



Structure Design and Platform Development of Universal Template  
for Humanoid Algorithm Interface (UTHAI)

การออกแบบโครงสร้างและพัฒนาระบบที่มีความหลากหลาย  
เพื่อการศึกษาและวิจัย

นายจิรภูร์ ศรีรัตนอากรณ์  
นายเจษฎากร ท่าไชยวงศ์  
นายวุฒิภัทร โชคอนันตทรัพย์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร  
ปริญญาวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมหุ่นยนต์และระบบอัตโนมัติ  
สถาบันวิทยาการหุ่นยนต์ภาครสนา  
มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี  
ปีการศึกษา 2560





Structure Design and Platform Development of Universal Template  
for Humanoid Algorithm Interface (UTHAI)

การออกแบบโครงสร้างและพัฒนาระบบที่นิร្ឣยานสำหรับหุ่นยนต์ชีวนิร्णย์  
เพื่อการศึกษาและวิจัย

นายจิรภูริ ศรีรัตนอากรณ์  
นายเจษฎากร ท่าไชยวงศ์  
นายวุฒิภัทร โชคอนันตทรัพย์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร  
ปริญญาวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมหุ่นยนต์และระบบอัตโนมัติ  
สถาบันวิทยาการหุ่นยนต์ภาครสนา  
มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี  
ปีการศึกษา 2560

การออกแบบโครงสร้างและพัฒนาระบบพื้นฐานสำหรับหุ่นยนต์ชีวภาพอยู่ด้วย  
เพื่อการศึกษาและวิจัย

นายจิรภัทร์ ศรีรัตนอาภรณ์

นายเจษฎากร ทาไชยวงศ์

นายวุฒิภัทร โชคอนันตทรัพย์

วิทยานิพนธ์เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร

ปริญญาวิศวกรรมศาสตรบัณฑิต สาขาวิชาชีวกรรมหุ่นยนต์และระบบอัตโนมัติ

สถาบันวิทยาการหุ่นยนต์ภาคสนาม

ปีการศึกษา 2560

---

อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

คณะกรรมการสอบ

.....

ประธานกรรมการ

(นายธนัชชา ชูพจน์เจริญ)

(ดร.อาบทิพย์ รีวงศ์กิจ)

.....

อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม

..... กรรมการ

(ดร.ปิติวุฒิ ธีรกิตติกุล)

.....

..... กรรมการ

(รศ.ดร.ชิต เหล่าวัฒนา)

(ดร.สุวัชัย วงศ์บุณย์ยงค์)

<b>ชื่อวิทยานิพนธ์</b>	การออกแบบโครงสร้างและพัฒนาระบบที่นักศึกษาสามารถเข้าใจได้ดีเพื่อการศึกษาและวิจัย
<b>หน่วยกิต</b>	6
<b>ผู้เขียน</b>	นายจิรภูริษฐ์ ศรีรัตนอกรรณ์ นายเจษฎากร หาญยวงศ์ นายวุฒิภัทร โขคอนันตทรัพย์
<b>อาจารย์ที่ปรึกษา</b>	ที่ปรึกษาวิทยานิพนธ์หลัก นายธนัชชา ชูพจน์เจริญ ที่ปรึกษาวิทยานิพนธ์ร่วม รศ.ดร.ชิต เหล่าวัฒนา
<b>หลักสูตร</b>	วิศวกรรมศาสตรบัณฑิต
<b>สาขาวิชา</b>	วิศวกรรมหุ่นยนต์และระบบอัตโนมัติ
<b>คณะ</b>	สถาบันวิทยาการหุ่นยนต์ภาคสนาม
<b>ปีการศึกษา</b>	2560

---

## บทคัดย่อ

งานวิทยานิพนธ์นี้เป็นงานที่เกี่ยวกับการออกแบบและจัดทำแพลตฟอร์มหุ่นยนต์ขึ้นอย่างดีด้วยเครื่องพิมพ์สามมิติ จุดประสงค์คือเพื่อ

## กิตติกรรมประกาศ

ขอขอบพระคุณอาจารย์ ดร.นัชชา ชูพจน์เจริญ อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก ที่ได้สละเวลามาให้คำปรึกษา ชี้แนะแนวทาง ให้ความรู้ในด้านต่างๆ ที่จำเป็นต่องานวิจัย รวมถึงการให้การสนับสนุนในเรื่องอุปกรณ์ในการทำวิจัย ตลอดจนช่วยตรวจสอบและแก้ไขวิทยานิพนธ์ให้เป็นไปอย่างสมบูรณ์

ขอขอบพระคุณรองศาสตราจารย์ ดร.ชิต เหล่าวัฒนา อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม ที่ได้ชี้แนะแนวทางให้คำแนะนำ และให้เกียรติเข้าร่วมการสอบวิทยานิพนธ์

ขอขอบพระคุณอาจารย์ ดร.ภิวดา มณีวรรณ และนายวิษณุ จุราวี ที่ได้ให้คำแนะนำในการแก้ไขปัญหาด้านต่างๆ ที่เกิดขึ้นระหว่างการทำวิจัย และได้ให้การสนับสนุนอุปกรณ์สำคัญที่ใช้ในการทำวิจัย

ขอขอบพระคุณอาจารย์ อับพิพัฒ์ ธิรวงศ์กิจ ที่กรุณาให้เกียรติเป็นประธานกรรมการสอบวิทยานิพนธ์ ให้คำแนะนำที่เป็นประโยชน์ต่อการจัดทำวิทยานิพนธ์ให้ดำเนินไปอย่างสมบูรณ์

ขอขอบพระคุณอาจารย์ ดร.ปิติวุฒย์ ธีรกิตติคุล ที่กรุณาให้เกียรติเป็นกรรมการสอบวิทยานิพนธ์ ให้คำแนะนำที่เป็นประโยชน์ต่อการวิจัย และการแก้ไขปรับปรุงงานวิจัย ตลอดจนตรวจแก้วิทยานิพนธ์ให้ดำเนินไปอย่างสมบูรณ์

ขอขอบพระคุณอาจารย์ ดร.สุภาชัย วงศ์บุณย์ยง ที่กรุณาให้เกียรติเป็นกรรมการสอบวิทยานิพนธ์ ให้คำแนะนำที่เป็นประโยชน์ต่อการวิจัย และการแก้ไขปรับปรุงงานวิจัย ตลอดจนตรวจแก้วิทยานิพนธ์ให้ดำเนินไปอย่างสมบูรณ์

ขอขอบพระคุณคณาจารย์ และบุคลากรในสถาบันวิทยาการหุ่นยนต์ภาคนามทุกท่าน ที่ได้ให้คำปรึกษาและช่วยเหลือด้านสถานที่พร้อมทั้งส่งเสริมความต้องการต่างๆ ในระหว่างการทำวิทยานิพนธ์

ขอขอบคุณนักศึกษาปริญญาตรี สถาบันวิทยาการหุ่นยนต์ภาคนามทุกท่าน ที่ได้ให้คำแนะนำ ถ้ามี แล้วเป็นกำลังใจมาโดยตลอด

และสุดท้ายนี้ ขอน้อมรำลึกถึงพระคุณบิดา มารดา และครอบครัว ที่ส่งเสริมให้กำลังใจ และให้การสนับสนุนในเรื่องต่างๆ จนกระทั้งข้าพเจ้าประสบความสำเร็จในการศึกษา

นายจิรภูริศ ศรีรัตนอาภรณ์  
นายเจษฎากร หาไซวงศ์  
นายวุฒิภัทร โชคอนันตทรัพย์

## สารบัญ

เรื่อง	หน้า
<b>บทคัดย่อ .....</b>	<b>ค</b>
<b>กิตติกรรมประกาศ .....</b>	<b>๔</b>
<b>สารบัญ .....</b>	<b>๕</b>
<b>รายการรูปภาพ .....</b>	<b>๗</b>
<b>รายการตาราง.....</b>	<b>๘</b>
<b>รายการสัญลักษณ์.....</b>	<b>๙</b>
<b>ประมวลศัพท์และตัวย่อ.....</b>	<b>๑๐</b>
<b>บทที่ 1 บทนำ.....</b>	<b>๑</b>
1.1 ที่มาและความสำคัญ.....	1
1.2 วัตถุประสงค์.....	2
1.3 ประโยชน์ที่คาดว่าจะได้รับ .....	2
1.4 ขอบเขตการดำเนินงาน.....	2
1.5 ภาพรวมของระบบและขั้นตอนการดำเนินงาน.....	2
1.5.1 ศึกษาค้นคว้าเอกสารและงานวิจัยที่เกี่ยวข้อง.....	3
1.5.2 ส่วนโครงสร้างของหุ่นยนต์อิมานอยด์.....	3
1.5.3 ส่วนโปรแกรมของหุ่นยนต์อิมานอยด์ .....	3
1.5.4 ส่วนการออกแบบระบบพื้นฐานเพื่อการพัฒนาต่อไป.....	4
<b>บทที่ 2 ทฤษฎีและหลักการ.....</b>	<b>5</b>
2.1 ทฤษฎีและงานวิจัยที่เกี่ยวข้อง .....	5
2.1.1 หุ่นยนต์อิมานอยด์ .....	5
2.1.2 ทฤษฎีที่เกี่ยวข้องกับมนุษย์ .....	8
2.1.2.1 การวิเคราะห์การเดินของมนุษย์.....	8
2.1.2.2 การวิเคราะห์องศาอิสระของมนุษย์.....	9
2.1.3 ทฤษฎีที่เกี่ยวข้องกับหุ่นยนต์อิมานอยด์ .....	10
2.1.3.1 ส่วนประกอบของหุ่นยนต์อิมานอยด์ .....	10
2.1.3.2 วัสดุการผลิตหุ่นยนต์อิมานอยด์ .....	10
2.1.3.3 การสร้างและการควบคุมการเดินแบบสมดุลสถิต .....	11

## สารบัญ (ต่อ)

เรื่อง	หน้า
2.1.3.4 การสร้างและการควบคุมการเดินแบบสมดุลพลวัต .....	11
2.1.3.5 จุดศูนย์กลางมวลของหุ่นยนต์ .....	12
2.1.4 ตัวอย่างหุ่นยนต์อิมามานอยด์ .....	13
2.2 การออกแบบโครงสร้างของหุ่นยนต์ .....	18
2.2.1 ความแตกต่างระหว่างโครงสร้างของมนุษย์กับโครงสร้างของหุ่นยนต์.....	18
2.2.1.1 ความแตกต่างขององศาเสรี .....	18
2.2.1.2 ความแตกต่างของอัตราส่วน .....	18
2.2.1.3 กำลังและประสิทธิภาพของมอเตอร์.....	19
2.2.2 วัสดุและการขึ้นรูปโครงสร้างของหุ่นยนต์อิมามานอยด์ .....	19
2.2.3 อุปกรณ์ที่ใช้ในหุ่นยนต์อิมามานอยด์ .....	19
2.2.3.1 ตัวขับเคลื่อน .....	19
2.2.3.2 หน่วยประมวลผลควบคุม .....	20
2.2.3.3 เชนเชอร์ตตรวจหน้าสัมผัสที่พื้น .....	22
2.2.3.4 เชนเชอร์วัดความเฉื่อย .....	23
2.2.4 แนวคิดการออกแบบกลไกการเดินของหุ่นยนต์อิมามานอยด์ .....	23
2.3 การออกแบบโปรแกรมด้วย ROS .....	24
2.3.1 ระบบที่ใช้ช่วยในการพัฒนาหุ่นยนต์ .....	24
2.3.2 ระบบที่ใช้ในการจำลองการทำงานของหุ่นยนต์ .....	27
2.3.3 Robot Operating System .....	29
2.4 การออกแบบระบบพื้นฐาน .....	35
2.4.1 ความแตกต่างของ Operating Systems .....	35
2.4.2 ข้อแตกต่างระหว่าง Open platform กับ Non-open platform .....	35
2.4.3 มาตรฐานหน่วยวัดและการบวกพิกัด .....	35
2.4.4 Robot Operating System .....	36
<b>บทที่ 3 การดำเนินงานวิจัย .....</b>	<b>37</b>
3.1 แผนการดำเนินงาน .....	37
3.2 การออกแบบโครงสร้างของหุ่นยนต์ .....	38
3.2.1 โครงสร้างหุ่นยนต์ .....	38

## สารบัญ (ต่อ)

เรื่อง	หน้า
3.2.2 การเชื่อมต่อหุ่นยนต์ชีวามโนยด์.....	39
3.2.3 อุปกรณ์ที่ใช้ในหุ่นยนต์ชีวามโนยด์อุทัย .....	40
3.3 การออกแบบโปรแกรมด้วย ROS .....	42
3.3.1 Modelling .....	42
3.3.1.1 ROS packages for robot modelling.....	42
3.3.1.2 URDF .....	42
3.3.2 กำหนดพิกัดเฟรมให้กับหุ่นยนต์ชีวามโนยด์.....	45
3.3.3 Box model.....	46
3.3.4 Dynamic properties .....	47
3.3.5 โครงสร้างการติดต่อสื่อสารระหว่าง Node ใน ROS .....	48
3.4 การออกแบบระบบพื้นฐาน .....	53
3.4.1 ออกแบบสถาปัตยกรรมของหุ่นยนต์.....	53
3.4.1.1 หน่วยประมวลผลควบคุมระดับสูง (High level controller) .....	54
3.4.1.2 หน่วยประมวลผลควบคุมระดับต่ำ (Low level controller) .....	54
3.4.2 จัดทำคู่มือและเอกสารการใช้งาน .....	55
3.4.2.1 รายการวัสดุที่ใช้ในการทำหุ่นยนต์ชีวามโนยด์ UTHAI .....	55
<b>บทที่ 4 ผลการวิจัย .....</b>	<b>58</b>
4.1 การออกแบบโครงสร้างของหุ่นยนต์ .....	58
4.2 การออกแบบโปรแกรมด้วย ROS .....	58
4.2.1 Simulation Gazebo .....	58
4.3 การออกแบบระบบพื้นฐาน .....	58
<b>บทที่ 5 สรุปผลการทดลองและข้อเสนอแนะ .....</b>	<b>59</b>
5.1 การออกแบบโครงสร้างของหุ่นยนต์ .....	59
5.2 การออกแบบโปรแกรมด้วย ROS .....	59
5.3 การออกแบบระบบพื้นฐาน .....	59
5.4 สรุปภาพรวม .....	59
<b>เอกสารอ้างอิง.....</b>	<b>59</b>

## สารบัญ (ต่อ)

เรื่อง	หน้า
ภาคผนวก ก ผลงานตีพิมพ์เผยแพร่จากวิทยานิพนธ์.....	60
ก.1 บทความวิจัยเสนอในที่ประชุมวิชาการและมีการพิมพ์รวมเล่ม.....	60
ก.2 บทความวิชาการ .....	60
ภาคผนวก ข แหล่งข้อมูล Latex.....	61
ข.1 แหล่งข้อมูลออนไลน์.....	61

## รายการรูปภาพ

รูป	หน้า
รูปที่ 2.1 แสดงความแตกต่างของหุ่นยนต์อิมามาโนยด์แต่ละประเภท.....	5
รูปที่ 2.2 องค์ประกอบหลักของหุ่นยนต์อิมามาโนยด์.....	6
รูปที่ 2.3 วัสดุการเดินของมนุษย์.....	8
รูปที่ 2.4 ส่วนประกอบของหุ่นยนต์อิมามาโนยด์ .....	10
รูปที่ 2.5 วัสดุการเดินของหุ่นยนต์อิมามาโนยด์.....	10
รูปที่ 2.6 การควบคุมตำแหน่งของจุดรวมมวลให้อยู่ในพื้นที่ฐาน .....	11
รูปที่ 2.7 การควบคุมตำแหน่งของจุดโมเมนต์ศูนย์ให้ตรงกับแรงปฏิกิริยารวม.....	12
รูปที่ 2.8 หุ่นยนต์อิมามาโนยด์ปีอปปี .....	13
รูปที่ 2.9 หุ่นยนต์อิมามาโนยด์ไอคัพ .....	14
รูปที่ 2.10 หุ่นยนต์อิมามาโนยด์ดาร์วิน .....	15
รูปที่ 2.11 หุ่นยนต์อิมามาโนยด์น่าโฉะ .....	16
รูปที่ 2.12 หุ่นยนต์อิมามาโนยด์ราบอท .....	17
รูปที่ 2.13 ตัวอย่างตำแหน่งและการหมุนของข้อต่อของหุ่นยนต์เพื่อการอ้างอิง .....	18
รูปที่ 2.14 รอแก้ไข .....	19
รูปที่ 2.15 รอแก้ไข .....	19
รูปที่ 2.16 รอแก้ไข .....	20
รูปที่ 2.17 รอแก้ไข .....	21
รูปที่ 2.18 ลักษณะโครงสร้างของตัวตรวจจับแรงกด FSR .....	22
รูปที่ 2.19 การทำงานของตัวตรวจจับแรงกด FSR .....	22
รูปที่ 2.20 เชนเชอร์วัตความเฉียบ .....	23
รูปที่ 2.21 player project middleware .....	24
รูปที่ 2.22 yarp middleware.....	24
รูปที่ 2.23 urbi middleware .....	25
รูปที่ 2.24 miro middleware .....	25
รูปที่ 2.25 openrdk middleware.....	25
รูปที่ 2.26 ROS middleware Rviz .....	26
รูปที่ 2.27 ROS algitecture .....	26

## รายการรูปภาพ (ต่อ)

รูป	หน้า
รูปที่ 2.28 ROS Moveit .....	26
รูปที่ 2.29 ผลลัพธ์จากการใช้โปรแกรม USARSim .....	27
รูปที่ 2.30 ผลลัพธ์จากการใช้โปรแกรม MuRoSimF .....	27
รูปที่ 2.31 Mobile robot with gazebo .....	28
รูปที่ 2.32 Quadrotor with gazebo .....	28
รูปที่ 2.33 ตัวอย่างสถาปัตยกรรมของ ROS .....	29
รูปที่ 2.34 ตัวอย่างไฟล์ package.xml .....	31
รูปที่ 2.35 ตัวอย่างการแสดงผลใน rqt .....	33
รูปที่ 2.36 ตัวอย่างการแสดงผลใน RViz .....	34
รูปที่ 2.37 ตัวอย่างหุ่นยนต์ชีวมานอยด์ Poppy .....	34
รูปที่ 2.38 การตั้งแgnตามกฎหมายข่าว .....	35
 รูปที่ 3.1 ภาพแสดงแสดงโครงสร้างของหุ่นยนต์ UTHAI .....	 38
รูปที่ 3.2 ภาพแสดงการติดตั้งเซนเซอร์ในจุดต่างๆ .....	39
รูปที่ 3.3 แสดงประสิทธิภาพของมอเตอร์ EX-106+ .....	40
รูปที่ 3.4 แสดงเซนเซอร์ IMU MPU9250 .....	41
รูปที่ 3.5 ตัวอย่าง link ใน urdf .....	43
รูปที่ 3.6 การอธิบาย link ใน URDF ไฟล์ .....	43
รูปที่ 3.7 ตัวอย่าง joint ใน urdf .....	44
รูปที่ 3.8 การอธิบาย Joint ใน URDF ไฟล์ .....	44
รูปที่ 3.9 ภาพแสดงช่วงล่างหุ่นยนต์ชีวมานอยด์ .....	47
รูปที่ 3.10 การติดต่อสื่อสารระหว่าง Node .....	48
รูปที่ 3.11 สถาปัตยกรรมของหุ่นยนต์ชีวมานอยด์ UTHAI .....	53
รูปที่ 3.12 บอร์ดคอนโทรลเลอร์ Odroid XU4 .....	54
รูปที่ 3.13 บอร์ดคอนโทรลเลอร์ Nucleo F411RE .....	54
รูปที่ 3.14 ภาพตัวอย่างการวาดออฟเจ็คต่างๆ .....	56
รูปที่ 3.15 ภาพตัวอย่างการวาดเฟรมของแขนกล .....	56

## รายการตาราง (ต่อ)

ตาราง	หน้า
รูปที่ 3.16 ภาพตัวอย่างการวัดเฟรมของหุ่นยนต์ชีวมานอยด์.....	57

## รายการตาราง

ตาราง	หน้า
ตารางที่ 2.1 ความสามารถในการหมุนของแต่ละข้อต่อของมนุษย์.....	9
ตารางที่ 2.2 ตัวอย่างชื่อและข้อมูลของ Message .....	30
ตารางที่ 2.3 ตารางแสดงหน่วยวัดมาตรฐาน.....	36
ตารางที่ 3.1 ตารางแสดงสมบัติทางกลของวัสดุต่าง ๆ .....	39
ตารางที่ 3.2 Message Geometry Point .....	47
ตารางที่ 3.3 Message Geometry Point .....	48
ตารางที่ 3.4 Message Geometry Quaternion .....	49
ตารางที่ 3.5 Message Geometry Pose.....	49
ตารางที่ 3.6 Message Geometry Vector3.....	49
ตารางที่ 3.7 Message Geometry Twist .....	49
ตารางที่ 3.8 Message Navigation Odometry .....	49
ตารางที่ 3.9 Message Geometry Pose2D.....	50
ตารางที่ 3.10 Message Navigation Path.....	50
ตารางที่ 3.11 Message Geometry PoseStamped.....	50
ตารางที่ 3.12 Message Trajectory JointTrajectory.....	51
ตารางที่ 3.13 Message Trajectory JointTrajectoryPoint.....	51
ตารางที่ 3.14 Message Sensor JointState .....	51
ตารางที่ 3.15 Message Geometry Wrench.....	51
ตารางที่ 3.16 Message Sensor Imu .....	52
ตารางที่ 3.17 Message Sensor MegneticField .....	52
ตารางที่ 3.18 ตารางแสดงรายการของวัสดุต่าง ๆ .....	55

## รายการสัญลักษณ์

$\theta$	เชิงตัว
$d$	distance
kg	Kilogram
$m^2$	Square Metre

## ประมวลศัพท์และตัวย่อ

UTHAI	Universal Template for Humanoid Algorithm Interface
ROS	Robot Operating System
IMU	Inertial Measurement Unit
Dof	Degree of Freedom
CoM	Center of Mass
ZMP	Zero Moment Point
PLA	Polylactic acid
ABS	Acrylonitrile butadiene styrene
KMUTT	King Mongkut's University of Technology Thonburi
Liws	ลูกิวส์ โซลูชันส์ ทรัพย์
$\theta$	เชิงตัว

## บทที่ 1

### บทนำ

#### 1.1 ที่มาและความสำคัญ

หุ่นยนต์อิวามานอยด์เป็นหุ่นยนต์ที่สร้างขึ้นเพื่อเลียนแบบสรีระร่างกายของมนุษย์ ซึ่งมีข้อจำกัดนวนมากเพื่อให้มีการเคลื่อนไหวคล้ายมนุษย์ ลักษณะเด่นของหุ่นยนต์อิวามานอยด์คือ การเคลื่อนที่ด้วยขาสองข้างด้วยการเคลื่อนที่โดยการใช้ขาัน ทำให้หุ่นยนต์อิวามานอยด์สามารถเคลื่อนที่ได้ อย่างคล่องแคล่วในทุกสภาพพื้นผิว ทั้งทางเรียบ ทางชุขอหรือพื้นต่างระดับ<sup>1</sup> ซึ่งนั่นทำให้หุ่นยนต์ที่เดินสองขาแตกต่างจากหุ่นยนต์ที่เคลื่อนที่ด้วยล้อ ด้วยโครงสร้างของหุ่นยนต์ที่คล้ายมนุษย์นั้นเอง จึงทำให้หุ่นยนต์อิวามานอยด์สามารถทำงานได้หลากหลายและยืดหยุ่น สามารถที่จะใช้อุปกรณ์ที่ไม่ถูกออกแบบขึ้นมาเพื่อใช้กับมนุษย์ได้ ซึ่งหมายความว่าในอนาคตนั้นหุ่นยนต์อิวามานอยด์สามารถที่จะทำงานทดแทนแรงงานของมนุษย์ได้<sup>2</sup> งานที่หุ่นยนต์อิวามานอยด์จะเข้ามาทดแทนแรงงานของมนุษย์นั้น จะเป็นงานที่ต้องทำซ้ำๆ จนเกินความเมื่อยล้า งานที่อยู่ในพื้นที่อันตรายหรือที่เสี่ยงต่อการเกิดอุบัติเหตุ

สถาบันวิจัยหลายแห่งทั่วโลกกำลังให้ความสนใจสนับสนุนด้านการศึกษาวิจัยและพัฒนาหุ่นยนต์อิวามานอยด์ เพื่อให้ทำการกิจต่างๆ ยกตัวอย่างเช่น DARPA Robotics Challenge (DRC)<sup>3</sup> เป็นรายการแข่งขันหุ่นยนต์กึ่งอัตโนมัติเพื่อทำการกิจกู้ภัยในสถานการณ์ภัยพิบัติที่อันตราย ซึ่งสถาบันวิจัยหุ่นยนต์ทั่วโลกได้ส่งหุ่นยนต์อิวามานอยด์ของตนเข้าร่วมการแข่งขัน ในปัจจุบันได้มีการพัฒนาหุ่นยนต์อิวามานอยด์ขึ้นมาหลากหลายตัว เช่น ASIMO, HRP-3, LOLA และ WATHLETE-1 การพัฒนาหุ่นยนต์อิวามานอยด์นั้นได้ก่อให้เกิดงานศึกษาวิจัย และทฤษฎี ต่อยอด ต่างๆ มากมาย เช่น การวางแผนการเดิน การเดินแบบสติต การเดินแบบพลวัต การติดต่อสื่อสารของระบบ การมองเห็นและการประมวลผลภาพ การพูดคุยโต้ตอบกับมนุษย์ ปัญญาประดิษฐ์ ฯลฯ ซึ่งงานวิจัยเหล่านี้ สามารถที่จะนำไปประยุกต์ใช้กับระบบหุ่นยนต์ระบบอื่นๆ ได้ แม้ว่าจะมีการพัฒนาหุ่นยนต์อิวามานอยด์มากมาย แล้ว แต่การเริ่มต้นทำงานวิจัยที่มีความเกี่ยวข้องกับหุ่นยนต์อิวามานอยด์นั้น ต้องใช้ความรู้ความสามารถ เครื่องมือ ระยะเวลา งบประมาณ และ ความพยายามเป็นอย่างมาก การสร้างหุ่นยนต์อิวามานอยด์ขึ้นมาเป็นระบบพื้นฐาน ให้มีความพร้อมสำหรับการพัฒนาต่อไป แก่นักศึกษาหรือนักวิจัย จะช่วยประหยัดเวลาและงบประมาณที่ต้องใช้ได้อย่างมาก ซึ่งนั่นหมายความว่า นักวิจัยจะสามารถทำงานวิจัยได้อย่างมีประสิทธิภาพมากขึ้น

ในงานวิจัยนี้ เป็นการออกแบบหุ่นยนต์อิวามานอยด์และพัฒนาระบบพื้นฐานของหุ่นยนต์อิวามานอยด์ สำหรับให้นักศึกษาหรือนักวิจัยสามารถพัฒนาต่อไปได้ โดยหุ่นยนต์อิวามานอยด์ที่ออกแบบมานั้น สามารถที่จะปรับปรุงแก้ไข ดัดแปลงได้ง่าย ตัวโครงสร้างจะใช้เป็น พลาสติก PLA ที่สามารถพิมพ์ได้ โดยการใช้เครื่องพิมพ์สามมิติ มีเซนเซอร์ตรวจการสัมผัสพื้นที่ฝ่าเท้าของหุ่นยนต์ มีเซนเซอร์สำหรับการวัดมุมอีking ที่ลำตัวของหุ่นยนต์ และเพื่อที่จะทำให้ง่ายต่อการศึกษาทำความเข้าใจ บำรุงรักษา จึงได้มีการจัดทำคู่มือและเอกสารวิธีการใช้งานอย่างชัดเจน โดยจะเก็บในรูปแบบของเอกสารออนไลน์

<sup>1</sup> การเคลื่อนที่ของหุ่นยนต์รูปแบบต่างๆ ราชวิถีส่วนดุสิต

<sup>2</sup> ณัฐพงษ์ วรีประเสริฐ และภรณ์ ล้ำดี (2552: 374)

<sup>3</sup> DARPA 2015 [<https://www.darpa.mil/about-us/about-darpa>]

## 1.2 วัตถุประสงค์

ออกแบบโครงสร้างของหุ่นยนต์ชีวามโนยด์ที่สามารถแก้ไขปรับเปลี่ยนได้ง่าย พัฒนาระบบที่มีพื้นฐาน ระบบจำลองสำหรับหุ่นยนต์ชีวามโนยด์เพื่อใช้ในการศึกษาวิจัย รวมรวมเครื่องมือที่เป็นประโยชน์ต่อการพัฒนาหุ่นยนต์ และจัดทำเอกสารออนไลน์ให้บุคคลที่สนใจสามารถเข้ามาศึกษาได้

