การเข้ารหัสลับข้อมูลด้วยระบบอลวนโดยใช้ FPAA

DATA CRYPTOGRAPHY BASED ON CHAOTIC SYSTEMS USING FPAA

โดย

นายกันตพงศ์ บุญมาก

นายชินพัฒน์ อมรสู่สวัสดิ์

นายปิติพัฒน์ สารปรัง

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2565

การเข้ารหัสลับข้อมูลด้วยระบบอลวน โดยใช้ FPAA

DATA CRYPTOGRAPHY BASED ON CHAOTIC SYSTEMS USING FPAA

โดย

นายกันตพงศ์ บุญมาก 62010048

นายชินพัฒน์ อมรสู่สวัสดิ์ 62010200

นายปิติพัฒน์ สารปรัง 62010556

อาจารย์ที่ปรึกษา

รศ.ดร.พิพัฒน์ พรหมมี

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2565

ปริญญานิพนธ์ปีการศึกษา 2565

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การเข้ารหัสลับข้อมูลด้วยระบบอลวนโดยใช้ FPAA

DATA CRYPTOGRAPHY BASED ON CHAOTIC SYSTEMS USING FPAA

ผู้จัดทำ

1. นายกันตพงศ์ บุญมาก รหัสนักศึกษา 62010048
2. นายชินพัฒน์ อมรสู่สวัสดิ์ รหัสนักศึกษา 62010200
3. นายปิติพัฒน์ สารปรัง รหัสนักศึกษา 62010556

อาจารย์ที่ปรึกษา

……

(รศ.ดร.พิพัฒน์ พรหมมี)

กิตติกรรมประกาศ

การดำเนินโครงงานเรื่อง “การเข้ารหัสลับข้อมูลด้วยระบบบอลวนโดยใช้ FPAA” จะไม่สามารถสำเร็จลุล่วงไปด้วยดี หากไม่ได้รับความช่วยเหลือ และความอนุเคราะห์อย่างดียิ่งจากอาจารย์ที่ปรึกษา คือ รศ.ดร.พิพัฒน์ พรหมมี ที่ได้คอยให้คำปรึกษา คำสั่งสอน และคำแนะนำ และยังคอยตรวจสอบ แก้ไขข้อบกพร่องทุกขั้นตอนของการจัดทำโครงงานชิ้นนี้ รวมถึงสนับสนุนสถานที่ เครื่องมือ และอุปกรณ์ต่างๆ ที่ใช้ระหว่างการจัดทำโครงงาน รวมทั้งพี่นักศึกษาปริญญาโท ปริญญาเอก ภาควิชาวิศวกรรมโทรคมนาคมของห้อง T112-A ที่กรุณาให้คำแนะนำ คำปรึกษา และแนวทางการแก้ปัญหาที่เป็นประโยชน์ต่อการศึกษา ค้นคว้าวิจัยให้โครงงานนี้สำเร็จสมบูรณ์ยิ่งขึ้น

ขอกราบขอบพระคุณ บิดา มารดา และครอบครัว ที่ให้ความรัก ความห่วงใย และเป็นกำลังใจที่สำคัญเสมอมา และที่สำคัญคือการสนับสนุนให้โอกาสทางด้านการศึกษาอันมีค่ายิ่งแก่ผู้จัดทำ

นายกันตพงศ์ บุญมาก
นายชินพัฒน์ อมรสู่สวัสดิ์
นายปิติพัฒน์ สารปรัง
ผู้จัดทำ

การเข้ารหัสลับข้อมูลด้วยระบบอลวนโดยใช้ FPAA
DATA CRYPTOGRAPHY BASED ON CHAOTIC SYSTEMS USING FPAA

โดย นายกันตพงศ์ บุญมาก 62010048
นายชินพัฒน์ อมรสู่สวัสดิ์ 62010200
นายปิติพัฒน์ สารปรัง 62010556

อาจารย์ที่ปรึกษา รศ.ดร.พิพัฒน์ พรหมมี

บทคัดย่อ

ปริญญานิพนธ์นี้นำเสนอระบบการเข้ารหัสลับของข้อมูลด้วยระบบอลวน (Lorenz และ jerk) โดยระบบจะใช้บอร์ด Field Programmable Analog Array (FPAA) ทั้งภาคส่ง และภาครับ ซึ่งสามารถใช้ MATLAB/Simulink ในการจำลองผลสำหรับการปรับค่าพารามิเตอร์ต่างๆ ได้ ในส่วนการทดลองที่ภาคส่งจะใช้ข้อมูลรูปภาพดิจิทัลเข้ารหัสด้วยระบบอลวน และส่งผ่านพอร์ตสื่อสารแบบอนุกรม ซึ่งภาครับจะสามารถถอดรหัสได้สมบูรณ์หากระบบอลวนที่ใช้ และพารามิเตอร์ของระบบตรงกันกับภาคส่ง โดยในการทดลองจะใช้เวลาทั้งหมด 24 นาทีในการส่งทั้งหมด 10,002,432 บิต และสามารถวัดอัตราบิตผิดพลาดได้เท่ากับ 42.39×10^{-6} สำหรับ Lorenz system และ 38.29×10^{-6} สำหรับ jerk system

ABSTRACT

This senior project presents a data cryptography system based on chaotic systems (Lorenz and jerk). The system implementation consists of Field Programmable Analog Array (FPAA) and a microcontroller for transmit and receiver sides. MATLAB/Simulink provides simulation results in various parameters. The experimental setup is exhibited by applying the data of digital pictures, which are encrypted by the selected chaos model and transmitted by using a serial port. The receiver can be correctly decrypted if and only if the chosen chaos model and parameters are perfectly matched with the transmitter. The experimental results of BER were measured around 42.39×10^{-6} for Lorenz system and 38.29×10^{-6} for jerk system with a time of approximately 24 minutes to recover all 10,002,432 bits.

สารบัญ

สารบัญ (ต่อ)

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญตาราง

หน้า

ตารางที่

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

เนื่องจากระบบอลวน (Chaotic system) นั้นมีคุณสมบัติเป็นระบบเชิงกำหนดที่คาดเดาไม่ได้ และงานวิจัยเกี่ยวกับวิทยาการการเข้ารหัสลับด้วยระบบอลวนเป็นหัวข้อที่ได้รับความสนใจมากขึ้นในปัจจุบัน โครงงานนี้จึงมีจุดประสงค์ในการศึกษา และสร้างระบบอลวนเพื่อประยุกต์ใช้สำหรับความปลอดภัยด้านการสื่อสารด้วยอุปกรณ์ Field Programmable Analog Array (FPAA) ซึ่งจะเป็นชุดต้นแบบสำหรับการศึกษาเกี่ยวกับการเข้ารหัสลับด้วยระบบอลวนในอนาคตต่อไป

1.2 วัตถุประสงค์

1) เพื่อศึกษาระบบอลวน (Chaotic system)

2) นำระบบอลวนมาสร้าง และออกแบบเพื่อประยุกต์ใช้สำหรับการเข้ารหัสลับของข้อมูล

3) นำอุปกรณ์ Field Programmable Analog Array (FPAA) มาออกแบบเพื่อกำเนิดสัญญาณอลวน

1.3 ขอบเขตของปริญญานิพนธ์

1) ใช้ FPAA ในการกำเนิดสัญญาณอลวนเพื่อนำมาประยุกต์สำหรับการเข้ารหัสลับของข้อมูล

2) ใช้โปรแกรม AnadigmDesigner2 ในการสร้างระบบอลวนบน FPAA

3) ใช้กระบวนการ Chaotic masking communication สำหรับการเข้ารหัส และถอดรหัสข้อมูลในการสื่อสาร

บทที่ 2

ทฤษฎีและหลักการที่เกี่ยวข้อง

2.1 หลักการของการกำเนิดสัญญาณอลวน

2.1.1 ระบบเชิงเส้น, ระบบไม่เป็นเชิงเส้น และระบบอลวน

ระบบเชิงเส้น และไม่เป็นเชิงเส้นถูกใช้ในการอธิบายความสัมพันธ์ระหว่างฟังก์ชัน $y = f(x)$ และตัวแปรอิสระ x โดยระบบเชิงเส้นนั้นจะอ้างถึงความสัมพันธ์ที่เป็นสัดส่วนกันของสองตัวแปรอิสระ ซึ่งสามารถแทนได้ด้วยการเคลื่อนที่ที่สม่ำเสมอ และราบเรียบในพื้นที่ และเวลาหนึ่ง ส่วนระบบไม่เป็นเชิงเส้นจะอ้างถึงความสัมพันธ์ที่ไม่เป็นสัดส่วนกันของสองตัวแปรที่ไม่อิสระต่อกัน ซึ่งสามารถแทนได้ด้วยการเคลื่อนที่ที่ไม่สม่ำเสมอ และมีการเปลี่ยนแปลงในพื้นที่ และเวลาหนึ่ง ยกตัวอย่างเช่นองค์ประกอบทางความถี่ของสัญญาณเชิงเส้นจะเท่าเดิมเสมอแต่ถ้าหากเป็นความสัมพันธ์แบบไม่เป็นเชิงเส้นจะนำไปสู่การเปลี่ยนแปลงของความถี่ ไม่ว่าจะเป็น ความถี่ผลรวม, ความถี่ผลต่าง หรือความถี่สองเท่า เป็นต้น

การเปลี่ยนแปลงของระบบจะเกิดขึ้นจากความสัมพันธ์แบบไม่เป็นเชิงเส้น แต่การเปลี่ยนแปลงเล็กน้อยของความสัมพันธ์แบบไม่เป็นเชิงเส้นไม่สามารถนำไปสู่การเปลี่ยนแปลงพฤติกรรมของระบบได้ แต่พฤติกรรมของระบบสามารถเกิดการเปลี่ยนแปลงได้ทันทีเมื่อความสัมพันธ์แบบไม่เป็นเชิงเส้นไปถึงค่าที่ต้องการค่าหนึ่ง โดยการเปลี่ยนแปลงนั้นจะเกิดที่ชุดของค่าพารามิเตอร์ของระบบไม่เป็นเชิงเส้น และการเปลี่ยนแปลงแต่ละครั้งจะมาพร้อมกับองค์ประกอบความถี่ใหม่ ซึ่งในที่สุดระบบจะเข้าสู่สภาวะอลวน ดังนั้นหากเกิดระบบอลวนระบบนั้นจะต้องเป็นระบบไม่เป็นเชิงเส้น

ปรากฏการณ์อลวนเป็นปรากฏการณ์ที่ไม่สามารถคาดเดาได้ (Unpredictable) แต่กำหนดได้ (Determined) โดยเมื่อดูภายนอกแล้วเหมือนการเคลื่อนที่ที่เป็นการสุ่ม (Random motion) แต่เมื่อเปรียบเทียบกับการสุ่มแล้วระบบอลวนสามารถกำหนดค่าที่แน่นอนออกมาได้ และความคาดเดาไม่ได้ของระบบมาจากความไม่แน่นอนภายในของระบบ ระบบอลวนเป็นระบบที่อ่อนไหวต่อเงื่อนไขเริ่มต้น (Initial condition) และการเปลี่ยนแปลงเพียงเล็กน้อย ดังนั้นหลังจากเวลาที่ผ่านไปช่วงหนึ่ง ระบบจะเบี่ยงเบนออกจากทิศทางเดิม ดังนั้นระบบอลวนคือระบบที่ไม่สามารถคาดคะเนการเคลื่อนที่ที่เกิดจากระบบเชิงกำหนด (Deterministic system) [1]

2.1.2 ลักษณะของระบบอลวน

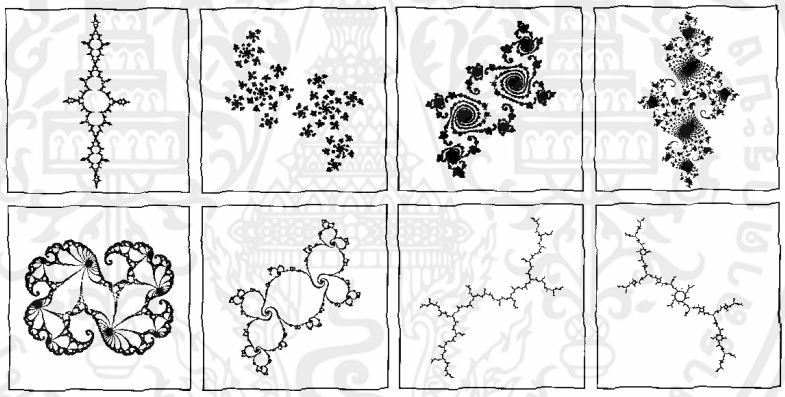
ระบบอลวนมีพฤติกรรมที่เป็นพลวัตที่มีขอบเขตไม่สม่ำเสมอในระบบพลวัตเชิงกำหนดแบบไม่เป็นเชิงเส้น ซึ่งอ่อนไหวต่อเงื่อนไขเริ่มต้น ที่เหมือนเป็นการสุ่ม (Random-like) ในระบบเชิงกำหนด โดยมีลักษณะสำคัญดังนี้

1. การมีขอบเขต (Boundedness) ระบบอลวนจะถูกกำหนดการเคลื่อนที่ให้อยู่ภายในพื้นที่หนึ่งที่เรียกว่า Chaotic attractor ซึ่งไม่ว่าระบบอลวนจะมีความไม่เสถียร (Instable) เพียงใดการเคลื่อนที่ของระบบก็จะอยู่แต่ใน Attractor

2. Ergodicity หมายถึงแนวทางที่ระบบอลวนจะผ่านทุกๆจุดสถานะภายในช่วงเวลาที่จำกัดและในพื้นที่ Attractor นั้นๆ

3. การสุ่ม (Randomness) ระบบอลวนมีพฤติกรรมที่ไม่มีความแน่นอนซึ่งสร้างจากระบบที่มีความแน่นอน และเป็นการสุ่มด้วยตัวมันเองโดยปราศจากปัจจัยภายนอก แม้ว่าสมการของระบบจะสามารถกำหนดได้ แต่ด้วยความเป็นพฤติกรรมแบบพลวัต (Dynamic) จะทำให้การกำหนดนั้นยากซึ่งใน Attractor จะมีฟังก์ชันความหนาแน่นความน่าจะเป็นไม่เท่ากับศูนย์

4. มิติแฟร็กทัล (Fractal Dimension) เป็นลักษณะพฤติกรรมของวิถีการเคลื่อนที่ของระบบอลวนในพื้นที่เฟส ที่มีโครงสร้างการเคลื่อนที่หลายชั้น และส่วนที่แตกแขนงย่อยที่มีโครงสร้างที่คล้ายกันอย่างไม่มีที่สิ้นสุด



รูปที่ 2.1 ภาพอันเกิดจาก จูเลีย เซ็ตส์ หลายรูปแบบ [2]

5. คุณสมบัติมาตราส่วน (Scaling property) หมายถึงการเคลื่อนที่ของระบบอลวนจะถูกกำหนดในสถานะที่ไม่กำหนดไม่ได้ นั่นคือไม่ว่าอุปกรณ์ในการวัดจะดีเพียงใดจะเห็นการเคลื่อนที่ของระบบในพื้นที่ Attractor ขนาดเล็กๆ

6. ความเป็นสากลกัน (Universality) หมายถึงระบบต่างกันจะแสดงคุณสมบัติที่เหมือนกันในการที่เข้าสู่สภาวะอลวน และไม่ต้องเปลี่ยนสมการของระบบ หรือพารามิเตอร์ใดๆ โดยระบบอลวนทุกระบบสามารถใช้ค่าคงที่เดียวในการระบุประสิทธิภาพ เช่น ค่าคงที่ Feigenbaum ซึ่งความเป็นสากลกันของระบบอลวนจะสะท้อนให้เห็นถึง ความสม่ำเสมอโดยธรรมชาติของระบบอลวน

7. Positive Lyapunov exponent โดย Lyapunov exponent เป็นค่าปริมาณที่บอกการเข้าใกล้ หรือออกจากของเส้นทางที่สร้างโดยระบบไม่เป็นเชิงเส้น ซึ่ง Positive

Lyapunov exponent จะบ่งบอกถึงเส้นทางที่เกิดความไม่เสถียรขึ้นในแต่ละช่วง และเส้นทางที่แยกออกจากกัน นอกจากนี้ยังบอกการสูญเสียของข้อมูลของแต่ละจุด โดยยิ่งมีค่ามากจะมีการสูญหายของข้อมูลที่มากขึ้น และมีดีกรีของระบบอลวนที่มากขึ้น [1]

2.2 รูปแบบของสัญญาณอลวน

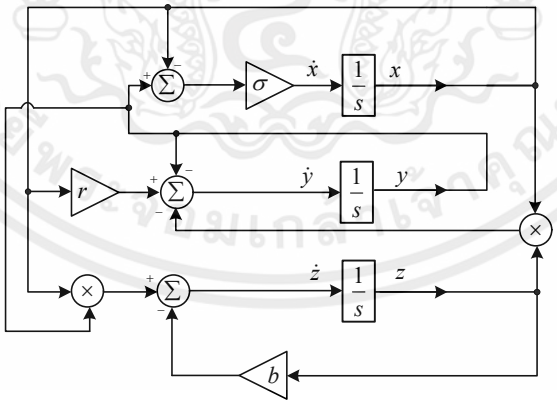
2.2.1 Lorenz System

ถูกค้นพบโดย Edward Lorenz ขณะที่กำลังศึกษาแบบจำลองการพาความร้อนในชั้นบรรยากาศ โดยได้ทำการสรุปสมการสามมิติ และพบปรากฏการณ์ผีเสื้อ (Butterfly effect) ซึ่งใช้อธิบายความซับซ้อนของการมีหลายปัจจัยเข้ามาเกี่ยวข้องในเหตุการณ์ จนเกิดผลกระทบที่เปลี่ยนแปลงออกไปแบบคาดไม่ถึงเช่น การที่ผีเสื้อตัวเล็กกระพือปีกในที่หนึ่งอาจทำให้เกิดการเปลี่ยนแปลงของบรรยากาศขนาดเป็นพายุใหญ่ขึ้นมาได้ [3] สำหรับระบบของ Lorenz จะเป็นสมการอนุพันธ์ไม่เป็นเชิงเส้นอันดับที่หนึ่งสามมิติประกอบด้วยสามตัวแปร ซึ่งในท้ายที่สุดได้กลายมาเป็นตัวอย่างในการศึกษาทฤษฎีอลวน และการประยุกต์ใช้ [1]

โดยระบบของ Lorenz สามารถเขียนเป็นสมการได้ดังนี้ [4]

$$\begin{cases} \dot{x} = \sigma(y-x), \\ \dot{y} = rx - y - xz, \\ \dot{z} = xy - bz. \end{cases} \tag{2.1}$$

และสามารถเขียนออกมาเป็น Block Diagram ได้ดังรูปที่ 2.2

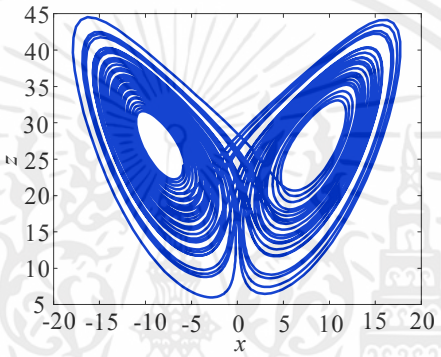


รูปที่ 2.2 Block Diagram ของ Lorenz system

โดยที่ σ, r และ b คือพารามิเตอร์ของระบบ สมการอนุพันธ์อันดับที่สามจะอิงมาจากโมเดลอย่างง่ายของการถ่ายเทความร้อนระหว่างที่ราบ โดยที่ตัวแปรของสมการจะไม่มีเวลา จะเรียกสมการนี้ว่า Autonomous Equation โดย x แทนด้วยกำลังในการพาความร้อน; y แทนด้วยความแตกต่างอุณภูมิ

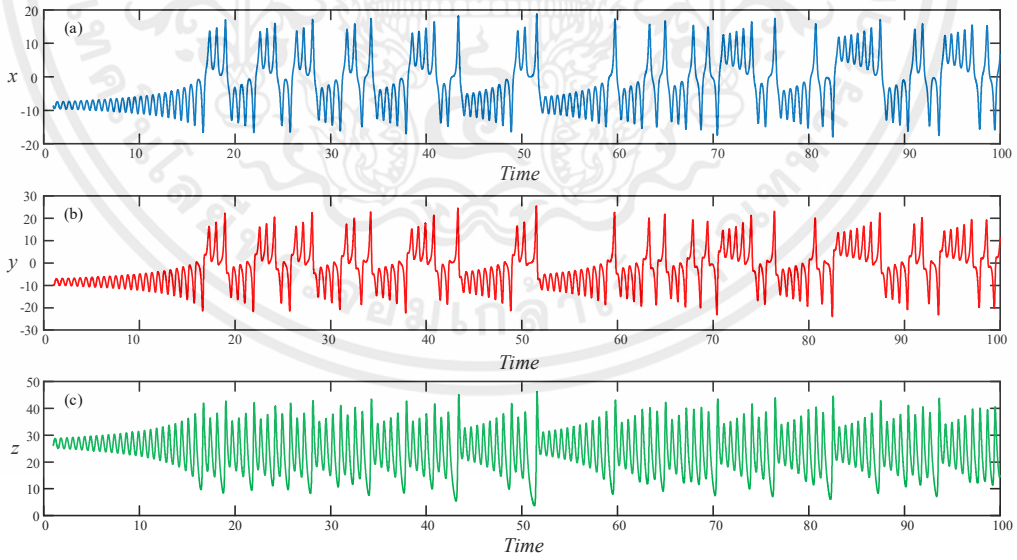
ระหว่างอากาศที่ไหลขึ้น และอากาศที่ไหลลง; z แทนด้วยความหนาแน่นแบบไม่เป็นเชิงเส้นของการกระจายอุณหภูมิในแนวตั้ง; r แทนด้วยอัตราส่วนระหว่างปัจจัยในการขับเคลื่อน และปัจจัยยับยั้ง ที่ทำให้เกิดการพาความร้อน และกระแสอากาศที่แปรปรวน; b แทนด้วยอัตราส่วนรูปร่างที่เกี่ยวกับการพาความร้อน; σ คือ Prandtl value โดยทั้ง b และ σ เป็นค่าคงที่ ซึ่งตัวแปรเหล่านี้เป็นพารามิเตอร์หลักในการควบคุมระบบ

โดยถ้าหากให้ $\sigma = 10$, $r = 28$ และ $b = 8/3$ และทำการจำลองระบบด้วยโปรแกรม MATLAB สำหรับระบบของ Lorenz จะได้ Attractor ที่มีลักษณะคล้ายผีเสื้อดังรูปที่ 2.3



รูปที่ 2.3 Lorenz attractor บนระนาบ x-z

สามารถดูรูปสัญญาณเอาต์พุตของระบบ Lorenz ตามแกนเวลาแสดงได้ดังรูปที่ 2.4



รูปที่ 2.4 รูปสัญญาณ Lorenz system เอาต์พุต; (a) x เทียบกับเวลา (b) y เทียบกับเวลา (c) z เทียบกับเวลา

2.2.2 Jerk model

ในระบบของ Lorenz ที่มีตัวสัมประสิทธิ์ทั้งหมด 7 ตัว และฟังก์ชันไม่เป็นเชิงเส้น 2 ฟังก์ชัน เพื่อลดความซับซ้อนในการสร้าง และศึกษาระบบอลวน ระบบอันดับสาม (Thid-order system) หรือ Jerk model จึงถูกใช้ในการสร้างสัญญาณอลวนโดยมีข้อได้เปรียบคือการที่มีสัมประสิทธิ์ที่น้อยกว่า และมีความหลากหลายกว่าในการใช้ฟังก์ชันไม่เป็นเชิงเส้นมาประยุกต์ใช้ในการสร้างสัญญาณอลวนรูปแบบต่างๆ โดยปราศจากการใช้ฟังก์ชันคูณ (Multiplier function) [5]

โดย Jerk equation [6] สามารถเขียนในรูปอนุพันธ์อันดับสามของการกระจัดในทางกลศาสตร์ได้ดังนี้

$$\dddot{x} = F(\ddot{x}, \dot{x}, x) \tag{2.2}$$

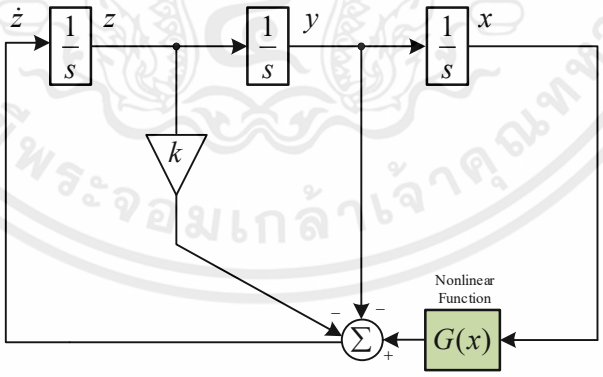
โดยที่ $x, \dot{x}, \ddot{x}$ และ $\dddot{x}$ คือการกระจัด, ความเร็ว, ความเร่ง และการกระตุก (Jerk) ตามลำดับ ซึ่งในระบบอันดับสามนี้ยังประกอบไปด้วยค่าสัมประสิทธิ์ (k) และฟังก์ชันไม่เป็นเชิงเส้น $(G(x))$ โดยสามารถแสดงเป็นสมการได้ดังนี้

$$\dddot{x} = -k\ddot{x} - \dot{x} + G(x) \tag{2.3}$$

หรือ

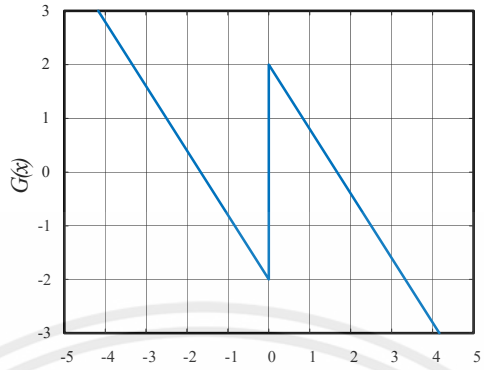
$$\begin{cases} \dot{x} = y, \\ \dot{y} = z, \\ \dot{z} = -kz - y + G(x). \end{cases} \tag{2.4}$$

