การแก้ไขข้อผิดพลาดบนคิวอาร์โค้ดด้วยรหัสตรวจสอบสภาวะคู่หรือคี่ชนิดความหนาแน่นต่ำ

นางสาวสุมามาลย์ สุขสาคร

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2558

ERROR CORRECTION IN QR CODE USING LOW DENSITY PARITY CHECK

Miss Sumaman Sooksakorn

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Science Program in Computer Science

Department of Computer Engineering

Faculty of Engineering

Chulalongkorn University

Academic Year 2015

หัวข้อวิทยานิพนธ์	การแก้ไขข้อผิดพลาดบนคิวอาร์โค้ดด้วยรหัสตรวจสอบสภาวะคู่หรือคี่ชนิดความหนาแน่นต่ำ
โดย	นางสาวสุมามาลย์ สุขสาคร
สาขาวิชา	วิทยาศาสตร์คอมพิวเตอร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	รองศาสตราจารย์ ดร.สาธิต วงศ์ประทีป

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

.....คณบดีคณะวิศวกรรมศาสตร์
(รองศาสตราจารย์ ดร.สุพจน์ เตชวรสินสกุล)

คณะกรรมการสอบวิทยานิพนธ์

.....ประธานกรรมการ
(ศาสตราจารย์ ดร.ประภาส จงสถิตย์วัฒนา)

.....อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(รองศาสตราจารย์ ดร.สาธิต วงศ์ประทีป)

.....กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.อาทิตย์ ทองทักษ์)

.....กรรมการภายนอกมหาวิทยาลัย
(ดร.มหศักดิ์ เกตุฉ่ำ)

สุมามาลย์ สุขสาคร : การแก้ไขข้อผิดพลาดบนคิวอาร์โค้ดด้วยรหัสตรวจสอบสภาวะคู่หรือคี่ชนิดความหนาแน่นต่ำ (ERROR CORRECTION IN QR CODE USING LOW DENSITY PARITY CHECK) อ.ที่ปรึกษาวิทยานิพนธ์หลัก: รศ. ดร.สาธิต วงศ์ประทีป, 88 หน้า.

ในงานวิจัยนี้ รหัสการแก้ไขข้อผิดพลาดใน Quick Response (QR code) ได้ถูกนำมาศึกษาค้นคว้า รหัสการแก้ไขข้อผิดพลาด Reed Solomon (RS) เป็นรหัสการแก้ไขข้อผิดพลาด ที่ถูกใช้ใน QR code ในปัจจุบัน และยังเป็นรหัสที่ใช้หลากหลายในระบบการสื่อสารแล้วระบบการเก็บข้อมูลดิจิตอล แต่อย่างไรก็ตาม รหัสการแก้ไขข้อผิดพลาด RS มีความซับซ้อนในการเข้ารหัสและถอดรหัส ในงานวิจัยนี้นำเสนอ รหัสการแก้ไขข้อผิดพลาด Low Density Parity Check (LDPC) หรือรหัสการแก้ไขข้อผิดพลาดสภาวะคู่หรือคี่ชนิดความหนาแน่นต่ำ เพื่อประยุกต์ใช้ใน QR code และเปรียบเทียบประสิทธิภาพในแง่ต่างๆ กับ รหัสการแก้ไขข้อผิดพลาด RS

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต .....

สาขาวิชา วิทยาศาสตร์คอมพิวเตอร์ ลายมือชื่อ อ.ที่ปรึกษาหลัก .....

ปีการศึกษา 2558

5571003421 : MAJOR COMPUTER SCIENCE
KEYWORDS: QR CODE / LDPC / LOW DENSITY PARITY CHECK / QUICK RESPONSE

SUMAMAN SOOKSAKORN: ERROR CORRECTION IN QR CODE USING LOW DENSITY PARITY CHECK. ADVISOR: SARTID VONGPRADHIP, 88 pp.

In this research, the error correction section in quick response code (QR code) is investigated. Reed Solomon (RS) is the current standard error correction algorithm in QR and widely used for applications in communication and digital data storage system. However, RS has a high complexity of encoding and decoding. Hence, this paper proposes to use another efficient error correction algorithm, Low Density Parity Check (LDPC). These two performance of algorithms are compared by using RS in QR code and using LDPC in QR Code.

Department: Computer Engineering Student's Signature .....
Field of Study: Computer Science Advisor's Signature .....
Academic Year: 2015

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยดี ขอกราบขอบพระคุณรองศาสตราจารย์ ดร.สาธิต วงศ์ประทีป ซึ่งเป็นอาจารย์ที่ปรึกษาวิทยานิพนธ์ ท่านได้กรุณาสละเวลาและดูแลการวิจัยให้คำแนะนำเกี่ยวกับการทำวิจัย และการสนับสนุนเป็นอย่างดีจนทำให้การวิจัยในครั้งนี้สำเร็จออกมาด้วยดี

ขอกราบขอบพระคุณคณะกรรมการสอบวิทยานิพนธ์ทุกท่านเป็นอย่างสูง ขอขอบพระคุณอาจารย์ประจำภาควิชาวิศวกรรมคอมพิวเตอร์ จุฬาลงกรณ์มหาวิทยาลัย ทุกท่านเป็นอย่างสูงที่ให้ข้อคิดและแนวทางในการวิจัย ขอขอบคุณเจ้าหน้าที่ประจำภาควิชาวิศวกรรมคอมพิวเตอร์ทุกท่าน เพื่อนๆทุกคนในห้องวิจัย Digital System Engineering Laboratory (DSEL)

ขอกราบขอบพระคุณ คุณพ่อ-คุณแม่ และญาติ ๆ ที่ให้การสนับสนุนและเป็นกำลังใจที่ดีเสมอ

สารบัญ

หน้า

บทคัดย่อภาษาไทย ..... ง

บทคัดย่อภาษาอังกฤษ ..... จ

กิตติกรรมประกาศ ..... ฉ

สารบัญ ..... ช

สารบัญตาราง 1

สารบัญภาพ 2

บทที่ 1 6

1.1 ความเป็นมาและความสำคัญของปัญหา 6

1.2 วัตถุประสงค์ของงานวิจัย 6

1.3 ขอบเขตของงานวิจัย 6

1.4 ข้อตกลงเบื้องต้น 6

1.5 ประโยชน์ที่คาดว่าจะได้รับ 7

1.6 ลำดับขั้นตอนในการเสนอผลการวิจัย 7

บทที่ 2 8

2.1 ส่วนประกอบของ QR Code 8

2.1.1 Binary String 8

2.1.2 QR Code 8

2.1.3 ภาษาที่ใช้ในการบรรจุลงใน QR Code มี ดังนี้ 10

2.1.4 การแก้ไขข้อผิดพลาด (Error Correction) 10

2.1.5 เวอร์ชั่น (Version) และขนาด (Sizes) ของ QR Code [4] 12

2.1.6 ความจุข้อมูล (Data capacity) ที่เก็บได้ใน QR Code 12

2.1.7 การเข้ารหัสใน QR Code ในส่วน Data codewords 14

หน้า

2.1.8 การเข้ารหัสใน QR Code ในส่วน Error Correction Codewords (EC Codewords).....16
2.1.9 โครงสร้าง Final Message ของ QR Code (Structure Final Message).....17
2.1.10 การวาง Codeword ในเมตริก (Codeword Placement In Matrix).....19
2.1.11 ขั้นตอนการ Masking ใน QR Code 30
2.1.12 การเข้ารหัสในส่วน Format Information 32
2.1.13 การถอดรหัส QR Code (Decoding Procedure) 33
2.2 การเข้ารหัสช่องสัญญาณ 42
2.3 รหัสแก้ไขข้อผิดพลาด Low Density Parity Check (LDPC) [9] 42
2.4 รหัสแก้ไขข้อผิดพลาด Reed Solomon (RS).....54
2.5 การวัดประสิทธิภาพโดย Bit Error Rate.....56
2.7 สัญญาณรบกวนไวต์เกาส์เซียนแบบบวก (Additive White Gaussian Noise: AWGN) 56
2.8 อัตราส่วนสัญญาณต่อสัญญาณรบกวน (Signal to noise Ratio: SNR).....57
2.9 งานวิจัยที่เกี่ยวข้อง.....57
บทที่ 3 60
3.1 ขั้นตอนการเปรียบเทียบประสิทธิภาพการแก้ไขข้อผิดพลาดโดย RS และ LDPC โดยสุ่มพหุนาม.....61
3.2 ขั้นตอนการเปรียบเทียบประสิทธิภาพการแก้ไขข้อผิดพลาดโดย RS และ LDPC โดยกำหนดสมการพหุนามตามหลักเกณฑ์ของ QR Code เวอร์ชัน 1 Error Correction M68
3.2.1 ขั้นตอนการเตรียมข้อมูลในการสร้างโปรแกรมจำลอง สำหรับ RS 68
3.2.2 ขั้นตอนการเตรียมข้อมูลในการสร้างโปรแกรมจำลอง สำหรับ LDPC 69
3.2.3 สร้างโปรแกรมจำลองในการเข้ารหัสข้อมูล แบบ RS โดยใช้สมการพนุนาม ตามกำหนดของ QR Cod และ LDPC 70

3.2.4 ทำการทดสอบประสิทธิภาพ RS และ LDPC โดย RS ใช้พหุนามตามกำหนดของ QR Code เวอร์ชัน 1 ระดับ Error Correction ที่ M 71

บทที่ 4 72

4.1 ผลการเปรียบเทียบประสิทธิภาพการแก้ไขข้อผิดพลาดโดย RS และ LDPC โดยสุ่มพหุนาม 72

4.2 ผลการเปรียบเทียบประสิทธิภาพการแก้ไขข้อผิดพลาด RS และ LDPC โดยกำหนดสมการพหุนามตามหลักเกณฑ์ของ QR Code 78

บทที่ 5 83

5.1 บทสรุป 83

5.1.1 ผลการเปรียบเทียบประสิทธิภาพการแก้ไขข้อผิดพลาดโดย RS และ LDPC โดยสุ่มพหุนาม.....83

5.1.2 บทสรุปการประยุกต์ใช้ LDPC ใน QR Code.....84

5.1.3 ข้อเสนอแนะ 84

รายการอ้างอิง 85

ประวัติผู้เขียนวิทยานิพนธ์ 88

สารบัญตาราง

ตารางที่ **หน้าที่**

ตารางที่ 1ตัวอย่าง Error correction characteristics สำหรับ QR Code 2005.....11

ตารางที่ 2 ตัวอย่าง Number of symbol characters and input data capacity for QR Code.....13

ตารางที่ 3 การระบุโหมดในการเข้ารหัสของ QR Code.....14

ตารางที่ 4 ตารางการเข้ารหัสและถอดรหัสสำหรับโหมด Alphanumeric.....15

ตารางที่ 5 รูปแบบ Mask.....30

ตารางที่ 6 ตารางการให้คะแนนในการเลือกรูปแบบ Mask.....31

ตารางที่ 7 ตารางตัวชี้วัด Error Correction Level สำหรับ QR Code.....32

ตารางที่ 8 ตารางแสดงค่า Code rate ที่ใช้ในการเปรียบเทียบประสิทธิภาพ.....61

ตารางที่ 9 ตารางแสดงค่า EbNo ที่ใช้เป็นสัญญาณในการรบกวน.....61

ตารางที่ 10 แสดงสมการพนุนามที่ใช้ในการสร้างรหัสแก้ไขข้อผิดพลาดRS ตามจำนวน Error Correction Codewords.....69

ตารางที่ 11 ตารางแสดงค่า E_bN_o ที่ใช้เป็นสัญญาณรบกวนนในการทดลอง 3.2.....71

ตารางที่ 12 แสดง Bit Error Rate ในแต่ละการใส่ค่าสัญญาณรบกวนในการทดสอบรหัส RS และ LDPC โดยสุ่มพหุนาม.....72

ตารางที่ 13 แสดง เวลาที่ใช้ในการทำการแก้ไขข้อผิดพลาด แต่ละการใส่ค่าสัญญาณรบกวนในการทดสอบรหัส RS และ LDPC โดยสุ่มพหุนาม.....73

ตารางที่ 14 แสดง Bit Error Rate ในแต่ละการใส่ค่าสัญญาณรบกวนในการทดสอบรหัส RS และ LDPC ที่มีค่าสมการพนุนามตามหลักเกณฑ์ของ QR Code เวอร์ชัน 1 Error Correction M.....78

ตารางที่ 15 แสดง เวลาที่ใช้ในการทำการแก้ไขข้อผิดพลาด แต่ละการใส่ค่าสัญญาณรบกวนในการทดสอบรหัส RS และ LDPC ที่มีค่าสมการพนุนามตามหลักเกณฑ์ของ QR Code เวอร์ชัน 1 Error Correction M.....79

สารบัญภาพ

ภาพที่ **หน้า**

ภาพที่ 1 แสดงโครงสร้างของ QR CODE [3].....8

ภาพที่ 2 แสดงการวางของ Data และ Error correction codewords **[2]** 9

ภาพที่ 3 แสดงโครงสร้างของ QR Code เวอร์ชั่น 1 เวอร์ชั่น 3 และเวอร์ชั่น 6.....12

ภาพที่ 4 โครงสร้างส่วน Final Message Codeword 18

ภาพที่ 5 การวางบิตในรูปแบบปกติในทิศทางขึ้นและทิศทางลง.....19

ภาพที่ 6 ก.การวางแบบปกติ ข.การวางแบบผิดปกติเมื่อทิศทางในการวางเปลี่ยนทิศทาง.....20

ภาพที่ 7 ตัวอย่างการวางบิตที่ติดกับ Alignment Pattern 21

ภาพที่ 8 แสดงลำดับการวาง Bitstream บน QR Code เวอร์ชัน **1**.....21

ภาพที่ 9 การเรียง Codewords ใน QR Code เวอร์ชัน 1-L.....22

ภาพที่ 10 การเรียง Codewords ใน QR Code เวอร์ชัน 1-M 22

ภาพที่ 11 การเรียง Codewords ใน QR Code เวอร์ชัน 1-Q.....23

ภาพที่ 12 การเรียง Codewords ใน QR Code เวอร์ชัน 1-H.....23

ภาพที่ 13 การเรียง Codewords ใน QR Code เวอร์ชัน 3-L 24

ภาพที่ 14 การเรียง Codewords ใน QR Code เวอร์ชัน 3-M 24

ภาพที่ 15 การเรียง Codewords ใน QR Code เวอร์ชัน 3-Q.....25

ภาพที่ 16 การเรียง Codewords ใน QR Code เวอร์ชัน 3-H.....25

ภาพที่ 17 การเรียง Codewords ใน QR Code เวอร์ชัน 6-L 26

ภาพที่ 18 การเรียง Codewords ใน QR Code เวอร์ชัน 6-M 27

ภาพที่ 19 การเรียง Codewords ใน QR Code เวอร์ชัน 6-Q.....28

ภาพที่ 20 การเรียง Codewords ใน QR Code เวอร์ชัน 6-H.....29

ภาพที่ 21 รูปแบบ Masking ใน QR Code เวอร์ชัน 1 30

ภาพที่ 22 แสดงขั้นตอนการ Mask.....31

ภาพที่ 23 ตำแหน่ง Format Information.....33

ภาพที่ 24 ลำดับขั้นตอนในการถอดรหัสใน QR code.....34

ภาพที่ 25 การ Scan Line ใน Finder Pattern.....35

ภาพที่ 26 Finder Pattern ด้านบน.....36

ภาพที่ 27 Finder Pattern และ Version Information.....37

ภาพที่ 28 Finder Pattern และ Alignment Patterns.....38

ภาพที่ 29 จุดศูนย์กลางของ Alignment Pattern.....39

ภาพที่ 30 พื้นที่ซ้ายบนของ QR Code.....40

ภาพที่ 31 พื้นที่ขวาล่างของ QR Code.....41

ภาพที่ 32 ตัวอย่าง parity-check matrix H.....43

ภาพที่ 33 กราฟแทนเนอร์ที่สอดคล้องกับ parity-check matrix ภาพที่ 32.....44

ภาพที่ 34 ตัวอย่าง parity-check matrix H ในรูป row-echelon form.....44

ภาพที่ 35 ตัวอย่าง parity-check matrix H ในรูป reduced row-echelon form.....45

ภาพที่ 36 ตัวอย่าง parity-check matrix H หลังจากการทำ column permutation.....45

ภาพที่ 37 ตัวอย่าง parity-check matrix H_D.....45

ภาพที่ 38 Matrix ตัวสร้างรหัส G.....46

ภาพที่ 39 กราฟแสดงการคำนวณ $r33(0)$.....50

ภาพที่ 40 กราฟแสดงการคำนวณ $q33(1)$.....52

ภาพที่ 41 แสดงตัวอย่างคู่ของ DMC ที่มีการถูกทำให้เสียหาย.....58

ภาพที่ 42 ตารางเปรียบเทียบความสามารถในการกู้ข้อมูลจาก DMC 23 คู่.....58

ภาพที่ 43 กระบวนการสร้าง QR Code โดยใช้ RS กับ LDPC สำหรับการแก้ไขข้อผิดพลาด...60

ภาพที่ 44 แสดงการใส่ค่า Code rate ที่ใช้ในการทดลอง.....64

ภาพที่ 45 แสดงค่า Codeword หลังจากการเข้ารหัสโดย LDPC.....64

ภาพที่ 46 แสดงค่า Codeword หลังจากการเข้ารหัสโดย RS 65

ภาพที่ 47 แสดงการจำลองใส่สัญญาณรบกวนลงใน Codewords.....65

ภาพที่ 48 แสดงการใส่ค่าสัญญาณรบกวนเพื่อใช้ในการทดสอบ 66

ภาพที่ 49 แสดงค่า Codewords หลังจากทำการถูกแก้ไข้ข้อผิดพลาดโดย LDPC.....67

ภาพที่ 50 แสดงค่า Codewords หลังจากทำการถูกแก้ไข้ข้อผิดพลาดโดย RS.....67

ภาพที่ 51 ตัวอย่างการแสดงผลค่า BER และ เวลาที่ใช้ในการเข้ารหัสและถอดรหัส 68

ภาพที่ 52 แสดงตัวอย่างการเข้ารหัส RS โดยสมการพหุนามกำหนดโดย QR Code.....70

ภาพที่ 53 แสดงตัวอย่างการเข้ารหัส LDPC ตามกำหนดจำนวน Error Correction Codewords ของ QR Code 70

ภาพที่ 54 แสดงตัวอย่าง Error Correction Codeword ที่ได้จากการเข้ารหัสโดย RS.....71

ภาพที่ 55 กราฟแสดงค่า BER & EbNo ในการแก้ไขข้อผิดพลาดที่ Code rate = (19/26).....75

ภาพที่ 56 กราฟแสดงค่า BER & EbNo ในการแก้ไขข้อผิดพลาดที่ Code rate = (16/26).....75

ภาพที่ 57 กราฟแสดงค่า BER & EbNo ในการแก้ไขข้อผิดพลาดที่ Code rate = (13/26).....76

ภาพที่ 58 กราฟแสดงค่า BER & EbNo ในการแก้ไขข้อผิดพลาดที่ Code rate = (9/26).....76

ภาพที่ 59 กราฟแสดงเวลาในการแก้ไขข้อผิดพลาดโดย RS และ LDPC มีค่าสมการพหุนามที่คงที่ Code rate = (19/26) 77

ภาพที่ 60 กราฟแสดงเวลาในการแก้ไขข้อผิดพลาดโดย RS และ LDPC มีค่าสมการพหุนามที่คงที่ Code rate = (16/26) 77

ภาพที่ 61 กราฟแสดงเวลาในการแก้ไขข้อผิดพลาดโดย RS และ LDPC มีค่าสมการพหุนามที่คงที่ Code rate = (13/26) 78

ภาพที่ 62 กราฟแสดงเวลาในการแก้ไขข้อผิดพลาดโดย RS และ LDPC มีค่าสมการพหุนามที่คงที่ Code rate = (9/26).....78

ภาพที่ 63 กราฟแสดงค่า BER & EbNo ในการแก้ไขข้อผิดพลาดโดยใช้สมการพหุนามคงที่ Code rate = (16/26).....80

ภาพที่ 64 กราฟแสดงเวลาที่ใช้ในการแก้ไขข้อผิดพลาดโดยใช้สมการพหุนามคงที่ Code rate = (16/26).....80

ภาพที่ 65 กราฟแสดงค่า BER ฿ EbNo ของ LDPC ในการใช้ Matrix H เพื่อเข้ารหัสที่ต่างกัน81

ภาพที่ 66 Bitstream ของข้อความก่อนเข้ารหัส LDPC.....81

ภาพที่ 67 Bitstream ของข้อความหลังจากทำการแก้ไขข้อผิดพลาดที่เกิดจากการส่งสัญญาณ EbNo = 30 โดยรหัส LDPC 82

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

เนื่องจาก QR CODE อาจถูกทำลายจากสภาวะแวดล้อมต่างๆ ดังนั้น การเข้ารหัสเพื่อแก้ไขความผิดพลาดหรือการทำ Error Correction จึงมีความสำคัญ และถูกนำมาใช้งานใน QR CODE ซึ่งปัจจุบันนี้รหัสแก้ไขความผิดพลาดที่ใช้ก็คือ รหัสรีดโซโลมอน (Reed Solomon code) [1] ซึ่งเป็นรหัสแก้ไขความผิดพลาดที่มีประสิทธิภาพสูง แต่ในปัจจุบัน รหัสรีดโซโลมอนกำลังถูกแทนที่ด้วย รหัสการแก้ไขข้อผิดพลาดที่กำลังนิยมคือ LDPC (Low Density Parity Check) เช่น ในเทคโนโลยีการผลิตฮาร์ดไดรฟ์ต้องการความจุของข้อมูลที่มากขึ้นและส่งผลให้รหัสแก้ไขความผิดพลาดจำเป็นต้องมีอัตราการเข้ารหัสสูง โดยที่รหัสรีดโซโลมอนอาจจะมีศักยภาพในการแก้ไขความผิดพลาดได้ไม่เพียงพอ ดังนั้น รหัส LDPC หรือ รหัสตรวจสอบสภาวะคู่หรือคี่ชนิดความหนาแน่นต่ำ จึงถูกนำมาประยุกต์ใช้ [2]

ในงานวิจัยชิ้นนี้จะเป็นการศึกษารหัส LDPC เพื่อนำมาประยุกต์ใช้กับการทำ Error Correction ใน QR CODE โดยนำมาแทนที่ รหัสรีดโซโลมอน แล้วเปรียบเทียบประสิทธิภาพในการทำ Error Correction ของทั้งสองรหัสด้วย QR CODE ที่มีการถูกทำให้เสียหาย เหมือนกันในหลาย ๆ รูปแบบ

1.2 วัตถุประสงค์ของงานวิจัย

ศึกษาและประยุกต์ใช้รหัสการแก้ไขความผิดพลาดแบบ LDPC ใน QR CODE เพื่อเพิ่มประสิทธิภาพในการกู้ข้อมูลกลับคืน

1.3 ขอบเขตของงานวิจัย

1. เป็นการศึกษาการนำรหัสการแก้ไขความผิดพลาดแบบ LDPC มาใช้ในการทำ Error Correction ใน QR CODE
2. เปรียบเทียบประสิทธิภาพรหัสการแก้ไขข้อผิดพลาดแบบ RS และ LDPC
3. หาวิธีการใช้รหัสแก้ไขความผิดพลาดแบบ LDPC ใน QR Code
4. เปรียบเทียบความเร็วในการเข้ารหัสของทั้งสอง รหัสแก้ไขข้อผิดพลาด

1.4 ข้อตกลงเบื้องต้น

1. งานวิจัยจะใช้ MATLAB โปรแกรมสำเร็จรูปเพื่อทดสอบประสิทธิภาพของรหัสการแก้ไขข้อผิดพลาดแบบ RS และ LDPC

2. งานวิจัยจะใช้ หลักเกณฑ์ QR Code เวอร์ชัน 1 มาใช้ในการทดสอบประสิทธิภาพของรหัส RS และ LDPC
3. งานวิจัยนี้ศึกษาในส่วนของการทำการแก้ไขข้อผิดพลาดใน QR Code และปรับใช้รหัส LDPC ให้เหมาะกับ QR Code เวอร์ชัน 1 ระดับ Error correction M

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. ได้ศึกษาการแก้ไขข้อผิดพลาดบน QR Code โดยรหัสการแก้ไขข้อผิดพลาดแบบ RS
2. ได้ประยุกต์ใช้รหัสการแก้ไขข้อผิดพลาดแบบ LDPC บน QR Code
3. เป็นแนวทางการวิจัยด้านการพัฒนา Error Correction ใน QR Code โดยใช้รหัสการแก้ไขความผิดพลาดแบบ LDPC ซึ่งกำลังเป็นที่สนใจมากในขณะนี้
4. เพิ่มความสามารถในการถูกทำลายหรือถูกทำให้เสียหายบน QR Code

1.6 ลำดับขั้นตอนในการเสนอผลการวิจัย

1. ศึกษาทฤษฎี หลักการพื้นฐานที่ใช้ในการวิจัยและงานวิจัยที่เกี่ยวข้อง
2. ศึกษาเทคนิคต่างๆที่มีอยู่ถึงแนวคิด หลักการ พร้อมทั้งศึกษาข้อดีและข้อบกพร่องของแต่ละเทคนิคที่อยู่ในขอบเขต
3. ออกแบบและนำเสนอวิธีการใช้รหัสการแก้ไขข้อผิดพลาดแบบ LDPC เพื่อใช้ในการกู้ข้อมูลในคิวอาร์โค้ด
4. ทดสอบวิธีการที่นำเสนอ
5. วิเคราะห์ผลการทดลอง
6. สรุปผลและเรียบเรียงวิทยานิพนธ์

บทที่ 2

ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

ในส่วนนี้เป็นแนวคิดและทฤษฎีที่เกี่ยวกับ QR Code และการแก้ไขข้อบกพร่องในระบบดิจิตอล ซึ่งจะกล่าวถึงข้อมูลทั่วไปเกี่ยวกับบาร์โค้ดสองมิติ โครงสร้างส่วนประกอบของ QR Code และขั้นตอนการ

2.1 ส่วนประกอบของ QR Code

2.1.1 Binary String

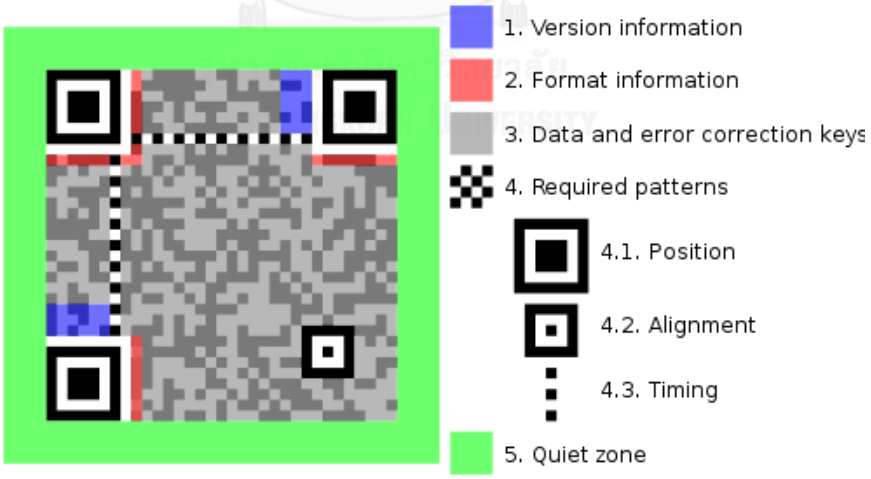
เป็นสายข้อมูลจะอยู่ในรูปของ Bit (บิต) ซึ่งเป็นหน่วยที่เล็กที่สุด ของข้อมูลในระบบคอมพิวเตอร์ มีค่าเป็นเลขฐานสองเดี่ยว คือ บิต 0 หรือ บิต 1 Data Character ทั่วไปจะสามารถถูกเปลี่ยนมาอยู่ในรูปแบบของ Bit ได้

2.1.2 QR Code

1) โมดูล

โมดูลคือ สี่เหลี่ยมสี่ดำและสีขาว ที่เห็นบน QR CODE การอ้างตำแหน่งใน QR CODE จะอ้างอิงเป็นแถวและคอลัมน์ โดยจะใช้ I แทนแถว และใช้ J แทนคอลัมน์ นับจากซ้ายไปขวาจะนับเริ่มต้นที่ 0 เช่น โมดูล (0,0) จะอยู่มุมบนซ้ายของ QR CODE โดย บิต 1 เป็นลักษณะของโมดูลสีดำ และบิต 0 เป็นลักษณะของโมดูลสีขาว

2) โครงสร้าง QR Code



ภาพที่ 1 แสดงโครงสร้างของ QR CODE [3]

1. Version information

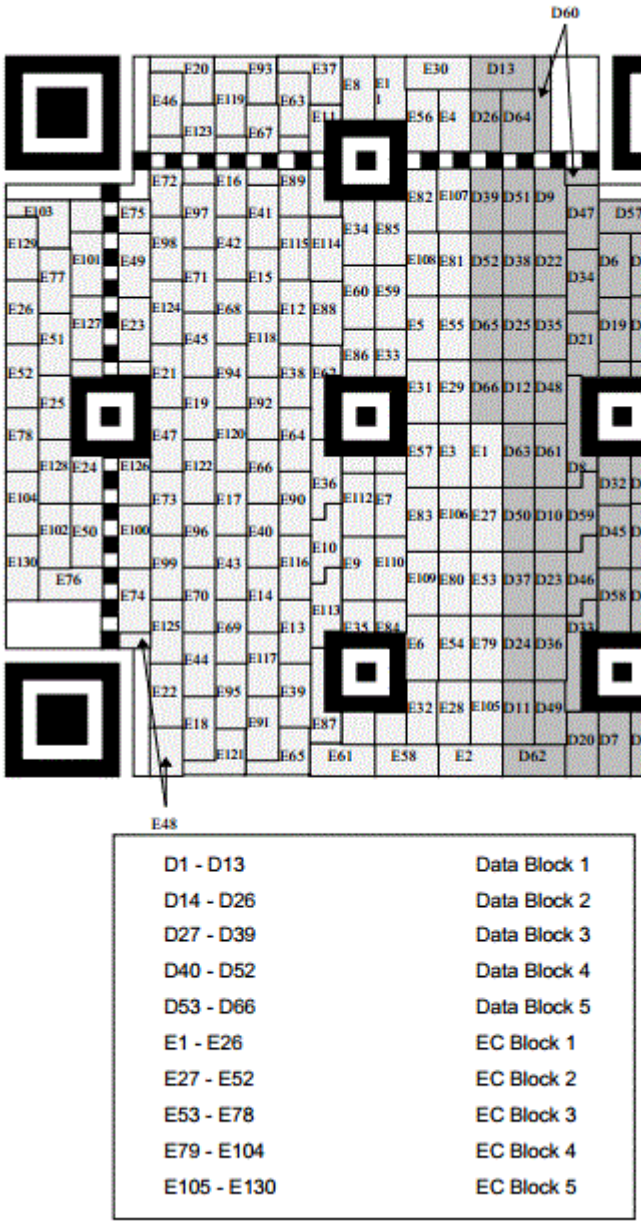
อยู่ตรงพื้นที่ในส่วนสีฟ้า มีหน้าที่บอกเวอร์ชันของ QR CODE ว่าเป็น เวอร์ชันใด

