GloMu

ถุงมือแปลภาษามืออัจฉริยะ

สมาชิกกลุ่ม

ชื่อ รัชพล สกุล ลองซุม	รหัสนศ 2311310862
ชื่อ กิติกร สกุล ขันธะเสน	รหัสนศ 2311310888
ชื่อ จักรพันธ์ สกุล ดวงจิตต์เจริญ	รหัสนศ 2311310896

โครงงานนี้เป็นส่วนหนึ่งของรายวิชา CPE-414 Real Time Embedded Systems

สาขาวิชาคอมพิวเตอร์และปัญญาประดิษฐ์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีไทย-ญี่ปุ่น

เทอม 2 ปีการศึกษา 2568

คำนำ

โครงงานเรื่อง การพัฒนาระบบถุงมือแปลภาษามืออัจฉริยะ(GloMu) จัดทำขึ้นเพื่อเป็นส่วนหนึ่งของการเรียนในรายวิชา CPE-414: Real Time Embedded Systems โดยมีวัตถุประสงค์เพื่อออกแบบและพัฒนาระบบที่สามารถตรวจจับการเคลื่อนไหวของนิ้วมือและองศาการเอียงของมือผ่านเซนเซอร์ จากนั้นนำข้อมูลที่ได้มาประมวลผลด้วยเทคนิคปัญญาประดิษฐ์ (Artificial Intelligence) เพื่อจำแนกท่าทางภาษามือ และแสดงผลออกมาในรูปแบบข้อความพร้อมแปลงเป็นเสียงพูดอัตโนมัติ

แรงบันดาลใจในการจัดทำโครงงานนี้เกิดจากปัญหาด้านการสื่อสารของผู้ที่มีความบกพร่องทางการได้ยินหรือการพูด ซึ่งอาจก่อให้เกิดข้อจำกัดในการใช้ชีวิตประจำวันและการสื่อสารกับบุคคลทั่วไป คณะผู้จัดทำจึงได้พัฒนาถุงมืออัจฉริยะที่ติดตั้ง Potentiometer จำนวน 4 ตัว สำหรับวัดระดับการงอของนิ้วมือแต่ละนิ้ว และใช้เซนเซอร์ GY-521 เพื่อวัดค่าความเร่งและองศาการเอียงของมือ โดยมีไมโครคอนโทรลเลอร์ ESP32 ทำหน้าที่อ่านค่าและส่งข้อมูลเข้าสู่กระบวนการประมวลผลด้วยอัลกอริทึมการเรียนรู้ของเครื่องเพื่อเพิ่มความแม่นยำในการจำแนกท่าทางภาษามือ อีกทั้งยังมีการใช้งานลำโพงเพื่อแสดงผลการทำนายของโมเดลในรูปแบบเสียง ทำให้ผู้ใช้งานสามารถสื่อสารกับบุคคลทั่วไปได้สะดวกและเข้าใจง่ายมากขึ้น

คณะผู้จัดทำขอขอบพระคุณอาจารย์ที่ปรึกษา รวมถึงผู้ที่ให้คำแนะนำและช่วยเหลือในการพัฒนาโครงงานนี้ ตลอดจนเพื่อนร่วมชั้นเรียนที่คอยให้ความร่วมมือและสนับสนุน จนทำให้โครงงานนี้สำเร็จลุล่วงได้ด้วยดี คณะผู้จัดทำหวังเป็นอย่างยิ่งว่าโครงงานนี้จะเป็นประโยชน์ทั้งต่อผู้สอนและผู้เรียน รวมถึงสามารถเป็นแนวทางในการพัฒนาต่อยอดในอนาคตได้

ลงชื่อ รัชพล ลองชุม
ลงชื่อ นายกิติกร ขันธะเสน
ลงชื่อ นายจักรพันธ์ ดวงจิตต์เจริญ

คณะผู้จัดทำ
วันที่ 14 กุมภาพันธ์ 2569

ชื่อโครงงานภาษาไทย : ถุงมือแปลภาษามืออัจฉริยะ

ชื่อโครงงานภาษาอังกฤษ : GloMu

ชื่อคณะผู้จัดทำ : ชื่อ รัชพล ลองซุม รหัสนศ 2311310862

ชื่อ กิติกร ขันธะเสน รหัสนศ 2311310888

ชื่อ จักรพันธ์ ดวงจิตต์เจริญ รหัสนศ 2311310896

คณะวิชา วิศวกรรมศาสตร์ สาขาวิศวกรรมคอมพิวเตอร์และปัญญาประดิษฐ์

ปีที่ทำโครงงาน : 2569

บทคัดย่อ

โครงงานเรื่อง การพัฒนาระบบถุงมือแปลภาษามืออัจฉริยะ (GloMu) จัดทำขึ้นในรายวิชา CPE-414: Real Time Embedded Systems โดยมีวัตถุประสงค์เพื่อออกแบบและพัฒนาระบบที่สามารถตรวจจับการงอของนิ้วมือและการเอียงของมือ เพื่อนำข้อมูลไปประมวลผลด้วยเทคนิคปัญญาประดิษฐ์ในการจำแนกท่าทางภาษามือ และแสดงผลเป็นข้อความพร้อมเสียงพูดแบบเรียลไทม์ เพื่อช่วยลดข้อจำกัดด้านการสื่อสารของผู้ที่มีความบกพร่องทางการได้ยินหรือการพูด

ระบบประกอบด้วย Potentiometer จำนวน 4 ตัว สำหรับวัดการงอนิ้ว และเซนเซอร์ตรวจจับการเคลื่อนไหว GY-521 โดยมีไมโครคอนโทรลเลอร์ ESP32 ทำหน้าที่อ่านค่าและส่งข้อมูลเข้าสู่กระบวนการประมวลผลด้วยอัลกอริทึมการเรียนรู้ของเครื่อง ก่อนแสดงผลผ่านระบบแปลงข้อความเป็นเสียง (Text-to-Speech)

ผลการพัฒนาพบว่าระบบสามารถจำแนกท่าทางภาษามือพื้นฐานและแสดงผลเสียงได้อย่างเหมาะสมต่อการสื่อสารแบบเรียลไทม์ ทั้งยังเป็นการบูรณาการความรู้ด้านระบบสมองกลฝังตัวและการเรียนรู้ของเครื่องตามวัตถุประสงค์ของรายวิชา

กิตติกรรมประกาศ

โครงงานนี้สำเร็จลุล่วงได้ด้วยความกรุณาจาก อาจารย์ฐิติชญา ธนมิตรสมบูรณ์ อาจารย์ที่ปรึกษาโครงงานที่ได้ให้คำแนะนำ แนวคิด ตลอดจนแก้ไขข้อบกพร่องต่าง ๆ มาโดยตลอด จนโครงงานเล่มนี้เสร็จสมบูรณ์ ผู้ศึกษาจึงขอกราบขอบพระคุณเป็นอย่างสูง

คณะผู้จัดทำ

รัชพล ลองซุม

กิติกร ขันธะเสน

จักรพันธ์ ดวงจิตต์เจริญ

สารบัญ

สารบัญภาพ

รูปที่ **หน้า**

สารบัญตาราง

ตารางที่ **หน้า**

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญ

ปัจจุบันผู้ที่มีความบกพร่องทางการได้ยินหรือการพูดต้องเผชิญกับข้อจำกัดด้านการสื่อสารในชีวิตประจำวัน แม้ว่าภาษามือจะเป็นเครื่องมือสำคัญในการสื่อสาร แต่บุคคลทั่วไปจำนวนมากยังไม่สามารถเข้าใจภาษามือได้ ส่งผลให้เกิดช่องว่างทางการสื่อสารและลดโอกาสในการเข้าถึงบริการหรือการมีปฏิสัมพันธ์ทางสังคม

จากปัญหาดังกล่าว คณะผู้จัดทำจึงมีแนวคิดในการพัฒนาระบบถุงมือแปลภาษามืออัจฉริยะ (GloMu) โดยอาศัยเทคโนโลยีระบบสมองกลฝังตัวและการเรียนรู้ของเครื่อง เพื่อช่วยแปลงท่าทางภาษามือให้เป็นข้อความและเสียงพูดแบบเรียลไทม์ ระบบนี้ใช้ Potentiometer สำหรับวัดการงอนิ้ว และใช้เซนเซอร์ตรวจจับการเคลื่อนไหว GY-521 ร่วมกับไมโครคอนโทรลเลอร์ ESP32 เพื่อประมวลผลข้อมูล ซึ่งสอดคล้องกับองค์ความรู้ในรายวิชา Real Time Embedded Systems และสามารถต่อยอดสู่การใช้งานจริงได้ในอนาคต

1.2 วัตถุประสงค์ของการทำโครงงาน

1.2.1. เพื่อออกแบบและพัฒนาต้นแบบถุงมือแปลภาษามืออัจฉริยะที่สามารถตรวจจับการงอนิ้วและการเคลื่อนไหวของมือได้อย่างถูกต้อง

1.2.2. เพื่อพัฒนาโมเดลการเรียนรู้ของเครื่องสำหรับจำแนกท่าทางภาษามือพื้นฐาน

1.2.3. เพื่อพัฒนาระบบแปลงผลการจำแนกเป็นข้อความและเสียงพูดแบบเรียลไทม์

1.3 ขอบเขตของการทำโครงงาน

1.3.1. การสร้างโครงสร้างมือจำลองเพื่อใช้ในการติดตั้งเซนเซอร์และทดสอบความแม่นยำของอัลกอริทึมในการจำแนกท่าทาง

- ออกแบบและพัฒนาโครงสร้างมือจำลองที่ติดตั้ง Potentiometer จำนวน 4 ตัว สำหรับวัดระดับการงอของนิ้วมือ และใช้ GY-521 สำหรับวัดค่าความเร่งและองศาการเอียงของมือ โดยใช้ ESP32 เป็นหน่วยประมวลผลหลักในการอ่านค่าเซนเซอร์และส่งข้อมูลเข้าสู่ระบบ

1.3.2. การพัฒนาและฝึกสอนโมเดล Random Forest

- รวบรวมข้อมูลท่าทางภาษามือพื้นฐาน สร้างชุดข้อมูล (Dataset) และนำข้อมูลเข้าสู่กระบวนการเตรียมข้อมูล เช่น การกรองสัญญาณและการจัดรูปแบบข้อมูล ก่อนนำไปฝึกสอนโมเดลด้วยอัลกอริทึม Random Forest เพื่อจำแนกประเภทของท่าทาง

