กล้องรักษาความปลอดภัยผ่านเครือข่าย IP

SECURITY CAMERA VIA IP NETWORK

โดย

นายศุภโชค กุมาร

นายศุภลักษณ์ แบนใจวาง

นายสาธิต ชอบทองหลาง

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2548

ผ่านการตรวจชิ้นงานแล้ว

(ลงชื่อ).....ผู้ตรวจ

ผ่านการตรวจรูปเล่มแล้ว

(ลงชื่อ).....ผู้ตรวจ

กล้องรักษาความปลอดภัยผ่านเครือข่าย IP

SECURITY CAMERA VIA IP NETWORK

โดย

นายศุภโชค กุมาร 45010780

นายศุภลักษณ์ แบนใจวาง 45010784

นายสาธิต ชอบทองหลาง 45010819

อาจารย์ที่ปรึกษา

ดร. พรชัย ทรัพย์นิธิ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2548

ปริญญานิพนธ์ปีการศึกษา 2548

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง **กล้องรักษาความปลอดภัยผ่านเครือข่าย IP**

SECURITY CAMERA VIA IP NETWORK

ผู้จัดทำ

1. **นายศุภโชค กุมาร 45010780**
2. **นายศุภลักษณ์ แบนใจวาง 45010784**
3. **นายสาธิต ชอบทองหลาง 45010819**

..... อาจารย์ที่ปรึกษา

(ดร. พรชัย ทรัพย์นิธิ)

กล้องรักษาความปลอดภัยผ่านเครือข่าย IP

SECURITY CAMERA VIA IP NETWORK

โดย นายศุภโชค กุมาร 45010780

นายศุภลักษณ์ แบนใจวาง 45010784

นายสาธิต ชอบทองหลาง 45010819

อาจารย์ที่ปรึกษา ดร. พรชัย ทรัพย์นิธิ

บทคัดย่อ

โครงงานนี้นำเสนอการสร้างระบบกล้องโทรทัศน์วงจรปิดรักษาความปลอดภัยผ่านเครือข่าย IP โดยรับสัญญาณภาพจากกล้องโทรทัศน์ซึ่งตั้งอยู่ตามที่ต่างๆ สัญญาณภาพจะถูกสวิตช์เข้าสู่เครื่องคอมพิวเตอร์ที่เป็นผู้ให้บริการ ซึ่งผู้ให้บริการสามารถให้บริการแสดงผลภาพบนจอคอมพิวเตอร์เพื่อให้ผู้ใช้บริการรับชมได้

Abstract

This project aims to design a security camera system via the IP network . The host computer receives the signal from camera switcher and display video streaming. User is able to see the video streaming via the IP network.

กิตติกรรมประกาศ

ปริญญานิพนธ์นี้สำเร็จลุล่วงได้ด้วยดีเนื่องมาจากคำชี้แนะจากอาจารย์ ดร. พรชัย ทรัพย์นิธิ ที่กรุณาให้ความรู้ ความช่วยเหลือ ให้คำปรึกษาแนะนำ รวมถึงอนุเคราะห์เครื่องมือ และ อุปกรณ์ต่างๆ ตลอดจนตรวจสอบแก้ไขจนสำเร็จลงได้

คณาจารย์ทุกท่าน ที่ประสิทธิ์ประสาทวิชาความรู้ และ พี่ เพื่อนนักศึกษาทุกท่านที่ช่วยเหลือให้คำปรึกษาและแนะนำในหลายๆด้าน

สุดท้ายนี้ขอกราบขอบคุณ คุณพ่อ คุณแม่ ที่ให้การเลี้ยงดูอบรม และส่งเสริมการศึกษาเป็นอย่างดีตลอดมาในอดีตจนถึงปัจจุบัน ทำให้ผู้จัดทำประสบความสำเร็จในชีวิตตลอดมา

สารบัญ

หน้า

สารบัญ (ต่อ)

สารบัญรูปภาพ

สารบัญรูปภาพ (ต่อ)

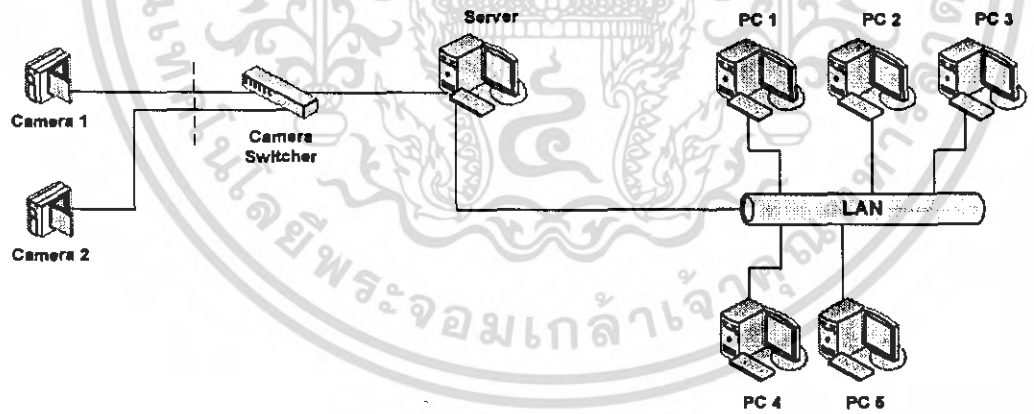
บทที่1
บทนำ

ในปัจจุบันเครือข่ายอินเทอร์เน็ตได้มีส่วนสำคัญในการสื่อสารอย่างมาก เนื่องจากเป็นเครือข่ายที่เชื่อมต่อกันได้ทั่วโลกมีบริการที่หลากหลายที่กระทำผ่านเครือข่าย และ ความเร็วในการส่งข้อมูลก็เพิ่มขึ้น ด้วยเหตุนี้โครงงานนี้จึงได้นำเอาประโยชน์ของอินเทอร์เน็ตมาประยุกต์เข้ากับการรักษาความปลอดภัย โดยสามารถสังเกตการณ์พื้นที่ ที่ต้องการจากต่างพื้นที่ ที่สามารถเชื่อมโยงสู่เครือข่ายอินเทอร์เน็ตได้ โดยระบบประกอบด้วยกล้องที่ติดตั้งไว้ตามที่ต่างๆ ที่ส่งข้อมูลเข้าสู่คอมพิวเตอร์ที่ติดต่ออยู่กับเครือข่ายอินเทอร์เน็ต และส่งข้อมูลไปยังเครื่องปลายทางที่ต้องการดูภาพจากกล้องโดยผ่านเครือข่ายอินเทอร์เน็ตโดยเครื่องปลายทางจะต้องติดตั้งโปรแกรมที่ใช้เพื่อเรียกดูภาพจากกล้องโทรทัศน์วงจรปิด

1.1 วัตถุประสงค์ของโครงงาน

1. เพื่อศึกษาการสื่อสารโดยส่งข้อมูลภาพเคลื่อนไหวผ่านเครือข่าย IP (Internet Protocol)
2. เพื่อศึกษาการเขียนโปรแกรมประเภทติดต่อกับเครือข่าย
3. เพื่อสร้างเครื่องสลับสัญญาณภาพแบบอัตโนมัติ และแบบกำหนดตำแหน่งอินพุทได้

1.2 โครงสร้างโดยรวมของโครงงาน



รูปที่ 1.1 โครงสร้างของระบบ

หมายเหตุ

Camera1 = กล้องโทรทัศน์วงจรปิดตัวที่ 1 Camera2 = กล้องโทรทัศน์วงจรปิดตัวที่ 2

LAN = เครือขายท้องถิ่น (Local Area Network) PC = เครื่องคอมพิวเตอร์ส่วนบุคคล (Personal Computer)

บทที่ 2
ทฤษฎีและหลักการ

2.1 กล้องโทรทัศน์วงจรปิด

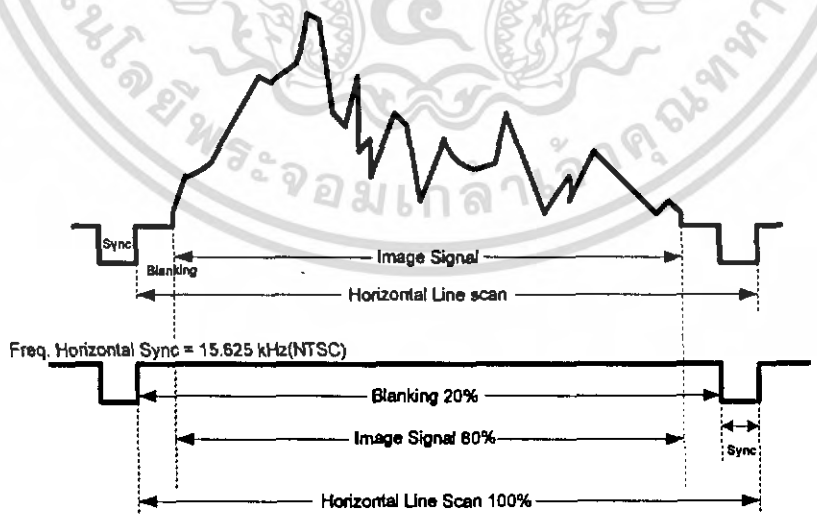
โดยทั่วไปแล้วจะถูกแบ่งง่ายๆ ออกเป็นสองชนิดคือ กล้องสี และกล้องขาวดำซึ่งขึ้นอยู่กับความต้องการ ของผู้ใช้และสถานที่ในการติดตั้งซึ่งต้องประกอบไปด้วยลักษณะของการใช้งานจริง ยกตัวอย่างเช่น

กล้องสี ควรใช้งานกับสถานที่ที่มีแสงสม่ำเสมอ เช่น ซุปเปอร์มาเก็ต, มินิมาร์ท, ร้านทอง เป็นต้น จากกลุ่มที่ยกตัวอย่างให้เห็นนั้นมีความเหมาะสมกล่าวคือ กล้องสีสามารถแยกแยะรายละเอียดหรือสีของสิ่งของได้ดี และในสถานที่ที่ยกตัวอย่างดังกล่าวก็มีการใช้แสงสว่างค่อนข้างมาก และสม่ำเสมอ ภาพที่มาปรากฏบนหน้าจอมอนิเตอร์ ก็จะมีความชัดเจน ซึ่งกล้องสีจะแบ่งออกได้เป็นสองชนิดคือ
กล้อง CCD(Charge Coupled Device) และกล้อง CMOS(Complementary Metal Oxide Semiconductor)

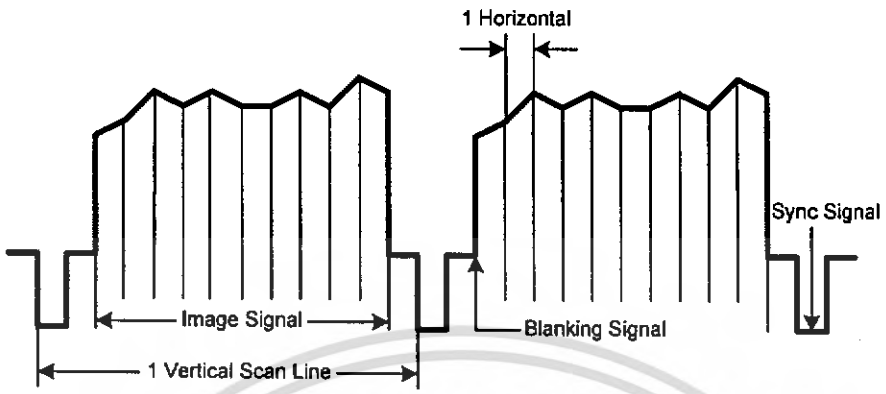
กล้องขาวดำ กล้องชนิดนี้เป็นกล้องที่ใช้แสงในการรับภาพต่ำมาก เหมาะอย่างยิ่งที่จะใช้งานในด้านการรักษาความปลอดภัยเนื่องจากสามารถดูในเวลากลางคืนได้ดีกว่ากล้องสี เหมาะสำหรับโรงงานอุตสาหกรรม เช่น ในอาคาร, คลังสินค้า, โรงงาน, กระบวนการผลิต, พื้นที่อันตราย, เคาเตอร์เก็บเงิน, ลานจอดรถ, ปั๊มน้ำมัน หรือสถานที่ที่ใช้อุปกรณ์ดูแลรักษาความปลอดภัย

2.2 ลักษณะของสัญญาณภาพ

สัญญาณภาพโดยทั่วไปจะมีลักษณะเป็นคอมโพสิทวีดีโอ ที่ประกอบไปด้วยข้อมูล สัญญาณภาพ (Image Signal), สัญญาณซิงค์(Sync Signal) และสัญญาณแบลงกิ้ง(Blanking Signal) โดยลักษณะของสัญญาณมีดังรูปที่ 2.1 และ 2.2

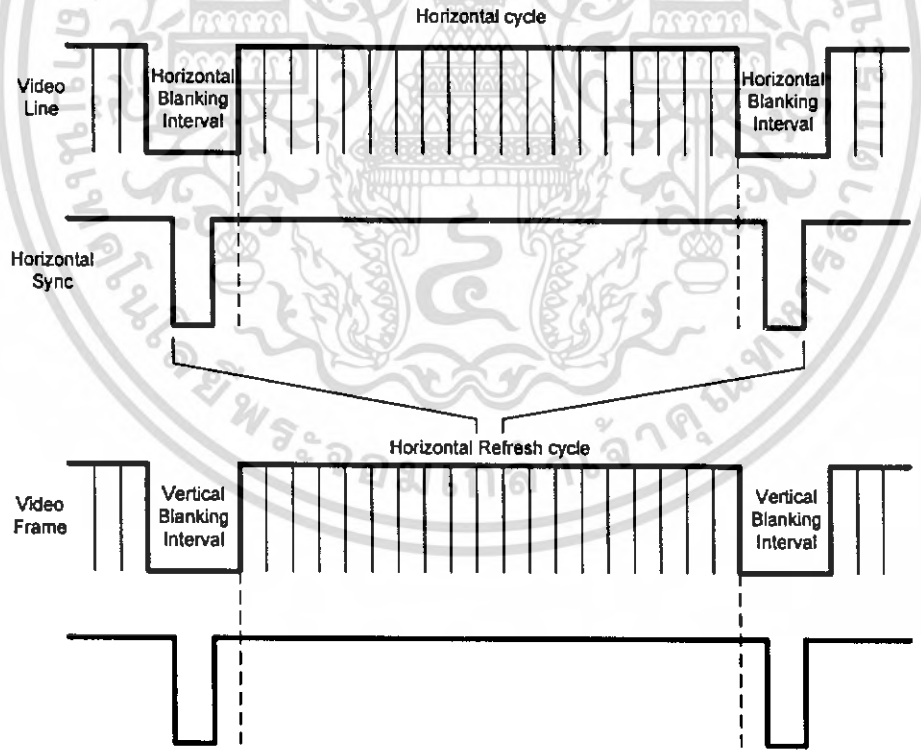


รูปที่ 2.1 ลักษณะของสัญญาณทางด้านแนวนอน



รูปที่ 2.2 ลักษณะของสัญญาณทางด้านแนวตั้ง

สัญญาณดังกล่าวจะถูกส่งไปยังมอนิเตอร์ ทำให้เกิดการสแกนที่หน้าจอมอนิเตอร์ซึ่งจะทำให้เกิดเป็นภาพขึ้นมา



รูปที่ 2.3 ความสัมพันธ์ระหว่างสัญญาณทางด้านแนวนอนและแนวตั้ง

2.2.1 องค์ประกอบของภาพ

ภาพนั้นมีองค์ประกอบมาจากจุดสีขาวและดำมากมาย มาเรียงกันประกอบขึ้นเป็นภาพ จุดเหล่านี้เองที่เรียกว่า เป็นองค์ประกอบของภาพ หรือ พิกเจอร์ อิลีเมนต์ หรือ พิกเซล ทำนองเดียวกัน ภาพที่ปรากฏทางจอภาพก็เอามาจากหลักการนี้ ภาพที่เกิดขึ้นบนจอภาพ หรือจอโทรทัศน์ประกอบด้วยเส้นขวางเล็กๆในแนวนอนเป็นจำนวนมากยิ่งภาพมีจำนวนเส้นมากเท่าไรรายละเอียดภาพก็จะยิ่งมากขึ้นเท่านั้น แต่ละเส้นนั้นมีทั้งส่วนที่ดำสนิด ส่วนที่จาง และส่วนที่สว่างร่วมกัน เส้นเหล่านี้เราได้มาจากการกวาดลำแสง(Scan) ความแตกต่างกันบนเส้นสแกนเหล่านี้เองที่จัดว่าเป็นองค์ประกอบของภาพ ซึ่งจากการทดสอบพบว่าภาพที่พอดูได้จะมีองค์ประกอบไม่ต่ำกว่า 200,000 พิกเซล

2.2.2 การสแกนภาพ

จุดที่เห็นสว่างในหลอดภาพของจอภาพ เกิดขึ้นเพราะ อิเล็กตรอนที่หลุดออกจากแคโทดและถูกดึงดูดให้วิ่งเป็นลำไปกระทบแอโนดหรือจอหลอดภาพ ซึ่งฉาบวัสดุเรืองแสงบางชนิดเอาไว้ การสแกนหลอดภาพมี 2 วิธี คือ

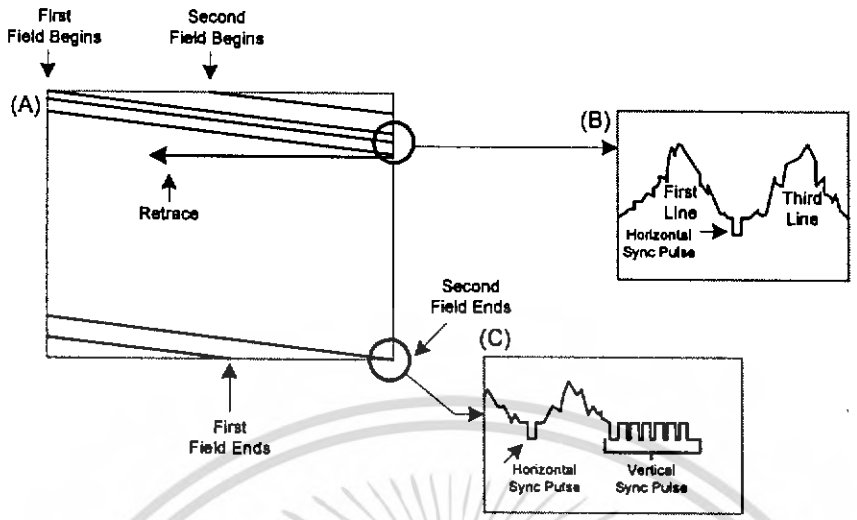
1. การสแกนแบบก้าวหน้า(Progressive scanning)
2. การสแกนแบบสลับเส้น(Interlaced scanning)

การที่จะให้การสแกนมีความต่อเนื่องขององค์ประกอบภาพจะต้องคำนึงถึงหลัก 3 ประการคือ

1. ลำอิเล็กตรอนที่กวาดในแนวนอน (Horizontal scanning) ในแต่ละครั้งจะต้องครอบคลุมองค์ประกอบภาพทั้งหมดของเส้นนั้นๆ

2. ในแต่ละเส้นของการสแกนลำอิเล็กตรอน ลำแสงจะต้องกวาดไปทางด้านซ้ายเพื่อเริ่มต้นเส้นภาพทางแนวนอนลำดับต่อไป เวลาของการสะบัดกลับเราเรียกว่า "รีเทรซ" หรือ "ฟลายแบ็ค" ในกรณีดังกล่าวจะต้องไม่มีข้อมูลภาพใดๆ เพราะว่าทั้งกล้องถ่ายและหลอดภาพจะเกิดอาการแบลงค์เอาท์ ในขณะนั้น

3. ในขณะที่เส้นสแกนสะบัดกลับมาเพื่อเริ่มต้นทางซ้ายใหม่ ตำแหน่งทางแนวตั้งต้องต่ำกว่าตำแหน่งเดิมเพื่อทำให้การสแกนเส้นต่อไปไม่ทับกัน ทั้งนี้โดยการควบคุมของสัญญาณใน แนวตั้ง(Vertical scanning)



รูปที่ 2.4 (A) จุดเริ่มต้นสแกนครั้งที่ 1 และครั้งที่ 2
รูปที่ 2.5 (B) สัญญาณสแกนในแนวตั้ง บนลงล่าง
รูปที่ 2.6 (C) สัญญาณสแกนในแนวนอน ซ้ายไปขวา

การสแกนที่ใช้ในเครื่องรับโทรทัศน์ ถึงแม้เราจะพบว่าหากให้มีการเรียงภาพเกินกว่า 16 ภาพต่อวินาทีแล้ว สายตาก็จะเห็นเป็นภาพต่อเนื่อง แต่แม้ว่าภาพที่เกิดขึ้นจะเกิด 24 ภาพต่อวินาทีแล้วก็ตาม ก็ยังมีการกระพริบ(Flicker) เกิดขึ้นเนื่องจากว่าในขณะที่การสแกนเริ่มจากขอบบนลงมาด้านล่าง เมื่อเส้นสแกนลงมาถึงขอบด้านล่าง ในความรู้สึกของมนุษย์แสงทางด้านบนจะเริ่มมืดกว่าด้านล่าง เวลาที่ลำแสงทำการสแกนวกกลับไปด้านบน ด้านล่างก็เกิดปัญหาเช่นเดียวกัน จะรู้สึกว่าเกิดแสงกระพริบหรือวูบวาบขึ้น จึงต้องใช้การสแกนสลับเส้น โดยครั้งแรกจะสแกนฟิลด์คู่(Even line trace) เป็นการสแกนแบบเส้นเว้นเส้น นั่นหมายถึงว่า การได้ภาพหนึ่งภาพ หรือที่เรียกว่าภาพหนึ่งเฟรม ต้องใช้การสแกนถึง 2 ครั้ง หรือ 2 ฟิลด์

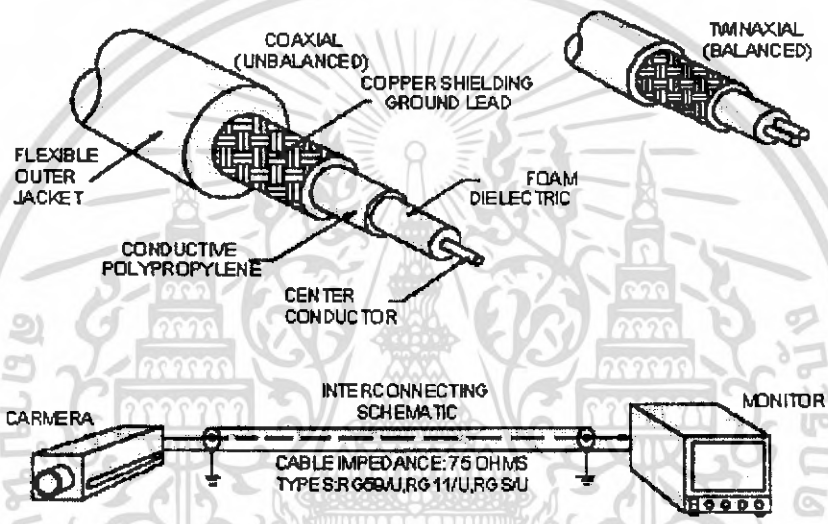
2.3 สายโคแอ็คเชียล

สายโคแอ็คเชียลจะใช้ในการเดินสายแบบระยะใกล้ และระยะกลางคือ หลายร้อยฟุตจนถึงหลายพันฟุต เพราะคุณสมบัติทางอิเล็กทรอนิกส์ของมัน เหมาะสมอย่างมากกับการส่งสัญญาณแบบ Full-Signal Bandwidth จากกล้องไปยังมอนิเตอร์ ซึ่งสัญญาณจากกล้อง CCTV (Closed Circuit Television) ประกอบไปด้วยความถี่สูงและความถี่ต่ำ สายโคแอ็คเชียลทุกชนิดสามารถส่งสัญญาณความถี่ต่ำได้ตั้งแต่ 20 เฮิร์ตถึง หลายพันเฮิร์ตในความเป็นจริงสายหลายชนิดสามารถใช้ในการสื่อสารทางโทรศัพท์ได้แต่จะใช้สายโคแอ็คเชียลชนิดพิเศษสำหรับฟูลสเป็กทรัมของความถี่ จาก 20 เฮิร์ตถึง 6 เมกกะเฮิร์ตโดยที่ไม่มีการถูกลดทอน ซึ่งเป็นสิ่งที่ต้องการสำหรับภาพที่มีคุณภาพสูง

สายโคแอ็คเชียลแบบพื้นฐานจะแบ่งออกเป็น 4 ชนิด เพื่อใช้ในระบบการส่งวีดีโอ

1. 75-โอห์ม Unblanced indoor cable
2. 75-โอห์ม Unbalanced outdoor cable
3. 124-โอห์ม Twin-axial balanced indoor cable
4. 124-โอห์ม Twin-axial balanced outdoor cable

โครงสร้างของสายโคแอ็คเชียลแบบ Twin-axial จะแสดงในรูปที่ 2.7



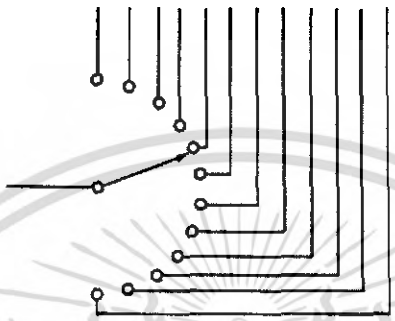
รูปที่ 2.7 โครงสร้างของสายโคแอ็คเชียล

2.4 ตัวแอนะล็อกมัลติเพลกซ์

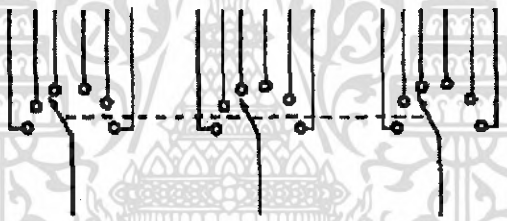
แอนะล็อกมัลติเพลกซ์ หรือเรียกอีกอย่างว่า สวิตช์ชิ่ง ที่อาจจะเรียกได้ว่าเป็นรูปแบบพื้นฐานทางอิเล็กทรอนิกส์ ที่มีหลายชนิดโดยเป็นอุปกรณ์ทางกลที่เป็นตัวเชื่อมต่อระหว่างเส้นทางที่อาจจะมีสองเส้นทางหรือมากกว่านั้นเพื่อควบคุมวงจรหนึ่งหรือหลายวงจรย่อยๆ โดยเส้นทางที่ได้รับการเชื่อมต่อจะเรียกว่า "Close" หรือ "On" ส่วนเส้นทางที่ไม่ได้รับการเชื่อมต่อจะเรียกว่า "Open" หรือ "Off"

สวิตช์นั้นจะสามารถแบ่งออกได้เป็นหลายชนิดมาก ซึ่งแต่ละชนิดก็จะมีรูปแบบการทำงานแตกต่างกันออกไป แต่สำหรับในโครงงานนี้เราจะใช้สวิตช์ที่มีชื่อว่า "โรตารี่ สวิตช์"

โรตารี่ สวิตช์ เป็นสวิตช์ที่อาศัยการทำงานของการหมุนของสไลด์ ซึ่งจะหมุนไปตามเลขของตัวที่ใช้สับสวิตช์และ ตำแหน่งที่ต้องการสับไปโดยจะควบคุมด้วยระบบดิจิทัล(Digitally Controlled Rotary Switches) และจะแสดงรูปโรตารี่ สวิตช์ที่มี 1 โพล์ 12 โทรว์(Single Pole Twelve Trow : 1P12T) ในรูปที่ 2.8 และ โรตารี่ สวิตช์ที่มี 3 โพล์ 6 โทรว์ (Three Pole Six Throw : 3P6T)ในรูปที่ 2.9



รูปที่ 2.8 โรตารี่ สวิตช์ที่มี 1 โพล์ 12 โทรว์ (Single Pole Twelve Trow : 1P12T)

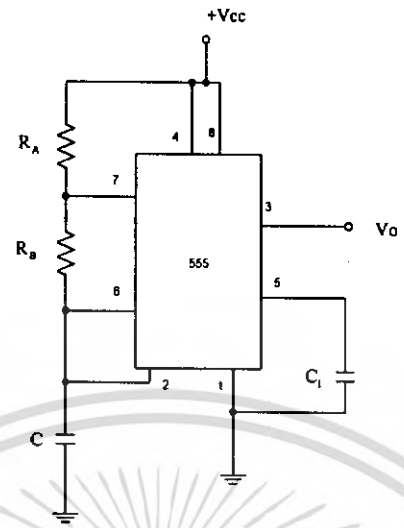


รูปที่ 2.9 โรตารี่ สวิตช์ที่มี 3 โพล์ 6 โทรว์(Three Pole Six Throw : 3P6T)

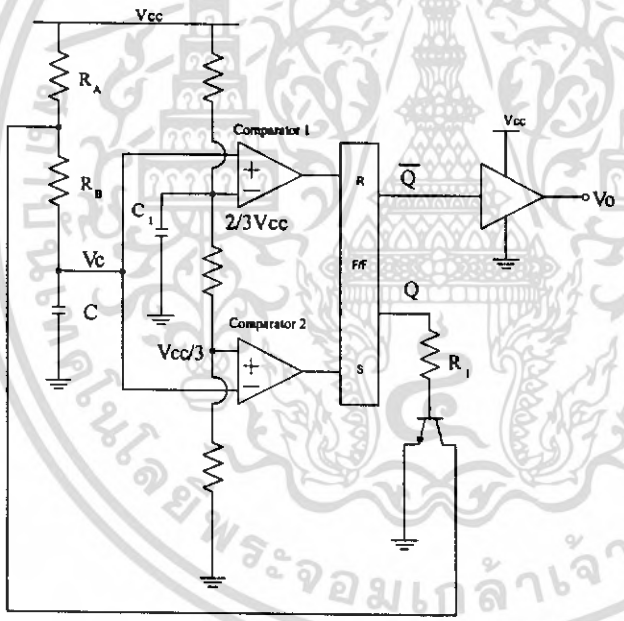
การสับสวิตช์ระหว่างช่องสัญญาณ ในขณะที่จะสับช่องสัญญาณไปอีกช่องสัญญาณหนึ่งต้องรอให้ช่องสัญญาณเดิมที่เชื่อมต่ออยู่ "Off" หรือ Disconnect เสร็จเรียบร้อยเสียก่อนจึงจะสามารถสับไปอีกช่องสัญญาณหนึ่งได้ ซึ่งช่วงเวลาของการสับระหว่างช่องสัญญาณก็จะใช้เวลาน้อยมากๆซึ่งขึ้นอยู่กับคุณสมบัติของตัวอุปกรณ์

2.5 วงจรอะสเตเบิลมัลติไวเบรเตอร์

เป็นวงจรมัลติไวเบรเตอร์ที่ทำงานได้ด้วยตัวเอง ไม่จำเป็นต้องอาศัยสัญญาณอินพุทมาขับให้วงจรทำงาน ทำหน้าที่คล้ายวงจรกำเนิดพัลส์ คือตัวมันเองสามารถทำงานและหยุดทำงานสลับกันตลอดเวลา บางครั้งจะเรียกว่า ฟรีรันนิงมัลติไวเบรเตอร์ สัญญาณเอาท์พุทจะเปลี่ยนสภาวะตลอดเวลา โดยจะไม่คู่เข้าสู่ค่าใดค่าหนึ่ง