2. Format information

อยู่ตรงพื้นที่ในส่วนสีชมพู ประกอบด้วยส่วนการแก้ไขข้อผิดพลาดและรูปแบบ Mask ของ QR CODE ซึ่งจะเป็นส่วนแรกที่อ่านเมื่อมีการถอดรหัส QR CODE

3. Data and error correction keys

เป็นพื้นที่ส่วนสีเทา ซึ่งใช้ในการเก็บข้อมูลและส่วนที่แก้ไขข้อผิดพลาด มีลักษณะเป็นอะเรย์ มีแถวและคอลัมน์เป็น โมดูลแต่ละโมดูลจะถูกเก็บเป็นเลขฐานสอง 0 และ 1 โดยส่วนนี้จะเป็นการเรียงต่อกันโดยเริ่มจาก Data Codewords ตัวอย่างการวาง data and error correction codeword แสดงดังภาพ ที่ 2



ภาพที่ 2 *แสดงการวางของ Data และ Error correction codewords* ***[2]***

ภาพที่ 2 แสดง QR CODE เวอร์ชัน 7 ที่ Error Correction H โดยแสดงการวาง Data Codewords (D1-D66) และ Error Correction Codewords (E1-E103)

4. Required patterns

เป็นส่วนที่ใช้จับตำแหน่งพร้อมทั้งปรับรูปร่างของ QR CODE ให้พร้อมสำหรับ การถอดรหัสข้อมูล ซึ่งประกอบด้วยส่วนย่อยดังนี้

4.1 Position จะประกอบด้วย Position Detection Patterns ที่เหมือนกัน 3 มุม ใน QR CODE ซึ่งมี มุมบนขวา มุมบนซ้าย มุมล่างซ้าย ประโยชน์คือ มีไว้สำหรับทำให้สามารถอ่านได้ 360 องศา รอบทิศทาง และสามารถอ่านได้ด้วยความเร็วสูง

4.2 Alignment จะเริ่มต้นมีในเวอร์ชัน 2 ขึ้นไป เริ่มแรกจะอยู่ตำแหน่งขวาล่าง มีหน้าที่แก้ไขการบิดเบือนเมื่อ QR CODE โค้งหรืองอ และจำนวน Alignment จะมีมากขึ้น เมื่อเวอร์ชันสูงขึ้น

4.3 Timing เป็นโมดูลสีขาวสีดำสลับกัน ใช้ตรวจสอบพิกัด Timing จะวางอยู่ระหว่าง Position

5. Quiet zone

จะเป็นบริเวณที่ว่างเปล่าปราศจากเครื่องหมายใดๆ โดยจะอยู่รอบทั้ง 4 ด้านของ QR Code โดยจะมีลักษณะเป็นพื้นที่สว่าง (Light) สำหรับ QR Code ที่มีความกว้างด้านละ 4X

2.1.3 ภาษาที่ใช้ในการบรรจุลงใน QR Code มี ดังนี้

Numeric data (0-9) เก็บข้อมูลที่เป็นเฉพาะตัวเลขเท่านั้น ตามทฤษฎีจะสามารถบรรจุข้อมูลได้สูงสุด 7,089 ตัว หรือน้อยกว่า

Alphanumeric data (0-9, A-Z, $%*+-./:) เก็บข้อมูลที่เป็นตัวเลข ตัวอักษรภาษาอังกฤษและอักขระพิเศษ ตามทฤษฎีจะสามารถบรรจุข้อมูลได้สูงสุด 4,296 ตัว หรือน้อยกว่า

8-bit byte data เก็บข้อมูลประเภทไบนารี 8 บิต ตามทฤษฎีจะสามารถบรรจุข้อมูลได้สูงสุด 2,953 ตัว หรือน้อยกว่า

Kanji data เก็บข้อมูลประเภท ตัวอักษรคันจิ/คะนะ ตามทฤษฎีจะสามารถบรรจุข้อมูลได้สูงสุด 1,817 ตัว หรือน้อยกว่า

2.1.4 การแก้ไขข้อผิดพลาด (Error Correction)

คือการแก้ไขความผิดพลาด เมื่อมีการสูญเสียข้อมูล ซึ่งอาจมีสัญญาณรบกวนที่มีผลต่อการอ่านค่าในโมดูลนั้น ในปัจจุบัน QR Code ใช้ Reed Solomon Code ในการทำ Error Correction ซึ่งเป็นการเข้ารหัสแบบ 8 บิต ถ้าข้อมูลนั้นหายไป 1 บิต ข้อมูลที่เหลืออีก 7 บิตจะสูญเสียไปด้วย การแก้ไขข้อผิดพลาดใน QR Code จะสามารถทำได้ 4 ระดับ ดังนี้

Level L คือ ความผิดพลาด 7% หรือ น้อยกว่า จะสามารถกู้ข้อมูลกลับมาได้

Level M คือ ความผิดพลาด 15% หรือ น้อยกว่า จะสามารถกู้ข้อมูลกลับมาได้

Level Q คือ ความผิดพลาด 25% หรือ น้อยกว่า จะสามารถกู้ข้อมูลกลับมาได้

Level H คือ ความผิดพลาด 30% หรือ น้อยกว่า จะสามารถกู้ข้อมูลกลับมาได้

การคำนวณหาค่า Error Correction จะคำนวนในส่วนของการวาง Data Codewords และ Error Correction Codewords เปอร์เซ็นต์ของ Error Correction จะหาได้โดย

$$= \frac{\text{จำนวน Codeword ที่สูญเสีย}}{(\text{Data Codewords} + \text{Error Correction Codewords})} x100 \quad (2.1)$$

ตารางที่ 1ตัวอย่าง *Error correction characteristics* สำหรับ *QR Code 2005*

Version	Total number of codewords	Error correction level	Number of error correction codewords	Number of error correction blocks	Error correction code per block (c, k, r)
1	26	L	7	1	(26, 19, 2)
		M	10	1	(26, 16, 4)
		H	13	1	(26, 13, 6)
		Q	17	1	(26, 9, 8)

ตัวอย่าง จากตารางที่ 1 แสดง QR Code เวอร์ชั่น 1-L โดย ความหมาย ของ Error correction code per block คือ

(c, k, r) c = จำนวน Data Codewords + จำนวน Error correction codewords เท่ากับ 26

k = จำนวน Data Codewords เท่ากับ 19

r = จำนวน Codewords ที่สูญเสียแล้วยังทำให้ QR Code เวอร์ชั่น 1-L สามารถ อ่านได้ปกติ เท่ากับ 2

โดยสามารถคิดเป็นเปอร์เซ็นได้เท่ากับ (2/26)x100 = 7.69%

ในทำนองเดียวกัน

จำนวน Codewords ที่สูญเสียแล้วยังทำให้ QR Code เวอร์ชั่น 1-M สามารถอ่านได้เท่ากับ 4 สามารถคิดเป็นเปอร์เซ็นต์ได้เท่ากับ (4/26)x100 = 15.38%

จำนวน Codewords ที่สูญเสียแล้วยังทำให้ QR Code เวอร์ชั่น 1-Q สามารถอ่านได้เท่ากับ 6 สามารถคิดเป็นเปอร์เซ็นต์ได้เท่ากับ (6/26)x100 = 23.07%

จำนวน Codewords ที่สูญเสียแล้วยังทำให้ QR Code เวอร์ชั่น 1-H สามารถอ่านได้เท่ากับ 8 สามารถคิดเป็นเปอร์เซ็นต์ได้เท่ากับ (8/26)x100 = 30.76%

ค่า Error correction code per block ของทุกเวอร์ชั่นใน QR Code สามารถหาได้จาก ภาคผนวก ก-2 Error correction characteristics for QR Code 2005

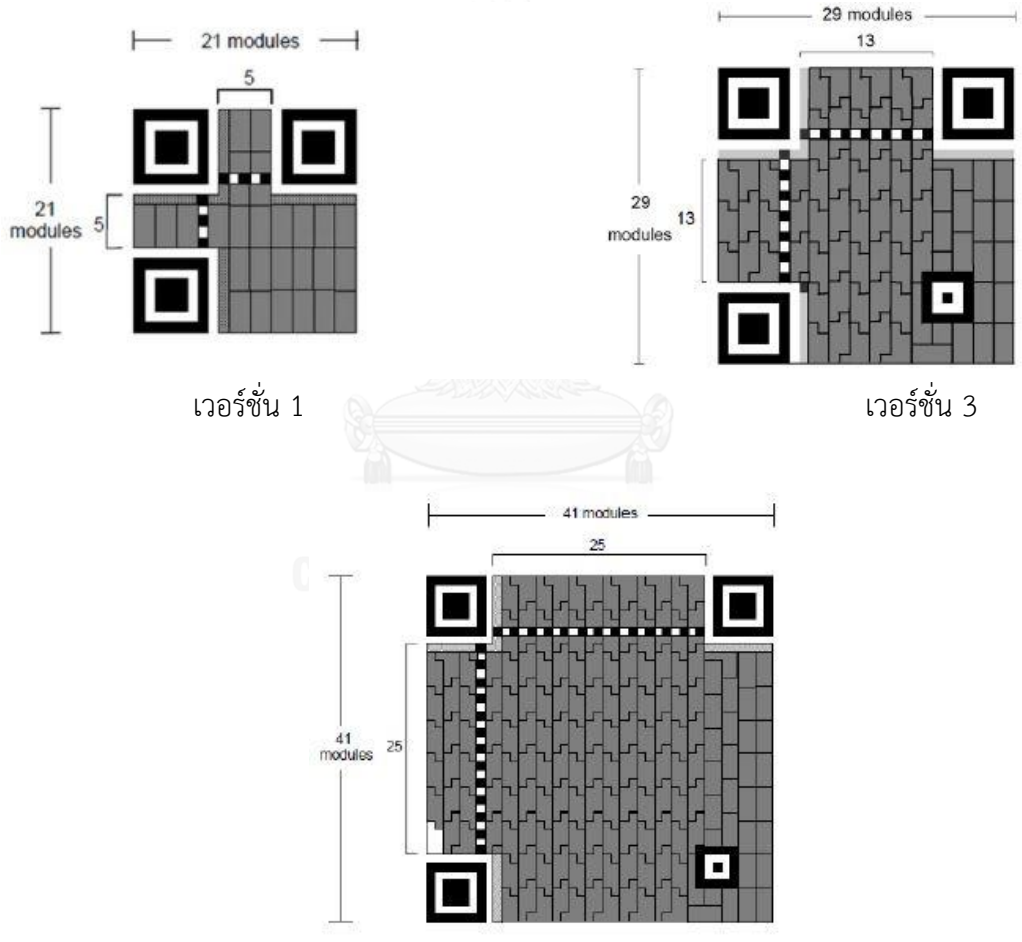
2.1.5 เวอร์ชั่น (Version) และขนาด (Sizes) ของ QR Code [4]

จำนวนเวอร์ชั่นของ QR Code มีตั้งแต่เวอร์ชั่น 1 – 40 โดยเวอร์ชั่น 1 มีขนาด 21 x 21 โมดูล เมื่อเพิ่มเวอร์ชั่นขึ้น 1 เวอร์ชั่น จะเพิ่มโมดูลทั้งด้านกว้างและด้านยาวอีกอย่างละ 4 โมดูล ซึ่งเป็นไปตามสมการ 2.2 โดย V แทน เวอร์ชั่นของ QR Code

จำนวนโมดูล QR Code = (21 + ((V-1) x 4)) x (21 + ((V – 1) x 4)) โมดูล (2.2)

ดังนั้นเวอร์ชั่น 40 = {(21 + (39 x 4)) x ((21 + (39 x 4))} = 177 x 177 โมดูล

จากภาพที่ 3 แสดงโครงสร้างของ QR Code เวอร์ชั่น 1 เวอร์ชั่น 3 และ เวอร์ชั่น 6



เวอร์ชั่น 1 เวอร์ชั่น 3

เวอร์ชั่น 6

ภาพที่ 3 แสดงโครงสร้างของ *QR Code* เวอร์ชั่น *1* เวอร์ชั่น *3* และเวอร์ชั่น *6*

2.1.6 ความจุข้อมูล (Data capacity) ที่เก็บได้ใน QR Code

จำนวนตัวอักษร (Characters) ที่สามารถเก็บได้สูงสุดจะขึ้นอยู่กับเวอร์ชั่นและโหมดของภาษาที่เข้ารหัส ดังแสดงในตาราง ที่ 2

ตารางที่ 2 *ตัวอย่าง Number of symbol characters and input data capacity for QR Code*

Version	Error correction Level	Number of data codewords	Number of data bits	Data capacity			
				Numeric	Alphanumeric	8-bit	Kanji
1	L	19	152	41	25	17	10
	M	16	128	34	20	14	8
	Q	13	104	27	16	11	7
	H	9	72	17	10	7	4
2	L	34	272	77	47	32	20
	M	28	224	63	38	26	16
	Q	22	176	48	29	20	12
	H	16	128	34	20	14	8
3	L	55	440	127	77	53	32
	M	44	352	101	61	42	26
	Q	34	272	77	47	32	20
	H	26	208	58	35	24	15
4	L	80	640	187	114	78	48
	M	64	512	149	90	62	38
	Q	48	384	111	67	46	28
	H	36	288	82	50	34	21
5	L	108	864	255	154	106	65
	M	86	688	202	122	84	52
	Q	62	496	144	87	60	37
	H	46	368	106	64	44	27
6	L	136	1088	322	195	134	82
	M	108	864	255	154	106	65
	Q	76	608	178	108	74	45

	H	60	480	139	84	58	36

จากตัวอย่าง ถ้าต้องการเข้ารหัสตัวอักษรรูปแบบ Alphanumeric data จำนวน Data capacity มากกว่า 65 ตัวอักษร จากตารางที่ 2 ต้องเข้ารหัสในเวอร์ชั่น 6 ในระดับ Error correction L ขึ้นไป เป็นต้น

2.1.7 การเข้ารหัสใน QR Code ในส่วน Data codewords

การเข้ารหัส QR Code ในส่วน Data Codewords เป็นส่วนประกอบหนึ่งใน Encoding Region ซึ่งขั้นตอนแรกในการสร้าง QR Code คือ การสร้าง Bitstream ที่เป็นการเปลี่ยนข้อมูล (Data) มาเป็น Binary และมีการใส่ข้อมูลที่เกี่ยวกับโหมด ของรูปแบบภาษาที่ QR Code เก็บข้อมูล รวมถึงความยาวของข้อมูล โดยจะสาธิตการเข้ารหัส “SUMAMA” ที่ Error correction เป็น ระดับ L (ความผิดพลาด 7%)

1. การเข้ารหัสเพื่อระบุโหมด

โหมดนี้จะใช้ Bitstream จำนวน 4 บิตในการเลือกรูปแบบ ภาษาที่ QR Code ใช้เก็บข้อมูลนั้น โดยจะมีการแบ่งตามลักษณะดังนี้

1) โหมด Numeric จะใช้ Bitstream เป็น 0001
2) โหมด Alphanumeric จะใช้ Bitstream เป็น 0010
3) โหมด 8-bit byte จะใช้ Bitstream เป็น 0100
4) โหมด Kanji จะใช้ Bitstream เป็น 1000

โหมดนี้จะใช้ Bitstream จำนวน 4 บิตในการเลือกรูปแบบ ภาษาที่ QR Code ใช้เก็บข้อมูล ดังนั้นจะสามารถเริ่มต้น Bitstream ที่ “0010”

2. การเข้ารหัสความยาวของข้อมูล

ขั้นตอนนี้เป็นการบอกถึงความยาวของข้อมูลที่เข้ารหัสใน QR Code โดยความยาวของ Bitstream จะขึ้นอยู่กับเวอร์ชั่นของ QR Code และรูปแบบของภาษาดังตารางที่ 3

ตารางที่ 3 การระบุโหมดในการเข้ารหัสของ *QR Code*

เวอร์ชั่นของ QR Code	โหมด			
	Numeric (บิต)	Alphanumeric (บิต)	8-bit byte (บิต)	Kanji (บิต)
1-9	10	9	8	8
10-26	12	11	16	10
27-40	14	13	16	12

จากการเข้ารหัสคำว่า “SUMAMA” มีทั้งหมด 6 ตัวอักษรสามารถแปลงเป็น เลขฐานสองได้เป็น 110 เมื่อทำการเข้ารหัส QR Code เวอร์ชั่น 1 โดยโหมดที่ใช้เป็นโหมด Alphanumeric จากข้อมูลข้างต้น ทำให้ต้องเข้ารหัสเป็น Binary String จำนวน 9 บิต “000000110” ดังนั้นเมื่อรวม Binary String จากเริ่มแรกจะได้ “0010 000000110”

3. การแปลงข้อมูลเป็น Binary String

1) ทำการแบ่งคำที่ต้องการเข้ารหัสออกเป็นคู่ จากตัวอย่างจะได้เป็น “SU, MA, MA”

2) ทำการแปลงตัวอักษรเป็นรหัส ตามตารางที่ 4 ดังนี้

ตารางที่ 4 ตารางการเข้ารหัสและถอดรหัสสำหรับโหมด *Alphanumeric*

Char	Value	Char	Value
0	0	O	24
1	1	P	25
2	2	Q	26
3	3	R	27
4	4	S	28
5	5	T	29
6	6	U	30
7	7	V	31
8	8	W	32
9	9	X	33
A	10	Y	34
B	11	Z	35
C	12	SP	36
D	13	$	37
E	14	%	38
F	15	*	39
G	16	+	40
H	17	-	41
I	18	/	42
J	19	:	43
K	20		44
L	21		
M	22		

N	23		

โดยนำอักษรที่แปลงจากตารางที่ 4 ค่าอักษรตัวแรกคูณกับ 45 แล้วนำมาบวกกับ อักษรที่แปลงจากตารางที่ 4 ตัวที่สอง เมื่อคำนวณเสร็จให้แปลงเป็น Bitstream จำนวน 11 บิต

3) ถ้าตัวอักษรที่อยู่สุดท้ายไม่มีคู่ ให้เปลี่ยนแปลงรหัส ตามตารางรูปที่ 2-4 แล้วแปลงเป็น Bitstream จำนวน 6 บิต จากขั้นตอนทั้งหมดสามารถแปลงตัวอย่างสาธิตได้ดังนี้

SU = (45*28)+30 = 1290 = 10100001010

MA = (45*22)+10 = 1000 = 1111101000

MA = (45*22)+10 = 1000 = 1111101000

เมื่อเรียง Bitstream ทั้งหมดถึงตอนนี้จะได้เป็น

“0010 000000110 10100001010 1111101000 1111101000”

4) การใส่บิตปิดท้าย (Terminator) เป็นการใส่บิตปิดท้ายของ Bitstream เป็นการแสดงว่าข้อมูลนั้นสิ้นสุดโดยให้ใส่ 0000 เพื่อปิดท้าย เมื่อเรียง Bitstream ทั้งหมดถึงตอนนี้ จะได้เป็น

“0010 000000110 10100001010 1111101000 1111101000 0000”

5) ทำการจัดเรียงให้เป็น 8 บิต เติม 0 ถ้าจำเป็น ทำการจัดกลุ่มโดยจัดให้เป็น กลุ่มละ 8 บิต ถ้าชุดสุดท้ายไม่เต็ม ให้เติม 0 จนครบ เมื่อเรียง Bitstream ทั้งหมดถึงตอนนี้จะได้เป็น

“00100000 00110101 00001010 11111010 00111110 10000000”

6) การใส่ Pad Characters เพื่อเพิ่มความยาว Bitstream ถ้า Bitstream มี ความยาวสั้นเกินกว่า Data Codewords ที่กำหนดไว้ในแต่ละเวอร์ชั่น ขั้นตอนนี้เป็นการเติม Bitstream 11101100 และ 00010001 สลับกันจนได้จำนวน Data Codewords เท่ากับตาราง 2-2

พบว่าในเวอร์ชั่น 1 และ Error correction Level L ต้องทำการสร้าง Data codewords จำนวน 19 Codewords ขนาดความยาว 152 bit ดังนั้น จะได้ Bitstream สมบูรณ์ดังนี้

“00100000 00110101 00001010 11111010 00111110 10000000 11101100 00010001 11101100 00010001 11101100 00010001 11101100 00010001 11101100 00010001 11101100 00010001 11101100 00010001 11101100” หรือ “32 53 10 250 62 128 236 17 236 17 236 17 236 17 236 17 236 17 236”

2.1.8 การเข้ารหัสใน QR Code ในส่วน Error Correction Codewords (EC Codewords)

Error Correction Codewords เป็นส่วนที่ทำต่อจากส่วน Data Codewords ซึ่งเป็นส่วนประกอบใน Encoding Region เป็นการสร้างข้อมูลเพื่อให้แน่ใจว่า QR Code ยังสามารถ อ่านได้แม้ว่าส่วนหนึ่งหายไป โดยจะใช้ Reed-Solomon Error correction (RS Code) โดยมีกระบวนการที่สร้างเป็น Polynomial โดยใช้ Bitstream จากส่วนการเข้ารหัส Data codewords

เพื่อสร้าง Error Correction Codewords โดย Generator Polynomial เรียกอีกอย่างหนึ่งว่า A Finite Field, A Galois Field ซึ่งมี 256 องค์ประกอบ GF(256) โดย The Generator Polynomial จะอยู่ในรูปแบบ

$$(X - \alpha)(X - \alpha^2) .. (X - \alpha^t) \qquad (2.3)$$

โดยความยาว Bitstream ของ Error Correction Codewords (EC Codewords) นั้นจะขึ้นอยู่กับเวอร์ชั่นและ Error Correction ใน QR Code ซึ่งจากตัวอย่างสาธิต เป็น QR Code เวอร์ชั่นที่ 1 ที่ Error Correction ระดับ L จากตารางที่ 1 จำนวน Error Correction Codewords (ภาคผนวก ก-2) จะเท่ากับ 7 จะสามารถหา Generator Polynomial ได้จาก ภาคผนวก (ตาราง A. 1 ใน International standard ISO/IEC18004) ดังนี้

$$g(X) = X^7 + \alpha^{87}X^6 + \alpha^{229}X^5 + \alpha^{146}X^4 + \alpha^{149}X^3 + \alpha^{238}X^2 + \alpha^{102}X + \alpha^{21} \qquad (2.4)$$

จากหัวข้อ 8. สามารถหา Data Codewords ได้ดังนี้ “32 53 10 250 62 128 236 17 236 17 236 17 236 17 236 17 236 17 236”

ซึ่งสามารถหา Coefficients calculation f(X) โดยพิจารณาให้จำนวนสัมประสิทธิ์เท่ากับจำนวน Data Codewords ซึ่ง X ที่มีตัวยกกำลังมากที่สุดจะเท่ากับ 25 ได้ดังนี้

$$f(X) = 32X^{25} + 53X^{24} + 10X^{23} + 250X^{22} + 62X^{21} + 128X^{20} + 236X^{19} + 17X^{18} + 236X^{17} + 17X^{16} + 236X^{15} + 17X^{14} + 236X^{13} + 17X^{12} + 236X^{11} + 17X^{10} + 236X^9 + 17X^8 + 236X^7 + 0X^6 + 0X^5 + 0X^4 + 0X^3 + 0X^2 + 0X^1 + 0X^0 \qquad (2.5)$$

หลังจากคำนวณเสร็จสิ้นจะได้ค่าสัมประสิทธิ์พหุนามหรือค่า Error Correction Codewords ในเลขฐาน 10 คือ 134 58 142 175 250 92 207 หรือเท่ากับ

“1 0 0 0 0 1 1 0 0 0 1 1 1 0 1 0 1 0 0 0 1 1 1 0 1 0 1 0 1 1 1 1 1 1 1 1 1 0 1 0 0 1 0 1 1 1 1 0 0 1 1 0 0 1 1 1 1” ในเลขฐานสอง

2.1.9 โครงสร้าง Final Message ของ QR Code (Structure Final Message)

จำนวน Codeword ที่คำนวณได้จากในส่วน Data Codewords และ Error Codewords ต้องเท่ากับจำนวน Codewords ที่แสดงในตารางที่ 2 และภาคผนวก ก-2

ขั้นตอนต่อไปนี้จะต้องปฏิบัติตามเพื่อวางข้อมูลส่วน Data และ Error Correction กับส่วน Remainder Codeword ถ้ามีความจำเป็น

1) ทำการแบ่ง Data Codewords ออกเป็น N Block ตามที่กำหนดไว้ในภาคผนวก ก-2 ตามเวอร์ชันและ Error Correction Level

2) สำหรับแต่ละ Data Block ให้คำนวณ Error Correction Codewords ที่สัมพันธ์กันดังในหัวข้อ 8. และตาราง A .1 ใน International Standard ISO/IEC18004

3) รวบรวมส่วน Data และ Error Correction Codewords แต่ละ Block จากในตัวอย่าง ถ้ามี 4 block จะเป็นดังนี้ data block 1 , codeword 1 ; data block 2, codeword 1 ; data block 4, codeword 1; data block 1 , codeword 2; .....จนถึง data block 3 , codeword สุดท้าย ; data block 4 , codeword สุดท้าย ; และต่อมาจะเป็น error correction block 1 , codeword 1 , error correction block 2 , codeword 1 , ...จนถึง error correction block 4 , codeword สุดท้าย โดยใน QR Code บางเวอร์ชันเมื่อเติม Data Codeword และ Error Codeword จะไม่ครบ 8 bit ซึ่งจะต้องการ Remainder Bits จำนวน 3 bit , 4 bit หรือ 7 bit เพื่อทำให้ส่วน Encoding Region ครบสมบูรณ์

เมื่อยกตัวอย่าง QR Code เวอร์ชัน 5-H จะประกอบด้วย Data และ Error Correction อย่างละ 4 blocks โดย 2 blocks แรกประกอบด้วย 11 data และ 22 error correction codeword ตามลำดับ และในส่วน block 3 และ block 4 ประกอบด้วย 12 data และ 22 error correction codeword ตามลำดับซึ่งแสดงดังภาพ 4 โดย $\mathbf{D}_n$ จะแทนด้วย block ของ Data Codewords และ $\mathbf{E}_n$ จะแทนด้วย block ของ Error Correction Codewords โดยลำดับของ Codeword ที่วางจะเรียงจากบนลงล่างในแต่ละหลัก

	Data codewords					Error correction codewords			
Block 1	D_1	D_2	.....	D_{11}		E_1	E_2	.....	E_{22}
Block 2	D_{12}	D_{13}	.....	D_{22}		E_{23}	E_{24}	.....	E_{44}
Block 3	D_{23}	D_{24}	.....	D_{33}	D_{34}	E_{45}	E_{46}	.....	E_{66}
Block 4	D_{35}	D_{36}	.....	D_{45}	D_{46}	E_{67}	E_{68}	.....	E_{88}

ภาพที่ 4 โครงสร้างส่วน *Final Message Codeword*

ดังนั้นการเรียง Codeword ในเวอร์ชัน 5 – H จะเรียงในมีลักษณะ D_1 , D_{12} ,D_{23} ,D_{35} , D_2 ,D_{13} , D_{24} ,D_{36}.....D_{11} ,D_{22} ,D_{33} ,D_{45}, D_{34} ,D_{46} ,E_1 , E_{23} ,E_{45} ,E_{67} ,E_2 ,E_{24} ,E_{46} ,E_{68}..... ,E_{22} ,E_{44} , E_{66} ,E_{88}

เมื่อได้ Bitstream ของส่วน Data Codewords จากหัวข้อ 8. และส่วน Error Codewords จากหัวข้อ 9. ตัวอย่างสาธิต QR Code เวอร์ชัน 1 ที่ Error Correction Level L จะประกอบด้วย Data และ Error Correction อย่างละ 1 blocks และมี Codewords ทั้งหมด 26 ประกอบด้วย 19 data codeword และ 7 error correction codeword ตามลำดับดังนี้

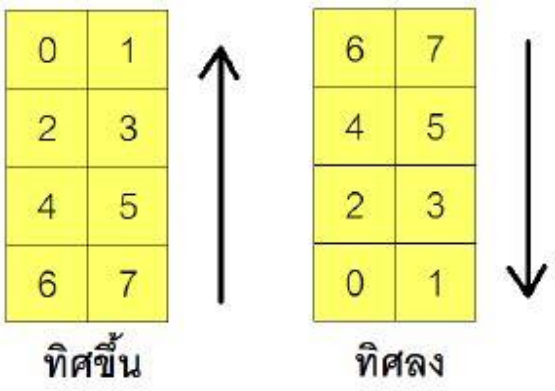
“0 0 1 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 1 0 1 1 0 1 0 0 1 1 1 0 1 1
0 0 1 1 0 0 0 1 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1
0 0 1 1 0 0 1 1 1 1 0 0 1 1 0 0 1 1 0 0 1 1 1 0 1 1 0 0
0 0 0 1 0 0 0 1 0 0 1 1 0 0 1 1”

2.1.10 การวาง Codeword ในเมตริก (Codeword Placement In Matrix)

การวาง Codeword ในเมตริกจะมี 2 ลักษณะ คือการวางแบบปกติ (regular) และ แบบไม่ปกติ (irregular) ซึ่งขึ้นอยู่กับตำแหน่งเมื่อเทียบกับสัญลักษณ์อื่นและ Function Patterns ส่วนมาก Codeword จะมีขนาด 2x4 โมดูล มีสองทางสำหรับ Codeword นี้ คือ แนวตั้ง (กว้าง 2 โมดูล และสูง 4 โมดูล) และในแนวนอน (กว้าง 4 โมดูล และสูง 2 โมดูล) ในการวางที่ผิดปกติจะเป็นการวางเพื่อเป็นการเปลี่ยนทิศทาง ซึ่งตำแหน่งในการวางจะขึ้นอยู่กับ Alignment Patterns หรือ Funtion Patterns อื่นๆ ดังภาพที่ 4 ภาพที่ 5 และภาพที่ 6 โดยตำแหน่งการวางจะเริ่มจากมุมล่างขวาจะวางต่อกันขึ้นไปโดยสลับขวาไปซ้าย โดยการกำหนดทิศทางมีหลักการดังนี้