1.3.3. การประเมินประสิทธิภาพของโมเดล

- ประเมินผลการทำงานของโมเดลโดยใช้ตัวชี้วัด เช่น Accuracy และ Confusion Matrix รวมถึงทดสอบความหน่วงเวลา (Latency) ของระบบตั้งแต่การตรวจจับท่าทางจนถึงการแสดงผลเสียง

1.3.4. การแปลงข้อความเป็นเสียงพูด

- พัฒนาระบบแปลงข้อความเป็นเสียง (Text-to-Speech) เพื่อแสดงผลลัพธ์การจำแนกท่าทางในรูปแบบเสียงผ่านลำโพง โดยเน้นให้สามารถทำงานได้แบบเรียลไทม์

1.4 ผลที่คาดว่าจะได้รับ

1.4.1. ได้ต้นแบบระบบถุงมือแปลภาษามือที่สามารถใช้งานได้จริงในระดับพื้นฐาน

1.4.2. ได้โมเดลการเรียนรู้ของเครื่องที่สามารถจำแนกท่าทางภาษามือได้อย่างมีประสิทธิภาพ

1.4.3. ลดข้อจำกัดด้านการสื่อสารระหว่างผู้ใช้ภาษามือและบุคคลทั่วไป

1.4.4. ผู้จัดทำได้รับความรู้และทักษะด้านระบบสมองกลฝังตัว การประมวลผลสัญญาณ และการพัฒนาโมเดลปัญญาประดิษฐ์

บทที่ 2
ทฤษฎีและโครงงานที่เกี่ยวข้อง

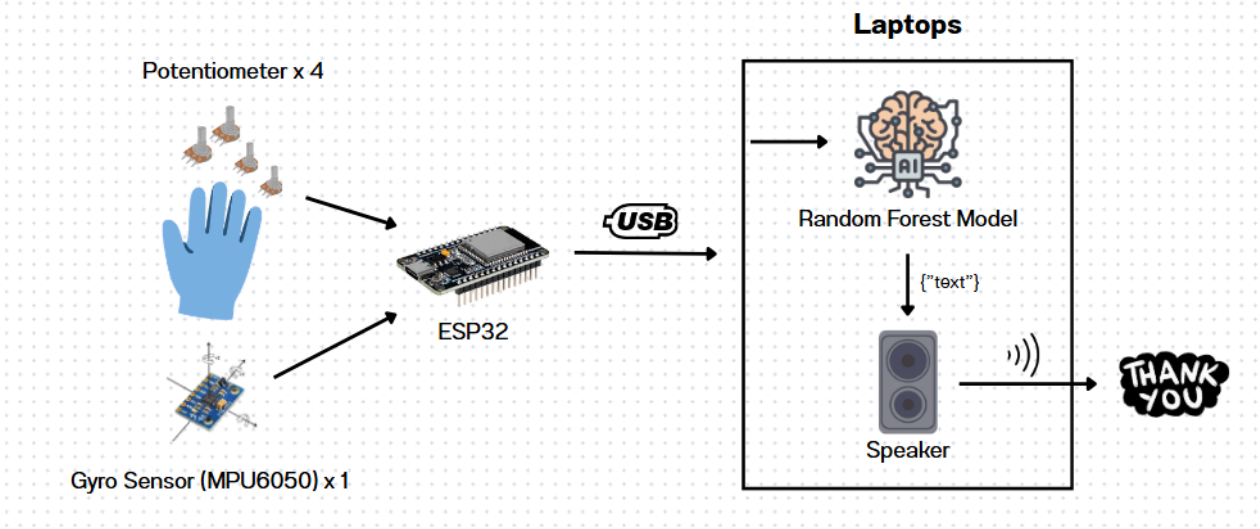
ในบทนี้ ผู้จัดทำโครงงานได้อธิบายถึงการการศึกษาองค์ความรู้ทั้งหมดที่เกี่ยวข้องรวมถึงโครงงานที่มีความสำคัญต่อการดำเนินการทำโครงงานนี้ โดยแบ่งออกเป็นหัวข้อดังนี้ หัวข้อ 2.1 อธิบายเกี่ยวกับฮาร์ดแวร์ และซอฟต์แวร์ ต่อมาคือหัวข้อ 2.2 อธิบายเกี่ยวกับทฤษฎีการเรียนรู้ของเครื่องจักร (Machine Learning)

2.1 ภาพรวมของระบบ

ระบบถุงมือแปลภาษามืออัจฉริยะ (GloMu) ถูกออกแบบให้ทำงานในลักษณะ Real-Time Embedded System โดยมีการทำงานแบ่งออกเป็น 3 ส่วนหลัก ได้แก่

- Sensing – ทำหน้าที่ตรวจจับการเคลื่อนไหวของนิ้วมือและองศาการเอียงของมือ
- Processing – ทำหน้าที่ประมวลผลข้อมูลและจำแนกท่าทางด้วยโมเดล Machine Learning
- Output – แสดงผลลัพธ์เป็นข้อความและเสียงพูด

ในส่วนของ Sensing ระบบใช้ Potentiometer จำนวน 4 ตัวสำหรับวัดการงอนิ้ว และใช้เซนเซอร์ตรวจจับการเคลื่อนไหว GY-521 สำหรับวัดค่าความเร่งและองศาการหมุนของมือ โดยมี ESP32 เป็นหน่วยประมวลผลหลักในการอ่านค่าเซนเซอร์ผ่าน ADC และ I2C จากนั้นส่งข้อมูลไปยังคอมพิวเตอร์เพื่อทำการประมวลผลด้วยโมเดล Random Forest และแสดงผลผ่านระบบ Text-to-Speech



(รูปที่ 2.1 System Diagram)

2.2 ฮาร์ดแวร์และซอฟต์แวร์

ระบบถูกพัฒนาโดยอาศัยการทำงานร่วมกันระหว่างฮาร์ดแวร์และซอฟต์แวร์ เพื่อให้สามารถทำงานได้อย่างครบวงจรตั้งแต่การตรวจจับสัญญาณจนถึงการแสดงผลลัพธ์

2.1.1 ฮาร์ดแวร์

1. Potentiometer ใช้สำหรับวัดระดับการงอของนิ้วมือ อ่านค่าผ่าน ADC ความละเอียด 12-bit ของ ESP32

2. GY-521 ใช้สำหรับวัดค่า Acceleration และ Angular Velocity ใน 3 แกน (X, Y, Z) เพื่อคำนวณองศาการเอียงของมือ
3. ESP32 ทำหน้าที่เป็นหน่วยประมวลผลหลัก อ่านค่าเซนเซอร์ ส่งข้อมูลผ่าน Serial และควบคุมการทำงานของระบบ

2.1.2 ซอฟต์แวร์

1. ส่วนพัฒนาโมเดล Machine Learning
    - JupyterLab
    - VS Code
    - ไลบรารีสำหรับจัดการข้อมูล เช่น NumPy, Pandas
    - ไลบรารีสำหรับแสดงข้อมูล เช่น Matplotlib, seaborn
    - ไลบรารีสำหรับสร้างโมเดล เช่น Scikit-learn
2. ส่วนประมวลผลแบบเรียลไทม์บน ESP32 ด้วย RTOS
    - ระบบประมวลผลแบบเรียลไทม์ถูกพัฒนาบน ESP32 ซึ่งรองรับการทำงานด้วยระบบปฏิบัติการแบบ Real-Time Operating System (RTOS) โดยใช้ FreeRTOS ที่ทำงานอยู่ภายในตัวไมโครคอนโทรลเลอร์ การทำงานของระบบถูกแบ่งออกเป็นหลาย Task เพื่อให้สามารถจัดการงานแบบขนาน (Multitasking)

3. แพลตฟอร์มที่ใช้ในการพัฒนา

    แพลตฟอร์มที่ใช้ในการพัฒนาโครงงานนี้ประกอบด้วยระบบปฏิบัติการและเครื่องมือที่รองรับการพัฒนา Machine Learning และ Embedded System ดังนี้

    - ระบบปฏิบัติการ Windows หรือ macOS
    - สภาพแวดล้อมสำหรับพัฒนา Machine Learning บน Jupyter Notebook
    - โปรแกรม Anaconda ร่วมกับ JupyterLab หรือ Visual Studio Code (VS Code)
    - Arduino IDE หรือ PlatformIO สำหรับพัฒนาโปรแกรมบน ESP32

2.2 ทฤษฎีของ Machine Learning

2.2.1 นิยามของ Machine Learning

Machine Learning เป็นสาขาหนึ่งของปัญญาประดิษฐ์ (Artificial Intelligence) ที่เน้นการพัฒนาและการออกแบบของระบบที่สามารถเรียนรู้และปรับปรุงความสามารถต่างๆ ได้จากข้อมูลโดยไม่จำเป็นต้องมีการโปรแกรมโดยตรงในทุกกรณี หมายถึงการตรวจจับแนวโน้มหรือลักษณะในข้อมูลและสร้างโมเดลหรืออัลกอริทึมที่สามารถทำนายผลลัพธ์หรือการกระทำในอนาคตได้ ซึ่งหากนำการเรียนรู้ของเครื่องจักรไปปรับใช้ได้อย่างถูกวิธี Machine Learning จะทำให้มนุษย์สามารถลดเวลาการทำงานในการวิเคราะห์ข้อมูลและการตอบสนองต่างๆ ส่วนในเชิงธุรกิจก็จะสามารถลดต้นทุนแรงงานได้อย่างมหาศาล หลักการของ Machine Learning คือการให้ระบบสามารถเรียนรู้และปรับตัวเองได้โดยใช้ข้อมูลโดยมักจะแบ่ง Machine Learning ได้แก่

• Supervised Learning (การเรียนรู้แบบมีผู้สอน) โมเดลจะเรียนรู้จากข้อมูลที่มีคำตอบโดยมีการกำหนดคำตอบล่วงหน้าในข้อมูลเพื่อให้โมเดลจะทำนายผลลัพธ์สำหรับข้อมูลใหม่ โดยมีรูปแบบ ดังตารางที่ 2.2

