ระบบแปลภาษามืออัจฉริยะ

HandSense AI

โดย

นางสาว สรินดา กันภัย รหัสนักศึกษา 67010927 สาขา 1 ปริญญา

นางสาว สุคนธ์ทิพย์ ด้วงวังหิน รหัสนักศึกษา 67010960 สาขา 1 ปริญญา

อาจารย์ที่ปรึกษา

อ. ไพศาล สิทธิโยภาสกุล

รศ. ดร. บุณย์ชนะ ภู่ระหงษ์

รายงานเล่มนี้เป็นส่วนหนึ่งของรายวิชา

01236256 MICROCONTROLLER AND EMBEDDED SYSTEMS

หลักสูตร วิศวกรรมคอมพิวเตอร์ สาขา วิศวกรรมระบบไอโอทีและสารสนเทศ

ประจำภาคการศึกษาที่ 1 ปีการศึกษา 2568

ชื่อโครงงานภาษาไทย ระบบแปลภาษามืออัจฉริยะ

ภาษาอังกฤษ HandSense AI

คณะจัดทำ นางสาว สรินดา กันภัย รหัสนักศึกษา 67010927 สาขา 1 ปริญญา

นางสาว สุคนธ์ทิพย์ ด้วงวังหิน รหัสนักศึกษา 67010960 สาขา 1 ปริญญา

อาจารย์ที่ปรึกษา อ. ไพศาล สิทธิโยภาสกุล

รศ. ดร. บุณย์ชนะ ภู่ระหงษ์

ภาคการศึกษา 1/2568

บทคัดย่อ

โครงการ “HandSense AI: ระบบแปลภาษามืออัจฉริยะ” เป็นโครงการที่มุ่งเน้นการพัฒนานวัตกรรมด้าน ปัญญาประดิษฐ์ (Artificial Intelligence) และ การประมวลผลภาพ (Computer Vision) เพื่อสร้างระบบที่สามารถตรวจจับ จดจำ และแปลภาษามือให้เป็นข้อความภาษาไทยโดยอัตโนมัติ โดยใช้กล้องเป็นอุปกรณ์อินพุตในการจับภาพท่าทางของมือและนิ้ว จากนั้นทำการประมวลผลด้วยโมเดล Deep Learning เพื่อจำแนกและถอดรหัสสัญญาณภาษามือให้เป็นข้อความที่เข้าใจได้ง่าย

โครงการนี้มีวัตถุประสงค์เพื่อลดช่องว่างทางการสื่อสารระหว่างผู้พิการทางการได้ยินและบุคคลทั่วไป ช่วยให้การสื่อสารเป็นไปอย่างเท่าเทียมและสะดวกมากขึ้น อีกทั้งยังสามารถประยุกต์ใช้ได้ในหลากหลายด้าน เช่น การศึกษา การบริการสาธารณะ การแพทย์ และสถานประกอบการ โดยระบบสามารถทำงานแบบ Real-Time และต่อยอดสู่การแปลเป็นเสียงพูดอัตโนมัติในอนาคตได้

กิตติกรรมประกาศ

โครงการ "ระบบแปลภาษามืออัจฉริยะ" นี้สำเร็จลุล่วงตามวัตถุประสงค์ของผู้จัดทำได้ด้วยความช่วยเหลือและการสนับสนุนจากหลายฝ่าย ข้าพเจ้าขอขอบพระคุณอาจารย์ที่ปรึกษาที่ให้คำแนะนำ องค์ความรู้และข้อเสนอแนะที่เป็นประโยชน์ตลอดกระบวนการดำเนินโครงการ นอกจากนี้ขอขอบคุณเพื่อนร่วมทีมที่ร่วมแรง ร่วมใจและทุ่มเทความสามารถในการพัฒนาโครงการให้สำเร็จลุล่วง

คณะผู้จัดทำขอขอบพระคุณอาจารย์สำนักวิชาวิศวกรรมศาสตร์ทุกๆท่าน ที่ประสิทธิ์ประสาทความรู้แก่ ศิษย์ขอขอบคุณบิดามารดาที่คอยให้การสนับสนุนให้กำลังใจและขอขอบคุณ เพื่อนๆทุกคน รวมถึงบุคคลอื่นๆที่ ไม่ได้กล่าวไว้ ณ ที่นี้ที่ให้ความช่วยเหลือในทุกๆ ด้านด้วยดีมาโดย ตลอด จนสำเร็จเป็นโครงงานฉบับนี้

นางสาว สรินดา กันภัย

นางสาว สุคนธ์ทิพย์ด้วงวังหิน

สารบัญ

บทที่ 1

บทนำ

การสื่อสารถือเป็นปัจจัยสำคัญในการดำเนินชีวิตประจำวันของมนุษย์ อย่างไรก็ตาม ผู้พิการทางการได้ยินหรือผู้ที่ใช้ภาษามือยังคงเผชิญกับอุปสรรคในการสื่อสารกับบุคคลทั่วไปที่ไม่เข้าใจภาษามือ ส่งผลให้เกิดข้อจำกัดในการเข้าถึงข้อมูล การศึกษา การทำงาน และการใช้ชีวิตในสังคม การแก้ไขปัญหาดังกล่าวจึงจำเป็นต้องมีเครื่องมือหรือนวัตกรรมที่สามารถเชื่อมโยงช่องว่างทางการสื่อสารระหว่างสองกลุ่มบุคคลนี้ได้อย่างมีประสิทธิภาพ

ในปัจจุบัน เทคโนโลยีปัญญาประดิษฐ์ (Artificial Intelligence: AI) และการประมวลผลภาพ (Computer Vision) ได้เข้ามามีบทบาทอย่างกว้างขวางในการพัฒนาแอปพลิเคชันและอุปกรณ์อัจฉริยะ โดยเฉพาะการประยุกต์ใช้โมเดลการเรียนรู้เชิงลึก (Deep Learning) ที่สามารถตรวจจับและจดจำลักษณะท่าทางหรือวัตถุได้อย่างแม่นยำ เทคโนโลยีดังกล่าวสามารถนำมาใช้เพื่อสร้างระบบที่ช่วยแปลภาษามือให้เป็นข้อความหรือเสียงพูดได้แบบเรียลไทม์ ซึ่งจะเป็นประโยชน์อย่างยิ่งต่อผู้พิการทางการได้ยิน

ดังนั้น โครงการ **“HandSense AI: ระบบแปลภาษามืออัจฉริยะ”** จึงถูกจัดทำขึ้นเพื่อพัฒนาระบบต้นแบบที่สามารถตรวจจับและแปลภาษามือไทยเป็นข้อความภาษาไทยได้อัตโนมัติ โดยใช้กล้องเป็นอุปกรณ์อินพุตและประมวลผลผ่านโมเดล AI ที่ออกแบบมาเฉพาะ จุดมุ่งหมายคือเพื่อลดข้อจำกัดทางการสื่อสาร เพิ่มความสะดวกสบาย และสร้างโอกาสที่เท่าเทียมในการเข้าถึงข้อมูลและการใช้ชีวิตประจำวันของผู้พิการทางการได้ยิน รวมถึงสามารถต่อยอดไปสู่การใช้งานเชิงพาณิชย์และการพัฒนาในระดับอุตสาหกรรมได้ในอนาคต

1.1 วัตถุประสงค์ของโครงงาน

1.1.1 เพื่อลดช่องว่างทางการสื่อสารระหว่างผู้พิการทางการได้ยินและบุคคลทั่วไป

- ระบบสามารถตรวจจับและแปลภาษามือเป็นข้อความได้โดยอัตโนมัติ ทำให้บุคคลทั่วไปสามารถเข้าใจความหมายได้ทันที
- ลดความจำเป็นในการใช้ล่ามภาษามือในสถานการณ์ทั่วไป ช่วยเพิ่มความสะดวกและความเป็นอิสระให้กับผู้พิการทางการได้ยิน
- ส่งเสริมให้การสื่อสารในชีวิตประจำวันมีความเท่าเทียมและไร้อุปสรรค

1.1.2 เพื่อเพิ่มประสิทธิภาพในการจดจำและแปลความหมายของภาษามือด้วยเทคโนโลยี AI (LSTM Model)

- ระบบได้ใช้โมเดลการเรียนรู้เชิงลึกประเภท Bi-LSTM (Bidirectional Long Short-Term Memory) ซึ่งสามารถวิเคราะห์ลำดับการเคลื่อนไหวของมือได้ทั้งในทิศทางอดีตและอนาคต ช่วยให้การแปลความหมายของภาษามือมีความแม่นยำสูงขึ้น แม้ในกรณีที่ผู้แสดงภาษามือมีความแตกต่างกันเล็กน้อย สามารถประมวลผลได้แบบเรียลไทม์ ทำให้ผู้ใช้งานสามารถสื่อสารได้ทันทีโดยไม่เกิดความล่าช้า

1.1.3 เพื่อส่งเสริมการประยุกต์ใช้ในด้านต่าง ๆ

- สามารถนำไปใช้ในภาคการศึกษา เพื่อช่วยให้ครูและนักเรียนที่ใช้ภาษามือสามารถสื่อสารได้สะดวกยิ่งขึ้น
- ประยุกต์ใช้ในภาคการแพทย์และการบริการสาธารณะ เช่น โรงพยาบาล หน่วยงานราชการ หรือศูนย์บริการลูกค้า เพื่ออำนวยความสะดวกแก่ผู้พิการทางการได้ยิน
- เพิ่มโอกาสในการทำงานร่วมกันในสถานประกอบการที่มีพนักงานซึ่งใช้ภาษามือ

