เครื่องรับวิทยุดิจิทัลกรณีเกิดเหตุฉุกเฉิน

DIGITAL RADIO RECEIVER IN CASE OF EMERGENCY

โดย

นายภูวกฤต รัตนะมงคลกุล

นายสารัช พรรณเชษฐ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2566

เครื่องรับวิทยุดิจิทัลกรณีเกิดเหตุฉุกเฉิน

DIGITAL RADIO RECEIVER IN CASE OF EMERGENCY

โดย

นายภูวกฤต รัตนะมงคลกุล 63010772

นายสารัช พรรณเชษฐ์ 63010963

อาจารย์ที่ปรึกษา

ผศ. ดร.นภัทร สระเอี่ยม

ผศ. ดร.สมปอง วิเศษพานิชกิจ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2566

ปริญญานิพนธ์ปีการศึกษา 2566
ภาควิชาวิศวกรรมโทรคมนาคม
คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
เรื่อง เครื่องรับวิทยุดิจิทัลกรณีเกิดเหตุฉุกเฉิน
DIGITAL RADIO RECEIVER IN CASE OF EMERGENCY

ผู้จัดทำ
1. นายภูวกฤต รัตนะมงคลกุล 63010772
2. นายสารัช พรรณเชษฐ์ 63010963

..... **อาจารย์ที่ปรึกษา**
(ผศ. ดร.นภัทร สระเอี่ยม)

..... **อาจารย์ที่ปรึกษาร่วม**
(ผศ. ดร.สมปอง วิเศษพานิชกิจ)

กิตติกรรมประกาศ

การดำเนินปริญญานิพนธ์เรื่อง “เครื่องรับวิทยุดิจิทัลกรณีเกิดเหตุฉุกเฉิน” จะไม่สามารถสำเร็จลุล่วงไปได้ด้วยดี หากไม่ได้รับความช่วยเหลือ และความอนุเคราะห์อย่างดียิ่งจากอาจารย์ที่ปรึกษาปริญญานิพนธ์ คือ ผศ. ดร.นภัทร สระเอี่ยม และ ผศ. ดร.สมปอง วิเศษพานิชกิจ อาจารย์ที่ปรึกษาร่วม ที่กรุณาให้คำสั่งสอน และแนวทางการแก้ไขปัญหาตลอดระยะเวลาในการจัดทำปริญญานิพนธ์นี้ รวมทั้งสนับสนุนสถานที่ เครื่องมือ และอุปกรณ์ต่างๆที่จำเป็นต้องใช้ในระหว่างการจัดทำปริญญานิพนธ์ ขอขอบพระคุณท่านในความห่วงใย และความหวังดีที่ให้แก่คณะผู้จัดทำเป็นอย่างยิ่ง

ขอขอบคุณท่านอาจารย์ประจำวิชาภาควิชาวิศวกรรมโทรคมนาคม คณะวิศวกรรมศาสตร์สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังทุกท่าน ที่ได้อบรมสั่งสอนและประสิทธิ์ประสาทวิชาความรู้ให้แก่คณะผู้จัดทำ

ขอขอบคุณผู้มีส่วนเกี่ยวข้องทุกท่าน อาทิ บิดา มารดา และเพื่อนนักศึกษา ที่คอยสนับสนุนแนะนำช่วยเหลือ และให้กำลังใจแก่คณะผู้จัดทำเสมอมา จนกระทั่งปริญญานิพนธ์สำเร็จลุล่วงไปได้ด้วยดี

นายภูวกฤต รัตนะมงคลกุล
นายสารัช พรรณเชษฐ์
ผู้จัดทำ

เครื่องรับวิทยุดิจิทัลกรณีเกิดเหตุฉุกเฉิน
DIGITAL RADIO RECEIVER IN CASE OF EMERGENCY

โดย นายภูวกฤต รัตนะมงคลกุล 63010772
นายสารัช พรรณเชษฐ์ 63010963

อาจารย์ที่ปรึกษา ผศ. ดร นภัทร สระเอี่ยม
อาจารย์ที่ปรึกษาร่วม ผศ. ดร.สมปอง วิเศษพานิชกิจ

บทคัดย่อ

ภัยพิบัติเป็นหนึ่งในสิ่งที่พรากชีวิตประชากรบนโลกมากที่สุด และมีแนวโน้มจะเกิดขึ้นในทุกๆ ปี ดังนั้นการสื่อสารในช่วงเวลาภัยพิบัติสถานการณ์ที่วิกฤติจึงมีบทบาทสำคัญในการเพิ่มโอกาสการรอดชีวิต ในปัจจุบันวิทยุที่ออกอากาศในประเทศไทย (AM และ FM) มีคุณภาพปานกลางเสียงรบกวนมาก เพื่อเพิ่มประสิทธิภาพการสื่อสารในช่วงเวลาภัยพิบัติให้สามารถสื่อสารได้อย่างแม่นยำยิ่งขึ้นคณะผู้จัดทำจึงเล็งเห็นความสำคัญของวิทยุดิจิทัลจึงได้นำมาออกแบบอุปกรณ์รับสัญญาณมีการออกแบบฟังก์ชันการใช้งานให้สลับสถานีไปยังช่องทางฉุกเฉิน และติดต่อสื่อสาร และแสดงผลผ่านโทรศัพท์เคลื่อนที่ได้

ABSTRACT

Disasters are one of the most life-threatening things on Earth and tend to occur every year. Therefore, communication during disaster incidents and critical situations plays a significant part in increasing the chances of survival. Radio broadcast in Thailand (AM and FM) has only usable conditions and lots of noise. To upgrade communication efficiency during disaster times to be more accurate, this project realized the importance of digital radio and build a receiver device with a design function that can switch stations to emergency channels and communicate and display via mobile phone.

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

หน้า

รูปที่

สารบัญรูป (ต่อ)

สารบัญตาราง

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในยามภัยพิบัติการได้รับข่าวสารข้อมูลคำแนะนำว่าควร และไม่ควรทำอย่างไร เป็นสิ่งที่ทางคณะผู้จัดทำให้ความเห็นว่ามีความสำคัญเป็นอย่างยิ่งยวดในการดูแลพื้นที่เสี่ยงต่อภัยพิบัติ และการเตรียมความพร้อมในสถานการณ์ภัยพิบัติที่ค่อนข้างซับซ้อน และยากลำบากที่ต้องอาศัยการสื่อสารที่มีประสิทธิภาพ รวดเร็ว และเชื่อถือได้ระหว่างหน่วยงานที่กำกับดูแลผู้ประสบภัย ในเวลาที่เกิดภัยพิบัติขึ้นการมีเครื่องมือที่สามารถทำงานได้โดยผู้ประสบภัยสามารถรับสารได้อย่างมีประสิทธิภาพพึ่งพาได้แม้ระบบสื่อสารส่วนใหญ่จะล่ม หรือเกิดความล่าช้า การกระจายเสียงด้วยวิทยุเป็นหนึ่งตัวเลือกที่ไว้ใจ และพึ่งพาได้แต่วิทยุในปัจจุบันในประเทศไทย (FM และ AM) นั้นประสิทธิภาพความละเอียดของสัญญาณต่ำ จึงอาจได้รับสารที่ไม่ครบถ้วน หรือผิดพลาด วิทยุดิจิทัลจึงเป็นตัวเลือกเทคโนโลยียุคใหม่ที่ควรเข้ามามีบทบาทสำคัญในบริบทนี้ ที่มาของโครงงานนี้อาศัยแนวคิด และความต้องการในการพัฒนาเทคโนโลยีวิทยุดิจิทัลในขอบเขตของการตอบสนองต่อภัยพิบัติ

1.2 วัตถุประสงค์

1) เพื่อให้วิทยุดิจิทัลใช้งานได้แม้ในสภาวะที่ไม่เอื้ออำนวยซึ่งช่องทางการสื่อสารแบบดั้งเดิมอาจล้มเหลวในกรณีที่เกิดเหตุฉุกเฉิน
2) เพื่อพัฒนาวิทยุดิจิทัลให้เป็นมิตรกับผู้ใช้มากขึ้น และสามารถใช้งานได้ง่าย

1.3 ขอบเขตของปริญญานิพนธ์

1) ทำการสร้างออกแบบให้วิทยุรับสัญญาณได้จริง
2) ทำให้วิทยุดิจิทัลสามารถที่จะทำการเข้าถึงได้ด้วยโทรศัพท์เคลื่อนที่
3) ในสถานการณ์ฉุกเฉินสามารถเปลี่ยนไปเป็นช่องสัญญาณเฉพาะได้โดยอัตโนมัติ

บทที่ 2

ทฤษฎีและหลักการที่เกี่ยวข้อง

ปริญญานิพนธ์เรื่อง “เครื่องรับวิทยุดิจิทัลกรณีเกิดเหตุฉุกเฉิน” มีทฤษฎี และหลักการที่สำคัญที่เกี่ยวข้องดังต่อไปนี้

2.1 เทคโนโลยีวิทยุกระจายเสียงระบบดิจิทัล Digital Audio Broadcasting Plus (DAB+)

2.1.1 ความเป็นมาของเทคโนโลยี DAB+

ที่มาเกิดจากการจัดตั้ง Eureka Project 147 ขึ้นในปีค.ศ. 1987 จุดประสงค์โครงการเพื่อพัฒนาระบบออกอากาศที่ส่งได้ทั้งเสียง และข้อมูล โดยรองรับเครื่องรับทั้งแบบอยู่กับที่ (fixed), แบบพกพา (portable) และแบบเคลื่อนที่ (mobile) จึงเกิดเป็นเทคโนโลยี DAB (Digital Audio Broadcasting) เพื่อทดแทนวิทยุกระจายเสียงระบบ AM และ FM ที่มีการใช้งานมาเป็นเวลานาน โดยที่จะมีคุณสมบัติที่เหนือกว่าระบบแอนะล็อกเดิม เช่น

1. ประสิทธิภาพการใช้งานความถี่ : DAB ช่วยลดปัญหาที่เกี่ยวข้องกับความถี่ เนื่องจากมีการใช้งานความถี่ที่มีประสิทธิภาพสูงกว่า การใช้การถอดรหัสดิจิทัลช่วยลดการแสดงผลที่เป็นผลมาจากการขาดหายของสัญญาณ และการรบกวน

2. คุณภาพของเสียง : DAB มีคุณภาพเสียงที่ดีมากกว่า AM และ FM โดยทั่วไป การใช้การถอดรหัสเสียงดิจิทัล (Digital Audio) ช่วยลด หรือป้องกันปัญหาที่เกี่ยวกับสัญญาณที่มีประสิทธิภาพต่ำ หรือสะท้อน

3. บริการเสริมต่าง ๆ ที่ระบบเดิมไม่มี : DAB สามารถให้บริการข้อมูลเสริม (data services) ที่ไม่สามารถทำได้ในระบบ AM และ FM. นอกจากการส่งเสียง DAB ยังสามารถส่งข้อมูล เช่น ชื่อศิลปิน, ชื่อเพลง, ข้อมูลการจราจร, ข่าวสาร, และอื่น ๆ ที่เกี่ยวข้อง

ผลที่ได้จากการพัฒนา จึงเกิดเป็นเทคโนโลยี “DAB” ที่ได้รับการแนะนำโดย ITU ให้เป็นเทคโนโลยีวิทยุกระจายเสียงสำหรับการส่งภาคพื้นดิน และการส่งผ่านดาวเทียม

การประมวลผลสัญญาณของ DAB ก่อนที่จะทำการออกอากาศมีหลายขั้นตอน ขั้นตอนหนึ่งคือการแปลงสัญญาณจากแอนะล็อกไปเป็นดิจิทัลพร้อมบีบอัดซึ่ง DAB เลือกใช้มาตรฐาน ISO/IEC 11172-3 และ ISO/IEC 13818-3 (เรียกว่า “MPEG Audio Layer II”) การใช้ทั้งสองมาตรฐานนี้ช่วยให้ DAB สามารถเข้ากันได้กับอุปกรณ์ที่ใช้ทั้ง MPEG-1 Audio และ MPEG-2 Audio ทำให้มีความยืดหยุ่น และเข้ากันได้กับสภาพแวดล้อมที่หลากหลายขึ้น

1. ISO/IEC 11172-3 : รองรับการใช้ bitrate ตั้งแต่ 32 kbps ถึง 320 kbps
2. ISO/IEC 13818-3 : รองรับการใช้ bitrate ตั้งแต่ 8 kbps ถึง 320 kbps

DAB+ ก็คือ DAB ที่ถูกพัฒนาไปอีกระดับในส่วนของเทคโนโลยีการแปลง และบีบอัดสัญญาณดิจิทัล จากเดิมที่ใช้ ISO/IEC 11172-3 [1] และ ISO/IEC 13818-3 [2] ไปใช้เป็น MPEG-4 HE AAC v2 [3] โดยมีข้อดีหลัก ๆ ดังนี้

1. DAB จะ Multiplex โดยใช้การบีบอัดแบบ OFDM (Orthogonal Frequency Division Multiplexing) ซึ่งสามารถรองรับหลายสถานีในช่องสัญญาณเดียว แต่จำกัดในการปรับปรุงคุณภาพเสียง และการรองรับข้อมูล ส่วน DAB+ ยังใช้การบีบอัดแบบ OFDM ในการจัด Multiplex แต่มีการปรับปรุงการใช้แบนด์วิดท์ และการรองรับที่ดีขึ้น สามารถรองรับจำนวนสถานีที่มากยิ่งขึ้นใน Multiplex เดียว ซึ่งสามารถเป็นไปได้ถึง 3 เท่าของจำนวนที่ DAB สามารถรองรับได้

2. Transmission cost จะต่ำกว่ากรณีส่งระบบ DAB เนื่องจาก DAB+ ใช้รูปแบบการบีบอัดที่ทันสมัย และมีประสิทธิภาพสูง ช่วยลดขนาดข้อมูลที่ต้องถูกส่งไป ช่วยลดการใช้แบนด์วิดท์ และทำให้สามารถรองรับการถ่ายทอดสัญญาณวิทยุดิจิทัลได้มากขึ้นในช่องสัญญาณเดียว

3. เครื่องรับที่รองรับ DAB+ มีดังนี้

3.1. Backward Compatible : เครื่องรับที่รองรับ DAB+ และ backward compatible สามารถรับสัญญาณ DAB และ DAB+ ได้ ผู้ใช้จึงสามารถใช้เครื่องรับนี้ในพื้นที่ที่มีการกระจายสัญญาณ DAB หรือ DAB+ ได้

3.2. Feature Scrolling Text : คือ ความสามารถในการแสดงข้อความที่เลื่อนไปข้างหน้า หรือข้างหลังบนหน้าจอของเครื่องรับ โดยเป็นลักษณะที่บ่งบอกถึงการแสดงข้อมูลเพิ่มเติมเช่น ชื่อศิลปิน, ชื่อเพลง, หรือข้อมูลอื่น ๆ ที่เกี่ยวข้อง

3.3. Multimedia Service : คือ ความสามารถในการรองรับบริการที่มีเนื้อหาหลากหลายมากกว่าการกระจายสัญญาณเสียงแบบธรรมดา นอกจากเสียงแล้วยังสามารถรองรับรูปภาพ, ข้อความ, หรือแม้กระทั่งวิดีโอได้

การรองรับ backward compatible และ feature scrolling text และ multimedia service ทำให้ผู้ใช้สามารถใช้งานทั้ง DAB และ DAB+ ได้โดยไม่ต้องเปลี่ยนเครื่องรับเมื่อมีการอัพเกรดหรือการปรับปรุงในการกระจายสัญญาณ

4. การออกอากาศจะให้ความชัดเจนของสัญญาณมากกว่า DAB เนื่องจากรูปแบบการบีบอัดช่วยลดขนาดข้อมูลที่ต้องถูกส่งไป ทำให้สามารถใช้แบนด์วิดท์ได้มีประสิทธิภาพ และส่งสัญญาณที่มีคุณภาพสูง รวมถึงมีความยืดหยุ่นในการรองรับจำนวนสถานีที่มากขึ้นในช่องสัญญาณเดียว ทำให้ DAB+ สามารถให้บริการในพื้นที่ที่มีจำนวนสถานีมากกว่า ทำให้ผู้ใช้สามารถเข้าถึงความหลากหลายของสถานีได้มากขึ้น

5. เครื่องรับจะมี Zapping delay ที่ต่ำ เนื่องจาก

5.1. การใช้เทคโนโลยีการส่งข้อมูลที่มีประสิทธิภาพสูง : DAB+ ใช้เทคโนโลยีการส่งข้อมูลที่ทันสมัย และมีประสิทธิภาพสูง ซึ่งช่วยลดเวลาที่จำเป็นในการสลับสถานี

5.2. การใช้หลายช่องสัญญาณ (Multiplexing) : DAB+ มีความสามารถในการส่งข้อมูลจากหลายสถานีไปยังเครื่องรับใน Multiplex เดียวกัน ซึ่งช่วยลดเวลาในการสลับสถานีเมื่อผู้ฟังต้องการเปลี่ยนสถานี

5.3. การใช้โครงข่ายที่มีประสิทธิภาพสูง : การใช้โครงข่ายที่มีความเร็ว และประสิทธิภาพสูงช่วยลดเวลาในการสลับสถานี

6. ทำให้การส่ง MPEG Surround (เป็นมาตรฐานการบีบอัดเสียงที่ถูกพัฒนาโดย Moving Picture Experts Group เพื่อให้สามารถสร้างเสียงที่มีความพิเศษ และมีมิติมากขึ้นในการรับฟัง) มีการใช้ bitrate ต่ำกว่าการส่งด้วย DAB

2.1.2 คุณสมบัติ และ จุดเด่นของ DAB+

1. คุณภาพเสียงดีกว่าระบบแอนะล็อก เนื่องจาก DAB+ ใช้รูปแบบการบีบอัดเสียงที่ทันสมัย และมีประสิทธิภาพสูง (HE-AAC v2) จึงช่วยลด bitrate ที่ต้องใช้ในการถ่ายทอดเสียงทำให้มีคุณภาพเสียงที่ดี ใช้แบนด์วิดท์ลดลง และมีความสามารถในการต้านสัญญาณรบกวน

2. บริการข้อมูลเสริม ซึ่งไม่มีในระบบแอนะล็อก เช่น ข้อมูลการจราจร, ข้อมูลการเดินทาง, EPG (Electronic Programming Guide), ข้อมูลสภาพอากาศ และ Broadcast Website เป็นต้น

3. บริการข้อมูลประเภท audio-related data เช่น ชื่อเพลง ชื่อคนแต่งเพลง แสดงบนเครื่องรับชนิดมีหน้าจอแสดงผล

4. การฟังในรูปแบบ Mobile reception เช่น การรับฟังรายการขณะอยู่บนรถที่กำลังเคลื่อนที่ยังสามารถรับฟังรายการนั้นได้ตลอดเส้นทางโดยที่ไม่ต้องมีการเปลี่ยนความถี่รับ

2.1.3 ประเทศไหนใช้ DAB+ บ้าง

อ้างอิงจาก worlddab.org (ข้อมูล ณ วันที่ 17 พฤษภาคม 2561) มีการใช้งาน DAB/DAB+ แล้วกว่า 24 ประเทศ เช่น ออสเตรเลีย เบลเยียม เยอรมัน สเปน เกาหลีใต้ และอยู่ในระหว่างการทดลองอีกกว่า 27 ประเทศ เช่น นิวซีแลนด์ อินโดนีเซีย และประเทศไทย

2.1.4 หลักการทำงานของเทคโนโลยี DAB+

ในส่วนของหลักการทำงานของภาคส่งเป็นไปตามรูปที่ 2.1 [4] โดยการประมวลผลสัญญาณตั้งต้น (audio และ data) ตั้งแต่ตอนเป็นบิตข้อมูลดิจิทัลผ่านขั้นตอนต่าง ๆ จนกระทั่งเกิดเป็นสัญญาณคลื่นวิทยุที่พร้อมส่งออกอากาศ โดยในส่วนของเครื่องรับการทำงานก็จะเป็นเหมือนภาคส่งแต่จะเป็นขั้นตอนที่ย้อนกลับ ถ้าจะแบ่งการทำงานของภาคส่งออกเป็น function ส่วนใหญ่จะแบ่งออกได้ดังนี้

1. การบีบอัด (compress) สัญญาณเสียงด้วยมาตรฐานที่กำหนด (DAB+ ใช้ MPEG-4 HEAAC v2)

2. การทำ Conditional Access (CA) เพื่อรักษาความปลอดภัย, ควบคุมการเข้าถึง, และการจัดการสิทธิ์ของเนื้อหาที่ถูกบริการผ่านระบบวิทยุดิจิทัล

3. การเพิ่มความแข็งแรงทนทานต่อสัญญาณรบกวน (Robustness) ของสัญญาณที่จะส่งออกไป (เรียกว่าการทำ “Channel Encoding”) ผ่านทางภาคต่าง ๆ คือ

3.1. Energy dispersal scrambler : ทำงานโดยการสลับ และกระจายพลังงานของสัญญาณข้อมูลที่จะถูกส่งไป ข้อมูลที่ส่งไปมักจะมีลักษณะที่มีความกลมกลืน หรือความเป็นรูปภาพจากสัญญาณที่ส่งไปติดต่อกัน

3.2. Convolutional encoder : ใช้ในการเข้ารหัสข้อมูลที่จะถูกส่งไป เพื่อเพิ่มความปลอดภัยในการถ่ายทอดข้อมูล มักใช้ในการเพิ่ม error ในการถ่ายทอดข้อมูลทำให้มีการแก้ไขข้อผิดพลาดได้

3.3. Time interleaver : เป็นขั้นตอนที่ใช้ในการสลับลำดับข้อมูลในเวลาที่ทำให้ข้อมูลที่ถูกส่งไปไม่ต่อเนื่องในช่วงเวลาสั้น ๆ มักใช้ในการลดผลกระทบจากข้อผิดพลาดที่เกิดขึ้นติดต่อกัน

4. การทำ Multiplexing เพื่อทำการส่งออกเป็นแบบสตรีมข้อมูล

5. การทำให้เป็นสัญญาณ COFDM (Coded Orthogonal Frequency Division Multiplex)

มีหลักการทำงานดังนี้

1. แบ่งความถี่ : COFDM แบ่งความถี่ที่ใช้ในการถ่ายทอดข้อมูลเป็นช่องย่อย ๆ ที่มีความถี่หลาย ๆ ความถี่ โดยความถี่แต่ละช่องย่อยนั้นถูกนำมาใช้ในการส่งข้อมูลพร้อมกัน ทำให้มีการถ่ายทอดข้อมูลได้มีประสิทธิภาพ

2. การใช้เทคนิค Orthogonal Frequency Division : ซึ่งทำให้สัญญาณในแต่ละช่องย่อยมีความไม่มีอิทธิพลต่อกัน ช่วยลดปัญหา Interference ที่เกิดจากสัญญาณที่ส่งไปติดต่อกัน

3. การใช้การถอดรหัส และการแก้ไขข้อผิดพลาด : COFDM มีความสามารถในการแก้ไขข้อผิดพลาดจากสัญญาณข้อมูลที่ถูกส่งไปโดยใช้การถอดรหัส และเทคนิคการแก้ไขข้อผิดพลาดช่วยให้ COFDM มีความยืดหยุ่นในการรับสัญญาณที่มีข้อผิดพลาด

4. มีประสิทธิภาพดีในการรับสัญญาณในสภาพแวดล้อมที่มีข้อบกพร่อง

ข้อดีของ COFDM มีดังนี้

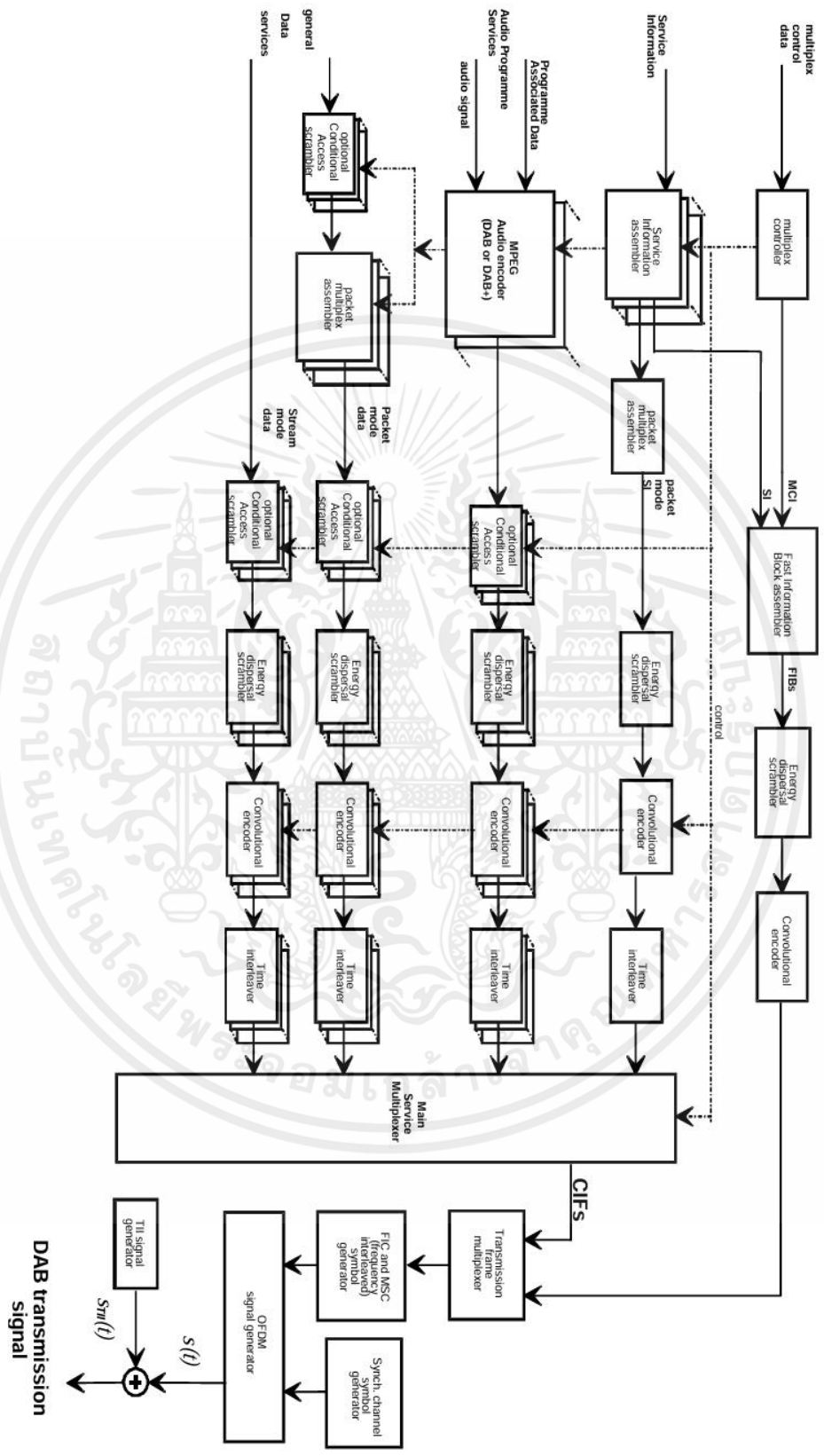
1. มีความยืดหยุ่นในการใช้งานทางวิทยุที่ดี ทำให้เหมาะสำหรับการใช้ในสภาพแวดล้อมมีข้อบกพร่อง

2. ทำให้สามารถมีการถ่ายทอดข้อมูลในรูปแบบความถี่ที่กว้าง

3. ช่วยลดปัญหา Interference จากสัญญาณที่ส่งไปติดต่อกัน

4. มีความสามารถในการถอดรหัส และแก้ไขข้อผิดพลาดจากสัญญาณข้อมูลที่ถูกส่ง

6. การแปลงให้เป็นสัญญาณในย่านความถี่วิทยุ (RF) ขยายกำลังส่ง และพร้อมส่งออกอากาศ



รูปที่ 2.1 บล็อกไดอะแกรมระบบส่งของวิทยุกระจายเสียงระบบดิจิทัล DAB

2.1.5 ช่องสัญญาณในระบบออกอากาศ DAB+

การทำงานของเทคโนโลยี DAB+ ในระบบส่งจะแบ่งการประมวลผล และส่งผ่านข้อมูลผ่านทาง 3 ช่องทาง คือ

1. Main Service Channel (MSC) : เป็นช่องทางสำหรับบรรทุก digital audio service ต่าง ๆ รวมถึง data service

2. Fast Information Channel (FIC) : บรรทุกข้อมูลที่จำเป็นสำหรับเครื่องรับปลายทางเพื่อใช้ในการประมวลผล ถอดรหัสสัญญาณที่รับได้ และแสดงผล service ออกมาให้ผู้รับชม/ฟังได้อย่างถูกต้อง ตัวอย่างของข้อมูลที่ถูกส่งไปใน FIC เช่น Multiplex Configuration Information (MCI) ที่จะบอกเครื่องรับว่า multiplexed stream ที่รับได้นั้นใช้รูปแบบการป้องกันความผิดพลาดของข้อมูล (Error Protection) แบบใด หรือจะเป็น service label ที่จะแสดงผลบนหน้าจอของเครื่องรับ เป็นต้น

3. Synchronization Channel (SC) : จะบรรจุข้อมูลสำหรับเครื่องรับเพื่อใช้ในการทำ synchronization เช่น สัญญาณ phase reference

2.1.6 ย่านความถี่ใช้งาน

เทคโนโลยี DAB+ สามารถส่งออกอากาศได้ในย่านความถี่ อ้างอิงจากตาราง 2.1 ดังนี้

1. ย่าน VHF Band I : ความถี่ 47 MHz ถึง 88 MHz
2. ย่าน VHF Band II : ความถี่ 87.5 MHz ถึง 108 MHz
3. ย่าน VHF Band III : ความถี่ 174 MHz ถึง 230 MHz (ย่านความถี่ที่ส่วนใหญ่ถูกใช้)
4. ย่าน UHF Band IV : ความถี่ 470 MHz ถึง 582 MHz
5. ย่าน UHF Band V : ความถี่ 582 MHz ถึง 806 MHz
6. ย่าน L-Band : ความถี่ 1,452 MHz ถึง 1,492 MHz
7. ความถี่ตั้งแต่ 3,000 MHz ลงมา

2.1.7 ลักษณะโครงข่ายส่งสัญญาณ

DAB+ รองรับการส่งได้ในโครงข่าย 2 รูปแบบ คือ

1. โครงข่ายแบบภาคพื้นดิน (Terrestrial) : เป็นระบบการสื่อสารที่ใช้สิ่งแวดล้อมบนพื้นดิน เช่น ทางถนน, ทางราง, หรือสัญญาณทางอากาศที่ถูกส่งไปผ่านเครือข่ายตรง ๆ จากสถานีส่งสัญญาณไปยังเครื่องรับที่ติดตั้งบนพื้นดิน ได้แก่ โทรทัศน์ดิจิทัลทางพื้นดิน (Digital Terrestrial Television), วิทยุดิจิทัล และเครือข่ายโทรศัพท์ที่ใช้ทางพื้นดิน

2. โครงข่ายสายเคเบิล (Cable) : เป็นระบบการสื่อสารที่ใช้สายส่งสัญญาณ ซึ่งสายเคเบิลนี้สามารถเป็นทางเดียว หรือทางคู่ และสามารถถูกติดตั้งในท่อ หรือดิน ได้แก่ ระบบโทรทัศน์เคเบิล (Cable TV), อินเทอร์เน็ตเคเบิล (Cable Internet) และบริการโทรศัพท์ที่ใช้สายเคเบิล

2.1.8 ความกว้างช่องสัญญาณ (Bandwidth) และผังการใช้ช่องความถี่

ช่องสัญญาณ (Bandwidth) ที่ออกอากาศภายใต้เทคโนโลยี DAB+ 1 ช่อง กว้าง 1.536 MHz รายละเอียดผังการใช้ช่องสัญญาณแสดงรูปที่ 2.2 – 2.5

T-DAB Block number	T-DAB Block label	Centre frequency (MHz)	Block corner frequencies (MHz)	Lower/upper guard distance (kHz)
Band I: 47,0 to 68,0 MHz				
(1)	2A	47,936	47,168 to 48,704	168/176
(2)	2B	49,648	48,880 to 50,416	176/176
(3)	2C	51,360	50,592 to 52,128	176/176
(4)	2D	53,072	52,304 to 53,840	176/320
(5)	3A	54,928	54,160 to 55,696	320/176
(6)	3B	56,640	55,872 to 57,408	176/176
(7)	3C	58,352	57,584 to 59,120	176/176
(8)	3D	60,064	59,296 to 60,832	176/336
(9)	4A	61,936	61,168 to 62,704	336/176
(10)	4B	63,648	62,880 to 64,416	176/176
(11)	4C	65,360	64,592 to 66,128	176/176
(12)	4D	67,072	66,304 to 67,840	176/160
Band III: 174,0 to 240,0 MHz				
13	5A	174,928	174,160 to 175,696	160/176
14	5B	176,640	175,872 to 177,408	176/176
15	5C	178,352	177,584 to 179,120	176/176
16	5D	180,064	179,296 to 180,832	176/336
17	6A	181,936	181,168 to 182,704	336/176
18	6B	183,648	182,880 to 184,416	176/176
19	6C	185,360	184,592 to 186,128	176/176
20	6D	187,072	186,304 to 187,840	176/320
21	7A	188,928	188,160 to 189,696	320/176
22	7B	190,640	189,872 to 191,408	176/176
23	7C	192,352	191,584 to 193,120	176/176
24	7D	194,064	193,296 to 194,832	176/336
25	8A	195,936	195,168 to 196,704	336/176
26	8B	197,648	196,880 to 198,416	176/176
27	8C	199,360	198,592 to 200,128	176/176
28	8D	201,072	200,304 to 201,840	176/320
29	9A	202,928	202,160 to 203,696	320/176
30	9B	204,640	203,872 to 205,408	176/176
31	9C	206,352	205,584 to 207,120	176/176
32	9D	208,064	207,296 to 208,832	176/336
33	10A	209,936	209,168 to 210,704	336/(176)
NOTE 1	10N	210,096	209,328 to 210,864	
34	10B	211,648	210,880 to 212,416	(176)/176
35	10C	213,360	212,592 to 214,128	176/176
36	10D	215,072	214,304 to 215,840	176/320
37	11A	216,928	216,160 to 217,696	320/(176)
NOTE 1	11N	217,088	216,320 to 217,856	
38	11B	218,640	217,872 to 219,408	(176)/176

รูปที่ 2.2 รายละเอียดช่องความถี่สำหรับใช้ออกอากาศระบบ DAB+ ภาคพื้นดินในยุโรป

T-DAB Block number	T-DAB Block label	Centre frequency (MHz)	Block corner frequencies (MHz)	Lower/upper guard distance (kHz)
39	11C	220,352	219,584 to 221,120	176/176
40	11D	222,064	221,296 to 222,832	176/336
41	12A	223,936	223,168 to 224,704	336/(176)
NOTE 1	12N	224,096	223,328 to 224,864	
42	12B	225,648	224,880 to 226,416	(176)/176
43	12C	227,360	226,592 to 228,128	176/176
44	12D	229,072	228,304 to 229,840	176/176
45	13A	230,784	230,016 to 231,552	176/176
46	13B	232,496	231,728 to 233,264	176/176
47	13C	234,208	233,440 to 234,976	176/32
48	13D	235,776	235,008 to 236,544	32/176
49	13E	237,488	236,720 to 238,256	176/176
50	13F	239,200	238,432 to 239,968	176/32
L-Band: 1452,0 to 1467,5 MHz				
51	LA	1452,960	1452,192 to 1453,728	192/176
52	LB	1454,672	1453,904 to 1455,440	176/176
53	LC	1456,384	1455,616 to 1457,152	176/176
54	LD	1458,096	1457,328 to 1458,864	176/176
55	LE	1459,808	1459,040 to 1460,576	176/176
56	LF	1461,520	1460,752 to 1462,288	176/176
57	LG	1463,232	1462,464 to 1464,000	176/176
58	LH	1464,944	1464,176 to 1465,712	176/176
59	LI	1466,656	1465,888 to 1467,424	176/–

รูปที่ 2.3 รายละเอียดช่องความถี่สำหรับใช้ออกอากาศระบบ DAB+ ภาคพื้นดินในยุโรป(ต่อ)

T-DAB Block number	T-DAB Block label	Centre frequency (MHz)	Block corner frequencies (MHz)
L-Band: 1452,0 to 1467,5 MHz			
60	LJ	1468,368	1476,600 to 1469,136
61	LK	1470,080	1469,312 to 1470,848

รูปที่ 2.4 รายละเอียดช่องความถี่สำหรับใช้ออกอากาศระบบ DAB+ ผ่านดาวเทียมในยุโรป

T-DAB Block number	T-DAB Block label	Centre frequency (MHz)	Block corner frequencies (MHz)
62	LL	1471,792	1471,024 to 1472,560
63	LM	1473,504	1472,736 to 1474,272
64	LN	1475,216	1474,448 to 1475,984
65	LO	1476,928	1476,160 to 1477,696
66	LP	1478,640	1477,872 to 1479,408
67	LQ	1480,352	1479,584 to 1481,120
68	LR	1482,064	1481,296 to 1482,832
69	LS	1483,776	1483,008 to 1484,544
70	LT	1485,488	1484,720 to 1486,256
71	LU	1487,200	1486,432 to 1487,968
72	LV	1488,912	1488,144 to 1489,680
73	LW	1490,624	1489,856 to 1491,392

รูปที่ 2.5 รายละเอียดช่องความถี่สำหรับใช้ออกอากาศระบบ DAB+ ผ่านดาวเทียมในยุโรป (ต่อ)

2.1.9 รูปแบบการส่ง (Transmission Mode ; TM) ของเทคโนโลยี DAB+

เทคโนโลยี DAB+ มี Transmission Mode ให้เลือกใช้งาน 4 mode แต่ละโหมดแตกต่างกันในปัจจัยต่าง ๆ เช่น เรื่องความถี่ที่ใช้ส่ง และรูปแบบการใช้งาน แสดงดังตารางที่2.1