1) ลำดับการวางบิต (Bit) ในหลักกการวางจะวางจากขวาไปซ้าย ทิศทางขึ้นไปหรือทิศทางลงตามทิศทางในการวาง

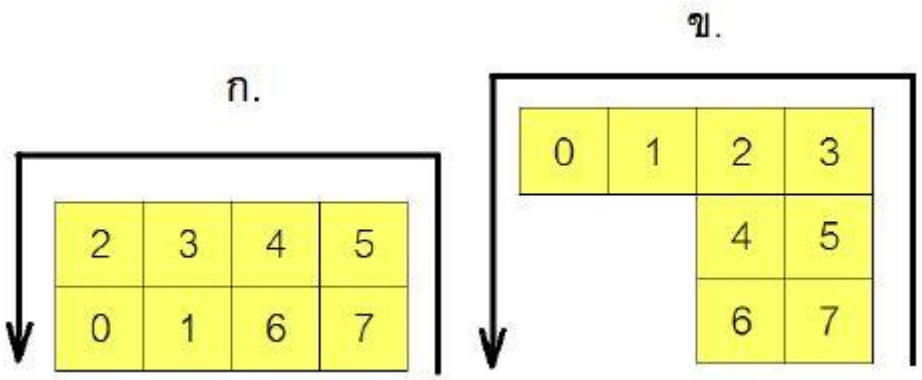
2) Significant Bit (Bit7) ของในแต่ละ Codeword จะเป็นการวางตำแหน่งแรก บิตที่ตามมาจะถูกวางในลำดับถัดไป Significant Bit จะอยู่ตำแหน่งขวาล่างในทิศทางขึ้นและตำแหน่งขวาบนเมื่อทิศทางลงดังภาพที่ 5



ภาพที่ 5 การวางบิตในรูปแบบปกติในทิศทางขึ้นและทิศทางลง

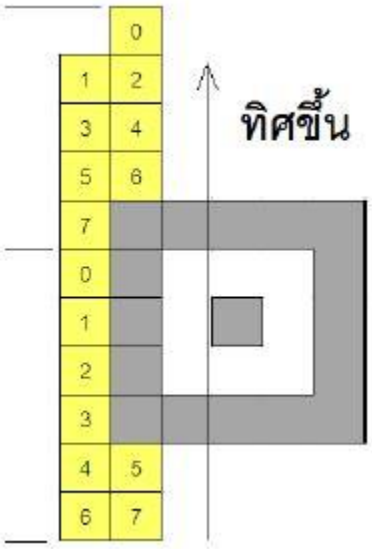
3) เมื่อพบขอบแนวนอนของ Alignment Pattern หรือ Timing Pattern ให้ทำการวางบิตให้อยู่เหนือหรือใต้ของ Pattern ทั้งสองโดยให้รหัสต่อเนื่องกัน

4) เมื่อวางบิตถึงขอบบนหรือขอบล่างของสัญลักษณ์ (เช่นขอบของ QR Code) Format Information Version Information หรือ Seperator ให้วางบิตที่เหลือไปทางซ้ายและทำการกลับทิศทางดังภาพที่ 6



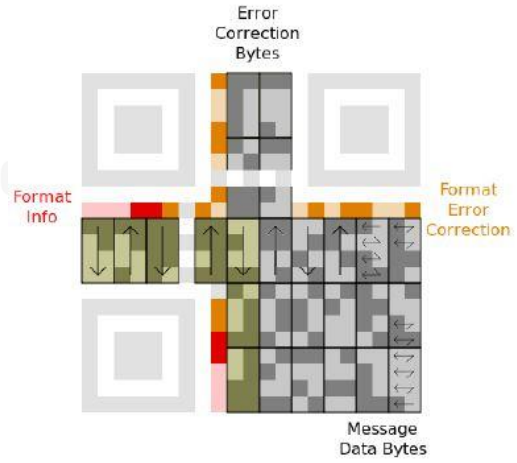
ภาพที่ 6 ก.การวางแบบปกติ ข.การวางแบบผิดปกติเมื่อทิศทางในการวางเปลี่ยนทิศทาง

5) เมื่อโมดูลทางขวามือพบ Alignment Pattern หรือพื้นที่ Version Information บิตจะถูกวางแบบไม่ปกติ โดยจะทำการขยายคอลัมน์หลักที่ติดอยู่กับ Alignment Pattern หรือ Version Information ถ้า Codeword สิ้นสุดก่อนจะวางคอลัมน์ถัดไป ให้วาง Significant Bit ของ Codeword ต่อไปในตำแหน่งที่ 1 คอลัมน์ดังภาพที่ 7



ภาพที่ 7 ตัวอย่างการวางบิตที่ติดกับ *Alignment Pattern*

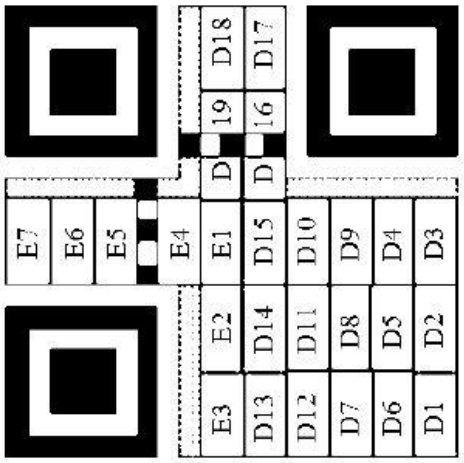
โดยใน QR Code เวอร์ชัน 1 Codeword จะเป็นการเรียงโมดูลเป็นแบบปกติขนาด 2x4 โมดูล โดยในการวาง Bitstream จะแบ่งวางทีละชุด ชุดละ8 บิต ลงใน Codeword แต่ละชุด ซึ่งบิต 1 จะแทนด้วยโมดูลสีดำและบิต 0 จะแทนด้วยโมดูลสีขาว



ภาพที่ 8 แสดงลำดับการวาง *Bitstream* บน *QR Code* เวอร์ชัน ***1***

ภาพที่ 8 แสดงการวาง Message Data Bytes หรือ Bitstream โดยหลักการวาง Bitstream นั้นจะเริ่มการวางบิตแรกจากมุมขวาล่างใน QR Code โดยรูปแบบการวางบิตลงบน Codeword นั้นจะมี 2 แบบ คือ รูปแบบวางบิตขึ้นไปข้างบน(Upwards) และรูปแบบการวางบิตลงมาด้านล่าง (Downwards)

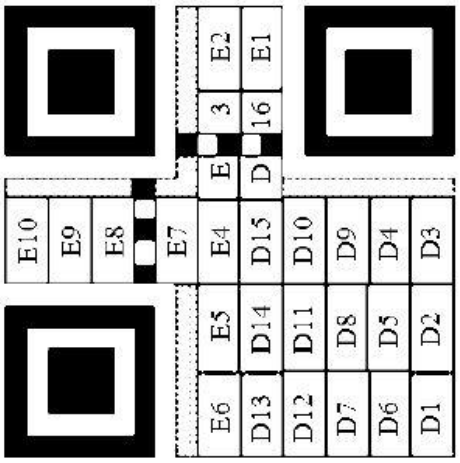
โดยหลักวาง Codeword ลงในโครงสร้างของ QR Code เวอร์ชัน 1 ที่ Error Correction Level L , M ,Q และ H จะมีลักษณะดังภาพที่ 9 ถึงภาพที่ 12



ภาพที่ 9 การเรียง *Codewords* ใน *QR Code* เวอร์ชัน *1-L*

D1 – D15 Data Block 1

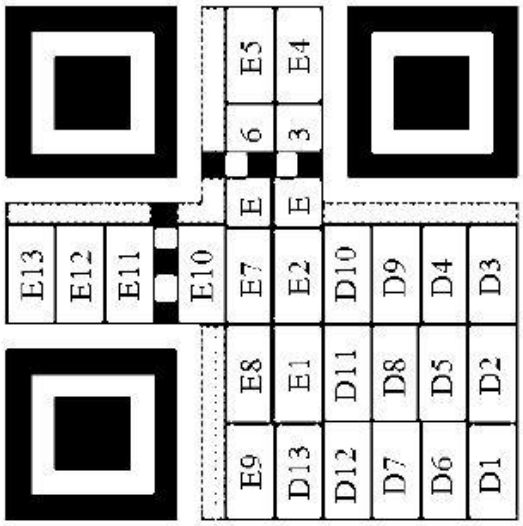
E1 – E7 EC Block 1



ภาพที่ 10 การเรียง *Codewords* ใน *QR Code* เวอร์ชัน *1-M*

D1 – D16 Data Block 1

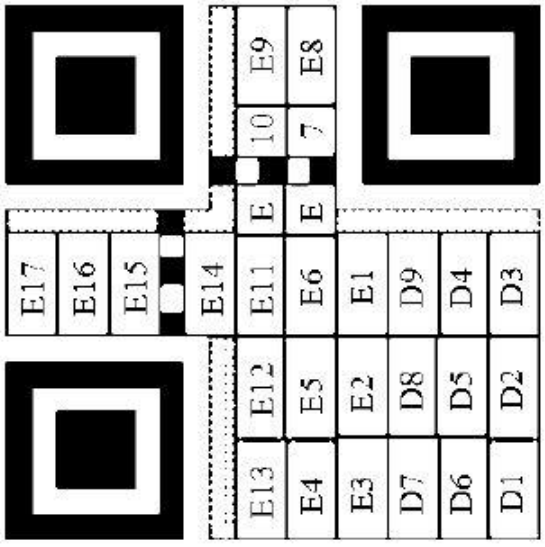
E1 – E10 EC Block 1



ภาพที่ 11 การเรียง *Codewords* ใน *QR Code* เวอร์ชัน *1-Q*

D1 – D13 Data Block 1

E1 – E13 EC Block 1

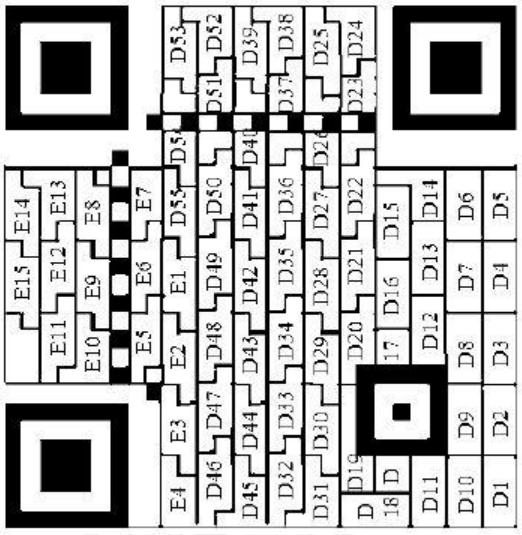


ภาพที่ 12 การเรียง *Codewords* ใน *QR Code* เวอร์ชัน *1-H*

D1 – D9 Data Block 1

E1 – E17 EC Block 1

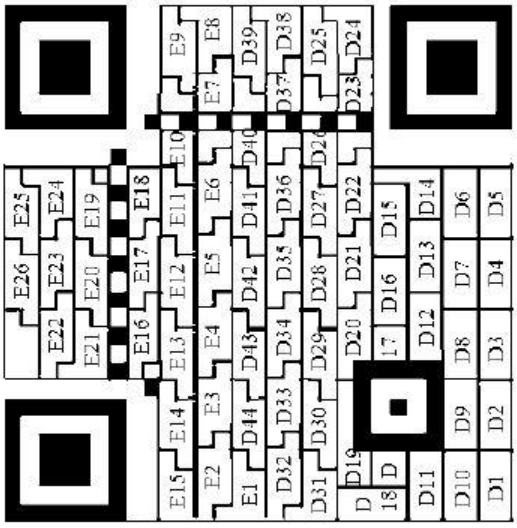
เมื่อวาง Codword ลงในโครงสร้างของ QR Code เวอร์ชัน 3 ที่ Error Correction Level L , M , Q และ H จะมีลักษณะดังภาพที่ 13 ถึงภาพที่ 16



ภาพที่ 13 การเรียง *Codewords* ใน *QR Code* เวอร์ชัน *3-L*

D1 – D55 Data Block 1

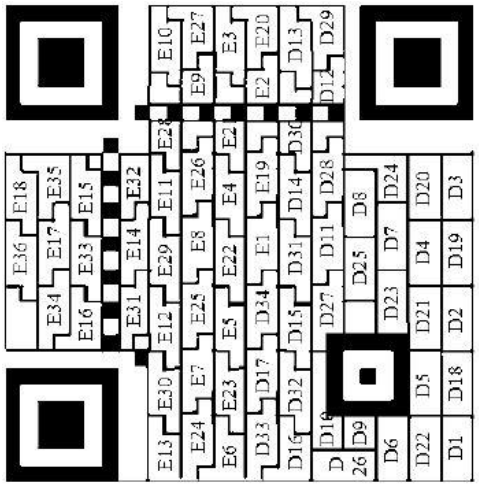
E1 – E15 EC Block 1



ภาพที่ 14 การเรียง *Codewords* ใน *QR Code* เวอร์ชัน *3-M*

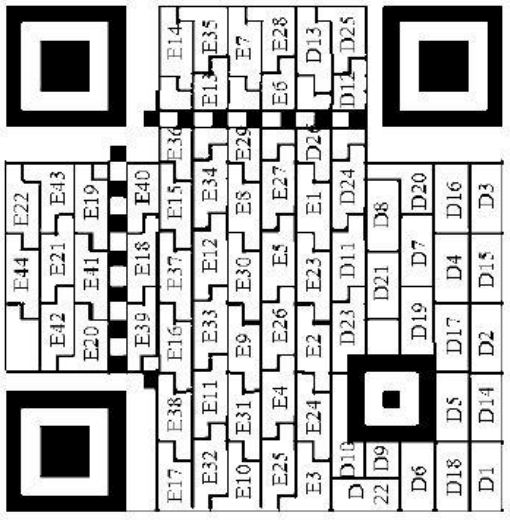
D1 – D44 Data Block 1

E1 – E26 EC Block 1



ภาพที่ 15 การเรียง *Codewords* ใน *QR Code* เวอร์ชัน *3-Q*

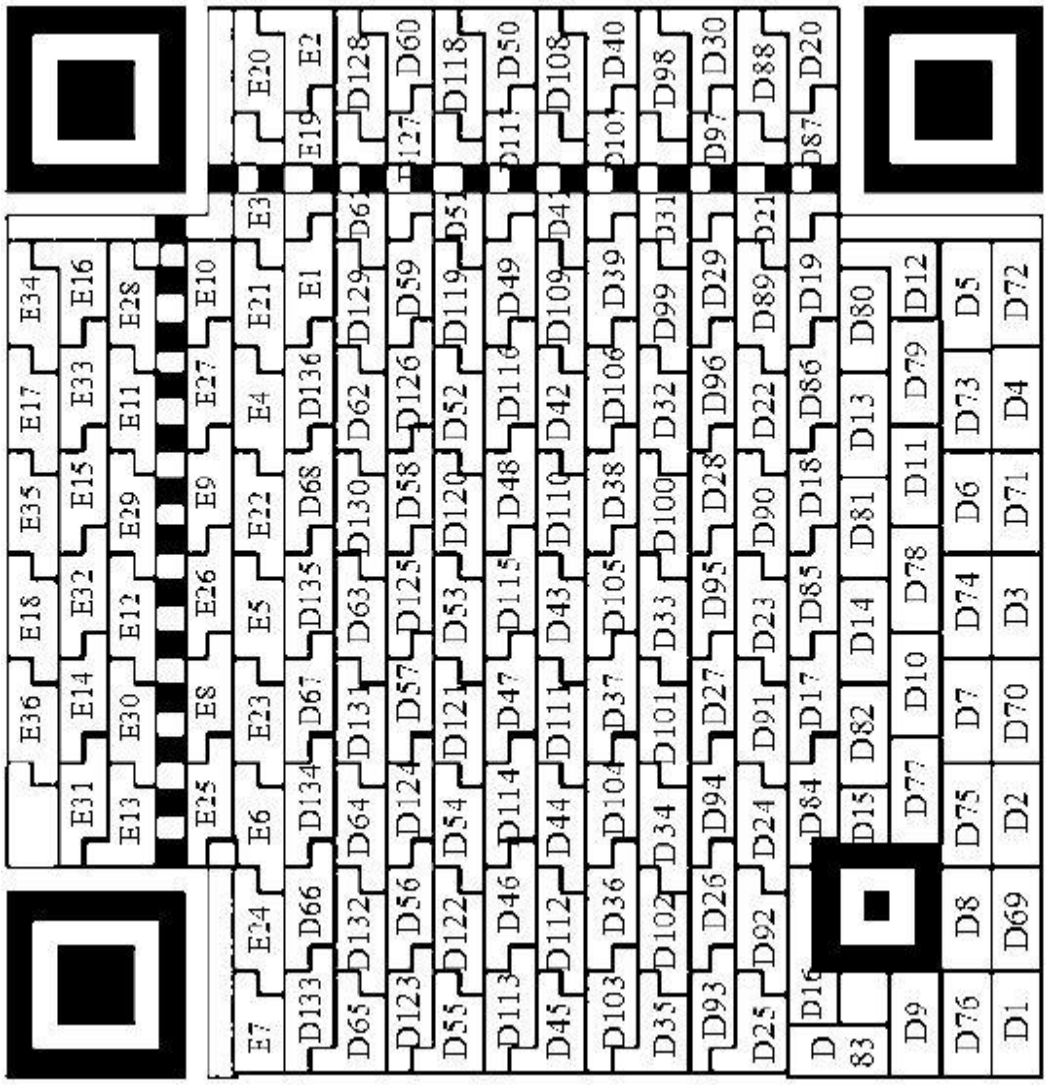
D1 – D17	Data Block 1
D18 – D34	Data Block 2
E1 – E18	EC Block 1
E19 – E36	EC Block 2



ภาพที่ 16 การเรียง *Codewords* ใน *QR Code* เวอร์ชัน *3-H*

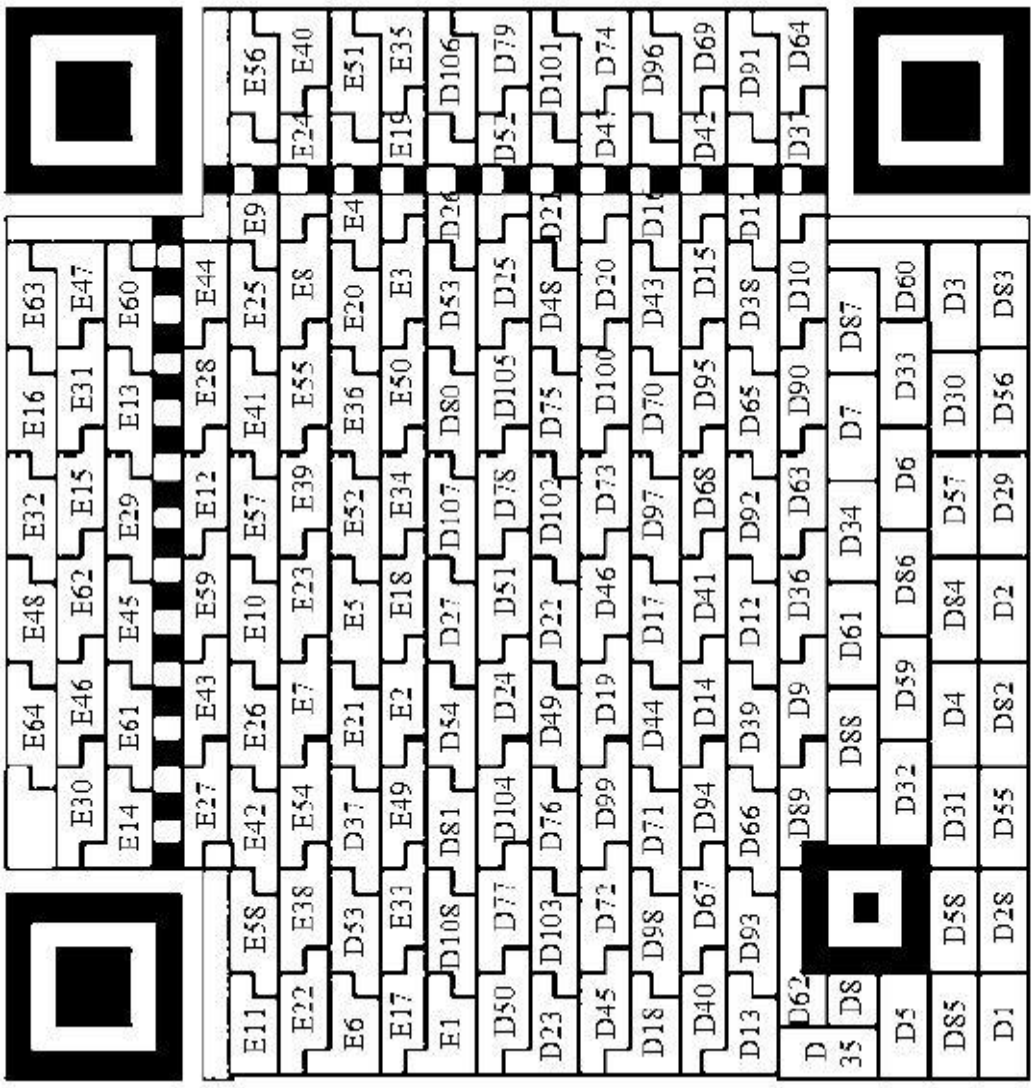
D1 – D13	Data Block 1
D14 – D26	Data Block 2
E1 – E22	EC Block 1
E23 – E44	EC Block 2

เมื่อวาง Codeword ลงในโครงสร้างของ QR Code เวอร์ชัน 6 ที่ Error Corrction Level L , M , Q และ H จะมีลักษณะดังภาพที่ 17 ถึงภาพที่ 20



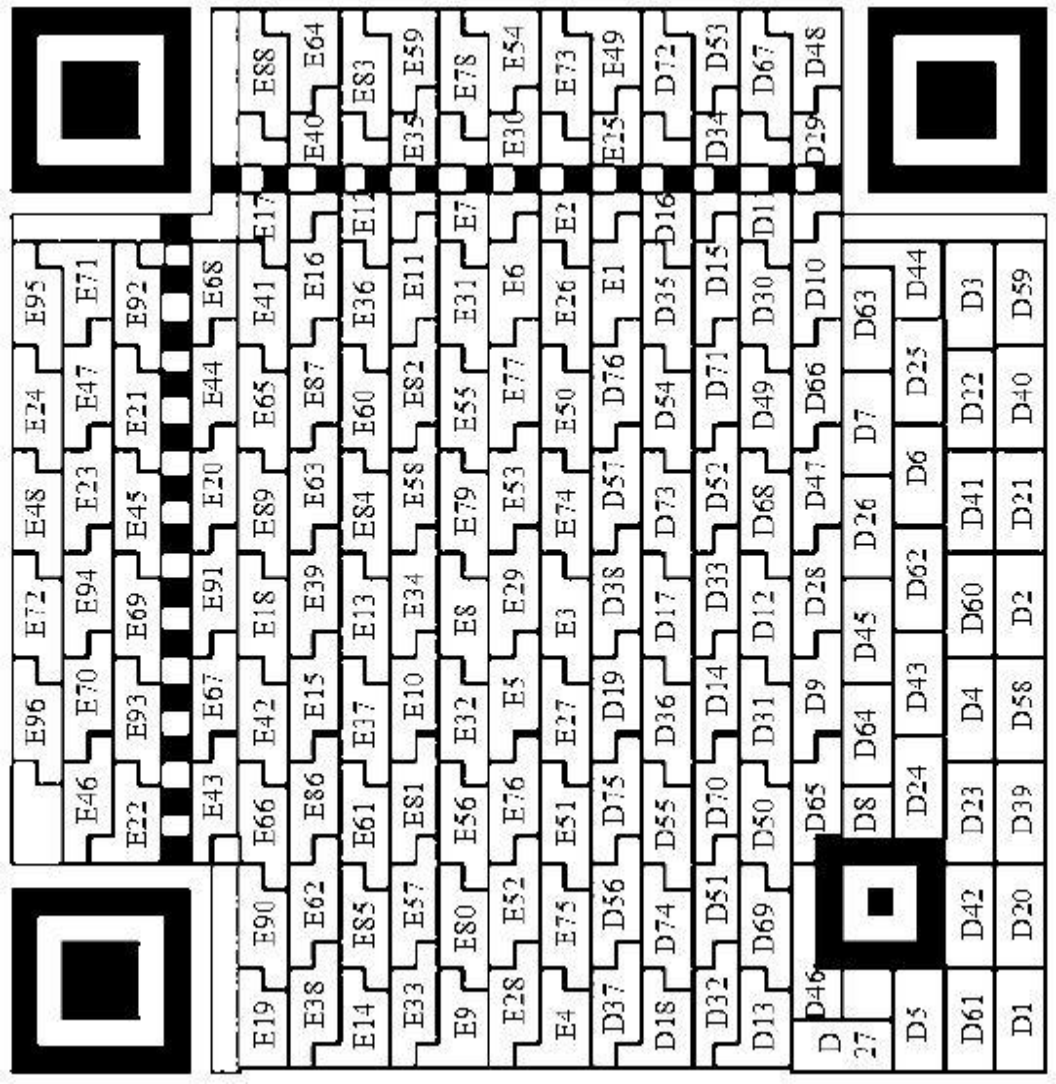
ภาพที่ 17 การเรียง *Codewords* ใน *QR Code* เวอร์ชัน *6-L*

D1 – D68	Data Block 1
D69 – D136	Data Block 2
E1 – E18	EC Block 1
E19 – E36	EC Block 2



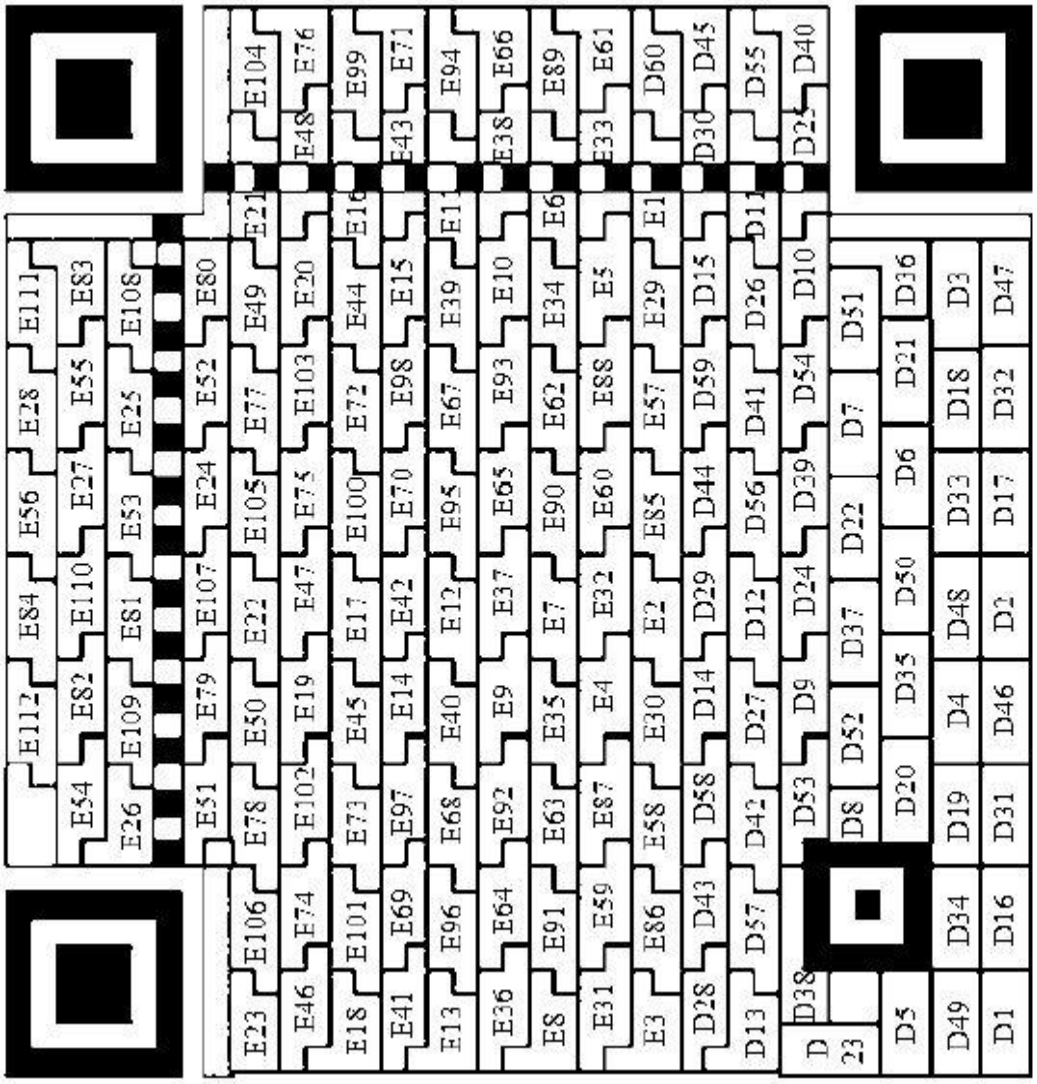
ภาพที่ 18 การเรียง *Codewords* ใน *QR Code* เวอร์ชัน *6-M*

D1 – D27	Data Block 1
D28 – D54	Data Block 2
D55 – D81	Data Block 3
D82 – D108	Data Block 4
E1 – E16	EC Block 1
E17 – E32	EC Block 2
E33 – E48	EC Block 3
E49 – E64	EC Block 4



ภาพที่ 19 การเรียง *Codewords* ใน *QR Code* เวอร์ชัน *6-Q*

D1 – D19	Data Block 1
D20 – D38	Data Block 2
D39 – D57	Data Block 3
D58 – D76	Data Block 4
E1 – E24	EC Block 1
E25 – E48	EC Block 2
E49 – E72	EC Block 3
E73 – E96	EC Block 4



ภาพที่ 20 การเรียง *Codewords* ใน *QR Code* เวอร์ชัน *6-H*

D1 – D15	Data Block 1
D16 – D30	Data Block 2
D31– D45	Data Block 3
D46 – D60	Data Block 4
E1 – E28	EC Block 1
E29 – E56	EC Block 2
E57 – E84	EC Block 3
E85 – E112	EC Block 4

