# ระบบควบคุมแขนกลด้วยการติดตามโครงกระดูกและท่าทางมือ

# The Robotic Arm with Skeleton Tracking and Hand Gesture


**นายปารเมศ เมฆสูงเนิน**

**นายปิยังกูร วุฒิเจริญภูรี**

**นายเมศิษฏ์ นาคำ**


**ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต**

**สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์**

**สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง**

**ปีการศึกษา 2563**

ปริญญานิพนธ์ปีการศึกษา 2563

ภาควิชาวิศวกรรมศาสตร์คอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบควบคุมแขนกลด้วยการติดตามโครงกระดูกและท่าทางมือ

THE ROBOTIC ARM WITH SKELETON TRACKING AND HAND GESTURE


ผู้จัดทำ

| | | | |
|---|---|---|---|
| 1.นายปารเมศ | เมฆสูงเนิน | รหัสนักศึกษา | 60010611 |
| 2.นายปิยังกูร | วุฒิเจริญภูรี | รหัสนักศึกษา | 60010621 |
| 3.นายเมศิษฎ์ | นาคำ | รหัสนักศึกษา | 60010839 |


_____อาจารย์ที่ปรึกษา

(อาจารย์จิระศักดิ์ สิทธิกร)

# ระบบควบคุมแขนกลด้วยการติดตามโครงกระดูกและท่าทางมือ


นายปารเมศ เมฆสูงเนิน 60010611

นายปิยังกูร วุฒิเจริญภูรี 60010621

นายเมศิษฎ์ นาคำ 60010839

อาจารย์จิระศักดิ์ สิทธิกร อาจารย์ที่ปรึกษา


ปีการศึกษา 2563

## บทคัดย่อ


มือเป็นอวัยวะที่มีความสำคัญในร่างกาย ในการทำงานหลายอย่าง อาทิเช่น การสัมผัสสารเคมีที่มีอันตรายต่อผิวหนัง การใช้เครื่องจักรในโรงงานผลิตหรือการเก็บกู้วัตถุอันตรายมีโอกาสเสี่ยงต่อการเกิดอุบัติเหตุกับมือจนอาจทำให้ไม่สามารถใช้งานได้

โครงงานจึงได้พัฒนาแขนกลมีที่ขนาดและรูปร่างคล้ายมือ และแขนของมนุษย์ สามารถขยับและเคลื่อนไหวได้เหมือนการเคลื่อนไหวของมือและช่วงแขนของมนุษย์ มาช่วยลดอันตรายต่าง ๆ จากการสัมผัสโดยตรงของมนุษย์และยังสามารถจับวัตถุและเคลื่อนย้ายวัตถุได้โดยระบบควบคุมแขนกลด้วยการติดตามโครงกระดูกและท่าทางมือ (The Robotic Arm with Skeleton Tracking and Hand Gesture) โดยใช้อุปกรณ์ Leap Motion และ Kinect มาเป็นอุปกรณ์ตรวจจับท่าทางเพื่อที่จะสั่งการให้แขนกลทำงานตามท่าทางที่มนุษย์เคลื่อนไหว

# The Robotic Arm with Skeleton Tracking and Hand Gesture


| | | |
|---|---|---|
| Mr. Paramete | Meksoongnern | 60010611 |
| Mr. Piyangkul | Wutthicharoenpuree | 60010621 |
| Mr. Mesit | Nakam | 60010839 |
| Mr. Jirasak | Sittigorn | Advisor |

Academic Year 2563



## ABSTRACT

Hands are an integral part of life, so they should be used especially when handling them so not to cause accidents or injuries to them. Hand accidents can be caused by use of sharp objects, chemicals that are harmful to the skin, Large machines in the factory and different pathogens contact by touch and Repair object or will be a storage of dangerous objects.

This project has created a mechanical arm that is the same size and shape as a hand and human arms can move and move just like the movement of human hands and arms. In order to reduce the danger of human direct contact and can also grasp and move objects. using the Leap Motion device and Kinect is a gesture sensor to instruct the mechanical arm to perform gestures.

# กิตติกรรมประกาศ

ปริญญานิพนธ์เล่มนี้สำเร็จลุล่วงไปด้วยดี ด้วยคำแนะนำและความช่วยเหลือจากอาจารย์ที่ปรึกษา อาจารย์จิระศักดิ์ สิทธิกร ที่ให้คำปรึกษา คำแนะนำ การวางแผนการตลอดการทำงาน นอกจากนี้ยังช่วยแก้ไขปัญหาที่เกิดขึ้น ทางคณะผู้จัดทำต้องขอบคุณอาจารย์เป็นอย่างสูง และขอบคุณภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่เอื้อเฟื้อสถานที่สำหรับการทำงาน และค้นคว้าหาความรู้

ขอขอบคุณเพื่อน ๆ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่คอยให้คำแนะนำตลอดการทำงาน ขอขอบคุณครอบครัว โดยเฉพาะคุณพ่อ และคุณแม่ที่คอยเป็นกำลังใจและให้การสนับสนุนผู้จัดทำตลอดมา

ปารเมศ เมฆสูงเนิน
ปิยังกูร วุฒิเจริญภูรี
เมศิษฎ์ นาคำ

# สารบัญ

หน้า

# สารบัญ(ต่อ)

**หน้า**

# สารบัญรูป

**รูป** **หน้า**

# สารบัญรูป(ต่อ)

**รูป** **หน้า**

# สารบัญรูป(ต่อ)

**รูป** **หน้า**

# สารบัญตาราง

**ตาราง** **หน้า**

# บทที่ 1
# บทนำ

## 1.1 ความเป็นมาของปัญหา

มือเป็นอวัยวะที่สำคัญในการใช้ชีวิต ถ้าหากเกิดอุบัติเหตุหรือการบาดเจ็บจนไม่สามารถใช้งานมือได้ปกติทำให้การใช้ชีวิตลำบากขึ้น ดังนั้นควรให้ความสำคัญในการใช้มือไม่ให้เกิดอุบัติเหตุเช่น จากการใช้สิ่งของมีคมต่าง ๆ สารเคมีที่มีอันตรายต่อผิวหนัง เครื่องจักรขนาดใหญ่ในโรงงานผลิต เชื้อโรคต่าง ๆ ที่ติดต่อได้จากการสัมผัสได้หรือไม่ว่าจะเป็นในด้านการทดลองคิดค้น การซ่อมแซมสิ่งของ หรือการเก็บกู้วัตถุอันตราย ทุกอย่างล้วนมีความเสี่ยงและอันตราย

ดังนั้นจึงได้พัฒนาแขนกลมีขนาดและรูปร่างคล้ายมือและช่วงแขนของมนุษย์ สามารถขยับและเคลื่อนไหวได้เหมือนการเคลื่อนไหวของมือและช่วงแขนของมนุษย์ ไม่ว่าจะเป็นส่วนของข้อต่อนิ้วต่าง ๆ การงอนิ้ว การพับข้อมือหรือการขยับอื่น ๆ ของมือก็สามารถเคลื่อนไหวได้เสมือนของมนุษย์ โดยจะใช้การสั่งการจากมนุษย์โดยการตรวจจับการเคลื่อนไหวของนิ้วมือ มือ ข้อศอก และหัวไหล่ ซึ่งแขนกลนี้จะมาช่วยลดอันตรายต่าง ๆ จากการสัมผัสโดยตรงของมนุษย์

## 1.2 วัตถุประสงค์ของโครงงาน

1) เพื่อพัฒนาแขนกลต้นแบบเป็นอุปกรณ์ที่ทำหน้าที่แทนมือและแขนของมนุษย์
2) เพื่อใช้แขนกลเป็นอุปกรณ์ทดแทนการสัมผัสสิ่งแปลกปลอมหรือสิ่งของที่อันตรายโดยตรง
3) เพื่อพัฒนาระบบควบคุมแขนกลโดยใช้อุปกรณ์ Leap Motion และ Kinect

## 1.3 ขอบเขตของโครงงาน

1) พัฒนาแขนกลต้นแบบที่สามารถเคลื่อนไหวได้ใกล้เคียงกับการเคลื่อนไหวของมือและแขนของมนุษย์
2) สามารถใช้แขนกลสามารถหยิบจับวัตถุสิ่งของและสามารถเคลื่อนย้ายวัตถุได้
3) สามารถใช้ข้อมูลที่ได้รับจากอุปกรณ์ Leap Motion และ Kinect ได้
4) สามารถส่งข้อมูลเพื่อควบคุมแขนกลผ่านเครือข่ายอินเตอร์เน็ตได้

## 1.4 ประโยชน์ที่คาดว่าจะได้รับ

1) ได้รับความรู้ในเรื่องระบบการทำงานของแขนกล
2) สามารถใช้เป็นต้นแบบในการพัฒนาและนำไปใช้ได้

3) ได้รับความรู้จากการใช้งานอุปกรณ์ Leap Motion และ Kinect
4) ได้รับความรู้ในการพัฒนาแขนกลให้เหมาะสมกับอุปกรณ์ที่ใช้งานร่วมกัน
5) สามารถนำความรู้ต่าง ๆ ที่ได้รับจากการทำโครงงานไปประยุกต์ใช้ในชีวิตประจำวัน

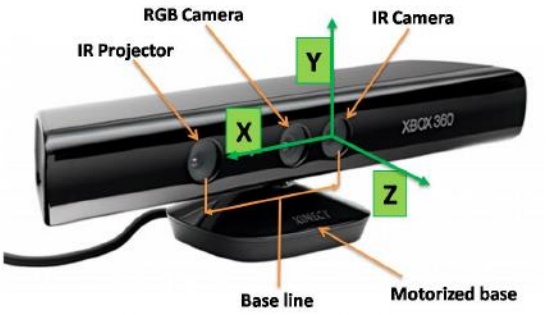
# บทที่ 2
# ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

## 2.1 Kinect

อุปกรณ์ Kinect เป็นอุปกรณ์ที่มีการตรวจจับการเคลื่อนไหวของมนุษย์โดย Kinect จะตรวจจับทั้งตัวของมนุษย์ใช้ระยะการตรวจจับอยู่ที่ความห่างประมาณ 0.8 – 3.5 เมตรในอุปกรณ์ Kinect ประกอบด้วย อุปกรณ์ฉายแสงอินฟาเรด (Infrared) กล้องวัดความลึกของภาพ (Dept Camera) กล้องวีดีโอ (Video Camera) ไมโครโฟน และเซนเซอร์ประมวลผล (Sensor) โดยการทำงานของอุปกรณ์นี้จะเริ่มจากการที่อุปกรณ์ Kinect ฉายแสงอินฟาเรดออกมาเป็นลักษณะจุด ๆ ตามแนวหลังจากนั้นกล้องวัดความลึกจะรับภาพจากความสว่างของแสงอินฟาเรดที่กระทบลงบนวัตถุส่งกลับไปยังเซนเซอร์เพื่อให้ได้ค่าแกน Z ซึ่งจะทำให้สามารถจำลองผลเป็น 3 มิติได้และเมื่อได้ความลึกมาแล้วเซนเซอร์จะสามารถแยกมนุษย์ออกจาสภาพแวดล้อมได้เช่น ผนัง ซึ่งโครงงานนี้ได้มีการใช้ทฤษฎี Skeleton Tracking มาใช้เพื่อที่จะให้อุปกรณ์ Kinect แสดงค่าพิกัด X Y และ Z ในแต่ละจุดบนร่างกาย

### 2.1.1 Skeleton Tracking for Kinect

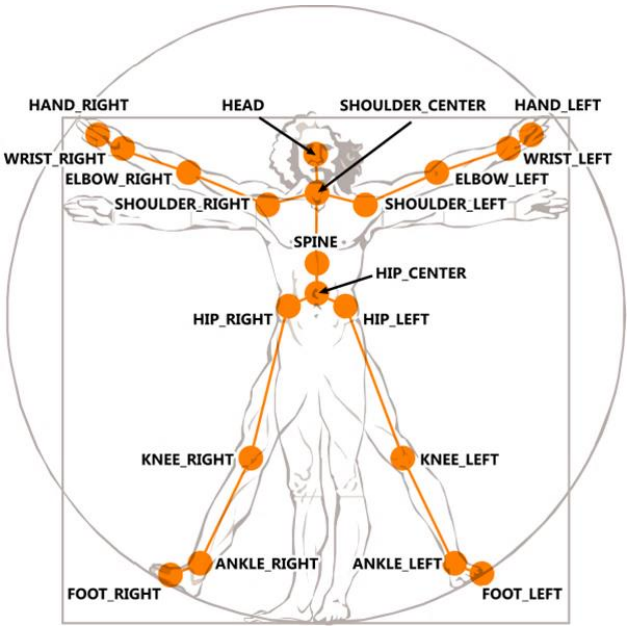
จากความสามารถในการตรวจจับการเคลื่อนไหวและระบุตำแหน่งของร่างกายโดยไม่มีการทำเครื่องหมายบนส่วนต่าง ๆ บนร่างกายของกล้อง Kinect จึงมีการนำไปประยุกต์ใช้ในงานต่าง ๆ รวมถึงงานวิจัยนี้การตรวจจับการเคลื่อนไหวของร่างกายมนุษย์โดยใช้กล้อง Kinect และระบุข้อมูลตำแหน่งของข้อต่อมนุษย์ที่ถูกตรวจจับโดยข้อมูลตำแหน่งของข้อต่อจะอยู่ในรูปแบบระบบพิกัดสามมิติ X Y และ Z ตามรูปที่ 2.1




**รูปที่ 2.1 แกน X, Y, Z ของกล้อง Kinect**

ที่มา: https://www.researchgate.net/publication/236962544_Improving_2D_Reactive_Navigators_with_Kinect

การตรวจจับของกล้อง Kinect ให้ข้อมูลโครง Skeleton ที่เป็นตำแหน่งของข้อต่อของผู้ที่ถูกตรวจจับได้โดยตำแหน่งของกล้องจะเป็นตำแหน่งจุดกำเนิดตามที่แสดงตามรูปที่ 2.1 ซึ่งแกน Z จะเป็นทิศทางที่กล้องตรวจจับและมีค่าเป็นบวกเสมอ แกน X จะเป็นค่าแสดงระยะทางที่ห่างออกไปจากจุดกำเนิดตามแนวนอนที่กล้องตรวจจับ และแกน Y จะเป็นค่าแสดงระยะทางที่ห่างออกไปจากจุดกำเนิดตามแนวตั้งที่กล้องตรวจจับ



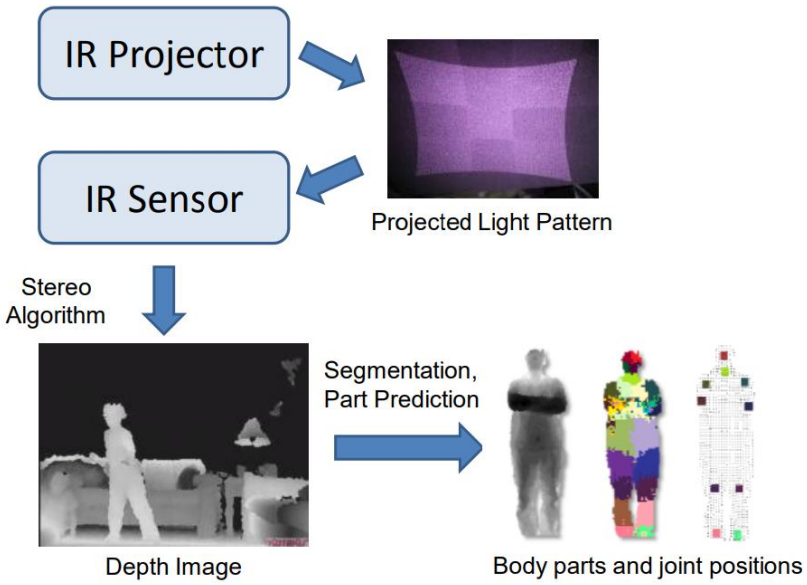

**รูปที่ 2.2 ตำแหน่งข้อต่อของร่างกายมนุษย์ 20 ตำแหน่ง**

ที่มา: https://kinectasia.wordpress.com/tag/tracking/

ตำแหน่งข้อต่อที่ตรวจจับได้จะสัมพันธ์กับตำแหน่งของร่างกายมนุษย์ 20 ตำแหน่งแต่ละตำแหน่งข้อต่อจะมีค่า X Y และ Z ในการตรวจจับตำแหน่งข้อต่อบนร่างกาย หากผู้ที่ถูกตรวจจับยืนหันหน้าเข้าหากล้องในระยะห่าง 1.2 – 3.5 เมตรค่า X Y และ Z ของข้อต่อที่ถูกตรวจจับได้จะมีค่าใกล้เคียงกับการวัดจริงแต่หากมีบางข้อต่อที่กล้องไม่สามารถตรวจจับได้หรือถูกบดบังไป

### 2.1.2 ขั้นตอนการทำ Skeleton Tracking ด้วย Kinect

ภาพรวมขั้นตอนการทำงานของ Kinect ในการทำ Skeleton Tracking

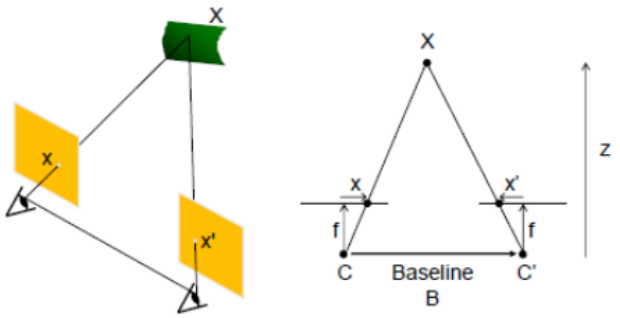



**รูปที่ 2.3 ขั้นตอนการทำ Skeleton Tracking ด้วย Kinect**

ที่มา: http://pages.cs.wisc.edu/~dyer/cs540/notes/17_kinect.pdf

#### 2.1.2.1 Stereo Algorithm

การทำ Stereo Algorithm เป็นขั้นตอนที่ประมวลผลเพื่อสร้างความลึกของภาพ โดยกล้อง Kinect นั้นจะประกอบไปด้วยส่วนของ IR Projector ซึ่งจะเป็นตัวฉายแสงไปยังวัตถุ จากนั้นกล้อง IR Sensor จะทำการพิจารณาแสงเหล่านั้นพร้อมทั้งส่งค่าที่ได้ไปคำนวณหาค่าความลึก




**รูปที่ 2.4 Basic Stereo Matching Algorithm**

ที่มา: http://pages.cs.wisc.edu/~dyer/cs540/notes/17_kinect.pdf

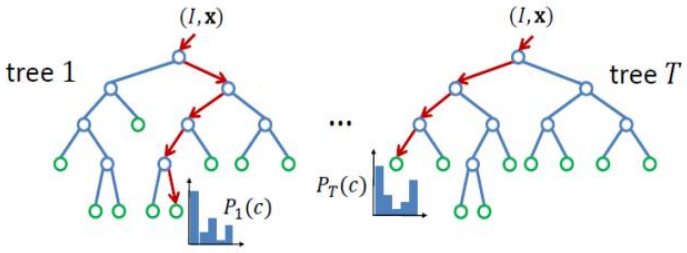
**2.1.2.2 แบ่งส่วนต่าง ๆ ของร่างกายจากความลึกของภาพ (Depth Image)**