ตารางที่ 2.1 รูปแบบการส่ง Transmission Mode

หัวข้อ	รูปแบบการส่ง (Transmission Mode; TM)			
	TM I	TM II	TM III	TM IV
ย่านความถี่ส่ง	VHF (Band I – III)	VHF (Band I – III), UHF (Band IV – V) L-Band (1452 – 1492 MHz)	ตั้งแต่ 3,000 MHz ลงมา	VHF (Band I – III), UHF (Band IV – V) L-Band (1452 – 1492 MHz)
รูปแบบการใช้งานทั่วไป	Terrestrial VHF	Terrestrial L-Band	Satellite L-Band	Terrestrial Urban L-Band
ความทนทานต่อสัญญาณสะท้อน (echo)[9] ที่เกิดขึ้นระหว่างเส้นทางจากจุดส่งมายังเครื่องรับ	อันดับ ๑ (ดีที่สุด)	อันดับ ๓	อันดับ ๔	อันดับ ๒
หมายเหตุ	ออกแบบมาให้เหมาะสมกับการส่งแบบโครงข่ายความถี่เดียว (SFN)	สามารถส่งเป็น SFN ขนาดเล็ก/กลาง ในย่าน L-Band ได้	ในการส่งผ่านสายเคเบิ้ลจะนิยมใช้ TM III เนื่องจากรองรับช่วงความถี่ใช้งานที่กว้างกว่าใน TM I, II และ IV	ส่งเป็น L-Band SFN ในพื้นที่เมือง (Urban Area) ได้
ความเร็วสูงสุดของยานพาหนะ** (รับสัญญาณในรูปแบบ mobile reception) (VHF-Band)	260/390 km/h	n.a	n.a	520/780 km/h
ความเร็วสูงสุดของยานพาหนะ** (รับสัญญาณในรูปแบบ mobile reception) (L-Band)	40/60 km/h	160/240 km/h	320/480 km/h	80/120 km/h

DAB+ ถูกออกแบบให้ใช้งานได้ทั้งในย่านความถี่ VHF (Band I–III), UHF (Band IV-V) และ L-Band (1452 - 1492 MHz) โดยใช้ความกว้างแถบคลื่นความถี่ของสัญญาณ (Bandwidth) เท่ากับ 1.536 MHz (ยังไม่นับรวม guard band ทั้ง 2 ฝั่งของช่องสัญญาณ) การออกอากาศสามารถทำได้ทั้งในรูปแบบโครงข่ายความถี่เดียว (Single Frequency Network ; SFN) และโครงข่ายหลายความถี่ (Multi Frequency Network ; MFN) ขณะที่เมื่อพิจารณาถึงเส้นทางการส่งออกอากาศพบว่าสามารถเลือกออกอากาศได้ทั้งในเส้นทางภาคพื้นดิน (Terrestrial) หรือผ่านทางสายเคเบิล การรับสัญญาณ DAB+ สามารถทำได้ทั้งในสภาวะ fixed, portable หรือ mobile ดังนี้

1. Fixed (คงที่) : คือ การใช้งานในที่ตั้ง และไม่มีการเคลื่อนไหว เช่น ในบ้าน หรือสถานที่ทำงานที่มีอุปกรณ์รับสัญญาณ DAB+ ติดตั้งอยู่ และเครื่องรับสัญญาณ DAB+ ที่ติดตั้งที่บ้าน เป็นต้น

2. Portable (พกพา) : คือ การใช้งานบนอุปกรณ์พกพา เช่น วิทยุ DAB+ พกพาที่มีแบตเตอรี่ในตัว เครื่องพกพานี้สามารถให้บริการในที่ต่าง ๆ ที่ผู้ใช้สามารถเคลื่อนไหวได้

3. Mobile (เคลื่อนที่) : คือ การใช้งานในรถ หรือยานพาหนะที่เคลื่อนที่อยู่ เช่น รถยนต์, รถบัส, หรือรถไฟ ระบบ DAB+ สามารถถ่ายทอดสัญญาณในยานพาหนะที่กำลังเคลื่อนที่ได้โดยตรง

อย่างไรก็ดีความถี่ออกอากาศ รูปแบบโครงข่าย และลักษณะเส้นทางการส่งออกอากาศที่เหมาะสมขึ้นกับ Transmission Mode (TM) ที่เลือกใช้โดย DAB+ ถูกออกแบบมาให้รองรับ TM ได้ทั้งสิ้น 4 ประเภท (TM I – TM IV) กล่าวคือ

1. Transmission Mode I (TM I) : เหมาะสำหรับการส่งภาคพื้นดินในย่านความถี่ VHF โดยรองรับการสร้างโครงข่ายในลักษณะ SFN ได้

2. Transmission Mode II (TM II) : เหมาะกับการส่งภาคพื้นดินในย่านความถี่ L-Band โดยสามารถทำเป็นโครงข่าย SFN ขนาดเล็ก/กลาง ได้

3. Transmission Mode III (TM III) : เหมาะสมกับกรณีการส่งในรูปแบบผ่านดาวเทียมบนย่าน L-Band นอกจากนี้ยังเป็น TM ที่เหมาะสมในกรณีการส่งผ่านทางสายเคเบิลเนื่องจากรองรับช่วงความถี่ใช้งานที่กว้างกว่า TM I , II และ IV

4. Transmission Mode IV (TM IV) : เหมาะกับการส่งภาคพื้นดินบนย่านความถี่ L-Band ในเขตพื้นที่เมือง (urban) ที่ผลกระทบจาก multipath effect จะมีมากกว่าในพื้นที่ชนบททั่วไป โดยรองรับการทำเป็นโครงข่าย SFN ได้

2.2 แผนความถี่วิทยุการกระจายเสียงระบบดิจิทัลเพื่อการทดลอง หรือทดสอบในประเทศไทย

อ้างอิงจาก ประกาศคณะกรรมการกิจการกระจายเสียง กิจการโทรทัศน์ และกิจการโทรคมนาคมแห่งชาติ (กสทช.) ณ วันที่ 26 พฤศจิกายน พ.ศ. 2563 โดยมีขอบข่ายดังนี้

1. กำหนดย่านความถี่วิทยุ 174 – 230 เมกะเฮิรตซ์ (MHz) สำหรับกิจการกระจายเสียงระบบดิจิทัล เพื่อการทดลอง หรือทดสอบ โดยจะนำความถี่วิทยุเท่าที่จำเป็นมาใช้งานอย่างคุ้มค่า และมีประสิทธิภาพ โดยปราศจากการรบกวนซึ่งกันและกัน

2. การใช้สิ่งอำนวยความสะดวกด้านกระจายเสียง หรือโทรทัศน์ พิจารณาจากความเหมาะสมด้านเศรษฐศาสตร์ และวิศวกรรมศาสตร์เป็นสำคัญ

3. กำหนดให้ใช้ระบบ Digital Audio Broadcasting (DAB) ที่มีการเข้ารหัสสัญญาณเสียงแบบ MPEG-4 HE AAC v2 หรือที่เรียกว่า DAB+ Audio

4. กำหนดพื้นที่นำร่องสำหรับตั้งสถานีวิทยุคมนาคม จำนวน 10 พื้นที่ ได้แก่ กรุงเทพมหานคร พัทยา ศรีราชา เชียงใหม่ ขอนแก่น นครราชสีมา นครศรีธรรมราช ภูเก็ต หัวหิน

และสงขลา โดยแต่ละพื้นที่มีโครงข่าย (Network) จำนวน 3 โครงข่าย แบ่งเป็น โครงข่ายระดับชาติ (National Network) 1 โครงข่าย และโครงข่ายระดับท้องถิ่น (Local Network) 2 โครงข่าย

5. กำหนดคุณลักษณะทางเทคนิคของสถานีวิทยุคมนาคมในพื้นที่นำร่องเท่าที่จำเป็น เพื่อเป็นการป้องกันการรบกวนการใช้ความถี่วิทยุ และให้เกิดความยืดหยุ่นในการทดลอง และทดสอบ

โดยการเปิดให้ใช้งานจริงนั้นยังไม่มีประกาศออกมาอย่างแน่ชัด เนื่องจากยังอยู่ในช่วงของการทดลองใช้งาน และผลักดันให้มีการใช้งานจริง จึงต้องรอการประกาศอย่างเป็นทางการจาก กสทช.

2.2.1 คลื่นความถี่

1. ย่านความถี่วิทยุ (Frequency Range) กำหนดให้ใช้ย่านความถี่วิทยุ 174–230 เมกะเฮิรตซ์ (MHz)

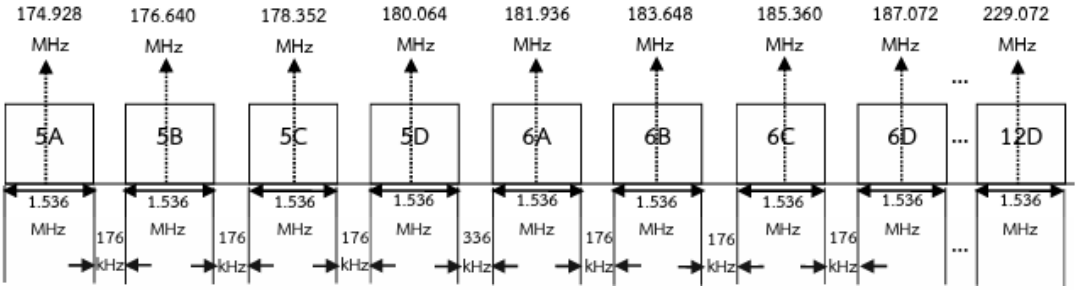
2. ช่องความถี่วิทยุ (Frequency Channel) บล็อกความถี่วิทยุ (Frequency Block) ความกว้างแถบคลื่นความถี่ (Bandwidth) และความกว้างแถบคลื่นความถี่ป้องกัน (Guard Band) กำหนดให้ใช้ช่องความถี่วิทยุ ช่องที่ 5 ถึง ช่องที่ 12 โดยแต่ละช่อง แบ่งออกเป็น 4 บล็อกความถี่วิทยุ ได้แก่ A, B, C และ D แต่ละบล็อกมีความถี่วิทยุ ความกว้างแถบคลื่นความถี่ และความกว้างแถบคลื่นความถี่ป้องกัน ตามที่กำหนดไว้ ซึ่งได้แสดงไว้ในตารางที่ 2.2 และ รูปที่ 2.6 [5]

ตารางที่ 2.2 ช่องความถี่วิทยุ บล็อกความถี่วิทยุ ความกว้างแถบคลื่นความถี่ และความกว้างแถบคลื่นความถี่ป้องกัน อ้างอิงจาก กสทช. ผว. 103-2563

Channel	Block	Frequency			Bandwidth (MHz)	Guard Band	
		Lower (MHz)	Center (MHz)	Upper (MHz)		Lower (kHz)	Upper (kHz)
5	A	174.160	174.928	175.696	1.536	–	176
	B	175.872	176.640	177.408	1.536	176	176
	C	177.584	178.352	179.120	1.536	176	176
	D	179.296	180.064	180.832	1.536	176	336

ตารางที่ 2.2 ช่องความถี่วิทยุ บล็อกความถี่วิทยุ ความกว้างแถบคลื่นความถี่ และความกว้างแถบคลื่นความถี่ป้องกัน (ต่อ) อ้างอิงจาก กสทช. ผว. 103-2563

Channel	Block	Frequency			Bandwidth (MHz)	Guard Band	
		Lower (MHz)	Center (MHz)	Upper (MHz)		Lower (kHz)	Upper (kHz)
6	A	181.168	181.936	182.704	1.536	336	176
	B	182.880	183.648	184.416	1.536	176	176
	C	184.592	185.360	186.128	1.536	176	176
	D	186.304	187.072	187.840	1.536	176	320
7	A	188.160	188.928	189.696	1.536	320	176
	B	189.872	190.640	191.408	1.536	176	176
	C	191.584	192.352	193.120	1.536	176	176
	D	193.296	194.064	194.832	1.536	176	336
8	A	195.168	195.936	196.704	1.536	336	176
	B	196.880	197.648	198.416	1.536	176	176
	C	198.592	199.360	200.128	1.536	176	176
	D	200.304	201.072	201.840	1.536	176	320
9	A	202.160	202.928	203.696	1.536	320	176
	B	203.872	204.640	205.408	1.536	176	176
	C	205.584	206.352	207.120	1.536	176	176
	D	207.296	208.064	208.832	1.536	176	336
10	A	209.168	209.936	210.704	1.536	336	176
	B	210.880	211.648	212.416	1.536	176	176
	C	212.592	213.360	214.128	1.536	176	176
	D	214.304	215.072	215.840	1.536	176	320
11	A	216.160	216.928	217.696	1.536	320	176
	B	217.872	218.640	219.408	1.536	176	176
	C	219.584	220.352	221.120	1.536	176	176
	D	221.296	222.064	222.832	1.536	176	336
12	A	223.168	223.936	224.704	1.536	336	176
	B	224.880	225.648	226.416	1.536	176	176
	C	226.592	227.360	228.128	1.536	176	176
	D	228.304	229.072	229.840	1.536	176	–



รูปที่ 2.6 แผนภาพบล็อกความถี่วิทยุ ความถี่วิทยุกึ่งกลาง ความกว้างแถบคลื่นความถี่ และความกว้างแถบคลื่นความถี่ป้องกัน

2.2.2 การส่งสัญญาณ

การส่งสัญญาณในกิจการกระจายเสียงระบบดิจิทัลต้องเป็นไปตามมาตรฐานการส่งสัญญาณที่กำหนด ดังนี้

2.2.2.1 ระบบ (System)

กำหนดให้ระบบส่งสัญญาณในกิจการกระจายเสียงระบบดิจิทัลเป็นระบบ Digital Audio Broadcasting (DAB) ตามที่กำหนดไว้ใน ETSI EN 300 401 V2.1.1 (2017-01) [6]

1. ช่วงคลื่น และ ความถี่ : มาตรฐานกำหนดช่วงคลื่น และความถี่ที่ใช้ในการกระจายเสียง DAB เป็นส่วนสำคัญที่กำหนดให้ระบบ DAB ทำงานในช่วงคลื่นความถี่ที่ถูกกำหนดไว้

2. การเข้ารหัสเสียง : มาตรฐานระบุวิธีการเข้ารหัสสัญญาณเสียงในระบบ DAB เพื่อให้สามารถถ่ายทอดได้ในรูปแบบดิจิทัล

3. การแบ่งช่องสัญญาณ : กำหนดวิธีการแบ่งช่องสัญญาณเพื่อให้สามารถรับสัญญาณจากรูปแบบหลาย ๆ ช่องทางที่ต่างกัน

4. การจัดการข้อมูล : ระบบ DAB มีการจัดการข้อมูลที่ถูกส่งไปพร้อมกับสัญญาณเสียง เช่น ข้อมูลเพลง, ข้อมูลการจราจร และข้อมูลเสริมอื่น ๆ

5. การรับสัญญาณ และการหาเส้นทางสัญญาณ : กำหนดข้อกำหนดเกี่ยวกับการรับสัญญาณ DAB และการหาเส้นทางสัญญาณที่เหมาะสม

6. ความเสถียรของสัญญาณ : มาตรฐานนี้ระบุข้อกำหนดเกี่ยวกับความเสถียรของสัญญาณที่จำเป็นในการรับสัญญาณ DAB

2.2.2.2 การมัลติเพล็กซ์ (Multiplex)

กำหนดให้การมัลติเพล็กซ์เป็นการมัลติเพล็กซ์แบบ Orthogonal Frequency Division Multiplex (OFDM) มีหลักการดังนี้

1. OFDM แบ่งคลื่นหลักออกเป็นช่องย่อย ๆ ที่มีความถี่ต่ำแต่มีจำนวนมาก แต่ละช่องนี้เรียกว่า "subcarrier"

2. แต่ละ subcarrier ถูกออกแบบให้มีคุณสมบัติที่ orthogonal กัน ซึ่งหมายความว่ามีมุมของคลื่นที่ไม่ทับกัน ช่วยลดปัญหาที่เกิดจากการโต้แย้งของสัญญาณ

3. OFDM ช่วยลดผลกระทบจาก multipath fading ซึ่งเกิดจากการสะท้อน และการหลุดของสัญญาณไปยังทางที่แตกต่างกัน

4. OFDM ช่วยในการปรับเปลี่ยนความถี่ของ subcarriers โดยอัตโนมัติในกรณีที่มีการรบกวน หรือสัญญาณสื่อสารจากแหล่งอื่น

5. OFDM สามารถรับสัญญาณที่มีความซับซ้อนได้ดี ทำให้เหมาะสำหรับการใช้ในสภาพแวดล้อมที่มีการรบกวน

2.2.2.3 การมอดูเลต (Modulation)

กำหนดให้การมอดูเลตเป็นการมอดูเลตแบบ Differential Quadrature Phase Shift Keying (D-QPSK) มีหลักการดังนี้

1. Differential Encoding : การมอดูเลตแบบ D-QPSK มีการใช้ Differential Encoding เพื่อเพิ่มความเสถียรของการรับสัญญาณ ใน Differential Encoding ข้อมูลถูกมอดูลเลตในรูปแบบของการเปลี่ยนแปลงของเฟสระหว่างสัญญาณที่ต่อเนื่อง

2. การเปลี่ยนแปลงของเฟส (Phase Changes) : ทุกครั้งที่ข้อมูลเปลี่ยนแปลง สถานะของเฟสที่ถูกมอดูลเลตจะเปลี่ยนแปลงตามไปด้วย

3. การลดการผิดพลาดในการรับสัญญาณ : การใช้ D-QPSK ช่วยลดผลกระทบจากการสูญเสียสัญญาณ หรือการถูกทำลายที่เกิดขึ้นในการถ่ายทอดข้อมูล

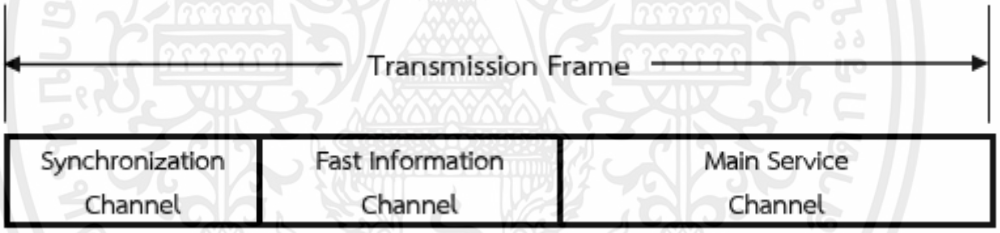
4. Fading : D-QPSK เหมาะสำหรับการใช้ในสภาพแวดล้อมที่มีการรบกวน เนื่องจากการใช้การเปลี่ยนแปลงของเฟส

2.2.2.4 โหมด (Mode)

กำหนดให้โหมดการส่งสัญญาณเป็น Model ที่มีพารามิเตอร์สำหรับการส่งสัญญาณเป็นไปตามที่ กำหนดไว้ใน ETSI EN 300 401 V2.1.1 (2017-01) โดยแสดงดังตารางที่ 2.3 และมีโครงสร้างเฟรมส่งสัญญาณ (Transmission Frame) แสดงดังรูปที่ 2.7 [5]

ตารางที่ 2.3 พารามิเตอร์สำหรับการส่งสัญญาณ Mode I อ้างอิงจาก กสทช. ผว. 103-2563

Transmitted Carriers	
Number of Transmitted Carriers	1 536
Carrier Spacing	1 kHz
Time	
Transmission Frame Duration	96 ms
OFDM Symbol Duration	1 246 µs
Guard Interval	246 µs
Null Symbol Duration	1 297 µs
OFDM Symbols	
Number of OFDM Symbols/Transmission Frame	77
Number of OFDM Symbols with Synchronization Channel	2
Number of OFDM Symbols with Fast Information Channel	3
Number of OFDM Symbols with Main Service Channel	72



รูปที่ 2.7 โครงสร้างเฟรมสัญญาณ

2.2.2.5 การเข้ารหัสแบบคอนโวลูชัน (Convolutional Coding)

กำหนดให้การเข้ารหัสแบบคอนโวลูชัน (Convolutional Coding) เป็นประเภท Equal Error Protection (EEP) เซต A ที่มีการเข้ารหัสเป็นจำนวนเท่าของ 8 กิโลบิตต่อวินาที (kbit/s) เป็นไปตามที่กำหนดไว้ใน ETSI EN 300 401 V2.1.1 (2017-01) มีหลักการดังนี้

1. Convolutional Coding : เป็นเทคนิคการเข้ารหัสที่ใช้การผสมข้อมูลเข้ากับข้อมูลที่ผ่านไปเพื่อสร้างรหัสการรับสัญญาณ ใน DAB มักถูกนำมาใช้เพื่อลดการสูญเสียข้อมูล และเพิ่มความเสถียรของการรับสัญญาณ

2. การลดการสูญเสีย (Error Reduction) : การใช้ convolutional coding ช่วยให้ระบบสามารถกลับคืนข้อมูลที่สูญเสียได้ในกรณีที่เกิดการรบกวน หรือสภาพแวดล้อมที่ไม่แน่นอน

3. Equal Error Protection (EEP) : EEP จะแนะนำการให้การปกป้องข้อมูลทุกระดับอย่างเท่าเทียมกัน (equal) ต่อความผิดพลาด (errors) ที่เกิดขึ้นในช่องสัญญาณ

4. การปกป้องจากความผิดพลาดทั้งสามระดับ : เซต A ของ EEP ใน DAB จะปกป้องข้อมูลจากระดับความผิดพลาดทั้งสามระดับ คือ level 1, level 2 และ level 3 ซึ่งแต่ละระดับมีความสามารถในการปกป้องข้อมูลจากความผิดพลาดต่าง ๆ ที่เกิดขึ้นในการถ่ายทอดสัญญาณ

5. การเพิ่มประสิทธิภาพในสภาพแวดล้อมที่เสี่ยงต่อการสูญเสีย : Convolutional coding ที่ใช้ EEP เซต A ช่วยเพิ่มประสิทธิภาพของระบบในสภาพแวดล้อมที่มีความเสี่ยงต่อการสูญเสียสัญญาณ เช่น ในการรับสัญญาณที่มีการเข้าสัญญาณรบกวน หรือสูญเสียได้

2.2.2.6 อัตราบิตสุทธิ (Net BitRate)

อัตราบิตสุทธิที่ได้จากโหมดการส่งสัญญาณ Mode I เท่ากับ 1,152 กิโลบิตต่อวินาที (kbit/s)

2.2.2.7 การเข้ารหัสสัญญาณเสียง (Audio Coding)

กำหนดให้การเข้ารหัสสัญญาณเสียงเป็นการเข้ารหัสแบบ MPEG-4 High Efficiency Advanced Audio Coding version 2 (MPEG-4 HE AAC v2) เป็นไปตามที่กำหนดไว้ใน ETSI TS 102 563 v1.2.1 (2010-05) [7] โดยเรียกการเข้ารหัสสัญญาณเสียงนี้ว่า DAB+ Audio

2.2.2.8 กำลังส่งออกอากาศสูงสุด (Maximum Effective Radiated Power)

กำหนดให้กำลังส่งออกอากาศสูงสุดต้องมีค่าไม่เกินที่กำหนดไว้ในตารางที่ 2.4-2.5

2.2.2.9 โพลาไรเซชันของการแพร่กระจายคลื่น (Transmitted Polarization)

กำหนดให้โพลาไรเซชันของการแพร่กระจายคลื่นเป็นโพลาไรเซชันแบบแนวตั้ง (Vertical Polarization)

2.2.2.10 การแพร่นอกแถบ (Out-of-band Emissions)

1. การแพร่นอกแถบกรณีวิกฤติ (Out-of-band Emission in Critical Case) กำหนดให้การแพร่นอกแถบกรณีวิกฤติใช้สำหรับการส่งสัญญาณในพื้นที่ที่มีการใช้งานบล็อก ความถี่วิทยุข้างเคียงกัน

2. การแพร่นอกแถบกรณีไม่วิกฤติ (Out-of-band Emission in Uncritical Case) กำหนดให้การแพร่นอกแถบกรณีไม่วิกฤติใช้สำหรับการส่งสัญญาณแบบอื่นที่ไม่เป็นไปตาม ข้อ 1

ตารางที่ 2.4ตารางแผนความถี่วิทยุการกระจายเสียงระบบดิจิทัลเพื่อการทดลอง หรือ ทดสอบ และคุณลักษณะทางเทคนิคของสถานีวิทยุคมนาคมในโครงข่ายระดับชาติ อ้างอิงจาก กสทช. ผว. 103-2563

No.	Station Name	Province	Location		EC	Eld	Blk	CF (MHz)	Max. ERP (kW)	Max. Ht (m)	Max. Ref. CA	
			Latitude	Longitude							Area (km^2)	Fig.
1.	Bangkok - N1	Bangkok	13.790514	100.525346	N1	0010010100000000	6C	185.360	10.0	185	9 853.34	4
2.	Pattaya - N1	Chon Buri	12.921483	100.866270	N1	0010010100000000	6C	185.360	0.4	60	1 040.24	5
3.	Si Racha - N1	Chon Buri	13.189822	100.950564	N1	0010010100000000	6C	185.360	0.5	43	2 184.01	6
4.	Chiang Mai - N1	Chiang Mai	18.853972	98.959528	N1	0010010100000000	6C	185.360	10.0	120	4 361.13	7
5.	Khon Kaen - N1	Khon Kaen	16.459040	102.848059	N1	0010010100000000	6C	185.360	10.0	80	7 218.04	8
6.	Nakhon Ratchasima - N1	Nakhon Ratchasima	14.947722	102.003760	N1	0010010100000000	6C	185.360	0.5	153	4 250.49	9
7.	Nakhon Sri Thammarat - N1	Nakhon Sri Thammarat	8.366633	99.977356	N1	0010010100000000	6C	185.360	2.0	110	2 761.67	10
8.	Phuket - N1	Phuket	7.899133	98.395480	N1	0010010100000000	6C	185.360	1.0	60	747.66	11
9.	Hua Hin - N1	Prachuap Khiri Khan	12.565142	99.935176	N1	0010010100000000	6C	185.360	0.5	60	1 883.81	12
10.	Song Khla - N1	Song Khla	7.037696	100.518640	N1	0010010100000000	6C	185.360	3.0	80	5 754.48	13

ตารางที่ 2.5ตารางแผนความถี่วิทยุการกระจายเสียงระบบดิจิทัลเพื่อการทดลอง หรือ ทดสอบ และคุณลักษณะทางเทคนิคของสถานีวิทยุคมนาคมในโครงข่ายระดับท้องถิ่น อ้างอิงจาก กสทช. ผว. 103-2563

No.	Station Name	Province	Location	EC	Eld	Blk	CF (MHz)	Max. ERP (kW)	Max. Ht (m)
1.	Bangkok - L1	Bangkok	Within Ref. CA of Fig. 4	L1	0010010100000001	5C	178.352	10.0	185
2.	Bangkok - L2	Bangkok	Within Ref. CA of Fig. 4	L2	0010010100000010	8C	199.360	10.0	185
3.	Pattaya - L1	Chon Buri	Within Ref. CA of Fig. 5	L1	0010010100000001	5C	178.352	0.4	60
4.	Pattaya – L2	Chon Buri	Within Ref. CA of Fig. 5	L2	0010010100000010	8C	199.360	0.4	60
5.	Si Racha - L1	Chon Buri	Within Ref. CA of Fig. 6	L1	0010010100000001	5C	178.352	0.5	43
6.	Si Racha - L2	Chon Buri	Within Ref. CA of Fig. 6	L2	0010010100000010	8C	199.360	0.5	43
7.	Chiang Mai - L1	Chiang Mai	Within Ref. CA of Fig. 7	L3	0010000001000001	7C	192.352	10.0	120
8.	Chiang Mai - L2	Chiang Mai	Within Ref. CA of Fig. 7	L4	0010000001000010	9C	206.352	10.0	120
9.	Khon Kaen - L1	Khon Kaen	Within Ref. CA of Fig. 8	L5	0010001011000001	10C	213.360	10.0	80
10.	Khon Kaen - L2	Khon Kaen	Within Ref. CA of Fig. 8	L6	0010001011000010	11C	220.352	10.0	80
11.	Nakhon Ratchasima - L1	Nakhon Ratchasima	Within Ref. CA of Fig. 9	L5	0010001011000001	10C	213.360	0.5	153
12.	Nakhon Ratchasima - L2	Nakhon Ratchasima	Within Ref. CA of Fig. 9	L6	0010001011000010	11C	220.352	0.5	153
13.	Nakhon Sri Thammarat - L1	Nakhon Sri Thammarat	Within Ref. CA of Fig. 10	L7	0010011111000001	7C	192.352	2.0	110
14.	Nakhon Sri Thammarat – L2	Nakhon Sri Thammarat	Within Ref. CA of Fig. 10	L8	0010011111000010	9C	206.352	2.0	110
15.	Phuket - L1	Phuket	Within Ref. CA of Fig. 11	L7	0010011111000001	7C	192.352	1.0	60
16.	Phuket - L2	Phuket	Within Ref. CA of Fig. 11	L8	0010011111000010	9C	206.352	1.0	60
17.	Hua Hin - L1	Prachuap Khiri Khan	Within Ref. CA of Fig. 12	L1	0010010100000001	5C	178.352	0.5	60
18.	Hua Hin - L2	Prachuap Khiri Khan	Within Ref. CA of Fig. 12	L2	0010010100000010	8C	199.360	0.5	60
19.	Song Khla - L1	Song Khla	Within Ref. CA of Fig. 13	L7	0010011111000001	7C	192.352	3.0	80
20.	Song Khla - L2	Song Khla	Within Ref. CA of Fig. 13	L8	0010011111000010	9C	206.352	3.0	80

2.2.3 การรับสัญญาณ

การรับสัญญาณในกิจการกระจายเสียงระบบดิจิทัลอ้างอิงตามมาตรฐานการรับสัญญาณที่กำหนด ดังนี้

2.2.3.1 ประเภทการรับสัญญาณ (Reception Mode)

กำหนดให้ประเภทของการรับสัญญาณเป็นการรับแบบเคลื่อนที่ (Mobile Reception)

2.2.3.2 ความแรงของสัญญาณต่ำสุด (Minimum Field Strength)

กำหนดให้ความแรงของสัญญาณต่ำสุดเป็นความเข้มของสนามไฟฟ้าสมมูลมัธยฐานต่ำสุด (Minimum Median Equivalent Field Strength) ที่สามารถรับสัญญาณแบบเคลื่อนที่ได้มีค่า 42.84 เดซิเบลไมโครโวลต์ต่อเมตร (dBuV/m) คำนวณโดยใช้ความถี่วิทยุ 200 เมกะเฮิรตซ์ ซึ่งเป็นความถี่วิทยุอ้างอิงสำหรับย่านความถี่วิทยุ 174 – 230 เมกะเฮิรตซ์ และที่ความสูงของสายอากาศรับสัญญาณ 1.50 เมตร (m) จากระดับพื้นดินเฉลี่ย โดยความแรงสัญญาณต่ำสุดที่ค่าดังกล่าวจะครอบคลุมพื้นที่ไม่ต่ำกว่าร้อยละ 99 ภายใต้สภาวะที่มีเฉพาะสัญญาณรบกวนจากสิ่งประดิษฐ์ที่มนุษย์สร้างขึ้น (Man-made Noise) ทั้งนี้การคำนวณค่าความแรงสัญญาณต่ำสุดที่ความถี่วิทยุข้างต้นให้เป็นไปตามตัวอย่างใน Recommendation ITU-R BS.1660-8 (06/2019) [8]

2.2.3.3 อัตราส่วนป้องกันการรบกวน (Protection Ratio)

อัตราส่วนป้องกันการรบกวน คือ อัตราส่วนระหว่างค่าความแรงสัญญาณที่ต้องการ (Wanted Signal) ต่อค่าความแรงสัญญาณรบกวน (Interfering Signal) ตามที่กำหนดใน Recommendation ITU-R BS.638 (1986) [9] โดยกำหนดให้อัตราส่วนป้องกันการรบกวนระหว่างบล็อกความถี่วิทยุให้เป็นไปตาม Recommendation ITU-R BS.1660-8 (06/2019) ซึ่งแสดงไว้ในตารางที่ 2.6

ตารางที่ 2.6 อัตราส่วนป้องกันการรบกวน อ้างอิงจาก กสทช. ผว. 103-2563

ระยะห่างจากบล็อกความถี่วิทยุ (บล็อก)	อัตราส่วนป้องกันการรบกวน (dB)
0	12
±1	-40
±2	-45
±3	-45

จากอัตราส่วนป้องกันการรบกวนที่กำหนดในตารางที่ 2.6 หากความแรงสัญญาณที่ต้องการมีค่า 42.84 เดซิเบลไมโครโวลต์ต่อเมตร สัญญาณรบกวนจากบล็อกความถี่วิทยุเดียวกัน ต้องต่ำกว่า 42.84 – 12 = 30.84 เดซิเบลไมโครโวลต์ต่อเมตร

2.2.3.4 พื้นที่การกระจายเสียงอ้างอิง (Reference Coverage Area)

กำหนดให้พื้นที่การกระจายเสียงอ้างอิง เป็นพื้นที่สำหรับอ้างอิงของการรับสัญญาณแบบเคลื่อนที่ที่มีความแรงของสัญญาณไม่น้อยกว่าความแรงของสัญญาณต่ำสุด และความแรงของสัญญาณใช้งาน (Usable Field Strength) โดยความแรงสัญญาณดังกล่าวต้องครอบคลุมพื้นที่ไม่ต่ำกว่าร้อยละ 99 และครอบคลุมระยะเวลาไม่ต่ำกว่าร้อยละ 95 สำหรับการใช้งานคลื่นความถี่ภายในโครงข่ายความถี่เดียว (Single Frequency Network) หรือครอบคลุมพื้นที่ไม่ต่ำกว่าร้อยละ 99 และครอบคลุมระยะเวลาไม่ต่ำกว่าร้อยละ 99 สำหรับการใช้งานคลื่นความถี่แบบอื่น ตามที่กำหนดในตารางที่ 2.4

2.2.4 โครงข่าย

กำหนดให้โครงข่ายแบ่งออกเป็น 2 ประเภท ดังนี้

2.2.4.1 โครงข่ายระดับชาติ (National Network)

โครงข่ายระดับชาติ หมายถึง ระบบเชื่อมโยงของกลุ่มเครื่องส่ง หรือถ่ายทอดสัญญาณเสียง ภาพ หรือข้อมูล ที่ใช้ในการส่งข่าวสารสาธารณะ หรือรายการจากสถานีวิทยุคมนาคมไปยังเครื่องรับ โดยมีวัตถุประสงค์เพื่อให้มีพื้นที่การกระจายเสียงอ้างอิงของรายการเดียวกันครอบคลุมทั้งประเทศ

2.2.4.2 โครงข่ายระดับท้องถิ่น (Local Network)

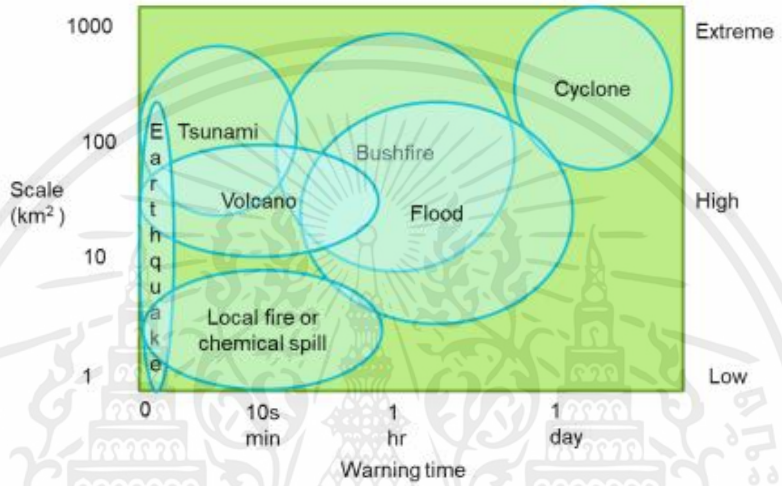
โครงข่ายระดับท้องถิ่น หมายถึง ระบบเชื่อมโยงของกลุ่มเครื่องส่ง หรือ ถ่ายทอดสัญญาณเสียง ภาพ หรือข้อมูล ที่ใช้ในการส่งข่าวสารสาธารณะ หรือรายการจากสถานีวิทยุคมนาคมไปยังเครื่องรับ โดยมีวัตถุประสงค์เพื่อให้มีพื้นที่การกระจายเสียงอ้างอิงของรายการเดียวกันครอบคลุมภายในจังหวัด หรือในกลุ่มจังหวัด

2.3 การออกอากาศและคำเตือนฉุกเฉิน DAB+

ปริญญานิพนธ์นี้อ้างอิงมาตรฐาน การออกอากาศ และคำเตือนฉุกเฉิน DAB+ เพื่อเป็นมาตรฐานในการสร้างระบบการตอบสนองต่อสถานการณ์ฉุกเฉินให้เป็นสากล

คำเตือนฉุกเฉินเป็นข้อความเพื่อแจ้งเตือนชุมชนถึงเหตุฉุกเฉินที่กำลังจะเกิดขึ้นโดยผู้กระจายเสียงวิทยุควรจะสามารถเป็นผู้ให้ข้อมูลที่สำคัญในสถานการณ์ฉุกเฉิน และมีหน้าที่ในการดูแลต่อสาธารณะ ปริมาณของข้อมูลที่ต้องแจ้งนั้นขึ้นอยู่กับสถานการณ์ ตัวอย่างเช่น อาจมีการรายงานความคืบหน้าเป็นประจำเกี่ยวกับเหตุเพลิงไหม้ในโรงงาน หรือการรั่วไหลของสารพิษ เกี่ยวกับการเข้าถึงพื้นที่ได้รับผลกระทบ และมาตรการสนับสนุนสำหรับเหตุการณ์ที่มีผลกระทบเป็นวงกว้าง เช่น น้ำท่วม หรือพายุ โดยจะต้องสามารถให้ข้อมูลได้อย่างต่อเนื่องว่าต้องทำอย่างไรจึงจะปลอดภัย ควรไปที่ไหน และไม่ควรไปบริเวณไหน

ระยะเวลาที่สามารถแจ้งเตือนชุมชนได้ขึ้นอยู่กับเหตุการณ์ที่เกิดขึ้น แสดงดังรูปที่ 2.8 [10] พายุ และเหตุการณ์น้ำท่วมบางส่วนสามารถคาดการณ์ล่วงหน้าได้หลายวัน เพราะตัวภัยพิบัติเคลื่อนไหวช้า และคาดเดาได้ ไฟป่าอาจมีการเตือนบริเวณที่เป็นอันตรายล่วงหน้าหลายวัน แต่จะมีการแจ้งเตือนบางประเภทสามารถแจ้งล่วงหน้าได้เพียงชั่วโมง หรือนาทีเท่านั้น เช่น การโจมตีของผู้ก่อการร้าย และแผ่นดินไหว บางประเภทก็ไม่สามารถแจ้งล่วงหน้าได้เลย เช่น การปะทุของภูเขาไฟมักเกิดขึ้นโดยไม่มีการแจ้งเตือนล่วงหน้า