1.1.4 เพื่อเป็นแนวทางในการวิจัยและพัฒนาต่อยอดในอนาคต

- ศึกษาความเป็นไปได้ในการขยายระบบจากการแปลเป็นข้อความ ไปสู่การแปลเป็นเสียงพูดอัตโนมัติ
- พัฒนาโมเดลที่สามารถเรียนรู้แบบอัตโนมัติ (Self-Learning Model) เพื่อให้ระบบสามารถปรับตัวตามลักษณะท่าทางของผู้ใช้งานแต่ละคนได้โดยไม่ต้องฝึกใหม่ทั้งหมดเป็นต้นแบบในการสร้างอุปกรณ์หรือซอฟต์แวร์เชิงพาณิชย์ที่ช่วยยกระดับคุณภาพชีวิตของผู้พิการทางการได้ยินและส่งเสริมสังคมที่เท่าเทียม

1.2 ขอบเขตของการดำเนินงาน

1.2.1 การตรวจจับท่าทางมือและนิ้ว

- ระบบสามารถตรวจจับภาพจากกล้องเพื่อระบุ ตำแหน่ง รูปร่าง และการเคลื่อนไหวของมือและนิ้ว ได้อย่างแม่นยำ
- ใช้เทคนิคการประมวลผลภาพ (Computer Vision) และโมเดลการเรียนรู้เชิงลึก (Deep Learning) ในการแยกแยะลักษณะท่าทางที่แตกต่างกัน

1.2.2 การจดจำและแปลภาษามือ

- ระบบสามารถแปลสัญญาณภาษามือให้เป็นข้อความภาษาไทยได้อย่างอัตโนมัติ
- ระบบถูกออกแบบให้สามารถทำงานได้แบบ เรียลไทม์ (Real-time Processing) เพื่อให้ผู้ใช้งานสามารถสื่อสารได้ทันทีโดยไม่เกิดความล่าช้า

1.2.3 การแสดงผลลัพธ์

- แสดงผลการแปลภาษามือเป็นข้อความภาษาไทยบนหน้าจอ LCD ผ่านการเชื่อมต่อกับ Raspberry Pi เพื่อแสดงผลแบบเรียลไทม์

1.2.4 การจำกัดขอบเขตการพัฒนา

- ในระยะเริ่มต้น ระบบจะเน้นการจดจำภาษามือพื้นฐานของไทย เช่น สวัสดี ขอบคุณ ขอโทษ ฉัน รัก เธอ
- ไม่ครอบคลุมการแปลประโยคภาษามือที่ซับซ้อนหรือการเคลื่อนไหวต่อเนื่องหลายท่าพร้อมกัน แต่สามารถต่อยอดได้ในอนาคต

1.3 ขั้นตอนการดำเนินงาน

เป็นออกเป็นทั้งหมด 4 ขั้นตอนคือ

1.3.1 การเก็บและเตรียมข้อมูล (Data Collection & Preprocessing)

- ทำการรวบรวมข้อมูลท่าทางภาษามือ เช่น สวัสดี ขอบคุณ ขอโทษ ฉัน รัก เธอ
- จัดเก็บข้อมูลเป็นรูปภาพหรือวิดีโอจากกล้องเพื่อใช้ในการฝึกสอนโมเดล
- ทำการปรับปรุงคุณภาพข้อมูล เช่น การปรับแสง การตัดส่วนของมือ และการทำ Data Augmentation เพื่อเพิ่มความหลากหลายของชุดข้อมูล
- ใช้ MediaPipe ตรวจจับมือและดึง Landmark (x, y, z) ของมือในแต่ละเฟรม
- จัดการความยาวของลำดับ Landmark ให้คงที่ (Padding / Truncating) และบันทึกเป็นไฟล์ .npz

1.3.2 การสร้างและฝึกสอนโมเดล (Model Training)

- ใช้เทคโนโลยี Deep Learning ด้วย Bi-LSTM (Bidirectional LSTM) เพื่อเรียนรู้ลำดับการเคลื่อนไหวของมือ
- สร้างโมเดลด้วย Keras Sequential Model ประกอบด้วย Bidirectional LSTM Layers , Dropout Layers เพื่อลด Overfitting และ Dense Layers สำหรับจำแนกคลาสภาษามือ
- ใช้ Google Colab เป็นแพลตฟอร์มหลักสำหรับฝึกสอนโมเดลด้วย GPU
- กำหนด Optimizer (Adam) และ Loss Function (Categorical Crossentropy) พร้อมใช้ Early Stopping เมื่อโมเดลไม่พัฒนาบน Validation Set
- ประเมินผลโมเดลด้วยชุดข้อมูล Validation และบันทึกโมเดลที่ฝึกเสร็จ

1.3.3 การพัฒนาและแสดงผลผ่านระบบ (Implementation & Visualization)

- นำโมเดล Bi-LSTM ที่ฝึกเสร็จแล้ว มาใช้งานจริง โดยพัฒนาโปรแกรมด้วย ภาษา Python
- ใช้ Visual Studio Code (VS Code) เพื่อประสานโมเดล AI กับ Node-RED
- ใช้ MQTT/EMQX เป็นตัวกลางในการส่งข้อความระหว่าง AI (LSTM Model), Raspberry Pi, และ LCD
- แสดงผลการแปลภาษามือเป็นข้อความภาษาไทยแบบ Real-Time บน LCD

1.3.4 การปรับปรุงและพัฒนาต่อยอด (Improvement & Future Work)

- วิเคราะห์ข้อจำกัดของระบบ เช่น การแปลที่ยังไม่ครอบคลุมคำศัพท์ซับซ้อนหรือการเคลื่อนไหวต่อเนื่อง
- ศึกษาวิธีเพิ่มฟังก์ชันการทำงาน เช่น การแปลงข้อความเป็นเสียง (Text-to-Speech) เพื่อให้การสื่อสารสมบูรณ์มากยิ่งขึ้น
- วางแผนการพัฒนาเพื่อให้ระบบสามารถทำงานได้ครอบคลุมในอนาคต

โดยทั้ง 4 ขั้นตอนสามารถ แสดงด้วยตารางการดำเนินงานดัง ตารางที่ 1.1

ตารางที่1.1 แผนการดำเนินงานโครงงาน

ขั้นตอนการดำเนินงาน	สัปดาห์ที่			
	1	2	3	4
1. เสนอโครงร่าง	✓			
2. เขียนโครงงานเพื่อพิจารณางบประมาณ	✓			
3. เตรียมอุปกรณ์สำหรับการศึกษา		✓	✓	
4. ดำเนินการสร้างตัวชิ้นงานและดำเนินการทดลอง		✓	✓	
5. สรุปผล			✓	
6. สรุปอภิปรายผลและข้อเสนอแนะ				✓
7. จัดทำรายงานและนำเสนอผลงาน				✓

1.4 สถานที่ดำเนินงาน

- อาคาร 12 ชั้น ห้อง 1008 สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

1.5 ผลที่คาดว่าจะได้รับ

- ได้ระบบต้นแบบที่สามารถตรวจจับและแปลภาษามือไทยเป็นข้อความภาษาไทยได้อย่างอัตโนมัติ โดยมีความแม่นยำและสามารถทำงานแบบเรียลไทม์
- ผู้พิการทางการได้ยินสามารถสื่อสารกับบุคคลทั่วไปได้สะดวกมากขึ้น โดยไม่จำเป็นต้องมีล่ามภาษามือในทุกสถานการณ์ ช่วยลดช่องว่างทางการสื่อสารในสังคม

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

ในบทที่ 2 นี้กล่าวถึงทฤษฎีที่เกี่ยวข้องกับโครงงานระบบไอโอทีทั้งหมด โดยแบ่งเป็นหัวข้อย่อย ดังนี้

2.1 โปรแกรม Google Colab

Google Colab (Google Collaboratory) เป็นแพลตฟอร์มออนไลน์ที่ให้บริการฟรีจาก Google สำหรับการเขียนและรันโค้ดภาษา Python ผ่านเว็บเบราว์เซอร์ โดยไม่จำเป็นต้องติดตั้งโปรแกรมเพิ่มเติมในเครื่องผู้ใช้ เหมาะสำหรับการเรียนรู้ การทดลองทางวิทยาการข้อมูล (Data Science), ปัญญาประดิษฐ์ AI, และการประมวลผลเชิงลึก (Deep Learning) เพราะรองรับการทำงานร่วมกับ GPU และ TPU เพื่อเพิ่มประสิทธิภาพการประมวลผลได้อย่างรวดเร็ว

ภาพที่ 2.1 Google Colab

2.1.1 คุณสมบัติหลักของ Google Colab

- ทำงานบนระบบคลาวด์ (Cloud-based) ผู้ใช้สามารถเขียนและรันโค้ดได้ทุกที่ ทุกเวลา โดยไม่ต้องติดตั้งโปรแกรมเพิ่มเติม เพียงเข้าสู่ระบบด้วยบัญชี Google
- รองรับภาษา Python สามารถใช้เขียนโปรแกรมภาษา Python ได้อย่างเต็มรูปแบบ เหมาะสำหรับงานด้านการเรียนรู้ของเครื่องและข้อมูลขนาดใหญ่
- รองรับการใช้งาน GPU และ TPU ช่วยเพิ่มความเร็วในการประมวลผลโมเดล AI โดยสามารถเลือกประเภทของหน่วยประมวลผลได้ตามความต้องการ

- เชื่อมต่อกับ Google Drive ได้โดยตรง ช่วยให้สามารถบันทึก แก้ไข และเข้าถึงไฟล์หรือชุดข้อมูลต่าง ๆ ได้สะดวกและปลอดภัย
- รองรับการทำงานร่วมกันแบบเรียลไทม์ (Collaborative Work) ผู้ใช้หลายคนสามารถเปิดและแก้ไขไฟล์โน้ตบุ๊กเดียวกันได้พร้อมกัน เหมาะสำหรับการทำงานเป็นทีม
- รองรับการติดตั้งไลบรารีเพิ่มเติม ผู้ใช้สามารถติดตั้งไลบรารีภายนอกเพิ่มเติมได้โดยใช้คำสั่ง !pip install เช่น TensorFlow, OpenCV, scikit-learn เป็นต้น

