การบีบอัดข้อมูลคลื่นหัวใจ โดยวิธีเลมเพลซิว

ECG Compression Using Lempel – Ziv Technique

นายกมล ธีระกาญจน์ รหัส 46361564

นายกิตติภูมิ สมศรี รหัส 46363107

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

ปีการศึกษา 2549

ใบรับรองโครงงานวิศวกรรม

หัวข้อโครงงาน	การบีบอัดข้อมูลคลื่นหัวใจโดยวิธีเลมเพลซิว			
ผู้ดำเนินโครงงาน	นายกมล	ธีระกาญจน์	รหัส	46361564
	นายกิตติภูมิ	สมศรี	รหัส	46363107
อาจารย์ที่ปรึกษา	ผู้ช่วยศาสตราจารย์ ดร.สุชาติ แย้มเม่น			
สาขาวิชา	วิศวกรรมไฟฟ้า			
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์			
ปีการศึกษา	2549			

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้โครงงานฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมไฟฟ้า

คณะกรรมการสอบโครงงานวิศวกรรม

.....ประธานกรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.สุชาติ แย้มเม่น)

.....กรรมการ

(ดร.ชัยรัตน์ พินทอง)

.....กรรมการ

(ดร.พนมขวัญ ริยะมงคล)

หัวข้อโครงงาน	การบีบอัดข้อมูลคลื่นหัวใจโดยวิธีเลมเพลซิว
ผู้ดำเนินโครงงาน	นายกมล ธีระกาญจน์ รหัส 46361564
	นายกิตติภูมิ สมศรี รหัส 46363107
อาจารย์ที่ปรึกษา	ผู้ช่วยศาสตราจารย์ ดร.สุชาติ แย้มเม่น
สาขาวิชา	วิศวกรรมไฟฟ้า
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์
ปีการศึกษา	2549

บทคัดย่อ

โครงงานฉบับนี้ ได้ทำการศึกษา ออกแบบ และทดสอบโปรแกรมการบีบอัดข้อมูลคลื่นหัวใจ (ECG) โดยใช้วิธีการบีบอัดข้อมูลแบบเลมเพลซิว (Lempel-Ziv) ซึ่งเป็นการนำอักขระของข้อมูลนำมาจัดวางอยู่ในรูปแบบตารางจัดเก็บข้อมูลเริ่มแรก ต่อมานำอักขระในลำดับถัดไปมาทำการวิเคราะห์ว่าข้อมูลเหล่านี้มีการซ้ำกันของอักระเดิมที่ถูกจัดเก็บหรือไม่ ถ้ามีการซ้ำกันในตารางแล้ว ให้ทำการพิจารณาตัวอักขระถัดไปเรื่อยๆ ถ้าอักขระไม่มีการซ้ำกันกับข้อมูลในตาราง ให้บันทึกอักขระที่ไม่ซ้ำนั้นเข้าไปในตารางจัดเก็บข้อมูลทันที ต่อมานำอักขระที่เหลืออยู่เปรียบเทียบการซ้ำของข้อมูลในตารางเดิมอีกครั้งหนึ่ง ถ้าอักขระยังไม่มีการซ้ำกันช่องข้อมูลในตาราง ให้บันทึกค่าที่ไม่ซ้ำอีกครั้งหนึ่ง ทำจนหมดอักขระตัวสุดท้ายเมื่อสิ้นสุดข้อมูลก็จะทำการบันทึกค่าจากอักขระที่ซ้ำมาเป็นข้อมูลที่ถูกการบีบอัด

จากผลการบีบอัดข้อมูล ECG พบว่าอัตราการบีบอัดข้อมูลสูงสุดมีค่าเท่ากับ 1.6544 โดย SNR และ PSNR มีค่าเข้าใกล้เลขอนันต์ ซึ่งแสดงถึงข้อมูลที่ถูกบีบอัดไม่มีการสูญเสีย

Project Title	ECG Data Compression Using Ziv-Lempel Techniques
Name	Mr. Kamon Theerakarn ID. 46361564
	Mr. Kittipoom Somsri ID. 46363107
Project Advisor	Assistance Professor Suchart Yamman , Ph.D.
Major	Electrical Engineering
Department	Electrical and Computer Engineering
Academic Year	2006

.....

Abstract

The project is to study, design and test a program for data compression in ECG (Electrocardiogram) by using the Lempel Ziv method which puts on the first character of ECG data in form of table format. Next, the subsequence character is brought to analyze and check about repetition of characters. If so, the program will find out the next character. If not, the character will be saved in the table. Then, the program will do it in the same previous fashion until the last character is checked. After that all repeated character are compressed into a binary file.

From the ECG compression result, it has been found that the maximum value of the compression ratio is 1.6544 while SNR and PSNR approach infinity. This shows that the compressed data (ECG) are lossless.

กิตติกรรมประกาศ

โครงงานวิศวกรรมไฟฟ้าฉบับนี้ สามารถดำเนินการให้สำเร็จลุล่วงมาได้เป็นอย่างดี เนื่องจากการได้รับความกรุณาเป็นอย่างสูงจากอาจารย์ที่ปรึกษาโครงงานคือ ผู้ช่วยศาสตราจารย์ ดร.สุชาติ แย้มเม่น ซึ่งท่านได้เป็นผู้ให้ความรู้ คำแนะนำ ตลอดจนให้ความช่วยเหลือในทุกๆ ด้าน จนโครงงานเสร็จสมบูรณ์ ผู้ดำเนินโครงงานจึงขอขอบพระคุณอาจารย์ไว้ ณ ที่นี้

ขอขอบพระคุณ ดร.ชัยรัตน์ พินทอง และ ดร.พนมขวัญ ริยะมงคล กรรมการผู้ทรงคุณวุฒิที่ได้ให้คำปรึกษาแนะนำ และตรวจสอบแก้ไขข้อบกพร่อง จนโครงงานวิศวกรรมไฟฟ้าฉบับนี้เสร็จสมบูรณ์

ขอขอบพระคุณบิดา มารดา ตลอดจนญาติพี่น้องของผู้ดำเนินโครงงาน ที่ได้ให้กำลังใจ และให้ความสนับสนุนเป็นอย่างดี

ขอขอบพระคุณทุกท่านที่ยังไม่ได้เอ่ยนาม ที่ให้ความช่วยเหลือในด้านต่างๆ จนทำให้โครงงานประสบผลสำเร็จ

สุดท้าย คณะผู้ดำเนินโครงงานวิศวกรรมไฟฟ้า ขอมอบคุณงามความดี แด่ผู้มีพระคุณทุกท่าน และถ้าหากผู้มีความรู้ท่านใด ที่ได้พบข้อผิดพลาดหรือข้อบกพร่องประการใด ของโครงงานวิศวกรรมไฟฟ้าฉบับนี้ คณะผู้ดำเนินโครงงานต้องขออภัยมา ณ ที่นี้ด้วย และพร้อมที่จะรับฟังคำชี้แนะเพื่อนำไปปรับปรุงในโอกาสต่อไป

คณะผู้จัดทำ

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญ

เนื่องจากข้อมูลทางภาพการเต้นของหัวใจนั้นเป็นสิ่งที่สำคัญ จะล่าช้าหรือเสียหายไม่ได้ หากล่าช้าไปแค่ 10 นาที คนไข้อาจจะตายได้ หรือหากเราได้ข้อมูลผิดพลาดไปอาจจะทำให้ชีวิตคนไข้เป็นอันตรายได้ ดังนั้นกลุ่มของข้าพเจ้าจึงคิดโครงงานนี้ขึ้นมา ซึ่งเป็นโครงงานที่ใช้วิธีการบีบอัดข้อมูลประเภทหนึ่ง ที่เรียกว่า วิธีของลิมเพลซิว (Lempel -Ziv Techniques) ซึ่งเป็นวิธีการบีบอัดข้อมูลที่ใช้ได้ผลในการบีบอัดและคลายการบีบอัดข้อมูล ซึ่งเหมือนกับโปรแกรมวินซิปที่เราใช้อยู่ในปัจจุบัน ซึ่งวิธีการบีบอัดจะอยู่ในส่วนของทฤษฎี วิธีการบีบอัดนี้สามารถได้ข้อมูลเดิม แต่มีขนาดน้อยลง และเนื่องจากมีขนาดน้อยลงทำให้สามารถบรรจุข้อมูลอื่นได้มากขึ้น

1.2 วัตถุประสงค์

เพื่อที่จะศึกษาวิธีการบีบอัดข้อมูลโดยวิธีลิมเพลซิว

1.3 ขอบเขตของการทำโครงงาน

1. ต้องการที่จะทำโปรแกรมการบีบอัดข้อมูลลิมเพลซิว
2. ต้องการดูข้อมูลที่จะบีบอัดว่าลดขนาดข้อมูลได้จริงและไม่มีการสูญเสีย

1.4 แนวทางการทำโครงงาน

แนวทางทำโครงงาน ทำโครงงานจากการศึกษาบีบอัดข้อมูล โดยวิธีลิมเพลซิว และทำการศึกษาโปรแกรม MATLAB

1.5 ขั้นตอนของการดำเนินงาน

ศึกษาวิธีการบีบอัดโดยใช้เทคนิคลิมเพลซิว
ศึกษาการใช้โปรแกรม MATLAB
เขียนโปรแกรมบีบอัดข้อมูล
ทดสอบและแก้ไขข้อมูล
จัดทำเป็นรูปเล่ม

1.6 แผนการทำงาน (Gatt chart)

ตารางที่ 1.1 แผนการทำงาน

กิจกรรม	ระยะเวลาดำเนินการ (เดือน)												ผู้รับผิดชอบ/ผู้ปฏิบัติ	สถานที่ในการดำเนินการวิจัย
	1	2	3	4	5	6	7	8	9	10	11	12		
1. การศึกษาข้อมูล และหาความรู้เพิ่มเติม	⟷	⟷											นายกมล ธีระกาญจน์ นายกิตติภูมิ สมศรี	ศึกษาที่ IRPUS
2. การออกแบบจำลองโครงสร้างการทำงาน			⟷										นายกมล ธีระกาญจน์ นายกิตติภูมิ สมศรี	ศึกษาที่ IRPUS
3. การเขียนโปรแกรม ทดสอบการทำงานจากแบบจำลอง				⟷	⟷	⟷							นายกมล ธีระกาญจน์ นายกิตติภูมิ สมศรี	ศึกษาที่ IRPUS
4. การทดสอบประสิทธิภาพการทำงานและการSimulation							⟷	⟷					นายกมล ธีระกาญจน์ นายกิตติภูมิ สมศรี	ศึกษาที่ IRPUS
5. การแก้ไข และประเมินประสิทธิภาพระบบการทำงาน									⟷	⟷			นายกมล ธีระกาญจน์ นายกิตติภูมิ สมศรี	ศึกษาที่ IRPUS
6. การประเมินผลรวบยอด ในทุกๆด้าน											⟷		นายกมล ธีระกาญจน์ นายกิตติภูมิ สมศรี	ศึกษาที่ IRPUS
7. จัดทำรายงานและเตรียมนำเสนอผลงาน												⟷	นายกมล ธีระกาญจน์ นายกิตติภูมิ สมศรี	ศึกษาที่ IRPUS

1.7 ผลที่คาดว่าจะได้รับ

สามารถบีบอัดข้อมูลโดยที่เวลาบีบอัดข้อมูลแล้ว เวลาคลายข้อมูลยังได้รูปเดิมออกมา

1.8 งบประมาณ

ค่าวัสดุ	1000 บาท
ค่าใช้สอย	1000 บาท
รวมเป็นเงิน	**2000 บาท**
	(สองพันบาทถ้วน)

(หมายเหตุ) ถัวเฉลี่ยทั้งหมด

บทที่ 2
ทฤษฎีและหลักการที่เกี่ยวข้อง

2.1 ข้อมูลเบื้องต้นของ ECG

การรักษาทางการแพทย์ในกรณีที่อยู่ไกลกัน แต่กรณีควบคู่ไปกับเครือข่ายคอมพิวเตอร์แพทย์ต้นทางและปลายทางสามารถติดต่อกันด้วยภาพเคลื่อนไหวและเสียง ทำให้สามารถแลกเปลี่ยนข้อมูลคนไข้ระหว่างกันและกัน ทั้งทางด้านภาพเช่นฟิล์มเอกซเรย์และเสียง สัญญาณจากเครื่องมือแพทย์เช่น การเต้นของหัวใจคลื่นหัวใจ (ECG) พร้อมๆกับแลกเปลี่ยนประสบการณ์ และปรึกษาหารือกันเสมือนกับแพทย์ต้นทางแพทย์ปลายทาง และคนไข้อยู่ในห้องเดียวกัน นอกจากนั้นการแพทย์ทางไกลยังนำมาใช้ในการประชุมปรึกษาหารือกันทางไกล (Video Conferencing) การศึกษาต่อเนื่องทางไกล (Distance Learning) และการเชื่อมโยงเครือข่ายคอมพิวเตอร์ระหว่างส่วนกลางและส่วนภูมิภาคอีกด้วย กระทรวงสาธารณสุขได้ตระหนักถึงปัญหา ในการขาดแคลนแพทย์ผู้เชี่ยวชาญในท้องที่ชนบทห่างไกล ซึ่งทำให้ประชาชนที่เจ็บป่วยมุ่งเข้ามารับการรักษาพยาบาลในโรงพยาบาลใหญ่ๆในเมือง อันทำให้เป็นภาระหนักของโรงพยาบาลเหล่านั้น และเป็นภาระทางด้านค่าใช้จ่ายและเวลาของประชาชนที่ต้องเดินทางเข้ามารับบริการกระทรวงสาธารณสุข จึงได้ริเริ่มดำเนินงานโครงแพทย์ทางไกลผ่านดาวเทียมขึ้น โดยมีวัตถุประสงค์เพื่อเพิ่มประสิทธิภาพ ในการให้บริการรักษาพยาบาลของโรงพยาบาลชุมชนที่มีแพทย์ประจำอยู่อย่างจำกัด และเพื่อพัฒนาบุคลากรทางการแพทย์และสาธารณสุขในท้องถิ่นห่างไกล โดยจัดการเรียนการสอนทางไกล เพื่อให้บุคลากรเหล่านั้นได้มีโอกาสศึกษาต่อเนื่องเพิ่มเติมโดยไม่ต้องลาเรียนต่อและได้อยู่ ปฏิบัติงานที่หน่วยงานต่อไปได้ โดยได้จัดทำแผนการดำเนินงานโครงการแพทย์ทางไกลผ่านดาวเทียมเสนอต่อคณะรัฐมนตรีเพื่ออนุมัติ ซึ่งคณะรัฐมนตรีในการประชุมเมื่อวันที่ 30 สิงหาคม 2537 ได้ ให้ความเห็นชอบให้กระทรวงสาธารณสุขดำเนินงานโครงการแพทย์ทางไกลผ่านดาวเทียมระยะเวลา 4 ปี โดยเริ่มตั้งแต่ปี 2538 -2541

2.2 เซลล์ไฟฟ้าหัวใจ

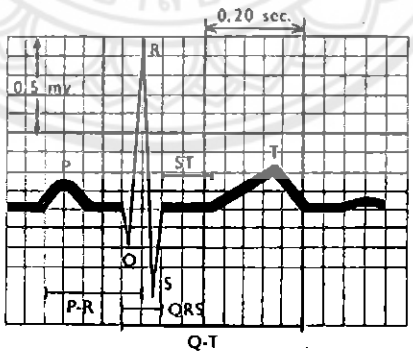
หัวใจของคนเราประกอบไปด้วยหัวใจห้องล่างสองห้องคือซ้ายและขวา(Left or Right Ventricle)และหัวใจห้องบนสองห้องคือขวาและซ้าย (Left or Right Atrium) การทำงานของหัวใจจะทำหน้าที่สูบฉีดเลือดไปเลี้ยงร่างกาย หัวใจจะได้รับไฟฟ้าซึ่งเกิดจากเซลล์ชนิดพิเศษในหัวใจซึ่งสามารถสร้างกระแสไฟฟ้าเองซึ่งเรียกว่า Sinus node กระแสไฟฟ้าจะวิ่งผ่านไปตามกล้ามเนื้อหัวใจทำให้หัวใจบีบตัว

2.3 คลื่นไฟฟ้าหัวใจ

การตรวจคลื่นไฟฟ้าหัวใจไม่ใช่เครื่องเช็คหัวใจ ไม่ใช่ว่าตรวจออกมาเป็นกราฟ แล้วจะบอกความผิดปกติทุกอย่าง ความเป็นจริงแล้วคลื่นไฟฟ้าหัวใจบอกให้เราทราบข้อมูลเกี่ยวกับหัวใจค่อนข้างจำกัด เช่น จังหวะการเต้น ความสม่ำเสมอ การนำไฟฟ้าในหัวใจ ชนิดของการเต้นผิดจังหวะ (ตรวจขณะมีอาการ) หัวใจโตหรือไม่ กล้ามเนื้อหัวใจตาย กล้ามเนื้อหัวใจขาดเลือด (ในบางรายเท่านั้น เพราะต้องตรวจขณะมีอาการ) ความผิดของระดับเกลือแร่บางชนิดในร่างกาย เป็นต้น ข้อมูลที่ได้มาก็ต้องนำมาแปลผลอีกครั้ง โดยอาศัยประวัติ การตรวจร่างกาย ความชำนาญของแพทย์ จึงจะสรุปอีกครั้งว่าคลื่นไฟฟ้า หัวใจผิดปกติหรือไม่ บ่อยครั้งที่ผู้ป่วยได้รับการบอกจากแพทย์ว่าเป็นโรคหัวใจโต หลอดเลือดตีบ 3 เส้น ทั้งๆที่ผู้ป่วยไม่ได้เป็นเลย เนื่องจากแพทย์ขาดความรู้ความชำนาญในการนำข้อมูลที่ได้จาก ECG มาแปลผลอย่างถูกต้อง

2.4 การตรวจคลื่นไฟฟ้าหัวใจ (Electrocardiogram : ECG)

การตรวจคลื่นไฟฟ้าหัวใจเป็นวิธีการตรวจหาโรคหัวใจที่ง่ายและได้ผลดี การตรวจคลื่นไฟฟ้าหัวใจเป็นตรวจกระแสไฟฟ้าที่กล้ามเนื้อหัวใจผลิตออกมาขณะที่หัวใจบีบตัว โดยเริ่มต้นที่จุดที่เซลล์กล้ามเนื้อหัวใจชนิดพิเศษที่สามารถสร้างกระแสไฟฟ้าได้เอง เราเรียกจุดนี้ว่า Sinus node กระแสไฟฟ้าจะวิ่งผ่านกล้ามเนื้อหัวห้องบน เกิดกระแสไฟฟ้าที่เราตรวจได้เรียก P wave กระแสจะมาหยุดที่รอยต่อระหว่างหัวใจห้องบนและห้องล่างเรียกว่า AV Node หลังจากนั้นกระแสไฟฟ้าจะวิ่งไปหัวใจห้องข่างล่างทั้งซ้ายและขวาและทำให้เกิดกระแสไฟฟ้าที่เรียกว่า QRS complex ดูตัวอย่างกราฟไฟฟ้าหัวใจของคนปกติ



รูปที่ 2.1 รูปกราฟหัวใจ

2.5 จุดอ่อนของการตรวจคลื่นไฟฟ้าหัวใจ

คลื่นไฟฟ้าหัวใจปกติ ไม่ได้หมายความว่าหัวใจปกติ ปราศจากโรคในโรคหัวใจขาดเลือดหรือหลอดเลือดหัวใจตีบนั้น คลื่นไฟฟ้าหัวใจจะผิดปกติก็ต่อเมื่อเป็นโรคขั้นรุนแรงจนเกิดกล้ามเนื้อหัวใจตายแล้วถ้าเป็นเพียงหลอดเลือดตีบแต่ไม่รุนแรง ก็อาจตรวจไม่พบได้นอกจากการตรวจคลื่นไฟฟ้าหัวใจขณะ ออกกำลังกายจะช่วยเพิ่มความไวในการตรวจขึ้นหัวใจโต การอ่านคลื่นไฟฟ้าหัวใจว่ามีหัวใจโตก็เป็นสิ่งที่ต้องระมัดระวังมาก เพราะการแปลผลอาศัย ความสูงของคลื่นไฟฟ้า (voltage) เป็นสำคัญ ความสูงของคลื่นนี้ก็แปรผันมากเหลือเกิน ทั้งความอ้วน ผอม โรคปอด อายุ ฯลฯ ดังนั้นบ่อยครั้งที่คลื่นไฟฟ้าหัวใจบอกว่าโต แต่ความจริงแล้วไม่โตก็ได้ ในทางตรงกันข้ามการตรวจขนาดของหัวใจโดยอาศัยคลื่นไฟฟ้าหัวใจนั้น มีความไวที่ต่ำมาก หมายความว่า หัวใจอาจจะโต โดยที่คลื่นไฟฟ้าหัวใจปกติ

