

## สารบัญ

เรื่อง	หน้า
<b>สารบัญ .....</b>	<b>ก</b>
<b>บทที่ 1 บทนำ.....</b>	<b>1</b>
1.1 ที่มาและความสำคัญ.....	1
1.2 วัตถุประสงค์.....	2
1.3 ประโยชน์ที่คาดว่าจะได้รับ .....	2
1.4 ขอบเขตการดำเนินงาน.....	2
1.5 ภาพรวมของระบบและขั้นตอนการดำเนินงาน .....	3
<b>บทที่ 2 ทฤษฎีและหลักการ.....</b>	<b>4</b>
2.1 ทฤษฎีและงานวิจัยที่เกี่ยวข้อง .....	4
2.1.1 หุ่นยนต์อิมานอยด์ .....	4
2.1.2 ทฤษฎีที่เกี่ยวข้องกับมนุษย์ .....	7
2.1.3 ทฤษฎีที่เกี่ยวข้องกับหุ่นยนต์อิมานอยด์ .....	9
2.1.4 ตัวอย่างหุ่นยนต์อิมานอยด์ .....	12
2.2 การออกแบบโครงสร้างของหุ่นยนต์ .....	17
2.2.1 ความแตกต่างระหว่างโครงสร้างของมนุษย์กับโครงสร้างของหุ่นยนต์.....	17
2.2.2 อุปกรณ์ที่ใช้ในหุ่นยนต์อิมานอยด์.....	18
2.3 การออกแบบโปรแกรมด้วย ROS .....	21
2.3.1 ระบบที่ใช้ช่วยในการพัฒนาหุ่นยนต์ .....	21
2.3.2 ระบบที่ใช้ในการจำลองการทำงานของหุ่นยนต์.....	24
2.3.3 Robot Operating System.....	26
2.4 การออกแบบระบบพื้นฐาน .....	32
2.4.1 ความแตกต่างของ Operating Systems.....	32
2.4.2 ข้อแตกต่างระหว่าง Open platform กับ Non-open platform.....	32
2.4.3 มาตรฐานหน่วยวัดและการบวกพิกัด .....	32
2.4.4 Robot Operating System.....	33
<b>บทที่ 3 การดำเนินงานวิจัย.....</b>	<b>34</b>
3.1 แผนการดำเนินงาน .....	34

## สารบัญ (ต่อ)

เรื่อง	หน้า
3.2 การออกแบบโครงสร้างของหุ่นยนต์ .....	34
3.2.1 โครงสร้างหุ่นยนต์ .....	35
3.2.2 จัดทำขั้นส่วนโครงสร้างและประกอบ.....	37
3.2.3 อุปกรณ์ที่ใช้ในหุ่นยนต์ชีวามโนยด์อุทัย .....	38
3.2.4 การเชื่อมต่อตัวขับเคลื่อนและตัวรับรู้.....	44
3.2.5 การตั้งค่ามอเตอร์.....	45
3.2.6 การเชื่อมต่อบอร์ดประมวลผล .....	48
3.2.7 Dynamic properties .....	49
3.3 การออกแบบโปรแกรมด้วย ROS .....	62
3.3.1 Modelling .....	62
3.3.2 กำหนดพิกัดเฟรมให้กับหุ่นยนต์ชีวามโนยด์.....	65
3.3.3 Box model.....	66
3.3.4 โครงสร้างการติดต่อสื่อสารระหว่าง Node ใน ROS .....	67
3.4 การออกแบบระบบพื้นฐาน .....	72
3.4.1 วางแผนโครงสร้างของระบบพื้นฐาน .....	72
3.4.2 ออกแบบสถาปัตยกรรมของหุ่นยนต์.....	74
3.4.3 จัดทำคู่มือและเอกสารการใช้งาน .....	76
<b>บทที่ 4 ผลการดำเนินงาน .....</b>	<b>77</b>
4.1 โครงสร้างของหุ่นยนต์ .....	77
4.1.1 การออกแบบขา.....	78
4.1.2 การออกแบบเท้า .....	85
4.1.3 การออกแบบโครงสร้างลำตัว.....	90
4.1.4 Engineer drawing .....	94
4.2 การออกแบบโปรแกรมด้วย ROS .....	95
4.2.1 รับคำจากมอเตอร์แล้วมาแสดงผลใน RViz .....	96
4.2.2 Simulation Gazebo .....	97

## สารบัญ (ต่อ)

เรื่อง	หน้า
4.3 การออกแบบระบบพื้นฐาน .....	98
4.3.1 GitHub ของหุ่นยนต์อุทัย .....	98
4.3.2 ตัวอย่างเฟรมของหุ่นยนต์อุทัย .....	99
4.3.3 ตัวอย่างเอกสารข้อมูลของหุ่นยนต์อุทัย .....	100
4.3.4 ภาพรวมระบบพื้นฐานของหุ่นยนต์อุทัย .....	101
<b>บทที่ 5 สรุปผลการทดลองและข้อเสนอแนะ .....</b>	<b>106</b>
5.1 การออกแบบโครงสร้างของหุ่นยนต์ .....	106
5.2 การออกแบบโปรแกรมด้วย ROS .....	106
5.3 การออกแบบระบบพื้นฐาน .....	106
5.4 สรุปภาพรวม .....	106
<b>ประวัติผู้เขียน .....</b>	<b>107</b>

## บทที่ 1

### บทนำ

#### 1.1 ที่มาและความสำคัญ

หุ่นยนต์อิวามานอยด์เป็นหุ่นยนต์ที่สร้างขึ้นเพื่อเลียนแบบสรีระร่างกายของมนุษย์ ซึ่งมีข้อจำนวนมากเพื่อให้มีการเคลื่อนไหวคล้ายมนุษย์ ลักษณะเด่นของหุ่นยนต์อิวามานอยด์คือ การเคลื่อนที่ด้วยขาสองข้าง ซึ่งทำให้หุ่นยนต์สามารถเคลื่อนที่ได้ อย่างคล่องแคล่วในทุกสภาพพื้นผิว ทั้งทางเรียบ ทางชันและหีบหัก ระดับ<sup>1</sup> ซึ่งนั่นทำให้หุ่นยนต์ที่เดินสองขาแตกต่างจากหุ่นยนต์ที่เคลื่อนที่ด้วยล้อ ด้วยโครงสร้างของหุ่นยนต์ที่คล้ายมนุษย์นั้นเอง จึงทำให้หุ่นยนต์อิวามานอยด์สามารถทำงานได้หลากหลายและยืดหยุ่น สามารถที่จะใช้อุปกรณ์ทั่วไปที่ถูกออกแบบขึ้นมาเพื่อใช้กับมนุษย์ได้ ซึ่งหมายความว่าในอนาคตนั้นหุ่นยนต์อิวามานอยด์สามารถที่จะทำงานทดแทนแรงงานของมนุษย์ได้<sup>2</sup> งานที่หุ่นยนต์อิวามานอยด์จะเข้ามาทดแทนแรงงานของมนุษย์นั้น จะเป็นงานที่ต้องทำซ้ำๆ จนเกินความเมื่อยล้า งานที่อยู่ในพื้นที่อันตรายหรือที่เสี่ยงต่อการเกิดอุบัติเหตุ

สถาบันวิจัยหลายแห่งทั่วโลกกำลังให้ความสนใจสนับสนุนด้านการศึกษาวิจัยและพัฒนาหุ่นยนต์อิวามานอยด์ เพื่อให้ทำการกิจกรรมต่างๆ ยกตัวอย่างเช่น DARPA Robotics Challenge (DRC)<sup>3</sup> เป็นรายการแข่งขันหุ่นยนต์กึ่งอัตโนมัติเพื่อทำการกิจกรรมในสถานการณ์ภัยพิบัติที่อันตราย ซึ่งสถาบันวิจัยหุ่นยนต์ทั่วโลกได้ส่งหุ่นยนต์อิวามานอยด์ของตนเข้าร่วมการแข่งขัน ในปัจจุบันได้มีการพัฒนาหุ่นยนต์อิวามานอยด์ขึ้นมาหลากหลายตัวเช่น ASIMO, HRP-3, LOLA และ WATHLITE-1 การพัฒนาหุ่นยนต์อิวามานอยด์นั้นได้ก่อให้เกิดงานศึกษาวิจัย และทดสอบต่อยอด ต่างๆ มากมาย เช่น การวางแผนการเดิน การเดินแบบสติติ การเดินแบบพลวัต การติดต่อสื่อสารของระบบ การมองเห็นและการประมวลผลภาพ การพูดคุยกับมนุษย์ ปัญญาประดิษฐ์ ฯลฯ ซึ่งงานวิจัยเหล่านี้ สามารถที่จะนำไปประยุกต์ใช้กับระบบหุ่นยนต์ระบบอื่นๆ ได้ แม้ว่าจะมีการพัฒนาหุ่นยนต์อิวามานอยด์มากรามาก แล้ว แต่การเริ่มต้นทำงานวิจัยที่มีความเกี่ยวข้องกับหุ่นยนต์อิวามานอยด์นั้น ต้องใช้ความรู้ความสามารถ เครื่องมือ ระยะเวลา งบประมาณ และ ความพยายามเป็นอย่างมาก การสร้างหุ่นยนต์อิวามานอยด์ขึ้นมาใหม่นั้นต้องใช้งบประมาณสูง ดังนั้นการสร้างระบบจำลองของหุ่นยนต์อิวามานอยด์ขึ้นมาเป็นระบบพื้นฐาน ให้มีความพร้อมสำหรับการพัฒนาต่อยอดแก้ไขศึกษาหรือนักวิจัย จะช่วยประหยัดเวลาและงบประมาณที่ต้องใช้เดือยอย่างมาก ซึ่งนั่นหมายความว่านักวิจัยจะสามารถทำงานวิจัยได้อย่างมีประสิทธิภาพมากขึ้น

ในงานวิจัยนี้เป็นการออกแบบหุ่นยนต์อิวามานอยด์และพัฒนาระบบที่พื้นฐานของหุ่นยนต์อิวามานอยด์ สำหรับให้นักศึกษาหรือนักวิจัยสามารถพัฒนาต่อยอดได้ โดยหุ่นยนต์อิวามานอยด์ที่ออกแบบมานั้น สามารถที่จะปรับปรุงแก้ไข ดัดแปลงได้จ่าย ตัวโครงสร้างจะใช้เป็น พลาสติก PLA ที่สามารถขึ้นรูปได้ โดยการใช้เครื่องพิมพ์สามมิติ มีเซนเซอร์ตรวจการสัมผัสพื้นที่ฝ่าเท้าของหุ่นยนต์ มีเซนเซอร์สำหรับการวัดมุมเอียง ที่ถูกตัวของหุ่นยนต์ และเพื่อที่จะทำให้ง่ายต่อการศึกษาทำความเข้าใจ บำรุงรักษา จึงได้มีการจัดทำคู่มือและเอกสารวิธีการใช้งานอย่างชัดเจน โดยจะเก็บในรูปแบบของเอกสารออนไลน์

<sup>1</sup>การเคลื่อนที่ของหุ่นยนต์รูปแบบต่างๆ รายวัสดุสนับสนุน

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

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

## 1.2 วัตถุประสงค์

ผู้วิจัยมีวัตถุประสงค์ในการทำวิทยานิพนธ์เกี่ยวกับหุ่นยนต์ชีวามโนยด์แพลตฟอร์มนี้ขึ้นมา กี เพื่อที่จะออกแบบโครงสร้างของหุ่นยนต์ชีวามโนยด์ที่สามารถแก้ไขปรับเปลี่ยนได้ง่าย พัฒนาระบบพื้นฐาน ระบบจำลอง สำหรับหุ่นยนต์ชีวามโนยด์เพื่อใช้ในการศึกษาวิจัย รวบรวมเครื่องมือที่เป็นประโยชน์ต่อการพัฒนาหุ่นยนต์ และจัดทำเอกสารออนไลน์ ให้บุคคลที่สนใจสามารถเข้ามาศึกษาได้

## 1.3 ประโยชน์ที่คาดว่าจะได้รับ

- 1 มีต้นแบบหุ่นยนต์ชีวามโนยด์สำหรับใช้ในงานวิจัยแขนงต่างๆ
- 2 มีระบบพื้นฐานสำหรับพัฒนาหุ่นยนต์ชีวามโนยด์รุ่นใหม่ในสถาบัน
- 3 มีระบบจำลองสำหรับจำลองการทำงานของหุ่นยนต์ชีวามโนยด์
- 4 มีแหล่งรวมเครื่องมือสำหรับการพัฒนาหุ่นยนต์
- 5 มีคู่มือ เอกสาร วิธีการใช้งาน และรายละเอียดของหุ่นยนต์สำหรับพัฒนาต่อยอด

## 1.4 ขอบเขตการดำเนินงาน

- 1 ใช้ ROS เป็นกรอบการทำงานสำหรับพัฒนาระบบพื้นฐาน
- 2 ออกแบบโครงสร้างใหม่มีความแข็งแรง สามารถรับน้ำหนักอุปกรณ์ที่ติดตั้งอยู่บนตัวหุ่นยนต์ได้
- 3 น้ำหนักของหุ่นยนต์รวมกันทั้งตัว ไม่เกิน 5 กิโลกรัม
- 4 ใช้ Solidworks 3D เป็นโปรแกรมสำหรับออกแบบโครงสร้าง และคำนวณ
- 5 หุ่นยนต์มีความสูงไม่ต่ำกว่า 100 เซนติเมตร และสูงไม่เกิน 120 เซนติเมตร
- 6 หุ่นยนต์มี 2 แขน 2 ขา มีองศาอิสระของขาข้างละ 6 และแขนข้างละ 2 องศาอิสระ
- 7 หุ่นยนต์สามารถทำงานได้ภายในสภาพแวดล้อมแบบบีด
- 8 หุ่นยนต์ใช้พลังงานจากแหล่งจ่ายพลังงานที่มีขนาดแรงดันไฟฟ้า 12 โวลต์
- 9 หุ่นยนต์ใช้ตัวขับเคลื่อนแบบดิจิตอลสำหรับแต่ละข้อต่อเป็น Dynamixel Digital Servo
- 10 ใช้ Gazebo สำหรับจำลองระบบของหุ่นยนต์
- 11 ติดตั้งเซนเซอร์วัดการกด (Ground contact) ที่ฝ่าเท้าของหุ่นยนต์
- 12 ติดตั้งเซนเซอร์หน่วยวัดความเร็ว (IMU) ที่บริเวณลำตัวของหุ่นยนต์
- 13 จัดทำคู่มือ เอกสารการใช้งาน และรายละเอียดส่วนประกอบของหุ่นยนต์

## 1.5 ภาพรวมของระบบและขั้นตอนการดำเนินงาน

งานวิจัยนี้การดำเนินงานวิจัยถูกแบ่งออกเป็นสามส่วน คือ ส่วนที่หนึ่งส่วนโครงสร้างของหุ่นยนต์ชีวามนอยด์ เป็นส่วนที่ทำในส่วนของการขึ้นรูปชิ้นงาน ออกแบบโมเดลสามมิติ รวมไปถึงระบบอิเล็กทรอนิกส์ ติดตั้งบอร์ดและเซนเซอร์ไวตามจุดต่างๆ เพื่อสร้างโครงสร้างของหุ่นยนต์ให้สามารถรองรับการเดินได้ ส่วนที่สองส่วนโปรแกรมของหุ่นยนต์ชีวามนอยด์ เป็นส่วนที่ทำในส่วนของการสั่งการตัวขับเคลื่อนต่างๆ อันค่าสถานะเซนเซอร์จากคอนโโอลเลอร์ รวมไปถึงระบบจำลองการทำงานของหุ่นยนต์ และส่วนที่สามส่วนการออกแบบระบบพื้นฐานเพื่อการพัฒนาต่ออยอด ส่วนนี้จะเป็นส่วนที่ทำให้ผู้ที่จะมาวิจัยต่ออยอดสามารถทำงานได้ง่ายขึ้น จัดการเอกสารคู่มือ การใช้งานต่างๆให้เป็นระบบระเบียบ สามารถแยกขั้นตอนการทำงานของแต่ละส่วนออกเป็นข้อดังนี้

### ศึกษาค้นคว้าเอกสารและงานวิจัยที่เกี่ยวข้อง

- ศึกษาเกี่ยวกับส่วนประกอบของหุ่นยนต์ชีวามนอยด์
- ศึกษาทฤษฎีที่เกี่ยวข้องกับของมนุษย์
- ศึกษาทฤษฎีที่เกี่ยวข้องกับหุ่นยนต์ชีวามนอยด์
- ศึกษาความแตกต่างระหว่างมนุษย์กับหุ่นยนต์ชีวามนอยด์
- ศึกษาวิธีการและวัสดุที่ใช้ในการสร้างหุ่นยนต์
- ศึกษาระบบที่ใช้ในการจำลองการทำงานของหุ่นยนต์
- ศึกษาระบบที่ใช้ในการจำลองการทำงานของหุ่นยนต์
- ศึกษาการใช้งาน ROS พื้นฐาน

### 1) ส่วนโครงสร้างของหุ่นยนต์ชีวามนอยด์

- ออกแบบโครงสร้างของหุ่นยนต์ชีวามนอยด์
- จัดสร้างโครงสร้างหุ่นยนต์ชีวามนอยด์
- ทดสอบการทำงานของหุ่นยนต์ชีวามนอยด์

### 2) ส่วนโปรแกรมของหุ่นยนต์ชีวามนอยด์

- ออกแบบโปรแกรมของหุ่นยนต์ชีวามนอยด์
- สร้างโปรแกรมส่วนล่างสำหรับหุ่นยนต์ชีวามนอยด์
- สร้างโปรแกรมส่วนบนสำหรับหุ่นยนต์ชีวามนอยด์
- ทดสอบการทำงานของโปรแกรม

### 3) ส่วนการออกแบบระบบพื้นฐานเพื่อการพัฒนาต่ออยอด

- ติดตั้งระบบ
- วางระบบพื้นฐาน
- รวบรวมเครื่องมือที่เป็นประโยชน์
- จัดทำคู่มือ

## บทที่ 2

### ทฤษฎีและหลักการ

#### 2.1 ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

##### 2.1.1 หุ่นยนต์อิริยาบถอยด์

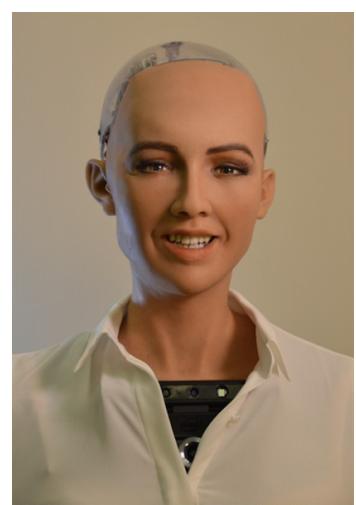
หุ่นยนต์อิริยาบถอยด์ คือ หุ่นยนต์ที่ลูกสร้างขึ้นมาให้มีรูปร่างคล้ายคลึงกับสรีระโครงสร้างของมนุษย์ มักถูกออกแบบขึ้นมาเพื่อจุดประสงค์เฉพาะอย่าง เช่น การใช้เครื่องมือต่างๆของมนุษย์ การอยู่ในสภาพแวดล้อมของมนุษย์ การศึกษาการเคลื่อนไหวของร่ายกายมนุษย์ หรือเพื่อวัตถุประสงค์อื่นๆ โดยทั่วไปแล้ว หุ่นยนต์อิริยาบถอยด์ประกอบไปด้วย 4 ส่วนคือ หัว ลำตัว เแขน และขา แต่การสร้างหุ่นยนต์อิริยาบถอยด์นั้นก็ไม่จำเป็นที่จะต้องมีส่วนประกอบทุกส่วนดังที่กล่าวไว้ ในบางครั้งอาจมีเพียงแค่ส่วนบนเท่านั้น ดังรูปที่ 2.1ก หุ่นยนต์นะโมจากสถาบันวิทยาการหุ่นยนต์ภาคสนาม เป็นหุ่นยนต์ที่มีส่วนบนเหมือนมนุษย์ แต่มีส่วนล่างเป็นล้อ หรือหุ่นยนต์อิริยาบถอยด์ที่มีเพียงแค่ส่วนล่าง ดังรูปที่ 2.1ข หุ่นยนต์ส้มจุก เป็นหุ่นยนต์อิริยาบถอยด์ที่มีเพียงแค่ส่วนขาเท่านั้น หรือหุ่นยนต์อิริยาบถอยด์ที่มีเพียงใบหน้าเหมือนมนุษย์ ดังรูปที่ 2.1ค หุ่นยนต์โซเฟีย เป็นแอนดรอยด์ที่มีหน้าตาคล้ายมนุษย์มาก มีตา มีปาก สามารถพูดปฏิสัมพันธ์กับมนุษย์ได้



(ก) หุ่นยนต์ประชาสัมพันธ์โนม



(ข) หุ่นยนต์เดินสองขาส้มจุก



(ค) หุ่นยนต์แอนดรอยด์โซเฟีย

รูปที่ 2.1: แสดงความแตกต่างของหุ่นยนต์อิริยาบถอยด์แต่ละประเภท

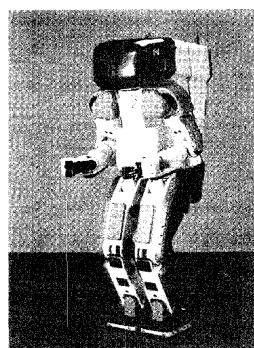
งานวิจัยทางด้านหุ่นยนต์อิริยาบถอยด์จำกัดจิตใจด้วยจิตใจ ปัจจุบันมีการพัฒนาความสามารถของการเดินของหุ่นยนต์ เช่น เริ่มต้นจากแรกสุดจะเป็นการพัฒนาให้หุ่นยนต์สามารถเดินหน้าได้ ต่อมาเกิดเพิ่มความสามารถให้หุ่นยนต์สามารถเดินบนพื้นเรียบ พื้นชุ่มชื้น เช่นเดินเลี้ยวซ้ายขวา เดินขึ้นลงบันได ฯลฯ เป็นต้น นอกจากนี้ยังมีการพัฒนาปรับปรุงสมดุลของการเดินแบบสองขาอีกด้วย สมดุลของการเดินสามารถแบ่งได้สองแบบหลัก คือ การเดินแบบสมดุลสถิติ และการเดินแบบสมดุลพลวัต งานในยุคแรกนั้นจะพัฒนาให้เดินได้แบบสมดุลสถิติ ต่อมาเป็นสมดุลกึ่งพลวัต และเป็นสมดุลพลวัต การพัฒนาตัวควบคุมการเดินของหุ่นยนต์ จำเป็นที่จะต้องใช้ความรู้ทางด้านกลศาสตร์ค่อนข้างมาก มีการใช้สมการที่มีความซับซ้อน

Zheng และคณะ (1988) พัฒนาหุ่นยนต์สองขาที่สามารถเดินบนพื้นราบได้ ให้สามารถเดินต่อเนื่องไปบนพื้นอุปกรณ์ได้ด้วย พื้นอุปกรณ์ที่ใช้มีลักษณะเป็นพื้นอุปกรณ์ที่ใช้ในงานนี้มีข้อต่อสะโพก (hip), ข้อเท้า (ankle) และลำตัว (torso) มีเซนเซอร์วัดแรงดึง (force sensor) ติดตั้งอยู่ที่ปลายเท้าและสันเท้าแต่ละข้างเพื่อใช้วัดตำแหน่งของน้ำหนักโดยรวม (center of gravity) ของหุ่นยนต์ การเดินของงานวิจัยจะพิจารณาเฉพาะการเดินในแนวหน้าหลัง โดยมีหลักการคือ การเดินบนพื้นอุปกรณ์ที่หุ่นยนต์ยังเดินในท่าทางเหมือนกับตอนที่เดินบนพื้นราบจะทำให้น้ำหนักโดยรวมของหุ่นยนต์เลื่อนไปข้างหลัง ดังนั้นการที่หุ่นยนต์ขับลำตัวไปด้านหน้าจะทำให้น้ำหนักโดยรวมของหุ่นยนต์กลับมาอยู่ตรงกลางของพื้นที่รับน้ำหนักเหมือนเดิม ซึ่งจะทำให้หุ่นยนต์มีความสมดุลได้ ดังนั้นข้อมูลที่ได้จากหน่วยวัดแรงดึงที่เท้าจะถูกนำมาคำนวณตลอดการเดินเพื่อให้ในการปรับเปลี่ยนมุกการขับของลำตัว การเดินบนพื้นราบเป็นแบบสมดุลสถิตและการเดินบนพื้นอุปกรณ์ยังคงเป็นแบบสมดุลสถิตเช่นกัน

Inaba<sup>1</sup> และคณะ (1995) สร้างหุ่นยนต์เลียนแบบลิง (ape-like biped) ประกอบด้วยสองมือและสองขา มีการเดินแบบสมดุลสถิต งานวิจัยนี้มีความคิดว่าจากการทำให้หุ่นยนต์สองขาเดินได้โดยไม่ล้มแล้ว ควรจะทำหุ่นยนต์ที่สามารถลุกขึ้นเองได้หลังจากที่ล้มแล้วด้วย ดังนั้นในงานนี้ หุ่นยนต์ถูกพัฒนาให้สามารถเดิน เมื่อล้มแล้ว ก็สามารถพลิกตัวและลุกขึ้นมาเดินให้ได้

Kun และ Miller<sup>2</sup> (1996) ได้นำโครงข่ายประสาทเทียม มาประยุกต์ใช้ในการปรับเปลี่ยนท่าทางการเดิน โดยอัตโนมัติของหุ่นยนต์สองขา การที่หุ่นยนต์สามารถปรับเปลี่ยนท่าทางได้โดยอัตโนมัตินี้มีประโยชน์ทำให้หุ่นยนต์เดินได้บนพื้นผิวหลากหลายลักษณะมากขึ้น ในงานนี้พิจารณาทั้งสมดุลในแนวหน้าหลัง (sigittal plane) และแนวซ้ายขวา (frontal plane) และการเดินของหุ่นยนต์เป็นแบบสมดุลพลวัต หลักการทำงานประกอบด้วยตัวสร้างท่าทางการเดินหนึ่งตัว และตัวปรับท่าทางการเดินทั้งแนวหน้าหลังและซ้ายขวาอีกหนึ่งตัว โดยค่าการปรับเปลี่ยนนั้นจะได้มาจากการเดินที่เท้า ความยาวการก้าวเท้า ความสูงของการยกเท้า เป็นต้น นอกจากนี้ในปัจจุบัน ทั้งสองได้ใช้หลักการที่ใช้ในงานนี้ไปใช้กับการเดินของหุ่นยนต์อีกด้วย (Kun and Miller, 1997)

Hirai<sup>3</sup> และคณะ (1998) พัฒนาหุ่นยนต์ชิวามานอยด์ ซึ่งตัวหุ่นยนต์มีความคล้ายมนุษย์มาก สามารถเดินได้อย่างราบรื่นคล้ายมนุษย์มากที่สุด เช่น สามารถเดินได้ในพื้นผิวนิ่มต่างๆ เดินได้บนพื้นอุปกรณ์ที่มีความลึก 4.7 กิโลเมตรต่อชั่วโมง หุ่นยนต์ประกอบไปด้วย แขนห้างละ 9 องศา อิสระ ขาห้างละ 6 องศา อิสระ ที่บริเวณหัวมีกล้องติดตั้งอยู่ 4 ตัว นอกจากนี้ยังมีอุปกรณ์ที่ใช้ในการรักษาสมดุลอีก 4 ตัว ที่ติดตั้งบริเวณลำตัว และ Force sensor ที่ติดที่เท้าทั้งสองข้าง



รูปที่ 2.2: Honda asimo โดย Kazou Hirai

<sup>1</sup>Yuki Asano\*, Kei Okada and Masayuki Inaba

<sup>2</sup>Modelling of Walking Humanoid Robot With Capability of Floor Detection and Dynamic Balancing Using Colored Petri Net, Saeid Pashazadeh and Saeed Saeedvand

<sup>3</sup>Kazuo Hirai, (1999) "The Honda humanoid robot: development and future perspective", Industrial Robot: An International Journal, Vol. 26 Issue: 4, pp.260-266, <https://doi.org/10.1108/01439919910277431>

องค์ประกอบของหุ่นยนต์ที่ไว้ประจำประกอบไปด้วยกระบวนการตอบสนองต่างๆที่เป็นระบบ ซึ่งความสามารถจำแนกการอกรับเป็นส่วนหลักๆได้สามส่วนคือ ส่วนการรับรู้ ส่วนการประมวลผล และส่วนการขับเคลื่อน ทั้งหมด เมื่อนำมารวมเข้าด้วยกันแล้ว เราก็สามารถที่จะควบคุมการทำงานของหุ่นยนต์ขึ้มมาอยู่ได้

### **การรับรู้ของหุ่นยนต์ขึ้มมาอยู่**

การรับรู้ของหุ่นยนต์ขึ้มมาอยู่นั้นมีความยากมากกว่าหุ่นยนต์ชนิดอื่นๆเนื่องจาก หุ่นยนต์ขึ้มมาอยู่ได้ ก็ต้องการนำก้านต่อหดายๆขึ้นเข้ามาเชื่อมต่อกันด้วยข้อต่อ ทำให้หุ่นยนต์ขึ้มมาอยู่สามารถเคลื่อนไหวเป็นท่าทางต่างๆได้ และไม่มีส่วนใดถูกตึงยืดติดกับพื้นโลก ซึ่งทำให้เราไม่สามารถที่จะอ้างอิงท่าทางของหุ่นยนต์ได้ จึงจำเป็นที่จะต้องเพิ่มส่วนของการรับรู้เข้าไปเพื่อช่วยแก้ปัญหาในส่วนนี้ เช่นเซอร์วิสที่เพิ่มเข้าไปมีหลากหลายชนิด และแต่ละชนิดก็ทำหน้าที่แตกต่างกัน ยกตัวอย่างเช่น เช่นเซอร์วิสโอนโคดเดอร์ที่ใช้สำหรับอ่านสถานะตำแหน่งและความเร็วของข้อต่อได้ เช่นเซอร์วิสวัดความเมื่อยที่ใช้สำหรับหามุมเอียงของตัวหุ่นยนต์ และเซนเซอร์วัดแรงที่ผ่าเท้าที่จะช่วยในการบอกว่าเท้าของหุ่นยนต์สัมผัสพื้นหรือไม่ เป็นต้น

### **การประมวลผลของหุ่นยนต์ขึ้มมาอยู่**

ในปัจจุบันนี้หน่วยประมวลผลของหุ่นยนต์ขึ้มมาอยู่มีประสิทธิภาพเพียงพอต่อการควบคุมหุ่นยนต์ขึ้มมาอยู่แบบเรียลไทม์ได้ การประมวลผลนั้นสามารถที่จะแบ่งออกเป็นหดายๆส่วนได้ ยกตัวอย่างเช่น

**หุ่นยนต์ขึ้มมาอยู่ Thormang** ได้แบ่งตัวประมวลผลเป็นตัวประมวลควบคุมสำหรับสั่งการตัวขับเคลื่อน ตัวประมวลควบคุมสำหรับอ่านสถานะตัวรับรู้ และตัวประมวลควบคุมภายนอกสำหรับคำนวนท่าทางการเดินและการวางแผนเท้า

**หุ่นยนต์ขึ้มมาอยู่ Robotis OP3** ได้แบ่งตัวประมวลผลเป็นตัวประมวลผลระดับสูงสำหรับคำนวนท่าทางการเคลื่อนไหว และตัวประมวลผลระดับล่างสำหรับสั่งการตัวขับเคลื่อนและอ่านสถานะตัวรับรู้

**หุ่นยนต์ขึ้มมาอยู่ Poppy** ไม่ได้แบ่งตัวประมวลผล แต่ใช้เพียงจัวเดียวในการสั่งการตัวขับเคลื่อน อ่านสถานะตัวรับรู้ และประมวลผลการคำนวนทั้งหมด

นอกจากการประมวลผลและควบคุมแล้ว ยังรวมไปถึงหน่วยแสดงผลที่สามารถนำค่าสถานะต่างๆจากหุ่นยนต์ขึ้มมาอยู่ออกไปสร้างเป็นกราฟ หรือแบบจำลองสามมิติได้อีกด้วย

### **การขับเคลื่อนของหุ่นยนต์ขึ้มมาอยู่**

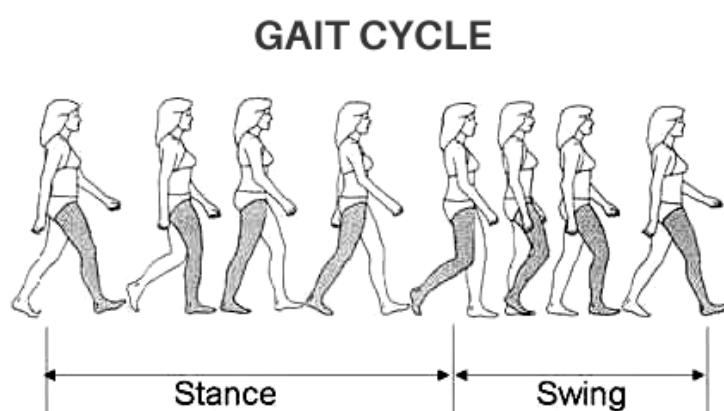
การที่หุ่นยนต์ขึ้มมาอยู่สามารถที่จะทำท่าทางต่างๆได้นั้น เกิดจากการเปลี่ยนแปลงตำแหน่งของข้อต่อ ดังนั้นการที่หุ่นยนต์จะขับเคลื่อนข้อต่อได้นั้น ก็จึงจำเป็นที่จะต้องมีตัวขับเคลื่อนที่ข้อต่อ โดยที่ไว้แล้วเราจะติดมอเตอร์เซอร์วิสไว้ที่ตำแหน่งเดียวกับข้อต่อเลย เหตุผลที่นิยมใช้มอเตอร์เซอร์วิสเนื่องจาก เซอร์วิสสามารถที่จะควบคุมตำแหน่งและความเร็วได้

## 2.1.2 ทฤษฎีเกี่ยวกับมนุษย์

### 2.1.2.1 การวิเคราะห์การเดินของมนุษย์

การเคลื่อนที่ของหุ่นยนต์ชีวามโนยด์นั้นจะเลียนแบบจากการเดินของมนุษย์ ดังนั้นการวิเคราะห์ลักษณะการเดินของมนุษย์ จะเป็นการศึกษาเพื่อทำความเข้าใจถึงธรรมชาติการเดิน ก่อนนำไปทำการออกแบบกลไกทางกลและระบบควบคุมของหุ่นยนต์ชีวามโนยด์ การก้าวเดินของมนุษย์โดยปกติแล้ว จะมีลักษณะเป็นวัฏจักร วนซ้ำไปเรื่อยๆ ในทิศทางที่ต้องการจนกว่าจะทำการหยุดเดิน การทรงตัวในระหว่างการยืนหรือการเดินนั้น เป็นไปตามสัญชาตญาณซึ่งเกิดจากการรักษาความสมดุลของร่างกายบนที่<sup>4</sup> ส่งสัญญาณผ่านเส้นประสาทไปยังกล้ามเนื้อส่วนต่างๆ ที่ทำหน้าที่ให้เกิดการเคลื่อนที่

การเคลื่อนที่ของมนุษย์ในการเดินไปข้างหน้าสามารถแบ่งออกเป็นช่วงๆ ดังนี้



รูปที่ 2.3: วัฏจักรการเดินของมนุษย์

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

<sup>4</sup>Rose, J. and Gamble, J., 1993, Human Walking, Williams & Wilkins, Philadelphia, pp. 10-44.