รูปที่ 2.10 วงจรอะสเตเบิลมัลติไวเบรเตอร์



ตารางค่าความจริง

R	S	Q	$\overline{Q}$
0	0	Q^{n-1}	Q^{n-1}
0	1	0	1
1	0	1	0
1	1	1	1

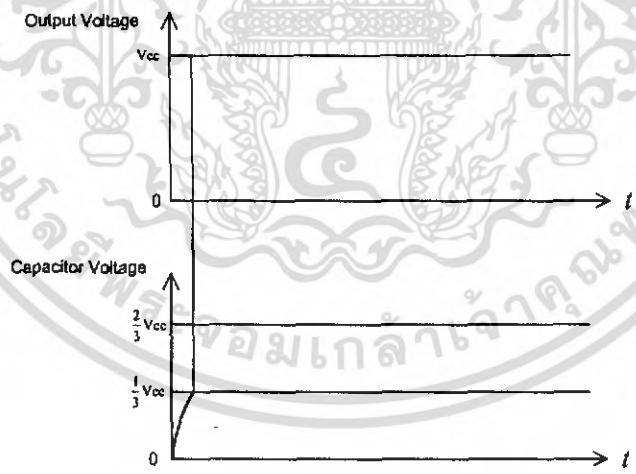
รูปที่ 2.11 วงจรอะสเตเบิลมัลติไวเบรเตอร์ที่แสดงวงจรภายในของ IC Timer 555

2.5.1 การทำงานของวงจรอะสเตเบิลมัลติไวเบรเตอร์

เริ่มแรกจะอธิบายวงจรเปรียบเทียบ ภายในไอซีก่อนโดย Comparator แต่ละตัวนั้น จะมีผลตอบสนอง ออกมาโดยการเปรียบเทียบระหว่างขาอินพุทบวกและลบตัว Comparator จะทำงาน 2 ช่วงคือ Saturate ด้านบวก และ Saturate ด้านลบ โดยตัวของมันจะมีค่าเกนอยู่ 1 ค่าซึ่งค่านั้นจะต้องเป็นค่าที่มากๆ และ

เมื่อทำการเปรียบเทียบระหว่างขาบวกและลบ แล้วทำให้เกิดผลลัพธ์ออกมาค่าหนึ่ง ซึ่งค่าค่านั้นเป็นบวกเมื่อนำมาคูณกับค่าเกนแล้วจะทำให้ได้ค่าที่มีค่าบวกมากๆ ซึ่งจะต้องเกินไฟเลี้ยง(Vcc) อย่างแน่นอน แต่ผลที่ออกมาจะมีค่าได้แค่เพียง Vcc เท่านั้น และถ้าค่านั้นเป็นลบ เมื่อนำมาคูณกับค่าเกน จะทำให้ได้ค่าที่เป็นลบมากๆ และเกินไฟเลี้ยงที่เป็นลบเช่นกัน ซึ่งในวงจรนี้ให้เป็นกราวน์ ผลที่ออกมาจึงเป็น 0V ผลที่ออกมาเป็น + Vcc และ 0 V นั้น เมื่อเข้ามาที่ RS Flip-Flop จะหมายถึงลอจิก 1 และ 0 ตามลำดับ ดังนั้นเพื่อให้ง่ายต่อการอธิบายต่อไป เมื่อ Voltage ที่เข้าขาบวกมีค่ามากกว่า Voltage ที่เข้าขาลบ ของ Comparator ผลตอบสนองจาก RS Flip-Flop จะเป็น 1 ตรงกันข้าม ถ้า Voltage ที่เข้าขาบวกมีค่าน้อยกว่าขาลบของ Comparator ผลตอบสนองจาก RS Flip-Flop จะเป็น 0

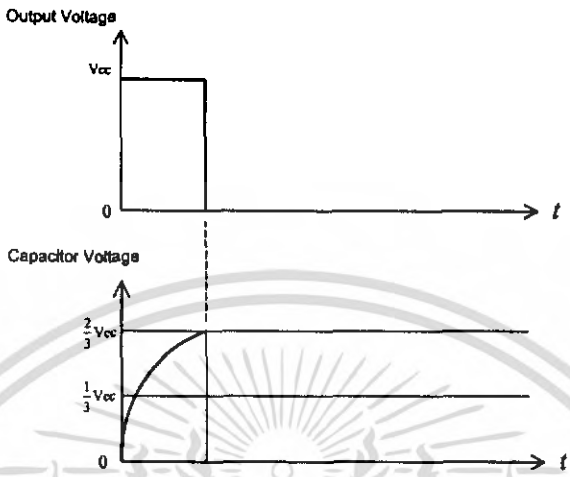
เนื่องจากตัวเก็บประจุ (Capacitor: C) ในวงจร ณ ที่เวลา $t < 0$ C จะต้องมีค่า Initial Condition โดยที่ไม่ทราบว่าค่านั้นคือค่าใด แต่เนื่องจากวงจรอะสเตเบิลมัลติไวเบรเตอร์จะทำงานด้วยตัวของมันเอง ดังนั้นเมื่อ $t < 0$ ไม่ว่าค่า Initial นั้นเป็นค่าเท่าใด มันก็จะทำงานต่อเนื่องไป เพราะฉะนั้น เพื่อให้ง่ายต่อการอธิบายและความเข้าใจ จะกำหนดให้ที่ $t = 0$, $Q = 1$ และ $\overline{Q} = 0$ และเมื่อ $t > 0$, $Vc = 0$ Voltageที่เข้าขาบวกของ Comparator I เป็น 0 V เมื่อเทียบกับ 2/3Vcc ที่ขาลบ จะเห็นได้ว่า ที่ขาบวก มีค่าน้อยกว่าที่ขาลบ ดังนั้นเอาท์พุทมีค่าลอจิกเป็น 0 และ Voltage ที่เข้าขาลบของ Comparator 2 เป็น 0 V เมื่อเทียบกับ Vcc/3 จะเห็นได้ว่าที่ขาบวก มีค่ามากกว่าที่ขาลบ ดังนั้นเอาท์พุทมีค่าลอจิกเป็น 1 $R = 0, S = 1$ จากตารางค่าความจริงจะได้ $Q = 0$, $\overline{Q} = 1$ ทำให้ C เริ่ม ชาร์จประจุ Vc มีค่าเพิ่มขึ้นจนมากกว่า Vcc/3 แต่ยังน้อยกว่า 2/3Vcc และเอาท์พุทที่ได้เป็น +Vcc ดังรูปที่ 2.12



รูปที่ 2.12 Wave Form แสดง Vo และ Vc ขณะ C ชาร์จประจุจนถึง Vcc/3

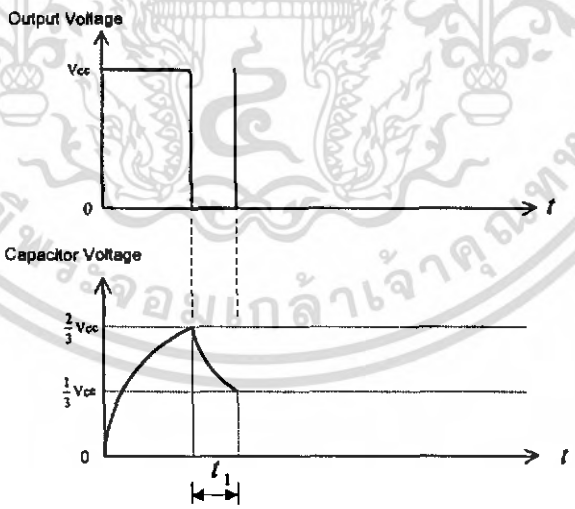
เมื่อ $Vc = Vcc/3$ ถึง 2/3Vcc เข้าขาบวกของ Comparator 1 เมื่อเทียบกับ2/3Vcc ที่ขาลบ จะมีค่าลอจิกเป็น 0 และเมื่อ Vc เข้าที่ขาลบ ของ Comparator 2 เทียบกับ Vcc/3 ที่ขาบวกจะมีค่าลอจิกเป็น 0 จาก

ตารางค่าความจริง R = 0, S = 0 Q และ $\overline{Q}$ จะคงสถานะเดิมคือ Q = 0, $\overline{Q}$ = 1 C จะชาร์จประจุต่อไปจน Vc มีค่ามากกว่า 2/3Vcc และ Vo = +Vcc จะได้ Wave Form ดังรูปที่ 2.13



รูปที่ 2.13 Wave Form แสดง Vo และ Vc ขณะ C ชาร์จประจุจนถึง 2/3Vcc

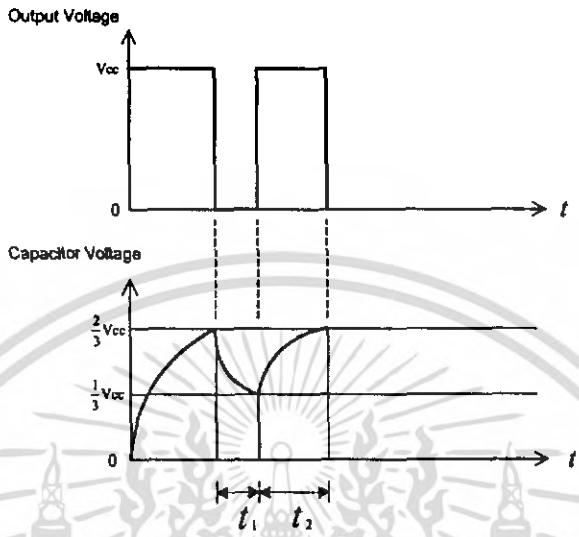
เมื่อ Vc > 2/3Vcc เข้าที่ขาบวกของ Comparator 1 เทียบกับ Vcc/3 ที่ขาลบ จะเห็นได้ว่าขาบวกมีค่ามากกว่าขาลบจะมีค่าลอจิกเป็น 1 และเมื่อ Vc เข้าที่ขาลบ ของ Comparator 2 เทียบกับ Vcc/3 ที่ขาบวกจะเห็นได้ว่าขาบวกมีค่าน้อยกว่า จะมีค่าลอจิกเป็น 0 R = 1, S = 0 จะได้ Q = 1 และ $\overline{Q}$ = 0 C จะคายประจุจน Vcc < Vcc/3 และ Vo = 0 V จะได้ Wave Form ดังรูปที่ 2.14 ที่มีช่วงเวลา t_1 เกิดขึ้น



รูปที่ 2.14 Wave Form แสดง Vo และ Vc ขณะ C คายประจุจนถึง Vcc/3

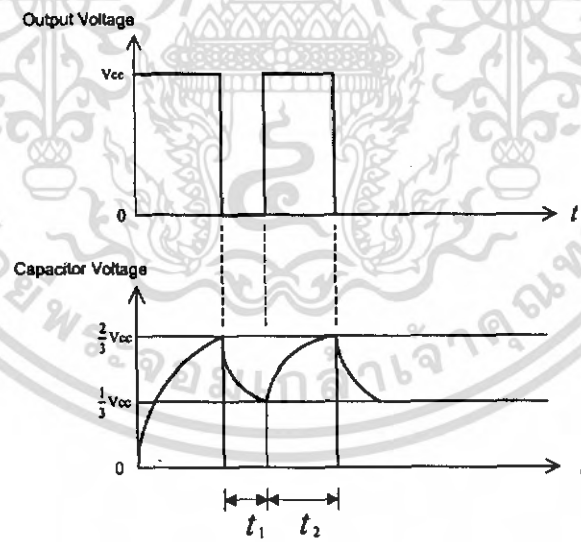
เมื่อ Vc < Vcc/3 เข้าที่ขาบวก ของ Comparator 1 เมื่อเทียบกับ 2/3Vcc ที่ขาลบจะเห็นได้ว่าขาบวกมีค่าน้อยกว่า จะได้ค่าลอจิกเป็น 0 และ Vc เข้าที่ขาลบของ Comparator 2 เมื่อเทียบกับ Vcc/3 ที่ขาบวกได้ว่าขา

บวกมีค่ามากกว่า จะได้ค่าลอจิกเป็น 1 ตัวเก็บประจุจะทำการชาร์จประจุอีกครั้ง และจะได้ Vo = +Vcc จะได้ Wave Form ดังรูปที่ 2.15 ที่มีช่วงเวลา t_2 เกิดขึ้น



รูปที่ 2.15 Wave Form แสดง Vo และ Vc ขณะ C ชาร์จประจุจนถึง Vcc/3

เมื่อ Vc > 2/3Vcc C ก็จะทำการคายประจุอีกครั้ง Vo ก็จะกลับเป็น 0V ด้วย ดังรูปที่ 2.16



รูปที่ 2.16 Wave Form แสดง Vo และ Vc ขณะ C คายประจุจนถึง Vcc/3

วงจรจะทำงานแบบนี้ซ้ำกันตลอดเวลา ซึ่งช่วงเวลา t_1 และ t_2 นี้จะไม่สมมาตรกัน โดย t_2 จะยาวนานกว่า t_1 และความไม่สมมาตรนี้สามารถแสดงได้ด้วยค่า Duty cycle ซึ่งจะแสดงสมการไว้ในภายหลัง

2.6 เทคโนโลยีเครือข่าย

การพัฒนาอย่างรวดเร็วของเทคโนโลยีด้าน Internet Protocol (IP) ทำให้อินเทอร์เน็ตซึ่งเป็นเครือข่ายคอมพิวเตอร์ที่มีแหล่งข้อมูลที่ใหญ่ที่สุดในโลกปัจจุบัน มีความจำเป็น สำหรับประชาชนทุกคนที่สามารถเข้าค้นหาข้อมูลและข่าวสารต่างๆ ที่ต้องการได้อย่างรวดเร็ว และยังรวมถึงการใช้ติดต่อสื่อสารระหว่างบุคคล และแหล่งข้อมูลต่างๆ ได้ทั่วทุกมุมโลกตลอดเวลา

IP Network หรือ โครงข่ายไอ-พี เป็นโครงข่ายมาตรฐานที่นิยมใช้ในการติดต่อสื่อสาร มีความทันสมัย และความเร็วสูง สามารถใช้รับส่งสัญญาณได้ทั้งเสียง ข้อมูล ภาพวีดีโอ และภาพเคลื่อนไหว โดยอยู่ในรูปสัญญาณดิจิทัลที่เรียกว่า ไอ-พี แพ็กเกจ โครงข่ายนี้รองรับการใช้งานร่วมกับโครงข่ายโทรคมนาคมใหม่ๆ และทันสมัยหลายแบบ ได้แก่ โครงข่ายใยแก้วความเร็วสูง แบบ SDH (Synchonous Digital Hierachy), โครงข่ายดิจิทัลสวิตช์ที่รองรับหลายความเร็วแบบ ATM (Asynchronous Transfer Mode), โครงข่ายใยแก้วความเร็วสูงแบบ Optical Internet Protocol โดยส่งผ่านสัญญาณที่ระดับความเร็วสูงตั้งแต่ 2 Mbps - 2.5 Gbps และพร้อมพัฒนาใช้งานร่วมกับอินเทอร์เน็ตแบบไร้สาย ในโครงข่าย โทรศัพท์เคลื่อนที่ยุคที่ 3 (Third - Generation Digital Mobile Communications Systems)

2.6.1 ความหมายของระบบเครือข่าย

ระบบเครือข่ายคอมพิวเตอร์ (Computer Network) หรือจะเรียกสั้นๆว่า ระบบเครือข่าย(Network) ประกอบไปด้วยเครื่องคอมพิวเตอร์หลาย ๆ เครื่องที่สามารถติดต่อกันเพื่อแลกเปลี่ยนข้อมูลกันได้ การติดต่อจะผ่านทางช่องการสื่อสารต่าง ๆ เช่น สายโทรศัพท์ สายไฟฟ้า หรือผ่านทางสื่อแบบอื่น ๆ ได้แก่ ไมเด็ม ไมโครเวฟ สัญญาณอินฟราเรด เป็นต้น ซึ่งโดยทั่วไปจะหมายถึง การที่เรานำเครื่องคอมพิวเตอร์อย่างน้อย 2 เครื่องมาเชื่อมต่อกัน วัตถุประสงค์ที่ต้องต่อกันนี้ มักเกิดจากความต้องการที่จะใช้ทรัพยากรของระบบร่วมกัน เช่น ใช้เนื้อที่เก็บข้อมูลในดิสก์ร่วมกัน ใช้งานเครื่องพิมพ์เลเซอร์ที่มีอยู่เครื่องเดียวร่วมกัน ต้องการส่งข้อมูลให้กับบุคคลอื่นในระบบไปใช้งาน หรือต้องการติดต่อสื่อสารระหว่างกัน เป็นต้น ฉะนั้น ระบบเครือข่าย คือ ระบบที่นำเอาเครื่องคอมพิวเตอร์ส่วนบุคคล (PC หรือ Personal Computer) แต่ละเครื่องมาต่อเชื่อมกันด้วยกลวิธีทางระบบคอมพิวเตอร์นั่นเอง

การแบ่งชนิดของระบบเครือข่าย สามารถดูได้จากลักษณะการติดตั้งใช้งานทางภูมิศาสตร์ จึงสามารถแบ่งระบบเครือข่ายได้เป็น 3 แบบ ด้วยกัน คือ

1. ระบบเครือข่ายระดับท้องถิ่น (Local Area Network หรือ LAN) เป็นระบบเครือข่ายที่ใช้งานอยู่ในบริเวณไม่กว้างนัก อาจใช้อยู่ภายในอาคารเดียวกันหรืออาคารที่อยู่ใกล้กัน เช่น ภายในมหาวิทยาลัย อาคารสำนักงาน คลังสินค้า หรือโรงงาน เป็นต้น การส่งข้อมูลสามารถทำได้ด้วยความเร็วสูง และมีข้อผิดพลาดน้อย ระบบเครือข่ายระดับท้องถิ่นนี้จึงถูกออกแบบมาให้ช่วยลดต้นทุนและเพื่อเพิ่มประสิทธิภาพในการทำงาน และใช้งานอุปกรณ์ต่าง ๆ ร่วมกัน

2. ระบบเครือข่ายระดับประเทศ (Wide Area Network หรือ WAN) เป็นระบบเครือข่ายที่ติดตั้งใช้งานอยู่ในบริเวณกว้าง เช่น ระบบเครือข่ายที่ติดตั้งใช้งานทั่วโลก โดยปกติมีอัตราการส่ง ข้อมูลที่ต่ำและมีโอกาสเกิดข้อผิดพลาด การส่งข้อมูลอาจใช้อุปกรณ์ในการสื่อสาร เช่น โมเด็ม มาช่วย

3. ระบบเครือข่ายระดับเมือง (Metropolitan Area Network หรือ MAN) เป็นระบบเครือข่ายที่มีขนาดอยู่ระหว่าง LAN และ WAN เป็นระบบเครือข่ายที่ใช้ภายในเมืองหรือจังหวัดเท่านั้น

2.7ระบบอ้างอิง โมเดล OSI

องค์กรมาตรฐานสากลหรือ ISO (International Organization for Standardization) ได้มีการศึกษาและหาแนวทางในการกำหนดมาตรฐานของเครือข่าย ซึ่งจะทำให้การเชื่อมต่อระหว่างเครือข่ายสามารถทำได้ ISO ค้นพบว่า ระบบเครือข่ายจะมีกิจกรรมพื้นฐาน ต่าง ๆ เช่น การรับส่งข้อมูล การเข้าใช้งานในเครือข่าย การสั่งพิมพ์ที่เครื่องพิมพ์ในเครือข่าย เป็นต้น ดังนั้น ISO ได้จัดแบ่งกิจกรรมเหล่านั้นออกเป็นงานย่อย และกำหนดเป็นโมเดลแบ่งเป็นชั้น ๆ ตามลำดับ เรียกว่า มาตรฐาน Open System Interconnection หรือ OSI ด้วยวิธีการแบ่งกิจกรรมที่ซับซ้อนในเครือข่ายออกเป็นงานย่อยๆ เพื่อจะช่วยทำให้การออกแบบและใช้งานเครือข่าย รวมถึงการเชื่อมโยงกันเป็นไปได้ด้วยความสะดวก และมีวิธีการทำงานอยู่ในกรอบเดียวกัน โมเดล OSI นี้เป็นต้นแบบแนวคิดในการสร้างเครือข่าย

ระบบย่อยของการสื่อสารข้อมูล เป็นส่วนที่มีความซับซ้อนมาก ไม่ว่าจะเป็นทางด้าน Hardware หรือ Software ในสมัยแรกๆ นั้น การพยายามสร้างโปรแกรม เพื่อการติดต่อสื่อสารนั้น จะสร้างโปรแกรมที่ค่อนข้างซับซ้อน ด้วยการมีส่วนประกอบหลายส่วนใน 1 โปรแกรม โปรแกรมประเภทนี้ โดยมากจะเขียนด้วยภาษา แอสแซมบลี ซึ่งผลมันก็คือ โปรแกรมเหล่านี้จะยุ่งยากในการทดสอบหรือปรับปรุงแก้ไข และเพื่อจะแก้ไขปัญหาที่เกิดขึ้นนี้ ISO จึงได้นำเอาวิธีการของระบบลำดับชั้น มาใช้ในเรื่องของ Reference Model ระบบย่อยการสื่อสาร ข้อมูลภายในเครื่องคอมพิวเตอร์จะถูกแบ่งออกเป็นลำดับชั้น หลาย ๆ ชั้น ซึ่งแต่ละชั้นก็จะทำหน้าที่ซึ่งได้กำหนดไว้โดยเฉพาะ

2.7.1 OSI Reference Model

OSI Reference Model ถูกออกแบบมาให้ประกอบด้วย 7 ลำดับชั้น ได้แก่

2.7.1.1 Application Layer

เป็นชั้นบนสุดของโมเดลจะเป็นส่วนที่ทำการติดต่อระหว่างแอปพลิเคชั่นของเครือข่ายกับผู้ใช้ ตัวอย่างแอปพลิเคชั่น ของเครือข่ายก็เช่น ระบบ จดหมายอิเล็คทรอนิค การโอนถ่ายแฟ้มข้อมูล (File Transfer) การขอเข้าใช้ระบบคอมพิวเตอร์ในเครือข่าย(Host Terminal)การจัดแฟ้มข้อมูล ในลักษณะ ต่างๆ เป็นต้น นอกจากนั้นยังรวมถึงการบริการทางด้านการแลกเปลี่ยนเอกสารข้อความต่าง ๆ Application Layer จะทำ

หน้าที่จัดการเรื่องต่าง ๆ ของเครือข่ายตามที่ผู้ใช้ต้องการนั่นเอง โดยจะอยู่ในระดับบนที่ใกล้ชิดกับผู้ใช้ที่สุด การเข้าไปใช้บริการระดับ Application Layer โดยปกติจะใช้โดยการพิมพ์คำสั่งต่างๆ ผ่านทางระบบโปรแกรมควบคุมเครื่อง(Operating System หรือ OS) โดย OS จะถือว่าเป็นโปรแกรม ในการสื่อสารนั่นเอง การบริการของลำดับชั้นนี้ จะแสดงให้ผู้ใช้ได้เข้าใจในทันที โดยไม่ว่าการสื่อสารจะประสบความสำเร็จหรือไม่ และหากมีข้อผิดพลาดประการใด ผู้ใช้ก็ สามารถทราบได้จากข้อความที่แสดงออกมา นอกเหนือจากบริการด้านต่าง ๆ ที่กล่าวข้างต้นแล้ว ยังมีบริการอื่น ๆ ที่ Application Layer จัดให้ดังนี้ คือ

- การระบุระบบคอมพิวเตอร์ปลายทาง โดยอาจจะใช้วิธีการเรียกตามที่อยู่ก็ได้
- กำหนดว่าเครื่องปลายทางที่ต้องการติดต่อนั้น อยู่ในสถานะที่พร้อมสำหรับการสื่อสาร
- กำหนดอำนาจหน้าที่ ความสำคัญ ในการติดต่อสื่อสาร
- การร่วมตกลงในระบบการใส่รหัสเพื่อความปลอดภัยของข้อมูล
- การรับรองเครื่องคอมพิวเตอร์ที่ต้องการสื่อสารด้วยว่า อยู่ในสถานะที่ต้องการจะสื่อสารได้หรือไม่
- กำหนดและเลือกข้อมูลประเภทต่าง ๆ ที่จำเป็นต้องใช้เมื่อมีการสื่อสาร
- การร่วมตกลงเกี่ยวกับความรับผิดชอบ กรณีการกู้ข้อมูลที่เกิดความเสียหาย
- กำหนดข้อจำกัดต่าง ๆ เกี่ยวกับรูปแบบของข้อมูลที่จะใช้ส่ง เช่น รูปแบบตัวอักษรที่จะใช้ หรือ โครงสร้างของข้อมูล

2.7.1.2 Presentation Layer

เป็นชั้นที่มีการกำหนดหน้าที่ไม่ชัดเจนนัก และมีการนำไปใช้ไม่มาก หน้าที่หลัก คือ เป็นส่วนที่จัดรูปแบบและนำเสนอข้อมูลระหว่างการสื่อสาร ให้เป็นไปตามที่ต้องการ โดยมีการกำหนดรูปแบบการส่งข้อมูลสำหรับใช้ในการแลกเปลี่ยน รูปแบบที่จะมีการกำหนดไว้ใน 2 ลักษณะ ทั้งนี้ยังรวมไปถึงการจัดแปลงข้อมูลในรูปมาตรฐาน ASCII หรือ EBCDIC, การลดขนาดข้อมูล (Data Compression) การเข้ารหัสหรือถอดรหัส ข้อมูลเพื่อความปลอดภัยในการสื่อสารแต่ส่วนใหญ่แล้วแอปพลิเคชั่นจะเป็นตัวจัดการแทนได้

ตัวอย่างของ Presentation Layer ที่เข้าใจง่าย ๆ เช่น การพูดโทรศัพท์ระหว่างบุคคลที่พูดภาษาฝรั่งเศสและบุคคลที่พูดภาษาสเปน บุคคลทั้งสอง ใช้ล่ามในการแปรภาษา และภาษากลางที่ใช้ก็คือภาษาอังกฤษ ล่ามแต่ละฝ่ายก็จะแปรภาษาทางฝ่ายของตนเองออกเป็นภาษาอังกฤษ แล้วฝ่ายที่รับ ก็จะแปรภาษาอังกฤษนั้น กลับเป็นภาษาของตนเองอีกต่อหนึ่ง บุคคลผู้พูดทางโทรศัพท์ก็เปรียบเสมือนกับโปรแกรม 2 โปรแกรม ที่ต้องการสื่อสารกัน ตัวล่ามก็เปรียบเสมือนกับ Presentation Layer ในเครื่องคอมพิวเตอร์ทั้ง 2 ฝ่าย ภาษาฝรั่งเศสและภาษาสเปน เปรียบเสมือนกับ Abstract Data Syntax ส่วนภาษาอังกฤษเปรียบได้กับ Transfer หรือ Concrete syntax ทั้งนี้ต้องนึกเสมอว่า ในการสื่อสารข้อมูลนั้น จะต้องมีการใช้ภาษากลางอันเป็นที่เข้าใจกันโดยทั่วไปภาษาหนึ่งเสมอ นอกจากนั้นตัวกลางในการแปรภาษา จากต้นทางและปลายทางให้เป็นภาษากลาง ก็ไม่จำเป็น

จะต้องเข้าใจในความหมายของข้อมูลที่สื่อสารนั้นเสมอไป หน้าที่อีกประการหนึ่งของ Presentation Layer ก็คือ เรื่องความปลอดภัยของข้อมูลโดยมีการเข้ารหัสข้อมูล ก่อนส่งออกไป และเมื่ออีกฝ่ายหนึ่ง ได้รับข้อมูลแล้ว ก่อนนำมาใช้ก็จะมีการถอดรหัสตามรูปแบบตามที่ได้มีการกำหนดไว้ล่วงหน้าทั้งสองฝ่าย แม้ว่าวิธีการเข้ารหัสข้อมูลนี้จะมิได้มี การกำหนดไว้ในมาตรฐานใด ๆ เลยก็ตามแต่ก็ได้มีผู้นำมาใช้กันอย่างแพร่หลาย

2.7.1.3 Session Layer

เป็นชั้นที่จัดการในเรื่องของการสร้างการติดต่อแต่ละครั้ง หรือ Session ให้ระบุคอมพิวเตอร์ทั้งสองฝ่าย กล่าวง่าย ๆ คือ จะให้บริการแก่โปรแกรมสื่อสาร ได้จัดการรวบรวมข้อมูลต่าง ๆ ที่จะใช้ในการติดต่อสื่อสาร โดยทำหน้าที่ตั้งแต่เริ่มการติดต่อ ดูแลให้การส่งผ่านข้อมูลในการติดต่อครั้งนั้น ๆ เป็นไปได้โดยไม่มีปัญหา จนถึงการเลิกติดต่อเมื่อเสร็จงาน ซึ่ง Session Layer นี้จะรับผิดชอบเกี่ยวกับการจัดสรรช่องการสื่อสารระหว่าง Application Layer ทั้งสองฝ่ายให้สามารถสื่อสารกันได้จนเสร็จสมบูรณ์ โดยต้องอาศัยความช่วยเหลือจาก Presentation Layer นอกจากนั้นยังให้บริการด้านต่าง ๆ คือ

- ให้บริการจัดเงื่อนไขโต้ตอบข้อมูลซึ่งจะให้บริการทั้งในแบบส่งข้อมูลไม่พร้อมกัน (Half-Duplex) และแบบพร้อมกัน (Full-Duplex)
- การสื่อสารในทิศทางเดียวกัน ในการสื่อสารผ่านระบบเครือข่ายระยะไกล ถ้าหากเกิดข้อผิดพลาดนั้น ระหว่างการสื่อสารในจุดใดจุดหนึ่ง Session Layer จะถูกอนุญาตให้ผู้ใช้เลือกที่จะทำการรับ-ส่ง ข้อมูลใหม่อีกครั้งในเวลาใดก็ได้
- การรายงานเกี่ยวกับข้อผิดพลาดถ้าในระหว่างการสื่อสารเกิดข้อผิดพลาดที่ไม่สามารถแก้ไขได้ Session Layer จะทำการส่งสัญญาณเพื่อแจ้งให้ Application Layer รู้ถึงข้อผิดพลาดนั้น

2.7.1.4 Transport Layer

ทำหน้าที่ควบคุมปริมาณและรายละเอียดวิธีการรับส่งข้อมูลให้เป็นไปตามกำหนดที่ได้ตั้งไว้ และจัดการให้การเชื่อมโยงเครือข่ายเป็นไปอย่างราบรื่น Transport Layer เป็นชั้นสุดท้ายที่จัดการเรื่องของเส้นทางในการส่งข้อมูล และจัดการตรวจสอบความผิดพลาดของข้อมูล ซึ่งส่วนของ TCP (Transmission Control Protocol) ใน TCP/IP แบบที่ใช้ในอินเทอร์เน็ต จะทำงาน ที่ระดับชั้นนี้ Transport Layer จะทำหน้าที่เป็นตัวเชื่อมระหว่าง 2 Layer คือ Application - Oriented Layer ซึ่งอยู่เหนือกว่า กับ Network - Dependent Procol Layer ซึ่งอยู่ต่ำกว่า และ มีหน้าที่ในการเตรียมข้อความต่าง ๆ ในการสื่อสารแบบ Session Layer