2.1.11 ขั้นตอนการ Masking ใน QR Code

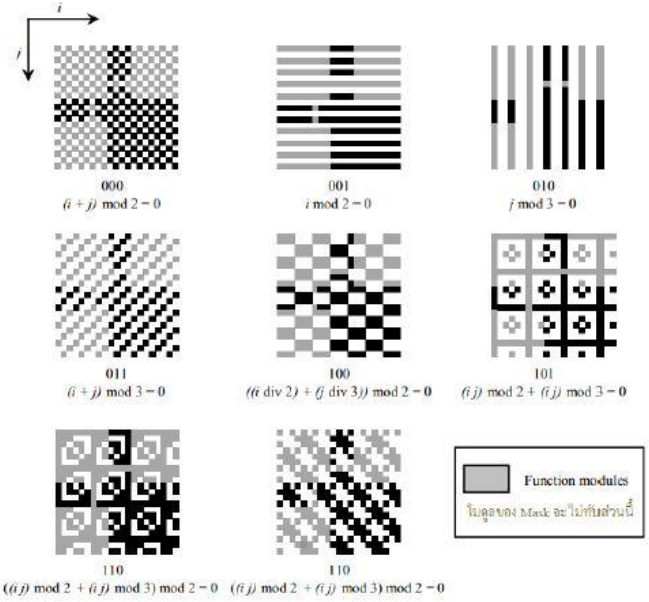
หลังจากที่ทำการลง Binary String ลงบน QR Code ต้องมีการ Mask ใน QR Code โดยมีวัตถุประสงค์เพื่อกระจายโมดูล ขาว ดำ ของ QR Code โดยการ Masking นั้นจะมี Mask ทั้งหมด 8 รูปแบบด้วยกันโดยจะต้องเลือกรูปแบบที่ดีที่สุด ซึ่งจะขึ้นตามเงื่อนไขของรูปแบบ Mask ตามตารางที่ 5 ดังนี้

ตารางที่ 5 รูปแบบ *Mask*

อ้างอิงรูปแบบ Mask	เงื่อนไข
000	(i + j) mod 2 = 0
001	i mod 2 = 0
010	j mod 3 = 0
011	(i + j) mod 3 = 0
100	((i div 2) + (j div 3)) mod 2 = 0
101	(i * j) mod 2 + (i * j) mod 3 = 0
110	((i * j) mod 2 + (i * j) mod 3) mod 2 = 0
111	((i * j) mod 3 + (i * j) mod 2) mod 2 = 0

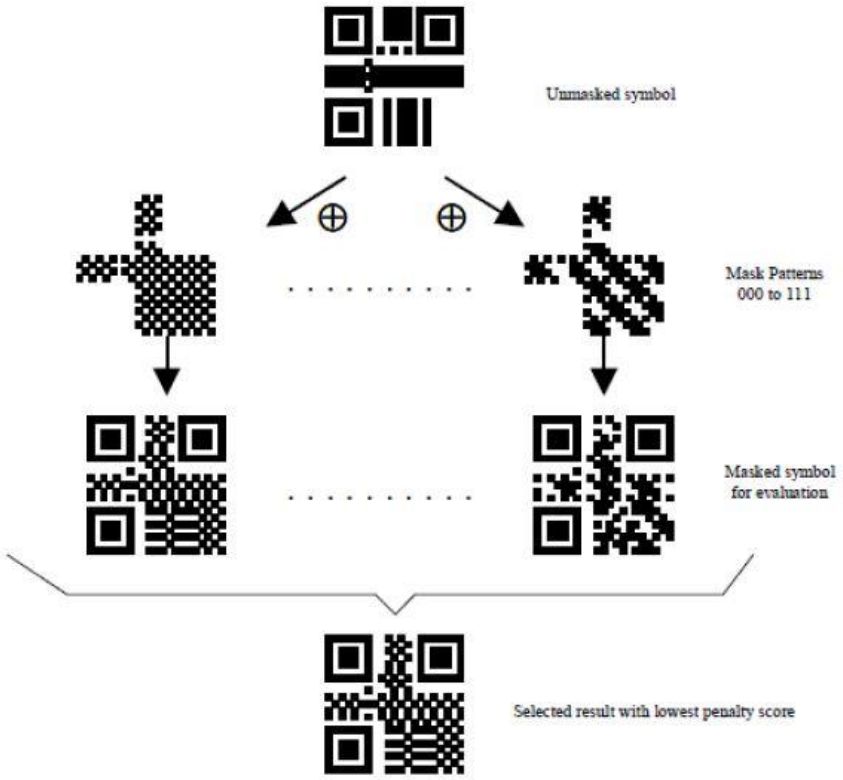
*i , j คือตำแหน่งที่อ้างอิงของโมดูลใน QR Code ถ้าโมดูลที่ตำแหน่ง i , j เป็นไปตามเงื่อนไขข้างต้น โมดูลที่ตำแหน่งนั้นจะสลับสีภายในโมดูล

จากตารางที่ 5 ทำให้สามารถหา Mask ของ QR Code เวอร์ชัน 1 ใน QR Code ได้ดังภาพที่ 21 ดังนี้



ภาพที่ 21 รูปแบบ *Masking* ใน *QR Code* เวอร์ชัน *1*

โดยขั้นตอน Masking นั้นยังสามารถใช้ Mask ที่มา XOR กับบิตที่วางในโครงสร้าง QR Code ไว้แล้วในหัวข้อ 8. ซึ่งจะไม่ทำการ Mask ที่บริเวณ Function Patterns ดังแสดงในภาพที่ 22



ภาพที่ 22 แสดงขั้นตอนการ *Mask*

เมื่อทำการ Mask กับทั้ง 8 รูปแบบแล้ว ต้องทำตามเงื่อนไขในตารางที่ 6 ด้านล่างเพื่อเลือกคะแนนที่น้อยที่สุด

ตัวแปร N1- N4 เป็นตัวแทนของค่าการถ่วงน้ำหนักโทษของคะแนน (N1= 3 , N2 = 3 , N3 = 40 , N4 = 10)

ตารางที่ 6 ตารางการให้คะแนนในการเลือกรูปแบบ *Mask*

รูปแบบ	**เงื่อนไข**	**คะแนนต่อจุด**
โมดูลที่อยู่ติดกันแถวหรือคอลัมน์ที่มีสีเดียวกัน	ติดกันมากกว่า 5 โมดูลขึ้นไป จำนวนของโมดูล = (5+i)	N1 + i
บล็อคของโมดูลสีเดียวกัน	บล็อกขนาด mxn	N2 x (m-1) x (n-1)
อัตราส่วน 1 : 1 : 3 : 1 : 1		N3

(ดำ,ขาว,ดำ,ขาว,ดำ) ในหลักหรือแถว		
สัดส่วนของโมดูลมืดในโมดูลทั้งหมด	50± (5 x k) %ถึง 50± (5 x (k+1)) %	N4 x k

2.1.12 การเข้ารหัสในส่วน Format Information

Format Information ประกอบด้วยข้อมูล 15 บิต ที่เรียงต่อกัน 5 บิตเป็น Data bit และ 10 บิตเป็น Error Correction โดยใช้ (5,15) BCH Code ในการเข้ารหัสในส่วน Format Informatin โดยจะอ้างอิงจาก ใน International Standard ISO/IEC 18004 โดยข้อมูล 2 บิตแรกจะเป็นข้อมูลของ Error Correction Level [5] ดังตารางที่ 7

ตารางที่ 7 ตารางตัวชี้วัด *Error Correction Level* สำหรับ *QR Code*

Error Correction Level	Binary Indicator
L	01
M	00
Q	11
H	10

ข้อมูลบิตที่ 3 ถึงบิตที่ 5 ของ Format Information จะเป็นข้อมูล Mask Pattern ตามตารางที่ 6 ส่วน 10 บิตที่เหลือได้จากการคำนวณ Error Correction แล้วนำมารวมกับ 5 บิตแรก เมื่อได้ข้อมูลครบทั้ง 15 บิตแล้ว ให้ทาการ XOR กับ Mask Pattern 101010000010010

จากตัวอย่าง

สมมติ Error Correction Level M : 00

รูปแบบ Data Mask Pattern: 101

ข้อมูล (Data) : 00101

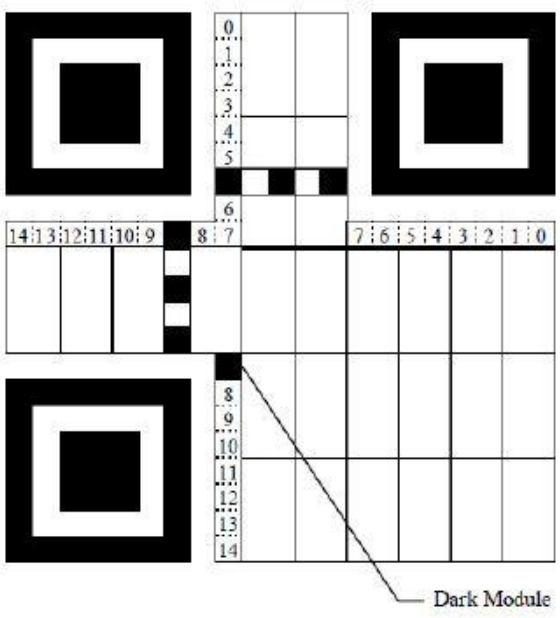
BCD Bits : 0011011100

ลำดับบิตที่ยังไม่ผ่านการ Mask : 001010011011100

Mask Pattern สำหรับ XOR : 101010000010010

รูปแบบ Format Information : 100000011001110

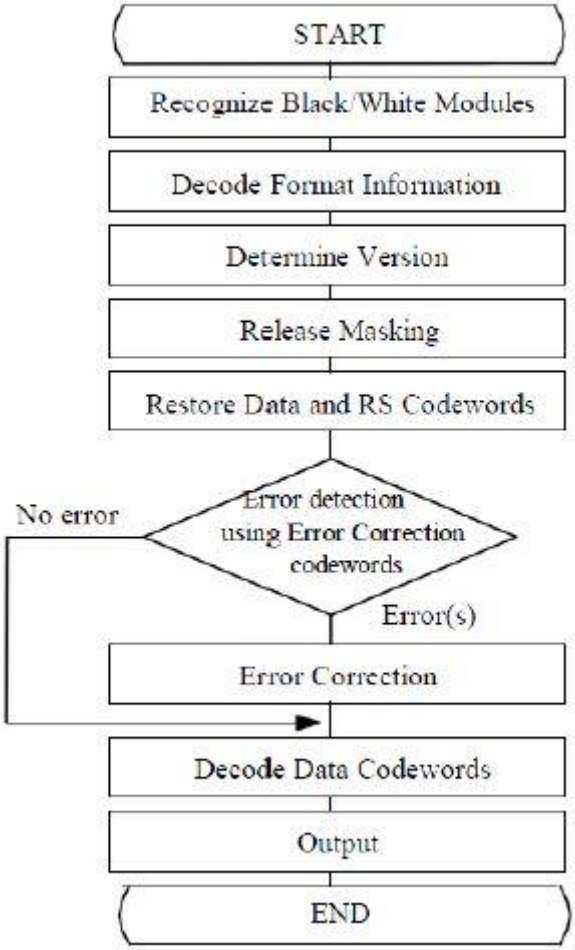
จาก Format Information ที่ได้จากด้านบนจะสามารถนำไปวางใน QR code ตามลำดับบิตดังในภาพที่ 23 โดยในการวางจะทำการวาง 2 ชุด โดยบิตที่น้อยที่สุดจะวางที่ตำแหน่ง 0 และบิตที่มากที่สุดจะวางที่ตำแหน่ง 14 โดย Dark Module ที่ตำแหน่ง (4V +9 , 8) โดย V แสดงเวอร์ชันของ QR Code จะไม่มีส่วนเกี่ยวข้องในการเข้ารหัส [6]



ภาพที่ 23 ตำแหน่ง *Format Information*

โดยรูปแบบ Format Information ทั้งหมดจะแสดงในภาคผนวก C

2.1.13 การถอดรหัส QR Code (Decoding Procedure)



ภาพที่ 24 ลำดับขั้นตอนในการถอดรหัสใน *QR code*

ขั้นตอนการถอดรหัสการอ่าน QR Code ทำได้โดยย้อนกลับจากการเข้ารหัส [7] ซึ่งมีขั้นตอนดังต่อไปนี้

1) หาตำแหน่งของภาพและทำให้โมดูลเรียนรู้แสงสว่างและมืดแล้วทำการแปลงเป็นอารเรย์บิต “0” และบิต “1”

2) อ่าน Format Information (ในส่วนของประเภท Masking Pattern และระดับ Error Correction)

3) อ่านข้อมูลเวอร์ชันของ QR Code

4) ทำการ Masking โดยใช้โอเปอร์เรเตอร์ XOR

5) อ่านตำแหน่งในรูปแบบการวางโมดูลตามหลักเกณฑ์

6) หากพบข้อผิดพลาดใช้ Error Correction Codeword ที่สอดคล้องกับระดับใดๆ

7) แบ่ง Codeword ข้อมูลออกเป็นส่วนๆ ตามตัวชี้วัดและโหมดนับจำนวนตัวอักษร

8) สุดท้ายจะสามารถถอดรหัสตัวอักษรข้อมูลตามโหมดในการใช้งานและส่งผลออกมา

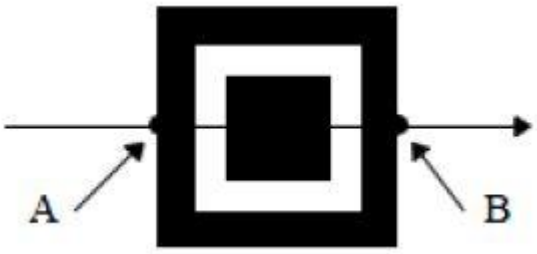
2.1.14 การอ้างอิงอัลกอริทึม (Algorithm) การถอดรหัสของ QR Code [8]

ในการอ้างอิงอัลกอริทึมในการถอดรหัส QR Code จะหาสัญลักษณ์ใน QR Code และทำการถอดรหัสโดยการอ้างอิงจากรูปมืด (Dark) และสว่าง (Light) ใน QR Code

1) พิจารณา Global Threshold ใน QR Code ที่ต้องการถอดรหัสโดยหาค่ากลางจากค่าสูงสุดและค่าต่ำสุด แล้วทำการแปลงเป็นพิกเซลเป็น สว่าง (Light) และ มืด (Dark) โดยใช้ Global Threshold

2) ทำการค้นหาส่วน Finder Pattern ซึ่งประกอบด้วย Finder Pattern ที่ตำแหน่ง 3 ตำแหน่งสำหรับการหามุมทั้ง 4 ของ QR Code โดย Finder Pattern พื้นที่ของแต่ละส่วนจะมีอัตราส่วน 1 : 1 : 3 :1 :1 โดยความคลาดเคลื่อนไม่เกิน 0.5 (เช่นช่วง 0 , 5-1 , 5 สำหรับ Single Module Box และ 2 , 5-3 , 5 สำหรับ Three Module Box)

ก. เมื่อพบตำแหน่งที่สนใจในจุดแรก (A) และจุดสุดท้าย (B) ที่ขอบภายนอกของ Finder Pattern ดังภาพที่ 25 จากในแกน X เราสามารถหาจุดศูนย์ได้จากการลากเส้นผ่านจากจุดทั้งสอง



ภาพที่ 25 การ *Scan Line* ใน *Finder Pattern*

ข. ทำซ้ำในขั้นตอน ก โดยพิจารณาแกน Y

ค. ตำแหน่งจุดศูนย์กลางของ Pattern เกิดจากการลากเส้นจากจุด A และ B ทั้งแกน X และแกน Y ตัดกัน

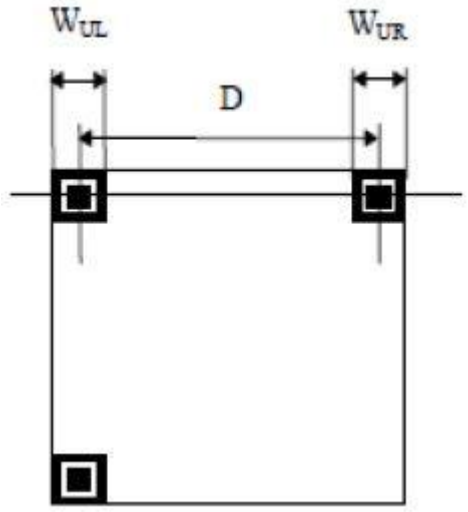
ง. ทำซ้ำขั้นตอน ก – ค สำหรับ Finder Pattern ใน 2 ตำแหน่ง ที่เหลือ

จ. ไม่มีพื้นที่ที่สนใจ ให้กลับไปทำการแปลงพิกเซลเป็นสว่าง (Light) และมืด (Dark) ใหม่ก่อนทำขั้นตอน ข

ฉ. ถ้าพบ Finder Pattern เพียง 1 ตำแหน่งอีก 2 ตำแหน่งค้นหาไม่พบขั้นตอนจะเปลี่ยนอ้างอิงการถอดรหัสของ Micro QR Code

3) กำหนดตำแหน่งในการหมุนของสัญลักษณ์ โดยทำการวิเคราะห์จุดศูนย์กลางของคู่ Finder Pattern เพื่อหา Finder Pattern ในตำแหน่งมุมซ้ายบนและหามุมในการหมุนของสัญลักษณ์

4) ทำการกำหนด ก. D ระยะระหว่างศูนย์กลางของ Finder Pattern มุมบนซ้ายและศูนย์กลางของ Finder Pattern มุมบนขวาและ ข. ความกว้างของ Pattern WUL และ WUR ดังแสดงในภาพที่ 26



ภาพที่ 26 *Finder Pattern* ด้านบน

5) ทำการคำนวณ Nominal X dimension ของ QR Code จากสมการ 2.6

$$X = (Wul + Wur) / 14) \quad (2.6)$$

6) ทำการกำหนดเวอร์ชั่น (v) ของ QR Code จากสมการ 2.7

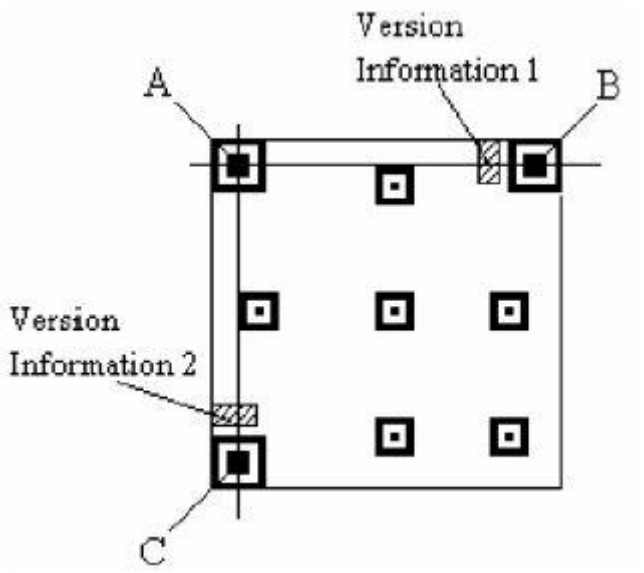
$$V = [(D / X) - 10)] / 4) \quad (2.7)$$

7) ถ้าค่าเวอร์ชั่นของ QR Code เป็น 6 หรือน้อยกว่าให้กำหนดเป็นเวอร์ชั่นของ QR Code แต่ถ้าค่าเวอร์ชั่นเป็น 7 หรือมากกว่า ให้ทำถอดรหัสส่วนของ Version Information ตามขั้นตอนดังนี้

ก. หาขนาดของโมดูลด้วยการหารความกว้าง WUR ด้วย 7 จากสมการ 2.8

$$CP_{UR} = W_{UR} / 7 \quad (2.8)$$

ข. ทำการหาเส้น Guide Line ของ AC จากจุด A, C และ AB จากจุด A, B โดยแต่ละเส้นจะผ่านจุดกึ่งกลางของ Finder Pattern ทั้ง 3 ตำแหน่ง ดังภาพที่ 27 ซึ่ง Sampling Grid ที่ผ่านจุดศูนย์กลางของ Module ในพื้นที่ Version Information 1 สามารถคำนวณได้จากเส้นที่ขนานกับ Guide Line และขนาดของโมดูล $\mathrm{CP_{UR}}$ ซึ่งค่าไบนารี “0” และ “1” สามารถกำหนดได้จากโมดูลสว่าง (Light) หรือโมดูลมืด (Dark) บน Sampling Grid

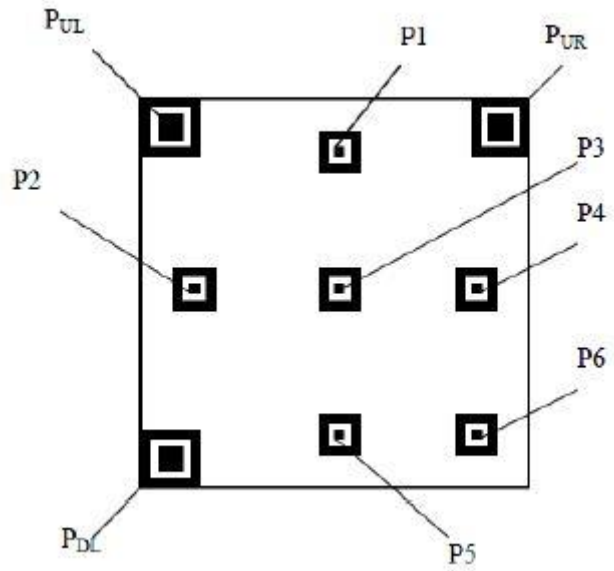


ภาพที่ 27 *Finder Pattern* และ *Version Information*

ค. ทำการหาเวอร์ชั่น โดยการตรวจหา Error Correcting ตามในภาคผนวก D.2 ใน International standard ISO/IEC 18004

ง. ถ้าหากค่า Error มากกว่า Error Correction ที่มีการตรวจพบ ให้ทำการคำนวณความกว้างที่ Finder Pattern ที่อยู่มุมล่างซ้าย โดยทำขั้นตอนเหมือนกับขั้นตอน 1) 2) และ 3) เพื่อถอดรหัส Version Information 2

8) ใน QR Code เวอร์ชั่น 1 ให้กำหนดค่า X เป็นค่าประมาณช่องว่างจุดศูนย์กลางโมดูลและโทดูลสว่างของ Timing Pattern ทางด้านบนและกำหนดค่า Y เป็นค่าประมาณระหว่างจุดศูนย์กลางโมดูลมืดและโมดูลสว่างของ Timing Pattern ทางด้านซ้าย ทำการสร้าง Samping Grid ให้ลากผ่านแนวนอนของ Timing Pattern ด้านบน แล้วทำการลากเส้นขนานในตำแหน่งช่องว่างจุดศูนย์กลางในแนวแกนตั้ง Y ซึ่งมี 6 เส้นและทำการลากเส้นผ่าน Timing Pattern ทางด้านซ้าย แล้วทำการลากเส้นขนานในตำแหน่งช่องว่างจุดศูนย์กลางในแนวแกนนอน X ซึ่งมี 6 เส้น จากการอ้างอิงในแนวตั้ง ซึ่งจะถูกกำหนดไว้ในแต่ละเวอร์ชั่นของ QR Code สำหรับในเวอร์ชั่นที่มากกว่าเวอร์ชั่น 1 ให้กำหนดกึ่งกลางของแต่ละ Alignment Pattern จากที่กำหนดไว้ใน ภาคผนวก E ใน International standard ISO/IEC 18004 และสร้าง Samping Grid ด้วยระยะห่างที่เท่ากันระหว่างจุด ดังแสดงในภาพที่ 28 ซึ่งมีขั้นตอนดังนี้



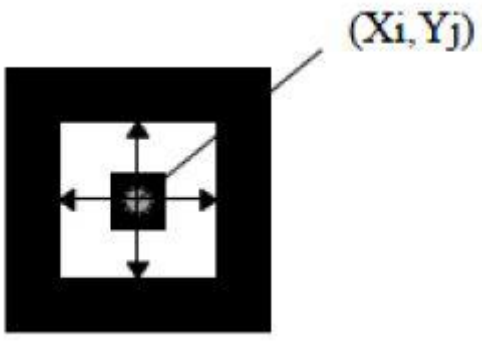
ภาพที่ 28 *Finder Pattern* และ *Alignment Patterns*

ก. ทำการหาขนาดของโมดูล CP_{UL} โดยคำนวณจากความกว้าง Pattern ของ W_{UL} ของ Finder Pattern ทางซ้ายบน (P_{UL}) หารด้วย 7 ดังสมการ 2.9

$$CP_{UL} = W_{UL} / 7 \quad (2.9)$$

ข. ทำการกำหนด Alignment Patterns P1 และ P2 ดังภาพที่ 32 ซึ่งจะขึ้นอยู่กับจุดศูนย์กลาง A ของ Finder Pattern มุมซ้ายบน (P_{UL}) เส้นที่ขนานกับ AB และ AC รวมถึงขนานกับโมดูล CP_{UL}

ค. ทำการค้นหาขอบนอกของสี่เหลี่ยมสีขาวของ Alignment Pattern P1 และ Alignment Patterns P2 โดยเริ่มจากพิกเซลจุดศูนย์กลาง X_i และ Y_i ดังภาพที่ 29



ภาพที่ 29 จุดศูนย์กลางของ *Alignment Pattern*

ง. ทำการประมาณจุดศูนย์กลาง Alignment Pattern ของจุด P3 โดยจะขึ้นอยู่กับศูนย์กลางของ Finder Pattern มุมบนซ้าย ($\mathbf{P_{UL}}$) และจุดศูนย์กลางของ Alignment Patterns P1 และ Alignment Patterns P2 จากขั้นตอนที่ ค.

จ. ทำการหาพิกัดของ Alignment Pattern P3 ตามขั้นตอนที่ ค.

ฉ. ทำการหา L_X ระยะทางจากกึ่งกลางระหว่าง Alignment Pattern P2 และ Alignment Pattern P3 และทำการหา L_Y ระยะทางจากกึ่งกลางระหว่าง Alignment Pattern P1 และ Alignment Pattern P3 ต่อมาทำการแบ่ง L_X กับ L_Y เพื่อหา CP_X และ CP_Y ของพื้นที่บนซ้ายใน QR Code ดังแสดงในภาพ 2-30 ได้ดังนี้

$$CP_X = L_X / AP \qquad (2.10)$$

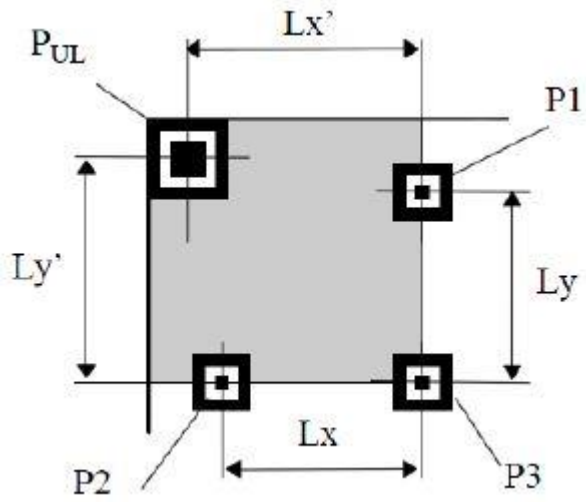
$$CP_Y = L_Y / AP \qquad (2.11)$$

AP คือ ที่ว่างของ Module ระหว่างจุดศูนย์กลางของ Alignment Pattern (สามารถดูได้ที่ตารางภาคผนวก E.1 ใน Internationalstandard ISO/IEC 18004)

ในทำนองเดียวกันจะสามารถหา $L_{X'}$ ซึ่งเป็นระยะห่างในแกนนอนระหว่างจุดศูนย์กลางของ Finder Pattern มุมซ้ายบน (P_{UL}) และจุดศูนย์กลางของ Alignment Pattern P1 และ $L_{Y'}$ ซึ่งเป็นระยะห่างในแกนตั้งระหว่างจุดศูนย์กลางของ Finder Pattern มุมซ้ายบน (P_{UL}) และจุดศูนย์กลางของ Alignment Pattern P2 โดยจะสามารถแบ่ง $L_{X'}$ และ $L_{Y'}$ เพื่อหา $CP_{X'}$ และ $CP_{Y'}$ ของพื้นที่บนซ้ายใน QR Code ดังสมการดังนี้

$CP_{X'} = L_{X'}$ / คอลัมน์ของจุดศูนย์กลางใน Alignment Pattern P1 – คอลัมน์ของจุดศูนย์กลางใน Finder Pattern (P_{UL}) (2.12)

$CP_{Y'} = L_{Y'}$ / คอลัมน์ของจุดศูนย์กลางใน Alignment Pattern P2 – คอลัมน์ของจุดศูนย์กลางใน Finder Pattern (P_{UL}) (2.13)

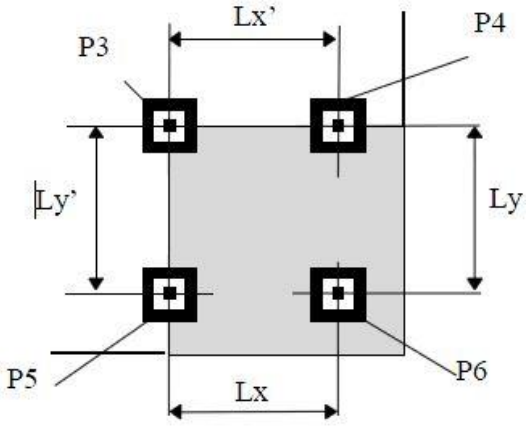


ภาพที่ 30 พื้นที่ซ้ายบนของ *QR Code*

ช. ทำการกำหนด Sampling Grid ให้ครอบคลุมพื้นที่ซ้ายบนของ QR Code โดยให้ขึ้นอยู่กับขนาดโมดูลที่คำนวณได้จาก CP_X $CP_{X'}$ CP_Y และ $CP_{Y'}$

ซ. ในทำนองเดียวกันกำหนด Sampling Grid สำหรับพื้นที่ขวาบนของ QR Code (ครอบคลุมถึง Finder Pattern (P_{UR}) Alignment Pattern P1 P3 และ P4) และพื้นที่ซ้ายล่างของ QR Code (ครอบคลุมถึง Finder Pattern (P_{DL}) Alignment Pattern P2 P3 และ P5

ฌ. จาก Alignment Pattern P6 ดังแสดงใน ภาพที่ 31 สามารถทำการประมาณจุดศูนย์กลางได้จาก $CP_{X'}$ และ CP_Y ซึ่งแต่ละค่าสามารถหาได้จาก Alignment Pattern P3 P4 และ P5 ซึ่ง Guide Line ผ่านจุดศูนย์กลางของ Alignment Pattern P3 P4 และ Alignment Pattern P3 P5 ตามลำดับ โดยเป็นคู่อันดับศูนย์กลางของ Pattern



