อุปกรณ์แปลภาษาจากเสียงพูด
E - TRANSLATOR

โดย
นายพลายุทธ อินทรแก้วศรี
นายภูริวัฒน์ สุขสวัสดิ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมโทรคมนาคม
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2559

อุปกรณ์แปลภาษาจากเสียงพูด

E - TRANSLATOR

โดย

นายพลายุทธ	อินทรแก้วศรี	56010826
นายภูริวัฒน์	สุขสวัสดิ์	56010956

อาจารย์ที่ปรึกษา

ผศ.ดร. นภัทร สระเอี่ยม

ผศ.ดร. ธเนศ พัฒนธาดาพงษ์

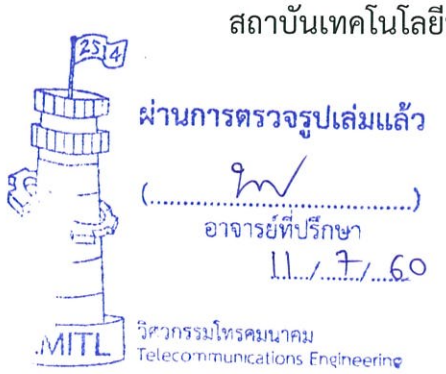
ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

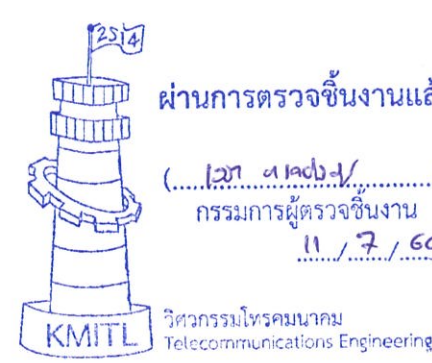
ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2559





ปริญญานิพนธ์ปีการศึกษา 2559

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง อุปกรณ์แปลภาษาจากเสียงพูด

E - TRANSLATOR

ผู้จัดทำ

1. นายพลายุทธ อินทรแก้วศรี 56010826
2. นายภูริวัฒน์ สุขสวัสดิ์ 56010956

..... **อาจารย์ที่ปรึกษา**

(ผศ.ดร.นภัทร สระเอี่ยม)

..... **อาจารย์ที่ปรึกษาร่วม**

(ผศ.ดร.ธเนศ พัฒนธาดาพงษ์)

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สามารถสำเร็จลุล่วงไปด้วยดี ด้วยความช่วยเหลือของ ผศ.ดร. นภัทร สระเอี่ยม อาจารย์ที่ปรึกษาปริญญานิพนธ์ และผศ.ดร.ธเนศ พัฒนธาดาพงษ์ อาจารย์ที่ปรึกษาร่วม ที่ได้ให้คำแนะนำและข้อคิดเห็นต่าง ๆ อันเป็นประโยชน์อย่างยิ่งในการทำปริญญานิพนธ์เล่มนี้ อีกทั้งยังช่วยแก้ปัญหาต่าง ๆ ที่เกิดขึ้นระหว่างการดำเนินงานอีกด้วย นอกจากนี้ขอขอบคุณเพื่อน ๆ ที่เป็นกำลังใจและให้ความช่วยเหลือในการทำปริญญานิพนธ์เล่มนี้

สุดท้ายนี้ ขอขอบคุณบิดามารดา และครอบครัว ซึ่งเปิดโอกาสให้ได้รับการศึกษาเล่าเรียน ตลอดจนคอยช่วยเหลือและให้กำลังใจเสมอมา

พลายุทธ อินทรแก้วศรี
ภูริวัฒน์ สุขสวัสดิ์
ผู้จัดทำ

อุปกรณ์แปลภาษาจากเสียงพูด
E – TRANSLATOR

โดย นายพลายุทธ อินทรแก้วศรี 56010826
นายภูริวัฒน์ สุขสวัสดิ์ 56010956

อาจารย์ที่ปรึกษา ผศ.ดร. นภัทร สระเอี่ยม
อาจารย์ที่ปรึกษา (ร่วม) ผศ.ดร. ธเนศ พัฒนธาดาพงษ์

บทคัดย่อ

ปริญญานิพนธ์นี้ศึกษาเกี่ยวกับการออกแบบอุปกรณ์แปลภาษาจากเสียงพูด โดยใช้ราสเบอรี่พายในการประมวลผลและพัฒนาโปรแกรมด้วยภาษาไพทอน ซึ่งอาศัยระบบการรู้จำเสียงพูดของกูเกิ้ลมาช่วยในการวิเคราะห์และแปลงเสียงเป็นข้อความ โดยในขณะนี้ได้ตั้งค่าอุปกรณ์แปลภาษาจากเสียงพูดให้สามารถแปลภาษาได้ 5 ภาษา คือ ไทย, อังกฤษ, จีน, สเปน และเยอรมัน ซึ่งสามารถกำหนดภาษาได้ อีกทั้งสามารถเชื่อมต่อกันได้หลายๆเครื่องผ่านระบบเครือข่าย และแปลภาษาได้พร้อมกัน ซึ่งอุปกรณ์แปลภาษาจากเสียงพูดมีข้อจำกัดคือผู้พูดควรพูดด้วยน้ำเสียงที่ชัดเจนและควรใช้ในบริเวณที่มีเสียงรบกวนน้อยเพื่อลดความผิดพลาดในการวิเคราะห์เสียงของอุปกรณ์

ABSTRACT

This thesis studies about the design of E-TRANSLATOR. By using the Raspberry Pi to process and develop the program on Python. That use google speech recognition to convert speech to text. This device was set to be able to translate 5 languages such as Thai, English, Chinese, Spanish and German. Which can be assigned a default language and the language you want to translate. Also devices can connect to another devices by LAN and can be translated into different languages in the same time. Device has limitations such as the speaker should speak with a clear tone and should be used in areas where there are less noise for reduce error of the device.

สารบัญ

สารบัญ (ต่อ)

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญตาราง

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

เนื่องจากการประชุม หรือการปรึกษาหารือต่าง ๆ ระหว่างบุคคลของแต่ละประเทศมีความยากลำบากในการสื่อสาร เพราะแต่ละประเทศก็จะมีภาษาต่างกันตามประเทศของตน ซึ่งการประชุม หรือการปรึกษาหารือนี้ มีมากขึ้นทุกวันในปัจจุบัน หากต้องการความสะดวกในการสื่อสารก็ต้องใช้ล่ามในการแปลภาษา ซึ่งเป็นการเปลืองทรัพยากรบุคคล และถ้าหากว่ามีอุปกรณ์ที่สามารถแปลภาษาโดยฟังเสียงพูดจากภาษาเริ่มต้นแล้วแปลเป็นเสียงพูดในอีกภาษาหนึ่งซึ่งเป็นภาษาที่ต้องการ ก็จะทำให้เกิดความสะดวกสบายยิ่งขึ้น

คณะผู้จัดทำจึงมีความสนใจที่จะจัดทำอุปกรณ์แปลภาษาอัตโนมัติ โดยจะใช้ราสเบอรี่พายเป็นอุปกรณ์ประมวลผลหลัก ซึ่งผู้เข้าร่วมการประชุม หรือการปรึกษาหารือแต่จะท่านจะมีอุปกรณ์แปลภาษาคนละหนึ่งเครื่อง โดยแต่ละเครื่องนั้นจะถูกตั้งค่าไว้ว่าต้องการให้แปลเป็นภาษาอะไรตามผู้เข้าร่วมการประชุมจะตั้งค่า และจะทำการส่งรับข้อมูลกันผ่านทางระบบเครือข่าย

1.2 วัตถุประสงค์

1) เพื่อออกแบบและสร้างอุปกรณ์แปลภาษาจากเสียงพูด
2) เพื่อศึกษาการเขียนโปรแกรม
3) เพื่อช่วยให้ง่ายต่อการสื่อสารเมื่อใช้ภาษาที่ต่างกันภายในห้องประชุม

1.3 ขอบเขตของปริญญานิพนธ์

1) ออกแบบและสร้างอุปกรณ์แปลภาษาจากเสียงพูดโดยใช้ราสเบอรี่พายในการประมวลผล
2) เขียนโปรแกรมโดยใช้ภาษาไพทอน เพื่อใช้ในการแปลภาษาจากภาษาต้นทางไปยังภาษาปลายทางที่เลือก
3) เขียนโปรแกรมเพื่อให้อุปกรณ์แต่ละตัวสามารถส่งข้อมูลผ่านระบบเครือข่ายได้ขณะอยู่ในห้องประชุม

บทที่ 2

ทฤษฎีและหลักการที่เกี่ยวข้อง

2.1 คอมพิวเตอร์บอร์ดเดี่ยว (Single Board Computer: SBC)

คอมพิวเตอร์บอร์ดเดี่ยว (SBC) เป็นคอมพิวเตอร์เครื่องเดียวที่สร้างขึ้นจากแผงวงจรเดียวมีไมโครโปรเซสเซอร์, หน่วยความจำ, อินพุต/เอาต์พุต และคุณสมบัติอื่น ๆ ที่จำเป็นสำหรับคอมพิวเตอร์ที่ใช้งานได้ คอมพิวเตอร์บอร์ดเดี่ยวเป็นระบบสาธิตหรือพัฒนาสำหรับระบบการศึกษาหรือใช้เป็นตัวควบคุมคอมพิวเตอร์ฝังตัว คอมพิวเตอร์ในบ้านหรือคอมพิวเตอร์พกพาได้ถูกรวมฟังก์ชันทั้งหมดไว้บนแผงวงจรพิมพ์เดี่ยว ไม่เหมือนกับคอมพิวเตอร์ส่วนบุคคลแบบตั้งโต๊ะ คอมพิวเตอร์แบบบอร์ดเดียวมักไม่ใช้ช่องเสียบสำหรับอุปกรณ์ต่อพ่วงต่าง ๆ คอมพิวเตอร์บอร์ดเดี่ยวถูกสร้างขึ้นโดยใช้ไมโครโปรเซสเซอร์ที่แตกต่างกันหลายยี่ห้อ โดยในโครงงานของเราได้เลือกใช้ราสเบอรรี่พายสำหรับการประมวลผล

2.1.1 ราสเบอรี่พาย (Raspberry Pi)

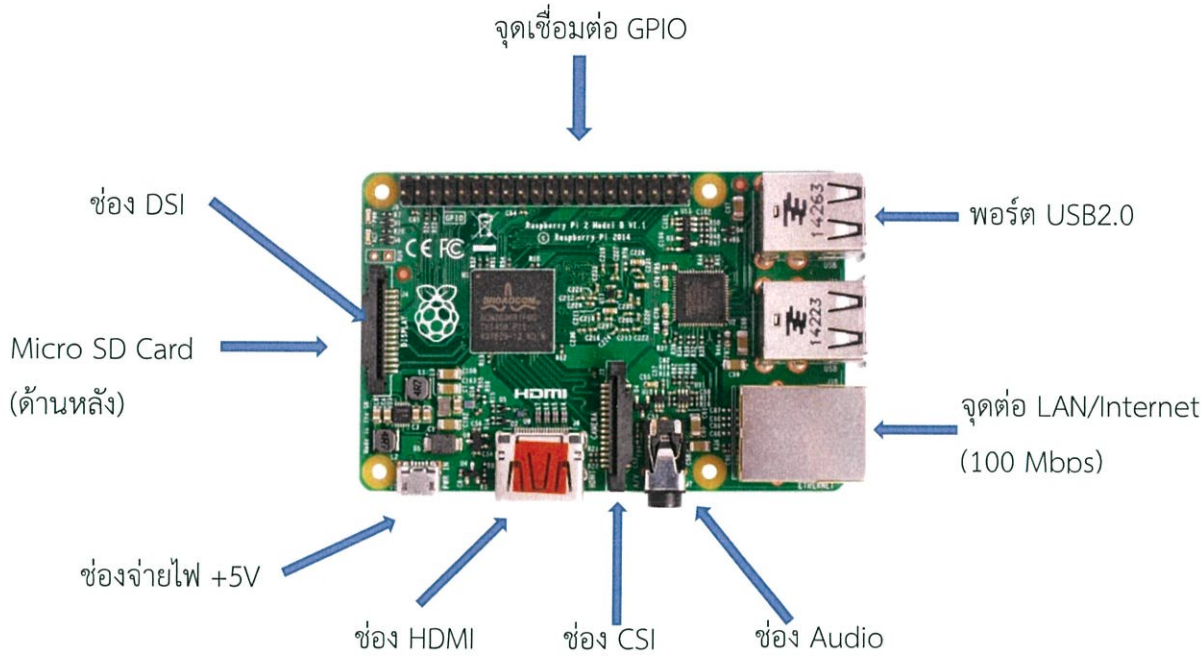
ราสเบอรี่พาย เป็นคอมพิวเตอร์บอร์ดเดี่ยว (Single Board Computer: SBC) สามารถเชื่อมต่อกับหน้าจอคอมพิวเตอร์หรือโทรทัศน์ผ่านพอร์ตเอชดีเอ็มไอ เชื่อมต่อกับเมาส์และคีย์บอร์ด เพื่อใช้งานได้เหมือนเครื่องคอมพิวเตอร์ขนาดเล็ก ใช้ทำงานเอกสาร ใช้เชื่อมต่ออินเตอร์เน็ตด้วยเว็บเบราว์เซอร์ สามารถเล่นไฟล์มัลติมีเดียต่าง ๆ ทั้งเสียงและวิดีโอ รองรับระบบปฏิบัติการลีนุกส์ต่าง ๆ เช่น ราสเบียน (พื้นฐานมาจากดีเบียน), Snappy Ubuntu Core, OpenELEC, RaspBMC, Pidora (พื้นฐานมาจาก Fedora) และRISC OS เป็นต้น

2.1.1.1 คุณสมบัติราสเบอรี่พาย ดังแสดงในตารางที่ 2.1

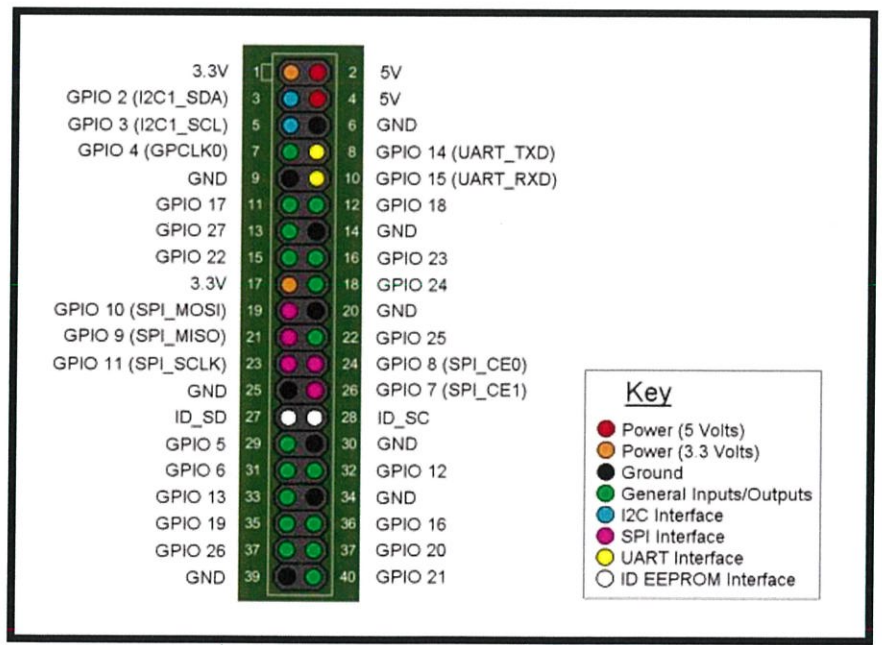
ตารางที่ 2.1 คุณสมบัติของราสเบอรี่พาย

ลำดับที่	คุณสมบัติ	รายละเอียด
1	หน่วยประมวลผลกลาง	quad-core ARM Cortex-A7 900 MHz
2	หน่วยความจำหลัก	1 GB LPDDR2 SDRAM
3	หน่วยประมวลผลภาพ	VideoCore IV 3D
4	พอร์ต USB 2.0	4 พอร์ต
5	จุดเชื่อมต่อ GPIO (General Purpose Input/Output)	40 จุด
6	พอร์ต HDMI	1 พอร์ต
7	พอร์ต Ethernet 10/100 Mbps	1 พอร์ต
8	ช่องสัญญาณเสียงขนาด 3.5 มม.	1 ช่อง
9	ช่องต่อสัญญาณกล้องแบบ CSI	1 ช่อง
10	ช่องต่อสัญญาณภาพแบบ DSI	1 ช่อง
11	ช่องใส่ Micro SD card	1 ช่อง
12	กระแสและแรงดันที่ใช้งาน	1.8A , 5V
13	ระบบปฏิบัติการ	Raspbian, Snappy Ubuntu Core, OpenELEC, RaspBMC, Pidora, RISC OS

2.1.1.2 ส่วนประกอบของราสเบอรี่พาย ดังแสดงในรูปที่ 2.1 และรูปที่ 2.2



รูปที่ 2.1 ส่วนประกอบของราสเบอรี่พาย [1]



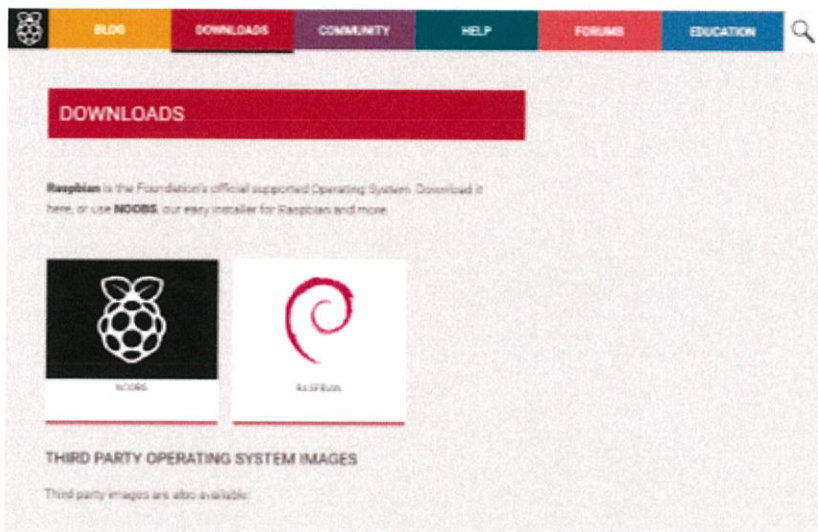
รูปที่ 2.2 ส่วนประกอบของจุดเชื่อมต่อ GPIO [2]

2.1.2 ระบบปฏิบัติการราสเบียน

การทำงานของราสเบอรี่พาย จะต้องอาศัยระบบปฏิบัติการราสเบียน ซึ่งมีพื้นฐานมาจากระบบปฏิบัติการลีนุกส์ ตระกูลดีเบียนที่นำมาดัดแปลงให้เข้ากับหน่วยประมวลผลกลาง ARM Cortex-A7

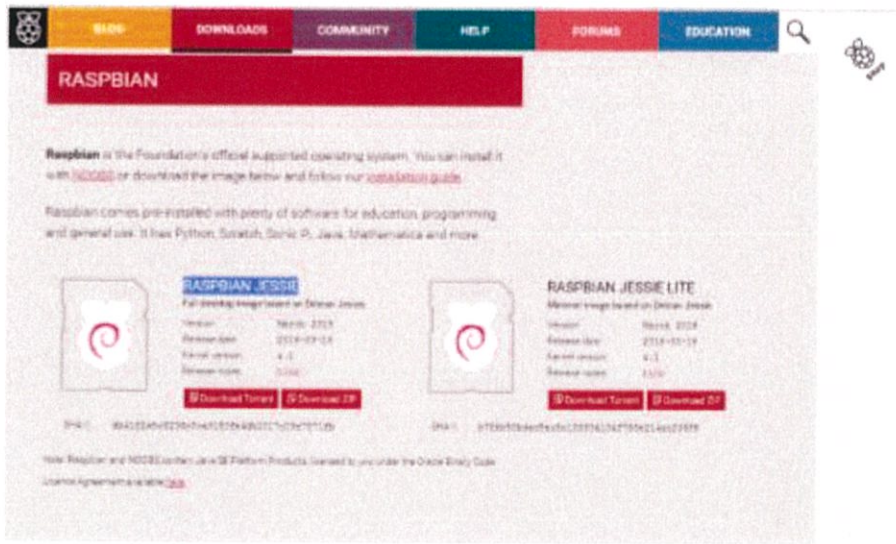
2.1.2.1 การติดตั้งระบบปฏิบัติการราสเบียนมีขั้นตอนดังแสดงในรูปที่ 2.3 – รูปที่ 2.6

1) ดาวน์โหลดระบบปฏิบัติการจากเว็บไซต์ https://www.raspberrypi.org/downloads/ ดังแสดงในรูปที่ 2.3



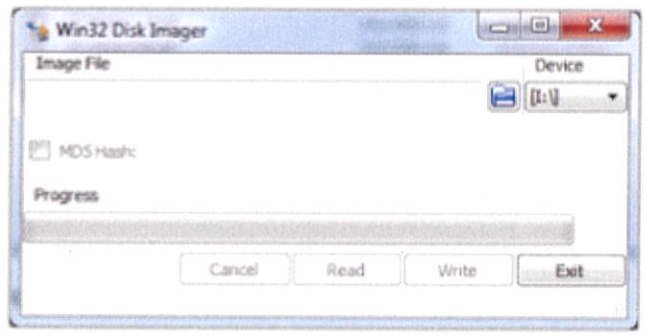
รูปที่ 2.3 เว็บไซต์ https://www.raspberrypi.org/downloads/ [3]