2.7.1.5 Network Layer

ทำหน้าที่ควบคุมวิธีการส่งผ่านข้อมูลระหว่างเครือข่ายให้ถูกต้อง และเป็นไปตามเส้นทางที่กำหนด ทำหน้าที่ในการเชื่อมต่อเส้นทางการสื่อสารระหว่าง Transport Layer ของเครือข่ายต้นทางและปลายทาง โดยจะจัดคำสั่งการทำงานเกี่ยวกับการค้นหาที่หมายปลายทางและควบคุมการไหลของข้อมูลในการเชื่อมต่อระหว่างเครื่องคอมพิวเตอร์กับเครือข่าย และในกรณีติดต่อระหว่างเครือข่าย Network Layer ก็จะมีการจัดเตรียมคำสั่งการทำงาน เพื่อทำให้การติดต่อเป็นไปอย่างราบรื่นสมบูรณ์ในปัจจุบันเมื่อมีการใช้เครือข่ายมากขึ้น ขนาดใหญ่ขึ้น ซับซ้อนมากขึ้น จะมีการจัดการแบ่งเครือข่ายนั้นให้เป็นส่วนย่อยหรือ Network Segment หรือ เรียกว่า Subnetwork โดยใช้อุปกรณ์ Bridge หรือ Router ทั้งนี้ Network Layer จะจัดส่งผ่าน Packet ข้อมูล คือ ข้อมูลที่ถูกจัดให้อยู่ใน รูปที่กำหนดไว้แล้วนั่นเอง ผ่านอุปกรณ์ต่างๆ ไปยังเครือข่ายย่อยให้อย่างถูกต้องตามที่ต้องการนอกจากนี้ Network Layer จะจัดการดูแลเส้นทางในการส่งข้อมูล (Routing Table) และกั้นหรือกรอง Packet ข้อมูลที่ส่งไปยังที่หมายภายในเครือข่ายย่อยเดียวกัน ไม่ให้ข้ามไปยังเครือข่ายย่อยอื่น ซึ่งจะช่วยลดปริมาณข้อมูลที่วิ่งบนเครือข่ายได้ส่วนหนึ่ง

2.7.1.6 Data Link Layer

จะจัดเตรียมข้อมูลเกี่ยวกับการเชื่อมต่อให้กับ Network Layer และจะทำหน้าที่เรียกใช้หรือกำหนดช่องทางในการส่งข้อมูลที่ถูกต้อง เช่น Ethernet , Token Ring หรือ FDDI (Fiber Distributed Data Interface) เป็นต้น รวมถึงการควบคุมลำดับและอัตราการรับส่งข้อมูลและสถานที่ ที่จะส่งข้อมูลไป ทั้งนี้ Data Link Layer เป็นชั้นแรกที่จัดการแปลงข้อมูลจากบิตให้อยู่ในรูปของ Packet โดยจะมีการเพิ่มข้อมูลเพื่อตรวจสอบความถูกต้องในกรณีส่งข้อมูลออกไป หรือในกรณีอ่านข้อมูลเข้ามาก็จะตรวจสอบส่วน Checksum เพื่อดูว่าข้อมูลที่ได้รับมาถูกต้องครบถ้วน และถ้าได้รับ Packet ข้อมูลที่ไม่ถูกต้องก็จะไม่เอาข้อมูลนั้นไปใช้งานต่อ และบอกไปยังต้นทางให้ส่งมาใหม่ ตัวอย่างเช่น คำสั่งในการตรวจสอบข้อผิดพลาด หรือมีข้อผิดพลาดแล้วต้องส่งข้อมูลใหม่ ก็จะมีคำสั่งในการส่งข้อมูลซ้ำ โดยปกติแล้วจะมีการให้บริการ ใน 2 รูปแบบ ได้แก่

1. Connection Less ถ้าหากมีการตรวจสอบและพบข้อผิดพลาดก็จะตัดการสื่อสารทันที
2. Connection Oriented ให้บริการในลักษณะพยายามไม่ให้เกิดข้อผิดพลาดใด ๆ ทั้งสิ้น

2.7.1.7 Physical Layer

จะให้บริการเกี่ยวกับวิธีการส่งข้อมูลเป็นบิต ระหว่างอุปกรณ์ 2 ชนิด ได้แก่ เครื่องมือและอุปกรณ์ทางฝ่ายผู้ใช้ และอุปกรณ์ของเครือข่ายโดยข้อมูลต่าง ๆ จะถูกส่งให้กับ Data Link Layer นอกจากนั้นยังรับผิดชอบดูแลในรายละเอียดการส่งข้อมูลในด้านฮาร์ดแวร์จริง เช่น การควบคุม Network Interface Card การส่ง

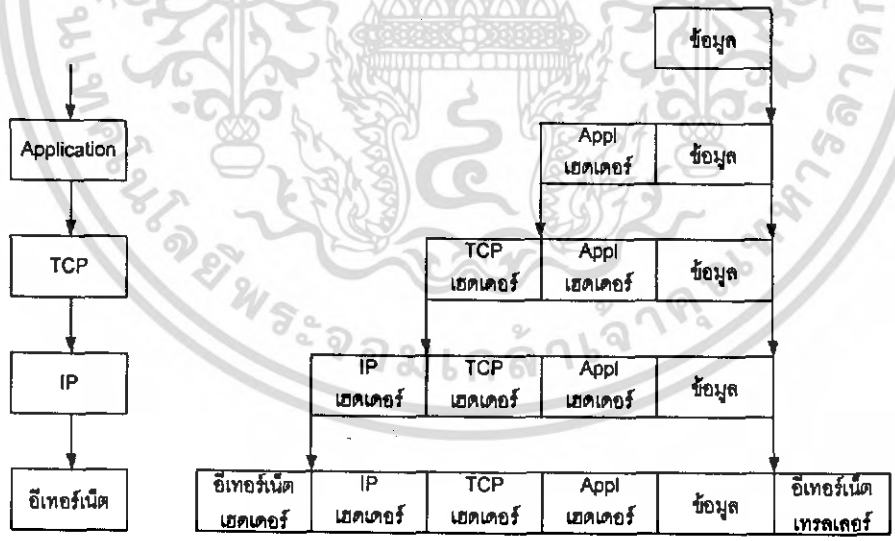
สัญญาณผ่านสายสัญญาณแบบต่าง ๆ การเชื่อมต่อเข้าเครือข่ายแบบต่าง ๆ โดย Physical Layer จะจัดสัญญาณทางไฟฟ้า สัญญาณ เสียง หรือสัญญาณแสงที่จำเป็นในการสื่อสารโดยตรง

2.8 การส่งถ่ายข้อมูลระหว่างชั้น

โปรโตคอลในแต่ละชั้นล้วนมีหน้าที่เกี่ยวข้องในการส่งผ่านข้อมูลจากสถานีต้นทางไปยังสถานีปลายทาง ข้อมูลจะถูกส่ง ผ่านจากโปรโตคอลระดับบนสุดจากสถานีต้นทางไปยังระดับล่างจนกระทั่งข้อมูลถูกแปลงให้อยู่ในรูปของสัญญาณไฟฟ้าแล้วเดินทางผ่านสถานีเครือข่ายไปยังสถานีปลายทาง โปรโตคอลระดับล่างสุดที่สถานีปลายทางจะรับสัญญาณและส่งผ่านขึ้นไปยังโปรโตคอลระดับบนต่อไป

เมื่อข้อมูลผ่านแต่ละระดับชั้น โปรโตคอลในชั้นนั้นจะผนวกข่าวสารกำกับการทำงานประจำโปร - โตคอลซึ่งเรียกว่าโปรโตคอลเฮดเดอร์ เข้ากับข้อมูลเฮดเดอร์และตัวข้อมูลจากระดับบนจะถูกส่งผ่านไปยังระดับล่าง โปรโตคอลระดับล่างจะมองเฮดเดอร์และตัวข้อมูลรวมเป็นเสมือนข้อมูลและเพิ่มเฮดเดอร์ประจำชั้นเข้าไปข้อมูลเดิมจึงมีแค่เฮดเดอร์หุ้มเป็นชั้นๆ กระบวนการนี้เรียกว่า การเอ็นแคปซูเลต ตัวอย่างในรูปที่ 2.17 การแสดงเอ็นแคปซูเลตแพ็กเก็ต TCP/IP ในอีเทอร์เน็ต

เมื่อสถานีปลายทางได้รับแพ็กเก็ตก็จะดำเนินการส่งไปตามลำดับชั้น โปรโตคอลประจำชั้นจะทำการถอดเฮดเดอร์ออกและส่งส่วนที่เหลือไปยังชั้นถัดไป เฮดเดอร์จะถูกถอดออกเหลือเฉพาะข้อมูลเมื่อถึงชั้นบนสุด กระบวนการนี้เรียกว่าการดีแคปซูเลต



รูปที่ 2.17 การห่อหุ้มข้อมูลตามลำดับโปรโตคอลแสตค

2.9 เลขหมายอินเทอร์เน็ต(IP Address)

หมายเลข IP เป็นหมายเลขขนาด 32 บิต ที่ใช้ระบุอุปกรณ์สื่อสารใน อินเทอร์เน็ตโดยเลขนี้จะไม่ซ้ำกันเลยหมายเลข IP จะใช้ระบุอุปกรณ์สื่อสาร(Network Interface) ของเครื่องคอมพิวเตอร์ เครื่องคอมพิวเตอร์เครื่องหนึ่งอาจมี อุปกรณ์สื่อสารหลายตัวก็ได้ เช่น ที่ทำหน้าที่เป็น Gateway อาจมี LAN card อยู่ 2 ถึง 3 card หมายเลข IP มักนิยมเขียนในรูปเลขฐาน 10 สี่กลุ่มโดยใช้จุดคั่นแต่ละช่วงมีค่าตั้งแต่ 0 ถึง 255 เช่น 161.246.18.44 เป็นต้นแต่เวลาเครื่องคอมพิวเตอร์นำเลขเหล่านี้ไปประมวลผลมันจะมองเป็นเลขฐานสอง 32 บิต ต่อเนื่องกัน เลข IP นี้แบ่งออกเป็นสองส่วนคือส่วนที่ชี้เน็ตเวิร์ค กับ ส่วนที่ชี้เครื่อง แต่ละส่วนจะใช้กี่ บิตนั้นขึ้นอยู่กับ Class ของเลข IP นั้นๆ ซึ่ง Class ของเลข IP นั้นจะมีอยู่ 5 Class ด้วยกันคือ

1. Class A : ช่วงตั้งแต่ 0.0.0.0 ถึง 127.255.255.255
2. Class B : ช่วงตั้งแต่ 128.0.0.0 ถึง 191.255.255.255
3. Class C : ช่วงตั้งแต่ 192.0.0.0 ถึง 223.255.255.255
4. Class D : ช่วงตั้งแต่ 224.0.0.0 ถึง 239.255.255.255
5. Class E : ช่วงตั้งแต่ 240.0.0.0 ถึง 247.255.255.255

เนื่องจากเลขหมายอินเทอร์เน็ตมีจำกัดจึงจำเป็นต้องมีการจำกัดบางหมายเลขเพื่อใช้เป็นการภายในเพื่อให้ผู้ที่ต้องการใช้ เลขหมายอินเทอร์เน็ต ที่ไม่ต้องการติดต่อกับเครือข่ายภายนอกได้ใช้ ซึ่งได้แก่

Class A: ช่วงตั้งแต่ 10.0.0.0 ถึง 10.255.255.255

Class B: ช่วงตั้งแต่ 172.16.0.0 ถึง 172.31.255.255

Class C: ช่วงตั้งแต่ 192.168.0.0 ถึง 192.168.255.255

2.10 การแบ่ง Subnet

เราสามารถแบ่งหนึ่ง เน็ตเวิร์ค ของเราออกเป็นโครงข่ายย่อยๆ โดยเรามักจะทำเมื่อ IP ของเราสามารถอ้างอิงเครื่องภายในหนึ่งเน็ตเวิร์คได้มากกว่าความต้องการจริงๆ ของเรามากๆ

วิธีการแบ่ง Subnet นั้นเราจะอาศัยความรู้จากเรื่องของ เลขหมายอินเทอร์เน็ตที่ว่าเลข IP แบ่งออกเป็นสองส่วนคือ ส่วนชี้เน็ตเวิร์คกับส่วนชี้เครื่อง ในเมื่อจำนวนบิต ที่เราใช้ในการชี้เครื่องมากเกินไป เราก็ทำการลดจำนวนโดยนำไปเพิ่มให้กับส่วนใช้ชี้เน็ตเวิร์คทั้งหมดที่กล่าวมาจะสามารถทำได้โดยการใช้เน็ตเวิร์ค ซึ่งเป็นเลข 32 บิต เหมือนเลขหมายอินเทอร์เน็ตมีหน้าที่ระบุขอบเขตของเลข IP ว่าบิตใดใช้ชี้เน็ตเวิร์คและ บิต ใด

ใช้ชี้เครื่อง ปกติเน็ตเวิร์ค Class A,B,C จะมี เน็ตมาร์ค ที่ตายตัวอยู่แล้วคือ 255.0.0.0 , 255.255.0.0 และ 255.255.255.0 ตามลำดับวิธีการใช้ เน็ตมาร์คนั้นเราจะนำมา And กับ IP Address ผลที่ได้จากการ And คือ หมายเลข เน็ตเวิร์ค เช่น 161.246.18.44 จะมี เน็ตมาร์ค มาตรฐานเป็น 255.255.0.0 (เพราะเป็น IP Class B) เมื่อนำมา And กันจะได้ 161.246.0.0 ซึ่งก็คือหมายเลข เน็ตเวิร์ค นั้นเอง บิต ที่ถูกตัดทิ้งไปก็คือหมายเลขเครื่อง

เราสามารถทำการเปลี่ยนแปลง เน็ตมาร์ค มาตรฐานได้ โดยการเพิ่มจำนวน บิต ที่ใช้ชี้ เน็ตเวิร์ค นั้น คือเราจะเพิ่มจำนวน บิต ที่เป็น 1 ของ เน็ตมาร์ค เช่นเราอาจเพิ่ม บิต ที่ใช้ในการชี้เน็ตเวิร์คของ IP Class B ขึ้นอีก 8 บิต โดยดึงมาจากบิต ที่เคยใช้ในการชี้เครื่อง ในกรณีนี้เราจะได้ เน็ตมาร์คเป็น 255.255.255.0 ทำให้เมื่อนำไป And กับ IP 161.246.18.44 จะได้ 161.246.12.18.0 เป็นหมายเลข เน็ตเวิร์ค จะเห็นว่าจะได้ เน็ตเวิร์ค เพิ่มขึ้นมาเป็น 256 เน็ตเวิร์ค และ แต่ละ เน็ตเวิร์ค มีหมายเลขเครื่อง 254 เครื่อง

2.11 Java Programming

2.11.1 สถาปัตยกรรมของ Java

สถาปัตยกรรมของ Java ประกอบด้วยส่วนสำคัญ 4 ส่วน คือ

1. Java programming language คือโปรแกรมที่เขียนขึ้นโดยใช้ภาษาจาวา อยู่ในรูปเทกซ์ เรียกว่า ซอร์สโค้ด
2. Java Class file คือ ซอร์สโค้ดที่ถูกแปลงคอมไพล์แล้ว
3. Java API (Application Programming Interface)

- API คือโค้ดที่คอมไพล์แล้ว ซึ่งช่วยให้โปรแกรมเข้าถึง ในส่วนของ System services ของระบบปฏิบัติการ

- Java API คือ กลุ่มของ Ready-made software components โดยรวบรวมเป็นไลบรารีของคลาส และอินเตอร์เฟสที่สัมพันธ์กันในรูปของแพ็คเกจที่สามารถนำมาใช้ในโปรแกรมของเราโดยไม่ต้องเขียนขึ้นเองซึ่งมีความยากและซับซ้อนยิ่ง

4. JVM (Java Virtual Machine) คือส่วนที่จะติดต่อสั่งงานโดยตรงต่อคอมพิวเตอร์

2.11.2 Graphical User Interface

GUI (Graphical User Interface) คือ ส่วนของโปรแกรมที่ใช้ติดต่อกับผู้ใช้โปรแกรม ซึ่งเป็นการเขียนโปรแกรมที่พยายามให้ผู้ใช้สามารถสื่อสารและทำงานกับโปรแกรมได้อย่างมีประสิทธิภาพ

AWT(Abstract Windowing Toolkit) เป็นเครื่องมือพื้นฐานในการเขียนโปรแกรมด้วย GUI ใน Java

Swing เป็น API ตัวหนึ่งใน Java Foundation Class ซึ่งเป็นแพคเกจที่พัฒนาขึ้นมาเพื่อช่วยสนับสนุนการทำงานของ AWT ให้ดีขึ้น ซึ่งในการเขียนโปรแกรมจะใช้ทั้ง ATW และ Swing ทำงานร่วมกัน โดยนำสิ่งที่ดีของทั้ง 2 มาใช้

โครงสร้างของ Swing Component

- Top Level Containers คือ Containers ระดับบนสุดสำหรับ Swing Component ประกอบด้วย JFrame, JApplet, JDialog, JWindow

- General Purpose Containers คือ Containers ระดับกลาง สำหรับไว้ใช้โดยทั่วไปในงานต่างๆ ประกอบด้วย JPanel, JScrollPane, JSplitPane, JTabbedPane, JToolBar

- Special Purpose Containers คือ Containers ระดับกลางสำหรับไว้ใช้ในงานเฉพาะ ประกอบด้วย JInternalFrame, JLayeredPane, JRootPane

- Basic Controls คือ ส่วนที่ใช้ติดต่อกับผู้ใช้โดยการรับข้อมูลมาจากผู้ใช้ ประกอบด้วย JButton, JCheckBox, JComboBox, JList, JTextField, JMenu, JSlider

- Uneditable Information Displays คือ ส่วนที่ใช้สำหรับให้ข้อมูลต่อผู้ใช้ และไม่สามารถเปลี่ยนค่าได้โดยผู้ใช้ ประกอบด้วย JToolTip, JLabel, JProgressBar

- Editable Displays of Format Information คือ ส่วนที่ให้ข้อมูลต่อผู้ใช้ และสามารถเปลี่ยนค่าได้โดยผู้ใช้ ประกอบด้วย JText, JTree, JTable, JFileChooser, JColorChooser

ทุกๆโปรแกรมที่ใช้ Swing Component จะต้องมี Containers ที่เรียกว่า Top-level container อย่างน้อยหนึ่งอย่าง เพื่อใช้ในการวาดภาพ Component ต่างๆลงบนจอ และจัดการเหตุการณ์ต่างๆ โดยแต่ละ Top level container จะต้องมี Content pane (ซึ่งเรียกใช้โดย getContentPane method) เป็นที่สำหรับวาง Visible component ต่างๆและเป็นที่กำหนด Layout Manager โดยจะไม่ใส่ค่าแต่ละ Component เช่น JButton, JLabel เป็นต้น โดยตรงไปที่ Top level container แต่จะนำไปใส่ที่ Content pane

2.11.3 Java Media Framework

JMF(Java Media Framework) ถูกออกแบบมาเพื่อใช้ในการติดต่อโดยประยุกต์ใช้โปรแกรมสำหรับสื่อข้อมูลที่ขึ้นอยู่กับเวลา (Time-based media) ในงานประยุกต์ของจาวา (Java applications) และ แอพเพลต

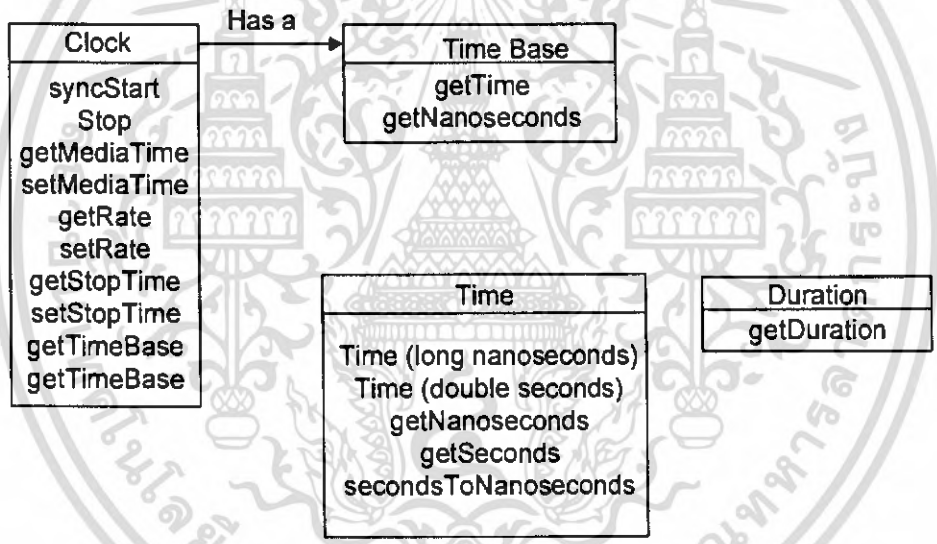
สื่อข้อมูลที่ขึ้นอยู่กับเวลามีลักษณะที่สำคัญคือ การรับส่งและการวิเคราะห์ที่มีเวลาเข้ามาเกี่ยวข้องโดยการเริ่มต้นจากการไหลของข้อมูลทั้งในรูปแบบการรับและการแสดงข้อมูล สื่ออาจนำมาจากไฟล์ในดิสก์ไดร์ฟหรืออาจมาจากเน็ตเวิร์ค หรือออกมาจากอุปกรณ์รับสื่อเช่นกล้องหรือไมโครโฟน เป็นต้น สื่อมักจะแบ่งออกเป็นหลายช่องสัญญาณของข้อมูลหรือเรียกว่า แทร็ก ยกตัวอย่างเช่นไฟล์ๆหนึ่งอาจจะประกอบด้วยแทร็กของออดิโอและแทร็กของวีดีโอ สื่อสามารถประกอบไปด้วยแทร็กหลายๆแทร็ก

2.11.3.1 การทำงานของ JMF

JMF มีสถาปัตยกรรมที่รวมวิธีหลายๆอย่างในการจัดการกับสิ่งที่รับมา กระบวนวิธีในการโพรเซสและส่งสื่อที่ขึ้นอยู่กับเวลา JMF สนับสนุนและรองรับการทำงานของสื่อหลายๆประเภท เช่น GSM, MPEG, Quick Time และ WAV เป็นต้น

JMF ยังรักษาคุณสมบัติของจาวาที่ว่า เขียนครั้งเดียวสามารถนำไปรันที่ไหนก็ได้ เพื่อเป็นประโยชน์กับนักพัฒนาโปรแกรมที่ต้องการพัฒนาจาวาโปรแกรมเกี่ยวกับเสียงและวีดีโอ แบบจำลองทางเวลา และให้ความเที่ยงตรงทางเวลาอยู่ในระดับนาโนวินาที ส่วนที่เกี่ยวข้องกับเวลามักถูกนำเสนอโดยออบเจ็ค Time บางคลาสยังสนับสนุนการระบุเวลาในหน่วยนาโนวินาที

คลาสซึ่งสนับสนุนแบบจำลองทางเวลาของ JMF อิมพลีเมนต์มาจากอินเตอร์เฟส Clock เพื่อรักษาเวลาของแทร็กสำหรับสื่อที่เฉพาะเจาะจง อินเตอร์เฟส Clock นิยามพื้นฐานของเวลาและการจัดการ การซิงโครไนซ์ที่จำเป็นสำหรับการควบคุมการนำเสนอข้อมูลสื่อ



รูปที่ 2.18 แบบจำลองทางเวลาของ JMF

อินเตอร์เฟส Clock ใช้อินเตอร์เฟส TimeBase เพื่อรักษาเวลาของแทร็กในขณะที่ถูกนำเสนอ อินเตอร์เฟส TimeBase ให้เวลาที่คงที่คล้ายๆกับคริสตอลออสซิเลเตอร์ในนาฬิกา ข้อมูลเพียงอย่างเดียวที่อินเตอร์เฟส TimeBase ให้คือเวลาในปัจจุบันของมัน ซึ่งเรียกว่า เวลา TimeBase เวลา TimeBase ไม่สามารถหยุดหรือตั้งค่าใหม่ได้ เวลา TimeBase ขึ้นอยู่กับนาฬิกาของระบบ

ออบเจ็ค Clock ของเวลาของสื่อแสดงถึงตำแหน่งปัจจุบันภายในสื่อ จุดเริ่มต้นของสายสื่อคือเวลาของสื่อเท่ากับศูนย์ จุดสิ้นสุดของสายสื่อคือเวลาสื่อมากที่สุดของสาย Duration ของสายสื่อคือเวลาตั้งแต่เริ่มต้น

จนถึงจุดสิ้นสุด หรือคือความยาวของเวลาที่ใช้ในการนำเสนอสายสื่อ เพื่อที่จะรักษาแทร็คของเวลาสื่อปัจจุบัน อินเตอร์เฟส Clock ใช้สิ่งต่อไปนี้

- The time-base start-time คือ เวลาที่ TimeBase รายงานว่าการนำเสนอเริ่มต้น

- The media start-time คือ ตำแหน่งในสายสื่อซึ่งการนำเสนอเริ่มต้น

- The playback rate บอกว่า Clock วิ่งเร็วแค่ไหนเมื่อเทียบกับ TimeBase rate เป็นสเกลซึ่งนำมาใช้กับ TimeBase เช่น Rate 1.0 หมายถึง อัตราการเล่นที่ปกติสำหรับสายของสื่อในขณะที่ Rate 2.0 หมายถึงการนำเสนอที่เร็วกว่า 2 เท่าของอัตราปกติ เมื่อการนำเสนอเริ่มต้นเวลาของสื่อจะถูกแม็ปกับเวลา TimeBase และประโยชน์ของเวลา TimeBase ใช้สำหรับวัดระยะของเวลา ระหว่างการนำเสนอเวลาสื่อปัจจุบันสามารถคำนวณได้จากสูตร

MediaTime = MediaStartTime + Rate(TimeBaseTime-TimeBaseStartTime)

เมื่อการนำเสนอหยุด เวลาสื่อหยุดแต่เวลา TimeBase ยังคงเดินต่อไปเพื่อใช้ประโยชน์ ถ้าการนำเสนอเริ่มต้นใหม่เวลาของสื่อจะแม็ปอีกครั้งกับเวลา TimeBase ปัจจุบัน

2.11.3.2 การจัดการ (Management)

JMF API ประกอบไปด้วยอินเตอร์เฟสที่สำคัญๆ ซึ่งนิยามพฤติกรรมและการติดต่อระหว่างออปเจ็คซึ่งใช้ในการจับโพรเซสและนำเสนอสื่อที่ขึ้นอยู่กับเวลา การอิมพลีเมนต์อินเตอร์เฟสเหล่านี้ดำเนิน การภายในโครงสร้างของเฟรมเวิร์กโดยใช้ออปเจ็คที่เรียกว่า Managers ซึ่ง JMF ใช้ Managers 4 อย่าง มีดังต่อไปนี้

- Managers จัดโครงสร้างของ Players, Processors, DataSources และ DataSinks

- PackageManager จัดการเกี่ยวกับการลงทะเบียนของ แพ็คเกจที่บรรจุในคลาสของ JMF เช่น Players, Processors, DataSources และ DataSinks

- CaptureDeviceManager จัดการเกี่ยวกับการลงทะเบียนของอุปกรณ์จับที่มีอยู่

- PlugInManager จัดการเกี่ยวการลงทะเบียนของปลั๊กอินของ JMF ส่วนประกอบโพรเซส เช่น มัลติเพล็กเซอร์ ดีมัลติเพล็กเซอร์ โคเด็คเอฟเฟ็ก และ เรนเดอร์การเขียนโปรแกรมโดยใช้ JMFจำเป็นที่จะต้องใช้คลาส Manager นั้นสร้างเม็ธธอดเพื่อสร้าง Players, Processors, DataSources และ DataSinks

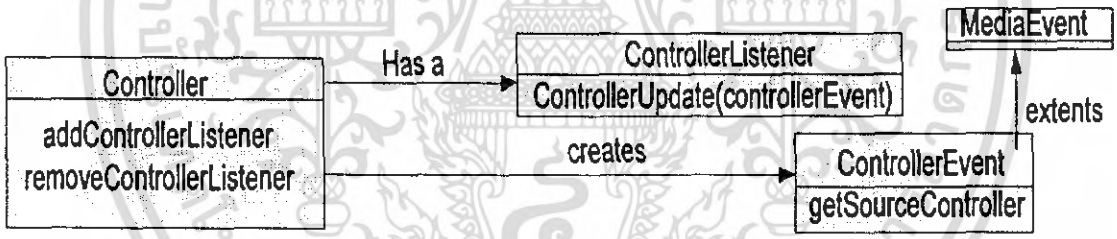
สำหรับงานประยุกต์ ถ้าจับข้อมูลสื่อจากอุปกรณ์อินพุทจะต้องใช้คลาส CaptureDeviceManager เพื่อจะหาว่ามี อุปกรณ์อะไรอยู่บ้างและเข้าถึงข้อมูลเกี่ยวกับมัน และถ้าต้องการที่จะควบคุมว่าจะใช้โพรเซสอะไรในการแสดงข้อมูลจะต้องค้นหาโดยใช้คลาส PlugInManager เพื่อที่จะให้แน่ใจว่าปลั๊กอินนั้นได้ลงทะเบียนหรือยัง

เมื่อต้องการที่จะสืบทอดหน้าที่ของ JMF โดยการเพิ่มเติมปลั๊กอินใหม่สามารถที่จะลงทะเบียนกับ PlugInManager เพื่อให้สามารถหาได้กับ Processors นั้นที่สนับสนุนปลั๊กอินเอพีไอเพื่อที่จะใช้ Player, Processor, DataSource, DataSink กับ JMF สามารถลงทะเบียนแพ็คเกจที่ต่างออกไปกับ PackageManager

2.11.3.3 แบบจำลองเหตุการณ์ (Event Model)

JMF ใช้เหตุการณ์ที่มีโครงสร้างรายงานกลไกที่ใช้ในโปรแกรม เพื่อแจ้งให้ทราบว่าสภาวะปัจจุบันของระบบสื่อ (Media system) และทำให้การโปรแกรมกับ JMF นั้นรองรับ ต่อเหตุการณ์ผิดพลาดที่เกิดขึ้นเช่น ไม่มีข้อมูลและแหล่งที่อยู่ไม่มีอยู่ เมื่อใดก็ตามที่ออปเจ็คของ JMF ต้องการที่จะรายงานสภาวะปัจจุบันจะโพสไปยัง MediaEvent MediaEvent เป็นสับคลาสที่เฉพาะเจาะจงกับแต่ละประเภทของเหตุการณ์ ออปเจ็คเหล่านี้จะสร้างรูปแบบของจาวาบีน สำหรับเหตุการณ์ แต่ละประเภทของออปเจ็คของ JMF ที่สามารถโพสไปยัง MediaEvents JMF ได้นิยามอินเตอร์เฟส Listener ที่รับผิดชอบทางด้านรับ เพื่อที่จะรับการแจ้งมาเมื่อ MediaEvents ถูกส่งต่อ สามารถอิมพลีเมนต์ อินเตอร์เฟส Listener ที่ถูกต้อง และ ลงทะเบียนคลาส ของ Listener กับออปเจ็คที่โพสเหตุการณ์โดยเรียก เม็ธธอด addListener