ภาพที่ 31 พื้นที่ขวาล่างของ *QR Code*

ญ. ทำซ้ำขั้นตอน จ. – ซ. ทำการกำหนด Sampling Grid สำหรับพื้นที่ขวาล่างของ QR Code

ฎ. ด้วยหลักการเดียวกัน สามารถใช้การตรวจสอบพื้นที่ของ QR Code ที่ยังไม่ครอบคลุมโดยใช้ Sampling Grid ได้

9) ทำการ Global Threshold ใน QR Code ตัวอย่างขนาด 3×3 พิกเซล แล้วทำการแบ่งกลุ่มเป็นกลุ่มมืด (Dark) แทนด้วย 1 และกลุ่มสว่าง (Light) แทนด้วย 0

10) ทำการถอดรหัส Format Information ที่อยู่ใกล้บริเวณ Finder Pattern ด้านซ้ายบน จะทำให้ทราบ Error Correction Level และ Data Mask Pattern ที่ใช้บน QR Code ถ้าค่า Error มากกว่าค่า Error Correction ของ Format Information จะทำการย้ายไปถอดรหัสที่อยู่ใกล้บริเวณ Finder Pattern ขวาบนและ Finder Pattern ซ้ายล่าง

11) ในกรณี Format Information ไม่สามารถอ่านได้ ให้ดำเนินการโดยใช้การถอดรหัสรูปแบบ Mirror Image Symbol ในตำแหน่งแถวและหลักที่เป็นคู่อันดับ Transposed

12) ทำการ XOR ส่วน Data Mask Pattern ในบริเวณ Encoding Region ใน QR Code เพื่อทำการถอด Data Masking และเพื่อนำข้อมูลในส่วน Data Codewords และ Error Codewords ออกมา ขั้นตอนนี้จะเป็นวิธีตรงกันข้ามกับขั้นตอนในการเข้ารหัสใน QR Code

13) ทำการกำหนด Codewords ตามกฎว่าด้วยการจัดวางในหัวข้อ 2.1.10

ก. ทำการจัดลำดับ Codewords ใหม่ โดยจัดให้เป็น Blocks ตามข้อกำหนดของเวอร์ชั่นและ Error Correction Level

ข. ตรวจหาข้อผิดพลาดและกู้ข้อผิดพลาดตาม Annex B ใน International standard ISO/ICE 18004 เพื่อทำการแก้ไขข้อผิดพลาดให้ถูกต้องสำหรับเวอร์ชั่นและ Error Correction Level

ค. ทำการกู้ข้อมูลกลับให้เป็น Bitstream โดยเรียงตาม Data Blocks ตามลำดับ

ง. ทำการแบ่ง Bitstream ออกเป็นส่วนต่างๆ ซึ่งมีการระบุโหมด (Mode Indicator) ความยาวตัวอักษร ซึ่งถูกกำหนดด้วย Character Count Indicator

จ. ทำการถอดรหัสแต่ละส่วนตามโหมด (Mode) ที่กำหนดไว้

2.2 การเข้ารหัสช่องสัญญาณ

การเข้ารหัสช่องสัญญาณแบ่งเป็น 2 ประเภท

1. การเข้ารหัสรูปคลื่น (Waveform Coding)
2. ลำดับเชิงโครงสร้าง (Structured Sequence) เป็นการเพิ่มบิตเกิน(Redundant bits) เข้าไปในลำดับบิตเดิมที่ต้องการส่งไปยังแหล่งปลายทางให้เป็นลำดับบิตใหม่ ซึ่งจะช่วยวงจรถอดรหัสในการตรวจหาและแก้ไขข้อผิดพลาด

ในงานวิจัยนี้ จะสนใจเฉพาะการเข้ารหัสช่องสัญญาณแบบลำดับเชิงโครงสร้างเท่านั้น หรือเรียกสั้น ๆ ว่า รหัสแก้ไขข้อผิดพลาดข้างหน้า (Forward error correction code) หรือ รหัสแก้ไขข้อผิดพลาด (ECC: error correction code) รหัสแก้ไขข้อผิดพลาดที่นิยมนำมาใช้ในการเข้ารหัสช่องสัญญาณนั้นมีด้วยกันสองชนิดคือรหัสบล็อก (Block codes) และรหัสคอนโวลูชั่น (Convolutional code) โดยในงานวิจัยนี้ได้มุ่งความสนใจไปยังรหัสบล็อก (Block codes) ซึ่งรหัสบล็อกจะมีลักษณะการเข้ารหัสและถอดรหัสทีละบล็อก บล็อกละหลาย ๆ บิต โดยอาศัยความรู้ทางด้านฟิลด์จำกัด (Finite field)และพีชคณิตนามธรรม (Abstract algebra) ในการเข้าและถอดรหัส เช่น รหัสบล็อกเชิงเส้น (linear block code), รหัสวน (cyclic code), รหัสแฮมมิง (Hamming code) และ รหัสรีดโซโลมอน (Reed Solomon code) ที่ใช้ใน QR CODE.

การเข้ารหัสบล็อกแบบเชิงเส้นนั้นเป็นการนำข้อมูลข่าวสาร (message) ที่ต้องการเข้ารหัสมาแบ่งเป็นชุดๆ หรือบล็อกย่อยๆ ซึ่งแต่ละชุดนั้นมีขนาดเท่ากับ *k* บิต แล้วนำมาหาค่าคำรหัส (codeword)
ขนาด n บิต โดยที่ n > k แทนด้วยสัญลักษณ์ (n, k) บิตเกินที่เพิ่มขึ้นมาจำนวน n – k บิตเรียกว่าบิตพาริตี้ (parity bits) ซึ่งพาริตี้บิตจะแบ่งได้เป็น 2 ชนิดคือ พาริตี้คี่ และพาริตี้คู่

พาริตี้คี่ (Odd parity) คือจำนวนบิตทั้งหมดที่มีค่าเป็น “1” ของบิตข้อมูล ซึ่งรวมพาริตี้ด้วยแล้วจะเป็น

จำนวนคี่

พาริตี้คู่ (Even parity) คือจำนวนบิตทั้งหมดที่มีค่าเป็น “1” ของบิตข้อมูล ซึ่งรวมพาริตี้บิตด้วยแล้วจะเป็นจำนวนคู่

โดยหน้าที่ของบิตพาริตี้ ตรวจสอบสภาวะคู่หรือคี่หรือเรียกสั้นๆว่าบิตตรวจสอบก็คือ ตรวจสอบและแก้ไขข้อผิดพลาดที่เกิดจากช่องสัญญาณ ซึ่งปริมาณและวิธีการในการเพิ่มบิตตรวจสอบจะบ่งบอกถึงระดับในการป้องกันและแก้ไขข้อผิดพลาดที่เกิดขึ้นจากช่องสัญญาณ โดยอัตราส่วนของจำนวนบิตข้อความต่อจำนวนบิตของคำรหัสจะเรียกว่าอัตรารหัส (code rate) r = k/n

ในที่นี้แทนบิตข่าวสารด้วย $m = [m_1\ m_2 \ldots m_k]$ และแทนคำรหัสด้วย $c = [c_1\ c_2 \ldots c_n]$ ดังนั้นจำนวนบิตตรวจสอบสภาวะคู่หรือคี่ที่ทำการเพิ่มเข้าไปจะมีจำนวนเท่ากับ n – k

2.3 รหัสแก้ไขข้อผิดพลาด Low Density Parity Check (LDPC) [9]

รหัสแก้ไขข้อผิดพลาด LDPC (low density parity-check codes) หรือรหัสแก้ไขข้อผิดพลาดสภาวะคู่หรือคี่ชนิดความหนาแน่นต่ำ ถูกจัดว่าเป็นรหัสบล็อกเชิงเส้น (linear block

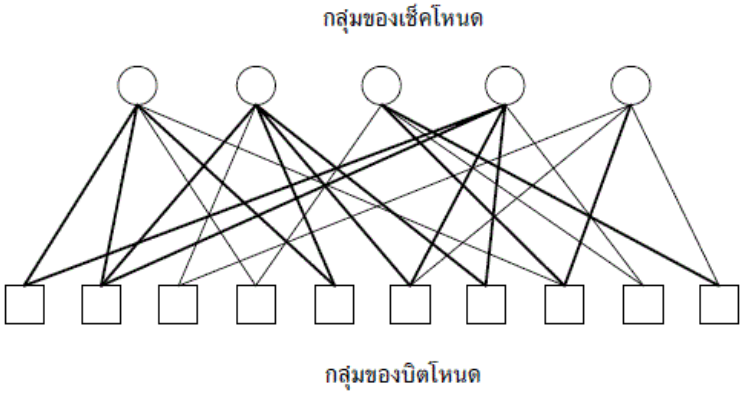
codes) ชนิดหนึ่งถูกนำเสนอครั้งแรกโดย R. Gallager [10] และถูกนำกลับมาศึกษาและทำให้เป็นที่สนใจอีกครั้งโดย D.Mackay และ R.Neal [11] รหัส LDPC มีประสิทธิภาพเข้าใกล้ความจุสูงสุดของช่องสัญญาณแบบต่าง ๆ หลายช่องสัญญา ทำให้รหัส LDPC ถูกประยุกต์ใช้ในการสื่อสารเชิงดิจิทัลหลายรูปแบบ รหัสชนิดนี้ จะถูกนิยามด้วย parity-check matrix ที่มีความหนาแน่นต่ำ ซึ่งความหนาแน่นของ parity-check matrix หมายถึงอัตราส่วนของจำนวนของสมาชิกที่มีค่าเป็น 1 ต่อสมาชิกที่มีค่าเท่ากับ 0 ใน parity-check matrix นั้นมีค่าต่ำ โดย parity-check matrix ใดๆ จะถูกเรียกว่ามีความหนาแน่นต่ำก็ต่อเมื่อจำนวนสมาชิกที่เป็น 1 มีน้อยกว่าครึ่งหนึ่งของสมาชิกทั้งหมด ด้วยลักษณะนี้จึงถูกเรียกว่ารหัสแก้ไขข้อผิดพลาดสภาวะคู่หรือคี่ชนิดความหนาแน่นต่ำ ตัวอย่าง parity-check matrix ของ LDPC

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

ภาพที่ 32 ตัวอย่าง *parity-check matrix H*

รหัส LDPC จะเป็นชนิด regular ถ้า parity-check matrix มีจำนวนสมาชิกที่เป็น 1 ในแต่ละแถวและแต่ละหลักเป็นค่าคงที่ แต่ถ้าไม่เป็นไปตามนั้นจะถูกจัดว่าเป็นชนิด irregular ดังนั้นตัวอย่าง parity-check matrix ที่แสดงจึงจัดว่าเป็นชนิด irregular [12]

รหัส LDPC มักจะถูกแสดงให้อยู่ในรูปของกราฟซึ่งเรียกว่ากราฟของแทนเนอร์ โดยที่ กราฟของแทนเนอร์คือกราฟที่ประกอบด้วยสองเซตของโหนด (node or vertex) ซึ่งประกอบด้วย เซตที่หนึ่งเรียกว่าบิตโหนด (bit node) โดยที่บิตโหนดจะมีทั้งหมด n โหนด ซึ่งแสดงถึง n บิตของคำรหัส และเซตที่สองจะถูกเรียกว่าเช็คโหนด (check node) แสดงถึง parity-check equation จึงมีจำนวน node ทั้งหมดเท่ากับจำนวนแถวของ parity-check matrix ซึ่งแทนด้วย m โดยที่กราฟนี้จะมีเส้นเชื่อม (edge) ระหว่างสองเซตของโหนดดังกล่าว ซึ่งมีความสัมพันธ์กับตำแหน่งของสมาชิก 1 ที่ปรากฏใน parity-check matrix โดยที่ parity-check matrix ภาพที่ 32 สามารถแสดงให้อยู่ในรูปของกราฟได้ดังต่อไปนี้



ภาพที่ 33 กราฟแทนเนอร์ที่สอดคล้องกับ *parity-check matrix* ภาพที่ *32*

วงปิด (cycle) ในกราฟ คือ เส้นทาง (path) ของกราฟที่เริ่มและจบลงที่โหนดเดียวกัน และความยาวของ cycle ที่สั้นที่สุดในกราฟจะถูกเรียกว่า เกิร์ท (girth) ดังนั้นในกรณีของกราฟของแทนเนอร์ ความยาวของ cycle ที่สั้นที่สุดที่เป็นไปได้คือ 4 โดยที่ค่า girth นี้จะส่งผลต่อการลู่เข้าสู่คำตอบของการถอดรหัสแบบวนซ้ำ สำหรับรหัส LDPC จากรูปที่ 4 cycle ที่มีความยาวเท่ากับในกราฟแทนเนอร์ แสดงไว้โดยเส้นทึบ

การเข้ารหัสของ LDPC ทำได้โดยการแปลงรูป H ให้อยู่ในรูป Systematic form โดยใช้ Gaussian elimination เพื่อหา G ที่มีความสัมพันธ์กับ H โดยจะยกตัวอย่างการหา G ของรหัส LDPC ที่มี H matrix ภาพที่ 32

การหา Matrix G

1. แปลงให้อยู่ในรูป row-echelon form โดยใช้ row operation จาก Matrix H ในภาพที่ 32 จะได้เป็น

$$\mathbf{H}' = \left[\begin{array}{cccccccccc} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{array}\right]$$

ภาพที่ 34 ตัวอย่าง *parity-check matrix H* ในรูป row-echelon form

2. แปลง matrix ภาพที่ 34 ให้อยู่ในรูป reduced row-echelon form ซึ่งได้เป็น

$$\mathbf{H}^{'} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

ภาพที่ 35 ตัวอย่าง *parity-check matrix H* ในรูป *reduced* row-echelon form

3. กรณีต้องการให้ k-bit แรกของ codeword เป็น data codeword bit สามารถทำได้โดยใช้การสับเปลี่ยนหลัก (column permutation) ซึ่งจะได้เป็น

$$\mathbf{H}_{sys} = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

ภาพที่ 36 ตัวอย่าง *parity-check matrix H* หลังจากการทำ column permutation

ซึ่งในกรณีนี้จะต้องทำการสับเปลี่ยนหลักของ matrix H ในภาพที่ 32 ด้วย (ในขั้นตอนนี้คุณสมบัติของ H ไม่เปลี่ยนเพียงแต่ทำให้ลำดับของ parity bit เปลี่ยน) ดังนั้นสุดท้ายแล้วจะได้ H ที่ใช้ในการถอดรหัสเป็น [13]

$$\mathbf{H}_D = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

ภาพที่ 37 ตัวอย่าง parity-check matrix H_D

4. สุดท้ายสามารถหา G ได้จากภาพที่ 36 ตามความสัมพันธ์สมการ $\mathbf{G}_{sys} = [I_{k\times k} \quad P_{k\times(n-k)}]$ และสมการ $\mathbf{H}_{sys} = [P^T{}_{(n-k)\times k} \quad I_{(n-k)}]$ ซึ่ง G ที่ได้จะมีความสัมพันธ์กับ H_D (รวมถึง H_{sys} ด้วย) ตามสมการ $\mathbf{HG}^{\mathbf{T}} = \mathbf{0}$

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

ภาพที่ 38 Matrix ตัวสร้างรหัส G

Matrix G ในภาพที่ 38คือ matrix ตัวสร้างรหัส สมมุต data codeword คือ m=[1 0 1 0 1] ตามความสัมพันธ์สมการ c=mG จะได้ Error correction codeword เป็น c=[1 0 1 0 1 1 0 1 0 0] ดังนั้น จะได้ว่า code rate ของคำรหัส LDPC ที่มี parity-check matrix ภาพที่ 32 คือ R=5/10

2.5 การถอดรหัสแบบวนซ้ำสำหรับรหัสตรอจสอบสภาวะคู่หรือคี่ชนิดความหนาแน่นต่ำ

ความแตกต่างของรหัส LDPC จากรหัสบล็อกแบบเชิงเส้นอื่นๆ คือ รหัส LDPC จะใช้การถอดรหัสแบบวนซ้ำ (iterative decoding) ซึ่งมักถูกเรียกว่า message passing algorithm (MPA), sum product algorithm (SPA) หรือ belief propagation algorithm (BPA) [3], [14] ซึ่งความหนาแน่นที่ต่ำของ parity check matrix ส่งผลให้ความซับซ้อนในการถอดรหัสนั้นต่ำตามไปด้วย

การอ้างถึงค่าของเมตริกซ์ที่ตำแหน่งต่างๆ จะใช้สัญลักษณ์ **H**(m,n) เมื่อ m และ n หมายถึงแถวและหลักตามลำดับ เนื่องจาก parity check matrix ของรหัส LDPC มีความหนาแน่นต่ำ จึงนิยมใช้สัญลักษณ์ในการแทนตำแหน่งของสมาชิกที่ไม่เป็น 0 โดยสัญลักษณ์ที่ใช้มีดังต่อไปนี้

a. N_m คือเซตของสมาชิกในแถวที่ m ใน **H** ที่ไม่เป็น 0
b. $N_{m/n}$ คือเซตของสมาชิกในแถวที่ m ยกเว้นหลักที่ n ใน **H** ที่ไม่เป็น 0
c. M_n คือเซตของสมาชิกในหลักที่ n ใน **H** ที่ไม่เป็น 0
d. $M_{n/m}$ คือเซตของสมาชิกในหลักที่ n ยกเว้นแถวที่ m ใน **H** ที่ไม่เป็น 0 เช่น หากต้องการระบุเซตของสมาชิกที่ไม่เป็น 0 ของ **H** จะได้ว่า

$N_1 = \{3,6,7,9,10\}$, $N_{1/6} = \{3,7,9,10\}$, $M_3 = \{1,3,5\}$ และ $M_{3/5} = \{1,3\}$ เป็นต้น

กรณีที่เป็นการส่งข้อมูลผ่านช่องสัญญาณที่มีการรบกวนแบบ AWGN (additive white Gaussian noise) และใช้การมอดูเลตทางเฟส (binary phase shift keying) คำรหัสจะถูกเปลี่ยนไปเป็นสัญญาณส่ง t (transmitted signal) ด้วยสมการ

$$t_n = (2c_n - 1)a \qquad (2.14)$$

เมื่อ a คือ แอมปลิจูด (amplitude) ของสัญญาณ สัญญาณส่ง t จะถูกส่งผ่านช่องสัญญาณที่มีการรบกวนแบบ AWGN โดยที่ความแปรปรวน (avariance) $\sigma^2 = N_0/2$ ที่ภาครับจะได้รับสัญญาณรับ r (received signal) และจะทำการคำนวณหา channel posterior probability ด้วยสมการด้านล่างนี้

$$P(C_n = 1 \mid r_n) = \frac{1}{1+e^{-2ar_n/\sigma^2}} \quad (2.15)$$

และสุดท้ายก็จะทำการหาค่าประมาณของคำรหัส (estimated codeword ในที่นี้แทนด้วย $\hat{C}_n$) $_n$) โดยวงจรตรวจขีดเริ่มเปลี่ยนซึ่งจะแสดงการส่งผ่านข้อมูลในกรณีนี้ด้วยตัวอย่างต่อไปนี้

ตัวอย่างที่ 2 การส่งข้อมูลผ่านช่องสัญญาณ AWGN โดยใช้การโมดูเลตทางเฟส

1. จากเมตริกซ์ตัวสร้างรหัสในภาพที่ 38 เมื่อให้บิตข่าวสารคือ m = [1 0 1 0 1] เมื่อเข้ารหัสแล้วจะได้คำรหัสเป็น

$$C = [\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 0\ ] \quad (2.16)$$

2. คำรหัส C จะถูกเปลี่ยนไปเป็นสัญญาณส่ง t ด้วยสมการ (2.14) โดยให้แอมปลิจูด a = 1 จะได้สัญญาณส่งเป็น

$$\boldsymbol{t} = [\ 1\ \ -1\ \ 1\ \ -1\ \ 1\ \ 1\ \ -1\ \ 1\ \ -1\ \ -1\ ] \quad (2.17)$$

3. เมื่อส่งผ่านช่องสัญญาณที่มีการรบกวนแบบ AWGN ที่มี ในที่นี้ให้ค่าความแปรปรวน $\sigma^2 = 1$ ที่ภาครับได้รับสัญญาณรับ r เป็น

$$\boldsymbol{r} = [\ 2.01\ \ 2.58\ \ 0.92\ \ -1.68\ \ -0.02\ \ -0.23\ \ -0.71\ \ 0.571\ \ -0.94\ \ -1.36\ ] \quad (2.18)$$

4. ภาครับทำการคำนวณหา Channel Posterior Probability ด้วยสมการ (2.15) และได้ค่าเป็น

$$\boldsymbol{P(C = 1 \mid r)} = [\ 0.98\ \ 0.01\ \ 0.86\ \ 0.03\ \ 0.48\ \ 0.38\ \ 0.19\ \ 0.75\ \ 0.13\ \ 0.06\ ] \quad (2.19)$$

5. เมื่อให้ขีดเริ่มเปลี่ยนเท่ากับ 0.5 จะได้ค่าประมาณของคำรหัสเป็น

$$\hat{\boldsymbol{C}} = [\ 1\ \ 0\ \ 1\ \ 0\ \ \underline{0}\ \ \underline{0}\ \ 0\ \ 1\ \ 0\ \ 0\ ] \quad (2.20)$$

ซึ่งสัญญาณรบกวนก่อให้เกิดข้อผิดพลาดขึ้น 2 บิต ได้แก่ C_5 และ C_6 (แสดงโดยบิตที่ขีดเส้นใต้)

การถอดรหัสของรหัส LDPC จะใช้หลักการของ BPA ซึ่งเป็นการถอดรหัสแบบซอฟท์ (soft decision coding) โดยจะใช้ข้อมูลจากช่องสัญญาณ (ซึ่งก็คือสัญญารับ) เป็นข้อมูลในการถอดรหัส อัลกอริทึม BPA จะทำการแพร่กระจาย (propagate) ข้อมูลที่ได้รับจากช่องสัญญาณไปตาม Tanner's Graph จนกว่าคำรหัสที่ถูกถอดรหัสนั้นจะทำให้ทุก parity –check equation เป็นจริงหรือทำงานเกินจำนวนรอบสูงสุดที่ได้กำหนดไว้ ซึ่งสามารถสรุปวิธีการถอดรหัสโดยใช้อัลกอริทึม BPA ได้ดังนี้

1. กำหนดค่าเริ่มต้นโดยให้ $p_n(1) = P(c_n = 1 \mid r_n)$ และกำหนดจำนวนรอบสูงสุดในการถอดซึ่งในที่นี้แทนด้วยสัญลักษณ์ $\#_{max}$

2. เมื่อกำหนดให้ใช้ H ในการถอดรหัส สำหรับทุกๆ ค่าที่ H $(m,n) \neq 0$, บิตโหนดจะทำการส่งข้อมูลที่ได้จากช่องสัญญาณไปให้ยังเช็คโหนดโดยข้อมูลนั้นคือ

$$q_{mn}(1) = p_n(1) \tag{2.21}$$

$$q_{mn}(0) = 1 - q_{mn}(1) \tag{2.22}$$

3. สำหรับทุก ๆ ค่าที่ H $(m,n) \neq 0$, เช็คโหนดจะทำการคำนวณข้อมูลที่ได้รับและส่งกลับไปให้ยังบิตโหนดโดยข้อมูลนั้นคือ

$$r_{mn}(0) = \frac{1}{2} + \frac{1}{2} \prod_{n \in N_{m \setminus n}} (1 - 2q_{mn}(1)) \tag{2.23}$$

$$r_{mn}(1) = 1 - r_{mn}(0) \tag{2.24}$$

4. สำหรับทุก ๆ ค่าที่ H $(m,n) \neq 0$ บิตโหนดจะปรับปรุงข้อมูลด้วยข้อมูลที่ได้รับจากเช็คโหนดในขั้นตอนที่ 3 ด้วยสมการต่อไปนี้

$$q_{mn}(0) = \alpha_{mn}(1 - p_n(1)) \prod_{m \in M_{n \setminus m}} (r_{mn}(0)) \tag{2.25}$$

$$q_{mn}(1) = \alpha_{mn} p_n(1) \prod_{m \in M_{n \setminus m}} (r_{mn}(1)) \tag{2.26}$$

โดยที่ α_{mn} คือค่าคงที่ที่เลือกมาเพื่อให้สมการ $q_{mn}(0) + q_{mn}(1) = 1$ เป็นจริงและต่อไปทำการคำนวณหา

$$q_n(0) = \alpha_n(1-p_n(1)) \prod_{m \in M_n} (r_{mn}(0)) \tag{2.27}$$

$m \in M_n$

$$q_n(1) = \alpha_n p_n(1) \prod (r_{mn}(1)) \quad (2.28)$$

โดยที่ค่า α_n คือค่าคงที่ที่เลือกมาเพื่อให้สมการ $q_n(0) + q_n(1) = 1$ เป็นจริง

5. ทำการตัดสินใจเพื่อหาค่าประมาณของคำรหัสโดยใช้เงื่อนไขด้านล่างดังนี้

$$\hat{\boldsymbol{c}}_n = \begin{cases} 1, & q_n(1) > q_n(0) \\ 0, & q_n(0) \geq q_n(1) \end{cases} \quad (2.29)$$

โดยอัลกอริทึมจะหยุดก็ต่อเมื่อ $H\hat{\boldsymbol{c}}_n^T = 0$ หรือรอบการคำนวณเกินค่า $\#_{max}$ ที่ได้กำหนดเอาไว้โดยถ้าเงื่อนไขทั้งสองข้อยังไม่เป็นจริงอัลกอริทึมก็จะวนกลับไปคำนวณในขั้นตอนที่ 3 ซึ่งจะแสดงวิธีการถอดรหัสด้วยอัลกอริทึมนี้อย่างละเอียดด้วยตัวอย่างต่อไปนี้

ตัวอย่างที่ 3 การถอดรหัสด้วยอัลกอริทึม BPA
จากข้อมูลในตัวอย่างที่ 1 และตัวอย่างที่ 2 เมื่อใช้ H ตามภาพที่ 37 ในการถอดรหัสจะได้ว่า

1 . ให้ $p_n(1) = P(c_n = 1 \mid r_n)$ และ $\#_{max} = 10$ จะได้ว่า

$$p_n(1) = [\,0.98 \quad 0.01 \quad 0.86 \quad 0.03 \quad 0.48 \quad 0.38 \quad 0.19 \quad 0.75 \quad 0.13 \quad 0.06\,] \quad (2.30)$$

2 . ให้ $q_{mn}(1) = p_n(1)$ จะได้ว่า

$$q_{mn}(1)$$

$$\begin{pmatrix} & & 0.86 & & & 0.38 & 0.19 & & 0.13 & 0.06 \\ 0.98 & 0.01 & & & & & 0.19 & 0.75 & & 0.06 \\ & & 0.86 & 0.03 & 0.48 & & & & 0.13 & \\ 0.98 & 0.01 & & 0.03 & & 0.38 & 0.19 & & & \\ 0.98 & & 0.86 & & 0.48 & & & 0.75 & & \end{pmatrix} \quad (2.31)$$

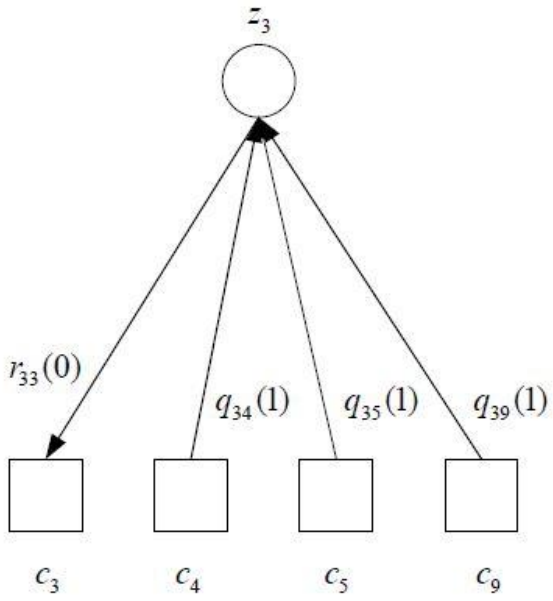
3 . คำนวณหา $r_{mn}(0)$ และ $r_{mn}(1)$, ซึ่งในที่นี่จะยกตัวอย่างแค่ $r_{33}(0)$

$$r_{33}(0) = \frac{1}{2} + \frac{1}{2} \prod_{\underline{n} \in n_{3 \setminus 3}} (1 - 2q_{3n}(1))$$

$$= \frac{1}{2} + \frac{1}{2} \left[(1 - 2q_{34}) (1 - 2q_{35}) (1 - 2q_{39}) \right]$$

$$= 0.51$$

ซึ่งในขั้นตอนนี้สามารถอธิบายด้วยภาพที่ 39



ภาพที่ 39 กราฟแสดงการคำนวณ $r_{33}(0)$

เมื่อคำนวณค่า $r_{mn}(0)$ จนครบทุกค่าจะได้เป็น

$$r_{mn}(0) = \begin{pmatrix} & & 0.55 & & & 0.36 & 0.45 & & 0.45 & 0.46 \\ 0.36 & 0.63 & & & & & 0.72 & 0.24 & & 0.65 \\ & & 0.51 & 0.49 & 0.25 & & & & 0.49 & \\ 0.56 & 0.44 & & 0.43 & & 0.23 & 0.40 & & & \end{pmatrix}$$

0.50 0.51 0.32 0.51

(2.32)

4 . คำนวณหา $q_{mn}(0)$ และ $q_{mn}(1)$, ในที่นี้จะยกตัวอย่างแค่ $q_{33}(0)$ และ $q_{33}(1)$

$$q_{33}(0) = \alpha_{33}\,(\,1-p_3(1)\,)\prod_{\underline{m}\epsilon M_{3\backslash 3}}(r_{m3}(0))$$

$$= \alpha_{33}\,(\,1-p_3(1)\,)\,[\,(\,r_{13}(0)\,)\,(\,r_{53}(0)\,)\,]$$

$$= 0.04\,\alpha_{33}$$

$$q_{33}(1) = \alpha_{33}\,(\,p_3(1)\,)\prod_{\underline{m}\epsilon M_{3\backslash 3}}(r_{m3}(1))$$

$$= \alpha_{33}\,(p_3(1)\,)\,[\,(\,r_{13}(1)\,)\,(\,r_{53}(1)\,)\,]$$

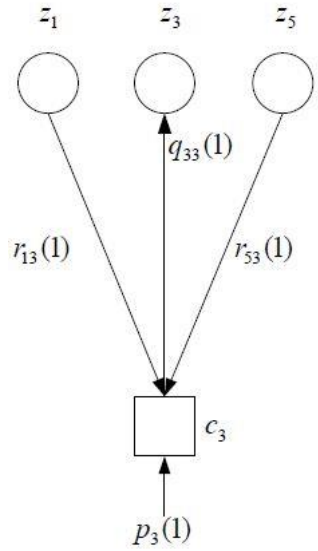
$$= 0.19\,\alpha_{33}$$

$$0.04\,\alpha_{33} + 0.19\,\alpha_{33} = 1$$

$$\alpha_{33} = 4.37$$

$$\therefore\ q_{33}(0) = 0.16 \text{ และ } q_{33}(1) = 0.84$$

ซึ่งในขั้นตอนนี้สามารถอธิบายด้วยรูปภาพที่ 40



ภาพที่ 40 กราฟแสดงการคำนวณ $q_{33}(1)$

เมื่อคำนวณ $q_{mn}(1)$ จนครบทุกค่าจะได้เป็น

$$q_{mn}(1) = \begin{pmatrix} & & 0.86 & & & 0.68 & 0.13 & & 0.14 & 0.33 \\ 0.98 & 0.01 & & & & & 0.31 & 0.75 & & 0.07 \\ & & 0.51 & 0.49 & 0.25 & & & & 0.49 & \\ 0.99 & 0.01 & & 0.03 & & 0.53 & 0.11 & & & \\ 0.99 & & 0.83 & & 0.74 & & & 0.91 & & \end{pmatrix} \quad (2.33)$$