คลื่นไฟฟ้าหัวใจไม่ได้บอกความผิดปกติของลิ้นหัวใจ หรือ หลอดเลือดหัวใจโดยตรง แต่เป็นการตรวจผลเสียที่เกิดขึ้นเนื่องจากโรคของลิ้นหัวใจหรือหลอดเลือดหัวใจต่างหากใน หลายกรณีการตรวจ จะได้ประโยชน์เฉพาะเมื่อตรวจขณะเกิดอาการ เช่น ใจสั่น หัวใจเต้นผิดปกติ เจ็บหน้าอก เป็นต้น ถ้าเช่นนั้นใครบ้างควรตรวจ ECG ในกรณีที่อายุน้อย สบายดี ไม่มีอาการผิดปกติ ตรวจร่างกายปกติ การตรวจคลื่นไฟฟ้าหัวใจก็ไม่มีประโยชน์เลยในกรณีที่อายุมาก เช่นมากกว่า 40 ปี แข็งแรงดี ไม่มีอาการผิดปกติ ตรวจร่างกายปกติ การตรวจคลื่น ไฟฟ้าหัวใจในกรณีเช่นนี้ได้ประโยชน์น้อยมาก ท่านอาจตรวจเพียงครั้งเดียวเพื่อเก็บไว้เป็น record สำหรับเปรียบเทียบกับการตรวจในอนาคตที่อาจจะเกิดขึ้นเนื่องจากเกิดโรคหัวใจ เน้นอีกครั้ง คลื่นไฟฟ้า หัวใจปกติวันนี้ ไม่ได้หมายความว่าไม่มีโรคหัวใจซ่อนอยู่ และ ไม่ได้หมายความว่าจะไม่เป็น โรคหัวใจในอนาคตในกรณีที่อายุมาก เช่นมากกว่า 40 ปี และมีปัจจัยเสี่ยงต่อโรคหัวใจขาดเลือด เช่น ไขมันในเลือดสูง สูบบุหรี่ เบาหวาน ควรได้รับการตรวจคลื่นไฟฟ้าหัวใจ อย่างน้อย 1 ครั้ง แม้ว่าจะไม่มีอาการของ โรคหัวใจก็ตาม เนื่องจากอยู่ในกลุ่มมีความเสี่ยงสูง อาจเคยมีกล้ามเนื้อหัวใจตาย โดยที่ไม่มี อาการก็ได้ และเป็นการตรวจเพื่อเป็น record ไว้เปรียบเทียบการเปลี่ยนแปลงในอนาคตถ้ามีอาการผิดปกติ เช่น ใจสั่น เจ็บหน้าอก เหนื่อยง่ายผิดปกติ หรือ ตรวจร่างกายผิดปกติ เช่น ความดันโลหิตสูง ลิ้นหัวใจรั่ว กรณีดังกล่าวท่านควรตรวจคลื่นไฟฟ้าหัวใจ แต่ในหลายๆครั้ง ที่จำเป็น ต้องตรวจพิเศษเพิ่มเติม เช่น ตรวจคลื่นไฟฟ้าหัวใจขณะออกกำลัง ติดเครื่องบันทึกคลื่นไฟฟ้า หัวใจ 24-48 ชั่วโมง เป็นต้น กลุ่มนี้อาจต้องได้รับการตรวจหลายครั้ง จนบางครั้งผู้ป่วยบ่นว่าตรวจแล้วตรวจอีกจะเห็นว่า จริงๆแล้วการตรวจและการแปลผลคลื่นไฟฟ้าหัวใจนั้นมีข้อจำกัดมากผู้ป่วยจำนวนส่วนหนึ่งอาจไม่ได้ประโยชน์จากการตรวจนี้ในขณะที่ค่าตรวจตั้งแต่100 ถึง 350 บาท นับว่าแพงมาก

2.6 พื้นฐานการบีบอัดข้อมูล

2.6.1 การบีบอัดแบบไม่สูญหาย (Lossless)

การบีบอัดข้อมูลประเภทไม่มีการสูญเสีย มีคุณสมบัติสำคัญคือ ข้อมูลที่ผ่านการบีบอัดแล้วสามารถนำมาใช้ในการสร้างข้อมูลต้นฉบับกลับคืนมาได้อย่างครบถ้วนสมบูรณ์ดังเดิมทุกประการ การบีบอัดชนิดนี้มีบทบาทสำคัญอย่างมากกับการประยุกต์ใช้งานที่ต้องการความถูกต้องแม่นยำ ไฟล์ข้อมูลคอมพิวเตอร์ เป็นตัวอย่างของข้อมูลที่เมื่อผ่านกระบวนการบีบอัดแล้ว ต้องแน่ใจว่าไม่มีการสูญเสียเกิดขึ้น เนื่องจากการสูญเสียข้อมูลในไฟล์เหล่านี้แม้เพียงเล็กน้อย ข้อมูลที่บรรจุในไฟล์ดังกล่าวจะไม่มีประโยชน์หรือใช้งานไม่ได้อีกต่อไป ตัวอย่างของโปรแกรมที่มีการใช้งานอยู่แพร่หลายสำหรับใช้บีบอัดไฟล์ข้อมูลบนคอมพิวเตอร์โดยไม่มีการสูญเสีย ได้แก่ PKZIP, WinZIP, bzip2, gzip, compress, stuffit, RAR และ Lempel ZIP เป็นต้น ไฟล์ข้อมูลภาพที่ใช้งานทางการแพทย์ ก็เป็นอีกหนึ่งตัวอย่างที่ต้องใช้วิธีการบีบอัดที่ไม่มีการสูญเสีย เพราะความถูกต้องของข้อมูลในภาพมีความสำคัญเป็นอันดับแรก การวินิจฉัยโรคของแพทย์จะต้องอาศัยข้อมูลภาพ เช่น ภาพเอ็กซเรย์ ภาพอัลตราซาวนด์ หรือภาพถ่ายที่มีความถูกต้องละเอียดและแม่นยำที่สุด เป็นต้น

โดยปกติการบีบอัดที่ไม่มีการสูญเสียไม่สามารถรับประกันได้ว่าข้อมูลที่ผ่านกระบวนการบีบอัดแล้วจะมีขนาดเล็กลง ทั้งนี้ประสิทธิภาพของการบีบอัดขึ้นอยู่กับคุณลักษณะของข้อมูลและกรรมวิธีการบีบอัดที่เลือกใช้ ยกตัวอย่างเช่น ถ้าต้องการบีบอัดข้อมูลต่อไปนี้ SSSSSSSSLLLLL หากเลือกใช้วิธีการบีบอัดโดยการนับจำนวนอักขระที่ต่อเนื่องกัน (run length) จะได้ผลการบีบอัดเป็น 8S5L ซึ่งจัดได้ว่าเป็นการบีบอัดที่มีประสิทธิภาพดีมาก แต่หากข้อมูลมีการสับเปลี่ยนลำดับไปเป็น LSSLSLSS LSLSS การใช้วิธีการบีบอัดแบบเดิมจะได้ผลเป็น 1L2S1L1S1L2S1L1S1L2S ซึ่งมีขนาดใหญ่ขึ้นกว่าข้อมูลต้นฉบับ ฉะนั้น การพัฒนากรรมวิธีการบีบอัดที่ไม่มีการสูญเสียจึงมีความน่าสนใจมาก และต้องได้รับการวิจัยอย่างละเอียดเพื่อให้ได้เทคนิคที่มีประสิทธิภาพเหมาะสมกับข้อมูล

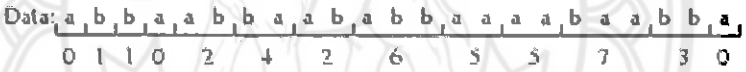
2.6.2 การบีบอัดแบบสูญหาย (Lossy)

คุณลักษณะหลักของการบีบอัดข้อมูลชนิดที่มีการสูญเสียคือ การยอมให้มีการสูญเสียของข้อมูลต้นฉบับไปในระหว่างกระบวนการบีบอัด ซึ่งหมายความว่า ระบบไม่สามารถสร้างข้อมูลต้นฉบับกลับคืนมาให้เหมือนเดิมได้อีก อย่างไรก็ดี การบีบอัดประเภทนี้มีจุดเด่นตรงที่สามารถบีบอัดได้อย่างมีประสิทธิภาพมาก กล่าวคือ หลังการบีบอัดจะได้เป็นข้อมูลที่มีขนาดเล็กลงอย่างมาก เมื่อเทียบกับขนาดเดิมของข้อมูลต้นฉบับ ด้วยเหตุนี้ จึงมีการนำการบีบอัดประเภทนี้มาประยุกต์ใช้งานอย่างกว้างขวางกับการประยุกต์ใช้งานที่ต้องการอัตราส่วนการบีบอัด (Compression ratio) ที่ดีมากๆ โดยที่คุณภาพของข้อมูลหลังการบีบอัดส่งผลต่อความรู้สึกหรือการรับรู้ของมนุษย์ไม่มากนัก หรือยังอยู่ในระดับที่ยอมรับได้ ยกตัวอย่างเช่น สัญญาณเสียง (Voice Signal) ข้อมูลภาพ (Images) หรือสัญญาณวีดีโอ (Video Signal) เป็นต้น ในกรณีการบีบอัดสัญญาณเสียงด้วยวิธี RPELTP(Regular Pulse Excited-Long Term Prediction) ที่ใช้ในระบบโทรศัพท์เคลื่อนที่ GSM (Global System for Mobile Communications) มีอัตราบิตที่จัดว่า

ต่ำมากเพียงประมาณ 13 kbps หากแต่คุณภาพของเสียงที่ได้แตกต่างและค่อนข้างเลวกว่าสัญญาณต้นฉบับตัวจริงอย่างเห็นได้ชัด ถึงกระนั้น การใช้งานโทรศัพท์เคลื่อนที่ GSM กลับมีปริมาณเพิ่มขึ้นอย่างรวดเร็วทั่วโลก การบีบอัดข้อมูลภาพตามมาตรฐาน JPEG (Joint Photographic Experts Group) เป็นตัวอย่างของการบีบอัดภาพที่ดีใกล้เคียงกับภาพต้นฉบับมาก แต่มีขนาดของไฟล์ที่เล็กกว่ามาก ทำให้สามารถจัดเก็บภาพในหน่วยความจำหรือส่งผ่านช่องสัญญาณสื่อสารได้อย่างมีประสิทธิภาพ

2.7 การบีบอัดข้อมูลโดยใช้เทคนิค Lempel-Ziv,Welch

การบีบอัดข้อมูลโดยใช้เทคนิค Lempel-Ziv,Welch นั้นเริ่มขึ้นเมื่อปี 1977 โดยนักวิจัยชาวอิสราเอล Abraham Lempel และ Jacob Ziv และต่อมา Terry Welch ได้นำมาแก้ไขปรับปรุงระบบอัลกอริทึมให้ดีขึ้น จนในที่สุดจึงเรียกว่าวิธีการบีบอัดข้อมูลแบบ Lempel-Ziv,Welch (LZW) ซึ่งอธิบายได้จากการนำตัวอักษรที่เรียงเป็นบรรทัดกันอยู่ นำมาเรียงข้อมูลใส่ dictionary ซึ่งจะทำให้จากอักษรแต่ละตัวจะถูกแทนที่ด้วยจำนวนบิตไปเรื่อยๆ เมื่อตัวอักษรนั้นไม่ได้ซ้ำกัน เมื่อเรียงข้อมูลไปจนเจอตัวอักษรที่ซ้ำก็จะกลับมาดูที่ตารางที่บันทึกบิตข้างต้นไว้แล้ว ดังนั้น code ที่ได้จะเป็นแบบ variable-to-fixed length code



Dictionary			
Index	Entry	Index	Entry
0	a	7	b a a
1	b	8	a b a
2	a b	9	a b b a
3	b b	10	a a a
4	b a	11	a a b
5	a a	12	b a a b
6	a b b	13	b b a

รูปที่ 2.2 วิธีการบีบอัดแบบ Lempel Ziv

ต่อไปจะเป็นการแยกอธิบายออกเป็นสองส่วน ได้แก่ ภาคการเข้ารหัส และภาคการถอดรหัส โดยมีข้อมูลดังนี้ l l l w k l s a v l l w k s k s k s m m m m m x

2.7.1 การเข้ารหัส LZW

ในการเข้ารหัสภาคส่งจะต้องมีการสร้างพจนานุกรมเริ่มต้นที่ประกอบด้วยอักขระที่เป็นไปได้ทั้งหมด ในตัวอย่างนี้มีทั้งสิ้น 8 ตัว คือ {a,k,l,m,s,v,w,x} ดังแสดงในตารางที่ 2.1

ตารางที่ 2.1 รายละเอียดของพจนานุกรมเริ่มต้นสำหรับการเข้ารหัส LZW

ดัชนี index	ข้อมูล entry
1	a
2	k
3	l
4	m
5	s
6	v
7	w
8	x

การเข้ารหัสมีขั้นตอนดังนี้ พิจารณาอักขระตัวแรกคือ l ตรวจสอบดูว่าเป็น entry หนึ่งในพจนานุกรมหรือไม่ พบว่า l เป็น entry ที่มีดัชนีเท่ากับ 3 นำอักขระตัวถัดมาคือ l มาต่อท้ายได้เป็นเลขคู่อักขระ ll เมื่อตรวจสอบว่าคู่อักขระนี้เป็น entry ในพจนานุกรมหรือไม่ พบว่าไม่ปรากฏในพจนานุกรม ในกรณีนี้ให้บรรจุอักขระนี้เพิ่มเติมเข้าไปในพจนานุกรมโดยกำหนดเป็นดัชนีเท่ากับ 9 จากนั้นสร้างคำรหัส <3> เพื่อระบุว่าเป็นอักขระ l

พิจารณาอักขระตัวที่สองคือ l ตรวจดูว่าอักขระนี้เป็น entry หนึ่งในพจนานุกรมหรือไม่ พบว่า l เป็น entry ที่มีดัชนีเท่ากับ 3 นำอักขระตัวถัดมาคือ l มาต่อท้ายได้เป็นคู่อักขระ ll ตรวจสอบว่าคู่อักขระนี้เป็น entry ในพจนานุกรมหรือไม่ พบว่าตรงกับ entry ที่มีดัชนีเป็น 9 นำอักขระตัวถัดมาคือ w มาต่อท้ายได้เป็น llw ตรวจสอบว่าอักขระที่ต่อกันสามตัวนี้เป็น entry ในพจนานุกรมหรือไม่ พบว่าไม่ปรากฏในพจนานุกรม ในกรณีเช่นนนี้ให้บรรจุอักขระที่ต่อกันสามตัวนี้เพิ่มเข้าไปในพจนานุกรมโดยกำหนดให้ใช้ดัชนีที่มีค่าเท่ากับ 10 จากนั้นสร้างคำรหัส <9> เพื่อระบุว่าเป็นอักขระ ll

พิจารณาอักขระตัวที่สี่คือ w ตรวจดูว่าอักขระนี้เป็น entry หนึ่งในพจนานุกรมหรือไม่ พบว่า w เป็น entry ที่มีดัชนีเท่ากับ 7 นำอักขระตัวถัดมาคือ k มาต่อท้ายได้เป็นคู่อักขระ wk ตรวจสอบว่าคู่

อักขระนี้เป็น entry หรือไม่ พบว่าไม่ปรากฏในพจนานุกรม ในกรณีนี้ให้บรรจุคู่อักขระนี้เพิ่มเติมเข้าไปในพจนานุกรมโดยกำหนดเป็นดัชนีเท่ากับ 11 จากนั้นสร้างคำรหัส <7> เพื่อระบุว่าเป็นอักขระ w

พิจารณาอักขระตัวที่ห้าคือ k ตรวจดูว่าอักขระนี้เป็น entry หนึ่งในพจนานุกรมหรือไม่ พบว่า k เป็น entry ที่มีดัชนีเท่ากับ 2 นำอักขระตัวถัดมาคือ l มาต่อท้ายได้เป็นคู่อักขระ kl ตรวจสอบว่าคู่อักขระนี้เป็น entry หรือไม่ พบว่าไม่ปรากฏในพจนานุกรม ในกรณีนี้ให้บรรจุคู่อักขระนี้เพิ่มเติมเข้าไปในพจนานุกรมโดยกำหนดเป็นดัชนีเท่ากับ 12 จากนั้นสร้างคำรหัส <2> เพื่อระบุว่าเป็นอักขระ k

พิจารณาอักขระตัวที่หกคือ l ตรวจดูว่าอักขระนี้เป็น entry หนึ่งในพจนานุกรมหรือไม่ พบว่า l เป็น entry ที่มีดัชนีเท่ากับ 3 นำอักขระตัวถัดมาคือ s มาต่อท้ายได้เป็นคู่อักขระ ls ตรวจสอบว่าคู่อักขระนี้เป็น entry หรือไม่ พบว่าไม่ปรากฏในพจนานุกรม ในกรณีนี้ให้บรรจุคู่อักขระนี้เพิ่มเติมเข้าไปในพจนานุกรมโดยกำหนดเป็นดัชนีเท่ากับ 13 จากนั้นสร้างคำรหัส <3> เพื่อระบุว่าเป็นอักขระ l

พิจารณาอักขระตัวที่เจ็ดคือ s ตรวจดูว่าอักขระนี้เป็น entry หนึ่งในพจนานุกรมหรือไม่ พบว่า s เป็น entry ที่มีดัชนีเท่ากับ 5 นำอักขระตัวถัดมาคือ a มาต่อท้ายได้เป็นคู่อักขระ sa ตรวจสอบว่าคู่อักขระนี้เป็น entry หรือไม่ พบว่าไม่ปรากฏในพจนานุกรม ในกรณีนี้ให้บรรจุคู่อักขระนี้เพิ่มเติมเข้าไปในพจนานุกรมโดยกำหนดเป็นดัชนีเท่ากับ 14 จากนั้นสร้างคำรหัส <5> เพื่อระบุว่าเป็นอักขระ s

พิจารณาอักขระตัวที่แปดคือ a ตรวจดูว่าอักขระนี้เป็น entry หนึ่งในพจนานุกรมหรือไม่ พบว่า a เป็น entry ที่มีดัชนีเท่ากับ 1 นำอักขระตัวถัดมาคือ v มาต่อท้ายได้เป็นคู่อักขระ av ตรวจสอบว่าคู่อักขระนี้เป็น entry หรือไม่ พบว่าไม่ปรากฏในพจนานุกรม ในกรณีนี้ให้บรรจุคู่อักขระนี้เพิ่มเติมเข้าไปในพจนานุกรมโดยกำหนดเป็นดัชนีเท่ากับ 15 จากนั้นสร้างคำรหัส <1> เพื่อระบุว่าเป็นอักขระ a

พิจารณาอักขระตัวที่เก้าคือ v ตรวจดูว่าอักขระนี้เป็น entry หนึ่งในพจนานุกรมหรือไม่ พบว่า v เป็น entry ที่มีดัชนีเท่ากับ 6 นำอักขระตัวถัดมาคือ l มาต่อท้ายได้เป็นคู่อักขระ vl ตรวจสอบว่าคู่อักขระนี้เป็น entry หรือไม่ พบว่าไม่ปรากฏในพจนานุกรม ในกรณีนี้ให้บรรจุคู่อักขระนี้เพิ่มเติมเข้าไปในพจนานุกรมโดยกำหนดเป็นดัชนีเท่ากับ 16 จากนั้นสร้างคำรหัส <6> เพื่อระบุว่าเป็นอักขระ v

พิจารณาอักขระตัวที่สิบคือ l ตรวจดูว่าอักขระนี้เป็น entry หนึ่งในพจนานุกรมหรือไม่ พบว่า l เป็น entry ที่มีดัชนีเท่ากับ 3 นำอักขระตัวถัดมาคือ l มาต่อท้ายได้เป็นคู่อักขระ ll ตรวจสอบว่าคู่อักขระนี้เป็น entry หรือไม่ พบว่าตรงกับ entry ที่มีดัชนีเป็น 9 นำอักขระตัวถัดมาคือ w มาต่อท้ายจะได้เป็น llw ตรวจสอบว่าอักขระที่ต่อกันสามตัวนี้เป็น entry ในพจนานุกรมหรือไม่ พบว่าตรงกับ entry ที่มีดัชนีเป็น 10 นำอักขระตัวถัดมาคือ k มาต่อท้ายได้เป็น llwk ตรวจสอบว่าอักขระที่ต่อกันสี่ตัวนี้เป็น entry ในพจนานุกรมหรือไม่ พบว่าไม่ปรากฏในพจนานุกรม ในกรณีนี้ให้บรรจุอักขระที่ต่อกันสี่ตัวนี้เพิ่มเติมเข้าไปในพจนานุกรมโดยกำหนดเป็นดัชนีเท่ากับ 17 จากนั้นสร้างคำรหัส <10> เพื่อระบุว่าเป็นอักขระ llw

พิจารณาอักขระตัวที่สิบสามคือ k ตรวจดูว่าอักขระนี้เป็น entry หนึ่งในพจนานุกรมหรือไม่ พบว่า k เป็น entry ที่มีดัชนีเท่ากับ 2 นำอักขระตัวถัดมาคือ s มาต่อท้ายได้เป็นคู่อักขระ ks ตรวจสอบว่า

คู่อักขระนี้เป็น entry หรือไม่ พบว่าไม่ปรากฏในพจนานุกรม ในกรณีนี้ให้บรรจุคู่อักขระนี้เพิ่มเติมเข้าไปในพจนานุกรมโดยกำหนดเป็นดัชนีเท่ากับ 18 จากนั้นสร้างคำรหัส <2> เพื่อระบุว่าเป็นอักขระ k