### 2.1.2.2 การวิเคราะห์องศาอิสระของมนุษย์

การที่มนุษย์สามารถเคลื่อนที่ได้เป็นผลเนื่องมาจากการเคลื่อนที่ของข้อต่อต่างๆ ซึ่งประกอบไปด้วย ข้อต่อส่วนสะโพก ส่วนหัวเข่า และส่วนข้อเท้า แรงบิดที่เกิดขึ้นของแต่ละข้อต่อมีความสัมพันธ์ต่อกัน ส่งผลให้เกิดเสถียรภาพในการเดินของมนุษย์ เมื่อวิเคราะห์ลักษณะโครงสร้างในแต่ละส่วน พบว่าข้อต่อส่วนสะโพกมีลักษณะเป็นทรงกลม ทำให้ข้อต่อส่วนสะโพกสามารถหมุนได้ 3 องศาอิสระ ส่วนหัวเข่าของมนุษย์มีจุดต่อของข้อที่มีลักษณะเป็นทรงกลม สองลูกประกอบเข้าด้วยกันทำให้การเคลื่อนที่ถูกบังคับให้สามารถเคลื่อนที่ได้เพียง 1 องศาอิสระ ในส่วนของข้อเท้ามีลักษณะการเคลื่อนที่เหมือนสะโพกคือสามารถเคลื่อนที่ได้ 3 องศาอิสระ

จากทั้งหมดที่ได้ทำการวิเคราะห์มาข้างต้นพบว่าในขาหนึ่งข้างของมนุษย์ประกอบด้วย 7 องศาอิสระ ซึ่งส่งผลให้การเคลื่อนที่ของมนุษย์มีความคล่องแคล่วสูง แต่ในทางออกแบบกลไกการเดินและการควบคุม ของหุ่นยนต์สองขาถือว่ามีจำนวนองศาอิสระเกินความจำเป็นในการเคลื่อนที่บนปริภูมิ(space) และยกต่อการควบคุม(underactuated) ดังนั้นการกำหนดจำนวนองศาอิสระเพื่อให้หุ่นยนต์เดินได้สมดุลมนุษย์จึงมีผลในการออกแบบกลไกทางกลและการควบคุมของหุ่นยนต์สองขา

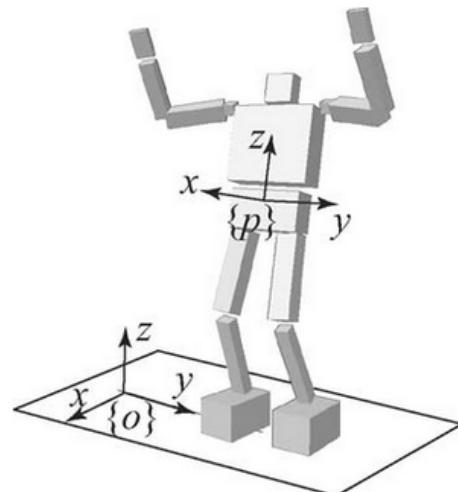
ข้อต่อ	องศาอิสระ	องศาการหมุน	
		สูงสุด	ต่ำสุด
หัว	$\theta_x$	+60	-30
	$\theta_y$	+70	-70
	$\theta_z$	+80	-80
หลัง	$\theta_x$	+30	-30
	$\theta_y$	+55	-55
	$\theta_z$	+45	-45
หัวไห lateral	$\theta_x$	+180	-80
	$\theta_y$	+45	-135
	$\theta_z$	+30	0
ศอก	$\theta_x$	0	-155
สะโพก	$\theta_x$	+120	-40
	$\theta_y$	+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 และได้ใช้ติดต่อโลเซอร์ไวของบริษัท Robotis เป็นตัวขับเคลื่อนข้อต่อ เนื่องจากภายในเซอร์โวมีตัวรับรู้สถานะของตัวเอง และเซอร์โวนี้ถูกออกแบบมาให้สามารถติดตั้ง และสั่งการได้ง่าย

### 2.1.3 ทฤษฎีเกี่ยวกับหุ่นยนต์ชีวภาพอยู่ด้วยกัน

#### 2.1.3.1 ส่วนประกอบของหุ่นยนต์ชีวภาพอยู่ด้วยกัน

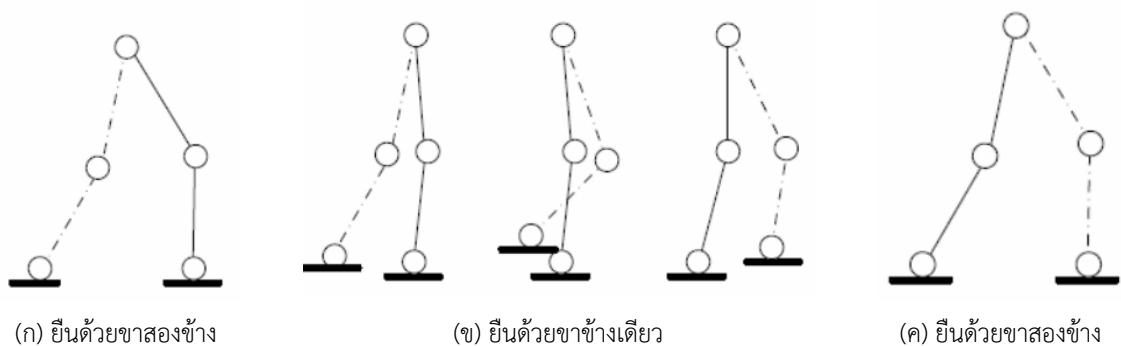


รูปที่ 2.4: ส่วนประกอบของหุ่นยนต์ชีวภาพอยู่ด้วยกัน

หุ่นยนต์ชีวภาพอยู่ด้วยกันต่อห้ายาก้านที่นำมาต่อ กัน ลักษณะโครงสร้างนั้นจะเป็นแบบโซ่อเปิด (Open kinematic chain) และแต่ละก้านต่อจะเชื่อมต่อกันด้วยข้อต่อแบบหมุน เราสามารถแบ่งโครงสร้างของหุ่นยนต์ชีวภาพอยู่ด้วยกันเป็น 2 ส่วน ส่วนแรกคือ ส่วนก้านต่อของลำตัวหุ่นยนต์ (Torso) ซึ่งเราสามารถที่จะรวมไปถึงส่วนแขนกับหัวด้วย และในส่วนที่สองคือ ส่วนก้านต่อของขาหุ่นยนต์ (Legs) ซึ่งเป็นส่วนของหุ่นยนต์ทั้งสองข้างที่สามารถนำไปที่สัมผัสกับพื้นได้ ทั้งสองก้านต่ออนี้ถูกเชื่อมต่อกันด้วยส่วนของสะโพก (Hip) ที่อยู่ระหว่างส่วนลำตัวกับส่วนของขาหุ่นยนต์ ดังรูปที่ 2.4

#### 2.1.3.2 วิธีการเดินของหุ่นยนต์ชีวภาพอยู่ด้วยกัน

วิธีการเดินของหุ่นยนต์ คือ การที่หุ่นยนต์จะต้องมีการถ่ายน้ำหนักไปมาระหว่างเท้าซ้ายและเท้าขวา มีบางช่วงที่น้ำหนักตกลงบนเท้าข้างใดข้างหนึ่งหรือทั้งสองข้างพร้อมกัน สามารถแบ่งออกเป็นช่วงได้สองช่วง คือ ช่วงการยืนด้วยขาข้างเดียว และช่วงการยืนด้วยขาทั้งสองข้าง



รูปที่ 2.5: วิธีการเดินของหุ่นยนต์ชีวภาพอยู่ด้วยกัน

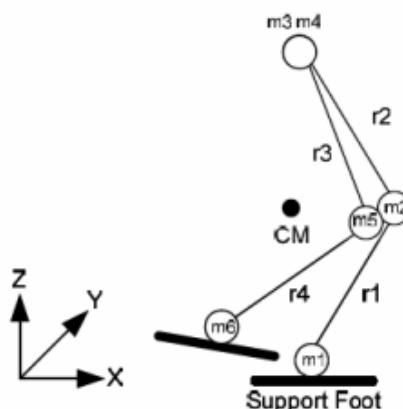
1) การยืนด้วยขาข้างเดียว : เป็นช่วงที่มีเท้าของหุ่นยนต์สัมผัสพื้นเพียงข้างเดียว ส่วนเท้าอีกข้างของหุ่นยนต์จะถูกยกอยู่จากพื้น โดยที่ไม่มีส่วนใดๆของขาข้างนั้นสัมผัสกับพื้นเลย ช่วงนี้จะเกิดขึ้นเมื่อมีการแกะง่ำเท้าจากข้างหลังไปข้างหน้า ดังจากภาพที่ 2.5x

2) การยืนด้วยขาสองข้าง : เป็นช่วงที่เท้าทั้งสองข้างของหุ่นยนต์สัมผัสกับพื้น ช่วงนี้จะเกิดตั้งแต่หุ่นยนต์วางเท้าลงที่สันเท้าและกับพื้น ไปจนถึง ปลายเท้าของขาอีกข้างหลุดออกจากพื้น

การเดินได้โดยไม่ล้มนั้น ตัวหุ่นยนต์จะต้องรักษาสมดุลของการเดินให้ได้ตลอดช่วงเวลาของการเดิน ซึ่งสมดุลของการเดินแบบสองขาสามารถแบ่งตามลักษณะการเดินและการถ่ายน้ำหนักได้เป็น 2 รูปแบบหลัก คือ การเดินแบบสมดุลสถิติ (static balance walking) และ การเดินแบบสมดุลพลวัต (dynamic balance walking)

#### 2.1.3.3 การสร้างและการควบคุมการเดินแบบสมดุลสถิติ

การเดินของหุ่นยนต์ในลักษณะนี้ จุดศูนย์กลางมวลตัวหุ่นยนต์จะไม่มีการเคลื่อนไหวออกนอกบริเวณฐานรับน้ำหนัก (Supporting Area) ตลอดช่วงเวลาการเดิน ไม่ว่าจะเป็นช่วงเวลาที่รับน้ำหนักด้วยเท้าข้างเดียวหรือทั้งสองข้างก็ตาม หมายความว่า โครงสร้างของหุ่นยนต์จะไม่ล้มแน่นอน เนื่องจากการสร้างรูปแบบการเดินด้วยวิธีนี้ จะควบคุมให้ตำแหน่งของจุดรวมมวล (CoM) อยู่ภายใต้พื้นที่ฐานรับน้ำหนักของหุ่นยนต์ตลอดเวลา



รูปที่ 2.6: การควบคุมตำแหน่งของจุดรวมมวลให้อยู่ในพื้นที่ฐาน

ข้อดีของการสร้างและควบคุมการเดินของหุ่นยนต์ด้วยวิธีนี้คือ สามารถสร้างรูปแบบการเดินได้โดยที่มีความซับซ้อนไม่มากนัก สามารถสั่งให้หุ่นยนต์หยุดค้างในท่าทางใดๆก็ได้ตลอดเวลาโดยหุ่นยนต์ไม่ล้ม หุ่นยนต์ที่มีฝ่าเท้าใหญ่จะทำให้ง่ายต่อการก้าวเดินมากขึ้น นอกจากการควบคุมการก้าวขาแล้วอาจเพิ่มการควบคุมส่วนลำตัวเพิ่มเติม เพื่อเป็นการเพิ่มเสถียรภาพในการเดินและการถ่ายน้ำหนัก โดยที่อาจจะมีการเพิ่มเซนเซอร์วัดแรงที่ฝ่าเท้าเพื่อตรวจสอบการกระจายแรงกดที่ฝ่าเท้า เพื่อตรวจสอบว่าตำแหน่งของจุดรวมน้ำหนักอยู่บนพื้นที่ฝ่าเท้าหรือไม่ หรือเพื่อตรวจสอบเสถียรภาพของการเดินเพื่อแก้ไขท่าทางการเดินไม่ให้เกิดการล้ม

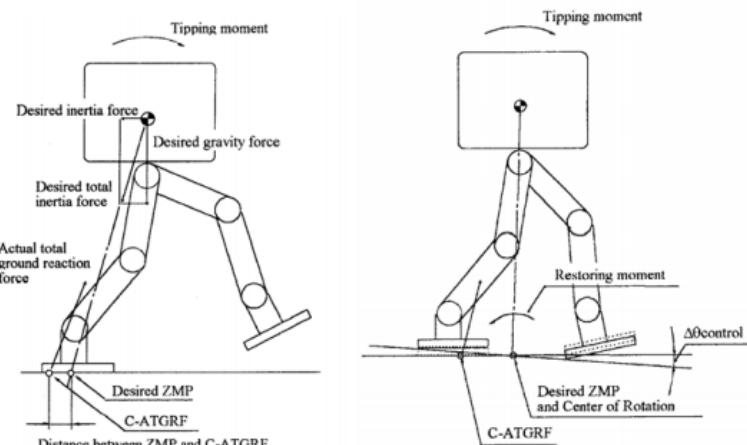
ข้อเสียของการควบคุมการเดินด้วยวิธีนี้คือ หุ่นยนต์จะใช้เวลาในการก้าวเดินมาก ใช้พลังงานในการเดินมากกว่าการเดินแบบสมดุลพลวัต และท่าทางที่ได้จะมีความแตกต่างจากท่าทางการเดินของมนุษย์

#### 2.1.3.4 การสร้างและการควบคุมการเดินแบบสมดุลพลวัต

การสร้างรูปแบบการเดินและควบคุมการเดินในลักษณะนี้ท่าทางการเดินของหุ่นยนต์นั้นจะคล้ายกับการเดินของมนุษย์มากกว่าแบบสถิติ เนื่องจากมีหลักการในการสร้างท่าทางที่เหมือนกับการเดินของมนุษย์ซึ่งมีขั้นตอนดังนี้คือ เอียงตัวไว้ให้ล้มไปในทิศทางที่ต้องการเดิน เมื่อเริ่มเกิดการล้มขึ้นหุ่นยนต์จะเปลี่ยนตำแหน่งการวางเท้าไปยังตำแหน่งใหม่ เพื่อปรับให้โครงสร้างเข้าสู่ภาวะสมดุลอีกรั้ง

โดยธรรมชาติแล้วมนุษย์มีการถ่ายน้ำหนักในขณะที่เคลื่อนที่หรือยืนอยู่กับที่เพื่อรักษาสมดุลของท่าทางนั้นไว้ แต่หากการถ่ายโอนน้ำหนักนั้นเกิดสภาวะไม่สมดุล ร่างกายจะปรับสภาพโดยการเคลื่อนตำแหน่งของเท้าซึ่งเป็นพื้นที่ฐานออกจากเดิมไปยังตำแหน่งใหม่ เพื่อรักษาสมดุลไว้ หลักการดังกล่าวถูกนำมาใช้กับการควบคุมการเดินของหุ่นยนต์อิวามานอยด์ ในขณะที่หุ่นยนต์กำลังเคลื่อนไหว ผลจากแรงเฉือนของการเคลื่อนที่และผลจากแรงดึงดูดของโลกมีผลต่อการเพิ่มและลดความเร่งให้การเดินของหุ่นยนต์ แรงเหล่านี้เรียกว่าแรงเฉือนของการเคลื่อนที่ และเมื่อเท้าหุ่นยนต์สัมผัสกับพื้นจะได้รับผลกระทบของแรงนี้ เรียกว่า แรงปฏิกิริยาจากพื้น

การตัดกันระหว่างแรงปฏิกิริยาจากพื้นและแนวแรงเฉือนรวม ตำแหน่งนั้นหากทำให้โมเมนต์เท่ากับศูนย์เรียกจุดตัดนี้ว่าจุดโมเมนต์ศูนย์ ( $ZMP_{robot}$ ) และจุดที่แรงปฏิกิริยาลงสู่พื้นว่า จุดปฏิกิริยาพื้นฐาน ท่าทางการเดินของหุ่นยนต์จะถูกกำหนดและถูกส่งให้กับชุดควบคุมข้อต่อจุดต่างๆ ของหุ่นยนต์ โดยให้สอดคล้องกับแรงเฉือนรวมที่เกิดขึ้นจากการคำนวณ เรียกว่าแรงเฉือนรวม เป้าหมาย และจุดโมเมนต์ศูนย์ที่ได้จากการคำนวณเรียกว่าจุดโมเมนต์ศูนย์เป้าหมาย ( $ZMP_{target}$ ) เมื่อหุ่นยนต์เกิดสมดุลในขณะที่ทำการเดินได้อย่างสมบูรณ์ แนวแกนของแรงเฉือนรวมเป้าหมายและแรงปฏิกิริยาที่พื้นจะเป็นตำแหน่งเดียวกัน แต่ในขณะที่หุ่นยนต์เดินผ่านพื้นผิวที่มีความชรุขระหรือไม่เรียบตำแหน่งสองจุดดังกล่าว จะไม่ใช่ตำแหน่งเดียวกันทำให้หุ่นยนต์เกิดการล้มได้ แรงที่ทำให้เกิดการล้มนี้เกิดจากตำแหน่งของจุดโมเมนต์ศูนย์และตำแหน่งแรงปฏิกิริยารวมที่พื้นไม่ตรงกัน ซึ่งเป็นสาเหตุหลักที่ทำให้เกิดความไม่สมดุลขึ้น และเมื่อหุ่นยนต์เสียสมดุลระบบที่จะสามารถป้องกันการล้มและทำให้หุ่นยนต์เดินต่อไปได้อย่างต่อเนื่องคือ ระบบควบคุมแรงปฏิกิริยา ระบบควบคุมจุดโมเมนต์ศูนย์ และระบบควบคุมการวางแผนเท้า



รูปที่ 2.7: การควบคุมตำแหน่งของจุดโมเมนต์ศูนย์ให้ตรงกับแรงปฏิกิริยารวม

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

### 2.1.3.5 จุดศูนย์กลางมวลของหุ่นยนต์

หากต้องการให้หุ่นยนต์สามารถที่จะทรงตัวอยู่ได้โดยไม่ล้มนั้น จึงต้องรู้ตำแหน่งจุดศูนย์กลางมวลของหุ่นยนต์ตลอดเวลา และต้องให้จุดศูนย์กลางมวลอยู่ต่ำในบริเวณฐานรับน้ำหนักของหุ่นยนต์โดยหากพื้นที่ที่ฝ่าเท้าสัมผัสกับพื้น วิธีการนี้เป็นวิธีการทางสถิตศาสตร์

#### 2.1.4 ตัวอย่างหุ่นยนต์ชีวามโนยด์

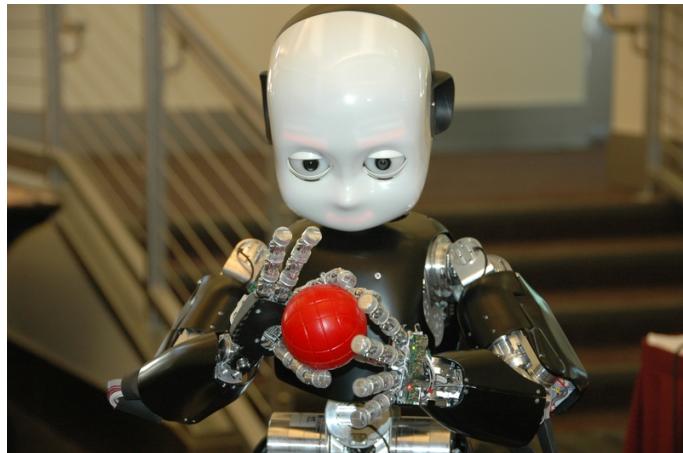
Poppy Humanoid



รูปที่ 2.8: หุ่นยนต์ชีวามโนยด์ปีอปปี้

หุ่นยนต์ชีวามโนยด์ปีอปปี้ ถูกสร้างขึ้นมาเพื่อใช้ในงานศิลปะ การวิจัยและการศึกษาโดยเฉพาะ หุ่นยนต์ปีอปปี้ประกอบด้วยส่วนของฮาร์ดแวร์และซอฟแวร์ที่เปิดเป็นโอเพนซอร์ซให้ผู้ที่สนใจสามารถเข้ามาศึกษาได้ โปรแกรมของหุ่นยนต์ใช้โมดูลที่มีชื่อว่า Pypot ที่เป็นส่วนเสริมของภาษา Python ในการพัฒนาซอฟแวร์ ทุกคนสามารถเข้าถึงข้อมูลเชิงเทคนิคของหุ่นยนต์ชีวามโนยด์ปีอปปี้ได้ เช่น ส่วนรายละเอียดการทำงาน คลิปวีดีโอสอน การประกอบ การใช้ระบบจำลอง และการพัฒนาต่างๆผ่านทางเว็บไซต์ <http://www.poppy-project.org> หุ่นยนต์ปีอปปี้มีส่วนของโครงสร้างที่ผลิตมาจากพลาสติก PLA และ ABS โดยใช้เทคนิคการฉีดรูปด้วยเครื่องพิมพ์สามมิติ ตัวขับเคลื่อนข้อต่อต่างๆใช้เป็น Dynamixel Digital Servo และควบคุมคำสั่งของตัวขับเคลื่อนด้วยคอมพิวเตอร์ขนาดเล็ก Odroid UX4 ใช้ระบบปฏิบัติการ Ubuntu 14.04 ตัวของหุ่นยนต์มีความสูง 83 เซนติเมตร น้ำหนัก 3.5 กิโลกรัม ใช้เซนเซอร์วัดมุม เอียง เป็น IMU ที่มีองศาอิสระ เท่ากับ 9 องศาอิสระ ในการควบคุมเส้นสายภาพในการเดินของตัวเอง มีองศาอิสระหรือจำนวนตัวขับเคลื่อนทั้งหมด 25 องศา ประกอบไปด้วย ขาข้างละ 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 สำหรับส่วนที่ต้องรับภาระความเครียดน้อย 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 เพื่อทำให้ใช้งานง่ายลง ใช้เส้นเอ็นในการส่งถ่ายแรงขับเคลื่อนไปยังส่วนของข้อต่อส่วนมือและขา นิ้วของหุ่นยนต์ถูกร้อยด้วยสายเคเบิลเคลือบ Teflon อยู่ภายใต้ และความต้านทานของสายเคเบิลที่ต้องการ ใช้เซนเซอร์วัดมุมของข้อต่อแต่ละตัวใช้การออกแบบให้มี Hall-effect ติดอยู่ ช่วยในการอ่านค่าของตำแหน่งและความเร็วที่เกิดขึ้นที่ข้อต่อนั้น หุ่นยนต์ไอคัพมีองศาสือิสระรวมกันทั้งหมด 53 องศาสือิสระ ประกอบไปด้วย แขนข้างละ 7 องศาสือิสระ มือข้างละ 9 องศาสือิสระ หัว 6 องศาสือิสระ ลำตัว 3 องศาสือิสระ และขาข้างละ 6 องศาสือิสระ ในส่วนของหัวจะประกอบไปด้วย กล้องสองตัวเพื่อทำการสืบสานและรับเสียงจากสภาพแวดล้อมภายนอก และไฟแสดงอารมณ์บริเวณปากและคิ้ว หุ่นยนต์นี้ไม่ได้ถูกออกแบบให้มีการทำงานเป็นแบบอัตโนมัติ ซึ่งก็คือตัวหุ่นยนต์นี้ไม่มีแบตเตอรี่ภายในตัว แต่ใช้แหล่งพลังงานจากการส่งเข้าไปผ่านสายเคเบิล และเชื่อมต่อกับอินเทอร์เน็ตผ่านสายแลน (LAN)

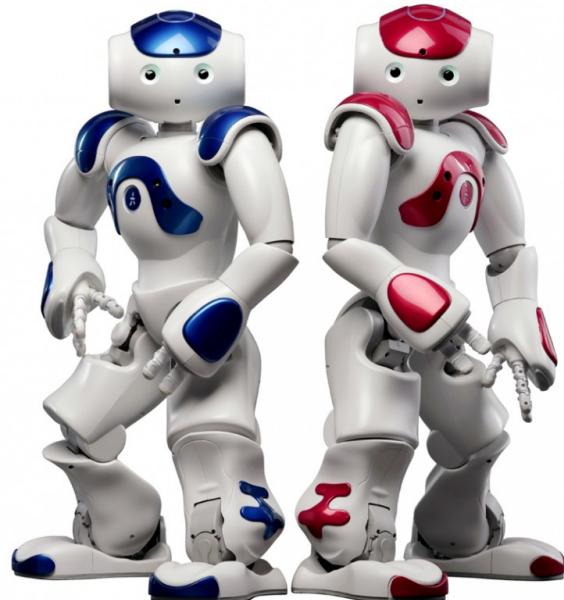
### Darwin-OP Humanoid



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

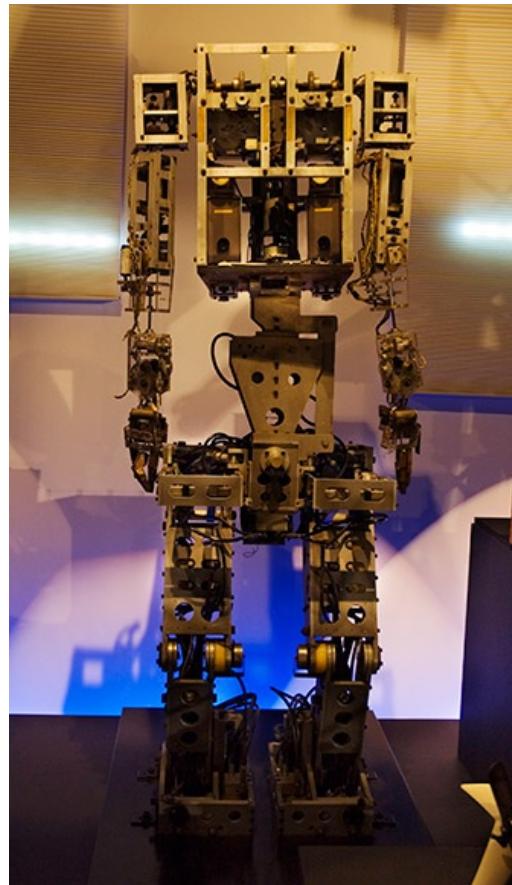
### Nao Humanoid



รูปที่ 2.11: หุ่นยนต์อิวามานอยด์นาโอะ

หุ่นยนต์อิวามานอยด์นาโอะ เป็นหุ่นยนต์อิวามานอยด์ขนาดกลาง ถูกผลิตมาจากประเทศฝรั่งเศษ พัฒนาโดยบริษัท Aldebaran Robotics เมื่อปี 2004 และในปี 2007 หุ่นยนต์อิวามานอยด์นาโอะได้นำไปแทนที่หุ่นยนต์สูนัขของ Sony ซึ่ง Aibo ขณะนั้นใช้ในการแข่งขัน 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 จะประกอบไปด้วยเซนเซอร์วัดมุมอุ่น 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 สามารถถือโน้ตเพลงและเล่นดนตรีโดยใช้ electronic organ แบบง่ายๆ ได้ และในปี 1985 บริษัท Hitachi ได้สร้างหุ่นยนต์ WHL-11 ที่มีสองขาเหมือนมนุษย์ ซึ่งสามารถเดินแบบสมดุลสถิต (Static Walking) บนพื้นราบได้ด้วยความเร็ว 13 วินาทีต่อหนึ่งก้าว และสามารถเลี้ยวได้ซ้ายและขวาได้

## 2.2 การออกแบบโครงสร้างของหุ่นยนต์

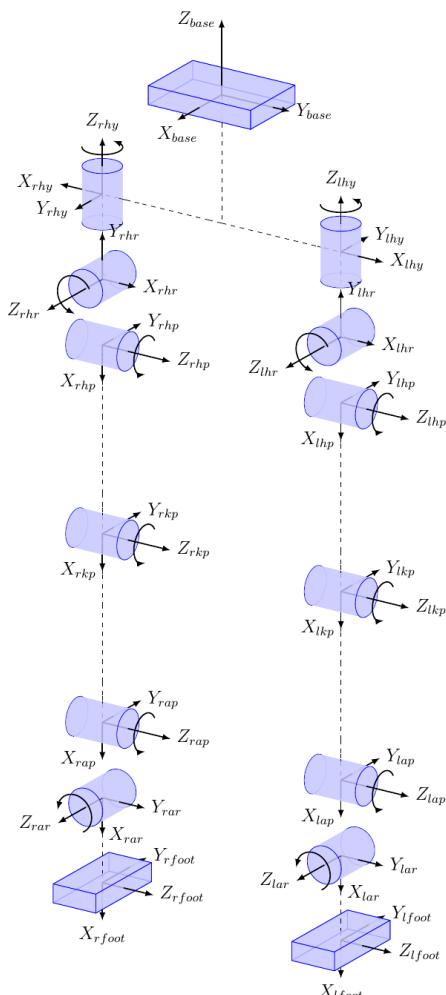
### 2.2.1 ความแตกต่างระหว่างโครงสร้างของมนุษย์กับโครงสร้างของหุ่นยนต์

#### 2.2.1.1 ความแตกต่างขององค์การเสรี

เนื่องจากลักษณะข้อต่อของมนุษย์มีความซับซ้อนมากกว่าโครงสร้างของหุ่นยนต์ ทำให้ข้อต่อแต่ละจุดของมนุษย์นั้นสามารถหมุนได้หลายทิศทาง รวมถึงขอบเขตของการหมุนของข้อต่อในแต่จุดก็มีความแตกต่างกัน ใน การนำรูปแบบการเดินของมนุษย์ไปใช้กับหุ่นยนต์จึงต้องปรับค่ามุมที่ข้อต่อให้มีความเหมาะสมกับโครงสร้าง และ ข้อจำกัดเกี่ยวกับการหมุนของข้อต่อจุดต่างๆของหุ่นยนต์ที่จะใช้ทดสอบด้วย

#### 2.2.1.2 ความแตกต่างของอัตราส่วน

นอกจากความแตกต่างขององค์การเสรี (DoF) ระหว่างมนุษย์กับหุ่นยนต์แล้ว ความแตกต่างของอัตราส่วนระหว่างโครงสร้างแต่ละส่วนของมนุษย์กับหุ่นยนต์เป็นอีกสาเหตุหนึ่ง ที่ต้องทำการปรับแต่งใหม่มีความเหมาะสม เนื่องจากความยาวของโครงสร้างแต่ละส่วน รวมทั้งระยะห่างระหว่างจุดหมุนแต่ละจุดของมนุษย์กับหุ่นยนต์ที่มีความแตกต่างกัน ดังนั้นจึงต้องกำหนดระบบพิกัดสำหรับหุ่นยนต์ที่มีความยาวต่างกัน เช่น ขาหน้าและขาหลัง ที่มีความยาวต่างกัน ทำให้สามารถอ้างอิงจุดหมุน และความยาวของโครงสร้างในส่วนต่างๆ



รูปที่ 2.13: ตัวอย่างตำแหน่งและการหมุนของข้อต่อของหุ่นยนต์เพื่อการอ้างอิง

### 2.2.1.3 กำลังและประสิทธิภาพของมอเตอร์

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

### 2.2.2 อุปกรณ์ที่ใช้ในหุ่นยนต์ชีวามโนยด์

#### ตัวขับเคลื่อน

ในการสร้างหุ่นยนต์ชีวามโนยด์นั้นระบบการขับเคลื่อนถือว่าเป็นเรื่องสำคัญ เนื่องจากว่าถ้าหากระบบขับเคลื่อนไม่สามารถทำงานได้อย่างเต็มประสิทธิภาพ หรือหากมีการอุบัติเหตุที่ผิดพลาด จะส่งผลทำให้หุ่นยนต์ชีวามโนยด์นั้นมีประสิทธิภาพในการทำงานลดลงตามไปด้วย ภายในงานวิจัยนี้ทางผู้จัดทำได้ใช้ตัวขับเคลื่อนเป็น Dynamixel digital servo EX-106 ซึ่งเป็นเซอร์โวมอเตอร์ที่เหมาะสมสำหรับทำหุ่นยนต์โดยเฉพาะ ภายในประกอบไปด้วย มอเตอร์กระแสตรง ชุดเฟืองมอเตอร์ ไดเรอර์คอนโทรลเลอร์ สามารถเชื่อมต่อกันผ่าน BUS RS-485 มีการควบคุมแบบ PID และแรงบิดที่สูง<sup>5</sup>



รูปที่ 2.14: ตัวขับเคลื่อนที่ใช้ในหุ่นยนต์ชีวามโนยด์

<sup>5</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)]

## หน่วยประมวลผลควบคุม

ในการควบคุมหุ่นยนต์ชีวามโนยด้วยความสามารถทำกิจกรรมต่างๆ คือ หน่วยประมวลผลระบบควบคุม ถ้าหากไม่มีระบบประมวลผลควบคุมแล้ว อุปกรณ์ต่างๆ ที่ติดตั้งอยู่ภายในตัวของหุ่นยนต์ชีวามโนยด้วยจะไม่สามารถติดต่อสื่อสารกันได้ ซอฟท์แวร์ของหุ่นยนต์ที่พัฒนามาทั้งหมดจะไม่สามารถใช้ได้ ทำให้หุ่นยนต์ชีวามโนยด้วยไม่สามารถทำงานในสิ่งที่ต้องการ การวางแผนระบบควบคุมที่นิยมใช้ในระบบหุ่นยนต์ชีวามโนยด้วยส่วนใหญ่ จะแบ่งออกเป็น 2 ส่วนด้วยกัน คือส่วนของหน่วยประมวลผลควบคุมระดับสูง และหน่วยประมวลผลควบคุมระดับต่ำ

### หน่วยประมวลผลควบคุมระดับสูง (High level controller)

หน่วยประมวลผลควบคุมระดับสูงเป็นส่วนที่ใช้ประมวลผลการทำงานที่มีความซับซ้อนของระบบ เช่น จลนศาสตร์ของหุ่นยนต์ การคำนวณหาเส้นทางการเดิน ในการคำนวณทางคณิตศาสตร์ของระบบเหล่านี้จำเป็นต้องมีการประมวลผลที่เร็ว และมีประสิทธิภาพ ในสมัยที่มีการพัฒนาหุ่นยนต์ชีวามโนยด้วยคุ้นเคยเริ่มนั้น หน่วยประมวลผลควบคุมระดับสูง จะใช้คอมพิวเตอร์เป็นตัวในการประมวลผลการคำนวณ ซึ่งคอมพิวเตอร์สมัยนั้นมีขนาดใหญ่ น้ำหนักมาก และต้องใช้พลังงานสูง ซึ่งต่างจากปัจจุบันนี้ที่มีการพัฒนาของเทคโนโลยีที่ก้าวหน้ามากขึ้น ทำให้คอมพิวเตอร์มีขนาดเล็กลงเทียบเท่ากับบอร์ดคอนโทรลเลอร์ทั่วไป<sup>6</sup>



รูปที่ 2.15: ตัวประมวลผลระดับสูงของ Thormang Humanoid

### หน่วยประมวลผลควบคุมระดับต่ำ (Low level controller)

หน่วยประมวลผลควบคุมระดับต่ำเป็นส่วนที่รับคำสั่งมาจากหน่วยประมวลผลควบคุมระดับสูง มีประสิทธิภาพในการประมวลผลการคำนวณที่น้อยกว่า เนื่องจากการออกแบบสถาปัตยกรรมภายในระบบไม่เอื้ออำนวยต่อการคำนวณที่มีความซับซ้อน แต่มีความสามารถในการประมวลผลระบบที่เป็นควบคู่ด้วยแม่นยำ ในด้านการทำหุ่นยนต์ชีวามโนยดันนั้นมากจะใช้หน่วยประมวลผลควบคุมระดับต่ำ ในการติดต่อกับอุปกรณ์ต่างๆ บนตัวของหุ่นยนต์ชีวามโนยด้วยตรง เช่น ตัวขับเคลื่อน เชนเชอร์รับค่า หรือไฟแสดงสถานะต่างๆ ของหุ่นยนต์



รูปที่ 2.16: ตัวประมวลผลระดับต่ำของ Robotis OP3 Humanoid

<sup>6</sup>Robot controller Robotis, Thormang, [http://jp.robotis.com/index/product.php?cate\\_code=111410](http://jp.robotis.com/index/product.php?cate_code=111410)

