ระบบควบคุมอุปกรณ์ไฟฟ้าและสังเกตการณ์สำหรับห้องเรียน

ELECTRICAL DEVICES CONTROLLING AND MONITORING SYSTEM FOR CLASSROOM

โดย

นางสาวชัชนี เนียมสวรรค์

นางสาวชัญลักษณ์ ดิษฐ์สูงเนิน

นายนัฏฐภัทร์ ร่มฟ้าไทย

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2549

ผ่านการตรวจสอบแล้ว

(ลงชื่อ).....ผู้ตรวจ

ผ่านการตรวจรูปเล่มแล้ว

(ลงชื่อ).....ผู้ตรวจ

ระบบควบคุมอุปกรณ์ไฟฟ้าและสังเกตการณ์สำหรับห้องเรียน

ELECTRICAL DEVICES CONTROLLING AND MONITORING SYSTEM FOR CLASSROOM

โดย

นางสาวชัชนี เนียมสวรรค์ 46010155

นางสาวธัญลักษณ์ ดิษฐ์สูงเนิน 46010304

นายนัฏฐภัทร์ ร่มฟ้าไทย 46010349

อาจารย์ที่ปรึกษา

รศ.ดร. ปราโมทย์ วาดเขียน

ผศ.ดร. จีรสุดา โกษียาภรณ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2549

ผ่านการตรวจแล้ว

(ลงชื่อ).....ผู้ตรวจ

ปริญญานิพนธ์ปีการศึกษา 2549

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง **ระบบควบคุมอุปกรณ์ไฟฟ้าและสังเกตการณ์สำหรับห้องเรียน**

ELECTRICAL DEVICES CONTROLLING AND MONITORING SYSTEM FOR CLASSROOM

ผู้จัดทำ

1. นางสาวชัชนี เนียมสวรรค์ 46010155

2. นางสาวชัญลักษณ์ ดิษฐ์สูงเนิน 46010304

3. นายนัฏฐภัทร์ ร่มฟ้าไทย 46010349

.....อาจารย์ที่ปรึกษา

(รศ.ดร.ปราโมทย์ วาดเขียน)

.....อาจารย์ที่ปรึกษา

(ผศ.ดร.จีรสุดา โกษียาภรณ์)

ระบบควบคุมอุปกรณ์ไฟฟ้าและสังเกตการณ์สำหรับห้องเรียน

ELECTRICAL DEVICES CONTROLLING AND MONITORING SYSTEM FOR CLASSROOM

โดย นางสาวชัชนี เนียมสวรรค์ 46010155

นางสาวธัญลักษณ์ ดิษฐ์สูงเนิน 46010304

นายนัฏฐภัทร์ ร่มฟ้าไทย 46010349

อาจารย์ที่ปรึกษา รศ.ดร.ปราโมทย์ วาดเขียน

ผศ.ดร.จีรสุดา โกษียาภรณ์

บทคัดย่อ

ปริญญานิพนธ์นี้เป็นการสร้างอุปกรณ์ ควบคุมการใช้พลังงานภายในห้องเรียน(เครื่องปรับอากาศและไฟฟ้าแสงสว่าง) โดยใช้คอมพิวเตอร์ในเครือข่ายแลนเดียวกันกับคอมพิวเตอร์ภายในห้องเรียน ในการสั่งงานผ่านคอมพิวเตอร์ที่ติดตั้งภายในห้องเรียนซึ่งจะประกอบไปด้วยวงจรที่ใช้ในการควบคุมเครื่องปรับอากาศและหลอดไฟซึ่งต่ออยู่กับเครื่องคอมพิวเตอร์ภายในห้องเรียน คอมพิวเตอร์ในห้องเรียนจะส่งคำสั่งควบคุมการเปิด-ปิดเครื่องปรับอากาศและหลอดไฟนอกจากนี้ยังสามารถตรวจสอบได้ว่าห้องเรียนดังกล่าวมีการใช้งานอยู่หรือไม่ โดยสามารถเรียกดูภาพจากกล้องที่ติดตั้งที่คอมพิวเตอร์ภายในห้องเรียนผ่านทางคอมพิวเตอร์ในเครือข่ายแลนเดียวกัน

Abstract

The power controlled system in classroom (air-condition and light electric) is presented in this project by using sever computer to control client computer in each room. The proposed system consists of a server computer, client computers connected with a controlled power circuit. The server computer transmits the control signal to client computer to turn on/off air-condition and light bulbs. Furthermore, the status of each room can be examined by viewing the image received from the installed digital camera in each room.

กิตติกรรมประกาศ

โครงงานและรายงานฉบับนี้ได้รับความอนุเคราะห์และความเมตตาจากอาจารย์ที่ปรึกษา คือ ท่านรศ.ดร.ปราโมทย์ วาดเขียน และ ผศ.ดร.จีรสุดา โกษียาภรณ์ ที่คอยให้คำปรึกษา กวดขัน เอาใจใส่ และให้คำชี้แนะ อีกทั้งพี่ปริญญาโทและปริญญาเอกทุกท่านที่ให้คำปรึกษาและเป็นกำลังใจมาโดยตลอด

สารบัญ

สารบัญ (ต่อ)

สารบัญรูปภาพ

สารบัญรูปภาพ (ต่อ)

สารบัญรูปภาพ (ต่อ)

สารบัญตาราง

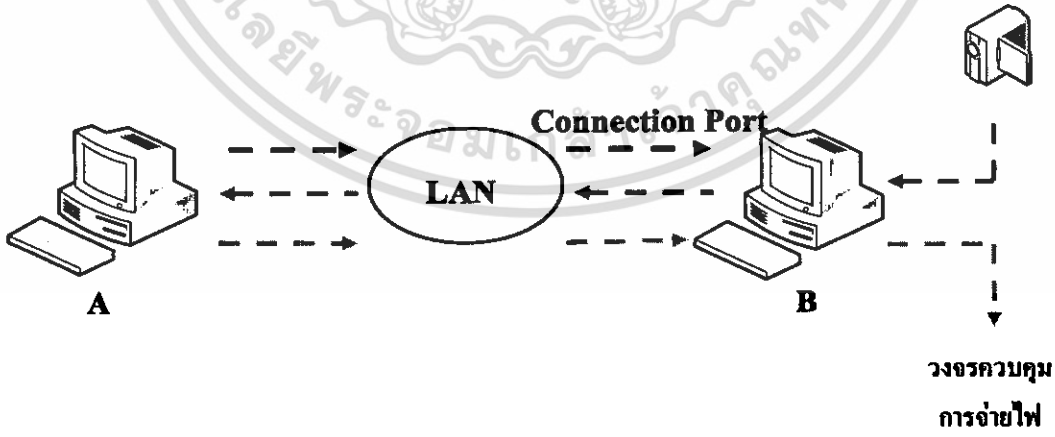
บทที่ 1
บทนำ

เนื่องจากปัจจุบันนี้ได้มีการรณรงค์ให้มีการประหยัดพลังงานภายในองค์กรต่าง ๆ อย่างแพร่หลาย แต่ยังมีบุคลากรในหลายหน่วยงานที่ยังใช้พลังงานไฟฟ้าอย่างสิ้นเปลืองโดยเฉพาะการเปิดไฟฟ้าแสงสว่างทิ้งไว้ในบริเวณห้องโดยที่มิได้มีผู้ใดอยู่ภายในห้อง ดังนั้นเพื่อเป็นการสนับสนุนโครงการของรัฐบาลที่มีวัตถุประสงค์ให้มีการประหยัดพลังงาน เพื่อเป็นการช่วยลดพลังงานสิ้นเปลืองไปโดยเปล่าประโยชน์ ด้วยเหตุนี้จึงได้จัดทำโครงงานระบบควบคุมอุปกรณ์ไฟฟ้าและสังเกตการณ์สำหรับห้องเรียนนี้ขึ้น ซึ่งเป็นการควบคุมอุปกรณ์ไฟฟ้าภายในห้องเรียน โดยมีกล้องแสดงภาพเคลื่อนไหวสามารถตรวจสอบได้ว่ามีผู้ใช้งานห้องเรียนนั้นอยู่หรือไม่ และสามารถควบคุมการจ่ายไฟให้กับอุปกรณ์ไฟฟ้าภายในห้องเรียนนั้นจากเครื่องคอมพิวเตอร์เครื่องใดก็ได้ที่อยู่ภายในเครือข่ายเดียวกัน ซึ่งโครงงานนี้สามารถช่วยประหยัดพลังงานได้อีกทางหนึ่ง

1.1 วัตถุประสงค์ของโครงงาน

1. เพื่อสร้างระบบประหยัดพลังงาน
2. เพื่อศึกษาและออกแบบโปรแกรมรับ-ส่งข้อมูลภาพระหว่างเครื่องคอมพิวเตอร์
3. สามารถสร้างอุปกรณ์ควบคุมการจ่ายไฟ
4. เพื่อศึกษาและออกแบบโปรแกรมสำหรับติดต่อและควบคุมฮาร์ดแวร์

1.2 โครงสร้างของโครงงาน



รูปที่ 1.1 แสดงบล็อกไดอะแกรมระบบควบคุมการใช้พลังงานภายในห้องเรียน

1.3 หน้าที่ของเครื่องคอมพิวเตอร์ภายในห้องเรียนและเครื่องคอมพิวเตอร์ภายในเครือข่ายแลนเดียวกัน

ด้านเครื่องคอมพิวเตอร์ภายในห้องเรียน

1. รอการเชื่อมต่อเพื่อเปิดพอร์ตอนุกรมที่ต่อกับอุปกรณ์
2. ส่งภาพไปยังเครื่องคอมพิวเตอร์ภายในเครือข่ายแลนเดียวกันที่เรียกดู
3. สั่งงานควบคุมการจ่ายไฟที่วงจรควบคุมการจ่ายไฟ
4. แจ้งสถานะของอุปกรณ์ไฟฟ้าภายในห้องเรียนกลับไปยังเครื่องคอมพิวเตอร์ที่เรียกดู

ด้านเครื่องคอมพิวเตอร์ภายในเครือข่ายแลนเดียวกัน

1. เชื่อมต่อกับเครื่องคอมพิวเตอร์ภายในห้องเรียน
2. รับภาพจากกล้องที่เครื่องคอมพิวเตอร์ภายในห้องเรียนเพื่อสังเกตการณ์
3. ส่งคำสั่งควบคุมการจ่ายไฟไปยังเครื่องคอมพิวเตอร์ภายในห้องเรียน

บทที่ 2
ทฤษฎีและหลักการ

2.1 สถาปัตยกรรมโปรโตคอลทีซีพี/ไอพี (TCP/IP)

โปรโตคอล (Protocol)

คือกฎ ระเบียบ หรือข้อกำหนดต่าง ๆ รวมถึงมาตรฐานที่ใช้ เพื่อให้ตัวรับและตัวส่งสามารถดำเนินการติดต่อสื่อสารได้สำเร็จ

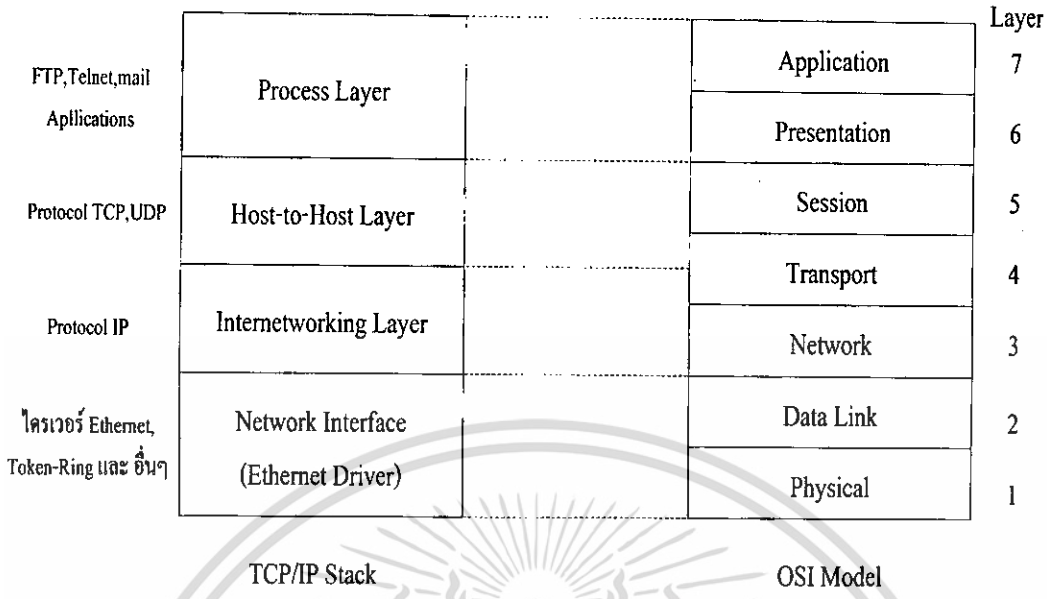
ทีซีพี/ไอพี (TCP: Transmission Control Protocol/Internet Protocol) เป็นชุดของโปรโตคอลที่ถูกใช้ในการสื่อสารผ่านเครือข่ายอินเตอร์เน็ต ออกแบบมาสำหรับการเชื่อมโยงระหว่างเน็ตเวิร์คขนาดใหญ่เชื่อมต่อกับเครือข่ายแวน (WAN: Wide Area Network) ทีซีพี/ไอพีได้รับการพัฒนาขึ้นในปีค.ศ.1969 โดยหน่วยงานของรัฐบาลอเมริกามีชื่อว่า Defense Advanced Research Projects Agency (DARPA) จุดประสงค์ของทีซีพี/ไอพีคือเพื่อเชื่อมโยงเน็ตเวิร์คการติดต่อสื่อสารที่ความเร็วสูง

ปกติแล้วเราทราบว่าตามองค์กรมาตรฐานสากลจะแบ่งโมเดลโปรโตคอลการสื่อสารเป็น 7 เลเยอร์ แต่โปรโตคอลทีซีพี/ไอพีสามารถแม็พเป็นโมเดลในแนวความคิดไว้ 4 เลเยอร์ ซึ่งรู้กันว่าโมเดล DARPA NET เลเยอร์ทั้งสี่ของโมเดล DARPA NET จะเป็น ดังรูปที่ 2.1

โปรโตคอลทีซีพี/ไอพีมีการจัดการแบ่งกลไกการทำงานออกเป็นชั้น ๆ หรือเลเยอร์เหมือนกับมาตรฐานโอเอสไอโมเดล (OSI Model) และสามารถเทียบเคียงกับมาตรฐานของโอเอสไอโมเดลได้ ซึ่งในแต่ละเลเยอร์ของโปรโตคอลทีซีพี/ไอพีจะประกอบด้วย

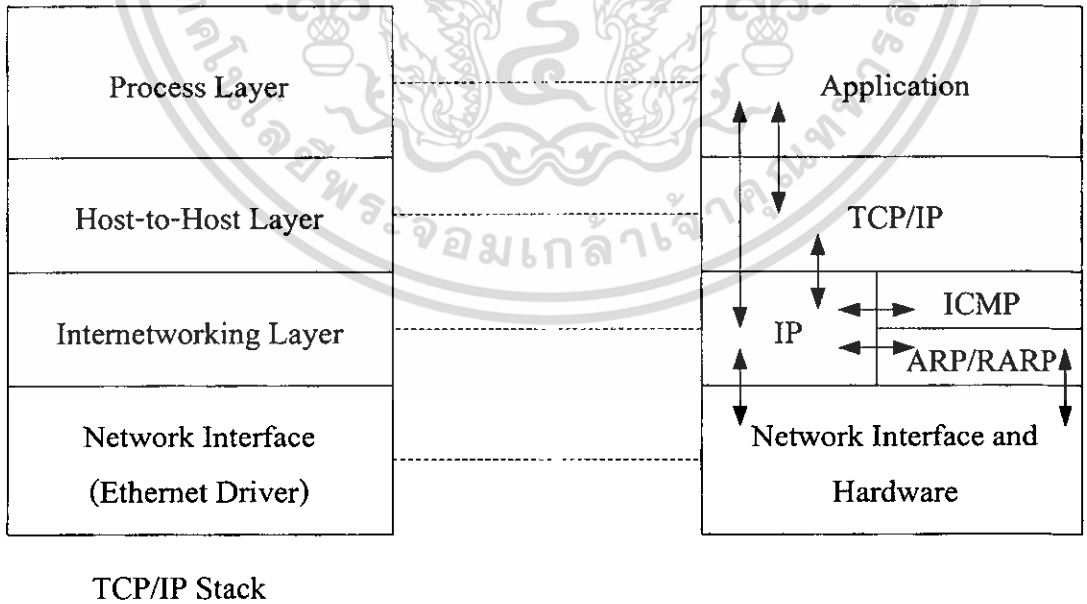
- โปรเซสเลเยอร์หรือแอพพลิเคชั่นเลเยอร์
- โฮสต์ทูโฮสต์เลเยอร์หรือทรานสปอร์ตเลเยอร์
- อินเตอร์เน็ตเวิร์คเลเยอร์
- เน็ตเวิร์คอินเตอร์เฟสเลเยอร์

โดยเทียบกับมาตรฐานโอเอสไอโมเดลแล้วจะได้ดังรูปที่ 2.1 ซึ่งจะเห็นว่าบางเลเยอร์ของโปรโตคอลทีซีพี/ไอพีเทียบได้กับมาตรฐานโอเอสไอโมเดลถึงสองเลเยอร์และบางเลเยอร์ก็จะทำงานคาบเกี่ยวกับหลาย ๆ เลเยอร์ของโอเอสไอโมเดล



รูปที่ 2.1 แสดงทีซีพี/ไอพีสแตก (TCP/IP Stack) เปรียบเทียบกับมาตรฐานโอเอสไอ

ในแต่ละกลไกของโปรโตคอลทีซีพี/ไอพีจะมีโปรโตคอลอื่น ๆ ในชุดของทีซีพี/ไอพีร่วมทำงานอยู่ด้วย จึงทำให้เป็นที่มาของชื่อเรียกโปรโตคอลสแตก (Protocol Stack) เนื่องจากมีโปรโตคอลซ้อนทับกันอยู่ เพื่อช่วยกันทำงานดังรูปที่ 2.2

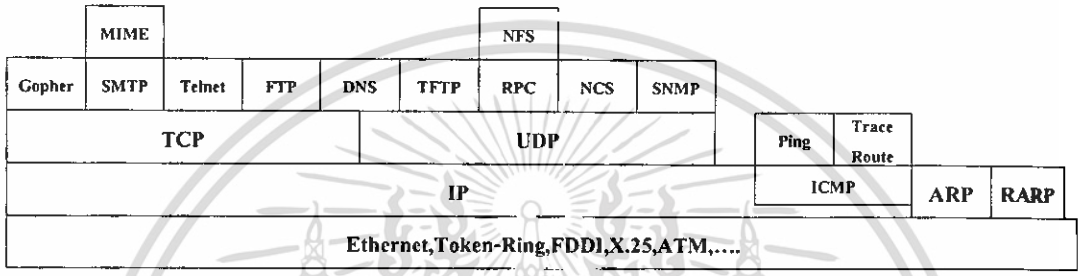


รูปที่ 2.2 แสดงความสัมพันธ์ระหว่างโปรโตคอลต่าง ๆ ของทีซีพี/ไอพี

จากรูปที่ 2.2 จะเห็นได้ว่ามีโปรโตคอลในแต่ละระดับซ้อนทับกันอยู่หลายตัวด้วยกัน ซึ่งเราจะดูในรายละเอียดต่อไป การซ้อนทับกันเป็นชั้น ๆ หรือแต่ละเลเยอร์นี้หากเป็นโอเอสไอโมเดลข้อบังคับให้แต่ละชั้นติดต่อได้กับเฉพาะชั้นที่ติดกับตนเองเท่านั้นแต่สำหรับทีซีพี/ไอพีสแตกแล้วจะเห็นว่าบางชั้นสามารถละเลยหรือข้ามไปติดต่อกับชั้นที่ไม่ติดกับตนได้

2.1.1 การแบ่งชั้นของทีซีพี/ไอพี

โปรเซสเลเยอร์ (Process Layer) หรือแอพพลิเคชั่นเลเยอร์ (Application Layer)



รูปที่ 2.3 แสดงแอพพลิเคชั่นต่าง ๆ ในทีซีพี/ไอพีสแตก

จากรูปที่ 2.3 แสดงลำดับชั้นการทำงานของโปรโตคอลทีซีพี/ไอพีเทียบกับมาตรฐานโอเอสไอโมเดลนั้น ในชั้นบนสุดเรียกว่าโปรเซสเลเยอร์ทำงาน 2 หน้าที่เทียบได้กับแอพพลิเคชั่นเลเยอร์และพรีเซนเตชั่นเลเยอร์ (Presentation Layer) ในชั้นนี้จะรองรับการทำงานของแอพพลิเคชั่นต่าง ๆ ที่ทำงานเป็นโปรเซสอยู่ในเครื่องเซิร์ฟเวอร์ที่ให้บริการและเครื่องที่ขอใช้บริการหรือไคลแอนท์ ซึ่งจะติดต่อผ่านโปรโตคอลเฉพาะแอพพลิเคชั่นอีกทีหนึ่ง การทำงานของแอพพลิเคชั่นต่าง ๆ จะอยู่ที่โปรเซสเลเยอร์นี้และมีการติดต่อกันตามแต่ละโปรโตคอลเฉพาะแล้วแต่แอพพลิเคชั่นที่ใช้งานจากการที่โปรเซสเลเยอร์ของทีซีพี/ไอพีรองรับให้โปรโตคอลอื่นทำงานได้หลายโปรเซสและหลายโปรโตคอลได้พร้อมกันนั้น ทำให้ผู้ใช้สามารถเปิดโปรแกรมใช้งานได้หลาย ๆ อย่างพร้อมกัน

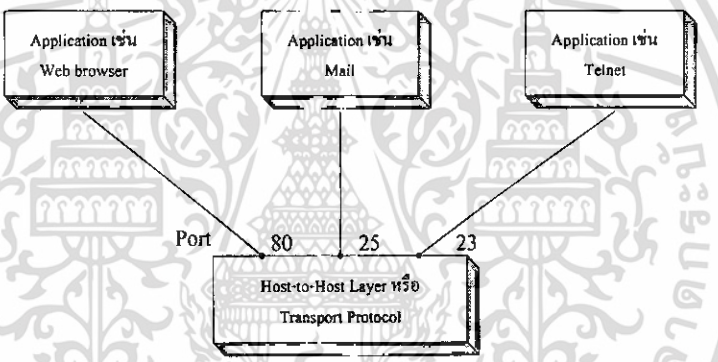
โปรโตคอลหลัก ๆ ที่ทำงานในโปรเซสเลเยอร์ซึ่งผู้ใช้มักจะคุ้นเคยกันดีได้แก่ เอฟทีพี (FTP : File Transfer Protocol) เทลเน็ต (Telnet) เอชทีทีพี (HTTP : Hyper Text Transfer Protocol) และ เอสเอ็มทีพี (SMTP : Simple Mail Transfer Protocol) นอกจากนี้ยังมีโปรโตคอลอื่นที่อยู่เบื้องหลัง ซึ่งทำงานโดยที่ผู้ใช้ไม่สามารถมองเห็นได้จากโปรแกรมหรือไม่ได้มีการใช้งานโดยตรง เช่น

- โปรโตคอลดีเอ็นเอส (DNS : Domain Name System) ที่ทำหน้าที่แปลงข้อมูลชื่อโดเมนเนม หรือ ชื่อเว็บไซท์ทั้งหลายให้เป็นหมายเลขไอพีแอดเดรส
- โปรโตคอลเอสเอ็นเอ็มพี (SNMP : Simple Network Management Protocol) ใช้ในการควบคุมและตรวจสอบอุปกรณ์ที่อยู่ในเครือข่าย

- โปรโตคอลดีเอชซีพี (DHCP : Dynamic Host Configuration Protocol) ทำหน้าที่แจกจ่ายข้อมูลพารามิเตอร์ของเครือข่ายให้กับเครื่องไคลเอนท์ที่เชื่อมต่ออยู่

โฮสต์ทูโฮสต์เลเยอร์ (Host-to-Host Layer) หรือ ทรานสปอร์ตเลเยอร์ (Transport Layer)

การทำงานที่ชั้นของโฮสต์ทูโฮสต์เลเยอร์จะมีบทบาทในการจัดการการต่อจากโปรเซสเลเยอร์ บางครั้งเรามักเรียกชั้นโฮสต์ทูโฮสต์เลเยอร์ว่าเป็นทรานสปอร์ตเลเยอร์ซึ่งไม่ใช่ชั้นของทรานสปอร์ตเลเยอร์ในมาตรฐานโอเอสไอโมเดลการทำงานของโฮสต์ทูโฮสต์เลเยอร์นี้จะมีการสร้างคอนเน็กชั่น (Connection) หรือการเชื่อมต่อกันระหว่างแอพพลิเคชั่นกับโฮสต์ทูโฮสต์เลเยอร์โดยจุดที่เชื่อมต่อกันเพื่อรับส่งข้อมูลนี้เรียกว่าพอร์ตหรือซ็อกเก็ต และในแต่ละแอพพลิเคชั่นก็จะสร้างการเชื่อมต่อผ่านพอร์ตได้พร้อมกันหลายแอพพลิเคชั่นซึ่งการใช้งานพอร์ตของแต่ละแอพพลิเคชั่นที่อยู่ในชั้น โปรเซสเลเยอร์จะแตกต่างกันตามหมายเลขที่กำหนดไว้ และแต่ละโปรโตคอลจะมีการใช้งานพอร์ตหมายเลขต่าง ๆ ไม่ซ้ำกันตามรูปที่ 2.4

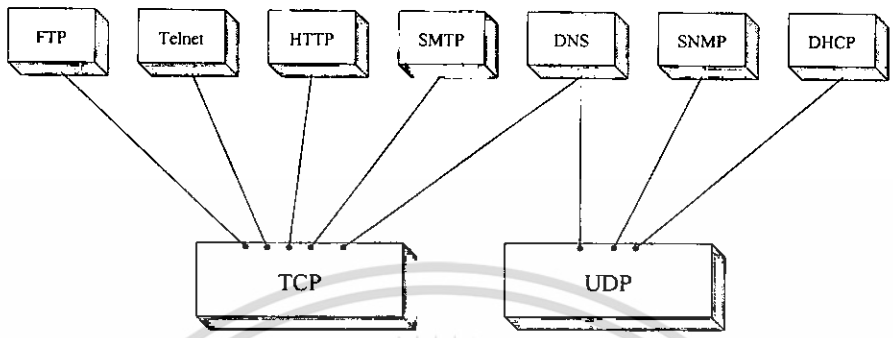


รูปที่ 2.4 แอพพลิเคชั่นหรือโปรเซสต่าง ๆ สื่อสารกับโฮสต์ทูโฮสต์เลเยอร์ ผ่านจุดเชื่อมต่อหรือพอร์ต

เมื่อแอพพลิเคชั่นทำงานผ่านโปรโตคอลในชั้นโปรเซสเลเยอร์ จะมีการส่งผ่านข้อมูลไปยังโฮสต์ทูโฮสต์เลเยอร์ที่ชั้นนี้จะมีการเชื่อมต่อผ่านพอร์ตที่กำหนดทำให้การรับส่งข้อมูลในแต่ละโปรโตคอลทำงานได้ถูกต้องถึงแม้ว่าในเครื่องเซิร์ฟเวอร์ที่ให้บริการจะมีการทำงานอยู่หลายโปรเซสที่แตกต่างกันก็ตามหรือมีผู้ใช้บริการเข้ามาใช้งานพร้อมกันจำนวนมากและหมายเลขแอพพลิเคชั่นในเวลาเดียวกันในชั้นโฮสต์ทูโฮสต์เลเยอร์ หรือทรานสปอร์ตเลเยอร์ของทีซีพี/ไอพีนี้จะมีโปรโตคอลทำงานอยู่ 2 โปรโตคอลที่แตกต่างกัน คือ โปรโตคอลทีซีพี (Transmission Control Protocol) และโปรโตคอลยูดีพี (UDP: User Datagram Protocol) ในการส่งข้อมูลผ่านลงไปในชั้นถัด ๆ ไปเราจะเห็นว่าโปรโตคอลทีซีพี และยูดีพีจะถูกผนึกเข้าไปในโปรโตคอลไอพีอีกทีหนึ่งและส่งต่อไปยังเครือข่ายอินเตอร์เน็ตต่อไป

ตัวโปรโตคอลทีซีพีและยูดีพีจะมีแอพพลิเคชั่นเฉพาะเพื่อเรียกใช้งานแยกกันคือ แอพพลิเคชั่นที่ใช้โปรโตคอลเอฟทีพี เทลเน็ต เอชทีทีพีและเอสเอ็มทีพี จะมีการส่งผ่านข้อมูลโดยเรียกใช้โปรโตคอลทีซีพี ส่วนแอพพลิเคชั่นที่ใช้โปรโตคอลเอสเอ็นเอ็มพีและดีเอชซีพีจะส่งผ่านข้อมูลโดย

เรียกใช้โปรโตคอลยูดีพีและสำหรับโปรโตคอลดีเอ็นเอสนั้นจะสามารถเรียกใช้งานได้ทั้งทีซีพีและยูดีพี ดังรูปที่ 2.5 ซึ่งเหตุผลที่มีการเรียกใช้โปรโตคอลทีซีพีและยูดีพีแตกต่างกัน ก็เนื่องจากวิธีการของทั้งสองโปรโตคอลต่างกันนั่นเอง



รูปที่ 2.5 โปรเซสต่าง ๆ ที่เรียกใช้ทรานสปอร์ตเลเยอร์ซึ่งในแต่ละโปรเซสจะเรียกใช้งานพอร์ตเฉพาะแตกต่างกัน ยกเว้นดีเอ็นเอสที่สามารถใช้งานได้ทั้งทีซีพีและยูดีพี

โปรโตคอลทีซีพี

โปรโตคอลทีซีพีเป็นโปรโตคอลที่มีการรับส่งข้อมูลแบบสตรีมโอเรนท์เต็ดโปรโตคอล (stream oriented protocol) หมายความว่า การรับส่งข้อมูลจะไม่คำนึงถึงปริมาณข้อมูลที่จะส่งไป แต่จะแบ่งข้อมูลเป็นส่วนย่อย ๆ ก่อน แล้วจึงจะส่งไปยังปลายทางอย่างต่อเนื่องเป็นลำดับข้อมูลในกรณีที่ข้อมูลส่วนหนึ่งส่วนใดสูญหายไป ก็จะส่งข้อมูลส่วนนั้นใหม่อีกครั้ง สำหรับปลายทางก็จะทำหน้าที่จัดเรียงส่วนของข้อมูลดาต้าแกรม (Datagram) ใหม่ให้ต่อเนื่องกันและประกอบกลับเป็นข้อมูลทั้งหมดได้ซึ่งจะแยกข้อมูลที่ไม่ถูกต้องออก ดังนั้นแอพพลิเคชั่นหรือโปรเซสใดที่อาศัยการส่งผ่านข้อมูลด้วยโปรโตคอลทีซีพีจะต้องใช้หน่วยความจำและขนาดของช่องสัญญาณ (Bandwidth) มากกว่ายูดีพี

การติดต่อระหว่างกันจะต้องเป็นแบบคอนเน็กชั่นโอเรนท์เต็ด (Connection-Oriented) คือต้องมีการสร้างการติดต่อกันเป็นเซสชั่น (Session) ทั้ง 2 ด้านเสียก่อน แล้วจึงจะรับส่งข้อมูลไปได้พร้อมกัน (full duplex) ในระหว่างการรับส่งข้อมูลนี้โปรโตคอลทีซีพีจะเพิ่มขบวนการสอบทานข้อมูลเพื่อให้ข้อมูลมีความถูกต้องไม่ผิดพลาดไปจากเดิม โดยการส่งสัญญาณสอบทานข้อมูล (Acknowledgement) และส่งข้อมูลให้ใหม่อีกครั้ง ถ้าปลายทางไม่ได้รับหรือเกิดความผิดพลาดขึ้นความน่าเชื่อถือของการส่งผ่านข้อมูลโดยโปรโตคอลทีซีพีมีมากกว่า แต่ก็อาศัยทรัพยากรของระบบมากกว่าในการทำงานเช่นกัน