## 1.3 ประโยชน์ที่คาดว่าจะได้รับ

- 1.3.1 มีต้นแบบหุ่นยนต์ชีวามโนยด์สำหรับใช้ในงานวิจัยแขนงต่างๆ
- 1.3.2 มีระบบพื้นฐานสำหรับพัฒนาหุ่นยนต์ชีวามโนยด์รุ่นใหม่ในสถาบัน
- 1.3.3 มีระบบจำลองสำหรับจำลองการทำงานของหุ่นยนต์ชีวามโนยด์
- 1.3.4 มีแหล่งรวมรวมเครื่องมือสำหรับการพัฒนาหุ่นยนต์
- 1.3.5 มีคู่มือ เอกสาร วิธีการใช้งาน และรายละเอียดของหุ่นยนต์สำหรับพัฒนาต่อยอด

## 1.4 ขอบเขตการดำเนินงาน

- 1.4.1 ใช้ ROS เป็นกรอบการทำงานสำหรับพัฒนาระบบที่มีพื้นฐาน
- 1.4.2 ออกแบบโครงสร้างให้มีความแข็งแรง สามารถรับน้ำหนักอุปกรณ์ที่ติดตั้งอยู่บนตัวหุ่นยนต์ได้
- 1.4.3 น้ำหนักของหุ่นยนต์รวมกันทั้งตัว ไม่เกิน 5 กิโลกรัม
- 1.4.4 ใช้ Solidworks 3D เป็นโปรแกรมสำหรับออกแบบโครงสร้าง และคำนวณ
- 1.4.5 หุ่นยนต์มีความสูงไม่ต่ำกว่า 100 เซนติเมตร และสูงไม่เกิน 120 เซนติเมตร
- 1.4.6 หุ่นยนต์มี 2 แขน 2 ขา มีองศาอิสระของขาข้างละ 6 และแขนข้างละ 2 องศาอิสระ
- 1.4.7 หุ่นยนต์สามารถทำงานได้ภายในสภาพแวดล้อมแบบปิด
- 1.4.8 หุ่นยนต์ใช้พลังงานจากแหล่งจ่ายพลังงานที่มีขนาดแรงดันไฟฟ้า 12 โวลต์
- 1.4.9 หุ่นยนต์ใช้ตัวขับเคลื่อนแบบดิจิตอลสำหรับแต่ละข้อต่อเป็น Dynamixel Digital Servo
- 1.4.10 ใช้ Gazebo สำหรับจำลองระบบของหุ่นยนต์
- 1.4.11 ติดตั้งเซนเซอร์วัดการกด (Ground contact) ที่ฝ่าเท้าของหุ่นยนต์
- 1.4.12 ติดตั้งเซนเซอร์วัดมุมเอียง (IMU) ที่บริเวณลำตัวของหุ่นยนต์
- 1.4.13 จัดทำคู่มือ เอกสารการใช้งาน และรายละเอียดส่วนประกอบของหุ่นยนต์

## 1.5 ภาพรวมของระบบและขั้นตอนการดำเนินงาน

งานวิจัยนี้การดำเนินงานวิจัยถูกแบ่งออกเป็นสามส่วน คือ ส่วนที่หนึ่งส่วนโครงสร้างของหุ่นยนต์ชีวามโนยด์ เป็นส่วนที่ทำในส่วนของการขึ้นรูปชิ้นงาน ออกแบบโมเดลสามมิติ รวมไปถึงระบบอิเล็กทรอนิกส์ ติดตั้งบอร์ดและเซนเซอร์ไว้ตามจุดต่างๆ เพื่อสร้างโครงสร้างของหุ่นยนต์ให้สามารถรองรับการเดินได้ ส่วนที่สองส่วน โปรแกรมของหุ่นยนต์ชีวามโนยด์ เป็นส่วนที่ทำในส่วนของการสั่งการตัวขับเคลื่อนต่างๆ อ่านค่าสถานะเซนเซอร์ จากคอนโซลเลอร์ รวมไปถึงระบบจำลองการทำงานของหุ่นยนต์ และส่วนที่สามส่วนการออกแบบระบบพื้นฐาน เพื่อการพัฒนาต่อยอด ส่วนนี้จะเป็นส่วนที่ทำให้ผู้ที่จะมาวิจัยต่อยอดสามารถทำงานได้ง่ายขึ้น จัดการเอกสารคู่มือ การใช้งานต่างๆให้เป็นระบบเบเย็บ สามารถแยกขั้นตอนการทำงานของแต่ละส่วนออกเป็นข้อดังนี้

### 1.5.1 ศึกษาค้นคว้าเอกสารและงานวิจัยที่เกี่ยวข้อง

- ศึกษาเกี่ยวกับส่วนประกอบของหุ่นยนต์ชีวามโนยด์
- ศึกษาทฤษฎีที่เกี่ยวข้องกับของมนุษย์
- ศึกษาทฤษฎีที่เกี่ยวข้องกับหุ่นยนต์ชีวามโนยด์
- ศึกษาความแตกต่างระหว่างมนุษย์กับหุ่นยนต์ชีวามโนยด์
- ศึกษาวิธีการและวัสดุที่ใช้ในการสร้างหุ่นยนต์
- ศึกษาระบบที่ใช้ช่วยในการพัฒนาหุ่นยนต์
- ศึกษาระบบที่ใช้ในการจำลองการทำงานของหุ่นยนต์
- ศึกษาการใช้งาน ROS พื้นฐาน

### 1.5.2 ส่วนโครงสร้างของหุ่นยนต์ชีวามโนยด์

#### 1 ออกแบบโครงสร้างของหุ่นยนต์ชีวามโนยด์

- ออกแบบโครงสร้างของหุ่นยนต์
- จำลองและทดสอบความแข็งแรงด้วยเทคนิค FEA
- วางแผนไฟฟ้า
- การเลือกใช้แหล่งพลังงาน
- การเลือกใช้คอมพิวเตอร์
- การไดร์ว์มอเตอร์
- การเขียนต่อเซนเซอร์ชนิดต่างๆ

#### 2 จัดสร้างโครงสร้างหุ่นยนต์ชีวามโนยด์

- จัดทำส่วนที่ต้องออกแบบเอง
- ประกอบโครงสร้างทางกล
- ประกอบโครงสร้างทางไฟฟ้า

#### 3 ทดสอบการทำงานของหุ่นยนต์ชีวามโนยด์

- ทดสอบการเคลื่อนไหวของหุ่นยนต์
- ตรวจสอบองศาลิมิตของข้อต่อ
- ตรวจสอบทิศทางการหมุน

### 1.5.3 ส่วนโปรแกรมของหุ่นยนต์ชีวามโนยด์

#### 1 ออกแบบโปรแกรมของหุ่นยนต์ชีวามโนยด์

- ประยุกต์ระบบที่ใช้ในการพัฒนาหุ่นยนต์
- ประยุกต์ระบบจำลองการทำงานของหุ่นยนต์

#### 2 สร้างโปรแกรมของหุ่นยนต์

- สร้างโมเดลไฟล์ URDF

#### 3 ทดสอบการทำงานของโปรแกรม

- ทดสอบการใช้งาน Simulation
- ทดสอบการอ่านค่าจากเซนเซอร์ IMU
- ทดสอบการอ่านค่าจากเซนเซอร์ Ground contact
- ทดสอบการสั่งการมอเตอร์

## 1.5.4 ส่วนการออกแบบระบบพื้นฐานเพื่อการพัฒนาต่อ�อด

### 1 ติดตั้งระบบ

- Setup Version Control [GitHub]
- ติดตั้ง ROS ในคอมพิวเตอร์ที่อยู่ในหุ่นยนต์
- ติดตั้ง ROS ในคอมพิวเตอร์ที่ควบคุมหุ่นยนต์
- ตั้งค่า ROS Workspace
- ตั้งค่าการติดต่อสื่อสารระหว่างระบบ

### 2 วางแผนระบบพื้นฐาน

### 3 จัดทำคู่มือ

- รายการของที่จำเป็นต้องใช้
- คู่มือการประกอบของหุ่นยนต์
- คู่มือรายละเอียดข้อมูลของหุ่นยนต์
- คู่มือลิมิตของแต่ละข้อต่อของหุ่นยนต์
- คู่มือการวางแผนสร้างของระบบ

### 4 สื่อการสอน

- การติดตั้ง ROS
- การใช้งาน ROS เป็องตัน
- การใช้งาน Simulation ของหุ่นยนต์
- การใช้งาน ROS กับหุ่นยนต์ชีวามโนยด์
- การใช้งาน ROS บน Mbed

## บทที่ 2

### ทฤษฎีและหลักการ

#### 2.1 ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

##### 2.1.1 หุ่นยนต์อิริวามาโนยด์

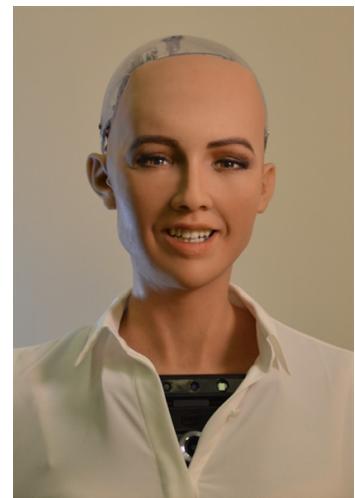
หุ่นยนต์อิริวามาโนยด์ คือ หุ่นยนต์ที่ถูกสร้างขึ้นมาให้มีรูปร่างคล้ายคลึงกับสรีระโครงสร้างของมนุษย์ มักถูกออกแบบขึ้นมาเพื่อจุดประสงค์เฉพาะอย่าง เช่น เพื่อให้ใช้เครื่องมือต่างๆของมนุษย์ เพื่อให้อยู่ในสภาพแวดล้อมของมนุษย์ เพื่อศึกษาการเคลื่อนไหวของร่ายกายมนุษย์ เพื่อศึกษาระบบการทำงานของมนุษย์ เพื่อทำงานในสิ่งที่มนุษย์ทำได้ยาก หรือเพื่อวัตถุประสงค์อื่นๆ โดยทั่วไปแล้ว หุ่นยนต์อิริวามาโนยด์จะประกอบไปด้วย 4 ส่วนคือ ส่วนของหัว ส่วนของลำตัว ส่วนของแขน และส่วนของขา แต่การสร้างหุ่นยนต์อิริวามาโนยด์นั้นก็ไม่จำเป็นที่จะต้องมีส่วนประกอบทุกส่วนดังที่กล่าวไป ในบางครั้งอาจมีเพียงแค่ส่วนบนเท่านั้น ดังรูปที่ 2.1 ก หุ่นยนต์นามจากสถาบันวิทยาการหุ่นยนต์ภาคสนาม เป็นหุ่นยนต์ที่มีส่วนบนเหมือนมนุษย์ แต่มีส่วนล่างเป็นล้อ หรือหุ่นยนต์อิริวามาโนยด์ที่มีเพียงแค่ส่วนล่าง ดังรูปที่ 2.1 ข หุ่นยนต์สัมจุก เป็นหุ่นยนต์อิริวามาโนยด์ที่มีเพียงแค่ส่วนขาเท่านั้น หรือหุ่นยนต์อิริวามาโนยด์ที่มีเพียงใบหน้าเหมือนมนุษย์ ดังรูปที่ 2.1 ค หุ่นยนต์โซเฟีย เป็นแอนดรอยด์ที่มีหน้าตาคล้ายมนุษย์มาก มีปาก สามารถพูดปฏิสัมพันธ์กับมนุษย์ได้



(ก) หุ่นยนต์ประชาสัมพันธ์โรงแรม



(ข) หุ่นยนต์เดินสองขาสัมจุก



(ค) หุ่นยนต์แอนดรอยด์โซเฟีย

รูปที่ 2.1: แสดงความแตกต่างของหุ่นยนต์อิริวามาโนยด์แต่ละประเภท

งานวิจัยทางด้านหุ่นยนต์อิริวามาโนยด์จากอดีตจนถึงปัจจุบันส่วนใหญ่จะเป็นการพัฒนาความสามารถของ การเดินของหุ่นยนต์ เช่น เริ่มต้นจากแรกสุดจะเป็นการพัฒนาให้หุ่นยนต์สามารถเดินหน้าได้ ต่อมาเกิดความ สามารถให้หุ่นยนต์สามารถเดินบนพื้นอิ่ม พื้นชุ่มชะ เดินเลี้ยวซ้ายขวา เดินขึ้ลงบันได ฯลฯ เป็นต้น นอกจากนี้ ยังมีการพัฒนาปรับปรุงสมดุลของการเดินแบบสองขาอีกด้วย สมดุลของการเดินสามารถแบ่งได้สองแบบหลัก คือ การเดินแบบสมดุลสถิต และการเดินแบบสมดุลพลวัต งานในยุคแรกนั้นจะพัฒนาให้เดินได้แบบสมดุลสถิต ต่อมา เป็นสมดุลกึ่งพลวัต และเป็นสมดุลพลวัต การพัฒนาตัวควบคุมการเดินของหุ่นยนต์ จำเป็นที่จะต้องใช้ความรู้ทาง ด้านกลศาสตร์ค่อนข้างมาก มีการใช้สมการที่มีความซับซ้อน

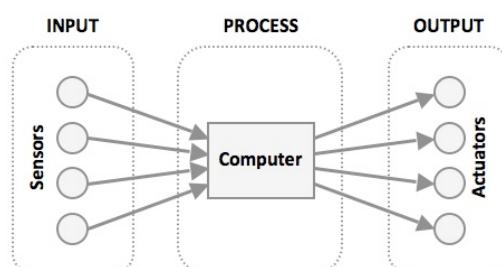
Zheng และคณะ (1988) พัฒนาหุ่นยนต์สองขาที่สามารถเดินบนพื้นราบได้ ให้สามารถเดินต่อเนื่องไปบนพื้นเอียงได้ด้วย พื้นเอียงที่ใช้มีลักษณะเป็นพื้นเอียงขึ้น หุ่นยนต์ที่ใช้งานนี้มีข้อต่อสะโพก (hip), ข้อเท้า (ankle) และลำตัว (torso) มีเซนเซอร์วัดแรงดึง (force sensor) ติดตั้งอยู่ที่ปลายเท้าและสันเท้าแต่ละข้างเพื่อใช้วัดตำแหน่งของน้ำหนักโดยรวม (center of gravity) ของหุ่นยนต์ การเดินของงานวิจัยจะพิจารณาเฉพาะการเดินในแนวหน้าหลัง โดยมีหลักการคือ การเดินบนพื้นเอียงโดยที่หุ่นยนต์ยังเดินในท่าทางเหมือนกับตอนที่เดินบนพื้นราบจะทำให้น้ำหนักโดยรวมของหุ่นยนต์เลื่อนไปข้างหลัง ดังนั้นการที่หุ่นยนต์ขับลำตัวไปด้านหน้าจะทำให้น้ำหนักโดยรวมของหุ่นยนต์กลับมาอยู่ตรงกลางของพื้นที่รับน้ำหนักเหมือนเดิม ซึ่งจะทำให้หุ่นยนต์มีความสมดุลได้ ดังนั้นข้อมูลที่ได้จากหน่วยวัดแรงดึงที่เท้าจะถูกนำมาคำนวณตลอดการเดินเพื่อให้ในการปรับเปลี่ยนมุกการขับของลำตัว การเดินบนพื้นราบเป็นแบบสมดุลสถิตและการเดินบนพื้นเอียงก็ยังคงเป็นแบบสมดุลสถิตเช่นกัน

Inaba และคณะ (1995) สร้างหุ่นยนต์เลียนแบบลิง (ape-like biped) ประกอบด้วยสองมือและสองขา มีการเดินแบบสมดุลสถิต งานวิจัยนี้มีความคิดว่าหากการทำให้หุ่นยนต์สองขาเดินได้โดยไม่ล้มแล้ว ควรจะทำหุ่นยนต์ที่สามารถลุกขึ้นเองได้หลังจากที่ล้มแล้วด้วย ดังนั้นในงานนี้ หุ่นยนต์ถูกพัฒนาให้สามารถเดิน เมื่อล้มแล้ว ก็สามารถพลิกตัวและลุกขึ้นมาเดินให้ได้

Kun และ Miller (1996) ได้นำโครงข่ายประสาทเทียม มาประยุกต์ใช้ในการปรับเปลี่ยนท่าทางการเดิน โดยอัตโนมัติของหุ่นยนต์สองขา การที่หุ่นยนต์สามารถปรับเปลี่ยนท่าทางได้โดยอัตโนมัตินี้มีประโยชน์ทำให้หุ่นยนต์เดินได้บนพื้นผิวหลากหลายลักษณะมากขึ้น ในงานนี้พิจารณาทั้งสมดุลในแนวหน้าหลัง (sigittal plane) และแนวซ้ายขวา (frontal plane) และการเดินของหุ่นยนต์เป็นแบบสมดุลพลวัต หลักการทำงานประกอบด้วยตัวสร้างท่าทางการเดินหนึ่งตัว และตัวปรับท่าทางการเดินทั้งแนวหน้าหลังและซ้ายขวาอีกหนึ่งตัว โดยค่าการปรับเปลี่ยนนั้นจะได้มาจากการรับสัญญาณจากเซ็นเซอร์ที่ติดตั้งอยู่ที่เท้า ความยาวการก้าวเท้า ความสูงของการยกเท้า เป็นต้น นอกจากนี้ในปัจจุบัน ทั้งสองตัวได้ใช้หลักการที่ใช้ในงานนี้ไปใช้กับการเดินของหุ่นยนต์อีกด้วย (Kun and Miller, 1997)

Hirai และคณะ (1998) พัฒนาหุ่นยนต์ชีวนิรภัย ซึ่งตัวหุ่นยนต์มีความคล้ายมนุษย์มาก สามารถเดินได้อย่างราบลื่นคล้านมันุษย์มากที่สุด เช่น สามารถเดินได้ในพื้นผิวนิดต่างๆ เดินได้บนพื้นเอียงขึ้นเอียงลง เดินขึ้นลงบันไดได้ เดินเข็นรถได้ เป็นต้น การเดินในทุกสถานการณ์เป็นการเดินแบบสมดุลพลวัต หุ่นยนต์สามารถเดินได้ด้วยความเร็วสูงสุด 4.7 กิโลเมตรต่อชั่วโมง หุ่นยนต์ประกอบไปด้วย แขนข้างละ 9 องศาอิสระ ขาข้างละ 6 องศาอิสระ ที่บริเวณหัวมีกล้องติดตั้งอยู่ 4 ตัว นอกจากนี้ยังมีอุปกรณ์ที่ใช้ในการรักษาสมดุลอีก 4 อุปกรณ์ ที่ติดตั้งบริเวณลำตัว และ Force sensor ที่ติดที่เท้าทั้งสองข้าง

องค์ประกอบของหุ่นยนต์ที่นำไปประยุกต์ใช้ในระบบ ซึ่งเราสามารถจำแนกออกเป็นส่วนหลักๆ ได้สามส่วนคือ ส่วนการรับรู้ ส่วนการประมวลผล และส่วนการขับเคลื่อน ทั้งหมด เมื่อนำมารวมเข้าด้วยกันแล้ว เราสามารถที่จะควบคุมการทำงานของหุ่นยนต์ชีวนิรภัยได้



รูปที่ 2.2: องค์ประกอบหลักของหุ่นยนต์ชีวนิรภัย

## การรับรู้ของหุ่นยนต์อิวามาโนยด์

การรับรู้ของหุ่นยนต์อิวามาโนยด์นั้นมีความยากมากกว่าหุ่นยนต์ชนิดอื่นๆ เพราะหุ่นยนต์จะมีการเคลื่อนที่และการเคลื่อนที่นั้นทำให้เซนเซอร์โดนรบกวนได้ ยกตัวอย่างเช่น ภาพที่ได้จากการกล้องนั้นอาจจะเบลอได้ถ้าความเร็วของชัตเตอร์ซ้าเกินไป หรือว่าภาพเปลี่ยนขณะที่กำลังกดชัตเตอร์ ข้อมูลตำแหน่งของตัวเองก็มีความแన่นอนที่น้อยกว่าหุ่นยนต์เคลื่อนที่ด้วยล้อ เพราะเซนเซอร์ที่วัดตำแหน่งเทียบกับเฟรมโลกไม่มีความเสถียร หุ่นยนต์ที่เคลื่อนที่ด้วยล้อปกติถ้าติดกล้อง ตัวกล้องจะมีความสูงจากพื้นคงที่ แต่หุ่นยนต์อิวามาโนยด์ไม่ใช่ โดยหุ่นยนต์อิวามาโนยด์นั้น จะต้องมีการคำนวณ forward kinematics จากเท้าที่สัมผัสกับพื้นมา焉กกล้องเพื่อหาตำแหน่งและการหมุนของกล้อง ส่วนการวัดตำแหน่งของตัวหุ่นยนต์นั้น โดยทั่วไปแล้วจะใช้เซนเซอร์ inertia measurement unit (IMU) และเซนเซอร์ Encoders สำหรับหาตำแหน่งของข้อต่อต่างๆ ปกติจะติดเซนเซอร์ IMU ไว้ที่ลำตัวของหุ่นยนต์ใกล้ๆ กับ center of mass ของหุ่นยนต์ ส่วน Encoder นั้นจะติดไว้ที่ข้อต่อของหุ่นยนต์

## การประมวลผลของหุ่นยนต์อิวามาโนยด์

ในปัจจุบันนี้หุ่นยนต์อิวามาโนยด์มีความสามารถในการคำนวณที่สูงมากเมื่อเทียบกับเมื่อก่อน บอร์ดที่เราสามารถเห็นได้โดยทั่วไปเช่น Raspberry Pi, Odroid, Intel NUCs ซึ่งตัวบอร์ดมีขนาดเล็กจึงทำให้เข้าไปอยู่ในตัวของหุ่นยนต์ได้ แม่บอร์ดพากนี้ยังมี GPUs และ CPU หลายคอร์อีกด้วย บางครั้งก็มีคันที่เอ้าพากบอร์ดพากนี้มาทำงานร่วมกันหลายๆ ตัว ประมวลผลแบบ pararell เพื่อที่จะเพิ่มประสิทธิภาพในการประมวลผล โดยเชื่อมต่อระหว่างกันผ่าน Ethernet network

## การขับเคลื่อนของหุ่นยนต์อิวามาโนยด์

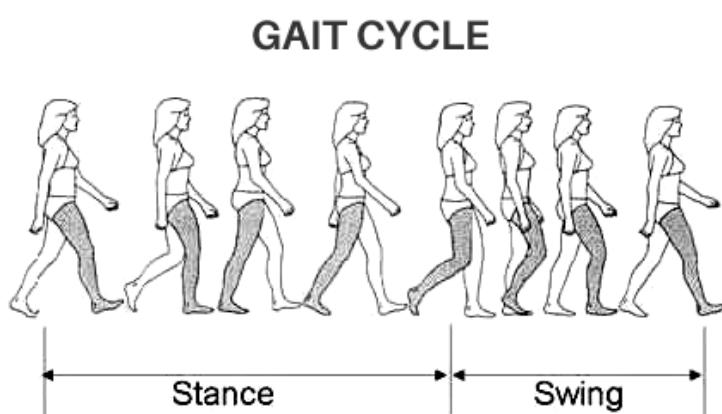
หุ่นยนต์อิวามาโนยด์ส่วนใหญ่จะมีข้อต่ออยู่หลายๆ จุด แต่ละข้อต่อจะมีตัวขับเคลื่อน ตัวขับเคลื่อนมีอยู่หลักๆ ส่องแบบคีอิ แบบกล้ามเนื้อของมนุษย์ และแบบมอเตอร์ที่ติดตรงที่ข้อต่อเลย ที่นิยมใช้คีอิแบบมอเตอร์ที่ติดที่ข้อต่อเลย เพราะทำให้ตัวของหุ่นยนต์มีขนาดเล็ก ใช้พื้นที่น้อย การใช้เส้นเอ็นดึงนั้นจะการทำให้ข้อต่อไปยังตำแหน่งที่ต้องการได้ยากกว่า ตัวขับเคลื่อนนั้นต้องการแรงมากน้อยขึ้นอยู่กับ น้ำหนักของตัวหุ่นยนต์ เพื่อที่จะทำให้หุ่นยนต์นั้นยังยืนได้

## 2.1.2 ทฤษฎีเกี่ยวกับมนุษย์

### 2.1.2.1 การวิเคราะห์การเดินของมนุษย์

การเคลื่อนที่ของหุ่นยนต์ชีวามโนย์นั้นจะเลียนแบบจากการเดินของมนุษย์ ดังนั้นการวิเคราะห์ลักษณะการเดินของมนุษย์ จะเป็นการศึกษาเพื่อทำความเข้าใจถึงธรรมชาติการเดิน ก่อนนำไปทำการออกแบบกลไกทางกลและระบบควบคุมของหุ่นยนต์ชีวามโนย์ การก้าวเดินของมนุษย์โดยปกติแล้ว จะมีลักษณะเป็นวัฏจักร วนซ้ำไปเรื่อยๆ ในทิศทางที่ต้องการจนกว่าจะทำการหยุดเดิน การทรงตัวในระหว่างการยืนหรือการเดินนั้น เป็นไปตามสัญชาตญาณซึ่งเกิดจากการรักษาความสมดุลของร่างกายบัน្តาในทุก<sup>1</sup> ส่งสัญญาณผ่านเส้นประสาทไปยังกล้ามเนื้อส่วนต่างๆ ที่ทำหน้าที่ให้เกิดการเคลื่อนที่

การเคลื่อนที่ของมนุษย์ในการเดินไปข้างหน้าสามารถแบ่งออกเป็นช่วงๆ ดังนี้



รูปที่ 2.3: วัฏจักรการเดินของมนุษย์

1. ช่วงเริ่มการวางเท้าเพื่อเข้าสู่ช่วงเริ่มต้นเหวี่ยงเท้า เป็นช่วงที่เท้าเกิดการกระแทกลงบนพื้นหลังจากทำการเหวี่ยงมาจากด้านหลัง โดยธรรมชาติมนุษย์จะทำการวางสันเท้าลงเพื่อลดแรงกระแทกที่เกิดขึ้นในช่วงนี้ ดังนั้นทางกายภาพในส่วนของสันเท้ามนุษย์จึงมีลักษณะอ่อนนุ่ม
2. ช่วงเริ่มต้นเหวี่ยงเท้าเพื่อเข้าสู่ช่วงเหวี่ยงเท้า หลังจากทำการวางสันเท้าลงกับพื้นแล้ว ข้อเข้าจะปรับมุมเพื่อให้ฝ่าเท้าแนวพื้นสนิท ขณะเดียวกันขาอีกข้างจะยกสูงขึ้นเพื่อถ่ายเทน้ำหนักไปยังเท้าที่เพิ่งวางลง
3. ช่วงเหวี่ยงเท้า เป็นช่วงที่ขาหนีงยกอยู่ในอากาศและขาที่วางแนบกับพื้นจะรองรับน้ำหนักทั้งหมดของร่างกาย
4. ช่วงเตรียมการวางเท้า เป็นช่วงที่ขาที่วางแนบกับพื้นเหวี่ยงไปข้างหน้าเพื่อเตรียมเข้าสู่ช่วงรองรับ ในขณะเดียวกันขาที่รับน้ำหนักอยู่จะทำการผลักตัวเพื่อเริ่มทำการถ่ายเทน้ำหนักไปข้างหน้า

<sup>1</sup>Rose, J. and Gamble, J., 1993, Human Walking, Williams & Wilkins, Philadelphia, pp. 10-44.