พิจารณาอักขระตัวที่สิบสี่คือ s ตรวจดูว่าอักขระนี้เป็น entry หนึ่งในพจนานุกรมหรือไม่ พบว่า s เป็น entry ที่มีดัชนีเท่ากับ 5 นำอักขระตัวถัดมาคือ k มาต่อท้ายได้เป็นคู่อักขระ sk ตรวจสอบว่าคู่อักขระนี้เป็น entry หรือไม่ พบว่าไม่ปรากฏในพจนานุกรม ในกรณีนี้ให้บรรจุคู่อักขระนี้เพิ่มเติมเข้าไปในพจนานุกรมโดยกำหนดเป็นดัชนีเท่ากับ 19 จากนั้นสร้างคำรหัส <5> เพื่อระบุว่าเป็นอักขระ s

พิจารณาอักขระตัวที่สิบห้าคือ k ตรวจดูว่าอักขระนี้เป็น entry หนึ่งในพจนานุกรมหรือไม่ พบว่า k เป็น entry ที่มีดัชนีเท่ากับ 2 นำอักขระตัวถัดมาคือ s มาต่อท้ายได้เป็นคู่อักขระ ks ตรวจสอบว่าคู่อักขระนี้เป็น entry หรือไม่ พบว่าตรงกับ entry ที่มีดัชนีเป็น 18 นำอักขระตัวถัดมาคือ k มาต่อท้ายจะได้เป็น ksk ตรวจสอบว่าอักขระที่ต่อกันสามตัวนี้เป็น entry ในพจนานุกรมหรือไม่ พบว่าไม่ปรากฏในพจนานุกรม ในกรณีนี้ให้บรรจุอักขระที่ต่อกันสามตัวนี้เพิ่มเติมเข้าไปในพจนานุกรมโดยกำหนดเป็นดัชนีเท่ากับ 20 จากนั้นสร้างคำรหัส <18> เพื่อระบุว่าเป็นอักขระ ks

ถึงขั้นตอนนี้เราได้ทำการเข้ารหัสสัญลักษณ์ข้อมูลไปแล้วทั้งสิ้นจำนวน 16 ตัว ได้แก่ l l l w k l s a v l l w k s k s โดยได้คำรหัสคือ 3 9 7 2 3 5 1 6 10 2 5 18 และพจนานุกรมของภาคเข้ารหัสมายละเอียดดังแสดงในตารางที่ 2.2

ตารางที่ 2.2 รายละเอียดของพจนานุกรมหลังจากเข้ารหัสสัญลักษณ์ข้อมูลไปแล้ว 16 ตัว ได้แก่ l l l w k l s a v l l w k s k s

ดัชนี Index	ข้อมูล entry	ดัชนี index	ข้อมูล entry
1	a	11	wk
2	k	12	kl
3	l	13	ls
4	m	14	sa
5	s	15	av
6	v	16	vl
7	w	17	llwk
8	x	18	ks
9	ll	19	sk
10	llw	20	ksk

สำหรับการเข้ารหัสสัญลักษณ์ข้อมูลที่เหลืออยู่ 8 ตัวประกอบด้วย k s m m m m m x มีรายละเอียดดังนี้

พิจารณาอักขระตัวแรกของส่วนที่เหลืออยู่คือ k ตรวจดูว่าอักขระนี้เป็น entry หนึ่งในพจนานุกรมหรือไม่ พบว่า k เป็น entry ที่มีดัชนีเท่ากับ 2 นำอักขระตัวถัดมาคือ s มาต่อท้ายได้เป็นคู่อักขระ ks ตรวจสอบว่าคู่อักขระนี้เป็น entry หรือไม่ พบว่าตรงกับ entry ที่มีดัชนีเป็น 18 นำอักขระตัวถัดมาคือ m มาต่อท้ายจะได้เป็น ksm ตรวจสอบว่าอักขระที่ต่อกันสามตัวนี้เป็น entry ในพจนานุกรมหรือไม่ พบว่าไม่ปรากฏในพจนานุกรม ในกรณีนี้ให้บรรจุอักขระที่ต่อกันสามตัวนี้เพิ่มเติมเข้าไปในพจนานุกรมโดยกำหนดเป็นดัชนีเท่ากับ 21 จากนั้นสร้างคำรหัส <18> เพื่อระบุว่าเป็นอักขระ ks

พิจารณาอักขระตัวที่สามคือ m ตรวจดูว่าอักขระนี้เป็น entry หนึ่งในพจนานุกรมหรือไม่ พบว่า m เป็น entry ที่มีดัชนีเท่ากับ 4 นำอักขระตัวถัดมาคือ m มาต่อท้ายได้เป็นคู่อักขระ mm ตรวจสอบว่าคู่อักขระนี้เป็น entry หรือไม่ พบว่าไม่ปรากฏในพจนานุกรม ในกรณีนี้ให้บรรจุคู่อักขระนี้เพิ่มเติมเข้าไปในพจนานุกรมโดยกำหนดเป็นดัชนีเท่ากับ 22 จากนั้นสร้างคำรหัส <4> เพื่อระบุว่าเป็นอักขระ m

พิจารณาอักขระตัวที่สี่คือ m ตรวจดูว่าอักขระนี้เป็น entry หนึ่งในพจนานุกรมหรือไม่ พบว่า m เป็น entry ที่มีดัชนีเท่ากับ 4 นำอักขระตัวถัดมาคือ m มาต่อท้ายได้เป็นคู่อักขระ mm ตรวจสอบว่าคู่อักขระนี้เป็น entry หรือไม่ พบว่าตรงกับ entry ที่มีดัชนีเป็น 22 นำอักขระตัวถัดมาคือ m มาต่อท้ายจะได้เป็น mmm ตรวจสอบว่าอักขระที่ต่อกันสามตัวนี้เป็น entry ในพจนานุกรมหรือไม่ พบว่าไม่ปรากฏในพจนานุกรม ในกรณีนี้ให้บรรจุอักขระที่ต่อกันสามตัวนี้เพิ่มเติมเข้าไปในพจนานุกรมโดยกำหนดเป็นดัชนีเท่ากับ 23 จากนั้นสร้างคำรหัส <22> เพื่อระบุว่าเป็นอักขระ mm

พิจารณาอักขระตัวที่หกคือ m ตรวจดูว่าอักขระนี้เป็น entry หนึ่งในพจนานุกรมหรือไม่ พบว่า m เป็น entry ที่มีดัชนีเท่ากับ 4 นำอักขระตัวถัดมาคือ m มาต่อท้ายได้เป็นคู่อักขระ mm ตรวจสอบว่าคู่อักขระนี้เป็น entry หรือไม่ พบว่าตรงกับ entry ที่มีดัชนีเป็น 22 นำอักขระตัวถัดมาคือ x มาต่อท้ายจะได้เป็น mmx ตรวจสอบว่าอักขระที่ต่อกันสามตัวนี้เป็น entry ในพจนานุกรมหรือไม่ พบว่าไม่ปรากฏในพจนานุกรม ในกรณีนี้ให้บรรจุอักขระที่ต่อกันสามตัวนี้เพิ่มเติมเข้าไปในพจนานุกรมโดยกำหนดเป็นดัชนีเท่ากับ 24 จากนั้นสร้างคำรหัส <22> เพื่อระบุว่าเป็นอักขระ mm

พิจารณาอักขระตัวสุดท้ายคือ x ตรวจสอบว่าเป็น entry หนึ่งในพจนานุกรมหรือไม่ พบว่า x เป็น entry ที่มีดัชนีเท่ากับ 7 สร้างคำรหัส <7> เพื่อระบุว่าเป็นอักขระ x โดยไม่ต้องมีการปรับพจนานุกรมอีกต่อไป

ผลการเข้ารหัสทั้งหมดจะได้เป็นคำรหัส 3 9 7 2 3 5 1 6 10 2 5 18 18 4 22 22 7 และพจนานุกรมของภาคเข้ารหัสมีรายละเอียดดังแสดงในตารางที่ 2.3

ตารางที่ 2.3 รายละเอียดของพจนานุกรมหลังจากเข้ารหัสสัญลักษณ์ข้อมูลจนครบทั้ง 24 ตัว

ดัชนี Index	ข้อมูล entry	ดัชนี index	ข้อมูล entry	ดัชนี index	ข้อมูล entry
1	a	11	wk	21	ksm
2	k	12	kl	22	mm
3	l	13	ls	23	mmm
4	m	14	sa	24	mmx
5	s	15	av	25	
6	v	16	vl	26	
7	w	17	llwk	27	
8	x	18	ks	28	
9	ll	19	sk	29	
10	llw	20	ksk	30	

2.7.2 การถอดรหัส LZW

ในการถอดรหัสภาครับจะต้องมีการสร้างพจนานุกรมเริ่มต้นที่ประกอบด้วยอักขระที่เป็นไปได้ทั้งหมด ในลักษณะเดียวกับที่มีในภาคส่ง ดังแสดงในตารางที่ 2.4 ชุดรหัสที่ประสงค์จะถอดรหัสประกอบด้วย 3 9 7 2 3 5 1 6 10 2 5 18 18 4 22 22 7

ตารางที่ 2.4 รายละเอียดของพจนานุกรมเริ่มต้นสำหรับการเข้ารหัส LZW

ดัชนี index	ข้อมูล entry
1	a
2	k
3	l
4	m
5	s
6	v
7	w
8	x

วิธีการถอดรหัสให้ดำเนินการดังนี้ ดัชนีตัวแรกคือ 3 เมื่อดูจากตารางพบว่าใช้แทนอักขระ l ฉะนั้นอักขระตัวแรกที่ถอดได้คือ l ในขณะเดียวกันก็จะมีการปรับพจนานุกรมใหม่เพื่อเพิ่มข้อมูล entry

ที่มีค่าดัชนีเท่ากับ 9 โดยอาศัยวิธีการเดียวกับที่กระทำ ณ ภาคเข้ารหัส อย่างไรก็ดี ณ ตอนนี้ เราทราบเพียงว่าข้อมูล entry ใหม่จะต้องประกอบด้วยอักขระที่ต่อกันสองตัว โดยตัวแรกต้องเป็น 1 อย่างแน่นอน การที่จะทราบค่าอักขระตัวที่สองจำเป็นต้องอ่านคำรหัสตัวถัดมาก่อน ซึ่งในตัวอย่างนี้คือ 9 ปัญหาคือ ณ จังหวะเวลานี้เรายังไม่ทราบข้อมูล entry ของ 9 อย่างสมบูรณ์ แต่กระนั้นข้อมูลที่เราสนใจมีเพียงอักขระตัวแรกของดัชนี 9 คือ ll ถึงตรงนี้พจนานุกรมจะมีลักษณะดังแสดงในตารางที่ 2.5 ฉะนั้น หลังจากถอดรหัสคำรหัสสองตัวแรกคือ 3 9 แล้ว ได้เป็นอักขระ l l l

ตารางที่ 2.5 พจนานุกรมของการเข้ารหัส LZW หลังจากถอดคำรหัสสองตัวแรก ได้แก่ 3 9

ดัชนี index	ข้อมูล entry
1	a
2	k
3	l
4	m
5	s
6	v
7	w
8	x
9	ll

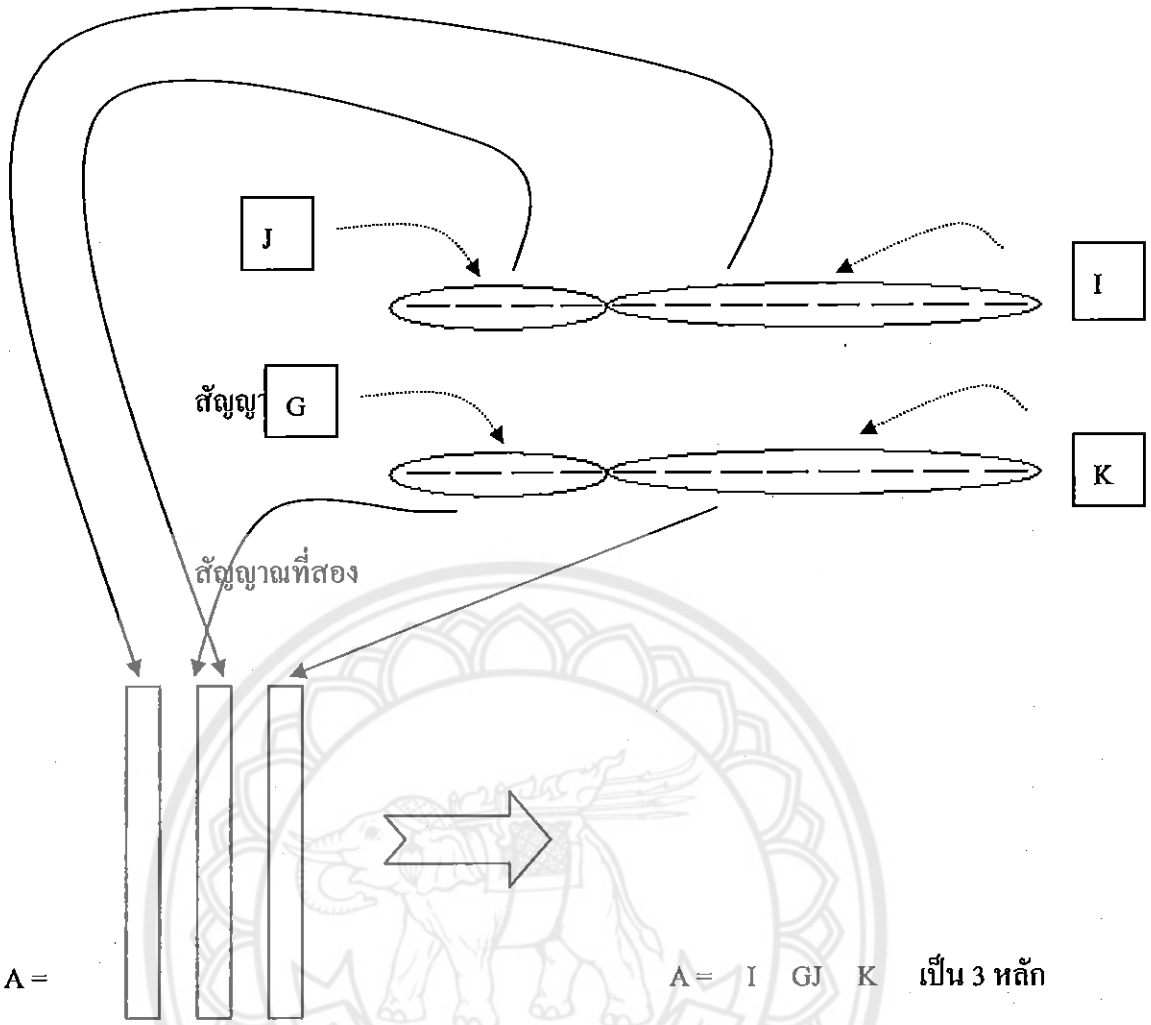
ทำการถอดรหัสตัวที่สามคือ 7 ซึ่งจากตารางใช้แทนอักขระ w การปรับพจนานุกรม สามารถทำได้โดยง่ายคือ เพิ่มดัชนีหมายเลข 10 โดยใช้แทนอักขระสามตัวติดกันคือ llw ถึงตรงนี้เราได้ผลการถอดรหัสเป็น l l l w ทำการถอดรหัสตัวที่สี่คือ 2 เนื่องจากในตารางระบุว่า 2 ใช้แทนอักขระ k ฉะนั้นเราได้ผลการถอดรหัสเป็น l l l w k การปรับพจนานุกรมสามารถทำได้โดยง่ายคือ เพิ่มดัชนีหมายเลข 11 โดยใช้แทนอักขระสองตัวติดกันคือ wk ทำการถอดรหัสตัวที่ห้าต่อกันคือ 3 ซึ่งเป็นดัชนีใช้แทนอักขระ l ฉะนั้น ผลการเข้ารหัสที่ได้ตอนนี้คือ l l l w k l ส่วนการปรับพจนานุกรมก็สามารถทำได้โดยง่ายคือเพิ่มดัชนีหมายเลข 12 โดยใช้แทนอักขระสองตัวติดกันคือ kl ณ เวลานี้ พจนานุกรมจะมีรายละเอียดดังแสดงตารางที่ 2.6 ซึ่งตรงกันกับตารางของภาคการเข้ารหัสทุกประการ สำหรับการถอดคำรหัสที่เหลือสามารถกระทำได้ในลักษณะเดียวกัน

ตารางที่ 2.6 พจนานุกรมของการเข้ารหัส LZW หลังจากถอดคำรหัสห้าตัวแรก ได้แก่ 3 9 7 2 3

ดัชนี Index	ข้อมูล entry	ดัชนี index	ข้อมูล entry
1	a	11	wk
2	k	12	kl
3	l		
4	m		
5	s		
6	v		
7	w		
8	x		
9	ll		
10	llw		

2.8 Format212

มีสัญญาณ ECG 2 สัญญาณ ใช้ 12 บิต ในการเก็บสัญญาณ โดยที่ตอนเราได้วัดสัญญาณ ECG นั้น ค่าของมันจะเป็นหน่วย mV ซึ่งจะมีค่าตั้งแต่ -5 ถึง 5 mV ซึ่งจากนั้นเราต้องทำให้ค่าของมันไม่ติดทศนิยม โดยการคูณค่าตัวหนึ่งเข้าไป (เช่น คูณด้วย 200 เป็นต้น) ซึ่งเรียกค่านี้ว่า gain or number of integers per mV หลังจากนั้นแม้เราคูณค่าของ gain เข้าไปแล้ว แต่จำนวนตัวเลขยังเกิดการติดลบอยู่ เราจะทำการยกระดับค่าเพื่อให้ค่าไม่เกิดการติดลบ(เช่นยกระดับค่าจาก 0 ไป1024) เราจะเรียกการยกระดับค่านี้ว่า zerovalue or integer value of ECG zero point ซึ่งหลังจากนั้น เราจะแบ่ง สัญญาณ 12 บิต 2 สัญญาณ ให้ออกเป็น ข้อมูล 8 บิต 3 ข้อมูล โดยที่สัญญาณแรกนั้น เราจะข้อมูล 8 บิตหลัง มาเป็น 8 บิตแรกของข้อมูล และเอาสัญญาณที่ 2 นั้น เราจะใช้ข้อมูล 8 บิตหลัง มาเป็น 8 บิตสุดท้ายของข้อมูล ส่วน 8 บิตตรงกลางเราจะใช้ 4 บิตแรกที่เหลืออยู่ของทั้ง 2 สัญญาณมาต่อกันโดยที่จะให้ 4 บิตแรกของสัญญาณที่ 2 อยู่ขึ้นหน้าก่อน



รูปที่ 2.3 วิธีการ Format 212

2.9 การหาค่าความผิดพลาด (SNR และPSNR) และการหาค่าอัตราส่วนการบีบอัดข้อมูล

อัตราส่วนของการบีบอัดข้อมูล (Compression Ratio : CR) คือ การเปรียบเทียบค่าระหว่างขนาดของข้อมูลต้นฉบับ (n1) กับข้อมูลที่ถูกบีบอัดแล้ว (n2) ดังสมการที่ 2.1

$$CR = \frac{n1}{n2} \qquad \textbf{2.1}$$

การหาความผิดพลาดของการบีบอัดข้อมูลนั้น สามารถใช้หลักการของ Signal to Noise Ratio (SNR) และ Peak Signal to Noise Ratio (PSNR) โดยจะเป็นตัววัดคุณภาพของการบีบอัดข้อมูลว่า เมื่อบีบอัดข้อมูลแล้วคุณภาพของสัญญาณที่ได้ใหม่จะมีการสูญเสียมากน้อยเพียงไหน ดังสมการที่ 2.2 และสมการที่ 2.3

$$SNR_{(dB)} = 10\log_{10}\left\{\frac{\left[\sum_{n=0}^{N-1}\sum_{m=0}^{M-1}(s(m,n))^2\right]}{\sum_{n=0}^{N-1}\sum_{m=0}^{M-1}\left(s(m,n)-\hat{s}(m,n)\right)^2}\right\} \quad \textbf{2.2}$$

$$PSNR_{(dB)} = 10\log_{10}\left\{\frac{\max|s(m,n)|}{\frac{1}{MN}\sum_{n=0}^{N-1}\sum_{m=0}^{M-1}\left(s(m,n)-\hat{s}(m,n)\right)^2}\right\} \quad \textbf{2.3}$$