2) เลือกไอคอน RASPBIAN > Download ZIP ที่ RASPBIAN JESSIE ดังแสดงในรูปที่ 2.4



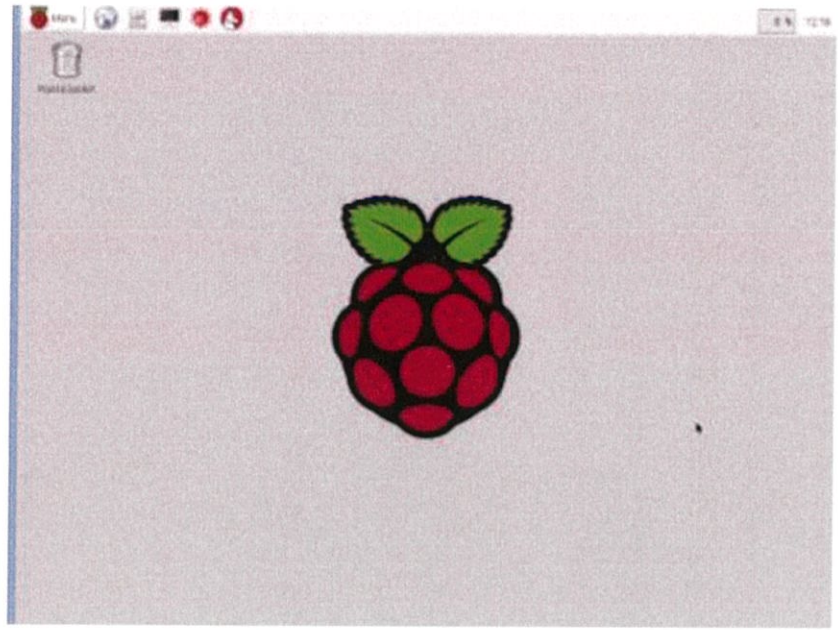
รูปที่ 2.4 การดาวน์โหลดระบบปฏิบัติการราสเบียนเจซซี่ [4]

3) ดาวน์โหลดโปรแกรม Win32DiskImage เพื่อใช้ในการเขียน Image ของระบบปฏิบัติการราสเบียน โดยมีลิงค์อยู่ในเว็บไซต์ของทางราสเบอรี่พายอยู่แล้ว โดยใส่ไมโครเอสดีการ์ด > เลือกไฟล์ 2016-03-18-raspbian-jessie.img > กดปุ่ม Write เพื่อเขียนระบบปฏิบัติ การลงในไมโครเอสดีการ์ด ดังแสดงในรูปที่ 2.5



รูปที่ 2.5 โปรแกรม Win32DiskImage [5]

4) นำไมโครเอสดีการ์ดไปใส่ในบอร์ดราสเบอรี่พาย เพื่อใช้งานบนระบบปฏิบัติการราสเบียน ดังแสดงในรูปที่ 2.6



รูปที่ 2.6 ระบบปฏิบัติการราสเบียน

2.2 ภาษาไพทอน (Python)

ภาษาไพทอน เป็นภาษาคอมพิวเตอร์ประเภทโอเพนซอร์ส (Open Source Computer Language) ซึ่งเป็นกลุ่มซอฟต์แวร์ (software) ที่เปิดเผยชุดคำสั่ง (source code) ของโปรแกรมทำให้สามารถแก้ไข ดัดแปลงชุดคำสั่งได้หมด เป็นการให้สิทธิเสรีแก่ผู้ที่จะนำไปใช้เพื่อการพัฒนาซอฟต์แวร์ร่วมกันในสังคมซอฟต์แวร์ ลักษณะของภาษาไพทอนมีรากฐานคำสั่งมาจากภาษาซี ซึ่งการประมวลผลจะทำในแบบอินเทอร์พรีเตอร์ (interpreter) คือจะประมวลผลไปทีละบรรทัดและปฏิบัติตามคำสั่งที่ได้รับ และเป็นภาษาที่สามารถพัฒนาให้ใช้งานแบบโต้ตอบกับผู้ใช้ได้ ถูกสร้างขึ้นในปี 1989 โดย Guido van Rossum ซึ่งถูกพัฒนาขึ้นมาโดยไม่ยึดติดกับแพลตฟอร์ม กล่าวคือสามารถรันภาษาไพทอนได้ทั้งบนระบบ Unix, Linux, Windows หรือแม้แต่ระบบ FreeBSD ในโครงงานนี้ได้เลือกใช้ภาษาไพทอนในการเขียนชุดคำสั่งสำหรับการทำงานของระบบ

2.2.1 ไอดีอีภาษาไพทอน (Python IDE)

ภาษาไพทอนเป็นภาษาระดับสูง โปรแกรมที่เขียนขึ้นจึงต้องถูกแปลให้เป็นภาษาเครื่อง ก่อนที่จะใช้สั่งงานคอมพิวเตอร์ได้โดยตรง การแปลภาษาไพทอนนี้ต้องใช้ตัวแปลภาษาไพทอน (python interpreter) ซึ่งผู้เขียนโปรแกรมต้องดำเนินการหลายขั้นตอน กว่าที่จะได้โปรแกรมไพทอนที่ถูกต้องสมบูรณ์นำไปใช้งานได้ ดังนั้นผู้เขียนโปรแกรมจึงนิยมใช้ซอฟต์แวร์เพื่อ

การพัฒนาโปรแกรมที่เรียกว่า ไอดีอี (Integrated Development Environment: IDE) ซึ่งประกอบด้วยเครื่องมือสำหรับแก้ไขซอร์สโค้ด (source code editor) เครื่องมือสำหรับแก้ไขจุดบกพร่องของโปรแกรม (debugger) และเครื่องมือที่ช่วยรัน (run) ซอร์สโค้ด ปัจจุบันได้มีผู้สร้างไอดีอีสำหรับภาษาไพทอนจำนวนมากให้เลือกใช้ตามความถนัดของผู้เขียนโปรแกรม โดยไอดีอีภาษาไพทอนจะทำงานได้ทั้งในโหมดอิมมีเดียท (immediate mode) และโหมดสคริปต์ (script mode)

2.2.1.1 โหมดอิมมีเดียท เป็นการพิมพ์คำสั่งทีละคำสั่ง แล้วตัวแปลภาษาไพทอนจะทำงานตามคำสั่งดังกล่าวทันที

2.2.1.2 โหมดสคริปต์ เป็นการพิมพ์คำสั่งหลายคำสั่งเก็บไว้เป็นไฟล์ก่อน เมื่อผู้เขียนโปรแกรมสั่งให้ทำงาน ตัวแปลภาษาไพทอนจะทำงานตามคำสั่งในโปรแกรมตั้งแต่คำสั่งแรกจนถึงคำสั่งสุดท้ายต่อเนื่องกันไป

2.2.2 องค์ประกอบของโปรแกรมเขียนภาษาไพทอน

2.2.2.1 ข้อมูลเข้า เป็นข้อมูลที่ผู้ใช้นำเข้าสู่โปรแกรมขณะที่โปรแกรมกำลังทำงานอยู่ เพื่อนำไปประมวลผล โดยโปรแกรมสามารถรับข้อมูลจากแป้นพิมพ์ อ่านจากไฟล์ หรือรับข้อมูลจากอุปกรณ์อื่น

2.2.2.2 ข้อมูลออก เป็นผลลัพธ์ที่ได้จากการที่คอมพิวเตอร์ทำงานตามโปรแกรม โดยข้อมูลออกจะแสดงทางจอภาพ ไฟล์ หรืออุปกรณ์แสดงผลอื่น เช่น เครื่องพิมพ์ ลำโพง

เมื่อผู้ใช้ป้อนคำสั่งบรรทัดแรกเสร็จสิ้นจะพบว่าตัวแปลภาษาไพทอนจะ ทำงานทันที โดยคำสั่ง print จะแสดงข้อความที่กำหนดไว้ภายในเครื่องหมาย ' ' ออกมา และเมื่อป้อนคำสั่งที่สองเสร็จสิ้น คำสั่ง print จะแสดงผลลัพธ์ของนิพจน์ที่อยู่ภายในเครื่องหมายวงเล็บออกทางจอภาพ ถ้าหากว่าในวงเล็บเป็นนิพจน์คณิตศาสตร์ก็จะคำนวณก่อนแล้วแสดงผลลัพธ์ออกมา ในที่นี้ภายในวงเล็บมีนิพจน์ 2+2 คำสั่ง print จึงแสดงผลลัพธ์เป็น 4 สังเกตว่าในโหมดอิมมีเดียทนี้ ตัวแปลภาษาไพทอนจะทำงานทันทีที่ผู้ใช้ป้อนแต่ละคำสั่ง ดังแสดงในตัวอย่างที่ 2.1

ตัวอย่างที่ 2.1 การใช้ไอดีอีภาษาไพทอนในโหมดอิมมีเดียทเพื่อแสดงผลลัพธ์ของ 2+2 ให้ผู้ใช้ป้อนคำสั่งต่อไปนี้ทีละบรรทัด

```
>>> print('This is my first python program to calculate an arithmetic expression.') ↵
This is my first python program to calculate an arithmetic expression.
>>> print(2+2) ↵
4
```

หมายเหตุ เครื่องหมาย ↵ แทนการกดแป้น Enter

จะเห็นว่าในโหมดสคริปต์ผู้เขียนโปรแกรมต้องเขียนโปรแกรมไพทอนให้เสร็จครบทุกบรรทัดก่อน แล้วจึงสั่งให้ไอดีอีภาษาไพทอนทำงานกับทุกคำสั่งตั้งแต่ต้นจนจบ โปรแกรมในตัวอย่างทั้งสองข้างต้นเป็นโปรแกรมขนาดเล็ก มีแต่การส่งข้อมูลออกไม่มีการรับข้อมูล เข้า สำหรับการรันคำสั่งภาษาไพทอนในโหมดอิมมีเดียทนี้ เหมาะสำหรับการทดสอบผลการทำงานของคำสั่งทีละคำสั่ง แต่ในโหมดสคริปต์ผู้เขียนโปรแกรมจะบันทึกโปรแกรมที่เขียนขึ้นทั้งหมดเก็บไว้เป็นไฟล์ได้ เพื่อให้สามารถนำมาแก้ไข ปรับปรุง และเรียกให้ทำงานใหม่ได้ในภายหลัง ดังแสดงในตัวอย่างที่ 2.2

ตัวอย่างที่ 2.2 การใช้ไอดีอีภาษาไพทอนในโหมดสคริปต์เพื่อแสดงผลลัพธ์ของ 2+2 ให้ผู้ใช้ป้อนคำสั่งทั้งสองบรรทัดต่อไปนี้ในหน้าต่างโหมดสคริปต์ แล้วสั่งรันโปรแกรม

```
print('This is my first python program to calculate an arithmetic expression.')
print(2+2)
```

ผลลัพธ์ที่ได้คือ

```
This is my first python program to calculate an arithmetic expression.
4
```

2.2.2.3 การรันโปรแกรม คือการสั่งให้คอมพิวเตอร์ทำงานตามคำสั่งในโปรแกรมตั้งแต่ต้นจนจบ โดยแต่ละไอดีอีจะมีวิธีการสั่งรันโปรแกรมที่แตกต่างกันเล็กน้อย

2.2.2.4 ข้อผิดพลาด (error) คือความผิดพลาดที่เกิดขึ้นจากการเขียนโปรแกรม ข้อผิดพลาดแบ่งได้เป็น 3 ประเภท ได้แก่

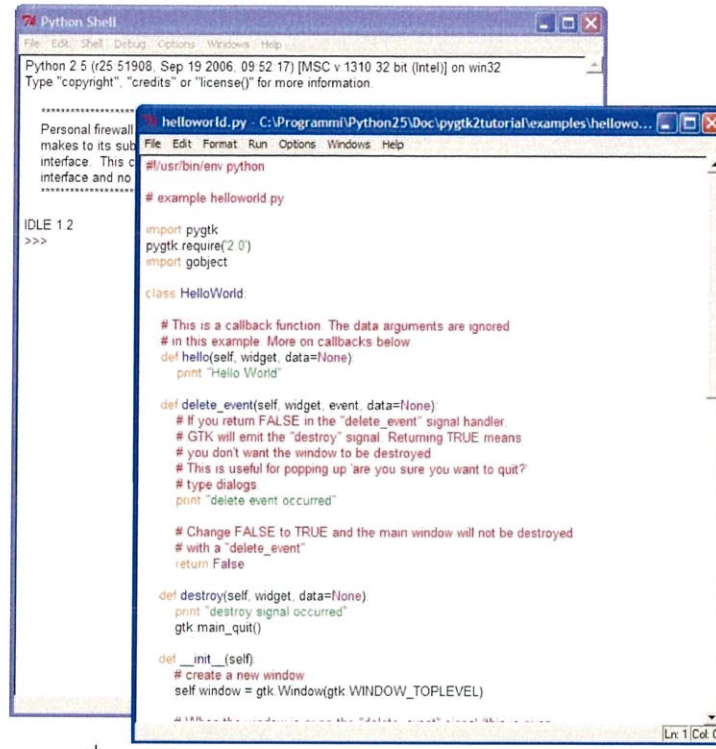
1) ข้อผิดพลาดทางไวยากรณ์ (syntax error) เป็นการเขียนโปรแกรมไม่ถูกต้องตามไวยากรณ์ของภาษา ทำให้โปรแกรมไม่สามารถทำงานได้

2) ข้อผิดพลาดขณะโปรแกรมทำงาน (runtime error) หรือเรียกว่า สิ่งผิดปกติ (exception) ซึ่งไม่ได้ผิดที่ไวยากรณ์ของโปรแกรม แต่เกิดความผิดพลาดขึ้นขณะที่โปรแกรมทำงาน ทำให้ไม่สามารถทำงานตามคำสั่งต่อไปจนสำเร็จได้ ตัวอย่างของสิ่งผิดปกติที่เกิดขึ้นบ่อย เช่น การหารเลขจำนวนที่มีตัวหารเป็นศูนย์ หรือการดำเนินการทางคณิตศาสตร์ระหว่างตัวเลขและข้อความ

3) ข้อผิดพลาดทางความหมาย (semantic error) เป็นข้อผิดพลาดที่หาสาเหตุได้ยากที่สุด เพราะว่าโปรแกรมยังสามารถทำงานได้จนจบ แต่ผลลัพธ์ไม่ถูกต้องตามที่ต้องการ ทำให้ผู้เขียนโปรแกรมต้องตรวจสอบว่ากระบวนการทำงานในคำสั่งหรือขั้นตอนใดที่ทำให้ผลลัพธ์ไม่ได้ตามที่ต้องการ ตัวอย่างเช่น การใส่วงเล็บไม่ถูกตำแหน่งในนิพจน์คณิตศาสตร์ ทำให้ลำดับการคำนวณไม่ถูกต้อง

2.2.2.5 การแก้จุดบกพร่อง (debugging) เป็นกระบวนการในการตรวจหาข้อผิดพลาดในโปรแกรมที่เขียนขึ้น โดยเมื่อมีความผิดพลาดในโปรแกรมซึ่งทำให้ผลลัพธ์ของโปรแกรมไม่ถูกต้องตามต้องการ จุดบกพร่องในโปรแกรมลักษณะนี้เรียกว่าบัก (bug) กระบวนการแก้ไขจุดบกพร่องนี้ต้องอาศัยประสบการณ์ ความรู้ด้านไวยากรณ์ของภาษาที่ใช้ในโปรแกรม และความเข้าใจในลำดับการประมวลผล เพื่อให้ได้ผลลัพธ์ที่ถูกต้องตามต้องการ โดยดำเนินการควบคู่ไปกับการเขียนโปรแกรมจนกว่าจะได้โปรแกรมที่สมบูรณ์

2.2.2.6 คอมเมนต์ (comment) เป็นคำอธิบายที่ผู้เขียนโปรแกรมใส่ไว้ เพื่อเตือนความจำ หรืออธิบายการทำงานของโปรแกรม ซึ่งเป็นข้อความที่ไม่มีผลต่อการทำงานของโปรแกรม ในภาษาไพทอนจะใช้สัญลักษณ์ # แสดงจุดเริ่มต้นของคอมเมนต์ในแต่ละบรรทัด ซึ่งมีตัวอย่างรูปแบบของโปรแกรมเขียนภาษาไพทอน ดังแสดงในรูปที่ 2.7



รูปที่ 2.7 ตัวอย่างรูปแบบของโปรแกรมเขียนภาษาไพทอน

2.3 ความรู้เกี่ยวกับการแปล

2.3.1 นิยามของการแปล

2.3.1.1 พจนานุกรมฉบับราชบัณฑิตยสถาน พ.ศ. 2525 ได้ให้ความหมายของคำว่า “แปล” ซึ่งเป็นคำกริยาไว้ 2 ความหมาย ดังนี้

1) ถ่ายทอดความหมายจากภาษาหนึ่งมาเป็นอีกภาษาหนึ่ง

2) ทำให้เข้าใจความหมาย

2.3.1.2 พจนานุกรมนิวเว็บสเตอร์ ได้ให้ความหมายเกี่ยวกับการแปลในเรื่องของภาษาไว้ว่า “to render into another language; to interpret; to explain by using other words; to express in other terms” ซึ่งอาจสรุปได้ว่าการแปล คือ

1) การถ่ายทอดข้อความจากภาษาหนึ่งไปยังอีกภาษาหนึ่ง

2) การตีความหมายหรือการทำให้เข้าใจความหมาย

3) การอธิบายโดยใช้ถ้อยคำสำนวนอย่างอื่น

2.3.1.3 จอห์น วายคลิฟ (John Wycliffe) กล่าวว่า การแปล คือการแปลประโยคให้ได้ความชัดเจนโดยใช้ภาษาของคนสามัญ

2.3.1.4 มาร์ติน ลูเธอร์ (Martin Luther) กล่าวว่า การแปล คือการสามารถถ่ายทอดวิญญาณต้นฉบับ โดยสามารถทำให้สามัญชนเข้าใจได้

2.3.1.5 ยูจีน ไนดา (Eugene A. Nida) ผู้เชี่ยวชาญทฤษฎีการแปลชาวอเมริกัน กล่าวว่า การแปล คือ การถ่ายทอดความหมายของข้อความจากภาษาหนึ่งไปยังอีกภาษาหนึ่ง โดยรักษารูปแบบของข้อความไว้ได้ตรงตามต้นฉบับ

2.3.1.6 ดานิกา เซเลสโกวิตซ์ (Danica Seleskovitch) ผู้เชี่ยวชาญเรื่องการแปล และการสอนวิทยาการแปล กล่าวว่า สิ่งสำคัญในการแปลมี 3 อย่าง ได้แก่ ข้อความ ความหมายแฝง และการถ่ายทอดความหมายออกมาเป็นภาษาแปลตามธรรมชาติ

สรุป การแปล (Translation) คือการถ่ายทอดความหมายจากภาษาต้นฉบับไปเป็นอีกภาษาหนึ่งและให้ความหมายเท่ากัน หรือใกล้เคียงกับภาษาต้นฉบับมากที่สุด

Mildred L. Larson ได้สรุปเกี่ยวกับการแปลว่า ผู้แปลจะประสบความสำเร็จในงานแปลก็ต่อเมื่อ ผู้อ่านไม่ทราบเลยว่ากำลังอ่านงานแปล แต่คิดว่ากำลังอ่านข้อเขียนในภาษาของตนเพื่อความรู้และความบันเทิง

2.3.2 รูปแบบและประเภทของการแปล

รูปแบบของการแปลอาจแบ่งได้เป็น 2 ชนิด คือการแปลตามรูปของภาษา (Form) และการแปลตามความหมาย (Meaning) โดยในโครงงานนี้ได้เลือกใช้รูปแบบการแปลแบบการแปลตามความหมาย (Meaning)

2.3.2.1 การแปลตามความหมาย (Meaning) หรือการแปลสรุปความ (Free Translation) เป็นการแปลที่ไม่ได้มุ่งรักษาโครงสร้าง ตามความหมายหรือรูปแบบของต้นฉบับอย่างเคร่งครัด มีการโยกย้ายขยายความ หรือตัดทอน หรือเปลี่ยนแปลงรูปคำหรือไวยากรณ์ การแปลลักษณะนี้นิยมใช้กับเรื่องที่ไม่จำเป็นต้องรักษาความถูกต้องของต้นฉบับ เป็นการแปลที่ใช้ในสื่อมวลชนทุกประเภทโดยเฉพาะเพื่อความบันเทิง ผู้แปลอาจอ่านจบทีละย่อหน้า ทำความเข้าใจกับเนื้อหา วิธีคิด จุดมุ่งหมายของผู้เขียน และสิ่งที่ละไว้ในฐานที่เข้าใจ เมื่อสรุปเนื้อหาหลักของต้นฉบับแล้วจึงถ่ายทอดออกมาโดยเรียบเรียบใหม่ และการแปลลักษณะนี้เป็นการแปลที่นิยมแพร่หลาย ตัวอย่างของการแปลลักษณะนี้ เช่น การแปลนิยายเรื่องสั้น นิทาน บทวิทยุ โทรทัศน์

2.3.3 ความสำคัญของการแปล

ภาษาอังกฤษเป็นภาษาที่นำมาถ่ายทอดเป็นภาษาต่าง ๆ มากที่สุดในโลก โดยเฉพาะการถ่ายทอดเป็นภาษาไทย เพราะแหล่งความรู้สมัยใหม่มักเขียนเป็นภาษาอังกฤษซึ่งเป็นภาษากลางการสื่อสารต่าง ๆ ล้วนให้ความสำคัญกับภาษาอังกฤษ เช่น Internet ภาพยนตร์ จดหมายสมัครงาน การสัมภาษณ์งาน การเจรจาธุรกิจ และการทำสัญญาต่าง ๆ เป็นต้น ภาษาอังกฤษจึงเป็นกุญแจสำคัญที่ใช้เปิดประตูเข้าสู่โลกแห่งความรู้ เช่น การอ่านตำราภาษาอังกฤษ การสนทนาแลกเปลี่ยนความรู้กับชาวต่างประเทศ ดังนั้นการแปลจึงมีความสำคัญตามไปด้วยเพราะจำเป็นต้องถ่ายทอดข้อมูลมาเป็นอีกภาษาหนึ่ง [6]

2.3.4 หลักการแปล

ลักษณะของภาษาในงานแปลที่ดีการแปลให้ได้ทั้งความถูกต้องและสำนวนที่ไพเราะอาจยึดหลักการแปลง่าย ๆ 4 ประการ ดังนี้

2.3.4.1 มีความชัดเจน คือเป็นภาษาที่มีลักษณะกระชับ ไม่ใช้คำที่ไม่จำเป็น รูปประโยคควรเป็นประโยคสั้น ๆ หลีกเลี่ยงโครงสร้างประโยคที่ซับซ้อน มีการใช้ข้อความที่ถ่ายทอดความคิดได้แจ่มแจ้ง เช่น ประโยคเดียวแสดงความคิดเดียว ไม่กำกวมหรือชวนให้ตีความได้หลายแง่หลายมุม