### 2.1.2.2 การวิเคราะห์ของศ้าอิสระของมนุษย์

การที่มนุษย์เราสามารถเคลื่อนที่ได้นั้น เป็นผลเนื่องมาจากการเคลื่อนที่ของข้อต่อต่าง ๆ ที่อยู่บนขา ซึ่งประกอบไปด้วย ข้อต่อส่วนสะโพก ข้อต่อส่วนหัวเข่า และข้อต่อส่วนข้อเท้า แรงบิดที่เกิดขึ้นของแต่ละข้อต่อมีความสัมพันธ์กับกัน ส่งผลให้เกิดเสถียรภาพในการเดินของมนุษย์ เมื่อวิเคราะห์ลักษณะโครงสร้างในแต่ละส่วน พบร่วมข้อต่อส่วนสะโพกมีลักษณะเป็นทรงกลม ทำให้ข้อต่อส่วนสะโพกสามารถหมุนได้ 3 องศาอิสระ ส่วนหัวเขาร่วมกับมนุษย์มีจุดต่อของข้อที่มีลักษณะเป็นทรงกลม สองลูกประกอบเข้าด้วยกันทำให้การเคลื่อนที่ถูกบังคับให้สามารถเคลื่อนที่ได้เพียง 1 องศาอิสระ ในส่วนของข้อเท้ามีลักษณะการเคลื่อนที่เหมือนสะโพกคือสามารถเคลื่อนที่ได้ 3 องศาอิสระ

จากทั้งหมดที่ได้ทำการวิเคราะห์มาข้างต้นพบว่าในขาหนึ่งข้างของมนุษย์ประกอบด้วย 7 องศาอิสระ ซึ่งส่งผลให้การเคลื่อนที่ของมนุษย์มีความคล่องแคล่วสูง แต่ในทางออกแบบกลไกการเดินและการควบคุม ของหุ่นยนต์ สองขาถือว่ามีจำนวนองศาอิสระเกินความจำเป็นในการเคลื่อนที่บนปริภูมิ(rspace) และยกต่อการควบคุม (under actuated) ดังนั้นการกำหนดจำนวนองศาอิสระเพื่อให้หุ่นยนต์เดินได้เสื่อมมนุษย์จึงมีผลในการออกแบบกลไกทางกลและการควบคุมของหุ่นยนต์สองขา

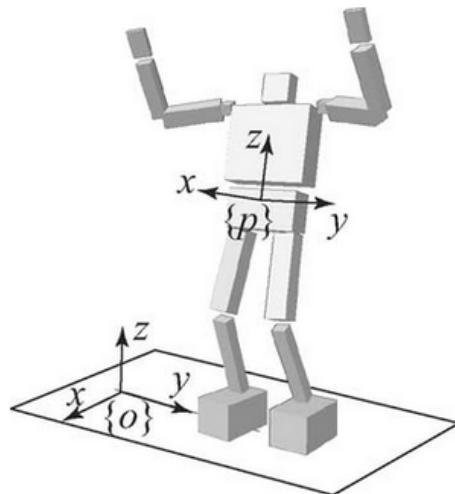
ข้อต่อ	องศาอิสระ	องศาสารหมุน	
		สูงสุด	ต่ำสุด
หัว	$\theta_x$	+60	-30
	$\theta_y$	+70	-70
	$\theta_z$	+80	-80
หลัง	$\theta_x$	+30	-30
	$\theta_y$	+55	-55
	$\theta_z$	+45	-45
หัวไห่ล'	$\theta_x$	+180	-80
	$\theta_y$	+45	-135
	$\theta_z$	+30	0
ศอก	$\theta_x$	0	-155
สะโพก	$\theta_x$	+120	-40
	$\theta_y$	+40	-50
	$\theta_z$	+60	-50
หัวเข่า	$\theta_x$	0	-130
ข้อเท้า	$\theta_x$	+30	-60
	$\theta_y$	+45	-20
	$\theta_z$	+20	-60

ตารางที่ 2.1: ความสามารถในการหมุนของแต่ละข้อต่อของมนุษย์

ผู้เขียนได้ข้อสรุปในการออกแบบขาหนึ่งข้างของหุ่นยนต์ให้มีองศาอิสระเท่ากับ 6 องศาอิสระ และได้ใช้ตัวขับเคลื่อนเป็นแบบดิจิตอล (Digital Servo) ของบริษัท Robotics เนื่องจากภายในมีตัวรับสัญญาณของตัวขับเคลื่อนต่างๆ และตัวขับเคลื่อนนี้ถูกออกแบบมาให้สามารถติดตั้ง และสั่งการได้ง่าย

### 2.1.3 ทฤษฎีเกี่ยวกับหุ่นยนต์อิวามาโนยด์

#### 2.1.3.1 ส่วนประกอบของหุ่นยนต์อิวามาโนยด์

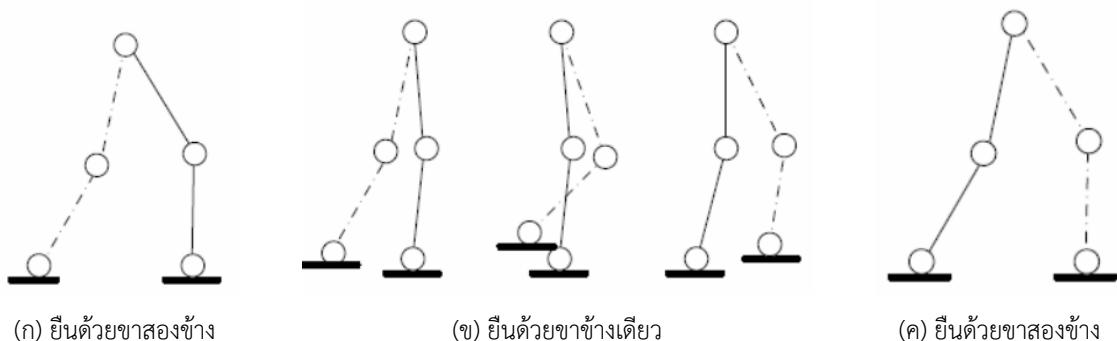


รูปที่ 2.4: ส่วนประกอบของหุ่นยนต์อิวามาโนยด์

หุ่นยนต์อิวามาโนยด์ประกอบด้วยก้านต่อหอยๆ ก้านที่นำมาต่อกัน ลักษณะโครงสร้างนั้นจะเป็นแบบแบบโข่าย เปิด และแต่ละก้านต่อจะเชื่อมต่อกันด้วยข้อต่อ เรากำหนดที่จะแบ่งจ่าจุกเป็น 2 ส่วน ส่วนแรกคือ ส่วนก้านต่อของลำตัวหุ่นยนต์ (Torso) ซึ่งเราระบุที่จะเหมาร่วมไปถึงส่วนแขนกับหัวด้วย และในส่วนที่สองคือ ส่วนก้านต่อของขาหุ่นยนต์ (Legs) ซึ่งเป็นส่วนของหุ่นยนต์ทั้งสองข้างที่สามารถนำไปใช้สัมผัสนับพื้นได้ ทั้งสองก้านต่อเนี้ยูก็เชื่อมต่อกันด้วยส่วนของสะโพก (Hip) ที่อยู่ระหว่างส่วนลำตัวกับส่วนของขาหุ่นยนต์ ดังรูปที่ 2.4

#### 2.1.3.2 วัฏจักรการเดินของหุ่นยนต์อิวามาโนยด์

วัฏจักรการเดินของหุ่นยนต์ คือ การที่หุ่นยนต์จะต้องมีการถ่ายน้ำหนักไปมาระหว่างเท้าซ้ายและเท้าขวา มีบางช่วงที่น้ำหนักตกลงบนเท้าซ้ายได้ข้างหนึ่งหรือทั้งสองข้างพร้อมกัน สามารถแบ่งออกเป็นช่วงได้สองช่วง คือ ช่วงการยืนด้วยขาซ้ายเดียว และช่วงการยืนด้วยขาทั้งสองข้าง



รูปที่ 2.5: วัฏจักรการเดินของหุ่นยนต์อิวามาโนยด์

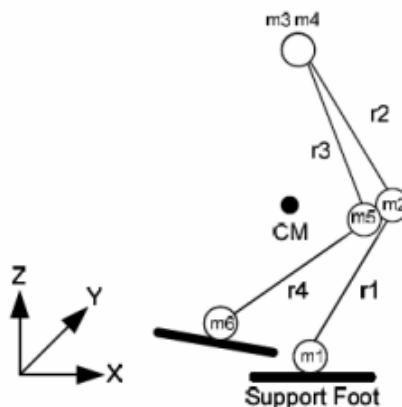
1) การยืนด้วยขาซ้ายเดียว : เป็นช่วงที่มีเท้าของหุ่นยนต์สัมผัสนับพื้นเพียงข้างเดียว ส่วนเท้าอีกข้างของหุ่นยนต์จะถูกยกออกจากพื้น โดยที่ไม่มีส่วนใดของขาข้างนั้นสัมผัสนับพื้นเลย ช่วงนี้จะเกิดขึ้นเมื่อมีการแกว่งเท้าจากข้างหลังไปข้างหน้า ดังภาพที่ 2.5(ก)

2) การยืนด้วยขาสองข้าง : เป็นช่วงที่เท้าทั้งสองข้างของหุ่นยนต์สัมผัสกับพื้น ช่วงนี้จะเกิดตั้งแต่หุ่นยนต์วางเท้าขณะที่สันเท้าแตะกับพื้น ไปจนถึง ปลายเท้าของขาอีกข้างหลุดออกจากพื้น

การเดินได้โดยไม่ล้มนั้น ตัวหุ่นยนต์จะต้องรักษาสมดุลของการเดินให้ได้ตลอดช่วงเวลาของการเดิน ซึ่ง สมดุลของการเดินแบบสองขาสามารถแบ่งตามลักษณะการเดินและการถ่ายน้ำหนักได้เป็น 2 รูปแบบหลัก คือ การเดินแบบสมดุลสถิต (static balance walking) และ การเดินแบบสมดุลพลวัต (dynamic balance walking)

### 2.1.3.3 การสร้างและการควบคุมการเดินแบบสมดุลสถิต

การเดินของหุ่นยนต์ในลักษณะนี้ น้ำหนักตัวหุ่นยนต์จะไม่มีการเคลื่อนไหวออกนอกบริเวณฐานรับน้ำหนัก (Supporting Area) ตลอดช่วงเวลาการเดิน ไม่ว่าจะเป็นช่วงเวลาที่รับน้ำหนักด้วยเท้าข้างเดียวหรือทั้งสองข้างก็ตาม หมายความว่า โครงสร้างของหุ่นยนต์จะไม่ล้มแน่นอน เนื่องจากการสร้างรูปแบบการเดินด้วยวิธีนี้จะควบคุมให้ตำแหน่งของจุดรวมมวล (CoM) อยู่ภายใต้พื้นที่ฐานรับน้ำหนักของหุ่นยนต์ตลอดเวลา



รูปที่ 2.6: การควบคุมตำแหน่งของจุดรวมมวลให้อยู่ในพื้นที่ฐาน

ข้อดีของการสร้างและควบคุมการเดินของหุ่นยนต์ด้วยวิธีนี้คือ สามารถสร้างรูปแบบการเดินได้โดยที่มีความซับซ้อนไม่มากนัก สามารถสั่งให้หุ่นยนต์หยุดค้างในท่าทางใดๆ ก็ได้ตลอดเวลาโดยหุ่นยนต์ไม่ล้ม หุ่นยนต์ที่มีฝ่าเท้าใหญ่จะทำให้ง่ายต่อการก้าวเดินมากขึ้น นอกจากการควบคุมการก้าวขาแล้วอาจเพิ่มการควบคุมส่วนลำตัวเพิ่มเติม เพื่อเป็นการเพิ่มเสถียรภาพในการเดินและการถ่ายเท้น้ำหนัก โดยที่อาจจะมีการเพิ่มเซนเซอร์วัดแรงที่ฝ่าเท้าเพื่อตรวจสอบการกระจายแรงกดที่ฝ่าเท้า เพื่อตรวจสอบว่าตำแหน่งของจุดรวมน้ำหนักอยู่บนพื้นที่ฝ่าเท้า หรือไม่ หรือเพื่อตรวจสอบเสถียรภาพของการเดินเพื่อแก้ไขท่าทางการเดินไม่ให้เกิดการล้ม

ข้อเสียของการควบคุมการเดินด้วยวิธีนี้คือ หุ่นยนต์จะใช้เวลาในการก้าวเดินมาก ใช้พลังงานในการเดินมากกว่าการเดินแบบสมดุลพลวัต และท่าทางที่ได้จะมีความแตกต่างจากท่าทางการเดินของมนุษย์

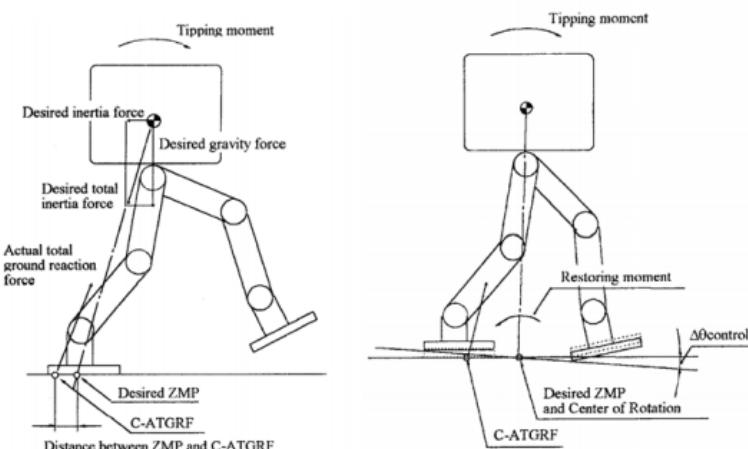
### 2.1.3.4 การสร้างและการควบคุมการเดินแบบสมดุลพลวัต

การสร้างรูปแบบการเดินและควบคุมการเดินในลักษณะนี้ท่าทางการเดินของหุ่นยนต์นั้นจะคล้ายกับการเดินของมนุษย์มากกว่าแบบสถิต เนื่องจากมีหลักการในการสร้างท่าทางที่เหมือนกับการเดินของมนุษย์ซึ่งมีขั้นตอนดังนี้คือ เอียงตัวให้ล้มไปในทิศทางที่ต้องการเดิน เมื่อเริ่มเกิดการล้มขึ้นหุ่นยนต์จะเปลี่ยนตำแหน่งการวางเท้าไปยังตำแหน่งใหม่ เพื่อปรับให้โครงสร้างเข้าสู่สภาวะสมดุลอีกครั้ง

โดยธรรมชาติแล้วมนุษย์มีการถ่ายน้ำหนักในขณะที่เคลื่อนที่หรือยืนอยู่กับที่เพื่อรักษาสมดุลของท่าทางนั้นไว้ แต่หากการถ่ายโอนน้ำหนักนั้นเกิดสภาวะไม่สมดุล ร่างกายจะปรับสภาพโดยการเคลื่อนตำแหน่งของเท้าซึ่งเป็นพื้นที่ฐานออกจากเดิมไปยังตำแหน่งใหม่ เพื่อรักษาสมดุลไว้ หลักการดังกล่าวถูกนำมาใช้กับการควบคุม

การเดินของหุ่นยนต์อิวามาโนย์ ในขณะที่หุ่นยนต์กำลังเคลื่อนไหว ผลจากแรงเฉือนดูดของโลกมีผลต่อการเพิ่มและลดความเร่งให้การเดินของหุ่นยนต์ แรงเหล่านี้เรียกว่าแรงเฉียบรวมของการเคลื่อนที่ และเมื่อเท้าหุ่นยนต์สัมผัสกับพื้นจะได้รับผลกระทบของแรงนี้ เรียกว่า แรงปฏิกิริยาจากพื้น

การตัดกันระหว่างแรงปฏิกิริยาจากพื้นและแนวแรงเฉียบรวม ตำแหน่งนั้นหากทำให้โมเมนต์เท่ากับศูนย์ เรียกจุดดั้นนี้ว่าจุดโมเมนต์ศูนย์ ( $ZMP_{robot}$ ) และจุดที่แรงปฏิกิริยาลงสู่พื้นว่า จุดปฏิกิริยาพื้นฐาน ท่าทางการเดินของหุ่นยนต์จะถูกกำหนดและถูกส่งให้กับชุดควบคุมข้อต่อจุดต่างๆ ของหุ่นยนต์ โดยให้สอดคล้องกับแรงเฉียบรวมที่เกิดขึ้นจากการคำนวณ เรียกว่าแรงเฉียบรวมเป้าหมาย และจุดโมเมนต์ศูนย์ที่ได้จากการคำนวณเรียกว่าจุดโมเมนต์ศูนย์เป้าหมาย ( $ZMP_{target}$ ) เมื่อหุ่นยนต์เกิดสมดุลในขณะที่ทำการเดินได้อย่างสมบูรณ์ แนวแกนของแรงเฉียบรวมเป้าหมายและแรงปฏิกิริยาที่พื้นจะเป็นตำแหน่งเดียวกัน แต่ในขณะที่หุ่นยนต์เดินผ่านพื้นผิวที่มีความชุ่มชื้นหรือไม่เรียบตัวแห้งน้ำ ก็จะไม่ใช่ตำแหน่งเดียวกันทำให้หุ่นยนต์เกิดการล้มได้ แรงที่ทำให้เกิดการล้มนี้เกิดจากตำแหน่งของจุดโมเมนต์ศูนย์และตำแหน่งแรงปฏิกิริยาร่วมที่พื้นไม่ตรงกัน ซึ่งเป็นสาเหตุหลักที่ทำให้เกิดความไม่สมดุลขึ้น และเมื่อหุ่นยนต์เสียสมดุลระบบที่จะสามารถป้องกันการล้มและทำให้หุ่นยนต์เดินต่อไปได้อย่างต่อเนื่องคือ ระบบควบคุมแรงปฏิกิริยา ระบบควบคุมจุดโมเมนต์ศูนย์ และระบบควบคุมการวางแผนเท้า



รูปที่ 2.7: การควบคุมตำแหน่งของจุดโมเมนต์ศูนย์ให้ตรงกับแรงปฏิกิริยาร่วม

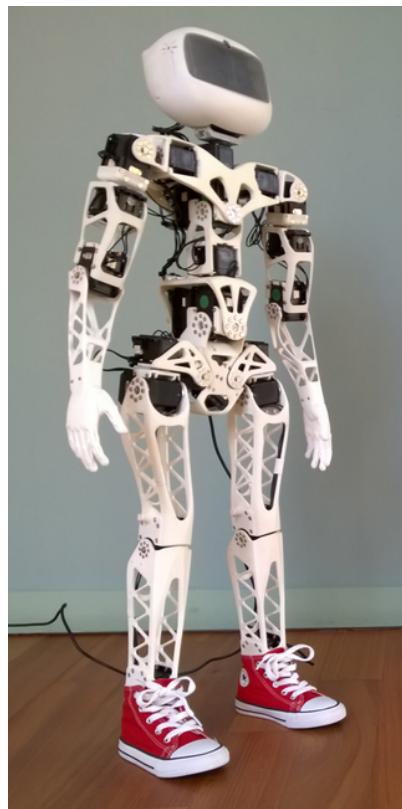
อย่างไรก็ตาม การสร้างท่าทางการเดินในลักษณะนี้ต้องใช้สมการในการคำนวณที่ซับซ้อนมาก เนื่องจากต้องหาความสัมพันธ์ระหว่างองค์ประกอบหลายส่วน เช่น น้ำหนักของโครงสร้างในแต่ละส่วน แรงบิดที่แต่ละข้อต่อ และโมเมนต์โดยรวมของระบบ นอกจากนี้ยังต้องใช้อุปกรณ์การตรวจวัดต่างๆ เช่น เชเซอร์วัดแรง เชเซอร์วัดมุม เชเซอร์วัดแรงบิด ติดตั้งตามจุดต่างๆ ของโครงสร้างเพื่อวัดค่าอุกมา ก่อนที่จะทำการคำนวณตำแหน่ง และสร้างท่าทางการเดินของหุ่นยนต์อิวามาโนย์ ท่าทางการเดินที่ได้จากการควบคุมด้วยวิธีนี้ จะมีความคล้ายคลึงกับท่าทางการเดินของมนุษย์มาก

### 2.1.3.5 จุดศูนย์กลางมวลของหุ่นยนต์

หากต้องการให้หุ่นยนต์สามารถที่จะทรงตัวอยู่ได้โดยไม่ล้มนั้น จึงต้องรู้ตำแหน่งจุดศูนย์กลางมวลของหุ่นยนต์ตลอดเวลา และต้องให้จุดศูนย์กลางมวลอยู่ต่ำในบริเวณฐานรับน้ำหนักของหุ่นยนต์โดยหากพื้นที่ที่ฝ่าเท้าสัมผัสกับพื้น วิธีการนี้เป็นวิธีการทำงานทางสถิตศาสตร์

#### 2.1.4 ตัวอย่างหุ่นยนต์ชีวามโนยด์

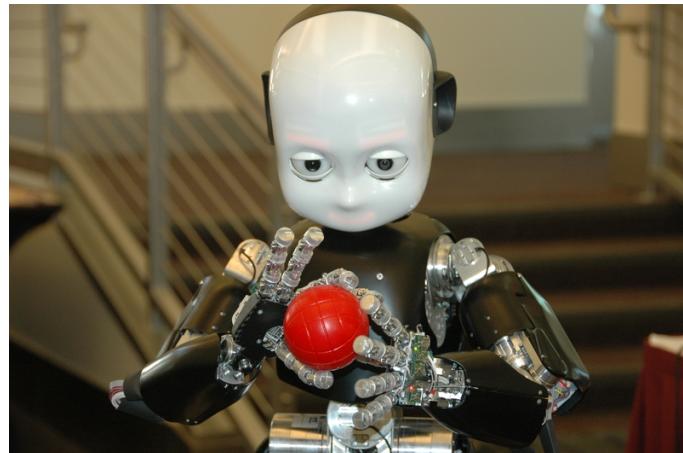
Poppy Humanoid



รูปที่ 2.8: หุ่นยนต์ชีวามโนยด์ปีอปปี

หุ่นยนต์ชีวามโนยด์ปีอปปี ถูกสร้างขึ้นมาเพื่อใช้ในงานศิลปะ การวิจัยและการศึกษาโดยเฉพาะ หุ่นยนต์ปีอปปีประกอบด้วยส่วนของขาเร็วและซอฟแวร์ที่เปิดเป็นโอเพนซอร์ซให้ผู้ที่สนใจสามารถเข้ามาศึกษาได้ โปรแกรมของหุ่นยนต์ใช้โมดูลที่มีชื่อว่า Pypot ที่เป็นส่วนเสริมของภาษา Python ในการพัฒนาซอฟแวร์ ทุกคน สามารถเข้าถึงข้อมูลเชิงเทคนิคของหุ่นยนต์ชีวามโนยด์ปีอปปีได้ เช่น ส่วนรายละเอียดการทำงาน คลิปวีดีโอสอน การประกอบ การใช้ระบบจำลอง และการพัฒนาต่างๆผ่านทางเว็บไซต์ <http://www.poppy-project.org> หุ่นยนต์ปีอปปีมีส่วนของโครงสร้างที่ผลิตมาจากพลาสติก PLA และ ABS โดยใช้เทคนิคการขึ้นรูปด้วยเครื่องพิมพ์สามมิติ ตัวขับเคลื่อนข้อต่อต่างๆใช้เป็น Dynamixel Digital Servo และควบคุมคำสั่งของตัวขับเคลื่อนด้วยคอมพิวเตอร์ขนาดเล็ก Odroid UX4 ใช้ระบบปฏิบัติการ Ubuntu 14.04 ตัวของหุ่นยนต์มีความสูง 83 เซนติเมตร น้ำหนัก 3.5 กิโลกรัม ใช้เซนเซอร์วัดมุมเอียง เป็น IMU ที่มีองศาอิสระเท่ากับ 9 องศาอิสระ ในการควบคุม เส้นสายภาพในการเดินของตัวเอง มีองศาอิสระหรือจำนวนตัวขับเคลื่อนทั้งหมด 25 องศา ประกอบไปด้วย ขา ข้างละ 6 องศาอิสระ แขนข้างละ 4 องศาอิสระ ลำตัว 3 องศาอิสระ และ หัว 2 องศาอิสระ

## iCub Humanoid



รูปที่ 2.9: หุ่นยนต์อิวามานอยด์ไอคัพ

หุ่นยนต์อิวามานอยด์ไอคัพ ถูกออกแบบโดยมหาวิทยาลัยหลายแห่งในยุโรปรวมกลุ่มกันขึ้นมาในชื่อ RobotCub และถูกสร้างขึ้นโดย Istituto Italiano di Tecnologia (IIT) ตัวหุ่นยนต์ไอคัพนั้นมีความสูงอยู่ที่ 1 เมตร น้ำหนักโดยรวมทั้งหมดประมาณ 22 กิโลกรัม วัสดุที่ใช้ในการสร้างแตกต่างกันไปในแต่ละส่วนของร่างกายโดยจะใช้ aluminum alloy Al6082 สำหรับส่วนที่ต้องรับภาระความเครียดบนน้อย ใช้ aluminum alloy 7075(Ergal) สำหรับส่วนที่ต้องรับภาระความเครียดปานกลางถึงสูง และใช้ Stainless Steel 17-4PH ในส่วนของเพลาข้อต่อต่างๆ เพื่อให้มีความแข็งแรงสูง ตัวหุ่นยนต์ถูกออกแบบให้มีลักษณะเหมือนเด็กอายุ 3-4 ขวบ ควบคุมโดยใช้บอร์ดไมโครคอนโทรเลอร์เป็นรุ่น PC104 Controller ภาษาที่ใช้ในการพัฒนาใช้เป็นภาษา C++ ในการเขียนโปรแกรม การติดต่อสื่อสารกับตัวขับเคลื่อนหรือมอเตอร์ตามข้อต่อต่างๆ และเซนเซอร์ ผ่านทางproto-col CAN Bus เพื่อทำให้ใช้สายน้อยลง ใช้เส้นเอ็นในการส่งถ่ายแรงขับเคลื่อนไปยังส่วนของข้อต่อส่วนมือและขา นิ้วของหุ่นยนต์ถูกร้อยด้วยสายเคเบิลเคลือบ Teflon อยู่ภายนอก และคล้ายตัวกลับสู่สภาพแวดล้อมได้ด้วยแรงของสปริง เช่นเชอร์วัดมุมของข้อต่อแต่ละตัวใช้การออกแบบให้มี Hall-effect ติดอยู่ ช่วยในการอ่านค่าของตำแหน่งและความเร็วที่เกิดขึ้นที่ข้อต่อนั้น หุ่นยนต์ไอคัพมีองศาอิสระรวมกันทั้งหมด 53 องศาอิสระ ประกอบไปด้วย แขนข้างละ 7 องศาอิสระ มือข้างละ 9 องศาอิสระ หัว 6 องศาอิสระ ลำตัว 3 องศาอิสระ และขาข้างละ 6 องศาอิสระ ในส่วนของหัวจะประกอบไปด้วย กล้องสองตัวเพื่อทำการติดตามใบหน้า ไมโครโฟนสำหรับรับเสียงจากสภาพแวดล้อมภายนอก และไฟแสดงอารมณ์บริเวณปากและคิ้ว หุ่นยนต์ไม่ได้ถูกออกแบบให้มีการทำงานเป็นแบบอัตโนมัติ ซึ่งก็คือตัวหุ่นยนต์นั้นไม่มีแบตเตอรี่ภายในตัว แต่ใช้แหล่งพลังงานจากการส่งเข้าไปผ่านสายเคเบิล และเชื่อมต่อกับอินเทอร์เน็ตผ่านสายแลน (LAN)

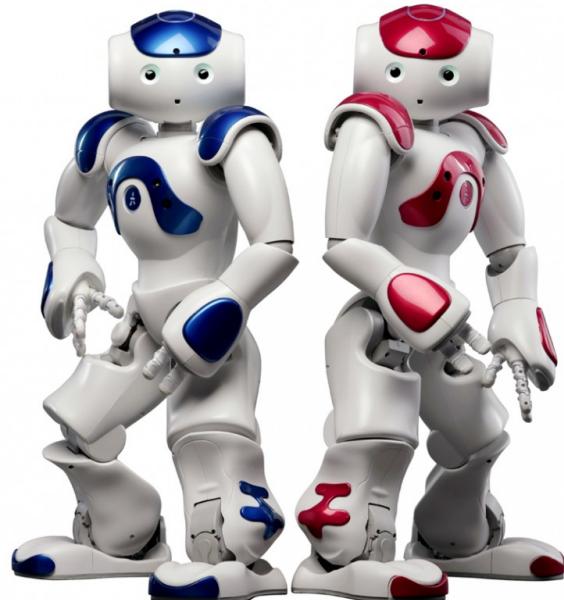
### Darwin-OP Humanoid



รูปที่ 2.10: หุ่นยนต์อิวามาโนย์ดาร์วิน

หุ่นยนต์อิวามาโนย์ดาร์วิน (Darwin-OP) เป็นชื่อที่ย่อมาจากคำว่า Dynamic Anthropomorphic Robot with Intelligence–Open Platform เป็น OpenSource Platform ที่ถูกออกแบบและพัฒนาโดย Korean robot manufacturer Robotis โดยมีความร่วมมือกับ Virginia polytechnic institute and state university, Purdue university และ University of Pennsylvania หุ่นยนต์อิวามาโนย์ดาร์วินมีความสามารถในการรับภาระโหลดได้สูง เนื่องจากมีการพัฒนามอเตอร์เป็นของตัวเอง อีกทั้งยังมีความสามารถในการเคลื่อนที่แบบ พลวัต (Dynamic) หุ่นยนต์ดาร์วิน มีองศาอิสระทั้งหมด 20 องศาอิสระ ซึ่งประกอบไปด้วย ขาข้างละ 6 องศาอิสระ แขนข้างละ 3 องศาอิสระ และหัว 2 องศาอิสระ ขับเคลื่อนข้อต่อต่างๆด้วยเซอร์โวมอเตอร์ Dynamixel MX-28T ที่มีการเชื่อมต่อแบบ RS485 ในการประยัดสายที่ใช้ในการสั่งการ มอเตอร์แต่ละตัวมีเซนเซอร์วัดตำแหน่ง และความเร็วอยู่ภายใน ตัวหุ่นยนต์มีความสูงทั้งหมด 45 เซนติเมตร มีน้ำหนักโดยประมาณ 2.9 กิโลกรัม ระบบภายในใช้คอมพิวเตอร์ขนาดเล็กเป็น 1.6 GHz Intel Atom Z530 (32 bit) ใช้คอนโทรลเลอร์ ARM CortexM3 STM32F103RE 72 MHz และมีเซนเซอร์วัดมุมเอียงเป็น 3-axis gyro, 3-axis accelerometer เพื่อช่วยในการควบคุมเสถียรภาพในการเดิน

