{new}} &= x_{\text{cur}} \\ y_{\text{new}} &= y_{\text{cur}} \\ \text{angle}_{\text{new}} &= \text{angle}_{\text{cur}} + \alpha \end{aligned}$$

anpha âm khi robot xoay sang phải (thuận chiều kim đồng hồ) và dương khi xoay sang trái (ngược chiều kim đồng hồ). Giá trị **angle_cur** nằm trong khoảng $0 \div 2 * \text{PI}$ nên lưu ý khi góc này nằm ngoài khoảng này thì:

$$\text{angle_cur} < 0 : \text{angle_cur} = \text{angle_cur} + 2 * \text{PI}$$

$$\text{angle_cur} > 2 * \text{PI} : \text{angle_cur} = \text{angle_cur} - 2 * \text{PI}$$

7.3 Tính toán tọa độ Kinect



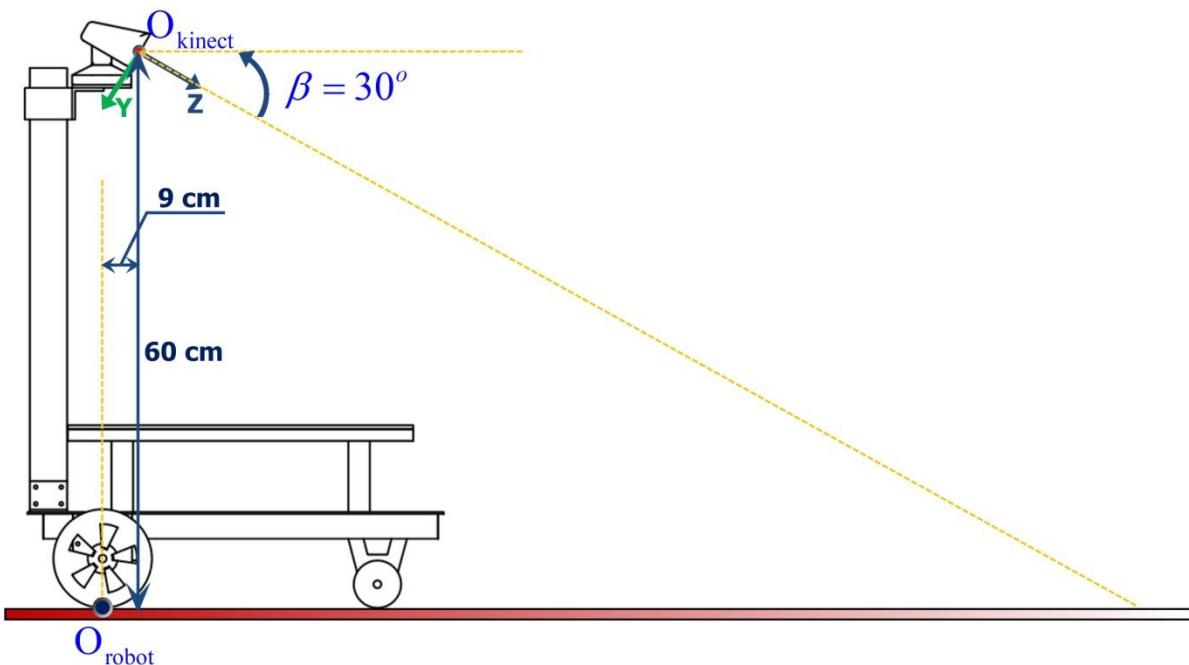
Hình 7.5: Hệ trục tọa độ Kinect

Thông tin về môi trường trong quá trình di chuyển của robot được Kinect đưa về xử lý ở dạng 3D nhằm đưa ra những lệnh truyền cho robot một cách chính xác nhất. Với sự hỗ trợ mạnh mẽ của thư viện Point Cloud, Kinect cho ta những thông tin chính xác về tọa độ các điểm hay vật thể trong môi trường thực tế so với tọa độ gốc tại Kinect. Một nhược điểm lớn của Kinect đó là vùng mù (Kinect không thể nhìn thấy) nằm trong khoảng $0 \div 0.50$ mét, là một khoảng cách lớn đối với chức năng tránh vật

cản của robot. Chính vì lý do đó mà ta bố trí Kinect được đặt trên cao và hướng xuống với một góc nghiêng nhất định, đồng thời được đặt lùi phía sau robot để nhìn được một cách tổng quan nhất môi trường phía trước robot.

Chính điều này đặt ra bài toán chuyển đổi hệ trực tọa độ của Kinect về hệ trực tọa độ của robot để đồng bộ hóa. Có chút khác biệt giữa hai hệ trực: đối với hệ trực tọa độ Kinect, phương Z đại diện cho khoảng cách tới vật thể, phương Y đại diện cho chiều cao vật thể lần lượt tương ứng với phương Y và Z của hệ trực tọa độ robot. Phương X còn lại không có gì thay đổi.

Mục đích của ta là đưa mặt phẳng OXZ của Kinect về trùng với mặt phẳng OXY của robot. Quan sát hình 7.6, ta thấy ngoài việc nghiêng một góc 30° , Kinect còn lệch theo chiều Z một đoạn 9 cm và chiều Y một đoạn 60 cm.



Hình 7.6: Đồng nhất hệ trực tọa độ Kinect và robot

Ta lần lượt thực hiện các công việc sau đối với hệ trực Kinect:

- Xoay mặt phẳng OYZ quanh trục X ngược chiều kim đồng hồ một góc 30° .

$$\begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \end{bmatrix} = \begin{bmatrix} X & Y & Z \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \beta & \sin \beta \\ 0 & -\sin \beta & \cos \beta \end{bmatrix} = \begin{bmatrix} X \\ Y \cos \beta - Z \sin \beta \\ Y \sin \beta + Z \cos \beta \end{bmatrix}$$

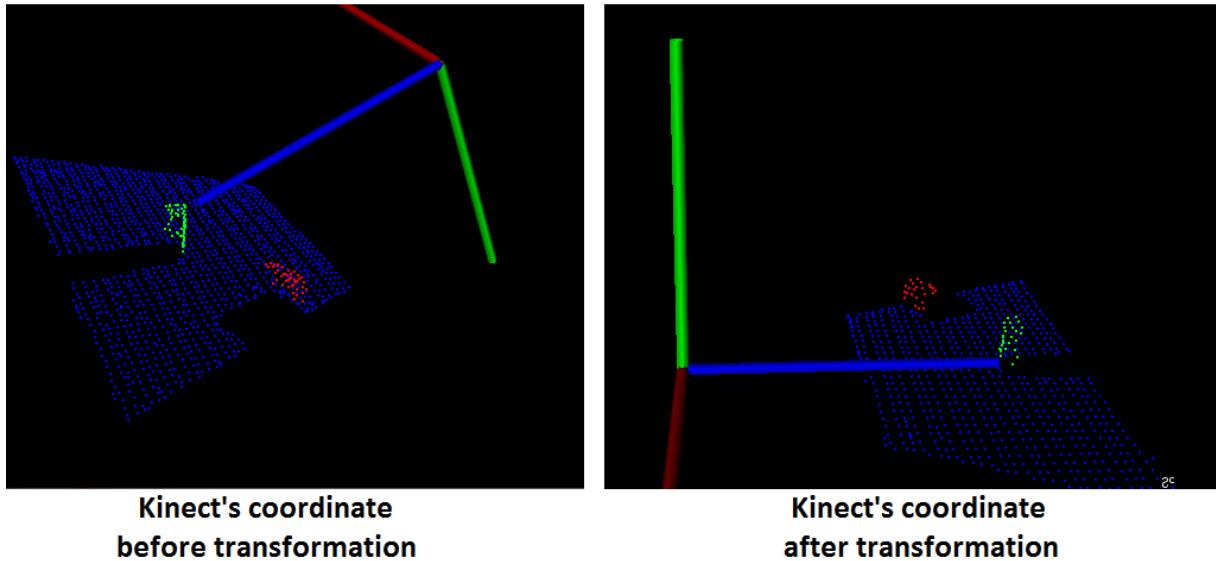
- Tịnh tiến trục Y xuống một đoạn 60 cm và tịnh tiến trục Z về sau một đoạn 9 cm.