ตารางที่ 2.2 ประเภทของการเรียนรู้แบบมีผู้สอน

รูปแบบของการเรียนรู้แบบมีผู้สอน	วัตถุประสงค์ของการเรียนรู้
การจำแนกประเภท (Classification)	การทำนายประเภทหรือกลุ่มของข้อมูล ซึ่งประเภทของข้อมูลเป็นข้อมูลที่มีลักษณะไม่ต่อเนื่อง เช่น การจำแนกว่าภาพถ่ายเป็นภาพของสุนัขหรือแมว
การทำนายค่า (Regression)	การทำนายค่าต่อเนื่อง ซึ่งค่าที่ต้องการทำนายมักจะเป็นตัวเลข เช่น การทำนายราคาของสินค้าต่างๆ

• Unsupervised Learning (การเรียนรู้แบบไม่มีผู้สอน) โมเดลจะเรียนรู้โดยการค้นหาลักษณะหรือโครงสร้างในข้อมูลโดยไม่มีการกำหนดคำตอบล่วงหน้าโดยที่โมเดลจะหาลักษณเด่นหรือกลุ่มที่มีความสัมพันธ์กันในข้อมูล โดยมีรูปแบบ ดังตารางที่ 2.3

ตารางที่ 2.3 ประเภทของการเรียนรู้แบบไม่มีผู้สอน

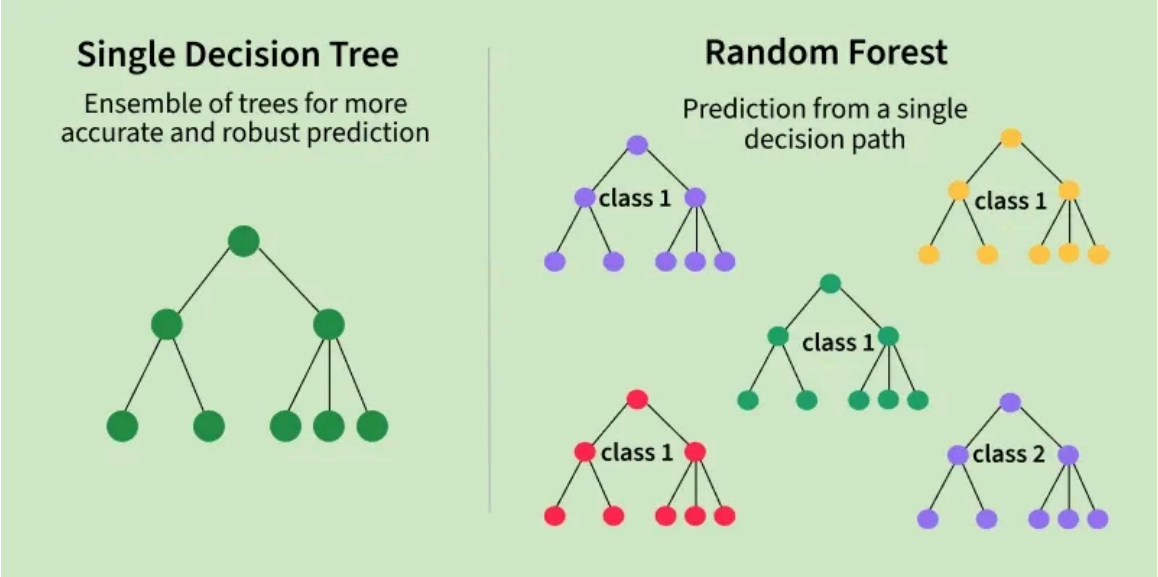
รูปแบบของการเรียนรู้แบบไม่มีผู้สอน	วัตถุประสงค์ของการเรียนรู้
การจัดกลุ่ม (Clustering)	การหาโครงสร้างหรือความสัมพันธ์ระหว่างข้อมูลโดยการแบ่งข้อมูลออกเป็นกลุ่มที่มีความคล้ายคลึงกัน โดยไม่มีการให้ข้อมูลเพิ่มเติมว่าแต่ละกลุ่มคืออะไร เช่น การจัดกลุ่มข่าวในสื่อออนไลน์เป็นกลุ่มข่าวในหมวดหมู่เดียวกัน
การลดมิติ (Dimensionality Reduction)	กระบวนการลดจำนวนตัวแปรหรือมิติของชุดข้อมูล โดยการลดมิตินี้มักจะช่วยให้ง่ายต่อการจัดเก็บข้อมูลและประมวลผลข้อมูลต่อมา เช่นการใช้ Principal Component Analysis (PCA)

2.2.2 นิยามของ Random Forest Algorithm

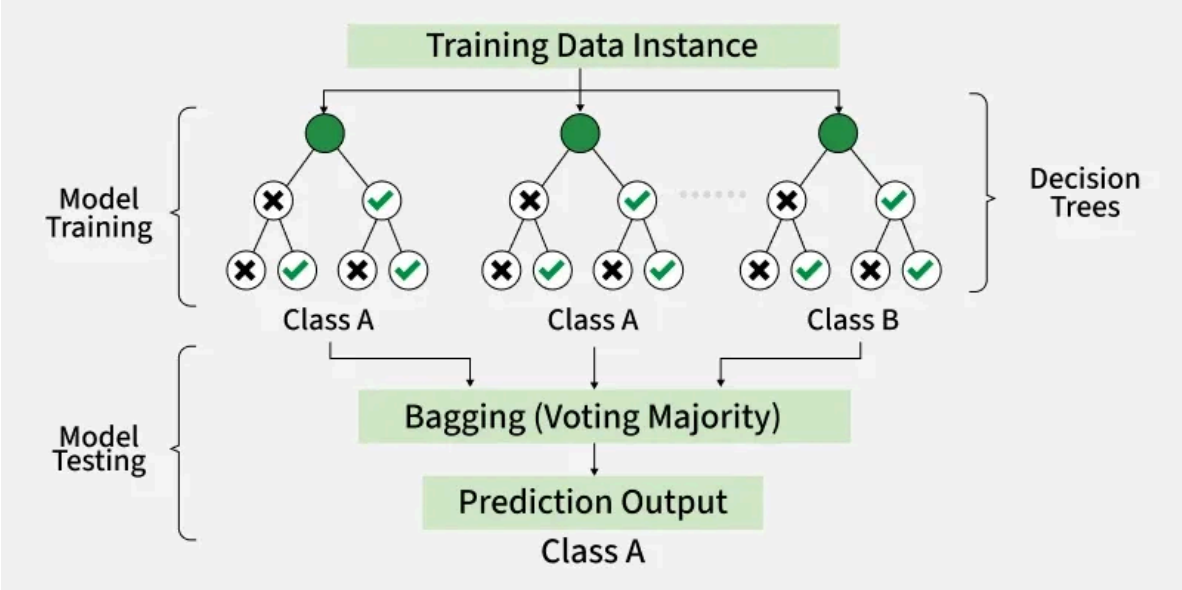
Random Forest เป็นอัลกอริธึมการเรียนรู้ของเครื่องที่ใช้ต้นไม้ตัดสินใจจำนวนมากเพื่อทำนายผลลัพธ์ได้ดียิ่งขึ้น ต้นไม้แต่ละต้นจะพิจารณาส่วนต่างๆ ของข้อมูลแบบสุ่ม และผลลัพธ์จะถูกรวมเข้าด้วยกันโดยการลงคะแนนสำหรับการจำแนกประเภทหรือการหาค่าเฉลี่ยสำหรับการถดถอย ซึ่งทำให้เป็นเทคนิคการเรียนรู้แบบกลุ่ม (ensemble learning) วิธีนี้ช่วยเพิ่มความแม่นยำและลดข้อผิดพลาด

โดยมีหลักการเบื้องต้น ดังนี้

- สร้างหลาย Decision Trees ขึ้นมา
- เลือก features แบบสุ่ม เพื่อสร้างความแตกต่างระหว่างต้นไม้แต่ละต้น
- แต่ละต้นไม้จะทำการ ทำนายผลลัพธ์ ของตัวเอง
- รวมผลการทำนายจากทุกต้นไม้เพื่อให้ได้คำตอบสุดท้าย (ใช้ได้ทั้ง classification และ regression)



(รูปที่ 2.2 Random Forest)



(รูปที่ 2.3 Training of Random Forest Model)

ตารางที่ 2.4 เปรียบเทียบข้อดี-ข้อเสียของ Random Forest Model

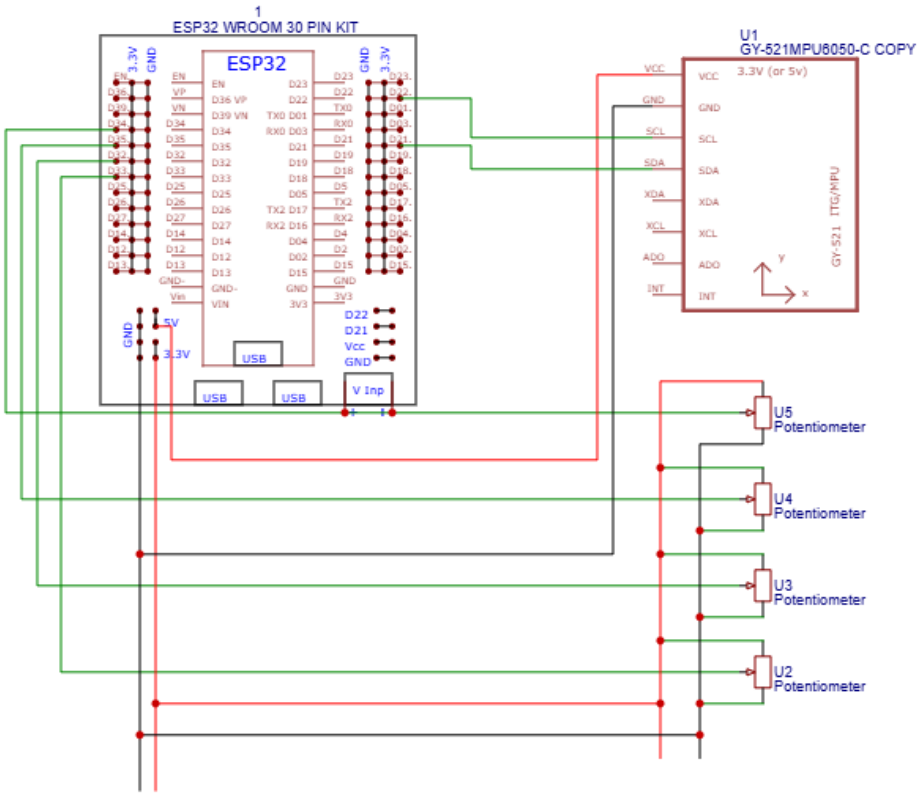
ข้อดี	ข้อจำกัด
ให้ผลทำนายที่แม่นยำแม้กับข้อมูลขนาดใหญ่	ใช้เวลาและทรัพยากรคำนวณมากเมื่อมีจำนวนต้นไม้เยอะ
ไม่จำเป็นต้องทำ normalization หรือ standardization	
ลดความเสี่ยงของ overfitting เมื่อรวมหลายต้นไม้เข้าด้วยกัน	ยากต่อการตีความเมื่อเทียบกับโมเดลง่าย ๆ อย่าง Decision Tree
แสดง feature importance ว่าตัวแปรไหนมีผลต่อการทำนาย	