กำหนดให้ : ค่า m และ n เป็นค่าของแกนนอนและแกนตั้ง

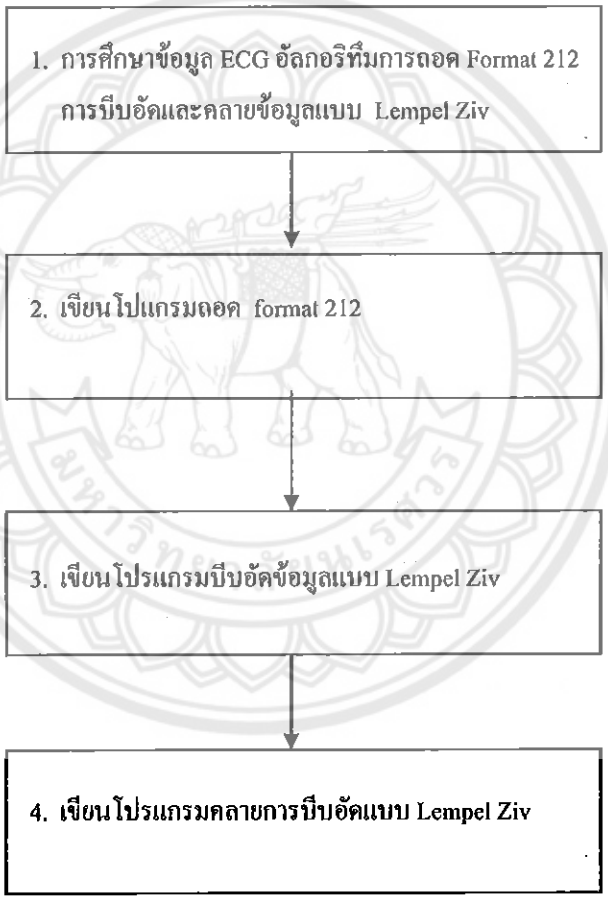
ค่า $s(m,n)$ เป็นข้อมูลของสัญญาณต้นฉบับ

ค่า $\hat{s}(m,n)$ เป็นข้อมูลของสัญญาณที่ถูกบีบอัดแล้ว

ค่า max เป็นค่าที่มากที่สุด

บทที่ 3
วิธีการดำเนินงานโครงงานวิศวกรรม

ในวิธีการบีบอัดข้อมูล ECG โดยวิธี Lempel Ziv ได้แบ่งขั้นตอนการโครงงานออกเป็น 4 ขั้นตอน โดยเริ่มจากการศึกษาข้อมูล ECG อัลกอริทึมการถอด format 212 การบีบอัดและคลายข้อมูลแบบ Lempel Ziv และขั้นตอนการออกแบบ เขียนโปรแกรมสร้างวิธีการบีบอัดแบบ Lempel Ziv เขียนโปรแกรมวิธีการคลายการบีบอัดแบบ Lempel Ziv ดังที่แสดงในรูป 3.1



รูปที่ 3.1 ขั้นตอนการดำเนินโครงงาน

3.1 ศึกษาอัลกอรีทึมการถอด format 212 และขั้นตอนการออกแบบ

จากบทที่ 2 ศึกษาทฤษฎีและอัลกอริทึมที่เกี่ยวข้อง เราสามารถนำทฤษฎีและอัลกอริทึมที่ได้มาออกแบบโปรแกรมการบีบอัดข้อมูลECG โดยวิธี Lempel Ziv ดังในรูป 3.2 และนำมาเขียนโปรแกรมถอด format 212 และเขียนโปรแกรมบีบอัดและคลายการบีบอัดข้อมูลโดยวิธี Lempel Ziv

3.2 วิธีการทำ

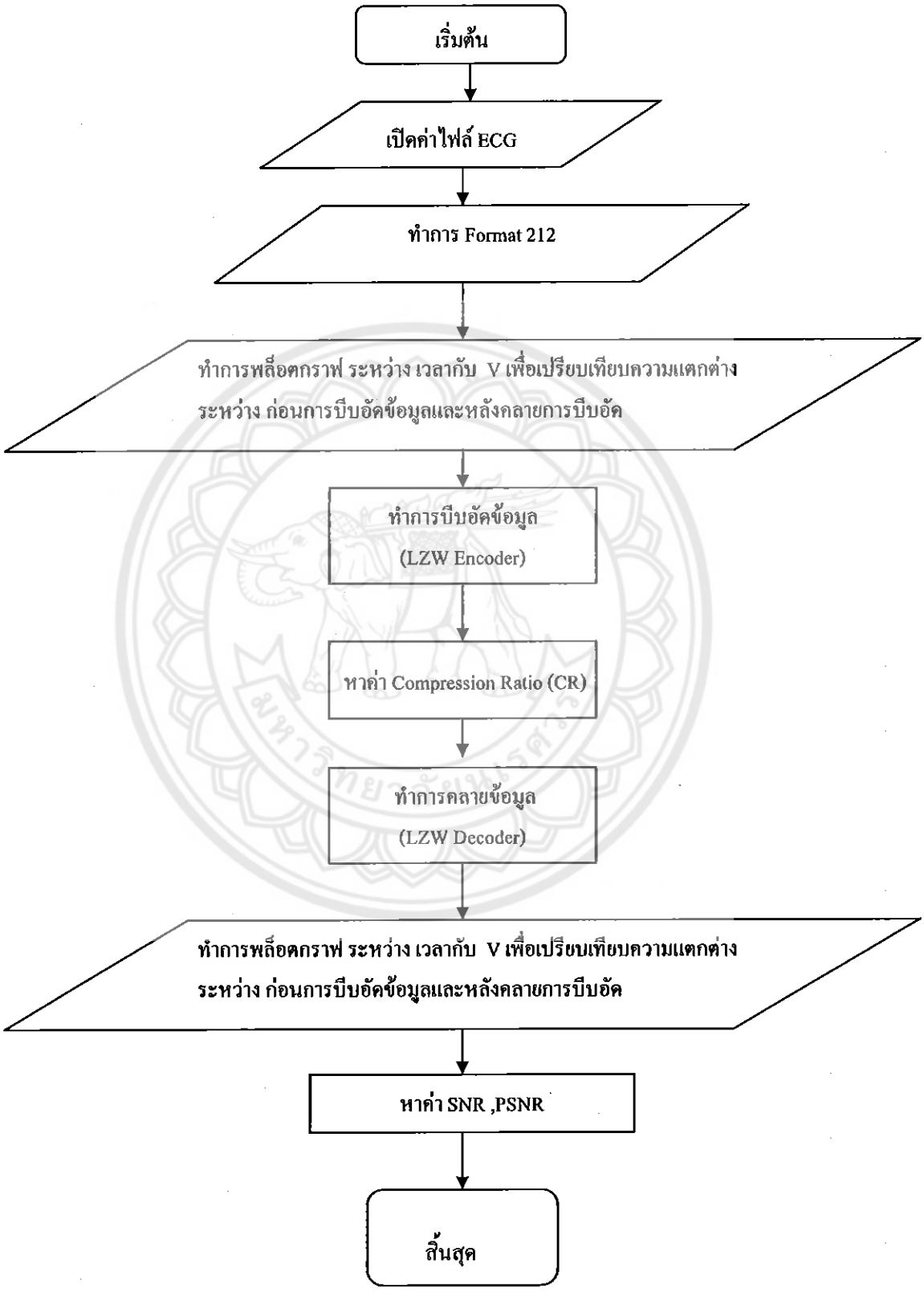
โดยขั้นตอนแรกของการทำคือ เริ่มจากการที่รับข้อมูลจากไฟล์ข้อมูล โดยใช้โปรแกรมแมท-แลปเรียกออกมา เราจะทำการพล็อตกราฟ โดยที่กราฟนี้จะมีหน่วยแกนเอ็กซ์คือเวลาเป็นวินาที ส่วนหน่วยในแกนวายคือ จำนวนโวล จากนั้นไฟล์จะติดอยู่ในรูป format 212 ซึ่งเราต้องถอด format 212 ออกมาก่อน โดยวิธีการถอดได้อธิบายอยู่ในบทที่ 2 พอ ถอด format 212 ออกมาแล้ว ข้อมูลจะอยู่ในรูป 8 บิต 3 หลัก ที่นี้จะต้องทำการลดจำนวนข้อมูล 3 หลักให้เหลือเพียงหลักเดียว โดยการนำหลักที่ 2 มาต่อหลักที่ 1 จากนั้นนำหลักที่ 3 มาต่อหลักที่ 2 แล้วถึงเริ่มทำการบีบอัดข้อมูล สาเหตุที่ต้องทำเช่นนี้เพราะว่าต้องการที่จะลดจำนวนครั้งในการบีบอัดข้อมูลเพราะว่าถ้าใช้ 3 หลัก จะต้องบีบอัดข้อมูลถึง 3 ครั้ง แต่ถ้าลดเหลือเพียงหลักเดียวก็จะสามารถบีบอัดให้เหลือเพียง 1 ครั้ง จากนั้นเราจะทำการบีบอัดข้อมูลตามทฤษฎีในบทที่ 2 จากนั้นเราจึงทำการคลายข้อมูลที่ถูกการบีบอัดมาโดยใช้ทฤษฎีในบทที่ 2 จากนั้นเราจึงทำการใส่ format 212 เข้าไปก่อน จากนั้นเราจะทำการพล็อตกราฟออกมาโดยที่แกนเอ็กซ์คือเวลาเป็นวินาที ส่วนหน่วยในแกนวายนั้นคือ จำนวนโวลต์ เพื่อเปรียบเทียบกันว่า กราฟทั้งสองสัญญาณเหมือนกันหรือไม่โดยที่เราได้ใช้ฟังก์ชั่นในการเรียกดูว่าค่ากราฟที่หลังจากการคลายการบีบอัดออกมามีค่าตรงกันกับข้อมูลชุดดั้งเดิมหรือไม่ ซึ่งค่าข้อมูลทั้งสองก่อนและหลังการคลายข้อมูลต้องหักล้างกันเป็นศูนย์พอดี จากนั้นเราหาคำนวณหาค่า Compression Ratio , SNR , PSNR โดยที่เราจะดูจากทฤษฎีบทที่ 2

3.3 การออกแบบโปรแกรม

จากบทที่ 2 เราสามารถนำทฤษฎีและวิธีการที่ได้ศึกษามา ออกแบบโปรแกรมการบีบอัดข้อมูล ECG โดยใช้วิธี Lempel Ziv ดังจะแสดงดังต่อไปนี้

ส่วนของโปรแกรม โดยจะใช้ Flow Chart อธิบายวิธีการทำงาน (สามารถดูโปรแกรมจากภาคผนวก) โดยที่จะมี Flow Chart แบบภาพรวม , Format 212 , บีบอัดข้อมูลแบบ Lempel Ziv และคลายข้อมูลแบบ Lempel Ziv

3.4 แผนผังการทำงาน (Flow Chart)

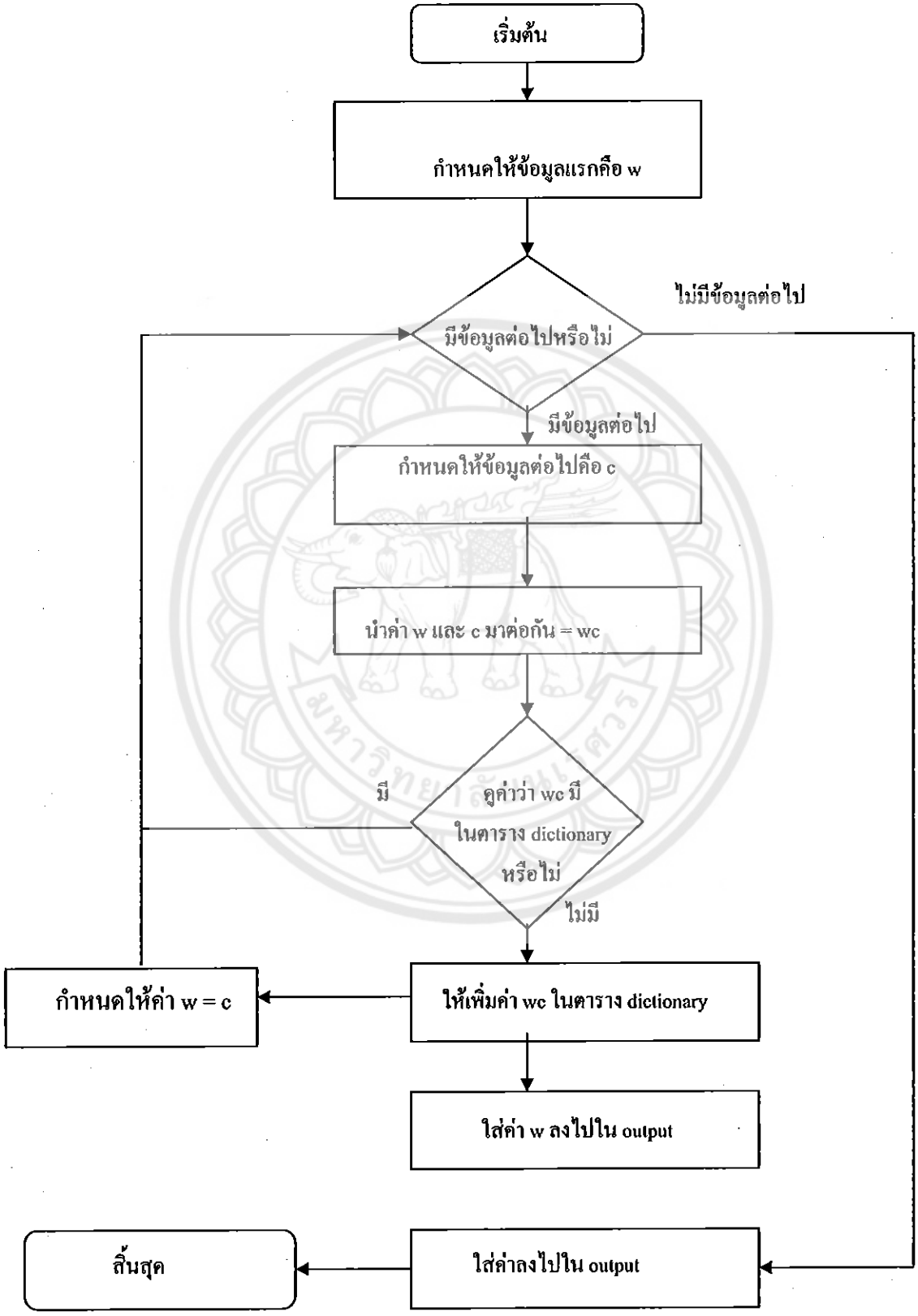


รูปที่ 3.2 แสดงแผนผังการทำงานของวิธีการบีบอัดข้อมูล ECG โดยวิธี Lempel Ziv

3.4.1 Flow Chart ของการ Format 212

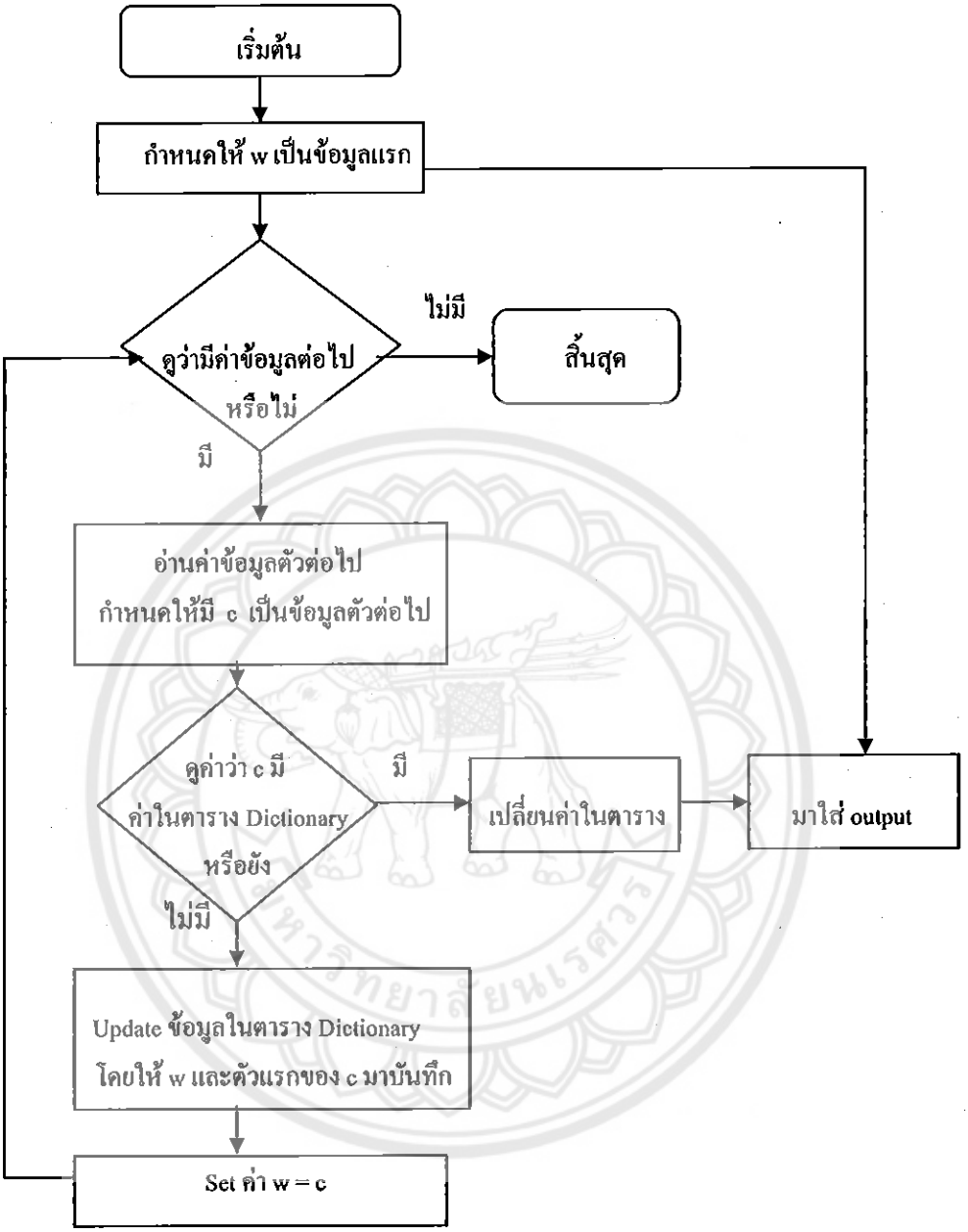
รูปที่ 3.3 แสดงถึงวิธีการ Format 212

3.4.2 Flow Chart ของการ การบีบอัดข้อมูลแบบ Lempel Ziv



รูปที่ 3.4 แสดงถึงวิธีการบีบอัดข้อมูลแบบ Lempel Ziv

3.4.3 Flow Chart ของการคลายข้อมูลแบบ Lempel Ziv

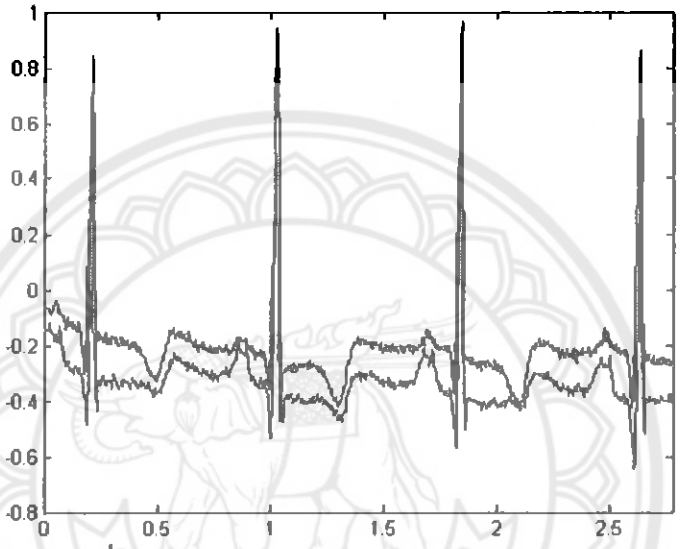


รูปที่ 3.5 แสดงถึงวิธีการคลายบีบอัดข้อมูลแบบ Lempel Ziv

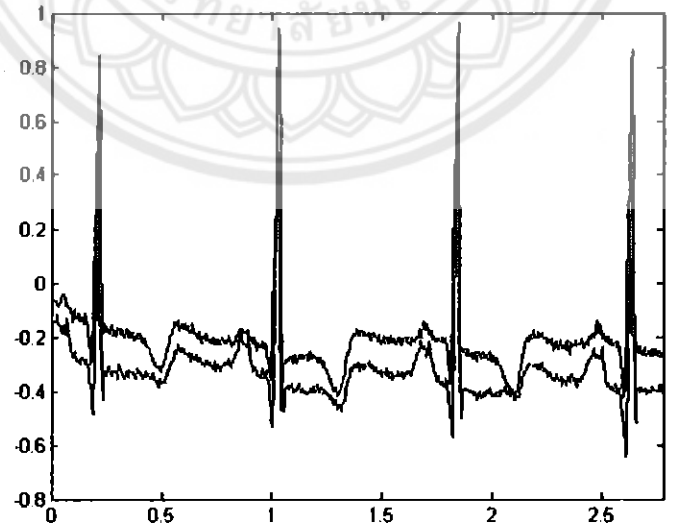
บทที่ 4

ผลการทดลอง

โหลดไฟล์เข้ามาในโปรแกรม โดยที่แปลงไฟล์จะได้ 3 หลัก แต่เราจะรวม 3 หลักเข้าด้วยกัน โดยเอาหลักที่ 2 มาต่อหลักที่ 1 เอาหลักที่ 3 มาต่อหลักที่ 2 แล้วค่อยบีบอัด



รูปที่ 4.1 สัญญาณ ECG ไฟล์ 100 ก่อนการบีบอัด



รูปที่ 4.2 สัญญาณ ECG ไฟล์ 100 หลังการคลายการบีบอัด (CR = 1.6066 , SNR = ∞ และ PSNR = ∞)