$$\begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \end{bmatrix} = \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \end{bmatrix} + \begin{bmatrix} 0 \\ -60 \\ 9 \end{bmatrix} = \begin{bmatrix} X \\ Y \cos \beta - Z \sin \beta - 60 \\ Y \sin \beta + Z \cos \beta + 9 \end{bmatrix}$$

- Đảo chiều Y.

$$\begin{bmatrix} X_3 \\ Y_3 \\ Z_3 \end{bmatrix} = \begin{bmatrix} X_2 \\ -Y_2 \\ Z_2 \end{bmatrix} = \begin{bmatrix} X \\ -Y \cos \beta + Z \sin \beta + 60 \\ Y \sin \beta + Z \cos \beta + 9 \end{bmatrix}$$

Hình 7.7 cho ta thấy hệ trục Kinect sau khi thực hiện các phép chuyển đổi. Hệ trục có các chiều X, Y, Z tương ứng với các màu đỏ, xanh lá, xanh dương.



Hình 7.7: Tọa độ Kinect trước (trái) và sau (phải) khi chuyển trục

Chương 8: Chương trình điều khiển

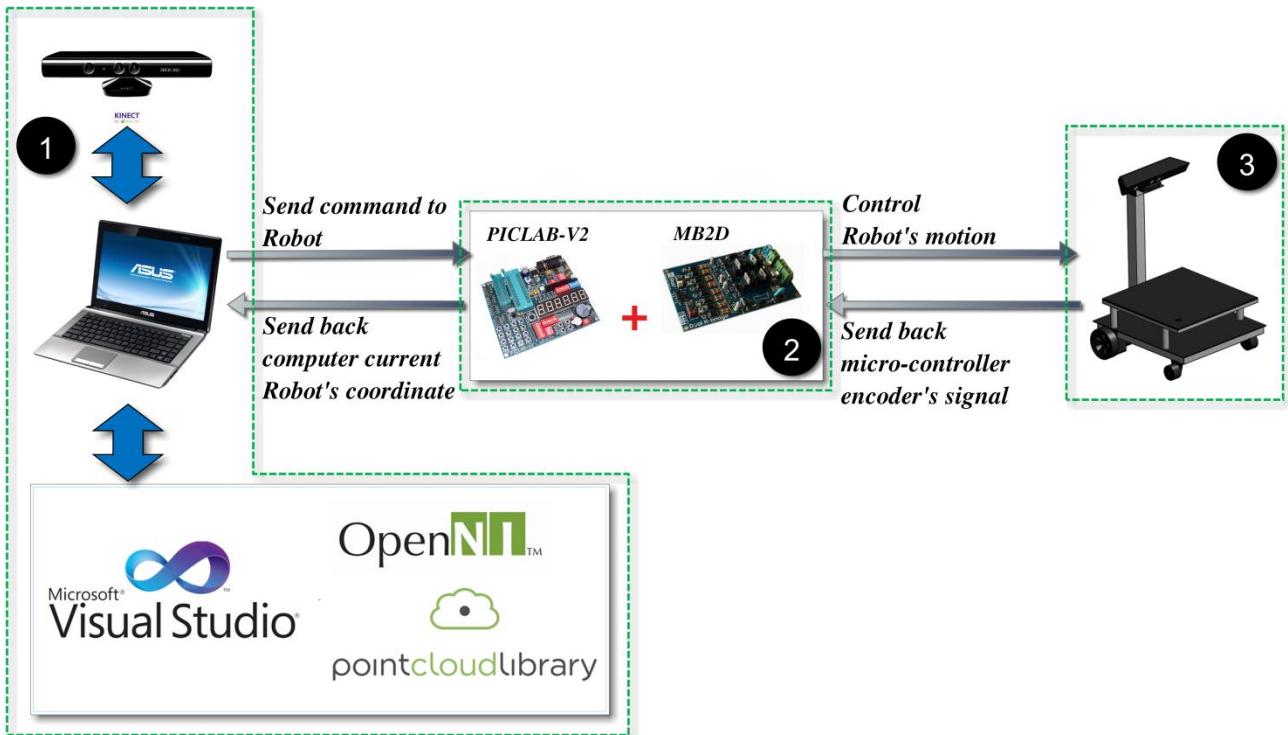
Nội dung chính

8.1 Nội dung chương trình điều khiển

8.2 Giải thuật chương trình do máy tính xử lý

8.3 Giải thuật chương trình do vi điều khiển xử lý

8.1 Nội dung chương trình điều khiển



Hình 8.1: Nội dung chương trình điều khiển

Hình 8.1 khái quát cho ta các khối chính và nhiệm vụ của các khối. Máy tính kết hợp Kinect cùng các công cụ lập trình tạo nên khối số 1, mạch vi điều khiển và mạch công suất tạo nên khối số 2, khối số 3 là mô hình cơ khí của mobile robot.

Khối 1 đóng vai trò như cặp mắt và bộ não của robot, giúp robot phân tích không gian phía trước và truyền lệnh thích hợp xuống khối số 2. Khối 2 sẽ thực hiện điều khiển robot (khối 3) theo đúng yêu cầu của khối 1, các động tác như quay trái, phải hay chạy tiến, lùi với một giá trị xác định. Sau khi tính toán tọa độ từ tín hiệu phản hồi encoder trên robot, giá trị vị trí hiện tại sẽ được khối 2 truyền lại cho khối 1 cập nhật và hiển thị.

8.2 Giải thuật chương trình do máy tính xử lý

Máy tính thực hiện công việc điều khiển bằng ba luồng xử lý song song: xử lý Kinect, xử lý kế hoạch di chuyển của robot và xử lý giao tiếp với vi điều khiển.

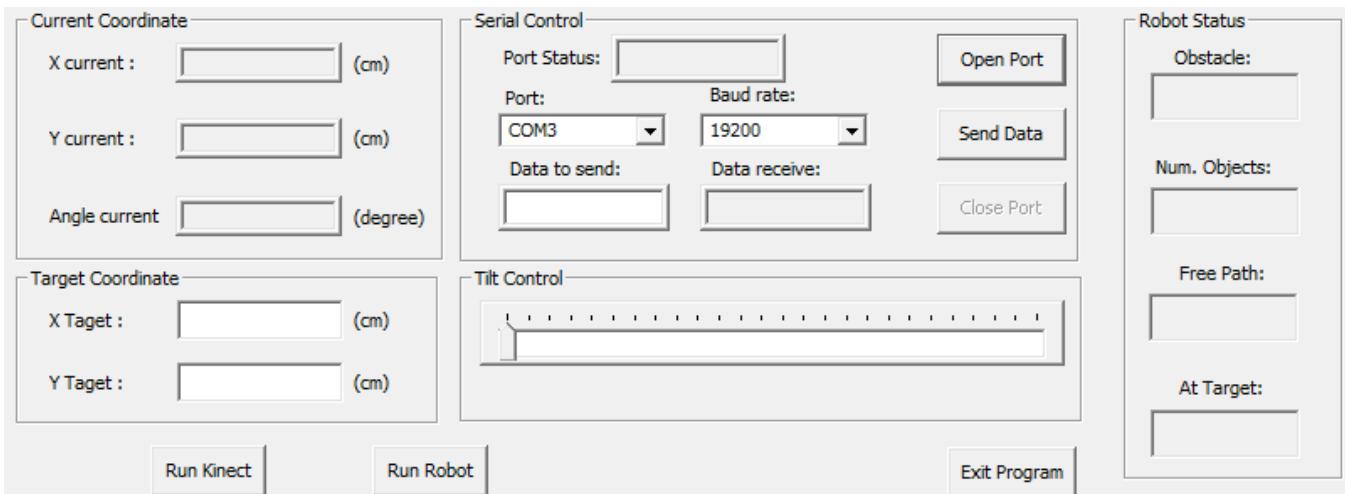


Hình 8.2: Xử lý đa tiến trình

Luồng Kinect (số 1) làm nhiệm vụ phân tích môi trường, đưa ra các thông tin về kích thước, vị trí vật cản nếu xuất hiện trước robot, đã được trình bày ở mục 4.3.