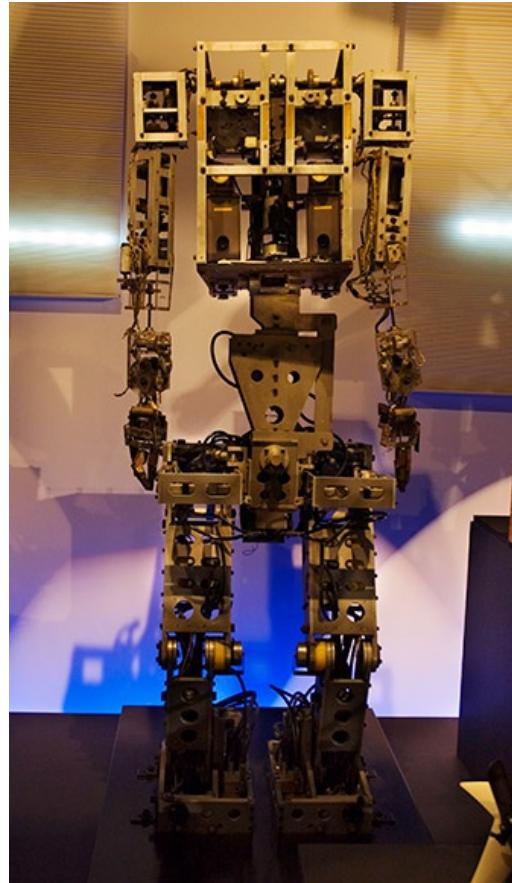
### Nao Humanoid



รูปที่ 2.11: หุ่นยนต์อิวามานอยด์นาโอะ

หุ่นยนต์อิวามานอยด์นาโอะ เป็นหุ่นยนต์อิวามานอยด์ขนาดกลาง ถูกผลิตมาจากประเทศฝรั่งเศส พัฒนาโดยบริษัท Aldebaran Robotics เมื่อปี 2004 และในปี 2007 หุ่นยนต์อิวามานอยด์นาโอะได้นำไปแทนที่หุ่นยนต์สูนๆ ของ Sony ชื่อ Aibo ขณะถูกใช้ในการแข่งขัน RoboCup Standard Platform League (SPL) หุ่นยนต์นาโอะได้ถูกนำไปใช้ใน Robocup 2008 และ 2009 หุ่นยนต์นาโอะถูกพัฒนาออกแบบมาหลายรุ่น มีองศาอิสระตั้งแต่ 14 องศาอิสระ 21 องศาอิสระ และ 25 องศาอิสระ สำหรับเพื่องานวิจัยนั้นมีถึง 25 องศาอิสระ โดยเพิ่มเติมมือสองข้างเอวเข้าไปเพื่อให้สามารถยกจับสิ่งของได้ ภายในหุ่นยนต์ถูกควบคุมด้วยระบบปฏิบัติการ NAO 2.0 (Linux-based) ตัวหุ่นยนต์มีความสูง 58 เซนติเมตร น้ำหนัก 4.3 กิโลกรัม ส่วนเซนเซอร์การรับรู้ต่างๆ จะประกอบไปด้วยเซนเซอร์วัดมุมอุ่ย 3-axis gyro, 3-axis accelerometer, Ultrasound captors, ไมโครโฟน 4 ตัว ลำโพง 2 ตัว กล้อง 2 ตัว เพื่อใช้ประโยชน์ในการทำงานวิจัยต่างๆ ตอนนี้ความสามารถของหุ่นยนต์นาโอะที่ทำได้คือ สามารถเห็นสีได้ เดินขึ้นลงบันไดและทางลาดชันได้ ระหว่างการเดินนั้นสามารถวางแผนการวางเท้าได้อย่างรวดเร็ว อีกทั้งยังสามารถที่จะเดินหลบหลีกสิ่งกีดขวางได้ด้วย

## Wabot



รูปที่ 2.12: หุ่นยนต์อิวามานอยด์ว้าบอท

หุ่นยนต์อิวามานอยด์มีการพัฒนาในช่วงแรกเริ่มมาตั้งแต่ปี 1973 หุ่นยนต์อิวามานอยด์ ตัวแรกชื่อ Wabot-1 เริ่มสร้างโดยมหาวิทยาลัย Waseda ที่ประเทศญี่ปุ่น ตัวของหุ่นยนต์มีความสูง 180 เซนติเมตร น้ำหนัก 210 กิโลกรัม โดยหุ่นยนต์สามารถติดต่อสื่อสารกับมนุษย์ได้ด้วยภาษาญี่ปุ่น สามารถวัดระยะและทิศทางได้โดยใช้การรับรู้ผ่านทางตาและหูเทียม หุ่นยนต์ Wabot-1 นั้นสามารถเดินได้ด้วยขาของตนเองที่มีสองข้าง สามารถหยิบและเคลื่อนย้ายวัสดุด้วยมือ ต่อมาในปี 1984 มหาวิทยาลัย Waseda ได้พัฒนาหุ่นยนต์อิวามานอยด์ที่ชื่อ Wabot-2 โดยหุ่นยนต์สามารถสื่อสารกับมนุษย์ได้ สามารถถ่ายโน้ตเพลงและเล่นดนตรีโดยใช้ electronic organ แบบง่ายๆ ได้ และในปี 1985 บริษัท Hitachi ได้สร้างหุ่นยนต์ WHL-11 ที่มีสองขาเหมือนมนุษย์ ซึ่งสามารถเดินแบบสมดุลสถิต (Static Walking) บนพื้นราบได้ด้วยความเร็ว 13 วินาทีต่อหนึ่งก้าว และสามารถเลี้ยวได้ซ้ายและขวาได้

## 2.2 การออกแบบโครงสร้างของหุ่นยนต์

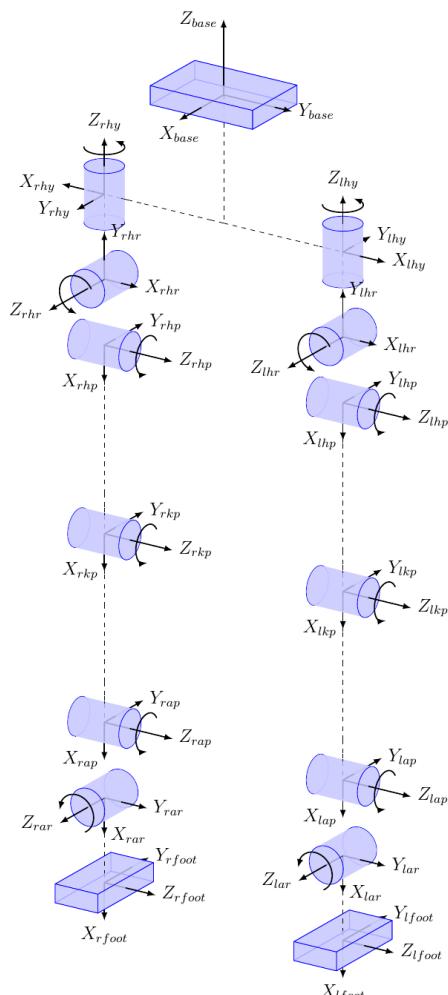
### 2.2.1 ความแตกต่างระหว่างโครงสร้างของมนุษย์กับโครงสร้างของหุ่นยนต์

#### 2.2.1.1 ความแตกต่างขององค์การเสรี

เนื่องจากลักษณะข้อต่อของมนุษย์มีความซับซ้อนมากกว่าโครงสร้างของหุ่นยนต์ ทำให้ข้อต่อแต่ละจุดของมนุษย์นั้นสามารถหมุนได้หลายทิศทาง รวมถึงขอบเขตของการหมุนของข้อต่อในแต่จุดก็มีความแตกต่างกัน ใน การนำรูปแบบการเดินของมนุษย์ไปใช้กับหุ่นยนต์จึงต้องปรับค่ามุมที่ข้อต่อให้มีความเหมาะสมกับโครงสร้าง และ ข้อจำกัดเกี่ยวกับการหมุนของข้อต่อจุดต่างๆ ของหุ่นยนต์ที่จะใช้ทดสอบด้วย

#### 2.2.1.2 ความแตกต่างของอัตราส่วน

นอกจากความแตกต่างขององค์การเสรี (DoF) ระหว่างมนุษย์กับหุ่นยนต์แล้ว ความแตกต่างของอัตราส่วนระหว่างโครงสร้างแต่ละส่วนของมนุษย์กับหุ่นยนต์เป็นอีกสาเหตุหนึ่ง ที่ต้องทำการปรับแต่งใหม่มีความเหมาะสม เนื่องจากความยาวของโครงสร้างแต่ละส่วน รวมทั้งระยะห่างระหว่างจุดหมุนแต่ละจุดของมนุษย์กับหุ่นยนต์ที่มี ความแตกต่างกัน ดังนั้นจึงต้องกำหนดระบบพิกัดสำหรับหุ่นยนต์ที่มีความยาวต่างกัน เช่น ขาหน้าและขาหลัง ที่มีความยาวต่างกัน ทำให้ในการอ้างอิงจุดหมุน และความยาวของโครงสร้างในส่วนต่างๆ

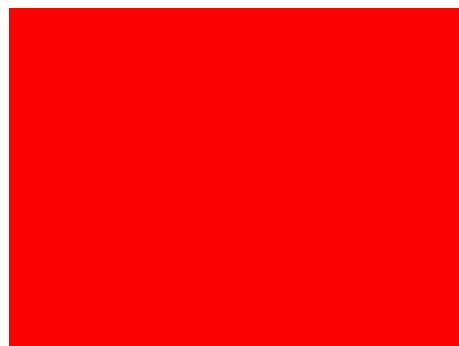


รูปที่ 2.13: ตัวอย่างตำแหน่งและการหมุนของข้อต่อของหุ่นยนต์เพื่อการอ้างอิง

### 2.2.1.3 กำลังและประสิทธิภาพของมอเตอร์

ความสามารถในการรับน้ำหนักของข้อต่อแต่ละจุดมีความแตกต่างกัน การเคลื่อนไหวของมนุษย์นั้นจะมีกล้ามเนื้อ และเลี้นเอ็นเป็นตัวออกแรงดึงส่วนต่างๆของร่างกายเพื่อทำให้เกิดการเคลื่อนไหวซึ่งจะมีความยืดหยุ่นและแรงดึงที่มีค่าสูง สำหรับการเคลื่อนไหวของหุ่นยนต์ จะใช้การบิดแกนของเซอร์โวมอเตอร์ (Servo Motor) หรือมอเตอร์ที่ติดอยู่ที่ข้อต่อจุดต่างๆ ทำให้ความสามารถในการรับน้ำหนัก แรงบิดและความยืดหยุ่นที่ข้อต่อขึ้นกับกำลังของมอเตอร์เป็นหลัก การสร้างท่าทางของหุ่นยนต์จึงต้องคำนึงถึงความสามารถในการรับน้ำหนักและกำลังของเซอร์โวมอเตอร์ที่ใช้ด้วยเข่นกัน

### 2.2.2 วัสดุและการขึ้นรูปโครงสร้างของหุ่นยนต์อิมานอยด์

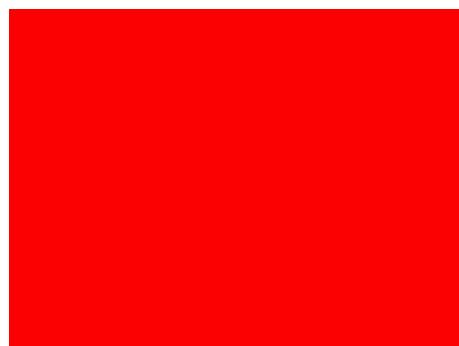


รูปที่ 2.14: ร้อแก๊ไข

### 2.2.3 อุปกรณ์ที่ใช้ในหุ่นยนต์อิมานอยด์

#### 2.2.3.1 ตัวขับเคลื่อน

ในการสร้างหุ่นยนต์อิมานอยด์นั้นระบบการขับเคลื่อนก็ถือว่าเป็นเรื่องสำคัญ เนื่องจากว่าถ้าหากระบบขับเคลื่อนไม่สามารถทำงานได้อย่างเต็มประสิทธิภาพ หรือหากมีการออกแบบที่ผิดพลาด จะส่งผลทำให้หุ่นยนต์อิมานอยด์นั้นมีประสิทธิภาพในการทำงานลดลงตามไปด้วย ภายในงานวิจัยนี้ทางผู้จัดทำได้ใช้ตัวขับเคลื่อนเป็น Dynamixel digital servo EX-106 ซึ่งเป็นเซอร์โวมอเตอร์ที่เหมาะสมสำหรับทำหุ่นยนต์โดยเฉพาะ ภายในประกอบไปด้วย มอเตอร์กระแสตรง ชุดเฟืองมอเตอร์ ไดรเวอร์คอนโทรเลอร์ สามารถเชื่อมต่อกันผ่าน BUS RS-485 มีการควบคุมแบบ PID และแรงบิดที่สูง<sup>2</sup>



รูปที่ 2.15: ร้อแก๊ไข

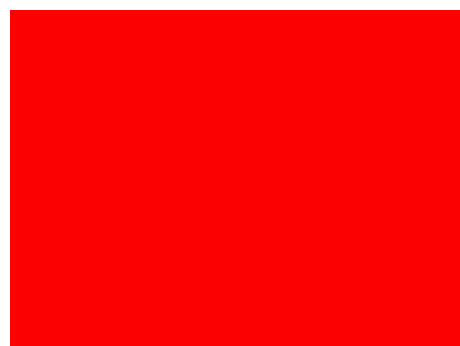
<sup>2</sup>Robot Actuator [http://support.robotis.com/en/product/actuator/dynamixel/ex\_series/ex-106.htm]

### 2.2.3.2 หน่วยประมวลผลควบคุม

ในการควบคุมหุ่นยนต์อิวามานอยด์ให้สามารถทำกิจกรรมต่างๆได้ ด้านนี้ ส่วนที่มีความสำคัญที่ขาดไปไม่ได้ คือ หน่วยประมวลผลระบบควบคุม ถ้าหากไม่มีระบบประมวลผลควบคุมแล้ว อุปกรณ์ต่างๆ ที่ติดตั้งอยู่ภายในตัวของหุ่นยนต์อิวามานอยด์จะไม่สามารถติดต่อสื่อสารกันได้ ซอฟท์แวร์ของหุ่นยนต์ที่พัฒนามาทั้งหมดจะไม่สามารถใช้ได้ ทำให้หุ่นยนต์อิวามานอยด์ไม่สามารถทำงานในสิ่งที่ต้องการได้ การวางแผนระบบควบคุมที่นิยมใช้ในระบบหุ่นยนต์อิวามานอยด์ส่วนใหญ่ จะแบ่งออกเป็น 2 ส่วนด้วยกัน คือส่วนของหน่วยประมวลผลควบคุมระดับสูง และหน่วยประมวลผลควบคุมระดับต่ำ

#### หน่วยประมวลผลควบคุมระดับสูง (High level controller)

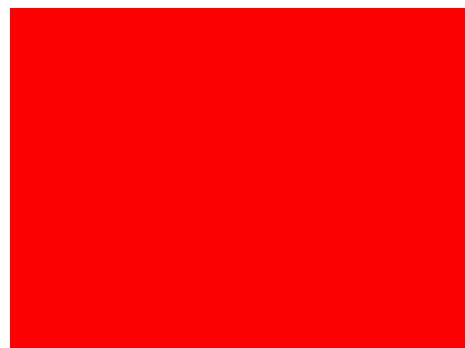
หน่วยประมวลผลควบคุมระดับสูงเป็นส่วนที่ใช้ประมวลผลการทำงานที่มีความซับซ้อนของระบบ เช่น จลนศาสตร์ของหุ่นยนต์ การคำนวณหาเส้นทางการเดิน ในการคำนวณทางคณิตศาสตร์ของระบบเหล่านี้ จำเป็นต้องมีการประมวลผลที่เร็ว และมีประสิทธิภาพ ย้อนไปในสมัยที่มีการพัฒนาหุ่นยนต์อิวามานอยด์ยุคแรกเริ่มเนี้ยน หน่วยประมวลผลควบคุมระดับสูง จะใช้คอมพิวเตอร์เป็นตัวในการประมวลผลการคำนวณ ซึ่งคอมพิวเตอร์สมัยนั้น มีขีดความสามารถให้กับ น้ำหนักมาก และต้องใช้หลังงานสูง ซึ่งต่างจากปัจจุบันนี้ที่มีการพัฒนาของเทคโนโลยีที่ก้าวหน้ามากขึ้น ทำให้คอมพิวเตอร์มีขีดความสามารถเด็กลงเทียบเท่ากับบอร์ดคอนโทรลเลอร์ที่ว่าไป ในที่นี้จะทำการยกตัวอย่างของบอร์ดคอมพิวเตอร์ที่มีวงจรนำร่องในปัจจุบัน และทำการรวมรวมเทียบเคียงประสิทธิภาพ ของบอร์ดคอมพิวเตอร์แต่ละชนิดไว้ดังนี้



รูปที่ 2.16: รอกäge

#### หน่วยประมวลผลควบคุมระดับต่ำ (Low level controller)

หน่วยประมวลผลควบคุมระดับต่ำ เป็นส่วนที่รับคำสั่งบางอย่างมาจากการ หน่วยประมวลผลควบคุมระดับสูง มีประสิทธิภาพในการประมวลผลการคำนวณที่น้อยกว่า เนื่องจากการออกแบบสถาปัตยกรรมภายในระบบไม่เอื้ออำนวยต่อการคำนวณที่มีความซับซ้อน แต่มีความสามารถในการประมวลผลระบบที่เป็นมาตรฐานอย่างแม่นยำ ในด้านการทำหุ่นยนต์อิวามานอยด์นั้nm กจะใช้หน่วยประมวลผลควบคุมระดับต่ำ ในการติดต่อกับอุปกรณ์ต่างๆ บนตัวของหุ่นยนต์อิวามานอยด์โดยตรง เช่น ตัวขับเคลื่อน เชนเชอร์รับค่า หรือไฟแสดงสถานะต่างๆ ของหุ่นยนต์ จากตารางข้างต้นเป็นตารางเปรียบเทียบประสิทธิภาพและความเหมาะสมกับการใช้งาน จะเห็นได้ว่า Nucleo มีประสิทธิภาพมากกว่าในหลายด้านไม่ว่าจะเป็น Core Clock ที่เร็วถึง 100 MHz หรือ GPIO ที่มีมาให้ 50 ช่อง การเชื่อมต่อ อีกทั้งยังมี I2C ซึ่งเป็นรูปแบบที่ใช้ในการติดต่อกับ IMU ที่ต้องใช้ ทั้งนี้ ทั้ง 2 รุ่นที่ได้ทำการเปรียบเทียบมีร่องรับ RS – 485 โดยตรง ซึ่งเป็นรูปแบบการสื่อสารที่จะใช้กับการติดต่อกับตัวขับเคลื่อน



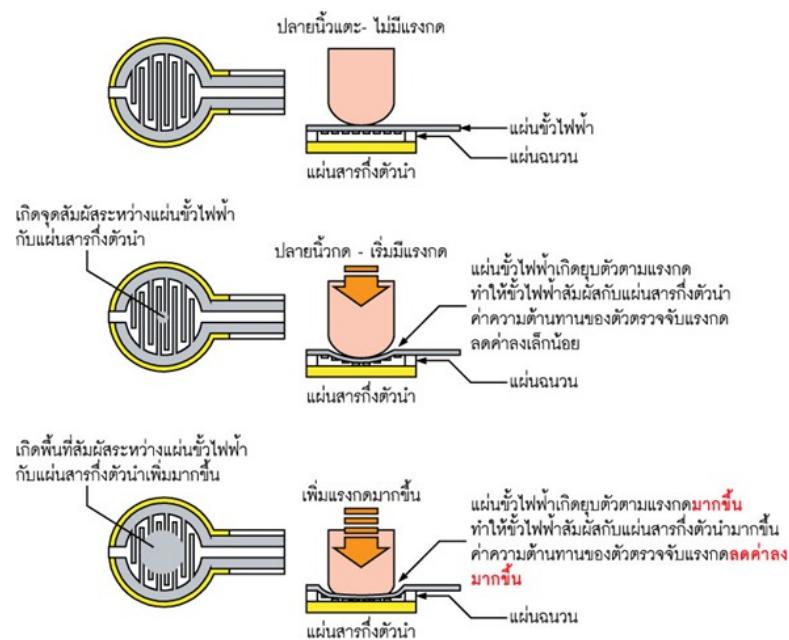
រូបថត 2.17: ទូរការណ៍ឲ្យ

### 2.2.3.3 เช่นเชอร์ตรวจหน้าสัมผัสที่พื้น

เช่นเชอร์ตรวจหน้าสัมผัสที่พื้นเป็นเช่นเชอร์ที่ถูกติดตั้งบริเวณฝ่าเท้า เพื่อตรวจสอบการเดินของหุ่นยนต์ ขีวามนอยด์ว่าขณะนี้มีการสัมผัสดของฝ่าเท้าของหุ่นยนต์กับพื้นหรือไม่ ซึ่งในงานวิจัยนี้ได้ใช้หลักการตัวตรวจจับแรงกดแบบค่าความต้านทานหรือ Force Sensing Resistor (FSR) ที่ใช้เทคโนโลยีฟิล์มโพลีเมอร์แบบหนาโดยที่ เช่นเชอร์สามารถเปลี่ยนแรงที่มากระทำให้อยู่ในรูปของการเปลี่ยนแปลงค่าความต้านทานไฟฟ้า ตัวเช่นเชอร์มีลักษณะเป็นแผ่น มีโครงสร้าง 5 ชั้น โดยสองชั้นนอกสุดเป็นฟิล์มของโพลีเอสเตอร์ ส่วนชั้นถัดเข้ามาเป็นฟิล์มของโลหะที่เป็นตัวนำไฟฟ้า และชั้นในสุดเป็นหมึกที่มีความไวในการตอบสนองต่อแรงภายนอกที่มากระทำ (Pressure sensitive ink) และโครงสร้างทั้ง 5 ชั้น ถูกรูมเข้าด้วยกันด้วยวิธีลามิเนท จึงทำให้เช่นเชอร์วัดแรงนี้มีลักษณะแบบมีความยืดหยุ่นสูง ด้วยเหตุนี้จึงทำให้เช่นเชอร์สามารถโค้งงอได้ง่าย แรงดันไฟฟ้าที่ต่อกคร่อมตัวตรวจจับจะลดลง เมื่อมีแรงกดมากระทำบนแผ่นตรวจจับ มีโครงสร้างของตัวตรวจจับแสดงในรูปที่ 2.18<sup>3</sup>



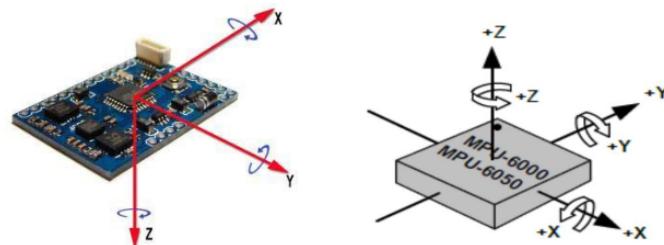
รูปที่ 2.18: ลักษณะโครงสร้างของตัวตรวจจับแรงกด FSR



รูปที่ 2.19: การทำงานของตัวตรวจจับแรงกด FSR

<sup>3</sup>[UNICON] Force sensor with UNICON [<http://doc.inex.co.th/force-sensor-with-unicon/>]

#### 2.2.3.4 เชนเซอร์วัดความเฉื่อย



รูปที่ 2.20: เชนเซอร์วัดความเฉื่อย

Inertial Measurement Unit (IMU) เป็นส่วนประกอบหลักที่ใช้ในการนำร่องเครื่องบิน ยาน-อวกาศ ดาวเทียม เรือ ขีปนาวุธ ซึ่งในตัวของ IMU ประกอบไปด้วยสองส่วนหลักคือ Accelerometers 3 ทิศทาง ใน การรับความเร่งเชิงเส้น และ Gyroscopes 3 ทิศทาง ในการบอกความเร็วเชิงมุม เชนเซอร์ตัวนี้สามารถนำมาใช้ในการหาทิศทางการหมุนของตัวหุ่นยนต์ชีวามโนยได้

ชนเซอร์วัดความเร็ว (Gyroscope)<sup>4</sup> เป็นอุปกรณ์สำหรับการวัดความเร็ว หรือการรักษาการปรับทิศทาง ขึ้นอยู่กับหลักการของการอนุรักษ์โมเมนตัมเชิงมุม ถ้าไม่มีการเคลื่อนที่ อัตราการเปลี่ยนแปลงมุมจะมีค่าเท่ากับศูนย์

ชนเซอร์วัดความเร่ง (Accelerometer)<sup>5</sup> เป็นอุปกรณ์ที่ใช้วัดความเร่ง เชิงเส้น โดยอาศัยการวัดแรงที่กระทำต่อน้ำหนัก ถ้าองที่เกิดจากแรงโน้มถ่วงโลก ซึ่งแรงโน้มถ่วงของโลกจะเป็นเวกเตอร์ซึ่งไปที่แกนกลางโลก เสมอ ตามกฎของนิวตัน

#### 2.2.4 แนวคิดการออกแบบกลไกการเดินของหุ่นยนต์ชีวามโนย

การออกแบบหุ่นยนต์ชีวามโนยด้วยสามารถเดินสองขาได้เหมือนมนุษย์โดยใช้จำนวนของสาอิสระให้เท่ากับมนุษย์นั้นพบว่า มีข้อจำกัดทางด้านการออกแบบอยู่มาก เนื่องมาจากอุปกรณ์ที่ใช้ในการขับเคลื่อนข้อต่อต่างๆ มีอยู่อย่างจำกัด รวมถึงข้อจำกัดทางด้านตัวรับรู้ตัวขับของหุ่นยนต์ ดังนั้นผู้จัดทำจึงออกแบบหุ่นยนต์ให้มีองศาอิสระของข้อต่อ ในขนาดนึงข้าง เท่ากับหกองศาอิสระ ทั้งนี้หุ่นยนต์ยังสามารถเคลื่อนที่ได้ในปริภูมิ และองศาอิสระเพียงพอต่อการใช้งาน

<sup>4</sup>Mechanic gyroscope two-degree of freedom [[https://www.bosch-sensortec.com/bst/products/motion/gyroscope/overview\\_gyroscopesensors](https://www.bosch-sensortec.com/bst/products/motion/gyroscope/overview_gyroscopesensors)]

<sup>5</sup>Accelerometer and Gyroscopes Sensor [<https://www.maximintegrated.com/en/app-notes/index.mvp/id/5830>]

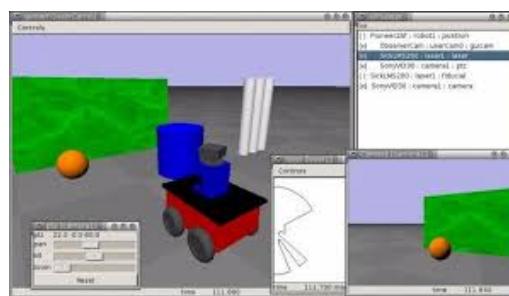
## 2.3 การออกแบบโปรแกรมด้วย ROS

### 2.3.1 ระบบที่ใช้ช่วยในการพัฒนาหุ่นยนต์

Robot Middleware เป็นกรอบการทำงาน(framework) ที่มีความยืดหยุ่นสำหรับการพัฒนาซอฟแวร์ที่ซับซ้อนในการควบคุมของหุ่นยนต์ ตัว Robot Middleware ถูกออกแบบมาให้ใช้ในการจัดการระบบที่มีความยุ่งยาก โดยมีเครื่องมือที่ช่วยติดต่อสื่อสารระหว่างอุปกรณ์ต่างๆของหุ่นยนต์ Robot Middleware ส่วนใหญ่จะใช้การติดต่อสื่อสารผ่านระบบเครือข่ายเน็ตเวิร์ก ทำให้การสื่อสารในระบบพื้นฐานเป็นอิสระต่อกัน และสามารถติดต่อสื่อสารกันกับอุปกรณ์ที่อยู่ภายนอกผ่านเครือข่ายเดียวกันได้

ปัจจุบันมี Robot Middleware ที่ถูกพัฒนาขึ้นมาให้ใช้อยู่หลายตัว เช่น

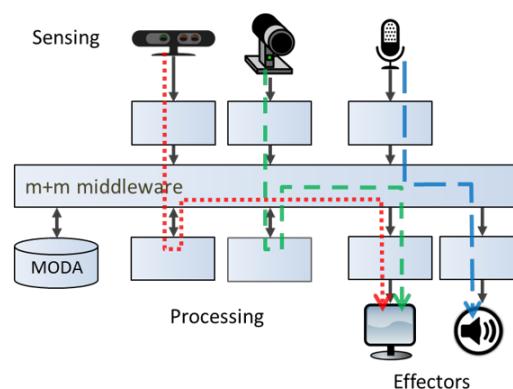
#### Player Project



รูปที่ 2.21: player project middleware

เป็นโปรเจกท์ที่ใช้ในการสร้างซอฟแวร์เพื่อการศึกษาวิจัยที่มีความเกี่ยวข้องกับหุ่นยนต์และระบบเซนเซอร์ภายในประกอบไปด้วยระบบตัวกลาง และระบบจำลองการทำงานของหุ่นยนต์