2.1.2 องค์ประกอบหลักของ Google Colab

- **Code Cell (เซลล์โค้ด)**

  พื้นที่สำหรับเขียนและรันโค้ดภาษา Python โดยแต่ละเซลล์สามารถทำงานแยกจากกันได้

- **Text Cell (เซลล์ข้อความ)**

  ใช้สำหรับเขียนคำอธิบาย บันทึก หรือใส่สูตรทางคณิตศาสตร์ด้วย Markdown หรือ LaTeX

- **Output Area (พื้นที่แสดงผลลัพธ์)**

  ส่วนที่แสดงผลลัพธ์ของโค้ด เช่น ข้อความ ตาราง กราฟ หรือภาพ

- **File System (ระบบไฟล์)**

  ใช้จัดการไฟล์ เช่น การอัปโหลด ดาวน์โหลด หรือเชื่อมต่อกับ Google Drive เพื่อเข้าถึงข้อมูล

- **Runtime Environment (สภาพแวดล้อมการทำงาน)**

  เป็นส่วนที่ใช้ในการประมวลผลโค้ด โดยสามารถเลือกได้ว่าจะใช้ CPU, GPU หรือ TPU

- **Integration Tools (เครื่องมือเชื่อมต่อ)**

  เป็นเครื่องมือที่ช่วยเชื่อมต่อกับบริการภายนอก เช่น GitHub, BigQuery หรือ API อื่น ๆ เพื่อดึงข้อมูลและบันทึกผลลัพธ์ได้อย่างสะดวก

2.1.3 การใช้งาน Google Colab

2.1.3.1 เริ่มต้นใช้งาน

- เข้าสู่ระบบด้วยบัญชี Google
- สร้างโน้ตบุ๊กใหม่ (New Notebook) หรือเปิดไฟล์โน้ตบุ๊กที่มีอยู่แล้วจาก Google Drive หรือ GitHub

2.1.3.2การเขียนและรันโค้ด

- พิมพ์โค้ด Python ลงใน Code Cell
- กดปุ่ม Run หรือกด Shift + Enter เพื่อรันโค้ด
- ดูผลลัพธ์ใน Output Area ของแต่ละเซลล์

2.1.3.3การจัดการไฟล์และข้อมูล

- อัปโหลดไฟล์จากเครื่องผู้ใช้โดยตรงหรือเชื่อมต่อกับ Google Drive
- อ่านและบันทึกข้อมูล เช่น CSV, Excel หรือไฟล์ภาพ

2.1.3.4การติดตั้งไลบรารีเพิ่มเติม

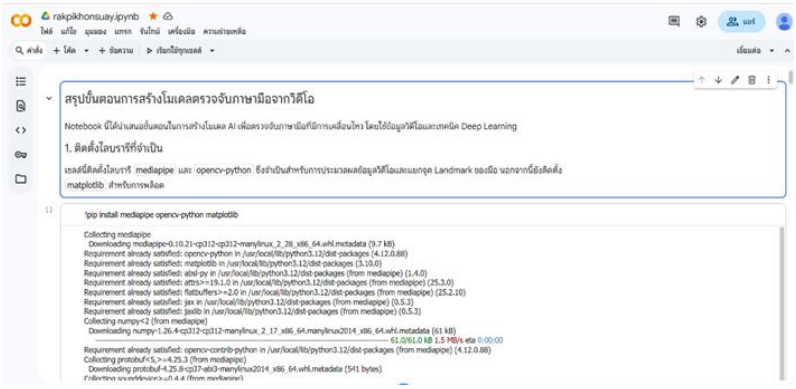
- ใช้คำสั่ง !pip install <ชื่อไลบรารี> เพื่อติดตั้งไลบรารี Python ที่ต้องการใช้งาน

2.1.3.5การใช้งาน GPU/TPU

- เข้าไปที่เมนู Runtime > Change runtime type
- เลือก Hardware accelerator เป็น GPU หรือ TPU เพื่อเพิ่มความเร็วในการประมวลผล

2.1.3.6การทำงานร่วมกัน

- แชร์โน้ตบุ๊กกับผู้ใช้คนอื่นผ่าน Google Drive
- สามารถแก้ไขโค้ดและดูผลลัพธ์แบบเรียลไทม์ร่วมกัน
- ใช้ MQTT/EMQX ในการเชื่อมต่ออุปกรณ์ IoT เช่น กล้อง, Raspberry Pi และ LCD เพื่อส่งและรับข้อความภาษามือแบบ Real-Time



ภาพที่ 2.1.3 การใช้งาน Google Colab

2.1.4 การใช้งาน LSTM ใน Google Colab สำหรับ HandSense AI

เพื่อให้ระบบ HandSense AI สามารถจดจำท่าทางภาษามือแบบต่อเนื่องได้ ระบบจึงนำโมเดล LSTM (Long Short-Term Memory) มาใช้ โดย LSTM เป็นโครงข่ายประสาทเทียมชนิด Recurrent Neural Network (RNN) ที่เหมาะสมกับข้อมูลลำดับเวลา เช่น การเคลื่อนไหวของมือในแต่ละเฟรม

2.1.4.1 เตรียมข้อมูลเป็น Sequence

- ข้อมูลภาพจากกล้องถูกประมวลผลด้วย Mediapipe เพื่อดึง keypoints ของมือ แต่ละเฟรม
- keypoints เหล่านี้จะถูกจัดเก็บเป็นลำดับของเฟรม (Sequence) เช่น 30 เฟรมต่อคำศัพท์
- แต่ละเฟรมประกอบด้วยคุณลักษณะของจุดบนมือ (x, y, z)

2.1.4.2 สร้างและฝึกสอนโมเดล LSTM

- ใช้ Google Colab เป็นแพลตฟอร์มในการสร้างโมเดล LSTM ด้วย TensorFlow/Keras
- โมเดลประกอบด้วยหลายชั้น LSTM เพื่อเรียนรู้ลักษณะการเคลื่อนไหวของมือในแต่ละคำศัพท์
- ใช้ Dense Layer และ activation function softmax เพื่อจำแนกคำศัพท์ภาษามือ
- ฝึกสอนโมเดลด้วยชุดข้อมูลที่เตรียมไว้ โดย Google Colab จะใช้ GPU/TPU เพื่อเพิ่มความเร็วในการประมวลผล

2.1.4.3การประมวลผลแบบเรียลไทม์

- หลังจากโมเดลถูกฝึกเสร็จ ระบบสามารถรับ sequence ของ keypoints จากกล้องแบบเรียลไทม์
- ส่งข้อมูล sequence เข้าโมเดล LSTM เพื่อทำนายคำศัพท์ภาษามือ
- ผลลัพธ์ที่ได้สามารถส่งต่อไปยัง Node-RED ผ่าน MQTT/EMQX เพื่อนำไปแสดงผลบน LCD หรืออุปกรณ์อื่น ๆ

2.1.4.4 ข้อดีของการใช้ Bi-LSTM

- สามารถ จำแนกลำดับท่าทางที่ต่อเนื่องได้ดี ทั้งจากข้อมูลอดีตและอนาคต ทำให้แม่นยำ แม้ผู้ใช้แต่ละคนมีลักษณะการเคลื่อนไหวต่างกันเล็กน้อย
- รองรับการประมวลผลแบบ เรียลไทม์ (Real-Time) ทำให้ผู้ใช้สามารถสื่อสารได้ทันที
- มีความยืดหยุ่นสูง สามารถต่อยอดไปยัง การแปลประโยคภาษามือต่อเนื่อง หรือการทำ Text-to-Speech ได้ในอนาคต

2.2 Visual Studio Code

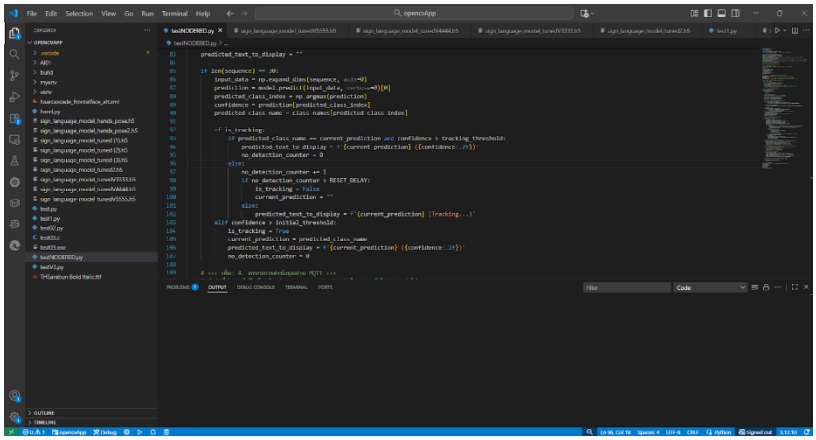
Visual Studio Code (VS Code) คือ โปรแกรมแก้ไขโค้ด (Source Code Editor) ที่พัฒนาโดยบริษัท Microsoft มีจุดเด่นที่ใช้งานง่าย ขนาดเล็ก และสามารถปรับแต่งได้ตามต้องการ ผ่านระบบ ส่วนขยาย (Extensions) ที่ช่วยเพิ่มความสามารถให้รองรับภาษาต่าง ๆ เช่น Python, C/C++, JavaScript, HTML, CSS และภาษาอื่น ๆ อีกมากมาย

โปรแกรมนี้ได้รับความนิยมอย่างมากในหมู่นักศึกษาและนักพัฒนา เพราะมี ระบบดีบัก (Debugger) สำหรับตรวจสอบข้อผิดพลาดของโปรแกรมแบบเรียลไทม์, Terminal ในตัว สำหรับรันคำสั่งได้ทันทีโดยไม่ต้องสลับหน้าต่าง, และ Git Integration สำหรับจัดการซอร์สโค้ดร่วมกันบน GitHub หรือ GitLab นอกจากนี้ Visual Studio Code ยังสามารถทำงานได้ทั้งในระบบปฏิบัติการ Windows, macOS และ Linux จึงเป็นเครื่องมือที่เหมาะสำหรับทั้งการเรียนรู้และการพัฒนาโปรเจกต์ทางด้าน IoT, Embedded Systems, และ Software Development

