เก้าอี้เข็นควบคุมได้สำหรับผู้พิการอวัยวะส่วนบนลงมา

CONTROLLABLE WHEEL CHAIR FOR UPPER AND LOWER LIMBS DISABILITY PERSON

โดย

นาย รัฐกานต์ บันที

นาย วรพล สุทธิมานะกิจ

นาย ศักดาเดช สุภารักษ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2560

เก้าอี้เข็นควบคุมได้สำหรับผู้พิการอวัยวะส่วนบนลงมา

CONTROLLABLE WHEEL CHAIR FOR UPPER AND LOWER LIMBS DISABILITY PERSON

โดย

นาย รัฐกานต์ บันที	57011058
นาย วรพล สุทธิมานะกิจ	57011105
นาย ศักดาเดช สุภารักษ์	57011233

อาจารย์ที่ปรึกษา

รศ. ดร. พิพัฒน์ พรหมมี

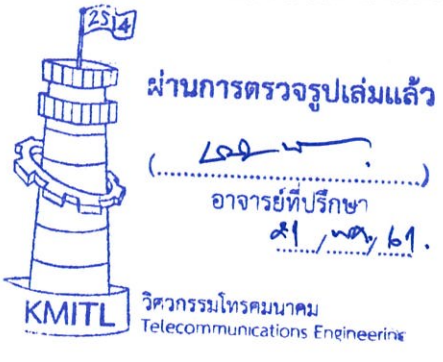
ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2560



ปริญญานิพนธ์ปีการศึกษา 2560

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง เก้าอี้เข็นควบคุมได้สำหรับผู้พิการอวัยวะส่วนบนลงมา

CONTROLLABLE WHEEL CHAIR FOR UPPER AND LOWER LIMBS DISABILITY PERSON

ผู้จัดทำ

1. นาย รัฐกานต์ บันที 57011058
2. นาย วรพล สุทธิมานะกิจ 57011105
3. นาย ศักดาเดช สุภารักษ์ 57011233

..... อาจารย์ที่ปรึกษา

(รศ. ดร. พิพัฒน์ พรหมมี)

กิตติกรรมประกาศ

ปริญญานิพนธ์ในหัวข้อเรื่อง “เก้าอี้เข็นควบคุมได้สำหรับผู้พิการอวัยวะส่วนบนลงมา” จะไม่สามารถสำเร็จลุล่วงเลย หากขาดการสนับสนุนจาก รศ.ดร. พิพัฒน์ พรหมมี โดยท่านเป็นอาจารย์ที่ปรึกษาโครงงานฉบับนี้ ซึ่งคอยให้คำปรึกษาในการศึกษาค้นคว้าและจัดทำโครงงาน สนับสนุนอุปกรณ์ที่ใช้ในการทำโครงงาน ตลอดจนชี้แนะแนวทางการแก้ไขปัญหาที่เกิดขึ้น

ขอขอบคุณ คุณเพชร บันที ที่ให้ความอนุเคราะห์และให้คำแนะนำในการดัดแปลงโครงสร้างและจัดทำชุดกลไกขับเคลื่อนเก้าอี้เข็น

ขอขอบคุณ คุณณรงค์ศักดิ์ มโนสิทธิชัย ที่ให้ความช่วยเหลือในการซ่อมบำรุงเก้าอี้เข็นจนสามารถใช้งานได้ตามปกติ

ขอขอบคุณ บิดา มารดา ครอบครัวของผู้จัดทำ ที่ให้การสนับสนุนด้านทุนทรัพย์ในการศึกษาและคอยให้กำลังใจในยามท้อถอย รวมไปถึงสมาชิกในกลุ่มที่มีความร่วมแรงร่วมใจช่วยเหลือเกื้อกูลกันเป็นอย่างดีในการทำโครงงานครั้งนี้จนกระทั่งประสบความสำเร็จด้วยดี ผู้จัดทำจึงขอกราบขอขอบพระคุณทุก ๆท่านเป็นอย่างสูงมา ณ โอกาสนี้

นาย รัฐกานต์ บันที
นาย วรพล สุทธิมานะกิจ
นาย ศักดาเดช สุภารักษ์
ผู้จัดทำ

เก้าอี้เข็นควบคุมได้สำหรับผู้พิการอวัยวะส่วนบนลงมา
Controllable wheel chair for upper and lower limbs disability person

โดย นาย รัฐกานต์ บันที 57011058
นาย วรพล สุทธิมานะกิจ 57011105
นาย ศักดาเดช สุภารักษ์ 57011233

อาจารย์ที่ปรึกษา รศ. ดร. พิพัฒน์ พรหมมี

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้นำเสนอการออกแบบและสร้างเก้าอี้เข็นควบคุมได้สำหรับผู้พิการอวัยวะส่วนบนลงมาและผู้สูงอายุ เก้าอี้เข็นนี้ขับเคลื่อนด้วยมอเตอร์ไฟฟ้ากระแสตรงที่ถูกควบคุมด้วยบอร์ดไมโครคอนโทรลเลอร์ สามารถเลือกควบคุมทิศทางการเคลื่อนที่ของเก้าอี้เข็นได้ตามความเหมาะสมของสภาพร่างกาย ได้แก่ การใช้ก้านควบคุม, การควบคุมทิศทางด้วยการเอียงศีรษะของผู้ใช้งานผ่านการตรวจวัดมุมเอียงของศีรษะด้วยเซนเซอร์ความเร่งเชิงเส้น ซึ่งออกแบบสำหรับผู้สูงอายุและผู้พิการที่ไม่สามารถใช้งานอวัยวะส่วนล่างในการควบคุมเก้าอี้เข็น และการควบคุมทิศทางบนแอปพลิเคชันสำหรับผู้ดูแล เก้าอี้เข็นนี้มีระบบเรียกผู้ดูแลเพื่อขอความช่วยเหลือโดยแจ้งเตือนให้ผู้ดูแลทราบผ่านแอปพลิเคชันบนสมาร์ตโฟน

ABSTRACT

This thesis presents the design and implementation of a controllable wheelchair for the upper and lower limbs disability person or elder person. This wheelchair is driven by a DC motors, sensors and a micro controller unit. The direction and movement can be controlled by 3 methods, manual joystick, head tilt and manual on smart phone for a caregiver. The angle of user's head tilt is measured by using linear acceleration sensor that is suitable for hand disability user. The wheel chair also has an automatic notification on smart phone to a caregiver when user call for help.

สารบัญ

สารบัญ (ต่อ)

สารบัญรูป

หน้า

รูปที่

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญตาราง

สารบัญตาราง (ต่อ)

สารบัญตาราง (ต่อ)

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

เก้าอี้เข็นสำหรับผู้พิการเป็นอุปกรณ์ที่ออกแบบมาเพื่อช่วยเหลืออำนวยความสะดวกแก่ผู้พิการและผู้สูงอายุ ซึ่งเป็นบุคคลที่สูญเสียความสามารถของอวัยวะร่างกายในการเคลื่อนไหว โดยเก้าอี้เข็นที่มีจำหน่ายทั่วไปตามท้องตลาดมี 2 ชนิดคือ ชนิดที่ผู้ใช้ต้องใช้มือหมุนล้อให้ขับเคลื่อนไปได้เอง และชนิดที่ขับเคลื่อนด้วยมอเตอร์ไฟฟ้าโดยใช้การควบคุมจากก้านควบคุมหรือปุ่มกด แต่เนื่องจากเก้าอี้เข็นทั้งสองประเภทนี้ไม่ได้ผลิตมาเพื่อรองรับการใช้งานของผู้พิการที่นอกจากจะไม่สามารถเคลื่อนที่ด้วยการเดินแล้ว อีกทั้งยังไม่สามารถใช้แขนหรือมือบังคับก้านควบคุมหรือปุ่มกดเพื่อควบคุมเก้าอี้เข็นได้ ผู้จัดทำจึงมีแนวคิดที่จะสร้างเก้าอี้เข็นนั่งสำหรับผู้พิการตั้งแต่อวัยวะส่วนบนลงมาและผู้สูงอายุที่สามารถเลือกลักษณะการควบคุมทิศทางการเคลื่อนที่ของเก้าอี้เข็นตามความเหมาะสมของสภาพร่างกายผู้ใช้งาน ได้แก่ การใช้ก้านควบคุม และการเอียงศีรษะของผู้ใช้งาน ทั้งนี้ผู้ดูแลสามารถควบคุมทิศทางเก้าอี้เข็นบนแอปพลิเคชันเพื่อให้ความช่วยเหลือผู้พิการในเบื้องต้น โดยเก้าอี้เข็นจะขับเคลื่อนด้วยมอเตอร์ที่ถูกควบคุมด้วยไมโครคอนโทรลเลอร์ อีกทั้งยังมีระบบเรียกผู้ดูแลเพื่อขอความช่วยเหลือโดยแจ้งเตือนให้ผู้ดูแลทราบผ่านแอปพลิเคชันบนสมาร์ตโฟน

1.2 วัตถุประสงค์

1) เพื่อออกแบบและสร้างเก้าอี้เข็นควบคุมได้ที่ช่วยอำนวยความสะดวกในการเคลื่อนที่ให้กับผู้พิการอวัยวะส่วนบนลงมาและผู้สูงอายุ
2) เพื่อออกแบบและสร้างเก้าอี้เข็นที่ควบคุมการเคลื่อนที่โดยควบคุมจากอวัยวะส่วนที่เหลือของผู้ใช้งาน
3) เพื่อออกแบบแอปพลิเคชันบนสมาร์ตโฟนแจ้งเตือนผู้ดูแลเมื่อมีการเรียกขอความช่วยเหลือจากผู้ใช้งาน

1.3 ขอบเขตของปริญญานิพนธ์

ออกแบบและสร้างเก้าอี้เข็นควบคุมได้สำหรับผู้พิการอวัยวะส่วนบนลงมาที่มีอาการบาดเจ็บไขสันหลังแบบสมบูรณ์โดยอัมพาตทั้งแขนและขาทั้ง 2 ข้างอันเกิดจากภยันตรายต่อไขสันหลังส่วนคอ (Quadriplegia) หรือผู้สูงอายุ ซึ่งไม่สามารถใช้แขนและมือในการบังคับควบคุมเก้าอี้เข็นได้ตามปกติ โดยผู้ใช้งานสามารถเลือกใช้การควบคุมทิศทางการเคลื่อนที่ของเก้าอี้เข็นตามความเหมาะสม

กับสภาพร่างกายของผู้ใช้งานได้ 3 แบบได้แก่ การควบคุมทิศทางการเคลื่อนที่ของเก้าอี้เข็นด้วยอวัยวะส่วนที่เหลือของผู้ใช้งานผ่านการเอียงศีรษะของผู้ใช้งานด้วยการตรวจวัดมุมเอียงของศีรษะด้วยเซนเซอร์ความเร่งเชิงเส้น หรือการควบคุมทิศทางผ่านการบังคับก้านควบคุมด้วยมือ ทั้งนี้ผู้ดูแลสามารถควบคุมทิศทางของเก้าอี้เข็นผ่านแอปพลิเคชันบนสมาร์ตโฟน เพื่อช่วยเหลือผู้ใช้งานในเบื้องต้นหากเกิดปัญหาในการเคลื่อนที่ของเก้าอี้เข็น ซึ่งเก้าอี้เข็นนี้ขับเคลื่อนด้วยมอเตอร์ไฟฟ้ากระแสตรงที่ถูกควบคุมโดยไมโครคอนโทรลเลอร์ เก้าอี้เข็นยังมีระบบเรียกผู้ดูแลเพื่อขอความช่วยเหลือ โดยแจ้งเตือนให้แก่ผู้ดูแลทราบผ่านแอปพลิเคชันบนสมาร์ตโฟน

บทที่ 2

ทฤษฎีและหลักการที่เกี่ยวข้อง

ผู้พิการทางการเคลื่อนไหวเป็นบุคคลที่มีข้อจำกัดในการปฏิบัติกิจวัตรประจำวันหรือการมีส่วนร่วมในกิจกรรมของสังคม อันเนื่องมาจากความบกพร่องหรือสูญเสียความสามารถในการควบคุมอวัยวะในการเคลื่อนไหวก่อให้เกิดความยากลำบากต่อการดำเนินชีวิต ดังนั้นในการเสริมสร้างความสามารถหรือคงสมรรถภาพทางร่างกายของผู้พิการให้ดีขึ้นจึงมีความสำคัญและมีความจำเป็นอย่างยิ่ง โดยอาศัยเครื่องมืออุปกรณ์ในทางการแพทย์ เช่น เก้าอี้เข็น ไม้เท้า เป็นต้น เพื่ออำนวยความสะดวกและส่งเสริมให้ผู้พิการสามารถปฏิบัติกิจกรรมในชีวิตประจำวัน รวมถึงสามารถอยู่ร่วมกันกับสังคมได้อย่างบุคคลทั่วไป ปริญญานิพนธ์นี้จึงนำเสนอเก้าอี้เข็นควบคุมได้สำหรับผู้พิการอวัยวะส่วนบนลงมา ซึ่งเป็นบุคคลที่ไม่สามารถควบคุมหรือเคลื่อนไหวอวัยวะส่วนล่าง ได้แก่ แขน ขา ในการเคลื่อนที่หรือทำกิจกรรมต่างๆ โดยจะมีลักษณะของความพิการดังต่อไปนี้

2.1 การบาดเจ็บของไขสันหลัง (Acute Traumatic Spinal Cord Injury)

การบาดเจ็บของไขสันหลังสามารถเกิดขึ้นโดยลำพัง (Isolated Spinal Cord Injury) หรือเกิดร่วมกับภาวะกระดูกสันหลังและข้อต่อเคลื่อนหลุด (associated Spinal Fracture and Dislocation) สาเหตุส่วนใหญ่ของการบาดเจ็บของไขสันหลังในประเทศไทย ได้แก่ อุบัติเหตุต่างๆ โดยเฉพาะอุบัติเหตุบนท้องถนน, การตกจากที่สูงการเล่นกีฬา โดยพบในเพศชายมากกว่าเพศหญิง ช่วงอายุที่พบบ่อยได้แก่ 16-30 ปี

2.1.1 ไขสันหลังบาดเจ็บแบบสมบูรณ์ (Complete Spinal Cord Injury)

คือภาวะที่ไขสันหลังได้รับภยันตรายทั้งหมด ทำให้สูญเสียการทำงานทั้งหมดตั้งแต่ระดับที่มีพยาธิสภาพลงไป เมื่อผ่านพ้นระยะ Spinal Shock แล้วไม่หลงเหลือการทำงานของระบบประสาทใดๆในระดับต่ำกว่าระดับของไขสันหลังที่ได้รับบาดเจ็บ แสดงว่ามีการตัดขาดการทำงานของไขสันหลังส่วนบนออกจากส่วนล่างที่ต่ำกว่าภยันตรายอย่างสมบูรณ์ การสั่งงานของสมองไม่สามารถผ่านไขสันหลังไปสั่งงานกล้ามเนื้อ และไม่สามารถรับรู้ความรู้สึกจากส่วนของร่างกายที่อยู่ต่ำกว่าระดับภยันตรายไปยังสมองได้ แบ่งออกเป็น 2 แบบ ได้แก่

2.1.1.1 Quadriplegia หมายถึง อัมพาตทั้งแขนและขาทั้ง 2 ข้าง เกิดจากภยันตรายต่อไขสันหลังส่วนคอ

2.1.1.2 Paraplegia หมายถึง อัมพาตเฉพาะส่วนขา เกิดจากภยันตรายของไขสันหลังต่ำกว่าคอลงมา

2.1.2 ไขสันหลังบาดเจ็บแบบไม่สมบูรณ์ (Incomplete Spinal Cord Injury)

หมายถึงไขสันหลังเฉพาะส่วนใดส่วนหนึ่งได้รับอันตราย เมื่อผ่านพ้นระยะ Spinal shock แล้ว ยังหลงเหลือการทำงานของระบบประสาทในระดับต่ำกว่าระดับของไขสันหลังที่ได้รับบาดเจ็บ ไม่ว่าจะเป็นกำลังกล้ามเนื้อ การรับรู้ความรู้สึก การทำงานของระบบ ระบบประสาทอัตโนมัติ โดยอาการแสดงจะแตกต่างกันแล้วแต่ตำแหน่งและความรุนแรงของไขสันหลังที่ได้รับบาดเจ็บ ได้แก่

2.1.2.1 Central Cord Syndrome การที่ไขสันหลังส่วนกลางได้รับภยันตรายก่อนบริเวณรอบ ๆ ส่วนนอกเป็นชนิดที่พบบ่อยที่สุด ส่วนใหญ่พบในผู้ป่วยสูงอายุที่มี การเสื่อมของข้อต่อกระดูกสันหลัง (Preexisting Cervical Spondylosis) ซึ่งจะทำให้มีแรงอัดจากทางด้านหน้าและด้านหลังของไขสันหลังพร้อม ๆ กับเลือดออกจากตรงกลางก่อนที่จะกระจายไปสู่รอบข้าง ผู้ป่วยจะมีอาการอ่อนแรงของกล้ามเนื้อชาที่บริเวณแขนและมือก่อน และรุนแรงกว่ากล้ามเนื้อของขา บางรายสูญเสียการทำงานของระบบประสาททั้งแขนและขาแต่ยังหลงเหลือความรู้สึกบริเวณรอบทวารหนักยังสามารถขมิบกลามเนื้อหูรูดรอบทวารหนักได้

2.1.2.2 Anterior cord Syndrome การที่ไขสันหลังส่วนหน้าได้รับภยันตราย มักเกิดจากกระดูกที่แตกยุบและย้อนเข้าไปกระดูกไขสันหลังจากทางด้านหน้าหรือเกิดจากหมอนรองกระดูกสันหลัง (Traumatic Disc Herniation) เคลื่อนกดทับไขสันหลังโดยตรง ทำให้เกิดการขาดเลือด ผู้ป่วยจะอ่อนแรงแขนและขาทั้งสองข้าง เสียการรับรู้ความรู้สึกเจ็บปวด, อุณหภูมิ และสัมผัสของร่างกายทั้งสองข้าง โดยยังคงเหลือระบบรับรู้ความรู้สึกการทรงตัว การรับรู้ความรู้สึกของข้อ (proprioception sense) และการสั่นสะเทือน (vibration sense) อยู่

2.1.2.3 Brown–Sequard Syndrome คือการที่ไขสันหลงซีกใดซีกหนึ่งได้รับภยันตราย ผู้ป่วยจะมีอาการอ่อนแรงของกล้ามเนื้อและสูญเสียการรับรู้ความรู้สึกการทรงตัว, การรับรู้ความรู้สึกของข้อ และการสั่นสะเทือน ของร่างกายซีกเดียวกับที่ไขสันหลังบาดเจ็บและสูญเสียการรับรู้ความรู้สึกเจ็บปวด, อุณหภูมิ, และ สัมผัสของร่างกายซีกตรงกันข้ามกับไขสันหลังที่ได้รับบาดเจ็บ

2.1.2.4 Posterior cord syndrome พบได้น้อยที่สุด ผู้ป่วยจะสูญเสียการรับรู้ความรู้สึกการทรงตัว, การรับรู้ความรู้สึกของข้อ และการสั่นสะเทือนของร่างกายทั้งสองข้าง โดยมักไม่พบการอ่อนแรงกล้ามเนื้อหรือเสียการรบรู้ความรู้สึกเจ็บปวด อุณหภูมิ และสัมผัส

2.1.2.5 Mixed syndrome พบได้บ่อยเช่นเดียวกับ Central Cord Syndrome อาการของผู้ป่วย ไม่สามารถจัดเขาได้กับอาการของ syndrome ในข้างต้น

2.1.2.6 Conus medullaris syndrome ผู้ป่วยจะเกิดอัมพาตของขาทั้งสองข้าง สูญเสียการทำงานของระบบรับรู้ความรู้สึก พบการตอบสนองไวกว่าปกติ (Hyperreflexia) ผู้ป่วยจะสูญเสียการควบคุมการทำงานของระบบปัสสาวะ กล้ามเนื้ออหูรูดบริเวณทวารหนักอ่อนแรง และสูญเสียความรู้สึกรอบทวารหนักโดยส่วนแขนยังคงทำงานได้ดีอยู่

2.1.3 กลุ่มอาการกดทับรากประสาทส่วน Cauda Equina (Cauda Equina Compression Syndrome)

ผู้ป่วยจะมีอาการปวดร้าวลงขา น่อง ตามแนวเส้นประสาท กล้ามเนื้อขาอ่อนแรงหลายมัด มีอาการชารอบรูทวารหนัก (Saddle Anesthesia) อัมพาตของกล้ามเนื้อหูรูดทวารหนัก กลั้นหรือถ่ายปัสสาวะ อุจจาระเองไม่ได้ โอกาสการฟื้นตัวของระบบประสาทดีกว่าการบาดเจ็บของไขสันหลัง เนื่องจากกลุ่มอาการนี้เป็นการบาดเจ็บของรากประสาท

เก้าอี้เข็นนั้นเป็นอุปกรณ์ทางการแพทย์ที่นิยมใช้สำหรับผู้พิการที่ไม่สามารถควบคุมหรือเคลื่อนไหวอวัยวะส่วนล่าง เช่น แขน ขา หรือผู้ป่วยที่มีอาการอัมพาตท่อนล่าง รวมไปถึงผู้สูงอายุที่ไม่สามารถเดินได้สะดวกให้สามารถเคลื่อนที่ไปตามต้องการได้ ทั้งนี้เก้าอี้เข็นที่มีจำหน่ายในท้องตลาดนั้นเป็นเก้าอี้เข็นที่ต้องใช้แรงเข็นโดยผู้พิการเองหรือเข็นโดยผู้ดูแลในการขับเคลื่อน ซึ่งไม่สะดวกต่อผู้พิการหรือผู้สูงอายุ ดังนั้นการนำมอเตอร์ไฟฟ้ามาช่วยในการขับเคลื่อนเก้าอี้เข็นจึงช่วยอำนวยความสะดวกสบายให้กับผู้พิการเป็นอย่างมาก

2.2 มอเตอร์ไฟฟ้า

2.2.1 ชนิดของมอเตอร์ไฟฟ้า

มอเตอร์ไฟฟ้าแบ่งออกตามชนิดของกระแสไฟฟ้า คือ มอเตอร์ไฟฟ้ากระแสสลับ (Alternating Current Motor) และ มอเตอร์ไฟฟ้ากระแสตรง (Direct Current Motor)

2.2.2 ส่วนประกอบของมอเตอร์ไฟฟ้ากระแสตรง

2.2.2.1 ส่วนที่อยู่กับที่ (Stator)

1) เฟรมหรือโยค (Frame/Yoke) เป็นโครงสร้างหลักของตัวมอเตอร์ ทำจากโลหะม้วนเป็นรูปทรงกระบอก ทำหน้าที่ยึดส่วนประกอบอื่นๆของมอเตอร์เข้าด้วยกัน และทำหน้าที่เป็นเส้นทางเดินของเส้นแรงแม่เหล็กจากขั้วเหนือไปยังขั้วใต้อีกด้วย

2) ขั้วแม่เหล็ก (Pole) ประกอบด้วย 2 ส่วนหลักคือแกนขั้วแม่เหล็ก (Pole Core) และขดลวดสนามแม่เหล็ก (Field Coil) ส่วนแรกแกนขั้วแม่เหล็ก (Pole Core) มีหน้าที่ทำให้ขั้วแม่เหล็ก (Pole Shoes) และโรเตอร์ (Rotor) อยู่ติดกันมากที่สุดเพื่อลดช่องอากาศระหว่างขั้วแม่เหล็กและโรเตอร์ ซึ่งจะส่งผลให้เส้นแรงไหลผ่านไปยังโรเตอร์ได้มาก ซึ่งจะแปรผันตรงกับกำลังของมอเตอร์ไฟฟ้า ขดลวดสนามแม่เหล็ก (Field Coil) ทำหน้าที่รับกระแสไฟฟ้าเพื่อสร้างเส้นแรงแม่เหล็ก เส้นแรงแม่เหล็กนี้จะการหักล้างและเสริมกันกับสนามแม่เหล็กของอาเมเจอร์ทำให้เกิดแรงบิดขึ้น

2.2.2.2 โรเตอร์ (Rotor) หรือตัวหมุน มีตลับลูกปืนอยู่บริเวณหัวและท้ายของมอเตอร์ ตัวโรเตอร์ประกอบด้วย 4 ส่วนด้วยกันได้แก่

1) แกนเพลา (Shaft) ทำหน้าที่ยึดคอมมิวเตเตอร์ และแกนอาร์มาเจอร์ (Armature Core) เข้าด้วยกัน วางอยู่บนตลับลูกปืนส่งผลให้สามารถหมุนได้โดยไม่มีการสั่นสะเทือน

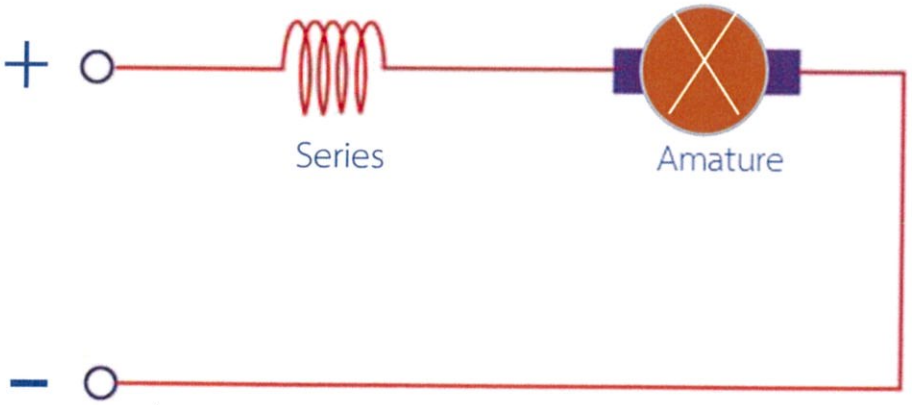
2) แกนเหล็กอาร์มาเจอร์ (Armature Core) เป็นพื้นที่สำหรับพันขดลวดอาร์มาเจอร์เพื่อสร้างแรงบิดเพื่อใช้ในการหมุนมอเตอร์

3) คอมมิวเตเตอร์ (Commutator) คือส่วนเคลื่อนที่อีกส่วนหนึ่ง ทำหน้าที่เป็นขั้วรับแรงดันไฟตรงที่จ่ายมาจากแปรงถ่าน เพื่อส่งไปให้ขดลวดอาร์เมอร์ คอมมิวเตเตอร์ถูกยึดติดเข้ากับอาร์เมเจอร์และแกนเพลา ทำจากแท่งทองแดงรูปทรงกระบอก แต่ละแท่งทองแดงของคอมมิวเตเตอร์จะถูกแยกออกจากกันด้วยฉนวนไมก้า (Mica)

4) ขดลวดอาร์มาเจอร์ (Armature Winding) เป็นขดลวดที่พันอยู่รอบแกนอามาเจอร์โดยมีจำขนวนรอบและขนาดของขดลวดที่แตกต่างกันออกไปตามลักษณะการใช้งาน

2.2.3 มอเตอร์ไฟฟ้ากระตรงแบบอนุกรม (Series Motor)

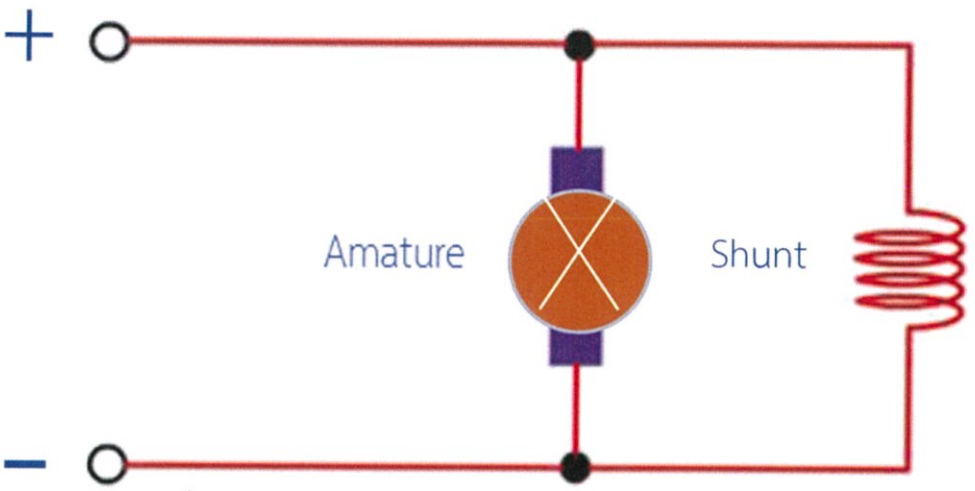
มอเตอร์ชนิดนี้จะต่ออนุกรมกันระหว่างขดลวดสนามแม่เหล็กกับขดลวดอาร์มาเจอร์ ดังรูปที่ 2.1 มอเตอร์ชนิดนี้มีจุดเด่นคือให้แรงบิดที่สูง และมีความเร็วรอบที่สูงในกรณีที่ไม่มีโหลด แต่หากมีโหลดความเร็วจะลดลงตามขนาดของโหลด ในกรณีนี้จะไม่ก่อให้เกิดความเสียหายต่อขดลวด พบในเครื่องใช้ไฟฟ้าหลายประเภท มอเตอร์ชนิดนี้จะให้รอบที่สูงมากจนอาจจะเกิดอันตรายต่อผู้ใช้หากไม่ต่อโหลด ดังนั้นจึงต้องมีโหลดต่อเสมอก่อนทำการใช้งานมอเตอร์ชนิดนี้



รูปที่ 2.1 วงจรการทำงานของมอเตอร์ไฟฟ้ากระตรงแบบอนุกรม

2.2.4 มอเตอร์ไฟฟ้ากระแสตรงแบบขนาน (Shunt Motor)

มอเตอร์ชนิดนี้จะมีการวางตัวขนานกันระหว่างขดลวดสนามแม่เหล็กกับขดลวดอาเมเจอร์ตามรูปที่ 2.2 มอเตอร์ชนิดนี้จะมีความเร็วรอบคงที่ แต่จะมีแรงบิดเริ่มหมุนที่ต่ำ พบได้ในเครื่องใช้ไฟฟ้าที่ต้องการรอบคงที่

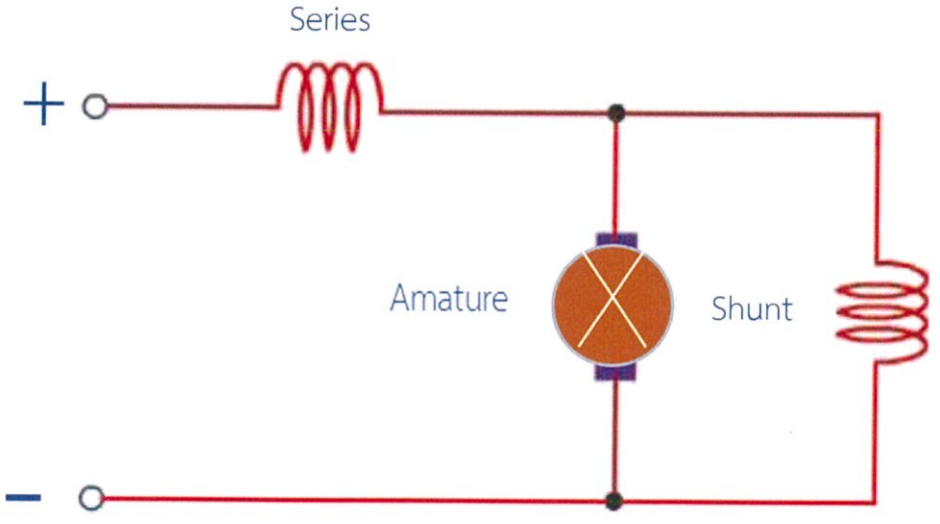


รูปที่ 2.2 วงจรการทำงานมอเตอร์ไฟฟ้ากระแสตรงแบบขนาน

2.2.5 มอเตอร์ไฟฟ้ากระแสตรงแบบผสม (Compound Motor)

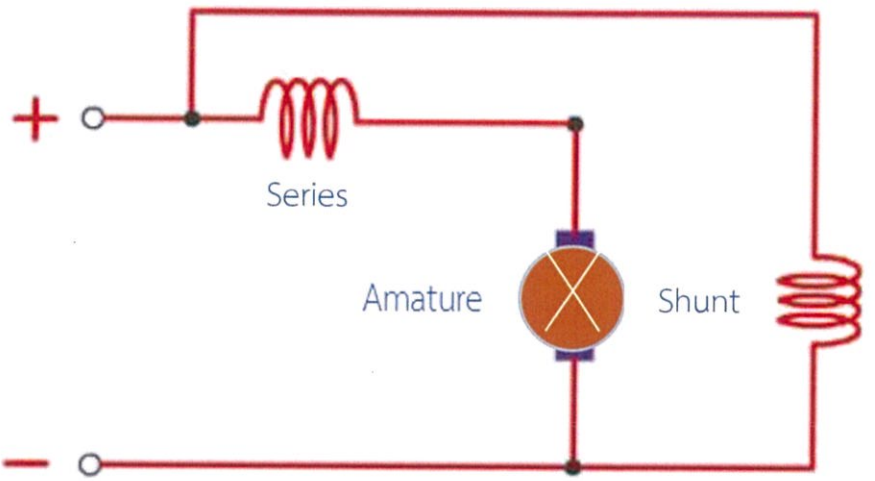
มอเตอร์ชนิดนี้จะนำข้อดีของมอเตอร์ไฟฟ้าแบบอนุกรมที่มีแรงบิดสูง แต่มีความเร็วรอบไม่คงที่กับมอเตอร์ไฟฟ้าแบบขนานที่มีความเร็วรอบคงที่มารวมกัน ทำให้ได้มอเตอร์ไฟฟ้าที่มีแรงบิดสูงและมีความเร็วรอบคงที่ โดยไม่แปรผันตามขนาดของโหลดที่ต่อไปยังมอเตอร์ชนิดนี้ มอเตอร์แบบผสมนี้จะมีวิธีการต่อแบบ 2 วิธี โดยจะขึ้นกับลักษณะการวางตัวของขวดลวดขนาน

วิธีที่ 1 ชอทชันท์ (Short Shunt Compound Motor) ขดลวดขนานจะวางตัวขนานกับอาเมเจอร์ดังรูปที่ 2.3



รูปที่ 2.3 วงจรการทำงานมอเตอร์ไฟฟ้ากระตรงแบบชอร์ทชั้นท์คอมปาว์ดมอเตอร์

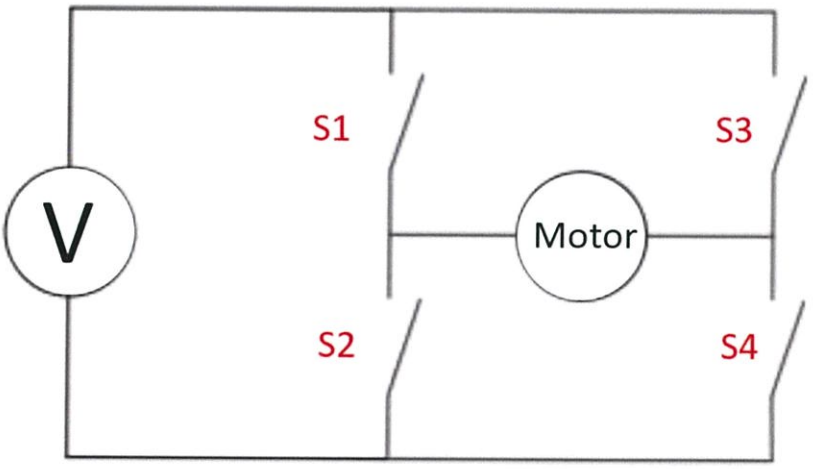
วิธีที่ 2 ลองชั้นท์คอมปาวด์มอเตอร์ (Long shunt motor) ขดลวดขนานจะต่อขนานกับขดลวดอาเมเจอร์ที่ต่ออนุกรมอยู่กับขดลวดอนุกรมดังรูปที่ 2.4



รูปที่ 2.4 วงจรการทำงานมอเตอร์ไฟฟ้ากระตรงแบบลองชั้นท์คอมปาวด์มอเตอร์

2.2.6 หลักการของมอเตอร์กระแสไฟฟ้าตรง

หลักการทำงานของมอเตอร์ไฟฟ้ากระแสตรงคือการตัดกันของไฟฟ้ากระแสตรงกับสนามแม่เหล็กจนเกิดการหมุนของมอเตอร์ มอเตอร์จะเริ่มทำงานเมื่อมีไฟฟ้ากระแสตรงจ่ายเข้าไปยังมอเตอร์ กระแสไฟฟ้าจะไหลไปยังขดลวดสนามแม่เหล็กซึ่งจะก่อให้เกิดสนามแม่เหล็กในแนวขั้วเหนือขั้วใต้ โดยเส้นแรงแม่เหล็กของทั้งสองขั้วนั้นจะไม่ตัดกัน หากมีทิศตรงข้าม เส้นแรงแม่เหล็กทั้งสองจะหักล้างกัน หากมีทิศเดียวกันจะเสริมกัน จึงเกิดแรงบิดของมอเตอร์ขึ้น กระแสไฟอีกส่วนหนึ่งจะไหลผ่านคอมมิวเตเตอร์ไปยังขดลวดอาร์มาเจอร์ก่อให้เกิดสนามแม่เหล็กขึ้น



รูปที่ 2.5 วงจรควบคุมทิศทางและความเร็วของมอเตอร์

2.2.7 การควบคุมทิศทาง และความเร็วรอบของมอเตอร์

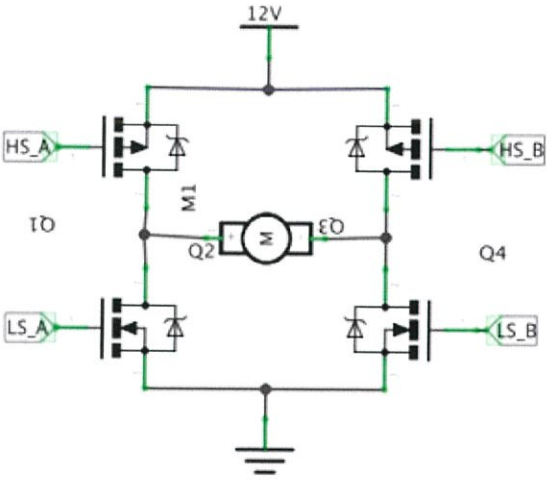
ในการควบคุมรอบมอเตอร์ รวมไปถึงการควบคุมทิศทางนั้นจำเป็นต้องมีวงจรที่ใช้ในการควบคุม วงจรชนิดหนึ่งที่เป็นที่นิยมอย่างแพร่หลายคือวงจร H-Bridge motor driver วงจรชนิดนี้สามารถควบคุมได้ทั้งทิศทางว่าจะหมุนในทิศทางปกติหรือหมุนในทิศตรงกันข้ามรวมถึงสามารถควบคุมความเร็วของมอเตอร์โดยใช้หลักการของ PWM ด้วย ในกรณีแรกที่ต้องการให้มอเตอร์หมุนตามเข็ม (Clockwise) จะทำการปิดวงจรที่ S1 และ S4 และเปิดวงจรที่ S2 และ S3 และในกรณีที่ต้องการให้มอเตอร์หมุนทวนเข็ม (Conter Clockwise) จะปิดวงจรที่ S2 และ S3 และเปิดวงจรที่ S1 และ S4 จะสังเกตุเห็นได้ว่าเการเปิดและปิดสวิตช์วงจรนั้นจะปิดและเปิดพร้อมกันเป็นคู่ๆ ต่อมามีการพัฒนาโดยใช้มอสเฟตเพื่อการใช้งานที่สะดวกกว่าเดิม

2.3 วงจรควบคุมความเร็วมอเตอร์

2.3.1 วงจรควบคุมความเร็วมอเตอร์

กระแสไฟฟ้าที่ส่งมาจากไมโครคอนโทรลเลอร์จะมีแรงดันและกระแสที่ต่ำเกินกว่าจะนำไปขับมอเตอร์ที่มีกำลังสูงกว่ามากได้ H-Bridge จึงจำเป็นต้องมีวงจรที่ใช้ในการขยายกระแสและแรงดันให้มากพอที่จะนำไปขับมอเตอร์ได้

2.3.2 โครงสร้างแบบ H-bridge

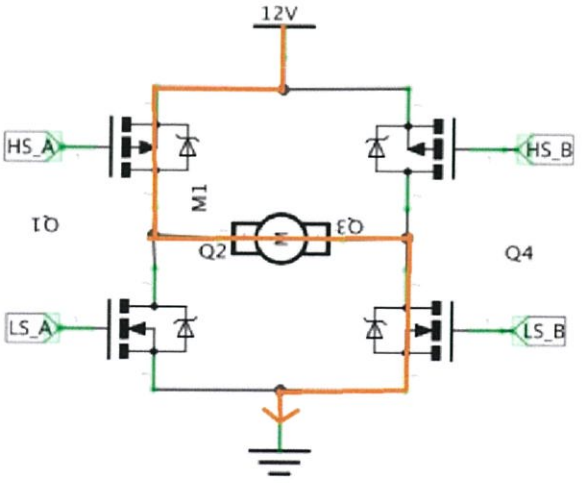


รูปที่ 2.6 ผังวงจร H-bridge

โครงสร้างแบบ H-bridge ประกอบด้วย Switching Element ทำหน้าที่ตัดต่อกระแส เพื่อเปลี่ยนขั้วแรงดันที่จ่ายไปยังมอเตอร์ ส่งผลให้มอเตอร์สามารถหมุนกลับหลังได้ ในที่นี้นิยมใช้เป็นมอสเฟตกับมอเตอร์ ดังรูปที่ 2.6

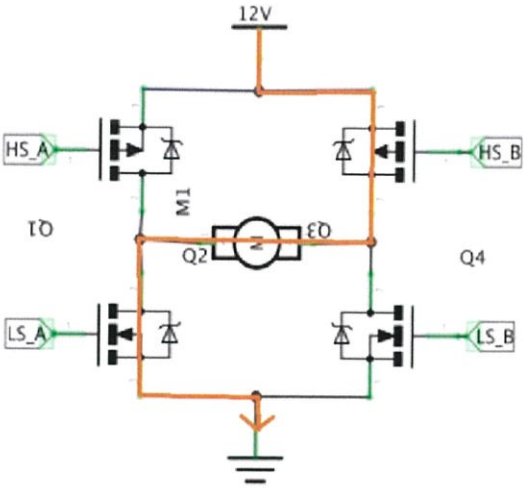
2.3.3 การทำงานของ H-bridge (Static operation)

2.3.3.1 HS_A และ LS_B นำกระแสดังรูปที่ 2.7



รูปที่ 2.7 HS_A และ LS_B นำกระแส

2.3.3.2 HS_B และ LS_A นำกระแส ดังรูปที่ 2.8



รูปที่ 2.8 HS_B และ LS_A นำกระแส

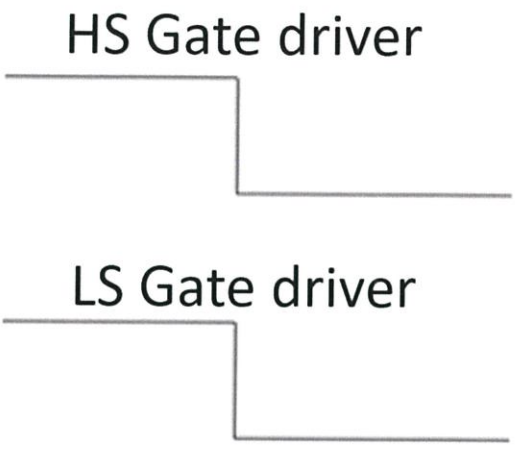
ในรูปที่ 2.7 และรูปที่ 2.8 ไม่ควรให้ HS_A และ HS_B ทำงานพร้อมกันเนื่องจากจะทำให้เกิดการลัดวงจรเช่นเดียวกับ LS_A และ LS_B

2.3.4 มอสเฟต

MOSFET (Metal–Oxide–Semiconductor Field-Effect Transistor) ทำหน้าที่ตัดต่อกระแสเหมือนกับ Switching Element ข้อดีของมอสเฟตคือจ่ายกระแสได้สูง อัตราสูญเสียต่ำ

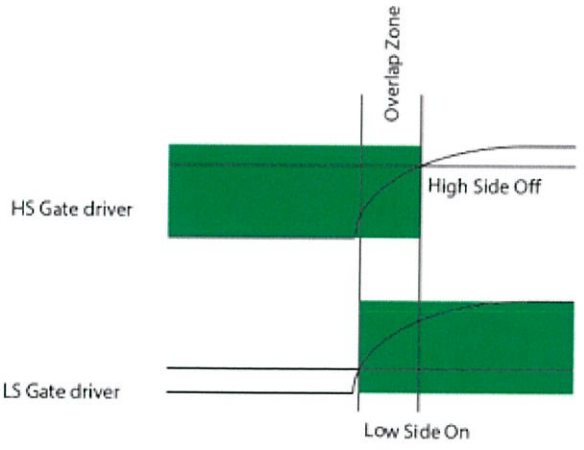
2.3.4.1 มอสเฟตของวงจรขับมอเตอร์ H-Bridge นั้นจะประกอบด้วย P-Channel และ N-Channel ในด้าน High side และ Low side ตามลำดับ ขา Source ของ P-Channel จะจ่ายแรงดันไว้เลี้ยงมอเตอร์ ซึ่งมีแรงดันอยู่ระหว่าง 7-15V ในขณะที่ขา Source ของ N-Channel จะทำหน้าที่เป็น Ground

2.3.4.2 เมื่ออยู่ใน Switching state หากต้องการให้เอาต์พุตของ MOSFET เปลี่ยนจาก 12V เป็น 0V หรือ 0V เป็น 12V สัญญาณที่ควบคุม Gate Driver จะมีรูปร่างดังรูปที่ 2.9

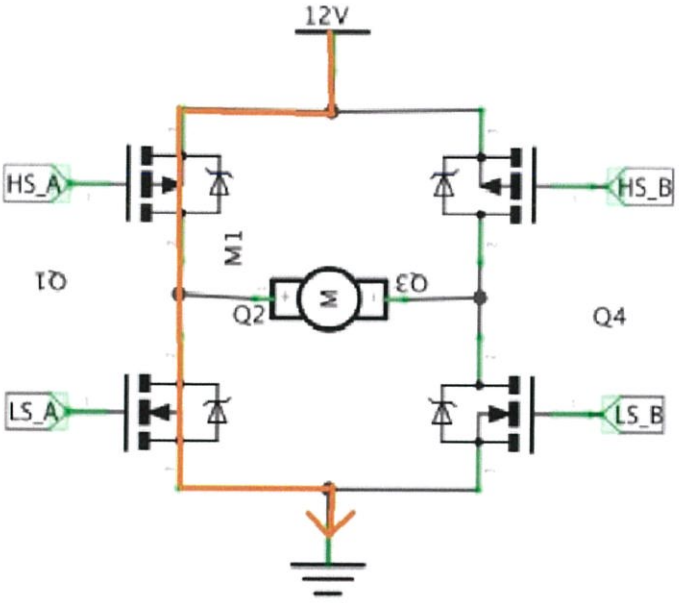


รูปที่ 2.9 สัญญาณที่ควบคุม Gate Driver

ทั้งขา Gate และ Source จะมีโครงสร้างเสมือนตัวเก็บประจุอย่างหนึ่ง หากจ่ายกระแสจะทำให้แรงดัน Gate นั้นเปลี่ยนแปลงไปด้วย ในกรณีที่ HS และ LS ถูกจ่ายไฟหรือทำงานพร้อมกัน จะพบว่าเกิดการซ้อนทับดังรูปที่ 2.10 ทำให้เกิดการลัดวงจรได้



รูปที่ 2.10 การเปลี่ยนแปลงของระดับแรงดัน Gate เมื่อจ่ายกระแส

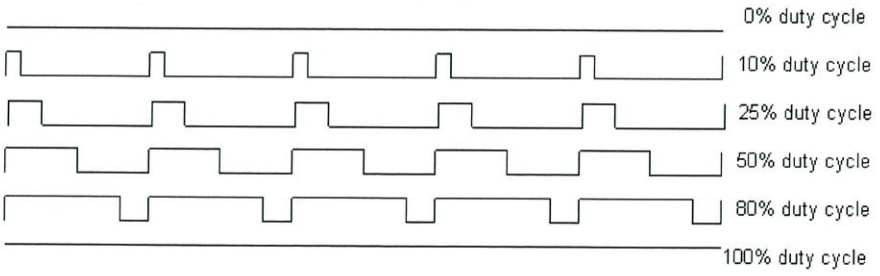


รูปที่ 2.11 การลัดวงจร (Shoot Through)

จากรูปที่ 2.11 จะพบว่ามีการจ่ายไฟไปที่ HS_A และ LS_A พร้อมกัน จึงทำให้เกิดการลัดวงจร หรือเรียกอีกอย่างว่า Shoot Through ซึ่งอาจสร้างความเสียหายแก่ MOSFET ได้ จึงต้องหาวิธีป้องกันไม่ให้เกิดขึ้น โดยใช้วงจร Shoot through Protection เพื่อไม่ให้เกิดการ Overlap หรือลัดวงจรขึ้น ดังนั้นต้องปรับสัญญาณควบคุมไม่ให้เกิดการ Overlap ขึ้น ซึ่งจากสัญญาณที่ต้องการ สามารถใช้วงจรไฟฟ้าช่วยทำให้เกิดการเหลื่อมเฟสได้

2.3.5 การนำมาปรับใช้ร่วมกับ PWM

PWM หรือ Pulse Width Modulation เป็นการมอดูเลตสัญญาณที่มีตัวแปรสำคัญ คือ Duty Cycle โดยจะพิจารณาจากสถานะ High ของสัญญาณเมื่อเทียบกับสถานะ Low ดังรูปที่2.12



รูปที่ 2.12 Duty cycle

สัญญาณ PWM จะเป็นสัญญาณที่นำมาจ่ายให้ H-Bridge เพื่อนำไปควบคุมความเร็วหรือแรงบิดของมอเตอร์ได้ โดยมีวิธีการคำนวนดังนี้

$$Average\ power = \frac{Max\ power \bullet t_{on}}{t} \qquad (2.1)$$

โดย $Average\ power$ คือ กำลังเฉลี่ย

$Max\ power$ คือ กำลังมอเตอร์ เมื่อจ่ายกระแสอย่างต่อเนื่อง

t_{on} คือ ระยะเวลาที่มอเตอร์ทำงานมีกระแสไหลทำให้หมุน

t คือ ระยะเวลาทั้งหมดของ PWM

2.3.6 วงจรขับมอเตอร์กระแสตรง SE-HB40-1

มีรายละเอียดทางเทคนิคดังต่อไปนี้

2.3.6.1 Output: Single motor driver

- Motor DC Supply 12-24 V 40A (Max.)
- Full-Complementary Power MOSFET Driver

With ultra-fast reverse recovery protection diodes

2.3.6.2 Input:

- Full Opto-isolated ground and input interface signals
- Input Signal : 3-5V / 8 mA TTL – Level

2.3.6.3 Drive Mode: independently with:

- Break / Free run Stop
- Direction Control
- Speed Control (PWM Drives)

2.3.6.4 PWM Frequency: 400 Hz - 2000 Hz

2.3.6.5 PWM Duty Cycle Range: 0 – 100%

ตารางที่ 2.1 การทำงานของวงจร H-Bridge ในการควบคุมมอเตอร์

EN/PWM	IN1	IN2	การทำงานของมอเตอร์
GND	X	X	Free Run Stop
5V / PWM	GND	5V	หมุนเดินหน้า
5V / PWM	5V	GND	หมุนกลับทาง
5V	5V	5V	Fast Stop / Break
5V	GND	GND	Fast Stop / Break

เก้าอี้เข็นที่ใช้มอเตอร์ไฟฟ้ากระแสตรงในการขับเคลื่อนส่วนใหญ่จะใช้ก้านควบคุมในการควบคุมทิศทางในการเคลื่อนที่ แต่สำหรับปริญญานิพนธ์นี้ซึ่งนำเสนอเก้าอี้เข็นที่สามารถใช้การเอียงศีรษะเพื่อควบคุมทิศทางการเคลื่อนที่ โดยใช้เซนเซอร์ตรวจวัดความเร่งในการตรวจวัดมุมเอียงของศีรษะของผู้ใช้งาน เพื่อนำไปประมวลผลเพื่อระบุทิศทางการเคลื่อนที่ของเก้าอี้เข็นต่อไป เซนเซอร์ความเร่งเชิงเส้นนั้นจะมีหลักการการทำงานดังต่อไปนี้

2.4 เซนเซอร์ตรวจวัดความเร่ง

2.4.1 MEMS

ระบบเครื่องกลไฟฟ้าจุลภาค (MEMS : Micro Electro Mechanical Systems) หมายถึง ระบบขนาดเล็กตั้งแต่ 1 ไมโครเมตรถึง 1 มิลลิเมตร โดยผลิตขึ้นด้วยเทคโนโลยีการสร้างระดับไมครอน (Micro fabrication) ซึ่งเป็นกระบวนการเดียวกันกับกระบวนการที่ใช้ในการผลิต IC (Integrated Circuit) และได้นำมาประยุกต์ใช้ในการสร้างระบบเครื่องกลไฟฟ้าจุลภาคนี้ โดยประกอบด้วยสองส่วนหลัก ได้แก่ ส่วนที่เป็นระบบเครื่องกลและส่วนที่เป็นอิเล็กทรอนิกส์ โดยส่วนที่เป็นเครื่องกลจะเคลื่อนไหวหรือเคลื่อนที่เพื่อปฏิบัติหน้าที่ใดหน้าที่หนึ่ง และระบบอิเล็กทรอนิกส์จะควบคุมการเคลื่อนไหวของระบบเครื่องกล หรือเก็บข้อมูลที่ได้จากการเคลื่อนไหวหรือเคลื่อนที่ของระบบเครื่องกล โดยทั่วไปแล้วเทคโนโลยีระบบเครื่องกลไฟฟ้าจุลภาคจะหมายรวมถึง เทคโนโลยีที่เกี่ยวข้องกับวัสดุหรือกระบวนการที่ใช้ในการสร้างชิ้นส่วน MEMS หรือการประกอบรวมชิ้นส่วน MEMS ตลอดจนการประยุกต์ใช้อุปกรณ์ที่เป็น MEMS นอกจากนี้ ระบบขนาดเล็ก (Microsystem) จำพวก Microchemical reactor หรือ Microthermal system และ Smart material ซึ่งถูกใช้ในลักษณะของเครื่องตรวจจับ (sensor) หรือแหล่งกำเนิดพลังงาน (power source) จะจัดเป็น

MEMS ด้วยเช่นกัน ตัวอย่างของการนำระบบ MEMS มาใช้งานแพร่หลายในปัจจุบัน ได้แก่ MEMS Accelerometer และ Gyroscope

2.4.2 หลักการเบื้องต้นของเซนเซอร์วัดความเร่ง

จากกฎข้อที่สองของนิวตันกล่าวว่าความเร่งของวัตถุเป็นสัดส่วนโดยตรงและมีทิศทางเดียวกับแรงสุทธิที่กระทำต่อวัตถุและเป็นสัดส่วนผกผันกับมวลของวัตถุนั้น

$$a = \frac{F}{m} \tag{2.2}$$

เมื่อ a คือความเร่ง ($\frac{m}{s^2}$), F คือแรง (N) และ m คือมวล (kg)

โดยที่ความเร่งนั้นจะสร้างแรงที่สามารถตรวจจับได้จากกลไกการตรวจจับแรงกดของเครื่องวัดความเร่งเชิงเส้น ดังนั้นแรงจึงเป็นสิ่งที่เซนเซอร์วัดความเร่งได้ไม่ใช่ความเร่งจริง โดยทั่วไปจะวัดความเร่งทางอ้อมผ่านแรงที่ทำกับแกนหนึ่งของเซนเซอร์วัดความเร่ง เซนเซอร์วัดความเร่งนั้นเป็นอุปกรณ์ไฟฟ้าเชิงกล ประกอบไปด้วย สปริงและช่องสำหรับกลไกที่สร้างด้วยเทคโนโลยีการสังเคราะห์ระดับไมครอน ในการวัดความเร่งจะทำโดยการตรวจจับการเคลื่อนที่ของมวลที่สัมพันธ์กับอิเล็กโทรด

2.4.3 กลไกการตรวจวัดความเร่งของเซนเซอร์วัดความเร่งเชิงเส้น

วิธีการทั่วไปที่ใช้ในคือการตรวจวัดความจุไฟฟ้า ซึ่งเกี่ยวข้องกับการเปลี่ยนแปลงความจุที่เป็นไปตามแผ่นเพลทที่เคลื่อนที่ วิธีการนี้ถือเป็นวิธีที่มีความแม่นยำ มีความเสถียร มีการกระจายพลังงานต่ำและมีโครงสร้างที่เรียบง่ายในการสร้าง จึงทำให้ไม่เกิดเสียงดังและไม่มีการแปรเปลี่ยนไปตามอุณหภูมิ ซึ่งแบนด์วิดท์สำหรับเครื่องวัดความเร่งชนิดนี้มีเพียงไม่กี่ร้อยเฮิรตซ์เท่านั้น เนื่องด้วยโครงสร้างที่เป็นรูปทรงเรขาคณิต(สปริง) และช่องอากาศภายในที่เสมือนทำหน้าที่เป็นตัวกันกระแทก

$$C = \frac{\varepsilon_0 \varepsilon_r A}{D} \tag{2.3}$$

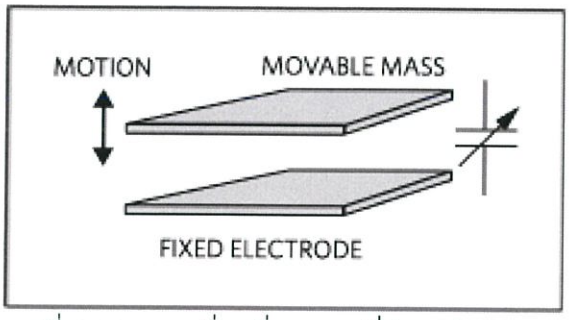
โดย C คือ ค่าความจุ

ε_0 คือ ค่าคงที่ไดอิเล็กทริกของสุญญากาศ

ε_r คือ ค่าค่าคงที่ไดอิเล็กทริกสัมพัทธ์

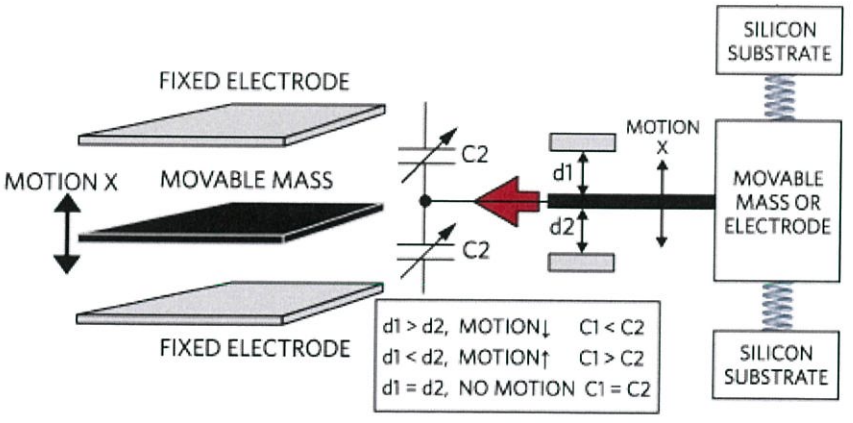
A คือ พื้นที่หน้าตัดของสารตัวนำที่เป็นแผ่นเพลท

D คือ ระยะห่างระหว่างแผ่นเพลททั้งสอง



รูปที่ 2.13 การเคลื่อนที่ของมวลที่ส่งผลต่อความจุ

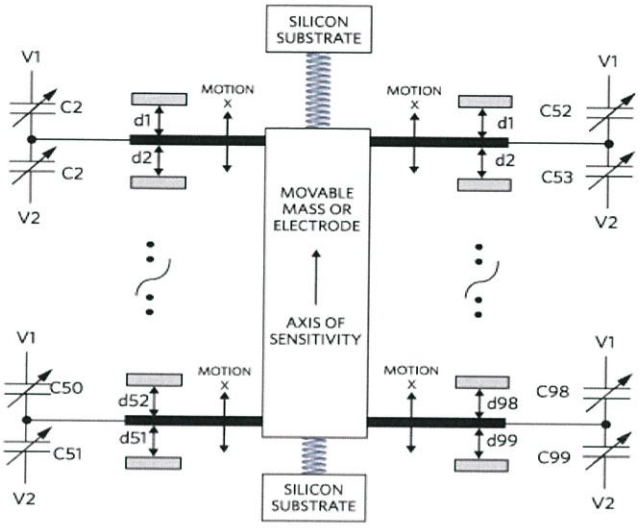
สำหรับการตรวจวัดความจุที่ใช้นั้นสามารถจัดได้เป็นแบบด้านเดียวหรือแบบคู่ผลต่าง จากรูปที่ 2.13 แสดงการตรวจวัดความจุแบบคู่ผลต่าง ประกอบด้วยเพลทผิวเรียบที่สามารถเคลื่อนย้ายได้ ซึ่งวางอยู่พร้อมกับสปริงเชิงกลระหว่างเพลทผิวซิลิคอนหรืออิเล็กโทรดสองแผ่น เห็นได้ชัดว่าการเคลื่อนที่ของแผ่นเพลทสัมพันธ์กับขั้วไฟฟ้าคงที่ (d1 และ d2) และทำให้เกิดการเปลี่ยนแปลงความจุ (C1 และ C2) โดยการคำนวณความแตกต่างระหว่าง C2 และ C1 สามารถหาได้จากการเคลื่อนที่และทิศทางของแผ่นเพลท



รูปที่ 2.14 ความเร่งที่เกี่ยวข้องกับมวลเคลื่อนที่เดี่ยว

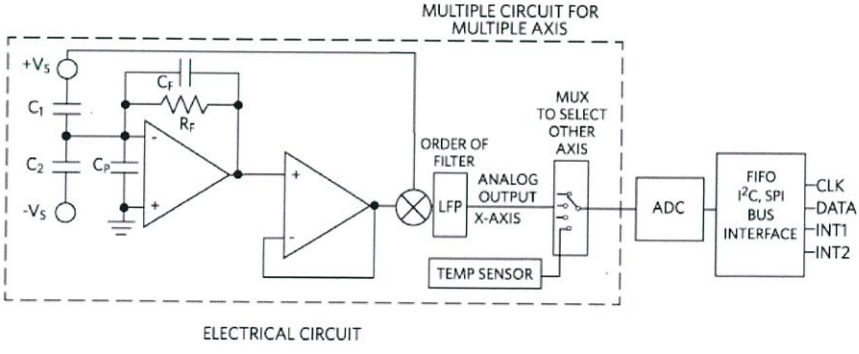
การเคลื่อนที่ของมวลที่เคลื่อนที่ได้เกิดจากการเร่งความเร็วและทำให้เกิดการเปลี่ยนแปลงความจุที่เหมาะสมในการตรวจวัดความเร่ง ในการใช้อิเล็กโทรดทั้งแบบเคลื่อนย้ายได้และแบบคงที่ โดยเชื่อมต่อกันทั้งหมดในรูปแบบขนาน ซึ่งช่วยให้สามารถเปลี่ยนความจุได้มากขึ้นจนสามารถตรวจจับได้อย่างแม่นยำมากขึ้นและทำให้ความสามารถในการรับรู้มีความเป็นไปได้มากขึ้น จากรูปที่ 2.15 แสดงให้เห็นถึงการเชื่อมต่อแบบขนานจำนวนมากขึ้น โดยที่แรงทำให้เกิดการกระจัด

ของมวล ทำให้เกิดการเปลี่ยนแปลงค่าความจุของตัวเก็บประจุ การวางอิเลคโทรดหลายตัวแบบขนานจะช่วยให้มีขนาดใหญ่ขึ้น ซึ่งจะสามารถตรวจพบได้ง่ายขึ้น



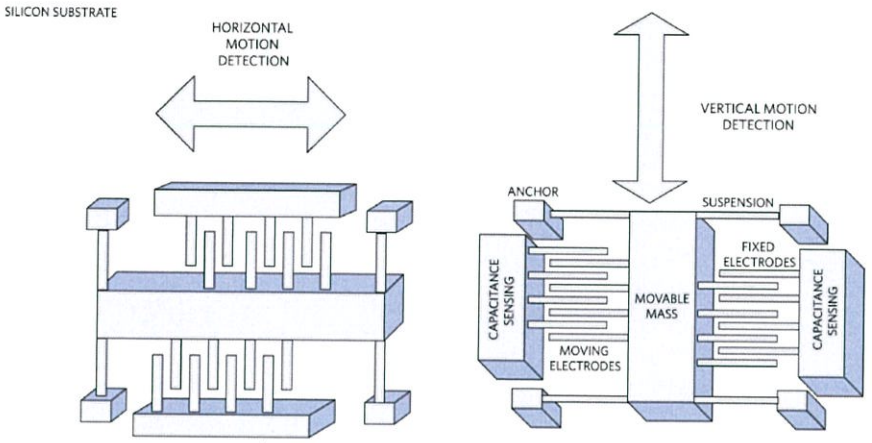
รูปที่ 2.15 ความเร่งที่เกี่ยวข้องกับมวลเคลื่อนที่หลายๆตัว

แรงดันไฟฟ้าจะผ่านการขยายสัญญาณ และกรองเอาสัญญาณรบกวนออก จากนั้นจะถูกแปลงเป็นดิจิตอลโดยใช้ ADC และส่งผ่านไปยังบัฟเฟอร์ FIFO ที่แปลงสัญญาณอนุกรมเป็นกระแสข้อมูลแบบคู่ขนาน สตรีมข้อมูลแบบขนานนั้นสามารถแปลงได้โดยใช้โปรโตคอลแบบอนุกรม เช่น I2C หรือ SPI ก่อนที่จะถูกส่งไปยังโฮสต์เพื่อประมวลผลต่อไป ดังรูปที่ 2.16



รูปที่ 2.16 บล็อกไดอะแกรมของเครื่องวัดความเร่ง

2.4.4 การวัดความเร่งแบบหลายแกน



รูปที่ 2.17 การวัดความเร่งแบบ 2 แกน

จากรูปที่ 2.17 แสดงการวัดความเร่งแบบ 2 แกนโดยการวางเซนเซอร์วัดความเร่งแบบแกนเดี่ยวสองแกนตั้งฉากกัน หรือจะใช้เพียงเซนเซอร์วัดความเร่งที่สามารถวัดได้สองแกนเพียงตัวเดียว

2.4.5 การเลือกเซนเซอร์วัดความเร่ง

2.3.5.1 แบนด์วิธของเซ็นเซอร์ ระบุช่วงของความถี่การสั่นสะเทือนที่เซนเซอร์ตอบสนองหรือความถี่ในการอ่านความเร่งที่เชื่อถือได้ โดยปกตินั้นร่างกายมนุษย์ไม่สามารถเคลื่อนไหวได้ไกลเกินกว่าช่วง 10 Hz ถึง 12 Hz ด้วยเหตุนี้แบนด์วิดธ์ของเซนเซอร์ในช่วง 40 Hz ถึง 60 Hz จึงเพียงพอสำหรับการตรวจจับการเอียงหรือการเคลื่อนไหวของมนุษย์

2.3.5.2 ความไว (mV/g หรือ LSB/g) เป็นตัวบ่งชี้ปริมาณการเปลี่ยนแปลงของสัญญาณเอาท์พุทสำหรับการเปลี่ยนแปลงความเร่งที่กำหนด เซนเซอร์ความเร่งที่มีความไวละเอียดจะมีความแม่นยำสูง

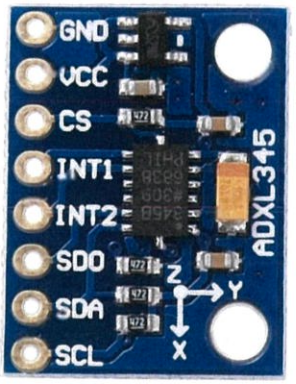
2.3.5.3 ความหนาแน่นของแรงดันไฟฟ้า ความเร่งที่เปลี่ยนแปลงไปที่เซนเซอร์สามารถวัดได้ยิ่งมีความไว ความแม่นยำในการวัดก็จะลดลง สัญญาณรบกวนจึงเป็นสิ่งที่มีผลต่อประสิทธิภาพของเซนเซอร์อย่างมาก เมื่อให้เซนเซอร์ทำงานด้วยเงื่อนไขค่า g ที่ต่ำที่ให้สัญญาณเอาต์พุตน้อย

1) ผลตอบสนองทางความถี่ เป็นการระบุช่วงความถี่และช่วงการชดเชยสำหรับเซนเซอร์ในการตรวจจับการเคลื่อนไหว โดยความคลาดเคลื่อนของช่วงที่ระบุช่วยให้

ผู้ใช้สามารถคำนวณความไวของอุปกรณ์ที่เบี่ยงเบนไปจากความไวอ้างอิงที่ความถี่ใดๆ ภายในช่วงความถี่ที่ระบุ

2) ช่วงความเร่งที่วัดได้ (g) เป็นช่วงระหว่างขนาดความเร่งต่ำสุดและสูงสุดที่เซนเซอร์สามารถตรวจวัดได้ก่อนที่สัญญาณจะผิดเพี้ยน

2.4.6 ADXL345 Digital Accelerometer



รูปที่ 2.18 ADXL345

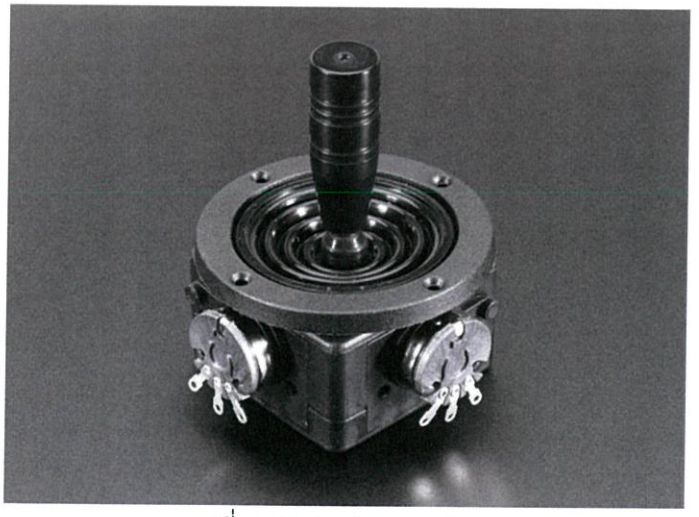
ADXL345 เป็นเซนเซอร์วัดความเร่งแบบ 3 แกน มีขนาดเล็กเหมาะสมกับการใช้งานในอุปกรณ์ขนาดพกพา โดยมีความละเอียดสูงถึง 13 บิต และสามารถวัดความเร่งได้สูงสุด ± 16 g ข้อมูลเอาต์พุตอยู่ในรูปข้อมูลดิจิตอลสูงสุด 16 บิต และสามารถเข้าถึงได้ผ่านการเชื่อมต่อแบบ SPI หรือ I2C ได้ สำหรับ ADXL345 สามารถวัดความเร่งสถิตย์อันเนื่องจากแรงโน้มถ่วง รวมทั้งความเร่งแบบจลน์อันเกิดจากการเคลื่อนไหวหรือการกระแทก โดยมีความละเอียดสูงถึง 3.9 mg/LSB ช่วยให้สามารถวัดการเปลี่ยนแปลงความเอียงได้น้อยกว่า 1.0 °

ตารางที่ 2.2 คำอธิบายขาการเชื่อมต่อ ADXL345

ขาเชื่อมต่อ	คำอธิบาย
GND	GND
VCC	แรงดันไฟเลี้ยง 3.3 V
CS	Chip Select
INT1	Interrupt 1 Output
INT2	Interrupt 2 Output
SDO	Serial Data Output
SDA	Serial Data (I2C)
SCL	Serial Communications Clock

2.5 ก้านควบคุม (Analog Joystick)

ก้านควบคุมประกอบด้วยตัวต้านทานแบบปรับค่าได้จำนวนสองตัวในแนวแกน X และแกน Y ทั้งสองฝั่งจะมีสปริงอยู่ที่แกนของคันโยกเพื่อดึงก้านควบคุมกลับมาที่ตำแหน่งกึ่งกลาง ที่ตัวต้านทานทั้งสองตัวนั้นขาข้างหนึ่งจะต้อกับแรงดันอ้างอิง ส่วนอีกขาจะต่อไปยังกราวด์ ขากลางของทั้งคู่จะต่อไปยังบอร์ดไมโครคอนโทรลเลอร์เพื่ออ่านค่าแรงดันอนาล็อกเพื่อนำไปใช้ควบคุมทิศทาง

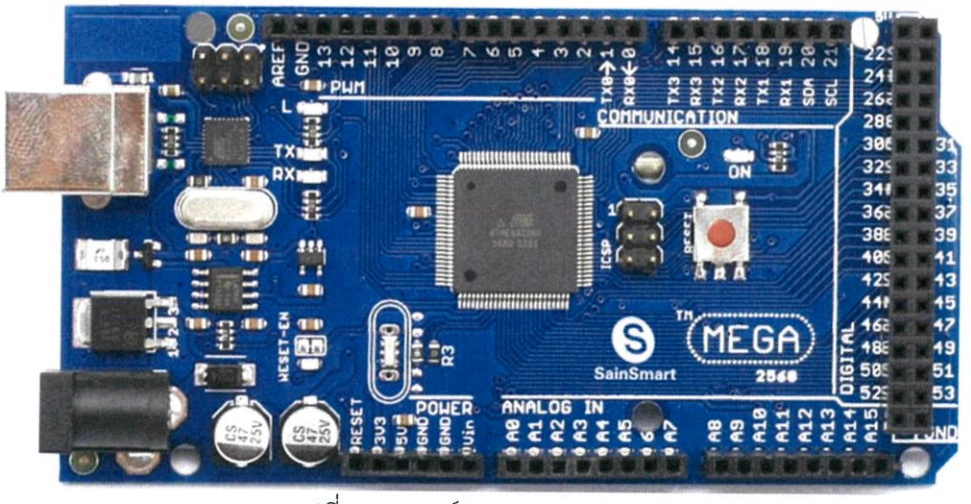


รูปที่ 2.19 Analog Joystick

สำหรับระบบของเก้าอี้เข็นนั้นจะใช้ไมโครคอนโทรลเลอร์ในการประมวลผลทิศทางจากเซนเซอร์และควบคุมการขับมอเตอร์ในการขับเคลื่อนเก้าอี้เข็น โดยจะใช้ไมโครคอนโทรเลอร์ Arduino สำหรับการประมวลผลทิศทางและความคุมเป็นบอร์ดหลัก และจะใช้ NodeMCU ในการเชื่อมต่อระบบให้สามารถสื่อสารกับระบบฐานข้อมูลและแอปพลิเคชัน

2.6 บอร์ดไมโครคอนโทรลเลอร์ชนิดอาร์ดูโน่ (Arduino)

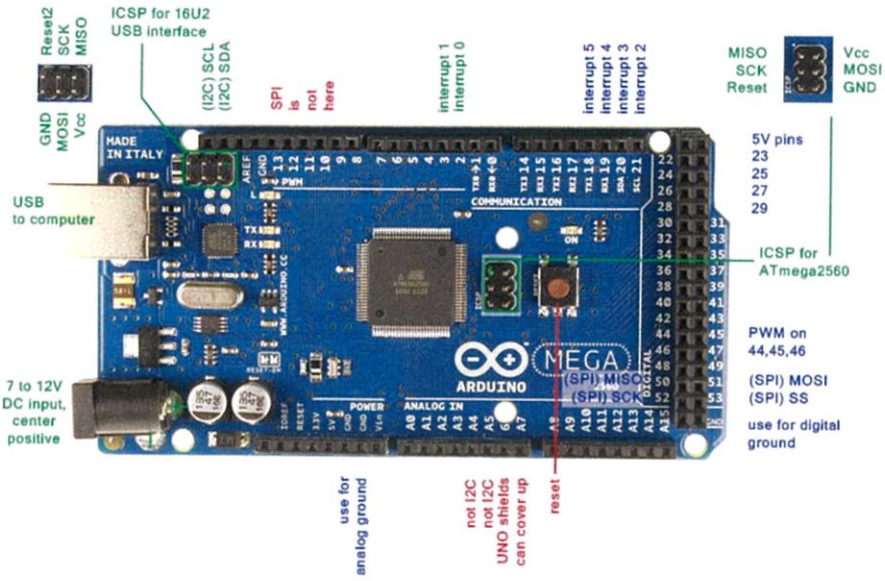
เป็นบอร์ดไมโครคอนโทรเลอร์ตระกูล AVR เป็นบอร์ดที่เหมาะสำหรับผู้เริ่มต้นใช้งาน เนื่องจากสามารถศึกษาและใช้งานได้ง่าย สามารถนำไปพัฒนาได้อย่างหลากหลาย ประกอบไปด้วย ขา Input และ Output ซึ่งสามารถต่อกับอุปกรณ์,บอร์ด หรือเซนเซอร์อื่นๆได้ ซึ่งมีหลายรุ่นเพื่อรองรับความต้องการของผู้ใช้งานที่แตกต่างกัน รูปที่ 2.20 แสดงตัวอย่างบอร์ด Arduino MEGA



รูปที่ 2.20 บอร์ด Arduino MEGA

2.6.1 คุณสมบัติของบอร์ด Arduino MEGA

บอร์ด Arduino MEGA ใช้ไฟเลี้ยง 7-12V โดยใช้ชิพ ATmaga2560 ที่มีหน่วยความจำแฟลช 256 KB แรม 8 KB ซึ่งมีขาอินพุตและขาเอาต์พุต 54 ขา สามารถใช้งานเป็นขาพอร์ต PWM ได้ 15 ขา, ขาอนาล็อกอินพุต 16 ขา และ UART 4 ชุด โดยความถี่คริสตรัลบนบอร์ดคือ 16 MHz เชื่อมต่อข้อมูลระหว่างคอมพิวเตอร์ผ่านพอร์ต USB บนบอร์ดได้โดยตรง อีกทั้งรูปแบบการออกแบบยังออกแบบให้รองรับการใช้งานร่วมกับโมดูลต่างๆได้โดยตรง โดยรองรับการพัฒนาโปรแกรมบนแพลตฟอร์ม Arduino อย่างเต็มรูปแบบArduino MEGA สามารถใช้งานได้โดยสั่งงานผ่านภาษา C++ รองรับทั้งระบบปฏิบัติการ Windows, Linux และ Mac OS



รูปที่ 2.21 การใช้งานขาพอร์ตอินพุตและเอาต์พุตของบอร์ด Arduino MEGA

2.6.2 ขาพอร์ตอินพุตและเอาต์พุตของบอร์ด Arduino MEGA

การใช้งานขาพอร์ตอินพุตและเอาต์พุตของบอร์ด Arduino MEGA แสดงดังรูปที่ 2.21 โดยมีรายละเอียดดังต่อไปนี้

- USBPort: ใช้อัพโหลดโปรแกรมจากคอมพิวเตอร์ไปยังบอร์ด รวมไปถึงไฟเลี้ยงด้วย

- Reset Button: ปุ่มรีเซ็ตการทำงานของบอร์ดเมื่อต้องการให้บอร์ดเริ่มทำงานใหม่

- ICSP Port: ใช้สำหรับโปรแกรม Visual Com port บน Atmega16U2 และ ใช้สำหรับโปรแกรม Bootloader

- I/O Port D0-D53: Digital I/O ตั้งแต่ขา D0 ถึง D53 นอกจากนี้ บาง Pin จะทำหน้าที่อื่นๆ เพิ่มเติมด้วย เช่น Pin0,1 เป็นขา Tx และ Rx แบบ Serial หรือ Pin44, 45 และ 46 เป็นขาสำหรับการใช้งาน PWM

- MCU Atmega328: เป็นหน่วยประมวลผลของบอร์ด

- I/OPort A0-A15: สามารถรับได้ทั้ง Digital input และ Output รวมไปถึงสัญญาณอนาล็อก

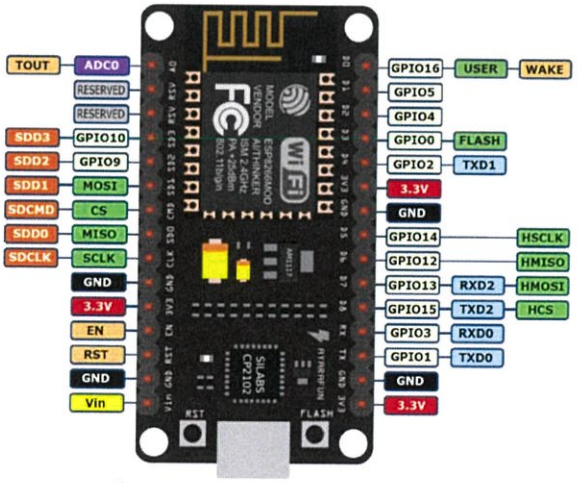
- Power Port: ประกอบด้วยขาแรงดันไฟเลี้ยง +3.3 V, +5V, GND, Vin เป็นพอร์ตแรงดันไฟเลี้ยงของบอร์ด เมื่อต้องการจ่ายไฟให้กับวงจรภายนอก