เมื่อได้รับค่าความลึกของภาพมาใช้งานแล้ว ในขั้นตอนถัดไปจะทำการแบ่งภาพออกเป็นส่วนต่าง ๆ ของร่างกายทั้งหมด 31 ส่วน ด้วยวิธีการ Random Decision Forests ซึ่งจะมีการนำกลุ่มตัวอย่างเป็นบุคคลที่มีขนาดและสัดส่วนของร่างกายที่แตกต่างกันจากนั้นนำค่าความลึกของภาพที่ได้มาจากกลุ่มตัวอย่างนั้นมาทำการเรียนรู้เพื่อสร้าง Tree สำหรับการแบ่งประเภทของกลุ่มตัวอย่างมีข้อมูลรายละเอียดของการสร้าง Tree ดังต่อไปนี้

1) มีทั้งหมด 3 Tree ซึ่งความลึกของแต่ละ Tree เท่ากับ 20
2) ใช้ Training Image ทั้งหมด 300,000 ภาพ (เลือกแบบสุ่มจาก 1 ล้านภาพ)
3) Training Image แต่ละภาพเลือกใช้ 2,000 พิกเซล
4) Feature (ค่าความลึกของแต่ละพิกเซล) ที่ใช้ในการเรียนรู้ทั้งหมด 2,000 รูปแบบ

**รูปที่ 2.5 Extract Body Pixels by Thresholding Depth**

ที่มา: http://pages.cs.wisc.edu/~dyer/cs540/notes/17_kinect.pdf

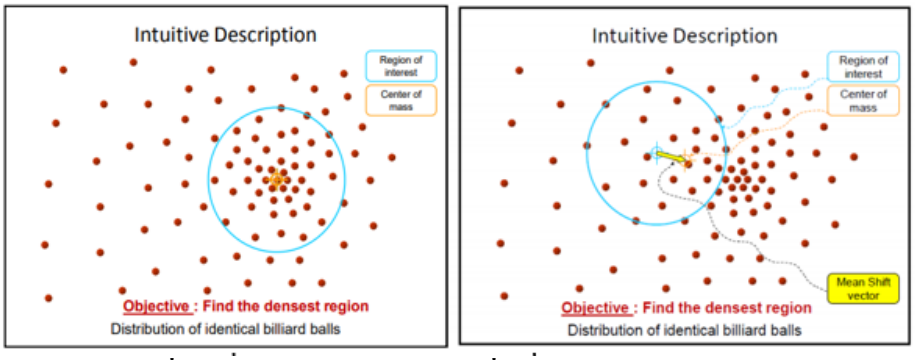



**รูปที่ 2.6 การเรียนรู้ด้วยวิธี Random Forests**

ที่มา: http://pages.cs.wisc.edu/~dyer/cs540/notes/17_kinect.pdf

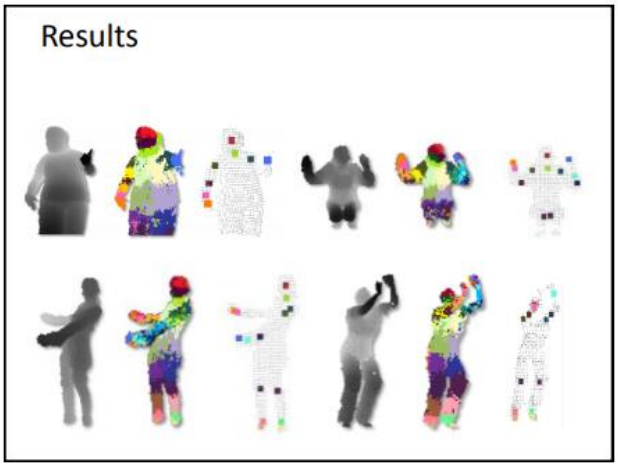
#### 2.1.2.3 การประมาณตำแหน่งของข้อต่อ

เมื่อทำการแบ่งแยกภาพออกเป็นส่วนของร่างกายได้แล้ว ขั้นตอนต่อไปคือการประมาณตำแหน่งของข้อต่อด้วยวิธีการ Mean-Shift Clustering ซึ่งเป็นการสร้างพื้นที่แล้วทำการค้นหาพื้นที่ของแต่ละส่วนของร่างกายว่าพื้นที่ส่วนใดที่มีความหนาแน่นมากที่สุด




**รูปที่ 2.7 Mean-Shift Cluster เพื่อหา Densest Region**

ที่มา: http://pages.cs.wisc.edu/~dyer/cs540/notes/17_kinect.pdf




**รูปที่ 2.8 ผลลัพธ์ของการทำ Skeleton Tracking จาก Depth Image**

ที่มา: http://pages.cs.wisc.edu/~dyer/cs540/notes/17_kinect.pdf

### 2.1.3 ข้อจำกัดในการทำ Skeleton Tracking for Kinect

ในการทำ Skeleton Tracking ของกล้อง Kinect นั้นมีข้อกำหนดดังนี้

1) สามารถตรวจจับจำนวนคนได้สูงสุด 6 คน
2) จำนวนคนที่สามารถระบุรายละเอียดได้สูงสุด 2 คน
3) เซนเซอร์จะต้องสามารถตรวจจับส่วนของศีรษะและท่อนบนของร่างกายได้
4) ระยะห่างจากกล้องที่สามารถตรวจจับได้อยู่ที่ระยะ 80 เซนติเมตรถึง 4 เมตร

**2.1.4 สถานะในการตรวจจับ Skeleton (Skeleton tracking state)**

กล้อง Kinect จะทำการแจ้งสถานะว่าสามารถตรวจจับ Skeleton ได้หรือไม่เป็น 3 แบบดังนี้

1) ตรวจจับได้
2) ตรวจจับได้แต่ได้เฉพาะตำแหน่งโดยรวมของ Skeleton
3) ตรวจจับไม่ได้

**2.1.5 คุณภาพในการตรวจจับ Skeleton (Skeleton Information Quality)**

การตรวจจับจะมีประสิทธิภาพลดลง เมื่อผู้ควบคุมอยู่นอกระยะที่กล้องสามารถตรวจจับได้ เช่น อยู่ใกล้หรือไกลเกินไป อยู่สูงหรือต่ำเกินไป หรืออยู่ทางซ้ายหรือขวาของกล้องมากไป โดยกล้อง Kinect จะมีการตรวจจับข้อมูลดังกล่าว โดยดูว่าผู้ควบคุมอยู่ในตำแหน่งที่ตัดกับเฟรมของกล้องหรือไม่ โดยให้ผลลัพธ์ออกมาในตัวแปรที่ชื่อว่า Clipped Edges ซึ่งตัวแปรนี้จะเป็นผลรวมของสถานะ Frame Edges ที่มีค่าที่เป็นไปได้ดังนี้

1) 0 เท่ากับ สถานะปกติไม่อยู่ชิดขอบเฟรม
2) 1 เท่ากับ อยู่ชิดขอบเฟรมด้านขวา
3) 2 เท่ากับ อยู่ชิดขอบเฟรมด้านซ้าย
4) 4 เท่ากับ อยู่ชิดขอบเฟรมด้านบน
5) 8 เท่ากับ อยู่ชิดขอบเฟรมด้านล่าง

**2.1.6 สถานะในการตรวจจับข้อต่อ (Joint Tracking State)**

ตำแหน่งของข้อต่อ (Joints) ที่ได้รับจาก API สามารถระบุตำแหน่งที่สัมพันธ์กับร่างกายมนุษย์ได้ 20 ตำแหน่ง เมื่อกล้อง Kinect สามารถทำการตรวจจับร่างกายของมนุษย์ได้จะทำการแสดงตำแหน่งข้อต่อที่สลับข้างซ้ายขวากันแต่ละตำแหน่งของข้อต่อมีค่า X Y และ Z โดยมีค่าความเชื่อมั่น (Confidence) ที่แสดงถึงค่าความเชื่อมั่นในการตรวจจับตำแหน่งข้อต่อนั้น ๆ บางครั้งกล้อง Kinect ก็สามารถตรวจจับข้อต่อได้ชัดเจน หรือบางครั้งก็ไม่ได้ขึ้นอยู่กับหลายปัจจัย ดังนั้นตัวกล้องเองก็จะแสดงค่าสถานะ (Joint Tracking State) ของแต่ละข้อต่ออยู่ 3 แบบดังนี้

1) ตรวจจับได้
2) คาดการณ์ได้คือตรวจจับข้อต่อไม่ได้แต่พอคาดการณ์ได้ว่าข้อต่อนั้นอยู่ที่ไหน จากข้อต่อข้างเคียงสถานะนี้อาจเกิดขึ้นในกรณีที่ข้อต่อนั้นถูกบดบังด้วยวัตถุอื่นหรือด้วยเสื้อผ้าที่สวม
3) ตรวจจับไม่ได้

## 2.2 Leap Motion

ระบบ Leap Motion ทำหน้าที่ตรวจสอบและติดตามมือนิ้วมือของผู้ควบคุมโดยอุปกรณ์จะทำงานในระยะใกล้ชิดด้วยความแม่นยำสูงและรายงานตำแหน่งการเคลื่อนไหวอย่างต่อเนื่องเมื่อมีการเคลื่อนไหว

Leap Motion ใช้เซ็นเซอร์ออปติคัลและแสงอินฟราเรด เซ็นเซอร์จะฉายขึ้นไปตามแกน y เมื่อตัวควบคุมอยู่ในตำแหน่งมาตรฐาน และมีมุมมองประมาณ 150 องศา โดยช่วงที่มีประสิทธิภาพของ Leap Motion Controller ประมาณ 25 ถึง 600 มม. เหนืออุปกรณ์ (1 นิ้วถึง 2 ฟุต)

**รูปที่ 2.9 ตัวอย่างการใช้งาน Leap Motion**

ที่มา: https://developer-archive.leapmotion.com

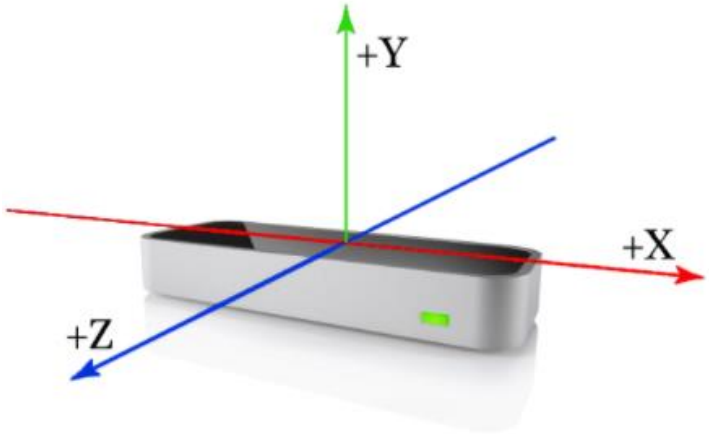
การตรวจจับและติดตามจะทำงานได้ดีที่สุดเมื่อผู้ควบคุมมีมุมมองที่ชัดเจนและคอนทราสต์ของภาพเงาของวัตถุสูงซอฟต์แวร์ Leap Motion จะทำการรวมข้อมูลเซ็นเซอร์เข้ากับแบบจำลองของมือมนุษย์

### 2.2.1 Coordinate Systems

งานพื้นฐานเมื่อใช้ตัวควบคุม Leap Motion ในแอปพลิเคชันคือการแมปค่าพิกัดที่ได้รับจากผู้ควบคุมกับระบบพิกัดที่กำหนดโดยแอปพลิเคชันให้เหมาะสม โดย Leap Motion จะใช้พิกัดเป็นหน่วยมิลลิเมตรแต่ความเป็นจริงภายใน Leap Motion นั่นคือถ้ากำหนดตำแหน่งของปลายนิ้วเป็น (x, y, z) = (±100, 100, ±100) เป็นมิลลิเมตรหรือเท่ากับ x = ±10 cm, y = 10 cm, z = ±10 cm Leap Controller นั้นเป็นศูนย์กลางของกรอบอ้างอิง จุดกำเนิดจะอยู่ที่ด้านบนตรงกลางของตัวอุปกรณ์ นั่นคือถ้าทำการแตะตรงกลางของ Leap Motion พิกัดของปลายนิ้วของคุณจะเป็น (0, 0, 0)

**ตารางที่ 2.1 หน่วยวัดปริมาณทางกายภาพอุปกรณ์ Leap Motion**

| | |
|---|---|
| Distance | Millimeters |
| Time | Microseconds (unless otherwise noted) |
| Speed | Millimeter/second |
| Angle | radians |



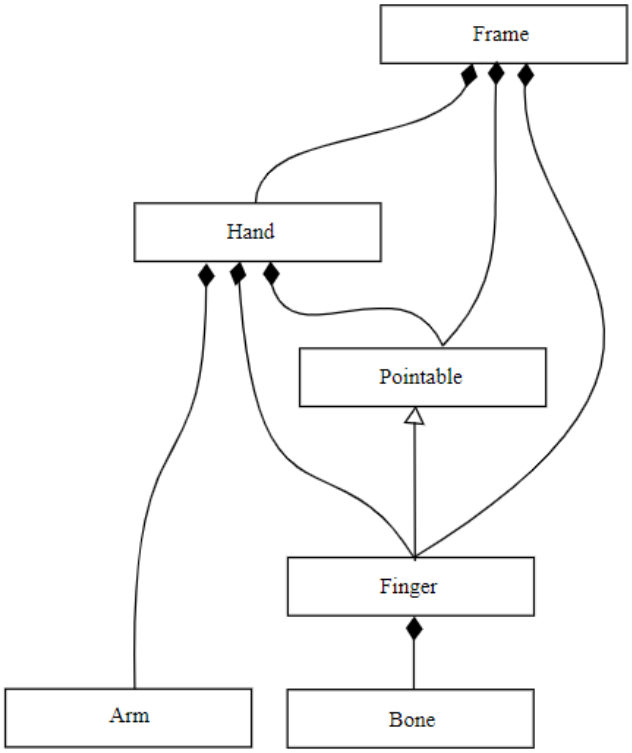

**รูปที่ 2.10 แกน X, Y, Z ของอุปกรณ์ Leap Motion**

ที่มา: https://developer-archive.leapmotion.com/documentation

จากรูปที่ 2.10 คือตำแหน่งปกติวางบนโต๊ะโดยมีผู้ใช้อยู่ด้านหนึ่งและอีกด้านหนึ่งคือจอคอมพิวเตอร์ของผู้ใช้อยู่ด้านหน้า ( + Z ) ของผู้ควบคุมและหน้าจอมอนิเตอร์อยู่ข้างหลัง ( - Z ) ตัวควบคุม หากผู้ใช้เปิดใช้งานการวางแนวอัตโนมัติซอฟต์แวร์ Leap Motion จะปรับระบบพิกัดหากตัวควบคุมกลับด้าน (LED สีเขียวหันออกจากผู้ใช้) อย่างไรก็ตามหากผู้ใช้วาง Leap Motion ในตำแหน่งอื่น (คว่ำหรือด้านข้าง) ซอฟต์แวร์ Leap Motion จะไม่สามารถตรวจจับหรือปรับค่านี้ได้ ผู้ใช้งานไม่ควรเข้าใกล้ Leap Motion Controller มากเกินไปด้วยตัวชี้นำภาพ การวางมือไว้เหนือตัวควบคุมจะเพิ่มโอกาสที่นิ้วและมือจะปิดกั้นมุมมอง

### 2.2.2 Motion tracking data

เนื่องจาก Leap Motion ติดตามมือและนิ้วในมุมมองของอุปกรณ์จึงมีการอัปเดตข้อมูลเป็นชุดหรือ Frame ออบเจ็กต์ Frame จะมีรายละเอียดมือที่ถูกติดตามในช่วงเวลานั้นๆ และนำมาเป็นแบบจำลองข้อมูล Leap Motion




**รูปที่ 2.11 Tracking Model**

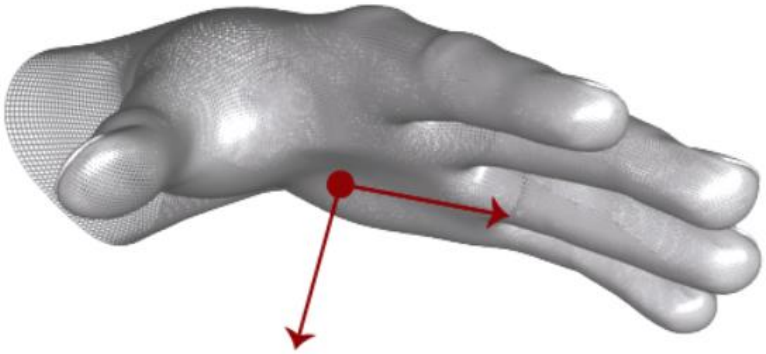
ที่มา: https://developer-archive.leapmotion.com/documentation/javascript/devguide

#### 2.2.2.1 Frames

Leap Motion API จะทำการนำข้อมูลการติดตามการเคลื่อนไหวไปยังแอปพลิเคชันเป็นชุดภาพรวมที่เรียกว่า rFame ข้อมูลการติดตามแต่ละ Frame ประกอบด้วยตำแหน่งที่วัดได้และข้อมูลอื่น ๆ เกี่ยวกับแต่ละEntityที่ตรวจพบในแต่ละการบันทึกในช่วงเวลานั้น รายละเอียดของการรับวัตถุ Frame จาก Leap Motion API วัตถุ Frame แต่ละชิ้นมีข้อมูลของฉากที่บันทึกโดยตัวควบคุม Leap Motion ในทันทีที่ Leap Motion WebSocket ส่งFrameไปยังแอปพลิเคชัน เมื่อสร้างขึ้น ข้อมูลเหล่านี้จะถูกส่งไปยัง Leap Motion JavaScript API (leap.js)

**2.2.2.2 Hands**

แบบจำลองมือให้ข้อมูลเกี่ยวกับตัวตนตำแหน่งและลักษณะอื่น ๆ ของมือที่ตรวจพบแขนที่ติดกับมือและนิ้วที่เกี่ยวข้องกับมือ



**รูปที่ 2.12 เวกเตอร์ชี้ออกจากมือในแนวตั้งฉากทิศทางชี้ไปข้างหน้า**

ที่มา: https://developer-archive.leapmotion.com/documentation/javascript/devguid

ซอฟต์แวร์ Leap Motion ใช้แบบจำลองมือมนุษย์เพื่อให้การติดตามคาดเดาแม้ว่าจะมองไม่เห็นส่วนต่าง ๆ ของมือก็ตาม แบบจำลองมือจะให้ตำแหน่งสำหรับห้านิ้วเสมอแม้ว่าการติดตามจะเหมาะสมที่สุดเมื่อมองเห็นเงาของมือและนิ้วทั้งหมดได้อย่างชัดเจน ซอฟต์แวร์ใช้ส่วนที่มองเห็นได้ของมือแบบจำลองภายในและการสังเกตในอดีตเพื่อคำนวณตำแหน่งที่เป็นไปได้มากที่สุดของชิ้นส่วนที่มองไม่เห็นในปัจจุบัน การเคลื่อนไหวเล็กน้อยของนิ้วที่ติดกับมือเซ็นเซอร์ Leap Motion จะไม่สามารถตรวจจับได้และมือมากกว่าสองมือสามารถปรากฏในเฟรมได้หากมีมือของบุคคลมากกว่าหนึ่งคนหรือวัตถุที่คล้ายมืออื่น ๆ อยู่ในมุมมอง แต่ในความจริงไม่ควรเกินสองมือในมุมมองของ Leap Motion Controller เพื่อคุณภาพการติดตามการเคลื่อนไหวที่ดีที่สุด