ออปเจ็ค Controller (เช่น Players และ Processors) และออปเจ็คที่ใช้ในการควบคุม เช่น GainControl จะโพสเหตุการณ์ของสื่อ (Media Events)



รูปที่ 2.19 แบบจำลองเหตุการณ์ของ JMF

2.11.3.4 แหล่งข้อมูล (Data Source)

แหล่งข้อมูลจะห่อหุ้มสายของมีเดียคล้ายๆ กับซีดีเพลงใน JMF ออปเจ็ค DataSource นำเสนอสื่อออดิโอ สื่อวีดีโอ หรือรวมกันทั้งสองอย่าง DataSource สามารถเป็นไฟล์หรือสายของข้อมูลที่มาจากอินเทอร์เน็ต สิ่งที่ดีสำหรับคลาสนี้คุณสามารถระบุที่ตั้งหรือโปรโตคอล DataSource จะห่อหุ้มทั้งที่ตั้งของสื่อโปรโตคอลและซอฟแวร์ที่ใช้ในการส่งสื่อ เมื่อ DataSource ถูกสร้างขึ้นมาแล้วจะถูกส่งนำไปให้ Player เพื่อนำมาแสดงผล ซึ่ง Player จะไม่สนใจว่า DataSource มีต้นกำเนิดมาจากที่ใด

ข้อมูลสื่อสามารถมาจากหลายๆ ที่ เช่น ไฟล์ที่มาจากเน็ตเวิร์คหรือเครือข่ายท้องถิ่น หรือการวัดค่าจากอินเทอร์เน็ต

2.11.3.5 Player

Player จะเอาอินพุทของออดิโอ หรือ วีดีโอ แล้วส่งไปยังลำโพงหรือหน้าจอเช่นเดียวเครื่องเล่นซีดีที่อ่านข้อมูลจากซีดีและส่งเสียงดนตรีไปยังลำโพง Player สามารถทำงานได้โดยอัตโนมัติ เพราะว่า Player มีการเตรียมตัวมันเองและ DataSource ของมัน ก่อนที่จะเริ่มเล่นสื่อ เพื่อที่จะเข้าใจถึงสิ่งนี้ลองใส่ซีดีเข้าไปในเครื่องเล่นซีดีของคุณและเล่นเพลงที่สี่และดูถึงผลที่ตามมาเมื่อเครื่องเล่นซีดียังไม่ได้เล่นเพลงอย่างแรกที่มันทำคือเริ่มหาแทร็กของเพลงที่สี่ที่จะเริ่มต้นและทำบางสิ่งอื่นเพื่อเป็นการเตรียมตัว หลังจากประมาณครึ่งวินาที (ขึ้นอยู่กับเครื่องเล่นซีดี) จะเริ่มได้ยินเสียงดนตรีเช่นเดียวกัน JMF Player ก็เตรียมตัวก่อนที่จะได้ยินเสียงออดิโอหรือได้ดูวีดีโอ ในวิธีการโดยทั่วไป Player จะมีลำดับของสภาวะไปจนถึงสภาวะสุดท้ายสภาวะของ Player ใน JMF นิยามไว้ 6 สภาวะดังนี้

Unrealized : ในสภาวะนี้ออปเจ็คของเพลเยอร์เพิ่งจะเริ่มต้นซึ่งไม่รู้สภาวะแวดล้อมของตัวเอง ไม่รู้อะไรเลยเกี่ยวกับสื่อของมัน

Realizing : Player เปลี่ยนจากสภาวะ Unrealized เป็นสภาวะ Realizing เมื่อ realize() ของ Player ในสภาวะ Realizing Player จะอยู่ในกระบวนการเลือกความต้องการของแหล่งของ Player

Realized : ซึ่งจะเปลี่ยนมาจากสภาวะ Realizing ในสภาวะนี้ Player จะรู้ว่าแหล่งอะไรที่มันต้องการและมีข้อมูลเกี่ยวกับประเภทของสื่อที่จะแสดง มันยังแสดงส่วนประกอบและการควบคุมที่มองเห็นได้อีกด้วยและมีการเชื่อมต่อไปยังออบเจ็คในระบบ

Prefetching : เมื่อเม็ธธอด prefetch() ถูกเลือก Player เข้าไปสู่สภาวะPrefetching สภาวะนี้ Player จะเตรียมตัวที่จะแสดงสื่อ Player จะโหลดข้อมูลสื่อและสิ่งอื่นที่จำเป็นในการเล่นข้อมูลสื่อ

Prefetched : สภาวะนี้จะเริ่มต้นเมื่อสิ้นสุดกระบวนการ Prefetching มันพร้อมที่จะเล่น

Started : สภาวะนี้จะเริ่มต้นเมื่อคุณเรียกเม็ธธอด start() Player ตอนพร้อมที่จะแสดงผลข้อมูลสื่อแล้ว

2.11.3.6 Processor

Processor เป็นอีกประเภทหนึ่งของ Player ในจีเอ็มเอฟเอพีไอ อินเตอร์เฟส Processor สืบทอดมาจาก Player ดังนั้น Processor จึงสนับสนุนการควบคุมการนำเสนอเช่นเดียวกับ Player แต่สิ่งที่ไม่เหมือนกับ Processor คือ Processor ควบคุมสิ่งที่กระบวนการถูกแสดงบนสายของสื่อ นอกจากนี้แล้วในการส่งแหล่งของข้อมูล Processor ยังแสดงเอาท์พุทของข้อมูลสื่อโดยผ่านทาง Data Source ดังนั้นมันสามารถถูกนำเสนอโดย Player หรือ Processor นอกจากสภาวะ 6 สภาวะของ Player Processor ยังรวมสภาวะอีกสองสภาวะ ซึ่งเกิดขึ้นก่อน Processor เข้าสู่สภาวะ Realizing แต่หลังจากสภาวะ Unrealized Configuring : Processor เข้าสู่สภาวะ Configuring

2.11.3.7 DataSink

DataSink เป็นอินเตอร์เฟสพื้นฐานสำหรับออบเจ็คที่ใช้ในการอ่านสื่อที่ส่งมาจาก DataSource และส่งไปยังปลายทาง

2.11.3.8 Format

ออบเจ็ค Format จะแสดง Format ที่ถูกต้องแก่สื่อ Formatโดยตัวของมันเองแล้วจะไม่มีพารามิเตอร์เกี่ยวกับการเข้ารหัสโดยเฉพาะ หรือ ข้อมูลเกี่ยวกับเวลา มันจะบ่งบอกถึงชื่อของการเข้ารหัสและประเภทของของข้อมูลที่ Format ต้องการ สับคลาสของFormatจะรวม Audio Format และ Video Format โดยที่ Video Format ประกอบด้วยสับคลาสได้แก่ H261Format, H263Format, Indexed Color Format, JPEG Format, RGB Format และ YUV Format

2.11.3.9 Manager

ออบเจ็ค Manager เป็นออบเจ็คที่ใช้ในการเชื่อมต่อ อิมพิเมนต์เฟสหลายๆ อินเตอร์เฟสที่ใช้กับคลาสที่เกิดขึ้น ไม่มีอยู่ในโลกความเป็นจริงในระบบสเตอริโอ แต่คุณสามารถจินตนาการ Manager คล้ายกับเป็นสิ่งเชื่อมต่อวัตถุสองสิ่งที่แตกต่างกัน ยกตัวอย่างเช่น สร้าง Player จาก DataSource JMF นำเสนอ Manager 3 ประเภท

1. Manager: ใช้ Manager สร้าง Players, Processor, DataSources และ DataSinks ยกตัวอย่าง เช่น ถ้าต้องการที่จะส่ง DataSources สามารถใช้ Manager สำหรับ การสร้าง Player สำหรับ DataSources

2. Package Manager: Manager นี้เป็นตัวเก็บรักษาค่าที่ลงทะเบียนของแพ็คเกจที่บรรจุไว้ในคลาสของ JMF เช่น Players , Processor, DataSources และ DataSinks

3. Capture Device Manager: Manager นี้เก็บรักษาค่าลงทะเบียนไว้ของอุปกรณ์ที่ตรวจพบ

2.11.3.10 การสร้าง Player

การโปรแกรมมัลติมิเดียโดยใช้ JMF งานหนึ่งที่สำคัญที่สุดคือการสร้าง Player โดยการเรียกเมื่ทธอด createPlayer() Manager ใช้ URL ของสื่อหรือ MediaLocator ที่ระบุไว้ในการสร้าง Player ที่เหมาะสม เมื่อทำการสร้าง Player แล้วสามารถสร้างส่วนประกอบที่สามารถมองเห็นได้ของ Player โดยใช้หน้าต่างหรือใช้ Applet เพื่อที่จะแสดงวัตถุที่มองเห็นได้ของออปเจ็ค Player จะต้อง เรียกส่วนประกอบที่มองเห็นได้โดยเรียกใช้เม็ธธอด getVisualComponent() และ ใส่ส่วนประกอบที่มองเห็นได้โดยเรียกหน้าต่างของแอปพลิเคชั่นหรือ Applet

Player สามารถรวมพาเนลที่ใช้ควบคุมกับปุ่มที่มีปุ่มเริ่มต้น หยุดชั่วคราว และหยุดสายของสื่อเช่นเดียวกับการควบคุมเสียง เหมือนกับที่ใช้บนเครื่องเล่นซีดี

เม็ธธอดของ Player สามารถเรียกได้ตั้งแต่ Player อยู่ในสภาวะ Realized เพื่อจะให้แน่นอนว่ามันอยู่ในสภาวะนี้ โดยสามารถเรียกใช้เม็ธธอดของ Manager ที่เรียกว่า createRealizedPlayer() สร้าง Player เม็ธธอดนี้เป็นวิธีที่สะดวกที่จะสร้าง Player ภายในขั้นตอนเดียว เมื่อถูกเรียก มันจะปิดกั้นจน Player อยู่ในสภาวะ Realized ต่อไปจากนี้ เม็ธธอด start() จะพยายามเปลี่ยนแปลง Player ไปยังสภาวะ Started จากสภาวะที่เป็นอยู่ ยกตัวอย่างเช่น สามารถเรียก start() ทันทีหลังจากที่ Player ถูกสร้างขึ้น เม็ธธอด start() จะเรียกเม็ธธอดที่จำเป็นทั้งหมดเพื่อที่จะนำ Player ไปสู่สภาวะ Started

2.11.3.11 ข้อมูลสื่อที่ถูกจับ

การจับสื่อเป็นสิ่งสำคัญอีก อย่างหนึ่งในการโปรแกรมโดย JMF สามารถทำการจับสื่อโดยใช้อุปกรณ์จับสื่อ เช่น Micropiione หรือกล้องวีดีโอ มันสามารถโพรเซสและส่งต่อ หรือเก็บไว้ในFormat สื่อ เพื่อจับข้อมูลสื่อ จำเป็นที่จะต้อง

1. ติดตั้งอุปกรณ์ตรวจจับที่ต้องการใช้โดยการค้นหาใช้ CaptureDeviceManager
2. รับออปเจ็ค CaptureDeviceInfo สำหรับอุปกรณ์
3. รับ Media Locator จากออปเจ็ค CaptureDeviceInfo และใช้เมื่อสร้าง DataSource
4. สร้าง Player หรือ Processor โดยใช้ DataSource
5. เริ่มต้น Player หรือ Processor เพื่อที่จะเริ่มต้นกระบวนการจับ

2.11.3.12 การโพรเซสมัลติมีเดียที่เป็นแบบเวลาจริง (Realtime Multimedia Processing)

สามารถใช้ JMF ในการส่งหรือรับการบรอดคาสท์ เช่น วิทยุที่กระจายเสียงหรือการออกอากาศของทีวี หรือประชุมผ่านเครือข่ายอินเทอร์เน็ตหรืออินทราเน็ต

คุณลักษณะของสื่อที่เป็นแบบเวลาจริงจะแตกต่างกับการเข้าถึงข้อมูลแบบคงที่ โปรโตคอลที่ใช้แบบเวลาจริงไม่ประกันว่า แพ็คเกจทุกแพ็คเกจจะมาถึงโดยปลอดภัย สิ่งที่สำคัญมากที่สุดคือ ทำอย่างไรจึงจะไม่เกิดการสูญเสียโดยรวมและประกันว่าไม่มีความล่าช้าในการรับ เมื่อเล่นข้อมูลโดยปราศจากการโหลดข้อมูลทั้งหมดมา การส่งผ่านอินเทอร์เน็ตในแบบเวลาจริงต้องการแบนด์วิดท์ที่มากฝ่ายรับจะสามารถเล่นข้อมูลสื่อได้อย่างต่อเนื่องโดยใช้โปรโตคอลของมันโดยเฉพาะเพื่อส่งแพ็คเกจของสายของสื่อแบบเวลาจริง องค์กร IETF(Internet Engineering Task Force) ได้กำหนดโปรโตคอลที่เรียกว่า RTP (Real Time Protocol) ซึ่งเป็นโปรโตคอลที่เหมาะสมสำหรับงานประยุกต์ที่ต้องการส่งข้อมูลแบบเวลาจริงเช่น ออดิโอ, วีดีโอ ผ่านการบริการแบบมัลติคลาส หรือแบบ Unicast RTP มักจะใช้บน UDP(User Datagram Protocol) ไม่มีการรับประกันว่าข้อมูลที่ส่งแบบ RTP จะมาถึงตามลำดับที่ มันถูกส่งมา ในความเป็นจริงแล้ว ไม่มีการรับประกันว่ามันจะมาทั้งหมดมันขึ้นอยู่กับฝั่งรับว่าจะรวมแพ็คเกจถูกส่งตามลำดับหรือไม่ และตรวจ

โดยใช้ข้อมูลที่ถูกส่งมาในส่วนหัวของแพ็คเกจ และมีการใช้ RTCP (Real Time Transport Control Protocol) ที่มีการใส่ของที่อยู่ต้นทาง และประกันคุณภาพของการบริการงานประยุกต์ที่ใช้ RTP จะสามารถแบ่งกลุ่มได้เป็น RTP เซอร์เวอร์ (งานประยุกต์ที่จำเป็นที่ใช้ในการส่งผ่านเน็ตเวิร์ค) อย่างไรก็ตาม บางแอปพลิเคชัน เช่น การประชุมทางไกล จะสร้างเซสชั่น RTP ขึ้นมาจับและส่งข้อมูล เช่นเดียวกับทางรับ

JMF API ที่ถูกนิยามในแพ็คเกจก็คือ javax.media.rtp,javax.media.rtp.event และ javax.media.rtp.rtcp สำหรับการเล่นสายของ RTP ในการเล่นสายและการส่งของ JMF RTP API สามารถทำงานได้กับอุปกรณ์จับของ JMF, Players, Processors และ ความสามารถในการ โพรเซส นอกจาก Managers สื่อย่างที่ได้อธิบายไปยังมี Manager อีกอย่างที่ใช้ในการจัดการกับ RTP เซสชั่น คือ SessionManager มันยังคงเก็บแทร็กของเซ็จชันที่เข้าร่วม และสายที่ถูกส่ง เช่นเดียวกับการจัดการควบคุมโดยใช้ RTP และสนับสนุน RTP สำหรับทั้ง ภาคส่ง และ ภาครับ

2.12 Java Network Programming

java.net package ประกอบด้วยคลาสที่สำคัญดังนี้

2.12.1 InetAddress

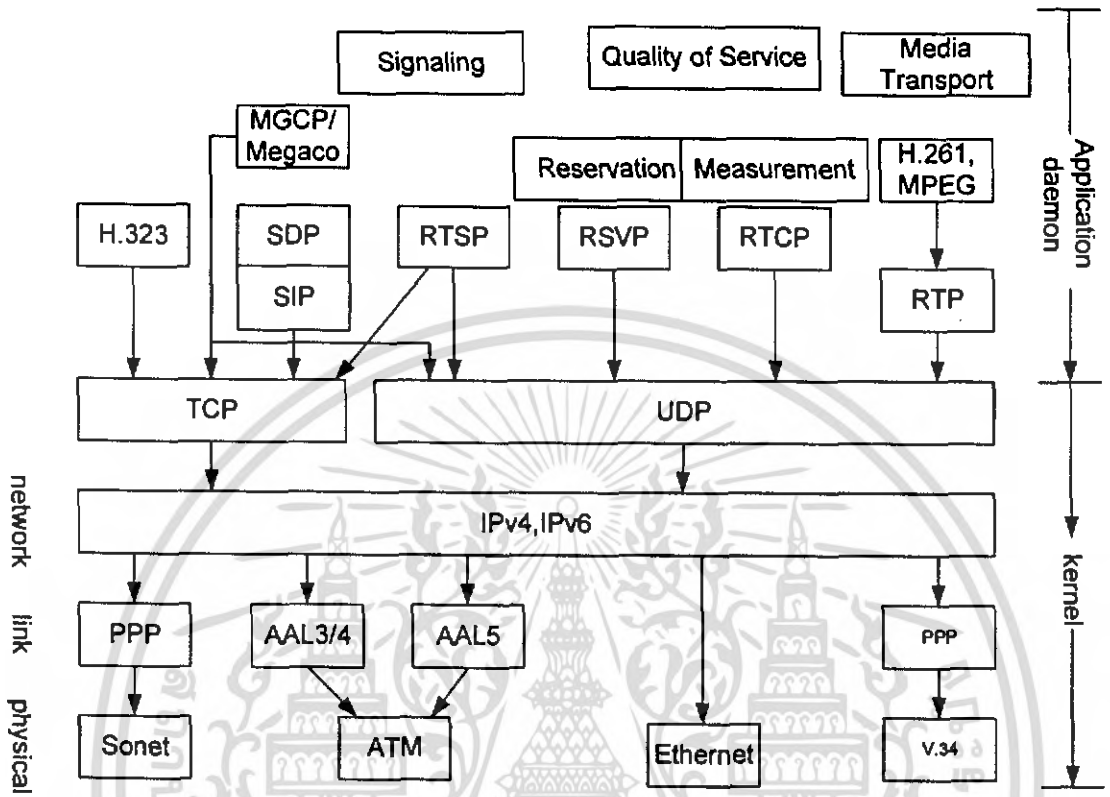
ใช้สำหรับเก็บแสดง IP Address ซึ่งใช้ได้กับที่ TCP และ UDP โดยปกติใน Instance ของคลาส InetAddress จะมีข้อมูลเกี่ยวกับ IP Address และอาจมี Domain name ของ IP Address นั้นด้วย

คลาส InetAddress ไม่มี Constructor แต่มี public static factory methods สำหรับสร้าง Instance ของคลาส เช่น public static InetAddress getLocalHost throws UnknownHostException; ซึ่งจะให้ InetAddress ของเครื่องที่ใช้งาน หลังจากนั้นเราอาจใช้ public String getHostName(); หรือ public String getHostAddress(); เพื่อเรียกขอ Domain name และ IP address ของ InetAddress นั้นออกมาในรูปของ String

2.12.2 Socket

- ใช้สำหรับสร้าง Socket ซึ่งมีกลไกสำหรับสร้างการติดต่อไปสู่เครื่องเป้าหมายที่ Port หนึ่ง โดยเครื่องเป้าหมายอาจเป็นเครื่องผู้รับหรือผู้ส่งก็ได้
- สร้าง Stream สำหรับอ่านหรือเขียนข้อมูลไปที่เครื่องเป้าหมาย
- ทำการปิดการติดต่อไปสู่เครื่องเป้าหมาย
- คลาส Socket มี Constructors หลายตัวเช่น public Socket (String host,int port) throws UnknownHost Exception IOException;, public Socket (String host, int port) throws IOException; จะสร้าง Socket ไปยังเครื่อง Host ที่ระบุโดยพารามิเตอร์ตัวแรกที่เป็น String หรือ InetAddress โดยติดต่อเข้าไปที่ Port เลขที่ระบุด้วยพารามิเตอร์ตัวที่สอง ซึ่งเป็น int

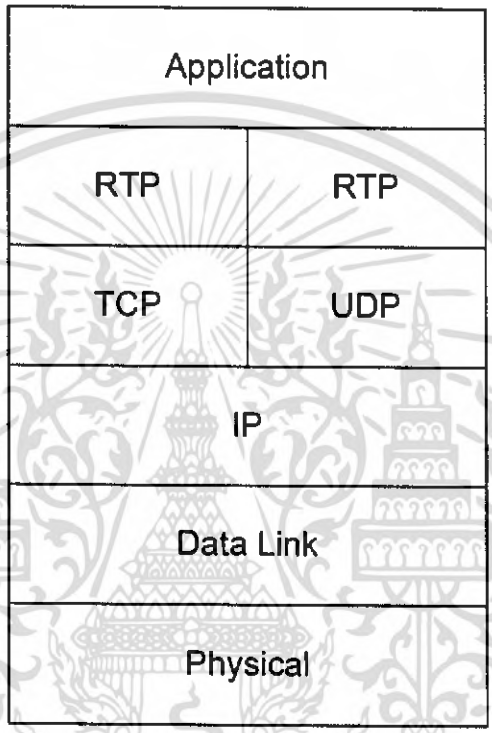
2.13 RTP (Real Time Protocol)



รูปที่ 2.20 แสดงแสตคของโปรโตคอล

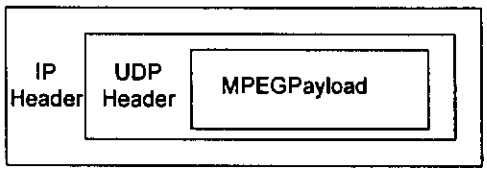
2.13.1 สถาปัตยกรรม RTP

1. ใช้ในการส่งข้อมูลผ่านเครือข่ายสำหรับการส่งข้อมูลพวกใช้เวลาจริง (Real Time) เช่น ทางวีดีทัศน์
2. สามารถใช้งานได้ร่วมกับโปรแกรมประยุกต์โครงข่ายมัลติมีเดียอื่นๆ ได้
3. RTP ไม่เป็นแบบ Connection Oriented
4. ไม่มีความผิดพลาดในการเรียงลำดับข้อมูล ซึ่งแตกต่างจาก UDP เมื่อทำการส่งแล้วมีปัญหาในการลำดับก่อนหลังของเฟรม
5. ข้อมูลที่ส่งจะถูกควบคุมด้วย RTCP
6. RTP ไม่มีการรับประกันคุณภาพของข้อมูลที่ส่ง หมายความว่าไม่ได้มีกลไกใดๆ ในการยืนยันข้อมูลว่าส่งได้สำเร็จหรือไม่



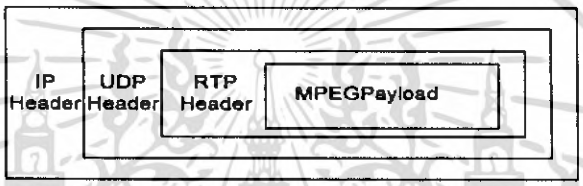
รูปที่ 2.21 แสดงสถาปัตยกรรม RTP

2.13.2 ลักษณะการส่งข้อมูล



รูปที่ 2.22 แสดงลักษณะการส่งข้อมูล

2.13.3 ลักษณะการส่งข้อมูลของโครงงาน



รูปที่ 2.23 แสดงลักษณะการส่งข้อมูลของโครงงาน

2.13.4 ลักษณะของ RTP Package header

รูปที่ 2.24 แสดง RTP Package header

RTP Package header ประกอบด้วย

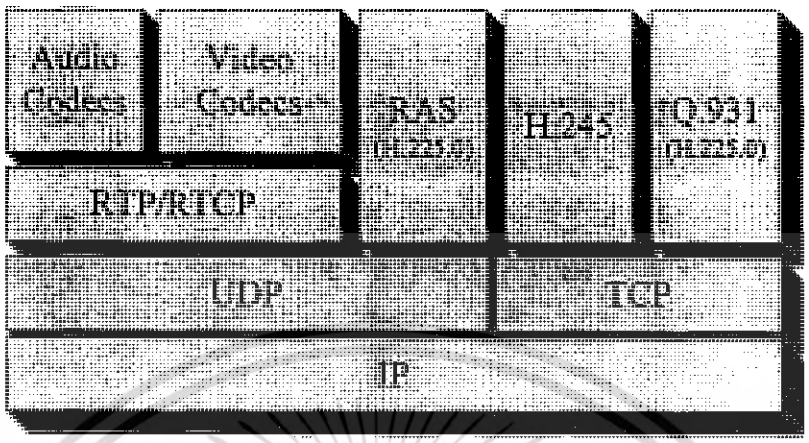
1. Payload Type Field Payload Type จะใช้เป็นตัวบ่งบอกถึงรูปแบบการเข้ารหัสทางวีดิทัศน์ ยกตัวอย่างเช่น มาตรฐาน JPEG, MPEG1, MPEG2, H.261

2. Sequence Number Field

3. Timestamp Field สามารถนำมาจัดเรียงข้อมูลเป็นลำดับได้อย่างถูกต้อง

4. Synchronization Source Identifier (SSRC)

2.13.5 RTP โปรโตคอลแสตค



รูปที่ 2.25 แสดงแสตคของโปรโตคอล

RTP เป็นโปรโตคอลที่สร้างขึ้นสำหรับรองรับบริการส่งข้อมูลจากต้นทางไปยังปลายทางด้วยคุณลักษณะของข้อมูลแบบเวลาจริง เช่น การส่งสัญญาณโต้ตอบของเสียงและวีดีโอการบริการนี้จะประกอบไปด้วยการบ่งบอกชนิดของข้อมูล การลำดับหมายเลข กำกับช่วงเวลา ติดตามขบวนการส่งข้อมูลการประยุกต์ใช้งานโดยทั่วไป RTP จะทำงานบน UDP จะทำให้มันมีการรวบรวมและตรวจสอบความผิดพลาดของข้อมูลโดยตัวโปรโตคอลจะสนับสนุนการทำงานต่างๆ ของชั้นทรานสปอร์ตและเน็ตเวิร์ค เลเยอร์ ซึ่ง RTP ถูกออกแบบมาเพื่อจุดประสงค์ดังกล่าว เพราะเมื่อใช้ TCP นั้นมีปัญหาเรื่องเสียโอเวอร์เฮดในการแจ้งข้อผิดพลาดกลับทุกครั้งเมื่อข้อมูลส่งเกิดการผิดพลาด RTP อาจจะถูกนำไปใช้กับโครงข่ายหรือโปรโตคอลในระดับล่างที่มีความเหมาะสมได้เช่นกันและยังจะสนับสนุนการส่งข้อมูลไปยังหลายๆปลายทาง ด้วยการใช้รูปแบบการกระจายตัว

RTP เองจะไม่มีกลไกการรับรองในเรื่องเวลาการส่งข้อมูลหรือรับรองในคุณภาพของบริการ (Quality of Service หรือ QOS) แต่จะพึ่งพาให้เลเยอร์ชั้นต่ำกว่าทำหน้าที่แทน ไม่รองรับการส่งข้อมูลหรือป้องกันข้อมูลสูญหาย นั่นหมายความว่าเครือข่ายระดับล่างนั้นต้องมีความน่าเชื่อถือและชุดข้อมูลที่ส่งไปจะต้องมีค่าลำดับ

หมายเลข และการลำดับหมายเลขที่อยู่ในโปรโตคอลนี้จะทำให้ด้านรับสามารถที่จะสร้างชุดข้อมูลของผู้ส่งขึ้นมาใหม่ให้เป็นไปตามความถูกต้องและต่อเนื่อง หมายเลขของลำดับอาจถูกนำมาพิจารณาตำแหน่งลำดับความถูกต้องในชุดข้อมูล

สำหรับ RTP จุดมุ่งหมายหลักเองจะถูกออกแบบมาให้เหมาะสมกับการใช้ทำงานโต้ตอบทางสื่อมัลติมีเดียแต่ไม่มีการจำกัดหากจะนำไปใช้งานเฉพาะอื่นๆ เช่น ใช้จัดเก็บข้อมูลแบบต่อเนื่องแปลงชุดข้อมูลโต้ตอบ ประยุกต์ใช้งานในการควบคุมและการวัด

RTP ประกอบด้วย 2 ส่วนหลักๆคือ

- RTP สำหรับพาข้อมูลที่มีคุณลักษณะของเวลาจริง

- RTP จัดให้มีหน้าที่ตรวจตราคุณภาพในการบริการและนำพาข้อมูลที่เกี่ยวข้องในรูปแบบของการควบคุมชุดข้อมูลที่รวมกันอยู่และลักษณะส่วนท้ายของ RTP เป็นลักษณะการควบคุมที่มีความยืดหยุ่นสูง เช่น หากไม่มีการแจ้งการควบคุมและใช้งานจากสมาชิก มันก็ไม่มีความจำเป็นที่จะต้องใช้การควบคุมชุดข้อมูลแบบเต็มรูปแบบโดยจะเปลี่ยนไปใช้การควบคุมพิเศษเฉพาะบางส่วน และ RTP แสดงให้เห็นว่าเป็นโปรโตคอลรูปแบบใหม่ตามหลักการจัดเฟรมและมีการร่วมกันประมวลผลในชั้นเลเยอร์ที่เสนอขึ้นโดยนายคาร์กและเทนเนนฮอร์สที่ตั้งใจให้มันเป็นรูปแบบที่หลากหลายในการประยุกต์ใช้งานกับข้อมูลรูปแบบต่างๆ ดังต่อไปนี้

1.13.5.1 รูปแบบพื้นฐานในการประชุมทางเสียง

โดยใช้บริการบริการไอพีมัลติแคสกระจายข้อมูลไปยังอินเทอร์เน็ตในรูปแบบการติดต่อสื่อสารทางเสียงโดยใช้หมายเลขแอดเดรสแบบกระจายและพอร์ตเป็นคู่โดยใช้พอร์ตหนึ่งเป็นข้อมูลเสียงและอีกพอร์ตใช้เป็นพอร์ตข้อมูลควบคุมตำแหน่งและพอร์ตจะมีการสร้างข้อมูลให้กระจายไปยังผู้ที่มีส่วนร่วม

ข้อมูลเสียงที่ประยุกต์ใช้ในงานการประชุมทางเสียงจะเป็นการส่งข้อมูลเสียงโดยการสไลด์เวลาของผู้ที่มีส่วนร่วมในการประชุมเป็นช่วงสั้นประมาณ 20 มิลลิวินาที แต่ละช่วงของข้อมูลเสียงจะดำเนินกระบวนการโดย RTP เฮดเดอร์ แล้วข้อมูลจะถูกบรรจุเข้าไปในยูดีพีแพ็คเกจ RTP เฮดเดอร์ จะมีการบ่งบอกว่าเป็นข้อมูลชนิดใด ในการเข้ารหัสเสียง เช่น พีซีเอ็ม (Pulse Code Modulation) เอดีพีซีเอ็ม (Adaptive Differential Pulse Code Modulation) หรือ แอลพีซี (Linear Predictive Coding) ที่ถูกบรรจุในแต่ละแพ็คเกจทำให้ผู้ส่งสามารถเปลี่ยนแปลงช่วงเวลาวิธีการเข้ารหัสสัญญาณในการประชุมให้เหมาะสมเช่นผู้ร่วมประชุมบางท่านอาจจะเชื่อมต่อด้วยลิงค์ที่มีอัตราการส่งผ่านข้อมูลต่ำ