2.3.4.2 ใช้ภาษาได้เหมาะสม ผู้แปลต้องเลือกใช้ลีลาการเขียนให้สอดคล้องกับลักษณะของเรื่องที่จะแปล เช่น การแปลนิยายอาจใช้สำนวนให้เกิดภาพพจน์ การแปลงานด้านกฎหมายหรือการแพทย์ต้องใช้ศัพท์เฉพาะ และลีลาการเขียนที่สั้น ๆ ไม่ใช้คำหรูหราหรือสำนวนอ้อมค้อม

2.3.4.3 ใช้ภาษาเรียบง่าย ใช้ภาษาที่เรียบง่ายและสัมพันธ์กับความคิดที่กระจ่างแจ้งและตรงตามต้นฉบับ

2.3.4.4 มีความสมเหตุสมผล ซึ่งในแต่ละภาษามีความสมเหตุสมผลต่างกัน ดังนั้นภาษาที่ใช้ในการแปลต้องมีความสมเหตุสมผลเท่า ๆ กับภาษาต้นฉบับด้วย

2.4 การ์ดเสียง (Sound card)

2.4.1 ความหมายของการ์ดเสียง

การ์ดเสียง (sound card) คืออุปกรณ์คอมพิวเตอร์ที่ทำหน้าที่แปลงข้อมูลดิจิทัลที่เก็บรายละเอียดเกี่ยวกับเสียงต่าง ๆ แปลงเป็นสัญญาณเสียงในรูปแบบสัญญาณทางไฟฟ้า โดยใน

โครงงานนี้ได้ใช้การ์ดเสียงแบบยูเอสบี เนื่องจากช่องเสียบหูฟัง 3.5 มิลลิเมตรของราสเบอรี่พายไม่รองรับสัญญาณเสียงขาเข้าได้

รูปที่ 2.8 การ์ดเสียงแบบยูเอสบี

เสียงเป็นส่วนสำคัญของระบบมัลติมีเดียไม่น้อยกว่าภาพ ดังนั้นการ์ดเสียงจึงเป็นอุปกรณ์จำเป็นที่สำคัญของระบบคอมพิวเตอร์มัลติมีเดีย การ์ดเสียงได้รับการพัฒนาคุณภาพอย่างรวดเร็วเพื่อให้ได้ประสิทธิภาพของเสียงและความผิดเพี้ยนน้อยที่สุด ตลอดจนระบบเสียง 3 มิติในปัจจุบัน ความชัดเจนของเสียงจะมีประสิทธิภาพดีเพียงใดนั้นขึ้นอยู่กับปัจจัยหลัก 2 ประการ คือ อัตราการสุ่มตัวอย่างและความแม่นยำของตัวอย่างที่ได้ ซึ่งความแม่นยำของตัวอย่างนั้นถูกกำหนดโดยความสามารถของ A/D Converter (Analog-to-Digital Converter) ว่ามีความละเอียดมากน้อยเพียงใด ทำอย่างไรจึงจะประมาณค่าสัญญาณดิจิทัลได้ใกล้เคียงกับสัญญาณเสียงมากที่สุด ความละเอียดของ A/D Converter นั้นถูกกำหนดโดยจำนวนบิตของสัญญาณดิจิทัลเอาต์พุต เช่น A/D Converter 8 bit จะสามารถแสดงค่าที่ต่างกันได้ 256 ระดับ และ A/D Converter 16 bit จะสามารถแสดงค่าที่ต่างกันได้ 65,536 ระดับ หากจำนวนระดับมากขึ้นจะทำให้ความละเอียดยิ่งสูงขึ้นและการผิดเพี้ยนของสัญญาณเสียงยิ่งน้อยลง นั่นคือประสิทธิภาพที่ของเสียงที่ได้รับดีขึ้นนั่นเอง แต่จำนวนบิตต่อหนึ่งตัวอย่างจะมากขึ้นด้วย

2.4.2 หลักการทำงานของการ์ดเสียงแบบยูเอสบี

2.4.2.1 หลักการทำงานของการ์ดเสียงแบบยูเอสบีขณะรับเสียงมีดังนี้

1) เมื่อเราทำการพูดหรือเปิดไมโครโฟนให้ทำงาน สัญญาณเสียงก็จะถูกส่งจากไมโครโฟนเข้าสู่การ์ดเสียงแบบยูเอสบีในรูปแบบของคลื่นเสียงที่มีรูปแบบที่ต่างกันทั้งค่าของความถี่ (Frequency) และความสูงของคลื่น (Amplitude)

2) คลื่นสัญญาณเสียงที่ได้รับเข้ามา จะถูกภาคของ A/D Converter ทำการแปลงสัญญาณให้อยู่ในรูปของสัญญาณดิจิทัลในค่าของ 0 และ 1

3) ข้อมูลดิจิทัลจะถูกส่งออกมาสู่บัสของระบบผ่านทางอินเตอร์เฟสของการ์ดเสียงแบบยูเอสบี เพื่อถูกส่งตรงไปให้กับหน่วยประมวลผลกลางในการประมวลผลหรือจัดเก็บต่อไป

2.4.2.2 หลักการทำงานของการ์ดเสียงแบบยูเอสบีขณะเล่นเสียงมีดังนี้

1) ทำการอ่านข้อมูลจากแฟ้มเสียงในฮาร์ดดิสก์แล้วส่งต่อไปยังหน่วยประมวลผลกลาง

2) หน่วยประมวลผลกลางทำการส่งผ่านข้อมูลไปยังการ์ดเสียงแบบยูเอสบี และข้อมูลถูกส่งต่อให้กับภาคของ D/A Converter (Digital-to-Analog Converter) ทันที เพื่อทำการแปลงสัญญาณข้อมูลให้อยู่ในรูปของสัญญาณอนาล็อกเพื่อที่จะส่งต่อออกไปยังลำโพงหรือหูฟังต่อไป [7]

2.5 การรู้จำเสียงพูด (Speech Recognition)

การรู้จำเสียง (Speech Recognition) คือการประมวลผลโดยแปลงสัญญาณเสียงพูดให้อยู่ในรูปแบบของคำที่มีการเรียงลำดับต่อกันด้วยอัลกอริทึมที่ใช้ในโปรแกรมคอมพิวเตอร์ เช่น การป้อนเสียงที่เป็นตัวเลขของชั้นบนลิฟต์โดยสารอัจฉริยะผ่านไมโครโฟน เมื่อระบบรู้จำเสียงได้รับเสียงก็จะทำการประมวลผลให้ลิฟต์โดยสารขึ้นลงตามเลขชั้นที่ได้รับเข้ามา โดยตัวเลขชั้นต่างๆ จะต้องมีการสอนให้ระบบรู้จำก่อน เพื่อให้คอมพิวเตอร์สามารถเรียนรู้เสียงพูดจากคน ซึ่งจะแตกต่างไปตามลักษณะของคน เช่น เพศ อายุ แล้วจดจำไว้ใช้ในอนาคต เนื่องจากข้อจำกัดดังกล่าว จึงทำให้ระบบรู้จำเสียงในยุคแรกมีความสามารถแยกแยะคำได้อย่างถูกต้องก็ต่อเมื่อผู้พูดจะต้องมีการเว้นจังหวะคำพูด หรือพูดทีละคำ (Isolated Word) ต่อมามีการพัฒนาระบบรู้จำเสียงให้สามารถรู้จำเสียงพูดแบบต่อเนื่องได้ (Continuous Speech)

ความยากง่ายในการรู้จำเสียงยังมีปัจจัยประกอบอื่น ๆ เช่น อารมณ์และน้ำเสียงของผู้พูดในขณะนั้น ซึ่งจะทำให้คำคำเดียวมีการออกเสียงที่ต่างกันไปได้อีกด้วย อีกทั้งระบบรู้จำเสียงที่กล่าวข้างต้นทั้งหมดจำเป็นต้องอาศัยฐานข้อมูลเสียงขนาดใหญ่ เพื่อจะทำให้ระบบรู้จำเสียงให้ได้มากที่สุด ดังนั้นจำเป็นที่จะต้องใช้เนื้อที่จัดเก็บ และใช้เวลามากในการสอนคอมพิวเตอร์ให้รู้จำเสียง เนื่องจากประสิทธิภาพของระบบรู้จำจะขึ้นอยู่กับจำนวนคำที่ต้องการให้ระบบรู้จำ ตัวอย่างเช่น ในงานการควบคุมการปฏิบัติการของแขนกลหุ่นยนต์คอมพิวเตอร์ด้วยคำสั่งเสียงภาษาไทย ถ้าต้องการควบคุมการปฏิบัติงานของแขนกลหุ่นยนต์ให้มีประสิทธิภาพมากขึ้น เช่น ขึ้น 5

เซนติเมตร จำเป็นต้องสอนให้หุ่นยนต์รู้จำเพิ่มในเรื่องของระยะทางนอกเหนือจากคำสั่งที่กำหนดไว้เพียง 6 คำสั่ง ได้แก่ ขึ้น ลง ซ้าย ขวา ยืด และหด

ดังนั้นเพื่อแก้ไขข้อจำกัดของระบบรู้จำที่จะต้องรู้จำเสียงจากฐานข้อมูลเสียงขนาดใหญ่ และต้องใช้เวลาในการฝึกฝนคอมพิวเตอร์ผันแปรตามขนาดของฐานข้อมูลเสียง จึงใช้การค้นหาคำหลัก (Keyword Spotting) ในเสียงพูดนั้น เพื่อฝึกฝนของระบบรู้จำเสียง โดยการสอนคอมพิวเตอร์ให้รู้จำเฉพาะคำหลักที่ต้องการเท่านั้น จึงทำให้ประหยัดเนื้อที่ในการจัดเก็บ และประหยัดเวลาในการสอน เนื่องจากระบบไม่จำเป็นต้องเรียนรู้คำทุกคำ แต่ถ้าคำหลักที่ต้องการค้นหาไม่ได้ถูกสอนให้คอมพิวเตอร์รู้จัก ก็จะทำให้ระบบค้นหาคำหลักนั้นไม่พบจำเป็นต้องสอนคอมพิวเตอร์ให้รู้จำคำหลักที่เพิ่มเข้ามาใหม่ ดังนั้นถ้าระบบค้นหาคำหลักสามารถสร้างคำหลักที่ไม่ได้ถูกสอนได้อัตโนมัติก็จะทำให้ประหยัดเวลาในการสอนคอมพิวเตอร์เมื่อมีคำหลักใหม่เข้ามา

นับเป็นเรื่องสำคัญมากในการที่จะสอนให้คอมพิวเตอร์รู้จำ เนื่องจากเสียงพูดเป็นสัญญาณที่มีความผันผวนตามเวลา ดังนั้นจึงจะต้องทำการกำหนดรูปแบบของการจำลองเสียงพูดให้มีความสามารถที่จะแสดงให้เห็นถึงคุณสมบัติเด่นของเสียง โดยโมเดลนั้นจะต้องสามารถจำลองข้อมูลที่มีความผันผวนตามเวลา และใช้งานได้อย่างเหมาะสม โมเดลที่ว่านี้ คือ โมเดลฮิดเด็นมาร์คอฟ (Hidden Markov Model : HMM)

2.5.1 โมเดลฮิดเด็นมาร์คอฟ (Hidden Markov Model : HMM)

โมเดลฮิดเด็นมาร์คอฟ สามารถนำไปใช้เพื่อจำลองคำหรือหน่วยเสียงก็ได้ ขึ้นอยู่กับการใช้งานของระบบรู้จำเสียง เช่น ถ้าระบบรู้จำเสียงมีข้อจำกัดคือ ให้รู้จำคำศัพท์ที่น้อยและรู้จำคำพูดที่ไม่ติดต่อกัน การทำโมเดลเสียงเป็นคำเสียงจะดีกว่า เพราะจะให้ความแม่นยำมากกว่า ส่วนการทำโมเดลหน่วยเสียงจะทำได้ยากกว่า เพราะจะต้องทำการสร้างแบบหน่วยเสียงขึ้นมาก่อนที่จะนำไปรู้จำหน่วยเสียงนั้น ๆ เมื่อรู้จำหน่วยเสียงได้แล้วก็จะต้องมีวิธีสร้างคำจากหน่วยเสียงที่ได้รู้จำมาว่าคำแต่ละคำประกอบไปด้วยหน่วยเสียงอะไรบ้าง ถ้ามีการรู้จำหน่วยเสียงหนึ่งผิดไปก็จะทำให้คำคำนั้นรู้จำผิดได้ แต่สำหรับการทำโมเดลคำเสียง ความแม่นยำในการรู้จำจะขึ้นตรงกับโมเดลคำเสียงที่ได้ฝึกฝนมา จึงไม่ต้องทำการสร้างคำจากหน่วยเสียง แต่ถ้าหากระบบที่มีคำศัพท์มากกว่า 1000 คำ และต้องการรู้จำเสียงพูดแบบต่อเนื่องแล้ว ระบบรู้จำที่ใช้โมเดลหน่วยเสียงจะมีประสิทธิภาพมากกว่า เพราะจำนวนโมเดลหน่วยเสียงที่ต้องการจะมีน้อยกว่าโมเดลที่สร้างคำจากเสียง และมีความเป็นไปได้สูงที่คำหนึ่งคำจะเปลี่ยนไปเมื่อใช้กับคำอื่นๆ

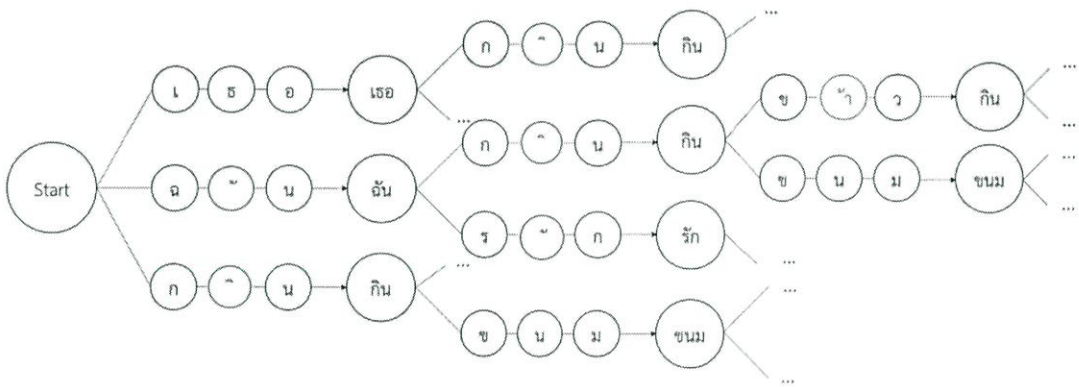
ทฤษฎีโมเดลฮิดเด็นมาร์คอฟ คือการเก็บรวบรวมสถานะหลาย ๆ สถานะที่ถูกเชื่อมโยงโดยสถานะในที่นี้คือ ลักษณะของสัญญาณเสียง หรือเครื่องหมายที่หมายถึงสัญญาณเสียงนั้น ๆ

ณ เวลาหนึ่ง ๆ ส่วนการเปลี่ยนสถานะคือ การเลื่อนหรือเปลี่ยนแปลงของสัญญาณเสียงจากเวลาหนึ่งไปหาอีกเวลาหนึ่ง สำหรับการเปลี่ยนแปลงสามารถเปลี่ยนสถานะจากเดิมไปสถานะถัดไป หรือวนอยู่สถานะเดิมเท่านั้น ไม่มีการย้อนกลับ จึงเรียกโมเดลฮิดเด็นมาร์คอฟแบบนี้ว่า Left-to-Right model

โมเดลฮิดเด็นมาร์คอฟ มีค่าความน่าจะเป็นอยู่สองชนิดคือ ค่าความน่าจะเป็นที่จะมีการเปลี่ยนสถานะจากสถานะหนึ่งไปอีกสถานะหนึ่ง และทุก ๆ สถานะสามารถให้ผลลัพธ์ โดยใช้ค่าความน่าจะเป็นที่เรียกว่า ค่าความน่าจะเป็นของผลลัพธ์ (Output Symbol) เมื่อมีการเปลี่ยนสถานะเกิดขึ้น ผลลัพธ์ที่ว่านี้จะนำไปใช้ในการตัดสินใจว่าเสียงที่ได้ยินนั้นเป็นเสียงอะไร (Output Probability)

2.5.2 โครงข่ายคำ (Word Network)

โครงข่ายคำ ทำหน้าที่รวบรวมลำดับของคำ (Word Sequence) ที่เป็นไปได้ทั้งหมด ดังรูปที่ 2.9 โดยในแต่ละเส้นทางของโครงข่ายจะประกอบด้วยค่าความน่าจะเป็นของลำดับคำ ซึ่งจะถูกคำนวณโดยโมเดลภาษา (Language Model) และความน่าจะเป็นของการเกิดสัญญาณเสียงนั้น เมื่อมีการกำหนดลำดับคำใด ๆ จะถูกคำนวณโดยโมเดลเสียง (Acoustic Model) ส่วนการค้นหาคำนั้นจะทำการค้นหาตามเส้นทางที่ให้ค่าความน่าจะเป็นรวมสูงที่สุด



รูปที่ 2.9 โครงข่ายคำ

2.5.3 ขั้นตอนของระบบรู้จำเสียง

ระบบรู้จำเสียง (Speech Recognition) โดยทั่วไป ดังรูปที่ 2.10 ประกอบไปด้วย

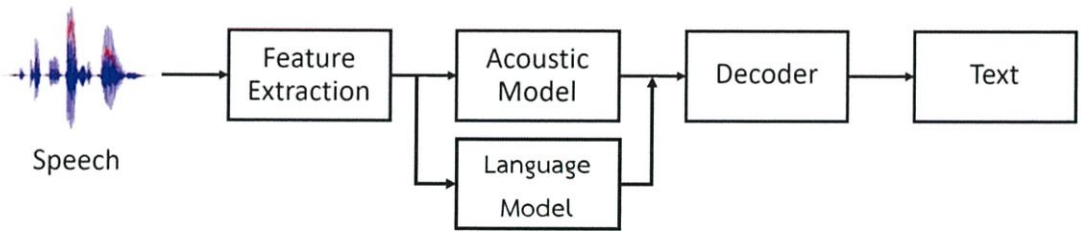
2.5.3.1 การสกัดลักษณะสำคัญ (Feature Extraction) ของสัญญาณเสียงที่เข้ามาในระบบ

2.5.3.2 การกำหนดแบบจำลองเสียง (Acoustic Modeling) ทำหน้าที่วิเคราะห์ว่าสัญญาณที่เข้ามาในระบบเป็นเสียงอะไร

2.5.3.3 การกำหนดแบบจำลองภาษา (Language Modeling) ทำหน้าที่วิเคราะห์ความน่าจะเป็นในการเกิดขึ้นของคำจากลำดับของคำที่กำหนดไว้

2.5.3.4 ส่วนถอดรหัส (Decoder) มีหน้าที่ตีความสัญญาณเสียงเป็นข้อความ

ระบบรู้จำเสียงโดยทั่วไปจะมีการใช้ทั้งโมเดลเสียง และโมเดลภาษาร่วมกัน หากใช้แต่โมเดลเสียงเพียงอย่างเดียว ก็จะทำให้ระบบเพียงรู้แต่ว่าเสียงนั้นเป็นเสียงอะไร แต่ถ้าใช้โมเดลภาษามาช่วย จะทำให้ส่วนถอดรหัสสามารถวิเคราะห์ได้ว่าเสียงนั้นเป็นคำอะไร ซึ่งเป็นการลดอัตราการผิดพลาดที่จะเกิดขึ้น ยกตัวอย่างเช่น สัญญาณเสียงของประโยค “ฉัน กิน ข้าว” ถ้าสัญญาณเสียงมีการออกเสียงคำว่า “กิน” ไม่ชัด แล้วใช้เพียงโมเดลเสียงอย่างเดียว ผลตีความจากส่วนถอดรหัสอาจเป็นคำว่า “กิ่ง” แทน แต่ถ้าใช้โมเดลภาษาเข้ามาช่วยวิเคราะห์ลำดับของคำ ว่าคำก่อนหน้านั้นคือคำว่า “ฉัน” และคำที่ตามมาหลังคือคำว่า “ข้าว” ก็จะทำให้ความน่าจะเป็นของคำนั้นคือคำว่า “กิน” มากกว่าคำว่า “กิ่ง” เป็นต้น



รูปที่ 2.10 ระบบรู้จำเสียง

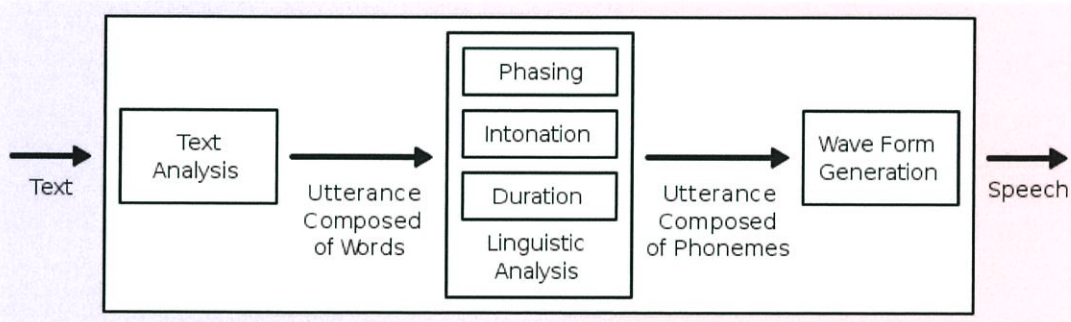
ระบบรู้จำเสียงที่โครงงานนี้เลือกใช้คือ ระบบรู้จำเสียงของกูเกิ้ล (Google Speech Recognition) เพราะว่ารองรับได้ 80 ภาษา, สามารถแปลงเสียงเป็นคำพูดได้แบบ Real-time, มีความแม่นยำแม้จะอยู่ในที่ที่มีเสียงรบกวน

2.6 การสังเคราะห์เสียง (Speech synthesis)