Luồng giao tiếp với vi điều khiển (số 3) theo chuẩn RS232 làm nhiệm vụ truyền lệnh xuống và nhận thông tin vị trí lên từ vi điều khiển.

Luồng xử lý kế hoạch di chuyển của robot sử dụng thông tin từ luồng số 1 và luồng số 3, điều khiển robot di chuyển hợp lý về đích.



Hình 8.3: Giao diện chương trình điều khiển

Hình 8.3 là giao diện điều khiển viết bằng MFC trong bộ Visual Studio 2010 của Microsoft. Vị trí đích cần di chuyển đến sẽ được nhập vào khía **Target Coordinate**; tọa độ hiện tại của robot sẽ tự động hiện thị trong khía **Current Coordinate**; tùy chọn

các thông số cho giao tiếp RS232, trạng thái cổng giao tiếp được đặt trong khối **Serial Control**; trong khối **Robot Status** hiển thị các thông tin về *cò phát hiện vật cản, cò phát hiện đường trống*, số lượng vật cản và trạng thái robot đã ở vị trí đích hay chưa. **Tilt Control** điều khiển góc ngẳng Kinect.

➤ **Định dạng điều khiển qua giao tiếp RS232**

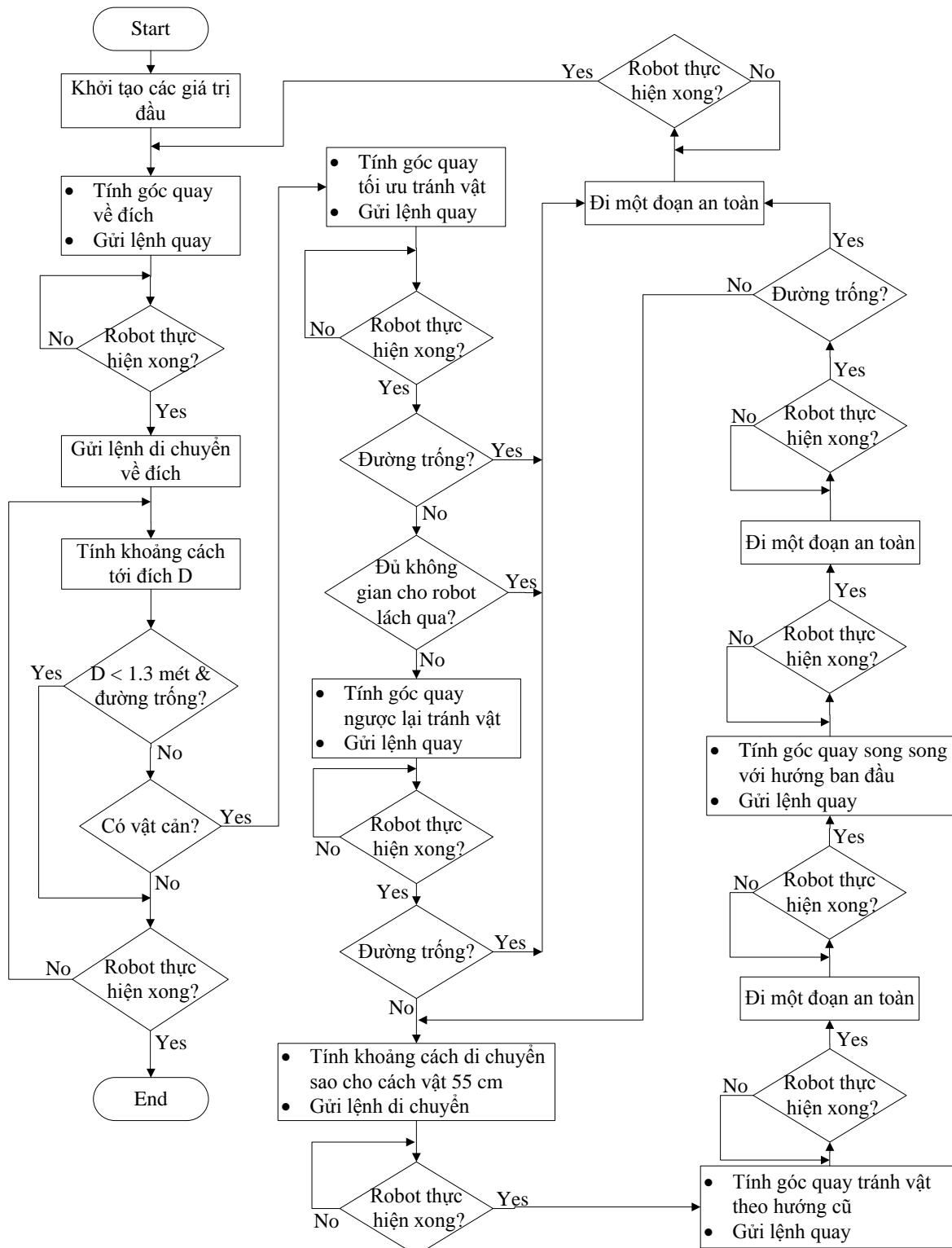
Lệnh điều khiển truyền xuống qua giao tiếp nối tiếp được định dạng theo mẫu sau: *[kí tự điều khiển][giá trị điều khiển]#, trong đó:

- **kí tự điều khiển:** R(quay phải), L(quay trái), F(đi thẳng), T(đi thẳng về đích).
- **giá trị điều khiển:** là giá trị góc quay hay quãng đường di chuyển, định dạng kiểu dữ liệu double, lấy ba chữ số thập phân, đơn vị là centimet.
- **‘*’, ‘#’:** là các ký tự đặc biệt cho biết ký tự bắt đầu và kết thúc chuỗi.

Lệnh nhận lên từ vi điều khiển có dạng: X[giá trị 1]Y[giá trị 2]A[giá trị 3]N, trong đó:

- **X, Y, A:** lần lượt là ký tự bắt đầu cho các giá trị đại diện tọa độ X, Y, góc hiện tại của robot. **N** là ký tự kết thúc chuỗi.
- **giá trị 1 ÷ 3:** tương ứng là giá trị tọa độ, góc hiện tại của robot.