**2.2.2.3 Arms**

เป็นวัตถุคล้ายกระดูกที่มีการวางแนวความยาวความกว้างและจุดสิ้นสุดของแขน เมื่อไม่ได้สนใจข้อศอก ตัวควบคุม Leap Motion จะประมาณตำแหน่งตามการสังเกตในอดีตรวมทั้งสัดส่วนของมนุษย์ทั่วไป

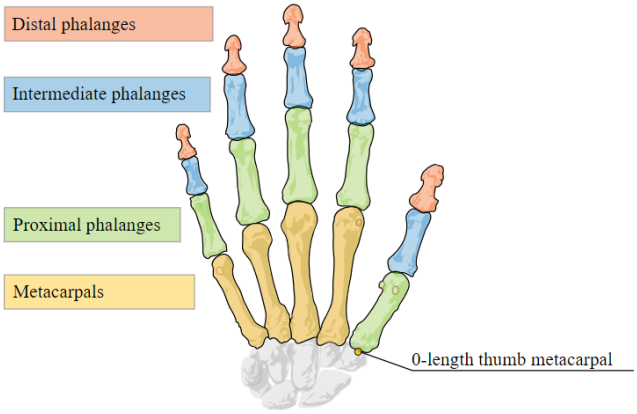
**2.2.2.4 Fingers**

ตัวควบคุม Leap Motion ให้ข้อมูลเกี่ยวกับนิ้วแต่ละนิ้วบนมือ หากมองไม่เห็นนิ้วทั้งหมดหรือบางส่วนจะมีการประมาณลักษณะของนิ้วตามการสังเกตล่าสุดและแบบจำลองทางกายวิภาคของมือ นิ้วจะถูกระบุตามชื่อประเภทเช่นนิ้วโป้ง นิ้วกลาง และนิ้วก้อย

นิ้วมือมีคลาส PointableList และ FingerList มีโครงสร้างที่คล้ายกันได้ทำการออกแบบมาเพื่อทำหน้าที่เหมือนอาร์เรย์แบบเวกเตอร์ สามารถลบหรือเปลี่ยนแปลงอ็อบเจ็กต์สมาชิกของข้อมูลที่ได้รับจาก Leap API และสามารถรวมรายการประเภท อ็อบเจ็กต์เดียวกันได้ คลาส PointableList และ FingerList จะกำหนดฟังก์ชันเพิ่มเติมสำหรับการรับค่าของรายการตามตำแหน่งต่าง ๆ ภายในระบบพิกัด Leap ฟังก์ชันเหล่านี้ ได้แก่ leftmost() rightmost() และ frontmost()

**รูปที่ 2.13 เวกเตอร์ตำแหน่งและทิศทางให้ตำแหน่งของปลายนิ้วและทิศทางทั่วไปที่นิ้วชี้**

ที่มา: https://developer-archive.leapmotion.com/documentation/javascript/devguide




**รูปที่ 2.14 กระดูกบริเวณนิ้วมือ**

ที่มา: https://developer-archive.leapmotion.com/documentation/javascript/devguide

โดยมีการระบุดังนี้

1) Metacarpal – กระดูกภายในมือเชื่อมต่อนิ้วกับข้อมือ (ยกเว้นนิ้วหัวแม่มือ)
2) Proximal Phalanx – กระดูกที่ฐานของนิ้วเชื่อมต่อกับฝ่ามือ
3) Intermediate Phalanx – กระดูกตรงกลางของนิ้วระหว่างปลายและฐาน
4) Distal Phalanx – กระดูกขั้วที่ปลายนิ้ว

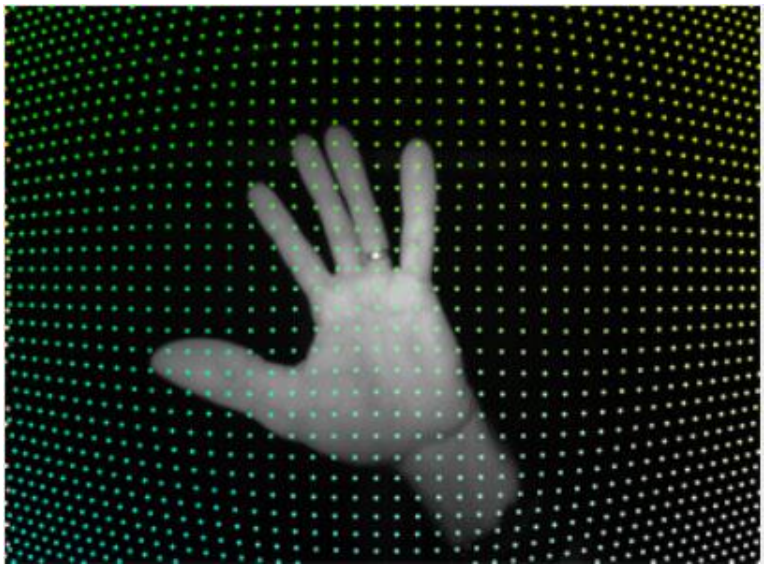
จากรูปที่ 2.14 นิ้วหัวแม่มือนี้ไม่ตรงกับระบบการตั้งชื่อทางกายวิภาคมาตรฐาน นิ้วหัวแม่มือจริงมีกระดูกน้อยกว่านิ้วอื่น ๆ เพื่อความสะดวกในการเขียนโปรแกรมแบบจำลอง Leap Motion thumb ประกอบด้วยกระดูกฝ่ามือที่มีความยาวเป็นศูนย์เพื่อให้นิ้วหัวแม่มือมีกระดูกจำนวนเท่ากันกับนิ้วอื่น ๆ ด้วยเหตุนี้กระดูกฝ่ามือทางกายวิภาคของนิ้วหัวแม่มือจึงถูกระบุว่าเป็นอวัยวะใกล้เคียงและอวัยวะใกล้เคียงทางกายวิภาคจะถูกระบุว่าเป็นกลุ่มกลางในแบบจำลองกระดูกนิ้ว Leap Motion

**2.2.2.5 Transforming Finger Coordinates**

การได้รับพิกัดของนิ้วมือตามกรอบอ้างอิงของมือก็มีประโยชน์ซึ่งวิธีนี้ช่วยให้จัดเรียงนิ้วในเชิงพื้นที่และสามารถวิเคราะห์ตำแหน่งนิ้วได้ง่ายขึ้น โดยสามารถสร้างเมทริกซ์การแปลงโดยใช้คลาส Leap Matrix เพื่อเปลี่ยนตำแหน่งนิ้วและพิกัดทิศทาง

**2.2.3 Sensor Images**

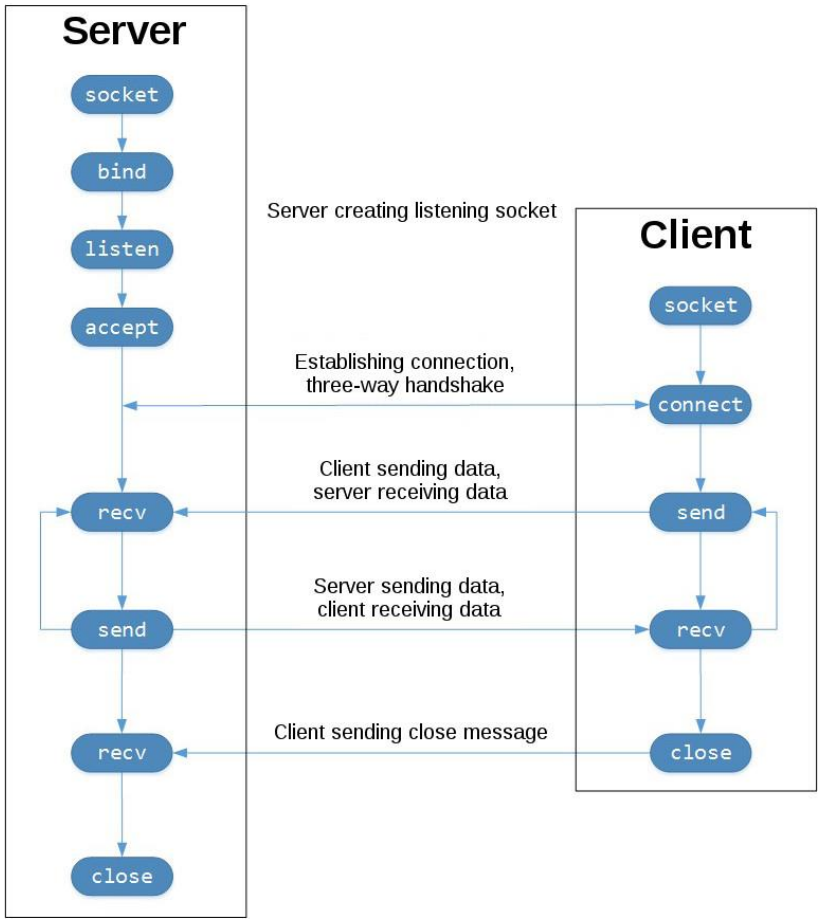
นอกจากข้อมูลที่คำนวณแล้วยังรับภาพเซ็นเซอร์จากกล้อง Leap Motion ได้ข้อมูลภาพประกอบด้วยค่าความสว่าง IR ที่วัดได้และข้อมูลการปรับเทียบที่จำเป็นเพื่อแก้ไขความผิดเพี้ยนของเลนส์



**รูปที่ 2.15 ภาพเซ็นเซอร์จากกล้อง Leap Motion**

ที่มา: https://developer-archive.leapmotion.com/documentation/javascript/devguide

## 2.3 TCP Sockets




**รูปที่ 2.16 TCP Socket Flow**

ที่มา: https://realpython.com/python-sockets

จากรูปที่ 2.16 ทางซ้ายมือแสดงถึงเซิร์ฟเวอร์ ทางด้านขวามือคือไคลเอนต์สังเกตว่า API เรียกใช้เซิร์ฟเวอร์เพื่อตั้งค่าซ็อกเก็ต (listening socket) โดยจะมีฟังก์ชั่นที่ใช้กำหนดค่าต่าง ๆ ดังนี้ socket(), bind(), listen(), accept() การตั้งค่าซ็อกเก็ต (listening socket) จะทำการรอรับการเชื่อมต่อจาก ไคลเอนต์ เมื่อไคลเอนต์เชื่อมต่อเซิร์ฟเวอร์จะเรียกใช้ฟังก์ชั่น accept () เพื่อยอมรับหรือดำเนินการเชื่อมต่อให้เสร็จสมบูรณ์ หลังจากนั้นไคลเอ็นต์เรียกใช้ connect () เพื่อสร้างการเชื่อมต่อกับเซิร์ฟเวอร์และเริ่มทำการตั้งค่าเบื้องต้น (three-way handshake) ขั้นตอนตั้งค่าเบื้องต้น (three-way handshake) เป็นสิ่งสำคัญเนื่องจากทำให้มั่นใจได้ว่าการเชื่อมต่อแต่ละด้านสามารถเข้าถึงได้ในเครือข่าย กล่าวอีกนัยหนึ่งคือไคลเอนต์และเซิร์ฟเวอร์สามารถเข้าถึงซึ่งกันและกันได้ตรงกลางคือส่วนที่ข้อมูลมีการแลกเปลี่ยนข้อมูลระหว่างไคลเอนต์และเซิร์ฟเวอร์โดยใช้ฟังก์ชั่น send() เพื่อส่งและ recv () เพื่อรับข้มูลที่ด้านล่างไคลเอนต์และเซิร์ฟเวอร์จะปิดซ็อกเก็ตโดยใช้ฟังก์ชั่น close()

## 2.4 ระบบพิกัดสามมิติ การแปลงทางเรขาคณิต และการหาทิศทาง

กล้อง Kinect จะใช้ระบบพิกัดสามมิติในการทำงานค่าของตัวเลขที่ใช้ในการอธิบายตำแหน่งของจุดบนระนาบโดยใช้วิธีการกำหนดอย่างเป็นระบบด้วยการกำหนดค่าตัวเลขชุดอันดับสามตัวแทนตำแหน่งของแต่ละจุดบนระนาบชุดอันดับหนึ่งชุดจะสื่อถึงตำแหน่งเพียงหนึ่งตำแหน่งเท่านั้น เช่น จุด P เป็นจุดใด ๆ ที่อยู่ระหว่างแกน X Y และ Z โดยเมื่อระบุตำแหน่ง P จะได้ค่าตัวเลขออกมาหนึ่งชุดเป็นค่า (X, Y, Z) ในส่วนการแปลงทางเรขาคณิต (Geometric Trans - formation)

### 2.4.1 การย้าย (Translation)

Translation หมายถึงการเคลื่อนตำแหน่งของวัตถุในทุก ๆ จุดเป็นระยะทางที่เท่ากันและเคลื่อนไปในทิศทางเดียวกันโดยการย้ายในระบบพิกัดสามมิติสามารถนำเสนอในรูปแบบเมตริกได้ดังสมการที่ 2.1 ถึง 2.7

$$\acute{P} = P + d \tag{2.1}$$

$$\begin{bmatrix} \acute{P}_x \\ \acute{P}_y \\ \acute{P}_z \end{bmatrix} = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} + \begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix} \tag{2.2}$$

เมื่อ $P$ คือ จุดใด ๆ ในระบบพิกัดสามมิติ

$d$ คือ ระยะการเคลื่อนในแต่ละแกน

$\acute{P}$ คือ จุดใหม่ที่เกิดจากการเคลื่อนตำแหน่ง

โดยหากจัดให้อยู่ในรูปแบบเมตริกแบบ Homogeneous Coordinate จะได้เมตริกทั้งสองรูปแบบ

$$\acute{P} = d \cdot P \tag{2.3}$$

$$\begin{bmatrix} \acute{P}_x \\ \acute{P}_y \\ \acute{P}_z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} \tag{2.4}$$

สามารถเขียนเป็นสมการได้ดังนี้

$$\acute{P}_x = P_x + d_x \tag{2.5}$$

$$\acute{P}_y = P_y + d_y \tag{2.6}$$

$$\acute{P}_z = P_z + d_z \tag{2.7}$$

**2.4.2 การหมุน (Rotation)**

Rotation หมายถึง การเคลื่อนตำแหน่งทุก ๆ จุดของวัตถุรอบจุดตรึงโดยการหมุนในระบบพิกัดสามมิติจะพิจารณาถึงแกนที่จะใช้ในการหมุนได้แก่ การหมุนในแกน X แกน Y หรือแกน Z โดยเขียนเป็นเมตริกการหมุนได้ดังสมการที่ 2.8 ถึง 2.10 ตามลำดับ

$$R_x(a_x) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(a_x) & -\sin(a_x) \\ 0 & \sin(a_x) & \cos(a_x) \end{bmatrix} \tag{2.8}$$

$$R_y(a_y) = \begin{bmatrix} \cos(a_y) & 0 & \sin(a_y) \\ 0 & 1 & 0 \\ -\sin(a_y) & 0 & \cos(a_y) \end{bmatrix} \tag{2.9}$$

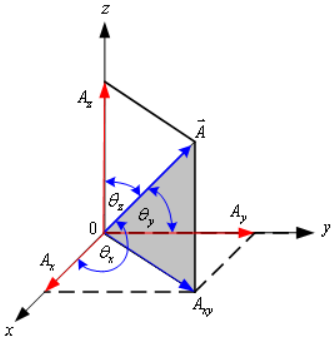
$$R_y(a_y) = \begin{bmatrix} \cos(a_y) & 0 & \sin(a_y) \\ 0 & 1 & 0 \\ -\sin(a_y) & 0 & \cos(a_y) \end{bmatrix} \tag{2.10}$$

เมื่อ $R_x(a_x)$ คือ เมตริกการหมุนรอบ แกน x ด้วยมุม $a_x$

$R_y(a_y)$ คือ เมตริกการหมุนรอบ แกน x ด้วยมุม $a_y$

$R_z(a_z)$ คือ เมตริกการหมุนรอบ แกน z ด้วยมุม $a_z$

การหาทิศทางในระบบพิกัดสามมิตินั้นจะใช้หลักการทางเวกเตอร์ (Vector) โดยองค์ประกอบของเวกเตอร์ในระบบ 3 มิติกำหนดให้เวกเตอร์ $\overline{A}$ อยู่บนระนาบ X Y และ Z โดยเวกเตอร์ $\overline{A}$ ทำมุมกับแกน X แกน Y และแกน Z เป็นมุม $q_x = (\theta_x)$ $q_y = (\theta_y)$ และ $q_z = (\theta_z)$ ตามรูปที่ 2.17




**รูปที่ 2.17 การทำมุมของ $\overline{A}$ กับแกน $x$ $y$ และ $z$**

ที่มา: https://sites.google.com/site/wektexr36/swn-prakxb-khxng-wek-texr

ขนาดของ $\overline{A}_x$ เขียนแทนด้วย $A_x = A \cdot \cos(\theta_x)$ โดยที่ $cos(\theta_x) = \frac{A_x}{A}$
ขนาดของ $\overline{A}_y$ เขียนแทนด้วย $A_y = A \cdot \cos(\theta_y)$ โดยที่ $cos(\theta_y) = \frac{A_y}{A}$
ขนาดของ $\overline{A}_z$ เขียนแทนด้วย $A_z = A \cdot \cos(\theta_z)$ โดยที่ $cos(\theta_z) = \frac{A_z}{A}$
ขนาดของ $\overline{A}$ เท่ากับ $A = \sqrt{A_x^2 + A_y^2 + A_z^2}$
ดังนั้นทิศทางของ $\overline{A}$ คือมุมที่ $\overline{A}$ ทำกับแกน X แกน Y และแกน Z หาโดย
$\theta_x = cos^{-1}\left(\frac{A_x}{A}\right)$ $\theta_y = cos^{-1}\left(\frac{A_y}{A}\right)$ และ $\theta_z = cos^{-1}\left(\frac{A_z}{A}\right)$ เมื่อได้ค่า $\theta_x$ $\theta_y$ และ $\theta_z$ แล้วบันทึกเป็นจุดของอุปกรณ์พร้อมกับตั้งค่าความคลาดเคลื่อนของตำแหน่งอุปกรณ์ไว้โดยเทียบวัดจากขนาดของอุปกรณ์