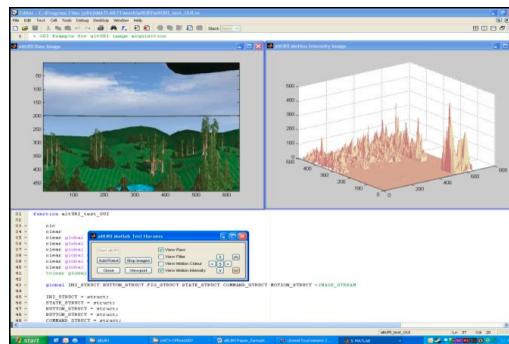
#### YARP



รูปที่ 2.22: yarp middleware

เป็น open source ที่เขียนด้วยภาษา C++ ในการเชื่อมต่อกับเซนเซอร์ หน่วยประมวลผล และตัวขับเคลื่อนของหุ่นยนต์

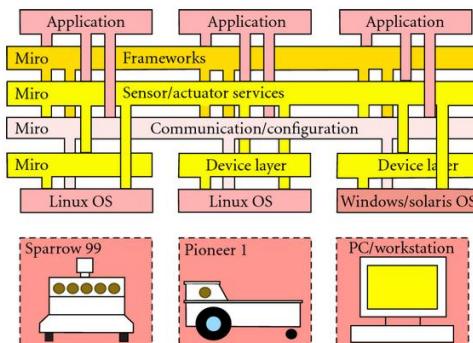
## URBI



รูปที่ 2.23: urbi middleware

เป็น open source สำหรับพัฒนาแอ�플ิเคชันที่เกี่ยวข้องกับหุ่นยนต์หรือระบบที่มีความซับซ้อนใช้ภาษาพื้นฐานเป็นภาษา C++ ติดต่อสื่อสารได้ภายในเครือข่ายเดียวกันเท่านั้น (Local Network)

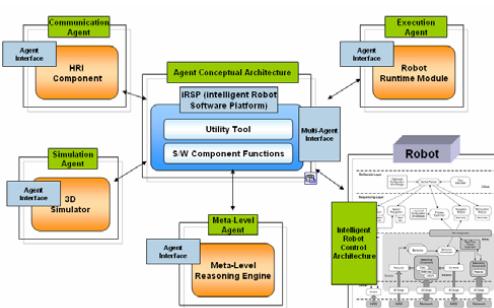
## MIRO



รูปที่ 2.24: miro middleware

เป็นกรอบการทำงานของหุ่นยนต์ที่เคลื่อนที่โดยใช้ในลักษณะเป็น OOP

## OpenRDK

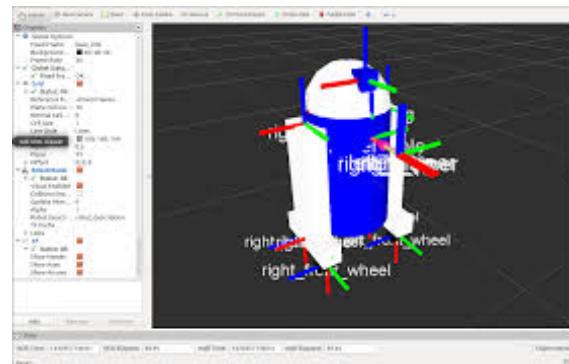


รูปที่ 2.25: openrdk middleware

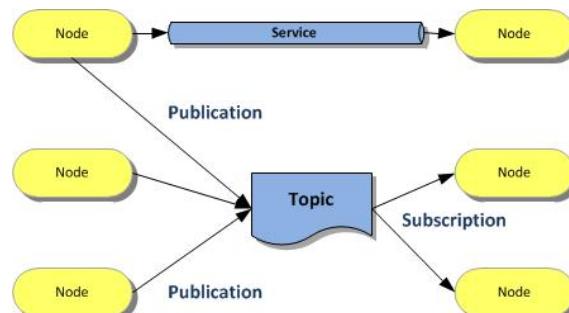
เป็น open source สำหรับพัฒนาระบบที่มีความเป็นอิสระต่อกัน (Modules) สามารถใช้ช่องทางการติดต่อสื่อสารและหน่วยความจำร่วมกันได้

## ROS

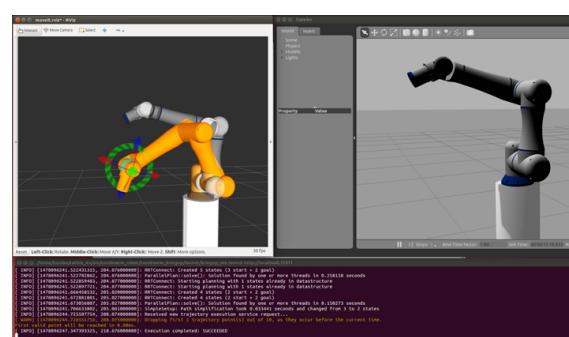
Robot Operating System หรือ ROS ถูกพัฒนาโดยบริษัท Willow Garage, แต่เดิมแล้วคือพัฒนาเพื่อใช้ งานกับหุ่นยนต์ PR2 ในปี 2007 ซึ่งพัฒนาเป็น open source framework สำหรับนักพัฒนาซอฟแวร์ที่เกี่ยวข้อง กับหุ่นยนต์ มีความสามารถในการทำงานแบบ parallel บนคอมพิวเตอร์หลายเครื่องได้ สามารถทำงานได้หลาย OS นอกจากนี้ยังมีคลังที่ครอบเก็บของไว้เป็น libraries อีกด้วย การใช้ ROS จะช่วยทำให้เราสามารถพัฒนาหุ่นยนต์ได้อย่างรวดเร็วมากขึ้น ประหยัดเวลา ประหยัดทรัพยากร



รูปที่ 2.26: ROS middleware Rviz



รูปที่ 2.27: ROS algitecture



รูปที่ 2.28: ROS Moveit

### 2.3.2 ระบบที่ใช้ในการจำลองการทำงานของหุ่นยนต์

โปรแกรมจำลองการทำงานของหุ่นยนต์นั้นเป็นเครื่องมือที่สำคัญสำหรับนักหุ่นยนต์ การใช้โปรแกรมจำลองนั้นจะช่วยเพิ่มประสิทธิภาพในการทำงานหลายอย่าง เช่น ให้รู้ว่าหุ่นยนต์ที่ออกแบบนั้นสามารถทำงานได้อย่างที่ต้องการหรือไม่ กระบวนการคิดถูกต้องหรือไม่ โปรแกรมจำลองระบบส่วนใหญ่จะคำนวณพลวัตของหุ่นยนต์โดยใช้เครื่องมือคำนวณ open dynamics engine (ODE)

#### USARSim



รูปที่ 2.29: ผลลัพธ์จากการใช้โปรแกรม USARSim

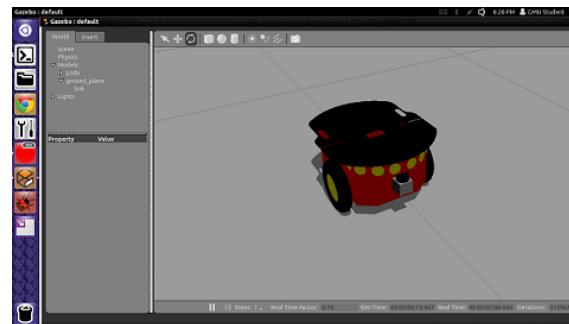
USARSim เป็นโอเพนซอร์ซและเหมาะสมสำหรับทำหุ่นยนต์ประเภทกีฬาในชากเมือง โดยมีฐานการพัฒนามาจาก Unreal Tournament game engine ภายใต้โปรแกรมมีเครื่องมือสำหรับการทำงานวิจัย มีเซนเซอร์ของหุ่นยนต์ที่หลากหลาย เช่น เซนเซอร์รับภาพ หรือเซนเซอร์ตรวจความเคลื่อนไหว

#### MuRoSimF

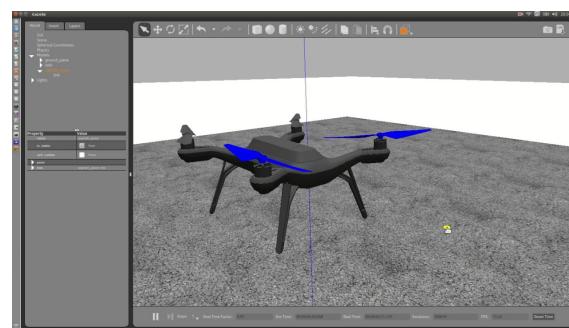


รูปที่ 2.30: ผลลัพธ์จากการใช้โปรแกรม MuRoSimF

MuRoSimF ย่อมาจากคำว่า Multi-Robot Simulation Framework เป็นเครื่องมือที่ช่วยทำระบบจำลองจาก Darmstadt University โปรแกรมระบบจำลองนี้มีการใช้งานที่ง่าย เหมาะสำหรับหุ่นยนต์หลายประเภท เช่น หุ่นยนต์เคลื่อนที่ด้วยล้อ หุ่นยนต์สองขา หรือหุ่นยนต์หลายขา สามารถคำนวณพลวัตร และการขัดกันของก้านต่อต่างๆได้



รูปที่ 2.31: Mobile robot with gazebo



รูปที่ 2.32: Quadrotor with gazebo

### Gazebo

Gazebo เป็นโปรแกรมจำลองการทำงานของหุ่นยนต์ ที่มีความสามารถในการคำนวณการเดินและการเคลื่อนที่ของหุ่นยนต์ที่สลับซับซ้อนได้ สามารถเห็นภาพกราฟฟิกของหุ่นยนต์ขณะทำงาน โดยผู้ใช้สามารถกำหนดค่าตัวแปรทางพิสิกส์ต่าง ๆ ได้ เช่น น้ำหนัก ค่าความเร็ว แรงเสียดทานของข้อต่อ ทำให้การออกแบบหุ่นยนต์หรือทดลองโปรแกรมได้เหมือนกับโลกจริง มีแสง มีเงา และ พื้นผิวของวัสดุ และที่พิเศษคือสามารถสังเคราะห์ค่าของเซนเซอร์ เช่นเซอร์พร้อมสัญญาณรบกวน ค่าระยะทาง แรงบิด และอื่นๆ คำนวณผลศาสตร์ของหุ่นยนต์โดยใช้ตัวคำนวณทางพิสิกส์เป็น Bullet หรือ Simbody ใน การจำลองหุ่นยนต์ในโปรแกรมนี้จำเป็นต้องได้รับไฟล์ข้อมูลของหุ่นยนต์มาก่อนซึ่งอยู่ในรูปแบบของ URDF ซึ่ง URDF คือ ประเภทของไฟล์ที่บ่งบอกถึงความสัมพันธ์ของข้อต่อและก้านต่อแต่ละชิ้นในตัวหุ่นยนต์ มีความสามารถในการอธิบายถึงกลศาสตร์และการเคลื่อนที่ของหุ่นยนต์ รวมถึงตรวจสอบการขัดกันของก้านต่อในหุ่นยนต์ได้ ภายในไฟล์นี้จะประกอบไปด้วย

**Link :** คือก้านต่อของหุ่นยนต์ซึ่งภายในจะสามารถบอกขนาด รูปร่าง สี และสามารถ import 3d mesh เข้ามาได้ด้วย อีกทั้งยังสามารถใส่รายละเอียดของการเคลื่อนที่ ของก้านต่อได้ เช่น inertial matrix และ collision properties

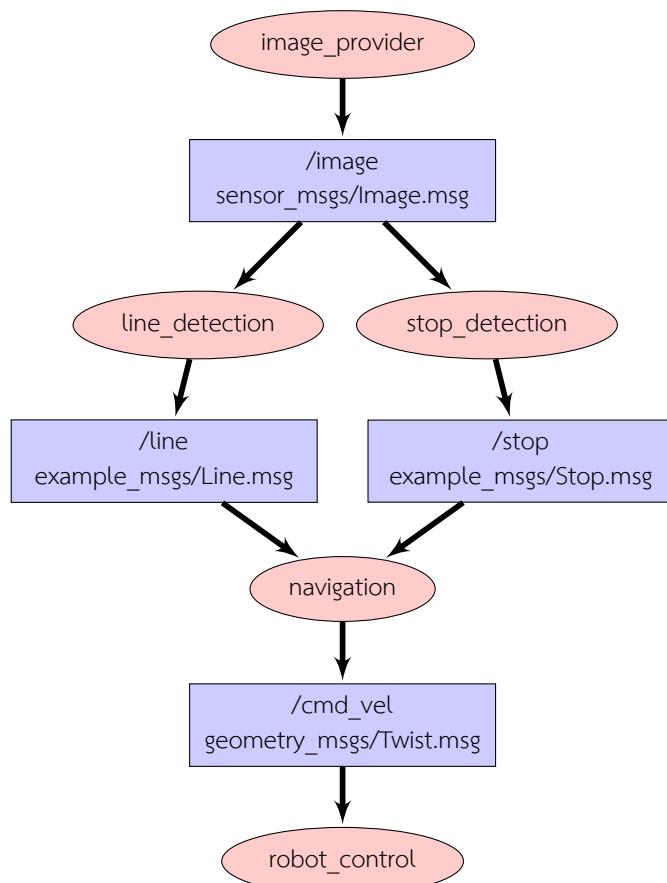
**Joint :** คือข้อต่อของหุ่นยนต์สามารถกำหนดกลศาสตร์และการเคลื่อนที่ได้ เช่น Joint limits ของข้อต่อที่กำลังหมุนและความเร็วการหมุน ซึ่งข้อต่อมีหลายแบบที่สามารถกำหนดได้ เช่น ข้อต่อแบบหมุน, ข้อต่อแบบเลื่อน, ข้อต่อแบบบิดติด, ข้อต่อแบบต่อเนื่อง

### 2.3.3 Robot Operating System

Robot Operating System หรือ ROS ถูกพัฒนาโดยบริษัท Willow Garage, แต่เดิมแล้วเป็นเครื่องมือเพื่อใช้งานกับหุ่นยนต์ PR2 ในปี 2007 ซึ่งพัฒนาเป็น open source framework สำหรับนักพัฒนาซอฟแวร์ที่เกี่ยวข้องกับหุ่นยนต์ มีความสามารถในการทำงานแบบ parallel บนคอมพิวเตอร์หลายเครื่องได้ สามารถทำงานได้หลาย OS แต่ที่ซัพพอร์ทจริงๆ ก็คือ Ubuntu และ Debian นอกจากนี้ยังมีคลังที่ค่อยเก็บซอฟแวร์ต่างๆไว้เป็น libraries อีกด้วย การใช้ ROS จะช่วยทำให้เราสามารถพัฒนาหุ่นยนต์ได้อย่างรวดเร็วมากขึ้น ประหยัดเวลา ประหยัดทรัพยากรในส่วนนี้จะกล่าวถึง ROS คร่าวๆ

#### Nodes

Node เป็นหนึ่งหน่วยประมวลผลในระบบ ROS, Node สามารถที่จะส่งข้อมูลหาโนนด์อื่นๆได้ ผ่าน Topics หรือ Services ในทางปฏิบัติแล้วโนนด์เป็นตัวประมวลผลอยู่ๆ ที่ค่อยทำงานที่เฉพาะ ยกตัวอย่าง เช่น โนนด์ตัวแรกเชื่อมต่อกับกล้อง เพื่อที่จะนำภาพจากกล้องออกมานะ โนนด์ตัวที่สองใช้ในการหาลูกบล็อกที่อยู่ในภาพที่ได้มาจากโนนด์ตัวแรก และโนนด์ตัวที่สามใช้ในการคำนวนหาตำแหน่งของลูกบล็อกที่อยู่บนโลกจริงๆ จากตำแหน่งของลูกบล็อกที่ได้มาจากการคำนวนที่สอง ดังนั้นจะเห็นว่าแต่ละโนนด์จะทำงานเฉพาะของตัวเอง ซึ่งสามารถนำมาร่วมกันได้ การเขียนเป็นแบบโนนด์จะช่วยทำให้เราสามารถที่จะนำโปรแกรมกลับมา แก้ไขปรับปรุงให้ใช้ใหม่ได้ง่าย ในกรณีที่จะนำไปทำงานอย่างอื่น ยกตัวอย่าง เช่น โนนด์ที่เอาภาพจากกล้องออกมานะ อาจจะมีโนนด์อีกตัว ทำงานที่ในการหาโกล์ดเป้าหมาย และหาทิศทางการเคลื่อนที่ของหุ่นยนต์ได้ ดังนั้นการพัฒนาโนนด์เป็นส่วนย่อยๆเล็กๆ ที่เพื่อที่จะทำให้การแก้ไขหรือปรับปรุงได้ง่าย



รูปที่ 2.33: ตัวอย่างสถาปัตยกรรมของ ROS

จากตัวอย่างสถาปัตยกรรมของ ROS ดังรูปที่ 2.33 นั้นสามารถอธิบายได้ว่า หุ่นยนต์เคลื่อนที่ด้วยล้อมีภารกิจคือ เคลื่อนที่ตามเส้นไปเรื่อยๆจนกว่าจะเจอเครื่องหมายหยุด Node คือตัวที่แสดงด้วยรูปวงรี ข้างในเป็นชื่อ Hind ส่วน Topic จะแสดงด้วยรูปสี่เหลี่ยม ซึ่งข้างในเป็นชื่อของ Topic และชนิดของ Message ที่ใช้ในการส่งข้อมูล มาดูกันก่อนอื่น ภาพถูกส่งมาจากการล้อ แลกกับ Hind แสดงตัวในการดูเส้น และเครื่องหมายหยุด จากการที่ได้มา เมื่อ Hind ได้ข้อมูลแล้วก็นำมาประมวลผลการเดินของหุ่นยนต์โดยส่งไปยัง node navigation และ Hind นี้ ก็จะทำหน้าที่คำนวณความเร็วและทิศทางของหุ่นยนต์ ส่งไปยัง node robot\_control ซึ่งเป็นตัวสั่งการมอเตอร์ของหุ่นยนต์อีกด้วย

Twist.msg			Stop.msg
geometry_msgs/Vector3	linear		uint8 RED = 0
geometry_msgs/Vector3	angular		uint8 GREEN = 1
(ก) Message Twist		(ข) Message Stop	

ตารางที่ 2.2: ตัวอย่างชื่อและข้อมูลของ Message

ตัวอย่างของ Message สองอันนี้ Twist message (รูปที่ 2.2ก) คือ message ที่เอาไว้บอกความเร็วเชิงเส้น และความเร็วเชิงมุม ซึ่ง ROS มี message ชนิดนี้ให้อยู่แล้ว ส่วน Stop message (รูปที่ 2.2ข) คือ message ที่เอาไว้บอกระยะทางและสีของป้าย Stop ซึ่ง message นี้ถูกสร้างขึ้นมาใหม่เพื่อใช้กับงานนี้โดยเฉพาะ

### Topics and Messages

Messages เป็นตัวหลักสำคัญในการติดต่อสื่อสารกันระหว่าง Hind ใน ROS โดยที่ message จะถูกส่งผ่านไปยัง topic เสมอ แต่ละ Node สามารถที่จะ subscribe หรือ publish ไปที่ topic นี้ได้ การเชื่อมต่อกันระหว่าง Node นั้นสามารถส่งอยู่ภายในเครื่องคอมพิวเตอร์เครื่องเดียวกัน หรือเครื่องอื่นได้ที่อยู่ใน network เดียวกัน โดยจะติดต่อสื่อสารโดยใช้ TCP/IP การใช้คอมพิวเตอร์หลายเครื่องก็จะช่วยให้การประมวลผลมีประสิทธิภาพมากยิ่งขึ้น นอกจากนั้นยังสามารถที่จะแบ่งหน้าที่การทำงานออกจากกันได้ เราสามารถที่จะสร้าง Topic หรือ Message ขึ้นมาเองได้ หากต้องการใช้งานที่เฉพาะทาง

### roscore

roscore เป็นส่วนกลางในการรันระบบทั้งหมด เราจะเรียกว่า rosmaster ซึ่งมีหน้าที่ในการจัดการ topics ทั้งหมด ที่ต้องการจะเชื่อมต่อกันไม่ว่าจะเป็นการ publish หรือ subscribe แต่ rosmaster จะเป็นแค่ตัวจัดการเท่านั้นไม่ได้เป็นตัวที่เก็บ message ต่างๆที่ส่งไปมา ดังนั้น rosmaster จะไม่ทำให้เกิดคอกขด เวลา rans ระบบ ในกระบวนการที่คือ subscribe node จะถาม rosmaster ว่ามี topic ที่ต้องการรับข้อมูลใหม่ ส่วนตัว master ที่เก็บค่า topic message เอาไว้ ก็จะส่งไปยัง subscribe node ถ้าหากมีข้อมูลตามที่ร้องขอมา และ rosmaster ก็จะจำไว้ว่ามี node ไหนเชื่อมต่อกับ node ไหนบ้าง

rosparameter server เป็นตัวในการเก็บค่าต่างๆที่เป็น global key-value ซึ่งช่วยให้ node ทุกตัวสามารถใช้ข้อมูลตัวเดียวกันได้ สามารถปรับเปลี่ยนระหว่างการทำงานอยู่ได้ โดยใช้ rqt plugin ซึ่งจะกล่าวในส่วนถัดไป

roslog เป็นตัวที่ใช้สำหรับ logging ข้อมูลต่างๆ ซึ่งจะถูก publish ออกมายัง topic /rosout ซึ่งเราสามารถที่จะเขียนโปรแกรม subscribe จากตัว topic นี้ไปเก็บเป็นไฟล์ได้

## Services

Services หรืออีกชื่อนึงคือ remote procedure calls (RPC) ความหมายคือเป็นการส่ง messages แบบที่ไม่ได้เจาะจงว่าจะส่งไปที่ไหน เมื่อเราเรียก service และระบบจะรอนกว่าจะมีการตอบกลับ เราจะเรียกกระบวนการนี้ว่า request และ response message Node ที่ค่อยทำงานเมื่อมีการเรียกใช้ service จะเรียกว่า service server และ node ที่เรียก service จะเรียกว่า service client การใช้งาน service เหมาะสำหรับงานที่ต้องการความรวดเร็ว (fast task) แต่ไม่ควรใช้กับระบบที่ต้องใช้เวลานาน เพราะระบบจะหยุดไม่ยอมทำต่อ ต้องรอให้ service ทำงานเสร็จก่อน สำหรับงานที่ต้องใช้เวลาในการคำนวณจะไปใช้ action แทน จะกล่าวในส่วนถัดไป

## Actions

Actions จะใช้กับการทำงาน การประมวลผลที่ต้องใช้เวลาในการทำงาน หรือที่เรียกว่า asynchronously task ในแต่ละ action จะมี message อよู่ 3 ชนิด คือ goal, feedback และ result Node ที่เป็นตัวรับและรอให้ node อื่นมาเรียก จะเรียกว่า action server ส่วน node ที่เรียกการทำงาน action จะเรียกว่า action client การใช้งาน action จะเริ่มจาก action client จะส่ง message goal ไปยัง action server และ action server จะพยายามทำงานตาม goal ที่ได้รับมา ในระหว่างที่ action client ก็จะทำงานของตัวเองต่อไป แต่จะได้รับ feedback จาก action server อยู่ตลอดเวลา และเมื่อถึง goal ที่กำหนดแล้ว server จะแจ้งมาทาง result message

## Code Organization

ส่วนที่เล็กที่สุดของการจัดการซอฟแวร์ใน ROS ก็คือ package ภายใน package จะมีไฟล์ที่ชื่อว่า package.xml ซึ่งไฟล์นี้จะทำหน้าที่ในการ อธิบายและบอกข้อมูลต่างๆที่เกี่ยวกับ package นี้ ยกตัวอย่างเช่น ชื่อของ package, ชื่อของผู้เขียน, ลิขสิทธิ์ และ dependencies ที่ต้องใช้กับ package นี้ นอกจากนี้ยังสามารถใส่ข้อมูลอื่นๆเกี่ยวกับ node ลงไปเพิ่มเติมได้



```

1 <package>
2   <name>example_package</name>
3   <version>1.0.0</version>
4   <description>Short example for a package.xml.</description>
5   <maintainer email="ex@example.org">Jane Doe</maintainer>
6   <license>BSD</license>
7   <buildtool_depend>catkin</buildtool_depend>
8   <build_depend>example_2</build_depend>
9   <run_depend>std_msgs</run_depend>
10 </package>

```

รูปที่ 2.34: ตัวอย่างไฟล์ package.xml

แต่ละ tags ใช้ในการบอกข้อมูลของ package นี้ ใครเป็นเจ้าของ ใครเป็นคนเขียน รวมไปถึง dependencies ที่จำเป็นต้องใช้ของ package นี้ด้วย ดังรูปที่ 2.34

## Code Distribution

การที่จะนำ Nodes กลับมาใช้ใหม่หรือเอาอกมาแชร์ได้นั้น จะต้องมีการทำเอกสารของ Packages นั้นๆ ด้วย โดยปกติแล้วจะถูกนำไปเก็บไว้ที่ GitHub และ package dependencies จะบอกไว้ในไฟล์ package.xml เรียบร้อยแล้ว เพื่อให้ง่ายต่อการนำไปติดตั้ง หากผู้ที่นำไปใช้พัฒนาต่อหรือแก้ไขข้อผิดพลาดก็สามารถที่จะช่วยกันได้ โดยการ Pull request หรือ Report issues ได้

### ROS Packages ที่ใช้ในงานวิจัย

Package คือพื้นฐานของ ROS, แอพพลิเคชันทั้งหมดใน ROS จะพัฒนาโดยมี package เป็นรากฐาน ใน package นั้นจะเก็บพวกไฟล์ configuration ไปจนถึงไฟล์ launch ที่สามารถไปรัน package หรือ node อื่นๆ ได้ ตอนนี้ ROS มี packages มากกว่า 5000 packages แล้ว

Metapackage เป็นการรวมกันของ packages ที่ทำหน้าที่คล้ายๆ กันหลายๆ ตัวมารวมไว้ที่เดียวเพื่อจะได้ใช้งานง่าย ตัวอย่าง Navigation metapackage ประกอบไปด้วย 10 packages เช่น [AMCL(partical filter), DWA, EKF(extended kalman filter) และ map\_server] ซึ่งหากติดตั้ง metapackage ตัวนี้ก็จะได้มาหมดเลย

ในส่วนนี้จะอธิบายคร่าวๆ ถึง ROS standard packages ที่จะนำมาใช้ในงานวิจัยครั้งนี้

**rosbag** rosbag เป็นแพกเกจที่สามารถบันทึก message ที่ส่งหากันในระหว่างที่ ROS กำลังทำงานได้ไฟล์ที่บันทึกจะเรียกว่า rosbag ประโยชน์ของมันคือเราสามารถเอาเข้ามาใช้ในการตรวจสอบ หรือนำมาเล่นซ้ำได้อีกทั้งยังง่ายต่อการค้นหาข้อมูลพลาดอีกด้วย

**tf2** tf2 เป็นแพกเกจที่สามารถติดตามการเปลี่ยนแปลงของ Coordinate frame เราสามารถใช้ในการหาความสัมพันธ์ระหว่าง frame ได้ ยกตัวอย่างเช่นหากเราต้องการหาตำแหน่งของ foot เทียบกับ pelvis ก็สามารถใช้ tf2 หาได้

**robot\_state\_publisher** robot\_state\_publisher แพกเกจที่ subscribe JointState message เพื่อที่จะนำตำแหน่งของของข้อต่อ และแปลงให้อยู่ในรูปข้อมูลของ tf2, tf2 สามารถเรียกจาก Node ใดๆก็ได้เพื่อที่จะหา Coordinate frame ที่ต้องการได้

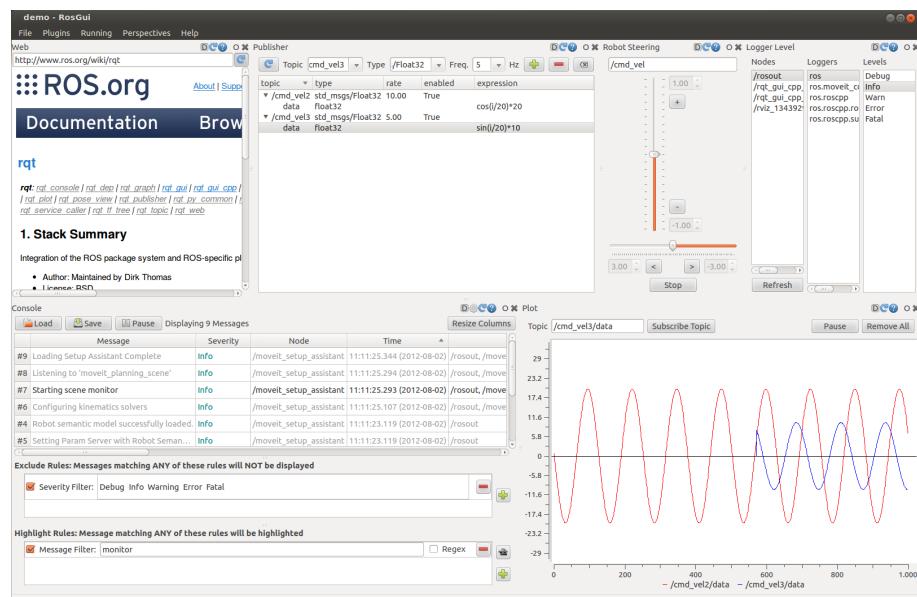
**URDF** Unified Robot Description Format (URDF) เป็นไฟล์ XML ที่เอาไว้อธิบายลักษณะของหุ่นยนต์ ใน ROS มีแพกเกจที่ใช้สำหรับการอ่านไฟล์ คือ urdf\_parser แต่ไฟล์นี้มีการใช้งานโดย tf2 เช่นกัน