บทที่ 3
วิธีการดำเนินงาน

บทนี้นำเสนอขั้นตอนและกระบวนการพัฒนาระบบถุงมือแปลภาษามืออัจฉริยะอย่างเป็นลำดับ ตั้งแต่การออกแบบและพัฒนาด้านฮาร์ดแวร์ การสร้างและฝึกสอนโมเดลปัญญาประดิษฐ์ การพัฒนาระบบแปลงข้อความเป็นเสียงพูด ตลอดจนการประกอบและทดสอบรวมระบบทั้งหมดเข้าด้วยกัน โดยใช้ไมโครคอนโทรลเลอร์ ESP32 เป็นหน่วยประมวลผลหลักในการจัดการข้อมูลแบบเรียลไทม์

3.1 ฮาร์ดแวร์

3.1.1 การออกแบบวงจร



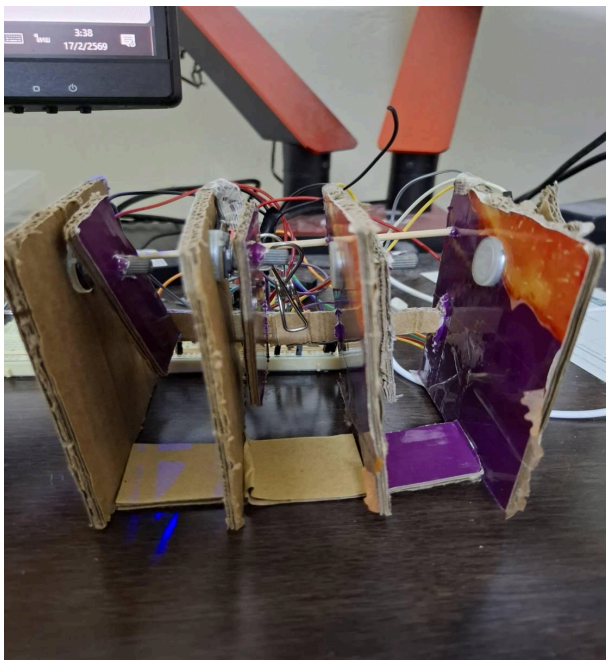
(รูปที่ 3.1.1 Circuit design)

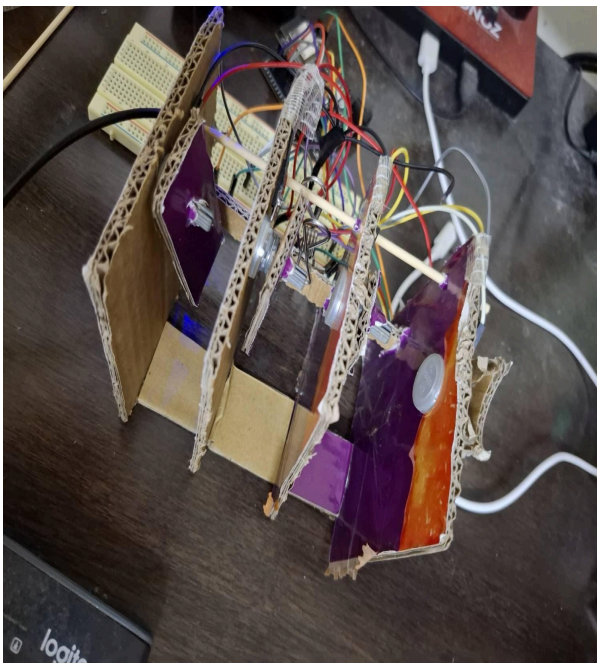
ตารางที่ 3.1.1 รายการอุปกรณ์ที่ใช้

อุปกรณ์	ขาอุปกรณ์	ขา ESP32	ประเภทสัญญาณ	หน้าที่
Potentiometer 1 (U1)		D34	Analog Input	ตรวจจับนิ้วที่ 1 (p0)
Potentiometer 2 (U2)		D35	Analog Input	ตรวจจับนิ้วที่ 2 (p1)
Potentiometer 3 (U3)		D32	Analog Input	ตรวจจับนิ้วที่ 3 (p2)
Potentiometer 4 (U4)		D33	Analog Input	ตรวจจับนิ้วที่ 4 (p3)
GY-521 (MPU6050)	SDA	D21	Digital (I2C)	ส่งข้อมูลความเร่ง/มุม (Data)
GY-521 (MPU6050)	SCL	D22	Digital (I2C)	ส่งสัญญาณนาฬิกา (Clock)
ทุกอุปกรณ์	VCC / VIN	3.3V	Power	จ่ายไฟเลี้ยงอุปกรณ์
ทุกอุปกรณ์	GND	GND	Ground	สายดินร่วมกัน

3.1.2 การออกแบบโครงสร้างถุงมือ

จำลองมือขึ้นมาด้วยกระดาษลัง ตัดประกอบให้ออกมาเป็น 4 ช่อง เจาะรูบริเวณปลายห่างออกมาจากขอบประมาณหนึ่ง ขนาดกว้างเท่าฐานของ Potentiometer และต่อสายไฟให้เสร็จเรียบร้อย





(รูปที่ 3.1.2 ภาพชิ้นงาน)

3.1.3 การพัฒนาโปรแกรมด้วย RTOS

โปรแกรมในการพัฒนา : Visual Studio Code โดยใช้ extension PlatformIO

ไลบรารีที่ใช้ :

```
#include <Arduino.h>

#include <Wire.h>

#include <MPU6050_light.h>
```

ตัวแปรที่ประกาศ

```
// indexFinger 34

// middleFinger 35

// ringFinger 32

// littelFinger 33
```

```
const int potPins[] = {34, 35, 32, 33};

struct DataType
{
  uint32_t timestamp; // Current Time
  float imu[6];       // ax, ay, az, gx, gy, gz; // Accelerometer Gyroscope
  int pot[4];         // Potentiometer 4 นิ้ว
};

QueueHandle_t sensorQueue;
```

ฟังก์ชัน :

taskSensorCollect = เก็บข้อมูลจาก GY-521 และ Potentiometer ทั้ง 4 ตัว และส่งไปให้ taskSerial

assignData = แสดงผลลัพท์ทาง Serial Monitor

taskSerial = รับข้อมูลจาก taskSensorCollect และแสดงผลลัพท์เมื่อมีการพิมพ์ตัวอักษรจากทาง Serial ผ่านไฟล์ Python และจะเรียกใช้ assignData เพื่อส่งข้อมูลให้ไปบันทึกข้อมูล

3.1.4 การเก็บข้อมูล (Data Acquisition)

เก็บข้อมูลทาง Serial ที่เชื่อมต่อกันระหว่าง ESP32 และเครื่องคอมพิวเตอร์หรือLaptop จะแบ่งไฟล์ออกเป็น 2 ไฟล์ ได้แก่ Local_Serial.py และ main.py โดยใช้ ไลบรารีดังนี้

```
import serial, csv, time , os , sys
```

1. **serial (PySerial):** ใช้สำหรับ สื่อสารผ่านพอร์ต USB (Serial Port) ระหว่างคอมพิวเตอร์กับ ESP32
2. **csv:** ใช้สำหรับ จัดการไฟล์นามสกุล .csv (Comma Separated Values)
3. **time:** ใช้จัดการเรื่อง เวลาและการหน่วงเวลา
4. **os (Operating System):** ใช้สั่งงานที่เกี่ยวกับระบบปฏิบัติการ (Windows/Linux/Mac)
5. **sys (System):** ใช้เข้าถึง ตัวแปรและฟังก์ชันที่เกี่ยวข้องกับ Python Interpreter

ในไฟล์ Local_Serial.py

มีฟังก์ชัน:

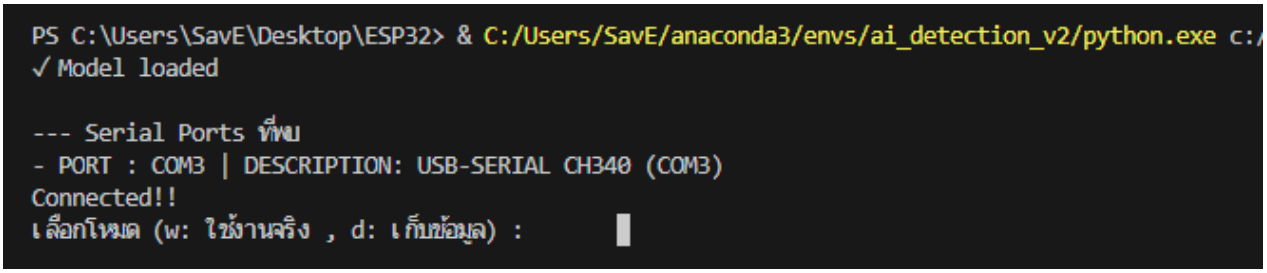
- **save_csv** = ทำการนำข้อมูลที่ได้มาบันทึกลงไฟล์ในรูปแบบ .csv
- **find_ports** = หาพอร์ตที่ ESP32 กับเครื่องที่เชื่อมต่ออยู่ตอนนี้
- **init_serial** = ทำการตั้งค่าเริ่มต้นเชื่อมต่อ port
- **dynamic_record_gesture** = บันทึกค่าในช่วงเวลาหนึ่งถึงอีกช่วงเวลาหนึ่ง
- **static_record_gesture** = บันทึกค่าในตอนนั้นแค่ครั้งเดียว
- **execute_gesture** = อ่านค่าแบบ Real-time เพื่อใช้งานจริง