- Power Jack: รับไฟจาก Adapter 7-12 V

- MCU ATmega16U2: เป็นหน่วยประมวลผลที่ทำหน้าที่เป็น USB to Serial โดย ATmega2560 จะติดต่อกับคอมพิวเตอร์ผ่าน ATmega16U2

2.7 NodeMCU

NodeMCU ดังรูปที่ 2.22 ประกอบไปด้วย ชิป ไมโครคอนโทรลเลอร์และโมดูล WiFi สามารถนำไปใช้งานแทนไมโครคอนโทรลเลอร์ได้ NodeMCU มีโมดูล ESP8266 เป็นชื่อเรียกของชิปของโมดูลสำหรับติดต่อสื่อสารบนมาตรฐาน WiFi ทำงานที่ระดับแรงดันไฟฟ้า 3.0-3.6V ที่กระแสไฟฟ้าโดยเฉลี่ย 80mA โดยในการนำไปใช่ร่วมกับเซ็นเซอร์ชนิดอื่นๆจะใช้แรงดัน 5V จึงต้องใช้วงจรแบ่งแรงดันมาช่วยเพื่อไม่ให้โมดูลพังเสียหาย กระแสที่โมดูลสามารถใช้งานสูงสุดคือ 200mA ความถี่ 40MHz ทำให้เมื่อนำไปใช้งานอุปกรณ์ที่ทำงานรวดเร็วตามความถี่ เช่น LCD การแสดงผลข้อมูลจะรวดเร็วกว่าบอร์ด Arduino โดย NodeMCU สามารถรองรับคำสั่ง deep sleep เพื่อการประหยัดพลังงาน ซึ่งจะใช้กระแสน้อยกว่า 10 mA และสามารเรียกกลับมาใช้งานส่งข้อมูลด้วยเวลาน้อยกกว่า 2 ms ทั้งนี้ยังมี Low Power MCU 32 บิต จึงใช้เขียนโปรแกรมเพื่อสั่งงานได้ NodeMCU มีวงจร Analog Digital Converter ที่ทำให้สามารถอ่านค่าจากสัญญาณ Analog และแปลงเป็นข้อมูลดิจิตอลที่ความละเอียด 10 บิต ทำงานได้ที่อุณหภูมิ -40 ถึง 125 องศาเซลเซียส NodeMCU จะติดต่อกับ WI-FI แบบ Serial จึงสามารถเขียนโปรแกรมลงไปในชิป โดยใช้โปรแกรม Arduino IDE คล้ายกับการใช้ Arduino



รูปที่ 2.22 ขาอุปกรณ์ของ NodeMCU

2.7.1 ข้อมูลทางเทคนิคของ NodeMCU

2.7.1.1 โมดูล ESP8266-12E ที่ภายในมีไมโครคอนโทรลเลอร์ 32 บิต หน่วยความจำแบบแฟลช ความจุ 4 MB และโมดูล WiFi ในตัว

2.7.1.2 ชิป CP2102 สำหรับแปลงสัญญาณพอร์ต USB เป็น UART เพื่อเชื่อมต่อคอมพิวเตอร์สำหรับโปรแกรมเฟิร์มแวร์

2.7.1.3 ใช้ไฟเลี้ยงภายนอก +5V มีวงจรควบคุมแรงดันไฟเลี้ยงสำหรับอุปกรณ์ 3.3V กระแสไฟฟ้าสูงสุด 800mA

2.7.1.4 มีขาพอร์ต SPI สำหรับติดต่อกับ SD การ์ด

2.7.1.5 มีสวิตช์ RESET และ Flash สำหรับโปรแกรมเฟิร์มแวร์ใหม่

2.7.1.6 มีขาอินพุตและขาเอาต์พุตดิจิตอล (ระดับแรงดัน 3.3V) รวม 16 ขา

2.7.1.7 มีอินพุตอนาล็อค 1 ช่อง สามารถรับแรงดันไฟตรง 0 ถึง +1Vdc เข้าสู่วงจรแปลงสัญญาณอนาล็อคเป็นข้อมูลดิจิตอลที่มีความละเอียด 10 บิต

2.7.1.8 มีแผงวงจรสำหรับต่ออุปกรณ์หรือนำไปติดตั้งบนแผงวงจรประยุกต์ที่ออกแบบขึ้นเองได้สะดวก

2.7.2 การพัฒนาโปรแกรมบน NodeMCU

2.7.2.1 AT Command

โดยทั่วไปแล้วบอร์ดส่วนใหญ่จะมีเฟิร์มแวร์ AT Command ซึ่งจะช่วยให้ใช้งานอุปกรณ์เหล่านี้เป็นโมเด็มไร้สายที่ควบคุมผ่านพอร์ตอนุกรม วิธีนี้ถือเป็นวิธีหลักในการใช้ชิป ESP8266 เพื่อให้สามารถเชื่อมต่อ Wi-Fi กับไมโครคอนโทรลเลอร์อื่น ๆ ได้ง่ายเช่นเดียวกับบอร์ด Arduino

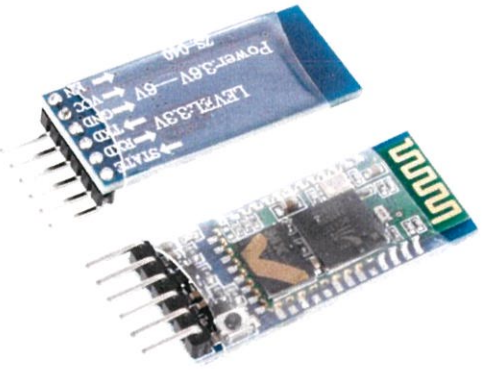
2.7.2.2 NodeMCU Development toolkit

เป็นเฟิร์มแวร์ eLua สำหรับ ESP SoC โดย Espressif เฟิร์มแวร์ ซึ่งอ้างอิงกับ Espressif NON-OS SDK 2.0.0 ผ่านคำสั่ง NodeMCU toolkit จะถูกส่งไปยัง ESP8266 ผ่านอินเตอร์เฟส Serial UART โดยที่เฟิร์มแวร์ของ NodeMCU ช่วยให้สามารถบันทึกแอปพลิเคชันเป็นสคริปต์ในหน่วยความจำแฟลชของ ESP8266 และสั่งให้เรียกใช้โปรแกรมแอปพลิเคชันทุกครั้งที่เริ่มระบบใหม่

2.7.2.3 Arduino IDE

เป็นการสนับสนุนชิป ESP8266 ให้สามารถใช้งานร่วมกับ Arduino ทำให้สามารถพัฒนาโปรแกรมด้วยการใช้ฟังก์ชันและไลบรารีของ Arduino โดยอัพโหลดโปรแกรมให้กับโมดูล ESP8266 และเรียกใช้งานได้โดยตรงเมื่อเปิดใช้งาน ESP8266 โดยปกติแล้ว Arduino จะประกอบไปด้วยไลบรารีที่มีประโยชน์มากมายในการประยุกต์ใช้ในการสื่อสารแบบ WiFi ไม่ว่าจะเป็นการใช้ระบบไฟล์ในหน่วยความจำแฟลชทำงานกับการ์ด SD, อุปกรณ์ SPI และ I2C ที่สนับสนุนโมดูล ESP8266 จะถูกเพิ่มลงใน Arduino IDE (เวอร์ชั่น 1.6.4 และใหม่กว่า)

2.8 โมดูลสื่อสารไร้สายบลูทูธ HC-05



รูปที่ 2.23 โมดูลบลูทูธ HC-05

HC-05 โมดูลบลูทูธดังรูปที่ 2.23 เป็นโมดูลไร้สายที่ใช้สื่อสารด้วย Bluetooth Serial Port Protocol โดย Serial Port เป็นมาตรฐาน Bluetooth V2.0+EDR (Enhance Data Rate) มีความเร็วในการมอดูเลตรับส่งข้อมูล 3 Mbps ใช้งานในความถี่ 2.4 GHz รองรับ Baud Rate: 9600, 19200, 38400, 57600, 115200, 230400 และ 460800 โดยในค่าปกติจะมี Baud Rate 38400, 8 Data Bits, Stop Bits 1 และไม่มี Parity Bit สามารถใช้เชื่อมต่อกับอุปกรณ์บลูทูธอื่นๆได้ โดยโมดูล HC-05 มีการกำหนดคุณสมบัติขาเชื่อมต่อดังตารางที่ 2.3

ตารางที่ 2.3 คุณสมบัติขาเชื่อมต่อของโมดูลบลูทูธ HC-05

ขาเชื่อมต่อของโมดูล HC-05	หน้าที่ของขาเชื่อมต่อ
STATE	สถานะของโมดูล
RXD	เชื่อมกับขาส่งของไมโครคอนโทรลเลอร์
TXD	เชื่อมกับขารับของไมโครคอนโทรลเลอร์
GND	GND
VCC	เชื่อมกับแรงดัน 5 V หรือ 3.3 V
EN	เชื่อมกับแรงดัน 5 V เมื่อต้องการเข้า AT Mode

2.9 โมดูลเซนเซอร์ตรวจวัดระยะอัลตราโซนิค

รูปที่ 2.24 โมดูลเซนเซอร์ตรวจวัดระยะอัลตราโซนิค HC-SR04

HC-SR04 เป็นเซนเซอร์สำหรับตรวจจับวัตถุและวัดระยะทางแบบไม่สัมผัส โดยใช้สัญญาณเสียงอัลตราโซนิค ซึ่งเป็นคลื่นเสียงความถี่สูงเกินกว่าการได้ยินของมนุษย์ สามารถวัดระยะได้ตั้งแต่ 2 – 400 เซนติเมตร สามารถต่อใช้งานกับไมโครคอนโทรลเลอร์ได้ง่าย ใช้พลังงานในการทำงานต่ำ ตารางที่ 2.4 แสดงคุณลักษณะทางไฟฟ้าของโมดูลเซนเซอร์ตรวจวัดระยะอัลตราโซนิค HC-SR04 และตารางที่ 2.5 แสดงการกำหนดขาเชื่อมต่อกับบอร์ดไมโครคอนโทรลเลอร์

ตารางที่ 2.4 คุณลักษณะทางไฟฟ้าของโมดูลเซนเซอร์อัลตราโซนิค HC-SR04

Electrical Parameters	HC-SR04
Operating Voltage	DC-5V
Operating Current	15mA
Operating Frequency	40KHZ
Farthest Range	4m
Nearest Range	2cm
Measuring Angle	15 Degree
Input Trigger Signal	10us TTL pulse
Output Echo Signal	Output TTL level signal, proportional with range
Dimensions	45*20*15mm

ตารางที่ 2.5 การกำหนดขาเชื่อมต่อ HC-SR04 กับบอร์ดไมโครคอนโทรลเลอร์

Pin Symbol	Pin Function Description
VCC	5V power supply
Trig	Trigger Input pin
Echo	Receiver Output pin
GND	Power ground

2.10 โมดูลเสียง Buzzer

รูปที่ 2.25 Buzzer Module

โมดูลเสียง Buzzer เป็นอุปกรณ์สัญญาณเสียง สามารถนำไปประยุกต์ใช้ได้หลากหลาย เช่น อุปกรณ์แจ้งเตือน นาฬิกาจับเวลา เป็นต้น โดยโครงสร้างภายในประกอบไปด้วยทรานส์ดิวเซอร์ในการแปลงสัญญาณไฟฟ้าเป็นสัญญาณเสียง สามารถใช้ไฟเลี้ยงจากบอร์ดไมโครคอนโทรลเลอร์ได้โดยตรง ในตารางที่ 2.6 แสดงขาการเชื่อมต่อของโมดูล Buzzer

ตารางที่ 2.6 คำอธิบายขาการเชื่อมต่อของโมดูล Buzzer

ขาการเชื่อมต่อ	คำอธิบาย
VCC	ไฟเลี้ยง 3.3 V – 5 V
I/O	Input
GND	GND

2.11 Firebase Realtime Database

Firebase คือระบบฐานข้อมูลที่ถูกออกแบบมาให้เป็น API และ Cloud Storage สำหรับพัฒนา Realtime Application รองรับหลาย Platform ทั้ง IOS App, Android App, Web App โดยถูกสร้างขึ้นจากคุณสมบัติเสริมที่ว่านักพัฒนาสามารถผสมและจับคู่เพื่อให้พอดีกับความต้องการของตน และมีการพัฒนาให้สามารถ backend จากเก็บข้อมูลอย่างเดียว มาเป็นแพลตฟอร์มครบวงจรสำหรับนักพัฒนาแอปพลิเคชัน รองรับบริการทุกประเภทที่ต้องใช้งานดังต่อไปนี้

- Firebase Analytics บริการวิเคราะห์ข้อมูล โดยใช้เทคโนโลยีมาจาก Google Analytics เปิดให้ใช้ฟรีแบบไม่จำกัดปริมาณข้อมูลใดๆ

- Firebase Cloud Messaging (FCM) เป็นระบบส่งข้อความแจ้งเตือน โดยใช้งานฟรีไม่จำกัดปริมาณข้อความ

- Firebase Storage บริการพื้นที่เก็บข้อมูล เอาไว้เก็บภาพ วิดีโอ หรือไฟล์ขนาดใหญ่จากแอปพลิเคชันของผู้ใช้งานอยู่บน Google Cloud Storage

- Firebase Remote Config ตัวช่วยในการปรับปรุงการตั้งค่าของแอปพลิเคชันสำหรับปรับแต่งค่าต่างๆ ในแอปพลิเคชันจากระยะไกล สามารถใช้ร่วมกับ Firebase Analytics เพื่อกำหนดผู้ใช้งานแยกเป็นกลุ่มๆ ได้

- Firebase Crash Reporting ตัวรายงานการแครชของแอปพลิเคชันที่รองรับทั้ง iOS และ Android

- Firebase Test Lab for Android บริการทดสอบแอปพลิเคชันบนฮาร์ดแวร์จริง

- Firebase Notifications เป็นส่วนสำหรับนักพัฒนา เพื่อส่งข้อความผ่าน FCM ไปยังผู้ใช้ สำหรับเชิญชวนหรือกระตุ้นให้ผู้ใช้งานกลับมาเปิดใช้งานแอปพลิเคชันอีกครั้ง

- Firebase Dynamic Links บริการ URL กลางที่สามารถชี้ทางไปยังเพจต่างๆ แปรผันตามอุปกรณ์หรือคุณสมบัติของผู้ใช้

- Firebase Invites ระบบเชิญชวนมาใช้แอปพลิเคชัน

- Firebase App Indexing ช่วยให้ Google Search ค้นหาเนื้อหาภายในแอปพลิเคชัน

ในงานด้านแอปพลิเคชัน Firebase ครอบคลุมทุกการบริการสำหรับพัฒนา Realtime Application ถือเป็นบริการฐานข้อมูลออนไลน์ ซึ่งแอปพลิเคชันส่วนใหญ่ต้องใช้งานฐานข้อมูลส่วนนี้ จึงถือว่าเป็นตัวกลางการเชื่อมต่อทุกอุปกรณ์เข้าด้วยกันได้ โดยมีจุดเด่นคือสามารถบันทึกข้อมูล ณ เวลาปัจจุบันไว้ได้ ในด้านของ API นั้นFirebase ไม่ได้อ้างอิงการใช้งานไปกับภาษาใดภาษาหนึ่ง โดย Firebase มี API ของหลายภาษาให้เลือกใช้งาน ทั้งภาษา Python, JavaScript และรวมไปถึงใน ESP8266 ที่ใช้ Arduino IDE ด้วย

ฐานข้อมูล Firebase Realtime เป็นฐานข้อมูลแบบ Cloud-hosted ข้อมูลจะถูกเก็บเป็น JSON และซิงโครไนส์ในแบบเรียลไทม์กับทุกไคลเอ็นต์ที่เชื่อมต่อ เมื่อสร้างแอปพลิเคชันข้ามแพลตฟอร์มด้วย SDK ของ iOS, Android และ JavaScript ข้อมูลของผู้ใช้งานทั้งหมดจะถูกอัปเดตและรับค่าโดยใช้ข้อมูลล่าสุด

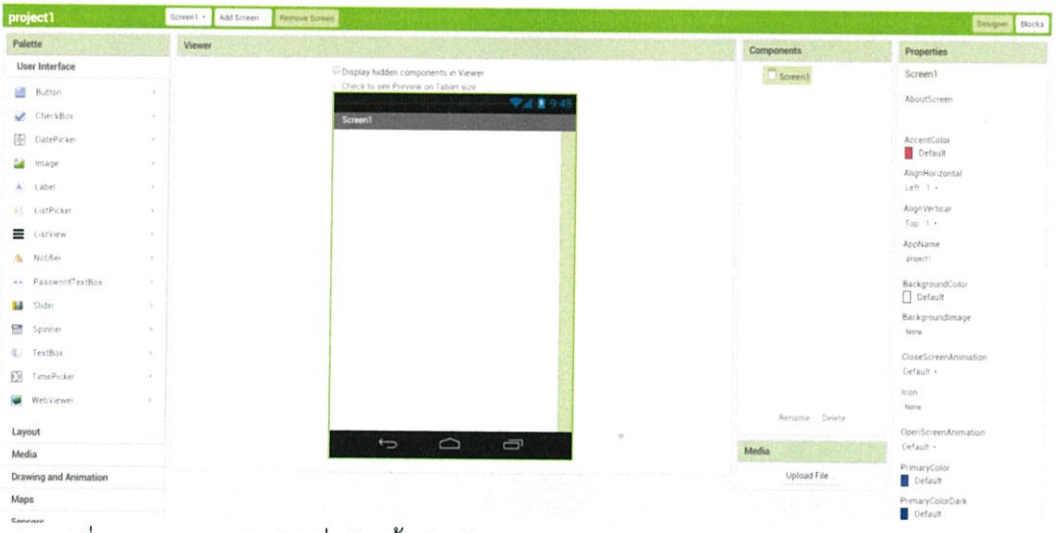
2.12 App Inventor

App Inventor เป็นเครื่องมือที่ใช้สำหรับสร้างแอปพลิเคชันสำหรับสมาร์ทโฟนและแท็บเล็ตที่เป็นระบบปฏิบัติการ Android ซึ่งบริษัท Google ร่วมมือกับ MIT พัฒนาโปรแกรม App inventor ขึ้น โดยลักษณะการเขียนโปรแกรมแบบ Visual Programming คือ เขียนโปรแกรมด้วยการต่อบล็อกคำสั่งและเน้นการออกแบบเพื่อแก้ปัญหา (problem solving) ด้วยการสร้างโปรแกรมที่ผู้เรียนสนใจบนสมาร์ทโฟน

App Inventor สามารถพัฒนาแอปพลิเคชันสำหรับโทรศัพท์ระบบปฏิบัติการ Android ซึ่งทำผ่านการใช้เว็บเบราเซอร์และทดสอบบนโทรศัพท์ที่เชื่อมต่ออยู่กับคอมพิวเตอร์หรือทดสอบบนโทรศัพท์จำลองบนเครื่องคอมพิวเตอร์ โปรเจคที่สร้างทั้งหมดจะถูกจัดเก็บไว้บน

เซิร์ฟเวอร์ของ App Inventor ซึ่งช่วยให้สามารถพัฒนางานต่อที่เครื่องคอมพิวเตอร์เครื่องใดก็ได้เพียงแค่มีการเชื่อมต่อกับระบบอินเทอร์เน็ตไว้เท่านั้น

การสร้างแอปพลิเคชันจะแบ่งการทำงานออกเป็นสองส่วน คือ ส่วนออกแบบ (App Inventor Designer) โดยเลือกส่วนประกอบที่ต้องการสำหรับที่จะให้สร้างแอปพลิเคชัน ส่วนที่สองเป็นส่วนการเขียนโปรแกรม (App Inventor Blocks Editor) ด้วยการต่อบล็อกต่างๆ เข้าด้วยกันเป็นคำสั่ง ซึ่งจะเป็นการกำหนดพฤติกรรมหรือเหตุการณ์ที่เกิดขึ้นกับคอมโพเนนท์ การเขียนโปรแกรมจะเสมือนการต่อชิ้นส่วนตัวต่อจิ๊กซอว์เข้าด้วยกัน ในแต่ละขั้นตอนการสร้างจะสามารถทำการทดสอบได้ทุกขณะและเมื่อสร้างเสร็จสมบูรณ์แล้วจะสามารถนำไปใช้งานบนโทรศัพท์ระบบปฏิบัติการ Android เครื่องใดก็ได้ หรือหากไม่มีโทรศัพท์ระบบปฏิบัติการ Android ก็สามารถที่จะทดสอบได้บนโทรศัพท์จำลองที่ทำงานอยู่บนคอมพิวเตอร์ซึ่งจะมีลักษณะการทำงานเหมือนโทรศัพท์จริงทุกประการ สภาพแวดล้อมในการพัฒนาด้วยโปรแกรม App Inventor นั้น สนับสนุนระบบปฏิบัติการที่หลากหลาย ไม่ว่าจะเป็นระบบปฏิบัติการ Mac OS X, GNU / Linux และระบบปฏิบัติการ Windows และแอพพลิเคชันที่สร้างขึ้นนั้นสามารถติดตั้งและทำงานได้บนโทรศัพท์ระบบปฏิบัติการ Android หลากหลายรุ่นที่เป็นที่นิยมในปัจจุบัน



รูปที่ 2.26 แอปพลิเคชันที่สร้างขึ้นด้วยโปรแกรม App Inventor ในหน้าต่างเว็บบราวเซอร์

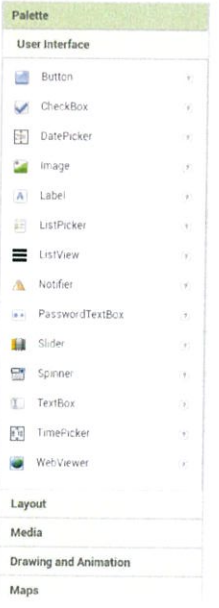
2.12.1 ส่วนออกแบบ (App Inventor Designer)

ในขั้นตอนแรกของการสร้างแอปพลิเคชันด้วย App Inventor เริ่มจากการเลือกคอมโพเนนท์ที่ต้องการและจัดวางลงในส่วนของการออกแบบโดยจะทำผ่านส่วนของการออกแบบ (App

Inventor Designer) ดังที่แสดงในภาพที่ 2.26 แสดงให้เห็นถึงแอปพลิเคชันที่สร้างขึ้นในหน้าต่างเว็บ โดยด้านซ้ายจะเป็นส่วนของคอมโพเนนท์ที่ App Inventor เตรียมไว้ให้จัดเรียงเป็นหมวดหมู่ เช่น ปุ่ม (button) ข้อความ (label) กล่องข้อความ (text box) เป็นต้น ผู้ใช้ทำการเพิ่มคอมโพเนนท์ที่เลือกแล้วลากลงไปวางไว้ในโปรเจค

รูปที่ 2.27 หน้าจอการจัดการโปรเจค (My Projects)

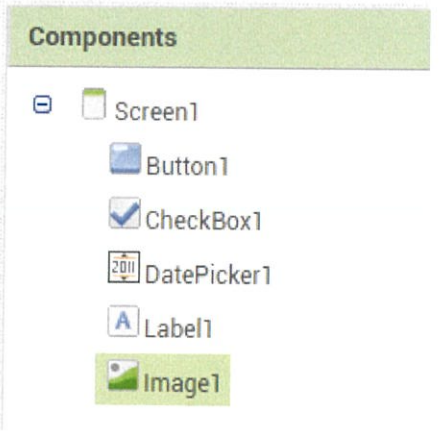
บนหน้าเว็บ App Inventor นั้นจะประกอบด้วยแท็บที่จะปรากฏในส่วนบนของหน้าเว็บซึ่งจะใช้ในการเข้าไปจัดการโปรเจค (My Projects) ส่วนการออกแบบ (Design) ส่วนการเรียนรู้คำสั่ง (Learn) ในหน้าจอการจัดการโปรเจคดังรูปที่ 2.27 จะสามารถเข้าไปจัดการสร้าง ลบ ดาวน์โหลด หรือเลือกโปรเจคที่สร้างและได้ทำการบันทึกไว้เพื่อกลับมาแก้ไขในหน้าจอส่วนการออกแบบได้



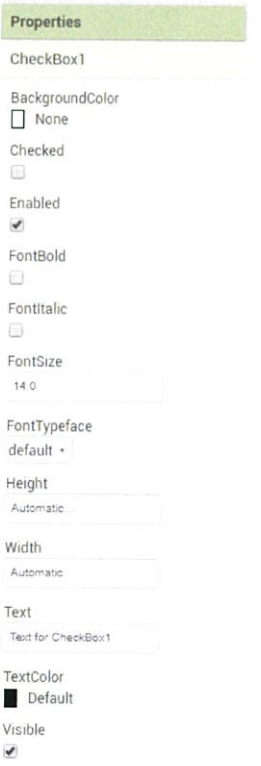
รูปที่ 2.28 หน้าจอส่วนคอมโพเนนท์

รูปที่ 2.29 หน้าจอการออกแบบ (Viewer)

ในส่วนหน้าจอการออกแบบ ปุ่มที่อยู่ทางด้านบนจะใช้เพื่อการบันทึกโปรเจคในลักษณะต่างๆ การเพิ่มและลบหน้าจอ Screen ปุ่มสำหรับการเปิดส่วนการเขียนโปรแกรม (Open the Blocks Editor) และการจัดแพ็คเกจแอปพลิเคชันเพื่อนำไปใช้งานบนโทรศัพท์ระบบปฏิบัติการ Android ต่อไป ในการสร้างแอปพลิเคชันที่หน้าจอส่วนการออกแบบนี้ ผู้ใช้จะเลือกคอมโพเนนท์ที่อยู่ทางด้านซ้ายของหน้าจอดังรูปที่ 2.28 คลิกลากเพื่อนำมาวางลงในส่วน Viewer ที่อยู่ตรงกลางหน้าจอดังรูปที่ 2.29 หลังจากนั้นคอมโพเนนท์ที่เลือกนำมาวางจะปรากฏในส่วน Viewer ตามมุมมองของผู้ใช้ซึ่งสามารถเลือกจัดวางลงในตำแหน่งที่เหมาะสมได้ตามต้องการ และคอมโพเนนท์นั้นยังปรากฏในส่วนรายการคอนโพเนนท์ (Components) ดังรูปที่ 2.30 เรียงกันเป็นรายการ สามารถเลือกคอมโพเนนท์ที่ต้องการกำหนดคุณสมบัติจากรายการนี้แล้วกำหนดคุณสมบัติต่างๆ ที่หน้าจอส่วนคุณสมบัติ (Properties) ดังรูปที่ 2.31 ซึ่งจะเป็นคุณสมบัติเฉพาะของคอมโพเนนท์นั้นๆ



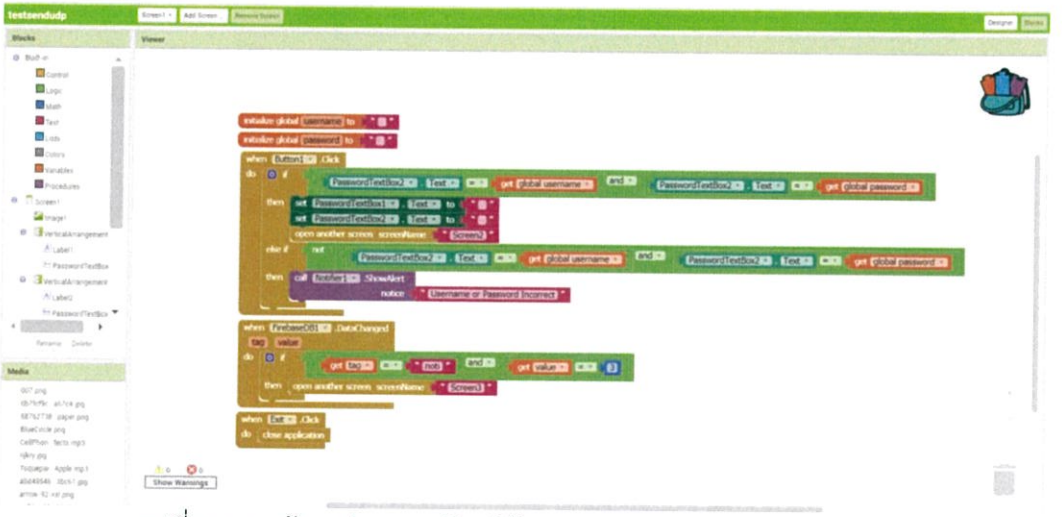
รูปที่ 2.30 หน้าจอส่วนคอมโพเนนท์ (Components) ที่เลือกนำมาใช้ในโปรเจค



รูปที่ 2.31 หน้าจอส่วนคุณสมบัติของคอมโพเนนท์ (Properties)

นอกจากในกลุ่มของคอมโพเนนท์ทั่วไปแล้วยังมีคอมโพเนนท์ที่มองไม่เห็น (Non-Visible Components) ซึ่งเมื่อนำมาวางในหน้าจอ Viewer แล้วจะไม่ปรากฏคอมโพเนนท์ดังกล่าวที่หน้าจอ Viewer แต่จะปรากฏที่หน้าจอรายการคอมโพเนนท์แทน คอมโพเนนท์ที่มองไม่เห็นนี้จะประกอบไปด้วยคอมโพเนนท์ในกลุ่ม Sensors ซึ่งประกอบไปด้วยคอมโพเนนท์ที่เกี่ยวข้องกับการ

เรียกใช้ตัวตรวจจับต่างๆ ที่มีอยู่ในโทรศัพท์ เช่น ระบบ GPS หรือ Accelerometers เป็นต้น กลุ่ม Notifiers ซึ่งเกี่ยวข้องกับความสามารถในการแจ้งเตือนต่างๆ หรือการเขียนบันทึกกิจกรรมของโทรศัพท์ ซึ่งคอมโพเนนท์ในกลุ่ม Notifiers นั้นจะมองไม่เห็นหรือถูกซ่อนไว้ แต่จะสามารถมองเห็นได้เมื่อเกิดการแจ้งเตือนหรือสอบถามโดยมีการโต้ตอบกับผู้ใช้ในรูปของข้อความ เสียง ปุ่ม หรือช่องสำหรับกรอกข้อมูลที่จะแสดงให้ผู้ใช้เห็นเป็นครั้งคราวเท่านั้น กลุ่ม Clocks ซึ่งเกี่ยวข้องกับฟังก์ชันของเวลา ตัวจับเวลา และการตั้งค่าเวลา กลุ่ม ActivityStarters ซึ่งเกี่ยวข้องกับการสั่งให้แอพพลิเคชั่นอื่นที่ติดตั้งอยู่ในโทรศัพท์ทำงาน

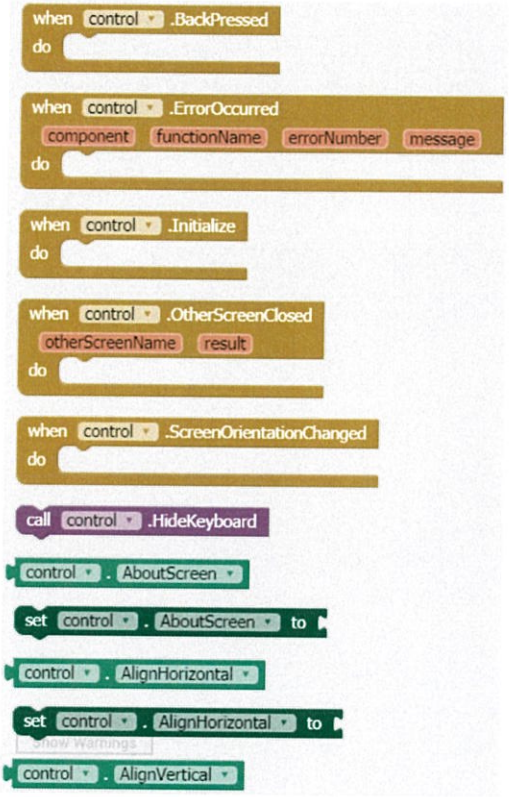


รูปที่ 2.32 หน้าจอส่วนการเขียนโค้ด (App Inventor Blocks Editor)

2.12.2 ส่วนการเขียนโค้ด (App Inventor Blocks Editor)

หลังจากที่ทำการเลือกจัดวางคอมโพเนนท์ที่จะใช้สำหรับโปรเจคครบแล้ว ผู้ใช้จะสามารถเขียนโปรแกรมคำสั่งสำหรับแอปพลิเคชันได้ในส่วนการเขียนโค้ด (App Inventor Blocks Editor) สำหรับพื้นที่การทำงานในส่วนหน้าจอการเขียนโค้ดแสดงดังรูปที่ 2.32 ซึ่งจะประกอบไปด้วยคำสั่งที่อยู่ในรูปของบล็อกรวบรวมไว้บริเวณด้านซ้ายของหน้าจอ ผู้ใช้สามารถเลือกคำสั่งที่ต้องการโดยการคลิกลากบล็อกคำสั่งมาวางไว้ในโปรเจคคือบริเวณที่เป็นพื้นที่วางตรงกลางหน้าจอ ตัวอย่างของบล็อกคำสั่งดังแสดงในรูปที่ 2.33 ซึ่งจะเป็นคำสั่งพื้นฐานที่ผู้ใช้จะนำมาใช้ในการสร้างแอปพลิเคชันขึ้นมา บล็อกเหล่านี้จะถูกแยกและจัดแบ่งออกเป็นกลุ่มๆ ตามลักษณะของคำสั่ง ตัวอย่างเช่น บล็อกข้อความที่ใช้ในการทำงานที่เกี่ยวข้องกับข้อความที่เป็นสายอักขระ บล็อกทางคณิตศาสตร์ที่ใช้ในการทำงานที่เกี่ยวข้องกับฟังก์ชันทางคณิตศาสตร์ ตัวเลขหรือเครื่องหมายทางคณิตศาสตร์ เป็นต้น App Inventor ยังสามารถสร้างกระบวนการทำงาน (Procedure) และตัวแปร

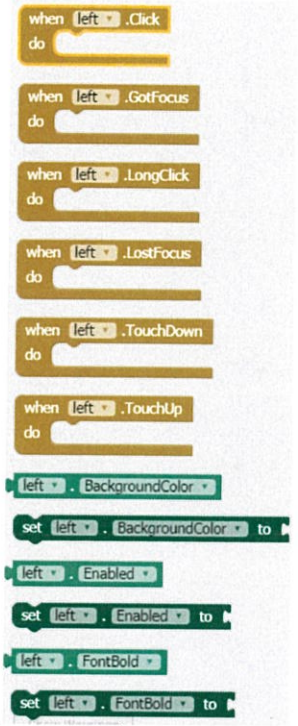
(Variable) ได้โดยการเลือกใช้บล็อกในส่วนที่เกี่ยวกับการสร้างกระบวนการทำงานและเหตุการณ์ (Event Handler) ที่เกิดกับคอมโพเนนท์ โดยบล็อกที่เกี่ยวข้องกับคอมโพเนนท์จะถูกจัดเตรียมไว้ให้ตามคอมโพเนนท์ที่ผู้ใช้เลือกนำมาวางไว้ในโปรเจคและจัดเก็บรวมกันไว้ในแท็บ My Blocks แยกไว้ต่างหาก บล็อกที่เกี่ยวข้องกับคอมโพเนนท์เหล่านี้จะแบ่งออกได้เป็น 4 แบบตามประเภทของคำสั่ง คือ ประเภทการเรียกค่าคุณสมบัติจากคอมโพเนนท์ (Property Getter) ประเภทการกำหนดค่าคุณสมบัติให้กับคอมโพเนนท์ (Property Setter) ประเภทเหตุการณ์ (Event Handler) และประเภทการเรียกใช้กระบวนการทำงาน (Method call)



รูปที่ 2.33 ตัวอย่างของบล็อกคำสั่งที่ใช้แทนการเขียนโปรแกรม

2.12.2.1 การเรียกค่าคุณสมบัติจากคอมโพเนนท์ (Property Getter) บล็อกประเภทที่ใช้เรียกค่าคุณสมบัติจากคอมโพเนนท์จะมีลักษณะเป็นช่องต่ออยู่ทางด้านซ้ายดังรูปที่ 2.34 โดยการทำงานจะทำการอ่านค่าคุณสมบัติจากคอมโพเนนท์แล้วส่งค่านั้นกลับมาในรูปของข้อความ ตัวเลข หรือค่าทางตรรกศาสตร์ แต่ในบางคอมโพเนนท์อาจมีค่าที่มีรูปแบบที่ซับซ้อนมากกว่าเช่น ค่า GPS จากคอมโพเนนท์ตรวจจับตำแหน่ง (Location Sensor) เป็นต้น ซึ่งจะมี

รูปแบบเฉพาะแตกต่างกันไป แต่ทั้งนี้การอ่านค่านั้นทำได้ง่ายมาก ถึงแม้จะเป็นการอ่านค่า GPS ซึ่งโดยปกติมีกระบวนการทำงานที่ซับซ้อน แต่ผู้ใช้สามารถอ่านค่า GPS ได้ผ่านคอมโพเนนท์ตรวจจับตำแหน่งเหมือนอ่านค่าข้อความจากกล่องข้อความ ด้วยกระบวนการนี้ทำให้มั่นใจได้ว่าผู้ใช้จะไม่ต้องกังวลในเรื่องความซับซ้อนของการเข้าถึงค่าและข้อมูลต่างๆ ที่ต้องการ

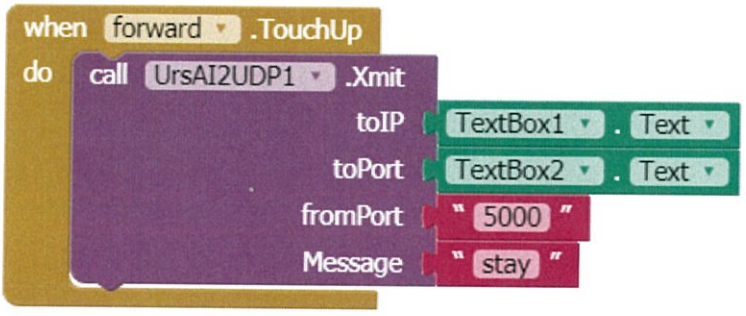


รูปที่ 2.34 ตัวอย่างของบล็อกคำสั่งประเภทที่ใช้เรียกค่าคุณสมบัติจากคอมโพเนนท์ (Property Getter)

2.12.2.2 การกำหนดค่าคุณสมบัติให้กับคอมโพเนนท์ (Property Setter) บล็อกประเภทที่ใช้กำหนดค่าคุณสมบัติให้กับคอมโพเนนท์จะมีลักษณะเป็นช่องต่ออยู่ทางด้านขวา ดังรูปที่ 2.35 โดยจะสามารถทำการกำหนดค่าหรือเปลี่ยนแปลงค่าคุณสมบัติให้กับคอมโพเนนท์ที่ต้องการด้วยค่าของบล็อกที่นำมาต่อเข้ากับช่องต่อที่อยู่ทางด้านขวา ช่องต่อนี้จะมีรูปร่างเป็นช่องรับซึ่งจะต่อเข้าได้พอดีกับบล็อกที่มีรูปร่างเหมือนบล็อกประเภทที่ใช้เรียกค่าคุณสมบัติจากคอมโพเนนท์ ซึ่งจะทำให้ผู้ใช้เลือกบล็อกที่จะนำมาต่อเข้าด้วยกันได้อย่างง่ายได้และลดข้อผิดพลาดในการเลือกต่อบล็อกที่ไม่ถูกต้อง

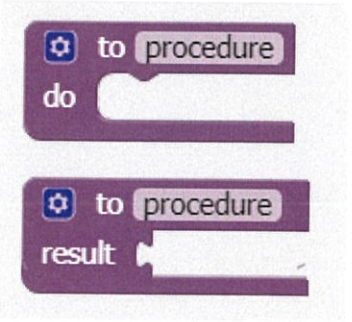
รูปที่ 2.35 ตัวอย่างของบล็อกคำสั่งประเภทที่ใช้กำหนดค่าคุณสมบัติให้กับคอมโพเนนท์ (property setter)

2.12.2.3 เหตุการณ์ (Event Handler) บล็อกประเภทเหตุการณ์จะมีลักษณะเป็นช่องต่ออยู่ทางด้านล่างดังรูปที่ 2.36 ซึ่งบล็อกประเภทนี้จะทำงานเมื่อเกิดเหตุการณ์ต่างๆ ขึ้นกับคอมโพเนนท์ เช่น การคลิกที่ปุ่ม ซึ่งจะทำงานตามบล็อกคำสั่งที่ต่อลงไปทางด้านล่างภายในบล็อกเหตุการณ์ ตัวอย่างเช่นในรูปที่ 2.36 แสดงให้เห็นถึงเหตุการณ์เมื่อมีการคลิกปุ่มแล้วให้มีการแสดงหน้าต่างข้อความโต้ตอบขึ้นมาเพื่อเตือนให้ผู้ใช้ทำการป้อนข้อมูลลงในกล่องข้อความ เป็นต้น



รูปที่ 2.36 ตัวอย่างของบล็อกคำสั่งประเภทเหตุการณ์ (event handler)

2.12.2.4 การเรียกใช้กระบวนการทำงาน (Method Call) บล็อกประเภทเรียกใช้กระบวนการทำงานจะมีลักษณะเหมือนกับบล็อกประเภทที่ใช้เรียกค่าคุณสมบัติจากคอมโพเนนท์ที่มีลักษณะเป็นช่องต่ออยู่ทางด้านซ้าย บล็อกประเภทนี้จะถูกสร้างขึ้นเมื่อผู้ใช้ได้มีการสร้างกระบวนการทำงานโดยอาศัยบล็อกประเภทกระบวนการทำงานในการสร้างดังรูปที่ 2.37 เช่น การสร้างฟังก์ชันการทำงาน การสร้างตัวแปร เป็นต้น และจะมีชื่อเรียกเฉพาะตัวตามที่ผู้ใช้เป็นผู้ตั้งให้เมื่อมีการสร้างกระบวนการทำงานขึ้น บล็อกประเภทเรียกใช้กระบวนการทำงานก็จะถูกสร้างขึ้นและปรากฏในตัวเลือกเพื่อให้ผู้ใช้เลือกที่มาวางลงในโปรเจคเมื่อต้องการให้เกิดการเรียนใช้กระบวนการทำงานดังกล่าว



รูปที่ 2.37 ตัวอย่างของบล็อกคำสั่งประเภทกระบวนการทำงาน

2.12.3 ส่วนของการแพ็คเกจและการเรียกใช้งานแอพพลิเคชัน

เมื่อแอปพลิเคชันได้ถูกออกแบบและทำการเขียนโค้ดคำสั่งเป็นที่เรียบร้อย ผู้ใช้สามารถสั่งให้โปรแกรม App Inventor ทำการแพ็คเกจแอปพลิเคชันดังกล่าวให้อยู่รูปของไฟล์ที่พร้อมจะนำไปติดตั้งเพื่อนำไปติดตั้งบนโทรศัพท์ระบบปฏิบัติการ Android ต่อไป เมื่อผู้ใช้เลือกคลิกที่ปุ่ม Package for Phone ที่อยู่ในด้านบนของหน้าจอส่วนออกแบบ จะทำการแพ็คเกจบนเซิร์ฟเวอร์ App Inventor และส่งไฟล์ที่พร้อมจะนำไปติดตั้งออกมาให้ผู้ใช้ทำการดาวน์โหลดเก็บไว้ในเครื่องคอมพิวเตอร์เพื่อจะนำไปติดตั้งและเรียกใช้งานบนโทรศัพท์ระบบปฏิบัติการ Android เครื่องใดก็ได้ หรือหากไม่มีโทรศัพท์ระบบปฏิบัติการ Android สามารถที่จะทดสอบการทำงานของแอปพลิเคชันได้บนโทรศัพท์จำลองที่ทำงานอยู่บนคอมพิวเตอร์ซึ่งจะมีลักษณะการทำงานเหมือนโทรศัพท์จริงทุกประการดังรูปที่ 2.38

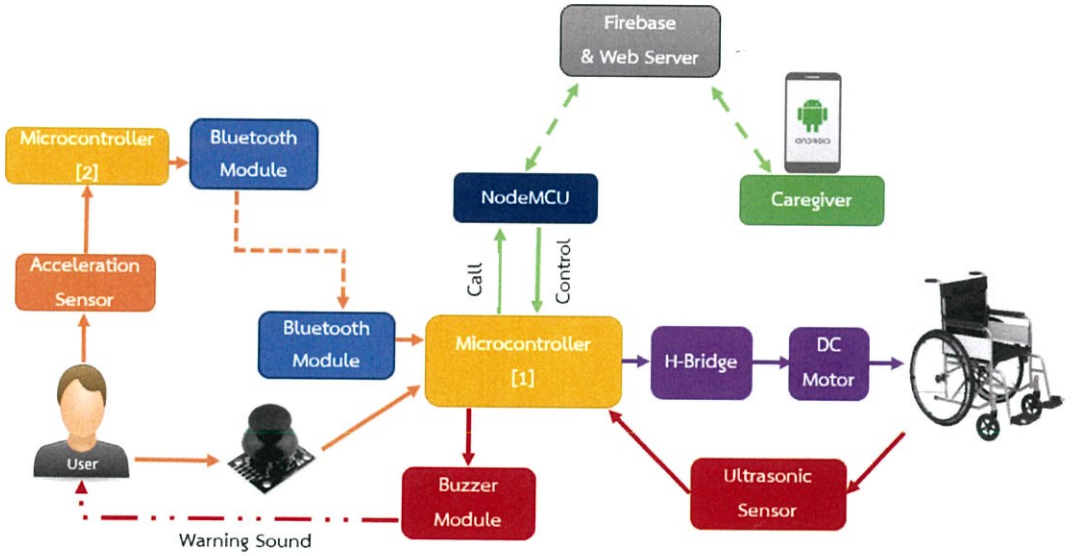
รูปที่ 2.38 โทรศัพท์จำลองระบบปฏิบัติการ Android

บทที่ 3

การออกแบบและการจัดทำปริญญานิพนธ์

ในการจัดทำปริญญานิพนธ์เล่มนี้ ผู้จัดทำได้ออกแบบเก้าอี้เข็นควบคุมได้สำหรับผู้พิการอวัยวะส่วนบนลงมานี้ให้สามารถควบคุมได้ด้วยการเอียงศีรษะและควบคุมด้วยการใช้ก้านควบคุม เพื่ออำนวยความสะดวกให้แก่ผู้พิการและผู้สูงอายุที่ไม่สามารถเดินได้ รวมไปถึงผู้พิการที่นอกจากจะไม่สามารถเดินได้แล้วยังไม่สามารถใช้มือในการควบคุมและขับเคลื่อนเก้าอี้เข็นได้ โดยแบ่งการออกแบบทั้งหมด 5 ส่วน ได้แก่ การออกแบบกลไกขับเคลื่อน, การออกแบบระบบควบคุมทิศทางของเก้าอี้เข็น, การออกแบบการเชื่อมต่อสื่อสารในเก้าอี้เข็น, การออกแบบระบบแจ้งเตือนการชน และการออกแบบพัฒนาแอปพลิเคชัน

3.1 การออกแบบ

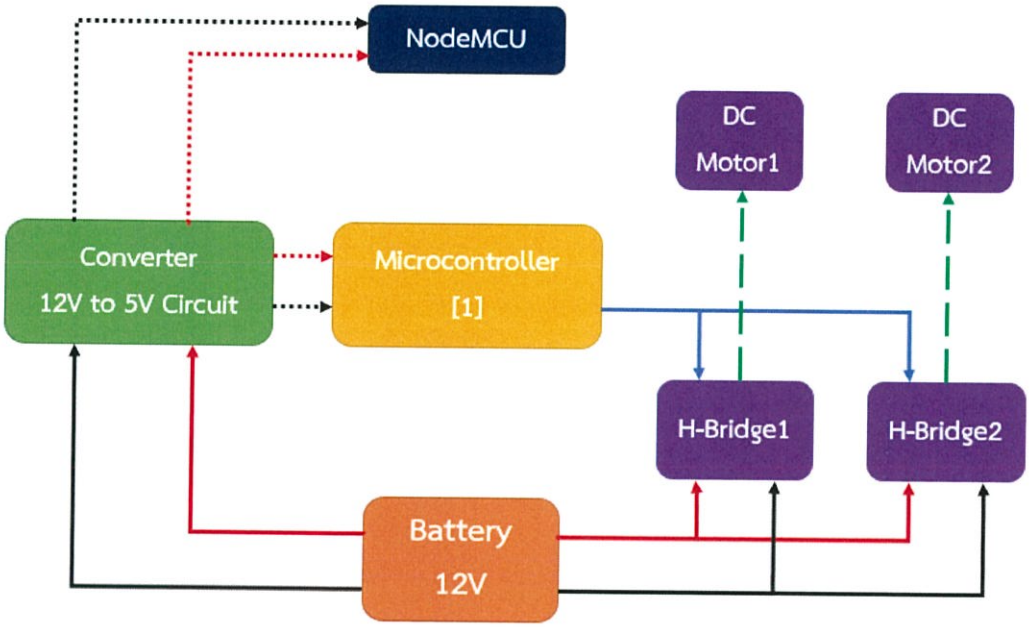


รูปที่ 3.1 บล็อกไดอะแกรมของเก้าอี้เข็นควบคุมได้สำหรับผู้พิการอวัยวะส่วนบนลงมา

จากรูปที่ 3.1 แสดงบล็อกไดอะแกรมของเก้าอี้เข็น ผู้จัดทำได้ออกแบบให้สามารถควบคุมทิศทางการเคลื่อนที่ได้ด้วยการเอียงศีรษะของผู้ใช้งานผ่านการตรวจจับมุมเอียงของศีรษะด้วยเซนเซอร์ความเร่งเชิงเส้น ADXL345 และการใช้ก้านควบคุมบังคับทิศทาง โดยจะขับเคลื่อนเก้าอี้เข็นด้วยมอเตอร์ไฟฟ้าที่ควบคุมโดยไมโครคอนโทรลเลอร์ผ่านวงจรขับมอเตอร์ไฟฟ้ากระแสตรง ซึ่งเก้าอี้เข็นควบคุมได้นี้มีระบบตรวจจับป้องกันการชนวัตถุกีดขวางเมื่อเก้าอี้เข็นเคลื่อนที่ถอยหลัง

ด้วยเซนเซอร์อัลตรานิค และมีระบบเรียกผู้ดูแลเพื่อให้ผู้ใช้งานเรียกผู้ดูแลเพื่อร้องขอความช่วยเหลือ โดยจะแจ้งเตือนให้ผู้ที่ดูแลผู้พิการหรือผู้สูงอายุทราบผ่านแอปพลิเคชันบนสมาร์ตโฟน ทั้งนี้ผู้ดูแลสามารถควบคุมการเคลื่อนที่เก้าอี้เข็นผ่านแอปพลิเคชันดังกล่าวในกรณีเกิดปัญหาติดขัด เพื่อช่วยเหลือผู้ใช้งานในเบื้องต้น

3.1.1 โครงสร้างและกลไกการขับเคลื่อนเก้าอี้เข็น



รูปที่ 3.2 ระบบกลไกขับเคลื่อนเก้าอี้เข็น

ในรูปที่ 3.2 แสดงระบบกลไกขับเคลื่อนเก้าอี้เข็น เมื่อไมโครคอนโทรลเลอร์ได้ประมวลผลทิศทางจากส่วนควบคุมแล้ว ไมโครคอนโทรลเลอร์จะทำการควบคุมการหมุนของมอเตอร์ไฟฟ้าผ่านวงจรขับแรงดันมอเตอร์ (วงจร H-Bridge) ให้เคลื่อนที่ไปยังทิศทางตามการสั่งการ โดยชุดกลไกนี้ออกแบบให้สามารถใช้งานในกรณีที่ไม่ได้ใช้มอเตอร์ไฟฟ้าในการขับเคลื่อนล้อเก้าอี้เข็นแต่ใช้แรงคนเข็นแทนได้

3.1.1.1 การปรับแต่งโครงสร้างและกลไกขับเคลื่อนของเก้าอี้เข็น

1) โครงสร้างเก้าอี้เข็น

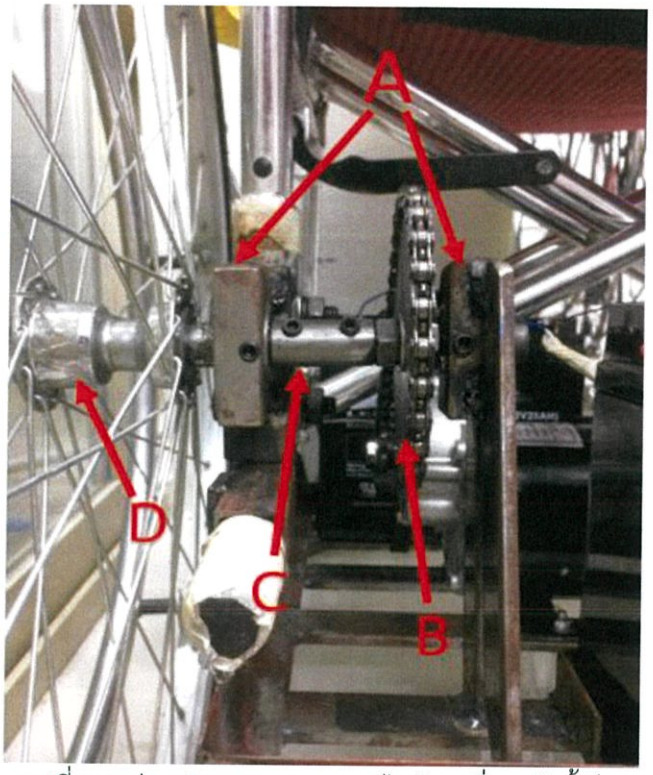
รูปที่ 3.3 เก้าอี้เข็นที่นำมาปรับแต่งโครงสร้าง

ผู้จัดทำได้ใช้เก้าอี้เข็นดังรูปที่ 3.3 ซึ่งเป็นเก้าอี้เข็นทั่วไปที่มีเส้นผ่าศูนย์กลางของล้อเท่ากับ 22 นิ้ว โดยในรูปที่ 3.4 ผู้จัดทำการปรับแต่งโครงสร้างของเก้าอี้เข็นดังกล่าว เพื่อใช้ในการวางมอเตอร์ แหล่งจ่ายไฟฟ้า และกลไกที่ใช้ในการขับเคลื่อนเก้าอี้เข็น

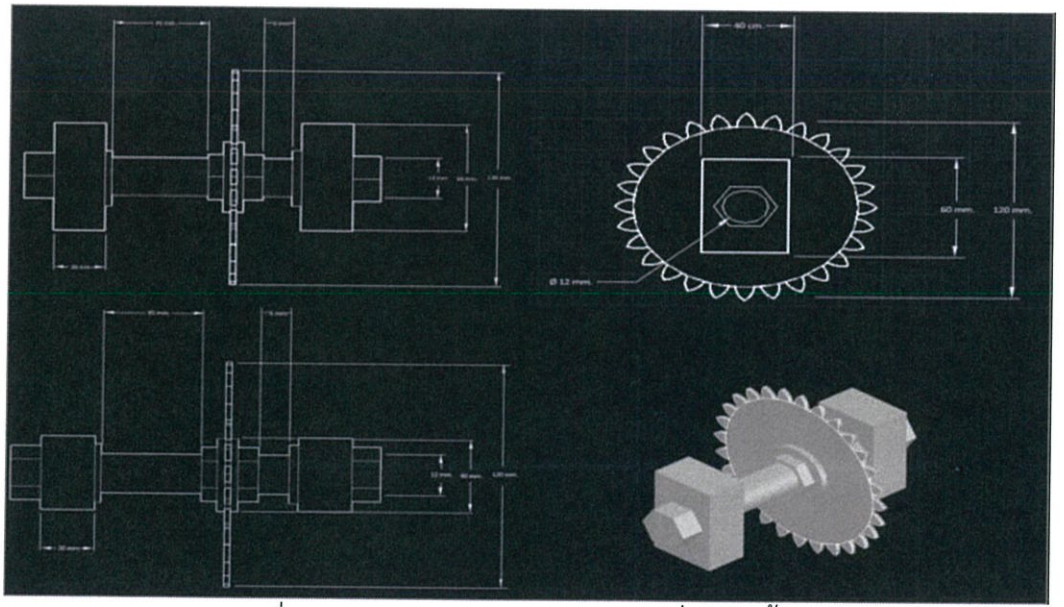
รูปที่ 3.4 โครงสร้างช่วงล่างที่ติดตั้งเพื่อรองรับอุปกรณ์และกลไกในการขับเคลื่อน

2) ชุดกลไกขับเคลื่อนเก้าอี้เข็น

ในรูปที่ 3.5 แสดงส่วนประกอบของชุดกลไกของเก้าอี้เข็น และในรูปที่ 3.6 แสดงการออกแบบชุดกลไกดังกล่าวบนโปรแกรม AutoCAD 2018 โดยจะประกอบไปด้วยชิ้นส่วนดังต่อไปนี้



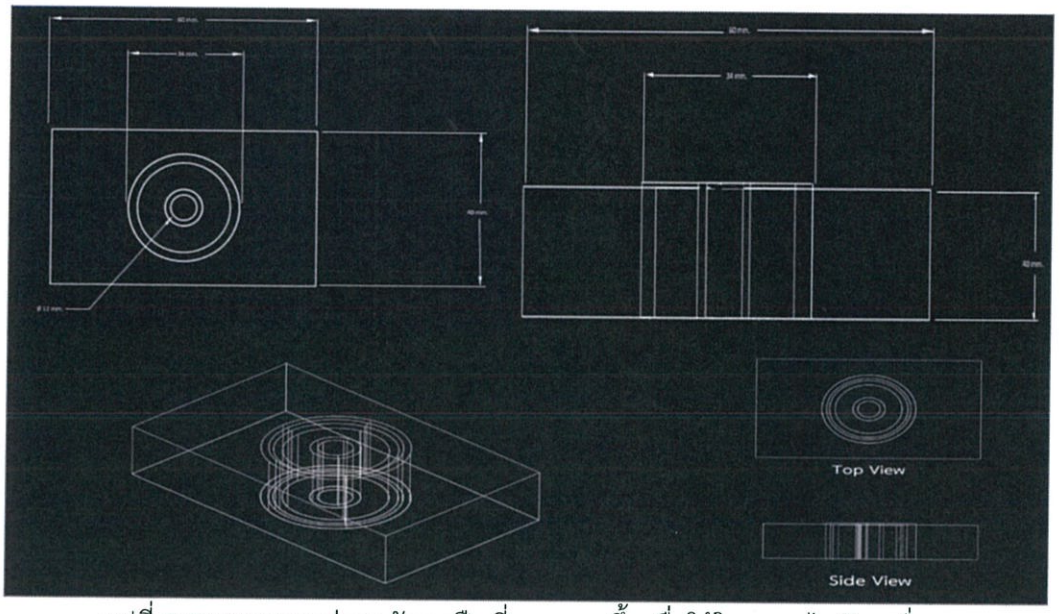
รูปที่ 3.5 ส่วนประกอบของชุดกลไกขับเคลื่อนเก้าอี้เข็น



รูปที่ 3.6 การออกแบบระบบกลไกขับเคลื่อนเก้าอี้เข็น

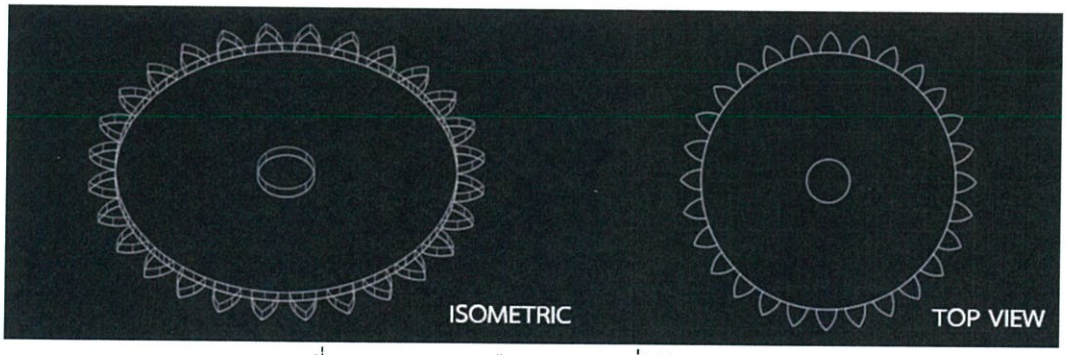
- ชิ้นส่วน A คือ กล่องตลับลูกปืนซึ่งผู้จัดทำได้ออกแบบกล่องตลับลูกปืนขึ้นมาใหม่สำหรับสวมใส่แกนเพลาและเฟืองที่ใช้ในการขับเคลื่อนล้อของเก้าอี้เข็น ให้มีอิสระ

ในการหมุนไม่ว่าจะเป็นการหมุนเพื่อเคลื่อนที่ไปข้างหน้าหรือหมุนเพื่อเคลื่อนที่ถอยหลัง เพื่อตอบสนองการใช้งานในกรณีที่ไม่ได้ใช้การขับเคลื่อนล้อด้วยมอเตอร์ไฟฟ้าแต่สามารถใช้แรงคนเข็นแทนได้ ในรูปที่ 3.7 แสดงแบบของกล่องตลับลูกปืนที่ใช้ในชุดกลไก



รูปที่ 3.7 แบบของกล่องตลับลูกปืนที่ออกแบบขึ้นเพื่อใช้ในชุดกลไกขับเคลื่อน

- ชิ้นส่วน B คือ เฟืองทดที่มีจำนวนฟันทั้งหมด 28 ฟัน โดยนำมาใช้ทดความเร็วรอบของมอเตอร์ รูปที่ 3.8 แสดงแบบของเฟืองที่ใช้ในชุดกลไก



รูปที่ 3.8 แบบของเฟือง 28 ฟันที่ใช้ในชุดกลไก

- ชิ้นส่วน C คือ ส่วนที่เป็นแกนเพลา โดยจะเป็นแกนเพลาที่เชื่อมต่อกับล้อรถเข็นโดยตรงเพื่อให้สามารถเคลื่อนที่ไปข้างหน้าหรือถอยหลังได้อย่างอิสระ

- ชิ้นส่วน D คือล้อของเก้าอี้เข็นที่มีขนาดเส้นผ่าศูนย์กลางขนาด 22 นิ้ว

3.1.1.2 มอเตอร์ไฟฟ้า

รูปที่ 3.9 มอเตอร์ไฟฟ้า

มอเตอร์ไฟฟ้าที่เลือกใช้ในการขับเคลื่อนจะเป็นมอเตอร์ไฟฟ้ากระแสตรง 24 โวลต์ 350 วัตต์ดังรูปที่ 3.9 โดยใช้วงจร H-Bridge ในการควบคุมมอเตอร์ให้ขับเคลื่อนในรูปแบบต่างๆ ซึ่งเมื่อได้รับการจ่ายแรงดันไฟฟ้ากระแสตรง 12 โวลต์ มอเตอร์จะสามารถทำความเร็วรอบได้ประมาณ 200 รอบต่อนาที โดยจะใช้มอเตอร์ 1 ตัวต่อการขับเคลื่อน 1 ล้อ เพื่อให้สะดวกในการใช้บอร์ดไมโครคอนโทรลเลอร์ควบคุมทิศทางจากการป้อนสัญญาณ PWM และเพื่อให้มอเตอร์ทั้งสองตัวสามารถขับเคลื่อนและรองรับโหลด (น้ำหนักผู้ใช้งาน) ได้ ทั้งนี้เพื่อให้มีความเร็วรอบการหมุนที่เหมาะสมในการขับเคลื่อนของผู้ใช้งานซึ่งเป็นผู้พิการและผู้สูงอายุ จึงนำเฟืองมาต่อทดรอบการหมุนของมอเตอร์ให้เหลือเพียง 64 รอบต่อนาที ซึ่งเมื่อคำนวณหาความเร็วในการเคลื่อนที่ของเก้าอี้เข็นโดยใช้ความเร็วรอบค่าดังกล่าวจะมีความเร็วเท่ากับ 6.772 km/hr โดยใช้อัตราการทดเฟืองเป็น 1:3 โดยมีการคำนวณความเร็วรอบหลังการทดรอบดังต่อไปนี้

$$n_2 = n_1\left(\frac{d_1}{d_2}\right) \tag{3.1}$$

โดย n_1 คือ ความเร็วรอบก่อนทด (รอบต่อวินาที)

n_2 คือ ความเร็วรอบหลังทด (รอบต่อวินาที)

d_1 คือ จำนวนฟันของเฟืองมอเตอร์ที่ต้องการทด

d_2 คือ จำนวนฟันของเฟืองที่จะใช้ทดรอบ

จากสมการที่ (3.1)

$$n_2 = n_1\left(\frac{d_1}{d_2}\right)$$

กำหนดให้ ความเร็วรอบก่อนทด (n_1) มีค่าเท่ากับ 200 รอบต่อนาที, จำนวนฟันของเฟืองมอเตอร์ที่ต้องการทด (d_1) จำนวน 9 ฟัน, จำนวนฟันของเฟืองที่จะใช้ทดรอบ (d_2) จำนวน 28 ฟัน จะได้ว่า

$$n_2 = n_1\left(\frac{d_1}{d_2}\right)$$

$$n_2 = 200\,\text{rpm} \times \left(\frac{9}{28}\right)$$

$$n_2 = 64.29 \text{ รอบต่อนาที}$$

จากสมการความเร็วในการเคลื่อนที่ คำนวณหาความเร็วในการเคลื่อนที่ของเก้าอี้เข็นโดยใช้ความเร็วรอบที่คำนวณได้

$$v = \frac{s}{t} \qquad (3.2)$$

โดย v คือ ความเร็วในการเคลื่อนที่ (m/s)

s คือ ระยะทางในการเคลื่อนที่ (m)

t คือ เวลาที่ใช้ในการเคลื่อนที่ (s)

กำหนดให้ ความเร็วรอบของมอเตอร์ (n_2) เท่ากับ 64.29 รอบต่อนาที, รัศมีล้อของเก้าอี้เข็น (r) เท่ากับ 22 นิ้ว

$$v = \left(\frac{n_2 \times 2\pi r}{t_{1\min}}\right)$$

$$v = \left(\frac{64.29 \times 2 \times \pi \times 11 \times 2.54 \times 10^{-2}}{60} \right) \times \frac{18}{5} \ \frac{m}{s}$$

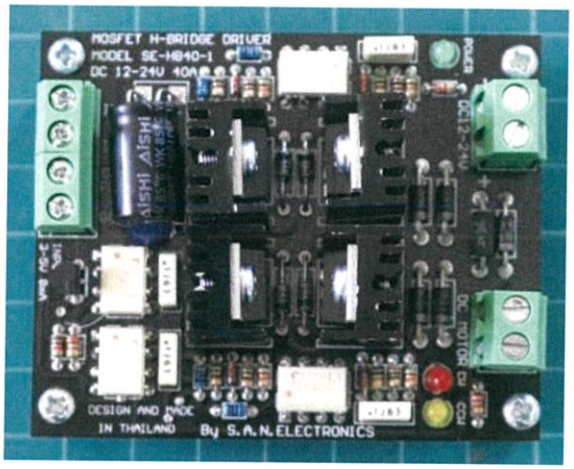
$$v = 6.772 \frac{km}{hr}$$

จากการคำนวณจะได้ความเร็วสูงสุดในการเคลื่อนที่ของเก้าอี้เข็นเท่ากับ 6.772 กิโลเมตรต่อชั่วโมง (มอเตอร์ไฟฟ้าได้รับการจ่ายแรงดันไฟฟ้ากระแสตรงขนาด 12 โวลต์) ซึ่งเป็นความเร็วของเก้าอี้เข็นในขณะที่ยังไม่มีโหลดหรือยังไม่มีผู้ใช้งานนั่งบนเก้าอี้เข็น

3.1.1.3 การใช้วงจรขับมอเตอร์ไฟฟ้ากระแสตรง (H-Bridge) ในการควบคุมมอเตอร์

ผู้จัดทำเลือกใช้วงจรขับมอเตอร์กระแสตรงแบบ H-Bridge รุ่น SE-HB40-1 ดังรูปที่ 3.10 ซึ่งในการควบคุมมอเตอร์ให้ขับเคลื่อนนั้นจะกระทำโดยให้บอร์ดไมโครคอนโทรลเลอร์ Arduino สร้างสัญญาณสี่เหลี่ยม (Pulse Width Modulation) โดยป้อนสัญญาณดังกล่าวให้กับวงจร H-Bridge ซึ่งเป็นวงจรขับมอเตอร์กระแสตรงที่จะทำการขยายระดับแรงดันไฟฟ้าของสัญญาณ PWM ให้สูงขึ้นเพียงพอต่อการใช้ควบคุมและขับเคลื่อนมอเตอร์

ในการใช้งานวงจรขับมอเตอร์นั้นจะกำหนดการความกว้างของลูกคลื่นสี่เหลี่ยม (Duty Cycle) ซึ่งส่งผลต่อความเร็วในการหมุนของมอเตอร์ เมื่อเพิ่มความกว้างของลูกคลื่นสี่เหลี่ยม จะทำให้มอเตอร์มีความเร็วรอบที่สูงขึ้น และกำหนดทิศทางการหมุนของมอเตอร์ของทั้ง 2 ล้อจากการป้อนแรงดันไฟฟ้าไปยังวงจร (IN1 และ IN2) โดยมีการกำหนดค่าความกว้างของลูกคลื่นสี่เหลี่ยมและการการป้อนแรงดันไฟฟ้าอินพุตตามทิศทางการเคลื่อนที่ของเก้าอี้เข็นดังตารางที่ 3.1



รูปที่ 3.10 วงจรขับมอเตอร์กระแสตรงแบบ H-Bridge รุ่น SE-HB40-1

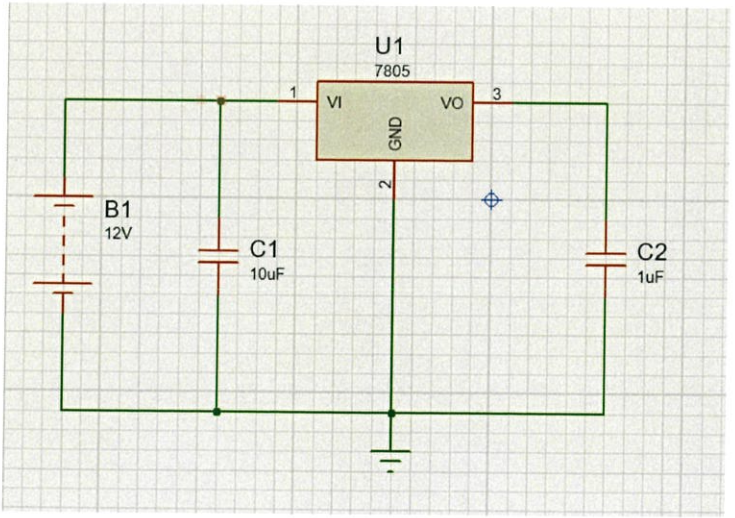
ตารางที่ 3.1 การควบคุมจ่ายแรงดันไฟฟ้าให้กับวงจร H-Bridge

ทิศทางของเก้าอี้เข็น	INPUT H-Bridge 1 (มอเตอร์1)			INPUT H-Bridge 2 (มอเตอร์2)		
	PWM (%)	IN1 (V)	IN2 (V)	PWM (%)	IN1 (V)	IN2 (V)
STAY	0	-	-	0	-	-
FORWARD	48	0	5	48	5	0
BACKWARD	48	5	0	48	0	5
LEFT	48	0	5	0	-	-
RIGHT	0	-	-	48	5	0
BREAK	100	0	0	100	0	0

3.1.1.4 ระบบจ่ายกำลังไฟฟ้า

รูปที่ 3.11 แบตเตอรี่12 โวลต์

ผู้จัดทำได้เลือกใช้แบตเตอรี่แห้งขนาด 12 โวลต์ดังรูปที่ 3.11 ซึ่งนิยมใช้ในจักรยานไฟฟ้า เพื่อจ่ายกำลังไฟฟ้าให้กับระบบโดยแบตเตอรี่จะเชื่อมต่อกับวงจรขับมอเตอร์ไฟฟ้าเพื่อขับเคลื่อนมอเตอร์ไฟฟ้า และเชื่อมต่อกับวงจรแปลงแรงดันไฟฟ้าจาก 12 โวลต์เป็น 5 โวลต์เพื่อป้อนบอร์ดไมโครคอนโทรลเลอร์ดังรูปที่ 3.12



รูปที่ 3.12 วงจรแปลงแรงดันไฟฟ้า

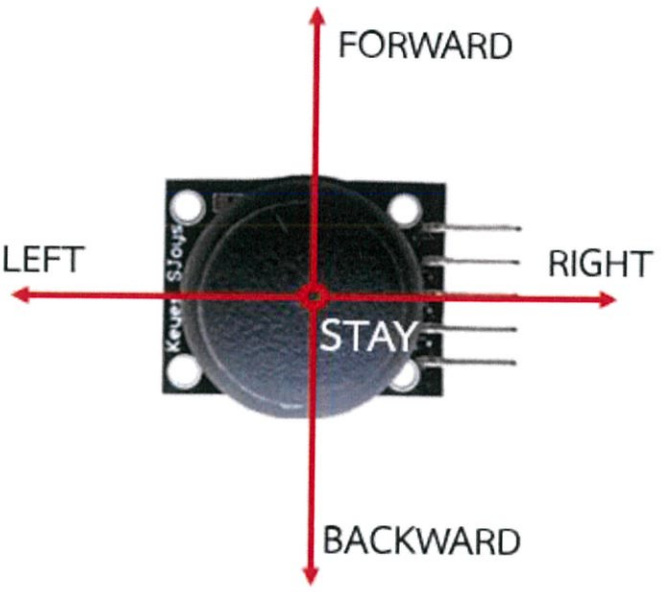
3.1.2 ระบบควบคุมทิศทางของเก้าอี้เข็น

3.1.2.1 การใช้ก้านควบคุม (Joystick) ในการควบคุมทิศทางเก้าอี้เข็น

รูปที่ 3.13 ก้านควบคุม (Joystick)

การใช้ก้านควบคุมในการควบคุมทิศทางเก้าอี้เข็นดังรูปที่ 3.13 ออกแบบขึ้นสำหรับผู้พิการอวัยวะส่วนล่างลงไปหรือไม่สามารถเคลื่อนที่ด้วยการเดิน แต่ยังสามารถใช้มือกดปุ่มหรือใช้บังคับก้านควบคุมได้ เนื่องจากก้านควบคุมสามารถใช้วัดการเคลื่อนไหวในทิศทางต่างๆในระนาบ 2 มิติ โดยเป็นอุปกรณ์ที่ประกอบไปด้วย ตัวต้านทานปรับค่าได้ ซึ่งทำหน้าที่เป็นเซนเซอร์ตรวจวัดค่าแรงดันไฟฟ้าที่เปลี่ยนแปลงไป โดยเมื่อมีการเคลื่อนไหวก้านควบคุมจะทำให้ความต้านทานของตัวต้านทานปรับค่าได้เปลี่ยนแปลง ส่งผลต่อแรงดันไฟฟ้านั้นแปรผันตามการหมุนของอุปกรณ์รอบเพลา

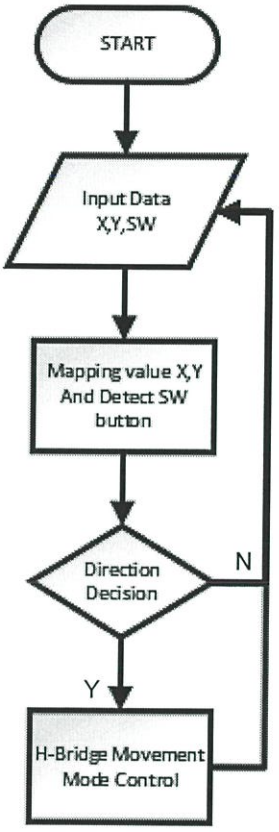
การใช้ก้านควบคุมบังคับทิศทางเก้าอี้เข็นนั้น จะแบ่งการควบคุมทิศทางออกเป็น 5 สถานะ ได้แก่ หยุดนิ่ง, เคลื่อนที่ไปข้างหน้า, เลี้ยวซ้าย, เลี้ยวขวา และถอยหลัง โดยจะกำหนดเงื่อนไขในการตัดสินใจเพื่อแสดงผลทิศทางจากค่าข้อมูลที่อ่านได้จากก้านควบคุม ซึ่งค่าข้อมูลดังกล่าวเป็นค่าที่อ่านได้นั้นเป็นค่าแรงดันไฟฟ้าของแต่ละแกน (ตัวต้านทานปรับค่าได้แต่ละตัว) สำหรับการควบคุมให้มีการหยุดรถ (Break) นั้น จะกำหนดให้เป็นการกดปุ่มกดของก้านควบคุม 1 ครั้ง และการเรียกผู้ดูแลจะกำหนดให้เป็นการกดปุ่มสวิตช์ 1 ครั้ง ในรูปที่ 3.14 แสดงการกำหนดลักษณะการบังคับก้านควบคุมตามสถานะทิศทางการเคลื่อนที่ต่างๆ ในตารางที่ 3.2 แสดงการเชื่อมต่อก้านควบคุมเข้ากับบอร์ดไมโครคอนโทรลเลอร์ของระบบ และรูปที่ 3.15 แสดงแผนผังการทำงานของโปรแกรมการใช้ก้านควบคุมในการควบคุมทิศทางเก้าอี้เข็น



รูปที่ 3.14 การกำหนดการเคลื่อนก้านควบคุมตามสถานะทิศทางการเคลื่อนที่ต่างๆ

ตารางที่ 3.2 การเชื่อมต่อก้านควบคุมเข้ากับบอร์ดไมโครคอนโทรลเลอร์ของระบบเก้าอี้เข็น

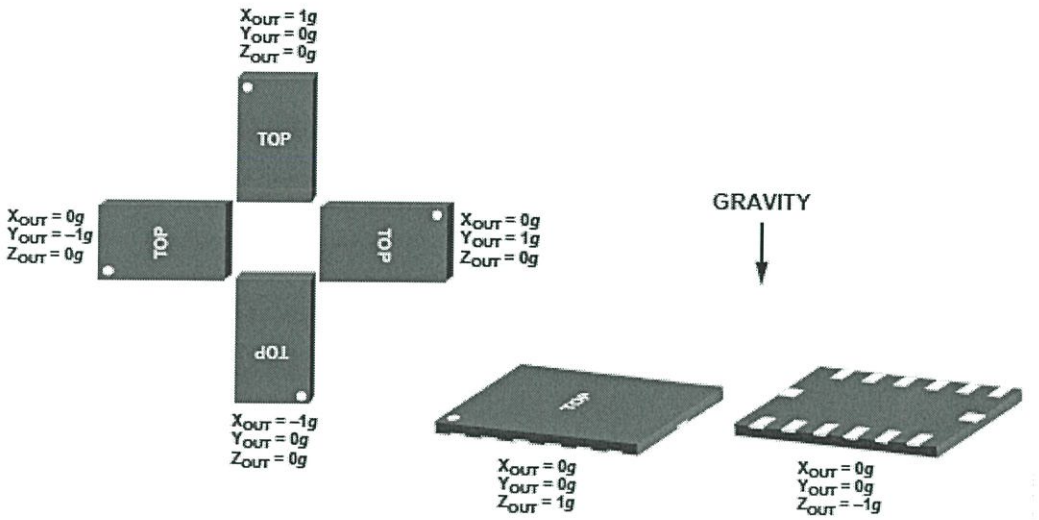
Joystick	Arduino MEGA
GND	GND
VCC	5V
VRx	A0
VRy	A1
sw	A2



รูปที่ 3.15 แผนผังการทำงานของโปรแกรมการใช้ก้านควบคุมในการควบคุมทิศทางเก้าอี้เข็น

3.1.2.2 การควบคุมทิศทางเก้าอี้เข็นด้วยศีรษะโดยใช้เซนเซอร์ความเร่งเชิงเส้น ADXL345