คำอธิบายรูป 4.1 คือรูปสัญญาณ ECG ที่ก่อนการบีบอัดข้อมูล ซึ่งเป็นรูป ECG 2 สัญญาณที่ใช้ในการวิจัย แกน y คือ mV กับแกน x คือ เวลา โดยที่เส้นสีน้ำเงินคือสัญญาณ ECG ทั่วไป ที่ไม่มีอาการป่วย เส้นสีแดงคือสัญญาณ ECG ทั่วไป ที่มีอาการป่วยเป็นโรคหวัด

คำอธิบายรูป 4.2 คือรูปสัญญาณ ECG หลังจากการคลายการบีบอัดแล้ว ซึ่งเป็นรูป ECG 2 สัญญาณที่ใช้ในการวิจัย แกน y คือ mV กับแกน x คือ เวลา โดยที่เส้นสีน้ำเงินคือสัญญาณ ECG ทั่วไป ที่ไม่มีอาการป่วย เส้นสีแดงคือสัญญาณ ECG ทั่วไป ที่มีอาการป่วยเป็นโรคหวัด

เราเอาข้อมูล ECG ทั้ง 3 หลัก มาต่อรวมกันหมดเป็นหลักเดียว เพื่อที่จะทำการบีบอัดข้อมูลได้ง่ายขึ้น

ตารางที่ 4.1 ขนาดข้อมูลไฟล์ 100

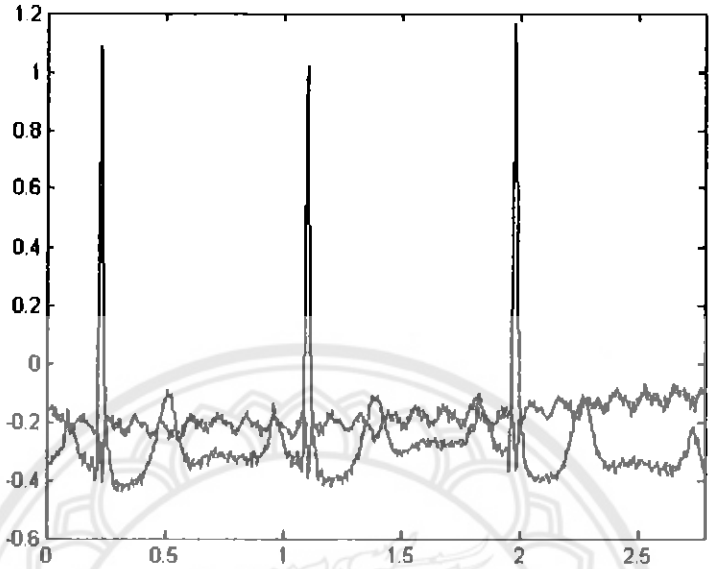
ค่าข้อมูล	ขนาด
ค่าข้อมูลดั้งเดิม	24000 บิต
ค่าข้อมูลตอนบีบอัด	14938 บิต

ความแตกต่างเท่ากับ 9062 บิต

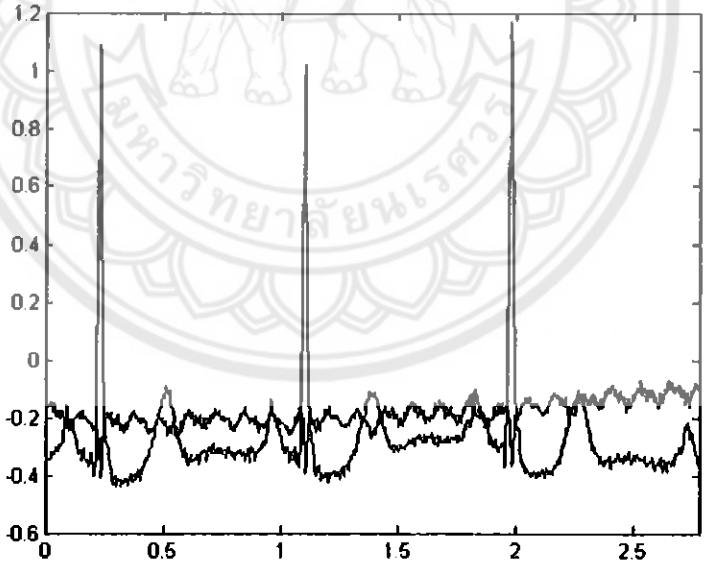
เมื่อเราป้อนคำสั่งลงในช่อง Command ของ matlab max(packed) แล้วกด enter ค่าจะออกมา ค่าที่มากที่สุดของ Packed (ที่ถูกบีบอัด) = 1537

ค่าความแตกต่างระหว่างก่อนการบีบอัดและหลังคลายการบีบอัด โดยอาศัยตัวเลข เมื่อตัวเลขก่อนการบีบอัดและหลังคลายการบีบอัดที่ตำแหน่งเดียวกันลบกันต้องมีค่าเท่ากับ 0 เสมอ ก่อนการบีบอัดใช้ข้อมูล 8 บิต จำนวน 3000 ข้อมูล เท่ากับ 8*3000 = 24000 บิตพอบีบอัดแล้วใช้ข้อมูล 11 บิต เพราะตัวข้อมูลมีค่าสูงสุด 1537 ซึ่งสูงกว่า 8 บิต (255) มีจำนวน 1358 ข้อมูล เท่ากับ 11*1358 = 14938 ข้อมูลจากตอนแรกก่อนการบีบอัดมีข้อมูล 24000 บิต พอบีบอัดแล้วมีข้อมูล 14938 บิตสามารถบีบอัดได้ 24000 – 14938 = 9062 บิตคิดเป็นค่า CR = 24000/14938 = 1.6066แสดงว่าบีบอัดข้อมูลได้จริง ซึ่งเป็นการบีบอัดที่ไม่เกิดการสูญเสีย

เริ่มแรกเมื่อเราเปิดโปรแกรมขึ้นมา โดยกรอกข้อมูลเป็นไฟล์ 101 โดยที่ไฟล์มันจะมี 3 หลัก แล้วเราจะรวม 3 หลักมาเป็นหลักเดียว โดยเอาหลักที่ 2 มาต่อหลักที่ 1 เอาหลักที่ 3 มาต่อหลักที่ 2



รูปที่ 4.3 สัญญาณ ECG ไฟล์ 101 ก่อนการบีบอัด



รูปที่ 4.4 สัญญาณ ECG ไฟล์ 101 หลังการคลายการบีบอัด (CR = 1.6554 , SNR = ∞ และ PSNR = ∞)

คำอธิบายรูป 4.3 คือรูปสัญญาณ ECG ที่ก่อนการบีบอัดข้อมูล ซึ่งเป็นรูป ECG 2 สัญญาณที่ใช้ในการวิจัย แกน y คือ mV กับแกน x คือ เวลา โดยที่เส้นสีแดงคือสัญญาณ ECG ทั่วๆไป ที่มีอาการป่วยของโรคหวัด เส้นสีน้ำเงิน คือสัญญาณ ECG ที่บริเวณข้อเท้าของคนที่เป็นโรคหวัด

คำอธิบายรูป 4.4 คือรูปสัญญาณ ECG หลังจากการคลายการบีบอัดแล้ว ซึ่งเป็นรูป ECG 2 สัญญาณที่ใช้ในการวิจัย แกน y คือ mV กับแกน x คือ เวลา โดยที่เส้นสีแดงคือสัญญาณ ECG ทั่วๆไป ที่มีอาการป่วยของโรคหวัด เส้นสีน้ำเงิน คือสัญญาณ ECG ที่บริเวณข้อเท้าของคนที่เป็นโรคหวัด เราเอาข้อมูล ECG ทั้ง 3 หลัก มาต่อรวมกันหมดเป็นหลักเดียว เพื่อที่จะทำการบีบอัดข้อมูลได้ง่ายขึ้น

ตารางที่ 4.2 ขนาดข้อมูลไฟล์ 101

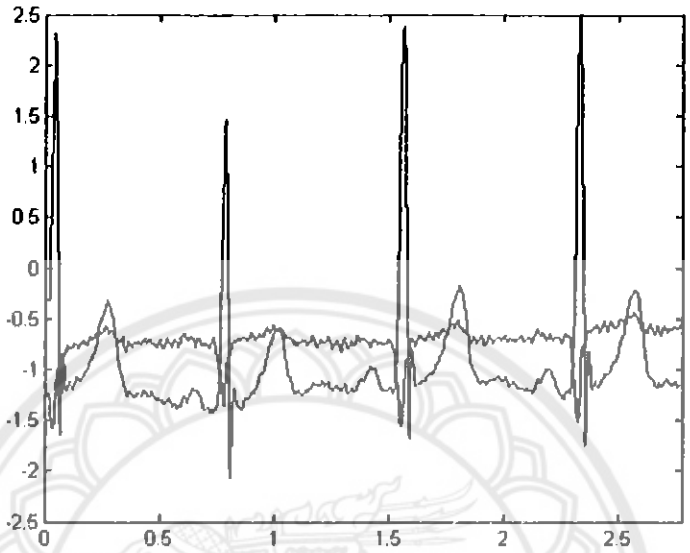
ค่าข้อมูล	ขนาด
ค่าข้อมูลดั้งเดิม	24000 บิต
ค่าข้อมูลตอนบีบอัด	14498 บิต

ความแตกต่าง เท่ากับ 9502 บิต

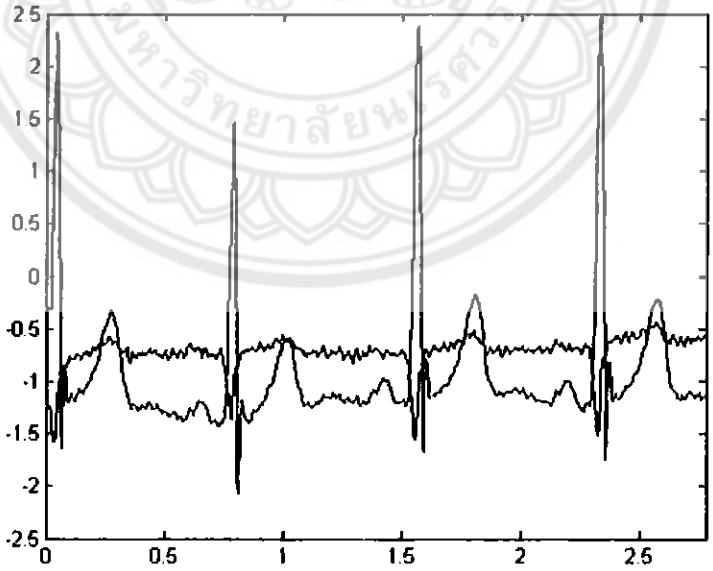
เมื่อเราป้อนคำสั่งลงในช่อง Command ของ matlab max(packed) แล้วกด enter ค่าจะออกมาค่าที่มากที่สุดของ Packed (ที่ถูกบีบอัด) = 1568

ค่าความแตกต่างระหว่างก่อนการบีบอัดและหลังคลายการบีบอัด โดยอาศัยตัวเลข เมื่อตัวเลขก่อนการบีบอัดและหลังคลายการบีบอัดที่ตำแหน่งเดียวกันลบกันต้องมีค่าเท่ากับ 0 เสมอ ก่อนการบีบอัดใช้ข้อมูล 8 บิต จำนวน 3000 ข้อมูล เท่ากับ 8*3000 = 24000 บิตพอบีบอัดแล้วใช้ข้อมูล 11 บิต เพราะตัวข้อมูลมีค่าสูงสุด 1568 ซึ่งสูงกว่า 8 บิต (255) และมีจำนวน 1318 ข้อมูล เท่ากับ 11*1318 = 14498 ข้อมูล จากตอนแรกก่อนการบีบอัดมีข้อมูล 24000 บิต พอบีบอัดแล้วมีข้อมูล 14498 บิต สามารถบีบอัดได้ 24000 – 14498 = 9502 บิต คิดเป็นค่าCR = 24000/14498 = 1.6554 แสดงว่าบีบอัดข้อมูลได้จริงซึ่งเป็นการบีบอัดที่ไม่เกิดการสูญเสีย

เริ่มแรกเมื่อเราเปิดโปรแกรมขึ้นมา โดยกรอกข้อมูลเป็นไฟล์ 116 โดยที่ไฟล์มันจะมี 3 หลัก แล้วเราจะรวม 3 หลักมาเป็นหลักเดียว โดยเอาหลักที่ 2 มาต่อหลักที่ 1 เอาหลักที่ 3 มาต่อหลักที่ 2



รูปที่ 4.5 สัญญาณ ECG ไฟล์ 116 ก่อนการบีบอัด



รูปที่ 4.6 สัญญาณ ECG ไฟล์ 116 หลังการคลายการบีบอัด (CR = 1.3191 , SNR = ∞ และ PSNR =∞)

คำอธิบายรูป 4.3 คือรูปสัญญาณ ECG ที่ก่อนการบีบอัดข้อมูล ซึ่งเป็นรูป ECG 2 สัญญาณที่ใช้ในการวิจัย แกน y คือ mV กับแกน x คือ เวลา โดยที่เส้นสีแดงคือสัญญาณ ECG ทั่วๆไป ที่มีอาการป่วยของโรคหวัด เส้นสีน้ำเงิน คือสัญญาณ ECG ที่บริเวณข้อมือของคนที่เป็นโรคหวัด

คำอธิบายรูป 4.4 คือรูปสัญญาณ ECG หลังจากการคลายการบีบอัดแล้ว ซึ่งเป็นรูป ECG 2 สัญญาณที่ใช้ในการวิจัย แกน y คือ mV กับแกน x คือ เวลา โดยที่เส้นสีแดงคือสัญญาณ ECG ทั่วๆไป ที่มีอาการป่วยของโรคหวัด เส้นสีน้ำเงิน คือสัญญาณ ECG ที่บริเวณข้อมือของคนที่เป็นโรคหวัด เราเอาข้อมูล ECG ทั้ง 3 หลัก มาต่อรวมกันหมดเป็นหลักเดียว เพื่อที่จะทำการบีบอัดข้อมูลได้ง่ายขึ้น

ตารางที่ 4.3 ขนาดข้อมูลไฟล์ 116

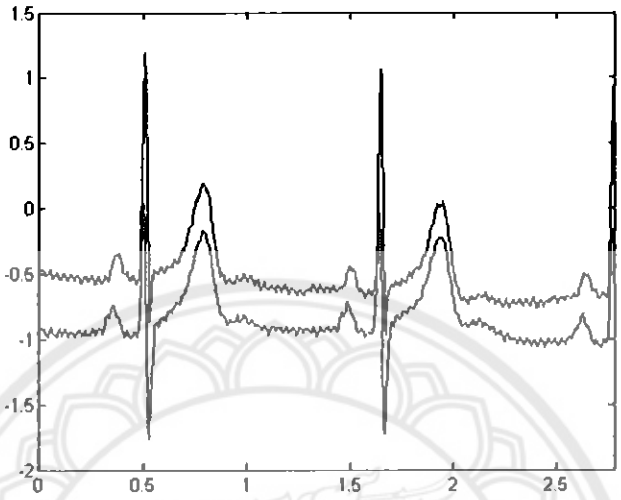
ค่าข้อมูล	ขนาด
ค่าข้อมูลดั้งเดิม	24000 บิต
ค่าข้อมูลตอนบีบอัด	18194 บิต

ความแตกต่าง เท่ากับ 5806 บิต

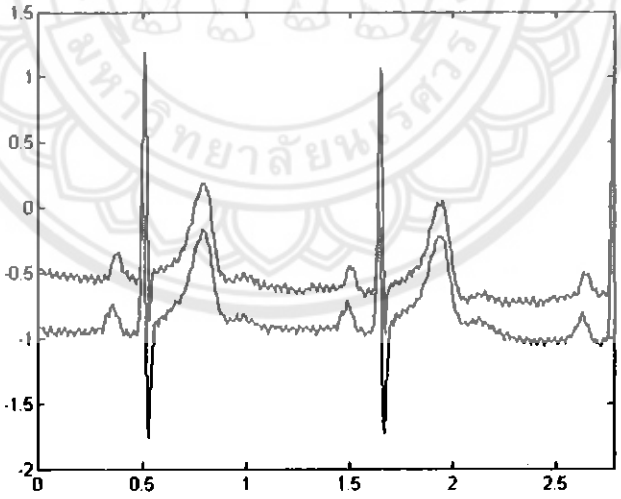
เมื่อเราป้อนคำสั่งลงในช่อง Command ของ matlab max(packed) แล้วกด enter ค่าจะออกมา ค่าที่มากที่สุดของ Packed (ที่ถูกบีบอัด) = 1869

ค่าความแตกต่างระหว่างก่อนการบีบอัดและหลังคลายการบีบอัด โดยอาศัยตัวเลข เมื่อตัวเลขก่อนการบีบอัดและหลังคลายการบีบอัดที่ตำแหน่งเดียวกันลบกันต้องมีค่าเท่ากับ 0 เสมอ ก่อนการบีบอัดใช้ข้อมูล 8 บิต จำนวน 3000 ข้อมูล เท่ากับ 8*3000 = 24000 บิต พอบีบอัดแล้วใช้ข้อมูล 11 บิต เพราะตัวข้อมูลมีค่าสูงสุด 1869 ซึ่งสูงกว่า 8 บิต (255) มีจำนวน 1654 ข้อมูล เท่ากับ 11*1654 = 18194 ข้อมูล จากตอนแรกก่อนการบีบอัดมีข้อมูล 24000 บิต พอบีบอัดแล้วมีข้อมูล 18194 บิต สามารถบีบอัดได้ 24000 – 18194 = 5806 บิต คิดเป็นค่าCR = 24000/18194 = 1.3191 แสดงว่าบีบอัดข้อมูลได้จริงซึ่งเป็นการบีบอัดที่ไม่เกิดการสูญเสีย

เริ่มแรกเมื่อเราเปิดโปรแกรมขึ้นมา โดยกรอกข้อมูลเป็นไฟล์ 117 โดยที่ไฟล์มันจะมี 3 หลัก แล้วเราจะรวม 3 หลักมาเป็นหลักเดียว โดยเอาหลักที่ 2 มาต่อหลักที่ 1 เอาหลักที่ 3 มาต่อหลักที่ 2



รูปที่ 4.7 สัญญาณ ECG ไฟล์ 117 ก่อนการบีบอัด



รูปที่ 4.8 สัญญาณ ECG ไฟล์ 117 หลังการคลายการบีบอัด (CR = 1.3765 , SNR = ∞ และ PSNR = ∞)

คำอธิบายรูป 4.3 คือรูปสัญญาณ ECG ที่ก่อนการบีบอัดข้อมูล ซึ่งเป็นรูป ECG 2 สัญญาณที่ใช้ในการวิจัย แกน y คือ mV กับแกน x คือ เวลา โดยที่เส้นสีแดงคือสัญญาณ ECG บริเวณหน้าอกด้านขวา ที่ไม่มีอาการป่วย เส้นสีน้ำเงิน คือสัญญาณ ECG ที่บริเวณหน้าอกด้านขวาของคนที่เป็นโรคไข้เลือดออก

คำอธิบายรูป 4.4 คือรูปสัญญาณ ECG หลังจากการคลายการบีบอัดแล้ว ซึ่งเป็นรูป ECG 2 สัญญาณที่ใช้ในการวิจัย แกน y คือ mV กับแกน x คือ เวลา โดยที่เส้นสีแดงคือสัญญาณ ECG บริเวณหน้าอกด้านขวา ที่ไม่มีอาการป่วย เส้นสีน้ำเงิน คือสัญญาณ ECG ที่บริเวณหน้าอกด้านขวาของคนที่เป็นโรคไข้เลือดออก

เราเอาข้อมูล ECG ทั้ง 3 หลัก มาต่อรวมกันหมดเป็นหลักเดียว เพื่อที่จะทำการบีบอัดข้อมูลได้ง่ายขึ้น

ตารางที่ 4.4 ขนาดข้อมูลไฟล์ 117

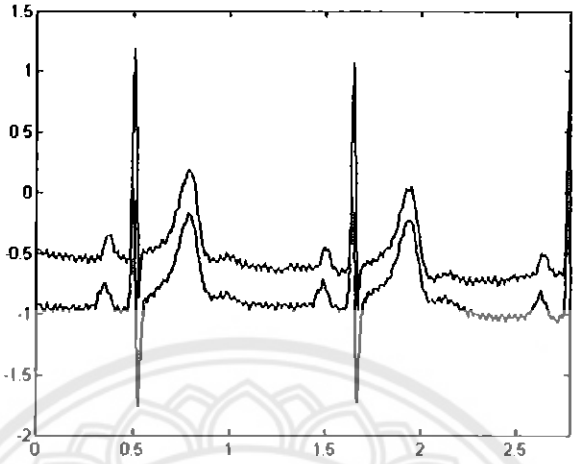
ค่าข้อมูล	ขนาด
ค่าข้อมูลดั้งเดิม	24000 บิต
ค่าข้อมูลตอนบีบอัด	17435 บิต