ในไฟล์ main.py

มีฟังก์ชัน:

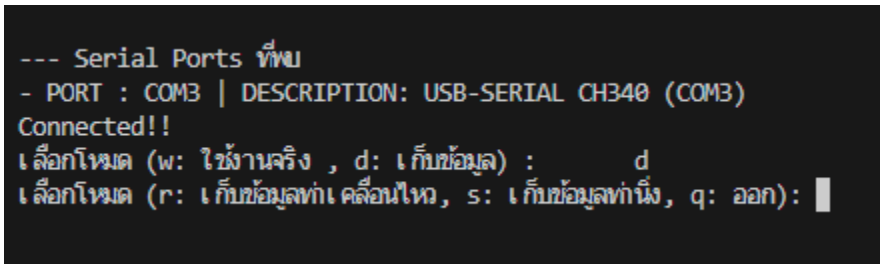
main = เป็นฟังก์ชันหลักในการใช้งาน โดยมีรายละเอียดการทำงานดังนี้

1.เมื่อรันไฟล์จะให้เลือกโหมดที่จะทำงาน



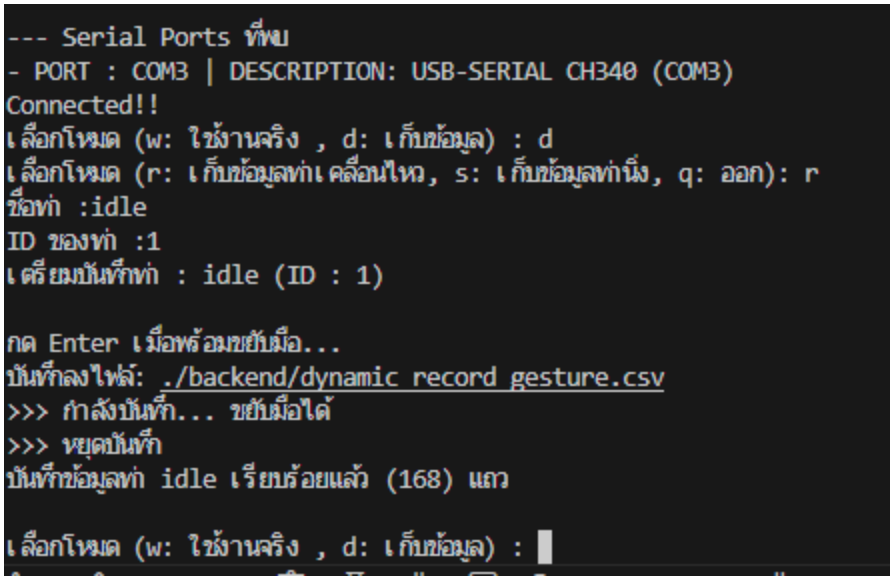
(รูปที่ 3.1.4.1 ผลลัพท์เมื่อเริ่มทำงาน)

2.1. เมื่อกด ‘d’ หรือ ‘D’ ก็จะทำงานในส่วนของการเก็บข้อมูลสามารถเลือกต่อไปได้คือ r และ s หรือถ้าต้องการหยุดก็กด q เพื่อออก



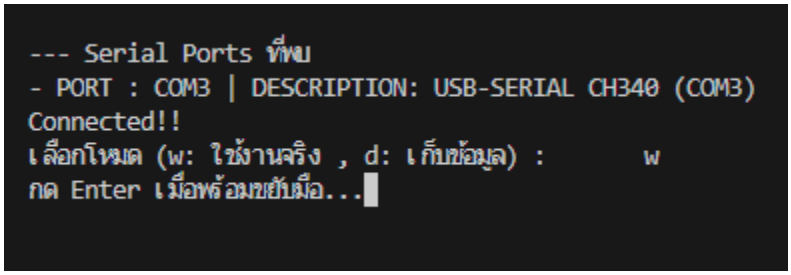
(รูปที่ 3.1.4.2 ผลลัพท์การทำงานเมื่อพิมพ์ ‘d’)

2.2. เมื่อกด ‘r’ || ‘R’ หรือ ‘s’ || ‘S’ ก็จะมายังส่วนของการกรอกข้อมูล ชื่อท่า, ครั้งที่ทำ และให้กด Enter เพื่อเริ่มการทำงาน เมื่อเสร็จแล้วก็จะบันทึกข้อมูลไปที่ไฟล์ dynamic_record_gesture.csv ถ้ากด s ก็จะบันทึกไปยังไฟล์ static_record_gesture.csv และกลับไปทำงานที่ขั้นตอนที่ 1 ใหม่อีกครั้ง



(รูปที่ 3.1.4.3 ผลลัพท์การทำงานเมื่อพิมพ์ ‘r’)

3.เมื่อกด ‘w’ หรือ ‘W’ กดจะเริ่มใช้งานจริงได้ เมื่อพร้อมกด Enter และโปรแกรมจะเริ่มทำงาน



(รูปที่ 3.1.4.4 ผลลัพท์การทำงานเมื่อพิมพ์ ‘w’)

3.1. เมื่อทำงานแล้วจะแสดงผลลัพท์ค่าต่างๆที่ใช้ เช่น ค่าของโมดูล GY-521 และ Potentiometer ทั้ง 4 พร้อมทั้งบอกว่าข้อมูลนี้ตรงกับท่าอะไรที่ทำการสร้าง AI Model ไว้ซึ่งจะระบุความมั่นใจไว้ด้วย

```
Connected!!
เลือกโหมด (w: ใช้งานจริง , d: เก็บข้อมูล) :       w
กด Enter เมื่อพร้อมขยับมือ...
ข้อมูล :       ax    ay    az    gx    gy    gz  p0    p1    p2    p3
0   0.00  0.00  0.99 -0.15  0.03  0.11   0     0  4095  4095
1  -0.01 -0.00  0.99 -0.14  0.03  0.18   0  2962  4095  4095
2  -0.01  0.02  1.00  0.38  0.26 -0.62   0    68  4095  4095
3  -0.01  0.00  1.00  0.08  0.00  0.22   0     0  4095  4095
4  -0.01 -0.00  0.99  0.22  0.02  0.44   0  4095  4095  4095
..   ...   ...   ...   ...   ...   ...  ..   ...   ...   ...
95 -0.01  0.00  0.99 -0.18 -0.03  0.40   0     0  4095  4095
96 -0.01 -0.01  0.99 -0.40  0.15  0.14   0  3018  4095  4095
97 -0.01 -0.00  0.99 -0.11 -0.18 -0.17   0    15  4095  4095
98 -0.01  0.00  0.99 -0.04 -0.14  0.22   0     0  4095  4095
99 -0.01 -0.01  0.98  0.20 -0.01 -0.06   0  4095  4095  4095

[100 rows x 10 columns] ท่าทางที่ตรวจพบ: 0.0 (ความมั่นใจ: 0.51)
ข้อมูล :       ax    ay    az    gx    gy    gz  p0    p1    p2    p3
0  -0.01  0.00  0.99 -0.01  0.02  0.00   0     0  4095  4095
1  -0.01 -0.00  0.99 -0.09 -0.04 -0.01   0  2992  4095  4095
2  -0.00 -0.00  0.99 -0.01 -0.01  0.34   0    61  4095  4095
3  -0.00  0.00  0.99 -0.04 -0.03  0.06   0     0  4095  4095
4  -0.01 -0.00  0.99  0.08 -0.18 -0.07   0  4095  4095  4095
..   ...   ...   ...   ...   ...   ...  ..   ...   ...   ...
95 -0.00 -0.00  0.99  0.09  0.05  0.29   0     0  4095  4095
96 -0.01 -0.00  0.99 -0.11  0.20  0.15   0  3123  4095  4095
97 -0.01  0.00  0.99 -0.15  0.02 -0.30   0     0  4095  4095
98 -0.01 -0.00  0.99 -0.01  0.25  0.15   0     0  4095  4095
99 -0.01 -0.01  0.98 -0.18 -0.14  0.00   0  3759  4095  4095

[100 rows x 10 columns] ท่าทางที่ตรวจพบ: 0.0 (ความมั่นใจ: 0.51)
```

(รูปที่ 3.1.4.5 ผลลัพท์การทำงานหลังพิมพ์ ‘w’)