### 2.4.3 เมตริกการแปลง (Transform)

ในกรณีที่มีการเคลื่อนย้ายกล้อง Kinect ตำแหน่งของอุปกรณ์ที่ได้มีการตั้งค่าไปแล้วนั้นจะต้องมีการคำนวณการย้ายด้วย โดยคำนวณจากจุดกำเนิดของกล้อง ณ ตำแหน่งที่ติดตั้งจะได้ค่า $t_x$ $t_y$ และ $t_z$ โดยเริ่มจากการทำเมตริกการแปลง (Transform) ซึ่งการแปลงเป็นระยะการเคลื่อนจากจุดกำเนิดในแต่ละแกนโดยนำค่า $a_x$ $a_y$ $a_z$ $t_x$ $t_y$ และ $t_z$ ที่วัดได้มาแทนค่าในสมการเพื่อสร้างเมตริกการแปลงตำแหน่งที่เปลี่ยนไปตามสมการที่ 2.11

$$\begin{bmatrix} c(a_z)c(a_y) & -s(a_z)c(a_y) + c(a_z)s(a_y)s(a_x) & s(a_z)s(a_x) + c(a_z)s(a_y)c(a_x) & t_x \\ s(a_z)c(a_y) & c(a_z)c(a_x) + s(a_z)s(a_y)s(a_x) & -c(a_z)s(a_x) + s(a_z)s(a_y)c(a_x) & t_y \\ -s(a_y) & c(a_y)s(a_x) & c(a_y)c(a_x) & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.11)$$

เมื่อ $a_x$ $a_y$ และ $a_x$ คือ ขนาดมุมที่ใช้ในการหมุนในแต่ละแกน

$t_x$ $t_y$ และ $t_z$ คือ ระยะการเคลื่อนจากจุดกำเนิดของในแต่ละแกน

$s(a_x)$ แทน $sin(a_x)$

$c(a_x)$ แทน $cos(a_x)$

$r_{11}$ ถึง $r_{33}$ คือ ค่าใหม่ที่คำนวณได้

นำค่าตำแหน่งของอุปกรณ์ที่ได้ใหม่มาคูณกับเมตริกการแปลง (Transform) ตามที่แสดงในสมการที่ 2.13 จะทำให้ได้ตำแหน่งของอุปกรณ์ที่มีค่าอ้างอิงจากจุดกำเนิดโดยมีค่า X Y และ Z ตามที่แสดงในสมการที่ 2.14 ถึง 2.16 ตามลำดับ

$$\acute{P} = [\text{transform}] \cdot P \tag{2.12}$$

$$\begin{bmatrix} \acute{P}_x \\ \acute{P}_y \\ \acute{P}_z \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} \tag{2.13}$$

$$\acute{P}_x = r_{11}P_x + r_{12}P_y + r_{13}P_z + t_x \tag{2.14}$$

$$\acute{P}_y = r_{21}P_x + r_{22}P_y + r_{23}P_z + t_y \tag{2.15}$$

$$\acute{P}_z = r_{31}P_x + r_{32}P_y + r_{33}P_z + t_z \tag{2.16}$$

เมื่อ $P$ คือ จุดพิกัดของข้อต่อใด ๆ

$\acute{P}$ คือ จุดพิกัดของข้อต่อนั้นหลังปรับระบบพิกัด

ทำการคำนวณมุมของ $\acute{P}_x$ $\acute{P}_y$ และ $\acute{P}_z$ ดังนี้

ขนาดของ $\acute{P}_x$ เขียนแทนด้วย $\acute{P}_x = \acute{P} \cdot \cos(\theta_x)$ โดยที่ $cos(\theta_x) = \frac{\acute{P}_x}{\acute{P}}$

ขนาดของ $\acute{P}_y$ เขียนแทนด้วย $\acute{P}_y = \acute{P} \cdot \cos(\theta_y)$ โดยที่ $\cos(\theta_y) = \frac{\acute{P}_y}{\acute{P}}$

ขนาดของ $\acute{P}_z$ เขียนแทนด้วย $\acute{P}_z = \acute{P} \cdot \cos(\theta_z)$ โดยที่ $cos(\theta_z) = \frac{\acute{P}_z}{\acute{P}}$

ขนาดของ $\overline{\acute{P}}$ เท่ากับ $\acute{P} = \sqrt{\acute{P}_x^2 + \acute{P}_y^2 + \acute{P}_z^2}$

ดังนั้นทิศทางของเวกเตอร์ $\overline{\acute{P}}$ คือ มุมที่ $\overline{\acute{P}}$ ทำมุมกับแกน X Y และ Z หาได้โดย

$\theta_x = cos^{-1}\left(\frac{\acute{P}_x}{\acute{P}}\right)$ $\theta_y = cos^{-1}\left(\frac{\acute{P}_y}{\acute{P}}\right)$ และ $\theta_z = cos^{-1}\left(\frac{\acute{P}_z}{\acute{P}}\right)$
จากนั้นนำไปเทียบกับตำแหน่งของอุปกรณ์หากตรงกันก็จะทำการส่งค่าต่อไปทำงานในส่วนต่อไป

## 2.5 ทฤษฎีจลนศาสตร์ของแขนกล

จลนศาสตร์ หรือที่เรียกว่า Kinematics เป็นการศึกษาการเคลื่อนที่ ที่เกี่ยวข้องกับตำแหน่ง (Position) ความเร็ว (Velocity) และความเร่ง (Accelerate) ของวัตถุ จลนศาสตร์ของแขนกลคือการศึกษาทางด้านเรขาคณิตโดยเฉพาะการเคลื่อนที่ของแขนกลในหุ่นยนต์โดยไม่คิด แรง รูปร่าง ขนาด และน้ำหนัก การศึกษาจะคำนวณหา ตำแหน่ง ความเร็ว และความเร่งของการเคลื่อนไหวของแขนกลผ่านระบบหรือกลไก โดยจลนศาสตร์ที่ใช้สำหรับโครงงานคือจลนศาสตร์แบบไปข้างหน้า

### 2.5.1 จลนศาสตร์แบบไปข้างหน้า (Forward Kinematics)

เป็นการคำนวณหาตำแหน่งส่วนปลายของแขนกล (End Effectors) จากมุมของข้อต่อ (Joint Angle) ของแขนกลที่เคลื่อนที่ไป สำหรับตำแหน่งส่วนปลายของแขนกลจะใช้ระบบพิกัด (Coordinate System) ในการอ้างอิง การวิเคราะห์จลนศาสตร์แบบไปข้างหน้าสำหรับแขนกลที่เคลื่อนที่ในระนาบสามารถทำได้โดยใช้วิธีการทางเรขาคณิตซึ่งง่ายต่อการวิเคราะห์ โดยมองทุกข้อต่อให้อยู่ในระนาบเดียวกันตามรูปที่ 2.12 เมื่อทำการวิเคราะห์หาตำแหน่งส่วนปลายของแขนกลซึ่งก็คือตำแหน่งปลายสุดของข้อต่อที่ 3($l_3$) จากมุมของข้อต่อทั้ง 3 แกน ($\theta_1, \theta_2, \theta_3$) จะได้สมการการเคลื่อนที่แบบจลนศาสตร์แบบไปข้างหน้าดังสมการที่ 2.17 ถึง 2.19 ตามลำดับ

$$x = l_1 \cos(\theta_1) + l_2 \cos(\theta_1+\theta_2) + l_3 \cos(\theta_1 + \theta_2 + \theta_3) \quad (2.17)$$

$$y = l_1 \sin(\theta_1) + l_2 \sin(\theta_1+\theta_2) + l_3 \sin(\theta_1 + \theta_2 + \theta_3) \quad (2.18)$$

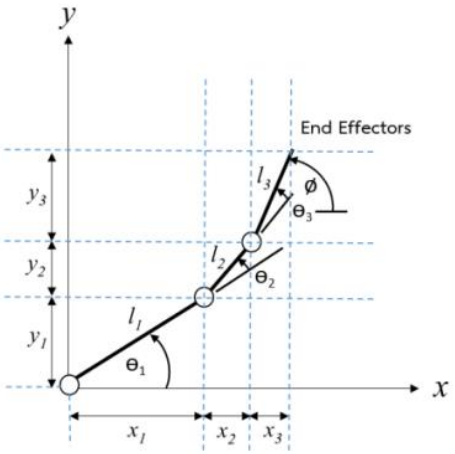
$$\emptyset = \theta_1 + \theta_2 + \theta_3 \quad (2.19)$$

เมื่อ $x$ คือ ระยะของตำแหน่งส่วนปลายของแขนกลในแนวแกน $x$
$y$ คือ ระยะของตำแหน่งส่วนปลายของแขนกลในแนวแกน $y$
$l_1$ คือ ความยาวของก้านต่อที่ 1
$l_2$ คือ ความยาวของก้านต่อที่ 2
$l_3$ คือ ความยาวของก้านต่อที่ 3
$\theta_1$ คือ มุมการเคลื่อนที่ของก้านต่อที่ 1 อ้าวอิงกับแกน $x$

$\theta_2$ คือ มุมการเคลื่อนที่ของก้านต่อที่ 2 อ้าวอิงกับก้าน 1

$\theta_3$ คือ มุมการเคลื่อนที่ของก้านต่อที่ 3 อ้าวอิงกับก้าน 2

$\emptyset$ คือ มุมตำแหน่งส่วนปลายของแขนกลอ้างอิงแกน $x$

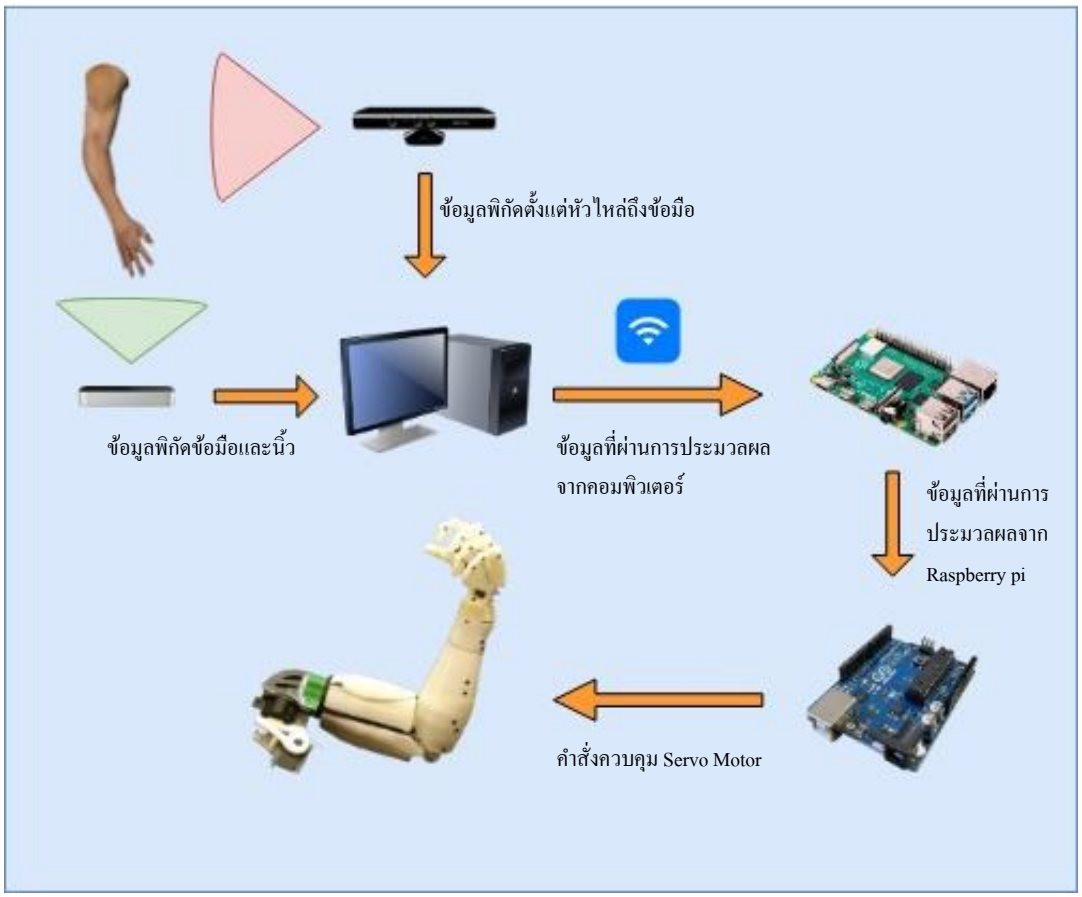



**รูปที่ 2.18 ตัวอย่างแขนกล 3 แกน 3 ก้านที่เคลื่อนที่ในระนาบ $x - y$**

# บทที่ 3
# การออกแบบและพัฒนาระบบ

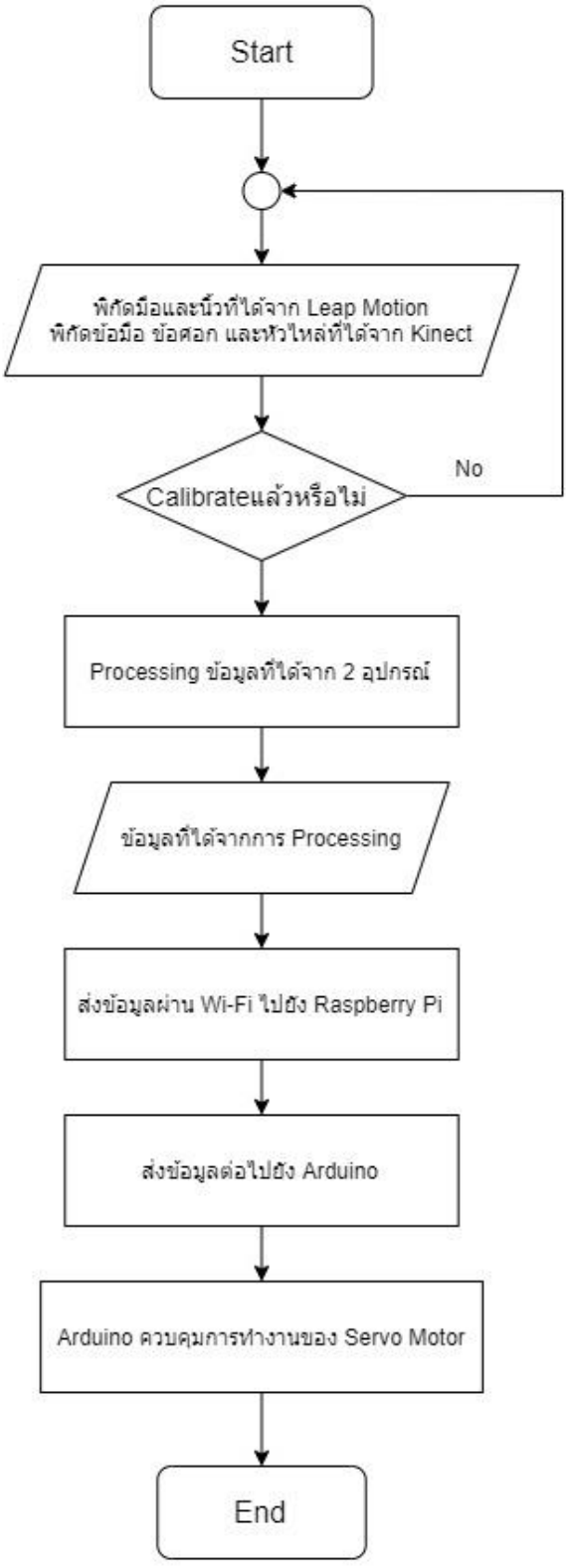
## 3.1 ภาพรวมของระบบ

ระบบควบคุมแขนกลด้วยการติดตามโครงกระดูกและท่าทางมือ ต้องมีการปรับเทียบการหมุนของ Servo Motor ด้วยท่าทางของมือและแขนจากผู้ใช้งานจริง จากนั้นเมื่อเริ่มการทำงานระบบจะทำการรับค่าพิกัดที่ได้จากอุปกรณ์ Leap Motion และ Kinect นำค่าที่รับมาจากอุปกรณ์ทั้ง 2 มาทำการประมวลผลและส่งค่า ผ่าน Wi-Fi จากทางเครื่องคอมพิวเตอร์ไปยัง Raspberry Pi แล้ว Arduino จะรับค่าต่อจาก Raspberry Pi ผ่านทาง Serial และสุดท้าย Arduino จะทำการสั่งให้ Servo Motor ทำงานตามที่ได้ทำการคำนวณไว้ในขั้นตอนการปรับเทียบ