อินเทอร์เน็ตก็เหมือนโครงข่ายแพ็คเกจอื่นๆ ในกรณีที่มีการสูญหายและเปลี่ยนตำแหน่งของแพ็คเกจและเกิดความล่าช้าซึ่งเป็นตัวแปรที่ขึ้นอยู่กับเวลา ใน RTP เฮดเดอร์จะมีการบรรจุข้อมูลของเวลาและลำดับข้อมูลที่ให้สิทธิกับด้านรับให้สามารถสร้างโครงสร้างของเวลาขึ้นมาดังเดิม โดยเหมือนแหล่งข้อมูลตัวอย่างเช่นแหล่งข้อมูลของ RTP แพ็คเกจในการประชุม ลำดับของข้อมูลที่ผู้รับสามารถนำมาประเมินว่ามีจำนวนข้อมูลเท่าไรที่มีการสูญหายไป และเมื่อสมาชิกเข้าร่วมหรือออกจากกลุ่ม มันจะมีประโยชน์ที่จะรับรู้ว่ามีใครเข้าร่วมอยู่บ้าง ณ เวลานี้ และความสามารถรับส่งข้อมูลได้ดีแค่ไหน ในส่วนนี้จะมีการเพิ่มชื่อของผู้ใช้บน RTCP (พอร์ตควบคุม) การรับข้อมูลจะมีการรายงานให้ทราบสถานะภาพผู้รับฟังและการใช้งานการถอดรหัสการควบคุม ในการเพิ่มส่วนของชื่อผู้ใช้ และบ่งบอกข้อมูลอื่นๆ ที่อาจจะรวมไปกับแบนด์วิดท์ที่จำกัด และจะมีการส่ง RTP บายแพ็คเกจ เมื่อมีการยกเลิกการประชุม

2.13.5.2 การประชุมด้วยภาพและเสียง

เป็นการสื่อสารรวมกันทั้งข้อมูลภาพและเสียงที่ใช้ในการประชุม โดยจะถูกจัดส่งโดยแยกส่วน RTP และ RTP แพ็คเกจในแต่ละตัวกลางโดยการใช้คู่ของพอร์ตยูดีพีที่แตกต่างกัน หรืออาจใช้ร่วมกับการระบุตำแหน่งแบบจุดต่อจุดโดยที่จะไม่มีการเชื่อมต่อร่วมกันโดยตรงในระดับของ RTP ระหว่างส่วนของเสียงและวีดีโอ ยกเว้นการประชุมที่มีการกำหนดชื่อของทั้งสองส่วนที่เหมือนกันใน RTCP แพ็คเกจเพื่อให้มีการรวมข้อมูลกัน

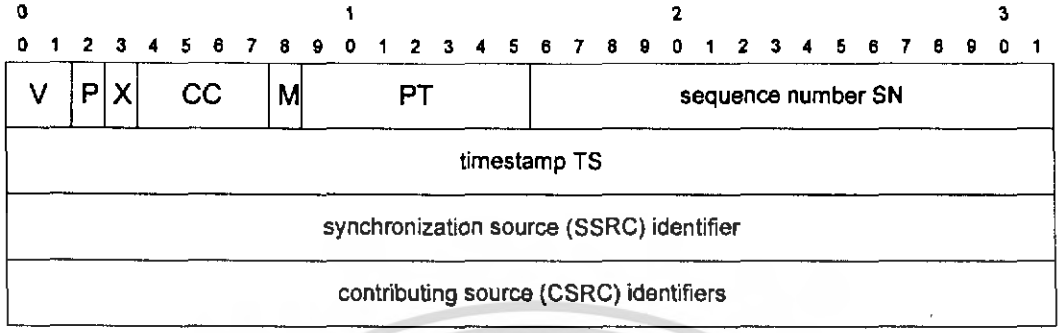
2.13.5.3 มิกเซอร์และการทรานสเลเตอร์

ในกรณีนี้เราสมมติว่าทุกส่วนต้องการรับข้อมูลมีเดียในรูปแบบเดียวกัน พิจารณาในกรณีผู้เข้าร่วมประชุมหนึ่งอยู่ในพื้นที่ซึ่งเชื่อมต่อข้อมูลได้ที่ความเร็วต่ำไปยังส่วนหลักที่มีผู้ประชุมอื่นรวมกันอยู่ด้วยโครงข่ายความเร็วสูง มันจะเป็นตัวนำในการบังคับให้ทุกคนใช้แบนด์วิดท์ที่ต่ำลงมา และสร้างการเข้ารหัสเสียงเพื่อรองรับคุณภาพใหม่โดย RTP ในส่วนที่เรียกว่ามิกเซอร์อาจจะนำมาวางตรงตำแหน่งที่ใกล้กับพื้นที่แบนด์วิดท์ต่ำ โดยมิกเซอร์นี้จะทำการรีซิงโครไนซ์กับการเข้ามาของแพ็คเกจเสียงไปยังการสร้างขึ้นใหม่ของผู้ส่งใน 20 มิลลิวินาที จะทำการรวมสายข้อมูลเสียงไปเป็นสายข้อมูลเดี่ยว แปลงการเข้ารหัสเสียงไปยังแบนด์วิดท์ที่ต่ำกว่า และทำการส่งกลับผ่านการเชื่อมต่อแบบความเร็วต่ำ โดยแพ็คเกจอาจส่งไปยังผู้รับเดี่ยวหรือผู้รับหลายท่านด้วยแอดเดรสที่ต่างกัน

RTP จะมีความสามารถในการบ่งชี้ชนิดข้อมูล ลำดับเลขข้อมูล ระบุเวลาและตรวจตราการส่งข้อมูล สามารถลำดับเลขข้อมูลใหม่หากข้อมูลนั้นมาถึงด้านรับโดยไม่เป็นลำดับ การมีเลขลำดับทำให้สามารถค้นหาแพคเกจข้อมูลที่สูญหายได้ การระบุเวลามีเพื่อใช้ในคุณสมบัติการเล่นมีเดีย ข้อมูลที่มาถึงทางด้านรับจะถูกตรวจสอบอย่างต่อเนื่องโดย RTCP ซึ่งอยู่ในเลเยอร์เดียวกับ RTP เพื่อให้มีการปรับการเข้ารหัสและแสดงพารามิเตอร์ในการส่งข้อมูล ตัวอย่างเช่นถ้าพบว่ามีแพ็คเกจสูญหายจำนวนมากมันจะแจ้งให้มีการส่งข้อมูลด้วยอัตราเร็วที่ต่ำลง

จำนวนเลขที่ระบุอยู่ในโปรโตคอลจะเป็นเลขที่อ้างอิงบิดที่อยู่ในรูปแบบออกเตจ โดยเลขลำดับไบต์นี้จะรู้จักกันในชื่อบิ๊กเอนเดียน ทุกๆ ส่วนของข้อมูลเฮดเดอร์จะถูกจัดเรียงเป็นช่วงความยาวแบบทั่วไป เช่น 16, 32, 64 บิต โดยจัดให้เป็นไบต์จำนวนคู่ ระบบเวลาจะถูกแสดงโดยรูปแบบการประทับเวลาของเอ็นทีพี (NTP: Network Time Protocol) ซึ่งเกี่ยวเนื่องกับเวลายูทีซีอัตราการกำหนดเวลาของเอ็นทีพีเต็มที่คือ 60 บิต โดยมีส่วนหลักใน 32 บิตแรก และแยกย่อยใน 32 บิตหลัง

2.13.6 RTP เฮดเดอร์



รูปที่ 2.26 แสดงชุดข้อมูลส่วนหัวของ RTP

รูปแบบส่วนหัวของ RTP จะกำหนดให้ประกอบไปด้วยชุดข้อมูล โดยที่ 12 ออคเตจ แรกจะเกิดขึ้นในทุกๆRTPแพ็คเกจ ขณะที่ส่วนของ CSRC จะแสดงขึ้นเฉพาะในบางแพ็คเกจ เมื่อมันทำหน้าที่เป็นมิกเซอร์แต่ละส่วนนั้นมีความหมายดังนี้

version (V) : จำนวน 2 บิต ส่วนนี้จะแสดงเวอร์ชั่นของโปรโตคอลRTP ซึ่งค่าปัจจุบันเป็นค่า เท่ากับ 2 โดยค่า 1 ถูกใช้ให้เป็นเวอร์ชั่นทดลอง

padding (P) : จำนวน 1 บิต ส่วนนี้ถ้าถูกตั้งเป็น 1 บิต ส่วนนี้ถ้าตั้งเป็น 1 ชุดข้อมูลจะถูกบรรจุด้วยส่วนเพิ่มเติมที่เป็นออคเตจในตอนท้าย แสดงให้เห็นว่าเมื่อจบเฮดเดอร์แล้วยังไม่ใช่ส่วนของชุดข้อมูลที่จะตามมา ส่วนนี้มีความสำคัญในการทำอัลกอริทึมซึ่งต้องการกำหนดขนาดของบล็อกข้อมูล เพื่อนำพาRTP หลายๆแพ็คเกจในเลเยอร์ที่อยู่ต่ำลงมา

extension (X) : จำนวนบิต 1 บิต ส่วนนี้ถ้าตั้งเป็น 1 แสดงว่าส่วนหัวของข้อมูลจะยังมีส่วนขยายตามมาอีก

CSRC count (CC) : จำนวน 4 บิต ส่วนนี้จะเป็นส่วนที่บ่งบอกจำนวน CSRC ที่จะเพิ่มเติมมาในส่วนของเฮดเดอร์ที่ได้กล่าวไว้ก่อนหน้านี้

marker (M) : จำนวน 1 บิต เป็นบิตที่ถูกกำหนดขึ้นเพื่อแสดงให้ทราบช่วงระยะขอบเขตของแพ็คเกจสตรีมแต่ละเฟรม เช่น RTP มาร์คบิตจะถูกตั้งเป็น 1 ถ้าแพ็คเกจบรรจุจำนวนบิตที่เหลือต่อจากเฟรมที่อยู่ก่อนหน้า

payload type (PT) : จำนวน 7 บิต ส่วนนี้จะแสดงชนิดของข้อมูลที่ถูกนำพาโดย RTP แพ็คเกจว่าเป็นรูปแบบใด เช่น MPEG , M-JPEG , PCM

Sequence Number (SN) : จำนวน 16 บิต เป็นส่วนแสดงเลขลำดับของชุดข้อมูล RTP ที่ได้ทำการส่ง เพื่อให้ฝั่งรับสามารถตรวจเช็คข้อมูลที่สูญหายและเรียงลำดับแพ็คเกจใหม่ ค่าเริ่มต้นของเลขลำดับจะถูกสุ่มขึ้นเพื่อเริ่มต้น

timestamp (TS) : จำนวน 32 บิต ส่วนนี้จะบรรจุข้อมูลเวลาที่แสดงการสุ่มค่าคงที่ของออคเตจแรกในข้อมูลแพ็คเกจ RTP และการสุ่มค่าคงที่นี้จะต้องกระทำโดยนาฬิกาที่เป็นลิเนียร์ที่สามารถซิงโครไนซ์และคำนวณการคลาดเคลื่อนของเวลาทางด้านรับได้ โดยค่าเริ่มต้นจะเกิดขึ้นโดยการสุ่ม

synchronization Source(SSRC) : จำนวน 32 บิต ส่วนนี้จะเป็นส่วนซิงโครไนซ์แหล่งข้อมูลที่ทำการส่งและรับโดยRTPเซสชั่นแบบการสุ่ม และเป็นส่วนช่วยในการแก้ปัญหาการชนกันของข้อมูล

Contributing Source (CSRC) : 0-15 ชุด แต่ละชุดมี 32 บิต ส่วนนี้เป็นข้อมูลเพิ่มเติมที่ใช้ในการแบ่งบอก ตำแหน่งภายในชุดข้อมูลที่มีรูปแบบต่างกัน จำนวนของส่วนนี้จะแสดงไว้ที่ CC หน้าที่ของส่วนนี้เช่นบอกตำแหน่งที่แตกต่างกันของสัญญาณเสียงที่เชื่อมต่อกันอยู่ในชุดของข้อมูล

2.13.7 โปรโตคอลอาร์ทีซีพี (RTP Control Protocol)

อาร์ทีซีพีเป็นโปรโตคอลที่ช่วยในการความคุมการส่งข้อมูลตามช่วงเวลาให้กับทุกการเชื่อมต่อโดยใช้วิธีการในลักษณะ เดียวกับกับการส่งแพ็คเกจของข้อมูล โปรโตคอลที่อยู่ชั้นต่ำกว่าจะต้องทำการจัดเตรียมกระบวนการเพื่อรองรับการมัลติเพลกซ์ และ ควบคุม แพ็คเกจ โดยอาร์ทีซีพีมีหน้าที่การทำงาน 4 ข้อดังนี้

1. หน้าที่หลักคือทำการรายงานผลคุณภาพของการส่งผ่านข้อมูลและเกี่ยวข้องกับฟังก์ชันการทำงานในส่วนของทรานสปอร์ตโปรโตคอล โดยทำการควบคุมการไหลและบีบอัดข้อมูลของโปรโตคอลในชั้นทรานสปอร์ต อื่นๆทำหน้าที่ส่งรายงานผลการรับข้อมูลไปยังผู้ร่วมใช้งานทุกคน ให้สิทธิ์แก่ผู้เฝ้าสังเกตการณ์ประเมินค่าปัญหาที่เกิดขึ้น ด้วยการกระจายตัวที่เหมือนกลไกการมัลติแคส เป็นผู้ให้บริการที่ 3 ที่ทำหน้าที่ตรวจตราปัญหาที่เกิดกับเครือข่าย

2. RTP จะทำหน้าที่นำพาข้อมูลที่มีการระบุแหล่งของ RTP ที่เรียกว่า Canonical Name หรือ CNAME เมื่อการระบุ SSRC อาจเกิดเปลี่ยนแปลงถ้าหากค้นพบข้อขัดข้องหรือโปรแกรมต้องเริ่มต้นทำงานใหม่ ทางด้านรับต้องการซีเนมที่จะทำการเก็บรักษาข้อมูลในแทร็กของผู้ใช้งานแต่ละคนและเพื่อเข้าไปมีส่วนร่วมในชุดข้อมูลรวมที่สัมพันธ์กันกับส่วนของ RTP เช่นตัวอย่าง การซิงโครไนซ์สัญญาณเสียงและวีดีโอ

3. ประมาณถึงจำนวนผู้เข้าร่วมใช้งานมากที่สุด โดยผู้เข้าร่วมแต่ละคนจะส่งแพ็คเกจควบคุมไปยังคนอื่นๆโดยแต่ละคน สามารถทราบถึงจำนวนผู้เข้าร่วมใช้งานรวม ซึ่งจำนวนเลขนี้สามารถนำไปคำนวณอัตราแพ็คเกจที่ทำการส่งไปแล้วได้

4. นำพาข้อมูลการควบคุมส่วนย่อย เช่น ตัวอย่างความสามารถในการระบุชื่อผู้เข้าร่วมใช้งานที่สามารถแสดงผลบนอินเตอร์เฟสของผู้ใช้ RTCP เป็นช่องทางที่สะดวกในการส่งข้อมูลไปยังผู้ร่วมใช้งานทุกคน

2.13.8 รูปแบบของแพ็คเกจ RTCP

แพ็จเกจอาร์ทีซีพีสามารถเป็นไปได้ทั้งชนิดผู้ส่งและผู้รับ ด้านผู้รับจะส่ง SR (Sender Report) ถ้าเป็นส่วนร่วมในการเข้าประชุมของเซสชั่น มิฉะนั้นแล้วก็จะส่งเป็นรายงานการรับในส่วนเพิ่มเติมไปยัง SR และ RR แพ็คเกจ RTPยังมีชนิดแพ็คเกจอื่นอีกเช่น SDES (Source Description), BYE และ APP (Application Defined) โดยค่าที่แสดงชนิดแพ็คเกจจะแสดงไว้ถัดไป

แสดงรายละเอียดของอาทีซีพีแพ็คเกจที่แสดงข้อมูลการควบคุมต่างๆมีดังนี้

SR : (Sender Report) รายงานผู้ส่ง ในเรื่องสถานะภาพการส่งและรับจากผู้เข้าร่วมใช้งานที่ทำหน้าที่ส่งข้อมูล

RR : (Receiver Report) รายงานผู้รับ ในเรื่องสถานะภาพการรับจากผู้เข้าร่วมใช้งานที่ไม่ได้ทำหน้าที่ส่งข้อมูล

SDES : (Source Description Items) ส่วนที่บรรยายลักษณะแหล่งจ่าย

BYE : แสดงสถานการณ์ยกเลิกใช้งานจากตัวอย่างจะมีส่วนที่เพิ่มเติมมาจากอาร์ทีซีพีเช่น

Reception Report Count (RC) : ส่วนนี้ 5 บิต จะแสดงจำนวนของบล็อกรายงานด้านรับที่บรรจุมาในแพ็คเกจ RTP

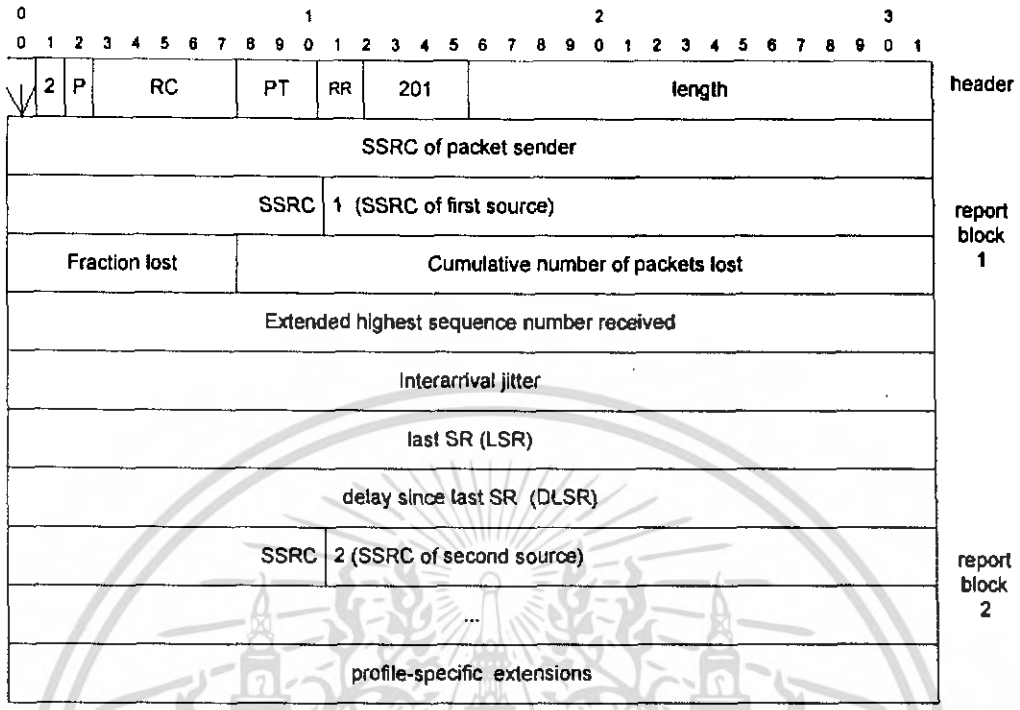
Fraction Lost : ส่วนนี้ประกอบด้วย 7 บิต แสดงแพ็คเกจ RTP ที่สูญหายไปก่อนที่แพ็คเกจ SR หรือ RR ถูกส่ง

Cumulative number of packets lost : ส่วนนี้ 24 บิต แสดงจำนวนเลขรวมของข้อมูลRTPแพ็คเกจทั้งหมดที่สูญหายจากการเริ่มส่งในเซสชั่น

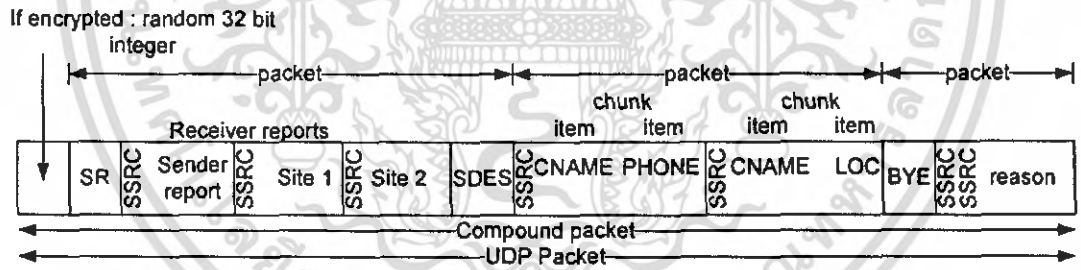
Extended highest sequence number received : ส่วนนี้ 32 บิต 16 บิต นัยสำคัญน้อยจะบรรจุหมายเลขลำดับที่ได้ทำการรับมาในข้อมูลRTPแพ็คเกจและ 16 บิต ที่มีนัยสำคัญมากกว่าเป็นส่วนขยายที่แสดงหมายเลขลำดับที่มีการโต้ตอบกลับไป

Interarrival jitter : ส่วนนี้ 32 บิตจะแสดงการประมาณค่าความแตกต่างของเวลาในแพ็คเกจ RTP ที่มาถึงซึ่งวัดโดยไทม์แสตมป์ยูนิต

Last SR timestamp : ส่วนนี้ 32 บิต จะบรรจุ 32 บิต กลางของ 64 บิต ที่เอ็นทีพีไทม์แสตมป์ของ SR แพ็คเกจที่ถูกรับมาล่าสุด



รูปที่2.27 แสดงรูปแบบ RTCP แพ็คเกจที่ใช้ในการรายงานโดยด้านรับ



รูปที่ 2.28 แสดงโครงสร้างของ RTCP แพ็คเกจ

2.14 ชนิดของการบีบอัดข้อมูล

ถ้าพิจารณาที่คุณภาพของไฟล์ก่อนและหลังการบีบอัดข้อมูลเป็นหลัก สามารถจำแนกการบีบอัดข้อมูลได้เป็น 2 ชนิดคือ

2.14.1 Lossless compression

คือ การบีบอัดที่ไม่มีการสูญเสียของข้อมูล หมายความว่าข้อมูลก่อนการบีบอัดและหลังจากคลายออกมาจากการบีบอัด เป็นข้อมูลที่เหมือนกันทุกประการ จุดเด่นของการบีบอัดข้อมูลชนิดนี้คือ ไม่ว่าจะทำการบีบอัดในลักษณะนี้กี่ครั้ง ข้อมูลที่คลายออกมาจะยังคงเหมือนกับต้นฉบับก่อนการบีบอัด จุดด้อยของการบีบอัดข้อมูลชนิดนี้คือ ขนาดของไฟล์ที่ถูกบีบอัด จะมีขนาดไม่เล็กลงมากเมื่อเทียบกับไฟล์ต้นฉบับ

2.14.2 Lossy compression

คือ การบีบอัดข้อมูลที่มีข้อมูลบางส่วนเสียหายไป ตรงกันข้ามกับการบีบอัดแบบ Lossless ไฟล์ที่คลายออกหลังจากการบีบอัดจะไม่เหมือนไฟล์ต้นฉบับ โดยมีบางส่วนถูกตัดทิ้งไปในขั้นตอนของการบีบอัดข้อมูล ดังนั้น ข้อดีของการบีบอัดชนิดนี้ คือ ทำให้ได้ไฟล์บีบอัดแล้วที่มีขนาดเล็กลงกว่าไฟล์ต้นฉบับมาก ส่วนข้อเสียคือ คุณภาพของไฟล์ที่ลดลงถ้าเป็นไฟล์ภาพหรือวีดีโอถ้าพิจารณาเทคนิค หรือขอบเขตการบีบอัดข้อมูลที่นำมาใช้กับไฟล์ภาพเคลื่อนไหวเป็นหลัก สามารถจำแนกการบีบอัดข้อมูลได้เป็น 2 ชนิดคือ

2.14.2.1 Intra-frame compression

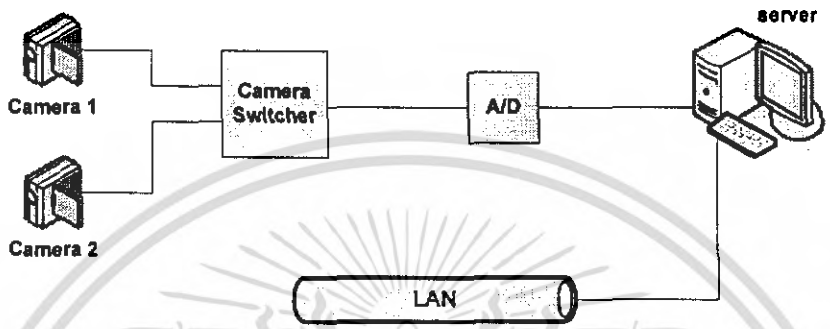
คือ เทคนิคในการบีบอัดข้อมูลที่พิจารณาเฉพาะข้อมูลที่อยู่ในเฟรมใดเฟรมหนึ่งเพียงเฟรมเดียวเท่านั้น หรืออาจกล่าวได้ว่าเทคนิคนี้เปรียบเสมือนกับการบีบอัดไฟล์ของภาพนิ่งเพียงภาพเดียวเท่านั้น ตัวอย่างเช่น เทคนิคการบีบอัดไฟล์ภาพเคลื่อนไหวหรือวีดีโอแบบ MJPEG คือ การใช้เทคนิคการบีบอัดแบบ Intra-frame compression เท่านั้น โดยทำการบีบอัดไฟล์ภาพนิ่งในแบบ JPEG กับแต่ละเฟรมภาพย่อย ทุกเฟรมในไฟล์ภาพเคลื่อนไหว และในการบีบอัดแต่ละเฟรมนั้นจะไม่มีการอ้างอิง หรือเกี่ยวข้องใดๆ กับเฟรมก่อนหน้าและเฟรมถัดจากเฟรมนั้นๆ ข้อดีของเทคนิคการบีบอัดในลักษณะนี้คือ แต่ละเฟรมจะเป็นอิสระต่อกัน สามารถที่จะเข้าถึง ตัดต่อ หรือแก้ไขเฟรมใดเฟรมหนึ่งได้โดยไม่กระทบกับเฟรมอื่นๆ ส่วนข้อเสียคือขนาดของไฟล์จะลดลงจากต้นฉบับในระดับหนึ่งซึ่งไม่มากนัก เนื่องจากยังมีส่วนของการเก็บข้อมูลที่ซ้ำๆ กันในระหว่างเฟรมอยู่นั่นเอง เทคนิคในการบีบอัดในลักษณะนี้เหมาะสำหรับการบีบอัดไฟล์วีดีโอที่จะนำไปแก้ไขต่อไป

2.14.2.2 Inter-frame compression

คือ เทคนิคในการบีบอัดข้อมูลที่ไม่เพียงพิจารณาข้อมูลที่อยู่ภายในเฟรมใดเฟรมหนึ่งเท่านั้น แต่เป็นการพิจารณาข้อมูลจากเฟรมอื่นๆ ที่ต่อเนื่องกัน คือเป็นการพิจารณาเฟรม ปัจจุบัน, เฟรมก่อนหน้า, และเฟรมถัดไป โดยจะทำการพิจารณาเก็บข้อมูลที่ซ้ำกัน และมีการเคลื่อนไหวน้อยที่สุดในแต่ละเฟรมเป็นเฟรมหลักไว้เพียงเฟรมเดียว ส่วนเฟรมอื่นๆ จะเก็บเฉพาะข้อมูลในส่วนที่มีการเคลื่อนไหว ตัวอย่างเช่น วีดีโอของการรายงานข่าวที่มีภาพของผู้อ่านข่าวนั่งอ่านข่าวอยู่ที่โต๊ะ เมื่อพิจารณาแล้วจะพบว่าส่วนที่ซ้ำกัน และเคลื่อนไหวน้อยที่สุดคือฉากหลัง หรือส่วนลำตัวของผู้อ่านข่าว ก็จะทำการเก็บข้อมูลนี้ไว้ในเฟรมแรกหรือเฟรมหลัก ส่วนเฟรมถัดๆ มาจะเก็บข้อมูลเฉพาะที่มีการเปลี่ยนแปลง หรือมีการเคลื่อนไหว เช่น ลำตัว, ส่วนศีรษะ, หรือส่วนใบหน้าเท่านั้น สำหรับข้อดีของเทคนิคการบีบอัดในลักษณะนี้คือ ขนาดของไฟล์ที่ลดขนาดลงมากเมื่อเทียบกับต้นฉบับ เนื่องจากมีการกำจัดข้อมูลที่ซ้ำซ้อนกันระหว่างเฟรม ส่วนข้อเสียคือไม่สามารถเข้าถึง, ตัดต่อ หรือแก้ไขเฟรมใดเฟรมหนึ่งได้ เนื่องจากข้อมูลที่เก็บอยู่ในเฟรมนั้นๆ เป็นข้อมูลบางส่วนที่มีการเปลี่ยนแปลงนอกเหนือจากข้อมูลที่เก็บอยู่ในเฟรมหลัก เทคนิคการบีบอัดข้อมูลวีดีโอในลักษณะนี้เหมาะสำหรับการบีบอัดเพื่อการเผยแพร่หรือการจัดเก็บเพื่อที่จะประหยัดเนื้อที่ในการจัดเก็บ หรือเพื่อให้สามารถเผยแพร่บนช่องทางสื่อสารที่จำกัดได้อย่างมีประสิทธิภาพ

