วิธีการส่งภาพด้วยวิธีไบต์อาเรย์ผ่านช่องสัญญาณความเร็วอัตราบิตต่ำ

IMAGE TRANSMISSION WITH BYTE ARRAY METHOD VIA LOW BIT RATE CHANNEL

ปณิธาน บุญจนาวิโรจน์
PANITHAN BOONJANAWIROD

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมโทรคมนาคม
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ. 2559
KMITL-2016-EN-M-010-210

วิธีการส่งภาพด้วยวิธีไบต์อาเรย์ผ่านช่องสัญญาณความเร็วอัตราบิตต่ำ

IMAGE TRANSMISSION WITH BYTE ARRAY METHOD
VIA LOW BIT RATE CHANNEL

ปณิธาน บุญจนาวิโรจน์
PANITHAN BOONJANAWIROD

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมโทรคมนาคม
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ.2559

KMITL-2016-EN-M-010-210

IMAGE TRANSMISSION WITH BYTE ARRAY METHOD VIA LOW BIT RATE CHANNEL

PANITHAN BOONJANAWIROD

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENT FOR THE DEGREE OF
MASTER OF ENGINEERING IN TELECOMMUNICATIONS ENGINEERING
FACULTY OF ENGINEERING
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
2016

KMITL-2016-EN-M-010-210

คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ใบรับรองวิทยานิพนธ์

หัวข้อวิทยานิพนธ์ วิธีการส่งภาพด้วยวิธีไบต์อาเรย์ผ่านช่องสัญญาณความเร็วอัตราบิตต่ำ
Thesis Title Image Transmission with Byte Array Method via Low Bit Rate Channel
นักศึกษา นายปณิธาน บุญจนาวิโรจน์
รหัสประจำตัว 54611851
ปริญญา วิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชา วิศวกรรมโทรคมนาคม
อาจารย์ที่ปรึกษาวิทยานิพนธ์ รศ.ดร.สุวิพล สิทธิชีวภาค
หมายเลขวิทยานิพนธ์ KMITL-2016-EN-M-010-210

คณะกรรมการสอบวิทยานิพนธ์		ลายมือชื่อ
ศ.ดร.ประยุทธ	อัครเอกฒาลิน	
ดร.สถาพร	พรหมวงศ์	
ผศ.ดร.พิชญ	สุพรรณกูล	
ผศ.ดร.สุทธิชัย	นพนาคีพงษ์	
รศ.ดร.สุวิพล	สิทธิชีวภาค	

วัน / เดือน/ ปี ที่สอบ วันอังคารที่ 20 ธันวาคม พ.ศ. 2559 เวลา 09.00-11.00 น.
สถานที่สอบ ณ อาคาร A ชั้น 5 ห้องประชุม 1

คณะวิศวกรรมศาสตร์ รับรองแล้ว

(รองศาสตราจารย์ ดร. คมสัน มาลีสี)
คณบดี คณะวิศวกรรมศาสตร์
วันที่ 20 ธันวาคม พ.ศ. 2559

หัวข้อวิทยานิพนธ์	วิธีการส่งภาพด้วยวิธีไบต์อาเรย์ผ่านช่องสัญญาณความเร็วอัตราบิตต่ำ
นักศึกษา	นายปณิธาน บุญจนาวิโรจน์
รหัสประจำตัว	54611851
ปริญญา	วิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชา	วิศวกรรมโทรคมนาคม
พ.ศ.	2559
อาจารย์ที่ปรึกษาวิทยานิพนธ์	รศ.ดร.สุวิพล สิทธิชีวภาค

บทคัดย่อ

วิทยานิพนธ์ฉบับนี้ นำเสนอระบบการส่งภาพโดยผ่านกระบวนการแปลงไฟล์รูปภาพเป็นบัฟเฟอร์อิมเมจ จากนั้นทำการเข้ารหัสเป็นไบต์อาเรย์เอาท์พุตสตรีมต่อด้วยกระบวนการแปลงเป็นไบต์อาเรย์ที่พร้อมสำหรับการส่งข้อมูลผ่านสายสัญญาณแบบอนุกรม จากนั้นทางด้านฝั่งรับข้อมูลจะนำข้อมูลที่ได้รับไปเข้ากระบวนการไบต์อาเรย์ เพื่อทำการถอดเฮดเดอร์ของชุดข้อมูลแปลงกลับเป็นไบต์อาเรย์เอาท์พุตสตรีม และทำการถอดรหัสข้อมูลแต่ละชุดให้สามารถนำไปเรียงตามรูปแบบของบัฟเฟอร์อิมเมจที่กำหนดไว้ เมื่อได้ข้อมูลที่ครบถ้วนจะสามารถนำไปเข้ากระบวนการแสดงผลต่อไป ในวิทยานิพนธ์ฉบับนี้ยังได้กล่าวถึงวิธีการปรับเปลี่ยนค่าของไบต์อาเรย์ และผลการทดลองเพื่อหาค่าของไบต์อาเรย์ที่เหมาะสมสำหรับการส่งข้อมูลภาพ

Thesis	Image Transmission with Byte Array Method via Low Bit Rate Channel
Student	Mr.Panithan Boonjanawirod
Student ID.	54611851
Degree	Master of Engineering
Program	Telecommunications Engineering
Year	2016
Thesis Advisor	Assoc.Prof.Dr.Suvepon Sittichivapak

ABSTRACT

This article presents a new method for image communication via low bit rate channel that transfer JPEG image to buffered Image storage and encoding to Byte Array Output Stream then receiving byte Array data that ready to transfer by used Byte Array processing method. Then Byte Array will be transferred via serial port with Low Bit Rate channel to destination. After destination received data, the data will be changed Byte Array to Byte Array Output Stream by using Byte Array processing method as same as source data. Then, Byte Array Output Stream is decoded and transferred to Buffered Image storage for showing the result. This article also presents about Byte Array modification and the correct result.

กิตติกรรมประกาศ

วิทยานิพนธ์เล่มนี้สำเร็จได้ด้วยความกรุณาจากอาจารย์ที่ปรึกษา รศ.ดร.สุวิพล สิทธิชีวภาค ที่ให้ความช่วยเหลือ ให้คำชี้แนะช่วยแก้ปัญหาตลอดจนให้ความรู้และประการณ์ที่ดีแก่ข้าพเจ้า

ขอขอบคุณ คุณกฤษฎิ์ธนิก ศรีธนสาร ที่คอยให้คำปรึกษาและชี้แนะแนวทางการออกแบบระบบการส่งสัญญาณภาพ โดยอาศัยระบบซอฟต์แวร์เสรี (Open Source)

สุดท้ายต้องขอกราบขอบพระคุณ บิดา มารดา และครอบครัวของข้าพเจ้าที่เป็นกำลังใจ และให้การสนับสนุนในทุกๆเรื่อง ทำให้ข้าพเจ้าสามารถทำวิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ด้วยดี

สำหรับคุณงามความดีอันใดที่เกิดจากวิทยานิพนธ์ฉบับนี้ ข้าพเจ้าขอมอบให้กับบิดามารดา ญาติผู้ใหญ่ ซึ่งเป็นที่รักและเคารพยิ่ง ตลอดจนครูบาอาจารย์ที่เคารพทุกท่านที่ได้ประสิทธิ์ประสาทวิชาความรู้และถ่ายทอดประสบการณ์ที่ดีให้แก่ข้าพเจ้าตลอดมา

ปณิธาน บุญจนาวิโรจน์

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ หน้า

สารบัญรูป

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

1.1ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบันการส่งสัญญาณผ่านคลื่นวิทยุย่านเอชเอฟ(HF ; High Frequency) มีความสามารถในการสื่อสารในระยะทางไกล และนิยมใช้กันอย่างแพร่หลายในรูปแบบของการส่งสัญญาณเสียง เช่น วิทยุคลื่นสั้น, วิทยุสมัครเล่น และการสื่อสารทางการบินที่ระยะข้ามเส้นขอบฟ้า ดังนั้นถ้ามีการปรับปรุงวิธีการในการส่งข้อมูลให้เหมาะสมจะสามารถส่งสัญญาณภาพได้ ซึ่งจะเป็นการเพิ่มขีดความสามารถให้แก่ระบบส่งสัญญาณผ่านคลื่นวิทยุ

วีดีโอสตรีมมิ่ง(Video Streaming) เป็นรูปแบบหนึ่งของสื่อมัลติมีเดีย(Multimedia) ที่นิยมใช้งานผ่านระบบเครือข่ายอินเตอร์เน็ต ซึ่งต้องการเทคนิคและวิธีการในการปรับแต่งคุณสมบัติเพื่อให้สามารถใช้งานผ่านระบบเครือข่ายอินเตอร์เน็ต วิธีการเหล่านั้น คือ เทคนิคของการเข้ารหัสวิดีโอประเภทต่างๆ(Video Encoding) เพื่อลดขนาดของข้อมูลได้มากที่สุดสำหรับส่งผ่านระบบเครือข่ายอินเตอร์เน็ตที่จำกัดความสามารถในการส่งผ่านของข้อมูล(Throughput) ดังนั้นการส่งภาพผ่านระบบส่งสัญญาณคลื่นวิทยุหรือช่องสัญญาณความเร็วอัตราบิตต่ำ ต้องอาศัยกลไกในการควบคุมปริมาณข้อมูลของภาพ เพื่อส่งผ่านระบบช่องสัญญาณความเร็วอัตราบิตต่ำ เช่นเดียวกัน

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา

วิทยานิพนธ์ฉบับนี้มุ่งหวังเพื่อศึกษาระบบการส่งไฟล์ภาพผ่านช่องทางการสื่อสารความเร็วอัตราบิตต่ำ โดยใช้วิธีการไบต์อาเรย์เพื่อควบคุมปริมาณของข้อมูลภาพ และทดลองการปรับเปลี่ยนขนาดของไฟล์ภาพ, อัตราการรับส่งข้อมูล, ขนาดของไบต์อาเรย์ เพื่อเปรียบเทียบระยะเวลาตั้งแต่เริ่มทำการส่งข้อมูลภาพจนถึงการแปลงไฟล์ภาพให้สามารถแสดงผล รวมถึงการหาค่าที่เหมาะสมกับความสามารถของช่องทางการสื่อสารความเร็วอัตราบิตต่ำ

1.3 สมมุติฐานของการศึกษา

การส่งสัญญาณภาพเพื่อให้สามารถส่งผ่านข้อมูลผ่านช่องทางการสื่อสารความเร็วอัตราบิตต่ำมีความจำเป็นต้องใช้วิธีการหลายวิธีร่วมด้วยกัน โดยหลักการ คือ การลดปริมาณข้อมูลที่จะส่งผ่านช่องทางการสื่อสารความเร็วอัตราบิตต่ำ การปรับความละเอียดของเฟรมภาพ(Resolution) การปรับค่าความลึกของสี(Color Depth) แม้กระทั่งการเข้ารหัสภาพรูปแบบต่างๆ(Image File Formats) เนื่องจากมีข้อจำกัดทางการส่งผ่านของข้อมูล และทางฝั่งรับข้อมูลมีความสามารถในการรับข้อมูลที่ต่ำมาก ดังนั้นจึงต้องอาศัยทางฝั่งส่งข้อมูลในการปรับเปลี่ยนขนาด ความละเอียดและรูปแบบการเข้ารหัส รวมไปถึงการควบคุมปริมาณข้อมูลของภาพ

ด้วยปัญหาข้างต้นที่ได้กล่าวมานี้จึงเป็นสาเหตุหนึ่งที่ทำให้ต้องศึกษาและหาวิธีที่จะแก้ปัญหาที่เกิดขึ้นดังกล่าว จึงได้ศึกษาวิจัยวิธีการต่างๆ โดยใช้วิธีกำหนดขนาดและความละเอียดให้แน่นอนส่วนรูปแบบการเข้ารหัสใช้เป็นแบบเจเพ็ก(JPEG ; Joint Photographic Experts Group) เพราะเป็นการ

เข้ารหัสรูปภาพที่มีการบีบอัดได้มากที่สุด จากนั้นใช้วิธีการไบต์อาเรย์ เพื่อให้สามารถควบคุมปริมาณของข้อมูลภาพให้มีความเหมาะสมกับขีดความสามารถของช่องทางการสื่อสารความเร็วอัตราบิตต่ำ

1.4 ทฤษฎีหรือแนวความคิดที่ใช้ในการวิจัย

การส่งภาพผ่านช่องสัญญาณความเร็วอัตราบิตต่ำมีข้อจำกัดทางการส่งผ่านของข้อมูล โดยเฉพาะอย่างยิ่งถ้าปริมาณของข้อมูลที่ส่งผ่านมีมากเกินจะทำให้ทางด้านฝั่งรับข้อมูลไม่สามารถที่จะรับข้อมูลที่ส่งมาได้ทันทั้งหมดจึงเกิดการสูญหายของข้อมูล ดังนั้นถ้าสามารถทำการจำกัดปริมาณของข้อมูลที่จะส่ง จะทำให้ทางด้านฝั่งรับข้อมูลสามารถที่จะรับข้อมูลทั้งหมดได้ทัน

1.5 ขอบเขตการวิจัย

ในวิทยานิพนธ์ฉบับนี้นำเสนอวิธีการส่งภาพผ่านสายสัญญาณแบบอนุกรม โดยผ่านพอร์ตอนุกรมอาร์เอส 232 (Serial Port RS232) และใช้คอนเนคเตอร์แบบดีบี 9 (DB9 Connector) ซึ่งถือเป็นช่องสัญญาณความเร็วอัตราบิตต่ำรูปแบบหนึ่ง ส่วนทางด้านฝั่งส่งข้อมูลที่ต้องมีการเข้ารหัสและการจำกัดปริมาณของข้อมูลที่ใช้ในการส่ง ทางด้านฝั่งรับข้อมูลที่ต้องทำการรับข้อมูลมาให้ครบถ้วนและทำการถอดรหัส เพื่อที่จะนำไปแสดงผลต่อไปนั้น ได้ใช้โปรแกรมชนิดไอดีอี (IDE ; Integrated Development Environment) เช่น เน็ตบีนส์(NetBeans) เป็นเครื่องมือช่วยในการพัฒนาแอพพลิเคชัน(Application) ด้วยภาษาจาวา(Java) ที่ใช้ทดสอบระบบ และทำงานร่วมกับเอพีไอ(API ; Application Programming Interface) มีชื่อว่าเจเอ็มเอฟ (JMF ; Java Media Framework) เพื่อจัดการกับข้อมูลด้านมัลติมีเดียโดยเฉพาะ และใช้อัลกอริทึมสำหรับการแบ่งเฟรม(Frame) ในบัฟเฟอร์(Buffer) รวมถึงวิธีการแปลงไบต์อาเรย์ โดยมีสภาวะแวดล้อมหรือเงื่อนไขที่ใช้งานการทดสอบดังนี้

1.การติดต่อสื่อสารระหว่างด้านฝั่งส่งข้อมูลและด้านฝั่งรับข้อมูลมีการเชื่อมต่อกันผ่านพอร์ตอนุกรมอาร์เอส 232 และมีอัตราการรับส่งข้อมูล(Baud Rate) ไม่ต่ำกว่า 1200 บิตต่อวินาที(bps ; Bit Per Second)

2.ความละเอียดของเฟรมภาพที่จะทำการส่งต้องมีความละเอียดไม่เกิน 800x600 พิกเซล (Pixels)

3. ขนาดของเฟรมภาพที่จะทำการส่งต้องมีขนาดไม่เกิน 90,000 ไบต์(bytes)

4.ประเภทของภาพที่จะทำการส่งต้องอยู่ในรูปแบบเจพีจี(JPEG ; Joint Photographic Experts Group) หรือพีเอ็นจี(PNG ; Portable Network Graphic)

ผลที่ได้คือการทดสอบเพื่อเปรียบเทียบปริมาณของไบต์อาเรย์ที่ทำการส่งเทียบกับระยะเวลาในการรับข้อมูล โดยที่จะทำการจับเวลาตั้งแต่การรับข้อมูลเข้ามาจนถึงการแสดงผลภาพ และจะทำการเปรียบเทียบผลลัพธ์ดังกล่าวกันให้รูปแบบของกราฟ

1.6 ขั้นตอนของการศึกษา

วิทยานิพนธ์ฉบับนี้ได้แบ่งเนื้อหาออกเป็น 5 บทด้วยกัน คือ

บทที่ 1 กล่าวถึงความเป็นมาของงานวิจัย ความมุ่งหมาย และวัตถุประสงค์ สมมติฐานทฤษฎีที่ใช้ขอบเขตของงานวิจัยและขั้นตอนของการศึกษา

บทที่ 2 กล่าวถึงทฤษฎีที่เกี่ยวข้องกับงานวิจัย

บทที่ 3 กล่าวถึงการออกแบบระบบการส่งข้อมูลภาพจากด้านฝั่งส่งข้อมูลไปยังด้านฝั่งรับข้อมูล และวิธีการทำงานของไบต์อาเรย์

บทที่ 4 กล่าวถึงผลการทดลองต่างๆ

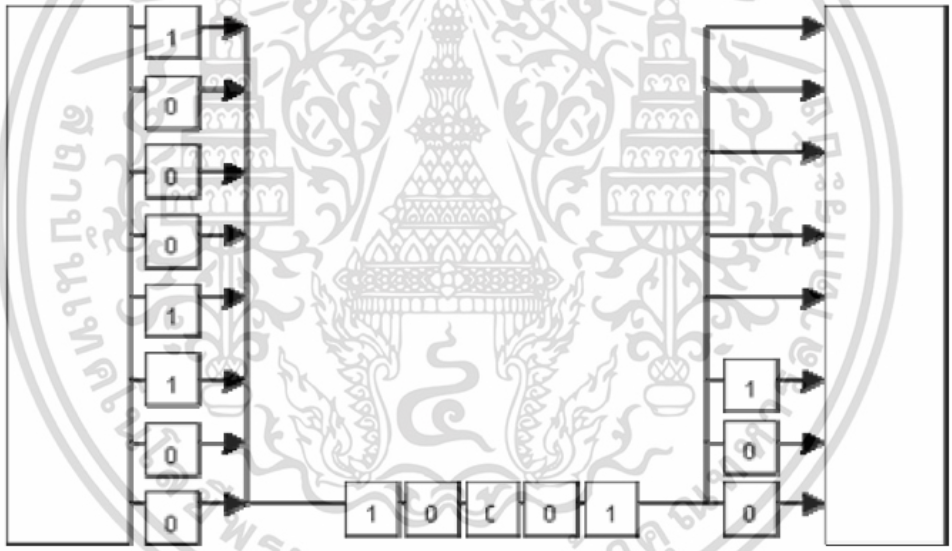
บทที่ 5 เป็นบทสรุปผลการวิจัยและข้อเสนอแนะ

บทที่ 2
ทฤษฎีที่เกี่ยวข้องกับงานวิจัย

ในบทนี้จะกล่าวถึงทฤษฎีและหลักการของพื้นฐานการส่งข้อมูลภาพ โดยในส่วนแรกจะกล่าวถึงการสื่อสารข้อมูลแบบอนุกรม รวมไปถึงสมรรถนะและการทดสอบคุณภาพของการรับส่งข้อมูล ต่อด้วยมีเดียสตรีมมิ่งที่จะกล่าวถึงระบบสีพื้นฐาน วิธีการบีบอัดไฟล์ภาพ ในส่วนหลังจะกล่าวถึงหลักการและแนวคิดของ OOP และส่วนท้ายสุดจะกล่าวถึงงานวิจัยและแนวความคิดที่มีความเกี่ยวข้องกับงานวิจัย

2.1 การสื่อสารข้อมูลแบบอนุกรม (Serial Transmission)

รูปแบบการส่งผ่านข้อมูลในลักษณะนี้ทุกบิตที่เข้ารหัสแทนข้อมูลหนึ่งตัวอักษรจะถูกส่งผ่านไปตามสายส่งเรียงลำดับกันไปทีละบิตในสายส่งเพียงเส้นเดียว ดังรูปที่ 2.1



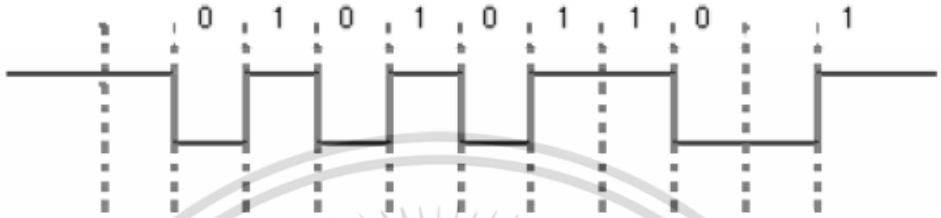
รูปที่ 2.1 การส่งข้อมูลแบบอนุกรม

จากรูปที่ 2.1 ตัวอักษรจะประกอบด้วย 8 บิต เรียงเป็นลำดับ ข้อมูลจะถูกส่งออกมาทีละบิตระหว่างต้นทาง และปลายทาง และปลายทางจะรวบรวมบิตเหล่านี้ทีละบิตจนครบ 8 บิต เป็น 1 ตัวอักษร จะเห็นว่าการส่งข้อมูลแบบนี้จะช้ากว่าแบบขนาน แต่ค่าใช้จ่ายจะถูกกว่าแบบขนาน ซึ่งเหมาะสำหรับการส่งระยะทางไกลๆ

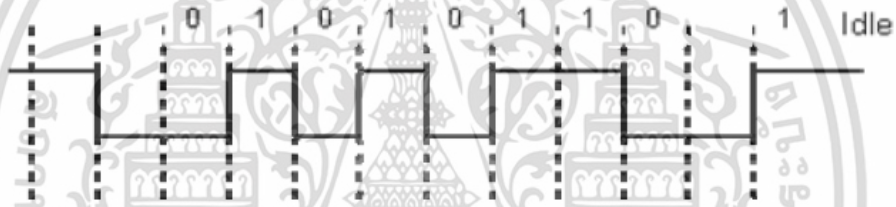
โดยทั่วไปแล้วการส่งข้อมูลนั้นจะประกอบไปด้วยกลุ่มของตัวอักษร ดังนั้นในการส่งข้อมูลแบบอนุกรมนี้จึงเกิดปัญหาขึ้นว่า แล้วต้นทางและปลายทางจะทราบได้อย่างไรว่า จะแบ่งแต่ละตัวอักษรตรงบิตใด จึงเกิดวิธีการสื่อสารข้อมูลขึ้น 2 แบบคือ การสื่อสารแบบอะซิงโคนัส (Asynchronous Transmission) และการสื่อสารแบบซิงโคนัส (Synchronous Transmission)

2.1.1 การสื่อสารแบบอะซิงโคนัส

เรียกอีกอย่างหนึ่งว่าเป็น การสื่อสารแบบระบุจุดเริ่มต้น และจุดสิ้นสุด (Start-Stop Transmission) ลักษณะของสัญญาณที่ใช้ในการติดต่อสื่อสารกันจะประกอบไปด้วย บิตเริ่มต้น (Start bit) บิตของข้อมูลที่สื่อสาร (Transmission data) จำนวน 8 บิต บิตตรวจข้อผิดพลาด (Parity bit) และบิตสิ้นสุด (Stop bit) สำหรับบิตตรวจสอบข้อผิดพลาดจะใช้หรือไม่ใช้ก็ได้ ดังนั้นสัญญาณจึงต้องประกอบ ด้วยส่วนประกอบอย่างน้อย 3 ส่วน ดังรูปที่ 2.2



รูปที่ 2.2 การสื่อสารแบบอะซิงโคนัสที่ไม่ได้ใช้พารีตี้บิต



รูปที่ 2.3 การสื่อสารแบบอะซิงโคนัสที่ใช้พารีตี้บิต

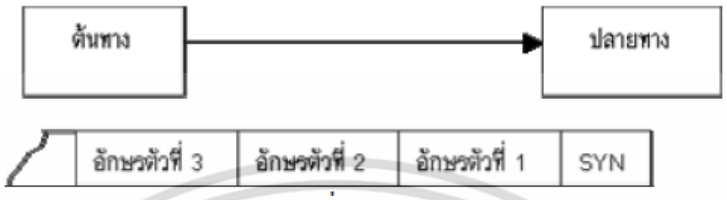
จากรูปที่ 2.3 จะเห็นว่าขณะที่ไม่มีข้อมูลส่งออกมาสถานะของการส่งจะเป็นแบบว่าง ซึ่งจะมีระดับของสัญญาณเป็น 1 ตลอดเวลา เพื่อความแน่ใจว่าปลายทาง หรือฝ่ายรับยังคงติดต่อกับต้นทาง หรือฝ่ายส่งอยู่ เมื่อเริ่มจะส่งข้อมูลสัญญาณของอะซิงโคนัสจะเป็น 0 หนึ่งช่วงสัญญาณนาฬิกา ซึ่งบิตนี้เราเรียกว่าบิตเริ่มต้น ตามหลังของบิตเริ่มต้นจะเป็นบิตข้อมูลสำหรับ 1 ตัวอักษร ตามหลังบิตข้อมูลก็จะเป็นบิตตรวจข้อผิดพลาด แล้วจะตามด้วยบิตสิ้นสุด ถ้าไม่ใช่บิตตรวจข้อผิดพลาด ตามหลังบิตข้อมูลก็จะเป็นบิตสิ้นสุดเลย หลังจากนั้นถ้าไม่มีข้อมูลส่งออกมาสัญญาณจะกลับไปอยู่ที่สถานะแบบว่างอีก เพื่อรอการส่งข้อมูลต่อไป