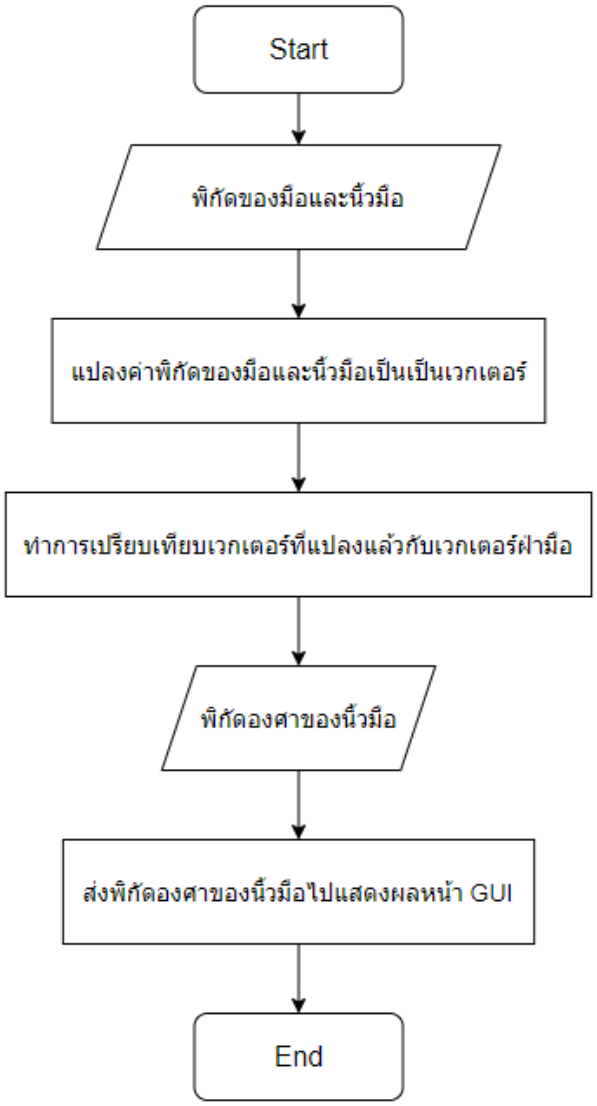

**รูปที่ 3.1 ภาพรวมการทำงานของระบบ**

## 3.2 ผังงานโดยรวมของระบบ (Flow Chart)




**รูปที่ 3.2 ผังงานโดยรวมของระบบ**

## 3.3 การประมวลผลของ Leap Motion

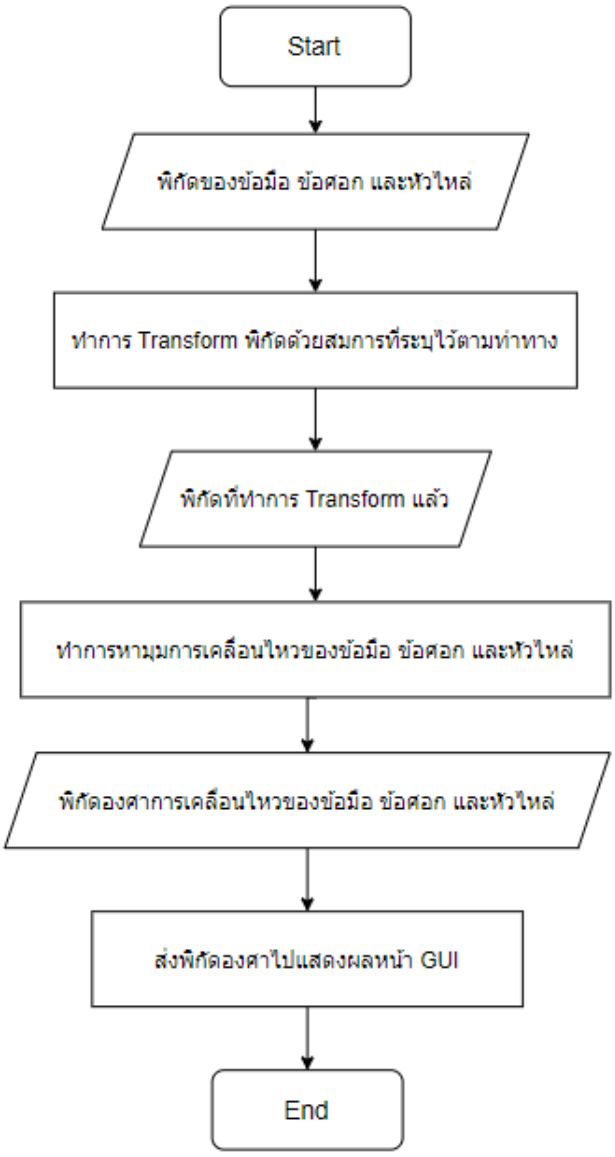



**รูปที่ 3.3 ขั้นตอนการประมวลผลของ Leap Motion**

Leap Motion เป็นอุปกรณ์ที่มีเซนเซอร์ที่มีความสามารถในตรวจจับได้ทุกข้อนิ้วและข้อมือด้วยอัตราการส่งข้อมูลที่สูงถึง 120 fps และมีความแม่นยำในระดับ 0.01 มิลลิเมตร โดยขั้นตอนการประมวลผลเริ่มจากการตรวจจับมือเพื่อหาพิกัดของมือและนิ้วมือ จากนั้นจะนำพิกัดที่ได้จากการตรวจจับมาทำการแปลงค่าพิกัดต่าง ๆ ให้เป็นเวกเตอร์และนำเวกเตอร์ที่แปลงแล้วมาเปรียบเทียบกับเวกเตอร์ฝ่ามือเพื่อหาค่าพิกัดองศาการเคลื่อนไหวของมือและนิ้วมือหลังจากที่ได้ค่าพิกัดองศาเรียบร้อยแล้วจะทำการส่งค่าที่ได้ไปแสดงผลที่หน้า GUI และนำค่าพิกัดนั้นส่งไปยัง Raspberry Pi

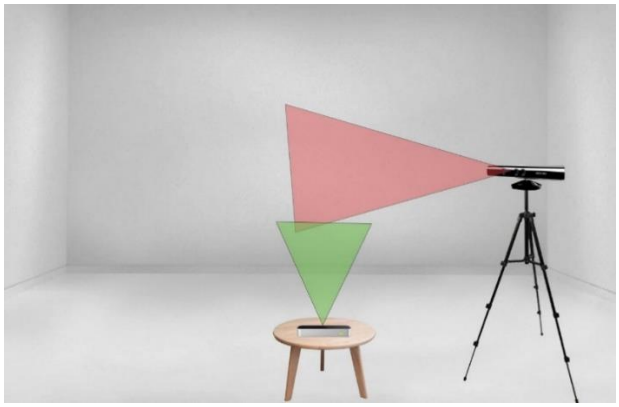
เพื่อที่จะทำการส่งต่อไปที่ Arduino ให้ควบคุมการทำงานของ Servo Motor อุปกรณ์นี้ก็ยังมีข้อจำกัดในการทำงานหลาย ๆ อย่างอาทิเช่น ระยะในการตรวจจับจะอยู่ที่ประมาณ 1 ฟุตบริเวณรอบ ๆ ผู้ควบคุมและถ้าวาง Leap Motion ให้ตัวเซนเซอร์โดนแสงลงพอดีจะเกิดปัญหาในการตรวจจับไม่พบบ้างในบางกรณี

## 3.4 การประมวลของ Kinect



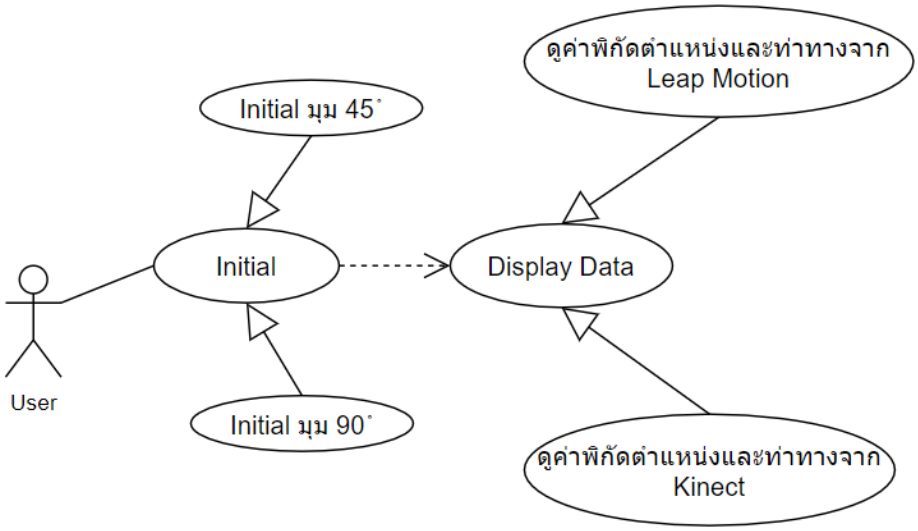

**รูปที่ 3.4 ขั้นตอนการประมวลผลของ Kinect**

กล้อง Kinect ทำการ Skeleton Tracking จะทำให้ได้ค่าพิกัด X Y และ Z ของแต่ละจุดใน 20 จุดมาซึ่งในตัวโครงงานนี้จะใช้ค่าพิกัดจากกล้อง Kinect ในการตรวจจับแค่บริเวณหัวไหล่ไปถึงข้อมือเป็นจำนวน 3 จุดด้วยกัน แต่ค่าพิกัดที่ได้มานี้ยังไม่สามารถนำมาใช้งานได้เนื่องจากตำแหน่งที่วางอุปกรณ์ Kinect ทำมุมฉากกับผู้ควบคุมยังไม่ครอบคลุมท่าทางการเคลื่อนไหวมากพอ ดังนั้นจึงต้องทำการหาตำแหน่งการวาง Kinect ให้ครอบคลุมมากขึ้น สาเหตุที่ต้องทำการ Transform ค่าพิกัดจาก 3 จุดเพื่อให้ได้ค่าพิกัดที่แท้จริง และนำค่าพิกัดที่แท้จริงมาทำการคำนวณหามุมในการเคลื่อนไหว หลังจากที่ได้มุมในการเคลื่อนไหวแล้วจะทำการส่งค่าที่ได้ไปแสดงผลที่หน้า GUI และนำค่าพิกัดนั้นส่งไปยัง Raspberry Pi เพื่อที่จะทำการส่งต่อไปที่ Arduino ให้ควบคุมการทำงานของ Servo Motor



**รูปที่ 3.5 ตำแหน่งและทิศทางการตรวจจับของ Leap Motion และ Kinect**

## 3.5 แผนภาพยูสเคส (Use Case Diagram)




**รูปที่ 3.6 แผนภาพยูสเคส**

จากแผนภาพยูสเคสสามารถอธิบายได้ดังนี้

1) ผู้ใช้งานสามารถ Initial ในมุม 45 องศา
2) ผู้ใช้งานสามารถ Initial ในมุม 90 องศา
3) ผู้ใช้งานสามารถดูค่าพิกัดตำแหน่งและท่าทางจาก Leap Motion
4) ผู้ใช้งานสามารถดูค่าพิกัดตำแหน่งและท่าทางจาก Kinect

**ตารางที่ 3.1 รายละเอียดของ Use Case Initial มุม 45 องศา**

| | |
|---|---|
| Use Case Name | Initial มุม 45 องศา |
| Actor | ผู้ใช้งาน |
| Description | ส่วนของผู้ใช้งานทำการบันทึกค่าจากการ Initial มุม 45 องศา |
| Pre-Condition | - |
| Post-Condition | ทำการบันทึก Initial มุม 45 องศาสำเร็จ |

**ตารางที่ 3.2 รายละเอียดของ Use Case Initial มุม 90 องศา**

| | |
|---|---|
| Use Case Name | Initial มุม 90 องศา |
| Actor | ผู้ใช้งาน |
| Description | ส่วนของผู้ใช้งานทำการบันทึกค่าจากการ Initial มุม 90 องศา |
| Pre-Condition | - |
| Post-Condition | ทำการบันทึก Initial มุม 90 องศาสำเร็จ |

**ตารางที่ 3.3 รายละเอียดของ Use Case ดูค่าพิกัดตำแหน่งและท่าทางจาก Leap Motion**

| | |
|---|---|
| Use Case Name | ดูค่าพิกัดตำแหน่งและท่าทางจาก Leap Motion |
| Actor | ผู้ใช้งาน |
| Description | ส่วนของผู้ใช้งานดูค่าพิกัดตำแหน่งและท่าทางจาก Leap Motion |
| Pre-Condition | - |
| Post-Condition | สามารถดูค่าพิกัดตำแหน่งและท่าทางจาก Leap Motion ได้สำเร็จ |

**ตารางที่ 3.4 รายละเอียดของ Use Case ดูค่าพิกัดตำแหน่งและท่าทางจาก Kinect**

| | |
|---|---|
| Use Case Name | ดูค่าพิกัดตำแหน่งและท่าทางจาก Kinect |
| Actor | ผู้ใช้งาน |
| Description | ส่วนของผู้ใช้งานดูค่าพิกัดตำแหน่งและท่าทางจาก Kinect |
| Pre-Condition | ทำการบันทึก Initial มุม 45 องศา และมุม 90 องศา |
| Post-Condition | สามารถดูค่าพิกัดตำแหน่งและท่าทางจาก Kinect ได้สำเร็จ |

## 3.6 การออกแบบ Graphical User Interface (GUI)

ส่วนของ GUI ออกแบบมาเพื่อให้ผู้ใช้งานสามารถดูการแสดงค่าพิกัดตำแหน่งและท่าทางจากอุปกรณ์ Leap Motion และ Kinect ซึ่งผู้ใช้งานสามารถควบคุมการทำงานของทั้ง 2 อุปกรณ์ผ่านตัวโปรแกรมได้อีกด้วย

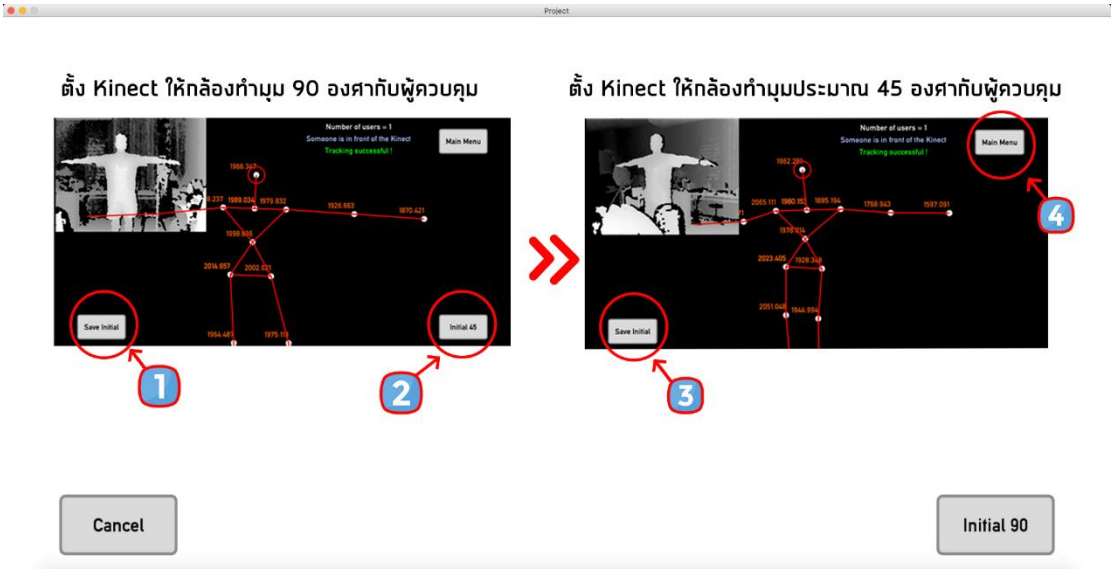
### 3.6.1 หน้าแสดงเมนูหลักในการใช้งาน (Main Menu)

ส่วนของหน้าแสดงเมนูหลักในการใช้งานจะเป็นหน้าแรกที่แสดงหลังจากการเข้าโปรแกรม ซึ่งภายในหน้านี้จะแสดงปุ่มสองปุ่มประกอบด้วยปุ่ม Start และปุ่ม Initial โดยที่ถ้ากดปุ่ม Start จะปลี่ยนเป็นหน้าแสดงผลค่าพิกัดตำแหน่งและท่าทาง ส่วนถ้ากดปุ่ม Initial จะเปลี่ยนเป็นหน้าแนะนำการทำ Initial

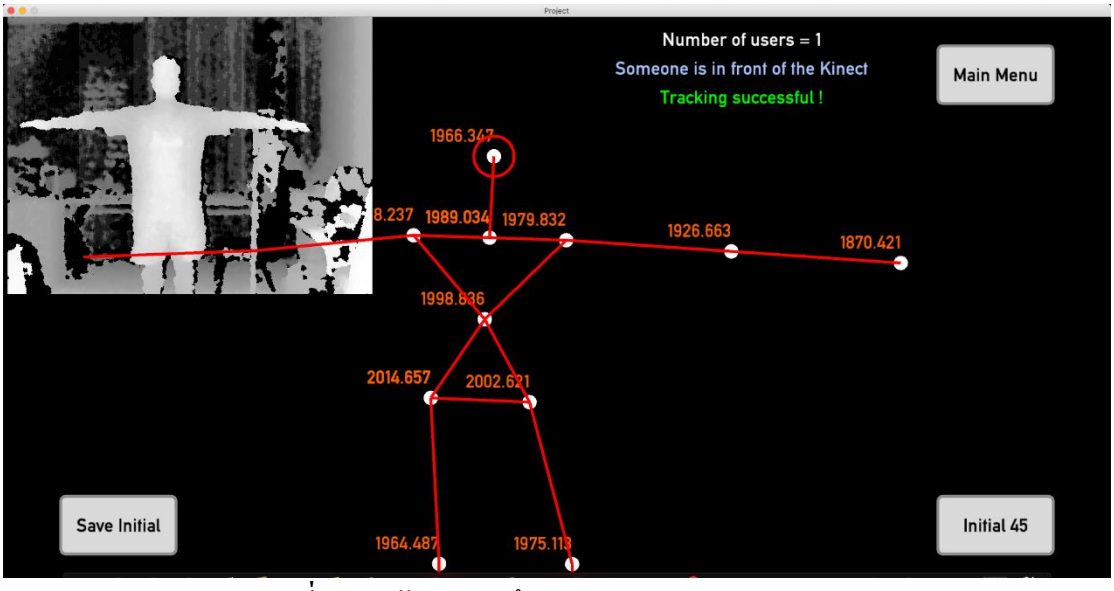

**รูปที่ 3.7 หน้าแสดงเมนูหลักในการใช้งาน**

### 3.6.2 หน้าการปรับค่าเริ่มต้น (Initial)

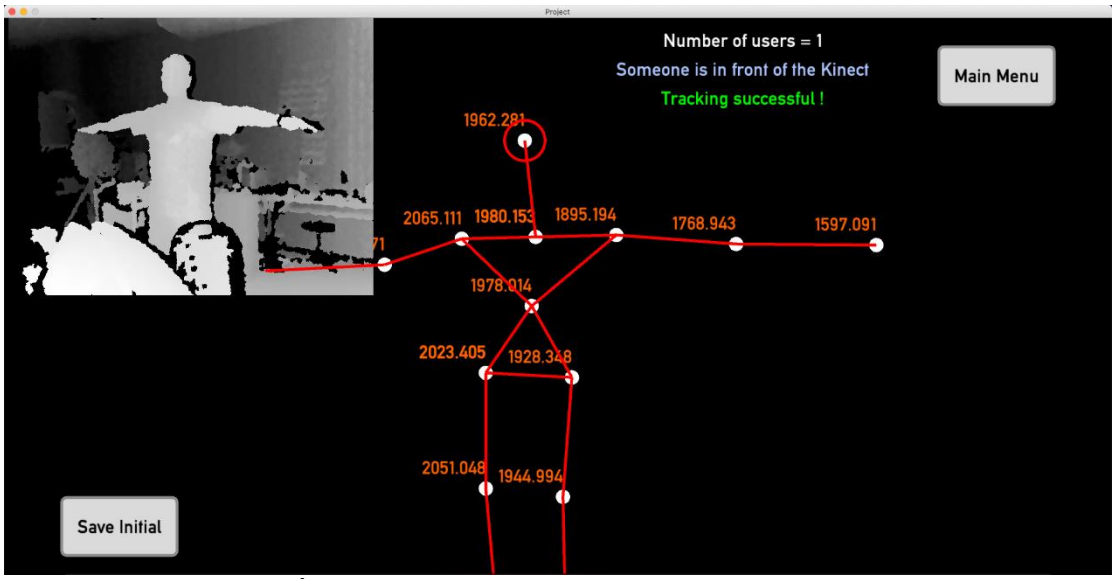
ส่วนของการปรับค่าเริ่มต้นจะเป็นส่วนที่ผู้ใช้งานจะต้องทำก่อนที่จะทำการดูค่าพิกัดตำแหน่งและท่าทางจากอุปกรณ์ Leap Motion และ Kinect เนื่องจากบางท่าทางในการเคลื่อนไหวของผู้ควบคุมมุมของกล้อง Kinect ที่วางทำมุม 90 องศากับผู้ควบคุมไม่สามารถตรวจจับได้หมด ดังนั้นจึงได้ให้ทำการบันทึกค่าพิกัดในการวางกล้อง Kinect มุม 90 องศาและมุม 45 องศาโดยสาเหตุที่เลือกมุม 45 องศาเนื่องจากสามารถตรวจจับได้ครอบคลุมมากขึ้น เมื่อผู้ใช้งานทำการ Initial เรียบร้อยแล้วให้กลับไปที่หน้าแสดงเมนูหลักในการใช้งาน