ภาพที่ 2.2 Visual Studio Code

2.2.1 ส่วนประกอบหลักของหน้าต่างโปรแกรม Visual Studio Code

- Activity Bar (แถบกิจกรรม) : อยู่ทางด้านซ้ายสุดของหน้าจอ ใช้สำหรับสลับระหว่างมุมมองต่าง ๆ เช่น Explorer, Search, Source Control, Run and Debug และ Extensions
- Side Bar (แถบด้านข้าง) : แสดงรายการไฟล์และโฟลเดอร์ในโปรเจกต์ รวมถึงเครื่องมือเสริมตามมุมมองที่เลือกจาก Activity Bar
- Editor (พื้นที่แก้ไขโค้ด) : ส่วนกลางของหน้าจอ ใช้สำหรับเขียน แก้ไข และดูโค้ด สามารถเปิดหลายไฟล์พร้อมกันแบบแท็บ (Tabs) ได้
- Panel (แผงด้านล่าง) : ใช้แสดงผลการรันโปรแกรม ข้อผิดพลาด (Errors), Output, Debug Console หรือ Terminal
- Status Bar (แถบสถานะ) : อยู่ด้านล่างสุด แสดงข้อมูลของไฟล์ เช่น ภาษาที่ใช้ บรรทัดปัจจุบัน หรือสถานะการเชื่อมต่อกับ Git



ภาพที่ 2.2.1 ส่วนประกอบหลักของ Visual Studio Code

2.3 Node – Red

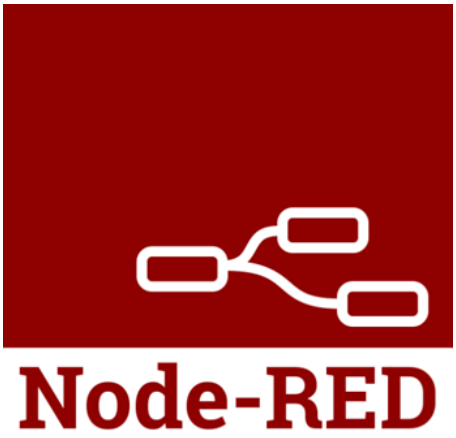
Node-RED เป็นเครื่องมือพัฒนาซอฟต์แวร์แบบโอเพ่นซอร์สที่ใช้หลักการ Flow-based Programming (การเขียนโปรแกรมเชิงโฟลว์) ช่วยให้ผู้ใช้สามารถเชื่อมโยงฮาร์ดแวร์, API, และบริการออนไลน์ต่างๆ เข้าด้วยกันได้โดยไม่ต้องเขียนโค้ดจำนวนมาก

Node-RED พัฒนาโดย IBM และทำงานบน Node.js ทำให้สามารถรันได้บนหลายแพลตฟอร์ม (เช่น Windows, Linux, Raspberry Pi, Cloud) โครงสร้างหลักของมันคือการใช้ "โหนด" (Nodes) ซึ่งแบ่งเป็นโหนดรับข้อมูล (Input), โหนดประมวลผล (Processing), และโหนดส่งข้อมูล (Output) มาเชื่อมต่อกันจนเกิดเป็น "โฟลว์" (Flow) หรือกระบวนการทำงานอัตโนมัติ

นอกจากนี้ Node-RED ยังสามารถขยายความสามารถได้โดยติดตั้งโหนดเพิ่มเติมจาก Library และสามารถสร้างหน้า Dashboard เพื่อแสดงผลข้อมูล (เช่น กราฟ) หรือสร้างปุ่มควบคุมได้

2.3.1 คุณสมบัติหลักของ Node-RED

การพัฒนาแบบ Low-Code: ใช้หลักการลากและวาง (Drag-and-Drop) ในการออกแบบโฟลว์การทำงาน รองรับโหนดหลากหลายประเภท: มีโหนดมาตรฐานที่รองรับการเชื่อมต่อ MQTT, HTTP, WebSockets, Database และอื่นๆ สามารถขยายเพิ่มเติมได้: ผู้ใช้สามารถสร้างโหนดของตนเองหรือใช้โหนดเสริมจาก Node-RED Library รองรับ IoT และ Edge Computing: เหมาะสำหรับการพัฒนาโซลูชัน IoT โดยสามารถรันบน Raspberry Pi, Docker, Kubernetes และแพลตฟอร์มคลาวด์ต่างๆ ใช้ JavaScript และ Node.js: พัฒนาโดยใช้ Node.js และสามารถเขียนโค้ดเพิ่มเติมใน JavaScript ได้



ภาพที่ 2.3 Node-red

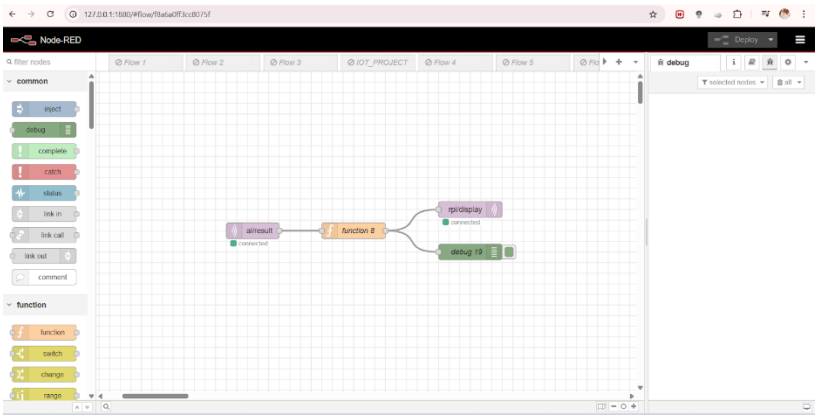
2.3.2 องค์ประกอบหลักของ Node-RED

- โหนด (Nodes) มี โหนดอินพุต (Input Nodes): รับข้อมูลจากแหล่งภายนอก เช่น HTTP Request, MQTT, WebSockets
- โหนดประมวลผล (Processing Nodes): ใช้สำหรับแปลงข้อมูล เช่น ฟังก์ชัน JavaScript, JSON, 3.Switch โหนดเอาต์พุต (Output Nodes): ส่งข้อมูลออกไปยังปลายทาง เช่น MQTT Broker, Database, API
- โฟลว์ (Flows) เป็นชุดของโหนดที่เชื่อมต่อกันเพื่อสร้างกระบวนการทำงาน
- แดชบอร์ด (Dashboard) ใช้สำหรับสร้าง UI แสดงผลข้อมูล เช่น กราฟ แผนภูมิ และปุ่มควบคุม

2.3.3 การใช้งาน Node-RED กับ IoT และระบบอัตโนมัติ

Node-RED นิยมใช้ในงานด้าน IoT, การเชื่อมต่อเซนเซอร์ และระบบอัตโนมัติ เช่น

- Home Automation: ควบคุมอุปกรณ์อัจฉริยะผ่าน MQTT และ Home Assistant
- Industrial IoT (IIoT): ใช้ในการเชื่อมต่อ SCADA, OPC-UA, Modbus
- API Integration: เชื่อมต่อ REST API, WebSockets และฐานข้อมูล
- Machine Learning & AI: ประมวลผลข้อมูลร่วมกับ TensorFlow.js และ AI API



ภาพที่ 2.3.2 องค์ประกอบหลักของ Node-RED

2.4 MQTT (Message Queuing Telemetry Transport)

MQTT เป็นโปรโตคอลการสื่อสารแบบ publish/subscribe ที่ออกแบบมาสำหรับการส่งข้อมูลที่มีขนาดเล็กผ่านเครือข่ายที่มีข้อจำกัดด้านแบนด์วิดท์หรือพลังงาน เช่น ระบบ Internet of Things (IoT) และเครือข่ายที่มีการเชื่อมต่อแบบไร้สายที่ไม่เสถียร โปรโตคอลนี้ถูกพัฒนาโดย IBM ในปี 1999 และได้รับความนิยมอย่างแพร่หลายเนื่องจากมีการใช้ทรัพยากรต่ำ มีโครงสร้างที่เรียบง่าย และสามารถใช้งานได้ในระบบที่มีข้อจำกัดด้านพลังงาน เช่น อุปกรณ์ IoT, ระบบเซ็นเซอร์ไร้สาย และอุปกรณ์สมาร์ทโฮม

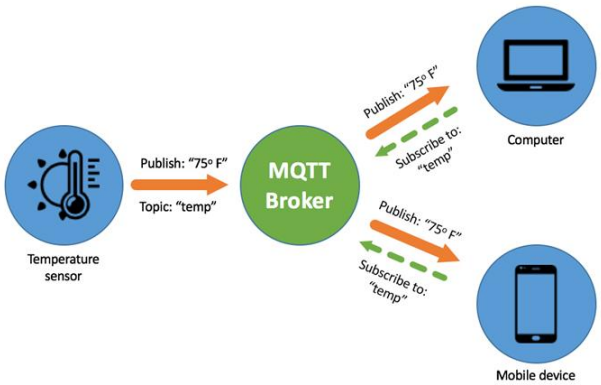
2.4.1 หลักการทำงานของ MQTT

- MQTT ใช้โมเดล publish/subscribe (การเผยแพร่และการสมัครรับข้อมูล) ซึ่งแตกต่างจากการสื่อสารแบบไคลเอนต์-เซิร์ฟเวอร์ทั่วไป เช่น HTTP โดยมีองค์ประกอบหลัก 3 ส่วน ดังนี้
- Publisher (ผู้เผยแพร่ข้อมูล) เป็นอุปกรณ์ที่ส่งข้อมูลไปยัง Broker ไม่ต้องรู้ว่ามีใครจะรับข้อมูล เพียงแค่ส่งไปยังหัวข้อ (Topic) ที่กำหนด ตัวอย่าง: เซ็นเซอร์อุณหภูมิที่ส่งค่ามายัง MQTT Broker
- Broker (ตัวกลางจัดการข้อมูล) ทำหน้าที่รับข้อมูลจาก Publisher และกระจายไปยัง Subscriber ที่สมัครรับข้อมูลอยู่ เป็นหัวใจหลักของ MQTT ซึ่งช่วยให้ระบบมีความยืดหยุ่นสูง ตัวอย่าง: Mosquitto หรือ EMQX เป็น MQTT Broker ยอดนิยม
- Subscriber (ผู้รับข้อมูล) เป็นอุปกรณ์ที่สมัครรับข้อมูลจากหัวข้อที่ต้องการจะได้รับข้อมูลเมื่อ Publisher ส่งข้อมูลมายัง Broker ตัวอย่าง: แอปพลิเคชันบนมือถือที่สมัครรับค่าจากเซ็นเซอร์อุณหภูมิ