โปรโตคอลยูดีพี

ในโฮสต์ทูโฮสต์เลเยอร์นอกจากจะมีโปรโตคอลทีซีพีทำงานแล้วก็ยังมีโปรโตคอลยูดีพีที่มีคุณสมบัติแตกต่างกันอยู่ด้วย ในการรับส่งข้อมูลผ่านโปรโตคอลยูดีพีจะเป็นแบบที่ทั้งสองด้านไม่จำเป็นต้องอาศัยการสร้างช่องทางเชื่อมต่อกัน (Connectionless) ระหว่างเครื่องเซิร์ฟเวอร์ให้บริการกับเครื่องที่ขอใช้บริการ โดยไม่ต้องแจ้งให้ฝ่ายรับข้อมูลเตรียมรับข้อมูลเหมือนโปรโตคอลทีซีพีและไม่มีการตรวจสอบความถูกต้องครบถ้วนในการรับส่งข้อมูลนั้น ๆ ด้วย เนื่องจากโปรโตคอลยูดีพีไม่มีสัญญาณ

สอบทานข้อมูล (Acknowledgement) ในการรับส่งข้อมูลแต่ละครั้ง และไม่มีการส่งข้อมูลใหม่อีกในกรณีที่เกิดความผิดพลาดของการส่งข้อมูล เมื่อเป็นเช่นนี้แอพพลิเคชั่นหรือโปรเซสใดที่ต้องอาศัยโปรโตคอลยูดีพีในการส่งผ่านข้อมูลก็อาจจะต้องสร้างขบวนการตรวจสอบข้อมูลขึ้นมาเอง

ตัวอย่างขั้นตอนกลไกการทำงานโดยใช้โปรโตคอลยูดีพีมีดังต่อไปนี้

1.ในชั้นของโปรเซสเลเยอร์เมื่อโปรแกรมควบคุมอุปกรณ์เครือข่าย เช่น โปรแกรมเน็ตเวิร์คแมนเนจเมนท์ (Network Management) ต้องการส่งข้อมูลไปยังอุปกรณ์ที่ต้องการแอพพลิเคชั่นนั้นจะติดต่อผ่านโปรโตคอลเอสเอ็นเอ็มพีในชั้นโปรเซสเลเยอร์

2.โปรโตคอลเอสเอ็นเอ็มพีจะติดต่อกับโปรโตคอลยูดีพีในชั้นถัดไปเพื่อขอติดต่อผ่านพอร์ตที่กำหนด

3.โปรโตคอลเอสเอ็นเอ็มพีเตรียมข้อมูลที่จะส่ง รวมทั้งที่อยู่ปลายทาง

4.โปรโตคอลเอสเอ็นเอ็มพีส่งผ่านข้อมูลให้โปรโตคอลยูดีพีที่อยู่ในชั้นโฮสต์ทูโฮสต์เลเยอร์

5.โปรโตคอลยูดีพีทำหน้าที่ผนึกข้อมูลหรือดาต้าแกรมนั้นไปกับโปรโตคอลไอพีในชั้นถัดลงไปเพื่อส่งข้อมูลออกจากเครื่อง

ซึ่งจะเห็นว่ามีกลไกที่ต่างจากการส่งข้อมูลด้วยโปรโตคอลทีซีพีซึ่งจะต้องมีการติดต่อกันก่อนและทั้งสองฝ่ายรับทราบการรับส่งข้อมูลของช่องการส่งข้อมูลนั้น

อินเตอร์เน็ตเวิร์คเลเยอร์ (Internetwork Layer)

ในระดับล่างต่อมาในชั้นอินเตอร์เน็ตเวิร์คเลเยอร์มีหน้าที่รับผิดชอบการระบุที่อยู่ การแพคเกตและฟังก์ชั่นต่าง ๆ ส่งผ่านข้อมูลในระหว่างเครือข่าย โดยมีโปรโตคอลที่ทำงานเป็นกลไกสำคัญในการส่งผ่านข้อมูลไปยังเครือข่ายใด ๆ บนอินเตอร์เน็ตคือโปรโตคอลไอพี (IP : Internet Protocol) นอกจากนี้ในชั้นอินเตอร์เน็ตเวิร์คเลเยอร์ยังมีโปรโตคอลทำงานอยู่ด้วยอีก 2 ชนิด คือ โปรโตคอลไอซีเอ็มพี (ICMP : Internet Control Message Protocol) และโปรโตคอลเออาร์พี (ARP : Address Resolution Protocol)

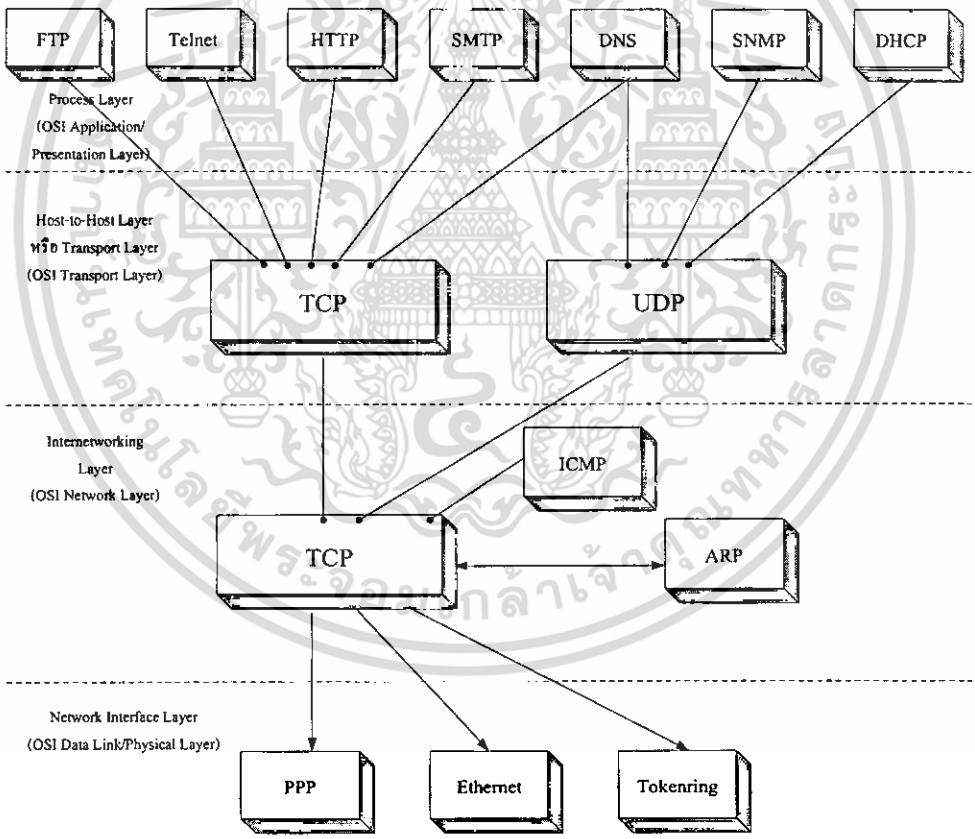
โปรโตคอลแกนหลักของอินเตอร์เน็ตเวิร์คเลเยอร์

- ไอพี : รับผิดชอบเกี่ยวกับที่อยู่ไอพี
- ไอซีเอ็มพี : มีหน้าที่ส่งข่าวสารและแจ้งข้อผิดพลาดให้แก่ไอพี
- เออาร์พี : อยู่ในลิงค์เลเยอร์ (Link Layer) ทำหน้าที่เปลี่ยนระหว่างไอพีแอดเดรส ให้เป็นแอดเดรสของเน็ตเวิร์คอินเตอร์เฟสเรียกว่าแมคแอดเดรส (MAC Address) ในการติดต่อระหว่างกันแมคแอดเดรสคือหมายเลขประจำของฮาร์ดแวร์อินเตอร์เฟส (Hardware Interface) ซึ่งในโลกนี้จะไม่มีแมคแอดเดรสที่ซ้ำกัน มีลักษณะเป็นเลขฐาน 16 ยาว 6 ไบต์ เช่น 23:43:45:AF:3D:78 โดย 3 ไบต์แรกจะเป็นรหัสของผู้ผลิต และ 3 ไบต์หลังจะเป็นรหัสของผลิตภัณฑ์
- ไอจีเอ็มพี (IGMP : Internet Group Management Protocol) รับผิดชอบการจัดการของกรุ๊ปมัลติคาสท์ไอพี

เน็ตเวิร์คอินเตอร์เฟสเลเยอร์ (Network Interface Layer)

เนื่องจากในด้านกายภาพของเครือข่ายนั้น มีหลายวิธีการและหลายรูปแบบในการเชื่อมต่อระบบให้เป็นเครือข่ายแต่อย่างไรก็ตามเครือข่ายอินเตอร์เน็ตนี้ ข้อมูลหรือไอพีดาต้าแกรม (IP Datagram) จะถูกถ่ายทอดและส่งผ่านไปยังปลายทางโดยไม่คำนึงถึงรูปแบบการเชื่อมต่อทางกายภาพ ไม่ว่าจะเป็นการใช้เครือข่ายใยแก้วนำแสงหรือเครือข่ายสายอันชิลด์ทวิสแพร์ (Unshielded Twist Pair: UTP) เชื่อมต่อเป็นแบบเครือข่ายอีเธอร์เน็ต (Ethernet) ธรรมดาหรือเครือข่ายโทเค็นริงเอทีเอ็ม (Token Ring ATM) ไอเอสดีเอ็น (ISDN) ก็ตาม

การทำงานในระดับล่างสุดต่อจากอินเตอร์เน็ตเวิร์คเลเยอร์จะเป็นการแปลงข้อมูลไอพีดาต้าแกรมให้อยู่ในรูปที่เหมาะสมและแปลงให้เป็นสัญญาณไฟฟ้าส่งไปยังเครือข่ายต่อไป ซึ่งในชั้นเน็ตเวิร์คอินเตอร์เฟสเลเยอร์นี้เมื่อเทียบกับมาตรฐานโอเอสไอโมเดลแล้วจะเป็นการรวม 2 เลเยอร์เข้าด้วยกันคือ ดาต้าลิงค์เลเยอร์ (Data Link Layer) และฟิสิกคอลเลเยอร์ (Physical Layer) กล่าวโดยสรุปก็คือการทำงานในชั้นต่าง ๆ ตามโครงสร้างของโปรโตคอลทีซีพี/ไอพีจะมีลักษณะดังรูปที่ 2.6

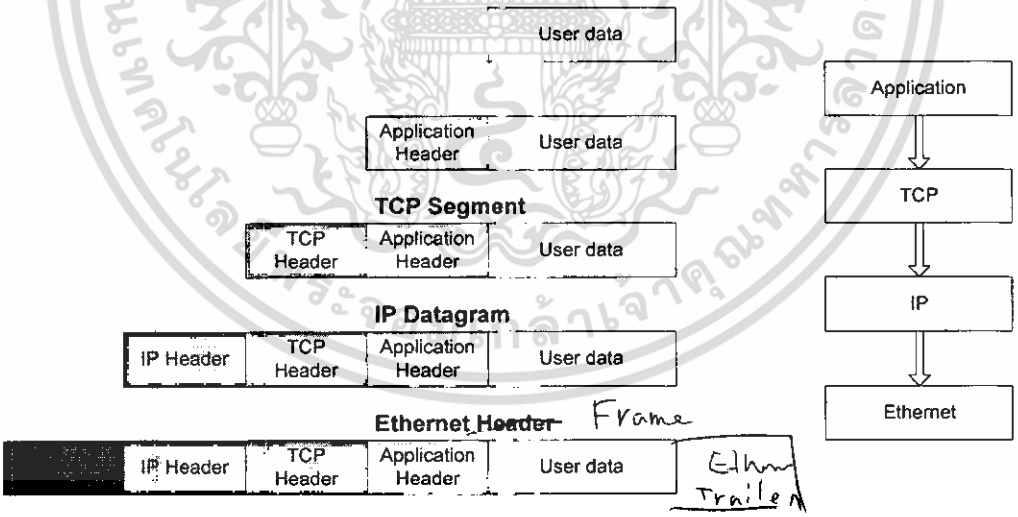


รูปที่ 2.6 โครงสร้างของโปรโตคอลทีซีพี/ไอพีในแต่ละชั้นหรือเลเยอร์จะมีโปรโตคอลหลักทำหน้าที่ต่าง ๆ และส่งผ่านข้อมูลไปยังเครือข่ายและออกสู่อินเตอร์เน็ต

กล่าวโดยสรุปก็คือ โปรโตคอลทีซีพี/ไอพีทำงานโดยแบ่งเป็นชั้นเทียบกับโอเอสไอโมเดลได้ กลไกในการทำงานของโปรโตคอลทีซีพี/ไอพีมี 4 ขั้นซึ่งในขั้นแรกคือโปรเซสเลเยอร์ทำหน้าที่ติดต่อกับแอพพลิเคชั่นและโปรโตคอลที่แอพพลิเคชั่นนั้น ๆ ใช้งาน และส่งต่อมาให้ชั้นโฮสต์ทูโฮสต์เลเยอร์เพื่อติดต่อกันระหว่างเครื่องเซิร์ฟเวอร์ให้บริการกับเครื่องผู้ขอใช้บริการในชั้นนี้จะมีการสร้างเซสชั่นหรือการเชื่อมต่อระหว่างระบบขึ้นตามแต่ละโปรโตคอลที่ต้องการ ต่อมาเป็นการผนึกข้อมูลไปเป็นไอพีดาต้าแกรมที่ชั้นอินเตอร์เน็ตเวิร์คเลเยอร์โดยอาศัยโปรโตคอลไอพี เพื่อให้สามารถติดต่อส่งข้อมูลข้ามเครือข่ายไปยังเครือข่ายและเครื่องที่ถูกต้องได้ และสุดท้ายการส่งออกข้อมูลสู่โลกภายนอกต้องอาศัยกลไกในชั้นเน็ตเวิร์คอินเตอร์เฟส เลเยอร์เพื่อแปลงข้อมูลใหม่ เพิ่มข้อมูลที่จำเป็นในการอ้างอิงตำแหน่งและแปลงข้อมูลเป็นสัญญาณไฟฟ้าและส่งออกไปยังเครือข่ายและอาจออกไปยังเกทเวย์ (Gateway) หรือเราเตอร์ (Router) เพื่อข้ามเครือข่ายออกไปยังเส้นทางที่กำหนดไว้ในอินเตอร์เน็ตต่อไป

2.1.2 เอนแคปซูเลชั่น/ดีมัลติเพลกซิ่ง (Encapsulation/Demultiplexing)

เวลาส่งข้อมูล เมื่อข้อมูลถูกส่งผ่านในแต่ละเลเยอร์ แต่ละเลเยอร์จะทำการประกอบข้อมูลที่ได้รับมากับส่วนควบคุมซึ่งอยู่ส่วนหัวของข้อมูลเรียกว่าเฮดเดอร์ (Header) ภายในเฮดเดอร์จะบรรจุข้อมูลที่สำคัญของโปรโตคอลที่ทำการเอนแคปซูเลตเมื่อผู้รับได้รับข้อมูล ก็จะเกิดกระบวนการทำงานย้อนกลับคือโปรโตคอลเดียวกัน ทางฝั่งผู้รับก็จะได้รับข้อมูลส่วนที่เป็นเฮดเดอร์ก่อนและนำไปประมวลและทราบว่าข้อมูลที่ตามมามีลักษณะอย่างไร ซึ่งกระบวนการย้อนกลับนี้เรียกว่าดีมัลติเพลกซิ่ง



รูปที่ 2.7 ขั้นตอนการเอนแคปซูเลชั่นเมื่อข้อมูลถูกส่งผ่านโปรโตคอลต่าง ๆ

ข้อมูลที่ผ่านการเอนแคปซูเลตในแต่ละระดับมีชื่อเรียกแตกต่างกัน ข้อมูลที่มาจากผู้ใช้หรือก็คือข้อมูลที่ผู้ใช้เป็นผู้ป้อนให้กับแอพพลิเคชั่นเรียกว่ายูสเซอร์ดาต้า (User Data) เมื่อแอพพลิเคชั่นได้รับข้อมูลจากผู้ใช้ก็จะนำมาประกอบกับส่วนหัวของแอพพลิเคชั่น เรียกว่าแอพพลิเคชั่นดาต้า (Application Data)

และส่งต่อไปยังโปรโตคอลทีซีพีเมื่อโปรโตคอลทีซีพีได้รับแอพพลิเคชั่นดาต้าก็จะนำมารวมกับเฮดเดอร์ของโปรโตคอลทีซีพีเรียกว่าทีซีพีเซกเมนท์ (TCP Segment) และส่งต่อไปยังโปรโตคอลไอพี เมื่อโปรโตคอลไอพีได้รับทีซีพีเซกเมนท์ก็จะนำมารวมกับเฮดเดอร์ของโปรโตคอลไอพี เรียกว่าไอพีดาต้าแกรม (IP Datagram) และส่งต่อไปยังเลเยอร์ดาต้าลิงค์เลเยอร์ (Data link Layer) ในระดับดาต้าลิงค์จะนำไอพีดาต้าแกรมมาเพิ่มส่วนแอร์เรอร์คอร์เรกชั่น (Error Correction) และแฟลก (flag) เรียกว่าอีเธอร์เน็ตเฟรม (Ethernet Frame) ก่อนจะแปลงข้อมูลเป็นสัญญาณไฟฟ้าส่งผ่านสายสัญญาณที่เชื่อมโยงอยู่ต่อไป

2.2 อินเตอร์เน็ตโปรโตคอล (Internet Protocol)

ด้วยเหตุที่ไอพี เป็นโปรโตคอลหลักในการสื่อสารข้อมูล และถือได้ว่าเป็นหัวใจสำคัญของโปรโตคอลทีซีพี/ไอพีจะขอยกมาอธิบายก่อน เพื่อให้ง่ายต่อการอธิบายโปรโตคอลตัวอื่น ๆ ต่อไป ซึ่งจะได้เรียนรู้เกี่ยวกับหน้าที่และลักษณะของโปรโตคอลไอพี อินเตอร์เน็ตแอดเดรส (Internet Address) รูปร่างของไอพีเฮดเดอร์ (IP Header) การเราติ้ง (Routing) และการจัดสรรไอพีด้วยซับเน็ต (Subnet)

ไอพีเป็นโปรโตคอลที่ทำหน้าที่รับภาระในการนำข้อมูลไปส่งยังผู้รับที่เชื่อมต่ออยู่ในระบบเน็ตเวิร์คซึ่งทั้งสองฝั่งอาจอยู่คนละเน็ตเวิร์คกันก็ได้โปรโตคอลอื่น ๆ ในระดับเน็ตเวิร์คเลเยอร์ขึ้นไปทั้งทีซีพี ยูดีพี ไอซีเอ็มพี ต่างก็ต้องอาศัยโปรโตคอลไอพี ในการรับส่งข้อมูลทั้งสิ้น

โปรโตคอลไอพีมีความสามารถในการค้นหาเส้นทางจากผู้รับไปยังผู้ส่ง มีกลไกที่ชาญฉลาดในการค้นหาเส้นทาง สามารถค้นหาเส้นทางไปถึงผู้รับได้เอง หากมีเส้นทางที่สามารถไปได้ แต่ไม่ได้ติดต่อระหว่างผู้รับกับผู้ส่งโดยตรงและไม่มีการยืนยันว่าข้อมูลถึงผู้รับจริงหรือไม่ ทั้งนี้อาจเกิดจากหลายสาเหตุ เช่น ที่อยู่ของผู้รับไม่มีการเชื่อมต่ออยู่ในระบบอินเตอร์เน็ต กล่าวได้ว่าโปรโตคอลไอพีมีหน้าที่ในการค้นหาเส้นทางเท่านั้น ไม่มีการยืนยันผลสำเร็จในการส่งข้อมูล หากเกิดข้อผิดพลาดในการส่งข้อมูล แม้ว่าจะมีการส่งไอซีเอ็มพีแมสเซจ (ICMP massage) กลับมารายงานข้อผิดพลาด แต่ก็รับประกันไม่ได้อยู่ดีว่าไอซีเอ็มพีแมสเซจจะกลับมาถึงเรียบร้อยหรือไม่ ด้วยเหตุนี้ จึงถือว่าไอพีเป็นโปรโตคอลที่ไม่มีความน่าเชื่อถือ (reliable)

2.2.1 ไอพีแอดเดรสซิ่ง (IP Addressing)

ทุกอินเตอร์เฟสที่ต่ออยู่บนอินเตอร์เน็ตจะต้องมีหมายเลขประจำตัวเพื่อใช้ในการสื่อสารข้อมูล เครื่องคอมพิวเตอร์ทุกเครื่องที่ในเครือข่ายอินเตอร์เน็ตจะสามารถติดต่อถึงกันได้ โดยเครื่องคอมพิวเตอร์ต้นทางจะระบุที่อยู่ของเครื่องคอมพิวเตอร์ปลายทางที่ต้องการติดต่อด้วย ซึ่งที่อยู่ ของเครื่องคอมพิวเตอร์นี้จะถูกกำหนดใช้มาตรฐานที่เรียกว่า หมายเลขไอพี (IP Address) โดยค่าไอพีแอดเดรสนี้จะเป็นหมายเลขจำนวน 32 บิต แต่แทนที่จะกำหนดให้เลขทั้ง 32 บิตนั้นถูกนับต่อเนื่องกันไป ก็จะใช้วิธีการแบ่งหมายเลขดังกล่าวออกเป็นกลุ่มของเลขขนาด 8 บิตจำนวน 4 ชุด และคั่นแต่ละชุดด้วยจุด โดยในแต่ละกลุ่มจะสามารถแทนค่าเป็นจำนวนเต็มที่อยู่ในช่วง 0-255 (2^8 -1) ได้ และใช้จุดเป็นตัวแบ่งกลุ่ม เช่น

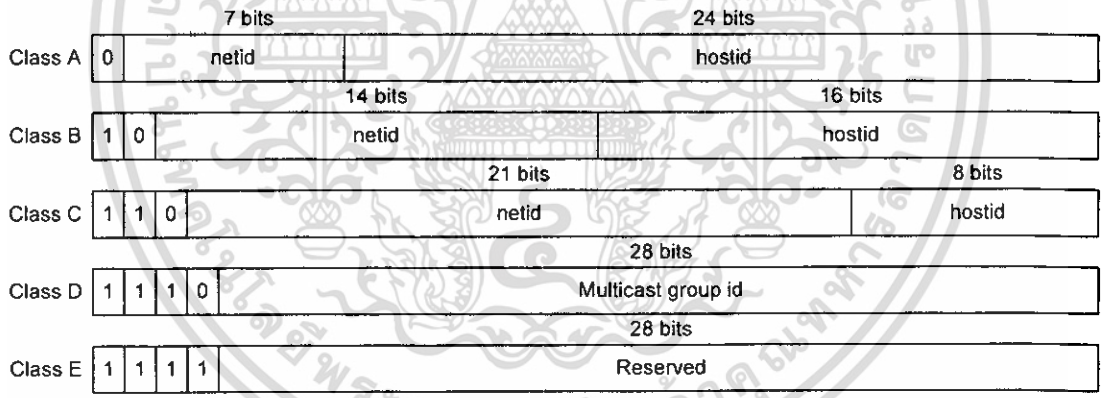
172.17.3.12

202.44.135.9

หมายเลขไอพีหนึ่ง ๆ จะแสดงถึงข้อมูลด้านเครือข่ายอยู่ 2 ส่วนด้วยกัน ส่วนแรกเป็น ส่วนแสดงหมายเลขเครือข่าย (Network Address) ส่วนที่สองเป็นส่วนแสดงหมายเลขคอมพิวเตอร์ (Host Address) ที่อยู่ในเครือข่ายหมายเลขนั้น การจัดแบ่งหมายเลขไอพีสามารถ แบ่งออกได้เป็น 5 คลาส (Class) คือ A B C D และ E แต่โดยทั่วไปนั้นจะใช้งานหลักอยู่เพียง 3 คลาส ได้แก่ คลาส A B และ C

เนื่องจากจำนวนคอมพิวเตอร์ทั่วโลกที่ทำการเชื่อมต่อเข้าสู่เครือข่ายอินเตอร์เน็ตได้เพิ่มขึ้นอย่างต่อเนื่อง ดังนั้นอีกในไม่ช้านี้ระบบหมายเลขไอพีที่ใช้ในปัจจุบันซึ่งเป็นข้อมูลตัวเลขขนาด 32 บิตจะไม่เพียงพอที่จะกำหนดให้แก่คอมพิวเตอร์ได้ทั่วโลก ดังนั้นจึงได้มีการเตรียมคิดค้นระบบหมายเลขไอพีแบบใหม่โดยขยายขนาดเป็นข้อมูลตัวเลขขนาด 128 บิต ซึ่งมีชื่อเรียกว่าอินเตอร์เน็ตโปรโตคอลเวอร์ชั่น 6 (Internet Protocol Version 6 หรือเรียกโดยย่อว่า IPv6)

ในการที่คอมพิวเตอร์แต่ละเครื่องถูกกำหนดให้มีหมายเลขไอพีที่ไม่ซ้ำกันเลยนั้น จะเห็นได้ว่าเป็นวิธีที่สะดวกที่คอมพิวเตอร์กับคอมพิวเตอร์ด้วยกันจะสามารถที่จะใช้อ้างอิงเพื่อติดต่อถึงกันได้ แต่สำหรับผู้ใช้งานเครือข่ายนั้นคงจะเกิดความยุ่งยากและสับสนเป็นอย่างมากในการที่จะต้องจดจำหมายเลขไอพีของเครื่องคอมพิวเตอร์ต่าง ๆ ที่ต้องการติดต่อด้วยจำนวนมาก ดังนั้นเพื่อเป็นการอำนวยความสะดวกให้แก่ผู้ใช้งานจึงเกิดแนวคิดในการสร้างระบบ “ชื่อโดเมน (Domain Name)” เพื่อใช้อ้างอิงในระดับผู้ใช้งานแทนระบบหมายเลขไอพีซึ่งก็จะคงถูกใช้อ้างอิงในระดับคอมพิวเตอร์ขึ้นมา



รูปที่ 2.8 การกำหนดไอพีแอดเดรสในคลาสต่าง ๆ

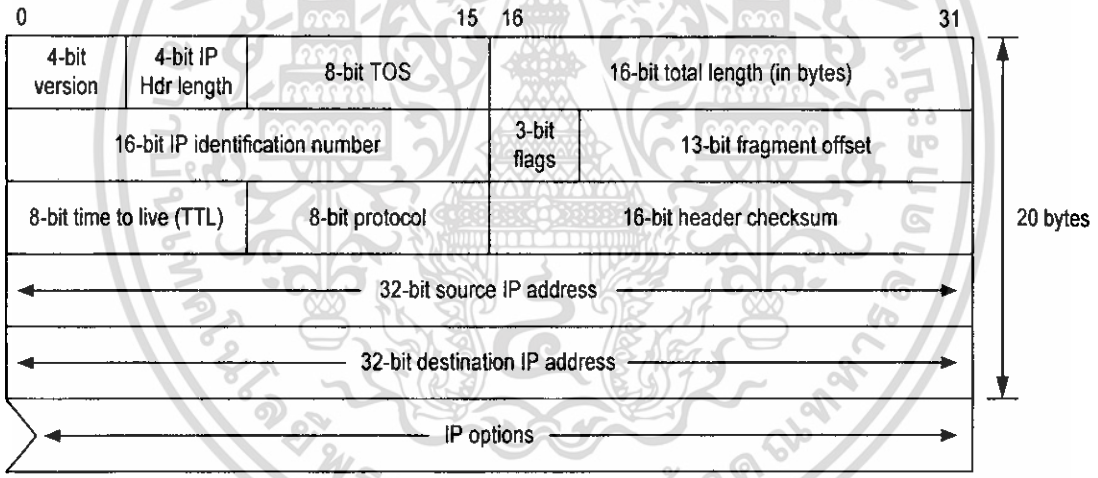
จากข้อกำหนดในการแบ่งคลาสของไอพีแอดเดรส หากลองนำบิตที่อยู่ในตอนต้นของไอพีแอดเดรสในแต่ละคลาสมาแปลงเป็นไอพีแอดเดรส ในเลขฐานสิบ จะเห็นว่าแต่ละคลาสครอบคลุมไอพีแอดเดรส ช่วงต่าง ๆ ดังตารางที่ 2.1

ตารางที่ 2.1 แสดงช่วงของไอพีแอดเดรสในแต่ละคลาส

Class	IP Range
A	0.0.0.0 - 127.255.255.255
B	128.0.0.0 - 191.255.255.255
C	192.0.0.0 - 223.255.255.255
D	224.0.0.0 - 239.255.255.255
E	240.0.0.0 - 255.255.255.255

2.2.2 ไอพีเฮดเดอร์ (IP Header)

เมื่อข้อมูลถูกส่งลงมาจากชั้นทรานสปอร์ตเลเยอร์สู่ชั้นเน็ตเวิร์คเลเยอร์กระบวนการเอนแคปซูเลทของไอพีโปรโตคอลจะทำการเพิ่มส่วนเฮดเดอร์ลงไปเฮดเดอร์ของไอพีดาต้าแกรมมีขนาด 20-32 ไบต์ มีส่วนประกอบต่าง ๆ ดังแสดงในรูปที่ 2.9



รูปที่ 2.9 แสดงไอพีเฮดเดอร์

ตารางที่ 2.2 แสดงตำแหน่งต่าง ๆ ของไอพีเฮดเดอร์

<table>
<tr><th>ตำแหน่ง</th><th>ชื่อ</th><th>อธิบาย</th></tr>
<tr><td>0-3</td><td>Version</td><td>มีขนาด 4 บิตเป็นเวอร์ชั่นของไอพี ปัจจุบันค่านี้ถูกกำหนดให้เป็น 4</td></tr>
<tr><td>4-7</td><td>Lenght</td><td>มีขนาด 4 บิตเป็นค่าความยาวของเฮดเดอร์นี้ โดยปกติจะเป็น 5 หมายความว่า 5*32 บิต = 20 ไบต์</td></tr>
<tr><td>8-15</td><td>Type of Service</td><td>เป็นข้อมูลขนาด 8 บิต ปัจจุบันไม่ได้ใช้งานแล้ว</td></tr>
<tr><td>16-31</td><td>Total length</td><td>เป็นฟิลด์ที่บอกจำนวนไบต์ทั้งหมดของไอพีดาต้าแกรม ด้วยขนาด 16 บิตทำให้ดาต้าแกรมมีขนาดสูงสุดไม่เกิน 65535 ไบต์ และมีขนาดเล็กสุดไม่ต่ำกว่า 512 ไบต์</td></tr>
<tr><td>32-47</td><td>Identification</td><td>ใช้ในกรณีที่มีการแบ่งดาต้าแกรมออกเป็นแฟรกเมนต์ เมื่อนำกลับมารวมกันใหม่จะได้รู้ว่ามาจากดาต้าแกรมเดียวกัน</td></tr>
<tr><td>48-50</td><td>Flag</td><td>ใช้ในกรณีที่มีการแบ่งข้อมูลออกเป็นแฟรกเมนต์ มีความหมายดังนี้
<table>
<tr><td>บิต 0 : reserved</td><td>เป็น 0 เสมอ</td></tr>
<tr><td>บิต 1 (DF)</td><td>0 = May Fragment 1 = Don't Fragment</td></tr>
<tr><td>บิต 2 (MF)</td><td>0 = Last Fragment 1 = More Fragments.</td></tr>
</table></td></tr>
<tr><td>51-63</td><td>fragment offset</td><td>เป็นส่วนระบุข้อมูลที่ใช้แยกรวมข้อมูล เพื่อให้ข้อมูลที่ถูกแยกออกเป็นแฟรกเมนต์กลับมารวมกันได้อย่างถูกต้องตามลำดับ</td></tr>
<tr><td>64-71</td><td>Time to Live (TTL)</td><td>เป็นจำนวนครั้งสูงสุดที่ดาต้าแกรมนี้จะถูกส่งผ่านเครือข่ายไปยังปลายทางได้ เพื่อป้องกันไม่ให้ดาต้าแกรมถูกเราต์ไปเรื่อย ๆ อย่างไม่สิ้นสุด ปกติค่านี้จะเริ่มต้นที่ 32 และจะถูกลดค่าลงทีละ 1 เมื่อมีการเราต์ จนค่านี้มีค่าเป็น 0 ก็จะไม่ถูกเราต์อีกต่อไป</td></tr>
<tr><td>72-79</td><td>Protocol</td><td>เป็นข้อมูลที่ระบุโปรโตคอลที่ส่งดาต้าแกรมนี้มา ตัวอย่างโปรโตคอลที่ใช้บ่อย ๆ ได้แก่
<table>
<tr><th>โปรโตคอล</th><th>ค่าในฟิลด์ Protocol</th><th>อธิบาย</th></tr>
<tr><td>ICMP</td><td>1</td><td>Internet Control Message Protocol</td></tr>
<tr><td>TCP</td><td>6</td><td>Transmission Control Protocol</td></tr>
<tr><td>UDP</td><td>17</td><td>User Datagram Protocol</td></tr>
</table></td></tr>
</table>