รูปที่ 2.8 ระดับของภัยพิบัติ และระยะเวลาในการแจ้งเตือนภัยพิบัติ

2.3.1 หน่วยงานที่ดูแลการเตือนภัยฉุกเฉิน

โดยทั่วไปคำเตือนฉุกเฉินจะออกโดยองค์กรอย่างเป็นทางการ เช่น รัฐบาล หน่วยงานที่มีความรับผิดชอบ และความสามารถในการตรวจจับ และตอบสนองต่อสถานการณ์ฉุกเฉิน หน่วยงานดังกล่าวมักจะได้รับข้อมูลจากชุมชนที่ได้รับผลกระทบซึ่งให้บริการอยู่บน "ภาคพื้นดิน" ซึ่งสามารถสังเกต และติดตามสถานการณ์ที่กำลังเกิดขึ้นได้อย่างรวดเร็ว

บางประเทศมีเครือข่ายเตือนภัยฉุกเฉินอยู่แล้ว ซึ่งสามารถเปิดใช้งานได้โดยหน่วยงานที่เกี่ยวข้อง และรวมถึงเว็บไซต์ แอปพลิเคชันบนโทรศัพท์เคลื่อนที่ หรือระบบเสียงเตือน และลำโพง

2.3.2 การสลับการประกาศสัญญาณเตือน (AAS)

การสลับการประกาศสัญญาณเตือน (AAS) โดยจะให้ความสามารถในการเปลี่ยนเส้นทางเครื่องรับ DAB+ เพื่อเล่นบริการสถานีวิทยุที่กำหนดไว้ล่วงหน้า ซึ่งจะส่งเสียงฉุกเฉินข้อมูลคำแนะนำและทิศทางของภัยพิบัติ คุณสมบัติการเตือนฉุกเฉินของ AAS นี้เหมาะสำหรับสถานการณ์เร่งด่วนที่ต้องมีข้อความเตือนภัยเข้าถึงผู้ใช้งานโดยเร็วที่สุด เช่น ภายในสิบนาที

ระยะเวลาการเตือน ยกตัวอย่างเช่น หนึ่งชั่วโมงเป็นเวลาเพียงพอที่จะเตือนประชาชนเกี่ยวกับสถานการณ์ด้วยวาจาผ่านทางผู้บรรยายวิทยุ เมื่อสถานการณ์ฉุกเฉินบรรเทาลง AAS จะถูกปิดใช้งาน และเครื่องรับจะกลับสู่สถานะเดิมกลับสู่สถานีที่เลือกไว้ก่อนหน้า

2.4 การแจ้งเตือนภัยผ่าน RDS บนระบบ FM Analog

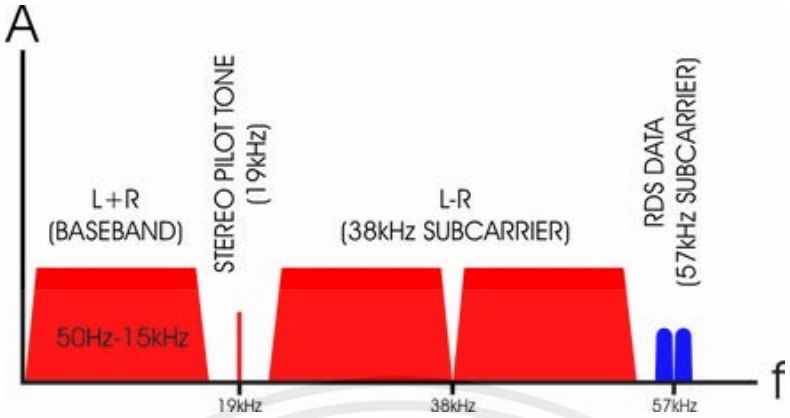
ปริญญานิพนธ์นี้อ้างอิงมาตรฐานการแจ้งเตือนภัยผ่าน RDS บนระบบ FM Analog เพื่อใช้เปรียบเทียบกับมาตรฐานการออกอากาศฉุกเฉินของ DAB+ และนำไปพัฒนาระบบการออกอากาศฉุกเฉินของ DAB+ ให้ดียิ่งขึ้น

RDS (Radio Data System) คือ การส่งข้อมูลที่เป็น Data ไปพร้อมกับสัญญาณ carrier ของสัญญาณเอฟเอ็ม โดยใช้ Subcarrier ที่ความถี่วิทยุ 57 kHz แสดงดังรูปที่ 2.9 [11] เพื่อใช้ในการส่งสัญญาณข้อมูลวิทยุ ในความเป็นจริงแล้วระบบ RDS ได้มีการพัฒนาระบบมานานกว่า 20 ปี เพื่อให้สามารถส่งข้อมูลที่เป็น text และ picture ได้ เช่น ข้อมูลเกี่ยวกับการจราจร การเตือนภัยพิบัติต่าง ๆ โฆษณาประชาสัมพันธ์ เพื่อให้ผู้ฟังได้เข้าถึงข้อมูลไปพร้อม ๆ กับการฟังเพลง หรือข่าวสารต่าง ๆ ได้

จากแผนความถี่วิทยุกิจการกระจายเสียงระบบเอฟเอ็ม ตามประกาศของ กสทช. [12] ได้กำหนดให้แต่ละสถานีส่งข้อมูลผ่านระบบ RDS เพื่อเป็นการระบุตัวตน และเพื่อตรวจสอบสถานีวิทยุกระจายเสียง โดยมีรายละเอียด ดังนี้

1. No. ลำดับที่
2. Station Name ชื่อสถานีวิทยุกระจายเสียง
3. Call Sign สัญญาณเรียกขานของสถานีวิทยุกระจายเสียง (Station Call Sign)
4. PI Code รหัสรายการ (Program Identification Code: PI Code)
5. Lat (N) ละติจูด (Latitude) ของพิกัดที่ตั้งสายอากาศในหน่วยองศาเหนือ
6. Long (E) ลองจิจูด (Longitude) ของพิกัดที่ตั้งสายอากาศในหน่วยองศาตะวันออก
7. Freq (MHz) ความถี่วิทยุ (Frequency) ในหน่วยเมกะเฮิรตซ์ (MHz)
8. Max ERP (kW) กำลังส่งออกอากาศสูงสุด ในหน่วยกิโลวัตต์ (kW)
9. Ht (m) ความสูงของจุดกึ่งกลางสายอากาศจากระดับพื้นดิน (Antenna Height) ในหน่วยเมตร

โดยการที่จะรับข้อมูลที่ส่งผ่านระบบ RDS ของระบบเอฟเอ็มได้นั้น ผู้ฟังจำเป็นต้องมีเครื่องรับวิทยุที่รองรับระบบ RDS ด้วย ซึ่งปัจจุบันเครื่องรับวิทยุประเภทนี้มีการจำหน่ายในหลายประเทศในยุโรป และอเมริกา



รูปที่ 2.9 RDS-Signal in the FM Spectrum

2.5 การเปรียบเทียบ DAB+ กับ RDS

DAB+ เป็นระบบดิจิทัลทั้งหมด ดังนั้นในการส่งข้อมูลเพิ่มเติมไปพร้อมกับเสียงจึงไม่ใช่ปัญหา หลายสถานียังมีการส่งรูปภาพเป็นสไลด์โชว์ (MOT) โดยจะมีการแสดง โลโก้ของสถานี, รูปภาพปกอัลบั้ม ฯลฯ เรียกว่า MOT (Multimedia Object Transfer)

ในการเปรียบเทียบระหว่าง DAB+ กับระบบ RDS ค่อนข้างมีความแตกต่างกันโดยไม่สามารถเทียบกันได้ โดยใน DAB+ ข้อมูลบางอย่าง เช่น ชื่อเพลง และศิลปิน จะถูกแทรกลงในสตรีมเสียง HE-AAC โดยตรง เรียกข้อมูลนี้ว่า PAD (Program Associated Data)

โดยใน DAB+ ยังสามารถใช้พื้นที่เพิ่มเติมในสตรีม HE-AAC ที่เราเรียกว่า EXTENDED PAD (ส่วนเพิ่มเติมพิเศษ 8Kbps) โดยเราสามารถใช้ส่วนเพิ่มเติมนี้เพื่อทำการส่งสไลด์โชว์ (MOT) สิ่งสำคัญคือต้องตระหนักว่าสถานีมีอัตราบิตที่ 88 Kbps แบ่งเป็นส่วนของสไลด์โชว์ 8Kbps ดังนั้นจึงมีการสตรีมเสียงที่อัตราบิต 80 Kbps เท่านั้น

2.6 ระบบปฏิบัติการ Linux

ปริญญานิพนธ์นี้ใช้ระบบปฏิบัติการ Linux ร่วมกับ Raspberry Pi4 โดยจะเป็นระบบปฏิบัติการหลักที่ใช้ในการประมวลผลข้อมูล รวมถึงใช้ในการสร้าง Receiver Module

Linux คือ ระบบปฏิบัติการ (Operating System) ประเภทหนึ่งเช่นเดียวกับ Windows หรือ Unix และระบบอื่น ๆ [13] ตามความหมายของ Linux แล้วจริง ๆ หมายถึง Linux kernel หรือ operating system kernel ซึ่งทำหน้าที่เป็นตัวกลางเชื่อมต่อระหว่าง Hardware และ application เพื่อบริหารจัดการ resource ที่มีอยู่ให้เหมาะสม พูดสั้น ๆ ก็คือระบบปฏิบัติการหนึ่งที่ใช้ควบคุมอุปกรณ์อิเล็กทรอนิกส์ตั้งแต่เครื่องคอมพิวเตอร์ไปจนถึงอุปกรณ์ IoT ขนาดเล็ก โดยระบบปฏิบัติการถูกออกแบบมาให้เป็น open source กล่าวคือ เป็นระบบที่เปิดเผยโค้ด ใคร ๆ ก็สามารถเข้าถึงโค้ด

และเข้าร่วมพัฒนาได้ จึงไม่ผูกขาด และไม่ได้ทำกำไรจากการหาเงินมากมายเหมือนระบบปฏิบัติการดัง ๆ ที่เราใช้กัน

2.6.1 ข้อดีของระบบปฏิบัติการ Linux

1. เป็นระบบปฏิบัติการที่ใช้งานได้ฟรี ไม่มีค่าลิขสิทธิ์

2. ทำงานได้บนเครื่องพีซีทั่วไป ที่มีหน่วยประมวลผลกลางตั้งแต่ 80386 ขึ้นไป จึงเป็นระบบปฏิบัติการที่มีความต้องการทรัพยากรของระบบในขั้นต่ำ

3. สามารถทำงานได้รวดเร็ว เนื่องจากมีระบบการจัดการหน่วยความจำเสมือน (Virtual Memory) การจัดการทำงานแบบ Multitasking และระบบป้องกันการรบกวนการทำงานระหว่าง Process ต่าง ๆ

4. มีความสามารถแบบ UNIX เนื่องจากไลบรารีที่ใช้ส่วนใหญ่เป็นไลบรารีมาตรฐานของ UNIX ซึ่งช่วยให้โปรแกรมที่พัฒนาบน UNIX สามารถทำงานบน Linux ได้

5. สามารถใช้งานร่วมกับดอส (DOS) และ Microsoft Windows โดยการแบ่งพาร์ทิชันบนฮาร์ดไดรฟ์ เพื่อให้แต่ละระบบปฏิบัติการมีพื้นที่เก็บข้อมูลของตัวเอง เมื่อแบ่งพาร์ทิชันเสร็จสิ้น ตัว Bootloader ที่ใช้สำหรับการเลือก Boot ระหว่างระบบปฏิบัติการที่ติดตั้งจะทำงาน โดยสามารถเลือกที่จะ Boot เข้าระบบปฏิบัติการที่ต้องการ โดยจะมีการแชร์ข้อมูลระหว่างระบบปฏิบัติการทำให้สามารถเข้าถึงข้อมูลในพาร์ทิชันที่มีการแบ่งร่วมได้

6. เป็นระบบปฏิบัติการแบบเปิด เนื่องจากทุกฟังก์ชันมี Source Code แนบมา ทำให้มีผู้พัฒนาจากทั่วโลกสามารถเข้ามาพัฒนา และแก้ไขข้อบกพร่องของระบบได้ตลอด ช่วยให้ระบบปฏิบัติการ Linux ถูกพัฒนาอย่างต่อเนื่อง และมีประสิทธิภาพ

7. รองรับการใช้งานของผู้ใช้หลาย ๆ คนได้พร้อม ๆ กัน หมายความว่าผู้ใช้แต่ละคนสามารถที่จะ remote login ผ่านโปรแกรม telnet หรือ secure shell เพื่อเข้าไปใช้งานเครื่อง Server ที่ใช้ระบบปฏิบัติการ Linuxได้หลาย ๆ คนพร้อม ๆ กัน

8. ระบบ Linux นั้นมีโปรแกรมแทบจะทุกอย่างให้ใช้ฟรี ซึ่งสามารถทำงานได้ดีพอๆกับโปรแกรมในระบบ Windows

2.6.2 ส่วนประกอบของ Linux operating system

1. The Bootloader : เป็น software ที่ทำหน้าที่จัดการเรื่องการ boot ของ computer สำหรับ user หรือก็คือหน้าจอที่แสดงขึ้นมาช่วงที่กำลังเริ่มเข้าสู่ระบบปฏิบัติการ

2. The kernel : ส่วนนี้เรียกได้ว่าเปรียบเสมือนคำเรียกของ “Linux” เพราะมันคือระบบส่วนกลางที่ทำหน้าจัดการทรัพยากรต่าง ๆ เช่น CPU, memory และอุปกรณ์เสริมต่าง เป็น layer ต่ำสุดที่อยู่ใกล้กับ OS

3. Daemons : เป็นส่วนที่ทำงานอยู่เบื้องหลัง (background service) เริ่มทำงานตั้งแต่ระหว่างที่ boot และเริ่ม login เข้าสู่ระบบ

4. The Shell : เป็นคำที่มักจะคุ้นเคยกันสำหรับ Linux เพราะว่า shell คือการทำงานของคำสั่งที่ทำให้สามารถควบคุม และสั่งการผ่านการพิมพ์ตัวอักษรเข้าไป ซึ่งเป็นส่วนหนึ่งที่ทำให้ user หลายคนค่อนข้างกลัวในการใช้งาน แต่ใน Linux desktop รุ่นใหม่ไม่จำเป็นต้องใช้ command line แล้ว

5. Graphical Server : เป็นระบบที่ช่วยเสริมการแสดงผลบนจอ monitor

6. Desktop Environment : คือส่วนที่ user ใช้งานจริง ซึ่งมีให้เลือกได้หลายที่โดยซึ่งก็คือชุดของ application ต่าง ๆ ที่ถูกนำมารวมกัน เช่น managers, configuration tools, web browsers และ games

7. Applications : เนื่องจาก Desktop environment นั้นไม่ได้จัด application มาครบเหมือน Window หรือ Mac เนื่องจาก Linux มี software ที่มีคุณภาพที่ง่ายต้องการค้นหาแล้วติดตั้ง Linux ที่ได้รับความนิยมส่วนใหญ่มักจะมีเครื่องมือที่ใช้สำหรับค้นหา และติดตั้ง application ติดมาให้ เช่น Ubuntu Linux ก็จะมี software center คือ apt ที่ใช้ในการ download และติดตั้ง application จากศูนย์กลาง

2.6.3 ระบบปฏิบัติการ Ubuntu

Ubuntu คือ ระบบปฏิบัติการคอมพิวเตอร์ที่เป็นระบบแบบเปิด [14] แสดงดังรูปที่ 2.10 [15] ซึ่งมีพื้นฐานจาก Linux Distribution หรือ Linux Distro ระบบปฏิบัติการที่ถูกออกแบบเพื่อการแบ่งปัน มีการปรับแต่ง และเพิ่มซอฟต์แวร์พื้นฐานต่าง ๆ สำหรับพร้อมใช้งานได้ทันที และเป็น Open Source ภายใต้สัญญาอนุญาตแบบ GNU/GPL ที่สามารถนำไปใช้, ปรับปรุง และเปลี่ยนแปลง ได้อย่างอิสระเสรี โดยไม่มีค่าใช้จ่าย

Ubuntu มีทั้งหมด 3 ประเภท ได้แก่ Ubuntu Desktop, Ubuntu Server และ Ubuntu Core โดย Ubuntu ได้รับการสนับสนุน และพัฒนาต่อมาจาก Debian ซึ่งเป็นชุดของซอฟต์แวร์เสรีหรือซอฟต์แวร์ที่สามารถนำไปใช้, แก้ไข, ดัดแปลง, พัฒนา และจำหน่ายแจกจ่ายได้ โดยไม่ต้องเสียค่าลิขสิทธิ์ที่ใช้ Linux เป็น Kernel หรือส่วนประกอบหลักของระบบปฏิบัติการ ซึ่งคอยดูแลบริหารทรัพยากรของระบบ และใช้เครื่องมือต่าง ๆ ในโครงการ GNU ประกอบกันเป็นระบบปฏิบัติการ

รูปที่ 2.10 ระบบปฏิบัติการ Ubuntu

2.7 Python

ภาษา Python คือ ภาษาโปรแกรมคอมพิวเตอร์ระดับสูง [16] แสดงดังรูปที่ 2.11 [17] โดยถูกออกแบบมาให้เป็นภาษาสคริปต์ที่อ่านง่าย โดยตัดความซับซ้อนของโครงสร้างของภาษาออกไป ในส่วนของการแปลงชุดคำสั่งที่เขียนให้เป็นภาษา Python มีการทำงานแบบ Interpreter คือเป็นการแปลชุดคำสั่งทีละบรรทัด เพื่อป้อนเข้าสู่หน่วยประมวลผลให้คอมพิวเตอร์ทำงานตามที่เราต้องการ นอกจากนั้นภาษาโปรแกรม Python ยังสามารถนำไปใช้ในการเขียนโปรแกรมได้หลากหลายประเภท โดยไม่ได้จำกัดอยู่ที่งานเฉพาะทางใดทางหนึ่ง (General-purpose language) จึงทำให้มีการนำไปใช้กันแพร่หลายในหลายองค์กรใหญ่ระดับโลก เช่น Google, YouTube, Instagram, Dropbox และ NASA เป็นต้น

รูปที่ 2.11 ภาษา Python

2.8 ภาษา C++

ภาษาซีพลัสพลัส (C++) เป็นภาษาโปรแกรมคอมพิวเตอร์อเนกประสงค์ [18] แสดงดังรูปที่ 2.12 [19] มีโครงสร้างภาษาที่มีการจัดชนิดข้อมูลแบบสแตติก (statically typed) และสนับสนุนรูปแบบการเขียนโปรแกรมที่หลากหลาย (multi-paradigm language) ได้แก่ การโปรแกรมเชิงกระบวนคำสั่ง, การนิยามข้อมูล, การโปรแกรมเชิงวัตถุ, และการโปรแกรมแบบเจเนริก (generic programming) ภาษาซีพลัสพลัสเป็นภาษาโปรแกรมเชิงพาณิชย์ที่นิยมมากภาษาหนึ่งนับตั้งแต่ช่วงทศวรรษ 1990

C++ มีการเพิ่ม OOP เข้ามา ทำให้ไม่เร็วเท่า C (โดยจะช้ากว่ากันประมาณ 1.5 เท่า) ความซับซ้อนเกิดจากการเพิ่มคุณสมบัติเด่น ๆ ของการเขียนโปรแกรมสมัยใหม่ให้ C เช่น OOP และ generic programming ทำให้ตรวจสอบข้อผิดพลาดระหว่างคอมไพล์ได้ดีกว่า และลดปัญหาที่อาจจะเกิดระหว่าง execute ได้ดีกว่า

รูปที่ 2.12 ภาษา C++

2.9 IEEE 802.11 มาตรฐานการทำงานของระบบเครือข่ายไร้สาย (WIFI)

IEEE 802.11 คือ มาตรฐานการทำงานของระบบเครือข่ายไร้สาย [20] กำหนดโดยสถาบันวิชาชีพวิศวกรไฟฟ้า และอิเล็กทรอนิกส์ (IEEE : Institute of Electrical and Electronics Engineers) เป็นสถาบันระดับนานาชาติที่ไม่หวังผลกำไร ทำหน้าที่ดูแลเกี่ยวกับไฟฟ้า และคอมพิวเตอร์ เป็นมาตรฐานกลางที่ได้นำมาใช้เพื่อเชื่อมโยงอุปกรณ์ไร้สายเข้าด้วยกันในระบบ

เทคโนโลยี IEEE 802.11 มีหลายมาตรฐาน โดยมาตรฐานที่นิยมใช้กัน ได้แก่

1) IEEE 802.11a มีการรับส่งข้อมูลได้สูงสุด 56Mbps บนความถี่ 5GHz ซึ่งมีคลื่นรบกวนน้อยกว่า 2.4GHz แต่ด้วยการใช้ความถี่ในการส่งสัญญาณที่สูงทำให้ระยะในการส่งสัญญาณค่อนข้างใกล้กว่า 2.4GHz ทะลุทะลวงสิ่งกีดขวางได้น้อยประมาณ 35 เมตรในโครงสร้างปิด และ 120 เมตรในพื้นที่โล่ง

2). IEEE 802.11b มีการรับส่งข้อมูลได้สูงสุดที่ 11Mbps บนความถี่ 2.4GHz ด้วยความถี่ที่ต่ำทำให้สามารถส่งสัญญาณ และทะลุทะลวงสิ่งกีดขวางได้ดีกว่า 5GHz โดยอยู่ที่ 38เมตร ในโครงสร้างแบบปิด และ 140เมตร ในพื้นที่โล่ง

3). IEEE 802.11g มีการรับส่งข้อมูลได้สูงสุดที่ 36-54 Mbps สามารถปรับความเร็วลดลงได้ต่ำที่สุดอยู่ที่ 2Mbps บนความถี่ 2.4GHz ด้วยความถี่ที่ต่ำทำให้สามารถส่งสัญญาณ และทะลุทะลวงสิ่งกีดขวางได้ดีกว่า 5GHz โดยอยู่ที่ 38เมตร ในโครงสร้างแบบปิด และ 140เมตร ในพื้นที่โล่ง

4). IEEE 802.11n มีการรับส่งข้อมูลได้สูงสุดที่ 300Mbps บนความถี่ 2.4GHz และ 5GHz สามารถส่งสัญญาณได้ในโครงสร้างแบบปิดอยู่ที่ 70เมตร และ 250เมตร ในพื้นที่โล่ง เพิ่มความสามารถในการป้องกันสัญญาณรบกวนจากอุปกรณ์อื่น ๆ ที่ใช้ความถี่ 2.4GHz และ รองรับอุปกรณ์ที่มีมาตรฐาน IEEE 802.11b และ 802.11g ได้

5). IEEE 802.11-2012 รองรับการรับส่งข้อมูลไร้สายในความถี่ทั้ง 2.4GHz และ 5GHz รองรับความเร็วในการถ่ายโอนข้อมูลตั้งแต่ 1Mbps ไปจนถึง 600Mbps หรือมากกว่านั้น ขึ้นอยู่กับเทคโนโลยีการสื่อสารที่ใช้ เช่น 802.11n และ 802.11ac

6). IEEE 802.11ac มาตรฐาน 5GHz เพื่อลดการแทรกบางประการ และเพิ่มความเร็วในการรับส่งข้อมูล สามารถรับส่งข้อมูลได้สูง 500Mbps ถึง 1Gbps ใช้ RF แบนด์วิดท์ที่กว้าง 80–160 MHz รองรับการเชื่อมต่อช่องส่งข้อมูล เพื่อเพิ่มความกว้างของช่องส่งข้อมูล ทำให้สามารถรับส่งข้อมูลได้มากพร้อมกัน

7). IEEE 802.11ad หรือ WiGig ใช้ความถี่ในช่วง 60GHz เพื่อให้มีความเร็วสูง และเพิ่มประสิทธิภาพในการรับส่งข้อมูล รองรับความเร็วในการรับส่งข้อมูลที่สูงมาก โดยมีความเร็วสูงสุดถึง 7 Gbps เหมาะสำหรับการรับส่งข้อมูลในระยะใกล้ เนื่องจากสัญญาณในช่วงความถี่ 60GHz มีการแพร่กระจายน้อย

2.10 RTMP (Real-Time Messaging Protocol)

RTMP (Real-Time Messaging Protocol) เป็นโปรโตคอลที่ถูกพัฒนาขึ้นโดย Adobe Systems [21] เพื่อให้บริการการสื่อสารแบบ real-time ระหว่างไคลเอนต์ (client) และเซิร์ฟเวอร์ (server) โดยเฉพาะในการสตรีมมิ่งวิดีโอ และเสียงในรูปแบบ real-time หรือการส่งข้อมูลสื่อที่ต้องการความลื่น และความเร็วในการส่ง โดยการทำงานของ RTMP มีดังนี้

1). การสตรีมมิ่งสด (Live Streaming) : RTMP มักถูกใช้ในการส่งสัญญาณวิดีโอ และเสียงในเวลาที่เกิดขึ้น (real-time) โดยเฉพาะในการสตรีมมิ่งสดทางอินเทอร์เน็ต

2). การสตรีมเนื้อหามัลติมีเดีย (Multimedia Streaming) : นอกจากการสตรีมมิ่งสด โปรโตคอลนี้ยังสามารถใช้ในการส่งข้อมูลมัลติมีเดียอื่น ๆ ที่ไม่ต้องการรองรับการสตรีมมิ่งสด

3). การสื่อสารแบบ Real-time : RTMP ให้ความสามารถในการสื่อสารแบบ real-time ระหว่างไคลเอนต์ และเซิร์ฟเวอร์ ซึ่งเป็นสิ่งสำคัญในการสตรีมมิ่ง และการติดต่อกับแอปพลิเคชันที่ต้องการการตอบสนองทันที

4). รูปแบบการสื่อสาร : RTMP มีรูปแบบการสื่อสารที่ได้รับการออกแบบมาเพื่อรองรับการส่งข้อมูลสื่อแบบ real-time โดยมีการควบคุมการส่งข้อมูลแบบ chunk หรือชุดข้อมูลที่ถูกแบ่งเป็นส่วน ๆ

โครงสร้างการทำงานของ RTMP มีดังนี้

1). การเชื่อมต่อ (Handshake) : เริ่มต้นด้วยขั้นตอนการเชื่อมต่อระหว่างไคลเอนต์ และเซิร์ฟเวอร์ โดยที่ทั้งไคลเอนต์ และเซิร์ฟเวอร์จะทำการ handshake เพื่อกำหนดการทำงานร่วมกัน

2). การส่งข้อมูล (Data Transmission) : หลังจากการเชื่อมต่อเสร็จสมบูรณ์ ข้อมูลวิดีโอและเสียงจะถูกส่งระหว่างไคลเอนต์ และเซิร์ฟเวอร์ในรูปแบบ real-time

3). การควบคุม (Control) : RTMP มีการควบคุมที่เกี่ยวข้องกับการจัดการการสตรีมมิ่งและการควบคุมการสื่อสารของไคลเอนต์กับเซิร์ฟเวอร์

4). การจัดการเนื้อหา (Content Management) : RTMP สามารถจัดการการส่งข้อมูลวิดีโอ และเสียงในรูปแบบ real-time และสนับสนุนการสตรีมมิ่งข้อมูลต่าง ๆ ให้รับส่งได้อย่างต่อเนื่องไม่ขาดตอน (smooth)

5). การสตรีมมิ่ง (Streaming) : RTMP ถูกออกแบบมาเพื่อการสตรีมมิ่งข้อมูลต่าง ๆ ในเวลาจริง ซึ่งมีการรองรับการสตรีมมิ่งวิดีโอ และเสียงในรูปแบบ live streaming

6). การรองรับความหลากหลาย (Versatility) : RTMP มีความยืดหยุ่น และสามารถใช้งานได้ในหลายแพลตฟอร์ม รวมถึงแอปพลิเคชันที่ต้องการการสื่อสาร real-time

2.11 HLS (HTTP Live Streaming)

HTTP Live Streaming (เรียกโดยย่อว่า HLS) คือโปรโตคอลการส่งผ่านเครือข่ายสื่อสตรีมมิ่งแบบ HTTP [22] ที่เสนอโดย Apple เป็นส่วนหนึ่งของระบบซอฟต์แวร์ QuickTime ทำงานโดยแบ่งสตรีมทั้งหมดเป็นไฟล์ที่ใช้ HTTP ขนาดเล็กเพื่อดาวน์โหลด และดาวน์โหลดเพียงไม่กี่ครั้งต่อการสตรีม เมื่อสตรีมสื่อกำลังเล่นไคลเอนต์สามารถเลือกที่จะดาวน์โหลดทรัพยากรเดียวกันในอัตราที่แตกต่างกันจากแหล่งข้อมูลสำรองหลายแหล่งทำให้เซสชันสื่อการสตรีมสามารถปรับให้เข้ากับอัตราข้อมูลที่แตกต่างกันได้

เมื่อเริ่มเซสชันสื่อการสตรีม ไคลเอนต์จะดาวน์โหลดไฟล์เพลย์ลิสต์ M3U (m3u8) ซึ่งเป็นหมวดหมู่ของสตรีมที่เป็นไปได้ทั้งหมด และจากนั้นจะร้องขอสตรีมที่เหมาะสมด้วย HTTP ในการโหลดสื่อแต่ละส่วน ส่วนเนื้อหาสื่อจะถูกส่งผ่านไฟล์วิดีโอ และเสียงที่เป็น HTTP ที่สามารถดาวน์โหลดผ่านไฟร์วอลล์ หรือเซิร์ฟเวอร์ HTTP ได้ จะต่างกับ Real-Time Transport Protocol (RTP) โดยที่ในการใช้ RTP ไคลเอนต์ และเซิร์ฟเวอร์จะต้องเชื่อมต่อ และรับส่งข้อมูลผ่านการสื่อสารแบบ real-time โดยใช้ UDP เป็นส่วนหลัก ๆ เนื่องจาก RTP ถูกออกแบบมาเพื่อการสื่อสารแบบ real-time

การทำงานของ HLS มีดังนี้

1). การแบ่งส่วนของไฟล์วิดีโอ : ไฟล์วิดีโอในรูปแบบหนึ่งถูกแบ่งออกเป็นส่วน ๆ ที่เรียกว่า "ส่วนย่อย" หรือ "ส่วนที่แยกออกมา" แต่ละส่วนนี้มีความยาวเวลาที่ต่างกัน

2). การสร้าง Playlists : มีสองประเภทของ Playlists คือ "Master Playlist" และ "Media Playlist" โดยที่ Master Playlist จะประกอบไปด้วยข้อมูลทั้งหมดเกี่ยวกับวิดีโอ หรือเสียงที่ถูกแบ่งออกมา แต่ในส่วน Media Playlist จะมีรายการของส่วนย่อย (segment) ที่ต้องการในขณะนี้

3). การส่ง Playlists และส่วนย่อยผ่าน HTTP : Playlists และส่วนย่อยถูกส่งผ่านโปรโตคอล HTTP ไปยังผู้รับบริการ

4). การเลือก Playlists ตามสถานะเครือข่าย : ตามเครือข่าย และสถานะการเชื่อมต่อผู้รับบริการสามารถเลือก Playlists ที่มีคุณภาพ และขนาดที่เหมาะสม

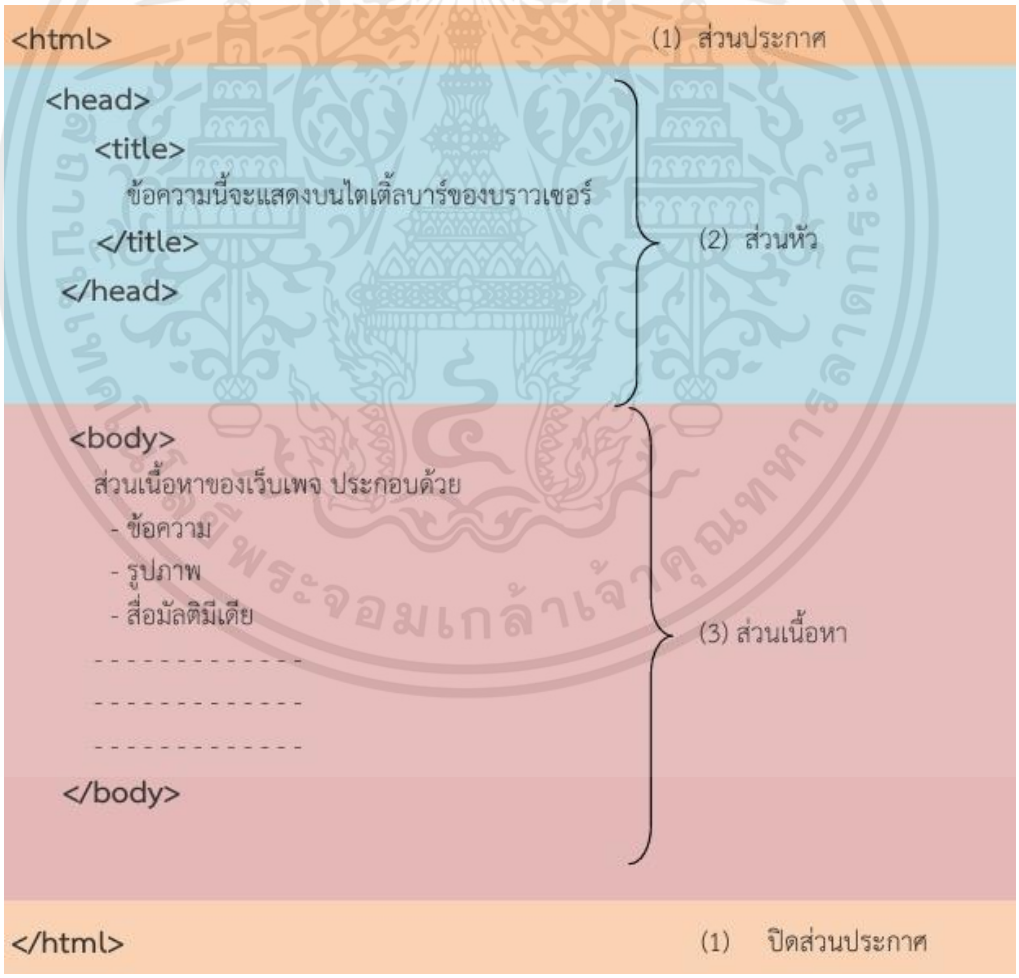
5). การเล่นสตรีมผ่านตัวเลือกที่ได้รับ : ผู้รับบริการจะดึง Playlists และส่วนย่อยที่เหมาะสมมาเล่นในอุปกรณ์ที่ต้องการ

2.12 HTML (Hyper Text Markup Language)

ภาษา HTML (Hyper Text Markup Language) เป็นภาษาที่ใช้ในการแสดงผลบนเครือข่ายอินเทอร์เน็ตในลักษณะของข้อความ รูปภาพ เสียง และภาพเคลื่อนไหวต่าง ๆ [23] ภาษา HTML เป็นภาษาที่ง่ายต่อการเรียนรู้ สามารถกำหนดรูปแบบ และโครงสร้างได้ง่าย ทำให้ได้รับความนิยม และมีการพัฒนาอย่างต่อเนื่องเพื่อให้ใช้งานง่ายขึ้น และตอบสนองต่องานด้านกราฟิกมากยิ่งขึ้น และสนับสนุนการแสดงผลในเว็บเบราว์เซอร์ และบันทึกในรูปของไฟล์นามสกุล htm หรือ html ในการพัฒนาเว็บเพจในปัจจุบันมีเครื่องมือช่วยอำนวยความสะดวกมากมาย สามารถใช้เครื่องมือพื้นฐานที่มีอยู่แล้วให้เป็นประโยชน์มากที่สุด

2.12.1 โครงสร้างหลักของภาษา HTML (Hyper Text Markup Language)

ในการเขียนภาษา HTML นั้น จะมีรูปแบบโครงสร้างการเขียนแบ่งออกเป็น 3 ส่วน โดยจะแสดงดังรูปที่ 2.13 [23]



รูปที่ 2.13 รูปแบบโครงสร้างการเขียนภาษา HTML

1). ส่วนประกาศ : เป็นส่วนที่กำหนดให้เบราว์เซอร์ทราบว่านี่คือภาษา HTML และจะต้องทำการแปลผลอย่างไร มีคำสั่งคู่เดียวคือ <html> และ </html> ปรากฏที่หัว และท้ายไฟล์

2). ส่วนหัวเรื่อง (head) : เป็นส่วนที่แสดงผลข้อความบนไตเติลบาร์ของเบราว์เซอร์ และอาจมีคำสั่งสำหรับกำหนดรายละเอียดด้านเทคนิคอื่น ๆ อีก ซึ่งแทรกอยู่ระหว่างคำสั่ง <head> และ </head>

3). ส่วนเนื้อหา (body) : เป็นส่วนที่มีความซับซ้อนมากที่สุด และสามารถใส่เทคนิคลูกเล่นเพื่อดึงดูดความสนใจจากผู้ชมได้มาก แสดงความแตกต่างระหว่างเว็บไซต์ต่าง ๆ แสดงความมีฝีมือของผู้จัดทำ ศิลปะในการออกแบบจะอยู่ในส่วนนี้ทั้งหมด ซึ่งจะแทรกอยู่ระหว่างคำสั่ง <body> และ </body>

2.12.2 การทำงานกับรูปภาพ

การสร้างเว็บเพจนั้นสามารถนำรูปภาพมาประกอบบนหน้าเว็บเพจได้ เช่น แทรกภาพในเว็บเพจ ใส่เส้นกรอบเป็นรูปภาพ และการแสดงภาพให้เป็นพื้นหลังของเว็บเพจ ชนิดของภาพที่จะนำมาประกอบบนเว็บเพจควรจะต้องมีขนาดเล็ก เพื่อนำไปเรียกใช้บนเว็บเพจได้อย่างรวดเร็ว เช่น GIF, JPEG และ PNG ซึ่งในการเลือกใช้ฟอร์แมตภาพได้อย่างเหมาะสมควรต้องทำความเข้าใจลักษณะการบีบอัดข้อมูลของแต่ละฟอร์แมต เพราะแต่ละแบบจะบีบอัดข้อมูลได้อย่างมีประสิทธิภาพสูงสุด เมื่อนำมาใช้กับภาพที่เหมาะสม