3.2 ปัญญาประดิษฐ์และการเรียนรู้ของเครื่อง

3.2.1 การเตรียมข้อมูล

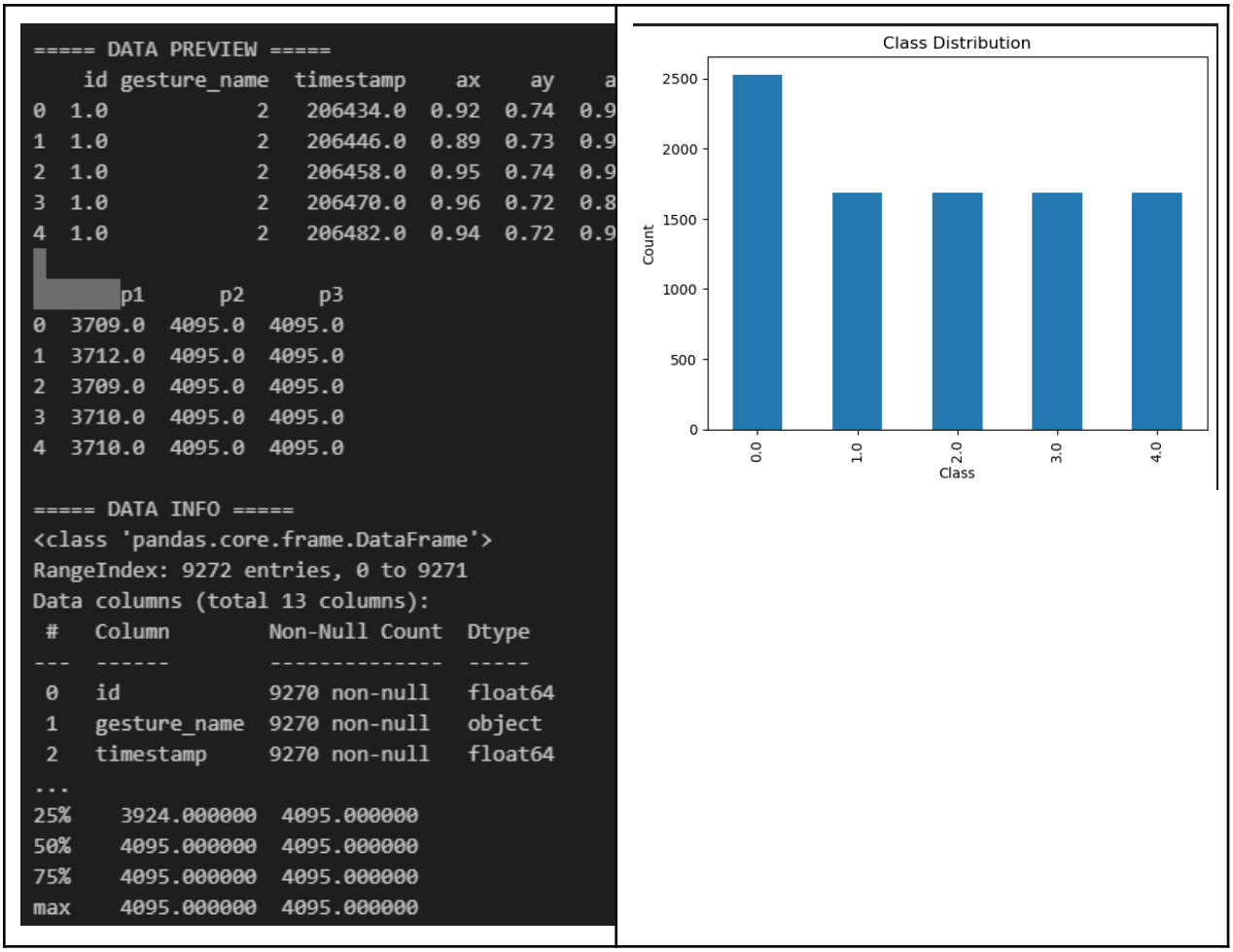
- โหลดข้อมูลจากไฟล์ “dynamic_record_gesture.csv”
- ตรวจสอบข้อมูลด้วย
    - df.head()
    - df.info()
    - Missing Values
    - ค่าสถิติเบื้องต้น (describe())
- ข้อมูลประกอบด้วยสัญญาณจาก
    - Accelerometer: ax, ay, az
    - Gyroscope: gx, gy, gz

- Potentiometer: p0, p1, p2, p3

- มีทั้งหมด 5 คลาส คือ idle (กำมือ), 1, 2, 3, 4
- ทำ Label Encoding เป็นตัวเลข 0–4

```
id,gesture_name,timestamp,ax,ay,az,gx,gy,gz,p0,p1,p2,p3

1,2,206434,0.92,0.74,0.91,-0.23,-2.38,2.28,3859,3709,4095,4095
1,2,206446,0.89,0.73,0.91,-2.03,-4.92,4.78,3867,3712,4095,4095
1,2,206458,0.95,0.74,0.91,-1.09,-5.62,3.35,3877,3709,4095,4095
1,2,206470,0.96,0.72,0.88,-1.13,-5.5,1.04,3873,3710,4095,4095
1,2,206482,0.94,0.72,0.9,-0.95,-4.49,-0.51,3861,3710,4095,4095
1,2,206494,0.92,0.72,0.9,-0.48,-6.29,-0.29,3863,3709,4095,4095
1,2,206506,0.92,0.72,0.9,-0.45,-7.04,-0.24,3843,3708,4095,4095
1,2,206518,0.93,0.73,0.9,0.7,-7.08,-1.55,3894,3717,4095,4095
1,2,206530,0.93,0.74,0.91,0.1,-5.16,-2.38,3870,3721,4095,4095
1,2,206542,0.92,0.74,0.9,-0.75,-5.22,-2.04,3873,3723,4095,4095
```

(รูปที่ 3.2.1 ตัวอย่างข้อมูล dynamic_record_gesture.csv)



```
===== DATA PREVIEW =====
    id gesture_name  timestamp    ax    ay    a
0  1.0            2   206434.0  0.92  0.74  0.9
1  1.0            2   206446.0  0.89  0.73  0.9
2  1.0            2   206458.0  0.95  0.74  0.9
3  1.0            2   206470.0  0.96  0.72  0.8
4  1.0            2   206482.0  0.94  0.72  0.9

      p1      p2      p3
0  3709.0  4095.0  4095.0
1  3712.0  4095.0  4095.0
2  3709.0  4095.0  4095.0
3  3710.0  4095.0  4095.0
4  3710.0  4095.0  4095.0

===== DATA INFO =====
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9272 entries, 0 to 9271
Data columns (total 13 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   id            9270 non-null   float64
 1   gesture_name  9270 non-null   object
 2   timestamp     9270 non-null   float64
...
25%    3924.000000  4095.000000
50%    4095.000000  4095.000000
75%    4095.000000  4095.000000
max    4095.000000  4095.000000
```


(รูปที่ 3.2.1.2 Load data & Label Encoding)

3.2.2 การสร้าง Dataset

เนื่องจากข้อมูลเป็นลักษณะสัญญาณต่อเนื่อง จึงใช้เทคนิค Sliding Window

- Window Size = 50 samples
- Step Size = 25 samples

กล่าวคือ เลื่อนหน้าต่างข้อมูลไปครั้งละ 25 ค่า เพื่อสร้างตัวอย่างใหม่จากข้อมูล 50 จุดต่อหนึ่ง Window จากแต่ละ Window ทำการสกัดคุณลักษณะ (Feature Extraction) ดังนี้

สำหรับแต่ละสัญญาณ 10 ตัว คำนวณ

- Mean
- Standard Deviation
- Maximum

- Minimum
- Range
- RMS (Root Mean Square)

รวมได้ 10 สัญญาณ × 6 ค่า = 60 Features และเพิ่ม Acceleration Magnitude (mean) และGyroscope Magnitude (mean) รวมทั้งหมด 62 Features ต่อหนึ่ง Window

จากนั้นแบ่งข้อมูลเป็น

- Training Set = 80%
- Test Set = 20%
- ใช้ Stratified Sampling เพื่อรักษาสัดส่วนของแต่ละคลาส

```
✓ Total feature names: 62

===== FEATURE MATRIX =====
X shape: (364, 62)
y shape: (364,)
✓ Feature count matched (62)

Example first 10 features: [0.9298     0.01891983 0.96       0.89       0.07       0.92999247
 0.7328     0.01114271 0.77       0.71       ]

===== FEATURE EXPLANATION =====

00 | mean_ax                 = 0.929800
01 | std_ax                  = 0.018920
02 | max_ax                  = 0.960000
03 | min_ax                  = 0.890000
04 | range_ax                = 0.070000
05 | rms_ax                  = 0.929992
06 | mean_ay                 = 0.732800
07 | std_ay                  = 0.011143
08 | max_ay                  = 0.770000
09 | min_ay                  = 0.710000
10 | range_ay                = 0.060000
11 | rms_ay                  = 0.732885
...
60 | mean_acc_magnitude      = 1.481179
61 | mean_gyro_magnitude     = 4.949838
```

(รูปที่ 3.2.2 FEATURE EXTRACTION)

3.2.3 การฝึกสอนโมเดล Random Forest

โมเดลที่เลือกใช้คือ Random Forest เนื่องจาก

- รองรับข้อมูลหลาย Feature ได้ดี
- ลดปัญหา Overfitting เมื่อเทียบกับ Decision Tree เดี่ยว
- ไม่ต้องทำ Feature Scaling

กำหนดพารามิเตอร์หลักดังนี้

- n_estimators = 200
- random_state = 42
- n_jobs = -1

3.2.4 การเตรียมโมเดลสำหรับ Deployment

หลังจากฝึกสอนเสร็จสิ้น ทำการบันทึกโมเดลด้วยไลบรารี joblib

ไฟล์ที่ได้ :

(รูปที่ 3.2.4 model.pkl)

โมเดลนี้สามารถนำไปใช้ในระบบจริง เช่น

- **โหลดโมเดลบนคอมพิวเตอร์**
- **รับค่าจากเซนเซอร์**
- **ทำ Feature Extraction แบบเดียวกับขั้นตอนฝึก**
- **ส่งเข้าโมเดลเพื่อทำ Prediction แบบ Real-Time**

3.3 การแปลงข้อความเป็นเสียงพูด

3.3.1 การเลือก Text-to-Speech Engine

ในโครงงาน ถุงมือแปลภาษามืออัจฉริยะ ระบบจำเป็นต้องมีการแปลงผลลัพธ์การจำแนกท่าทางให้อยู่ในรูปแบบเสียงพูด โดยมีเงื่อนไขสำคัญคือ

- ทำงานได้อย่าง เสถียร
- ไม่ใช้ทรัพยากรระบบมากเกินไป
- มีความหน่วงต่ำ (Low Latency)

- เหมาะสมกับงานลักษณะ ระบบฝังตัว (Embedded System)
- โครงงานนี้เลือกใช้ไลบรารี **pyttsx3** สำหรับภาษา Python เนื่องจากมีข้อดีดังนี้
    - รองรับการทำงานแบบ Offline ไม่ต้องเชื่อมต่ออินเทอร์เน็ต
    - ลดความหน่วงเวลาในการตอบสนอง
    - ใช้งานง่ายและติดตั้งสะดวก
    - สามารถปรับ ความเร็ว (Rate) และ ระดับเสียง (Volume) ได้ตามความเหมาะสม

3.3.2 การเชื่อมต่อกับระบบหลัก

กระบวนการทำงานของระบบมีลำดับขั้นตอนดังนี้

1. ESP32 ทำหน้าที่อ่านค่าจาก potentiometer และเซนเซอร์
2. ส่งข้อมูลผ่าน Serial Communication ไปยังคอมพิวเตอร์
3. ระบบทำการประมวลผลข้อมูล และจำแนกท่าทางด้วยโมเดล Machine Learning
4. ผลลัพธ์การจำแนก (เช่น ตัวเลข 1–4) ถูกส่งต่อไปยังโมดูล Text-to-Speech
5. โมดูล TTS แปลงผลลัพธ์เป็นเสียงพูด และแสดงผลผ่านลำโพง