การสังเคราะห์เสียง คือการสร้างเสียงพูดเทียมของมนุษย์ขึ้นมา เช่น ระบบแปลงข้อความเป็นเสียงพูด (Text-To-Speech : TTS) เป็นต้น โดยการสร้างเสียงสังเคราะห์นั้นทำได้โดยนำชิ้นส่วนของคำพูดที่เก็บในฐานข้อมูลมาต่อกัน ซึ่งคุณภาพของการสังเคราะห์เสียงนั้นวัดจากความ

คล้ายคลึงกับเสียงมนุษย์และความสามารถในการสื่อสารให้เข้าใจ ระบบแปลงข้อความเป็นเสียงพูดนั้นสามารถช่วยให้ผู้พิการสามารถอ่าน ฟัง และเขียนคำบนคอมพิวเตอร์ได้ หลายระบบปฏิบัติการได้มีระบบสังเคราะห์เสียงอยู่แล้วตั้งแต่ปี 1990

ระบบแปลงข้อความเป็นเสียงพูดประกอบด้วย 2 ส่วน โดยส่วนแรกคือ front-end เป็นการพัฒนาระบบส่วนที่ติดต่อกับผู้ใช้งาน ซึ่งจะรับชุดอักขระทั้งที่เป็นข้อความภาษาไทย ภาษาอังกฤษ ตัวเลข สัญลักษณ์ต่าง ๆ เพื่อนำมาวิเคราะห์วิธีการอ่านที่ถูกต้อง พร้อมทั้งแปลงวิธีการอ่านเป็นชุดสัญลักษณ์ทางหน่วยเสียงสำหรับใช้สร้างเสียง นอกจากนี้ยังต้องวิเคราะห์ข้อมูลทางภาษาศาสตร์อื่น ๆ ที่แฝงอยู่ในข้อความ เช่น หน้าที่ของคำ เพื่อใช้ประกอบในกระบวนการสังเคราะห์เสียง และอีกส่วนคือ back-end มักเรียกว่าส่วนการสังเคราะห์ เป็นการแปลงสัญลักษณ์ทางหน่วยเสียงเป็นเสียงพูด



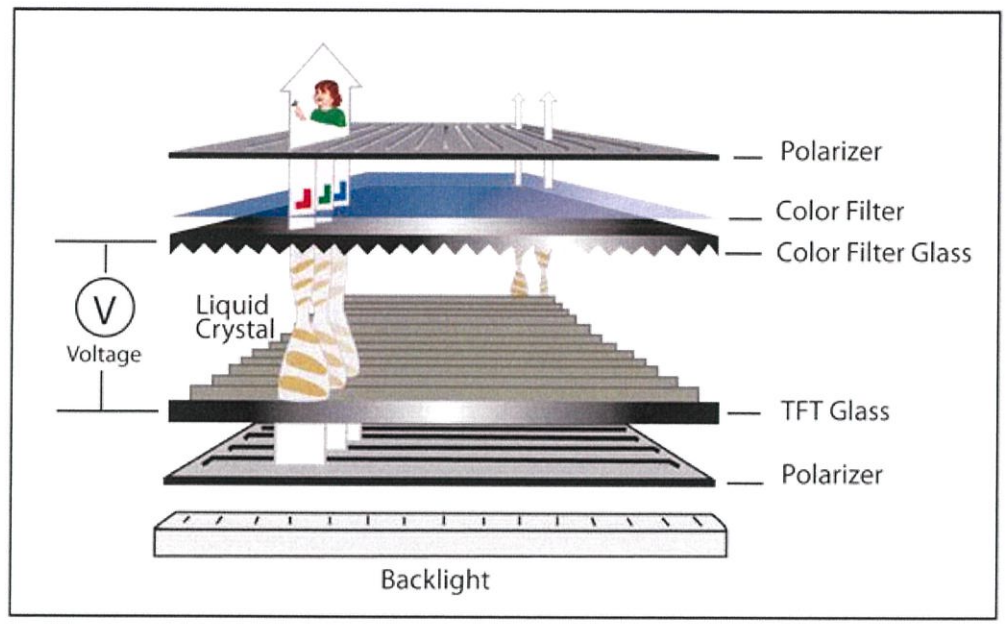
รูปที่ 2.11 ระบบสังเคราะห์เสียง [8]

2.7 จอแบบแอลซีดี (LCD)

LCD (Liquid Crystal Displays) หรือจอแบบแอลซีดี มักถูกใช้ทำเป็นจอของ HDTV หลาย ๆ รุ่น รวมถึงจอภาพของเครื่องคอมพิวเตอร์ โน้ตบุ๊ก แท็บเลตและมือถือด้วย ซึ่งเทคโนโลยีนี้ได้รับการปรับปรุงอย่างมากมายในช่วงหลาย ๆ ปีนี้ โดยในโครงงานได้ใช้จอแอลซีดีแบบทรานซิสเตอร์ที่มีฟิล์มบางเป็นสารตั้งต้น (Thin-Film Transister : TFT) เพื่อแสดงผลหน้าจอของระบบ

2.7.1 Thin-Film Transister (TFT)

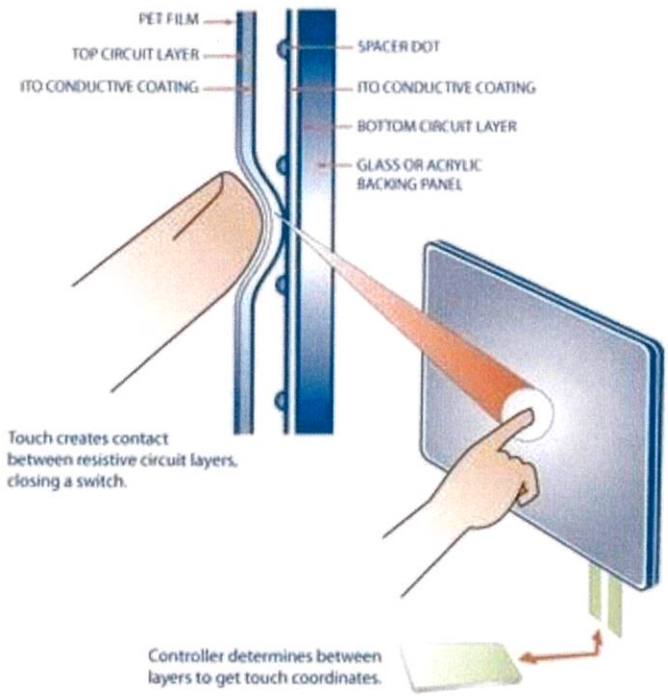
เป็นจอแสดงผลที่ใช้ทรานซิสเตอร์ที่มีฟิล์มบาง เพื่อปรับปรุงคุณภาพของการแสดงผล เช่น ความคมชัด มองเห็นได้จากหลายมุม มีสีสันในอัตราสูง มีการตอบสนองต่อการเปลี่ยนแปลงของภาพที่เร็วขึ้น โดยจอ LCD แบบ TFT เป็นจอที่มีการเรียงตัวแบบ Active Matrix ซึ่งทุกตัวจะเชื่อมต่อกันหมด และจะมีตัวส่งสัญญาณกำกับอยู่ในทุก ๆ ช่อง ทำให้มีการตอบสนองค่อนข้างไว



รูปที่ 2.12 หน้าจอ LCD แบบ TFT [9]

2.7.2 หน้าจอทัชสกรีนแบบ Resistive

เทคโนโลยี Resistive ถือว่าเป็นแบบที่ประหยัดและเหมาะกับการใช้งานประเภทต่างๆ ได้กว้างขวาง เช่น ร้านอาหาร, ร้านค้าที่ใช้เครื่อง POS, งานควบคุมทางด้านอุตสาหกรรม รวมทั้งใช้ในอุปกรณ์พกพาอย่าง PDA, Mobile เป็นต้น ทัชสกรีนแบบ Resistive จะประกอบด้วย เลเยอร์ด้านบนที่ยืดหยุ่น และเลเยอร์ด้านล่างที่อยู่บนพื้นแข็ง คั่นกลางระหว่าง 2 เลเยอร์ด้วยเม็ดฉนวนซึ่งทำหน้าที่แยกไม่ให้ด้านในของ 2 เลเยอร์สัมผัสกันเพราะด้านในของ 2 เลเยอร์นี้จะเคลือบด้วยสารตัวนำไฟฟ้าที่มีคุณสมบัติโปร่งแสง และเมื่อกดที่ทัชสกรีนจะทำให้วงจร 2 เลเยอร์ต่อถึงกัน จากนั้นวงจรควบคุมก็จะคำนวณค่ากระแสไฟฟ้า ซึ่งจะแตกต่างกันไปตามตำแหน่งที่สัมผัส เมื่อคำนวณค่ากระแสตามแนวตั้งและแนวนอนก็จะได้ตำแหน่งที่สัมผัสบนหน้าจอ



รูปที่ 2.13 หน้าจอทัชสกรีนแบบ Resistive [10]

2.8 เฮดเซ็ท (Headset)

รูปที่ 2.14 เฮดเซ็ท

เฮดเซ็ท เป็นอุปกรณ์เครื่องเสียงชนิดหนึ่ง จัดอยู่ในประเภทอุปกรณ์แสดงผลข้อมูลในรูปแบบเสียง โดยมีหน้าที่คล้ายกับลำโพง ประกอบด้วยตัวหูฟัง จะได้ยินเสียงเมื่อนำไปครอบกับหู และไมโครโฟนขนาดเล็กในตัวสำหรับใช้ติดต่อสื่อสารเพื่อการพูดได้ เช่น ทางโทรศัพท์ คอมพิวเตอร์ เป็นต้น รวมถึงใช้เป็นสิ่งบันเทิงในการฟังเพลง และเล่นวิดีโอเกม โดยปรับให้เข้ากับกระบวนการทำงานต่าง ๆ ที่ต้องใช้เสียง และสามารถพกพาไปในสถานที่ต่าง ๆ ได้เพราะมีน้ำหนักเบา

2.8.1 ประวัติ

หูฟัง มีต้นกำเนิดมาจาก นาทาเนียล บอลด์วิน (Nathaniel Baldwin) นักศึกษาจากมหาวิทยาลัยสแตนฟอร์ด เป็นผู้ประดิษฐ์ชุดหูฟังวิทยุคนแรก โดยแรกเริ่มการคิดค้นยังไม่ได้รับความสนใจจากนักธุรกิจมากนัก จนกระทั่งช่วงต้นสงครามโลกครั้งที่หนึ่ง กองทัพเรือสหรัฐอเมริกาได้สั่งซื้อชุดหูฟัง 100 ชุด ทำให้ชุดหูฟังเป็นที่รู้จักมากขึ้น หลังจากนั้นนักบินสองคนซึ่งเป็นผู้ก่อตั้งบริษัทแพลนทรอนิกส์ (Plantronics) ได้เริ่มผลิตชุดหูฟังที่มีน้ำหนักเบาเหมาะสำหรับการสวมใส่ โดยทดลองใช้ในเครื่องบินเป็นครั้งแรก เพื่อแก้ปัญหาความยากลำบากที่ได้รับจากการใช้หูฟังขนาดใหญ่ ทำให้หูฟังเป็นอุปกรณ์ที่ได้รับความนิยมมาจนถึงปัจจุบัน

2.8.2 ประเภทของหูฟัง

หูฟังแบ่งออกได้หลัก ๆ 3 ประเภท ได้แก่ หูฟังขนาดใหญ่, หูฟังขนาดกลาง และหูฟังขนาดเล็ก โดยในโครงงานได้เลือกใช้หูฟังขนาดกลางเนื่องจากมีคุณภาพเสียงที่ดีและสามารถพกพาได้สะดวก

2.8.2.1 หูฟังขนาดกลาง เป็นหูฟังที่มีขนาดใหญ่กว่าหูฟังขนาดเล็กเพียงเล็กน้อย โดยถูกออกแบบมาเพื่อความสะดวกสบายในการพกพาที่มากขึ้นเมื่อเทียบกับหูฟังขนาดใหญ่ ลักษณะของหูฟังขนาดกลาง จะมีขนาดใกล้เคียงกับใบหู เมื่อใส่แล้วจะแนบหูพอดี ไม่ได้ครอบปิดหูทั้งหมดเหมือนแบบหูฟังขนาดใหญ่ จุดเด่นของหูฟังขนาดกลางคือ จะได้คุณภาพเสียงที่ดีขึ้นจากหูฟังขนาดเล็กเพราะว่าหูฟังขนาดกลางจะมีตัวขับเสียงที่ใหญ่กว่าหูฟังขนาดเล็ก ทำให้มีช่วงของเสียงที่กว้างกว่า จุดด้อยของหูฟังขนาดกลางคือ การป้องกันเสียงรบกวนจากภายนอก จะทำได้ไม่ดีเท่าหูฟังขนาดเล็ก และเนื่องจากน้ำหนักและขนาดที่เพิ่มขึ้นเมื่อเทียบกับหูฟังขนาดเล็ก ทำให้หูฟังขนาดกลางจะไม่เหมาะสมสำหรับการทำกิจกรรมต่าง ๆ ที่ต้องมีการเคลื่อนไหวตลอดเวลา เช่น การวิ่ง เป็นต้น

2.8.3 หลักการทำงาน

ชุดหูฟังประกอบไปด้วย 2 ส่วนใหญ่ ๆ คือ ส่วนลำโพง และ ส่วนไมโครโฟน

2.8.3.1 การทำงานของลำโพง โดยหน้าที่ของลำโพงคือ เปลี่ยนสัญญาณทางไฟฟ้าที่ได้มาจากเครื่องขยายเป็นสัญญาณเสียง (Sound wave) ลำโพงที่ดีจะต้องสร้างเสียงให้เหมือนกับต้นฉบับเดิมมากที่สุด โดยมีการผิดเพี้ยนน้อยที่สุด เสียงเป็นคลื่นตามยาว เสียงแหลมและทุ้มขึ้นกับความถี่ ส่วนเสียงดังหรือค่อยขึ้นอยู่กับขนาดแอมพลิจูดของคลื่นนั้น โดยเมื่อมีการป้อนสัญญาณไฟฟ้าให้กับขดลวดเสียงของลำโพง หรือมีการนำลำโพงไปต่อกับเครื่องขยายสัญญาณเสียงจะมีสัญญาณเสียงออกมาที่ลำโพง หลักการคือเมื่อมีสัญญาณไฟฟ้าป้อนเข้ามาจะเกิดเส้นแรงแม่เหล็กเกิดขึ้นโดยรอบ อำนาจของเส้นแรงแม่เหล็กจะดูดและผลักกับเส้นของแม่เหล็กถาวรตามสัญญาณไฟฟ้าที่ได้จากความถี่เสียง ซึ่งมีความถี่เสียงตั้งแต่ 20 Hz - 20 KHz ที่มีการเปลี่ยนแปลงเฟสตลอดเวลาทำให้กรวยกระดาษที่ยึดติดกับขดลวดเสียงจะเกิดการเคลื่อนที่ดูดและผลักอากาศ จึงเกิดเป็นคลื่นเสียงขึ้น

2.8.3.2 การทำงานของไมโครโฟน ซึ่งไมโครโฟนเป็นอุปกรณ์ทางอิเล็กทรอนิกส์ ทำหน้าที่เปลี่ยนสัญญาณคลื่นเสียง เช่น เสียงพูด เสียงเพลง เสียงดนตรี ให้เป็นสัญญาณแม่เหล็กไฟฟ้า โดยไมโครโฟนส่วนใหญ่จะประกอบไปด้วยแผ่นไดอะแฟรมที่เป็นโลหะบางโดยการทำงานจะใช้หลักการของตัวเก็บประจุ คือเมื่อมีคลื่นเสียงเข้ามากระทบแผ่นไดอะแฟรม จะทำให้แผ่นไดอะแฟรมมีการเคลื่อนที่เข้าออก ทำให้ระยะห่างระหว่างแผ่นเพลทเปลี่ยนแปลงเป็นผลให้ค่าความจุของไมโครโฟนเปลี่ยนแปลงตามไปด้วย เมื่อต่อเข้ากับตัวต้านทานและแบตเตอรี่ก็จะเกิดการเปลี่ยนแปลงแรงดันไฟฟ้าที่ตกคร่อมตัวต้านทานตามไปด้วย จึงเกิดเป็นสัญญาณไฟฟ้าขึ้น

2.9 การรับส่งข้อมูลระหว่างอุปกรณ์

สถาปัตยกรรมแบบไคลเอนต์-เซิร์ฟเวอร์ ประกอบไปด้วย 2 ส่วนคือ ระบบคอมพิวเตอร์ที่มีประสิทธิภาพสูงทำหน้าที่เป็นเซิร์ฟเวอร์ ซึ่งจะให้บริการอยู่ตลอดเวลา และเครื่องไคลเอนต์ ได้แก่ Desktop PC, Notebook และ Smartphone เป็นต้น ซึ่งจะทำหน้าที่ร้องขอบริการ ไคลเอนต์อาจจะเป็นเครื่องที่ทำงานตลอดเวลาหรือไม่ก็ได้ และไม่สามารถสื่อสารกันเองได้ สำหรับเครื่องเซิร์ฟเวอร์จะต้องมีหมายเลขไอพีคงที่ (Fixed IP Address)

2.9.1 หมายเลขไอพีแอดเดรส (IP Address)

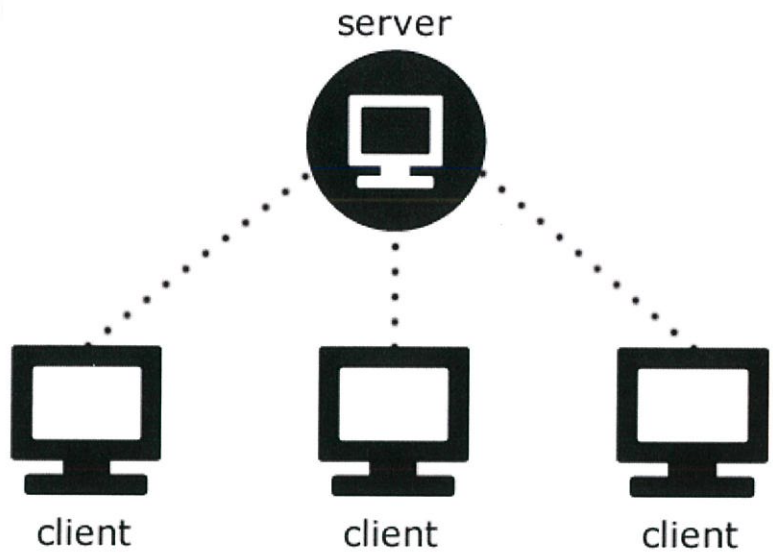
คอมพิวเตอร์ที่เชื่อมต่อกับระบบเครือข่ายจำเป็นต้องมีหมายเลขไอพีแอดเดรสเพื่อใช้สำหรับติดต่อสื่อสารกัน โดยหมายเลขไอพีดังกล่าวต้องไม่ซ้ำกันเลย (Unique) สำหรับรูปแบบของ

ไอพีแอดเดรสคือ ddd.ddd.ddd.ddd โดย d คือเลขฐานสิบและไอพีแอดเดรสแต่ละกลุ่มจะมีค่าระหว่าง 0-255 ตัวอย่างเช่น 192.168.5.10 เป็นต้น แต่การจดจำหมายเลขไอพีแอดเดรสเป็นเรื่องยากเนื่องจากมีความยาวมาก ดังนั้นจึงนิยมเรียกชื่อเครื่องแทน เช่น John-PC, Computer001 เป็นต้น

ชื่อเครื่องที่ใช้สำหรับทดสอบการทำงานของโปรแกรมเบื้องต้นเรียกว่า "localhost" มีหมายเลขไอพีแอดเดรสคือ 127.0.0.1 หรือเรียกว่า IP Loopback ก็ได้ หมายเลขไอพีดังกล่าวจะไม่สามารถเชื่อมต่อกับระบบเครือข่ายได้ แต่มีหน้าที่สำหรับทดสอบการทำงานของแอปพลิเคชันที่พัฒนาขึ้นก่อนการนำไปใช้จริงเท่านั้น

2.9.2 พอร์ต, เซิร์ฟเวอร์ และไคลเอนต์ (Ports, Servers and Client)

พอร์ต คือช่องทางการเชื่อมต่อระหว่างเครื่องเซิร์ฟเวอร์และไคลเอนต์ เปรียบเสมือนท่าเรือขนส่งสินค้า ซึ่งเรือทุก ๆ ลำจะต้องนำเรือมาจอดที่ท่าเรือ (เปรียบเสมือนพอร์ต) จากนั้นก็จะทำการขนถ่ายสินค้า (เปรียบเสมือนข้อมูล) พอร์ตก็จะถูกกำหนดเป็นเลขจำนวนเต็มที่มีค่าระหว่าง 0-65535 โดยพอร์ตตั้งแต่ 0-1023 ถูกจองไว้เพื่อใช้งานพิเศษหรือเรียกว่า Well known ports เช่น ทำหน้าที่เป็นเว็บเซิร์ฟเวอร์ (พอร์ต80), จดหมายอิเล็กทรอนิกส์ (พอร์ต25), DNS (พอร์ต53) เป็นต้น ดังนั้นหมายเลขพอร์ตตั้งแต่ 0-1023 จึงไม่ควรนำไปใช้ในการเขียนโปรแกรม สำหรับเครื่องไคลเอนต์โดยปกติเมื่อมีการสื่อสารจะต้องใช้พอร์ตเช่นเดียวกับเครื่องเซิร์ฟเวอร์ แต่ระบบปฏิบัติการจะสุ่มเลือกขึ้นมาให้เองอัตโนมัติ การเขียนโปรแกรมเชื่อมต่อระหว่างเซิร์ฟเวอร์และไคลเอนต์จะใช้โมดูลซ็อกเก็ต (Socket) ในการทำงาน ซึ่งซ็อกเก็ตจะทำหน้าที่สร้างลิงค์เพื่อเชื่อมต่อระหว่างโพรเซสที่ทำงานอยู่ในฝั่งเซิร์ฟเวอร์กับไคลเอนต์ในลักษณะโฮสต์ (Host) ต่อโฮสต์ (เซิร์ฟเวอร์ 1 เครื่องกับไคลเอนต์ 1 เครื่อง) เรียกว่าการเปิดซ็อกเก็ต หรือเปิดพอร์ตก็ได้ ผู้เขียนโปรแกรมสามารถเปิดซ็อกเก็ตได้มากกว่า 1 ซ็อกเก็ตบนหมายเลขพอร์ตเดียวกันได้ (เหมือนท่าเรือที่สามารถรองรับการจอดเรือได้หลายลำ) โดยปกติแล้วความเร็วในการสื่อสารระหว่างไคลเอนต์และเซิร์ฟเวอร์จะแตกต่างกัน เนื่องจากเซิร์ฟเวอร์จะมีประสิทธิภาพการทำงานที่สูงกว่าไคลเอนต์มาก และเซิร์ฟเวอร์จะต้องรองรับการร้องขอจากไคลเอนต์เป็นจำนวนมากในเวลาเดียวกัน ส่งผลให้การสื่อสารระหว่างเซิร์ฟเวอร์และไคลเอนต์อาจจะเกิดปัญหาในการรับส่งข้อมูลได้ ดังนั้นเพื่อแก้ปัญหาดังกล่าวจึงใช้บัฟเฟอร์ (Buffer) เข้ามาช่วยในการทำงาน ถ้าจะอธิบายให้ง่าย ๆ บัฟเฟอร์คือ การจัดเรียงข้อมูลไว้ในพื้นที่ที่ได้จัดเตรียมไว้ก่อน เมื่อเครื่องคอมพิวเตอร์พร้อมทำงานแล้ว จะนำข้อมูลในบัฟเฟอร์ไปทำงานนั่นเอง จากรูปที่ 2.15 แสดงความสัมพันธ์ระหว่างเซิร์ฟเวอร์, ไคลเอนต์