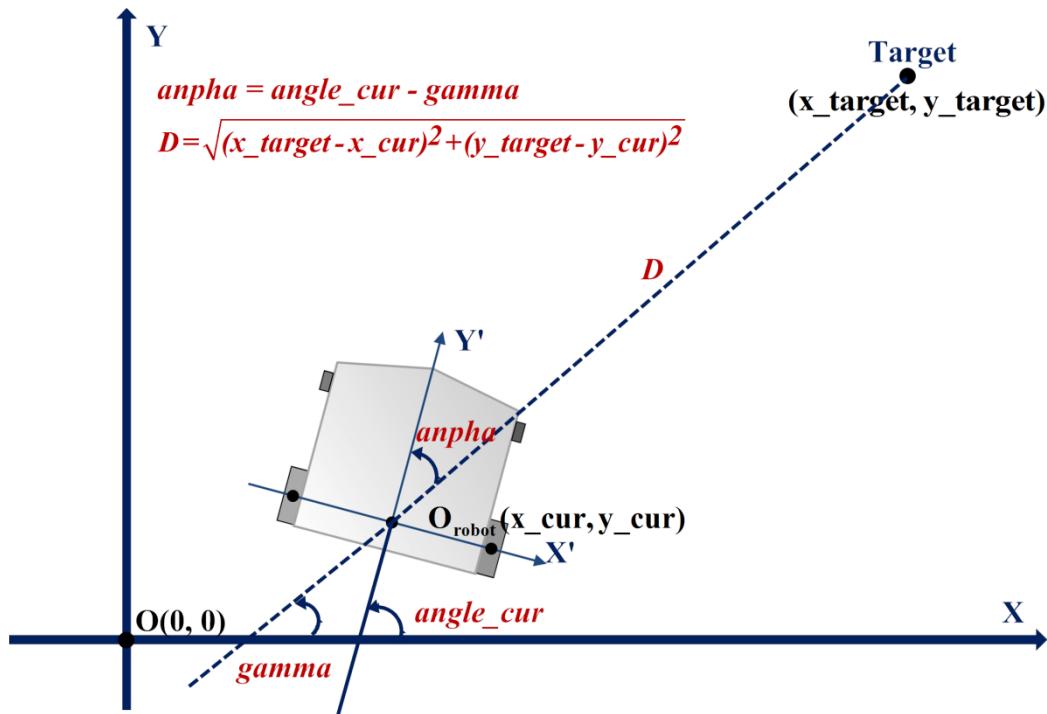
➤ **Giải thuật điều khiển robot ứng xử với vật cản trên đường đi đến đích:**



Hình 8.4: Sơ đồ giải thuật điều khiển robot do máy tính xử lý

Một số hàm chú ý:

- **Tính góc quay về đích:**



Hình 8.5: Tính góc quay về đích

Hình 8.5 mô tả một trường hợp đích nằm bên phải robot, lúc này robot sẽ quay một góc **anpha** về bên phải để hướng về đích. Ta có cách tính cho các trường hợp như sau (ta giả sử y_{cur} luôn dương, robot luôn di chuyển về phía trước):

$$\gamma = \arctan\left(\frac{y_{target} - y_{cur}}{x_{target} - x_{cur}}\right)$$

✓ **$\gamma < 0$ (đích nằm bên trái robot):**

- **$0 \leq angle_cur < PI + \gamma$:**
 - + $\alpha = PI - angle_cur + \gamma$
 - + Quay về bên TRÁI robot
- **$PI + \gamma \leq angle_cur < 2*PI + \gamma$:**
 - + $\alpha = angle_cur - (PI + \gamma)$

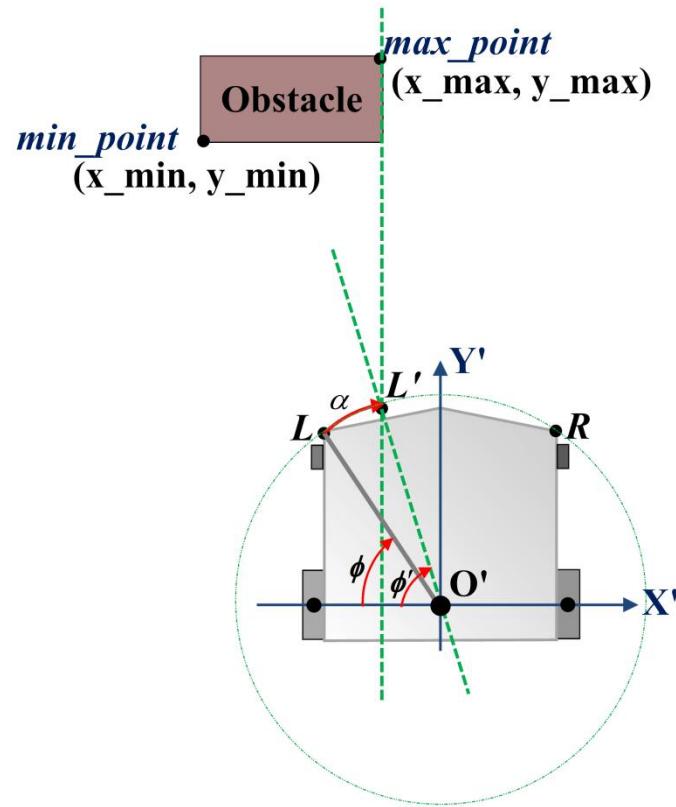
- + Quay về bên PHẢI robot
- **$2*PI + gamma \leq angle_cur \leq 2*PI:$**
 - + anpha = $3*PI - angle_cur + gamma$
 - + Quay về bên TRÁI robot
- ✓ **$gamma \geq 0$ (đích nằm bên phải robot):**
 - **$0 \leq angle_cur < gamma:$**
 - + anpha = $gamma - angle_cur$
 - + Quay về bên TRÁI robot
 - **$gamma \leq angle_cur < PI + gamma:$**
 - + anpha = $angle_cur - gamma$
 - + Quay về bên PHẢI robot
 - **$PI + gamma \leq angle_cur \leq 2*PI:$**
 - + anpha = $2*PI - angle_cur + gamma$
 - + Quay về bên TRÁI robot

- **Tính góc quay tối ưu tránh vật:**

Mục đích là điều khiển robot theo hướng có góc quay nhỏ hơn khi tránh vật cản.

Các trường hợp có thể xảy ra là:

- Vật cản nằm bên trái robot

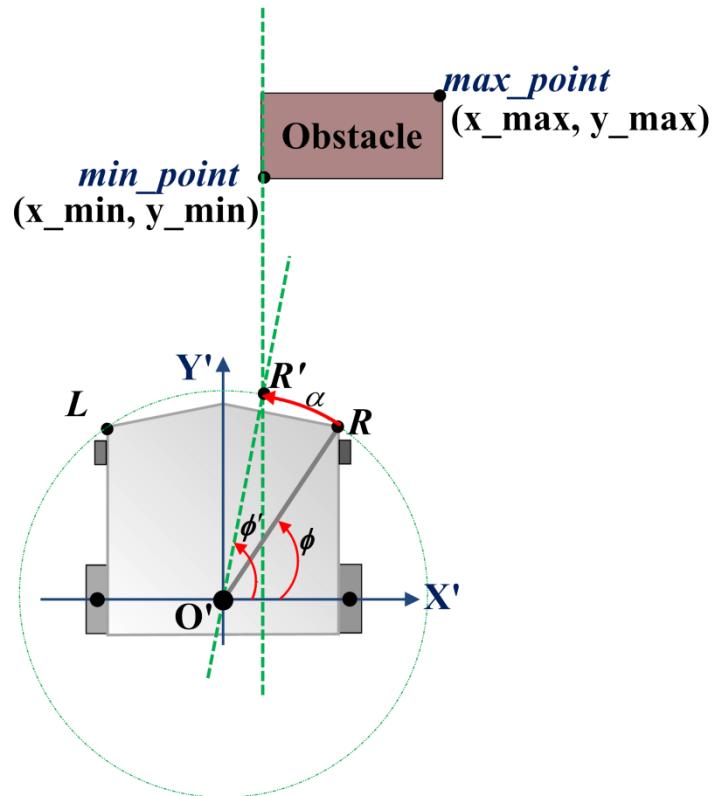


Hình 8.6: Vật cản bên trái robot

Quan sát hình 8.6, góc quay cần thiết để thoát khỏi vật cản là **anpha**(α) khi robot ở vị trí cách vật cản một đoạn. Góc **anpha** hoàn toàn tính được khi ta biết được tọa độ **max_point**, góc ϕ và độ lớn $O'L$.