บทที่3
การคำนวณและการสร้าง

กล้องรักษาความปลอดภัยผ่านเครือข่าย IP มีบล็อกไดอะแกรมทางภาคส่งดังนี้



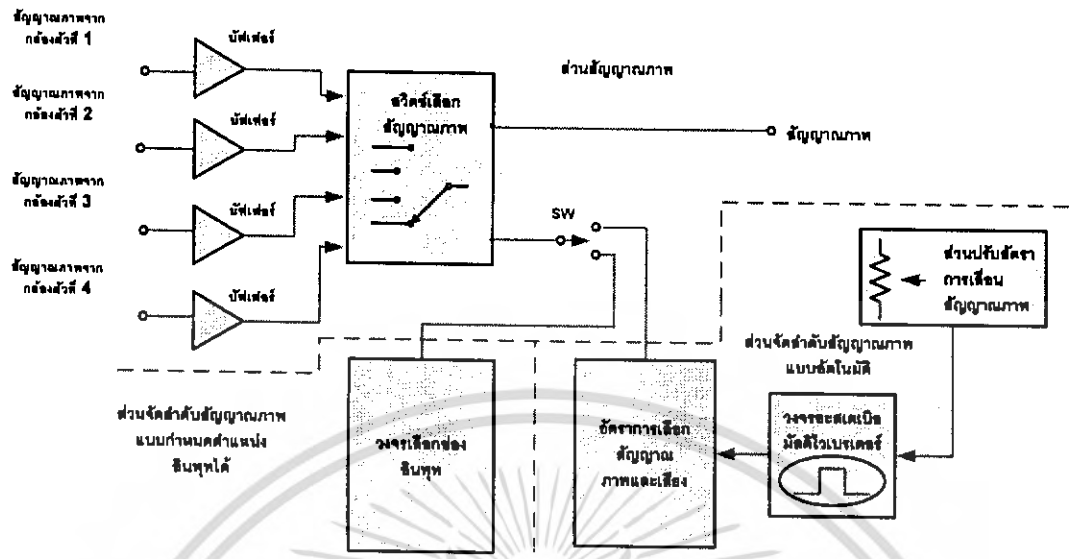
รูปที่ 3.1 บล็อกไดอะแกรมภาคส่ง

แบ่งการทำงานออกเป็นส่วนประกอบ 3 ส่วนคือ

1. วงจรสวิตช์ช่องสัญญาณ
2. วงจรแปลงสัญญาณแอนะล็อกเป็นสัญญาณดิจิทัล (Analog to Digital Converter)
3. โปรแกรมของด้านเซิร์ฟเวอร์

3.1 วงจรสวิตช์ช่องสัญญาณ

ในรูปที่ 3.2 เป็นไดอะแกรมแสดงโครงสร้างอย่างง่ายๆ ของวงจรสวิตช์ช่องสัญญาณภาพจากกล้อง สามารถแบ่งการทำงานออกเป็น 3 ส่วนหลักๆคือ ภาคสัญญาณภาพ,ส่วนจัดลำดับสัญญาณภาพแบบอัตโนมัติ และ แบบกำหนดตำแหน่งอินพุทได้ เริ่มจากภาคสัญญาณภาพ สัญญาณภาพจะถูกต่อเข้ามายังอินพุทโดย อินพุทแต่ละจุดถูกต่อไว้ด้วยตัวต้านทาน 75 โอห์มเพื่อป้องกันการสะท้อนของสัญญาณซึ่งเป็นสาเหตุหลักที่ทำ ให้ภาพขาดความชัดเจนจากนั้นสัญญาณภาพจะถูกส่งไปยังวงจรบัฟเฟอร์หรือวงจรขยายสัญญาณที่มีค่าเกน เท่ากับ 1 ผ่านไปยังส่วนเลือกสัญญาณภาพสัญญาณภาพที่มาจากกล้องวงจรปิด จะถูกเลือกโดยส่วนการทำงาน นี้ ก่อนจะออกไปยังเอาท์พุทเพื่อเข้าสู่คอมพิวเตอร์ต่อไป ส่วนของการจัดลำดับสัญญาณภาพ หน้าที่หลักของ ส่วนนี้ก็คือเลือกว่าสัญญาณภาพจากอินพุทช่องใดที่จะถูกส่งไปยังเอาท์พุท ในขณะเดียวกันก็ควบคุมการเลื่อน ตำแหน่งของอินพุท ให้เวียนไปตามลำดับ โดยอาศัยพัลซ์ที่มาจากวงจรอะสเตเบิลมัลติไวเบรเตอร์ เป็นตัว กำกับว่าจะให้มีการเลื่อนตำแหนงของอินพุทเร็วมากน้อยเพียงใด ในการทำงานโหมดอัตโนมัติ

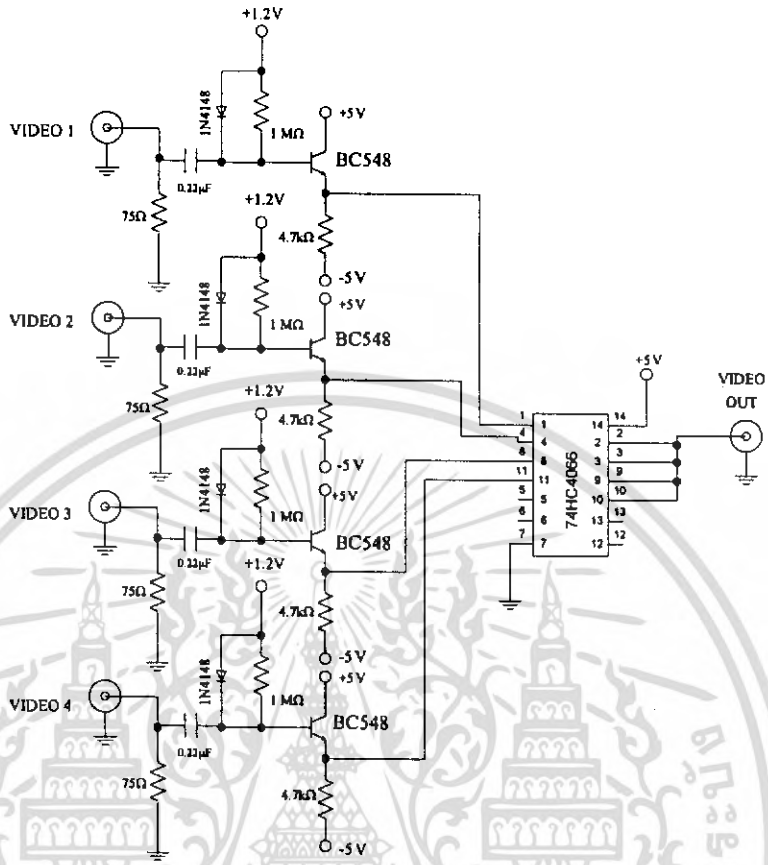


รูปที่ 3.2 ไดอะแกรมแสดงโครงสร้างของสวิตช์

สัญญาณจากกล้องวงจรปิด แต่ละตัวถูกสลับสับเปลี่ยนกันแสดงผลยังจอคอมพิวเตอร์ได้โดยมีลำดับก่อนหลัง และ ส่วนของวงจรเลือกช่องสัญญาณอินพุทในกรณีที่ผู้ใช้ต้องการเจาะจงดูอินพุทตำแหน่งใดๆ

3.1.1 ส่วนสัญญาณภาพ

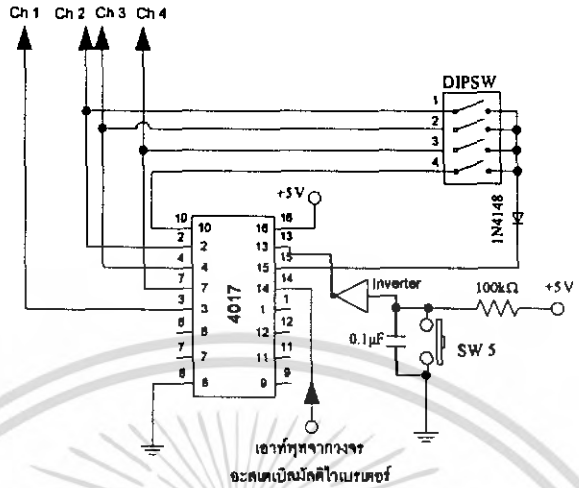
ส่วนของสัญญาณภาพ แม้ว่าไดอะแกรมในรูปที่ 3.2 จะใช้โรตารี่สวิตช์เป็นสัญลักษณ์แทนส่วนการทำงานดังกล่าวก็จริงแต่ในทางปฏิบัติการทำงานของส่วนดังกล่าวจะทำโดยอิเล็กทรอนิกส์สวิตช์ซึ่งแต่ละช่องสัญญาณจะต่ออยู่กับอิเล็กทรอนิกส์สวิตช์ที่อยู่ภายใน IC 74HC4066 โดยไม่ว่าเวลาใดก็ตามจะมีสวิตช์เพียงตัวเดียวเท่านั้นที่ทำงานอยู่และทำให้สัญญาณภาพจากอินพุทที่ต่ออยู่กับสวิตช์คู่นั้นสามารถส่งผ่านไปยังเอาท์พุทได้ จากรูปวงจรจะเห็นว่าอินพุทของสัญญาณทั้งหมดต่อวงจรบัฟเฟอร์เพื่อป้องกันความเพี้ยนของรูปสัญญาณ



รูปที่ 3.3 วงจรส่วนสัญญาณภาพ

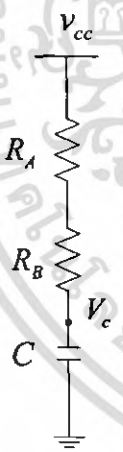
3.1.2 ส่วนจัดลำดับสัญญาณภาพแบบอัตโนมัติ

ส่วนของลำดับสัญญาณภาพแบบอัตโนมัติทำงานโดยใช้ไอซีนับสิบ (Decade counter) ซึ่งเป็นเคาน์เตอร์แบบ Johnson-Type ที่มีเอาท์พุทใช้งานได้ทั้งหมด 10 เอาท์พุทแต่ในที่นี้จะใช้งานเพียง 4 เอาท์พุทแรก เพื่อควบคุมอิเล็กทรอนิกส์สวิตช์ 4 คู่โดยจะใช้วิธีป้อนกลับสัญญาณที่ได้จากการนับด้วยไอซีนี้กลับมายังขามาสเตอร์รีเซต ขาที่ 15 เพื่อบังคับให้การนับทำค่าที่ต่ำกว่า 10 ได้ แต่ว่าจะนับไปเท่าใดนั้นก็ขึ้นอยู่กับการเซตที่ดิปสวิตช์ ซึ่งการเซตนี้ควรสอดคล้องกับจำนวนกล้องวงจรปิดที่มีการติดตั้งไว้เพื่อให้สมารถสลับภาพจากกล้องได้ครบทุกตัว สัญญาณความถี่ที่ใช้อ้างอิงในการสลับสัญญาณภาพจากอินพุทแต่ละช่องได้มาจากวงจรอะสเตเบิลมัลติไวเบรเตอร์สามารถปรับความถี่ได้โดยปรับที่ความต้านทานปรับค่าโดยค่าจะอยู่ระหว่าง 0.144 เฮิร์ต ถึง 30.30 เฮิร์ตโดยเอาท์พุทจากวงจรอะสเตเบิลมัลติไวเบรเตอร์จะต่อเข้าที่ขา 14 ของไอซี 4017 เพื่อควบคุมการนับ ในสภาวะปกตินั้นอินพุทขา 13 จะมีสถานะเป็นลอจิก "0" การนับก็จะดำเนินไปตามสัญญาณอินพุทจากขา 14 แต่เมื่อ ใดที่ อินพุทที่ขา 13 เป็น "1" จะทำให้การนับหยุดชั่วคราวทำให้เอาท์พุทค้างอยู่ที่ตำแหน่งเดิม และการนับจะดำเนินต่อไปอีกครั้งเมื่ออินพุทที่ขา 13 มีค่าเป็น "0"



รูปที่ 3.4 วงจรส่วนจัดลำดับสัญญาณภาพแบบอัตโนมัติ

ส่วนวงจรอะสเตเบิลมัลติไวเบรเตอร์นั้น สามารถทำการควบคุมการนับได้ด้วยการคำนวณดังนี้ จากรูปที่ 2.16 การหาค่าช่วงเวลา t_2 สามารถหาได้จากวงจรสมมูลดังรูปที่ 3.5



รูปที่ 3.5 วงจรสมมูลเมื่อตัวเก็บประจุทำการชาร์จประจุ

$$\frac{V_C - V_{CC}}{R_A + R_B} + \frac{C dV_C}{dt} = 0$$

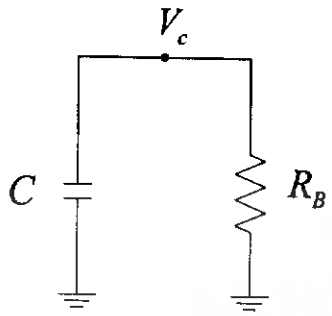
$$\frac{-dt}{(R_A + R_B)C} = \frac{dV_C}{(V_C - V_{CC})}$$

$$\frac{-1}{(R_A + R_B)C}\int_0^{t_2} dt = \int_{1/3V_{CC}}^{2/3V_{CC}} \frac{d(V_C - V_{CC})}{(V_C - V_{CC})}$$

$$\frac{-t_2}{(R_A + R_B)C} = \ln\left[V_C - V_{CC}\right]_{1/3V_{CC}}^{2/3V_{CC}}$$

$$\therefore t_2 = (R_A + R_B)C\ln 2 \quad (3.1)$$

สำหรับการหาค่าช่วงเวลา t_1 สามารถหาได้จากวงจรสมมูลดังรูปที่ 3.6



รูปที่ 3.6 วงจรสมมูลเมื่อตัวเก็บประจุทำการคายประจุ

$$\frac{V_C - 0}{R_B} + \frac{C dV_C}{dt} = 0$$

$$\frac{-1}{R_B C} dt = \frac{dV_C}{V_C}$$

$$\frac{-1}{R_B C} \int_0^{t_1} dt = \int_{2/3V_{cc}}^{1/3V_{cc}} \frac{dV_c}{V_c}$$

$$\frac{-1}{R_B C}(t_1) = \ln\left[V_C\right]_{2/3V_{cc}}^{1/3V_{cc}} = \ln\left[\frac{1/3V_{CC}}{2/3V_{CC}}\right] = \ln\frac{1}{2}$$

$$\therefore t_1 = R_B C \ln 2 \qquad (3.2)$$

จาก (3.1) $t_2 = (R_A + R_B) C \ln 2$

และ (3.2) $t_1 = R_B C \ln 2$

เนื่องจาก $T = t_1 + t_2 = R_B C \ln 2 + R_{AB} C \ln 2$

$$\therefore T = C(R_B + R_{AB}) \ln 2 \qquad (3.3)$$

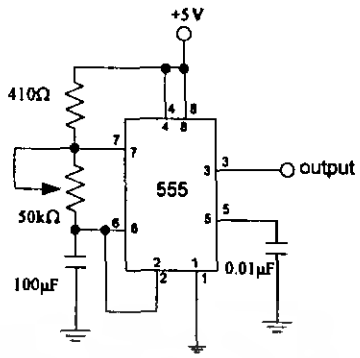
และ $$f = \frac{1}{C(R_B + R_{AB}) \ln 2}$$

$$f = \frac{1}{T} \approx \frac{1.44}{(R_A + 2R_B) C_1} \qquad (3.4)$$

และค่า Duty cycle หาได้จาก

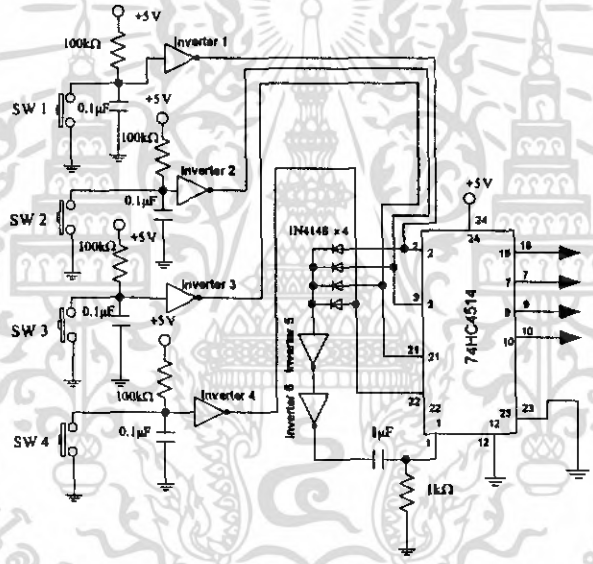
$$\text{Duty cycle} = \frac{t_2}{T} \times 100\% = \frac{R_A + R_B}{R_A + 2R_B} \times 100\% \qquad (3.5)$$

ในวงจรข้างต้น ถ้า R_B มีค่ามากเมื่อเทียบกับ R_A ความถี่การทำงานของวงจรก็จะกำหนดด้วยค่าของ R_B และ R_A และรูปคลื่นเอาท์พุทที่สมมาตรก็จะเกิดขึ้นและวงจรวงจรอะสเตเบิลมัลติไวเบรเตอร์ที่ใช้ในโครงงานนี้จะใช้ ตัวต้านทานปรับค่าได้ 50 k แทน R_B โดยมีรูปวงจรดังรูปที่ 3.7



รูปที่ 3.7 วงจรวงจรอะสเตเบิลมัลติไวเบรเตอร์ที่ใช้ในโครงงานนี้

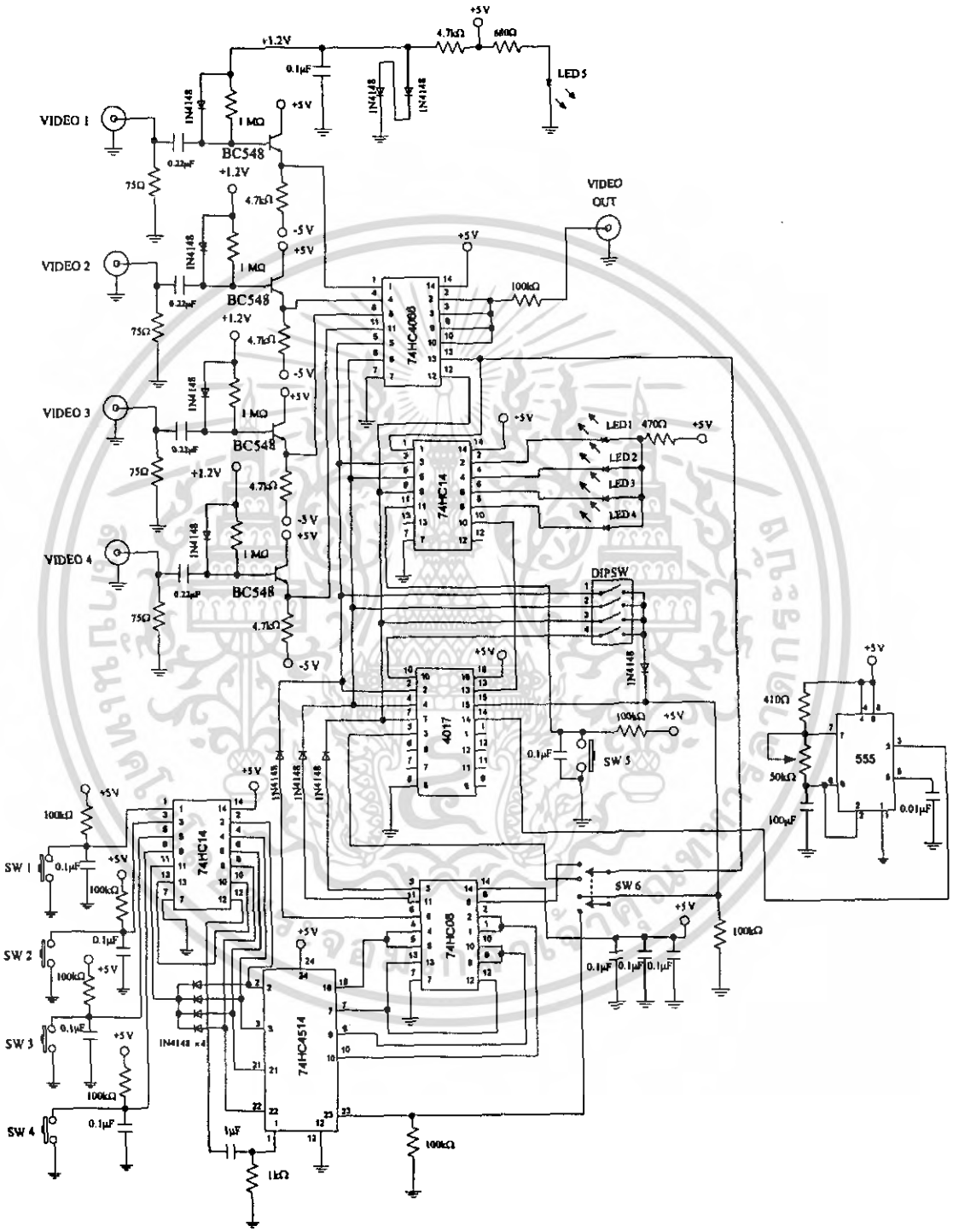
3.1.3 ส่วนจัดลำดับสัญญาณภาพแบบกำหนดตำแหน่งอินพุทได้



รูปที่ 3.8 วงจรส่วนจัดลำดับสัญญาณภาพแบบกำหนดตำแหน่งอินพุทได้

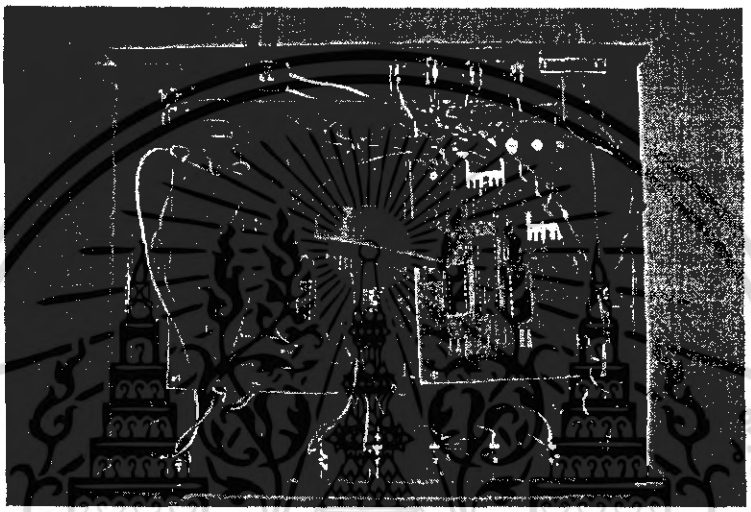
กลไกการทำงานของส่วนเลือกช่องสัญญาณภาพมี IC 74HC4514 ซึ่งเป็นไอซีดีโค้ดเดอร์แบบ 4-to-16 Line Decoder With Latch เป็นตัวถอดรหัสจากการกดปุ่ม สวิตช์ตัวที่ 1 - 4 ซึ่งจะไปควบคุมอิเล็กทรอนิกส์สวิตช์ภายใน IC 74HC4066 ที่ควบคุมตำแหน่งอินพุทของสัญญาณภาพโดยในทันทีที่กดสวิตช์ตัวที่ 1 - 4 ซึ่งแต่ละตัวถูกต่ออยู่กับตัวต้านทาน,ตัวเก็บประจุและเกตอินเวอร์เตอร์เพื่อลดผลจากการกระเพื่อมที่หน้าสัมผัสของสวิตช์ลงทำให้เอาท์พุทของเกตอินเวอร์เตอร์ที่ต่ออยู่กับสวิตนั้นๆเปลี่ยนจากเดิมที่เป็นลอจิก "0" ไปเป็นลอจิก "1"การเปลี่ยนแปลงที่เกิดขึ้นจะถูกส่งไปยังขาอินพุทตามขาที่ต่ออยู่กับสวิตช์ที่กด ในขณะเดียวกันลอจิกที่เปลี่ยนไปก็จะถูกส่งผ่านไปยังขา 1 ทำให้แลตช์ค่าที่อ่านได้จากขาอินพุททั้ง 4 ขาเอาไว้ และให้เอาท์พุท

ตามเงื่อนไขจากการกดสวิตช์โดยเอาท์พุทที่ได้มาก็จะมี IC 74HC08 ช่วยขับกระแสและมีไดโอดช่วยจัดทิศทางการไหลของข้อมูลเพื่อส่งไปควบคุมยังอิเล็กทรอนิกส์สวิตช์ที่ตำแหน่งอินพุทที่ต้องการนั้น



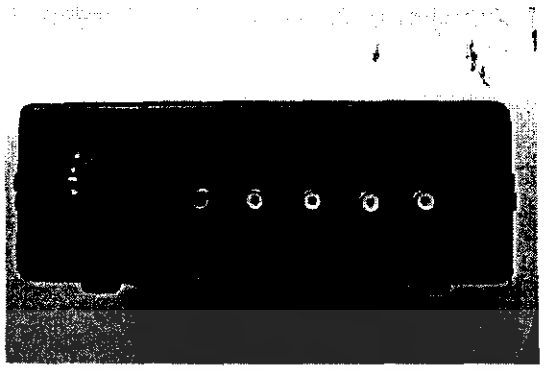
รูปที่ 3.9 วงจรรวม

วงจรรวมเกิดจากการนำวงจรส่วนต่างๆต่อเข้าด้วยกันจะทำให้ได้วงจรสลับช่องสัญญาณภาพที่สมบูรณ์สามารถทำงานได้ทั้งแบบอัตโนมัติหรือ แบบที่สามารถเลือกช่องสัญญาณโดยผู้ใช้ โดยกดสวิตช์ตัวที่ 6 โดยเมื่อกดสวิตช์เข้าจะทำให้ไฟเลี้ยงป้อนเข้าที่ขา 15 ของ IC 74HC4017 ซึ่งเป็นขารีเซ็ตทำให้การทำงานและการเปลี่ยนแปลงที่เอาท์พุทของตัวไอซีหยุดลงคือโหมดการทำงานแบบอัตโนมัติหยุดลงขณะเดียวกันที่ขา 23 ของ IC 74HC4514 ก็ได้รับลอจิก "1" ด้วยจึงทำให้โหมดที่เลือกโดยผู้ใช้พร้อมทำงาน



รูปที่ 3.10 แสดงวงจรภายในเครื่องสลับสัญญาณภาพ ที่ประกอบสมบูรณ์

รูปที่ 3.11 แสดงด้านหน้าของเครื่องสลับสัญญาณภาพที่เสร็จสมบูรณ์



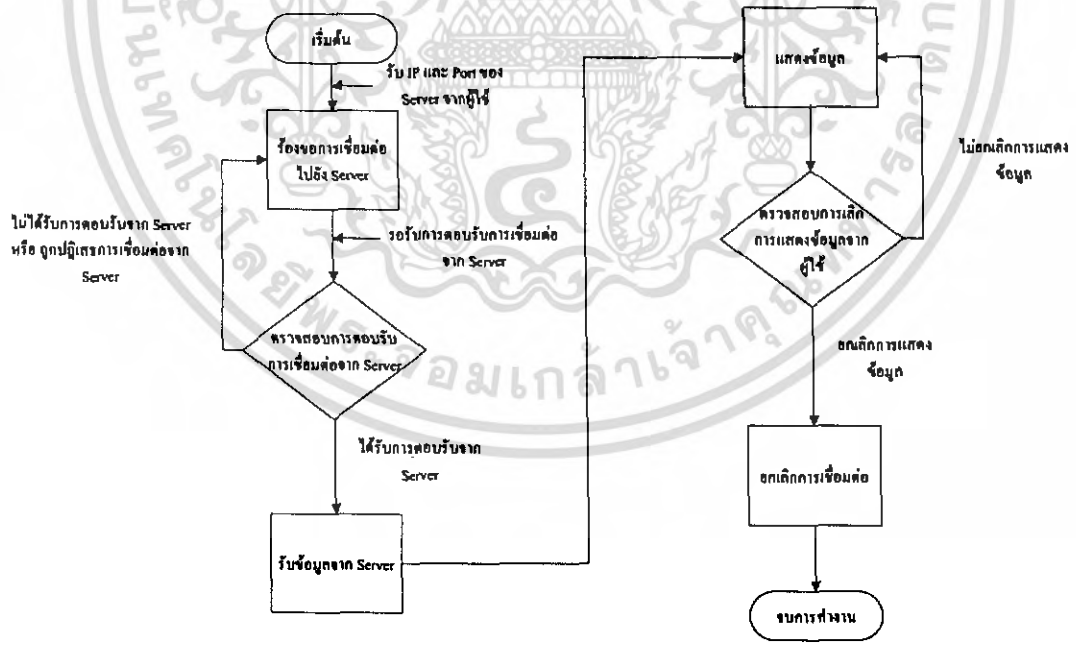
รูปที่ 3.12 แสดงด้านหลังของเครื่องสลับสัญญาณภาพที่เสร็จสมบูรณ์

3.2 โปรแกรม

โปรแกรมที่ผู้จัดทำออกแบบนั้นแบ่งออกเป็น 2 โปรแกรม ดังนี้

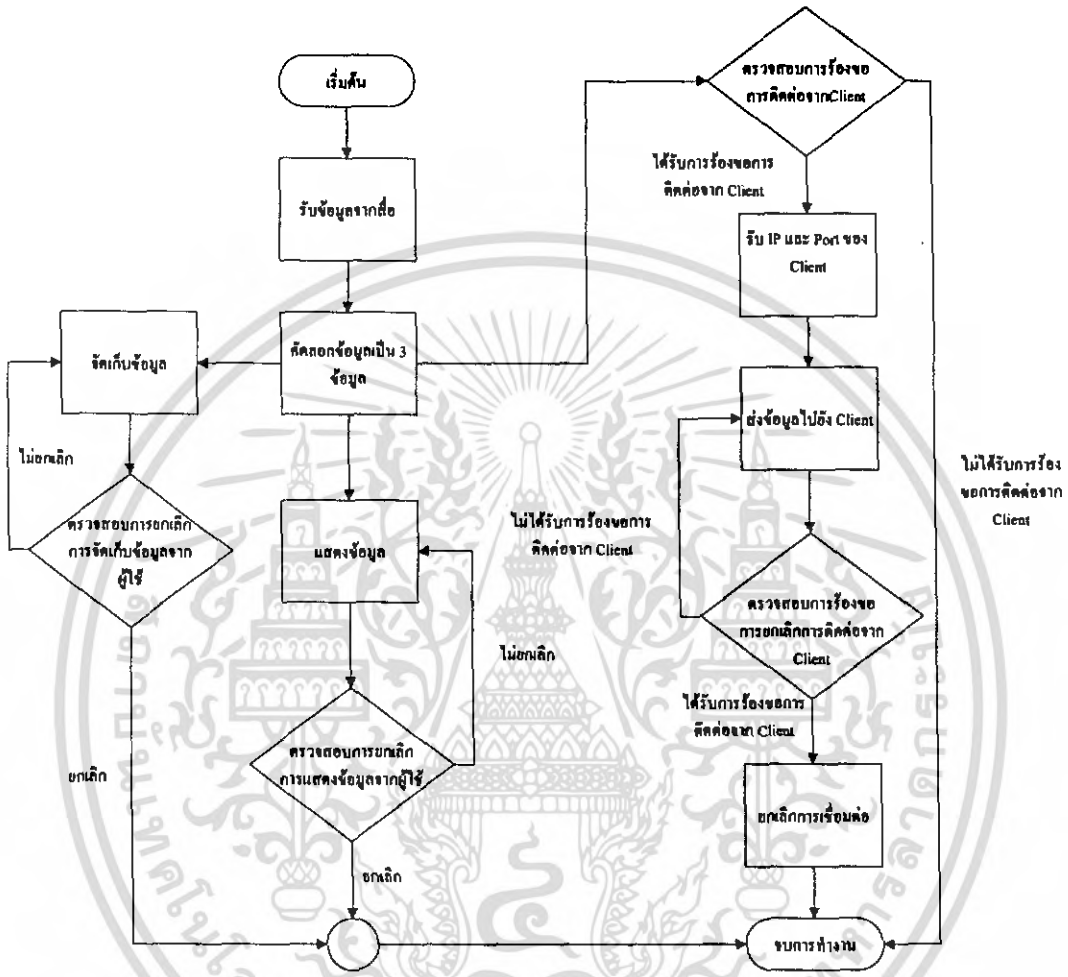
1. โปรแกรมรับข้อมูล
2. โปรแกรมส่งข้อมูล

3.2.1 โปรแกรมรับข้อมูล



รูปที่ 3.13 แผนภาพแสดงการทำงานของโปรแกรมด้านไคลเอนท์

3.2.2 โปรแกรมส่งข้อมูล



รูปที่ 3.14 แผนภาพแสดงการทำงานของโปรแกรมด้าน เซิร์ฟเวอร์

บทที่ 4

การทดลองและผลการทดลอง

4.1 การทดลองวงจรอะสเตเบิลมัลติไวเบรเตอร์

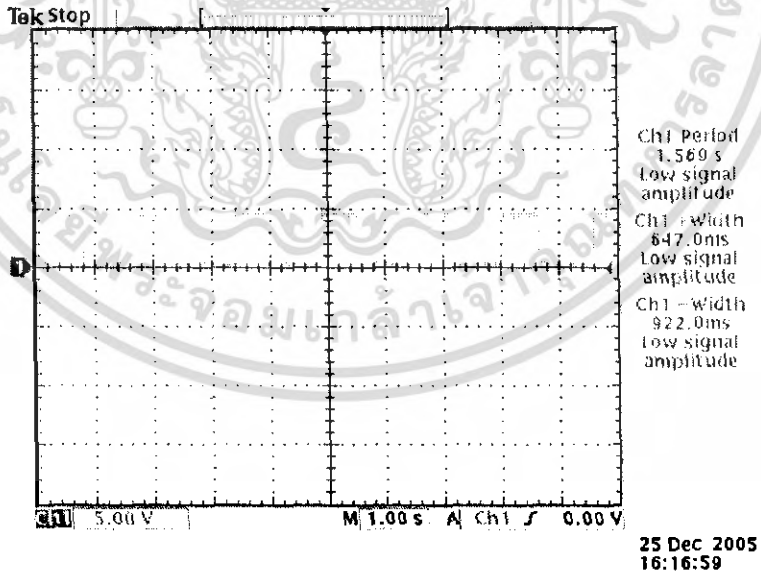
วัตถุประสงค์

- เพื่อทดลองการทำงานของวงจรอะสเตเบิลมัลติไวเบรเตอร์
- เพื่อเปรียบเทียบผลที่ได้จริงกับทฤษฎี