จะเห็นว่าการสื่อสารแบบอะซิงโคนัสนี้ มีลักษณะเป็นไปทีละตัวอักษร และสัญญาณที่ส่งออกมา มีบางส่วนเป็นบิตเริ่มต้น บิตสิ้นสุด และบิตตรวจข้อผิดพลาดทำให้ความเร็วในการส่งข้อมูลต่อวินาทีน้อยลงไป เนื่องจากต้อง สูญเสียช่องทางการสื่อสารให้กับบิตเริ่มต้น บิตสิ้นสุด และบิตตรวจข้อผิดพลาดตลอดเวลา การสื่อสารแบบอะซิงโคนัสนี้มักใช้ในการติดต่อระหว่างคอมพิวเตอร์กับอุปกรณ์รอบข้าง

2.1.2 การสื่อสารแบบซิงโคนัส

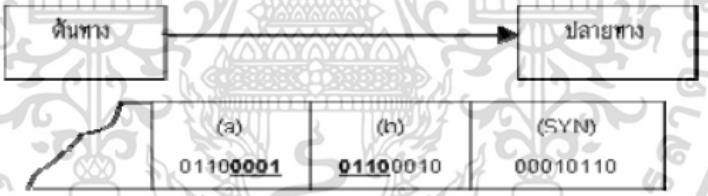
การสื่อสารแบบซิงโคนัสนี้มักใช้ในการติดต่อกันระหว่างคอมพิวเตอร์ ซึ่งจะทำการจัดกลุ่มของข้อมูลเป็นกลุ่มๆ และทำการสง ข้อมูลทั้งกลุ่มไปพร้อมกันในทีเดียว เราเรียกกลุ่มของข้อมูลนี้ว่า

บล็อกของข้อมูล (Block of Data) ซึ่งตัวอักษรตัวแรก และตัวถัดไปที่อยู่ในบล็อกเดียวกันจะไม่มีอะไรมาคั่นเหมือนอย่างแบบอะซิงโคนัส ที่ต้องใช้บิตเริ่มต้นและบิตสิ้นสุดคั่นทุกๆ ตัวอักษร แต่จะมีข้อมูลเริ่มต้นซึ่งเป็นลักษณะของบิตพิเศษที่ส่งมาเพื่อให้รู้ว่านั้นคือ จุดเริ่มต้นของกลุ่มตัวอักษรที่กำลังส่งเรียงกันเข้ามา เช่น อักขระซิง (SYN character) โดยที่อักขระซิงมีรูปแบบบิต คือ 00010110 ตัวอย่างของการส่งแสดงได้ดังรูปที่ 2.4



รูปที่ 2.4 การสื่อสารแบบซิงโคนัส

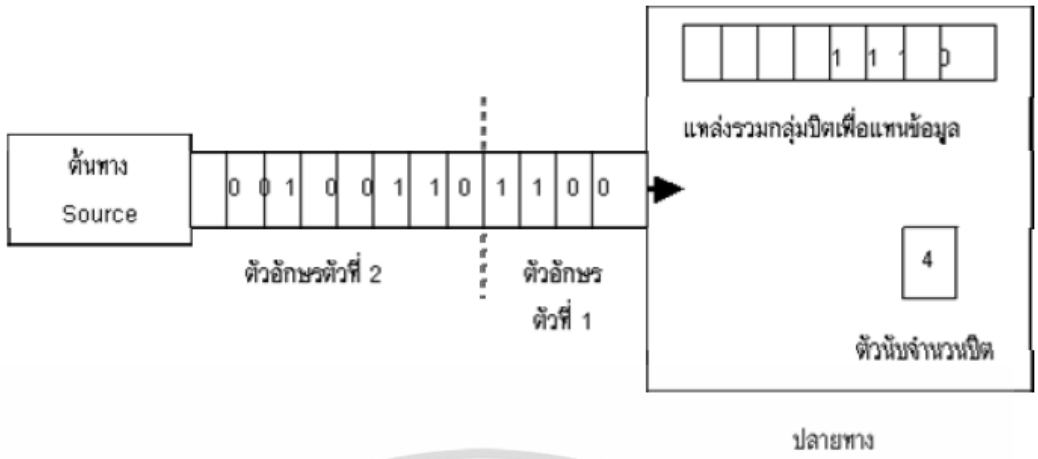
จากรูปที่ 2.4 เมื่อลายทางตรวจพบอักขระซิง หรือ 00010110 แล้วจะทราบได้ทันทีว่าบิตที่ตามมา คือบิตตัวอักษรแต่ละตัวแต่การใช้อักขระซิงเพียงตัวเดียว อาจเกิดข้อผิดพลาดได้ เช่น ถ้าเราส่งตัวอักษร b และตัวอักษร a ติดต่อกันไป ซึ่งตัวอักษร b มีรูปแบบบิตคือ 01100010 และตัวอักษร a มีรูปแบบบิตคือ 01100001 การส่งจะแสดงได้ดังรูปที่ 2.5



รูปที่ 2.5 ตัวอย่างการใช้อักขระซิงในการสื่อสารแบบซิงโคนัส

จะเห็นว่าเครื่องปลายทางจะตรวจพบอักขระซิงโคนัสระหว่างบิตของตัวอักษร b และตัวอักษร a ทำให้เข้าใจว่าบิตต่อไปจะเป็นบิตของกลุ่มข้อมูล ซึ่งจะทำให้การรับข้อมูลนั้นเกิดผิดพลาดขึ้นได้ ดังนั้นจึงแก้ปัญหาด้วยการใช้อักขระซิง 2 ตัวต่อกันเป็นลักษณะของบิตพิเศษที่บอกให้ทราบว่าเป็นจุดเริ่มต้นบิตของกลุ่มข้อมูล ตัวอย่างของการใช้อักขระซิง 2 ตัวในการสื่อสารแบบซิงโคนัส และการตัดแถวของบิตข้อมูลออกเป็นกลุ่มทีละ 8 บิต เพื่อแทนข้อมูลแสดงได้ดังรูปที่ 2.6 และรูปที่ 2.7

รูปที่ 2.6 ตัวอย่างการใช้อักขระซิง 2 ตัวในการสื่อสารแบบซิงโครัส



รูปที่ 2.7 การตัดแถวของบิตออกเป็นกลุ่มๆ ละ 8 บิต

2.1.3 มาตรฐานการส่งข้อมูลแบบอนุกรม

การสื่อสารแบบอนุกรม นับว่ามีความสำคัญต่อการใช้งานไมโครคอนโทรลเลอร์ (Microcontroller) มาก เพราะสามารถใช้แป้นพิมพ์ และจอภาพเป็นอินพุต(Input) และเอาต์พุต (Output) ในการติดต่อหรือควบคุมไมโครคอนโทรลเลอร์ ด้วยสัญญาณอย่างน้อย เพียง 3 เส้น คือ สายส่งสัญญาณ(TX ; Transmitter), สายรับสัญญาณ(RX ; Receiver) และ สายกราวด์(GND ; Ground)

2.1.4 พอร์ตอนุกรมอาร์เอส 232 (RS232 Serial Port)

พอร์ตที่ได้รับความนิยมในการสื่อสารแบบอนุกรม คือ พอร์ตอนุกรมอาร์เอส 232 ซึ่งแบ่งได้ 2 แบบ ตามลักษณะการใช้งาน ดังนี้

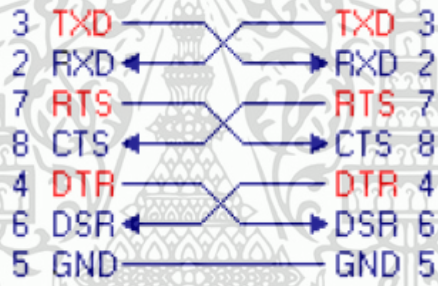
1. พอร์ตอนุกรมของคอมพิวเตอร์ จะเป็นคอนเนคเตอร์แบบดีบี 9 ตัวผู้ (Male)
2. พอร์ตอนุกรมของอุปกรณ์ภายนอก จะเป็นคอนเนคเตอร์แบบดีบี 9 ตัวเมีย (Female)

รูปที่ 2.8 พอร์ตอนุกรมดีบี 9 ตัวผู้ และตัวเมีย พร้อมแสดงหมายเลขขาของพอร์ต

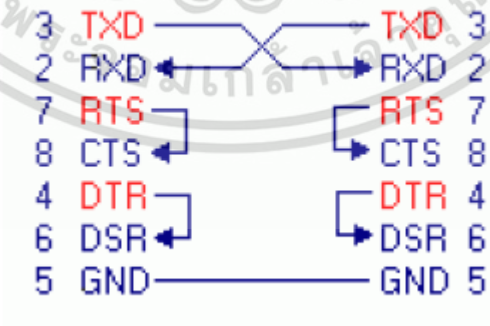
ตารางที่ 2.1 โครงสร้างขาของพอร์ตอนุกรมดีบี 9

ขา	คำอธิบาย	ชนิด
1	Data Carrier Detect (DCD)	Input
2	Received Data (RXD)	Input
3	Transmitted Data (TXD)	Output
4	Data Terminal Ready (DTR)	Output
5	Signal Ground (GND)	Input
6	Data Set Ready (DSR)	Input
7	Request To Send (RTS)	Output
8	Clear to Send (CTS)	Input
9	Ring Indicator (RI)	Input

2.1.5 การเชื่อมต่ออุปกรณ์ภายนอกเข้ากับคอมพิวเตอร์ด้วยสายดีบี 9



รูปที่ 2.9 การเชื่อมต่ออุปกรณ์ภายนอกผ่านดีบี 9 แบบ Null Modem



รูปที่ 2.10 การต่ออุปกรณ์ภายนอกผ่านดีบี 9 แบบ 3 เส้น

2.1.6 การทำงานของขาสัญญาณดีบี 9

1. TXD เป็นขาที่ใช้ส่งข้อมูล

2. RXD เป็นขาที่ใช้รับข้อมูล

3. DTR แสดงสภาวะพอร์ตว่าเปิดใช้งาน ,DSR ตรวจสอบว่าพอร์ต ที่ติดต่อด้วยเปิดอยู่หรือไม่ เมื่อเปิดพอร์ตอนุกรม ขา DTR จะ ON เพื่อให้อุปกรณ์ได้รับทราบว่า ต้องการติดต่อด้วย ในขณะ เดียวกันก็จะตรวจสอบขา DSR ว่าอุปกรณ์พร้อมหรือไม่

4. RTS แสดงสภาวะพอร์ตว่าต้องการส่งข้อมูล ,CTS ตรวจสอบว่าพอร์ตที่ติดต่ออยู่ต้องการส่งข้อมูลหรือไม่ เมื่อต้องการส่งข้อมูลขา RTS จะ ON และจะส่งข้อมูลออกที่ขา TXD เมื่อส่งเสร็จก็จะ OFF ในขณะเดียวกันก็จะตรวจสอบขา CTS ว่าอุปกรณ์ต้องการที่จะส่งข้อมูลหรือไม่

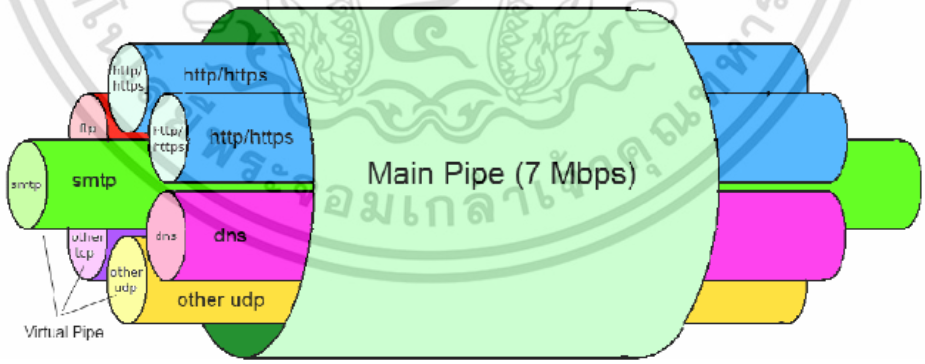
5. GND ขากราวด์

2.2 สมรรถนะและการทดสอบคุณภาพของเครือข่าย

การตรวจสอบระบบเครือข่ายมีอยู่ด้วยกันสองลักษณะ คือ เฝ้าสังเกตการณ์ เช่น การดูพฤติกรรมของ Traffic Flow ณ จุดต่างๆ ของเราเตอร์ด้วย SNMP Sniffer หรือ MRTG เป็นต้นการตรวจสอบลักษณะนี้เป็นแบบ Passive ซึ่งจะไม่มีการรบกวนการทำงานของระบบ ในขณะเดียวกันการตรวจสอบในลักษณะแบบ Active จำเป็นต้องส่งข้อมูลบางแพ็กเก็ต ให้วิ่งผ่านเข้าไปในระบบแล้วดูผลการตอบสนองของระบบเครือข่าย การตรวจสอบระบบเครือข่ายไม่ว่าจะด้วยวิธีใดก็ตาม มักจะใช้ค่าพารามิเตอร์ดังต่อไปนี้

2.2.1 ทรูพุต (Throughput)

ทรูพุต (Throughput) คือ ปริมาณข้อมูลที่ส่งได้จริงต่อหน่วยเวลา สมมติให้สายสัญญาณหนึ่งมีความกว้างช่องสัญญาณในการสื่อสาร (Bandwidth) เป็น 7 Mbps เพื่อใช้สำหรับให้บริการ Service ต่างๆ แสดงดังรูปที่ 2.11



รูปที่ 2.11 การจัดสรรช่องสัญญาณในการสื่อสาร

จากรูปที่ 2.11 พบว่าช่องสัญญาณในการสื่อสารมีขนาด 7 Mbps ถูกซอยออกเป็นช่องสัญญาณย่อยๆ สำหรับให้บริการ Service ต่างๆ เช่น ในกรณีการใช้งาน web ปริมาณข้อมูลที่สามารถส่งผ่านได้สูงสุด คือ 2 Mbps หมายความว่า Throughput หรือปริมาณข้อมูลที่ส่งได้จริงต่อ

หน่วยเวลาสำหรับการใช้งาน web อาจน้อยกว่า หรือเท่ากับ 2 Mbps ในกรณีการใช้งาน Service อื่นๆ ที่เหลือก็เช่นกัน Throughput จริงจะมีได้ไม่เกิน 1 Mbps ตามที่ได้กำหนดไว

2.2.2 ดีเลย (Delay)

ความล่าช้าหรือดีเลย์ (Latency or Delay) คือ ความหน่วงเวลาที่เกิดขึ้นเมื่อแพ็กเก็ตข้อมูลเดินทางผ่านสถานีต่างๆ แบบจุดต่อจุดในเครือข่ายระหว่างต้นทางที่แพ็กเก็ตข้อมูลถูกส่งมาจนถึงปลายทาง ค่าดีเลย์ที่เกิดสามารถจำแนกออกมาได้ดังต่อไปนี้

2.2.2.1 เวลาที่ถูกหน่วงไปกับการประมวลผล (Processing Delay)

เวลาที่ถูกหน่วงไปกับการประมวลผล คือ เวลาที่ต้องใช้ในการประมวลผลข้อมูล เช่น จากการเข้า/ถอดรหัส เพื่อแปลงรูปแบบของสัญญาณอะนาลอกเป็นดิจิตอลที่เกิดขึ้นภายในอุปกรณ์ซึ่งอาจเป็นเราท์เตอร์ (Nodal Processing)

2.2.2.2 เวลาที่ถูกหน่วงไปกับการกระจายข้อมูล (Propagation Delay)

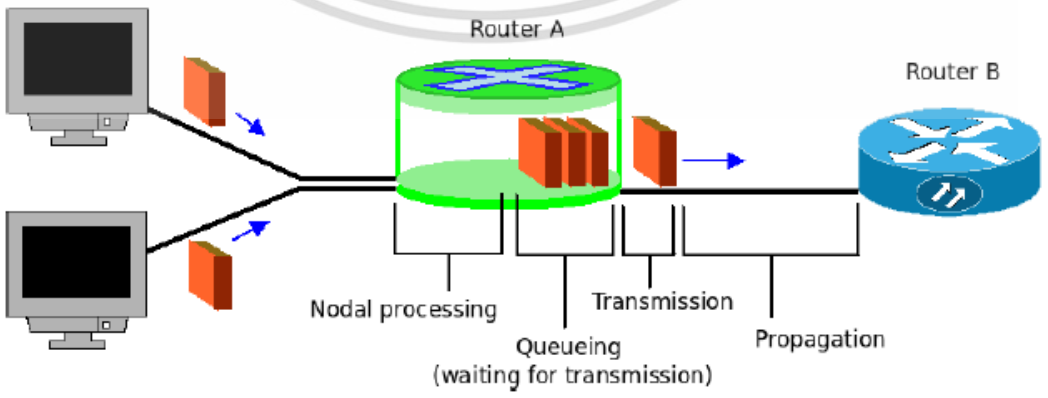
เวลาที่ถูกหน่วงไปกับการกระจายข้อมูล คือ เวลาที่ข้อมูลใช้ในการเดินทางผ่านสื่อแบบจุดต่อจุด (Hob by Hob) ถึงแม้ว่าข้อมูลที่เดินทางด้วยความเร็วที่ใกล้ความเร็วแสง หากแพ็กเก็ตเดินทางอ้อมโดยไม่จำเป็น ค่าหน่วงเวลาชนิดนี้ก็จะเกิดขึ้น ให้ d คือ ระยะห่างระหว่างเราท์เตอร์ และ s คือความเร็วของลิงค์ระหว่างเราท์เตอร์ ค่าหน่วงเวลาการกระจายข้อมูล คือ d/s Propagation Delay จะขึ้นอยู่กับลักษณะเฉพาะตัวของสื่อที่ใช้ในการส่งข้อมูลนั้นโดยทั่วไปค่าที่ได้จะอยู่ในหน่วยมิลลิวินาที

2.2.2.3 เวลาที่ถูกหน่วงในคิว (Queuing Delay)

เวลาที่ถูกหน่วงในคิว คือ เวลาที่แพ็กเก็ตต้องคอยเพื่อเตรียมส่งต่อไปยังลิงค์ ระยะเวลาของคิวนี้ขึ้นอยู่กับจำนวนของแพ็กเก็ตที่เดินทางมาถึงก่อนหน้านี้และยังคงค้างอยู่ในคิว เพื่อรอเวลาสำหรับการส่งผ่านไปยังลิงค์ต่อไป โดยทั่วไปค่าที่ได้จะอยู่ในระดับไมโครวินาทีถึงมิลลิวินาที

2.2.2.4 เวลาที่ถูกหน่วงเมื่อส่งผ่าน (Transmission Delay)

เวลาที่ถูกหน่วงเมื่อส่งผ่าน คือ เวลาที่เกิดจากอัตราของการส่งข้อมูลต่อแพ็กเก็ต ถ้าให้ L แทนความยาวของแพ็กเก็ตในหน่วยบิต และ R แทนอัตราการส่งผ่านของลิงค์ระหว่างเราท์เตอร์หรือโหนดทั้งสองในหน่วยบิตต่อวินาที แล้ว Transmission Delay คือ L/R โดยทั่วไปค่าที่ได้จะอยู่ในระดับไมโครวินาทีถึงมิลลิวินาที



รูปที่ 2.12 การเกิดดีเลย์ ณ ส่วนต่างๆ ของระบบ

ถ้าสมมติให้ d_{proc}, d_{queue}, d_{trans}, และ d_{prop} แทน processing, queuing, transmission และ propagation ดีเลย์ แล้ว ค่าดีเลย์ของโหนด (Nodal Delay) แสดงดังสมการที่ 2.1

$$d_{nodal} = d_{\text{proc}} + d_{\text{queue}} + d_{\text{trans}} + d_{\text{prop}} \quad (2.1)$$

ในความเป็นจริงแพ็กเก็ตเดินทางจากต้นทางไปยังปลายทางในลักษณะ End-to-End มีการเดินทางผ่านเราท์เตอร์ในแต่ละเส้นทาง ให้ N แทนจำนวนเราท์เตอร์ที่แพ็กเก็ตวิ่งผ่านดังนั้นเราท์เตอร์ระหว่างต้นทางและปลายทางที่แพ็กเก็ตวิ่งผ่าน คือ N-1 สมมติให้ในระบบเครือข่ายไม่มีความคับคั่ง (คิวดีเลย์ที่เกิดตลอดเส้นทางของแพ็กเก็ตที่วิ่งผ่านน้อยมาก) แล้ว d_{proc}, d_{trans}, และ d_{prop} แทน processing, transmission และ propagation ดีเลย์บนแต่ละลิงค์ ดังนั้นค่าดีเลย์สะสมของแต่ละโหนด (End-to-End delay) แสดงดังสมการที่ 2.2

$$d_{end-end} = N(d_{\text{proc}} + d_{\text{trans}} + d_{\text{prop}}) \quad (2.2)$$

ซึ่ง d_{trans}=L/R เมื่อ L คือ ขนาดของแพ็กเก็ต และ R แทนอัตราการส่งผ่านของลิงค์

2.2.2.5 จิตเตอร์ (Jitter)

จิตเตอร์ (Jitter) คือ ค่าความต่างหรือความคลาดเคลื่อนของเวลา (Time-base error) โดยปกติแล้วในระบบเครือข่าย Jitter จะเป็นค่าความเปลี่ยนแปลงของดีเลย์ที่เกิดขึ้นเนื่องจากการ
เดินทางของแพ็กเก็ตข้อมูลแต่ละแพ็กเก็ตบนเครือข่ายอินเตอร์เน็ตแม้จะมีปลายทางเดียวกันอาจใช้เส้นทางที่แตกต่างกันและสภาพความคับคั่งของเครือข่ายที่เปลี่ยนแปลงได้ตลอดเวลา อาจมีผลทำให้เกิดจิตเตอร์ได้เช่นกัน

2.2.2.6 แพ็กเก็ตที่เสียไป (Packet Lost)

เนื่องด้วยภายในอุปกรณ์ที่แพ็กเก็ตข้อมูลวิ่งผ่านมีขนาดของคิวที่จำกัด ซึ่งขึ้นอยู่กับการออกแบบของอุปกรณ์ หมายความว่าแพ็กเก็ตที่เดินทางมาถึงช้าอาจถูกแทนที่ด้วยแพ็กเก็ตที่ตามมาถึงก่อนหน้าหากพบว่า ณ ขณะที่แพ็กเก็ตเดินทางมาถึงโหนด หรือ เราท์เตอร์ แต่เกิดคิวเต็ม หรือในกรณีที่อุปกรณ์ไม่สามารถประมวลผลได้ทันหรือไม่สามารถพักข้อมูลแพ็กเก็ตได้ อุปกรณ์อาจตัดสินใจละทิ้งชุดแพ็กเก็ตดังกล่าวไป

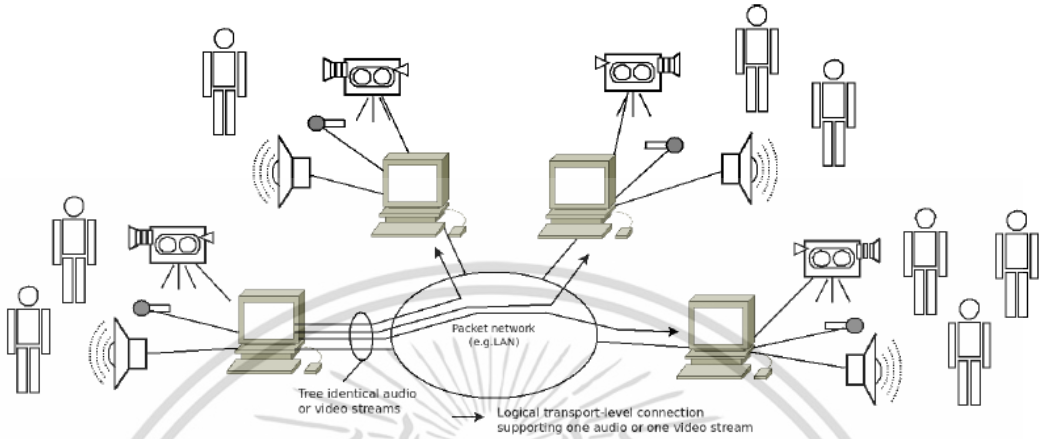
2.3 มีเดียสตรีมมิ่ง (Media Streaming)