ผู้จัดทำได้เลือกใช้งานเซนเซอร์ความเร่งเชิงเส้น ADXL345 ซึ่งเป็นเซนเซอร์วัดความเร่งแบบ 3 แกนขนาดเล็ก สามารถวัดความเร่งเชิงเส้นอันเป็นผลจากความเร่งโน้มถ่วงที่กระทำต่อตัวเซนเซอร์ในแต่ละแนวแกนได้ ADXL345 เป็นเซนเซอร์ความเร่งแบบดิจิตอลที่มีตัวแปลงอนาล็อคเป็นดิจิตอลในตัว จึงสามารถใช้ไมโครคอนโทรลเลอร์รับค่าได้โดยตรง โดยจะอาศัยการเคลื่อนไหวของแผงสปริงที่มีผลต่อความเปลี่ยนแปลงของค่าความจุไฟฟ้าภายในโครงสร้างบนแผ่นเพลท ซึ่งเซนเซอร์จะทำการตรวจวัดแรงดันไฟฟ้าดังกล่าวแล้วแปลงเป็นข้อมูลดิจิตอล ซึ่งเป็นข้อมูลที่บ่งบอกถึงความเร่งโน้มถ่วงที่กระทำต่อแกนอ้างอิงนั้นๆ รูปที่ 3.16 แสดงค่าความเร่งโน้มถ่วง (1g) ที่กระทำต่อแต่ละแกนอ้างอิงของเซนเซอร์เมื่อจัดวางในตำแหน่งที่ให้ความเร่งกระทำต่อแกนโดยตรง จะเห็นได้ว่าแต่ละตำแหน่งการวางเซนเซอร์ในบางแกนจะมีค่า 1 g ($9.8\ m/s^2$) มากระทำต่อแกนนั้น ในขณะที่บางแกนเป็นค่า 0g



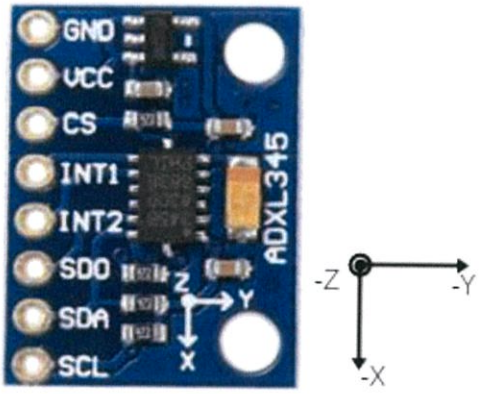
รูปที่ 3.16 ค่าความเร่งโน้มถ่วงที่กระทำต่อแต่ละแกนของเซนเซอร์ในตำแหน่งต่างๆ

การเชื่อมต่อเซนเซอร์ ADXL345 กับบอร์ดไมโครคอนโทรลเลอร์ Arduino นั้น สามารถทำได้ 2 โหมดคือ โหมด SPI และโหมด I2C ในที่นี้จะเลือกใช้งานโหมด I2C ซึ่งเป็นการเชื่อมต่อสื่อสารดิจิตอล โดย ADXL345 ได้รองรับการเชื่อมต่อแบบ I2C ในตัว ตารางที่ 3.3 แสดงขาการเชื่อมต่อ ADXL345 เข้ากับบอร์ดไมโครคอนโทลเลอร์ Arduino

ตารางที่ 3.3 การเชื่อมต่อ ADXL345 เข้ากับบอร์ดไมโครคอนโทรลเลอร์ Arduino

ADXL345	Arduino Nano
GND	GND
VCC	3.3V
CS	3.3V
SDO	GND
SDA	A4
SCL	A5

1) การอ่านค่าข้อมูลจากเซนเซอร์ความเร่งเชิงเส้น ADXL345



รูปที่ 3.17 เซนเซอร์ความเร่งเชิงเส้น ADXL345

จากรูปที่ 3.17 แสดงเซนเซอร์ความเร่งเชิงเส้น ADXL345 ซึ่งจะมีการระบุสัญลักษณ์แกนอ้างอิงของตัวเซนเซอร์เอง เนื่องจากเซนเซอร์จะมีแกนอ้างอิงทั้งในแกนบวกและลบ โดยสัญลักษณ์บนตัวเซนเซอร์ ADXL345 นั้นแสดงถึงแกนลบของเซนเซอร์ โดยหมายถึงหากให้ความเร่งโน้มถ่วงกระทำต่อแกนอ้างอิงที่เป็นแกนลบ ค่าข้อมูลเอาต์พุตที่ได้จะมีค่าเป็นลบเช่นกัน

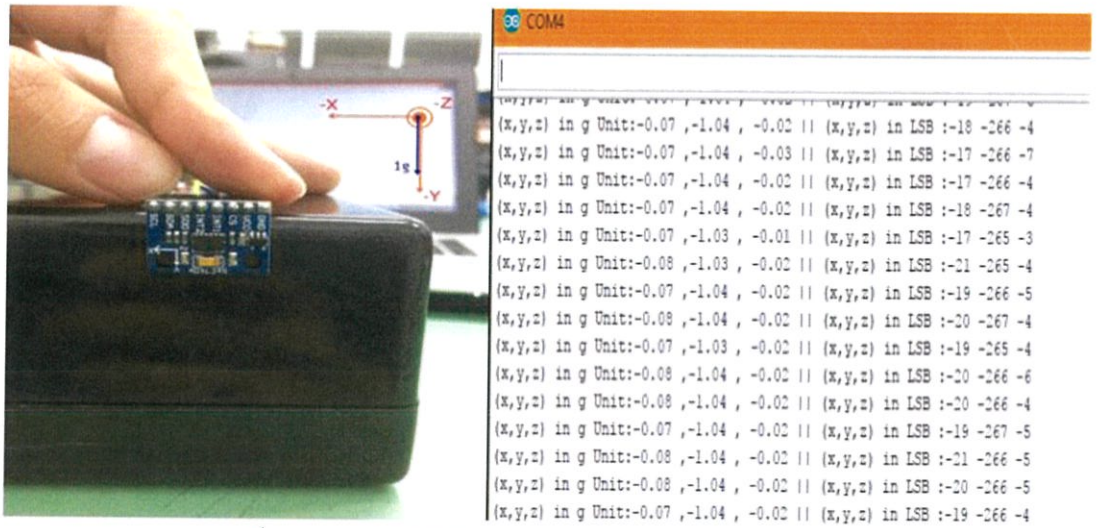
ข้อมูลเอาต์พุตที่ได้จาก ADXL345 (x_{LSB} , y_{LSB} , z_{LSB}) เป็นข้อมูลที่ผ่านการแปลงเป็นข้อมูลดิจิตอลบิต โดยในการอ่านค่าข้อมูลจากเซนเซอร์นั้นจะเป็นอ่านข้อมูลไบต์จากรีจิสเตอร์ตำแหน่งที่ 0x32 ถึง 0x37 โดยที่ตำแหน่ง 0x32 กับ 0x33 เป็นข้อมูลในแกน X, 0x34 กับ 0x35 เป็นข้อมูลในแกน Y และ 0x36 กับ 0x37 เป็นข้อมูลในแกน Z จากรูปที่ 3.18 เป็นการวางเซนเซอร์ ADXL345 แนบกับขอบของวัสดุทรงสี่เหลี่ยมพื้นผิวราบเรียบ เพื่อให้เซนเซอร์นั้นถูกจัดวางในตำแหน่งที่ความเร่งโน้มถ่วงจะกระทำต่อแกนอ้างอิงเพียงแกนเดียวของเซนเซอร์เท่านั้น จากรูปจะเป็นการวางให้ความเร่งโน้มถ่วงกระทำต่อแกน -Y ของเซนเซอร์ พบว่าบน Serial Monitor แสดงค่าข้อมูลความเร่งที่อ่านได้นั้น จะมีค่ามากในแกน Y โดยค่าที่ติดลบนั้นแสดงถึงความเร่งโน้มถ่วงนั้นกระทำกับแกนอ้างอิงลบของเซนเซอร์ รวมทั้งแสดงค่าความเร่งในหน่วย g ซึ่งเป็นค่าที่คำนวณได้จากสมการที่ (3.3) ถึงสมการที่ (3.5)

$$x_{[g]} = x_{LSB} \times ScaleFactor \tag{3.3}$$

$$y_{[g]} = y_{LSB} \times ScaleFactor \tag{3.4}$$

$$z_{[g]} = z_{LSB} \times ScaleFactor \tag{3.5}$$

โดย *ScaleFactor* คือ ค่าความละเอียดของค่าข้อมูลความเร่งที่ได้จากเซนเซอร์ โดยจะแปรเปลี่ยนไปตามช่วงความไว (Sensitivity) ที่เลือกใช้ เช่น เลือกใช้ที่ช่วงความไว 2g จะต้องใช้ค่าความละเอียด 3.9 mg/LSB ในการแปลงให้อยู่ในหน่วยความเร่ง g



รูปที่ 3.18 ความเร่งโน้มถ่วงกระทำต่อแกน -Y ของเซนเซอร์

2) การใช้เซนเซอร์ความเร่งเชิงเส้น ADXL345 ในการวัดมุมเอียง

เนื่องจากเซนเซอร์ความเร่ง ADXL345 นั้นมีการกำหนดแกนอ้างอิงสำหรับตัวเซนเซอร์ในระบบพิกัดฉาก 3 มิติ การตรวจวัดมุมเอียงจึงสามารถทำได้ด้วยการนำข้อมูลเอาต์พุตที่ได้จาก ADXL345 มาคำนวณด้วยสมการทางตรีโกณมิติดังสมการที่ (3.6) ถึงสมการที่ (3.8) โดยเป็นการแยกคำนวณมุมที่กระทำต่อแต่ละแกนอ้างอิง จากรูปที่ 3.19 แสดงให้เห็นเมื่อมีการเอียงของเซนเซอร์จากแนวแกนอ้างอิงปกติ (ตำแหน่งปกติที่ความเร่งโน้มถ่วงกระทำต่อแกน Z เท่านั้น) ทำให้เกิดมุมระหว่างแกนอ้างอิงของตัวเซนเซอร์และแกนอ้างอิงในตำแหน่งปกติ จึงใช้ค่ามุมดังกล่าวบ่งบอกถึงมุมความเอียงของเซนเซอร์ได้ รูปที่ 3.20 Serial Monitor แสดงผลค่ามุมเอียงของแต่ละแกนในขณะที่เซนเซอร์แนบกับขอบของวัสดุทรงสี่เหลี่ยม จะเห็นได้ว่ามีมุมเอียง θ และ ϕ เกิดขึ้นประมาณ 90 องศา เพราะมีการเอียงจากแกนอ้างอิงปกติของแกน X และแกน Z

$$\theta = \tan^{-1}\left(\frac{x_{out[LSB]}}{\sqrt{y_{out[LSB]}^2 + z_{out[LSB]}^2}}\right) \quad (3.6)$$

$$\psi = \tan^{-1}\left(\frac{y_{out[LSB]}}{\sqrt{x_{out[LSB]}^2 + z_{out[LSB]}^2}}\right) \quad (3.7)$$

$$\phi = \tan^{-1}\left(\frac{\sqrt{x_{out[LSB]}^2 + y_{out[LSB]}^2}}{z_{out[LSB]}}\right) \quad (3.8)$$

โดย θ คือ มุมที่เซนเซอร์กระทำต่อแกนอ้างอิง X

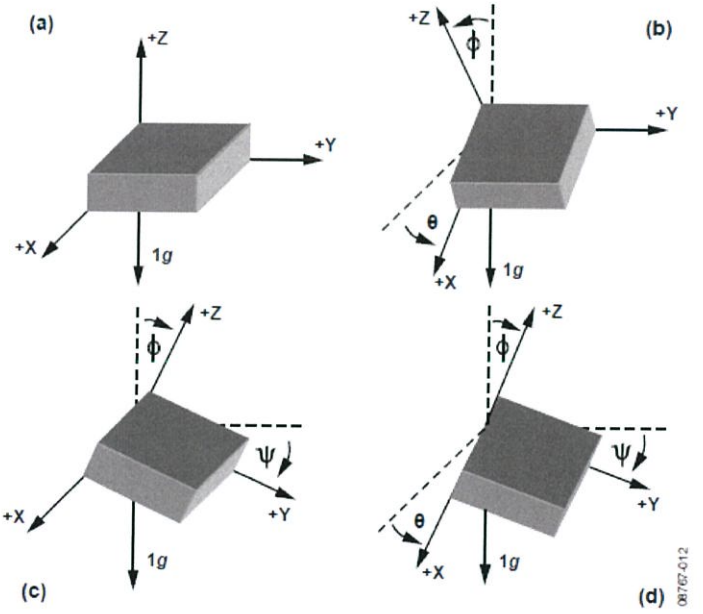
ψ คือ มุมที่เซนเซอร์กระทำต่อแกนอ้างอิง Y

ϕ คือ มุมที่เซนเซอร์กระทำต่อแกนอ้างอิง Z

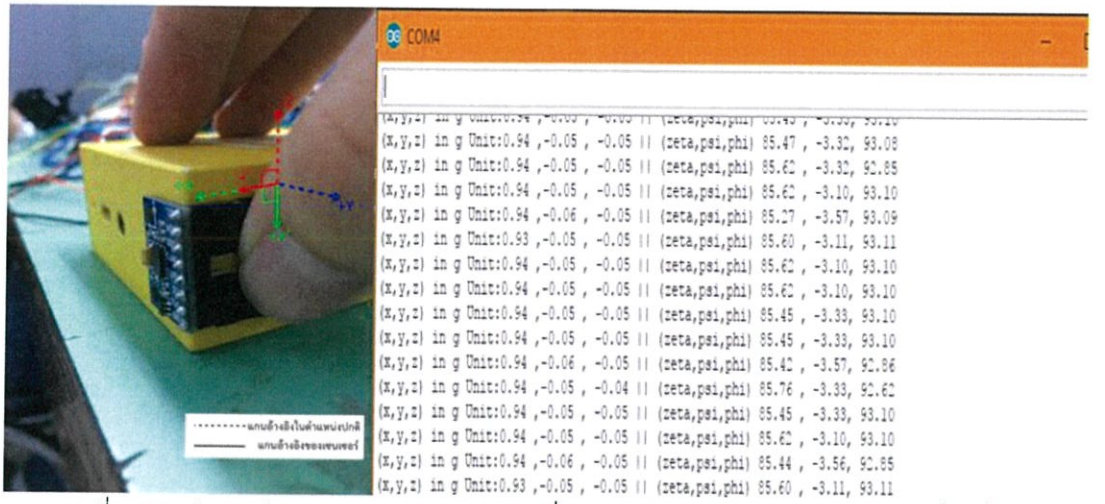
$x_{out[LSB]}$ คือ ค่าข้อมูลเอาต์พุตเซนเซอร์ความเร่งของแกน X

$y_{out[LSB]}$ คือ ค่าข้อมูลเอาต์พุตเซนเซอร์ความเร่งของแกน Y

$z_{out[LSB]}$ คือ ค่าข้อมูลเอาต์พุตเซนเซอร์ความเร่งของแกน Z



รูปที่ 3.19 มุมเอียงระหว่างแกนอ้างอิงของตัวเซนเซอร์และแกนอ้างอิงในตำแหน่งปกติ



รูปที่ 3.20 ค่ามุมเอียงของแต่ละแกนในขณะที่เซนเซอร์แนบกับขอบของวัสดุทรงสี่เหลี่ยม

3) การปรับเทียบความแม่นยำในการวัดมุมเอียงของ ADXL345

การนำเซนเซอร์ความเร่งเชิงเส้น ADXL345 มาใช้ตรวจวัดมุมเอียงนั้น ค่ามุมที่วัดได้อาจมีความคลาดเคลื่อนจากค่ามุมจริง เนื่องจากการใช้งานเซนเซอร์อาจถูกรบกวนด้วยแรงกดบนตัวเซนเซอร์จนกระทบต่อโครงสร้างภายใน รวมไปถึงข้อจำกัดในการปรับเทียบเซนเซอร์โดยผู้ผลิต ซึ่งล้วนส่งผลต่อค่าความไวของเซนเซอร์และความแม่นยำในการใช้งานวัดค่ามุมเอียง

จากสมการที่ (3.6) ถึงสมการที่ (3.8) ที่ใช้ในการคำนวณมุมเอียงนั้นเป็นสมการที่ไม่เป็นเชิงเส้น ทำให้ค่ามุมที่ได้นั้นจึงมีความแม่นยำในบางช่วงมุม เพราะในแต่ละช่วงมุมนั้นจะเกิดค่าความคลาดเคลื่อนที่ไม่เท่ากัน ทำให้จึงมีความยุ่งยากในการชดเชยค่าความคลาดเคลื่อนโดยตรง ดังนั้นในการปรับเทียบจะทำการคำนวณหาค่า Offset ($A_{OFF}[g]$) และค่า Gain ($Gain$) ของค่าความเร่งแต่ละแกน เพื่อนำไปหาค่าปรับเทียบในสมการที่ (3.11) ซึ่งเป็นค่าความเร่งเชิงเส้นที่ผ่านการชดเชยความคลาดเคลื่อนแล้ว ($A_{Calibrate[g]}$) จากนั้นจึงนำไปเข้าสมการเพื่อคำนวณค่ามุมเอียงต่อไป

$$A_{OFF}[g] = 0.5 \times (A_{+1g} + A_{-1g}) \tag{3.9}$$

$$Gain = 0.5 \times \left(\frac{A_{+1g} - A_{-1g}}{1g} \right) \tag{3.10}$$

$$A_{Calibrate[g]} = \frac{A_{out[g]} - A_{OFF[g]}}{Gain} \tag{3.11}$$

โดย $A_{OFF}[g]$ คือ ค่าชดเชย (Offset)

$Gain$ คือ อัตราขยาย

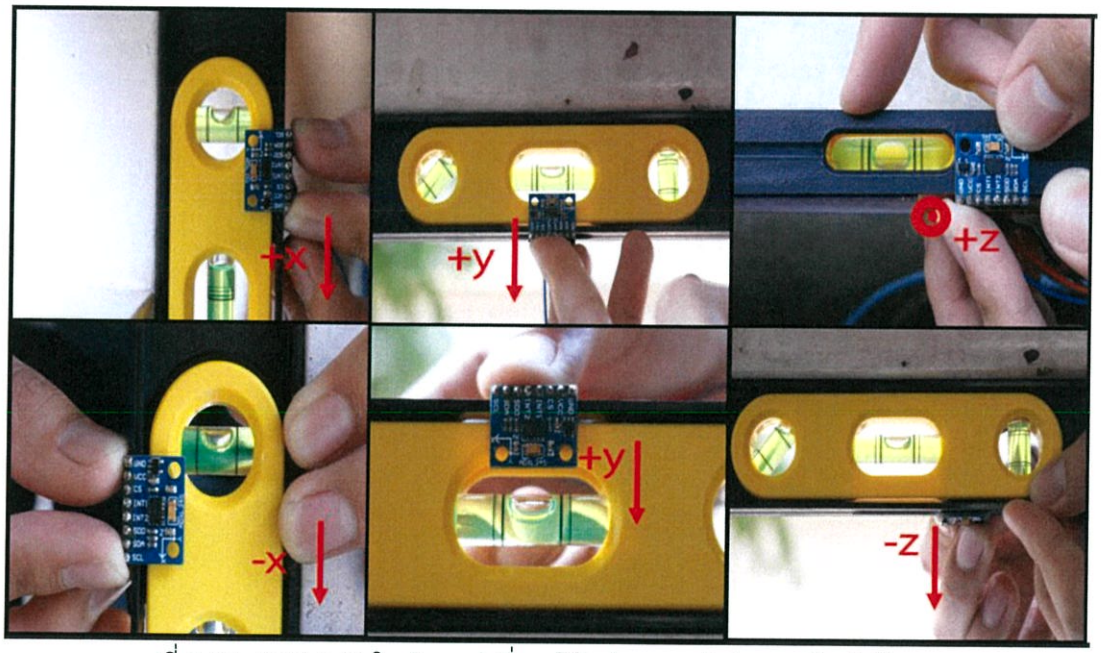
$A_{Calibrate[g]}$ คือ ค่าความเร่งเชิงเส้นหลังการชดเชยความคลาดเคลื่อน

$A_{out[g]}$ คือ ค่าข้อมูลเอาต์พุตเซนเซอร์ความเร่งเชิงเส้น

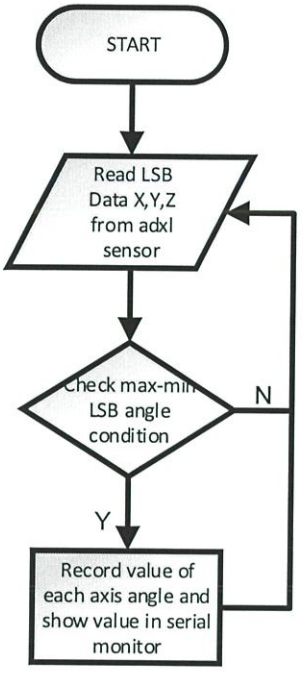
A_{+1g} คือ ค่าความเร่งสูงสุดที่กระทำต่อแกนอ้างอิงบวก

A_{-1g} คือ ค่าความเร่งต่ำสุดที่กระทำต่อแกนอ้างอิงลบ

ในการหาค่าความเร่งเชิงเส้นสูงสุดกระทำต่อแกนอ้างอิงบวก (A_{+1g}) และค่าความเร่งเชิงเส้นต่ำสุดกระทำต่อแกนอ้างอิงลบ (A_{-1g}) จะกระทำได้โดยจัดวางเซนเซอร์ ADXL345 ในตำแหน่งที่แรงโน้มถ่วงกระทำโดยตรงต่อแกนอ้างอิงของเซนเซอร์ในทุก ๆ แกนทั้งหมด 6 ตำแหน่งดังรูปที่ 3.21 โดยโปรแกรมในการจัดเก็บค่าความเร่งสูงสุดและต่ำสุดจะมีการทำงานดังรูปที่ 3.22



รูปที่ 3.21 ADXL345 ในตำแหน่งที่แรงโน้มถ่วงกระทำต่อแกนอ้างอิงโดยตรง



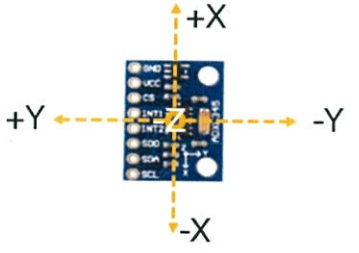
รูปที่ 3.22 แผนผังการทำงานของโปรแกรมจัดเก็บค่าความเร่งเชิงเส้นสูงสุดและต่ำสุด

4) การควบคุมทิศทางเก้าอี้เข็นโดยใช้ ADXL345

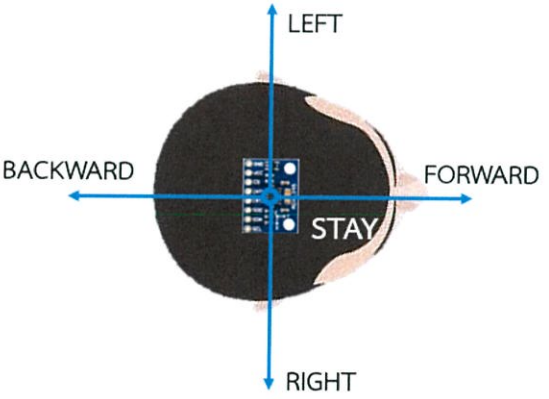
การใช้ศีรษะควบคุมทิศทางของเก้าอี้เข็นโดยใช้ ADXL345 นั้น ออกแบบขึ้นเพื่อรองรับการใช้งานโดยผู้พิการอวัยวะส่วนบนลงมาที่ไม่สามารถใช้มือในการบังคับทิศทางเก้าอี้เข็นด้วยก้านควบคุมหรือปุ่มกดได้ สำหรับการใช้เซนเซอร์ความเร่งเชิงเส้น ADXL345 ในการตรวจวัดค่าความเร่งจากการขยับศีรษะจะกำหนดให้ใช้ความไว (Sensitivity) ที่ $\pm 2g$ ซึ่งเป็นช่วงความไวที่สามารถรับรู้ความเร่งจากการเคลื่อนไหวเพียงเล็กน้อย จึงเหมาะสำหรับการใช้งานตรวจวัดการเคลื่อนไหวของศีรษะ ผู้ใช้งานจะใช้การเอียงศีรษะไปในทิศทางที่ต้องการเคลื่อนที่ไป โดยแบ่งการควบคุมทิศทางเก้าอี้เข็นออกเป็น 5 สถานะ ได้แก่ หยุดนิ่ง (STAY), เคลื่อนที่ไปข้างหน้า (FORWARD), เลี้ยวซ้าย (LEFT), เลี้ยวขวา (RIGHT) และถอยหลัง (BACKWARD) โดยจะแบ่งช่วงเงื่อนไขจากมุมการเอียงของศีรษะในการพัฒนาโปรแกรมดังตารางที่ 3.4 ซึ่งแสดงการกำหนดเงื่อนไขเริ่มต้นของมุมเอียงศีรษะในโปรแกรมการควบคุมทิศทางการเคลื่อนที่ โดยในรูปที่ 3.23 แสดงลักษณะการจัดวางเซนเซอร์ ADXL345 บนศีรษะ

ตารางที่ 3.4 การกำหนดเงื่อนไขเริ่มต้นมุมเอียงศีรษะในการควบคุมทิศทางการเคลื่อนที่

ทิศทาง	การเอียงศีรษะ	มุมเอียงศีรษะที่กระทำต่อแกนอ้างอิง (องศา)		
		θ	ψ	ϕ
หยุดนิ่ง (STAY)	ศีรษะตั้งตรง	$\lvert\theta\rvert < 25$	$\lvert\psi\rvert < 25$	$\lvert\phi\rvert < 25$
เคลื่อนที่ไปข้างหน้า (FORWARD)	ก้มศีรษะลง 1 ครั้ง	$\lvert\theta\rvert < \lvert\psi\rvert$, $\lvert\theta\rvert < \lvert\phi\rvert$	$\psi \le -25$	$\phi \ge 25$
ถอยหลัง (BACKWARD)	เงยศีรษะขึ้น	$\lvert\theta\rvert < \lvert\psi\rvert$, $\lvert\theta\rvert < \lvert\phi\rvert$	$\psi \ge 25$	$\phi \ge 25$
เลี้ยวซ้าย (LEFT)	เอียงศีรษะไปทางซ้าย	$\theta \ge 25$	$\lvert\psi\rvert < \lvert\theta\rvert$, $\lvert\psi\rvert < \lvert\phi\rvert$	$\phi \ge 25$
เลี้ยวขวา (RIGHT)	เอียงศีรษะไปทางขวา	$\theta \le -25$	$\lvert\psi\rvert < \lvert\theta\rvert$, $\lvert\psi\rvert < \lvert\phi\rvert$	$\phi \ge 25$



แกนอ้างอิงของเซนเซอร์ ADXL345

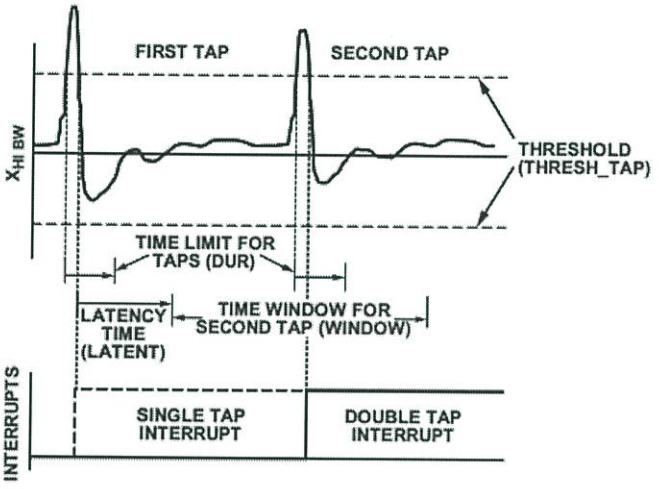


ลักษณะการเอียงศีรษะเทียบกับทิศทางการเคลื่อนที่

รูปที่ 3.23 ลักษณะการจัดวางเซนเซอร์ความเร่งเชิงเส้น ADXL345 บนศีรษะ

เซนเซอร์ความเร่งเชิงเส้น ADXL345 จะมีฟังก์ชันการขัดจังหวะ (Interrupt) ซึ่งเป็นฟังก์ชันที่เซนเซอร์ตรวจจับการเกิดเหตุการณ์ Tap หรือ Double Tap โดยการเกิด Tap เป็นเหตุการณ์ที่ค่าความเร่งเชิงเส้นกระทำต่อแกนที่สนใจมีค่าเพิ่มขึ้นอย่างสูงสุด (ใน

แนวแกนบวก) หรือลดลงต่ำสุด (ในแนวแกนลบ) (Peak) อย่างรวดเร็วในเวลาที่กำหนด (DUR Time) โดยจะมีการเว้นระยะเวลาเผื่อมีค่าความเร่งเชิงเส้นแฝง (LATENT Time) ไว้ด้วย ส่วนการเกิด Double Tap จะเป็นเหตุการณ์ที่เกิดการ Tap หลังจากการเกิด Tap ครั้งแรกภายในระยะเวลาที่กำหนด (WINDOW Time) ในรูปที่ 3.24 แสดงการเกิด Tap และ Double Tap โดยมีการกำหนดรีจิสเตอร์เพื่อใช้งานฟังก์ชันขัดจังหวะดังตารางที่ 3.5



รูปที่ 3.24 การเกิด Tap และ Double Tap

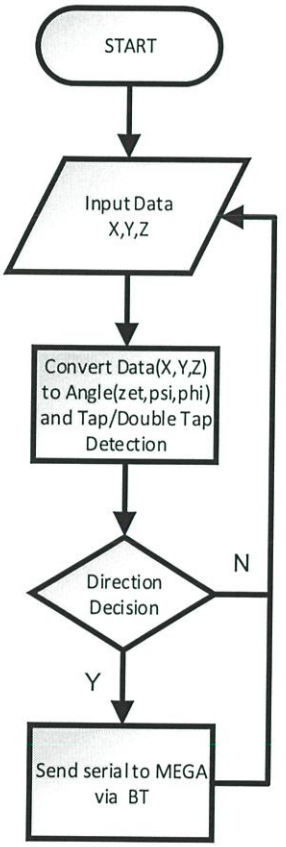
ตารางที่ 3.5 ตำแหน่งรีจิสเตอร์ที่ใช้งานกำหนดฟังก์ชันการขัดจังหวะ (Interrupt)

ตำแหน่ง	คำอธิบาย	Scale Factor
0x1D	Tap Threshold	62.5 mg/LSB
0x21	Tap Duration	625 us/LSB
0x22	Tap Latency	1.25 ms/LSB
0x23	Tap Window	1.25 ms/LSB

สำหรับการควบคุมเพื่อเปิด/ปิดการใช้ศีรษะควบคุมเก้าอี้เข็นจะกำหนดให้มีการเกิด Tap ของเซนเซอร์ความเร่งเชิงเส้น ซึ่งเป็นการตรวจจับค่าความเร่งเชิงเส้นที่เพิ่มขึ้นและลดลง (Peak) ในเวลาที่กำหนด 1 ครั้งโดยตัวเซนเซอร์เอง โดยสรุปได้ดังตารางที่ 3.6

ตารางที่ 3.6 การควบคุมเก้าอี้เข็นผ่านฟังก์ชันขัดจังหวะ (Interrupt)

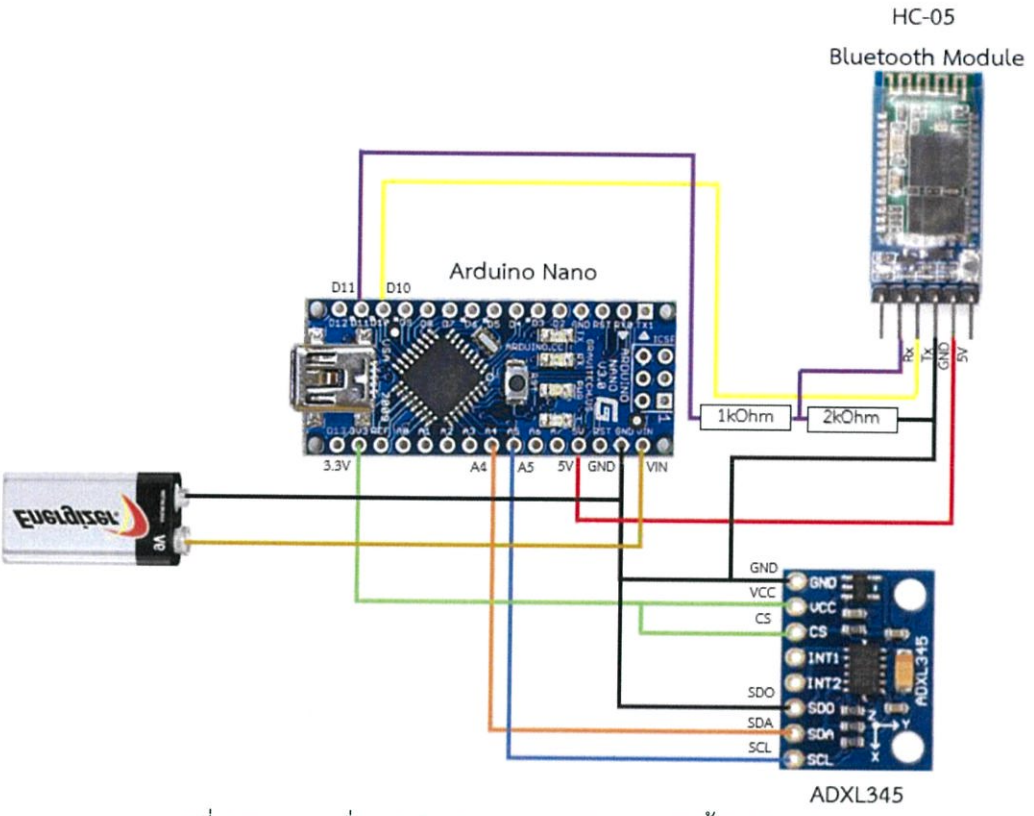
คำสั่ง	การขยับศีรษะ	เงื่อนไขการขัดจังหวะ (Interrupt)
การเปิด/ปิดการใช้ศีรษะควบคุมเก้าอี้เข็น	เอียงศีรษะไปด้านข้าง 1 ครั้ง	Tap



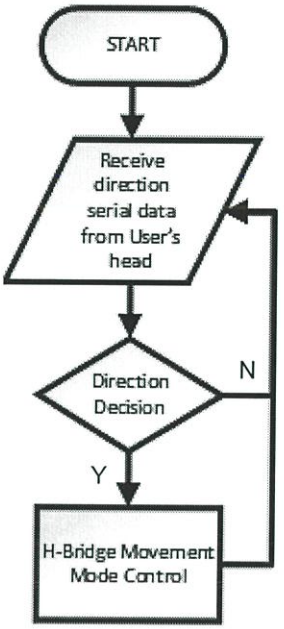
รูปที่ 3.25 แผนผังการทำงานของโปรแกรมส่วนควบคุมทิศทางบนศีรษะ

ในรูปที่ 3.25 แสดงแผนผังการทำงานของโปรแกรมในส่วนควบคุมทิศทางเก้าอี้เข็นด้วยศีรษะ โดยเริ่มจากการอ่านค่าจากเซนเซอร์ ADXL345 โดยบอร์ดไมโครคอนโทรลเลอร์ Arduino Nano ซึ่งจะนำค่าข้อมูลที่อ่านได้นั้นไปคำนวณหาค่ามุมเอียงที่เกิดขึ้น จากนั้นนำค่ามุมเอียงศีรษะที่คำนวณได้ไปเข้าเงื่อนไขทิศทางการเคลื่อนที่ จากนั้นจึงส่งค่าทิศทางไปยังบอร์ดไมโครคอนโทรลเลอร์ที่อยู่กับเก้าอี้เข็น ซึ่งเป็นส่วนควบคุมชุดกลไกขับเคลื่อน

เก้าอี้เข็นผ่านโมดูลสื่อสารไร้สายบลูทูธ ในการเชื่อมต่อในส่วนควบคุมทิศทางเก้าอี้เข็นด้วยศีรษะ แสดงได้ดังรูปที่ 3.26



รูปที่ 3.26 การเชื่อมต่อในส่วนควบคุมทิศทางเก้าอี้เข็นด้วยศีรษะ

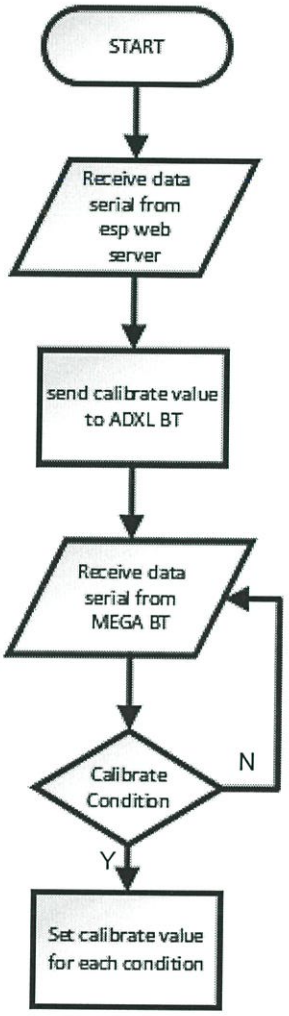


รูปที่ 3.27 แผนผังการทำงานของโปรแกรมส่วนควบคุมทิศทางที่เก้าอี้เข็น

ในรูปที่ 3.27 แสดงแผนผังการทำงานของโปรแกรมส่วนควบคุมทิศทางที่เก้าอี้เข็น โดยเริ่มจากบอร์ดไมโครคอนโทรลเลอร์ Arduino MEGA รับค่าทิศทางที่ส่งมากจากส่วนบนศีรษะผ่านโมดูลบลูทูธ จากนั้นจึงนำค่าที่รับมาไปเข้าเงื่อนไขตัดสินใจควบคุมทิศทางเก้าอี้เข็นและทำการควบคุมการขับเคลื่อนมอเตอร์ตามทิศทางต่อไป

5) ระบบปรับเทียบค่าเงื่อนไขทิศทางของบุคคล

เนื่องจากผู้ใช้งานแต่ละคนมีสรีระที่ต่างกันและมีการเอียงศีรษะด้วยมุมเอียงมากน้อยแตกต่างกันไป เพื่อความสะดวกในการใช้งานเก้าอี้เข็นจึงออกแบบส่วนการปรับเทียบค่าเงื่อนไขมุมเอียงศีรษะของบุคคล โดยผู้ดูแลจะควบคุมการปรับเทียบผ่านแอปพลิเคชันให้สามารถตั้งค่าเงื่อนไขมุมการเอียงศีรษะเพื่อควบคุมทิศทางเก้าอี้เข็นได้ตามที่ผู้ใช้งานต้องการ โดยโปรแกรมการปรับเทียบมีการทำงานดังแผนผังในรูปที่ 3.28



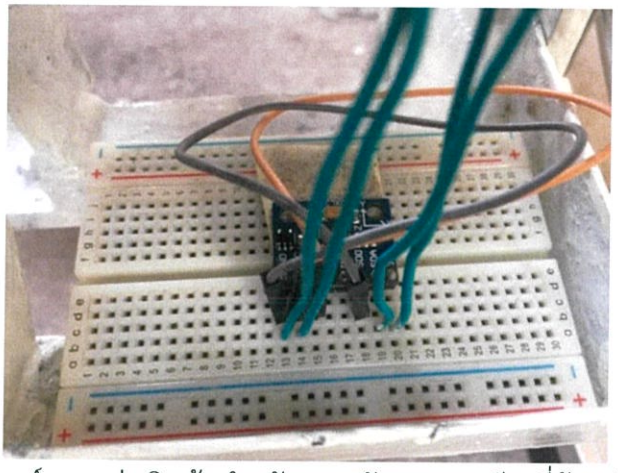
รูปที่ 3.28 แผนผังการทำงานของโปรแกรมปรับเทียบค่าเงื่อนไขทิศทางของบุคคล

จากแผนผังโปรแกรมจะเริ่มจากไมโครคอนโทรลเลอร์ในส่วนควบคุมชุดกลไกขับเคลื่อนจะส่งค่าผานบลูทูธดังตารางที่ 3.8 ไปยังไมโครคอนโทลเลอร์ในส่วนควบคุมบนศีรษะผู้ใช้งาน ให้เข้าสู่โหมดการปรับเทียบเงื่อนไขทิศทาง ซึ่งจะทำการบันทึกค่ามุมเอียงที่ผู้ใช้งานเอียงศีรษะในขณะนั้น เก็บเป็นค่าเงื่อนไขในทิศทางที่กำหนด โดยจะมีการปรับเทียบด้วยกันทั้งหมด 4 ทิศ ได้แก่ เคลื่อนที่ไปข้างหน้า, ถอยหลัง, เลี้ยวซ้าย และเลี้ยวขวา

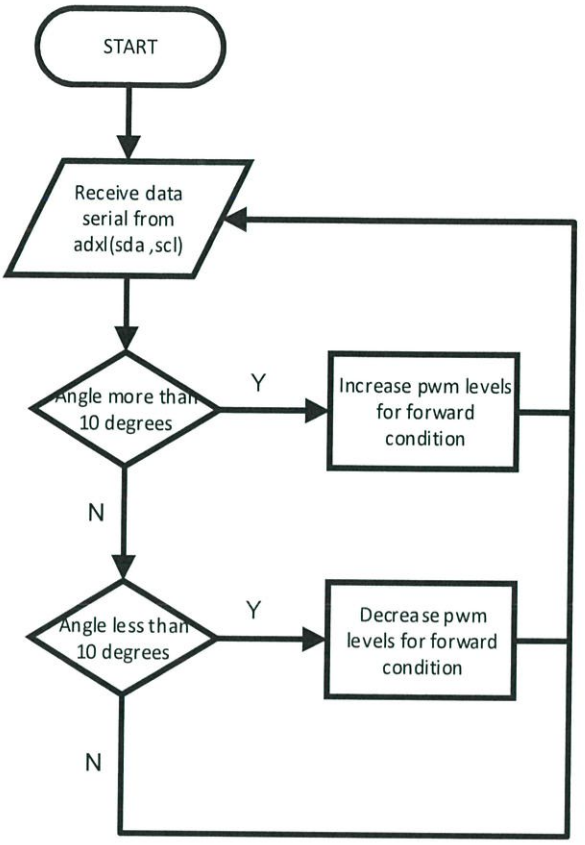
6) ระบบชดเชยความลาดเอียงเมื่อเคลื่อนที่บนพื้นลาดชัน

เนื่องจากแรงโน้มถ่วงส่งผลต่อโดยตรงเซนเซอร์ความเร่งเชิงเส้น เมื่อผู้ใช้งานเก้าอี้เข็นนำรถเข็นขึ้นหรือลงพื้นที่ลาดเอียง ด้วยระดับความลาดเอียงของพื้นที่เปลี่ยนไปย่อมส่งผลต่อการตรวจวัดมุมเอียงของเซนเซอร์บนศีรษะอย่างหลีกเลี่ยงไม่ได้ ผู้จัดทำจึงออกแบบ

ระบบชดเชยความลาดเอียงโดยจะติดตั้งเซนเซอร์ความเร่งเชิงเส้นอีกตัวติดไว้กับเก้าอี้เข็นดังรูปที่ 3.29 จะทำการอ่านค่าเซนเซอร์และคำนวณมุมเอียงที่เก้าอีเข็น ซึ่งหากตรวจพบการเคลื่อนขึ้นพื้นลาดเอียงจะมีการเพิ่มความเร็วรอบของมอเตอร์ และหากตรวจพบการเคลื่อนลงพื้นลาดเอียงจะมีการชะลอความเร็วรอบของมอเตอร์ ในรูปที่ 3.30 แสดงแผนผังการทำงานของโปรแกรมระบบชดเชยความลาดเอียง



รูปที่ 3.29 เซนเซอร์ความเร่งเชิงเส้นสำหรับตรวจวัดทางลาดเอียงที่ด้านล่างของเก้าอี้เข็น



รูปที่ 3.30 แผนผังการทำงานของโปรแกรมระบบชดเชยความลาดเอียง

3.1.3 การออกแบบระบบแจ้งเตือนการชน

รูปที่ 3.31 โมดูลเซนเซอร์ตรวจวัดระยะอัลตราโซนิค HC-SR04

ผู้จัดทำได้เลือกใช้โมดูลเซนเซอร์อัลตราโซนิค HC-SR04 ดังรูปที่ 3.31 ในส่วนของระบบการแจ้งเตือนการชนวัตถุทางด้านหลังเก้าอี้เข็น โดยในขณะที่ผู้ใช้งานได้มีการควบคุมทิศทางของเก้าอี้เข็นให้เคลื่อนที่ถอยหลังนั้น ผู้ใช้งานจะไม่สามารถมองเห็นด้านหลังได้ จึงใช้โมดูลดังกล่าว

ในระบบเพื่อตรวจจับระยะของวัตถุทางด้านหลังเก้าอี้เข็นและส่งเสียงแจ้งเตือนให้กับผู้ใช้งานได้ทราบ

สำหรับโมดูลเซนเซอร์ HC-SR04 จะประกอบไปด้วยตัวส่งอัลตราโซนิค (ขา Trig) และตัวรับอัลตราโซนิค (ขา Echo) ในการใช้งานโมดูลจะต้องป้อนสัญญาณพัลส์ที่มีความกว้างพัลส์ 10 us เข้าที่ขา Trig ก่อน จากนั้นโมดูลจะเริ่มส่งสัญญาณคลื่นเสียงอัลตราโซนิคที่มีความถี่ 40 kHz จำนวน 8 ลูกคลื่นออกไปทางตัวส่งอัลตราโซนิคด้วยอัตราเร็วในอากาศที่อุณหภูมินั้นๆ ซึ่งสามารถคำนวณอัตราเร็วเสียงในอากาศได้จากสมการที่ (3.12) จากนั้นตัวรับอัลตราโซนิคจะได้รับสัญญาณสะท้อนจากวัตถุที่อยู่ในระยะ โดยความกว้างพัลส์ของสัญญาณสะท้อนที่ได้รับนี้จะขึ้นกับระยะทางของวัตถุที่อยู่ห่างออกไป

$$v_{air} = 331 + (0.606T) \tag{3.12}$$

โดย v_{air} คือ อัตราเร็วเสียงในอากาศ (m/s)

T คือ อุณหภูมิ (c°)

ตัวรับอัลตราโซนิคจะคอยรับสัญญาณที่สะท้อนจากการกระทบกับวัตถุ เมื่อทราบอัตราเร็วในการเคลื่อนที่ของสัญญาณเสียงและเวลาที่ใช้ในการเดินทางไปกลับ จะสามารถคำนวณหาระยะห่างของวัตถุนั้นๆได้จากสมการที่ (3.13)

$$s = v_{air}\frac{t}{2} \tag{3.13}$$

โดย s คือ ระยะห่างระหว่างเซนเซอร์กับวัตถุที่คลื่นเสียงตกกระทบ (m)

v_{air} คือ อัตราเร็วเสียงในอากาศ (m/s)

t คือ เวลาที่คลื่นเสียงใช้ในการเดินทางไปกลับ (s)

ในการใช้งานโมดูลเซนเซอร์ HC-SR04 วัดระยะทางระหว่างเก้าอี้เข็นและวัตถุที่อยู่ห่างออกไป โดยทำการเชื่อมต่อโมดูลเข้ากับบอร์ดไมโครคอนโทรลเลอร์ Arduino Mega ตามการเชื่อมต่อในตารางที่ 3.7 โดยที่อุณหภูมิห้อง 25 c° อัตราเร็วของเสียงในอากาศจะคำนวณได้ดังนี้

จากสมการที่ (1)

$$v_{air} = 331 + (0.606T)$$

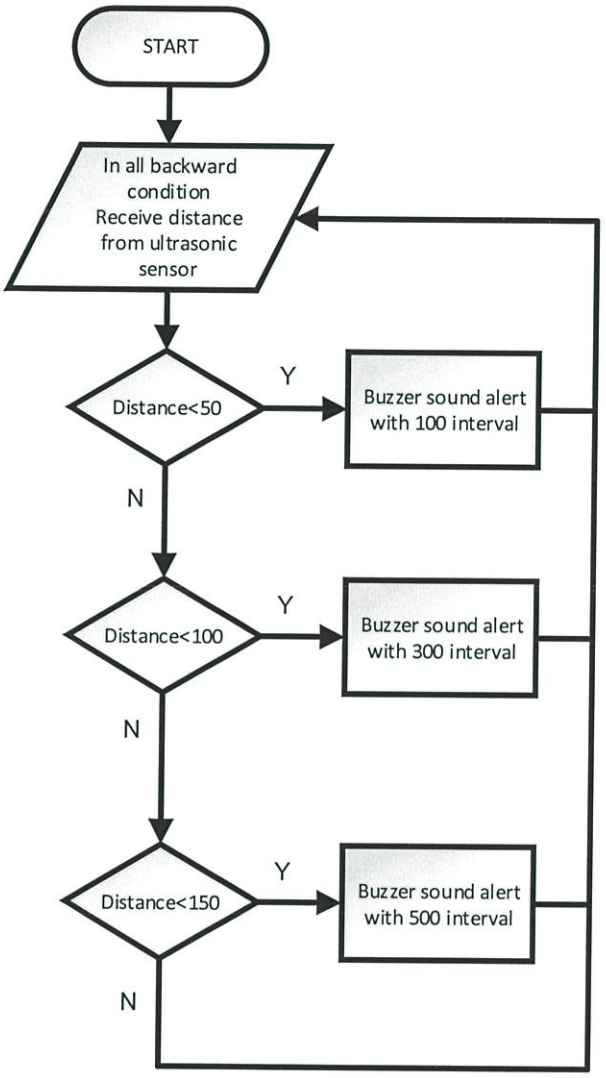
$$v_{air} = 331 + (0.606 \times 25)$$

$$v_{air} = 346.15\frac{m}{s}$$

ตารางที่ 3.7 การเชื่อมต่อโมดูล HC-SR04 เข้ากับบอร์ดไมโครคอนโทรลเลอร์ Arduino MEGA

HC-SR04 Pin Symbol	Arduino MEGA Pin
VCC	GND
Trig	5V
Echo	A3
GND	A4

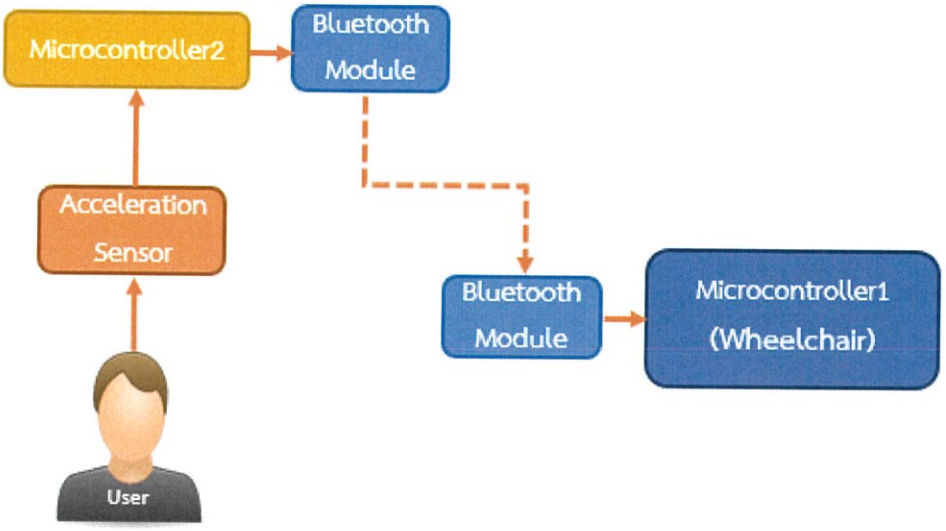
ในการพัฒนาโปรแกรมระบบแจ้งเตือนการชนนั้นจะใช้คำสั่ง pulseIn() ในการนับช่วงความกว้างพัลส์ของสัญญาณเสียงที่รับได้จากขา Echo เมื่อทราบความเร็วเสียงในอากาศและความกว้างพัลส์ของสัญญาณเสียงที่ได้รับแล้วจะสามารถคำนวณระยะห่างระหว่างเซนเซอร์กับวัตถุที่ห่างออกไปได้ด้วยสมการที่ (3.13) จากนั้นจะค่าระยะห่างที่คำนวณได้เข้าเงื่อนไขระยะห่างที่กำหนดไว้เพื่อส่งเสียงแจ้งเตือนผู้ใช้งานผ่านโมดูลเสียง (Active Buzzer Module) ตามระยะความห่างของวัตถุ ในรูปที่ 3.32 แสดงแผนผังการทำงานของโปรแกรมในส่วนระบบแจ้งเตือนการชน



รูปที่ 3.32 แผนผังการทำงานของโปรแกรมระบบแจ้งเตือนการชน

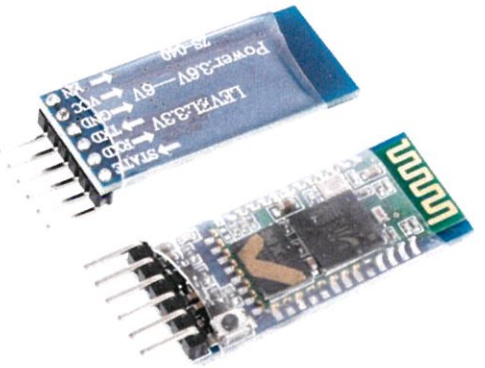
3.1.4 การออกแบบการเชื่อมต่อสื่อสารในเก้าอี้เข็น

3.1.4.1 การเชื่อมต่อสื่อสารระหว่างส่วนควบคุมทิศทางบนศีรษะและส่วนควบคุมชุดกลไกขับเคลื่อนเก้าอี้เข็น



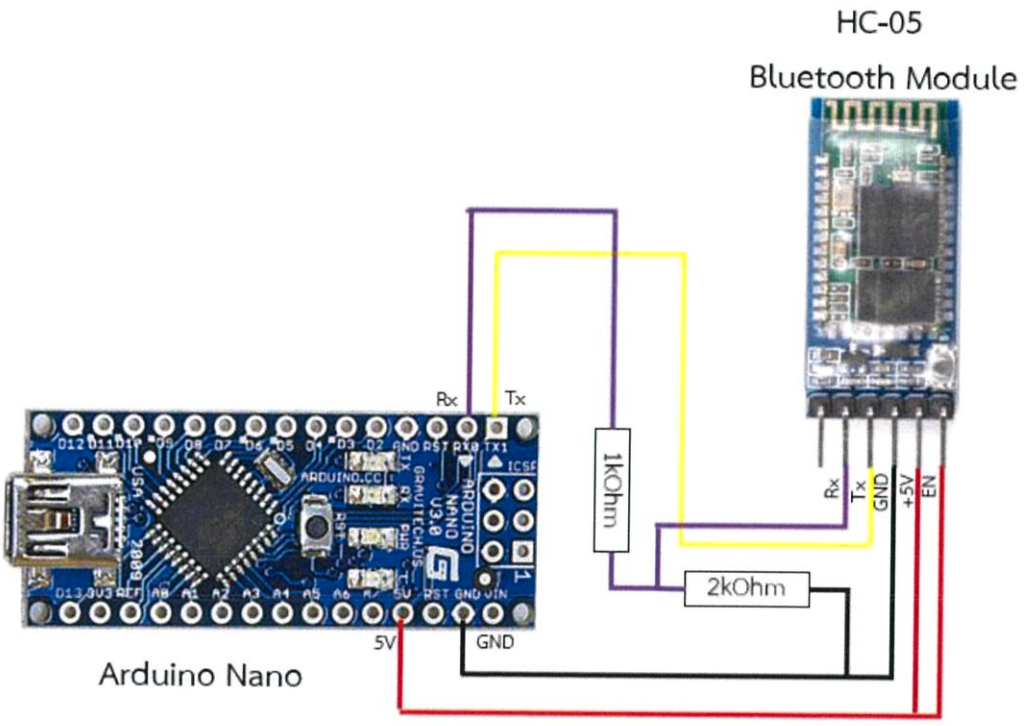
รูปที่ 3.33 การเชื่อมต่อสื่อสารระหว่างส่วนควบคุมทิศทางบนศีรษะและส่วนควบคุมชุดกลไกขับเคลื่อนเก้าอี้เข็น

จากรูปที่ 3.33 แสดงแผนภาพรวมของการเชื่อมต่อสื่อสารระหว่างบอร์ดไมโครคอนโทรลเลอร์ในส่วนควบคุมทิศทางบนศีรษะ และไมโครคอนโทรลเลอร์ในส่วนควบคุมชุดกลไกขับเคลื่อนที่อยู่กับเก้าอี้เข็น กำหนดให้มีภาคส่งและภาครับ โดยที่ภาคส่งคือส่วนที่อยู่กับศีรษะผู้ใช้งาน โมดูลบลูทูธในส่วนนี้จะส่งค่าทิศทางจากไมโครคอนโทรลเลอร์ (Microcontroller2) กำหนดให้โมดูลบลูทูธส่วนนี้ทำหน้าที่เป็น Slave Mode สำหรับในส่วนของภาครับที่อยู่กับเก้าอี้เข็น โมดูลบลูทูธในส่วนนี้จะรับค่าทิศทางที่ส่งมาจากโมดูลบลูทูธในภาคส่ง กำหนดให้โมดูลบลูทูธส่วนนี้ทำหน้าที่เป็น Master Mode



รูปที่ 3.34 โมดูลบลูทูธ HC-05

ผู้จัดทำได้เลือกใช้โมดูลสื่อสารไร้สาย HC-05 ดังรูปที่ 3.34 ในระบบการควบคุมทิศทางการเคลื่อนที่ของเก้าอี้เข็นด้วยศีรษะ ในการจะใช้งานโมดูลได้นั้นจะต้องตั้งค่าโมดูลผ่านคำสั่ง AT โดยต่อวงจรกับไมโครคอนโทรลเลอร์ดังรูปที่ 3.35 เพื่อให้โมดูลเข้าสู่ AT Command Mode โดยสามารถสังเกตได้จากไฟกระพริบบนตัวโมดูลเมื่อเข้าสู่โหมดนี้แล้วจะมีการกระพริบทุกๆ 2 วินาที ในรูปที่ 3.36 แสดงการสั่งการตั้งค่าโมดูลผ่าน Serial Monitor โดยได้ตรวจสอบเวอร์ชันของโมดูล ตำแหน่งแอดเดรสของอุปกรณ์ กำหนดค่า Role ของอุปกรณ์ว่าจะทำหน้าที่เป็น Master หรือ Slave กำหนดโหมดในการจับคู่และกำหนดแอดเดรสของอุปกรณ์ที่ต้องการจับคู่ด้วย รวมไปถึง Baud Rate ที่ใช้ในการสื่อสาร ในตารางที่ 3.8 สรุปการตั้งค่าโมดูลสื่อสารไร้สายบลูทูธทั้งภาคส่งและภาครับ



รูปที่ 3.35 การต่อวงจรตั้งค่าโมดูล HC-05 ในโหมด AT Command

COM4		COM4	
SLAVE (ภาคส่งที่ศีรษะ)		MASTER (ภาครับที่เก้าอี้เข็น)	
OK	<<< AT	OK	<<< AT
+VERSION:2.0-20100601	<<< AT+VERSION?	+VERSION:2.0-20100601	<<< AT+VERSION?
OK		OK	
+ADDR:98d3:34:91135a	<<< AT+ADDR?	+ADDR:98d3:32:20f3ac	<<< AT+ADDR?
OK		OK	
+ROLE:0	<<< AT+ROLE?	+ROLE:1	<<< AT+ROLE?
OK		OK	
+PSWD:1234	<<< AT+PSWD?	+PSWD:1234	<<< AT+PSWD?
OK		OK	
+UART:115200,0,0	<<< AT+UART?	+UART:115200,0,0	<<< AT+UART?
OK		OK	
+CMOD:0	<<< AT+CMODE?	+CMOD:0	<<< AT+CMODE?
OK		OK	
+BIND:98d3:32:20f3ac	<<< AT+BIND??	+BIND:98d3:34:91135a	<<< AT+BIND??
OK		OK	

รูปที่ 3.36 การตั้งค่าโมดูล HC-05 ในโหมด AT Command

ตารางที่ 3.8 การตั้งค่าโมดูล HC-05 ทั้งภาคส่งและภาครับ

คำสั่ง AT	โมดูล HC-05 ในภาคส่ง	โมดูล HC-05 ในภาครับ
AT+ADDR? (Bluetooth Address)	98d3:34:91135a	98d3:32:20f3ac
AT+ROLE= (Module Role)	0 (Slave Mode)	1 (Master Mode)
AT+UART= (Baud Rate, Stop Bit, Parity Bit)	115200,0,0 (1 stop bit, None Parity bit)	115200,0,0 (1 stop bit, None Parity bit)
AT+CMODE= (Connection Mode)	0 (Connect to Specified ADDR)	0 (Connect to Specified ADDR)
AT+BIND= (bind Bluetooth ADDR)	98d3:32:20f3ac	98d3:34:91135a

ในตารางที่ 3.9 แสดงการกำหนดค่าแทนทิศทางที่ใช้ส่งผ่านบลูทูธไปยังบอร์ดไมโครคอนโทรลเลอร์ส่วนควบคุมชุดขับเคลื่อน ทั้งนี้โปรแกรมจะมีการกำหนดให้ส่วนควบคุมทิศทางทำการปรับเทียบค่าเงื่อนไขทิศทางส่วนบุคคลด้วย โดยในตารางที่ 3.10 แสดงค่าที่ใช้ในการส่งข้อมูลผ่านบลูทูธ ซึ่งกำหนดให้แทนคำสั่งสำหรับสั่งการส่วนควบคุมทิศทางบนศีรษะทำการปรับเทียบค่าเงื่อนไขทิศทาง

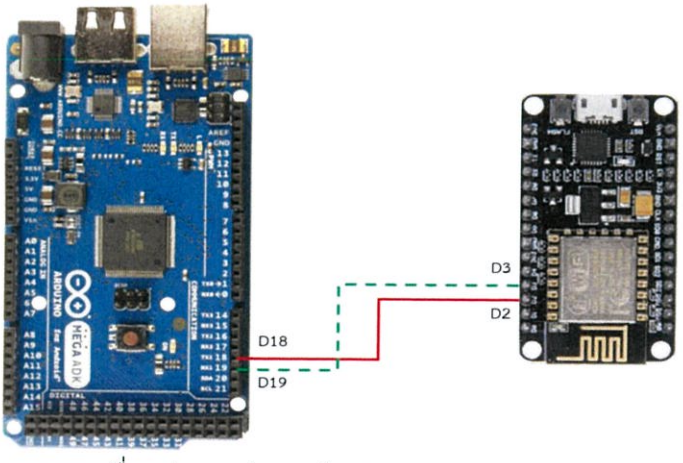
ตารางที่ 3.9 ค่าทิศทางที่ส่งผ่านบลูทูธไปยังบอร์ดไมโครคอนโทรลเลอร์ส่วนควบคุมชุดขับเคลื่อน

ทิศทาง	ค่าที่ส่ง
STAY	1
FORWARD	2
BACKWARD	3
RIGHT	4
LEFT	5
Tap	6

ตารางที่ 3.10 ค่าที่ส่งผ่านบลูทูธไปยังบอร์ดไมโครคอนโทรลเลอร์ส่วนควบคุมทิศทางบนศีรษะ

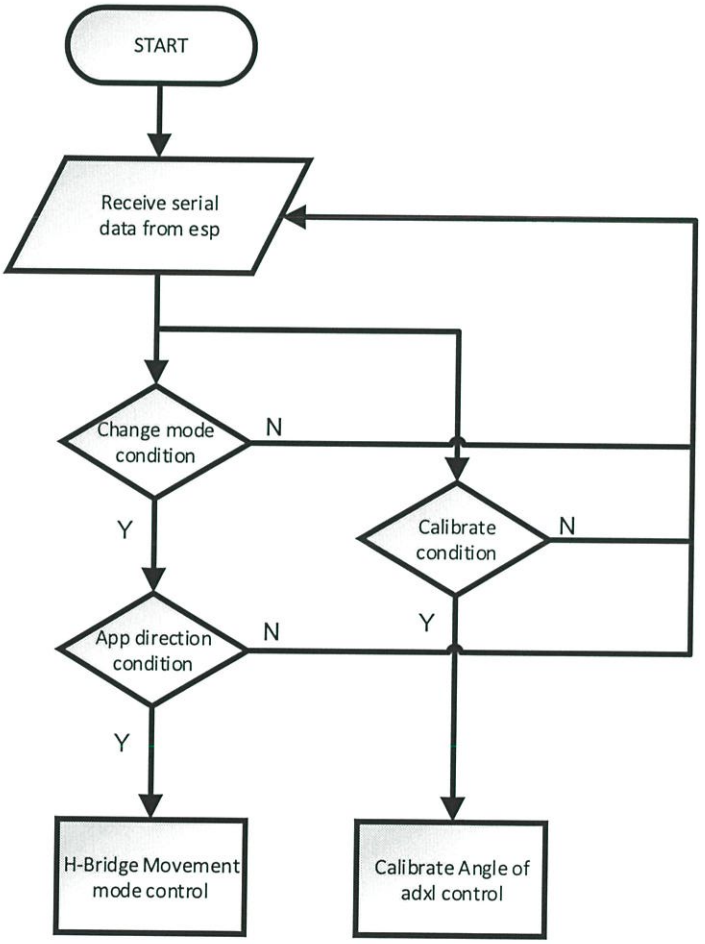
ค่าที่ส่ง	คำสั่ง
60	Calibrate FORWARD
65	Calibrate LEFT
70	Calibrate RIGHT
75	Calibrate BACKWARD
80	Reset Calibrate

3.1.4.2 การเชื่อมต่อระหว่างบอร์ดไมโครคอนโทรลเลอร์ Arduino MEGA และ NodeMCU



รูปที่ 3.37 การเชื่อมต่อระหว่างบอร์ด Arduino MEGA และ NodeMCU

ในการเชื่อมต่อดังรูปที่ 3.37 จะเป็นการเชื่อมต่อระหว่างบอร์ด Arduino MEGA และ NodeMCU ผ่านการสื่อสารแบบอนุกรม (Serial Connection) ซึ่งบอร์ด MEGA นั้นสามารถมีพอร์ตสื่อสารอนุกรมรองรับอยู่หลายพอร์ต โดยจะใช้พอร์ต Serial ในการเชื่อมต่อส่วนนี้ สำหรับบอร์ด Arduino MEGA จะทำการรับส่งค่า เช่น รับค่าทิศทางจากแอปพลิชันผ่าน NodeMCU หรือส่งค่าในการเรียกผู้ดูแลไปยังระบบฐานข้อมูล เป็นต้น ในรูปที่ 3.38 แสดงแผนผังการทำงานของโปรแกรมในส่วนการสื่อสารระหว่างบอร์ด Arduino MEGA และ NodeMCU

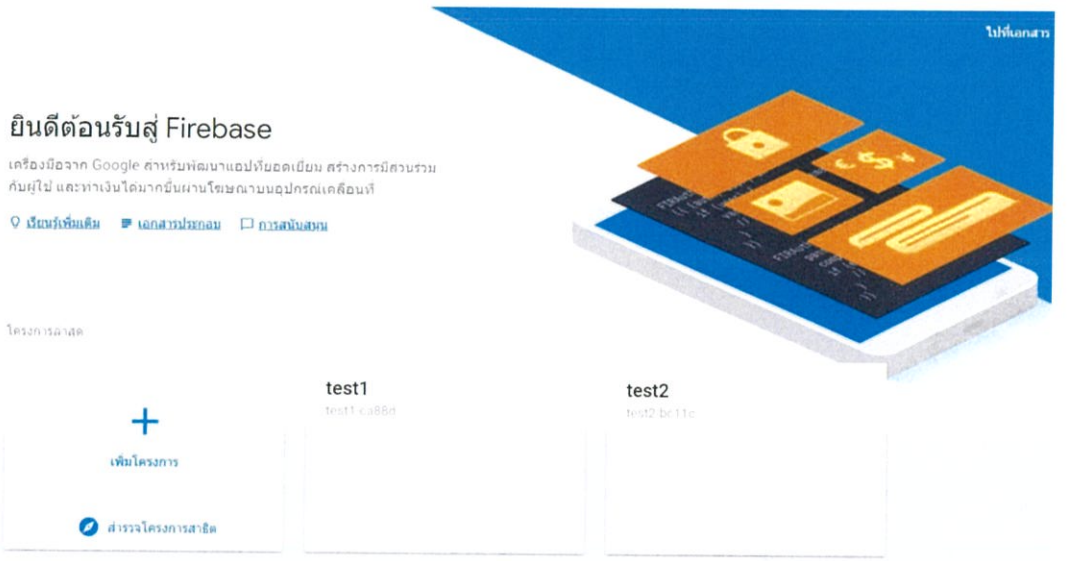


รูปที่ 3.38 แผนผังการทำงานของโปรแกรมการสื่อสารระหว่างบอร์ด Arduino MEGA และ NodeMCU

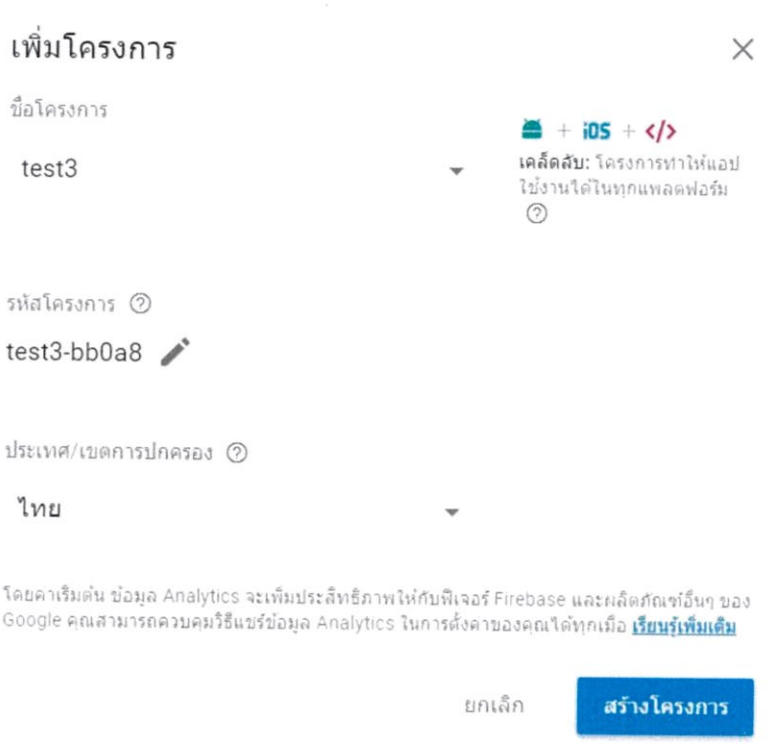
3.1.4.3 การใช้งาน Firebase Real Time Database

Firebase real time Database เป็น nosql cloud database ที่มีการจัดเก็บข้อมูลในรูปของ json มีการซิงค์ข้อมูลแบบเรียลไทม์กับทุกๆ อุปกรณ์ที่เชื่อมต่อแบบ Autonomous รองรับการทำงานเมื่อออฟไลน์ ข้อมูลจะถูกเก็บไว้จนกว่าจะกลับมาออนไลน์ระบบจะทำการซิงค์ข้อมูลแบบอัตโนมัติ จึงนำ Firebase real time Database มาเชื่อมต่อกับ NodeMCU เพื่อใช้ในการรับค่า Notification รวมไปถึงการรับที่อยู่ IP ที่ต้องใช้ในการเชื่อมต่อกับเก้าอี้เข็นด้วย

ในการใช้งานระบบฐานข้อมูล Firebase จะเริ่มทำการสร้าง Project ใหม่ โดยทำการเพิ่มโครงการที่หน้า Console ของ Firebase ดังรูปที่ 3.39 และกำหนดชื่อ Project ดังรูปที่ 3.40

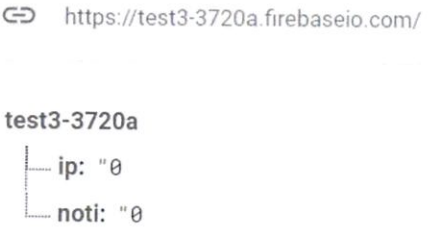


รูปที่ 3.39 หน้า Console ของFirebase



รูปที่ 3.40 การกำหนดชื่อ Project Firebase

ทำการกำหนด tag ที่ใช้ในการเก็บข้อมูลของผู้ใช้งาน โดยข้อมูลจะอยู่ในลักษณะที่เรียกว่า json tree ประกอบด้วย tag แรก คือ ที่อยู่ IP ซึ่งเป็น tag สำหรับข้อมูลเลข IP ที่ใช้ในการเชื่อมต่อระหว่างแอปพลิเคชันกับเก้าอี้เข็น โดยเมื่อเมื่อเก้าอี้เข็นเริ่มทำงาน NodeMCU จะทำการเชื่อมต่อกับเครือข่ายอินเทอร์เน็ตและจะส่งเลข IP ที่ใช้ในการเชื่อมต่อกับเก้าอี้เข็นมายัง Firebase ใน tag ที่ชื่อ IP แอปพลิเคชันจะทำการรับเลข IP นี้ไปเพื่อใช้ในการเชื่อมต่อและควบคุมเก้าอี้เข็น tag ที่สองคือ noti ซึ่งเป็น tag ที่ใช้ในการรับค่าตัวแปรเพื่อสั่งให้แอปพลิเคชันทำการแจ้งเตือนว่าเกิดปัญหากับผู้ใช้งานเก้าอี้เข็นดังรูปที่ 3.41



รูปที่ 3.41 การกำหนด tag ที่ใช้ในการเก็บข้อมูล Firebase

ก่อนที่จะนำ Firebase ไปใช้งานจะต้องทำการกำหนดสิทธิ์การเข้าถึงข้อมูลใน Firebase ว่าผู้ใดจะสามารถอ่านหรือเขียนข้อมูลใน Firebase Project นี้ได้บ้าง โดยใช้ใช้ Firebase Authentication เป็นตัวกำหนดสิทธิ์การจำกัดการเข้าถึง โดยจะกำหนดให้เป็นแบบ Public ดังรูปที่ 3.42 การกำหนดสิทธิ์การเข้าถึงแบบ Public จะอนุญาตให้ทุกคนสามารถเข้าถึงข้อมูลนี้ได้โดยไม่ต้องยืนยันตัวตน (Authentication)

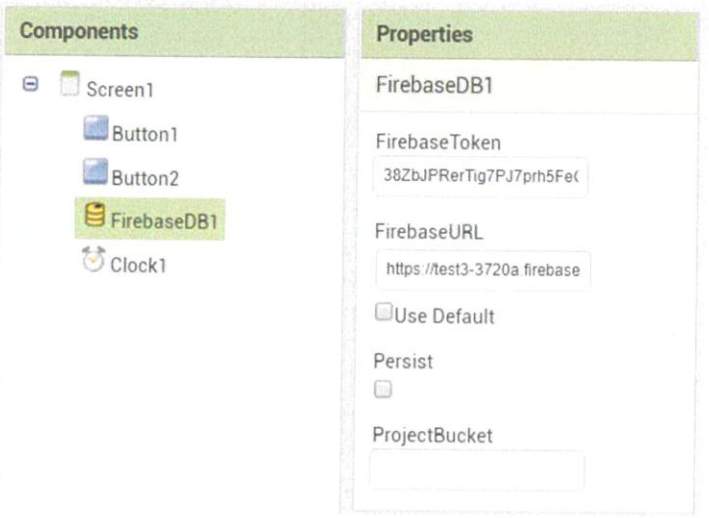
```
{
  "rules": {
    ".read": "auth != true",
    ".write": "auth != true"
  }
}
```

รูปที่ 3.42 ตัวกำหนดสิทธิ์เข้าถึง Firebase แบบ Public

การจะทำการเชื่อมต่อระหว่าง Firebase กับแอปพลิเคชันนั้นจำเป็นจะต้องทราบ FirebaseURL ซึ่งจะอยู่ด้านบนของ tag ข้อมูลในรูปที่ 3.41 และ Firebase Token ซึ่งจะอยู่ในหัวข้อข้อมูลลับฐานข้อมูลของ Firebase ดังรูปที่ 3.43 ข้อมูลทั้งสองนี้จะต้องทำการระบุใน Properties ของ Firebase Component ใน MIT App inventor2 ดังรูปที่ 3.44 หลังจากทำการระบุค่าทั้งสองแล้ว แอปพลิเคชันจะสามารถทำการเชื่อมต่อไปยัง Firebase เพื่ออ่านและเขียนข้อมูลได้

ฐานข้อมูล	ข้อมูลลับ
test3-3720a	38ZbJPRerTig7PJ7prh5FeCamTIecbBnDrWswUqq

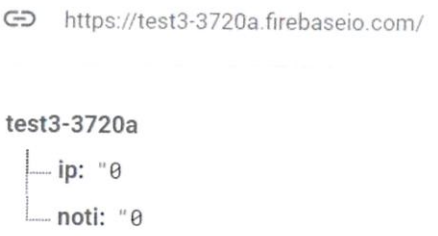
รูปที่ 3.43 ค่า Firebase Token ที่ใช้ในการเชื่อมต่อระหว่าง Firebase กับแอปพลิเคชัน



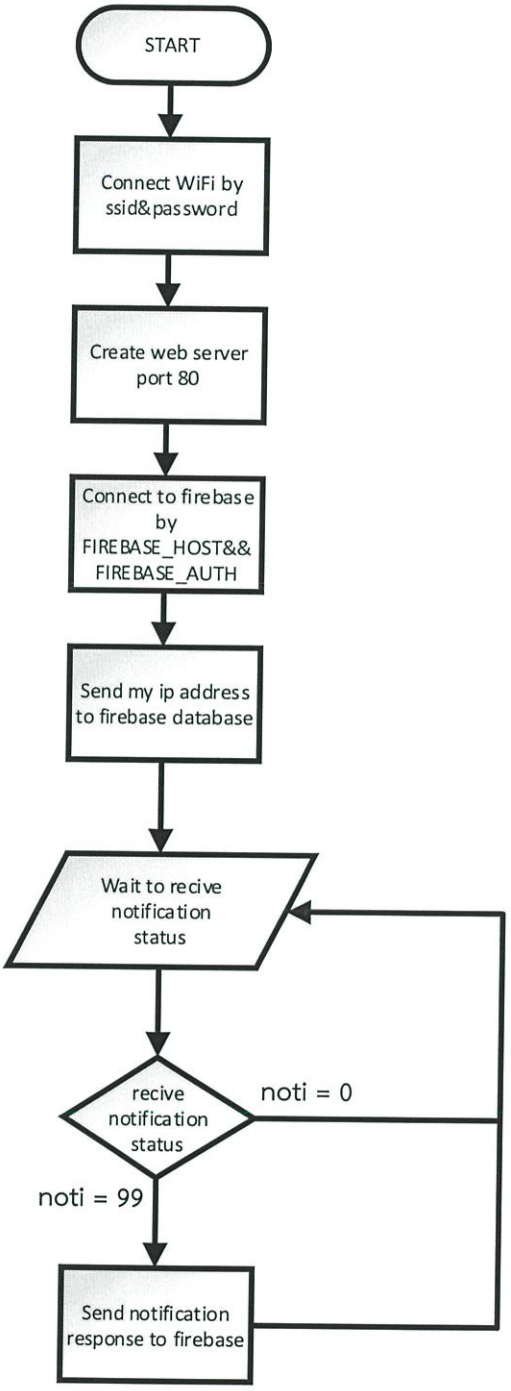
รูปที่ 3.44 การระบุ Firebase URL และ Firebase Token ใน MIT App inventor2

3.1.4.4 การเชื่อมต่อระหว่าง NodeMCU และฐานข้อมูล Firebase

การเชื่อมต่อระหว่าง NodeMCU และฐานข้อมูล Firebase ออกแบบขึ้นเพื่อรองรับการเชื่อมต่อกับแอปพลิเคชันในระยะที่ห่างออกไป ในการออกแบบจะให้ NodeMCU ทำการส่ง IP Address ที่ได้รับจาก Access Point ที่เชื่อมต่อไปยัง Firebase เพื่อให้แอปพลิชันดึงค่าที่อยู่ IP ดังกล่าวไปแสดงผลบนแอปพลิชัน ทั้งนี้เมื่อผู้ใช้งานต้องการเรียกผู้ดูแล NodeMCU จะส่งค่าการเรียกผู้ดูแลไปยังฐานข้อมูล โดยแอปพลิชันจะตรวจสอบสถานะของค่าดังกล่าว ซึ่งหากพบการเรียกผู้ดูแลจะมีการแจ้งเตือนขึ้นบนแอปพลิเคชัน ในรูปที่ 3.45 แสดงฐานข้อมูล Firebase ที่ประกอบไปด้วยตัวแปร ip ที่ใช้เก็บที่อยู่ IP ของ NodeMCU และ noti ที่ใช้เก็บค่าการเรียกผู้ดูแล ในรูปที่ 3.46 แสดงแผนผังการทำงานของโปรแกรมในส่วนการเชื่อมต่อสื่อสารระหว่าง NodeMCU และ Firebase