1). การแทรกรูปภาพในเว็บเพจ

จะใช้คำสั่ง <IMG SRC="ชื่อรูปภาพ"> คือ คำสั่งในการแทรกรูปภาพในหน้าเว็บเพจ โดยที่ตำแหน่งที่จัดเก็บไฟล์เว็บกับตำแหน่งไฟล์รูปภาพต้องอยู่ในโฟลเดอร์เดียวกัน

2). การจัดวางตำแหน่งของรูปภาพ

จะใช้คำสั่ง <IMG SRC = "ชื่อรูปภาพ" ALIGN="ตำแหน่ง" > โดย align=left หรือ right หรือ center หรือ top หรือ bottom เป็นการกำหนดตำแหน่งของรูปว่าจะให้อยู่ด้านซ้าย ขวา หรือตรงกลาง ส่วน top กับ bottom ใช้จัดตำแหน่งอักษร เช่น จัดรูปให้อยู่ด้านขวา <img src="sura.jpg" align=right>

2.12.3 การเชื่อมโยงหน้าเว็บเพจ

การเชื่อมโยงหน้าเว็บเพจ คือ การกำหนดส่วนของข้อความ หรือรูปภาพที่ต้องการเพื่อเป็นจุดเชื่อมโยงไปยังเนื้อหาจุดอื่น ๆ ซึ่งเนื้อหาที่จะเชื่อมโยงไปอาจจะอยู่ภายในหน้าเดียวกัน คนละหน้ากันก็ได้ หรืออาจจะอยู่คนละเว็บไซต์เลยก็ได้ การเชื่อมโยงหน้าเว็บเพจ ประกอบด้วยส่วนประกอบที่สำคัญ 2 ส่วน คือ

1). จุดที่ใช้สำหรับเชื่อมโยง (Link) : เป็นส่วนที่ใช้เป็นจุดเชื่อมโยงไปยังเป้าหมายปลายทาง ซึ่งอาจจะเป็นรูปภาพ หรือข้อความก็ได้

2). เป้าหมาย (Target) : เป็นส่วนปลายทาง หรือจุดที่ต้องการให้เชื่อมโยงมาถึง จะอยู่ส่วนใดของเอกสารก็ได้แล้วแต่ผู้เขียนเว็บไซต์จะกำหนด

2.13 Docker

Docker คือ แพลตฟอร์มซอฟต์แวร์ที่ช่วยให้สามารถสร้าง, ทดสอบ และติดตั้งแอปพลิเคชันได้อย่างรวดเร็ว [24] แสดงดังรูปที่ 2.14 [25] Docker จะบรรจุซอฟต์แวร์ลงไปในหน่วยที่เป็นมาตรฐานเรียกว่า คอนเทนเนอร์ ซึ่งจะมีทุกสิ่งที่ซอฟต์แวร์ต้องใช้ในการเรียกใช้งาน รวมทั้งไลบรารี เครื่องมือสำหรับระบบโค้ด และรันไทม์ เมื่อใช้ Docker จะสามารถติดตั้งใช้จริง และปรับขนาดแอปพลิเคชันให้เหมาะกับทุกสภาพแวดล้อม

Docker ทำงานโดยการช่วยสร้างวิธีมาตรฐานในการเรียกใช้โค้ด Docker เป็นระบบปฏิบัติการสำหรับคอนเทนเนอร์ คอนเทนเนอร์จะจำลองระบบปฏิบัติการของเซิร์ฟเวอร์ ซึ่งคล้ายคลึงกับวิธีการเครื่องเสมือนจำลอง (ลดความจำเป็นในการจัดการโดยตรง) โดยฮาร์ดแวร์ของเซิร์ฟเวอร์ Docker ได้รับการติดตั้งลงบนแต่ละเซิร์ฟเวอร์ และสร้างคำสั่งง่าย ๆ ที่สามารถใช้ในการสร้าง เริ่มต้น หรือหยุดคอนเทนเนอร์

รูปที่ 2.14 Docker

2.14 Docker Compose

Docker Compose คือ เครื่องมือ หรือคำสั่งที่จะทำให้เราสามารถรัน Docker containers หลาย ๆ คอนเทนเนอร์ได้อย่างสะดวก และง่ายในคำสั่งเดียว [26] แทนที่จะรันทีละ container ช่วยให้สามารถจัดการแอปพลิเคชันที่ประกอบด้วยหลาย services ได้อย่างสะดวก โดยกำหนดการตั้งค่าต่าง ๆ ผ่านไฟล์ docker-compose.yml แทนการใช้คำสั่ง docker run แยกทีละคอนเทนเนอร์ แสดงดังรูปที่ 2.15 [27]

โดยที่ Docker Compose จะมีข้อดีหลัก ๆ ดังนี้

1). ลดความซับซ้อน โดยแทนที่จะรันคอนเทนเนอร์ทีละตัว ก็สามารถที่จะใช้ Docker Compose ช่วยจัดการรันคอนเทนเนอร์ทั้งหมดภายในครั้งเดียวได้เลย

2). สามารถกำหนดค่าต่าง ๆ เช่น port, volume (สำหรับทำ data mapping) และรวมไปถึง Environment Variables ต่าง ๆ ได้อย่างง่ายดายภายในไฟล์ docker-compose.yml

3). สามารถทดสอบ และรันแอปพลิเคชันแบบ Multi-Containers (หลายคอนเทนเนอร์) ได้อย่างรวดเร็ว

รูปที่ 2.15 Docker Compose

2.15 JavaScript

JavaScript คือ ภาษาคอมพิวเตอร์สำหรับการเขียนโปรแกรมบนระบบอินเทอร์เน็ต [28] แสดงดังรูปที่ 2.16 [29] ที่กำลังได้รับความนิยมอย่างสูง JavaScript เป็นภาษาสคริปต์เชิงวัตถุที่เรียกกันว่า "สคริปต์" (script) ซึ่งในการสร้าง และพัฒนาเว็บไซต์ ใช้ร่วมกับ HTML เพื่อให้เว็บไซต์ดูมีการเคลื่อนไหว สามารถตอบสนองผู้ใช้งานได้มากขึ้น ซึ่งมีวิธีการทำงานในลักษณะ "แปลความ และดำเนินงานไปทีละคำสั่ง" (interpret) หรือเรียกว่า อ็อปเจ็คโอเรียลเต็ด (Object-Oriented Programming) ที่มีเป้าหมายในการออกแบบ และพัฒนาโปรแกรมในระบบอินเทอร์เน็ต สำหรับผู้เขียนด้วยภาษา HTML สามารถทำงานข้ามแพลตฟอร์มได้ โดยทำงานร่วมกับ ภาษา HTML และ JavaScript ได้ทั้งทางฝั่งไคลเอนต์ (Client) และทางฝั่งเซิร์ฟเวอร์ (Server)

โดยที่ JavaScript จะมีประโยชน์หลัก ๆ ดังนี้

1). ทำให้สามารถใช้เขียนโปรแกรมแบบง่าย ๆ ได้ โดยไม่ต้องพึ่งภาษาอื่น

2). มีคำสั่งที่ตอบสนองกับผู้ใช้งาน เช่นเมื่อผู้ใช้คลิกที่ปุ่ม หรือ Checkbox ก็สามารถสั่งให้เปิดหน้าใหม่ได้ ทำให้เว็บไซต์มีปฏิสัมพันธ์กับผู้ใช้งานมากขึ้น

3). สามารถเขียน หรือเปลี่ยนแปลง HTML Element ได้ นั่นคือสามารถเปลี่ยนแปลงรูปแบบการแสดงผลของเว็บไซต์ได้ หรือหน้าแสดงเนื้อหาสามารถซ่อน หรือแสดงเนื้อหาได้แบบง่าย ๆ

4). สามารถใช้ตรวจสอบข้อมูลได้ สังเกตว่าเมื่อเรากรอกข้อมูลบางเว็บไซต์ เช่น Email เมื่อเรากรอกข้อมูลผิดจะมีหน้าต่างแจ้งขึ้นมาว่ากรอกข้อมูลผิด หรือลืมกรอกอะไรบางอย่าง เป็นต้น

รูปที่ 2.16 JavaScript

2.16 VNC (Virtual Network Computing)

VNC หรือ Virtual Network Computing คือ การควบคุมแบบรีโมทจากคอมพิวเตอร์เครื่องอื่นในระบบเน็ตเวิร์คเดียวกัน หรือต่างกันก็ได้ [30] แสดงดังรูปที่ 2.17 [31] เพื่อใช้งานแบบ Graphic ในที่นี้จะเป็นการใช้งาน Personal Computer (คอมพิวเตอร์ส่วนบุคคล) ในการควบคุม Raspberry PI โดยการ VNC เพื่อจะได้สะดวกในการใช้งานโดยที่เราไม่จำเป็นต้องเชื่อมต่อกับอุปกรณ์อื่น ๆ เช่น เมาส์, คีย์บอร์ด และจอมอนิเตอร์ กับ Raspberry PI ก็สามารถเข้ามาควบคุมได้

ประโยชน์ของ VNC

1). File-Transfer คือ การรับ-ส่งไฟล์ต่าง ๆ

2). Text Chat คือ การสนทนาผ่านเครือข่าย

3). MS Log-on (Microsoft Log-on) คือ การเข้าสู่ระบบผ่านบัญชีผู้ใช้ของ Microsoft (MS) หรือ Active Directory (AD) เมื่อใช้ RealVNC Connect ในการเข้าระบบควบคุมระยะไกลผ่านเครือข่าย โดยสามารถเข้าสู่ระบบ VNC Server บนเครื่อง หรืออุปกรณ์ที่เชื่อมต่อกับเครือข่ายขององค์กรผ่านบัญชีผู้ใช้ของ Microsoft ได้โดยอัตโนมัติ

4). Multi Monitor Support คือ การดูแลคอมพิวเตอร์หลาย ๆ เครื่องจากจอเดียว

5). Auto Reconnect คือ การเชื่อมต่อใหม่โดยอัตโนมัติ

รูปที่ 2.17 REALVNC

2.17 Autodesk Fusion 360

Autodesk Fusion 360 คือ โปรแกรมการออกแบบ 3 มิติ [32] แสดงดังรูปที่ 2.18 [33] ที่รวมการออกแบบวิศวกรรมอิเล็กทรอนิกส์ และการผลิต ประกอบไปด้วย CAD, CAM, CAE และ PCB ไว้ในแพลตฟอร์มซอฟต์แวร์เดียวกันที่ทำงานบนระบบคลาวด์เบส สำหรับงานพัฒนาผลิตภัณฑ์ ซึ่งตลอดเวลาหลายปีที่ผ่านมา Autodesk ได้พัฒนาขีดความสามารถของ Autodesk Fusion ให้มีประสิทธิภาพ และสามารถทำงานได้หลากหลายมากยิ่งขึ้น

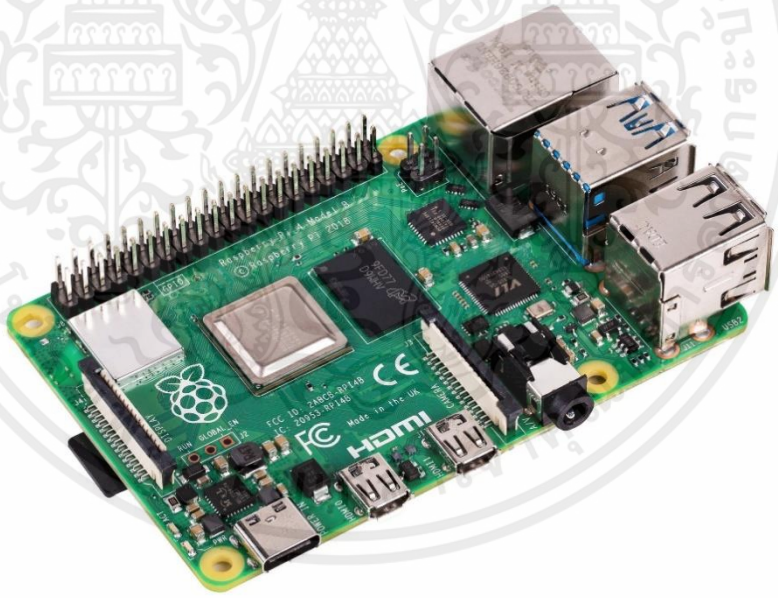
ปริญญานิพนธ์นี้ ใช้ Autodesk Fusion 360 เพื่อนำมาใช้ในการออกแบบ Packaging ในส่วนของ Hardware

รูปที่ 2.18 Autodesk Fusion 360

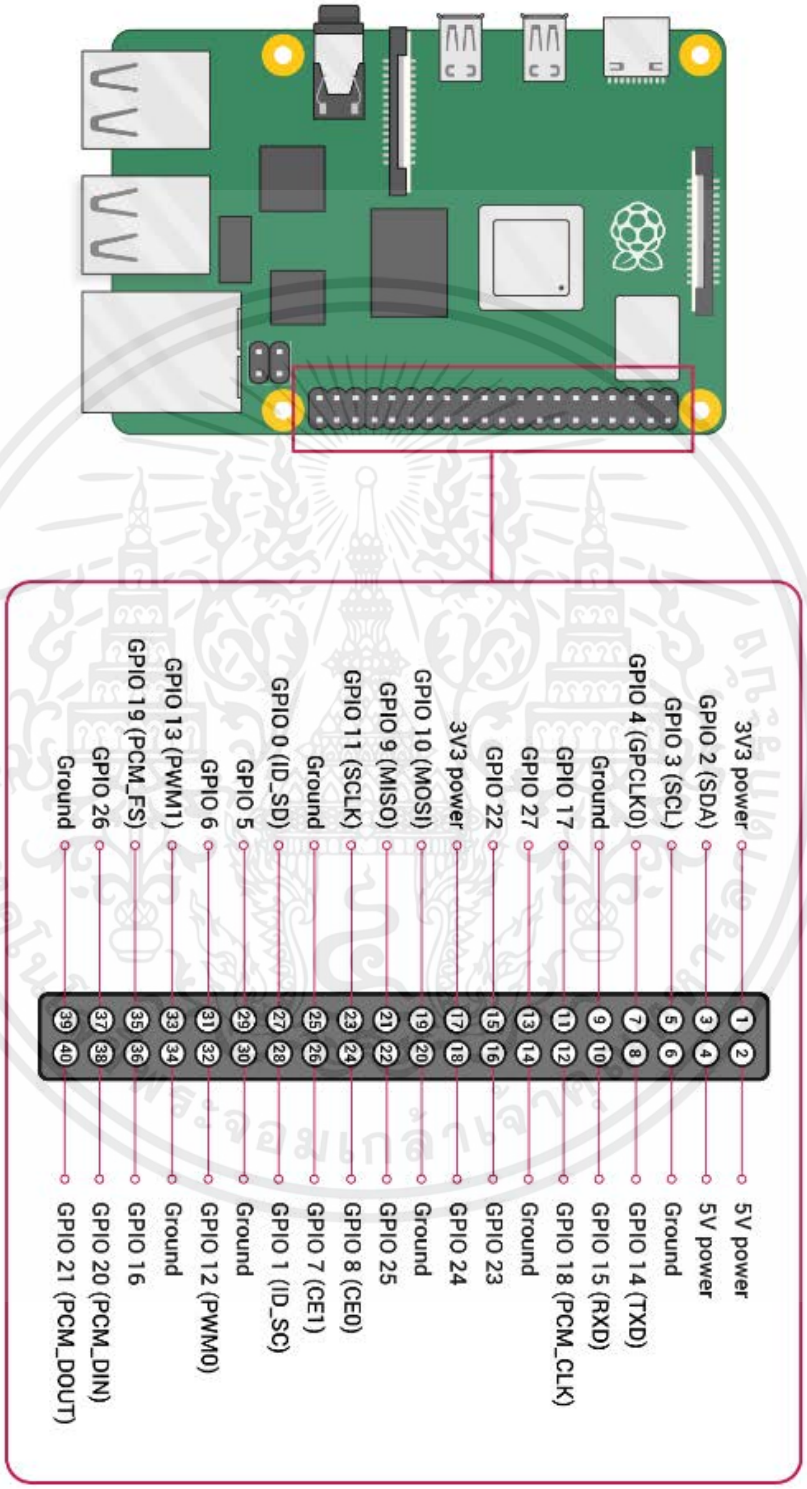
2.18 Raspberry Pi4 Model B

Raspberry Pi4 Model B คือ บอร์ดคอมพิวเตอร์เดี่ยว [34] มีความเร็วโปรเซสเซอร์ที่เพิ่มขึ้น 4 เท่า เทียบกับความเร็วของ Raspberry Pi 3 Model B+ ที่เป็นรุ่นก่อนหน้านี้ มีประสิทธิภาพการทำงานของมัลติมีเดียที่ยอดเยี่ยม พร้อมหน่วยความจำที่มากขึ้น รวมถึงการเชื่อมต่อที่พัฒนาให้ดีขึ้น Raspberry Pi4 Model B โดดเด่นด้วยโปรเซสเซอร์ 64 บิตแบบ 4 แกน ทำงานที่ความเร็ว 1.5GHz รองรับการแสดงผลแบบสองจอความคมชัดสูงสุด 4K ที่อัตรา 60 fps มีหน่วยความจำแรมสูงสุด 8GB, LAN ไร้สายที่รองรับทั้งคลื่น 2.4 และ 5.0GHz, บลูทูธ 5.0/BLE, เครือข่าย Gigabit Ethernet คือ การรองรับ หรือการทำงานที่ความเร็วถึงระดับ Gigabit Ethernet (GbE) ที่ 1 Gigabit per second (1 Gbps) และ USB 3.0 นอกจากนี้ยังมีส่วนของ Extension โดยมีการใช้งานเทคโนโลยี PoE ที่เป็นการส่งพลังงานไฟฟ้าผ่านสายเคเบิล Ethernet เพื่อให้อุปกรณ์ที่เชื่อมต่อกับเคเบิลนั้นได้รับไฟฟ้า เช่น กล้องวงจรปิด และโทรศัพท์ IP

สำหรับปริญญานิพนธ์นี้ใช้ Raspberry Pi4 Model B แสดงดังรูปที่ 2.19 [35] ซึ่งใช้ในการทำการประมวลผลข้อมูล และเป็น core หลักของ Receiver module โดยจะมี Pinout แสดงดังรูปที่ 2.20 [36]



รูปที่ 2.19 Raspberry Pi 4 Model B 8GB RAM



รูปที่ 2.20 Raspberry Pi 4 Model B Pinout

2.19 RTL-SDR

อุปกรณ์ RTL-SDR คือ อุปกรณ์รับสัญญาณวิทยุที่ใช้เทคโนโลยีที่เรียกว่า "ซอฟต์แวร์ดิไฟน์เรดิโอ" (Software-Defined Radio) [37] แสดงดังรูปที่ 2.21 [38] เพื่อที่จะมุ่งเน้นการปรับแต่งและประมวลผลสัญญาณวิทยุด้วยซอฟต์แวร์แทนการใช้ฮาร์ดแวร์ในการทำงาน การใช้ RTL-SDR นั้นช่วยลดความซับซ้อน และเป็นอุปกรณ์ที่มีความยืดหยุ่นในการรับสัญญาณวิทยุ

คุณสมบัติ และส่วนประกอบของอุปกรณ์ RTL-SDR มีดังนี้

1. ฮาร์ดแวร์ RTL2832U : RTL-SDR ใช้ชิป RTL2832U จาก Realtek ซึ่งถูกออกแบบเพื่อปรับแต่งสัญญาณ DVB-T (การรับโทรทัศน์ดิจิทัล) แต่ถูกปรับแต่งให้ใช้ในการรับสัญญาณวิทยุทั่วไป

2. อุปกรณ์ USB : RTL-SDR มีการเชื่อมต่อ USB ที่ทำให้สามารถเชื่อมต่อเข้ากับคอมพิวเตอร์ผ่านพอร์ต USB

3. Antenna Connector : RTL-SDR มักมาพร้อมกับ Antenna Connector เพื่อเชื่อมต่อกับสายอากาศ หรืออุปกรณ์ส่งสัญญาณอื่น ๆ เช่น เสาอากาศ และแท่นรับสัญญาณ

4. ซอฟต์แวร์ : ต้องใช้ซอฟต์แวร์ที่เหมาะสมบนคอมพิวเตอร์เพื่อควบคุม RTL-SDR และปรับแต่งการรับสัญญาณวิทยุ มีซอฟต์แวร์ที่เปิดใช้งานฟรีที่ใช้ร่วมกับ RTL-SDR เช่น SDR# (SDRSharp), HDSDR, GQRX, CubicSDR, และอื่น ๆ

รูปที่ 2.21 RTL-SDR RTL2832U

2.20 Telescopic Antenna

Telescopic antenna หรือเสาอากาศดึงยืด แสดงดังรูปที่ 2.22 [39] เป็นอุปกรณ์ทางไฟฟ้าที่ใช้ในการรับ หรือส่งสัญญาณวิทยุ เสาอากาศนี้มักถูกออกแบบให้สามารถยืดหดได้ตามความต้องการ ซึ่งทำให้มีความยืดหยุ่นในการใช้งาน มักจะใช้ในอุปกรณ์ต่าง ๆ เช่น วิทยุพกพา, โทรทัศน์พกพา หรืออุปกรณ์ทางไกล และอื่น ๆ ที่ต้องการรับ หรือส่งสัญญาณวิทยุ โดยมีการออกแบบลักษณะแบบท่อยาวปรับความยาวได้โดยจะสามารถรับคลื่นได้ตั้งแต่ 174MHz-230MHz เพื่อรับสัญญาณในระบบ DAB+ มีวัสดุเป็นสแตนเลส หรืออะลูมิเนียม

รูปที่ 2.22 สายอากาศวิทยุแบบยืดได้

2.21 Speaker

speaker เป็นอุปกรณ์ไฟฟ้าเชิงกลอย่างหนึ่ง [40] แสดงดังรูปที่ 2.23 [41] ทำหน้าที่แปลงสัญญาณไฟฟ้าให้เป็นเสียง มีด้วยกันหลายแบบ คำว่าลำโพงมักจะเรียกรวมกันทั้งดอกลำโพงหรือตัวขับ (driver) และลำโพงทั้งตู้ (speaker system) โดยไฟล์เสียงที่เล่นจาก Raspberry Pi เป็นดิจิทัลถือว่าคุณภาพดีสามารถเชื่อมต่อลำโพงคุณภาพสูงกับ Raspberry Pi4 ผ่าน 3.5mm jack audio output ที่มีบนบอร์ด หากต้องการประสิทธิภาพเสียงที่ดีขึ้น สามารถใช้ USB DAC (Digital-to-Analog Converter) เพื่อเพิ่มคุณภาพเสียง ในรูปที่ 2.23 เป็นเพียงตัวอย่างลำโพงคุณภาพดีที่นำมาทดลองซึ่งอาจไม่จำเป็นในฮาร์ดแวร์ตัวจริง

รูปที่ 2.23 ตัวอย่าง Speaker

2.22 Battery

แบตเตอรี่ (battery) เป็นอุปกรณ์ที่ประกอบด้วยเซลล์ไฟฟ้าเคมีหนึ่งเซลล์ หรือมากกว่าที่มีการเชื่อมต่อภายนอกเพื่อให้กำลังงานกับอุปกรณ์ไฟฟ้า [42] แสดงดังรูปที่ 2.24 [43] แบตเตอรี่มีขั้วบวก (anode) และขั้วลบ (cathode) โดยขั้วที่มีเครื่องหมายบวกจะมีพลังงานศักย์ไฟฟ้าสูงกว่าขั้วที่มีเครื่องหมายลบ ขั้วที่มีเครื่องหมายลบ คือแหล่งที่มาของอิเล็กตรอนที่เมื่อเชื่อมต่อกับวงจรภายนอก แบตเตอรี่ทำงานโดยการเกิดปฏิกิริยาเคมีระหว่างสารที่อยู่ในเซลล์ของแบตเตอรี่ เมื่อพลังงานไฟฟ้าถูกใช้งาน สารเคมีนี้จะผลิตกระแสไฟฟ้า และปล่อยพลังงานให้กับอุปกรณ์ที่ต้องการ ในขณะเดียวกัน ยังสามารถชาร์จได้โดยรับพลังงานไฟฟ้าเพื่อเก็บไว้ในแบตเตอรี่ เมื่อต้องการใช้งานในภายหลัง

สำหรับปริญญานิพนธ์นี้ใช้ Lithium Power Expansion Board for Raspberry Pi มีพารามิเตอร์ดังนี้

- ความจุแบตเตอรี่ : 3800mAh
- กระแสจำหน่ายสูงสุด : 1.8A
- โหลดแรงดันขาออก : 5.0V- 5.2V
- กระแสไฟชาร์จมาตรฐาน / แรงดันไฟฟ้า : 1.0A5.0V
- แบตเตอรี่ลิเทียมมีแรงดันไฟฟ้าอยู่ในช่วง 4.75V – 5.25V
- ตัวแบตเตอรี่สามารถชาร์จได้โดยแหล่งจ่ายไฟที่มีพอร์ต Micro USB
- สามารถจ่ายไฟให้ Raspberry Pi โดยผ่านพอร์ต USB type C

จากผลการทดสอบแรงดันไฟฟ้าของแหล่งจ่ายไฟ 5.04V เมื่อกระแสออกมากขึ้นแรงดันจะลดลง และเมื่อกระแสอยู่ที่ 1.9A Power supply นี้จะทำการ limiting protection

โดยเมื่อนำมาทดสอบร่วมกับ Raspberry Pi LCD HDMI Touch Screen Display 7 inch แล้วนั้นพบว่าสามารถใช้งานได้โดยเฉลี่ยที่ประมาณ 6 ชั่วโมง โดยจะใช้แรงดันไฟฟ้าในช่วง 5V

รูปที่ 2.24 Lithium Power Expansion Board for Raspberry Pi

2.23 Welle.io

welle.io เป็นโปรแกรมคอมพิวเตอร์ที่ใช้สำหรับรับสัญญาณวิทยุดิจิทัล (Digital Radio Signals) [44] โดยเฉพาะสำหรับสถานีวิทยุดิจิทัลด้วยมาตรฐาน DAB (Digital Audio Broadcasting) และ DAB+ (Digital Audio Broadcasting Plus) ซึ่งเป็นมาตรฐานสำหรับการถ่ายทอดเสียงแบบดิจิทัลในระบบวิทยุ แสดงดังรูปที่ 2.25 [45]

โดยที่ welle.io จะช่วยให้ผู้ใช้สามารถรับสัญญาณวิทยุดิจิทัล และฟังสถานีวิทยุดิจิทัลแบบ DAB/DAB+ บนคอมพิวเตอร์ได้ โดยใช้อุปกรณ์รับสัญญาณที่เหมาะสม เช่น RTL-SDR และซอฟต์แวร์ welle.io ในการดูแลการรับสัญญาณ

การใช้ welle.io จะช่วยให้ผู้ใช้สามารถเข้าถึงสถานีวิทยุดิจิทัลที่มีการถ่ายทอดผ่าน DAB/DAB+ และฟังเนื้อหาวิทยุในรูปแบบดิจิทัลคุณภาพสูงได้ นอกจากนี้ยังมีการพัฒนา และใช้งานกันในระบบปฏิบัติการ Linux และระบบปฏิบัติการอื่น ๆ อีกด้วย

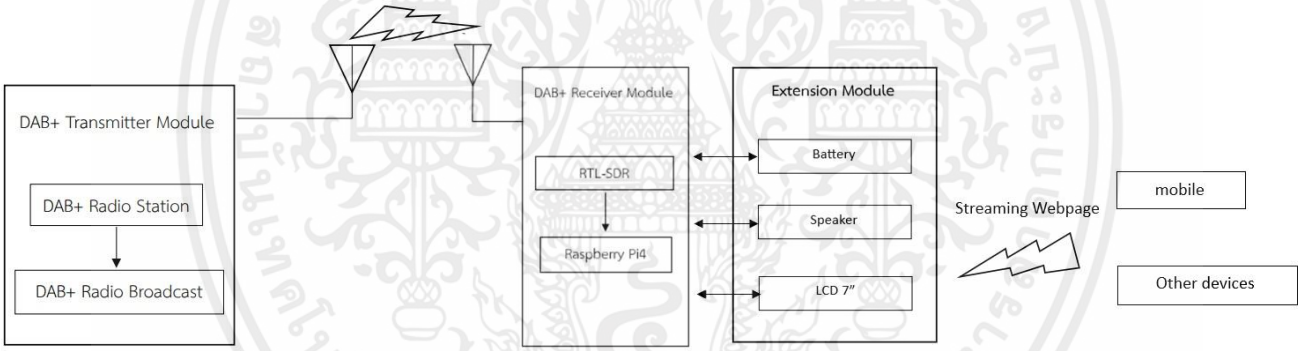
รูปที่ 2.25 โปรแกรม Welle.io

บทที่ 3

การออกแบบและการจัดทำปริญญานิพนธ์

3.1 การออกแบบ

ปริญญานิพนธ์นี้เป็นการออกแบบ และพัฒนาระบบวิทยุดิจิทัล (DAB+) เพื่อใช้ในสถานการณ์ที่เกิดเหตุภัยพิบัติ และระบบสื่อสารแบบอื่น ๆ อาจไม่สามารถใช้งานได้ โดยมีเป้าหมายหลักคือการสื่อสารที่มีประสิทธิภาพระหว่างหน่วยงานช่วยเหลือ และผู้ประสบภัยในเวลาที่เกิดภัยพิบัติ ซึ่งจะช่วยให้การสื่อสารเป็นไปอย่างต่อเนื่อง และยังสามารถใช้งานได้ในพื้นที่ประสบภัย โดยใช้เทคโนโลยีภาครับของวิทยุดิจิทัลทำงานร่วมกับโทรศัพท์เคลื่อนที่ ซึ่งหากมีภัยพิบัติที่ทำให้เข้าถึงสถานีฐานของโทรศัพท์เคลื่อนที่ไม่ได้ก็ยังใช้วิทยุดิจิทัลในการรับฟังข่าวสารต่าง ๆ และสามารถสลับสถานีแจ้งเหตุฉุกเฉินได้อัตโนมัติเมื่อได้รับสัญญาณเตือนภัย ระบบที่นำเสนอแสดงได้ดังบล็อกไดอะแกรมในรูปที่ 3.1



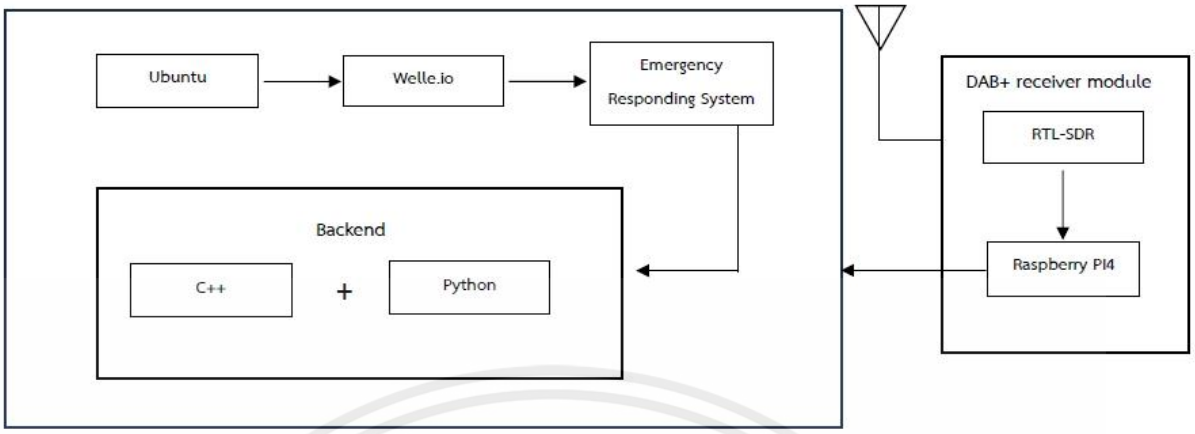
รูปที่ 3.1 บล็อกไดอะแกรมแสดงภาพรวมการทำงานของเครื่องรับวิทยุดิจิทัลกรณีเกิดเหตุฉุกเฉิน

3.1.1 การออกแบบ DAB+ Receiver Module

ในส่วนนี้จะเป็นส่วนของการออกแบบอุปกรณ์ เพื่อรับสัญญาณวิทยุดิจิทัล DAB+ โดยคำนึงถึงช่วงเวลาภัยพิบัติเป็นหลัก จึงได้แบ่งขั้นตอนการทำงานออกเป็นสองส่วนหลัก ๆ 1) Hardware 2) Software

3.1.1.1 การออกแบบ DAB+ Receiver Module (Hardware)

ในการออกแบบเราได้พิจารณาเรื่องคุณภาพเสียง และสัญญาณเป็นหลักจึงได้เลือกอุปกรณ์ประมวลผลที่ฉลาดพอจะ compile source code หรือรองรับ OS ที่ซับซ้อนได้ จึงตัดสินใจเลือกใช้ Raspberry Pi4 นำมาใช้งานประกอบกับ RTL-SDR ซึ่งแสดงดังรูปที่ 3.2



รูปที่ 3.2 บล็อกไดอะแกรมในส่วนของ DAB+ Receiver Module (Hardware)

3.1.1.2 การออกแบบ DAB+ Receiver Module (Software)

ในการออกแบบภาค Software เราได้เลือกคำสั่งรูปแบบโปรแกรมที่จะสามารถนำมาติดตั้งบน Raspberry Pi4 ได้เป็นหลักจึงได้มีการทดลองนำเอาคำสั่งจำลอง Receiver มาใช้ และพัฒนาให้รับสัญญาณได้จริงโดยอาศัยการทำงานร่วมกับ Driver SDR โดยซอฟต์แวร์ที่เลือกใช้คือ Welle.io แสดงดังรูปที่ 3.3 และ Subprogram แสดงดังภาคผนวก ก-ง

การทำงานของ Welle.io สามารถแบ่งออกเป็นขั้นตอนดังนี้

1. เชื่อมต่อฮาร์ดแวร์ SDR : ก่อนที่ Welle.io จะทำงานได้ จำเป็นต้องเชื่อมต่อ RTL-SDR กับคอมพิวเตอร์ โดยใช้พอร์ต USB หรือต่อแบบไร้สาย (Wireless) และติดตั้งไดร์เวอร์ หรือซอฟต์แวร์เสริมที่จำเป็นตามรุ่นของ RTL-SDR

2. กำหนดค่าการรับสัญญาณ : Welle.io ช่วยให้กำหนดค่าการรับสัญญาณตามความถี่ที่ต้องการ และโหมดการรับสัญญาณ (เช่น AM, FM, SSB, หรือดิจิทัล) สามารถเลือกความถี่ของสถานีวิทยุที่ต้องการรับสัญญาณ และตั้งค่าอื่น ๆ ตามความต้องการ

3. การแสดงผล : Welle.io จะแสดงผลข้อมูลที่ได้รับจากสัญญาณวิทยุดิจิทัลบนหน้าจอแสดงผลโดยแสดงคลื่นเสียง หรือข้อมูลดิจิทัลในรูปแบบที่เหมาะสมตามโหมดการรับสัญญาณที่เลือก

Welle.io Main Features

1. รองรับ DAB and DAB+

2. รองรับระบบปฏิบัติการ Windows 10/11, Linux และ macOS

3. ทำงานได้บนเครื่องประมวลผลขนาดเล็ก เช่น Raspberry Pi

4. รองรับ Airspy R2, Airspy Mini, rtl-sdr (RTL2832U), rtl_tcp, SoapySDR และ rawfile support

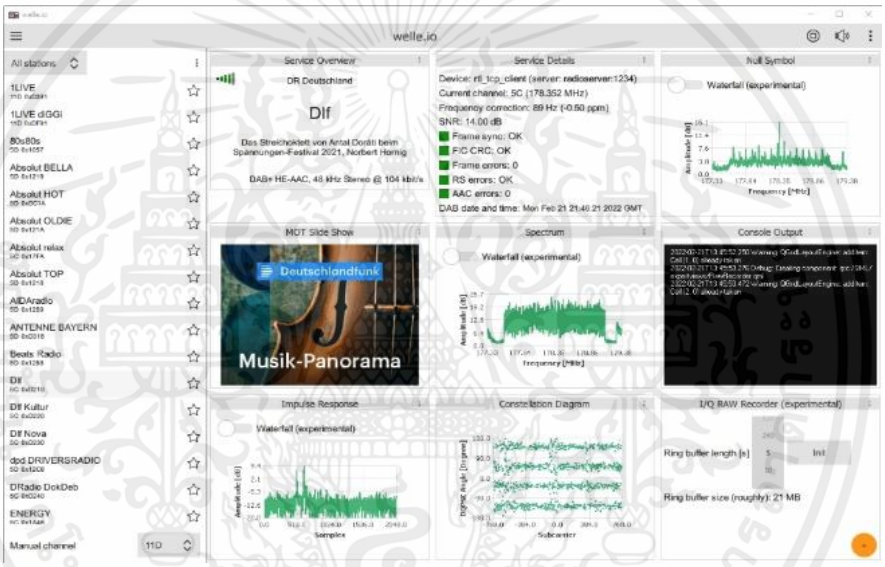
5. มีอินเทอร์เฟซกราฟฟิกที่ถูกออกแบบให้ใช้งานได้ดีกับหน้าจอที่สามารถใช้ระบบสัมผัสได้ (Touch optimized GUI)

6. มี Channel scan ที่สามารถสแกนหาช่องวิทยุที่ใช้งานได้โดยอัตโนมัติโดยไม่ต้องป้อนค่าความถี่เอง

7. Slide show (MOT slide show) เป็นระบบการถ่ายทอดที่ช่วยให้สามารถส่งข้อมูลที่มีตัวอักษรได้ และใช้สำหรับเพิ่มเนื้อหามัลติมีเดียเพิ่มเติม