**xacro** xacro เป็นไฟล์ XML เช่นเดียวกับ URDF โดยไฟล์ xacro นี้มีประโยชน์มากในการใช้งานใน ROS เพราะว่าทำให้การเขียนไฟล์ URDF ง่ายขึ้น เพราะสามารถทำเป็นมาโครได้ สามารถปรับแต่งค่าตัวแปรต่างๆ ได้ง่ายขึ้น

## Visualization

จุดแข็งสำคัญของ ROS อยู่ที่ว่ามีเครื่องมือที่ช่วยในการแสดงผล Visualization ได้ นอกจากนี้จากระบบ publisher-subscriber การใช้ Visualization tools นี้จะช่วยให้การทำงานง่ายขึ้นและประหยัดเวลา many ขึ้น ในการนำข้อมูลต่างๆจากหุ่นยนต์ออกมาระบบสามารถที่จะ subscribe จาก topic ที่มีการใช้งานอยู่แล้วมาแสดงผลได้ทันที ใน ROS มีเครื่องมือสำคัญอยู่ 2 ตัวที่ใช้สำหรับการ Visualization ซึ่งสามารถที่จะปรับแต่งให้กลายเป็นเวอร์ชั่นของเราเองได้

**rqt** rqt เป็น UI ที่มีฐานมาจาก QT ซึ่งมาพร้อมกับการเชื่อมต่อ ROS เป็นส่วนเสริมในรูปแบบของ QWidget เราสามารถที่จะแสดงผลหลายๆ widgets ได้ภายในเวลาเดียวกัน สามารถที่จะย่อขยาย เปลี่ยนตำแหน่ง ลักษณะได้ การเชื่อมต่อกับ ROS นั้นสามารถนำการแสดงผลแบบ 2D ไปแสดงได้ดังรูปที่ 2.35 เป็นการแสดงภาพของกราฟที่ได้รับข้อมูลมาจาก topic หลายตัว และสามารถที่จะปรับแต่งค่าและ publish ออกไปได้ ด้วยการเขียนโปรแกรมเข้าไป ซึ่งจะเป็นประโยชน์อย่างมากเวลาที่ใช้ในการปรับจูนพารามิเตอร์ต่างๆ เพราะว่าเราสามารถที่จะเปลี่ยนค่าได้ทันที ไม่ต้องรันโปรแกรมใหม่ ในรูปที่ 2.35 เป็นการนำ rqt มาใช้เป็น GUI ให้ผู้ใช้สามารถใช้งานได้ง่ายและสามารถที่จะปรับแต่งพารามิเตอร์ต่างๆได้เรียลไทม์

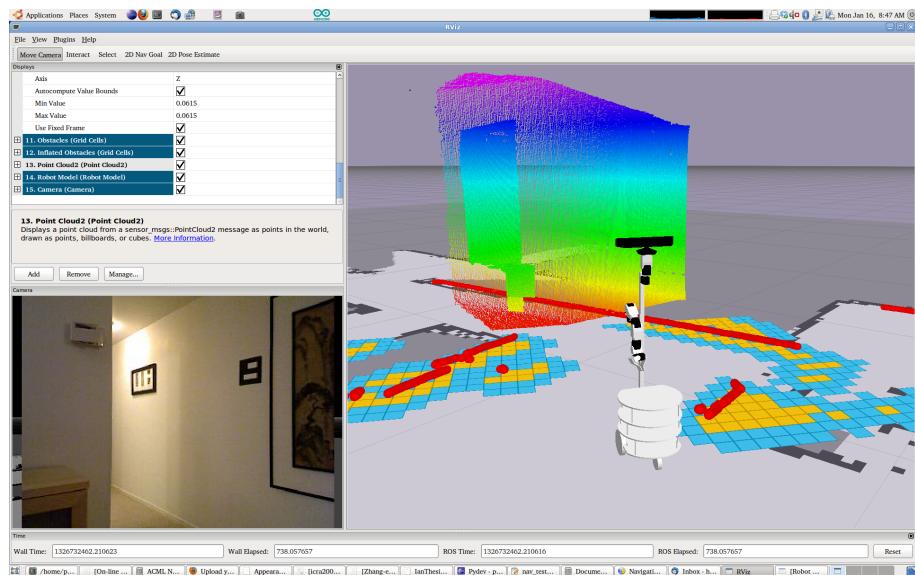


รูปที่ 2.35: ตัวอย่างการแสดงผลใน rqt

**RViz** RViz เป็น 3D visualization ของสถานะต่างๆของหุ่นยนต์และสภาพแวดล้อม โดยใช้ไฟล์ URDF เป็นมาตรฐานการแสดงถึงหุ่นยนต์ ซึ่งสามารถที่จะแสดงตำแหน่งปัจจุบันของข้อต่อต่างๆในหุ่นยนต์ได้ สามารถที่จะแสดงค่าเซ็นเซอร์เป็น marker ได้ การใช้งานจะเป็นเหมือนการบอกริกัดเฟรม ลักษณะการแสดงผลใน RViz มีหลากหลายรูปแบบไม่ว่าจะเป็น camera images, depth clouds, laser scans หรือ point clouds อย่างไร ก็ตามการแสดงผลใน RViz นั้นจะไม่ได้คำนึงถึงแรงที่เข้ามากระทำกับตัวของหุ่นยนต์ แต่ถ้าเป็นการเคลื่อนที่ ที่มีพิกัดเฟรมแล้วสามารถนำมาแสดงได้ ดังรูปที่ 2.36 เป็นเคสของหุ่นยนต์เคลื่อนที่ด้วยล้อ และทำแผนที่ด้วยข้อมูลความลึกที่ได้มาจากการ Kinect

## Simulation

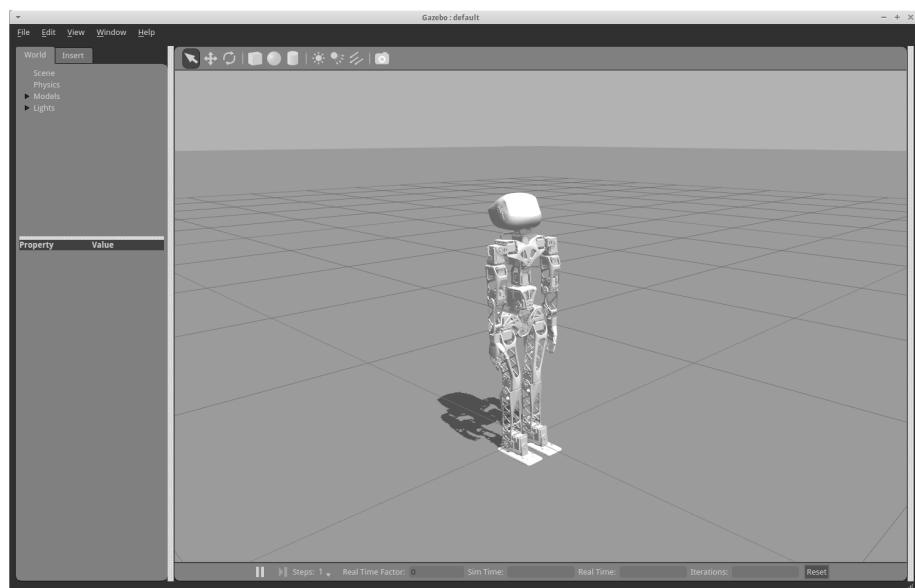
Simulation เป็นส่วนที่สำคัญมากสำหรับการพัฒนาโปรแกรมของหุ่นยนต์ เพราะว่าเราสามารถที่จะสร้างโปรแกรมและทดสอบได้โดยไม่จำเป็นต้องมี hardware ซึ่งในส่วนนี้จะลดความเสียหายจากบักหรือโปรแกรมผิดพลาด ที่อาจจะเกิดขึ้นกับหุ่นยนต์ของเรา การจำลองจะช่วยลดเวลาในการพัฒนาลงได้ ระบบจำลองปัจจุบันมี



รูปที่ 2.36: ตัวอย่างการแสดงผลใน RViz

มากมายหลายตัวแต่ ตัวที่ได้รับคำแนะนำมากที่สุดคือ Gazebo เพราะว่า gazebo สามารถที่จะเชื่อมต่อกับ ROS ได้โดยตรง และนักพัฒนาส่วนใหญ่ใช้ gazebo

การจะใช้ gazebo ได้นั้นเราจะต้องใช้ไฟล์ URDF ซึ่งเป็นไฟล์ที่เอาไว้แสดงหุ่นยนต์ในระบบจำลอง และสามารถที่จะคำนวณหา collision ให้เราได้อีกด้วย



รูปที่ 2.37: ตัวอย่างหุ่นยนต์อิฐมนต์ Poppy

## 2.4 การออกแบบระบบพื้นฐาน

### 2.4.1 ความแตกต่างของ Operating Systems

เป็นที่รู้กันโดยทั่วไปว่า hardware และ software ของคอมพิวเตอร์นั้นถูกจัดการโดยโปรแกรมในคอมพิวเตอร์ ซึ่ง operating system (OS) งานพื้นฐานที่ OS ทำก็เช่น การควบคุมและจองหน่วยความจำ การจัดลำดับความสำคัญของระบบ โดยดูแลควบคุมอุปกรณ์ต่างๆ ที่เชื่อมต่อเข้ากับคอมพิวเตอร์ การเชื่อมต่อระบบเน็ตเวิร์ค การจัดการไฟล์ข้อมูล อีกทั้งยังรวมไปถึงการให้บริการต่างๆ เช่น การจัดการกระบวนการประมวลผล จัดการไฟล์ ของระบบต่างๆ ระบบป้องกัน อื่นๆ

ปัจจุบันมี OS อยู่หลายตัวเช่น Windows, Mac OS X, UNIX, Solaris BS3000, MS-Dos และอื่นๆ ซึ่งทั้งหมดนี้เป็นส่วนหนึ่งของระบบคอมพิวเตอร์ที่จะคอยช่วยจัดการและควบคุมดูแลการทำงานต่างๆ ของคอมพิวเตอร์ ระบบคอมพิวเตอร์นั้นอาจจะอยู่ในรูปแบบอื่นๆ เช่น workstation, server, personal computer, smartphone, navigation device หรือแม้กระทั่งระบบที่มีความคลาดในตัวมันเอง เช่น ที่นั่นยนต์ และ OS นั้นจะสามารถทำงานบนชาร์ทแวร์อุปกรณ์ใดๆ ก็ได้

### 2.4.2 ข้อแตกต่างระหว่าง Open platform กับ Non-open platform

ที่นั่นยนต์ Open platform คือ การออกแบบระบบพื้นฐานของที่นั่นยนต์ที่เปิดให้ผู้ที่ต้องการศึกษาหรือผู้ใช้ทั่วไปสามารถเข้าถึงข้อมูลต่างๆ ที่เกี่ยวข้องกับที่นั่นยนต์นั้นๆ ได้ ผู้ใช้สามารถที่จะนำข้อมูลเหล่านั้นมาแก้ไข ปรับปรุง แต่งเติม หรือเรียนรู้และพัฒนาตามได้ด้วยตนเอง ซึ่งข้อมูลที่กล่าวมานั้นสามารถได้จากเว็บไซต์ของผู้พัฒนาที่นั่นยนต์ ปัจจุบันมีที่นั่นยนต์ที่มีความสามารถอยู่ที่เป็นเปิดให้เข้าถึงหลายรูปแบบแตกต่างกันไป

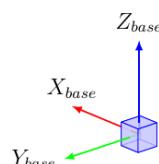
ส่วนที่นั่นยนต์ Non-open source platform คือที่นั่นยนต์ที่สร้างมาเฉพาะเจาะจงสำหรับการวิจัย การสำรวจ หรือการแข่งขันโดยเฉพาะ ไม่เปิดให้บุคคลภายนอกเข้าศึกษาหรือแก้ไขปรับปรุง ซึ่งทำให้ที่นั่นยนต์ประเภทนี้ไม่เหมาะสมสำหรับผู้วิจัยที่จะเรียนรู้และศึกษาด้วยตนเอง เพราะมีขั้นตอนใหญ่ ใช้ทรัพยากร่มาก และการออกแบบมีความซับซ้อน เรียนรู้ยากกว่าที่นั่นยนต์แบบ Open platform

### 2.4.3 มาตรฐานหน่วยวัดและการนองพิกัด

การใช้หน่วยวัดที่ไม่ตรงกันอาจจะทำให้เกิดปัญหาขึ้นได้ เนื่องจากเป็นแพลตฟอร์มนั้นจะมีบุคคลอื่นช่วยกันพัฒนาหลายคน จึงควรที่จะมีมาตรฐานในการวัดและการกำหนดพิกัดต่างๆ ที่ตรงกันเพื่อให้เกิดความชัดเจนในการทำความเข้าใจ

**หน่วยวัด** การวัดนั้นใช้มาตรฐานการวัดเป็น SI Units ซึ่งมาตรฐานนี้ใช้กันอย่างแพร่หลายและเป็นสากล โดยหน่วยการวัดนี้ได้รับการยืนยันจาก Bureau International des Poids et Mesures ตามตารางที่ 2.3

**พิกัดเฟรม** การบอกทิศทางการหมุนนั้นใช้หลักตามกฎมือขวา โดยการตั้งแกนนั้นหากเทียบกับมือแล้ว X ไปข้างหน้า Y ไปทางซ้าย Z ผุ้ขึ้น ตามภาพที่ 2.38



รูปที่ 2.38: การตั้งแกนตามกฎมือขวา

ปริมาณ(Quantity)	หน่วยวัด(Unit)	สัญลักษณ์(Symbol)
ความยาว(Length)	เมตร(metre)	<i>m</i>
มวล(Mass)	กิโลกรัม(kilogram)	<i>kg</i>
เวลา(Time)	วินาที(second)	<i>s</i>
กระแสไฟฟ้า(Electric Current)	แอมเปอร์(ampere)	<i>A</i>
มุม(Angle)	เรเดียน(radian), องศา(degree)	<i>rad, deg</i>
ความถี่(Frequency)	เฮิร์ต(Hertz)	<i>Hz</i>
แรง(Force)	นิวตัน(Newton)	<i>N</i>
กำลัง(Power)	วัตต์(Watt)	<i>P</i>
แรงดันไฟฟ้า(Voltage)	โวลต์(Volt)	<i>V</i>
อุณหภูมิ(Temperature)	เซลเซียส(Celsius)	<i>C</i>

ตารางที่ 2.3: ตารางแสดงหน่วยวัดมาตรฐาน

#### 2.4.4 Robot Operating System

ROS เป็นกรอบการทำงานที่ได้รับความนิยมและมีประสิทธิภาพในการทำงานมากที่สุดในปัจจุบัน เนื่องจาก ROS ได้รวบรวมซอฟแวร์เครื่องมือที่หลากหลายเอาไว้เป็นหมวดหมู่ เช่น การเข้ามือถือกับฮาร์ดแวร์ การสร้างระบบควบคุมให้กับอุปกรณ์ต่างๆ อีกทั้งสามารถที่จะเขียนโปรแกรมแล้วนำกลับมาใช้ใหม่ได้ ภายในระบบมีกระบวนการรับส่งข้อมูลต่างๆ เป็นของตัวเอง ทำให้ช่วยลดความซับซ้อนและเพิ่มประสิทธิภาพในการทำงานกับแพลตฟอร์มของหุ่นยนต์ กระบวนการเขียนโปรแกรมของ ROS นั้นจะใช้รูปแบบ Graph architecture ซึ่งจะทำให้สามารถแบ่งโปรแกรมต่างๆออกเป็นส่วนๆ เช่น เชนเชอร์หลายๆตัว ระบบควบคุม ระบบวางแผน ระบบขับเคลื่อน ระบบสื่อสารภายนอก ด้วยตัวระบบของ ROS นั้น ไม่ใช่ Real Time OS แต่อย่างไรก็ตามเราสามารถใช้งานผสมกับ Real Time ได้

ROS ประกอบไปด้วยแพ็กเกจเจตต่างๆ มาประกอบกันเป็น โนนด(Node) โดยมีตัวกลางทำหน้าที่ในการติดต่อสื่อสารระหว่างอุปกรณ์ที่เป็นโนนดต่างๆ ให้สามารถส่งข้อมูลหากันได้ รูปแบบการสื่อสารใน ROS จะใช้หลักการแบบ Publish/Subscribe ทำให้ไม่จำเป็นที่จะต้องระบุโปรแกรมที่จะรับ ภาษาในการพัฒนามีให้เลือกที่หลากหลาย เช่น C++, Python, Lisp, MATLAB หรือ JavaScript

#### ประโยชน์จากการใช้ ROS

ROS เป็นกรอบการทำงาน ที่อยู่ระหว่าง OS และ Robot ทำให้เราไม่ต้องกังวลเรื่องการจัดการระบบภายใน เพราะ ROS จะช่วยจัดการให้เราทั้งหมด ก่อนจะมี ROS นั้น นักวิจัยจะต้องใช้เวลาไปกับพัฒนาพื้นฐานให้หุ่นยนต์ ซึ่งจะต้องมีทักษะทางด้านเครื่องกล ไฟฟ้า และโปรแกรม ซึ่งบอยครั้งที่นักวิจัยหรือนักพัฒนานั้นไม่มีความรู้ หรือประสบการณ์ในการสร้างหุ่นยนต์ ทำให้การทำงานเป็นไปด้วยความลำบาก

## บทที่ 3

### การดำเนินงานวิจัย

#### 3.1 แผนการดำเนินงาน

การออกแบบหุ่นยนต์ชีวามโนยด้วย Universal Template for Humanoid Algorithm Interface โดยใช้ชื่อว่า อุทัย (UTHAI) มาจากภาษาอังกฤษคำว่า Universal Template for Humanoid Algorithm Interface เพื่อให้สมกับเป็นหุ่นยนต์ชีวามโนยด้วย Universal Template for Humanoid Algorithm Interface ที่ใช้สำหรับงานวิจัยและพัฒนาต่อยอด การดำเนินงานและการออกแบบการสร้างหุ่นยนต์ชีวามโนยด้วย Universal Template for Humanoid Algorithm Interface มีแผนการทำงานซึ่งแบ่งออกเป็นสามส่วนดังนี้ ส่วนแรกคือ ส่วนของอาร์ดแวร์ที่เกี่ยวกับโครงสร้างทางกลของหุ่นยนต์ชีวามโนยด้วย Universal Template for Humanoid Algorithm Interface เช่น ข้อต่อ ก้านต่อ ฝ่าเท้า รวมไปถึงระบบอิเล็กทรอนิกส์ ตัวประมวลผลการควบคุม เช่นเซอร์วิล์ฟ์ ตัวขับเคลื่อนต่างๆ และส่วนที่สองคือ ส่วนของซอฟต์แวร์ที่เกี่ยวข้องกับการติดต่อสื่อสารกันเบื้องต้น การควบคุมตัวขับเคลื่อนที่ข้อต่อ การอ่านค่าเซนเซอร์ และส่วนที่สาม คือระบบพื้นฐานสำหรับการนำไปศึกษาและพัฒนา

ในการเริ่มต้นทำงานวิจัยเกี่ยวกับหุ่นยนต์ชีวามโนยด้วย Universal Template for Humanoid Algorithm Interface ที่ต้องทำในขั้นตอนแรกคือการศึกษาสิ่งที่เคยมีอยู่ หรืองานวิจัยได้ทำเอาไว้แล้ว ศึกษาทำความเข้าใจใน ข้อดี-ข้อเสีย ของวิธีหรือกระบวนการต่างๆ เพื่อนำมาประยุกต์ใช้กับหุ่นยนต์ชีวามโนยด้วย Universal Template for Humanoid Algorithm Interface เพื่อให้ได้สิ่งที่ดีที่สุด โดยการศึกษานั้นจะเริ่มต้นจากการศึกษาโครงสร้างทางกลของหุ่นยนต์ชีวามโนยด้วย Universal Template for Humanoid Algorithm Interface ที่มีอยู่แล้วและมีสิ่งที่ต้องดูเป็นพิเศษคือ วิธีการเชื่อมต่อ กันระหว่างก้านต่อและข้อต่อ จุดที่ใช้ติดตั้ง IMU และการเลือกใช้วัสดุให้เหมาะสม รวมถึงการทำงานของเซนเซอร์ที่จำเป็นต้องใช้ในการควบคุมการทำงานของหุ่นยนต์ และการทำงานของอุปกรณ์ขับเคลื่อนต่างๆ เช่น การสั่งงานมอเตอร์ Digital servo ผ่าน Protocol ของ Dynamixel โดยขั้นตอนในการทำงานวิจัยครั้นนี้สามารถแบ่ง การทำงานได้ 7 ส่วน คือ

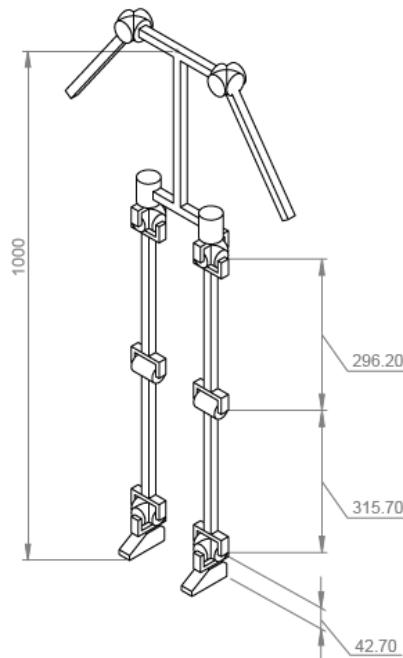
1. ออกแบบระบบโครงสร้างทางกลของหุ่นยนต์ชีวามโนยด้วย Universal Template for Humanoid Algorithm Interface (Solidworks)
2. นำแบบจำลองของหุ่นยนต์เข้าทดสอบการเคลื่อนไหวด้วยโปรแกรมจำลองระบบ (Gazebo)
3. ทดสอบเซนเซอร์ต่างๆ เพื่ออ่านค่าอุปกรณ์ใช้งาน
4. จัดทำชิ้นส่วนโครงสร้างทางกล และประกอบหุ่นยนต์ชิ้นจริง
5. วางแผนระบบการเชื่อมต่อสื่อสาร และการส่งข้อมูลของอุปกรณ์ต่างๆ
6. ติดตั้งอุปกรณ์บอร์ดควบคุมการทำงาน และเซนเซอร์เข้ากับตัวหุ่นยนต์
7. จัดทำคู่มือและเอกสารการใช้งานส่วนต่างๆ ของหุ่นยนต์

### 3.2 การออกแบบโครงสร้างของหุ่นยนต์

การออกแบบทางโครงสร้างทางกลนั้น ผู้วิจัยได้เลือกใช้โปรแกรม Solidworks เป็นเครื่องมือที่ช่วยในการพัฒนาโมเดลของหุ่นยนต์ขีวามโนยด์ เนื่องจากโปรแกรม Solidworks เป็นโปรแกรมที่มีความสามารถในการขึ้นรูปและวาดแบบทางวิศวกรรม สามารถวิเคราะห์โครงสร้างทางกลของแบบจำลองได้ และมีการใช้งานอย่างแพร่หลาย อีกทั้งยังสามารถดาวน์โหลดโมเดลต่างๆ ที่มีคนพัฒนาเข้ามาใช้ร่วมกับการออกแบบของเราได้ และด้วยทางผู้วิจัยมีความชำนาญถึงความสามารถในการพัฒนาต่อยอดเป็นหลัก ดังนั้นการออกแบบโครงสร้างทางกลของหุ่นยนต์ ขีวามโนยด์ UTHAI จึงถูกออกแบบมาเพื่อให้สามารถรองรับการเปลี่ยนแปลง แก้ไขส่วนต่างๆ ของตัวหุ่นยนต์เอง ได้ในอนาคตอีกด้วย

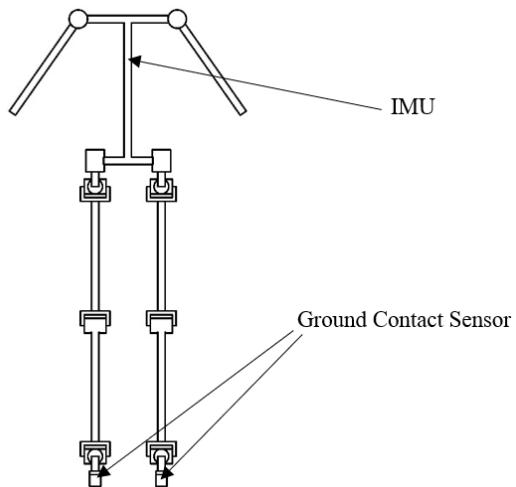
#### 3.2.1 โครงสร้างหุ่นยนต์

หุ่นยนต์ที่สร้างขึ้นประกอบด้วยส่วนของลำตัวและส่วนขา ในขาแต่ละข้างออกแบบให้เป็นลักษณะของข้อต่อหมุน (Revolute joint) เลียนแบบโครงสร้างของมนุษย์ซึ่งประกอบด้วย ส่วนของสะโพกที่มีองศาอิสระจำนวน 3 องศาอิสระ ส่วนของหัวเข่า 1 องศาอิสระและส่วนของข้อเท้า 2 องศาอิสระ รวมขาข้างละ 6 องศาอิสระ ระบบต้นกำลังที่ใช้เป็น Dynamixel servo การออกแบบหุ่นยนต์นั้น สิ่งแรกที่ต้องทำ คือ การกำหนดโครงสร้างของข้อต่อและก้านต่อขึ้นมาก่อน โดยวัดขึ้นมาเป็นเหมือนโครงกระดูก ซึ่งโครงสร้างนั้นทางผู้วิจัยได้อ้างอิงมาจากสัดส่วนของมนุษย์จริง ที่ประกอบด้วยส่วนของขาข้างละ 6 องศาอิสระ และมีจุด CoM อยู่บริเวณกระดูกเชิงกรานของตัวหุ่นยนต์เอง



รูปที่ 3.1: ภาพแสดงแสดงโครงสร้างของหุ่นยนต์ UTHAI

เมื่อเราได้แบบจำลองของหุ่นยนต์ขีวามโนยด์ออกมาแล้ว ลำดับต่อไปคือการกำหนดตำแหน่งการติดตั้งเซนเซอร์และตัวขับเคลื่อนต่างๆ เข้าไป โดยมี Ground contact sensor ติดตั้งที่ใต้ฝ่าเท้าของหุ่นยนต์, IMU sensor ติดตั้งไว้ให้ใกล้กับจุด COM ของหุ่นยนต์ และ Dynamixel servo ติดตั้งไว้ที่ข้อต่อในแต่ละข้อต่อ



รูปที่ 3.2: ภาพแสดงการติดตั้งเซนเซอร์ในจุดต่างๆ

ส่วนโครงสร้างหุ่นยนต์ข้อมานอยด์ UTHAI ทางผู้วิจัยเลือกใช้วัสดุหลักเป็น PLA ที่ขึ้นรูปด้วยวิธีการขึ้นรูปสามมิติ และมีวัสดุเสริมบางชิ้นส่วนจากอลูมิเนียม เนื่องจากจะทำให้โครงสร้างมีน้ำหนักเบา สามารถปรับปรุงแก้ไขง่าย และมีราคาที่สมเหตุสมผล

Material	Longitudinal Tensile Strength (ksi)	Density (g/cm <sup>3</sup> )
Carbon Fiber	300	1.55
Steel	100	7.7
Titanium	120	4.34
Aluminum	35	2.7
PLA 3D printing (50 % infill)	3.5	1.26
PLA 3D printing (100 % infill)	5.5	1.26

ตารางที่ 3.1: ตารางแสดงสมบัติทางกลของวัสดุต่าง ๆ

### 3.2.2 จัดทำขึ้นส่วนโครงสร้างและประกอบ

ในการจัดทำขึ้นส่วนโครงสร้างนั้นทางผู้จัดทำได้คำนึงถึงความแข็งแรงเป็นหลักซึ่งมีความสำคัญมาก ในการเคลื่อนที่ของหุ่นยนต์ และยังคงต้องมีน้ำหนักที่เบาอีกด้วย ดังนั้นจึงได้ใช้การขึ้นรูปขึ้นงานด้วยเทคนิค การพิมพ์ 3 มิติ โดยจะใช้วัสดุหลักเป็นพลาสติก PLA ซึ่งมีความแข็งมากกว่าและขึ้นรูปง่ายกว่าพลาสติกชนิด ABS เพื่อให้ตอบโจทย์กับหุ่นยนต์แพลตฟอร์มเพื่อพัฒนาต่อยอดในอนาคต ซึ่งผู้ใช้ทุกคนสามารถพิมพ์ขึ้นมาได้ด้วยตนเอง<sup>1</sup>