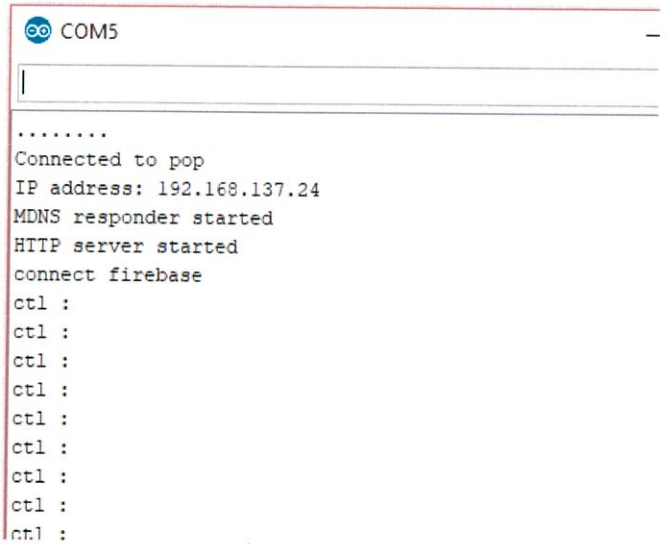
รูปที่ 3.45 ตัวแปร ip และ noti ในระบบฐานข้อมูล Firebase



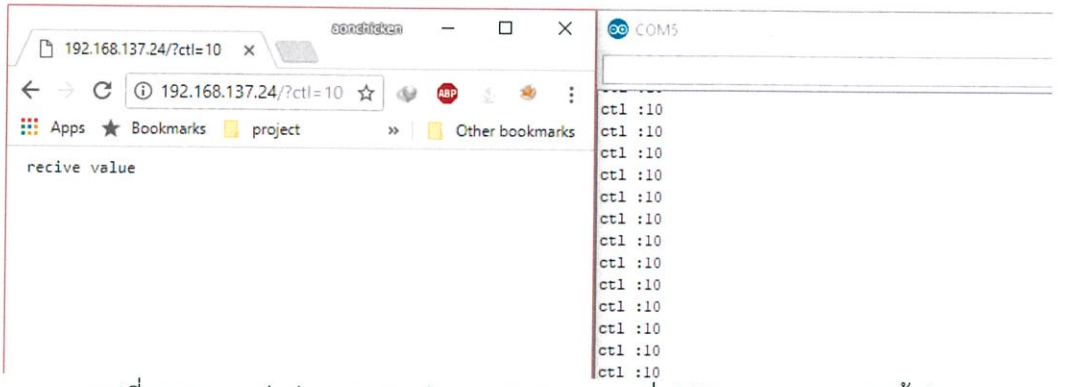
รูปที่ 3.46 แผนผังการทำงานของโปรแกรมการเชื่อมต่อสื่อสารระหว่าง NodeMCU และ Firebase

3.1.4.5 การเชื่อมต่อระหว่าง NodeMCU Web Server และแอปพลิเคชันบนสมาร์ตโฟน

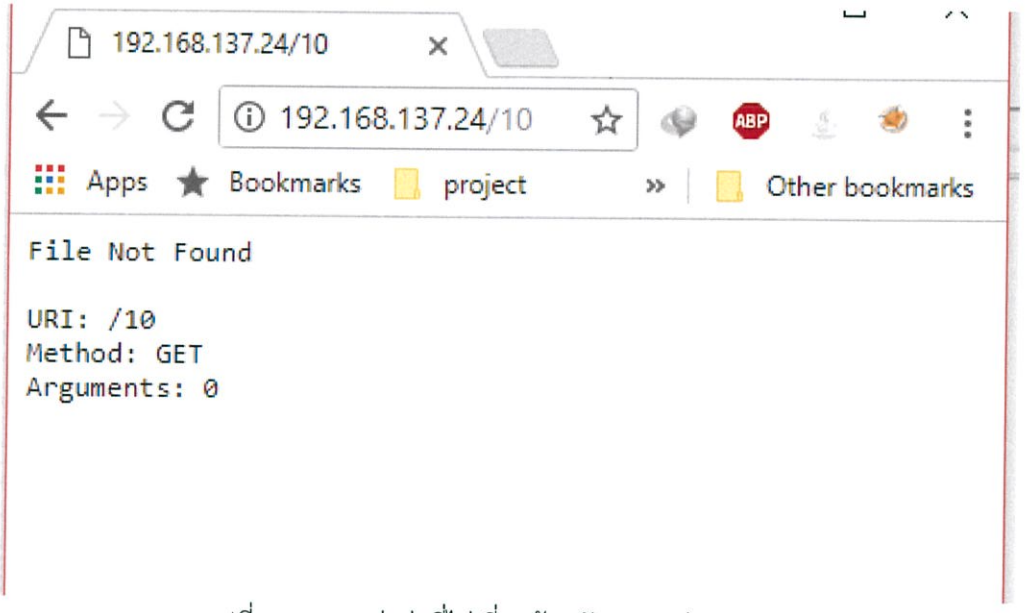
ในการเชื่อมต่อระหว่าง NodeMCU และแอปพลิชันบนสมาร์ตโฟนเพื่อใช้ในส่วนการควบคุมทิศการเคลื่อนที่ทางเก้าอี้เข็นผ่านแอปพลิเคชัน ซึ่งเป็นส่วนที่ต้องการการตอบสนองที่รวดเร็วและต่อเนื่อง จะเป็นการเชื่อมต่อกันผ่าน Web Server จะกำหนดให้ NodeMCU เป็น Web Server และแอปพลิเคชันบนสมาร์ตโฟนนั้นเป็น Client ของ NodeMCU ซึ่ง Web Server เป็น Server ที่ให้บริการเว็บไซต์แก่ Client ที่ร้องขอหน้าเว็บโดยใช้ Hypertext Transfer Protocol (HTTP) โดยจะนำ NodeMCU มาสร้างเป็น Web Server ที่ทำงานเช่นเดียวกับ TCP Server ที่เปิดใช้งาน Port 80 ไว้ เพื่อทำหน้าที่รอรับการร้องขอข้อมูลจาก Client (แอปพลิเคชัน) ซึ่งเป็นข้อมูลนั้นเป็นการเรียกขอหน้าเว็บไซต์จาก Web browser เพื่อการเข้าถึง File ที่อยู่ใน root โดยทั่วไปแล้วการ /(root) โดยใช้ HTTP เมื่อ Web Server ได้รับการร้องขอก็จะส่งข้อมูลที่ถูกร้องขอกลับไปยัง Client เพื่อนำไปแสดงผล แล้วจึงตรวจสอบข้อมูลที่ส่งเข้ามาเพื่อนำไปใช้ควบคุมทิศทางของเก้าอี้เข็นโดยผ่าน Arduino MEGA ต่อไป ในรูปที่ 3.47 Serial Monitor แสดงการเชื่อมต่อ Access Point และทำการสร้าง Web Server บน NodeMCU รูปที่ 3.48 แสดงการส่งค่า ctl=10 จากแอปพลิชันไปยัง NodeMCU ผ่าน Web Server เพื่อส่งให้ Arduino MEGA นำไปใช้ในการควบคุมเก้าอี้เข็น และรูปที่ 3.49 แสดงการส่งค่าอื่นๆที่ไม่เกี่ยวข้องในระบบเก้าอี้เข็นไปยัง Web Server



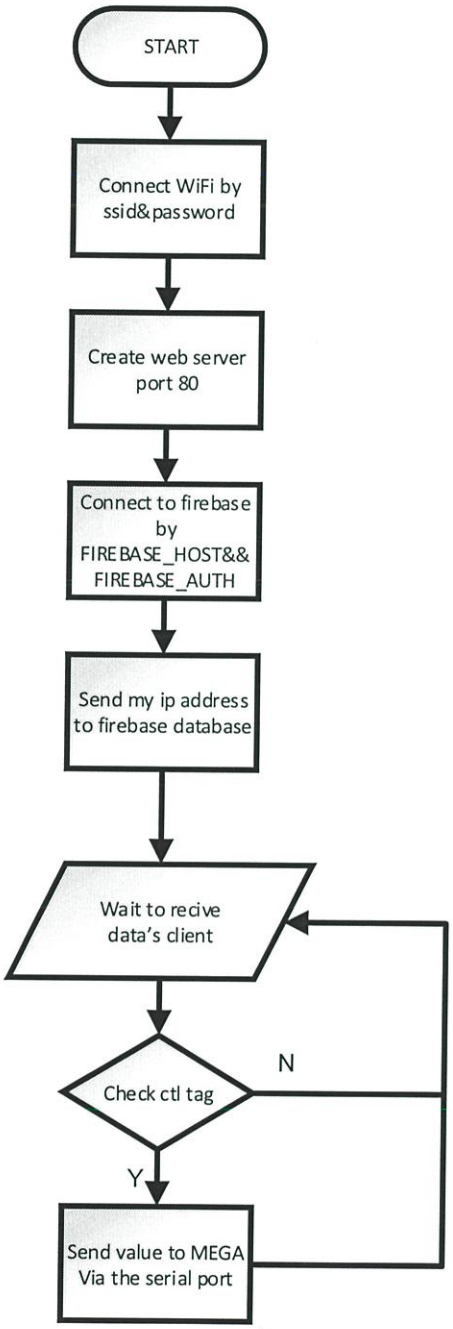
รูปที่ 3.47 NodeMCU ทำการเชื่อมต่อ Access Point และสร้าง Web Server



รูปที่ 3.48 การส่งค่า ctl=10 ผ่าน Web Server เพื่อใช้ในการควบคุมเก้าอี้เข็น



รูปที่ 3.49 การส่งค่าที่ไม่เกี่ยวข้องกับระบบผ่าน Web Server



รูปที่ 3.50 แผนผังการทำงานของโปรแกรมการสื่อสารระหว่าง NodeMCU และแอปพลิเคชัน

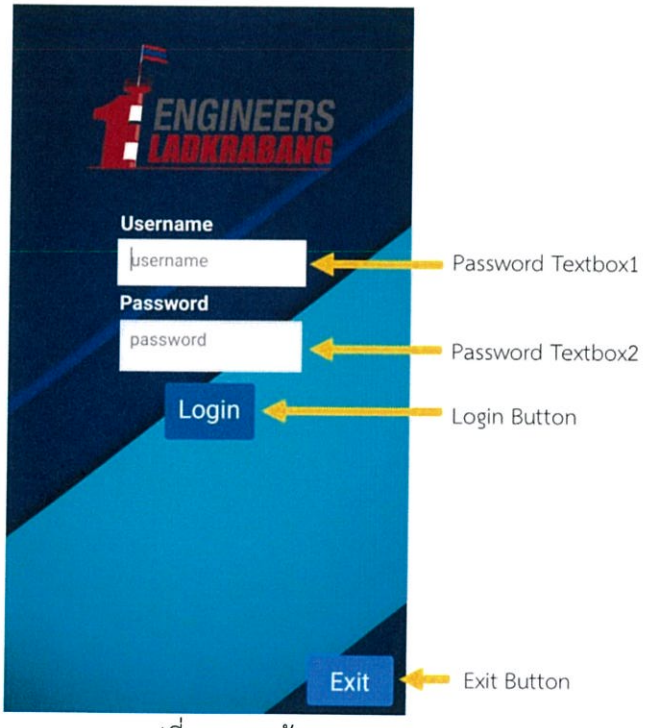
3.1.5 การออกแบบพัฒนาแอปพลิเคชัน MIT App Inventor 2

3.1.5.1 หน้าจอScreen1

ทำการพัฒนาแอปพลิเคชันด้วย MIT App Inventor 2 ซึ่งเป็นเครื่องมือที่ใช้สร้างแอปพลิเคชันที่ทำงานบนระบบปฏิบัติการแอนดรอยด์ (Android) ใช้หลักการพัฒนาซอฟต์แวร์เชิงคอมโพเนนต์ (Component-based Software Development) ในส่วนของแอปพลิเคชันนั้นจะทำหน้าที่เป็นส่วนที่ใช้ในการควบคุมการเคลื่อนที่ การปรับเลือกโหมดการใช้งานของเก้าอี้เข็น รวมไปถึงการปรับเทียบมุมองศาการเอียงของศีรษะของผู้ใช้งานแต่ละคนซึ่งมีมุมองศาการใช้งานที่แตกต่างกันออกไป

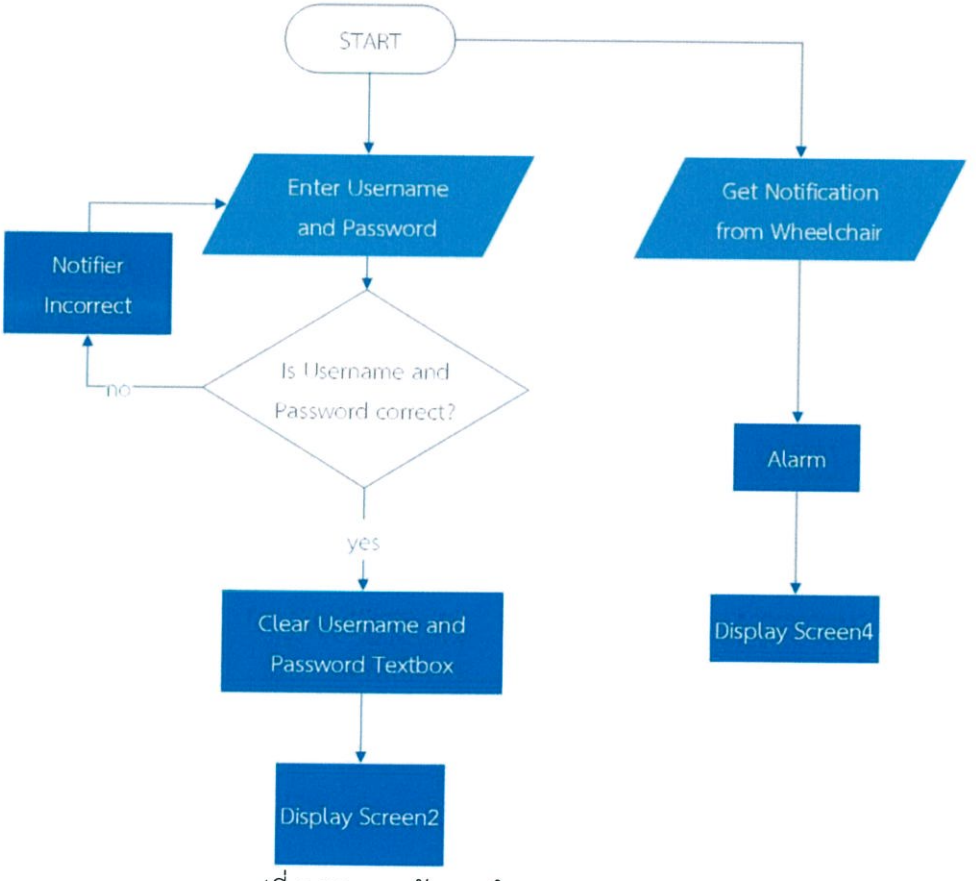
1) การออกแบบองค์ประกอบของหน้าจอ Screen1

หน้า Screen1 เป็นหน้าการใช้งานแรกเมื่อแอปพลิเคชันถูกเปิดใช้งาน ถูกออกแบบให้มีการกรอกรหัส Username และ Password เพื่อยืนยันตัวตนก่อนเข้าใช้งานแอปพลิเคชัน หน้า Screen1 ประกอบไปด้วย Password Textbox1 สำหรับกรอก Username และ Password Textbox 2 สำหรับกรอก Password ด้านล่างของ Password Textbox คือปุ่ม Login เพื่อเข้าใช้งานแอปพลิเคชัน และปุ่ม Exit เพื่อใช้ในการปิดการใช้งานของแอปพลิเคชัน ดังรูปที่ 3.51



รูปที่ 3.51 หน้า Screen1

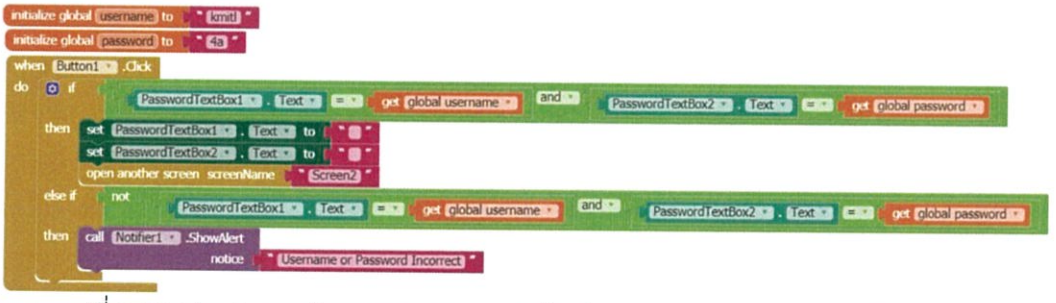
หน้าจอ Screen1 จะมีแผนผังการทำงาน (Flowchart) ตามรูปที่ 3.52 เมื่อเริ่มต้นใช้งานแอปพลิเคชันผู้ใช้จะต้องทำการกรอก Username และ Password หาก Username และ Password ที่ผู้ใช้งานแอปพลิเคชันกรอกไปถูกต้อง แอปพลิเคชันจะทำการเปิดหน้า Sceen2 ขึ้นมาซึ่งเป็นหน้าจอที่ใช้ในการควบคุมรถ หาก Username และ Password ไม่ถูกต้องจะขึ้นข้อความแจ้งเตือนว่า “Username or Password Incorrect” และผู้ใช้งานแอปพลิเคชันจะต้องทำการกรอก Username และ Password ใหม่ ในขณะที่กำลังเปิดหน้า Screen1 อยู่นั้น หากแอปพลิเคชันได้รับการแจ้งเตือนจากเก้าอี้เข็นโดยการส่งค่ามาที่ Firebase ก็จะทำการแสดงเสียงแจ้งเตือนรวมไปถึงเปิดระบบสั่นของสมาร์ทโฟนของผู้ใช้งานและจะเปิดหน้า Screen4 ซึ่งเป็นหน้าจอสำหรับแจ้งเตือนผู้ใช้งานขึ้นมา



รูปที่ 3.52 แผนผังการทำงานของ Screen1

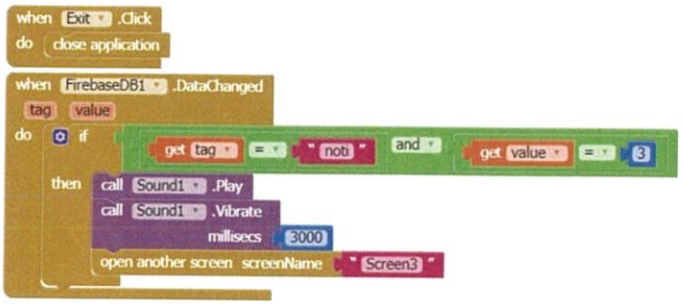
2) การเขียนโปรแกรมการทำงานของหน้าจอ Screen1

ทำการกำหนดเงื่อนไขของ Username และ Password ตาม Block การทำงานในรูปที่ 3.53 กำหนด Username และ Password ของแอปพลิเคชันเป็นคำว่า “kmitl” และ “4a” ตามลำดับจากนั้นทำการกำหนดเงื่อนไขของปุ่ม Login ในบล็อคคำสั่ง “when Button1 .Click” หากข้อมูลที่ผู้ใช้งานแอปพลิเคชันกรอกใน Password Textbox1 และ Password Textbox2 ตรงกับ Password และ Username ที่กำหนดจะทำการรีเซ็ตค่าใน Password Textbox ให้กลายเป็น Textbox ว่างเพื่อป้องกันการค้างค่าใน Textbox ซึ่งอาจจะทำให้ผู้อื่นสามารถใช้งานแอปพลิเคชันโดยไม่ได้รับอนุญาตได้ จากนั้นจะทำการเปิดหน้า Screen2 ซึ่งเป็นหน้าการใช้งานสำหรับการควบคุมเก้าอี้เข็น หากข้อมูลที่ผู้ใช้งานแอปพลิเคชันกรอกนั้นไม่ตรงกับ Username และ Password ที่ถูกต้อง จะแสดงข้อความแจ้งเตือนเป็นข้อความ Text ว่า “Username or Password Incorrect” และผู้ใช้งานจะต้องกรอก Username และ Password ใหม่ให้ถูกต้อง จึงจะสามารถเข้าใช้งานแอปพลิเคชันได้



รูปที่ 3.53 Block การทำงานของ Screen1 ในส่วนของ Username และ Password

กำหนดเงื่อนไขของปุ่ม “Exit” โดยเมื่อทำการกดปุ่มจะทำการปิดแอปพลิเคชันลง นอกจากฟังก์ชันการ Login หน้าจอ Screen1 จะเชื่อมต่อไปยัง FirebaseDB ซึ่งเป็นระบบฐานข้อมูลออนไลน์แบบเรียลไทม์ หากแอปพลิชันได้รับค่า 3 จาก FirebaseDB ซึ่งเป็นค่าที่ถูกส่งมาจากปุ่มเรียกผู้ดูแลฉุกเฉินซึ่งจะถูกติดตั้งอยู่บนเก้าอี้เข็น แอปพลิเคชันจะแสดงเสียงแจ้งเตือนรวมไปถึงเปิดระบบสั่นของอุปกรณ์สมาร์โฟนเป็นเวลา 3 วินาทีก่อนจะแสดงหน้าจอที่มีชื่อว่า Screen4 ซึ่งเป็นหน้าจอสำหรับการแจ้งเตือนผู้ใช้งานแอปพลิเคชัน

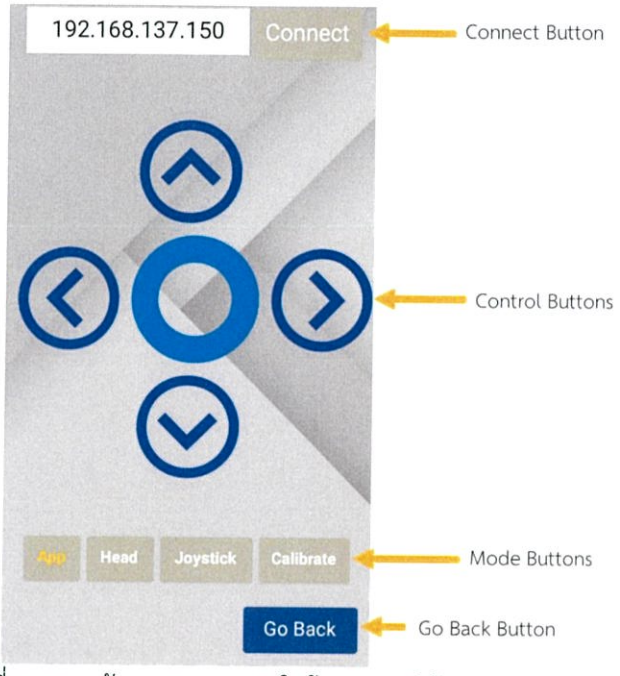


รูปที่ 3.54 Block การทำงานของ Screen1 ในส่วนของการแจ้งเตือนผู้ใช้งาน

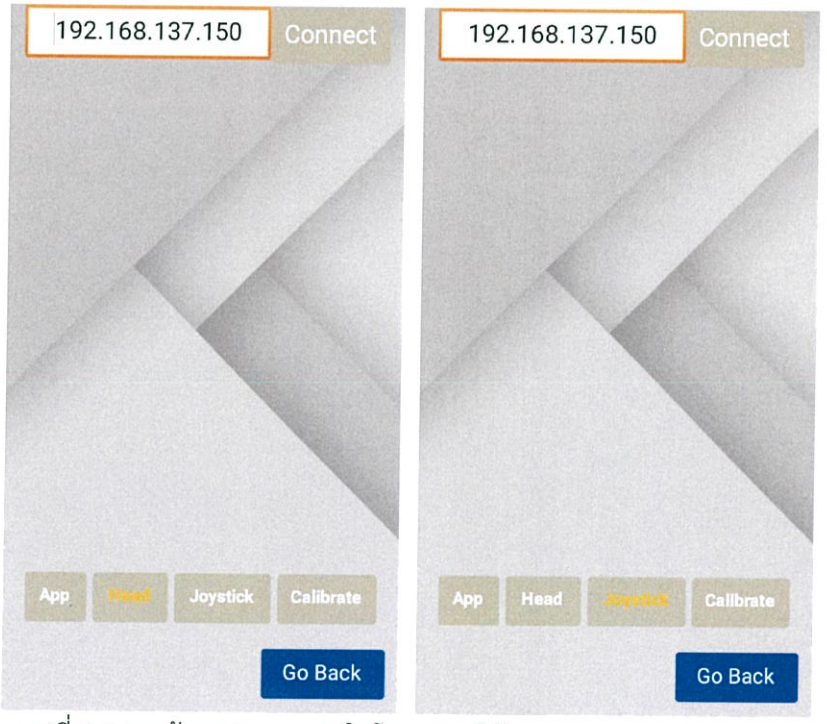
3.1.5.2 หน้าจอ Screen2

1) การออกแบบองค์ประกอบของหน้า Screen2

หน้าจอ Sceen2 เป็นหน้าการใช้งานหลักสำหรับการควบคุมการเคลื่อนที่รวมไปถึงการเลือกโหมดการใช้งานของเก้าอี้เข็น ด้านบนของหน้าจอประกอบไปด้วยปุ่ม Connect ซึ่งจะเป็นการรับค่า URL ที่ส่งมาจากเก้าอี้เข็นผ่านทาง Firebase เมื่อผู้ใช้ได้รับค่า URL นี้มาจะสามารถเชื่อมต่อกับ NodeMCU และจะสามารถควบคุมเก้าอี้เข็นผ่านสมาร์ทโฟนได้ บริเวณกลางหน้าจอจะประกอบไปด้วยปุ่มควบคุมทิศทางเก้าอี้เข็นจำนวน 5 ปุ่ม ได้แก่ Forward, Left, Right, Back และ Stop ด้านล่างของหน้าจอจะประกอบไปด้วยปุ่มเลือกโหมดการใช้งานจำนวน 4 ปุ่ม คือ App, Head, Joystick และ Calibrate เมื่อกดปุ่ม App จะเป็นการควบคุมเก้าอี้เข็นผ่านปุ่มควบคุมบนแอปพลิเคชัน ดังรูปที่ 3.55 เมื่อกดปุ่ม Head แอปพลิเคชันจะทำการซ่อนปุ่มควบคุมทั้ง 5 ปุ่มและจะเข้าสู่การควบคุมทิศทางการเคลื่อนที่ของเก้าอี้เข็นโดยใช้ ADXL345 ซึ่งเป็นเซนเซอร์วัดความเร่งเชิงเส้นที่จะติดอยู่บนศีรษะของผู้ใช้งานเก้าอี้เข็น เมื่อกดปุ่ม Joystick จะเป็นการควบคุมเก้าอี้เข็นโดยใช้ก้านควบคุมแบบอนาล็อกที่ถูกติดตั้งอยู่บนเก้าอี้เข็น ดังรูปที่ 3.56 และเมื่อกดปุ่ม Calibrate จะเป็นการเข้าสู่หน้าการปรับเทียบมุมองศาการใช้งานของ ADXL345 ซึ่งออกแบบมาสำหรับผู้ที่มีลักษณะการเอียงของศีรษะที่แตกต่างจากบุคคลทั่วไป เช่น ผู้พิการที่ไม่สามารถตั้งศีรษะให้อยู่ในลักษณะตรงตามปกติได้ จึงจำเป็นจะต้องทำการ Calibrate มุมองศาของศีรษะก่อนที่จะเริ่มใช้งานในโหมดHead และด้านล่างสุดของหน้าจอจะประกอบด้วยปุ่ม Go back เพื่อกลับไปยังหน้า Screen1

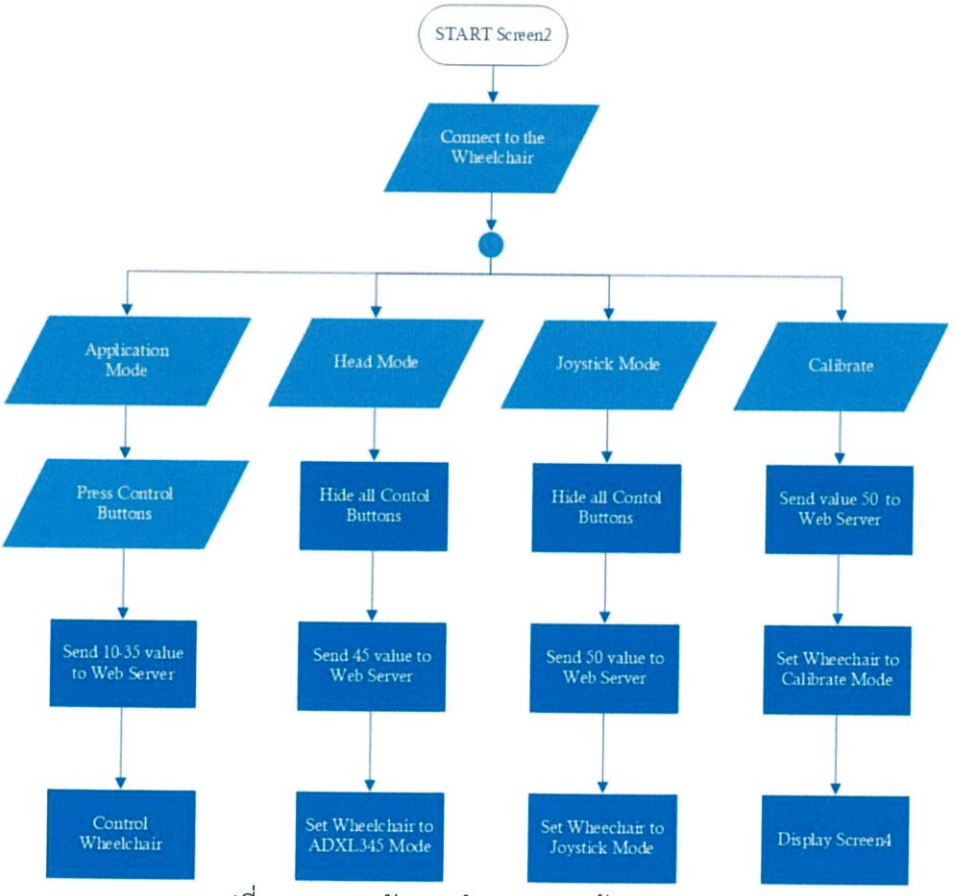


รูปที่ 3.55 หน้าจอ Screen2 ในโหมดการใช้งาน Application



รูปที่ 3.56 หน้าจอ Screen2 ในโหมดการใช้งาน Head และ Joystick

แผนผังการทำงานของหน้าจอ Screen2 แสดงดังรูปที่ 3.57 เมื่อทำการเปิดหน้า Screen2 ผู้ใช้จะต้องทำการเชื่อมต่อกับเก้าอี้เข็น โดยการกดปุ่ม Connect ที่อยู่ด้านบนของหน้าจอ เมื่อทำการกดเลือกโหมดการควบคุมเก้าอี้เข็นโดยใช้แอปพลิเคชัน แอปพลิเคชันจะทำการส่งตัวแปรctl เป็นค่าต่างๆไปยัง Web Server ของ NodeMCU ก็ต่อเมื่อผู้ใช้งานแอปพลิเคชันกดปุ่มควบคุมทั้ง 5 ปุ่มได้แก่ Forward, Left, Right, Back และ Stop ซึ่งค่าที่ส่งไปยัง Server นี้ NodeMCU จะส่งต่อไปยังบอร์ด ATmega32 เพื่อควบคุมให้เก้าอี้เข็นเคลื่อนที่ต่อไป กรณีที่ผู้ใช้งานแอปพลิเคชันเลือกโหมดการทำงานเป็น Head หรือ Joystick แอปพลิเคชันจะทำการซ่อนปุ่มไม่ให้ผู้ใช้งานสามารถควบคุมเก้าอี้เข็นโดยใช้แอปพลิเคชันได้ และใช้เซนเซอร์ความเร่งเชิงเส้นบนศีรษะหรือก้านควบคุมอนาล็อกในการควบคุมเก้าอี้เข็นแทน ในกรณีสุดท้ายคือการเข้าสู่โหมด Calibrate ซึ่งเป็นการปรับเทียบมุมองศาการเอียงศีรษะของผู้ใช้งานเก้าอี้เข็นแต่ละคน แอปพลิเคชันจะส่งค่าไปยัง Web Server เพื่อปรับโหมดการทำงานให้อยู่ในโหมด Calibrate และจะเปิดหน้าจอ Screen4 ขึ้นมาซึ่งเป็นหน้าจอที่ใช้ในการปรับเทียบมุมองศาการเอียงของศีรษะผู้ใช้งาน



รูปที่ 3.57 แผนผังการทำงานของหน้า Screen2

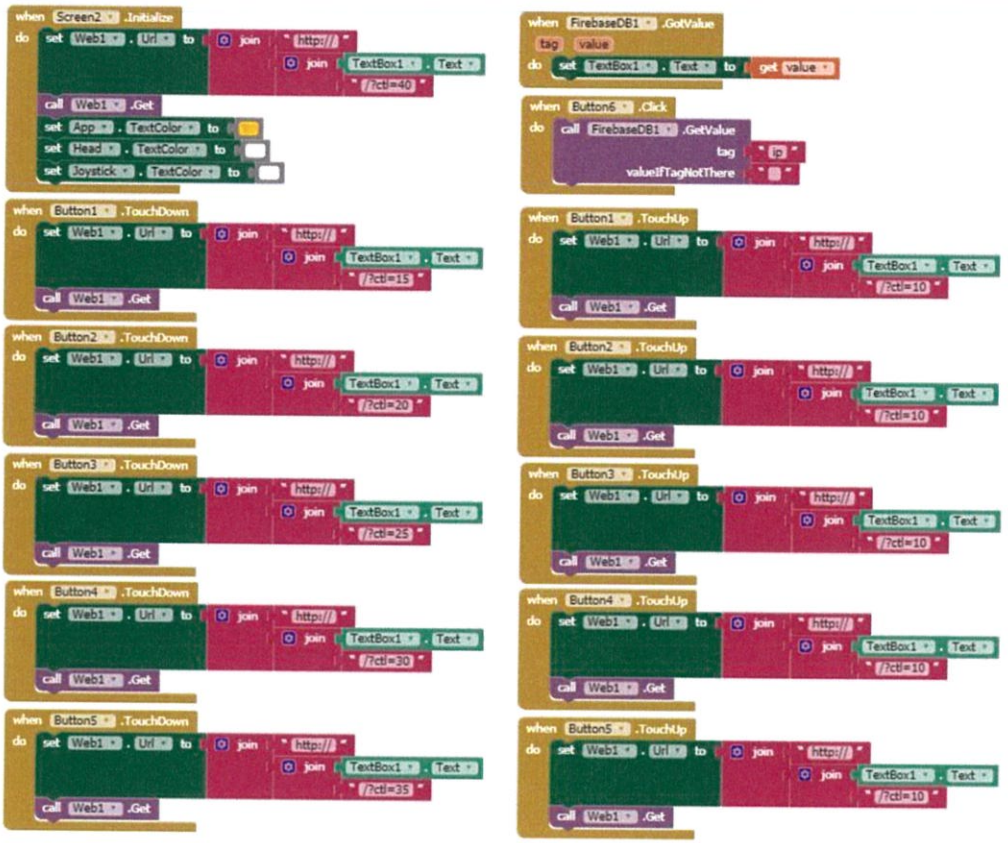
2) การเขียนโปรแกรมการทำงานของหน้า Screen2

Block การทำงานของหน้า Screen2 แสดงดังรูปที่ 3.58 แอปพลิเคชันจะเชื่อมต่อกับเก้าอี้เข็นผ่านการส่งค่า ctl เป็นค่าต่างๆไปยัง Web Server ของโมดูลไวไฟ NodeMCU เพื่อให้ NodeMCU นำข้อมูลส่งต่อไปยังบอร์ดไมโครคอนโทรลเลอร์ Arduino MEGA เพื่อประมวลผลควบคุมเก้าอี้เข็นอีกครั้งหนึ่ง ทำการกำหนด Block Initialize ซึ่งเป็นการกำหนดการทำงานเมื่อเปิดหน้า Screen2 ขึ้นมา โดยกำหนดให้ส่งตัวแปร ctl เป็น 40 ไปยัง Server ของ NodeMCU เพื่อให้เก้าอี้เข็นเข้าสู่โหมดควบคุมโดยใช้แอปพลิเคชัน และกำหนดให้ text colour ของปุ่ม App เป็นสีเหลืองเพื่อแสดงว่าผู้ใช้งานอยู่ในโหมดการใช้งานนี้ ปุ่ม Connect ที่อยู่ด้านบนของแอปพลิเคชันจะเชื่อมต่อไปยัง Firebase เมื่อเก้าอี้เข็นเริ่มทำงานจะส่งที่อยู่ IP ไปยัง Firebase และแอปพลิเคชันจะรับที่อยู่ IP นี้ทันทีที่กดปุ่ม Connect เพื่อใช้ในการเชื่อมต่อกับเก้าอี้เข็น เมื่อผู้ใช้งานกดปุ่มควบคุมรถทั้ง 5 ปุ่ม ได้แก่ Forward, Left, Right, Back และ Stop จะทำการส่งค่า ctl ที่แตกต่างกันไปยัง Server ดังตารางที่ 3.11

ตารางที่ 3.11 ค่า ctl ที่แอปพลิเคชันส่งไปยัง Server ของ NodeMCU ในการควบคุมทิศทางเก้าอี้เข็น

Ctl Variable	Condition
10	Stay
15	Forward
20	Left
25	Break
30	Right
35	Back

จาก Block การทำงาน การ Touchdown ปุ่มควบคุมทิศทางจะส่งค่าที่ทำให้รถอยู่ในเงื่อนไขที่ทำให้รถเกิดการเคลื่อนที่ไปในทิศทางใดทิศทางหนึ่งรวมไปถึงการเบรก ประกอบไปด้วย Forward, Left, Right, Back และ Stop แต่เมื่อใดที่ทำการ Touch Up ปุ่มควบคุม จะกำหนดให้อยู่ในเงื่อนไข Stay เสมอ



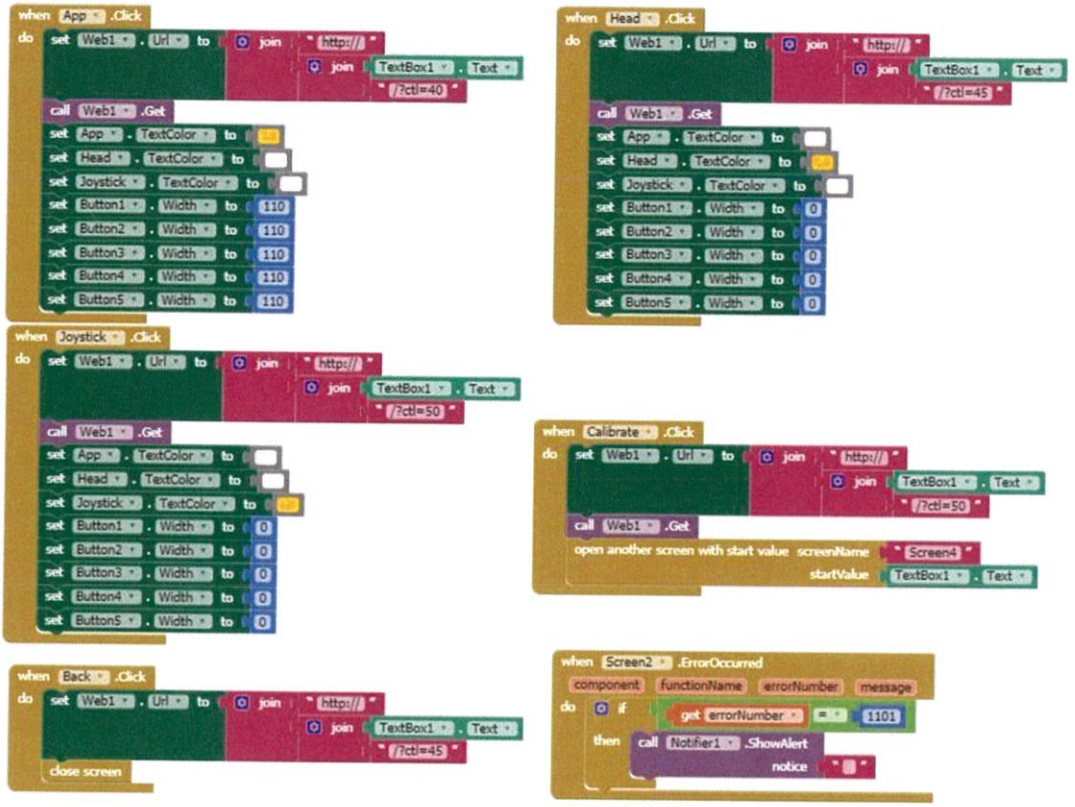
รูปที่ 3.58 Block การทำงานของ Screen2 ในส่วนของการควบคุมเก้าอี้เข็น

ที่ด้านล่างของ Screen2 จะประกอบด้วยปุ่มเลือกโหมดการควบคุม 4 ปุ่มได้แก่ App, Joystick, Head และ Calibrate ซึ่งจะส่งค่าตัวแปรต่างๆไปยัง Web Server ของ NodeMCU เพื่อทำการปรับเปลี่ยนโหมดการใช้งานของเก้าอี้เข็นดังตารางที่ 3.12 เมื่อทำการกดปุ่มAppจะทำการส่งตัวแปร ctl=40 ไปยัง Server เพื่อให้เก้าอี้เข็นอยู่ในโหมดการควบคุมผ่านแอปพลิเคชัน และจะกำหนดให้ปุ่มควบคุมทิศทางทั้ง 5 ปุ่มแสดงในหน้าจอ และกำหนดให้ text colour ของปุ่ม App แสดงเป็นสีเหลืองเพื่อบอกผู้ใช้งานแอปพลิเคชันว่าอยู่ในโหมดการใช้งานนี้ เมื่อทำการกดปุ่ม Head จะทำการส่งตัวแปร ctl=45 ไปยัง Server เพื่อให้เก้าอี้เข็นอยู่ในโหมดการควบคุมผ่านเซนเซอร์ความเร่งเชิงเส้น ADXL345 และจะกำหนดให้ปุ่มควบคุมทิศทางทั้ง 5 ปุ่มหายไปจากหน้าจอ และกำหนดให้ text colour ของปุ่ม Head แสดงเป็นสีเหลืองเพื่อบอกผู้ใช้งานแอปพลิเคชันว่าอยู่ในโหมดการใช้งาน Head เมื่อทำการกดปุ่ม Joystick จะทำการส่งตัวแปร ctl=50 ไปยัง Server เพื่อให้เก้าอี้เข็นอยู่ในโหมดการควบคุมผ่านก้านควบคุมแบบอนาล็อก และจะกำหนดให้ปุ่มควบคุมทิศทางทั้ง 5 ปุ่มหายไปจากหน้าจอ และกำหนดให้ text colour ของปุ่ม

Joystick แสดงเป็นสีเหลืองเพื่อบอกผู้ใช้งานแอปพลิเคชันว่าอยู่ในโหมด Joystick และเมื่อกดปุ่ม Calibrate จะทำการส่งตัวแปร ctl=50 ไปยัง Server เพื่อปรับเข้าสู่โหมดการใช้งานโดยใช้ก้านควบคุมอนาล็อกอีกครั้ง และจะเข้าสู่โหมดการปรับเทียบมุมองศาของศีรษะ โดยจะทำการเปิดหน้า Screen4 ซึ่งเป็นหน้าจอที่ใช้ในการ Calibrate โดยจะมีการบันทึกค่า URL ที่ใช้ในการส่งค่าไปยัง Server ไปใช้ยังหน้า Screen4ด้วย แอปพลิเคชันจะมีการป้องกัน Error ในกรณีที่ไม่สามารถเชื่อมต่อกับ URL ของ Server ได้ โดยจะปิดการแจ้งเตือนไม่ให้ Error1101 แสดงขึ้นมาในหน้าจอ ซึ่งอาจส่งผมให้แอปพลิเคชันไม่สามารถใช้งานได้ ด้านล่างของหน้าจอจะมีปุ่ม Go back โดยเมื่อทำการกดปุ่มจะทำการปิดหน้า Screen2 เพื่อกลับไปยัง Screen1 Block การทำงานดังกล่าวแสดงในรูปที่ 3.59

ตารางที่ 3.12 ค่า ctl ที่ส่งจากแอปพลิเคชันไปยัง Server ของ NodeMCU ในส่วนของการปรับเปลี่ยนโหมดการใช้งาน

Ctl Variable	Condition
40	Application Mode
45	Head Mode
50	Joystick Mode, Calibrate

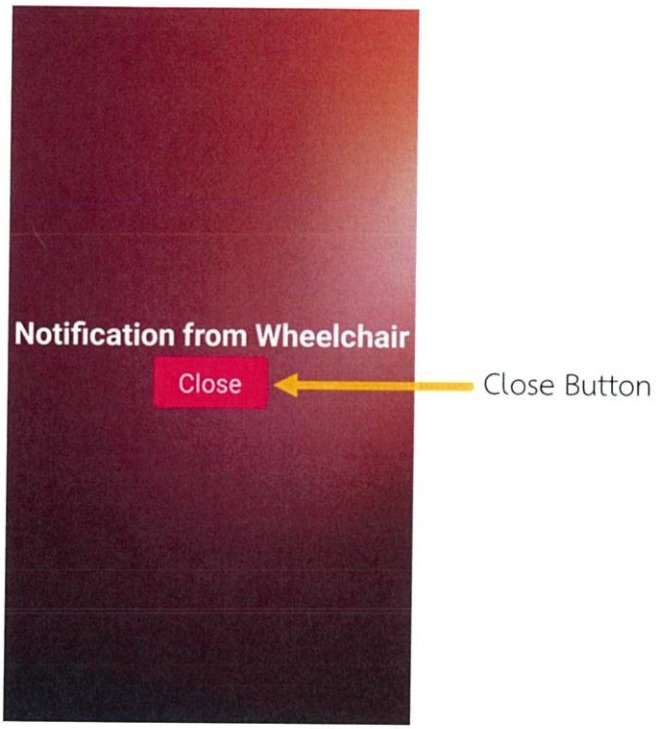


รูปที่ 3.59 Block การทำงานของหน้า Screen2 ในส่วนของปุ่มกดเพื่อเลือกโหมดการทำงานของเก้าอี้เข็น

3.1.5.3 หน้าจอ Screen3

1) การออกแบบองค์ประกอบของหน้าScreen3

หน้า Screen3 แสดงดังรูปที่ 3.60 ซึ่งเป็นหน้าจอที่แสดงการแจ้งเตือนซึ่งจะส่งมาจากการกดปุ่มแจ้งเตือนบนเก้าอี้เข็น โดยจะแสดงเป็นแบบ Push Notification บนหน้าจอซึ่งจะแจ้งเตือนแม้ว่าผู้ใช้งานแอปพลิเคชันจะปิดหน้าจอของสมาร์ทโฟนอยู่ดังแสดงในรูปที่ 3.61 ประกอบไปด้วยเสียงแจ้งเตือนและเปิดระบบสั่นขออุปกรณ์สมาร์ทโฟน เมื่อผู้ใช้กดปุ่ม close ก็จะกลับไปยังหน้าการใช้งานก่อนหน้าพร้อมทั้งหยุดเสียงแจ้งเตือนรวมไปถึงหยุดการสั่นของอุปกรณ์



รูปที่ 3.60 หน้าจอ Screen3

รูปที่ 3.61 การแสดงผลแจ้งเตือนแบบ Push Notification

แผนผังการทำงานของหน้า Screen3 แสดงดังรูปที่ 3.62 เมื่อเริ่มเปิดหน้า Screen3 จะเริ่มทำการแสดงเสียงแจ้งเตือนรวมไปถึงเปิดระบบสั่นของอุปกรณ์สมาร์ตโฟนของผู้ใช้งาน จากนั้นจะแสดงการแจ้งเตือนแบบ Push Notification แอปพลิเคชันจะไม่หยุดแจ้งเตือนจนกว่าจะผู้ใช้งานจะกดปุ่ม close ที่แสดงอยู่กลางหน้าจอ และจะส่งค่า 0 ไปยัง tag ที่ชื่อว่า noti ใน Firebase เพื่อเซ็ตค่าเป็นค่าเริ่มต้น จากนั้นจะปิดหน้า Screen3 เพื่อกลับไปยังหน้าการใช้งานก่อนหน้า

รูปที่ 3.62 แผนผังการทำงานของหน้า Screen3

2) การเขียนโปรแกรมการทำงานของหน้าจอScreen3

หน้า Screen3 มี Block การทำงานแสดงดังรูปที่ 3.63 กำหนดให้เมื่อเริ่มเปิดหน้าจอ Screen3 จะแสดง Push Notification เป็นคำว่า “Notification from WheelChair” และกำหนดให้มีเสียงแจ้งเตือนและกำหนดให้เปิดการสั่นของอุปกรณ์แจ้งเตือนเป็นระยะเวลา 1 วินาทีและจะแสดงจนกว่าจะผู้ใช้จะกดปุ่มปิดหน้าจอแจ้งเตือนและเมื่อกดปิดหน้าจอจะกำหนดให้ค่าของ FirebaseDB ใน tag ที่ชื่อว่า noti เท่ากับ 0 เพื่อรีเซ็ตค่าใหม่

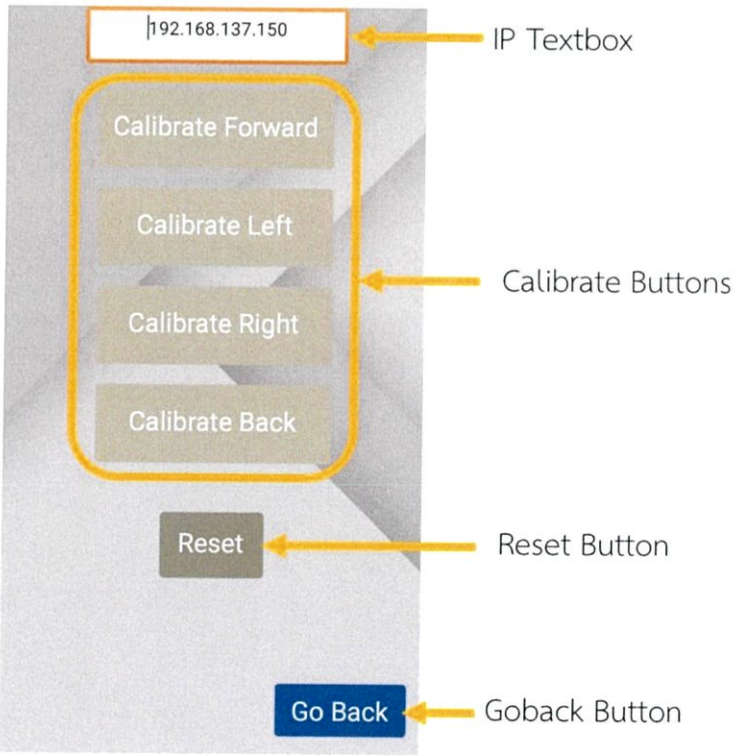
รูปที่ 3.63 Block การทำงานของ Screen3

3.1.5.4 หน้าจอScreen4

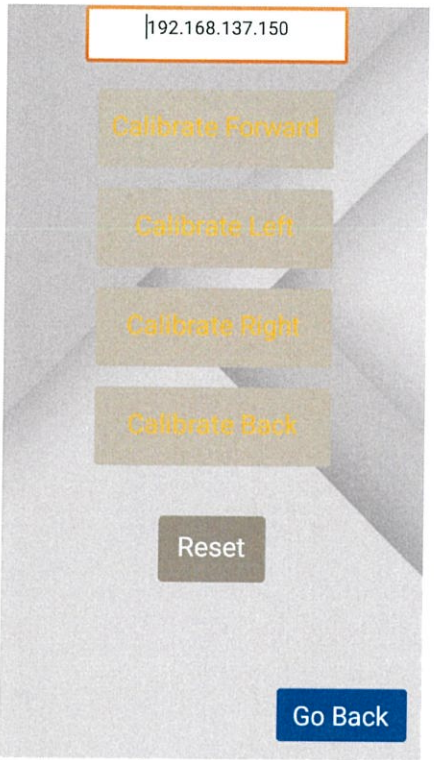
1) การออกแบบองค์ประกอบของหน้าScreen4

หน้าจอ Scren4 แสดงดังรูปที่ 3.64 เป็นหน้าจอที่จะแสดงหลังจากผู้ใช้งานแอปพลิเคชันกดปุ่ม Calibrate ที่อยู่ในหน้า Screen2 หน้า Screen4 ถูกออกแบบมาเพื่อให้ผู้ใช้งานเก้าอี้เข็นที่ไม่สามารตั้งศีรษะให้อยู่ในมุมองศาที่เหมือนกับบุคลทั่วไปสามารถ

ปรับเทียบมุมองศาการเอียงของศีรษะในการควบคุมทิศทางเก้าอี้เข็นให้แตกต่างออกไปในแต่ละบุคคลได้ หน้า Screen4 จะประกอบไปด้วยปุ่ม Calibrate จำนวน 4 ปุ่ม ได้แก่ Calibrate Forward, Calibrate Left, Calibrate Right และ Calibrate Back ผู้ใช้จะต้องทำการเอียงศีรษะไปยังมุมองศาที่ต้องการใช้งานและกดปุ่ม Calibrate เพื่อทำการบันทึกค่ามุมนั้นๆ โดยเมื่อกดปุ่ม Calibrate แล้วตัวอักษรบนปุ่มจะเปลี่ยนสถานะเป็นสีเหลืองเพื่อบอกผู้ใช้งานว่าได้ทำการ Calibrate ค่านั้นไปแล้ว ดังแสดงในรูปที่ 3.65 และเมื่อกลับไปโหมดการควบคุมด้วยศีรษะแล้วก็จะสามารถใช้งานได้ตามค่าที่ผู้ใช้งานได้ทำการ Calibrate ไว้ ด้านล่างของปุ่ม Calibrate จะมีปุ่ม Reset เพื่อลบค่า Calibrate ที่ได้บันทึกไว้แล้วและจะทำการเซ็ตสีของปุ่ม Calibrate ทั้ง 4 ปุ่มให้กลับไปเป็นสีขาวเพื่อบอกสถานะผู้ใช้ว่าเก้าอี้เข็นนี้ไม่ได้ทำการ Calibrate มุมองศาการใช้งานใดๆ และค่ามุมองศากลับไปเป็นค่าเริ่มต้นที่โปรแกรมได้ทำการกำหนดไว้ ด้านล่างของจอจะเป็นปุ่ม Go back เพื่อกลับไปยัง Screen2 ซึ่งเป็นหน้าการใช้งานหลักในการควบคุมทิศทางเก้าอี้เข็น

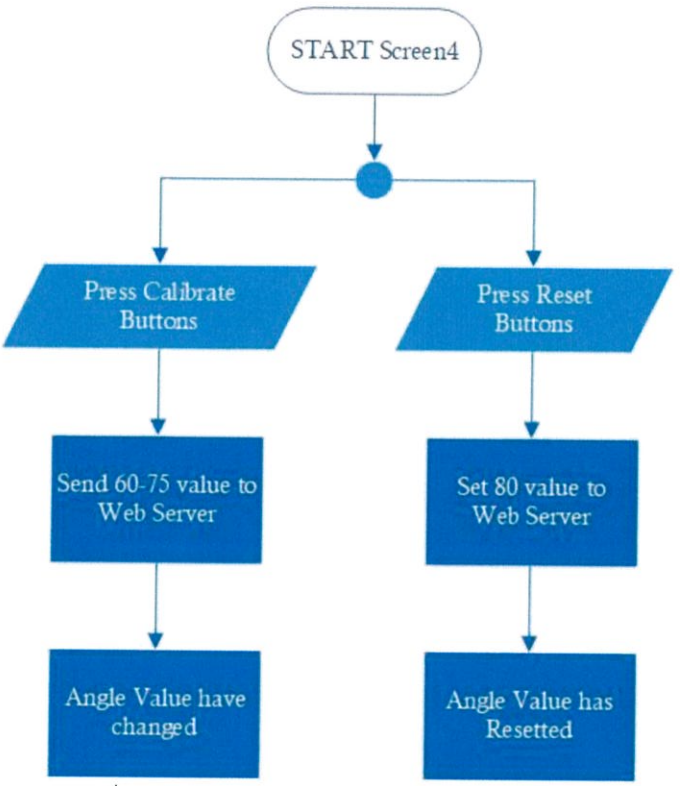


รูปที่ 3.64 หน้าจอ Screen4



รูปที่ 3.65 หน้าจอ Screen4 เมื่อทำการ Calibrate มุมองศาการใช้งานแล้ว

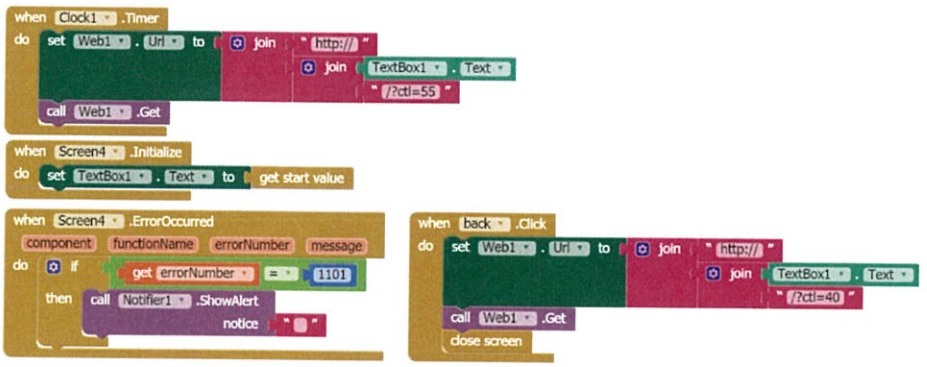
แผนผังการทำงานของหน้า Screen4 จะเป็นแสดงดังรูปที่ 3.66 เมื่อเริ่มเปิดหน้าจอ Screen4 ผู้ใช้งานจะต้องทำการกดปุ่ม Calibrate ทั้ง 4 ปุ่มเพื่อเป็นการปรับเทียบมุมองศาการใช้งาน แต่ละปุ่มที่ผู้ใช้งานได้ทำการกดแอปพลิเคชันจะทำการส่งตัวแปร ctl=60-75 ไปยัง Server ของ NodeMCU และ NodeMCU จะนำตัวแปรที่ได้ไปประมวลผลเพื่อบันทึกค่ามุมที่ผู้ใช้ได้ทำการปรับเทียบ หากผู้ใช้ต้องการทำการลบค่าที่บันทึกไว้ทั้งหมด ผู้ใช้จะต้องทำการกดปุ่มรีเซ็ตซึ่งจะตั้งอยู่ด้านล่างของแอปพลิเคชัน แอปพลิเคชันจะส่งตัวแปร ctl=80 ไปยัง Server ของ NodeMCU และ NodeMCU จะนำตัวแปรที่ได้ไปประมวลผลเพื่อลบค่ามุมที่ผู้ใช้ได้ทำการปรับเทียบไว้ก่อนหน้านี้ทั้งหมด



รูปที่ 3.66 แผนผังการทำงานของหน้า Screen4

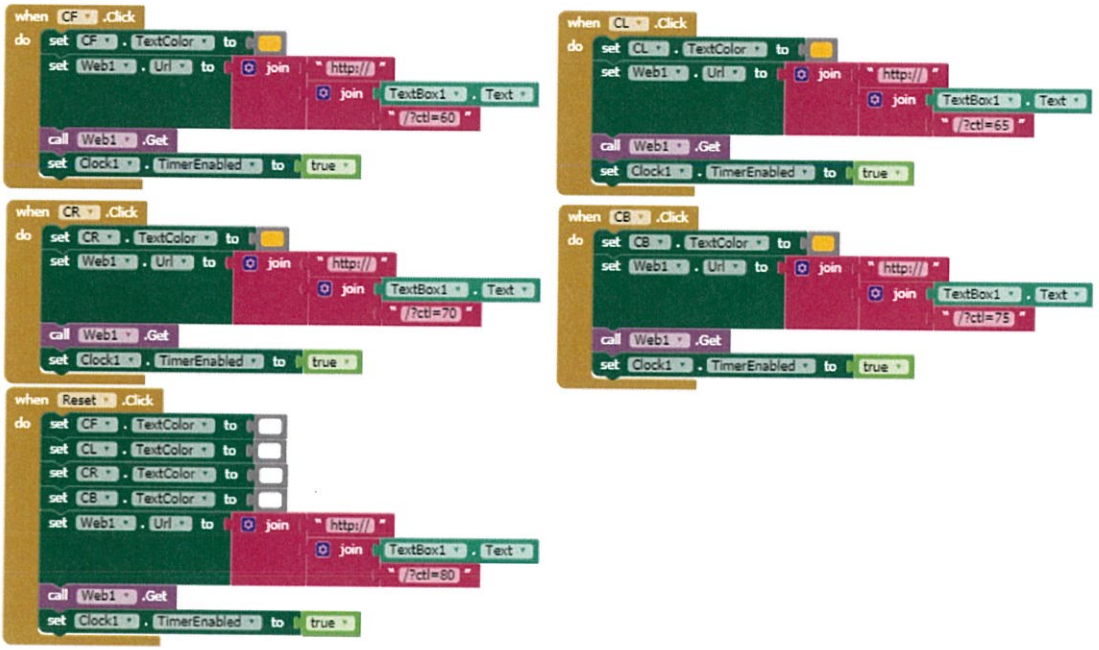
2) การเขียนโปรแกรมการทำงานของหน้าจอ Screen4

Block การทำงานของหน้า Screen4 จะแสดงดังรูปที่ 3.67 และรูปที่ 3.68 เมื่อเปิดหน้า Screen4 ขึ้นจะทำการรับค่าจาก Screen2 เป็นค่า URL ที่ใช้ในการส่งค่า ctl ไปยัง Server ของ NodeMCU ที่หน้า Screen4 จะมีการกำหนด delay clock เป็นเวลา 2 วินาที นอกจากนี้จะมีการป้องกันการเกิด Error ในกรณีที่แอปพลิเคชันไม่สามารถเชื่อมต่อกับ Server ได้ ใน Block คำสั่ง “When Screen4 .ErrorOccured” เมื่อเกิด Error=1101 จะทำการ Block การแจ้งเตือนเพื่อป้องกันการเกิดปัญหาของแอปพลิเคชัน จากนั้นทำการกำหนดค่าปุ่ม Goback ให้ส่งค่า ctl=40 ซึ่งเป็นการกำหนดเงื่อนไขในกรณีที่ผู้ใช้จะกลับไปหน้า Screen2 ซึ่งโปรแกรมจะเริ่มต้นหน้า Screen2 ด้วยโหมดการใช้งานแบบแอปพลิเคชันเสมอ



รูปที่ 3.67 Block การทำงานของ Screen4 ในส่วนของการกำหนดเงื่อนไขเริ่มต้นของหน้าจอ

เมื่อทำการกดปุ่ม Calibrate ปุ่มใดปุ่มหนึ่ง แอปพลิเคชันจะแสดง text colour เป็นสีเหลืองเพื่อแสดงให้ผู้ใช้ทราบว่าได้ทำการ Calibrate ค่านั้นๆไปแล้ว และจะส่งค่าตัวแปร ctl เป็นค่าอยู่ระหว่าง 60-80 เป็นระยะเวลา 1 วินาทีไปยัง Server ของ NodeMCU เพื่อนำไปประมวลผลเพื่อใช้ในการปรับเทียบมุมองศาการใช้งานของเก้าอี้เข็นต่อไปดังแสดงในตารางที่ 3.13 เมื่อครบ1วินาทีตัวแปร ctl จะกลับไปเป็นค่า55ซึ่งเป็นค่า ctl พื้นฐานของการปรับเทียบมุมองศาการใช้งานเก้าอี้เข็น เมื่อผู้ใช้ทำการกดปุ่ม Reset จะทำการส่งค่า 80 ไปยัง Server และจะเซ็ตค่ามุมทั้งหมดเป็นค่าเริ่มต้นของโปรแกมที่ถูกกำหนดไว้ในเก้าอี้เข็น รวมถึงเซ็ต text colour เป็นสีขาวทั้งหมดเหมือนกับตอนเริ่มต้นใช้งานหน้า Screen4 Block การทำงานของปุ่ม Calibrate ทั้ง 4 ปุ่มจะแสดงดังรูปที่ 3.68



รูปที่ 3.68 Block การทำงานของ Screen4 ในส่วนของการกำหนดเงื่อนไขของปุ่ม Calibrate

ตารางที่ 3.13 ค่า ctl ที่ใช้ในการกำหนดเงื่อนไขการ Calibrate

Ctl Variable	Condition
60	Calibrate Forward
65	Calibrate Left
70	Calibrate Right
75	Calibrate Back
80	Reset

3.2 เครื่องมือที่ใช้ในการทดลอง

3.2.1 ส่วนควบคุมทิศทางบนศีรษะ

อุปกรณ์	จำนวน
1. บอร์ดไมโครคอนโทรลเลอร์ Arduino Nano	1 ตัว
2. เซนเซอร์ความเร่งเชิงเส้น ADXL345	1 ตัว
3. โมดูลสื่อสารไร้สายบลูทูธ HC-05	1 ตัว
4. สายรัดศีรษะ	1 เส้น
5. ถ่าน 9 โวลต์	1 ก้อน
6. ตัวต้านทานขนาด 1 กิโลโอห์ม	1 ตัว
7. ตัวต้านทานขนาด 2 กิโลโอห์ม	1 ตัว
8. ระดับน้ำ	1 ตัว
9. โปรแทรกเตอร์	1 ตัว
10. สายไฟ	6 เส้น

3.2.2 ส่วนควบคุมกลไกขับเคลื่อนเก้าอี้เข็น

อุปกรณ์	จำนวน
1. บอร์ดไมโครคอนโทรลเลอร์ Arduino MEGA	1 ตัว
2. NodeMCU V2.0	1 ตัว
3. วงจร H-Bridge รุ่น SE-HB40-1	2 ตัว
4. มอเตอร์ไฟฟ้ากระแสตรง 24 V 350 W	2 หม้อ
5. เก้าอี้เข็นขนาดเส้นผ่าศูนย์กลางล้อ 22 นิ้ว	1 คัน
6. แบตเตอรี่ 12 โวลต์	1 ใบ
8. เซนเซอร์ความเร่งเชิงเส้น ADXL345	1 ตัว
9. ก้านควบคุม	1 ตัว
10. โมดูลสื่อสารไร้สายบลูทูธ HC-05	1 ตัว
11. โมดูลเซนเซอร์อัลตราโซนิค HC-SR04	1 ตัว
12. โมดูลเสียง (Active Buzzer Module)	1 ตัว
13. IC7805	1 ตัว

14. ตัวเก็บประจุขนาด 1 uF 1 ตัว
15. ตัวเก็บประจุขนาด 10 uF 1 ตัว
16. ปุ่มกด 1 ตัว
17. สวิตช์เปิด/ปิด 1 ตัว

3.3 การจัดเก็บผลการทดลอง

3.3.1 การทดสอบกลไกการขับเคลื่อนเก้าอี้เข็น

3.3.1.1 การทดสอบวงจรขับมอเตอร์ไฟฟ้ากระแสตรงในการขับเคลื่อนมอเตอร์

การทดสอบโดยให้บอร์ดไมโครคอนโทรลเลอร์ Arduino ทำการควบคุมวงจร H-Bridge เพื่อขับเคลื่อนมอเตอร์ โดยกำหนดความเร็วของมอเตอร์ด้วยความกว้างของลูกคลื่นสี่เหลี่ยม (Duty Cycle) และกำหนดทิศทางการหมุนของมอเตอร์ของทั้ง 2 ล้อจากการป้อนแรงดันไฟฟ้าไปยังวงจร (IN1 และ IN2) ตามทิศทางการเคลื่อนที่ของเก้าอี้เข็นในตารางที่ 3.1 สังเกตและบันทึกผลสัญญาณเอาต์พุตของวงจร H-Bridge ในแต่ละทิศทางบนออสซิลโลสโคป

3.3.1.2 การทดสอบวงจรแปลงแรงดันไฟฟ้า

ในการทดสอบวงจรแปลงแรงดันไฟฟ้านั้นจะป้อนแรงดันไฟฟ้าขนาด 12 โวลต์ให้กับวงจร จากนั้นสังเกตและบันทึกผลเอาต์พุตของวงจรบนออสซิลโลสโคป

3.3.1.3 การทดสอบความเร็วของเก้าอี้เข็น

การทดสอบความเร็วของเก้าอี้เข็นโดยให้เก้าอี้เข็นเคลื่อนที่โดยบรรทุกโหลดผู้ใช้งานที่มีน้ำหนักแตกต่างกัน โดยให้เคลื่อนที่เป็นระยะ 5.5 เมตร ทำการจับเวลาที่เก้าอี้เข็นเคลื่อนที่ได้ แล้วนำไปคำนวณในสมการที่ (3.2) เพื่อหาความเร็วเฉลี่ยของเก้าอี้เข็นเมื่อรองรับโหลดที่มีน้ำหนักแตกต่างกันไป

3.3.2 การใช้ก้านควบคุม (Joystick) ในการควบคุมทิศทางเก้าอี้เข็น

การทดลองใช้ก้านควบคุมเพื่อควบคุมทิศทางของเก้าอี้เข็นนั้น จะทำการบังคับก้านควบคุมไปยังทิศทางที่กำหนด ได้แก่ หยุดนิ่ง (ไม่มีการบังคับควบคุม), เคลื่อนที่ไปข้างหน้า (โน้มก้านควบคุมไปข้างหน้า), เลี้ยวซ้าย(โน้มก้านควบคุมไปทางซ้าย), เลี้ยวขวา (โน้มก้านควบคุมไปทางขวา) และถอยหลัง (โน้มก้านควบคุมไปข้างหลัง) บันทึกผลการแสดงทิศทางบน Serial Monitor จากนั้น

ทำการทดลองใช้ก้านควบคุมในการควบคุมการหยุดโดยการกดปุ่มกดของก้านควบคุม 1 ครั้ง และการเรียกผู้ดูแลโดยการกดปุ่มสวิตช์ บันทึกผลการแสดงการควบคุมบน Serial Monitor

3.3.3 การควบคุมทิศทางเก้าอี้เข็นด้วยศีรษะโดยใช้เซนเซอร์ความเร่งเชิงเส้น ADXL345

3.3.3.1 การปรับเทียบความแม่นยำของเซนเซอร์ ADXL345

หาค่าความเร่งเชิงเส้นสูงสุดกระทำต่อแกนอ้างอิงบวก (A_{+1g}) และค่าความเร่งเชิงเส้นต่ำสุดกระทำต่อแกนอ้างอิงลบ (A_{-1g}) โดยสังเกตและบันทึกผลจาก Serial Monitor จากนั้นคำนวณหาค่า Offset ($A_{OFF}[g]$) และค่า Gain ($Gain$) ของค่าความเร่งแต่ละแกน เพื่อนำไปหาค่าความเร่งเชิงเส้นที่ผ่านการชดเชยความคลาดเคลื่อนแล้ว ($A_{Calibrate[g]}$)

3.3.3.2 การทดสอบการตรวจวัดมุมเอียงของเซนเซอร์ ADXL345

ในส่วนของการทดสอบการตรวจวัดมุมเอียงของเซนเซอร์ความเร่ง ADXL345 นั้นจะทดสอบโดยอ้างอิงกับลักษณะการเอียงศีรษะที่สอดคล้องกับทิศทางการเคลื่อนที่ของเก้าอี้เข็น โดยจะแบ่งการทดสอบออกเป็น 2 ส่วน ได้แก่

1) การทดสอบการตรวจวัดมุมเอียงของเซนเซอร์ในระนาบ X-Z

ในระนาบ X-Z จะถูกกำหนดให้สอดคล้องกับการเอียงศีรษะไปทางด้านซ้ายและขวา ซึ่งแทนการควบคุมเก้าอี้เข็นให้ไปในทิศทางเป็นเลี้ยวซ้ายและขวาตามลำดับ ในระนาบนี้จะมีค่ามุมที่เกิดขึ้นอยู่ด้วยกัน 2 มุม คือ มุม θ ที่กระทำต่อแกน X และมุม ϕ ที่กระทำต่อแกน Z โดยวัดที่มุม -90 องศา, -60 องศา, -30 องศา, 0 องศา, 30 องศา, 60 องศา และ 90 องศา ทำการบันทึกค่ามุม θ, มุม ψ และมุม ϕ ที่สังเกตได้จากบน Serial Monitor แล้วจึงคำนวณหาความคลาดเคลื่อนของการวัดมุม

2) การทดสอบการตรวจวัดมุมเอียงของเซนเซอร์ในระนาบ Y-Z

ในระนาบ Y-Z จะถูกกำหนดให้สอดคล้องกับการก้มศีรษะไปข้างหน้าและเงยศีรษะขึ้น ซึ่งแทนการควบคุมเก้าอี้เข็นให้ไปในทิศทางเป็นเคลื่อนที่ไปข้างหน้าและถอยหลังตามลำดับ ในระนาบนี้จะมีค่ามุมที่เกิดขึ้นอยู่ด้วยกัน 2 มุม คือ มุม ψ ที่กระทำต่อแกน Y และมุม ϕ ที่กระทำต่อแกน Z โดยจะทำการตรวจวัดมุมเอียงในระนาบ Y-Z ที่มุม -90 องศา, -60 องศา, -30 องศา, 0 องศา, 30 องศา, 60 องศา และ 90 องศา ทำการบันทึกค่ามุม θ, มุม ψ และมุม ϕ ที่สังเกตได้จากบน Serial Monitor แล้วจึงคำนวณหาความคลาดเคลื่อนของการวัดมุม

ในการคำนวณความคลาดเคลื่อนของมุมเอียงได้จากสมการที่ (3.14) โดยค่ามุมที่วัดได้จะถูกเทียบความต่างของมุมกับค่ามุมจริงที่กำหนดในการตรวจวัด

$$\textit{ความคลาดเคลื่อนมุม} = \pm |\textit{ค่ามุมจริง} - \textit{ค่ามุมที่วัดได้}| \quad (3.14)$$

3.3.3.3 การควบคุมทิศทางเก้าอี้เข็นด้วยศีรษะ

1) การทดสอบการใช้ศีรษะควบคุมทิศทางในการเคลื่อนที่นั้น จะให้ผู้ทดลองสวมใส่อุปกรณ์บนศีรษะ จากนั้นให้ผู้ทดลองทำการเอียงศีรษะไปในทิศทางทั้ง 5 ทิศทาง ได้แก่ หยุดนิ่ง (ศีรษะตั้งตรง), เคลื่อนที่ไปข้างหน้า (ก้มศีรษะลง), เลี้ยวซ้าย (เอียงศีรษะไปทางซ้าย), เลี้ยวขวา (เอียงศีรษะไปทางขวา) และถอยหลัง (เงยศีรษะขึ้น) โดยสังเกตการแสดงผลสถานะทิศทางและมุมการเอียงของศีรษะบน Serial Monitor

2) การใช้งานฟังก์ชันขัดจังหวะ (Interrupt) เป็นการเปิด/ปิดการใช้ศีรษะเพื่อควบคุมเก้าอี้เข็นผ่านการเกิด Tap โดยสังเกตการแสดงผลสถานะทิศทางบน Serial Monitor

3.3.3.4 การปรับเทียบค่าเงื่อนไขทิศทางของบุคคล

ในส่วนนี้จะทดสอบการปรับเทียบค่าเงื่อนไขมุมเอียงศีรษะเพื่อควบคุมทิศทางตามความต้องการของผู้ใช้งาน ให้ผู้ทดลองทำการเอียงศีรษะตามต้องการเพื่อปรับเทียบไปใน 4 ทิศทาง ได้แก่ เคลื่อนที่ไปข้างหน้า (ก้มศีรษะลง), เลี้ยวซ้าย (เอียงศีรษะไปทางซ้าย), เลี้ยวขวา (เอียงศีรษะไปทางขวา) และถอยหลัง (เงยศีรษะขึ้น) โดยสังเกตการแสดงผล การปรับเทียบมุมการเอียงของศีรษะบน Serial Monitor รวมไปถึงทดสอบควบคุมทิศทางของเก้าอี้เข็นโดยใช้ค่าเงื่อนไขมุมเอียงที่ปรับเทียบนี้บน Serial Monitor

3.3.3.5 การทดสอบเคลื่อนที่ขึ้นและเคลื่อนที่ลงทางลาดเอียง

ในส่วนนี้จะทดสอบใช้การเอียงศีรษะควบคุมเก้าอี้เข็นเคลื่อนที่ขึ้นทางลาดเอียงและเคลื่อนที่ลงทางลาดเอียง สังเกตและบันทึกผลบน Serial Monitor และสัญญาณเอาต์พุตของวงจร H-Bridge บนออสซิลโลสโคป

3.3.4 การทดสอบระบบแจ้งเตือนการชน

ระบบแจ้งเตือนการชนนี้จะทำงานเมื่อมีการควบคุมทิศทางการเคลื่อนที่ให้ถอยหลังเพื่อจะส่งสัญญาณเสียงแจ้งเตือนให้ผู้ใช้งานทราบ โดยทำการทดสอบการตรวจวัดระยะห่างของวัตถุและการแจ้งเตือนผู้ใช้งานเมื่อวัตถุอยู่ด้านหลังของเก้าอี้เข็นเป็นระยะน้อยกว่า 50 เซนติเมตร, ระยะตั้งแต่ 50 เซนติเมตร ถึง 100 เซนติเมตร และระยะมากกว่า 100 เซนติเมตร สังเกตผลบน Serial Monitor

3.3.5 การเชื่อมต่อสื่อสารในระบบเก้าอี้เข็นควบคุมได้

3.3.5.1 การเชื่อมต่อสื่อสารระหว่างส่วนควบคุมทิศทางบนศีรษะและส่วนควบคุมชุดกลไกขับเคลื่อนเก้าอี้เข็น

ในส่วนนี้จะเป็นการทดสอบสื่อสารกันผ่านโมดูลสื่อสารไร้สายบลูทูธ โดยจะเป็นการส่งค่าทิศทางและค่าสำหรับการให้ส่วนควบคุมทิศทางทำการปรับเทียบเงื่อนไข โดยสังเกตผลบน Serial Monitor ทั้งในส่วนควบคุมทิศทางและส่วนควบคุมชุดกลไกขับเคลื่อน

3.3.5.2 การใช้งานแอปพลิเคชันและการสื่อสารผ่าน Web Server บน NodeMCU

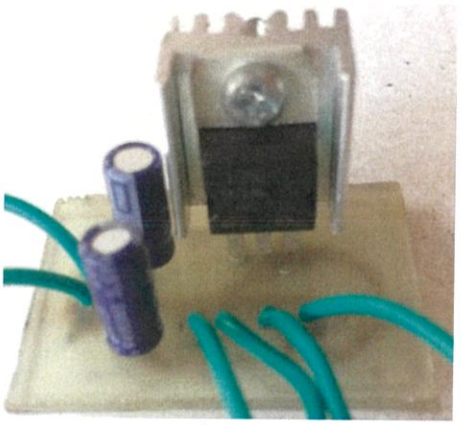
ในส่วนนี้จะทำการทดสอบการใช้งานแอปพลิเคชัน โดยจะประกอบไปด้วยการลงชื่อเข้าใช้งานแอปพลิเคชัน, การควบคุมทิศทางการเคลื่อนที่เก้าอี้เข็นผ่านแอปพลิเคชัน, การสั่งการผ่านแอปพลิเคชันให้ส่วนควบคุมทิศทางทำการปรับเทียบเงื่อนไขทิศทาง, การเปลี่ยนโหมดการควบคุมเก้าอี้เข็นผ่านแอปพลิเคชัน และ การเรียกผู้ดูแลโดยแจ้งเตือนผ่านแอปพลิเคชัน โดยสังเกตผลบนแอปพลิเคชันและ Serial Monitor ของ NodeMCU และ Arduino MEGA

บทที่ 4

ผลการทดลอง

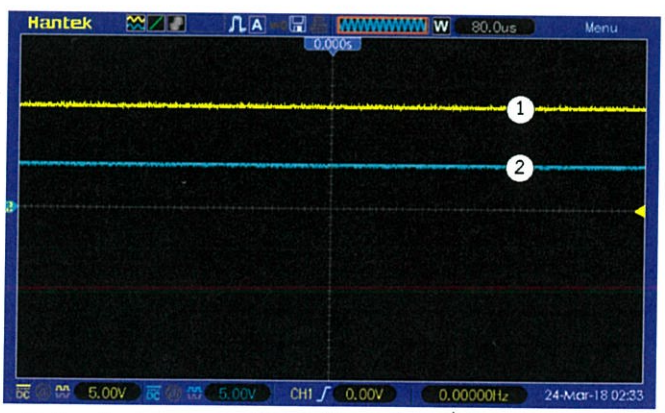
4.1 การทดสอบกลไกการขับเคลื่อนเก้าอี้เข็น