$$\alpha = \phi' - \phi = \cos^{-1}\left(\frac{-x_{max}}{O'L}\right) - \phi, \text{ hướng quay về phía bên phải robot.}$$

- Vật cản nằm bên phải robot

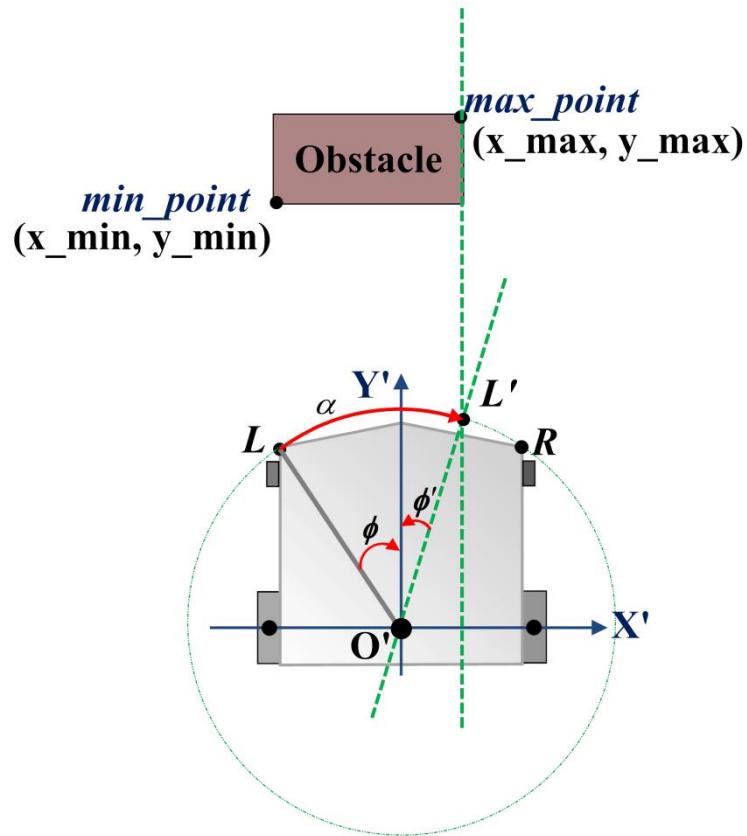


Hình 8.7: Vật cản bên phải robot

Tương tự trường hợp vật cản nằm bên trái, góc **anpha** được tính như sau:

$$\alpha = \phi' - \phi = \cos^{-1}(x_{\text{min}} / O'L) - \phi, \text{ hướng quay về phía bên trái robot.}$$

- Vật cản nằm ở giữa đường robot



Hình 8.8: Vật cản nằm ở giữa đường di chuyển của robot

Trong trường hợp này, robot sẽ tìm góc quay né vật sao cho **anpha** nhỏ nhất; lúc đó, nếu độ lớn vật cản nhiều hơn về phía bên trái thì robot quay về bên phải (như trên hình 8.8) và ngược lại. Công thức tính trong từng trường hợp: (trong đó góc ϕ và $O'L = O'L'$ biết trước)

Quay trái:

$$\alpha = \phi + \phi' = \phi + \sin^{-1} \left(\frac{-x_{\min}}{O'L} \right)$$

Quay phải:

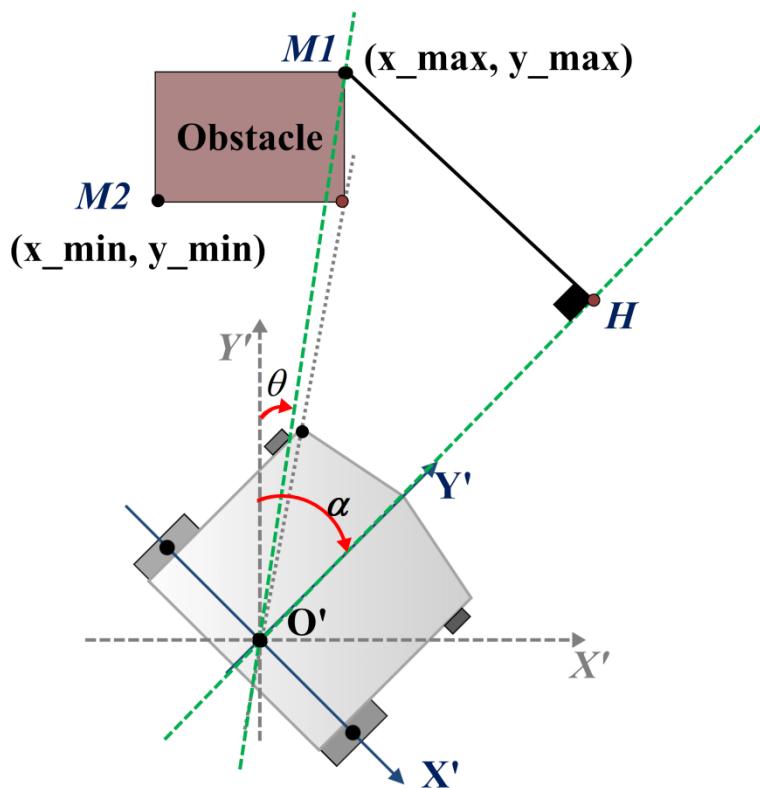
$$\alpha = \phi + \phi' = \phi + \sin^{-1} \left(\frac{x_{\max}}{O'L} \right)$$

- **Tính góc quay ngược lại tránh vật:**

Như các trường hợp quay một góc xác định né vật cản, nhưng thay vì quay theo hướng có góc quay nhỏ hơn thì robot sẽ quay theo hướng ngược lại. Công thức tính hoàn toàn tương tự.

- **Đi một đoạn an toàn:**

Sau khi quay một góc tránh vật, robot sẽ di chuyển tiếp một đoạn để thoát hoàn toàn khỏi vật.



Hình 8.9: Đi một đoạn an toàn về phía phải vật cản

Quãng đường di chuyển an toàn để thoát khỏi vật cản bằng đoạn O'H cộng thêm một giá trị cố định vừa đủ **deltaMove**(ở đây chọn **deltaMove = 15 cm** dựa trên thực nghiệm).

Đi bên phải vật cản:

$$OH' = OM_1 \times \cos(\alpha - \theta)$$

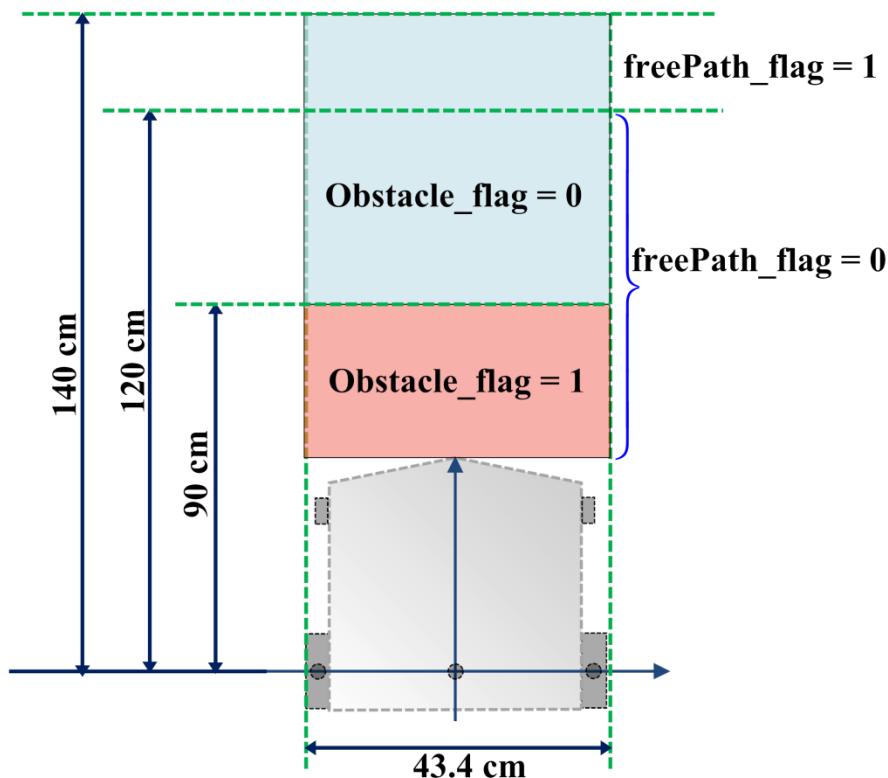
$$= \sqrt{(x_{\max} - x_{\text{cur}})^2 + (y_{\max} - y_{\text{cur}})^2} \times \cos(\alpha - \tan^{-1}(\frac{|x_{\max}|}{y_{\max}}))$$