ลำดับต่อมาคำนวณหา $q_n(0)$ และ $q_n(1)$, ในที่นี้จะยกตัวอย่างแค่ $q_5(0)$ และ $q_5(1)$

$$q_5(0) = \alpha_5(1 - p_5(1)) \prod_{\underline{m} \epsilon M_5} (r_{m5}(0))$$

$$= \alpha_5 (1 - p_5(1)) [(r_{35}(0)) (r_{55}(0))]$$

$$= 0.04\ \alpha_5$$

$$q_5(1) = \alpha_5(p_5(1)) \prod_{\underline{m} \epsilon M_5} (r_{m5}(1))$$

$$= \alpha_5 (p_5(1)) [(r_{35}(1)) (r_{55}(1))]$$

$$= 0.24\ \alpha_5$$

$$0.04\ \alpha_5 + 0.24\ \alpha_5 = 1$$

$$\alpha_5 = 3.49$$

$$\therefore \quad q_5(0) = 0.15 \text{ และ } q_5(1) = 0.85$$

เมื่อคำนวณจนครบทุกค่าจะได้

$$q_n(1) = [\ 0.98\ \ 0.01\ \ 0.83\ \ 0.04\ \ 0.85\ \ 0.79\ \ 0.15\ \ 0.90\ \ 0.16\ \ 0.04\ ] \quad (2.34)$$

5 . ทำการตัดสินใจเพื่อหาค่าประมาณของคำรหัสโดยใช้สมการที่ (2.29) จะได้ว่า

$$\hat{c}_n = [\ 1\ \ 0\ \ 1\ \ 0\ \ 1\ \ 1\ \ 0\ \ 1\ \ 0\ \ 0\ ] \quad (2.35)$$

ค่าประมาณของคำรหัสนี้ทำให้สมการ $H\hat{c}_n^T = 0$ เป็นจริง ส่งผลให้การถอดรหัสสิ้นสุดลงในกรอบแรก และในตัวอย่างนี้ค่าประมาณของคำรหัสคือคำรหัสที่ถูกต้อง ดังนั้นในตัวอย่างนี้อัลกอริทึม BPA สามารถแก้ไขข้อผิดพลาดที่เกิดขึ้นได้ในทั้งสองบิต

อัลกอริทึม BPA ที่ได้กล่าวไปแล้ว จะเห็นได้ว่าเกิดการคูณกันของความน่าจะเป็นหลายครั้ง เมื่อใช้รหัสที่มีความยาวของบล็อกสูง ๆ จะส่งผลให้ค่าที่อยู่ในรอบการคำนวณมีค่าเข้าใกล้ศูนย์ซึ่งส่งผลต่อเสถียรภาพเชิงตัวเลข (numerical stability) ปัญหาในจุดนี้สามารถแก้ไขได้โดยเปลี่ยนการถอดรหัสนี้ให้อยู่ในรูปของอัตราส่วนของความน่าจะเป็นที่อยู่ในเทอมของล็อก (log likelihood ratio) ซึ่งจะเปลี่ยนการคูณเป็นการบวกแทน การถอดรหัสโดยใช้ข้อมูลที่อยู่ในเทอม log likelihood ratio นี้มักจะถูกเรียกว่า sum – product algorithm (SPA) ซึ่งมีหลักการในการถอดรหัสสรุปได้ดังนี้

1. กำหนดให้ r_n คือสัญญาณรับ และจำนวนรอบสูงสุดในการถอดรหัสคือ $\#_{max}$
2. คำนวณค่า Channel Reliability ตามสมการด้านล่างนี้

$$R_c = \frac{2\sqrt{a}}{\sigma^2} \quad (2.36)$$

เมื่อกำหนดให้ใช้ H ในการถอดรหัส สำหรับทุกๆค่าที่ $H(m,n) \neq 0$

$$\eta_{mn} = 0 \quad (2.37)$$

3. สำหรับทุกๆ ค่าที่ **H *(m,n)* ≠ 0** ,

$$\eta_{mn} = -2\tanh^{-1}\left(\prod_{\underline{n}\epsilon N_{m/n}}\tanh\left(-\frac{\lambda_j-\eta_{mn}}{2}\right)\right) \quad (2.38)$$

4. . สำหรับทุกๆ ค่าที่ **H *(m,n)* ≠ 0** ,

$$\lambda_n = R_c r_n + \sum_{m\epsilon M_n}\eta_{mn} \quad (2.39)$$

5. ทำการตัดสินใจเพื่อหาค่าประมาณของคำรหัสโดย

$$\hat{c}_n = \begin{cases} 1, & \lambda_n > 0 \\ 0, & \lambda_n \leq 0 \end{cases} \quad (2.40)$$

โดยอัลกอริทึม BPA จะหยุดก็ต่อเมื่อ $\mathbf{H}\hat{\boldsymbol{c}}_{\boldsymbol{n}}^{\boldsymbol{T}} = \mathbf{0}$ หรือรอบการคำนวณเกินค่า $\#_{max}$ ที่ได้กำหนดเอาไว้โดยถ้าเงื่อนไขทั้งสองข้อยังไม่เป็นจริงอัลกอริทึมจะวนกลับไปคำนวณในขั้นตอนที่ 3

โดยที่กล่าวมาทั้งหมด นี้คือหลักการเบื้องต้นของการเข้ารหัสช่องสัญญาณในระบบสื่อสารเชิงดิจิทัล การเข้ารหัสบล็อกแบบเชิงเส้นและรหัส LDPC โดยรหัส LDPC นี้มีศักยภาพในการแก้ไขข้อผิดพลาดเข้าใกล้ค่าความจุของสัญญาณ การเข้ารหัสช่องสัญญาณโดยใช้รหัส LDPC จึงเป็นวิธีการป้องกันการผิดพลาดล่วงหน้าที่สามารถแก้ไขความผิดพลาดของข้อมูลที่ได้รับจากสัญญาณรบกวนอย่างมีประสิทธิภาพ ซึ่งปัจจุบันรหัส LDPC ได้ถูกประยุกต์ใช้งานในระบบสื่อสารเชิงดิจิตัลอย่างแพร่หลาย เช่น ระบบการส่งสัญญาณโทรทัศน์ระบบดิจิตัลผ่านดาวเทียมและระบบการสื่อสารเชิงแสง เป็นต้น

2.4 รหัสแก้ไขข้อผิดพลาด Reed Solomon (RS)

รหัสรีดโซโลมอนเป็นรหัสที่ได้รับการยอมรับและใช้งานอย่างแพร่หลายทั้งในแง่ของการสื่อสารข้อมูลและเก็บบันทึกข้อมูล แม้รหัสรีดโซโลมอนจะจัดอยู่ในรหัส BCH ทั้งข้อมูลของรหัสและตำแหน่งของความผิดพลาดนั้นจะอยู่ในสนามขยาย (Extension field) ดังนั้นรหัสรีดโซโลมอนจึงเป็นรหัสวนรอบแบบ Non-binary คือแต่ละสัญญาลักษณ์จะมี m บิต (ปกติ m เป็นความยาวบวกและมากกว่า 2) ในบรรดารหัสเชิงเส้นรีดโซโลมอนจะมีระยะทางของรหัสต่ำสุด คือ

$$\text{Minimum distance : } d_{min} = n - k + 1 \quad (2.41)$$

ปกติแล้วความยาวบล็อคของรีดโซโลมอน คือ $n=2^m - 1$ แต่อาจขยายได้มากที่สุดถึง $n = 2^m + 1$

รหัสรีดโซโลมอนได้ถูกนำเสนอโดย Irving Reed และ Gus Solomon ในปี ค.ศ. 1960 รหัสรีดโซโลมอนแทนด้วย RS(n,k) คือความยาวบล็อค n สัญญลักษณ์ และข้อมูลจำนวน k สัญญลักษณ์ โดย (n-k) = 2t เมื่อ t คือจำนวนสูงสุดของสัญญลักษณ์ที่อาจผิดพลาดได้ ดังนั้น

$$t = \left(\frac{d_{min}-1}{2}\right) = \left(\frac{n-k}{2}\right) \quad (2.42)$$

และเมื่อ ρ คือ Erasure รหัสรีดโซโลมอนจะแก้ไขได้คือ

$$\rho = (n - k) = 2t \quad (2.43)$$

หรือเมื่อ α และ β คือจำนวนสัญญลักษณ์ผิดพลาดและจำนวนสัญญลักษณ์ ที่โดนลบแท้จริง ตามลำดับ

$$2\alpha + \beta \leq (n - k) \quad (2.44)$$

1. การเข้ารหัสรีดโซโลมอน

พหุนามกำเนิด (generator polynomial) ของรหัสรีดโซโลมอน เขียนได้คือ

$$g(x) = \sum_{i=0}^{2t} g_i\, x^i = g_0 + g_1 x + g_2 x^2 + \cdots + g_{2t} x^{2t} \quad (2.45)$$

เมื่อ $g_i \in GF(2^m)$ ดีกรีของพหุนามจำนวนพหุนามกำเนิดคือ $2t$ ดังนั้นจึงสามารถที่จะกำหนดจากจำนวนรากที่ต่อเนื่องจำนวน $2t$ ราก หรือเมื่อเขียนเป็นรูปทั่วไป

$$g(x) = \prod_{i=j_0}^{j_0+2t-1} (x - \alpha^i) \quad (2.46)$$

ปกติแล้วจะให้ $j_0 = 1$ แต่บางกรณี $j_0 \neq 1$ ก็ได้ เช่น $2(j_0 + t) = 2^m$ในกรณีต้องการสัมประสิทธิ์ $g(x)$ แบบสมมาตร ในตัวอย่างนี้จะใช้ RS(15,9) ในเมื่อ $GF(2^4)$ โดย $p(x) = x^4 + x + 1$

ตัวอย่าง

สำหรับ RS(15,9) , (n – k) = 2t = 6 แต่ละสัญญลักษณ์ จะมี 4 บิต 1 บล็อคมี 15 สัญญลักษณ์ หรือ 60 บิต สามารถแก้ไขความผิดพลาดได้สูงสุด 3 สัญญลักษณ์ หรือ 12 บิต โดยส่วนข้อมูลมี 9 สัญญลักษณ์ หรือ 36 บิต

กรณี $j_0 = 1$ จะหาพนุนามกำเนิดได้คือ

$$g(x) = \prod_{i=j_0}^{j_0+2t-1} (x-\alpha^i) = \prod_{i=1}^{6} (x-\alpha^i)$$
$$= (x-\alpha)(x-\alpha^2)(x-\alpha^3)(x-\alpha^4)(x-\alpha^5)(x-\alpha^6)$$
$$= \alpha^6 + \alpha^9 x + \alpha^6 x^2 + \alpha^4 x^3 + \alpha^{14} x^4 + \alpha^{10} x^5 + x^6$$
$$= 12 + 10x + 12x^2 + 3x^3 + 9x^4 + 7x^5 + x^6$$

2.5 การวัดประสิทธิภาพโดย Bit Error Rate

Bit Error Rate (BER) เป็นค่าที่แสดงความผิดพลาดของการรับข้อมูลในระบบดิจิทัล ตั้งแต่ 0 คือไม่ผิดพลาดเลยไปจนถึงหลักร้อย หลักพัน หรือ หลักหมื่น เป็นต้น ค่านี้มีประโยชน์ ในส่วนของอุปกรณ์ดิจิทัล เพื่อให้รู้ว่า ในการติดตั้งอุปกรณ์ดิจิทัลนั้นมีความสมบูรณ์มากน้อยเพียงใด ค่า BER จะทำให้เรารู้ถึงรายละเอียดของสัญญาณ หาก BER เป็น 0 หรือต่ำมาก แสดงว่า การรับสารดิจิทัลได้สมบูรณ์ครบถ้วนในแต่ละบิท

1. การรับ ส่งข้อมูลในระบบสื่อสาร

ส่ง	1	รับ	1	=	ถูกต้อง
ส่ง	0	รับ	0	=	ถูกต้อง
ส่ง	1	รับ	0	=	เกิดความผิดพลาด
ส่ง	0	รับ	1	=	เกิดความผิดพลาด

2. การคำนวณ BER

   BER = จำนวนบิตที่ผิดพลาด/จำนวนบิตที่ส่งทั้งหมด

ตัวอย่าง ส่งข้อมูลทั้งหมด 1 ล้านบิต มีข้อมูลผิดพลาด 1 บิต

BER = 1/1,000,000 = 10^{-6}

ปัจจุบัน ITU กำหนดให้ BER <= 10^{-7}

2.7 สัญญาณรบกวนไวต์เกาส์เซียนแบบบวก (Additive White Gaussian Noise: AWGN)

AWGN เป็นสัญญาณไม่พึงประสงค์ที่ปะปน กับสัญญาณที่ต้องการในลักษณะบวกทับสัญญาณที่ต้องการ ทำให้สัญญาณที่ผู้รับได้รับผิดเพี้ยนไปจากสัญญาณที่ควรจะเป็น โดยขนาดของสัญญาณไม่พึงประสงค์นี้ มีการกระจายโอกาสแบบเกาส์เซียนและมีสเปกตรัมระดับความเข้มที่คงที่ซึ่งมาจากแหล่งกำเนิดในธรรมชาติ จึงเป็นอุปสรรคที่มีอยู่ในทุกระบบการสื่อสาร

2.8 อัตราส่วนสัญญาณต่อสัญญาณรบกวน (Signal to noise Ratio: SNR)

อัตราส่วนระหว่างกำลังของสัญญาณที่ต้องการต่อกำลังของสัญญาณรบกวนที่เข้ามาในระบบ ใช้เป็นหน่วยวัดเพื่อทราบถึงประสิทธิภาพของระบบการสื่อสารว่ามีระดับการใช้กำลังงานในการส่งสัญญาณหรือมีระดับของสัญญาณรบกวนที่เข้ามาในระบบเท่าใด

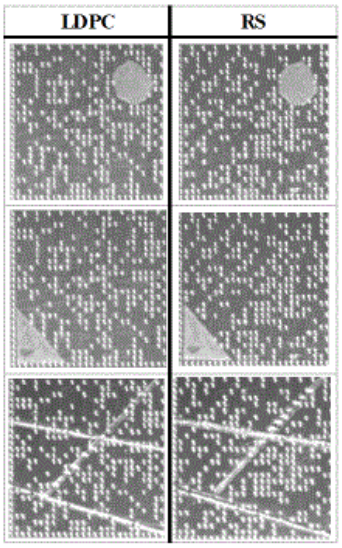
2.9 งานวิจัยที่เกี่ยวข้อง

QR Code เป็นสัญลักษณ์แบบสองมิติ ถูกพัฒนาจากบริษัท Denso Wave [14] และถูกเผยแพร่ในปี 1994 นอกจาก QR Code แล้วยังมีบาร์โค้ดสองมิติบางประเภทถูกพัฒนาอีกมากมาย

ในขณะที่บาร์โค้ดทั่วไปสามารถเก็บได้มากสุด 20 ดิจิตแต่ QR Code สามารถเก็บได้มากกว่าหลายเท่าทำให้ได้ข้อมูลที่มากกว่า QR Code สามารถจุข้อมูลได้ทุกรูปแบบ เช่น Numeric Alphanumeric Kanji Kana Hiragana Symbols Binary และ Control Code สูงถึง 7,089 อักขระแปลงเป็น 1 สัญลักษณ์

ในส่วนของการทำ Error Correction และรหัสต่างๆ ที่ใช้ในการทำ Error Correction ในปัจจุบันงานวิจัยส่วนใหญ่จะมุ่งเน้นไปที่รหัส LDPC โดยการชี้ให้เห็นประสิทธิภาพของ LDPC เทียบกับรหัสแก้ไขข้อผิดพลาดแบบอื่นๆ รวมถึงการนำรหัส LDPC ไปประยุกต์ใช้ในการแก้ไขข้อผิดพลาดของข้อมูลเชิงดิจิทัล ซึ่งผลงานจำนวนหนึ่งได้เสนอออกมานั้นจะขอกล่าวในลักษณะสรุปสั้นๆ เพื่อให้เห็นถึงวิธีการคร่าวๆ ดังนี้

ผลงานวิจัยของ Pross W. **[15]** ซึ่งได้กล่าวถึง 2D Barcode ชนิดหนึ่ง คือ Data Matrix Code (DMC) ที่ถูกใช้ในโรงงานอุตสาหกรรมต่าง ๆ ซึ่งมีโอกาสที่ Barcode นั้นจะถูกทำให้เสียหายโดยปัจจัยหลายอย่าง Pross W. จึงหาวิธีที่จะช่วยเพิ่มประสิทธิภาพในการกู้ข้อมูลกลับมาเมื่อ Barcode นั้นถูกทำให้เสียหาย ซึ่งในงานวิจัยของ Pross W. นั้นก็ได้นำ รหัสแก้ไขข้อผิดพลาดแบบ LDPC มาแทน Reed Solomon ใน DMC โดยทำการทดลอง กับ DMC 23 คู่ แต่ละคู่จะมีการถูกทำให้เปื้อนเสียหายเหมือนกัน



ภาพที่ 41 แสดงตัวอย่างคู่ของ *DMC* ที่มีการถูกทำให้เสียหาย

ผลการทดลอง ภาพที่ 41จะแสดงตัวอย่าง 3 คู่ของ DMC ที่มีการถูกทำให้เปื้อนเสียหาย โดยที่ คู่บนสุดจะสามารถกู้ข้อมูลกลับมาได้สำเร็จทั้งแบบใช้ LDPC และ RS คู่กลางและคู่ล่างนั้น DMC ที่ใช้ LDPC เท่านั้นที่สามารถกู้ข้อมูลกลับมาได้

	LDPC	RS
one succeeded	9	0
both succeeded	6	
both failed	8	

ภาพที่ 42 ตารางเปรียบเทียบความสามารถในการกู้ข้อมูลจาก *DMC 23* คู่

ภาพที่ 42 แสดงผลการถอดรหัสข้อมูลของ DMC ทั้ง 23 คู่ที่ทำการทดลอง จากตาราง 15/23 = 65% ของรหัส LDPC สามารถถอดรหัสจาก DMC ออกมาได้ และเพียง 6/23 = 26% ของรหัส RS (Reed Solomon) ที่สามารถถอดรหัสจาก DMC ได้ แสดงให้เห็นว่าการแก้ไขข้อผิดพลาดโดยใช้รหัส LDPC บน DMC มีความสามารถมากกว่า การแก้ไขข้อผิดพลาดแบบ RS 65/26 = 2.5 เท่า

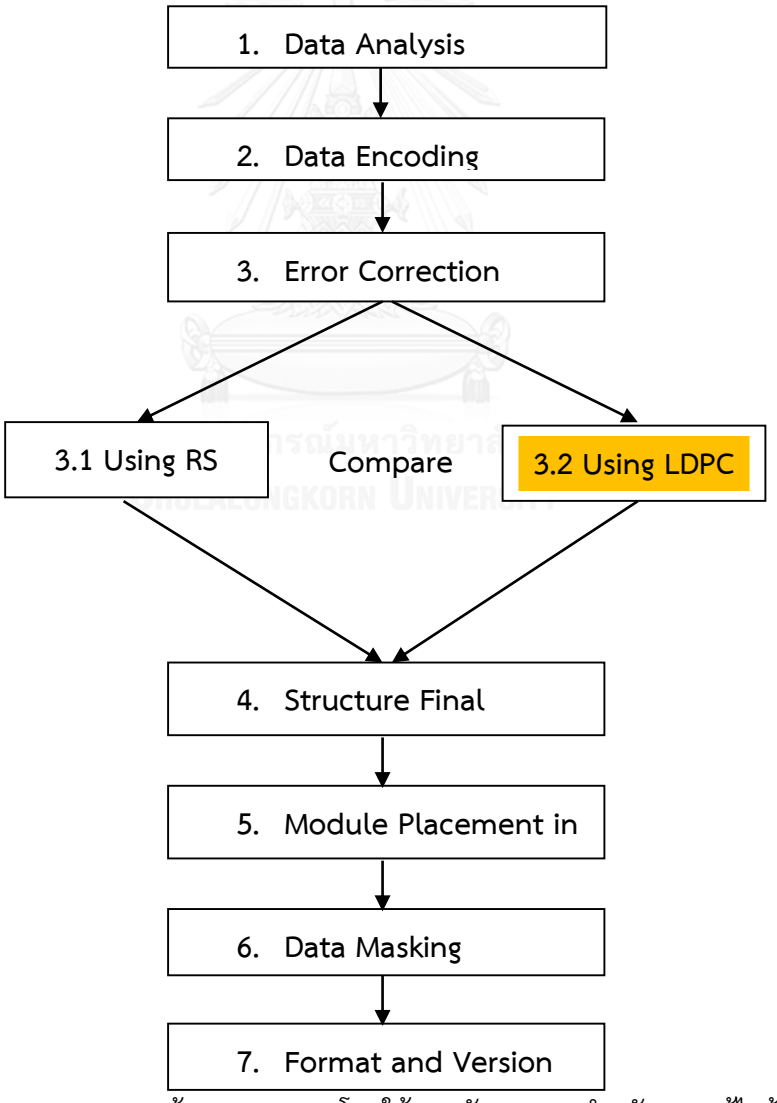
งานวิจัยของ Junbin Chen [10] ได้ทำการเปรียบเทียบประสิทธิภาพของรหัส Non-Binary LDPC ซึ่งเป็น LDPC ชนิดหนึ่ง กับรหัส Reed Solomon โดยทดสอบประสิทธิภาพผ่านช่องสัญญาณรบกวน (Burst Noise channel) และช่องสัญญาณรบกวนเกาส์เซียนแบบขาว (AWGN channel) ผล

การทดลอง Non-Binary LDPC มีประสิทธิภาพเหนือกว่า รหัส Reed Solomon ทั้งการผ่าน Burst Noise channel และผ่าน AWGN channel สามารถแก้ไขข้อผิดพลาดของเสียงสัญญาณได้ดีกว่า และยิ่งไปกว่านั้น Non-Binary LDPC ยังสามารถถูกพัฒนาในเรื่องโครงสร้างของการเข้ารหัสได้อีกด้วย และผู้วิจัยคาดว่า Non-Binary LDPC จะเป็นทางเลือกที่ดีที่จะมาแทนรหัส Reed Solomon.

บทที่ 3

วิธีดำเนินการวิจัย

การดำเนินงานวิจัยประกอบไปด้วยการเข้ารหัสคิวอาร์โค้ดโดยมีการสร้างบิตตรวจสอบด้วยรหัสการแก้ไขข้อผิดพลาด Reed Solomon (RS) และสร้างบิตตรวจสอบด้วยรหัสการแก้ไขข้อผิดพลาด Low density parity check (LDPC) แล้วนำมาเปรียบเทียบประสิทธิภาพของการทำการแก้ไขข้อผิดพลาดเมื่อมี สิ่งรบกวน โดยการเปรียบเทียบ Bit error rate (BER) จากเครื่องมือสำเร็จรูป MATLAB แล้วนำ Codewords ซึ่งประกอบด้วย Data codewords และ Error correction codewords มาทำการจัดเรียงตามโครงสร้างของ QR Code ขั้นตอนการดำเนินงานวิจัยเป็นดังนี้



ภาพที่ 43 กระบวนการสร้าง *QR Code* โดยใช้ *RS* กับ *LDPC* สำหรับการแก้ไขข้อผิดพลาด

ภาพที่ 43 แสดงขั้นตอนการสร้าง QR Code โดยสัญลักษณ์  เป็นขั้นตอนการเข้ารหัสแก้ไขข้อผิดพลาดโดย LDPC โดยโปรแกรม Matlab และสัญลักษณ์  เป็นกระบวนการเข้ารหัสใน QR Code แบบดั้งเดิม โดยการดำเนินงานวิจัยจะมีกระบวนการดังต่อไปนี้

3.1 ขั้นตอนการเปรียบเทียบประสิทธิภาพการแก้ไขข้อผิดพลาดโดย RS และ LDPC โดยสุ่มพหุนาม

ในการทดลองส่วนนี้ จะทดสอบประสิทธิภาพของรหัสการแก้ไขข้อผิดพลาดของ RS และ LDPC โดยใช้ Code rate ที่แตกต่างกันออกไป ข้อมูลที่ทำการเข้ารหัสจะอยู่ในรูปแบบของ Bitstream การวัดประสิทธิภาพจะวัดจาก Bit Error Rate และเวลาที่ใช้ในการแก้ไขข้อผิดพลาด ตัวทำลายข้อมูลจะจำลองโดยใช้ AWGN ซึ่งเป็นการจำลองสัญญาณรบกวนแบบเกาส์เซียน

ตารางที่ 8 ตารางแสดงค่า *Code rate* ที่ใช้ในการเปรียบเทียบประสิทธิภาพ

QR Version	Error correction Level	Number of codewords (n)	Number of data codewords (k)	Code rate (k/n)
1	L	26	19	(19/26)
	M		16	(16/26)
	H		13	(13/26)
	Q		9	(9/26)

จากตารางที่ 8 ในการทดสอบจะใช้ค่า Code rate ต่างๆ โดยค่า Code rate นั้นจะอ้างอิงมาจาก ตารางค่าความจุใน QR Code ของแต่ละเวอร์ชัน ในการทดลองนี้จะใช้ กฎของ QR Code ในเวอร์ชันที่ 1 และ 2 และใช้ Code rate ในสัดส่วน ของจำนวน Data codewords ต่อจำนวน Codewords ทั้งหมดที่สามารถบรรจุได้ใน QR Code เวอร์ชันที่ 1 และ 2

ตารางที่ 9 ตารางแสดงค่า *EbNo* ที่ใช้เป็นสัญญาณในการรบกวน

QR Version	QR Error Correction level	Code Rate	EbNo
1	L	(19/26)	1
			2
			3

			4
			5
			6
			7
			8
			10
	M	(16/26)	1
			2
			3
			4
			5
			6
			7
			8
			10
	H	(13/26)	1
			2
			3
			4
			5
			6
			7
			8
			10
	Q	(9/26)	1
			2
			3
			4
			5
			6

			7
			8
			10

จากตารางที่ 9 จะเป็นค่า E_bN_o หรือ SNR ต่อ 1 บิต เป็นสัญญาณรบกวนที่เปลี่ยนแปลงข้อมูลที่เราทำการเข้ารหัสแต่ละ QR Code เวอร์ชันจะทำการทดสอบในทุกๆ ระดับ Error correction และ Code rate จะขึ้นอยู่กับ จำนวน Data codewords และจำนวน Codewords ทั้งหมด โดยจะแต่ละ Code rate จะถูกทดสอบด้วย ค่า E_bN_o 1-10 และประเมินประสิทธิภาพโดย BER และเวลาที่ใช้ในการเข้ารหัสและถอดรหัส

i. ข้อมูลที่ใช้ในการทดสอบ

ข้อมูลที่ใช้ในการทดสอบจะเป็น Bitstream ที่สร้างขึ้นมาจากหลักเกณฑ์การแปลงเป็น Bitstream ของ QR Code คำว่า “SUMAMA” ดังตัวอย่างหัวข้อ 2.1.7

“00100000 00110101 00001010 11111010 00111110 10000000 11101100 00010001 11101100 00010001 11101100 00010001 11101100 00010001 11101100 00010001 11101100 00010001 11101100”

ii. โปรแกรมที่ใช้ในการทดสอบ

การทดสอบจะใช้โปรแกรม MATLAB เวอร์ชัน R2015a

iii. ขั้นตอนการทดสอบประสิทธิภาพ

ก. โปรแกรมจำลองจะทำการทดสอบโดยเริ่มด้วย Code rate แรก (19/26) ตามตารางที่ 8 โดยค่า k = 19 และค่า n = 26

```
k = 19 %k = input('k:');
n = 26 %input('n:');
%%
fprintf('Message\n');
fprintf('*********************************\n');
message = [0 0 1 0 0 0 0 0;
0 0 1 1 0 0 1 0;
0 0 1 0 1 1 0 1;
0 0 1 1 1 0 1 1;
0 0 1 1 0 0 0 1;
0 0 1 1 1 1 0 0;
0 0 0 0 0 0 0 0;
0 0 1 1 0 0 1 1;
0 0 1 1 0 0 1 1;
1 1 0 0 1 1 0 0;
1 1 0 0 1 1 0 0;
1 1 1 0 1 1 0 0;
0 0 0 1 0 0 0 1;
0 0 1 1 0 0 1 1;
0 0 1 1 0 0 1 1;
1 1 1 0 1 1 0 0;
0 0 0 1 0 0 0 1;
1 1 1 0 1 1 0 0;
    0 0 0 1 0 0 0 1]
[nr, nc] = size(message);
```

ภาพที่ 44 แสดงการใส่ค่า *Code rate* ที่ใช้ในการทดลอง

ข. เมื่อทำการ Run โปรแกรม ค่า Bitstream ที่ใช้ทดสอบจะถูกเข้ารหัส ดังภาพที่ 45 และ ภาพที่ 46

```
LDPC Code Word
*********************************

coded_word_ldpc =

     0     0     1     0     0     0     0     0     1     0
     0     0     1     1     0     0     1     0     0     1
     0     0     1     0     1     1     0     1     0     0
     0     0     1     1     1     0     1     1     1     0
     0     0     1     1     0     0     0     1     1     0
     0     0     1     1     1     1     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     1     1     0     0     1     1     0     0
     0     0     1     1     0     0     1     1     0     0
     1     1     0     0     1     1     0     0     0     0
     1     1     0     0     1     1     0     0     0     0
     1     1     1     0     1     1     0     0     1     0
     0     0     0     1     0     0     0     1     0     0
     0     0     1     1     0     0     1     1     0     0
     0     0     1     1     0     0     1     1     0     0
     1     1     1     0     1     1     0     0     1     0
     0     0     0     1     0     0     0     1     0     0
     1     1     1     0     1     1     0     0     1     0
     0     0     0     1     0     0     0     1     0     0
```