## เซนเซอร์ตรวจหน้าสัมผัสที่พื้น

เซนเซอร์ตรวจหน้าสัมผัสที่พื้นเป็นเซนเซอร์ที่ถูกติดตั้งบริเวณฝ่าเท้า เพื่อตรวจสอบการเดินของหุ่นยนต์ ขีวนามอยด์ว่าขณะนี้มีการสัมผัสดของฝ่าเท้าของหุ่นยนต์กับพื้นหรือไม่ ซึ่งในงานวิจัยนี้ได้ใช้หลักการตัวตรวจจับแรงกดแบบค่าความต้านทานหรือ Force Sensing Resistor (FSR)<sup>7</sup> ที่ใช้เทคโนโลยีฟิล์มโพลีเมอร์แบบหนาโดยที่เซนเซอร์สามารถเปลี่ยนแรงที่มากระทำให้อยู่ในรูปของการเปลี่ยนแปลงค่าความต้านทานไฟฟ้า ตัวเซนเซอร์มีลักษณะเป็นแผ่น มีโครงสร้าง 5 ชั้น โดยสองชั้นนอกสุดเป็นฟิล์มของโพลีเอสเตอร์ ส่วนชั้นถัดเข้ามาเป็นฟิล์มของโลหะที่เป็นตัวนำไฟฟ้า และชั้นในสุดเป็นหมึกที่มีความไวในการตอบสนองต่อแรงภายนอกที่มากระทำ (Pressure sensitive ink) และโครงสร้างทั้ง 5 ชั้น ถูกรวบเข้าด้วยกันด้วยวิธีลามิเนท จึงทำให้เซนเซอร์วัดแรงนี้มีลักษณะแบบมีความยืดหยุ่นสูง ด้วยเหตุนี้จึงทำให้เซนเซอร์สามารถโค้งงอได้ง่าย แรงดันไฟฟ้าที่ต่อกลับจะลดลง เมื่อมีแรงกดมากระทำบนแผ่นตรวจจับ มีโครงสร้างของตัวตรวจจับแสดงในรูปที่ 2.17



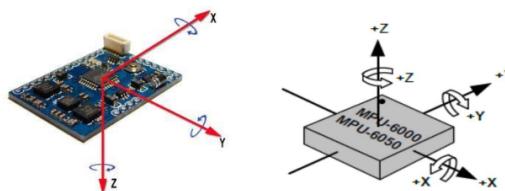
รูปที่ 2.17: ลักษณะโครงสร้างของตัวตรวจจับแรงกด FSR

## เซนเซอร์วัดความเร็ว

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

เซนเซอร์วัดความเร็ว (Gyroscope)<sup>8</sup> เป็นอุปกรณ์สำหรับการวัดความเร็ว หรือการรักษาการปรับทิศทาง ขึ้นอยู่กับหลักการของการอนุรักษ์โมเมนตัมเชิงมุม ถ้าไม่มีการเคลื่อนที่ อัตราการเปลี่ยนแปลงมุมจะมีค่าเท่ากับศูนย์

เซนเซอร์วัดความเร่ง (Accelerometer)<sup>9</sup> เป็นอุปกรณ์ที่ใช้วัดความเร่งเชิงเส้น โดยอาศัยการวัดแรงที่กระทำต่อน้ำหนัก อ้างอิงที่เกิดจากแรงโน้มถ่วงโลก ซึ่งแรงโน้มถ่วงของโลกจะเป็นเวกเตอร์ซึ่งไปที่แกนกลางโลกเสมอ ตามกฎของนิวตัน



รูปที่ 2.18: เซนเซอร์วัดความเร็ว

<sup>7</sup>[UNICON] Force sensor with UNICON [http://doc.inex.co.th/force-sensor-with-unicon/]

<sup>8</sup>Mechanic gyroscope two-degree of freedom [https://www.bosch-sensortec.com/bst/products/motion/gyro-scope/overview\_gyroscopesensors]

<sup>9</sup>Accelerometer and Gyroscopes Sensor [https://www.maximintegrated.com/en/app-notes/index.mvp/id/5830]

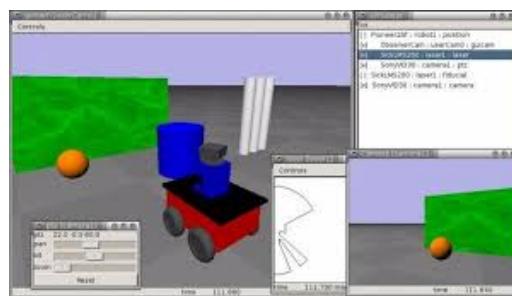
## 2.3 การออกแบบโปรแกรมด้วย ROS

### 2.3.1 ระบบที่ใช้ช่วยในการพัฒนาหุ่นยนต์

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

ปัจจุบันมี Robot Middleware ที่ถูกพัฒนาขึ้นมาให้ใช้อยู่หลายตัว เช่น

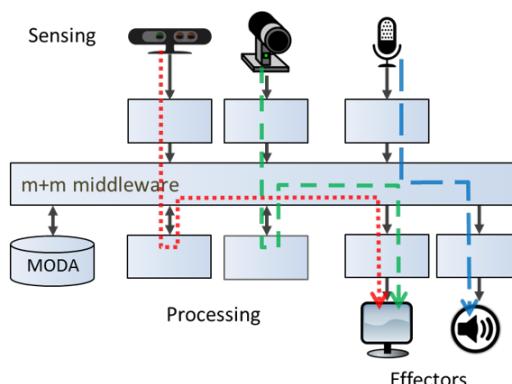
#### Player Project



รูปที่ 2.19: player project middleware

เป็นโปรเจกต์ที่ใช้ในการสร้างซอฟแวร์เพื่อการศึกษาวิจัยที่มีความเกี่ยวข้องกับหุ่นยนต์และระบบเชนเชอร์ ภายในประกอบไปด้วยระบบตัวกลาง และระบบจำลองการทำงานของหุ่นยนต์

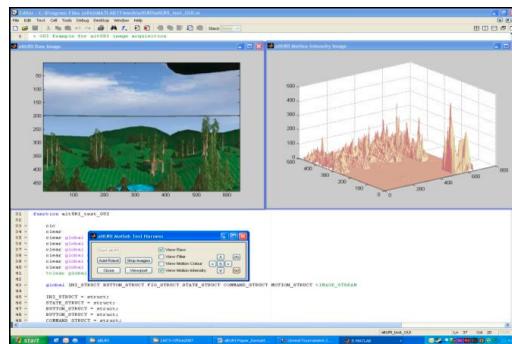
#### YARP



รูปที่ 2.20: yarp middleware

เป็น open source ที่เขียนด้วยภาษา C++ ในการเข้มต่อกับเชนเชอร์ หน่วยประมวลผล และตัวขับเคลื่อนของหุ่นยนต์

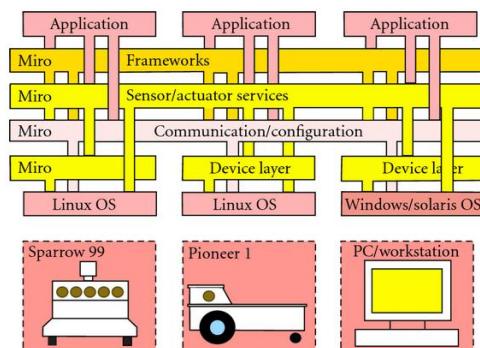
## URBI



รูปที่ 2.21: urbi middleware

เป็น open source สำหรับพัฒนาแอพพลิเคชันที่เกี่ยวข้องกับหุ่นยนต์หรือระบบที่มีความซับซ้อน ใช้ภาษาพื้นฐานเป็นภาษา C++ ติดต่อสื่อสารได้ภายในเครือข่ายเดียวกันเท่านั้น (Local Network)

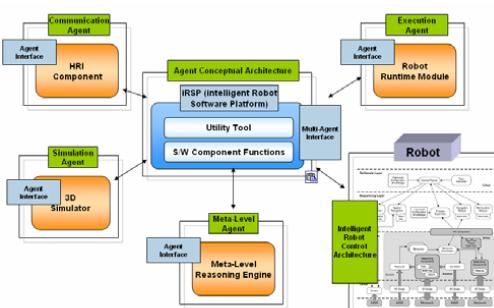
## MIRO



รูปที่ 2.22: miro middleware

เป็นกรอบการทำงานของหุ่นยนต์ที่เคลื่อนที่โดยใช้ในลักษณะเป็น OOP

## OpenRDK

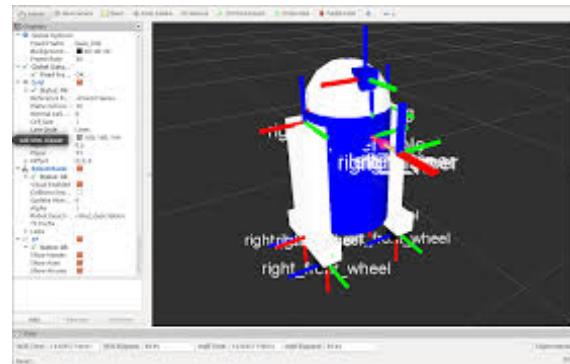


รูปที่ 2.23: openrdk middleware

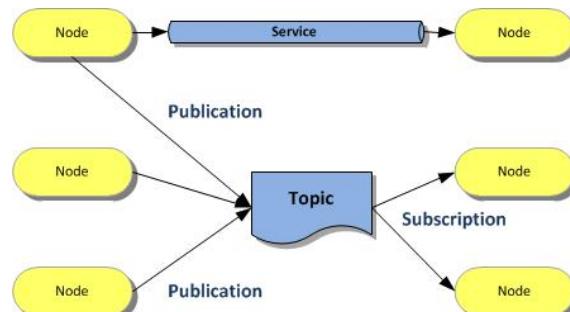
เป็น open source สำหรับพัฒนาระบบที่มีความเป็นอิสระต่อกัน (Modules) สามารถใช้ช่องทางการติดต่อสื่อสารและหน่วยความจำร่วมกันได้

## Robot Operating System

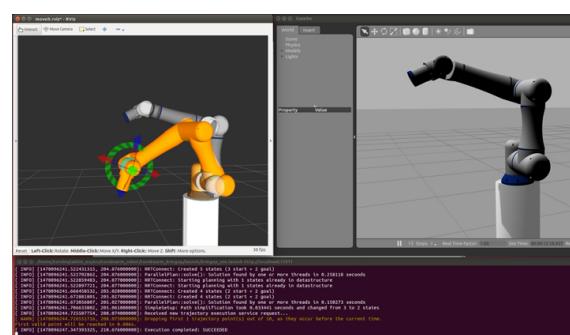
Robot Operating System หรือ ROS ถูกพัฒนาโดยบริษัท Willow Garage, แต่เดิมแล้ว ROS ถูกพัฒนาเพื่อใช้งานกับหุ่นยนต์ PR2 ในปี 2007 ซึ่งพัฒนาเป็น Open Source framework สำหรับนักพัฒนาซอฟแวร์ที่เกี่ยวข้อง กับหุ่นยนต์ มีความสามารถในการทำงานแบบ parallel บนคอมพิวเตอร์หลายเครื่องได้ สามารถทำงานได้ทั้งหลาย OS นอกจากนี้ยังมีคลังที่ค่อยเก็บซอฟแวร์ต่างๆไว้เป็น libraries อีกด้วย การใช้ ROS จะช่วยทำให้เราสามารถพัฒนาหุ่นยนต์ได้อย่างรวดเร็วมากขึ้น ประหยัดเวลา ประหยัดทรัพยากร



รูปที่ 2.24: ROS middleware Rviz



รูปที่ 2.25: ROS algitecture



รูปที่ 2.26: ROS Moveit

### 2.3.2 ระบบที่ใช้ในการจำลองการทำงานของหุ่นยนต์

โปรแกรมจำลองการทำงานของหุ่นยนต์นั้นเป็นเครื่องมือที่สำคัญสำหรับนักหุ่นยนต์ การใช้โปรแกรมจำลองนั้นจะช่วยเพิ่มประสิทธิภาพในการทำงานหลายอย่าง เช่น ให้รู้ว่าหุ่นยนต์ที่ออกแบบนั้นสามารถทำงานได้อย่างที่ต้องการหรือไม่ กระบวนการคิดถูกต้องหรือไม่ โปรแกรมจำลองระบบส่วนใหญ่จะคำนวณพลวัตของหุ่นยนต์โดยใช้เครื่องมือคำนวณ Open Dynamics Engine (ODE)

#### USARSim



รูปที่ 2.27: ผลลัพธ์จากการใช้โปรแกรม USARSim

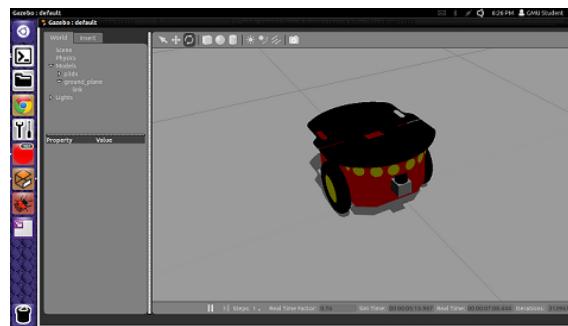
USARSim เป็นโอเพนซอร์ซและเหมาะสมสำหรับทำหุ่นยนต์ประเภทกู้ภัยในชา衾เมือง โดยมีฐานการพัฒนามาจาก Unreal Tournament game engine ภายใต้โปรแกรมมีเครื่องมือสำหรับการทำงานวิจัย มีเซนเซอร์ของหุ่นยนต์ที่หลากหลาย เช่น เซนเซอร์รับภาพ หรือเซนเซอร์ตรวจความเคลื่อนไหว

#### MuRoSimF

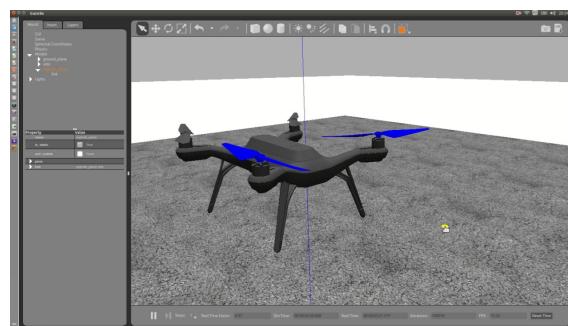


รูปที่ 2.28: ผลลัพธ์จากการใช้โปรแกรม MuRoSimF

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



รูปที่ 2.29: Mobile robot with gazebo



รูปที่ 2.30: Quadrotor with gazebo

### Gazebo

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

**Link :** คือก้านต่อของหุ่นยนต์ซึ่งภายในจะสามารถออกแบบขนาด รูปร่าง สี และสามารถ import 3d mesh เข้ามาได้ด้วย อีกทั้งยังสามารถใส่รายละเอียดของการเคลื่อนที่ของก้านต่อได้ เช่น inertial matrix และ collision properties

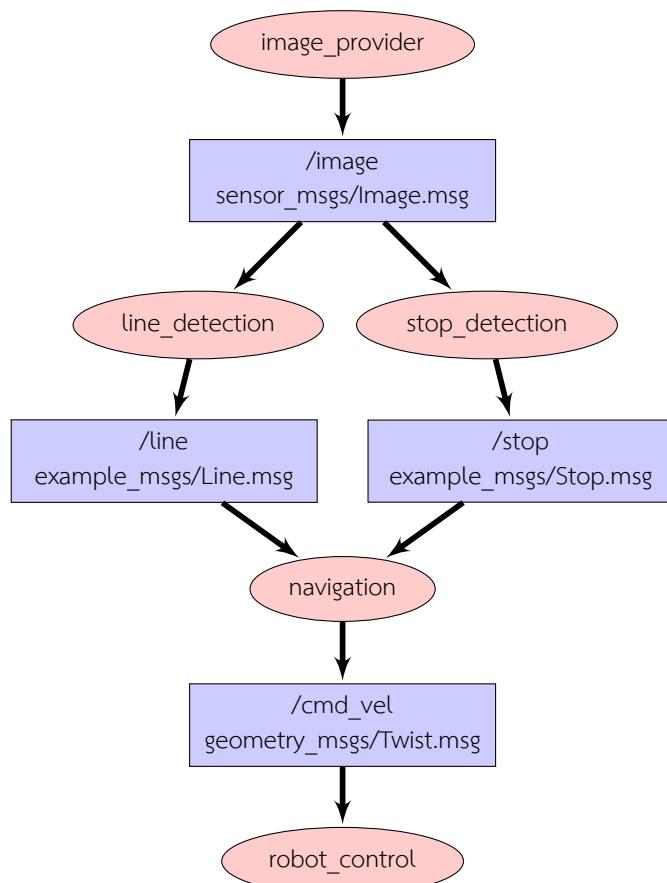
**Joint :** คือข้อต่อของหุ่นยนต์สามารถกำหนดกลศาสตร์และการเคลื่อนที่ได้ เช่น Joint limits ของข้อต่อที่กำลังหมุนและความเร็วการหมุน ซึ่งข้อต่อมีหลายแบบที่สามารถกำหนดได้ เช่น ข้อต่อแบบหมุน, ข้อต่อแบบเลื่อน, ข้อต่อต่อแบบบิดติด, ข้อต่อแบบต่อเนื่อง

### 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

#### Node

Node เป็นหนึ่งในหน่วยประมวลผลในระบบ ROS, Node สามารถที่จะส่งข้อมูลหา Node อื่นๆได้ ผ่าน Topics หรือ Services ในทางปฏิบัติแล้วโหนดเป็นตัวประมวลผลอยู่ๆ ที่ค่อยทำงานน้ำที่เฉพาะ ยกตัวอย่าง เช่น โหนดตัวแรกเชื่อมต่อกับกล้อง เพื่อที่จะนำภาพจากกล้องออกมานะ โหนดตัวที่สองใช้ในการหาลูกบล็อกที่อยู่ในภาพที่ได้มาจากโหนดตัวแรก และโหนดตัวที่สามใช้ในการคำนวนหาตำแหน่งของลูกบล็อกที่อยู่บนโลกจริงๆ จากตำแหน่งของลูกบล็อกที่ได้มาจากโหนดที่สอง ดังนั้นจะเห็นว่าแต่ละโหนดจะทำงานเฉพาะของตัวเอง ซึ่งสามารถนำมาร่วมกันได้ การเขียนเป็นแบบโหนดจะช่วยทำให้เราสามารถที่จะนำโปรแกรมกลับมา แก้ไขปรับปรุงให้ใช้ใหม่ได้ง่าย ในกรณีที่จะนำไปทำงานอย่างอื่น ยกตัวอย่าง เช่น โหนดที่เอาภาพจากกล้องออกมานะ อาจจะมีโหนดอีกตัว ทำงานน้ำที่ในการหาโกล์ดเป้าหมาย และหาทิศทางการเคลื่อนที่ของหุ่นยนต์ได้ ดังนั้นการพัฒนาโหนดเป็นส่วนย่อยๆเล็กๆ ก็เพื่อที่จะทำให้การแก้ไขหรือปรับปรุงได้ง่าย



รูปที่ 2.31: ตัวอย่างสถาปัตยกรรมของ ROS

จากตัวอย่างสถาปัตยกรรมของ ROS ดังรูปที่ 2.31 นั้นสามารถอธิบายได้ว่า หุ่นยนต์เคลื่อนที่ด้วยล้อมีภารกิจคือ เคลื่อนที่ตามเส้นไปเรื่อยๆจนกว่าจะเจอเครื่องหมายหยุด Node คือตัวที่แสดงด้วยรูปวงรี ข้างในเป็นชื่อ Hind ส่วน Topic จะแสดงด้วยรูปสี่เหลี่ยม ซึ่งข้างในเป็นชื่อของ Topic และชนิดของ Message ที่ใช้ในการส่งข้อมูล มาดูกันก่อนอื่น ภาพถูกส่งมาจากการล้อ แลก็มี Hind สองตัวในการดูเส้น 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 สອอันนี้ 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 จะไม่ทำให้เกิดคอกขัด เวลา rans ระบบ ในกระบวนการที่คือ subscribe node จะถาม rosmaster ว่ามี topic ที่ต้องการรับข้อมูลใหม่ ส่วนตัว master ที่เก็บค่า topic message เอาไว้ ก็จะส่งไปยัง subscribe node ถ้าหากมีข้อความที่ร้องขอมา 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 หรืออีกชื่อหนึ่งคือ remote procedure calls (RPC) เป็นการส่ง messages แบบที่ไม่ได้เจาะจงว่าจะส่งไปที่ไหน เมื่อ service ถูกเรียกแล้วระบบจะรอนกว่าจะมีการตอบกลับ เราจะเรียกกระบวนการนี้ว่า request และ response message Node ที่อยู่ทำงานเมื่อมีการเรียกใช้ service จะเรียกว่า service server และ node ที่เรียก service จะเรียกว่า service client การใช้งาน service เหมาะสำหรับงานที่ต้องการความรวดเร็ว (fast task) แต่ไม่ควรใช้กับระบบที่ต้องใช้เวลานาน เพราะระบบจะหยุดไม่ยอมทำต่อ ต้องรอให้ service ทำงานเสร็จก่อน สำหรับงานที่ต้องใช้เวลาในการคำนวณนานจะไปใช้ 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 แล้ว 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

```
<package>
    <name>example_package</name>
    <version>1.0.0</version>
    <description>Short example for a package.xml.</description>
    <maintainer emanil="ex@example.org">Jane Doe</maintainer>
    <license>BSD</license>
    <buildtool_depend>catkin</buildtool_depend>
    <build_depend>example_2</build_depend>
    <run_depend>std_msgs</run_depend>
</package>
```

รูปที่ 2.32: ตัวอย่างไฟล์ package.xml

แต่ละ tags ใช้ในการบอกข้อมูลของ package นี้ ใครเป็นเจ้าของ ใครเป็นคนเขียน รวมไปถึง dependencies ที่จำเป็นต้องใช้ของ package นี้ด้วย ดังรูปที่ 2.32

## Code Distribution

การที่จะนำ Nodes กลับมาใช้ใหม่หรือเอาอกมาแบ่งปันให้ผู้อื่นได้นั้น จะต้องมีการทำเอกสารของ Packages นั้นๆด้วย 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 ที่ทำหน้าที่คล้ายๆกันหลายๆตัวมารวมไว้ที่เดียวเพื่อจะได้ใช้งานง่าย ตัวอย่าง Navigation metapackage ประกอบไปด้วย 10 packages เช่น [AMCL(partical filter), DWA, EKF(extended kalman filter) และ map\_server] ซึ่งหากติดตั้ง metapackage ตัวนี้ก็จะได้มาหมดเลย

ในส่วนนี้จะอธิบายคร่าวๆถึง ROS standard packages ที่จะนำมาใช้ในงานวิจัยครั้งนี้

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

**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

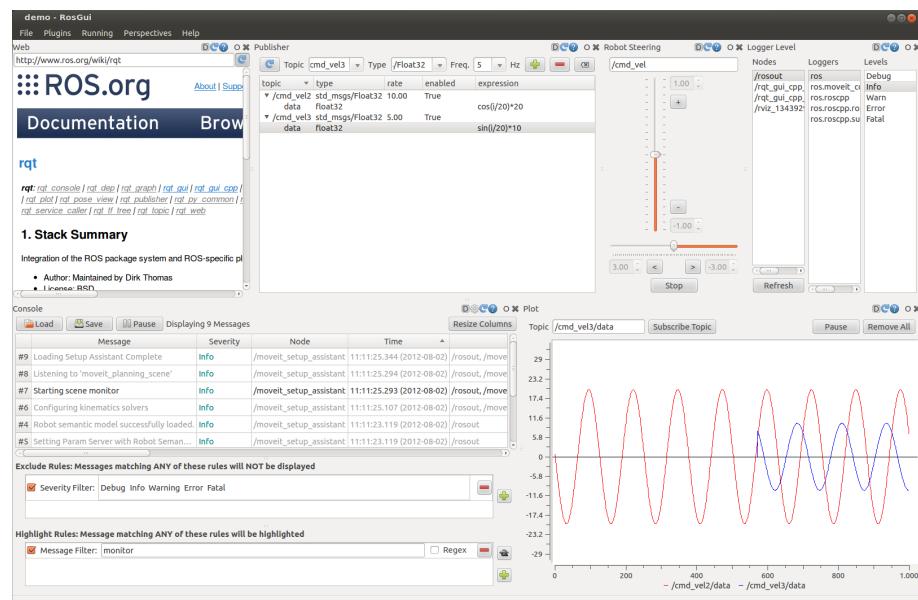
**URDF** Unified Robot Description Format (URDF) เป็นไฟล์ XML ที่เอาไว้อธิบายลักษณะของหุ่นยนต์ ใน ROS มีแพกเกจที่ใช้สำหรับการอ่านไฟล์ คือ urdf\_parser แต่ไฟล์นี้มีการใช้งานโดย tf2 เช่นกัน

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

## Visualization

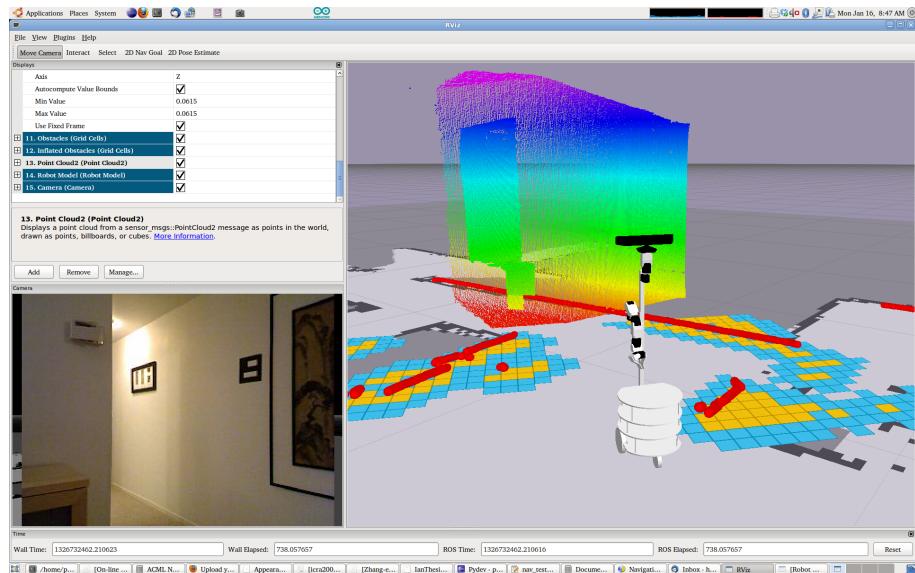
จุดเด่นสำคัญของ ROS อีกอย่างคือ มีเครื่องมือที่ช่วยในการแสดงผล Visualization ได้ นอกจากนี้จากระบบ publisher-subscriber การใช้ Visualization tools นี้จะช่วยให้การทำงานง่ายขึ้นและประหยัดเวลามากขึ้น ในการนำข้อมูลต่างๆจากหุ่นยนต์ออกมาระบบการแสดงผล เพราะว่า Visualization tools นี้สามารถที่จะ subscribe จาก topic ที่มีการใช้งานอยู่แล้วมาแสดงผลได้ทันที ใน ROS มีเครื่องมือสำคัญ 2 ตัวที่ใช้สำหรับการ Visualization ซึ่งสามารถที่จะปรับแต่งให้กลายเป็นเวอร์ชั่นของเราเองได้

**rqt** rqt เป็น UI ที่มีฐานมาจาก QT ซึ่งมาพร้อมกับการเชื่อมต่อ ROS เป็นส่วนเสริมในรูปแบบของ QWidget เราสามารถที่จะแสดงผลหลายๆ widgets ได้ภายในเวลาเดียวกัน สามารถที่จะย่อขยาย เปลี่ยนตำแหน่ง ลากวางได้ การเชื่อมต่อกับ ROS นั้นสามารถนำการแสดงผลแบบ 2D ไปแสดงได้ดังรูปที่ 2.33 เป็นการแสดงภาพของกราฟที่ได้รับข้อมูลมาจากการ publish หลากหลายตัว และสามารถที่จะปรับแต่งค่าและ publish ออกไปได้ด้วยการเขียนโปรแกรมเข้าไป ซึ่งจะเป็นประโยชน์อย่างมากเวลาที่ใช้ในการปรับจูนพารามิเตอร์ต่างๆ เพราะว่าเราสามารถที่จะเปลี่ยนค่าได้ทันที ไม่ต้องรันโปรแกรมใหม่ ในรูปที่ 2.33 เป็นการนำ rqt มาใช้เป็น GUI ให้ผู้ใช้สามารถใช้งานได้ง่ายและสามารถที่จะปรับแต่งพารามิเตอร์ต่างๆได้เรียลไทม์



รูปที่ 2.33: ตัวอย่างการแสดงผลใน rqt

**RViz** RViz เป็น 3D visualization ของสถานะต่างๆของหุ่นยนต์และสภาพแวดล้อม โดยใช้ไฟล์ URDF เป็นมาตรฐานการแสดงถึงหุ่นยนต์ ซึ่งสามารถที่จะแสดงตำแหน่งปัจจุบันของข้อต่อต่างๆในหุ่นยนต์ได้ สามารถที่จะแสดงค่าเซนเซอร์เป็น marker ได้ การใช้งานจะเป็นเหมือนการบอกพิกัดเฟรม ลักษณะการแสดงผลใน RViz มีหลากหลายรูปแบบไม่ว่าจะเป็น camera images, depth clouds, laser scans หรือ point clouds อย่างไร ก็ตามการแสดงผลใน Rviz นั้นจะไม่ได้คำนึงถึงแรงที่เข้ามากระทำกับตัวของหุ่นยนต์ แต่ถ้าเป็นการเคลื่อนที่ที่มีพิกัดเฟรมแล้วสามารถนำมาแสดงได้ ดังรูปที่ 2.34 เป็นตัวอย่างของหุ่นยนต์เคลื่อนที่ด้วยล้อ และทำแผนที่ด้วยข้อมูลความลึกที่ได้มาจากการ Kinect

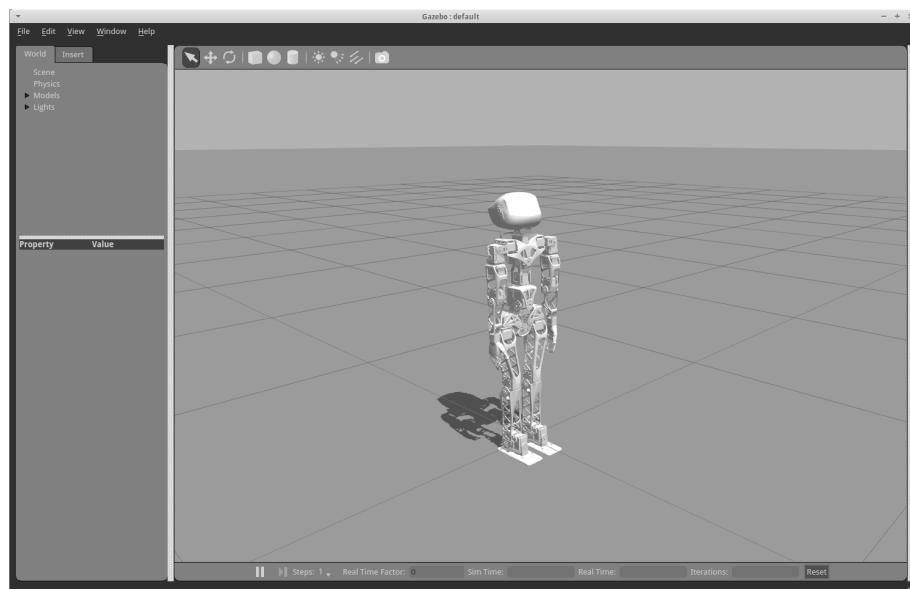


รูปที่ 2.34: ตัวอย่างการแสดงผลใน RViz

### Simulation

Simulation เป็นส่วนที่สำคัญมากสำหรับการพัฒนาโปรแกรมของหุ่นยนต์ เพราะว่าเราสามารถที่จะสร้างโปรแกรมและทดสอบได้โดยไม่จำเป็นต้องมี hardware ซึ่งในส่วนนี้จะช่วยลดความเสียหายจากบักหรือโปรแกรมผิดพลาด ที่อาจจะเกิดขึ้นกับหุ่นยนต์ของเรา การจำลองจะช่วยลดเวลาในการพัฒนาลงได้ ระบบจำลองปัจจุบันมีมากมายหลายตัวแต่ ตัวที่ได้รับคำแนะนำมากที่สุดคือ Gazebo เพราะว่า gazebo สามารถที่จะเชื่อมต่อกับ ROS ได้โดยตรง และนักพัฒนาส่วนใหญ่ใช้ gazebo

การจะใช้ gazebo ได้นั้นเราจะต้องใช้ไฟล์ URDF ซึ่งเป็นไฟล์ที่เอาไว้แสดงหุ่นยนต์ในระบบจำลอง และสามารถที่จะคำนวณหา collision ให้เราได้อีกด้วย



รูปที่ 2.35: ตัวอย่างหุ่นยนต์ชีวนิรภัย 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 นั้นจะสามารถทำงานบนฮาร์ดแวร์อุปกรณ์ใดๆ ก็ได้

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

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

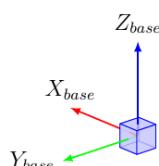
ส่วนที่นั่นยนต์ Non-open source platform คือที่นั่นยนต์ที่สร้างมาเฉพาะเจาะจงสำหรับการวิจัย การสำรวจ หรือการแข่งขันโดยเฉพาะ ไม่เปิดให้บุคคลภายนอกเข้าศึกษาหรือแก้ไขปรับปรุง ซึ่งทำให้ที่นั่นยนต์ประเภทนี้ไม่เหมาะสมสำหรับผู้ใช้ที่จะเรียนรู้และศึกษาด้วยตนเอง เพราะมีขั้นตอนใหญ่ ใช้ทรัพยากร่มาก และการออกแบบมีความซับซ้อน 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 2.36



รูปที่ 2.36: 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)	แอม培ร์(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>W</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 ได้รวมซอฟต์แวร์เครื่องมือที่หลากหลายเอาไว้เป็นหมวดหมู่ เช่น การเชื่อมต่อกับฮาร์ดแวร์ การสร้างระบบควบคุมให้กับอุปกรณ์ต่างๆ อีกทั้งสามารถที่จะเขียนโปรแกรมแล้วนำกลับมาใช้ใหม่ได้ ภายในระบบมีกระบวนการรับส่งข้อมูลต่างๆ เป็นของตัวเอง ทำให้ช่วยลดความซับซ้อนและเพิ่มประสิทธิภาพในการทำงานกับแพลตฟอร์มของหุ่นยนต์ กระบวนการเขียนโปรแกรมของ ROS นั้นจะใช้รูปแบบ Graph architecture ซึ่งจะทำให้สามารถแบ่งโปรแกรมต่างๆออกเป็นส่วนๆ เช่น เชนเซอร์หลายๆตัว ระบบควบคุม ระบบวางแผน ระบบขับเคลื่อน ระบบสื่อสารภายนอก ด้วยตัวระบบของ ROS นั้น ไม่ใช่ Real Time OS แต่อย่างไรก็ตามเราสามารถใช้งานผสมกับ Real Time ได้

ROS ประกอบไปด้วยแพ็กเกจเจตต่างๆ มาประกอบกันเป็น Node โดยมีตัวกลางทำหน้าที่ในการติดต่อสื่อสารระหว่างอุปกรณ์ที่เป็นโนندต่างๆ 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

วิทยานิพนธ์เรื่องการออกแบบโครงสร้างและพัฒนาระบบพื้นฐานของหุ่นยนต์ชีวภาพอยู่ด้านนี้ ผู้วิจัยได้มีการตั้งชื่อให้หุ่นยนต์โดยใช้ชื่อว่า อุทัย (UTHAI) มาจากภาษาอังกฤษคำว่า Universal Template for Humanoid Algorithm Interface เพื่อให้สมกับเป็นหุ่นยนต์ชีวภาพอยู่ด้วยที่ใช้สำหรับงานวิจัยและพัฒนาต่ออยอด