ตารางที่ 2.2 (ต่อ) แสดงตำแหน่งต่าง ๆ ของไอพีเฮดเดอร์

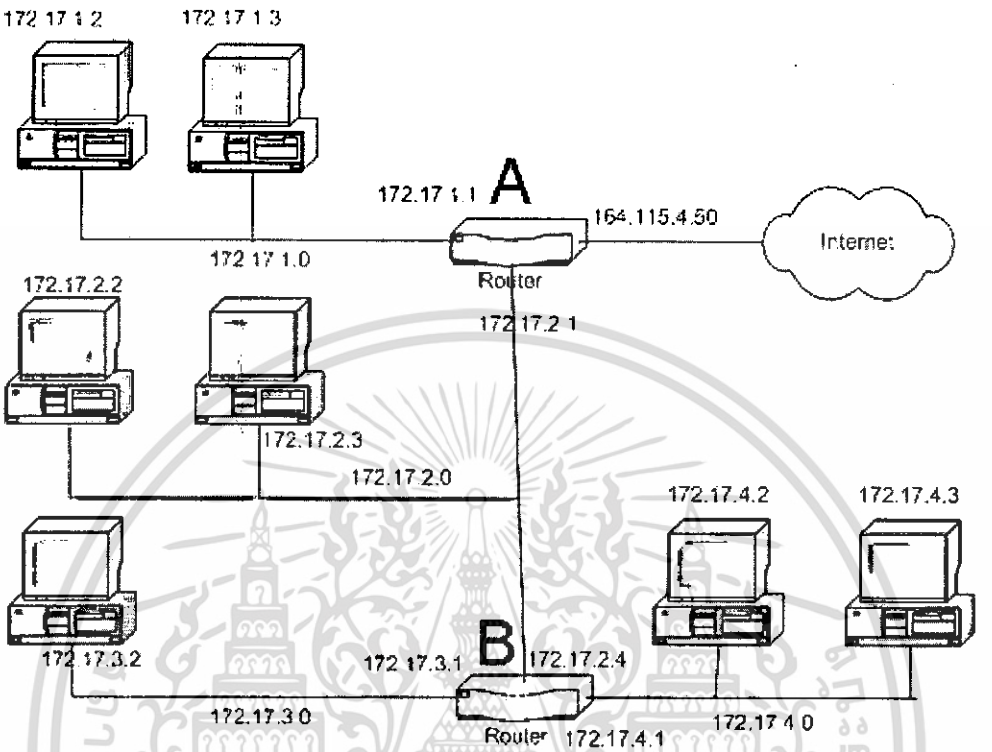
ตำแหน่ง	ชื่อ	อธิบาย
80-95	Header Checksum	เป็นส่วนตรวจสอบความถูกต้องของข้อมูลในเฮดเดอร์โดยไม่เกี่ยวกับส่วนข้อมูลที่อยู่ภายในเพย์โหลด (payload) ค่านี้จะถูกคำนวณใหม่ทุกครั้งที่มีการเปลี่ยนแปลงข้อมูลในเฮดเดอร์ (เช่น ทีทีแอลที่มีการเปลี่ยนแปลงทุกครั้งที่ไอพีดาต้าแกรม ถูกส่งผ่านเราเตอร์)
86-127	Source IP Address	คือไอพีแอดเดรสของผู้ส่งดาต้าแกรม
128-163	Destination IP Address	คือไอพีแอดเดรสของผู้รับดาต้าแกรม
ไม่แน่นอน	Option	มีขนาดข้อมูลไม่แน่นอน ใช้สำหรับกำหนดค่าพารามิเตอร์ปลีกย่อย ซึ่งส่วนใหญ่ไม่มีการนำไปใช้งาน
ขึ้นอยู่กับออพชั่น	Padding	มีข้อมูลว่างเปล่า ใช้เป็นส่วนเติมเต็มของฟิลด์ออพชั่นให้ครบ 32 ไบต์

2.2.3 ไอพีเราติ้ง (IP Routing)

ไอพีเราติ้งเป็นกระบวนการค้นหาเส้นทางในการส่งผ่านข้อมูลจากต้นทางไปยังปลายทางโดยผ่านการส่งต่อข้อมูลไปจนกว่าจะถึงปลายทาง นับเป็นกลไกสำคัญที่ทำให้ไอพี เป็นโปรโตคอลที่สามารถส่งข้อมูลจากโฮสต์หนึ่งไปอีกโฮสต์หนึ่งได้แม้ว่าจะอยู่ไกลแสนไกล เริ่มต้นทฤษฎีของไอพีเราติ้งด้วยการทำความรู้จักกับส่วนประกอบต่าง ๆ ของเน็ตเวิร์ค ในแง่ของไอพีเราติ้งกันก่อน

- โฮสต์ เป็นอุปกรณ์ที่ทำหน้าที่ให้กำเนิดข้อมูลในกรณีเป็นผู้ส่ง หรือทำหน้าที่รับข้อมูลไปใช้งานในกรณีเป็นผู้รับ การสื่อสารข้อมูลใด ๆ จะต้องเป็นการสื่อสารจากโฮสต์ไปยังโฮสต์เสมอ สำหรับไอพีแพคเกต (IP Packet) แล้วข้อมูลในเฮดเดอร์ที่ปรากฏอยู่ในฟิลด์ซอร์สแอดเดรส (Source Address) และเดสติเนชั่นแอดเดรส (Destination Address) ซึ่งเรียกว่าไอพีแอดเดรส จะเป็นหมายเลขระบุตำแหน่งของโฮสต์ต้นทางและโฮสต์ปลายทางเท่านั้น
- เน็ตเวิร์ค เป็นเครือข่ายที่มีการเชื่อมต่อกันของโฮสต์ 2 ตัวขึ้นไป โฮสต์แต่ละตัวในเน็ตเวิร์คเดียวกันสามารถเชื่อมต่อถึงกันได้โดยตรง
- เราเตอร์ทำหน้าที่ในการส่งผ่านข้อมูลจากเน็ตเวิร์คหนึ่งไปยังอีกเน็ตเวิร์คหนึ่ง ตำแหน่งของเราเตอร์จะอยู่ในจุดที่เชื่อมต่อระหว่างสองเน็ตเวิร์คเข้าด้วยกัน ด้วยข้อกำหนดของไอพี ข้อมูลจะส่งไปถึงกันโดยตรงข้ามเน็ตเวิร์คไม่ได้จะต้องอาศัยเราเตอร์เป็นผู้ทำหน้าที่ส่งผ่านข้อมูลไปให้ ในเราเตอร์จะมีเราติ้งเทเบิล (Routing Table) สำหรับเก็บข้อมูล เพื่อใช้ในการพิจารณาเลือกเส้นทางในการส่งดาต้าแกรม ซึ่งจะอธิบายกลไกการทำงานในหัวข้อถัดไป ในการอธิบาย

กระบวนการเราติ้งให้เป็นที่เข้าใจในเบื้องต้นจะขออธิบายจากเน็ตเวิร์คตัวอย่างที่แสดงในรูปที่ 2.10



รูปที่ 2.10 เน็ตเวิร์คตัวอย่าง

การเราติ้งจะเป็นไปตามขั้นตอนดังนี้

1. ถ้าโฮสต์ต้นทางและปลายทางต่อเชื่อมร่วมอยู่ในเน็ตเวิร์คเดียวกัน มีการเชื่อมต่อถึงกันโดยตรง เช่น อีเธอร์เน็ตหรือโทเค็นริง ดังแสดงในรูปที่ 2.10 เป็นการติดต่อระหว่าง 172.17.2.2 และ 172.17.2.3 ไอพีดาต้าแกรมก็จะถูกส่งไปยังโฮสต์ปลายทางโดยตรง

2. หากโฮสต์ต้นทางและปลายทางไม่ได้อยู่ในเน็ตเวิร์คเดียวกันไอพีดาต้าแกรมจะถูกส่งไปยังดีฟอลต์เราเตอร์ (default router)

3. เมื่อเราเตอร์ได้รับไอพีดาต้าแกรมจากข้อ 2 แล้วตรวจสอบดู หากพบว่าโฮสต์ปลายทางต่อร่วมอยู่บนเน็ตเวิร์คเดียวกันกับเราเตอร์ ให้ทำการส่งดาต้าแกรมไปที่โฮสต์นั้น เช่น หาก 172.17.3.2 ต้องการส่งดาต้าแกรมไปยัง 172.17.4.2 จะต้องส่งดาต้าแกรมไปที่เราเตอร์ B และเราเตอร์ B จะส่งดาต้าแกรมต่อไปยังโฮสต์ปลายทาง

4. หากไม่ได้ต่อร่วมกันก็ส่งดาต้าแกรมไปที่เราเตอร์ตัวต่อไป โดยเราเตอร์จะเป็นผู้เลือกเส้นทาง ซึ่งมีอยู่ 2 กรณีคือ

1. ถ้ามีข้อมูลของโฮสต์ปลายทางอยู่ในเราติ้งเทเบิล เราเตอร์จะส่งดาต้าแกรมไปยังเราเตอร์ตัวที่ระบุไว้ในเราติ้งเทเบิล

2. ถ้าไม่มีข้อมูลของโฮสต์ปลายทางอยู่ในเราติ้งเทเบิล เราเตอร์จะส่งดาต้าแกรมไปยังดีฟอล์ตเราเตอร์ และกลับไปที่ขั้นตอนในข้อ 3 ใหม่ จนกว่าไอพีดาต้าแกรมจะเดินทางถึงปลายทางหรือหมดเวลาในการส่ง (TTL=0)

สมมติว่าเครื่อง 172.17.1.3 ต้องการติดต่อกับ 172.17.4.3 จะต้องส่งไอพีด้าต้าแกรมไปยังเราเตอร์ A หากเราเตอร์ A มีข้อมูลเกี่ยวกับ 172.17.4.3 อยู่ก็จะรู้ว่าต้องส่งดาต้าแกรมไปยังเราเตอร์ B คือ 172.17.2.4 และ เราเตอร์ B ก็จะส่งไอพีดาต้าแกรมไปยังโฮสต์ปลายทางได้สำเร็จ

2.2.4 ซับเน็ตแอดเดรสซิ่ง/ซับเน็ตมาส์ก (Subnet Addressing / Subnet Mask)

ในการใช้งานโปรโตคอลทีซีพี/ไอพีในอินเตอร์เน็ต นั้นการแบ่งออกเป็นแอดเดรสของเน็ตเวิร์ค (net id) และแอดเดรสของ โฮสต์ ตามที่ระบุของแต่ละคลาสค่อนข้างจะขาดประสิทธิภาพ คือในเน็ตเวิร์คคลาส A และ B แต่ละเน็ตเวิร์คนั้น สามารถมีจำนวนโฮสต์ได้มาก ซึ่งการที่จะนำไอพีแอดเดรสมาใช้อย่างทั่วถึงนั้นมีโอกาสเป็นไปได้ยากมากทั้งคลาส A และคลาส B เพราะมีโอกาสน้อยมากที่จะมีเน็ตเวิร์คใดในโลกมีจำนวนโฮสต์มากมายขนาดนั้นอยู่ภายในเน็ตเวิร์คเดียว ดังนั้นไอพีแอดเดรสที่จัดสรรให้ไปในแต่ละเน็ตเวิร์คของ คลาสเหล่านี้จึงถูกใช้ไม่หมดและไม่สามารถนำไปใช้ประโยชน์ที่อื่นได้เลย ดังนั้นเพื่อให้การจัดสรรไอพี เป็นไปอย่างมีประสิทธิภาพ จึงมีการนำส่วนของโฮสต์ไอดี (host id) มาแบ่งย่อยเป็นสองส่วน คือ ซับเน็ตไอดี (subnet id) และโฮสต์ไอดีทำให้ได้เน็ตเวิร์คย่อยหลาย ๆ เน็ตเวิร์ค โดยในแต่ละเน็ตเวิร์ค มีจำนวนโฮสต์ไม่มากเกินไปและเพียงพอต่อการใช้งาน

การแบ่งซับเน็ตใช้เทคนิคที่เรียกว่าซับเน็ตมาส์กซึ่งเป็นตัวเลขมีความยาว 32 บิต แบ่งออกเป็น 4 ชุด เช่นเดียวกับไอพี แต่ค่าของซับเน็ตมาส์กจะขึ้นอยู่กับความต้องการในการแบ่งซับเน็ตว่าต้องการจำนวนซับเน็ตเท่าใดและมีจำนวนโฮสต์เท่าใด หากนำซับเน็ตมาส์กมาเขียนเป็นเลขฐานสอง จะมีลักษณะพิเศษคือ ขึ้นต้นด้วยเลข 1 มีจำนวนกี่ตัวก็ได้ ตามแต่ความต้องการในการแบ่งซับเน็ตและตำแหน่งที่เหลือจะมีค่าเป็น 0 ความสัมพันธ์ระหว่างไอพีแอดเดรส ซับเน็ตมาส์ก โฮสต์ไอดี เน็ตไอดี จำนวนซับเน็ต และ จำนวนโฮสต์จะเป็นดังนี้

- host ID = (IP address) AND ~ (subnet mask)
- net ID = (IP address) AND (subnet mask)
- จำนวน host = ((2^จำนวนบิตที่เป็น 0 ของ subnet mask)-2)

เนื่องจากไอพีแรกของซับเน็ตถูกใช้เป็นเน็ตไอพีและไอพีสุดท้ายของซับเน็ตถูกใช้เป็นบรอดคลาสไอดี (broadcast ID)

- จำนวน subnet = (2^(จำนวนบิตที่เป็น 1 ของซับเน็ตมาส์กในตำแหน่งที่เป็นโฮสต์ไอดีของไอพีแอดเดรส))

2.2.5 ลักษณะการใช้ไอพีแอดเดรสในองค์กร

ลักษณะการใช้ไอพีแอดเดรสในองค์กร มีวิธีการจัดสรรและกำหนดเพื่อให้ใช้งาน แต่เนื่องจากหลายหน่วยงานติดขัดด้วยจำนวนหมายเลขที่ได้รับ เช่น องค์กรขนาดใหญ่ แต่ได้รับคลาส C จึงย่อมสร้างความยุ่งยากในการสร้างเครือข่ายผู้ใช้อินเตอร์เน็ตทุกคนต้องเกี่ยวข้องกับไอพีแอดเดรส อย่างน้อย พีซีที่ต่ออยู่กับอินเตอร์เน็ตต้องมีการกำหนดไอพีแอดเดรส

คำว่าไอพีแอดเดรส จึงหมายถึงเลขหรือรหัสที่บ่งบอกตำแหน่งของเครื่องที่ต่ออยู่บนอินเตอร์เน็ต ตัวเลขรหัสไอพีแอดเดรสจึงเสมือนเป็นรหัสประจำตัวของเครื่องที่ใช้ ตั้งแต่พีซี ของผู้ใช้จนถึงเซิร์ฟเวอร์ให้บริการอยู่ทั่วโลก ทุกเครื่องต้องมีรหัสไอพีแอดเดรสและต้องไม่ซ้ำกันเลยทั่วโลก
ไอพีแอดเดรสที่ใช้กันอยู่นี้เป็น ตัวเลขไบนารีขนาด 32 บิตหรือ 4 ไบต์

11101001 11000110 00000010 01110100

แต่เมื่อต้องการเรียกไอพีแอดเดรสจะเรียกแบบไบนารีคงไม่สะดวก จึงแปลงเลขไบนารีหรือเลขฐานสองแต่ละไบต์ (8 บิต) ให้เป็นตัวเลขฐานสิบโดยมีจุดคั่น

11101001	11000110	00000010	01110100
158	108	2	71

เมื่อตัวเลขไอพีแอดเดรสจำเป็นอย่างยิ่งสำหรับกำหนดให้กับเครื่อง และอินเตอร์เน็ตเติบโตรวดเร็วมาก เป็นผลทำให้ไอพีแอดเดรสเริ่มหายากขึ้น

การกำหนดไอพีแอดเดรสเน้นให้องค์กรจดทะเบียนเพื่อขอไอพีแอดเดรสและมีการแบ่งไอพีแอดเดรส ออกเป็นกลุ่มสำหรับองค์กร เรียกว่า คลาส โดยแบ่งเป็น คลาส A คลาส B คลาส C

คลาส A กำหนดตัวเลขในฟิลด์แรกเพียงฟิลด์เดียว ที่เหลืออีกสามฟิลด์ให้องค์กรเป็นผู้กำหนด ดังนั้นจึงมีไอพีแอดเดรสในองค์กรเท่ากับ 256 x 256 x 256

คลาส B กำหนดตัวเลขให้สองฟิลด์ ที่เหลืออีกสองฟิลด์ให้องค์กรเป็นผู้กำหนดดังนั้นองค์กรจึงมีไอพีแอดเดรส ที่กำหนดได้ถึง 256 x 256 = 65536 แอดเดรส

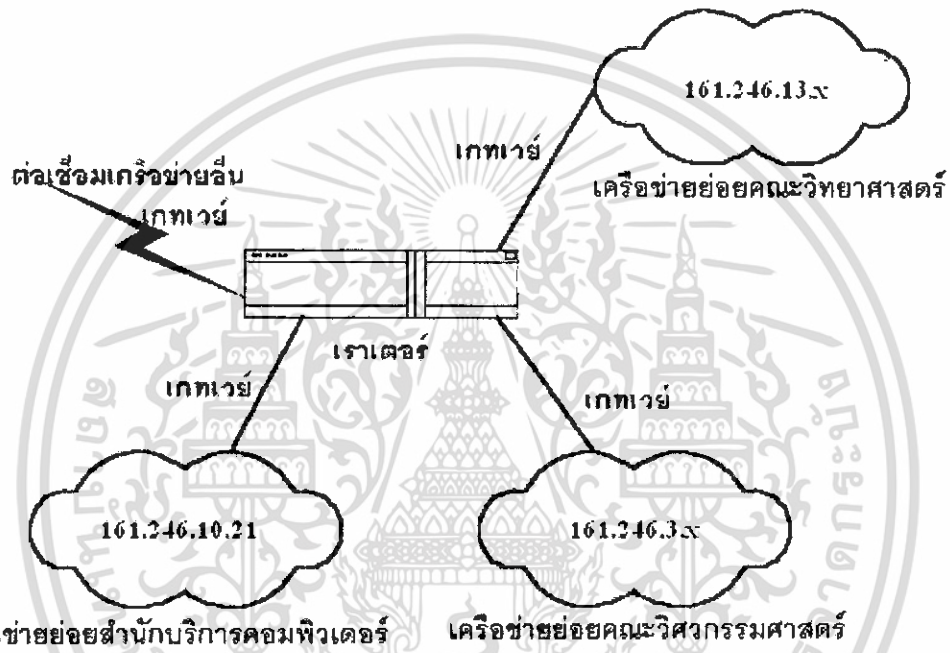
คลาส C กำหนดตัวเลขให้สามฟิลด์ ที่เหลือให้องค์กรกำหนดได้เพียงฟิลด์เดียว คือมีไอพีแอดเดรส 256

เมื่อพิจารณาตัวเลขไอพีแอดเดรส หากไอพีแอดเดรสใดมีตัวเลขขึ้นต้น 1-126 ก็จะเป็นคลาส A ดังนั้นคลาส A จึงมีได้เพียง 126 องค์กรเท่านั้น หากขึ้นต้นด้วย 128-191 ก็จะเป็นคลาส B เช่น ไอพีแอดเดรสของสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังขึ้นต้นด้วย 161 จึงอยู่ในคลาส B และหากขึ้นต้นด้วย 192-223 ก็เป็นคลาส C

ลักษณะการใช้ไอพีแอดเดรสในองค์กรจึงมีวิธีการจัดสรรและกำหนดเพื่อให้ใช้งาน แต่เนื่องจากหลายหน่วยงานติดขัดด้วยจำนวนหมายเลขที่ได้รับ เช่น องค์กรขนาดใหญ่ แต่ได้รับคลาส C จึงย่อมสร้างความยุ่งยากในการสร้างเครือข่าย

สำหรับสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังเป็นสถาบันที่มีไอพีคลาส B จึงได้แบ่งและจัดสรรไอพีแอดเดรสให้กับหน่วยงานต่าง ๆ ได้อย่างพอเพียง

ไอพีแอดเดรสแต่ละกลุ่มที่ได้รับการจัดสรร จะได้รับการควบคุมการกำหนดเส้นทางโดยอุปกรณ์จำพวก เราเตอร์ และสวิตชิ่ง



รูปที่ 2.11 แสดงเครือข่ายไอพีแอดเดรสในองค์กร

ทำนองเดียวกัน หน่วยงานย่อยรับแอดเดรสไปเป็นกลุ่มก็สามารถนำไอพีแอดเดรส ที่ได้รับไปจัดสรรแบ่งกลุ่มด้วยอุปกรณ์เราเตอร์หรือสวิตชิ่งได้ การกำหนดแอดเดรสจะต้องอยู่ภายในกลุ่มของตนเท่านั้น มิฉะนั้นอุปกรณ์เราเตอร์จะไม่สามารถทำงานรับส่งข้อมูลได้

ไอพีแอดเดรสจึงเป็นรหัสหลักที่จำเป็นในการสร้างเครือข่าย เครือข่ายทุกเครือข่ายจะต้องมีการกำหนดแอดเดรส สำนักบริการคอมพิวเตอร์ได้จัดสรรกลุ่มไอพีไว้ให้หน่วยงานต่าง ๆ อย่างพอเพียงโดยที่แอดเดรสทุกแอดเดรสที่ใช้ในกลุ่ม เช่น การเซ็ตให้กับพีซีแต่ละเครื่องต้องไม่ซ้ำกัน รหัสไอพีแอดเดรสจึงเป็นโครงสร้างพื้นฐานอย่างหนึ่งที่มีค่าสำหรับองค์กร

2.3 เน็ตเวิร์คคอนเซ็ปท์ (Network Concept)

ในปัจจุบันมีการนำคอมพิวเตอร์เข้ามาใช้งานในหน่วยงานประเภทต่าง ๆ มากมาย ซึ่งมีผลทำให้การทำงานในองค์กรหรือหน่วยงาน สามารถทำงานได้อย่างเป็นระบบ และสามารถพัฒนาการทำงานได้

อย่างต่อเนื่อง ซึ่งการนำคอมพิวเตอร์เข้ามาใช้ในองค์กรหรือหน่วยงานก็เริ่มมีการพัฒนาขึ้นแทนที่จะใช้ในลักษณะหนึ่งเครื่องต่อหนึ่งคน ก็ให้มีการใช้เครื่องคอมพิวเตอร์ อุปกรณ์ และข้อมูลต่าง ๆ ร่วมกัน โดยนำคอมพิวเตอร์มาต่อเชื่อมกัน ซึ่งเรียกสิ่งนี้ว่า "ระบบแลน" ความจริงแล้วระบบแลนถูกนำมาใช้เป็นเวลานานแล้ว แต่จะจำกัดการใช้งานอยู่ในเฉพาะกลุ่มคนบางกลุ่มเท่านั้น แต่ในปัจจุบันระบบแลนถูกนำมาใช้อย่างแพร่หลายมากขึ้น จึงจำเป็นต้องมีการจัดระบบการใช้งาน นิยามความหมายของเน็ตเวิร์คสามารถจำกัดได้มากมายหลายวิธี เช่น

- ตามขนาด: แบ่งเป็นเวิร์คกรุ๊ป (Workgroup) แลน (LAN) แมน (MAN) และ แวน (WAN)
- ลักษณะการทำงาน: แบ่งเป็นเพียร์ทูเพียร์ (peer-to-peer) และไคล์เอนท์เซิร์ฟเวอร์ (client-server)
- ตามรูปแบบ: แบ่งเป็น บัส ริง และสตาร์
- ตามแบนด์วิท: แบ่งเป็นเบสแบนด์และบรอดแบนด์หรือว่าเป็นเมกะบิตและจิกกะบิตต่อวินาที
- ตามสถาปัตยกรรม: แบ่งเป็นอีเธอร์เน็ตหรือโทเค็นริง (Token-Ring)

แบ่งตามขนาด

การเชื่อมต่อเครื่องคอมพิวเตอร์เข้าด้วยกันเป็นระบบเครือข่ายเน็ตเวิร์ค จึงมีการนำมาใช้กันมาก ซึ่งจะแบ่งได้เป็น 3 ระบบ คือ

1. ระบบเครือข่ายระยะไกล (Wide Area Network หรือ WAN)
2. ระบบเครือข่ายเน็ตเวิร์คระยะกลาง (Metropolitan Area Network หรือ MAN)
3. ระบบเครือข่ายเน็ตเวิร์คระยะใกล้ (Local Area Network หรือ LAN)

ซึ่งระบบแลนจะเป็นที่นิยมใช้กันอย่างแพร่หลาย ภายในชั้น ภายในตึก หรือระหว่างตึกที่อยู่ในบริเวณเดียวกัน หรือในสำนักงานทั่วไป ระบบเน็ตเวิร์คระยะใกล้หรือแลน สามารถติดตั้งได้ง่าย ส่งข้อมูลได้ด้วยความเร็วสูง มีข้อผิดพลาดน้อย และลงทุนน้อยกว่าระบบเน็ตเวิร์คระยะไกล และระยะกลาง ซึ่งต้องลงทุนสูงเนื่องจากเป็นระบบใหญ่ ใช้ติดต่อกันในระดับประเทศ

แบ่งตามลักษณะการทำงานของแลน

แลนแบ่งลักษณะการทำงานได้เป็น 2 ประเภทคือเพียร์ทูเพียร์และไคลเอนท์เซิร์ฟเวอร์

1. แบบเพียร์ทูเพียร์

เครื่องคอมพิวเตอร์แต่ละเครื่องจะสามารถแบ่งทรัพยากรต่าง ๆ ไม่ว่าจะเป็นไฟล์หรือเครื่องพิมพ์ซึ่งกันและกันภายในเน็ตเวิร์ค เครื่องแต่ละเครื่องจะทำงานในลักษณะที่ทัดเทียมกัน การเชื่อมต่อแบบนี้มักทำในระบบที่มีขนาดเล็ก ๆ เช่น หน่วยงานขนาดเล็กที่มีเครื่องที่ทำการเชื่อมต่อกันประมาณไม่เกิน 10 เครื่อง เน็ตเวิร์คประเภทนี้สามารถจัดตั้งได้ง่าย ๆ ด้วยซอฟต์แวร์ธรรมดา ๆ เช่น Windows 95 และ 98

โดยเครื่องคอมพิวเตอร์ในระบบจะสามารถเป็นได้ทั้งเครื่องลูกข่าย (client) และเครื่องผู้ให้บริการ (server) โดยขึ้นอยู่กับว่าขณะใดขณะหนึ่งเครื่องไหนเป็นผู้ร้องขอทรัพยากรหรือว่าเป็นผู้แบ่งปันทรัพยากร

2. แบบไคลเอนท์เซิร์ฟเวอร์

เป็นระบบที่เครื่องคอมพิวเตอร์เครื่องหนึ่ง ต่อเข้ากับคอมพิวเตอร์อีกเครื่องหนึ่งเป็นอย่างน้อย ซึ่งเครื่องที่เชื่อมต่อด้วยนี้จะมีขนาดใหญ่ มีโปรเซสเซอร์ตั้งแต่หนึ่งตัวขึ้นไป ซึ่งอาจเป็นไปได้ทั้งเครื่องในระดับ Pentium หรือ RISC(Reduced Instruction Set Computing เช่น DEC Alpha AXP) แล้วก็ใช้ระบบปฏิบัติการที่เป็นเน็ตเวิร์ค (NOS หรือ Network Operating System)

โดยเฉพาะ เช่น Windows NT Server ซึ่งจะมีประสิทธิภาพสูงกว่า Windows95และ98 อีกทั้งยังได้รับการออกแบบและปรับแต่งมาเพื่อการทำงานในระบบสภาวะแวดล้อมแบบเน็ตเวิร์คโดยเฉพาะอีกด้วย หน้าที่ของเครื่องแม่ข่ายได้แก่ การควบคุมความปลอดภัยในระบบจัดการความคับคั่งในระบบเน็ตเวิร์ค หยิบยื่นทรัพยากรต่าง ๆ เช่น ข้อมูล โปรแกรม หรือการขอใช้อุปกรณ์ร่วมต่าง ๆ ตามแต่เครื่องลูกข่ายจะร้องขอ สำหรับเครื่องลูกข่าย จะเป็นเครื่องคอมพิวเตอร์ตั้งโต๊ะ (ไม่ใช่พวกเทอร์มินัล) ซึ่งก็จะใช้โอเอส (OS) ธรรมดา เช่น Windows 95 Windows 98 windows NT Workstation ซึ่งเครื่องลูกข่ายเหล่านี้โดยปกติจะใช้ความสามารถด้านการประมวลผลของตัวเองเพื่อจัดการกับข้อมูลที่ได้รับมาจากเซิร์ฟเวอร์ และในการทำงานร่วมกันระหว่างไคลเอนท์กับเซิร์ฟเวอร์นี้

เราจะเรียกการทำงานที่ด้านของเครื่องลูกข่ายว่าฟรอนต์เอนโปรเซสซิ่ง (Front-end Processing) และเรียกการทำงานในส่วนของเซิร์ฟเวอร์ว่าแบคเอนโปรเซสซิ่ง (Back-end Processing) หลักการไคลเอนท์เซิร์ฟเวอร์จะมีความยืดหยุ่นสูง เพราะนอกเหนือจากการเชื่อมต่อเข้าด้วยกันตามปกติแล้วยังสามารถเลือกที่จะเชื่อมต่อทั้งระบบเข้ากับเครื่องในระดับมินิคอมพิวเตอร์หรือเมนเฟรมได้อีกด้วย โดยเครื่องที่ทำหน้าที่ฟรอนต์เอ็น จะยังคงสามารถใช้งานในสภาวะแวดล้อมและโปรแกรมที่เราคุ้นเคยได้ดี ในขณะที่ผู้ใช้งานสามารถเลือกทำงานได้ทั้งงานในรูปแบบเครื่องเดี่ยว (stand alone) หรือแบบที่ประสานงานกับผู้ใช้รายอื่น รวมไปถึงการทำงานโดยอาศัยข้อมูลจำนวนเก็บอยู่ในเครื่องเมนเฟรมอีกด้วย