ภาพที่ 45 แสดงค่า *Codeword* หลังจากการเข้ารหัสโดย *LDPC*

```
RS Code Word
**********************************

coded_word_rs =

     0     0     1     0     0     0     0     0     1     1
     0     0     1     1     0     0     1     0     1     1
     0     0     1     0     1     1     0     1     1     1
     0     0     1     1     1     0     1     1     1     1
     0     0     1     1     0     0     0     1     1     1
     0     0     1     1     1     1     0     0     1     0
     0     0     0     0     0     0     0     0     0     0
     0     0     1     1     0     0     1     1     1     1
     0     0     1     1     0     0     1     1     1     1
     1     1     0     0     1     1     0     0     1     1
     1     1     0     0     1     1     0     0     1     1
     1     1     1     0     1     1     0     0     0     1
     0     0     0     1     0     0     0     1     1     1
     0     0     1     1     0     0     1     1     1     1
     0     0     1     1     0     0     1     1     1     1
     1     1     1     0     1     1     0     0     0     1
     0     0     0     1     0     0     0     1     1     1
     1     1     1     0     1     1     0     0     0     1
     0     0     0     1     0     0     0     1     1     1
```

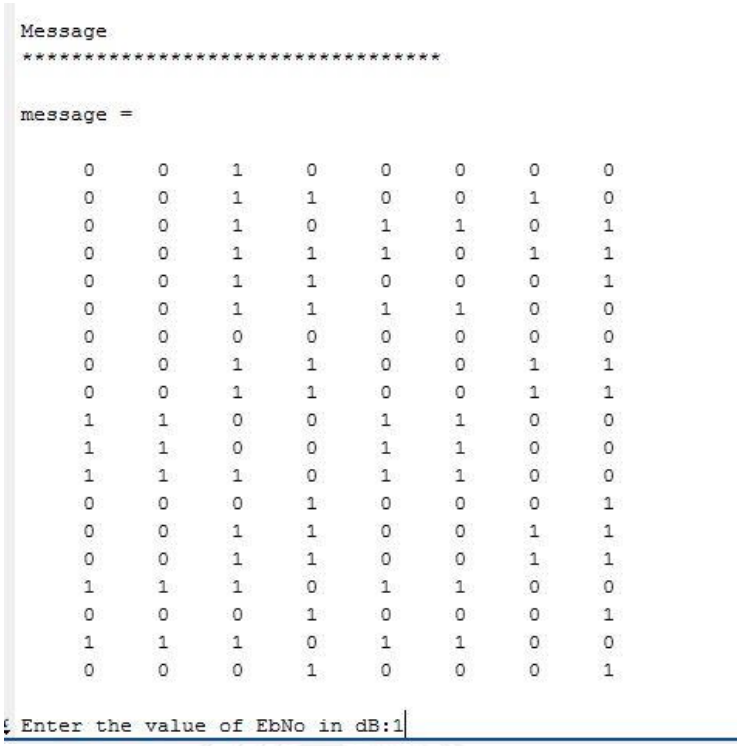
ภาพที่ 46 แสดงค่า *Codeword* หลังจากการเข้ารหัสโดย *RS*

ค. Codewords ที่ได้ในข้อ ข. จะถูกส่งสัญญาณรบกวนตามค่า E_bN_o ในตารางที่ 9

โดยวิธีการ AWNG (Additive White Gaussian Noise) ดังตัวอย่างในภาพที่ 47 โปรแกรมจะรอรับข้อมูลระดับสัญญาณที่จะทำการรบกวน ดังภาพที่ 48

```
EbNo = input('Enter the value of EbNo in dB:'); %EbNo = input('Enter the value of EbNo in dB:');
ldpctime = tic;
for i = 1 : nr
    code_ldpc = step(hEnc, message(i, :)');                          %Encode Data Source With Encoder
    coded_word_ldpc(i, :) = code_ldpc';
    %%                                                          Construct a BPSK Modulation Object.
    modObj = comm.BPSKModulator();
    hAWGN = comm.AWGNChannel('NoiseMethod', 'Signal to noise ratio (SNR)', 'SNR', ...
        25, 'SignalPower', 3);% add noise to the channel
    demodObj = comm.BPSKDemodulator('DecisionMethod','Approximate log-likelihood ratio', ...
                'Variance', 1/10^(hAWGN.SNR/10));
    %% Modulated Signal
    bpskSignal_ldpc = step(modObj, code_ldpc);
    %bpskSignal_rs = step(modObj, code_rs);
    %%
    hAWGN.SNR = EbNo;
    sigma = sqrt(10^(-EbNo/10));                                 %Noise Variance.
    receivedSignal_ldpc = step(hAWGN, bpskSignal_ldpc);             %Demodulate Signal
    demodulateSignal_ldpc = step(demodObj, receivedSignal_ldpc);    %Decode received signal
```

ภาพที่ 47 แสดงการจำลองใส่สัญญาณรบกวนลงใน *Codewords*



ภาพที่ 48 แสดงการใส่ค่าสัญญาณรบกวนเพื่อใช้ในการทดสอบ

ง. Codewords ที่ถูกรบกวนโดยสัญญาณที่ทำการใส่ลงไปตามข้อ ค. นั้น จะถูกทำการแก้ไขให้ถูกต้องตามข้อมูลต้นทางที่ใส่ โดยรหัสแก้ไขข้อผิดพลาดแบบ RS และ LDPC ดังภาพที่ 49 และ ภาพที่ 50

```
LDPC Error Corrected Message

ans =

     0     1     1     0     0     0     1     0
     0     0     1     1     0     0     1     1
     0     1     1     0     1     0     0     1
     0     0     1     0     1     0     1     0
     0     0     1     1     0     0     1     0
     0     0     1     1     1     0     0     0
     0     0     0     0     0     1     0     1
     0     0     1     1     0     0     1     1
     0     0     1     1     0     0     1     1
     1     1     0     0     1     0     0     0
     1     0     0     0     1     1     0     0
     1     1     1     0     1     1     0     0
     1     0     0     1     0     0     0     1
     0     0     1     1     1     0     1     1
     0     0     1     1     0     1     0     1
     1     1     1     0     0     1     0     0
     0     1     0     1     0     0     0     1
     1     1     1     0     1     1     0     0
     0     0     0     1     0     0     0     1
```

ภาพที่ 49 แสดงค่า *Codewords* หลังจากทำการถูกแก้ไข้ข้อผิดพลาดโดย *LDPC*

```
RS Error Corrected Message

ans =

     1     0     0     1     0     0     1     1
     1     1     1     0     1     1     0     0
     1     1     0     1     1     1     1     0
     0     0     0     0     0     1     0     0
     1     1     0     0     1     1     1     0
     1     0     1     0     0     0     1     1
     0     1     1     1     0     1     1     1
     1     1     1     0     1     1     0     0
     0     1     0     0     1     1     0     0
     1     1     1     0     0     0     0     1
     0     1     0     1     1     1     1     1
     0     0     0     1     0     0     1     1
     1     1     1     0     1     1     1     0
     0     1     0     0     1     1     0     0
     0     1     0     0     1     1     0     0
     0     0     0     1     0     0     1     1
     1     1     0     0     1     1     1     0
     0     0     0     1     1     1     0     1
     1     1     1     0     1     1     1     0
```

ภาพที่ 50 แสดงค่า *Codewords* หลังจากทำการถูกแก้ไข้ข้อผิดพลาดโดย *RS*

จ. หลังจากเสร็จสิ้นการแก้ไขข้อผิดพลาด โปรแกรมจะแสดงค่า Bit Error Rate และเวลาที่ใช้ในการเข้ารหัสและถอดรหัสพร้อมด้วยการแก้ไขข้อผิดพลาด ดังภาพที่ 51

```
LDPC Time Encode/Decode = 0.230611
RS Time Encode/Decode = 0.210694
Bit Error Rate for LDPC Code for EbNo (dB) = 30.000000 is 0.000000
Bit Error Rate for RS Code for EbNo (dB) = 30.000000 is 0.940789
>>
```

ภาพที่ 51 ตัวอย่างการแสดงผลค่า *BER* และ เวลาที่ใช้ในการเข้ารหัสและถอดรหัส

ฉ. ทำซ้ำข้อ ก – จ จนครบจำนวนค่าตัวแปร E_bN_o ตามตารางที่ 8

3.2 ขั้นตอนการเปรียบเทียบประสิทธิภาพการแก้ไขข้อผิดพลาดโดย RS และ LDPC โดยกำหนดสมการพหุนามตามหลักเกณฑ์ของ QR Code เวอร์ชัน 1 Error Correction M

ขั้นตอนนี้จะเป็นการทดลองเปรียบเทียบประสิทธิภาพของรหัส RS ที่มีวิธีการเข้ารหัสโดยสมการพนุนามที่เป็นไปตามกำหนดของ QR Code ในเวอร์ชันที่ 1 ระดับ Error Correction ที่ M และ LDPC

3.2.1 ขั้นตอนการเตรียมข้อมูลในการสร้างโปรแกรมจำลอง สำหรับ RS

ขั้นตอนนี้จำเป็นจะต้องทราบว่าการเข้ารหัสแบบ RS ใน QR Code นั้นมีวิธีในการเข้ารหัสอย่างไร

ก. การหา Bitstream ตามหลักของ QR Code เวอร์ชัน 1 ระดับ Error Correction ที่ M

การหา Bitstream เป็นไปตามขั้นตอนที่ 2.1.6 ในการทดลองนี้จะใช้คำว่า “SUMAMA” ซึ่งแปลงเป็น Bitstream ตามหลักเกฎฑ์ ของ QR Code เวอร์ชัน 1 ระดับ Error Correction ที่ M ได้ดังนี้

“00100000 00110101 00001010 11111010 00111110 10000000 11101100 00010001 11101100 00010001 11101100 00010001 11101100 00010001 11101100 00010001”

ข. ในการเข้ารหัสแบบ RS จะต้องแปลงเลขฐานสองให้เป็นเลขฐานสิบ ได้ดังนี้

“32 53 10 250 62 128 236 17 236 17 236 17 236 17 236 17”

และมีจำนวน Error Correction Codewords เท่ากับ 10 ตามกฎของ QR Code เวอร์ชัน 1 ระดับ Error correction M ตามตารางที่ 8 และจะสามารถหา Generator Polynomial ได้ตามตารางที่ 10 นี้

ค. หาสมการพหุนามที่ใช้ในการเข้ารหัสโดย RS เวอร์ชันที่ 1 ระดับ Error Correction ที่ M

ตารางที่ 10 แสดงสมการพหุนามที่ใช้ในการสร้างรหัสแก้ไขข้อผิดพลาด*RS* ตามจำนวน *Error Correction Codewords*

Number of error correction codewords	Generator polynomials
2	$x^2 + \alpha^{25}x + \alpha$
5	$x^5 + \alpha^{113}x^4 + \alpha^{164}x^3 + \alpha^{166}x^2 + \alpha^{119}x + \alpha^{10}$
6	$x^6 + \alpha^{116}x^5 + x^4 + \alpha^{134}x^3 + \alpha^{5}x^2 + \alpha^{176}x + \alpha^{15}$
7	$x^7 + \alpha^{87}x^6 + \alpha^{229}x^5 + \alpha^{146}x^4 + \alpha^{149}x^3 + \alpha^{238}x^2 + \alpha^{102}x + \alpha^{21}$
8	$x^8 + \alpha^{175}x^7 + \alpha^{238}x^6 + \alpha^{208}x^5 + \alpha^{249}x^4 + \alpha^{215}x^3 + \alpha^{252}x^2 + \alpha^{196}x + \alpha^{28}$
10	$x^{10} + \alpha^{251}x^9 + \alpha^{67}x^8 + \alpha^{46}x^7 + \alpha^{61}x^6 + \alpha^{118}x^5 + \alpha^{70}x^4 + \alpha^{64}x^3 + \alpha^{94}x^2 + \alpha^{32}x + \alpha^{45}$

ดังนั้นตัวอย่างข้อมูลในการทดลองจะต้องใช้ Generator polynomial ดังนี้

$$g(x) = x^{10} + \alpha^{251}x^9 + \alpha^{67}x^8 + \alpha^{46}x^7 + \alpha^{61}x^6 + \alpha^{118}x^5 + \alpha^{70}x^4 + \alpha^{64}x^3 + \alpha^{94}x^2 + \alpha^{32}x + \alpha^{45} \quad (3.1)$$

3.2.2 ขั้นตอนการเตรียมข้อมูลในการสร้างโปรแกรมจำลอง สำหรับ LDPC

ก. การหา Bitstream ตามหลักของ QR Code เวอร์ชัน 1 ระดับ Error Correction ที่ M

ขั้นตอนนี้จะใช้ข้อมูลเดียวกับ 3.2.1 ก โดยใช้คำว่า “SUMAMA” ซึ่งแปลงเป็น Bitstream ตามหลักเกฎฑ์ ของ QR Code เวอร์ชัน 1 ระดับ Error Correction ที่ M ได้ดังนี้

“00100000 00110101 00001010 11111010 00111110 10000000 11101100 00010001 11101100 00010001 11101100 00010001 11101100 00010001 11101100 00010001”

ข. การหา Low Density Parity Check matrix หรือ Matrix H นั้นจะใช้ Function การทำงานของ MATLAB เพื่อสร้าง Matrix H แล้วแปลง Matrix H เป็น Matrix G ตามหลักการสร้าง Low Density Parity Check ในข้อที่ 2.3

3.2.3 สร้างโปรแกรมจำลองในการเข้ารหัสข้อมูล แบบ RS โดยใช้สมการพหุนามตามกำหนดของ QR Cod และ LDPC

ขั้นตอนนี้จะเป็นการสร้างโปรแกรมจำลองเพื่อใช้ในการทดสอบประสิทธิภาพของรหัสการแก้ไขข้อผิดพลาดทั้ง RS และ LDPC โดยใช้ Function ของ MATLAB

```
    0 0 0 1 0 0 0 1;
    1 1 1 0 1 1 0 0;
    0 0 0 1 0 0 0 1;
    1 1 1 0 1 1 0 0;
    0 0 0 1 0 0 0 1;
    1 1 1 0 1 1 0 0;
    0 0 0 1 0 0 0 1;
    1 1 1 0 1 1 0 0;
    0 0 0 1 0 0 0 1]'
    %1 1 1 0 1 1 0 0;
    %0 0 0 1 0 0 0 1;
    %1 1 1 0 1 1 0 0];
t = tic;
msg = bi2de(message', 'left-msb');
N = 26;                          %number of code words
K = 16;                          %number of message words
message = reshape(message, 1, K * b)
%% Given from the table.
a0 = 1; a251 = 216; a67 = 194; a46 = 159; a61 = 111; a118 = 199; a70 = 94;
a64 = 95; a94 = 113; a32 = 157; a45 = 193;
%% Taken polynomial of order 10
gen = de2bi([1 a251 a67 a45 a61 a118 a70 a64 a94 a32 a45], 8, 'left-msb')';

gp = reshape(gen, 1, (N - K + 1) * b);          %polynomial 10
f = [message zeros(1, (N - K) * b)];
%% Finding Parity Check Bits
[q, r] = gfdeconv(fliplr(f), fliplr(gp));
EC_Codeword = r                                          %parity bits
```

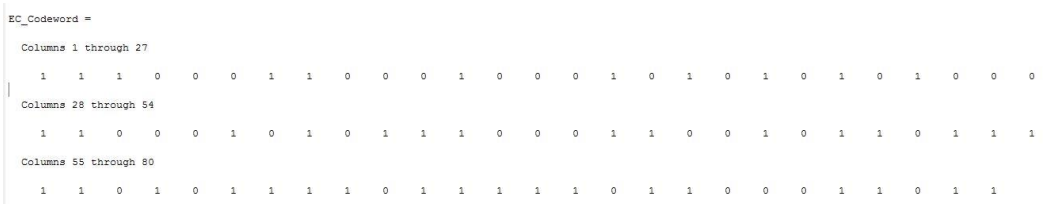
ภาพที่ 52 *แสดงตัวอย่างการเข้ารหัส RS โดยสมการพหุนามกำหนดโดย QR Code*


```
    0 0 0 1 0 0 0 1;
    1 1 1 0 1 1 0 0;
    0 0 0 1 0 0 0 1];
%%
t = tic;
[nr, nc] = size(message);                          %getting size of a message
msg = reshape(message, 1, nr * nc)
L = 208;                                           %total length of a code
n = L/nc;
%code_rate = nr/n;
K = nc                                             %total number of message bits
N = (nc * ceil(n/nr))                              %coded block size
pol = cyclpoly(N, K)                               %Creating Parity Check Matrix
[H, genmat,k] = cyclgen(N, pol)
%% LDPC Encoder and Decoder
paritymat = H;                                     %parity matrix
hEnc = comm.LDPCEncoder(sparse(paritymat))         %Construct a default LDPC encoder object
hDec = comm.LDPCDecoder(sparse(paritymat))         %Construct a companion LDPC decoder object
%%

%%                                                 Construct a BPSK Modulation Object.
modObj = comm.BPSKModulator();
```

ภาพที่ 53 *แสดงตัวอย่างการเข้ารหัส LDPC ตามกำหนดจำนวน Error Correction Codewords ของ QR Code*

เมื่อโปรแกรมทำการเข้ารหัสตัวอย่างข้อมูลคำว่า “SUMAMA” โดยใช้สมการพหุนามตามกำหนดของ QR Code จะได้ จำนวน Error Correction codewords ตามเกณฑ์ของ QR Code ดังนี้



ภาพที่ 54 *แสดงตัวอย่าง Error Correction Codeword ที่ได้จากการเข้ารหัสโดย RS*

3.2.4 ทำการทดสอบประสิทธิภาพ RS และ LDPC โดย RS ใช้พหุนามตามกำหนดของ QR Code เวอร์ชัน 1 ระดับ Error Correction ที่ M

ทำการรันโปรแกรมของการเข้ารหัสแบบ RS และ LDPC โดยเปลี่ยนค่า E_bN_o ตั้งแต่ 1 – 10 แล้วเปรียบเทียบ BER และ เวลาที่ใช้ในการเข้ารหัสและถอดรหัส

ตารางที่ 11 ตารางแสดงค่า E_bN_o ที่ใช้เป็นสัญญาณรบกวนนในการทดลอง *3.2*

QR Version	QR Error Correction level	Code Rate	E_bN_o
1	M	(16/26)	1
			2
			3
			4
			5
			6
			7
			8
			10

บทที่ 4

การทดลองและผลการทดลอง

ในบทนี้จะกล่าวถึง ผลการทดลองของการเปรียบเทียบประสิทธิภาพของการแก้ไขข้อผิดพลาดแบบ RS และ LDPC โดยผลการทดลองจะแบ่งเป็น 2 ส่วน ดังนี้

4.1 ผลการเปรียบเทียบประสิทธิภาพการแก้ไขข้อผิดพลาดโดย RS และ LDPC โดยสุ่มพหุนาม

ตารางที่ 12 *แสดง Bit Error Rate ในแต่ละการใส่ค่าสัญญาณรบกวนในการทดสอบรหัส RS และ LDPC โดยสุ่มพหุนาม*

Code Rate	EbNo	BER LDPC	BER RS
	1	0.112	0.822
	2	0.145	0.829
	3	0.086	0.842
	4	0.092	0.836
	5	0.086	0.875
	6	0.053	0.914
	7	0.013	0.914
	8	0.026	0.941
	9	0.007	0.941
(19/26)	10	0.007	0.941
	1	0.199	0.779
	2	0.199	0.853
	3	0.088	0.853
	4	0.118	0.838
	5	0.037	0.919
	6	0.051	0.904
	7	0.015	0.934
(16/26)	8	0.022	0.912

	9	0.007	0.949
	10	0.000	0.956
	1	0.164	0.757
	2	0.151	0.776
	3	0.092	0.816
	4	0.092	0.803
	5	0.059	0.822
	6	0.072	0.763
	7	0.033	0.783
	8	0.026	0.789
	9	0.000	0.849
(13/26)	10	0.000	0.789
	1	0.194	0.750
	2	0.208	0.778
	3	0.125	0.736
	4	0.028	0.681
	5	0.042	0.736
	6	0.056	0.639
	7	0.014	0.722
	8	0.000	0.653
	9	0.000	0.583
(9/26)	10	0.000	0.583

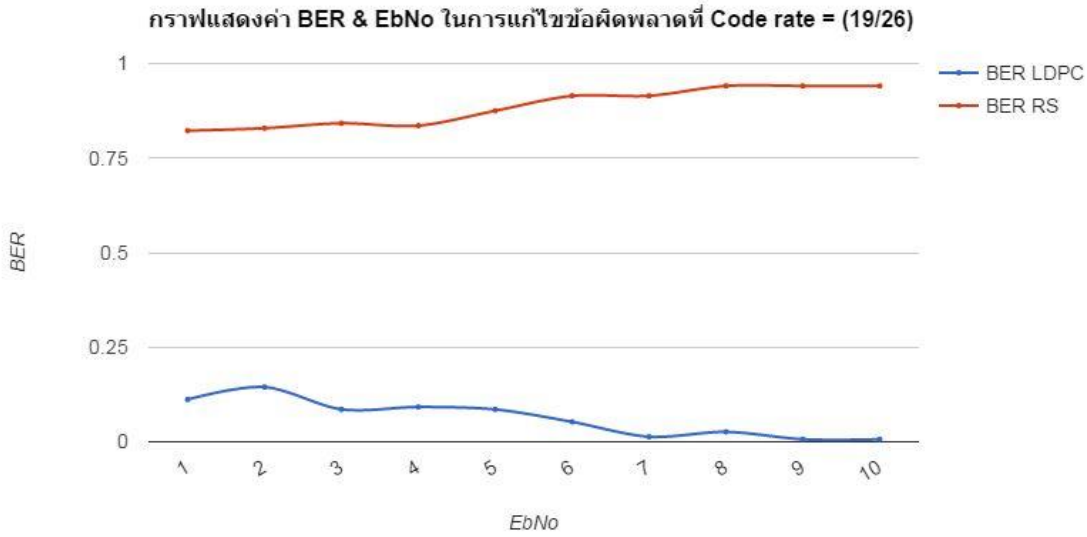
ตารางที่ 13 *แสดง เวลาที่ใช้ในการทำการแก้ไขข้อผิดพลาด แต่ละการใส่ค่าสัญญาณรบกวนในการทดสอบรหัส RS และ LDPC โดยสุ่มพหุนาม*

Code Rate	EbNo	Time LDPC	Time RS
(19/26)	1	0.221	0.209
	2	0.240	0.230
	3	0.247	0.212
	4	0.235	0.212

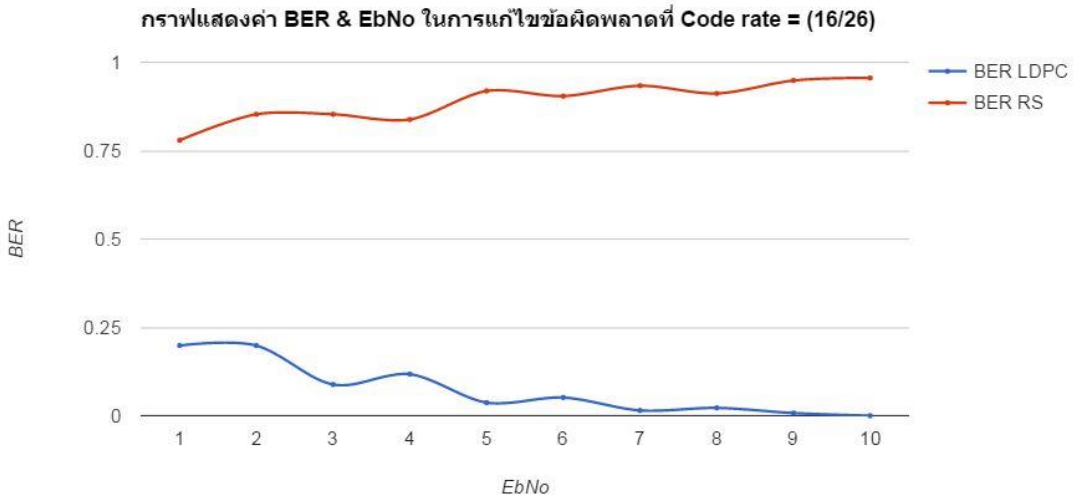
	5	0.234	0.216
	6	0.219	0.213
	7	0.220	0.215
	8	0.230	0.202
	9	0.271	0.246
	10	0.222	0.214
(16/26)	1	0.231	0.221
	2	0.213	0.220
	3	0.217	0.210
	4	0.210	0.206
	5	0.209	0.208
	6	0.204	0.211
	7	0.215	0.208
	8	0.213	0.206
	9	0.212	0.208
	10	0.210	0.198
(13/26)	1	0.192	0.189
	2	0.185	0.188
	3	0.185	0.181
	4	0.185	0.180
	5	0.185	0.187
	6	0.181	0.182
	7	0.185	0.180
	8	0.185	0.178
	9	0.189	0.185
	10	0.191	0.180
(9/26)	1	0.149	0.174
	2	0.130	0.128
	3	0.118	0.140
	4	0.128	0.128
	5	0.127	0.136
	6	0.127	0.132
	7	0.126	0.133

	8	0.122	0.133
	9	0.127	0.128
	10	0.125	0.130

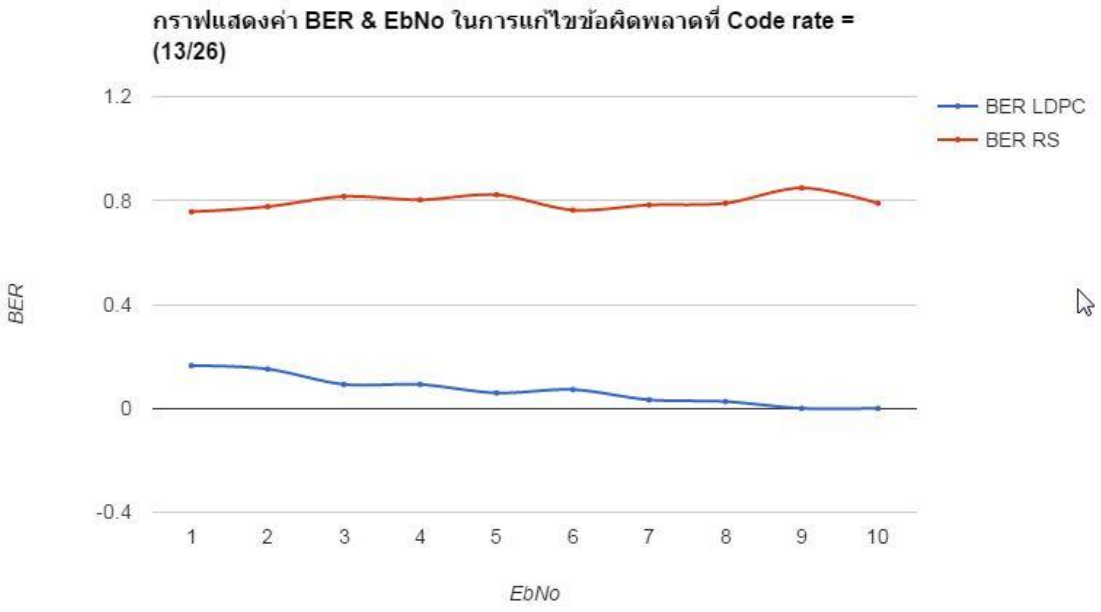
ส่วนนี้จะเป็นการเปรียบเทียบโดยการสร้างกราฟ เพื่อดูการเปลี่ยนแปลงของ BER ในแต่ละสัญญาณรบกวนที่เพิ่มมากขึ้น ของการแก้ไขข้อผิดพลาดแบบ RS และ LDPC ที่มีค่าพหุนามไม่คงที่



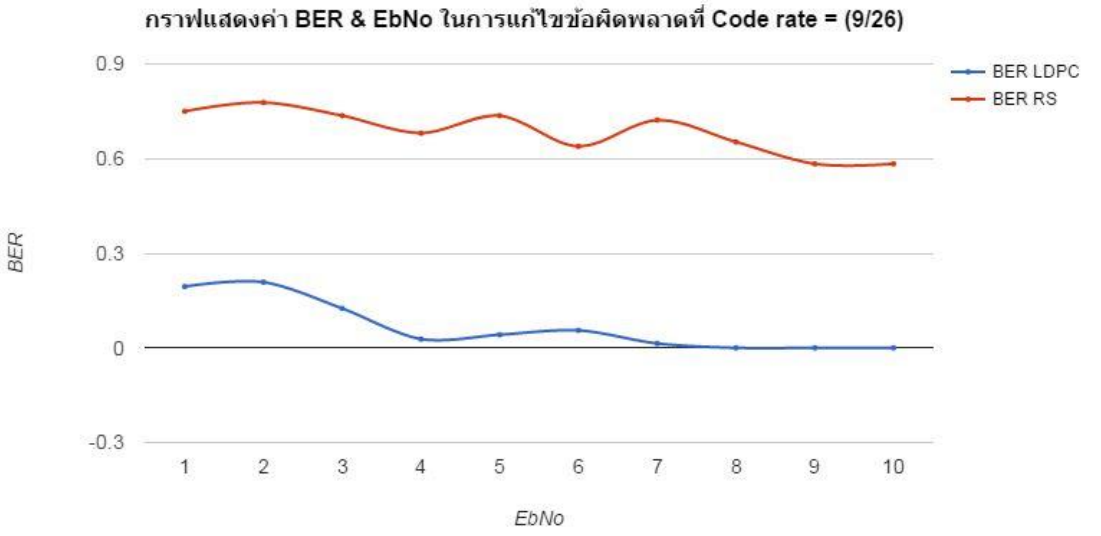
ภาพที่ 55 *กราฟแสดงค่า BER & EbNo ในการแก้ไขข้อผิดพลาดที่ Code rate = (19/26)*