**รูปที่ 3.8 หน้าแนะนำขั้นตอนการทำ Initial**



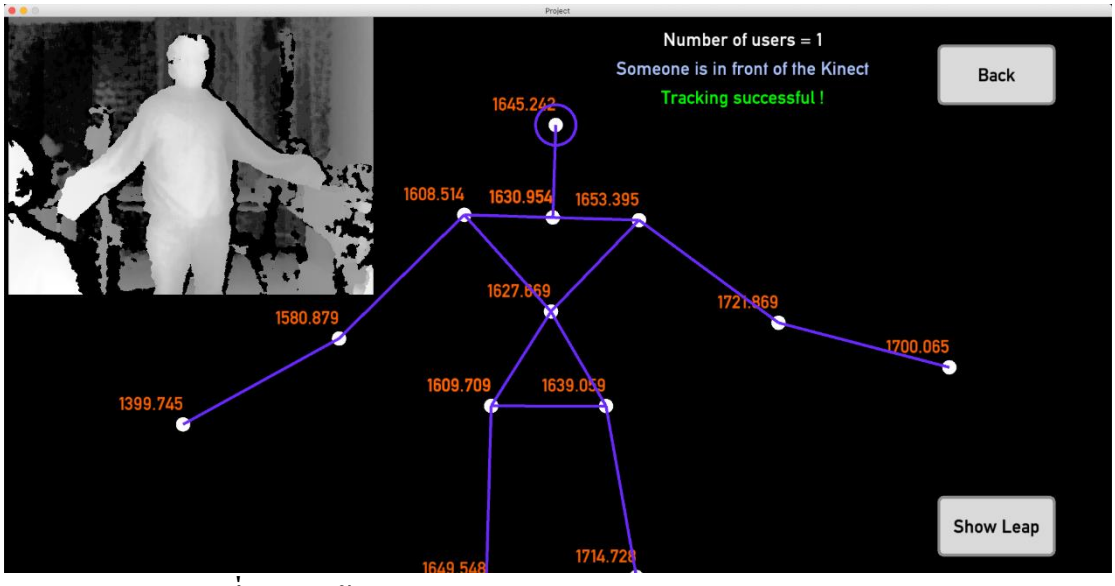

**รูปที่ 3.9 หน้าการบันทึกค่า Initial มุม 90 องศา**



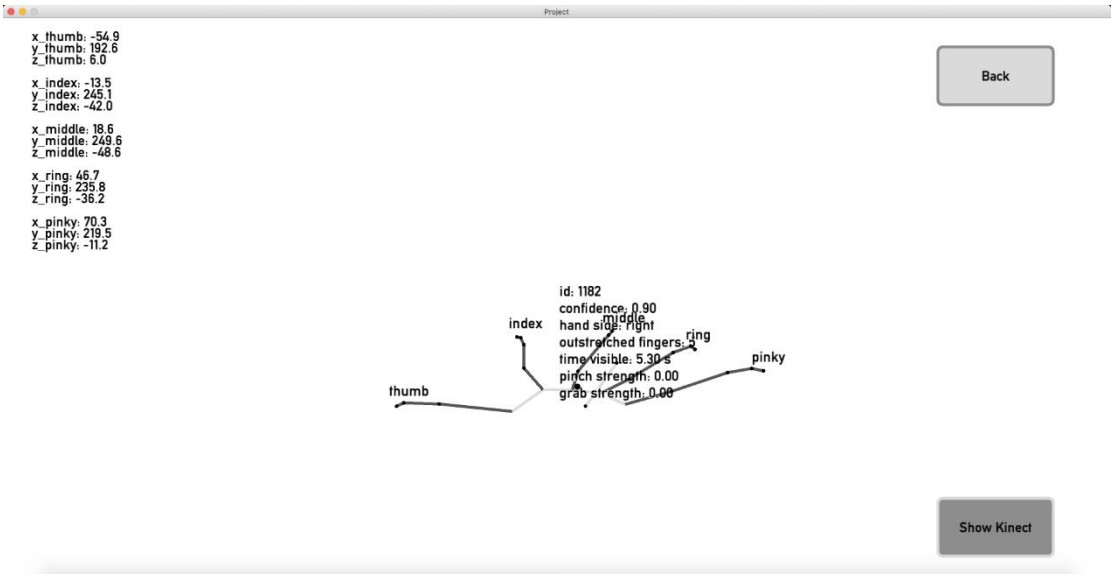

**รูปที่ 3.10 หน้าการบันทึกค่า Initial มุม 45 องศา**

### 3.6.3 หน้าการแสดงค่าพิกัดตำแหน่งและท่าทาง

ส่วนของหน้าการแสดงค่าพิกัดตำแหน่งและท่าทางจะแสดงจากสองอุปกรณ์คือ Kinect และ Leap Motion ซึ่งการกดปุ่ม Start จากหน้าแสดงเมนูหลักในการใช้งานจะถือว่าเป็นการสั่งให้อุปกรณ์ทั้งสองเริ่มทำการส่งข้อมูลหลังจากกดแล้วหน้าก็จะเปลี่ยนไปแสดงค่าพิกัดตำแหน่งและท่าทางจากอุปกรณ์ Kinect โดยถ้าจะเปลี่ยนไปดูของอุปกรณ์ Leap Motion ก็ทำได้เช่นกันและถ้าหากกดปุ่ม Back อุปกรณ์ทั้งสองก็จะหยุดการส่งข้อมูลและหน้าแสดงผลก็จะเปลี่ยนเป็นหน้าแสดงเมนูหลักในการใช้งาน



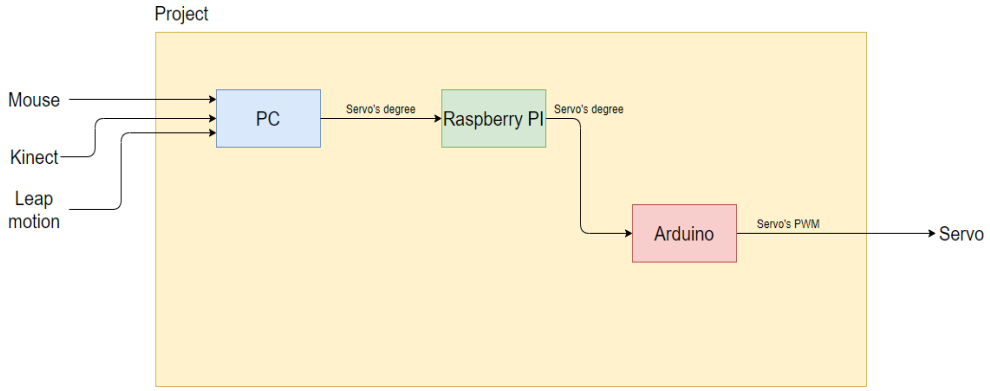

**รูปที่ 3.11 หน้าแสดงค่าพิกัดตำแหน่งและท่าทางจาก Kinect**




**รูปที่ 3.12 หน้าแสดงค่าพิกัดตำแหน่งและท่าทางจาก Leap Motion**

## 3.7 แผนภาพการออกแบบส่วนฮาร์ดแวร์ (Top-down Design)

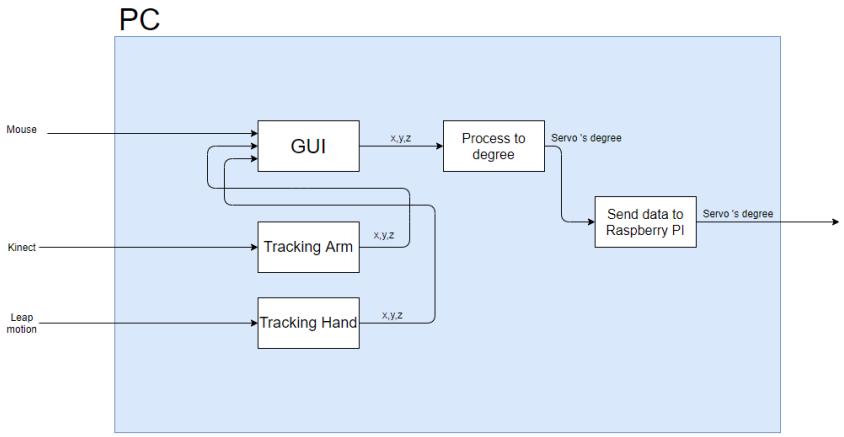
### 3.7.1 Top Level




**รูปที่ 3.13 ภาพรวมการออกแบบส่วนฮาร์ดแวร์**

โดยระบบนี้จะใช้ PC เป็นอุปกรณ์หลักของระบบโดยจะมีหน้าที่ประมวลผลข้อมูลที่ได้รับเข้ามาแล้วส่งข้อมูลที่ได้ผ่านการประมวลผลแล้วไปยัง Raspberry Pi ผ่านทางเครือข่ายอินเตอร์เน็ต Wi-Fi โดย Raspberry Pi จะรับค่าและส่งค่าไปยัง Arduino จากนั้น Arduino จะทำการสั่งการควบคุม Servo Motor ให้ทำการหมุนตามพิกัดที่ได้รับค่ามา โดยแหล่งจ่ายไฟจะประกอบด้วย 2 อย่าง Power Supply 12 V และ แบตเตอรี่ 12 V ใช้ในกรณีที่ไม่สามารถใช้งาน Power Supply ได้

### 3.7.2 PC




**รูปที่ 3.14 ภาพรวมส่วน PC**

ส่วน Input ประกอบไปด้วย 3 ส่วน

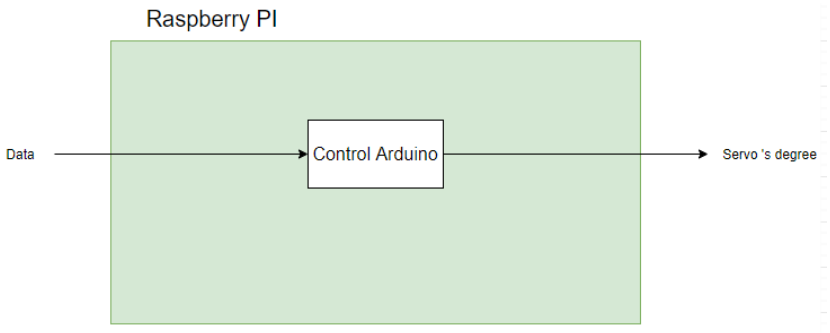
1) Mouse
2) Kinect
3) Leap Motion

ส่วน Output ประกอบด้วย 1 ส่วน

1) Servo’s degree

โดยการทำงานของส่วน PC จะเริ่มจากการรับค่าจากInputจากอุปกรณ์ Kinect จะส่งค่าพิกัดข้อมือ ข้อศอก และหัวไหล่ ส่วนอุปกรณ์ Leap Motion จะส่งค่าพิกัดมือและนิ้วมือ จากนั้นข้อมูลทั้ง 2 จะไปรวมกันแสดงผลที่หน้า GUI จากนั้นนำพิกัดไปทำการประมวลผลเพื่อหาค่าองศาการหมุนของ Servo Motor เมื่อได้ค่าเรียบร้อย PC ทำการส่งค่าองศาการหมุนของ Servo Motor ผ่านเครือข่ายอินเตอร์เน็ต Wi-Fi ไปยัง Raspberry Pi

### 3.7.3 Raspberry Pi




**รูปที่ 3.15 ภาพรวมส่วน Raspberry Pi**

ส่วน Input ประกอบด้วย 1 ส่วน

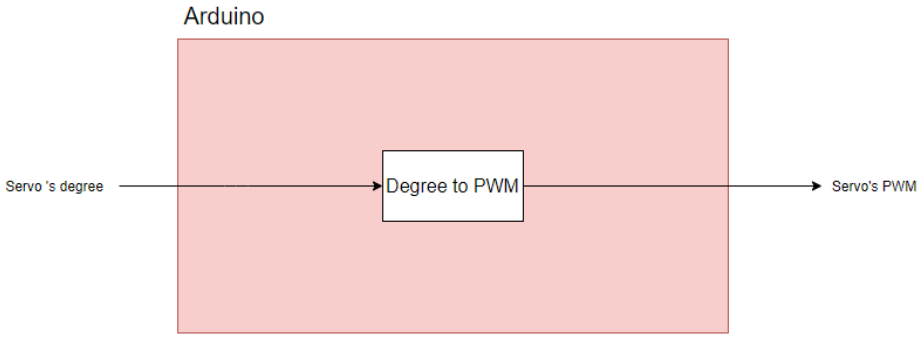
1) ชุดข้อมูลค่าองศาการหมุนของ Servo Motor ที่ได้มาจากการประมวลผล

ส่วน Output ประกอบด้วย 1 ส่วน

1) ข้อมูลค่าองศาการหมุนของ Servo Motor แต่ละตำแหน่งส่วน

โดยการทำงานในส่วนของ Raspberry Pi จะทำการรับชุดข้อมูลที่ทำการประมวลจาก PC ผ่านทางเครือข่ายอินเตอร์เน็ต Wi-Fi จากนั้นจะทำการส่งข้อมูลพิกัดแบ่งออกเป็นส่วน ๆ เพื่อจะส่งต่อให้ Arduino ผ่านทางสาย

### 3.7.4 Arduino




**รูปที่ 3.16 ภาพรวมส่วน Arduino**

ส่วน Input ประกอบด้วย 1 ส่วน

1) ข้อมูลค่าองศาการหมุนของ Servo Motor แต่ละตำแหน่งส่วน

ส่วน Output ประกอบด้วย 1 ส่วน

1) ข้อมูลค่า PWM

โดยภาพรวมการทำงานของ Arduino ทำหน้าที่ส่งข้อมูลค่า PWM ไปสั่งการควบคุมการทำงานของ Servo Motor ซึ่งภายในระบบมีการใช้งาน Servo Motor ทั้งหมด 10 ตัว ดังนั้นจึงใช้ Arduino Mega ในการส่งค่า PWM ควบคุม Servo Motor ทั้ง 10 ตัว

# บทที่ 4
# การใช้งานและผลการทดลอง

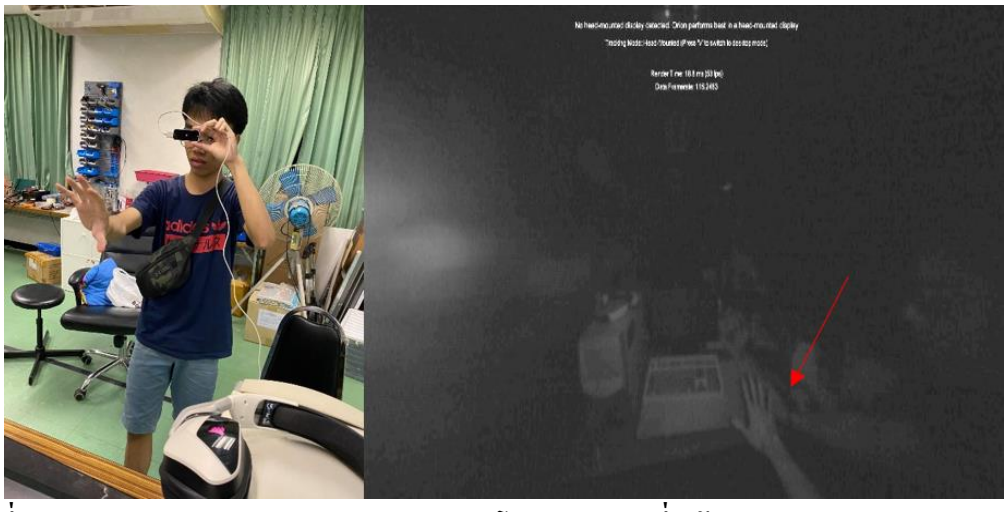
## 4.1 การทดลอง Leap Motion ในส่วนของการหาตำแหน่งการใช้งาน

### 4.1.1 จุดประสงค์การทดลอง

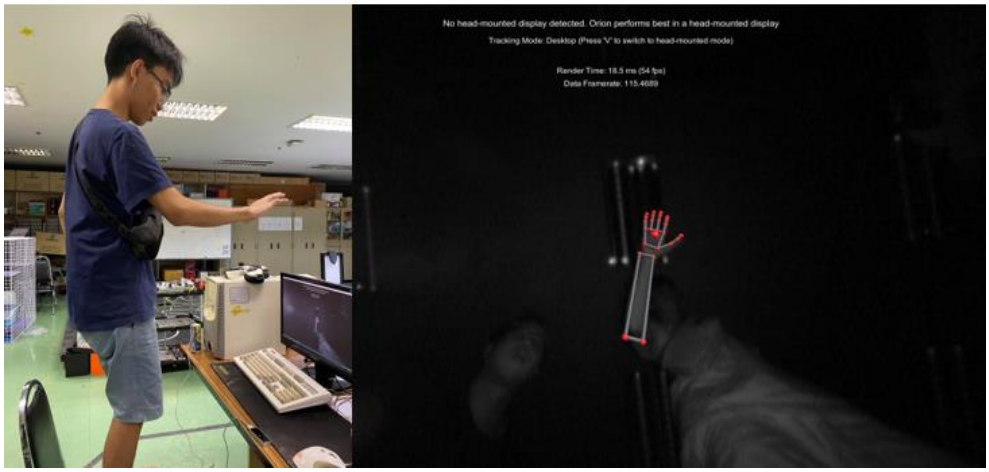
เพื่อหาข้อจำกัดของ Leap Motion ว่าสามารถใช้งานได้ในสถานการณ์ต่าง ๆ ที่เกี่ยวข้องกับโครงงานได้หรือไม่เนื่องจากในบางท่าทางที่เคลื่อนไหวของมือและนิ้ว Leap Motion อาจจะไม่สามารถตรวจจับได้

### 4.1.2 วิธีการทดลอง

ทำการเคลื่อนย้าย Leap Motion ไปตำแหน่งต่าง ๆที่คิดว่าน่าจะเหมาะสมและสามารถที่จะทำการตรวจจับในส่วนของมือและนิ้วมือได้ครอบคลุม โดยดูจากพิกัดจุดต่าง ๆ ของมือและนิ้วที่แสดงเมื่อทำท่าทางต่าง ๆ ว่าตำแหน่งของ Leap Motion จุดไหนที่แสดงพิกัดได้ครบสมบูรณ์ที่สุด



**รูปที่ 4.1 การทดลองหาตำแหน่ง Leap Motion โดยทำการติดที่หน้าผากและผลการตรวจจับ**




**รูปที่ 4.2 การทดลองหาตำแหน่ง Leap Motion โดยให้ทำการสั่งงานระยะไกลและผลการตรวจจับ**

### 4.1.3 ผลการทดลอง

จากการทดลองได้มีการทดลองติดอุปกรณ์ตามส่วนต่าง ๆ บนร่างกายติดโดยเริ่มจากทดลองติดอุปกรณ์ไว้ที่บริเวณหน้าผากแล้วทำการทำการหันตามการเคลื่อนของมือพบว่าได้ว่ามีจุดที่อุปกรณ์ไม่สามารถตรวจจับได้เหมือนดังรูปที่ 4.1 จากนั้นได้ทดลองเปลี่ยนตำแหน่ง Leap Motion โดยจะทำอุปกรณ์ยึด Leap Motion ให้เข้ามาใกล้กับมือและนิ้วตามรูปที่ 4.2 ตำแหน่งที่เหมาะสมกับการใช้งานของโครงงานนี้คือวาง Leap Motion บนโต๊ะให้อยู่ต่ำกว่ามือที่ใช้ในการควบคุมประมาณ 30 เซนติเมตรตามรูปที่ 4.2 จะเป็นตำแหน่งที่เหมาะสมที่สุดในการวางอุปกรณ์ซึ่งสามารถตรวจจับการทำท่าทางได้ครอบคลุมที่สุด