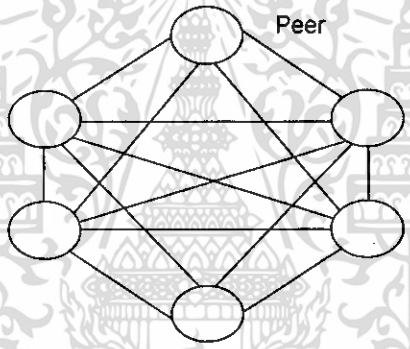
2.4 เพียร์ทูเพียร์ (Peer-to-Peer)

ในปัจจุบันเทคโนโลยีระบบเครือข่ายแบบเพียร์ทูเพียร์ได้รับความสนใจ และเข้ามามีบทบาทในการใช้อินเตอร์เน็ตมากขึ้น เทคโนโลยีนี้ช่วยทำให้ผู้ใช้สามารถแลกเปลี่ยนข้อมูล บริการ และทรัพยากรอื่น ๆ ในเครื่องคอมพิวเตอร์ที่อยู่บนเครือข่ายได้สะดวกมากยิ่งขึ้น ดังเช่น แนปสเตอร์ (Napster) ฟรีเน็ต (Freenet) ซึ่งเป็นโปรแกรมประยุกต์ที่ยอมให้ผู้ใช้อินเตอร์เน็ตค้นหา และแลกเปลี่ยนไฟล์ข้อมูลต่าง ๆ ระหว่างคอมพิวเตอร์ซึ่งกันและกันได้ โดยไม่จำเป็นต้องมีคอมพิวเตอร์แม่ข่าย (Central Server) ซึ่งต่างจากระบบไคล์เอนท์เซิร์ฟเวอร์ ซึ่งต้องมีคอมพิวเตอร์แม่ข่าย (Server) คอยให้บริการตามคำขอของเครื่องลูกข่าย (Client) ในการขอข้อมูล บริการ และไฟล์ข้อมูล ดังตัวอย่างที่พบเห็นโดยทั่วไปคือเวิร์ลไวด์เว็บ (World Wide Web) ทั่วไปที่มีอยู่โดยผู้ใช้อินเตอร์เน็ตซึ่งเปรียบได้เสมือนเครื่องลูกข่าย จะใช้เว็บบราวเซอร์ในการแสดงผลข้อมูลที่มาจากเครื่องแม่ข่าย (Web Server) โดยใช้ โปรโตคอลเอชทีทีพี (HTTP) เป็นมาตรฐานในการสื่อสารและมีรูปแบบการแสดงผลเป็นแบบเอชทีเอ็มแอล (HTML) ซึ่งหากจะเปรียบไปแล้วเทคโนโลยีระบบเครือข่ายแบบเพียร์ทูเพียร์จะมีการทำงานในลักษณะที่เป็นดีเซ็นทรัลไลเซชั่น (Decentralization) ส่วนระบบไคลเอนท์เซิร์ฟเวอร์มีการทำงานเป็นแบบเซ็นทรัลไลเซชั่น (Centralization)

นั้นเองเพียร์ทูเพียร์ สามารถแบ่งออกได้เป็น 3 แบบ คือ เพียวเพียร์ทูเพียร์ (Pure Peer-to-peer) ไฮบริดเพียร์ทูเพียร์ (Hybrid Peer-to-peer) และซุปเปอร์เพียร์ (Super Peer)

เพียวเพียร์ทูเพียร์

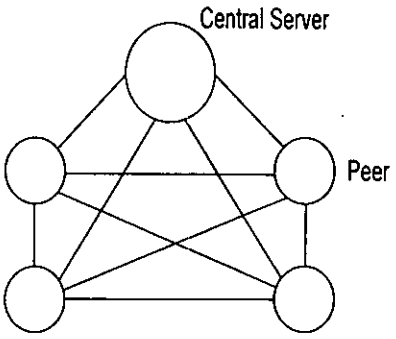
โมเดลแบบเพียวเพียร์ทูเพียร์จะมีลักษณะที่ตรงข้ามกับโมเดลแบบศูนย์กลางตรงที่ทุก ๆ เพียร์สามารถติดต่อและแลกเปลี่ยนข้อมูลกันได้โดยตรงโดยไม่ต้องผ่านเครื่องเซิร์ฟเวอร์กลาง จุดเด่นของโมเดลแบบนี้คือความสามารถในการขยายขนาดเครือข่าย ความคงทน (fault tolerant) โดยถ้ามีเพียร์เสียหรือออกไปจากระบบก็จะไม่ส่งผลกระทบต่อระบบโดยรวม แต่โมเดลแบบนี้ก็มีข้อจำกัดตรงที่ควบคุมการไหลของข้อมูลได้ยากทำให้มีปัญหาเรื่องการใช้แบนด์วิทธ์สิ้นเปลือง และโมเดลแบบนี้จะมีความปลอดภัยที่ต่ำ เนื่องจากแต่ละเพียร์สามาถเข้าสู่เครือข่ายได้โดยไม่ต้องมีการทำอัทเธนทิเคชั่น (Authentication: โมเดลแบบนี้ทำอัทเธนทิเคชั่นได้ยาก) และสามารถที่จะส่งข้อมูลที่อันตรายเข้าสู่เครือข่ายได้โดยง่าย เนื่องจากข้อเสียที่มากของโมเดลแบบนี้ทำให้โมเดลนี้ไม่เป็นที่นิยมเท่าที่ควร



รูปที่ 2.12 โมเดลเพียวเพียร์ทูเพียร์

ไฮบริดเพียร์ทูเพียร์

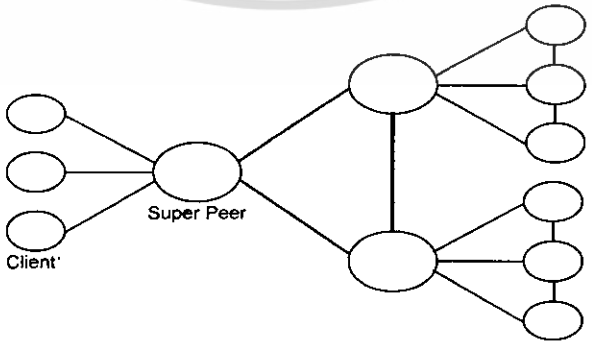
โมเดลแบบไฮบริดเพียร์ทูเพียร์นี้จะมีเครื่องเซิร์ฟเวอร์ ที่ทำหน้าที่ควบคุมรายละเอียดของข้อมูลที่อยู่ภายในเครือข่ายแต่การส่งข้อมูลจะเป็นแบบเดียวกับโมเดลเพียวเพียร์ทูเพียร์ (ส่งถึงกันโดยตรง) โมเดลแบบนี้จะช่วยลดปัญหาเรื่องการจัดการข้อมูลที่ทำได้ยากในโมเดลแบบเพียวเพียร์ทูเพียร์ โดยเครื่องเซิร์ฟเวอร์ จะทำหน้าที่คอยตรวจสอบสถานะของทุก ๆ เพียร์ และควบคุมการไหลของข้อมูลในเครือข่าย แต่เพราะยังต้องใช้เครื่องเซิร์ฟเวอร์กลางอยู่ดังนั้นถ้าเครื่องเซิร์ฟเวอร์กลางเสียไปก็จะเสียการควบคุมข้อมูลไปแต่ละเพียร์ ก็จะยังคงสามารถแลกเปลี่ยนข้อมูลกันได้อยู่ เนื่องจากมีการควบคุมข้อมูลที่ดี ดังนั้นโมเดลนี้จึงมีความสามารถในการขยายขนาดเครือข่ายได้ดีกว่าโมเดลเพียวเพียร์ทูเพียร์แต่ก็ยังมีขีดจำกัดของการขยายอยู่ที่จำนวนเครื่องลูกของเครื่องเซิร์ฟเวอร์ที่จะรับได้ โมเดลแบบนี้มีประสิทธิภาพที่จะนำไปใช้ กับแอปพลิเคชั่นต่าง ๆ แต่ไม่สามารถนำไปใช้กับแอพพลิเคชั่นที่มีขนาดของปัญหาใหญ่ ๆ ได้



รูปที่ 2.13 โมเดลไฮบริดเพียร์ทูเพียร์

ซุปเปอร์เพียร์

โมเดลแบบซุปเปอร์เพียร์เป็นโมเดลใหม่ที่เพิ่งจะเกิดขึ้นไม่นานมานี้โดยเป็นการเอาระบบแบบศูนย์กลางไปรวม อยู่ในระบบแบบกระจาย โมเดลแบบซุปเปอร์เพียร์จะช่วยลดปริมาณในการจัดการของเซิร์ฟเวอร์ อีกทั้งช่วยเพิ่มความสามารถในเรื่องของการขยายขนาดและความคงทนของเครือข่าย และลดปัญหาอื่น ๆ ที่เกิดขึ้นในโมเดลแบบเพียวเพียร์ทูเพียร์และไฮบริดเพียร์ทูเพียร์ ซุปเปอร์เพียร์คือเพียร์ ที่ทำหน้าที่เหมือนเป็นเซิร์ฟเวอร์กลางให้กับกลุ่มของไคลเอนท์แต่ละกลุ่มไคลเอนท์จะส่งคำร้องขอและรับผลลัพธ์ของคำร้องขอนั้นจากซุปเปอร์เพียร์ในขณะที่ซุปเปอร์เพียร์แต่ละเพียร์ ก็จะเชื่อมต่อถึงกันด้วยเครือข่ายแบบเพียวเพียร์ทูเพียร์โดยซุปเปอร์เพียร์จะทำหน้าที่เป็นตัวควบคุม (controller) ปรับแต่ง (configuration) ดูแล (administration) และรักษาความปลอดภัย (security) ให้กับไคลเอนท์ ที่อยู่ในกลุ่ม ดังนั้นในแต่ละซุปเปอร์เพียร์จะต้องมีโปรโตคอลในการติดต่อสื่อสารอยู่ 2 โปรโตคอล คือโปรโตคอลในการติดต่อสื่อสารระหว่างซุปเปอร์เพียร์กับไคลเอนท์ และโปรโตคอลในการติดต่อสื่อสารระหว่างซุปเปอร์เพียร์กับซุปเปอร์เพียร์อื่น โมเดลแบบซุปเปอร์เพียร์มีจุดเด่นคือช่วยลดเวลาและแบนด์วิทที่ใช้ในการค้นหา แต่ละหน่วยจะมีความเป็นอิสระสูง สามารถควบคุมและจัดการได้ง่าย สามารถทำโหลดบาลานซิ่ง (load balancing) ได้เป็นต้น แต่โมเดลซุปเปอร์เพียร์นี้ถ้าซุปเปอร์เพียร์เสียก็จะทำให้ไคลเอนท์ที่อยู่ในกลุ่มนั้นไม่สามารถทำงานได้ แต่ปัญหานี้สามารถลดได้โดยการที่ให้มีซุปเปอร์เพียร์มากกว่าหนึ่งเพียร์ ในแต่ละกลุ่ม



รูปที่ 2.14 โมเดลซุปเปอร์เพียร์

2.5 ซ็อกเก็ต (Socket)

ซ็อกเก็ตถูกกำหนดหรือนิยามไว้ว่า เป็นคู่ของการสื่อสาร หรือคู่ของโปรเซส (หรือเธรด) โดยที่การสื่อสารบนเน็ตเวิร์คใช้คู่ของซ็อกเก็ตสำหรับแต่ละโปรเซสในซ็อกเก็ตหนึ่งประกอบไปด้วยไอพีแอดเดรสซึ่งมีขนาด 32 บิต (ในเวอร์ชั่นไอพี 4) กับหมายเลขพอร์ต (Port Number) ซึ่งจะมีขนาด 16 บิต

โดยทั่ว ๆ ไปซ็อกเก็ตใช้สถาปัตยกรรมไคลเอนท์เซิร์ฟเวอร์ เซิร์ฟเวอร์จะรอการเข้ามาตามการขอร้องของไคลเอนท์โดยการฟังที่พอร์ตเฉพาะเมื่อการขอร้องได้รับ เซิร์ฟเวอร์ก็จะยอมรับการเชื่อมต่อจากซ็อกเก็ตไคลเอนท์เพื่อให้สมบูรณ์ในการเชื่อมต่อ

เซิร์ฟเวอร์ที่สร้างการบริการเฉพาะ เช่น เทลเน็ต (telnet) เอฟทีพี (ftp) เมลล์ และเอชทีทีพี (http) จะฟัง (listen) ที่พอร์ตมีชื่อ เช่น เซิร์ฟเวอร์เทลเน็ตจะฟังที่พอร์ต 23 เซิร์ฟเวอร์เอฟทีพีจะฟังที่พอร์ต 21 หรือเซิร์ฟเวอร์เอชทีทีพีจะฟังที่พอร์ต 80 เป็นต้น

หมายเลขพอร์ตทั้งหมดที่ต่ำกว่า 1024 จะถูกพิจารณาว่าเป็นพอร์ตที่มีชื่อเสียง เราสามารถใช้พอร์ตเหล่านี้เพื่อสร้างการบริการตามมาตรฐานได้

ชนิดของซ็อกเก็ตมีอยู่ 3 ชนิดคือ

1. คอนเนกชั่นโอเรนท์เต็ดซ็อกเก็ต (Connection-Oriented Socket)
2. คอนเนกชั่นเลสซ็อกเก็ต (Connectionless Socket)
3. ลอว์ซ็อกเก็ต (Raw Socket)

คอนเนกชั่นโอเรนท์เต็ดซ็อกเก็ต

เป็นซ็อกเก็ตการเชื่อมต่อแบบต่อเนื่องที่อนุญาตให้โปรเซสเชื่อมต่อกับโปรเซสระยะไกล (Remote) ซึ่งใช้โปรโตคอลทีซีพี ดังนั้นด้วยวิธีการนี้ทำให้ข้อมูลเชื่อถือได้ เมื่อการเชื่อมต่อได้เกิดขึ้นโปรเซสก็จะมีการส่งข้อมูลกลับไปจนกระทั่งฝั่งใดฝั่งหนึ่งหรืออื่น ๆ มีการปิดการเชื่อมต่อ ชนิดของซ็อกเก็ตนี้บางครั้งเรียกว่า สตรีมซ็อกเก็ต (Stream Socket) ทั้งเอฟทีพีและเอชทีทีพีใช้ซ็อกเก็ตแบบนี้ในการสื่อสาร

คอนเนกชั่นเลสซ็อกเก็ต

หรือเรียกอีกอย่างว่า ดาต้าแกรม เป็นซ็อกเก็ตแบบไม่ต่อเนื่องและนำมาใช้เป็นประโยชน์ในการส่งเมสเสจสั้น ๆ ซึ่งไม่สามารถสนับสนุนส่วนหัว ดังนั้นจึงพิจารณาการเชื่อมต่อประเภทนี้เป็นแบบเชื่อถือไม่ได้ ซึ่งก็คือ การไม่รับประกันข้อมูลที่ถูกส่งออกไป ไม่เหมือนกับซ็อกเก็ตการเชื่อมต่อแบบต่อเนื่องที่ซ็อกเก็ตปลายทางถูกตรวจสอบเมื่อแพคเกตถูกส่งออกไป ซ็อกเก็ตแบบไม่ต่อเนื่อง เปรียบเสมือนกับการบริการของไปรษณีย์ที่ผู้ส่งจดหมายไปตามที่อยู่แล้วใส่ในกล่องรับจดหมาย ผู้ส่งจะไม่ทราบว่าผู้รับได้รับจดหมายหรือไม่ ซ็อกเก็ตแบบนี้นิยมใช้กันในเซิร์ฟเวอร์ดีเอ็นเอส (Domain Name System) ที่ใช้ซ็อกเก็ตดาต้าแกรมในการตอบสนองต่อการขอร้องที่เข้ามามาก ๆ

นอกจากนี้จะใช้คาต้าแกรมซ็อกเก็ตในการกระจาย (Broadcast) เมสเสจ หรือมัลติคาสต์ (Multicast) เพื่อไปยังปลายทางหลาย ๆ แห่งพร้อมกัน ซึ่งเหมือนกับการกระจายเสียงวิทยุหรือโทรทัศน์

ออว์ซ็อกเก็ต

เป็นซ็อกเก็ตที่อนุญาตให้การเข้าถึงโปรโตคอลทรานสปอร์ตลอว์ซ็อกเก็ตและสามารถนำมาใช้เพื่อจัดการข้อมูลส่วนหัวไอพี (IP Header) โดยไม่ใช้ระบบของวินโดวส์ (Windows) นอกจากนี้แล้วการใช้ซ็อกเก็ตชนิดนี้ ต้องการความรู้อย่างมากของโครงสร้างโปรโตคอลพื้นฐาน

2.6 มัลติมีเดีย

ในปัจจุบันมัลติมีเดีย (Multimedia) ถูกนำมาใช้เป็นสื่อเพื่อนำเสนอหรือแสดงผลงานต่าง ๆ กันอย่างแพร่หลาย เนื่องจากการนำเสนองานในรูปแบบของมัลติมีเดียจะใช้การผสมผสานกันระหว่างสื่อหลาย ๆ ประเภท เช่น ข้อความ ภาพนิ่ง และภาพเคลื่อนไหว เป็นต้น ทำให้ได้ผลงานที่น่าสนใจและดึงดูดผู้ชมได้เป็นอย่างดี

ความหมายของคำว่ามัลติมีเดีย

คำว่า มัลติ หมายถึงการนำหลาย ๆ สิ่งมาผสมรวมกัน ส่วนคำว่า มีเดีย หมายถึง ข่าวสาร หรือช่องทางการติดต่อสื่อสาร เมื่อนำทั้งสองคำมารวมกันเป็นคำว่า มัลติมีเดีย จึงหมายถึงการใช้คอมพิวเตอร์สื่อความหมายกับผู้ใช้อย่างมีปฏิสัมพันธ์เพื่อให้ผู้ใช้สามารถเรียนรู้และเข้าถึงสื่อต่าง ๆ ได้ด้วยตนเอง โดยผสมผสานองค์ประกอบต่าง ๆ เข้าไว้ด้วยกัน ได้แก่ ข้อความ ภาพนิ่ง ภาพเคลื่อนไหว และเสียง เพื่อให้บรรลุวัตถุประสงค์ที่ต้องการ แต่แท้จริงแล้วคำว่ามัลติมีเดียนั้นมีความหมายค่อนข้างกว้างขึ้นอยู่กับมุมมองของแต่ละคน

การเขียนโปรแกรมเพื่อพัฒนางานในรูปแบบของมัลติมีเดียจะต้องใช้ภาษาโปรแกรมมิ่ง ที่รองรับการทำงานกับมัลติมีเดียได้เป็นอย่างดี จึงจะสามารถพัฒนางานได้ง่ายและมีประสิทธิภาพ โดยภาษาจาวาเป็นภาษาโปรแกรมมิ่งที่รองรับการพัฒนางานในรูปแบบของมัลติมีเดียได้เป็น อย่างดี เนื่องจากภาษาจาวามีบิวต์อิน (Built-in) ของมัลติมีเดีย ที่มีประสิทธิภาพและถูกออกแบบมาให้สามารถทำงานร่วมกับระบบมัลติมีเดียได้เป็นอย่างดี

จาวาได้เตรียมเครื่องมือต่าง ๆ สำหรับใช้พัฒนา มัลติมีเดีย แอพพลิเคชั่น ไว้มากมาย โดยสามารถวาดภาพกราฟฟิกต่าง ๆ ได้ทั้งในรูปแบบ 2 มิติ และ 3 มิติได้ โดยใช้เครื่องมือของ จาวา นอกจากนี้ จาวายังได้เตรียมเครื่องมือสำหรับใช้ควบคุมการแสดงภาพ (Image) และเสียง (Audio) ไว้ด้วย

2.6.1 องค์ประกอบของมัลติมีเดีย

งานประเภทมัลติมีเดียจะต้องประกอบด้วยสื่อต่าง ๆ อย่างน้อย 2 องค์ประกอบขึ้นไป โดยสามารถแยกองค์ประกอบของมัลติมีเดียได้เป็น 5 ชนิด คือ ข้อความตัวอักษร (Text) ภาพนิ่ง (Still Image) ภาพเคลื่อนไหว (Animation) เสียง (Sound) และภาพวิดีโอหรือวิดีทัศน์ (Video) เมื่อนำองค์ประกอบต่าง ๆ มาทำงานร่วมกันเพื่อใช้สำหรับการปฏิสัมพันธ์หรือโต้ตอบระหว่างคอมพิวเตอร์ของผู้ใช้ ซึ่งถือเป็น

กิจกรรมที่ผู้ใช้สามารถเลือกกระทำกับระบบมัลติมีเดียได้ตามความต้องการ เช่น เลือกรายการในมัลติมีเดีย จึงถือเป็นส่วนที่มีความสำคัญไม่น้อยไปกว่าส่วนอื่น ดังนั้น การสร้างระบบปฏิสัมพันธ์จึงจำเป็นต้องคำนึงถึงเครื่องมือกับส่วนนี้ ระบบก็จะเชื่อมโยงไปยังส่วนข้อมูลที่เกี่ยวข้อง ซึ่งอาจเป็นไปได้ทั้ง ข้อความ ภาพเคลื่อนไหว เสียง หรือวิดีโอ ที่ได้มีการออกแบบไว้ล่วงหน้าให้พิจารณาองค์ประกอบต่าง ๆ ของมัลติมีเดีย ในส่วนของโครงงานมุ่งเน้นและสนใจเรื่องของวิดีโอหรือวิดีทัศน์เป็นหลักจึงขออธิบายความหมายในส่วนของวิดีโอหรือวิดีทัศน์เท่านั้น

2.6.2 วิดีโอหรือวิดีทัศน์ (Video)

วิดีโอหรือวิดีทัศน์ (Video) นับเป็นสื่ออีกรูปแบบหนึ่งที่นิยมใช้กับเทคโนโลยีมัลติมีเดีย เนื่องจากสามารถแสดงผลได้ทั้งภาพเคลื่อนไหวและเสียงไปพร้อมกัน โดยวิดีโอในระบบดิจิตอลสามารถนำเสนอข้อความหรือรูปภาพ (ภาพนิ่งหรือภาพเคลื่อนไหว) ประกอบเสียงได้สมบูรณ์มากกว่าองค์ประกอบชนิดอื่น ทำให้ได้งานที่น่าสนใจและมีความหลากหลายมากขึ้น อย่างไรก็ตามปัญหาหลักของการใช้วิดีโอในระบบมัลติมีเดีย คือ การใช้ทรัพยากรของหน่วยความจำเป็นจำนวนมาก เนื่องจากการแสดงวิดีโอจะต้องนำเสนอภาพในรูปแบบของเวลาจริง (Real Time) ด้วยอัตราขั้นต่ำ 30 ภาพต่อนาที หากไฟล์วิดีโอดังกล่าวไม่ได้ผ่านกระบวนการย่อขนาดหรือลดความละเอียดของภาพ การนำเสนอวิดีโอเพียง 1 นาที อาจต้องใช้พื้นที่ของหน่วยความจำมากถึง 100 เมกะบิต ส่งผลให้ขนาดของไฟล์ใหญ่เกินไป ทำให้ประสิทธิภาพในการทำงานลดลงตามไปด้วย แต่ในปัจจุบันเทคโนโลยีการลดขนาดหรือบีบอัดไฟล์วิดีโอได้รับการพัฒนาอย่างต่อเนื่อง ทำให้การนำเสนอภาพวิดีโอสิ้นเปลืองทรัพยากรของคอมพิวเตอร์น้อยลงและกลายเป็นสื่อที่มีบทบาทสำคัญต่อระบบมัลติมีเดีย (Multimedia System)

2.6.3 โปรโตคอลที่ใช้ในสตรีมมิ่งเทคโนโลยี

สิ่งที่ขาดไม่ได้ในการให้บริการมัลติมีเดียด้วยเทคโนโลยีสตรีมมิ่ง คือ โปรโตคอล ที่ใช้สื่อสาร ซึ่งโปรโตคอลนี้ออกแบบมาใหม่เพื่อให้มีความเหมาะสมในการทำงาน และยังใช้โปรโตคอลเดิมที่มีอยู่แล้วได้อีก สำหรับโปรโตคอลที่ใช้ กับการส่งข้อมูลแบบสตรีมมิ่งมีหลายตัว คือ

- เซสชั่นเดสคริปชั่นโปรโตคอล (Session Description Protocol : SDP) : เป็นโปรโตคอลที่ถูกออกแบบมาเพื่อใช้แสดงสื่อมัลติมีเดียในเซสชั่นต่าง ๆ
- เรียลไทม์ทรานสปอร์ตโปรโตคอล (Real Time Transport Protocol : RTP) : เป็นโปรโตคอลที่ใช้รูปแบบการทำงานของยูดีพีซึ่งจะเป็นการส่งข้อมูลในทิศทางเดียว แบบเซิร์ฟเวอร์ไปยังไคลเอนท์ โดยจะไม่มีการตรวจสอบความถูกต้องของข้อมูล ดังนั้นจึงสามารถส่งข้อมูลได้อย่างรวดเร็ว และได้ถูกนำมาใช้ในการส่งข้อมูลมัลติมีเดีย
- เรียลไทม์คอนโทรลโปรโตคอล (Real Time Control Protocol : RTCP) : เป็นคอนโทรลโปรโตคอลที่จะทำงานเกี่ยวข้องกับโปรโตคอลอาร์ทีพีด้วย ซึ่งอาร์ทีซีพีถูกใช้ในการควบคุมแพคเกตข้อมูลที่ถูกส่งออกไปเป็นระยะ ๆ เพื่อให้การทำงานมีประสิทธิภาพสูงสุด และตรวจสอบความถูกต้องด้วย

- ไฮเปอร์เท็กทรานสเฟอร์โปรโตคอล (Hypertext Transfer Protocol : HTTP) : เป็นโปรโตคอลที่ใช้ในการรับส่งข้อมูลไฮเปอร์เท็กซ์โดยขณะที่บราวเซอร์เรียกไปยังเว็บเซิร์ฟเวอร์ก็จะใช้โปรโตคอลนี้ในการรับส่งข้อมูลกับเซิร์ฟเวอร์ สำหรับการทำงานกับเทคโนโลยีสตรีมมิ่งนั้น โปรโตคอลเอชทีทีพีจะทำงานอยู่ในชั้นของแอพพลิเคชั่น ซึ่งถูกใช้สำหรับการติดต่อระหว่างเว็บเพจกับเว็บแอพพลิเคชั่น และเป็นโปรโตคอลที่ใช้ติดต่อผ่านทางไฟร์วอลล์ (Firewall) ด้วย
- เรียลไทม์สตรีมมิ่งโปรโตคอล (Real Time Streaming Protocol : RTSP) : เป็นโปรโตคอลที่ใช้รูปแบบไคลเอนท์เซิร์ฟเวอร์ ไคลเอนท์เซิร์ฟเวอร์ที่ถูกออกแบบเพื่อใช้ในการแสดงสื่อมัลติมีเดียสำหรับเรียลเซิร์ฟเวอร์ (Real server) เวอร์ชั่นใหม่ อาร์ทีเอสพีจะสนับสนุนชัวร์สตรีมทีเอ็ม (SureStreamTM) ซึ่งสามารถเลือกที่จะส่งข้อมูลที่อัตราความเร็วสูงสุดในขณะนั้นโดยอัตโนมัติ

2.7 ไมโครคอนโทรลเลอร์

ขนาดของตัวไอซีไมโครคอนโทรลเลอร์ก็เป็นส่วนหนึ่งของการพัฒนาใช้งาน ดั้งนั้นจึงมีการลดขนาดของไอซี ให้มีจำนวนของขาใช้งาน ให้น้อยลงเหลือเพียงขนาด 20 ขา ที่ไว้ใช้งานในกรณีที่ไม่มีความจำเป็นจะต้องใช้จำนวนพอร์ตมาก โดยจะกำหนดให้เหลือเพียง 2 พอร์ต และลดคุณสมบัติบางอย่างลงไป แต่ยังมีคุณสมบัติที่เป็นพื้นฐานเพื่อใช้งานอยู่ครบ คำสั่งและส่วนอื่น ๆ ที่สำคัญก็ยังจะคงเดิม และสามารถที่จะเลือกขนาด เนื้อที่ของหน่วยความจำโปรแกรมเป็นแบบแฟลช และคุณสมบัติส่วนอื่น ๆ เพิ่มเติมได้ โดยเลือกตามเบอร์ของไอซี เช่น ไอซีไมโครคอนโทรลเลอร์เบอร์ AT89C1051 AT89C2051 และไอซีเบอร์ AT89C4051 จะมีหน่วยความจำโปรแกรมภายในขนาด 1 กิโลไบต์ 2 กิโลไบต์ และ 4 กิโลไบต์ ตามลำดับ แสดงได้ดังรูปที่ 2.15

รูปที่ 2.15 ไอซีไมโครคอนโทรลเลอร์ของ ATMEL แบบแฟลชขนาด 20 ขา

ดังนั้นจะเห็นได้ว่าการพัฒนาเทคโนโลยีของการผลิตชิพไอซี ได้ก้าวหน้าไปอย่างรวดเร็ว จนถึงในปัจจุบันก็ยังมีไอซีไมโครคอนโทรลเลอร์อีกมากมายหลายแบบ หลายเบอร์ที่ผลิตขึ้นมา และมีคุณสมบัติอื่น ๆ ที่เพิ่มเติมในตัวแต่ไม่ได้กล่าวถึงในที่นี้ แม้กระทั่งภาษาที่จะนำมาเขียนสั่งงานให้กับไอซีไมโครคอนโทรลเลอร์เพื่อประมวลผลข้อมูลตามคำสั่งก็มีการพัฒนาตามขึ้นไปด้วย เช่น โปรแกรมที่เป็นภาษาซี ภาษาเบสิก เพื่อให้สะดวกในการ ใช้งานสำหรับผู้ที่ไม่ชอบเขียนภาษาแอสเซมบลี ดังนั้นในการที่เราจะนำไอซีไมโครคอนโทรลเลอร์ตัวใด เบอร์ไหน บริษัทใด หรือโปรแกรมแบบไหน ที่จะนำไปใช้งานจึงขึ้นอยู่กับงานที่เราต้องการจะออกแบบ และข้อมูลที่เราจะสะดวกในการค้นคว้าสร้างชิ้นงาน ที่สำคัญคือต้นทุน ในการสร้าง ส่วนข้อดีข้อเสียของแต่ละแบบก็คือการแข่งขันของแต่ละบริษัทที่จะต้องผลิตให้ดีกว่ากันแต่ที่แน่นอนเราจะต้องศึกษาให้เข้าใจและสามารถนำไปใช้งานให้ได้ก่อนเบอร์ใดเบอร์หนึ่ง

ดังนั้นในการเรียนรู้และทดลองไอซีไมโครคอนโทรลเลอร์ในนี้จะใช้งานไอซีไมโครคอนโทรลเลอร์เบอร์ AT89CX051 ของบริษัทแอทเมล (Atmel) เป็นพื้นฐาน

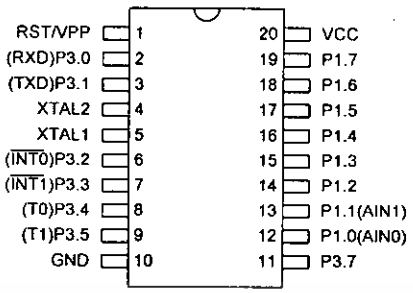
2.7.1 คุณสมบัติของไอซีไมโครคอนโทรลเลอร์ AT89CX051