<sup>2</sup>

Properties	ABS	PLA
Tensile Strength	27 MPa	37 MPa
Elongation	3.5 - 50%	6%
Flexural Modulus	2.1 - 7.6 GPa	4 GPa
Density	1.0 - 1.4 g/cm <sup>3</sup>	1.3 g/cm <sup>3</sup>
Melting Point	230°C - 240°C	215°C - 235°C
การย่อยสลายทางธรรมชาติ	ไม่มีเด้	ได้(ภายใต้เงื่อนไขที่ถูกต้อง)

ตารางที่ 3.2: ตารางแสดงสมบัติทางกลของวัสดุพลาสติก

### 3.2.3 การเชื่อมต่อหุ่นยนต์เข้ามานอยด์

โครงสร้างแพลตฟอร์มหุ่นยนต์อุทัยจะประกอบไปด้วยสองขา เพื่อทำให้เกิดองศาอิสระเป็น 12 องศาอิสระ (DOFs) ใช้ Dynamixel servos 12 ตัว มอเตอร์ Dynamixel ทั้งหมดเชื่อมต่อกันแบบ daisy chain ข้างนึงของมอเตอร์ตัวแรกเชื่อมต่อกับแบบเตอร์รี่ 12V และอีกข้างต่อไปกับ USB2Dynamixel ทั้งหมดนี้เป็นการเชื่อมต่อ Odroid เข้ากับหุ่นยนต์ ดังรูปที่

หุ่นยนต์เข้ามานอยด์อุทัยใช้เซอร์วิsmoเตอร์ 12 ตัว ทำให้เกิดเป็น 12 องศาอิสระ USB2Dynamixel ใช้เพื่อที่จะส่งการเซอร์วิsmoเตอร์ Dynamixel ผ่าน Odroid ตำแหน่งของเซอร์วิsmoเตอร์ Dynamixel EX-106+ นั้นมา จากเอนโคดเดอร์ที่อยู่ภายใน เช่นเซอร์ Gyro/Accelerometer ติดอยู่กับตัวของหุ่นยนต์ เพื่อช่วยในการทรงตัวของหุ่นยนต์ เช่นเซอร์ Accelerometer จะอัพเดตค่าของตัวเองเรื่อยๆ ฟังก์ชันส่วนเสริมจะมาจาก ROS และ Odroid เชื่อมต่อกับคอมพิวเตอร์ภายนอกผ่าน Wi-Fi

<sup>1</sup>Printing Guide [<https://filaments.ca/pages/temperature-guide>]

<sup>2</sup>PLA vs ABS [<https://www.3dhubs.com/knowledge-base/pla-vs-abs-whats-difference>]

### 3.2.4 อุปกรณ์ที่ใช้ในหุ่นยนต์อิริวานอยด์อุทัย

#### Dynamixel servo EX-106+

Dynamixel EX-106+ เป็นตัวขับเคลื่อนที่นิยมใช้ในปัจจุบัน ซึ่งเป็นเซอร์โวโมเตอร์ที่เหมาะสมสำหรับทำหุ่นยนต์โดยเฉพาะ ภายในประกอบไปด้วย มอเตอร์กระแสตรง ชุดเพื่องมอเตอร์ ไดรเวอร์คอนโทรลเลอร์ สามารถเชื่อมต่อกันผ่าน BUS RS-485<sup>3</sup> มีการควบคุมแบบ PID สามารถที่จะอ่านค่าความเร็ว แรงดันไฟฟ้า กระแสไฟฟ้า อุณหภูมิ ตำแหน่ง และแรงบิดจากมอเตอร์ทุกตัวได้ แต่ละมอเตอร์แต่ละตัวจะมีบอร์ดควบคุมของตัวเอง เราสามารถที่จะจ่ายไฟให้มอเตอร์และควบคุมผ่าน Serial ได้เลย

การทำงานของตัวขับเคลื่อนนี้สามารถทำได้ 2 รูปแบบคือ<sup>4</sup>

**Joint Mode** สามารถที่จะควบคุม Torque Speed และ position ได้ ความละเอียดในการควบคุม 10-bit (0-1023) หมุนได้อยู่ในช่วง 0-250 องศา

**Wheel Mode** สามารถที่จะควบคุม Torque Speed และ direction ได้ ความละเอียดของความเร็ว มอเตอร์เท่ากับ 10bit (0-1023) สามารถหมุนได้ครบ 360 องศาได้

EX-106 Stats		
<b>Operating Voltage</b>	18.5V	14.8V
<b>Holding Torque</b>	107 kg·cm 1,485 oz·in	84 kg·cm 1,166 oz·in
<b>No-load Speed</b>	0.143 sec/60°	0.182 sec/60°
<b>Weight</b>	154g	
<b>Size</b>	40.2 x 65.1 x 46 mm	
<b>Resolution</b>	0.06°	
<b>Reduction Ratio</b>	1/184	
<b>Operating Angle</b>	251° or Continuous Turn	
<b>Max Current</b>	7000mA	
<b>Standby Current</b>	55 mA	
<b>Operating Temp</b>	-5°C ~ 85°C	
<b>Protocol</b>	RS485 Asynchronous Serial	
<b>Module Limit</b>	254 valid addresses	
<b>Com Speed</b>	7343bps ~ 1Mbps	
<b>Position Feedback</b>	Yes	
<b>Temp Feedback</b>	Yes	
<b>Load Voltage Feedback</b>	Yes	
<b>Input Voltage Feedback</b>	Yes	
<b>Compliance/PID</b>	Yes	
<b>Material</b>	Metal Gears & Engineering Plastic Body	
<b>Motor</b>	Maxon RE-MAX	
<b>Manual Download</b>	<a href="#">EX-106 manual (PDF)</a>	
<b>Controller List</b>	<a href="#">USB2Dynamixel CM2+</a>	



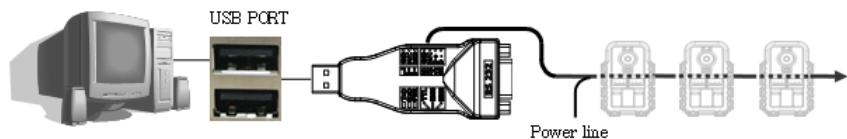
รูปที่ 3.3: แสดงประสิทธิภาพของมอเตอร์ EX-106+

<sup>3</sup>Robot Actuator [[http://support.robotis.com/en/product/actuator/dynamixel/ex\\_series/ex-106.htm](http://support.robotis.com/en/product/actuator/dynamixel/ex_series/ex-106.htm)]

<sup>4</sup>EX-106+ Mode [<http://www.trossenrobotics.com/dynamixel-ex-106-robot-actuator.aspx>]

### USB2Dynamixel connector

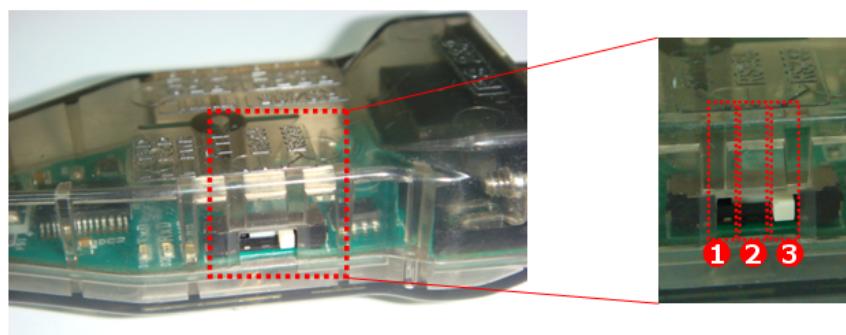
USB2Dynamixel เป็นอุปกรณ์สำหรับเชื่อมต่อ Odroid กับ Dynamixel โดยจะเชื่อมต่อผ่านพอร์ท USB ของ Odroid ไปยัง Dynamixel motor ผ่านสายทั้งหมด 4 เส้น เป็นการเชื่อมต่อแบบ RS-485<sup>5</sup>



รูปที่ 3.4: ภาพแสดงการติดต่อสื่อสารระหว่าง PC กับมอเตอร์ Dynamixel

ในการต่อใช้งานนั้นผู้ใช้งานจำเป็นต้องเลือกการติดต่อสื่อสารระหว่าง คอมพิวเตอร์กับ มอเตอร์ ซึ่งการติดต่อสื่อสารนั้น USB2Dynamixel ได้แบ่งการติดต่อสื่อสารออกเป็น 3 ส่วนคือ

1. TTL Communication : สำหรับมอเตอร์ Dynamixels ที่ใช้พอร์ทชนิด 3-pin เช่นในตระกูล AX Series เช่น AX-S1 AX-12+ ฯลฯ
2. RS485 Communication : สำหรับมอเตอร์ Dynamixels ที่ใช้พอร์ทชนิด 4-pin port เช่นในตระกูล DX Series เช่น RX Series, EX Series ฯลฯ
3. RS232 Communication : ใช้สำหรับติดต่อสื่อสารกับ controller ผ่านสายเคเบิลเช่น CM-5, CM-510 ฯลฯ



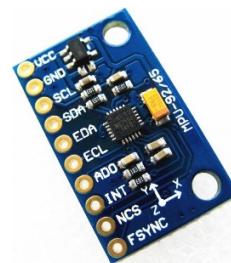
รูปที่ 3.5: ภาพแสดงการเลือกโหมดใช้งานของ USB2Dynamixel

### Inertial Measurement Unit (IMU)

ในการทำวิจัยครั้งนี้ผู้จัดทำได้เลือกนำเซนเซอร์ MPU-9250 มาใช้ในการอ่านค่ามุมเอียงของทุนยนต์เพื่อใช้ในการคุณสมบัติภาพของทุนยนต์ โดยเซนเซอร์ตัวนี้สามารถวัดค่าได้ 9 แกนคือ วัดค่าความเร็วเชิงมุม(gyroscope) 3 แกน วัดค่าความรุ่งเชิงเส้น(accelerometer) 3 แกน และวัดค่าสนามแม่เหล็กโลก(magnetometer) 3 แกน ซึ่งเซนเซอร์ตัวนี้จะติดตั้งบริเวณส่วนของลำตัวทุนยนต์ เนื่องจากว่าจะเป็นจุดที่สามารถบ่งบอกได้ถึงการเคลื่อนที่และมุมเอียงของทุนยนต์ในขณะนั้นได้ดีที่สุด<sup>6</sup>

<sup>5</sup>USB2Dynamixel [[http://support.robotis.com/en/product/auxdevice/interface/usb2dxl\\_manual.html](http://support.robotis.com/en/product/auxdevice/interface/usb2dxl_manual.html)]

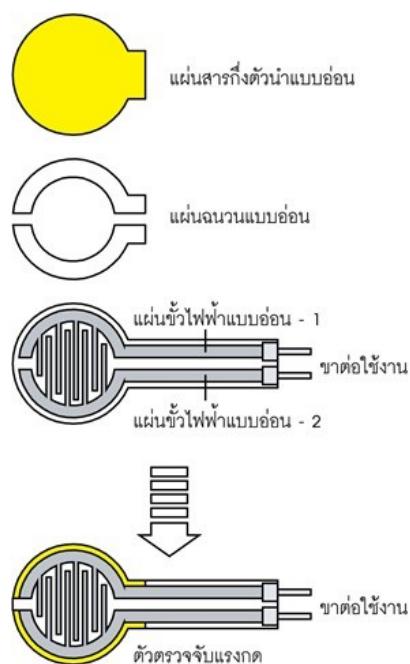
<sup>6</sup>MPU-9250 [<http://www.arduino.com/en/gy-series-axis-accelerometers/6924-gy9255-mpu9255-sensor-module-alternative-mpu9150-mpu9250-3809200640200.html>]



รูปที่ 3.6: แสดงเซนเซอร์ IMU MPU9250

### Ground contact sensor

เซนเซอร์ตรวจหน้าสัมผัสที่พื้นเป็นเซนเซอร์ที่ถูกติดตั้งบริเวณฝ่าเท้า เพื่อตรวจสอบการเดินของหุ่นยนต์หรือมนุษย์ว่าขณะนี้มีการสัมผัสของฝ่าเท้าของหุ่นยนต์กับพื้นหรือไม่ ซึ่งในงานวิจัยนี้ได้ใช้หลักการตัวตรวจจับแรงกดแบบค่าความด้านทานหรือ Force Sensing Resistor (FSR) ที่ใช้เทคโนโลยีฟิล์มโพลีเมอร์แบบหนา (Polymer Thick Film) โดยแรงดันไฟฟ้าที่ตกคร่อมตัวตรวจจับจะลดลง เมื่อมีแรงกดมากระทำบนแผ่นตรวจจับ มีโครงสร้างของตัวตรวจจับแสดงในรูปที่??

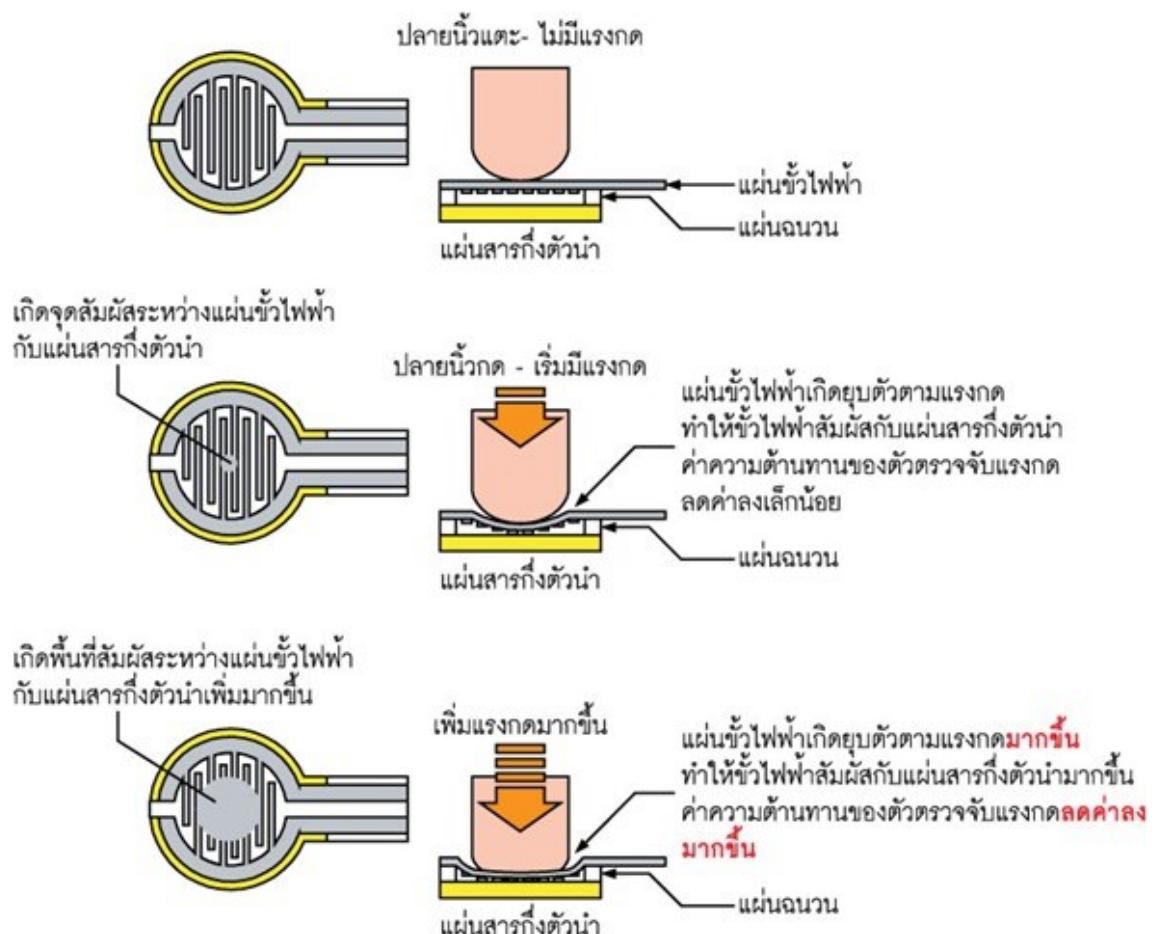


รูปที่ 3.7: ลักษณะโครงสร้างของตัวตรวจจับแรงกด FSR

ประกอบด้วยแผ่นสารกึ่งตัวนำแบบอ่อนที่เป็นตัวกำหนดค่าความด้านทานไฟฟ้าประกบ เข้ากับแผ่นข้าวไฟฟ้าแบบอ่อน โดยมีแผ่นวนแบบอ่อนคั่นกลาง ทำให้เกิดค่าความด้านทานไฟฟ้าขึ้นระหว่างขาต่อไป้งาน เมื่อมีการกดลงบนแผ่นข้าวไฟฟ้า จะทำให้เกิดการสัมผัสระหว่างสารกึ่งตัวนำกับข้าวไฟฟ้า ส่งผลให้ค่าความด้านทานไฟฟ้าเกิดการเปลี่ยนแปลง ดังแสดงกระบวนการทำงานในรูปที่??

### Wi-Fi Adapter

ตัวรับสัญญาณ WiFi ชนิดพกพาเพื่อใช้ในการติดต่อสื่อสารระหว่างคอมพิวเตอร์และบอร์ดควบคุม ซึ่งในโครงงานนี้ จะใช้ส่งข้อมูลที่ต้องการ การประมวลผลบนคอมพิวเตอร์ เช่น การวางแผนการเดิน การคำนวณเรื่องพลศาสตร์ของหุ่นยนต์ โดยจะมีตัวกลางในการรับส่งสัญญาณผ่านตัวกระจายสัญญาณ(wifi router)



รูปที่ 3.9: ตัวรับสัญญาณ wifi ของ RaspberryPi

### 3.3 การออกแบบโปรแกรมด้วย ROS

#### 3.3.1 Modelling

หลังจากที่เราได้ออกแบบและไม่เดลหุ่นยนต์ของเราขึ้นมาที่ใช้ CAD tools ต่างๆ เช่น AutoCAD, SolidWorks, Blender หรืออื่นๆ ก็เพื่อที่จะนำมาใช้ในการทำ Simulation การที่เราทำ Simulation นั้นก็จะสามารถมองเห็นหุ่นยนต์ และเห็นการทำงานของหุ่นยนต์เรา ก่อนที่เราจะสร้างมันขึ้นมาจริงๆ หุ่นยนต์จำลองที่เราสร้างขึ้นมา นั้นควรที่จะมีลักษณะให้ใกล้เคียงกับของจริงมากที่สุด ไม่ว่าจะเป็นรูปร่าง รูปทรง น้ำหนักต่างๆ

##### 3.3.1.1 ROS packages for robot modelling

ROS นั้นได้ให้เครื่องมือที่ช่วยให้เราสามารถสร้าง 3D robot models ได้ ใน ROS มี meta package ที่ชื่อว่า `robot_model` ซึ่งข้างในมี package ต่างๆ ที่ใช้สำหรับสร้าง 3D robot models

`urdf` เป็น 1 ในหลายๆ package ที่อยู่ใน `robot_model`, `urdf` เป็น xml ไฟล์ที่เอาไว้ใช้บอกลักษณะของหุ่นยนต์ ย่อมาจาก Unified Robot Description Format(URDF) เราสามารถระบุ robot model, sensors และ working environment โดยใช้ URDF การบอกนั้นจะสามารถบอกเป็นเหมือน tree structure ของ link ต่างๆ ในตัวหุ่นยนต์ สามารถบอก rigid link เชื่อมต่อกันผ่าน joints แต่ถ้าเป็น flexible link จะไม่สามารถบอกได้โดยใช้ `urdf`

`joint_state_publisher` เครื่องมือที่ใช้ในการ publish ข้อมูล robot URDF เพื่อสามารถรับรู้การเคลื่อนไหวของ joints ทุก joint ที่ไม่ใช่ fixed joints มาแสดงเป็น GUI sliders ทำให้เราสามารถเลื่อนๆ หมุนๆ ไปได้ อีกทั้งยังสามารถใช้งานร่วมกับ `rviz`

`robot_state_publisher` เป็นเครื่องมือที่ใช้ในการ publish 3d pose ของ link ต่างๆ ใน `urdf` การ publish นั้นจะใช้ ROS tf(transform) ROSTf คือการทำความสัมพันธ์ระหว่าง frame ของหุ่นยนต์

`xacro` ย่อมาจาก XML Macros หรือเราสามารถเรียกอีกอย่างว่า URDF plus add-ons. ซึ่งการทำงานเหมือนกับ `urdf` แต่ทำให้ไฟล์ `urdf` สั้นกว่า อ่านง่ายกว่า และสามารถใช้เพื่อทำให้สร้างหุ่นยนต์ที่มีความซับซ้อนง่ายขึ้น เราสามารถแปลงไฟล์ `xacro` เป็น `urdf` ได้

##### 3.3.1.2 URDF

ในส่วนนี้จะเป็นการอธิบายระบบทางกลของหุ่นยนต์ข้อมูลอยู่เป็นไฟล์ที่ใช้ร่วมกับ ROS ได้ เพื่อที่จะสามารถนำไปใช้กับ Simulation ในอนาคตได้ ในการอธิบายระบบทางกลนั้นผู้วิจัยได้ใช้ไฟล์ URDF (Universal Robotics Description Format) ซึ่งใช้ภาษาการเขียนเป็น XML ในการบอกส่วนประกอบแต่ละส่วนของหุ่นยนต์

##### Link

ในไฟล์ URDF แต่ละขั้นส่วนของหุ่นยนต์เราจะเรียกว่า link และใน link จะประกอบไปด้วยส่วนย่อยๆ 3 ส่วนคือ `<inertia>` ที่เอาไว้บอกถึงค่าตัวแปรทางฟิสิกส์, `<visual>` ที่เอาไว้แสดงผลให้เราเห็น, `<collision>` ที่เอาไว้ตรวจสอบว่าหุ่นยนต์มีการชนกันกับสิ่งแวดล้อมใหม่ ดังรูปที่ 3.5

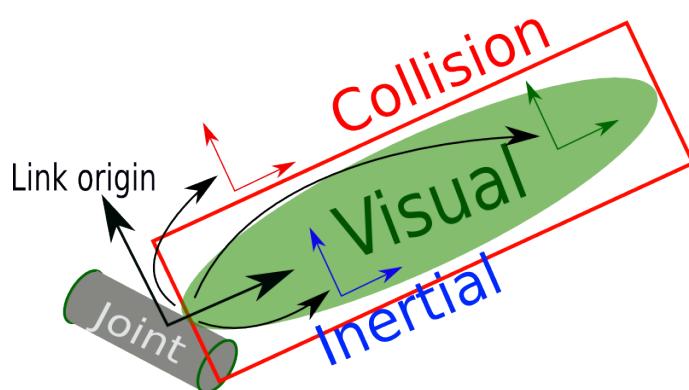
```

<link name="my_link">
  <inertia>
    <origin xyz="0 0 0.5" rpy="0 0 0"/>
    <mass value="1"/>
    <inertia ixx="100" ixy="0" ixz="0" iyy="100" iyz="0" izz="100"/>
  </inertia>
  <visual>
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <geometry>
      <box size="1 1 1" />
    </geometry>
    <material name="Cyan">
      <color rgba="0 1.0 1.0 1.0"/>
    </material>
  </visual>
  <collision>
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <geometry>
      <cylinder radius="1" length="0.5"/>
    </geometry>
  </collision>
</link>

```

รูปที่ 3.10: ตัวอย่าง link ใน urdf

ยังมี tags อีกหลายตัวที่ใช้ในการอธิบายแต่ละชิ้นส่วนของหุ่นยนต์ แต่ตัวอย่างเป็นเพียงแค่ส่วนหนึ่งเท่านั้น ในความเป็นจริงแล้วเราจะใช้ tags ต่างๆ ก็ตามที่เราต้องการ โดยใน URDF ไฟล์นั้นจะเอาไว้เก็บข้อมูลลักษณะเฉพาะของหุ่นยนต์เอาไว้ และยังสามารถใช้กับซอฟแวร์ตัวอื่นๆ ได้



รูปที่ 3.11: การอธิบาย link ใน URDF ไฟล์

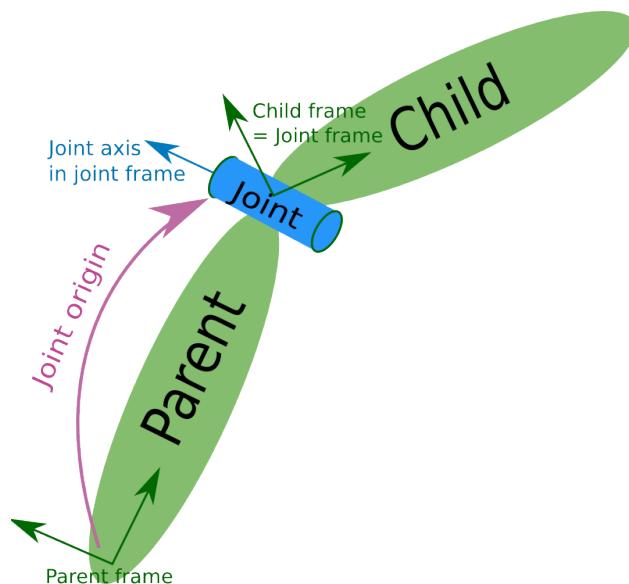
## Joint

อีกส่วนที่สำคัญสำหรับการสร้างไฟล์หุ่นยนต์ด้วย URDF ก็คือ Joint tag โดย tag นี้จะอธิบายถึงความสัมพันธ์ระหว่างก้านต่อสองอัน ส่วนนี้ไม่ได้มีเพียงแค่ทำข้อต่อให้เป็นแบบหมุนได้อย่างเดียว ยังมี Fix, Revolution, Linear และ Planar นอกเหนือจากนี้ เราจึงสามารถที่จะเพิ่มองศาสูงสุดต่ำสุดของข้อต่อ รวมไปถึง dynamic properties ต่างๆ ตามที่เห็นดังรูปที่ 3.7

```
<joint name="my_joint" type="floating">
    <origin xyz="0 0 1" rpy="0 0 3.1416"/>
    <parent link="link1"/>
    <child link="link2"/>
    <calibration rising="0.0"/>
    <dynamics damping="0.0" friction="0.0"/>
    <limit effort="30" velocity="1.0" lower="-2.2" upper="0.7"/>
    <safety_controller k_velocity="10" k_position="15"
        soft_lower_limit="-2.0" soft_upper_limit="0.5"/>
</joint>
```

รูปที่ 3.12: ตัวอย่าง joint ใน urdf

เมื่อเรานำ Joint และ Link มารวมกันเราจะต้องพิจารณาว่ามี wang รูปแบบเป็นไปตามรูปที่ 3.8 โดยจะมีระยะระหว่างแกนของแต่ละข้อต่อ กับ ก้านต่อ ขึ้นส่วนแรกของการสร้างไฟล์ URDF จะมีชื่อว่า base\_link และเฟรม origin จะเป็นเฟรมอ้างอิง เมื่อเราต่อ Joint เข้ากับ Link จะเรียกว่า parent โดยเฟรม origin ของข้อต่อจะอยู่จุดเดียวกับเฟรม origin ของก้านต่อ ในสถานะเดียวกันก้านต่อที่นำมาต่อจากข้อต่อ เราจะเรียกว่า child และเฟรม origin ของก้านต่อ child จะอยู่ที่จุดเดียวกับเฟรม origin ของข้อต่อ



รูปที่ 3.13: การอธิบาย Joint ใน URDF ไฟล์

### 3.3.2 กำหนดพิกัดเฟรมให้กับหุ่นยนต์อิวามาอยด์

การกำหนดเฟรมให้กับหุ่นยนต์อิวามาอยด์นั้นเราจะใช้หลักตามของ ROS Enhancement Proposals (REPs) ซึ่งจะทำให้เราสามารถใช้เครื่องมือต่างๆ ที่มีคุณสร้างขึ้นมาใช้งานได้ง่าย และช่วยทำให้เกิดความเข้าใจได้ง่าย

#### **base\_link**

เป็นเฟรมที่ติดอยู่กับฐานของหุ่นยนต์ โดยจะติดตำแหน่งหรือมุมเอียงได้ ก็ได้ ปกติแล้วจะติดที่สะโพกของหุ่นยนต์

#### **base\_footprint**

เป็นเฟรมที่เอาไว้แสดงว่าหุ่นยนต์อยู่ตรงไหนบนพื้นโลก โดยปกติแล้วจะมีระดับอยู่ที่จุดต่ำสุดของฝ่าเท้า  $z = \min(l\_sole\_z, r\_sole\_z)$  โดย  $l\_sole\_z$  และ  $r\_sole\_z$  คือความสูงของฝ่าเท้าซ้ายและขวา  $base\_footprint$  เมื่อ non 2D planar ที่บอกตำแหน่งของอิวามาอยด์ระหว่างที่กำลังเดินหรือทำอย่างอื่นอยู่