รูปที่ 2.15 ความสัมพันธ์ระหว่างเซิร์ฟเวอร์, ไคลเอนต์

2.9.3 การเขียนโปรแกรมฝั่งไคลเอนต์

ในตัวอย่างนี้จะแนะนำการเขียนโปรแกรมเพื่อรับข้อความจากเครื่องเซิร์ฟเวอร์มาแสดงผลในเครื่องไคลเอนต์ โดยอาศัยโมดูลซ็อกเก็ตช่วยในการเขียนโปรแกรม ซึ่งขั้นตอนการเขียนโปรแกรมฝั่งไคลเอนต์มีดังนี้ คือ

1) สร้างซ็อกเก็ต

2) เปิดซ็อกเก็ตด้วยพอร์ตที่สามารถใช้งานได้ (แนะนำให้ใช้พอร์ตตั้งแต่ 1,024 ขึ้นไป) ในเครื่องผู้ใช้เอง (Localhost) ในที่นี้จะใช้พอร์ตหมายเลข 5000

3) รับข้อความจากเครื่องเซิร์ฟเวอร์ผ่านซ็อกเก็ต

4) แสดงผลข้อความออกจอภาพ

สำหรับตัวอย่างโปรแกรมฝั่งไคลเอนต์แสดงดังรูปที่ 2.16

```
1   ''' Client for obtaining message from localhost.'''
2   from socket import *
3
4   HOST = 'localhost'
5   PORT = 5000
6   BUFFER_SIZE = 1024
7   ADDRESS = (HOST, PORT) #(127.0.0.1, 5000)
8
9   server = socket(AF_INET, SOCK_STREAM) #Create a socket
10  server.connect(ADDRESS) #Connect it to a host
11  data = server.recv(BUFFER_SIZE) #Read a string from it
12  message = bytes.decode(data)
13  print(message)
14  server.close() #Close the connection
```

รูปที่ 2.16 ตัวอย่างโปรแกรมฝั่งไคลเอนต์

จากตัวอย่างโปรแกรม แสดงโปรแกรมฝั่งไคลเอนต์ ทำหน้าที่รอรับข้อความจากเครื่องเซิร์ฟเวอร์มาแสดงผล บรรทัดที่ 2 นำเข้าโมดูลซ็อกเก็ตโดยเลือกคลาส, เมธอดทั้งหมด (*) ที่อยู่ในโมดูลดังกล่าวเข้ามาทำงาน บรรทัดที่ 4 กำหนดค่าให้กับตัวแปร HOST เท่ากับ 'localhost' (127.0.0.1) บรรทัดที่ 5 กำหนดค่าหมายเลขพอร์ต (PORT) เท่ากับ 5000 ซึ่งหมายเลขพอร์ตดังกล่าวอยู่บนฝั่งเซิร์ฟเวอร์ บรรทัดที่ 6 กำหนดขนาดของบัฟเฟอร์เท่ากับ 1024 โดยมีหน่วยเป็นไบต์ (Bytes) บรรทัดที่ 7 กำหนดค่าให้กับตัวแปร ADDRESS มีค่าไอพีเท่ากับ 127.0.0.1 และหมายเลขพอร์ตเท่ากับ 5000 (ในตัวอย่างนี้ เซิร์ฟเวอร์และไคลเอนต์จะทำงานอยู่ในเครื่องเดียวกัน) บรรทัดที่ 9 สร้างซ็อกเก็ตคอนเนคชั่นด้วยคลาสซ็อกเก็ตโดยคลาสดังกล่าวต้องการพารามิเตอร์ 2 ตัวคือ AF_INET ซึ่งเป็นมาตรฐานการเชื่อมต่อของโพรโทรคอล TCP/IP และ SOCK_STREAM คือการเชื่อมต่อเหมือนกับการสร้างท่อจำลอง (Pipe) ข้อมูลที่รับและส่ง โดยข้อมูลจะไหลในท่อดังกล่าวอย่างต่อเนื่อง บรรทัดที่ 10 สั่งให้เกิดการเชื่อมต่อระหว่างไคลเอนต์และเซิร์ฟเวอร์ด้วยหมายเลขไอพีแอดเดรสและพอร์ตที่กำหนดไว้ใน ADDRESS บรรทัดที่ 11 ไคลเอนต์ร้องขอข้อมูลจากเซิร์ฟเวอร์ด้วยเมธอด server.recv(BUFFER_SIZE) โดย BUFFER_SIZE คือขนาดของบัฟเฟอร์ที่ฝั่งไคลเอนต์จะรับข้อมูลได้สูงสุดในแต่ละรอบ ค่าที่ส่งกลับมาจากเซิร์ฟเวอร์คือข้อความเก็บไว้ในตัวแปรชื่อ data แต่เนื่องจากข้อมูลที่ส่งและรับของซ็อกเก็ตในไพธอน 3.0 ขึ้นไปจะใช้ข้อมูลชนิดไบต์แทนสตริง ดังนั้นจึงจำเป็นต้องทำการแปลงจากไบต์เป็นสตริงเสียก่อนโดยใช้เมธอด bytes.decode() ดังแสดงในบรรทัดที่ 12 ข้อมูลที่แปลงแล้วจะเก็บไว้ในตัวแปรชื่อ message ต่อจากนั้นบรรทัดที่ 13 พิมพ์ข้อมูลที่อยู่ในตัวแปร message บรรทัดที่ 14 สั่งให้โปรแกรมฝั่งไคลเอนต์ยุติการเชื่อมต่อ ส่งผลให้เซิร์ฟเวอร์ปิดช่องทางการเชื่อมต่อจากเครื่องไคลเอนต์ดังกล่าว

2.9.4 การเขียนโปรแกรมฝั่งเซิร์ฟเวอร์

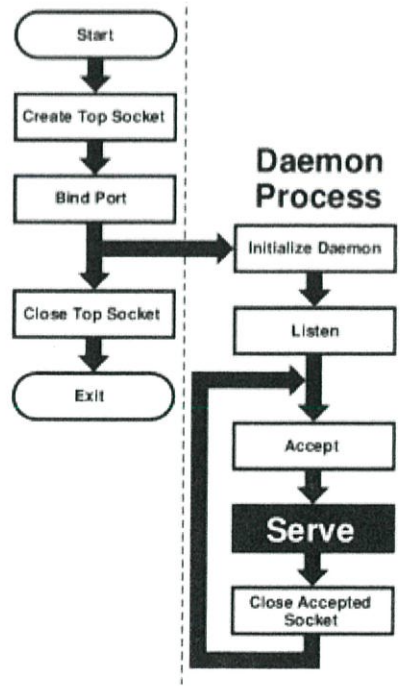
โปรแกรมในฝั่งเซิร์ฟเวอร์จะแตกต่างจากฝั่งไคลเอนต์เล็กน้อยคือ เมื่อโปรแกรมฝั่งไคลเอนต์เชื่อมต่อกับเซิร์ฟเวอร์แล้ว จากนั้นไคลเอนต์จะรับและส่งข้อมูลกับเซิร์ฟเวอร์อย่างต่อเนื่อง เมื่อรับส่งข้อมูลเสร็จเรียบร้อยแล้ว ไคลเอนต์จะร้องขอเพื่อยุติการเชื่อมต่อจากเซิร์ฟเวอร์ ซึ่งเป็นอันหมดหน้าที่ของไคลเอนต์ แต่สำหรับเซิร์ฟเวอร์แล้วจะไม่ยุติการทำงาน โดยเซิร์ฟเวอร์จะทำงานต่อไปเรื่อย ๆ เพื่อรอการเชื่อมต่อจากไคลเอนต์อื่น ๆ ต่อไป สำหรับขั้นตอนการเขียนโปรแกรมในฝั่งเซิร์ฟเวอร์มีดังนี้

1) สร้างซ็อกเก็ต

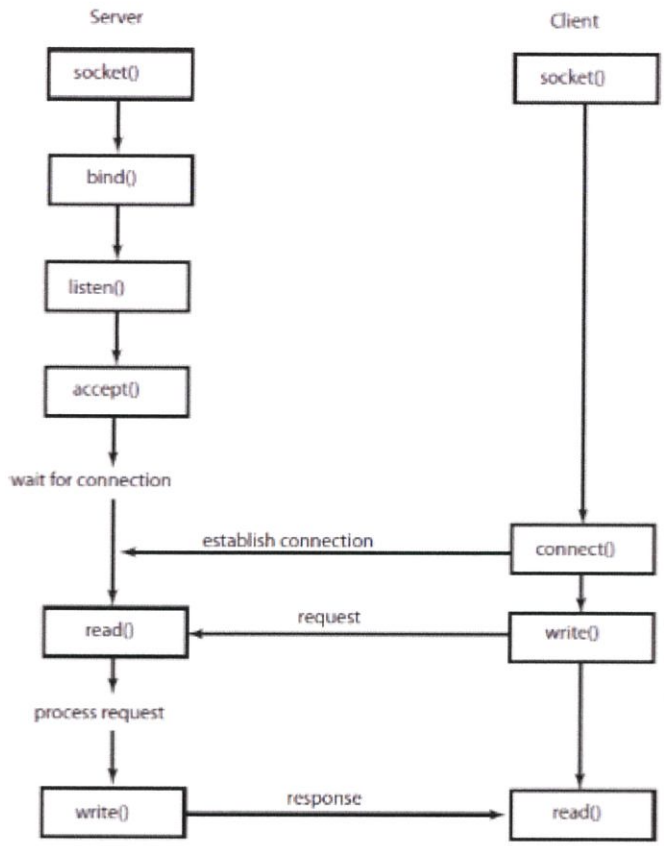
2) ผูก (Bind) อ็อบเจกต์ของซ็อกเก็ตที่ได้จากขั้นตอนที่ 1 เข้ากับหมายเลขไอพีแอดเดรสและพอร์ต

3) กำหนดจำนวนไคลเอนต์ที่สามารถเข้าใช้งานเซิร์ฟเวอร์ได้พร้อม ๆ กัน

4) รอรับการเชื่อมต่อ, สื่อสารข้อมูล และยุติการเชื่อมต่อจากไคลเอนต์ แสดงดังรูปที่ 2.17 และ 2.18



รูปที่ 2.17 การทำงานของเซิร์ฟเวอร์



รูปที่ 2.18 ภาพรวมการทำงานของไคลเอนต์ – เซิร์ฟเวอร์

สำหรับตัวอย่างโปรแกรมฝั่งเซิร์ฟเวอร์แสดงดังรูปที่ 2.19

```
''' Server for sending message'''
from socket import *

HOST = 'localhost'
PORT = 5000
BUFFER_SIZE = 1024
ADDRESS = (HOST, PORT) #(127.0.0.1, 5000)
MESSAGE = 'Hello World !'

server = socket(AF_INET, SOCK_STREAM
server.bind(ADDRESS)
server.listen(5)

while True:
    print('waiting for connection...')
    client, address = server.accept()
    print('connected from: ', address)
    data = str.encode(MESSAGE)
    client.send(data)
    client.close()
```

รูปที่ 2.19 ตัวอย่างโปรแกรมฝั่งเซิร์ฟเวอร์

จากตัวอย่างโปรแกรม แสดงโปรแกรมฝั่งเซิร์ฟเวอร์ ทำหน้าที่รอรับการเชื่อมต่อจากไคลเอนต์และจะส่งข้อความให้กับโปรแกรมฝั่งไคลเอนต์นำไปแสดงผล บรรทัดที่ 10 สร้างซ็อกเก็ตของเซิร์ฟเวอร์ชื่อ server บรรทัดที่ 12 กำหนดให้เซิร์ฟเวอร์สามารถรองรับการเชื่อมต่อจากไคลเอนต์ได้พร้อม ๆ กันเท่ากับ 5 เครื่อง บรรทัดที่ 14 กำหนดให้โปรแกรมทำงานตลอดเวลาเพื่อรอรับการเชื่อมต่อจากไคลเอนต์ บรรทัดที่ 15 เซิร์ฟเวอร์สั่งพิมพ์ข้อความว่า 'waiting for connection...' จากนั้นเซิร์ฟเวอร์จะเฝ้ารอการเชื่อมต่อจากไคลเอนต์ เมื่อไคลเอนต์เชื่อมต่อเข้ามา (บรรทัดที่ 16) และเซิร์ฟเวอร์ตอบรับการเชื่อมต่อแล้ว เซิร์ฟเวอร์จะพิมพ์ข้อความว่า 'connected from: ' ตามด้วยหมายเลขไอพีและพอร์ตตามลำดับ ดังในบรรทัดที่ 17 ในไพธอนเวอร์ชัน 3.0 ขึ้นไป จะไม่รองรับการส่งสตริงผ่านซ็อกเก็ต จึงจำเป็นต้องเปลี่ยนข้อความเป็นข้อมูลชนิดไบต์เสียก่อนด้วยเมธอด str.encode(MESSAGE) และเก็บไว้ในตัวแปรชื่อ data แสดงในบรรทัดที่ 18 จากนั้นโปรแกรมจะส่งวันเวลาที่ถูกแปลงเป็นไบต์แล้วให้กับเครื่องไคลเอนต์ด้วยเมธอด send() เมื่อเซิร์ฟเวอร์ส่งข้อมูลให้ไคลเอนต์เสร็จเรียบร้อยแล้ว จะปิดการเชื่อมต่อทันทีด้วยเมธอด close() [11]

บทที่ 3

การออกแบบและการจัดทำปริญญานิพนธ์

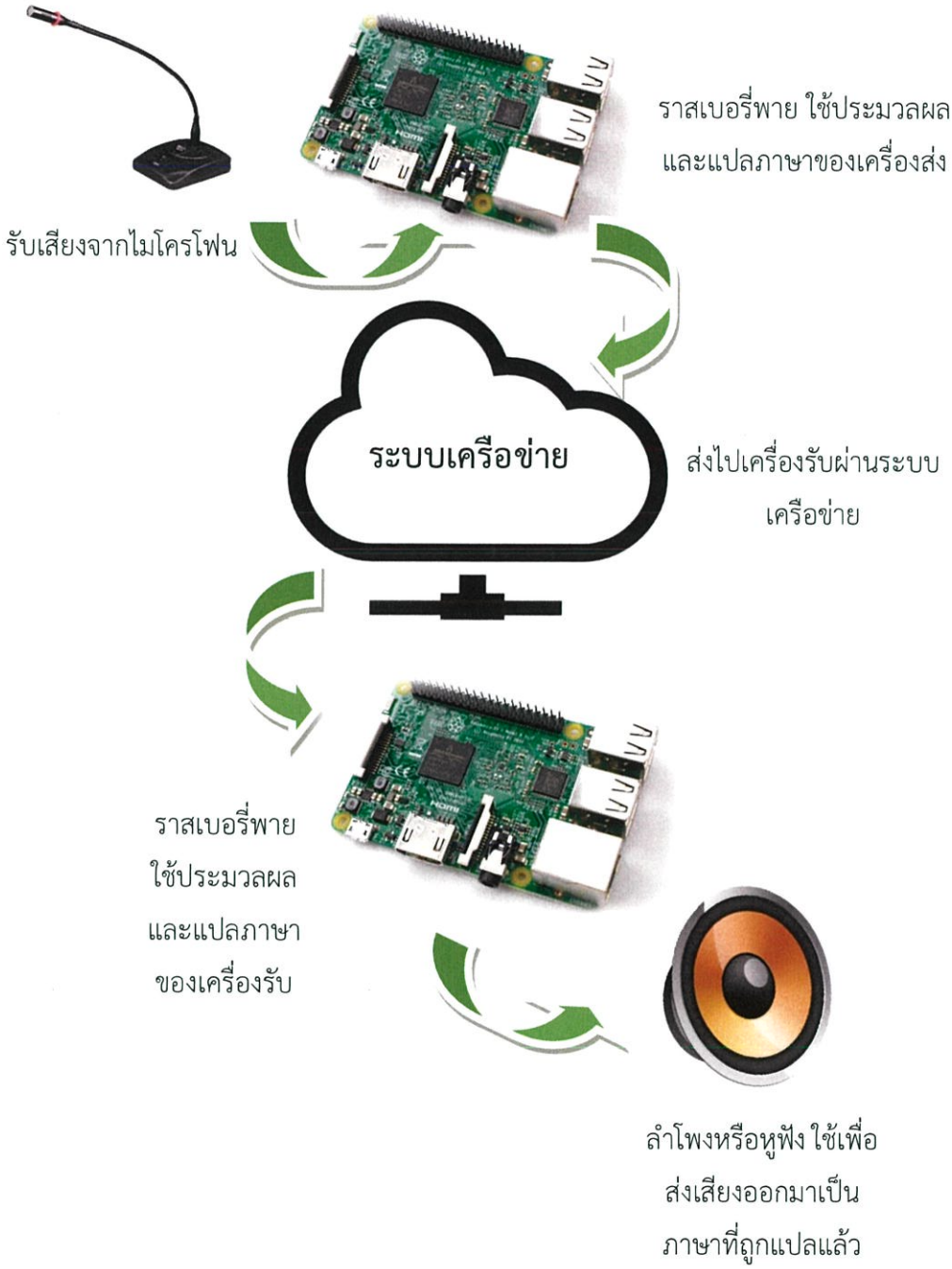
เนื่องจากการประชุม หรือการปรึกษาหารือต่าง ๆ ระหว่างบุคคลของแต่ละประเทศมีความยากลำบากในการสื่อสาร เพราะแต่ละประเทศก็จะมีภาษาต่างกันตามประเทศของตน ซึ่งการประชุม หรือการปรึกษาหารือนี้ มีมากขึ้นทุกวันในปัจจุบัน หากต้องการความสะดวกในการสื่อสารก็ต้องใช้ล่ามในการแปลภาษา ซึ่งเป็นการเปลืองทรัพยากรบุคคล และถ้าหากว่ามีอุปกรณ์ที่สามารถแปลภาษาโดยฟังเสียงพูดจากภาษาเริ่มต้นแล้วแปลเป็นเสียงพูดในอีกภาษาหนึ่งซึ่งเป็นภาษาที่ต้องการ ก็จะทำให้เกิดความสะดวกสบายยิ่งขึ้น

3.1 การออกแบบ

การออกแบบอุปกรณ์แปลภาษาจากเสียงพูด ประกอบไปด้วย 2 ส่วน โดยส่วนแรกคือการออกแบบโปรแกรมสำหรับการทำงานของอุปกรณ์ และส่วนที่สองคือการออกแบบโครงสร้างของอุปกรณ์

3.1.1 การทำงานของระบบ

การทำงานของระบบแปลภาษาจากเสียงพูด เมื่อผู้พูดพูดผ่านไมโครโฟน ระบบจะทำการแปลงเสียงของผู้พูดไปเป็นตัวอักษร และข้อความนั้นจะถูกส่งไปยังเครื่องผู้ฟังผ่านระบบเครือข่าย จากนั้นเครื่องผู้ฟังจะนำข้อความที่ได้รับมาแปลเป็นภาษาที่เลือก และแปลงจากข้อความที่ถูกแปลเป็นไฟล์เสียง ก่อนที่จะเล่นไฟล์เสียงนั้นให้ผู้ฟังได้ยินผ่านหูฟังหรือลำโพงต่อไป ดังรูปที่ 3.1