มีเดียสตรีมมิ่ง (Media Streaming) คือ มัลติมีเดียชนิดใดๆ ที่มีการส่งผ่านโดยตรงจากแหล่งที่มาไปยังส่วนของการประมวลผลเพื่อแสดงผลในลักษณะเวลาจริง (Real time) และมีการประมวลผลอย่างต่อเนื่องโดยไม่ต้องอาศัยพื้นที่เก็บข้อมูล(Storage) เพื่อบันทึกข้อมูลมีเดียทั้งหมดก่อนการประมวลผลเพื่อเล่นในภายหลัง

2.3.1 การเชื่อมต่อ Point-to-Point หลายการเชื่อมต่อ

จำนวนการเชื่อมต่อแบบสองทิศทางเกิดขึ้นระหว่างคู่สนทนาในระบบ Video Conference ซึ่งการเชื่อมต่ออาศัยหลักการของแพ็กเก็ตนั้นแตกต่างจากการเชื่อมต่อที่เกิดขึ้นในแบบของวงจรทางกายภาพที่มีใช้งานบนวงจรเช่า (Leased Line) หรือบนเครือข่าย Circuit-Switched แต่การติดต่อ

จริง คือ แบบลอจิกจะเกิดขึ้นระหว่างต้นทางและปลายทาง แสดงดังรูปที่ 2.13 การเชื่อมต่อแบบลอจิกทั้ง Video และ Audio แบ่งแพ็กเก็ตตามจำนวนข้อมูลที่แอพพลิเคชันส่งออกไปให้วิ่งผ่านบนระบบเครือข่ายแบบ Point-to-Point หลายการเชื่อมต่อ



รูปที่ 2.13 การเชื่อมต่อ Video Conference แบบ Point-to-Point หลายการเชื่อมต่อ

2.3.2 ระบบสีพื้นฐานของคอมพิวเตอร์

2.3.2.1 รูปแบบสีแบบอาร์จีบี

เป็นระบบสีพื้นฐานของคอมพิวเตอร์ที่ใช้ในการแสดงผล โดยจุดย่อยของภาพ(Pixel) จะประกอบด้วยค่าสี 3 ค่า คือ แดง (R ; Red) เขียว (G ; Green) และน้ำเงิน (B ; Blue) การผสมสีทั้งสามนี้ด้วยค่าต่างๆ กัน จะก่อให้เกิดสีที่แตกต่างกัน โดยคอมพิวเตอร์จะเก็บค่าสีนี้แยกกัน โดยใช้ขนาดข้อมูล 1ไบต์ต่อ 1 สี ทำให้ค่าของสีนั้นมีได้ 256 ระดับ และผสมได้สีทั้งหมด 16 ล้านสี สามารถแสดงโมเดลของสีได้ดังรูป 2.14

รูปที่ 2.14 รูปแบบของสีแบบอาร์จีบี

2.3.2.2 รูปแบบสีแบบระดับสีเทา

ภาพระดับสีเทา หรือ Gray Scale เป็นภาพที่ประกอบไปด้วยค่าของพิกเซลที่เป็นค่าเฉดสีของสีเทาซึ่งได้จากระดับสีจากการเปลี่ยนสีจากสีดำไปเป็นสีขาว โดยจะถือสีขาวเป็นสีที่มีแก่ที่สุดและสีดำเป็นสีที่อ่อนที่สุด เนื่องจากค่าของพิกเซลที่เป็นสีขาวจะมีค่ามากกว่าค่าของสีดำ สิ่งที่ภาพระดับสีเทาแตกต่างกับภาพขาว-ดำคือ ในภาพขาว-ดำ จะเก็บข้อมูลโดยมีอยู่สองสีคือสีขาวและสีดำ แต่ในภาพระดับสีเทาจะมีความละเอียดมากกว่า กล่าวคือจะเก็บค่าของเฉดสีเทาที่อยู่ระหว่างการเปลี่ยนจากสีดำเป็นสีขาว

การนำตัวเลขมาแทนค่าของพิกเซลในภาพระดับสีเทานั้น ใช้เป็นเปอร์เซ็นต์คือ 0%(สีดำ) ถึง 100%(สีขาว) แต่เมื่อนำมาใช้งานร่วมกับเครื่องพิมพ์จะมีการทำงานที่ตรงกันข้าม กล่าวคือจะต้องทำการกลับค่าของเปอร์เซ็นต์จึงจะพิมพ์ได้ตรงตามที่ต้องการ เนื่องจากการอ่านค่าของเครื่องพิมพ์คือ 0% คือ ไม่ปล่อยหมึกออกมา ซึ่งหมายความว่าส่วนนั้นจะเป็นสีขาว ส่วน 100% เครื่องพิมพ์จะปล่อยหมึกออกมา ซึ่งนั่นคือสีดำ ในทางคอมพิวเตอร์ เดิมจะมีการเก็บในลักษณะของเลขฐานสองจำนวนสี่บิต สามารถเก็บความแตกต่างได้ 16 ระดับ แต่ในปัจจุบันนี้ได้มีการเพิ่มข้อมูลที่เก็บโดยเพิ่มเป็นเลขฐานสองจำนวน 8 บิต (0-255) ซึ่งจะสามารถเก็บค่าความแตกต่างได้ 256 ระดับ แสดงได้ดังรูป 2.15

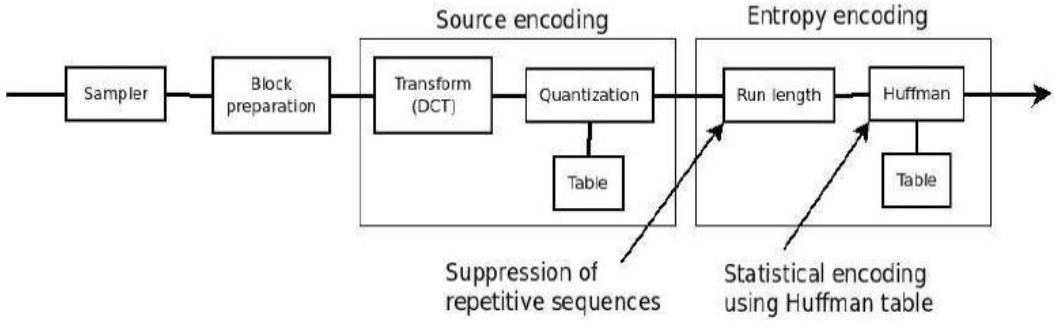
รูปที่ 2.15 รูปแบบสีแบบระดับสีเทา

2.3.3 การบีบอัดภาพและวีดีโอ (Image and Video Compression)

ข้อมูลในรูปแบบของสื่อมัลติมีเดีย นิยมแปลงให้อยู่ในรูปของบิตเรท ซึ่งสะดวกต่อการนำไปใช้งานบนเครือข่ายแต่ก็มีข้อจำกัดในเรื่องของปริมาณการจัดเก็บและบันทึก เช่น CD-ROM มีความจุข้อมูล 648 Mbytes โดยประมาณ และสามารถบันทึกได้ 72 นาที (สำหรับ Audio) โดยไม่มีการบีบอัดหรือย่อข้อมูลใดๆ CD-ROM ชนิดเดียวกันนี้สามารถบันทึกสื่อทีวีดิจิตอลคุณภาพระดับสตูดิโอโดยไม่บีบอัดได้ประมาณ 30 วินาที หากใช้ภาพขนาดความละเอียด 2000 x 2000 หรือ 4 ล้านพิกเซล อาจทำให้แต่ละไฟล์ภาพที่จัดเก็บมีขนาดใหญ่ถึง 12 Mbytes โดยประมาณ จึงต้องอาศัยเทคนิค ในการบีบอัด

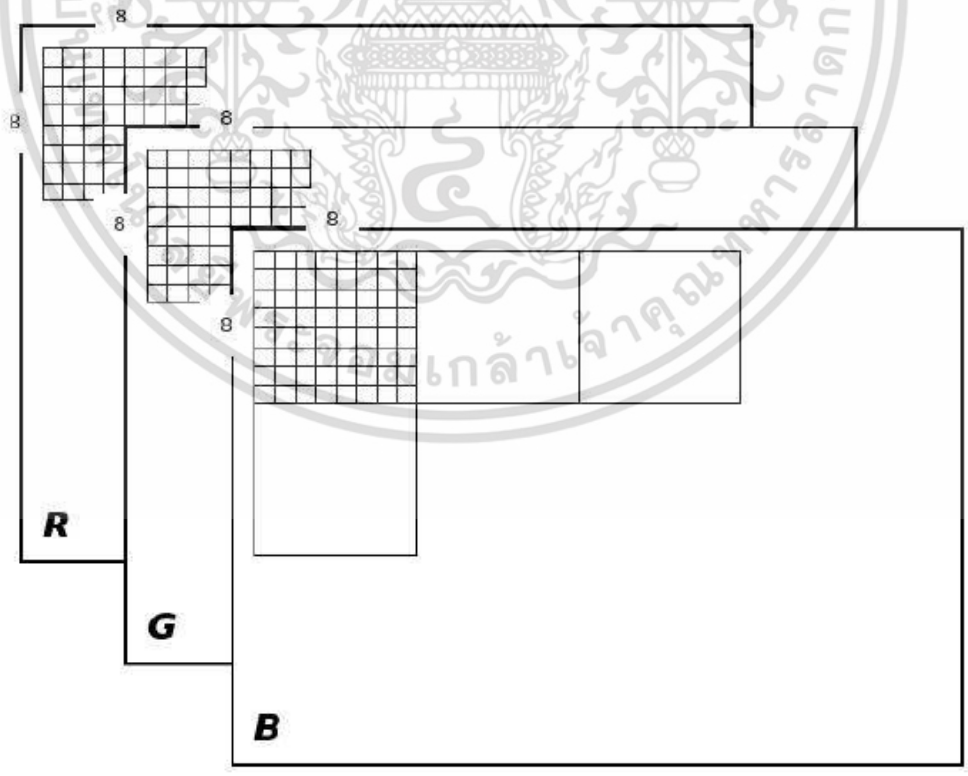
2.3.3.1 มาตรฐานการบีบอัดเจเพ็ก (JPEG)

การบีบอัดแบบ JPEG (Joint Photographic Expert Group) คือ มาตรฐานการบีบอัดภาพระดับสีเทา (Gray-Scale) หรือ ภาพสี (Color) โดยมีลำดับในการแปลง Discrete Cosine Transform , การทำ Quantization, Run-length และเทคนิคในการเข้ารหัสด้วย Huffman



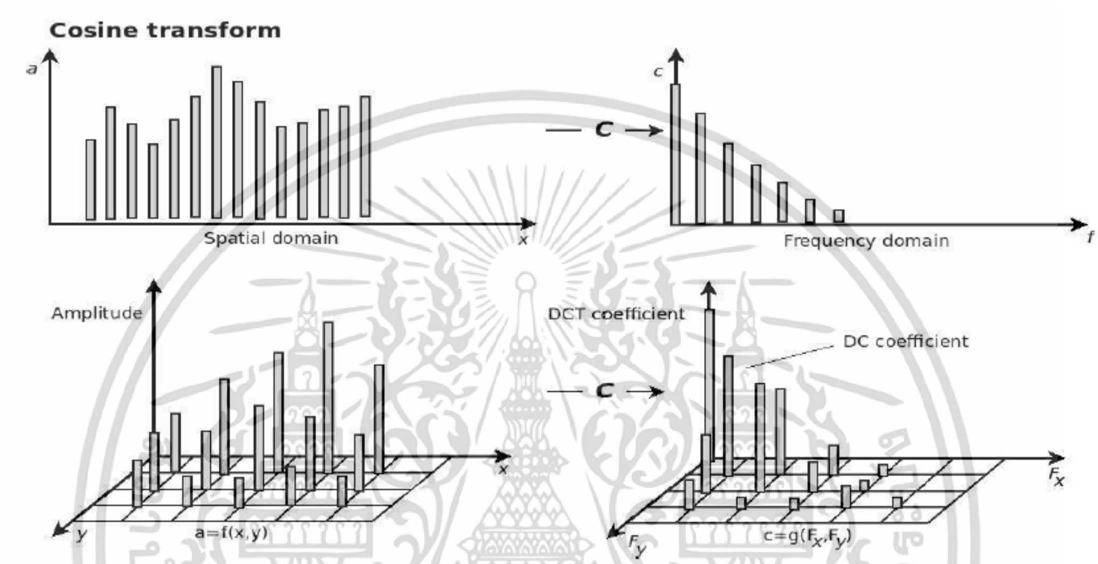
รูปที่ 2.16 ลำดับกระบวนการเข้ารหัสภาพ Continuous-Tone ด้วย JPEG

มีหลายวิธีการด้วยกันในการแทนค่าสีในแต่ละจุดของภาพ เช่น โมเดล RGB, YUV (European Television), YIQ (North American and Japanese Television), YCrCb ซึ่ง R, Y, Q หรือ Cb เป็นตัวอย่างขององค์ประกอบภาพ สามารถแทนค่าองค์ประกอบดังกล่าวได้ด้วยเมตริกซ์ เมื่อขนาดของภาพที่จะทำการบีบอัด คือ ตัวแปร JPEG ขนาด 8 บิตมีค่าตัวเลขที่เป็นองค์ประกอบของความสว่างสูงสุด คือ 255 (ระดับ 0-255) เมื่อพิจารณาภาพสีที่จะทำการบีบอัดให้แทนด้วยองค์ประกอบ R, G, B และองค์ประกอบแต่ละส่วนแบ่งเป็บล็อกขนาด 8×8 พิกเซล แสดงดังรูปที่ 2.17

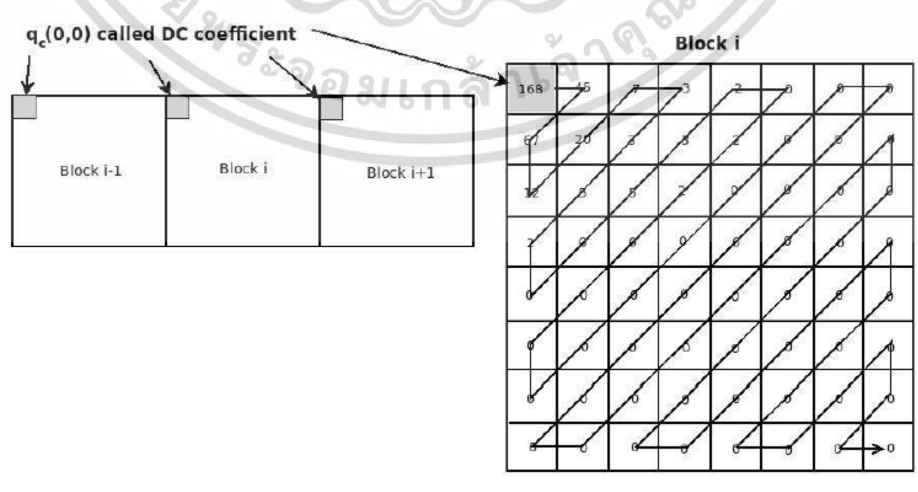


รูปที่ 2.17 การสร้างบล็อกขนาด 8×8 พิกเซล ในองค์ประกอบย่อย R, G และ B

ในแต่ละบล็อกขององค์ประกอบย่อย ประกอบไปด้วย R, G และ B มีถึง 64 ค่า ซึ่ง Amplitude ของสัญญาณที่ถูก Sample ในแต่ละองค์ประกอบ ดังนั้น Amplitude นี้สามารถใช้ฟังก์ชัน a = f(x,y) เป็น Discrete Cosine Transform โดยที่ x และ y คือโดเมนในสองมิติที่ใช้ระบุตำแหน่งข้อมูลในอาร์เรย์ a = f(x,y) เขียนให้อยู่ในรูป c = g(Fx,Fy) โดยที่ c เป็นสัมประสิทธ์ (Coefficient) ,Fx และ Fy คือ ลำดับความถี่เฉพาะของแต่ละตำแหน่ง ค่าสัมประสิทธิ์ DCT(Discrete Cosine Transform) แต่ละค่า คือ ความถี่เฉพาะซึ่งมีขนาดไม่เกิน Amplitude ของสัญญาณที่ถูก sampled ในตำแหน่ง (x, y) แสดงดังรูปที่ 2.18

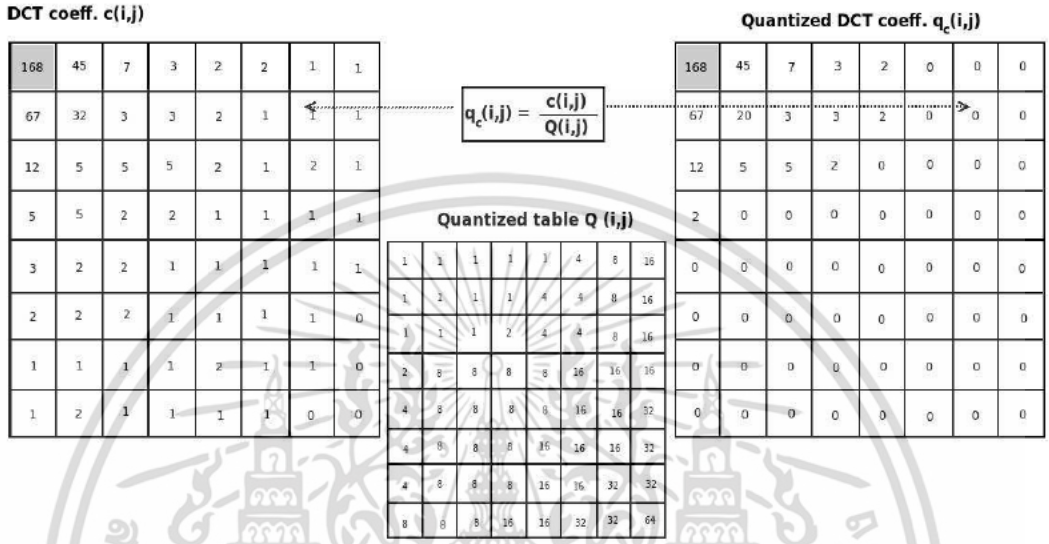


รูปที่ 2.18 Amplitude ของสัญญาณไปสู่ Discrete Cosine Transform โดย x และ y ระบุตำแหน่ง
ข้อมูลในโดเมนสองมิติ



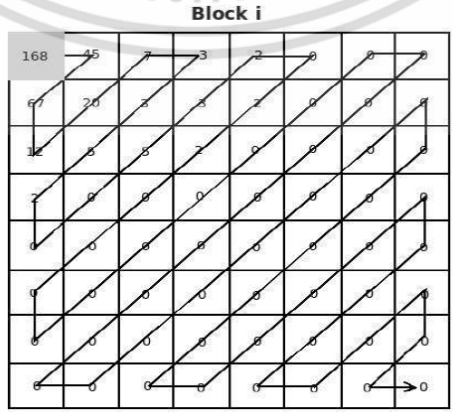
รูปที่ 2.19 DCT ใช้กับบล็อกขนาด 8×8 ในทุกองค์ประกอบของภาพ

ขั้นตอนต่อไป คือ Quantization กระบวนการทำ Quantization คือ สัดส่วนของสัมประสิทธ์ DCT แต่ละค่าต่อค่าที่อยู่ในตาราง Quantization Table ดังรูปที่ 2.20 และค่าที่อยู่ในตาราง Quantization Table นี้มีขนาดและมิติเป็น 8×8 เช่นเดียวกันกับขนาดของบล็อก แต่ละค่า (Elements) ในตารางเป็นจำนวนเต็ม (1 ถึง 255) จุดประสงค์ของการทำQuantization เพื่อให้ได้ตัวแทนจากข้อมูลที่มีอยู่มาก ขจัดข้อมูลที่ซ้ำซอนและเกินความจำเป็นในการนำไปใช้งาน



รูปที่ 2.20 ตัวอย่างการทำ Quantization

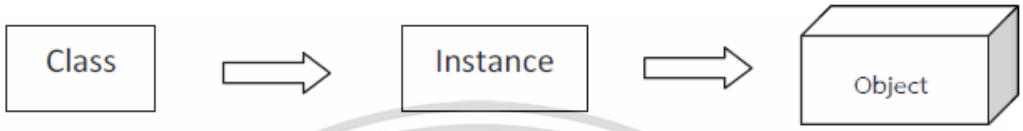
ค่าสัมประสิทธ์ที่ผ่านการ Quantization ถูกลำดับใหม่ในทิศทางแบบ Zig-Zag รูปที่ 2.21 เพื่อเพิ่มความน่าจะเป็นของค่าสัมประสิทธ์ที่เรียงตัวอย่างต่อเนื่อง ทำให้กระบวนการ Run-length ที่จะเกิดขึ้นในกลุ่มกรรมวิธี Entropy encoding มีประสิทธิภาพในการบีบอัดข้อมูลสูงโดยไม่สูญเสียข้อมูล (Lossless Data Compression) และการเข้ารหัสแบบ Huffman encoding ในขั้นตอนสุดก่อนนำภาพไปใช้งาน



รูปที่ 2.21 ทิศทางการลำดับแบบ Zig-Zag

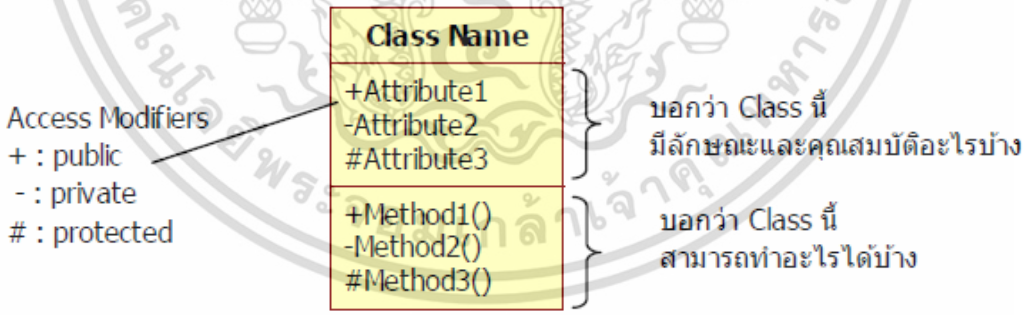
2.4 หลักการและแนวคิดของ OOP

OOP หรือ (Object Oriented Programming) เป็นการสร้างหรือการพัฒนาโปรแกรมเชิงวัตถุวิธี บนพื้นฐานแนวคิดที่ว่า วัตถุ (Object) คือ สิ่งที่มีคุณสมบัติ มีรูปลักษณ์ และสัมผัสได้ (การเลียนแบบสิ่งที่มีรูปธรรมในสภาพของซอฟต์แวร์) Object จะเกิดและมีตัวตนได้ต้องมาจากต้นแบบ Prototype ของ Object นั้น เรียกว่า Class ดังนั้น Class แบบใดจะเปรียบเสมือนพิมพ์เขียวของวัตถุชนิดนั้นๆ รูปที่ 2.22 แสดงการเกิด Object จาก Class



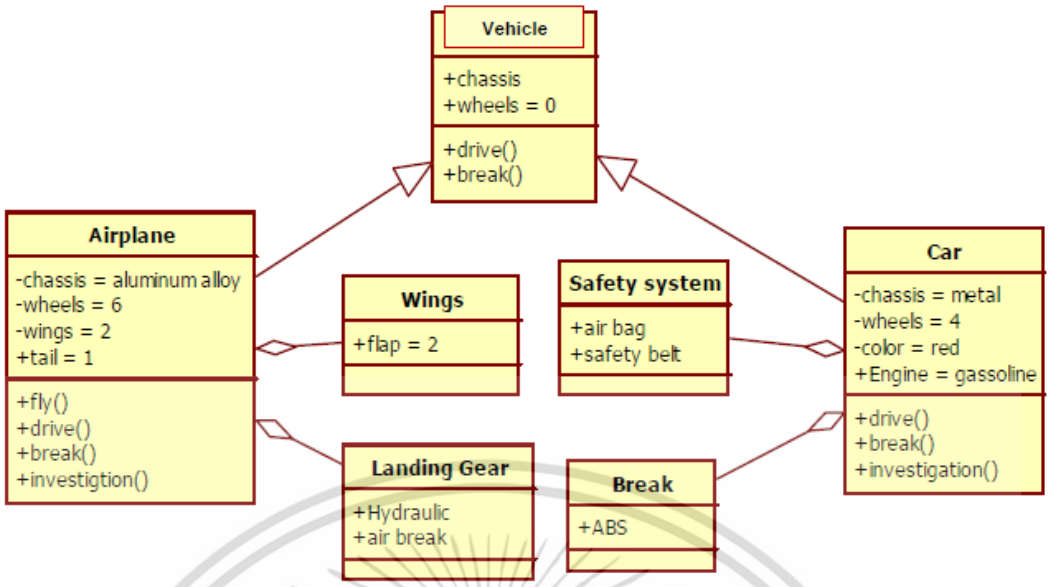
รูปที่ 2.22 การเกิดของ Object

Object ที่เกิดจากการ Instance อาศัยแนวคิด 3 ประการ คือ ความสามารถในการห่อหุ้มวิธีการและข้อมูลไว้ภายใน (Encapsulation), สามารถถ่ายทอดคุณสมบัติ (Inheritance) และสามารถแปรคุณสมบัติรวมทั้งพฤติกรรมของมันเอง (Polymorphism) สำหรับการนิยาม Class ซึ่งเป็นต้นแบบของ Object ใช้แนวคิดที่ว่า Object มีลักษณะหรือคุณสมบัติเฉพาะ (Attribute) เป็นอย่างไร และ Object นำไปใช้ประโยชน์หรือนำไปใช้งานด้วยวิธีการ (Method) ใดได้บ้าง แสดงส่วนประกอบของ Class ดังรูปที่ 2.23