Đi bên trái vật cản:

$$OH' = OM_1 \times \cos(\alpha - \theta)$$

$$= \sqrt{(x_{\min} - x_{\text{cur}})^2 + (y_{\max} - y_{\text{cur}})^2} \times \cos(\alpha - \tan^{-1}(\frac{|x_{\min}|}{y_{\max}}))$$

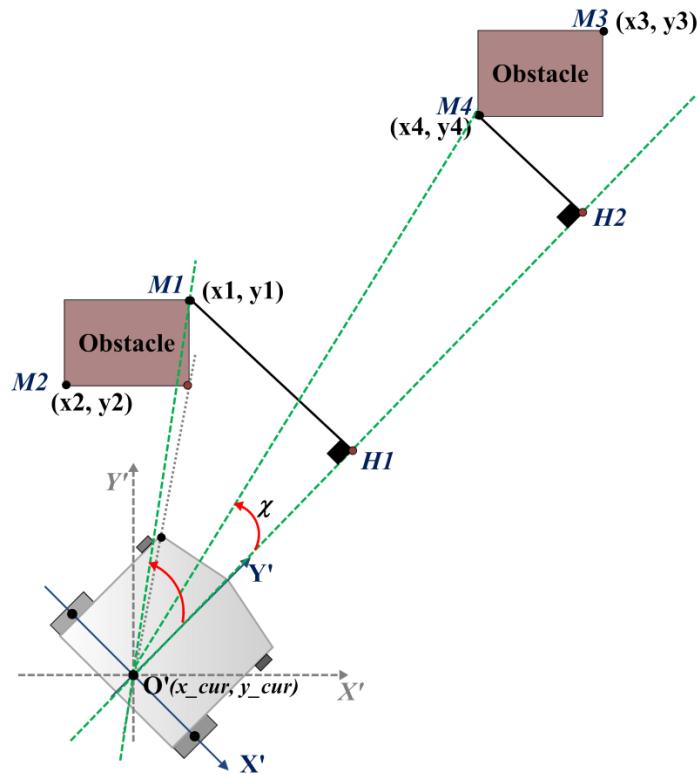
- **Cờ báo có vật cản (Obstacle_flag) và đường trống (freePath_flag):**



Hình 8.10: Cờ báo có vật cản và đường trống

Hình 8.10 biểu diễn không gian không xuất hiện vật cản phía trước robot, lúc đó các cờ tương ứng sẽ bật lên trong các vùng xác định. Tầm nhìn xa của robot giới hạn ở khoảng cách 140 cm.

- **Cờ báo đủ không gian cho robot lách qua:**



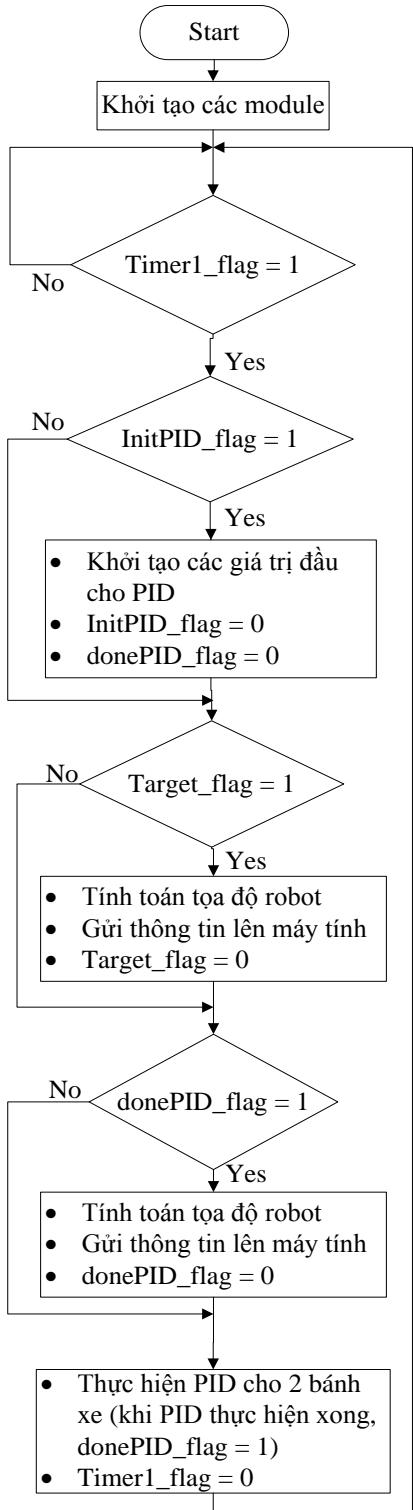
Hình 8.11: Không gian cho robot lách qua

Cờ báo đủ không gian cho robot lách qua sẽ bật lên 1 khi khoảng cách H_1H_2 (hình 8.11) đủ lớn cho robot lọt qua (không gian đủ cho robot lọt qua khi $H_1H_2 > 50$ cm). Trong đó $O'H_1$ được tính như trong hàm *đi một đoạn an toàn*, $O'H_2$ tương tự, ta có:

$$\begin{aligned}
 O'H_2 &= O'M_4 \times \cos(\chi) \\
 &= \sqrt{(x_4 - x_{cur})^2 + (y_4 - y_{cur})^2} \times \cos(\tan^{-1}(\frac{|x_4|}{y_4})) \\
 \Rightarrow H_1H_2 &= O'H_2 - O'H_1
 \end{aligned}$$

8.3 Giải thuật chương trình do vi điều khiển xử lý

Công việc điều khiển động cơ và tính toán tọa độ, thực hiện giao tiếp với máy tính sẽ được PIC 18F4550 xử lý. Để thực hiện các công việc này, PIC18F4550 kết hợp thông tin lệnh từ máy tính và tín hiệu hồi tiếp về từ encoder thông qua các ngắt ngoài. Sau đây là sơ đồ khái mô tả hoạt động của chương trình chính xử lý bởi PIC18F4550:



Hình 8.12: Sơ đồ giải thuật trên vi điều khiển

▪ ***Khởi tạo các module:***

Khởi tạo các module sử dụng trên PIC: các port I/O, timer, ngắt ngoài, PWM và giao tiếp RS232.

▪ ***Timer1_flag:***

Cờ báo bằng 1 khi timer1 tràn sau 5 ms, phục vụ cho việc lấy mẫu, tính toán PID.

▪ ***InitPID_flag:***

Cờ báo khởi tạo PID, bằng 1 khi nhận được lệnh từ máy tính.