2.4.2 ตัวอย่างการใช้งาน MQTT

- สมาร์ทโฮม (Smart Home) ใช้ MQTT เพื่อควบคุมอุปกรณ์ภายในบ้าน เช่น เปิด-ปิดไฟ แสดงค่าจากเซ็นเซอร์อุณหภูมิ หรือควบคุมเครื่องปรับอากาศผ่านมือถือ
- อุตสาหกรรม (Industrial IoT - IIoT)ใช้ MQTT ในการตรวจสอบสถานะเครื่องจักรและการแจ้งเตือนเมื่อพบปัญหา
- เกษตรอัจฉริยะ (Smart Farming) ใช้ MQTT ส่งข้อมูลอุณหภูมิ ความชื้น หรือค่าฝนตกจากเซ็นเซอร์ไปยังเซิร์ฟเวอร์เพื่อช่วยวิเคราะห์สภาพแวดล้อม

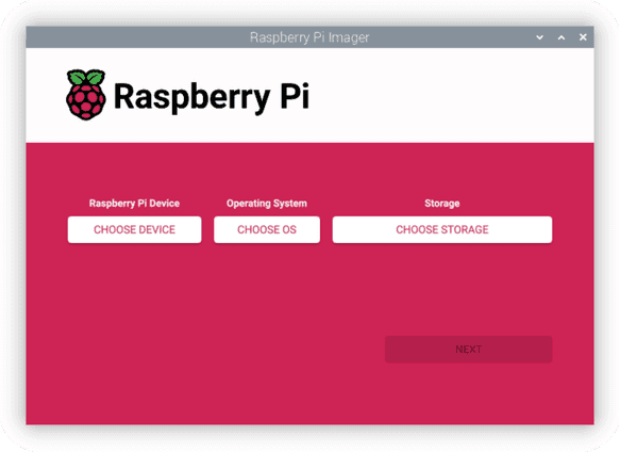
- ระบบติดตามตำแหน่ง (GPS Tracking) ใช้ MQTT เพื่อส่งข้อมูลตำแหน่ง GPS ของรถยนต์หรือยานพาหนะไปยังศูนย์ควบคุม



ภาพที่ 2.4.1 หลักการทำงาน MQTT

2.5 Raspberry Pi Imager

Raspberry Pi Imager คือ ซอฟต์แวร์เครื่องมือสำหรับติดตั้งระบบปฏิบัติการ (Operating System) ลงในหน่วยความจำของบอร์ด Raspberry Pi เช่น MicroSD Card หรือ USB Drive โดยโปรแกรมนี้พัฒนาอย่างเป็นทางการโดย Raspberry Pi Foundation เพื่อช่วยให้ผู้ใช้งานสามารถติดตั้งระบบปฏิบัติการได้อย่างง่ายดาย สะดวก และปลอดภัย โดยไม่ต้องใช้ขั้นตอนที่ซับซ้อนเหมือนในอดีต ก่อนที่จะมี Raspberry Pi Imager การติดตั้งระบบปฏิบัติการบน Raspberry Pi จำเป็นต้องดาวน์โหลดไฟล์อิมเมจ (.img) จากเว็บไซต์ จากนั้นใช้โปรแกรมอื่น เช่น balenaEtcher หรือ Win32DiskImager ในการเขียนลงการ์ดหน่วยความจำ ซึ่งมีขั้นตอนหลายขั้นและอาจเกิดข้อผิดพลาดได้ง่าย โดยเฉพาะกับผู้เริ่มต้น ดังนั้น Raspberry Pi Imager จึงถูกสร้างขึ้นเพื่อให้กระบวนการทั้งหมดรวมอยู่ในโปรแกรมเดียว



ภาพที่ 2.5 Raspberry Pi Imager

2.5.1 หน้าที่และความสามารถของ Raspberry Pi Imager

2.5.1.1 ดาวน์โหลดและติดตั้งระบบปฏิบัติการโดยอัตโนมัติ ผู้ใช้สามารถเลือกประเภทของระบบปฏิบัติการที่ต้องการติดตั้งได้จากเมนูภายในโปรแกรม เช่น

- Raspberry Pi OS (32-bit / 64-bit)
- Ubuntu Server / Ubuntu Desktop
- Kali Linux, RetroPie, Home Assistant OS
- หรือเลือก “Use custom” เพื่อติดตั้งอิมเมจอื่น ๆ ที่ผู้ใช้มีอยู่เอง

2.5.1.2เขียนระบบปฏิบัติการลงหน่วยความจำ (Flashing)

- เมื่อเลือก OS แล้ว โปรแกรมจะเขียนไฟล์ระบบลงใน MicroSD Card หรือ USB Drive โดยอัตโนมัติ พร้อมตรวจสอบความสมบูรณ์ของข้อมูลเพื่อป้องกันข้อผิดพลาดระหว่างการติดตั้ง

2.5.1.3 รองรับการตั้งค่าขั้นต้น (Advanced Options)

ผู้ใช้สามารถกำหนดค่าพื้นฐานของระบบล่วงหน้าได้ เช่น

- ตั้งชื่อเครื่อง (Hostname)
- เปิดใช้งาน SSH
- ตั้งชื่อผู้ใช้และรหัสผ่าน
- เชื่อมต่อ Wi-Fi โดยระบุชื่อเครือข่าย (SSID) และรหัสผ่าน
- ตั้งค่าภูมิภาคและภาษาที่ใช้งาน (Locale, Keyboard, Timezone) เมื่อเขียนระบบเสร็จ Raspberry Pi จะพร้อมใช้งานทันทีโดยไม่ต้องตั้งค่าใหม่อีก

2.5.1.3ใช้งานง่ายและรองรับทุกระบบปฏิบัติการหลัก

- Raspberry Pi Imager สามารถติดตั้งได้ทั้งบน Windows, macOS และ Linux โดยมีหน้าตาโปรแกรมที่เรียบง่ายและคล้ายกันในทุกแพลตฟอร์ม ทำให้ผู้ใช้ไม่ต้องเรียนรู้ใหม่เมื่อเปลี่ยนระบบ

2.5.2 ขั้นตอนการใช้งานโดยสรุป

- ดาวน์โหลดโปรแกรม Raspberry Pi Imager จากเว็บไซต์ทางการ https://www.raspberrypi.com/software
- เปิดโปรแกรมขึ้นมา แล้วเลือก CHOOSE OS เพื่อเลือกระบบปฏิบัติการ
- CHOOSE STORAGE เพื่อเลือก MicroSD Card หรือ USB Drive ที่ต้องการติดตั้ง

- คลิก WRITE เพื่อเริ่มกระบวนการเขียนระบบลงในหน่วยความจำ
- รอจนโปรแกรมขึ้นข้อความว่า “Write Successful” จึงนำการ์ดไปเสียบที่บอร์ด Raspberry Pi เพื่อเริ่มใช้งาน

2.5.3 ประโยชน์ของ Raspberry Pi Imager

- ช่วยให้ผู้เริ่มต้นสามารถติดตั้งระบบได้อย่างรวดเร็ว โดยไม่ต้องใช้คำสั่งซับซ้อน
- ลดความเสี่ยงในการเขียนข้อมูลผิดหรือระบบเสียหาย
- สนับสนุนระบบปฏิบัติการหลากหลายแบบในโปรแกรมเดียว
- มีฟังก์ชันปรับแต่งล่วงหน้าสำหรับโครงการ IoT และงาน Embedded
- ประหยัดเวลาและขั้นตอนในการเตรียมเครื่อง Raspberry Pi ให้พร้อมใช้งาน

2.6 Raspberry Pi 4

Raspberry Pi 4 Model B คือ ไมโครคอมพิวเตอร์ขนาดเล็ก (Single Board Computer: SBC) ที่ถูกพัฒนาโดยมูลนิธิ Raspberry Pi Foundation ประเทศอังกฤษ มีจุดประสงค์หลักเพื่อส่งเสริมการเรียนรู้ทางด้านคอมพิวเตอร์และอิเล็กทรอนิกส์ในราคาย่อมเยา แต่มีประสิทธิภาพสูงเพียงพอที่จะใช้งานได้เหมือนคอมพิวเตอร์ทั่วไป