รูปที่ 2.23 ส่วนประกอบของ Class

Class มีกลไกที่ควบคุมการเข้าถึง (Access Modifiers) ไปยัง Attribute และ Method ต่างๆ โดยใช้คำนำหน้าคุณสมบัติและวิธีการ เช่น Public (เข้าถึงได้ทุกที่), Private (เข้าถึงเฉพาะภายใน Class) และProtected (เข้าถึงได้แค่ภายใน Class และ Class ที่ถูกถ่ายทอดออกไป) เป็นคีย์เวิร์ดนั่นคือ ระดับของการเข้าถึงข้อมูลภายใน Class ตามลำดับ



รูปที่ 2.24 ความสัมพันธ์ของ Class ต่างๆ

Class ต่างๆ สามารถเขียนเพื่อแสดงให้รู้ถึงความสัมพันธ์ กับ Class อื่นๆ ได้ด้วย ClassDiagram ดังรูปที่ 2.24 เมื่อ Airplane และ Car เป็น Class ที่ได้รับการสืบทอด (Inherited) มาจากที่เดียวกันคือ Vehicle สำหรับ Wings และ Landing Gear คือ ส่วนประกอบย่อยของ Airplaneเช่นเดียวกับ Safety system และ Break เป็นส่วนประกอบย่อยของ Car

2.5 งานวิจัยและแนวความคิดที่มีความเกี่ยวข้องกับงานวิจัย

งานวิจัยของ Xin-Gang Liu, Kook-Yeol Yoo และ Kwang Deok Seo เป็นงานวิจัยที่เกี่ยวข้องกับวิทยานิพนธ์นี้ ซึ่งได้เสนอแนวคิดและวิธีผสานวีดีโอ สำหรับระบบ Video Conference ของผู้เข้าร่วม (Participant) จากแหล่งที่มาต่างๆ โดยใช้เทคนิค Hybrid Video Mixing โดยนำข้อดีจากวิธีการผสานระหว่าง Pixel-Domain Transcoder Mixing และวิธี Bit Stream-Domain Mixing เข้าด้วยกัน

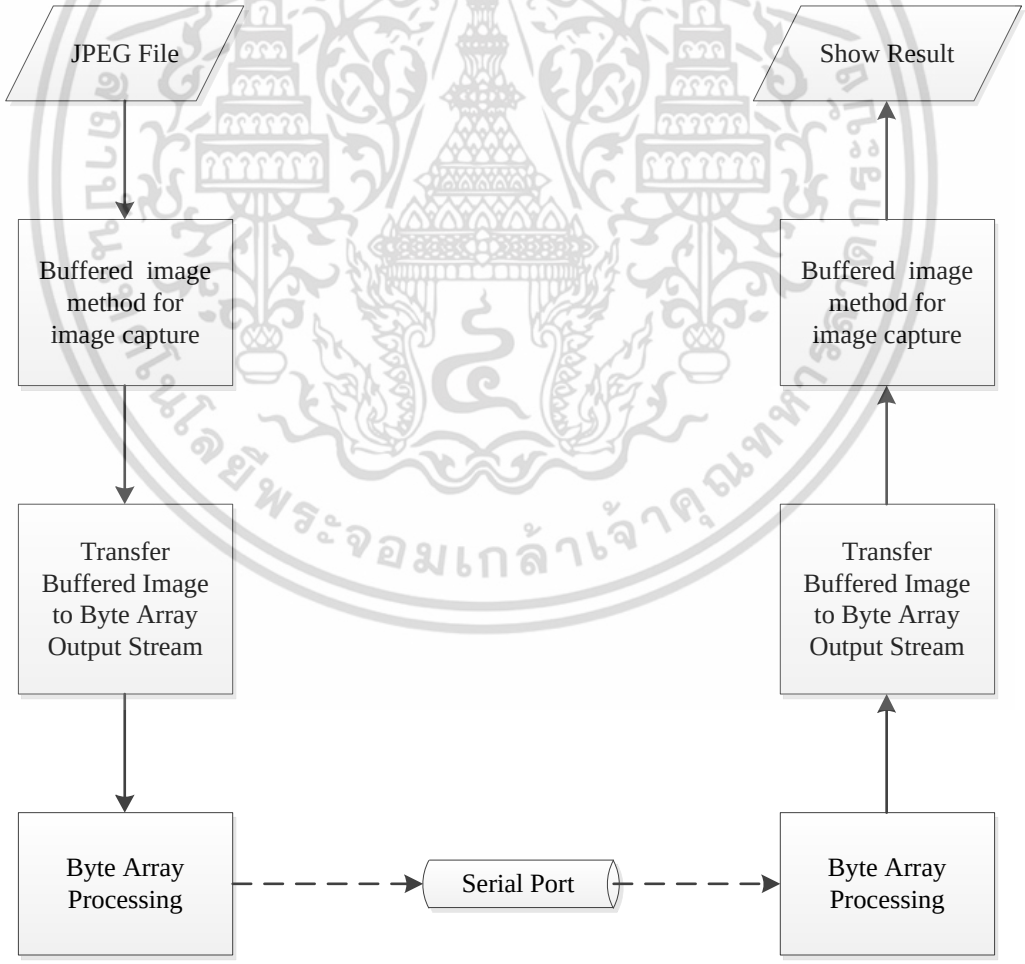
นอกจากนี้ยังมีงานวิจัยอีกหนึ่งงานวิจัยที่มีส่วนช่วยและเป็นแนวทางสำหรับการค้นคว้าให้กับวิทยานิพนธ์นี้เป็นอย่างมาก คือ งานวิจัยของ L. Mengual, J. Bobadilla, R. Caballero และ G.Hernández ซึ่งได้นำเสนอวิธีการออกแบบแอพพลิเคชันสำหรับ Video Conference และมีการเปรียบเทียบ เครื่องมือที่ช่วยในการสร้างและพัฒนาระหว่าง JMF (Java Media Framework) และ VIC (Video Conferencing Tool) สำหรับหลักการต่างๆ จะขออธิบายในบทต่อไป

บทที่ 3
วิธีการส่งภาพด้วยวิธีไบต์อาเรย์ผ่านช่องสัญญาณความเร็วอัตราบิตต่ำ

ในวิทยานิพนธ์นี้ได้นำเสนอวิธีการส่งไฟล์ภาพด้วยวิธีไบต์อาเรย์ผ่านช่องสัญญาณความเร็วอัตราบิตต่ำ เพื่อนำไปประยุกต์ใช้กับระบบส่งสัญญาณภาพผ่านคลื่นวิทยุ ย่านเฮชเอฟ โดยสามารถอธิบายเทคนิคและอัลกอริทึมที่นำเสนอแบ่งออกเป็นส่วนหลักๆดังนี้

3.1 โครงสร้างและการออกแบบ

วิธีการออกแบบระบบการส่งภาพด้วยไบต์อาเรย์ผ่านช่องสัญญาณความเร็วอัตราบิตต่ำ ถูกสร้างละโปรแกรมด้วยภาษา Java จากไลบรารี Java Media Framework หรือ JMF ช่วยในการเข้าถึงและประมวลผลข้อมูลภาพ



รูปที่ 3.1 โครงสร้างของระบบการส่งภาพด้วยไบต์อาเรย์ผ่านช่องสัญญาณความเร็วอัตราบิตต่ำ

3.2 Capture Screen

Capture Screen เป็นการแปลงไฟล์ภาพเป็นBuffered Image อันดับแรกสุดจำเป็นที่จะต้องหาขนาดของ Screen และสร้างพื้นที่ที่ต้องการจากขนาดของ Screen นั้น และลักษณะของการใช้คำสั่งจำเป็นที่จะต้องใช้ method ที่อยู่ใน Robot Class ก็สามารถที่จะใช้เครื่องมือเพื่อที่จะหาขนาดของ Screen ที่ต้องการได้

```
Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();

Rectangle rectangle = new Rectangle(0, 0,
screenSize.width,screenSize.height);
```

รูปที่ 3.1 Code หาขนาดของ Screen

จากนั้น ก็จะสร้างตัว Instance ขึ้นมาสำหรับการ Capture บนพื้นที่ในตัว Rectangle ตามต้องการ โดนการใช้ method Create Screen Capture เพื่อที่จะรับเอาค่า Rectangle และ return ออกมาในรูปแบบของ Buffer Image

```
Robot robot = new Robot();
BufferedImage image = robot.createScreenCapture(rectangle);
```

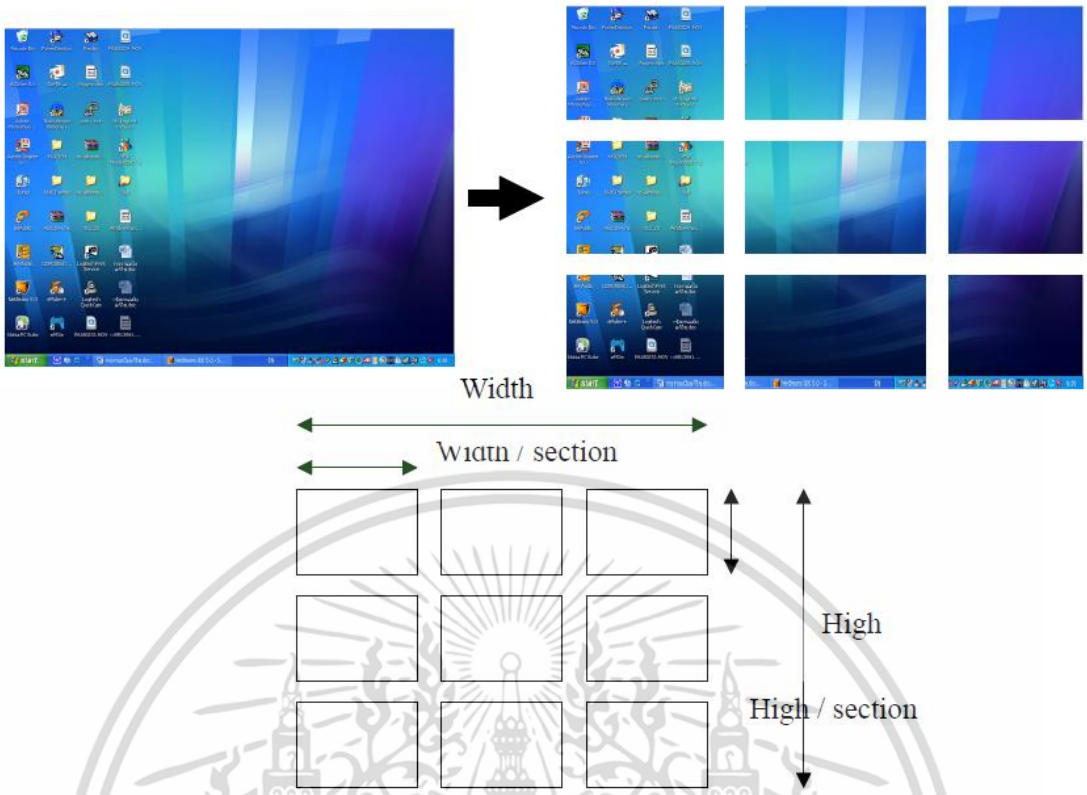
รูปที่ 3.2 Code แปลงเป็น Buffer Image

3.3 Image section

จะเป็นวิธีการทำงานสำหรับแยก Buffered Image ออกเป็นส่วนๆ เพื่อดูความแตกต่างของแต่ละส่วน หลักการเริ่มแรกคือต้องทราบค่าขนาดของความกว้างและความยาวของ Buffered Image ที่แน่นอนก่อน หลังจากนั้นจะเป็นการคำนวณ จำนวน Buffered Image ย่อยด้วยค่าพารามิเตอร์ที่ได้ถูกกำหนดในขั้นตอนสำหรับการสร้างคอนสตรักเตอร์

$$n_{SubBuffered\,Image} = Section^2 \tag{3.1}$$

ยกตัวอย่างขนาดของ Buffered Image คือ 800 x 600 ค่าพารามิเตอร์ Section เท่ากับ 3 จำนวนของ Buffered Image ย่อยคือ 9 ซึ่งลักษณะของภาพที่จะแบ่งออกมาจะเป็นดังรูปที่ 3.3



รูปที่ 3.3 ขนาดของ Buffer Image

Buffered Image ภายหลังจากที่ทำการตัดแบ่งแล้ว จะเรียกส่วนของ Buffered Image ย่อยๆว่า Subimage โดยสามารถที่จะแสดงขั้นตอน และตัวอย่างของการผ่านค่าของ Buffered Image เพื่อสร้าง Subimage ได้ ดังนี้

```
Public void sub_image(BufferedImage imgbuff, float img_quality)
        Picquality = img_quality;
        Cal_subImg(imgbuff);        //เรียกฟังก์ชันสำหรับคำนวณจำนวน
subimage
        new_secImg = new BufferedImage[n_section*n_section];
        int k = 0;
        for(int i=0; i<n_section ; i++){          //รอบ process แต่ละแถว
              for(int j=0; j<n_section; j++){     //รอบสำหรับ process แต่ละคอลัมน์
                    new_secImg[k] =
                    imgbuff.getSubimage(j*sub_w,i*sub_h,sub_,sub_h);
                    k++;
              }
        }
        Comparison_cellImage();     //เรียกฟังก์ชันสำหรับเปรียบเทียบ
subimage
    }
```

รูปที่ 3.4 การผ่านค่าของ Buffered Image เพื่อสร้าง Subimage

ตัวอย่างโค้ดของการกำหนด Header แต่ละ Header ในแต่ละ Subimage ซึ่งจะเป็นส่วนสำคัญมากสำหรับการทำงานและประมวลผลของด้านฝั่งรับข้อมูล

```
private void comparison_cellImage(){
    if(last_secImg==null){                  //กรณีแรกที่ยังไม่มี subimage ก่อนหน้า
        last_secImg = new BufferedImage[n_section*n_section];
        for(int c=0; c<(n_section*n_section); c++){
            Frame_encoderJPEG FrameEncodec = new Frame_encoderJPEG();
            outstream = FrameEncodec.encodeJPEG(new_secImg[c],picquality);
            byte [] sec_stream = outstream.toByteArray();
            new_sec = new byte[sec_stream.length+14];

            ////////////  กำหนด Header ให้แต่ละ Section ////////////
            // 1 byte สำหรับจำนวน section
            new_sec[0] = (byte)n_section ;

            // 4 byte สำหรับจำนวน section ที่ต่างกัน
            byte[] sub_diff_head = new byte[4];
            sub_diff_head = convertSeq.intToDWord(n_section*n_section);
            System.arraycopy(sub_diff_head,0,new_sec,1,sub_diff_head.length);

            // 4 byte สำหรับลำดับที่หรือตำแหน่งในตารางของ subimage ตัวนี้
            byte[] seq_sub_head = new byte[4];
            seq_sub_head = convertSeq.intToDWord(c);
            System.arraycopy(seq_sub_head,0,new_sec,5,seq_sub_head.length);

            // 5 byte section ตำแหน่งสุดท้าย
            new_sec[9] = 0 ;
            new_sec[10] = 0 ;
            new_sec[11] = 0 ;
            new_sec[12] = 0 ;
            new_sec[13] = 0 ;

            ////////////  marker Last sub image ////////////
            if(c==(n_section*n_section)-1){
            //หาก subimage นี้เป็นตำแหน่งสุดท้ายของ Matrix จะมีการกำหนดค่าเป็นดังนี้
                new_sec[9] = -2 ;
                new_sec[10] = 127 ;
                new_sec[11] = 0 ;
                new_sec[12] = -114 ;
                new_sec[13] = 8 ;
            }
            System.arraycopy(sec_stream,0,new_sec,14,new_sec.length-14);
```

รูปที่ 3.5 การกำหนด Header ในแต่ละ Subimage

3.4 การเข้ารหัสและถอดรหัส

สำหรับส่วนนี้เป็นการเข้ารหัสและถอดรหัสข้อมูล เพื่อใช้ในการส่งข้อมูลผ่านระบบเครือข่ายหรือช่องสัญญาณ

3.4.1 การเข้ารหัส

การเข้ารหัสภาพจาก Buffered Image มาเป็น JPEG จุดประสงค์ก็เพื่อทำให้ขนาดของภาพลดลง และสะดวกต่อการแปลงไปเป็นรูปแบบ (Format) อื่นได้ง่ายยิ่งขึ้น

อันดับแรกต้องสร้าง ByteArrayOutputStream ขึ้นมาเพื่อรองรับข้อมูลที่ต้องการจากการเข้ารหัสเป็น JPEG

```
ByteArrayOutputStream out = new ByteArrayOutputStream();
```

รูปที่ 3.6 การสร้าง ByteArrayOutputStream

จากนั้นสร้างออบเจ็กชื่อ encoder ในรูปแบบของ JPEGImageEncoder แล้วส่งพารามิเตอร์ที่เป็นแบบ ByteArrayOutputStream ที่เตรียมไว้ นั้นคือ out

```
JPEGImageEncoder encoder = JPEGCodec.createJPEGEncoder(out);
```

รูปที่ 3.7 การสร้าง JPEGImageEncoder

สร้างออบเจ็ก param ในรูปแบบของ JPEGEncoderParam เพื่อนำภาพจาก BufferedImage ที่เป็นออบเจ็กหลังจากการเรียกใช้คราส Capturescreen ที่ชื่อ image

```
JPEGEncodeParam param = encoder.getDefaultJPEGEncodeParam(image);
```

รูปที่ 3.8 การสร้าง JPEGEncoderParam เพื่อนำภาพจาก BufferedImage

ต่อไปจะเป็นการกำหนดค่าพารามิเตอร์ เพื่อกำหนดความละเอียดสำหรับการเข้ารหัส JPEG คือค่าของ picquality ที่เป็นไปได้อยู่ในช่วง 0.0 – 1.0 โดยค่ายิ่งเข้าใกล้ 1 จะมีความละเอียดของภาพหลังจากการเข้ารหัสเป็น JPEG มากที่สุด หลักจากนั้นจะเป็นการสั่งให้ทำการเข้ารหัส

```
JPEGEncodeParam param.setQuality(picquality, false);
        encoder.setJPEGEncodeParam(param);
        encoder.encode(image);
```

รูปที่ 3.9 การกำหนดความละเอียดสำหรับการเข้ารหัส JPEG

เพื่อความสะดวกและสามารถเรียกใช้งานได้ง่ายจึงสร้างเป็นคราสชื่อ Frame_encodeJPEG โดยนำโค้ดที่กล่าวมาแล้วข้างต้นมาปรับปรุง โดยรับพารามิเตอร์ 2 ค่า คือ BufferedImage และตัวแปลชนิดเก็บค่าตัวเลขทศนิยม float เพื่อกำหนดความละเอียดของการเข้ารหัสจากนั้นจะคืนค่าออกมาเป็น ByteArrayOutputStream

```
private  ByteArrayOutputStream out = new ByteArrayOutputStream();
private  JPEGImageEncoder encoder ;
private float picquality = 0.5f;
private  JPEGEncodeParam param;

public  ByteArrayOutputStream encodeJPEG(BufferedImage inimage,
                                         float picquality){
          try{
                  encoder = JPEGCodec.createJPEGEncoder(out);
                  param = encoder.getDefaultJPEGEncodeParam(inimage);

                  param.setQuality(picquality, false);
                  encoder.setJPEGEncodeParam(param);
                  encoder.encode(inimage);
                  return out;
          }
          catch (Exception e){
                  System.out.println(e.getMessage());
       }
          return out ;
    }
```

รูปที่ 3.10 โค้ดที่ทำการปรับปรุงโดยรับค่า BufferedImage และ float

3.4.2 การถอดรหัส

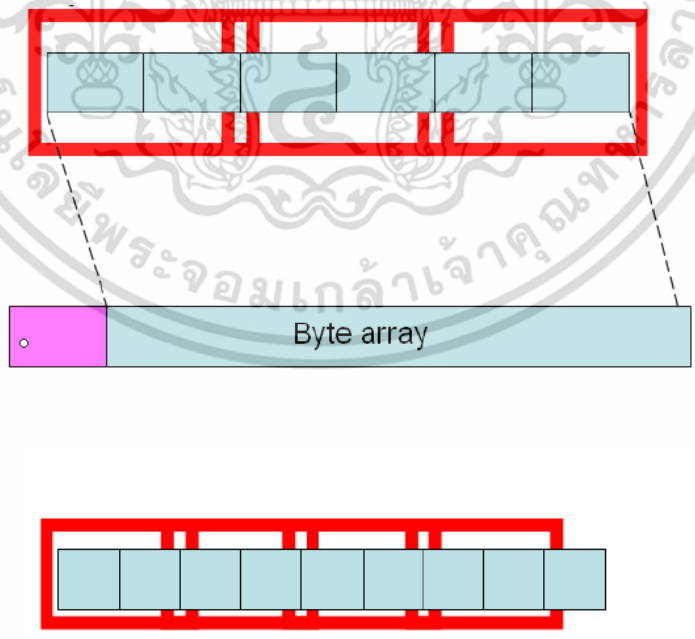
จะเป็นการถอดรหัสจากในรูปแบบของ JPEG (ByteArrayOutputStream) ให้มาอยู่ที่รูปแบบของ BufferedImage ที่ง่ายสำหรับการนำไปประมวลผลต่อไป เริ่มแรกนั้นจะต้องเตรียม ByteArrayInputStream เข้ามารองรับค่าของ ByteArrayOutputStream ที่เป็นของ JPEG และทำการเปลี่ยนรูปแบบเป็นไบต์อาเรย์ จากนั้นค่อยใช้ออบเจ็ก decoder ชนิด JPEGImageDecoder เรียก method createJPEGDecoder ที่จะต้องผ่านค่าของพารามิเตอร์ของ ByteArrayInputStream ที่ได้ทำการแปลงค่ารอไว้แล้วก่อนหน้า จากนั้นจึงแปลงออบเจ็ก decoder ให้เป็นรูปแบบของ BufferedImage ที่สามารถนำไปใช้ต่อไปได้ เพื่อความสะดวกและเรียกใช้งานได้ง่ายจึงสร้างเป็นคราสชื่อ Frame_decoderJPEG โดยทำหลักการที่ได้กล่าวมาแล้วข้างต้นมาปรับปรุงดังนี้

```
private ByteArrayInputStream in = null;       //เตรียม ByteArrayInputStream
private BufferedImage recoverbuff = null;     //เตรียม BufferedImage

    public BufferedImage decodeJPEG(ByteArrayOutputStream out){

            try{

             in = new ByteArrayInputStream(out.toByteArray());
             JPEGImageDecoder decoder = JPEGCodec.createJPEGDecoder(in);
             recoverbuff = decoder.decodeAsBufferedImage();

             return recoverbuff;

            }
            catch (Exception e){
             System.out.println(e.getMessage());
        }
            return recoverbuff;
    }
```

รูปที่ 3.11 โค้ดการถอดรหัส JPEG

3.5 กระบวนไบต์อาเรย์ (Byte Array Processing)

สำหรับส่วนนี้จะเป็นการจัดการเกี่ยวกับข้อมูล เพื่อที่จะนำไปใช้ในการจัดเรียงไบต์ โดยจำเป็นต้องเพิ่มในส่วนของเฮดเดอร์ (Header) ให้กับไบต์อาเรย์ เพื่อให้สามารถส่งข้อมูลไปยังระบบหรือช่องทางการส่งสัญญาณได้