3.4 การประกอบและบูรณาการระบบ

3.3.1 Data Flow ทั้งระบบ

การไหลของข้อมูลในระบบ เริ่มต้นจากการตรวจจับการเคลื่อนไหวของมือด้วยเซนเซอร์ จากนั้นข้อมูลจะถูกส่งไปประมวลผล และแสดงผลลัพธ์เป็นเสียงพูด ตามลำดับดังนี้

Sensor ⟶ ESP32 ⟶ Serial Communication ⟶ Machine Learning Model ⟶ Text Output ⟶ Text-to-Speech ⟶ Speaker

3.3.2 การทดสอบ End-to-End

การทดสอบแบบ End-to-End ดำเนินการโดย

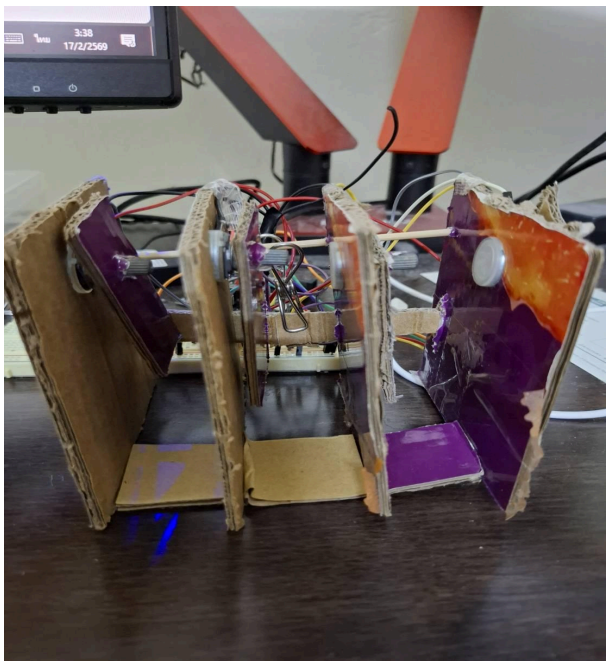
- ทำท่าทางภาษามือ
- ตรวจสอบการรับค่าจากเซนเซอร์
- ตรวจสอบผลการจำแนกจากโมเดล
- ตรวจสอบการแปลงผลลัพธ์เป็นเสียงพูด

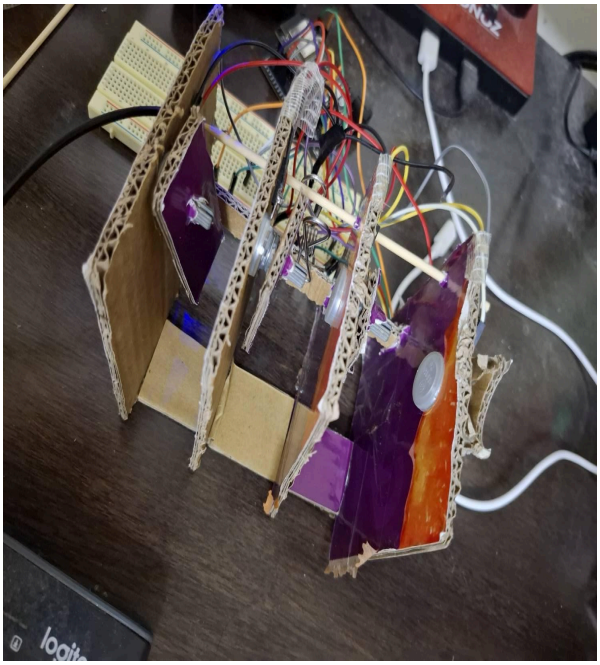
บทที่ 4

ผลการดำเนินงาน

4.1 ฮาร์ดแวร์

4.1.1 ผลลัพท์การสร้างต้นแบบของถุงมือแปลภาษาอัจฉริยะ





(รูปที่ 3.1.2 ภาพชิ้นงาน)

4.1.2 ผลการทดสอบอ่านค่า ADC และ โมดูล GY-521

```
Connected!!
เลือกโหมด (w: ใช้งานจริง , d: เก็บข้อมูล) :       w
กด Enter เมื่อพร้อมขยับมือ...
ข้อมูล :       ax    ay    az    gx    gy    gz  p0    p1    p2    p3
0   0.00  0.00  0.99 -0.15  0.03  0.11   0     0  4095  4095
1  -0.01 -0.00  0.99 -0.14  0.03  0.18   0  2962  4095  4095
2  -0.01  0.02  1.00  0.38  0.26 -0.62   0    68  4095  4095
3  -0.01  0.00  1.00  0.08  0.00  0.22   0     0  4095  4095
4  -0.01 -0.00  0.99  0.22  0.02  0.44   0  4095  4095  4095
..   ...   ...   ...   ...   ...   ... ..   ...   ...   ...
95 -0.01  0.00  0.99 -0.18 -0.03  0.40   0     0  4095  4095
96 -0.01 -0.01  0.99 -0.40  0.15  0.14   0  3018  4095  4095
97 -0.01 -0.00  0.99 -0.11 -0.18 -0.17   0    15  4095  4095
98 -0.01  0.00  0.99 -0.04 -0.14  0.22   0     0  4095  4095
99 -0.01 -0.01  0.98  0.20 -0.01 -0.06   0  4095  4095  4095

[100 rows x 10 columns] ท่าทางที่ตรวจพบ: 0.0 (ความมั่นใจ: 0.51)
ข้อมูล :       ax    ay    az    gx    gy    gz  p0    p1    p2    p3
0  -0.01  0.00  0.99 -0.01  0.02  0.00   0     0  4095  4095
1  -0.01 -0.00  0.99 -0.09 -0.04 -0.01   0  2992  4095  4095
2  -0.00 -0.00  0.99 -0.01 -0.01  0.34   0    61  4095  4095
3  -0.00  0.00  0.99 -0.04 -0.03  0.06   0     0  4095  4095
4  -0.01 -0.00  0.99  0.08 -0.18 -0.07   0  4095  4095  4095
..   ...   ...   ...   ...   ...   ... ..   ...   ...   ...
95 -0.00 -0.00  0.99  0.09  0.05  0.29   0     0  4095  4095
96 -0.01 -0.00  0.99 -0.11  0.20  0.15   0  3123  4095  4095
97 -0.01  0.00  0.99 -0.15  0.02 -0.30   0     0  4095  4095
98 -0.01 -0.00  0.99 -0.01  0.25  0.15   0     0  4095  4095
99 -0.01 -0.01  0.98 -0.18 -0.14  0.00   0  3759  4095  4095

[100 rows x 10 columns] ท่าทางที่ตรวจพบ: 0.0 (ความมั่นใจ: 0.51)
```

(รูปที่ 3.1.4.5 ผลลัพท์การทำงานหลังพิมพ์ ‘w’)

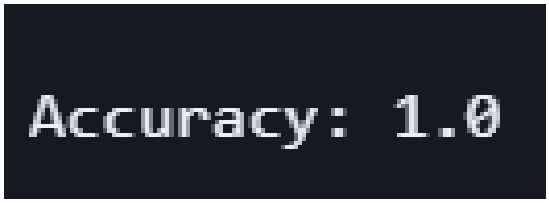
4.1.3 ผลการทดสอบการส่งข้อมูลผ่านโปรโตคอล Serial

```
--- Serial Ports ที่พบ
- PORT : COM3 | DESCRIPTION: USB-SERIAL CH340 (COM3)
Connected!!
เลือกโหมด (w: ใช้งานจริง , d: เก็บข้อมูล) : d
เลือกโหมด (r: เก็บข้อมูลท่าเคลื่อนไหว, s: เก็บข้อมูลท่านิ่ง, q: ออก): r
ชื่อท่า :idle
ID ของท่า :1
เตรียมบันทึกท่า : idle (ID : 1)

กด Enter เมื่อพร้อมขยับมือ...
บันทึกลงไฟล์: ./backend/dynamic_record_gesture.csv
>>> กำลังบันทึก... ขยับมือได้
>>> หยุดบันทึก
บันทึกข้อมูลท่า idle เรียบร้อยแล้ว (168) แถว

เลือกโหมด (w: ใช้งานจริง , d: เก็บข้อมูล) :
```

(รูปที่ 3.1.4.3 ผลลัพท์การทำงานเมื่อพิมพ์ ‘r’)

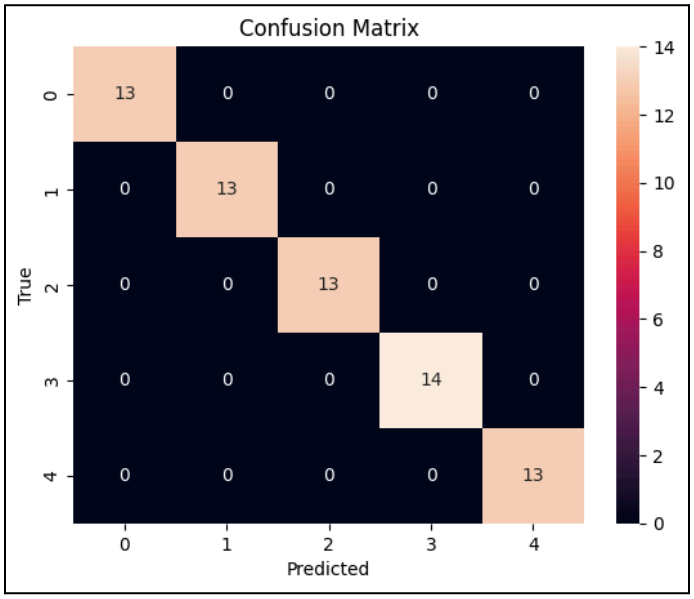
4.2 ปัญญาประดิษฐ์และการเรียนรู้ของเครื่อง

4.2.1 Accuracy



(รูปที่ 4.2.1 Accuracy)

4.2.2 Confusion Matrix



(รูปที่ 4.2.2 Confusion Matrix)