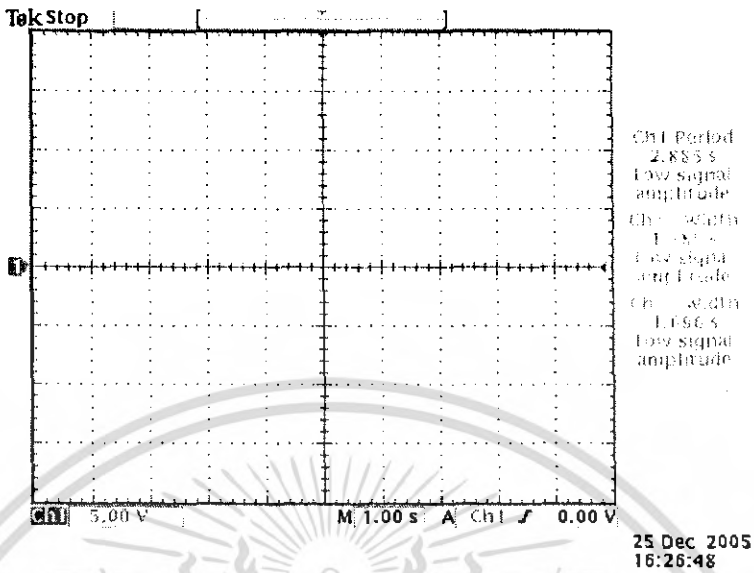
ขั้นตอนการทดลอง

1. ต่อวงจรอะสเตเบิลมัลติไวเบรเตอร์ดังรูปที่ 2.10
2. โดยกำหนดให้ $R_A = 470\Omega$, $R_B = 10k\Omega$, $C = 100\mu F$
3. นำออสซิลโลสโคป วัดที่ขา 3 ของไอซี 555
4. เปลี่ยนให้ $R_B = 20k\Omega$
5. นำออสซิลโลสโคป วัดที่ขา 3 ของไอซี 555 อีกครั้ง

ผลการทดลอง



รูปที่ 4.1 สัญญาณเอาท์พุท ที่ขา 3 ของไอซี 555 เมื่อ $R_B = 10k\Omega$



รูปที่ 4.2 สัญญาณเอาท์พุท ที่ขา 3 ของไอซี 555 เมื่อ $R_B = 20k\Omega$

จากรูปที่ 4.1 และ 4.2 สัญญาณเอาท์พุท ที่ขา 3 ของไอซี 555 มีคาบเท่ากับ 1.569 วินาที และ 2.885 วินาที ซึ่งค่าที่ได้จากทฤษฎีสามารถหาได้จากสมการที่ (3.3)

$$\begin{aligned} T &= C(R_B + R_{AB})\ln 2 \\ &= C(R_A + 2R_B)\ln 2 \end{aligned} \qquad (4.1)$$

เมื่อ $C = 100\mu F$, $R_A = 470\Omega$ และ $R_B = 10k\Omega$

จากสมการ (4.1)

$$\begin{aligned} T &= C(R_A + 2R_B)\ln 2 \\ &= (0.47 + 20)100\times 10^{-3}\ln 2 \\ &= 1.418 \text{ วินาที} \end{aligned}$$

เมื่อ $C = 100\mu F$, $R_A = 470\Omega$ และ $R_B = 20k\Omega$

จากสมการ (4.1)

$$\begin{aligned} T &= C(R_A + 2R_B)\ln 2 \\ &= (0.47 + 40)100\times 10^{-3}\ln 2 \\ &= 2.805 \text{ วินาที} \end{aligned}$$

4.2 การทดลองวงจรจัดลำดับสัญญาณภาพแบบอัตโนมัติ

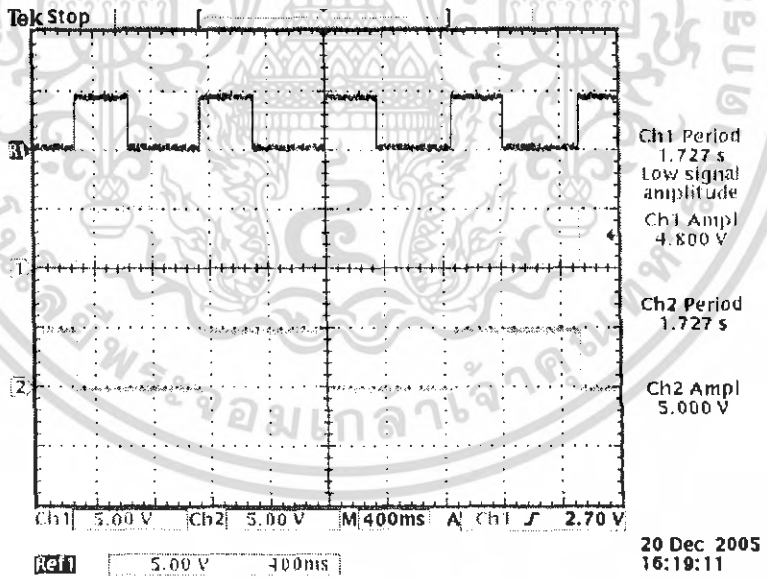
วัตถุประสงค์

- **เพื่อทดลองการทำงานของวงจรจัดลำดับสัญญาณภาพแบบอัตโนมัติ ที่ทำงานร่วมกับสัญญาณจากวงจรอะสเตเบิลมัลติไวเบรเตอร์**
- **เพื่อทดลองการทำงานของวงจรจัดลำดับสัญญาณภาพแบบอัตโนมัติเมื่อทำการกดสวิตช์ค้างไว้**

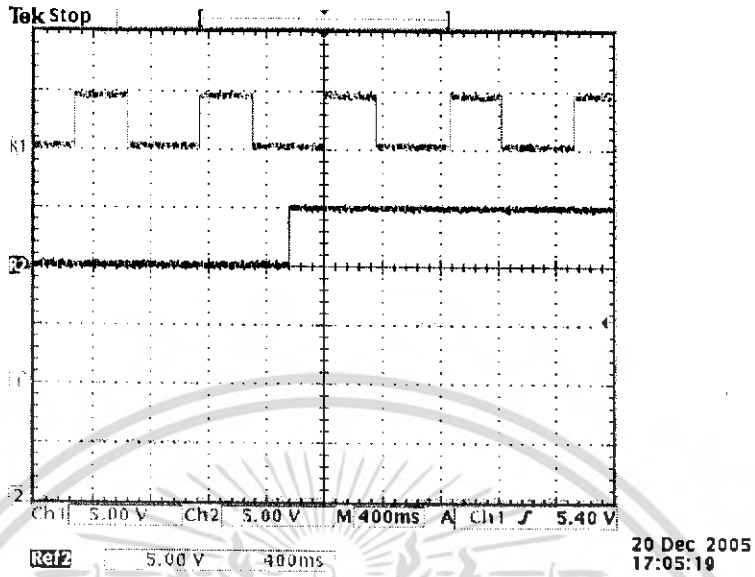
ขั้นตอนการทดลอง

1. ต่อวงจรจัดลำดับสัญญาณภาพแบบอัตโนมัติดังรูปที่ 3.4
2. กำหนดให้วงจรทำงาน 2 ช่องสัญญาณ
3. ต่อเอาท์พุทจากวงจรอะสเตเบิลมัลติไวเบรเตอร์ (T=868ms) เข้าที่ขา 14 ของไอซี 4017
4. นำออสซิลโลสโคป Ch1 และ Ch 2 วัดสัญญาณที่ขา 3 และขา 2 ของไอซี 4017 เทียบกับสัญญาณอินพุทที่ขา 14 ของไอซี 4017
5. จากข้อ 3 ทำการกดสวิตช์ที่ 5 ค้างไว้

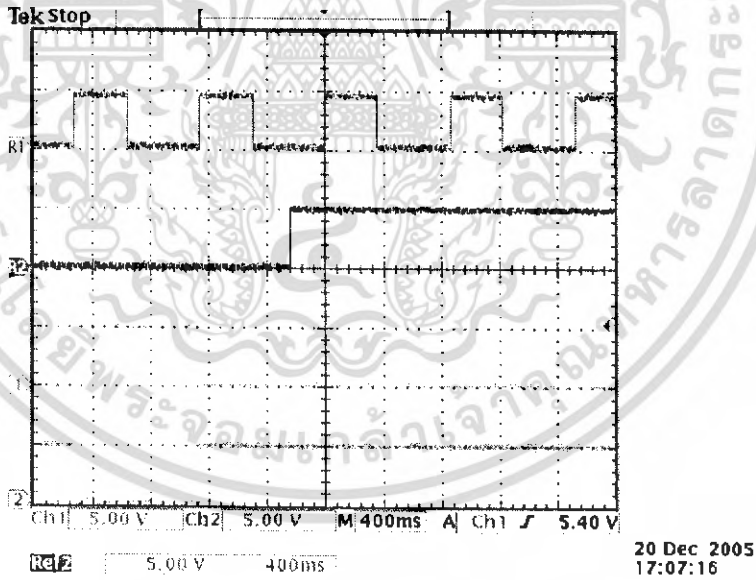
ผลการทดลอง



รูปที่ 4.3 สัญญาณที่ขา 3 และขา 2 เทียบกับสัญญาณอินพุทที่ขา 14



รูปที่ 4.4 แสดงสัญญาณที่ขา 3 และขา 2 เทียบกับสัญญาณอินพุทที่ขา 14
เมื่อทำการกดสวิตช์ที่ 5 ค้างไว้ ในขณะที่ขา 3 มีลอจิกเป็น 1



รูปที่ 4.5 แสดงสัญญาณที่ขา 3 และขา 2 เทียบกับสัญญาณอินพุทที่ขา 14
เมื่อทำการกดสวิตช์ที่ 5 ค้างไว้ในขณะที่ขา 2 มีลอจิกเป็น 1

4.3 การทดลองวงจรจัดลำดับสัญญาณภาพแบบกำหนดตำแหน่งอินพุทได้

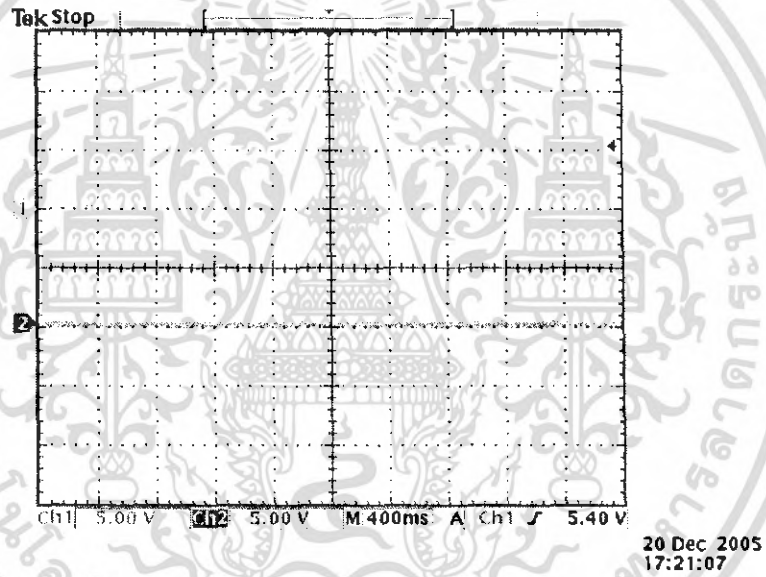
วัตถุประสงค์

- เพื่อทดลองการจัดลำดับสัญญาณภาพแบบกำหนดตำแหน่งอินพุทได้

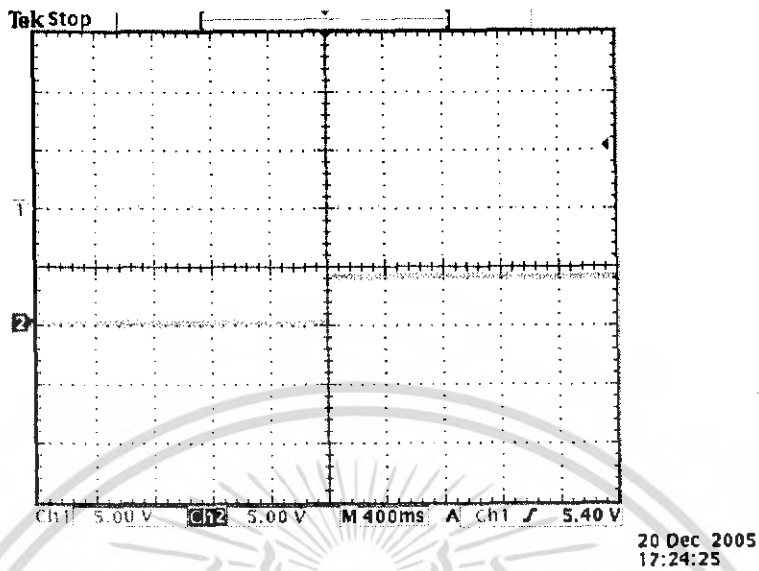
ขั้นตอนการทดลอง

1. ต่อวงจรจัดลำดับสัญญาณภาพแบบกำหนดตำแหน่งอินพุทได้ดังรูปที่ 3.8
2. กดสวิตช์ที่ 1 และสังเกตสัญญาณเอาท์พุทของ IC 74HC 4514 ที่ขา 9 และ 10
3. กดสวิตช์ที่ 2 และสังเกตสัญญาณเอาท์พุทของ IC 74HC 4514 ที่ขา 9 และ 10

ผลการทดลอง



รูปที่ 4.6 สัญญาณเอาท์พุทของ IC 74HC 4514 ที่ขา 9 และ 10 เมื่อกดสวิตช์ที่ 1



รูปที่ 4.7 สัญญาณเอาท์พุทของ IC 74HC 4514 ที่ขา 9 และ 10 เมื่อกดสวิตช์ที่ 2

4.4 การทดลองการรับส่งสัญญาณภาพ

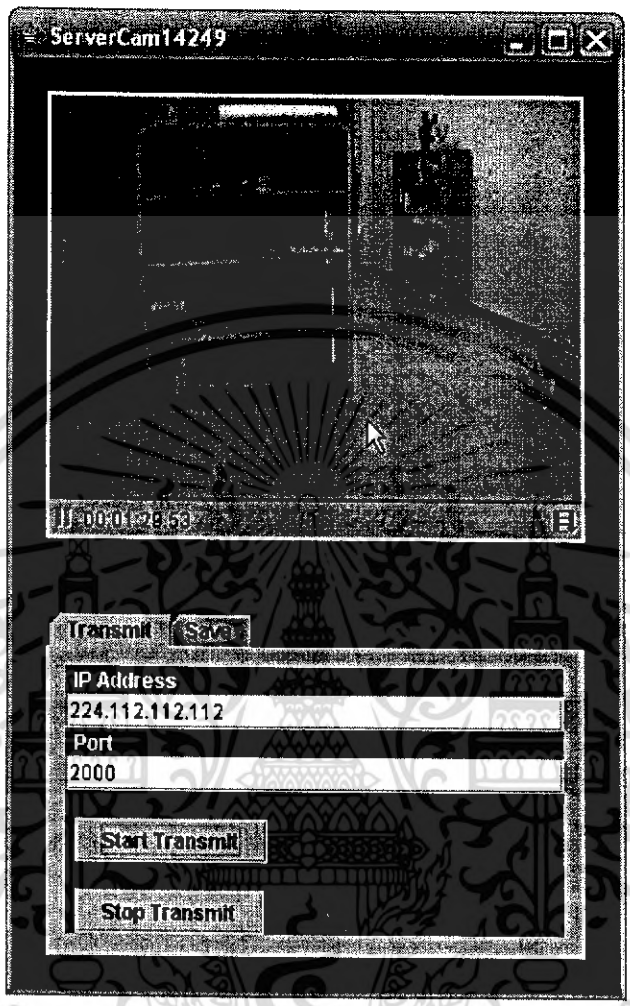
วัตถุประสงค์

- เพื่อทดลองการรับส่งสัญญาณภาพในเครือข่ายคอมพิวเตอร์ โดยใช้โปรแกรมที่สร้างขึ้น

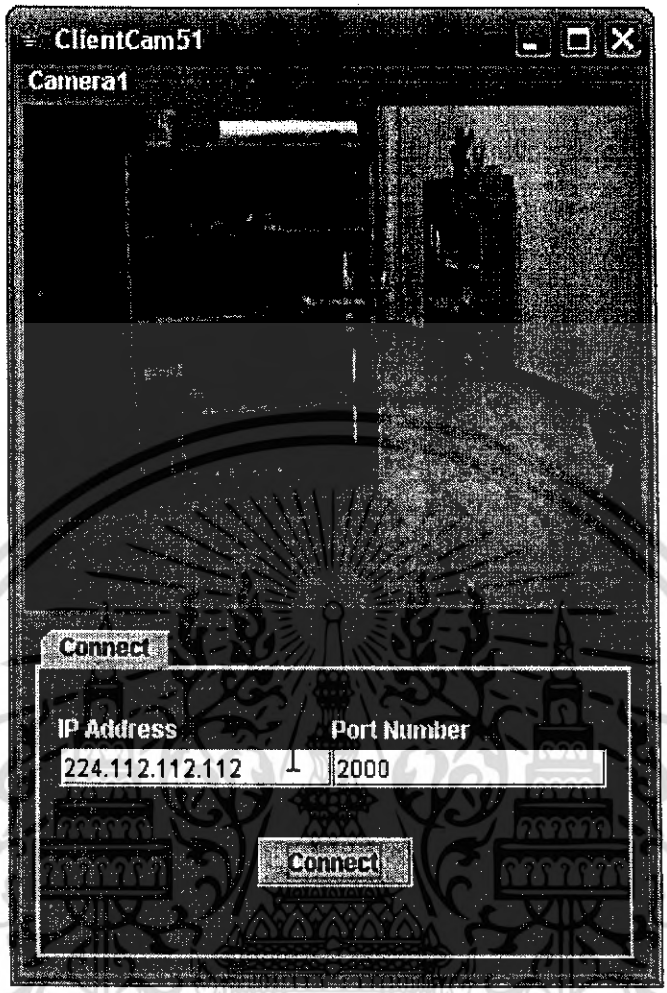
ขั้นตอนการทดลอง

1. เปิดโปรแกรม JCreator ขึ้นมา ทั้งในเครื่องที่จะให้ส่งสัญญาณภาพและเครื่องที่ทำการรับสัญญาณภาพ
2. ในด้านส่งทำการคอมไพลไฟล์ ServerCam14249 และในด้านรับคอมไพล์ไฟล์ ClientCam51 เมื่อไม่มีข้อผิดพลาดเกิดขึ้น การคอมไพล์โปรแกรมจะเสร็จสมบูรณ์ ให้ทำการ Execute ไฟล์ทั้งสอง
3. ในโปรแกรมด้านส่งให้ระบุ IP address และ Port ของด้านรับ คลิ๊กปุ่ม Start Transmit เพื่อส่งสัญญาณภาพ และคลิ๊กปุ่ม Stop Transmit เมื่อต้องการหยุดส่งสัญญาณภาพและในโปรแกรมด้านรับให้ระบุ IP address และ Port ของโปรแกรมด้านรับ คลิ๊กปุ่ม Connect
4. ในด้านส่งเมื่อต้องการบันทึกภาพ ให้คลิ๊กแท็บ Save ทำการระบุที่อยู่ของไฟล์ที่ต้องการบันทึก และคลิ๊กปุ่ม Start Save เมื่อต้องการยกเลิกการบันทึกให้คลิ๊กปุ่ม Stop Transmit

ผลการทดลอง



รูปที่ 4.8 แสดงจอภาพของโปรแกรมด้านส่งขณะส่งสัญญาณภาพ โดยด้านรับมี IP address เป็น 224.112.112.112 และ Port เป็น 2000



รูปที่ 4.9 แสดงจอภาพของโปรแกรมด้านรับ
โดยด้านส่งมี IP address เป็น 224.112.112.112 และ Port เป็น 2000

บทที่ 5
บทวิจารณ์และบทสรุป

5.1 สรุปผลการทดลอง

โครงงานนี้เป็นการสร้างระบบกล้องรักษาความปลอดภัย ซึ่งสามารถใช้สังเกตการณ์ในบริเวณต่างๆที่ต้องการได้ ผ่านเครือข่ายอินเทอร์เน็ต โดยส่วนฮาร์ดแวร์ ทำหน้าที่ในการสลับสัญญาณภาพจากกล้องที่ต่ออยู่ก่อนส่งสัญญาณต่อไปยังคอมพิวเตอร์ โดยสามารถทำการสลับสัญญาณจากกล้องได้สูงสุด 4 ตัว และ ทำงานทั้งโหมดสวิตช์ แบบอัตโนมัติ และแบบเลือกดูกล้องที่ต้องการจากผู้ใช้ ตัวฮาร์ดแวร์ แสดงผลด้วย LED เพื่อแสดงว่าภาพที่ปรากฏเป็นของกล้องตัวใด สามารถปรับความเร็วในการสวิตช์แบบอัตโนมัติ ได้ด้วยการปรับตัวต้านทานปรับค่าได้ โดยสามารถปรับให้แสดงแต่ละกล้องนานได้ตั้งแต่ 0.033 วินาที ถึง 6.90 วินาที และสามารถหยุดการสลับช่องสัญญาณเพื่อรับชมภาพจากกล้องที่กำลังแสดงบนหน้าจอคอมพิวเตอร์ โดยการกดสวิตช์ตัวที่ 5 ค้างไว้ได้นานตามต้องการ ส่วนการเปลี่ยนโหมดเพื่อเลือกดูกล้องที่ต้องการจากผู้ใช้สามารถทำได้ด้วยการกดสวิตช์ตัวที่ 6 จากนั้นทำการกดสวิตช์ตัวที่ 1 - 4 เพื่อเลือกกล้องในการรับชมได้ตามต้องการ

- การทดลองวงจรอะสเตเบิลมัลติไวเบรเตอร์ในครั้งที่หนึ่งพบว่ามีคาบเวลา เท่ากับ 1.569 วินาที และครั้งที่สองเท่ากับ 2.885 วินาที ซึ่งเมื่อเทียบกับที่คำนวณไว้คือ 1.418 วินาทีสำหรับการทดลองครั้งที่หนึ่ง และ 2.805 วินาที สำหรับการทดลองครั้งที่สองพบว่าค่าที่ได้ใกล้เคียงกัน

- การทดลองวงจรส่วนจัดลำดับสัญญาณภาพแบบอัตโนมัติ พบว่าผลที่ได้ คือเอาท์พุทจะเปลี่ยนทุกๆ คาบของสัญญาณอินพุทจากขา 14 และเอาท์พุทจะคงค่าเมื่อ ลอจิกที่ขา 13 เป็น "1"

- การทดลองวงจรส่วนจัดลำดับสัญญาณภาพแบบกำหนดตำแหน่งอินพุทได้ พบว่าผลที่ได้เมื่อกดสวิตช์ตัวที่ 1 ก็ทำให้เอาท์พุทที่ขา 9 ของ IC 74HC4514 เป็น "1" ค่าเอาท์พุทอื่นเป็น "0" และเอาท์พุทจะคงค่าไว้จนกว่าจะกดสวิตช์ตัวอื่นๆ

- การทดลองการรับส่งสัญญาณภาพ พบว่าสามารถรับส่งภาพเคลื่อนไหวไปยังเป้าหมายที่วางไว้ได้ และสามารถเก็บบันทึกภาพเคลื่อนไหวในลักษณะ วีดีโอได้

5.2 วิจารณ์ผลการทดลอง

สำหรับผลการวงจรอะสเตเบิลมัลติไวเบรเตอร์ ค่าที่ได้ไม่เท่ากับค่าที่คำนวณไว้ อาจเนื่องมาจากค่าความผิดพลาดจากตัวอุปกรณ์ที่ใช้ และ ค่าความผิดพลาดจากเครื่องมือวัด

ผลการทดลองวงจรส่วนจัดลำดับสัญญาณภาพแบบอัตโนมัติ และ วงจรส่วนจัดลำดับสัญญาณภาพแบบกำหนดตำแหน่งอินพุทได้ ให้ผลการทดลองตามที่คาดหมายไว้ทุกประการ

เนื่องจากการสลับสัญญาณภาพจากแต่ละกล้อง ถูกควบคุมโดยส่วนของฮาร์ดแวร์ทั้งหมด ดังนั้นผู้รับชมผ่านเครือข่ายอินเทอร์เน็ต จึงไม่สามารถสามารถควบคุมได้ด้วยตัวเอง

การรับส่งสัญญาณภาพ การรับส่งข้อมูลเป็นไปด้วยดี ในเครือข่ายแบบท้องถิ่น นอกจากนี้ยังสามารถบันทึกข้อมูลภาพได้เพื่อใช้เมื่อต้องการดูเหตุการณ์ย้อนหลัง

เอกสารอ้างอิง

1. Elliotte Rusty Harold , "Java Network Programming" O'Reilly , October 2004
2. Delton T.Horn, "Analog switches" Tab Books , 1990
3. http://www.sun.com
4. กิตติ ภักดีวัฒนะกุล, "Java ฉบับโปรแกรมเมอร์" กรุงเทพฯ : เคทีพีคอมพ์แอนด์คอนซัลท์, 2542
5. กิตติ ภักดีวัฒนะกุล, "คัมภีร์ JAVA" กรุงเทพฯ : เคทีพี คอมพ์ แอนด์ คอนซัลท์, 2546
6. Herman Kruegle, "CCTV Surveillance" Butterworth – Heinemann,1995
7. Vlado Damjanovski, "CCTV" Butterworth – Heinemann, 2000
8. ดร.มงคล เดชนครินทร์ และ ดร.ชาตรี ศรีไพพรรณ, "อิเล็กทรอนิคส์พื้นฐาน" บริษัท ซีเอ็ดยูเคชั่น จำกัด, 2521
9. Kelly Caudle and Kelly Cannon, "CCNA" Thomson/Course Technology, 2004