8. Radio text (dynamic label) จะสามารถดูข้อมูลเพิ่มเติม หรือข้อความที่เกี่ยวข้องกับเนื้อหาที่กำลังเล่นอยู่ในขณะนั้น

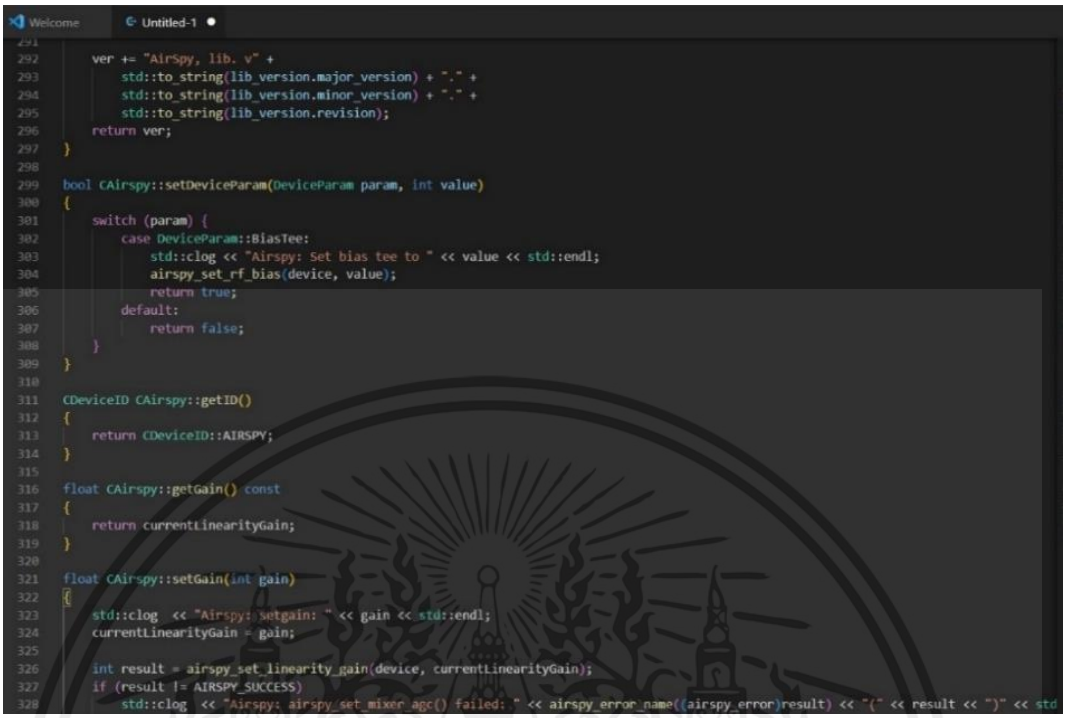
9. Expert mode สำหรับผู้ใช้ที่ต้องการการปรับแต่งการควบคุมที่ละเอียดของอุปกรณ์การรับสัญญาณที่มีการกำหนดเอง เช่น การแสดง Spectrum ของสัญญาณ



รูปที่ 3.3 ภาพหน้าต่างใช้งานของ DAB+ Receiver Module

3.1.1.2.1 ฟังก์ชันของ DAB+ Receiver Module (Software)

โดยตัว Software ที่เลือกมาติดตั้ง (Welle.io) มีการทำงานโดยรับสัญญาณจาก RTL-SDR โดยทำงานร่วมกับ Airspy ซึ่งเป็น software ที่ทำให้ตัว SDR ปรับจูนคลื่นสัญญาณได้ แสดงดังรูปที่ 3.4 และใช้ภาษา C++ สร้าง Adapter ทำให้รับข้อมูล และป้อนคำสั่งกลับไปที่ตัว Airspy ได้ โดยในหน้าต่างการใช้งานจะสามารถเลือกสถานีได้ และมีการแสดงผลภาพตามที่ได้รับสัญญาณมา ส่วนเพิ่มเติมที่เหลือจะเป็นฟังก์ชันให้มอนิเตอร์ดูองค์ประกอบของสัญญาณ ในเวลาเกิดภัยพิบัติทางคณะผู้จัดทำจะออกแบบให้สามารถสลับสถานีไปสู่ช่องทางสื่อสารฉุกเฉินตามที่มีการกำหนดในอัตโนมัติเมื่อได้รับสัญญาณเตือนภัย



รูปที่ 3.4 ภาพตัวอย่างTranscript C++ DAB+ Receiver Module

3.1.2 การออกแบบส่วน Extension module

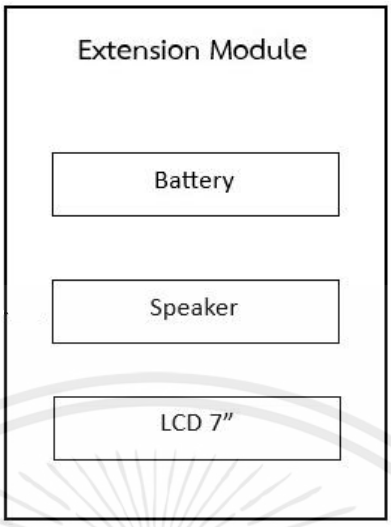
ในขั้นตอนนี้ทางคณะผู้จัดทำได้คำนึงถึงฟังก์ชันการแสดงผลของตัววิทยุเป็นหลักจึงได้ติดตั้งลำโพง Speaker จำนวน 1 คู่ และจอภาพขนาด 7 นิ้ว

ในส่วนของ Power Supply ที่จะช่วยให้วิทยุทำงานได้ในสถานการณ์ภัยพิบัติ เนื่องจาก Raspberry Pi4 ต้องการแหล่งพลังงานที่สามารถจ่ายไฟได้พอสมควร จึงได้ออกแบบมาให้สามารถพึ่งพาแหล่งจ่ายไฟได้สองทาง ดังนี้

1) สามารถใช้ไฟ DC เชื่อมต่อโดยตรงกับโมดูล DAB + ผ่านอะแดปเตอร์

2) ติดตั้งแบตเตอรี่ขนาดพกพาที่สามารถประจุไฟได้ และสามารถถอดเปลี่ยนแบตเตอรี่ได้

ในส่วนของฟังก์ชันการสื่อสารทางไกลทางคณะผู้จัดทำออกแบบให้ตัว Receiver module สามารถที่จะขยายเสียง และแสดงผลได้อย่างมีประสิทธิภาพ โดยในส่วนของ Extension module จะแสดงดังรูปที่ 3.5

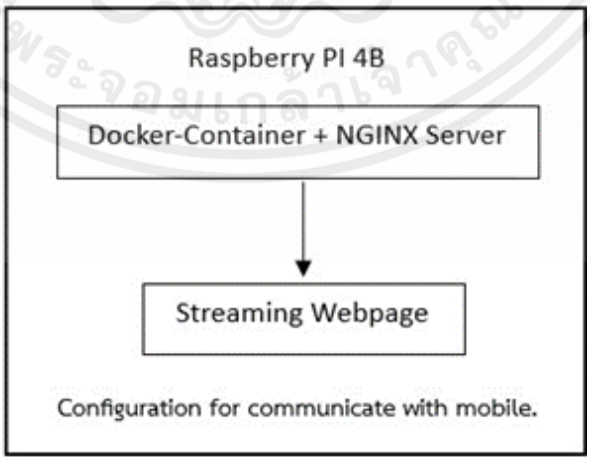


รูปที่ 3.5 บล็อกไดอะแกรมในส่วนของ Extension module

3.1.3 การออกแบบระบบติดต่อสื่อสารระหว่าง Receiver module กับ Mobile

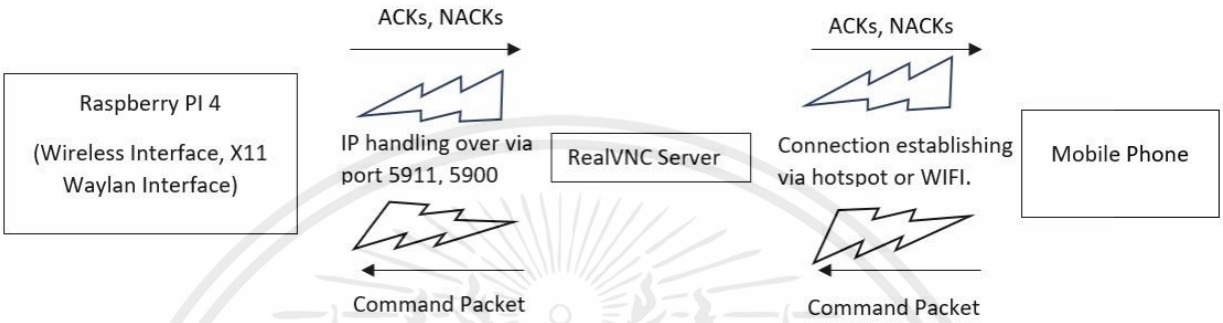
ในการออกแบบได้คำนึงถึงฟังก์ชันที่ต้องการ ดังนี้

1) สามารถแสดงผลจากตัววิทยุแบบ Real-Time ได้ โดยได้ใช้ docker container เข้ามาดึง image จาก tiangolo/nginx (ซึ่งเป็นชื่อของ Docker image ที่ถูกสร้างขึ้นโดยกลุ่ม หรือบุคคลทั่วไป) เพื่อสร้าง RTMP server ส่งผ่านข้อมูลผ่านพอร์ต 1935 และใช้ nginx.conf ในการ configuration RTMP server และส่งข้อมูลไปที่ HTTP server ผ่านพอร์ต 8080 ด้วย HLS protocol แสดงดังรูปที่ 3.6



รูปที่ 3.6 บล็อกไดอะแกรมในส่วนของการทำ Server

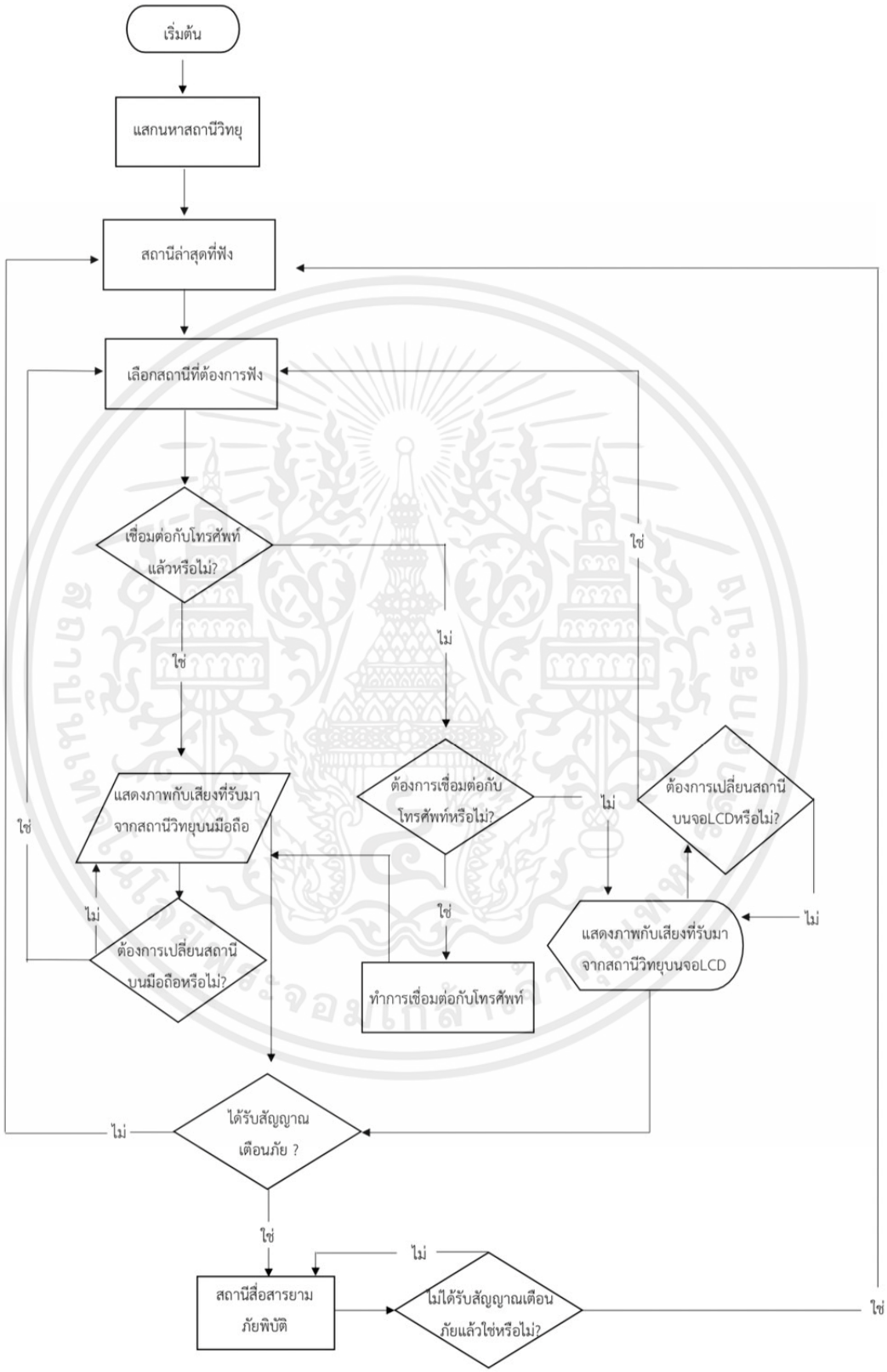
2) สามารถควบคุมเบื้องต้นผ่านทางโทรศัพท์เคลื่อนที่ เช่น เพิ่ม-ลดเสียง จึงได้นำโปรแกรม RealVNC มาใช้ ซึ่งเป็นโปรแกรมที่มีขนาดเล็ก และไม่ต้องการความสามารถในการประมวลผลที่สูงมากนัก จึงเหมาะสมกับอุปกรณ์ของระบบเป็นอย่างมาก ซึ่งสามารถแสดงผลบนจอโทรศัพท์เคลื่อนที่ และส่งคำสั่งจากโทรศัพท์เคลื่อนที่กลับมาที่ Raspberry Pi4 ได้ แสดงดังรูปที่ 3.7



รูปที่ 3.7 บล็อกไดอะแกรมการควบคุมผ่านทางโทรศัพท์เคลื่อนที่โดยใช้ RealVNC

3.1.4 สรุปภาพรวมการทำงานของระบบ

ผู้จัดทำได้ออกแบบขั้นตอนการทำงานดังนี้ เมื่อผู้ใช้งานเปิดตัววิทยุขึ้นมาจะทำการสแกนสถานี และคลื่นโดยอัตโนมัติจากนั้นให้เลือกสถานีที่ผู้ใช้อยากจะฟัง โดยจะทำการเลือกโดยสัมผัสจอ LCD ของตัววิทยุ และสามารถควบคุมเบื้องต้นได้ผ่านทางหน้าเว็บไซต์ ต่อมาในสถานการณ์ฉุกเฉินเมื่อได้รับสัญญาณเตือนภัยจากสถานีส่งจะทำการสลับสถานีไปยังสถานีฉุกเฉินตามที่กำหนดอัตโนมัติ แผนผังการทำงานโดยรวมของระบบ แสดงดังรูปที่ 3.8



รูปที่ 3.8 แผนผังแสดงการทำงานของเครื่องรับวิทยุดิจิทัลกรณีเกิดเหตุฉุกเฉิน

3.2 เครื่องมือที่ใช้ในการทดลอง

ในปริญญานิพนธ์นี้ มีอุปกรณ์ และเครื่องมือที่ใช้ในการทดลองดังนี้

3.2.1 จอแสดงผล LCD

จอภาพแบบสัมผัสสำหรับ Raspberry Pi ขนาด 7 นิ้ว มีความละเอียดการแสดงผล 1024x600 และระบบสัมผัสแบบ Capacitive Touch Screen ของ Waveshare [46] โดยตัวจอมีขนาดใหญ่กว่าตัวบอร์ด Raspberry PI ทำให้สามารถใส่ตัวบอร์ด Raspberry Pi ไว้ในเคสด้านหลังตัวจอได้สบาย ๆ โดยการเชื่อมต่อระหว่างตัวจอ และ Raspberry Pi นั้นจะใช้ high-speed SPI interface โดยสามารถใช้งานจอนี้เป็นจอแสดงผลสำหรับใช้งาน Raspberry Pi แบบพกพา

สำหรับปริญญานิพนธ์นี้ใช้ Raspberry Pi LCD HDMI Touch Screen Display 7 inch แสดงดังรูปที่ 3.9 [47] ซึ่งใช้ในการแสดงผลข้อมูล

รูปที่ 3.9 Raspberry Pi LCD HDMI Touch Screen Display 7 inch

3.2.2 Computer Laptop

สำหรับ Computer Laptop ที่ใช้ในปริญญานิพนธ์นี้ มีคุณสมบัติดังนี้

- หน่วยประมวลผลกลาง AMD Ryzen 7 3750H
- หน่วยประมวลผลกราฟฟิก NVIDIA GeForce GTX 1660Ti
- หน่วยความจำ 16.0 GB

3.3 การจัดเก็บผลการทดลอง

3.3.1 การติดตั้ง Ubuntu บน Computer Laptop

ทำการติดตั้ง Ubuntu 22.04 ลงบน Computer Laptop เพื่อใช้เป็น OS หลักสำหรับการทำปริญญานิพนธ์นี้ และเพื่อหลีกเลี่ยงการทำให้ Raspberry Pi4 เกิดการเสียหาย จึงได้ทดลองใช้บนอุปกรณ์นี้ก่อน

3.3.2 ทดลองทำการสร้าง Transmitter ที่ใช้ในการส่งสัญญาณวิทยุดิจิทัล DAB+ ลงใน Computer Laptop

ทดลองทำการสร้างเครื่องส่งสัญญาณเพื่อใช้ในการส่งสัญญาณวิทยุดิจิทัล DAB+

3.3.3 ติดตั้งโปรแกรมที่ใช้ในการรับสัญญาณวิทยุดิจิทัล DAB+ ลงใน Computer Laptop

ทำการติดตั้งโปรแกรมรับสัญญาณเพื่อใช้ในการรับสัญญาณวิทยุดิจิทัล DAB+

3.3.4 ทดลองทำการรับสัญญาณวิทยุดิจิทัล DAB+ บน Computer Laptop

ทำการทดลองรับสัญญาณบนโปรแกรมต่าง ๆ ที่ได้ติดตั้งไว้ เพื่อตัดสินใจว่าจะใช้โปรแกรม หรือคำสั่งใดในปริญญานิพนธ์นี้

3.3.5 การติดตั้ง Ubuntu บน Raspberry Pi4

ทำการติดตั้ง Ubuntu 22.04 ลงบน Raspberry Pi4 เพื่อใช้เป็น OS หลักสำหรับการทำปริญญานิพนธ์นี้

3.3.6 ติดตั้งโปรแกรม Welle.io ที่ใช้ในการรับสัญญาณวิทยุดิจิทัล DAB+ ลงใน Raspberry Pi4

ทำการติดตั้งโปรแกรม Welle.io ซึ่งตัดสินใจว่าจะเป็นโปรแกรมที่ใช้ในการรับสัญญาณวิทยุดิจิทัล DAB+ ในปริญญานิพนธ์นี้

3.3.7 ทดลองทำการรับสัญญาณวิทยุดิจิทัล DAB+ บน Raspberry Pi4

ทำการทดลองรับสัญญาณบนโปรแกรม Welle.io ที่ได้ติดตั้งไว้ บน Raspberry Pi4 เพื่อให้แน่ใจว่าจะสามารถใช้งาน และพัฒนาต่อไปได้

3.3.8 ทำการออกแบบ Protocol ใหม่สำหรับการสตรีมมิ่ง

ทำการออกแบบ RTMP server เพื่อใช้สำหรับการลำเลียงข้อมูลแบบ low-latency ด้วย open source เพื่อให้ง่ายต่อการ Implement โดย ณ ที่นี้ ได้ทำการเลือกเป็น Docker

3.3.9 ทำการออกแบบ HTTP Server ที่จะแสดงผลการสตรีมมิ่ง

ทำการออกแบบ HTTP Server เพื่อใช้สำหรับการแสดงผลการสตรีมมิ่งด้วย HLS Protocol โดย configuration ด้วย nginx.conf ส่งผ่านทาง port 8080

3.3.10 ทำการออกแบบ Frontend Webpage ที่จะเป็น portals ให้กับ devices อื่น ๆ

ทำการออกแบบหน้า webpage เพื่อรองรับ clients ที่จะเข้ามาใช้งาน welle.io

3.3.11 ทำการออกแบบระบบสำหรับการควบคุม Raspberry Pi4 ผ่านทางโทรศัพท์เคลื่อนที่ โดยการใช้ VNC Server

ทำการเชื่อมต่อ Raspberry Pi4 กับโทรศัพท์เคลื่อนที่โดยผ่าน IPV4

3.3.12 ทำการออกแบบ Emergency Alarm Method เพื่อแจ้งเตือนผู้ใช้ในยามฉุกเฉิน

ทำการออกแบบการส่งสัญญาณ Alarm โดยที่มีเงื่อนไขว่าไม่สามารถขอบันทึก หรือตัวอย่างไฟล์ DAB+ ที่รัฐบาลออกอากาศได้ จึงเขียนโปรแกรมภาษา Python base ออกมากำกับการจับภาพ MOT slideshow ของ DAB+ Station โดยกำหนดว่าเมื่อมีการเปลี่ยนแปลงของภาพ MOT slideshow จะทำการเปิดหน้าต่างการแจ้งเตือนขึ้นมา และบังคับรีบูตโปรแกรม welle.io ขึ้นมาใหม่ โดยเปลี่ยนเป็นสถานีที่เรากำหนดให้ว่าเสมือนเป็นสถานีฉุกเฉิน

3.3.13 ทำการออกแบบ Packaging สำหรับในส่วนของ Hardware

ทำการออกแบบ โดยใช้โปรแกรม Fusion 360 เพื่อให้ได้ Packaging รองรับ Hardware ทั้งหมดที่ใช้งาน

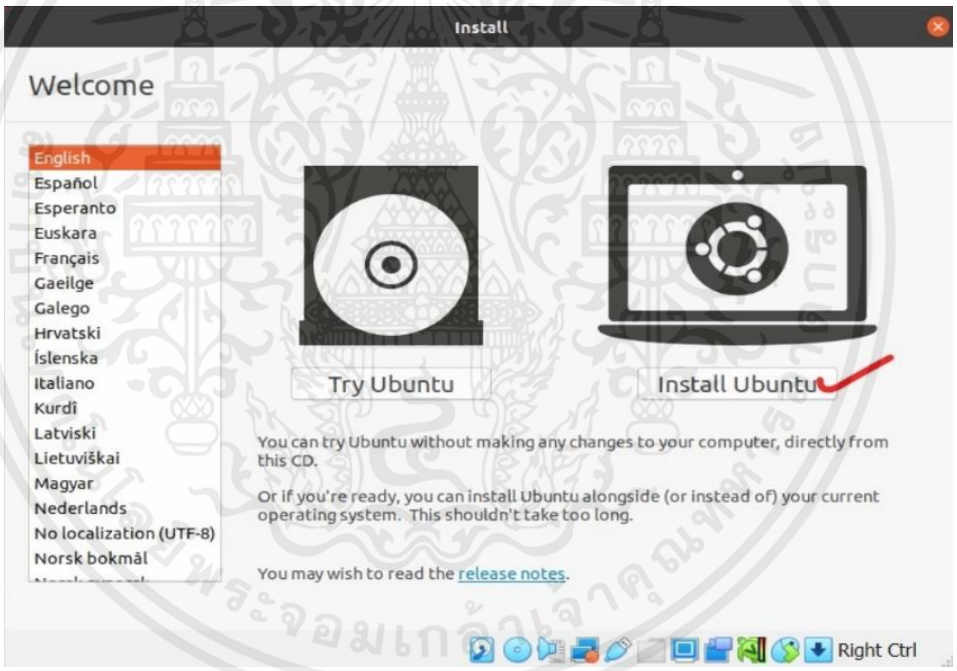
บทที่ 4

ผลการทดลอง

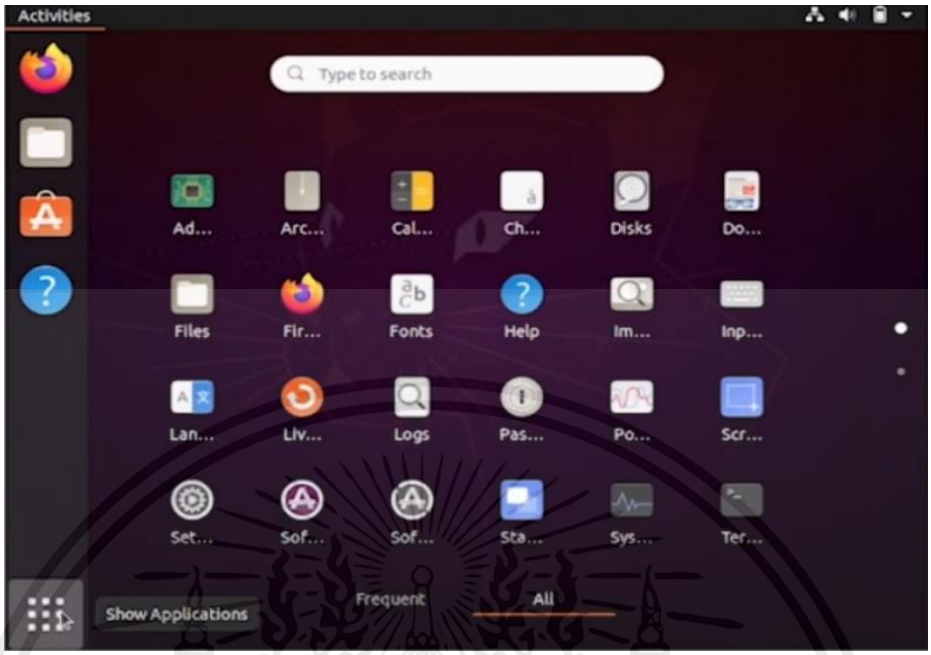
ผู้จัดทำได้ทำการเก็บผลการทำงานของระบบโดยแบ่งการทดลอง และจัดเก็บผลการทดลองเป็นส่วน ๆ ดังต่อไปนี้

4.1 การติดตั้ง Ubuntu บน Computer Laptop

ทำการติดตั้ง Ubuntu 22.04 ลงบน Computer Laptop เพื่อใช้เป็น OS หลักในการประมวลผล และเพื่อใช้ในการทดลองติดตั้งโปรแกรมที่จะใช้เป็น DAB+ Receiver Module โดยขั้นตอนระหว่างการติดตั้งจะแสดงดังรูปที่ 4.1 - 4.2



รูปที่ 4.1 ทำการติดตั้ง Ubuntu OS Version 22.04 ลงบน Computer Laptop



รูปที่ 4.2 ระบบปฏิบัติการ Ubuntu 22.04 เมื่อติดตั้งเสร็จ

4.2 ทดลองทำการสร้าง Transmitter ที่ใช้ในการส่งสัญญาณวิทยุดิจิทัล DAB+ ลงใน Computer Laptop

4.2.1 ทำการติดตั้ง ODR-AudioEnc Package บน Ubuntu

แพ็คเกจนี้ประกอบด้วยตัวเข้ารหัส DAB และ DAB+ ที่รวมเข้ากับ ODR-mmbTools ตัวเข้ารหัส DAB ขึ้นอยู่กับ toolname ตัวเข้ารหัส DAB+ ใช้ไลบรารีที่ได้รับการแก้ไขของโค้ด Fraunhofer FDK AAC จาก Android เพื่อทำการเข้ารหัสการออกอากาศ DAB+ ตัวเข้ารหัสทั้งสองเป็นส่วนหนึ่งของแพ็คเกจนี้ เครื่องมือหลักคือตัวเข้ารหัส odr-audioenc ซึ่งสามารถอ่านเสียงจากไฟล์ (raw หรือ wav) จากแหล่ง ALSA จาก JACK หรือใช้ libVLC หรือ GStreamer และเข้ารหัสเป็นไฟล์ไปป์ไปยัง EDI หรือ ZeroMQ เอาต์พุตเข้ากันได้กับ ODR-DabMux

ขั้นตอนการติดตั้ง ODR-AudioEnc Package แสดงดังรูปที่ 4.3 - 4.7

```
git clone https://github.com/opendigitalradio/dab-scripts.git
```

รูปที่ 4.3 ทำการคัดลอก Source Code DAB

```
# stable version:
git clone https://github.com/Opendigitalradio/ODR-AudioEnc.git

# or development version (at your own risk):
git clone https://github.com/Opendigitalradio/ODR-AudioEnc.git -b next
```

รูปที่ 4.4 ทำการคัดลอก Source Code ODR

```
cd ODR-AudioEnc
./bootstrap

# Select the features you need:
./configure --enable-alsa --enable-jack --enable-vlc --enable-gst
```

รูปที่ 4.5 ทำการ Config Function สำหรับ Configuration file

```
make
sudo make install
```

รูปที่ 4.6 ติดตั้ง ODR-AudioEnc

```
ALSASRC="default"
DST="tcp://yourserver:9000"
BITRATE=64
```

รูปที่ 4.7 กำหนดค่า ODR-DabMux สำหรับอินพุต EDI บนพอร์ต 9000

ทำการ Encode ไฟล์.wav ให้เป็น Format ที่ระบบ DAB+ รองรับโดยจะ save ออกมาเป็นไฟล์ station1.dabp แสดงดังรูปที่ 4.10

```
odr-audioenc -b $BITRATE -i wave_file.wav -o station1.dabp
```

รูปที่ 4.8 การเข้ารหัสไฟล์.wav สำหรับการประมวลผลที่ไม่ใช่เรียลไทม์

จากรูปที่ 4.8 ทำการแทนค่า $BITRATE ด้วย bitrate ที่ต้องการ (24 bitrate/48 kHz) และทำการแทนค่า wave_file.wav ด้วยชื่อไฟล์ที่เราได้เตรียมไว้ (testnew.wav) เมื่อทำการ Encode เสร็จสิ้นจะแสดงดังรูปที่ 4.9 - 4.10

รูปที่ 4.9 ทำการ Encode ไฟล์.wav เสร็จสิ้น

รูปที่ 4.10 ไฟล์ที่ได้จากการ Encode จะเป็นไฟล์ .dabp สำหรับเทคโนโลยีที่ DAB+

แต่เมื่อตรวจสอบโดยใช้โปรแกรม Welle.io ไม่สามารถเล่นได้เนื่องจากการ Format file .dabp ไม่ถูกต้องเป็นเหตุให้วิธีการนี้ตกไป

4.2.2 ทำการติดตั้ง ODR-PadEnc บน Ubuntu

ODR-PadEnc คือ ตัวเข้ารหัสสำหรับ Program Associated Data (PAD) และรองรับ

1. MOT Slideshow (รวมถึง catSLS)
2. DLS (รวมถึง DL Plus)

ในการเข้ารหัสข้อมูล DLS และสไลด์โชว์ ODR-Padenc เครื่องมือจะอ่านรูปภาพจากโฟลเดอร์ และข้อความ DLS จากไฟล์ และสร้างข้อมูล PAD สำหรับตัวเข้ารหัส

ขั้นตอนการติดตั้ง ODR- PadEnc แสดงดังรูปที่ 4.11 - 4. 12

```
sudo apt-get install libmagickwand-dev
```

รูปที่ 4.11 ทำการติดตั้ง libmagickwand

```
./bootstrap
./configure
make
sudo make install
```

รูปที่ 4.12 ติดตั้ง ODR-PadEnc

4.2.2.1 การใช้ MOT Slideshow และ DLS

คำสั่ง “odr-padenc” จะอ่านภาพจากโฟลเดอร์ที่ระบุ และสร้างข้อมูล PAD สำหรับตัวเข้ารหัส สิ่งนี้มีการสื่อสารผ่านซ็อกเก็ตไปยังตัวเข้ารหัส นอกจากนี้ยังอ่าน DLS จากไฟล์ และรวมข้อมูลนี้ไว้ใน PAD

หากมี ImageMagick จะสามารถอ่านรูปแบบไฟล์ทั้งหมดที่ ImageMagick รองรับ โดยค่าเริ่มต้นจะปรับขนาดเป็น 320x240 พิกเซล และบีบอัดเป็น JPEG หากไฟล์อินพุตเป็นไฟล์ JPEG ในขนาดที่ถูกต้องอยู่แล้ว และเล็กกว่า 50kB ไฟล์นั้นจะถูกส่งโดยไม่มีการบีบอัดเพิ่มเติม หากไฟล์อินพุตเป็น PNG ที่ตรงตามเกณฑ์เดียวกันไฟล์นั้นจะถูกส่งเป็น PNG โดยไม่มีการบีบอัดใหม่

Dynamic Label (DL) จะช่วยให้ผู้ให้บริการส่งข้อความพร้อมข้อมูล เช่น การเล่นเพลง ตอนนี้/ถัดไป, หัวข้อข่าว, สภาพอากาศ และผลการแข่งขันกีฬา เป็นต้น

4.3 ติดตั้งโปรแกรมที่ใช้ในการรับสัญญาณวิทยุดิจิทัล DAB+ ลงใน Computer Laptop

ทำการติดตั้งโปรแกรมรับสัญญาณเพื่อใช้ในการรับสัญญาณวิทยุดิจิทัล DAB+

4.3.1 ทำการสร้าง Console DABlin จากโปรแกรม Github บน Ubuntu

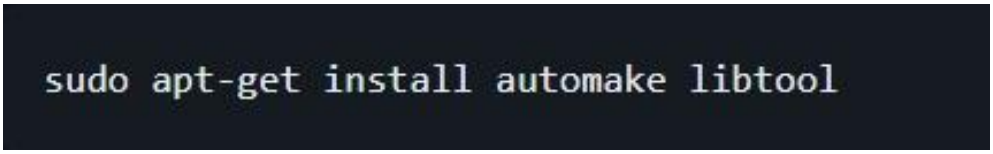
ทำการสร้าง Console Dablin ขึ้นมาเพื่อจำลอง Radio receiver ขั้นตอนระหว่างการติดตั้งจะแสดงดังรูปที่ 4.13 - 4.20

รูปที่ 4.13 การติดตั้ง Compile file

รูปที่ 4.14 การติดตั้ง Libraries

รูปที่ 4.15 การติดตั้ง library DAB+

รูปที่ 4.16 การติดตั้ง DABlin



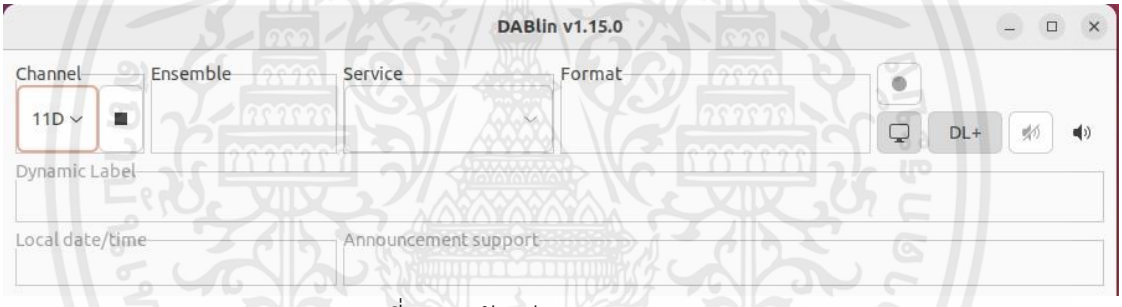
รูปที่ 4.17 การติดตั้ง Automake Libtool

```
git clone -b 2_9_2 https://github.com/knik0/faad2.git
cd faad2
./bootstrap
./configure
make
sudo make install
sudo ldconfig
```

รูปที่ 4.18 การสร้าง Configuration file เพื่อทำการ make

```
git clone https://github.com/Opendigitalradio/dablin.git
cd dablin
```

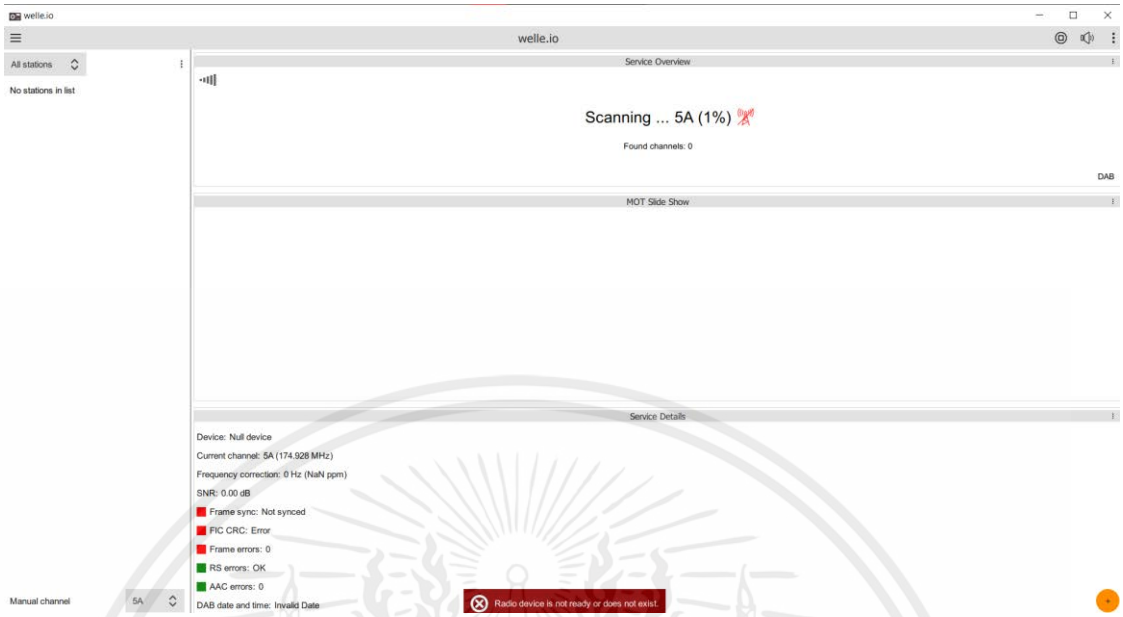
รูปที่ 4.19 ทำการติดตั้ง DABlin



รูปที่ 4.20 ตัวอย่าง Console DABlin

4.3.2 ติดตั้งโปรแกรม Welle.io บน Ubuntu

ทำการติดตั้งโปรแกรม Welle.io เพื่อทำการจำลอง Radio Receiver เมื่อติดตั้งเสร็จสิ้นและทดสอบทำการสแกนหาสัญญาณจะพบว่าไม่สามารถสแกนหาสัญญาณ หรือสถานีส่งได้ เนื่องจากยังไม่ได้ทำการเชื่อมต่อ RTL-SDR ในช่องเสียบ USB ของ Computer Laptop ซึ่งตัวโปรแกรมจะขึ้นข้อความแจ้งเตือนที่ด้านล่างของจอแสดงผลว่าตรวจไม่พบ RTL-SDR แสดงดังรูปที่ 4.21