4.1.1 การทดสอบวงจรแปลงแรงดันไฟฟ้า



รูปที่ 4.1 วงจรแปลงแรงดันไฟฟ้าจาก 12 โวลต์เป็น 5 โวลต์

วงจรแปลงแรงดันไฟฟ้าในรูปที่ 4.1 ถูกใช้งานในเก้าอี้เข็นเพื่อแปลงแรงดันไฟฟ้าจากแบตเตอรี่ขนาด 12 โวลต์ ให้มีขนาด 5 โวลต์ เพื่อป้อนให้กับบอร์ดไมโครคอนโทรลเลอร์ให้ทำงาน เมื่อทำการป้อนแรงดันไฟฟ้าขนาด 12 โวลต์ให้กับวงจร จะได้ผลค่าแรงดันไฟฟ้าที่ได้จากวงจรบนออสซิลโลสโคป ดังรูปที่ 4.2 โดยแรงดันไฟฟ้าขนาด 12 โวลต์คือสัญญาณหมายเลข 1 และแรงดันเอาต์พุตของวงจรที่มีขนาด 5 โวลต์คือสัญญาณหมายเลข 2

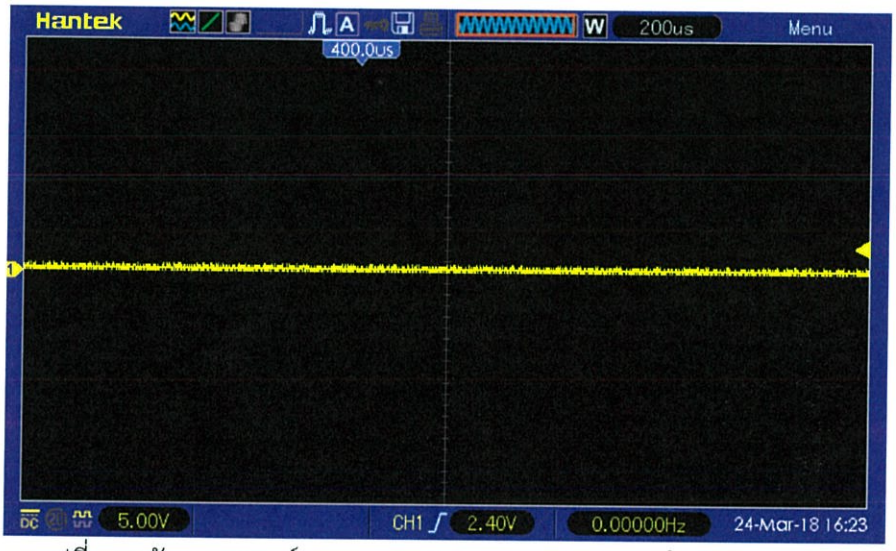


รูปที่ 4.2 ออสซิลโลสโคปแสดงแรงดันไฟฟ้าเอาต์พุตที่ได้จากวงจรแปลงแรงดันไฟฟ้า

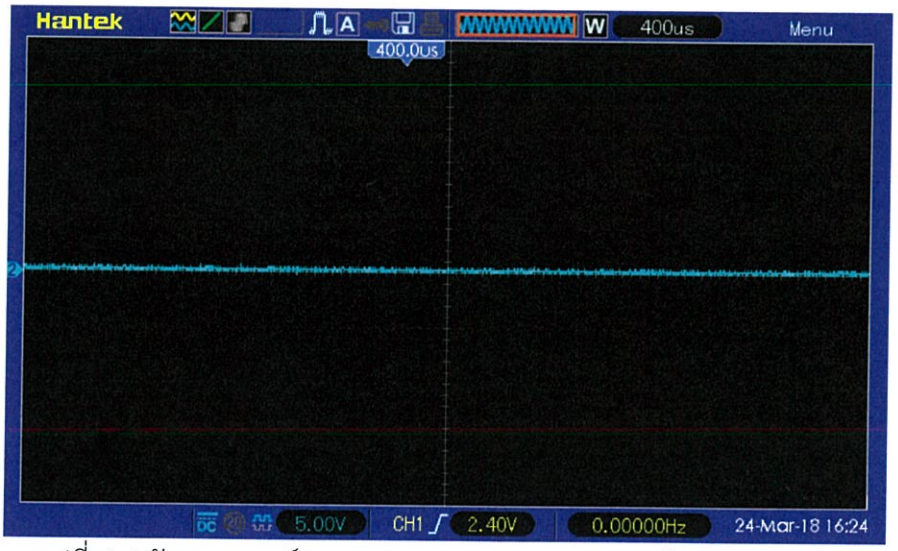
4.1.2 การทดสอบกลไกการขับเคลื่อนเก้าอี้เข็น

เมื่อให้บอร์ดไมโครคอนโทรลเลอร์ Arduino ทำการควบคุมวงจร H-Bridge เพื่อขับเคลื่อนมอเตอร์ โดยกำหนดความเร็วของมอเตอร์ด้วยความกว้างของลูกคลื่นสี่เหลี่ยม (Duty Cycle) และกำหนดทิศทางการหมุนของมอเตอร์ของทั้ง 2 ล้อจากการป้อนแรงดันไฟฟ้าไปยังวงจร (IN1 และ IN2) ตามทิศทางการเคลื่อนที่ของเก้าอี้เข็น จะได้ผลดังต่อไปนี้

เมื่ออยู่ในสถานะหยุดนิ่ง (STAY) วงจร H-Bridge 1 และวงจร H-Bridge 2 จะมีสัญญาณเอาต์พุต PWM ดังรูปที่ 4.3 และรูปที่ 4.4 ตามลำดับ

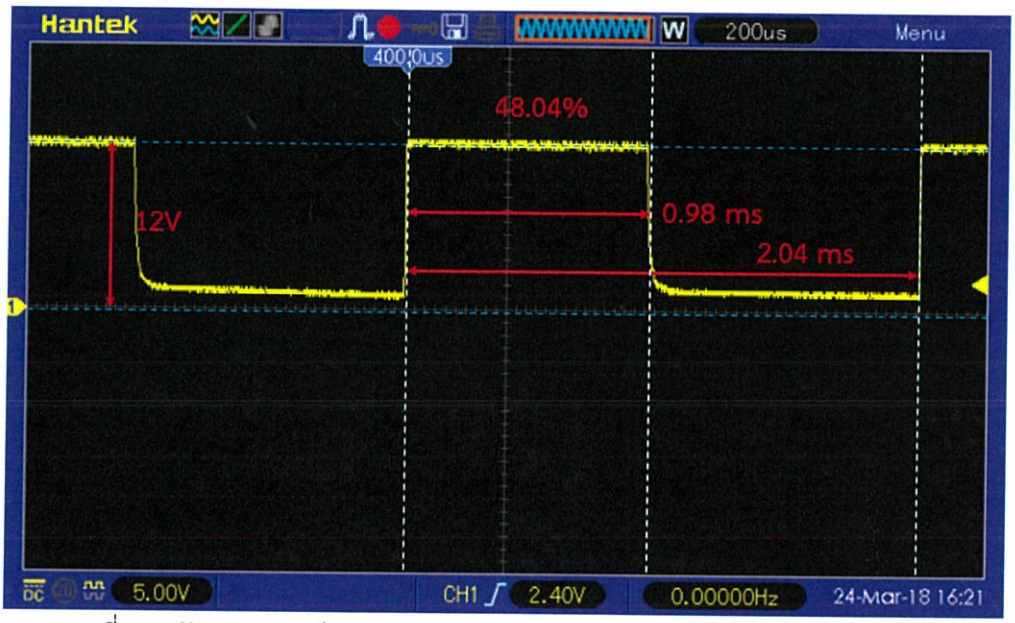


รูปที่ 4.3 สัญญาณเอาต์พุต PWM ของวงจร H-Bridge 1 ในสถานะ STAY

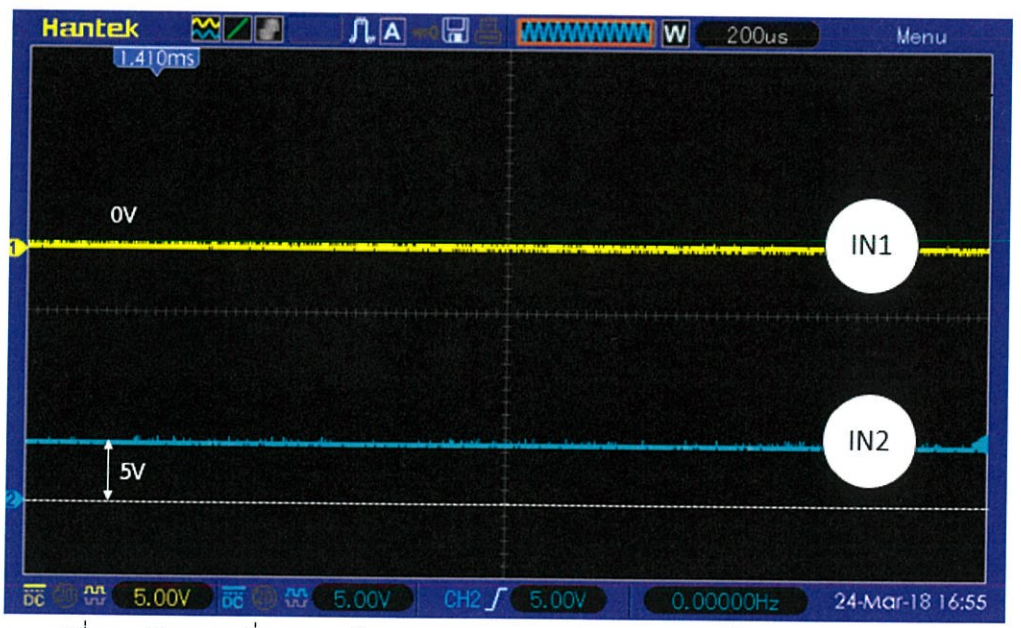


รูปที่ 4.4 สัญญาณเอาต์พุต PWM ของวงจร H-Bridge 2 ในสถานะ STAY

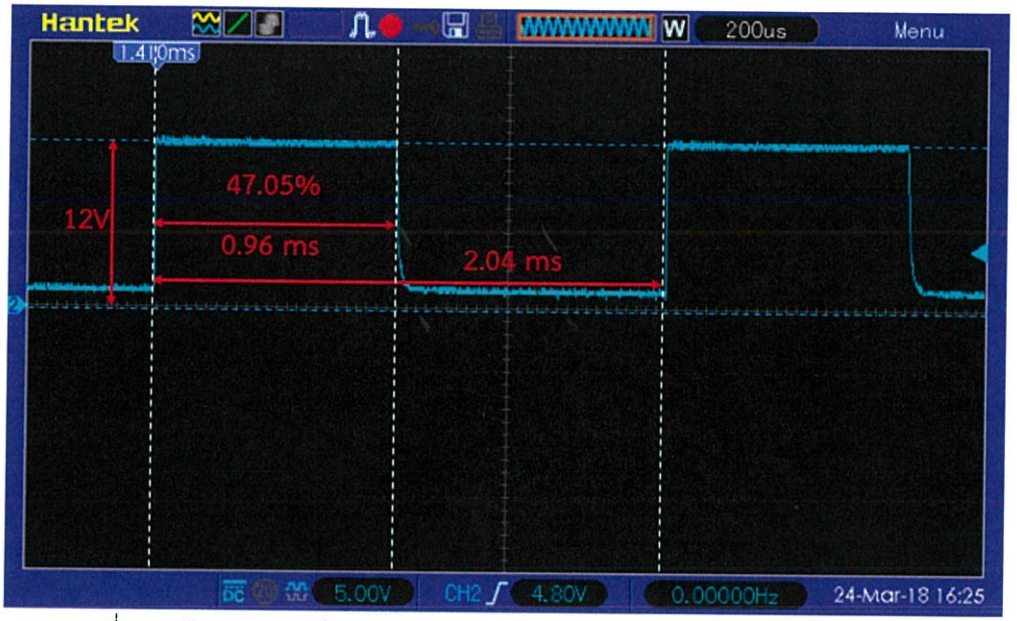
เมื่ออยู่ในสถานะเคลื่อนที่ไปข้างหน้า (FORWARD) วงจร H-Bridge 1 และวงจร H-Bridge 2 จะมีสัญญาณเอาต์พุต PWM และสัญญาณที่ขา IN1 กับ ขา IN2 ดังรูปที่ 4.5 ถึงรูปที่ 4.8 ตามลำดับ



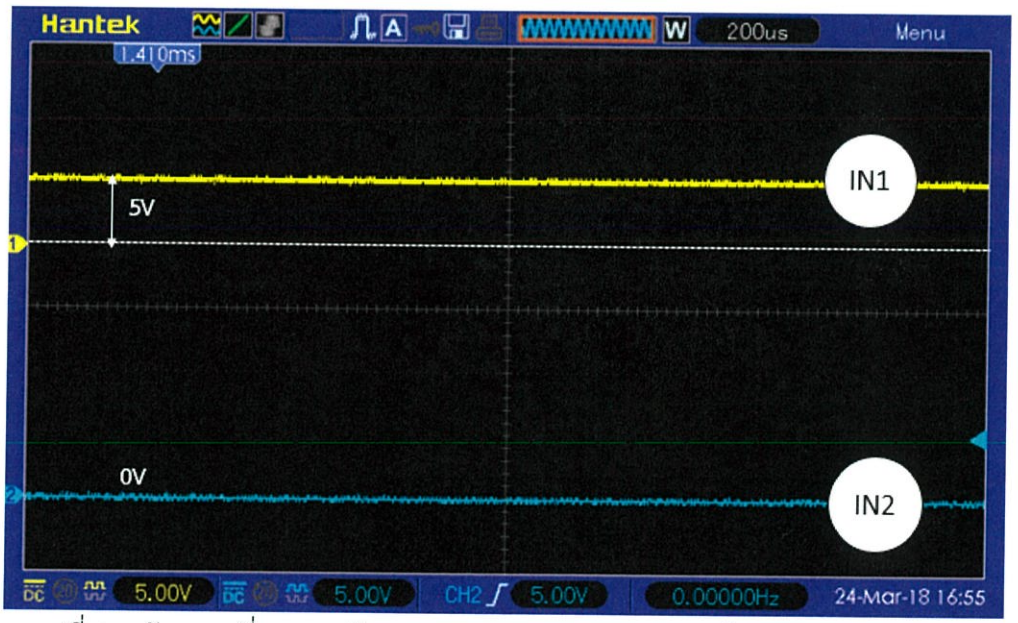
รูปที่ 4.5 สัญญาณเอาต์พุต PWM ของวงจร H-Bridge 1 ในสถานะ FORWARD



รูปที่ 4.6 สัญญาณที่ขา IN1 กับ ขา IN2 ของวงจร H-Bridge 1 ในสถานะ FORWARD

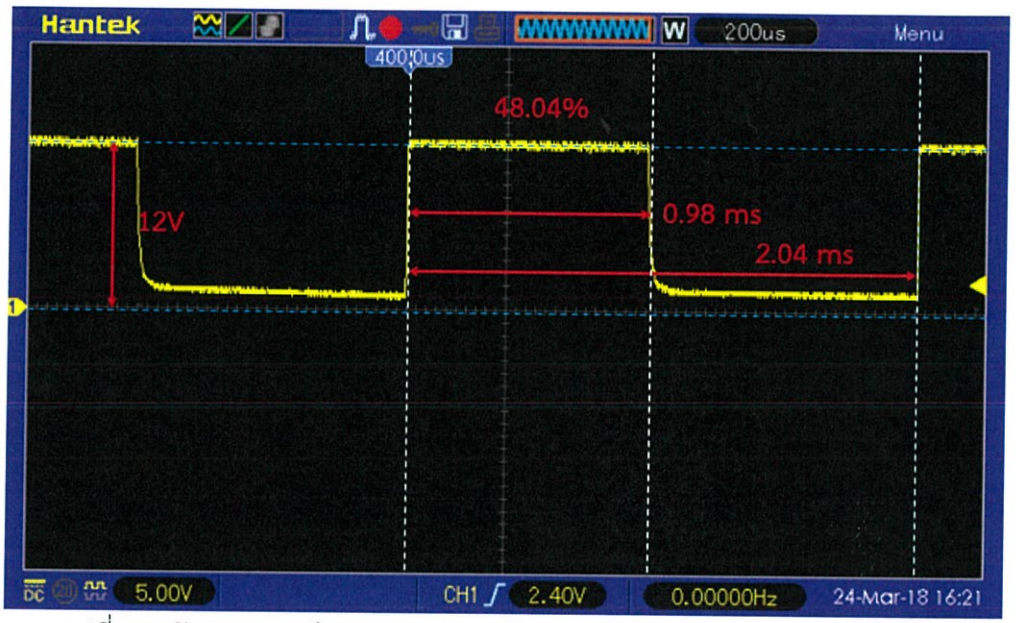


รูปที่ 4.7 สัญญาณเอาต์พุต PWM ของวงจร H-Bridge 2 ในสถานะ FORWARD

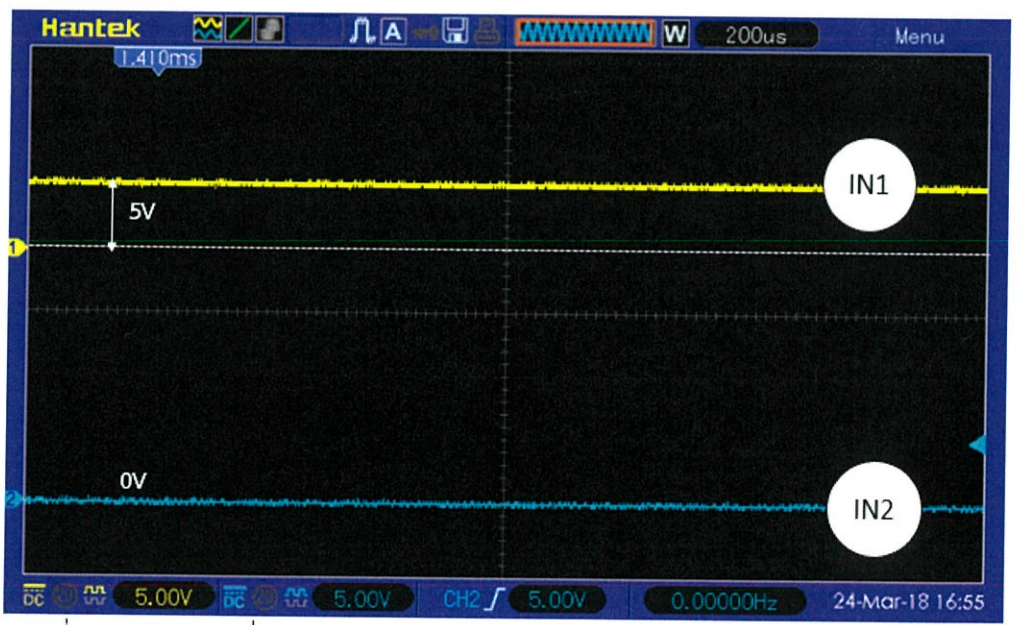


รูปที่ 4.8 สัญญาณที่ขา IN1 กับ ขา IN2 ของวงจร H-Bridge 2 ในสถานะ FORWARD

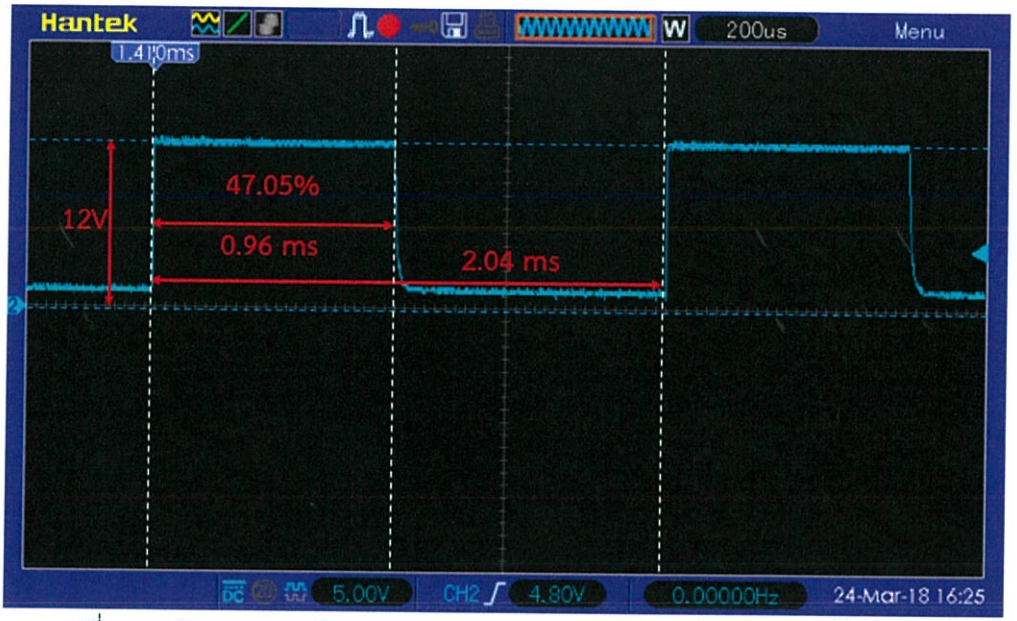
เมื่ออยู่ในสถานะเคลื่อนที่ถอยหลัง (BACKWARD) วงจร H-Bridge 1 และวงจร H-Bridge 2 จะมีสัญญาณเอาต์พุต PWM และสัญญาณที่ขา IN1 กับ ขา IN2 ดังรูปที่ 4.9 ถึงรูปที่ 4.12 ตามลำดับ



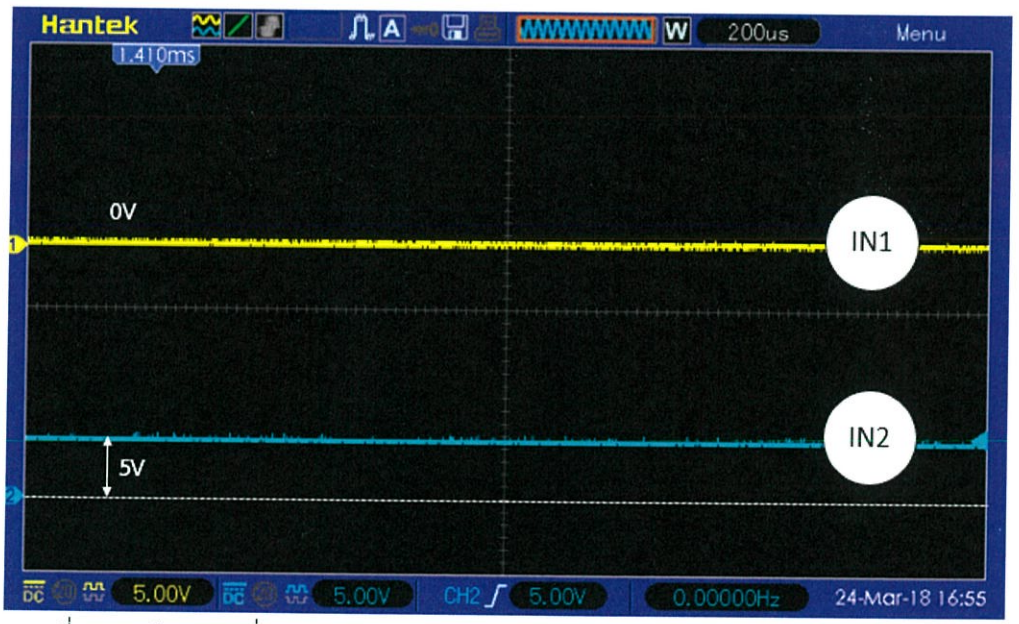
รูปที่ 4.9 สัญญาณเอาต์พุต PWM ของวงจร H-Bridge 1 ในสถานะ BACKWARD



รูปที่ 4.10 สัญญาณที่ขา IN1 กับ ขา IN2 ของวงจร H-Bridge 1 ในสถานะ BACKWARD

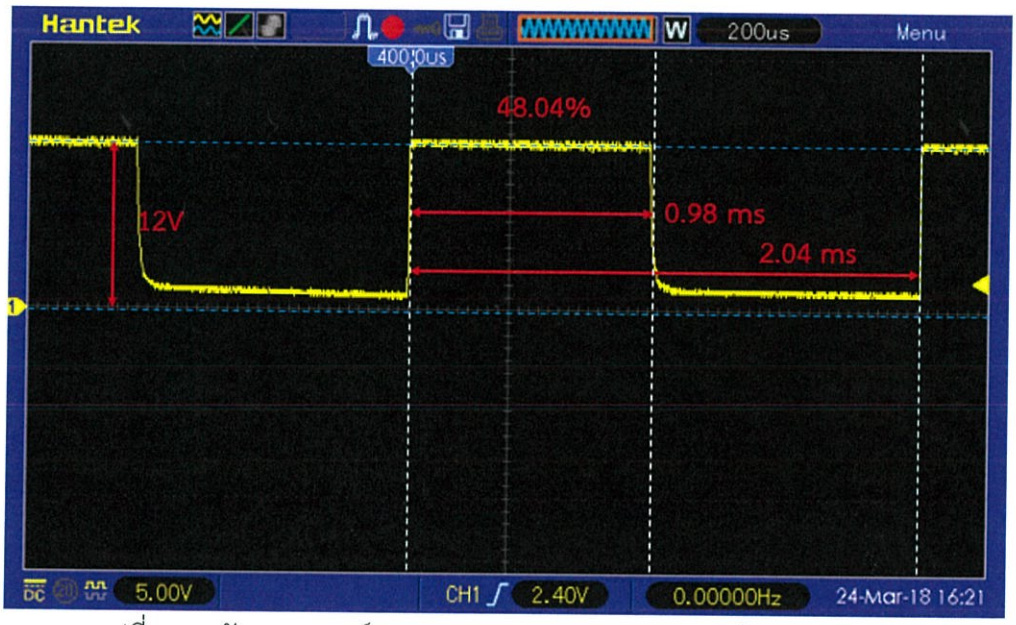


รูปที่ 4.11 สัญญาณเอาต์พุต PWM ของวงจร H-Bridge 2 ในสถานะ BACKWARD

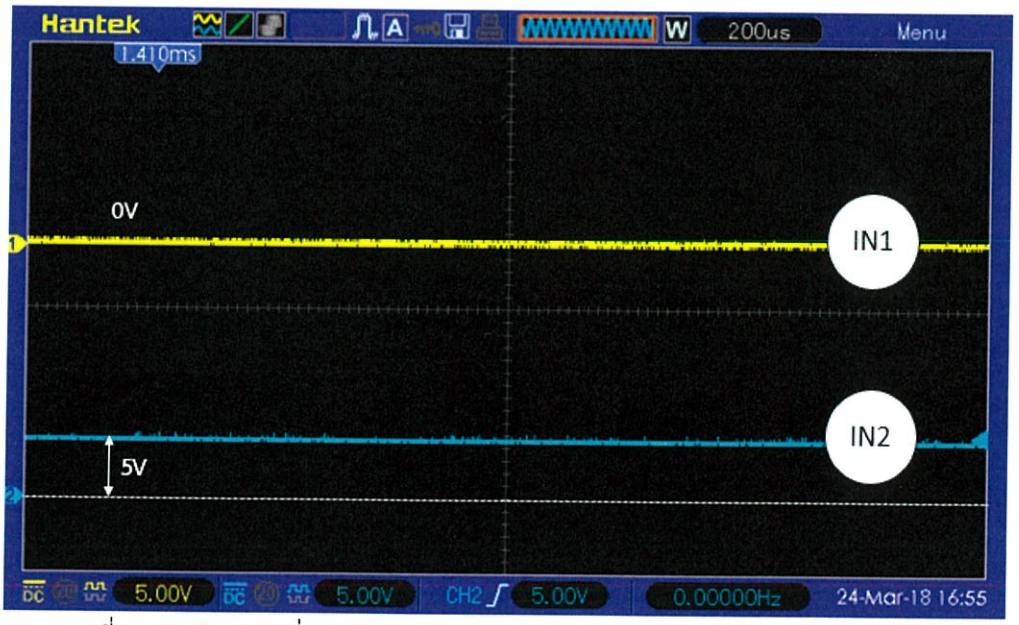


รูปที่ 4.12 สัญญาณที่ขา IN1 กับ ขา IN2 ของวงจร H-Bridge 2 ในสถานะ BACKWARD

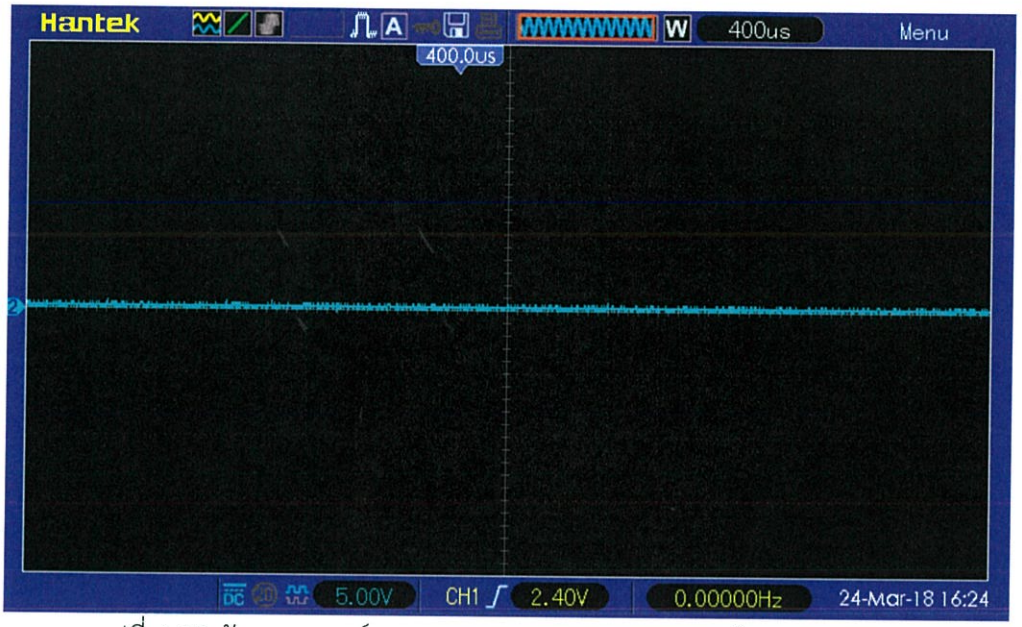
เมื่ออยู่ในสถานะสถานะเลี้ยวซ้าย (LEFT) วงจร H-Bridge 1 และวงจร H-Bridge 2 จะมีสัญญาณเอาต์พุต PWM และสัญญาณที่ขา IN1 กับ ขา IN2 ดังรูปที่ 4.13 ถึงรูปที่ 4.15 ตามลำดับ



รูปที่ 4.13 สัญญาณเอาต์พุต PWM ของวงจร H-Bridge 1 ในสถานะ LEFT

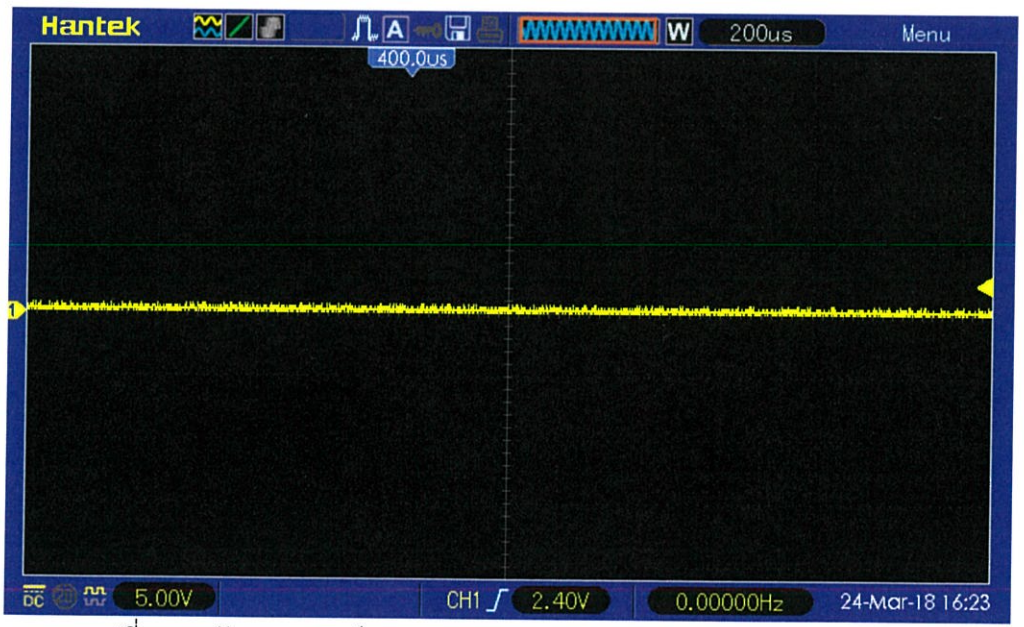


รูปที่ 4.14 สัญญาณที่ขา IN1 กับ ขา IN2 ของวงจร H-Bridge 1 ในสถานะ LEFT

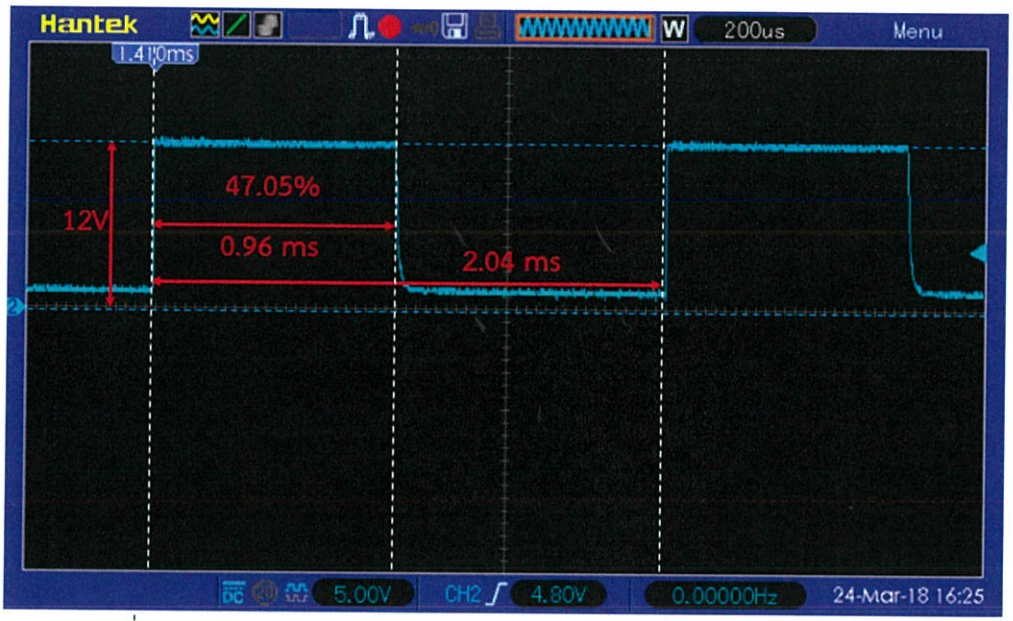


รูปที่ 4.15 สัญญาณเอาต์พุต PWM ของวงจร H-Bridge 2 ในสถานะ LEFT

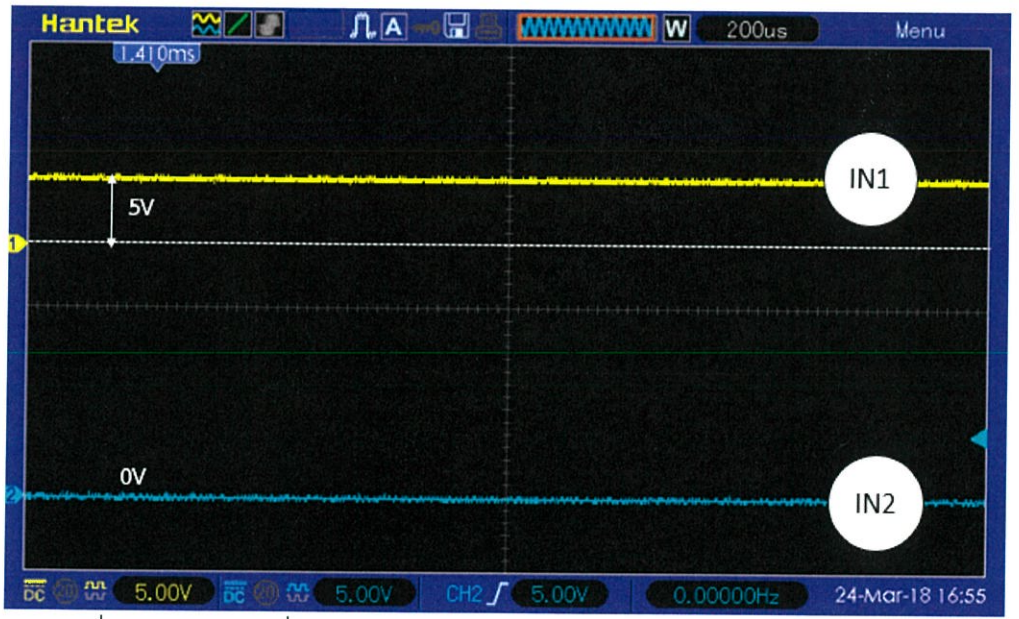
เมื่ออยู่ในสถานะสถานะเลี้ยวขวา (RIGHT) วงจร H-Bridge 1 และวงจร H-Bridge 2 จะมีสัญญาณเอาต์พุต PWM และสัญญาณที่ขา IN1 กับ ขา IN2 ดังรูปที่ 4.16 ถึงรูปที่ 4.18 ตามลำดับ



รูปที่ 4.16 สัญญาณเอาต์พุต PWM ของวงจร H-Bridge 1 ในสถานะ RIGHT

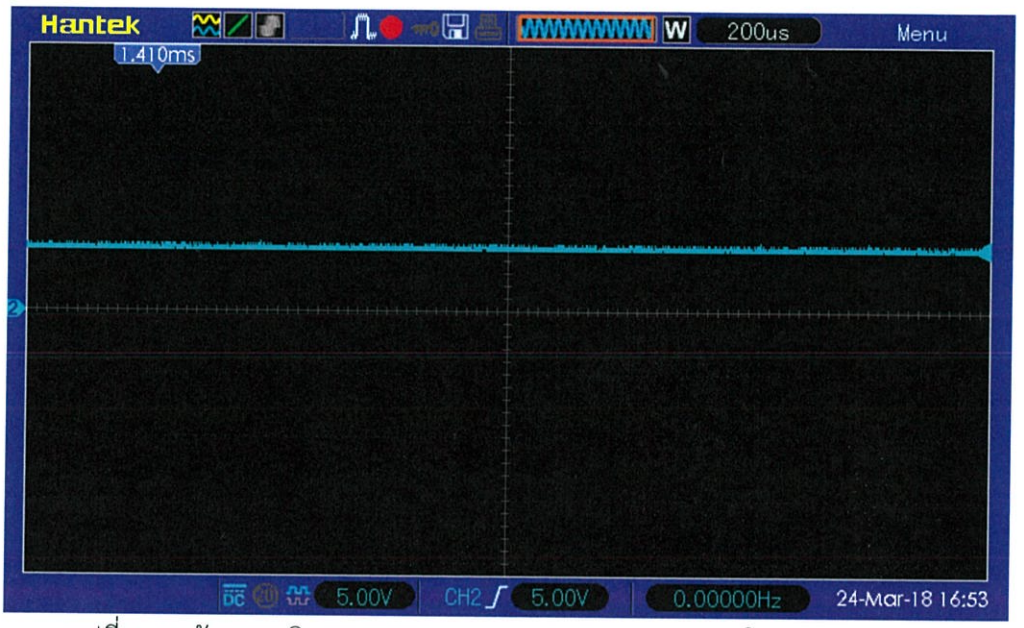


รูปที่ 4.17 สัญญาณเอาต์พุต PWM ของวงจร H-Bridge 2 ในสถานะ RIGHT

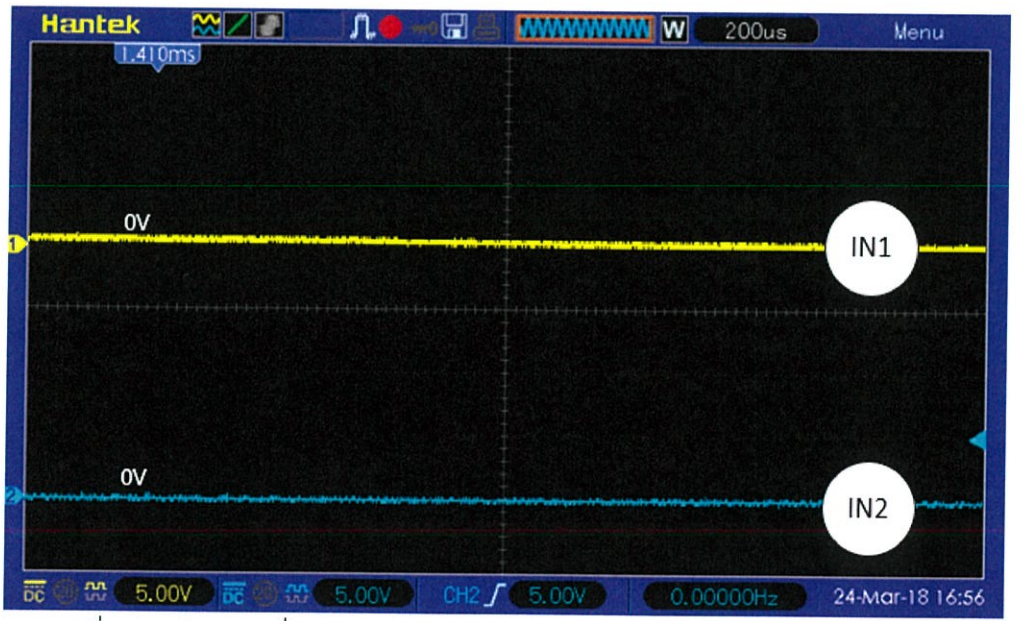


รูปที่ 4.18 สัญญาณที่ขา IN1 กับ ขา IN2 ของวงจร H-Bridge 2 ในสถานะ RIGHT

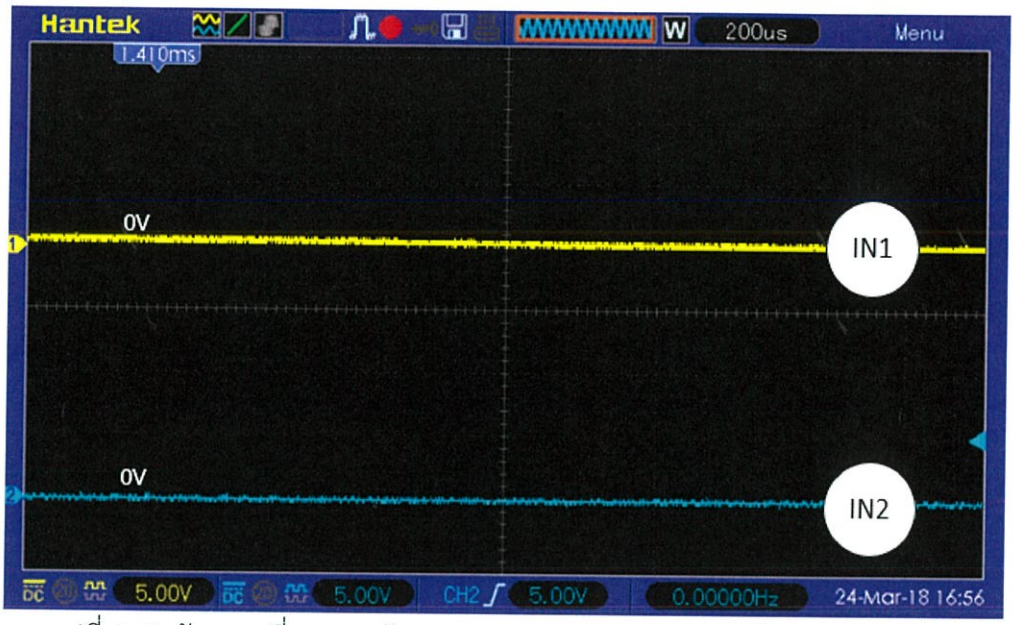
เมื่ออยู่ในสถานะสถานะหยุด (BREAK) วงจร H-Bridge 1 และวงจร H-Bridge 2 จะมีสัญญาณเอาต์พุต PWM เนื่องจากในการหยุดการทำงานของมอเตอร์จะไม่มีเอาต์พุตสัญญาณ PWM ให้กับมอเตอร์ จึงต้องทำการวัดทางอ้อมโดยการวัดที่อินพุตของวงจรแทนดังรูปที่ 4.19 และสัญญาณที่ขา IN1 กับ ขา IN2 ดังรูปที่ 4.19 และรูปที่ 4.21 ตามลำดับ



รูปที่ 4.19 สัญญาณอินพุต PWM ของวงจร H-Bridge 1 และ ในสถานะ BREAK



รูปที่ 4.20 สัญญาณที่ขา IN1 กับ ขา IN2 ของวงจร H-Bridge 1 ในสถานะ BREAK



รูปที่ 4.21 สัญญาณที่ขา IN1 กับ ขา IN2 ของวงจร H-Bridge 2 ในสถานะ BREAK

จากการให้บอร์ดไมโครคอนโทรลเลอร์ Arduino ควบคุมวงจร H-Bridge เพื่อขับเคลื่อนมอเตอร์ พบว่าสามารถทำการกำหนดความเร็วของมอเตอร์ด้วยความกว้างของลูกคลื่นสี่เหลี่ยม (Duty Cycle) ซึ่งจะส่งผลต่อความเร็วในการเคลื่อนที่ของเก้าอี้เข็น จึงต้องกำหนดด้วยความเหมาะสมโดยคำนึงถึงความปลอดภัยของผู้ใช้งาน รวมไปถึงสามารถกำหนดทิศทางการหมุนของมอเตอร์ของทั้ง 2 ล้อจากการป้อนแรงดันไฟฟ้าไปยังวงจร (IN1 และ IN2) ทำให้การใช้มอเตอร์ไฟฟ้าขับเคลื่อนเก้าอี้เข็นให้เคลื่อนที่ได้ในหลายทิศทาง

4.1.3 การทดสอบความเร็วของเก้าอี้เข็น

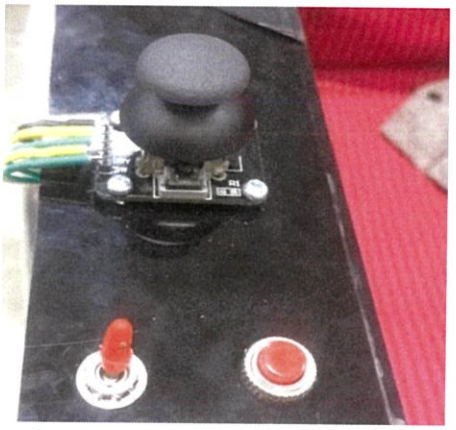
ในการทดสอบความเร็วของเก้าอี้เข็น โดยให้เก้าอี้เข็นเคลื่อนที่โดยบรรทุกโหลดที่เป็นผู้ใช้งานที่มีน้ำหนัก 70 กิโลกรัม, 80 กิโลกรัม และ 95 กิโลกรัม เป็นระยะทาง 5.5 เมตร ทำการจับเวลาที่เก้าอี้เข็นเคลื่อนที่ได้ แล้วนำมาคำนวณหาความเร็วเฉลี่ยที่เคลื่อนที่ได้ในแต่ละน้ำหนักที่บรรทุก ในตารางที่ 4.1 แสดงบันทึกเวลาที่เก้าอี้เข็นใช้ในการเคลื่อนที่เป็นระยะทาง เมตร

ตารางที่ 4.1 ผลการทดสอบความเร็วเก้าอี้เข็นในการเคลื่อนที่เป็นระยะ 5.5 เมตร

น้ำหนักของโหลด (kg)	การทดสอบครั้งที่	เวลาที่ใช้ในการเคลื่อนที่ (sec)	ความเร็วในการเคลื่อนที่ (m/s)
70	1	6.29	0.87
	2	6.33	0.87
	3	6.37	0.86
	เฉลี่ย		0.87
80	1	6.27	0.88
	2	6.15	0.89
	3	6.22	0.88
	เฉลี่ย		0.88
95	1	7.31	0.75
	2	7.11	0.77
	3	7.22	0.76
	เฉลี่ย		0.76

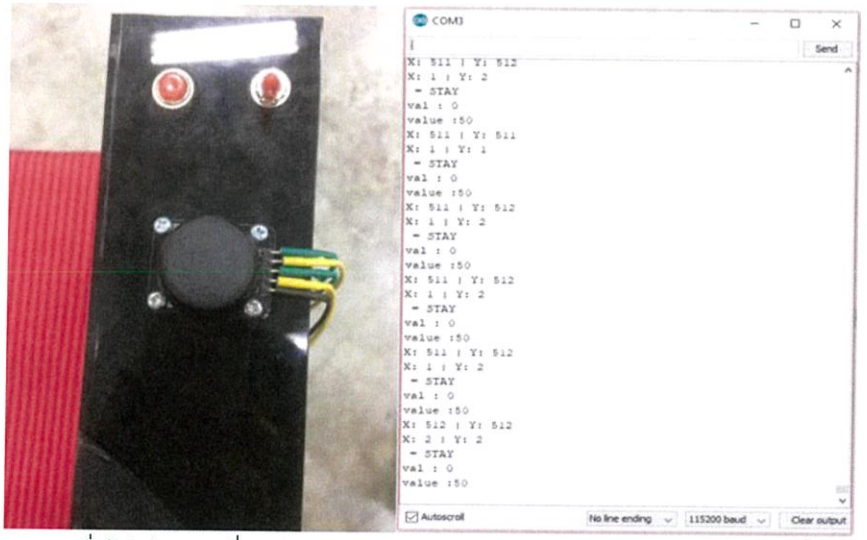
จากการทดสอบความเร็วในการเคลื่อนที่ของเก้าอี้เข็น พบว่าเก้าอี้เข็นสามารถขับเคลื่อนโดยรองรับโหลดที่เป็นน้ำหนักผู้ใช้งานได้ โดยน้ำหนักผู้ใช้งานอาจส่งผลต่อความเร็วของเก้าอี้เข็น หากโหลดซึ่งเป็นผู้ใช้งานมีน้ำหนักมาก ทำให้เก้าอี้เข็นมีความเร็วในการเคลื่อนที่ที่ช้าลง

4.2 การใช้ก้านควบคุม (Joystick) ในการควบคุมทิศทางเก้าอี้เข็น



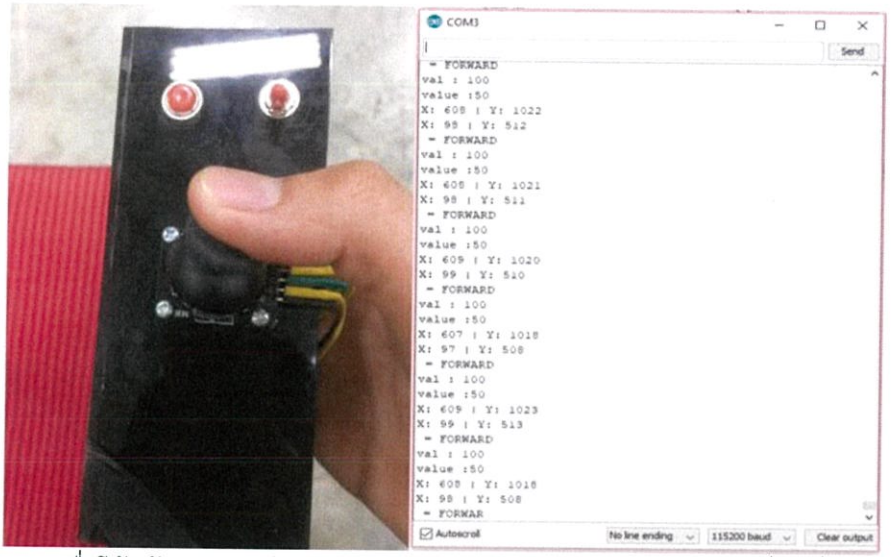
รูปที่ 4.22 ก้านควบคุมในการควบคุมทิศทางของเก้าอี้เข็น

ในรูปที่ 4.22 แสดงการทดลองใช้ก้านควบคุมในการควบคุมทิศทางของเก้าอี้เข็น โดยเมื่อบังคับก้านควบคุมไปในลักษณะการบังคับทิศทางที่กำหนด Serial Monitor จะแสดงผลทิศทางนั้นๆ ในรูปที่ 4.23 Serial Monitor จะแสดงผลสถานะหยุดนิ่ง (STAY) เมื่อไม่มีการเคลื่อนก้านควบคุม



รูปที่ 4.23 เมื่อไม่มีการเคลื่อนก้านควบคุม Serial Monitor แสดงสถานะหยุดนิ่ง (STAY)

เมื่อโน้มก้านควบคุมไปข้างหน้า Serial Monitor จะแสดงผลสถานะเคลื่อนที่ไปข้างหน้า (FORWARD) ดังรูปที่ 4.24

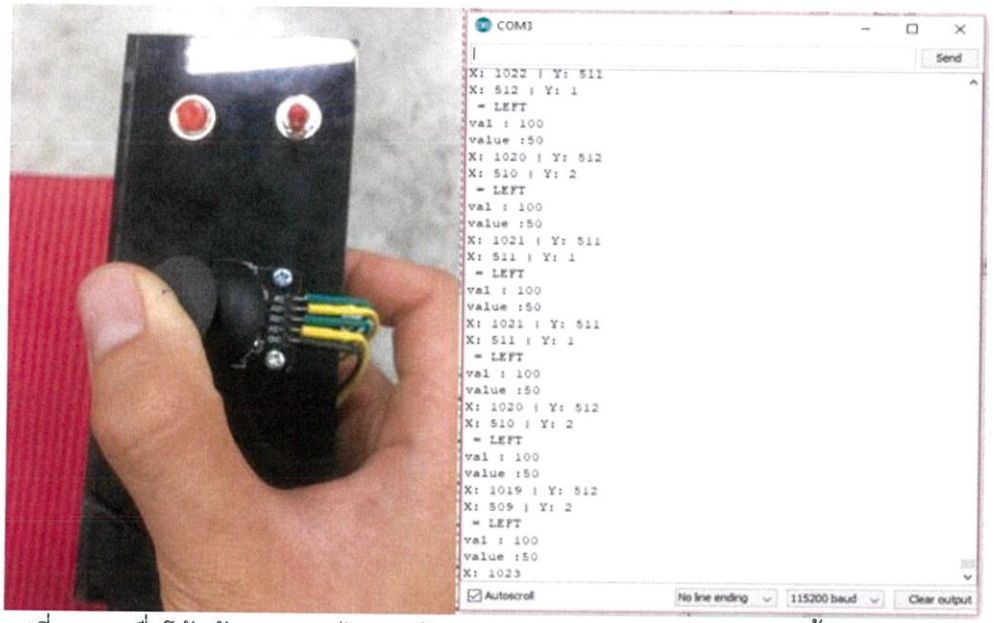


รูปที่ 4.24 เมื่อโน้มก้านควบคุมไปข้างหน้า Serial Monitor แสดงสถานะเคลื่อนที่ไปข้างหน้า (FORWARD)

เมื่อโน้มก้านควบคุมไปข้างหลัง Serial Monitor จะแสดงผลสถานะเคลื่อนที่ถอยหลัง (BACKWARD) ดังรูปที่ 4.25

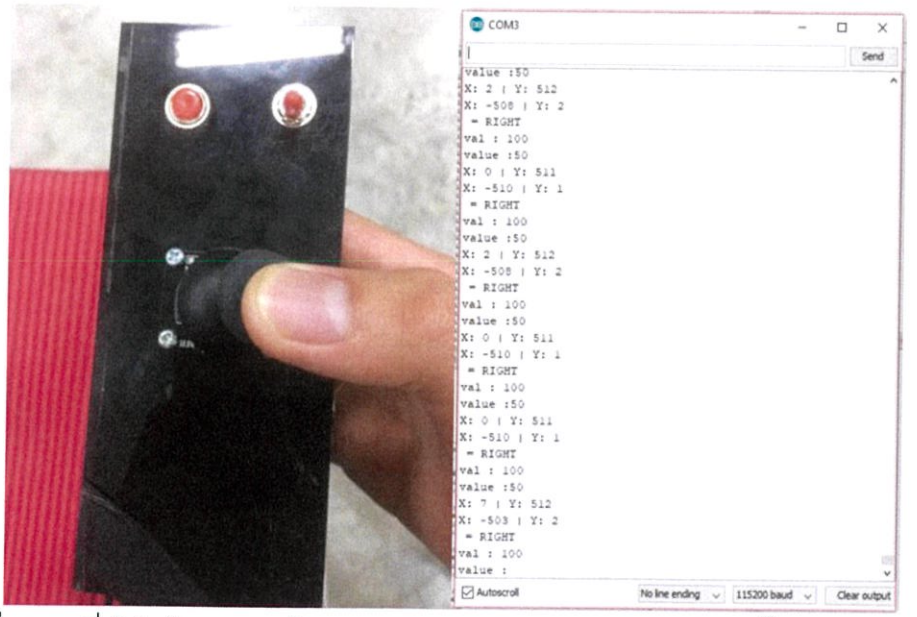
รูปที่ 4.25 เมื่อโน้มก้านควบคุมไปข้างหลัง Serial Monitor แสดงสถานะเคลื่อนที่ถอยหลัง (BACKWARD)

เมื่อโน้มก้านควบคุมไปทางซ้าย Serial Monitor จะแสดงผลสถานะเลี้ยวซ้าย(LEFT) ดังรูปที่ 4.26



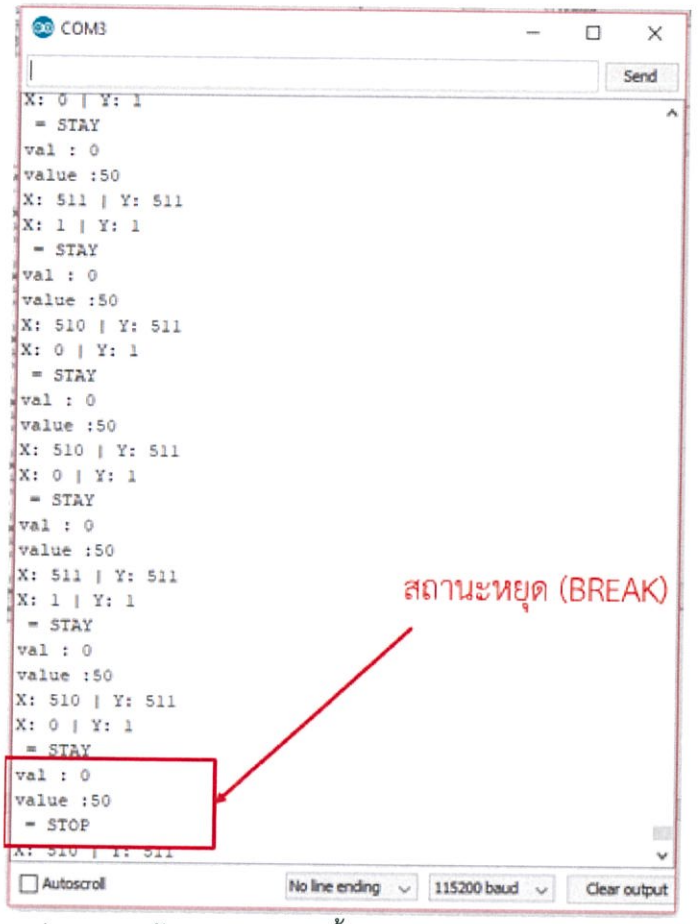
รูปที่ 4.26 เมื่อโน้มก้านควบคุมไปทางซ้าย Serial Monitor แสดงสถานะเลี้ยวซ้าย (LEFT)

เมื่อโน้มก้านควบคุมไปทางขวา Serial Monitor จะแสดงผลสถานะเลี้ยวขวา (RIGHT) ดังรูปที่ 4.27

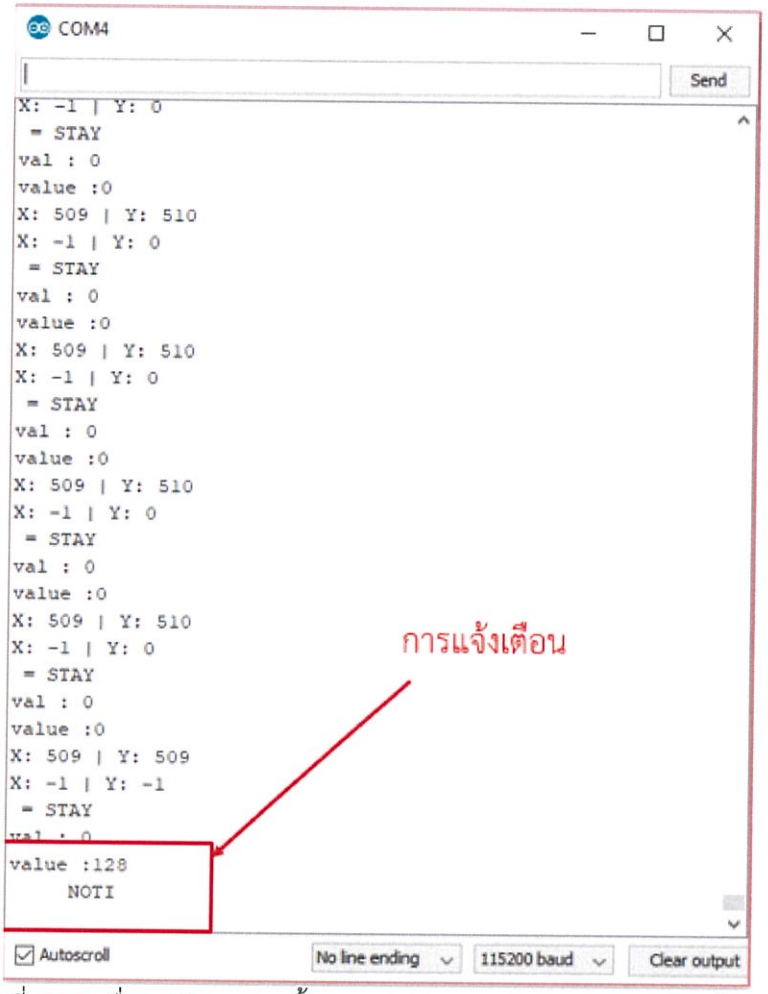


รูปที่ 4.27 เมื่อโน้มก้านควบคุมไปทางขวา Serial Monitor แสดงสถานะเลี้ยวขวา (RIGHT)

เมื่อกดปุ่มกดของก้านควบคุม 1 ครั้ง Serial Monitor จะแสดงผลสถานะหยุด (BREAK) ดังรูปที่ 4.28 และเมื่อกดปุ่มกด 1 ครั้ง Serial Monitor จะแสดงผลการเรียกผู้ดูแล ดังรูปที่ 4.29



รูปที่ 4.28 เมื่อกดปุ่มกดของก้านควบคุม 1 ครั้ง Serial Monitor แสดงสถานะหยุด (BREAK)

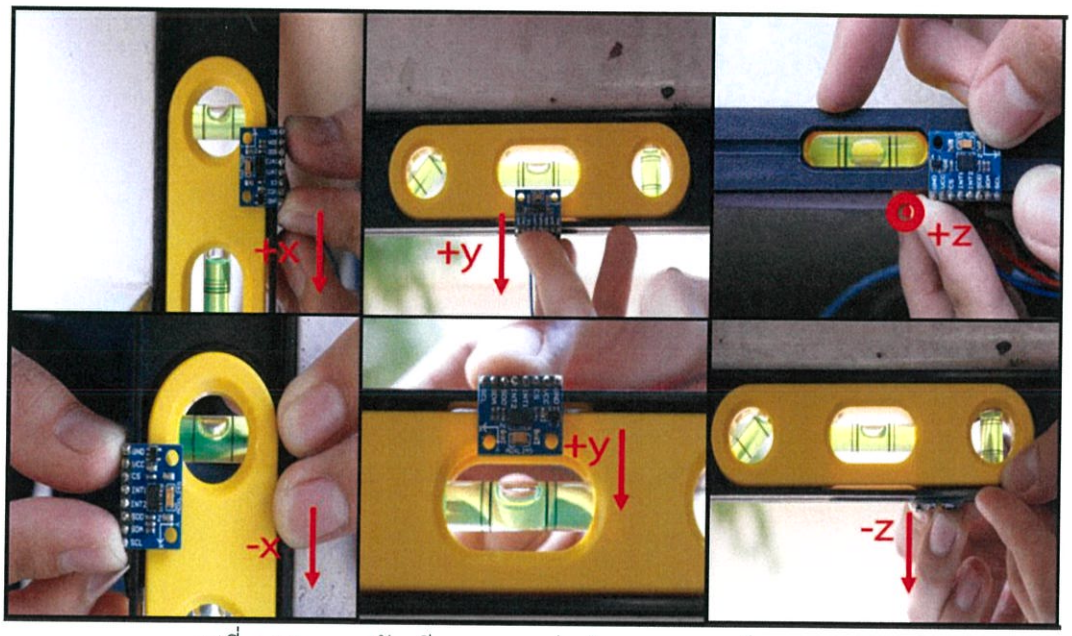


รูปที่ 4.29 เมื่อกดปุ่มกด 1 ครั้ง Serial Monitor แสดงสถานะเรียกผู้ดูแล

จากทดลองการใช้ก้านควบคุมในการควบคุมทิศทางการเคลื่อนที่เก้าอี้เข็น พบว่าสามารถใช้งานก้านควบคุมในการควบคุมทิศทางได้อย่างถูกต้องแม่นยำ และสามารถใช้การกดปุ่มเพื่อหยุดรถและการกดสวิตช์เพื่อเรียกผู้ดูแลได้

4.3 การควบคุมทิศทางเก้าอี้เข็นด้วยศีรษะโดยใช้เซนเซอร์ความเร่งเชิงเส้น ADXL345

4.3.1 การปรับเทียบความแม่นยำของเซนเซอร์ ADXL345



รูปที่ 4.30 การปรับเทียบความแม่นยำของเซนเซอร์ ADXL345

เมื่อจัดวางเซนเซอร์ ADXL345 ในตำแหน่งที่แรงโน้มถ่วงกระทำโดยตรงต่อแกนอ้างอิงของเซนเซอร์ในทุก ๆแกนดังรูปที่ 4.30 สังเกต Serial Monitorและบันทึกผลสามารถหาค่าความเร่งเชิงเส้นสูงสุดกระทำต่อแกนอ้างอิงบวก (A_{+1g}) และค่าความเร่งเชิงเส้นต่ำสุดกระทำต่อแกนอ้างอิงลบ (A_{-1g}) จะกระทำได้โดย จากนั้นคำนวณหาค่า Offset ($A_{OFF}[g]$) และค่า Gain ($Gain$) ของค่าความเร่งแต่ละแกน เพื่อนำไปหาค่าความเร่งเชิงเส้นที่ผ่านการชดเชยความคลาดเคลื่อนแล้ว ($A_{Calibrate[g]}$) ในรูปที่ 4.30 แสดงการเก็บค่าความเร่งเชิงเส้นสูงสุดและต่ำสุดในแต่ละแกน Serial Monitor ซึ่งเป็นข้อมูลเอาต์พุต (LSB) ที่ต้องนำไปคูณกับ ScaleFactor (3.9mg/LSB) เพื่อให้อยู่ในหน่วย g จึงจะนำไปคำนวณค่าในการปรับเทียบได้ ค่าในการปรับเทียบที่คำนวณได้แสดงในตารางที่ 4.2 และตารางที่ 4.3 โดยเป็นค่าปรับเทียบสำหรับเซนเซอร์ที่อยู่ในส่วนควบคุมทิศทางบนศีรษะและเซนเซอร์ที่อยู่กับเก้าอี้เข็นตามลำดับ

ตารางที่ 4.2 ค่าที่ใช้ในการปรับเทียบความแม่นยำของเซนเซอร์ ADXL345 ในส่วนควบคุมทิศทาง

แกน	ความเร่งเชิงเส้นสูงสุดในแกนอ้างอิงบวก (A_{+1g})	ความเร่งเชิงเส้นต่ำสุดในแกนอ้างอิงลบ (A_{-1g})	Offset ($A_{OFF}[g]$)	Gain ($Gain$)
X	0.9438	-1.0491	-0.0527	0.9965
Y	0.9516	-1.0452	-0.0468	0.9984
Z	0.9282	-1.0218	-0.0468	0.9750

เมื่อได้ค่า $A_{OFF}[g]$ และค่า $Gain$ แล้ว จึงนำไปเข้าสมการเพื่อหาค่า $A_{Calibrate[g]}$ สำหรับเซนเซอร์ที่อยู่ในส่วนควบคุมทิศทางบนศีรษะต่อไปได้ ดังสมการที่ (4.1) ถึง (4.3)

$$X_{Calibrate[g]} = \frac{X_{out[g]} - (-0.0527)}{0.9965} \tag{4.1}$$

$$Y_{Calibrate[g]} = \frac{Y_{out[g]} - (-0.0468)}{0.9984} \tag{4.2}$$

$$Z_{Calibrate[g]} = \frac{Z_{out[g]} - (-0.0468)}{0.9750} \tag{4.3}$$

ตารางที่ 4.3 ค่าที่ใช้ในการปรับเทียบความแม่นยำของเซนเซอร์ ADXL345 อยู่กับเก้าอี้เข็น

แกน	ความเร่งเชิงเส้นสูงสุดในแกนอ้างอิงบวก (A_{+1g})	ความเร่งเชิงเส้นต่ำสุดในแกนอ้างอิงลบ (A_{-1g})	Offset ($A_{OFF}[g]$)	Gain ($Gain$)
X	0.9867	-1.0101	-0.0117	0.9984
Y	0.9438	-1.0725	-0.0644	1.0082
Z	1.0881	-0.8931	0.0975	0.9906

เมื่อได้ค่า $A_{OFF}[g]$ และค่า $Gain$ แล้ว จึงนำไปเข้าสมการเพื่อหาค่า $A_{Calibrate[g]}$ สำหรับเซนเซอร์ที่อยู่กับเก้าอี้เข็นต่อไปได้ ดังสมการที่ (4.4) ถึง (4.5)

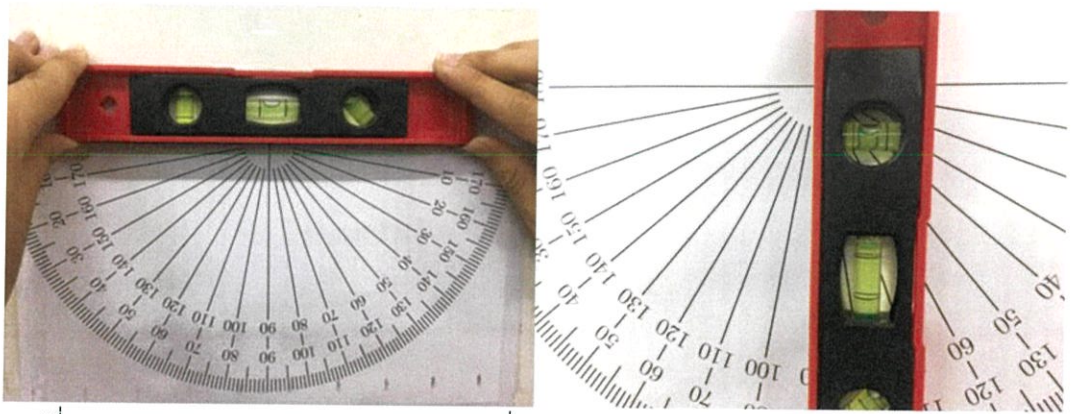
$$X_{Calibrate[g]} = \frac{X_{out[g]} - (-0.0117)}{0.9984} \tag{4.4}$$

$$Y_{Calibrate[g]} = \frac{Y_{out[g]} - (-0.0644)}{1.0082} \tag{4.5}$$

$$Z_{Calibrate[g]} = \frac{Z_{out[g]} - (0.0975)}{0.9906} \tag{4.6}$$

4.3.2 การทดสอบการตรวจวัดมุมเอียงของเซนเซอร์ ADXL345

ในส่วนของการทดสอบการตรวจวัดมุมเอียงของเซนเซอร์ความเร่ง ADXL345 นั้นจะทดสอบโดยอ้างอิงกับลักษณะการเอียงศีรษะที่สอดคล้องกับทิศทางการเคลื่อนที่ของเก้าอี้เข็น แบ่งการทดสอบออกเป็น 2 ส่วน ได้แก่ การทดสอบการตรวจวัดมุมเอียงของเซนเซอร์ในระนาบการเอียง X-Z และการทดสอบการตรวจวัดมุมเอียงของเซนเซอร์ในระนาบการเอียง Y-Z โดยจะทำการตรวจวัดมุมเอียงที่มุม -90 องศา, -60 องศา, -30 องศา, 0 องศา, 30 องศา, 60 องศา และ 90 องศา โดยที่ค่ามุมที่เป็นค่าลบนั้นเป็นมุมที่กระทำต่อแกนอ้างอิงที่เป็นแกนลบของเซนเซอร์ จากนั้นทำการบันทึกค่ามุม θ, มุม ψ และมุม ϕ ที่สังเกตได้จากบน Serial Monitor แล้วจึงคำนวณหาความคลาดเคลื่อนของในการวัดมุมเอียง

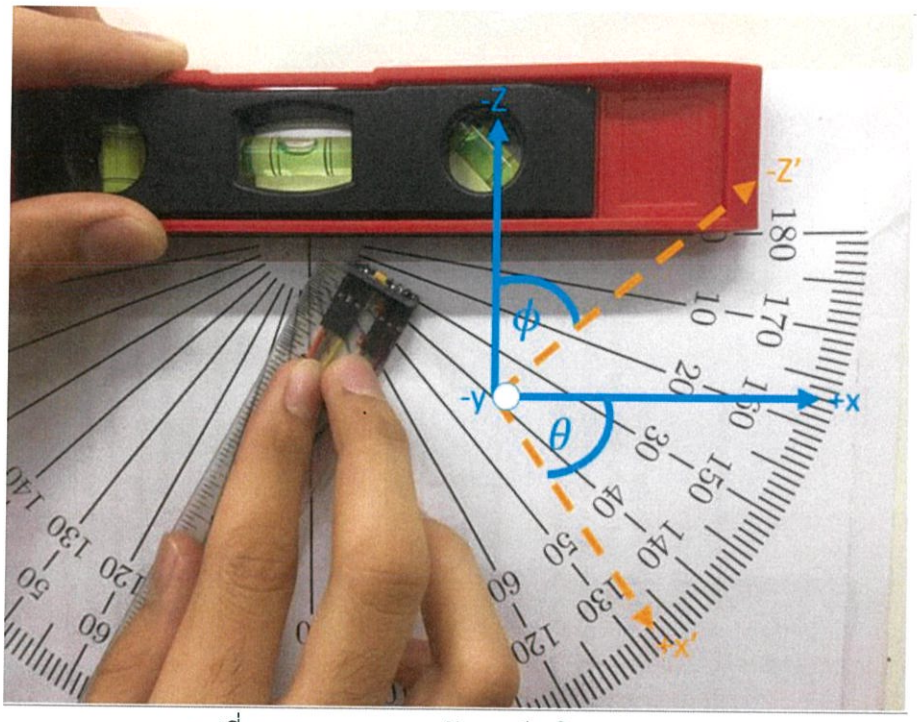


รูปที่ 4.31 การตรวจสอบแนวระดับและแนวดิ่งของโปรแทรกเตอร์อ้างอิงในการตรวจวัดมุมเอียง

จากรูปที่ 4.31 แสดงการตรวจวัดแนวระดับและแนวดิ่งของโปรแทรกเตอร์ที่ใช้อ้างอิงในการตรวจวัดมุมเอียง เพื่อใช้เป็นจุดอ้างอิงการเอียงของเซนเซอร์ความเร่ง ADXL345 ในแต่ละมุม ซึ่งจะเห็นได้ว่าในแนวระดับนั้นโปรแทรกเตอร์ถูกติดตั้งได้ระดับโดยสังเกตจากฟองน้ำที่อยู่กึ่งกลาง

ขีดกำหนดระดับพอดี และในแนวดิ่งโปรแทรกเตอร์ได้ติดตั้งโดยได้ระดับในแนวดิ่ง สังเกตจากฟองน้ำที่กึ่งกลางขีดกำหนดระดับพอดี จึงสามารถใช้โปรแทรกเตอร์ดังกล่าวเป็นจุดอ้างอิงการเอียงของเซนเซอร์ได้

4.3.2.1 ผลการทดสอบการตรวจวัดมุมเอียงของเซนเซอร์ในระนาบการเอียง X-Z



รูปที่ 4.32 การทดสอบวัดมุมเอียงในระนาบ X-Z

ในรูปที่ 4.32 แสดงการตรวจวัดมุมเอียงในระนาบ X-Z เนื่องจากเป็นระนาบที่กำหนดให้สอดคล้องกับการเอียงของศีรษะไปทางด้านซ้ายและด้านขวา ซึ่งแทนการควบคุมเก้าอี้เข็นให้มีสถานะทิศทางเป็นเลี้ยวซ้ายและขวาตามลำดับ โดยทำการสุ่มค่ามุมเอียงในแต่ละแกนจาก Serial Monitor ทำการบันทึกผลการวัดทั้งหมด 3 ครั้ง นำมาหาค่าเฉลี่ยเพื่อใช้หาความคลาดเคลื่อนของการตรวจวัดมุมเอียงของเซนเซอร์ ทั้งนี้จะบันทึกผลการทดสอบการวัดมุมเอียงของเซนเซอร์ก่อนและหลังปรับความแม่นยำของเซนเซอร์ โดยในตารางที่ 4.4 ถึงตารางที่ 4.6 แสดงผลการทดสอบตรวจวัดมุมเอียงของเซนเซอร์ในระนาบการเอียง X-Z ก่อนการปรับเทียบความแม่นยำ

ตารางที่ 4.4 ผลการทดสอบตรวจวัดมุม θ ของเซนเซอร์ในระนาบการเคลื่อนไหว X-Z ก่อนปรับเทียบความแม่นยำ

มุมที่ทำการวัด (องศา)	มุม θ (องศา)				ความคลาดเคลื่อน (องศา)
	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ค่ามุมเฉลี่ย	
-90	-87.79	-87.53	-87.73	-87.68	2.32
-60	-63.04	-63.23	-63.03	-63.10	3.10
-30	-36.40	-34.94	-34.63	-35.32	5.32
0	-4.27	-4.25	-4.01	-4.18	4.18
30	26.98	26.88	26.26	26.71	3.29
60	59.04	59.37	59.06	59.16	0.84
90	88.47	87.16	88.28	87.97	2.03

ตารางที่ 4.5 ผลการทดสอบตรวจวัดมุม ψ ของเซนเซอร์ในระนาบการเคลื่อนไหว X-Z ก่อนปรับเทียบความแม่นยำ

มุมที่ทำการวัด (องศา)	ค่ามุมจริง (องศา)	มุม ψ (องศา)				ความคลาดเคลื่อน (องศา)
		ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ค่ามุมเฉลี่ย	
-90	0	-0.63	-0.84	-0.84	-0.77	0.77
-60	0	-0.43	-0.65	-1.08	-0.72	0.72
-30	0	-1.37	-1.37	-1.14	-1.29	1.29
0	0	-2.61	-2.36	-2.60	-2.52	2.52
30	0	-1.94	-1.69	-1.95	-1.86	1.86
60	0	-1.70	-1.70	-1.22	-1.54	1.54
90	0	-2.61	-2.36	-2.60	-2.52	2.52

ตารางที่ 4.6 ผลการทดสอบตรวจวัดมุม ϕ ของเซนเซอร์ในระนาบการเคลื่อนไหว X-Z ก่อนปรับเทียบความแม่นยำ

มุมที่ทำการวัด (องศา)	มุม ϕ (องศา)				ความคลาดเคลื่อน (องศา)
	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ค่ามุมเฉลี่ย	
-90	92.11	92.32	92.11	92.18	2.18
-60	63.05	63.24	63.05	63.11	3.11
-30	36.44	34.98	34.65	35.36	5.36
0	5.00	4.86	4.78	4.88	4.88
30	27.06	26.94	26.63	26.88	3.12
60	59.09	59.43	59.09	59.20	0.80
90	90.96	92.41	90.95	91.44	1.44

ในตารางที่ 4.7 ถึงตารางที่ 4.9 แสดงผลการทดสอบตรวจวัดมุมเอียงของเซนเซอร์ในระนาบการเอียง X-Z หลังการปรับเทียบความแม่นยำ

ตารางที่ 4.7 ผลการทดสอบตรวจวัดมุม θ ของเซนเซอร์ในระนาบการเคลื่อนไหว X-Z หลังปรับเทียบความแม่นยำ

มุมที่ทำการวัด (องศา)	มุม θ (องศา)				ความคลาดเคลื่อน (องศา)
	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ค่ามุมเฉลี่ย	
-90	-89.20	-88.89	-89.89	-89.33	0.67
-60	-60.57	-60.52	-60.50	-60.53	0.53
-30	-30.19	-30.27	-30.08	-30.18	0.18
0	0.22	0.22	0.45	0.30	0.30
30	30.11	30.22	30.11	30.15	0.15
60	60.39	60.51	60.40	60.43	0.43
90	89.78	89.50	90.00	89.76	0.24