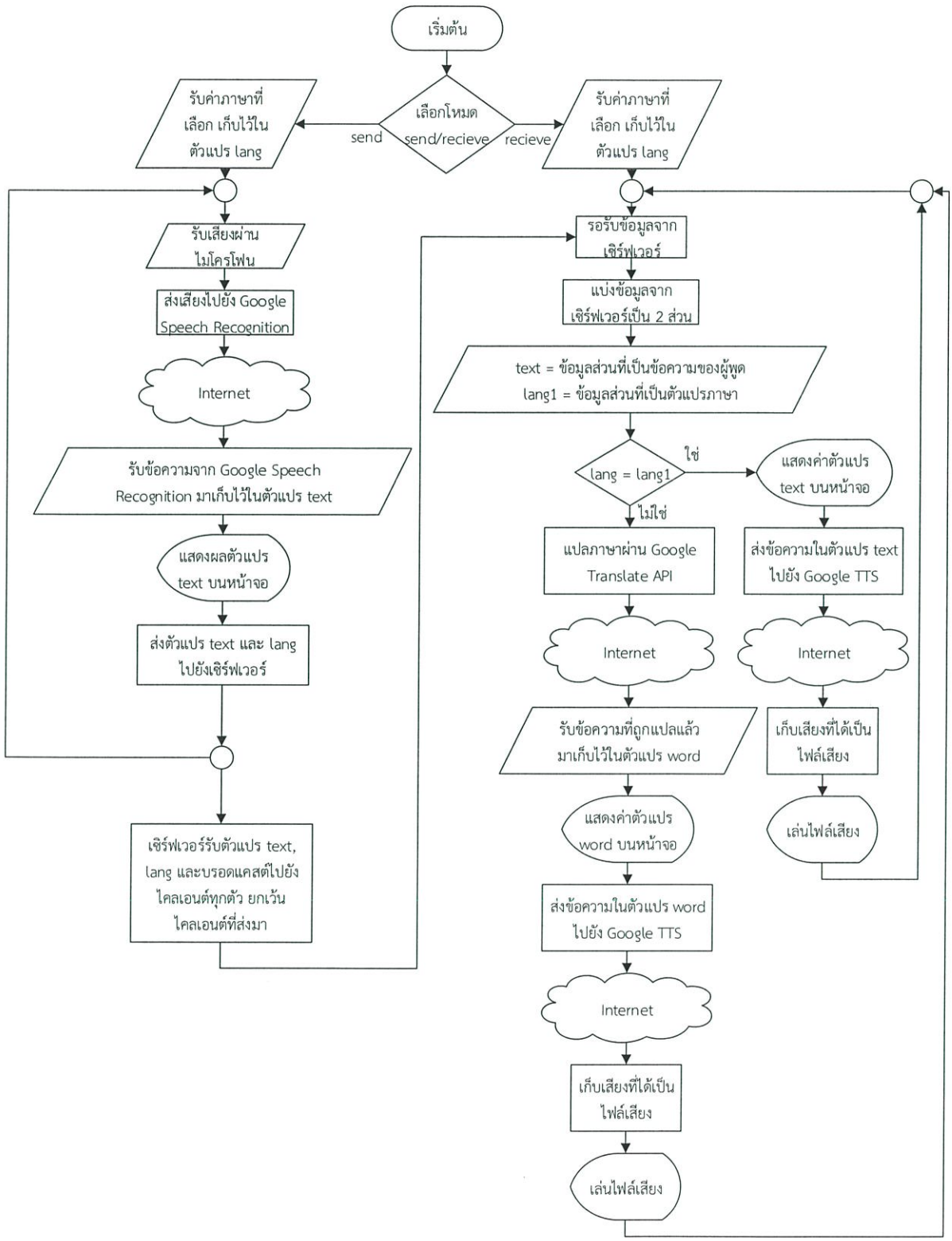


รูปที่ 3.1 บล็อกไดอะแกรมรวมของระบบ

3.1.2 ออกแบบโปรแกรม

3.1.2.1 โปรแกรมสำหรับการทำงานของอุปกรณ์

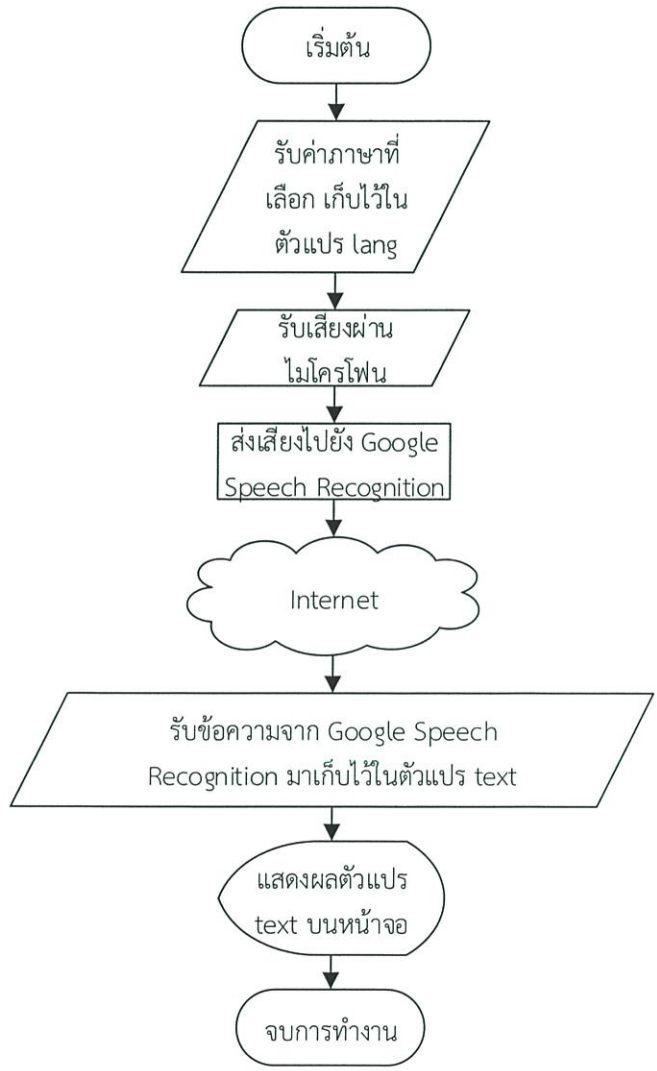
ขั้นตอนการทำงานของอุปกรณ์ เริ่มจากการเลือกโหมดระหว่างผู้พูดและผู้ฟัง จากนั้นทำการเลือกภาษาของตนเอง โดยเมื่อผู้พูดพูดผ่านไมโครโฟนระบบจะทำการแปลงเสียงของผู้พูดไปเป็นข้อความ ข้อความและตัวแปรที่เก็บตัวแปรของแต่ละภาษาจากเครื่องผู้พูดจะถูกส่งไปยังเซิร์ฟเวอร์ แล้วเซิร์ฟเวอร์จะทำการบรอดแคสต์ข้อความและตัวแปรของภาษาที่ได้รับมาไปยังเครื่องผู้ฟังทั้งหมด จากนั้นเครื่องผู้ฟังจะนำตัวแปรของภาษาที่ได้รับมาตัดสินใจว่าเป็นภาษาเดียวกับเครื่องตัวเองหรือไม่ หากเป็นภาษาเดียวกันจะทำการแปลงข้อความที่ได้รับมาเป็นไฟล์เสียง แล้วเล่นไฟล์เสียงนั้น แต่หากเป็นคนละภาษาข้อความที่ได้รับมาจะถูกแปลเป็นภาษาของเครื่องผู้ฟังก่อน และแปลงจากข้อความที่ถูกแปลแล้วเป็นไฟล์เสียง ก่อนที่จะเล่นไฟล์เสียงนั้นให้ผู้ฟังได้ยินผ่านหูฟังหรือลำโพงต่อไป โดยสามารถแบ่งการทำงานของโปรแกรมได้เป็น โปรแกรมแปลงเสียงเป็นข้อความ โปรแกรมแปลภาษา โปรแกรมแปลงข้อความเป็นเสียง และโปรแกรมส่งข้อมูลผ่านระบบเครือข่าย มีโฟลว์ชาร์ตดังรูปที่ 3.2



รูปที่ 3.2 โฟลว์ชาร์ตของโปรแกรมสำหรับการทำงานของอุปกรณ์

3.1.2.2 โปรแกรมแปลงเสียงเป็นข้อความ

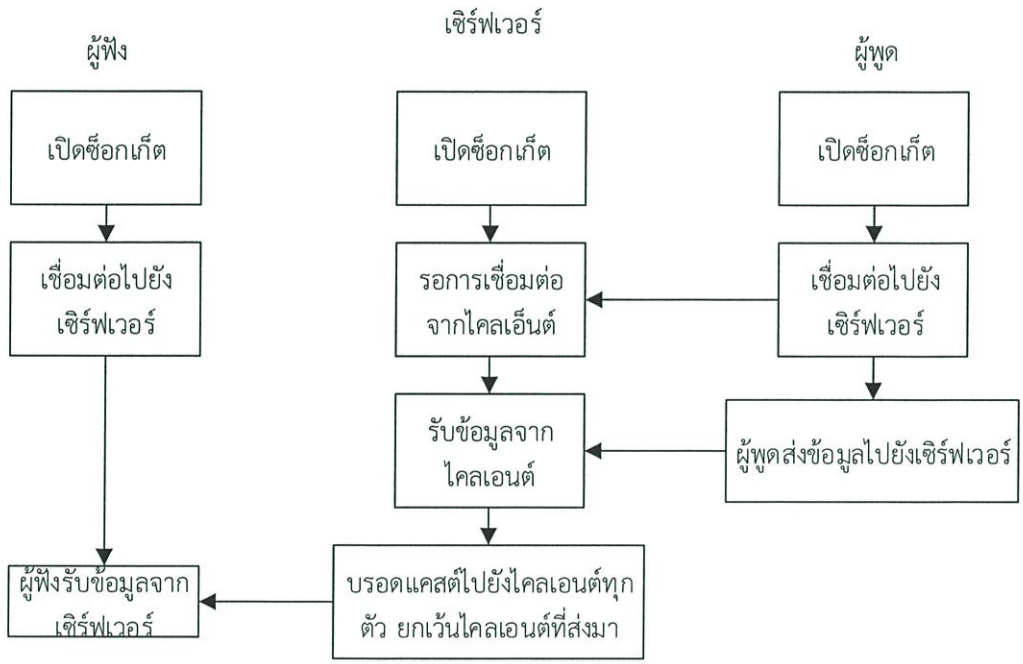
ขั้นตอนการทำงานของโปรแกรมแปลงข้อความเป็นเสียงนั้น เริ่มต้นจากกำหนดภาษาที่จะพูด จากนั้นพูดผ่านไมโครโฟน อุปกรณ์แปลภาษาจะทำการส่งเสียงที่ได้รับไปยังระบบรู้จำเสียงพูดของกูเกิ้ล (Google Speech Recognition) แล้วระบบรู้จำเสียงของกูเกิ้ลจะส่งเป็นข้อความกลับมาเก็บไว้ในตัวแปร และแสดงผลข้อความที่ได้บนจอแสดงผล โดยโฟลว์ชาร์ตแสดงการทำงานของโปรแกรมแปลงเสียงเป็นข้อความ ดังแสดงในรูปที่ 3.3



รูปที่ 3.3 โฟลว์ชาร์ตแสดงการทำงานของโปรแกรมแปลงเสียงเป็นข้อความ

3.1.2.3 โปรแกรมส่งข้อมูลผ่านระบบเครือข่าย

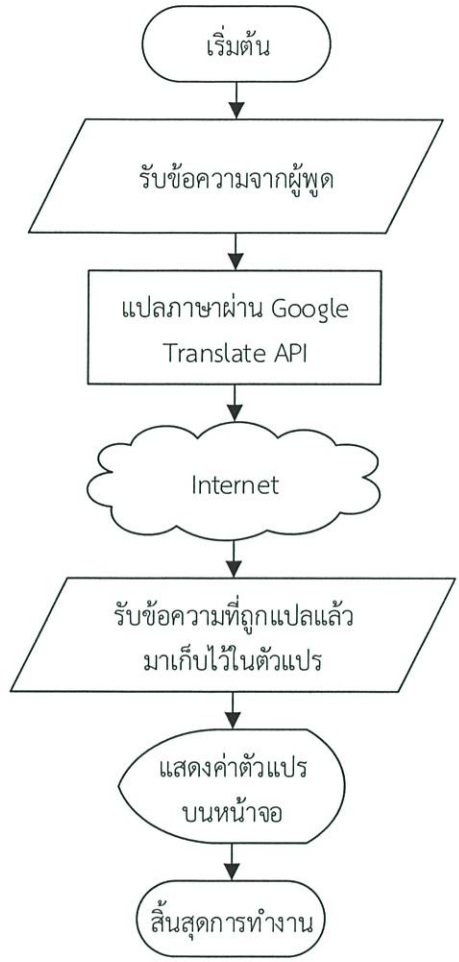
ขั้นตอนการทำงานของโปรแกรมเริ่มต้นจากเครื่องเซิร์ฟเวอร์เปิดซ็อกเก็ตเพื่อรอการเชื่อมต่อจากเครื่องไคลเอนต์ เมื่อเครื่องไคลเอนต์เชื่อมต่อไปยังเครื่องเซิร์ฟเวอร์ เครื่องเซิร์ฟเวอร์จะรอรับข้อมูลที่ถูกส่งมาจากไคลเอนต์ จากนั้นเครื่องผู้พูดส่งข้อความไปยังเซิร์ฟเวอร์ ด้านเซิร์ฟเวอร์ก็จะบรอดแคสต์ไปยังเครื่องผู้ฟังทุกเครื่อง เมื่อผู้ฟังได้รับจะสามารถนำข้อความที่ได้รับไปใช้ต่อไป โฟลว์ชาร์ตแสดงการทำงานของโปรแกรมส่งข้อมูลผ่านระบบเครือข่าย ดังแสดงในรูปที่ 3.4



รูปที่ 3.4 โฟลว์ชาร์ตแสดงการทำงานของโปรแกรมการส่งข้อมูลผ่านระบบเครือข่าย

3.1.1.4 โปรแกรมแปลภาษา

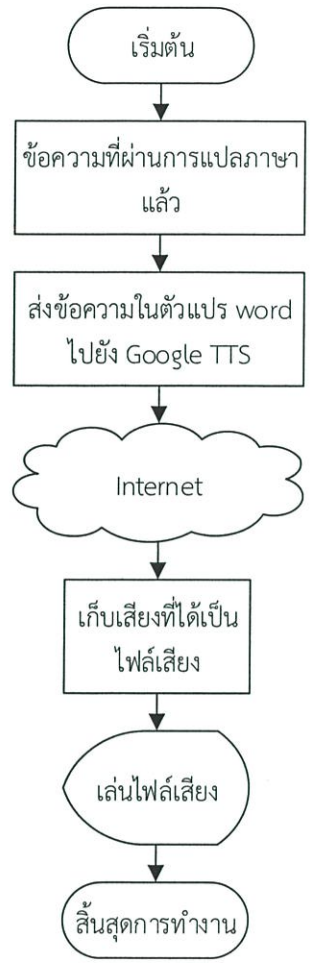
ขั้นตอนการทำงานของโปรแกรมเริ่มต้นจากรับข้อความจากผู้พูด และนำข้อความนั้นส่งไปยังระบบแปลภาษาของกูเกิ้ล (Google Translate API) จากนั้นรับข้อความที่ถูกแปลแล้วมาแสดงผลบนหน้าจอแสดงผล โฟลว์ชาร์ตแสดงการทำงานของโปรแกรมแปลภาษา ดังแสดงในรูปที่ 3.5



รูปที่ 3.5 โฟลว์ชาร์ตแสดงการทำงานของโปรแกรมแปลภาษา

3.1.1.4 โปรแกรมแปลงข้อความเป็นเสียง

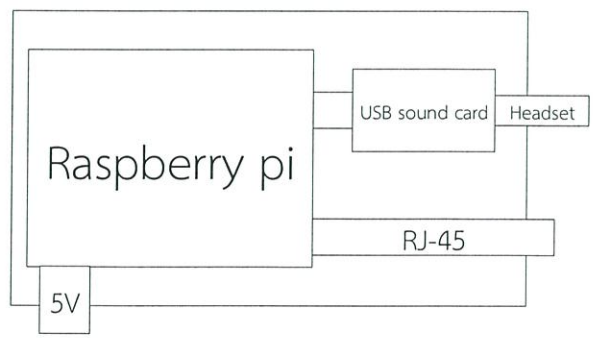
ขั้นตอนการทำงานของโปรแกรมนั้นเป็นการนำข้อความที่ถูกแปลแล้วมาแปลงจากข้อความเป็นเสียง โดยส่งข้อความไปยังระบบวิเคราะห์เสียงของกูเกิ้ล (Google TTS) จากนั้นเสียงที่ได้จะถูกเก็บเป็นไฟล์เสียง และเล่นเสียงผ่านลำโพงหรือหูฟัง โฟลว์ชาร์ตแสดงการทำงานของโปรแกรมโปรแกรมแปลงข้อความเป็นเสียง ดังแสดงในรูปที่ 3.6



รูปที่ 3.6 โฟลว์ชาร์ตแสดงการทำงานของโปรแกรมโปรแกรมแปลงข้อความเป็นเสียง

3.1.3 การออกแบบโครงสร้าง

ในการใช้งานนั้นต้องออกแบบอุปกรณ์แปลภาษาให้มีขนาดเล็ก เพื่อให้สามารถเคลื่อนย้ายได้สะดวก โดยต้องต่ออุปกรณ์ต่าง ๆ เข้ากับราสเบอรี่พายเพื่อให้สามารถทำงานได้



รูปที่ 3.7 โครงสร้างของอุปกรณ์

3.2 เครื่องมือที่ใช้ในการทดลอง

3.2.1 บอร์ดราสเบอรี่พาย

ใช้สำหรับการประมวลผลข้อมูล และต่ออุปกรณ์ต่าง ๆ เข้ากับบอร์ดราสเบอรี่พาย ไม่ว่าจะเป็น หูฟัง, การ์ดเสียงแบบยูเอสบี, จอแสดงผลแอลซีดีแบบทีเอฟทีขนาด 3.5 นิ้ว, สายสัญญาณอาร์เจ-45 ซึ่งราสเบอรี่พายมีลักษณะดังแสดงในรูปที่ 3.8

รูปที่ 3.8 บอร์ดราสเบอรี่พาย

3.2.2 การ์ดเสียงแบบยูเอสบี

ใช้ในการต่อไมโครโฟนสำหรับราสเบอรี่พาย เนื่องจากช่องเสียบหูฟัง 3.5 มิลลิเมตร ของราสเบอรี่พายไม่รองรับสัญญาณเสียงขาเข้าได้ จึงต้องต่อไมโครโฟนผ่านการ์ดเสียงแบบยูเอสบี โดยการ์ดเสียงแบบยูเอสบีมีลักษณะดังแสดงในรูปที่ 3.9

รูปที่ 3.9 การ์ดเสียงแบบยูเอสบี

3.2.3 จอแสดงผลแอลซีดีแบบทีเอฟที ขนาด 3.5 นิ้ว

ใช้สำหรับการแสดงผลกราฟฟิคเพื่อให้ผู้ใช้งานสามารถเลือกโหมดผู้ฟังหรือผู้พูด เลือกภาษาผ่านการสัมผัสปุ่มที่หน้าจอแสดงผล และสามารถอ่านข้อความที่ถูกแปลภาษาแล้วได้ด้วย ดังแสดงในรูปที่ 3.10

รูปที่ 3.10 จอแสดงผลแอลซีดีแบบทีเอฟที ขนาด 3.5 นิ้ว

3.2.4 เฮดเซ็ท

ในการพูดและฟังเสียงนั้นจะเกิดขึ้นผ่านเฮดเซ็ท เพื่อลดเสียงรบกวนหากใช้ลำโพง ซึ่งจะทำให้ระบบเกิดความผิดพลาดได้ โดยเฮดเซ็ทมีลักษณะดังแสดงในรูปที่ 3.11

รูปที่ 3.11 เฮดเซ็ท

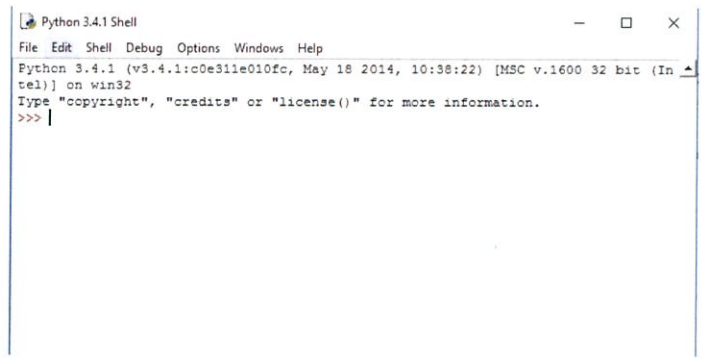
3.2.5 สายสัญญาณอาร์เจ-45

ใช้สำหรับเชื่อมการต่อกับเราเตอร์เพื่อให้สามารถเชื่อมต่ออินเตอร์เน็ตได้ โดยในขั้นตอนต่าง ๆ ต้องมีการติดต่อกับกูเกิ้ล จึงจำเป็นที่ต้องทำให้อุปกรณ์สามารถเชื่อมต่ออินเตอร์เน็ตได้ตลอดเวลา โดยสายสัญญาณอาร์เจ-45 มีลักษณะดังแสดงในรูปที่ 3.12

รูปที่ 3.12 สายสัญญาณอาร์เจ-45

3.2.6 โปรแกรมสำหรับเขียนโปรแกรมภาษาไพทอน

ใช้ในการเขียนชุดคำสั่งของโปรแกรมทั้งหมด เพื่อให้ระบบสามารถทำงานได้ตามที่ต้องการ โดยโปรแกรมสำหรับเขียนโปรแกรมภาษาไพทอนมีลักษณะดังรูปที่ 3.13



รูปที่ 3.13 โปรแกรมสำหรับเขียนโปรแกรมภาษาไพทอน

3.3 การจัดเก็บผลการทดลอง

3.3.1 เก็บผลลัพธ์ของโปรแกรมแปลงเสียงเป็นข้อความ

ทำการเก็บผลลัพธ์ของโปรแกรมแปลงเสียงเป็นข้อความ โดยการถ่ายภาพหน้าจอ และเก็บผลความหน่วงเวลารวมถึงความผิดพลาดขณะที่รอบข้างไม่มีเสียงรบกวน และมีเสียงรบกวน โดยเริ่มจับเวลาตั้งแต่ผู้พูดพูดจบ จนถึงหน้าจอแสดงผลข้อความที่พูดออกมา ทดลองอย่างละ 10 ครั้ง จากนั้นนำค่าความหน่วงเวลาทั้งหมดมาหาค่าเฉลี่ย และนำค่าความผิดพลาดมาคิดเป็นเปอร์เซ็นต์

3.3.2 เก็บผลลัพธ์ของโปรแกรมส่งข้อมูลผ่านระบบเครือข่าย

ทำการเก็บผลลัพธ์ของโปรแกรมส่งข้อมูลผ่านระบบเครือข่ายโดยการถ่ายภาพหน้าจอทั้งทางเซิร์ฟเวอร์ และทางด้านไคลเอนต์