Raspberry Pi 4 มาพร้อมกับ หน่วยประมวลผลแบบ Quad-Core 64-bit ARM Cortex-A72 ความเร็ว 1.5 GHz, หน่วยความจำ RAM หลายขนาดให้เลือกตั้งแต่ 2 GB, 4 GB และ 8 GB, และยังมีพอร์ต USB 3.0, พอร์ต Gigabit Ethernet, พอร์ต HDMI จำนวน 2 ช่อง (รองรับจอแสดงผลความละเอียดสูง 4K) รวมถึง พอร์ต GPIO (General Purpose Input/Output) จำนวน 40 ขา สำหรับเชื่อมต่อกับอุปกรณ์ภายนอก เช่น เซนเซอร์, มอเตอร์, LED และโมดูลต่าง ๆ
ระบบปฏิบัติการที่นิยมใช้กับ Raspberry Pi คือ Raspberry Pi OS (ชื่อเดิมคือ Raspbian) ซึ่งเป็นระบบปฏิบัติการที่พัฒนามาเฉพาะสำหรับบอร์ดนี้ โดยมีพื้นฐานมาจากระบบปฏิบัติการ Linux ทำให้สามารถติดตั้งซอฟต์แวร์ได้หลากหลาย เช่น Python, Node-RED, C/C++, หรือเครื่องมือจำลองการเชื่อมต่อ IoT ต่าง ๆ ได้อย่างสะดวก

ภาพที่ 2.6 Raspberry Pi 4 Model B

2.6.1 คุณสมบัติเด่นของ Raspberry Pi 4

- ประสิทธิภาพสูงขึ้นจากรุ่นก่อนหน้า ด้วย CPU แบบ ARM Cortex-A72 ทำให้ประมวลผลเร็วกว่า Raspberry Pi 3 ถึงประมาณ 3 เท่า รองรับงานที่ซับซ้อนมากขึ้น เช่น การประมวลผลภาพ, ระบบ AI เบื้องต้น, หรือการจำลอง Server
- รองรับการเชื่อมต่อหลากหลายรูปแบบ มีพอร์ต USB 2.0 และ USB 3.0, HDMI คู่, ช่องเสียบ MicroSD สำหรับระบบปฏิบัติการ, รวมถึง Bluetooth 5.0 และ Wi-Fi Dual Band (2.4/5 GHz)
- เหมาะสำหรับงาน IoT และ Embedded Systems พอร์ต GPIO 40 ขาช่วยให้สามารถต่อเซนเซอร์และอุปกรณ์ควบคุมได้ง่าย เหมาะกับโครงงานอัจฉริยะ เช่น ระบบบ้านอัจฉริยะ (Smart Home), ระบบเกษตรอัจฉริยะ (Smart Farm), หรือระบบตรวจวัดข้อมูลสิ่งแวดล้อม
- ประหยัดพลังงานและต้นทุนต่ำ ใช้พลังงานเพียงประมาณ 3–7 วัตต์ ขึ้นอยู่กับอุปกรณ์ที่ต่อพ่วง ทำให้สามารถใช้งานต่อเนื่องได้ยาวนานโดยใช้พลังงานน้อย

2.7 LCD1602

2.7.1 คุณสมบัติหลัก (Main Features)

- เป็นจอแสดงผลแบบ LCD (Liquid Crystal Display)
- แสดงผลได้ 2 บรรทัด (2 Line) แต่ละบรรทัดมี 16 ตัวอักษร (16 Characters)
- รองรับการแสดงผลแบบ อักษร (Character LCD)
- ใช้ แรงดันไฟ 5V DC

- ใช้ ตัวควบคุมหลัก (Controller Chip) คือ HD44780 ซึ่งเป็นมาตรฐานที่ใช้งานได้กับไมโครคอนโทรลเลอร์หลายรุ่น
- สามารถเชื่อมต่อแบบ 4-bit หรือ 8-bit Data Mode ได้
- มี Backlight สำหรับให้มองเห็นได้ในที่มืด (สีเขียว/น้ำเงินตามรุ่น)
- ประหยัดพลังงานและมีอายุการใช้งานยาวนาน

2.7.2 องค์ประกอบของ LCD1602 (Components)

- จอแสดงผล (Display Area) – ส่วนที่ใช้แสดงตัวอักษร 16x2
- พินเชื่อมต่อ (Pins) – มีทั้งหมด 16 พิน (บางรุ่นมี 18 พิน ถ้ามี I2C Module)
    - VSS: กราวด์
    - VDD: แหล่งจ่ายไฟ +5V
    - VO: ปรับความเข้มของจอ
    - RS (Register Select): เลือกโหมดระหว่างคำสั่ง/ข้อมูล
    - RW (Read/Write): เลือกโหมดอ่านหรือเขียนข้อมูล
    - E (Enable): ใช้กระตุ้นการรับข้อมูล
    - D0–D7: ขาสำหรับส่งข้อมูล (4-bit หรือ 8-bit)
    - A และ K: ขาไฟเลี้ยง Backlight (Anode และ Cathode)

- Backlight – ไฟพื้นหลัง LED สำหรับให้จอสว่าง
- ตัวควบคุม (Controller Chip) – เช่น HD44780 ใช้จัดการการแสดงผลอักษร

2.7.3 การใช้งาน (Usage/Application)

- ใช้แสดงผลข้อมูลจากไมโครคอนโทรลเลอร์ เช่น Arduino, Raspberry Pi
- แสดงข้อความหรือค่าต่าง ๆ เช่น
- อุณหภูมิ, ความชื้น, ระยะทาง, เวลา, สถานการณ์ทำงานของระบบ
- นิยมใช้ในโครงงาน IoT, ระบบควบคุมอัตโนมัติ, และ หุ่นยนต์
- เชื่อมต่อกับ I2C Module เพื่อประหยัดพินการเชื่อมต่อ (เหลือเพียง SDA และ SCL)

-

ภาพที่ 2.7 LCD1602

2.8 OpenCV

OpenCV (Open-Source Computer Vision Library) คือไลบรารีโอเพนซอร์สที่ทรงพลังและได้รับความนิยมอย่างกว้างขวางสำหรับการมองเห็นด้วยคอมพิวเตอร์ (**Computer Vision**) และงานประมวลผลภาพ

2.8.1 คุณสมบัติ

- โอเพนซอร์ส (Open Source) และฟรี: ผู้ใช้สามารถนำไปใช้ แก้ไข และแจกจ่ายได้โดยไม่เสียค่าใช้จ่าย
- รองรับหลายแพลตฟอร์ม: ทำงานได้บนระบบปฏิบัติการหลัก ๆ เช่น Windows, Linux, macOS, Android และ iOS
- ประสิทธิภาพสูง: ถูกเขียนด้วยภาษา C++ เป็นหลัก ทำให้ประมวลผลได้รวดเร็ว เหมาะสำหรับงาน เรียลไทม์ (Real-Time) และมีการรองรับการประมวลผลแบบขนาน (Parallel Processing)

บทที่ 3

วิธีการดำเนินงาน

3.1 การพัฒนาระบบระบบแปลภาษามืออัจฉริยะ

จากการดำเนินงานการพัฒนา HandSense AI: ระบบแปลภาษามืออัจฉริยะ สามารถแบ่งการทำงานออกได้เป็น 2 ส่วน คือ Hardware และ Software

3.1.1. การพัฒนาระบบ HandSense AI: ระบบแปลภาษามืออัจฉริยะ ส่วน Hardware

ในส่วนของ Hardware ระบบจะทำหน้าที่แสดงผลการแปลภาษามือบนจอ LCD1602 โดยเชื่อมต่อกับบอร์ด Raspberry Pi ผ่านสาย Jumper เพื่อรับส่งข้อมูลจากฝั่งซอฟต์แวร์มายังจอแสดงผล ซึ่งช่วยให้ผู้ใช้สามารถมองเห็นผลลัพธ์การแปลภาษามือได้แบบ Real-Time

3.1.2. การพัฒนาระบบ HandSense AI: ระบบแปลภาษามืออัจฉริยะ ส่วน Software

3.1.2.1 ติดตั้งไลบรารีที่จำเป็น

- ติดตั้งไลบรารี mediapipe และ opencv-python สำหรับประมวลผลวิดีโอและแยกจุด Landmark ของมือ
- ติดตั้ง matplotlib สำหรับการพล็อต

3.1.2.2 เชื่อมต่อ Google Drive

- เชื่อมต่อ Colab กับ Google Drive เพื่อเข้าถึงชุดข้อมูลวิดีโอที่จัดเก็บไว้

3.1.2.3 ประมวลผลข้อมูลวิดีโอและแยกจุด Landmark

- กำหนดพาธไปยังโฟลเดอร์ชุดข้อมูลวิดีโอ
- ใช้ mediapipe เพื่อตรวจจับมือและแยกจุด Landmark (x, y, z) ในแต่ละเฟรมของวิดีโอ
- จัดเก็บลำดับของจุด Landmark สำหรับแต่ละวิดีโอ
- จัดการความยาวของลำดับ Landmark ให้คงที่ (Padding หรือ Truncating)
- บันทึกข้อมูลจุด Landmark ที่ประมวลผลแล้วในรูปแบบ .npz

3.1.2.4 เตรียมข้อมูลสำหรับฝึกฝนโมเดล

- โหลดข้อมูลจุด Landmark ที่บันทึกไว้
- แปลงข้อมูลให้เหมาะสมกับการนำเข้าโมเดล (รวมข้อมูล Landmark และสร้าง Label)

- แปลง Label ให้เป็นรูปแบบ One-Hot Encoding
- แบ่งข้อมูลออกเป็นชุดฝึก (Training Set) และชุดทดสอบ (Test Set)

3.1.2.5 การเพิ่มข้อมูล (Data Augmentation)

- สร้างฟังก์ชันสำหรับการเพิ่มข้อมูล Landmark (เช่น การหมุน, การปรับขนาด, การเพิ่ม Noise) เพื่อช่วยให้โมเดลมีความทนทานต่อความหลากหลายของข้อมูล
- สร้าง TensorFlow Dataset Pipeline สำหรับการนำข้อมูลเข้าโมเดล พร้อมทั้งนำ Data Augmentation มาใช้กับชุดข้อมูลฝึก