ตารางที่ 4.8 ผลการทดสอบตรวจวัดมุม ψ ของเซนเซอร์ในระนาบการเคลื่อนไหว X-Z หลังปรับเทียบความแม่นยำ

มุมที่ทำการวัด (องศา)	ค่ามุมจริง (องศา)	มุม ψ (องศา)				ความคลาดเคลื่อน (องศา)
		ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ค่ามุมเฉลี่ย	
-90	0	0.67	0.44	0.67	0.59	0.59
-60	0	2.23	1.55	2.00	1.93	1.93
-30	0	0.45	0.23	0.67	0.45	0.45
0	0	-0.22	-0.44	-0.45	-0.37	0.37
30	0	0.23	0.45	0.23	0.30	0.30
60	0	0.23	0.23	0.00	0.15	0.15
90	0	-0.22	-0.23	-0.22	-0.22	0.22

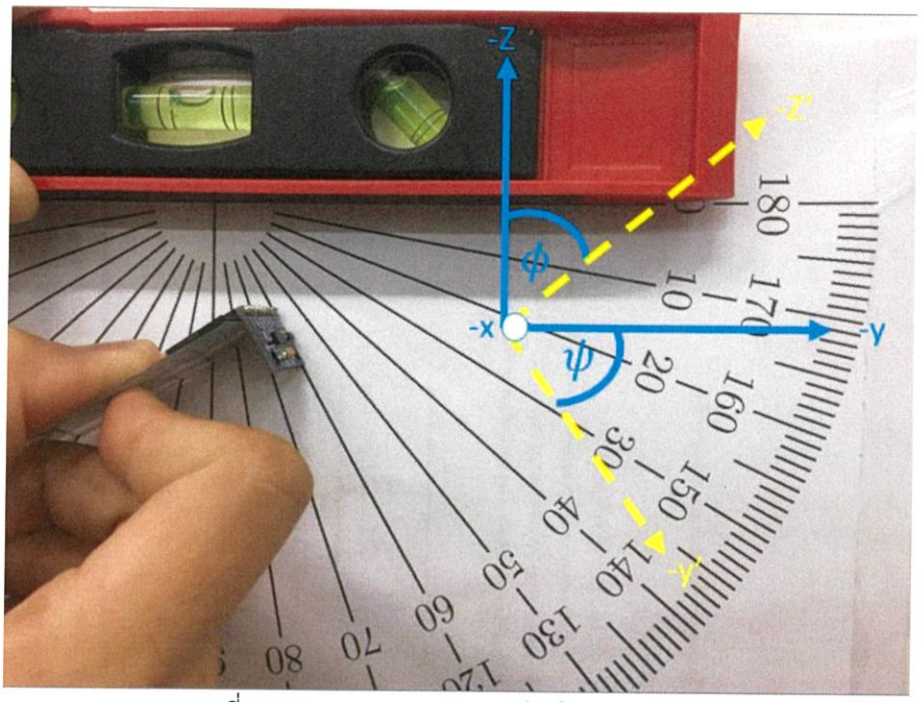
ตารางที่ 4.9 ผลการทดสอบตรวจวัดมุม ϕ ของเซนเซอร์ในระนาบการเคลื่อนไหว X-Z หลังปรับเทียบความแม่นยำ

มุมที่ทำการวัด (องศา)	มุม ϕ (องศา)				ความคลาดเคลื่อน (องศา)
	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ค่ามุมเฉลี่ย	
-90	89.11	89.33	88.89	89.11	0.89
-60	60.67	60.57	60.58	60.61	0.61
-30	30.19	30.28	30.20	30.22	0.22
0	0.22	0.45	0.50	0.39	0.39
30	30.11	30.23	30.11	30.15	0.15
60	60.40	60.51	60.40	60.44	0.44
90	90.22	90.45	90.22	90.30	0.30

เมื่อพิจารณาค่ามุมที่วัดได้ในระนาบการเอียง X-Z โดยในระนาบนี้จะมีค่ามุมที่เกิดขึ้นอยู่ด้วยกัน 2 มุม คือ มุม θ ที่กระทำต่อแกน X และมุม ϕ ที่กระทำต่อแกน Z ในส่วนมุม ψ นั้นจะมีค่ามุมใกล้เคียง 0 องศา เนื่องจากแกน Y จะถูกใช้เป็นจุดหมุนในการเอียงของเซนเซอร์ จึงทำให้ไม่เกิดมุมกระทำกับแกน Y ซึ่งเมื่อพิจารณาความคลาดเคลื่อนในการตรวจวัดมุม

จะเห็นว่าในการตรวจวัดมุมหลังการปรับเทียบความแม่นยำนั้นมีค่าความคลาดเคลื่อนที่น้อยกว่าเทียบกับการตรวจวัดมุมก่อนการปรับเทียบความแม่นยำ

4.3.2.2 ผลการทดสอบการตรวจวัดมุมเอียงของเซนเซอร์ในระนาบการเคลื่อนไหว Y-Z



รูปที่ 4.33 การทดสอบวัดมุมเอียงในระนาบ Y-Z

ในรูปที่ 4.33 แสดงการตรวจวัดมุมเอียงในระนาบ Y-Z ระนาบ ซึ่งเป็นระนาบที่กำหนดให้สอดคล้องกับการเอียงของศีรษะไปทางด้านหน้าและด้านหลัง ซึ่งแทนการควบคุมเก้าอี้เข็นให้มีสถานะทิศทางเป็นเคลื่อนที่ไปข้างหน้าและถอยหลังตามลำดับ โดยทำการสุ่มค่ามุมเอียงในแต่ละแกนจาก Serial Monitor ทำการบันทึกผลการวัดทั้งหมด 3 ครั้ง นำมาหาค่าเฉลี่ยเพื่อใช้หาความคลาดเคลื่อนของการตรวจวัดมุมเอียงของเซนเซอร์ ในตารางที่ 4.10 ถึงตารางที่ 4.12 แสดงผลการทดสอบตรวจวัดมุมเอียงของเซนเซอร์ในระนาบการเคลื่อนไหว Y-Z ก่อนการปรับเทียบความแม่นยำ

ตารางที่ 4.10 ผลการทดสอบตรวจวัดมุม θ ของเซนเซอร์ในระนาบการเคลื่อนไหว Y-Z ก่อนการปรับเทียบความแม่นยำ

มุมที่ทำการวัด (องศา)	ค่ามุมจริง (องศา)	มุม θ (องศา)				ความคลาดเคลื่อน (องศา)
		ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ค่ามุมเฉลี่ย	
-90	0	-1.94	-1.28	-1.72	-1.65	1.65
-60	0	-1.12	-1.78	-1.55	-1.48	1.48
-30	0	-1.63	-1.62	-1.40	-1.55	1.55
0	0	-3.09	-3.07	-2.84	-3.00	3.00
30	0	-1.69	-1.91	-1.92	-1.84	1.84
60	0	0.48	0.24	1.20	0.64	0.64
90	0	-0.23	-1.87	-2.33	-1.48	1.48

ตารางที่ 4.11 ผลการทดสอบตรวจวัดมุม ψ ของเซนเซอร์ในระนาบการเคลื่อนไหว Y-Z ก่อนการปรับเทียบความแม่นยำ

มุมที่ทำการวัด (องศา)	มุม ψ (องศา)				ความคลาดเคลื่อน (องศา)
	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ค่ามุมเฉลี่ย	
-90	-87.54	-87.33	-86.72	-87.20	2.80
-60	-61.82	-63.48	-62.41	-62.57	2.57
-30	-31.75	-31.54	-32.06	-31.78	1.78
0	-0.95	-0.71	-0.72	-0.79	0.79
30	28.92	29.22	31.27	29.80	0.20
60	61.94	62.58	60.62	61.71	1.71
90	87.19	86.82	87.16	87.06	2.94

ตารางที่ 4.12 ผลการทดสอบตรวจวัดมุม ϕ ของเซนเซอร์ในระนาบการเคลื่อนไหว Y-Z ก่อนการปรับเทียบความแม่นยำ

มุมที่ทำการวัด (องศา)	มุม ϕ (องศา)				ความคลาดเคลื่อน (องศา)
	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ค่ามุมเฉลี่ย	
-90	91.51	92.34	92.80	92.22	2.22
-60	61.84	63.55	62.46	62.62	2.62
-30	32.41	31.90	31.60	31.97	1.97
0	1.53	1.91	2.13	1.86	1.86
30	28.98	29.30	31.35	29.88	0.12
60	61.94	62.58	60.65	61.72	1.72
90	92.80	92.57	91.63	92.33	2.33

ในตารางที่ 4.13 ถึงตารางที่ 4.15 แสดงผลการทดสอบตรวจวัดมุมเอียงของเซนเซอร์ในระนาบการเคลื่อนไหว Y-Z หลังการปรับเทียบความแม่นยำ

ตารางที่ 4.13 ผลการทดสอบตรวจวัดมุม θ ของเซนเซอร์ในระนาบการเคลื่อนไหว Y-Z หลังการปรับเทียบความแม่นยำ

มุมที่ทำการวัด (องศา)	ค่ามุมจริง (องศา)	มุม θ (องศา)				ความคลาดเคลื่อน (องศา)
		ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ค่ามุมเฉลี่ย	
-90	0	0.22	0.23	0.22	0.22	0.22
-60	0	-0.23	-0.45	-0.22	-0.30	0.30
-30	0	0.00	-0.23	-0.23	-0.15	0.15
0	0	-0.22	0.22	0.22	0.07	0.07
30	0	0.22	0.45	0.22	0.30	0.30
60	0	0.22	0.23	0.45	0.30	0.30
90	0	0.22	0.22	0.00	0.15	0.15

ตารางที่ 4.14 ผลการทดสอบตรวจวัดมุม ψ ของเซนเซอร์ในระนาบการเคลื่อนไหว Y-Z หลังการปรับเทียบความแม่นยำ

มุมที่ทำการวัด (องศา)	มุม ψ (องศา)				ความคลาดเคลื่อน (องศา)
	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ค่ามุมเฉลี่ย	
-90	-89.78	-89.77	-90.00	-89.85	0.15
-60	-60.31	-60.30	-60.12	-60.24	0.24
-30	-30.11	-30.42	-30.31	-30.28	0.28
0	-0.22	-0.45	-0.23	-0.30	0.30
30	30.47	30.27	30.16	30.30	0.30
60	60.11	60.31	60..22	60.21	0.21
90	89.56	89.30	89.78	89.55	0.45

ตารางที่ 4.15 ผลการทดสอบตรวจวัดมุม ϕ ของเซนเซอร์ในระนาบการเคลื่อนไหว Y-Z หลังการปรับเทียบความแม่นยำ

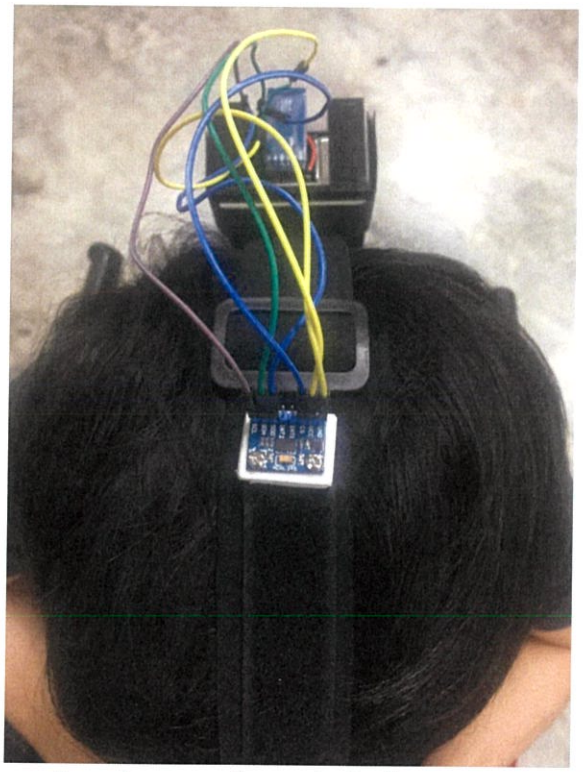
มุมที่ทำการวัด (องศา)	มุม ϕ (องศา)				ความคลาดเคลื่อน (องศา)
	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	ค่ามุมเฉลี่ย	
-90	90.00	90.23	90.22	90.15	0.15
-60	60.31	60.51	60.31	60.38	0.38
-30	30.31	30.42	30.28	30.34	0.34
0	0.22	0.45	0.32	0.33	0.33
30	30.47	30.28	30.39	30.38	0.38
60	60.12	60.31	60.22	60.22	0.22
90	90.44	90.22	90.00	90.22	0.22

เมื่อพิจารณาค่ามุมที่วัดได้ในระนาบการเอียง Y-Z โดยในระนาบนี้จะมีค่ามุมที่เกิดขึ้นอยู่ด้วยกัน 2 มุม คือ มุม ψ ที่กระทำต่อแกน Y และมุม ϕ ที่กระทำต่อแกน Z ในส่วนมุม θ นั้นจะมีค่ามุมใกล้เคียง 0 องศา เนื่องจากแกน X จะถูกใช้เป็นจุดหมุนในการเอียงของเซนเซอร์ จึงทำให้ไม่เกิดมุมกระทำกับแกน X ซึ่งเมื่อพิจารณาความคลาดเคลื่อนในการตรวจวัดมุม

จะเห็นว่าในการตรวจวัดมุมหลังการปรับเทียบความแม่นยำนั้นมีค่าความคลาดเคลื่อนที่น้อยกว่าเทียบกับการตรวจวัดมุมก่อนการปรับเทียบความแม่นยำ

จากการทดสอบการตรวจวัดมุมเอียงของเซนเซอร์ความเร่งเชิงเส้น ADXL345 พบว่าเซนเซอร์ ADXL345 สามารถวัดมุมเอียงได้ใกล้เคียงกับค่ามุมจริง ซึ่งเมื่อพิจารณาค่าความคลาดเคลื่อนหลังการปรับเทียบแล้วในการวัดมุมเอียงจะเห็นได้ว่ามีความคลาดเคลื่อนสูงสุด 1.93 องศา (จากตารางที่ 4.8) จากเดิมก่อนการปรับเทียบมีความคลาดเคลื่อนในการวัดมุมเอียงสูงสุด 5.36 องศา (จากตารางที่ 4.6)

4.3.3 การควบคุมทิศทางเก้าอี้เข็นด้วยศีรษะ

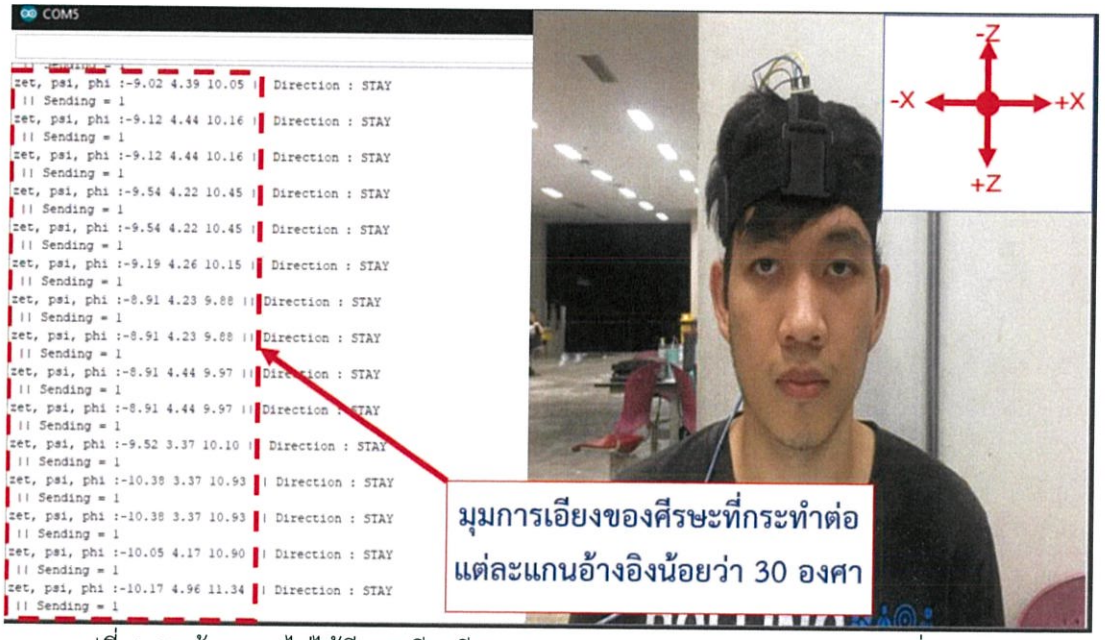


รูปที่ 4.34 ตำแหน่งเซนเซอร์ความเร่งเชิงเส้น ADXL345 บนศีรษะ

จากการทดสอบการใช้ศีรษะควบคุมทิศทางในการเคลื่อนที่ โดยให้ผู้ทดลองสวมใส่อุปกรณ์บนศีรษะ ให้เซนเซอร์ ADXL345 มีตำแหน่งอยู่บนศีรษะดังรูปที่ 4.34 จากนั้นจะทำการเอียงศีรษะไปในทิศทางทั้ง 5 ทิศทาง ได้แก่ หยุดนิ่ง (ศีรษะตั้งตรง), เคลื่อนที่ไปข้างหน้า (ก้มศีรษะลง), เลี้ยวซ้าย (เอียงศีรษะไปทางซ้าย), เลี้ยวขวา (เอียงศีรษะไปทางขวา) และถอยหลัง (เงยศีรษะขึ้น) จะพบว่า Serial Monitor ของบอร์ดไมโครคอนโทรลเลอร์ในส่วนควบคุมทิศทางบนศีรษะ

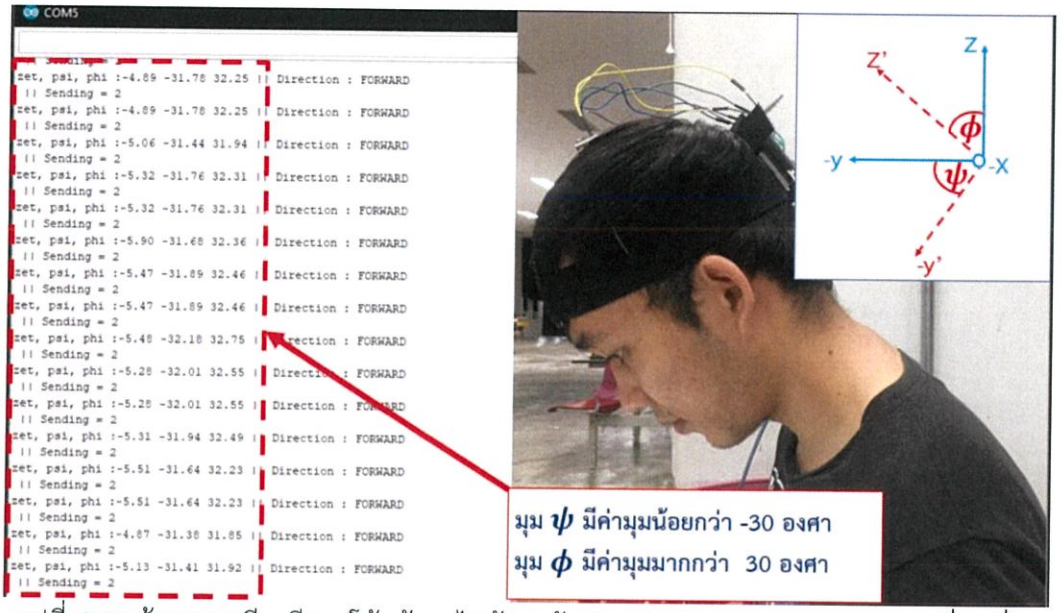
แสดงผลทิศทางไปตามการเอียงศีรษะของผู้ทดลองพร้อมกับแสดงผลค่ามุมเอียงที่กระทำต่อแต่ละแกนอ้างอิง (θ,ψ,ϕ)

เมื่อผู้ทดลองไม่ได้มีการเอียงศีรษะไปในทิศทางใดๆ Serial Monitor แสดงผลสถานะหยุดนิ่ง (STAY) ดังรูปที่ 4.35 ซึ่งมีมุมการเอียงของศีรษะที่กระทำต่อแต่ละแกนอ้างอิงน้อยว่า 30 องศา



รูปที่ 4.35 ผู้ทดลองไม่ได้มีการเอียงศีรษะ Serial Monitor แสดงสถานะหยุดนิ่ง (STAY)

เมื่อผู้ทดลองเอียงศีรษะโน้มก้มลงไปข้างหน้า Serial Monitor แสดงผลสถานะเคลื่อนที่ไปข้างหน้า (FORWARD) ดังรูปที่ 4.36 ซึ่งมีมุมการเอียงของศีรษะได้แก่ มุม ψ ที่กระทำต่อแกนอ้างอิง -Y จะมีค่ามุมน้อยกว่า -30 องศา และมุม ϕ ที่กระทำต่อแกนอ้างอิง Z จะมีค่ามุมมากกว่า 30 องศา

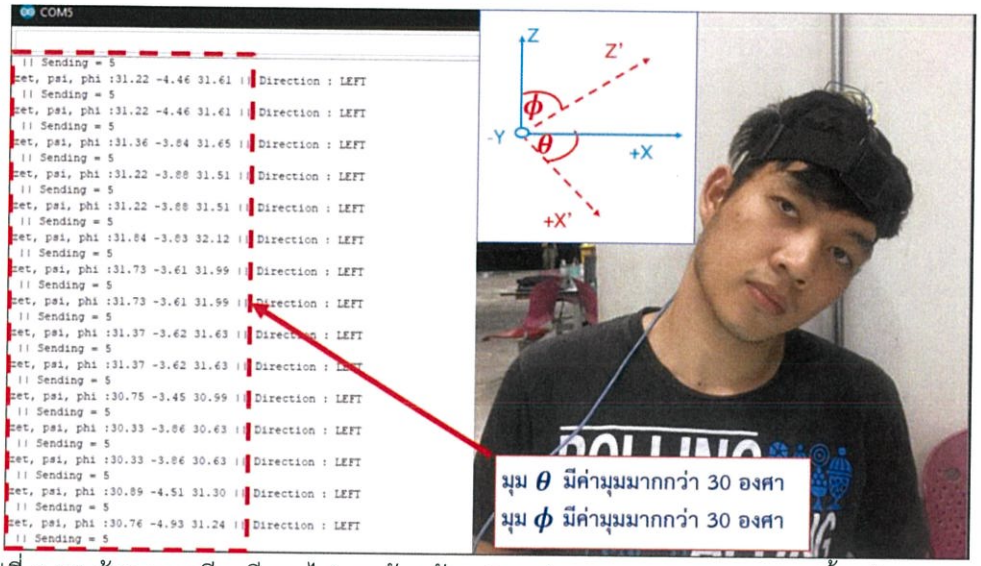


รูปที่ 4.36 ผู้ทดลองเอียงศีรษะโน้มก้มลงไปข้างหน้า Serial Monitor แสดงสถานะเคลื่อนที่ไปข้างหน้า (FORWARD)

เมื่อผู้ทดลองเงยศีรษะขึ้นไปข้างหลัง Serial Monitor แสดงผลสถานะถอยหลัง (BACKWARD) ดังรูปที่ 4.37 ซึ่งมีมุมการเอียงของศีรษะได้ มุม ψ ที่กระทำต่อแกนอ้างอิง Y จะมีค่ามุมมากกว่า 30 องศา และมุม ϕ ที่กระทำต่อแกนอ้างอิง Z จะมีค่ามุมมากกว่า 30 องศา

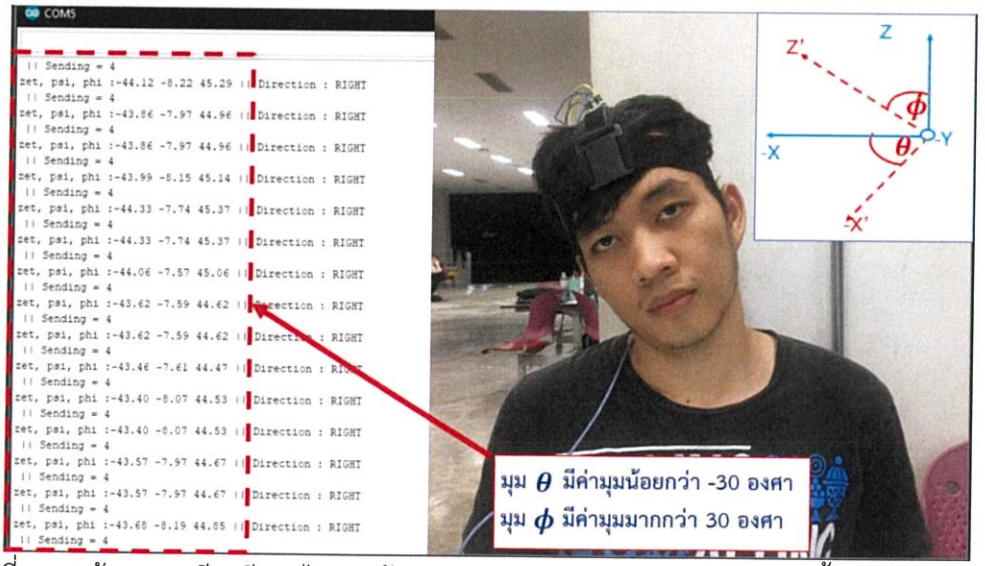
รูปที่ 4.37 ผู้ทดลองเงยศีรษะขึ้นไปข้างหลัง Serial Monitor แสดงสถานะถอยหลัง (BACKWARD)

เมื่อผู้ทดลองเอียงศีรษะไปทางด้านซ้าย Serial Monitor แสดงผลสถานะเลี้ยวซ้าย (LEFT) ดังรูปที่ 4.38 ซึ่งมีมุมการเอียงของศีรษะได้ มุม θ ที่กระทำต่อแกนอ้างอิง X จะมีค่ามุมมากกว่า 30 องศา และมุม ϕ ที่กระทำต่อแกนอ้างอิง Z จะมีค่ามุมมากกว่า 30 องศา



รูปที่ 4.38 ผู้ทดลองเอียงศีรษะไปทางด้านซ้าย Serial Monitor แสดงสถานะเลี้ยวซ้าย (LEFT)

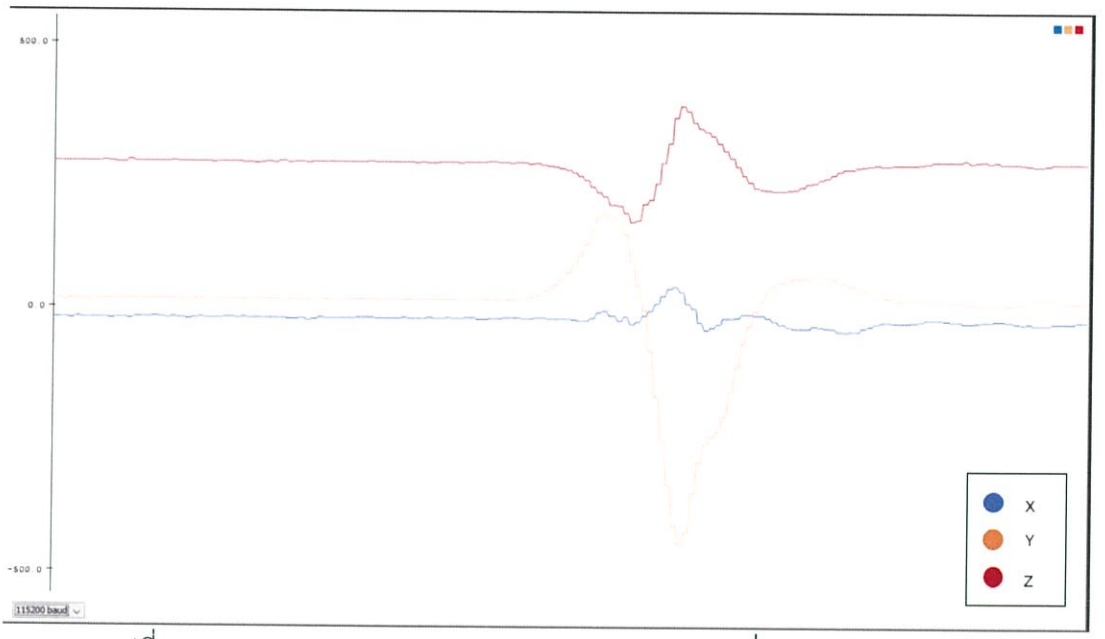
เมื่อผู้ทดลองเอียงศีรษะไปทางด้านขวา Serial Monitor แสดงผลสถานะเลี้ยวขวา (RIGHT) ดังรูปที่ 4.39 ซึ่งมีมุมการเอียงของศีรษะได้ มุม θ ที่กระทำต่อแกนอ้างอิง X จะมีค่ามุมน้อยกว่า -30 องศา และมุม ϕ ที่กระทำต่อแกนอ้างอิง Z จะมีค่ามุมมากกว่า 30 องศา



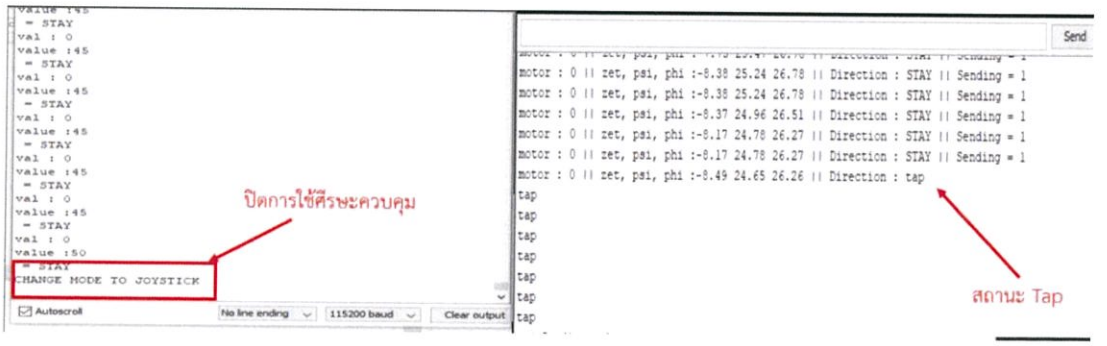
รูปที่ 4.39 ผู้ทดลองเอียงศีรษะไปทางด้านขวา Serial Monitor แสดงสถานะเลี้ยวขวา (RIGHT)

สำหรับการทดสอบใช้งานฟังก์ชันขัดจังหวะ (Interrupt) ในรูปที่ 4.40 Serial Plotter แสดงกราฟค่าความเร่งเชิงเส้นในแต่ละแกนในขณะที่ผู้ทดลองอยู่ในสถานะ STAY ซึ่งจะมีความเร่งเชิงเส้นกระทำต่อแกน Z สูงสุด เมื่อผู้ทดลองผงกศีรษะ 1 ครั้งเพื่อให้เกิด Tap พบว่า Serial Plotter ในรูปที่ 4.41 แสดงผลค่าความเร่งเชิงเส้นที่กระทำต่อแกน Z และแกน Y มีค่าลดลงอย่างรวดเร็วแล้วกลับมาสู่ในสถานะค่าเดิม อันเป็นการเกิดเหตุการณ์ Tap โดยในรูปที่ 4.42 Serial Monitor ที่ส่วนควบคุมทิศทางจะแสดงสถานะ Tap และส่วนที่ควบคุมชุดกลไกขับเคลื่อนจะแสดงผลสถานะเปลี่ยนโหมดการควบคุม

รูปที่ 4.40 Serial Plotter แสดงกราฟค่าความเร่งเชิงเส้นในแต่ละแกนในสถานะ STAY



รูปที่ 4.41 Serial Plotter แสดงกราฟค่าความเร่งเชิงเส้นเมื่อเกิดเหตุการณ์ Tap

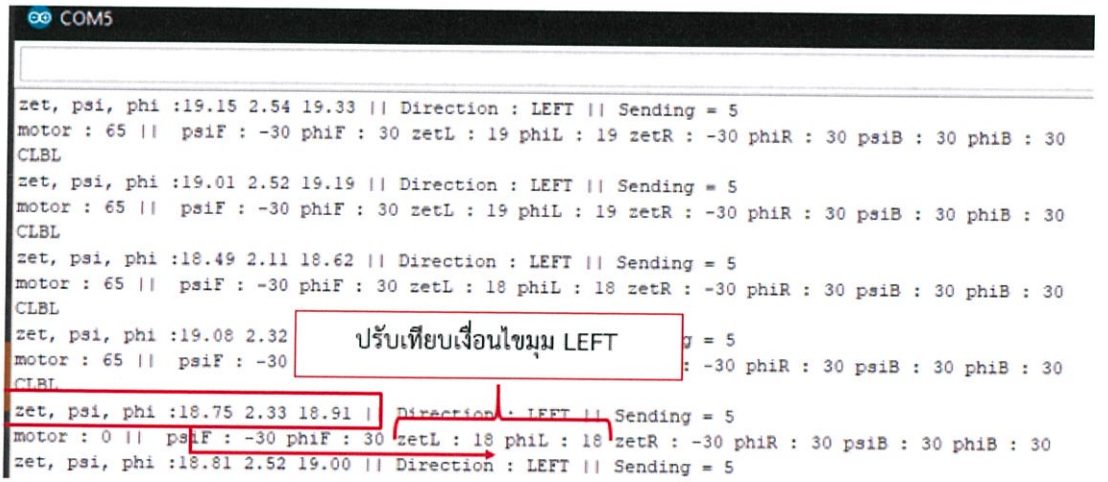


รูปที่ 4.42 Serial Monitor แสดงสถานะ CHANGE MODE (Tap)

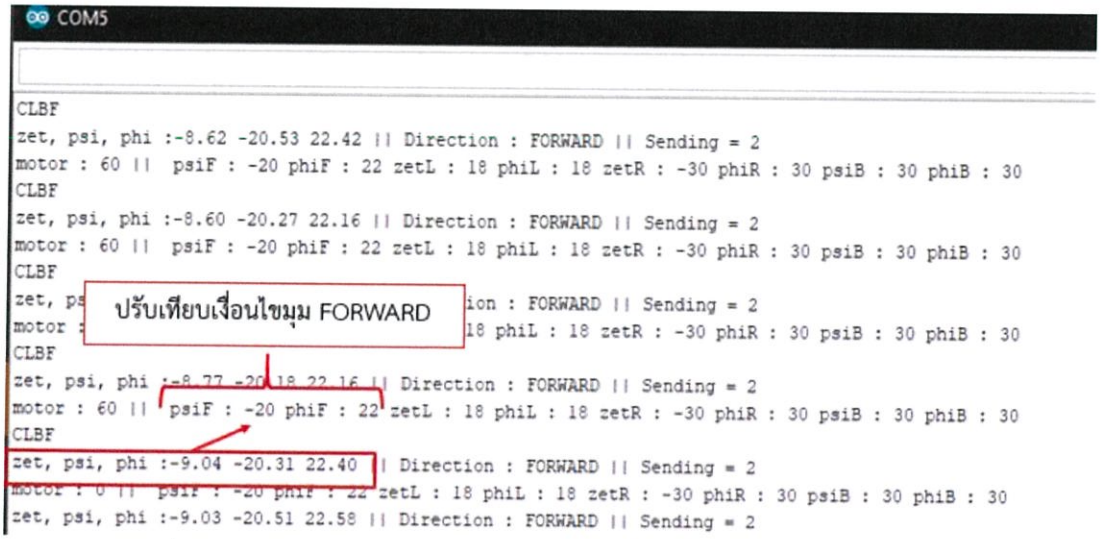
จากการทดลองควบคุมทิศทางเก้าอี้เข็นด้วยศีรษะโดยใช้เซนเซอร์ความเร่งเชิงเส้น ADXL345 พบว่า สามารถควบคุมทิศทางได้อย่างถูกต้อง และสามารถใช้งานฟังก์ชันขัดจังหวะเพื่อเปิด/ปิดการใช้ศีรษะควบคุมเก้าอี้เข็นได้ ทั้งนี้ค่าเงื่อนไขทิศทางในการทดลองส่วนนี้จะเป็นค่าเงื่อนไขเริ่มต้นของระบบ (ที่ 25 องศา) เนื่องจากผู้ใช้งานแต่ละคนมีมุมในการเอียงศีรษะที่มากน้อยแตกต่างกันไป ซึ่งส่งผลต่อการควบคุมทิศทางในการเคลื่อนที่ จึงออกแบบให้มีการปรับเทียบค่าเงื่อนไขทิศทางของบุคคล

4.3.4 การปรับเทียบค่าเงื่อนไขทิศทางของบุคคล

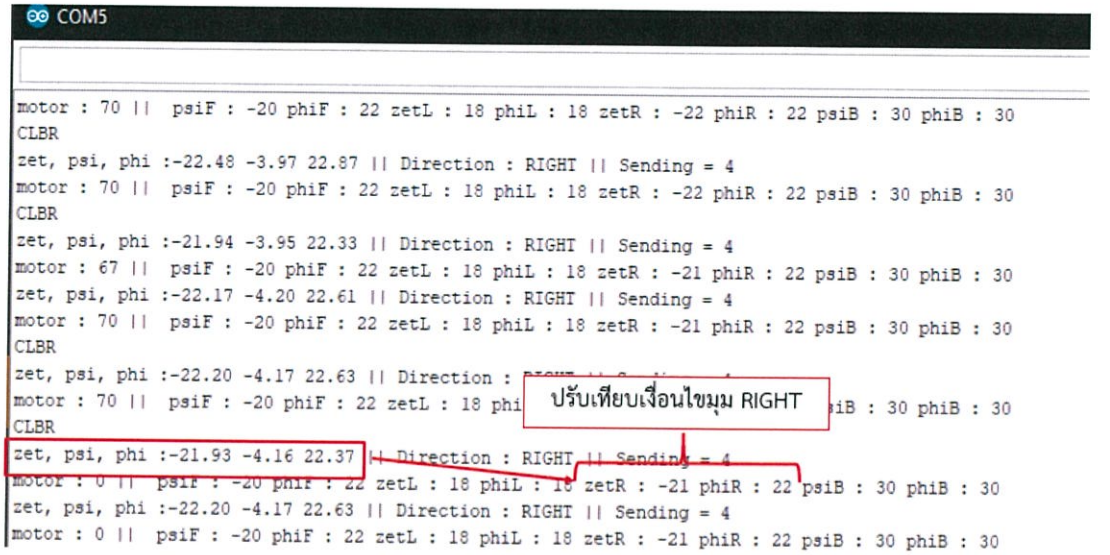
เมื่อใช้งานโหมดการปรับเทียบค่าเงื่อนไขทิศทาง ไมโครคอนโทรลเลอร์ในส่วนควบคุมกลไกขับเคลื่อนที่ติดอยู่กับเก้าอี้เข็นจะส่งค่าผ่านบลูทูธ เพื่อให้ไมโครคอนโทรลเลอร์ในส่วนควบคุมบนศีรษะเข้าสู่โหมดทำการปรับเทียบ ในรูปที่ 4.43 ถึงรูปที่ 4.48 Serial Monitor แสดงผลไมโครคอนโทรลเลอร์ในส่วนควบคุมบนศีรษะได้รับค่าให้ปรับเทียบใน 4 ทิศทาง ได้แก่ เคลื่อนที่ไปข้างหน้า, ถอยหลัง, เลี้ยวซ้าย และเลี้ยวขวา ให้ผู้ทดสอบจะเอียงศีรษะไปตามทิศทางโดยเอียงมากน้อยตามความต้องการ



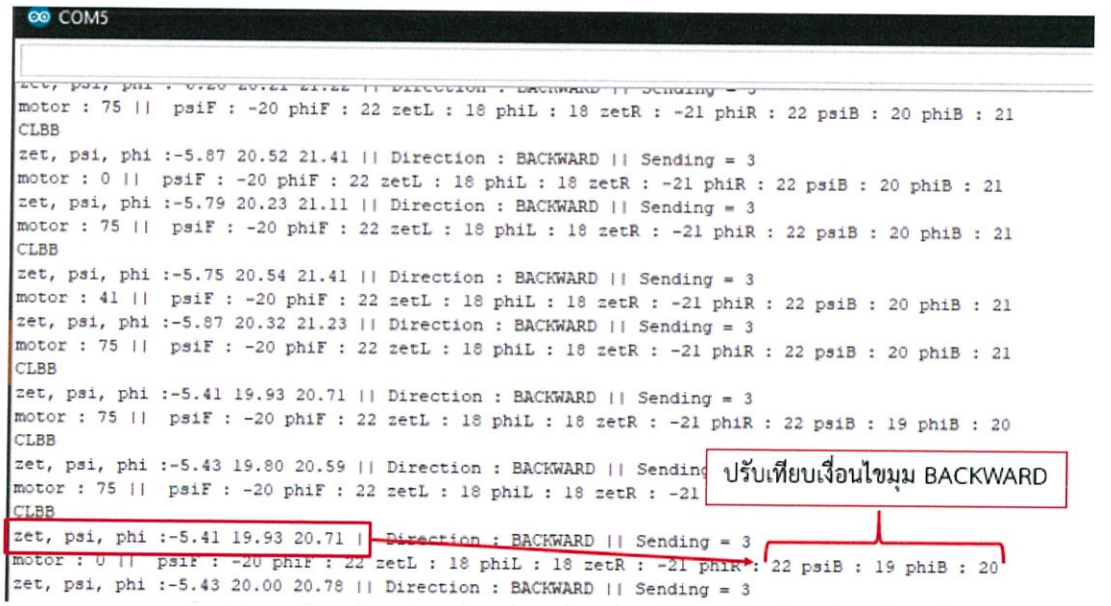
รูปที่ 4.43 การปรับเทียบค่าเงื่อนไขทิศทางเลี้ยวซ้าย (LEFT)



รูปที่ 4.44 การปรับเทียบค่าเงื่อนไขทิศทางเคลื่อนที่ไปข้างหน้า (FORWARD)

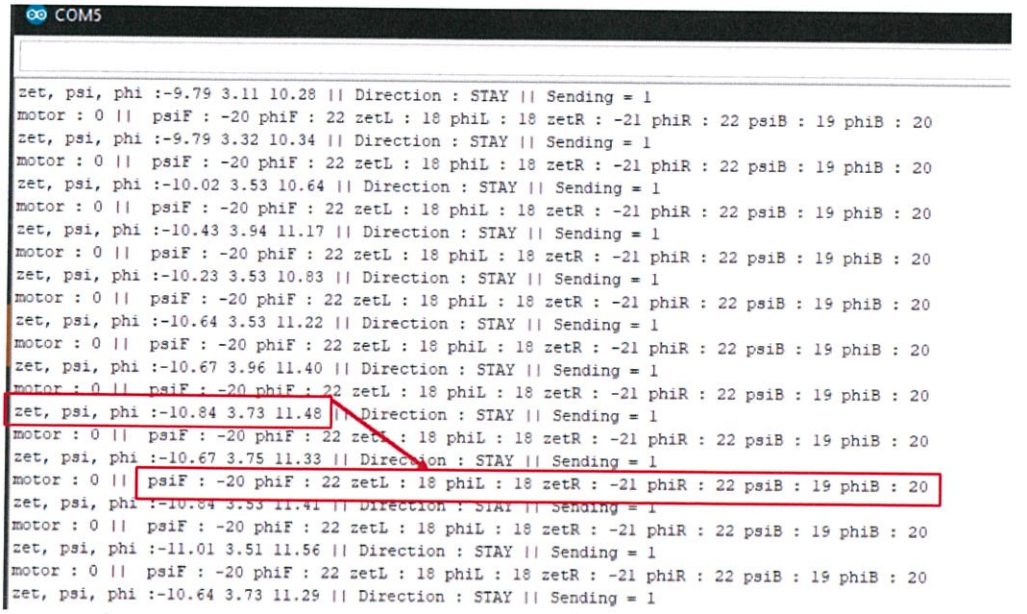


รูปที่ 4.45 การปรับเทียบค่าเงื่อนไขทิศทางเลี้ยวขวา (RIGHT)



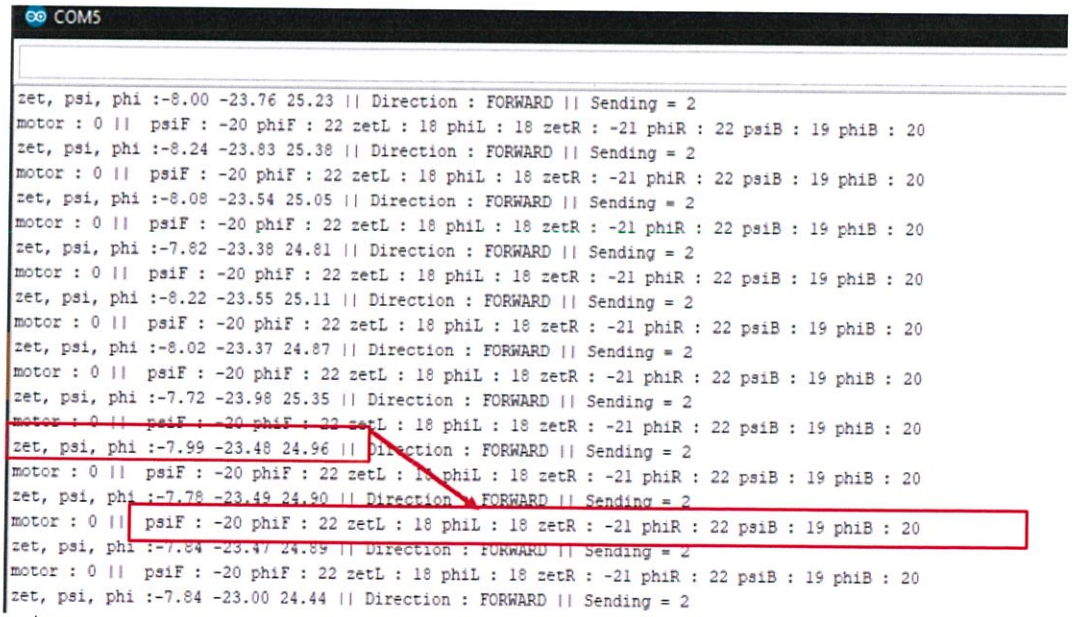
รูปที่ 4.46 การปรับเทียบค่าเงื่อนไขทิศทางถอยหลัง (BACKWARD)

หลังจากการเปรียบเทียบค่าเงื่อนไขทิศทาง ผู้ทดลองทดสอบใช้การเอียงศีรษะควบคุมทิศทางโดยใช้ค่าเงื่อนไขมุมเอียงที่ปรับเทียบนี้ สังเกตผลทิศทาง Serial Monitor โดยเมื่อผู้ทดลองไม่ได้มีการเอียงศีรษะไปในทิศทางใดๆ Serial Monitor แสดงผลสถานะหยุดนิ่ง (STAY) ดังรูปที่ 4.47 จะเห็นได้ว่ามีมุมการเอียงของศีรษะเป็นไปตามเงื่อนไขหยุดนิ่งที่ปรับเทียบไว้



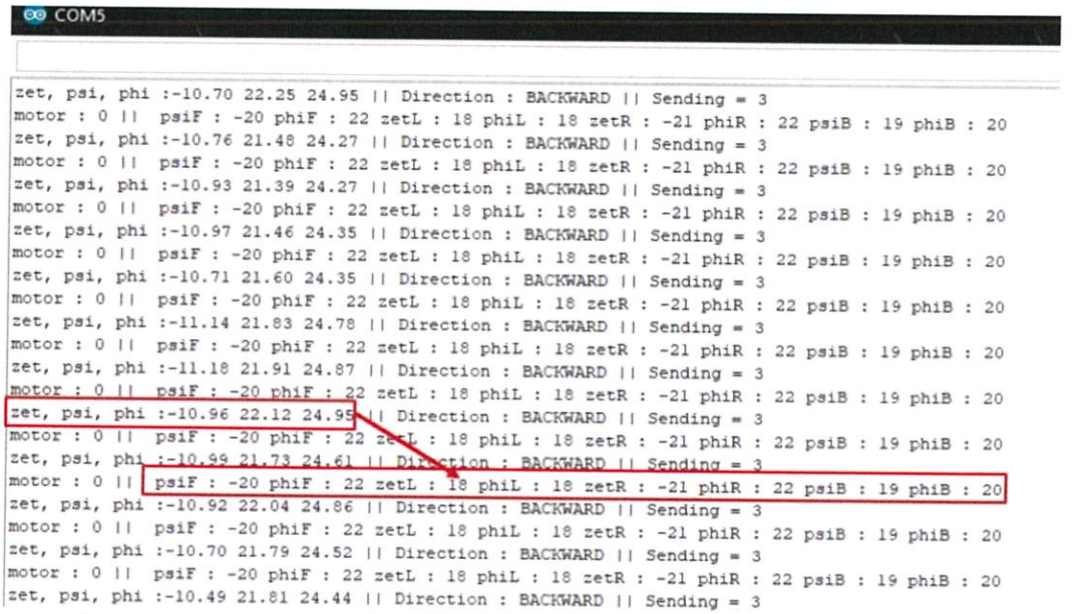
รูปที่ 4.47 Serial Monitor แสดงสถานะหยุดนิ่ง (STAY) ตามเงื่อนไขที่ปรับเทียบ

เมื่อผู้ทดลองเอียงศีรษะโน้มก้มลงไปข้างหน้า Serial Monitor แสดงผลสถานะเคลื่อนที่ไปข้างหน้า (FORWARD) ดังรูปที่ 4.48 จะเห็นได้ว่ามีมุมการเอียงของศีรษะเป็นไปตามเงื่อนไขทิศทางเคลื่อนที่ไปข้างหน้าที่ปรับเทียบไว้



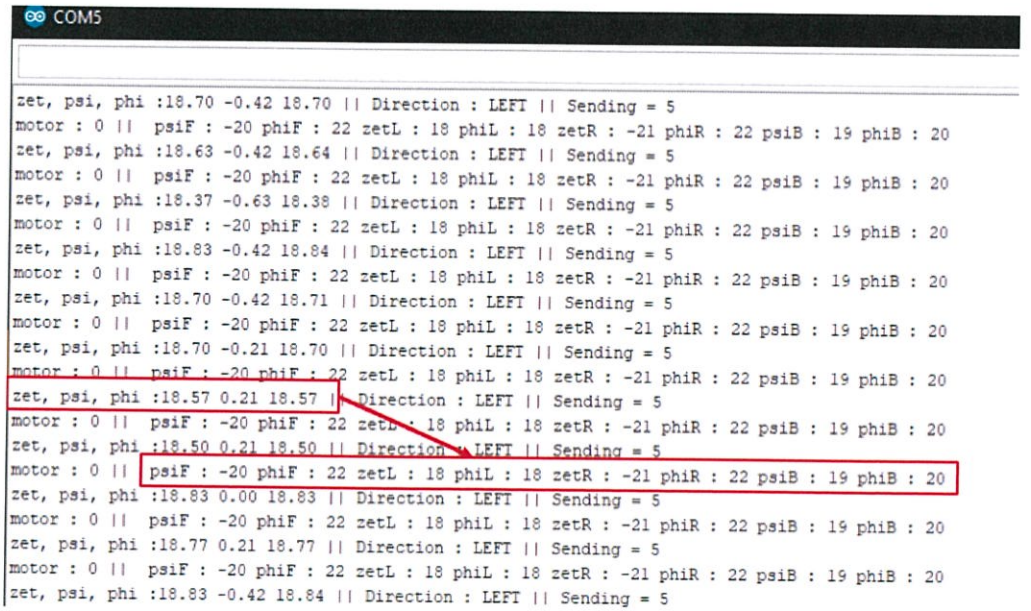
รูปที่ 4.48 Serial Monitor แสดงสถานะเคลื่อนที่ไปข้างหน้า (FORWARD) ตามเงื่อนไขที่ปรับเทียบ

เมื่อผู้ทดลองเงยศีรษะขึ้นไปข้างหลัง Serial Monitor แสดงผลสถานะถอยหลัง (BACKWARD) ดังรูปที่ 4.49 จะเห็นได้ว่ามีมุมการเอียงของศีรษะเป็นไปตามเงื่อนไขทิศทางถอยหลังที่ปรับเทียบไว้



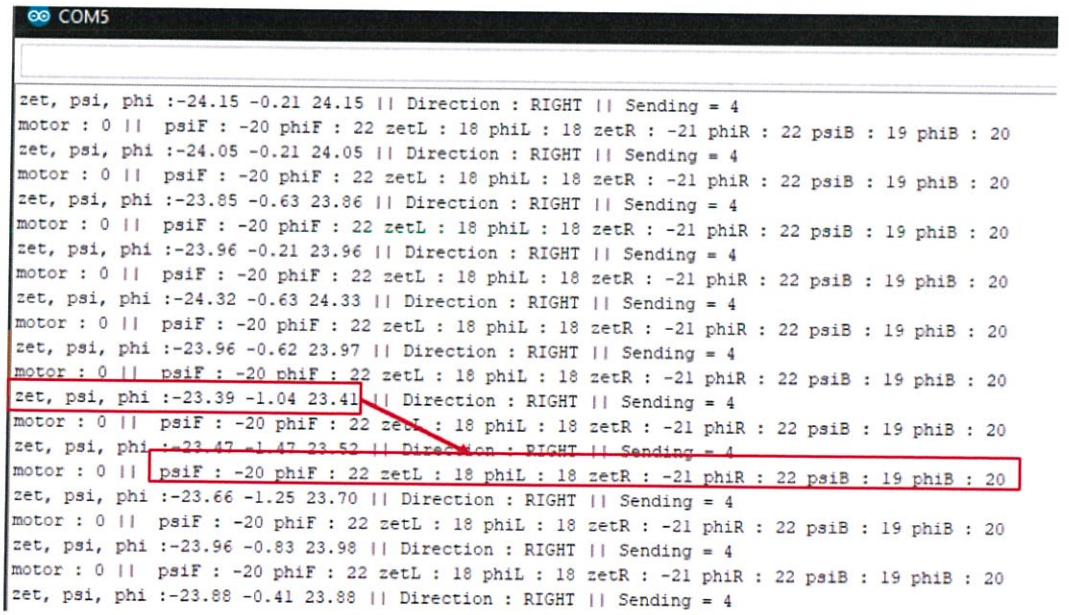
รูปที่ 4.49 Serial Monitor แสดงสถานะถอยหลัง (BACKWARD) ตามเงื่อนไขที่ปรับเทียบ

เมื่อผู้ทดลองเอียงศีรษะไปทางด้านซ้าย Serial Monitor แสดงผลสถานะเลี้ยวซ้าย (LEFT) ดังรูปที่ 4.50 จะเห็นได้ว่ามีมุมการเอียงของศีรษะเป็นไปตามเงื่อนไขทิศทางเลี้ยวซ้ายที่ปรับเทียบไว้



รูปที่ 4.50 Serial Monitor แสดงสถานะเลี้ยวซ้าย (LEFT) ตามเงื่อนไขที่ปรับเทียบ

เมื่อผู้ทดลองเอียงศีรษะไปทางด้านขวา Serial Monitor แสดงผลสถานะเลี้ยวขวา (RIGHT) ดังรูปที่ 4.51 จะเห็นได้ว่ามีมุมการเอียงของศีรษะเป็นไปตามเงื่อนไขทิศทางเลี้ยวขวาที่ปรับเทียบไว้

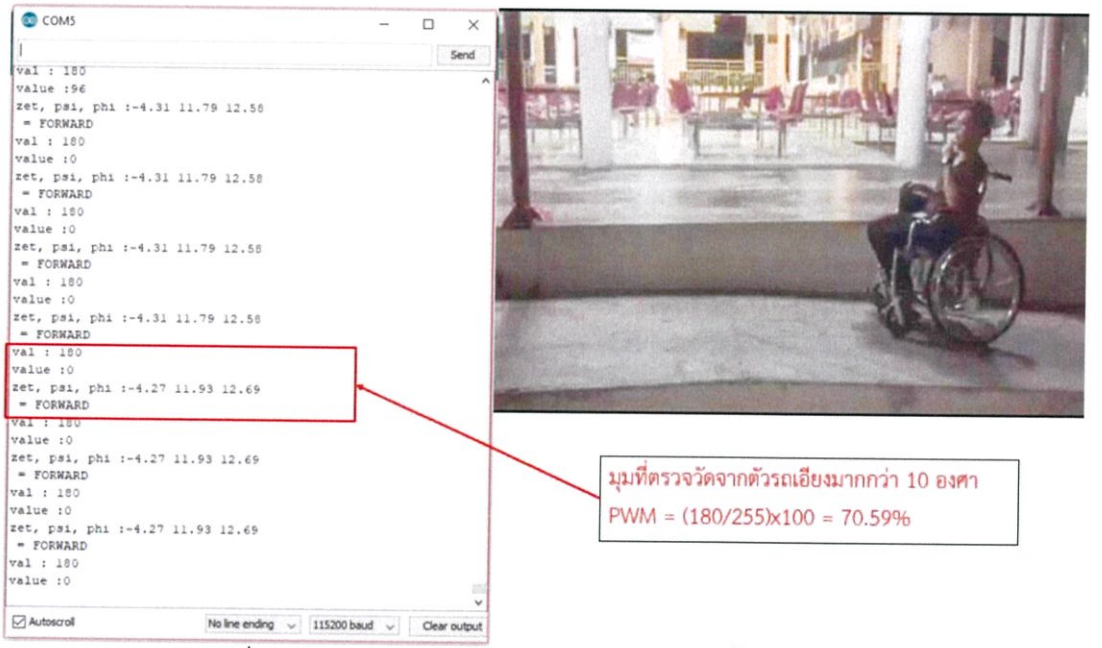


รูปที่ 4.51 Serial Monitor แสดงสถานะเลี้ยวขวา (RIGHT) ตามเงื่อนไขที่ปรับเทียบ

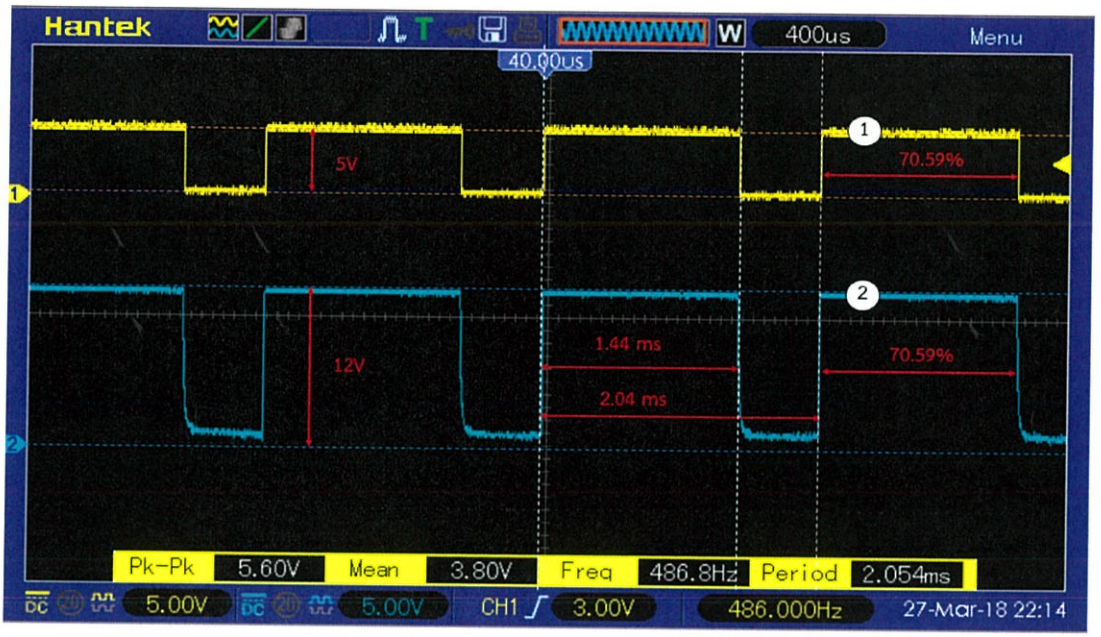
4.3.5 การทดสอบเคลื่อนที่ขึ้นและเคลื่อนที่ลงทางลาดชัน

4.3.5.1 การทดสอบเคลื่อนขึ้นทางลาดชัน

ในรูปที่ 4.52 เป็นการทดสอบการเคลื่อนขึ้นทางลาดชัน Serial Monitor แสดงผล val เท่ากับ 180 ซึ่งเป็นการกำหนดความกว้างพัลส์สำหรับวงจร H-Bridge โดยเป็นการเพิ่มความกว้างพัลส์ของสัญญาณ PWM ทำให้มอเตอร์ไฟฟ้าหมุนด้วยความเร็วรอบที่สูงขึ้นจนมีความเร็วเพียงพอที่จะเคลื่อนที่ขึ้นทางลาดได้ จะเห็นได้ว่ามีการแสดงมุมเอียงในแนวแกน Y ที่ตรวจวัดได้จากเซนเซอร์ความเร่งเชิงเส้นมากกว่า 10 องศา ในรูปที่ 4.53 แสดงสัญญาณเอาต์พุต PWM ขณะขึ้นทางลาดเอียง โดยสัญญาณหมายเลข 1 เป็นสัญญาณอินพุต และสัญญาณหมายเลข 2 สัญญาณเอาต์พุตของวงจร H-Bridge



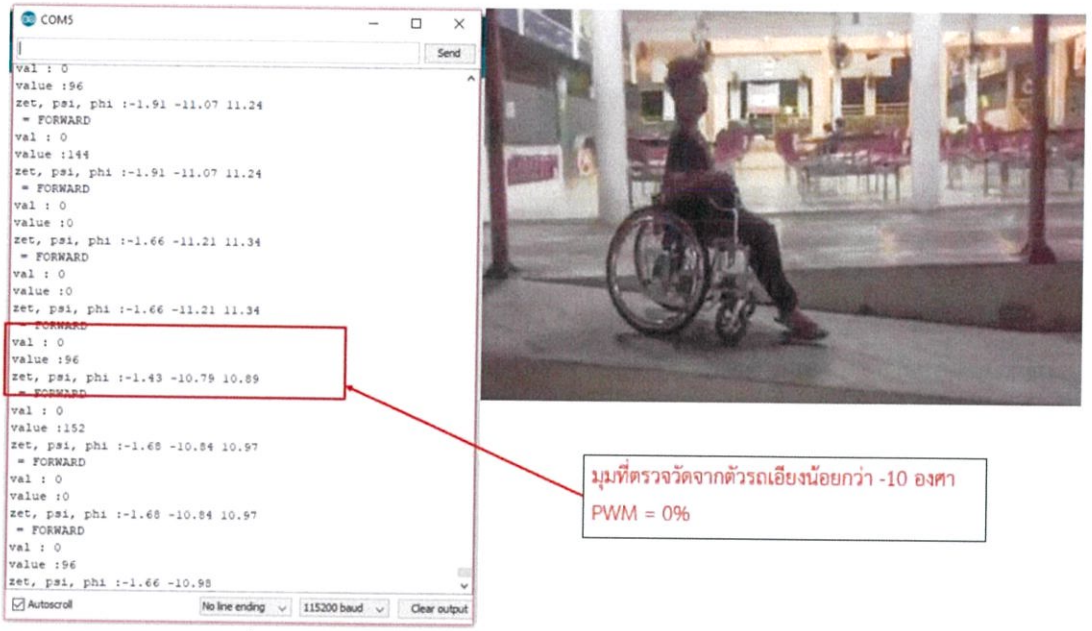
รูปที่ 4.52 Serial Monitor แสดงผลการเคลื่อนขึ้นทางลาดชัน



รูปที่ 4.53 สัญญาณอินพุตและเอาต์พุต PWM ของวงจร H-Bridge ขณะขึ้นทางลาดเอียง

4.3.5.2 การทดสอบเคลื่อนลงทางลาดชัน

ในรูปที่ 4.54 เป็นการทดสอบการเคลื่อนขึ้นทางลาดชัน Serial Monitor แสดงผล val เท่ากับ 0 ซึ่งเป็นการกำหนดความกว้างพัลส์สำหรับวงจร H-Bridge โดยเป็นการหยุดป้อนสัญญาณ PWM ทำให้มอเตอร์ไฟฟ้าหยุดหมุนเพื่อชะลอความเร็วของเก้าอี้เข็นให้สามารถเคลื่อนลงทางลาดได้อย่างนุ่มนวล โดยจะเห็นได้ว่ามีการแสดงมุมเอียงในแนวแกน Y ที่ตรวจวัดได้จากเซนเซอร์ความเร่งเชิงเส้นน้อยกว่า -10 องศา

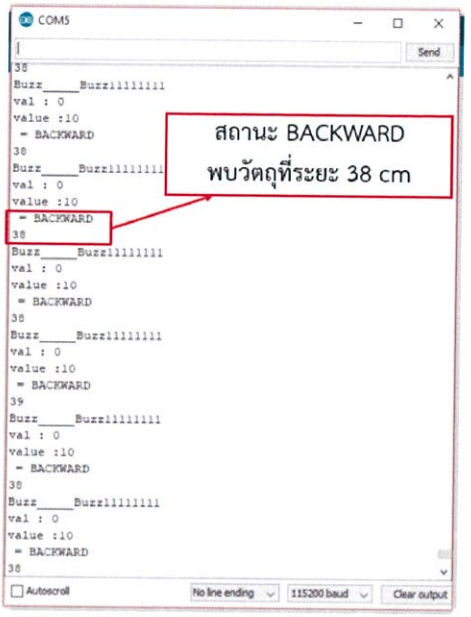


รูปที่ 4.54 Serial Monitor แสดงผลการเคลื่อนลงทางลาดชัน

4.4 การทดสอบระบบแจ้งเตือนการชน

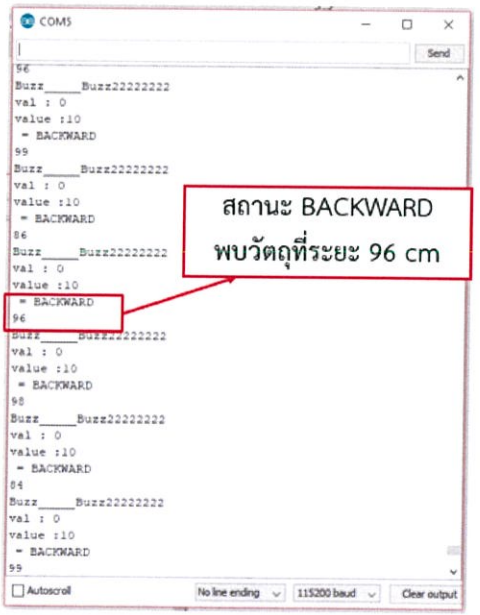
ระบบแจ้งเตือนการชนนี้จะทำงานเมื่อมีการควบคุมทิศทางการเคลื่อนที่ให้ถอยหลัง เพื่อจะส่งสัญญาณเสียงแจ้งเตือนให้ผู้ใช้งานทราบ โดยทำการทดสอบการตรวจวัดระยะห่างของวัตถุ และการแจ้งเตือนผู้ใช้งานเมื่อวัตถุอยู่ด้านหลังของเก้าอี้เข็นเป็นระยะน้อยกว่า 50 เซนติเมตร, ระยะตั้งแต่ 50 เซนติเมตร ถึง 100 เซนติเมตร และระยะมากกว่า 100 เซนติเมตร สังเกตผลบน Serial Monitor

ในรูปที่ 4.55 เมื่อวัตถุอยู่ห่างจากด้านหลังเก้าอี้เข็นเป็นระยะน้อยกว่า 50 เซนติเมตร Serial Monitor แสดงผลทิศทางถอยหลังและการตรวจพบวัตถุที่ระยะ 38 เซนติเมตร



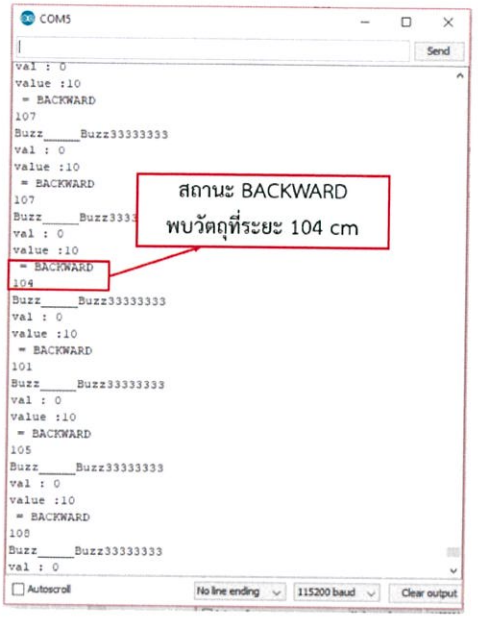
รูปที่ 4.55 Serial Monitor แสดงผลการตรวจพบวัตถุที่ระยะน้อย 50 เซนติเมตร

ในรูปที่ 4.56 เมื่อวัตถุอยู่ห่างจากด้านหลังเก้าอี้เป็นระยะตั้งแต่ 50 เซนติเมตร ถึง 100 เซนติเมตร Serial Monitor แสดงผลทิศทางถอยหลังและการตรวจพบวัตถุที่ระยะ 96 เซนติเมตร



รูปที่ 4.56 Serial Monitor แสดงผลการตรวจพบวัตถุที่ระยะตั้งแต่ 50 เซนติเมตร ถึง 100 เซนติเมตร

ในรูปที่ 4.57 เมื่อวัตถุอยู่ห่างจากด้านหลังเก้าอี้เข็นเป็นระยะมากกว่า 100 เซนติเมตร Serial Monitor แสดงผลทิศทางถอยหลังและการตรวจพบวัตถุที่ระยะ 104 เซนติเมตร

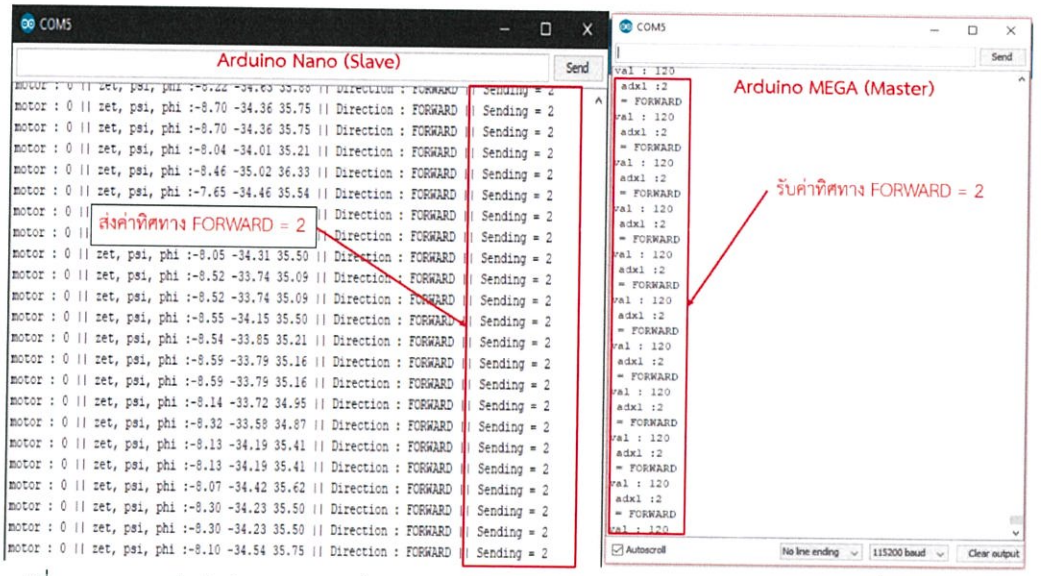


รูปที่ 4.57 Serial Monitor แสดงผลการตรวจพบวัตถุที่ระยะมากกว่า 100 เซนติเมตร

4.5 การเชื่อมต่อสื่อสารในระบบเก้าอี้เข็นควบคุมได้

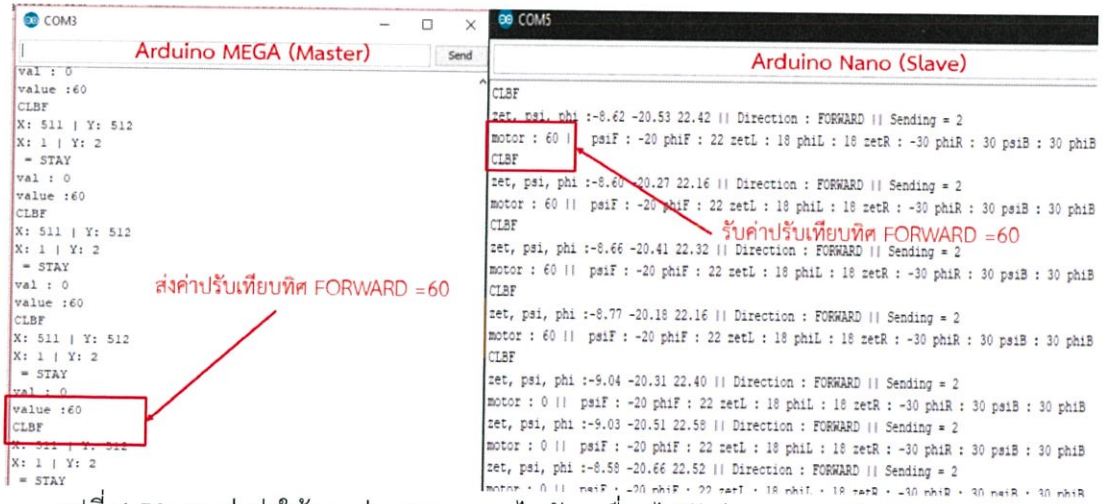
4.5.1 การเชื่อมต่อสื่อสารระหว่างส่วนควบคุมทิศทางบนศีรษะและส่วนควบคุมชุดกลไกขับเคลื่อนเก้าอี้เข็น

ในการสื่อสารกันผ่านโมดูลสื่อสารไร้สายบลูทูธระหว่างส่วนควบคุมทิศทางบนศีรษะ (Slave Mode) และส่วนควบคุมชุดกลไกขับเคลื่อนที่เก้าอี้เข็น (Master Mode) จะมีการส่งค่าทิศทางจากส่วนควบคุมทิศทาง ดังรูปที่ 4.58 Serial Monitor แสดงการส่งค่าทิศทางของเก้าอี้เข็นในสถานะเคลื่อนที่ไปข้างหน้า (FORWARD) ซึ่งกำหนดให้เป็นค่าเท่ากับ 2 โดยที่ส่วนควบคุมชุดกลไกขับเคลื่อนจะได้รับค่า 2 เช่นกัน



รูปที่ 4.58 การส่งค่าทิศทางจากส่วนควบคุมทิศทางไปยังส่วนควบคุมกลไกขับเคลื่อนผ่านบลูทูธ

นอกจากการส่งค่าทิศทางแล้ว ส่วนควบคุมกลไกขับเคลื่อนที่เก้าอี้เข็นจะมีการส่งค่าเพื่อให้ส่วนควบคุมทิศทางบนศีรษะทำการปรับเทียบเงื่อนไขทิศทาง ดังรูปที่ 4.59 Serial Monitor แสดงการส่งค่าให้ปรับเทียบในทิศทางเคลื่อนที่ไปข้างหน้า ซึ่งกำหนดให้เป็นค่าเท่ากับ 60 โดยที่ส่วนควบคุมทิศทางบนศีรษะจะได้รับค่า 60 เช่นกัน



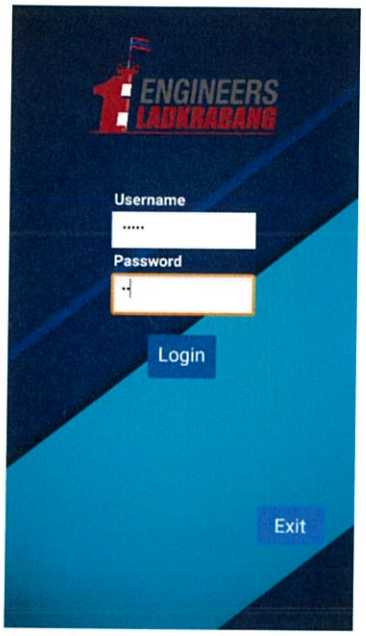
รูปที่ 4.59 การส่งค่าให้จากส่วนควบคุมกลไกขับเคลื่อนไปยังส่วนควบคุมทิศทางผ่านบลูทูธ

4.5.2 การใช้งานแอปพลิเคชันและการสื่อสารผ่าน Web Server บน NodeMCU

เมื่อทำการติดตั้งแอปพลิเคชันบนสมาร์ทโฟนจะปรากฏไอคอนของแอปพลิเคชันที่ชื่อว่า “testsendudp” ดังรูปที่ 4.60

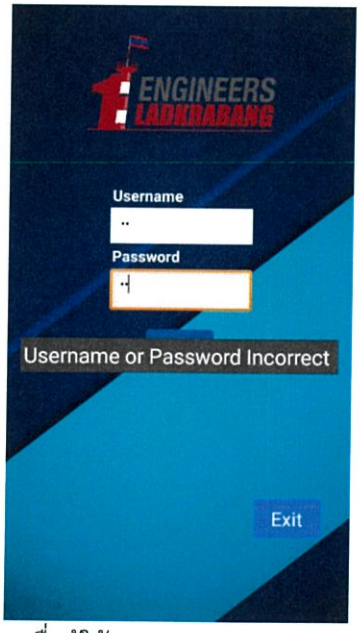
รูปที่ 4.60 แอปพลิเคชันที่ถูกติดตั้งบนสมาร์ตโฟน

เมื่อทำการเปิดแอปพลิเคชันขึ้นมาจะพบหน้า Screen1 ซึ่งเป็นหน้าการใช้งานสำหรับการยืนยันตัวตนเพื่อความปลอดภัยในการใช้งานเก้าอี้เข็นดังรูปที่ 4.61 ผู้ใช้จะต้องทำการกรอก Username และ Password เป็นคำว่า “kmitl” และ “4a” ตามลำดับ เมื่อทำการกดปุ่ม Login แอปพลิเคชันจะเปิดหน้า Screen2 ขึ้นมาซึ่งเป็นหน้าจอสำหรับการควบคุมเก้าอี้เข็น



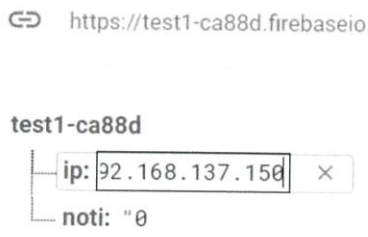
รูปที่ 4.61 หน้าScreen2สำหรับการกรอก Username และ Password

หากข้อมูลที่ผู้ใช้งานแอปพลิเคชันกรอกนั้นไม่ตรงกับ Username และ Password ที่ถูกต้อง จะแสดงข้อความแจ้งเตือนเป็นข้อความ Text ว่า “Username or Password Incorrect” ดังรูปที่ 4.62 และจะไม่สามารถเปิดหน้า Screen2 ที่ใช้ควบคุมเก้าอี้เข็นได้ ผู้ใช้งานจะต้องกรอก Username และ Password ใหม่ให้ถูกต้อง จึงจะสามารถเข้าใช้งานแอปพลิเคชันได้



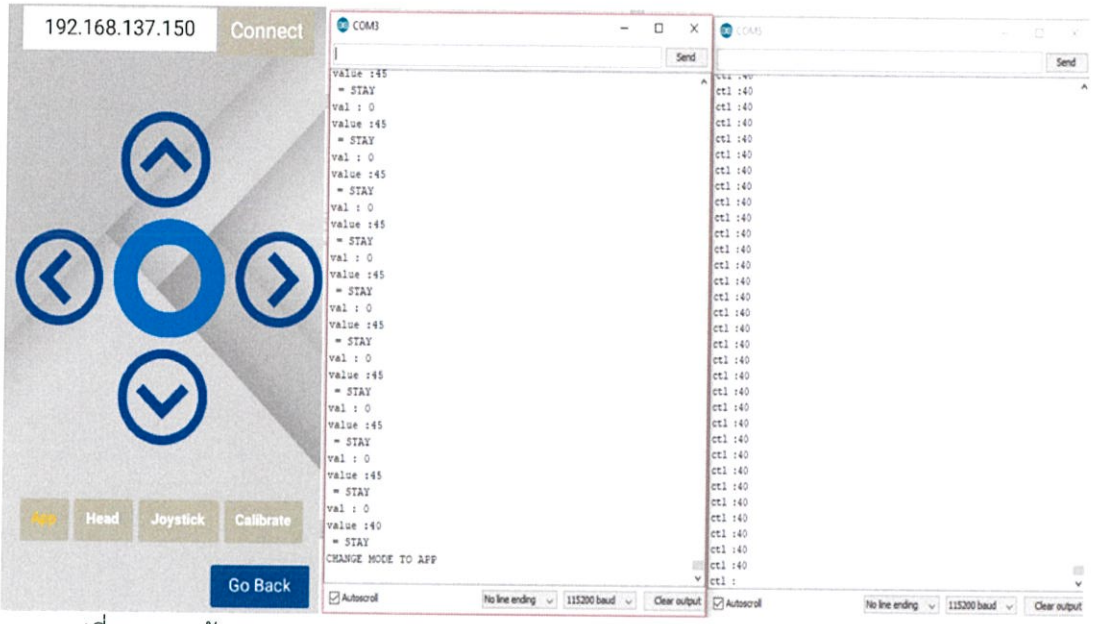
รูปที่ 4.62 ข้อความแจ้งเตือนเมื่อผู้ใช้งานกรอก Username หรือ Password ผิดพลาด