- มีโครงสร้างและใช้ชุดคำสั่งเดียวกับตระกูล MCS-51 ของอินเทล
- แหล่งจ่ายไฟใช้ได้ตั้งแต่ 2.7 โวลต์ ถึง 6 โวลต์
- มีหน่วยความจำโปรแกรมชนิดแฟลชเมมโมรี (Flash Memory) ขนาด 1 กิโลไบต์ 2 กิโลไบต์ และ 4 กิโลไบต์ ตามเบอร์ที่เลือกใช้
- มีหน่วยความจำแบบแรม 8 บิต ขนาด 64 ไบต์ สำหรับ ไอซีเบอร์ AT89C1051 และ 128 ไบต์ สำหรับไอซีเบอร์ AT89C2051 และ AT89C4051
- ทำงานที่ความเร็วสัญญาณนาฬิกาได้สูงสุดถึง 24 เมกะเฮิร์ตซ์
- มีอินพุตเอาต์พุตพอร์ต ขนาด 15 บิต
- พอร์ตสามารถซิงค์ (SINK) กระแสได้ 20 มิลลิแอมป์
- มีสัญญาณการอินเตอร์รัพท์ได้ 3 แหล่งสำหรับไอซีเบอร์ AT89C1051 และ 6 แหล่งสำหรับไอซีเบอร์ AT89C2051 และไอซีเบอร์ AT89C4051
- มีพอร์ตสื่อสารแบบอนุกรม 1 ช่อง (UART)
- มีวงจรตั้งเวลาและวงจรนับขนาด 16 บิตจำนวน 1 ช่อง สำหรับ 89C1051 และ 2 ช่อง สำหรับ AT89C2051 และ AT89C4051
- สามารถโปรแกรมข้อมูล เพื่อป้องกันการอ่านเขียน หรือคัดลอกโปรแกรมได้ 2 ระดับ
- มีวงจรเปรียบเทียบสัญญาณอนาล็อก (Analog Comparator Input) 1 ช่อง
- มีระบบประหยัดพลังงาน (Low Power Idle And Power Down Model)

ตารางที่ 2.3 คุณสมบัติของไมโครคอนโทรลเลอร์แต่ละหมายเลขในตระกูล MCS-51

ชื่อหมายเลข	หน่วยความจำภายใน		จำนวนไทเมอร์/ เคาน์เตอร์	จำนวน อินเตอร์รัปท์
	เก็บโปรแกรม	เก็บข้อมูล		
8052AH	8K x 8 ROM	256 x 8 RAM	3 x 16-Bit	6
8051AH	4K x 8 ROM	128 x 8 RAM	2 x 16-Bit	5
8051	4K x 8 ROM	128 x 8 RAM	2 x 16-Bit	5
8032AH	ไม่มี	256 x 8 RAM	2 x 16-Bit	6
8031AH	ไม่มี	128 x 8 RAM	2 x 16-Bit	5
8031	ไม่มี	128 x 8 RAM	2 x 16-Bit	5
8751H	4K x 8 EPROM	128 x 8 RAM	2 x 16-Bit	5
8751H-12	4K x 8 EPROM	128 x 8 RAM	2 x 16-Bit	5

2.7.2 หน้าที่แต่ละขาของไอซีไมโครคอนโทรลเลอร์ AT89CX051



รูปที่ 2.16 ลักษณะตัวถังและตำแหน่งขาของไอซี AT89C2051

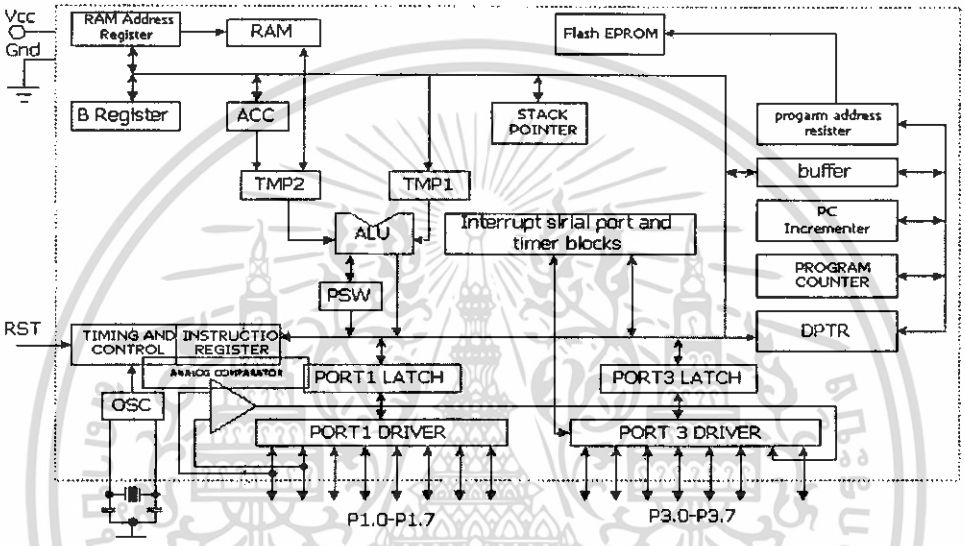
- V_{CC} เป็นขาที่ใช้สำหรับต่อไฟเพื่อเลี้ยงไอซี +5 โวลต์ดีซี
- GND เป็นขากราวด์สำหรับต่อกับกราวด์ของระบบ
- พอร์ต 1 (P1.0-P1.7) มีจำนวน 8 ขา แต่ละขาเรียกได้เป็น 1 บิต สามารถที่จะกำหนดให้เป็นได้ทั้งพอร์ตอินพุตและ พอร์ตเอาต์พุตสำหรับใช้งานทั่วไป ถ้าต้องการให้ขาพอร์ตใดเป็นอินพุตก็สามารถทำได้โดยการเขียนข้อมูล ลอจิก "1" ไปยังแต่ละบิตของพอร์ตที่ต้องการติดต่อ
- พอร์ต 3 (P3.0-P3.7) มีจำนวน 7 ขา แต่ละขาเรียกได้เป็น 1 บิต แต่ในส่วนของวงจรภายในไอซีจะมีขาของ พอร์ต 3 อยู่ทั้งหมด 8 ขา เพียงแต่ขา P3.6 จะไม่ได้ต่อออกมาใช้งานภายนอกของตัวไอซี แต่ใช้เป็นขารับสถานะของผลการเปรียบเทียบสัญญาณอนาล็อกคอมพาราเตอร์อินพุตระหว่างพอร์ต P1.0 และ P1.1 จากภายนอก ดังนั้นขาทั้ง 7 ขาที่ต่อใช้งานภายนอกของไอซีสามารถที่จะกำหนดให้เป็นได้ทั้งพอร์ตอินพุต และพอร์ตเอาต์พุต สำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ตใดเป็นอินพุต สามารถที่จะทำได้โดยการเขียนข้อมูลให้เป็นลอจิก "1" ไปยังแต่ละบิตของพอร์ตที่ต้องการจะติดต่อด้วย นอกจากนั้นขาของ พอร์ต 3 จะยังเป็นขาที่มีหน้าที่การใช้งานพิเศษ ซึ่งจะมีรายละเอียด ดังต่อไปนี้
  - P3.0 ใช้เป็นขาอินพุตสำหรับรับข้อมูลจากการสื่อสารแบบอนุกรม หรือขา RxD
  - P3.1 ใช้เป็นขาอินพุตสำหรับส่งข้อมูลจากการสื่อสารแบบอนุกรม หรือขา TxD
  - P3.2 ใช้เป็นขาอินพุตรับสัญญาณอินเตอร์รัพท์จากภายนอกช่อง 0 หรือขา INT0
  - P3.3 ใช้เป็นขาอินพุตรับสัญญาณอินเตอร์รัพท์จากภายนอกช่อง 1 หรือขา INT1
  - P3.4 ใช้เป็นขาอินพุตสำหรับรับสัญญาณไทเมอร์จากภายนอกช่อง 0 หรือขา T0
  - P3.5 ใช้เป็นขาอินพุตสำหรับรับสัญญาณไทเมอร์จากภายนอกช่อง 1 หรือขา T1
  - P3.7 ใช้เป็นขาอินพุตและเอาต์พุตสำหรับใช้งานทั่วไป P3.6 อยู่ภายในไอซีไม่ได้ต่อออกมาภายนอก แต่ใช้เป็นขารับสถานะของการเปรียบเทียบอนาล็อกคอมพาราเตอร์อินพุตระหว่างพอร์ต P1.0 และ P1.1 จากภายนอก
  - รีเซต (Reset) เป็นขาที่ใช้รับสัญญาณในการรีเซต โดยจะรีเซตระบบการทำงานของไมโครคอนโทรลเลอร์ ในการป้อนสัญญาณนั้นจะต้องทำให้สถานะ ที่ขานี้อยู่ในระดับ

ลอจิก "1" (high) อย่างน้อย 2 แมตชีนไซเคิล โดยที่วงจรกำเนิดสัญญาณนาฬิกาจะยังคงทำงานต่อเนื่องไปอย่างเป็นปกติ

- XTAL 1 และ XTAL 2 เป็นขาที่ใช้สำหรับต่อกับตัวคริสตอล เพื่อสร้างสัญญาณนาฬิกาในการกำหนดจังหวะการทำงานของไมโครคอนโทรลเลอร์

2.7.3 โครงสร้างของไอซีไมโครคอนโทรลเลอร์

โครงสร้างภายในของไอซีเบอร์ AT89CX051



รูปที่ 2.17 สถาปัตยกรรมภายในของไมโครคอนโทรลเลอร์เบอร์ AT89CX051

จากรูปที่ 2.17 แสดงให้เห็นสถาปัตยกรรมภายในของไมโครคอนโทรลเลอร์เบอร์ AT89CX051 ซึ่งภายในจะประกอบไปด้วยหน่วยการทำงานต่าง ๆ โดยแบ่งการทำงานออกเป็นบล็อก ๆ ซึ่งประกอบไปด้วยวงจรควบคุมรีจิสเตอร์ต่าง ๆ (Register) หน่วยความจำข้อมูล (RAM) และหน่วยความจำโปรแกรมที่เป็นแฟลช วงจรออสซิลเลเตอร์ (Oscillator) ส่วนที่ทำหน้าที่ทางคณิตศาสตร์และลอจิก (ALU: Arithmetic and logic Unit) โปรแกรมเคาน์เตอร์ (Program Counter) และพอร์ตที่ใช้ติดต่อกับอุปกรณ์ภายนอก ซึ่งในแต่ละส่วนจะถูกเชื่อมต่อกันด้วยบัสข้อมูลและบัสแอดเดรส

2.7.4 พอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 มีวงจรสื่อสารอนุกรมแบบฟูลดูเพล็กซ์ 1 ชุด (วงจรสื่อสารแบบฟูลดูเพล็กซ์ หมายถึงวงจรสื่อสารที่สามารถทำการรับและส่งข้อมูลในลักษณะ 2 ทิศทาง ได้ในเวลาเดียวกัน) โดยใช้ขาสัญญาณของพอร์ต 3 คือ ขา P3.0 เป็นขารับข้อมูลเข้าหรือหรือ RxD และขา P3.1 เป็นขาส่งข้อมูลออกหรือ TxD โดยวงจรสื่อสารข้อมูลแบบอนุกรมของไมโครคอนโทรลเลอร์ตระกูล MCS-51 แบบแฟลชเป็นแบบอะซิงโครนัส ปกติแล้วพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51

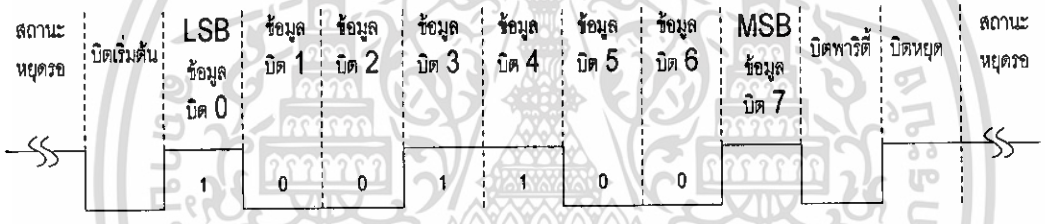
จะใช้ในการติดต่อสื่อสารกับพอร์ตอนุกรมของคอมพิวเตอร์ โดยใช้มาตรฐาน RS-232 แต่ในปัจจุบันสามารถติดต่อกันในมาตรฐาน RS-232 หรือ RS-485 ได้แล้ว โดยใช้ไอซีพิเศษที่ทำหน้าที่ในการแปลงสัญญาณการสื่อสารดังกล่าว

การสื่อสารข้อมูลแบบอะซิงโครนัส

การสื่อสารข้อมูลแบบอะซิงโครนัสคือการรับและส่งข้อมูลโดยไม่จำเป็นต้องมีสัญญาณนาฬิการ่วมด้วย แต่จะใช้การกำหนดอัตราเร็วในการรับและส่งข้อมูลให้มีค่าเท่ากัน เรียกอัตราเร็วนี้ว่า อัตราบอด หรือบอดเรต (Baud rate) มีหน่วยเป็นบิตต่อวินาที (bit per second : bps)

รูปแบบของข้อมูลที่ใช้ในการรับส่งแบบอะซิงโครนัสประกอบด้วย 4 ส่วนด้วยกันคือ

- บิตเริ่มต้น (start bit) มีขนาด 1 บิต
- บิตข้อมูลแบบอนุกรม มีขนาด 8 บิต
- บิตตรวจสอบพาริตี้ (parity bit) มีขนาด 1 บิตหรือไม่มี
- บิตปิดท้ายหรือบิตหยุด (stop bit) มีขนาด 1 บิต



รูปที่ 2.18 แสดงรูปแบบข้อมูลอนุกรมแบบอะซิงโครนัส

รูปที่ 2.18 แสดงรูปแบบของข้อมูลอนุกรมแบบอะซิงโครนัส เมื่อไม่มีการส่งข้อมูลขา DATA จะมีสถานะลอจิก "1" เรียกสถานะนี้ว่า สถานะหยุดรอ (waiting stage) การเริ่มต้นส่งข้อมูลจะเริ่มจากการใช้ขา DATA มีลอจิก "0" ด้วยช่วงระยะเวลา 1 บิต เรียกบิตนี้ว่า บิตเริ่มต้น (start bit) จากนั้นบิตข้อมูลจะถูกส่งออกไปโดยเริ่มจากบิตที่มีนัยสำคัญต่ำสุดหรือบิต LSB ก่อน ข้อมูลที่ต้องการส่งมีจำนวน 8 บิต ตามด้วยบิตพาริตี้ (parity bit) ซึ่งใช้ในการตรวจสอบความผิดพลาดที่เกิดขึ้นจากการส่งข้อมูล บิตสุดท้ายที่จะส่งคือบิตปิดท้ายหรือบิตหยุด (stop bit) โดยจะเป็นการทำให้ขา DATA มีสถานะลอจิก "1" อีกครั้งด้วยระยะเวลาอย่างน้อย 1 บิต 1.5 บิต หรือ 2 บิต เพื่อเป็นการแสดงว่าสิ้นสุดข้อมูลแล้วอัตราความเร็วในการรับและส่งข้อมูลของการรับส่งข้อมูลแบบอะซิงโครนัสหรืออัตราบอดหรือบอดเรตที่ใช้สำหรับพอร์ตอนุกรม RS – 232 มีด้วยกันหลายค่า ได้แก่ 110 150 300 600 1200 2400 4800 9600 และ 19200 บิตต่อวินาที โดยมีค่าเพิ่มมากขึ้นตามเทคโนโลยีของคอมพิวเตอร์ เนื่องจากอัตราบอดคือค่าของจำนวนบิตที่สามารถส่งได้ใน 1 วินาที สมมติว่า ข้อมูลอนุกรมมีขนาด 8 บิต ไม่มีการตรวจสอบพาริตี้มีบิตเริ่มต้น 1 บิต และบิตปิดท้าย 1 บิต ความยาวของข้อมูล 1 ไบต์จะเท่ากับความยาว 10 บิต ถ้าใช้บอดเรตในการส่งข้อมูลเท่ากับ 9,600 บิตต่อวินาที ก็สามารถรับส่งข้อมูลได้ด้วยความเร็ว 960 ไบต์ต่อวินาที

การตรวจสอบพาริตี้สามารถกำหนดให้เป็นแบบคี่ (odd) แบบคู่ (even) หรือไม่มีการตรวจสอบพาริตี้ก็ได้ การตรวจสอบแบบพาริตี้คี่ (หรือคู่) คือการนับจำนวนลอจิก "1" ของข้อมูลขนาด 1 ไบท์ และของบิตพาริตี้ว่ามีจำนวนรวมเป็นเลขคี่ (หรือคู่) หรือไม่ ถ้าใช่แสดงว่าข้อมูลที่ส่งมาถูกต้อง ยกตัวอย่างข้อมูลที่จะทำการส่งมีขนาด 8 บิต มีค่าเท่ากับ 99 ฐานสิบหก หรือ 10011001 ฐานสอง จะเห็นได้ว่าข้อมูลในไบต์มีจำนวนลอจิก "1" จำนวน 4 ตัว ซึ่งเป็นเลขคู่ ดังนั้นถ้ากำหนดการตรวจสอบค่าพาริตี้เป็นแบบคู่ ค่าของบิตพาริตี้จะต้องมีลอจิกเป็น "0" แต่ถ้ากำหนดการตรวจสอบพาริตี้เป็นแบบคี่ ค่าของบิตพาริตี้จะต้องเป็น "1" เพื่อให้จำนวนลอจิก "1" ข้อมูล 1 ไบต์ รวมทั้งบิตพาริตี้มีจำนวนบิตที่เป็นลอจิก "1" มีจำนวนรวมกันเป็นเลขคี่ ตารางที่ 2.4 แสดงตัวอย่างของบิตพาริตี้ในการรับส่งข้อมูลอนุกรม

ตารางที่ 2.4 แสดงบิตพาริตี้ของข้อมูล

ข้อมูล	บิตพาริตี้คู่	บิตพาริตี้คี่
00000000	0	1
00000001	1	0
00000010	1	0
00000011	0	1
00000100	1	0
11111110	1	0
11111111	0	1

บิตพาริตี้จะถูกสร้างขึ้นจากภาคส่งข้อมูลของ UART (Universal Asynchronous Receiver Transmitter: เป็นอุปกรณ์ที่ใช้ในการรับและส่งข้อมูลอนุกรม) ซึ่งทางภาครับจะต้องกำหนดคุณสมบัติการตรวจสอบพาริตี้ที่ตรงกันเอาไว้ว่าตรวจสอบพาริตี้คี่หรือพาริตี้คู่ จากนั้นภาครับของ UART จะทำการตรวจสอบพาริตี้ที่เกิดขึ้นว่าเป็นคู่หรือเป็นคี่ โดยการนับจำนวนลอจิก "1" ทั้งหมดรวมทั้งบิตพาริตี้ด้วย ถ้ากำหนดพาริตี้ไว้เป็นคู่แต่อ่านค่าตัวเลขในการนับออกมาได้ตัวเลขเป็นคี่ ทางภาครับจะแสดงข้อผิดพลาดออกมาให้ผู้ใช้ทราบ กระบวนการดังกล่าวเป็นวิธีการตรวจสอบความผิดพลาดที่เกิดขึ้นในการรับส่งข้อมูลที่ง่ายที่สุด แต่มันสามารถตรวจสอบได้เมื่อมีบิตข้อมูลที่ทำการรับส่งผิดพลาดเพียงบิตเดียวเท่านั้น ถ้าข้อมูลที่ทำการส่งมีบิตที่มีการผิดพลาดมากกว่า 1 บิต การตรวจสอบด้วยวิธีนี้จะไม่ได้ผล สำหรับการตั้งพาริตี้บิตเป็น NONE นั้นทั้งภาครับและภาคส่ง จะไม่มีการตรวจสอบพาริตี้

คอมพิวเตอร์ในรุ่น AT เกือบทั้งหมดจะใช้ UART หมายเลข 16450 และ 16550 ส่วนคอมพิวเตอร์ในรุ่น XT ใช้ UART หมายเลข 8250 UART ชิปเหล่านี้มีระดับแรงดันเป็นแบบทีทีแอล (0 และ +5 โวลต์) แต่เพื่อให้มีแรงดันเป็นไปตามมาตรฐาน RS-232 และเพื่อให้การรับส่งข้อมูลสามารถทำได้ที่ระยะทางไกลมากขึ้นระดับแรงดันทีทีแอลจะถูกแปลงเป็นระดับแรงดันที่สูงขึ้น โดยลอจิก "0" มีระดับแรงดัน +3 โวลต์ ถึง +12 โวลต์ ในขณะที่ลอจิก "1" มีระดับแรงดัน -3 โวลต์ จนถึง -12 โวลต์

รีจิสเตอร์ที่เกี่ยวข้องกับการทำงานของพอร์ตอนุกรมใน MCS-51

ในการทำงานของวงจรพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS-51 มีรีจิสเตอร์ที่ต้องเกี่ยวข้องอยู่ 2 ตัว มีรายละเอียดดังต่อไปนี้

1. รีจิสเตอร์บัฟเฟอร์ของพอร์ตอนุกรมหรือ SBUF (Serial Data Buffer Register) มีแอดเดรสอยู่ที่ 99H ในพื้นที่ของรีจิสเตอร์ฟังก์ชั่นพิเศษหรือ SFR มีขนาด 8 บิต แบ่งเป็น 2 ส่วนคือ รีจิสเตอร์บัฟเฟอร์สำหรับส่งข้อมูล (transmit buffer register) และรีจิสเตอร์บัฟเฟอร์สำหรับรับข้อมูล (receive buffer register) เมื่อมีการเขียนข้อมูลมายังรีจิสเตอร์ SBUF ข้อมูลนั้นจะถูกส่งต่อไปยังบัฟเฟอร์สำหรับส่งข้อมูล เพื่อส่งออกจากไมโครคอนโทรลเลอร์ผ่านทางขา TxD หรือ P3.1 ในกรณีที่มีการอ่านข้อมูลจากรีจิสเตอร์ SBUF ข้อมูลจะถูกส่งผ่านไปยังรีจิสเตอร์บัฟเฟอร์สำหรับรับข้อมูลเพื่อส่งต่อไปยังไมโครคอนโทรลเลอร์ต่อไป สำหรับการรับข้อมูลอนุกรมจากภายนอกนั้นจะผ่านมาทาง RxD หรือ P3.0 ของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

2. รีจิสเตอร์ควบคุมการทำงานของพอร์ตอนุกรมหรือ SCON (Serial Prot Control Register) เป็นรีจิสเตอร์ขนาด 8 บิต มีแอดเดรสอยู่ที่ 98H ในพื้นที่ของรีจิสเตอร์ SFR สามารถเข้าถึงได้ในระดับบิต มีรายละเอียดการทำงานดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
SM0	SM1	SM2	REN	TB8	RB8	TI	RI

รูปที่ 2.19 แสดงรีจิสเตอร์ควบคุมการทำงานของพอร์ตอนุกรม

SM0 – SM1: ใช้ในการเลือกโหมดการทำงานของพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์

SM2: ใช้ในการอีนาเบิลการสื่อสารในแบบมัลติโปรเซสเซอร์ (Multiprocessor) ในการทำงานของโหมด 2 และ 3 ของพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS-51 ในโหมด 2 และ 3 ถ้าบิตนี้เป็น “1” บิต RI จะไม่แอกตีฟถ้าบิตที่ 9 ที่รับเข้ามาเป็น “0” (ข้อมูลบิตที่ 9 เก็บไว้ที่บิต RB8) ในการทำงานโหมด 1 ถ้าบิตนี้ถูกเซตบิต RI จะไม่แอกตีฟถ้ายังไม่รับบิตหยุด ส่วนในโหมด 0 บิตนี้ไม่มีการใช้งาน

REN: ใช้ในการเอ็นเอเบิลการรับข้อมูลพอร์ตอนุกรม ทำการเซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์ ถ้าต้องการรับข้อมูลต้องเซตบิตนี้ให้เป็น “1”

TB8: ใช้สำหรับเก็บข้อมูลที่บิต 9 ที่ต้องการส่งออกไปขณะทำงานในโหมด 2 และ 3 ของพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS-51 เซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์

RB8: ใช้สำหรับข้อมูลบิตที่ 9 ที่เข้ามาขณะทำงานในโหมด 2 และ 3 ของพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS-51 แต่ถ้าหากพอร์ตอนุกรมทำงานอยู่ในโหมด 1 และบิต SM2 เป็น “0” ข้อมูลที่บิต RB8 คือ ข้อมูลของบิตหยุด (stop bit) สำหรับการทำงานใน

โหมด 0 บิตนี้จะไม่ใช้งาน บิต RB8 นี้สามารถเซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์

TI: ใช้ในการแสดงการเกิดอินเตอร์รัปต์เมื่อมีการส่งข้อมูลออกจากพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 สามารถเซตได้ด้วยกระบวนการทางฮาร์ดแวร์ เมื่อทำการส่งข้อมูลบิตที่ 8 เรียบร้อยแล้วในการทำงานโหมด 0 ส่วนการทำงานโหมดอื่น บิตนี้จะเซตเมื่อมีการเริ่มต้นส่งบิตหยุดออกไป การเคลียร์บิตนี้ต้องใช้กระบวนการทางซอฟต์แวร์เท่านั้น

RI: ใช้ในการแสดงการเกิดอินเตอร์รัปต์เมื่อมีการรับข้อมูลเข้าจากพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 สามารถเซตได้ด้วยกระบวนการทางฮาร์ดแวร์ เมื่อทำการรับข้อมูลบิตที่ 8 เรียบร้อยแล้วในการทำงานโหมด 0 ส่วนการทำงานโหมดอื่นบิตนี้จะเซตเมื่อมีการรับบิตหยุดของข้อมูลอนุกรมไปได้ครึ่งทางแล้ว ยกเว้นกรณีที่บิต SM2 มีการเซต บิตนี้จะเซตได้ก็ต่อเมื่อการรับบิตหยุดหรือบิตที่ 9 เกิดขึ้นอย่างสมบูรณ์แล้ว การเคลียร์บิตนี้ต้องใช้กระบวนการทางซอฟต์แวร์เท่านั้น

โหมดการทำงานของพอร์ตอนุกรมใน MCS-51

พอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชสามารถเลือกโหมดการทำงานได้ถึง 4 โหมดคือ

1.โหมด 0 เป็นการกำหนดให้พอร์ตอนุกรมทำงานในลักษณะชิฟต์รีจิสเตอร์

2.โหมด 1 เป็นการกำหนดให้เป็น UART ขนาด 8 บิต สามารถเลือกอัตราบอดได้

3.โหมด 2 เป็นการกำหนดให้เป็น UARTขนาด 9 บิต โดยมีอัตราบอดคงที่

4.โหมด 3 เป็นการกำหนดให้เป็น UART ขนาด 9 บิต สามารถเลือกอัตราบอดได้

การเลือกโหมดการทำงานของวงจรพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS-51 กระทำได้โดยการกำหนดข้อมูลให้แก่บิต SMO และ SMI ในรีจิสเตอร์ SCON

1. การทำงานในโหมด 0 ของวงจรพอร์ตอนุกรม (โหมดชิงโครนัส)

ข้อมูลอนุกรมจะผ่านเข้าและออกทางขา RxD ส่วนขา TxD ทำหน้าที่เป็นสัญญาณนาฬิกาของการเลื่อนข้อมูล (shift clock) ในโหมดนี้มีจำนวนข้อมูล 8 บิต โดยทำการรับและส่งข้อมูลในบิต LSB ก่อน อัตราในการรับส่งข้อมูลหรืออัตราบอดถูกกำหนดไว้คงที่ที่ 1/12 ของความถี่สัญญาณนาฬิกา

เริ่มต้นการส่งข้อมูลด้วยการเขียนข้อมูลที่ต้องการส่งมายังรีจิสเตอร์ SBUF สัญญาณเขียนข้อมูล SBUF แอกตีฟเป็น "1" ที่สเตต 6 เฟส 2 (S6P2) ของแมตชีนไซเคิล ส่งมายังวงจรควบคุมการส่ง (TX control) ทำให้วงจรควบคุมเริ่มต้นส่งข้อมูล สัญญาณส่ง (SEND) จะแอกตีฟเป็น "1" ตลอดการส่งข้อมูล

ข้อมูลจากรีจิสเตอร์ SBUF จะถูกเลื่อนออกที่ขา P3.0 หรือขา RxD ครั้งละบิต ตามจังหวะของสัญญาณนาฬิกาที่ส่งออกมาทางขา P3.1 หรือ TxD โดยสัญญาณนาฬิกาของการเลื่อนข้อมูลจะมีขอบขาลงของสัญญาณที่สเตต 3 เฟส และมีขอบขาขึ้นของสัญญาณที่สเตต 6 เฟส 1 ของแต่ละแมตชีนไซเคิลในกระบวนการส่งข้อมูล จนกระทั่งเมื่อส่งข้อมูลครบ 8 บิตแล้ว บิต TI ในรีจิสเตอร์ SCON จะเกิดการเซต

เป็นการแจ้งให้ทราบว่าส่งข้อมูลครบแล้ว หากการอินเตอร์รัปต์จากพอร์ตอนุกรมได้รับการเอ็นเอเบิลไว้ ก็จะเกิดการอินเตอร์รัปต์ขึ้นในระบบ เมื่อเสร็จสิ้นกระบวนการรับข้อมูล สัญญาณส่ง (SEND) จะกลายเป็น "0" จนกว่าจะเริ่มต้นกระบวนการรับข้อมูลใหม่

ในกระบวนการรับข้อมูลเริ่มต้นด้วยการเซต REN ให้เป็น "1" และเคลียร์บิต RI ในรีจิสเตอร์ SCON ก่อนที่สเตต 6 เฟส 2 ของแมตชีนไซเคิลถัดไป วงจรควบคุมการรับ (RX control) จะทำการเขียนข้อมูล 11111110 ไปยังชิฟต์รีจิสเตอร์สำหรับรับข้อมูลและทำการแอกตีฟสัญญาณรับ (RECEIVER) ให้เป็น "1" ในสัญญาณนาฬิกาลูกถัดไป

เมื่อสัญญาณรับ (RECEIVER) แอกตีฟ ก็จะเกิดการส่งสัญญาณนาฬิกาของการเลื่อนข้อมูล (Shift clock) ขึ้นผ่านทางขา P3.1 หรือ TxD เพื่อทำการกำหนดจังหวะการรับข้อมูลครั้งละบิต โดยสัญญาณนาฬิกานี้จะเกิดขึ้นในช่วงสเตต 3 เฟส 1 ถึงสเตต 6 เฟส 1 ของแต่ละแมตชีนไซเคิล การรับข้อมูลเข้ามาทางขา P3.0 หรือ RxD จะเกิดขึ้นที่สเตต 5 เฟส 2 ในแมตชีนไซเกิลเดียวกับสัญญาณนาฬิกาของการเลื่อนข้อมูล จนกระทั่งรับข้อมูลครบทั้ง 8 บิต บิต RI จะได้รับการเซตเพื่อแจ้งการเสร็จสิ้นกระบวนการรับข้อมูล หากการอินเตอร์รัปต์จากพอร์ตอนุกรมได้รับการเอ็นเอเบิลไว้ก็จะเกิดการอินเตอร์รัปต์ขึ้นในระบบ เมื่อเสร็จสิ้นกระบวนการรับข้อมูล สัญญาณรับ จะกลายเป็น "0" จนกว่าจะเริ่มต้นกระบวนการรับข้อมูลใหม่

การทำงานในโหมดนี้ของพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS-51 จะใช้ในการเชื่อมต่อกับไอซีรีจิสเตอร์ภายนอกเพื่อทำการขยายจำนวนพอร์ตอินพุตหรือเอาต์พุต แต่ไม่เป็นที่นิยมใช้มากนัก เนื่องจากในไมโครคอนโทรลเลอร์ MCS-51 เองมีพอร์ตอยู่ค่อนข้างมาก และติดต่อกับพอร์ตเหล่านั้นได้ง่ายและเร็วกว่ามาก