▪ ***Khởi tạo các giá trị đầu cho PID:***

Khởi tạo các giá trị ban đầu cho tính toán PID: các thông số k_p , k_i , k_d ; giá trị đặt.

▪ ***Target_flag:***

Cờ báo lên 1 khi robot đang di chuyển về đích và gặp vật cản.

▪ ***Tính toán tọa độ robot:***

Tính toán tọa độ robot, đã được trình bày tại mục 7.2.

▪ ***Gửi thông tin lên máy tính:***

Thông tin về tọa độ và góc hiện tại của robot: (x_{cur} , y_{cur} , $angle_{cur}$).

▪ ***Thực hiện PID trên hai bánh xe:***

Dựa trên giải thuật PID vị trí đã được trình bày ở mục 6.2. PID được thực hiện xong khi sai số nằm trong khoảng cho phép (xấp xỉ bằng không), lúc đó $donePID_flag$ bằng 1.

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Với mục đích xây dựng mô hình robot có khả năng di chuyển đến điểm đích xác định trước, tránh chướng ngại vật và đến đích an toàn. Nhóm đã áp dụng công nghệ xử lý ảnh 3D cho phần thị giác của robot với thiết bị chơi game Kinect, đáp ứng được độ tin cậy cao hơn so với các cảm biến truyền thống. Quá trình thực hiện đã hoàn thành những nhiệm vụ sau:

- Thiết kế và thi công mô hình mobile robot hoàn chỉnh.
- Chương trình xử lý ảnh khôi phục được không gian phía trước robot dưới dạng 3D, cung cấp đầy đủ thông tin về môi trường cho robot.
- Tốc độ xử lý ảnh trên máy tính khoảng 13-15 fps, đủ đáp ứng thời gian thực cho robot.
- Hoàn thành kế hoạch di chuyển đến đích có vật cản với sai số chấp nhận được ở môi trường trong nhà.
- **Một số điểm cần khắc phục:**



**Self-made
robot**



iRobot

- Phần cơ khí của mobile robot không được hoàn hảo nên ảnh hưởng đến sai số khi di chuyển. Ở điểm này, một con robot omni sẽ đáp ứng đầy đủ cho ứng dụng robot di chuyển trong nhà với sự linh hoạt, kết cấu gọn nhẹ và sự chính xác cao.

Một con robot rất đáng quan tâm là iRobot với các lựa chọn khác nhau, giá dao động từ 130\$ ÷ 300\$, chi tiết xem tại: <http://store.irobot.com>

- Phần thị giác robot chỉ quan sát được không gian phía trước robot nên robot có thể sẽ đụng vật cản bên hông hay phía sau khi di chuyển lùi. Vấn đề này có thể được giải quyết bằng biện pháp bị thêm các cảm biến phát hiện vật cản xung quanh robot; hoặc thiết kế thêm hệ thống xoay cho phần thị giác, thị giác sẽ xoay một góc xác định trước khi quyết định di chuyển.
- Tích hợp thêm các cảm biến định vị cần thiết để tăng sự chính xác.
- Chương trình được viết trên môi trường Windows với sự hỗ trợ từ thư viện Point Cloud còn nhiều hạn chế, nhất là ở tốc độ xử lý. Sẽ tối ưu nhất nếu viết trên môi trường Linux và nếu phát triển thành sản phẩm thương mại sẽ hạ được giá thành phần mềm xuống, tăng tính cạnh tranh.

- **Hướng phát triển:**

- Mục đích của đề tài tạo ra nền tảng cho việc xây dựng một mô hình robot dịch vụ hoàn chỉnh: một robot có khả năng làm các công việc thay cho con người như: bưng bê đồ ăn, lau nhà, hướng dẫn khách hàng, ...
- Bên cạnh đó, với sức mạnh của thiết bị Kinect có thể giúp ta xây dựng được bản đồ (mapping) dạng 3D. Và việc tích hợp lên robot sẽ giúp ta xây dựng được bản đồ ở những khu vực mà con người không thể vào được.

TÀI LIỆU THAM KHẢO

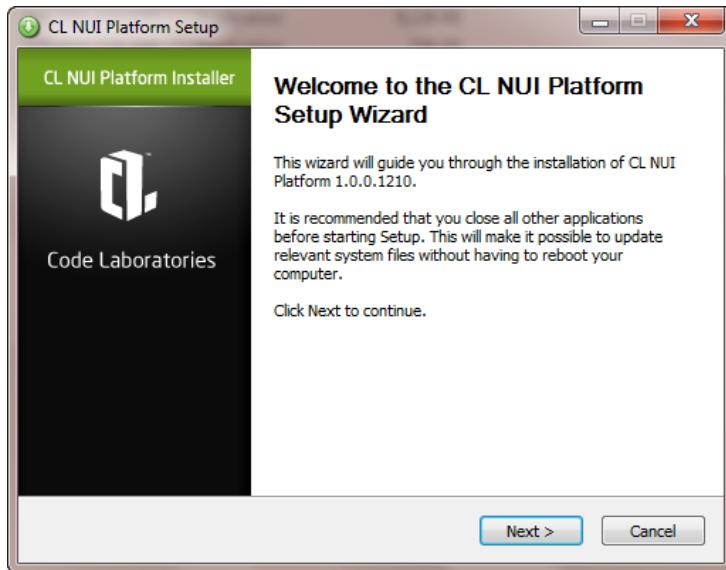
- [1] <http://en.wikipedia.org/wiki/Kinect>
- [2] http://www.ros.org/wiki/kinect_calibration/technical
- [3] **Mikkel Viager**, “Analysis of Kinect for Mobile Robots,” *Technical University of Denmark*, p. 11
- [4] <http://www.primesense.com/en/technology>
- [5] **Jacob Kjær**. *A Qualitative Analysis of Two Automated Registration Algorithms In a Real World Scenario Using Point Clouds from the Kinect*. June 27, 2011.
- [6] <http://nicolas.burrus.name/index.php/Research/KinectCalibration>
- [7] http://openkinect.org/wiki/Main_Page
- [8] <http://codelaboratories.com/nui>
- [9] <http://openni.org/Documentation>
- [10] <http://kinectforwindows.org>
- [11] http://www.brekel.com/?page_id=671
- [12] <http://pointclouds.org/documentation>
- [13] <http://www.dientuvietnam.net/forums>. Hiện tượng crosstalk.
- [14] http://www.roborealm.com/tutorial/Obstacle_Avoidance
- [15] **Radu Bogdan RUSU**. *PointCloud(2) processing in ROS*. May 2, 2010
- [16] <http://en.wikipedia.org/wiki/RANSAC>
- [17] http://tme.com.vn/Product.aspx?id=826&CategoryId=114#page=pro_info
- [18] <http://www.hocavr.com/index.php/app/dcservo>
- [19] **John A. Shaw**, The PID Control Algorithm , 2003
- [20] <http://www.uit.edu.vn/data/gtrinh/TH109/Htm/Chuong6.htm>
- [21] **Erick Ball, Greg Taschuk**. *Reverse Engineering the Kinect Stereo Algorithm*
- [22] **Michael YingYang, Wolfgang Forstner**. *Plane Detection in Point Cloud Data*.
January 25, 2010.

PHỤ LỤC 1: Kết hợp thư viện OpenNI và Code Laboratories Kinect (CL) để sử dụng chức năng điều khiển động cơ Kinect.

Như đã trình bày trong mục 3.2, thư viện OpenNI không hỗ trợ cho việc điều khiển động cơ của Kinect. Sau đây là thủ thuật bằng việc cài đặt thêm CL. Cả OpenNI và CL đều đóng vai trò như driver truy xuất phần cứng Kinect nên sẽ xung đột nếu cài đặt đồng thời hai thư viện này.