เมื่อทำการเปิดหน้า Screen2 ซึ่งเป็นหน้าจอหลักที่ใช้ในการควบคุมทิศทางรวมไปถึงการปรับโหมดการควบคุมทิศทางของเก้าอี้เข็น ผู้ใช้จะต้องทำการกดปุ่ม Connect เพื่อรับค่า IP จาก FirebaseDB ซึ่งเป็นค่าที่ NodeMCU ได้ส่งมายัง FirebaseDB จากรูปที่ 4.63 จะพบว่าเลข IP ที่ NodeMCU ส่งมาคือ “192.168.137.150”



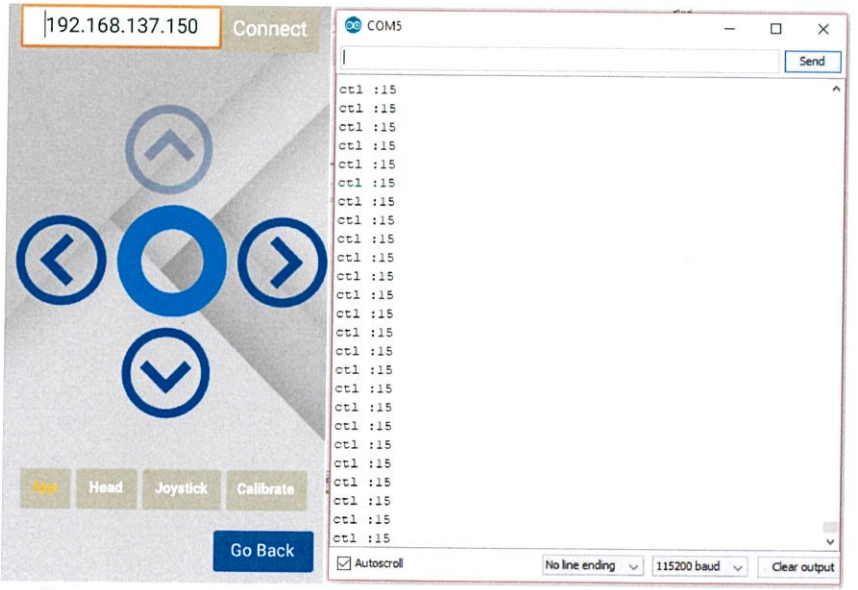
รูปที่ 4.63 ที่อยู่ IP ที่ส่งมาจาก NodeMCU

เมื่อทำการกดปุ่ม Connect แล้วจะปรากฏเลข IP ที่ใช้ในการเชื่อมต่อกับเก้าอี้เข็นคือ “192.168.137.150” ซึ่งบ่งบอกว่าแอปพลิเคชันจะทำการส่งค่าตัวแปร ctl ค่าต่างๆไปยัง http://192.168.137.150 ซึ่งเป็น Web Server ของ NodeMCU เมื่อผู้ใช้งานเลือกใช้การควบคุมเก้าอี้เข็นผ่านแอปพลิเคชัน แอปพลเคชันจะส่งตัวแปร ctl=40 ไปยัง Web Server เพื่อให้เก้าอี้เข็นเข้าสู่โหมดการใช้งานผ่านแอปพลิเคชัน รูปที่ 4.64 จะประกอบไปด้วยหน้า Screen2 ในโหมดการใช้งานแอปพลิเคชัน, Serial Monitor ของ Arduino MEGA และ NodeMCU ตามลำดับ จะพบว่าค่าตัวแปร ctl ที่ NodeMCU รับได้คือ 40 ซึ่งเป็นค่าของโหมดการใช้งานผ่านแอปพลิเคชันใน Serial Monitor ของ Arduino MEGA จะปรากฏคำว่า “CHANGE MODE TO APP” ซึ่งบอกถึงการใช้งานในโหมดแอปพลิเคชัน



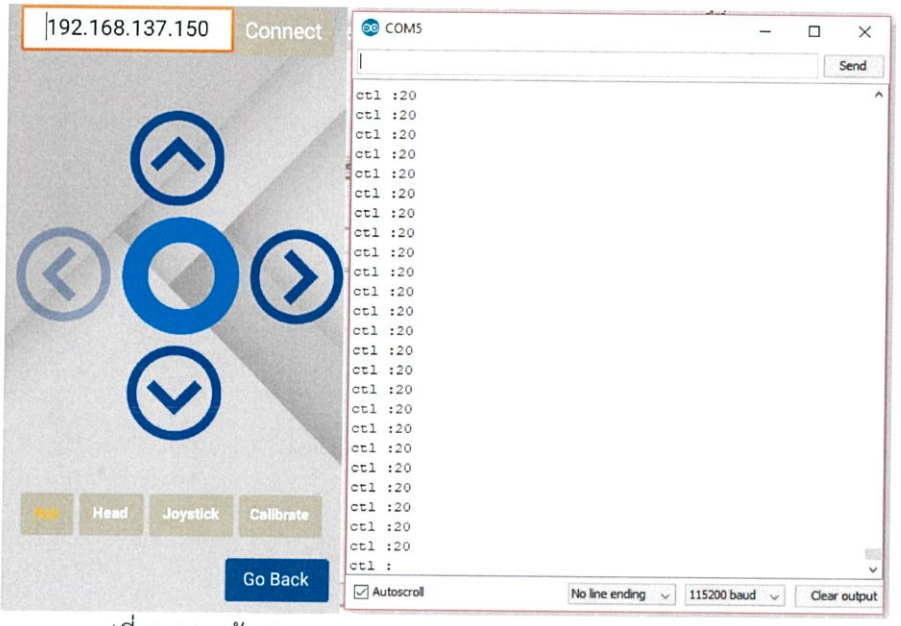
รูปที่ 4.64 หน้า Screen2 และ Serial Monitor ของ Arduino MEGA และ NodeMCU ในโหมด App

เมื่อผู้ใช้งานทำการ Touchdown ปุ่ม Forward จะพบว่าใน Serial Monitor ของ NodeMCU นั้น ค่าตัวแปร ctl ที่ NodeMCU รับได้คือ 15 ซึ่งเป็นค่าที่ใช้ควบคุมเก้าอี้เข็นให้เคลื่อนที่ไปข้างหน้า ดังรูปที่ 4.65



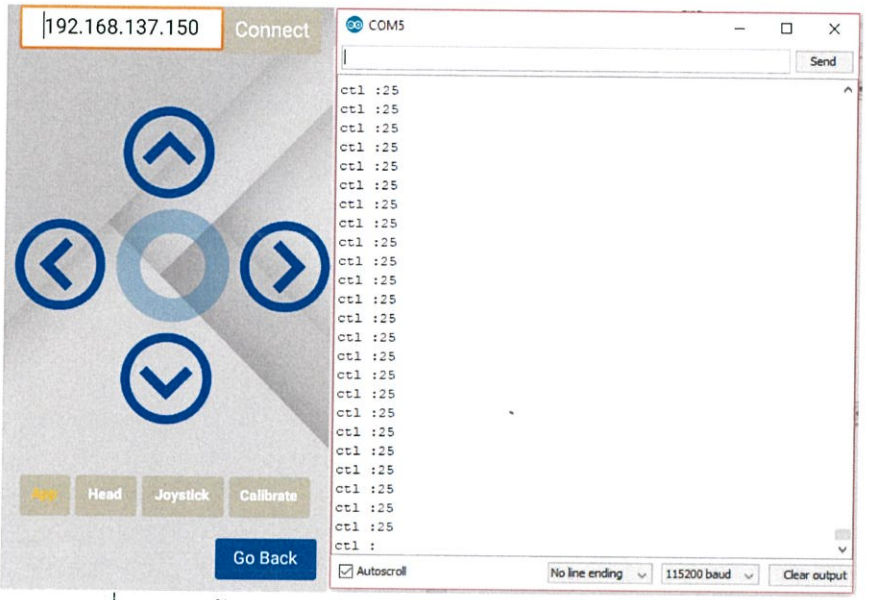
รูปที่ 4.65 หน้า Screen2 และ Serial Monitor ของ NodeMCU เมื่อทำการ Touchdown ปุ่ม Forward

เมื่อผู้ใช้งานทำการ Touchdown ปุ่ม Left จะพบว่าใน Serial Monitor ของ NodeMCU นั้น ค่าตัวแปร ctl ที่ NodeMCU รับได้คือ 20 ซึ่งเป็นค่าที่ใช้ควบคุมเก้าอี้เข็นให้เคลื่อนที่ไปทางซ้าย ดังรูปที่ 4.66



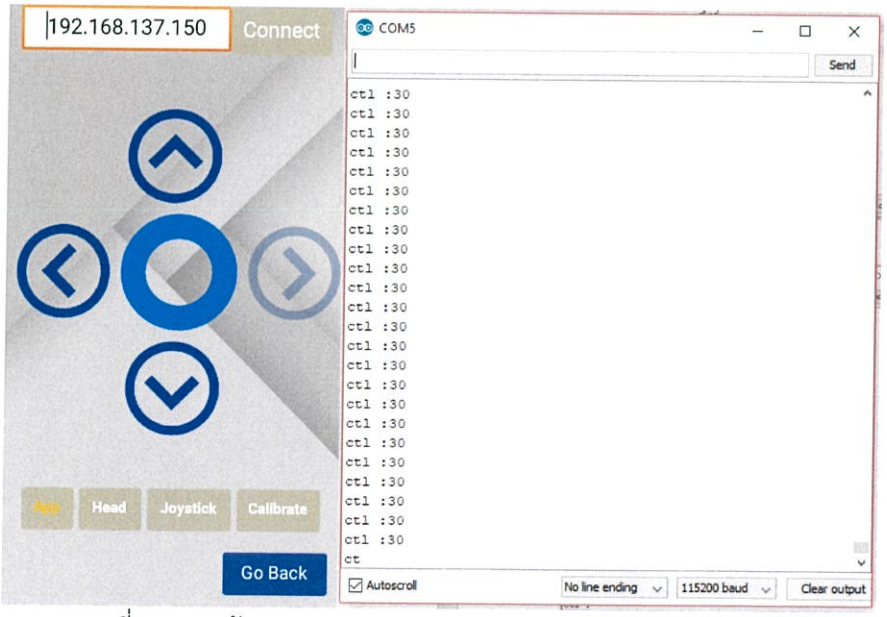
รูปที่ 4.66 หน้า Screen2 และ Serial Monitor ของ NodeMCU
เมื่อทำการ Touchdown ปุ่ม Left

เมื่อผู้ใช้งานทำการ Touchdown ปุ่ม Stop จะพบว่าใน Serial Monitor ของ NodeMCU นั้น ค่าตัวแปร ctl ที่ NodeMCU รับได้คือ 25 ซึ่งเป็นค่าที่ใช้ควบคุมเก้าอี้เข็นให้ทำการหยุด (BREAK) ดังรูปที่ 4.67



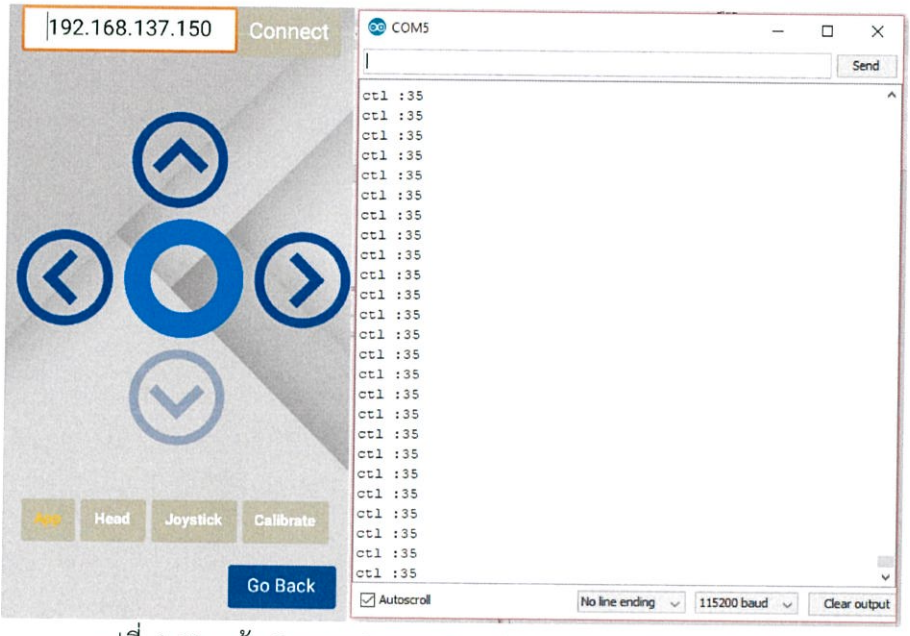
รูปที่ 4.67 หน้า Screen2 และ Serial Monitor ของ NodeMCU
เมื่อทำการ Touchdown ปุ่ม Stop

เมื่อผู้ใช้งานทำการ Touchdown ปุ่ม Right จะพบว่าใน Serial Monitor ของ NodeMCU นั้น ค่าตัวแปร ctl ที่ NodeMCU รับได้คือ 30 ซึ่งเป็นค่าที่ใช้ควบคุมเก้าอี้เข็นให้เคลื่อนที่ไปทางขวา ดังรูปที่ 4.68



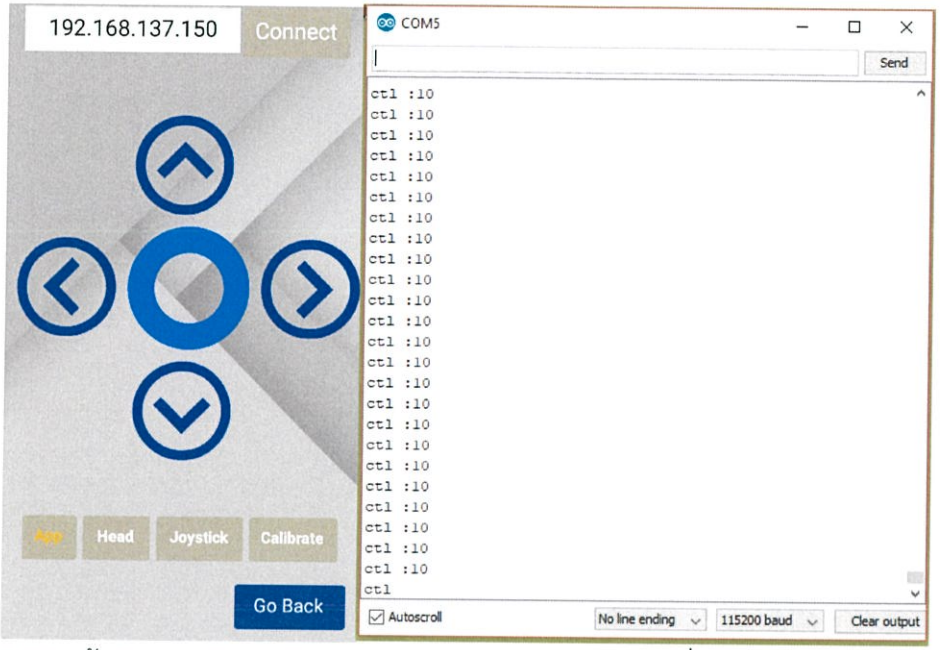
รูปที่ 4.68 หน้า Screen2 และ Serial Monitor ของ NodeMCU
เมื่อทำการ Touchdown ปุ่ม Right

เมื่อผู้ใช้งานทำการ Touchdown ปุ่ม Back พบว่าใน Serial Monitor ของ NodeMCU นั้น ค่าตัวแปร ctl ที่ NodeMCU รับได้คือ 35 ซึ่งเป็นค่าที่ใช้ควบคุมเก้าอี้เข็นให้เคลื่อนที่ไปข้างหลัง ดังรูปที่ 4.69



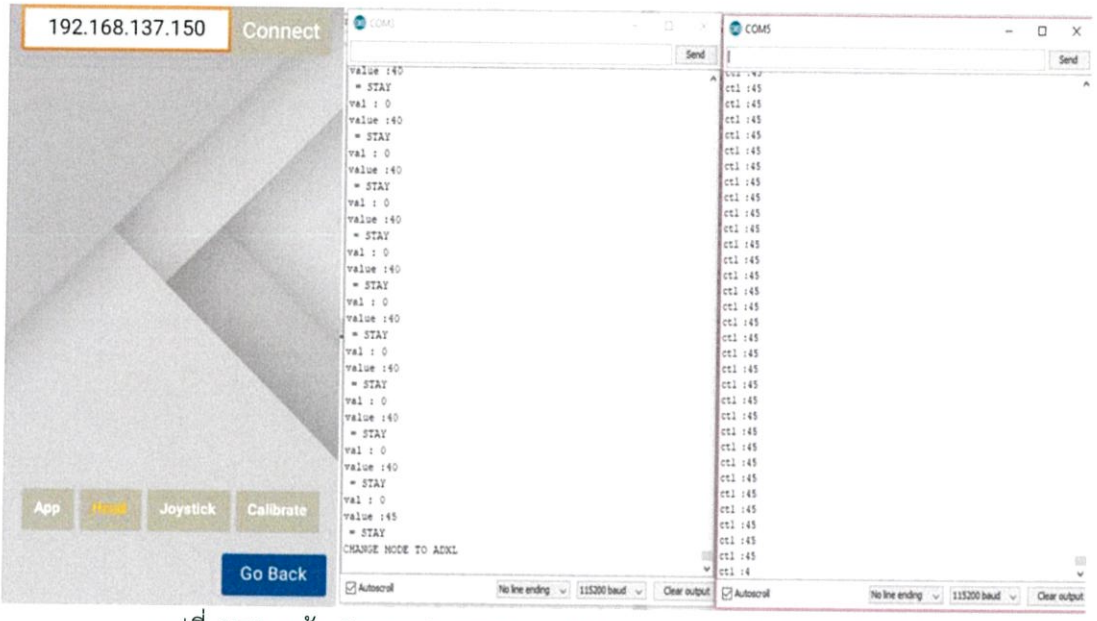
รูปที่ 4.69 หน้า Screen2 และ Serial Monitor ของ NodeMCU
เมื่อทำการ Touchdown ปุ่ม Back

เมื่อผู้ใช้งานทำการ Touchup ปุ่มใดปุ่มหนึ่งพบว่าใน Serial Monitor ของ NodeMCU นั้น ค่าตัวแปร ctl ที่ NodeMCU รับได้คือ 10 เป็นค่าที่ใช้ควบคุมเก้าอี้เข็นให้อยู่ในสถานะ Stay ดังรูปที่ 4.70



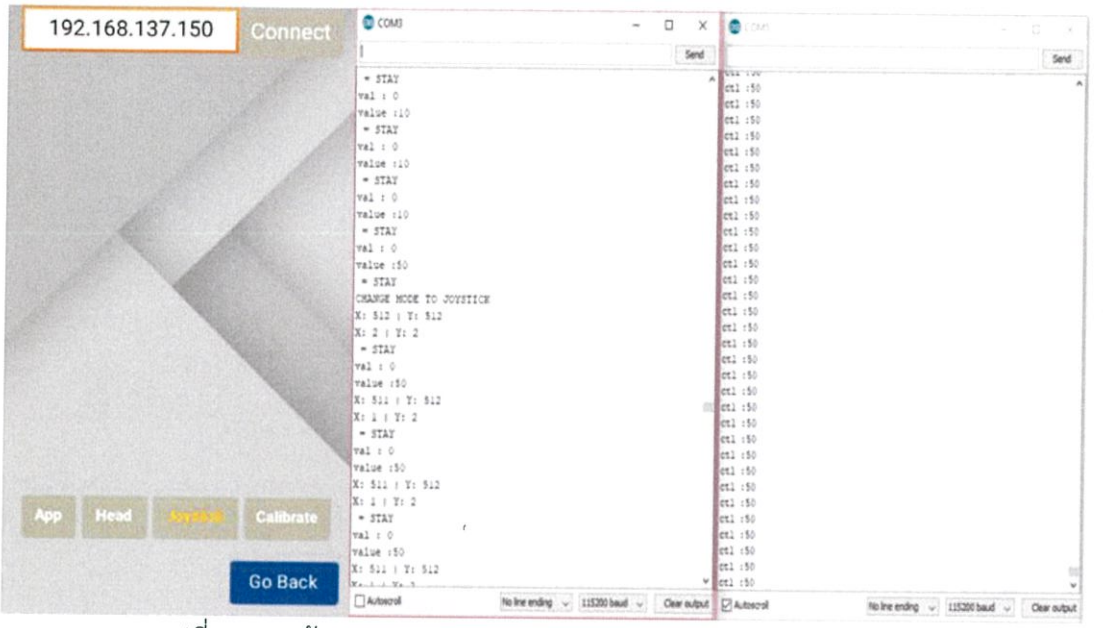
รูปที่ 4.70 หน้า Screen2 และ Serial Monitor ของ NodeMCU เมื่อทำการอยู่ในสถานะ Stay

เมื่อผู้ใช้งานแอปพลิเคชันต้องการทำการปรับเปลี่ยนโหมดการใช้งานเป็นการควบคุมเก้าอี้เข็นโดยการใช้เซ็นเซอร์วัดความเร่งเชิงเส้นที่ติดตั้งอยู่บนศีรษะของผู้ใช้งานเก้าอี้เข็น จะสามารถทำได้โดยการกดปุ่มเปลี่ยนโหมดการใช้งานที่ด้านล่างของหน้า Screen2 เมื่อทำการกดปุ่ม Head จะพบว่าปุ่มควบคุมทิศทางทั้ง 5 ปุ่มจะหายไป ทำให้ผู้ใช้งานแอปพลิเคชันไม่สามารถควบคุมเก้าอี้เข็นโดยใช้แอปพลิเคชันได้แล้ว ค่าที่ส่งไปยัง Web Server ของ NodeMCU จะมีค่าเท่ากับ 45 ซึ่งเป็นค่าที่ทำให้เก้าอี้เข็นปรับเปลี่ยนโหมดการใช้งานเป็นการควบคุมทิศทางเก้าอี้เข็นโดยใช้เซ็นเซอร์วัดความเร่งเชิงเส้นที่ติดอยู่บนศีรษะของผู้ใช้งานเก้าอี้เข็น จาก Serial Monitor ของ Arduino MEGA จะระบุว่า “CHANGE MODE TO ADXL” ซึ่งบ่งบอกว่าเก้าอี้เข็นได้ทำการปรับเปลี่ยนโหมดการควบคุมเรียบร้อยแล้ว ดังรูปที่ 4.71



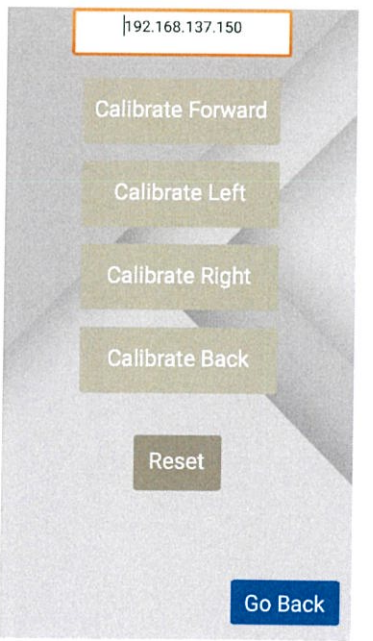
รูปที่ 4.71 หน้า Screen2 และ Serial Monitor ของ Arduino MEGA และ NodeMCU ในโหมด Head

เมื่อผู้ใช้งานแอปพลิเคชันต้องการทำการปรับเปลี่ยนโหมดการใช้งานเป็นการควบควบเก้าอี้เข็นโดยการใช้ก้านควบคุมแบบอนาล็อก จะสามารถทำได้โดยการกดปุ่มเปลี่ยนโหมดการใช้งานที่ด้านล่างของหน้า Screen2 เมื่อทำการกดปุ่ม Joystick จะพบว่าปุ่มควบคุมทิศทางทั้ง 5 ปุ่มจะหายไป ทำให้ผู้ใช้งานแอปพลิเคชันไม่สามารถควบคุมเก้าอี้เข็นโดยใช้แอปพลิเคชันได้ ค่าที่ส่งไปยัง Web Server ของ NodeMCU จะมีค่าเท่ากับ 50 ซึ่งเป็นค่าที่ทำให้เก้าอี้เข็นปรับเปลี่ยนโหมดการใช้งานเป็นการควบคุมทิศทางเก้าอี้เข็นโดยใช้ก้านควบคุมแบบอนาล็อก จาก Serial Monitor ของ Arduino MEGA จะระบุว่า “CHANGE MODE TO JOYSTICK” ซึ่งบ่งบอกว่าเก้าอี้เข็นได้ทำการปรับเปลี่ยนโหมดการควบคุมเรียบร้อยแล้ว ดังรูปที่ 4.72



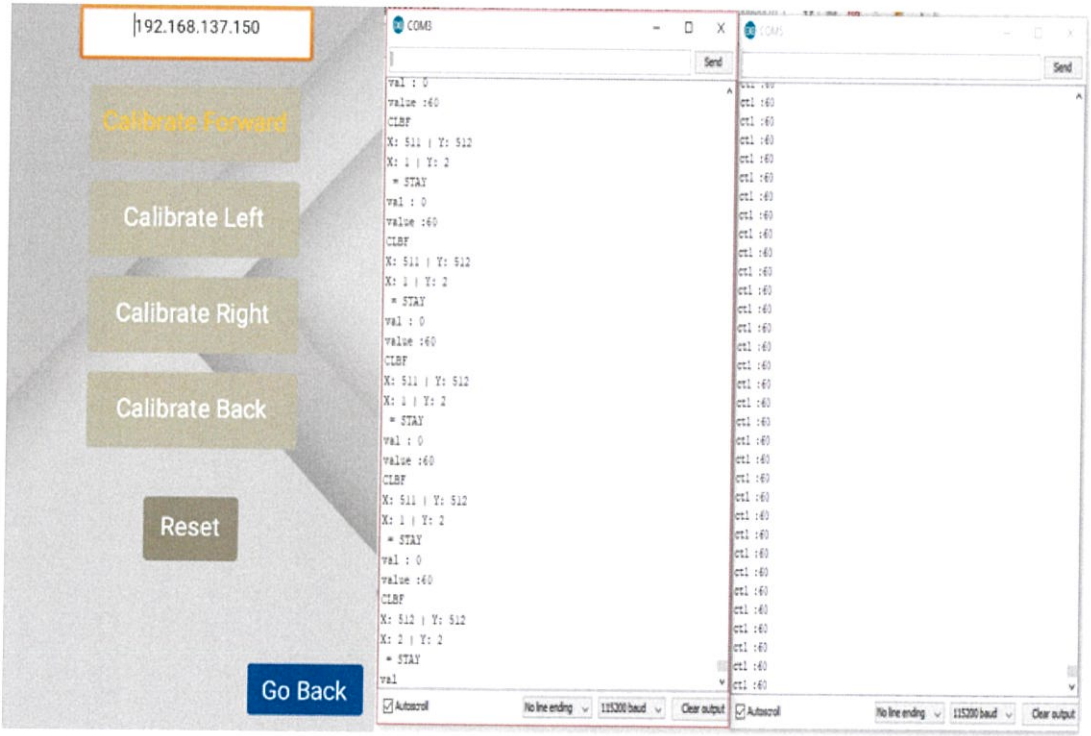
รูปที่ 4.72 หน้า Screen2 และ Serial Monitor ของ Arduino MEGA
และ NodeMCU ในโหมด Joystick

เมื่อผู้ใช้งานแอปพลิเคชันต้องการปรับเทียบมุมองศาการเอียงของศีรษะจะต้องทำการ Calibrate โดยเมื่อทำการกดปุ่ม Calibrate ในหน้า Screen2 แล้วจะปรากฏหน้า Screen4 ดังรูปที่ 4.73



รูปที่ 4.73 หน้า Screen4 สำหรับปรับเทียบมุมองศาการใช้งาน

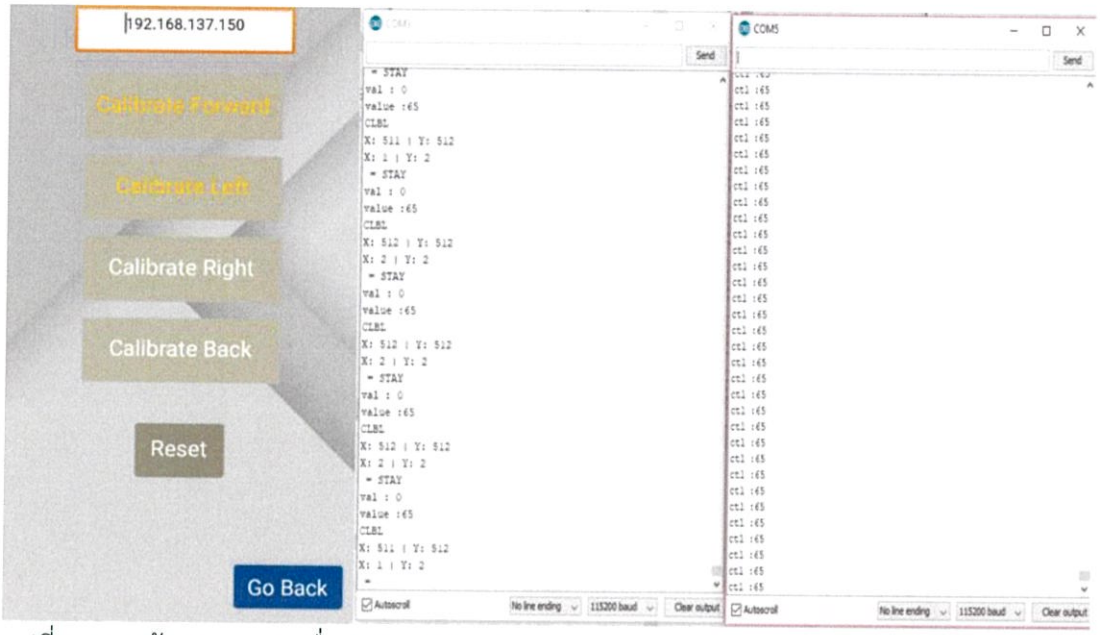
เมื่อผู้ใช้งานต้องการทำการปรับเทียบเงื่อนไขทิศทางมุมองศาการเอียงของเซนเซอร์ความเร่งเชิงเส้นให้เก้าอี้เข็นเคลื่อนที่ไปข้างหน้า ผู้ใช้งานจะต้องทำการเอียงศีรษะไปยังมุมองศาที่ต้องการใช้งานแล้วกดปุ่ม Calibrate Forward เมื่อทำการกดปุ่ม Calibrate Forward แล้ว text colour บนปุ่ม Calibrate Forward จะเปลี่ยนเป็นสีเหลืองเพื่อบอกผู้ใช้งานว่าได้ทำการปรับเทียบมุมองศาการใช้ของการเคลื่อนที่เก้าอี้เข็นไปข้างหน้าแล้ว ค่า ctl ที่ส่งไปยัง Web Server ของ NodeMCU จะเท่ากับ 60 และที่ Serial Monitor ของ Arduino MEGA จะแสดงคำว่า “CLBF” ซึ่งแสดงว่าตัวเก้าอี้เข็นได้ทำการปรับเทียบมุมองศาการเอียงศีรษะเพื่อเคลื่อนที่เก้าอี้เข็นไปด้านหน้าเรียบร้อยแล้วดังรูปที่ 4.74



รูปที่ 4.74 หน้า Screen4 เมื่อทำการ Calibrate Forward และ Serial Monitor ของ Arduino MEGA และ NodeMCU

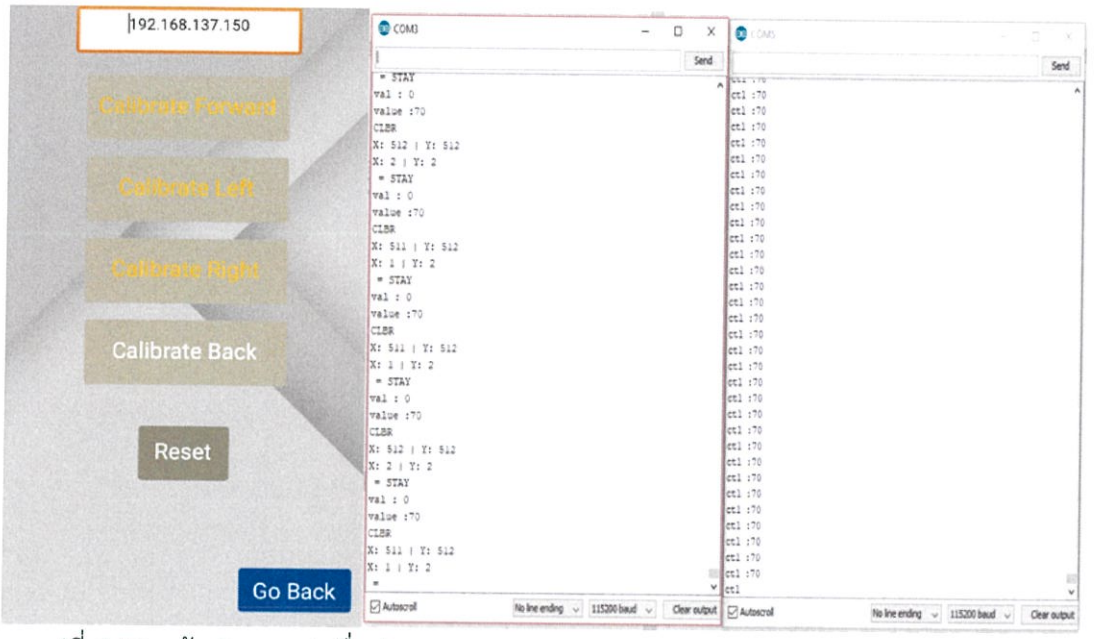
เมื่อผู้ใช้งานต้องการทำการปรับเทียบเงื่อนไขมุมองศาการเอียงของเซนเซอร์ความเร่งเชิงเส้นให้เก้าอี้เข็นเคลื่อนที่ไปทางซ้าย ผู้ใช้งานจะต้องทำการเอียงศีรษะไปยังมุมองศาที่ต้องการใช้งานแล้วกดปุ่ม Calibrate Left เมื่อทำการกดปุ่ม Calibrate Left แล้ว text colour บนปุ่ม Calibrate Left จะเปลี่ยนเป็นสีเหลืองเพื่อบอกผู้ใช้งานว่าได้ทำการปรับเทียบมุมองศาการใช้ของ

การเคลื่อนที่เก้าอี้เข็นไปทางซ้ายแล้ว ค่า ctl ที่ส่งไปยัง Web Server ของ NodeMCU จะเท่ากับ 65 และที่ Serial Monitor ของ Arduino MEGA จะแสดงคำว่า “CLBL” ซึ่งแสดงว่าตัวเก้าอี้เข็นได้ทำการปรับเทียบมุมองศาการเอียงศีรษะเพื่อเคลื่อนที่เก้าอี้เข็นไปด้านซ้ายเรียบร้อยแล้วดังรูปที่ 4.75



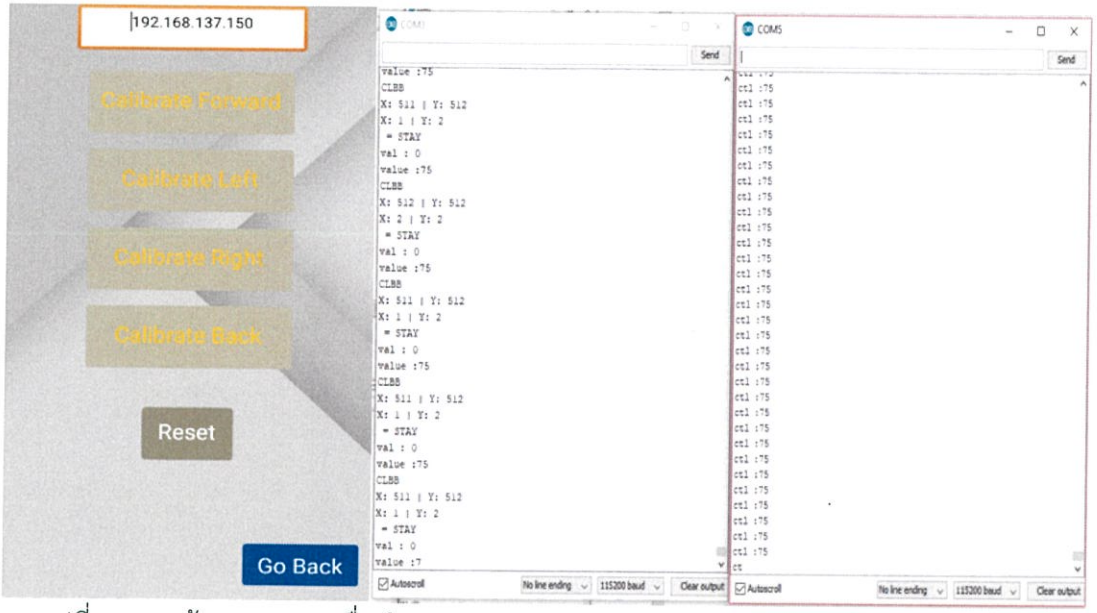
รูปที่ 4.75 หน้า Screen4 เมื่อทำการ Calibrate Left และ Serial Monitor ของ Arduino MEGA และ NodeMCU

เมื่อผู้ใช้งานต้องการทำการปรับเทียบเงื่อนไขมุมองศาการเอียงของเซนเซอร์ความเร่งเชิงเส้นให้เก้าอี้เข็นเคลื่อนที่ไปทางขวา ผู้ใช้งานจะต้องทำการเอียงศีรษะไปยังมุมองศาที่ต้องการใช้งานแล้วกดปุ่ม Calibrate Right เมื่อทำการกดปุ่ม Calibrate Right แล้ว text colour บนปุ่ม Calibrate Right จะเปลี่ยนเป็นสีเหลืองเพื่อบอกผู้ใช้งานว่าได้ทำการปรับเทียบมุมองศาการใช้ของการเคลื่อนที่เก้าอี้เข็นไปทางขวาแล้ว ค่า ctl ที่ส่งไปยัง Web Server ของ NodeMCU จะเท่ากับ 70 และที่ Serial Monitor ของ Arduino MEGA จะแสดงคำว่า “CLBR” ซึ่งแสดงว่าตัวเก้าอี้เข็นได้ทำการปรับเทียบมุมองศาการเอียงศีรษะเพื่อเคลื่อนที่เก้าอี้เข็นไปด้านขวาเรียบร้อยแล้วดังรูปที่ 4.76



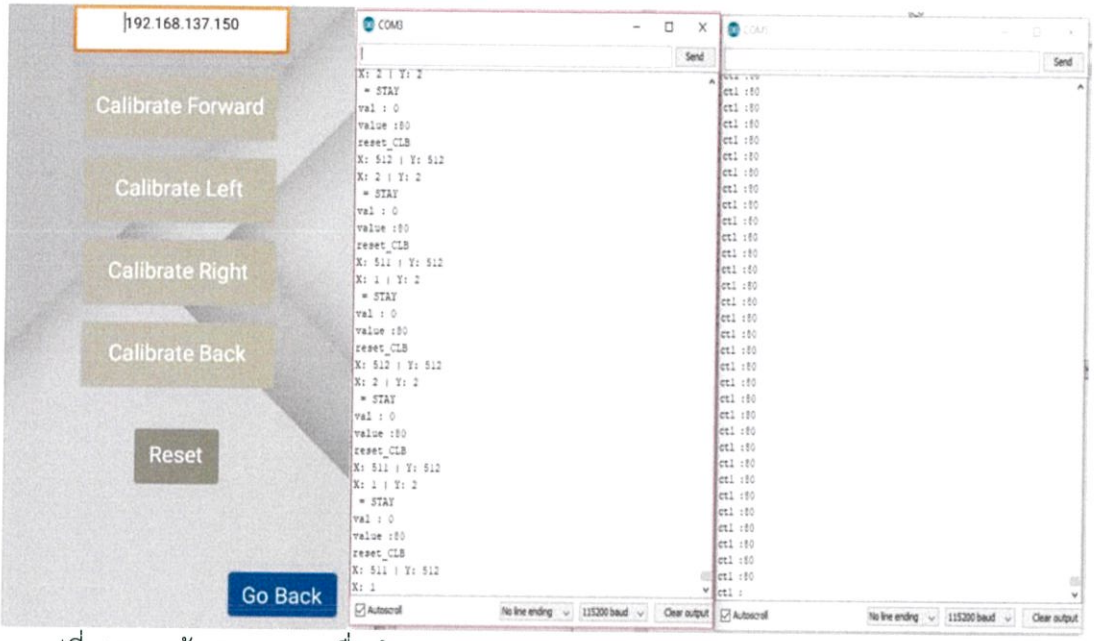
รูปที่ 4.76 หน้า Screen4 เมื่อทำการ Calibrate Right และ Serial Monitor ของ Arduino MEGA และ NodeMCU

เมื่อผู้ใช้งานต้องการทำการปรับเทียบเงื่อนไขมุมองศาการเอียงของเซนเซอร์ความเร่งเชิงเส้นให้เก้าอี้เข็นเคลื่อนที่ด้านหลัง ผู้ใช้งานจะต้องทำการเอียงศีรษะไปยังมุมองศาที่ต้องการใช้งานแล้วกดปุ่ม Calibrate Back เมื่อทำการกดปุ่ม Calibrate Back แล้ว text colour บนปุ่ม Calibrate Back จะเปลี่ยนเป็นสีเหลืองเพื่อบอกผู้ใช้งานว่าได้ทำการปรับเทียบมุมองศาการใช้ของการเคลื่อนที่เก้าอี้เข็นไปด้านหลังแล้ว ค่า ctl ที่ส่งไปยัง Web Server ของ NodeMCU จะเท่ากับ 75 และที่ Serial Monitor ของ Arduino MEGA จะแสดงคำว่า “CLBB” ซึ่งแสดงว่าตัวเก้าอี้เข็นได้ทำการปรับเทียบมุมองศาการเอียงศีรษะเพื่อเคลื่อนที่เก้าอี้เข็นไปข้างหลังเรียบร้อยแล้วดังรูปที่ 4.77



รูปที่ 4.77 หน้า Screen4 เมื่อทำการ Calibrate Back และ Serial Monitor ของ Arduino MEGA และ NodeMCU

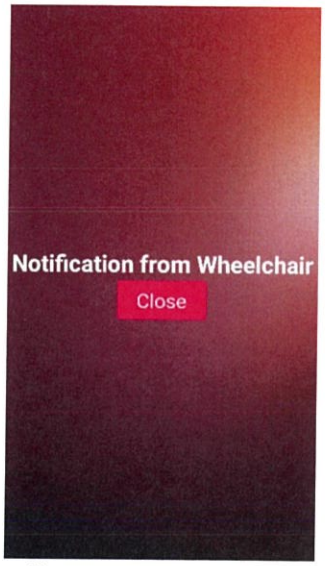
เมื่อผู้ใช้งานต้องการรีเซ็ตค่ามุมองศาการเอียงของเซนเซอร์ความเร่งเชิงเส้นให้เป็นค่าเริ่มต้นทั้งหมด ผู้ใช้งานจะต้องทำการกดปุ่ม Reset เมื่อทำการกดปุ่ม Reset แล้ว text colour บนปุ่ม Calibrate ทั้งสีปุ่มจะเปลี่ยนเป็นสีขาวเพื่อบอกผู้ใช้งานว่าได้ทำการรีเซ็ตมุมองศาการใช้ของการเคลื่อนที่เก้าอี้เข็นเป็นค่าเริ่มต้นแล้ว ค่า ctl ที่ส่งไปยัง Web Server ของ NodeMCU จะเท่ากับ 80 และที่ Serial Monitor ของ Arduino MEGA จะแสดงคำว่า “reset_CLB” ซึ่งแสดงว่าตัวเก้าอี้เข็นรีเซ็ตมุมองศาการเอียงศีรษะเรียบร้อยแล้วดังรูปที่ 4.78



รูปที่ 4.78 หน้า Screen4 เมื่อทำการ Reset Calibrate และ Serial Monitor ของ Arduino MEGA และ NodeMCU

ในแอปพลิเคชันจะมีฟังก์ชันการแจ้งเตือนจากเก้าอี้เข็นในกรณีที่ผู้ใช้งานเกิดปัญหาหรือต้องการเรียกผู้ดูแล ผู้ใช้งานเก้าอี้เข็นจะสามารถทำการแจ้งเตือนได้โดยการกดปุ่มแจ้งเตือนที่ติดตั้งอยู่บนเก้อี้เข็น NodeMCU จะทำการส่งค่า “3” ไปยัง FirebaseDB ใน tag ที่ชื่อว่า noti ดังรูปที่ 4.79 เมื่อเกิดการเปลี่ยนแปลงของค่าใน Firebase ตามเงื่อนไขที่แอปพลิเคชันได้กำหนดไว้ จะทำการเปิดหน้าจอ Screen3 ซึ่งเป็นหน้าจอหลักที่ใช้ในการแจ้งเตือนดังรูปที่ 4.80 และจะแจ้งเตือนโดยเริ่มแสดงเสียงแจ้งเตือนรวมไปถึงเปิดระบบสั่นของอุปกรณ์ นอกเหนือจากนี้จะมีฟังก์ชัน Push Notification ที่จะแสดงข้อความในหน้า Lock Screen ของอุปกรณ์สมาร์ตโฟนเป็นคำว่า “Notification from Wheelchair” ดังรูปที่ 4.81 เมื่อผู้ใช้งานทราบถึงการแจ้งเตือนแล้วจะต้องทำการกดปุ่ม Close เพื่อหยุดการแจ้งเตือน แอปพลิเคชันจะทำการย้อนกลับไปยังหน้าการใช้งานก่อนหน้า พร้อมทั้งส่งค่า “0” ไปยัง FirebaseDB ใน tag ที่ชื่อว่า noti ดังรูปที่ 4.82 เพื่อรีเซ็ตกลับเป็นค่าก่อนหน้าและพร้อมสำหรับการแจ้งเตือนครั้งต่อไป

รูปที่ 4.79 ค่า noti=3 ใน FirebaseDB ถูกส่งมาจาก NodeMCU



รูปที่ 4.80 หน้าจอ Screen3 สำหรับการแจ้งเตือน

รูปที่ 4.81 ฟังก์ชัน Push Notification แสดงในหน้า Lock Screen

รูปที่ 4.82 ค่า noti=0 ใน FirebaseDB ที่ถูกส่งมาจากแอปพลิเคชัน

บทที่ 5

สรุปผลและข้อเสนอแนะ

5.1 สรุปผล

ปริญญานิพนธ์ฉบับนี้นำเสนอการออกแบบและสร้างเก้าอี้เข็นควบคุมได้สำหรับผู้พิการอวัยวะส่วนบนลงมาและผู้สูงอายุ ซึ่งสามารถเลือกควบคุมทิศทางการเคลื่อนที่ของเก้าอี้เข็นได้ตามความเหมาะสมของสภาพร่างกาย ได้แก่ การใช้ก้านควบคุม การควบคุมทิศทางด้วยการเอียงศีรษะของผู้ใช้งานผ่านการตรวจวัดมุมเอียงของศีรษะด้วยเซนเซอร์ความเร่งเชิงเส้น ซึ่งออกแบบสำหรับผู้สูงอายุและผู้พิการที่ไม่สามารถใช้งานอวัยวะส่วนล่างในการควบคุมเก้าอี้เข็น และการควบคุมทิศทางบนแอปพลิเคชันสำหรับผู้ดูแล โดยมีระบบเรียกผู้ดูแลเพื่อขอความช่วยเหลือโดยแจ้งเตือนให้ผู้ดูแลทราบผ่านแอปพลิเคชันบนสมาร์ตโฟน

การดัดแปลงโครงสร้างและออกแบบชุดกลไกขับเคลื่อนเก้าอี้เข็นนั้น ผู้จัดทำได้คำนึงถึงการใช้งานจริงเพื่ออำนวยความสะดวกแก่ผู้พิการ โดยออกแบบให้มีความเร็วในการเคลื่อนที่ที่เหมาะสมจากการออกแบบชุดกลไกขับเคลื่อนที่ทดรอบความเร็วของมอเตอร์ ซึ่งสามารถขับเคลื่อนเก้าอี้เข็นได้ด้วยมอเตอร์ไฟฟ้าและการใช้แรงคนเข็นจากการออกแบบชุดกลไกที่สามารถเคลื่อนที่ได้โดยอิสระ สำหรับการใช้มอเตอร์ไฟฟ้าในการขับเคลื่อนนั้นจะใช้การควบคุมการจ่ายกำลังงานด้วยบอร์ดไมโครคอนโทรลเลอร์และวงจรขับแรงดันไฟฟ้ากระแสตรงแบบ H-Bridge ซึ่งจะทำให้เก้าอี้เข็นสามารถเคลื่อนที่ได้ในหลายทิศทาง

ในส่วนของการใช้งานก้านควบคุม (Joystick) ในการควบคุมทิศทางการเคลื่อนที่ของเก้าอี้เข็นนั้น ออกแบบขึ้นเพื่อรองรับผู้พิการที่ไม่สามารถเดินได้แต่ยังสามารถใช้อวัยวะส่วนบน เช่น มือในการควบคุมก้านควบคุมทิศทางเก้าอี้เข็น ก้านควบคุมจะถูกติดตั้งอยู่บริเวณที่พักแขนด้านขวาของตัวรถ จากการทดสอบจะพบว่าสามารถใช้งานก้านควบคุมในการควบคุมทิศทางที่ต้องการเคลื่อนที่ได้ดี สามารถเคลื่อนที่ได้อย่างนุ่มนวลทั้ง 4 ทิศทางได้แก่ การเคลื่อนที่ไปข้างหน้า (Forward) , การเคลื่อนที่ไปด้านหลัง (Backward) , การเลี้ยวซ้าย (Left) และการเลี้ยวขวา (Right) รวมไปถึงการหยุดรถในทันทีด้วยการกดปุ่มกดของก้านควบคุม

การใช้งานเซนเซอร์ความเร่งเชิงเส้น ADXL345 ในการใช้ศีรษะควบคุมทิศทางการเคลื่อนที่ของเก้าอี้เข็นนั้น เพื่อรองรับผู้พิการที่ไม่สามารถเดินได้และไม่สามารถใช้งานอวัยวะส่วนบน เช่น มือ ในการควบคุมก้านควบคุมบังคับทิศทางเก้าอี้เข็นหรือขับเคลื่อนเก้าอี้เข็นด้วยแรงแขน โดยผู้ใช้งานจะใช้การเอียงศีรษะไปยังทิศทางที่ต้องการเคลื่อนที่ไปเพื่อให้บอร์ดไมโครคอนโทรลเลอร์ทำการควบคุมชุดกลไกการเคลื่อนที่ให้ขับเคลื่อนตาม จากการทดลองพบว่าสามารถใช้งานเซนเซอร์ดังกล่าวในการควบคุมทิศทางที่ต้องการเคลื่อนที่ผ่านการตรวจวัดมุมเอียงได้ดี แต่เนื่องจากผู้ใช้งานแต่ละคนนั้นมีสรีระร่างกายและลักษณะการเอียงศีรษะที่แตกต่างกัน ซึ่งในการใช้งานจริงอาจส่งผลต่อการตรวจวัดสถานะทิศทางที่ผิดพลาด จึงติดตั้งระบบปรับเทียบมุมองศาการเอียงศีรษะของแต่ละบุคคล ซึ่งสามารถทำได้ผ่านแอปพลิเคชันบนอุปกรณ์สมาร์ทโฟน โดยเมื่อเข้าสู่ขั้นตอนการปรับเทียบมุมองศาการเอียง ผู้ใช้งานจะต้องทำการเอียงศีรษะไปยังมุมองศาที่ต้องการ จากการทดลองการปรับเทียบมุมองศาการเอียงของศีรษะพบว่ามุมองศาการเอียงศีรษะที่ใช้ในการควบคุมเก้าอี้เข็นนั้นเปลี่ยนไปตามมุมที่มีการปรับเทียบได้อย่างแม่นยำ ส่งผลให้ผู้ที่มีสรีระร่างกายที่แตกต่างกันสามารถใช้งานเก้าอี้เข็นคันนี้ได้

การใช้งานเก้าอี้เข็นในการเคลื่อนที่ไปยังทางลาดเอียง ตัวเก้าอี้เข็นได้ติดตั้งเซนเซอร์วัดความเร่งเชิงเส้นอีกตัวไว้ที่ด้านล่างของเก้าอี้เข็น เมื่อใดก็ตามที่เก้าอี้เข็นขึ้นไปยังทางที่มีความลาดเอียงที่มุมองศาของตัวรถจะเกิดการเปลี่ยนแปลง ซึ่งจะส่งผลให้มีการเพิ่มค่ากำลังของมอเตอร์สูงกว่าค่ากำลังปกติที่ใช้เคลื่อนที่ในแนวราบ และเมื่อใดก็ตามที่เก้าอี้เข็นจะต้องทำการลงไปยังทางลาดเอียง จะทำการเปลี่ยนสถานะควบคุมจากการเคลื่อนที่ไปข้างหน้าไปเป็นสถานะปกติเพื่อป้องกันไม่ให้ความเร็วในการเคลื่อนที่ของเก้าอี้เข็นนั้นสูงเกินไปจนเกิดอันตรายต่อผู้ใช้งานเก้าอี้เข็น จากการทดสอบพบว่าตัวเก้าอี้เข็นสามารถขึ้นพื้นที่มีความลาดเอียงได้เป็นอย่างดีโดยไม่มีการสูญเสียความเร็ว และในการลงทางลาดเอียงนั้นตัวเก้าอี้เข็นก็สามารถลงทางลาดเอียงได้อย่างปลอดภัยโดยที่ความเร็วไม่สูงจนเกินไปจนเกิดอันตรายแก่ผู้ใช้งานเก้าอี้เข็น

ในส่วนของแอปพลิเคชันควบคุมเก้าอี้เข็นนั้นถูกออกแบบมาเพื่อควบคุมทิศทางรวมไปถึงการปรับโหมดการควบคุมทิศทางของเก้าอี้เข็น โดยจะใช้การเชื่อมต่อไปยัง Web Server ของ NodeMCU ทั้งนี้มีระบบเรียกผู้ดูแลเพื่อแจ้งเตือนให้ผู้ดูแลทราบและเข้าทำการช่วยเหลือได้ทันท่วงที ซึ่งระบบแจ้งเตือนนั้นจะใช้การเชื่อมต่อไปยังฐานข้อมูล จากการทดสอบการใช้งานแอปพลิเคชันควบคุมเก้าอี้เข็นพบว่าเมื่อทำการเชื่อมต่อระหว่างเก้าอี้เข็นกับแอปพลิเคชันแล้ว จะสามารถปรับเปลี่ยนโหมดการควบคุมทิศทางได้ตามต้องการไม่ว่าจะเป็น การควบคุมทิศทางโดยใช้แอปพลิเคชัน, การควบคุมทิศทางโดยใช้ศีรษะ หรือการควบคุมทิศทางด้วยก้านควบคุม ในส่วนของ

การควบคุมทิศทางเก้าอี้เข็นโดยใช้แอปพลิเคชันนั้น สามารถตอบสนองต่อการควบคุมทิศทางได้ไม่แตกต่างจากการควบคุมด้วยก้านควบคุมหรือการใช้ศีรษะควบคุม นอกเหนือจากนี้ยังสามารถสั่งการปรับเทียบมุมเงื่อนไขทิศทางผ่านการเอียงของศีรษะผ่านแอปพลิเคชันได้อีกด้วย ในส่วนของการทดสอบระบบเรียกผู้ดูแล เมื่อผู้ทดสอบทำการกดปุ่มแจ้งเตือนที่ติดอยู่บนที่พักแขนด้านขวาของเก้าอี้เข็นแล้ว จะมีการแจ้งเตือนไปยังผู้ดูแลบนแอปพลิเคชัน โดยจะแสดงเสียงแจ้งเตือนรวมไปถึงเปิดระบบสั่นของอุปกรณ์สมาร์ตโฟน

5.2 ข้อเสนอแนะ

จากการทดสอบการนั่งเก้าอี้เข็นของผู้ใช้งานหลายคนซึ่งมีน้ำหนักที่แตกต่างกันออกไป พบว่า ความเร็วที่เก้าอี้เข็นสามารถวิ่งได้นั้นมีค่าไม่เท่ากัน หากผู้ใช้งานเก้าอี้เข็นมีน้ำหนักตัวมากก็จะส่งผลให้เก้าอี้เข็นวิ่งได้ช้า แต่หากผู้ใช้งานเก้าอี้เข็นมีน้ำหนักน้อยก็อาจทำให้เก้าอี้เข็นเคลื่อนที่เร็วจนเกินไป จึงควรมีการพัฒนาในส่วนของการกรอกน้ำหนักของผู้ใช้งานแต่ละคนเพื่อนำไปปรับเปลี่ยนค่ากำลังของมอเตอร์ที่จะต้องใช้ เพื่อทำให้ความเร็วของเก้าอี้เข็นมีความเหมาะสมกับผู้ใช้งานที่มีน้ำหนักตัวแตกต่างกันออกไป

บรรณานุกรม

[1] C.R.Engineering, LTD. "มอเตอร์ไฟฟ้ากระแสตรง (DC Motor)". http://www.crengineer.com/images/pulldown_1304840984/DC%20electric%20motors.pdf

[2] ธรรมธวัช หอสุวรรณ. "การออกแบบวงจรขับมอเตอร์กระแสตรงด้วยมอสเฟต (H-bridge)". http://menglab.blogspot.com/2017/04/h-bridge.html

[3] อลงกรณ์ พิมพ์พิณ. "MEMS : Micro Electro Mechanical Systems". http://www.vcharkarn.com/varticle/41941

[4] Majid Dadafshar. "Accelerometer and Gyroscopes Sensors". https://www.maximintegrated.com/en/app-notes/index.mvp/id/5830

[5] Analog Devices. "Digital Accelerometer ADXL345". ADXL345 datasheet, Jun. 2009.

[6] Analog Devices. "Using an Accelerometer for Inclination Sensing". ADXL345 Application Notes, Feb. 2010.

[7] Arduino. "Interfacing a Joystick". https://www.arduino.cc/en/Tutorial/JoyStick

[8] Analog Devices. "HC-05 Bluetooth Module". HC-05 datasheet, Jul. 2010.

[9] D. Wolber, H. Abelson, E. Spertus and L. Looney. "App Inventor". Sebastopol: O'Reilly, 2011.

ภาคผนวก ก

โปรแกรมในระบบเก้าอี้เข็นควบคุมได้

โปรแกรมที่ใช้ในการรับส่งข้อมูลคำสั่งและควบคุมเก้าอี้เข็น

```
#include <Wire.h>
#include <SoftwareSerial.h>
#include <math.h>
#include <stdlib.h>
#include <Pushbutton.h>
#define BUTTON_NOTI  11
#define BUTTON_PINCARA  12
#define BUTTON_PINCARJ  13
#define BUTTON_PINJOY  A2
Pushbutton buttonNOTI(BUTTON_NOTI);
Pushbutton buttonCARA(BUTTON_PINCARA);
Pushbutton buttonCARJ(BUTTON_PINCARJ);
Pushbutton buttonJOY(BUTTON_PINJOY);
int m = 1;
int n = 0;
int but;
byte b;
int adxl;

//int zet,phi,psi;
int zetL, phiL, psiL;
int zetR, phiR, psiR;
int zetF, phiF, psiF;
int zetB, phiB, psiB;
//int zet2,phi2,psi2;

int esp;
int s;
int s2;

int ctl;
int recv;
int noti;

//double xcal,ycal,zcal;

//////////////////////////////////////
#define DEVICE (0x53)                    //ADXL345 device address
#define TO_READ (6)                      //num of bytes we are going to read
each time (two bytes for each axis)
byte buf2[TO_READ] ;                     //6 bytes buffS1er for saving data
read from the device
//string buffS1er to transform data before sending it to the serial port
int regAddress = 0x32;                   //first axis-acceleration-data
register on the ADXL345

double xg, yg, zg;
double zet, phi, psi;
double xcal, ycal, zcal;
int x, y , z;                            //three axis acceleration data
//////////////////////////////////////////////////////////////
int xAxis;
int yAxis;
int swB;
int X;
int Y;
int val;
int speeds;
int xPin = A0;
int yPin = A1;
//int buttonPin = 2;
```

```
int xPosition = 0;
int yPosition = 0;
//int buttonState = 0;
////////////////////////////////////////
#define pwm1 8
#define pwm2 9
#define in11 2
#define in12 3
#define in21 4
#define in22 5

int u ;
int Mode;

//////////////////////////////////////////////////////////////////////
//backward buzzer  // defines pins numbers
const int trigPin = A3;
const int echoPin = A4;
//int buzzerPin = A3;
const int buzpin =  A5;
// defines variables
long duration;
int distance;
int buzstate = LOW;
unsigned long previousMillis = 0;
const long interval1 = 100;
const long interval2 = 300;
const long interval3 = 500;

////////////////////////////////////////
void setup()
{
  Serial.begin(115200);
  Serial1.begin(115200);
  Serial2.begin(115200);

  Wire.begin();         // join i2c bus (address optional for master)

  zetL = 30; phiL = 30; psiL = 0;
  zetR = -30; phiR = 30; psiR = 0;
  zetF = 0; phiF = 30; psiF = -30;
  zetB = 0; phiB = 30, psiB = 30;
  ////////////////////////////////////////////
  //Turning on the ADXL345
  writeTo(DEVICE, 0x2D, 0);
  writeTo(DEVICE, 0x2D, 16);
  writeTo(DEVICE, 0x2D, 8);
  delay(100);

  pinMode(pwm1, OUTPUT);
  pinMode(pwm2, OUTPUT);
  pinMode(in11, OUTPUT);
  pinMode(in12, OUTPUT);
  pinMode(in21, OUTPUT);
  pinMode(in22, OUTPUT);

  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  pinMode(buzpin, OUTPUT);

  Mode = 1;
  speeds = 100;
  val = speeds;
```

```
}

void loop()
{
  //mode 1 app40 2 adxl45 3joy50

  val = speeds;

  espserial();
  checkCLB();


  adxlBT();
  readadxl();

  //slope();

  //detectdirectionCAR();

  /*-----------------------------------------------------------------------------
*/
  swChange();

  //if (buttonJOY.getSingleDebouncedRelease()){
  if (buttonJOY.isPressed()) {

    Stop();
    //delay(1000);

  }
  /*-----------------------------------------------------------------------------
*/
  if (Mode == 1)
  {
    appcontrol();

    if (esp == 45) {
      Mode = 2;
      delay(500);
      Serial.println("CHANGE MODE TO ADXL");
      delay(1500);
    }
    if (esp == 50) {
      Mode = 3;
      delay(500);
      Serial.println("CHANGE MODE TO JOYSTICK");
      delay(1500);
    }
  }
  /*-----------------------------------------------------------------------------
*/
  if (Mode == 2)
  {
    // adxlBT();
    adxlcontrol();

    if (adxl == 6) {

      Stop();
      delay(100);

    }

    if (adxl == 7) {
```

```
      Mode = 3;

      delay(500);

      Serial.println("CHANGE MODE TO JOYSTICK");
      delay(1500);
      return;
    }


    if (esp == 40) {
      Mode = 1;
      delay(500);
      Serial.println("CHANGE MODE TO APP");
      delay(1500);
    }
    if (esp == 50) {
      Mode = 3;
      delay(500);
      Serial.println("CHANGE MODE TO JOYSTICK");
      delay(1500);
    }

  }
  /*-------------------------------------------------------------------------
*/
  if (Mode == 3)
  {
    readjoy();
    // adxlBT();
    joycontrol();

    if (esp == 40) {
      Mode = 1;
      delay(500);
      Serial.println("CHANGE MODE TO APP");
      delay(1500);
    }
    if (esp == 45) {
      Mode = 2;
      delay(500);
      Serial.println("CHANGE MODE TO ADXL");
      delay(1500);
    }
    if (adxl == 7) {
      Mode = 2;
      delay(500);

      Serial.println("CHANGE MODE TO ADXL");
      delay(1500);
      return;
    }

    //delay(200);
  }
  /*-------------------------------------------------------------------------
*/

  Serial.print("val : ");
  Serial.println(val);


}
/*-------------------------------------------------------------------------*/
```

```
void swChange() {
  if (buttonNOTI.getSingleDebouncedRelease()) {
    delay(500);

    //for(int i=0;i<=5;i++){
    Serial1.write(99);
    //  }
    Serial.println("     NOTI");
    delay(1000);
    return;

  }
  if (buttonCARJ.getSingleDebouncedRelease()) {
    delay(500);


    Mode = 3;
    esp = 0;
    Serial.println("CHANGE MODE TO JOYSTICK");
    delay(1500);

  }
  if (buttonCARA.getSingleDebouncedRelease()) {
    delay(500);


    Mode = 2;
    esp = 0;
    Serial.println("CHANGE MODE TO ADXL");
    delay(1500);

  }
}

void slope() {
  if (psi >= 10 && phi >= 10) {

    val = speeds + 60;
  }
  else if (psi <= -10 && phi >= 10) {

    val = speeds - 100;
  } else val = speeds;

}

void adxlcontrol()
{

  if (adxl == 1) //range of adxl for which the car remains stationary
  {

    Stay();
  }
  else if ( adxl == 5)
  {

    Left();


  }
  else if (adxl == 4)
  {
    Right();
  }
```

```
  else if (adxl == 2)
  {

    Forward();

  }
  else if (adxl == 3)
  {

    Backward();
  }
  /* else if(adxl== 6)
    {

       Stop();
    }
    else if(adxl== 7)
    {

       Stop();
    }*/
  else {

    Stay();

  }
}


void appcontrol()
{
  if ( esp == 10)
  {

    Stay();

  }
  else if ( esp == 20)
  {

    Left();

  }
  else if (esp == 30)
  {

    Right();

  }
  else if (esp == 15)
  {

    Forward();
  }

  else if (esp == 35)
  {

    Backward();

  }
  else if (esp == 25)
  {

    Stop();
```

```
  }
  else {

    Stay();

  }


}
void checkCLB() {

  ////clb ctl=55
  if (esp == 60) {
    /*psiF=psi;
      phiF=phi;*/
    Serial2.write(60);

    Serial.println("CLBF");
  }
  if (esp == 65) {
    /*zetL=zet;
      phiL=phi;*/
    Serial2.write(65);

    Serial.println("CLBL");
  }
  if (esp == 70) {
    /*zetR=zet;
      phiR=phi;*/
    Serial2.write(70);

    Serial.println("CLBR");
  }
  if (esp == 75) {
    /*psiB=psi;
      phiB=phi;*/
    Serial2.write(75);

    Serial.println("CLBB");
  }
  if (esp == 80) {
    /*zetL=30;phiL=30;psiL=0;
      zetR=-30;phiR=30;psiR=0;
      zetF=0;phiF=30;psiF=-30;
      zetB=0;phiB=30,psiB=30;*/
    Serial2.write(80);

    Serial.println("reset_CLB");
  }
  Serial2.write(0);


  /*Serial.print(" psiF : ");
    Serial.print(psiF);
    Serial.print(" phiF : ");
    Serial.println(phiF);
    Serial.print(" zetL : ");
    Serial.print(zetL);
    Serial.print(" phiL : ");
    Serial.println(phiL);
    Serial.print(" zetR : ");
    Serial.print(zetR);
    Serial.print(" phiR : ");
    Serial.println(phiR);
```

```
    Serial.print(" psiB : ");
    Serial.print(psiB);
    Serial.print(" phiB : ");
    Serial.println(phiB);*/


}
void readjoy() {

  xPosition = analogRead(xPin);
  yPosition = analogRead(yPin);
  // buttonState = digitalRead(buttonPin);
  X = xPosition - 510;
  Y = yPosition - 510;

}

void joycontrol()
{

  /*xAxis=analogRead(A0);//A0
    yAxis=analogRead(A1);//A1

    Serial.print("X = ");
    X=xAxis-510;
    Serial.println(X);
    Serial.print("Y = ");
    Y=yAxis-510;
    Serial.println(Y);*/

  /*
    xPosition = analogRead(xPin);
    yPosition = analogRead(yPin);
    buttonState = digitalRead(buttonPin);

    Serial.print("X: ");
    Serial.print(xPosition);
    Serial.print(" | Y: ");
    Serial.print(yPosition);
    Serial.print(" | Button: ");
    Serial.println(buttonState);*/



 Serial.print("X: ");
 Serial.print(xPosition);
 Serial.print(" | Y: ");
 Serial.println(yPosition);
 // Serial.print(" | Button: ");
 // Serial.println(buttonState);

 Serial.print("X: ");

 Serial.print(X);
 Serial.print(" | Y: ");

 Serial.println(Y);
 // Serial.print(" | Button: ");
 //Serial.println(buttonState);


 /*if (button.getSingleDebouncedRelease()){

    Serial.println(" BUTTON DebouncedRelease ");
    Stop();
```

```
      delay (1000);
      c=0;


     }
     if (button.isPressed()){

      Serial.println(" BUTTON ISPRESSED");


      c++;


     }
     if (c==15){

      Serial.println(" change ");

      c=0;
      delay (1000);
     }*/



  if (Y < -200 && 0 <= X < 50)
  {

    Backward();
    // Stop();
  }


  else if (Y > 200 && 0 <= X < 50)
  {
    Forward();

  }

  else if (X < -200 && 0 <= Y < 50 )
  {
    Right();

  }

  else if (X > 200 && 0 <= Y < 50)
  {

    Left();
  }
  else if (buttonJOY.isPressed()) {

    Stop();
    //delay(1000);

  }
  else
  {
    Stay();

  }

  // delay(200);
}
```

```
void readadxl() {

  readFrom(DEVICE, regAddress, TO_READ, buf2); //read the acceleration data
from the ADXL345
  //each axis reading comes in 10 bit resolution, ie 2 bytes.  Least
Significat Byte first!!
  //thus we are converting both bytes in to one int
  x = (((int)buf2[1]) << 8) | buf2[0];
  y = (((int)buf2[3]) << 8) | buf2[2];
  z = (((int)buf2[5]) << 8) | buf2[4];

  xg = x * 0.0039;
  yg = y * 0.0039;
  zg = z * 0.0039;

  xcal = (xg - (0.0078)) / 1.0218;
  x = xcal / 0.0039;
  ycal = (yg - (-0.0429)) / 1.0335;
  y = ycal / 0.0039;
  zcal = (zg - (0.0995)) / 0.9848;
  z = zcal / 0.0039;

  zet = atan2(x, sqrt(pow(y, 2) + pow(z, 2))) * 180 / 3.14159265; //x
  psi = atan2(y, sqrt(pow(x, 2) + pow(z, 2))) * 180 / 3.14159265; //y
  phi = atan2(sqrt(pow(x, 2) + pow(y, 2)), z) * 180 / 3.14159265; //z



  //we send the x y z values as a string to the serial port
  /*  Serial.print("x, y, z :");
              Serial.print(x);
              Serial.print(" ");
              Serial.print(y);
              Serial.print(" ");
              Serial.println(z);*/
  /*Serial.print("zet, psi, phi :");
               Serial.print(zet);
               Serial.print(" ");
               Serial.print(psi);
               Serial.print(" ");
               Serial.println(phi);*/



}
//Writes val to address register on device
void writeTo(int device, byte address, byte val) {
  Wire.beginTransmission(device); //start transmission to device
  Wire.write(address);        // send register address
  Wire.write(val);        // send value to write
  Wire.endTransmission(); //end transmission
}
//reads num bytes starting from address register on device in to buff array
void readFrom(int device, byte address, int num, byte buff[]) {
  Wire.beginTransmission(device); //start transmission to device k
  Wire.write(address);        //sends address to read from
  Wire.endTransmission(); //end transmission
  Wire.beginTransmission(device); //start transmission to device
  Wire.requestFrom(device, num);    // request 6 bytes from device
  int i = 0;
  while (Wire.available())   //device may send less than requested (abnormal)
  {
    buff[i] = Wire.read(); // receive a byte
    i++;
  }
```

```
  Wire.endTransmission(); //end transmission
}
void setRegisterBit(byte regAdress, int bitPos, bool state) {
  readFrom(DEVICE, regAdress, 1, &b);
  if (state) {
    b |= (1 << bitPos);  // forces nth bit of _b to be 1.  all other bits
left alone.
  }
  else {
    b &= ~(1 << bitPos); // forces nth bit of _b to be 0.  all other bits
left alone.
  }
  writeTo(DEVICE, regAdress, b);
}
void detectdirectionCAR() {
  if (abs(zet) < 30 && abs(psi) < 30 && abs(phi) < 30) //range of adxl for
which the car remains stationary
  {

    Serial.println("  STAY  ");
  }
  else if ( zet >= 30 && phi >= 30 && abs(zet) > abs(psi) && abs(phi) >
abs(psi))
  {

    Serial.println("  LEFT  ");


  }
  else if (zet <= -30 && phi >= 30 && abs(zet) > abs(psi) && abs(phi) >
abs(psi))
  {
    Serial.println("  RIGHT  ");
  }
  else if (psi <= -30 && phi >= 30 && abs(psi) > abs(zet) && abs(phi) >
abs(zet))
  {

    Serial.println("  FORWARD  ");

  }
  else if (psi >= 30 && phi >= 30 && abs(psi) > abs(zet) && abs(phi) >
abs(zet))
  {

    Serial.println("  BACKWARD  ");
  }
  else {

    Serial.println("  STAY  ");

  }
}

void espserial() {


  if (Serial1.available() > 0)
  {
    esp = Serial1.read();
    Serial.print("value :");
    Serial.println(esp);


  }
```

```
    Serial1.write(0);

  }


  void adxlBT()
  {

    if (Serial2.available() > 0) {

      adxl = Serial2.read();

      Serial.print(" adxl :");
      Serial.println(adxl);

    }
    else ;
    // Serial2.write(0);


  }

  ////////////////////////////
  void Stay()
  {
    Serial.println(" = STAY");

    val = 0;

    digitalWrite(pwm1, LOW);
    digitalWrite(pwm2, LOW);
  }
  ////////////////////////////////////////
  void Forward()
  {
    Serial.println(" = FORWARD");

    if (psi >= 5 && phi >= 5) {

      val = speeds + 80;
    }
    else if (psi <= -5 && phi >= 5) {

      val = 0;
    } else val = speeds + 20;
    //val=val+20;


    digitalWrite(in11, LOW);
    digitalWrite(in12, HIGH);
    digitalWrite(in21, HIGH);
    digitalWrite(in22, LOW);


    analogWrite(pwm1, val);
    analogWrite(pwm2, val);
  }
  ////////////////////////////////////////
  void  Backward()
  {
    Serial.println(" = BACKWARD");

    if (psi >= 5 && phi >= 5) {

      val = 0;
```

```
}
else if (psi <= -5 && phi >= 5) {

  val = 0;
} else val = speeds;
val = val;

digitalWrite(in11, HIGH);
digitalWrite(in12, LOW);
digitalWrite(in21, LOW);
digitalWrite(in22, HIGH);
analogWrite(pwm1, val);
analogWrite(pwm2, val);

/* ////////////////////////////////////////////
  // Clears the trigPin
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);
  // Calculating the distance
  distance= duration*0.034/2;
  // Prints the distance on the Serial Monitor
  //Serial.print("Distance: ");
  Serial.println(distance);

  if (distance < 50)
  {
  unsigned long currentMillis = millis();

  if (currentMillis - previousMillis >= interval1) {

   previousMillis = currentMillis;
   if (buzstate == LOW) {
     buzstate = HIGH;
   } else {
     buzstate = LOW;
   }
   digitalWrite(buzpin, buzstate);
  }
  Serial.println("Buzz_____Buzz11111111");
  }
  else if  (distance < 100 && distance >= 50)
  {
  unsigned long currentMillis = millis();

  if (currentMillis - previousMillis >= interval2) {

   previousMillis = currentMillis;
   if (buzstate == LOW) {
     buzstate = HIGH;
   } else {
     buzstate = LOW;
   }
   digitalWrite(buzpin, buzstate);
  }
  Serial.println("Buzz_____Buzz22222222");
  }
  else if  (distance < 150 && distance >= 100)
  {
  unsigned long currentMillis = millis();
```

```
    if (currentMillis - previousMillis >= interval3) {

     previousMillis = currentMillis;
     if (buzstate == LOW) {
       buzstate = HIGH;
     } else {
       buzstate = LOW;
     }
     digitalWrite(buzpin, buzstate);
    }
    Serial.println("Buzz_____Buzz33333333");
    }*/


 }
 ////////////////////////////////////////
 void Left()
 {
   Serial.println(" = LEFT");


   val = val + 40;



   digitalWrite(in11, LOW);
   digitalWrite(in12, LOW);
   // digitalWrite(in12,HIGH);
   digitalWrite(in21, HIGH);
   digitalWrite(in22, LOW);

   digitalWrite(in21, HIGH);
   digitalWrite(in22, LOW);
   analogWrite(pwm1, HIGH);
   analogWrite(pwm2, val);


 }
 ////////////////////////////////////////
 void Right()
 {
   Serial.println(" = RIGHT");


   val = val + 40;


   digitalWrite(in11, LOW);
   digitalWrite(in12, HIGH);
   //digitalWrite(in21,HIGH);
   digitalWrite(in21, LOW);
   digitalWrite(in22, LOW);

   digitalWrite(in11, LOW);
   digitalWrite(in12, HIGH);

   analogWrite(pwm1, val);
   //analogWrite(pwm2,LOW);
   analogWrite(pwm2, HIGH);


 }
 ////////////////////////////////////////
 void Stop()
```

```
{
  Serial.println(" = STOP");

  val = 255;


  digitalWrite(in11, LOW);
  digitalWrite(in12, LOW);
  digitalWrite(in21, LOW);
  digitalWrite(in22, LOW);


  analogWrite(pwm1, val);
  analogWrite(pwm2, val);
}
////////////////////////////////////////
```

โปรแกรมสร้าง Web Server ที่ใช้ติดต่อสื่อสารระหว่างแอปพลิเคชันและบอร์ดไมโครคอนโทรเลอร์

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <ESP8266mDNS.h>
#include <FirebaseArduino.h>
#include <SoftwareSerial.h>
#include <Wire.h>
SoftwareSerial NodeSerial(D2, D3); // RX | TX

const char* ssid = "Petch-WIFI";
const char* password = "0859078578";

#define FIREBASE_HOST "test1-ca88d.firebaseio.com"
#define FIREBASE_AUTH "VXqOiOBFRgvSdqvKRzBLYbATGbHEZlTiW3aOQ25k"
//#define FIREBASE_HOST "testprj-143fe.firebaseio.com"
//#define FIREBASE_AUTH "8UfwenlFnohkezJyjIvUa1syh3qncmASewdimqLw"
String ip ;
String str ;
int ctl;
int value;


String  data ="";
int motor_speed;

ESP8266WebServer server(80);
//WiFiClient client;

const int led = 13;

void handleRoot() {
  digitalWrite(led, 1);
  server.send(200, "text/plain", "hello from esp8266!");
  digitalWrite(led, 0);
}

void handleNotFound() {
  digitalWrite(led, 1);
  String message = "File Not Found\n\n";
  message += "URI: ";
  message += server.uri();
  message += "\nMethod: ";
  message += (server.method() == HTTP_GET) ? "GET" : "POST";
  message += "\nArguments: ";
  message += server.args();
  message += "\n";
  for (uint8_t i = 0; i < server.args(); i++) {
    message += " " + server.argName(i) + ": " + server.arg(i) + "\n";
  }
  server.send(404, "text/plain", message);
  Serial.println(" NotFound ARG ");
  digitalWrite(led, 0);
}

void setup(void) {
```

```
  Wire.begin();          // join i2c bus (address optional for master)
  NodeSerial.begin(115200);
  pinMode(D2, INPUT);
 pinMode(D3, OUTPUT);

  pinMode(led, OUTPUT);
  digitalWrite(led, 0);
  Serial.begin(115200);

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  Serial.println("");

 //Wait for connection
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  ip=WiFi.localIP().toString();
  Serial.println(ip);

  if (MDNS.begin("esp8266")) {
    Serial.println("MDNS responder started");
  }

//server.on("/", handleRoot);

  server.on("/inline", []() {
    server.send(200, "text/plain", "this works as well");
  });


  server.on("/", []() {

  String ctl = server.arg("ctl");

 //String state = url.substring(0,2);
 //String value = url.substring(3);
//motor_speed = value.toInt();
  Serial.println(ctl);
  str=ctl;



  server.send(200, "text/plain", " recive value ");
  });



server.onNotFound(handleNotFound);


  server.begin();
  Serial.println("HTTP server started");
  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
  Firebase.setString("ip",ip);
```