ภาคผนวก

โปรแกรม Server

```
/**
 * ServerCam14249.java
 */
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
import javax.media.*;
import javax.media.control.*;
import javax.media.format.*;
import javax.media.protocol.*;
import javax.media.Format.*;
import javax.swing.border.*;
import java.io.*;
import java.net.*;

import java.io.File;
import java.io.IOException;
import com.sun.media.format.WavAudioFormat;
import javax.media.datasink.*;

import java.util.Observable;

//for recieving the rtp stream
import javax.media.rtp.*;
import javax.media.rtp.event.*;
import javax.media.rtp.rtcp.*;
import java.net.InetAddress;
import com.sun.media.rtp.RTPSessionMgr;
import java.lang.Integer;
//for tagging the saved files
import java.util.Calendar;
import java.util.Date;
import java.util.TimeZone;
import java.util.GregorianCalendar;
import java.util.SimpleTimeZone;


/** asdf */
public class ServerCam14249 implements javax.media.ControllerListener{

    private static Processor player;
    private static ServerCam14249GUI mdGUI;
    private Object waitSync = new Object();
    private boolean stateTransitionOK = true;
    private static MotDetCodec mdCodec;
    private static Clone clone = null;
    private static Clone clone1 = null;
    private static Clone clone3 = null;

    private static VideoTransmit vt;
    private static DataSource cloneableDataSource = null;
    private static DataSource clonedDataSource = null;
    private static DataSource origDataSource = null;
    private static SaveVideo sav;
```

```
//      private SurveillanceToolObservable observable;

    /** Returnes the processor of this ServerCam14249.
     * @return a processor
     */
    public Processor getProcessor(){
        return player;
    }


    /** Sets up the media file for "url" and starts the playback of
that
     * media file.
     * @param url the url to a media file
     */
    public void open( String url ) {
        MediaLocator ml;

        // create a media locator from the url
        if ((ml = new MediaLocator(url)) == null) {
            System.out.println("Cannot build URL for " + url);
            System.exit(0);
        }

        try {
            origDataSource = Manager.createDataSource(ml);
        } catch (Exception e) {
            System.err.println("Cannot create DataSource from " + ml);
            System.exit(0);
        }
        cloneableDataSource =
Manager.createCloneableDataSource(origDataSource);
        if (cloneableDataSource == null) {
            System.err.println("Cannot clone DataSource");
            System.exit(0);
        }
        clone = new Clone();

        if (!clone.open(cloneableDataSource))
            System.exit(0);

        try {
            player = clone.getProcessor();
        }
        catch ( Exception e ) {
            System.out.println(e);
            System.out.println("Could not create player for " + ml);
            System.exit(0);
        }


        player.addControllerListener( this );

        // put the processor into configured state.
        player.configure();
```

```
        if ( !waitForState( player.Configured ) ) {
            System.err.println("Failed to configure the
processor.ipAddress");
            System.exit(0);
        }

        // to use the processor as a player
        player.setContentDescriptor(null);

        // obtain the track controls.
        TrackControl tc[] = player.getTrackControls();
        if (tc == null) {
            System.err.println("Failed to obtain track controls from
the processor.");
            System.exit(0);
        }

        // search for the video track control.
        TrackControl videoTrack = null;
        for (int i = 0; i < tc.length; i++) {
            if (tc[i].getFormat() instanceof VideoFormat) {
                videoTrack = tc[i];
                break;
            }
        }

        // if there is no video track control we exits
        if (videoTrack == null) {
            System.err.println("The input media does not contain a video
track.");
            System.exit(0);
        }

        // set up the codec chain for playback
        try{
            mdCodec = new MotDetCodec();
            Codec[] codec = {mdCodec};
            videoTrack.setCodecChain( codec );
        } catch(Exception e){
        }

        // realize the processor.
        player.prefetch();

        if (!waitForState(player.Prefetched)) {
            System.err.println("Failed to realize the processor.");
            System.exit(0);
        }

        // start media file playback
        if (player != null) {
            player.start();
        }

        /* create second clone to transmit */
        clone = new Clone();
```

```
        if
(!clone.open(((SourceCloneable)cloneableDataSource).createClone()))
            System.exit(0);

        clone = new Clone();
        if
(!clone.open(((SourceCloneable)cloneableDataSource).createClone()))
            System.exit(0);
    }


     /**
     * Block until the processor has transitioned to the given state.
     * Return false if the transition failed.
     *
     *
     * @param state the wanted state for the processor
     */
    boolean waitForState(int state) {
        // waits for a state confirmation for the player
        synchronized (waitSync) {
            try {
                while (player.getState() != state && stateTransitionOK)
                    waitSync.wait();
            } catch (Exception e) {}
        }
        return stateTransitionOK;
    }

     /** The update method for this ControllerListener, sets he proper
actions
     * when updated.
     *
     * @param event the occured event
     */
    public void controllerUpdate(ControllerEvent event) {

        // if the playeris dead - we leave
        if (player == null)
            return;

        if ( event instanceof ConfigureCompleteEvent ||
             event instanceof RealizeCompleteEvent   ||
             event instanceof PrefetchCompleteEvent ) {
            synchronized ( waitSync ) {
                stateTransitionOK = true;
                waitSync.notifyAll();
            }
        } else if ( event instanceof ResourceUnavailableEvent ) {
            synchronized ( waitSync ) {
                stateTransitionOK = false;
                waitSync.notifyAll();
            }
        // close the application
        } else if ( event instanceof EndOfMediaEvent       ||
                    event instanceof ControllerClosedEvent ||
                    event instanceof ControllerErrorEvent ) {
```

```
            player.close();
            System.exit(0);
        }
    }


     /** sets the number of frames between input and reference in the
codec
     * @param frames the number of frames between input and reference
in the codec
     */
    public void setFrameSpace(int frames) {
        mdCodec.setFrameSpace(frames);
    }
    /**
     * Sets the text to display in the video.
     *
     * @param string text to display in video.
     */
    public void setInVideoText(String string) {
        mdCodec.setVideoMsg(string);
    }
    /** Returns the movement percentage in the output media file
     *
     * @return movement percentage
     */
    public static int getMovementPercentage() {
        return mdCodec.getMovementPercentage();
    }

    /** Sets up and starts the detector application
     * @param args arguments for the application
     */
    public static void main( String args[] ) {
        ServerCam14249 md = new ServerCam14249();
        md.open("vfw://0");

        mdGUI = new ServerCam14249GUI( md );
        mdGUI.display();

        // checks every 50 ms, if we let the alarm go off
        int delay = 50; //milliseconds
        ActionListener taskPerformer = new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                if(mdGUI.alarmValue <= getMovementPercentage()){

                }else{

                }
                mdGUI.display();
            }
        };

        new Timer(delay, taskPerformer).start();
```

```
    }


    /**
     * ServerCam14249GUI.java
     *
     * This is a the GUI for the motion detection application. Is views
the
     * output from the motion detection and it has special components
for
     * setting options for the motion detection.
     *
     */
    private static class ServerCam14249GUI extends JFrame{

        int width=400, height = 540;
        private JPanel  mainPanel;// set up orientation of labels and
slider relative to each other
        private JPanel  videoPanel;
        private JPanel  advancedPanel;

        private JPanel  transmitPanel;
        private JPanel  savePanel;

        private JTabbedPane tabbedPane;


        private JTextField transmitPort;
        private JTextField transmitIP;
        private JLabel  transmitPortLabel;
        private JLabel  transmitIPLabel;
        private JLabel  transmitLabel1;
        private JLabel  transmitLabel2;
        private JLabel  transmitQualityLabel;
        private JButton transmitStartButton;
        private JButton transmitStopButton;
        private JSlider qualitySlider;

        private JTextField saveLocal;
        private JLabel  saveLocalLabel;
        private JLabel  saveLocal2Label;
        private JLabel  saveLocal3Label;
        private JLabel  saveLocal4Label;
        private JLabel  saveLocal5Label;
        private JLabel  saveLocal6Label;
        private JButton saveStartButton;
        private JButton saveStopButton;

        private Processor processor;
        private ServerCam14249 MD;
        public int alarmValue = 0;
        private boolean alarmFlag = false;

       String urlstring;
        private static String location[] = new String[3];
        private static String fileName[] = new String[3];
```

```
        private static SaveRun[] save = new SaveRun[3];
        private static Thread[] saveThread = new Thread[3];
        private static SaveVideo[] saveVideo = new SaveVideo[3];

        private String getStorageLocation(){
        return saveLocal.getText();
        }


                /**
         * Returns the storage location set by setStorageLocation() in
this class.
         *
         * @return the storage location as a String
         */

        /**
         * Names the file and starts saving to it.
         *
         * @param index the index of the stream to save
         */
        private void startSaving(int index){
            //create a filename
            fileName[index] = "Video" + (index+1) + "--";
            //the code forgetting the timetag is gotten from the
example for
            //GregorianCalendar class in java API
            // get the supported ids for GMT+01:00 (Sweden time)
            String[] ids = TimeZone.getAvailableIDs(1 * 60 * 60 *
1000);
            // if no ids were returned, something is wrong. get out.
            if (ids.length == 0)
                System.exit(0);
            // create a Sweden time zone
            SimpleTimeZone pdt = new SimpleTimeZone(1 * 60 * 60 * 1000,
ids[0]);
            // set up rules for daylight savings time
            pdt.setStartRule(Calendar.APRIL, 1, Calendar.SUNDAY, 2 * 60
* 60 * 1000);
            pdt.setEndRule(Calendar.OCTOBER, -1, Calendar.SUNDAY, 2 *
60 * 60 * 1000);
            // create a GregorianCalendar with the Sweden time zone
            // and the current date and time
            Calendar calendar = new GregorianCalendar(pdt);
            Date trialTime = new Date();
            calendar.setTime(trialTime);
            //set the tag for the file name
            fileName[index] = fileName[index] +
calendar.get(Calendar.YEAR) + "-"
            + calendar.get(Calendar.MONTH) + "-"
            + calendar.get(Calendar.DATE) + " "
            + calendar.get(Calendar.HOUR_OF_DAY) + "."
            + calendar.get(Calendar.MINUTE) + "."
            + calendar.get(Calendar.SECOND) + ".mov";

            //clone datasource to save the clone
            clone = new Clone();
```

```
            if
(!clone.open(((SourceCloneable)cloneableDataSource).createClone())){
                System.err.println("Cannot create clone for saving
stream: " + index);
                return;
            }

            //start saving
            saveVideo[index] = new SaveVideo(clone.getProcessor(),
location[index] + fileName[index]);
            saveVideo[index].startSave();
            //initiate a thread that stops the saving after 10s
            save[index] = new  SaveRun(index, 14000, saveVideo[index]);
            //save[index].addObserver(this);//add abserver to the
runnable object
            save[index].startSave();
            saveThread[index] = new Thread(save[index]);
            saveThread[index].start();
        }


        /**
         * Creates a new instance of
MotionDetecttransmitStopButtonionGUI
         *
         * @param md the motion detector application
         */
        public ServerCam14249GUI( ServerCam14249 md ) {

            super( "ServerCam14249" );
            MD = md;
            // retrieve the processor
            this.processor = md.getProcessor();

            // add closing functionality
            addWindowListener(new WindowAdapter() {
                public void windowClosing( WindowEvent event ) {
                    processor.close();
                    System.exit(0);
                }
            });

            // set up the view of the output from the motion detection
            videoPanel = new JPanel();
            videoPanel.setBackground( Color.BLACK );
            LineBorder lb = new LineBorder( Color.BLACK, 20, false );
            LineBorder lb2 = new LineBorder( Color.WHITE,2,  false );
            videoPanel.setBorder(BorderFactory.createCompoundBorder(
lb, lb2 ) );
            videoPanel.setLayout( new BorderLayout() );
            Component cc, vc;
            if ((vc = processor.getVisualComponent()) != null) {
                videoPanel.add("Center", vc);
            }

            if ((cc = processor.getControlPanelComponent()) != null) {
```

```
                videoPanel.add("South", cc);
            }


            // set up a button for Starting Transmit
            // adding a handler
            transmitStartButton = new JButton( "Start Transmit" );
            transmitStartButton.setSize( new Dimension( 35, 30 ) );
            transmitStartButton.addActionListener(new ActionListener()
{
                public void actionPerformed(ActionEvent event) {
                    //VideoTransmit
                    String ipAddress = transmitIP.getText();
                    String port = transmitPort.getText();
                    float quality = ((float)(qualitySlider.getValue()))
;
                    vt = new VideoTransmit(clone.getProcessor(),
ipAddress, port, quality);
                    vt.start();
                }
            });

            // set up a button for Stopping Transmit
            // adding a handler
            transmitStopButton = new JButton( "Stop Transmit" );
            transmitStopButton.setSize( new Dimension( 35, 30 ) );
            transmitStopButton.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent event) {
                    vt.stop();
                }
            });


            transmitPanel = new JPanel();
            transmitPanel.setBackground(Color.BLACK);
            transmitPanel.setLayout(new BoxLayout(transmitPanel,
BoxLayout.Y_AXIS));
            LineBorder lb0 = new LineBorder(Color.ORANGE, 10, false);
            transmitPanel.setBorder(lb0);

            transmitIPLabel = new JLabel("IP Address");
            transmitIPLabel.setAlignmentX(Component.LEFT_ALIGNMENT);
            transmitIPLabel.setForeground(Color.WHITE);
            transmitPortLabel = new JLabel("Port");
            transmitPortLabel.setAlignmentX(Component.LEFT_ALIGNMENT);
            transmitPortLabel.setForeground(Color.WHITE);

            transmitLabel1 = new JLabel("A");
            transmitLabel1.setAlignmentX(Component.LEFT_ALIGNMENT);
            transmitLabel2 = new JLabel("B");
            transmitLabel2.setAlignmentX(Component.LEFT_ALIGNMENT);

            transmitQualityLabel = new JLabel();

transmitQualityLabel.setAlignmentX(Component.LEFT_ALIGNMENT);

            transmitIP = new JTextField();
```

```
            //transmitIP.setSize(35,30);
            transmitPort = new JTextField();
            transmitIP.setText("224.112.112.112");
            transmitPort.setText("2000");

            qualitySlider = new JSlider(JSlider.HORIZONTAL, 0, 100,
50);



            transmitPanel.add(transmitIPLabel);
            transmitPanel.add(transmitIP);
            transmitPanel.add(transmitPortLabel);
            transmitPanel.add(transmitPort);

            transmitPanel.add(transmitLabel2);
            transmitPanel.add(transmitStartButton);
            transmitPanel.add(transmitLabel1);
            transmitPanel.add(transmitStopButton);



            savePanel = new JPanel();
            savePanel.setBackground(Color.BLACK);
            savePanel.setLayout(new BoxLayout(savePanel,
BoxLayout.Y_AXIS));
            LineBorder lb5 = new LineBorder(Color.ORANGE, 5, false);
            savePanel.setBorder(lb5);
            saveLocalLabel = new JLabel("Save Location");
            saveLocalLabel.setAlignmentX(Component.LEFT_ALIGNMENT);
            saveLocalLabel.setForeground(Color.WHITE);
            saveLocal2Label = new JLabel("A");
            saveLocalLabel.setAlignmentX(Component.LEFT_ALIGNMENT);
            saveLocal3Label = new JLabel();
            saveLocal3Label.setText("B");
            saveLocalLabel.setAlignmentX(Component.LEFT_ALIGNMENT);
            saveLocal4Label = new JLabel();
            saveLocal4Label.setText("B");
            saveLocalLabel.setAlignmentX(Component.LEFT_ALIGNMENT);
            saveLocal5Label = new JLabel();
            saveLocal5Label.setText("B");
            saveLocalLabel.setAlignmentX(Component.LEFT_ALIGNMENT);
            saveLocal6Label = new JLabel();
            saveLocal6Label.setText("B");
            saveLocalLabel.setAlignmentX(Component.LEFT_ALIGNMENT);
            saveLocal = new JTextField();
            saveLocal.setText("c:\\AppServ\\www\\");

                        saveStartButton = new JButton( "Start Save" );
            saveStartButton.setSize( new Dimension( 35, 30 ) );
                        saveStartButton.addActionListener(new
ActionListener() {
                public void actionPerformed(ActionEvent evt) {
                String saveLocal1 = saveLocal.getText();
                System.err.println("Save As "+ saveLocal1);
```

```
                sav = new SaveVideo(clone.getProcessor(), ipAddress,
port, quality);
                sav.start();

                //location[index] = (new
File(obs.getStorageLocation())).toURL().toString();
                }
            });

            saveStopButton = new JButton( "Stop Save" );
            saveStopButton.setSize( new Dimension( 35, 30 ) );
                    saveStopButton.addActionListener(new
ActionListener() {
                public void actionPerformed(ActionEvent evt) {
                System.err.println("Stop Save");
                }
            });

            savePanel.add(saveLocalLabel);
            savePanel.add(saveLocal);
            savePanel.add(saveLocal2Label);
            savePanel.add(saveLocal3Label);
            savePanel.add(saveLocal4Label);
            savePanel.add(saveStartButton);
            savePanel.add(saveLocal5Label);
            savePanel.add(saveStopButton);
            savePanel.add(saveLocal6Label);



            // set up a tabbedpane for all the user interactive
components
            tabbedPane = new JTabbedPane();
            tabbedPane.setBorder( new LineBorder( Color.BLACK, 20,
false ) );
            tabbedPane.addTab( "Transmit", transmitPanel );
            tabbedPane.addTab( "Save", savePanel );

            // set up the main container for the GUI
            Container c = getContentPane();
            c.setLayout( new BoxLayout( c, BoxLayout.Y_AXIS) );
            c.setBackground( Color.BLACK);
            c.add(videoPanel);
            c.add(tabbedPane);
        }


        /**
         * Packs and displays the GUI.
         */
        public void display() {
            pack();
            show();
        }
}
}
```

โปรแกรมด้าน Client

```
/**
 * ClientCam51.java
 */

import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;
import javax.swing.event.*;
import javax.media.*;
import javax.media.control.*;
import javax.media.format.*;
import java.util.Observer;
import javax.media.protocol.*;
import javax.media.Format.*;

import javax.media.rtp.*;
import javax.media.rtp.event.*;
import javax.media.rtp.rtcp.*;
import java.net.InetAddress;
import com.sun.media.rtp.RTPSessionMgr;
import java.lang.Integer;

/**

 */

public class ClientCam51{
    private static ClientCam51GUI gui;
    private static Object waitSync = new Object();
    private static boolean stateTransitionOK = true;

    //one for each videoPanel (same index points to the same panel)
    private static Processor[] p = new Processor[3];
    private static DataSource[] origDataSource = new DataSource[3];
    private static DataSource[] cloneableDataSource = new
DataSource[3];
    private static Clone[] clone = new Clone[3];
    private static RTPSessionMgr[] mngrSession = new RTPSessionMgr[3];
    private static String location[] = new String[3];
    private static String fileName[] = new String[3];

    /**
     * Starts up this application
     */
    public static void main( String[] args){
        gui = new ClientCam51GUI();
        gui.display();
    }

    /**

     *
```

```
     */
    private static class ClientCam51GUI extends JFrame implements
ControllerListener, Observer, ReceiveStreamListener{

        private JPanel mainPanel;//the panel to contain all components
        private VideoPanel videoPanel0, videoPanel1;//two panels
contains all other components
        private SurveillanceToolObservable observable0;//used as a
comon observable object for all components in videoPanel1


        /**
         * Creates a new instance of the GUI for the ClientCam51
application.
         */
        public ClientCam51GUI(){
            super("ClientCam51");
            initComponents();
            setAppearance();
        }

        /**
         * Initiates all the components of this GUI and sets up their
         * appearance.
         */
        public void initComponents(){
            //add the observable objects to this GUI
            observable0 = new SurveillanceToolObservable(0);


            observable0.addObserver(this);


            //initiate the three diferent video panels
            videoPanel0 = new VideoPanel("Camera1", observable0);


            //initiate and set up the properties for the main panel of
this GUI
            mainPanel = new JPanel();
            mainPanel.setBackground(Color.gray);//background colors are
the same for the whole application
            mainPanel.setLayout(new BoxLayout(mainPanel,
BoxLayout.X_AXIS));//set the layout manager
            //add all components along the x-axis
            mainPanel.add(videoPanel0);


        }

        /**
         * Sets up the apperance and functionalities of this GUI and
adds
         * all components.
         */
        public void setAppearance(){
```

```
            // add closing functionality to this GUI
            addWindowListener(new WindowAdapter() {
                public void windowClosing( WindowEvent event ) {
                    System.exit(0);
                }
            });
            // set up the container of this GUI
            Container c = getContentPane();
            c.setBackground(Color.GRAY);//background colors are the
same for the whole application
            c.setLayout(new BorderLayout());//set the layout manager
            c.add(mainPanel, BorderLayout.CENTER);//add the
component(containing all other components)
        }

        /**
         * Updates this GUI according to the parameters
         *
         * @param o the object that sent the update.
         * @param arg the message containing the updated information.
         */
        public void update(java.util.Observable o, Object arg) {
            //check if it as an alarm that has triggered the observer
            if (o instanceof AlarmRun){
                //it is a alarm -> start saving the right stream
                int index = ((AlarmRun)o).getIndex();

                return;
            }
            //check if it as an saveStop that has triggered the observer
            if (o instanceof SaveRun){
                //it is a save stop -> grab the first frame and add to
the
                //scrollable area
            int index = ((SaveRun)o).getIndex();

                //we have a video, create a thumbnail of it.
                //this might look ugly, but I find it a good way to do
it.
                //I couldn't think of any better way that is.
                final JButton streamLabel; // the button representing
the video
                Processor iconProc = null; // the processor for getting
a frame
                try {
                    /* create the processor */
                    iconProc = Manager.createProcessor(new
MediaLocator(location[index] + fileName[index]));
                }
                catch (Exception e) {
                    System.err.println("Cannot create processor for
creating a thumbnail. \n" + e);
                }
                iconProc.addControllerListener(this);
                iconProc.configure();
                if ( !waitForState(iconProc.Configured, iconProc) ) {
```

```
                    System.err.println("Failed to configure the
processor");
                }
                TrackControl tc[] = iconProc.getTrackControls();
                if (tc == null) {//check if any track controls, if not
tell the user
                    System.err.println("Failed to obtain track controls
from the processor.");
                }
                TrackControl videoTrack = null;
                for (int j = 0; j < tc.length; j++) {
                    if (tc[j].getFormat() instanceof VideoFormat) {
                        videoTrack = tc[j];
                        break;
                    }
                }
                // if there is no video track control we tell the user
                if (videoTrack == null) {
                    System.err.println("The input media does not
contain a video track.");
                }
                // set up the codec chain for playback, get the
framegrabcodec in there
                FrameGrabCodec fgCodec = new FrameGrabCodec();
                try {
                    /* add the magic codec that just saves a frame of
the video. */
                    Codec[] codec = { fgCodec };
                    videoTrack.setCodecChain( codec );
                } catch(Exception e) {
                    System.out.println("failed to set up codec for the
processor...");
                }
                iconProc.prefetch();
                if ( !waitForState( iconProc.Prefetched, iconProc) ) {
                    System.err.println("Failed to realize the
processor.");
                }

                //starting playback of the processor, we have to do
this to get the first frame
                if (iconProc != null) {
                    /* ok, run it to get the frame into the codec */
                    iconProc.start();
                }
                /* use ImageIconCreater to add the image from the codec
to the button */
                streamLabel = new JButton(fileName[index],
ImageIconCreater.create(fgCodec.getFrame()));
                try {
                                    /* store the URL of the file as the
tooltip.
                                     * actually a workaround in the
beginning that turned out ok. */
                    streamLabel.setToolTipText(location[index] +
fileName[index]);
                } catch (Exception e)
```

```
                {}
                                    /* and now, an actionlistener, we need
this since after adding the button
                                     * we have no other way to do stuff
with it. */
                streamLabel.addActionListener(new ActionListener() {
                    public void actionPerformed(ActionEvent event) {
                        /* the action to perform when button is clicked
*/
                        /* create a Quick 'n' Dirty Video Player */
                        QDVideoPlayer qdVP = new
QDVideoPlayer(streamLabel.getToolTipText());
                        qdVP.show();
                    }
                });
                // add the label to the correct panel.

                switch (index) {
                    case 0:
                        videoPanel0.addScrolledComponent(streamLabel);
                        break;
                    case 1:
                        //videoPanel1.add(streamLabel);
                        videoPanel1.addScrolledComponent(streamLabel);
                        break;

                    default:
                        break;
                }
                display();//displays the graphical user interface
                return;
            }

            //when not alarm we know it is a SurveillanceToolObservable
that trigged.
            //retrieve the SurveillanceToolObservable object from the
parameter
            SurveillanceToolObservable obs =
(SurveillanceToolObservable) o;
            //get its index
            int index = obs.getIndex();
            //if not an alarm choose the proper actions depending on
the index
            switch(obs.getSource()){
                case SurveillanceToolObservable.NO_SOURCE://uncounted
for
                    break;
                case SurveillanceToolObservable.ADD_STREAM://add a
stream to the correct videoPanel
                    addStream(index, obs.getIp(), obs.getPort());
                    break;

                default:
                    break;
            }
        }
```

```
        /**
         * Adds the stream found at the given ipaddress and port to the
panel
         * with index n.
         *
         * @param index index of panel
         * @param ipAddress the ip addess of the source of the
videostream
         * @param portNr the port number for the videostream on the
source
         */
        private void addStream(int index, String ipAddress, String
portNr) {
            //our incoming port, where we listen
            int nPort = 0;
            try{
                //transform the portNr to an int
                nPort = (new Integer(portNr)).intValue();
            }catch(NumberFormatException e){
                System.err.println("The port number is not an integer:
" + portNr);
            }
            int nTtl = 1;
            //lots of stuff from JMStudio
            //RTPSessionMgr       mngrSession;
            String              nameUser = null;
            String              cname;
            SessionAddress      addrLocal;
            InetAddress         addrDest;
            SessionAddress      addrSession;
            SourceDescription   arrUserDescr [];
            mngrSession[index] = (RTPSessionMgr)
RTPManager.newInstance();
            mngrSession[index].addReceiveStreamListener(this);
            cname = mngrSession[index].generateCNAME();
            try {
                nameUser = System.getProperty("user.name");
            } catch (SecurityException e){
                nameUser = "jmf-user";
            }
            addrLocal = new SessionAddress();
            try {
                addrDest = InetAddress.getByName( ipAddress ); //our
IP, for use in the Session
                addrSession = new SessionAddress( addrDest, nPort,
addrDest, nPort + 1 );
                SessionAddress localAddr, destAddr;
                if( addrDest.isMulticastAddress()) {
                    // local and remote address pairs are identical:
                    localAddr= new SessionAddress( addrDest,
                    nPort,
                    nTtl);
                    destAddr = new SessionAddress( addrDest,
                    nPort,
```

```
                    nTtl);

                } else {
                    localAddr= new SessionAddress(
InetAddress.getLocalHost(),
                    nPort);
                    destAddr = new SessionAddress( addrDest,
                    nPort);
                }
                mngrSession[index].initialize( localAddr);
                mngrSession[index].addTarget( destAddr);
            }
            catch ( Exception e ) {
            }
        }

        /**
         * Do the corresponding update to the event occured.
         *
         * @param event the event triggering
         */
        public void update(ReceiveStreamEvent event) {
            int index = -1;
            //find the sourceRTPSM for this event
            RTPManager source = (RTPManager)event.getSource();
            //determine which videoPanel the stream belongs to
            for(int i=0; i < 3; i++){
                if(source.equals(mngrSession[i])){
                    index = i;
                }
            }
            //get that videoPanel
            VideoPanel panel;
            switch(index){
                case 0:
                    panel = videoPanel0;
                    break;
                case 1:
                    panel = videoPanel1;
                    break;

                default:
                    return;
            }
            // create a new player if a new recvstream is detected
            if (event instanceof NewReceiveStreamEvent) {
                //  String cname = "Java Media Player";
                ReceiveStream stream = null;
                try {
                    // get a handle over the ReceiveStream
                    stream =
((NewReceiveStreamEvent)event).getReceiveStream();
                    // get a handle over the ReceiveStream datasource
                    DataSource dsource = stream.getDataSource();
                    // create a player by passing datasource to the
Media Manager
                    p[index] = Manager.createProcessor(dsource);
```

```
                } catch (Exception e) {
                    System.err.println("NewReceiveStreamEvent exception
for " +
                    "VideoPanel: " + index + ". \n" + e.getMessage());
                    return;
                }
                if (p[index] == null){
                    System.err.println("The processor" + index + "for
the new stream is null");
                    return;
                }
                //add this as a controller listener
                p[index].addControllerListener( this );
                //configuer the processor to be able to program it
                p[index].configure();
                if ( !waitForState(p[index].Configured, p[index]) ) {
                    System.err.println("Failed to configure the
processor: " + index);
                    return;
                }
                // prefetches the players to be able to start
                p[index].prefetch();
                if ( !waitForState( p[index].Prefetched, p[index]) ) {
                    System.err.println("Failed to realize processor: "
+ index);
                    return;
                }
                //start the playback
                if (p[index] != null) {
                    p[index].start();
                }else{
                    System.err.println("Cannot start playback for
processor: " + index);
                }
                //create a Clone of stream through the output from p
                // and the Clone class
                origDataSource[index] = p[index].getDataOutput();
                cloneableDataSource[index] =
Manager.createCloneableDataSource(origDataSource[index]);
                if (cloneableDataSource[index] == null) {
                    System.err.println("Cannot clone DataSource for
processor: " + index);
                    System.exit(0);
                }
                //first clone
                clone[index] = new Clone();
                //opens the cloned datasource, this is the only place
                //cloneableDataSource[] can be used. If we want a new
clone we
                //must use
((SourceCloneable)cloneableDataSource)[].createClone())
                if (!clone[index].open(cloneableDataSource[index]))
                    System.exit(0);
                try {
                    //get the processor from our clone
                    p[index] = clone[index].getProcessor();
```

```
                }
                catch (Exception e) {
                    System.out.println("Cannot retrieve the processor " +
index
                    + ", from the clone " + index + ". \n" + e);
                }
                if (p[index] == null){
                    System.out.println("The processor " + index + " for
the new stream is null");
                    return;
                }
                //add this GUI as the listener
                p[index].addControllerListener( this );
                //configuer the processor to be able to program it
                p[index].configure();
                if ( !waitForState(p[index].Configured, p[index]) ) {
                    System.err.println("Failed to configure processor:
" + index);
                    return;
                }
                // obtain the track controls.
                TrackControl tc[] = p[index].getTrackControls();
                if (tc == null) {
                    System.err.println("Failed to obtain track controls
from " +
                    "processor: " + index);
                    System.exit(0);
                }
                // search for the video track control.
                TrackControl videoTrack = null;
                for (int i = 0; i < tc.length; i++) {
                    if (tc[i].getFormat() instanceof VideoFormat) {
                        videoTrack = tc[i];
                        break;
                    }
                }
                // if there is no video track control we exits
                if (videoTrack == null) {
                    System.err.println("The input media does not
contain a " +
                    "video track for VideoPanel: " + index);
                    System.exit(0);
                }
                // set up the codec chain for playback with possible
detection
                /*try {
                    mdCodec[index] = new MotDetCodec();
                    Codec[] codec = { mdCodec[index] };
                    videoTrack.setCodecChain( codec );
                } catch(Exception e) {
                    System.err.println("Failed to set up the codec for
" +
                    "processor: " + index + ". \n" + e);
                }*/
                //set the contentDescriptor = null to use the processor
as a player
```

```
                p[index].setContentDescriptor(null);
                //prefetch to be able to start the playback
                p[index].prefetch();
                if ( !waitForState( p[index].Prefetched, p[index]) ) {
                    System.err.println("Failed to realize processor: "
+ index);
                    return;
                }
                //start the playback of the player
                if (p[index] != null) {
                    p[index].start();
                }else{
                    System.err.println("The processor " + index + " is
null");
                }
                // adds the hopefully found video component to the gui
                Component vc;
                if ((vc = p[index].getVisualComponent()) != null) {
                    panel.addStreamedComponent( vc);
                } else{
                    System.err.println("Processor " + index + " has no
visual component");
                }
            }
            display();//displays the graphical user interface
        }

        /**
         * The update method for this ControllerListener, sets he
proper actions
         * when updated.
         *
         * @param event the occured event
         */
        public void controllerUpdate(javax.media.ControllerEvent event)
{
            Object obj = event.getSource();
            Player p = (Player) obj;
            //if the player is dead - we leave
            if (p == null)
                return;
            //respond to the event that trigged the listener
            if (event instanceof ConfigureCompleteEvent ||
            event instanceof RealizeCompleteEvent ||
            event instanceof PrefetchCompleteEvent) {
                synchronized ( waitSync ) {
                    stateTransitionOK = true;
                    waitSync.notifyAll();
                }
                //stop waiting
            } else if ( event instanceof ResourceUnavailableEvent ) {
                System.err.println("Resource unavailable for the
processor");
                synchronized ( waitSync ) {
                    stateTransitionOK = false;
                    waitSync.notifyAll();
                }
```

```
                // close the application
            } else if ( event instanceof EndOfMediaEvent ||
            event instanceof ControllerClosedEvent ||
            event instanceof ControllerErrorEvent ) {
                p.close();
            }
        }

        /**
         * Packs and displays this GUI.
         */
        public void display() {
            pack();
            show();
        }

        /**
         * Block until the processor has transitioned to the given
state.
         * Return false if the transition failed.
         *
         * Copied from TestMotionDetection.java, located on
         * http://darnok.com/programming/motion-detection/,
         * written by Konrad Rzeszutek.
         *
         * @param state the wanted state for the processor
         * @p the processor that is transitioning
         */
        boolean waitForState(int state, Processor p) {
            // waits for a state confirmation for the process
            synchronized (waitSync) {
                try {
                    while (p.getState() != state && stateTransitionOK)
                        waitSync.wait();
                } catch (Exception e) {
                    System.err.println("Exception in waitForState: \n"
+ e);
                }
            }
            return stateTransitionOK;
        }
    }
}
```