ความแตกต่าง เท่ากับ 6565 บิต

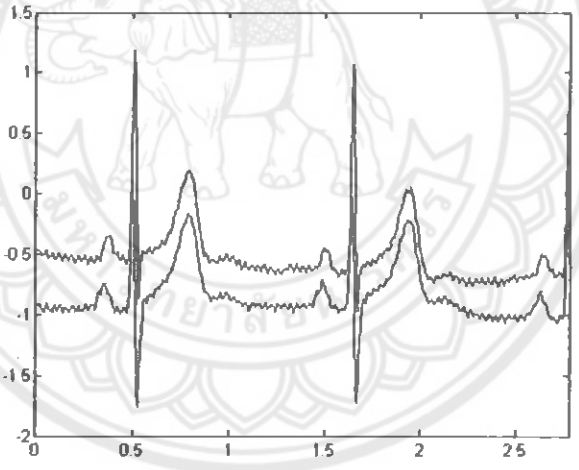
เมื่อเราป้อนคำสั่งลงในช่อง Command ของ matlab max(packed) แล้วกด enter ค่าจะออกมา ค่าที่มากที่สุดของ Packed (ที่ถูกบีบอัด) = 1816

ค่าความแตกต่างระหว่างก่อนการบีบอัดและหลังคลายการบีบอัด โดยอาศัยตัวเลข เมื่อตัวเลขก่อนการบีบอัดและหลังคลายการบีบอัดที่ตำแหน่งเดียวกันลบกันต้องมีค่าเท่ากับ 0 เสมอ ก่อนการบีบอัดใช้ข้อมูล 8 บิต จำนวน 3000 ข้อมูล เท่ากับ 8*3000 = 24000 บิตพอบีบอัดแล้วใช้ข้อมูล 11 บิต เพราะตัวข้อมูลมีค่าสูงสุด 1869 ซึ่งสูงกว่า 8 บิต (255) มีจำนวน 1654 ข้อมูล เท่ากับ 11*1585 = 17435 ข้อมูลจากตอนแรกก่อนการบีบอัดมีข้อมูล 24000 บิตพอบีบอัดแล้วมีข้อมูล 17435 บิตสามารถบีบอัดได้ 24000 – 17435 = 6565 บิตคิดเป็นค่า CR = 24000/17435 = 1.3765 แสดงว่าบีบอัดข้อมูลได้จริงซึ่งเป็นการบีบอัดที่ไม่เกิดการสูญเสีย

เริ่มแรกเมื่อเราเปิดโปรแกรมขึ้นมา โดยกรอกข้อมูลเป็นไฟล์ 118 โดยที่ไฟล์มันจะมี 3 หลัก แล้วเราจะรวม 3 หลักมาเป็นหลักเดียว โดยเอาหลักที่ 2 มาต่อหลักที่ 1 เอาหลักที่ 3 มาต่อหลักที่ 2



รูปที่ 4.9 สัญญาณ ECG ไฟล์ 118 ก่อนการบีบอัด



รูปที่ 4.10 สัญญาณ ECG ไฟล์ 118 หลังการคลายการบีบอัด (CR = 1.2169 , SNR = ∞ และ PSNR = ∞)

คำอธิบายรูป 4.3 คือรูปสัญญาณ ECG ที่ก่อนการบีบอัดข้อมูล ซึ่งเป็นรูป ECG 2 สัญญาณที่ใช้ในการวิจัย แกน y คือ mV กับแกน x คือ เวลา โดยที่เส้นสีแดงคือสัญญาณ ECG บริเวณหน้าอกด้านซ้าย ที่ไม่มีอาการป่วย เส้นสีน้ำเงิน คือสัญญาณ ECG ที่บริเวณหน้าอกด้านขวาของคนที่เป็นโรคไข้เลือดออก

คำอธิบายรูป 4.4 คือรูปสัญญาณ ECG หลังจากการคลายการบีบอัดแล้ว ซึ่งเป็นรูป ECG 2 สัญญาณที่ใช้ในการวิจัย แกน y คือ mV กับแกน x คือ เวลา โดยที่เส้นสีแดงคือสัญญาณ ECG บริเวณหน้าอกด้านซ้าย ที่ไม่มีอาการป่วย เส้นสีน้ำเงิน คือสัญญาณ ECG ที่บริเวณหน้าอกด้านขวาของคนที่เป็นโรคไข้เลือดออก

เราเอาข้อมูล ECG ทั้ง 3 หลัก มาต่อรวมกันหมดเป็นหลักเดียว เพื่อที่จะทำการบีบอัดข้อมูลได้ง่ายขึ้น

ตารางที่ 4.5 ขนาดข้อมูลไฟล์ 118

ค่าข้อมูล	ขนาด
ค่าข้อมูลดั้งเดิม	24000 บิต
ค่าข้อมูลตอนบีบอัด	19723 บิต

ความแตกต่าง เท่ากับ 6565 บิต

เมื่อเราป้อนคำสั่งลงในช่อง Command ของ matlab max(packed) แล้วกด enter ค่าจะออกมา ค่าที่มากที่สุดของ Packed (ที่ถูกบีบอัด) = 1968

ค่าความแตกต่างระหว่างก่อนการบีบอัดและหลังคลายการบีบอัด โดยอาศัยตัวเลข เมื่อตัวเลขก่อนการบีบอัดและหลังคลายการบีบอัดที่ตำแหน่งเดียวกันลบกันต้องมีค่าเท่ากับ 0 เสมอ ก่อนการบีบอัดใช้ข้อมูล 8 บิต จำนวน 3000 ข้อมูล เท่ากับ 8*3000 = 24000 บิตพอบีบอัดแล้วใช้ข้อมูล 11 บิต เพราะตัวข้อมูลมีค่าสูงสุด 1869 ซึ่งสูงกว่า 8 บิต (255) มีจำนวน 1654 ข้อมูล เท่ากับ 11*1968 = 19723 ข้อมูลจากตอนแรกก่อนการบีบอัดมีข้อมูล 24000 บิตพอบีบอัดแล้วมีข้อมูล 19723 บิตสามารถบีบอัดได้ 24000 – 19723 = 4277 บิตคิดเป็นค่า CR = 24000/19723 = 1.2169 แสดงว่าบีบอัดข้อมูลได้จริงซึ่งเป็นการบีบอัดที่ไม่เกิดการสูญเสีย

จากผลการทดลองการบีบอัดข้อมูล ECG โดยวิธี Lempel Ziv ดังกล่าวข้างต้นสามารถสรุปผลที่ได้จากการบีบอัดข้อมูล ECG โดยวิธี Lempel Ziv ดังตารางที่ 4.6

ตารางที่ 4.6 สรุปผลที่ได้จากการบีบอัดข้อมูล ECG โดยวิธี Lempel Ziv ทั้ง 5 ไฟล์

ลำดับที่	ชื่อไฟล์	CR	SNR	PSNR
1	100	1.6066	∞	∞
2	101	1.6554	∞	∞
3	116	1.3191	∞	∞
4	117	1.3765	∞	∞
5	118	1.2169	∞	∞

ผลการวิเคราะห์

จากผลการทดลองค่าในตารางที่ 4.6 สรุปผลที่ได้จากการบีบอัดข้อมูล ECG โดยวิธี Lempel Ziv ทั้ง 5 ไฟล์นั้น ค่าของการอัตราการบีบอัดข้อมูล (Compression Ratio) อยู่ที่ 1.2169-1.6554 โดยที่มีค่า CR สูงสุดอยู่ที่ 1.6554 ในขณะที่ค่า SNR และ PSNR มีค่าเข้าสู่เลขอนันต์ทุกไฟล์นั้น แสดงให้เห็นว่าการบีบอัดข้อมูลและคลายการบีบอัดข้อมูล ECG โดยวิธี Lempel Ziv มีคุณลักษณะเหมือนกันทุกประการ

บทที่ 5
สรุปผลการทดลอง

5.1 สรุปผลการทดลอง

การบีบอัดข้อมูล ECG โดยใช้วิธี Lempel Ziv สามารถบีบอัดข้อมูล ECG ได้ค่า CR (Compression Ratio) สูงสุด 1.6554 คือ ข้อมูล ECG ไฟล์ 101 รองลงมาคือ ข้อมูล ECG ไฟล์ 100 มีค่า CR เท่ากับ 1.6066 อันดับสามคือไฟล์ 117 มีค่า CR เท่ากับ 1.3765 อันดับสี่คือไฟล์ 116 มีค่า CR เท่ากับ 1.3191 อันดับสุดท้ายคือไฟล์ 118 มีค่า CR เท่ากับ 1.2169 โดยที่ทุกไฟล์มีค่า SNR และ PSNR มีค่าเข้าใกล้อนันต์ และไม่มีการสูญเสียข้อมูล ECG

5.2 ปัญหาในการทดลองและแนวทางการแก้ไข

ปัญหาที่เกิดขึ้น

ถ้าใช้ค่า SAMPLES2READ นำข้อมูลสัญญาณมามากเกินไป ซึ่งปกติจะประมาณ 30000 ข้อมูล จะทำให้เวลารันโปรแกรมใช้เวลานานมาก

แนวทางการแก้ไขปัญหา

ใช้ค่า SAMPLES2READ นำข้อมูลสัญญาณมาขนาดพอดี ประมาณ 1000 ข้อมูล ซึ่งข้อมูลขนาดแค่นี้จะสามารถทำให้การรันโปรแกรมทำได้เร็วมาก และรูปสัญญาณที่ได้สามารถอ่านได้

5.3 แนวทางการพัฒนาในอนาคต

เนื่องจากการการพยายามการบีบอัดข้อมูลสัญญาณ ECG โดยวิธี Lempel Ziv นั้นเวลาการรันโปรแกรมที่มี SAMPLES2READ จำนวนเยอะ ไม่สามารถทำได้โดยเร็วนัก ดังนั้นควรเขียนโปรแกรมที่สามารถใช้เวลาได้น้อยลงจากการรันโปรแกรมที่มีค่า SAMPLES2READ มากๆได้

เอกสารอ้างอิง

[1] หน่วยโรคหัวใจเด็ก Pediatric Cardiology ภาควิชากุมารเวชศาสตร์ คณะแพทยศาสตร์ โรงพยาบาลรามาธิบดี "การตรวจคลื่นไฟฟ้าหัวใจ" [Online]. Available : http://www.mahidol.ac.th/mahidol/ra/rapd/carsecg.htm. 2004.

[2] Steve Holmes "C Programming" [Online]. Available : http://www.imada.sdu.dk/~svalle/courses/dm14-2005/mirror/c/. 1995.

[3] Dr.Sandra I. Wooley "Interactive Data Compression Tutor. [Online]. Available : http://www.eee.bham.ac.uk/WoolleySI/links/datacomp.htm. 2005.

[4] Dave Marshall. "Lempel-Ziv-Welch (LZW) Algorithm" [Online]. Available : http://www.cs.cf.ac.uk/Dave/Multimedia/node214.html. 2001.

[5] Dr. John Brotherhood "The Electrocardiogram" [Online]. Available : http://www2.fhs.usyd.edu.au/ess/brotherhood/ECG.PPT. 2004.

[6] Duane Hanselman. Bruce Littlefield. **Mastering MATLAB 7.** 1st Ed., New Jersey : Personal Education, Inc. 2005.

[7] Adrian Biran. Moshe Breiner. **MATLAB 5 for Engineers.** 2nd Ed., Great Britain : Henry Ling Ltd. 1999.

[8] Amos Gilat. **MATLAB An Introduction with Applications.** 1st Ed., United State of America : John Wiley & Sons, Inc. 2004.

[9] Harold P.E. Stern. Samy A. Mahmoud. Communication Systems. 1st Ed.; New Jersey : Personal Education, Inc. 2004.

[10] Ashok Ambardar. **Analog and Digital Signal Processing**. 1st Ed., Boston : Mass-Pws. 1945.

[11] รศ.ดร. มนัส สังวรศิลป์. วรรัตน์ ภัทรอมรกุล. **คู่มือโปรแกรม MATLAB ฉบับสมบูรณ์.** พิมพ์ครั้งที่ 1. กรุงเทพฯ : สำนักพิมพ์ อินโฟเพรส. 2543.

[12] ดร.พิพัฒน์ หิรัณย์วณิชชากร. **ระบบการสื่อสารข้อมูล และเครือข่ายคอมพิวเตอร์.** พิมพ์ครั้งที่ 1. กรุงเทพฯ : ซีเอ็ดยูเคชั่น. 2542.

[13] บัณฑิต โรจน์อารยานนท์. **หลักการไฟฟ้าสื่อสาร Principle of Communication Systems.** พิมพ์ครั้งที่ 6. กรุงเทพฯ : โรงพิมพ์จุฬาลงกรมหาวิทยาลัย. 2540.

[14] สุธรรม ศรีเกษม และคณะ. **MATLAB เพื่อการแก้ปัญหาทางวิศวกรรม.** พิมพ์ครั้งที่ 1. กรุงเทพฯ : ซีเอ็ดยูเคชั่น. 2538.

[15] ยุทธนา ลีลาศวัฒนกุล. **คู่มือการเขียนโปรแกรมและใช้งาน ภาษา C/C++ ฉบับสมบูรณ์.** พิมพ์ครั้งที่ 1. นนทบุรี : อินโฟเพรส. 2542.

[16] นิรุธ อำนวยศิลป์. **C Programming เขียนโปรแกรมภาษา C ฉบับสมบูรณ์.** พิมพ์ครั้งที่ 1. กรุงเทพฯ : บริษัทเฮ็ช เอ็น กรุ๊ป จำกัด. 2549.

[17] วิทยา สุคตบวร. **การเขียนโปรแกรมคอมพิวเตอร์ด้วย C.** พิมพ์ครั้งที่ 1 กรุงเทพฯ : ซีเอ็ดยูเคชั่น. 2547.

[18] จารุทัศน์ วงษ์สันต์. **MATLAB สำหรับการแก้ปัญหาเชิงวิทยาศาสตร์ และวิศวกรรม.** พิมพ์ครั้งที่ 1. กรุงเทพฯ : ฟิสิกส์เซ็นเตอร์. 254

ภาคผนวก ก

ความเป็นมาของโปรแกรม Matlab

MATLAB เป็นซอฟท์แวร์ (software) สำหรับการวิเคราะห์เชิงตัวเลข (numerical analysis) เขียนขึ้นโดยDr. Cleve Moler ตั้งแต่ ค.ศ 1982 ในตอนแรกเป็นการรวบรวมโปรแกรมการคำนวณเกี่ยวกับชุดข้อมูลแมททริกซ์(matrices) ที่เขียนขึ้นโดยใช้ภาษาโฟร์แทรน (FORTRAN) แล้วเรียบเรียงเป็นหนังสือชื่อ *Demonstration ofMatrix Library* โดย C.B Moler ต่อมามีการพัฒนาจนกลายเป็นโปรแกรมขนาดใหญ่ ที่เขียนโดยใช้ภาษาซี(C) และแอสเซมเบลอร์ (assembler) รากฐานของ MATLAB นั้น มาจากโครงการ EISPACK (Eigen System Package) และ LINPACK (Linear System Package) ณ Argonne National Laboratory สหรัฐอเมริกา โดย EISPACK เป็นโปรแกรมย่อยภาษาโฟร์แทรน (FORTRAN SUBROUTINES) ที่ใช้คำนวณ eigen values และ eigen vectors สำหรับmatrices ประเภทต่างๆ จะมีคำอธิบายโดยละเอียดในหนังสือ Matrix Eigen System Routines : EISPACKGuide โดย Smith, Boyle, Dongarra, Garbow, Ikebe, Klema และ Moler

ส่วน LINPACK เป็นโปรแกรมย่อยภาษาโฟร์แทรน สำหรับการวิเคราะห์และแก้ระบบสมการพีชคณิตเชิงเส้น (simultaneous linear algebraic equations) และปัญหาด้าน linear least square ซึ่งมีคำอธิบายในThe LINPACK User's Guide โดย Dongarra, Bunch, Moler และ Stewart ปัจจุบันมีการใช้MATLABในการเรียนการสอนทางวิศวกรรมศาสตร์ของมหาวิทยาลัยทั่วไปในสหรัฐอเมริกาซอฟท์แวร์เป็นลิขสิทธิ์ของThe MathWorks, Inc., 21 Eliot St., South Natick MA01760ผู้ใช้และผู้สนใจสามารถหาข้อมูลเพิ่มเติมได้ที่เว็บไซท์(website) http:// www.mathworks.com /

ลักษณะโดยทั่วไปของ MATLAB

การหาคำตอบเชิงตัวเลข (numerical solution) ของปัญหาทางคณิตศาสตร์ต่างๆ แม้จะเป็นเพียงค่าโดยประมาณและใช้การคำนวณซ้ำๆกันอย่างมาก แต่ก็สามารถทำได้อย่างรวดเร็ว เมื่อใช้การคำนวณโดยคอมพิวเตอร์และมีความแม่นยำในระดับที่ยอมรับได้ เมื่อเทียบกับการหาคำตอบในเชิงวิเคราะห์ (analyticalsolution) ในการจัดการกับข้อมูลที่เป็นตัวเลขในปริมาณมาก หากจัดให้อยู่ในรูปของ matrix แล้วจึงทำการประมวลผลตามวิธีการที่วางเอาไว้ จะทำให้การตรวจสอบและแสดงผลง่ายขึ้น เพราะตัวแปรที่กำหนดขึ้นในMATLAB พร้อมที่จะรับข้อมูลในลักษณะที่เป็นmatrixอยู่แล้วพร้อมกับมีโปรแกรมย่อยที่เป็นฟังก์ชั่นสำเร็จรูป(built-in functions) ที่จะสามารถนำมาใช้งานได้ทันที

การคำนวณเบื้องต้น

การคำนวณใน MATLAB เป็นเช่นเดียวกับการใช้เครื่องคิดเลข โดยใช้เครื่องหมาย บวก(+), ลบ(-), คูณ(*), หาร (/) และการยกกำลัง (^)

ลำดับการทำงานจะเริ่มต้นจากซ้ายไปขวา โดยเครื่องหมายยกกำลังมี อันดับแรกสุด ตามด้วยเครื่องหมายคูณและหารซึ่งมีอันดับเท่ากันหลังจากนั้นเป็นอันดับของ เครื่องหมายบวกและลบ

ตัวอย่างที่ 1

```
>>  3^2 - 5 -6 / 3 * 2
ans =
        0

>> 3^2  - 5  - 6 /( 3 * 2 )
ans  =
        3

>> 4 * 3^2 + 1
ans  =
        37

>>  ( 4 * 3 )^2 + 1
ans  =
        145
```

ชุดข้อมูลตาราง(matrices)

ตัวแปร (variables) แต่ละตัวของ MATLAB สามารถรับข้อมูลที่เป็นชุดตัวเลขเดี่ยว ๆ หรือเป็นชุดได้ โดยไม่ต้องมีการกำหนดตั้ง (declaration) เช่นเดียวกับภาษาคอมพิวเตอร์โดยมากเช่น ซี (C), เบสิค (basic) เป็นต้น

การทำงานเป็นไป ตามแบบการทำงานของ matrix (matrix operations)

ตารางที่ ผ.1 การทำงานของ matrix และ เครื่องหมาย

การทำงาน	เครื่องหมาย
บวก	+
ลบ	-
คูณ	*
หาร ขวา (right division)	/
หาร ซ้าย (left division)	\
ยกกำลัง (powers)	^
ย้ายทแยง (transpose)	'

ข้อสังเกตเกี่ยวกับ เครื่องหมายหาร ซึ่งมี 2 แบบคือ หารขวาและ หารซ้าย หาก A เป็น ชุดข้อมูลตารางด้านเท่า (square matrix) และ b เป็น ชุดข้อมูลแถวเดี่ยว (vector) ที่สอดคล้องกัน ของสมการ

$$A * X = b$$

คำตอบของสมการจะเป็น $X = A \backslash b$

สำหรับสมการ

$$X * A = b$$

คำตอบของสมการ จะเป็น $X = b / A$

เครื่องหมาย คูณ(*), ยกกำลัง(^), หารซ้าย (\ ) และ หารขวา(/) เมื่อ นำหน้าด้วย จุด (period , .) จะหมายถึง ทำงานแบบตัวต่อตัวตามลำดับ (entrywise) ของชุดข้อมูลแถวเดี่ยวนั้น ๆ

ตัวอย่างที่ 2

```
>> [1,2,3,4].*[1,2,3,4]
answer =
          1   4   9   16
>> [1,2, 3,4 ].^2
answer =
          1   4   9   16
```

ฟังก์ชันสำเร็จรูป (built-in functions)

MATLAB มีฟังก์ชั่นสำหรับจัดการกับตัวเลขและชุดของตัวเลข ซึ่งสามารถเรียกมาใช้ได้ทันที ฟังก์ชันเบื้องต้นสำหรับปริมาณเดี่ยว (scalar function) ชุดข้อมูลแถวเดี่ยวหรือเวคเตอร์ (vector) และชุดข้อมูลตารางหรือแมททริก (matrix) แสดงไว้ดังตารางที่ 2.2

ตารางที่ ผ.2 ฟังก์ชันเบื้องต้นของ MATLAB

Scalar Functions	
sin (x)	Sine
cos (x)	Cosine
tan (x)	Tangent
asin (x)	Inverse sine
acos (x)	Inverse cosine
atan (x)	Inverse tangent
sinh (x)	Hyperbolic sine
Complex Number	
real (x)	Complex real part
imag (x)	Complex imaginary part
Vector Functions	
max (x)	Maximum value of columns
min (x)	Minimum value of columns
sort (x)	Sort columns in ascending order
sum (x)	Sum of elements in each column
mean (x)	Mean or average value of columns
median (x)	Median value of columns
std (x)	Standard deviation of columns