2. การทำงานในโหมด 1 ของวงจรพอร์ตอนุกรม

โหมดนี้ใช้ในการรับส่งข้อมูลรวม 10 บิต โดยส่งข้อมูลออกทางขา P3.1 หรือ TxD และรับข้อมูลเข้าทางขา P3.0 หรือ RxD ข้อมูลทั้ง 10 บิตประกอบด้วย บิตเริ่มต้น (มีค่าเป็น "0") 1 บิต บิตข้อมูล 8 บิต โดยรับหรือส่งข้อมูลในบิต LSB ก่อน และบิตหยุดหรือบิตปิดท้าย (มีค่าเป็น "1") ในการรับข้อมูลบิตหยุดจะถูกเก็บไว้ในบิต RB8 ในรีจิสเตอร์ SCON อัตราบอดในโหมดนี้ได้รับการกำหนดโดยอัตราการเกิดโอเวอร์โฟลว์ของไทเมอร์ 1 ใน AT89C51 ส่วนในไมโครคอนโทรลเลอร์เบอร์ AT89C52 และในอนุกรม AT89Sxx สามารถเลือกใช้อัตราการเกิดโอเวอร์โฟลว์ของไทเมอร์ 1 หรือไทเมอร์ 2 ในการกำหนดอัตราบอดได้

กระบวนการส่งข้อมูลเริ่มต้นด้วยการแอกตีฟสัญญาณเขียนข้อมูลมายังรีจิสเตอร์ SBUF ส่งมายังวงจรควบคุมการส่ง (TX control) จากนั้นวงจรควบคุมจะทำการแอกตีฟสัญญาณส่ง (SEND) ที่สเตต 1 เฟส 1 ของแมตชีนไซเกิลต่อมา โดยสัญญาณส่ง จะเป็น "0" ตลอดการส่งข้อมูลเมื่อสัญญาณส่งแอกตีฟจะทำการส่งบิตเริ่มต้นก่อนเป็นบิตแรก โดยมีคาบเวลาของบิตเริ่มต้นเท่ากับ 1 แมตชีนไซเคิล จากนั้นตามด้วยการส่งบิตข้อมูล 8 บิต เรียงลำดับจากบิต LSB โดยข้อมูลที่ทำการส่งถูกเรียกออกมาจากรีจิสเตอร์บัฟเฟอร์สำหรับการส่งข้อมูล ในทุก ๆ บิตข้อมูลที่ทำการส่งออกไป จะเกิดสัญญาณพัลส์ชิฟต์ (PULSE SHIFT) ขึ้น เพื่อให้เรียกข้อมูลในแต่ละบิตจากรีจิสเตอร์บัฟเฟอร์ การกำหนดจังหวะการส่งข้อมูลใช้สัญญาณนาฬิกาการส่ง (TX control) เป็นตัวกำหนด โดยสัญญาณนาฬิกานี้ได้มาจากการหารสัญญาณ

TCLK จากไทเมอร์ 1 ด้วย 16 หลังจากการส่งบิตข้อมูลก็จะทำการส่งบิตหยุดหรือบิตปิดท้าย 1 บิต ดังนั้น การส่งข้อมูลจะใช้สัญญาณนาฬิกาทั้งหมด 10 ลูกเมื่อทำการส่งข้อมูลครบเรียบร้อยแล้ว จะทำการเซต บิต TI ในรีจิสเตอร์ SCON หากการอินเตอร์รัปต์จากพอร์ตอนุกรมได้รับการเอ็นเอเบิลไว้ ก็จะเกิดการอินเตอร์รัปต์ขึ้นในระบบ หลังจากที่ทำการบริการอินเตอร์รัปต์หรือส่งข้อมูลเรียบร้อยแล้วต้องทำการเคลียร์บิต TI ก่อนเป็นอันดับแรก เพื่อให้การรับส่งข้อมูลทางพอร์ตอนุกรมดำเนินต่อไปได้

ด้านการรับข้อมูลจะทำการตรวจจับการเปลี่ยนแปลงระดับลอจิกจาก "1" เป็น "0" ที่ขา RxD โดยใช้อัตราการสุ่มเท่ากับ 1/16 เท่าของอัตราบอด เมื่อตรวจจับพบ ไทเมอร์/เคาน์เตอร์ที่ใช้ในการกำหนดอัตราบอดจะรีเซตและทำการเขียนข้อมูล 1FFH ไปยังชิฟต์รีจิสเตอร์ ข้อมูลจะเริ่มเดินทางเข้าสู่พอร์ตอนุกรมของไมโครคอนโทรลเลอร์ผ่านทางขา RxD ในการตีความว่าบิตที่เข้ามาเป็น "0" หรือ "1" จะใช้ผลการสุ่มข้างมาก โดยบิตของข้อมูลที่เข้ามาได้รับการแบ่งออกเป็น 16 สเตต การสุ่มข้อมูลจะทำการสุ่มสเตตที่ 7 8 และ 9 หาก 2 ใน 3 ของการสุ่มพบว่าข้อมูลเป็นลอจิกใด จะตีความข้อมูลในบิตนั้นเป็นตามเสียงข้างมาก ยกตัวอย่าง สุ่มพบลอจิก "1" 2 ใน 3 ครั้ง จะตีความว่าบิตของข้อมูลที่ได้รับนั้นเป็น "1"

ลำดับของการรับข้อมูลมีลักษณะเดียวกับการส่งข้อมูลคือ เริ่มด้วยบิตเริ่มต้นก่อนตามด้วยบิตข้อมูล และบิตปิดท้าย ในทุก ๆ การรับข้อมูลได้ 1 บิต จะมีพัลส์ชิฟต์เกิดขึ้น เพื่อทำการเลื่อนข้อมูลเข้าสู่รีจิสเตอร์บัฟเฟอร์การรับข้อมูล การกำหนดจังหวะการรับข้อมูลใช้สัญญาณนาฬิกาการรับข้อมูล (RX clock) หลังจากสัญญาณนาฬิกาลูกสุดท้าย อันหมายถึงสามารถรับข้อมูลได้ครบแล้ว วงจรควบคุมการรับข้อมูลจะทำการส่งข้อมูลจากรีจิสเตอร์บัฟเฟอร์ไปยังรีจิสเตอร์ SBUF และบิต RB8 ในรีจิสเตอร์ SCON โดยข้อมูลในบิต RB8 ก็คือข้อมูลของบิตหยุดนั่นเอง พร้อมกันนั้นยังทำการเซตบิต RI ในรีจิสเตอร์ SCON ด้วย หากการอินเตอร์รัปต์จากพอร์ตอนุกรมได้รับการเอ็นเอเบิลไว้ ก็จะเกิดการอินเตอร์รัปต์ขึ้นในระบบ หลังจากบริการอินเตอร์รัปต์หรือรับข้อมูลเรียบร้อยแล้ว ต้องทำการเคลียร์บิต RI ก่อน เพื่อให้การรับส่งข้อมูลทางพอร์ตอนุกรมดำเนินต่อไปได้

การทำงานในโหมดนี้ได้รับความนิยมสูงสุดเนื่องจากมีกระบวนการที่ไม่ซับซ้อนและสามารถรับส่งข้อมูลกับคอมพิวเตอร์ได้อย่างมีประสิทธิภาพ

3. การทำงานในโหมด 2 และ 3 ของวงจรพอร์ตอนุกรม

ในทั้งสองโหมดนี้จะใช้รูปแบบข้อมูลรวม 11 บิต ประกอบด้วยบิตเริ่มต้นมีค่าเป็น "0" จำนวน 1 บิต บิตข้อมูล 8 บิต โดยทำการรับและส่งบิต LSB ก่อน บิตข้อมูลบิตที่ 9 และบิตปิดท้ายมีค่าเป็น "1" จำนวน 1 บิตในการส่งข้อมูล ข้อมูลบิตที่ 9 จะได้รับการเก็บไว้ในบิต TB8 ในรีจิสเตอร์ SCON และในการรับข้อมูล ข้อมูลบิตที่ 9 จะนำไปเก็บไว้ในบิต RB8 ในรีจิสเตอร์ SCON สำรับอัตราบอดในโหมด 2 จะคงที่โดยเลือกได้ 2 ค่าคือ 1/32 หรือ 1/64 ของความถี่สัญญาณนาฬิกา สำหรับในโหมด 3 อัตราบอดสามารถปรับได้เหมือนกับในโหมด 1

อัตราบอดของพอร์ตอนุกรมในไมโครคอนโทรลเลอร์ MCS-51

1. โหมด0

อัตราบอดของโหมด 0 = ความถี่ โดยสามารถคำนวณได้จากสูตร

อัตราบอดในโหมด 0 = ความถี่ของสัญญาณนาฬิกา/12 มีหน่วยเป็นบิตต่อวินาที

2.โหมด 1 และ 3

ทั้งสองโหมดนี้สามารถเลือกแหล่งกำเนิดอัตราบอดได้ 2 แหล่งคือ จากอัตราโอเวอร์โฟลว์ของไทเมอร์ 1 และ 2 สำหรับอัตราบอดเมื่อใช้การโอเวอร์โฟลว์ของไทเมอร์ 1 จะต้องใช้ค่าของบิต SMOD ในรีจิสเตอร์ PCON มาพิจารณาประกอบด้วย โดยสามารถหาค่าอัตราบอดได้จาก

อัตราบอด = (2 ค่าของบิตSMOD/32) x อัตราโอเวอร์โฟลว์ของไทเมอร์ 1

ตารางที่ 2.5 แสดงการเลือกอัตราบอดของวงจรพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์ MCS-51

อัตราบอด (บิตต่อวินาที : bps)	ความถี่สัญญาณนาฬิกา	SMOD	ไทเมอร์ 1		
			C/T	โหมด	ค่ารีโหลด
โหมด 0 : สูงสุด 1 MHz	12 MHz	x	x	x	x
โหมด 2 : สูงสุด 375 k	12 MHz	1	x	x	x
โหมด 1,3 : 62.5 k	12 MHz	1	0	2	FFH
19.2 k (19,200)	11.0592 MHz	1	0	2	FDH
9.6 k (9,600)	11.0592 MHz	0	0	2	FDH
4.8 k (4,800)	11.0592 MHz	0	0	2	FAH
2.4 k (2,400)	11.0592 MHz	0	0	2	F4H
1.2 k (1,200)	11.0592 MHz	0	0	2	E8H
137.5	11.0592 MHz	0	0	2	1DH
110	6 MHz	0	0	2	72H
110	12 MHz	0	0	1	FEEBH

ถ้าหากในไทเมอร์ 1 ไม่ได้เอ็นเอเบิลการอินเตอร์รัปต์ไว้ สามารถคำนวณหาค่าอัตราบอดได้จาก

อัตราบอด = (2 ค่าของบิตSMOD/32) x (ความถี่สัญญาณนาฬิกา/{12x[256-(TH1)]})

กรณีที่ใช้ไทเมอร์ 2 ในการกำหนดอัตราบอด โดยกำหนดให้ไทเมอร์ 2 ทำงานในโหมดกำเนิดอัตราบอด (buad rate generator) สามารถคำนวณหาค่าอัตราบอดได้จาก

อัตราบอด = อัตราโอเวอร์โฟลว์ของไทเมอร์ 2/16 มีหน่วยเป็นบิตต่อวินาที

ถ้าหากกำหนดให้ไทเมอร์ 2 ทำงานในโหมดไทเมอร์หรือเคาน์เตอร์ตามปกติ สามารถคำนวณหาค่าอัตราบอดได้จาก

อัตราบอด = ความถี่สัญญาณนาฬิกา/(32x(65536-(RCAP2H RCAP2L)))

โดยที่ (RCAP2H RCAP2L) เป็นค่าของรีจิสเตอร์ RCAP2H และ RCAP2L มีขนาด16 บิตไม่คิดเครื่องหมาย

3.โหมด 2

ในโหมดนี้การกำหนดอัตราบอดจะขึ้นอยู่กับค่าของบิต SMOD ในรีจิสเตอร์ PCON ถ้า SMOD เป็น "0" อัตราบอดจะเท่ากับ 1-64 ของความถี่สัญญาณนาฬิกา ในกรณีที่ SMOD เป็น "1" อัตราบอดจะเท่ากับ 1-32 ของความถี่สัญญาณนาฬิกา สามารถแสดงเป็นสูตรคำนวณทางคณิตศาสตร์ได้ดังนี้

อัตราบอด = (2 ค่าของบิตSMOD/64) x ความถี่สัญญาณนาฬิกา

การกำหนดค่าของไทเมอร์เพื่อเลือกอัตราบอด

ในการใช้งานพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 สิ่งที่ต้องให้ความสนใจมากที่สุดประการหนึ่งคือ อัตราการถ่ายทอดข้อมูล หรือ อัตราบอด ซึ่งการกำหนดอัตราบอดนั้นจะขึ้นอยู่กับค่าความถี่ของสัญญาณนาฬิกาเป็นหลัก สำหรับโหมดการทำงานของพอร์ตอนุกรมที่สามารถเลือกอัตราบอดได้อย่างอิสระคือโหมด 1 และ 3 โดยกำหนดได้จากอัตราการเกิดโอเวอร์โฟลว์ของไทเมอร์ 1 หาก ไทเมอร์ 1 มีการเกิดโอเวอร์โฟลว์ในอัตราที่สูงมากอัตราบอดก็จะมีค่าสูงมากขึ้นตาม นั่นหมายความว่าอัตราในการทำงานถ่ายทอดข้อมูลจะสูงมาก สามารถถ่ายทอดข้อมูลได้อย่างรวดเร็ว ดังนั้นการกำหนดค่าและโหมดการทำงานของไทเมอร์ 1 จึงเป็นสิ่งที่มีผลต่อการเปลี่ยนแปลงของอัตราบอดด้วย

ในการใช้ไทเมอร์ 1 เพื่อกำหนดอัตราบอดในโหมด 1 และ 3 ของพอร์ตอนุกรมจะต้องกำหนดให้ไทเมอร์ ทำงานในโหมด 2 หรือโหมด 8 บิต แบบตั้งค่าการนับอัตโนมัติ ในกรณีการกำหนดค่ารีโหลดให้แก่รีจิสเตอร์ TH1 จะเป็นตัวแปรหลักที่ใช้ในการกำหนดอัตราบอดให้แก่พอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51

เริ่มต้นด้วยการเคลียร์บิต SMOD ซึ่งเป็นบิต 7 ของรีจิสเตอร์ PCON ให้เป็น "0" ค่าของการรีโหลดให้แก่ TH1 สามารถคำนวณได้จาก

TH1 = 256-((ค่าความถี่ของคริสตอล-384)/อัตราบอด)

แต่ถ้าบิต SMOD เกิดจากการเซต จะเป็นการเอ็นเอเบิลการทวีคูณของอัตราบอด ดังนั้นการกำหนดค่าให้แก่ TH1 จึงต้องคำนวณจาก

TH1 =256-((ค่าความจริงของคริสตอล/192)/อัตราบอด)

ยกตัวอย่าง ถ้าหากในไมโครคอนโทรลเลอร์ AT89C51 ใช้คริสตอล 11.0592 MHz ต้องการกำหนดอัตราบอดของพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ไว้ที่ 19200 บิตต่อวินาที ในกรณีที่ไม่เอ็นเอเบิลการทวีคูณของอัตราบอด ค่ารีโหลดของไมโครคอนโทรลเลอร์จะเท่ากับ

TH1 = 256-((ค่าความถี่ของคริสตอล/384)/อัตราบอด

= 256-((11059200/384)/19200)

= 256-(28800/19200)

= 256-1.5 = 254.5

เนื่องจากผลลัพธ์ที่ได้เป็นค่าที่ไม่ใช่จำนวนเต็ม ถ้าหากกำหนดค่าของ TH1 เป็น 254 เมื่อทำการแทนค่าเพื่อคำนวณหาค่าอัตราบอด จะได้อัตราบอดเท่ากับ 14400 บิตต่อวินาที และถ้าหากกำหนดค่าของ TH1 เป็น 255 อัตราบอดจะมีค่าเท่ากับ 28800 บิตต่อวินาที ดังนั้นจะเห็นได้ว่าค่าของ TH1 ที่ไม่เป็น

จำนวนเต็มจะไม่สามารถทำให้เกิดอัตราบอดตามต้องการได้

ทางแก้ไขคือให้ทำการเอ็นเอเบิลการทวีคูณอัตราบอด โดยการเซตบิต SMOD ในรีจิสเตอร์ PCON ให้เป็น "1" จากนั้นทำการแทนค่าลงในสมการหาค่า TH1 เมื่อมีการเซตบิต SMOD ได้ผลดังนี้

TH1 = 256-((ค่าความถี่ของคริสตอล/192)/อัตราบอด)

= 256-((11059200/192)/19200)

= 256-(57600/19200)

= 256-3 = 253

นำค่าของ TH1 ที่ได้ทำการแทนค่าคำนวณหาค่าอัตราบอดจะได้เท่ากับ 19200 บิตต่อวินาที สามารถสรุปขั้นตอนในการเลือกอัตราบอดโดยการกำหนดค่าของไทเมอร์ 1 ได้ดังนี้

1. กำหนดให้พอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 ทำงานในโหมด 1 หรือ 3

2. กำหนดให้ไทเมอร์ 1 ทำงานในโหมด 2 หรือโหมด 8 บิตตั้งค่าอัตโนมัติ

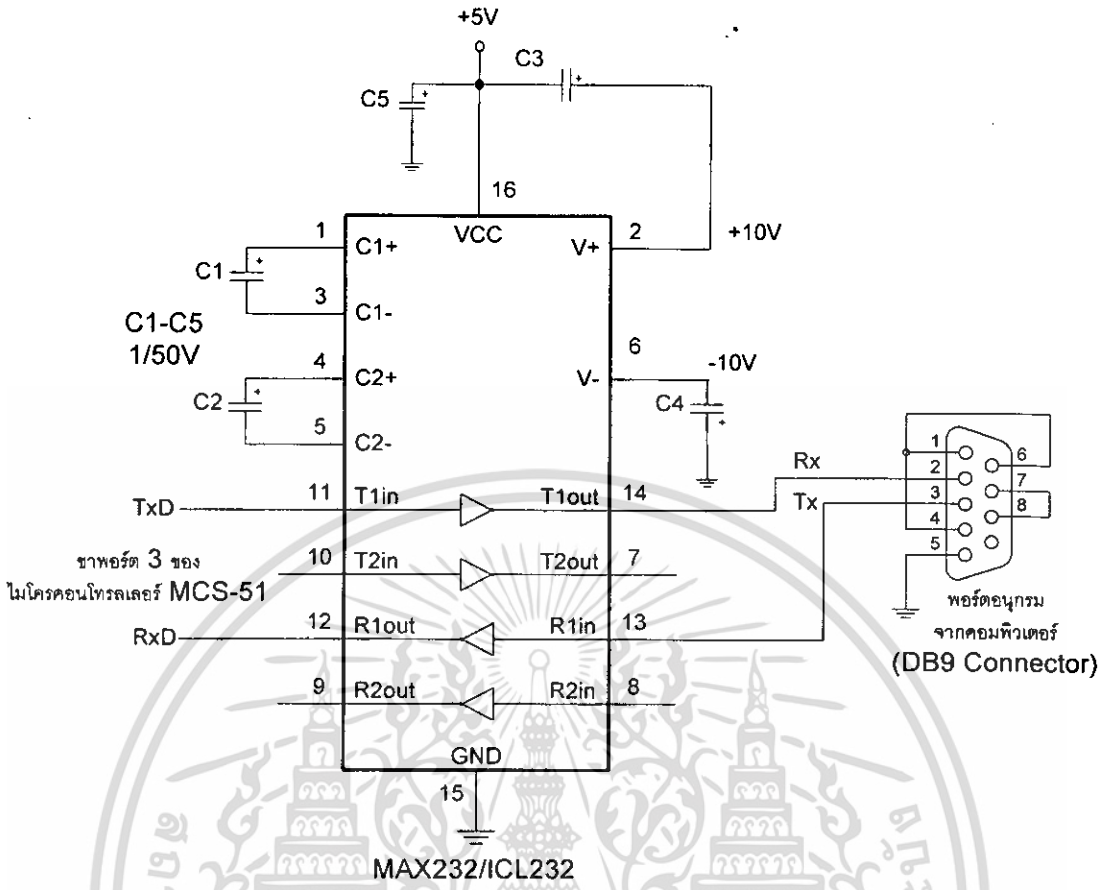
3. กำหนดข้อมูลให้แก่ TH1 เท่ากับ 253 เพื่อให้สามารถกำเนิดอัตราบอดได้ 19,200 บิตต่อวินาที ตามที่ต้องการ

4. ทำการเซตบิต SMOD ซึ่งเป็นบิต 7 ของรีจิสเตอร์ PCON เพื่อเอ็นเอเบิลการทวีคูณอัตราบอด

การเชื่อมต่อกับพอร์ตอนุกรมของคอมพิวเตอร์

การใช้งานวงจรพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 มักนิยมใช้ในการติดต่อเพื่อแลกเปลี่ยนข้อมูลกับคอมพิวเตอร์ ผ่านทางพอร์ตอนุกรมในมาตรฐาน RS-232 เป็นส่วนใหญ่ แต่เนื่องจากระดับสัญญาณของพอร์ตอนุกรม RS-232 มีระดับตั้งแต่ 3 ถึง 12 โวลต์ ในขณะที่ระดับสัญญาณของไมโครคอนโทรลเลอร์MCS-51 อยู่ในระดับทีทีแอล ดังนั้นจึงไม่สามารถเชื่อมต่อพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 เข้ากับพอร์ตอนุกรมของคอมพิวเตอร์ได้โดยตรงจึงต้องอาศัยการเชื่อมต่อผ่านไอซีพิเศษทำหน้าที่แปลงระดับสัญญาณ

ไอซีที่ทำหน้าที่ในการแปลงสัญญาณนี้จะต้องทำการแปลงข้อมูลส่งของไมโครคอนโทรลเลอร์ MCS-51 จากระดับทีทีแอลไปเป็นระดับของ RS-232 และทำการแปลงข้อมูลที่รับจากคอมพิวเตอร์จากระดับ RS-232 เป็นระดับทีทีแอลเพื่อให้สามารถถ่ายทอดไปยังไมโครคอนโทรลเลอร์ MCS-51 ได้อย่างสมบูรณ์ ไอซีดังกล่าวมีด้วยกันหลายเบอร์จากผู้ผลิต อาทิ MAX232 จาก MAXIX หรือ ICL232 จาก HARRIS เป็นต้น ในรูปที่ 2.20 แสดงส่วนวงจรของการต่อกับพอร์ตอนุกรมของคอมพิวเตอร์และไมโครคอนโทรลเลอร์



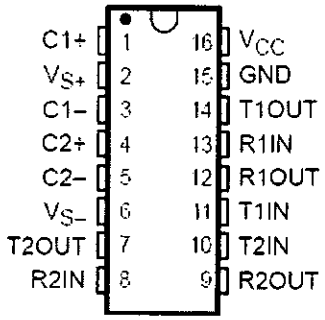
รูปที่ 2.20 แสดงวงจรเชื่อมต่อ MAX232 เข้ากับพอร์ตอนุกรมของคอมพิวเตอร์และไมโครคอนโทรลเลอร์

2.8 ไอซี MAX 232

เป็นไอซีที่ทำหน้าที่เปลี่ยนแรงดันที่เข้ามาจากพอร์ตอนุกรมไปเป็นแรงดันตามมาตรฐาน RS-232 โดยเปลี่ยนระดับแรงดัน TTL เพื่อทำให้ใช้ได้กับไมโครคอนโทรลเลอร์

2.8.1 การสื่อสารพอร์ตอนุกรม RS-232

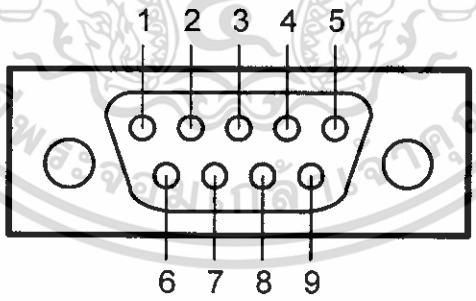
ลักษณะของการส่งข้อมูลแบบอนุกรมนั้น ข้อมูลที่ส่งออกมาทีละบิตจากตัวส่งไปตัวรับข้อมูล ช่วงสัญญาณในการส่งข้อมูลอาจใช้เพียง 1 หรือ 2 ช่องสัญญาณเท่านั้น ทำให้ค่าใช้จ่ายในการสื่อสารจะถูกกว่าแบบขนาน ในการส่งข้อมูลแบบอนุกรม ข้อมูลที่ต้องการส่งจะอยู่ในลักษณะเป็นไบต์จะทยอยส่งทีละบิต และทางตัวรับจะต้องรับข้อมูลเข้ามาทีละบิต แล้วมารับกันเป็นไบต์ซึ่งทางตัวรับต้องคอยตรวจสอบว่าบิตใดเป็นบิตเริ่มต้นหรือบิตสุดท้ายของข้อมูล การตรวจสอบนั้นจะขึ้นอยู่กับรูปแบบของรหัสของบิตข้อมูลที่ใช้ ซึ่งในการรับส่งข้อมูลแบบอนุกรมระหว่างไมโครคอมพิวเตอร์กับอุปกรณ์ภายนอกนั้นจำเป็นต้องมีมาตรฐานในการรับส่งข้อมูล ซึ่งมาตรฐานที่นิยมมากที่สุดคือ มาตรฐาน RS-232



รูปที่ 2.21 ไอซี MAX 232

2.8.2 มาตรฐาน RS-232

เพื่อที่จะทำให้อุปกรณ์จากผู้ผลิตต่างกันทำงานร่วมกันได้ มาตรฐานหลายชนิดจึงได้รับการออกแบบขึ้น มาตรฐานที่ใช้กันอย่างกว้างขวางมากที่สุดคือ RS-232C ซึ่งโดยปกติไมโครคอมพิวเตอร์จะมีพอร์ตที่เป็นแบบอนุกรมอยู่ในตัวแล้ว ตามจุดประสงค์ของมาตรฐาน RS-232C นั้นเพื่อจะสามารถเชื่อมต่อกันระหว่างอุปกรณ์รับส่งปลายทาง (DATA TERMINAL EQUIPMENT: DTE) เช่น พอร์ตของคอมพิวเตอร์หลักหรืออุปกรณ์ปลายทางกับอุปกรณ์ RS-232 เป็นข้อกำหนดของการอินเตอร์เฟสมาตรฐาน และสามารถใช้เพื่อจุดประสงค์อื่น ๆ ต่าง ๆ กันไป เช่น การสื่อสารแบบซิงโครนัส (Synchronous Communication) และรูปแบบการสื่อสารที่ต้องการสัญญาณนาฬิกา และสัญญาณกำหนดจังหวะเพิ่มเติมขึ้นมาในความเป็นจริงแล้วเราสามารถทำให้มีการสนทนาระหว่าง DTE และ DCE โดยการใช้สายสัญญาณเพียง 3 เส้นเท่านั้น คือใช้สาย TD สาย RD และสายกราวด์เท่านั้น



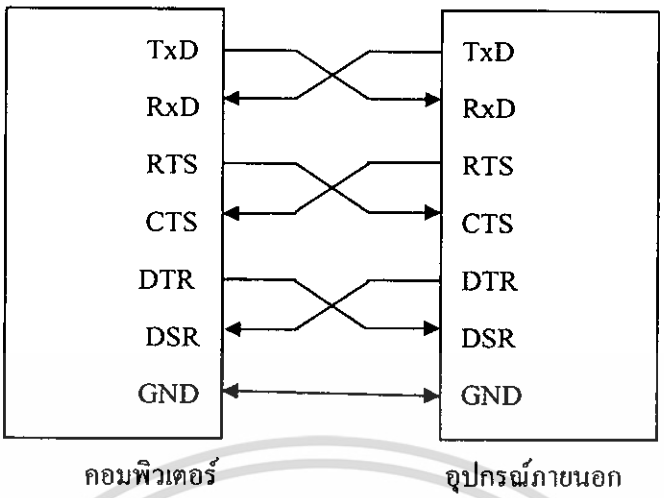
รูปที่ 2.22 คอนเน็กเตอร์ 9 ขาหรือแบบ DB-9 (มองจากด้านหลังคอมพิวเตอร์)

ตารางที่ 2.6 แสดงรายละเอียดการต่อคอนเน็กเตอร์แบบ DB9 มาตรฐาน RS-232

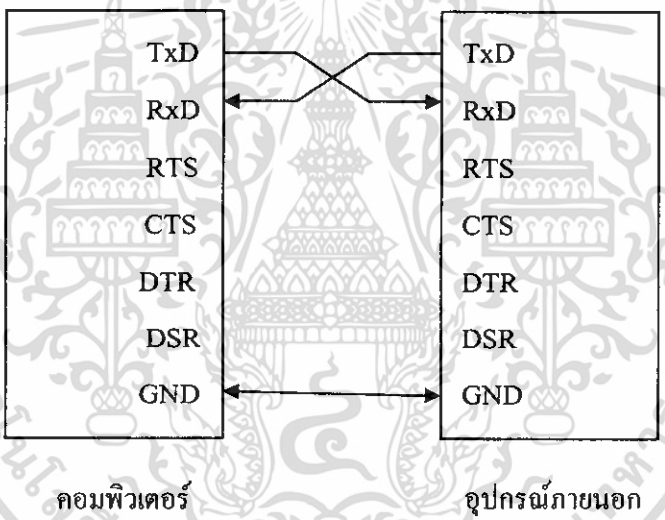
หมายเลขสัญญาณ	ชื่อของสายสัญญาณ
1	Data Carrier
2	Received Data
3	Transmitted Data
4	Data Terminal Ready
5	Signal Common
6	Data Set Ready
7	Request To Send
8	Clear To Send
9	Ring Indicator

ขั้นตอนการติดต่อระหว่างอุปกรณ์ DTE และ DCE

1. เมื่อจ่ายกำลังงานให้กับ DTE และอุปกรณ์ก็จะส่งสัญญาณ DTR ออกมา
2. อุปกรณ์ DCE ถูกเปิดขึ้นและรับรู้สัญญาณ DTR ที่ส่งมาจากอุปกรณ์ DTE
3. อุปกรณ์ DCE ส่งสัญญาณ DSR ออกมา และโมเด็มก็กระทำกระบวนการ OFF HOOK
4. ถ้าสายสัญญาณอยู่ในสภาพดีและปลายทางอีกด้านหนึ่งก็พร้อมจะรับข้อมูลแล้ว จะมีการตรวจจับสัญญาณพาหะแล้วอุปกรณ์ DCE ส่งสัญญาณ DCD ออกมา
5. อุปกรณ์ DCE จะตอบสนองด้วยการ ส่งสัญญาณ CTS ออกมา
6. การติดต่อสื่อสารก็เริ่มขึ้น โปรแกรมควบคุมจะทำการส่งหรือรับข้อมูล



(ก) การต่ออุปกรณ์ภายนอกเข้ากับคอมพิวเตอร์ แบบ Null modem



(ข) การต่ออุปกรณ์ภายนอกเข้ากับคอมพิวเตอร์ แบบ RS-232 โดยใช้สายสัญญาณเพียง 3 เส้น

รูปที่ 2.23 การต่ออุปกรณ์ภายนอกกับพอร์ตอนุกรมของคอมพิวเตอร์ในลักษณะต่าง ๆ

ส่วนลำดับขั้นในการตอบรับเป็นดังต่อไปนี้

1. อุปกรณ์ DTE จะส่งสัญญาณ DTR ออกมา
2. อุปกรณ์ DCE จะอยู่ในโหมดตอบรับอัตโนมัติ (auto answer mode) โดยมีสัญญาณออกมา
3. สถานีปลายทางส่งสัญญาณเรียกอุปกรณ์ DCE และอุปกรณ์ DCE ส่งสัญญาณ RI ออกมา
4. อุปกรณ์ DTE รับรู้ถึงอุปกรณ์ RI ที่ส่งมาจากเครื่องปลายทาง และอุปกรณ์ DCE ก็เข้าสู่สภาวะ OFF HOOK
5. อุปกรณ์ DCE ทำการแลกเปลี่ยนข้อมูลกับอุปกรณ์ DCE ที่อีกปลายทางหนึ่ง และมีการส่งสัญญาณ DCD ออกมา

6. อุปกรณ์ DTE จะส่งสัญญาณ RTS ออกมาหรืออาจจะรอข้อมูลก็ได้ขึ้นอยู่กับโปรแกรมควบคุม