และสามารถเขียนออกมาเป็น Block Diagram [7] ได้ดังรูปที่ 2.5

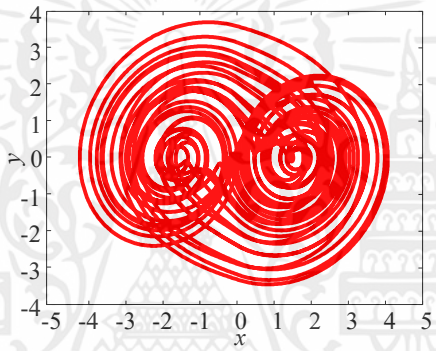


รูปที่ 2.5 Block Diagram ของ Jerk system

จากรูปที่ 2.5 จะทำการสร้างสัญญาณอลวนได้โดยการปรับค่าสัมประสิทธิ์ k และฟังก์ชันไม่เป็นเชิงเส้น $G(x)$ โดยถ้าให้ $k = 0.6$ และ $G(x) = -Bx + Csign(x)$ และให้ $B = 1.2, C = 2$ จะได้ ดังรูปที่ 2.6 และจะทำให้ระบบเข้าสู่สภาวะอลวน โดยสามารถแสดงได้ดังรูปที่ 2.7

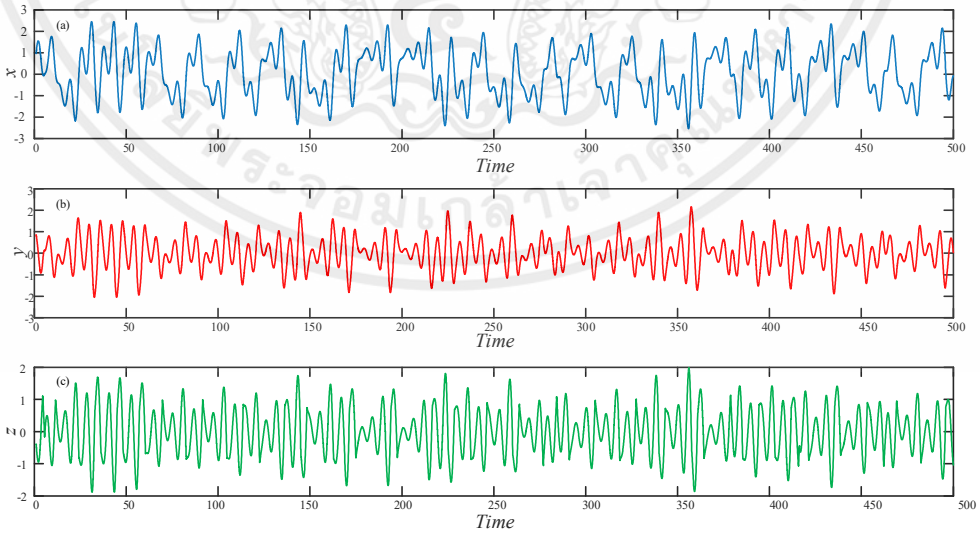


รูปที่ 2.6 ฟังก์ชัน $G(x) = -Bx + Csign(x)$ โดยให้ $B = 1.2, C = 2$



รูปที่ 2.7 Chaotic attractor ของ Jerk model บนระนาบ x-y

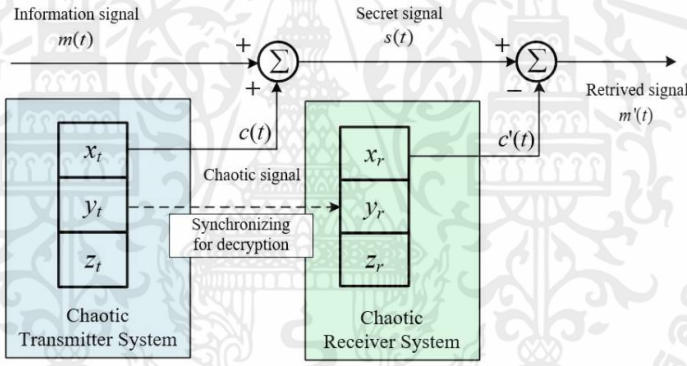
สามารถดูรูปสัญญาณเอาต์พุตของ Jerk model ตามแกนเวลาแสดงได้ดังรูปที่ 2.8



รูปที่ 2.8 รูปแบบสัญญาณ jerk system เอาต์พุต; (a) x เทียบกับเวลา (b) y เทียบกับเวลา (c) z เทียบกับเวลา

2.3 หลักการ Chaotic masking communication ของระบบอลวน [1]

หลักการ Chaotic masking communication เป็นวิธีที่ให้ความสะดวกอย่างมากในการนำมาประยุกต์ใช้กับการสื่อสารด้วยระบบอลวนเนื่องจากเป็นวิธีการเข้ารหัสที่ง่ายที่สุดสำหรับการสื่อสารแบบระบบแอนาลอก โดยภาคส่งสัญญาณจะทำการนำสัญญาณอลวนมารวมเข้ากับสัญญาณข้อมูลแล้วทำการส่งไปในช่องสัญญาณ สัญญาณที่ส่งไปในช่องสัญญาณนี้เรียกว่า สัญญาณลับ (Secret Signal) ซึ่งนอกเหนือจากสัญญาณลับที่ส่งออกไปแล้ว ภาคส่งจะต้องนำตัวแปรสถานะ (State Variable) ของระบบอลวนทางภาคส่งส่งสัญญาณควบคู่กันไปด้วย การส่งสัญญาณที่ควบคู่ไปกับสัญญาณลับนี้ จะทำให้ระบบการสื่อสารเกิดการ Synchronization [8,9] กันของภาคส่งและภาครับ โดยที่ตัวแปรสถานะที่ส่งไปนี้จะถูกระบบอลวนทางภาครับนำมาสร้างสัญญาณอลวนที่มีลักษณะเหมือนกันกับภาคส่งในทุกประการแล้วนำสัญญาณอลวนที่กำเนิดขึ้นทางภาครับมาลบออกจากสัญญาณลับที่เข้ามา จึงจะสามารถกู้คืนสัญญาณข้อมูลจากภาคส่งได้ โดยสามารถแสดงได้ดังรูปที่ 2.9

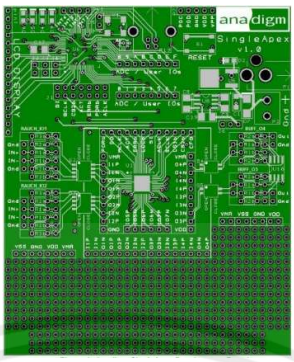


รูปที่ 2.9 รูปแบบการสื่อสารโดยอาศัยหลักการ Chaotic masking communication

2.4 การใช้งาน FPAA ด้วยโปรแกรม AnadigmDesigner2

2.4.1 Anadigm SingleApex Development Board

เป็นแพลตฟอร์มที่ออกแบบมาให้ใช้งานได้ง่ายในการประยุกต์ใช้ หรือทดสอบการออกแบบวงจรแอนาลอกบนชิปของ FPAA เช่น ชิปรุ่น AN231E04 dpASP โดยมี 32bit PIC32 microcontroller ภายในตัว และเป็นแพลตฟอร์มที่สามารถพัฒนาไปเป็นระบบฝังตัว (Embedded System) ได้อย่างมีประสิทธิภาพ [10]

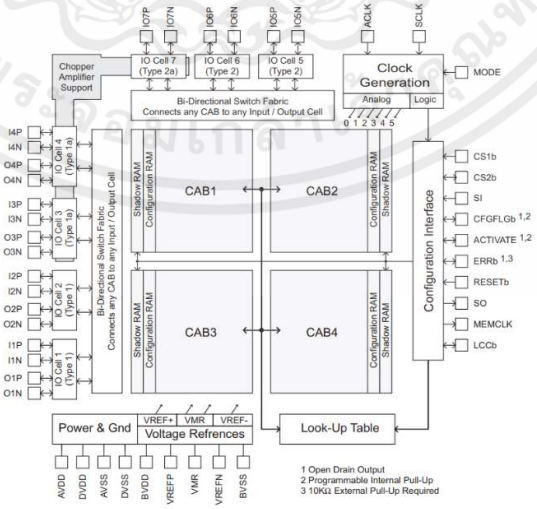


รูปที่ 2.10 Anadigm SingleApex Development Board

2.4.2 AN231E04 dpASP [11]

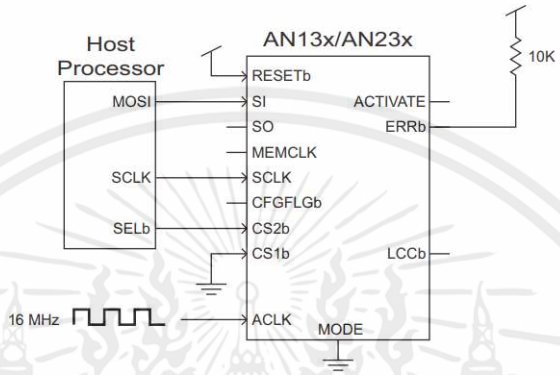
เป็นหน่วยประมวลผลสัญญาณแอนาลอกเหมาะสำหรับการปรับสภาพสัญญาณ, การกรองสัญญาณ (Filtering), การเพิ่มขนาดของสัญญาณ (Gain), การแก้ไขสัญญาณให้ถูกต้อง (Rectification), การบวกสัญญาณ (Summing), การลบสัญญาณ (Subtracting) หรือการคูณสัญญาณ (Multiplying) เป็นต้น นอกจากนี้หน่วยประมวลผลยังรองรับฟังก์ชันที่ไม่เป็นเชิงเส้น (Nonlinear function) เช่นการตอบสนองของเซนเซอร์ และการสังเคราะห์รูปคลื่น

AN231E04 จะประกอบไปด้วยเมทริกซ์ 2x2 แบบเต็มของ Configurable Analogs Blocks (CABs) ซึ่งถูกล้อมด้วยส่วนที่เป็นแหล่งจ่าย และอยู่ระหว่างองค์ประกอบที่สามารถโปรแกรมได้ กับเซลล์อินพุต และเอาต์พุตแบบแอนะลอกที่มีองค์ประกอบแบบแอคทีฟ (Active elements) และตัวสร้างสัญญาณนาฬิกา โดยควบคุมการสร้างสัญญาณนาฬิกาที่ไม่ซ้อนทับกันจากแหล่งกำเนิดสัญญาณนาฬิกาคงที่ภายนอก และประกอบด้วย look-up table ขนาด 8x256 บิตที่ทำให้สามารถสังเคราะห์รูปสัญญาณ และฟังก์ชันไม่เป็นเชิงเส้นขึ้นมาได้



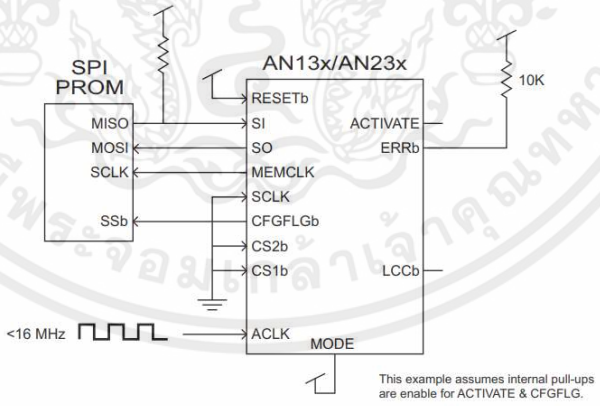
รูปที่ 2.11 โครงสร้างภาพรวมของ AN231E04

ในการบันทึกค่าข้อมูลที่กำหนดลงผ่านโปรแกรมจะถูกบันทึกบนหน่วยความจำ Static ram มีอินเทอร์เฟซแบบ SPI สำหรับการอัพโหลดข้อมูลแบบอนุกรมผ่านไมโครโพรเซสเซอร์ หรือ DSP ซึ่งจะทำให้สามารถกำหนดค่าของวงจรได้แบบพลวัต (Dynamic) โดยที่ไม่รบกวนกับฟังก์ชันการทำงานของวงจรขณะที่วงจรทำงานดังรูปที่ 2.12



รูปที่ 2.12 การเชื่อมต่อกับส่วน Host Processor ของ AN231E04 สำหรับการทำงานแบบพลวัต (Dynamic Operation)

ในการทำงานแบบคงที่ (Static operation) AN231E04 จะทำคำสั่งผ่านขา SO เพื่อข้อมูลที่กำหนดจาก SPI PROM และทำการส่งข้อมูลกลับแบบอนุกรมซึ่งจะถูกอ่านด้วยขา SI ดังรูปที่ 2.13



รูปที่ 2.13 การเชื่อมต่อกับส่วน Host Processor ของ AN231E04 สำหรับการทำงานแบบคงที่ (Static Operation)

สำหรับ Configurable Analogs Blocks (CABs) จะเป็นส่วนที่ใช้ในการประมวลผลสัญญาณแบบแอนะลอกบนสถาปัตยกรรมของวงจร switched capacitor ซึ่งในการประมวลสัญญาณจะทำแบบ Differential ทั้งหมดเพื่อเพิ่มความเที่ยงตรงของสัญญาณ และในทุกๆ CAB จะปรับไปด้วย

ออปแอมป์สองตัว, วงจรตัวเปรียบเทียบแรงดัน, ตัวเก็บประจุที่โปรแกรมได้ และแหล่งกำเนิดสัญญาณนาฬิกาโดยสัญญาณที่เซลล์อินพุต และเอาต์พุตที่เชื่อมต่อกับการใช้งานภายนอกจะอยู่ในรูปแบบ Differential เพื่อเป็น smoothing filter [12]

2.4.3 AnadigmDesigner2 [13]

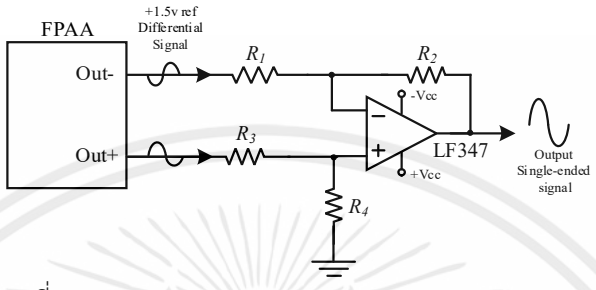
เป็นซอฟต์แวร์ที่ใช้ในการออกแบบ และประยุกต์ใช้วงจรแอนาลอก โดยสามารถกำหนดค่าพารามิเตอร์ต่างๆลงไปได้ โดยในการสร้างวงจรจะใช้การลาก Configurable Analog Modules (CAMs) มาวางซึ่งจะสามารถสร้างรวมกันออกมาเป็นวงจรแอนาลอกได้มากมายตามค่าพารามิเตอร์ที่ได้กำหนดไว้ นอกจากนี้ในซอฟต์แวร์ยังมีส่วนเสริมในการจำลองผลการทดสอบตามโดเมนของเวลาซึ่งช่วยให้สะดวกในการประเมินผลวงจรที่ได้ออกแบบได้โดยไม่ได้ต้องจัดเตรียมชุดปฏิบัติการ

ในส่วนการทำงาน AnadigmDesigner2 เป็น EDA ที่สามารถพัฒนาการออกแบบการใช้ FPAA ที่สามารถกำหนดค่าของวงจรที่ออกแบบผ่าน Microprocessor ได้โดย AnadigmDesigner2 จะนำการออกแบบบนหน้า GUI แปลเป็นภาษา C เพื่อ ให้ Microprocessor สามารถปรับ หรือควบคุมฟังก์ชันการทำงานของระบบที่ออกแบบได้ทันที โดยที่ยังมีความแม่นยำ

รูปที่ 2.14 หน้าต่างของโปรแกรม AnadigmDesigner2

2.5 วงจร Differential to Single-ended Output

วงจร Differential to Single-ended Output มีการใช้ออปแอมป์ เพียง 1 ตัว โดยทำการป้อนอินพุตที่มีค่าเป็นบวกเข้าที่ R_3 และอินพุตที่มีค่าเป็นลบเข้าที่ R_1



รูปที่ 2.15 วงจร Differential voltage to single mode

โดยสัญญาณเอาต์พุตที่ได้จะมีค่าดังสมการ

$$V_{out} = V_p\left(\frac{R_4}{R_3}\right)\left(\frac{1+\frac{R_2}{R_1}}{1+\frac{R_4}{R_3}}\right) - V_n\left(\frac{R_2}{R_1}\right) \tag{2.5}$$

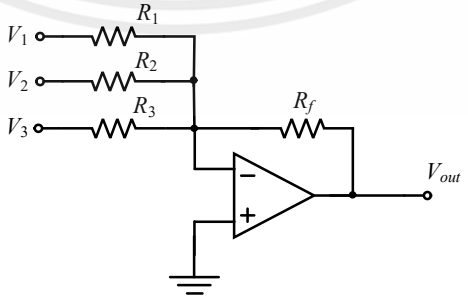
โดยเมื่อกำหนดให้ $R_2 = R_4 = R_f$ และ $R_1 = R_3 = R_g$ จะได้สมการ

$$V_{out} = \left(V_p - V_n\right)\frac{R_f}{R_g} \tag{2.6}$$

ดังนั้นจึงสามารถกำหนดอัตราขยายของสัญญาณเอาต์พุตได้ โดยขึ้นกับ $\frac{R_f}{R_g}$

2.6 วงจรขยายสัญญาณแบบรวมสัญญาณ

วงจรขยายสัญญาณแบบรวมสัญญาณเป็นวงจรที่ใช้ออปแอมป์ในการรวมอินพุตตั้งแต่ 2 อินพุตขึ้นไปมารวมกัน โดยแรงดันเอาต์พุตจะเท่ากับผลรวมของแรงดันอินพุตคูณกับอัตราขยาย [14]



รูปที่ 2.16 วงจรขยายสัญญาณแบบรวมสัญญาณ (Summing Amplifier)

จากรูปที่ 8 สามารถหาแรงดันเอาต์พุต V_{out} ได้จากสมการ

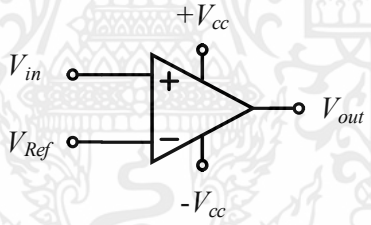
$$V_{\text{out}} = -\left(\frac{R_f}{R_1}V_1 + \frac{R_f}{R_2}V_2 + \frac{R_f}{R_3}V_3\right) \tag{2.7}$$

โดยเมื่อกำหนดให้ R_1, R_2 และ $R_3 = R_{\text{in}}$ จะได้สมการ

$$V_{\text{out}} = -\frac{R_f}{R_{in}}\left(V_1 + V_2 + V_3\right) \tag{2.8}$$

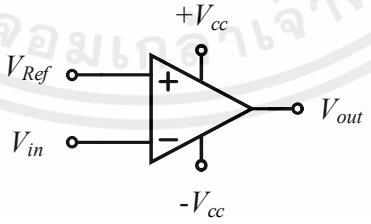
2.7 วงจรเปรียบเทียบแรงดัน

วงจรเปรียบเทียบแรงดัน เป็นวงจรที่ทำการเปรียบเทียบสัญญาณสองสัญญาณโดยใช้ออปแอมป์ ซึ่งจะต้องมีอินพุตสำหรับค่าแรงดันอ้างอิง V_{Ref} และอินพุตสำหรับสัญญาณที่ต้องการนำมาเปรียบเทียบ V_{in} กับค่าแรงดันอ้างอิง หากป้อนแรงดันอินพุตเข้าที่ขาบวก จะเรียกว่าวงจรเปรียบเทียบแบบไม่กลับเฟส (Non-inverting Comparator Circuit) และถ้าป้อนแรงดันอินพุตเข้าที่ขาลบ จะเรียกว่าวงจรเปรียบเทียบแบบกลับเฟส (Inverting Comparator Circuit) [15] โดยแรงดันเอาต์พุต V_{out} จะถูกจำกัดอยู่ไม่เกินแรงดันที่ป้อนให้กับออปแอมป์ ($+V_{\text{cc}}$ และ $-V_{\text{cc}}$)



รูปที่ 2.17 วงจรเปรียบเทียบแบบไม่กลับเฟส (Non-inverting Comparator Circuit)

จากรูปที่ 2.17 หาก $V_{\text{in}} > V_{\text{Ref}}$ จะได้ $V_{\text{out}} = +V_{\text{cc}}$ และ $V_{\text{in}} < V_{\text{Ref}}$ จะได้ $V_{\text{out}} = -V_{\text{cc}}$



รูปที่ 2.18 วงจรเปรียบเทียบแบบกลับเฟส (Inverting Comparator Circuit)

จากรูปที่ 2.18 หาก $V_{\text{in}} > V_{\text{Ref}}$ จะได้ $V_{\text{out}} = -V_{\text{cc}}$ และ $V_{\text{in}} < V_{\text{Ref}}$ จะได้ $V_{\text{out}} = +V_{\text{cc}}$

2.8 วงจรกรองความถี่ต่ำผ่านแบบแพสซิฟ

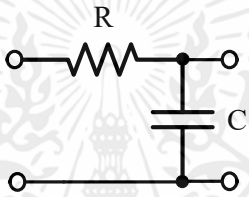
เป็นวงจรที่ทำหน้าที่ในการคัดกรองเอาสัญญาณที่มีความถี่สูงที่ไม่ต้องการออก โดยสามารถกำหนดช่วงของความถี่ที่ต่ำ (Cutoff frequency) ที่ต้องการได้ผ่านการคำนวณจากสมการ

$$f_c = \frac{1}{2\pi RC} \tag{2.9}$$

เมื่อกำหนดให้ f_c คือช่วงของความถี่ต่ำที่สามารถผ่านได้

R คือค่าความต้านทานที่ใช้ในวงจร

C คือค่าความจุของตัวเก็บประจุที่ใช้ในวงจร



รูปที่ 2.19 วงจรกรองความที่ต่ำผ่านแบบแพสซิฟโดยใช้ตัวต้านทานและตัวเก็บประจุ

2.9 ภาษา Python

ภาษา Python เป็นภาษาที่นำลักษะที่ดีของภาษาเดิมที่มีอยู่ (ABC, Modula-3, C, C++, Algol-68, SmallTalk and Unix shell and other scripting languages) และเพิ่มคุณลักษณะที่ดี เช่น คลาส และอื่นๆรวมถึงมี Interface ให้เขียนโปรแกรมได้สะดวกขึ้น [16]

2.9.1 Python Imaging Library (PIL)

PIL ย่อมาจากคำว่า (Python Imaging Library) หรือในเวอร์ชั่นใหม่จะเรียกว่า Pillow เป็นฟรี open-source ซึ่งเป็นไลบรารี่ของ ภาษา Python ที่เพิ่มการรองรับสำหรับการเปิดการจัดการและการบันทึกไฟล์รูปภาพหลายรูปแบบ พร้อมใช้งานสำหรับ Windows, Mac OS X และ Linux [17] ความสามารถของ PIL ประกอบไปด้วยขั้นตอนมาตรฐานสำหรับการปรับแต่งภาพ เช่น การกรองภาพ การเบลอเส้นขอบ หรือการปรับปรุงความคมชัด ความสว่าง หรือสีเป็นต้น โดยรูปแบบไฟล์ที่รองรับได้แก่ PPM, PNG, JPEG, GIF, TIFF และ BMP

2.9.2 NumPy

NumPy (Numeric Python) เป็นโมดูลส่วนเสริมของ Python ที่มีฟังก์ชันเกี่ยวกับคณิตศาสตร์และการคำนวณต่างๆ มาให้ใช้งาน โดยทั่วไปจะเกี่ยวกับการจัดการข้อมูลชุด (Array) ขนาดใหญ่และเมทริกซ์ [18] คุณสมบัติหลักของ NumPy คือ ชุดข้อมูล (Array) ซึ่งคล้ายกับ list