#### 3.1 แผนการดำเนินงาน

โดยจากที่กล่าวไปตอนต้นในบทนำ การดำเนินงานและการออกแบบการสร้างหุ่นยนต์ชีวภาพอยู่ด้วย มีแผนการทำงานซึ่งแบ่งออกเป็นสามส่วนดังนี้ ส่วนแรกคือ ส่วนของชาร์ดแวร์ที่เกี่ยวกับโครงสร้างทางกลของหุ่นยนต์ชีวภาพอยู่ด้วย เช่น ข้อต่อ ก้านต่อ ฝ่าเท้า รวมไปถึงระบบอิเล็กทรอนิกส์ ตัวประมวลผลการควบคุม เช่นเซอร์ตัวขับเคลื่อนต่าง ๆ และส่วนที่สองคือ ส่วนของซอฟต์แวร์ที่เกี่ยวข้องกับการติดต่อสื่อสารกันเบื้องต้น การควบคุมตัวขับเคลื่อนที่ข้อต่อ การอ่านค่าเซนเซอร์ และส่วนที่สาม คือระบบพื้นฐานสำหรับการนำไปศึกษาและพัฒนา โดยจะครอบคลุมไปถึงเอกสารวิธีการใช้งานทั้งในรูปแบบอฟฟิลайнและออนไลน์

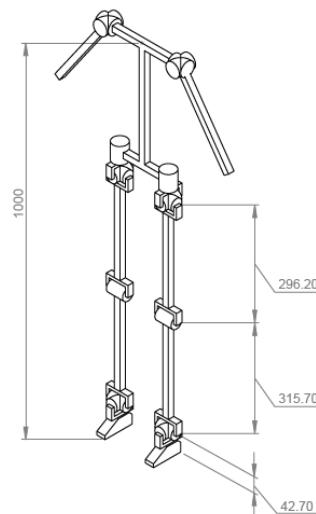
ในการเริ่มต้นทำงานวิจัยเกี่ยวกับชีวภาพอยู่ด้านนี้สิ่งจำเป็นที่ต้องทำในอันดับแรกคือการศึกษาสิ่งที่เคยมีอยู่ หรืองานวิจัยได้ทำเอาไว้แล้ว ศึกษาทำความเข้าใจใน ข้อดี-ข้อเสีย ของวิธีหรือกระบวนการต่างๆ เพื่อนำมาประยุกต์ใช้กับหุ่นยนต์ชีวภาพอยู่ด้วยของเรา โดยการศึกษานั้นจะเริ่มต้นจากการศึกษาโครงสร้างทางกลของหุ่นยนต์ชีวภาพอยู่ด้วยมีอยู่แล้วและมีสิ่งที่ต้องดูเป็นพิเศษคือ วิธีการเชื่อมต่องานระหว่างก้านต่อและข้อต่อ, จุดที่ใช้ติดตั้งเซนเซอร์ต่างๆ และการเลือกใช้วัสดุให้เหมาะสม รวมถึงการทำงานของเซนเซอร์และตัวขับเคลื่อนที่จำเป็นต้องใช้ในการควบคุมการทำงานของหุ่นยนต์ ในบทนี้ก็จะกล่าวถึงกระบวนการออกแบบและการดำเนินการตามแผนที่วางแผนไว้

#### 3.2 การออกแบบโครงสร้างของหุ่นยนต์

การออกแบบทางโครงสร้างทางกลนั้น ผู้วิจัยได้เลือกใช้โปรแกรม Solidworks เป็นเครื่องมือที่ช่วยในการพัฒนาโมเดลของหุ่นยนต์ชีวภาพอยู่ด้วย เนื่องจากโปรแกรม Solidworks เป็นโปรแกรมที่มีความสามารถในการขึ้นรูปและวาดแบบทางวิศวกรรม สามารถวิเคราะห์โครงสร้างทางกลของแบบจำลองได้ และมีการใช้งานอย่างแพร่หลาย อีกทั้งยังสามารถดาวน์โหลดโมเดลต่างๆ ที่มีคนพัฒนาเข้ามาใช้ร่วมกับการออกแบบของเราได้ และด้วยทางผู้วิจัยมีความชำนาญถึงความสามารถในการพัฒนาต่อยอดเป็นหลัก ดังนั้นการออกแบบโครงสร้างทางกลของหุ่นยนต์ชีวภาพอยู่ด้วย UTHAI จึงถูกออกแบบมาเพื่อให้สามารถรองรับการเปลี่ยนแปลง แก้ไขส่วนต่างๆ ของตัวหุ่นยนต์เองได้ในอนาคตอีกด้วย

### 3.2.1 โครงสร้างหุ่นยนต์

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



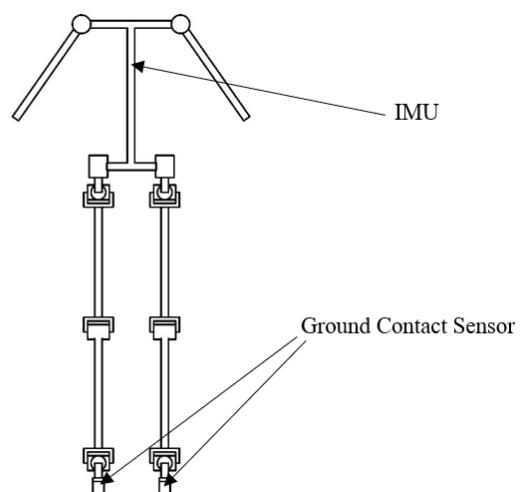
รูปที่ 3.1: ภาพแสดงแสดงโครงสร้างของหุ่นยนต์ UTHAI

เมื่อเราได้แบบจำลองของหุ่นยนต์ขึ้นมาอยู่แล้ว ลำดับต่อไปคือการกำหนดตำแหน่งการติดตั้งเซนเซอร์และตัวขับเคลื่อนต่างๆเข้าไป โดยมี Ground contact sensor ติดตั้งที่ใต้ฝ่าเท้าของหุ่นยนต์, IMU sensor ติดตั้งไว้ให้ใกล้กับจุด COM ของหุ่นยนต์ และ Dynamixel servo ติดตั้งไว้ที่ข้อต่อในแต่ละข้อต่อ

ส่วนโครงสร้างหุ่นยนต์ขึ้นมาอยู่ UTHAI ทางผู้วิจัยเลือกใช้วัสดุหลักเป็น PLA ที่ขึ้นรูปด้วยวิธีการขึ้นรูปสามมิติ และมีวัสดุเสริมบางชิ้นส่วนจากอลูминียม 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



รูปที่ 3.2: ภาพแสดงการติดตั้งเซนเซอร์ในจุดต่างๆ

### 3.2.2 จัดทำชิ้นส่วนโครงสร้างและประกอบ

ในการจัดทำชิ้นส่วนโครงสร้างนั้นทางผู้จัดทำได้คำนึงถึงความแข็งแรงเป็นหลักซึ่งมีความสำคัญมาก ในการเคลื่อนที่ของหุ่นยนต์ และยังคงต้องมีน้ำหนักที่เบาอีกด้วย<sup>1</sup> ดังนั้นจึงได้ใช้การเขียนรูปชิ้นงานด้วยเทคนิค การพิมพ์ 3 มิติ โดยจะใช้วัสดุหลักเป็นพลาสติก PLA ซึ่งมีความแข็งมากกว่าและเขียนรูปง่ายกว่าพลาสติกชนิด ABS เพื่อให้ตอบโจทย์กับหุ่นยนต์แพลตฟอร์มเพื่อพัฒนาต่ออยู่ในอนาคต ซึ่งผู้ใช้ทุกคนสามารถพิมพ์ชิ้นมาได้ด้วยตนเอง<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
การย่อสลายทางธรรมชาติ	ไม่ได้	ได้(ภายใต้เงื่อนไขที่ถูกต้อง)

ตารางที่ 3.2: ตารางแสดงสมบัติทางกลของวัสดุพลาสติก

แต่เนื่องจากว่าในปัจุบันนี้เครื่องพิมพ์ส่วนมากจะไม่รองรับการพิมพ์ชิ้นงานที่มีขนาดใหญ่ ซึ่งมีขนาดมากกว่า 30x30x30 ซม.(กว้างxยาวxสูง) ดังนั้นชิ้นงานที่เขียนรูปที่มีขนาดใหญ่เกินกว่านี้อาจจะต้องทำการตัดชิ้นงานออกก่อน และวิธีคือย่นมาประมาณกันทีหลังอีกรอบหนึ่ง โดยพื้นที่ทำการพิมพ์ของเครื่องพิมพ์ 3 มิติที่มีวางจำหน่าย และใช้งานแพร่หลายในท้องตลาดแสดงดังตาราง 3.3<sup>3</sup>

Printer	Actual Width	Actual Depth	Actual Height
MakerBot Replicator+	292	192	165
Ultimaker 3	188	185	200
LulzBot Mini	152	152	158
Dreammaker Overlord Pro Plus	79	79	255
New Matter MOD-t	145	95	125

ตารางที่ 3.3: ตารางแสดงขนาดของชิ้นงานที่สามารถพิมพ์ได้ในเครื่องพิมพ์ชนิดต่างๆ

<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>]

<sup>3</sup>The truth about 3D printer maximum print areas [<https://www.zdnet.com/article/what-manufacturers-don-t-want-you-to-know-the-truth-about-3d-printer-maximum-print-areas/>]

### 3.2.3 อุปกรณ์ที่ใช้ในหุ่นยนต์อิริวานอยด์อุทัย

#### Dynamixel servo EX-106+

Dynamixel EX-106+ เป็นตัวขับเคลื่อนที่นิยมใช้ในปัจจุบัน ซึ่งเป็นเซอร์โวโมเตอร์ที่เหมาะสมสำหรับทำหุ่นยนต์โดยเฉพาะ ภายในประกอบไปด้วย มอเตอร์กระแสตรง ชุดเพื่องมอเตอร์ ไดเรเวอร์คอนโทรลเลอร์ สามารถเชื่อมต่อกันผ่าน BUS RS-485<sup>4</sup> มีการควบคุมแบบ PID สามารถที่จะอ่านค่าความเร็ว แรงดันไฟฟ้า กระแสไฟฟ้า อุณหภูมิ ตำแหน่ง และแรงบิดจากมอเตอร์ทุกตัวได้ แต่ละมอเตอร์แต่ละตัวจะมีบอร์ดควบคุมของตัวเอง เราสามารถที่จะจ่ายไฟให้มอเตอร์และควบคุมผ่าน Serial ได้เลย

การทำงานของตัวขับเคลื่อนนี้สามารถทำได้ 2 รูปแบบคือ<sup>5</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>	



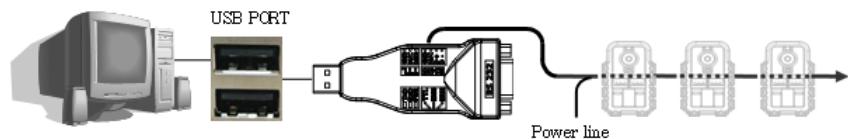
รูปที่ 3.3: แสดงประสิทธิภาพของมอเตอร์ EX-106+

<sup>4</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>5</sup>EX-106+ Mode [<http://www.trossenrobotics.com/dynamixel-ex-106-robot-actuator.aspx>]

### USB2Dynamixel connector

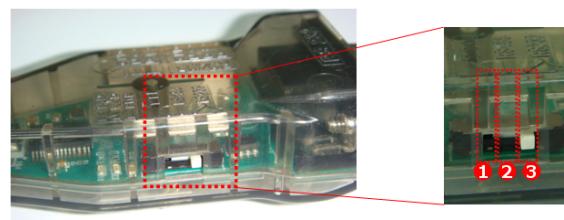
USB2Dynamixel เป็นอุปกรณ์สำหรับเชื่อมต่อ Odroid กับ Dynamixel Motor โดยจะเข้ามายังผ่านพอร์ต USB ของ Odroid ไปยัง Dynamixel motor ผ่านสายทั้งหมด 2 เส้น คือ D+ และ D- เป็นการซื้อต่อแบบ RS-485<sup>6</sup> ทำให้สามารถส่งข้อมูลระยะทางไกลได้ และสามารถที่จะมีหลายอุปกรณ์บนสายเส้นเดียวกันได้



รูปที่ 3.4: ภาพแสดงการติดต่อสื่อสารระหว่าง PC กับมอเตอร์ Dynamixel

ในการต่อใช้งานนั้นผู้ใช้งานจำเป็นต้องเลือกการติดต่อสื่อสารระหว่าง คอมพิวเตอร์กับมอเตอร์ ซึ่งการติดต่อสื่อสารนั้นหากใช้ USB2Dynamixel ตัวอุปกรณ์นี้ได้แบ่งการติดต่อสื่อสารออกเป็น 3 รูปแบบคือ

- 1 TTL Communication : สำหรับมอเตอร์ Dynamixels ที่ใช้พอร์ตชนิด 3-pin เช่นในตระกูล AX Series เช่น AX-S1 AX-12+ ฯลฯ
- 2 RS485 Communication : สำหรับมอเตอร์ Dynamixels ที่ใช้พอร์ตชนิด 4-pin เช่นในตระกูล DX Series เช่น RX Series, EX Series ฯลฯ
- 3 RS232 Communication : ใช้สำหรับติดต่อสื่อสารกับคอนโทรลเลอร์ผ่านสายเคเบิล



รูปที่ 3.5: ภาพแสดงการเลือกโหมดใช้งานของ USB2Dynamixel

แต่จากการทดลองนำมาใช้ผู้วิจัยพบว่า Dynamixels ที่ใช้ในการทำงานวิจัยครั้งนี้เป็นชนิด 4 pin ซึ่งใช้ RS485 ในการติดต่อสื่อสาร และด้วยขนาดของตัว USB2Dynamixel มีขนาดที่ใหญ่ทำให้การทำงานมีความลำบากในการติดตั้งลงบนตัวของหุ่นยนต์ จึงได้ทำการเปลี่ยนเป็น USB to RS485 แทน

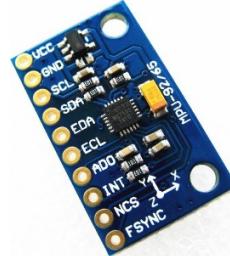


รูปที่ 3.6: USB2RS485 Module

<sup>6</sup>USB2Dynamixel [[http://support.robotis.com/en/product/auxdevice/interface/usb2dxl\\_manual.html](http://support.robotis.com/en/product/auxdevice/interface/usb2dxl_manual.html)]

### Inertial Measurement Unit (IMU)

ในการทำวิจัยครั้งนี้ผู้จัดทำได้เลือกนำเซนเซอร์ MPU-9250 มาใช้ในการอ่านค่ามุ่งเอียงของหุ่นยนต์เพื่อใช้ในการคุณลักษณะของหุ่นยนต์ โดยเซนเซอร์ตัวนี้สามารถวัดค่าได้ 9 แกนคือ วัดค่าความเร็วเชิงมุม(gyroscope) 3 แกน วัดค่าความเร่งเชิงเส้น(accelerometer) 3 แกน และวัดค่าสนามแม่เหล็กโลก(magnetometer) 3 แกน ซึ่งเซนเซอร์ตัวนี้จะติดตั้งบริเวณส่วนของลำตัวหุ่นยนต์ เนื่องจากว่าจะเป็นจุดที่สามารถบ่งบอกได้ถึงการเคลื่อนที่ และมุ่งเอียงของหุ่นยนต์ในขณะนั้นได้ดีที่สุด<sup>7</sup>



รูปที่ 3.7: แสดงเซนเซอร์ IMU MPU9250

### Wi-Fi Adapter

ตัวรับสัญญาณ wifi ชนิดพกพาเพื่อใช้ในการติดต่อสื่อสารระหว่างคอมพิวเตอร์ที่ติดตั้งอยู่บนตัวของหุ่นยนต์ และคอมพิวเตอร์ที่เป็นตัวสั่งการซึ่งอยู่นอกตัวของหุ่นยนต์ ซึ่งในงานวิจัยนี้ จะใช้ส่งข้อมูลที่ได้หลังจากการประมวลผลบนคอมพิวเตอร์ไปยังตัวหุ่นยนต์ เช่น การวางแผนการเดิน การคำนวนพลศาสตร์ของหุ่นยนต์ และอื่นๆ โดยการส่งข้อมูลไปยังคอมพิวเตอร์ที่อยู่บนตัวหุ่นยนต์นั้นจะมีตัวกลางในการรับส่งสัญญาณ คือ ตัวกระจายสัญญาณ (wifi router)



รูปที่ 3.8: ตัวรับสัญญาณ wifi ของ RaspberryPi

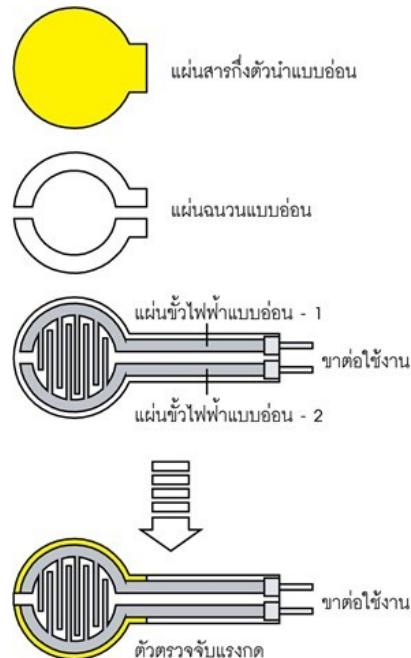


รูปที่ 3.9: ตัวกระจายและรับส่งสัญญาณ wifi

<sup>7</sup>MPU-9250 [http://www.arduino.com/en/gy-series-axis-accelerometers/6924-gy9255-mpu9255-sensor-module-alternative-mpu9150-mpu9250-3809200640200.html]

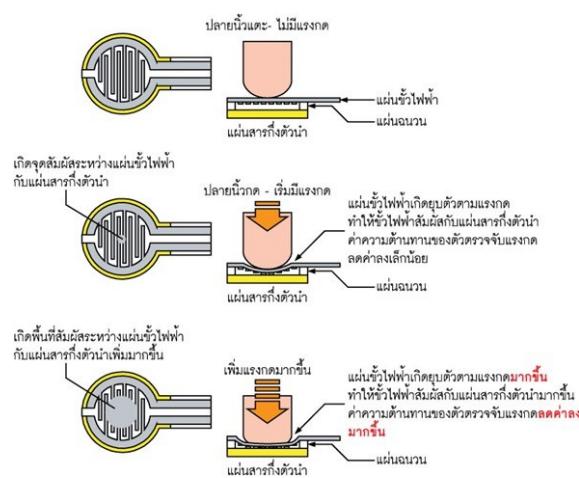
### Ground contact sensor

เซนเซอร์ตรวจหน้าสัมผัสที่พื้นเป็นเซนเซอร์ที่ถูกติดตั้งบริเวณฝ่าเท้า เพื่อตรวจสอบการเดินของหุ่นยนต์ขึ้นมาอยู่ด้วยจะมีการสัมผัสของฝ่าเท้าของหุ่นยนต์กับพื้นหรือไม่ ซึ่งในงานวิจัยนี้ได้ใช้หลักการตัวตรวจจับแรงกดแบบค่าความด้านทันทนาหรือ Force Sensing Resistor (FSR) ที่ใช้เทคโนโลยีฟิล์มโพลีเมอร์แบบหนา (Polymer Thick Film) โดยแรงดันไฟฟ้าที่ตกคร่อมตัวตรวจจับจะลดลง เมื่อมีแรงกดมากระทำบนแผ่นตรวจจับ มีโครงสร้างของตัวตรวจจับแสดง ดังรูปที่ 3.10



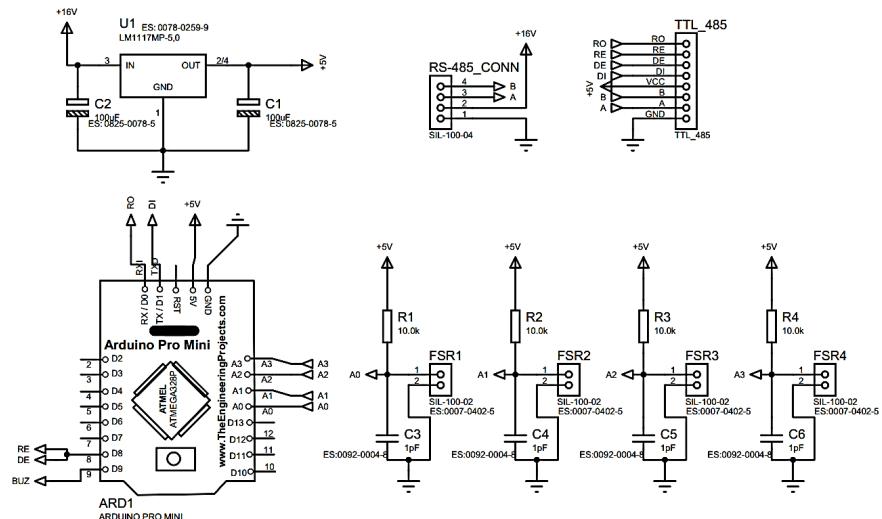
รูปที่ 3.10: ลักษณะโครงสร้างของตัวตรวจจับแรงกด FSR

ประกอบด้วยแผ่นสารกึ่งตัวนำแบบอ่อนที่เป็นตัวกำหนดค่าความด้านทันทนาไฟฟ้าประกอบ เข้ากับแผ่นชั้วไฟฟ้าแบบอ่อน โดยมีแผ่นฉนวนแบบอ่อนคั่นกลาง ทำให้เกิดค่าความด้านทันทนาไฟฟ้าขึ้นระหว่างขาต่อใช้งาน เมื่อมีการกดลงบนแผ่นชั้วไฟฟ้า จะทำให้เกิดการสัมผัสระหว่างสารกึ่งตัวนำกับชั้วไฟฟ้า ส่งผลให้ค่าความด้านทันทนาไฟฟ้าเกิดการเปลี่ยนแปลง ดังแสดงกระบวนการการทำงานในรูปที่ 3.11

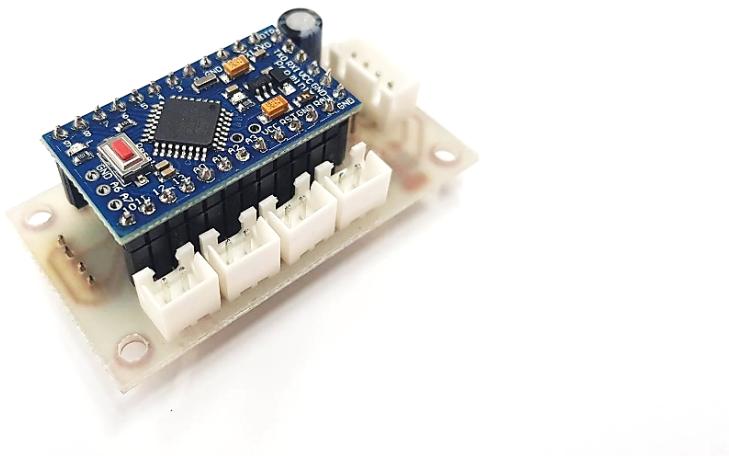


รูปที่ 3.11: การทำงานของตัวตรวจจับแรงกด FSR

แนวคิดการออกแบบหลัก คือการออกแบบให้สามารถติดตั้งกับตัวหุ่นยนต์ได้โดย ไม่ต้องเชื่อมต่อสายไฟ และสายส่งข้อมูลใหม่ โดยใช้สายไฟไฟเลี้ยง และสายสัญญาณชุดเดียวกับตัวขับเคลื่อน Dynamixel Servo Motor ซึ่งมีการติดต่อกันในลักษณะเป็นบัสแบบ RS-485 ดังนั้นแล้วผู้เขียนจึงเลือกที่จะทำโมดูลขึ้นมาใหม่ 1 โมดูล เพื่อที่ใช้ในการอ่านค่า Ground Contact Sensor ของหุ่นยนต์โดยเฉพาะ โดยมีการติดต่อรูปแบบบัส RS-485 ให้ลักษณะการติดต่อสื่อสาร(Protocol) เดียวกับตัวขับเคลื่อน Dynamixel และมีการพัฒนาจาก Arduino ซึ่งให้สามารถอ่านค่าได้ทั้ง Analog และ Digital ได้ อีกทั้งรองรับการต่อ Sensor แบบ Force sensitive resistor จำนวน 4 ตัว.



รูปที่ 3.12: Schematic ของวงจร Ground Contact Sensor



รูปที่ 3.13: แองเจิล Ground Contact Sensor ที่ประกอบเสร็จแล้ว

เซนเซอร์ที่เลือกใช้คือ Force Sensitive Resistor (FSR) เป็นเซนเซอร์ที่มีค่าความต้านทานภายในตัวเองโดยเซนเซอร์นี้มีหลักการทำงานคือ ค่าความต้านทานทางไฟฟ้าของตัวเซนเซอร์จะเปลี่ยนแปลงเมื่อมีแรงเข้ามากจะทำให้กับหน้าสัมผัส เมื่อมีแรงเข้ามากจะทำมาก จะทำให้ค่าความต้านทานต่ำ หากไม่มีแรงเข้ามาจะทำจะทำให้มีค่าความต้านทานสูง และเมื่อมีการนำเซนเซอร์นี้มาต่อ กับตัวต้านทานที่มีค่าคงที่ ในรูปแบบของ Voltage Divider ดังรูปที่ 3.12 จะทำให้สามารถอ่านค่าแรงดันไฟฟ้าที่เปลี่ยนแปลงตามแรงที่เกิดขึ้นกับหน้าสัมผัสของเซนเซอร์ FSR ได้

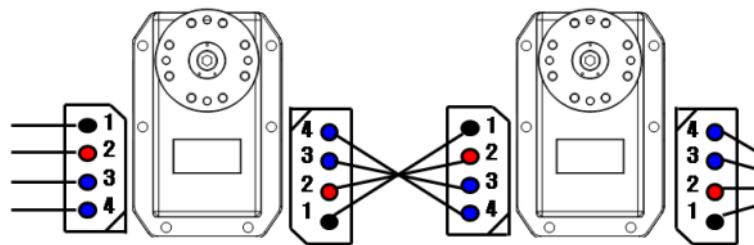


รูปที่ 3.14: Force Sensitive Resistor (FSR) ขนาด 0.5 นิ้ว

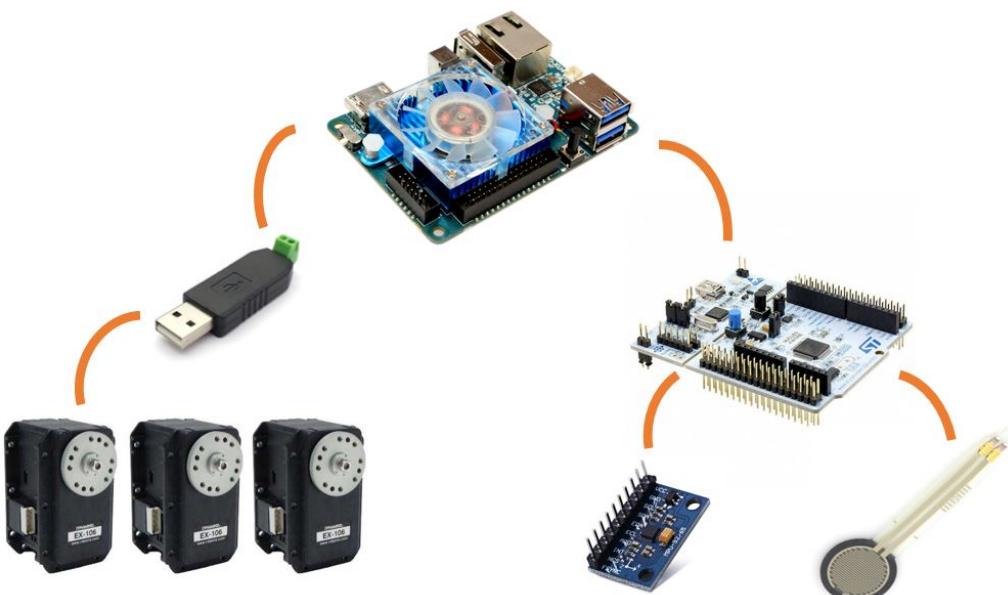
ข้อดีของ FSR นั้นคือ เป็นเซนเซอร์ที่ถูกพัฒนาและออกแบบมาเพื่อใช้สำหรับการวัดแรงโดยตรง จึงทำให้ใช้งานได้ง่าย และสะดวก ในราคาน้ำหนักกว่า เมื่อเทียบกับเซนเซอร์ Load cell ที่มีราคาสูงและการใช้งานจำเป็นที่จะต้องมีวงจรขยายสัญญาณที่ใช้ในการอ่านค่าการบิดของวัสดุจาก แต่ FSR นั้นมีข้อเสีย เช่น กันคือ ความไม่ทนทานต่อการขีดข่วน เนื่องจากตัวเซนเซอร์ถูกทำมาจากฟิล์มพลาสติกบางๆ ซึ่งหากเกิดการขีดข่วนเกิดขึ้นแล้วอาจทำให้ฟิล์มฉีกขาดได้ หากฟิล์มขาดจะทำให้ค่าความต้านทานออกมากไม่เหมือนเดิม ดังนั้นทางผู้เขียนจึงเลือกที่จะออกแบบโครงสร้างสำหรับเซนเซอร์ FSR เพื่อป้องกันจากการถูกขีดข่วนจากภายนอก

### 3.2.4 การเชื่อมต่อตัวขับเคลื่อนและตัวรับสัญญาณ

โครงสร้างของหุ่นยนต์ขี่วนอยู่ที่อุปกรณ์ที่มีขาสองข้างทำให้เกิดองศาอิสระ 12 องศาอิสระ จึงใช้ดิจิตอลเซอร์วอทั้งหมด 12 ตัว ดิจิตอลเซอร์วอทุกตัวเชื่อมต่อกันแบบเดซี่เชน (daisy chain) ดังรูปที่ 3.15 ข้างนี้ของมอเตอร์ตัวแรกเชื่อมต่อกับแบตเตอรี่ 12V และอีกข้างต่อกับ USB2RS485 เพื่อต่อไปยังตัวประมวลผลระดับสูง (Odroid) และเซนเซอร์หน่วยวัดความเร็วอย่างกับเซนเซอร์ตรวจจับหน้าสัมผัสที่พื้นที่เชื่อมต่อกับตัวประมวลผลระดับต่ำ (Nucleo F411RE) ดังรูปที่ 3.16



รูปที่ 3.15: การเชื่อมต่อ กันระหว่าง ดิจิตอลเซอร์วอท์



รูปที่ 3.16: การเชื่อมต่อระหว่าง ตัวรับสัญญาณ ตัวประมวลผล และตัวขับเคลื่อน

### 3.2.5 การตั้งค่ามอเตอร์



Roboplus

หลังจากที่เราเชื่อมต่อมอเตอร์เข้ากับระบบแล้วก็ต้องมีการตั้งค่ามอเตอร์ก่อน โดยการตั้งค่ามอเตอร์นั้นจะใช้โปรแกรม Roboplus เป็นเครื่องมือที่ช่วยให้เราสามารถติดต่อกับมอเตอร์ได้ แต่ใช้ได้เฉพาะใน Windows เท่านั้น ซึ่งสามารถดาวน์โหลดได้จากหน้าเว็บ Robotis เมื่อเราดาวน์โหลดโปรแกรมและติดตั้งเรียบร้อยแล้ว ให้เชื่อมต่อ มอเตอร์กับ USB2RS485 ที่ลําตัว และทำตามขั้นตอน ตามภาพ



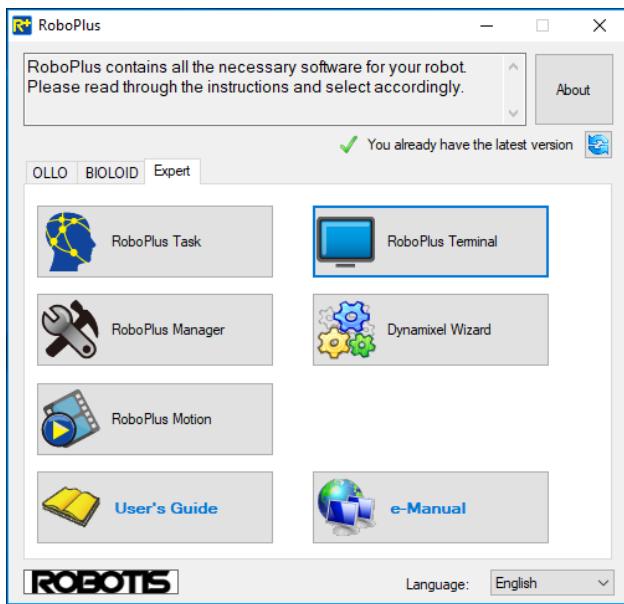
ต่อมอเตอร์เข้ากับคอมพิวเตอร์ด้วย USB2RS485



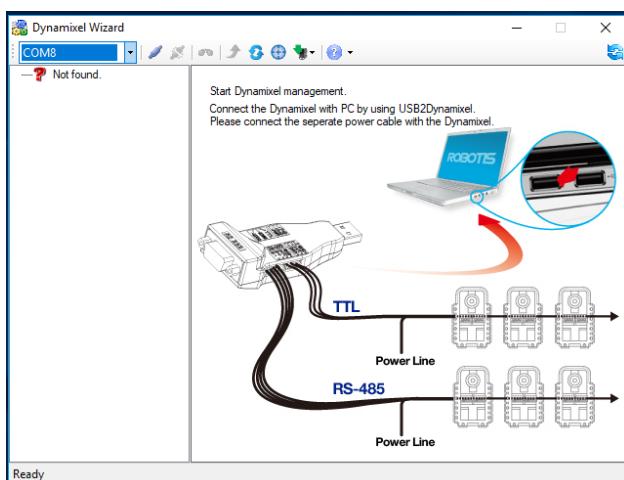
ต่อมอเตอร์เข้ากับพาวเวอร์ซัพพลาย



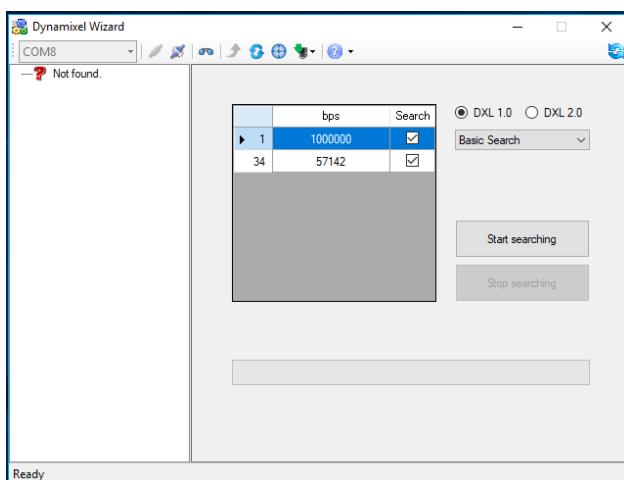
เปิดโปรแกรม Roboplus ขึ้นมา



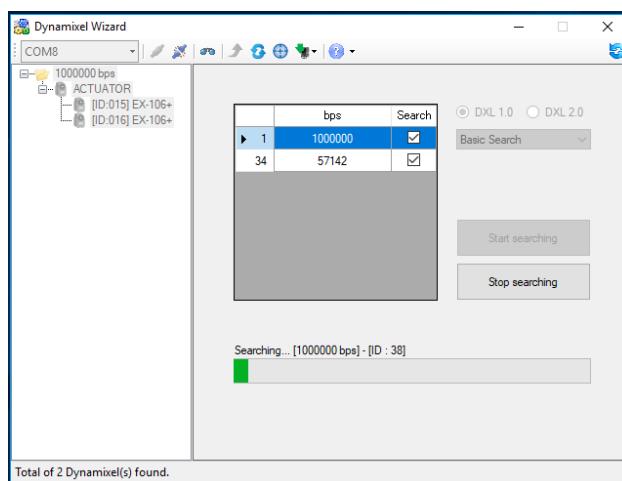
กดเข้าไปที่ Dynamixel Wizard



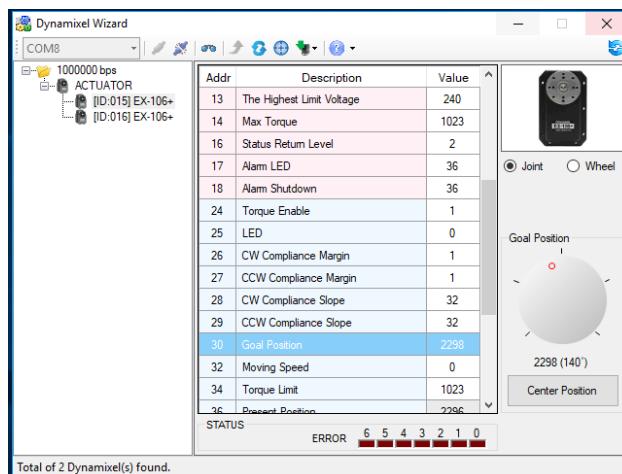
เลือก COM Port ให้ตรงกับ USB2RS485 จากนั้นกด Connect