7. อุปกรณ์ DCE จะตอบสนองด้วยการส่งสัญญาณ DTS กลับออกมา

8. การติดต่อสื่อสารก็จะเริ่มขึ้น

2.9 หลักการสื่อสารข้อมูลผ่านพอร์ตอนุกรม

การสื่อสารข้อมูลผ่านพอร์ตอนุกรมในไมโครคอนโทรลเลอร์นั้น ทำได้ 2 วิธี คือ

1. ใช้อินเตอร์รัปต์

เป็นวิธีที่ให้ผลการทำงานเร็วที่สุด แต่มีความยุ่งยากในการทำงานมากกว่าเนื่องจากตำแหน่งของการอินเตอร์รัปต์ ทั้งการรับและการส่งข้อมูลนั้นอยู่ที่ตำแหน่งเดียวกัน ต้องพิจารณาจากแฟลก TI หรือ RI ก่อนว่าเกิดการอินเตอร์รัปต์จากสาเหตุใด และต้องพิจารณาการใช้รีจิสเตอร์ในช่วงเวลานั้น ๆ ด้วยว่ามีโอกาสซ้อนทับกันหรือไม่ ทำให้โปรแกรมของการทำงานในส่วนนี้มีความซับซ้อนมากกว่า

2. วนโปรแกรมตรวจสอบแฟลก

เป็นวิธีที่มีความซับซ้อนน้อยกว่า โดยเขียนโปรแกรมให้วนตรวจสอบแฟลกอยู่ตลอดเวลา จนกว่าจะเกิดการเปลี่ยนแปลง ยกตัวอย่าง เมื่อต้องการตรวจสอบการส่งข้อมูลให้ทำการวนตรวจสอบแฟลก TI ว่าถูกเซตหรือไม่ เมื่อถูกเซต แสดงว่า มีการส่งข้อมูลเกิดขึ้นเรียบร้อยแล้ว จากนั้นให้ทำการเคลียร์แฟลก TI แล้วทำการส่งข้อมูลตัวถัดไป หรือทำงานในคำสั่งต่อไปได้

ในกรณีที่ต้องการตรวจสอบการรับข้อมูลให้ทำการตรวจสอบแฟลก RI ว่าถูกเซตหรือไม่ เมื่อตรวจสอบได้ว่าถูกเซต แสดงว่า เกิดการรับข้อมูลขึ้น ให้ทำการเคลียร์แฟลก RI แล้วนำค่าจากรีจิสเตอร์ SBUF มาใช้ได้ทันที แต่วิธีการนี้มีข้อเสียตรงที่เป็นการทำงานแบบเรียงลำดับ ทำให้ขั้นตอนในการทำงานช้ากว่าการใช้อินเตอร์รัปต์

หัวใจสำคัญของการสื่อสารข้อมูลผ่านพอร์ตอนุกรมคือ การกำหนดอัตราบอดและรูปแบบของข้อมูลว่า มีจำนวนบิตเริ่มต้น บิตของข้อมูล บิตหยุด หรือว่ามีการตรวจสอบบิตพาริตี้หรือไม่ ถ้าหากข้อกำหนดเหล่านี้ในตัวส่งและตัวรับไม่ตรงกัน จะทำให้การถ่ายทอดข้อมูลเกิดความผิดพลาดได้อย่างง่ายดาย ส่งผลให้การสื่อสารข้อมูลล้มเหลวอย่างสิ้นเชิง

บทที่ 3
หลักการคำนวณและการสร้าง

ระบบควบคุมอุปกรณ์ไฟฟ้าและสังเกตการณ์สำหรับห้องเรียนเป็นระบบที่ทำการตรวจสอบสถานะว่าได้มีบุคลากรภายในห้อง และยังทำการตรวจสอบว่าภายในห้องได้มีการทำการเปิดอุปกรณ์ไฟฟ้าอยู่หรือไม่และชนิดใดบ้าง โดยใช้กล้องเว็บแคม (Webcam) ในการแสดงภาพซึ่งสามารถดูได้ว่ามีบุคลากรอยู่ภายในห้องเรียนหรือไม่และใช้โปรแกรมแสดงสถานะการจ่ายไฟที่เครื่องคอมพิวเตอร์ภายในเครือข่ายเดียวกัน ในการตรวจสอบสถานะของอุปกรณ์ไฟฟ้าว่าทำงานอยู่หรือไม่ เพื่อที่จะทำการแจ้งสถานะดังกล่าวเพื่อให้เครื่องคอมพิวเตอร์ไคลเอนท์รับทราบข้อมูลในการรับส่งข้อมูลโดยข้อมูลจะถูกส่งให้เครื่องคอมพิวเตอร์ไคลเอนท์ผ่านทางเครือข่ายแลน (LAN) เมื่อเครื่องคอมพิวเตอร์ภายในห้องเรียนได้รับข้อมูล ก็จะนำข้อมูลนั้นเพื่อใช้ประกอบในการตัดสินใจ คือ ถ้ากรณีที่ไม่มีบุคลากรใช้งานอยู่แต่มีการเปิดใช้ไฟฟ้าโดยเครื่องคอมพิวเตอร์ไคลเอนท์สามารถที่จะทำการสั่งปิดอุปกรณ์ไฟฟ้าตามที่ต้องการได้ ในบทนี้จะแบ่งการคำนวณและการสร้างออกเป็น 3 ส่วน คือ

3.1 การออกแบบการติดต่อระหว่างคอมพิวเตอร์กับคอมพิวเตอร์

ใช้ทดสอบการทำงานส่งข้อมูลผ่านซ็อกเก็ตโดยทำการจำลองการรอรับการเชื่อมต่อจากอีกฝ่ายหนึ่งมาแล้วทำการตอบกลับไป เราต้องสร้างเซิร์ฟเวอร์ซ็อกเก็ตขึ้นก่อน โดยเราจะจำลองเป็นเซิร์ฟเวอร์ซ็อกเก็ตไว้ที่พอร์ตใดก่อนก็ได้ จากนั้นสร้างซ็อกเก็ตที่ใช้ในการเชื่อมต่อโดยการรอรับจากอีกฝั่งหนึ่งเพื่อที่จะทำการส่งข้อมูลไปยังอีกฝั่งหนึ่ง เราต้องใช้เอาต์พุตสตรีมในการส่งข้อมูล ก็จะทำการพริ้นท์ผ่านซ็อกเก็ต ที่เราสร้างขึ้น ซึ่งก็คือส่งสั่งให้ส่งข้อมูลนั้นเอง เมื่อทำการส่งข้อมูลเรียบร้อยก็จะต้องทำการปิดช่องที่ใช้ทำการเชื่อมต่อเพื่อจบการจำลอง

3.2 การออกแบบการใช้งานของระบบควบคุมอุปกรณ์ไฟฟ้าและสังเกตการณ์สำหรับห้องเรียน

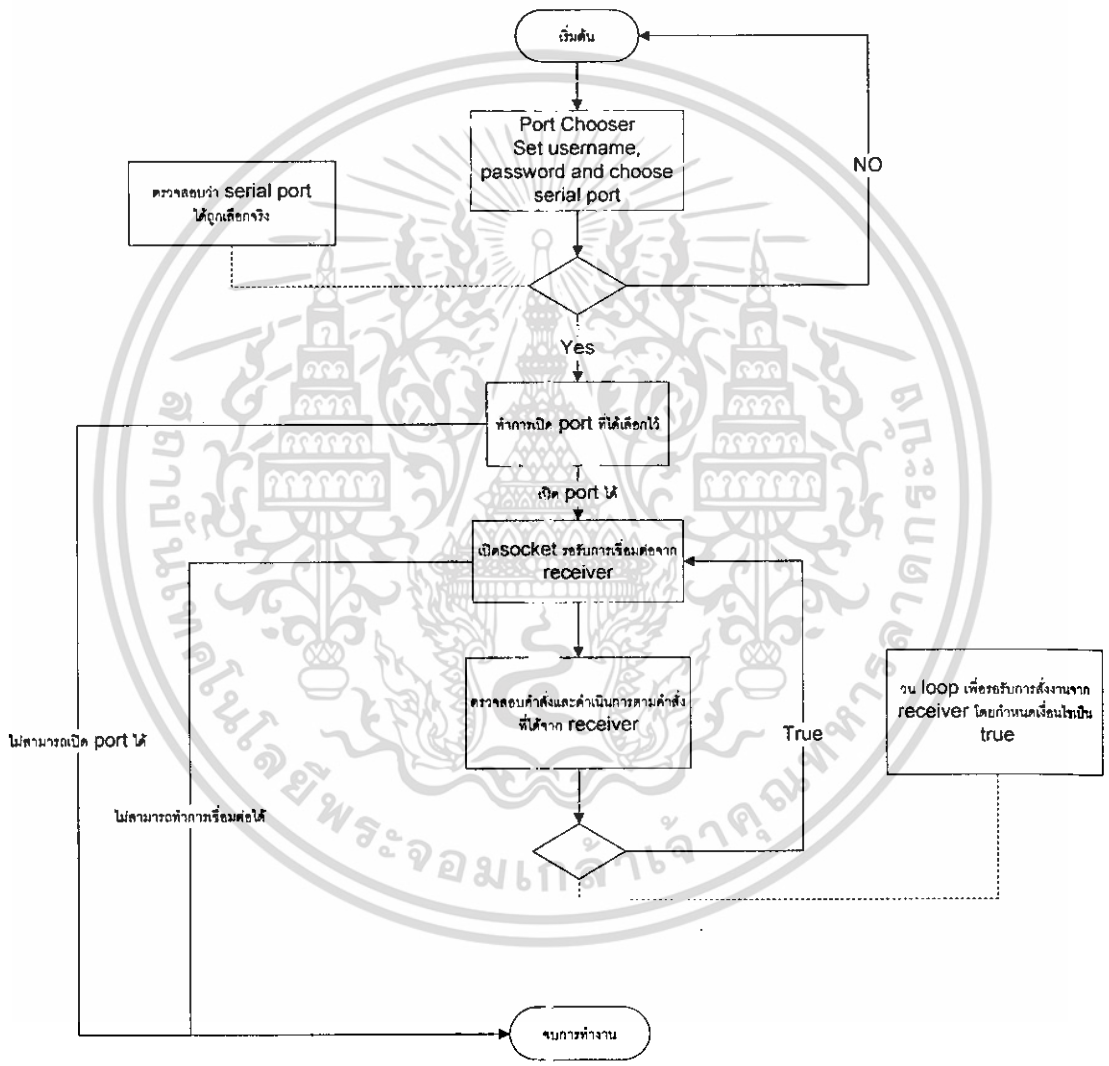
โดยการทำงานจะแบ่งเป็น 2 ส่วน คือ ส่วนของการควบคุม และส่วนของการรับภาพ โดยส่วนควบคุมนั้นระบบจะเริ่มต้นจากการที่เครื่องคอมพิวเตอร์ภายในห้องเรียนทำการตั้งค่า Username Password และพอร์ตอนุกรมของคอมพิวเตอร์ที่ให้ทำการเชื่อมต่อไปยังส่วนควบคุมอุปกรณ์ไฟฟ้า แล้วผู้ใช้บริการทำการใส่ไอพีแอดเดรสของเครื่องคอมพิวเตอร์ที่ต้องการติดต่อ Username และ Password เพื่อทำการร้องขอการใช้บริการ เครื่องคอมพิวเตอร์ภายในห้องเรียนจะทำการตรวจสอบรหัสผ่านแล้วส่งคำสั่งอนุญาตให้เครื่องคอมพิวเตอร์ที่อื่นสามารถทำการเข้าสู่หน้าต่างการควบคุมได้ หน้าต่างที่ใช้แสดงการควบคุมจะมีลักษณะ สามารถสังเกตการณ์ภายในห้องเรียนได้จากกล้อง และสามารถสั่งงานการควบคุมอุปกรณ์ไฟฟ้าได้ เราใช้โปรแกรมจาวา ส่งค่าเปิดและปิดให้กับส่วนควบคุมได้ซึ่งเป็นการจบการทำงาน อีกส่วนหนึ่งคือการรับภาพจากกล้อง การรับภาพนี้เราจะร้องขอไปยังเครื่องคอมพิวเตอร์ภายในห้องเรียน

ส่วนการทำงานนั้นจะเริ่มจากเครื่องคอมพิวเตอร์ภายในห้องเรียนจากนั้นจะเข้าสู่หน้าต่างการตรวจสอบรหัสผ่านเมื่อทำการส่งค่ารหัสผ่านให้แก่เครื่องคอมพิวเตอร์ภายในห้องเรียนทำการตรวจสอบ

แล้วถ้าถูกต้องเครื่องคอมพิวเตอร์ภายในห้องเรียนจะอนุญาตให้ทำการรับชมภาพที่ส่งมาจากกล้องพร้อมสามารถทำการควบคุมอุปกรณ์ไฟฟ้าได้ โดยการส่งค่าที่กำหนดไว้ผ่านพอร์ตอนุกรมให้ไมโครคอนโทรลเลอร์ประมวลผลต่อไป

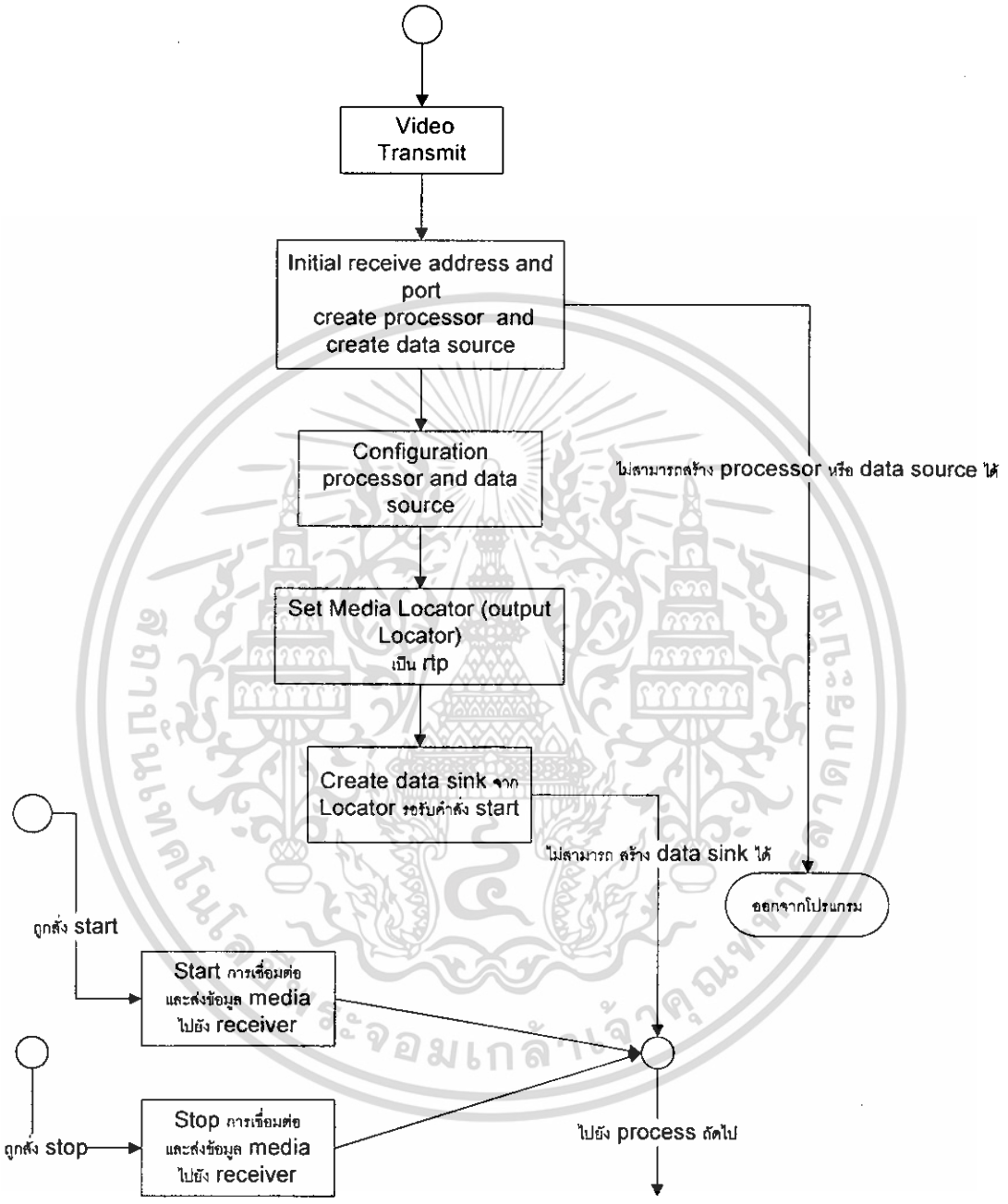
3.2.1 ส่วนโปรแกรมฝั่งเครื่องคอมพิวเตอร์ภายในห้องเรียน

ส่วนโปรแกรมฝั่งเครื่องคอมพิวเตอร์ภายในห้องเรียนเพื่อเปิดโปรแกรมควบคุมโปรแกรมย่อยต่าง ๆ



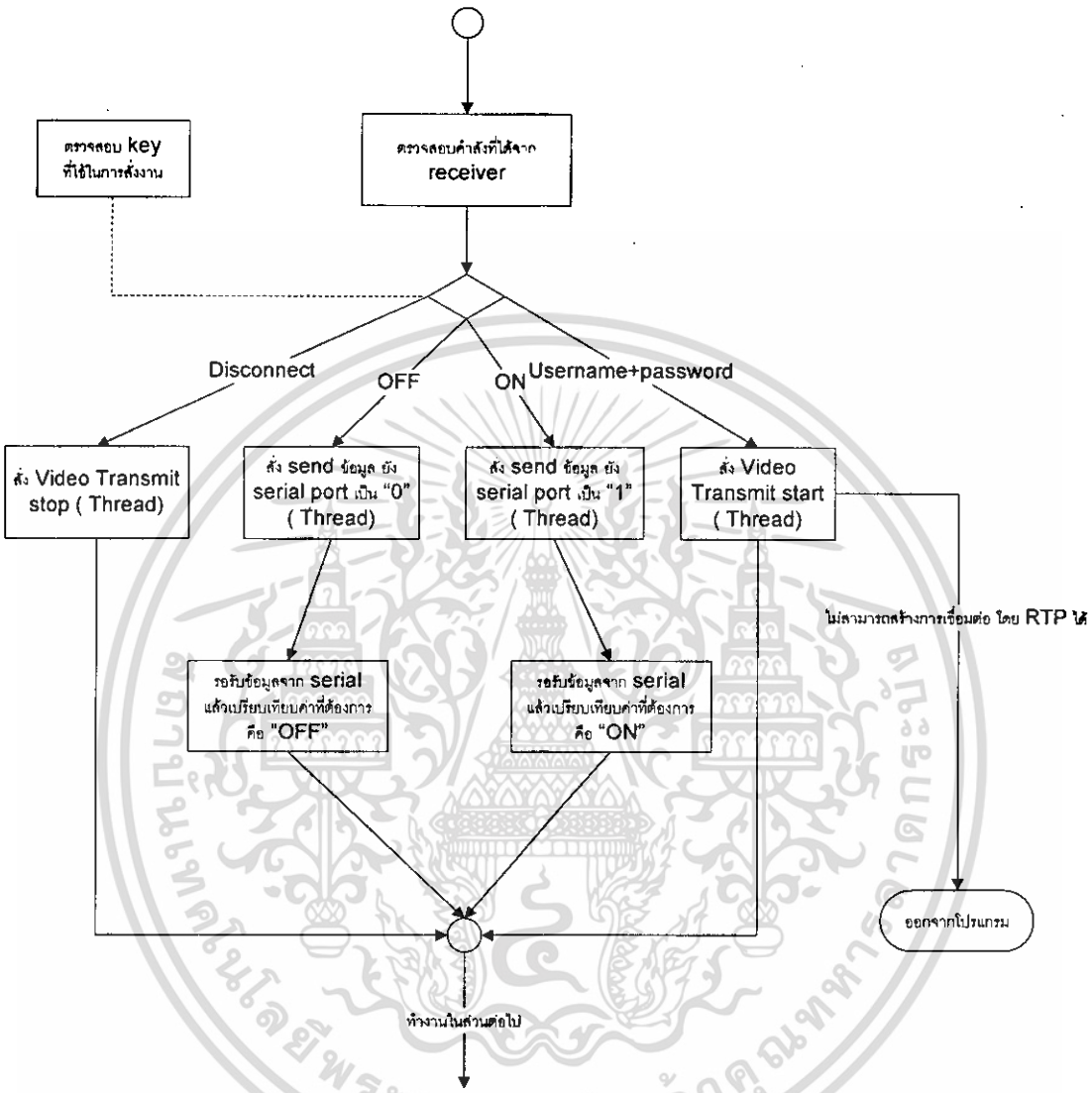
รูปที่ 3.1 แผนภาพการทำงานของโปรแกรมหลัก

ส่วนโปรแกรมย่อยฝั่งเครื่องคอมพิวเตอร์ภายในห้องเรียนเพื่อส่งภาพวีดีโอไปยังเครื่องคอมพิวเตอร์ในเครือข่ายแลนเดียวกันที่เรียกดู



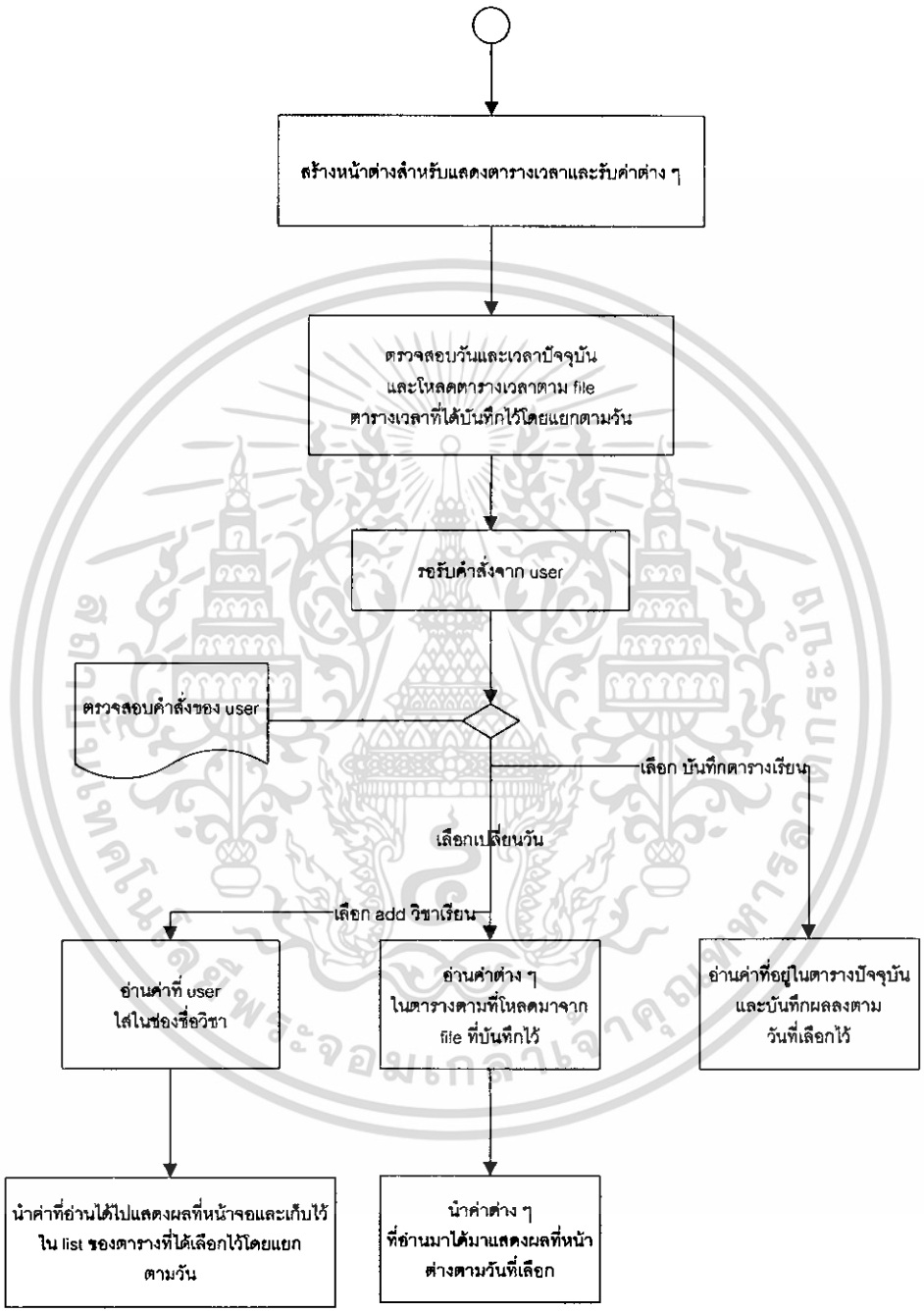
รูปที่ 3.2 แผนภาพการทำงานของโปรแกรมเพื่อส่งภาพวีดีโอ

ส่วนโปรแกรมย่อยฝั่งเครื่องคอมพิวเตอร์ภายในห้องเรียนเพื่อควบคุมการจ่ายไฟให้กับอุปกรณ์ไฟฟ้า

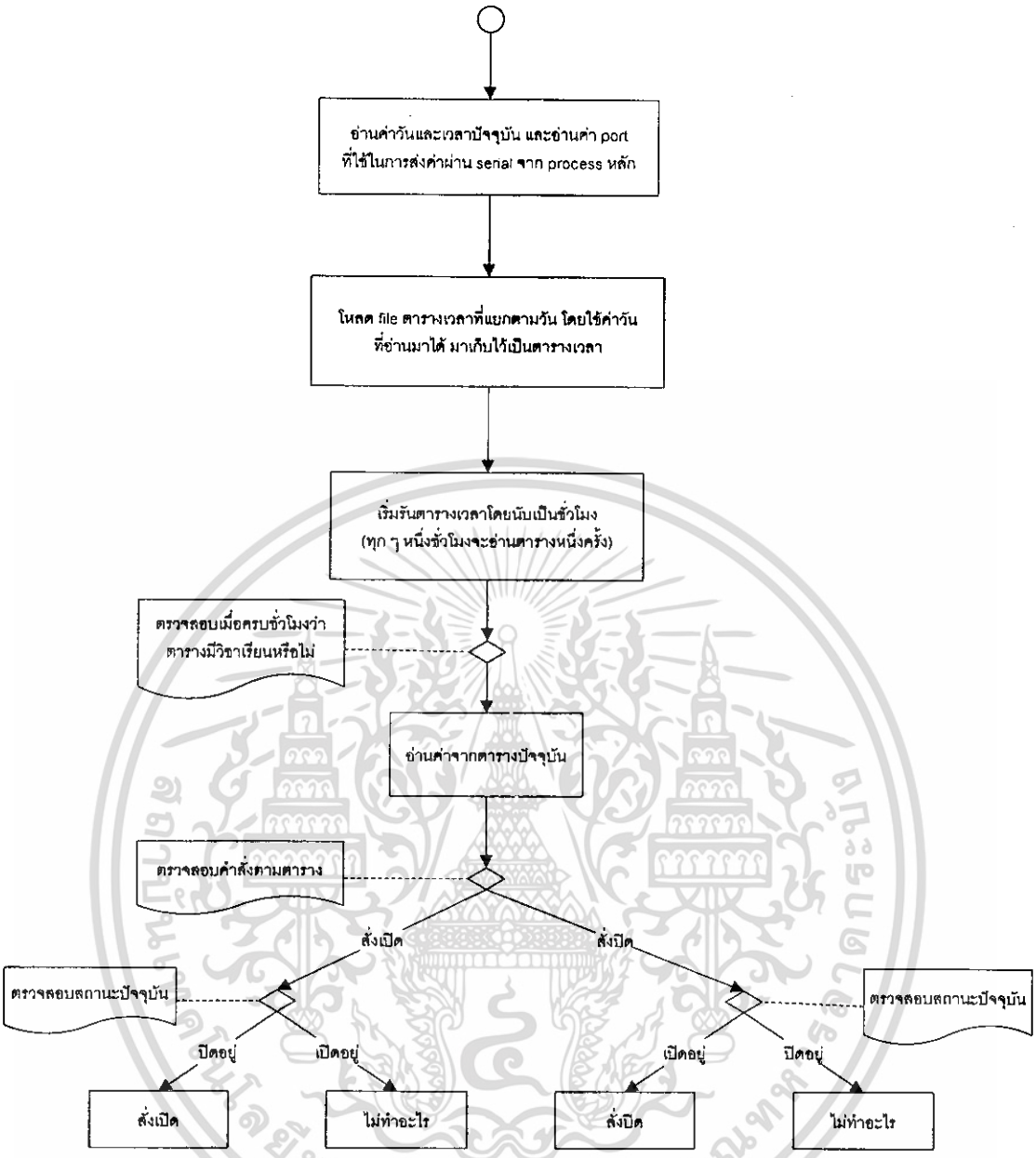


รูปที่ 3.3 แผนภาพการทำงานโปรแกรมเพื่อควบคุมการจ่ายไฟให้กับอุปกรณ์ไฟฟ้า

ส่วนโปรแกรมย่อยฝั่งเครื่องคอมพิวเตอร์ภายในห้องเรียน โดยบันทึกตารางเวลาและควบคุมอุปกรณ์ไฟฟ้าตามตารางเวลาที่กำหนดไว้ เพื่อควบคุมการจ่ายไฟของอุปกรณ์ควบคุมการจ่ายไฟให้กับอุปกรณ์ไฟฟ้า



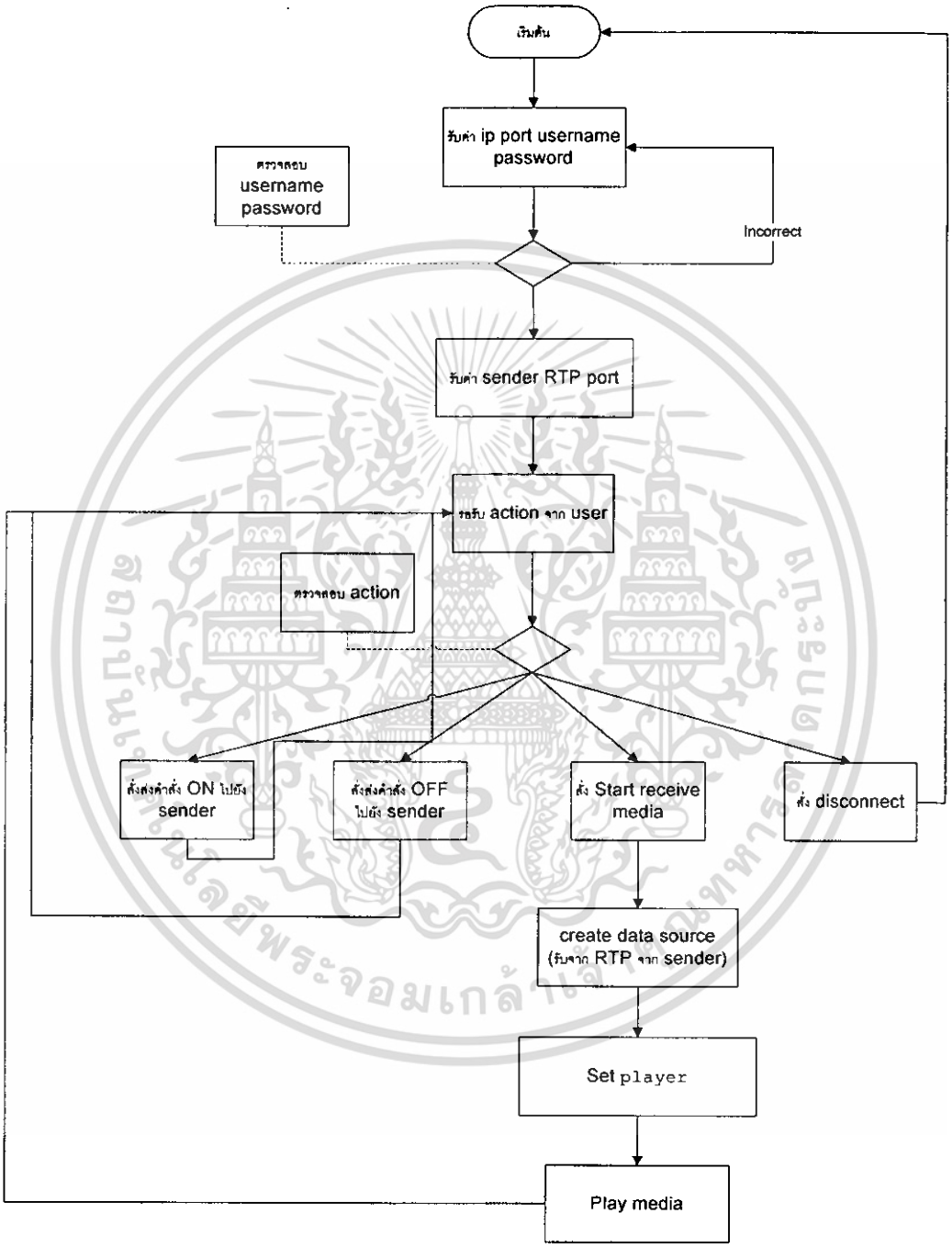
รูปที่ 3.4 แผนภาพการทำงานของโปรแกรมย่อยกระบวนการบันทึกตารางเวลา