ฟังก์ชั่นสำหรับเวคเตอร์จะใช้ได้กับทั้ง แถวนอน (row)และแถวตั้ง (column) หากใช้กับแมททริกจะทำงานไปทีละคอลัมน์

ตัวอย่างที่ 3

```
>> A =[ 1 2 3 ; 4 5 6 ; 7 8 9 ]
   ans A  =
            1     2     3
            4     5     6
            7     8     9

>>     max ( A )
       ans     =
                  7     8     9

>>     max ( max ( A ) )

       ans     =
                  9
```

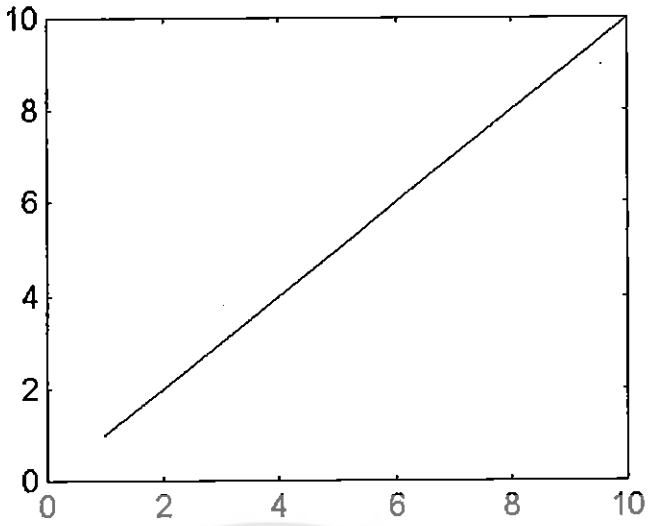
การพล็อตอย่างง่าย (simple plots)

การแสดงผลของ MATLAB สามารถแสดงออกมาในรูปของกราฟได้หลายแบบที่ง่ายที่สุดคือการเขียนคำสั่ง plot(A) เมื่อ A เป็น ข้อมูลที่ต้องการพล็อต แกน Y จะแสดงค่าของ A ส่วนแกน X จะเป็น อันดับของ ค่าในชุดข้อมูล

ตัวอย่างที่ 3

```
>> A = 1 : 10
   ans     A =
            1   2   3   4   5   6   7   8   9   10
>> plot ( A )
```

จะได้กราฟ



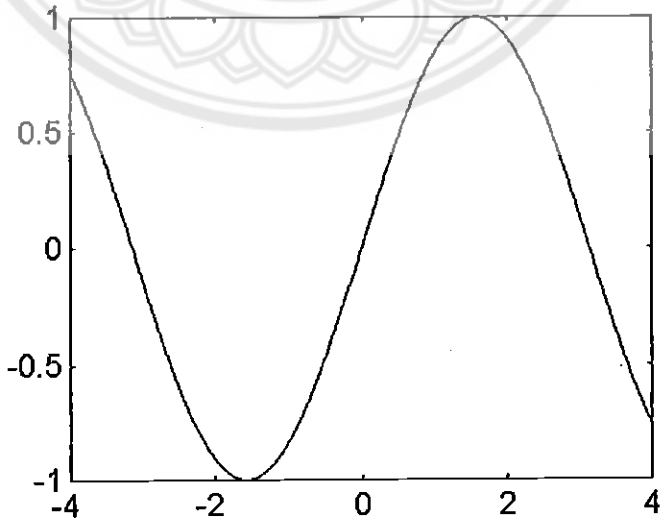
รูปที่ ผ.1 กราฟของตัวอย่างที่ 3

ในกรณีที่ต้องการพล็อตข้อมูลที่เป็นคู่ (coordinates) ก็จัดเตรียมข้อมูล 2 ชุด คือ X และ Y ที่มีจำนวนข้อมูลเท่ากันเขียนคำสั่ง plot(X , Y) ก็จะได้ กราฟของค่า X บนแกน x และค่า Y บนแกน y ตามลำดับ

ตัวอย่างที่ 4

```
>> x = - 4 : 01 : 4 ;
>> y = sin ( x )
>> plot( x, y )
```

จะได้กราฟ



รูปที่ ผ.2 กราฟของตัวอย่างที่ 4

นอกจากนี้ ยังสามารถที่จะพล็อตลงบนแกน semi-log และ log-log โดยใช้คำสั่งดังตารางที่ 2. 3

ตารางที่ ผ.3 คำสั่งที่ใช้พล็อตกราฟ

คำสั่ง		ความหมาย
semilogx	(x,y)	กราฟที่แกน x เป็น ค่า logarithmic scale
semilogy	(x,y)	กราฟ ที่แกน y เป็นค่า logarithmic scale
loglog	(x,y)	กราฟที่แกนทั้งสองเป็น logarithmic scale
bar	(y)	กราฟแท่งของ ค่า y
polar	(theta , rho)	กราฟวงกลมของมุม (angle) theta ในหน่วย เดียวกับแนวรัศมี (radius vector) rho

การเขียนคำอธิบายลงในกราฟ สามารถใช้คำสั่ง title ('string'), xlabel ('string') และ ylabel ('string ') โดย strings หมายถึงข้อความที่ต้องการใส่ ลงไป

ตัวอย่างที่ 5

```
>> t = 0 : ( .99 * pi / 2 ) : 500 ;
>> x = t . * cos ( t ) ;
>> plot ( x , t ) ;
>> title ( 'SAMPLE PLOT ' )
>> xlabel ( 'ABCISSA' )
>> ylabel ( 'COORDINATE' )
```

หมายเหตุ pi ในที่นี้คือค่า π เป็นอักษรพิเศษที่ MATLAB กำหนดขึ้นโดยเฉพาะ ซึ่งจะได้กล่าวถึงต่อไป

ตัวแปรและตัวแปรพิเศษ

ตัวแปรที่ใช้ใน MATLAB สามารถตั้งขึ้นโดยมีคุณสมบัติดังนี้

1. การใช้ตัวสะกดเล็กและใหญ่ ถือว่าเป็นคนละตัวกัน (case sensitive) เช่น Fruit เป็นคนละตัวกับ fruit

2. มีความยาวได้ถึง 19 ตัวอักษร

3. ต้องเริ่มต้นด้วยตัวอักษร

นอกจากนี้ ยังมีอักษรเฉพาะ ที่สงวนไว้สำหรับใช้เป็นตัวแปรพิเศษ ดังตารางที่ 2.4

ตารางที่ ผ.4 ตัวแปรพิเศษของ MATLAB

ตัวแปร	ความหมาย
ans	ตัวแปรใช้สำหรับผลลัพธ์
pi	พาย หรืออัตราส่วนของเส้นรอบวงต่อเส้นผ่าศูนย์กลาง
eps	เลขทศนิยมที่เล็กที่สุด ของคอมพิวเตอร์
inf	อสงไขย หรืออินฟินิตี้ เช่น 1/0
NaN	หาค่าไม่ได้(Not-a-Number) เช่น 0/0
i และ j	$i = j = \sqrt{-1}$
realmin	จำนวนจริง ที่น้อยที่สุด ที่เป็นบวก ที่ใช้ได้
realmax	จำนวนจริง ที่มากที่สุด ที่เป็นบวก ที่ใช้ได้

M-files

การเขียนโปรแกรมสำหรับ MATLAB จะต้องเก็บไว้ในไฟล์ที่ต่อท้ายด้วย .m (น่าจะหมายถึง MATLAB-file - ผู้เรียบเรียง) เป็นการรวบรวมเอาคำสั่งต่างๆที่โดยปกติจะพิมพ์ไว้ข้างหลังเครื่องหมาย >> ของแต่ละบรรทัด ซึ่งจะมีประโยชน์ในการที่ต้องการให้คอมพิวเตอร์ทำตามขั้นตอน ต่อเนื่องกัน และสามารถเรียกใช้ใหม่ได้ โดยพิมพ์เฉพาะชื่อไฟล์โดยที่ไม่ต้องมี .m ลงไป หลังเครื่องหมาย >> แล้วกด enter

ตัวอย่างที่ 6

ชื่อไฟล์ plots . m

```
% This demo show some interesting plots .
t = 0 : (.99 * pi / 2 ) : 500
x = t . * cos ( t ) ;
plot ( x , t ) ;
disp ( 'PRESS ANY KEY TO CONTINUE' )
pause ;
```

```
y : t . * sin ( t );
plot ( x , y )
end
```

หมายเหตุ

1. การ ใส่คำอธิบาย (comments) ลงในโปรแกรมจะใช้ ตามหลัง เครื่องหมาย % ซึ่งคอมพิวเตอร์จะไม่ใช้ประโยคนั้นในการคำนวณ

2. คำสั่ง disp คือ การแสดงข้อความในวงเล็บ ภายใต้เครื่องหมายคำพูดลงบนจอ

3. คำสั่ง pause คือการหยุดการทำงานชั่วคราวจนกว่าจะมีการกดปุ่มใด ๆ บนแป้นพิมพ์

ตัวอย่างนี้ เมื่อพิมพ์คำสั่ง >> plots จะให้กราฟ ของ cosine ในรูปแรก เมื่อกดปุ่มใดๆบนแป้นพิมพ์ก็จะได้กราฟของ sine ในรูปที่สอง (ลองทำดู)

เส้นทางการค้นไฟล์ (path)

M-file ที่ MATLAB สามารถเรียกมาใช้ได้จะต้องอยู่ภายในเส้นทางที่กำหนดไว้ สำหรับเครื่อง Apple Macintosh จะสามารถใส่ M-file ภายใน MATLAB folder ที่ใดก็ๆได้หรือใน directory ที่สร้างใหม่ ที่อยู่ภายใน MATLAB folder ซึ่ง MATLAB จะสามารถสร้างเส้นทางไปสู่ไฟล์ใหม่ได้โดยอัตโนมัติ

สำหรับ พีซี (PC) ที่ใช้ระบบ windows และ server ที่ใช้ระบบ Unix เส้นทางการค้นไฟล์จะถูกบันทึกไว้ในไฟล์ matlabrc.m ซึ่งอยู่ใน C:/matlab/bin การแก้ไขหรือเพิ่มเติมทำได้โดยการเพิ่มเส้นทางเข้าไปโดยใช้ รูปแบบของบรรทัดที่เขียนไว้ก่อนหน้านั้น หลังจากนี้ก็ออกจาก MATLAB แล้ว run ใหม่ เพื่อให้ path ใหม่ ถูก load ไว้ในระบบ

การตรวจสอบว่าไฟล์ใหม่อยู่ในเส้นทางการค้นหาหรือไม่ ให้ใช้คำสั่ง >> path และกด enter จะมีรายการของ directory ซึ่งอยู่ในเส้นทางการค้นหาปรากฏขึ้นบนจอ การกำหนด path จะวาง directory ไว้ ณ ที่ใดก็ได้ ไม่จำเป็นต้องอยู่ใน root directory ของ MATLAB

การทำงานแบบวนรอบ (loop operations)

การคำนวณ ซ้ำ ๆ กัน หลายครั้ง มักเป็นหน้าที่ที่เหมาะสมสำหรับการใช้งานของคอมพิวเตอร์ ภาษาคอมพิวเตอร์จึงต้องมีคำสั่งที่ใช้ ให้ทำงานแบบนี้

ลักษณะของคำสั่งจะเป็น

for I = 1 : n , x (I) = 0 , end

โดยที่ I มีค่าเริ่มต้น = 1 ซึ่งจะทำงานแล้วกลับมาเริ่มต้นใหม่โดย I = 2 ไปถึงคำสั่ง end ไป จนกระทั่ง I มีค่าเท่ากับ n เมื่อทำงานครบรอบแล้วจึงจะหยุดทำงาน

ตัวอย่างที่ 7

```
>> for n = I : 10
       x ( n ) = sin ( n * pi / 10 )
       end
>> plot ( x )
```

คำสั่งความสัมพันธ์ (relational operators)

เป็นการเปรียบเทียบเชิงปริมาณของตัวแปรต่าง ๆ ดังตารางที่ 3.1

ตารางที่ ผ.5 คำสั่งความสัมพันธ์ของ MATLAB

คำสั่ง (operators)	ความหมาย
<	น้อยกว่า
< =	น้อยกว่าหรือเท่ากับ
>	มากกว่า
>=	มากกว่าหรือเท่ากับ
=	เท่ากับ
~=	ไม่เท่ากับ

คำสั่งตรรกะ (logical operators)

เป็นการรวมหรือแยกของความสัมพันธ์

ตารางที่ ผ.6 คำสั่งตรรกะของ MATLAB

คำสั่ง (operators)	ความหมาย
&	และ (AND)
|	หรือ (OR)
~	ไม่ (NOT)

ตัวอย่างที่ 8

```
>> A = 1 : 9 ; B = 9 - A ;
>> tf = A > 4
    tf =
          0   0   0   0   1   1   1   1   1

>> tf = ( A >2 ) & ( A < 6 )
    tf =
          0   0   1   1   1   0   0   0   0
```

การทำงานโดยตรรกะ (logical operations)

การทำงานของคอมพิวเตอร์มักมีการใช้การเปรียบเทียบเพื่อการตัดสินใจ ลักษณะการใช้งานจะเป็น

```
if  n < 0
        x ( n ) = 0;
elseif rem  ( n, 2)  == 0
        x ( n ) = 2 ;
else
        x ( n )  = 1 ;
end
```

การใช้ if (ถ้า) เป็นความหมายของการตั้งเงื่อนไข หากเป็นจริงก็จะมีการตัดสินใจอย่างหนึ่ง หรือหากไม่เป็นจริงก็จะมีการตัดสินใจเป็นอย่างอื่น

ฟังก์ชั่นไฟล์ (function files)

ในการที่ต้องการใช้ฟังก์ชั่นใหม่ที่ไม่มีอยู่ภายใน MATLAB เราสามารถเขียน M-file ให้ทำหน้าที่เป็นฟังก์ชั่นใหม่ได้เองโดยการกำหนดคำสั่ง

```
function [return1, return2, ...] = filename(var1, var2)
```

เอาไว้เป็นบรรทัดแรก

ตัวอย่างที่ 9

ชื่อไฟล์ prodsqr . m

```
function  p = prodsqr  ( A , B )
%  product of  the  square are  of  two  matrices.
p  =  A^2  *  B^2 ;
```

เมื่อวางไฟล์ prodsqr.m เอาไว้ในเส้นทางการค้นของ MATLAB แล้ว เราจะสามารถเรียกใช้ prodsqr( ) ซึ่งเป็นฟังก์ชั่นที่สร้างขึ้นใหม่ ได้เช่นเดียวกับฟังก์ชั่นสำเร็จรูป

การใส่และการเก็บไฟล์ข้อมูล (load and save)

หลังจากการคำนวณ MATLAB จะสามารถเก็บข้อมูลทั้งหมดไว้ในไฟล์ข้อมูลเฉพาะเรียกว่า MAT-file โดยใช้คำสั่ง save filename แล้วสามารถเรียกมาใช้ใหม่ได้โดย คำสั่ง load filename ซึ่งชื่อไฟล์ข้อมูลนี้ จะต่อท้ายด้วย .mat

นอกจากนี้ยังสามารถเก็บค่าตัวแปร แต่ละค่าได้ในไฟล์ตัวหนังสือ (ASCII file, อ่านว่า แอสกี้-ไฟล์) ซึ่งสามารถเปิดอ่านโดยใช้โปรแกรมพิมพ์ (text editor) ทั่วไป โดยการเติมชื่อตัวแปรแล้วใส่ทางเลือก (tag) ลงไป เช่น save filename x -ascii ลงไป ส่วนการเรียกมาใช้ก็ใช้คำสั่ง load filename ซึ่งจะได้ค่าของตัวแปรตามชื่อไฟล์นั้น ชื่อไฟล์จะลงท้ายด้วยอะไรก็ได้ ยกเว้น .m และ .mat

ภาคผนวก ข

หาข้อมูล ECG จากอินเตอร์เน็ต โดยที่จะมีไฟล์ออกมาเป็น 2 ไฟล์ คือ headerfile and datafile ซึ่ง headerfile จะบอกว่าคุณสมบัติและลักษณะของข้อมูล ส่วนไฟล์ datafile นั้นจะต้องถอดรหัส Format212

เริ่มจากการเปิดโปรแกรมออกมา

```
%------ SPECIFY DATA
HEADERFILE= '101.hea';          % ดู headerfile ใน text format
DATAFILE='101.dat';             % data-file
SAMPLES2READ=1000;              % จำนวนที่จะอ่านข้อมูล
```

ต่อไปจะเป็นการอ่านค่าของ headfile

```
%------ LOAD HEADER DATA
fid1=fopen(HEADERFILE,'r');     % เป็นการเปิดไฟล์(ไม่ได้อ่าน)ถ้าเปิดได้จะรีเทิร์นตัวเลขที่
                                เป็น integer ออกมาให้ ในรูป fid1 (ถ้า error จะรีเทิร์นเลข
                                ลบ)
z= fgetl(fid1);                 % ถ้าใช้ครั้งแรกจะอ่านบรรทัดที่ 1 ใช้ครั้งที่ 2 อ่านบรรทัดที่
                                2 ถ้าไม่มีบรรทัดจะรีเทิร์น -1
A= sscanf(z, '%*s %d %d %d',[1,3]);   % อ่านค่า z โดย*s คือการอ่านค่า string (100.hea) แต่ไม่ต้อง
                                เอามาไว้ใน A d ตัวแรกคือ number of signal d ตัวต่อมา คือ
                                sample rate of data d ตัวที่สาม คือ number of sample
nosig= A(1);                    % number of signals ในที่นี้คือ 2 สัญญาณ
sfreq=A(2);                     % sample rate of data บอกไว้ที่เมื่อเปิดไฟล์ headerfile ใน
                                รูป notepad
clear A;                        % เคลียร์ A
for k=1:nosig                   % เป็นการวนรูปไฟล์ วน 2 รูป
  z= fgetl(fid1);               % อ่านค่าบรรทัดที่ 2
  A= sscanf(z, '%*s %d %d %d %d %d',[1,5]);    % (*s คือ อ่านค่า 100.hea)
  dformat(k)= A(1);     % บอก format ว่าเป็น 212 (d ตัวแรก)
  gain(k)= A(2);        % number of integers per mV เอาตัวเลขมาคูณ (d ตัวที่สอง)
  bitres(k)= A(3);      % bitresolution (d ตัวที่สาม)
```

```
   zerovalue(k)= A(4);     % integer value of ECG zero point หรือเป็นการยกระดับ (d ตัวที
firstvalue(k)= A(5);       % first integer value of signal (เป็นการตรวจสอบเผื่อเกิดการ error ) (d ตัวที่
                           ห้า)
end;
fclose(fid1);              % ปิดค่า fid (เมื่อเปิดแล้วต้องปิด)
clear A;
```