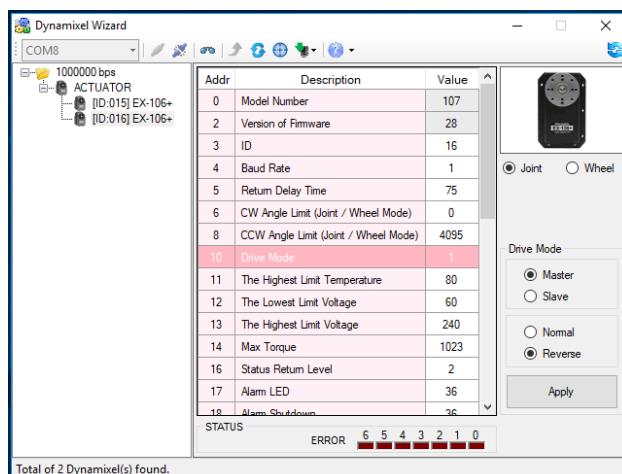
ติ๊กถูกที่ช่อง 1Mbps และกด Start searching



เมื่อเทินทางด้านซ้ายมือผลลัพธ์ ID ของเตอร์ขึ้นมา หากขึ้นแล้วก็สามารถกด Stop Searching ได้



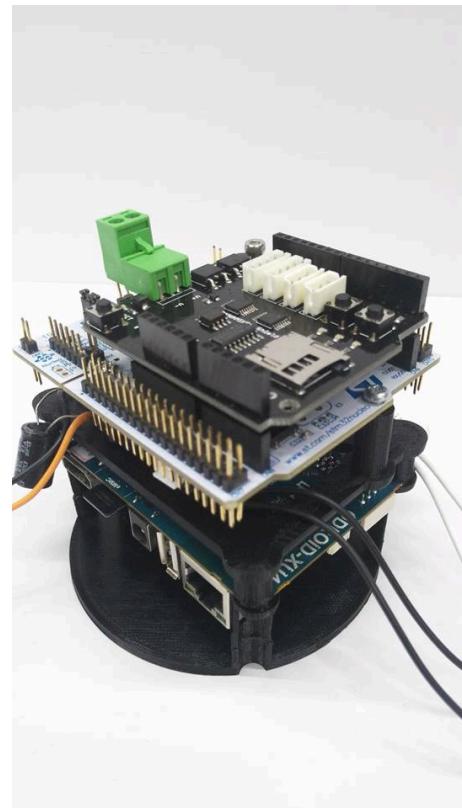
ทดสอบสั่งการมอเตอร์ที่ Addr 30 Goal position ว่าทิศทางถูกต้องหรือไม่



ถ้าทิศทางไม่ถูกต้องสามารถที่จะปรับได้ที่ Addr 10 Drive mode

### 3.2.6 การเชื่อมต่อบอร์ดประมวลผล

การเชื่อมต่อระหว่างบอร์ดประมวลผลระดับล่างกับบอร์ดประมวลผลระดับสูง โดยจะเชื่อมต่อกันผ่านสาย USB และส่งข้อมูลหากันผ่าน Serial โดยใช้ ROSserial นอกจากจะมีบอร์ดประมวลผลแล้วยังมีบอร์ดที่เอาไว้ใช้สำหรับควบคุม dynamixel servo เพื่อเพิ่มความสามารถให้ตัวประมวลผลควบคุมระดับล่างเป็นตัวสั่งการมอเตอร์ ดูรูปที่ 3.17



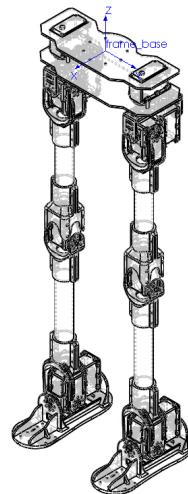
รูปที่ 3.17: การเชื่อมต่อกันระหว่างตัวประมวลผล

### 3.2.7 Dynamic properties

ข้อมูลพลศาสตร์ของหุ่นยนต์จะเอาไว้ใช้ในการทำ Simulation บนระบบ ROS และเอาไปใช้ในการคำนวณทางคณิตศาสตร์ได้ โดยข้อมูลนี้เอามาจาก SolidWorks แล้วปรับให้มีค่าใกล้เคียงกับของจริงมากที่สุด

ข้อมูลที่จำเป็นในการใช้งานจะประกอบไปด้วย มวล จุดศูนย์กลางมวล และโมเมนต์ความเฉื่อย

#### Overall Humanoid

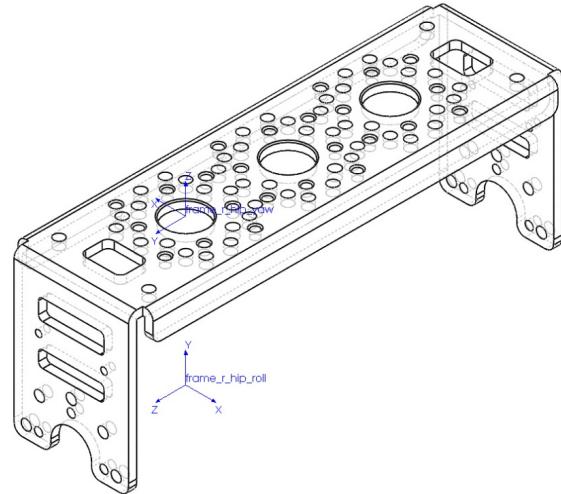


รูปที่ 3.18: ภาพแสดงช่วงล่างทั้งตัว

Link	All Link
Mass (kg)	3.31477475
CoM X (m)	-0.00855772
CoM Y (m)	0.00000000
CoM Z (m)	-0.33375492
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 3.4: ตารางแสดงค่าพารามิเตอร์ทั้งตัว

### Right Hip Yaw



รูปที่ 3.19: ภาพแสดงก้านต่อ Right Hip Yaw

Link	r_hip_yaw
Mass (kg)	0.09100000
CoM X (m)	0.00000000
CoM Y (m)	0.02864983
CoM Z (m)	-0.02500000
Inertia Ixx	0.00014158
Inertia Ixy	0.00000000
Inertia Ixz	0.00000000
Inertia Iyy	0.00014316
Inertia Iyz	0.00000000
Inertia Izz	0.00002022

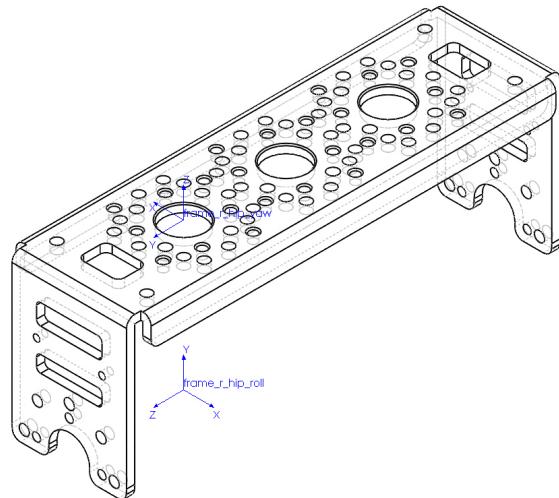
(ก) DH Parameter

Link	r_hip_yaw
Mass (kg)	0.09100000
CoM X (m)	0.00000000
CoM Y (m)	-0.02500000
CoM Z (m)	-0.00735017
Inertia Ixx	0.00014158
Inertia Ixy	0.00000000
Inertia Ixz	0.00000000
Inertia Iyy	0.00002022
Inertia Iyz	0.00000000
Inertia Izz	0.00014316

(ข) URDF

ตารางที่ 3.5: ตารางแสดงค่าพารามิเตอร์ Right Hip Yaw

### Left Hip Yaw



รูปที่ 3.20: ภาพแสดงก้านต่อ Left Hip Yaw

Link	l_hip_yaw
Mass (kg)	0.09100000
CoM X (m)	0.00000000
CoM Y (m)	0.02864983
CoM Z (m)	-0.02500000
Inertia Ixx	0.00014158
Inertia Ixy	0.00000000
Inertia Ixz	0.00000000
Inertia Iyy	0.00014316
Inertia Iyz	0.00000000
Inertia Izz	0.00002022

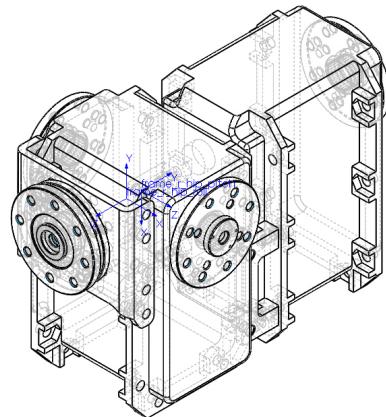
(ก) DH Parameter

Link	l_hip_yaw
Mass (kg)	0.09100000
CoM X (m)	0.00000000
CoM Y (m)	0.02500000
CoM Z (m)	-0.00735017
Inertia Ixx	0.00014158
Inertia Ixy	0.00000000
Inertia Ixz	0.00000000
Inertia Iyy	0.00002022
Inertia Iyz	0.00000000
Inertia Izz	0.00014316

(ข) URDF

ตารางที่ 3.6: ตารางแสดงค่าพารามิเตอร์ Left Hip Yaw

### Right Hip Roll



รูปที่ 3.21: ภาพแสดงก้านต่อ Right Hip Roll

Link	r_hip_roll
Mass (kg)	0.34300000
CoM X (m)	0.01526237
CoM Y (m)	0.02152630
CoM Z (m)	0.00000000
Inertia Ixx	0.00026846
Inertia Ixy	0.00000219
Inertia Ixz	-0.00000081
Inertia Iyy	0.00014760
Inertia Iyz	0.00000000
Inertia Izz	0.00032448

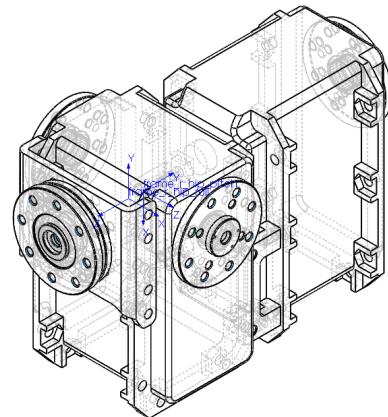
(ก) DH Parameter

Link	r_hip_roll
Mass (kg)	0.34300000
CoM X (m)	0.00000000
CoM Y (m)	-0.01526237
CoM Z (m)	-0.02652630
Inertia Ixx	0.00032448
Inertia Ixy	0.00000081
Inertia Ixz	0.00000000
Inertia Iyy	0.00026846
Inertia Iyz	0.00000219
Inertia Izz	0.00014760

(ข) URDF

ตารางที่ 3.7: ตารางแสดงค่าพารามิเตอร์ Right Hip Roll

### Left Hip Roll



รูปที่ 3.22: ภาพแสดงก้านต่อ Left Hip Roll

Link	l_hip_roll
Mass (kg)	0.34300000
CoM X (m)	0.01526237
CoM Y (m)	0.02152630
CoM Z (m)	0.00000000
Inertia Ixx	0.00026846
Inertia Ixy	0.00000219
Inertia Ixz	-0.00000081
Inertia Iyy	0.00014760
Inertia Iyz	0.00000000
Inertia Izz	0.00032448

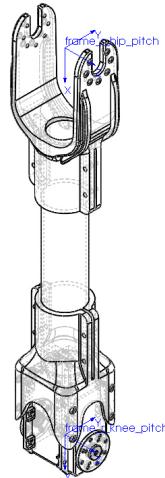
(ก) DH Parameter

Link	l_hip_roll
Mass (kg)	0.34300000
CoM X (m)	0.00000000
CoM Y (m)	-0.01526237
CoM Z (m)	-0.02652630
Inertia Ixx	0.00032448
Inertia Ixy	0.00000081
Inertia Ixz	0.00000000
Inertia Iyy	0.00026846
Inertia Iyz	0.00000219
Inertia Izz	0.00014760

(ข) URDF

ตารางที่ 3.8: ตารางแสดงค่าพารามิเตอร์ Left Hip Roll

### Right Hip Pitch



รูปที่ 3.23: ภาพแสดงก้านต่อ Right Hip Pitch

Link	r_hip_pitch
Mass (kg)	0.31800000
CoM X (m)	-0.07862011
CoM Y (m)	0.00000000
CoM Z (m)	0.00000000
Inertia Ixx	0.00011525
Inertia Ixy	0.00000000
Inertia Ixz	0.00000078
Inertia Iyy	0.00254669
Inertia Iyz	0.00000000
Inertia Izz	0.00250848

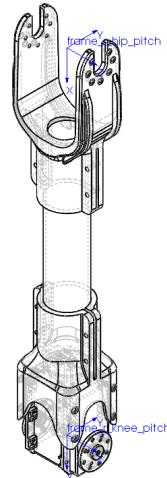
(ก) DH Parameter

Link	r_hip_pitch
Mass (kg)	0.31800000
CoM X (m)	0.22137989
CoM Y (m)	0.00000000
CoM Z (m)	0.00000000
Inertia Ixx	0.00011525
Inertia Ixy	0.00000000
Inertia Ixz	0.00000078
Inertia Iyy	0.00254669
Inertia Iyz	0.00000000
Inertia Izz	0.00250848

(ข) URDF

ตารางที่ 3.9: ตารางแสดงค่าพารามิเตอร์ Right Hip Pitch

### Left Hip Pitch



รูปที่ 3.24: ภาพแสดงก้านต่อ Left Hip Pitch

Link	<i>l_hip_pitch</i>
Mass (kg)	0.31800000
CoM X (m)	-0.07862011
CoM Y (m)	0.00000000
CoM Z (m)	0.00000000
Inertia Ixx	0.00011525
Inertia Ixy	0.00000000
Inertia Ixz	0.00000078
Inertia Iyy	0.00254669
Inertia Iyz	0.00000000
Inertia Izz	0.00250848

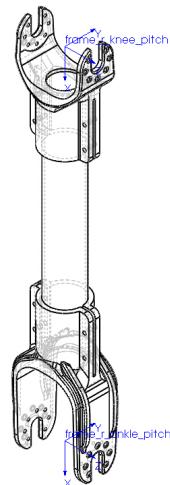
(ก) DH Parameter

Link	<i>l_hip_pitch</i>
Mass (kg)	0.31800000
CoM X (m)	0.22137989
CoM Y (m)	0.00000000
CoM Z (m)	0.00000000
Inertia Ixx	0.00011525
Inertia Ixy	0.00000000
Inertia Ixz	0.00000078
Inertia Iyy	0.00254669
Inertia Iyz	0.00000000
Inertia Izz	0.00250848

(ข) URDF

ตารางที่ 3.10: ตารางแสดงค่าพารามิเตอร์ Left Hip Pitch

### Right Knee Pitch



รูปที่ 3.25: ภาพแสดงก้านต่อ Right Knee Pitch

Link	r_knee_pitch
Mass (kg)	0.13800000
CoM X (m)	-0.15211782
CoM Y (m)	0.00000000
CoM Z (m)	0.00000000
Inertia Ixx	0.00011525
Inertia Ixy	0.00000000
Inertia Ixz	0.00000000
Inertia Iyy	0.00127592
Inertia Iyz	0.00000000
Inertia Izz	0.00124960

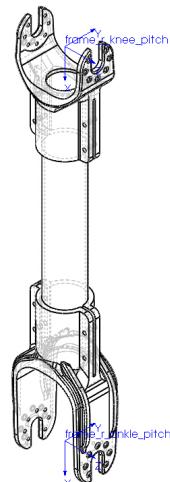
(ก) DH Parameter

Link	r_knee_pitch
Mass (kg)	0.13800000
CoM X (m)	0.16288218
CoM Y (m)	0.00000000
CoM Z (m)	0.00000000
Inertia Ixx	0.00005794
Inertia Ixy	0.00000000
Inertia Ixz	0.00000000
Inertia Iyy	0.00127592
Inertia Iyz	0.00000000
Inertia Izz	0.00124960

(ข) URDF

ตารางที่ 3.11: ตารางแสดงค่าพารามิเตอร์ Right Knee Pitch

### Left Knee Pitch



รูปที่ 3.26: ภาพแสดงก้านต่อ Left Knee Pitch

Link	l_knee_pitch
Mass (kg)	0.13800000
CoM X (m)	-0.15211782
CoM Y (m)	0.00000000
CoM Z (m)	0.00000000
Inertia Ixx	0.00011525
Inertia Ixy	0.00000000
Inertia Ixz	0.00000000
Inertia Iyy	0.00127592
Inertia Iyz	0.00000000
Inertia Izz	0.00124960

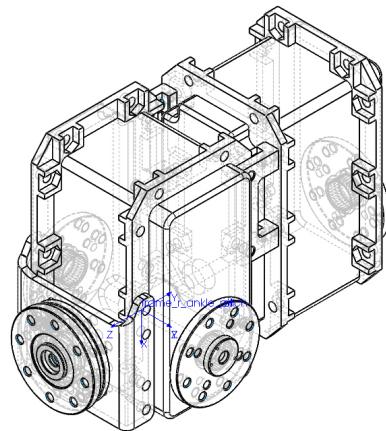
(ก) DH Parameter

Link	l_knee_pitch
Mass (kg)	0.13800000
CoM X (m)	0.16288218
CoM Y (m)	0.00000000
CoM Z (m)	0.00000000
Inertia Ixx	0.00005794
Inertia Ixy	0.00000000
Inertia Ixz	0.00000000
Inertia Iyy	0.00127592
Inertia Iyz	0.00000000
Inertia Izz	0.00124960

(ข) URDF

ตารางที่ 3.12: ตารางแสดงค่าพารามิเตอร์ Left Knee Pitch

### Right Ankle Pitch



รูปที่ 3.27: ภาพแสดงก้านต่อ Right Ankle Pitch

Link	r_ankle_pitch
Mass (kg)	0.33138738
CoM X (m)	-0.01526237
CoM Y (m)	0.00000000
CoM Z (m)	-0.02152630
Inertia Ixx	0.00025937
Inertia Ixy	0.00000000
Inertia Ixz	0.00000079
Inertia Iyy	0.00031349
Inertia Iyz	0.00000000
Inertia Izz	0.00014261

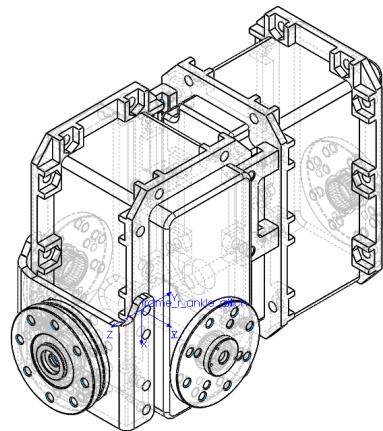
(ก) DH Parameter

Link	r_ankle_pitch
Mass (kg)	0.33138738
CoM X (m)	-0.01526237
CoM Y (m)	0.02152630
CoM Z (m)	0.00000000
Inertia Ixx	0.00025937
Inertia Ixy	0-0.00000212
Inertia Ixz	0.00000079
Inertia Iyy	0.00014261
Inertia Iyz	0.00000000
Inertia Izz	0.00031349

(ข) URDF

ตารางที่ 3.13: ตารางแสดงค่าพารามิเตอร์ Right Ankle Pitch

### Left Ankle Pitch



รูปที่ 3.28: ภาพแสดงก้านต่อ Left Ankle Pitch

Link	l_ankle_pitch
Mass (kg)	0.33138738
CoM X (m)	-0.01526237
CoM Y (m)	0.00000000
CoM Z (m)	-0.02152630
Inertia Ixx	0.00025937
Inertia Ixy	0.00000000
Inertia Ixz	0.00000079
Inertia Iyy	0.00031349
Inertia Iyz	0.00000000
Inertia Izz	0.00014261

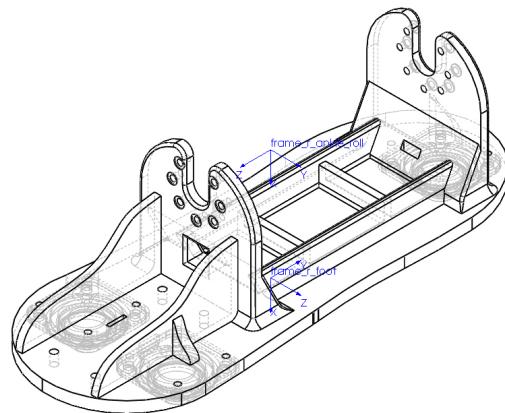
(ก) DH Parameter

Link	l_ankle_pitch
Mass (kg)	0.33138738
CoM X (m)	-0.01526237
CoM Y (m)	0.02152630
CoM Z (m)	0.00000000
Inertia Ixx	0.00025937
Inertia Ixy	0-0.00000212
Inertia Ixz	0.00000079
Inertia Iyy	0.00014261
Inertia Iyz	0.00000000
Inertia Izz	0.00031349

(ข) URDF

ตารางที่ 3.14: ตารางแสดงค่าพารามิเตอร์ Left Ankle Pitch

### Right Ankle Roll



รูปที่ 3.29: ภาพแสดงก้านต่อ Right Ankle Roll

Link	r_ankle_roll
Mass (kg)	0.10500000
CoM X (m)	-0.01454118
CoM Y (m)	-0.00034576
CoM Z (m)	-0.00019548
Inertia Ixx	0.00034591
Inertia Ixy	-0.00000857
Inertia Ixz	-0.00000013
Inertia Iyy	0.00004813
Inertia Iyz	-0.00000120
Inertia Izz	0.00032705

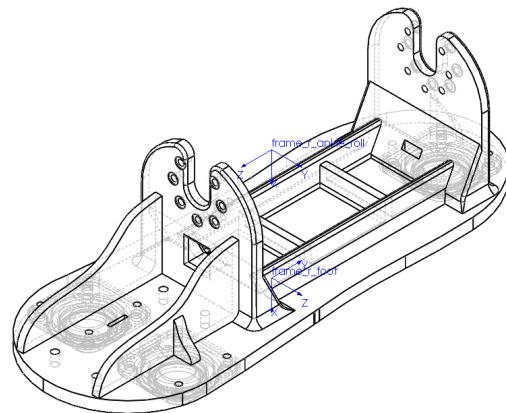
(ก) DH Parameter

Link	r_ankle_roll
Mass (kg)	0.10500000
CoM X (m)	0.03625882
CoM Y (m)	-0.00019548
CoM Z (m)	0.00034576
Inertia Ixx	0.00034591
Inertia Ixy	-0.00000013
Inertia Ixz	0.00000857
Inertia Iyy	0.00032705
Inertia Iyz	0.00000120
Inertia Izz	0.00004813

(ข) URDF

ตารางที่ 3.15: ตารางแสดงค่าพารามิเตอร์ Right Ankle Roll

### Left Ankle Roll



รูปที่ 3.30: ภาพแสดงก้านต่อ Left Ankle Roll

Link	l_ankle_roll
Mass (kg)	0.10500000
CoM X (m)	-0.01454118
CoM Y (m)	-0.00034576
CoM Z (m)	-0.00019548
Inertia Ixx	0.00034591
Inertia Ixy	-0.00000857
Inertia Ixz	-0.00000013
Inertia Iyy	0.00004813
Inertia Iyz	-0.00000120
Inertia Izz	0.00032705

(ก) DH Parameter

Link	l_ankle_roll
Mass (kg)	0.10500000
CoM X (m)	0.03625882
CoM Y (m)	-0.00019548
CoM Z (m)	0.00034576
Inertia Ixx	0.00034591
Inertia Ixy	-0.00000013
Inertia Ixz	0.00000857
Inertia Iyy	0.00032705
Inertia Iyz	0.00000120
Inertia Izz	0.00004813

(ข) URDF

ตารางที่ 3.16: ตารางแสดงค่าพารามิเตอร์ Left Ankle Roll

### 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` คือการหากความสัมพันธ์ระหว่าง 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

ในส่วนนี้จะเป็นการอธิบายระบบทางกลของหุ่นยนต์ข้ามมาน้อยเป็นไฟล์ที่ใช้ร่วมกับ 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 3.31

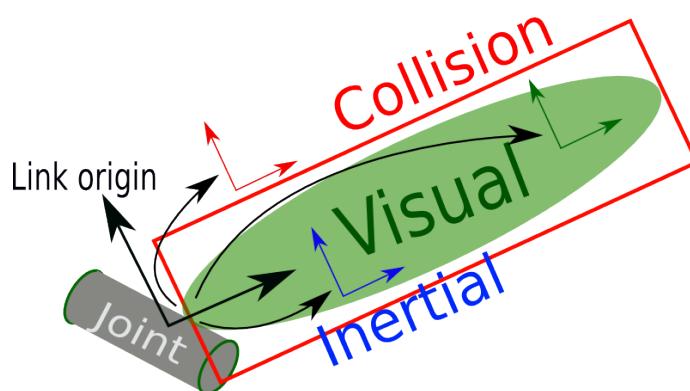
```

<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 3.31: ตัวอย่าง link ใน urdf

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



รูปที่ 3.32: การอธิบาย link ใน URDF ไฟล์

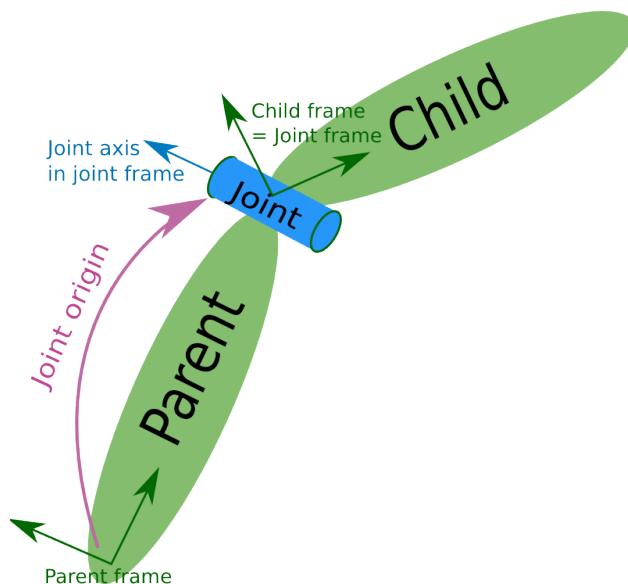
## Joint

อีกส่วนที่สำคัญสำหรับการสร้างไฟล์หุ่นยนต์ด้วย URDF ก็คือ Joint tag โดย tag นี้จะอธิบายถึงความสัมพันธ์ระหว่างก้านต่อสองอัน ส่วนนี้ไม่ได้มีเพียงแค่ทำข้อต่อให้เป็นแบบหมุนได้อย่างเดียว ยังมี Fix, Revolution, Linear และ Planar นอกเหนือจากนี้ เราจึงสามารถที่จะเพิ่มองศาสูงสุดต่ำสุดของข้อต่อ รวมไปถึง dynamic properties ต่างๆ ตามที่เห็นดังรูปที่ 3.33

```
<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 3.33: ตัวอย่าง joint ใน urdf

เมื่อเราเขียน Joint และ Link มารวมกันเราจะต้องพิจารณาว่ามีวารูปแบบเป็นไปตามรูปที่ 3.34 โดยจะมีรายละเอียดของแต่ละข้อต่อ กับ ก้านต่อ ชึ้นส่วนแรกของการสร้างไฟล์ URDF จะมีชื่อว่า base\_link และเฟรม origin จะเป็นเฟรมอ้างอิง เมื่อเราต่อ Joint เข้ากับ Link จะเรียกว่า ก้านต่อที่นำมาติดว่า parent โดยเฟรม origin ของข้อต่อจะอยู่จุดเดียวกับเฟรม origin ของก้านต่อ ในสถานะเดียวกัน ก้านต่อที่นำมาต่อจากข้อต่อ เราจะเรียกว่า child และเฟรม origin ของก้านต่อ child จะอยู่ที่จุดเดียวกับเฟรม origin ของข้อต่อ



รูปที่ 3.34: 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

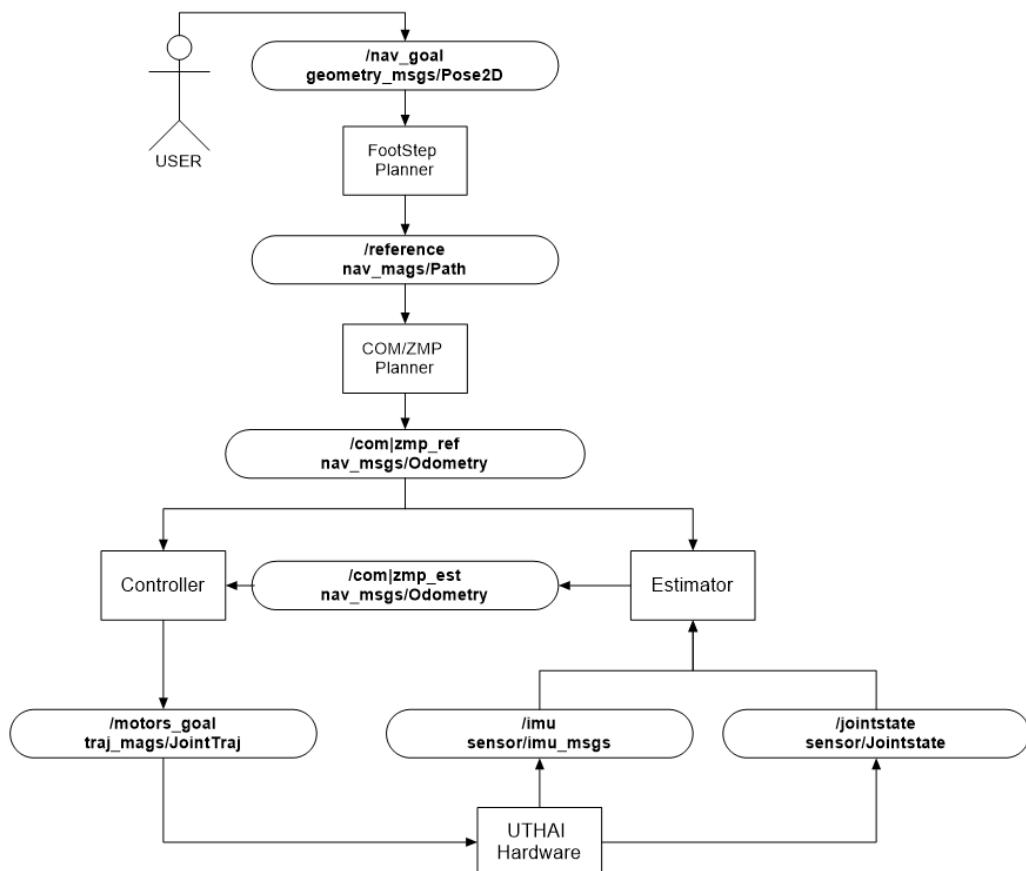
#### **torso**

เป็นเฟรมที่ติดอยู่กับลำตัวซึ่งล่างของหุ่นยนต์โดยจะเป็นตัวที่เชื่อม ขา แขน ตัว หัว เข้ามาไว้ด้วยกัน

### 3.3.3 Box model

### 3.3.4 โครงสร้างการติดต่อสื่อสารระหว่าง Node ใน ROS

การติดต่อสื่อสารกันภายใน ROS นั้นจะใช้การส่ง message หากัน ซึ่ง message แต่ละตัวก็จะใช้ในงานที่ต่างกัน ตามระบบที่ต้องการส่ง จากรูปที่ 3.35 เป็นโครงสร้างการส่งข้อมูลหากันของหุ่นยนต์อิวามานอยด์ ที่ผู้วิจัยได้ออกแบบไว้ โดยเริ่มจากผู้ใช้งานส่งตำแหน่งที่หุ่นยนต์จะต้องเดินไปเป็น Node ที่ทำการคำนวณและสร้างตำแหน่งการวางเท้าของหุ่นยนต์ และหลังจากนั้นจะส่งข้อมูลออกไปเป็น Path เส้นทางไปยัง Node ที่ทำการค้นหาตำแหน่งของ com, zmp ของหุ่นยนต์ เพื่อทำการควบคุมและส่งการหุ่นยนต์ต่อไป



รูปที่ 3.35: การติดต่อสื่อสารระหว่าง Node

#### การบอกตำแหน่งและมุมเอียง

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

geometry_msgs/Point	
float64	x
float64	y
float64	z

ตารางที่ 3.17: Message Geometry Point

geometry_msgs/Quaternion	
float64	x
float64	y
float64	z
float64	w

ตารางที่ 3.18: Message Geometry Quaternion

geometry_msgs/Pose	
geometry_msgs/Point	position
geometry_msgs/Quaternion	orientation

ตารางที่ 3.19: Message Geometry Pose

### การบอกรความเร็วเชิงเส้นและเชิงมุ่ง

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

geometry_msgs/Vector3	
float64	x
float64	y
float64	z

ตารางที่ 3.20: Message Geometry Vector3

geometry_msgs/Twist	
geometry_msgs/Vector3	linear
geometry_msgs/Vector3	angular

ตารางที่ 3.21: 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 3.22: Message Navigation Odometry

### ตำแหน่งของหุ่นยนต์

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

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

ตารางที่ 3.23: Message Geometry Pose2D

### ตำแหน่งการวางแผนทางของหุ่นยนต์

การจะให้หุ่นยนต์นำที้าไปวางในตำแหน่งที่เราต้องการจากที่ได้จากการคำนวณนั้น จะต้องบอกตำแหน่งและบอกมุมเอียงของจุดที่จะไป จากการสร้างจะได้เป็นรายการของที้าซ้ายและขวา โดยอิงจาก ตารางที่ 3.19

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

ตารางที่ 3.24: Message Navigation Path

geometry_msgs/PoseStamped	
std_msgs/Header	header
geometry_msgs/Pose	pose

ตารางที่ 3.25: Message Geometry PoseStamped

### ตำแหน่ง CoM Zmp ของหุ่นยนต์

ใน Message นี้จะใช้อよງ 2 จุดคือ ที่ได้จากการวางแผนของ Node CoM Planner และ Node CoM Estimator โดยทั้งสองจุดใช้ Message เมื่อกันส่งไปยัง Controller เพื่อควบคุมท่าทางต่างๆของหุ่นยนต์ต่อไป Message ที่ใช้คือ Message จากตารางที่ 3.22

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 3.26: Message Trajectory JointTrajectory

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

ตารางที่ 3.27: 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 3.28: Message Sensor JointState

### ค่าเซนเซอร์ฝ่าเท้าของหุ่นยนต์

ที่ฝ่าเท้าของหุ่นยนต์ชีวามาโนyd'm'i' มีเซนเซอร์ที่เอาไว้ใช้ในการอ่าน แรงกดที่ฝ่าเท้า ใช้ในการเอามากกว่า เท้าสัมผัสพื้นหรือไม่

geometry_msgs/Wrench	
geometry_msgs/Vector3	force
geometry_msgs/Vector3	torque

ตารางที่ 3.29: Message Geometry Wrench

### ค่าเซนเซอร์ IMU ของหุ่นยนต์

เซนเซอร์ IMU เป็นเซนเซอร์ที่เอาไว้ใช้ในการวัด ความเร็วเชิงมุม และ ความเร่งเชิงเส้น หากนำทั้งคู่มารวมกันจะสามารถที่จะแปลงให้วัดมุมอิบิของเซนเซอร์ได้ 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 3.30: Message Sensor Imu