รูปที่ 3.12 กระบวนการไบต์อาเรย์

3.6 RevThread

สำหรับคราสนี้เป็น การทำงาที่เกี่ยวข้องกับการดักจับข้อมูลจากเดต้าแกรมซอกเกต โดยหลักการจะมีรูปแบบการทำงานที่เป็น Thread และจะรับค่าไบต์อาเรย์ที่มีขนาด 1110 หลังจากการทำงาน โดยรับค่าไบต์อาเรย์มาแล้ว ก็จะทำการแจ้งอีเว้น(Event) เพื่อนำเอาไบต์อาเรย์ที่ได้มาจำนวน 1110 นี้ไปทำการประมวลผลต่อไป

```
public void run(){

        while(sruning){
            try{
                 DatagramPacket packet = new DatagramPacket(new byte
                                         [bytesToReceive], bytesToReceive);
                mySocket.receive(packet);           // wait for a packet
                                    address =  packet.getAddress();     // get
address

                try{
                    byte []data = packet.getData();
                    int len = packet.getLength();
                    int offset = packet.getOffset();
                    listener.onRevData(data, len, offset);

                                    // แจ้ง event เมื่อ ได้รับ ได้รับ ไบต์อาเรย์ ขนาด 1110
            }
            catch(Exception e){
                e.printStackTrace();
                System.out.println("Interface ERROR " +e.getMessage());
            }

        }
        catch (IOException e){
          System.out.println("UDPListener has a problem or Sys error: " + e);
          sruning = false;

        }
    }
}
```

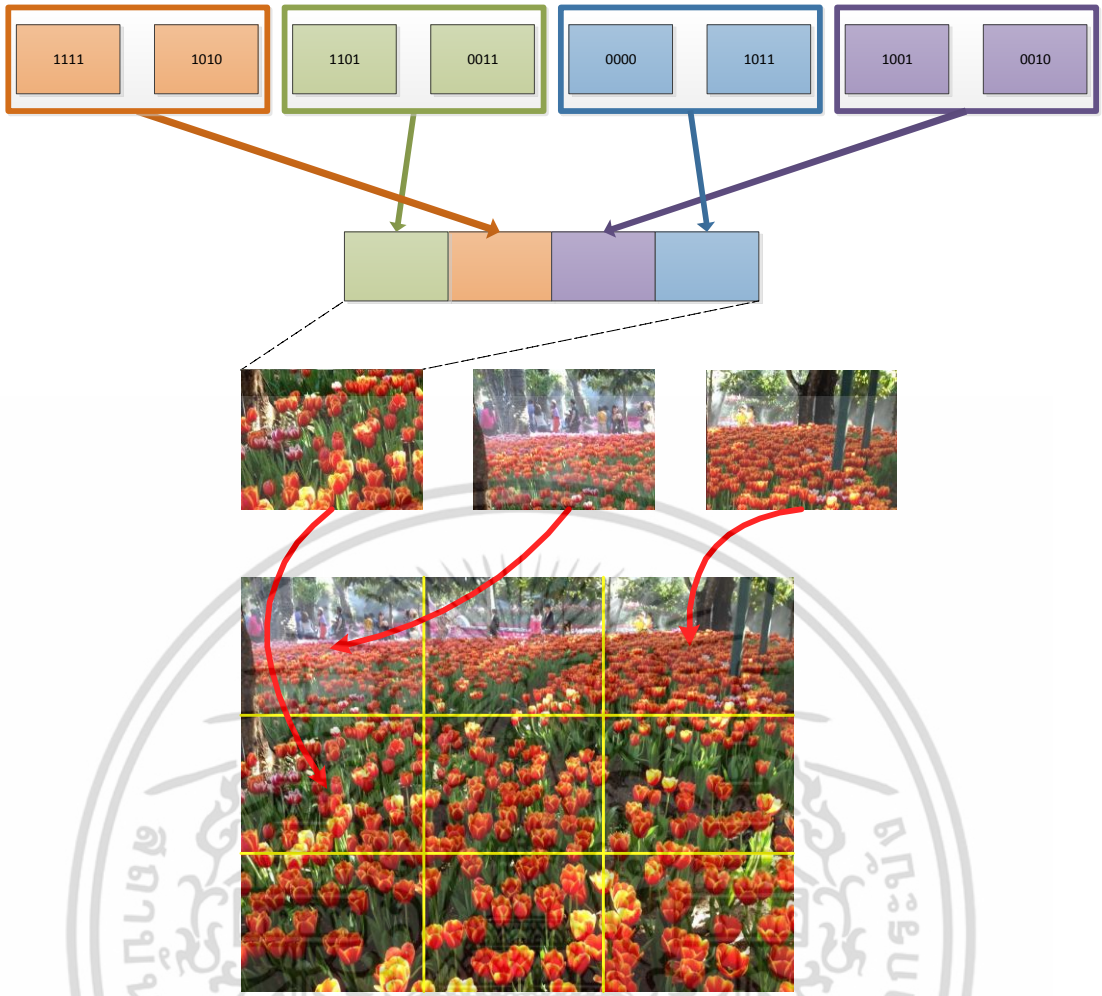
รูปที่ 3.13 การแจ้งอีเว้น

3.7 ManageOderSequence

เมื่อข้อมูลที่รับเข้ามาครั้งละ 1110 ไบต์ จากด้านฝั่งส่งข้อมูล โดยที่ 1110 ไบต์ ในแต่ละชุดที่ส่งมานั้นมีการกำหนดค่าของเฮดเดอร์ไว้เพียง 10 ไบต์แรก ส่วนที่เหลือจะเป็นข้อมูลจริง ซึ่งเฟรมภาพหนึ่งภาพ จะถูกแบ่งออกมาเป็นไบต์อาเรย์ที่มีขนาดใหญ่ เป็นสาเหตุให้ต้องมีการแบ่งไบต์อาเรย์ 1100 ไบต์ และเพิ่ม 10 ไบต์ในทุกๆ 1100 ไบต์ เพื่อให้สามารถระบุได้ว่า 1100 ไบต์ของแต่ละชุดนั้นเป็นลำดับที่เท่าไรในจำนวนทั้งหมดของเฟรมภาพ จึงเป็นไปได้ว่าลำดับการมาถึงของข้อมูลจากด้านฝั่งรับข้อมูลอาจจะได้รับข้อมูลไม่เป็นไปตามที่กำหนดเหมือนในลำดับการส่งของด้านฝั่งส่งข้อมูล จึงเป็นหน้าที่ของคราส ManageOderSequence ที่จะต้องอ่านค่า 10 ไบต์แรกของทุกๆชุดข้อมูลที่เข้ามาเพื่อดูว่าชุดข้อมูลเป็นลำดับที่เท่าไรในเฟรมภาพ จากนั้นจะทำการสำเนาเก็บไว้ในไบต์อาเรย์ชุดใหม่จนกว่าจะพบชุดข้อมูลสุดท้ายของเฟรมภาพ

```
private void performArray(byte [] coming){

        ////////// สร้างไบต์เพื่อรองรับ header ที่อยู่ภายในของ subimage //////////
        byte [] num_dif = new byte[4];      //ไบต์ที่ 1-4 จำนวนที่ต่างกันใน matrix
        byte [] num_seq = new byte[4];      //ไบต์ที่ 5-8 ลำดับที่ของ subimage ใน matrix
        byte [] last_sub = new byte[5];     //ไบต์ที่ 9-13 สำหรับ subimage ตัวสุดท้ายที่เข้ามา
        byte [] cutHeader = new byte[coming.length-14];  //ไบต์ที่เป็น subimage

        //ทำสำเนาลงไบต์ในแต่ละชุดที่เตรียมไว้
        System.arraycopy(coming,1,num_dif,0,num_dif.length);
        System.arraycopy(coming,5,num_seq,0,num_seq.length);
        System.arraycopy(coming,9,last_sub,0,last_sub.length);
        System.arraycopy(coming,14,cutHeader,0,cutHeader.length);

        //ขั้นตอนของการเข้ารหัส JPEG
        out = new ByteArrayOutputStream();
        out.write(cutHeader,0,cutHeader.length);

        BufferedImage imgaccep = null;
        Frame_decoderJPEG Framedecodec = new Frame_decoderJPEG();
        imgaccep = Framedecodec.decodeJPEG(out);

        //ตรวจสอบขนาดของ subimage พร้อมกับหาตำแหน่งที่วาดลงใน BufferImage
        int w = imgaccep.getWidth(), h = imgaccep.getHeight();
        int Cx =cal_origin((int)coming[0],man_seque.byteArrayToInt(num_seq,0)).x;
        int Cy =cal_origin((int)coming[0],man_seque.byteArrayToInt(num_seq,0)).y;


        //หาก subimage ที่เข้ามาเป็นของ section หรือ เฟรมชุดใหม่ ก็ต้องสร้าง BufferImage ใหม่
        byte b = coming[0];
        if(flag==0 || b!=a){
            total = new BufferedImage((int)coming[0]*w, (int)coming[0]*h,
                    BufferedImage.TYPE_INT_RGB);
            a = b;
            flag = 1;
        }

        //วาด subimage ที่ได้ลงใน BufferedImage ด้วยตำแหน่งที่ได้กำหนดไว้ใน matrix
        Graphics2D g = total.createGraphics();
        g.drawImage(imgaccep, Cx*w, Cy*h, null);
```

รูปที่ 3.14 โค้ดการเรียงลำดับเฟรมภาพ



รูปที่ 3.15 กระบวนการเรียงลำดับเฟรมภาพ

บทที่ 4
วิธีการและผลการทดลอง

วิทยานิพนธ์ฉบับนี้นำเสนอวิธีการการส่งไฟล์ภาพผ่านช่องทางการสื่อสารความเร็วอัตราบิตต่ำ โดยใช้วิธีการไบต์อาเรย์เพื่อควบคุมปริมาณของข้อมูลภาพ และทดลองการปรับเปลี่ยนขนาดของไฟล์ภาพ, อัตราการรับส่งข้อมูล, ขนาดของไบต์อาเรย์ เพื่อเปรียบเทียบระยะเวลาตั้งแต่เริ่มทำการส่งข้อมูลภาพจนถึงการแปลงไฟล์ภาพให้สามารถแสดงผล รวมถึงการหาค่าที่เหมาะสมกับความสามารถของช่องทางการสื่อสารความเร็วอัตราบิตต่ำ ดังนี้

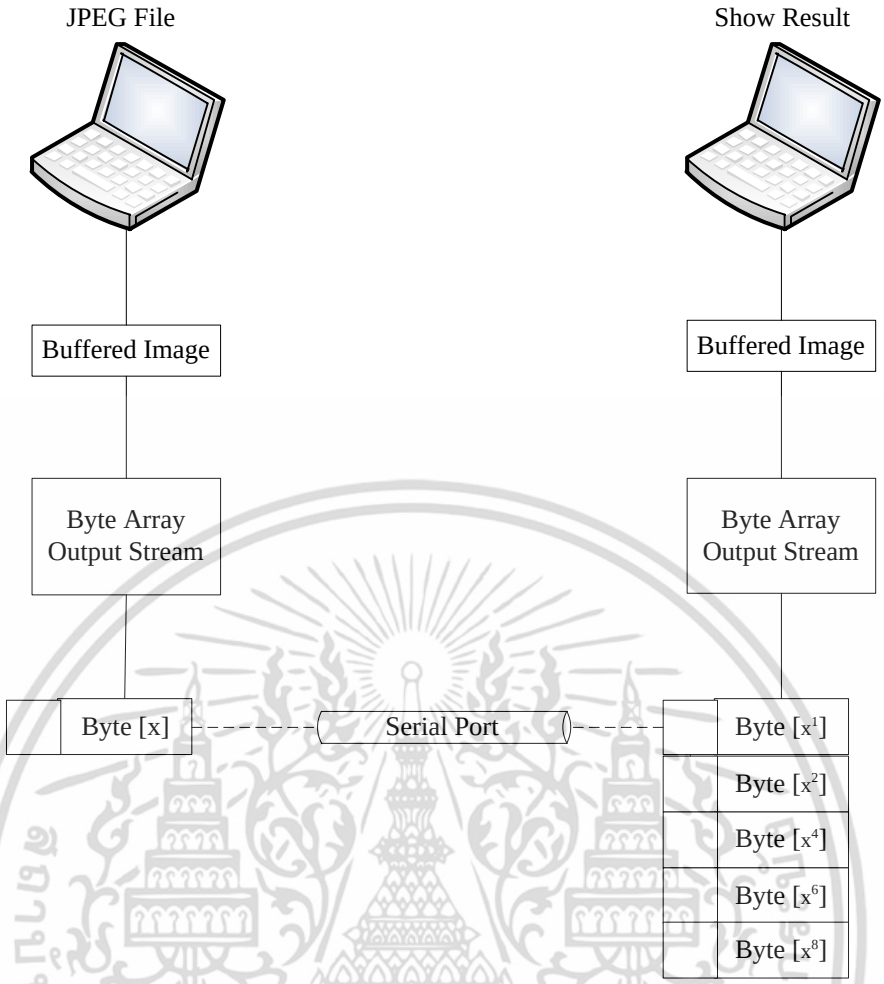
4.1 อุปกรณ์และโมเดลในการทดลอง

การเลือกอุปกรณ์และโมเดลในการทดลอง เป็นสิ่งสำคัญที่มีผลต่องานวิจัยนี้ ยิ่งเกี่ยวข้องกับสื่อมัลติมีเดียด้วยแล้ว เครื่องมือในการสร้างและออกแบบระบบมักเกี่ยวข้องกับการพัฒนาซอฟต์แวร์เพื่อใช้ในการทดสอบ โดยเฉพาะอย่างยิ่งภาษาที่ใช้พัฒนาแอพพลิเคชัน เครื่องมือที่ช่วยอำนวยความสะดวกในการพัฒนาแอพพลิเคชัน IDE (Integration Development Environment) และไลบรารี อันเป็นส่วนประกอบที่ทำให้การพัฒนาแอพพลิเคชันเฉพาะด้าน มีความสมบูรณ์มากยิ่งขึ้น ดังนั้น ในวิทยานิพนธ์นี้เลือกใช้ภาษา Java เป็นภาษาในการพัฒนาแอพพลิเคชัน อาศัย NetBeans IDE ช่วยอำนวยความสะดวก และ JMF (Java Media Framework) ซึ่งเป็นไลบรารี หรือ API (Application Programming Interface) สำหรับงานด้านมัลติมีเดียโดยเฉพาะ เครื่องมือและซอฟต์แวร์ทั้งหมดเป็นฟรีแวร์(Freeware) และโอเพนซอร์ส (Open Source) เป็นทางเลือกหนึ่งที่ช่วยลดงบประมาณในการวิจัยและคิดค้นอัลกอริทึม

การวิจัยวิธีการส่งภาพด้วยวิธีไบต์อาเรย์ผ่านช่องสัญญาณความเร็วอัตราบิตต่ำจะทดสอบวิธีการส่งภาพผ่านสายสัญญาณแบบอนุกรม โดยผ่านพอร์ตอนุกรมอาร์เอส 232 (Serial Port RS232) และใช้คอนเนคเตอร์แบบดีบี 9 (DB9 Connector) ซึ่งถือเป็นช่องสัญญาณความเร็วอัตราบิตต่ำรูปแบบหนึ่ง โดยการใช้โมลเดลในการทดลองแบ่งเป็นสามโมเดลดังต่อไปนี้

1. ทดลองปรับเปลี่ยนขนาดของไฟล์ภาพ
2. ทดลองปรับเปลี่ยนอัตราการรับส่งข้อมูล
3. ทดลองปรับเปลี่ยนขนาดของไบต์อาเรย์

โดยจะทำการกำหนดอัตราการรับส่งข้อมูลและขนาดของไบต์อาเรย์ ทำการทดลองส่งไฟล์ภาพขนาดต่างๆ เมื่อทางด้านฝั่งรับข้อมูลสามารถแสดงผลภาพได้จะทำการแสดงค่าระยะเวลาของกระบวนการรับข้อมูล จากนั้นทำการปรับอัตราการส่งข้อมูลให้มากขึ้น ต่อด้วยการปรับขนาดของไบต์อาเรย์ จนได้ผลการทดลองที่ครบถ้วน



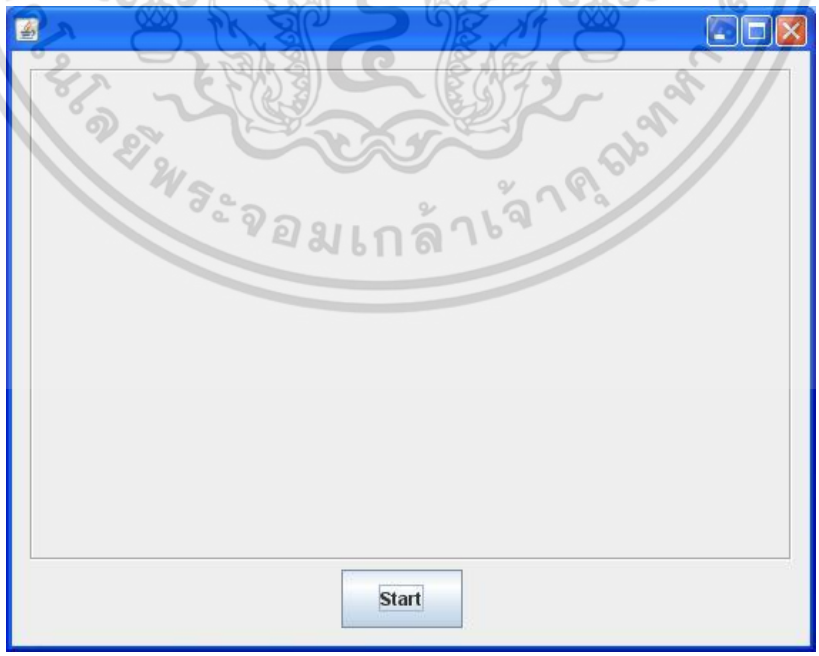
รูปที่ 4.1 กระบวนการส่งภาพด้วยวิธีไบต์อาเรย์ผ่านช่องสัญญาณความเร็วอัตราบิตต่ำ

จากตารางที่ 4.1 คือค่าความละเอียดของภาพ อัตราการรับส่งข้อมูล และขนาดของไบต์อาเรย์ที่ใช้ในการทดลอง การเลือกใช้ค่าตัวแปรตามในตารางนี้ จะเห็นได้ว่ามีการเพิ่มขึ้นของขนาดตัวแปรแบบเท่าตัว เพื่อให้สามารถให้ผลการทดลองได้ชัดเจนมากขึ้น

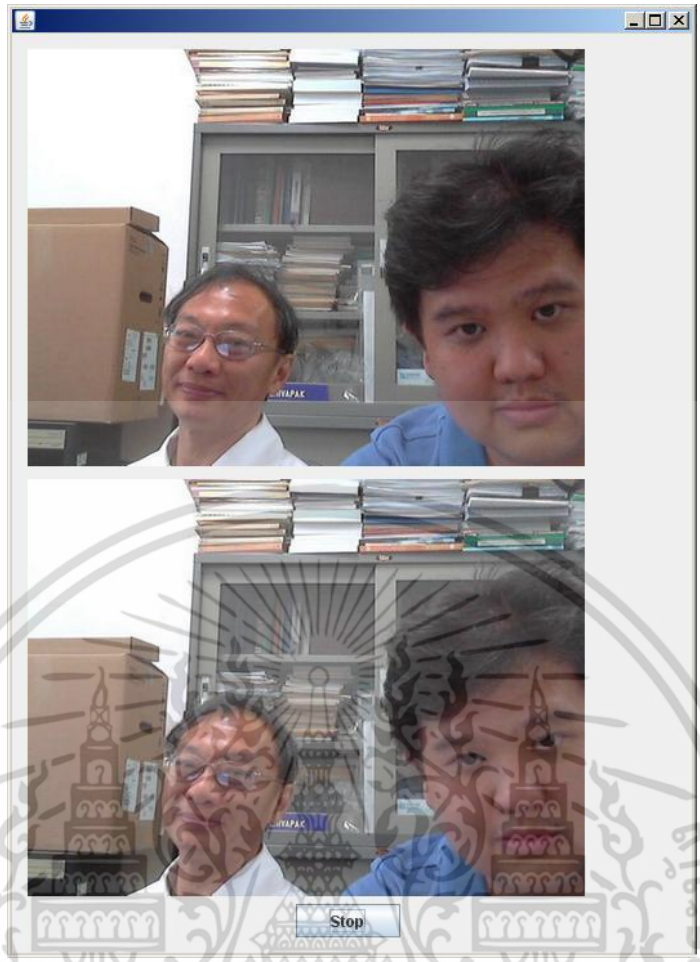
ตารางที่ 4.1 การทดลองเพิ่มค่าความละเอียดของภาพ อัตราการรับส่งข้อมูล และขนาดไบต์อาเรย์

Resolution	Baud Rate	Byte Array (Byte)
320*240	9600	524288
640*480	19200	1048576
800*600	38400	2097152

รูปที่ 4.2 แอพพลิเคชันฝั่งส่งข้อมูลที่พัฒนาขึ้นสำหรับการวิจัยและทดลอง



รูปที่ 4.3 แอพพลิเคชันฝั่งรับข้อมูลที่พัฒนาขึ้นสำหรับการวิจัยและทดลอง



รูปที่ 4.4 การแสดงผลของแอพพลิเคชันฝั่งส่งข้อมูล

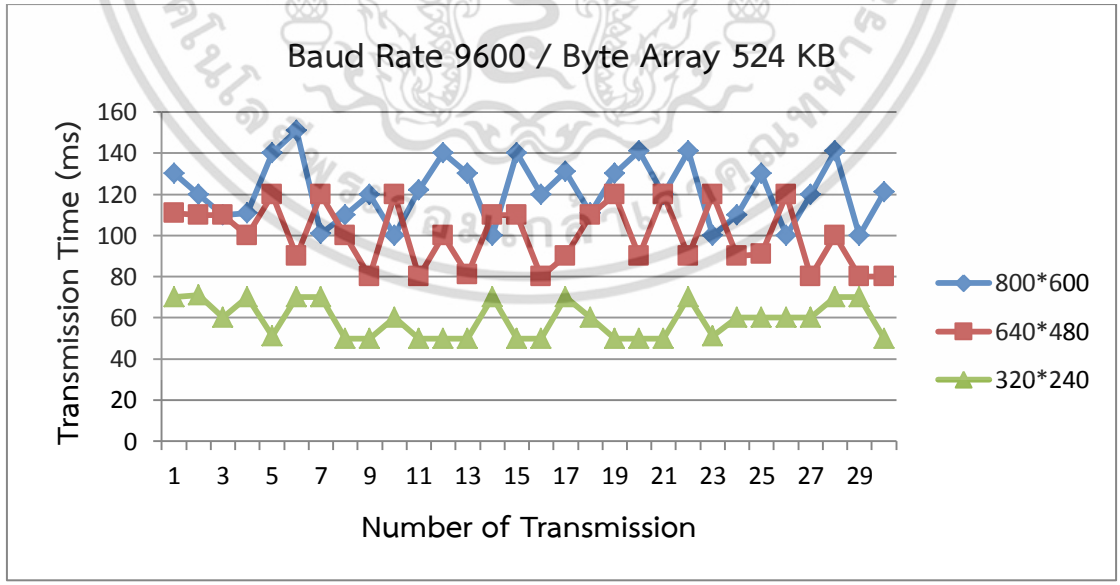
รูปที่ 4.5 การแสดงผลของแอพพลิเคชันฝั่งรับข้อมูล

```
Size:19496 Indx: 398 --> 0
Size:19496 Indx: 399 --> 3
Size:19496 Indx: 400 --> 1
Size:19496 Indx: 401 --> 1
Size:19496 Indx: 402 --> 1
Size:19496 Indx: 403 --> 1
Size:19496 Indx: 404 --> 1
Size:19496 Indx: 405 --> 1
Size:19496 Indx: 406 --> 1
Size:19496 Indx: 407 --> 1
Size:19496 Indx: 408 --> 1
Size:19496 Indx: 409 --> 0
Size:19496 Indx: 410 --> 0
Size:19496 Indx: 411 --> 0
Size:19496 Indx: 412 --> 0
Size:19496 Indx: 413 --> 0
Size:19496 Indx: 414 --> 0
```