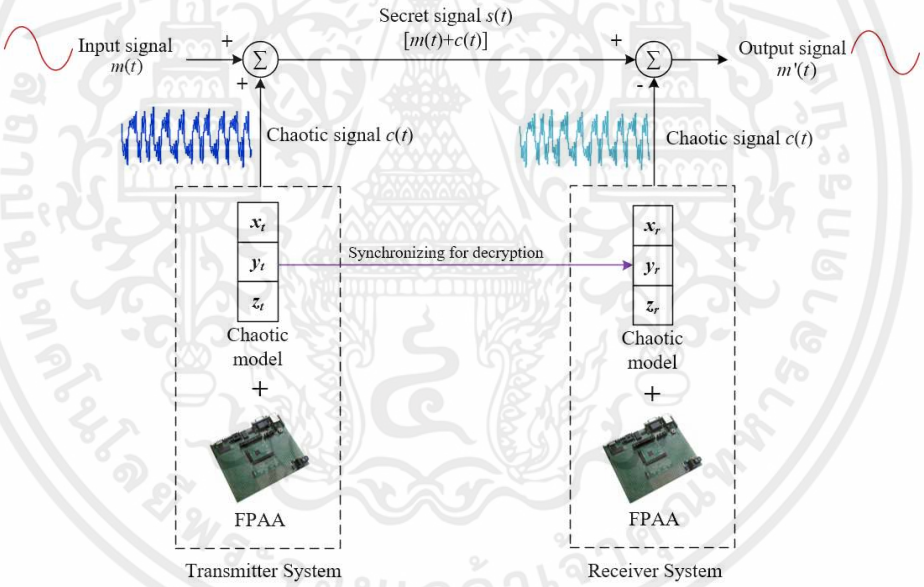
แต่สมาชิกในชุดข้อมูลต้องเป็นข้อมูลชนิดเดียวกันโดยทั่วไปจะเป็นตัวเลข เช่น float หรือ int ซึ่ง NumPy สามารถดำเนินการกับชุดข้อมูลที่เป็นตัวเลขได้อย่างมีประสิทธิภาพ

บทที่ 3

การออกแบบและการจัดทำปริญญานิพนธ์

3.1 การออกแบบระบบโดยรวม

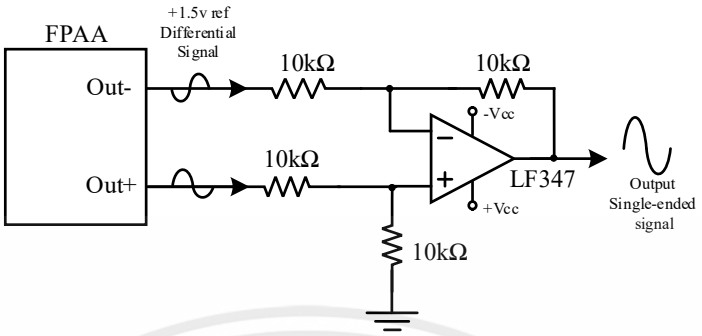
ในการจัดทำปริญญานิพนธ์เล่มนี้ ผู้จัดทำได้ออกแบบการประยุกต์ใช้ระบบอลวนสำหรับการเข้ารหัสลับข้อมูลด้วย FPAA ซึ่งในบทนี้จะนำเสนอการออกแบบโดยภาพรวมของการทำงานดังรูปที่ 3.1 โดยในการใช้ระบบอลวน (Chaotic system) มาประยุกต์ใช้การเข้ารหัสลับข้อมูลนั้นจะทำการสร้างสัญญาณอลวน (Chaotic signal) จากอุปกรณ์ Field Programmable Analog Array (FPAA) ซึ่งออกแบบด้วยโปรแกรม AnadigmDesigner2 และใช้กระบวนการ Chaotic masking communication สำหรับการเข้ารหัส และถอดรหัสข้อมูลในการสื่อสาร



รูปที่ 3.1 บล็อกไดอะแกรมสำหรับการเข้า-ถอดรหัสโดยใช้ FPAA

3.1.1 การออกแบบวงจร Differential voltage to single mode

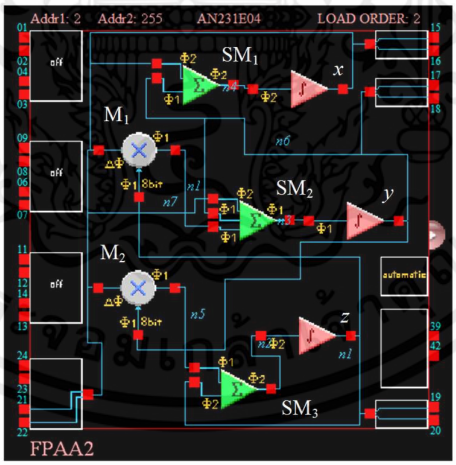
เนื่องจากสัญญาณ output port ของ FPAA จะถูกทำให้อยู่ในรูปแบบ Differential และจะถูกยกระดับของสัญญาณขึ้นไปอีก +1.5 V ดังนั้นหากจะได้สัญญาณของ Output ที่ถูกต้องตามที่ต้องการจะต้องใช้วงจร Differential voltage to single mode โดยฝั่ง Input ของวงจรนี้จะต้องรับ Output จาก FPAA ดังรูปที่ 3.2



รูปที่ 3.2 วงจร Differential voltage to single mode ที่ออกแบบ

3.1.2 การออกแบบ Lorenz system บน FPAA

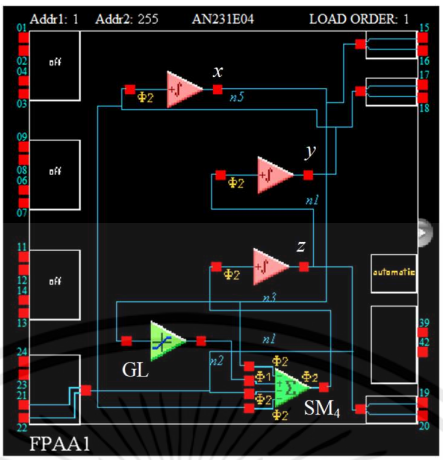
สำหรับการใช้ FPAA มาประยุกต์ในการสร้าง Lorenz system ขนาดของสัญญาณจะมีขนาดเล็กกว่าผลการจำลองด้วย MATLAB เนื่องจากแรงดันไฟฟ้าที่ FPAA (AN231E04) จะทำงานในช่วง ±1.5 V และ Parameters ของ CAMs จะขึ้นอยู่กับความถี่ที่ FPAA ใช้ ดังนั้นใน Lorenz system จึงมีการปรับค่า Parameters และ Option ต่างๆของ CAMs ใกล้เคียงกับผลการจำลองด้วยโปรแกรม MATLAB โดยสารมารถแสดงการออกแบบ Lorenz system บน AnadigmDesigner2 ด้วยการใช้ CAMs ดังนี้ Integrator, Multiplier (M_1, M_2), SumDiff (SM_1, SM_2, SM_3) โดยสารมารถแสดง ได้ดังรูปที่ 3.3



รูปที่ 3.3 การออกแบบ Lorenz system บน FPAA ด้วยโปรแกรม AnadigmDesigner2

3.1.3 การออกแบบ Jerk system บน FPAA

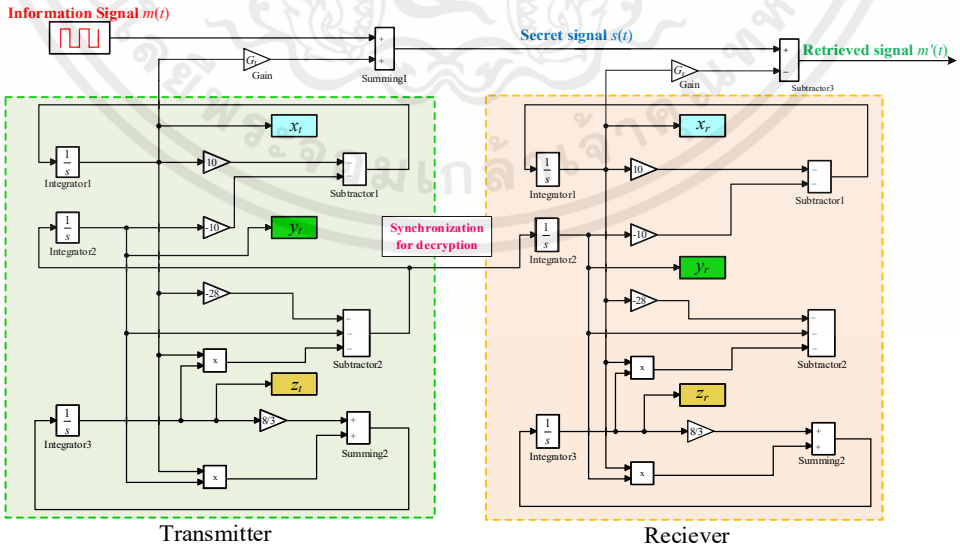
ทำนองเดียวกับ Lorenz system ใน Jerk system จึงมีการปรับค่า Parameters และ Option ต่างๆของ CAMs ให้ใกล้เคียงกับผลการจำลองด้วยโปรแกรม MATLAB โดยสามารถแสดงการออกแบบ Jerk system บน AnadigmDesigner2 ด้วยการใช้ CAMs ดังนี้ Integrator, SumDiff (SM4), and GainLimiter (GL) โดยสามารถแสดงได้ดังรูปที่ 3.4



รูปที่ 3.4 การออกแบบ jerk system บน FPAA ด้วยโปรแกรม AnadigmDesigner2

3.1.4 การออกแบบกระบวนการ Chaotic masking ของ Lorenz system ด้วย SIMULINK

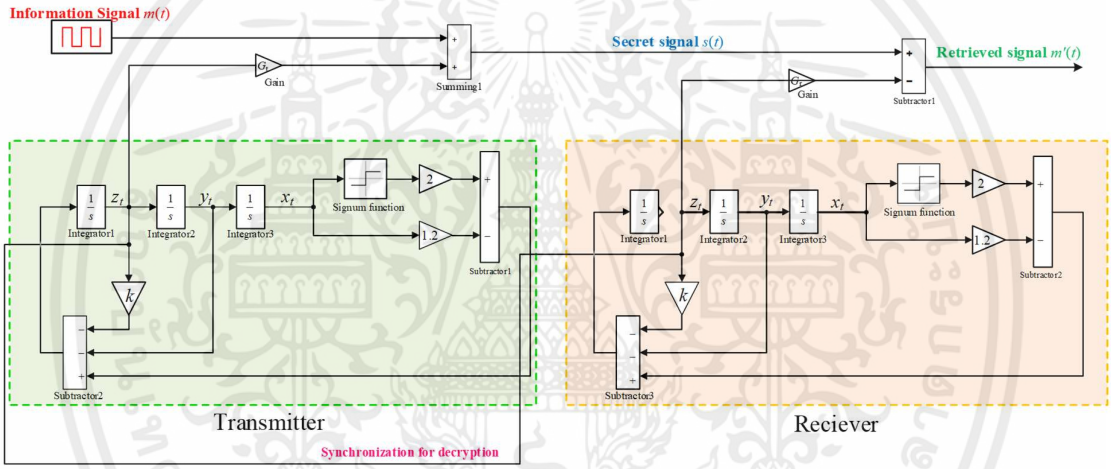
ในการออกแบบระบบ Chaotic masking ของ Lorenz system จะออกแบบระบบโดยผ่านโปรแกรม SIMULINK โดยทำการสร้าง Lorenz system ภาคส่ง (Transmitter) และภาครับ (Receiver) โดยในภาคส่งจะใช้ตัวแปรระบบ x ไปผ่านวงจรขยายสัญญาณไปยังวงจรรวมสัญญาณ (Summing) เพื่อใช้สัญญาณอลวน (Chaotic signal) ในการบดบังข้อมูล (Information Signal) และทำการเชื่อมตัวแปรระบบ y ไปยังภาครับเพื่อเป็นสัญญาณในการถอดรหัสสัญญาณ และทำการนำสัญญาณที่เข้ารหัส (Secret signal) ไปผ่านวงจรลบสัญญาณ (Subtractor) ด้วยตัวแปรระบบ x ที่ภาครับ เพื่อกู้สัญญาณข้อมูลกลับมา (Retrieved Signal) โดยสามารถแสดงได้ดังรูปที่ 3.5



รูปที่ 3.5 การสื่อสาร Chaotic masking communication ด้วย Lorenz system

3.1.5 การออกแบบกระบวนการ Chaotic masking communication ของ jerk system ด้วย SIMULINK

ในการออกแบบระบบ Chaotic masking communication ของ jerk system จะออกแบบระบบโดยผ่านโปรแกรม SIMULINK โดยทำการสร้าง jerk system ภาคส่ง (Transmitter) และภาครับ (Receiver) โดยในภาคส่งจะใช้ตัวแปรระบบ z ไปผ่านวงจรขยายสัญญาณไปยังวงจรรวมสัญญาณ (Summing) เพื่อใช้สัญญาณอลวน (Chaotic signal) ในการบดบังข้อมูล (Information Signal) และทำการเชื่อมตัวแปรระบบ z ไปยังภาครับเพื่อเป็นการถอดรหัสสัญญาณ และทำการนำสัญญาณที่เข้ารหัส (Secret signal) ผ่านวงจรลบสัญญาณ (Subtractor) ด้วยตัวแปรระบบ z ที่ภาครับ เพื่อที่จะกู้สัญญาณข้อมูลกลับมา (Retrieved Signal) โดยสามารถแสดงได้ดังรูปที่ 3.6

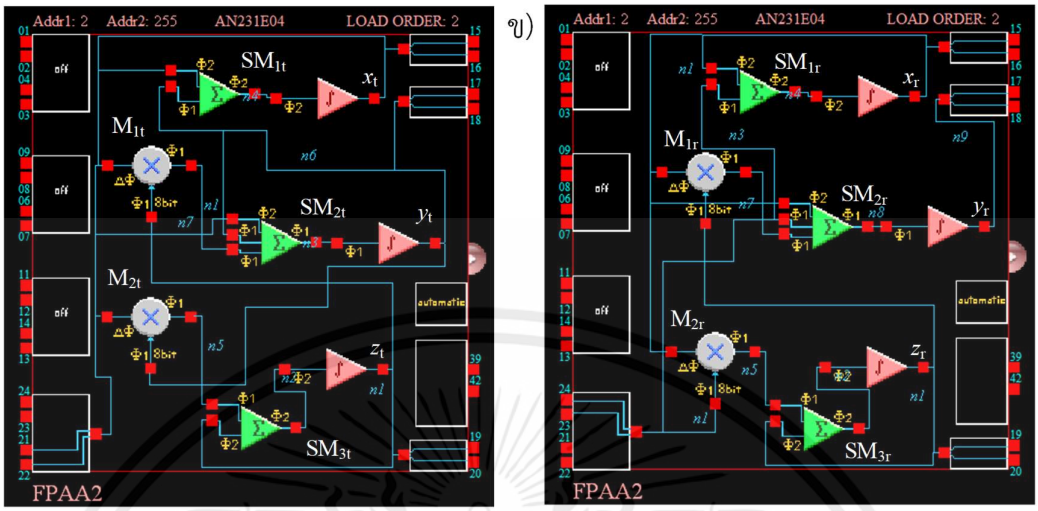


รูปที่ 3.6 การสื่อสาร Chaotic masking communication ด้วย jerk system

3.1.6 การออกแบบ Synchronization ของ Lorenz system ภาคส่ง และภาครับบน FPAA

ในการ Synchronization ของ Lorenz system บน FPAA จะทำการสร้างระบบ Lorenz system บน FPAA ดังที่แสดงไปในรายงานความคืบหน้าก่อนหน้า โดยจะสร้างเหมือนกันทั้งภาคส่ง และภาครับ ซึ่งในภาคส่งจะทำการเชื่อมตัวแปรระบบ (state variable) y ไปยังภาครับเพื่อเป็นสัญญาณในการใช้ถอดรหัส โดยสามารถแสดงบนโปรแกรม AnadigmDesigner2 ได้ดังรูปที่ 3.7

ก) ข)

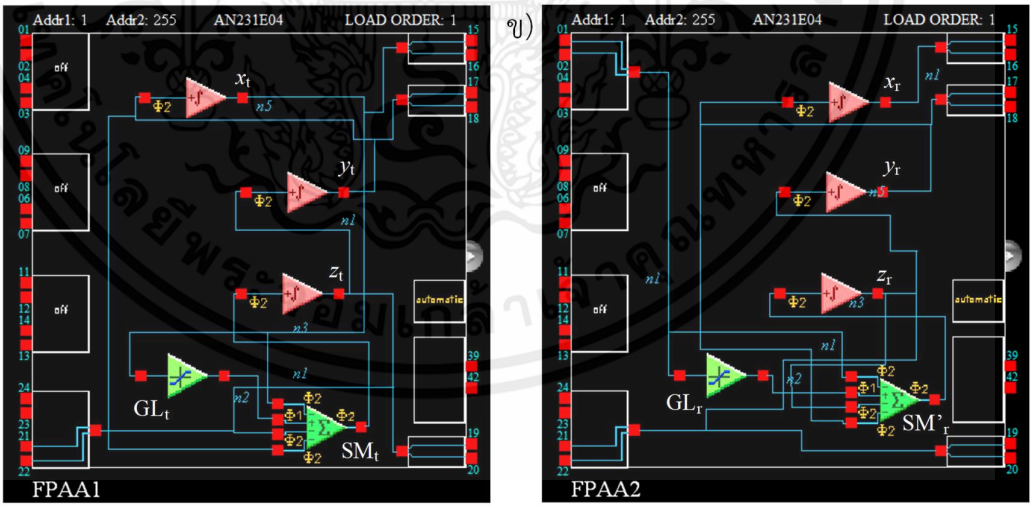


รูปที่ 3.7 Lorenz system ที่ออกแบบบน FPAA ; ก) ภาคส่ง ข) ภาครับ

3.1.7 การออกแบบ Synchronization ของ Jerk system บน FPAA

ในการ Synchronization ของ Jerk system บน FPAA จะทำการสร้างระบบ Jerk system บน FPAA ดังที่แสดงไปในรายงานความคืบหน้าก่อนหน้า โดยจะสร้างเหมือนกันทั้งภาคส่งและภาครับ ซึ่งในภาคส่งจะทำการเชื่อมตัวแปรระบบ (state variable) x ไปยังภาครับเพื่อเป็นสัญญาณในการใช้ถอดรหัส โดยสามารถแสดงบนโปรแกรม AnadigmDesigner2 ได้ดังรูปที่ 3.8

ก) ข)



รูปที่ 3.8 Jerk system ที่ออกแบบบน FPAA ; ก) ภาคส่ง ข) ภาครับ

3.1.8 เขียนโปรแกรมการแปลงรูปภาพเป็นข้อมูล Text เลขฐาน 16

ในปริญญานิพนธ์นี้จะใช้การส่งข้อมูลรูปภาพในการทดสอบกระบวนการสื่อสารรวมของระบบ สำหรับการแปลงข้อมูลรูปภาพเพื่อให้รูปแบบของเลขฐาน 16 เพื่อให้สามารถส่งผ่าน

ไมโครคอนโทรลเลอร์ได้นั้นจะใช้ไลบรารี่ของ ภาษา Python โดยจะใช้ Python Imaging Library (PIL) ในการเปิดไฟล์รูปภาพ และใช้ไลบรารี่ NumPy ในการแปลงรูปภาพเป็นข้อมูล Array ในรูปเลขฐาน 16 ซึ่งจะทำผ่าน IDE คือ Visual Code Studio โดยจะใช้ IPython Notebook (ipynb) เพื่อให้โค้ดสามารถทำงานได้ทีละเซลล์ โดยขั้นตอนในการเขียนโปรแกรมแปลงรูปภาพเป็นข้อมูล Text เลขฐาน 16 ทำได้ดังนี้

1. ทำการอ่านภาพโดยใช้ method open ของ class Image ใน PIL แสดงได้ดังรูปที่ 3.9

```
import PIL
from PIL import Image
import matplotlib.pyplot as plt
import csv
image=Image.open('tulips814x512.jpg')
print(image.format)
print(image.size)
print(image.mode)
plt.imshow(image)
```

รูปที่ 3.9 โค้ดการอ่านรูปภาพ โดยใช้ method open ของ open ของ class Image ใน PIL

2. ทำการแปลงข้อมูลรูปภาพเป็น ข้อมูล Array ด้วยฟังก์ชัน asarray และจัดให้อยู่ในรูปแบบเมทริกซ์ขนาดที่ต้องการเพื่อเป็นการจัดเฟรมในการส่งข้อมูล ซึ่งแสดงได้ดังรูปที่ 3.10

```
import numpy  as np
from numpy import asarray
data_image = asarray(image)
print(data_image)
data_reshape = data_image.reshape([22,1,60162])
print(type(data_reshape))
print(data_reshape.shape)
print(data_reshape)
```

รูปที่ 3.10 โค้ดการแปลงรูปภาพเป็นข้อมูล Array

3. ทำการแปลงจากข้อมูล Array อยู่ในรูป 0x (hexadecimal) และบันทึกเป็นไฟล์ text เพื่อให้สามารถนำไปเขียนข้อมูลลงไมโครคอนโทรลเลอร์ซึ่งจะใช้ ESP32 และเขียนด้วย ArduinoIDE ซึ่งสามารถแสดงโค้ดการบันทึกได้ดังรูปที่ 3.11

```python
from numpy import uint8
x=[]
data_test=[]
for i in range (0,22): #11byte for array in arduino
    tohex=data_reshape[i].reshape([-1,])
    print(tohex)
    print(tohex.shape)
    hex_array=[hex(x) for x in tohex]
    hex_array=np.array(hex_array)
    print(hex_array)
    data_test.append(hex_array)
with open('BIT0', 'w') as f:
        write=csv.writer(f)
        write.writerow(data_test[0])
with open('BIT1', 'w') as f:
        write=csv.writer(f)
        write.writerow(data_test[1])
with open('BIT2', 'w') as f:
        write=csv.writer(f)
        write.writerow(data_test[2])
```

รูปที่ 3.11 โค้ดการแปลงข้อมูล Array อยู่ในรูป 0x (hexadecimal) และบันทึกเป็นไฟล์ text

3.1.9 การเขียนโปรแกรมควบคุมการส่งข้อมูลด้วย ESP32 โดยใช้ Arduino IDE

จากหัวข้อก่อนหน้าในภาคส่งการส่งข้อมูลจะใช้ข้อมูลรูปภาพจากหัวข้อก่อนหน้าส่งผ่านช่องสื่อสารแบบอนุกรมด้วย ESP32 ในการควบคุมการส่งโดยจะทำการเขียนโปรแกรมบน Arduino IDE ซึ่งข้อมูลที่ใช้จะเป็นข้อมูลรูปภาพที่อยู่ในรูป 0x (hexadecimal) ส่งด้วย Baud rate เท่ากับ 9600 bits/sec ตามรูป 3.12

```
Serial.begin(115200);
Serial2.begin(9600, SERIAL_8N1, RXp2, TXp2);
pinMode(19, INPUT);
check = digitalRead(19);
while (checkloop == 1) {
  check = digitalRead(19);
  Serial.println("reading");
  if (check == HIGH)
  {
    Serial.println("Going");
    checkloop = 0;
  }
}

for (int k = 0; k < 60162; k++) {
  Serial2.write(EncodeSet0[k]);
  if (k % 407 == 0) {
    delay(110);
  }
}
delay(100);
for (int k = 0; k < 60162; k++) {
  Serial2.write(EncodeSet1[k]);
  if (k % 407== 0) {
    delay(110);
  }
}
delay(1000);
for (int k = 0; k < 60162; k++) {
  Serial2.write(EncodeSet2[k]);
  if (k % 407 == 0) {
    delay(110);
  }
}
delay(100);
```

รูปที่ 3.12 โค้ดการส่งข้อมูลผ่านช่องอนุกรมของ ESP32

จากรูปที่ 3.12 จะเห็นได้ว่า ตัวแปร EncodeSet0 จะทำการบันทึกข้อมูลรูปภาพที่ถูกแปลงอยู่ในรูป 0x (hexadecimal) ซึ่งถูกแบ่งไว้ 22 เฟรมดังที่กล่าวไว้ในหัวข้อก่อนหน้า และจะเก็บไว้ในตัวแปร EncodeSet0 – EncodeSet21 จากนั้นจะใช้คำสั่ง Serial2.write() ในการส่งข้อมูลแต่ละเฟรมผ่านช่องสื่อสารอนุกรม

3.1.10 การเขียนโปรแกรมควบคุมการส่งข้อมูลด้วย ESP32 โดยใช้ Arduino IDE และการเขียนโปรแกรมสร้างรูปภาพจากข้อมูลที่ได้รับด้วย Python

ในภาครับจะใช้ ESP32 ในการรับข้อมูลที่ทำการถอดรหัสแล้วโดยทำการเขียนโปรแกรมรับข้อมูลผ่านทางช่องสื่อสารอนุกรมโดยใช้คำสั่ง Serial2.read() และทำการแสดงข้อมูลบนหน้า Serial monitor ด้วยคำสั่ง Serial.println()

```
#define RXp2 16
#define TXp2 17
void setup() {
  Serial.begin(115200);
  Serial2.begin(9600, SERIAL_8N1, RXp2, TXp2);
}
void loop() {

  if ((Serial2.available()) > 0 ) {
    Serial.println(Serial2.read(), DEC);
  }
}
```

รูปที่ 3.13 โค้ดการส่งข้อมูลผ่านช่องอนุกรมของ ESP32

โดยในการเขียนโปรแกรมรับข้อมูลจะทำการอ่านข้อมูลจากช่องการสื่อสารอนุกรมของ ESP32 ด้วยไลบารี่ pySerial ของภาษา Python ด้วย Visual Code Studio แสดงได้ดังรูปที่ 3.14

```
if p==1:
    store_data.append("d")
    p=0
while line <= samples:
    getData=ser.readline()
    dataString = getData.decode('utf-8')
    data=dataString[0:][:-2]
    print(data)

    readings = data.split(",")
    # print(readings)

    store_data.append(readings)
    # print(sensor_data)
    line = line+1
```