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

ตารางที่ 3.31: Message Sensor MagneticField

### 3.4 การออกแบบระบบพื้นฐาน

### 3.4.1 วงศ์โครงสร้างของระบบพื้นฐาน

โครงสร้างระบบพื้นฐานของหุ่นยนต์อิวามานอยด์อุทัย เป็นการวางระบบให้ผู้วิจัยสามารถที่จะช่วยกันพัฒนาได้ และเพื่อทำให้มีความเป็นระบบระเบียบ ง่ายต่อการแก้ไข ปรับปรุง ซึ่งผู้วิจัยได้ออกแบบส่วนการทำงานต่างๆออกเป็นสามส่วน ตามหน้าที่การทำงาน คือ

## Hardware Devices

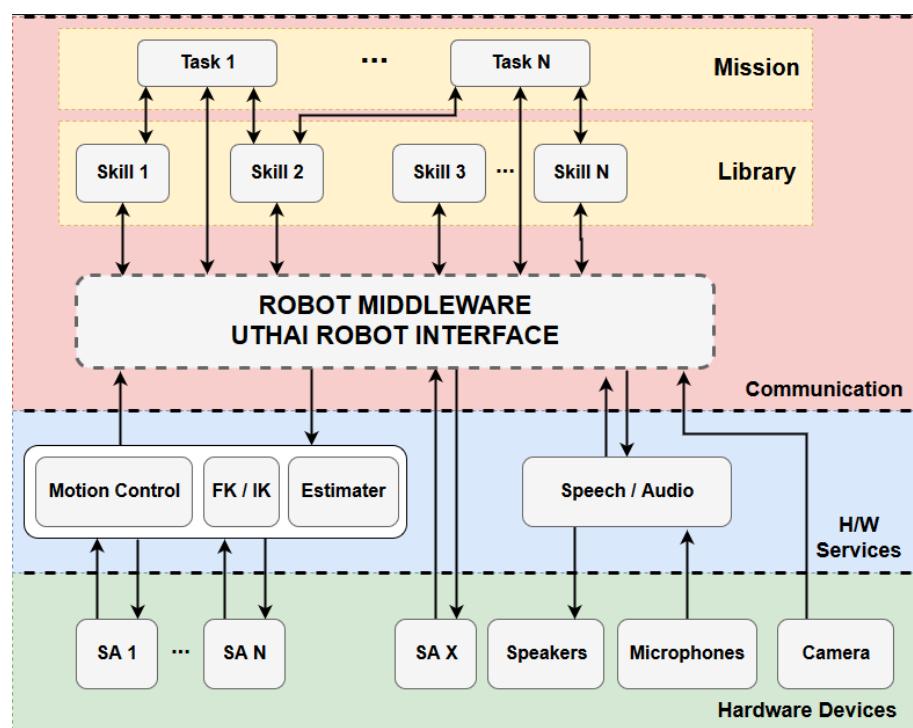
เป็นส่วนของอุปกรณ์haarvard เว็บที่ใช้งานกับหุ่นยนต์ขีมานอยด์ ยกตัวอย่าง เช่น ตัวขับเคลื่อน กล้อง ไมโครโฟน ลำโพง หรืออื่นๆ ซึ่งในส่วนนี้สามารถเปลี่ยนhaarvard เว็บจิรังฯ เป็นระบบจำลองได้

## Hardware Services

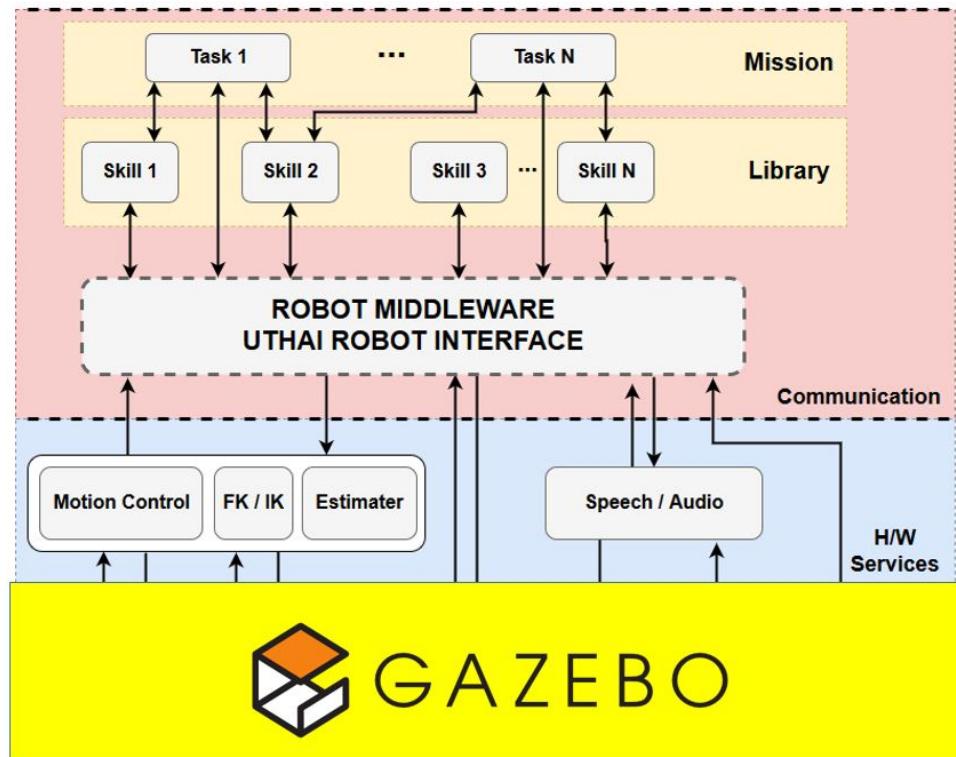
เป็นส่วนของซอฟต์แวร์ที่ช่วยทำให้อุปกรณ์ยาร์ดแวร์สามารถติดต่อสื่อสารกับระบบพื้นฐานของหุ่นยนต์ข้ามอยู่ได้ ซึ่งจะช่วยทำให้การส่งข้อมูลเป็นมาตรฐานเดียวกัน

## Communication

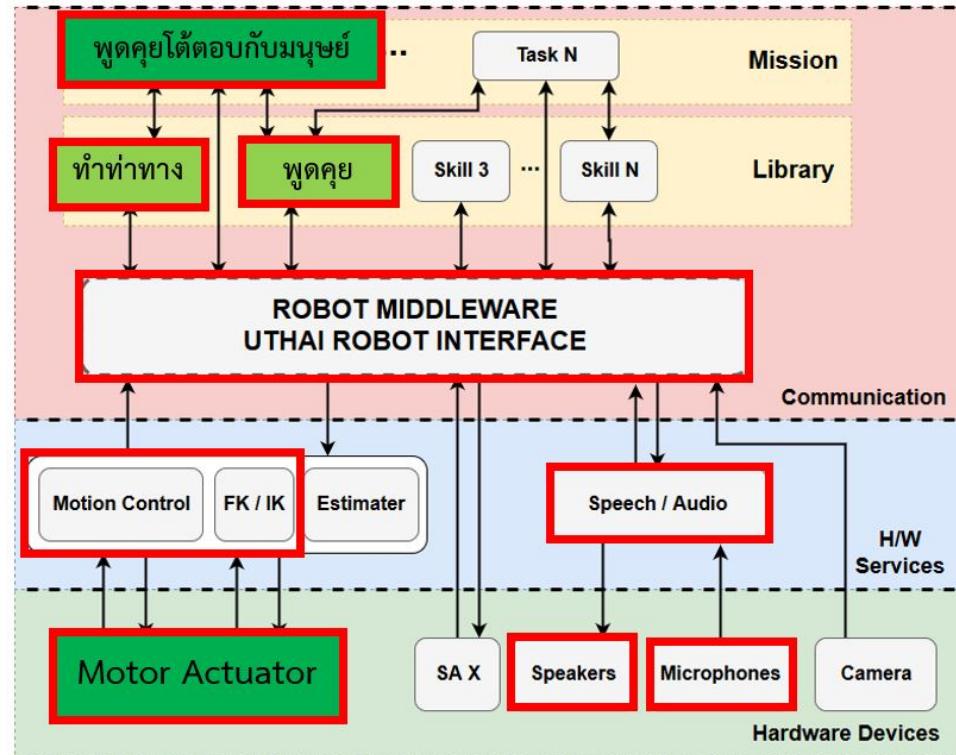
เป็นส่วนของการติดต่อสื่อสารระหว่างซอฟต์แวร์กับฮาร์ดแวร์ผ่านระบบพื้นฐานของทุนยนต์ฮิวมานอยด์ซึ่งจะค่อยจัดการให้ข้อมูลทุกอย่างสามารถเชื่อมต่อกันได้



รูปที่ 3.36: โครงสร้างพื้นฐานของหุ่นยนต์อุทัย



รูปที่ 3.37: ภาพการเปลี่ยนส่วนของชาร์ดแวร์เป็นระบบจำลอง

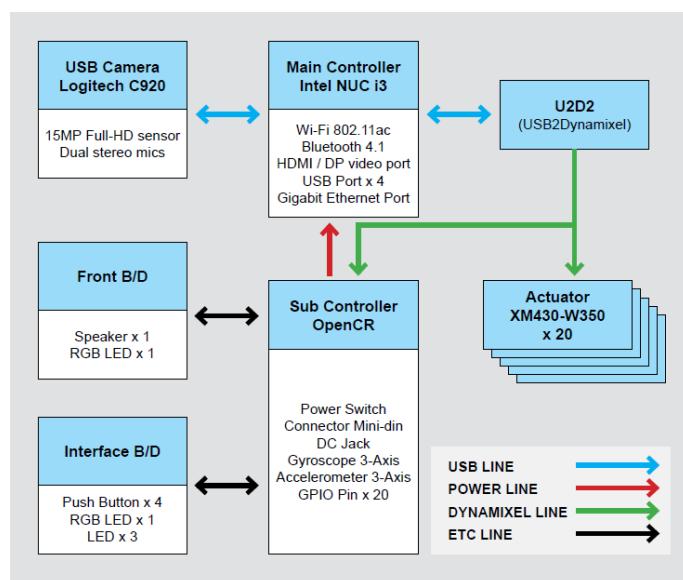


รูปที่ 3.38: ตัวอย่างการนำโครงสร้างพื้นฐานไปประยุกต์ใช้ ในแอปพลิเคชันการพูดคุยโดยตัดประกอบกับมนุษย์

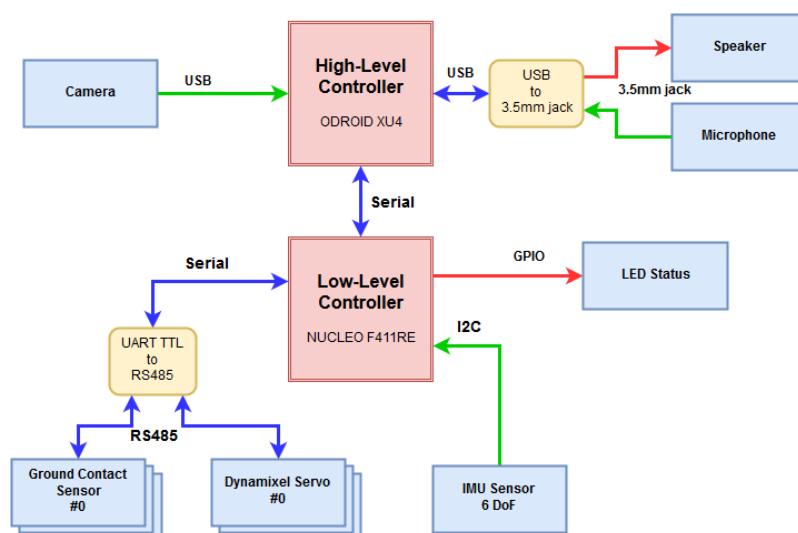
### 3.4.2 ออกแบบสถาปัตยกรรมของหุ่นยนต์

หลักการออกแบบสถาปัตยกรรมของหุ่นยนต์ชีวามโนยด์อุทัย จะออกแบบระบบให้อยู่บนระบบพื้นฐาน ROS เนื่องจากการใช้กรอบการทำงานที่มีประสิทธิภาพ และความยืดหยุ่นสูง จะช่วยทำให้สามารถปรับเปลี่ยนระบบการควบคุมของหุ่นยนต์ชีวามโนยด์ได้ง่ายและรวดเร็ว การออกแบบหน่วยประมวลผลนั้นมีหลากหลายรูปแบบ ดังที่กล่าวไว้ในบทที่ 2 สำหรับในงานวิจัยครั้งนี้ผู้วิจัยได้ศึกษาและพอกกว่าสถาปัตยกรรมที่เหมาะสมกับหุ่นยนต์ชีวามโนยด์อุทัย จะมีลักษณะใกล้เคียงกับหุ่นยนต์ชีวามโนยด์ Robotis OP3 ดังรูปที่ 3.39 ดังนั้นแล้วผู้วิจัยจึงได้แบ่งการประมวลผลออกเป็น 2 ส่วนคือ

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



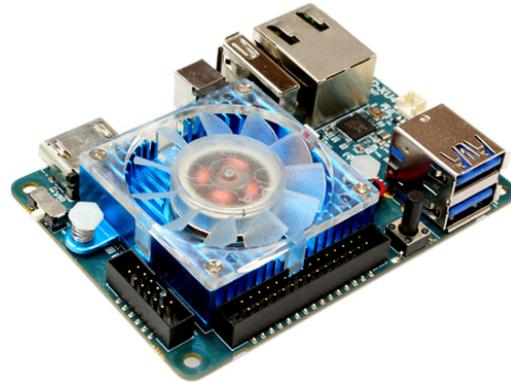
รูปที่ 3.39: สถาปัตยกรรมของหุ่นยนต์ Robotis OP3



รูปที่ 3.40: สถาปัตยกรรมของหุ่นยนต์ชีวามโนยด์อุทัย

### 3.4.2.1 หน่วยประมวลผลควบคุมระดับสูง (High level controller)

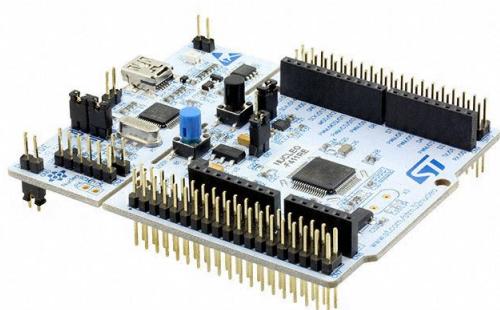
ระบบควบคุมหลักของหุ่นยนต์อุทิ扬นั้นจะอยู่ที่หน่วยประมวลผลขั้นสูง ใช้เป็นบอร์ดคอมพิวเตอร์ ODROID-XU4 ตัวประมวลผลหลักนี้ ทำหน้าที่ในการคำนวณเส้นทางการเดิน ทำให้หุ่นยนต์มีเสถียรภาพในการเดิน ตรวจสอบขัดกันของโครงสร้างของหุ่นยนต์ รวมไปถึงรับค่าข้อมูลตำแหน่ง ความเร็วจากข้อต่อ หลังจากนั้นทำการนำค่าทั้งหมดที่ได้จากการคำนวณ มาแปลงให้อยู่ในรูปของชุดข้อมูล และส่งออกไปให้ระบบบกλา (ROS) ในการส่งต่อไปให้อุปกรณ์อื่นต่อไป



รูปที่ 3.41: บอร์ดคอนโทรลเลอร์ Odroid XU4

### 3.4.2.2 หน่วยประมวลผลควบคุมระดับต่ำ (Low level controller)

ระบบควบคุมขั้นต่ำเป็นหน่วยประมวลผลที่รองรับมาจาก บอร์ดคอมพิวเตอร์ โดยใช้บอร์ดไมโครคอนโทรลเลอร์ Nucleo F411RE เป็นหน่วยประมวลผลขั้นต่ำ สำหรับในการติดต่อกับอุปกรณ์อิเล็กทรอนิกส์ต่างๆ ที่อยู่ภายนอกตัวของหุ่นยนต์ เช่น ค่าเซนเซอร์ที่ฝ่าเท้าซึ่งสามารถบอกได้ว่าควรใช้สมการไหนในการคำนวณพลวัต หรือค่าของเซนเซอร์หน่วยวัดความเร็วymic ความสำคัญมาก ในการทำให้หุ่นยนต์มีความสามารถเดินได้อย่างมีเสถียรภาพ เมื่ออ่านค่าเซนเซอร์ต่างๆได้แล้ว หน่วยประมวลผลขั้นต่ำจะนำค่าที่ได้จากการอ่านเซนเซอร์เหล่านี้แปลงให้อยู่ในลักษณะของชุดข้อมูล และส่งออกไปให้ระบบบกλา (ROS) นอกจากนี้หน่วยประมวลผลขั้นต่ำยังทำหน้าที่รับค่าคำสั่งมาจากระบบบกλา ในการสั่งงานให้หุ่นยนต์มีท่าทางต่างๆได้



รูปที่ 3.42: บอร์ดคอนโทรลเลอร์ Nucleo F411RE

### 3.4.3 จัดทำคู่มือและเอกสารการใช้งาน

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

#### ส่วนที่ทำคู่มือและเอกสารเบื้องต้นคือ

- 1 รายการวัสดุที่ใช้ในการทำหุ่นยนต์ชีวนิร园อยด์อุทัย
- 2 รายละเอียดการเชื่อมต่อระหว่างอุปกรณ์ที่อยู่ในตัวหุ่นยนต์
- 3 รายละเอียดการประกอบชิ้นส่วนทางกล
- 4 รายละเอียดการใช้งานโปรแกรมพื้นฐาน

#### 3.4.3.1 รายการวัสดุที่ใช้ในการทำหุ่นยนต์ชีวนิร园อยด์อุทัย

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 3.32: ตารางแสดงรายการของวัสดุต่าง ๆ

ใช้สำหรับแจกแจงค่าใช้จ่ายเบื้องต้นเท่านั้น ไม่สามารถใช้อ้างอิงงบประมาณแบบละเอียดได้

#### 3.4.3.2 รายละเอียดการเชื่อมต่อระหว่างอุปกรณ์ที่อยู่ในตัวหุ่นยนต์

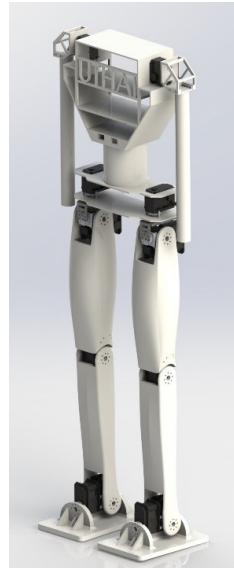
#### 3.4.3.3 รายละเอียดการประกอบชิ้นส่วนทางกล

#### 3.4.3.4 รายละเอียดการใช้งานโปรแกรมพื้นฐาน

## บทที่ 4

### ผลการดำเนินงาน

#### 4.1 โครงสร้างของหุ่นยนต์



รูปที่ 4.1: โครงสร้างหุ่นยนต์ในโปรแกรม 3 มิติ

โครงสร้างของหุ่นยนต์หลักๆ จะแบ่งออกเป็น 2 ส่วนคือ ส่วนท่อนบนและส่วนท่อนล่างโดยส่วนท่อนบนจะประกอบไปด้วย เอว 1 ส่วน ลำตัว 1 ส่วน แขน 2 ส่วน และท่อนล่างจะประกอบไปด้วย สะโพก 1 ส่วน ขา 2 ส่วน น่อง 2 ส่วน ฝ่าเท้า 2 ส่วน ในการเลือกใช้วัสดุนั้นได้แสดงในตาราง 4.2

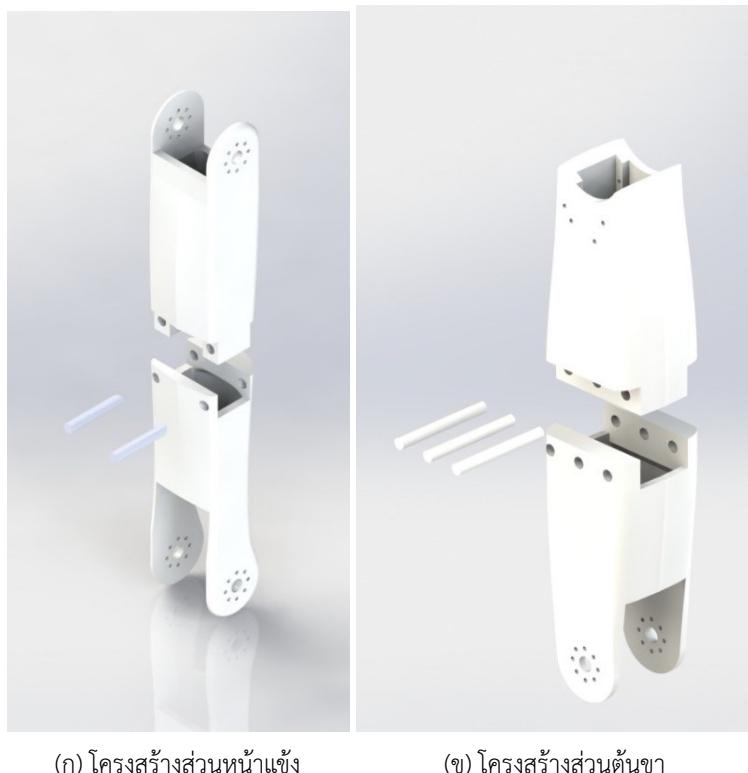
ชื่อส่วน	วัสดุที่ใช้ขึ้นรูป
แขน	ท่อคาร์บอนไฟเบอร์ ขนาด 30 มม.
ลำตัว	เครื่องพิมพ์ 3 มิติ โดยใช้วัสดุ PLA
เอว	ท่อคาร์บอนไฟเบอร์ ขนาด 88 มม.
สะโพก	อลูมิเนียมอัลลอยพับ
น่อง	เครื่องพิมพ์ 3 มิติ โดยใช้วัสดุ PLA
ขา	เครื่องพิมพ์ 3 มิติ โดยใช้วัสดุ PLA
ฝ่าเท้า	เครื่องพิมพ์ 3 มิติ โดยใช้วัสดุ PLA

ตารางที่ 4.1: ตารางแสดงวัสดุที่ใช้ขึ้นรูป UTHAI

#### 4.1.1 การออกแบบขา

##### 4.1.1.1 การออกแบบโครงสร้างส่วนขาครั้งที่ 1

การออกแบบโครงสร้างส่วนขาของหุ่นยนต์ชีวามานอยด์ ได้ออกแบบโดยคำนึงถึงการขึ้นรูปด้วยเครื่องพิมพ์สามมิติ (3D Printer) แต่เนื่องจากว่าเครื่องพิมพ์สามมิติที่ใช้ในการผลิตนั้นมีขนาดที่เล็กกว่าขนาดที่จะพิมพ์จริง จึงต้องทำการแยกส่วนของขาออก เป็นจำนวน 2 ส่วนในแต่ละในก้านต่อของขาท่อนบนและขาท่อนล่าง และหลังจากนั้นใช้การยึดชิ้นส่วนด้วยการตอกสลักเพื่อยึดติดชิ้นส่วนเข้าด้วยกัน เพื่อให้มีความแข็งแรงมากกว่าการต่อแบบที่ว่าไป



รูปที่ 4.2: รูปการออกแบบส่วนขาของหุ่นยนต์อุทัย

เมื่อทำการพิมพ์ชิ้นงานส่วนขาท่อนบนและท่อนล่าง ออกแบบจะได้น้ำหนักของชิ้นงานตามตาราง 4.2

ชิ้นส่วน	น้ำหนัก(กรัม)
ต้นขา	263
หน้าแข้ง	204

ตารางที่ 4.2: ตารางแสดงน้ำหนักของชิ้นส่วนขา

จากการทดสอบความสามารถในการเคลื่อนที่ พบร่วยว่าตัวขับเคลื่อนสามารถเคลื่อนที่เข้าตำแหน่งได้ถูกต้องตามมุ่ง ที่ป้อนเข้าไปให้ระบบ แต่หากทำให้ชิ้นส่วนของขาเคลื่อนที่ด้วยถีปุ่กลับสูงและด้วยความเร็วที่มาก จะทำให้ตัวขับเคลื่อนเกิดการโอเวอร์โหลด ซึ่งมีผลทำตัวขับเคลื่อนหยุดการทำงาน ซึ่งต้องทำการปิดเปิดตัวขับเคลื่อนใหม่

## ผลการทดสอบ

จากการทดสอบความแข็งแรงของชิ้นงาน พบร่วมกันมีความแข็งแรงเพียงพอที่จะทำให้ตัวขับเคลื่อนมีค่าแรงบิดเป็นค่าแรงบิดสูงสุด(Stall Torque) แล้วทำให้ชิ้นงานไม่เกิดความเสียหาย

จากการทดสอบระยะเวลาการทำงานของตัวขับเคลื่อน ด้วยการเขียนโปรแกรมให้ตัวขับเคลื่อน เคลื่อนที่ไปกลับ สลับตำแหน่งไปเรื่อยๆอย่างต่อเนื่อง เป็นเวลา 20 นาที พบร่วมกันว่า ตัวขับเคลื่อนทำงานได้เป็นปกติ

### ปัญหาที่พบในการออกแบบครั้งที่ 1

เนื่องจากว่าเป้าหมายของการสร้างหุ่นยนต์ตัวนี้ให้มีน้ำหนักที่เบา (น้อยกว่า 5 กิโลกรัม) จึงพบปัญหาว่า น้ำหนักของส่วนขาที่ได้ออกแบบมานั้นมีน้ำหนักมากเกินกว่าของหุ่นยนต์กัน(ซึ่งหุ่นยนต์ตัวเดิมก่อนจะเป็นอุทัย) ซึ่งเป็นผลทำให้เกิดปัญหาระดับของ motor ที่ต้องกระทำที่มีมากขึ้นจากเดิมและจะทำให้น้ำหนักของตัวหุ่นยนต์มากขึ้น เมื่อเปรียบเทียบผลลัพธ์น้ำหนักส่วนขาของหุ่นยนต์กันกับหุ่นยนต์อุทัยแล้วได้ผลดังตาราง 4.3

ชิ้นส่วน	หุ่นยนต์กัน(เดิม)(กรัม)	หุ่นยนต์ UTHAI
ขาท่อนบน	171	263
ขาท่อนล่าง	172	204

ตารางที่ 4.3: ตารางเปรียบเทียบน้ำหนักของชิ้นส่วนขาของหุ่นยนต์

จากข้อมูลในตารางนั้นจะเห็นได้ว่า หนักที่เพิ่มขึ้นมากจากการออกแบบใหม่แต่ละชิ้นนั้น มากถึง 124 กรัม ต่อขา 1 ข้าง และ 248 กรัมเมื่อเทียบกับขาทั้งหมดและเปรียบเทียบกับข้อมูลก่อนหน้า

หลังจากพบปัญหาดังกล่าวผู้จัดทำจึงได้ตัดสินใจทำการเปลี่ยนวัสดุที่ใช้ทำโครงสร้างในครั้งแรก ที่จะใช้การขึ้นรูปชิ้นงานด้วยการพิมพ์ขึ้นรูป 3 มิติ มาเป็นวัสดุผลกระทบระหว่างคาร์บอนไฟเบอร์และชิ้นงาน 3 มิติแทน โดยจะให้ชิ้นงาน 3 มิตินั้นทำหน้าที่เป็นตัวเชื่อมระหว่างวัสดุคุณภาพของไฟเบอร์กับมอเตอร์ และยึดกับวัสดุคุณภาพของไฟเบอร์ ด้วยการบีบ ซึ่งเหตุผลที่ต้องทำเช่นนี้ เพราะต้องการลดน้ำหนักของหุ่นยนต์ลง เพื่อไม่ให้มอเตอร์รับภาระที่หนักเกินไป

ชิ้นส่วน	วัสดุที่ใช้ขึ้นรูป(เก่า)	วัสดุที่ใช้ขึ้นรูป(ใหม่)
แขน	ท่อคาร์บอนไฟเบอร์ ขนาด 30 มม.	เดิม
ลำตัว	เครื่องพิมพ์ 3 มิติ โดยใช้วัสดุ PLA	เดิม
เอว	ท่อคาร์บอนไฟเบอร์ ขนาด 88 มม.	เดิม
สะโพก	อลูมิเนียมอลล์loyพับ	เดิม
น่อง	เครื่องพิมพ์ 3 มิติ โดยใช้วัสดุ PLA	วัสดุสมรรถนะระหว่างคาร์บอนไฟเบอร์และชิ้นงาน 3 มิติ
ขา	เครื่องพิมพ์ 3 มิติ โดยใช้วัสดุ PLA	วัสดุสมรรถนะระหว่างคาร์บอนไฟเบอร์และชิ้นงาน 3 มิติ
ฝ่าเท้า	เครื่องพิมพ์ 3 มิติ โดยใช้วัสดุ PLA	เดิม

ตารางที่ 4.4: ตารางแสดงวัสดุที่แก้ไขในการใช้ขึ้นรูป UTHAI

#### 4.1.1.2 การออกแบบโครงสร้างส่วนขาครั้งที่ 2

ครั้งนี้การออกแบบชิ้นส่วนขาของหุ่นยนต์นั้น ได้คำนึงถึงเรื่องน้ำหนักเป็นหลัก และยังคงให้ความสำคัญกับข้อต่อที่จะใช้รับน้ำหนักทั้งตัวหุ่นยนต์ และยังคงรับแรงบิดของมอเตอร์อีกด้วย ดังนั้นจึงได้ตัดสินใจที่จะเปลี่ยนจากการใช้วัสดุ 3D print ซึ่งเป็นพลาสติกทั้งหมด มาเป็นวัสดุผสม ระหว่าง Carbonfiber กับ 3D print ซึ่งขึ้นรูปจากพลาสติก PLA และทำการยึดติดกันด้วยการบีบอัดดังรูปที่ 4.3 เมื่อทำการพิมพ์ชิ้นงานส่วนขาท่อนบนและท่อนล่าง และทำการประกอบ จะได้น้ำหนักของชิ้นงานเปรียบเทียบกับของเดิมดังตาราง 4.5



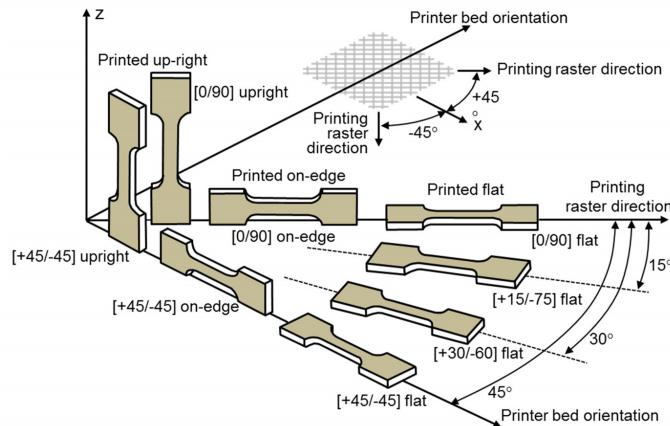
รูปที่ 4.3: รูปการออกแบบส่วนขาของหุ่นยนต์อุทัย (ใหม่)

ชิ้นส่วน	น้ำหนักเดิม(กรัม)	น้ำหนักใหม่(กรัม)
ตันขา	263	161
หน้าแข็ง	204	166

ตารางที่ 4.5: ตารางแสดงน้ำหนักเปรียบเทียบของชิ้นส่วนขา

### การขึ้นรูปชิ้นงาน

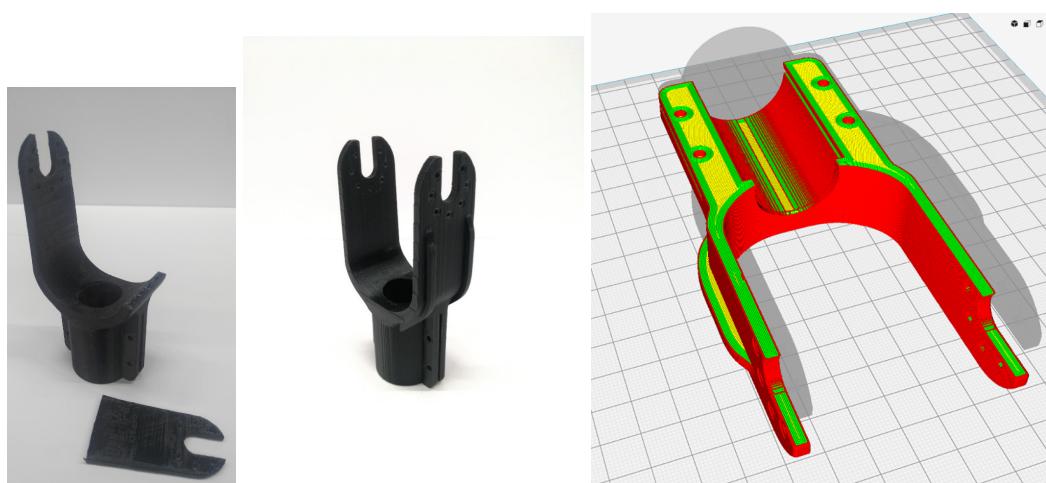
การขึ้นรูปชิ้นงานนั้น ได้ใช้การขึ้นรูปชิ้นตามความสูงแนวแกน Z ดังภาพ 4.4 เพื่อให้ชิ้นงานมีความสวยงามและสามารถสม่ำเสมอต่อ กับท่อนการบอนโดยไม่มีผิวสัมผัสมากที่สุด ในการยึดเกาะ<sup>1</sup>



รูปที่ 4.4: รูปการขึ้นรูปชิ้นงานของ 3D printer

#### 4.1.1.3 ทดสอบโครงสร้างและการขับเคลื่อน

จากการทดสอบความแข็งแรงของวัสดุโดยการนำไปประกอบกับตัวหุ่นยนต์จริง และทำการทดลองเดินพบว่าเมื่อทำการเดินจริงนั้น เกิดการฉีกขาดของชิ้นงานที่ ชั้นการพิมพ์ของชิ้นงานดังภาพ 4.9 ซึ่งเกิดจากการได้รับแรงบิดมากเกินไปจากน้ำหนักของชิ้นงานส่วนขา และแรงที่ชิ้นงานจะได้รับนั้น จะเป็นเพียงส่วนการเชื่อมกันติดของชิ้นพลาสติกเท่านั้น ซึ่ง ณ ที่นี่เส้นพลาสติกจะไม่ได้เป็นตัวรับแรงจึงทำให้ เกิดการประทักษิ่งกว่าดังนั้นจึงทำการออกแบบใหม่โดยการเพิ่มสันให้ชิ้นงานและเพิ่มความหนาบนหน้าแปลนเชื่อมกับตัว母ล้อ และทำการขึ้นรูปชิ้นงาน โดยให้ความสูงของชิ้นงานเป็นไปตามแกน X และทำการเติมเนื้อพลาสติกด้านในให้เต็ม 100% ดังรูปที่ 4.5



(ก) รูปแสดงการแตกหัก (ข) รูปแสดง ชิ้น งาน ที่ทำการ ออกแบบใหม่ (ค) รูปแสดงชิ้นของการพิมพ์ตามแนวแกน x โดย การเติมเนื้อพลาสติก 100%

รูปที่ 4.5: รูปแสดงการแตกหักและชิ้นการพิมพ์

<sup>1</sup>Experimental Characterization of the Mechanical Properties of 3D-Printed ABS and Polycarbonate Parts