3.3.3 เก็บผลลัพธ์ของโปรแกรมแปลภาษา

ทำการเก็บผลลัพธ์ของโปรแกรมแปลภาษาโดยการถ่ายภาพหน้าจอ และเก็บผลความหน่วงเวลา โดยทำการกำหนดภาษาเริ่มต้น ภาษาที่ต้องการจะแปล และข้อความที่จะแปล จากนั้นเริ่มจับเวลาเมื่อเริ่มต้นโปรแกรม จนถึงหน้าจอแสดงผลข้อความที่ถูกแปลแล้ว ทำการทดลองทั้งหมด 10 ครั้ง จากนั้นนำค่าความหน่วงเวลาทั้งหมดมาหาค่าเฉลี่ย

3.3.4 เก็บผลลัพธ์ของโปรแกรมแปลงข้อความเป็นเสียง

ทำการเก็บผลลัพธ์ของโปรแกรมแปลงข้อความเป็นเสียงโดยการถ่ายภาพหน้าจอ และเก็บผลความหน่วงเวลา โดยทำการกำหนดข้อความที่ต้องการแปลงเป็นเสียงไว้แล้ว เริ่มจับเวลาเมื่อเริ่มต้นโปรแกรม จนถึงเริ่มได้ยินเสียงอ่านของข้อความนั้น ทำการทดลองทั้งหมด 10 ครั้ง จากนั้นนำค่าความหน่วงเวลาทั้งหมดมาหาค่าเฉลี่ย

3.3.5 เก็บค่าความหน่วงเวลาของระบบ

ทำการเก็บผลความหน่วงเวลา โดยเปรียบเทียบระหว่างผู้พูดที่มีเพศและวัยที่แตกต่างกัน ทำการทดลองโดยเริ่มนับจากผู้พูดพูดเสร็จ จนถึงผู้รับเริ่มได้ยินเสียงของข้อความที่ผ่านการแปลภาษาแล้ว โดยทำการทดลองทั้งหมด 10 ครั้ง จากนั้นหาความหน่วงเวลาทั้งหมดมาหาค่าเฉลี่ย

3.3.6 เก็บผลลัพธ์ของหน้าจอกราฟฟิค

ทำการเก็บผลลัพธ์ของโปรแกรมแปลงข้อความเป็นเสียงโดยการถ่ายภาพหน้าจอ

บทที่ 4

ผลการทดลอง

4.1 ผลการทดสอบชุดคำสั่ง

4.1.1 ผลการทดสอบชุดคำสั่งของโปรแกรมแปลงเสียงเป็นข้อความ

ขั้นตอนการทำงานของโปรแกรมแปลงข้อความเป็นเสียงนั้น เริ่มต้นจากกำหนดภาษาที่จะพูด จากนั้นพูดผ่านไมโครโฟน อุปกรณ์แปลภาษาจะทำการส่งเสียงที่ได้รับไปยังระบบรู้จำเสียงพูดของกูเกิ้ล (Google Speech Recognition) แล้วระบบรู้จำเสียงของกูเกิ้ลจะส่งเป็นข้อความกลับมาเก็บไว้ในตัวแปร และแสดงผลข้อความที่ได้บนจอแสดงผล จะได้ผลลัพธ์ดังรูปที่ 4.1 และตารางวิเคราะห์ความหน่วงเวลาและเปอร์เซ็นต์ความผิดพลาดของโปรแกรมแปลงเสียงเป็นข้อความขณะมีเสียงรบกวนและไม่มีเสียงรบกวนดังตารางที่ 4.1 และ 4.2 ตามลำดับ

```
Your language     : en
Please talk ..
You said          : " Life is Beautiful with people like you in it "
>>>
```

รูปที่ 4.1 ผลการทดสอบชุดคำสั่งของโปรแกรมแปลงเสียงเป็นข้อความ

ตารางที่ 4.1 ตารางวิเคราะห์ความหน่วงเวลาและเปอร์เซ็นต์ความผิดพลาดของโปรแกรมแปลงเสียงเป็นข้อความขณะที่ไม่มีเสียงรบกวน

ข้อความ	ครั้ง	ความหน่วงเวลา(วินาที)	ครั้ง	ความผิดพลาด
A quick brown fox jumps over the lazy dog.	1	5.6	1	ข้อความตรงกับที่พูด
	2	6.7	2	ข้อความตรงกับที่พูด
	3	6.1	3	ข้อความไม่ตรงกับที่พูด
	4	5.2	4	ข้อความตรงกับที่พูด
	5	5.7	5	ข้อความตรงกับที่พูด
	6	6.4	6	ข้อความตรงกับที่พูด
	7	5.6	7	ข้อความตรงกับที่พูด
	8	3.4	8	ข้อความตรงกับที่พูด

ข้อความ	ครั้ง	ความหน่วงเวลา(วินาที)	ครั้ง	ความผิดพลาด
A quick brown fox jumps over the lazy dog.	9	5.4	9	ข้อความตรงกับที่พูด
	10	4.5	10	ข้อความตรงกับที่พูด
	เฉลี่ย	**5.5**	**รวม**	**10%**

ตารางที่ 4.2 ตารางวิเคราะห์ความหน่วงเวลาและเปอร์เซ็นต์ความผิดพลาดของโปรแกรมแปลงเสียงเป็นข้อความขณะมีเสียงรบกวน

ข้อความ	ครั้ง	ความหน่วงเวลา(วินาที)	ครั้ง	ความผิดพลาด
A quick brown fox jumps over the lazy dog.	1	8.5	1	ข้อความตรงกับที่พูด
	2	9.1	2	ข้อความตรงกับที่พูด
	3	9.1	3	ข้อความไม่ตรงกับที่พูด
	4	9.0	4	ข้อความตรงกับที่พูด
	5	9.1	5	ข้อความตรงกับที่พูด
	6	9.1	6	ข้อความไม่ตรงกับที่พูด
	7	9.0	7	ข้อความตรงกับที่พูด
	8	9.2	8	ข้อความตรงกับที่พูด
	9	9.1	9	ข้อความไม่ตรงกับที่พูด
	10	9.1	10	ข้อความตรงกับที่พูด
	เฉลี่ย	**9.1**	**รวม**	**30%**

4.1.2 ผลการทดสอบชุดคำสั่งของโปรแกรมสำหรับการส่งข้อมูลผ่านระบบเครือข่าย

เครื่องเซิร์ฟเวอร์เปิดซ็อกเก็ตเพื่อรอการเชื่อมต่อจากเครื่องไคลเอนต์ เมื่อเครื่องไคลเอนต์เชื่อมต่อไปยังเครื่องเซิร์ฟเวอร์ เครื่องเซิร์ฟเวอร์จะรอรับข้อมูลที่ถูกส่งมาจากไคลเอนต์ จากนั้นเครื่องผู้พูดส่งข้อความไปยังเซิร์ฟเวอร์ ด้านเซิร์ฟเวอร์ก็จะบรอดแคสต์ไปยังเครื่องผู้ฟังทุกเครื่อง เมื่อผู้ฟังได้รับจะสามารถนำข้อความที่ได้รับไปใช้ต่อไป โดยโปรแกรมจะแบ่งเป็น 2 ส่วน ได้แก่

4.1.2.1 ด้านเซิร์ฟเวอร์ จะทำการเปิดซ็อกเก็ตเพื่อรอรับการเชื่อมต่อมาจากฝั่งไคลเอนต์ โดยมีผลการทำงานดังรูปที่ 4.2, 4.3 และ 4.4

```
puripoom@puripoom-X450JF ~/project $ python chat_server.py
Chat server started on port 9009
```

รูปที่ 4.2 แสดงผลการทำงานของโปรแกรมด้านเซิร์ฟเวอร์ ขณะรอรับการเชื่อมต่อจากไคลเอนต์

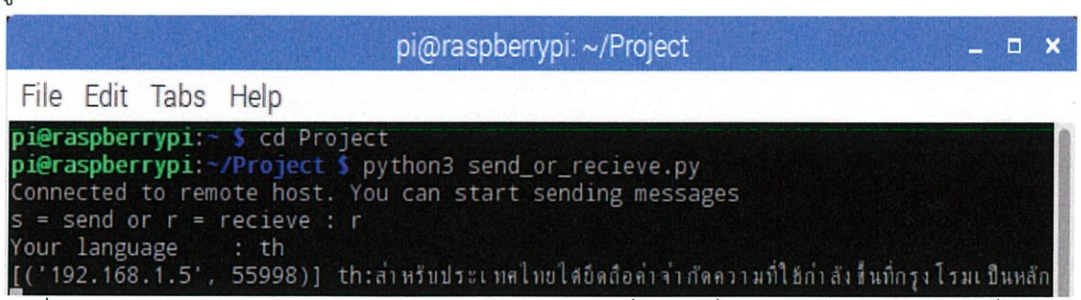
```
puripoom@puripoom-X450JF ~/project $ python chat_server.py
Chat server started on port 9009
Client (192.168.1.5, 55996) connected
Client (192.168.1.5, 55998) connected
Client (192.168.1.5, 56000) connected
```

รูปที่ 4.3 แสดงผลการทำงานของโปรแกรมด้านเซิร์ฟเวอร์ เมื่อมี 3 ไคลเอนต์เข้ามาเชื่อมต่อ

```
puripoom@puripoom-X450JF ~/project $ python chat_server.py
Chat server started on port 9009
Client (192.168.1.5, 55996) connected
Client (192.168.1.5, 55998) connected
Client (192.168.1.5, 56000) connected
[192.168.1.5:56000] th:สำหรับประเทศไทยได้ยึดถือคำจำกัดความที่ใช้กำลังขึ้นที่กรุงโรมเป็นหลัก
```

รูปที่ 4.4 แสดงผลการทำงานเมื่อเครื่องผู้พูดส่งข้อความมายังเซิร์ฟเวอร์

4.1.2.2 ด้านไคลเอนต์ เสียงที่ถูกแปลงเป็นข้อความของเครื่องผู้พูด จะถูกส่งไปยังเครื่องเซิร์ฟเวอร์ และเซิร์ฟเวอร์จะทำการบรอดแคสต์ไปยังเครื่องผู้ฟังทุกเครื่อง จะได้ผลลัพธ์ดังรูปที่ 4.5



รูปที่ 4.5 แสดงผลการทำงานของโปรแกรมด้านไคลเอนต์ที่เป็นเครื่องผู้ฟังจะแสดงข้อความที่ผู้ส่งพูด

4.1.3 ผลการทดสอบชุดคำสั่งของโปรแกรมแปลภาษา

การทำงานของโปรแกรมเริ่มต้นจากเลือกภาษาต้นทางและภาษาปลายทาง รับข้อความจากผู้พูด และนำข้อความนั้นส่งไปยังระบบแปลภาษาของกูเกิ้ล (Google Translate API) จากนั้นรับข้อความที่ถูกแปลแล้วมาแสดงผลบนหน้าจอแสดงผล จะได้ผลลัพธ์ดังรูปที่ 4.6 และ 4.7

```
listening..
You said          : " สวัสดีครับ "
processing
listening..
You said          : " ทำหอพักขอบพระคุณลูกค้าทุกท่าน "
processing
listening..
You said          : " ให้ความร่วมมืออย่างดีโดยเสมอมา "
```

รูปที่ 4.6 แสดงผลการทำงานของฝั่งผู้พูด โดยแปลงจากเสียงเป็นตัวอักษร

```
pi@raspberrypi:~ $ python3 Project/all1.py
Connected to remote host. You can start sending messages
press red button to be sender
press black button to select language
Language : English
Cannot connect to server socket err = No such file or directory
Cannot connect to server request channel
jack server is not running or cannot be started
Translate  text to  : "Hello"
Translate  text to  : "Residence thank all customers"
Translate  text to  : "Always cooperated well."
```

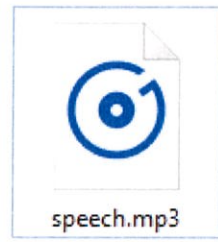
รูปที่ 4.7 แสดงผลการทำงานของฝั่งผู้ฟัง โดยจะแปลข้อความที่ผู้พูดส่งมาเป็นภาษาที่เลือก

ตารางที่ 4.3 ตารางวิเคราะห์ความหน่วงเวลาของโปรแกรมแปลภาษา

ข้อความ	ครั้ง	ความหน่วงเวลา (วินาที)
A quick brown fox jumps over the lazy dog.	1	2.5
	2	1.9
	3	2.3
	4	2.6
	5	1.7
	6	1.9
	7	2.4
	8	2.1
	9	2.3
	10	2.0
	เฉลี่ย	**2.2**

4.1.4 ผลการทดสอบชุดคำสั่งของโปรแกรมแปลงข้อความเป็นเสียง

การทำงานของโปรแกรมนั้นเป็นการนำข้อความที่ถูกแปลแล้วมาแปลงจากข้อความเป็นเสียง โดยส่งข้อความไปยังระบบวิเคราะห์เสียงของกูเกิ้ล (Google TTS) จากนั้นเสียงที่ได้จะถูกเก็บเป็นไฟล์เสียง และเล่นเสียงผ่านลำโพงหรือหูฟัง ดังรูปที่ 4.8



รูปที่ 4.8 ไฟล์เสียงของข้อความ

ตารางที่ 4.4 ตารางวิเคราะห์ความหน่วงเวลาของโปรแกรมแปลงข้อความเป็นเสียง

ข้อความ	ครั้ง	ความหน่วงเวลา (วินาที)
A quick brown fox jumps over the lazy dog.	1	6.5
	2	6.2
	3	6.1
	4	6.4
	5	5.9
	6	5.8
	7	6.4
	8	6.3
	9	6.2
	10	6.9
	เฉลี่ย	**6.3**

4.1.5 ผลการทดสอบความหน่วงเวลาของระบบ

ตารางที่ 4.5 ตารางวิเคราะห์ความหน่วงเวลาของระบบ (ทดสอบโดยผู้ชายอายุ 23ปี)

ข้อความ	ครั้ง	ความหน่วงเวลา(วินาที)	ข้อความที่แปลงมาจากเสียงพูด	ข้อความที่แปลเป็นภาษาไทย
A quick brown fox jumps over the lazy dog.	1	7.3	A quick brown fox jumps over the lazy dog.	สุนัขจิ้งจอกสีน้ำตาลกระโดดข้ามหมาขี้เกียจ
	2	6.4	A quick brown fox jumps over the lazy dog.	สุนัขจิ้งจอกสีน้ำตาลกระโดดข้ามหมาขี้เกียจ
	3	5.4	A quick brown fox jumps over the lazy dog.	สุนัขจิ้งจอกสีน้ำตาลกระโดดข้ามหมาขี้เกียจ
	4	6.0	A quick brown fox jumps over the lazy dog.	สุนัขจิ้งจอกสีน้ำตาลกระโดดข้ามหมาขี้เกียจ
	5	5.7	A quick brown fox jumps over the lazy dog.	สุนัขจิ้งจอกสีน้ำตาลกระโดดข้ามหมาขี้เกียจ
	6	6.2	A big brown fox jumps over the lazy dog.	สุนัขจิ้งจอกสีน้ำตาลขนาดใหญ่กระโดดข้ามหมาขี้เกียจ
	7	6.3	A quick brown fox jumps over the lazy dog.	สุนัขจิ้งจอกสีน้ำตาลกระโดดข้ามหมาขี้เกียจ
	8	5.8	A quick brown fox jumps over the lazy dog.	สุนัขจิ้งจอกสีน้ำตาลกระโดดข้ามหมาขี้เกียจ

ข้อความ	ครั้ง	ความหน่วงเวลา(วินาที)	ข้อความที่แปลงมาจากเสียงพูด	ข้อความที่แปลเป็นภาษาไทย
A quick brown fox jumps over the lazy dog.	9	5.5	A quick brown fox jumps over the lazy.	สุนัขจิ้งจอกสีน้ำตาลเร็วกระโดดข้ามขี้เกียจ
	10	5.2	A quick brown fox jumps over the lazy dog.	สุนัขจิ้งจอกสีน้ำตาลกระโดดข้ามหมาขี้เกียจ
	เฉลี่ย	**6.0**	**ผิดพลาด 20%**	**ผิดพลาด 20%**

ตารางที่ 4.6 ตารางวิเคราะห์ความหน่วงเวลาของระบบ (ทดสอบโดยผู้หญิงอายุ 22ปี)

ข้อความ	ครั้ง	ความหน่วงเวลา(วินาที)	ข้อความที่แปลงมาจากเสียงพูด	ข้อความที่แปลเป็นภาษาไทย
A quick brown fox jumps over the lazy dog.	1	6.0	A quick brown fox jumped over the lazy dog.	สุนัขจิ้งจอกสีน้ำตาลเร็ว ๆ กระโดดข้ามสุนัขขี้เกียจ
	2	6.4	A quick brown fox jumps over the lazy dog.	สุนัขจิ้งจอกสีน้ำตาลกระโดดข้ามหมาขี้เกียจ
	3	6.7	A quick brown fox jumps over the lazy up.	สุนัขจิ้งจอกสีน้ำตาลเร็วกระโดดขึ้นเหนือขี้เกียจ
	4	6.3	A quick brown fox jumps over the lazy dog.	สุนัขจิ้งจอกสีน้ำตาลกระโดดข้ามหมาขี้เกียจ
	5	5.9	A quick brown fox jumps over the lazy dog.	สุนัขจิ้งจอกสีน้ำตาลกระโดดข้ามหมาขี้เกียจ

ข้อความ	ครั้ง	ความหน่วงเวลา(วินาที)	ข้อความที่แปลงมาจากเสียงพูด	ข้อความที่แปลเป็นภาษาไทย
A quick brown fox jumps over the lazy dog.	6	7.2	A quick brown fox jumped over the lazy dog.	สุนัขจิ้งจอกสีน้ำตาลเร็ว ๆ กระโดดข้ามสุนัขขี้เกียจ
	7	7.1	A quick brown fox jumps over the lazy dog.	สุนัขจิ้งจอกสีน้ำตาลกระโดดข้ามหมาขี้เกียจ
	8	6.7	A quick brown fox jumps over the lazy dog.	สุนัขจิ้งจอกสีน้ำตาลกระโดดข้ามหมาขี้เกียจ
	9	6.5	A quick brown fox jumps over the lazy dog.	สุนัขจิ้งจอกสีน้ำตาลกระโดดข้ามหมาขี้เกียจ
	10	5.8	A quick brown fox jumps over the lazy dog.	สุนัขจิ้งจอกสีน้ำตาลกระโดดข้ามหมาขี้เกียจ
	เฉลี่ย	**6.5**	**ผิดพลาด 30%**	**ผิดพลาด 30%**

ตารางที่ 4.7 ตารางวิเคราะห์ความหน่วงเวลาของระบบ (ทดสอบโดยผู้ชายอายุ 55ปี)

ข้อความ	ครั้ง	ความหน่วงเวลา(วินาที)	ข้อความที่แปลงมาจากเสียงพูด	ข้อความที่แปลเป็นภาษาไทย
A quick brown fox jumps over the lazy dog.	1	6.1	A quick brown fox jumps over the lazy dog.	สุนัขจิ้งจอกสีน้ำตาลกระโดดข้ามหมาขี้เกียจ
	2	7.7	A quick brown fox jumps over the lazy dog.	สุนัขจิ้งจอกสีน้ำตาลกระโดดข้ามหมาขี้เกียจ

ข้อความ	ครั้ง	ความหน่วงเวลา(วินาที)	ข้อความที่แปลงมาจากเสียงพูด	ข้อความที่แปลเป็นภาษาไทย
A quick brown fox jumps over the lazy dog.	3	6.3	A quick brown fox jumps over the lazy day.	สุนัขจิ้งจอกสีน้ำตาลเร็วกระโดดข้ามวันขี้เกียจ
	4	5.8	A quick brown fox jumped over the lazy dog.	สุนัขจิ้งจอกสีน้ำตาลเร็ว ๆ กระโดดข้ามสุนัขขี้เกียจ
	5	6.3	A quick brown fox jumps over the lazy dog.	สุนัขจิ้งจอกสีน้ำตาลกระโดดข้ามหมาขี้เกียจ
	6	6.1	A quick brown fox jumps over the lazy dog.	สุนัขจิ้งจอกสีน้ำตาลกระโดดข้ามหมาขี้เกียจ
	7	5.8	A quick brown fox jumps over the lazy.	สุนัขจิ้งจอกสีน้ำตาลเร็วกระโดดข้ามขี้เกียจ
	8	6.7	A quick brown fox jumps over the lazy dog.	สุนัขจิ้งจอกสีน้ำตาลกระโดดข้ามหมาขี้เกียจ
	9	5.2	A quick brown fox jumps over the lazy dog.	สุนัขจิ้งจอกสีน้ำตาลกระโดดข้ามหมาขี้เกียจ
	10	6.4	A quick brown fox jumps over the lazy dog.	สุนัขจิ้งจอกสีน้ำตาลกระโดดข้ามหมาขี้เกียจ
	เฉลี่ย	**6.2**	**ผิดพลาด 20%**	**ผิดพลาด 20%**

ตารางที่ 4.8 ตารางวิเคราะห์ความหน่วงเวลาของระบบ (ทดสอบโดยผู้หญิงอายุ 54ปี)

ข้อความ	ครั้ง	ความหน่วงเวลา(วินาที)	ข้อความที่แปลงมาจากเสียงพูด	ข้อความที่แปลเป็นภาษาไทย
A quick brown fox jumps over the lazy dog.	1	6.1	A quick brown fox jumps over the lazy dog.	สุนัขจิ้งจอกสีน้ำตาลกระโดดข้ามหมาขี้เกียจ
	2	6.4	A quick brown fox jumps over the lazy dog.	สุนัขจิ้งจอกสีน้ำตาลกระโดดข้ามหมาขี้เกียจ
	3	6.5	A quick brown fox jumps over the lazy dog.	สุนัขจิ้งจอกสีน้ำตาลกระโดดข้ามหมาขี้เกียจ
	4	6.4	A quick brown fox jumps over the lazy dog.	สุนัขจิ้งจอกสีน้ำตาลกระโดดข้ามหมาขี้เกียจ
	5	5.9	A quick brown fox jumps over the lazy dog.	สุนัขจิ้งจอกสีน้ำตาลกระโดดข้ามหมาขี้เกียจ
	6	5.8	A quick brown fox jumps over the lazy dog.	สุนัขจิ้งจอกสีน้ำตาลกระโดดข้ามหมาขี้เกียจ
	7	6.1	A quick brown fox jumps over the lazy dog.	สุนัขจิ้งจอกสีน้ำตาลกระโดดข้ามหมาขี้เกียจ
	8	6.3	A quick brown fox jumps over the lazy dog.	สุนัขจิ้งจอกสีน้ำตาลกระโดดข้ามหมาขี้เกียจ