รูปที่ 4.21 โปรแกรม Welle.io เมื่อทำการติดตั้งเสร็จ และไม่สามารถสแกนหาสัญญาณ หรือสถานีส่งได้ เนื่องจากตรวจไม่พบ RTL-SDR

4.4 ทดลองทำการรับสัญญาณวิทยุดิจิทัล DAB+ บน Computer Laptop

ทำการทดลองรับสัญญาณบนโปรแกรมต่าง ๆ ที่ได้ติดตั้งไว้

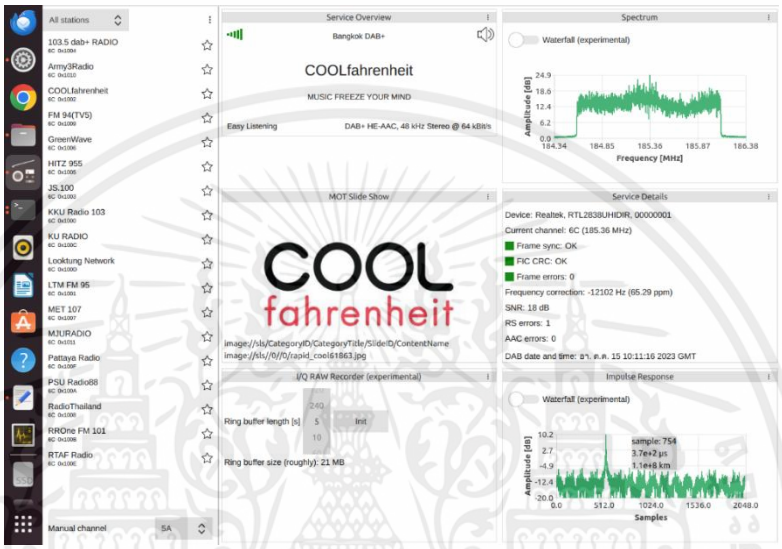
4.4.1 ทำการทดลองรับสัญญาณจาก Console DABlin บน Ubuntu

ทำการทดลองรับสัญญาณวิทยุดิจิทัล DAB+ โดยใช้ Console DABlin ที่ได้ติดตั้งไว้บน Computer Laptop ระบบปฏิบัติการ Ubuntu ผลที่ได้คือไม่สามารถรับสัญญาณวิทยุดิจิทัล DAB+ ได้จริง เนื่องจาก Console DABlin นั้นไม่สามารถเชื่อมต่อกับ SDR ได้แม้จะลองเขียน Adapter แต่ก็ไม่สามารถจูนหาอินพุตได้จึงทำให้วิธีการนี้ตกไป

4.4.2 ทำการทดลองรับสัญญาณจาก Welle.io บน Ubuntu

ทำการทดลองรับสัญญาณวิทยุดิจิทัล DAB+ โดยใช้ Welle.io ที่ได้ติดตั้งไว้บน Computer Laptop ระบบปฏิบัติการ Ubuntu ที่ใช้งานร่วมกับ RTL-SDR แล้ว โดยทำการทดสอบรับสัญญาณที่สถานีรถไฟฟ้า BTS พญาไท พิกัด ละติจูด 13.480997 ลองจิจูด 100.331427 ซึ่งอยู่ใกล้กับ Station ที่ใช้ในการออกอากาศสัญญาณ Bangkok-N1 พิกัด ละติจูด 13.790514 ลองจิจูด 100.525346 แสดงดังตารางที่2.4 ผลที่ได้คือสามารถรับสัญญาณได้จริง คุณภาพเสียงชัดเจน Pad System ทำงานได้ครบถ้วนกล่าวคือ สามารถแสดงผลชื่อสถานี, ชื่อเพลง และปกภาพอัลบั้ม โดยแสดง

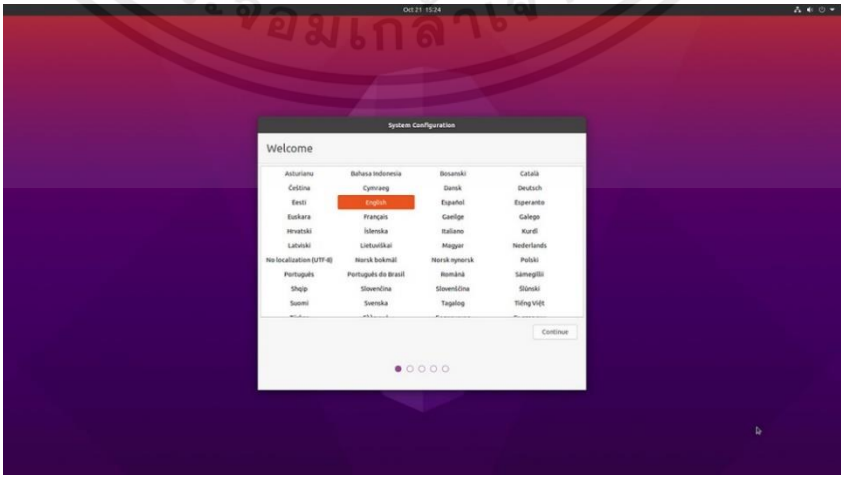
ดังรูปที่ 4.22 แต่ยังไม่สามารถบันทึกเสียงมาทำงานออฟไลน์ได้ จึงเป็นเหตุให้ต้องทดลองกับสัญญาณจริงเท่านั้นในเบื้องต้น แต่เนื่องจากในกรุงเทพฯ นั้นมีเสาส่งสัญญาณเพียงเสาเดียว และยังไม่มีการออกอากาศสัญญาณวิทยุดิจิทัล DAB+ แบบถาวร แต่อยู่ในช่วงทดสอบการออกอากาศเท่านั้น ประกอบกับช่วงเวลาการทดสอบออกอากาศที่ไม่แน่นอน จึงทำให้การทดสอบการรับสัญญาณเป็นไปได้ยาก



รูปที่ 4.22 ทดลองรับสัญญาณวิทยุดิจิทัล DAB+ โดยใช้ Welle.io ที่ได้ติดตั้งไว้บน Computer Laptop ระบบปฏิบัติการ Ubuntu

4.5 การติดตั้ง Ubuntu บน Raspberry Pi4

ทำการติดตั้ง Ubuntu 22.04 ลงบน Raspberry Pi4 เพื่อใช้เป็น OS หลักในการประมวลผล และเพื่อใช้ในการติดตั้งโปรแกรมที่จะใช้เป็น DAB+ Receiver Module โดยขั้นตอนระหว่างการติดตั้งจะแสดงดังรูปที่ 4.23



รูปที่ 4.23 การติดตั้ง Ubuntu OS Version 22.04 ลงบน Raspberry Pi4

4.5.1 ทำการติดตั้งไดร์เวอร์ LCD 7 inch ลงใน Ubuntu บน Raspberry Pi4

ทำการติดตั้งไดร์เวอร์สำหรับการใช้งานจอ LCD 7 inch บน Ubuntu ลงใน Raspberry Pi4 โดยใช้คำสั่งที่แสดงดังรูปที่ 4.24

```
sudo rm -rf LCD-show-ubuntu
git clone https://github.com/lcdwiki/LCD-show-ubuntu.git
chmod -R 755 LCD-show-ubuntu
cd LCD-show-ubuntu/

In case of 7inch HDMI Display-C-1024X600(MPI7002)
sudo ./LCD7C-show
```

รูปที่ 4.24 คำสั่งติดตั้งไดร์เวอร์จอ LCD 7 inch ลงใน Ubuntu บน Raspberry Pi4

4.6 การติดตั้งโปรแกรม Welle.io บน Raspberry Pi4

ทำการติดตั้งโปรแกรม Welle.io ซึ่งเป็นโปรแกรมที่ใช้ในการรับสัญญาณวิทยุดิจิทัล DAB+ โดยเป็นโปรแกรมที่ทดสอบบน Computer Laptop และประสบความสำเร็จรับสัญญาณได้จริง และมีขนาดโปรแกรมที่เหมาะสมกับ Raspberry Pi4

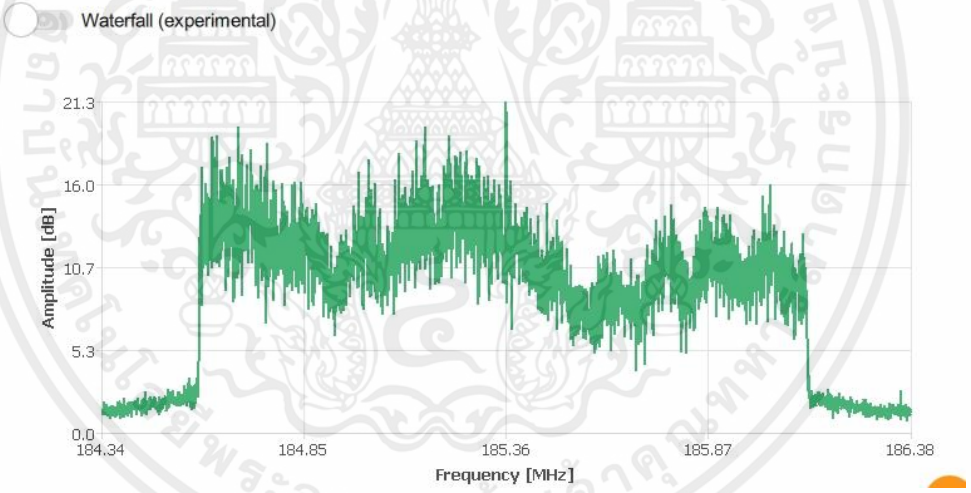
4.7 ทดลองทำการรับสัญญาณวิทยุดิจิทัล DAB+ บน Raspberry Pi4

ทำการทดลองรับสัญญาณบนโปรแกรมต่าง ๆ ที่ได้ติดตั้งไว้

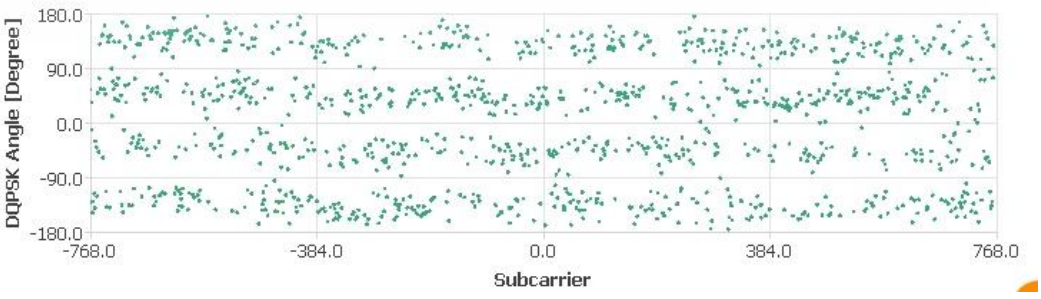
4.7.1 ทำการทดลองรับสัญญาณจาก Welle.io บน Raspberry Pi4

ทำการทดลองรับสัญญาณวิทยุดิจิทัล DAB+ โดยใช้ Welle.io ที่ได้ติดตั้งไว้บน Raspberry Pi4 ระบบปฏิบัติการ Ubuntu ที่ใช้งานร่วมกับ RTL-SDR แล้ว โดยทำการทดสอบรับสัญญาณที่ เซ็นทรัล ศรีราชา พิกัด ละติจูด 13.17915 ลองจิจูด 100.93124 ซึ่งอยู่ใกล้กับ Station ที่ใช้ในการออกอากาศสัญญาณ Si Racha-N1 พิกัด ละติจูด 13.189822 ลองจิจูด 100.950564 แสดงดังตารางที่ 2.4 ผลที่ได้คือสามารถรับสัญญาณได้จริง คุณภาพเสียงชัดเจน Pad System ทำงานได้ครบถ้วนกล่าวคือ แสดงผลชื่อสถานี, ชื่อเพลง, ปกภาพอัลบั้ม, และค่าสัญญาณต่าง ๆ ได้ แสดงดังรูปที่ 4.25 – 4.28 ที่มีเสาส่งที่มีการออกอากาศสัญญาณวิทยุดิจิทัล DAB+ ตลอดเวลา แต่ยังไม่สามารถบันทึกเสียงมาทำงานออฟไลน์ได้ จึงเป็นเหตุให้ต้องทดลองกับสัญญาณจริงเท่านั้นในเบื้องต้น

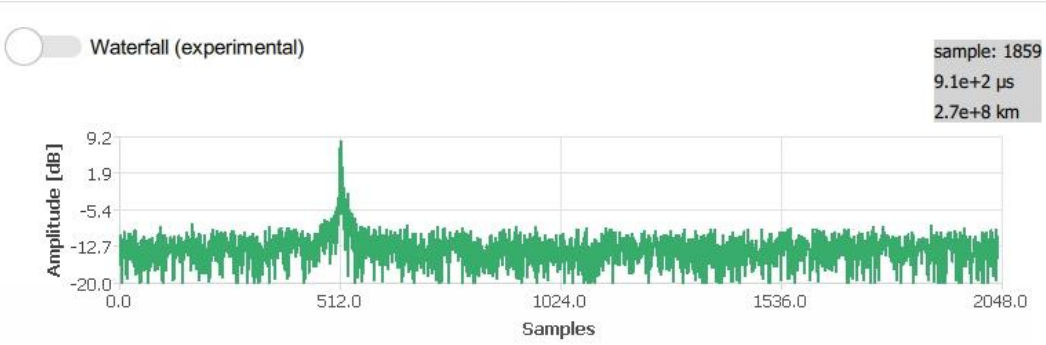
รูปที่ 4.25 ทดลองรับสัญญาณวิทยุดิจิทัล DAB+ โดยใช้ Welle.io ที่ได้ติดตั้งไว้บน Raspberry Pi4 ระบบปฏิบัติการ Ubuntu



รูปที่ 4.26 Spectrum ของช่องสัญญาณ 6C ที่ความถี่ 185.36 MHz



รูปที่ 4.27 DQPSK Constellation ของช่องสัญญาณ 6C ที่ความถี่ 185.36 MHz



รูปที่ 4.28 Impulse Response ของช่องสัญญาณ 6C ที่ความถี่ 185.36 MHz

4.8 ทำการออกแบบ Protocol ใหม่สำหรับการสตรีมมิ่ง

4.8.1 ทำการออกแบบ RTMP server

ทำการออกแบบ RTMP server เพื่อใช้สำหรับการลำเลียงข้อมูลแบบ low-latency ด้วย open source เพื่อให้ง่ายต่อการ Implement โดย ณ ที่นี้ได้ทำการเลือกเป็น Docker แสดงดังรูปที่ 4.29 – 4.32

ทำการติดตั้ง และ config server บน Docker ด้วย VS code

รูปที่ 4.29 Docker-compose.yml

ทำการ config ให้ Docker container กับ NGINX server

```
1   FROM tiangolo/nginx-rtmp
2
3   COPY nginx.conf /etc/nginx/nginx.com
```

รูปที่ 4.30 NGINX-RTMP

docker-compose.yml

C: > Users > mr.sunflower > Desktop > project > docker-compose.yml

```
1   version: "3.9"
2   services:
3     rtmp:
4       build: ./rtmp
5       ports:
6         - "1935:1935"
7         - "8080:8080"
8       container_name: rtmp_server
9       volumes:
10        - ./data:/tmp/hls
11
```

รูปที่ 4.31 NGINX Configuration

เมื่อเตรียมการทุกอย่างเสร็จสิ้นให้ใช้ Powershell ในการใช้คำสั่งให้ wsl function ทำงาน ซึ่งจะทำหน้าที่ให้ Windows OS ของเราใช้คำสั่งของ Linux OS ได้

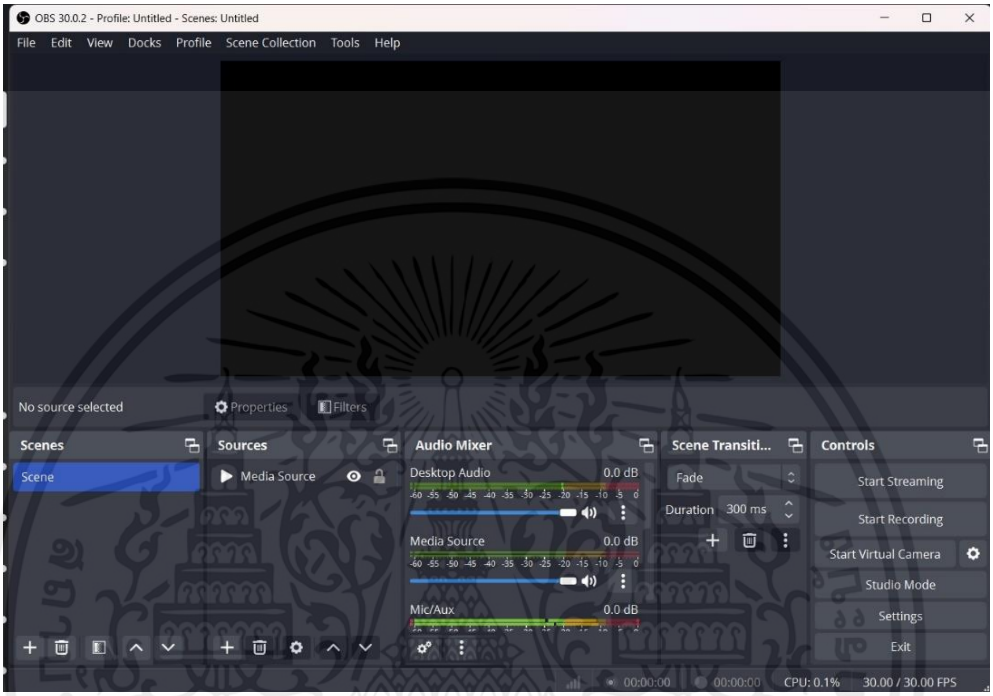
เมื่อเสร็จสิ้นก็จะกลับไปที่ไฟล์ Docker compose และใช้คำสั่ง “docker-compose build” ถ้าทำการ locate ไฟล์ได้ถูกต้อง Docker container จะทำการ deploy ขึ้นบน server ทันที และต่อมาทำการใช้คำสั่ง “docker-compose up” เพื่อ attach server container กับ RTMP Server

```
 => [rtmp internal] load metadata for docker.io/tiangolo/nginx-rtmp:latest
 => [rtmp auth] tiangolo/nginx-rtmp:pull token for registry-1.docker.io
 => [rtmp internal] load build context
 => => transferring context: 32B
 => [rtmp 1/2] FROM docker.io/tiangolo/nginx-rtmp@sha256:b1dd7f4ec79c4456755568ab824b380999e0ba0568f932296b78122c1717a83f
 => CACHED [rtmp 2/2] COPY nginx.conf /etc/nginx/nginx.com
 => [rtmp] exporting to image
 => => exporting layers
 => => writing image sha256:aa752b277069b34334a4f0759506a10f1fa3483dbf3f920965a7887b7af49042
 => => naming to docker.io/library/project-rtmp
PS C:\Users\mr.sunflower\Desktop\project> docker-compose up
[+] Running 1/0
 ✓ Container rtmp_server  Created
Attaching to rtmp_server
rtmp_server  | 172.18.0.1 [16/Jan/2024:07:13:15 +0000] PUBLISH "live" "test" "" - 1417863 529 "" "FMLE/3.0 (compatible; FMSc/1.0)" (4s)
```

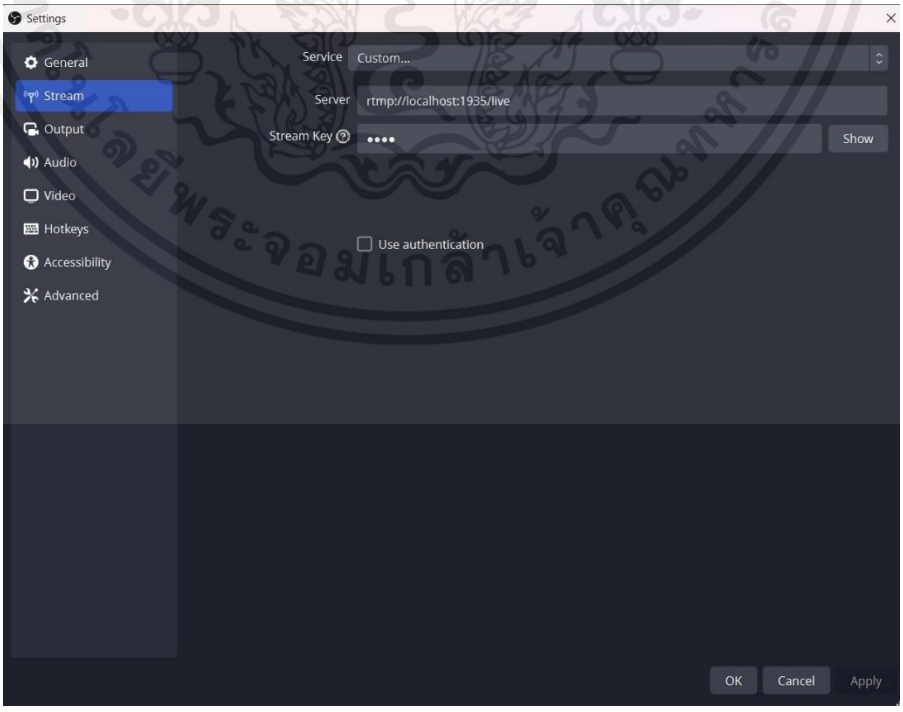
รูปที่ 4.32 Monitoring RTMP-server ด้วย Terminal

4.8.2 การสร้าง Content สำหรับการสตรีมมิ่งด้วย OBS-Studio

โดยทำการทดลองเลือก Media source เป็นคลิป และ setting ให้สตรีมมิ่งอยู่บน server “rtmp://localhost:1935/live” แสดงดังรูปที่ 4.33 – 4.37

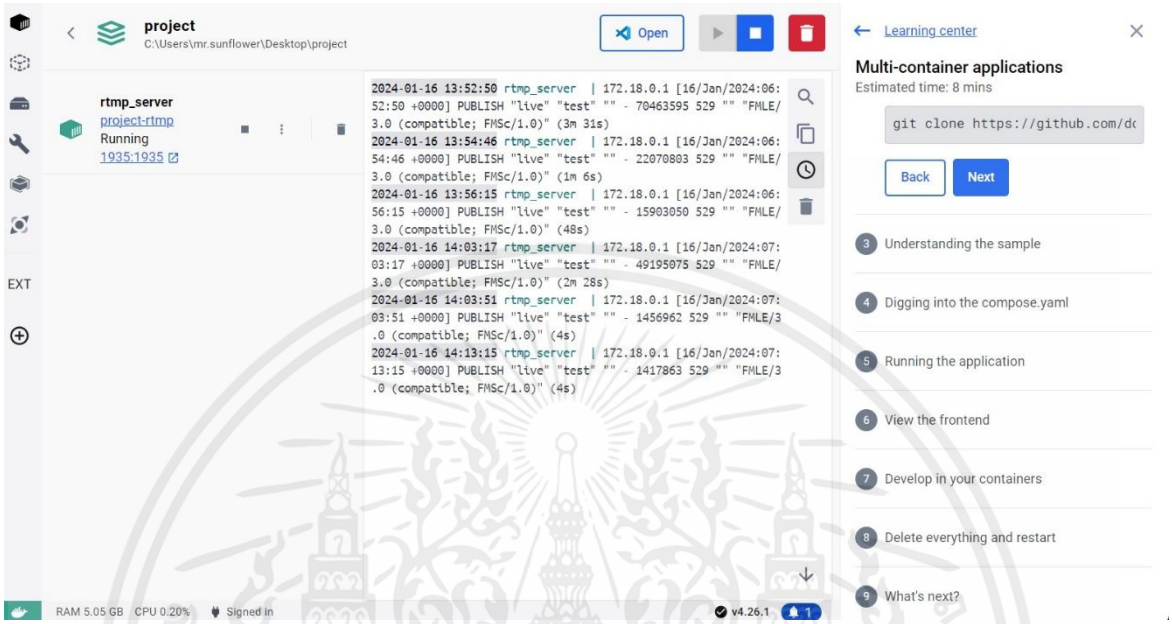


รูปที่ 4.33 OBS-Studio UI



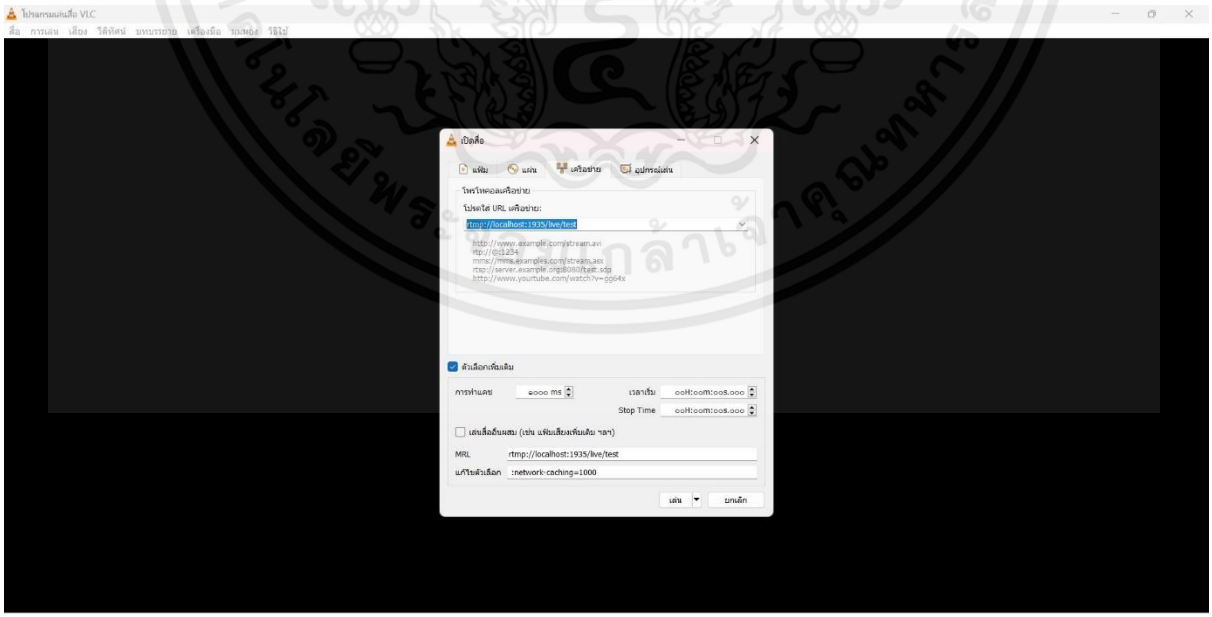
รูปที่ 4.34 การ Customize Server

เมื่อทำการ apply เสร็จสิ้นให้ทำการตรวจสอบที่ Docker container ว่ามีการสตรีมมิ่งเกิดขึ้นจริงหรือไม่



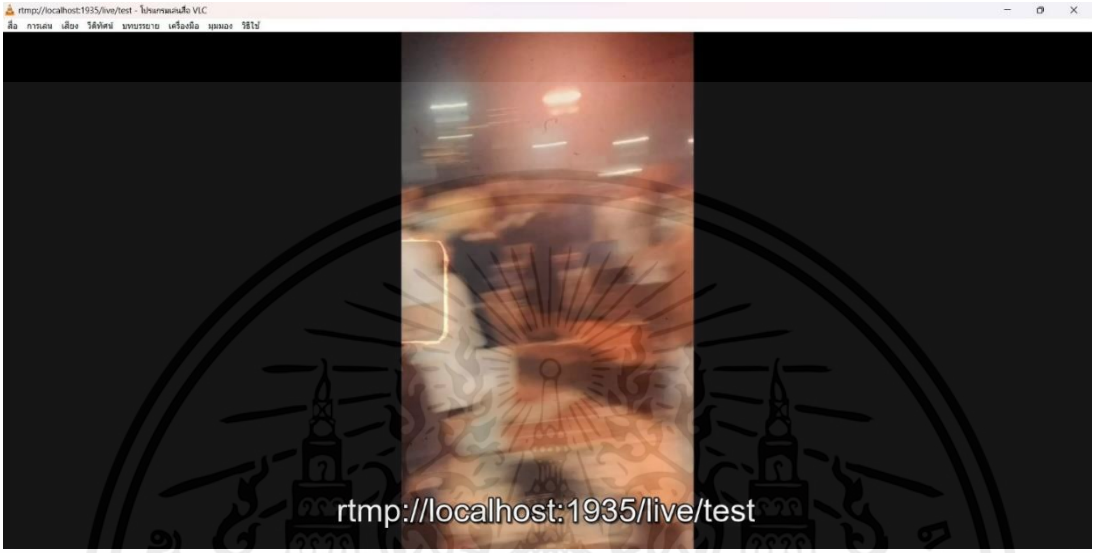
รูปที่ 4.35 Docker container

หากพบว่ามีการสตรีมมิ่งอยู่จริงจึงไปขั้นตอนต่อไป คือ การเปิดสตรีมมิ่งด้วย Media player ซึ่งคือ VLC



รูปที่ 4.36 VLC Media player

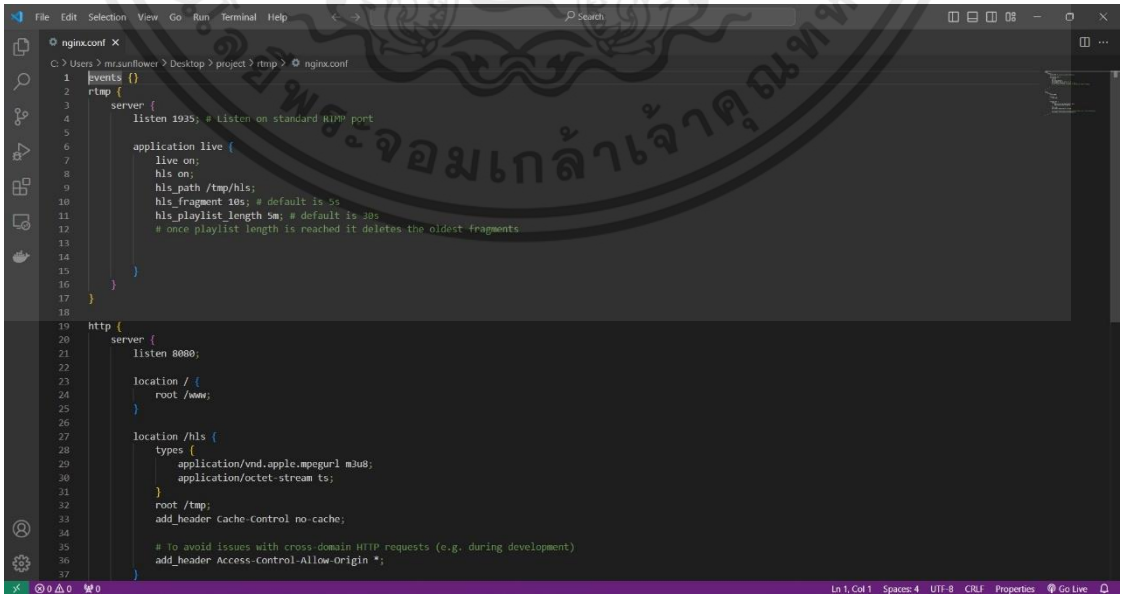
ให้ทำการเลือกสื่อผ่าน function เครือข่ายโดยจะใช้ “rtmp://localhost:1935/live” เป็น URL ของเครือข่าย ถ้าการสตรีมมิ่งทำงานถูกต้อง และ URL ถูกต้องก็จะสามารถรับชมการสตรีมมิ่งแบบ low-latency ได้



รูปที่ 4.37 การรับชมการสตรีมมิ่งแบบ low-latency

4.9 ทำการออกแบบ HTTP Server ที่จะแสดงผลการสตรีมมิ่ง

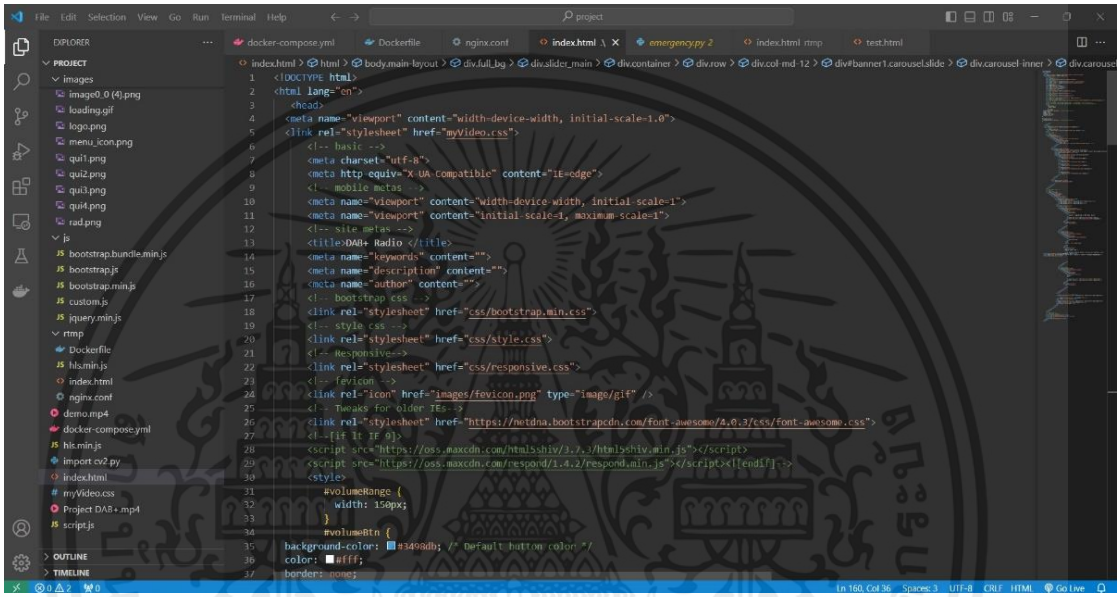
ทำการออกแบบ HTTP Server เพื่อใช้สำหรับการแสดงผลการสตรีมมิ่งด้วย HLS Protocol โดย configuration ด้วย nginx.conf ส่งผ่านทาง port 8080 แสดงดังรูปที่ 4.38



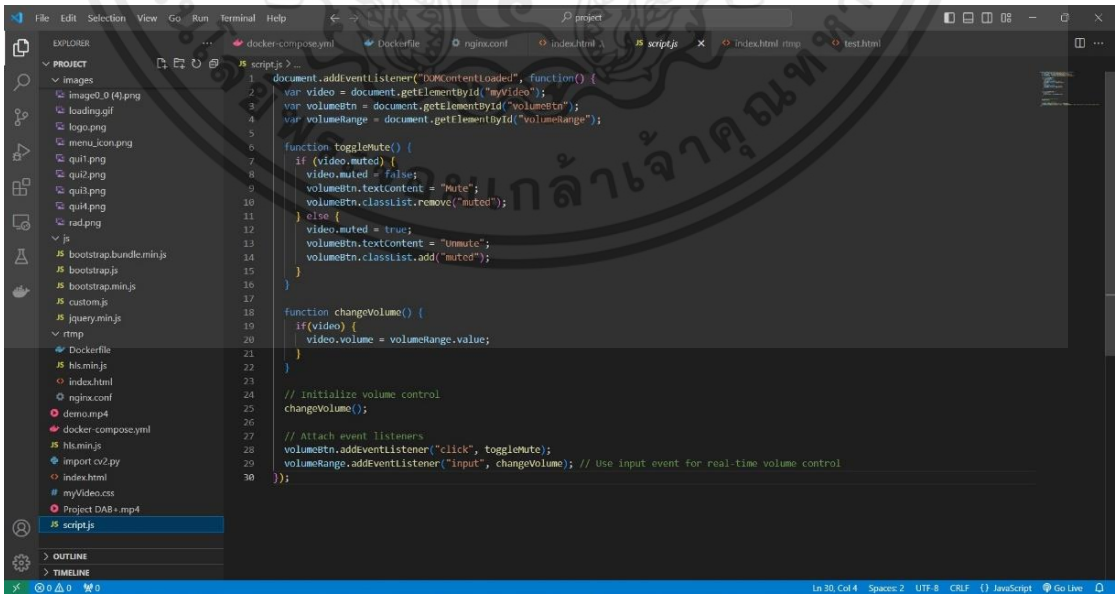
รูปที่ 4.38 nginx.conf

4.10 ทำการออกแบบ Frontend Webpage ที่จะเป็น portals ให้กับ devices อื่น ๆ

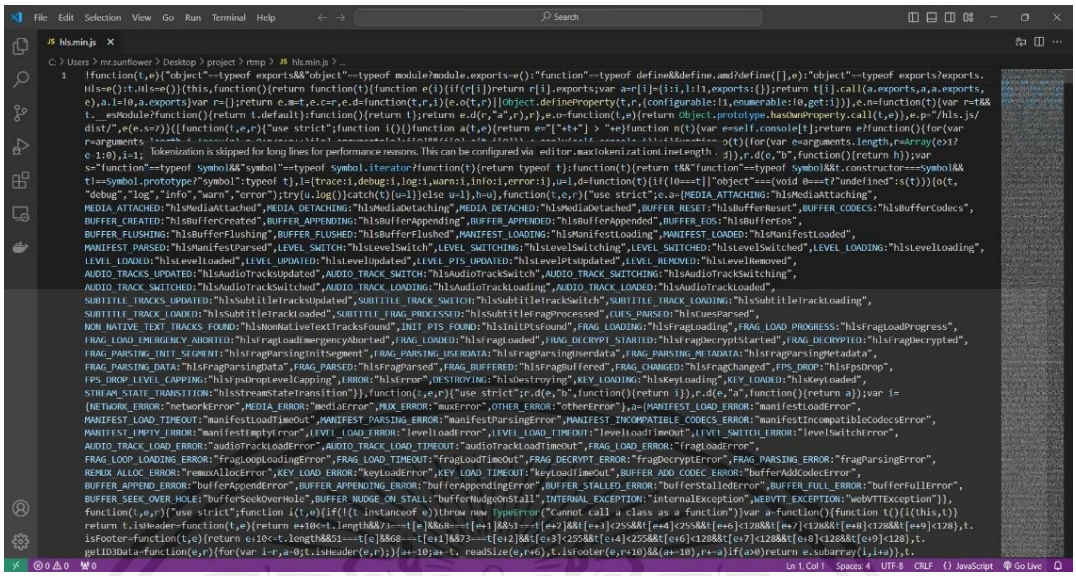
ทำการออกแบบหน้าเว็บสำหรับ clients ที่จะมาใช้งาน Welle.io โดยใช้ภาษา HTML base และใช้ CSS ในการตกแต่ง Interface และสร้างฟังก์ชันการใช้งานด้วย JavaScript และใช้งาน hls.min.js เพื่อ mapping ในการเรียกใช้ HLS protocol เพื่อดึงผลสตรีมมิ่งมาจาก RTMP Server ผ่านทาง http server port 8080 แสดงดังรูปที่ 4.39 – 4.42