รูปที่ 3.14 โค้ดการอ่านข้อมูลจากช่องสื่อสารอนุกรมด้วยภาษา Python

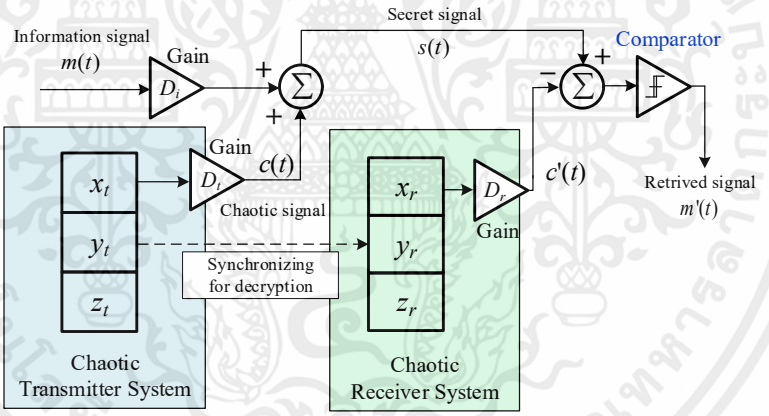
จากรูปที่ 3.14 จะทำการเก็บข้อมูลที่อ่านได้ลงในตัวแปร store_data จากนั้นทำเขียนโปรแกรมเพื่อนำข้อมูลที่ได้สร้างรูปภาพกลับคืนโดนการใช้ไลบารี่ PIL แสดงได้ดังรูป 3.15

```
df = pd.read_csv('10datam.csv')
df=df.values.flatten()
rxdata=df
print("Read Data: ",df)
#reshape
print("Reshaped Data:")
data3D_recover=df.reshape([542,814,3]) #reshape in to original shape
print(data3D_recover)
PIL_image_recover = Image.fromarray(data3D_recover.astype('uint8'), 'RGB')
plt.imshow(PIL_image_recover) #read image
```

รูปที่ 3.15 โค้ดการสร้างรูปภาพจากข้อมูลที่รับได้ด้วยภาษา Python

3.1.11 การออกแบบระบบการสื่อสารรวมทั้งภาคส่ง และภาครับบน Protoboard

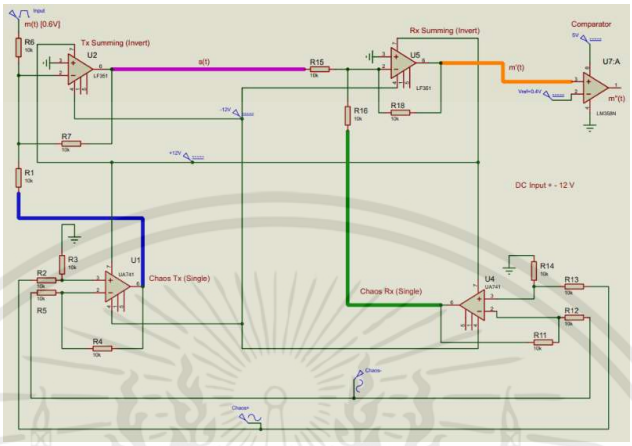
จากการศึกษากระบวนการ Chaotic masking communication [1] ได้ทำการออกแบบระบบการสื่อสารรวมของระบบเป็นบล็อกไดอะแกรมดังรูปที่ 3.16



รูปที่ 3.16 บล็อกไดอะแกรมระบบการสื่อสารโดยรวมที่ได้ออกแบบ

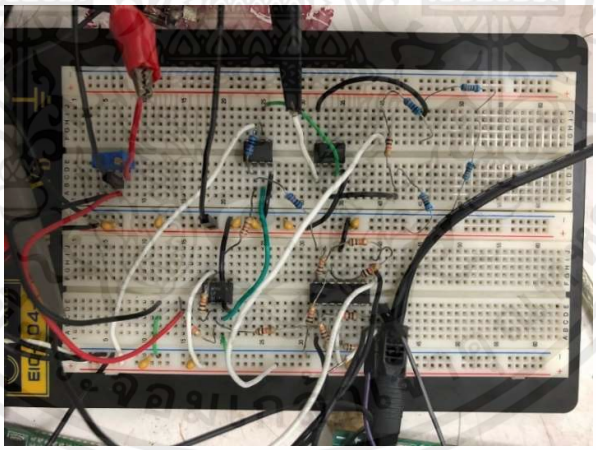
ภาพรวมของระบบทั้งหมดดังแสดงในรูปที่ 3.16 จะเริ่มจากภาคส่งทำการนำสัญญาณข้อมูลซึ่งเป็นข้อมูลรูปภาพ $m(t)$ ที่มีอัตราขยาย D_i รวมเข้ากับสัญญาณอลวน $c(t)$ ที่มีอัตราขยาย D_t สัญญาณที่ได้จากการรวมกันนั้นเรียกว่าสัญญาณลับ $s(t)$ ซึ่งถูกส่งผ่านสายสัญญาณไปยังภาครับ โดยทางภาครับจะมีการสร้างสัญญาณอลวน $c'(t)$ ที่มีลักษณะเหมือนกันกับสัญญาณอลวนของภาคส่งผ่านการ Synchronize ของตัวแปรสถานะของระบบ และทำการขยายตัวสัญญาณด้วยอัตราขยาย D_r จากนั้นจึงนำสัญญาณลับมาลบกับสัญญาณอลวนของภาครับเพื่อกู้สัญญาณข้อมูลกลับคืนมา แต่เนื่องจากสัญญาณที่ถูกกู้คืนมานั้นมีแอมพลิจูดที่แตกต่างกันกับสัญญาณเดิม จึงมีการนำวงจรเปรียบเทียบแรงดันหรือ Voltage Comparator มาใช้เพื่อทำให้แอมพลิจูดของสัญญาณที่กู้กลับมามีลักษณะเหมือนกันกับสัญญาณข้อมูลเดิม

เมื่อได้บล็อกไดอะแกรมโดยรวมทั้งภาคส่ง และภาครับแล้วสามารถออกแบบวงจรรวมของทั้งระบบภาคส่ง และภาครับได้ด้วยโปรแกรม Proteus โดยแสดงได้ดังรูปที่ 3.17

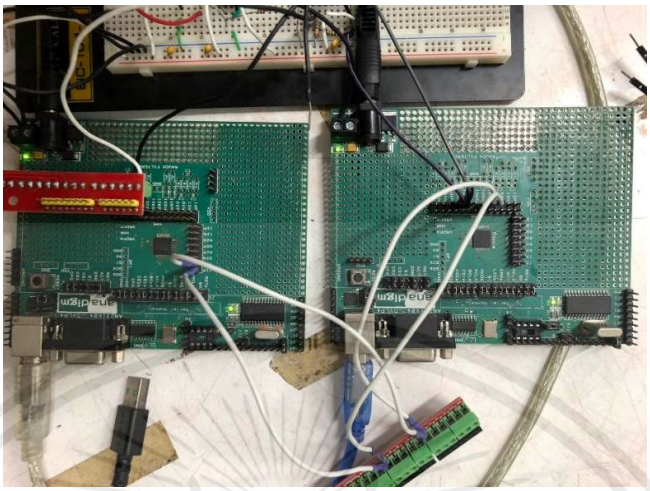


รูปที่ 3.17 วงจรรวมของทั้งระบบภาคส่ง และภาครับได้บนโปรแกรม Proteus

ทำการต่อวงจรจริงลงโปรโตบอร์ด โดยใช้ไฟเลี้ยงออปแอมป์วงจร Summing Amplifier ที่ ±12 V และไฟเลี้ยงของ Comparator ที่ +5 V และทำการต่อกับบอร์ด FPAA โดยที่ระบบอลวนบน FPAA มีการ Synchronize ระหว่างภาคส่ง และภาครับ สามารถแสดงได้ดังรูปที่ 3.18 และ 3.19



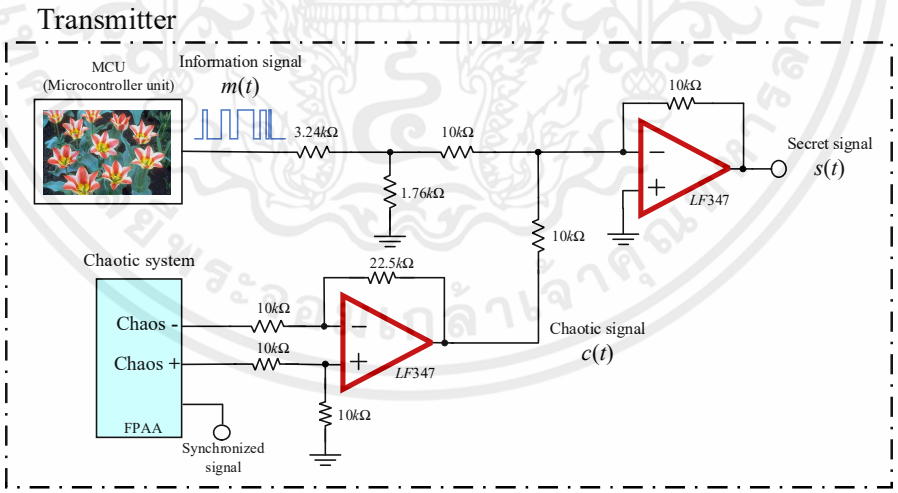
รูปที่ 3.18 การต่อวงจรทดสอบระบบการสื่อสารรวมทั้งระบบบนโปรโตบอร์ด



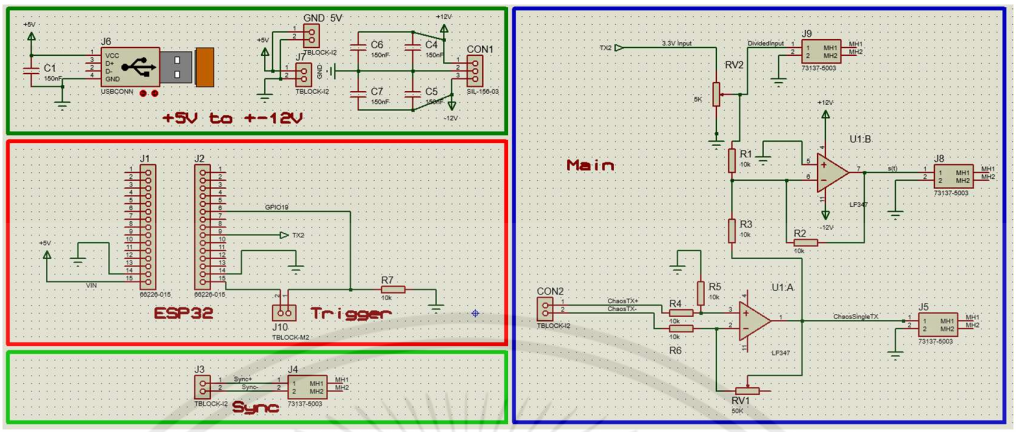
รูปที่ 3.19 การต่อวงจรระบบการสื่อสารรวมทั้งระบบกับบอร์ด FPAA

3.1.12 การออกแบบ PCB สำหรับวงจรภาคส่ง

เมื่อผลของระบบที่ออกแบบในหัวข้อก่อนหน้าตรงตามที่ศึกษาจึงทำการออกแบบ PCB โดยในการออกแบบ PCB ของวงจรภาคส่ง จำเป็นจะต้องออกแบบ Schematic ของวงจรก่อน ซึ่งประกอบไปด้วยวงจร Differential to Single-ended Output, วงจรขยายสัญญาณแบบรวมสัญญาณ , ESP32 และช่องสัญญาณสำหรับการ synchronize โดย Schematic สำหรับวงจรภาคส่ง และ Schematic สำหรับวงจรภาคส่งในโปรแกรม Proteus แสดงได้ดังรูป 3.20 และ 3.21 ตามลำดับ

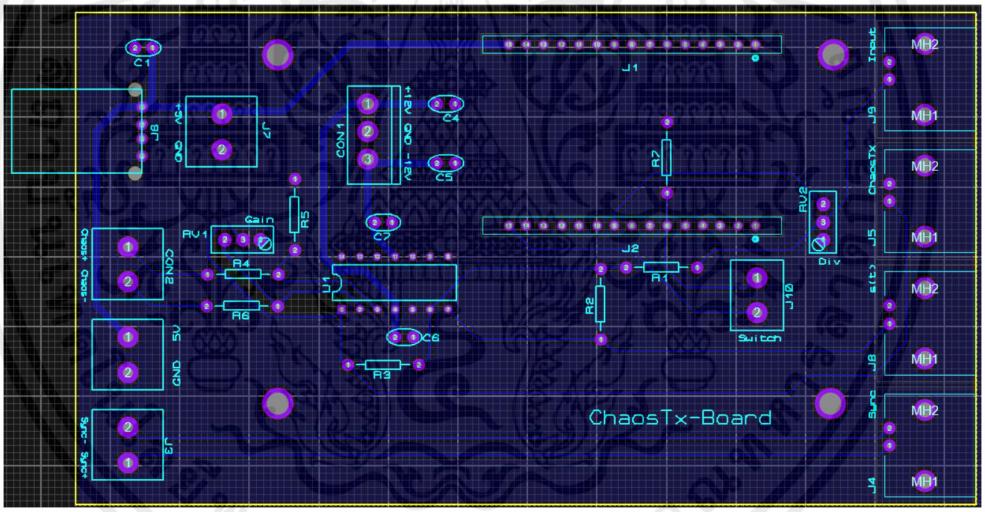


รูปที่ 3.20 Schematic วงจรภาคส่ง



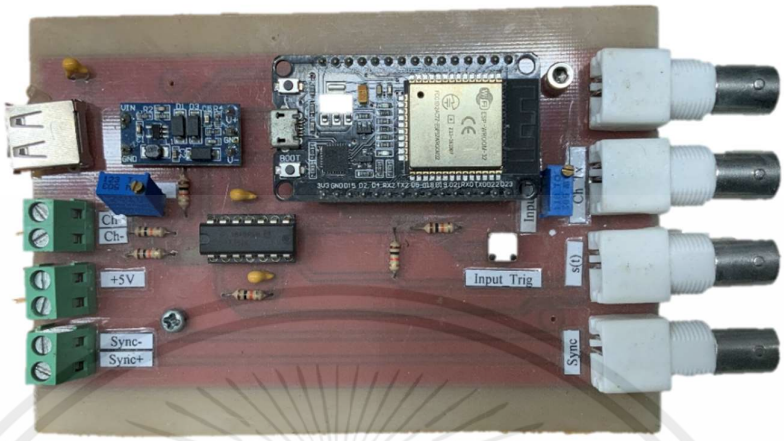
รูปที่ 3.21 Schematic วงจรภาคส่งในโปรแกรม Proteus

หลังจากนั้น ทำการออกแบบ PCB ในโปรแกรม Proteus โดยจัดวางตำแหน่งของอุปกรณ์ให้เหมาะสมกับการเชื่อมต่อ และทำการเดินลายทองแดงแบบ 1 Layer โดยแสดงดังรูป 3.22



รูปที่ 3.22 PCB Layout สำหรับวงจรภาคส่ง

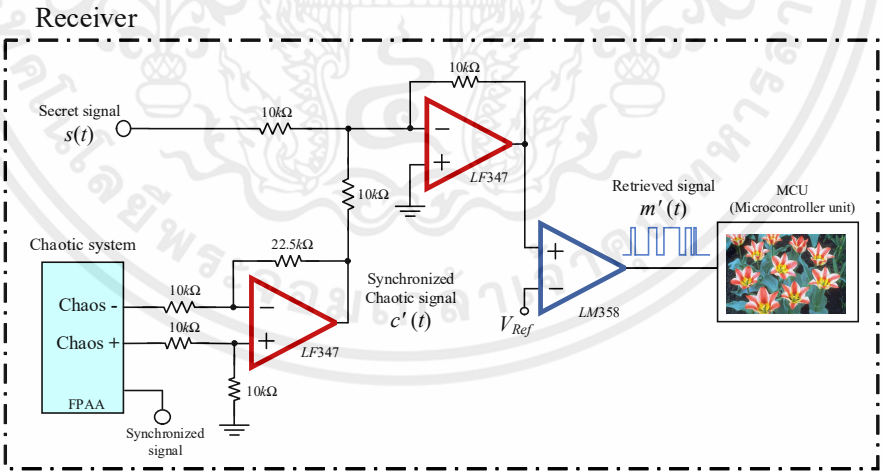
เมื่อทำการออกแบบ PCB เรียบร้อยแล้ว จากนั้นทำการกัดปริ้นลงบนแผ่นทองแดง แล้วทำการเจาะรูใส่อุปกรณ์ และบัดกรีอุปกรณ์ให้เรียบร้อยจะได้วงจรภาคส่งดังรูปที่ 3.23



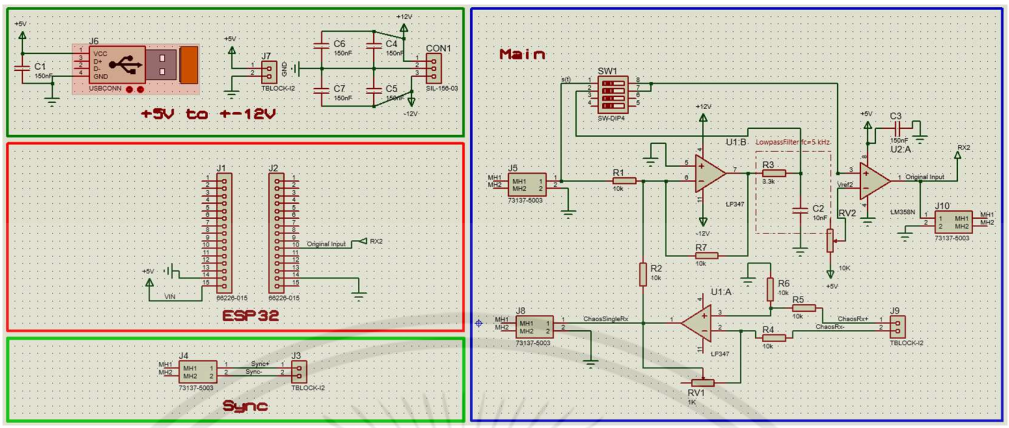
รูปที่ 3.23 แผ่น PCB สำหรับวงจรภาคส่ง

3.1.13 การออกแบบ PCB สำหรับวงจรภาครับ

ในการออกแบบ PCB ของวงจรภาครับ จำเป็นจะต้องออกแบบ Schematic ของวงจรภาครับก่อน ซึ่งประกอบไปด้วยวงจร Differential to Single-ended Output, วงจรขยายสัญญาณแบบรวมสัญญาณ, วงจรเปรียบเทียบแรงดัน, MCU และช่องสัญญาณสำหรับการ synchronize โดย Schematic สำหรับวงจรภาครับ และ Schematic สำหรับวงจรภาครับในโปรแกรม Proteus แสดงได้ดังรูปที่ 3.24 และ 3.25 ตามลำดับ

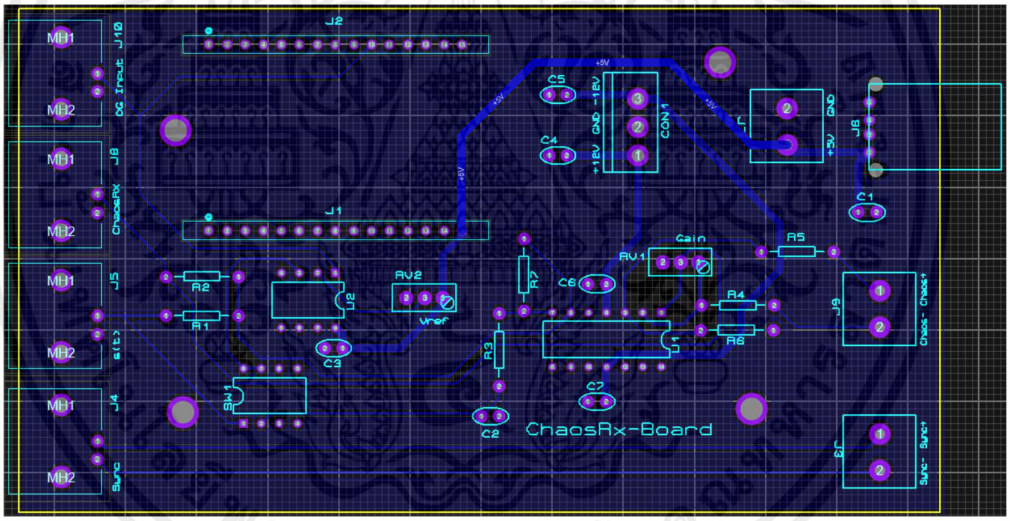


รูปที่ 3.24 Schematic วงจรภาครับ



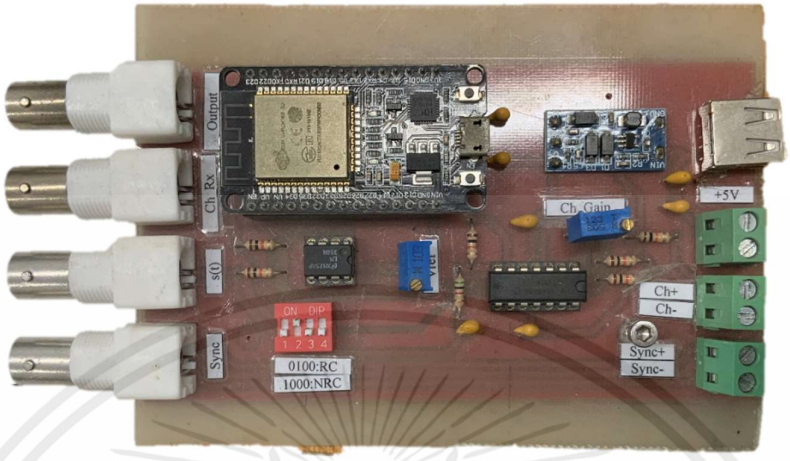
รูปที่ 3.25 Schematic วงจรภาครับในโปรแกรม Proteus

หลังจากนั้น ทำการออกแบบ PCB ในโปรแกรม Proteus โดยจัดวางตำแหน่งของอุปกรณ์ให้เหมาะสมกับการเชื่อมต่อ และทำการเดินลายทองแดงแบบ 1 Layer โดยแสดงดังรูป 3.26



รูปที่ 3.26 PCB Layout สำหรับวงจรภาครับ

เมื่อทำการออกแบบ PCB เรียบร้อยแล้ว จากนั้นทำการกัดปริ้นลงบนแผ่นทองแดง แล้วทำการเจาะรูใส่อุปกรณ์ และบัดกรีอุปกรณ์ให้เรียบร้อย จะได้วงจรภาครับดังรูปที่ 3.27



รูปที่ 3.27 แผ่น PCB สำหรับวงจรภาครับ

3.2 เครื่องมือที่ใช้ในการทดลอง

ในปริญญานิพนธ์นี้ มีอุปกรณ์และเครื่องมือที่ใช้ในการทดลองดังนี้

3.2.1 AnadigmDesigner2

3.2.2 AN231E04	2	เครื่อง
3.2.3 Oscilloscope	1	เครื่อง
3.2.4 LF347	2	ตัว
3.2.5 LM358	1	ตัว
3.2.6 ตัวต้านทาน 10 กิโลโอห์ม	13	ตัว
3.2.7 ตัวต้านทาน 3.3 กิโลโอห์ม	1	ตัว
3.2.8 ตัวต้านทานปรับค่าได้ 50 กิโลโอห์ม	2	ตัว
3.2.9 ตัวต้านทานปรับค่าได้ 10 กิโลโอห์ม	1	ตัว
3.2.10 ตัวต้านทานปรับค่าได้ 5 กิโลโอห์ม	1	ตัว
3.2.11 ตัวเก็บประจุ 150 นาโนฟารัด	11	ตัว
3.2.12 ตัวเก็บประจุ 10 นาโนฟารัด	1	ตัว
3.2.13 5V to ±12V DC-DC Converter	2	ตัว
3.2.14 บล็อกเทอร์มินัล 2 ขา 2.54 มม.	6	ตัว
3.2.15 BNC แจ็คตัวเมีย	8	ตัว
3.2.16 ดิปสวิตช์ 4 ทาง	1	ตัว
3.2.17 สวิตช์ 2 ขา	1	ตัว

3.2.18 USB type-A เพศเมีย	2	ตัว
3.2.19 IC socket 14 ขา	2	ตัว
3.2.20 IC socket 8 ขา	1	ตัว
3.2.21 บอร์ดพัฒนา ESP32	2	ตัว
3.2.22 แผ่นทองแดง 13x8 ซม.	2	แผ่น
3.2.23 แผ่นไม้ขนาด 25x17 ซม.	2	แผ่น

3.3 การจัดเก็บผลการทดลอง

3.3.1 ทดสอบวงจร Differential voltage to single mode ด้วย Proteus

ทำการทดสอบการทำงานของวงจร Differential voltage to single mode ว่า Output ที่ได้สอดคล้องกับ Input ที่เข้าไป และเป็นไปตามทฤษฎีที่ศึกษาเพื่อให้มีความเหมาะสมต่อ FPAA