▪ *Bước 1:*

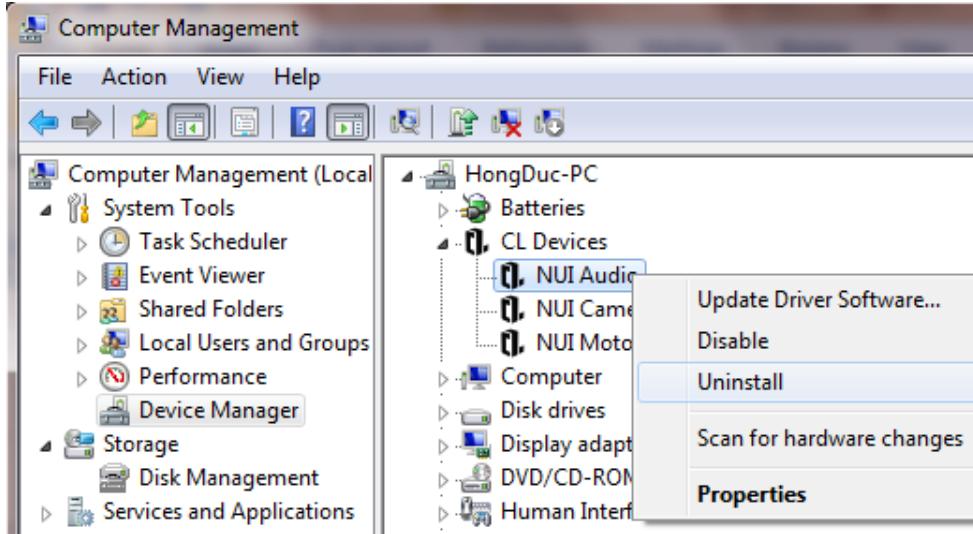
Cài đặt CL bình thường và đảm bảo chưa cài OpenNI và các chương trình đi kèm trước đó (nếu có thì phải gỡ bỏ).



CL có thể tải về tại: <http://codelaboratories.com/nui>

▪ *Bước 2:*

Mở cửa sổ Computer Management, và chọn Uninstall các mục dưới CL devices: NUI audio, NUI camera, NUI Motor.

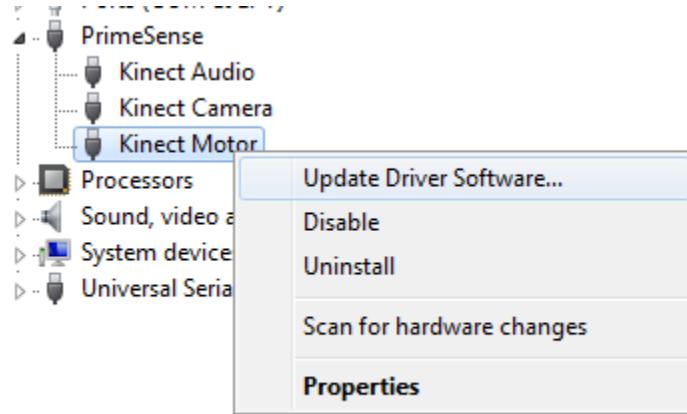


▪ **Bước 3:**

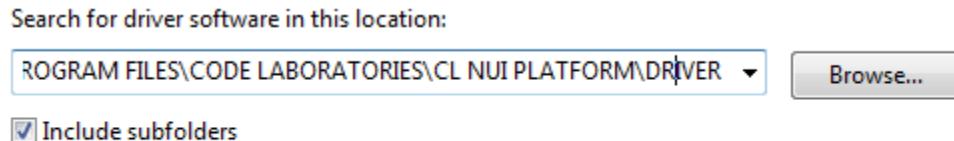
Cài đặt OpenNI và các chương trình kèm theo.

▪ **Bước 4:**

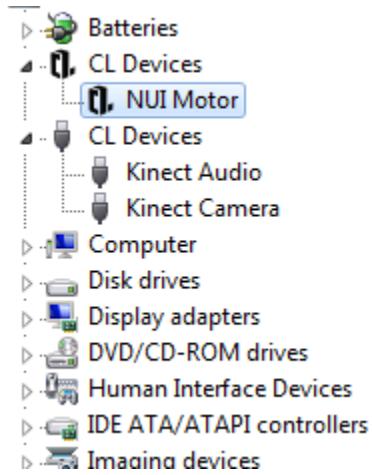
Vào lại cửa sổ như ở bước 1, lúc này Kinect được nhận bởi PrimeSense. Ta nhấp phải chuột trên Kinect Motor và chọn Update Driver Software



Và duyệt tới thư mục driver của CL:



Lúc này ta có thể sử dụng chức năng điều khiển động cơ Kinect trên thư viện OpenNI kết hợp với CL.



PHU LUC 2: Cách đấu dây dùng pin 12V thay adapter và tạo đế gắn lên robot cho Kinect

- **Cách đấu dây dùng pin 12V thay adapter cho Kinect**

Để tạo sự linh động cho robot, ta không dùng nguồn adapter mà thay bằng nguồn pin. Đầu tiên cắt dây adapter ra:



Sau đó dùng đầu cắm DC cho pin và đoạn dây trên:





- **Tạo đế cho Kinect để gắn lên robot**

Đây là bước quan trọng nhằm giữ sự an toàn cho Kinect và tránh bị rung khi di chuyển (sẽ ảnh hưởng tới kết quả xử lý ảnh trên máy tính).

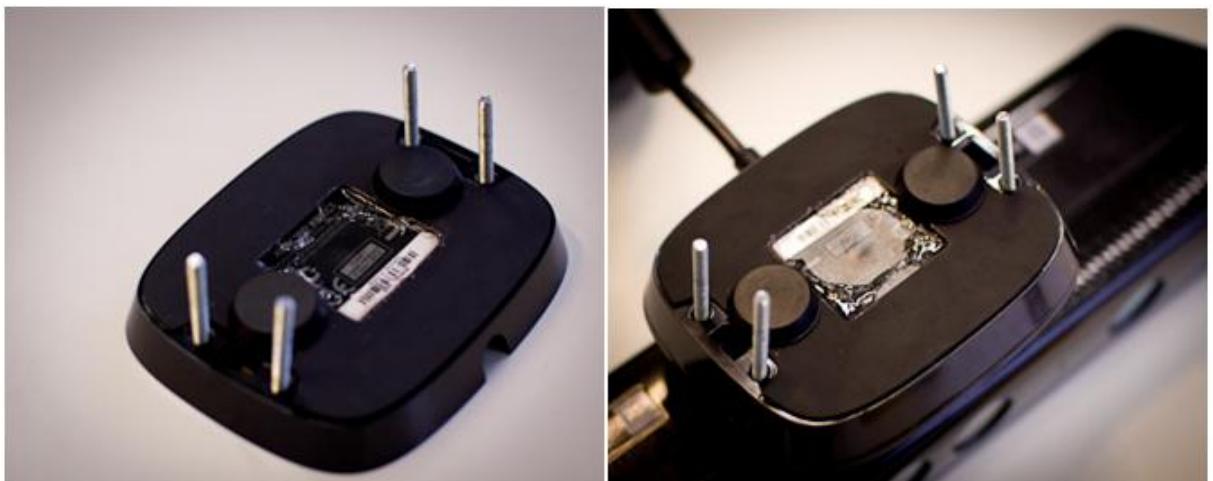
Đầu tiên tháo phần đế gắn trên Kinect:



Sau khi tháo:



Khoan bốn lỗ và gắn bốn ốc từ trong đế ra:



Và kết quả cuối cùng:



PHỤ LỤC 3: Kích thước robot

Gồm các kích thước cơ bản các chiều trên các hình chiếu đứng, bằng và cạnh. Đơn vị là milimet.