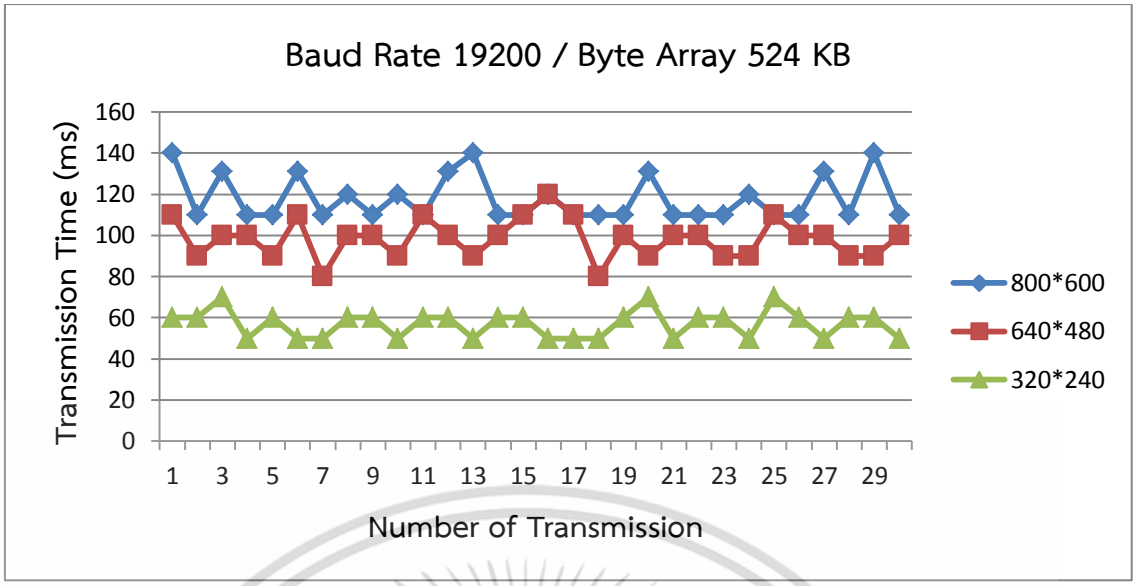
รูปที่ 4.6 ตัวอย่าง Log ไฟล์ที่เกิดขึ้นจริงจากการทดลอง

4.2 ผลทดลอง

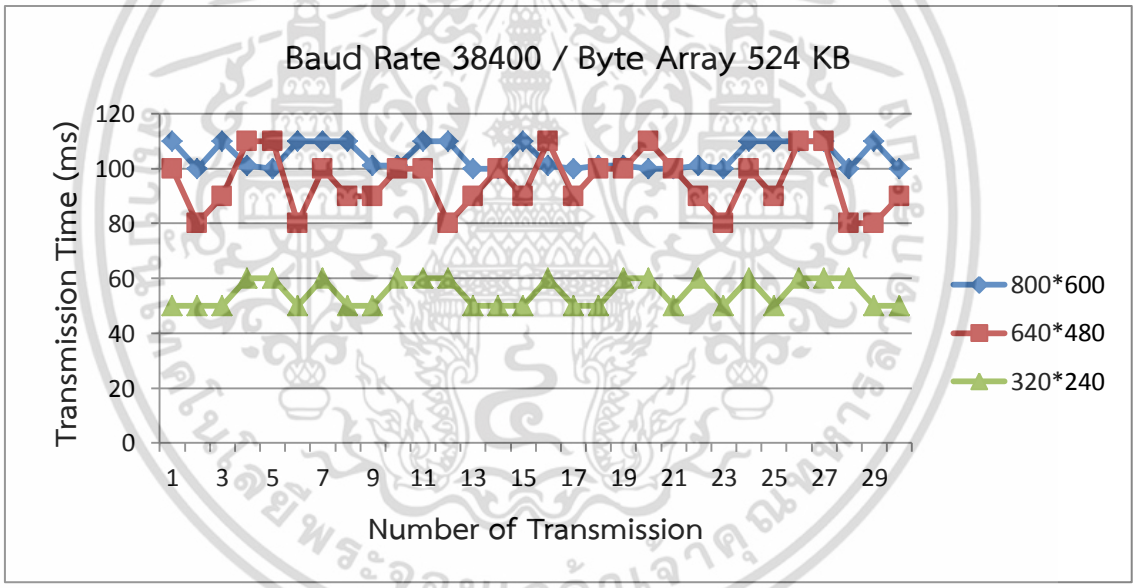
การทดลองนี้จะส่งข้อมูลไฟล์ภาพทางด้านฝั่งส่งข้อมูลไปยังฝั่งรับข้อมูลโดยผ่านช่องสัญญาณพอร์ตอนุกรมที่ต้องมีการเข้ารหัสและการจำกัดปริมาณของข้อมูลที่ใช้ในการส่งโดยใช้วิธีไบต์อาเรย์ ซึ่งจะมีการปรับเปลี่ยนขนาดของไฟล์ภาพ อัตราการรับส่งข้อมูล และขนาดของไบต์อาเรย์ในแต่ละรอบการทดลองให้มากขึ้น ส่วนทางด้านฝั่งรับข้อมูลจะทำการจับเวลาตั้งแต่เริ่มรับข้อมูลเข้ามาจนสามารถรับข้อมูลมาครบถ้วนและทำการถอดรหัส จนถึงการแสดงผลออกมาเป็นภาพ ถือว่าเป็นอันจบกระบวนการทดลอง



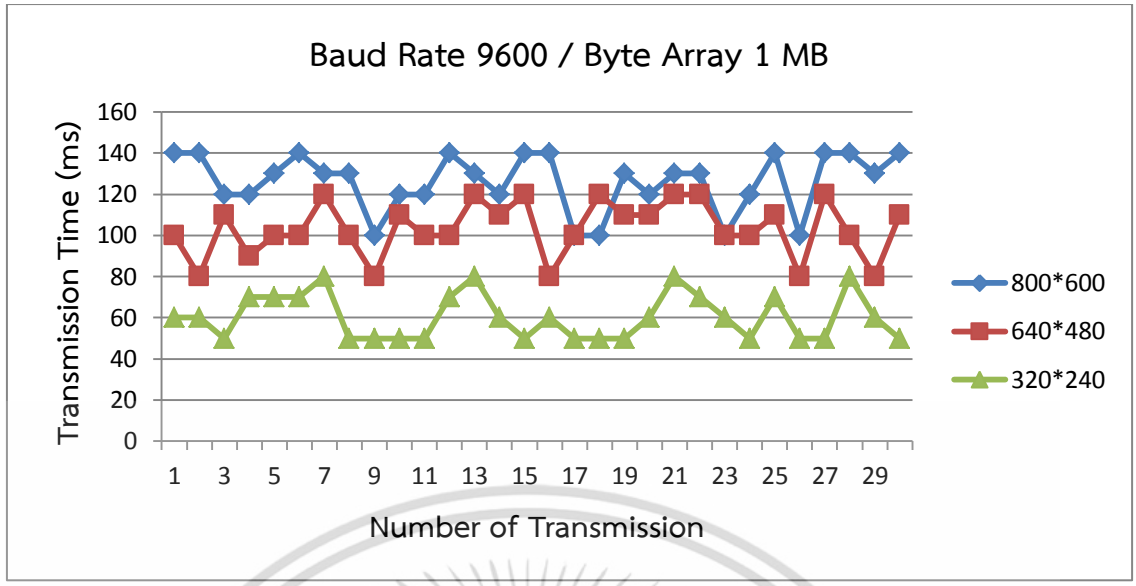
รูปที่ 4.7 ผลการทดลองของไบต์อาเรย์ขนาด 524 KB ที่อัตรารับส่งข้อมูล 9600 bps



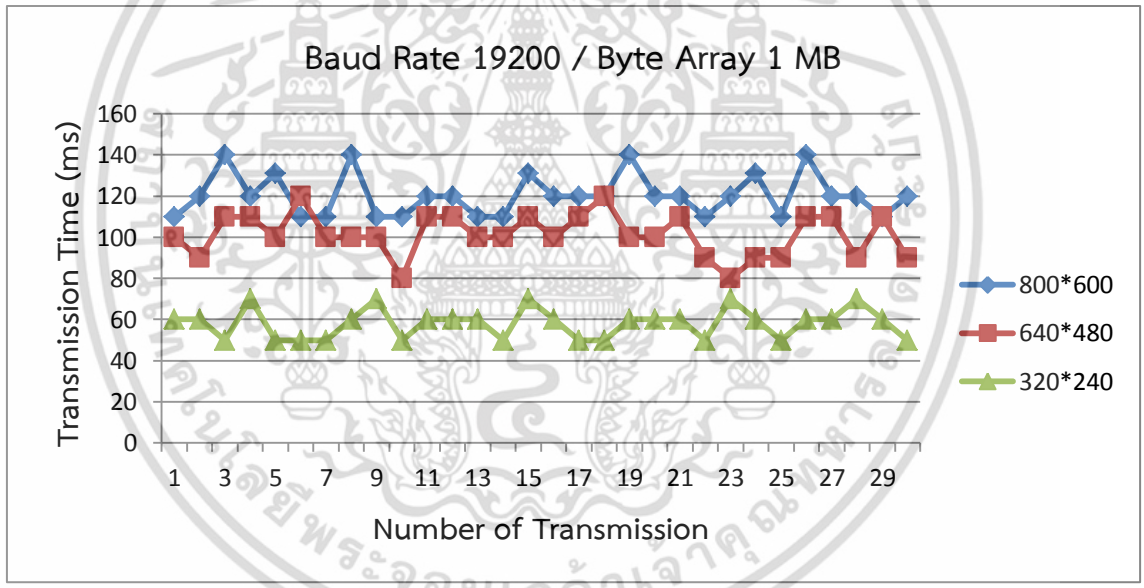
รูปที่ 4.8 ผลการทดลองของไบต์อาเรย์ขนาด 524 KB ที่อัตรารับส่งข้อมูล 19200 bps



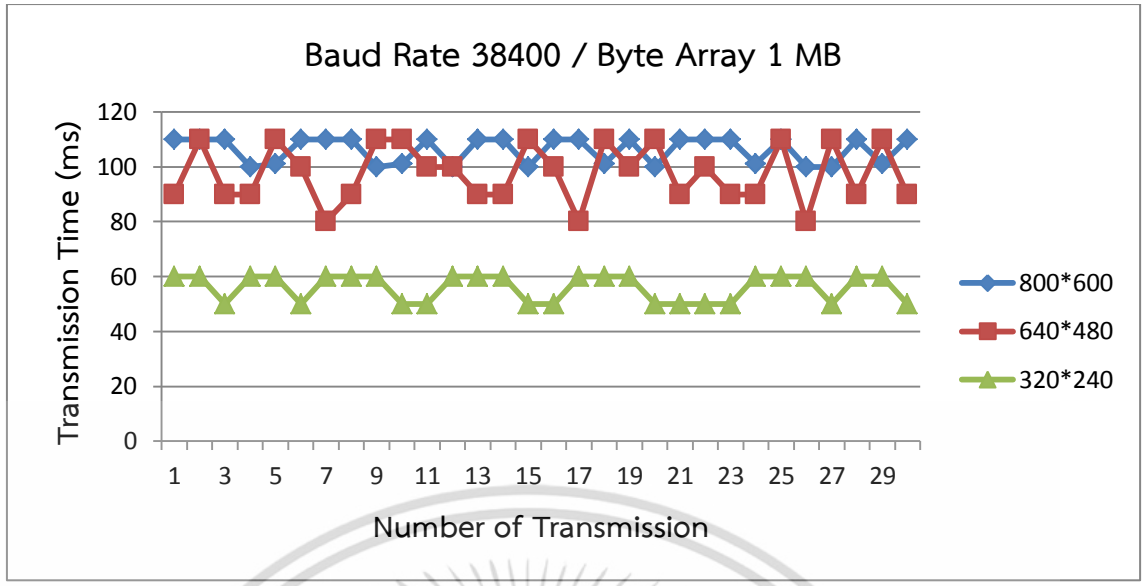
รูปที่ 4.9 ผลการทดลองของไบต์อาเรย์ขนาด 524 KB ที่อัตรารับส่งข้อมูล 38400 bps



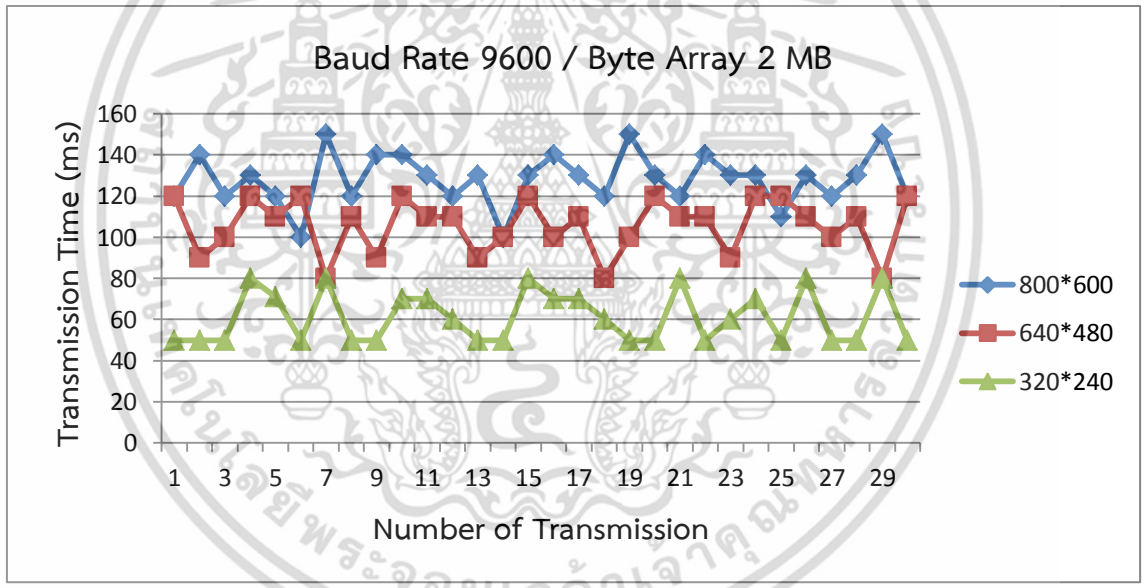
รูปที่ 4.10 ผลการทดลองของไบต์อาเรย์ขนาด 1 MB ที่อัตรารับส่งข้อมูล 9600 bps



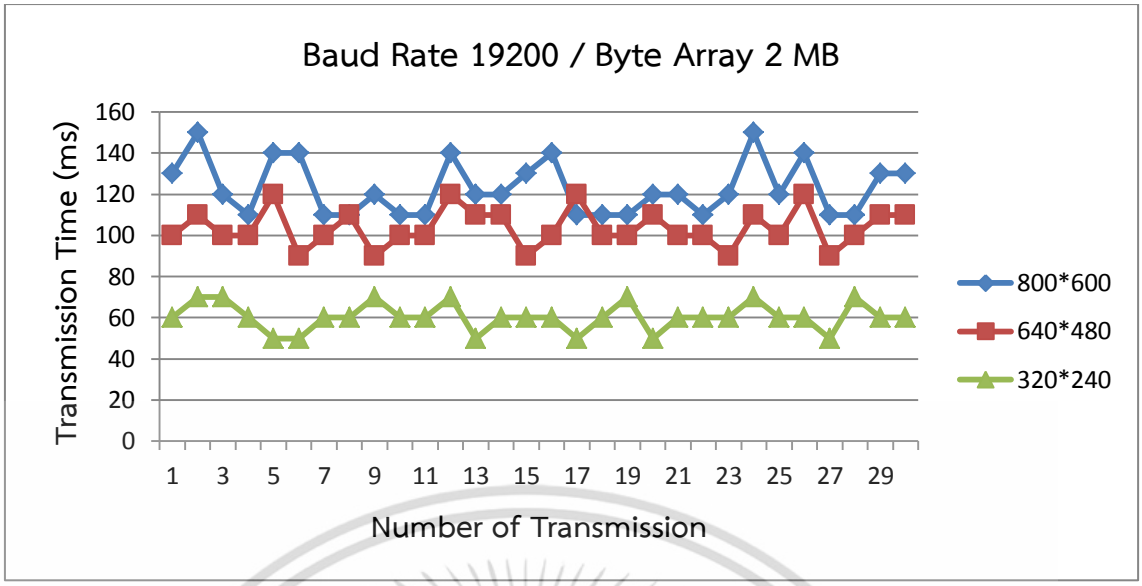
รูปที่ 4.11 ผลการทดลองของไบต์อาเรย์ขนาด 1 MB ที่อัตรารับส่งข้อมูล 19200 bps



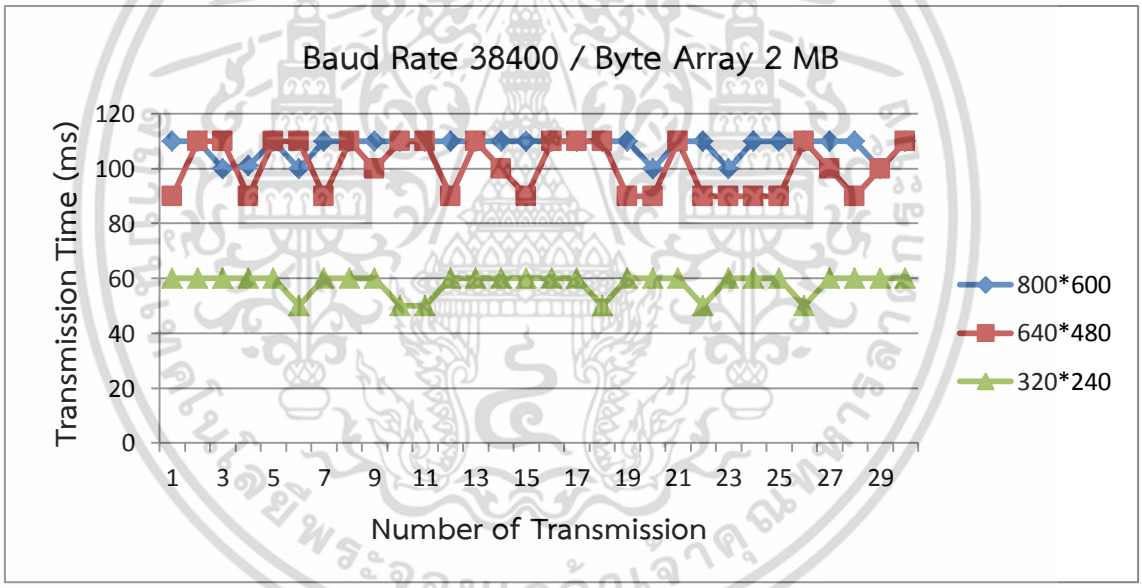
รูปที่ 4.12 ผลการทดลองของไบต์อาเรย์ขนาด 1 MB ที่อัตรารับส่งข้อมูล 38400 bps



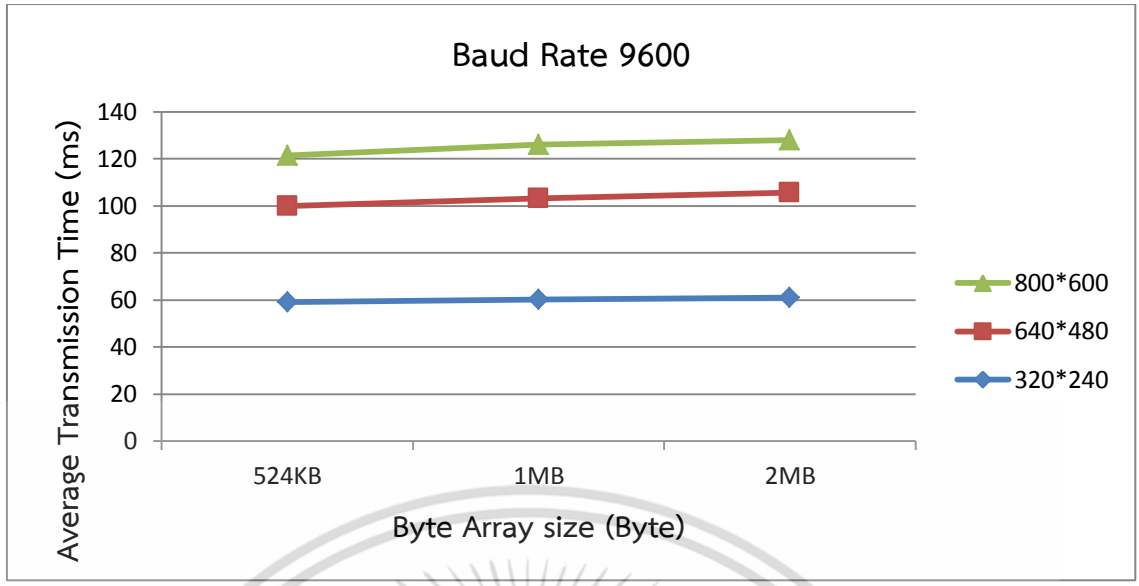
รูปที่ 4.13 ผลการทดลองของไบต์อาเรย์ขนาด 2 MB ที่อัตรารับส่งข้อมูล 9600 bps



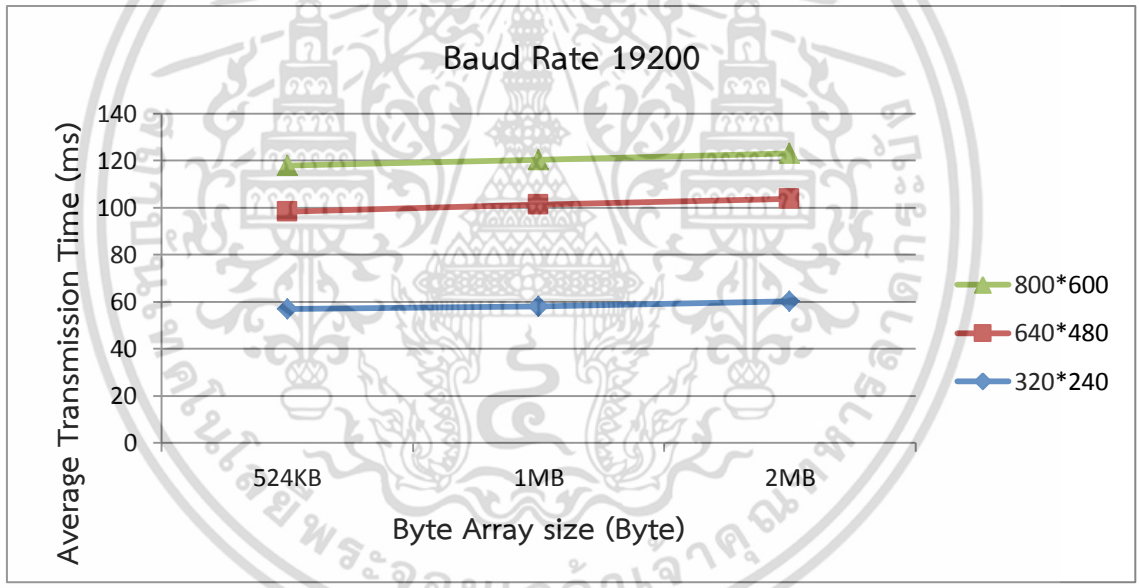
รูปที่ 4.14 ผลการทดลองของไบต์อาเรย์ขนาด 2 MB ที่อัตรารับส่งข้อมูล 19200 bps



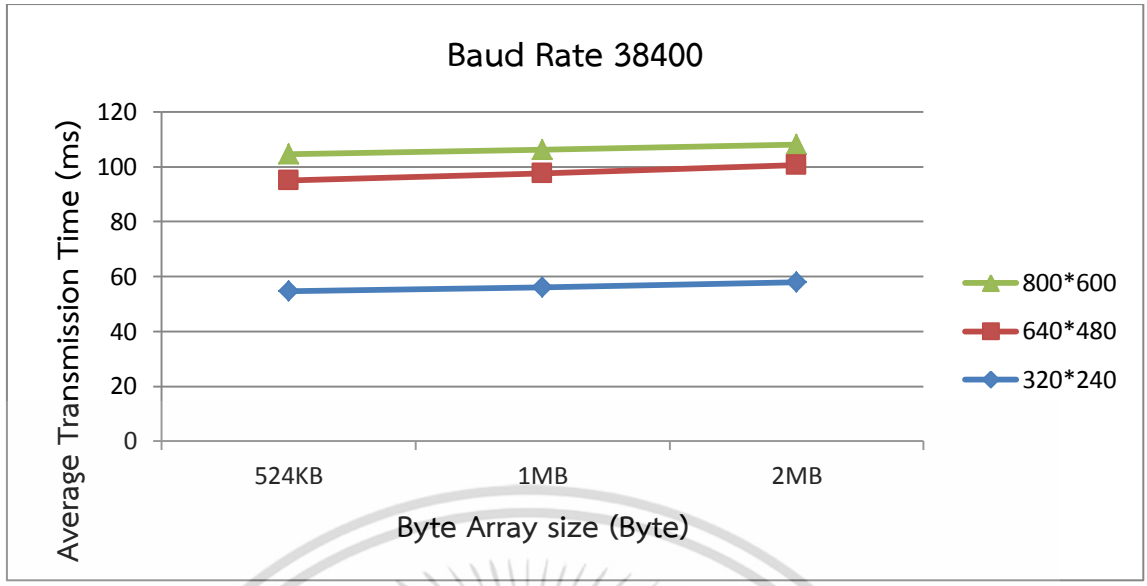
รูปที่ 4.15 ผลการทดลองของไบต์อาเรย์ขนาด 2 MB ที่อัตรารับส่งข้อมูล 38400 bps



รูปที่ 4.16 ผลค่าเฉลี่ยของเวลาในการรับข้อมูลในแต่ละไบต์อาเรย์ ที่อัตรารับส่งข้อมูล 9600 bps



รูปที่ 4.17 ผลค่าเฉลี่ยของเวลาในการรับข้อมูลในแต่ละไบต์อาเรย์ ที่อัตรารับส่งข้อมูล 19200 bps



รูปที่ 4.18 ผลค่าเฉลี่ยของเวลาในการรับข้อมูลในแต่ละไบต์อาเรย์ ที่อัตรารับส่งข้อมูล 38400 bps

บทที่ 5
สรุปผลการวิจัย และข้อเสนอแนะ

จากการทดลองวิธีการส่งภาพด้วยวิธีไบต์อาเรย์ผ่านช่องสัญญาณความเร็วอัตราบิตต่ำ ด้วยขนาดของไบต์อาเรย์ที่ต่างกัน จะเห็นได้ว่าเมื่อทำการส่งข้อมูลบ่อยครั้งโดยที่ขนาดของไบต์อาเรย์ยังเท่าเดิม ระยะเวลาที่ใช้ในการส่งข้อมูลจะไม่เปลี่ยนแปลงมากนัก แต่ถ้ามีการปรับเปลี่ยนขนาดของไบต์อาเรย์มากขึ้น ค่าเฉลี่ยของระยะเวลาการส่งข้อมูลจะมากขึ้นตามไปด้วย

วิธีการที่นำเสนอในวิทยานิพนธ์เป็นเทคนิคหนึ่งเท่านั้นที่ใช้ในการส่งข้อมูลไฟล์ภาพผ่านระบบช่องสัญญาณความเร็วอัตราบิตต่ำ แต่ก็อาจจะมีเทคนิคและวิธีการอื่นอีกที่น่าสนใจ ลามารถที่จะทำให้ส่งสัญญาณภาพผ่านช่องสัญญาณความเร็วอัตราบิตต่ำได้ เช่น วิธีการบีบอัดข้อมูลแบบใหม่ๆ ที่ทำให้ปริมาณข้อมูลลดลง หรือ วิธีการปรับแต่งข้อมูลภาพให้สามารถรับส่งข้อมูลได้เร็วและไม่สิ้นเปลืองอัตราการรับส่งข้อมูล

บรรณานุกรม

[1] James E. Fowler, Kenneth C. Adkins, Steven B. Bibyk, and Stanley C. Ahalt, "**Real time video compression using differential vector quantization**," in IEEE Trans.on Circuits and Systems for Video Technology, 1995, Pp. 14–24.