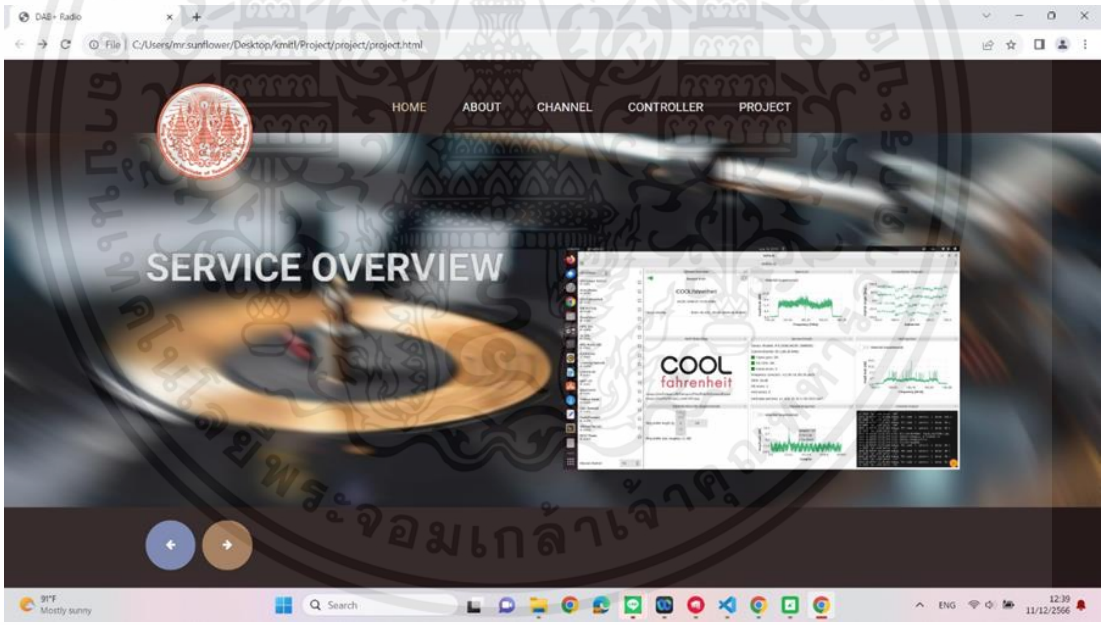
รูปที่ 4.39 HTML Script



รูปที่ 4.40 script.js (function)



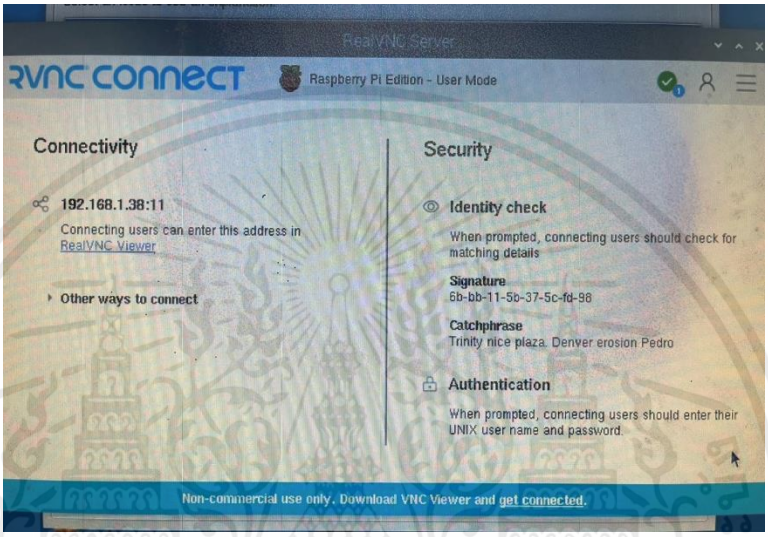
รูปที่ 4.41 hls.min.js (HLS protocol mapping)



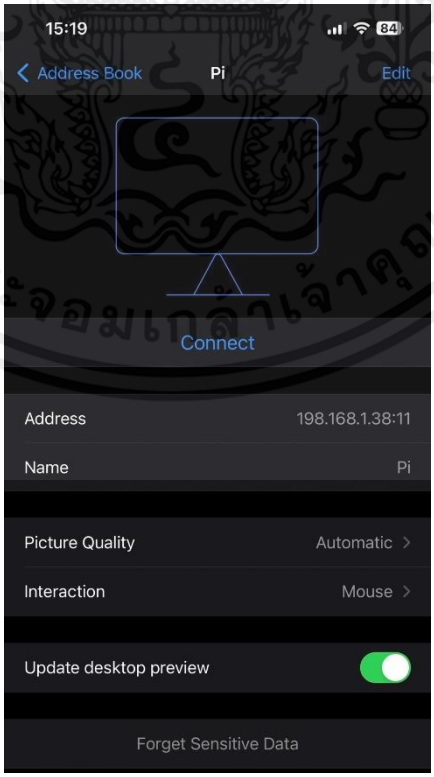
รูปที่ 4.42 หน้าต่างเว็บไซต์ที่ออกแบบสำหรับ Clients

4.11 ทำการออกแบบระบบสำหรับการควบคุม Raspberry Pi4 ผ่านทางโทรศัพท์เคลื่อนที่ โดยการใช้ VNC Server

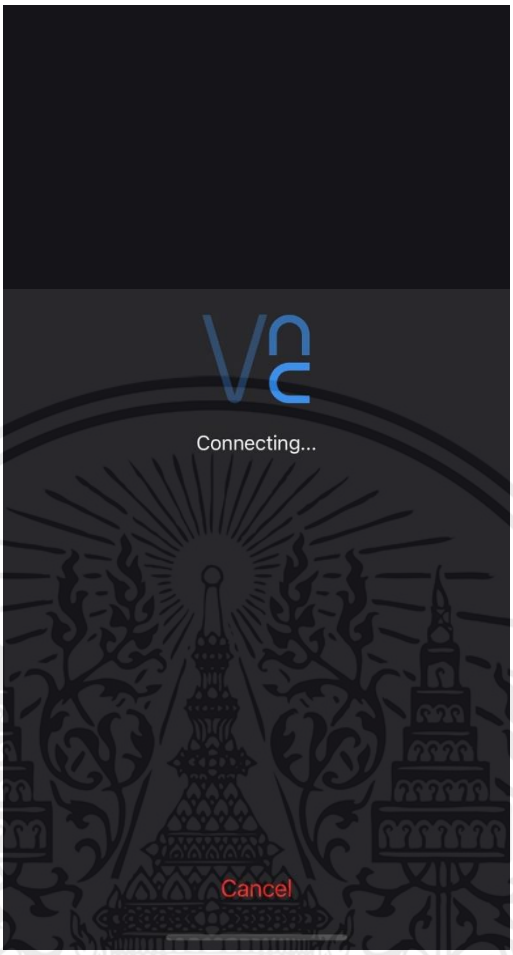
ทำการเชื่อมต่อระหว่าง Raspberry Pi4 กับโทรศัพท์เคลื่อนที่โดยผ่านทาง IPV4 แสดงดังรูปที่ 4.43 – 4.46



รูปที่ 4.43 ทำการติดตั้ง VNC จากนั้นทำการ Config IP บน Raspberry Pi4



รูปที่ 4.44 เชื่อมต่อโทรศัพท์เคลื่อนที่ด้วย IP ของ Raspberry Pi4 ที่ทำการ Config

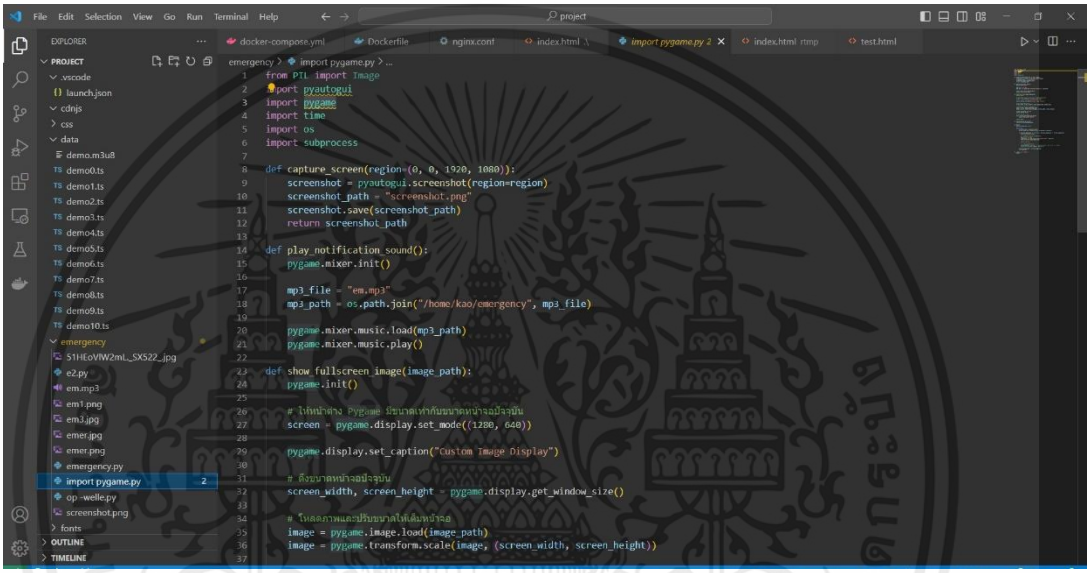


รูปที่ 4.45 ระหว่างการเชื่อมต่อ Raspberry Pi4 กับโทรศัพท์เคลื่อนที่

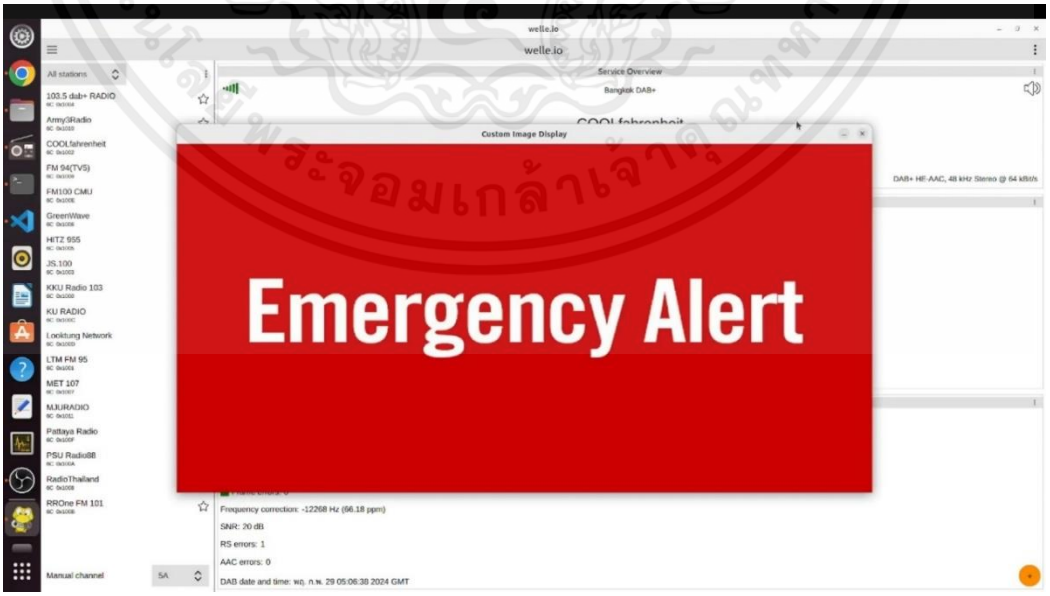
รูปที่ 4.46 ทำการเชื่อมต่อ Raspberry Pi4 กับโทรศัพท์เคลื่อนที่สำเร็จ

4.12 ทำการออกแบบ Emergency Alarm Method เพื่อแจ้งเตือนผู้ใช้ในยามฉุกเฉิน

ทำการออกแบบการส่งสัญญาณ Alarm โดยที่มีเงื่อนไขว่าไม่สามารถขอบันทึก หรือขอตัวอย่างไฟล์ DAB+ ที่รัฐบาลออกอากาศได้ จึงเขียนโปรแกรมภาษา Python base ออกมากำกับการจับภาพ MOT slideshow ของ DAB+ Station โดยกำหนดว่าเมื่อมีการเปลี่ยนแปลงของภาพ MOT slideshow จะทำการเปิดหน้าต่างการแจ้งเตือนขึ้นมา และบังคับรีบูตโปรแกรม welle.io ขึ้นมาใหม่ โดยเปลี่ยนเป็นสถานีที่เรากำหนดให้ว่าเสมือนเป็นสถานีฉุกเฉิน แสดงดังรูปที่ 4.47 – 4.48



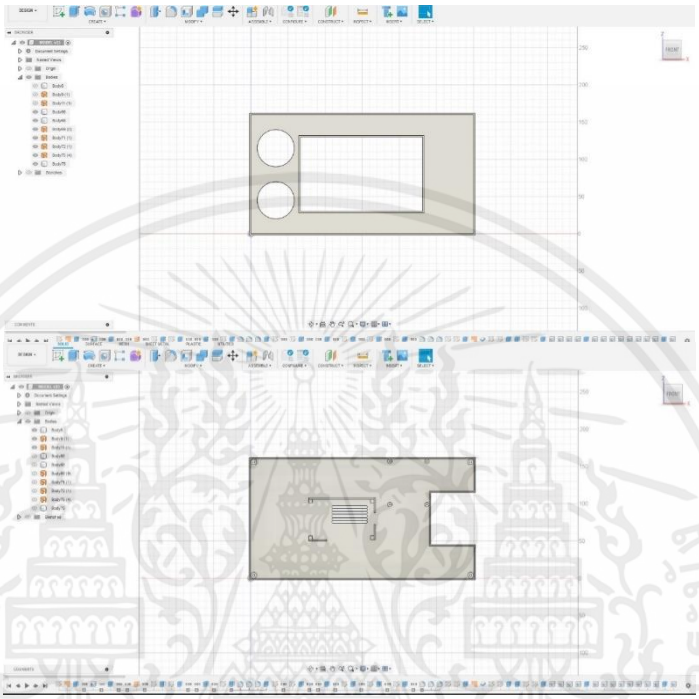
รูปที่ 4.47 Emergency Alarm Python Script



รูปที่ 4.48 Emergency Alarm Window

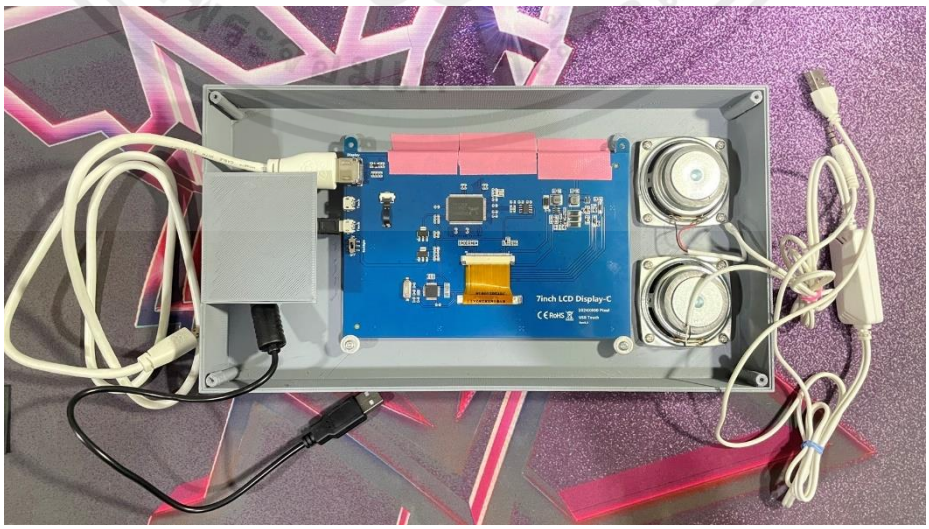
4.13 ทำการออกแบบ Packaging สำหรับในส่วนของ Hardware

ทำการออกแบบ โดยใช้โปรแกรม Fusion 360 เพื่อให้ได้ Packaging รองรับ Hardware ทั้งหมดที่ใช้งาน แสดงดังรูปที่ 4.49

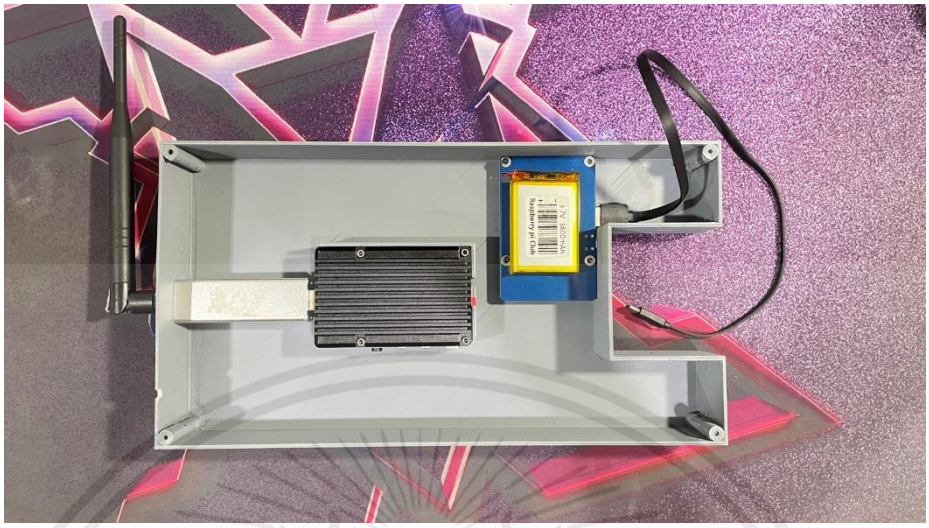


รูปที่ 4.49 การออกแบบ Packaging ในโปรแกรม Fusion 360

เมื่อได้ Packaging มาแล้วจึงทำการนำอุปกรณ์ Hardware ทั้งหมดที่ใช้งานมาติดตั้งใน Packaging แสดงดังรูปที่ 4.50 – 4.52



รูปที่ 4.50 ทำการนำอุปกรณ์ Hardware มาใส่ใน Packaging (ฝาด้านบน)



รูปที่ 4.51 ทำการนำอุปกรณ์ Hardware มาใส่ใน Packaging (ฝาด้านล่าง)

รูปที่ 4.52 เครื่องรับสัญญาณวิทยุดิจิทัล เมื่อทำการประกอบเสร็จสิ้น

บทที่ 5

สรุปผลและข้อเสนอแนะ

5.1 สรุปผล

ปริญญานิพนธ์ฉบับนี้ ประสบความสำเร็จในการศึกษาออกแบบ และพัฒนาระบบเบื้องต้นของเครื่องรับวิทยุดิจิทัลตามที่คาดการณ์ไว้ในภาคการศึกษา 2/2566 โดยสร้าง และพัฒนาอุปกรณ์ในการรับสัญญาณ และถอดรหัสวิทยุดิจิทัลด้วย Raspberry Pi4 ประกอบกับการใช้ RTL-SDR ในการรับสัญญาณ และใช้โปรแกรมซึ่งมีพื้นฐานโครงสร้างเป็นภาษา C++ และมีการติดตั้ง Extension Module ที่ประกอบด้วย Speaker, จอLCD 7 นิ้ว และ Battery เพื่อเพิ่มประสิทธิภาพการแสดงผลของ Raspberry Pi4 โดยได้พัฒนา Webpage สำหรับ clients ที่จะใช้งาน Welle.io ด้วย devices ของตัวเอง และพัฒนาระบบ Emergency Alarm ภายใต้เงื่อนไขที่กำหนดเนื่องจากไม่สามารถเก็บบันทึก Packet ที่ออกอากาศโดยรัฐบาล โดยได้เก็บผลการทดลองตั้งแต่เริ่มต้นระบบ (เปิดใช้งาน device) จนถึงสิ้นสุด (สิ้นสุดการส่งสัญญาณ Alarm) สรุปได้ว่าในส่วนของฝั่งรับนั้นพร้อมแล้วที่จะใช้งานรองรับกลุ่ม clients ขนาดเล็ก และให้บริการ DAB+ ด้วย Welle.io ผ่านทาง Webpage รวมถึงมีระบบ Emergency Alarm ในตัว

5.2 ข้อเสนอแนะ

ในปริญญานิพนธ์ฉบับนี้รับสัญญาณจาก DAB+ ยังไม่เสถียรมากนัก เนื่องจากการแพร่สัญญาณวิทยุดิจิทัลในประเทศไทยยังไม่เป็นทางการในหลายเขต กล่าวคือมีช่วงเวลาออกอากาศที่จำกัด สถานีที่ออกอากาศมีไม่มาก และบริเวณของสัญญาณยังไม่ครอบคลุมหลายแห่ง ตัวอย่างเช่น กรุงเทพฯ จึงทำให้หากทดลองรับสัญญาณด้วยอุปกรณ์อาจไม่พบสถานี

บรรณานุกรม

[1] ISO/IEC 11172-3 [ออนไลน์]. เข้าถึงได้จาก https://csclub.uwaterloo.ca/~pbarfuss/ISO11172-3.pdf

[2] ISO/IEC 13818-3 [ออนไลน์]. เข้าถึงได้จาก https://webstore.iec.ch/preview/info_isoiec13818-3%7Bed2.0%7Den.pdf

[3] MPEG-4 HE AAC v2 [ออนไลน์]. เข้าถึงได้จาก https://tech.ebu.ch/docs/techreview/trev_305-moser.pdf

[4] บล็อกไดอะแกรมระบบส่งของวิทยุกระจายเสียงระบบดิจิทัล DAB [ออนไลน์]. เข้าถึงได้จาก https://broadcast.nbtc.go.th/data/academic/file/610600000001.pdf

[5] แผนภาพบล็อกความถี่วิทยุ ความถี่วิทยุกึ่งกลาง ความกว้างแถบคลื่นความถี่ และความกว้างแถบคลื่นความถี่ป้องกัน [ออนไลน์]. เข้าถึงได้จาก https://www.nbtc.go.th/getattachment/News/publichearing/47045/%E0%B8%A3%E0%B9%88%E0%B8%B2%E0%B8%87_%E0%B8%9B%E0%B8%A3%E0%B8%B0%E0%B8%81%E0%B8%B2%E0%B8%A8%E0%B8%AF.pdf.aspx?lang=th-TH&page=hsn

[6] ETSI EN 300 401 V2.1.1 (2017-01) [ออนไลน์]. เข้าถึงได้จาก https://www.etsi.org/deliver/etsi_en/300400_300499/300401/02.01.01_60/en_300401v020101p.pdf

[7] ETSI TS 102 563 v1.2.1 (2010-05) [ออนไลน์]. เข้าถึงได้จาก https://www.etsi.org/deliver/etsi_ts/102500_102599/102563/01.02.01_60/ts_102563v010201p.pdf

[8] ITU-R BS.1660-8 (06/2019) [ออนไลน์]. เข้าถึงได้จาก https://lokaaldigitaal.vlaanderen/onewebmedia/R-REC-BS.1660-8-201906-I%21%21PDF-E.pdf

[9] ITU-R BS.638 (1986) [ออนไลน์]. เข้าถึงได้จาก https://www.itu.int/dms_pubrec/itu-r/rec/bs/R-REC-BS.638-0-198607-I!!PDF-E.pdf

[10] ระดับของภัยพิบัติ และระยะเวลาในการแจ้งเตือนภัยพิบัติ [ออนไลน์]. เข้าถึงได้จาก https://www.worlddab.org/public_document/file/1451/DAB__Emergency_warning_factsheet.pdf?1635785305

[11] RDS-Signal in the FM Spectrum [ออนไลน์]. เข้าถึงได้จาก https://www.prismsound.com/test_measure/support_subs/apps/fm_mpx.php

[12] แผนความถี่วิทยุกิจการกระจายเสียงระบบเอฟเอ็ม (สำนักงาน กสทช. 4 พฤศจิกายน พ.ศ. 2564) [ออนไลน์]. เข้าถึงได้จาก https://www.nbtc.go.th/getattachment//law/law_noti/nbtc_notification/%E0%B9%81%E0%B8%9C%E0%B8%99%E0%B8%84%E0%B8%A7%E0%B8%B2%E0%B9%81%E0%B8%9C%E0%B8%99%E0%B8%84%E0%B8%A7%E0%B8%B2%E0%B8%A1%E0%B8%96%E0%B8%B5%E0%B9%88%E0%B8%A7%E0%B8%B4%E0%B8%97%E0%B8%A2%E0%B8%B8%E0%B8%81%E0%B8%B4%E0%B8%88%E0%B8%81%E0%B8%B2%E0%B8%A3%E0%B8%81%E0%B8%A3%E0%B8%B0%E0%B8%88%E0%B8%B2%E0%B8%A2%E0%B9%80%E0%B8%AA%E0%B8%B5%E0%B8%A2%E0%B8%87%E0%B8%A3%E0%B8%B0%E0%B8%9A%E0%B8%9A%E0%B9%80%E0%B8%AD%E0%B8%9F%E0%B9%80%E0%B8%AD%E0%B9%87%E0%B8%A1/%E0%B8%9B%E0%B8%A3%E0%B8%B0%E0%B8%81%E0%B8%B2%E0%B8%A1/%E0%B8%9B%E0%B8%A3%E0%B8%B0%E0%B8%81%E0%B8%B2%E0%B8%A8%E0%B8%AF.PDF?lang=th-TH

[13] ระบบปฏิบัติการ Linux [ออนไลน์]. เข้าถึงได้จาก https://www.mindphp.com/

[14] ระบบปฏิบัติการ Ubuntu [ออนไลน์]. เข้าถึงได้จาก https://blog.openlandscape.cloud/ubuntu

[15] สัญลักษณ์ ระบบปฏิบัติการ Ubuntu [ออนไลน์]. เข้าถึงได้จาก https://www.neowin.net/news/canonical-updates-the-ubuntu-logo-in-time-for-2204-lts/

[16] Python [ออนไลน์]. เข้าถึงได้จาก https://www.9experttraining.com/articles/python

[17] สัญลักษณ์ Python [ออนไลน์]. เข้าถึงได้จาก https://www.9experttraining.com/articles/python

[18] ภาษา C++ [ออนไลน์]. เข้าถึงได้จาก https://th.wikipedia.org/wiki/%E0%B8%8B%E0%B8%B5%2B%2B

[19] สัญลักษณ์ ภาษา C++ [ออนไลน์]. เข้าถึงได้จาก https://fastwork.co/user/omslxmsc/tutoring-74003456

[20] IEEE 802.11 มาตรฐานการทำงานของระบบเครือข่ายไร้สาย (WIFI) [ออนไลน์]. เข้าถึงได้จาก https://iiot.riverplus.com/ieee-802-11/

[21] RTMP (Real-Time Messaging Protocol) [ออนไลน์]. เข้าถึงได้จาก https://chat.openai.com/share/06734a0a-cdaf-4e53-9ca7-51a6326a2378

[22] HLS (HTTP Live Streaming) [ออนไลน์]. เข้าถึงได้จาก https://hmong.in.th/wiki/HTTP_Live_Streaming

[23] HTML (Hyper Text Markup Language) [ออนไลน์]. เข้าถึงได้จาก https://academic.udru.ac.th/~samawan/content/HTML1.pdf

[24] Docker [ออนไลน์]. เข้าถึงได้จาก https://aws.amazon.com/th/docker/

[25] Docker [ออนไลน์]. เข้าถึงได้จาก https://www.novelbiz.co.th/what-is-docker/

[26] Docker Compose [ออนไลน์]. เข้าถึงได้จาก https://devhub.in.th/learn/docker/docker-compose

[27] Docker Compose [ออนไลน์]. เข้าถึงได้จาก https://medium.com/@jigarrathod2704/docker-compose-for-beginners-a-comprehensive-guide-29113e49f7da

[28] JavaScript [ออนไลน์]. เข้าถึงได้จาก https://www.mindphp.com/%E0%B8%84%E0%B8%B9%E0%B9%88%E0%B8%A1%E0%B8%B7%E0%B8%AD/73-%E0%B8%84%E0%B8%B7%E0%B8%B8%AD%E0%B8%B0%E0%B9%84%E0%B8%A3/2187-java-javascript-%E0%B8%84%E0%B8%B7%E0%B8%AD%E0%B8%AD%E0%B8%B0%E0%B9%84%E0%B8%A3.html

[29] JavaScript [ออนไลน์]. เข้าถึงได้จาก https://www.blockdit.com/posts/61bcae1768a0c32eec38e372

[30] VNC (Virtual Network Computing) [ออนไลน์]. เข้าถึงได้จาก https://595162020009.wixsite.com/iot-edu/chapter04

[31] Real VNC [ออนไลน์]. เข้าถึงได้จาก https://www.itpro.com/mobile/remote-access/368067/vnc-connect-review

[32] Autodesk Fusion 360 [ออนไลน์]. เข้าถึงได้จาก https://synergysoft.co.th/special-article/25-mfg-special-articles/456-fusion-360-2022

[33] Autodesk Fusion 360 [ออนไลน์]. เข้าถึงได้จาก https://forums.autodesk.com/t5/fusion-support/fusion-360-logo-for-download/td-p/7435770

[34] Raspberry Pi 4 Model B [ออนไลน์]. เข้าถึงได้จาก https://inex.co.th/home/product/raspberry-pi-4-4gb-ram/

[35] อุปกรณ์ Raspberry Pi 4 Model B [ออนไลน์]. เข้าถึงได้จาก https://th.element14.com/raspberry-pi/rpi4-modbp-8gb/raspberry-pi-4-model-b-cortex/dp/3369503

[36] Raspberry Pi 4 Model B Pinout [ออนไลน์]. เข้าถึงได้จาก https://www.raspberrypi.com/documentation/computers/raspberry-pi.html

[37] RTL-SDR [ออนไลน์]. เข้าถึงได้จาก https://th.fmuser.org/news/Audio/What-is-SDR.html

[38] RTL-SDR RTL2832U [ออนไลน์]. เข้าถึงได้จาก https://kamami.pl/en/dvb-t-tuners/583578-rtl-sdr-rtl2832u-sdr-500khz-1766mhz-radio-receiver.html

[39] สายอากาศวิทยุแบบยืดได้ [ออนไลน์]. เข้าถึงได้จาก https://th.aliexpress.com/i/1005005477003508.html

[40] Speaker [ออนไลน์]. เข้าถึงได้จาก https://th.wikipedia.org/wiki/%E0%B8%A5%E0%B8%B3%E0%B9%82%E0%B8%9E%E0%B8%87

[41] ตัวอย่าง Speaker [ออนไลน์]. เข้าถึงได้จาก https://www.remaxthailand.co.th/product/SpeakerNS-044-NUBWO-10367

[42] แบตเตอรี่ (battery) [ออนไลน์]. เข้าถึงได้จาก https://th.wikipedia.org/wiki/%E0%B9%81%E0%B8%9A%E0%B8%95%E0%B9%80%E0%B8%95%E0%B8%AD%E0%B8%A3%E0%B8%B5%E0%B9%88

[43] Lithium Power Expansion Board for Raspberry Pi [ออนไลน์]. เข้าถึงได้จาก http://www.iot.codemobiles.com/product/313/lithium-battery-expansion-board-for-raspberry-pi

[44] Welle.io [ออนไลน์]. เข้าถึงได้จาก https://www.welle.io/

[45] สัญลักษณ์ Welle.io [ออนไลน์]. เข้าถึงได้จาก https://www.welle.io/

[46] Raspberry Pi LCD HDMI Touch Screen Display 7 inch [ออนไลน์]. เข้าถึงได้จาก https://www.cybertice.com/product/3719/%E0%B8%88%E0%B8%AD-lcd-7-%E0%B8%99%E0%B8%B4%E0%B9%89%E0%B8%A7-hdmi-touch-screen-display-%E0%B8%AA%E0%B8%B3%E0%B8%AB%E0%B8%A3%E0%B8%B1%E0%B8%9A-raspberry-pi-2-3-4-%E0%B8%84%E0%B8%A7%E0%B8%B2%E0%B8%A1%E0%B8%A5%E0%B8%B0%E0%B9%80%E0%B8%AD%E0%%E0%B8%B5%E0%B8%A2%E0%B8%94- 1024x600?gad _source=1&gclid=Cj0KCQiA84CvBhCaARIsAMkAvkK3BmL4GpyNf90AXOln9bjdynpUfMP4spJansoV6n_Fj_o_1Vts5ToaAjwaEALw_wcB

[47] อุปกรณ์ Raspberry Pi LCD HDMI Touch Screen Display 7 inch [ออนไลน์]. เข้าถึงได้จาก http://wiki.sunfounder.cc/index.php?title=7_Inch_Screen_Manual

ภาคผนวก ก

โปรแกรม Welle.io
Subprogram Radio Receiver

```
 */

#include <string>
#include <iostream>
#include <memory>
#include "radio-receiver.h"

using namespace std;

const char* fftPlacementMethodToString(FFTPlacementMethod
fft_placement)
{
    switch (fft_placement) {
        case FFTPlacementMethod::EarliestPeakWithBinning:
            return "EarliestPeakWithBinning";
        case FFTPlacementMethod::StrongestPeak:
            return "StrongestPeak";
        case FFTPlacementMethod::ThresholdBeforePeak:
            return "ThresholdBeforePeak";
    }
    throw std::logic_error("Unhandled fft placement");
}

const char* freqSyncMethodToString(FreqsyncMethod method)
{
    switch (method) {
        case FreqsyncMethod::CorrelatePRS:
            return "CorrelatePRS";
        case FreqsyncMethod::GetMiddle:
            return "GetMiddle";
        case FreqsyncMethod::PatternOfZeros:
            return "PatternOfZeros";
    }
    throw std::logic_error("Unhandled freqsyncMethod placement");
}
```

```
RadioReceiver::RadioReceiver(
                RadioControllerInterface& rci,
                InputInterface& input,
                RadioReceiverOptions rro,
                int transmission_mode) :
    params(transmission_mode),
    mscHandler(params, false),
    ficHandler(rci),
    ofdmProcessor(input,
        params,
        rci,
        mscHandler,
        ficHandler,
        rro)
{ }

void RadioReceiver::restart(bool doScan)
{
    ofdmProcessor.set_scanMode(doScan);
    mscHandler.stopProcessing();
    ficHandler.clearEnsemble();
    ofdmProcessor.restart();
}

void RadioReceiver::restart_decoder()
{
    mscHandler.stopProcessing();
    ficHandler.clearEnsemble();
}

void RadioReceiver::stop()
{
    ofdmProcessor.stop();
    mscHandler.stopProcessing();
    ficHandler.clearEnsemble();
```

```
}

void RadioReceiver::setReceiverOptions(const RadioReceiverOptions rro)
{
    string fsm;
    switch (rro.freqsyncMethod) {
        case FreqsyncMethod::GetMiddle: fsm = "GetMiddle"; break;
        case FreqsyncMethod::CorrelatePRS: fsm = "CorrelatePRS"; break;
        case FreqsyncMethod::PatternOfZeros: fsm = "PatternOfZeros";
break;
    }

    clog << "New Receiver Options: " <<
        "TII: " << rro.decodeTII <<
        " disable coarse corr: " << rro.disableCoarseCorrector <<
        " freqsync: " << fsm <<
        "                fft                placement:                "                <<
fftPlacementMethodToString(rro.fftPlacementMethod) << endl;
    ofdmProcessor.setReceiverOptions(rro);
}

bool      RadioReceiver::playSingleProgramme(ProgrammeHandlerInterface&
handler,
        const std::string& dumpFileName, const Service& s)
{
    return playProgramme(handler, s, dumpFileName, true);
}

bool      RadioReceiver::addServiceToDecode(ProgrammeHandlerInterface&
handler,
        const std::string& dumpFileName, const Service& s)
{
    return playProgramme(handler, s, dumpFileName, false);
}

bool RadioReceiver::removeServiceToDecode(const Service& s)
```

```
{
    const auto comps = ficHandler.fibProcessor.getComponents(s);
    for (const auto& sc : comps) {
        if (sc.transportMode() == TransportMode::Audio) {
            const auto& subch = ficHandler.fibProcessor.getSubchannel(sc);
            if (subch.valid()) {
                return mscHandler.removeSubchannel(subch);
            }
        }
    }
    return false;
}

bool RadioReceiver::playProgramme(ProgrammeHandlerInterface& handler,
        const Service& s, const std::string& dumpFileName, bool unique)
{
    const auto comps = ficHandler.fibProcessor.getComponents(s);
    for (const auto& sc : comps) {
        if (sc.transportMode() == TransportMode::Audio) {
            const auto& subch = ficHandler.fibProcessor.getSubchannel(sc);

            if (subch.valid()) {
                if (unique) {
                    mscHandler.stopProcessing();
                }

                if (sc.audioType() == AudioServiceComponentType::DAB ||
                    sc.audioType() == AudioServiceComponentType::DABPlus) {
                    mscHandler.addSubchannel(
                            handler, sc.audioType(), dumpFileName, subch);
                    return true;
                }
            }
        }
    }
```

```
    return false;
}

uint16_t RadioReceiver::getEnsembleId(void) const
{
    return ficHandler.fibProcessor.getEnsembleId();
}

uint8_t RadioReceiver::getEnsembleEcc(void) const
{
    return ficHandler.fibProcessor.getEnsembleEcc();
}

DabLabel RadioReceiver::getEnsembleLabel(void) const
{
    return ficHandler.fibProcessor.getEnsembleLabel();
}

std::vector<Service> RadioReceiver::getServiceList(void) const
{
    return ficHandler.fibProcessor.getServiceList();
}

Service RadioReceiver::getService(uint32_t sId) const
{
    return ficHandler.fibProcessor.getService(sId);
}

std::list<ServiceComponent>          RadioReceiver::getComponents(const
Service& s) const
{
    return ficHandler.fibProcessor.getComponents(s);
}

bool RadioReceiver::serviceHasAudioComponent(const Service& s) const
{
```

```
            for (const auto& sc : getComponents(s)) {
                if (sc.transportMode() == TransportMode::Audio and
                        (sc.audioType() == AudioServiceComponentType::DAB or
                         sc.audioType() == AudioServiceComponentType::DABPlus)) {
                    return true;
                }
            }

            return false;
        }

        Subchannel RadioReceiver::getSubchannel(const ServiceComponent& sc)
const
        {
            return ficHandler.fibProcessor.getSubchannel(sc);
        }

        DABParams& RadioReceiver::getParams()
        {
          return params;
        }

        RadioReceiverStats RadioReceiver::getReceiverStats() const
        {
            RadioReceiverStats s;
            s.timeLastFCT0Frame                                          =
ficHandler.fibProcessor.getTimeLastFCT0Frame();
            return s;
        }
```