3.3.2 ทดสอบการสร้างสัญญาณอลวน ด้วย FPAA

ทำการทดสอบการสร้าง Lorenz system และ Jerk system บน FPAA โดยการวัดรูปสัญญาณ (waveform) และ Attractor ที่ได้ว่าเป็นไปตามทฤษฎีที่ได้ศึกษา

3.3.3 ทดสอบกระบวนการ Chaotic masking communication โดยใช้ Simulink

ทำการทดสอบกระบวนการสื่อสารแบบ Chaotic masking communication ที่ใช้ Lorenz system และ Jerk system บน Simulink เพื่อดูผลการจำลองว่าสอดคล้องไปตามทฤษฎีที่ได้ทำการศึกษา

3.3.4 ทดสอบกระบวนการ Synchronization บน FPAA

ทำการทดสอบการ synchronize ของ Lorenz system และ Jerk system บน ภาคส่งและภาครับบน FPAA โดยวัดรูปสัญญาณ (waveform) ของระบบเทียบกันทั้งภาคส่ง และภาครับ

3.3.5 ทดสอบวงจรสำหรับระบบการสื่อสารรวมทั้งภาคส่ง และภาครับ

ทำการทดสอบวงจรสำหรับระบบการสื่อสารรวมทั้งภาคส่ง และภาครับที่ได้ทำการออกแบบว่า Output มีความสอดคล้องกับ Input ที่เข้าไป และเป็นไปตามทฤษฎีที่ศึกษา

3.3.6 ทดสอบการแปลงรูปภาพเป็นข้อมูล Text เลขฐาน 16

ทำการทดสอบแปลงข้อมูลรูปภาพที่จะใช้ในการทดสอบส่งข้อมูล และรับข้อมูลเป็นข้อมูล Text ในรูป 0x (hexadecimal) เพื่อให้สามารถส่งผ่านช่องการสื่อสารอนุกรมของไมโครคอนโทรลเลอร์

3.3.7 การทดสอบระบบการสื่อสารโดยรวมทั้งหมดบนแผ่น PCB โดยใช้ข้อมูลรูปภาพในการสื่อสาร

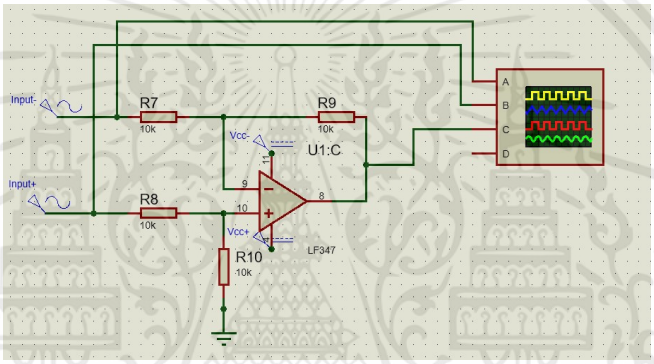
ทดสอบกระบวนการสื่อสารโดยรวมด้วยการส่งข้อมูลรูปภาพ และทำการเข้ารหัสถอดรหัสด้วยระบบบอลวน จากนั้นนำข้อมูลที่ได้สร้างรูปภาพกลับคืนมา และทำการทดสอบการเปลี่ยน Parameter ของระบบบอลวนที่ภาครับเพื่อดูความแตกต่างของรูปภาพที่ถอดรหัสได้

บทที่ 4

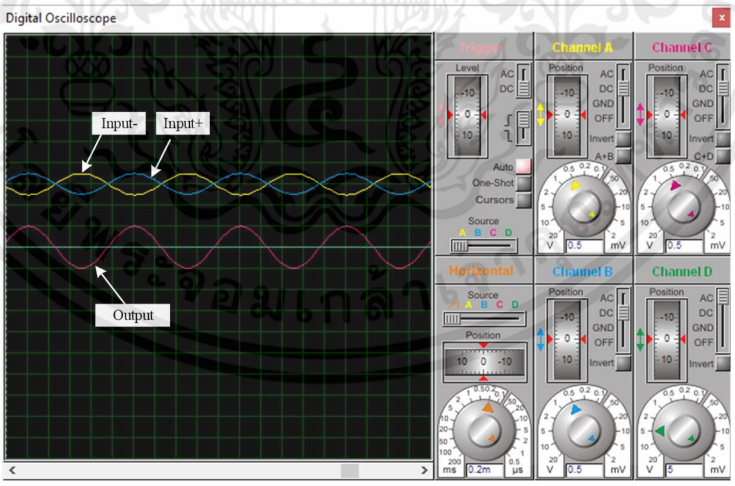
ผลการทดลอง

4.1 ทดสอบการใช้วงจร Differential voltage to single mode ด้วย Proteus

ในการทดสอบวงจร Differential voltage to single mode ในโปรแกรม Proteus ได้ออกแบบไว้ในรูปที่ 4.1 จะเห็นได้ว่าเมื่อป้อนสัญญาณ Input+ และสัญญาณ Input- ซึ่งเป็นสัญญาณแบบ Differential และยกระดับสัญญาณขึ้นไป +1.5 V เข้าวงจร Differential voltage to single mode จะได้สัญญาณ Output ในรูปแบบ Single mode ดังรูปที่ 4.2



รูปที่ 4.1 วงจร Differential voltage to single mode ในโปรแกรม Proteus



รูปที่ 4.2 Input/Output ของวงจร Differential voltage to single mode ในโปรแกรม Proteus

4.2 ทดสอบการสร้างสัญญาณอลวนด้วย FPAA

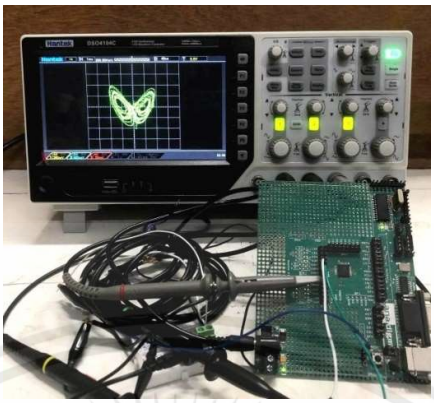
4.2.1 Lorenz system

ในขั้นตอนการทดสอบนี้จะเป็นการวัดสัญญาณอลวนของระบบ Lorenz system จาก โดยทำการอัพโหลดระบบที่ได้ออกแบบไว้ในโปรแกรม AnadigmDesigner2 จากบทที่แล้วเข้า FPAA โดยสามารถแสดงตาราง Options และค่า Parameters ต่างๆของ CAM ได้ดังตารางที่ 4.1

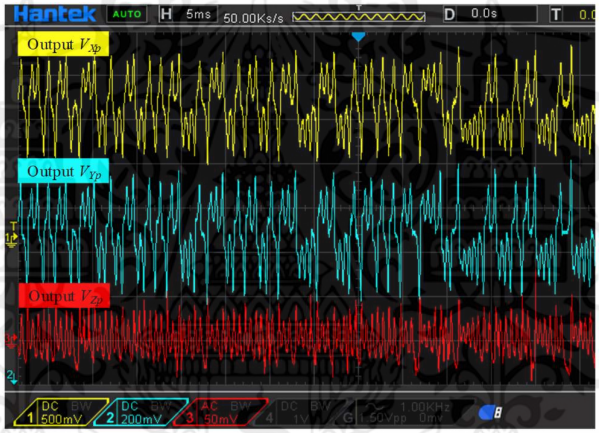
ตารางที่ 4.1 การปรับ Options และค่า Parameters ของ Lorenz system บน โปรแกรม AnadigmDesigner2

Name	Options	Parameters	Clocks
Multiplier1 (Multiplier v1.0.2)  *Anadigm (Approved)*	**Sample and Hold** *Off*	**Multiplication Factor** *1.00*	**ClockA** *250 kHz (Chip Clock 3)* **ClockB** *4 MHz (Chip Clock 0)*
Multiplier2 (Multiplier v1.0.2)  *Anadigm (Approved)*	**Sample and Hold** *Off*	**Multiplication Factor** *1.00*	**ClockA** *250 kHz (Chip Clock 3)* **ClockB** *4 MHz (Chip Clock 0)*
SumDiff1 (SumDiff v1.0.1)  *Anadigm (Approved)*	**Output Phase** *Phase 2* **Input 1** *Inverting* **Input 2** *Non-inverting* **Input 3** *Off* **Input 4** *Off*	**Gain 1 (UpperInput)** *5.30* **Gain 2 (LowerInput)** *15.1*	**ClockA** *250 kHz (Chip Clock 3)*
SumDiff2 (SumDiff v1.0.1)  *Anadigm (Approved)*	**Output Phase** *Phase 1* **Input 1** *Non-inverting* **Input 2** *Inverting* **Input 3** *Inverting* **Input 4** *Off*	**Gain 1 (UpperInput)** *5.80* **Gain 2 (MiddleInput)** *0.200* **Gain 3 (LowerInput)** *42.4*	**ClockA** *250 kHz (Chip Clock 3)*
SumDiff3 (SumDiff v1.0.1)  *Anadigm (Approved)*	**Output Phase** *Phase 2* **Input 1** *Non-inverting* **Input 2** *Inverting* **Input 3** *Off* **Input 4** *Off*	**Gain 1 (UpperInput)** *0.950* **Gain 2 (LowerInput)** *1.55*	**ClockA** *250 kHz (Chip Clock 3)*
Integrator1 (Integrator v1.1.1)  *Anadigm (Approved)*	**Polarity** *Non-inverting* **Input Sampling Phase** *Phase 1* **Compare Control To** *No Reset*	**Integration Const. [1/us]** *0.00250*	**ClockA** *250 kHz (Chip Clock 3)*
Integrator2 (Integrator v1.1.1)  *Anadigm (Approved)*	**Polarity** *Non-inverting* **Input Sampling Phase** *Phase 2* **Compare Control To** *No Reset*	**Integration Const. [1/us]** *0.00250*	**ClockA** *250 kHz (Chip Clock 3)*
Integrator3 (Integrator v1.1.1)  *Anadigm (Approved)*	**Polarity** *Non-inverting* **Input Sampling Phase** *Phase 2* **Compare Control To** *No Reset*	**Integration Const. [1/us]** *0.00250*	**ClockA** *250 kHz (Chip Clock 3)*

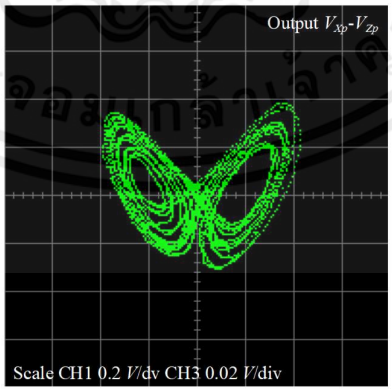
และทำการวัด Output ด้วย Oscilloscope ที่ขา 15(V_{Yp}), 17(V_{Xp}), 19(V_{Zp}) ดังรูปที่ 4.3 บน FPAA ในรูปแบบ Differential Voltage โดยรูปสัญญาณ (waveform) และ Attractor แสดงได้ดังรูปที่ 4.4 และ 4.5 ตามลำดับ



รูปที่ 4.3 การวัดรูปแบบสัญญาณ และ Attractor ของ Lorenz system บน FPAA ด้วย Oscilloscope



รูปที่ 4.4 รูปแบบสัญญาณ (waveform) ของ Lorenz system V_{Xp}, V_{Yp}, V_{Zp} ของตัวแปรระบบ x, y และ z ตามลำดับ

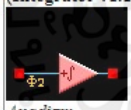


รูปที่ 4.5 Attractor ของ Lorenz system บนระนาบ x-z ที่วัดได้

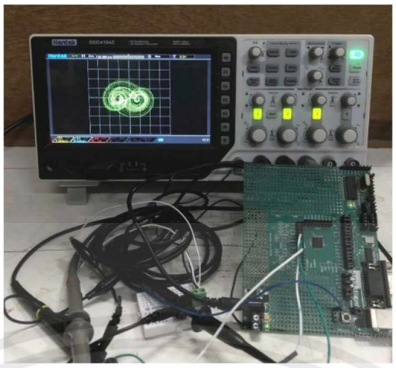
4.2.2 Jerk system

ในขั้นตอนการทดสอบนี้จะเป็นการวัดสัญญาณอลวนของระบบ jerk system จาก โดยทำการอัพโหลดระบบที่ได้ออกแบบไว้ในโปรแกรม AnadigmDesigner2 จากบทที่แล้วเข้า FPAA โดยสามารถแสดงตาราง Options และค่า Parameters ต่างๆของ CAM ได้ดังตารางที่ 4.2

ตารางที่ 4.2 การปรับ Options และค่า Parameters ของ jerk system บน โปรแกรม AnadigmDesigner2

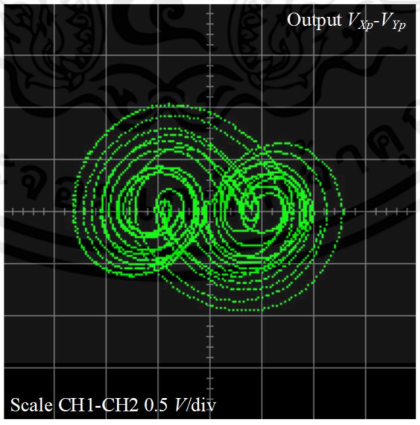
Name	Options	Parameters	Clocks
Integrator1 (Integrator v1.1.1)  *Anadigm (Approved)*	**Polarity** *Non-inverting* **Input Sampling Phase** *Phase 2* **Compare Control To** *No Reset*	**Integration Const. [1/us]** *0.00250*	**ClockA** *250 kHz (Chip Clock 3)*
Integrator2 (Integrator v1.1.1)  *Anadigm (Approved)*	**Polarity** *Non-inverting* **Input Sampling Phase** *Phase 2* **Compare Control To** *No Reset*	**Integration Const. [1/us]** *0.00250*	**ClockA** *250 kHz (Chip Clock 3)*
Integrator3 (Integrator v1.1.1)  *Anadigm (Approved)*	**Polarity** *Non-inverting* **Input Sampling Phase** *Phase 2* **Compare Control To** *No Reset*	**Integration Const. [1/us]** *0.00250*	**ClockA** *250 kHz (Chip Clock 3)*
SumDiff1 (SumDiff v1.0.1)  *Anadigm (Approved)*	**Output Phase** *Phase 2* **Input 1** *Inverting* **Input 2** *Non-inverting* **Input 3** *Inverting* **Input 4** *Inverting*	**Gain 1 (UpperInput)** *1.00* **Gain 2 (SecondInput)** *1.00* **Gain 3 (ThirdInput)** *0.550* **Gain 4 (LowerInput)** *1.00*	**ClockA** *250 kHz (Chip Clock 3)*
GainLimiter1 (GainLimiter v1.0.1)  *Anadigm (Approved*)*		**Gain** *20.0* **Output Voltage Limit** *1.00*	**ClockA** *250 kHz (Chip Clock 3)*

และทำการวัด Output ด้วย Oscilloscope ที่ขา 15(V_{Yp}), 17(V_{Xp}), 19(V_{Zp}) ดังรูปที่ 4.6 บน FPAA ในรูปแบบ Differential Voltage โดยรูปสัญญาณ (Waveform) และ Attractor แสดงได้ดังรูปที่ 4.7 และ 4.8 ตามลำดับ



รูปที่ 4.6 การวัดรูปแบบสัญญาณ และ Attractor ของ jerk system บน FPAA ด้วย Oscilloscope

รูปที่ 4.7 รูปแบบสัญญาณ (waveform) ของ jerk system V_{Xp}, V_{Yp}, V_{Zp} ของตัวแปรระบบ x, y และ z ตามลำดับ

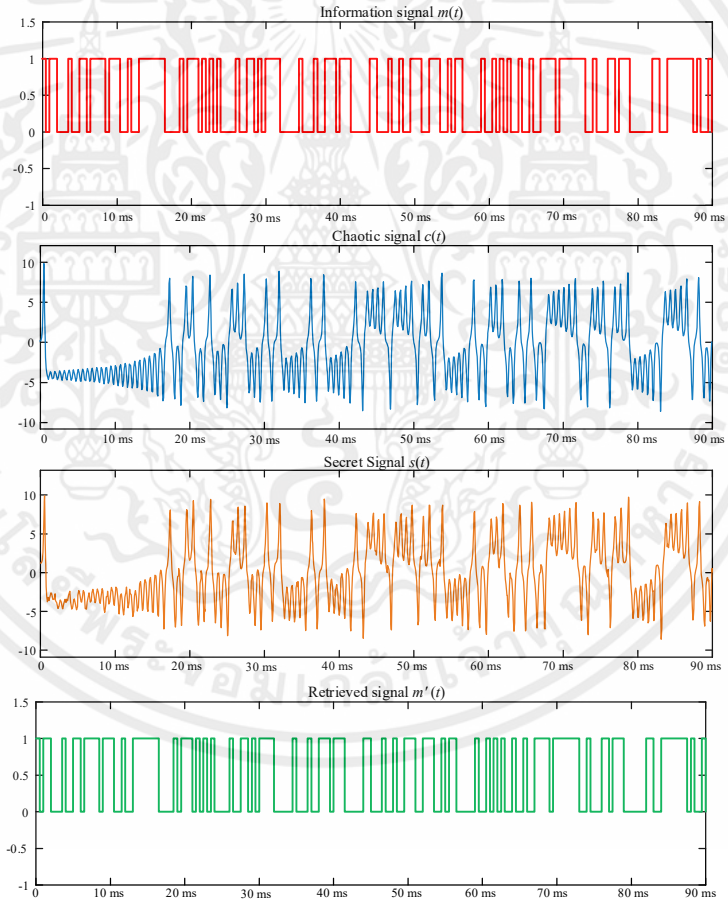


รูปที่ 4.8 Attractor ของ jerk system บนระนาบ x-y ที่วัดได้

4.3 ผลการทดสอบ Chaotic masking communication ด้วย Simulink

4.3.1 Lorenz system

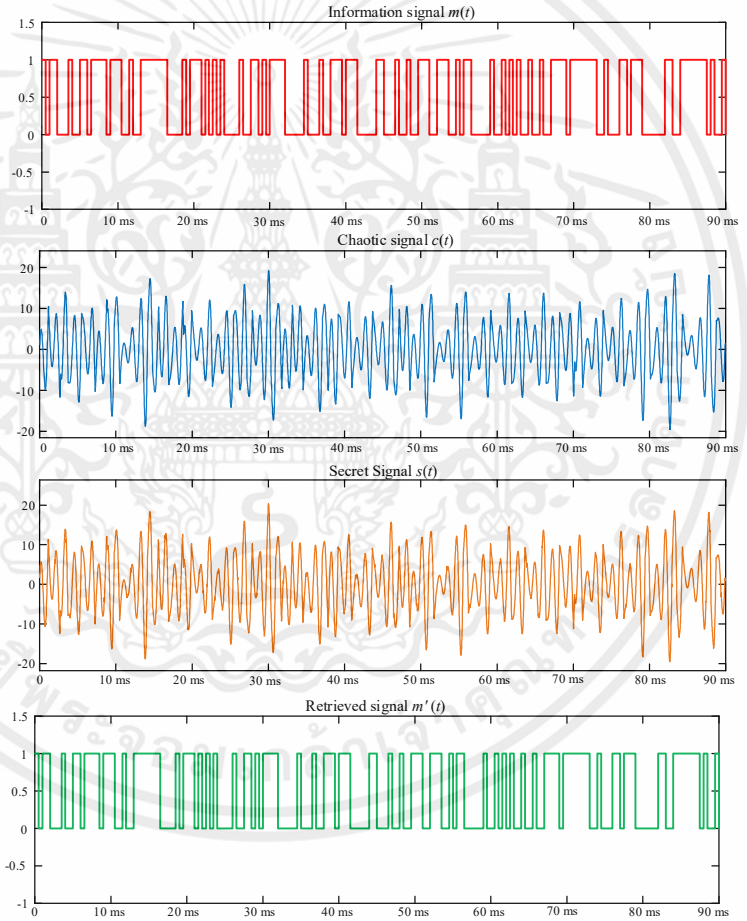
ผลการทดสอบ Chaotic masking communication ด้วย Lorenz system ผ่านโปรแกรม SIMULINK สามารถแสดงได้ดังรูปที่ 4.9 โดยสัญญาณข้อมูลหรือ $m(t)$ เป็นสัญญาณไบนารี 2000 bits/s แอมพลิจูด 1 V และสัญญาณอลวน (Chaotic signal) หรือ $c(t)$ มีแอมพลิจูดประมาณ 8 V ซึ่งถูกขยายด้วยวงจรขยายสัญญาณ $G_t = 0.5$ เพื่อให้มีขนาดใหญ่กว่าสัญญาณข้อมูลประมาณ 20 dB ทำการรวมเข้ากับสัญญาณข้อมูล $m(t)$ เป็นสัญญาณ Secret signal หรือ $s(t)$ หลังจาก Synchronization เสร็จสมบูรณ์สัญญาณที่ถูกกู้กลับมา (Retrieved signal) หรือ $m'(t)$ มีลักษณะตรงกับสัญญาณข้อมูลเดิมทุกประการ



รูปที่ 4.9 ผลการทดสอบ Chaotic masking communication ด้วย Lorenz system ที่ทำการออกแบบด้วย SIMULINK

4.3.2 Jerk system

ผลการทดสอบ Chaotic masking communication ด้วย jerk system ผ่านโปรแกรม Simulink สามารถแสดงได้ดังรูปที่ 4.10 โดยสัญญาณข้อมูลหรือ $m(t)$ เป็นสัญญาณไบนารี 2000 bits/s แอมพลิจูด 1 V และสัญญาณอลวน (Chaotic signal) หรือ $c(t)$ มีแอมพลิจูดประมาณ 10 V ซึ่งถูกขยายด้วยวงจรขยายสัญญาณ $G_t = 10$ เพื่อให้มีขนาดใหญ่กว่าสัญญาณข้อมูลประมาณ 20 dB ทำการรวมเข้ากับสัญญาณข้อมูล $m(t)$ เป็นสัญญาณ Secret signal หรือ $s(t)$ หลังจาก Synchronization เสร็จสมบูรณ์สัญญาณที่ถูกกู้กลับมา (Retrieved signal) หรือ $m'(t)$ มีลักษณะตรงกับสัญญาณข้อมูลเดิมทุกประการ



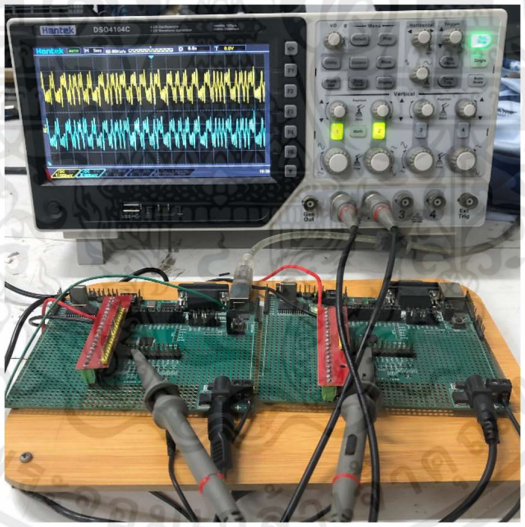
รูปที่ 4.10 ผลการทดสอบ Chaotic masking communication ด้วย jerk system ที่ทำการออกแบบด้วย SIMULINK

4.4 การทดสอบ Synchronize ของสัญญาณอลวน ระหว่างภาคส่ง และภาครับ

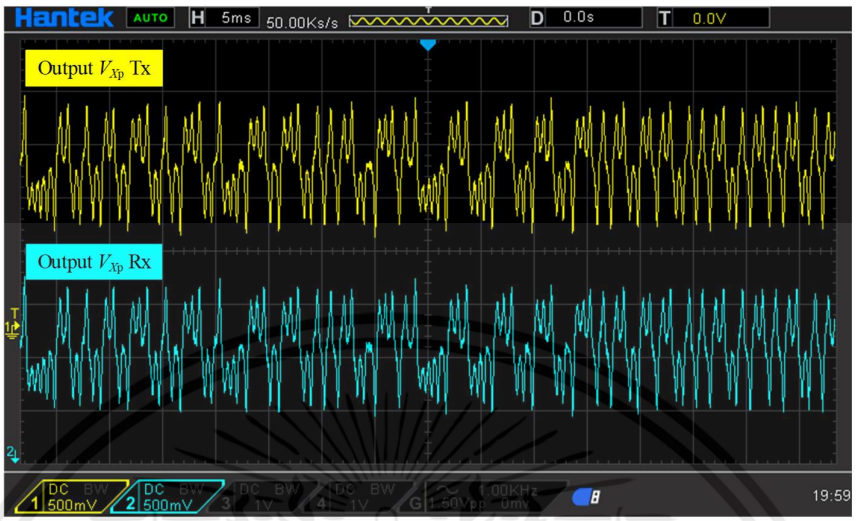
ในการครอบหน้ากากให้กับสัญญาณข้อมูล (Masking information signal) สัญญาณข้อมูลที่ภาคส่งถูกเข้ารหัสด้วยสัญญาณอลวน (Chaotic signal) และถูกส่งไปภาครับ ซึ่งภาครับสามารถสร้างสัญญาณอลวนที่มีลักษณะเหมือนกันกับสัญญาณอลวนที่ถูกสร้างที่ภาคส่งเพื่อถอดรหัส ดังนั้นเพื่อให้ถอดรหัสสำเร็จจะต้องมีการ Synchronization ของระบบอลวนระหว่างภาคส่ง และภาครับ ซึ่งผลลัพธ์จากการวัดผ่าน Oscilloscope จะแสดงได้ในหัวข้อ 4.4.1 สำหรับ Lorenz system และ 4.4.2 สำหรับ Jerk system