[2] Jain E. G. Richardson, **Video Codec Design: Developing Image and Video Compression Systems.** 1st ED. 2012.

[3] Andrew S. Tanenbaum. **Computer Networks**. 5th ED. October 2010. pp.432-434

[4] Ohm, Jens-Rainer. **Multimedia Communication Technology:Representation, Transmission and Identification of Multimedia Signals**. January 2004. Pp.135

[5] R.E. Parker, Jr. and M. Tummala, "**Modeling of H.263 encoded low bitrate video traffic for tactical video conferencing applications**," Conference Record of the Thirty-Second Asilomar Conference, Signals, Systems & Computers, vol.2, Pp.1626-1630, 1998.

[6] K. Tan, R. Ribier and S. Liou, "**Content-sensitive video streaming over low bit rate and lossy wireless network**," ACM 2011. 9th international conference on Multimedia, Pp.512–515, September 2001.

[7] L. Mengual, J. Bobadilla, R. Caballero, G. Hernández, "**Design and testing of two secure video conferencing applications based on JMF (Java Media Framework) and VIC (Video Conferencing Tool)**," ICDT 2006. International Conference on, 29-31 August 2006. [16] R. Gordon and S. Tally. **JMF Java Media Frame-work**. Prinice-Hall, November 1998.

[8] H. Abdel-Wahab, O. Kim, P. Kabore, and J.P. Favreau, "**Java-based Multimedia Collaboration and Application Sharing Environment**," Multimedia & Digital Video Technologies Group Information Technology Laboratory National Institute of Standards and Technology Gaithersburg, Md 20899, CFIP'99.,1999.

[9] กฤษฎิ์ธนิก ศรีธนสาร, สุวิพล สิทธิชีวภาค "เฟรมเรทที่ปรับเปลี่ยนได้และการเลือกเฟรมอัตโนมัติภายในบัฟเฟอร์บนเครื่องแม่ข่ายสำหรับผสานวีดีโอ" วิศวสารลาดกระบัง, ปีที่ 29 ฉบับที่ 3, กันยายน 2555, หน้า 25-30

ภาคผนวก

ภาคผนวก ก.

1. โปรแกรมส่งภาพด้วยวิธีไบต์อาเรย์ผ่านช่องสัญญาณความเร็วอัตราบิตต่ำ

โปรแกรมที่ทำงานด้านฝั่งเครื่องส่งข้อมูล โดยจะมีการทำงานแยกเป็น Class ต่างๆ สำหรับทำหน้าที่ในแต่ละส่วนไป ในกระบวนการส่งข้อมูลภาพไปยังเครื่องรับข้อมูล

คราสเชื่อมต่อกับพอร์ต RS232

```
package RS232Example;
import gnu.io.CommPortIdentifier;
import gnu.io.SerialPort;
import java.awt.image.BufferedImage;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.IOException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.imageio.ImageIO;
import vid2jpg.CompareFrame;
import vid2jpg.CompareFrame.RevByteListener;
import vid2jpg.Frame_encoderJPEG;
import vid2jpg.vid2jpg;
//import gnu.io.*;

public class RS232Example {
        public static MyForm myForm = new MyForm();
        public static vid2jpg myVideo = new vid2jpg("vfw://0");
        public static CompareFrame compare = new CompareFrame();
        static int numImgNxN = 1;   //4


        public void connect(String portName) throws Exception {
        CommPortIdentifier portIdentifier =
CommPortIdentifier.getPortIdentifier(portName);

        if (portIdentifier.isCurrentlyOwned()) {
            System.out.println("Port in use!");
        } else {
            // points who owns the port and connection timeout
            SerialPort serialPort = (SerialPort) portIdentifier.open("RS232Example", 2000);

            // setup connection parameters
            //Baud Rate 56000,#38400,19200,14400,9600,2400,1200
            serialPort.setSerialPortParams(
```

```
                19200, SerialPort.DATABITS_8, SerialPort.STOPBITS_1,
SerialPort.PARITY_NONE);

            // setup serial port writer
            CommPortSender.setWriterStream(serialPort.getOutputStream());

            // setup serial port reader
            new CommPortReceiver(serialPort.getInputStream()).start();
        }
    }

    public static void main(String[] args) throws Exception {

        // connects to the port which name (e.g. COM1) is in the first argument
        //new RS232Example().connect(args[0]);
          new RS232Example().connect("COM4");

         /*compare.addRevByteListener(
            new RevByteListener(){
               public void onRevByteData(int sequ, byte[] picData) {
                  for(int j = 0; j< picData.length; j++){
                     //System.out.println("MySubImg :"+sequ+" Index:"+j+"-->
"+picData[j]);
                  }
               }
               public void onRevByteData(int sequ, String label) {
                  System.out.println("Seque "+sequ+" Message "+label);
               }
            }
         );*/
          compare.setImageCellNxN(numImgNxN);

         // send HELO message through serial port using protocol implementation

        //---------- Send Only 1 buffered Image ----------
        myForm.setVisible(true);

        /*
```

```
        try{
            BufferedImage buff = ImageIO.read(new File("D:\\Sunset2.jpg"));
            Frame_encoderJPEG preEncode = new Frame_encoderJPEG();
            ByteArrayOutputStream preoutstream = preEncode.encodeJPEG(buff,5.5f);
            byte[] pre_sec_stream = preoutstream.toByteArray();
            System.out.println("Array Size: "+pre_sec_stream.length+"########");
            CommPortSender.send(pre_sec_stream);
        } catch (IOException ex){System.out.print("Test");}
        */

        //---------- End Send Only 1 buffered Image ---------


        //CommPortSender.send(new ProtocolImpl().getMessage("HELO"));
    }
}
```

คราสปรับขนาดของไบต์อาเรย์

```
package RS232Example;
public class ProtocolImpl implements Protocol {

    byte[] buffer = new byte[2097152];//1024,new524288,present460928
    int tail = 0;

    public void onReceive(byte b) {
        // simple protocol: each message ends with new line
        if (b=='\n') {
            onMessage();
        } else {
            buffer[tail] = b;
            tail++;
        }
    }

    public void onStreamClosed() {
        onMessage();
    }
```

```
    /*
     * When message is recognized onMessage is invoked
     */
    private void onMessage() {
        if (tail!=0) {
            // constructing message
            String message = getMessage(buffer, tail);
            System.out.println("RECEIVED MESSAGE: " + message);

            // this logic should be placed in some kind of
            // message interpreter class not here
            if ("HELO".equals(message)) {
                CommPortSender.send(getMessage("OK"));
            } else if ("OK".equals(message)) {
                CommPortSender.send(getMessage("OK ACK"));
            }
            tail = 0;
        }
    }

    // helper methods
    public byte[] getMessage(String message) {
        return (message+"\n").getBytes();
    }

    public String getMessage(byte[] buffer, int len) {
        return new String(buffer, 0, tail);
    }
}
```

คราสเปลี่ยนข้อมูลภาพเป็นบัฟเฟอร์อิมเมจ

```
package vid2jpg;
import RS232Example.CommPortSender;
import RS232Example.ProtocolImpl;
import com.sun.image.codec.jpeg.JPEGImageEncoder;
import java.awt.image.BufferedImage;
```

```
import java.io.ByteArrayOutputStream;
import java.util.EventListener;

public class CompareFrame {

    private BufferedImage[] preBufferedImage = null;
    private BufferedImage[] curBufferedImage = null;

    private ByteArrayOutputStream preoutstream = null;
    private ByteArrayOutputStream curoutstream = null;
    public ProtocolImpl myProtoCall =  new ProtocolImpl();

    private int devSection = 1;            // Image Cell = NxN = devSection^2
    private int NxN = 1;
    private float picquality = 5.5f;       //default
    RevByteListener myListener = null;

    public void setImageCellNxN(int sec) {
        devSection = sec;
        NxN = devSection*devSection;
    }

    public void inputBufferedImageSeries(BufferedImage[] images) {
        curBufferedImage = images;
        compareImages();
    }

    private void compareImages(){

        //IF preImgs = null create array and assign curImgs to preImgs
        //Else Compare Images and update
        if(preBufferedImage == null){
            preBufferedImage = new BufferedImage[NxN];
            for(int m=0; m <(devSection*devSection); m++){
                preBufferedImage[m] = curBufferedImage[m];
                System.out.println("----------First Frame Income--------");
            }
        }else{
```

```
            //System.out.println("--------Compare Section--------");
            for(int i=0; i<NxN; i++){
                Frame_encoderJPEG preEncode = new Frame_encoderJPEG();
                Frame_encoderJPEG curEncode = new Frame_encoderJPEG();

                preoutstream = preEncode.encodeJPEG(preBufferedImage[i],picquality);
                curoutstream = curEncode.encodeJPEG(curBufferedImage[i],picquality);

                byte[] pre_sec_stream = preoutstream.toByteArray();
                byte[] cur_sec_stream = curoutstream.toByteArray();

                //printByte(pre_sec_stream, i);

                //----------------------Camera On-----------------------
                //CommPortSender.send(pre_sec_stream);
                //CommPortSender.send(cur_sec_stream);
            }
        }
        //myListener.onRevByteData(23, "My name is");
    }

    private void printByte(byte[] pushByteArray, int sequence){
        for(int i = 0; i< pushByteArray.length; i++){
            //System.out.println("SubImg :"+sequence+" Index:"+i+"-->
"+pushByteArray[i]);

//CommPortSender.send(myProtoCall.getMessage(String.valueOf(pushByteArray[i])));
        }
        //System.out.println("----------------------");

        //CommPortSender.send(pushByteArray);
    }

    public interface RevByteListener extends EventListener{
        //public void onRevByteData(int sequ, byte picData[]);
        public void onRevByteData(int sequ, String label);
    }
    public void addRevByteListener(RevByteListener ls){
```

```
        myListener = ls;
    }
}
```

คราสการเข้ารหัสเป็นรูปแบบ JPEG

```
package vid2jpg;

import java.io.*;
import java.awt.*;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.media.*;
import javax.media.control.*;
import javax.media.format.*;
import javax.media.protocol.*;
import java.awt.image.*;
import com.sun.media.codec.video.jpeg.NativeEncoder;
import com.sun.image.codec.jpeg.*;
import java.util.ArrayList;
import javax.imageio.ImageIO;

public class vid2jpg extends Frame implements ControllerListener
{
        Processor p;
        Object waitObj = new Object();
        boolean stateOK = true;
        DataSourceHandler handler;
        imgPanel currPanel;int imgWidth; int imgHeight;
      private int sub_w, sub_h;                         // size of  subImage
        DirectColorModel dcm = new DirectColorModel(32, 0x00FF0000, 0x0000FF00,
0x000000FF);
      //DirectColorModel dcm = new DirectColorModel(32, 0x00FF0000, 0x0000FF00,
0x000000FF);
        MemoryImageSource sourceImage, sourceImage2;Image outputImage,
outputImage2;
        String sep = System.getProperty("file.separator");
        NativeEncoder e;
```

```
  int[] outvid;
  int startFr = 1;int endFr = 1000;int countFr = 0;
  boolean sunjava=true;

int frame_counter1 = 0;
int frame_counter2 = 0;

public static CompareFrame compare = new CompareFrame();

//------------ This part for Image Section nxn----------
static int numImgNxN = 9;
int subIndex = 0;

  /**
  * Static main method
  */
  public static void main(String[] args)
  {
        if(args.length == 0)
        {
              System.out.println("No media address.");
              new vid2jpg("vfw://0");

          compare.setImageCellNxN(numImgNxN);

        }
        else
        {
              String path = args[0].trim();
              System.out.println(path);
              new vid2jpg(path);
        }
  }

  /**
  * Constructor
  */
  public vid2jpg(String path)
```

```
    {
        MediaLocator ml;
      String args = path;

        if((ml = new MediaLocator(args)) == null)
        {
        System.out.println("Cannot build media locator from: " + args);
        }
        if(!open(ml))
        {
        System.out.println("Failed to open media source");
        }
  }

    /**
    * Given a MediaLocator, create a processor and start
    */
    private boolean open(MediaLocator ml)
    {
        System.out.println("Create processor for: " + ml);

        try
        {
            p = Manager.createProcessor(ml);
        }
        catch (Exception e)
        {
            System.out.println("Failed to create a processor from the given
media source: " + e);
            return false;
        }

        p.addControllerListener(this);

        // Put the Processor into configured state.
        p.configure();
        if(!waitForState(p.Configured))
        {
```

```
                System.out.println("Failed to configure the processor.");
                return false;
        }

        // Get the raw output from the Processor.
        p.setContentDescriptor(new
ContentDescriptor(ContentDescriptor.RAW));

        TrackControl tc[] = p.getTrackControls();
        if(tc == null)
        {
                System.out.println("Failed to obtain track controls from the
processor.");
                return false;
        }

        TrackControl videoTrack = null;
        for(int i = 0; i < tc.length; i++)
        {
                if(tc[i].getFormat() instanceof VideoFormat)
                {
                 tc[i].setFormat(new RGBFormat(new Dimension(640,480), -1,
Format.byteArray, -1.0F, 24, 3, 2, 1)); //720X540
                        videoTrack = tc[i];
                }
                else
                tc[i].setEnabled(false);
        }
        if(videoTrack == null)
        {
                System.out.println("The input media does not contain a video
track.");
                return false;
        }
        System.out.println("Video format: " + videoTrack.getFormat());

        p.realize();
        if(!waitForState(p.Realized))
```

```
        {
                System.out.println("Failed to realize the processor.");
                return false;
        }

        // Get the output DataSource from the processor and set it to the
DataSourceHandler.
        DataSource ods = p.getDataOutput();
        handler = new DataSourceHandler();
        try
        {
                handler.setSource(ods);         // also determines image size
        }
        catch(IncompatibleSourceException e)
        {
                System.out.println("Cannot handle the output DataSource from
the processor: " + ods);
                return false;
        }

        setLayout(new FlowLayout(FlowLayout.LEFT));
        currPanel = new imgPanel(new Dimension(imgWidth,imgHeight));
        add(currPanel);
        pack();
        //setLocation(100,100);
        setVisible(true);

        handler.start();

        // Prefetch the processor.
        p.prefetch();

        if(!waitForState(p.Prefetched))
        {
                System.out.println("Failed to prefetch the processor.");
                return false;
        }
```

```
            // Start the processor
            //p.setStopTime(new Time(20.00));
            p.start();

            return true;
        }

        /**
        * Sets image size
        */
        private void imageProfile(VideoFormat vidFormat)
        {
            System.out.println("Push Format "+vidFormat);
            Dimension d = (vidFormat).getSize();
            System.out.println("Video frame size: "+ d.width+"x"+d.height);

          //############ set perform size #########
            imgWidth=d.width;
            imgHeight=d.height;
        }
        /**
        * Called on each new frame buffer
        */
        private void useFrameData(Buffer inBuffer)
        {
          //############ unlimit to get frame #########
            //countFr++;
            //if(countFr<startFr || countFr>endFr)return;

            try
            {
                //printDataInfo(inBuffer);
              //showPixelvalue(new Dimension(imgWidth,imgHeight),inBuffer ,16);

                if(inBuffer.getData()!=null)       // vfw://0 can deliver nulls
                {
                    if(outvid==null)outvid = new
int[(imgWidth)*(imgHeight)];
```

```
                                outdataBuffer(outvid,(byte[])inBuffer.getData());

                            setImage(outvid);
                                // name so OS more likely to sort correctly if doing
jpg2vid
                                String paddedname =
"00000000000000000000"+inBuffer.getTimeStamp();
                                String sizedname =
paddedname.substring(paddedname.length()-20);

                                if(sunjava)
                                {
                               //############save to file #########

        saveJpeg(outputImage,"image_"+sizedname+".jpg");
                                //System.out.println("To File image_"+sizedname+".jpg");
                                }
                                else
                                {
                                    if(e==null){
                                  initJpeg((RGBFormat)inBuffer.getFormat());
                                  System.out.println("FORMAT = NULLLLLLLLL");
                                }

                                    byte[] b = fetchJpeg(inBuffer);
                                    String filename = "image_"+sizedname+".jpg";
                                    makeFile(filename, b);
                                }
                            }
                    }
                    catch(Exception e){System.out.println(e+" Array OutBound !!!");}
        }

        /**
        * Tidy on finish
        */
        public void tidyClose()
        {
```

```
                handler.close();
                p.close();
                if(e!=null)e.close();
                //dispose();      // frame
                System.out.println("Sources closed");
        }

        /**
        * Draw image to AWT frame
        */
      private void setImage(int[] outpix)
        {
                if(sourceImage==null)sourceImage = new
MemoryImageSource(imgWidth, imgHeight, dcm, outpix, 0, imgWidth);
                outputImage = createImage(sourceImage);
                currPanel.setImage(outputImage);
        }
        /**
        * Block until the processor has transitioned to the given state
        */
        private boolean waitForState(int state)
        {
                synchronized(waitObj)
                {
                        try
                        {
                                while(p.getState() < state && stateOK)
                                waitObj.wait();
                        }
                        catch (Exception e)
                        {
                        }
                }
                return stateOK;
        }
```

```
/**
* Controller Listener.
*/
public void controllerUpdate(ControllerEvent evt)
{
        if(evt instanceof ConfigureCompleteEvent ||
      evt instanceof RealizeCompleteEvent ||
      evt instanceof PrefetchCompleteEvent)
        {
                synchronized(waitObj)
                {
                        stateOK = true;
                        waitObj.notifyAll();
                }
        }
        else
        if(evt instanceof ResourceUnavailableEvent)
        {
                synchronized(waitObj)
                {
                        stateOK = false;
                        waitObj.notifyAll();
                }
        }
        else
        if(evt instanceof EndOfMediaEvent || evt instanceof StopAtTimeEvent)
        {
                tidyClose();
        }
}

/**
* Prints frame info
*/
private void printDataInfo(Buffer buffer)
{
        System.out.println(" Time stamp: " + buffer.getTimeStamp());
```

```
                System.out.println(" Time: " +
(buffer.getTimeStamp()/10000000)/100f+"secs");
                //System.out.println(" Sequence #: " + buffer.getSequenceNumber());
                System.out.println(" Data length: " + buffer.getLength());
                System.out.println(" Key Frame: " +
(buffer.getFlags()==Buffer.FLAG_KEY_FRAME)+" "+buffer.getFlags());
            System.out.println("Frame Conter 1: "+frame_counter1+" Frame Counter 2:
"+frame_counter2);
            System.out.println("--------------------------");
        }

          /**
          * Converts buffer data to pixel data for display
          */
          public void outdataBuffer(int[] outpix, byte[] inData) // could use
JavaRGBConverter
          {
                boolean flip=false;
                {
                    int srcPtr = 0;
                    int dstPtr = 0;
                    int dstInc = 0;
                    if(flip)
                    {
                        dstPtr = imgWidth * (imgHeight - 1);
                        dstInc = -2 * imgWidth;

                    }

                    for(int y = 0; y < imgHeight; y++)
                    {
                        for(int x = 0; x < imgWidth; x++)
                        {
                            byte red = inData[srcPtr + 2];
                            byte green = inData[srcPtr + 1];
                            byte blue = inData[srcPtr];
```

```
                                        int pixel = (red & 0xff) << 16 | (green & 0xff) <<
8 | (blue & 0xff) << 0;
                                        outpix[dstPtr] = pixel;
                                        srcPtr += 3;
                                        dstPtr += 1;
                                }
                                dstPtr += dstInc;
                        }
                }
                Thread.yield();
        }

        /**
        * Jpeg encoder and file writer
        */
        public void saveJpeg(Image img, String filename) throws IOException
        {
                BufferedImage bi = new BufferedImage(img.getWidth(null),
img.getHeight(null), BufferedImage.TYPE_INT_RGB);


            Graphics g = bi.getGraphics();
            g.drawImage(img, 0, 0, this);

            // -------- Split Image --------
            BufferedImage subIMG[] = splitImage(bi,numImgNxN);
            compare.inputBufferedImageSeries(subIMG);


        }

        /**
        * Inits a jpeg encoder - if using MS JVM - but frame sizes have to be multiples
of 8
        */
        private void initJpeg(RGBFormat vfin) throws Exception
        {
                float val=0.6F;
```

```
                int widpx=imgWidth;
                int hgtpx=imgHeight;

                // This encoder need multiples of 8 - use another if a problem
                if(widpx % 8 != 0 || hgtpx % 8 != 0)
                {
                        System.out.println("Width = "+imgWidth+" "+"Height =
"+imgHeight);
                        throw new Exception("Image sizes not /8");
                }

                VideoFormat vfout = new VideoFormat("jpeg", new
Dimension(widpx,hgtpx), widpx * hgtpx * 3, Format.byteArray, -1F);

              System.out.println("My DATA L :"+ widpx * hgtpx * 3);

                e = new NativeEncoder();
                e.setInputFormat(vfin);
                e.setOutputFormat(vfout);
                e.open();

                Control cs[] = (Control[])e.getControls();
                for (int i = 0; i < cs.length; i++)
                {
                        if (cs[i] instanceof QualityControl)
                        {
                                QualityControl qc = (QualityControl)cs[i];
                                qc.setQuality(val);
                                break;
                        }
                }
        }

        /**
        * Fetches a jpeg from buffer data - if using MS JVM
        */
        private byte[] fetchJpeg(Buffer inBuffer) throws Exception
        {
```

```
                Buffer outBuffer=new Buffer();// may need new to keep threadsafe if
extended
                int result = e.process(inBuffer, outBuffer);
                int lengthF = outBuffer.getLength();
                byte[] b = new byte[lengthF];
                System.arraycopy(outBuffer.getData(), 0, b, 0, lengthF);
                return b;
        }

        /**
        * Saves jpeg to file
        */
        public void makeFile(String filename, byte[] b)
        {
                BufferedOutputStream fw=null;
                try
                {
                        fw = new BufferedOutputStream(new
FileOutputStream("images"+sep+filename));
                        fw.write(b, 0, b.length);fw.close();

                        System.out.println(" *** Created file "+filename);
                }
                catch(IOException e ){System.out.println("makeFile "+e);}
        }

       /**
        * Show RGB ARRAy for 1 Frame
        * @param screen resolution
        * @param b buffer
        * @param Offset interval (present 1 pixel / n x n pixel)
        */
       public void showPixelvalue(Dimension screen, Buffer b, int Offset){
          int w0 = screen.width;
          int h0 = screen.height;
          int w = w0;
          int h = h0;
          int offset = 1;
```

```
        offset = Offset;
        byte[] x=(byte[])b.getData();
        System.out.println("Size of byte x[] "+x.length);
        ArrayList<String> raw = new ArrayList<String>();
            for(int i = 0 ; i < h ; i+=offset ){
                //col access loop
                String rawdata = "RGB";
                String preSpliter = "[";
                String posSpliter = "]";
                for(int j = 0 ; j < w ; j+=offset){
                   String RGB ="";
                   for(int k = 0 ; k < 3; k++){
                       RGB = RGB +x[3*(i*w+j)+k]+" : ";
                   }
                   rawdata = rawdata + preSpliter+RGB+posSpliter + " ";
                }
                //Add to ArrayList
                raw.add(rawdata);
             }
        //Print Array
        for(int m = 0; m < raw.size(); m++){
            System.out.println(raw.get(m));
        }
    }


/**
 * Get compress Pixel Array
 * @param screen
 * @param b
 * @param Offset
 * @return
 */
public void getCompressPixel(int[] outpix, Buffer b, int Offset){
    int w= imgWidth;
    int h = imgHeight;
    int offset = 1;
    offset = Offset;
```

```
        int count = 0;
        byte[] x=(byte[])b.getData();
        byte[] out = new byte[(w*h/Offset)/Offset*3];

        for(int i = 0 ; i < h ; i+=offset ){
            //col access loop
            for(int j = 0 ; j < w ; j+=offset){
                byte[] com = new byte[3];
                for(int k = 0 ; k < 3; k++){
                    com[k] = x[3*(i*w+j)+k] ;
                }
                byte red = com[2];
                byte green = com[1];
                    byte blue = com[0];
                int pixel = (red & 0xff) << 16 | (green & 0xff) << 8 | (blue & 0xff) << 0;
                outpix[j] = pixel;
            }
        }
    }
/**
 * Get SubImage (NxN) form 1 frame
 * @param bImg - Buffered Image (1 frame)
 * @param div  - Number of section (div = NxN )
 * @return  BufferedImage Array
 */

private BufferedImage[] splitImage(BufferedImage bImg, int div){
    BufferedImage imgs[] = new BufferedImage[div*div];
    //Dimension sub_dim = cal_subImg(bImg ,div);
    cal_subImg(bImg ,div);
    int r =0;
    while(r<imgs.length){
        for(int i=0; i<div; i++){
            for(int j=0; j<div; j++){
                imgs[r] = bImg.getSubimage(j*sub_w, i*sub_h, sub_w ,sub_h);
                r++;
            }
        }
```

```
        }
        return imgs;
    }

    private void cal_subImg(BufferedImage simg ,int split_nxn){
        sub_w = simg.getWidth()/split_nxn;
        sub_h = simg.getHeight()/split_nxn;
    }

        /**************************************************
        * Inner classes
        **************************************************/
        class DataSourceHandler implements BufferTransferHandler
        {
                DataSource source;
                PullBufferStream pullStrms[] = null;
                PushBufferStream pushStrms[] = null;
                Buffer readBuffer;

                /**
                * Sets the media source this MediaHandler should use to obtain
content.
                */
                private void setSource(DataSource source) throws
IncompatibleSourceException
                {
                        // Different types of DataSources need to handled differently.
                        if(source instanceof PushBufferDataSource)
                        {
                                pushStrms = ((PushBufferDataSource)
source).getStreams();

                                // Set the transfer handler to receive pushed data from
the push DataSource.
                                pushStrms[0].setTransferHandler(this);

                                // Set image size
```

```
                imageProfile((VideoFormat)pushStrms[0].getFormat());
            }
            else
            if(source instanceof PullBufferDataSource)
            {
                System.out.println("PullBufferDataSource!");

                // This handler only handles push buffer datasource.
                throw new IncompatibleSourceException();
            }

            this.source = source;
            readBuffer = new Buffer();
        }

        /**
        * This will get called when there's data pushed from the
PushBufferDataSource.
        */
        public void transferData(PushBufferStream stream)
        {
            try
            {
                stream.read(readBuffer);
            }
            catch(Exception e)
            {
                System.out.println(e);
                return;
            }

        //############ buffer Stream #########
            // Just in case contents of data object changed by some other
thread
            Buffer inBuffer = (Buffer)(readBuffer.clone());

            // Check for end of stream
            if(readBuffer.isEOM())
```

```
                {
                        System.out.println("End of stream");
                        return;
                }

                // Do useful stuff or wait
                useFrameData(inBuffer);
        }

        public void start()
        {
                try{source.start();}catch(Exception e){System.out.println(e);}
        }

        public void stop()
        {
                try{source.stop();}catch(Exception e){System.out.println(e);}
        }

        public void close(){stop();}

        public Object[] getControls()
        {
                return new Object[0];
        }

        public Object getControl(String name)
        {
                return null;
        }

}

/**
* Panel extension
*/
class imgPanel extends Panel
{
```

```
        Dimension size;
        public Image myimg = null;

        public imgPanel(Dimension size)
        {
                super();
                this.size = size;
        }

        public Dimension getPreferredSize()
        {
                return size;
        }

        public void update(Graphics g)
        {
                paint(g);
        }

        public void paint(Graphics g)
        {
                if (myimg != null)
                {
                        g.drawImage(myimg, 0, 0, this);
                }
        }

        public void setImage(Image img)
        {
                if(img!=null)
                {
                        this.myimg = img;
                        update(getGraphics());
                }
        }
    }
}
```