**ตารางที่ 4.1 ผลการทดลองหาตำแหน่งของ Leap Motion**

| อุปกรณ์ | ตำแหน่งของอุปกรณ์ | ตำแหน่งมือที่องศา | | | % ตรวจจับ |
|---|---|---|---|---|---|
| | | 45-135 | 0-45 | 315-0 | |
| Leap Motion | หน้าผาก | ตรวจจับได้ | ตรวจจับไม่ได้ | ตรวจจับไม่ได้ | 33.33 |
| | ใต้มือห่างจากมือตั้งแต่ 30 cm | ตรวจจับได้ | ตรวจจับได้ | ตรวจจับได้ | 100 |

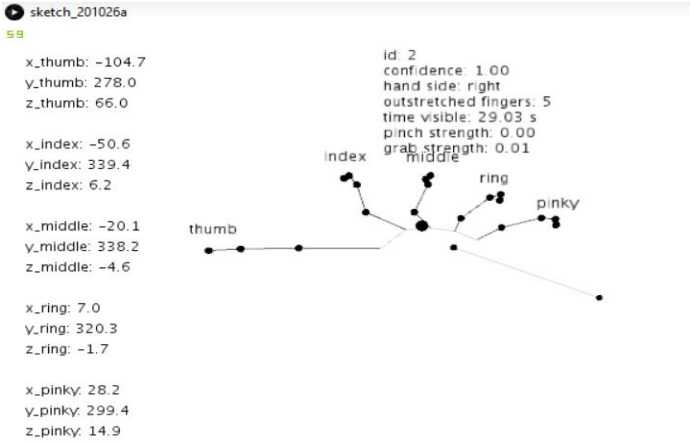
## 4.2 การทดลอง Leap Motion ในส่วนของการหาค่าพิกัด

### 4.2.1 จุดประสงค์การทดลอง

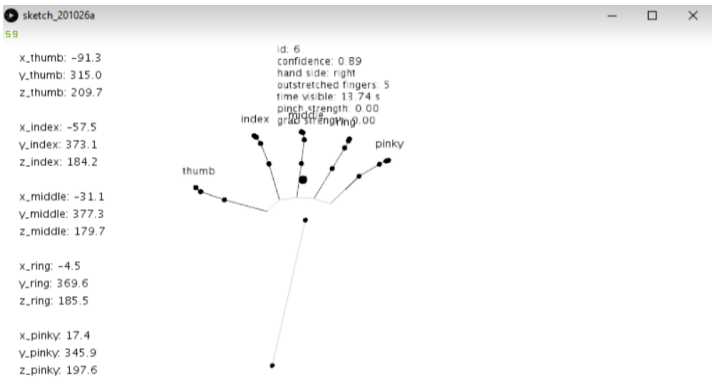
เพื่อนำค่าพิกัดที่ได้จากการตรวจจับของ Leap Motion ที่บริเวณมือและนิ้วมือมาทำการแปลงค่าพิกัดที่รับมาแล้วส่งค่าไปประมวลผลและส่งค่าจากการประมวลผลไปสั่งการควบคุมการทำงานของ Servo Motor

### 4.2.2 วิธีการทดลอง

ในการหาค่าพิกัดจุดต่าง ๆ ของ Leap Motion จะมี Leap Motion API ที่คอยช่วยในเรื่องการคำนวณหาค่าพิกัดต่าง ๆ มาใช้ได้โดยใช้โปรแกรม Leap Motion for Processing v2.3.1.6 จะทำให้ได้พิกัด X Y และ Z ในแต่ละจุด




**รูปที่ 4.3 ค่าพิกัดจาก Leap Motion ของปลายนิ้ววางอุปกรณ์ระยะห่าง 30 เซนติเมตรจากมือ**




**รูปที่ 4.4 ค่าพิกัดจาก Leap Motion ของปลายนิ้วโดยติดอุปกรณ์บริเวณหน้าผาก**

### 4.2.3 ผลการทดลอง

จากการทดลองใช้โปรแกรม Leap Motion for Processing v2.3.1.6 ได้ผลการทดลองตามรูปที่ 4.3 และ 4.4 โดยนำค่าพิกัดตามจุดต่าง ๆ บริเวณปลายนิ้วมือไปใช้ในการประมวลผลต่อ

**ตารางที่ 4.2 ผลการทดลองหาค่าพิกัดโดย Leap Motion**

| อุปกรณ์ | ตำแหน่งการวางอุปกรณ์ | นิ้ว | ค่าพิกัด x y และ z บริเวณปลายนิ้ว |
|---|---|---|---|
| Leap Motion | หน้าผาก | นิ้วโป้ง | -104.7, 278.0, 66.0 |
| | | นิ้วชี้ | -56.6, 339.4, 6.2 |
| | | นิ้วกลาง | -20.1, 338.2, -4.6 |
| | | นิ้วนาง | 7.0, 320.3, -1.7 |
| | | นิ้วก้อย | 28.2, 299.4, 14.9 |
| | ใต้มือห่างจากมือตั้งแต่ 30 cm | นิ้วโป้ง | -91.3, 315.0, 209.7 |
| | | นิ้วชี้ | -57.5, 373.1, 184.2 |
| | | นิ้วกลาง | -31.1, 377.3, 179.7 |
| | | นิ้วนาง | -4.5, 369.6, 185.5 |
| | | นิ้วก้อย | 14.7, 345.9, 197.6 |

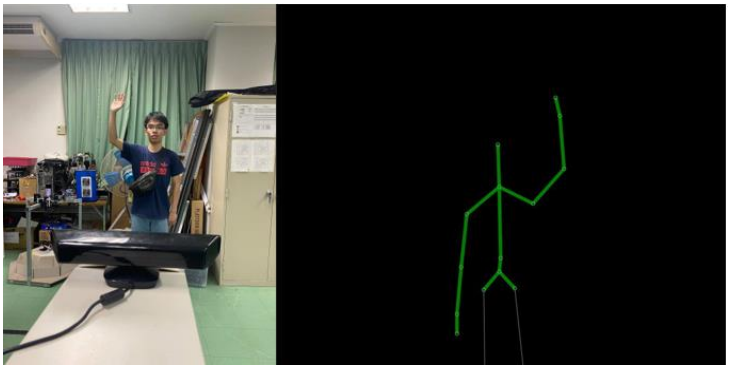
## 4.3 การทดลอง Kinect ในส่วนของการหาตำแหน่งการใช้งาน

### 4.3.1 จุดประสงค์การทดลอง

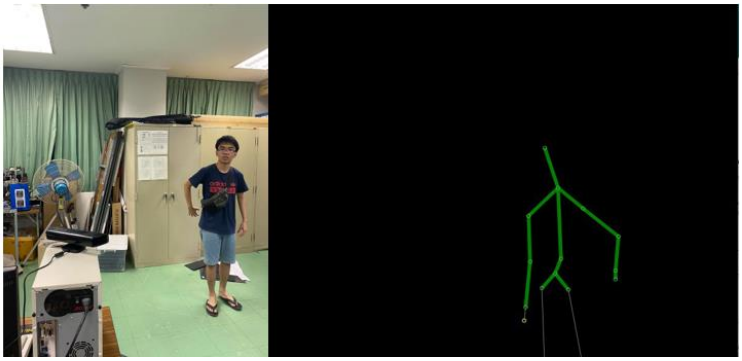
เพื่อหาจุดที่เหมาะสมในการวาง Kinect เนื่องจากการใช้งาน Kinect จะใช้ในการตรวจจับบริเวณหัวไหล่ถึงข้อมือ เนื่องจากการตรวจจับท่าทางของ Kinect ในบางท่าทางการเคลื่อนไหวไม่สามารถตรวจจับได้สมบูรณ์

### 4.3.2 วิธีการทดลอง

เคลื่อนย้าย Kinect พร้อมกับเปลี่ยนท่าทางในการทดลองเพื่อหาจุดที่สามารถตรวจจับท่าทางได้ครอบคลุมมากที่สุดโดยดูจากภาพที่แสดง Skeleton Tracking



**รูปที่ 4.5 การทดลองมุมกล้อง Kinect ทำมุมตรงกับผู้ควบคุมและผลการตรวจจับ**



**รูปที่ 4.6 การทดลองมุมกล้อง Kinect ทำมุมประมาณ 45 องศากับผู้ควบคุมและผลการตรวจจับ**

### 4.3.3 ผลการทดลอง

จากรูปที่ 4.5 ได้ทำการวางกล้อง Kinect ทำมุมตรงกับผู้ควบคุมซึ่งก็สามารถแสดงตรวจจับได้แต่จะมีบางท่าทางที่ไม่สามารถตรวจจับได้เนื่องจากมุมในการตรวจจับไม่กว้างพอ จากนั้นได้ทดลองปรับมุมกล้อง Kinect ตามรูปที่ 4.6 คือทำมุมประมาณ 45 องศากับผู้ควบคุมผล

การทดลองออกมาค่อนข้างดี เนื่องจากสามารถตรวจจับท่าทางการเคลื่อนไหวได้ครอบคลุมมากกว่าแบบที่ทำมุมตรงกับวัตถุ

**ตารางที่ 4.3 ผลการทดลองหาตำแหน่งของ Kinect**

| อุปกรณ์ | ตำแหน่งของอุปกรณ์ | ตำแหน่งมือที่องศา | | | % ตรวจจับ |
|---|---|---|---|---|---|
| | | 45-135 | 0-45 | 315-0 | |
| Kinect | 90 องศา | ตรวจจับได้ | ตรวจจับได้ | ตรวจจับไม่ได้ | 66.67 |
| | 45 องศา | ตรวจจับได้ | ตรวจจับได้ | ตรวจจับได้ | 100 |

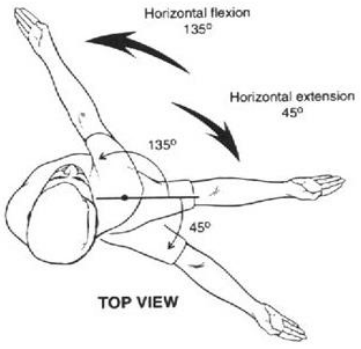
## 4.4 การทดลอง Kinect ในส่วนของการหาค่าพิกัด

### 4.4.1 จุดประสงค์การทดลอง

เพื่อนำค่าพิกัดที่อ่านได้จากตัว Kinect ตามจุดตั้งแต่หัวไหล่ไปถึงข้อมือไปทำการแปลงค่าให้ค่าพิกัดนั้นสามารถไปทำการประมวลผลและนำค่าที่ได้จากการประมวลผลไปทำการสั่งการควบคุม Servo Motor

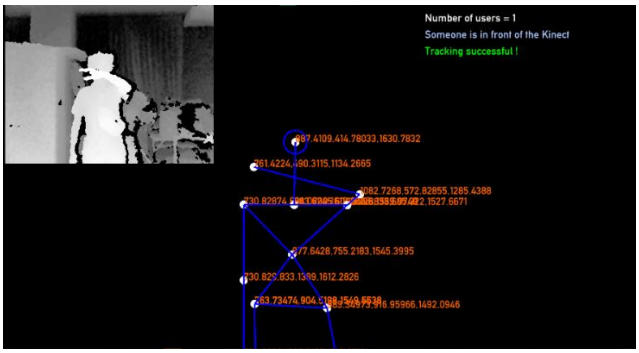
### 4.4.2 วิธีการทดลอง

ในการหาค่าพิกัดจาก Kinect เริ่มจากการทำ Skeleton tracking ซึ่งจะใช้วิธีการและทฤษฎีตามบทที่ 2 และใช้ไลบรารีของ Simple Open ni ในการคำนวณตำแหน่ง X Y และ Z บน Skeleton จะทำให้ได้ค่าพิกัด X Y และ Z ในแต่ละจุดบน Skeleton 20 จุด

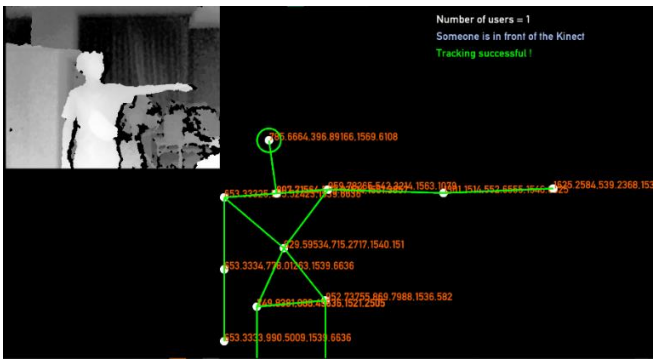



**รูปที่ 4.7 มุมการเคลื่อนไหวของแขนการทดลองค่าพิกัดบน Skeleton**

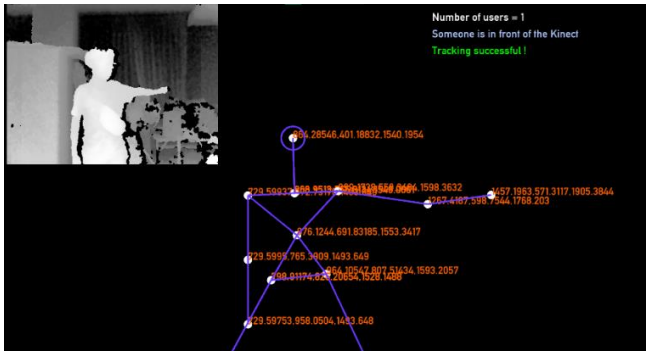
ที่มา: http://202.44.68.33/node/94124?page=0,5



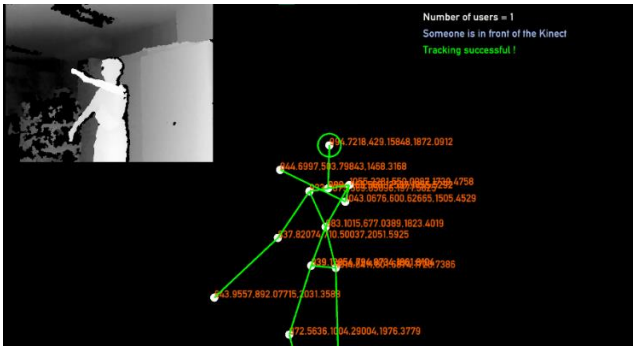

**รูปที่ 4.8 การทดลองหาค่าพิกัดบน Skeleton ในมุม 45-135 องศาโดยวางอุปกรณ์ 90 องศา**



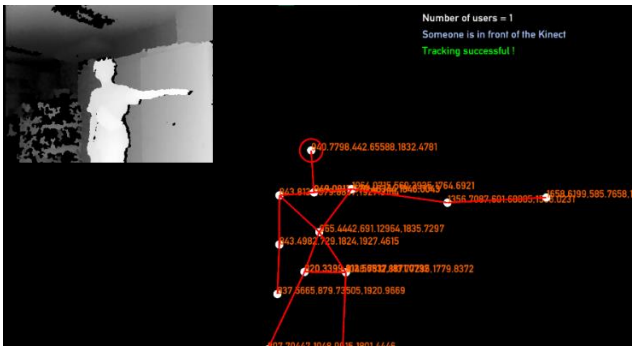

**รูปที่ 4.9 การทดลองหาค่าพิกัดบน Skeleton ในมุม 0-45 องศาโดยวางอุปกรณ์ 90 องศา**



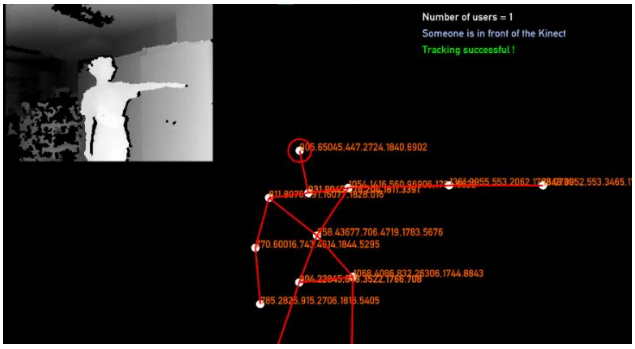

**รูปที่ 4.10 การทดลองหาค่าพิกัดบน Skeleton ในมุม 315-0 องศาโดยวางอุปกรณ์ 90 องศา**




**รูปที่ 4.11 การทดลองหาค่าพิกัดบน Skeleton ในมุม 45-135 องศาโดยวางอุปกรณ์ 45 องศา**




**รูปที่ 4.12 การทดลองหาค่าพิกัดบน Skeleton ในมุม 0-45 องศาโดยวางอุปกรณ์ 45 องศา**




**รูปที่ 4.13 การทดลองหาค่าพิกัดบน Skeleton ในมุม 315-0 องศาโดยวางอุปกรณ์ 45 องศา**

### 4.4.3 ผลการทดลอง

จากการทดลองทำให้ทราบว่าการวางกล้อง Kinect ในมุมตรงกับผู้ควบคุมตามรูปที่ 4.9 นั้นมีค่าพิกัด X Y และ Z ที่ผิดพลาดเนื่องจากการเคลื่อนไหวบางท่าทางทำให้กล้อง Kinect ไม่สามารถตรวจจับได้ ดังนั้นจึงมาทดลองโดยการเปลี่ยนมุมในการวางกล้อง Kinect ให้ทำมุม 45 องศากับผู้ควบคุมผลที่ได้คือสามารถแสดงค่าพิกัด X Y และ Z ได้แม่นยำและครอบคลุมมากกว่า

**ตารางที่ 4.4 ผลการทดลองหาค่าพิกัดโดย Kinect**

| อุปกรณ์ | ตำแหน่งการวางอุปกรณ์ | ตำแหน่งมือที่องศา | ค่าพิกัด x y และz ของมือ | ค่าพิกัด x y และz ของข้อศอก | ค่าพิกัด x y และz ของหัวไหล่ |
|---|---|---|---|---|---|
| Kinect | 90 องศา | 45-135 | 770, 490, 1134 | 1082, 572, 1285 | 930, 698, 1527 |
| | | 0-45 | 1625, 539, 1575 | 1314, 580, 1589 | 936, 699,1543 |
| | | 315-0 | 1457, 571, 1905 | 1267, 598, 1368 | 1000, 558, 1598 |
| | 45องศา | 45-135 | 844, 503, 1468 | 1043, 600, 1505 | 1055, 550, 1730 |
| | | 0-45 | 1658, 585, 1359 | 1356, 601, 1356 | 1064, 560, 1764 |
| | | 315-0 | 1847, 553, 1699 | 1361, 553, 1738 | 1054, 560, 1737 |