3.1.2.6 สร้างและฝึกฝนโมเดล Bidirectional LSTM

- สร้างโมเดล Neural Network โดยใช้ Keras Sequential Model
- ใช้ Bidirectional LSTM Layers เพื่อเรียนรู้รูปแบบจากลำดับเวลาของข้อมูล Landmark
- เพิ่ม Dropout Layers เพื่อป้องกัน Overfitting
- ใช้ Dense Layers สำหรับการจำแนกคลาสภาษามือ
- กำหนด Optimizer (Adam) และ Loss Function (Categorical Crossentropy)
- กำหนด Early Stopping เพื่อหยุดการฝึกเมื่อโมเดลหยุดพัฒนาบน Validation Set
- ทำการฝึกโมเดลด้วยชุดข้อมูลฝึกที่ผ่านการ Augmentation และประเมินผลบน Validation Set ในแต่ละ Epoch
- บันทึกโมเดลที่ฝึกฝนเสร็จแล้ว

3.1.2.7 ประเมินผลโมเดล (จากไฟล์ที่บันทึกไว้)

- โหลดโมเดลที่บันทึกไว้จาก Google Drive
- ใช้ชุดข้อมูลทดสอบที่เตรียมไว้เพื่อประเมินประสิทธิภาพสุดท้ายของโมเดล (ดูค่า Loss และ Accuracy)

3.1.2.8 การทำนาย (Prediction)

- โหลดโมเดลที่บันทึกไว้ (หากยังไม่ได้โหลด)
- ใช้ฟังก์ชัน extract_landmarks เพื่อประมวลผลวิดีโอใหม่ที่ต้องการทำนาย
- นำข้อมูล Landmark ที่ได้เข้าโมเดลเพื่อทำนายคลาสภาษามือ
- แสดงผลการทำนาย

3.2 วิธีการดำเนินงาน

การดำเนินงานพัฒนาโครงงาน HandSense AI: ระบบแปลภาษามืออัจฉริยะ ประกอบด้วยขั้นตอนหลัก ตั้งแต่การออกแบบระบบ การเตรียมข้อมูล การฝึกสอนโมเดล (Model Training) ไปจนถึงการนำไปใช้งานจริง โดยสามารถแบ่งขั้นตอนการดำเนินงานได้ดังนี้

3.2.1 การออกแบบและการพัฒนาระบบ HandSense AI

- ออกแบบ Flowchart แสดงลำดับการทำงานของระบบ ตั้งแต่ตรวจจับ ทำนายผล และแสดงผล
- ออกแบบโครงสร้าง Hardware และ Software ให้ทำงานร่วมกันได้อย่างมีประสิทธิภาพ
- ออกแบบการเชื่อมต่อระหว่าง Raspberry Pi – Node-RED – LCD เพื่อแสดงผลแบบเรียลไทม์
- ออกแบบขั้นตอนการประมวลผลภาพด้วย Google Colab และ VS Code เพื่อเพิ่มความแม่นยำของระบบ

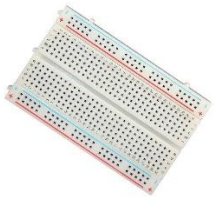
3.2.2 การพัฒนาโปรแกรมและการเทรนโมเดล

- ใช้ MediaPipe ในการเก็บข้อมูลและดึงจุดลักษณะของมือ (Hand Landmarks)
- ทำ Data Augmentation เพื่อเพิ่มความหลากหลายของข้อมูล
- ฝึกสอนโมเดล (Model Training) บน Google Colab ด้วย TensorFlow/Keras
- เขียนโปรแกรมใน VS Code เพื่อตรวจจับภาพจากกล้องและส่งผลลัพธ์ผ่าน MQTT/EMQX ไปยัง Node-RED
- พัฒนา Node-RED Flow เพื่อประสานการทำงานระหว่าง AI (LSTM Model), Raspberry Pi และ LCD โดย Node-RED จะรับข้อมูลการแปลภาษามือที่ผ่าน MQTT/EMQX และส่งต่อไปยัง Raspberry Pi เพื่อควบคุมการแสดงผลบน LCD แบบ Real-Time

3.2.3 การทดสอบระบบ

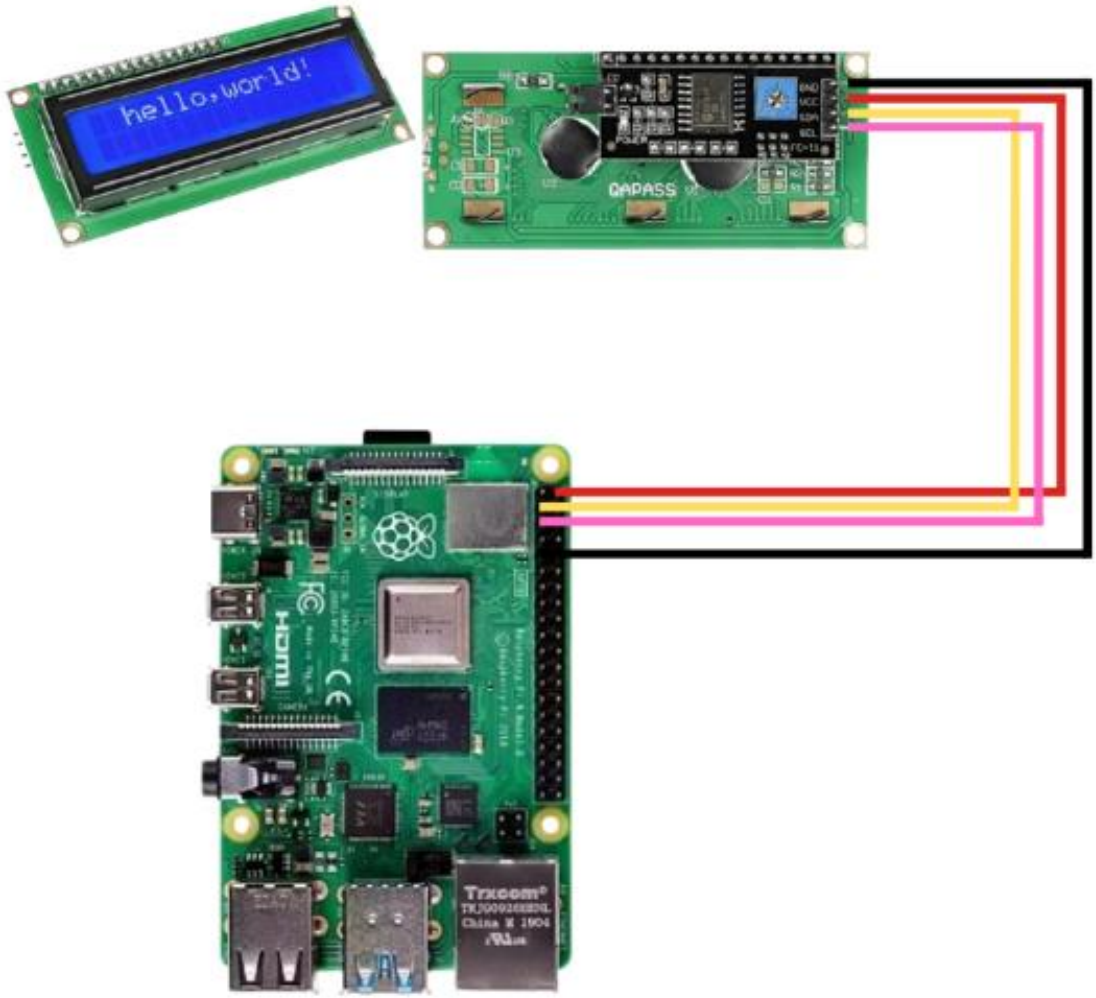
- ทดสอบความแม่นยำของกล้องและโมเดลในการตรวจจับภาษามือ
- ทดสอบการสื่อสารข้อมูลระหว่าง Node-RED, Raspberry Pi, และ LCD
- ทดสอบการแสดงผลลัพธ์ภาษามือแบบ Real-Time บนจอ LCD

3.3 อุปกรณ์

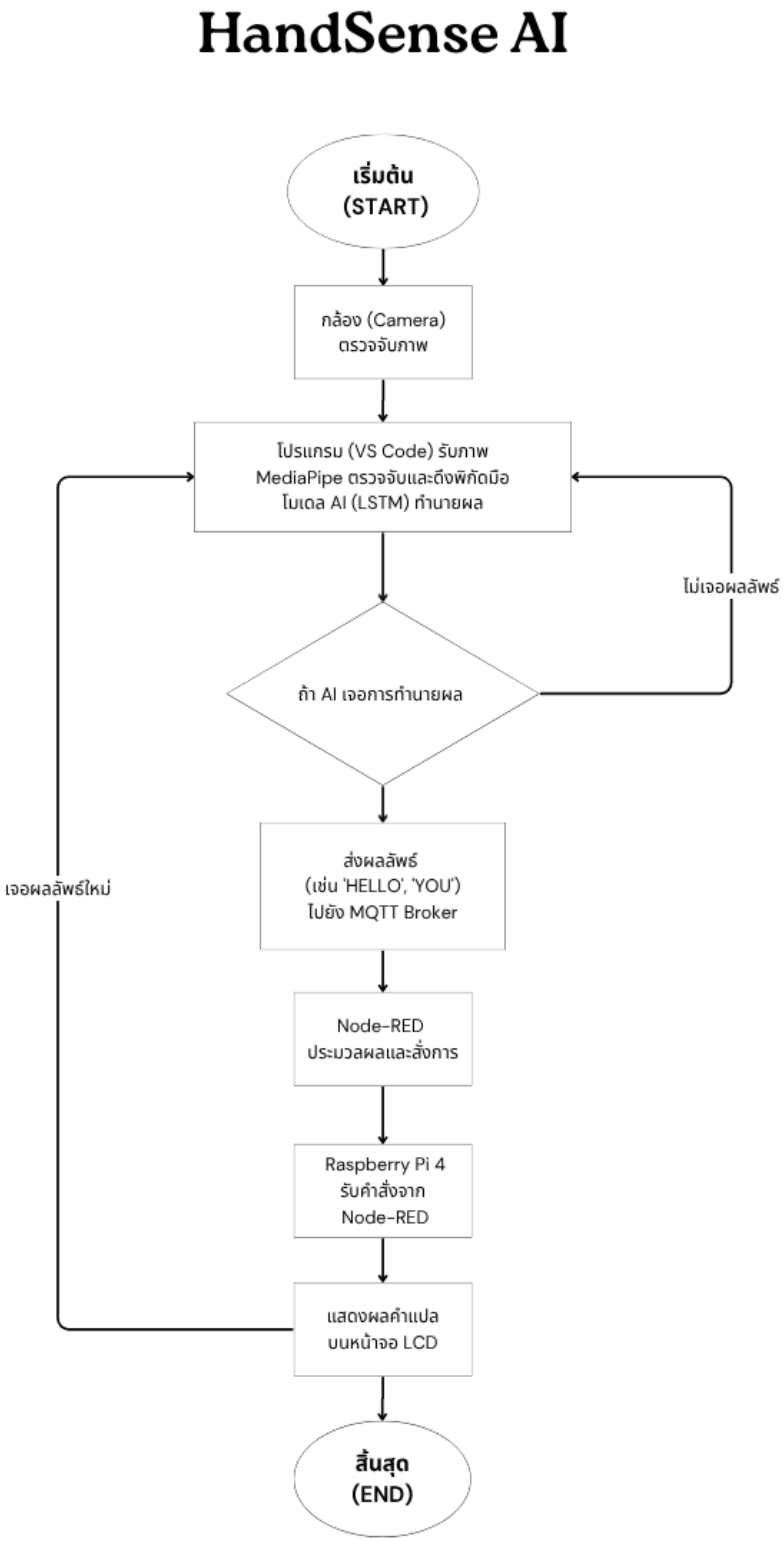
รูปอุปกรณ์	ชื่อ	ราคา
	Raspberry Pi 4	ฟรี
	LCD1602	ฟรี
	Jumper	ชุดละ 28 บาท ทั้งหมด 3 ชุด รวม 84 บาท
	ไม้พลาสวูด	120 บาท
	บานพับ	5 บาท
	Breadboard	ฟรี