ภาคผนวก ข

โปรแกรม Welle.io
Subprogram Tools

```
 */

#include "tools.h"


// --- MiscTools ------------------------------------------------------------
string_vector_t MiscTools::SplitString(const std::string &s, const char delimiter) {
    string_vector_t result;
    std::stringstream ss(s);
    std::string part;

    while(std::getline(ss, part, delimiter))
        result.push_back(part);
    return result;
}


// --- CalcCRC --------------------------------------------------------------
CalcCRC CalcCRC::CalcCRC_CRC16_CCITT(true, true, 0x1021);   //      0001 0000 0010 0001 (16, 12, 5, 0)
CalcCRC CalcCRC::CalcCRC_CRC16_IBM(true, false, 0x8005);    //      1000 0000 0000 0101 (16, 15, 2, 0)
CalcCRC CalcCRC::CalcCRC_FIRE_CODE(false, false, 0x782F);   //      0111 1000 0010 1111 (16, 14, 13, 12, 11, 5, 3, 2, 1, 0)

size_t CalcCRC::CRCLen = 2;

CalcCRC::CalcCRC(bool initial_invert, bool final_invert, uint16_t gen_polynom) {
    this->initial_invert = initial_invert;
    this->final_invert = final_invert;
    this->gen_polynom = gen_polynom;

    FillLUT();
```

```
}

void CalcCRC::FillLUT() {
    for(int value = 0; value < 256; value++) {
            uint16_t crc = value << 8;

            for(int i = 0; i < 8; i++) {
                    if(crc & 0x8000)
                            crc = (crc << 1) ^ gen_polynom;
                    else
                            crc = crc << 1;
            }

            crc_lut[value] = crc;
    }
}

uint16_t CalcCRC::Calc(const uint8_t *data, size_t len) {
    uint16_t crc;
    Initialize(crc);

    for(size_t offset = 0; offset < len; offset++)
            ProcessByte(crc, data[offset]);

    Finalize(crc);
    return crc;
}

void CalcCRC::ProcessBits(uint16_t& crc, const uint8_t *data, size_t len) {
    // byte-aligned start only

    size_t bytes = len / 8;
    size_t bits = len % 8;

    for(size_t offset = 0; offset < bytes; offset++)
            ProcessByte(crc, data[offset]);
```

```
    for(size_t bit = 0; bit < bits; bit++)
            ProcessBit(crc, data[bytes] & (0x80 >> bit));
}


// --- CircularBuffer ---------------------------------------------------------
CircularBuffer::CircularBuffer(size_t capacity) {
    buffer = new uint8_t[capacity];
    this->capacity = capacity;
    Clear();
}

CircularBuffer::~CircularBuffer() {
    delete[] buffer;
}

size_t CircularBuffer::Write(const uint8_t *data, size_t bytes) {
    size_t real_bytes = std::min(bytes, capacity - size);

    // split task on index rollover
    if(real_bytes <= capacity - index_end) {
            memcpy(buffer + index_end, data, real_bytes);
    } else {
            size_t first_bytes = capacity - index_end;
            memcpy(buffer + index_end, data, first_bytes);
            memcpy(buffer, data + first_bytes, real_bytes - first_bytes);
    }

    index_end = (index_end + real_bytes) % capacity;
    size += real_bytes;
    return real_bytes;
}

size_t CircularBuffer::Read(uint8_t *data, size_t bytes) {
    size_t real_bytes = std::min(bytes, size);
```

```
        if(data) {
            // split task on index rollover
            if(real_bytes <= capacity - index_start) {
                memcpy(data, buffer + index_start, real_bytes);
            } else {
                size_t first_bytes = capacity - index_start;
                memcpy(data, buffer + index_start, first_bytes);
                memcpy(data + first_bytes, buffer, real_bytes -
first_bytes);
            }
        }

        index_start = (index_start + real_bytes) % capacity;
        size -= real_bytes;
        return real_bytes;
    }


// --- BitReader ------------------------------------------------------
bool BitReader::GetBits(int& result, size_t count) {
    int result_value = 0;

    while(count) {
        if(data_bytes == 0)
            return false;

        size_t copy_bits = std::min(count, 8 - data_bits);

        result_value <<= copy_bits;
        result_value |= (*data & (0xFF >> data_bits)) >> (8 - data_bits -
copy_bits);

        data_bits += copy_bits;
        count -= copy_bits;

        // switch to next byte
```

```
                if(data_bits == 😎 {
                        data++;
                        data_bytes--;
                        data_bits = 0;
                }
        }

        result = result_value;
        return true;
}


// --- BitWriter ------------------------------------------------------
void BitWriter::Reset() {
        data.clear();
        byte_bits = 0;
}

void BitWriter::AddBits(int data_new, size_t count) {
        while(count) {
                // add new byte, if needed
                if(byte_bits == 0)
                        data.push_back(0x00);

                size_t copy_bits = std::min(count, 8 - byte_bits);
                uint8_t copy_data = (data_new >> (count - copy_bits)) & (0xFF
>> (8 - copy_bits));
                data.back() |= copy_data << (8 - byte_bits - copy_bits);

        //      fprintf(stderr, "data_new: 0x%04X, count: %zu / byte_bits: %zu,
copy_bits: %zu, copy_data: 0x%02X\n", data_new, count, byte_bits, copy_bits,
copy_data);

                byte_bits = (byte_bits + copy_bits) % 8;
                count -= copy_bits;
        }
```

```
}

void BitWriter::AddBytes(const uint8_t *data, size_t len) {
    for(size_t i = 0; i < len; i++)
            AddBits(data[i], 8);
}

void BitWriter::WriteAudioMuxLengthBytes() {
    size_t len = data.size() - 3;
    data[1] |= (len >> 😎 & 0x1F;
    data[2] = len & 0xFF;
}

const dab_channels_t dab_channels {
    { "5A",  174928},
    { "5B",  176640},
    { "5C",  178352},
    { "5D",  180064},

    { "6A",  181936},
    { "6B",  183648},
    { "6C",  185360},
    { "6D",  187072},

    { "7A",  188928},
    { "7B",  190640},
    { "7C",  192352},
    { "7D",  194064},

    { "8A",  195936},
    { "8B",  197648},
    { "8C",  199360},
    { "8D",  201072},

    { "9A",  202928},
    { "9B",  204640},
```

{ "9C", 206352},
{ "9D", 208064},

{"10A", 209936},
{"10N", 210096},
{"10B", 211648},
{"10C", 213360},
{"10D", 215072},

{"11A", 216928},
{"11N", 217088},
{"11B", 218640},
{"11C", 220352},
{"11D", 222064},

{"12A", 223936},
{"12N", 224096},
{"12B", 225648},
{"12C", 227360},
{"12D", 229072},

{"13A", 230784},
{"13B", 232496},
{"13C", 234208},
{"13D", 235776},
{"13E", 237488},
{"13F", 239200},

{ "LA", 1452960},
{ "LB", 1454672},
{ "LC", 1456384},
{ "LD", 1458096},

{ "LE", 1459808},
{ "LF", 1461520},

```
    { "LG", 1463232},
    { "LH", 1464944},

    { "LI", 1466656},
    { "LJ", 1468368},
    { "LK", 1470080},
    { "LL", 1471792},

    { "LM", 1473504},
    { "LN", 1475216},
    { "LO", 1476928},
    { "LP", 1478640},
};
```

ภาคผนวก ค

โปรแกรม Welle.io

Subprogram Airspy

```
*/

#include <iostream>
#include "airspy_sdr.h"

// For Qt translation if Qt is existing
#ifdef QT_CORE_LIB
    #include <QtGlobal>
#else
    #define QT_TRANSLATE_NOOP(x,y) (y)
#endif

static const int EXTIO_NS = 8192;
static const int EXTIO_BASE_TYPE_SIZE = sizeof(float);

CAirspy::CAirspy(RadioControllerInterface &radioController) :
    radioController(radioController),
    SampleBuffer(256 * 1024),
    SpectrumSampleBuffer(8192)
{
    std::clog << "Airspy: " << "Open airspy" << std::endl;

    device = {};

    int result = airspy_init();
    if (result != AIRSPY_SUCCESS) {
        std::clog << "Airspy: " << "airspy_init () failed: " << airspy_error_name((airspy_error)result) << "(" << result << ")" << std::endl;
        throw 0;
    }

    result = airspy_open(&device);
    if (result != AIRSPY_SUCCESS) {
        std::clog << "Airspy: " << "airpsy_open () failed: " << airspy_error_name((airspy_error)result) << "(" << result << ")" << std::endl;
```

```
            throw 0;
        }

        airspy_set_sample_type(device, AIRSPY_SAMPLE_FLOAT32_IQ);

        result = airspy_set_samplerate(device, AIRSPY_SAMPLERATE);
        if (result != AIRSPY_SUCCESS) {
            std::clog << "Airspy: " <<"airspy_set_samplerate() failed: " <<
airspy_error_name((airspy_error)result) << "(" << result << ")" << std::endl;
            throw 0;
        }

        if (sw_agc) {
            setAgc(true);
        }
        else {
            setAgc(false);
            setGain(currentLinearityGain);
        }

        running = false;

        return;
    }

    CAirspy::~CAirspy(void)
    {
        if (device) {
            int result = airspy_stop_rx(device);
            if (result != AIRSPY_SUCCESS) {
                std::clog << "Airspy: airspy_stop_rx () failed: " <<
airspy_error_name((airspy_error)result) << "(" << result << ")" << std::endl;
            }

            result = airspy_close(device);
            if (result != AIRSPY_SUCCESS) {
```

```
                    std::clog << "Airspy: airspy_close () failed: " <<
airspy_error_name((airspy_error)result) << "(" << result << ")" << std::endl;
                }
            }

            airspy_exit();
        }

        void CAirspy::setFrequency(int nf)
        {
            freq = nf;
            int result = airspy_set_freq(device, nf);

            if (result != AIRSPY_SUCCESS) {
                std::clog << "Airspy: airspy_set_freq() failed: " <<
airspy_error_name((airspy_error)result) << "(" << result << ")" << std::endl;
            }
        }

        int CAirspy::getFrequency() const
        {
            return freq;
        }

        bool CAirspy::restart(void)
        {
            int result;
            if (running)
                return true;

            SampleBuffer.FlushRingBuffer();
            SpectrumSampleBuffer.FlushRingBuffer();
            result = airspy_set_sample_type(device, AIRSPY_SAMPLE_FLOAT32_IQ);
            if (result != AIRSPY_SUCCESS) {
                std::clog << "Airspy: airspy_set_sample_type () failed: " <<
airspy_error_name((airspy_error)result) << "(" << result << ")" << std::endl;
```

```
            return false;
        }

        result = airspy_start_rx(device, (airspy_sample_block_cb_fn)callback,
this);
        if (result != AIRSPY_SUCCESS) {
            std::clog << "Airspy: airspy_start_rx () failed: " <<
airspy_error_name((airspy_error)result) << "(" << result << ")" << std::endl;
            return false;
        }

        running = true;
        return true;
    }

    bool CAirspy::is_ok()
    {
        // Check if airspy is still connected
        airspy_error status = (airspy_error) airspy_is_streaming(device);
        if(status != AIRSPY_TRUE && running == true) {
            std::clog << "Airspy: airspy is not working. Maybe it is unplugged.
Code: " << status << "running" << running << std::endl;
            radioController.onMessage(message_level_t::Error,
QT_TRANSLATE_NOOP("CRadioController", "airspy is unplugged."));

            stop();
        }

        return running;
    }

    void CAirspy::stop(void)
    {
        if (!running)
            return;
        int result = airspy_stop_rx(device);
```

```
        if (result != AIRSPY_SUCCESS) {
            std::clog << "Airspy: airspy_stop_rx() failed: " << airspy_error_name((airspy_error)result) << "(" << result << ")" << std::endl;
        }
        running = false;
    }

    int CAirspy::callback(airspy_transfer* transfer)
    {
        if (!transfer) {
            throw std::logic_error("AIRSPY: no transfer");
        }
        auto *p = static_cast<CAirspy*>(transfer->ctx);

        // AIRSPY_SAMPLE_FLOAT32_IQ:
        p->data_available(reinterpret_cast<const DSPCOMPLEX*>(transfer->samples),
                transfer->sample_count);
        return 0;
    }

    // Called from AirSpy data callback which gives us interleaved float32
    // I and Q according to setting given to airspy_set_sample_type() above.
    // The AirSpy runs at 4096ksps, we need to decimate by two.
    int CAirspy::data_available(const DSPCOMPLEX* buf, size_t num_samples)
    {
        if (num_samples % 2 != 0) {
            throw std::runtime_error("CAirspy::data_available() needs an even number of IQ samples to be able to decimate");
        }

        const DSPCOMPLEX* sbuf = reinterpret_cast<const DSPCOMPLEX*>(buf);

        std::vector<DSPCOMPLEX> temp(num_samples/2);
```

```
float maxnorm = 0;

for (size_t i = 0; i < num_samples/2; i++) {
    const auto z = 0.5f * (sbuf[2*i] + sbuf[2*i+1]);
    temp[i] = z;

    if (sw_agc and (num_frames % 10) == 0) {
        if (norm(z) > maxnorm) {
            maxnorm = norm(z);
        }
    }
}

if (sw_agc and (num_frames % 10) == 0) {
    const float maxampl = sqrt(maxnorm);
    //  std::clog  << "Airspy: maxampl: " << maxampl << std::endl;

    if (maxampl > 0.2f) {
        const int newgain = currentLinearityGain - 1;
        if (newgain >= AIRSPY_GAIN_MIN) {
            setGain(newgain);
        }
    }
    else if (maxampl < 0.02f) {
         const int newgain = currentLinearityGain + 1;
        if (newgain <= AIRSPY_GAIN_MAX) {
            setGain(newgain);
        }
    }
}

num_frames++;

SampleBuffer.putDataIntoBuffer(temp.data(), num_samples/2);
SpectrumSampleBuffer.putDataIntoBuffer(temp.data(), num_samples/2);
```

```
    return 0;
}

void CAirspy::reset(void)
{
    SampleBuffer.FlushRingBuffer();
    SpectrumSampleBuffer.FlushRingBuffer();
}

int32_t CAirspy::getSamples(DSPCOMPLEX* Buffer, int32_t Size)
{
    return SampleBuffer.getDataFromBuffer(Buffer, Size);
}

std::vector<DSPCOMPLEX> CAirspy::getSpectrumSamples(int size)
{
    std::vector<DSPCOMPLEX> buf(size);
    int sizeRead = SpectrumSampleBuffer.getDataFromBuffer(buf.data(),
size);
    if (sizeRead < size) {
       buf.resize(sizeRead);
    }
    return buf;
}

int32_t CAirspy::getSamplesToRead(void)
{
    return SampleBuffer.GetRingBufferReadAvailable();
}

int CAirspy::getGainCount()
{
    return 21;
}
```

```
void CAirspy::setAgc(bool agc)
{
    if (not agc) {
        int result = airspy_set_linearity_gain(device, currentLinearityGain);
        if (result != AIRSPY_SUCCESS)
            std::clog << "Airspy: airspy_set_mixer_agc() failed: " <<
airspy_error_name((airspy_error)result) << "(" << result << ")" << std::endl;

    }

    sw_agc = agc;
}

std::string CAirspy::getDescription()
{
    // Get airspy device name and version
    char Version[255] = {0};
    airspy_version_string_read(device, Version, 20);

    // Get airspy library version
    airspy_lib_version_t lib_version;
    airspy_lib_version(&lib_version);

    std::string ver = Version;

    ver += "AirSpy, lib. v" +
        std::to_string(lib_version.major_version) + "." +
        std::to_string(lib_version.minor_version) + "." +
        std::to_string(lib_version.revision);
    return ver;
}

bool CAirspy::setDeviceParam(DeviceParam param, int value)
{
    switch (param) {
        case DeviceParam::BiasTee:
```

```
                    std::clog << "Airspy: Set bias tee to " << value << std::endl;
                    airspy_set_rf_bias(device, value);
                    return true;
                default:
                    return false;
            }
        }

        CDeviceID CAirspy::getID()
        {
            return CDeviceID::AIRSPY;
        }

        float CAirspy::getGain() const
        {
            return currentLinearityGain;
        }

        float CAirspy::setGain(int gain)
        {
            std::clog  << "Airspy: setgain: " << gain << std::endl;
            currentLinearityGain = gain;

            int result = airspy_set_linearity_gain(device, currentLinearityGain);
            if (result != AIRSPY_SUCCESS)
                std::clog      << "Airspy:  airspy_set_mixer_agc()  failed:  "  <<
airspy_error_name((airspy_error)result) << "(" << result << ")" << std::endl;

            return currentLinearityGain;
}
```

ภาคผนวก ง

โปรแกรม Welle.io
Subprogram RTL-SDR

```
*/

#include <iostream>
#include <exception>

#include "rtl_sdr.h"

// For Qt translation if Qt is existing
#ifdef QT_CORE_LIB
    #include <QtGlobal>
#else
    #define QT_TRANSLATE_NOOP(x,y) (y)
#endif

#define READLEN_DEFAULT 8192

// Fallback if function is not defined in shared lib
int __attribute__((weak)) rtlsdr_set_bias_tee(rtlsdr_dev_t *dev, int on);

CRTL_SDR::CRTL_SDR(RadioControllerInterface& radioController) :
    radioController(radioController),
    sampleBuffer(1024 * 1024),
    spectrumSampleBuffer(8192)
{
    open_device();
}

void CRTL_SDR::open_device()
{
    int ret = 0;

    std::clog << "RTL_SDR: " << "Open rtl-sdr" << std::endl;

    // Get all devices
    uint32_t deviceCount = rtlsdr_get_device_count();
    if (deviceCount == 0) {
```

```
            std::clog << "RTL_SDR: " << "No devices found" << std::endl;
            throw 0;
        }
        else {
            std::clog << "RTL_SDR: " << "Found " << deviceCount << " devices.
Uses the first working one" << std::endl;
        }

        //      Iterate over all found rtl-sdr devices and try to open it. Stops if
one device is successful opened.
        for(uint32_t i=0; i<deviceCount; i++) {
            ret = rtlsdr_open(&device, i);
            if (ret >= 0) {
                std::clog << "RTL_SDR: " << " Opening rtl-sdr device" << i <<
std::endl;
                break;
            }
        }

        if (ret < 0) {
            std::clog << "RTL_SDR: " << " Opening rtl-sdr failed" << std::endl;
            throw 0;
        }

        // Set sample rate
        ret = rtlsdr_set_sample_rate(device, INPUT_RATE);
        if (ret < 0) {
            std::clog << "RTL_SDR: " << " Setting sample rate failed" << std::endl;
            throw 0;
        }

        // Get tuner gains
        uint32_t gainsCount = rtlsdr_get_tuner_gains(device, NULL);
        std::clog << "RTL_SDR: " << " Supported gain values" << gainsCount <<
std::endl;
        gains.resize(gainsCount);
```

```
        gainsCount = rtlsdr_get_tuner_gains(device, gains.data());

        for (int i = gainsCount; i > 0; i--) {
            std::clog << "RTL_SDR: " << " gain " << (gains[i - 1] / 10.0) <<
std::endl;
        }

        // Always use manual gain, the AGC is implemented in software
        rtlsdr_set_tuner_gain_mode(device, 1);

        // Enable AGC by default
        setAgc(true);

        rtlsdrUnplugged = false;
    }

    CRTL_SDR::~CRTL_SDR(void)
    {
        stop();

        rtlsdr_close(device);
    }

    void CRTL_SDR::setFrequency(int frequency)
    {
        stop();
        this->frequency = frequency;
        restart();
    }

    int CRTL_SDR::getFrequency(void) const
    {
        return frequency;
    }

    bool CRTL_SDR::restart(void)
```

```
{
    int ret;

    if(rtlsdrUnplugged) {
        try {
            open_device();
        } catch (...) {
            // An error occurred. Maybe the device isn't present.
            radioController.onMessage(message_level_t::Error,
QT_TRANSLATE_NOOP("CRadioController", "Error opening RTL-SDR. See log for details."));
            return false;
        }
    }

    if (rtlsdrRunning) {
        return true;
    }

    sampleBuffer.FlushRingBuffer();
    spectrumSampleBuffer.FlushRingBuffer();
    ret = rtlsdr_reset_buffer(device);
    if (ret < 0)
        return false;

    rtlsdr_set_center_freq(device, frequency + frequencyOffset);
    rtlsdrRunning = true;

    rtlsdrThread = std::thread(&CRTL_SDR::rtlsdr_read_async_wrapper, this);
    agcThread = std::thread(&CRTL_SDR::agc_timer_thread, this);

    return true;
}

bool CRTL_SDR::is_ok(void)
{
```

```
    return not rtlsdrUnplugged;
}

void CRTL_SDR::stop(void)
{
    if (not rtlsdrRunning)
        return;

    rtlsdrRunning = false;

    if (agcThread.joinable()) {
        agcThread.join();
    }

    rtlsdr_cancel_async(device);
    if (rtlsdrThread.joinable()) {
        rtlsdrThread.join();
    }
}

float CRTL_SDR::getGain() const
{
    return currentGain / 10.0;
}

float CRTL_SDR::setGain(int gain_index)
{
    if ((size_t)gain_index >= gains.size()) {
        std::clog << "RTL_SDR: " << "Unknown gain count" << gain_index <<
std::endl;
        return 0;
    }

    currentGainIndex = gain_index;
    currentGain = gains[gain_index];
```

```
            //std::clog << "RTL_SDR: " << "Set gain to" << currentGain / 10.0 << "db"
<< std::endl;

            int ret = rtlsdr_set_tuner_gain(device, currentGain);
            if (ret != 0) {
                std::clog << "RTL_SDR: " << "Setting gain failed" << std::endl;
            }

            return currentGain / 10.0;
        }

        int CRTL_SDR::getGainCount()
        {
            return gains.size() - 1;
        }

        void CRTL_SDR::setAgc(bool AGC)
        {
            if (AGC == true) {
                isAGC = true;
            }
            else {
                isAGC = false;
                setGain(currentGainIndex);
            }
        }

        bool CRTL_SDR::setDeviceParam(DeviceParam param, int value)
        {
            switch(param) {
                case DeviceParam::BiasTee:
                if(rtlsdr_set_bias_tee)
                {
                    std::clog << "RTL_SDR: "<< "Set bias tee to " << value <<
std::endl;
                    rtlsdr_set_bias_tee(device, value);
```

```
                }
                else
                {
                    std::clog << "RTL_SDR: " << "Error: rtlsdr_set_bias_tee() not
defined!" << std::endl;
                }
                return true;

                default: std::runtime_error("Unsupported device parameter");
            }

            return false;
        }

        std::string CRTL_SDR::getDescription()
        {
            char manufact[256] = {0};
            char product[256] = {0};
            char serial[256] = {0};

            rtlsdr_get_usb_strings(device, manufact, product, serial);

            std::string name;
            name += manufact;
            name += ", ";
            name += product;
            name += ", ";
            name += serial;
            return name;
        }

        CDeviceID CRTL_SDR::getID()
        {
            return CDeviceID::RTL_SDR;
        }
```

```
void CRTL_SDR::agc_timer_thread(void)
{
    while (rtlsdrRunning && not rtlsdrUnplugged) {
        std::this_thread::sleep_for(std::chrono::milliseconds(50));

        if (isAGC) {
            // Check for overloading
            if (minAmplitude == 0 || maxAmplitude == 255) {
                // We have to decrease the gain
                if (currentGainIndex > 0) {
                    setGain(currentGainIndex - 1);
                    //std::clog << "RTL_SDR: " << "Decreased gain to " << (float)currentGain / 10.0f << std::endl;
                }
            }
            else {
                if (currentGainIndex < ((ssize_t)gains.size() - 1)) {
                    // Calc if a gain increase overloads the device. Calc it from the gain values
                    int NewGain = gains[currentGainIndex + 1];
                    float DeltaGain = ((float) NewGain / 10) - ((float) currentGain / 10);
                    float LinGain = pow(10, DeltaGain / 20);

                    int NewMaxValue = (float) maxAmplitude * LinGain;
                    int NewMinValue = (float) minAmplitude / LinGain;

                    // We have to increase the gain
                    if(NewMinValue >= 0 && NewMaxValue <= 255) {
                        setGain(currentGainIndex + 1);
                        //std::clog << "RTL_SDR: " << "Increased gain to " << (float) currentGain / 10 << std::endl;
                    }
                }
            }
        }
```

```
                else { // AGC is off
                    if (minAmplitude == 0 || maxAmplitude == 255) {
                        std::string Text = QT_TRANSLATE_NOOP("CRadioController",
"ADC overload. Maybe you are using a too high gain.");
                        std::clog << "RTL_SDR: " << Text << std::endl;
                        radioController.onMessage(message_level_t::Information, Text);
                    }
                }
            }
        }

        int32_t CRTL_SDR::getSamples(DSPCOMPLEX *buffer, int32_t size)
        {
            std::vector<uint8_t> tempBuffer(2 * size);

            int32_t amount = sampleBuffer.getDataFromBuffer(tempBuffer.data(), 2
* size);

            // Normalise samples
            for (int i = 0; i < amount / 2; i++) {
                buffer[i] = DSPCOMPLEX(
                    (float(tempBuffer[2 * i] - 128)) / 128.0,
                    (float(tempBuffer[2 * i + 1] - 128)) / 128.0);
            }

            return amount / 2;
        }

        std::vector<DSPCOMPLEX> CRTL_SDR::getSpectrumSamples(int size)
        {
            std::vector<uint8_t> tempBuffer(2 * size);

            // Get samples
            int32_t amount = spectrumSampleBuffer.getDataFromBuffer(
                tempBuffer.data(), 2 * size);
```

```
    std::vector<DSPCOMPLEX> buffer(amount / 2);

    // Convert samples into generic format
    for (int i = 0; i < amount / 2; i++) {
        buffer[i] = DSPCOMPLEX(
                (float(tempBuffer[2 * i] - 128)) / 128.0,
                (float(tempBuffer[2 * i + 1] - 128)) / 128.0);
    }

    return buffer;
}

int32_t CRTL_SDR::getSamplesToRead(void)
{
    return sampleBuffer.GetRingBufferReadAvailable() / 2;
}

void CRTL_SDR::reset(void)
{
    sampleBuffer.FlushRingBuffer();
}

void CRTL_SDR::rtlsdr_read_callback(uint8_t* buf, uint32_t len, void* ctx)
{
    if (ctx) {
        CRTL_SDR rtlsdr = (CRTL_SDR)ctx;

        if (len != READLEN_DEFAULT) {
            std::clog << "RTL_SDR: " << "Short read" << std::endl;
            return;
        }

        int32_t tmp = rtlsdr->sampleBuffer.putDataIntoBuffer(buf, len);
        if ((len - tmp) > 0)
            rtlsdr->sampleCounter += len - tmp;
```

```
                rtlsdr->spectrumSampleBuffer.putDataIntoBuffer(buf, len);
                rtlsdr->putIntoRecordBuffer(*buf, len);

                // Check if device is overloaded
                rtlsdr->minAmplitude = 255;
                rtlsdr->maxAmplitude = 0;

                for (uint32_t i=0;i<len;i++) {
                    if (rtlsdr->minAmplitude > buf[i])
                        rtlsdr->minAmplitude = buf[i];

                    if (rtlsdr->maxAmplitude < buf[i])
                        rtlsdr->maxAmplitude = buf[i];
                }
            }
            else {
                std::clog << "RTL_SDR: " << "ERROR no ctx in RTLSDR callback" <<
std::endl;
            }
        }

        void CRTL_SDR::rtlsdr_read_async_wrapper()
        {
            std::clog << "RTL_SDR: " << "Start rtlsdr_read_async_wrapper() thread"
<< std::endl;
            rtlsdr_read_async(device,
                              (rtlsdr_read_async_cb_t)&CRTL_SDR::rtlsdr_read_callback,
                              (void*)this, 0, READLEN_DEFAULT);

            if(rtlsdrRunning) {
                radioController.onMessage(message_level_t::Error,
QT_TRANSLATE_NOOP("CRadioController", "RTL-SDR is unplugged."));
                rtlsdrUnplugged = true;
            }

            std::clog <<
```

ภาคผนวก จ

Emergency Alarm Script

```
from PIL import Image
import pyautogui
import pygame
import time
import os
import subprocess

def capture_screen(region=(0, 0, 1920, 1080)):
    screenshot = pyautogui.screenshot(region=region)
    screenshot_path = "screenshot.png"
    screenshot.save(screenshot_path)
    return screenshot_path

def play_notification_sound():
    pygame.mixer.init()

    mp3_file = "em.mp3"
    mp3_path = os.path.join("/home/kao/emergency", mp3_file)

    pygame.mixer.music.load(mp3_path)
    pygame.mixer.music.play()

def show_fullscreen_image(image_path):
    pygame.init()

    # ให้หน้าต่าง Pygame มีขนาดเท่ากับขนาดที่กำหนด
    screen = pygame.display.set_mode((1280, 640))

    pygame.display.set_caption("Custom Image Display")

    # ดึงขนาดหน้าจอปัจจุบัน
    screen_width, screen_height = pygame.display.get_window_size()

    # โหลดภาพและปรับขนาดให้เต็มหน้าจอ
    image = pygame.image.load(image_path)
    image = pygame.transform.scale(image, (screen_width, screen_height))
```

```
            screen.blit(image, (0, 0))
            pygame.display.flip()

            # รอจนกว่าเสียงจะเล่นจบ
            while pygame.mixer.music.get_busy():
                pygame.time.Clock().tick(1)

            pygame.quit()

        def open_program(program_path):
            subprocess.Popen(program_path)

        def main():
            previous_image_hash = None

            while True:
                screenshot_path = capture_screen()
                current_image_hash = hash(pyautogui.screenshot().tobytes())

                if previous_image_hash is not None and previous_image_hash !=
current_image_hash:
                    play_notification_sound()

                    # แก้ที่อยู่ของไฟล์ภาพเป็นที่ถูกต้อง
                    png_file = "em3.jpg"
                    image_path = os.path.join("/home/kao/emergency", png_file)
                    show_fullscreen_image(image_path)

                    print("em3.jpg")

                    # เปิดโปรแกรมที่ต้องการ
                    welle_command = "welle-io"  # หรือคำสั่งที่ใช้เรียก welle.io ในที่นี้
                    subprocess.run(welle_command, shell=True)

                previous_image_hash = current_image_hash
```

```
            time.sleep(10)

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

ภาคผนวก ฉ

FRONTEND WEBPAGE HTML SCRIPT

```
<!DOCTYPE html>
<html lang="en">
  <head>
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="myVideo.css">
    <!-- basic -->
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <!-- mobile metas -->
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="viewport" content="initial-scale=1, maximum-scale=1">
    <!-- site metas -->
    <title>DAB+ Radio </title>
    <meta name="keywords" content="">
    <meta name="description" content="">
    <meta name="author" content="">
    <!-- bootstrap css -->
    <link rel="stylesheet" href="css/bootstrap.min.css">
    <!-- style css -->
    <link rel="stylesheet" href="css/style.css">
    <!-- Responsive-->
    <link rel="stylesheet" href="css/responsive.css">
    <!-- fevicon -->
    <link rel="icon" href="images/fevicon.png" type="image/gif" />
    <!-- Tweaks for older IEs-->
    <link rel="stylesheet" href="https://netdna.bootstrapcdn.com/font-awesome/4.0.3/css/font-awesome.css">
    <!--[if lt IE 9]>
    <script src="https://oss.maxcdn.com/html5shiv/3.7.3/html5shiv.min.js"></script>
    <script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script><![endif]-->
    <style>
      #volumeRange {
        width: 150px;
```

```
        }
        #volumeBtn {
  background-color: #3498db; /* Default button color */
  color: #fff;
  border: none;
  padding: 8px 16px;
  cursor: pointer;
}

#volumeBtn.muted {
  background-color: #e74c3c; /* Red color when muted */
}

      </style>
    </head>
    <!-- body -->
    <body class="main-layout" style="background-color:#251A19;">
      <!-- loader  -->
      <div class="loader_bg">
        <div class="loader"><img src="images/loading.gif" alt="#"/></div>
      </div>
      <!-- end loader -->
      <!-- header -->
      <div class="header">
        <div class="container">
          <div class="row d_flex">
            <div class=" col-md-2 col-sm-3 col logo_section">
              <div class="full">
                <div class="center-desk">
                  <div class="logo">
                    <a      href="index.html"><img      src="images/logo.png"
alt="#" /></a>
                  </div>
                </div>
              </div>
            </div>
```

```
                    <div class="col-md-8 col-sm-12">
                      <nav class="navigation navbar navbar-expand-md navbar-dark
">
                        <button class="navbar-toggler" type="button" data-
toggle="collapse" data-target="#navbarsExample04" aria-controls="navbarsExample04"
aria-expanded="false" aria-label="Toggle navigation">
                        <span class="navbar-toggler-icon"></span>
                        </button>
                        <div class="collapse navbar-collapse"
id="navbarsExample04">
                          <ul class="navbar-nav mr-auto">
                            <li class="nav-item active">
                              <a class="nav-link" href="project.html">Home</a>
                            </li>
                            <li class="nav-item">
                              <a class="nav-link" href="about.html">About</a>
                            </li>
                            <li class="nav-item">
                              <a class="nav-link"
href="channel.html">Channel</a>
                            </li>
                            <li class="nav-item">
                              <a class="nav-link"
href="controller.html">controller</a>
                            </li>

                            <li class="nav-item">
                              <a class="nav-link" href="product.html">project</a>
                            </li>
                          </ul>
                        </div>
                      </nav>
                    </div>
                    <div class="col-md-2  d_none">
                      <ul class="email text_align_right">
```

```
                    </ul>
                  </div>
                </div>
              </div>
            </div>
            <!-- end header inner -->
            <!-- top -->
            <div class="full_bg">
              <div class="slider_main">
                <div class="container">
                  <div class="row">
                    <div class="col-md-12">
                      <!-- carousel code -->
                      <div id="banner1" class="carousel slide">
                        <ol class="carousel-indicators">
                          <li data-target="#banner1" data-slide-to="0"
class="active"></li>
                          <li data-target="#banner1" data-slide-to="1"></li>
                          <li data-target="#banner1" data-slide-to="2"></li>
                        </ol>
                        <div class="carousel-inner">
                          <!-- first slide -->
                          <div class="carousel-item active">
                            <div class="carousel-caption relative">
                              <div class="row">
                                <div class="col-md-6">
                                  <div class="dream">
                                    <h1>
                                      Telecom <br>Engineering <br>DAB+ Radio
Receiver
                                    </h1>
                                    <a                          class="read_more"
href="Javascript:void(0)">scan</a>
                                    <a                          class="read_more"
href="Javascript:void(0)">quickplay</a>
                                  </div>
```

```
                    </div>
                    <div class="col-md-6">
                      <div align="center" >

                      </div>
                    </div>
                  </div>
                </div>
                <!-- second slide -->
                <div class="carousel-item">
                  <div class="carousel-caption relative">
                    <div class="row">
                      <div class="col-md-6">
                        <div class="dream" align="top-center">
                          <h2>
                            service overview
                          </h2>
                        </div>
                        <div id = "tb1" class="dream">

                        </div>
                      </div>
                      <div class="col-md-6">
                        <div align="left"
                        padding bottom : 10% ;>

                        <script>
                          var video =
document.getElementById("video");
                          var videoSrc =
"http://localhost:8080/hls/test.m3u8";
                          if (Hls.isSupported()) {
                            var hls = new Hls();
                            hls.loadSource(videoSrc);
                            hls.attachMedia(video);
```

```
                }
                // hls.js is not supported on platforms that
do not have Media Source
                // Extensions (MSE) enabled.
                //
                // When the browser has built-in HLS
support (check using `canPlayType`),
                // we can provide an HLS manifest (i.e.
.m3u8 URL) directly to the video
                // element through the src property. This is
using the built-in support
                // of the plain video element, without using
hls.js.
                //
                // Note: it would be more normal to wait on
the 'canplay' event below however
                // on Safari (where you are most likely to
find built-in HLS support) the
                // video.src URL must be on the user-driven
white-list before a 'canplay'
                // event will be emitted; the last video
event that can be reliably
                // listened-for when the URL is not on the
white-list is 'loadedmetadata'.
                else                                    if
(video.canPlayType("application/vnd.apple.mpegurl")) {
                  video.src = videoSrc;
                }
              </script>
        <button id="volumeBtn" onclick="toggleMute()">Mute</button>
        <input type="range" id="volumeRange" min="0" max="1" step="0.01"
value="1" onchange="changeVolume()">
        <script src="script.js"></script>
```

```
                </div>
              </div>
            </div>
          </div>
        </div>
        <!-- third slide-->
        <div class="carousel-item">
          <div class="carousel-caption relative">
            <div class="row">
              <
```

ภาคผนวก ช

Script การสลับ Station เมื่อทำการรีเซ็ทโปรแกรม

```
        if (isChannelScan == true) {
            stopScan();
        }

        QStringList previousChannel = QStringList() << serialise_serviceid(service)
<< channel;

        bool isRestartOk = deviceRestart();

        // Set the channel to "6C" after restarting the device
        setChannel("6C", true);

        // Set the service ID to 0x1004
        service = deserialise_serviceid("0x1004");

        setService(service);

        QSettings settings;
        settings.setValue("lastchannel", previousChannel);

        if (isRestartOk) {
            isPlaying = true;
            emit isPlayingChanged(isPlaying);
        } else {
            resetTechnicalData();
            currentTitle = title;
            emit titleChanged();
            currentText = tr("Playback failed");
            emit textChanged();
        }
```