4.4.1 ผล Synchronization ของ Lorenz system บน FPAA

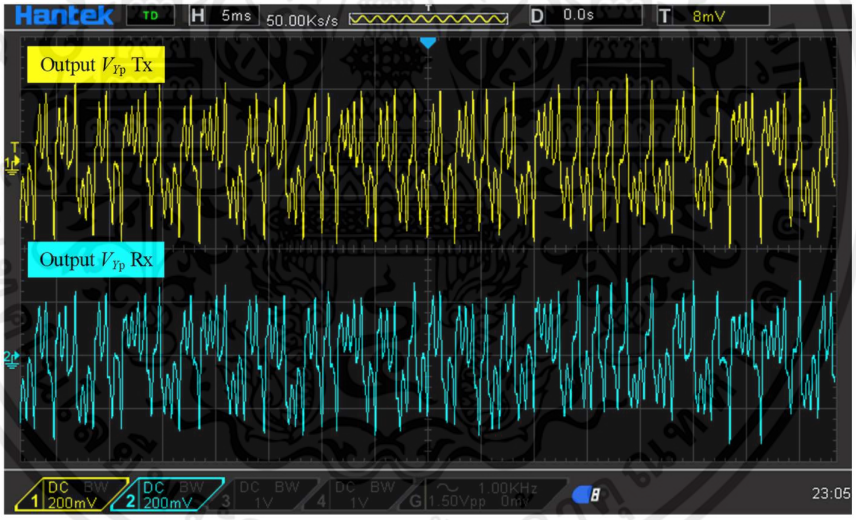
ในขั้นตอนการทดสอบนี้จะแสดงการวัดสัญญาณอลวนของ Lorenz system ที่ทำการ Synchronize แล้วโดยจะวัดสัญญาณของของตัวแปรระบบทั้ง x, y, และ z ของระบบบน FPAA ของทั้งภาคส่ง และภาครับ ดังรูปที่ 4.11 โดยใช้ Options และค่า Parameters ต่างๆของ CAM ได้ดังตารางที่ 4.1 ซึ่งสามารถแสดงผลการวัดได้ดังรูปที่ 4.12, รูปที่ 4.13 และรูปที่ 4.14 ตามลำดับ



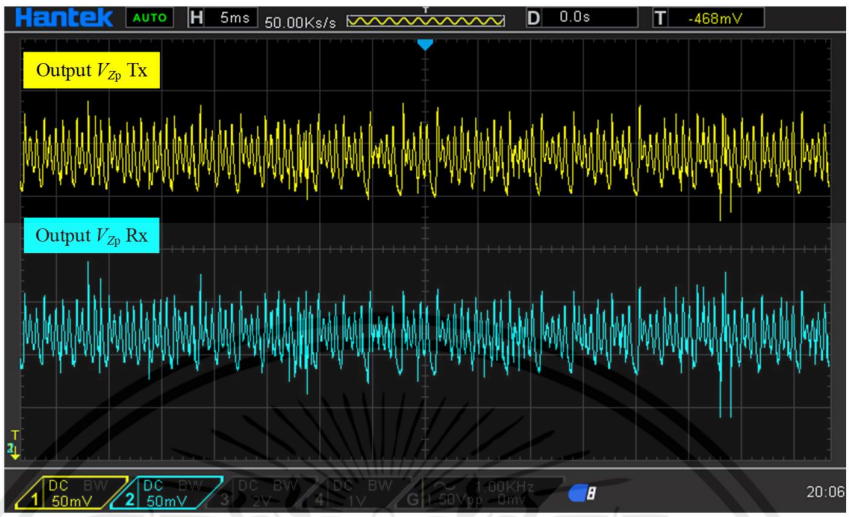
รูปที่ 4.11 การวัดรูปแบบสัญญาณ (waveform) ของ Lorenz system บน FPAA ทั้งภาคส่ง และภาครับ



รูปที่ 4.12 รูปแบบสัญญาณ (waveform) ภาคส่ง และภาครับของ Lorenz system V_{Xp} ของตัวแปรระบบ x



รูปที่ 4.13 รูปแบบสัญญาณ (waveform) ภาคส่ง และภาครับของ Lorenz system V_{Yp} ของตัวแปรระบบ y

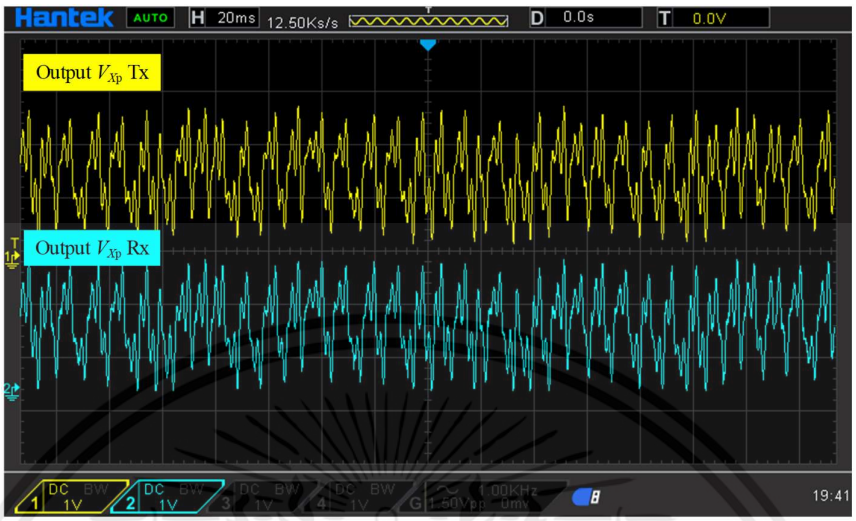


รูปที่ 4.14 รูปแบบสัญญาณ (waveform) ภาคส่ง และภาครับของ Lorenz system V_{Zp} ของตัวแปรระบบ z

4.4.2 ผล Synchronization ของ Jerk system บน FPAA

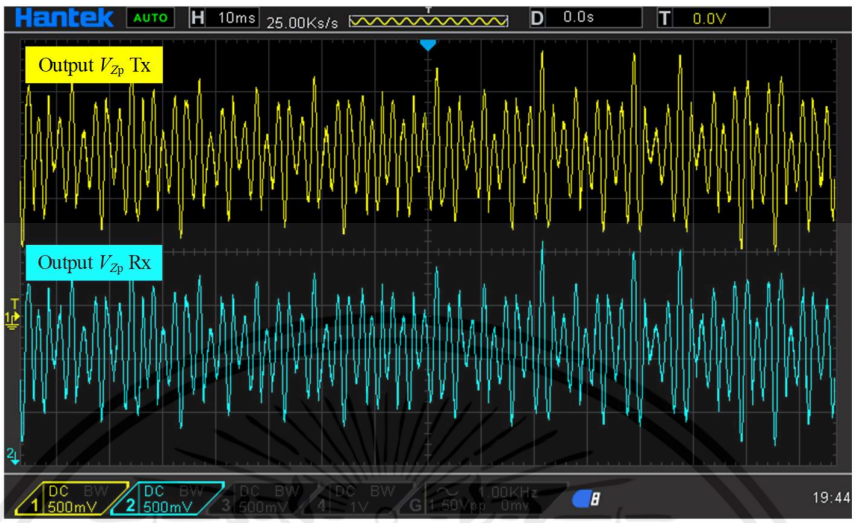
ในขั้นตอนการทดสอบนี้จะแสดงการวัดสัญญาณอลวนของ Lorenz system ที่ทำการ Synchronize แล้วโดยจะวัดสัญญาณของของตัวแปรระบบทั้ง x, y, และ z ของระบบบน FPAA ของทั้งภาคส่ง และภาครับ ดังรูปที่ 4.15 โดยใช้ Options และค่า Parameters ต่างๆของ CAM ได้ดังตารางที่ 4.2 ซึ่งสามารถแสดงผลการวัดได้ดังรูปที่ 4.16, รูปที่ 4.17 และรูปที่ 4.18 ตามลำดับ

รูปที่ 4.15 การวัดรูปแบบสัญญาณ (waveform) ของ Jerk system บน FPAA ทั้งภาคส่ง และภาครับ



รูปที่ 4.16 รูปแบบสัญญาณ (waveform) ภาคส่ง และภาครับของ Jerk system V_{Xp} ของตัวแปรระบบ x

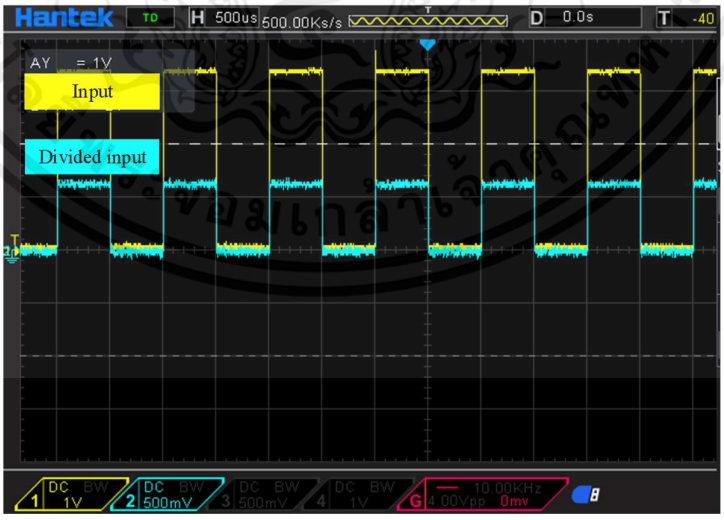
รูปที่ 4.17 รูปแบบสัญญาณ (waveform) ภาคส่ง และภาครับของ Jerk system V_{Yp} ของตัวแปรระบบ y



รูปที่ 4.18 รูปแบบสัญญาณ (waveform) ภาคส่ง และภาครับของ Jerk system V_{Zp} ของตัวแปรระบบ z

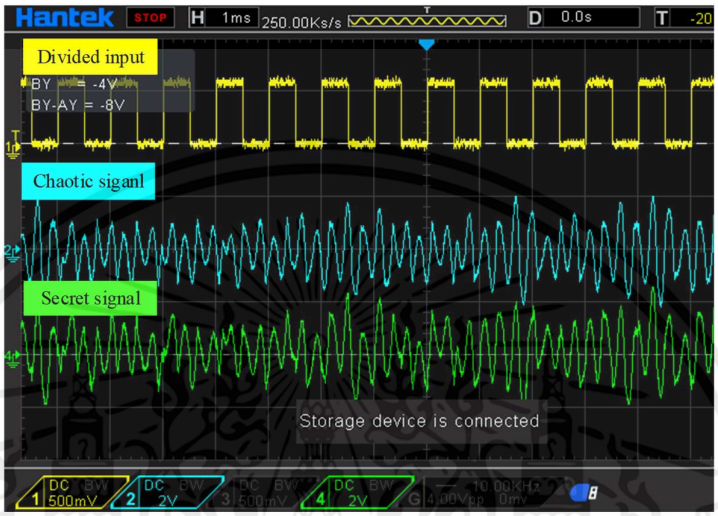
4.5 ผลทดสอบวงจรสำหรับระบบการสื่อสารรวมทั้งภาคส่ง และภาครับ

ในภาคส่งทำการทดสอบวงจรที่ต่อโดยการจ่ายอินพุตเป็นสัญญาณสี่เหลี่ยม (Square wave) ขนาด 3.3 V โดยทำการลดขนาดของสัญญาณอินพุตผ่านวงจร Voltage Divider ให้เหลือขนาด 0.6 V ดังรูปที่ 4.19 เพื่อทำให้สัญญาณอลวน (Chaotic signal) สามารถปิดทับสัญญาณอินพุตได้ทั้งหมดเมื่อทำการเข้ารหัสกันด้วยวงจร Summing Amplifier



รูปที่ 4.19 สัญญาณอินพุตที่ใช้ในการทดสอบ

โดยในการทดสอบนี้จะใช้สัญญาณอลวนของ Jerk model ตัวแปรสถานะ z ในการทดสอบ โดยทำการรวมสัญญาณอินพุตเข้ากับสัญญาณอลวน (Chaotic signal) จะได้เป็น Secret signal สามารถแสดงได้ดังรูปที่ 4.20



รูปที่ 4.20 สัญญาณอินพุตเมื่อทำการเข้ารหัส

ในภาครับทำการถอดรหัสโดยลบสัญญาณอลวนที่ Synchronize สำเร็จ และผ่านวงจร Passive Low Pass Filter ที่ความถี่ตัด 3.18 kHz เพื่อกรองเอาสัญญาณที่ไม่ต้องการออก แล้วทำการผ่านวงจร Voltage Comparator เพื่อสร้างสัญญาณเอาต์พุตที่เป็นสัญญาณ Square wave (Regenerated signal) ขนาด 3.3 V ดังรูปที่ 4.21

รูปที่ 4.21 สัญญาณเอาต์พุตเมื่อทำการถอดรหัส

4.6 ทดสอบการแปลงรูปภาพเป็นข้อมูล Text เลขฐาน 16

ในการทดสอบการแปลงภาพเพื่อนำข้อมูลไปเขียนลงบนไมโครคอนโทรลเลอร์จะทำผ่าน IDE คือ Visual Code Studio โดยจะใช้ IPython Notebook (ipynb) โดยรูปภาพที่ใช้ในการทดสอบจะเป็นรูปภาพ tulip ขนาด 10,002,432 บิตดังรูปที่ 4.22 และสามารถแสดงผลการแปลงรูปภาพได้ดังรูปที่ 4.23 และรูปที่ 4.24

รูปที่ 4.22 รูปภาพ tulips ที่ใช้ในการแปลงภาพ

```
[[[ 54  95  79 ... 151  52  47]]

 [[116  38  34 ...  60 119 113]]

 [[ 61 120 116 ...  10  39  37]]

 ...

 [[ 69 151 173 ...  48 104  91]]

 [[ 47 101  87 ... 156 107  40]]

 [[199 128  62 ...   7   8  12]]]
```

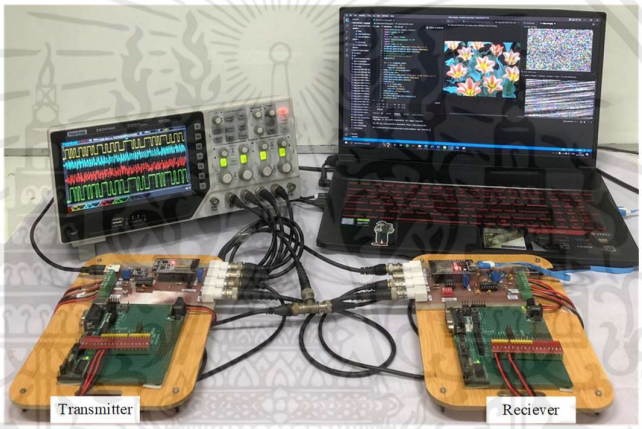
รูปที่ 4.23 ผลแปลงรูปภาพเป็นข้อมูล Array 3 มิติขนาด [22, 1, 60162]

0x13,0x10,0x17,0x1b,0x13,0x1e,0x1b,0x13,0x20,0x1c,0x14,0x21,0x1c,0x14,0x21,0x1a,0x14,
0x1e,0x17,0x11,0x1b,0x15,0xf,0x19,0x14,0xe,0x18,0x14,0xe,0x18,0x13,0xd,0x17,0x12,0xc,0
x16,0x12,0xe,0x16,0x13,0xc,0x14,0x12,0xb,0x13,0x11,0xa,0x12,0x11,0xa,0x12,0x14,0x9,0x1
7,0x14,0xe,0x19,0x13,0xd,0x19,0x11,0xf,0x1a,0x13,0x13,0x1f,0x17,0x19,0x25,0x1d,0x21,0x
2c,0x22,0x24,0x30,0x1f,0x1f,0x2b,0x20,0x1e,0x2b,0x22,0x1c,0x28,0x1f,0x19,0x25,0x1e,0x1
6,0x21,0x1d,0x15,0x20,0x1c,0x16,0x20,0x1b,0x18,0x21,0x1c,0x18,0x26,0x19,0x17,0x24,0x1
8,0x16,0x23,0x18,0x16,0x23,0x1a,0x18,0x25,0x1b,0x19,0x26,0x1b,0x19,0x26,0x1a,0x18,0x2
5,0x1a,0x18,0x23,0x19,0x17,0x22,0x19,0x17,0x22,0x19,0x17,0x22,0x1c,0x1a,0x25,0x1e,0x1
c,0x27,0x1e,0x1b,0x26,0x1a,0x17,0x22,0x1c,0x16,0x22,0x19,0x13,0x1f,0x16,0x10,0x1a,0x1
6,0x10,0x1a,0x18,0x10,0x1b,0x17,0xf,0x1a,0x16,0xe,0x19,0x15,0xd,0x18,0x17,0x11,0x1b,0x
18,0x15,0x1e,0x1c,0x19,0x22,0x1b,0x1b,0x23,0x1b,0x1b,0x23,0x18,0x1b,0x22,0x15,0x1a,0x
20,0x13,0x18,0x1e,0x13,0x15,0x21,0x16,0x18,0x24,0x19,0x19,0x25,0x17,0x17,0x23,0x15,0x
15.0x21.0x15.0x15.0x21.0x15.0x17.0x23.0x15.0x19.0x24.0x1f.0x23.0x2e.0x1d.0x26.0x2f.0x

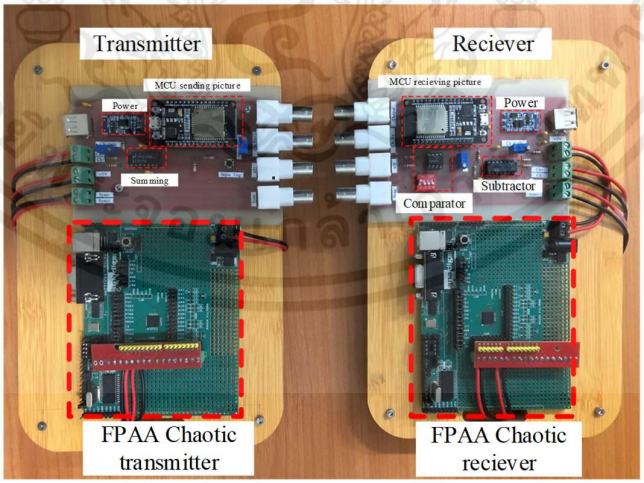
รูปที่ 4.24 การแปลงข้อมูล Array อยู่ในรูป 0x (hexadecimal) และบันทึกเป็นไฟล์ Text

4.7 ผลการทดสอบระบบการสื่อสารโดยรวมทั้งหมดบนแผ่น PCB โดยใช้ข้อมูลรูปภาพในการสื่อสาร

จากหัวข้อก่อนหน้านี้ วงจรที่ได้ทำการออกแบบสำหรับกระบวนการสื่อสารแบบ Chaotic masking สามารถสื่อสารสำเร็จได้ตามทฤษฎีที่ศึกษามา และสามารถแปลงข้อมูลรูปภาพเป็นข้อมูลรูปแบบ 0x (hexadecimal) เพื่อให้ส่งผ่านไมโครคอนโทรลเลอร์ ได้จึงทำการทดสอบระบบสื่อสารรวมทั้งหมดบนแผ่น PCB ที่ได้ทำการออกแบบ โดยในการทดสอบระบบที่ออกแบบจะใช้ไมโครคอนโทรลเลอร์เป็น ESP32 เป็นอุปกรณ์สำหรับใช้ในการส่ง และรับข้อมูลรูปภาพ โดยในการทดสอบสามารถแสดงได้ดังรูปที่ 4.25 และ 4.26



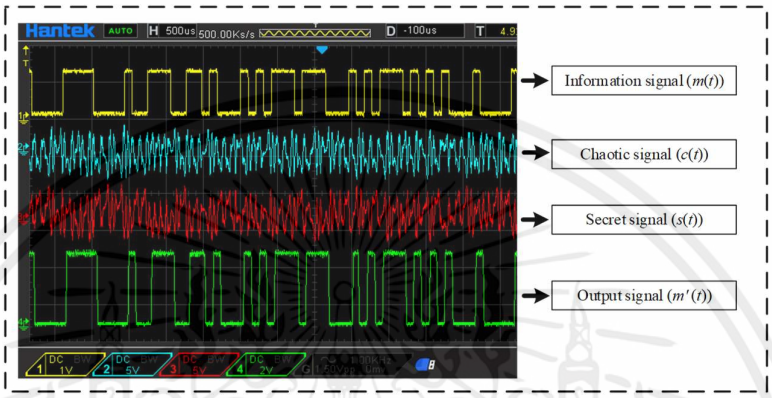
รูปที่ 4.25 ภาพรวมของชุดการทดสอบการสื่อสารโดยรวมบนแผ่น PCB



รูปที่ 4.26 การเชื่อมต่อระหว่างบอร์ด FPAA และ PCB ที่ทำการออกแบบ

4.7.1 ผลการทดสอบการสื่อสารโดยการข้ารหัส และถอดรหัสด้วย Lorenz system

สำหรับ Lorenz system ทำการกำหนด Options และค่า Parameter ต่างๆบน FPAA ดังตารางที่ 4.1 ทั้งภาคส่ง และภาครับซึ่งจะทำให้ Lorenz system ทั้งสองฝั่ง Synchronize ได้สำเร็จ เมื่อทำการส่งข้อมูล และวัดรูปแบบสัญญาณด้วย Oscilloscope สามารถแสดงดังรูปที่ 4.27

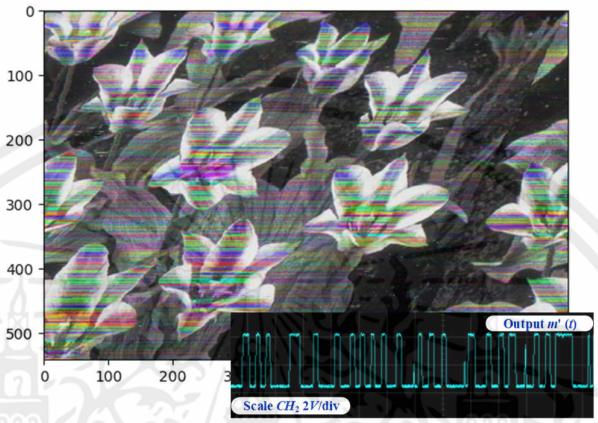


รูปที่ 4.27 รูปแบบสัญญาณของ Chaotic masking communication โดยใช้ Lorenz system

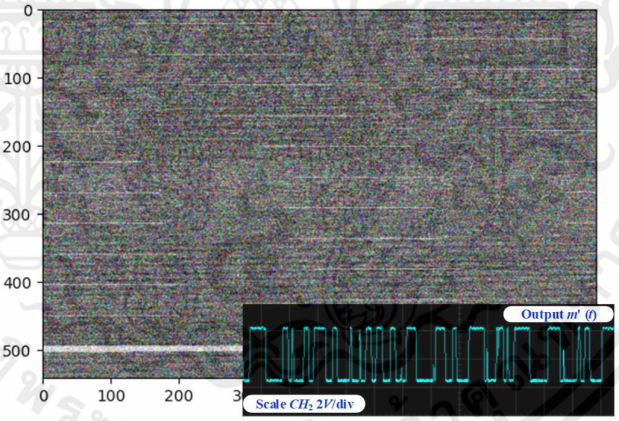
จากรูปที่ 4.27 Information signal $(m(t))$ คือ สัญญาณข้อมูลรูปภาพที่ถูกส่งผ่านช่องสื่อสารอนุกรมของ ESP32, Chaotic signal $(c(t))$ คือ สัญญาณอลวนของ Lorenz system ที่ถูกสร้างที่ FPAA ภาคส่งเพื่อใช้ในการเข้ารหัส, Secret signal $(s(t))$ คือ สัญญาณที่ถูกเข้ารหัสด้วยสัญญาณอลวนแล้วซึ่งจะมีลักษณะเหมือนกันการครอบสัญญาณ $m(t)$ ด้วย $c(t)$ ทำให้ $s(t)$ มีลักษณะคล้ายกับ $c(t)$ และ Output signal $(m'(t))$ คือ สัญญาณที่ทำการถอดรหัสที่ฝั่งรับได้สำเร็จซึ่งสามารถนำสัญญาณ $m'(t)$ ที่ได้มาสร้างเป็นรูปภาพได้ดังรูปที่ 4.28 โดยสามารถวัดอัตราบิตผิดพลาดได้เท่ากับ 42.39×10^{-6}