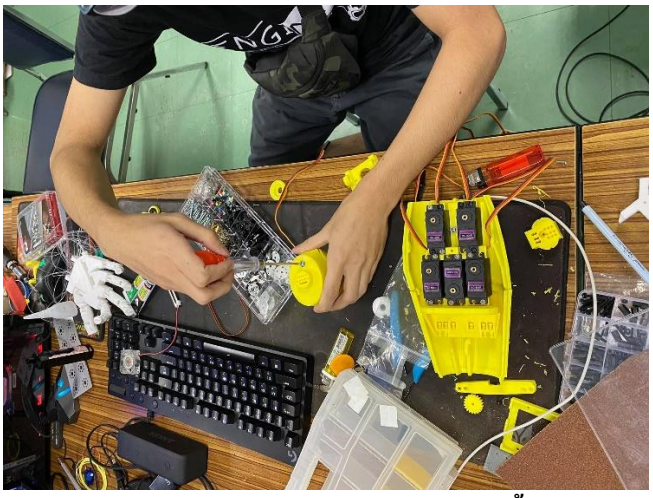
## 4.5 การทดลองการเคลื่อนไหวแขนกลในส่วนของมือ

### 4.5.1 จุดประสงค์การทดลอง

เพื่อทดลองการเคลื่อนไหวของข้อมือและทดลองประสิทธิภาพในการทำงานของ Servo Motor

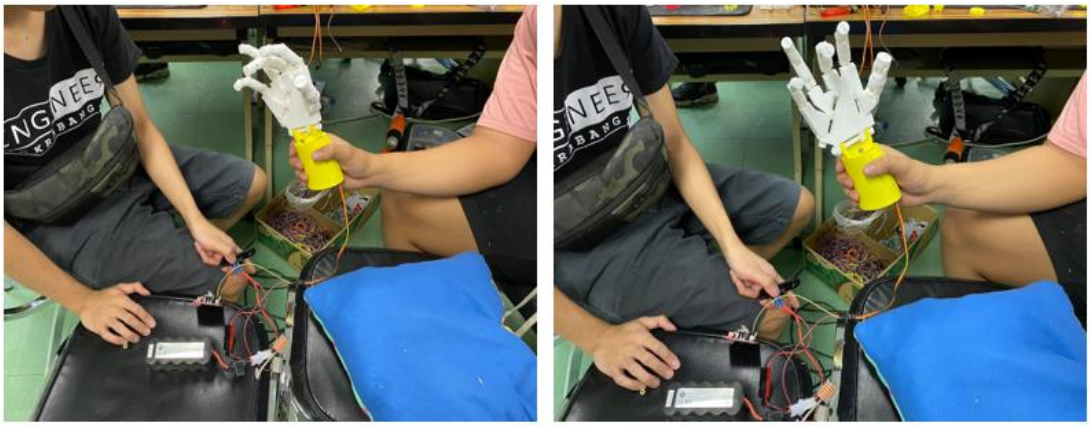
### 4.5.2 วิธีการทดลอง

1) ทำการประกอบชิ้นส่วนของมือและนิ้วมือเข้าด้วยกัน
2) ทำการร้อยเส้นเอ็นเข้ากับนิ้วต่าง ๆ
3) ทำการประกอบและติดตั้ง Servo Motor ที่ส่วนของข้อมือ
4) ทำการต่อ Arduino เข้ากับ Servo Motor
5) ทำการเริ่มทดลองการหมุนของแขนกลในส่วนข้อมือ



**รูปที่ 4.14 การประกอบแขนกลในส่วนของแขนและติดตั้ง Servo Motor**

**รูปที่ 4.15 การร้อยเส้นเอ็นเข้าส่วนของนิ้วมือ**



**รูปที่ 4.16 การทดลองการหมุนแขนกลในส่วนของข้อมือ**

**รูปที่ 4.17 แขนกลในส่วนของมือ นิ้วมือ และแขนส่วนล่าง**

### 4.5.3 ผลการทดลอง

หลังจากการประกอบชิ้นส่วนของแขนกลในส่วนมือ นิ้ว ข้อมือ และแขนส่วนล่างเป็นที่เรียบร้อยแล้วได้ทำการทดลองสั่งการทำงานของ Servo Motor ผ่าน Arduino ซึ่งผลการทดลองปรากฏว่าส่วนของข้อมือหมุนเคลื่อนไหวได้แต่ยังมีบางจังหวะของการหมุนยังติดขัดน่าจะเกิดจากการเสียดสีอาจจะต้องใช้จารบีมาช่วยให้อาการติดขัดลดลง

## 4.6 การทดลองการตอบสนองของแขนกล

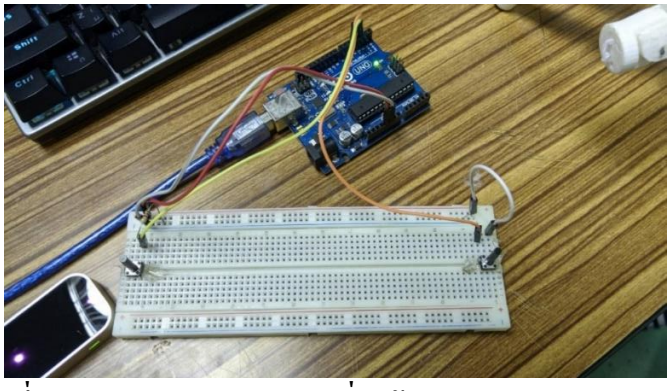
### 4.6.1 จุดประสงค์การทดลอง

เพื่อทดลองการตอบของแขนกล โดยในการทดลองนี้จะใช้เวลามาเป็นตัวชี้วัดในการเคลื่อนไหวของแขนกลหลังจากที่ผู้ควบคุมสั่งการ โดยการทดลองจะทำวงจรนับเวลาขึ้นมาและทำปุ่มกดขึ้นมาสองปุ่ม ซึ่งปุ่มแรกจะเป็นปุ่มเริ่มนับเวลาและปุ่มสองจะเป็นปุ่มหยุดเวลา โดยผู้ควบคุมจะทำการกดปุ่มเริ่มจากนั้นแขนกลก็จะทำท่ากดปุ่มเช่นเดียวกันแต่กดที่ปุ่มหยุดจะทำให้ทราบเวลาในการตอบสนองของแขนกล โดยจะแบ่งการทดลองออกเป็น 3 ส่วน

1) ผู้ควบคุมทำการเคลื่อนไหวแค่ส่วนของนิ้วมือ
2) ผู้ควบคุมทำการเคลื่อนไหวแค่ส่วนของข้อศอก
3) ผู้ควบคุมทำการเคลื่อนไหวแค่ส่วนของหัวไหล่

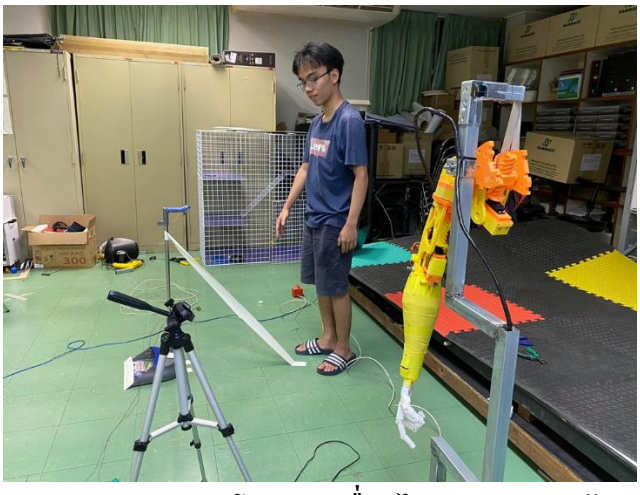
### 4.6.2 วิธีการทดลอง

1) ทำการต่อวงจรนับเวลาให้มีปุ่มเริ่มจับเวลากับปุ่มหยุดเวลา
2) ผู้ควบคุมทำการเคลื่อนไหวกดปุ่มเริ่มนับเวลาและรอแขนกลขยับกดปุ่มหยุด

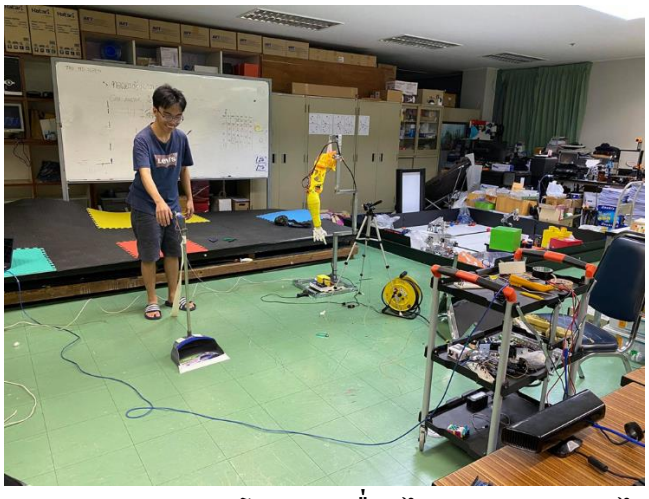


**รูปที่ 4.18 วงจรนับเวลาและปุ่มเริ่ม(ซ้าย)กับปุ่มหยุด(ขวา)**

**รูปที่ 4.19 การทดลองกดปุ่มโดยการเคลื่อนไหวแค่ส่วนของนิ้วมือ**



**รูปที่ 4.20 การทดลองกดปุ่มโดยการเคลื่อนไหวแค่ส่วนของข้อศอก**



**รูปที่ 4.21 การทดลองกดปุ่มโดยการเคลื่อนไหวแค่ส่วนของหัวไหล่**

**รูปที่ 4.22 การทดลองกดปุ่มโดยการเคลื่อนไหวแค่ส่วนของหัวไหล่**

### 4.6.3 ผลการทดลอง

จากตารางที่ 4.8 จะเห็นว่าในแต่ละครั้งที่ผู้ควบคุมทำการกดปุ่มเริ่มและแขนกลเคลื่อนไหวไปกดปุ่มหยุดนั้นในแต่ละส่วนการเคลลื่อนไหวเวลาที่ใช้จะไม่เท่ากัน ดังนั้นจึงสรุปได้ว่าการตอบสนองของแขนกลหลังจากการสั่งการใช้เวลาน้อยที่สุดอยู่ที่ 343.4 มิลลิวินาทีหรือ 0.3434 วินาทีถึงจะทำการเคลื่อนไหวได้ตามท่าทางที่สั่งการ ซึ่งเหตุผลที่เวลาในส่วนอื่นมีค่าต่างกันมากเนื่องจากเหตุผลข้อจำกัดทางด้าน Hardware

**ตารางที่ 4.5 ผลการทดลองการตอบสนองของแขนกลเคลื่อนไหวแค่ส่วนของนิ้วมือ**

| ครั้งที่ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | ค่าเฉลี่ย |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Delay (ms) | 199 | 369 | 230 | 210 | 291 | 442 | 310 | 367 | 306 | 710 | 343.4 |

**ตารางที่ 4.6 ผลการทดลองการตอบสนองของแขนกลเคลื่อนไหวแค่ส่วนของข้อศอก**

| ครั้งที่ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | ค่าเฉลี่ย |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Delay (ms) | 1286 | 974 | 902 | 1485 | 1052 | 982 | 1042 | 735 | 1421 | 1098 | 1097.7 |

**ตารางที่ 4.7 ผลการทดลองการตอบสนองของแขนกลเคลื่อนไหวแค่ส่วนของหัวไหล่**

| ครั้งที่ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | ค่าเฉลี่ย |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Delay (ms) | 1151 | 1860 | 1449 | 956 | 1112 | 1278 | 1314 | 1163 | 1386 | 1727 | 1339.6 |

**ตารางที่ 4.8 ผลการทดลองการตอบสนองของทั้ง 3 ส่วน**

| บริเวณส่วนที่เคลื่อนไหว | เวลาในการตตอบสนอง (วินาที) |
|---|---|
| นิ้วมือ | 0.3434 |
| ข้อศอก | 1.0977 |
| หัวไหล่ | 1.3396 |

## 4.7 การทดลองความแม่นยำของแขนกล

### 4.7.1 จุดประสงค์การทดลอง

เพื่อทดลองความแม่นยำในการรับค่าและการเคลื่อนไหวของแขนกล โดยการทดลองจะใช้การวาดเส้นเป็นตัวชี้วัดความแม่นยำว่าขนาดความยาวของเส้นที่ผู้ควบคุมวาดกับขนาดความยาวของเส้นที่แขนกลวาดนั้นมีขนาดใกล้เคียงกันเท่าใด

### 4.7.2 วิธีการทดลอง

1) ทำการติดปากกาไว้ที่นิ้วของแขนกล
2) ทำการติดตั้งแขนกลไว้กับขาตั้งเพื่อเตรียมการทดลอง
3) ผู้ควบคุมทำการเคลื่อนไหววาดเส้นลงกระดาษและรอแขนกลวาดเส้นลงกระดาษ

**รูปที่ 4.23 การติดตั้งแขนกลกับขาตั้ง**

**รูปที่ 4.24 การทดลองความแม่นยำ**

**4.7.3 ผลการทดลอง**

จากตารางที่ 4.6 จะเห็นได้ว่าค่าความแตกต่างที่คิดเป็นเปอร์เซ็นในสามครั้งค่อนข้างที่จะมีความห่างกัน โดยคิดเฉลี่ยแล้วค่าความแตกต่างโดยเฉลี่ยจะเท่ากับ 16.32 % ดังนั้นสรุปได้ว่าแขนกลมีการเคลื่อนไหวที่มีความแม่นยำโดยเฉลี่ยเท่ากับ 83.68 %

**ตารางที่ 4.9 ผลการทดลองความแม่นยำของแขนกล**

| ครั้งที่ | ขนาดความยาวเส้นของผู้ควบคุม (cm) | ขนาดความยาวเส้นของแขนกล (cm) | ขนาดความยาวแตกต่าง(cm) | ความแตกต่าง (%) |
|---|---|---|---|---|
| 1 | 3.5 | 4.3 | 0.8 | 22.85 |
| 2 | 6 | 6.9 | 0.9 | 15 |
| 3 | 9 | 10 | 1 | 11.11 |

# บทที่ 5
# สรุปผลและข้อเสนอแนะ

## 5.1 สรุปผล

ผู้ร่วมโครงงานได้พัฒนาต้นแบบแขนกลที่สามารถเคลื่อนไหวตามท่าทางการเคลื่อนไหวของมนุษย์ได้ โดยผู้ควบคุมไม่ต้องทำการสวมใส่อุปกรณ์ใด ๆ และส่วนของแขนกลได้ทำการพิมพ์สามมิติส่วนประกอบต่าง ๆ ของแขนกลและได้นำวัสดุเหล็กมาทำการดัดแปลงชิ้นส่วนบางอย่างให้เหมาะสมกับชิ้นงาน ส่วนของด้านการควบคุมและสั่งการได้ทำการศึกษาการอ่านข้อมูลและแปลงข้อมูลจากอุปกรณ์ Leap Motion และ Kinect แล้วได้ทำการส่งข้อมูลทั้งหมดผ่าน Wi-Fi เมื่อนำทั้งสองส่วนมารวมกันและทำการทดลอง ผลการตอบสนองในการเคลื่อนไหวของแขนกลใช้เวลาเฉลี่ย 0.38853 วินาทีและความแม่นยำในการเคลื่อนไหวตามท่าทางสั่งการเฉลี่ยอยู่ที่ 83.68 %

## 5.2 ปัญหาและอุปสรรค

1) เนื่องจากส่วนประกอบของแขนกลนั้นสั่งทำจากเครื่องพิมพ์ 3 มิติจึงทำให้มีปัญหาในเรื่องความแข็งแรง ความทนทาน และใช้เวลาค่อนข้างนานในการพิมพ์แต่ละชิ้นส่วน
2) ปัญหาตำแหน่งการวางอุปกรณ์ Kinect ให้สามารถตรวจจับท่าทางการเคลื่อนไหวของแขนและหัวไหล่ได้ครอบคลุมสุด
3) วัสดุอุปกรณ์ด้าน Hardware มีสามารถตอบสนองต่อข้อมูลที่ทำการส่งไปได้ไม่ดีพอมักจะเกิดปัญหาบ่อย ๆ เวลาที่มีการเปลี่ยนแปลงข้อมูลอย่างรวดเร็ว
4) ปัญหาความแม่นยำของอุปกรณ์ Kinect มีความคลาดเคลื่อน ซึ่งค่อนข้างมีหลายปัจจัยที่สามารถทำให้ตัวอุปกรณ์เกิดความคลาดเคลื่อนได้เลยทำให้อุปกรณ์มีปัญหาความแม่นยำ
5) ปัญหาอุปกรณ์ Leap Motion มีระยะในการตรวจจับสั้นไป เนื่องจากในระยะที่เว็บไซต์ได้ระบุไว้ไม่สามารถใช้งานได้จริงกับโครงงานนี้

## 5.3 แผนการดำเนินงานในอนาคต

เนื่องจากแขนกลนี้สามารถทำการเคลื่อนไหวและสามารถสั่งการควบคุมการทำงานตามที่ต้องการเบื้องต้นได้แล้ว ในอนาคตจะทำการพัฒนาส่วนของแขนกลให้สามารถเคลื่อนไหวได้ลื่นไหลและแม่นยำมากกว่าเดิม นอกจากนั้นจะมีการพัฒนาในเรื่องการรับข้อมูลให้มีการตอบสนองได้รวดเร็วมากขึ้น

# บรรณานุกรม


รังสิดล หงส์หิรัญพันธ์, และอื่นๆ. (2557). การควบคุมแขนกลขนาดเล็กด้วยคอมพิวเตอร์. บอร์ด Arduino (หน้า 31-35). ชลบุรี: ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ มหาวิทยาลัยบูรพา.

เจ้าของร้านFitrox Electronics. 2020. การสื่อสารแบบ I2C. [Online].
Available : http://fitrox.lnwshop.com/article/81/การสื่อสารแบบ-i2c

Abhishek Kar. 2011. Skeletal Tracking using Microsoft Kinect. [Online].
Available:https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.453.5802&rep=rep1&type=pdf

Antoine. 2017. Kinect - Skeleton. [Online].
Available : https://github.com/antoine1000/kinect-skeleton

D. Hoiem. 2010. Human Body Recognition and Tracking : How the Kinect Works. [Online].
Available : http://pages.cs.wisc.edu/~dyer/cs540/notes/17_kinect.pdf.

Darius Morawiec. 2016. leap-motion-processing. [Online].
Available : https://github.com/nok/leap-motion-processing

Giulio Marin, Fabio Dominio, Pietro Zanuttigh. 2014. Hand gesture recognition with leap motion and kinect devices. [Online].
Available : https://ieeexplore.ieee.org/document/7025313

SUPPORT THAIEASYELEC. 2020. บทความ ESPino32 ตอนที่ 8 การสื่อสารอนุกรมแบบ I2C. [Online].
Available : https://blog.thaieasyelec.com/espino32-ch8-how-to-use-i2c/

Totovr. 2019. SimpleOpenNI. [Online].
Available : https://github.com/totovr/SimpleOpenNI