4.2.3 Precision / Recall / F1-score

```
Accuracy: 0.9863013698630136

Classification Report:
              precision    recall  f1-score   support

         0.0       1.00      0.95      0.97        20
         1.0       0.93      1.00      0.96        13
         2.0       1.00      1.00      1.00        13
         3.0       1.00      1.00      1.00        14
         4.0       1.00      1.00      1.00        13

    accuracy                           0.99        73
   macro avg       0.99      0.99      0.99        73
weighted avg       0.99      0.99      0.99        73
```

(รูปที่ 4.2.3 Classification Report)

4.3 การแปลงข้อความเป็นเสียงพูด

ระบบเลือกใช้ไลบรารี pyttsx3 ซึ่งเป็น Text-to-Speech Engine สำหรับภาษา Python ที่สามารถทำงานแบบออฟไลน์ (Offline) ไม่ต้องเชื่อมต่ออินเทอร์เน็ต ลดความหน่วงเวลา (Latency) และเหมาะกับระบบที่ต้องการตอบสนองทันที

การทำงานของโมดูล TTS มีขั้นตอนดังนี้

1. รับผลลัพธ์การจำแนกจากระบบหลัก (เช่น ตัวเลข 1–4)
2. แปลงค่าตัวเลขเป็นข้อความอังกฤษ
3. ส่งข้อความเข้าสู่ TTS Engine
4. แสดงผลเป็นเสียงผ่านลำโพงของคอมพิวเตอร์

ตัวอย่างโค้ดสำหรับโมดูล Text-to-Speech

```python
import pyttsx3

def speak_number(num):
    # สร้าง engine ครั้งเดียว
    _engine = pyttsx3.init()
    voices = _engine.getProperty('voices')
    # ตั้งค่าเสียง (ปรับได้)
    _engine.setProperty("rate", 160)    # ความเร็วพูด
    _engine.setProperty("volume", 1.0) # ความดัง

    """
    รับเลข 1-4 แล้วพูดออกสาโพง
    """
    text_map = {
        1: "one",
        2: "two",
        3: "three",
        4: "four"
    }

    text = text_map.get(num, str(num))
    _engine.say(text)
    _engine.runAndWait()

```

(รูปที่ 4.3 ตัวอย่างโค้ดสำหรับโมดูล Text-to-Speech)

โค้ดดังกล่าวทำหน้าที่สร้าง engine เพียงครั้งเดียว และเรียกใช้ฟังก์ชัน speak_number() เมื่อมีผลลัพธ์ใหม่จากระบบ

4.4 การประกอบและบูรณาการระบบ

4.4.1 End-to-End Latency

End-to-End Latency คือระยะเวลาตั้งแต่ผู้ใช้ทำท่าทางภาษามือ จนกระทั่งระบบแสดงผลเป็นเสียงพูด โดยในโครงงานนี้

จากการทดสอบพบว่าระบบสามารถตอบสนองได้ในระดับเวลาที่เหมาะสมต่อการใช้งานจริง และไม่มีอาการดีเลย์ที่ส่งผลกระทบต่อการสื่อสาร

4.4.2 ความถูกต้องโดยรวม

ความถูกต้องของระบบถูกประเมินจากความสามารถในการ

- ตรวจจับท่าทางได้ถูกต้อง
- จำแนกผลลัพธ์ได้แม่นยำ
- แปลงข้อความเป็นเสียงได้ตรงตามผลลัพธ์

ในส่วนของ Text-to-Speech พบว่าเสียงที่ได้มีความชัดเจน และตรงกับค่าที่ระบบประมวลผล โดยไม่มีความผิดพลาดในการแมปค่าตัวเลขเป็นคำพูด

4.5 วิธีนำไปใช้

ระบบ GloMu สามารถนำไปใช้ในสถานการณ์ที่ต้องการช่วยเหลือผู้บกพร่องทางการได้ยินหรือการพูด โดยผู้ใช้เพียงบิด Potentiometer ที่ต้นแบบถุงมือเป็นทำท่าทางภาษามือ ระบบจะทำการตรวจจับ ประมวลผล และแปลงเป็นเสียงพูดโดยอัตโนมัติ

ขั้นตอนการใช้งานมีดังนี้

1. เปิดระบบและเชื่อมต่อ ESP32 กับคอมพิวเตอร์
2. เรียกใช้งานโปรแกรม Python
3. สวมถุงมือและทำท่าทางที่กำหนด
4. ระบบจะแสดงผลลัพธ์เป็นเสียงผ่านลำโพงทันที

แนวทางการพัฒนาเพิ่มเติมในอนาคตอาจรวมถึง

- รองรับคำศัพท์มากกว่าตัวเลข
- เพิ่มระบบแปลเป็นประโยค
- รองรับหลายภาษา

บทที่ 5
บทสรุปและข้อเสนอแนะ

5.1 สรุปผลการดำเนินงาน

จากการดำเนินงานตามลำดับตั้งแต่ส่วนของการออกแบบฮาร์ดแวร์จำลอง การรับส่งข้อมูลผ่านโปรโตคอล Serial การจัดทำชุดข้อมูลในรูปแบบ csv และการพัฒนาส่วนประมวลผลด้วย Python เพื่อรันระบบหลังบ้าน จนสามารถทำงานออกมาตามวัตถุประสงค์ที่ตั้งไว้ดังนี้

5.1.1. สามารถพัฒนาอุปกรณ์ต้นแบบของถุงมือแปลภาษามืออัจฉริยะที่มีเซนเซอร์ตรวจจับมุม GY-521 และตัวต้านทานปรับค่าได้

5.1.2. สามารถพัฒนาโมเดลการเรียนรู้ของเครื่องสำหรับจำแนกท่าทางภาษามือพื้นฐาน

5.1.3. สามารถพัฒนาระบบแปลงผลการจำแนกเป็นข้อความและเสียงพูดแบบเรียลไทม์

5.2 แนวทางการแก้ไขปัญหา

5.2.1.ปัญหาที่พบ

- โมเดลเกิดปัญหา Overfitting เนื่องจากจำนวนข้อมูลในบางคลาสมีน้อย ทำให้โมเดลจดจำข้อมูลเฉพาะชุดฝึกสอนมากเกินไปและไม่สามารถทั่วไป (Generalize) กับข้อมูลใหม่ได้ดี
- ปัญหาด้านความแข็งแรงของ Hardware สาเหตุมาจากใช้ลังกระดาษทำขึ้นเมื่อนำมาประกอบจะรู้สึกได้เลยว่าไม่แข็งแรง

5.2.2. แนวทางแก้ปัญหา

- เพิ่มปริมาณข้อมูลในแต่ละคลาสให้สมดุลกัน (Data Augmentation หรือเก็บข้อมูลเพิ่มในคลาสที่มีจำนวนน้อย) เพื่อให้โมเดลเรียนรู้รูปแบบได้หลากหลายและลดปัญหา Overfitting
- ประยุกต์ใช้วัสดุเสริมแรงเพื่อเพิ่มความคงตัวให้กับโครงสร้างมือจำลอง โดยมีการใช้เทปกาวเพื่อยึดจุดเชื่อมต่อให้แน่นหนา และนำวัสดุที่มีความแข็งแรงเชิงเส้นมาติดตั้งเป็นแกนคานรองรับโครงสร้าง เพื่อป้องกันการหย่อนคล้อยหรือการเสียรูปของวัสดุ (Anti-deformation) ช่วยให้การเก็บข้อมูลจากเซนเซอร์ในแต่ละท่าทางมีความคงที่

5.3 ข้อเสนอแนะจากการดำเนินงาน

5.3.1 ควรเก็บชุดข้อมูลที่ใหญ่และหลากหลายมากขึ้น

5.3.2 ควรทำให้เสร็จตามระยะที่กำหนดไว้

5.3.3 ควรวางแผนในการดำเนินงานต่อยอดโดยทำให้ระบบนี้เป็นระบบแบบไร้สาย

5.3.4 ในอนาคตจะเปลี่ยนจากการใช้ Potentiometer และโครงสร้างกระดาษ เป็น Flex Sensor และถุงมือผ้าเพื่อให้สวมใส่ได้จริง

บรรณานุกรม

GeeksforGeeks. (2024, February 22). Random forest algorithm in machine learning.http://www.geeksforgeeks.org/machine-learning/random-forest-algorithm-in-machine-learning/

GeeksforGeeks. (2025, February 18). What is machine learning pipeline? https://www.geeksforgeeks.org/blogs/machine-learning-pipeline/

Titichaya Thanamitsomboon. (2026a). 02-1-RTOS-introduction [เอกสารประกอบการสอน].02-1-RTOS-introduction.pdf - Google Drive

Titichaya Thanamitsomboon. (2026b). 02-2-FreeRTOS-on-ESP32 [เอกสารประกอบการสอน].02-2-FreeRTOS-on-ESP32.pdf - Google Drive

Titichaya Thanamitsomboon. (2026c). LAB02-RTOS_queue [เอกสารประกอบการสอน].LAB02-RTOS_queue.pdf - Google Drive

Titichaya Thanamitsomboon. (2026d). 03-1-FreeRTOS-Tasks [เอกสารประกอบการสอน].03-1-FreeRTOS-Tasks.pdf - Google Drive

Titichaya Thanamitsomboon. (2026e). 03-2-FreeRTOS-Q-Mut-Sem-part1 [เอกสารประกอบการสอน]. 03-2-FreeRTOS-Q-Mut-Sem-part1.pdf - Google Drive

Titichaya Thanamitsomboon. (2026f). LAB03-RTOS_BinarySemaphore [เอกสารประกอบการสอน]. LAB03-RTOS_BinarySemaphore.pdf - Google Drive

Titichaya Thanamitsomboon. (2026g). 04-FreeRTOS-Q-Mut-Sem-part2 [เอกสารประกอบการสอน]. 04-FreeRTOS-Q-Mut-Sem-part2.pdf - Google Drive

Titichaya Thanamitsomboon. (2026h). LAB04-RTOS_CountingSemaphore [เอกสารประกอบการสอน]. LAB04-RTOS_CountingSemaphore.pdf - Google Drive

Titichaya Thanamitsomboon. (2026i). 04-FreeRTOS-Q-Mut-Sem-part3 [เอกสารประกอบการสอน]. 04-FreeRTOS-Q-Mut-Sem-part3.pdf - Google Drive

Titichaya Thanamitsomboon. (2026j). LAB05-RTOS_Mutex [เอกสารประกอบการสอน].LAB05-RTOS_Mutex.pdf - Google Drive