คราสสร้างฟอร์มฝั่งส่งข้อมูล

```
package RS232Example;
import java.awt.Color;
import java.awt.image.BufferedImage;
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.imageio.ImageIO;

public class MyForm extends javax.swing.JFrame {
    myShowDATA myDataPanel1 ;
    myShowDATA myDataPanel2 ;
    private boolean switch_trigger = false;
    /**
     * Creates new form MyForm
     */
    public MyForm() {
        initComponents();
        myDataPanel1 = myShowDATA.getInstance(jPanel1.getWidth(),
jPanel1.getHeight());
        myDataPanel2 = myShowDATA.getInstance(jPanel2.getWidth(),
jPanel2.getHeight());
        //myDataPanel = new myShowDATA();
    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        jButton1 = new javax.swing.JButton();
        jPanel2 = new javax.swing.JPanel();
```

```
        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jPanel1.setBorder(javax.swing.BorderFactory.createEtchedBorder());

        javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 499, Short.MAX_VALUE)
        );
        jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 324, Short.MAX_VALUE)
        );

        jButton1.setText("Start");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        jPanel2.setBorder(javax.swing.BorderFactory.createEtchedBorder());

        javax.swing.GroupLayout jPanel2Layout = new
javax.swing.GroupLayout(jPanel2);
        jPanel2.setLayout(jPanel2Layout);
        jPanel2Layout.setHorizontalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 499, Short.MAX_VALUE)
        );
        jPanel2Layout.setVerticalGroup(

        jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
```

```
            .addGap(0, 320, Short.MAX_VALUE)
        );

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addContainerGap()
                        .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(220, 220, 220)
                        .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE,
81, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(0, 212, Short.MAX_VALUE))
                    .addGroup(layout.createSequentialGroup()
                        .addContainerGap()
                        .addComponent(jPanel2, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))
                .addContainerGap())
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jPanel2, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jButton1)
                .addContainerGap())
```

```
    );

    pack();
}// </editor-fold>

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:

    myDataPanel1.setSize(jPanel1.getWidth(), jPanel1.getHeight());
    myDataPanel2.setSize(jPanel2.getWidth(), jPanel2.getHeight());

    if( switch_trigger ){              //Stop engine
        jButton1.setText("Start");    //Set label to next state
        jPanel1.removeAll();
        jPanel2.removeAll();
        jPanel1.setBackground(Color.black);
        jPanel2.setBackground(Color.red);
        myDataPanel1.stopPanel();
        myDataPanel2.stopPanel();
        switch_trigger = false;

    }else{                          //Start engine
        jButton1.setText("Stop"); //Set label to next state
        jPanel1.add(myDataPanel1);
        jPanel2.add(myDataPanel2);
        myDataPanel1.startPanel();
        myDataPanel2.startPanel();
        switch_trigger = true;

        //------------------- Read Image file --------------------
        try {
            //BufferedImage img   =  ImageIO.read(new File("D://Sunset.jpg"));
            //BufferedImage img   =  ImageIO.read(new File("D://Bluehills.jpg"));
            //BufferedImage img   =  ImageIO.read(new File("D://Bluehills640.jpg"));
            //BufferedImage img   =  ImageIO.read(new File("D://Bluehills320.jpg"));
            //BufferedImage img   =  ImageIO.read(new File("D://SVGA.jpg"));
            //BufferedImage img   =  ImageIO.read(new File("D://VGA.jpg"));
            //BufferedImage img   =  ImageIO.read(new File("D://QVGA.jpg"));
```

```
                BufferedImage img   =  ImageIO.read(new File("D://PAO.jpg"));
                //BufferedImage img   =  ImageIO.read(new File("D://Bluehills.png"));
                //BufferedImage img   =  ImageIO.read(new File("D://Water.jpg"));
                myDataPanel1.setImage(img);
                ByteArrayOutputStream imgByte1 = new ByteArrayOutputStream();
                ImageIO.write( img, "jpg", imgByte1 );
                imgByte1.flush();
                byte[] imageInByte = imgByte1.toByteArray();
                imgByte1.close();

                //prepare imageInByte to Send

                //------------- Send PIC ---------------
                CommPortSender.send(imageInByte);

                //-------------- Print Byte --------------
                for(int i=0; i < imageInByte.length; i++){
                  System.out.println("Size:"+imageInByte.length+" Indx: "+i+" -->
"+imageInByte[i]);
                }
                //----------------------------------------
                InputStream inStream = new ByteArrayInputStream(imageInByte);
                BufferedImage recieveImage = ImageIO.read(inStream);
                myDataPanel2.setImage(recieveImage);


            } catch (IOException ex) {
Logger.getLogger(MyForm.class.getName()).log(Level.SEVERE, null, ex);}
            //----------------- End Read Image file -----------------
        }
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
```

```
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code
(optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look
and feel.
         * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(MyForm.class.getName()).log(java.util.logging.Level.SE
VERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(MyForm.class.getName()).log(java.util.logging.Level.SE
VERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(MyForm.class.getName()).log(java.util.logging.Level.SE
VERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(MyForm.class.getName()).log(java.util.logging.Level.SE
VERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new MyForm().setVisible(true);
```

```
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    // End of variables declaration
}
```

ภาคผนวก ก.

2. โปรแกรมรับภาพด้วยวิธีไบต์อาเรย์ผ่านช่องสัญญาณความเร็วอัตราบิตต่ำ

โปรแกรมที่ทำงานด้านฝั่งเครื่องรับข้อมูล โดยจะมีการทำงานแยกเป็น Class ต่างๆ สำหรับทำหน้าที่ในแต่ละส่วน ในกระบวนการแปลงไฟล์ข้อมูลภาพที่รับมา เพื่อนำไปแสดงผล

คราสเชื่อมต่อกับพอร์ต RS232

```
package RS232Example;
import gnu.io.CommPortIdentifier;
import gnu.io.SerialPort;

public class RS232Example {
        //public static MyPanel myPanel;
        public static MyForm myForm = new MyForm();
        public static ProtocolImpl myProtocalImpl;
        public void connect(String portName) throws Exception {
        CommPortIdentifier portIdentifier =
CommPortIdentifier.getPortIdentifier(portName);

        if (portIdentifier.isCurrentlyOwned()) {
            System.out.println("Port in use!");
        } else {
            // points who owns the port and connection timeout
            SerialPort serialPort = (SerialPort) portIdentifier.open("RS232Example", 2000);

            // setup connection parameters
            //Baud Rate 56000,#38400,19200,14400,9600,2400,1200
            serialPort.setSerialPortParams(
                19200, SerialPort.DATABITS_8, SerialPort.STOPBITS_1,
SerialPort.PARITY_NONE);

            // setup serial port writer
            CommPortSender.setWriterStream(serialPort.getOutputStream());

            // setup serial port reader
            new CommPortReceiver(serialPort.getInputStream()).start();
        }
    }

    public static void main(String[] args) throws Exception {
        myProtocalImpl = new ProtocolImpl();
        myProtocalImpl.getMessage("Hello");
```

```
        myForm.setVisible(true);

        // connects to the port which name (e.g. COM1) is in the first argument
        //new RS232Example().connect(args[0]);
          new RS232Example().connect("COM4");

        // send HELO message through serial port using protocol implementation
        //CommPortSender.send(new ProtocolImpl().getMessage("HELO"));
        CommPortSender.send(myProtocalImpl.getMessage("Hello"));
    }

}
```

คราสรับค่าจากพอร์ต RS232

```
package RS232Example;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Image;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.imageio.ImageIO;
import javax.swing.ImageIcon;
import javax.swing.JLabel;

public class CommPortReceiver extends Thread {

    BufferedImage  myBufferIMG = null;
    InputStream in;
    Protocol protocol = new ProtocolImpl();

    public CommPortReceiver(InputStream in) {
        this.in = in;
    }
```

```
    public void run() {
        try {
            int b;
            while(true) {

                // if stream is not bound in.read() method returns -1
                while((b = in.read()) != -1) {
                    protocol.onReceive((byte) b);
                }
                protocol.onStreamClosed();

                // wait 10ms when stream is broken and check again
                sleep(10);
            }
        } catch (IOException e) {
            e.printStackTrace();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

คราสตอบรับระหว่างฝั่งส่งและจับเวลาการรับข้อมูล

```
package RS232Example;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.image.BufferedImage;
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.IOException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.imageio.ImageIO;
```

```
import javax.swing.ImageIcon;
import javax.swing.JLabel;
import javax.swing.JPanel;
import vid2jpg.Frame_decoderJPEG;
import javax.swing.JOptionPane;

public class ProtocolImpl extends JPanel implements Protocol {


    byte[] buffer = new byte[2097152]; //1024, 4664,
28808,#230400,#present460928,new524288

    int tail = 0;

    ByteArrayOutputStream outStream;
    ByteArrayInputStream inStream;
    Frame_decoderJPEG frameDecode;
    BufferedImage myBufferIMG;

    long startsnape;
    long stopsnpe;
    boolean complete_output = false;

    public void onReceive(byte b) {
        // simple protocol: each message ends with new line
        /* Original
        if (b=='\n') {
            onMessage();
        } else {
            buffer[tail] = b;
            tail++;
        }*/

        //My impliment
        buffer[tail] = b;
        tail++;
        /*if(tail == 4663){
            //onMessage();
```

```
            inStream = new ByteArrayInputStream(buffer);
            try {
                BufferedImage inbuff = ImageIO.read(inStream);
                RS232Example.myForm.inputIMAGE(inbuff);
            } catch (IOException ex) {
                Logger.getLogger(ProtocolImpl.class.getName()).log(Level.SEVERE, null, ex);
            }
            //RS232Example.myForm.inputIMAGE(frameDecode.decodeJPEG(outStream));
        }*/
    }

    public void onStreamClosed() {
         //---------------Print Time--------------------//

            startsnape = System.currentTimeMillis();
        //---------------Print Time--------------------//
        onMessage();
        inStream = new ByteArrayInputStream(buffer);
        try {
            BufferedImage inbuff = ImageIO.read(inStream);
            RS232Example.myForm.inputIMAGE(inbuff);
            complete_output = true;
            if(inbuff == null){
            }else{


                stopsnpe   = System.currentTimeMillis();
                System.out.println("process time = "+(stopsnpe-startsnape)+" ms");
                JOptionPane.showMessageDialog(null, stopsnpe-startsnape );
                startsnape = 0;   //reset
                stopsnpe   = 0;   //reset


            }

        } catch (IOException ex) {
            Logger.getLogger(ProtocolImpl.class.getName()).log(Level.SEVERE, null, ex);
        }
```

```
            //-----------------Print Time---------------------//

            //-----------------Print Time---------------------//
        }

        /*
         * When message is recognized onMessage is invoked
         */

        private void onMessage() {
            if (tail!=0) {
                // constructing message
                String message = getMessage(buffer, tail);
                //System.out.println("RECEIVED MESSAGE: " + message);

                //Print Bit-------------------//
//                for(int i=0; i<buffer.length; i++ ){
//                    System.out.println("Size: "+buffer.length+" Indx: "+i+" ["+buffer[i]+"] Tail:
"+tail);
//                }

                // this logic should be placed in some kind of
                // message interpreter class not here
                if ("HELO".equals(message)) {
                    CommPortSender.send(getMessage("OK"));
                } else if ("OK".equals(message)) {
                    CommPortSender.send(getMessage("OK ACK"));
                }
                tail = 0;
            }
        }

        @Override
        public void paint(java.awt.Graphics graphics) {
            super.paint(graphics);
            java.awt.Graphics2D gr=(java.awt.Graphics2D) graphics;
            gr.drawImage(myBufferIMG, 0, 0, null); // see javadoc for more info on the
    parameters
```

```
    }

    // helper methods
    public byte[] getMessage(String message) {
        return (message+"\n").getBytes();
    }

    public String getMessage(byte[] buffer, int len) {
        return new String(buffer, 0, tail);
    }
}
```

คราสสร้างฟอร์มฝั่งรับข้อมูล

```
package RS232Example;
import java.awt.Color;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.imageio.ImageIO;

public class MyForm extends javax.swing.JFrame {
    myShowDATA myDataPanel ;
    private boolean switch_trigger = false;

    public MyForm() {
        initComponents();
        myDataPanel = myShowDATA.getInstance(jPanel1.getWidth(),
jPanel1.getHeight());
        //myDataPanel = new myShowDATA();
    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {
```

```
        jPanel1 = new javax.swing.JPanel();
        jButton1 = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jPanel1.setBorder(javax.swing.BorderFactory.createEtchedBorder());

        javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 505, Short.MAX_VALUE)
        );
        jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 324, Short.MAX_VALUE)
        );

        jButton1.setText("Start");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addContainerGap())
```

```
            .addGroup(layout.createSequentialGroup()
                .addGap(220, 220, 220)
                .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 81,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jButton1, javax.swing.GroupLayout.DEFAULT_SIZE, 39,
Short.MAX_VALUE)
                .addContainerGap())
        );

        pack();
    }// </editor-fold>

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:

        myDataPanel.setSize(jPanel1.getWidth(), jPanel1.getHeight());

        if( switch_trigger ){              //Stop engine
            jButton1.setText("Start");    //Set label to next state
            jPanel1.removeAll();
            jPanel1.setBackground(Color.black);
            myDataPanel.stopPanel();
            switch_trigger = false;

        }else{                             //Start engine
            jButton1.setText("Stop"); //Set label to next state
            jPanel1.add(myDataPanel);
            myDataPanel.startPanel();
```

```
            switch_trigger = true;

            //------------------- Read Image file -------------------
            /*try {
                BufferedImage img   =  ImageIO.read(new File("D:\\Water.jpg"));
                myDataPanel.setImage(img);
            } catch (IOException ex) {
Logger.getLogger(MyForm.class.getName()).log(Level.SEVERE, null, ex);}
            //----------------- End Read Image file -----------------
             *
             */
        }
    }

    public void inputIMAGE(BufferedImage img){ myDataPanel.setImage(img);}

    public static void main(String args[]) {

        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(MyForm.class.getName()).log(java.util.logging.Level.SE
VERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(MyForm.class.getName()).log(java.util.logging.Level.SE
VERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(MyForm.class.getName()).log(java.util.logging.Level.SE
VERE, null, ex);
```

```
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(MyForm.class.getName()).log(java.util.logging.Level.SE
VERE, null, ex);
        }
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new MyForm().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JPanel jPanel1;
    // End of variables declaration
}
```

คราสการแสดงผลการรับข้อมูล

```
package RS232Example;
import java.awt.image.BufferedImage;
import static java.lang.Thread.sleep;
import java.util.logging.Level;
import java.util.logging.Logger;

public class myShowDATA extends javax.swing.JPanel  implements Runnable{

    public Thread interacThread;
    private BufferedImage image_;

    /**
     * Creates new form myShowDATA
     */
    public myShowDATA() {
```

```
        initComponents();
    }

    public void startPanel(){
        interacThread   = new Thread(this);
        interacThread.start();
    }

    public void stopPanel(){
        interacThread = null;
    }

    public void setImage(BufferedImage buffIMG){
        image_ = buffIMG;
    }

    public void paintComponent( java.awt.Graphics graphics ){
        super.paintComponent(graphics);
        java.awt.Graphics2D gr=(java.awt.Graphics2D) graphics;

        //gr.setColor(new Color(255,0,0)); //Red
        //gr.fillOval(0, 0, getWidth()/2, getWidth()/2);
        gr.drawImage(image_,null, 0, 0);
    }

    public static myShowDATA getInstance(int W, int H) {
        myShowDATA panel=new myShowDATA();
        panel.setPreferredSize(new java.awt.Dimension(W,H));
        return panel;
    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
        this.setLayout(layout);
        layout.setHorizontalGroup(
```

```
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 301, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 249, Short.MAX_VALUE)
        );
    }// </editor-fold>


    // Variables declaration - do not modify
    // End of variables declaration

    @Override
    public void run() {
        while(interacThread == Thread.currentThread()){
            try {
                sleep(1000);
                repaint();
                System.out.println("TMM");
            } catch (InterruptedException ex) {
                Logger.getLogger(myShowDATA.class.getName()).log(Level.SEVERE, null,
ex);
            }
        }
    }
}
```

ภาคผนวก ข.

1. ผลงานวิจัยที่ได้รับการตีพิมพ์เผยแพร่

Panithan Boonjanawirod , Krittanik Srithanasarn, Suvepon Sittichivapak “A New Method for Image Communication via Low Bit Rate Channel”, The 20th International Computer Science and Engineering Conference 2016, 14 - 17 December 2016, Chiang Mai Orchid Hotel, Chiang Mai, Thailand

can be used infirefighters tracking, logistics and military service. Thissystem requires high accuracy in an indoor environment.In this paper evaluate and comparison betweenTrilateration and Min-Max localization techniques usingreceived signal strength (RSS) and time of arrival (TOA)parameters. This study based on the ultra-wideband(UWB) transmission model for an indoor localization. Thevector network analyzer (VNA) was used to measurementand recorders at FCC frequency band from 3.1 GHz to10.6 GHz. Using the biconical antennas were used as bothtransmitter (Tx) and receiver (Rx) antennas. The accuracyand precision are also studies in the term of distance error.The cumulative distribution function (CDF) of distanceerror of each case is shown. From the results, thetrilateration technique using TOA parameters very goodaccuracy.

E2-2A.6: A New Method for Image Communication via Low Bit Rate Channel

Panithan Boonjanawirod, Krittanik Srithanasarn, Suvepon Sittichivapak

Abstract:

This article presents a new method for image communication via low bit rate channel that transfer JPEG image to buffered Image storage and encoding to Byte Array Output Stream then receiving byte Array data that ready to transfer by used Byte Array processing method. Then Byte Array will be ransferred via serial port with Low Bit Rate channel to destination. After destination received data, the data will be changed Byte Array to Byte Array Output Stream by using Byte Array processing method as same as source data. Then, Byte Array Output Stream is decoded and transferred to Buffered Image storage for showing the result. This

ประวัติผู้เขียน

ชื่อ-นามสกุล	นายปณิธาน บุญจนาวิโรจน์
วัน เดือน ปีเกิด	2 กันยายน 2529 ที่จังหวัดเชียงราย
ที่อยู่	56/229 หมู่บ้านพฤษา 86 ตำบลศีรษะจรเข้น้อย อำเภอบางเสาธง สมุทรปราการ 10540
ประวัติการศึกษา	2553 วิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์ มหาวิทยาลัยเทคโนโลยีราชมงคลธัญบุรี
ความชำนาญเฉพาะด้าน	1.) โครงสร้างข้อมูลและการโปรแกรมคอมพิวเตอร์ 2.) ระบบคอมพิวเตอร์เน็ตเวิร์ค

ประสบการณ์การทำงานและผลงานวิจัย

พ.ศ.2554-2556	ตำแหน่งนักวิชาการคอมพิวเตอร์ วิทยาลัยนวัตกรรมการจัดการข้อมูล สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ.2556-2559	ตำแหน่งวิศวกร วิทยาลัยนวัตกรรมการจัดการข้อมูล สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ.2559-ปัจจุบัน	ศึกษาต่อระดับปริญญาโท