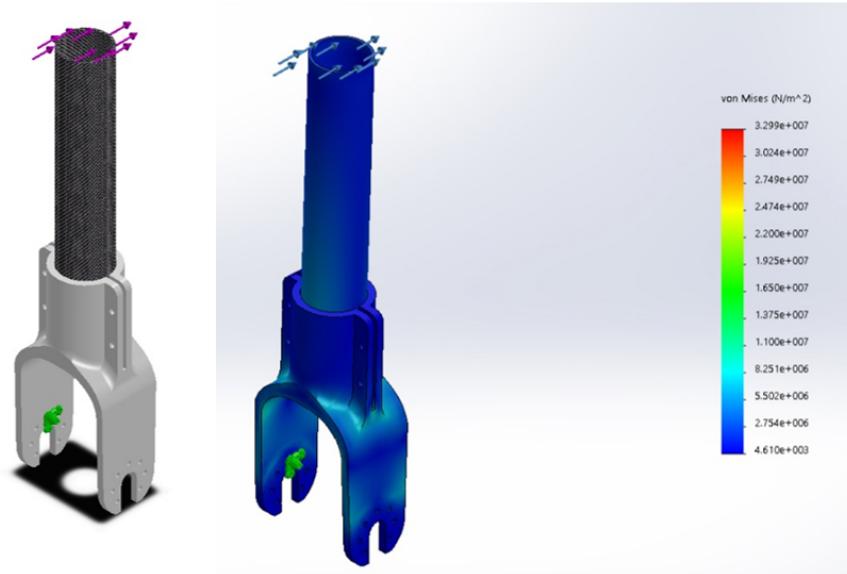
### การทดลองความแข็งแรงของชิ้นงานโดยวิธีการวิเคราะห์เชิงตัวเลข(Finite element)

ก่อนที่จะนำชิ้นงานที่ทำการออกแบบใหม่ที่เติมเนื้อพลาสติก 100% ไปใช้งานจริงนั้นจะต้องผ่านการวิเคราะห์แรงกระทำ โดยผ่านโปรแกรมจำลองเพื่อหาจุดที่เปราะบางของชิ้นงาน และนำข้อมูลนั้นไปวิเคราะห์เพื่อปรับปรุงชิ้นงานต่อไป ซึ่งได้ดังค่า คุณสมบัติของชิ้นงาน 3d print ไว้ดังนี้

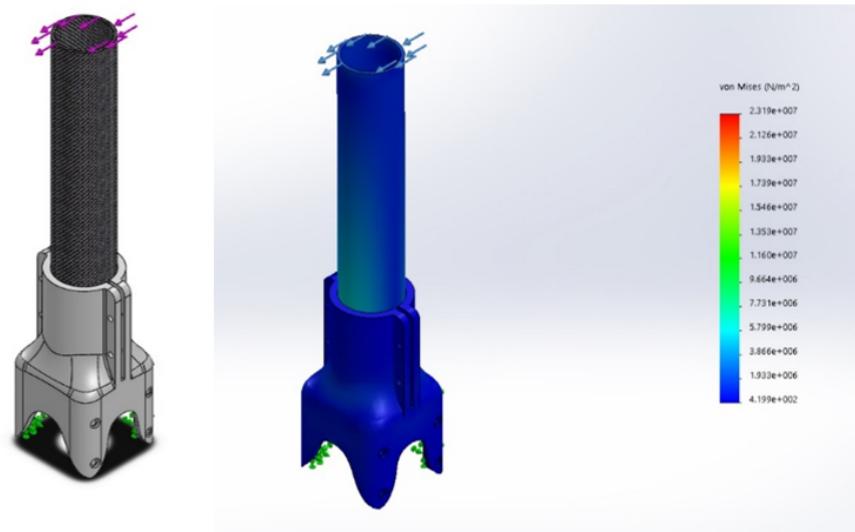
Print Orientation Side	flat
Ultimate Stress( $N/mm^2$ ) <sup>2</sup>	45.66
Young's Modulus( $N/mm^2$ ) <sup>3</sup>	1141.55
Yield strength( $N/mm^2$ ) <sup>4</sup>	23
Density( $kg/m^3$ ) <sup>5</sup>	1250
Poisson ratio <sup>6</sup>	0.33
Force(torque)( $N.m$ )	10.4

ตารางที่ 4.6: ตารางแสดงค่าคุณสมบัติของวัสดุ 3D print(PLA)

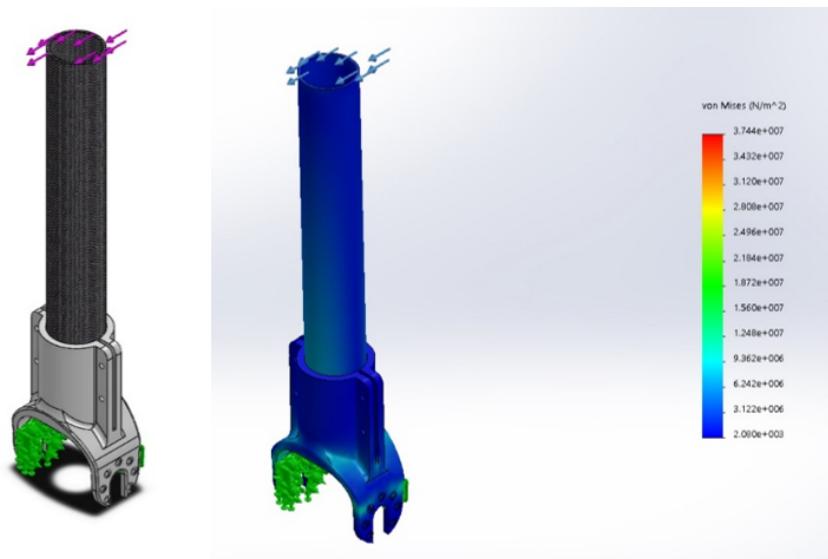
เมื่อทดลองนำค่าดังกล่าวไปใช้ในโปรแกรม solidwork และใช้ฟังก์ชัน mass property(คุณสมบัติของวัสดุ) เพื่อหาค่าน้ำหนักที่ผ่านการคำนวณโดยโปรแกรม และนำมาเทียบกับชิ้นงานจริงเพื่อถู่ว่ามีความคลาดเคลื่อนด้านน้ำหนักมากน้อยขนาดไหน พบร่ววค่าข้อมูลความคลาดเคลื่อนนั้นจะไม่เกินกว่าระหว่าง  $\pm 1\%$  กับค่าที่แสดงบนโปรแกรม หลังจากนั้นทำการวิเคราะห์ผ่านโปรแกรม solidword ด้วยการวิเคราะห์ FEA ได้ผลลัพธ์ดังนี้



รูปที่ 4.6: รูปการวิเคราะห์แรงของข้อต่อ 1



รูปที่ 4.7: รูปการวิเคราะห์แรงของข้อต่อ 2



รูปที่ 4.8: รูปการวิเคราะห์แรงของข้อต่อ 3

การวิเคราะห์นั้นจะทำการยึดจุดที่เป็นหน้าแปลนของมอเตอร์ไวเพื่อให้เบรียบเสมือนกับว่าขณะนี้ชิ้นงานได้เข้มติดกับตัวมอเตอร์ หลังจากนั้นกำหนดแรงกระทำเพื่อให้เกิดโมเมนต์กับชิ้นงานโดยค่าแรงที่กระทำนั้น ได้มาจากการคำนวณแรงของมอเตอร์ที่จะรับไหวเทียบกับระยะของแรง ที่กระทำกับชิ้นงาน ซึ่งได้ทดลองกับชิ้นงานตัวข้อต่อ 1 2 และ 3 ด้วยแรง 41.6 นิวตัน( $N$ ) เมื่อนำค่า ความตึงเครียดสูงสุด(*Max stress*)ของชิ้นงานมาวิเคราะห์เพื่อหา จุดประยุกต์ของวัสดุได้ค่าดังตาราง ??

ชื่อชิ้นงาน	ความตึงเครียดสูงสุดของชิ้นงาน( <i>Max stress</i> )( $N/mm^2$ )
ข้อต่อ 1	32.99
ข้อต่อ 2	23.19
ข้อต่อ 3	7.987

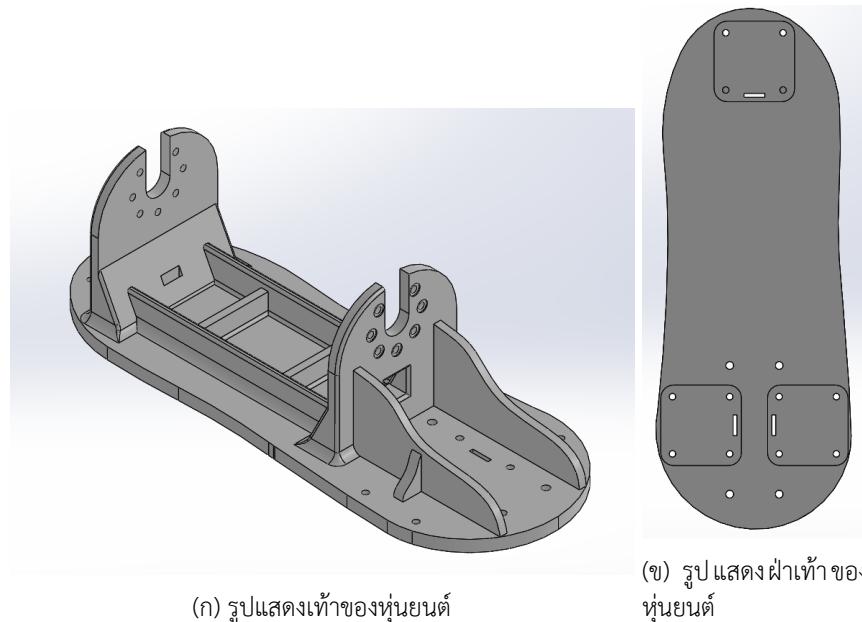
ตารางที่ 4.7: ตารางแสดงความตึงเครียดของชิ้นงาน(Stress)

จากผลการทดลองสรุปได้ว่า ค่าความตึงเครียดต่างๆที่ได้มาจากการทดสอบนั้นมีน้ำไปเทียบกับค่าความตึงเครียดสูงสุดที่วัสดุจะรับไหว ที่  $45.66 N/mm^2$  เห็นได้ว่ายังคงไม่มีวัสดุตัวไหนที่จะเกิดการแตกหักเมื่อเกิดแรงกระทำกับชิ้นงานตั้งนั้นชิ้นงานที่ทำการออกแบบนี้ พoSรุปได้ว่าจะไม่เกิดการแตกหักระหว่างการทำงาน ยกเว้นมีแรงกระทำจากภายนอกที่มากเกินไปจนมาผลทำให้เกิดความตึงเครียดของชิ้นงานสูง เกินกว่าค่าตั้งกล่าว

#### 4.1.2 การออกแบบเท้า

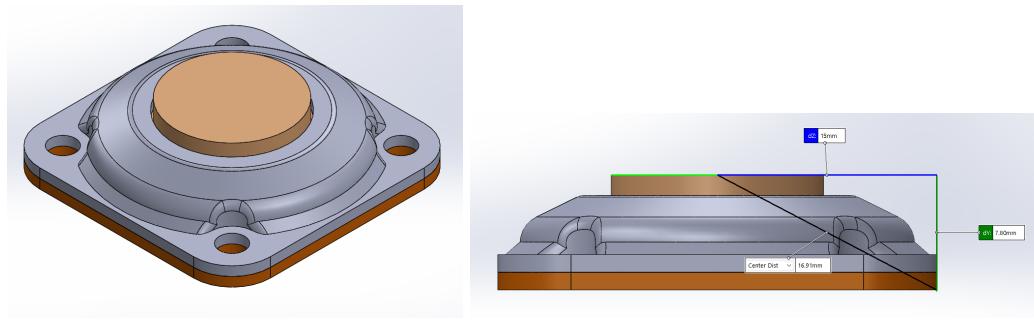
##### 4.1.2.1 การออกแบบโครงสร้างเท้าครั้งที่ 1

โครงสร้างเท้านี้ได้ออกแบบให้มีลักษณะคล้ายคลึงกับรองเท้าของมนุษย์จริงและมีขนาดที่เหมาะสมกับตัวหุ่นยนต์ โดยคำนึงถึงความแข็งแรง และการใช้งานเป็นหลัก และยังต้องขึ้นรูปด้วยเครื่องพิมพ์ 3 มิติได้อีกด้วย



รูปที่ 4.9: รูปแสดงเท้าของหุ่นยนต์อุทัย

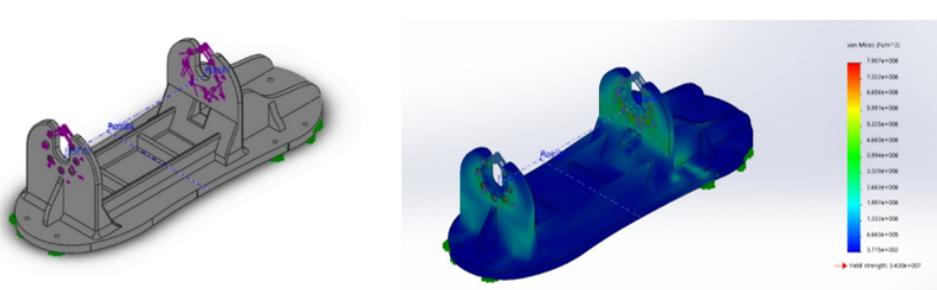
ในส่วนของโครงครอบ FSR นั้นได้ออกแบบให้มีการกดโดยตรงกับหน้าสัมผัสซึ่งส่วนที่สัมผัสกับหน้าสัมผัสนั้นจะเป็นเฉพาะส่วนของ 3d print ที่ออกแบบมา เฉพาะการกดโดยเฉพาะ ซึ่งจะทำให้สนองหน้าสัมผัสได้ดีกว่า การกดจากภายนอกโดยตรงซึ่งตัวเซนเซอร์นี้จะมีความสูงของมาจากการฝ่าเท้าเป็นระยะ 5.4 มิลลิเมตร โดยจะมีจำนวน 3 ตัวต่อเท้า 1 ข้าง



รูปที่ 4.10: รูปแสดงโดยรวมของโครงครอบ FSR

#### การทดลองความแข็งแรงของชิ้นงานโดยวิธีการวิเคราะห์เชิงตัวเลข(Finite element)

จากค่าคุณสมบัติของชิ้นงานดังตาราง 4.6 นำมาวิเคราะห์ด้วยเทคนิคเชิงตัวเลขได้ผลดังนี้



รูปที่ 4.11: รูปการวิเคราะห์แรงของเท้าหุ่นยนต์

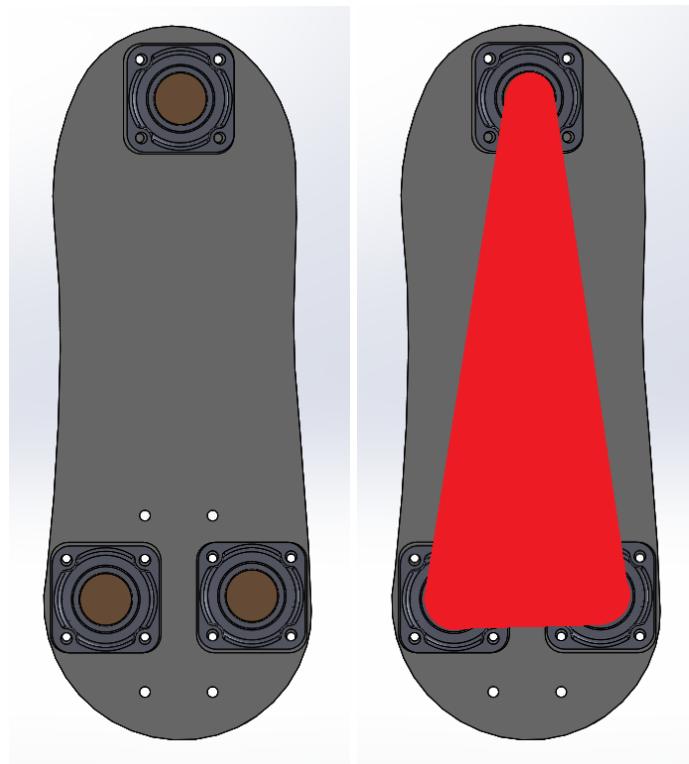
การวิเคราะห์นั้นมีวิธีการทำเหมือนกับการวิเคราะห์ชิ้นส่วนฯคือทำการยึดจุดที่เป็นหน้าแปลนของมอเตอร์ไว้เพื่อให้เปรียบเสมือนกับว่าขณะนี้ชิ้นงานได้เข้ามิติดกับตัวมอเตอร์ หลังจากนั้นกำหนดแรงกระทำเพื่อให้เกิดแรงบิดกับชิ้นงานโดยค่าแรงที่กระทำนั้นได้มาจากความถ่วงแรงของมอเตอร์ที่ทำกับชิ้นงานข้อเท้าที่แรงบิด 10.4 นิวตัน/เมตร ( $N.m$ ) ได้ผลดังตาราง 4.8

ชื่อชิ้นงาน	ความตึงเครียดสูงสุดของชิ้นงาน(Max stress)( $N/mm^2$ )
ฝ่าเท้า	37.44

ตารางที่ 4.8: ตารางแสดงความตึงเครียดของชิ้นงาน(Stress)ของฝ่าเท้า

## ปัญหาที่พบ

เมื่อทำการประกอบชิ้นส่วนของ FSR(Force Sensitive Resistor) กับฝ่าเท้าแล้วปัญหาที่พบคือ พื้นที่สัมผัสพื้นของฝ่าเท้า น้อยลงซึ่งเป็นผลทำให้ support polygon น้อยลงด้วยซึ่งเป็นเหตุทำให้การเดินของหุ่นยนต์นั้นยากลำบาก



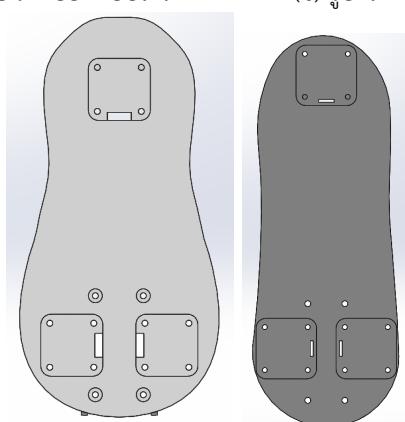
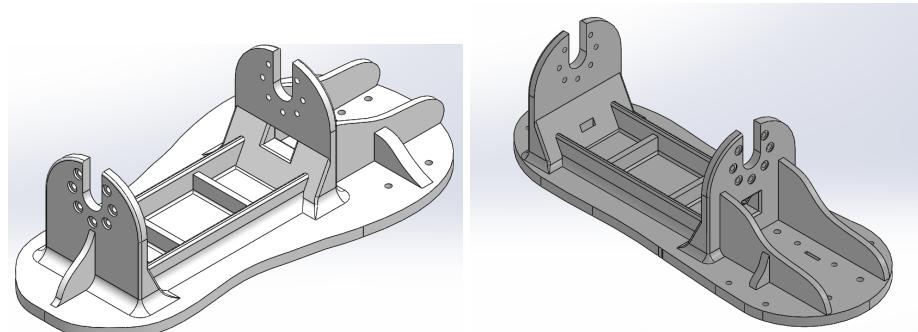
(ก) รูป แสดง เท้า ของ หุ่น ยนต์ เมื่อ (ข) รูปแสดง support polygon ของ  
ประกอบ FSR และ เท้าเมื่อประกอบ FSR

รูปที่ 4.12: รูปแสดงฝ่าเท้าและ support polygon

ดังนั้นจึงทำการแก้ไขโดย ออกแบบฝ่าเท้าให้มีขนาดใหญ่เพิ่มขึ้น และออกแบบเชนเชอร์ตรวจจับการเดิน ให้มีความบางลงอีกเพื่อให้หุ่นยนต์นั้น มี support polygon เพิ่มขึ้น

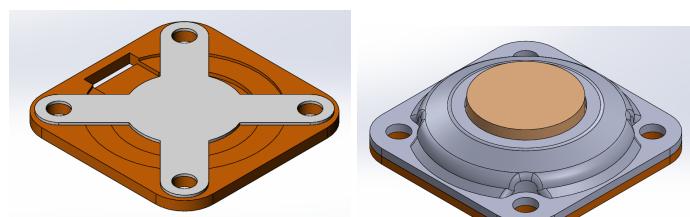
#### 4.1.2.2 การออกแบบโครงสร้างเท้าครั้งที่ 2

ในการออกแบบครั้งนี้ได้เพิ่มขนาดของฝ่าเท้าให้ใหญ่ขึ้นเพื่อเพิ่ม support polygon ซึ่งจะเป็นผลทำให้หุ่นยนต์นั้นมีการเดินที่ร้ายขึ้น ในการปรับขนาดครั้งนี้ได้เพิ่มขนาดที่ปลายเท้าให้ใหญ่ขึ้น และสันเท้ารองลงมา



รูปที่ 4.13: รูปแสดงการเปรียบเทียบระหว่างเท้าเดิมและเท้าที่ออกแบบใหม่

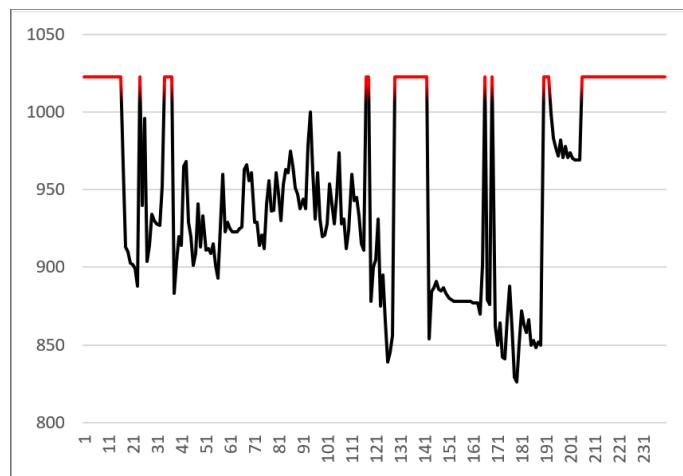
ในส่วนของเซนเซอร์ตรวจจับพื้นนี้ได้ทำการออกแบบใหม่ทั้งหมดให้มีความบางลงกว่าเดิม และให้มีส่วนที่ยื่นออกมา จากฝ่าเท้าน้อยที่สุดซึ่งความสูงที่ยื่นออกมาจะมีความสูงเพียง 0.4 มิลลิเมตร ต่างจากเดิมที่มีความสูงถึง 5.4 มิลลิเมตร



รูปที่ 4.14: รูปแสดงการเปรียบเทียบระหว่างฝ่าเท้าเดิมและฝ่าเท้าที่ออกแบบใหม่

#### 4.1.2.3 ทดสอบการใช้งานของเซนเซอร์ตรวจจับการสัมผัสพื้น

ในโครงงานนี้เซนเซอร์ตรวจจับพื้นนั้นได้ใช้งานเพื่อทำการเรียบของเท้าเท่านั้นว่ามีการแตะพื้นหรือไม่ ซึ่งจะกำหนดค่าไว้ในช่วง 0-1023 โดยจะกำหนดให้ช่วงที่มากกว่า 1000 หมายถึงเท้ามีการยกเกิดขึ้นและต่ำกว่า 1000 หมายถึงเท้ามีการเรียบ โดยจะแบ่งค่าเซนเซอร์ทั้ง 3 ค่าบนเท้า เป็นค่าเฉลี่ยน้ำหนัก ตามสมการ  $(sensor1 + sensor2 + 2(sensor3))/4$  โดยให้ค่าตามแกน X เป็นค่า sampling(ครั้ง) ของข้อมูลและ ค่าตามแกน Y เป็นค่าของ 0-1023 เมื่อทำการทดลองให้ทำการเรียบเท้าและยกเท้าจะได้ผลดังภาพ 4.15



รูปที่ 4.15: รูปภาพแสดงค่าที่วัดได้ของ FSR

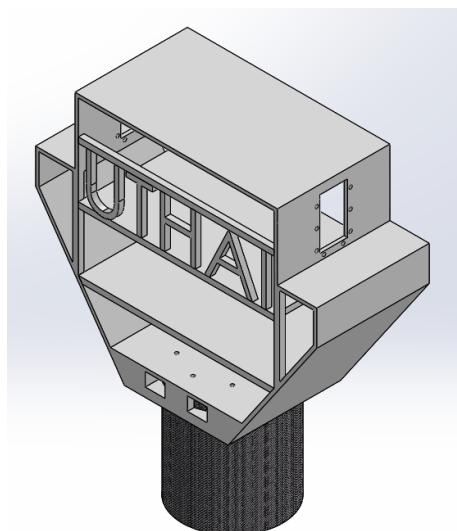
ผลที่ได้นี้เกิดจากการทดลองยกเท้าของหุ่นยนต์ขึ้นขณะหุ่นยนต์ยืนอยู่และยกเท้าขึ้น ค่าที่อ่านได้จะมีค่า 1023 หรือสูงสุดตามภาพ และเมื่อเท้าสัมผัสพื้นนั้น ค่าที่อ่านได้จะมีค่าที่ต่ำขึ้นอยู่กับแรงกดที่กระทำต่อเท้า ซึ่งถ้าค่าที่กระทำมากจะทำให้ค่าที่อ่านได้ต่ำมาก จะเห็นได้จากรูปภาพ 4.15 จะมีช่วงเวลาหนึ่งที่ค่าที่อ่านได้ต่ำสุดซึ่งขณะนั้น ได้ทดลองกดด้วยแรงจำนวนหนึ่งที่มากกว่าน้ำหนักตัวหุ่นยนต์เป็นเวลา 1-2 วินาที

### 4.1.3 การออกแบบโครงสร้างลำตัว

ลำตัวของหุ่นยนต์นั้นจะใช้สำหรับติดตั้งหน่วยประมวลผลควบคุมระดับต่ำและระดับสูง IMU รวมไปถึงบอร์ดแปลงไฟ 12v จากแบตเตอรี่ ให้เหลือ 5V เพื่อจ่ายไฟให้กับระบบ ละยังคงต้องจัดเก็บแบตเตอรี่สำหรับทำงานไร้สายได้อีกด้วย

#### 4.1.3.1 การออกแบบลำตัวครั้งที่ 1

การออกแบบครั้งนี้ได้ออกแบบให้ลำตัวนั้นขึ้นรูปด้วยเครื่องพิมพ์ 3 มิติ เพื่อให้ง่ายสำหรับผู้ต่อยอดที่มีเครื่องพิมพ์เป็นของตนเอง สามารถพิมพ์ และนำมาประกอบได้ โดยส่วนของเอวันั้นจะใช้เป็นท่อคาร์บอนไฟเบอร์ขนาดเส้นผ่าศูนย์กลางขนาด 91 มิลลิเมตร เพื่อให้เหมาะสมกับชิ้นงาน และเชื่อมยึดติดกันด้วยสกรูกับลำตัวและส่วนเอว

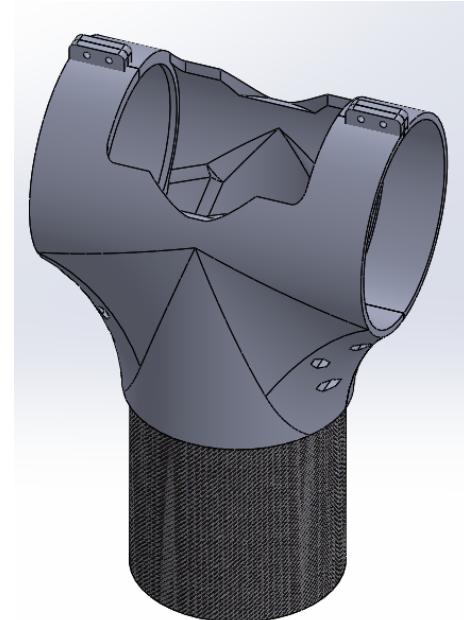


รูปที่ 4.16: รูปภาพแสดงตัวของหุ่นยนต์ที่ออกแบบเพื่อขึ้นรูปด้วยเครื่องพิมพ์ 3 มิติ

แต่เมื่อทำการทดลองหาค่ามวลในโปรแกรม solidwork แล้ว ได้ผลน้ำหนักคือ 711 กรัม ซึ่งเป็นน้ำหนักที่มากเกินไปอาจส่งผลทำให้มอเตอร์รับน้ำหนักของตัวมากเกินไป ดังนั้นจึงเป็นผลทำให้ต้องเปลี่ยนการออกแบบให้เบาลงกว่าเดิม คือลดขนาดของตัวที่ขึ้นรูปด้วยเครื่องพิมพ์ 3 มิติ ลงและใช้วัสดุสมรรถนะว่าการบอนไฟเบอร์มากขึ้น ซึ่งจะยังคงได้ความแข็งแรงและความเบาอีกด้วย

#### 4.1.3.2 การออกแบบโครงสร้างตัวครั้งที่ 2

จากปัญหาเรื่องน้ำหนักของชิ้นส่วนตัวของการออกแบบครั้งที่ 1 ได้แก้ไขโดยลดขนาดของส่วนพิมพ์ 3 มิติ ลงซึ่งจะแยกชิ้นส่วนออกเป็น 2 ส่วน คือส่วนหน้าและส่วนหลัง และใช้การยึดกับท่อคาร์บอนไฟเบอร์ซึ่งเป็นส่วนเอวท้ายการบีบซึ่งทำโดยการร้อยสกรูผ่านช่องที่ทำไว้สำหรับนตัวชิ้นงานพิมพ์ 3 มิติ แล้วขันให้ชิ้นงานมาประกับเข้าหากัน ซึ่งน้ำหนักที่วิเคราะห์ได้โดยโปรแกรม solidwork นั้นได้ค่าเท่ากับ 342 กรัม ซึ่งน้อยกว่าการออกแบบ



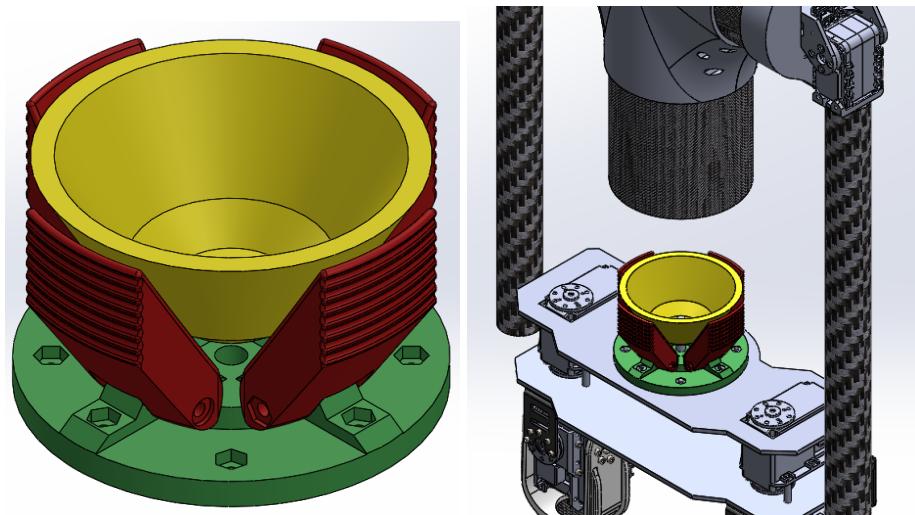
รูปที่ 4.17: รูปภาพแสดงตัวของหุ่นยนต์ที่ออกแบบเพื่อขึ้นรูปด้วยเครื่องพิมพ์ 3 มิติ(ใหม่)

ครั้งแรกถึง 2 เท่า ซึ่งมีน้ำหนักมากถึง 711 กรัม

#### การยึดลำตัวกับஸ์โพก

การยึดลำตัวกับஸ์โพกนี้จะใช้รูปแบบการยึดโดยการถ่างวัสดุที่ทำขึ้นมาเพื่อยึดลำตัวกับஸ์โพกออกแบบผ่านสกรู 1 ตัวที่ออกแบบไว้ โดยสกรูตัวนี้จะทำหน้าที่ดึงให้ถ่วงของตัวถ่าง เคลื่อนที่ลงมาและในขณะนั้นเองตัวถ่างด้านนอกอีก 4 ตัว จะค่อยๆขยับออกแบบให้มีแรงบีบกับขอบท่อคาร์บอนไฟเบอร์และยึดกันอย่างแน่นหนึ่ง

ข้อแนะนำในการยึดให้แน่นมากขึ้น ควรจะใช้วัสดุที่มีความหนืด เช่น ยางในรถจักรยาน หรือแผ่นกันเลื่อนยึดกับหน้าสัมผัสของตัวถ่างก่อนแล้วจึงนำไปยึดกับวัสดุจริง)

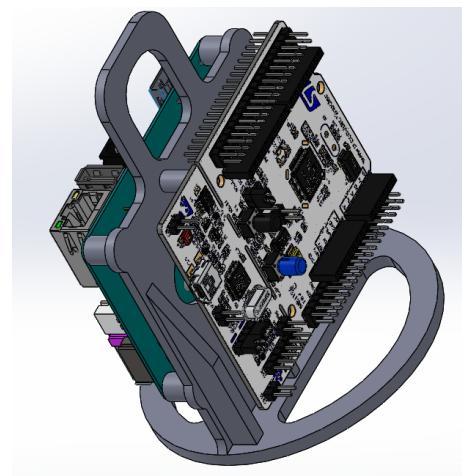


(ก) รูปภาพ แสดง อุปกรณ์ ยึด ระหว่าง ลำตัว กับ (ข) รูปภาพ แสดง การ ติด ตั้ง บน สะโพก พร้อม ทำการยึดกับลำตัวหุ่นยนต์

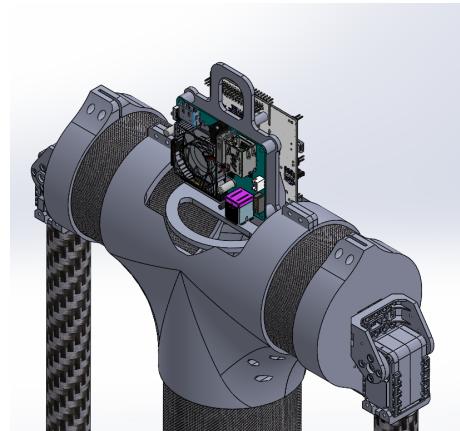
รูปที่ 4.18: รูปแสดงการใช้งานส่วนยึดสะโพกกับลำตัว

#### การติดตั้งบอร์ดควบคุมและแบตเตอรี่

เนื่องจากว่าบอร์ดควบคุมทั้ง 2 (Nucleo f411re, Odroid XU4) นั้นมีขนาดที่กระหัดรัด รวมถึงบอร์ด IMU และบอร์ดแปลงไฟ ที่มีขนาดเล็กเข่นกัน ฉะนั้นจึงได้ออกแบบ ฐานสำหรับยึดบอร์ดทั้งหมดไว้ในที่เดียว และเมื่อติดตั้งในฐานเรียบร้อยแล้วก็สามารถนำฐานนั้น สวมลงไปในตัวของหุ่นยนต์ได้พอดี



รูปที่ 4.19: รูปภาพแสดงฐานที่ติดตั้งบอร์ดควบคุม

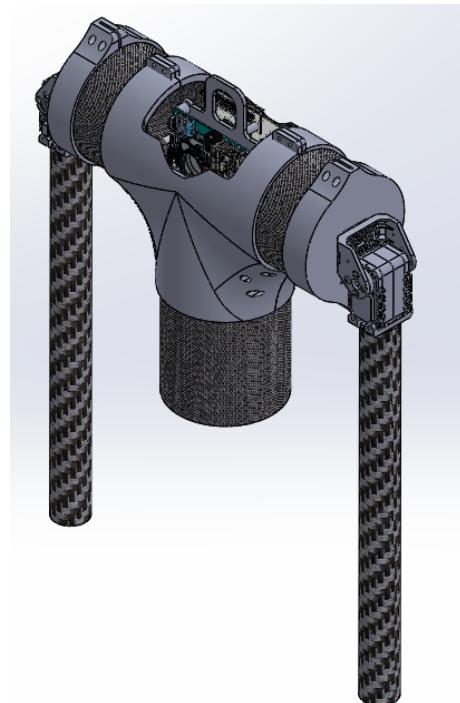


รูปที่ 4.20: รูปภาพแสดงตัวของหุ่นยนต์ที่ติดตั้งบอร์ดควบคุมในตัวของหุ่นยนต์

ซึ่งเมื่อทำการติดตั้งในตัวหุ่นยนต์แล้ว การยึดติดกับบอร์ดนั้นใช้หลักการยึดเดียวกับการยึดท่อคาร์บอนกับตัวคิว ใช้แรงของการบีบอัดจากสกรูบนลำตัวทั้งหมด ยึดให้อยู่กับที่ ส่วนของแบตเตอรี่นั้นจะใช้เป็นแบตเตอรี่ขนาด  $6000mAh$   $12.6V$  จะถูกติดตั้งในส่วนลำตัวของหุ่นยนต์บริเวณท่อคาร์บอนไฟเบอร์