#### **l\_wrist, r\_wrist**

เป็นเฟรมที่บอกตำแหน่งและมุมเอียงของแขนขาซ้ายและขวาแต่ไม่ได้คำนึงถึงการติดตั้งอุปกรณ์ใดๆเข้าไป

#### **l\_gripper, r\_gripper**

เป็นเฟรมที่บอกตำแหน่งและมุมเอียงของที่ปลายแขน (End effector) ถ้ามีอุปกรณ์อยู่ เฟรมนี้ก็จะไปอยู่ในตำแหน่งของอุปกรณ์นั้นๆ

#### **l\_ankle, r\_ankle**

เป็นเฟรมที่บอกตำแหน่งและมุมเอียงของขาซ้ายและขวาโดยไม่ได้คำนึงว่าจุดรับน้ำหนักของตัวอยู่ที่ไหน

#### **l\_sole, r\_sole**

เป็นเฟรมที่บอกตำแหน่งและมุมเอียงของขาซ้ายและขวาที่รองรับน้ำหนักตัวอยู่ โดยจะนบกการ projection ของ X,Y ใน 2D plane ที่สัมผัสพื้นและ Z จะอยู่ระดับเดียวกับพื้นสัมผัส

#### **l\_toe, r\_toe**

เป็นเฟรมที่บอกตำแหน่งและมุมเอียงของปลายเท้าซ้ายและขวา โดยอยู่บนพื้นผิวที่สัมผัสมอยู่

#### **gaze**

เป็นเฟรมที่บอกตำแหน่งและมุมเอียงของหัว โดยการเอียงนั้นจะบอกทิศทางของหัวโดยไม่ได้สนใจเช่นเชอร์ว่าจะติดตั้งอย่างไร

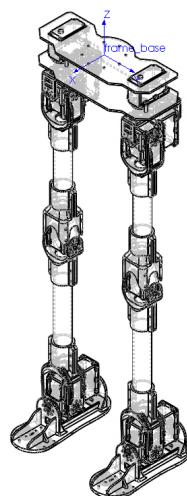
#### **torso**

เป็นเฟรมที่ติดอยู่กับลำตัวซึ่งล่างของหุ่นยนต์โดยจะเป็นตัวที่เชื่อม ขา แขน ตัว หัว เข้ามาไว้ด้วยกัน

### 3.3.3 Box model

### 3.3.4 Dynamic properties

ข้อมูล



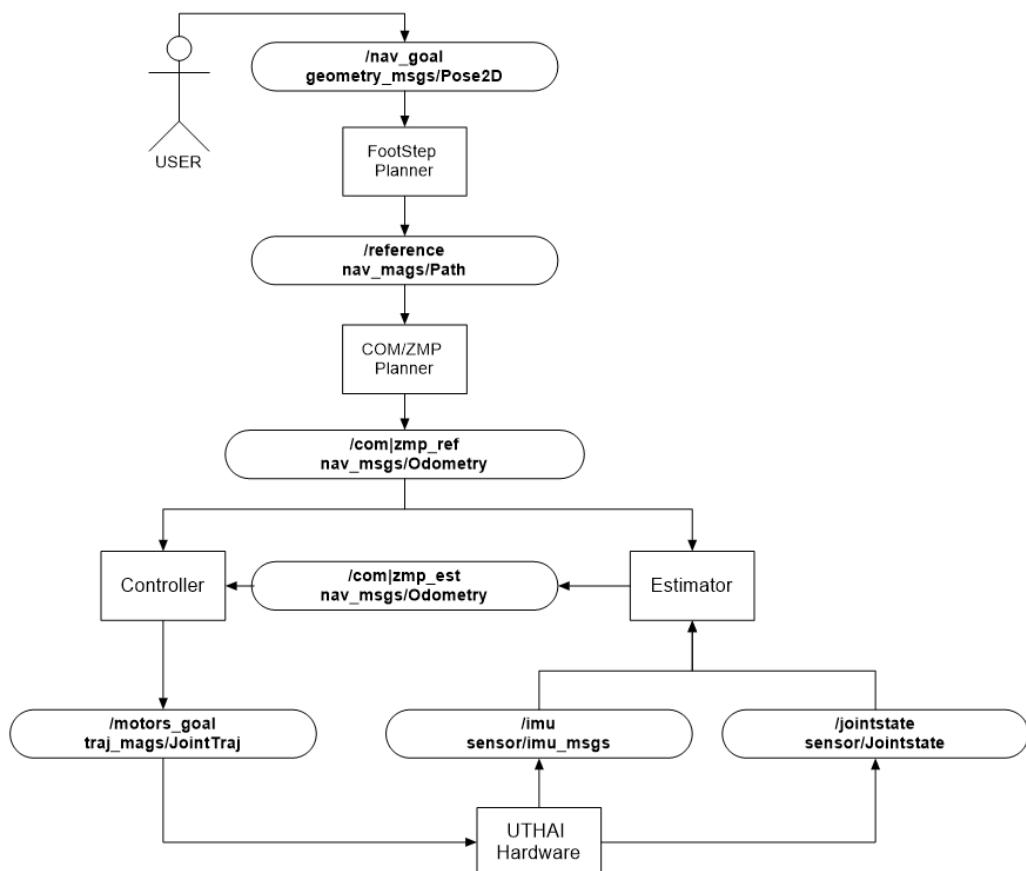
รูปที่ 3.14: ภาพแสดงช่วงล่างทุนยนต์ขีวมานอยด์

Link	All Link
Mass (kg)	3.31477475
CoM X (m)	-0.00855772
CoM Y (m)	0.00000000
Inertia Ixx	0.28641029
Inertia Ixy	-0.00000302
Inertia Ixz	-0.00048106
Inertia Iyy	0.26207601
Inertia Iyz	-0.00061103
Inertia Izz	0.02925799

ตารางที่ 3.3: Message Geometry Point

### 3.3.5 โครงสร้างการติดต่อสื่อสารระหว่าง Node ใน ROS

การติดต่อสื่อสารกันภายใน ROS นั้นจะใช้การส่ง message หากัน ซึ่ง message แต่ละตัวก็จะใช้ในงานที่ต่างกัน ตามระบบที่ต้องการส่ง จากรูปที่ 3.10 เป็นโครงสร้างการส่งข้อมูลหากันของหุ่นยนต์อิวามานอยด์ ที่ผู้วิจัยได้ออกแบบไว้ โดยเริ่มจากผู้ใช้งานส่งตำแหน่งที่หุ่นยนต์จะต้องเดินไปเป็น Node ที่ทำการคำนวณและสร้างตำแหน่งการวางเท้าของหุ่นยนต์ และหลังจากนั้นจะส่งข้อมูลออกไปเป็น Path เส้นทางไปยัง Node ที่ทำการค้นหาตำแหน่งของ com, zmp ของหุ่นยนต์ เพื่อทำการควบคุมและส่งการหุ่นยนต์ต่อไป



รูปที่ 3.15: การติดต่อสื่อสารระหว่าง Node

#### การบอกตำแหน่งและมุมเอียง

การบอกตำแหน่งใน 3 มิติ Point คือการบอก  $x, y, z$  และการบومุมเอียงจะใช้ Quaternion ในการบอกโดยใช้ตัวแปรสี่ตัว คือ  $x,y,z,w$  หากนำทั้งสองมารวมกันเราจะเรียกว่า Pose

geometry_msgs/Point	
float64	x
float64	y
float64	z

ตารางที่ 3.4: Message Geometry Point

geometry_msgs/Quaternion	
float64	x
float64	y
float64	z
float64	w

ตารางที่ 3.5: Message Geometry Quaternion

geometry_msgs/Pose	
geometry_msgs/Point	position
geometry_msgs/Quaternion	orientation

ตารางที่ 3.6: Message Geometry Pose

### การบอกรความเร็วเชิงเส้นและเชิงมุม

การบอกรความเร็วเชิงเส้นใน 3 มิติ คือการบอกรความเร็วตามแนวแกน  $x, y, z$  และการบอกรความเร็วเชิงมุม คือการบอกรความเร็วการหมุนรอบแกน  $x, y, z$  หากนำทั้งสองมารวมกันเราจะเรียกว่า Twist

geometry_msgs/Vector3	
float64	x
float64	y
float64	z

ตารางที่ 3.7: Message Geometry Vector3

geometry_msgs/Twist	
geometry_msgs/Vector3	linear
geometry_msgs/Vector3	angular

ตารางที่ 3.8: Message Geometry Twist

### การบอกรตำแหน่งและความเร็ว

หากนำทั้งสองมารวมกันจะรู้ว่า ตำแหน่ง(Pose) และความเร็ว (Twist) เราจะเรียกว่า Odometry แต่ที่เพิ่มเข้ามาคือ Covariance ซึ่งอาจทำให้เกิดความสับสนได้

nav_msgs/Odometry	
std_msgs/Header	header
geometry_msgs/PoseWithCovariance	pose
geometry_msgs/TwistWithCovariance	twist

ตารางที่ 3.9: Message Navigation Odometry

### ตำแหน่งของหุ่นยนต์

การบอกร่องรอยของหุ่นยนต์บนระนาบ 2 มิติ คือการบอก  $x$ ,  $y$  และ  $\theta$  การบอกนั้นจะบอกว่าตำแหน่งที่หุ่นยนต์อยู่นั้นอยู่ตรงไหนหากเทียบกับแผนที่ รวมไปถึงตำแหน่งของหุ่นยนต์ที่ต้องการจะเดินไปด้วย ซึ่งอ้างอิงมาจากการบอกร่องรอยเริ่มต้นของแผนที่

geometry_msgs/Pose2D	
float64	$x$
float64	$y$
float64	$\theta$

ตารางที่ 3.10: Message Geometry Pose2D

### ตำแหน่งการวางแผนทางของหุ่นยนต์

การจะให้หุ่นยนต์นำที้าไปวางในตำแหน่งที่เราต้องการจากที่ได้จากการคำนวณนั้น จะต้องบอกร่องรอยและบอกมุมเอียงของจุดที่จะไป จากการสร้างจะได้เป็นรายการของที้าซ้ายและขวา โดยอิงจาก ตารางที่ 3.5

nav_msgs/Path	
std_msgs/Header	header
geometry_msgs/PoseStamped[]	poses

ตารางที่ 3.11: Message Navigation Path

geometry_msgs/PoseStamped	
std_msgs/Header	header
geometry_msgs/Pose	pose

ตารางที่ 3.12: Message Geometry PoseStamped

### ตำแหน่ง CoM Zmp ของหุ่นยนต์

ใน Message นี้จะใช้อよุ่ 2 จุดคือ ที่ได้จากการวางแผนของ Node CoM Planner และ Node CoM Estimator โดยทั้งสองจุดใช้ Message เมื่อกันส่งไปยัง Controller เพื่อควบคุมท่าทางต่างๆของหุ่นยนต์ต่อไป Message ที่ใช้คือ Message จากตารางที่ 3.8

nav_msgs/Odometry	
std_msgs/Header	header
geometry_msgs/PoseWithCovariance	pose
geometry_msgs/TwistWithCovariance	twist

Message Navigation Odometry

### การควบคุมข้อต่อของหุ่นยนต์

ในการควบคุมข้อต่อแต่ละข้อของหุ่นยนต์ชีวามาโนyd'n'จะใช้ Message trajectory\_msgs/JointTrajectory ซึ่งสามารถส่ง ตำแหน่ง ความเร็ว ความเร่ง และ แรงบิด ไปได้ ทำให้หากต้องการเปลี่ยนระบบใหม่สามารถทำได้โดยง่าย

trajectory_msgs/JointTrajectory	
std_msgs/Header	header
string[]	joint_names
trajectory_msgs/JointTrajectoryPoint[]	points

ตารางที่ 3.13: Message Trajectory JointTrajectory

trajectory_msgs/JointTrajectoryPoint	
float64[]	positions
float64[]	velocities
float64[]	accelerations
float64[]	effort
duration	time_from_start

ตารางที่ 3.14: Message Trajectory JointTrajectoryPoint

### ค่าเซนเซอร์ข้อต่อของหุ่นยนต์

ที่ข้อต่อของหุ่นยนต์ชีวามาโนyd'm'i' มีเซนเซอร์ที่เอาไว้ใช้ในการอ่านค่าตำแหน่ง ความเร็ว และแรง อยู่ด้วย เราสามารถที่จะใช้ Message sensor\_msgs/JointState สำหรับอ่านค่าตำแหน่ง ความเร็ว แรง ของตัวขับเคลื่อนแล้วส่งให้ Estimator Node ได้

sensor_msgs/JointState	
std_msgs/Header	header
float64[]	position
float64[]	velocity
float64[]	effort

ตารางที่ 3.15: Message Sensor JointState

### ค่าเซนเซอร์ฝ่าเท้าของหุ่นยนต์

ที่ฝ่าเท้าของหุ่นยนต์ชีวามาโนyd'm'i' มีเซนเซอร์ที่เอาไว้ใช้ในการอ่าน แรงกดที่ฝ่าเท้า ใช้ในการเอามากกว่า เท้าสัมผัสพื้นหรือไม่

geometry_msgs/Wrench	
geometry_msgs/Vector3	force
geometry_msgs/Vector3	torque

ตารางที่ 3.16: Message Geometry Wrench

### ค่าเซนเซอร์ IMU ของหุ่นยนต์

เซนเซอร์ IMU เป็นเซนเซอร์ที่เอาไว้ใช้ในการวัด ความเร็วเชิงมุม และ ความเร่งเชิงเส้น หากนำทั้งคู่มารวมกันจะสามารถที่จะแปลงให้วัดมุมอิริยาบถของเซนเซอร์ได้ โดยจะใช้ Message std\_msgs/Imu ในการส่งให้ Node Estimator จากตัวหุ่นยนต์

sensor_msgs/Imu	
std_msgs/Header	header
geometry_msgs/Quaternion float64[9]	orientation
geometry_msgs/Vector3 float64[9]	orientation_covariance
geometry_msgs/Vector3 float64[9]	angular_velocity
	angular_velocity_covariance
	linear_acceleration
	linear_acceleration_covariance

ตารางที่ 3.17: Message Sensor Imu

sensor_msgs/MagneticField	
std_msgs/Header	header
geometry_msgs/Vector3 float64[9]	magnetic_field
	magnetic_field_covariance

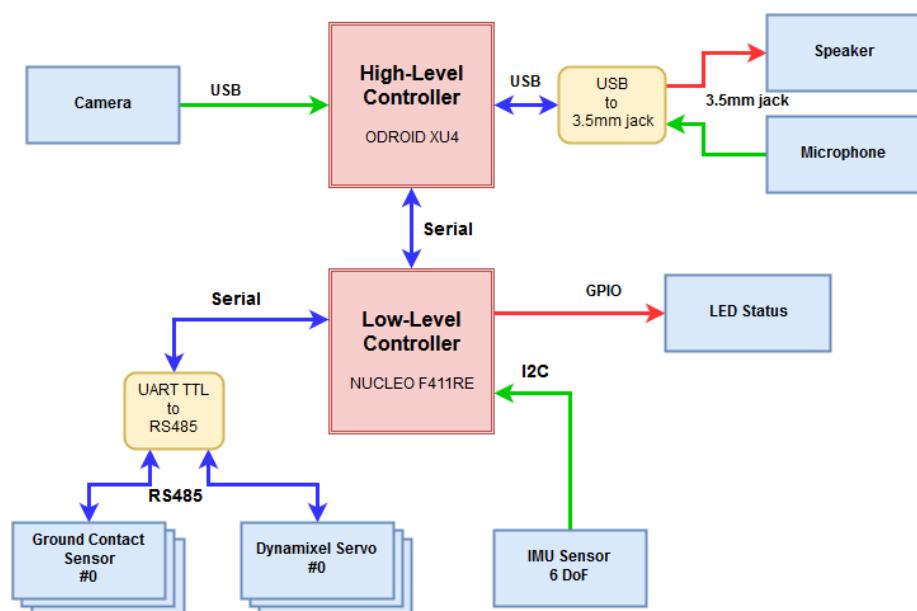
ตารางที่ 3.18: Message Sensor MagneticField

### 3.4 การออกแบบระบบพื้นฐาน

#### 3.4.1 ออกแบบสถาปัตยกรรมของหุ่นยนต์

หลักการออกแบบสถาปัตยกรรมของหุ่นยนต์ชีวภาพอยู่ที่ UTHAI จะออกแบบระบบให้อยู่บนระบบพื้นฐาน ROS เนื่องจากการใช้กรอบการทำงานที่มีประสิทธิภาพ และความยืดหยุ่นสูง จะช่วยทำให้สามารถปรับเปลี่ยนระบบการควบคุมของหุ่นยนต์ชีวภาพอยู่ได้ง่ายและรวดเร็ว ดังนั้นแล้วผู้วิจัยจึงได้แบ่งการประมวลผลออกเป็น 2 ส่วนคือ

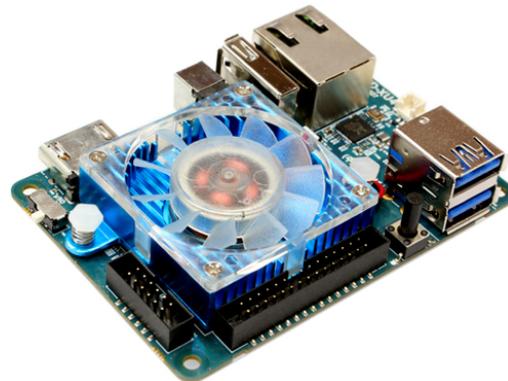
- 1 หน่วยประมวลผลควบคุมระดับสูง (High Level Controller)
- 2 หน่วยประมวลผลควบคุมระดับต่ำ (Low Level Controller)



รูปที่ 3.16: สถาปัตยกรรมของหุ่นยนต์ชีวภาพอยู่ที่ UTHAI

### 3.4.1.1 หน่วยประมวลผลควบคุมระดับสูง (High level controller)

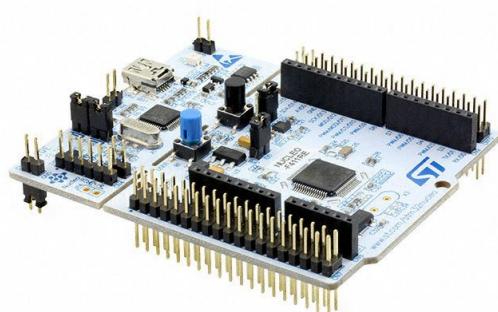
ระบบควบคุมหลักของหุ่นยนต์ UTHAI นั้นจะอยู่ที่หน่วยประมวลผลขั้นสูง ใช้เป็นบอร์ดคอมพิวเตอร์ ODROID-XU4 ตัวประมวลผลหลักนี้ มีหน้าที่ในการทำการคำนวน เส้นทางการเดินของหุ่นยนต์ให้มีเสถียรภาพ ตรวจการขัดกันของโครงสร้างของหุ่นยนต์ รวมไปถึงรับค่าข้อมูลจากกล้อง และข้อมูลเสียงจากไมโครโฟนมา ประมวลผล หลังจากนั้นจะทำการนำค่าทั้งหมดที่ได้จากการคำนวน มาแปลงให้อยู่ในรูปของชุดข้อมูล แล้วส่งออก ไปให้ระบบกลาง (ROS) ในการส่งต่อไปให้อุปกรณ์อื่นต่อไป



รูปที่ 3.17: บอร์ดคอนโทรลเลอร์ Odroid XU4

### 3.4.1.2 หน่วยประมวลผลควบคุมระดับต่ำ (Low level controller)

ระบบควบคุมขั้นต่ำเป็นหน่วยประมวลผลที่ร่องลงมาจาก บอร์ดคอมพิวเตอร์ โดยใช้บอร์ดไมโครคอนโทรลเลอร์ Nucleo F411RE เป็นหน่วยประมวลผลขั้นต่ำ สำหรับในการติดต่อกับอุปกรณ์อิเล็กทรอนิกส์ต่าง ๆ ที่อยู่ภายนอกในตัวของหุ่นยนต์ เช่น ค่าเซนเซอร์ที่ໄพาเท้าซึ่งสามารถบอกได้ว่าควรใช้สมการไหนในการคำนวนพลังงาน หรือ ค่าของเซนเซอร์ IMU มีความสำคัญมาก ในการทำให้หุ่นยนต์มีความนิยมเดินได้อย่างมีเสถียรภาพ เมื่ออ่านค่าเซนเซอร์ต่างๆได้แล้ว หน่วยประมวลผลขั้นต่ำจะนำค่าที่ได้จากการอ่านเซนเซอร์เหล่านี้แปลงให้อยู่ในลักษณะของชุดข้อมูล แล้วส่งออกไปให้ระบบกลาง(ROS) นอกจากนี้หุ่นยนต์ยังมีหน่วยประมวลผลขั้นต่ำยังทำหน้าที่รับค่าคำสั่งมาจากระบบกลาง ในการสั่งงานให้หุ่นยนต์มีท่าทางต่าง ๆ ตามต้องการได้



รูปที่ 3.18: บอร์ดคอนโทรลเลอร์ Nucleo F411RE

### 3.4.2 จัดทำคู่มือและเอกสารการใช้งาน

คู่มือจะเป็นส่วนที่ผู้มาพัฒนาต่อยอดสามารถที่จะอ่านทำความเข้าใจได้ โดยจะเขียนให้อยู่ในรูปของไฟล์ Markdown (.md) และเก็บเอาไว้ในเว็บไซต์ GitHub ซึ่งเป็นแหล่งรวม Source code ออนไลน์ สามารถเข้าไปดาวน์โหลดไฟล์ลงเครื่องผู้ใช้ แล้วทำการติดตั้งใช้งานได้เลย อีกทั้งผู้ใช้งานสามารถส่ง Code ของตัวเองเข้าระบบ เพื่อเพิ่มประสิทธิภาพในการทำงานของโปรแกรม

#### ส่วนที่เน้นในการทำคู่มือคือ

- 1 รายการวัสดุที่ใช้ในการทำหุ่นยนต์ชีวามโนยด์ UTHAI
- 2 รายละเอียดการเชื่อมต่อระหว่างอุปกรณ์ที่อยู่ในตัวหุ่นยนต์
- 3 รายละเอียดการประกอบขั้นส่วนทางกล
- 4 รายละเอียดการใช้งานโปรแกรมพื้นฐาน

#### 3.4.2.1 รายการวัสดุที่ใช้ในการทำหุ่นยนต์ชีวามโนยด์ UTHAI

รายการ	จำนวน(หน่วย)	ราคา/หน่วย(บาท)	ราคารวม(บาท)
===== Processing Unit	-	-	-
Odroid XU4 Embedded Computer	1	3800	3800
Shifter Shield for Odroid XU4	1	1000	1000
===== Sensor	-	-	-
Force sensitive Resistor	8	300	2400
Electronic Component	1	2000	2000
MPU9255 9 Axis IMU Module	1	500	500
===== Structure	-	-	-
อุปกรณ์ส่งกำลัง	1	3000	3000
ค่าวัสดุ เช่น Filament 3D printer , Carbon Fiber	1	8000	8000
สปริง	14	50	700
อุปกรณ์เปลี่ยน เช่น กระดาษทราย ฯลฯ	1	1000	1000
===== อุปกรณ์เสริม Motor Dynamixel	-	-	-
Frame สำหรับต่อพ่วงมอเตอร์	4	2000	8000
Horn Bearing	4	1400	5600
อุปกรณ์จ่ายพลังงาน	-	-	-
Power Supply	1	2000	2000
Battery Li-Po 4 cell	1	3000	3000
===== รวม	-	-	48000

ตารางที่ 3.19: ตารางแสดงรายการของวัสดุต่าง ๆ

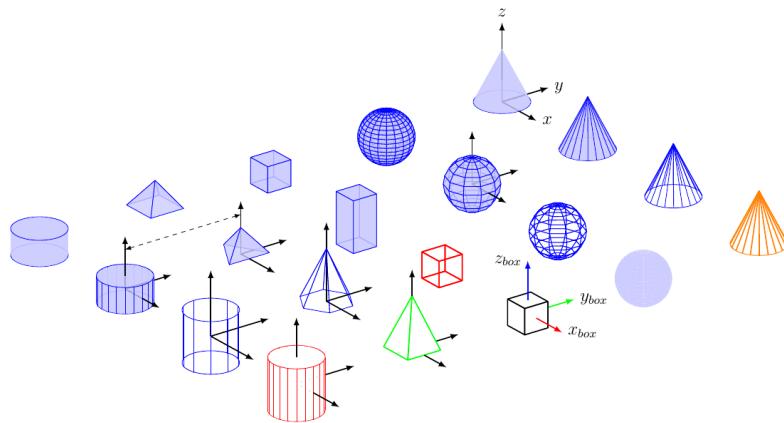
ใช้สำหรับแจกแจงค่าใช้จ่ายเบื้องต้นเท่านั้น ไม่สามารถใช้อ้างอิงงบประมาณแบบละเอียดได้

## UTHAI-Tools

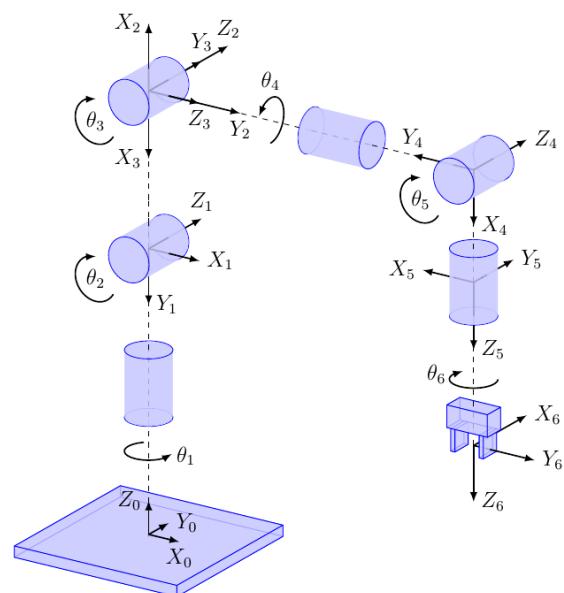
เครื่องมือสำหรับการทำงานในชีวามนอยด์

### sketch-lib

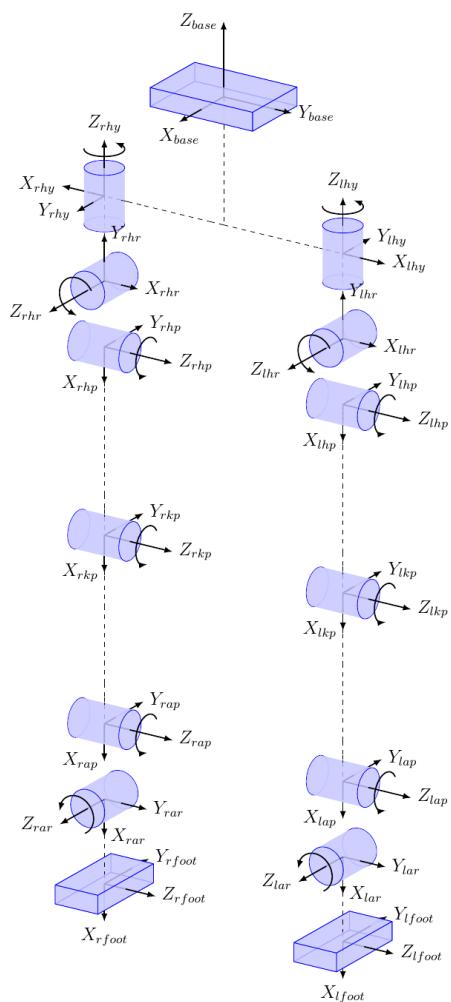
เป็นเครื่องมือที่ใช้สำหรับเอาไว้วัดรูปทรงของหุ่นยนต์



รูปที่ 3.19: ภาพตัวอย่างการวัดออบเจ็คต่างๆ



รูปที่ 3.20: ภาพตัวอย่างการวัดเฟรมของแขนกล



รูปที่ 3.21: ภาพตัวอย่างการรำดเฟรมของหุ่นยนต์ชีวมานอยด์

## บทที่ 4

### ผลการวิจัย

#### 4.1 การออกแบบโครงสร้างของหุ่นยนต์

#### 4.2 การออกแบบโปรแกรมด้วย ROS

##### 4.2.1 Simulation Gazebo

ต้องติดตั้ง package ต่อไปนี้

1 joint\_state\_controller

2 effort\_controller

3 controller\_manager\*

4 gazebo\_ros\_control\*

#### 4.3 การออกแบบระบบพื้นฐาน

## บทที่ 5

### สรุปผลการทดลองและข้อเสนอแนะ

5.1 การออกแบบโครงสร้างของหุ่นยนต์

5.2 การออกแบบโปรแกรมด้วย ROS

5.3 การออกแบบระบบพื้นฐาน

5.4 สรุปภาพรวม

ทำได้ดีนะจ๊ะ

ภาคผนวก

## ภาคผนวก ก

### ผลงานตีพิมพ์เผยแพร่จากวิทยานิพนธ์

ก.1 บทความวิจัยเสนอในที่ประชุมวิชาการและมีการพิมพ์รวมเล่ม

ก.2 บทความวิชาการ

## ภาคผนวก ข

### แหล่งข้อมูล Latex

#### ข.1 แหล่งข้อมูลออนไลน์