```
Serial.println("connect firebase");
}
void loop(void) {

  server.handleClient();

  if(str=="nt"){
    Firebase.setString("noti","3");
    str="";
  }
if(NodeSerial.available()>0)
            {
        value=NodeSerial.read();
  //Serial.print("value :");
  //Serial.println(value);
            }
if (value==99){
    delay(500);

 str="nt";
Serial.println("     NOTI");
delay(1000);


   }
ctl=str.toInt();
NodeSerial.write(ctl);
Serial.print("ctl :");
Serial.println(str);


  }
```

โปรแกรมที่ใช้ในการรับส่งค่าเพื่อใช้ADXL345ในการควบคุมเก้าอี้เข็น

```
#include <SoftwareSerial.h>
#include <Wire.h>
SoftwareSerial ntn(10, 11);// RX, TX

#define ADXL345_DEVID           0X00
#define ADXL345_RESERVED1  0X01
#define ADXL345_THRESH_TAP      0X1D
#define ADXL345_OFSX            0X1E
#define ADXL345_OFSY            0X1F
#define ADXL345_OFSZ            0X20
#define ADXL345_DUR             0X21
#define ADXL345_LATENT          0X22
#define ADXL345_WINDOW          0X23
#define ADXL345_THRESH_ACT      0X24
#define ADXL345_THRESH_INACT    0X25
#define ADXL345_TIME_INACT      0X26
#define ADXL345_ACT_INACT_CTL   0X27
#define ADXL345_THRESH_FF       0X28
#define ADXL345_TIME_FF         0X29
#define ADXL345_TAP_AXES        0X2A
#define ADXL345_ACT_TAP_STATUS  0X2B
#define ADXL345_BW_RATE         0X2C
#define ADXL345_POWER_CTL       0X2D
#define ADXL345_INT_ENABLE      0X2E
#define ADXL345_INT_MAP         0X2F
#define ADXL345_INT_SOURCE      0X30
#define ADXL345_DATA_FORMAT     0X31
#define ADXL345_DATA            0X32
#define ADXL345_FIFO_CTL        0X38
#define ADXL345_FIFO_STATUS     0X39

//interrupts PINs
#define ADXL345_INT1_PIN 0X00
#define ADXL345_INT2_PIN 0X01

/*interruptsrupt bit position*/
#define ADXL345_INT_DATA_READY_BIT 0X07
#define ADXL345_INT_SINGLE_TAP_BIT 0X06
#define ADXL345_INT_DOUBLE_TAP_BIT 0X05
#define ADXL345_INT_ACTIVITY_BIT   0X04
#define ADXL345_INT_INACTIVITY_BIT 0X03
#define ADXL345_INT_FREE_FALL_BIT  0X02
#define ADXL345_INT_WATERMARK_BIT  0X01
#define ADXL345_INT_OVERRUNY_BIT   0X00

#define ADXL345_DATA_READY 0X07
#define ADXL345_SINGLE_TAP 0X06
#define ADXL345_DOUBLE_TAP 0X05
#define ADXL345_ACTIVITY   0X04
#define ADXL345_INACTIVITY 0X03
#define ADXL345_FREE_FALL  0X02
```

```
#define ADXL345_WATERMARK   0X01
#define ADXL345_OVERRUNY    0X00

byte interrupts;
byte b;
int s;
int d,m;

#define DEVICE (0X53)      //ADXL345 device address
#define TO_READ (6)        //num of bytes we are going to read each time (two
bytes for each axis)
byte buf2[TO_READ] ;                    //6 bytes buffS1er for saving data read
from the device
      //string buffS1er to transform data before sending it to the serial port
int regAddress = 0X32;                  //first axis-acceleration-data register
on the ADXL345

double xg, yg, zg;
double zet,phi,psi;
int zet2,phi2,psi2;
double xcal, ycal, zcal;
int x, y ,z;    //three axis acceleration data

int zetL,phiL,psiL;
int zetR,phiR,psiR;
int zetF,phiF,psiF;
int zetB,phiB,psiB;

void setup(){
  Wire.begin();          // join i2c bus (address optional for master)
  ntn.begin(115200);
  Serial.begin(115200);  // start serial for output

 //Turning on the ADXL345
  writeTo(DEVICE, 0X2D, 0);
  writeTo(DEVICE, 0X2D, 16);
  writeTo(DEVICE, 0X2D, 8);
  delay(100);
//set activity/ inactivity thresholds (0-255)
  writeTo(DEVICE, ADXL345_THRESH_ACT,constrain(75, 0,255));
  writeTo(DEVICE, ADXL345_THRESH_INACT,constrain(75, 0, 255));
  writeTo(DEVICE, ADXL345_TIME_INACT,constrain(10, 0, 255));
  delay(100);
 //look of activity movement on this axes - 1==on; 0==off
  setRegisterBit(ADXL345_ACT_INACT_CTL, 6, 1);
  setRegisterBit(ADXL345_ACT_INACT_CTL, 5, 1);
  setRegisterBit(ADXL345_ACT_INACT_CTL, 4, 1);
  delay(100);
 //look of inactivity movement on this axes - 1==on; 0==off
  setRegisterBit(ADXL345_ACT_INACT_CTL, 2, 1);
  setRegisterBit(ADXL345_ACT_INACT_CTL, 1, 1);
  setRegisterBit(ADXL345_ACT_INACT_CTL, 0, 1);
  delay(100);
 //look of tap movement on this axes - 1==on; 0==off
  setRegisterBit(ADXL345_TAP_AXES, 2, 1);
  setRegisterBit(ADXL345_TAP_AXES, 1, 1);
  setRegisterBit(ADXL345_TAP_AXES, 0, 1);
```

```
  delay(100);
 //set values for what is a tap, and what is a double tap (0-255)
  writeTo(DEVICE, ADXL345_THRESH_TAP, constrain(20, 0, 255));//62.5mg/LSB 20
  writeTo(DEVICE, ADXL345_DUR, constrain(100, 0, 255));//625us/LSB 100
  writeTo(DEVICE, ADXL345_LATENT,constrain(240, 0, 255));//1.25ms/LSB 200
  writeTo(DEVICE, ADXL345_WINDOW,constrain(255, 0, 255));//1.25ms/LSB 255
  delay(100);
 //set values for what is considered freefall (0-255)
  writeTo(DEVICE, ADXL345_THRESH_FF,constrain(7, 0, 255));//(5-9)recommended -
62.5mg per increment
  writeTo(DEVICE, ADXL345_TIME_FF,constrain(45, 0, 255));//(20-70)recommended - 5ms
per increment
  delay(100);
 //setting all interruptsupts to take place on int pin 1
  setRegisterBit(ADXL345_INT_MAP, ADXL345_INT_SINGLE_TAP_BIT, ADXL345_INT1_PIN );
  setRegisterBit(ADXL345_INT_MAP, ADXL345_INT_DOUBLE_TAP_BIT, ADXL345_INT1_PIN );
  setRegisterBit(ADXL345_INT_MAP, ADXL345_INT_FREE_FALL_BIT, ADXL345_INT1_PIN );
  setRegisterBit(ADXL345_INT_MAP, ADXL345_INT_ACTIVITY_BIT, ADXL345_INT1_PIN );
  setRegisterBit(ADXL345_INT_MAP, ADXL345_INT_INACTIVITY_BIT, ADXL345_INT1_PIN );
  delay(100);
 //register interruptsupt actions - 1==on; 0==off
  setRegisterBit(ADXL345_INT_ENABLE, ADXL345_INT_SINGLE_TAP_BIT, 1);
  setRegisterBit(ADXL345_INT_ENABLE, ADXL345_INT_DOUBLE_TAP_BIT, 1);
  setRegisterBit(ADXL345_INT_ENABLE, ADXL345_INT_FREE_FALL_BIT, 1);
  setRegisterBit(ADXL345_INT_ENABLE, ADXL345_INT_ACTIVITY_BIT, 1);
  setRegisterBit(ADXL345_INT_ENABLE, ADXL345_INT_INACTIVITY_BIT, 1);
  delay(100);

  zetL=30;phiL=30;psiL=0;
  zetR=-30;phiR=30;psiR=0;
  zetF=0;phiF=30;psiF=-30;
  zetB=0;phiB=30,psiB=30;

}

void loop(){

  if(ntn.available()>0){
      m = ntn.read();
              Serial.print("motor : ");
              Serial.print(m);
              Serial.print(" || ");
            }
      else ;

  readFrom(DEVICE, regAddress, TO_READ, buf2); //read the acceleration data
from the ADXL345
              //each axis reading comes in 10 bit resolution, ie 2 bytes.
Least Significat Byte first!!
              //thus we are converting both bytes in to one int
  x = (((int)buf2[1]) << 8) | buf2[0];
  y = (((int)buf2[3]) << 8) | buf2[2];
  z = (((int)buf2[5]) << 8) | buf2[4];

  xg = x *0.0039;
  yg = y *0.0039;
```

```
   zg = z *0.0039;

   xcal = (xg-(-0.0527))/0.9965;
   x = xcal/0.0039;
   ycal = (yg-(-0.0468))/0.9984;
   y = ycal/0.0039;
   zcal = (zg-(-0.0468))/0.9750;
   z = zcal/0.0039;

   zet = atan2(x,sqrt(pow(y,2)+pow(z,2)))* 180/3.14159265;//x
   psi = atan2(y,sqrt(pow(x,2)+pow(z,2)))* 180/3.14159265;//y
   phi = atan2(sqrt(pow(x,2)+pow(y,2)),z)* 180/3.14159265;//z
 /*Serial.print(x);
 Serial.print(" ");
 Serial.print(y);
 Serial.print(" ");
 Serial.println(z);*/


 /*Serial.print(" psiF : ");
   Serial.print(psiF);
   Serial.print(" phiF : ");
   Serial.print(phiF);
   Serial.print(" zetL : ");
   Serial.print(zetL);
   Serial.print(" phiL : ");
   Serial.print(phiL);
   Serial.print(" zetR : ");
   Serial.print(zetR);
   Serial.print(" phiR : ");
   Serial.print(phiR);
   Serial.print(" psiB : ");
   Serial.print(psiB);
   Serial.print(" phiB : ");
   Serial.println(phiB);*/

if(m==60){
psiF=psi;
phiF=phi;
Serial.println("CLBF");
}
 if(m==65){
zetL=zet;
phiL=phi;
Serial.println("CLBL");
}
 if(m==70){
zetR=zet;
phiR=phi;
Serial.println("CLBR");
}
 if(m==75){
psiB=psi;
phiB=phi;
Serial.println("CLBB");
}
 if(m==80){
  zetL=30;phiL=30;psiL=0;
  zetR=-30;phiR=30;psiR=0;
  zetF=0;phiF=30;psiF=-30;
```

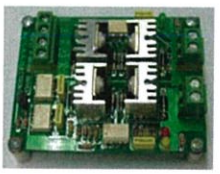
```
  zetL=0;phiB=30,psiB=30;
  Serial.println("reset_CLB");
  }else ;

//getinterruptsource clears all triggered actions after returning value
readFrom(DEVICE, ADXL345_INT_SOURCE, 1,&interrupts);
 //double tap

Serial.print("zet, psi, phi :");
            Serial.print(zet);
            Serial.print(" ");
            Serial.print(psi);
            Serial.print(" ");
            Serial.print(phi);
            Serial.print(" || Direction : ");

if((interrupts >> ADXL345_DOUBLE_TAP) & 1){
      for (int i=0; i<1; i++) {
      Serial.println("Double tap");
      ntn.write(7);
      delay(100);
     }
  }else if((interrupts >> ADXL345_SINGLE_TAP) & 1){
    for (int i=0; i<8; i++) {
      Serial.println("tap");
      ntn.write(6);
      delay(100);
     }
  }else



detectdirection();
ntn.write(d);
Serial.println(" ");
Serial.print(" || Sending = ");
Serial.println(d);

//delay(500);
}


//Writes val to address register on device
void writeTo(int device, byte address, byte val) {
  Wire.beginTransmission(device); //start transmission to device
  Wire.write(address);        // send register address
  Wire.write(val);        // send value to write
  Wire.endTransmission(); //end transmission
}
//reads num bytes starting from address register on device in to buff array
void readFrom(int device, byte address, int num, byte buff[]) {
  Wire.beginTransmission(device); //start transmission to device k
  Wire.write(address);        //sends address to read from
  Wire.endTransmission(); //end transmission
  Wire.beginTransmission(device); //start transmission to device
  Wire.requestFrom(device, num);    // request 6 bytes from device
  int i = 0;
  while(Wire.available())    //device may send less than requested (abnormal)
  {
    buff[i] = Wire.read(); // receive a byte
```

```
    i++;
  }
  Wire.endTransmission(); //end transmission
}
void setRegisterBit(byte regAdress, int bitPos, bool state) {
  readFrom(DEVICE,regAdress,1,&b);
  if (state){
    b |= (1 << bitPos);  // forces nth bit of _b to be 1. all other bits left
alone.
  }
  else {
    b &= ~(1 << bitPos); // forces nth bit of _b to be 0. all other bits left
alone.
  }
  writeTo(DEVICE,regAdress,b);
}
void detectdirection(){

if( zet>=zetL&&phi>=phiL &&abs(zet)>abs(psi)&&abs(phi)>abs(psi))
        {

        d=5;
       Serial.print("LEFT");


  }
  else if(zet<=zetR&&phi>=phiR&&abs(zet)>abs(psi)&&abs(phi)>abs(psi))
  {
         d=4;
       Serial.print("RIGHT");
  }
  else if(psi<=psiF&&phi>=phiF&&abs(psi)>abs(zet)&&abs(phi)>abs(zet))
  {

      d=2;
     Serial.print("FORWARD");

  }
  else if(psi>=psiB&&phi>=phiB&&abs(psi)>abs(zet)&&abs(phi)>abs(zet))
   {

      d=3;
      Serial.print("BACKWARD");
  }
  else{

      d=1;
      Serial.print("STAY");

    }
  }
```

ภาคผนวก ข

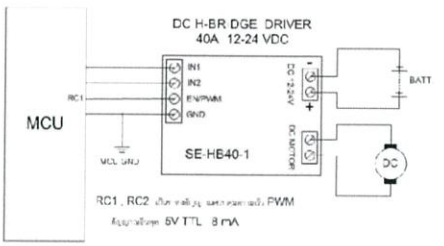
เอกสารที่เกี่ยวข้อง

บอร์ดขับมอเตอร์ดีซี แบบ H-Bridge รุ่น SE-HB40-1



<u>รายละเอียดทางเทคนิค</u>

1. Output : Single motor driver
   - Motor DC Supply 12-24 V 40A (Max.)
   - Full-Complementary Power MOSFET Driver
     With ultra-fast reverse recovery protection diodes
2. Input :
   - Full Opto-isolated input interface signals
   - 5V 8 mA TTL – Level
3. Drive Mode : independently with :
   - ON – OFF Control
   - Direction Control
   - Speed Control (PWM Drives)
4. PWM Frequency : 400 Hz - 1000 Hz (800 Hz Recommend)



EN/PWM	IN1	IN2	การทำงานของมอเตอร์
0V	X	X	Free Run Stop (หยุดเมื่อหมดแรงเฉื่อย)
5V	0V	5V	หมุนเดินหน้า
5V	5V	0V	หมุนกลับทาง
5V	5V	5V	Fast Stop หรือ Brake
5V	0V	0V	Fast Stop หรือ Brake

หากมีปัญหาเกี่ยวกับการใช้งาน ติดต่อฝ่ายบริการ 081-2535810

หากมีการขับเกินกำลังจนมอสเฟทพัง ให้ตรวจสอบโดยการใช้มิเตอร์วัดการลัดวงจรของมอสเฟท ทั้ง 4 ตัว หากตัวใดเสียขั้วมอสเฟทจะลัดวงจร ให้เปลี่ยนมอสเฟทได้เลย วงจรส่วนอื่นปลอดภัย ไม่จำเป็นต้องตรวจสอบ

การเปลี่ยนมอสเฟท เบอร์ IRF4905 ไม่มีจำหน่าย ให้ใช้เบอร์ IRF9540 แทน
เบอร์ IRFZ44 มีจำหน่าย หรือให้ใช้เบอร์ HUF75639P3
IRF3710PBF จะทำให้ทนแรงดันได้ดีกว่า

ANALOG DEVICES

3-Axis, ±2 g/±4 g/±8 g/±16 g Digital Accelerometer

ADXL345

FEATURES

- Ultralow power: as low as 23 µA in measurement mode and 0.1 µA in standby mode at V_S = 2.5 V (typical)
- Power consumption scales automatically with bandwidth
- User-selectable resolution
  - Fixed 10-bit resolution
  - Full resolution, where resolution increases with *g* range, up to 13-bit resolution at ±16 *g* (maintaining 4 mg/LSB scale factor in all *g* ranges)
- Patent pending, embedded memory management system with FIFO technology minimizes host processor load
- Single tap/double tap detection
- Activity/inactivity monitoring
- Free-fall detection
- Supply voltage range: 2.0 V to 3.6 V
- I/O voltage range: 1.7 V to V_S
- SPI (3- and 4-wire) and I²C digital interfaces
- Flexible interrupt modes mappable to either interrupt pin
- Measurement ranges selectable via serial command
- Bandwidth selectable via serial command
- Wide temperature range (−40°C to +85°C)
- 10,000 *g* shock survival
- Pb free/RoHS compliant
- Small and thin: 3 mm × 5 mm × 1 mm LGA package

APPLICATIONS

- Handsets
- Medical instrumentation
- Gaming and pointing devices
- Industrial instrumentation
- Personal navigation devices
- Hard disk drive (HDD) protection

GENERAL DESCRIPTION

The ADXL345 is a small, thin, ultralow power, 3-axis accelerometer with high resolution (13-bit) measurement at up to ±16 *g*. Digital output data is formatted as 16-bit twos complement and is accessible through either a SPI (3- or 4-wire) or I²C digital interface.

The ADXL345 is well suited for mobile device applications. It measures the static acceleration of gravity in tilt-sensing applications, as well as dynamic acceleration resulting from motion or shock. Its high resolution (3.9 mg/LSB) enables measurement of inclination changes less than 1.0°.

Several special sensing functions are provided. Activity and inactivity sensing detect the presence or lack of motion by comparing the acceleration on any axis with user-set thresholds. Tap sensing detects single and double taps in any direction. Free-fall sensing detects if the device is falling. These functions can be mapped individually to either of two interrupt output pins. An integrated, patent pending memory management system with a 32-level first in, first out (FIFO) buffer can be used to store data to minimize host processor activity and lower overall system power consumption.

Low power modes enable intelligent motion-based power management with threshold sensing and active acceleration measurement at extremely low power dissipation.

The ADXL345 is supplied in a small, thin, 3 mm × 5 mm × 1 mm, 14-lead, plastic package.

FUNCTIONAL BLOCK DIAGRAM

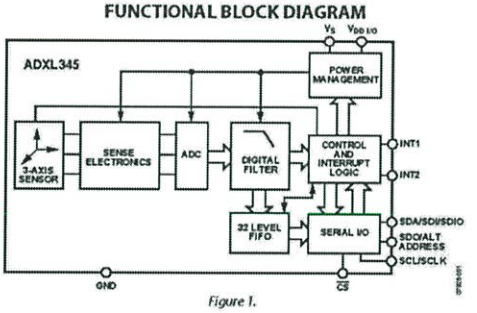


Figure 1.

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.
Tel: 781.329.4700 www.analog.com
Fax: 781.461.3113

SPECIFICATIONS

T_A = 25°C, V_S = 2.5 V, $V_{DD\ I/O}$ = 1.8 V, acceleration = 0 *g*, C_S = 10 µF tantalum, $C_{I/O}$ = 0.1 µF, output data rate (ODR) = 800 Hz, unless otherwise noted. All minimum and maximum specifications are guaranteed. Typical specifications are not guaranteed.

Table 1.

Parameter	Test Conditions	Min	Typ[1]	Max	Unit
SENSOR INPUT	Each axis				
Measurement Range	User selectable		±2, ±4, ±8, ±16		*g*
Nonlinearity	Percentage of full scale		±0.5		%
Inter-Axis Alignment Error			±0.1		Degrees
Cross-Axis Sensitivity[2]			±1		%
OUTPUT RESOLUTION	Each axis				
All *g* Ranges	10-bit resolution		10		Bits
±2 *g* Range	Full resolution		10		Bits
±4 *g* Range	Full resolution		11		Bits
±8 *g* Range	Full resolution		12		Bits
±16 *g* Range	Full resolution		13		Bits
SENSITIVITY	Each axis				
Sensitivity at X_{OUT}, Y_{OUT}, Z_{OUT}	All *g*-ranges, full resolution	230	256	282	LSB/*g*
	±2 *g*, 10-bit resolution	230	256	282	LSB/*g*
	±4 *g*, 10-bit resolution	115	128	141	LSB/*g*
	±8 *g*, 10-bit resolution	57	64	71	LSB/*g*
	±16 *g*, 10-bit resolution	29	32	35	LSB/*g*
Sensitivity Deviation from Ideal	All *g*-ranges		±1.0		%
Scale Factor at X_{OUT}, Y_{OUT}, Z_{OUT}	All *g*-ranges, full resolution	3.5	3.9	4.3	mg/LSB
	±2 *g*, 10-bit resolution	3.5	3.9	4.3	mg/LSB
	±4 *g*, 10-bit resolution	7.1	7.8	8.7	mg/LSB
	±8 *g*, 10-bit resolution	14.1	15.6	17.5	mg/LSB
	±16 *g*, 10-bit resolution	28.6	31.2	34.5	mg/LSB
Sensitivity Change Due to Temperature			±0.01		%/°C
0 *g* OFFSET	Each axis				
0 *g* Output for X_{OUT}, Y_{OUT}		−150	0	+150	mg
0 *g* Output for Z_{OUT}		−250	0	+250	mg
0 *g* Output Deviation from Ideal, X_{OUT}, Y_{OUT}			±35		mg
0 *g* Output Deviation from Ideal, Z_{OUT}			±40		mg
0 *g* Offset vs. Temperature for X-, Y-Axes			±0.4		mg/°C
0 *g* Offset vs. Temperature for Z-Axis			±0.8		mg/°C
NOISE					
X-, Y-Axes	ODR = 100 Hz for ±2 *g*, 10-bit resolution or all *g*-ranges, full resolution		0.75		LSB rms
Z-Axis	ODR = 100 Hz for ±2 *g*, 10-bit resolution or all *g*-ranges, full resolution		1.1		LSB rms
OUTPUT DATA RATE AND BANDWIDTH	User selectable				
Output Data Rate (ODR)[3, 4, 5]		0.1		3200	Hz
SELF-TEST[6]					
Output Change in X-Axis		0.20		2.10	*g*
Output Change in Y-Axis		−2.10		−0.20	*g*
Output Change in Z-Axis		0.30		3.40	*g*
POWER SUPPLY					
Operating Voltage Range (V_S)		2.0	2.5	3.6	V
Interface Voltage Range ($V_{DD\ I/O}$)		1.7	1.8	V_S	V
Supply Current	ODR ≥ 100 Hz		140		µA
	ODR < 10 Hz		30		µA
Standby Mode Leakage Current			0.1		µA
Turn-On and Wake-Up Time[7]	ODR = 3200 Hz		1.4		ms

ADXL345

Parameter	Test Conditions	Min	Typ[1]	Max	Unit
TEMPERATURE					
Operating Temperature Range		–40		+85	°C
WEIGHT					
Device Weight			30		mg

[1] The typical specifications shown are for at least 68% of the population of parts and are based on the worst case of mean ±1 σ, except for 0 *g* output and sensitivity, which represents the target value. For 0 *g* offset and sensitivity, the deviation from the ideal describes the worst case of mean ±1 σ.
[2] Cross-axis sensitivity is defined as coupling between any two axes.
[3] Bandwidth is the −3 dB frequency and is half the output data rate, bandwidth = ODR/2.
[4] The output format for the 3200 Hz and 1600 Hz ODRs is different than the output format for the remaining ODRs. This difference is described in the Data Formatting of Upper Data Rates section.
[5] Output data rates below 6.25 Hz exhibit additional offset shift with increased temperature, depending on selected output data rate. Refer to the Offset Performance at Lowest Data Rates section for details.
[6] Self-test change is defined as the output (*g*) when the SELF_TEST bit = 1 (in the DATA_FORMAT register, Address 0x31) minus the output (*g*) when the SELF_TEST bit = 0. Due to device filtering, the output reaches its final value after 4 × τ when enabling or disabling self-test, where τ = 1/(data rate). The part must be in normal power operation (LOW_POWER bit = 0 in the BW_RATE register, Address 0x2C) for self-test to operate correctly.
[7] Turn-on and wake-up times are determined by the user-defined bandwidth. At a 100 Hz data rate, the turn-on and wake-up times are each approximately 11.1 ms. For other data rates, the turn-on and wake-up times are each approximately τ + 1.1 in milliseconds, where τ = 1/(data rate).

PIN CONFIGURATION AND FUNCTION DESCRIPTIONS

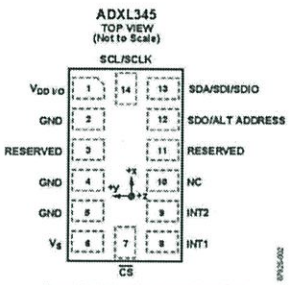


Figure 3. Pin Configuration (Top View)

Table 5. Pin Function Descriptions

Pin No.	Mnemonic	Description
1	$V_{DD\ I/O}$	Digital Interface Supply Voltage.
2	GND	This pin must be connected to ground.
3	RESERVED	Reserved. This pin must be connected to V_S or left open.
4	GND	This pin must be connected to ground.
5	GND	This pin must be connected to ground.
6	V_S	Supply Voltage.
7	$\overline{CS}$	Chip Select.
8	INT1	Interrupt 1 Output.
9	INT2	Interrupt 2 Output.
10	NC	Not Internally Connected.
11	RESERVED	Reserved. This pin must be connected to ground or left open.
12	SDO/ALT ADDRESS	Serial Data Output (SPI 4-Wire)/Alternate I²C Address Select (I²C).
13	SDA/SDI/SDIO	Serial Data (I²C)/Serial Data Input (SPI 4-Wire)/Serial Data Input and Output (SPI 3-Wire).
14	SCL/SCLK	Serial Communications Clock. SCL is the clock for I²C, and SCLK is the clock for SPI.

I²C

With $\overline{CS}$ tied high to $V_{DD\ I/O}$, the ADXL345 is in I²C mode, requiring a simple 2-wire connection, as shown in Figure 39. The ADXL345 conforms to the *UM10204 I²C-Bus Specification and User Manual*, Rev. 03—19 June 2007, available from NXP Semiconductor. It supports standard (100 kHz) and fast (400 kHz) data transfer modes if the bus parameters given in Table 11 and Table 12 are met. Single- or multiple-byte reads/writes are supported, as shown in Figure 40. With the ALT ADDRESS pin high, the 7-bit I²C address for the device is 0x1D, followed by the R/$\overline{W}$ bit. This translates to 0x3A for a write and 0x3B for a read. An alternate I²C address of 0x53 (followed by the R/$\overline{W}$ bit) can be chosen by grounding the ALT ADDRESS pin (Pin 12). This translates to 0xA6 for a write and 0xA7 for a read.

There are no internal pull-up or pull-down resistors for any unused pins; therefore, there is no known state or default state for the $\overline{CS}$ or ALT ADDRESS pin if left floating or unconnected. It is required that the $\overline{CS}$ pin be connected to $V_{DD\ I/O}$ and that the ALT ADDRESS pin be connected to either $V_{DD\ I/O}$ or GND when using I²C.

Due to communication speed limitations, the maximum output data rate when using 400 kHz I²C is 800 Hz and scales linearly with a change in the I²C communication speed. For example, using I²C at 100 kHz would limit the maximum ODR to 200 Hz. Operation at an output data rate above the recommended maximum may result in undesirable effect on the acceleration data, including missing samples or additional noise.

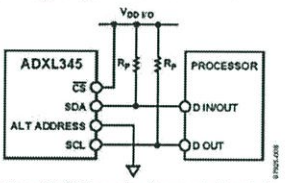


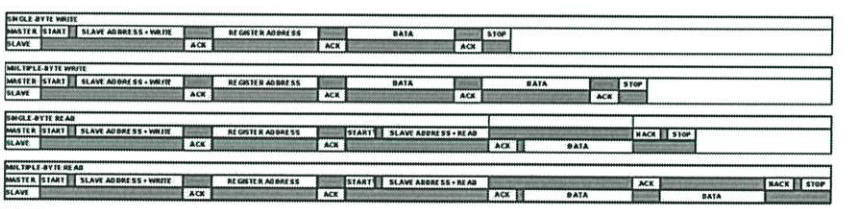
Figure 39. I²C Connection Diagram (Address 0x53)

If other devices are connected to the same I²C bus, the nominal operating voltage level of these other devices cannot exceed $V_{DD\ I/O}$ by more than 0.3 V. External pull-up resistors, R_P, are necessary for proper I²C operation. Refer to the *UM10204 I²C-Bus Specification and User Manual*, Rev. 03—19 June 2007, when selecting pull-up resistor values to ensure proper operation.

Table 11. I²C Digital Input/Output

Parameter	Test Conditions	Limit[1] Min	Limit[1] Max	Unit
Digital Input				
Low Level Input Voltage (V_{IL})			0.3 × $V_{DD\ I/O}$	V
High Level Input Voltage (V_{IH})		0.7 × $V_{DD\ I/O}$		V
Low Level Input Current (I_{IL})	$V_{IN} = V_{DD\ I/O}$		0.1	µA
High Level Input Current (I_{IH})	$V_{IN} = 0$ V	−0.1		µA
Digital Output				
Low Level Output Voltage (V_{OL})	$V_{DD\ I/O} < 2$ V, $I_{OL} = 3$ mA		0.2 × $V_{DD\ I/O}$	V
	$V_{DD\ I/O} \geq 2$ V, $I_{OL} = 3$ mA		400	mV
Low Level Output Current (I_{OL})	$V_{OL} = V_{OL,\ max}$	3		mA
Pin Capacitance	$f_{IN} = 1$ MHz, $V_{IN} = 2.5$ V		8	pF

[1] Limits based on characterization results; not production tested.



NOTES
1. THIS START IS EITHER A RESTART OR A STOP FOLLOWED BY A START.
2. THE SHADED AREAS REPRESENT WHEN THE DEVICE IS LISTENING.

Figure 40. I²C Device Addressing

Table 12. I²C Timing (T_A = 25°C, V_S = 2.5 V, $V_{DD\ I/O}$ = 1.8 V)

Parameter	Limit[1, 2] Min	Max	Unit	Description
f_{SCL}		400	kHz	SCL clock frequency
t_1	2.5		µs	SCL cycle time
t_2	0.6		µs	t_{HIGH}, SCL high time
t_3	1.3		µs	t_{LOW}, SCL low time
t_4	0.6		µs	$t_{HD,\ STA}$, start/repeated start condition hold time
t_5	100		ns	$t_{SU,\ DAT}$, data setup time
t_6[3, 4, 5, 6]	0	0.9	µs	$t_{HD,\ DAT}$, data hold time
t_7	0.6		µs	$t_{SU,\ STA}$, setup time for repeated start
t_8	0.6		µs	$t_{SU,\ STO}$, stop condition setup time
t_9	1.3		µs	t_{BUF}, bus-free time between a stop condition and a start condition
t_{10}		300	ns	t_R, rise time of both SCL and SDA when receiving
	0		ns	t_R, rise time of both SCL and SDA when receiving or transmitting
t_{11}		250	ns	t_F, fall time of SDA when receiving
		300	ns	t_F, fall time of both SCL and SDA when transmitting
	20 + 0.1 C_b[7]		ns	t_F, fall time of both SCL and SDA when transmitting or receiving
C_b		400	pF	Capacitive load for each bus line

[1] Limits based on characterization results, with f_{SCL} = 400 kHz and a 3 mA sink current; not production tested.
[2] All values referred to the V_{IH} and the V_{IL} levels given in Table 11.
[3] t_6 is the data hold time that is measured from the falling edge of SCL. It applies to data in transmission and acknowledge.
[4] A transmitting device must internally provide an output hold time of at least 300 ns for the SDA signal (with respect to $V_{IH(min)}$ of the SCL signal) to bridge the undefined region of the falling edge of SCL.
[5] The maximum t_6 value must be met only if the device does not stretch the low period (t_5) of the SCL signal.
[6] The maximum value for t_6 is a function of the clock low time (t_3), the clock rise time (t_{10}), and the minimum data setup time ($t_{5(min)}$). This value is calculated as $t_{6(max)} = t_3 - t_{10} - t_{5(min)}$.
[7] C_b is the total capacitance of one bus line in picofarads.

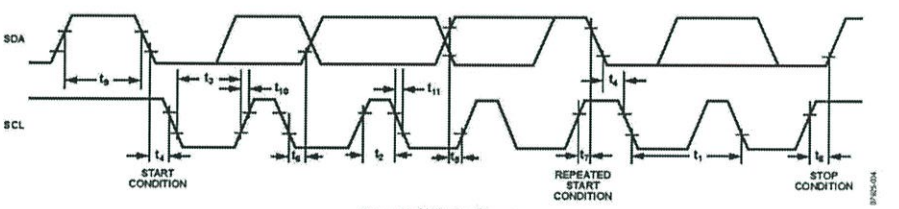


Figure 41. I²C Timing Diagram

INTERRUPTS

The ADXL345 provides two output pins for driving interrupts: INT1 and INT2. Both interrupt pins are push-pull, low impedance pins with output specifications shown in Table 13. The default configuration of the interrupt pins is active high. This can be changed to active low by setting the INT_INVERT bit in the DATA_FORMAT (Address 0x31) register. All functions can be used simultaneously, with the only limiting feature being that some functions may need to share interrupt pins.

Interrupts are enabled by setting the appropriate bit in the INT_ENABLE register (Address 0x2E) and are mapped to either the INT1 pin or the INT2 pin based on the contents of the INT_MAP register (Address 0x2F). When initially configuring the interrupt pins, it is recommended that the functions and interrupt mapping be done before enabling the interrupts. When changing the configuration of an interrupt, it is recommended that the interrupt be disabled first, by clearing the bit corresponding to that function in the INT_ENABLE register, and then the function be reconfigured before enabling the interrupt again. Configuration of the functions while the interrupts are disabled helps to prevent the accidental generation of an interrupt before desired.

The interrupt functions are latched and cleared by either reading the data registers (Address 0x32 to Address 0x37) until the interrupt condition is no longer valid for the data-related interrupts or by reading the INT_SOURCE register (Address 0x30) for the remaining interrupts. This section describes the interrupts that can be set in the INT_ENABLE register and monitored in the INT_SOURCE register.

DATA_READY

The DATA_READY bit is set when new data is available and is cleared when no new data is available.

SINGLE_TAP

The SINGLE_TAP bit is set when a single acceleration event that is greater than the value in the THRESH_TAP register (Address 0x1D) occurs for less time than is specified in the DUR register (Address 0x21).

DOUBLE_TAP

The DOUBLE_TAP bit is set when two acceleration events that are greater than the value in the THRESH_TAP register (Address 0x1D) occur for less time than is specified in the DUR register (Address 0x21), with the second tap starting after the time specified by the latent register (Address 0x22) but within the time specified in the window register (Address 0x23). See the Tap Detection section for more details.

Activity

The activity bit is set when acceleration greater than the value stored in the THRESH_ACT register (Address 0x24) is experienced on any participating axis, set by the ACT_INACT_CTL register (Address 0x27).

Inactivity

The inactivity bit is set when acceleration of less than the value stored in the THRESH_INACT register (Address 0x25) is experienced for more time than is specified in the TIME_INACT register (Address 0x26) on all participating axes, as set by the ACT_INACT_CTL register (Address 0x27). The maximum value for TIME_INACT is 255 sec.

FREE_FALL

The FREE_FALL bit is set when acceleration of less than the value stored in the THRESH_FF register (Address 0x28) is experienced for more time than is specified in the TIME_FF register (Address 0x29) on all axes (logical AND). The FREE_FALL interrupt differs from the inactivity interrupt as follows: all axes always participate and are logically AND'ed, the timer period is much smaller (1.28 sec maximum), and the mode of operation is always dc-coupled.

Watermark

The watermark bit is set when the number of samples in FIFO equals the value stored in the samples bits (Register FIFO_CTL, Address 0x38). The watermark bit is cleared automatically when FIFO is read, and the content returns to a value below the value stored in the samples bits.

Table 13. Interrupt Pin Digital Output

Parameter	Test Conditions	Limit[1] Min	Limit[1] Max	Unit
Digital Output				
Low Level Output Voltage (V_{OL})	I_{OL} = 300 µA		0.2 × $V_{DD\ I/O}$	V
High Level Output Voltage (V_{OH})	I_{OH} = −150 µA	0.8 × $V_{DD\ I/O}$		V
Low Level Output Current (I_{OL})	V_{OL} = $V_{OL,max}$	300		µA
High Level Output Current (I_{OH})	V_{OH} = $V_{OH,min}$		−150	µA
Pin Capacitance	f_{IN} = 1 MHz, V_{IN} = 2.5 V		8	pF
Rise/Fall Time				
Rise Time (t_R)[2]	C_{LOAD} = 150 pF		210	ns
Fall Time (t_F)[3]	C_{LOAD} = 150 pF		150	ns

[1] Limits based on characterization results, not production tested.
[2] Rise time is measured as the transition time from $V_{OL,max}$ to $V_{OH,min}$ of the interrupt pin.
[3] Fall time is measured as the transition time from $V_{OH,min}$ to $V_{OL,max}$ of the interrupt pin.

Overrun

The overrun bit is set when new data replaces unread data. The precise operation of the overrun function depends on the FIFO mode. In bypass mode, the overrun bit is set when new data replaces unread data in the DATAX, DATAY, and DATAZ registers (Address 0x32 to Address 0x37). In all other modes, the overrun bit is set when FIFO is filled. The overrun bit is automatically cleared when the contents of FIFO are read.

FIFO

The ADXL345 contains patent pending technology for an embedded memory management system with 32-level FIFO that can be used to minimize host processor burden. This buffer has four modes: bypass, FIFO, stream, and trigger (see FIFO Modes). Each mode is selected by the settings of the FIFO_MODE bits (Bits[D7:D6]) in the FIFO_CTL register (Address 0x38).

Bypass Mode

In bypass mode, FIFO is not operational and, therefore, remains empty.

FIFO Mode

In FIFO mode, data from measurements of the x-, y-, and z-axes are stored in FIFO. When the number of samples in FIFO equals the level specified in the samples bits of the FIFO_CTL register (Address 0x38), the watermark interrupt is set. FIFO continues accumulating samples until it is full (32 samples from measurements of the x-, y-, and z-axes) and then stops collecting data. After FIFO stops collecting data, the device continues to operate; therefore, features such as tap detection can be used after FIFO is full. The watermark interrupt continues to occur until the number of samples in FIFO is less than the value stored in the samples bits of the FIFO_CTL register.

Stream Mode

In stream mode, data from measurements of the x-, y-, and z-axes are stored in FIFO. When the number of samples in FIFO equals the level specified in the samples bits of the FIFO_CTL register (Address 0x38), the watermark interrupt is set. FIFO continues accumulating samples and holds the latest 32 samples from measurements of the x-, y-, and z-axes, discarding older data as new data arrives. The watermark interrupt continues occurring until the number of samples in FIFO is less than the value stored in the samples bits of the FIFO_CTL register.

Trigger Mode

In trigger mode, FIFO accumulates samples, holding the latest 32 samples from measurements of the x-, y-, and z-axes. After a trigger event occurs and an interrupt is sent to the INT1 or INT2 pin (determined by the trigger bit in the FIFO_CTL register), FIFO keeps the last n samples (where n is the value specified by the samples bits in the FIFO_CTL register) and then operates in FIFO mode, collecting new samples only when FIFO is not full. A delay of at least 5 µs should be present between the trigger event occurring and the start of reading data from the FIFO to allow the FIFO to discard and retain the necessary samples. Additional trigger events cannot be recognized until the trigger mode is reset. To reset the trigger mode, set the device to bypass mode and then set the device back to trigger mode. Note that the FIFO data should be read first because placing the device into bypass mode clears FIFO.

Retrieving Data from FIFO

The FIFO data is read through the DATAX, DATAY, and DATAZ registers (Address 0x32 to Address 0x37). When the FIFO is in FIFO, stream, or trigger mode, reads to the DATAX, DATAY, and DATAZ registers read data stored in the FIFO. Each time data is read from the FIFO, the oldest x-, y-, and z-axes data are placed into the DATAX, DATAY and DATAZ registers.

If a single-byte read operation is performed, the remaining bytes of data for the current FIFO sample are lost. Therefore, all axes of interest should be read in a burst (or multiple-byte) read operation. To ensure that the FIFO has completely popped (that is, that new data has completely moved into the DATAX, DATAY, and DATAZ registers), there must be at least 5 µs between the end of reading the data registers and the start of a new read of the FIFO or a read of the FIFO_STATUS register (Address 0x39). The end of reading a data register is signified by the transition from Register 0x37 to Register 0x38 or by the $\overline{CS}$ pin going high.

For SPI operation at 1.6 MHz or less, the register addressing portion of the transmission is a sufficient delay to ensure that the FIFO has completely popped. For SPI operation greater than 1.6 MHz, it is necessary to deassert the $\overline{CS}$ pin to ensure a total delay of 5 µs; otherwise, the delay is not sufficient. The total delay necessary for 5 MHz operation is at most 3.4 µs. This is not a concern when using I²C mode because the communication rate is low enough to ensure a sufficient delay between FIFO reads.

APPLICATIONS INFORMATION

POWER SUPPLY DECOUPLING

A 1 µF tantalum capacitor (C_S) at V_S and a 0.1 µF ceramic capacitor ($C_{I/O}$) at $V_{DD\ I/O}$ placed close to the ADXL345 supply pins is recommended to adequately decouple the accelerometer from noise on the power supply. If additional decoupling is necessary, a resistor or ferrite bead, no larger than 100 Ω, in series with V_S may be helpful. Additionally, increasing the bypass capacitance on V_S to a 10 µF tantalum capacitor in parallel with a 0.1 µF ceramic capacitor may also improve noise.

Care should be taken to ensure that the connection from the ADXL345 ground to the power supply ground has low impedance because noise transmitted through ground has an effect similar to noise transmitted through V_S. It is recommended that V_S and $V_{DD\ I/O}$ be separate supplies to minimize digital clocking noise on the V_S supply. If this is not possible, additional filtering of the supplies, as previously mentioned, may be necessary.

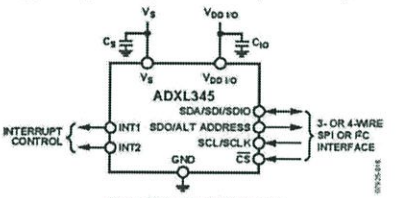


Figure 43. Application Diagram

MECHANICAL CONSIDERATIONS FOR MOUNTING

The ADXL345 should be mounted on the PCB in a location close to a hard mounting point of the PCB to the case. Mounting the ADXL345 at an unsupported PCB location, as shown in Figure 44, may result in large, apparent measurement errors due to undampened PCB vibration. Locating the accelerometer near a hard mounting point ensures that any PCB vibration at the accelerometer is above the accelerometer's mechanical sensor resonant frequency and, therefore, effectively invisible to the accelerometer. Multiple mounting points, close to the sensor, and/or a thicker PCB also help to reduce the effect of system resonance on the performance of the sensor.

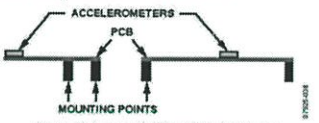


Figure 44. Incorrectly Placed Accelerometers

TAP DETECTION

The tap interrupt function is capable of detecting either single or double taps. The following parameters are shown in Figure 45 for a valid single and valid double tap event:

- The tap detection threshold is defined by the THRESH_TAP register (Address 0x1D).
- The maximum tap duration time is defined by the DUR register (Address 0x21).
- The tap latency time is defined by the latent register (Address 0x22) and is the waiting period from the end of the first tap until the start of the time window, when a second tap can be detected, which is determined by the value in the window register (Address 0x23).
- The interval after the latency time (set by the latent register) is defined by the window register. Although a second tap must begin after the latency time has expired, it need not finish before the end of the time defined by the window register.

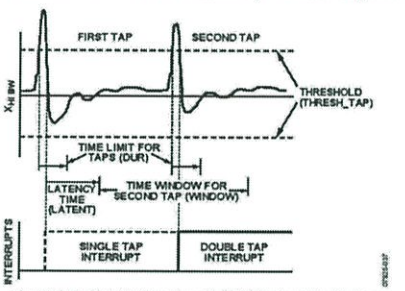


Figure 45. Tap Interrupt Function with Valid Single and Double Taps

If only the single tap function is in use, the single tap interrupt is triggered when the acceleration goes below the threshold, as long as DUR has not been exceeded. If both single and double tap functions are in use, the single tap interrupt is triggered when the double tap event has been either validated or invalidated.

Several events can occur to invalidate the second tap of a double tap event. First, if the suppress bit in the TAP_AXES register (Address 0x2A) is set, any acceleration spike above the threshold during the latency time (set by the latent register) invalidates the double tap detection, as shown in Figure 46.

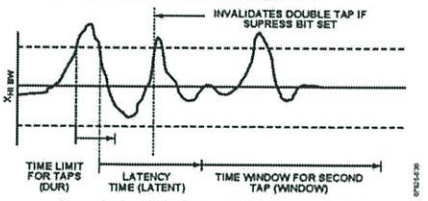


Figure 46. Double Tap Event Invalid Due to High g Event When the Suppress Bit Is Set

A double tap event can also be invalidated if acceleration above the threshold is detected at the start of the time window for the second tap (set by the window register). This results in an invalid double tap at the start of this window, as shown in Figure 47. Additionally, a double tap event can be invalidated if an acceleration exceeds the time limit for taps (set by the DUR register), resulting in an invalid double tap at the end of the DUR time limit for the second tap event, also shown in Figure 47.

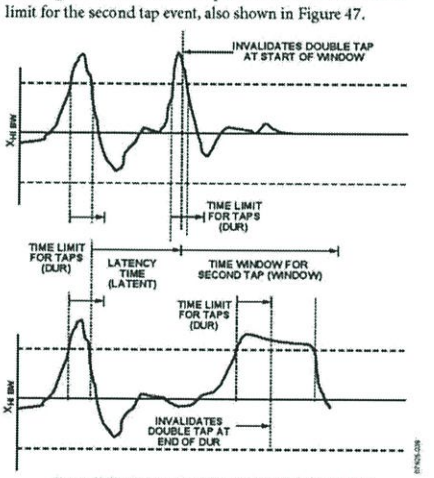


Figure 47. Tap Interrupt Function with Invalid Double Taps

Single taps, double taps, or both can be detected by setting the respective bits in the INT_ENABLE register (Address 0x2E). Control over participation of each of the three axes in single tap/double tap detection is exerted by setting the appropriate bits in the TAP_AXES register (Address 0x2A). For the double tap function to operate, both the latent and window registers must be set to a nonzero value.

Every mechanical system has somewhat different single tap/double tap responses based on the mechanical characteristics of the system. Therefore, some experimentation with values for the DUR, latent, window, and THRESH_TAP registers is required. In general, a good starting point is to set the DUR register to a value greater than 0x10 (10 ms), the latent register to a value greater than 0x10 (20 ms), the window register to a value greater than 0x40 (80 ms), and the THRESH_TAP register to a value greater than 0x30 (3 *g*). Setting a very low value in the latent, window, or THRESH_TAP register may result in an unpredictable response due to the accelerometer picking up echoes of the tap inputs.

After a tap interrupt has been received, the first axis to exceed the THRESH_TAP level is reported in the ACT_TAP_STATUS register (Address 0x2B). This register is never cleared but is overwritten with new data.

THRESHOLD

The lower output data rates are achieved by decimating a common sampling frequency inside the device. The activity, free-fall, and single tap/double tap detection functions without improved tap enabled are performed using undecimated data. Because the bandwidth of the output data varies with the data rate and is lower than the bandwidth of the undecimated data, the high frequency and high *g* data that is used to determine activity, free-fall, and single tap/double tap events may not be present if the output of the accelerometer is examined. This may result in functions triggering when acceleration data does not appear to meet the conditions set by the user for the corresponding function.

LINK MODE

The function of the link bit is to reduce the number of activity interrupts that the processor must service by setting the device to look for activity only after inactivity. For proper operation of this feature, the processor must still respond to the activity and inactivity interrupts by reading the INT_SOURCE register (Address 0x30) and, therefore, clearing the interrupts. If an activity interrupt is not cleared, the part cannot go into autosleep mode. The asleep bit in the ACT_TAP_STATUS register (Address 0x2B) indicates if the part is asleep.

SLEEP MODE VS. LOW POWER MODE

In applications where a low data rate and low power consumption is desired (at the expense of noise performance), it is recommended that low power mode be used. The use of low power mode preserves the functionality of the DATA_READY interrupt and the FIFO for postprocessing of the acceleration data. Sleep mode, while offering a low data rate and power consumption, is not intended for data acquisition.

However, when sleep mode is used in conjunction with the AUTO_SLEEP mode and the link mode, the part can automatically switch to a low power, low sampling rate mode when inactivity is detected. To prevent the generation of redundant inactivity interrupts, the inactivity interrupt is automatically disabled and activity is enabled. When the ADXL345 is in sleep mode, the host processor can also be placed into sleep mode or low power mode to save significant system power. When activity is detected, the accelerometer automatically switches back to the original data rate of the application and provides an activity interrupt that can be used to wake up the host processor. Similar to when inactivity occurs, detection of activity events is disabled and inactivity is enabled.

OFFSET CALIBRATION

Accelerometers are mechanical structures containing elements that are free to move. These moving parts can be very sensitive to mechanical stresses, much more so than solid-state electronics. The 0 *g* bias or offset is an important accelerometer metric because it defines the baseline for measuring acceleration. Additional stresses can be applied during assembly of a system containing an accelerometer. These stresses can come from, but are not limited to, component soldering, board stress during mounting, and application of any compounds on or over the component. If calibration is deemed necessary, it is recommended that calibration be performed after system assembly to compensate for these effects.

A simple method of calibration is to measure the offset while assuming that the sensitivity of the ADXL345 is as specified in Table 1. The offset can then be automatically accounted for by using the built-in offset registers. This results in the data acquired from the DATA registers already compensating for any offset.

In a no-turn or single-point calibration scheme, the part is oriented such that one axis, typically the z-axis, is in the 1 *g* field of gravity and the remaining axes, typically the x- and y-axis, are in a 0 *g* field. The output is then measured by taking the average of a series of samples. The number of samples averaged is a choice of the system designer, but a recommended starting point is 0.1 sec worth of data for data rates of 100 Hz or greater. This corresponds to 10 samples at the 100 Hz data rate. For data rates less than 100 Hz, it is recommended that at least 10 samples be averaged together. These values are stored as X_{0g}, Y_{0g}, and Z_{+1g} for the 0 *g* measurements on the x- and y-axis and the 1 *g* measurement on the z-axis, respectively.

The values measured for X_{0g} and Y_{0g} correspond to the x- and y-axis offset, and compensation is done by subtracting those values from the output of the accelerometer to obtain the actual acceleration:

$$X_{ACTUAL} = X_{MEAS} - X_{0g}$$

$$Y_{ACTUAL} = Y_{MEAS} - Y_{0g}$$

Because the z-axis measurement was done in a +1 *g* field, a no-turn or single-point calibration scheme assumes an ideal sensitivity, Sz for the z-axis. This is subtracted from Z_{+1g} to attain the z-axis offset, which is then subtracted from future measured values to obtain the actual value:

$$Z_{0g} = Z_{+1g} - S_Z$$

$$Z_{ACTUAL} = Z_{MEAS} - Z_{0g}$$

The ADXL345 can automatically compensate the output for offset by using the offset registers (Register 0x1E, Register 0x1F, and Register 0x20). These registers contain an 8-bit, twos complement value that is automatically added to all measured acceleration values, and the result is then placed into the DATA registers. Because the value placed in an offset register is additive, a negative value is placed into the register to eliminate a positive offset and vice versa for a negative offset. The register has a scale factor of 15.6 mg/LSB and is independent of the selected *g*-range.

As an example, assume that the ADXL345 is placed into full-resolution mode with a sensitivity of typically 256 LSB/*g*. The part is oriented such that the z-axis is in the field of gravity and x-, y-, and z-axis outputs are measured as +10 LSB, −13 LSB, and +9 LSB, respectively. Using the previous equations, X_{0g} is +10 LSB, Y_{0g} is −13 LSB, and Z_{0g} is +9 LSB. Each LSB of output in full-resolution is 3.9 mg or one-quarter of an LSB of the offset register. Because the offset register is additive, the 0 *g* values are negated and rounded to the nearest LSB of the offset register:

$$X_{OFFSET} = -Round(10/4) = -3 \text{ LSB}$$

$$Y_{OFFSET} = -Round(-13/4) = 3 \text{ LSB}$$

$$Z_{OFFSET} = -Round(9/4) = -2 \text{ LSB}$$

These values are programmed into the OFSX, OFSY, and OFXZ registers, respectively, as 0xFD, 0x03 and 0xFE. As with all registers in the ADXL345, the offset registers do not retain the value written into them when power is removed from the part. Power-cycling the ADXL345 returns the offset registers to their default value of 0x00.

Because the no-turn or single-point calibration method assumes an ideal sensitivity in the z-axis, any error in the sensitivity results in offset error. For instance, if the actual sensitivity was 250 LSB/*g* in the previous example, the offset would be 15 LSB, not 9 LSB. To help minimize this error, an additional measurement point can be used with the z-axis in a 0 *g* field and the 0 *g* measurement can be used in the Z_{ACTUAL} equation.

AXES OF ACCELERATION SENSITIVITY

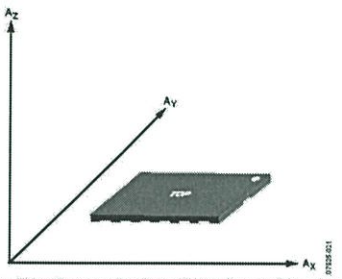


Figure 56. Axes of Acceleration Sensitivity (Corresponding Output Voltage Increases When Accelerated Along the Sensitive Axis)

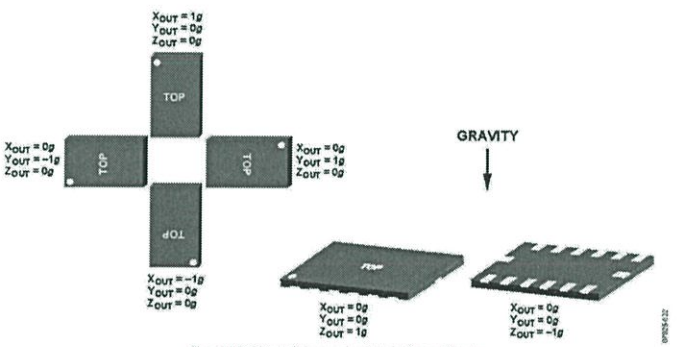


Figure 57. Output Response vs. Orientation to Gravity

ANALOG DEVICES

AN-1057
Application Note

One Technology Way • P.O. Box 9106 • Norwood, MA 02062-9106, U.S.A. • Tel: 781.329.4700 • Fax: 781.461.3113 • www.analog.com

Using an Accelerometer for Inclination Sensing

by Christopher J. Fisher

INTRODUCTION

One common method for determining the tilt or inclination of a system is to integrate the output of a gyroscope. Although this method is straightforward, error associated with null bias stability can quickly compound as the integration period is increased, causing an apparent rotation even when the device is stationary.

In some applications, where the net acceleration or force on a system over time is gravity, an accelerometer can be used to measure the static angle of tilt or inclination. Such applications include gaming, horizon detection in digital cameras, and detecting the heading of a device in industrial and medical applications.

The underlying assumption in inclination sensing with an accelerometer is that the only acceleration stimulus is that associated with gravity. In practice, signal processing can be performed on the signal output to remove high frequency content from the output signal, so some ac acceleration can be tolerated.

Inclination sensing uses the gravity vector and its projection on the axes of the accelerometer to determine the tilt angle. Because gravity is a dc acceleration, any forces that result in an additional dc acceleration corrupt the output signal and result in an incorrect calculation. Sources of dc acceleration include the period of time when a vehicle is accelerating at a constant rate and rotating devices that induce a centripetal acceleration on the accelerometer. In addition, rotating an accelerometer through gravity causes an apparent ac acceleration as the projection of gravity on the axes of interest changes. Any filtering of the acceleration signal before calculating the inclination affects how quickly the output settles to the new static value.

This application note discusses the basic principles for converting the output of an accelerometer to an angle of inclination. This discussion includes how to calculate the ideal inclination angle for a single-axis, dual-axis, or triple-axis solution. In addition, some basic information about calibration is included to reduce error from offset and sensitivity mismatch.

TILT/INCLINATION CALCULATION

Single-Axis Tilt Calculation

In applications where inclination sensing is needed only over a limited angle and with a somewhat coarse resolution, a single-axis device (or a single axis of a multiple-axis device) can be used.

For example, in Figure 1 a single axis (the x-axis in this example) is rotated through gravity. Because this approach uses only a single axis and requires the gravity vector, the calculated angle of inclination is accurate only when the device is oriented such that the x-axis is always in the plane of gravity. Any rotation about the other axes reduces the magnitude of the acceleration on the x-axis and results in error in the calculated angle of inclination.

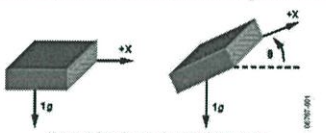


Figure 1. Single Axis Used for Tilt Sensing

Referring to basic trigonometry, the projection of the gravity vector on the x-axis produces an output acceleration equal to the sine of the angle between the accelerometer x-axis and the horizon. The horizon is typically taken to be the plane orthogonal to the gravity vector. For an ideal value of 1 *g* for gravity, the output acceleration is

$$A_{XOUT}\,[g] = 1\,g \times \sin(\theta) \qquad (1)$$

When using a single-axis solution, note that the sensitivity—that is, the change in output for some change in input—of the inclination calculation decreases as the angle between the horizon and the x-axis increases, approaching 0 as the angle approaches ±90°. This can be seen in Figure 2, where the output acceleration, in *g*, is plotted against the angle of inclination. Near ±90°, a large change in inclination angle results in a small change in output acceleration.

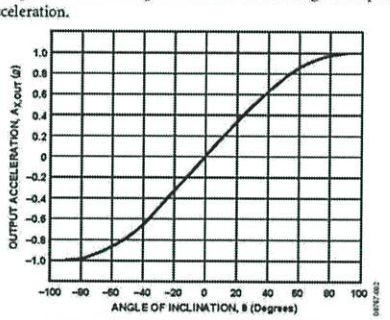


Figure 2. Output Acceleration vs. Angle of Inclination for Single-Axis Inclination Sensing

Because the inclination calculation is done digitally, the output acceleration is presented as a constant acceleration per least significant bit (LSB), or code, obtained either from an analog-to-digital converter (ADC) or directly from a digital output part. Because the output resolution is a constant acceleration, the resolution in degrees of inclination is variable, with the best resolution close to 0° and the worst resolution at ±90°.

Figure 3 and Figure 4 show the incremental sensitivity for 1° and 0.25° inclination angle steps. The incremental sensitivity is the output change, shown in mg, per inclination angle step, or

$$S\,[g] = 1\,g \times \left(\sin(N+P) - \sin(N)\right) \qquad (2)$$

where:
N is the current angle.
P is the step size.

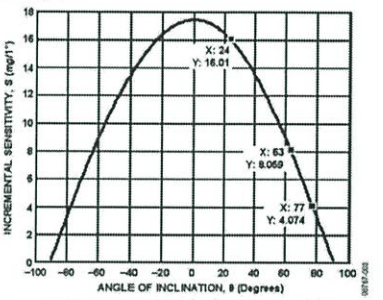


Figure 3. Incremental Inclination Sensitivity for 1° Steps

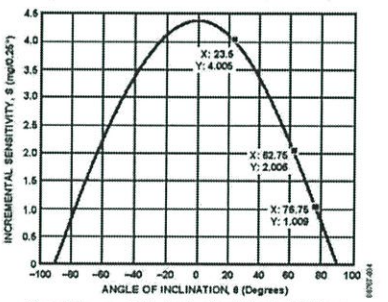


Figure 4. Incremental Inclination Sensitivity for 0.25° Steps

These curves can be used to determine the minimum necessary resolution when measuring the output acceleration in order to meet the desired inclination resolution for the entire range of an application. For example, designing for a maximum step size of 1°, a resolution of at least 8 mg/LSB is necessary for a range of ±63°. Similarly, to achieve a maximum step size of 0.25° for a range of ±63 requires a resolution of at least 2 mg/LSB. Note that, if enough dither is present, oversampling can be used to achieve better resolution.

Because the output of the accelerometer obeys a sinusoidal relationship as it is rotated through gravity, conversion from acceleration to angle is done using the inverse sine function.

$$\theta = \sin^{-1}\left(\frac{A_{X,OUT}\,[g]}{1\,g}\right) \quad (3)$$

where the inclination angle, θ, is in radians.

If a narrow range of inclination is required, a linear approximation can be used in place of the inverse sine function. The linear approximation relates to the approximation of sine for small angles.

$$\sin(\theta) \cong \theta, \quad \theta << 1 \quad (4)$$

where the inclination angle, θ, is in radians.

An additional scaling factor, k, can be included in the linear approximation for inclination angle, which allows the valid range for the approximation to be increased if the allowable error is increased.

$$\theta \cong k \times \left(\frac{A_{X,OUT}\,[g]}{1\,g}\right) \quad (5)$$

where the inclination angle, θ, is in radians.

Conversion to degrees is done by multiplying the result of Equation 5 by (180/π). Figure 5 shows a comparison between using the inverse sine function and the linear approximation with k equal to 1. As the magnitude of the inclination angle increases, the linear approximation begins to fail, and the calculated angle deviates from the actual angle.

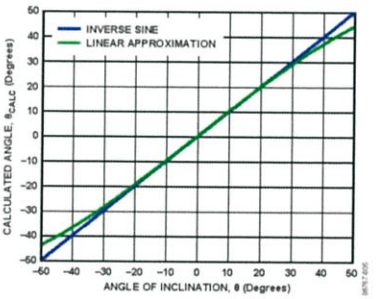


Figure 5. Comparison of Inverse Sine Function and Linear Approximation for Inclination Angle Calculation

Because the calculated angle is plotted against the actual angle of inclination, the linear approximation appears to bend near the ends. This is because the linear approximation is linear only when compared to the output acceleration and, as shown in Figure 2, the output acceleration behaves similarly as the actual angle of inclination is increased. However, the inverse sine function should produce an output that is one-to-one with the actual angle of inclination, causing the calculated angle to be a straight line when plotted against the actual angle of inclination.

As an example, if the desired resolution of inclination sensing is 1°, an error of ±0.5° is acceptable because it is below the rounding error of the calculation. If the error between the actual angle of inclination and the calculated angle of inclination is plotted for k equal to 1, as shown in Figure 6, the valid range for the linear approximation is only ±20°. If the scaling factor is adjusted such that the error is maximized but kept within the calculation rounding limits, the valid range of the linear approximation increases to greater than ±30°.

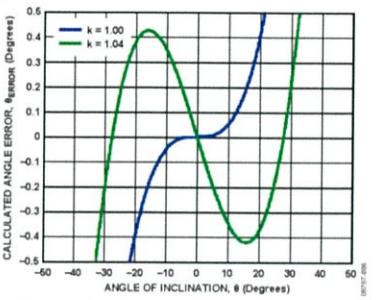


Figure 6. Calculated Angle Error for Different Scaling Factors

Dual-Axis Tilt Calculation

One limitation of single-axis inclination sensing is the need for a high resolution ADC or digital output to achieve a large range of valid inclination angles, as shown in Figure 3 and Figure 4. Another limitation is that a single-axis measurement cannot provide a 360° measurement, because the acceleration generated at an inclination of N° is the same as the acceleration generated at an inclination of 180° − N°. For some applications, this is acceptable, but for applications that require higher resolution or the ability to distinguish angles of inclination in a complete 360° arc, a second axis, as shown in Figure 7, or a second sensor is necessary. If a second sensor is used, it should be oriented such that the sensing axis of the second sensor is orthogonal to the sensing axis of the first sensor.

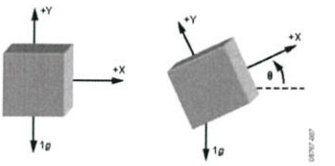


Figure 7. Two Axes Used for Tilt Sensing

There are three major benefits to including a second axis in determining the angle of inclination. These benefits are described in the following sections.

Constant Sensitivity

The first major benefit of using a second axis is due to the orthogonality of the axes. As in the single-axis solution, the acceleration detected by the x-axis is proportional to the sine of the angle of inclination. The y-axis acceleration, due to the orthogonality, is proportional to the cosine of the angle of inclination (see Figure 8). As the incremental sensitivity of one axis is reduced, such as when the acceleration on that axis approaches +1 *g* or −1 *g*, the incremental sensitivity of the other axis increases.

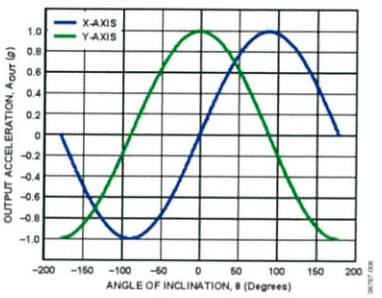


Figure 8. Output Acceleration vs. Angle of Inclination for Dual-Axis Inclination Sensing

One method to convert the measured acceleration to an inclination angle is to compute the inverse sine of the x-axis and the inverse cosine of the y-axis, similar to the single-axis solution. However, an easier and more efficient approach is to use the ratio of the two values, which results in the following:

$$\frac{A_{X,OUT}}{A_{Y,OUT}} = \frac{1\,g \times \sin(\theta)}{1\,g \times \cos(\theta)} = \tan(\theta) \tag{6}$$

$$\theta = \tan^{-1}\left(\frac{A_{X,OUT}}{A_{Y,OUT}}\right) \tag{7}$$

where the inclination angle, θ, is in radians.

Unlike the single-axis example, using the ratio of the two axes to determine the angle of inclination makes determining an incremental sensitivity very difficult. Instead, it is more useful to determine the minimum necessary accelerometer resolution, given a desired inclination resolution. Given that the incremental sensitivity of one axis increases as the other decreases, the net result is an effective incremental sensitivity that is roughly constant. This means that the selection of an accelerometer with sufficient resolution to achieve the desired inclination step size at one angle is sufficient for all angles.

To determine the minimum necessary accelerometer resolution, Equation 6 is examined to determine where the resolution limitations are. Because the output of each axis relies on the sine or cosine of the angle of inclination, and the angle of inclination for each function is the same, the minimum resolvable angle corresponds to the minimum resolvable acceleration.

As shown in Figure 3 and Figure 4, the sine function has the greatest rate of change near 0°, and it can be shown that the cosine function has the least rate of change at this point. For this reason, the change in acceleration on the x-axis due to a change in inclination is recognized before a change in acceleration on the y-axis. Therefore, the resolution of the system near 0° depends primarily on the resolution of the x-axis. To detect an inclination change of P°, the accelerometer must be able to detect a change of approximately

$$\Delta A_{OUT}\,[g] \cong 1\,g \times \sin(P) \tag{8}$$

Figure 9 can be used to determine the minimum necessary accelerometer resolution—or maximum accelerometer scale factor—for a desired inclination step size. Note that increased accelerometer resolution corresponds with a reduction in accelerometer scale factor and with the ability to detect a smaller change in output acceleration. Therefore, when selecting an accelerometer with the appropriate resolution, the scale factor should be less than the limit shown in Figure 9 for the intended inclination step size.

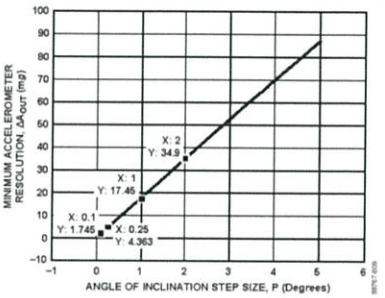


Figure 9. Minimum Accelerometer Resolution for a Desired Angle of Inclination Resolution

Reduced Dependence on Alignment with Plane of Gravity

The second major benefit of using at least two axes is that unlike the single-axis solution, where tilt in any axis other than the x-axis can cause significant error, the use of a second axis allows for an accurate value to be measured even when inclination in the third axis is present. This is because the effective incremental sensitivity is proportional to the root-sum-square (RSS) value of gravity on the axes of interest.

When gravity is completely contained in the xy-plane, the RSS value of acceleration detected on those axes is ideally equal to 1 *g*. If tilt is present in the xz- or yz-plane, the total acceleration due to gravity is reduced, which also reduces the effective incremental sensitivity. This, in turn, increases the inclination step size for a given accelerometer resolution, but still provides an accurate measurement. The resulting angle from the inclination calculation corresponds to the rotation in the xy-plane.

If the system is tilted enough, such that very little acceleration due to gravity is present in the xy-plane, the inclination angle step size will be too coarse to be useful; therefore, it is recommended that tilt in the xz- or yz-plane be limited.

Complete 360° Tilt Sensing

The third major benefit of using a second axis is the ability to distinguish between each quadrant and to measure angles throughout the entire 360° arc. As shown in Figure 10, each quadrant has a different combination of signs associated with the x- and y-axis acceleration.

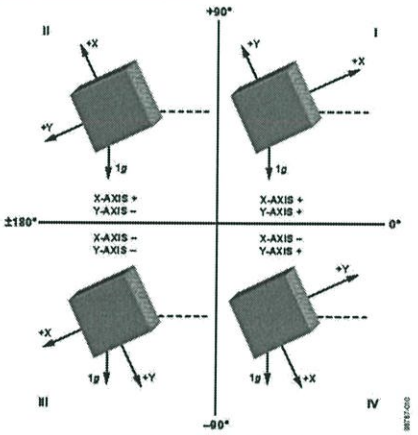


Figure 10. Angle of Inclination and Sign of Acceleration for Quadrant Detection

The inverse tangent function returns a value in Quadrant I if the operand, $A_{X,OUT}/A_{Y,OUT}$, is positive; if the operand is negative, the inverse tangent function returns a value in Quadrant IV. Because the operand in Quadrant II is negative, a value of 180° should be added to the result of the calculation when the angle is in that quadrant. Because the operand in Quadrant III is positive, a value of 180° should be subtracted from the result of the calculation when the angle is in that quadrant. The correct quadrant of the calculated angle can be determined by examining the sign of the measured acceleration on each axis.

Triple-Axis Tilt Calculation

When a third axis is introduced, the orientation of the sensor can be determined in a complete sphere. The classical method of rectangular (x, y, z) to spherical (ρ, θ, ϕ) conversion can be used to relate the angle of tilt in the xy-plane, θ, and the angle of inclination from the gravity vector, ϕ, to the measured acceleration in each axis, as follows:

$$\theta = \tan^{-1}\left(\frac{A_{X,OUT}}{A_{Y,OUT}}\right) \quad (9)$$

$$\phi = \cos^{-1}\left(\frac{A_{Z,OUT}}{\sqrt{A^2_{X,OUT} + A^2_{Y,OUT} + A^2_{Z,OUT}}}\right) \quad (10)$$

Given the assumption that the only measured acceleration is due to gravity, the denominator of the operand in Equation 10 can be replaced with a constant, ideally 1, because the RSS value of all the axes is constant when the only acceleration is gravity. The angles are shown in Figure 11, where Figure 11c shows θ only in the xy-plane, and Figure 11d shows ϕ as the angle between the z-axis and the gravity vector.

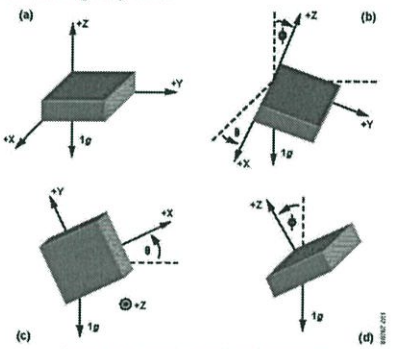


Figure 11. Angles of Spherical Coordinate System

Due to the similarities between the equations for the triple-axis method and the equations for the single- and dual-axis methods, the analysis of the triple-axis solution is the same as for the single- and dual-axis methods combined. The measurement of θ benefits from the ratio of two orthogonal axes, and a desired inclination resolution requires a minimum accelerometer resolution as described by Equation 8.

The measurement of ϕ corresponds to the measurement of the inclination angle for the single-axis solution, along with the method for determining the minimum accelerometer resolution needed for a specific inclination angle resolution over a desired range. The difference is that the use of the inverse cosine function to determine ϕ results in a maximum incremental sensitivity when ϕ is 90° and a minimum incremental sensitivity at 0° and 180°.

A plot similar to Figure 3 and Figure 4 can be generated by substituting cosine for sine in Equation 2. It is important to note that although θ ranges from −180° to +180°, ϕ ranges only from 0° to 180°. A negative angle for ϕ causes the angle of θ to become negative.

An alternative method for inclination sensing with three axes is to determine the angle individually for each axis of the accelerometer from a reference position. The reference position is taken as the typical orientation of a device with the x- and y-axes in the plane of the horizon (0 *g* field) and the z-axis orthogonal to the horizon (1 *g* field). This is shown in Figure 12 with θ as the angle between the horizon and the x-axis of the accelerometer, ψ as the angle between the horizon and the y-axis of the accelerometer, and ϕ as the angle between the gravity vector and the z-axis. When in the initial position of 0 *g* on the x- and y-axes and 1 *g* on the z-axis, all calculated angles would be 0°.

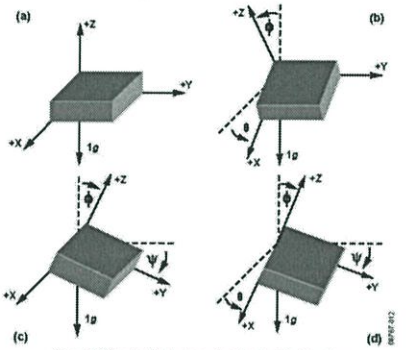


Figure 12. Angles for Independent Inclination Sensing

Basic trigonometry can be used to show that the angles of inclination can be calculated using Equation 11, Equation 12, and Equation 13.

$$\theta = \tan^{-1}\left(\frac{A_{X,OUT}}{\sqrt{A^2_{Y,OUT} + A^2_{Z,OUT}}}\right) \quad (11)$$

$$\psi = \tan^{-1}\left(\frac{A_{Y,OUT}}{\sqrt{A^2_{X,OUT} + A^2_{Z,OUT}}}\right) \quad (12)$$

$$\phi = \tan^{-1}\left(\frac{\sqrt{A^2_{X,OUT} + A^2_{Y,OUT}}}{A_{Z,OUT}}\right) \quad (13)$$

The apparent inversion of the operand in Equation 13 is due to the initial position being a 1 *g* field. If the horizon is desired as the reference for the z-axis, the operand can be inverted. A positive angle means that the corresponding positive axis of the accelerometer is pointed above the horizon, whereas a negative angle means that the axis is pointed below the horizon.

Because the inverse tangent function and a ratio of accelerations is used, the benefits mentioned in the dual-axis example apply, namely that the effective incremental sensitivity is constant and that the angles can be accurately measured for all points around the unit sphere.

CALIBRATION FOR OFFSET AND SENSITIVITY MISMATCH ERROR

The analysis in this application note was done under the assumption that an ideal accelerometer was used. This corresponds to a device with no 0 *g* offset and with perfect sensitivity (expressed as mV/*g* for an analog sensor or LSB/*g* for a digital sensor). Although sensors come trimmed, the devices are mechanical in nature, which means that any static stress on the part after assembly of the system may affect the offset and sensitivity. This, combined with the limits of factory calibration, can result in error beyond the allowable limits for the application.

Effects of Offset Error

To demonstrate how large the error can be, imagine first a dual-axis solution with perfect sensitivity but with a 50 mg offset on the x-axis. At 0° the x-axis reads 50 mg and the y-axis reads 1 *g*. The resulting calculated angle would be 2.9°, resulting in an error of 2.9°. At ±180° the x-axis would report 50 mg, whereas the y-axis would report −1 *g*. This would result in a calculated angle and error of −2.9°. The error between the calculated angle and the actual angle is shown in Figure 13 for this example. The error due to an offset may not only be large compared to the desired accuracy of the system, but it can vary, thus making it difficult to simply calibrate out an error angle. This becomes more complicated when an offset for multiple axes is included.

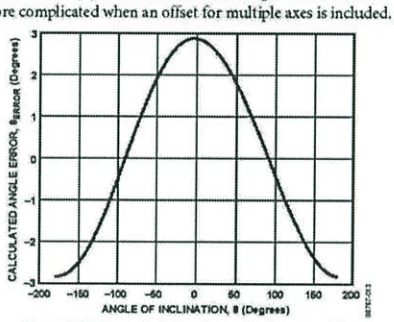


Figure 13. Calculated Angle Error Due to Accelerometer Offset

Effects of Sensitivity Mismatch Error

The main error component due to accelerometer sensitivity in a dual-axis inclination sensing application is when a difference in sensitivity exists between the axes of interest (as opposed to a single-axis solution, where any deviation between the actual sensitivity and the expected sensitivity results in an error). Because the ratio of the x- and y-axes is used, most of the error is cancelled if the sensitivities are the same.

As an example of the effect of accelerometer sensitivity mismatch, assume that a dual-axis solution is used with perfect offset trim, perfect sensitivity on the y-axis, and +5% sensitivity on the x-axis. This means that in a 1 *g* field, the y-axis reports 1 *g*, whereas the x-axis reports 1.05 *g*. Figure 14 shows the error in the calculated angle due to this sensitivity mismatch. Similar to offset error, the error due to accelerometer sensitivity mismatch varies over the entire range of rotation, making it difficult to compensate for the error after calculation of the inclination angle. Skewing the mismatch further by varying the y-axis sensitivity results in even greater error.

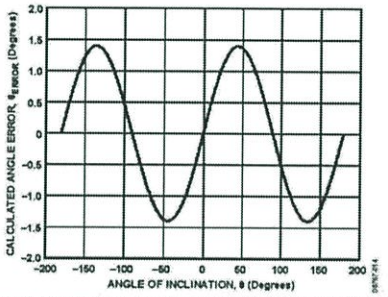


Figure 14. Calculated Angle Error Due to Accelerometer Sensitivity Mismatch

Basic Calibration Techniques

When the errors due to offset and sensitivity mismatch are combined, the error can become quite large and well beyond the acceptable limits in an inclination sensing application. To reduce this error, the offset and sensitivity should be calibrated and the calibrated output acceleration used to calculate the angle of inclination. When including the effects of offset and sensitivity, the accelerometer output is as follows:

$$A_{OUT}\,[g] = A_{OFF} + (Gain \times A_{ACTUAL}) \quad (14)$$

where:
A_{OFF} is the offset error, in *g*.
Gain is the gain of the accelerometer, ideally a value of 1.
A_{ACTUAL} is the real acceleration acting on the accelerometer and the desired value, in *g*.

A simple calibration method is to assume that the gain is 1 and to measure the offset. This calibration then limits the accuracy of the system to the uncalibrated error in sensitivity. The simple calibration method can be done by placing the axis of interest into a 0 *g* field and measuring the output, which would be equal to the offset. That value should then be subtracted from the output of the accelerometer before processing the signal. This is often referred to as a no-turn or single-point calibration, because the typical orientation of a device puts the x- and y-axes in a 0 *g* field. If a three-axis device is used, at least one turn or a second point should be included for the z-axis.

A more accurate calibration method is to use two points per axis of interest (up to six points for a three-axis design). When an axis is placed into a +1 *g* and −1 *g* field, the measured outputs are as follows:

$$A_{+1g}\,[g] = A_{OFF} + (1\,g \times Gain) \quad (15)$$

$$A_{-1g}\,[g] = A_{OFF} - (1\,g \times Gain) \quad (16)$$

where the offset, A_{OFF}, is in *g*.

These two points can be used to determine the offset and gain as follows:

$$A_{OFF}\,[g] = 0.5 \times (A_{+1g} + A_{-1g}) \quad (17)$$

$$Gain = 0.5 \times \left(\frac{A_{+1g} - A_{-1g}}{1\,g}\right) \quad (18)$$

where the +1 *g* and −1 *g* measurements, A_{+1g} and A_{-1g}, are in *g*.

This type of calibration also helps to minimize cross-axis sensitivity effects as the orthogonal axes are in a 0 *g* field when making the measurements for the axis of interest. These values would be used by first subtracting the offset from the accelerometer measurement and then dividing the result by the gain.

$$A_{ACTUAL}\,[g] = \frac{A_{OUT} - A_{OFF}}{Gain} \quad (19)$$

where A_{OUT} and A_{OFF} are in *g*.

The calculations of A_{OFF} and Gain in Equation 15 through Equation 19 assume that the acceleration values, A_{+1g} and A_{-1g}, are in *g*. If acceleration in mg is used, the calculation of A_{OFF} in Equation 17 remains unchanged, but the calculation of Gain in Equation 18 should be divided by 1000 to account for the change in units.