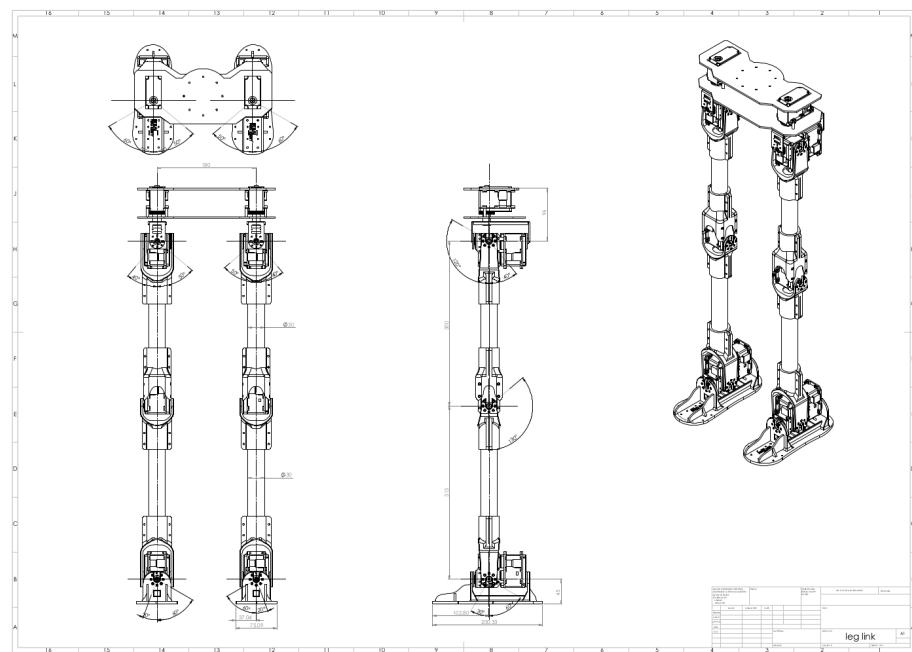
#### 4.1.3.3 การออกแบบแขน

แขนนั้นได้ออกแบบให้เรียบง่ายและน้ำหนักเบา ซึ่งในโครงงานนี้แขนจะเป็นหนึ่งในตัวแปรที่ช่วยในการเดินให้คล่องแคล่วมากขึ้น โดยวัสดุหลักที่ใช้มาทำแขนนั้นจะมาจากวัสดุคาร์บอนไฟเบอร์เป็นหลักและชิ้นส่วนพิมพ์ 3 มิติจะใช้สำหรับเชื่อมวัสดุทั้งหมดเข้าด้วยกัน

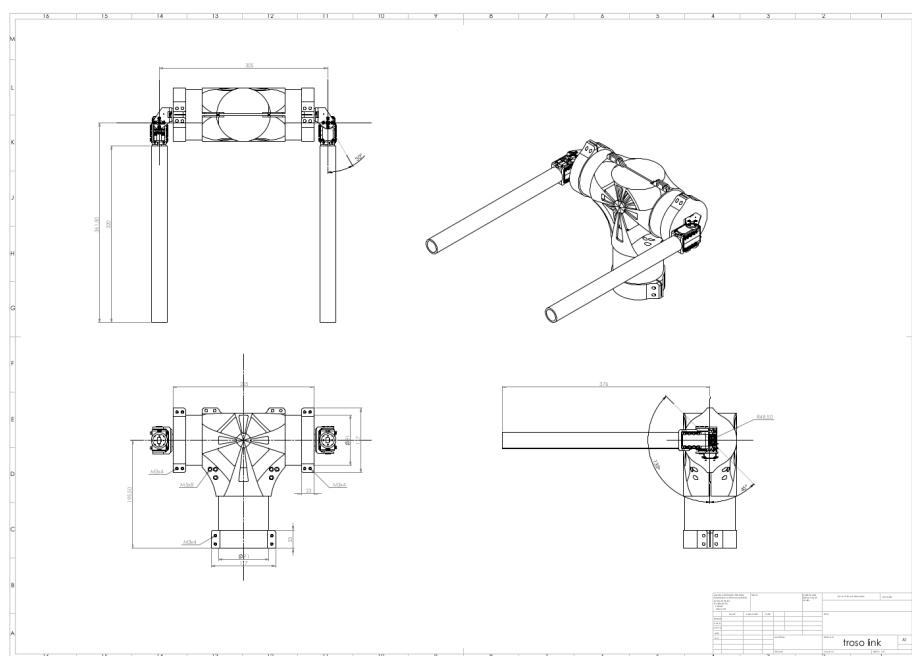


รูปที่ 4.21: รูปภาพแสดงตัวของหุ่นยนต์ที่ติดตั้งแขนทั้ง 2 ข้าง

#### 4.1.4 Engineer drawing



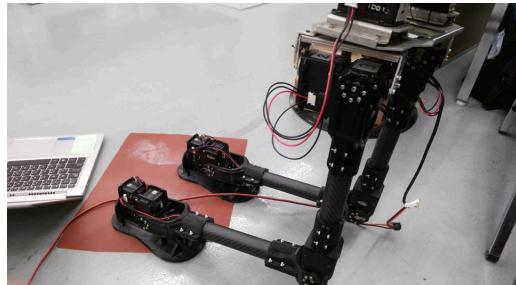
รูปที่ 4.22: ภาพ drawing ของขาหุ้นยนต์อิวามานอยด์อุทัย



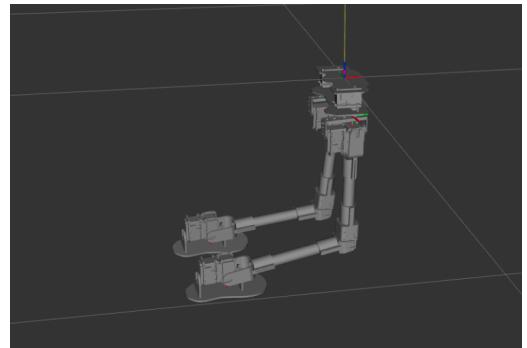
รูปที่ 4.23: ภาพ drawing ของตัวหุ้นยนต์อิวามานอยด์อุทัย

## 4.2 การออกแบบโปรแกรมด้วย ROS

#### 4.2.1 รับค่าจากมอเตอร์แล้วมาแสดงผลใน RViz

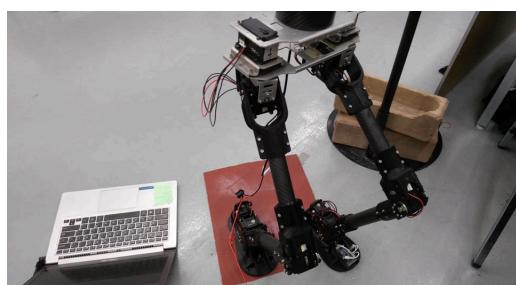


(ก) หุ่นยนต์ตัวจริง

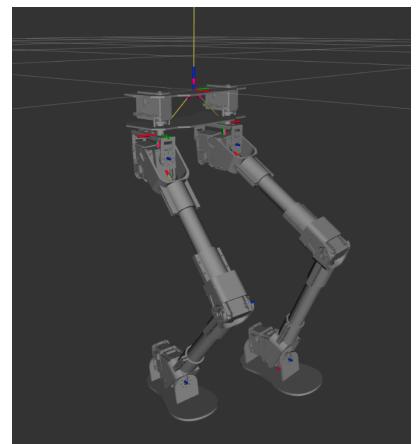


(ข) หุ่นยนต์ใน RViz

รูปที่ 4.24: การแสดงผลท่าทาง 1

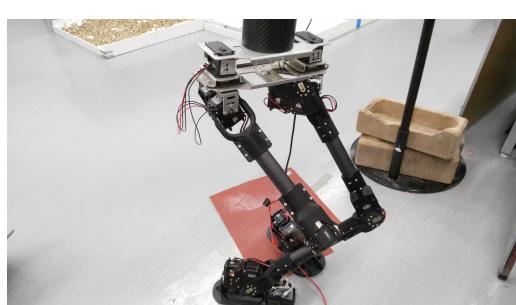


(ก) หุ่นยนต์ตัวจริง

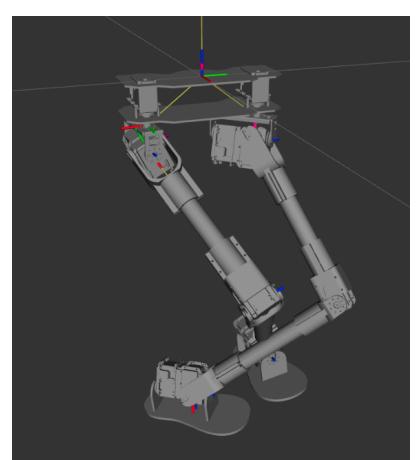


(ข) หุ่นยนต์ใน RViz

รูปที่ 4.25: การแสดงผลท่าทาง 2



(ก) หุ่นยนต์ตัวจริง



(ข) หุ่นยนต์ใน RViz

รูปที่ 4.26: การแสดงผลท่าทาง 3

#### 4.2.2 Simulation Gazebo

ต้องติดตั้ง package ต่อไปนี้

1 joint\_state\_controller

2 effort\_controller

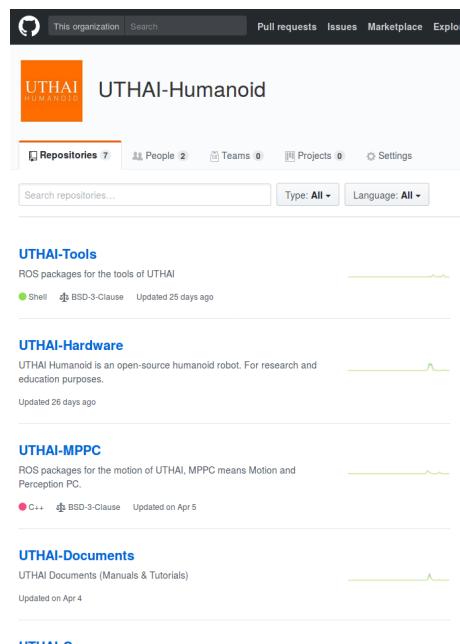
3 controller\_manager\*

4 gazebo\_ros\_control\*

## 4.3 การออกแบบระบบพื้นฐาน

### 4.3.1 GitHub ของหุ่นยนต์อุทัย

ไฟล์ข้อมูลทุกอย่างเกี่ยวกับหุ่นยนต์อิวามานอยด์อุทัยได้ถูกอัพโหลดขึ้นบนอินเทอร์เน็ต โดยอัปโหลดไปไว้ที่ GitHub [https://github.com/UTHAI-Humanoid] และมีการเขียน Wiki การใช้งานเบื้องต้นเอาไว้ สำหรับนักศึกษาหรือนักวิจัยที่ต้องการพัฒนาต่อ สามารถที่จะ Pull request เข้ามาได้ ซึ่งจะช่วยทำให้หุ่นยนต์อิวามาโนiyด์อุทัยสามารถพัฒนาต่อยอดต่อไปได้

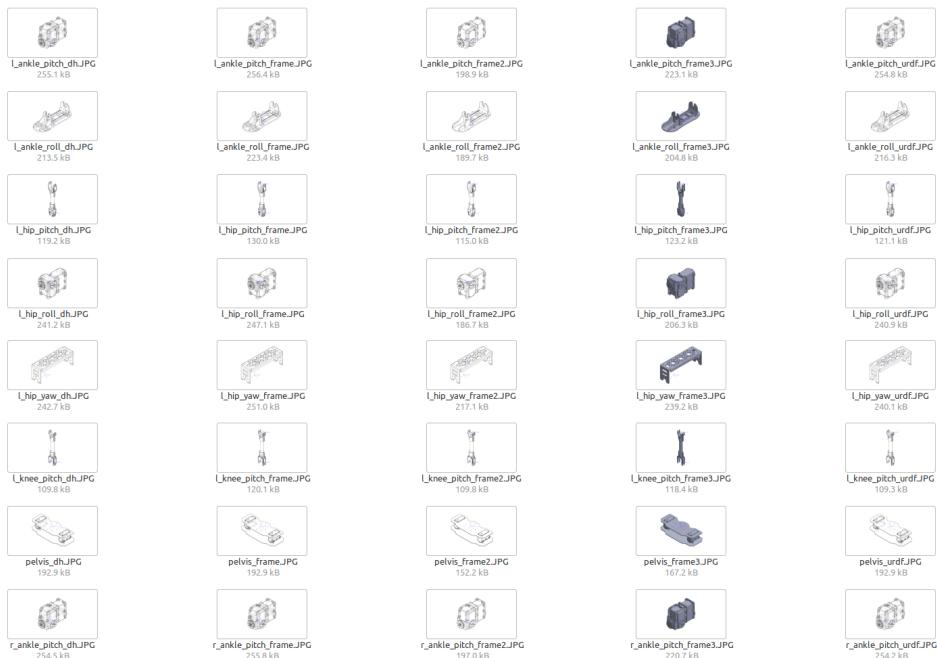


รูปที่ 4.27: GitHub ที่เก็บข้อมูลทั้งหมดของหุ่นยนต์อุทัย

รูปที่ 4.28: ตัวอย่าง Wiki การใช้งานเบื้องต้นของหุ่นยนต์อุทัย

### 4.3.2 ตัวอย่างเฟรมของหุ่นยนต์อุทัย

เฟรมของหุ่นยนต์อุทัยได้ถูกอพโหลดให้อยู่บนอินเทอร์เน็ต โดยอยู่ที่ <https://github.com/UTHAI-Humanoid/UTHAI-Hardware/tree/master/Mechanics/Frame>

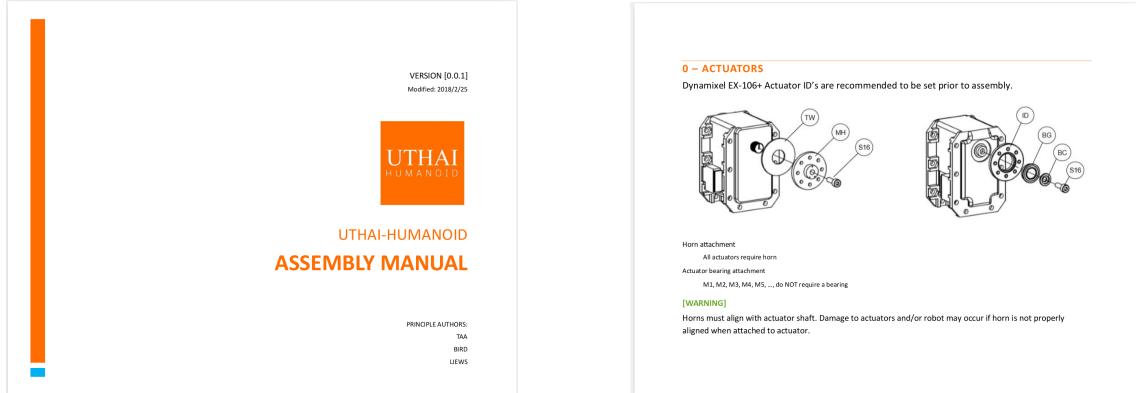


รูปที่ 4.29: ภาพเฟรมของแต่ละพาร์ทของหุ่นยนต์อุทัย

Commit	Message	Date
SwellLz Merge branch 'master' of https://github.com/UTHAI-Humanoid/UTHAI-Hard...	Latest commit 18d64b5 on Mar 10	2 months ago
..		
Merge branch 'master' of https://github.com/UTHAI-Humanoid/UTHAI-Hard...	2 months ago	
Merge branch 'master' of https://github.com/UTHAI-Humanoid/UTHAI-Hard...	2 months ago	
Merge branch 'master' of https://github.com/UTHAI-Humanoid/UTHAI-Hard...	2 months ago	
Merge branch 'master' of https://github.com/UTHAI-Humanoid/UTHAI-Hard...	2 months ago	
Merge branch 'master' of https://github.com/UTHAI-Humanoid/UTHAI-Hard...	2 months ago	
Merge branch 'master' of https://github.com/UTHAI-Humanoid/UTHAI-Hard...	2 months ago	
Merge branch 'master' of https://github.com/UTHAI-Humanoid/UTHAI-Hard...	2 months ago	
Merge branch 'master' of https://github.com/UTHAI-Humanoid/UTHAI-Hard...	2 months ago	
Merge branch 'master' of https://github.com/UTHAI-Humanoid/UTHAI-Hard...	2 months ago	
Merge branch 'master' of https://github.com/UTHAI-Humanoid/UTHAI-Hard...	2 months ago	
Merge branch 'master' of https://github.com/UTHAI-Humanoid/UTHAI-Hard...	2 months ago	
Merge branch 'master' of https://github.com/UTHAI-Humanoid/UTHAI-Hard...	2 months ago	
Merge branch 'master' of https://github.com/UTHAI-Humanoid/UTHAI-Hard...	2 months ago	
Merge branch 'master' of https://github.com/UTHAI-Humanoid/UTHAI-Hard...	2 months ago	
Merge branch 'master' of https://github.com/UTHAI-Humanoid/UTHAI-Hard...	2 months ago	
Merge branch 'master' of https://github.com/UTHAI-Humanoid/UTHAI-Hard...	2 months ago	
Merge branch 'master' of https://github.com/UTHAI-Humanoid/UTHAI-Hard...	2 months ago	
Merge branch 'master' of https://github.com/UTHAI-Humanoid/UTHAI-Hard...	2 months ago	
Merge branch 'master' of https://github.com/UTHAI-Humanoid/UTHAI-Hard...	2 months ago	
Merge branch 'master' of https://github.com/UTHAI-Humanoid/UTHAI-Hard...	2 months ago	

รูปที่ 4.30: ภาพ GitHub ของเฟรมแต่ละพาร์ทของหุ่นยนต์อุทัย

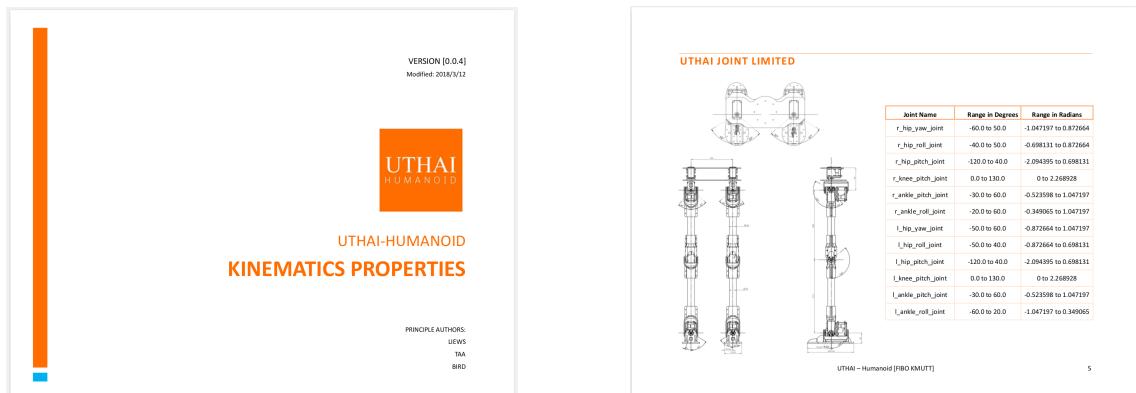
### 4.3.3 ตัวอย่างเอกสารข้อมูลของหุ่นยนต์อุตสาหกรรม



(ก) หน้าปกคู่มือการประกอบหุ่นยนต์อุตสาหกรรม

(ข) ตัวอย่างคู่มือการประกอบหุ่นยนต์อุตสาหกรรม

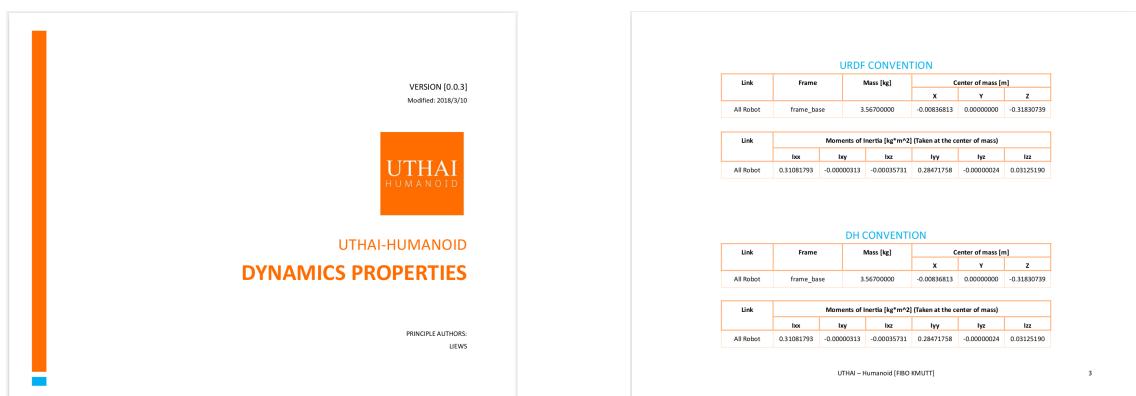
รูปที่ 4.31: UTHAI Assembly Manual



(ก) หน้าปกรายละเอียดของหุ่นยนต์อุตสาหกรรม

(ข) ตัวอย่างรายละเอียดของหุ่นยนต์อุตสาหกรรม

รูปที่ 4.32: UTHAI Kinematics Properties



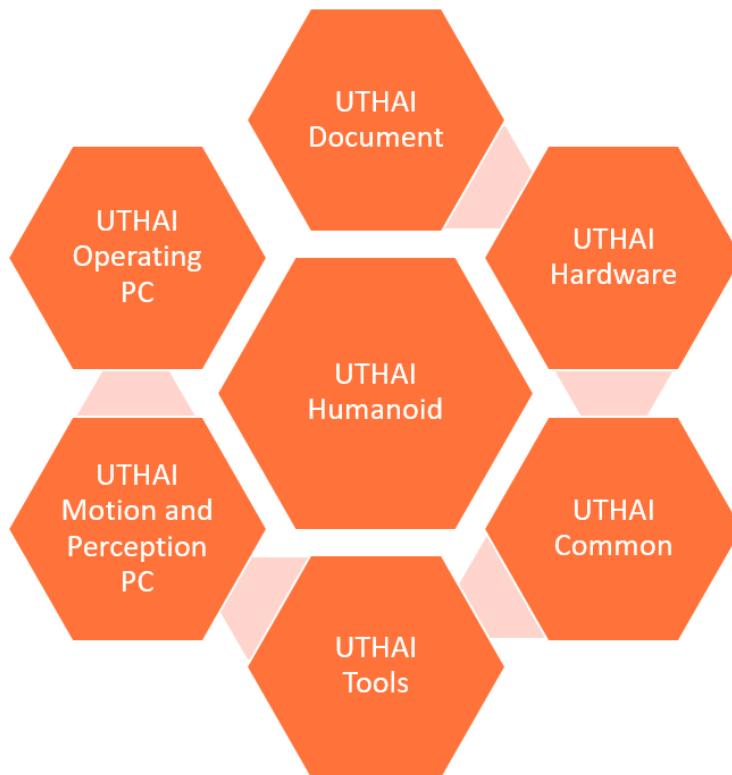
(ก) หน้าปกรายละเอียดของหุ่นยนต์อุตสาหกรรม

(ข) ตัวอย่างรายละเอียดของหุ่นยนต์อุตสาหกรรม

รูปที่ 4.33: UTHAI Dynamics Properties

#### 4.3.4 ภาพรวมระบบพื้นฐานของหุ่นยนต์อุตสาหกรรม

ระบบพื้นฐานที่ผู้วิจัยได้ทำการออกแบบเพื่อที่จะทำให้คนที่เข้ามาพัฒนาต่อยอดได้อย่างสะดวก และเป็นระบบระเบียบ มีรูปแบบแบบแผน ซึ่งผู้วิจัยได้วางระบบขึ้นมาจากการประสบการณ์การทำงานของผู้วิจัยเอง รวมถึงได้ค้นคว้าหาความรู้เพิ่มเติม จนคิดว่าระบบนี้จะทำให้หุ่นยนต์อิริวัต្តสามารถพัฒนาต่อได้ง่าย และเป็นประโยชน์ต่อผู้วิจัยท่านอื่นที่ต้องการนำระบบพื้นฐานนี้ไปใช้งาน



รูปที่ 4.34: ภาพรวมระบบพื้นฐานของหุ่นยนต์อุตสาหกรรม

ระบบพื้นฐานที่ผู้วิจัยออกแบบขึ้นมา จัดประกอบไปด้วยส่วนสำคัญอยู่ทั้งหมด 6 ส่วน คือ

1. UTHAI-Documents
2. UTHAI-Hardware
3. UTHAI-Common
4. UTHAI-MPPC
5. UTHAI-OPC
6. UTHAI-Tools

## UTHAI-Documents

ในส่วนนี้คือส่วนของงานเอกสารต่างๆที่เกี่ยวข้องกับหุ่นยนต์อิวามาโนยด์

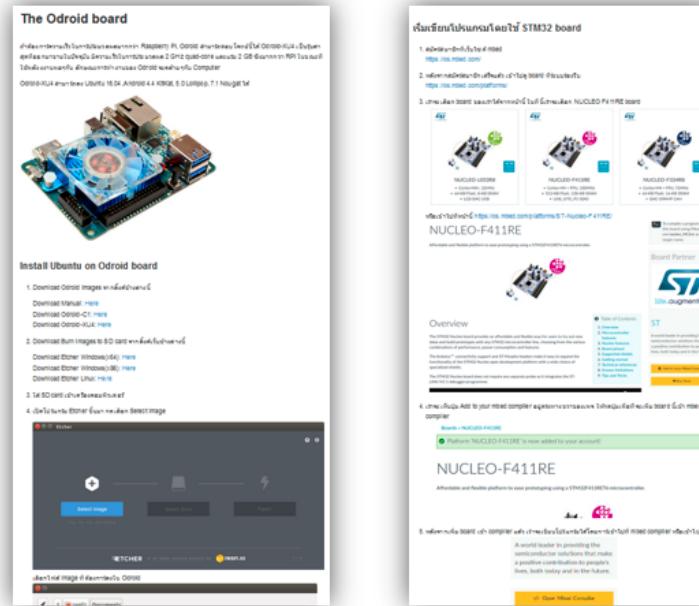
**Reports-Hardware** ใช้สำหรับเก็บรายงานวิทยานิพนธ์ทั้งฉบับร่างและฉบับสมบูรณ์ที่เกี่ยวข้องกับ การปรับเปลี่ยนโครงสร้างทางกล และทางไฟฟ้าของหุ่นยนต์อิวามาโนยด์อุทัย

- การออกแบบโครงสร้างและพัฒนาระบบที่มีความจำเพาะสำหรับหุ่นยนต์อิวามาโนยด์เพื่อการศึกษาและวิจัย

**Reports-Software** ใช้สำหรับเก็บรายงานวิทยานิพนธ์ทั้งฉบับร่างและฉบับสมบูรณ์ที่เกี่ยวข้องกับการเพิ่มความสามารถของหุ่นยนต์อิวามาโนยด์อุทัย

- การพัฒนาระบบการเคลื่อนที่สำหรับหุ่นยนต์อิวามาโนยด์

Wiki ใช้สำหรับเก็บ Tutorial ที่เกี่ยวกับการใช้งานของหุ่นยนต์อิวามาโนยด์อุทัย



รูปที่ 4.35: ภาพตัวอย่าง Tutorial ใน Wiki

ผู้วิจัยท่านอื่นสามารถที่จะซ้ายกันเขียนและพัฒนาได้โดยการ Clone Repository และทำการแก้ไขหลังจากนั้นก็ Pull request ขึ้นมาเพื่อแสดงให้ผู้วิจัยท่านอื่นเห็นด้วย

**UTHAI-Hardware**

ที่เก็บรายละเอียดเกี่ยวกับโครงสร้างทั้งทางกลและทางไฟฟ้า วิธีการประกอบ รายละเอียดการต่อสายไฟ

**UTHAI-Common**

เก็บเกี่ยวกับ ROS Package ที่เป็น model ของตัวหุ่นยนต์พร้อม Simulation

**UTHAI-MPPC**

เครื่อง Motion and Perception PC เก็บเกี่ยวกับ ROS Package ที่ใช้ใน Odroid ใช้สั่งการ ทำสมุด

**UTHAI-OPC**

เครื่อง Operating PC เก็บเกี่ยวกับ ROS Package ที่ใช้สำหรับเข้าไปดูการทำงานและควบคุมหุ่นยนต์ ดูสถานะต่างๆ Moveit Gazebo

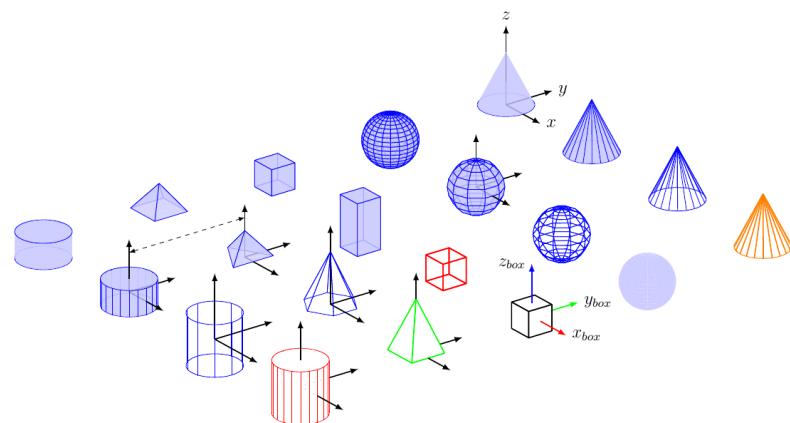
**UTHAI-msgs**

เป็นที่เก็บ message ที่สร้างขึ้นมาเอง รวมไปถึง service และ action ด้วย

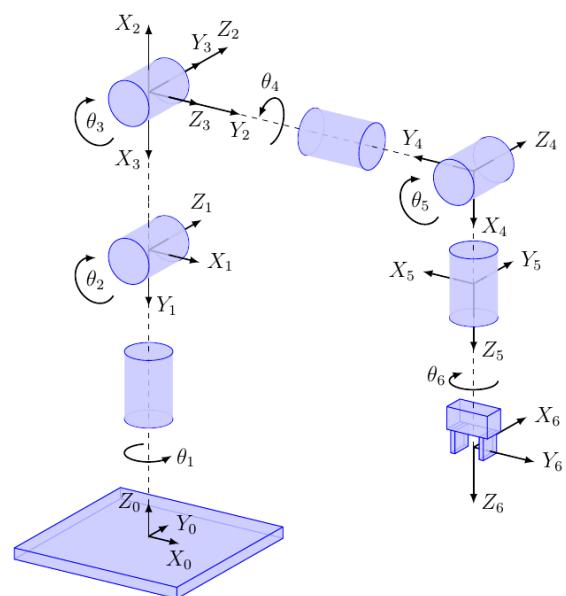
### UTHAI-Tools

เครื่องมือสำหรับการทำงานในชีวามนอยด์

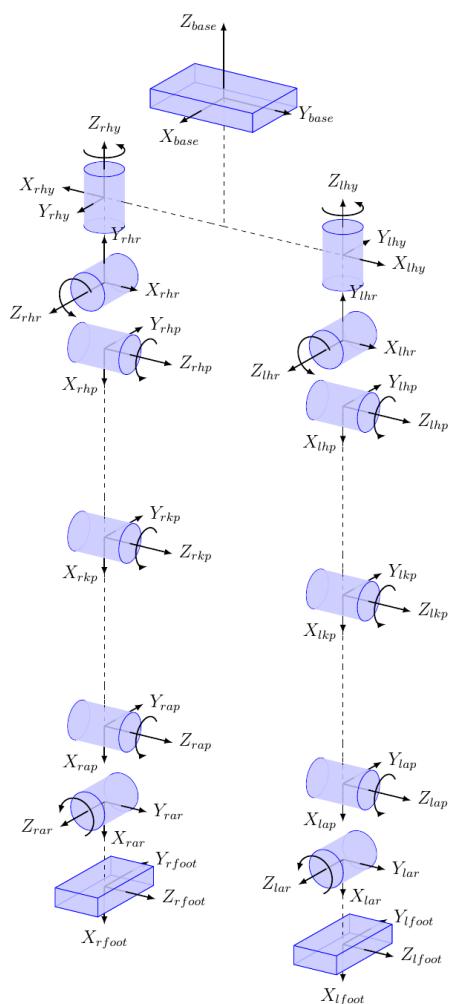
**sketch-lib** เป็นเครื่องมือที่ใช้สำหรับเอาไว้วาดรูปเฟรมของหุ่นยนต์



รูปที่ 4.36: ภาพตัวอย่างการวาดรูปเฟรมของหุ่นยนต์



รูปที่ 4.37: ภาพตัวอย่างการวาดรูปเฟรมของแขนกล



รูปที่ 4.38: ภาพตัวอย่างการวัดเฟรมของหุ่นยนต์ขั้วมานอยด์

## บทที่ 5

### สรุปผลการทดลองและข้อเสนอแนะ

5.1 การออกแบบโครงสร้างของหุ่นยนต์

5.2 การออกแบบโปรแกรมด้วย ROS

5.3 การออกแบบระบบพื้นฐาน

5.4 สรุปภาพรวม

ทำได้ดีนะจ๊ะ

## ประวัติผู้เขียน

นายจิรภพ ศรีรัตนอาภรณ์



ชื่อ สกุล	นายจิรภพ ศรีรัตนอาภรณ์
รหัสนักศึกษา	57340500067
วุฒิการศึกษา	วิศวกรรมศาสตรบัณฑิต
ชื่อสถาบัน	วิศวกรรมหุ่นยนต์และระบบอัตโนมัติ
ปีที่สำเร็จการศึกษา	มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี
ทุนการศึกษา	2560
	กินกันตายเทศ
	กิตาฟันนนทกัดา

## ประวัติผู้เขียน

นายเจษฎากร ท่าไชยวงศ์



ชื่อ สกุล	นายเจษฎากร ท่าไชยวงศ์
รหัสนักศึกษา	57340500067
วุฒิการศึกษา	วิศวกรรมศาสตรบัณฑิต
ชื่อสถาบัน	วิศวกรรมหุ่นยนต์และระบบอัตโนมัติ
ปีที่สำเร็จการศึกษา	มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี
ทุนการศึกษา	2560
	กินกันตายเทศ
	กิตาฟันนนทกัดๆ

## ประวัติผู้เขียน

นายวุฒิภัทร โชคอนันตทรัพย์



ชื่อ สกุล	นายวุฒิภัทร โชค_anantraphi
รหัสนักศึกษา	57340500067
วุฒิการศึกษา	วิศวกรรมศาสตรบัณฑิต
	วิศวกรรมหุ่นยนต์และระบบอัตโนมัติ
ชื่อสถาบัน	มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี
ปีที่สำเร็จการศึกษา	2560
ทุนการศึกษา	กินกันตายเทคโนโลยี

## ประวัติอาจารย์ที่ปรึกษา

นายธนัชชา ชูพจน์เจริญ



ชื่อ สกุล

วุฒิการศึกษา

ชื่อสถาบัน

ปีที่สำเร็จการศึกษา

นายธนัชชา ชูพจน์เจริญ

วิศวกรรมศาสตรบัณฑิต

วิศวกรรมหุ่นยนต์และระบบอัตโนมัติ

มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี

2560