ต่อไปจะเป็นส่วนของ Load binary data

```
%------ LOAD BINARY DATA ----------------------------------------------------
if dformat~= [212,212], error('this script does not  212.'); end; % เป็นการตรวจดูว่าไฟล์ที่มา 2
                                                                  สัญญาณเป็น แบบ format 212 หรือป่าว
fid2=fopen(DATAFILE,'r');                  %เปิดไฟล์ datafile
A= fread(fid2, [3, SAMPLES2READ], 'uint8')'; % เปิดอ่านค่า fid2 โดยให้เป็น matrix มี 3 หลัก มี
                                           มีจำนวนแถวเท่ากับ samples2read โดยให้อยู่ในรูป
                                           8 บิต ให้เก็บอยู่ใน A
fclose(fid2);                              % ปิด
M2H= bitshift(A(:,2), -4);                 % จากหลักที่ 2 แถวกลาง ตัด 4 บิตสุดท้ายทิ้ง
                                           เช่น 01100100  ตัด 0100 ออก จะได้ 4 บิตแรก
M1H= bitand(A(:,2), 15);                   % จากหลักที่ 2 แถวกลาง และใช้การ and (15) ไป
                                           คือ ถ้า 1 เหมือนกัน จะเป็นเลข 1 ถ้าไม่ใช่เลข 1
                                           เหมือนกันจะเป็นเลข 0
                                           เช่น 01100100 เอามา and กับ 1111 จะได้ 4 บิตสุดท้าย

PRL=bitshift(bitand(A(:,2),8),9);    % เป็นการเช็คบิต ซึ่งจะมีค่าเท่ากับ 0 เสมอ ถ้าไม่ใช่ 0 แสดงว่า
                                     %ข้อมูลเกิดการผิดพลาด
PRR=bitshift(bitand(A(:,2),128),5); % เป็นการเช็คบิต ซึ่งจะมีค่าเท่ากับ 0 เสมอ ถ้าไม่ใช่ 0 แสดงว่า
                                     %ข้อมูลเกิดการผิดพลาด
M( : , 1)= bitshift(M1H,8)+ A(:,1)-PRL;  % เป็นการแปลงสัญญาณกลับเข้ารูปเดิมเพื่อแปลงเป็นกราฟ
                                         % โดยที่ค่าออกมาคือสัญญาณ
M( : , 2)= bitshift(M2H,8)+ A(:,3)-PRR;  % เป็นการแปลงสัญญาณกลับเข้ารูปเดิมเพื่อแปลงเป็นกราฟ
                                         % โดยที่ค่าออกมาคือสัญญาณ
if M(1,:) ~= firstvalue, error('inconsistency in the first bit values'); end;   & เช็กสัญญาณว่าถูกต้อง
                                                                                 ไหม
```

```
M( : , 1)= (M( : , 1)- zerovalue(1))/gain(1);        % แปลงเป็นสัญญาณดั่งเดิมสัญญาณแรก
M( : , 2)= (M( : , 2)- zerovalue(2))/gain(2);        % แปลงเป็นสัญญาณดั่งเดิมสัญญาณที่สอง
TIME=(0:(SAMPLES2READ-1))/sfreq;                     % หาค่าของเวลา

clear A M1H M2H PRR PRL;
figure(1); clf, box on, hold on    % ใช้คำสั่งเพื่อให้สามารถมีกราฟได้ 2 สัญญาณ
plot(TIME, M(:,1),'r');            % พล็อตกราฟสัญญาณ 1
plot(TIME, M(:,2),'b');            % พล็อตกราฟสัญญาณ 2
xlim([TIME(1), TIME(end)]);        % กำหนดระยะเวลาจากเริ่มต้นไปจนสุด
ต่อไปเป็นการ Load binary data
%------ LOAD BINARY DATA ----------------------------------------------------
if dformat~= [212,212], error('this script does not 212.'); end;
% เช็คดูว่าเป็นสัญญาณที่เป็น format 212 หรือป่าว
fid2=fopen(DATAFILE,'r');                    % เปิดdatafile
AA= fread(fid2, [3, SAMPLES2READ], 'uint8')'; % เปิดอ่านค่า fid2 โดยให้เป็น matrix มี 3 หลัก มี
                                              มีจำนวนแถวเท่ากับ samples2read โดยให้อยู่ในรูป
                                              8 บิต ให้เก็บอยู่ใน AA
 fclose(fid2);                                % ปิด
AAA=[AA(:,1);AA(:,2);AA(:,3)];                % เอาหลักที่ 2 มาต่อหลักที่ 1 เอาหลักที่ 3 มาต่อ
                                              หลักที่ 2
[packed,table]=norm2lzw(uint8(AAA));          %ดูใน norm2lzw
% unpack it
[unpacked,table]=lzw2norm(packed);            %ดูในlzw2norm
A=double([unpacked(1:SAMPLES2READ)' unpacked(SAMPLES2READ+1:2*SAMPLES2READ)'
unpacked(2*SAMPLES2READ+1:3*SAMPLES2READ)']);        %รวมสัญญาณ

M2H= bitshift(A(:,2), -4);                   % จากหลักที่ 2 แถวกลาง ตัด 4 บิตสุดท้ายทิ้ง
                                               เช่น 01100100  ตัด 0100 ออก จะได้ 4 บิตแรก
M1H= bitand(A(:,2), 15);                     % จากหลักที่ 2 แถวกลาง และใช้การ and (15) ไป
                                               คือ ถ้า 1 เหมือนกัน จะเป็นเลข 1 ถ้าไม่ใช่เลข 1
                                               เหมือนกันจะเป็นเลข 0
                                               เช่น 01100100 เอามา and กับ 1111 จะได้ 4 บิตสุดท้าย
```

```
PRL=bitshift(bitand(A(:,2),8),9);     % เป็นการเช็คบิต ซึ่งจะมีค่าเท่ากับ 0 เสมอ ถ้าไม่ใช่ 0 แสดงว่า
                                      %ข้อมูลเกิดการผิดพลาด
PRR=bitshift(bitand(A(:,2),128),5);   % เป็นการเช็คบิต ซึ่งจะมีค่าเท่ากับ 0 เสมอ ถ้าไม่ใช่ 0 แสดงว่า
                                      %ข้อมูลเกิดการผิดพลาด
M( : , 1)= bitshift(M1H,8)+ A(:,1)-PRL;  % เป็นการแปลงสัญญาณกลับเข้ารูปเดิมเพื่อแปลงเป็นกราฟ
                                      % โดยที่ค่าออกมาคือสัญญาณแรก
M( : , 2)= bitshift(M2H,8)+ A(:,3)-PRR;  % เป็นการแปลงสัญญาณกลับเข้ารูปเดิมเพื่อแปลงเป็นกราฟ
                                      % โดยที่ค่าออกมาคือสัญญาณ
if M(1,:) ~= firstvalue, error('inconsistency in the first bit values'); end; & เช็กสัญญาณว่าถูกต้อง
                                                                              ไหม
M( : , 1)= (M( : , 1)- zerovalue(1))/gain(1);      % แปลงเป็นสัญญาณดั่งเดิมสัญญาณแรก
M( : , 2)= (M( : , 2)- zerovalue(2))/gain(2);      % แปลงเป็นสัญญาณดั่งเดิมสัญญาณที่สอง
TIME=(0:(SAMPLES2READ-1))/sfreq;  % หาค่าของเวลา


clear A M1H M2H PRR PRL;
figure(2); clf, box on, hold on           % ใช้คำสั่งเพื่อให้สามารถมีกราฟได้ 2 สัญญาณ
plot(TIME, M(:,1),'r');                   % พล็อตกราฟสัญญาณ 1
plot(TIME, M(:,2),'b');                   % พล็อตกราฟสัญญาณ 2
xlim([TIME(1), TIME(end)]);               % กำหนดระยะเวลาจากเริ่มต้นไปจนสุด
err=AA-A;                                 % หาค่าว่าเกิดการ Error หรือไม่
ต่อไปเป็นการบีบอัดข้อมูล ดูตามตารางหน้าต่อไปควบคู่ไปด้วย
function [output,table] = norm2lzw(vector)        %กำหนดฟังก์ชั่น
if ~isa(vector,'uint8'),                  % ดูว่าอินพุตเป็น uint 8 หรือป่าว
        error('input argument must be a uint8 vector')
end
vector = uint16(vector(:)');              % ให้เวกเตอร์เป็นuint 16 และทรานสโพท
% initialize table (don't use cellstr because char(10) will be turned to empty!!!)
table = cell(1,256);                      % เป็นการทำให้มี 1 แถว 256 หลัก และแต่ละเซลล์สามารถ
                                          บรรจุข้อมูลได้มากกว่า 1 ข้อมูล
for index = 1:256,                        % เป็นการวนลูป 256 ลูป
        table{index} = uint16(index-1);   % เป็นการวนลูปโดยให้ table{1} = 0 ไปเรื่อยๆถึง
                                          table{256} = 255
```

```
end
output = vector;                                % ให้เอาพ์พุท มีค่าเท่ากับ vector
% main loop ต่อไปเป็นการบีบอัด แบบ Lempel Ziv
outputindex = 1;
startindex = 1;
for index=2:length(vector),
        element = vector(index);
        substr = vector(startindex:(index-1));
        code = getcodefor([substr element],table);
        if isempty(code),
                % add it to the table
                output(outputindex) = getcodefor(substr,table);
                [table,code] = addcode(table,[substr element]);
                outputindex = outputindex+1;
                startindex = index;
        else,
                % go on looping
        end
end

substr = vector(startindex:index);
output(outputindex) = getcodefor(substr,table);

% remove not used positions
output((outputindex+1):end) = [];

% ###############################################
function code = getcodefor(substr,table)
code = uint16([]);
if length(substr)==1,
        code = substr;
else, % this is to skip the first 256 known positions
```

```
        for index=257:length(table),
            if isequal(substr,table{index}),
                code = uint16(index-1);   % start from 0
                break
            end
        end
end


% ######################################################
function [table,code] = addcode(table,substr)
code = length(table)+1;   % start from 1
table{code} = substr;
code = uint16(code-1);    % start from 0

% คำอธิบายโปรแกรม
%สมมุติมีค่า 227 226 225 227 226
%กำหนด outputindex = 1
% กำหนด startindex = 1
% วนลูปโดยที่ให้ index = 2 ถึง length
% ให้ค่า element = vector(2) = 226
% ให้ค่า substr = vector ( 1 : 1) = 227
%  code = getcodefor ([substr element] , table) = ([227 226] , table)
% ไปดูที่ getcodefor (substr , table)
% code = uint16 [] กำหนดให้ code มีค่าว่าง
%  if length (substr) = 1 ดูจำนวนข้อมูลว่ามี 1 ข้อมูลหรือไม่ แต่ที่นี้คือ มี 2 ข้อมูล ทำคำสั่งต่อไป
%  for index = 257 : length (table) ให้ค่า index = 257 ไปถึง length (table)
%  if isequal (substr , table{257} ซึ่งไม่ใช่ จึงออกจากลูปของฟังก์ชั่น getcoder
% if isempty (code) ดูว่า code คือ เซตว่างไหม ถ้าใช่ทำต่อไป
% ขั้นต่อไปคือการ add to the table
% output (outputindex) = getcodefor (substr,table) =getcodefor (227 , table)
% ไปดูที่ getcodefor (substr , table)
```

% code = uint16 [] กำหนดให้ code มีค่าว่าง

% if length (substr) = 1 ดูจำนวนข้อมูลว่ามี 1 ข้อมูลหรือไม่ แต่ที่นี้คือ มี 1 ข้อมูล คือ 227

% code = substr = 227

% output (outputindex) = output (1) = 227

% [table , code] = addcode (table , [substr element]) ในที่นี้คือ addcode (table , [227 226])

% ไปดูฟังก์ชั่น addcode

% function [table , code] = addcode (table , substr) เป็นฟังก์ชั่น เพิ่ม code ลงไป

% code = length(table) + 1 คือการเอา code เท่ากับ length ในที่นี้คือ 256+1

% table {257} = 227 226 เป็นการเพิ่ม code เข้าไปที่ที่ตำแหน่งของตาราง

% code = uint16 (256) กำหนดให้กลับมาอยู่ที่ตำแหน่ง 256 เพราะตำแหน่งเริ่มจาก 0 ไม่ใช่ 1 ไม่

%เหมือนความยาวของ table ที่เริ่มจาก 1 ไม่ใช่ 0

% ออกจากลูป กลับไปยังที่เดิม

% outputindex = outputindex + 1 เพิ่มขึ้นจากเดิม 1 + 1 เท่ากับ 2

% startindex = index มีค่าเท่ากับ index เท่ากับ 2

% ให้ index = 3

% % ให้ค่า element = vector(3) = 225

% ให้ค่า substr = vector (2 : 2) = 226

% code = getcodefor ([substr element] , table) = ([226 225] , table)

% ไปดูที่ getcodefor (substr , table)

% code = uint16 [] กำหนดให้ code มีค่าว่าง

% if length (substr) = 1 ดูจำนวนข้อมูลว่ามี 1 ข้อมูลหรือไม่ แต่ที่นี้คือ มี 2 ข้อมูล ทำคำสั่งต่อไป

% for index = 257 : length (table) ให้ค่า index = 257 ไปถึง length (table)

% if isequal (substr , table{257} ซึ่งไม่ใช่ จึงออกจากลูปของฟังก์ชั่น getcoder

% if isempty (code) ดูว่า code คือ เซตว่างไหม ถ้าใช่ทำต่อไป

% ขั้นต่อไปคือการ add to the table

% output (outputindex) = getcodefor (substr,table) =getcodefor (226 , table)

% ไปดูที่ getcodefor (substr , table)

% code = uint16 [] กำหนดให้ code มีค่าว่าง

% if length (substr) = 1 ดูจำนวนข้อมูลว่ามี 1 ข้อมูลหรือไม่ แต่ที่นี้คือ มี 1 ข้อมูล คือ 226

% code = substr = 226

% output (outputindex) = output (2) = 226

% [table , code] = addcode (table , [substr element]) ในที่นี้คือ addcode (table , [226 225])

% ไปดูฟังก์ชั่น addcode

% function [table , code] = addcode (table , substr) เป็นฟังก์ชั่น เพิ่ม code ลงไป

% code = length(table) + 1 คือการเอา code เท่ากับ length ในที่นี้คือ 257+1

% table {258} = 226 225 เป็นการเพิ่ม code เข้าไปที่ที่ตำแหน่งของตาราง

% code = uint16 (257) กำหนดให้กลับมาอยู่ที่ตำแหน่ง 257 เพราะตำแหน่งเริ่มจาก 0 ไม่ใช่ 1 ไม่

%เหมือนความยาวของ table ที่เริ่มจาก 1 ไม่ใช่ 0

% ออกจากลูป กลับไปยังที่เดิม

% outputindex = outputindex + 1 เพิ่มขึ้นจากเดิม 2 + 1 เท่ากับ 3

% startindex = index มีค่าเท่ากับ index เท่ากับ 4

% % ให้ค่า element = vector(4) = 227

% ให้ค่า substr = vector (3 : 3) = 225

% code = getcodefor ([substr element] , table) = ([225 227] , table)

% ไปดูที่ getcodefor (substr , table)

% code = uint16 [] กำหนดให้ code มีค่าว่าง

% if length (substr) = 1 ดูจำนวนข้อมูลว่ามี 1 ข้อมูลหรือไม่ แต่ที่นี้คือ มี 2 ข้อมูล ทำคำสั่งต่อไป

% for index = 257 : length (table) ให้ค่า index = 257 ไปถึง length (table)

% if isequal (substr , table{257} ซึ่งไม่ใช่ จึงออกจากลูปของฟังก์ชั่น getcoder

% if isempty (code) ดูว่า code คือ เซตว่างไหม ถ้าใช่ทำต่อไป

% ขั้นต่อไปคือการ add to the table

% output (outputindex) = getcodefor (substr,table) =getcodefor (225 , table)

% ไปดูที่ getcodefor (substr , table)

% code = uint16 [] กำหนดให้ code มีค่าว่าง

% if length (substr) = 1 ดูจำนวนข้อมูลว่ามี 1 ข้อมูลหรือไม่ แต่ที่นี้คือ มี 1 ข้อมูล คือ 225

% code = substr = 225

% output (outputindex) = output (3) = 225

% [table , code] = addcode (table , [substr element]) ในที่นี้คือ addcode (table , [225 227])

% ไปดูฟังก์ชั่น addcode

% function [table , code] = addcode (table , substr) เป็นฟังก์ชั่น เพิ่ม code ลงไป

% code = length(table) + 1 คือการเอา code เท่ากับ length ในที่นี้คือ 258+1

% table {259} = 225 227 เป็นการเพิ่ม code เข้าไปที่ที่ตำแหน่งของตาราง

% code = uint16 (258) กำหนดให้กลับมาอยู่ที่ตำแหน่ง 257 เพราะตำแหน่งเริ่มจาก 0 ไม่ใช่ 1 ไม่

%เหมือนความยาวของ table ที่เริ่มจาก 1 ไม่ใช่ 0

```
% ออกจากลูป กลับไปยังที่เดิม
% outputindex = outputindex + 1  เพิ่มขึ้นจากเดิม 3 + 1 เท่ากับ 4
% startindex = index มีค่าเท่ากับ index เท่ากับ 5
% % ให้ค่า element = vector(4) = 226
% ให้ค่า substr = vector ( 4 : 4) = 227
%  code = getcodefor ([substr element] , table) = ([227 226] , table)
% ไปดูที่ getcodefor (substr , table)
% code = uint16 [] กำหนดให้ code มีค่าว่าง
%  if length (substr) = 1 ดูจำนวนข้อมูลว่ามี 1 ข้อมูลหรือไม่ แต่ที่นี้คือ มี 2 ข้อมูล ทำคำสั่งต่อไป
%  for index = 257 : length (table) ให้ค่า index = 257 ไปถึง length (table)
%  if isequal (substr , table{257} ซึ่งใช่
%  คือเท่ากัน (227 226) = (227 226) ของ table
% code = uint16(257-1) = 256
% จึงออกจากลูปของฟังก์ชั่น getcoder
% if isempty (code) ดูว่า code คือ เซตว่างไหม ถ้าใช่ทำต่อไป แต่ตอนนี้ไม่ใช่ เพราะ code ไม่ว่าง
% substr = vector (startindex : index) = (227 226)
% output (outputindex) = getcodefor (subsr , table)
% ไปดูที่ getcodefor (substr , table)
% code = uint16 [] กำหนดให้ code มีค่าว่าง
%  if length (substr) = 1 ดูจำนวนข้อมูลว่ามี 1 ข้อมูลหรือไม่ แต่ที่นี้คือ มี 2 ข้อมูล ทำคำสั่งต่อไป
%  for index = 257 : length (table) ให้ค่า index = 257 ไปถึง length (table)
%  if isequal (substr , table{257} ซึ่งใช่
%  คือเท่ากัน (227 226) = (227 226) ของ table
% code = uint16(257-1) = 256
% output (outputindex) = getcodefor (subsr , table) = output(4) = 257
% output ((outputindex+1):end) = [] ในที่นี้ไม่มี
% ได้ข้อมูลคือ 227 226 225 257
% ต่อไปเป็นขั้นตอนคลายการบีบอัด
% เอาข้อมูลจากการบีบอัดมา คือ227 226 225 257
function [output,table] = lzw2norm(vector)
if ~isa(vector,'uint16'),
        error('input argument must be a uint16 vector')
```

```
end
vector = vector(:)';
table = cell(1,256);        % เป็นการทำให้มี 1 แถว 256 หลัก และแต่ละเซลล์สามารถ
                              บรรจุข้อมูลได้มากกว่า 1 ข้อมูล
for index = 1:256,          % เป็นการวนลูป 256 ลูป
        table{index} = uint16(index-1);
end
output = uint8([]);         %เป็นการกำหนดค่าข้อมูลให้เหมือนเดิมเป็น 256 ข้อมูล
code = vector(1);           % ขั้นตอนต่อไปเป็นขั้นตอนคลายการบีบอัดข้อมูล ซึ่งจะอธิบายไว้อีก หน้า
                            %ต่อไป
output(end+1) = code;
character = code;
for index=2:length(vector),
        element = vector(index);
        if (double(element)+1)>length(table),
                % add it to the table
                string = table{double(code)+1};
                string = [string character];
        else,
                string = table{double(element)+1};
        end
        output = [output uint8(string)];
        character = string(1);
        [table,code] = addcode(table,[table{double(code)+1} character]);
        code = element;
end
% ###############################################
function [table,code] = addcode(table,substr)
code = length(table)+1;  % start from 1
table{code} = substr;
code = uint16(code-1);   % start from 0
```

% คำอธิบายโปรแกรม

% เอาข้อมูลจากการบีบอัดมา คือ 227 226 225 257

% code = vector(1) = 227

% output (end+1) = นำค่า code มาเก็บต่อ output ตัวสุดท้าย

% character = code = 227

% for index = 2 : length (vector)

% element = vector (index) = 226

% if (double (element) + 1) > length (table) ซึ่งมีค่าไม่มากกว่า

% string = table{double (element) + 1} คือได้ table{227} = 226

% output = [output uint8(string)] ได้ค่าเป็น output = [output 226]

% character = string(1) มีค่าคือ 226

% [table,code] = addcode(table,[table{double(code)+1} character]); ไปดูที่ฟังก์ชั่น

% function [table,code] = addcode(table,substr) = addcode(table,227 226)

% code = length(table)+1; code = 257

% table{257} = 227 226;

% code = uint16(code-1); = 256

% code = element = 226

% index = 3

% element = vector (3) = 225

% if (double (225) + 1) > length (table) ซึ่งมีค่าไม่มากกว่า

% string = table{double (element) + 1} คือได้ table{226} = 225

% output = [output uint8(string)] ได้ค่าเป็น output = [output 225]

% character = string(1) มีค่าคือ 225

% [table,code] = addcode(table,[table{double(code)+1} character]); ไปดูที่ฟังก์ชั่น

% function [table,code] = addcode(table,substr) = addcode(table,226 225)

% code = length(table)+1; code = 258

% table{257} = 226 225;

% code = uint16(code-1); = 257

% code = element = 225

% index = 4

% element = vector (4) = 257

% if (double (257) + 1) > length (table) ซึ่งมีค่ามากกว่า

% string = table{double (code) + 1} คือได้ table{257} = 227 226

% string = [string character] คือ [227]

% output = [output uint8(string)] ได้ค่าเป็น output = [output 257]

% character = string(1) มีค่าคือ ไม่มี

% code = element = 257

ประวัติผู้เขียนโครงงาน

ชื่อ นายกมล ธีระกาญจน์

ภูมิลำเนา 231 หมู่ 1 ต.วงฆ้อง อ.พรหมพิราม จ.พิษณุโลก

ประวัติการศึกษา

- จบระดับมัธยมศึกษาจากโรงเรียนจุฬาภรณราชวิทยาลัย พิษณุโลก
- ปัจจุบันกำลังศึกษาอยู่ในระดับปริญญาตรีชั้นปีที่ 4

สาขาวิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์

มหาวิทยาลัยนเรศวร

Email : **ccth2008@hotmail.com**

ชื่อ นายกิตติภูมิ สมศรี

ภูมิลำเนา 141 หมู่.3 ต.บึงพระ อ.เมือง จ.พิษณุโลก

ประวัติการศึกษา

- จบระดับมัธยมศึกษาจากโรงเรียนจุฬาภรณราชวิทยาลัย พิษณุโลก
- ปัจจุบันกำลังศึกษาอยู่ในระดับปริญญาตรีชั้นปีที่ 4

สาขาวิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์

มหาวิทยาลัยนเรศวร

Email : **t_happyness@hotmail.com**