รวมงบประมาณที่ได้ใช้ในการทำระบบและโมเดล รวมทั้งหมด 209 บาท

3.4 การต่อวจร



3.5 หลักการทำงาน



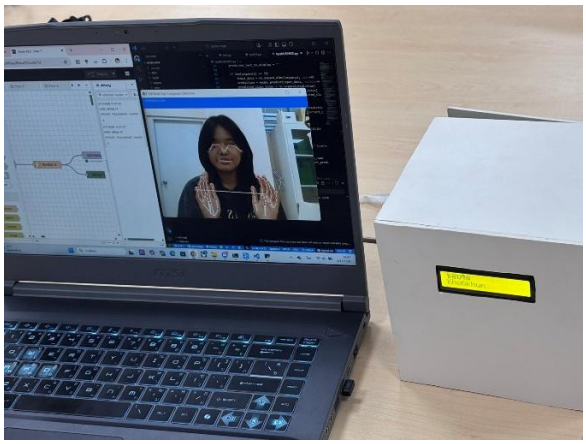
บทที่ 4

ผลการดำเนินงาน

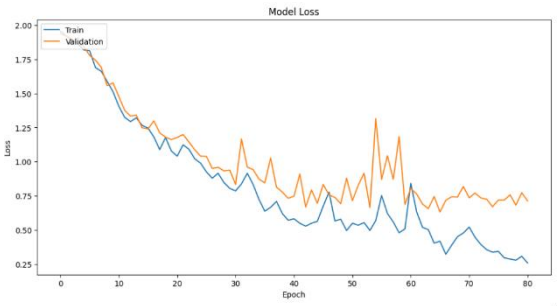
4.1 ผลการดำเนินงาน

จากการทดลองระบบ HandSense AI: ระบบแปลภาษามืออัจฉริยะ ที่พัฒนาขึ้น พบว่าระบบสามารถตรวจจับภาษามือ ได้อย่างถูกต้องและมีความแม่นยำสูง โดยเมื่อทำการทดสอบการใช้งานจริง ระบบสามารถตรวจจับท่าทางของมือผ่านกล้องและแปลผลออกมาเป็น ข้อความภาษาไทยแบบ Real-Time ได้อย่างต่อเนื่อง การแสดงผลมีความชัดเจนและตอบสนองรวดเร็ว นอกจากนี้ยังได้ทดสอบการทำงานของระบบบน Raspberry Pi โดยเชื่อมต่อกับกล้องและจอภาพ พบว่าสามารถทำงานได้เสถียร ประมวลผลและแสดงผลลัพธ์ได้แบบเรียลไทม์เช่นเดียวกัน ทำให้ระบบสามารถประยุกต์ใช้งานได้ทั้งบนคอมพิวเตอร์ทั่วไปและอุปกรณ์ขนาดเล็กอย่าง Raspberry Pi ได้อย่างมีประสิทธิภาพ

4.2 รูปโมเดล



4.3 กราฟแสดงผลการดำเนินงาน



Evaluating the Bidirectional LSTM model on the test data...
Test Loss (Bidirectional LSTM): 0.6307
Test Accuracy (Bidirectional LSTM): 0.8068

บทที่ 5

สรุปผลการดำเนินงาน

5.1 สรุปผลการดำเนินงาน

จากการพัฒนาโครงงาน “HandSense AI ระบบแปลภาษามืออัจฉริยะ” ระบบสามารถตรวจจับและจำแนกท่าทางภาษามือไทยพื้นฐาน ได้แก่ สวัสดี ,ขอบคุณ ,ขอโทษ ,ฉัน ,รัก และ เธอ ได้อย่างถูกต้อง โดยโมเดลที่พัฒนาโดยใช้เทคนิค Deep Learning มีความแม่นยำเฉลี่ยประมาณ 80% จากการทดสอบบนชุดข้อมูลภายนอก ระบบสามารถทำงานได้แบบ เรียลไทม์ ด้วยกล้องเว็บแคมทั่วไป โดยมีความหน่วงในการประมวลผลต่ำ (ประมาณ 0.5–1 วินาที) การแสดงผลลัพธ์เป็นข้อความภาษาไทยบนหน้าจอมีความชัดเจนและเข้าใจง่าย และ ระบบต้นแบบสามารถนำไปต่อยอดเพื่อใช้งานจริงในสถานที่ต่าง ๆ เช่น ศูนย์บริการประชาชน โรงพยาบาล หรือสถานศึกษา

5.2 ข้อเสนอแนะ

ข้อเสนอแนะสำหรับการพัฒนาในอนาคตคือ “HandSense AI: ระบบแปลภาษามืออัจฉริยะ” คือ ควรเพิ่มจำนวนชุดข้อมูลและคำศัพท์ภาษามือให้ครอบคลุมมากยิ่งขึ้น เช่น ตัวอักษร ก–ฮ ตัวเลข คำศัพท์พื้นฐาน และประโยคที่ใช้ในชีวิตประจำวัน เพื่อให้ระบบสามารถแปลภาษามือได้อย่างหลากหลายและสมบูรณ์ยิ่งขึ้น นอกจากนี้ควรปรับปรุงประสิทธิภาพของโมเดลให้สามารถทำงานได้ดีในสภาพแวดล้อมที่หลากหลาย เช่น พื้นที่ที่มีแสงน้อย แสงจ้า หรือมุมกล้องที่แตกต่างกัน รวมถึงเพิ่มความสามารถในการจำแนกท่าทางที่มีความคล้ายคลึงกันของผู้ใช้งานหลายคน เพื่อให้ระบบมีความยืดหยุ่นและมีความแม่นยำสูงขึ้น

อีกทั้งควรพัฒนาให้ระบบสามารถแปลภาษามือแบบต่อเนื่องได้ เพื่อให้สามารถแปลประโยคหรือข้อความยาว ๆ ได้อย่างสมบูรณ์ ไม่จำกัดเฉพาะคำเดี่ยว ซึ่งจะช่วยให้การสื่อสารมีความเป็นธรรมชาติและใกล้เคียงกับการสนทนาจริงมากขึ้น นอกจากนี้ควรเพิ่มฟังก์ชันการแปลงข้อความเป็นเสียง (Text-to-Speech: TTS) เพื่อให้ระบบสามารถสื่อสารกลับไปยังบุคคลทั่วไปได้อย่างมีประสิทธิภาพ โดยเฉพาะในสถานการณ์ที่ต้องการการสื่อสารแบบสองทาง

บรรณานุกรม

- https://www.th-sl.com/?openExternalBrowser=1
- https://dic.ttrs.or.th/home
- https://hilight.kapook.com/view/85839
- https://www.youtube.com/watch?v=ATcM_kNgbcM
- https://www.tiktok.com/@thaipbs/video/7173619253755514114
- https://www.tiktok.com/@okata_o/video/7382913357973638407
- https://www.youtube.com/watch?v=yAlkoCw_6-8
- https://www.tiktok.com/@minilego_non.k/video/7379516089455168775
- https://www.robotsiam.com/product/127/%E0%B9%82%E0%B8%A1%E0%B8%94%E0%B8%B9%E0%B8%A5-i2c-lcd-%E0%B8%9E%E0%B8%A3%E0%B9%89%E0%B8%AD%E0%B8%A1%E0%B8%AB%E0%B8%99%E0%B9%89%E0%B8%B2%E0%B8%88%E0%B8%AD-lcd-1602
- https://natdhanai-tula.medium.com/fra500-software-review-visual-studio-code-55bd7f7c575f
- https://www.raspberrypi.com/documentation/computers/raspberry-pi.html
- https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.raspberrypi.com%2Fnews%2Fraspberry-pi-imager-imaging-utility%2F&psig=AOvVaw3Upx7YXogcTLoAx7vmldkB&ust=1762071348880000&source=images&cd=vfe&opi=89978449&ved=0CBUQjRxqFwoTCPj-2rjB0JADFQAAAAAdAAAAABA7