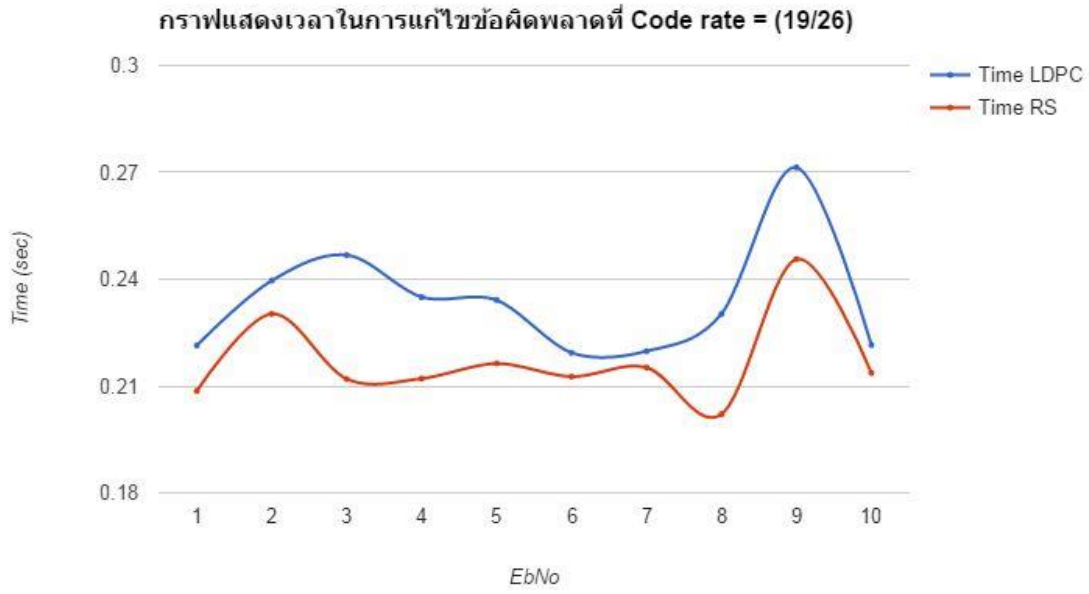
ภาพที่ 56 *กราฟแสดงค่า BER & EbNo ในการแก้ไขข้อผิดพลาดที่ Code rate = (16/26)*



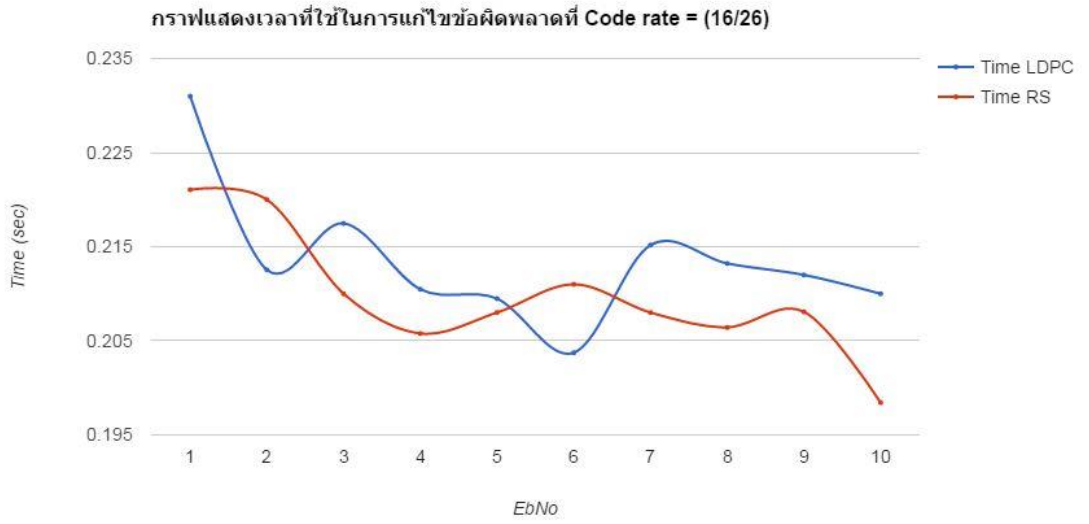
ภาพที่ 57 *กราฟแสดงค่า BER & EbNo ในการแก้ไขข้อผิดพลาดที่ Code rate = (13/26)*



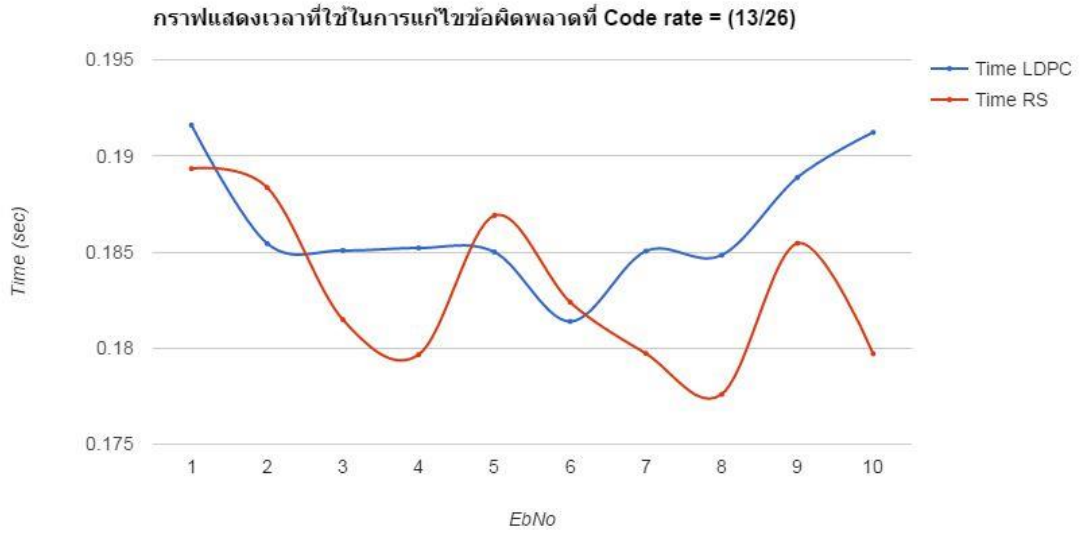
ภาพที่ 58 *กราฟแสดงค่า BER & EbNo ในการแก้ไขข้อผิดพลาดที่ Code rate = (9/26)*



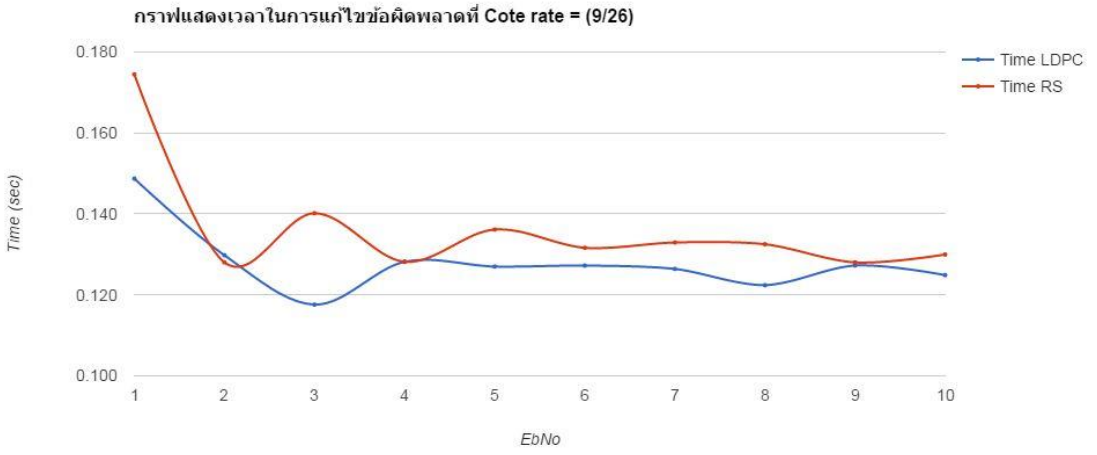
ภาพที่ 59 *กราฟแสดงเวลาในการแก้ไขข้อผิดพลาดโดย RS และ LDPC มีค่าสมการพหุนามที่คงที่ Code rate = (19/26)*



ภาพที่ 60 *กราฟแสดงเวลาในการแก้ไขข้อผิดพลาดโดย RS และ LDPC มีค่าสมการพหุนามที่คงที่ Code rate = (16/26)*



ภาพที่ 61 *กราฟแสดงเวลาในการแก้ไขข้อผิดพลาดโดย RS และ LDPC มีค่าสมการพหุนามที่คงที่ Code rate = (13/26)*



ภาพที่ 62 *กราฟแสดงเวลาในการแก้ไขข้อผิดพลาดโดย RS และ LDPC มีค่าสมการพหุนามที่คงที่ Code rate = (9/26)*

4.2 ผลการเปรียบเทียบประสิทธิภาพการแก้ไขข้อผิดพลาด RS และ LDPC โดยกำหนดสมการพหุนามตามหลักเกณฑ์ของ QR Code

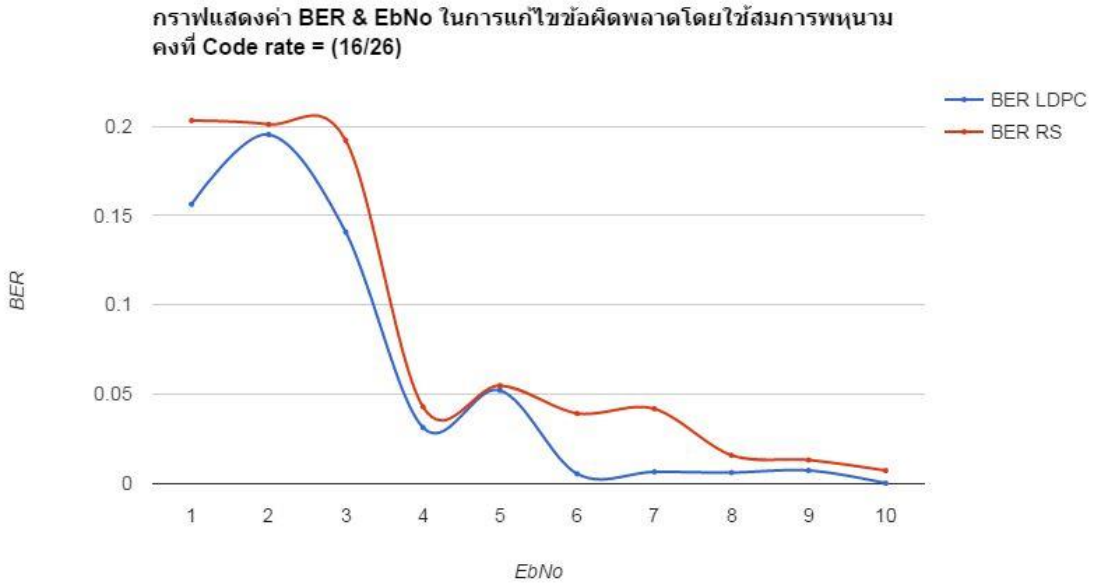
ตารางที่ 14 *แสดง Bit Error Rate ในแต่ละการใส่ค่าสัญญาณรบกวนในการทดสอบรหัส RS และ LDPC ที่มีค่าสมการพนุนามตามหลักเกณฑ์ของ QR Code เวอร์ชัน 1 Error Correction M*

Code Rate	EbNo	BER LDPC	BER RS
(16/26)	1	0.156	0.203
	2	0.195	0.125
	3	0.141	0.094
	4	0.031	0.043
	5	0.055	0.055
	6	0.005	0.039
	7	0.006	0.042
	8	0.006	0.016
	9	0.007	0.008
	10	0.000	0.000

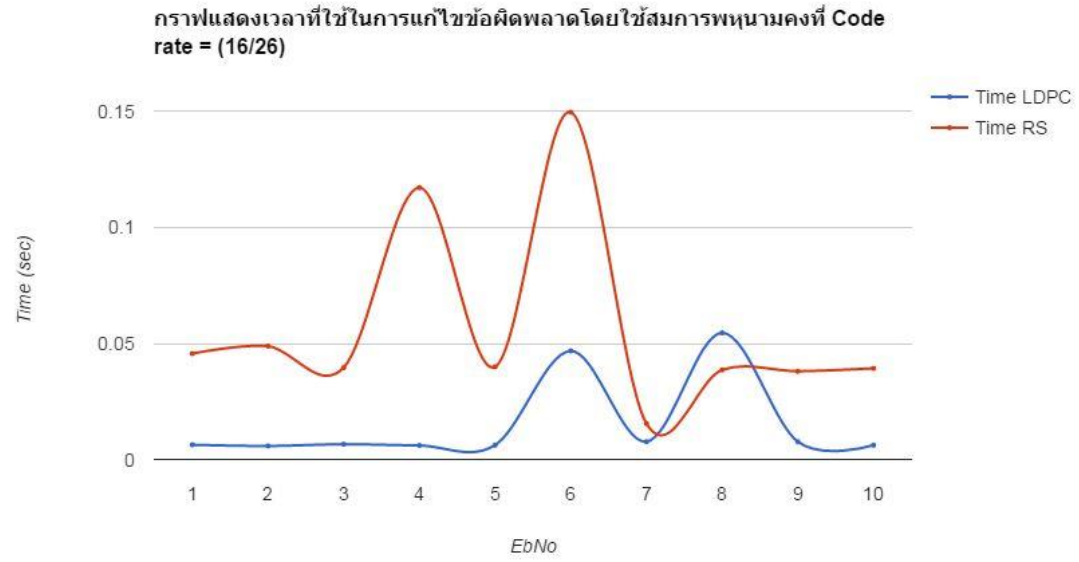
ตารางที่ 15 *แสดง เวลาที่ใช้ในการทำการแก้ไขข้อผิดพลาด แต่ละการใส่ค่าสัญญาณรบกวนในการทดสอบรหัส RS และ LDPC ที่มีค่าสมการพหุนามตามหลักเกณฑ์ของ QR Code เวอร์ชัน 1 Error Correction M*

Code Rate	EbNo	Time LDPC	Time RS
(16/26)	1	0.006	0.046
	2	0.006	0.049
	3	0.007	0.040
	4	0.006	0.117
	5	0.006	0.040
	6	0.047	0.150
	7	0.008	0.016
	8	0.055	0.039
	9	0.008	0.038
	10	0.006	0.039

ส่วนนี้จะเป็นการเปรียบเทียบโดยการสร้างกราฟ เพื่อดูการเปลี่ยนแปลงของ BER ในแต่ละสัญญาณรบกวนที่เพิ่มมากขึ้น ของการแก้ไขข้อผิดพลาดแบบ RS และ LDPC ที่มีสมการพหุนามคงที่ตามกฎเกณฑ์ของ QR Code เวอร์ชัน 1 ระดับ Error Correction M

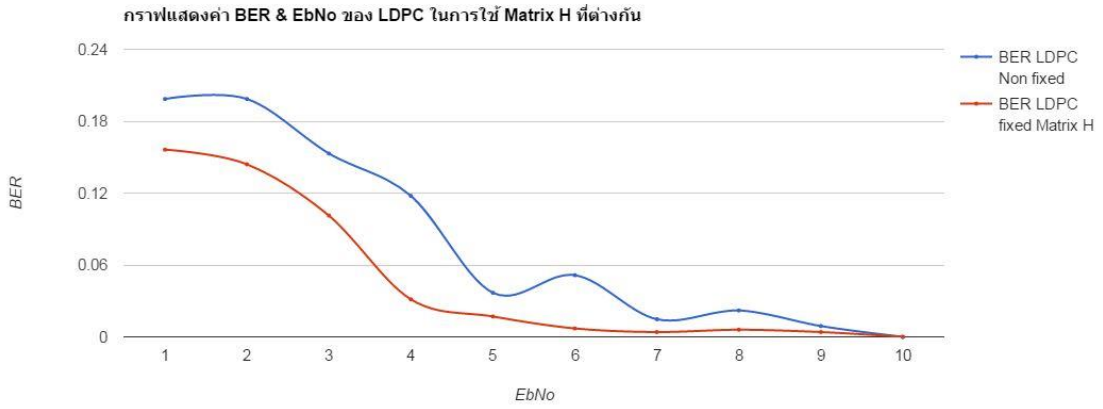


ภาพที่ 63 *กราฟแสดงค่า BER & EbNo ในการแก้ไขข้อผิดพลาดโดยใช้สมการพหุนามคงที่ Code rate = (16/26)*



ภาพที่ 64 *กราฟแสดงเวลาที่ใช้ในการแก้ไขข้อผิดพลาดโดยใช้สมการพหุนามคงที่ Code rate = (16/26)*

เมื่อนำค่า BER ของ LDPC ที่สุ่มพหุนามเพื่อสร้าง Parity Check Matrix H และ BER ของ LDPC ที่ระบุพหุนามเพื่อสร้าง Parity Check Matrix H ที่คงที่ กราฟผลการทดลอง ดังภาพที่ 63



ภาพที่ 65 *กราฟแสดงค่า BER ฿ EbNo ของ LDPC ในการใช้ Matrix H เพื่อเข้ารหัสที่ต่างกัน*

จากการทดลองพบว่าเมื่อค่าสัญญาณรบกวนที่มีค่าตั้งแต่ EbNo เท่ากับ 10 ขึ้นไปจะทำให้ค่า BER = 0 สำหรับการแก้ไขข้อผิดพลาดแบบ LDPC ดังภาพต่อไปนี้

```
Message
**********************************

msg =

  Columns 1 through 27

    0    0    0    0    0    1    1    0    1    0    1    0    1    0    1    0    0    0    0    1    1    0    1    0    1    0    1

  Columns 28 through 54

    0    1    0    1    0    0    1    0    0    1    0    1    0    1    0    1    0    1    0    1    0    1    0    0    1    0    0

  Columns 55 through 81

    0    1    0    1    0    1    0    1    0    1    0    0    1    0    0    0    1    0    1    0    1    0    1    0    1    0    0

  Columns 82 through 108

    0    1    1    0    0    1    0    1    0    1    0    1    0    1    0    0    0    0    1    0    0    0    0    0    0    0    0

  Columns 109 through 128

    0    0    0    0    0    0    0    0    1    0    0    1    0    1    0    1    0    1    0    1
```

ภาพที่ 66 *Bitstream ของข้อความก่อนเข้ารหัส LDPC*

```
LDPC Error Corrected Message

ans =

  Columns 1 through 27

     0     0     0     1     0     0     0     0     0     0     1     0     0     0     0     0     0     0     0     0     1     1     0     0     0     1     0

  Columns 28 through 54

     1     0     1     1     0     0     1     1     0     0     0     0     1     1     0     0     0     0     0     0     0     1     1     1     0     1     1

  Columns 55 through 81

     0     0     0     0     0     1     0     0     0     1     1     1     1     0     1     1     0     0     0     0     0     1     0     0     0     1     1

  Columns 82 through 108

     1     1     0     1     1     0     0     0     0     0     1     0     0     0     1     1     1     1     0     1     1     0     0     0     0     0     1

  Columns 109 through 128

     0     0     0     1     1     1     1     0     1     1     0     0     0     0     0     1     0     0     0     1

Time Decode = 0.007583
Bit Error Rate for LDPC Code for EbNo (dB) = 30.000000 is 0.000000
```

ภาพที่ 67 *Bitstream ของข้อความหลังจากทำการแก้ไขข้อผิดพลาดที่เกิดจากการส่งสัญญาณ EbNo = 30 โดยรหัส LDPC*

จะเห็นว่า การแก้ไขข้อผิดพลาดของ LDPC มีประสิทธิภาพแก้ไขได้อย่างถูกต้องในสัญญาณการรบกวนที่เพิ่มมากขึ้น

บทที่ 5

สรุปผลการวิจัย

ในบทนี้จะกล่าวถึงผลสรุปงานวิจัย ปัญหาที่พบ และขอเสนอแนะอันจะเป็นแนวทางในการพัฒนาวิธีการแก้ไข Error Correction ใน QR Code ให้มีประสิทธิภาพในแง่ของการแก้ไขข้อผิดพลาดที่มากขึ้นได้

5.1 บทสรุป

เพื่อศึกษาการแก้ไขข้อผิดพลาดบน QR Code ที่ใช้รหัสการแก้ไขแบบ LDPC โดยปรับ การเข้ารหัส LDPC ให้ได้ ออกแบบและนำเสนอวิธีการที่เหมาะสมในการปรับปรุงภาพ QR Code ที่มีความเสียหายจากรอยขีดข่วน โดยการใช้วิธีการประมวลผลภาพเพื่อให้ได้ภาพ QR Code ที่สามารถทำการถอดรหัสได้ ซึ่งบทสรุปจะแบ่งออกเป็น 2 ส่วน ดังนี้

5.1.1 ผลการเปรียบเทียบประสิทธิภาพการแก้ไขข้อผิดพลาดโดย RS และ LDPC โดยสุ่มพหุนาม

จากผลการทดลองใน 4.2 ซึ่งข้อมูลที่ใช้ในการทดลองคือ “SUMAMA” ที่แปลงเป็น Bitstream แล้วนำไปทำการเข้ารหัสเพื่อหา Error Correction Codewords โดยวิธี RS และ LDPC ซึ่งการคำนวณการเข้ารหัสของทั้งสองได้มีการสุ่มพหุนามที่ใช้ในการเข้ารหัสเพื่อหา Error Correction Codewords ในการสุ่มพหุนามนี้ทำให้ได้จำนวน Error Correction Codewords ที่ไม่คงที่ จากการทดลองการเข้ารหัสและใส่สัญญาณรบกวน E_bN_o = 1-10 นั้นจะเห็นได้ว่า BER หรือ Bit Error Rate ของรหัส RS นั้นสูงกว่า BER ของ LDPC ในทุกๆ การใส่ค่าสัญญาณรบกวนที่เท่ากัน และเมื่อดูผลการทดลองของรหัส RS เมื่อมีการใส่ค่าสัญญาณรบกวนที่มากขึ้นนั้น BER ค่อนข้างคงที่ถึงแม้ค่าสัญญาณรบกวนจะมากขึ้น แต่เมื่อดูผลการทดลองของรหัส LDPC จะเห็นว่า เมื่อมีการใส่ค่าสัญญาณรบกวนที่มากขึ้นนั้น BER จะมีค่าน้อยลงจนถึง 0 นั่นหมายความว่า LDPC มีประสิทธิภาพในการแก้ไขข้อผิดพลาดต่อสัญญาณการรบกวนที่มากขึ้น จำนวนบิตที่แก้ไขไม่ได้จึงไม่มีเลย หรือเท่ากับ 0 นั่นเอง แต่เมื่อเปรียบเทียบระยะเวลาการแก้ไขข้อผิดพลาดนั้นจะเห็นได้ว่า รหัส RS จะใช้เวลาในการแก้ไขข้อผิดพลาดน้อยกว่ารหัส LDPC

จากผลการทดลองใน 4.3 ซึ่งข้อมูลที่ใช้ในการทดลองคือ “SUMAMA” ที่แปลงเป็น Bitstream แล้วนำไปทำการเข้ารหัสเพื่อหา Error Correction Codewords โดยวิธี RS และ LDPC ซึ่งการคำนวณการเข้ารหัสของทั้งสองได้มีการกำหนดสมการพหุนามที่ใช้ในการเข้ารหัสเพื่อหา Error Correction Codewords ของ RS และกำหนดขนาดในการสร้าง Parity Check Matrix H ให้มีขนาดที่สร้างจำนวน Error Correction Codewords เท่ากับ 10 ซึ่งตรงตามข้อกำหนดของ QR Code เวอร์ชัน 1 Error Correction M จากการทดลองการเข้ารหัสและใส่สัญญาณรบกวน E_bN_o = 1-10 นั้นจะเห็นได้ว่า BER ของรหัส RS นั้นสูงกว่า BER ของ LDPC ในทุกๆ การใส่ค่าสัญญาณรบกวนที่เท่ากัน และเมื่อดูผลการทดลองของรหัส RS เมื่อมีการใส่ค่าสัญญาณรบกวนที่มากขึ้นนั้น BER ค่อนข้างคงที่

ถึงแม้ค่าสัญญาณรบกวนจะมากขึ้น แต่เมื่อดูผลการทดลองของรหัส LDPC จะเห็นว่า เมื่อมีการใส่ค่าสัญญาณรบกวนที่มากขึ้นนั้น BER จะมีค่าน้อยลงจนถึง 0 นั่นหมายความว่า LDPC มีประสิทธิภาพในการแก้ไขข้อผิดพลาดต่อสัญญาณการรบกวนที่มากขึ้น จำนวนบิตที่แก้ไขไม่ได้จึงไม่มีเลย หรือเท่ากับ 0 นั่นเอง แต่เมื่อเปรียบเทียบระยะเวลาการแก้ไขข้อผิดพลาดนั้นจะเห็นได้ว่า รหัส RS จะใช้เวลาในการแก้ไขข้อผิดพลาดน้อยกว่ารหัส LDPC

5.1.2 บทสรุปการประยุกต์ใช้ LDPC ใน QR Code

จากผลการทดลองจะเห็นได้ว่า เมื่อมีการปรับ Parity Check Matrix H ให้เข้ารหัสกับข้อมูลตามขนาดของ QR Code ในเวอร์ชัน 1 Error Correction M ในการทดลองที่ 3.2 เพื่อให้ได้จำนวน Error Correction Codewords ที่สามารถใส่เข้ากับ QR Code ตามกำหนดในเวอร์ชันนั้น ค่า BER ของ LDPC ที่มีการระบุพหุนามเพื่อสร้าง Parity Check Matrix H ที่คงที่นั้น มีค่าน้อยกว่า BER ของ LDPC ที่ไม่มีการระบุพหุนามตามภาพที่ 63 และเวลาที่ใช้ในการถอดรหัสน้อยกว่าเช่นกัน จากการศึกษาในงานวิจัยนี้ทำให้มั่นใจได้ว่า LDPC สามารถนำมาประยุกต์ใช้เพื่อแก้ไข Error Correction ตามกำหนดจำนวน Error Correction Codewords ใน QR Code เวอร์ชัน 1 Error Correction M ได้ โดยระบุพหุนามที่สร้าง Parity Check Matrix H และการแก้ไขข้อผิดพลาดของ LDPC นั้นทำงานได้อย่างถูกต้องตามภาพที่ 64 และ ภาพ 65 ที่แสดงค่า Bitstream ของข้อมูลก่อนถูกรบกวนและหลังจากทำการแก้ไขข้อผิดพลาดแล้ว มีค่าที่เท่ากัน

5.1.3 ข้อเสนอแนะ

เนื่องด้วยในงานวิจัยนี้มีการกำหนดเวอร์ชันของ QR Code ที่ใช้ในการทดลอง คือ QR Code เวอร์ชัน 1 ดังนั้นในการนำวิธีการที่นำเสนอไปใช้ใน QR Code เวอร์ชันอื่นๆ จำเป็นต้องมีการกำหนดการสร้างขนาด Parity Matrix H ที่สอดคล้องกับจำนวน Error Correction Codewords เวอร์ชันนั้นๆ LDPC ที่ถูกสร้างในงานวิจัยนี้ สามารถนำไปแทนที่ รหัส RS เพื่อสร้าง Bitstream และใส่เข้าไปในรูปแบบของ QR Codeได้เพราะมีขนาด Error Correction Codewords ที่ตรงตามกำหนดของ QR Code เวอร์ชัน

รายการอ้างอิง

1] *Generating your Binary String* [cited 2012 October 15]; Available from: http://www.thonky.com/qr-code-tutorial/part-1-encode-data.

[2] *About QR Code* 1994 [cited 2013 April 20]; Available from: http://www.denso-wave.com/qrcode/index-e.html.

[3] *QR Stuff QR Code Creator*. 2011; Available from: http://www.qrstuff.com/blog/2011/12/14/qr-code-error-correction.

[4] *QR Code Error Correction* 2011; Available from: http://www.qrstuff.com/blog/2011/12/14/qr-code-error-correction.

[5] Pross, W.Q., F. Otesteanu, M., *Using PEG-LDPC codes for object identification*, in *Electronics and Telecommunications (ISETC),* 2010 9*th International Symposium*. 2010, IEEE Trans. Inform. Theory. p. 361-364.

[6] Siam QR – สร้าง QR โค๊ด, *QR Code* คืออะไร. Available from: Available from: http://www.siamqr.com/?page_id=2.

[7] รศ.ดร.วิระสิทธิ์ อิ่มถวิล, *Design of High-Rate LDPC Codes with Low Encoding Complexity for Magnetic Recording Channel*. 2009, ศูนย์วิจัยร่วมเฉพาะทางด้านส่วนประกอบฮาร์ดดิสก์ไดร์ฟ.

[8] *QR code*. 2014; Available from: http://en.wikipedia.org/wiki/QR_code.

[9] Gallager, R.G., *Low-Density Parity-Check Codes*. 1962, IRE Trans. Inform. Theory. p. 21-28.

[10] K., V.S.a.H., *An Implementation of Process to Remove Scratches from QR Code with Mobile Phone*. Proceeding of the International Conference on Information Technology, Electronics and Communications (January 2013). p. 42-46.

[11] Li., J.C.L.W.Y., *Performance Comparison between Non-Binary LDPC Codes and Reed-Solomon Codes over Noise Bursts Channels*, in *Communications, Circuits and Systems,* 2005. *Proceedings.* 2005 *International Conference*. 2005. p. 1-4.

[12] Mackay, D.J.C., *Good codes based on very sparse matrices*, C.B. Lecture Notes Comput. Sci., Ed, Editor. 1995. p. 110-111.

[13] Makhapu, P., Sangthongpattana, K.,Jantarapatin, S. and Mitrpant, C., *The embedding of Thai in QR Code*, in *Telecommunications and Information Technology (ECTI-CON)* 2011. p. 516-519.

[14] ISO/IEC, *Information technology-Automatic identification and data capture techniques-Bar code*. 2005.

[15] ISO/IEC, *Information technology-Automatic identification and data capture techniques-Bar code symbology-QR Code*. 2006, First Edition. 18004. Switzerland: ISO copyright office.

ประวัติผู้เขียนวิทยานิพนธ์

นางสาวสุมามาลย์ สุขสาคร เกิดเมื่อวันที่ 5 มีนาคม 2528 จังหวัดกรุงเทพมหานคร สำเร็จการศึกษาหลักสูตรปริญญาตรีวิศวกรรมศาสตร์ สาขาวิชาวิศวกรรมคอมพิวเตอร์ จากมหาวิทยาลัยเทคโนโลยีสุรนารี ในปีการศึกษา 2552 จากนั้นได้ทำงานในตำแหน่งวิศวกรระบบแผนก IT ให้กับบริษัทผลิตฮาร์ดดิส ซีเกทเทคโนโลยี ไทยแลนด์ จนถึงปัจจุบัน เป็นระยะเวลา 8.5 ปี โดยมีหน้าที่หลักคือ ดูแลและออกแบบโครงสร้างระบบคอมพิวเตอร์เพื่อบริการข้อมูลในสายการผลิตฮาร์ดดิส และในปี 2555 ได้มีโอกาสเข้าศึกษาต่อในหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิทยาศาสตร์คอมพิวเตอร์ ที่ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในภาคการศึกษาต้น ปีการศึกษา 2555