รูปที่ 4.28 รูปภาพ tulip ที่ถูกสร้างกลับคืนจากสัญญาณ $m'(t)$ $(G_2 = 0.53)$

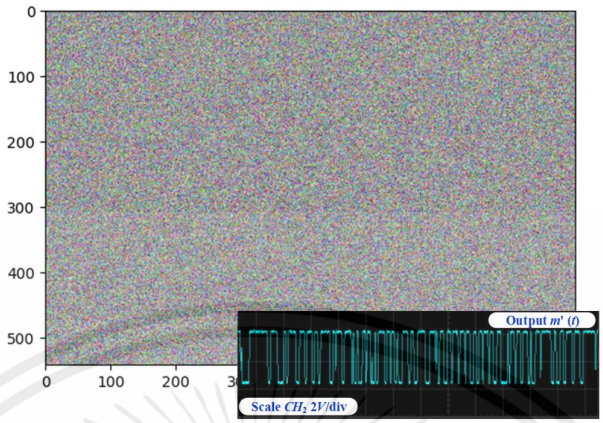
กรณีต่อมาคือกรณที่ FPAA ที่ภาครับมีการกำหนดค่า Parameters ของ CAMs ไม่ตรงกันกับภาคส่งโดยจะแสดงตารางการปรับค่าได้ดังภาคผนวก ข ซึ่งจะทำให้ Lorenz system ทั้งสองฝั่ง Synchronize ไม่สำเร็จ และไม่สามารถถอดรหัสได้ โดยจะสามารถนำสัญญาณ Output signal $(m'(t))$ มาสร้างเป็นรูปภาพแสดงได้ดังรูปที่ 4.29, รูปที่ 4.30 และ รูปที่ 4.31 ตามลำดับ



รูปที่ 4.29 รูปภาพ tulip ที่ถูกสร้างกลับคืนจากสัญญาณ $m'(t)$ $(G_2 = 0.43)$



รูปที่ 4.30 รูปภาพ tulip ที่ถูกสร้างกลับคืนจากสัญญาณ $m'(t)$ $(G_2 = 0.33)$

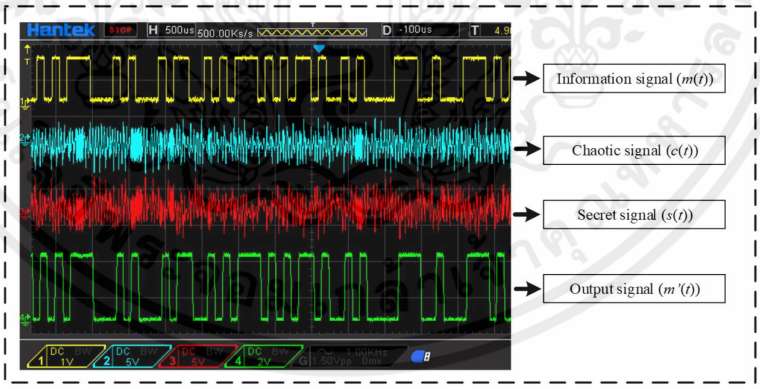


รูปที่ 4.31 รูปภาพ tulip ที่ถูกสร้างกลับคืนจากสัญญาณ $m'(t)$ $(G_2 = 0.23)$

จากรูปที่ 4.29, รูปที่ 4.30 และ รูปที่ 4.31 จะเห็นได้ว่ารูปภาพที่สร้างขึ้นมา และรูปสัญญาณที่ได้จะมีความผิดเพี้ยนมากขึ้นตามลำดับซึ่งเป็นผลมาจากค่า Parameter ของ Lorenz system ที่มีความผิดเพี้ยนมากขึ้นซึ่งจะแสดงได้ดังค่า G_2

4.7.2 ผลการทดสอบการสื่อสารโดยการเข้ารหัส และถอดรหัสด้วย Jerk system

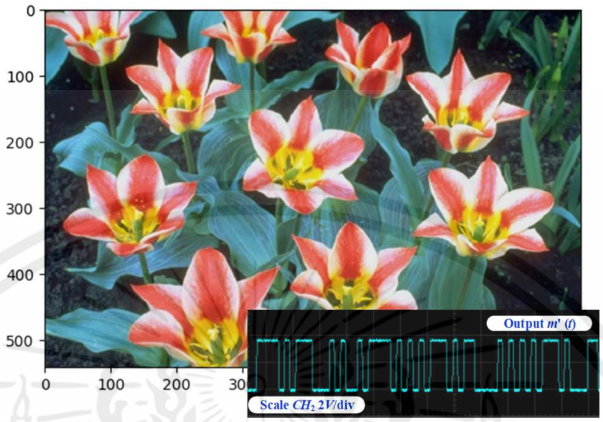
สำหรับ Jerk system ทำการกำหนด Options และค่า Parameter ต่างๆบน FPAA ดังตารางที่ 4.2 ทั้งภาคส่ง และภาครับซึ่งจะทำให้ Jerk system ทั้งสองฝั่ง Synchronize ได้สำเร็จ เมื่อทำการส่งข้อมูล และวัดรูปแบบสัญญาณด้วย Oscilloscope จะสามารถแสดงได้ดังรูปที่ 4.32



รูปที่ 4.32 รูปแบบสัญญาณของ Chaotic masking communication โดยใช้ Jerk system

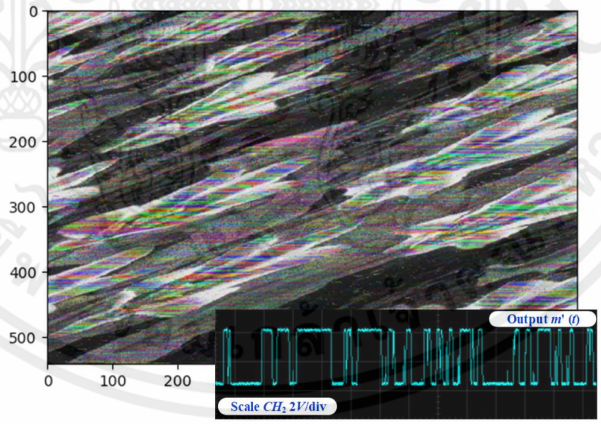
จากรูปที่ 4.32 Information signal $(m(t))$ คือ สัญญาณข้อมูลรูปภาพที่ถูกส่งผ่านช่องสื่อสารอนุกรมของ ESP32, Chaotic signal $(c(t))$ คือ สัญญาณอลวนของ Jerk system ที่ถูกสร้างที่ FPAA ภาคส่งเพื่อใช้ในการเข้ารหัส, Secret signal $(s(t))$ คือ สัญญาณที่ถูกเข้ารหัสด้วยสัญญาณอลวนแล้วซึ่งจะมีลักษณะเหมือนกันการครอบสัญญาณ $m(t)$ ด้วย $c(t)$ ทำให้ $s(t)$ มีลักษณะคล้ายกับ $c(t)$ และ Output signal $(m'(t))$ คือ สัญญาณที่ทำการถอดรหัสที่ฝั่งรับได้สำเร็จซึ่งสามารถ

นำสัญญาณ $m'(t)$ ที่ได้มาสร้างเป็นรูปภาพได้ดังรูปที่ 4.33 โดยสามารถวัดอัตราบิตผิดพลาดได้เท่ากับ 38.29×10^{-6}

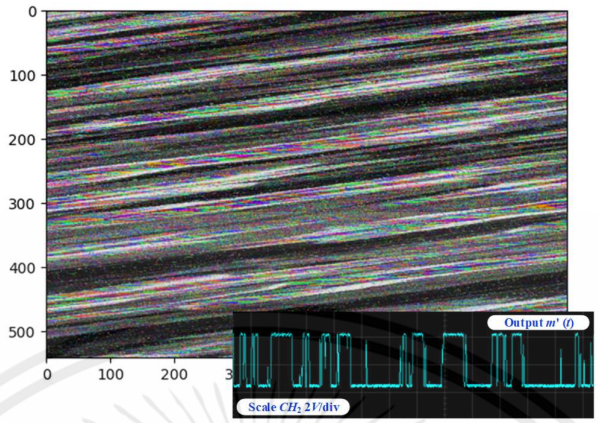


รูปที่ 4.33 รูปภาพ tulip ที่ถูกสร้างกลับคืนจากสัญญาณ $m'(t)$ $(G_{11} = 0.575)$

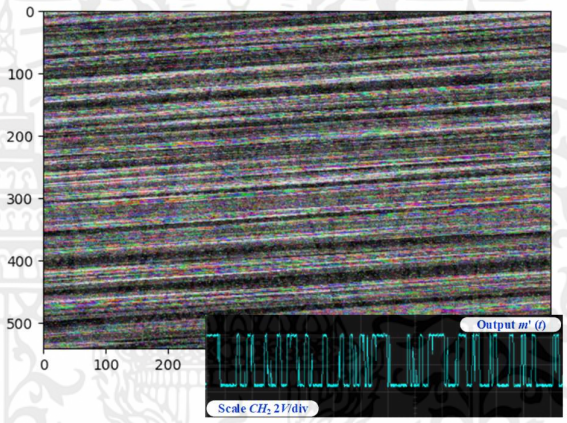
กรณีต่อมาคือกรณีที่ FPAA ที่ภาครับมีการกำหนดค่า Parameters ของ CAMs ไม่ตรงกันกับภาคส่งโดยจะแสดงตารางการปรับค่าได้ดังภาคผนวก ค ซึ่งจะทำให้ Jerk system ทั้งสองฝั่ง Synchronize ไม่สำเร็จ และไม่สามารถถอดรหัสได้ โดยจะสามารถนำสัญญาณ Output signal $(m'(t))$ มาสร้างเป็นรูปภาพแสดงได้ดังรูปที่ 4.34, รูปที่ 4.35 และ รูปที่ 4.36 ตามลำดับ



รูปที่ 4.34 รูปภาพ tulip ที่ถูกสร้างกลับคืนจากสัญญาณ $m'(t)$ $(G_{11} = 1.10)$



รูปที่ 4.35 รูปภาพ tulip ที่ถูกสร้างกลับคืนจากสัญญาณ $m'(t)$ $(G_{11} = 1.20)$

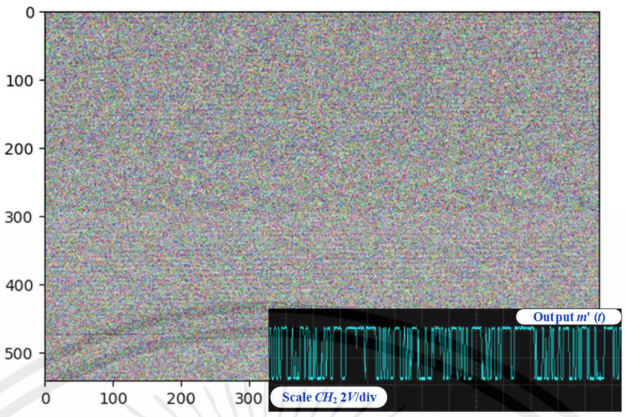


รูปที่ 4.36 รูปภาพ tulip ที่ถูกสร้างกลับคืนจากสัญญาณ $m'(t)$ $(G_{11} = 1.30)$

จากรูปที่ 4.34, รูปที่ 4.35 และ รูปที่ 4.36 จะเห็นได้ว่ารูปภาพที่สร้างขึ้นมา และรูปสัญญาณที่ได้จะมีความผิดเพี้ยนมากขึ้นตามลำดับซึ่งเป็นผลมาจากค่า Parameter ของ Jerk system ภาครับที่มีความผิดเพี้ยนมากขึ้นซึ่งจะแสดงได้ดังค่า G_{11}

4.7.3 ผลการทดสอบการสื่อสารโดยการเข้ารหัส และถอดรหัสด้วยระบบอลวนที่ต่างกัน

ในการทดสอบนี้จะใช้ Lorenz system ในการเข้ารหัส และ jerk system ในการถอดรหัส โดยจะกำหนด Options และ Parameters ตามตาราง 4.1 และตารางที่ 4.2 ตามลำดับโดยจะสามารถนำสัญญาณ Output signal $(m'(t))$ มาสร้างเป็นรูปภาพแสดงได้ดังรูปที่ 4.37



รูปที่ 4.37 รูปภาพ tulip ที่ถูกสร้างกลับคืนจากสัญญาณ $m'(t)$ ที่เข้ารหัส และถอดรหัสด้วยระบบอลวนที่ต่างกัน

จากรูปที่ 4.37 จะเห็นว่ารูปภาพที่สร้าง และรูปสัญญาณ $m'(t)$ มีความผิดเพี้ยนไปจากเดิมเนื่องจากการที่ใช้ระบบอลวนในการเข้ารหัส และถอดรหัสโดยการใช้ระบบอลวนที่ต่างกันจะไม่สามารถถอดรหัสข้อมูลออกมาได้

บทที่ 5

สรุปผลและข้อเสนอแนะ

5.1 สรุปผล

ปริญญานิพนธ์นี้ประสบความสำเร็จในการนำระบบอลวนมาประยุกต์ใช้กับการเข้ารหัสลับข้อมูลด้วยภาพสีด้วยระบบ Lorenz และ Jerk ที่ถูกสร้างขึ้นจากอุปกรณ์ Field Programmable Analog Array (FPAA) และอาศัยหลักการ Chaotic Masking Communication ซึ่งผลลัพธ์จากการถอดรหัสจะถูกแบ่งเป็น 3 กรณี ได้แก่ กรณีที่การ Synchronize สำเร็จ, กรณี Parameter ของระบบมีค่าไม่ตรงกัน และกรณีที่ภาคส่งและภาครับใช้ระบบอลวนต่างกัน โดยในกรณีที่การ Synchronize สำเร็จจะมีค่าอัตราบิตผิดพลาด (Bit error rate) อยู่ที่ 42.39×10^{-6} สำหรับระบบ Lorenz และ 38.29×10^{-6} สำหรับระบบ Jerk และผลลัพธ์จากในทุกกรณีจะถูกแสดงเป็นรูปภาพที่ได้จากการถอดรหัสที่ภาครับ

5.2 ข้อเสนอแนะ

ในการเพิ่มประสิทธิภาพด้านความเร็วการรับส่งสำหรับการเข้ารหัสลับสามารถทำได้โดยการปรับแต่งระบบอลวนให้มีความถี่สูงขึ้นเพื่อรองรับสัญญาณข้อมูลที่มีความถี่สูง แต่เนื่องจากการปรับแต่งระบบเพื่อให้ได้สัญญาณอลวนที่มีความถี่สูงยังติดข้อจำกัดในเรื่องของอุปกรณ์ที่ใช้สำหรับสร้างระบบอลวน หากสามารถใช้อุปกรณ์ที่สามารถกำเนิดความถี่สูงกว่านี้ได้ก็จะสามารถเพิ่มความเร็วสำหรับการรับส่งได้ นอกจากนี้ในส่วนความจุข้อมูลของไมโครคอนโทรลเลอร์ที่มีจำกัดยังส่งผลให้ชนิดของข้อมูลที่ต้องการส่งมีขนาดที่จำกัดตามไปอีกด้วย สามารถแก้ไขได้โดยการใช้ไมโครคอนโทรลเลอร์ที่มีความจุข้อมูลที่มากกว่าหรือการปรับอัลกอริทึมสำหรับการเข้ารหัสบิตข้อมูลใหม่ให้มีจำนวนบิตซ้ำซ้อนน้อยลงกว่าเดิม

บรรณานุกรม

[1] Sun, Kehui. Chaotic Secure Communication. Tsinghua University Press: Walter de Gruyter GmbH, 2016

[2] Capra, Fritjof. (1996). The Web of Life: A New Scientific Understanding of Living Systems. New York : Anchor Books.

[3] Sprott, Julien. (2004). Chaos and Time-Series Analysis. Great Britain : Oxford University Press

[4] Lorenz EN. Deterministic nonperiodic flows. J Atmos Sci 1963, 20, 130–41.

[5] J. C. Sprott, A New Class of Chaotic Circuit, Physics Letters A, vol. 266, 19-23, 2000.

[6] Siriburanon T, Srisuchinwong B, Nontapradit T. Compound structures of six new chaotic attractors in a solely-single-coefficient jerk model with arctangent nonlinearity. In: Proc. of Chinese control and decision conference; 2010. p.985-90. Xuzhou China.

[7] K. Karawanich, and P. Prommee. “High-complex chaotic system based on new nonlinear function and OTA-based circuit realization.” Chaos, Solitons and Fractals. vol. 162, pp. 1-23, 2022.

[8] T. Carroll, and L. Pecora, “Driving systems with chaotic signals,” Physical review. A, vol.44, pp.2374-2383, 1991.

[9] T. Carroll, and L. Pecora, “Synchronization in chaotic systems,” Physical Review Letters, vol.64, no.8, pp.821–824, 1990.

[10] Anadigm SingleApex Development Board, สืบค้นเมื่อวันที่ 15 กันยายน 2565 จาก https://www.anadigm.com/_doc/UM231001-K001.pdf

[11] AN231E04 Datasheet Rev 1.5, สืบค้นเมื่อวันที่ 15 กันยายน 2565 จาก https://www.anadigm.com/_doc/DS231000-U001.pdf

บรรณานุกรม (ต่อ)

[12] AnadigmApex dpASP Family User Manual, สืบค้นเมื่อวันที่ 15 กันยายน 2565 จาก https://www.anadigm.com/_doc/UM000231-U001.pdf

[13] AnadigmDesigner®2 EDA Software, สืบค้นเมื่อวันที่ 15 กันยายน 2565 จาก https://www.anadigm.com/anadigmdesigner2.asp

[14] ออปเปอเรชั่นแนล แอมปลิไฟร์เบื้องต้น, สืบค้นเมื่อวันที่ 16 กุมภาพันธ์ 2566 จาก https://www.g-tech.ac.th/vdo/ELECTRICdoc/วิชาช่าง/E-BOOK%20BASIC%20ELECTRIC%20AND%20ELECTRONICS/อุปกรณ์อิเล็กทรอนิกส์/บทที่%205%20ออปแอมป์%20เบื้องต้น

[15] Op-amp Comparator, สืบค้นเมื่อวันที่ 16 กุมภาพันธ์ 2566 จาก https://www.electronics-tutorials.ws/opamp/op-amp-comparator.html

[16] https://www2.cs.science.cmu.ac.th/courses/204101/lib/exe/fetch.php?media=w02-w03-lab-intro-to-python.pdf

[17] Python Imaging Library, สืบค้นเมื่อวันที่ 16 กุมภาพันธ์ 2566 จาก https://en.wikipedia.org/wiki/Python_Imaging_Library

[18] An introduction to Numpy and Scipy, สืบค้นเมื่อวันที่ 16 กุมภาพันธ์ 2566 จาก https://sites.engineering.ucsb.edu/~shell/che210d/numpy.pdf

[19] แนะนำ ESP32, สืบค้นเมื่อวันที่ 25 มีนาคม 2566 จาก http://ias.it.msu.ac.th/course/1201376-Unix-Linux/2-2561/BasicBook_ESP32.pdf

[20] ESP32 Technical Reference Manual, สืบค้นเมื่อวันที่ 25 มีนาคม 2566 จาก https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf

ภาคผนวก ก

Code MATLAB สำหรับการจำลองระบบบอลวน

Code MATLAB ในการสร้างรูปสัญญาณ และ Attractor

```
clc
clear

global a b c k
a=0.2;
b=0.2;
c=5.7;
k=10;

[T,Y]=ode45(@Chaos_Lorenz_Equation,0:0.001:60,[0 0 1]);
figure(2);
plot(T(1000:end),Y(1000:end,3))  % time domain
% subplot(3,3,[1 3]);plot(T(1000:end),Y(1000:end,1));
ylabel('\itx')
xlabel('\itTime')
% grid on
% subplot(3,3,[4 6]);plot(T(1000:end),Y(1000:end,2));
% ylabel('\ity')
% xlabel('\itTime')
% grid on
% subplot(3,3,[7 9]);plot(T(1000:end),Y(1000:end,3));
% ylabel('\itz')
% xlabel('\itTime')
% grid on
% subplot(4,4,[11 16]);plot(T(1000:end),Y(1000:end,1));
% grid on
% plot(Y(1000:end,1),Y(1000:end,3),'b')  % attractor 2D
% plot3(Y(1000:end,1),Y(1000:end,2),Y(1000:end,3),'r')  %attractor 3D
xlabel('\itx')
ylabel('\itz')
```

```
% zlabel('\itz')
% grid on
hold on;
```

Code MATLAB สมการของระบบบอลวนที่ใช้

```
function f=Chaos_Lorenz_Equation(t,y)
%Y(1)=x ,Y(2)=y ,Y(3)=z
global a b c k
f=zeros(3,1);
k=10;
A=28;
B=8/3;
%  ORIGINAL LORENZ EQUATION
% f(1)=(k*y(2))-(k*y(1));
% f(2)=(A*y(1))-y(2)-(y(1)*y(3));
% f(3)=(y(1)*y(2))-(B*y(3));

%  JERK EQUATION
f(1)=y(2);
f(2)=y(3);
f(3)=-y(2)-(0.62*y(3))+(-1.2*y(1)+2*sign(y(1)));
```

ภาคผนวก ข

การปรับ Options และค่า Parameters ของ Lorenz system บน AnadigmDesigner2 กรณีปรับค่า Parameters

ตารางกรณีการปรับค่า Options และค่า Parameters ของ Lorenz system ของภาครับกรณีปรับค่าตรงกับภาคส่งโดยปรับ Gain 1 SumDiff1 = 5.30

Name	Options	Parameters	Clocks
Multiplier1 **(Multiplier v1.0.2)**  *Anadigm (Approved)*	**Sample and Hold** *Off*	**Multiplication Factor** *1.00*	**ClockA** *250 kHz (Chip Clock 3)* **ClockB** *4 MHz (Chip Clock 0)*
Multiplier2 **(Multiplier v1.0.2)**  *Anadigm (Approved)*	**Sample and Hold** *Off*	**Multiplication Factor** *1.00*	**ClockA** *250 kHz (Chip Clock 3)* **ClockB** *4 MHz (Chip Clock 0)*
SumDiff1 **(SumDiff v1.0.1)**  *Anadigm (Approved)*	**Output Phase** *Phase 2* **Input 1** *Inverting* **Input 2** *Non-inverting* **Input 3** *Off* **Input 4** *Off*	**Gain 1 (UpperInput)** *5.30* **Gain 2 (LowerInput)** *13.1*	**ClockA** *250 kHz (Chip Clock 3)*
SumDiff2 **(SumDiff v1.0.1)**  *Anadigm (Approved)*	**Output Phase** *Phase 1* **Input 1** *Non-inverting* **Input 2** *Inverting* **Input 3** *Inverting* **Input 4** *Off*	**Gain 1 (UpperInput)** *3.75* **Gain 2 (MiddleInput)** *3.00* **Gain 3 (LowerInput)** *32.0*	**ClockA** *250 kHz (Chip Clock 3)*
SumDiff3 **(SumDiff v1.0.1)**  *Anadigm (Approved)*	**Output Phase** *Phase 2* **Input 1** *Non-inverting* **Input 2** *Inverting* **Input 3** *Off* **Input 4** *Off*	**Gain 1 (UpperInput)** *0.715* **Gain 2 (LowerInput)** *1.65*	**ClockA** *250 kHz (Chip Clock 3)*
Integrator1 **(Integrator v1.1.1)**  *Anadigm (Approved)*	**Polarity** *Non-inverting* **Input Sampling Phase** *Phase 1* **Compare Control To** *No Reset*	**Integration Const. [1/us]** *0.0445*	**ClockA** *250 kHz (Chip Clock 3)*
Integrator2 **(Integrator v1.1.1)**  *Anadigm (Approved)*	**Polarity** *Non-inverting* **Input Sampling Phase** *Phase 2* **Compare Control To** *No Reset*	**Integration Const. [1/us]** *0.0445*	**ClockA** *250 kHz (Chip Clock 3)*
Integrator3 **(Integrator v1.1.1)**  *Anadigm (Approved)*	**Polarity** *Non-inverting* **Input Sampling Phase** *Phase 2* **Compare Control To** *No Reset*	**Integration Const. [1/us]** *0.0465*	**ClockA** *250 kHz (Chip Clock 3)*

ตารางกรณีการปรับค่า Options และค่า Parameters ของ Lorenz system ของภาครับกรณีปรับค่าไม่ตรงกับภาคส่งโดยปรับ Gain 1 SumDiff1 = 4.30

Name	Options	Parameters	Clocks
Multiplier1 **(Multiplier v1.0.2)**  *Anadigm (Approved)*	**Sample and Hold** *Off*	**Multiplication Factor** *1.00*	**ClockA** *250 kHz (Chip Clock 3)* **ClockB** *4 MHz (Chip Clock 0)*
Multiplier2 **(Multiplier v1.0.2)**  *Anadigm (Approved)*	**Sample and Hold** *Off*	**Multiplication Factor** *1.00*	**ClockA** *250 kHz (Chip Clock 3)* **ClockB** *4 MHz (Chip Clock 0)*
SumDiff1 **(SumDiff v1.0.1)**  *Anadigm (Approved)*	**Output Phase** *Phase 2* **Input 1** *Inverting* **Input 2** *Non-inverting* **Input 3** *Off* **Input 4** *Off*	**Gain 1 (UpperInput)** *4.30* **Gain 2 (LowerInput)** *13.1*	**ClockA** *250 kHz (Chip Clock 3)*
SumDiff2 **(SumDiff v1.0.1)**  *Anadigm (Approved)*	**Output Phase** *Phase 1* **Input 1** *Non-inverting* **Input 2** *Inverting* **Input 3** *Inverting* **Input 4** *Off*	**Gain 1 (UpperInput)** *3.75* **Gain 2 (MiddleInput)** *3.00* **Gain 3 (LowerInput)** *32.0*	**ClockA** *250 kHz (Chip Clock 3)*
SumDiff3 **(SumDiff v1.0.1)**  *Anadigm (Approved)*	**Output Phase** *Phase 2* **Input 1** *Non-inverting* **Input 2** *Inverting* **Input 3** *Off* **Input 4** *Off*	**Gain 1 (UpperInput)** *0.715* **Gain 2 (LowerInput)** *1.65*	**ClockA** *250 kHz (Chip Clock 3)*
Integrator1 **(Integrator v1.1.1)**  *Anadigm (Approved)*	**Polarity** *Non-inverting* **Input Sampling Phase** *Phase 1* **Compare Control To** *No Reset*	**Integration Const. [1/us]** *0.0445*	**ClockA** *250 kHz (Chip Clock 3)*
Integrator2 **(Integrator v1.1.1)**  *Anadigm (Approved)*	**Polarity** *Non-inverting* **Input Sampling Phase** *Phase 2* **Compare Control To** *No Reset*	**Integration Const. [1/us]** *0.0445*	**ClockA** *250 kHz (Chip Clock 3)*
Integrator3 **(Integrator v1.1.1)**  *Anadigm (Approved)*	**Polarity** *Non-inverting* **Input Sampling Phase** *Phase 2* **Compare Control To** *No Reset*	**Integration Const. [1/us]** *0.0445*	**ClockA** *250 kHz (Chip Clock 3)*