ข้อความ	ครั้ง	ความหน่วงเวลา(วินาที)	ข้อความที่แปลงมาจากเสียงพูด	ข้อความที่แปลเป็นภาษาไทย
A quick brown fox jumps over the lazy dog.	9	6.2	A quick brown fox jumps over the lazy day.	สุนัขจิ้งจอกสีน้ำตาลเร็วกระโดดข้ามวันขี้เกียจ
	10	6.5	A quick brown fox jumped over the lazy dog.	สุนัขจิ้งจอกสีน้ำตาลเร็ว ๆ กระโดดข้ามสุนัขขี้เกียจ
	เฉลี่ย	**6.2**	**ผิดพลาด 20%**	**ผิดพลาด 20%**

4.1.6 ผลการทดสอบหน้าจอกราฟฟิค

เมื่อทำการรันโปรแกรมกราฟฟิค ซึ่งผู้ใช้งานสามารถเลือกภาษา และโหมดผู้ฟังหรือผู้รับโดยการสัมผัสปุ่มที่หน้าจอ นอกจากนี้ยังสามารถดูข้อความที่ได้รับการแปลแล้วด้วย ซึ่งมีผลลัพธ์ดังรูปที่ 4.9

รูปที่ 4.9 ผลลัพธ์ของโปรแกรมหน้าจอกราฟฟิค

บทที่ 5

สรุปผลและข้อเสนอแนะ

5.1 สรุปผล

การแปลภาษาด้วยเครื่องแปลภาษาอัตโนมัติ เมื่อโปรแกรมเริ่มทำงานผู้ใช้งานต้องเลือกโหมดว่าจะเป็นผู้ฟังหรือผู้พูด หากเลือกเป็นผู้พูดระบบจะรอรับเสียงของผู้ใช้งานผ่านไมโครโฟน เมื่อผู้ใช้งานพูดผ่านไมโครโฟนแล้วระบบจะนำเสียงไปแปลงเป็นข้อความ จากนั้นจะส่งข้อความไปยังเครื่องผู้รับผ่านเซิร์ฟเวอร์ เมื่อผู้รับได้รับข้อความจะนำข้อความนั้นไปแปลเป็นภาษาที่เลือก และจะแปลงข้อความที่ได้จากการแปลเป็นไฟล์เสียง แล้วเล่นให้ผู้ฟังได้ยินต่อไป

ในการทดลองหาความหน่วงเวลาและความผิดพลาดของโปรแกรมแปลงเสียงเป็นข้อความนั้น จะแบ่งออกเป็น 2 แบบ โดยแบบแรกคือ ทดลองในขณะที่ไม่มีเสียงรบกวนจะได้ความหน่วงเวลาเฉลี่ยที่ 5.5 วินาที ค่าความผิดพลาดอยู่ที่ร้อยละ 10 แบบสองคือทดลองในขณะที่มีเสียงรบกวนจะได้ความหน่วงเวลาเฉลี่ยที่ 9.1 วินาที ค่าความผิดพลาดอยู่ที่ร้อยละ 30 จะเห็นได้ว่าหากพูดในที่ที่มีเสียงรบกวนเยอะจะทำให้โปรแกรมประมวลผลได้ช้าและมีความผิดพลาดมาก ทั้งนี้ความผิดพลาดของโปรแกรมแปลงเสียงเป็นข้อความนั้นขึ้นอยู่กับการพูดของผู้ใช้งานด้วยเช่นกัน

ในการทดลองหาความหน่วงเวลาของโปรแกรมแปลภาษาพบว่า ความหน่วงเวลามีค่าเฉลี่ยเท่ากับ 2.2 วินาที

ในการทดลองหาความหน่วงเวลาของโปรแกรมแปลงข้อความเป็นเสียงพบว่าความหน่วงเวลาของข้อความสั้นมีค่าเฉลี่ยเท่ากับ 6.3 วินาที

ในการทดลองหาความหน่วงเวลาของระบบพบว่า ถ้าผู้พูดเป็นผู้ชายอายุ 23 ปีจะมีความหน่วงมีค่าเฉลี่ยเท่ากับ 6 วินาที ค่าความผิดพลาดอยู่ที่ร้อยละ 20 ถ้าผู้พูดเป็นผู้หญิงอายุ 22 ปีจะมีความหน่วงมีค่าเฉลี่ยเท่ากับ 6.5 วินาที ค่าความผิดพลาดอยู่ที่ร้อยละ 30 ถ้าผู้พูดเป็นผู้ชายอายุ 55 ปีจะมีความหน่วงมีค่าเฉลี่ยเท่ากับ 6.2 วินาที ค่าความผิดพลาดอยู่ที่ร้อยละ 20 ถ้าผู้พูดเป็นผู้หญิงอายุ 54 ปีจะมีความหน่วงมีค่าเฉลี่ยเท่ากับ 6.2 วินาที ค่าความผิดพลาดอยู่ที่ร้อยละ 20 ซึ่งพบผู้พูดที่มีเพศและอายุต่างกันนั้นจะมีน้ำเสียงที่แตกต่างกัน แต่ค่าความหน่วงเวลาและค่าความผิดพลาดของผู้พูดแต่ละคนนั้นยังคงมีค่าใกล้เคียงกัน ดังนั้นอุปกรณ์แปลภาษาจากเสียงพูดจึงรองรับการใช้งานได้กับทุกเพศทุกวัย ทั้งนี้ขึ้นอยู่กับความชัดเจนในการพูดของผู้พูดด้วย

5.2 ข้อเสนอแนะ

การทำงานของโปรแกรมจะเกิดการหน่วงเวลาในการทำงานขึ้นโดยแปรผันตรงตามความยาวของประโยค จึงไม่ควรใช้กับประโยคที่ยาวมากเกินไป และนอกจากนี้ถ้าหากทำการแปลประโยคที่ยาวจนเกินไปอาจจะทำให้เกิดความผิดพลาดในการจัดเรียงรูปประโยคหลังจากการแปลภาษาเสร็จสิ้นและทำให้เกิดความคลาดเคลื่อนของความหมายได้

บรรณานุกรม

[1] เริ่มต้นกับ Raspberry Pi ตอนที่ 1 : เกริ่นนำ. [ออนไลน์] เข้าถึงได้จาก:
http://www.sathittham.com/raspberry-pi/rpi-ep-1/

[2] Raspberry Pi Pinout Diagram. [ออนไลน์] เข้าถึงได้จาก:
http://www.jameco.com/Jameco/workshop/circuitnotes/raspberry-pi-circuit-note.html.

[3] DOWNLOADS RASPBIAN JESSIE. [ออนไลน์] เข้าถึงได้จาก:
https://www.raspberrypi.org/downloads/

[4] DOWNLOADS RASPBIAN JESSIE. [ออนไลน์] เข้าถึงได้จาก:
https://www.raspberrypi.org/downloads/raspbian/

[5] Getting Started RaspberryPi (แนะนำเริ่มต้น). [ออนไลน์] เข้าถึงได้จาก:
http://raspberrysource.lnwshop.com/article/getting-started-raspberrypi แนะนำเริ่มต้น

[6] ความหมายของการแปล. [ออนไลน์] เข้าถึงได้จาก:
https://www.gotoknow.org/posts/625348

[7] Sound card ทำงานอย่างไร. [ออนไลน์] เข้าถึงได้จาก:
http://sirapongsound-card.blogspot.com/2010/01/sound-card.html

[8] Speech synthesis. [ออนไลน์] เข้าถึงได้จาก:
https://en.wikipedia.org/wiki/Speech_synthesis

[9] How TFT Displays Work. [ออนไลน์] เข้าถึงได้จาก:
https://www.newhavendisplay.com/tft_page.html

[10] Compare All Resistive Touch Technology. [ออนไลน์] เข้าถึงได้จาก:
http://www.winmate.com.tw/resistive_touch.htm

[11] Tiponut, S. V., "Python Network Programming". Romania, Technical University Timisoara, 2001.

ภาคผนวก

ชุดคำสั่งที่ใช้

ชุดคำสั่งของระบบ

```
import sys
import sys, socket, select
import speech_recognition as sr
import pyaudio
from threading import Thread
from urllib.request import urlopen
from urllib.parse import quote
from gtts import gTTS
from pygame import mixer
from ctypes import *
from contextlib import contextmanager
import time
from PyQt4 import QtGui, QtCore


class LayoutTest(QtGui.QWidget):
    def __init__(self):
        super(LayoutTest, self).__init__()
        self.first_box  = QtGui.QVBoxLayout()
        self.second_box = QtGui.QVBoxLayout()
        self.third_box  = QtGui.QVBoxLayout()
        self.fourth_box = QtGui.QVBoxLayout()

        self.zvbox = QtGui.QVBoxLayout()

        vbox = QtGui.QVBoxLayout()
        vbox.addLayout(self.first_box)
        vbox.addLayout(self.second_box)
```

```
        vbox.addLayout(self.third_box)
        vbox.addLayout(self.fourth_box)
        self.setLayout(vbox)

        self.first_view()

        self.setGeometry(300, 200, 400, 300)
        #self.showFullScreen()

    def keyPressEvent(self, event):
        if event.key() == QtCore.Qt.Key_Escape:
            self.close()

    def first_view(self):
        self.langth = QtGui.QPushButton("TH")
        #self.langth.setGeometry(QtCore.QRect(10, 150, 75, 31))
        #self.langth.setObjectName("langth")
        #self.langth.clicked.connect(self.thstate)

        self.langen = QtGui.QPushButton("EN")
        #self.langen.setGeometry(QtCore.QRect(100, 150, 75, 31))
        #self.langen.setObjectName("langen")
        #self.langen.clicked.connect(self.enstate)

        self.langch = QtGui.QPushButton("CH")
        #self.langch.setGeometry(QtCore.QRect(190, 150, 75, 31))
        #self.langch.setObjectName("langch")
        #self.langch.clicked.connect(self.chstate)
```

```
        self.langde = QtGui.QPushButton("DE")
        #self.langde.setGeometry(QtCore.QRect(280, 150, 75, 31))
        #self.langde.setObjectName("langde")
        #self.langde.clicked.connect(self.destate)

        self.langsp = QtGui.QPushButton("ES")
        #self.langsp.setGeometry(QtCore.QRect(370, 150, 75, 31))
        #self.langsp.setObjectName("langsp")
        #self.langsp.clicked.connect(self.spstate)

        self.selectLanguage = QtGui.QLabel("Select language")
        self.selectLanguage.setFont(QtGui.QFont("SansSerif", 20))
        self.selectLanguage.setAlignment(QtCore.Qt.AlignCenter)

        self.first_box.addWidget(self.selectLanguage)
        self.first_box.addWidget(self.langth)
        self.first_box.addWidget(self.langen)
        self.first_box.addWidget(self.langch)
        self.first_box.addWidget(self.langsp)
        self.first_box.addWidget(self.langde)

        self.connect(self.langth, QtCore.SIGNAL("clicked()"), self.thState)
        self.connect(self.langen, QtCore.SIGNAL("clicked()"), self.enState)
        self.connect(self.langch, QtCore.SIGNAL("clicked()"), self.chState)
        self.connect(self.langsp, QtCore.SIGNAL("clicked()"), self.spState)
        self.connect(self.langde, QtCore.SIGNAL("clicked()"), self.deState)

    def thState(self):
        global lang
```

```
        lang = "th"
        self.first_second_view()

    def enState(self):
        global lang
        lang = "en"
        self.first_second_view()

    def chState(self):
        global lang
        lang = "zh-cn"
        self.first_second_view()

    def spState(self):
        global lang
        lang = "es"
        self.first_second_view()

    def deState(self):
        global lang
        lang = "de"
        self.first_second_view()

    def first_second_view(self):
        self.remove_first_view()
        self.second_view()

    def second_view(self):
        self.next3 = QtGui.QPushButton("Sender")
```

```
        self.next4 = QtGui.QPushButton("Receiver")
        self.back1 = QtGui.QPushButton("Choose language")
        self.selectMode = QtGui.QLabel("Select mode")
        self.selectMode.setFont(QtGui.QFont("SansSerif", 20))
        self.selectMode.setAlignment(QtCore.Qt.AlignCenter)

        self.second_box.addWidget(self.selectMode)
        self.second_box.addWidget(self.next3)
        self.second_box.addWidget(self.next4)
        self.second_box.addWidget(self.back1)

        self.connect(self.next3, QtCore.SIGNAL("clicked()"), self.second_third_view)
        self.connect(self.next4, QtCore.SIGNAL("clicked()"), self.second_fourth_view)
        self.connect(self.back1, QtCore.SIGNAL("clicked()"), self.second_first_view)

    def second_first_view(self):
        self.remove_second_view()
        self.first_view()

    def second_third_view(self):
        global mode
        mode = "send"
        self.remove_second_view()
        worker.connect(self)

    def second_fourth_view(self):
        global mode
        mode = "recieve"
        self.remove_second_view()
```

```
        worker.connect(self)

    def third_view(self):

        self.back2 = QtGui.QPushButton("Choose mode")
        self.listen = QtGui.QLabel()
        self.listen.setText("You said ...")
        self.speech = QtGui.QTextBrowser()

        self.third_box.addWidget(self.back2)
        self.third_box.addWidget(self.listen)
        self.third_box.addWidget(self.speech)
        self.connect(self.back2, QtCore.SIGNAL("clicked()"), self.third_second_view)

        self.speechText = sendThread(self)
        self.connect(self.speechText, QtCore.SIGNAL("ACTIVATED (QString)"),
self.ACTIVATED)
        self.speechText.start()
        # translate.speechText(self)

    def ACTIVATED(self, text):
        self.speech.setText(text)
        self.speech.setFont(QtGui.QFont("SansSerif", 18))

    def third_second_view(self):
        self.remove_third_view()
        self.second_view()
        global mode
        mode = " "
```

```
    #background_thread.stop()

def fourth_view(self):
    self.remove_second_view()
    self.back2 = QtGui.QPushButton("Choose mode")
    self.textTranslate = QtGui.QTextBrowser()
    self.textLabel = QtGui.QLabel("Message...")



    self.fourth_box.addWidget(self.back2)
    self.fourth_box.addWidget(self.textLabel)
    self.fourth_box.addWidget(self.textTranslate)

    self.connect(self.back2, QtCore.SIGNAL("clicked()"), self.fourth_second_view)

    self.receiveText = receiveThread(self)
    self.connect(self.receiveText, QtCore.SIGNAL("receive (QString)"), self.receive)
    self.receiveText.start()

def receive(self, word):
    self.textTranslate.setText(word)
    self.textTranslate.setFont(QtGui.QFont("SansSerif", 18))

def fourth_second_view(self):
    self.remove_fourth_view()
    self.second_view()
    global mode
    mode = " "
```

```
def remove_first_view(self):
    for cnt in reversed(range(self.first_box.count())):
        # takeAt does both the jobs of itemAt and removeWidget
        # namely it removes an item and returns it
        widget = self.first_box.takeAt(cnt).widget()
        if widget is not None:
            # widget will be None if the item is a layout
            widget.deleteLater()

def remove_second_view(self):
    for cnt in reversed(range(self.second_box.count())):
        # takeAt does both the jobs of itemAt and removeWidget
        # namely it removes an item and returns it
        widget = self.second_box.takeAt(cnt).widget()

        if widget is not None:
            # widget will be None if the item is a layout
            widget.deleteLater()

def remove_third_view(self):
    for cnt in reversed(range(self.third_box.count())):
        # takeAt does both the jobs of itemAt and removeWidget
        # namely it removes an item and returns it
        widget = self.third_box.takeAt(cnt).widget()

        if widget is not None:
            # widget will be None if the item is a layout
            widget.deleteLater()
```

```
    def remove_fourth_view(self):
        for cnt in reversed(range(self.fourth_box.count())):
            # takeAt does both the jobs of itemAt and removeWidget
            # namely it removes an item and returns it
            widget = self.fourth_box.takeAt(cnt).widget()

            if widget is not None:
                # widget will be None if the item is a layout
                widget.deleteLater()

    def checkItems(self):
        QtGui.QMessageBox.information(self, 'Count', "You have %s Items in Layout" %
self.dvbox.count(), QtGui.QMessageBox.Ok)


class worker():
    def connect(self):
        host = '161.246.105.214'
        port = 9009
        global s
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.settimeout(2)

        # connect to remote host
        c = 0
        for c in range(1):
            try:
                s.connect((host, port))
                c = 1
```

```
            except:
                print('Unable to connect')
                # sys.exit()
        if mode == "send":
            LayoutTest.third_view(self)
        elif mode == "recieve":
            LayoutTest.fourth_view(self)


class sendThread(QtCore.QThread):
    def __init__(self, parent=None):
        QtCore.QThread.__init__(self, parent)
        self.text = ''

    def run(self):
        while mode == "send":
            socket_list = [s]
            # Get the list sockets which are readable
            read_sockets, write_sockets, error_sockets = select.select(socket_list, [], [],
0)
            # lang = input('Your language    : ')
            global r
            r = sr.Recognizer()
            r.energy_threshold = 1000
            with sr.Microphone(sample_rate=44100) as source:
                #print("listening..")
                global audio
                audio = r.listen(source)
                #print("processing")
```

```
            background_thread = Thread(target=self.speech_texts, args=(audio, ))
            background_thread.start()

    def speech_texts(self, audio):
        try:
            self.text = r.recognize_google(audio, language=lang)
            #print("You said       :"+" \" "+text+" \"")
            self.emit(QtCore.SIGNAL("ACTIVATED (QString)"), self.text)
            msg = self.text
            msgs = '" ' + lang + ' : ' + msg + ' '
            s.send(msgs.encode('utf-8'))
        except sr.UnknownValueError:
            print("Google Speech Recognition could not understand audio")
        except sr.RequestError:
            print("Could not understand audio ")

class receiveThread(QtCore.QThread):
    def __init__(self, parent = None):
        QtCore.QThread.__init__(self, parent)
        self.word = " "

    def run(self):
        while mode == "recieve":
            #print("a")
            socket_list = [s]
            # Get the list sockets which are readable
            read_sockets, write_sockets, error_sockets = select.select(socket_list, [], [],
0)
```

```
#print("b")
for sock in read_sockets:
    # incoming message from remote server, s
    data = sock.recv(4096)
    #print(data)
    mixer.init()
    while mixer.music.get_busy():
        time.sleep(0.02)
    data = data.decode('utf-8')
    if data == '/n':
        break
    else:
        #print(data)
        lang1 = data.split('" ')[1].split(' :')[0]
        self.word = data.split(": ")[1].split(' /n')[0]
        if lang1 == lang:
            #print('Translate  text to  : ' + "\"" + text + "\"")
            self.emit(QtCore.SIGNAL("receive (QString)"), self.word)
            tts = gTTS(text=self.word, lang=lang)
            tts.save('/home/puripoom/Project/speech.mp3')
            mixer.music.load('/home/puripoom/Project/speech.mp3')
            mixer.music.play()
        else:
            # print (lang1)
            # print (text)
            self.word = quote(self.word)
            url =
'https://www.googleapis.com/language/translate/v2?key=AIzaSyB46al-
```

```
HVvYK0bGpNNYwTIcJ0OdPu8IT0c&q=' + self.word + '&source=' + lang1 + '&target='
+ lang
                        html = urlopen(url)
                        result = html.read()
                        result = result.decode()
                        self.word = result.split(': "')[1].split('"\n')[0]
                        #print('Translate  text to  : ' + "\"" + word + "\"")
                        self.emit(QtCore.SIGNAL("receive (QString)"), self.word)
                        tts = gTTS(text=self.word, lang=lang)
                        tts.save('/home/puripoom/Project/speech.mp3')
                        mixer.music.load('/home/puripoom/Project/speech.mp3')
                        mixer.music.play()


def run():

        app = QtGui.QApplication(sys.argv)
        ex = LayoutTest()
        ex.show()
        sys.exit(app.exec_())

if __name__ == "__main__":
        run()
```

ชุดคำสั่งของด้านเซิร์ฟเวอร์

```
# chat_server.py

import sys, socket, select

HOST = '192.168.1.4'
SOCKET_LIST = []
RECV_BUFFER = 4096
PORT = 9009

def chat_server():

    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    server_socket.bind((HOST, PORT))
    server_socket.listen(10)

    # add server socket object to the list of readable connections
    SOCKET_LIST.append(server_socket)

    print "Chat server started on port " + str(PORT)

    while 1:

        # get the list sockets which are ready to be read through select
        # 4th arg, time_out  = 0 : poll and never block
        ready_to_read,ready_to_write,in_error = select.select(SOCKET_LIST,[],[],0)

        for sock in ready_to_read:
```

```
            # a new connection request recieved
            if sock == server_socket:
                sockfd, addr = server_socket.accept()
                SOCKET_LIST.append(sockfd)
                print "Client (%s, %s) connected" % addr

                #broadcast(server_socket, sockfd, "[%s:%s] entered our chatting room\n"
% addr)

            # a message from a client, not a new connection
            else:
                # process data recieved from client,
                try:
                    # receiving data from the socket.
                    data = sock.recv(RECV_BUFFER)
                    print ("[%s:%s]" %addr + ' ' + data)
                    if data:
                        # there is something in the socket
                        broadcast(server_socket, sock, "\r" + '[' + str(sock.getpeername()) +
'] ' + data)
                    else:
                        # remove the socket that's broken
                        if sock in SOCKET_LIST:
                            SOCKET_LIST.remove(sock)

                        # at this stage, no data means probably the connection has been
broken
                        #broadcast(server_socket, sock, "Client (%s, %s) is offline\n" %
addr)
```

```
                # exception
                except:
                    #broadcast(server_socket, sock, "Client (%s, %s) is offline\n" % addr)
                    continue

    server_socket.close()

# broadcast chat messages to all connected clients
def broadcast (server_socket, sock, message):
    for socket in SOCKET_LIST:
        # send the message only to peer
        if socket != server_socket and socket != sock :
            try :
                socket.send(message)
            except :
                # broken socket connection
                socket.close()
                # broken socket, remove it
                if socket in SOCKET_LIST:
                    SOCKET_LIST.remove(socket)

if __name__ == "__main__":

    sys.exit(chat_server())
```