ตารางกรณีการปรับค่า Options และค่า Parameters ของ Lorenz system ของภาครับกรณีปรับค่าไม่ตรงกับภาคส่งโดยปรับ Gain 1 SumDiff1 = 3.30

Name	Options	Parameters	Clocks
Multiplier1 **(Multiplier v1.0.2)**  *Anadigm (Approved)*	**Sample and Hold** *Off*	**Multiplication Factor** *1.00*	**ClockA** *250 kHz (Chip Clock 3)* **ClockB** *4 MHz (Chip Clock 0)*
Multiplier2 **(Multiplier v1.0.2)**  *Anadigm (Approved)*	**Sample and Hold** *Off*	**Multiplication Factor** *1.00*	**ClockA** *250 kHz (Chip Clock 3)* **ClockB** *4 MHz (Chip Clock 0)*
SumDiff1 **(SumDiff v1.0.1)**  *Anadigm (Approved)*	**Output Phase** *Phase 2* **Input 1** *Inverting* **Input 2** *Non-inverting* **Input 3** *Off* **Input 4** *Off*	**Gain 1 (UpperInput)** *3.30* **Gain 2 (LowerInput)** *13.1*	**ClockA** *250 kHz (Chip Clock 3)*
SumDiff2 **(SumDiff v1.0.1)**  *Anadigm (Approved)*	**Output Phase** *Phase 1* **Input 1** *Non-inverting* **Input 2** *Inverting* **Input 3** *Inverting* **Input 4** *Off*	**Gain 1 (UpperInput)** *3.75* **Gain 2 (MiddleInput)** *3.00* **Gain 3 (LowerInput)** *32.0*	**ClockA** *250 kHz (Chip Clock 3)*
SumDiff3 **(SumDiff v1.0.1)**  *Anadigm (Approved)*	**Output Phase** *Phase 2* **Input 1** *Non-inverting* **Input 2** *Inverting* **Input 3** *Off* **Input 4** *Off*	**Gain 1 (UpperInput)** *0.715* **Gain 2 (LowerInput)** *1.65*	**ClockA** *250 kHz (Chip Clock 3)*
Integrator1 **(Integrator v1.1.1)**  *Anadigm (Approved)*	**Polarity** *Non-inverting* **Input Sampling Phase** *Phase 1* **Compare Control To** *No Reset*	**Integration Const. [1/us]** *0.0445*	**ClockA** *250 kHz (Chip Clock 3)*
Integrator2 **(Integrator v1.1.1)**  *Anadigm (Approved)*	**Polarity** *Non-inverting* **Input Sampling Phase** *Phase 2* **Compare Control To** *No Reset*	**Integration Const. [1/us]** *0.0445*	**ClockA** *250 kHz (Chip Clock 3)*
Integrator3 **(Integrator v1.1.1)**  *Anadigm (Approved)*	**Polarity** *Non-inverting* **Input Sampling Phase** *Phase 2* **Compare Control To** *No Reset*	**Integration Const. [1/us]** *0.0445*	**ClockA** *250 kHz (Chip Clock 3)*

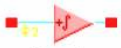
ตารางกรณีการปรับค่า Options และค่า Parameters ของ Lorenz system ของภาครับกรณีปรับค่าไม่ตรงกับภาคส่งโดยปรับ Gain 1 SumDiff1 = 2.30

Name	Options	Parameters	Clocks
Multiplier1 **(Multiplier v1.0.2)**  *Anadigm (Approved)*	**Sample and Hold** *Off*	**Multiplication Factor** *1.00*	**ClockA** *250 kHz (Chip Clock 3)* **ClockB** *4 MHz (Chip Clock 0)*
Multiplier2 **(Multiplier v1.0.2)**  *Anadigm (Approved)*	**Sample and Hold** *Off*	**Multiplication Factor** *1.00*	**ClockA** *250 kHz (Chip Clock 3)* **ClockB** *4 MHz (Chip Clock 0)*
SumDiff1 **(SumDiff v1.0.1)**  *Anadigm (Approved)*	**Output Phase** *Phase 2* **Input 1** *Inverting* **Input 2** *Non-inverting* **Input 3** *Off* **Input 4** *Off*	**Gain 1 (UpperInput)** *2.30* **Gain 2 (LowerInput)** *13.1*	**ClockA** *250 kHz (Chip Clock 3)*
SumDiff2 **(SumDiff v1.0.1)**  *Anadigm (Approved)*	**Output Phase** *Phase 1* **Input 1** *Non-inverting* **Input 2** *Inverting* **Input 3** *Inverting* **Input 4** *Off*	**Gain 1 (UpperInput)** *3.75* **Gain 2 (MiddleInput)** *3.00* **Gain 3 (LowerInput)** *32.0*	**ClockA** *250 kHz (Chip Clock 3)*
SumDiff3 **(SumDiff v1.0.1)**  *Anadigm (Approved)*	**Output Phase** *Phase 2* **Input 1** *Non-inverting* **Input 2** *Inverting* **Input 3** *Off* **Input 4** *Off*	**Gain 1 (UpperInput)** *0.715* **Gain 2 (LowerInput)** *1.65*	**ClockA** *250 kHz (Chip Clock 3)*
Integrator1 **(Integrator v1.1.1)**  *Anadigm (Approved)*	**Polarity** *Non-inverting* **Input Sampling Phase** *Phase 1* **Compare Control To** *No Reset*	**Integration Const. [1/us]** *0.0445*	**ClockA** *250 kHz (Chip Clock 3)*
Integrator2 **(Integrator v1.1.1)**  *Anadigm (Approved)*	**Polarity** *Non-inverting* **Input Sampling Phase** *Phase 2* **Compare Control To** *No Reset*	**Integration Const. [1/us]** *0.0445*	**ClockA** *250 kHz (Chip Clock 3)*
Integrator3 **(Integrator v1.1.1)**  *Anadigm (Approved)*	**Polarity** *Non-inverting* **Input Sampling Phase** *Phase 2* **Compare Control To** *No Reset*	**Integration Const. [1/us]** *0.0445*	**ClockA** *250 kHz (Chip Clock 3)*

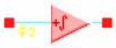
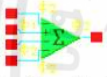
ภาคผนวก ค

การปรับ Options และค่า Parameters ของ Jerk system บน AnadigmDesigner2 กรณีปรับค่า Parameters ภาครับไม่ตรงกับภาคส่ง

ตารางกรณีการปรับค่า Options และค่า Parameters ของ Jerk system ของภาครับกรณีปรับค่าตรงกับภาคส่งโดยปรับ Gain 2 SumDiff1 =0.575

Name	Options	Parameters	Clocks
Integrator1 (Integrator v1.1.1)  *Anadigm (Approved)*	**Polarity** *Non-inverting* **Input Sampling Phase** *Phase 2* **Compare Control To** *No Reset*	**Integration Const. [1/us]** *0.203*	**ClockA** *250 kHz (Chip Clock 3)*
Integrator2 (Integrator v1.1.1)  *Anadigm (Approved)*	**Polarity** *Non-inverting* **Input Sampling Phase** *Phase 2* **Compare Control To** *No Reset*	**Integration Const. [1/us]** *0.203*	**ClockA** *250 kHz (Chip Clock 3)*
Integrator3 (Integrator v1.1.1)  *Anadigm (Approved)*	**Polarity** *Non-inverting* **Input Sampling Phase** *Phase 2* **Compare Control To** *No Reset*	**Integration Const. [1/us]** *0.203*	**ClockA** *250 kHz (Chip Clock 3)*
SumDiff1 (SumDiff v1.0.1)  *Anadigm (Approved)*	**Output Phase** *Phase 2* **Input 1** *Inverting* **Input 2** *Non-inverting* **Input 3** *Inverting* **Input 4** *Inverting*	**Gain 1 (UpperInput)** **Gain 2 (SecondInput)** **Gain 3 (ThirdInput)** **Gain 4 (LowerInput)**	**ClockA** *250 kHz (Chip Clock 3)*
GainLimiter1 (GainLimiter v1.0.1)  *Anadigm (Approved*)*		**Gain** *20.0* **Output Voltage Limit** *1.00*	**ClockA** *250 kHz (Chip Clock 3)*

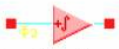
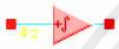
ตารางกรณีการปรับค่า Options และค่า Parameters ของ Jerk system ของภาครับกรณีปรับค่าไม่ตรงกับภาคส่งโดยปรับ Gain 2 SumDiff1 =1.10

Name	Options	Parameters	Clocks
Integrator1 (Integrator v1.1.1)  *Anadigm (Approved)*	**Polarity** *Non-inverting* **Input Sampling Phase** *Phase 2* **Compare Control To** *No Reset*	**Integration Const. [1/us]** *0.203*	**ClockA** *250 kHz (Chip Clock 3)*
Integrator2 (Integrator v1.1.1)  *Anadigm (Approved)*	**Polarity** *Non-inverting* **Input Sampling Phase** *Phase 2* **Compare Control To** *No Reset*	**Integration Const. [1/us]** *0.203*	**ClockA** *250 kHz (Chip Clock 3)*
Integrator3 (Integrator v1.1.1)  *Anadigm (Approved)*	**Polarity** *Non-inverting* **Input Sampling Phase** *Phase 2* **Compare Control To** *No Reset*	**Integration Const. [1/us]** *0.203*	**ClockA** *250 kHz (Chip Clock 3)*
SumDiff1 (SumDiff v1.0.1)  *Anadigm (Approved)*	**Output Phase** *Phase 2* **Input 1** *Inverting* **Input 2** *Non-inverting* **Input 3** *Inverting* **Input 4** *Inverting*	**Gain 1 (UpperInput)** *1.00* **Gain 2 (SecondInput)** *2.20* **Gain 3 (ThirdInput)** *1.79* **Gain 4 (LowerInput)** *0.825*	**ClockA** *250 kHz (Chip Clock 3)*
GainLimiter1 (GainLimiter v1.0.1)  *Anadigm (Approved*)*		**Gain** *20.0* **Output Voltage Limit** *1.00*	**ClockA** *250 kHz (Chip Clock 3)*

ตารางกรณีการปรับค่า Options และค่า Parameters ของ Jerk system ของภาครับกรณีปรับค่าไม่ตรงกับภาคส่งโดยปรับ Gain 2 SumDiff1 =1.20

Name	Options	Parameters	Clocks
Integrator1 (Integrator v1.1.1)  *Anadigm (Approved)*	**Polarity** *Non-inverting* **Input Sampling Phase** *Phase 2* **Compare Control To** *No Reset*	**Integration Const. [1/us]** *0.203*	**ClockA** *250 kHz (Chip Clock 3)*
Integrator2 (Integrator v1.1.1)  *Anadigm (Approved)*	**Polarity** *Non-inverting* **Input Sampling Phase** *Phase 2* **Compare Control To** *No Reset*	**Integration Const. [1/us]** *0.203*	**ClockA** *250 kHz (Chip Clock 3)*
Integrator3 (Integrator v1.1.1)  *Anadigm (Approved)*	**Polarity** *Non-inverting* **Input Sampling Phase** *Phase 2* **Compare Control To** *No Reset*	**Integration Const. [1/us]** *0.203*	**ClockA** *250 kHz (Chip Clock 3)*
SumDiff1 (SumDiff v1.0.1)  *Anadigm (Approved)*	**Output Phase** *Phase 2* **Input 1** *Inverting* **Input 2** *Non-inverting* **Input 3** *Inverting* **Input 4** *Inverting*	**Gain 1 (UpperInput)** *1.00* **Gain 2 (SecondInput)** *2.40* **Gain 3 (ThirdInput)** *1.79* **Gain 4 (LowerInput)** *0.825*	**ClockA** *250 kHz (Chip Clock 3)*
GainLimiter1 (GainLimiter v1.0.1)  *Anadigm (Approved*)*		**Gain** *20.0* **Output Voltage Limit** *1.00*	**ClockA** *250 kHz (Chip Clock 3)*

ตารางกรณีการปรับค่า Options และค่า Parameters ของ Jerk system ของภาครับกรณีปรับค่าไม่ตรงกับภาคส่งโดยปรับ Gain 2 SumDiff1 =1.30

Name	Options	Parameters	Clocks
Integrator1 (Integrator v1.1.1)  *Anadigm (Approved)*	**Polarity** *Non-inverting* **Input Sampling Phase** *Phase 2* **Compare Control To** *No Reset*	**Integration Const. [1/us]** *0.203*	**ClockA** *250 kHz (Chip Clock 3)*
Integrator2 (Integrator v1.1.1)  *Anadigm (Approved)*	**Polarity** *Non-inverting* **Input Sampling Phase** *Phase 2* **Compare Control To** *No Reset*	**Integration Const. [1/us]** *0.203*	**ClockA** *250 kHz (Chip Clock 3)*
Integrator3 (Integrator v1.1.1)  *Anadigm (Approved)*	**Polarity** *Non-inverting* **Input Sampling Phase** *Phase 2* **Compare Control To** *No Reset*	**Integration Const. [1/us]** *0.203*	**ClockA** *250 kHz (Chip Clock 3)*
SumDiff1 (SumDiff v1.0.1)  *Anadigm (Approved)*	**Output Phase** *Phase 2* **Input 1** *Inverting* **Input 2** *Non-inverting* **Input 3** *Inverting* **Input 4** *Inverting*	**Gain 1 (UpperInput)** *1.00* **Gain 2 (SecondInput)** *2.60* **Gain 3 (ThirdInput)** *1.78* **Gain 4 (LowerInput)** *0.825*	**ClockA** *250 kHz (Chip Clock 3)*
GainLimiter1 (GainLimiter v1.0.1)  *Anadigm (Approved*)*		**Gain** *20.0* **Output Voltage Limit** *1.00*	**ClockA** *250 kHz (Chip Clock 3)*

ภาคผนวก ง

Code การทำงานของ ESP32 ภาคส่ง และภาครับ

Code การทำงานของ ESP32 ภาคส่ง

```
#define RXp2 16
#define TXp2 17
unsigned char EncodeSet0[60162];
unsigned char EncodeSet1[60162];
unsigned char EncodeSet2[60162];
unsigned char EncodeSet3[60162];
unsigned char EncodeSet4[60162];
unsigned char EncodeSet5[60162];
unsigned char EncodeSet6[60162];
unsigned char EncodeSet7[60162];
unsigned char EncodeSet8[60162];
unsigned char EncodeSet9[60162];
unsigned char EncodeSet10[60162];
unsigned char EncodeSet11[60162];
unsigned char EncodeSet12[60162];
unsigned char EncodeSet13[60162];
unsigned char EncodeSet14[60162];
unsigned char EncodeSet15[60162];
unsigned char EncodeSet16[60162];
unsigned char EncodeSet17[60162];
unsigned char EncodeSet18[60162];
unsigned char EncodeSet19[60162];
unsigned char EncodeSet20[60162];

int checkloop = 1;
int check = LOW ;
char Send_data;
int k = 0;
void setup() {
```

```
Serial.begin(115200);
Serial2.begin(9600, SERIAL_8N1, RXp2, TXp2);
pinMode(19, INPUT);
check = digitalRead(19);
while (checkloop == 1) {
  check = digitalRead(19);
  Serial.println("reading");
  if (check == HIGH)
  {
    Serial.println("Going");
    checkloop = 0;
  }
}

for (int k = 0; k < 60162; k++) {
  Serial2.write(EncodeSet0[k]);


if (k % 407 == 0) {
    delay(110);
  }
}
delay(100);
for (int k = 0; k < 60162; k++) {
  Serial2.write(EncodeSet1[k]);
  if (k % 407== 0) {
    delay(110);
  }
}
delay(1000);
for (int k = 0; k < 60162; k++) {
```

```
  Serial2.write(EncodeSet2[k]);
  if (k % 407 == 0) {
    delay(110);
  }
}
delay(100);
for (int k = 0; k < 60162; k++) {
  Serial2.write(EncodeSet3[k]);
  if (k % 407 == 0) {
    delay(110);
  }
}
delay(100);
for (int k = 0; k < 60162; k++) {
  Serial2.write(EncodeSet4[k]);
  if (k % 407  == 0) {
    delay(110);
  }
}
delay(1000);
for (int k = 0; k < 60162; k++) {
  Serial2.write(EncodeSet5[k]);
  if (k % 407  == 0) {
    delay(110);
  }
}
delay(100);
for (int k = 0; k < 60162; k++) {
  Serial2.write(EncodeSet6[k]);
  if (k % 407  == 0) {
    delay(110);
```

```
  }
}
delay(100);
for (int k = 0; k < 60162; k++) {
  Serial2.write(EncodeSet7[k]);
  if (k % 407  == 0) {
    delay(110);
  }
}
delay(100);
for (int k = 0; k < 60162; k++) {
  Serial2.write(EncodeSet8[k]);
  if (k % 407  == 0) {
    delay(110);
  }
}
delay(100);
for (int k = 0; k < 60162; k++) {
  Serial2.write(EncodeSet9[k]);
  if (k % 407== 0) {
    delay(110);
  }
}
delay(100);
for (int k = 0; k < 60162; k++) {
  Serial2.write(EncodeSet10[k]);
  if (k % 407 == 0) {
    delay(110);
  }
}
delay(100);
```

```
for (int k = 0; k < 60162; k++) {
  Serial2.write(EncodeSet11[k]);
  if (k % 407 == 0) {
    delay(110);
  }
}
delay(100);
for (int k = 0; k < 60162; k++) {
  Serial2.write(EncodeSet12[k]);
  if (k % 407 == 0) {
    delay(110);
  }
}
delay(100);
for (int k = 0; k < 60162; k++) {
  Serial2.write(EncodeSet13[k]);
  if (k % 407 == 0) {
    delay(110);
  }
}
delay(100);
for (int k = 0; k < 60162; k++) {
  Serial2.write(EncodeSet14[k]);
  if (k % 407 == 0) {
    delay(110);
  }
}
delay(100);
for (int k = 0; k < 60162; k++) {
  Serial2.write(EncodeSet15[k]);
  if (k % 407== 0) {
```

```
    delay(110);
  }
}
delay(100);
for (int k = 0; k < 60162; k++) {
  Serial2.write(EncodeSet16[k]);
  if (k % 407 == 0) {
    delay(110);
  }
}
delay(100);
for (int k = 0; k < 60162; k++) {
  Serial2.write(EncodeSet17[k]);
  if (k % 407 == 0) {
    delay(110);
  }
}
delay(100);
for (int k = 0; k < 60162; k++) {
  Serial2.write(EncodeSet18[k]);
  if (k % 407 == 0) {
    delay(110);
  }
}
delay(100);
for (int k = 0; k < 60162; k++) {
  Serial2.write(EncodeSet19[k]);
  if (k % 407 == 0) {
    delay(110);
  }
}
```

```
  delay(100);
  for (int k = 0; k < 60162; k++) {
    Serial2.write(EncodeSet20[k]);
    if (k % 407 == 0) {
      delay(110);
    }
  }
  delay(100);
  for (int k = 0; k < 60162; k++) {
    Serial2.write(EncodeSet21[k]);
    if (k % 407 == 0) {
      delay(110);
    }
  }
  delay(100);
}
void loop() {
}
```

Code การทำงานของ ESP32 ภาครับ

```
#define RXp2 16
#define TXp2 17
void setup() {
  Serial.begin(115200);
  Serial2.begin(9600, SERIAL_8N1, RXp2, TXp2);
}
void loop() {
  if ((Serial2.available()) > 0 ) {
    Serial.println(Serial2.read(), DEC);
  }
                                                    }
```

ภาคผนวก จ

Code การแปลงข้อรูปรูปภาพเป็นข้อมูล 0x (hexadecimal)

Code การแปลงข้อรูปรูปภาพเป็นข้อมูล 0x (hexadecimal)

```
import numpy  as np
from numpy import as array
from numpy import uint8

import PIL
from PIL import Image
import matplotlib.pyplot as plt
import csv
image=Image.open('tulips814x512.jpg') #read imange to byte
data_image = asarray(image)
data_reshape = data_image.reshape([22,1,60162])#reshape to arduino
x=[]
data_test=[]
for i in range (0,22): #11byte for array in arduino
    tohex=data_reshape[i].reshape([-1,])
    print(tohex)
    print(tohex.shape)
    hex_array=[hex(x) for x in tohex]
    hex_array=np.array(hex_array)
    print(hex_array)
    data_test.append(hex_array)
with open('BIT0', 'w') as f:
        write=csv.writer(f)
        write.writerow(data_test[0])
with open('BIT1', 'w') as f:
        write=csv.writer(f)
        write.writerow(data_test[1])
with open('BIT2', 'w') as f:
        write=csv.writer(f)
        write.writerow(data_test[2])
```

```
with open('BIT3', 'w') as f:
        write=csv.writer(f)
        write.writerow(data_test[3])
with open('BIT4', 'w') as f:
        write=csv.writer(f)
        write.writerow(data_test[4])
with open('BIT5', 'w') as f:
        write=csv.writer(f)
        write.writerow(data_test[5])
with open('BIT6', 'w') as f:
        write=csv.writer(f)
        write.writerow(data_test[6])
with open('BIT7', 'w') as f:
        write=csv.writer(f)
        write.writerow(data_test[7])
with open('BIT8', 'w') as f:
        write=csv.writer(f)
        write.writerow(data_test[8])
with open('BIT9', 'w') as f:
        write=csv.writer(f)
        write.writerow(data_test[9])
with open('BIT10', 'w') as f:
        write=csv.writer(f)
        write.writerow(data_test[10])
with open('BIT11', 'w') as f:
        write=csv.writer(f)
        write.writerow(data_test[11])
with open('BIT12', 'w') as f:
        write=csv.writer(f)
        write.writerow(data_test[12])
with open('BIT13', 'w') as f:
```

```
        write=csv.writer(f)
        write.writerow(data_test[13])
with open('BIT14', 'w') as f:
        write=csv.writer(f)
        write.writerow(data_test[14])
with open('BIT15', 'w') as f:
        write=csv.writer(f)
        write.writerow(data_test[15])
with open('BIT16', 'w') as f:
        write=csv.writer(f)
        write.writerow(data_test[16])
with open('BIT17', 'w') as f:
        write=csv.writer(f)
        write.writerow(data_test[17])
with open('BIT18', 'w') as f:
        write=csv.writer(f)
        write.writerow(data_test[18])
with open('BIT19', 'w') as f:
        write=csv.writer(f)
        write.writerow(data_test[19])
with open('BIT20', 'w') as f:
        write=csv.writer(f)
        write.writerow(data_test[20])
with open('BIT21', 'w') as f:
        write=csv.writer(f)
        write.writerow(data_test[21])
```

ภาคผนวก ฉ

Code การรับข้อมูลที่ภาครับ และแปลงเป็นรูปภาพ

Code การรับข้อมูลที่ภาครับ

```
import serial
import csv

arduino_port = 'COM5' #serial port of Arduino
baud = 115200
fileName="10datam.csv" #name of the CSV file generated
ser = serial.Serial(arduino_port, baud)
print("Connected to Arduino port:" + arduino_port)
file = open(fileName, "a")
print("Created file")
samples = 1323564#how many samples to collect
print_labels = False
line = 1 #start at 0 because our header is 0 (not real data)
store_data = [] #store data
# collect the samples
print("Start Reading Data")
p = 1
if p==1:
    store_data.append("d")
    p=0
while line <= samples:
    getData=ser.readline()
    dataString = getData.decode('utf-8')
    data=dataString[0:][:-2]
    print(data)

    readings = data.split(",")
    # print(readings)
```

```
    store_data.append(readings)
    # print(sensor_data)
    line = line+1
with open(fileName, 'w', encoding='UTF8', newline='') as f:
    writer = csv.writer(f)
    writer.writerows(store_data)

print("Data collection complete!")
file.close()
ser.close()
```

Code การนำข้อมูลที่ได้รับแปลงเป็นรูปภาพ

```
import pandas as pd
import PIL
from PIL import Image
import matplotlib.pyplot as plt
import csv
df = pd.read_csv('10datam.csv')
df=df.values.flatten()
rxdata=df
print("Read Data: ",df)
#reshape
print("Reshaped Data:")
data3D_recover=df.reshape([542,814,3]) #reshape in to original shape
print(data3D_recover)
PIL_image_recover = Image.fromarray(data3D_recover.astype('uint8'), 'RGB')
plt.imshow(PIL_image_recover) #read image
```