รูปที่ 3.5 แผนภาพการทำงานของโปรแกรมย่อยกระบวนการส่งคำสั่งเปิดและปิดตามตารางเวลา

3.2.2 ส่วนโปรแกรมฝั่งเครื่องคอมพิวเตอร์ในเครือข่ายแลนเดียวกันที่เรียกดู

ส่วนโปรแกรมฝั่งเครื่องคอมพิวเตอร์ในเครือข่ายแลนเดียวกันที่เรียกดูภาพแล้วสั่งเปิดและปิดการจ่ายไฟ

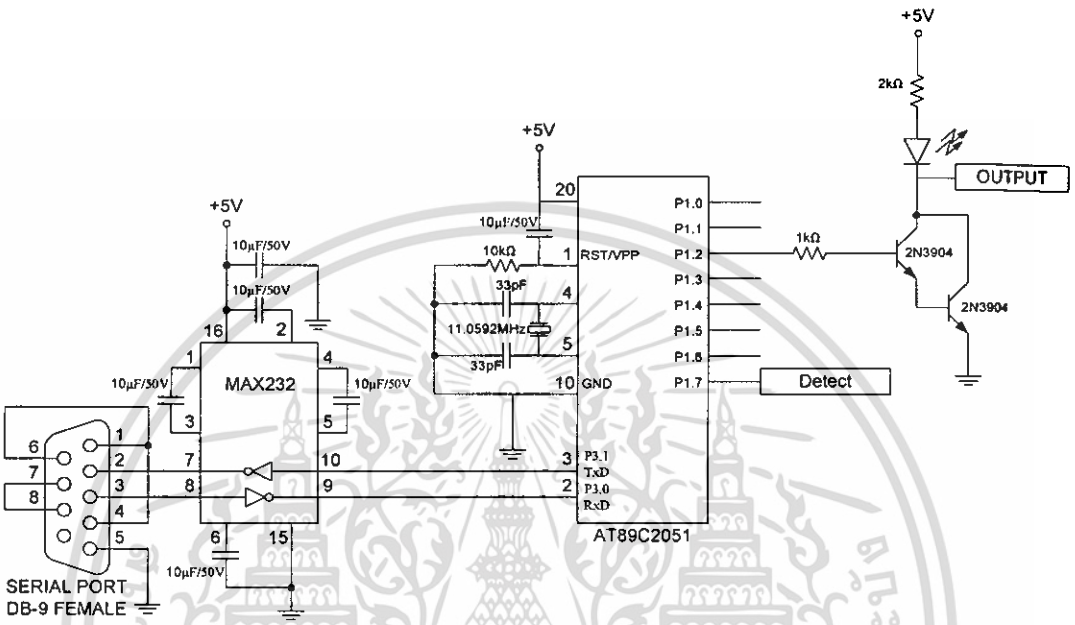


รูปที่ 3.6 แผนภาพการทำงานโปรแกรมฝั่งเครื่องคอมพิวเตอร์ในเครือข่ายแลนเดียวกัน

3.3 การสร้างอุปกรณ์ควบคุมการจ่ายไฟให้กับอุปกรณ์ไฟฟ้า

3.3.1 วงจรสื่อสารข้อมูลผ่านพอร์ตอนุกรมของไมโครคอนโทรลเลอร์

วงจรสื่อสารข้อมูลผ่านพอร์ตอนุกรมของไมโครคอนโทรลเลอร์นี้ใช้ไอซี MAX 232 เพื่อใช้สื่อสารระหว่างพอร์ตอนุกรมของคอมพิวเตอร์และไมโครคอนโทรลเลอร์ เพื่อให้ไมโครคอนโทรลเลอร์สามารถติดต่อสื่อสารกับคอมพิวเตอร์ได้



รูปที่ 3.7 วงจรสื่อสารข้อมูลผ่านพอร์ตอนุกรมของไมโครคอนโทรลเลอร์

3.3.2 วงจรควบคุมการจ่ายไฟ

วงจรควบคุมการจ่ายไฟมีหลักการทำงานดังนี้

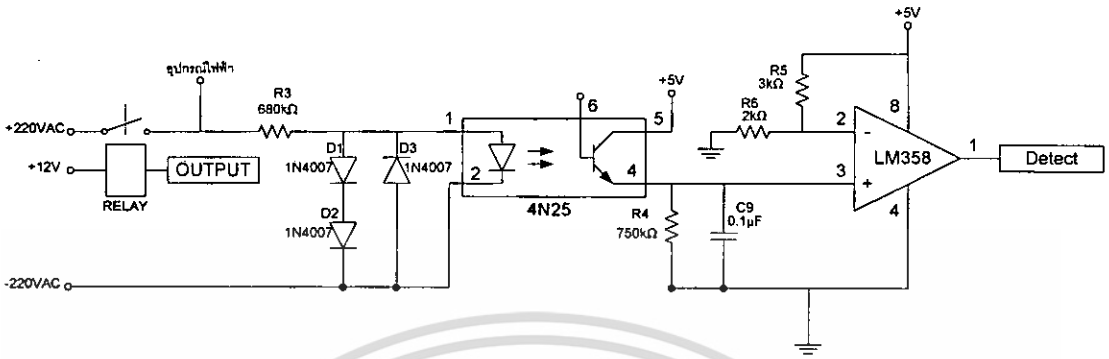
<u>กรณีที่รับข้อมูลให้จ่ายไฟ</u>

เมื่อไมโครคอนโทรลเลอร์ส่งแรงดัน 5 โวลต์ ที่ขา P1.2 แรงดัน 5 โวลต์ก็จะถูกขับด้วยทรานซิสเตอร์ 2 ตัว ที่นำมาต่อกันแบบดาร์ลิงตันเพื่อเพิ่มกระแสให้รีเลย์สามารถทำงานได้ เมื่อรีเลย์ทำงานก็จะมีไฟ 220 โวลต์ซึ่งเป็นกระแสสลับผ่านอุปกรณ์ไฟฟ้าที่ต่ออยู่กับวงจร เมื่อมีกระแสไฟฟ้าในวงจรภายในไอซีเบอร์ 4N25 จะเกิดการเหนี่ยวนำให้เกิดแรงดันอินพุตของวงจรเปรียบเทียบแรงดัน วงจรเปรียบเทียบแรงดันจะทำการเปรียบเทียบแรงดันอินพุตกับแรงดันอ้างอิง แรงดันอินพุตจะมากกว่าแรงดันอ้างอิงทำให้เอาต์พุตที่ได้จากวงจรเปรียบเทียบแรงดันจะมีค่าเป็นลอจิก “1”

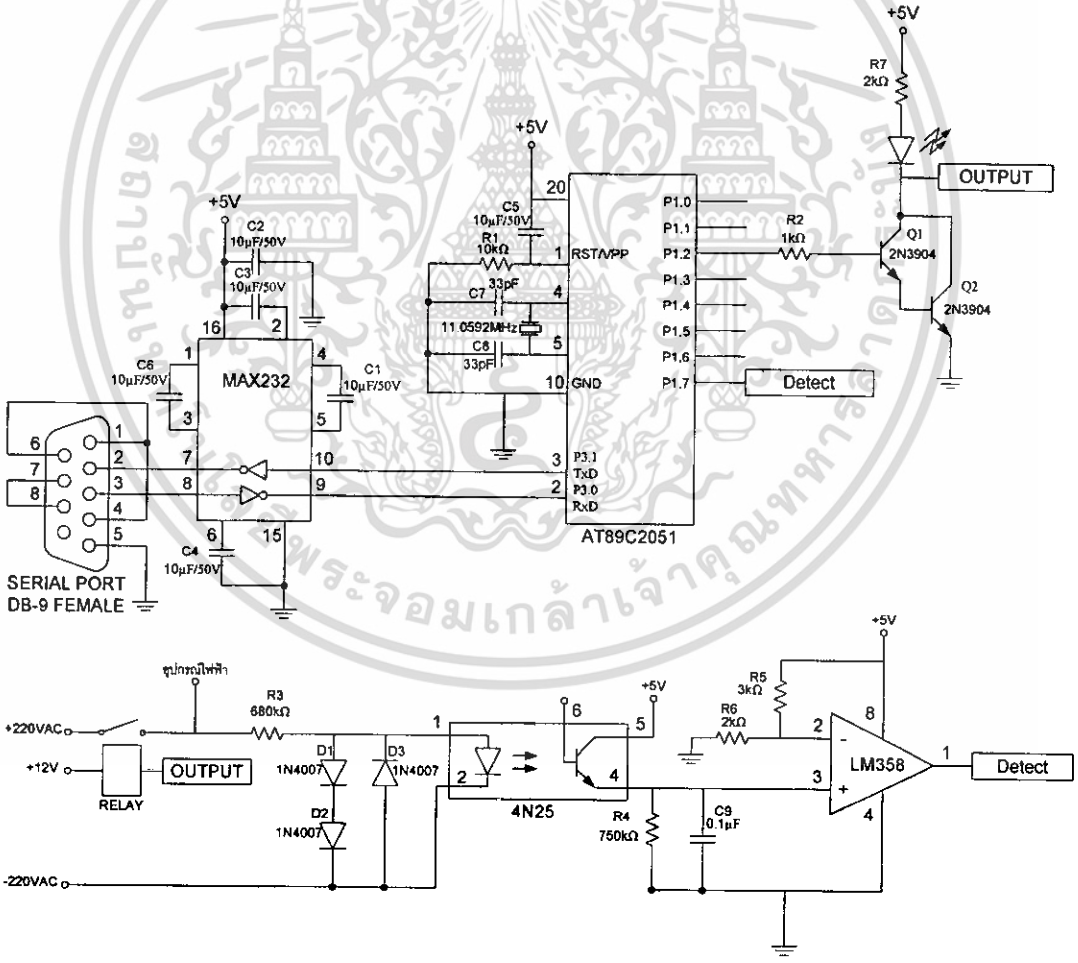
<u>กรณีที่รับข้อมูลให้ตัดไฟ</u>

เมื่อไมโครคอนโทรลเลอร์ ที่ขา P1.2 มีแรงดัน 0 โวลต์ไม่สามารถทำให้รีเลย์ทำงานได้ ดังนั้นภายในไอซีเบอร์ 4N25 จะไม่เหนี่ยวนำให้เกิดแรงดันอินพุตของวงจรเปรียบเทียบแรงดัน ดังนั้นแรงดันอินพุตของวงจรเปรียบเทียบแรงดันจึงเป็น 0 เมื่อเปรียบเทียบกับแรงดันอ้างอิง แรงดันอ้างอิงมีค่ามากกว่าแรงดันอินพุต เอาต์พุตของวงจรเปรียบเทียบแรงดันจึงมีค่าเป็นลอจิก “0”

เอาต์พุตที่ได้จากวงจรเปรียบเทียบแรงดันจะเป็นอินพุตให้กับไมโครคอนโทรลเลอร์ที่ขา P1.7 เพื่อให้ไมโครคอนโทรลเลอร์ส่งข้อมูลไปให้กับคอมพิวเตอร์ทราบว่าอุปกรณ์ไฟฟ้านั้นทำงานหรือไม่



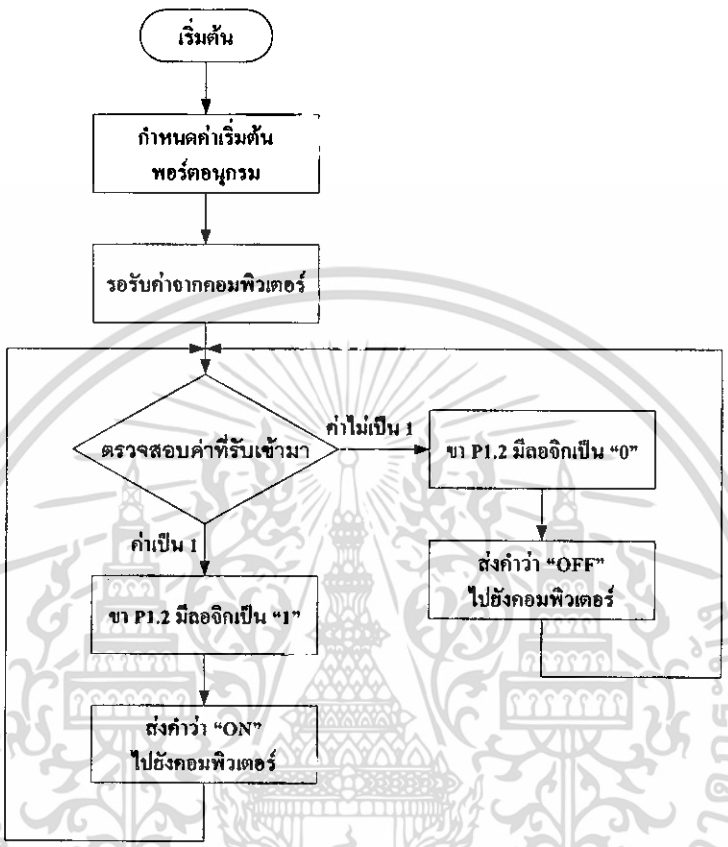
รูปที่ 3.8 วงจรควบคุมการจ่ายไฟ



รูปที่ 3.9 แสดงวงจรควบคุมการจ่ายไฟผ่านพอร์ตอนุกรม

3.3.3 ส่วนโปรแกรมควบคุมไมโครคอนโทรลเลอร์

ส่วนโปรแกรมควบคุมไมโครคอนโทรลเลอร์เพื่อควบคุมการจ่ายไฟให้กับอุปกรณ์ไฟฟ้าและตอบกลับมายังคอมพิวเตอร์ว่าอุปกรณ์ไฟฟ้านั้นทำงานหรือไม่



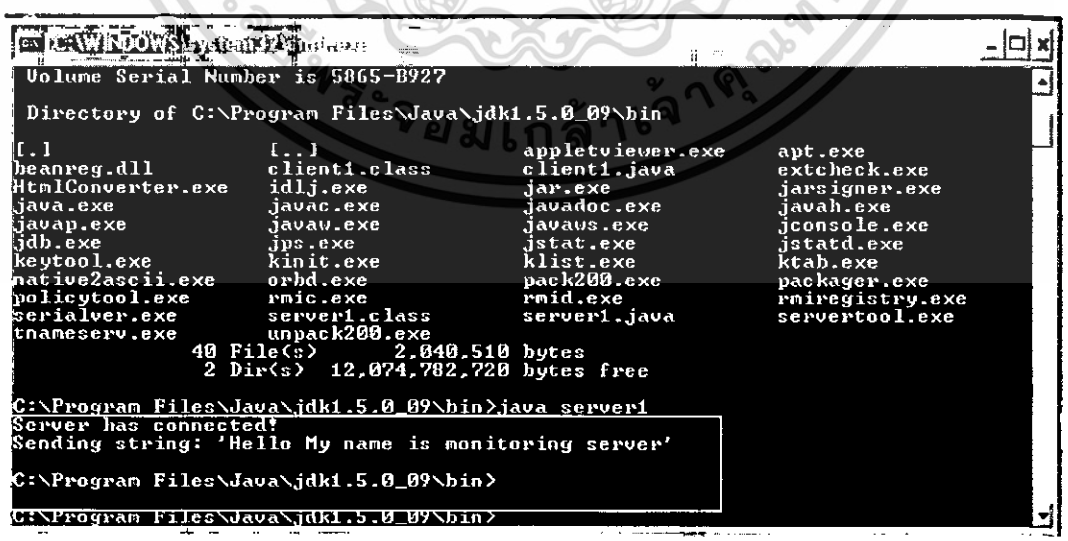
รูปที่ 3.10 แผนภาพการทำงานของโปรแกรมควบคุมไมโครคอนโทรลเลอร์

บทที่ 4
ผลการทดลอง

4.1 การทดลองโปรแกรมติดต่อระหว่างคอมพิวเตอร์กับคอมพิวเตอร์

การทดลองจะเป็นการทดสอบโดยใช้ไอพีแอดเดรสของเครื่องที่เราต้องการติดต่อ โดยจะมีขั้นตอนโปรแกรมดังที่ได้ออกแบบตามที่ได้กล่าวมาแล้วในบทที่ 3 เมื่อเขียนโปรแกรมแล้วจะใช้หน้า Command ในการทดลองโดยเริ่มรันโปรแกรมฝั่งไคลเอนท์ไว้ก่อน แล้วค่อยรันโปรแกรมฝั่งเซิร์ฟเวอร์ เมื่อฝั่งเซิร์ฟเวอร์มีการติดต่อเข้ามาแล้วก็จะมีการส่งข้อความไปยังฝั่งไคลเอนท์ ได้ดังรูป

รูปที่ 4.1 หน้าต่างCommand ฝั่งไคลเอนท์

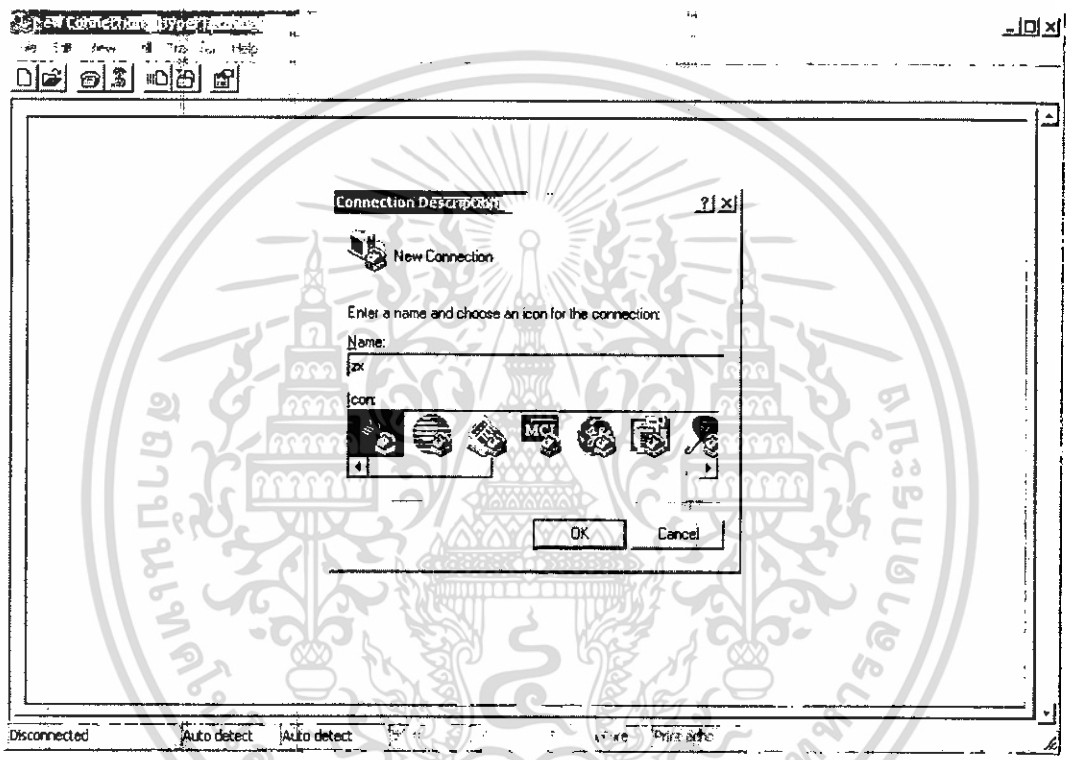


รูปที่ 4.2 หน้าต่างCommand ฝั่งเซิร์ฟเวอร์

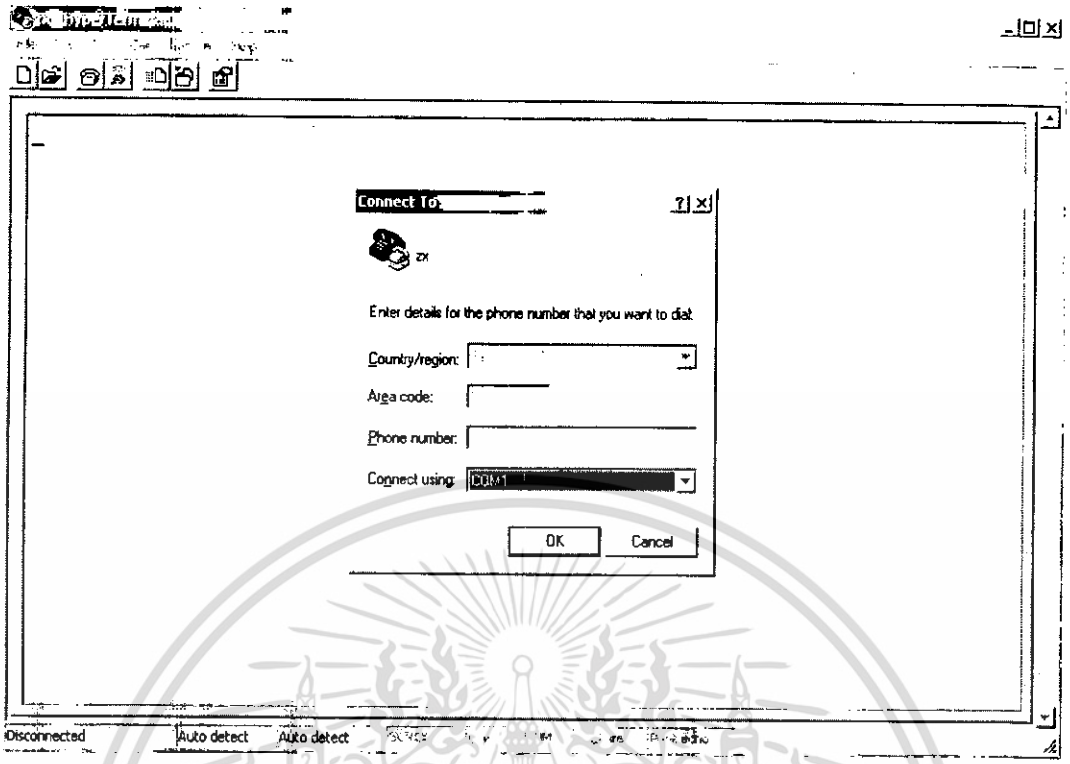
4.2การทดลองการรับ - ส่งข้อมูลระหว่างไมโครคอนโทรลเลอร์กับคอมพิวเตอร์เพื่อควบคุมอุปกรณ์ไฟฟ้า

ขั้นตอนการทดลอง

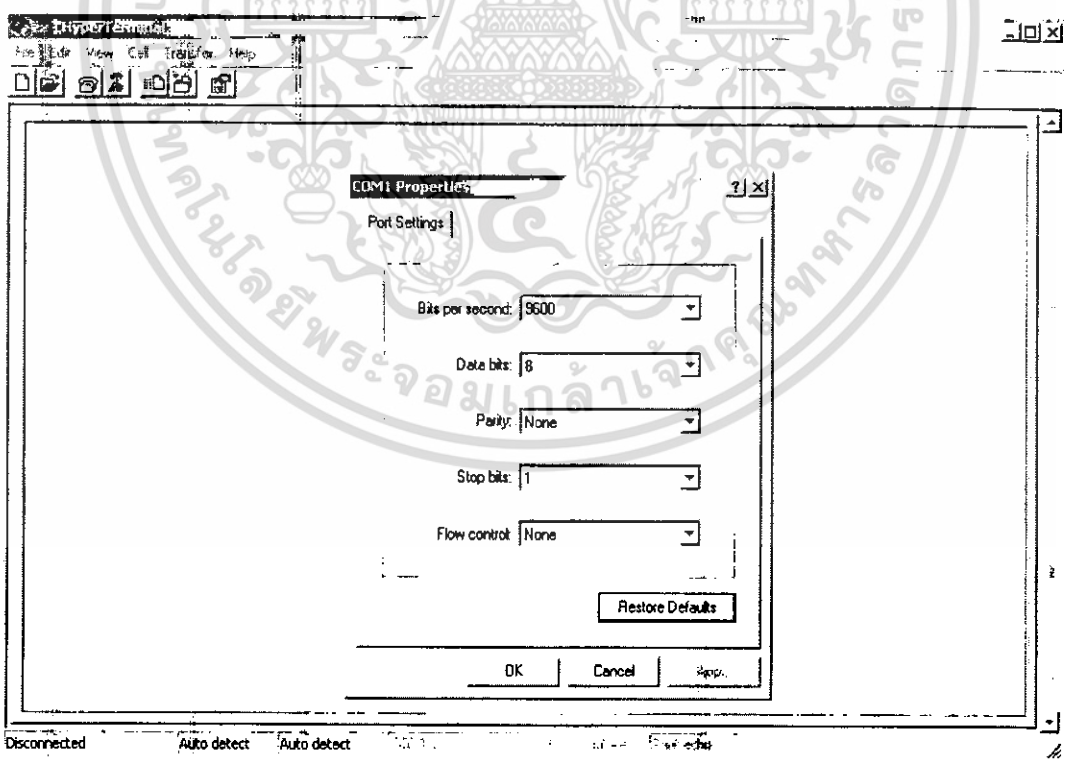
1. เขียนโปรแกรมรับ - ส่งข้อมูลระหว่างไมโครคอนโทรลเลอร์กับคอมพิวเตอร์ทางเดียว จากนั้นโปรแกรมข้อมูลลงในไอซีเบอร์ AT89C2051
2. ต่อวงจรตามรูปที่ 3.9
3. จ่ายไฟให้กับวงจร
4. เปิดโปรแกรมไฮเปอร์เทอร์มินอล (Hyper terminal)



รูปที่ 4.3 แสดงหน้าต่างของโปรแกรมไฮเปอร์เทอร์มินอล



รูปที่ 4.4 กำหนดโหมดการทำงานให้ไฮเปอร์เทอร์มินอล โดยการทดลองนี้เลือก COM1



รูปที่ 4.5 กำหนดคุณสมบัติให้กับพอร์ตอนุกรมที่เลือก

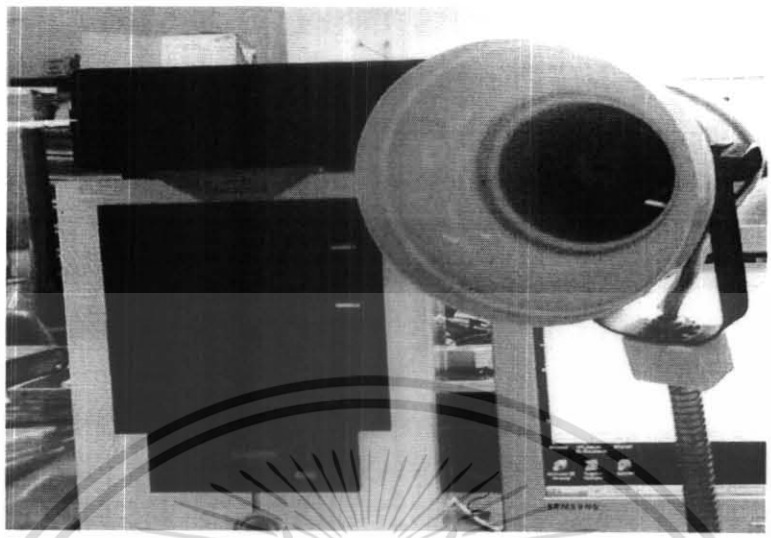
5. กด 1 ที่คีย์บอร์ด สังเกตหลอดแอลอีดีและอุปกรณ์ไฟฟ้าที่ต่ออยู่กับวงจรซึ่งในการทดลองนี้ใช้โคมไฟต่อกับอุปกรณ์

รูปที่ 4.6 ไฟที่หลอดแอลอีดีติดเมื่อกดคีย์บอร์ดเลข 1

6. ที่หน้าต่างไฮเปอร์เทอร์มินอลจะมีคำว่า ON ส่งมาจากไมโครคอนโทรลเลอร์

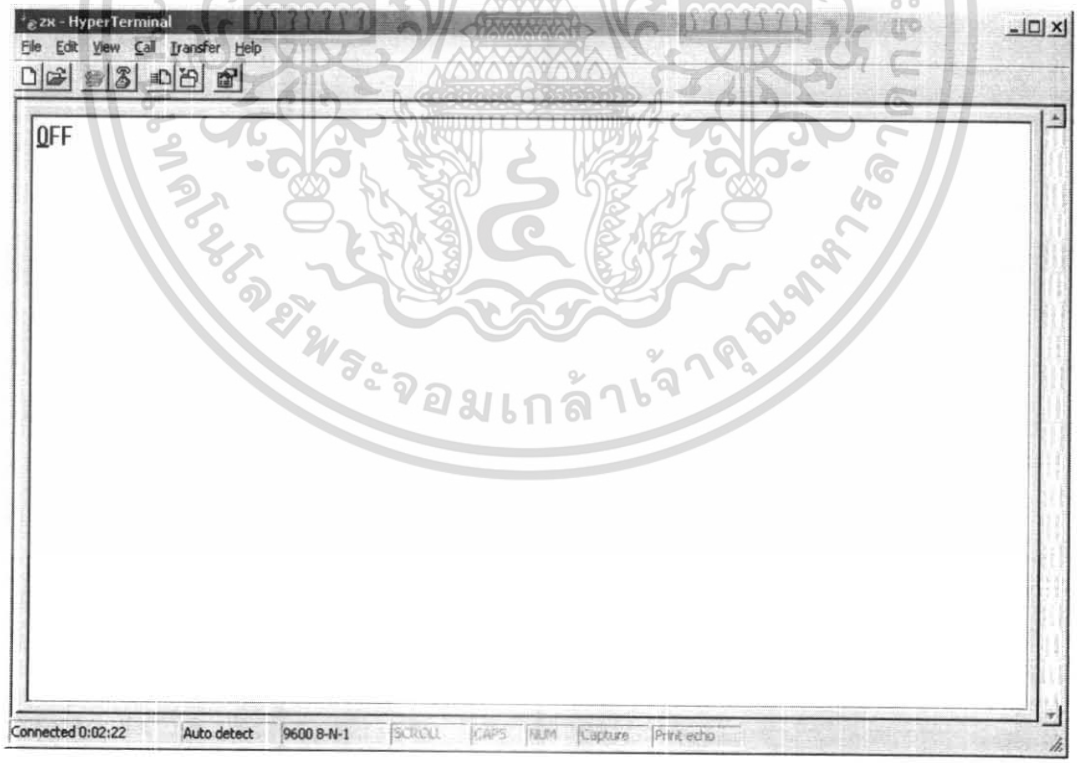
รูปที่ 4.7 ไมโครคอนโทรลเลอร์ส่งคำว่า ON มายังคอมพิวเตอร์แสดงบนไฮเปอร์เทอร์มินอล

7. กดตัวอักษรอื่นที่คีย์บอร์ด สังเกตหลอดแอลอีดีและโคมไฟ

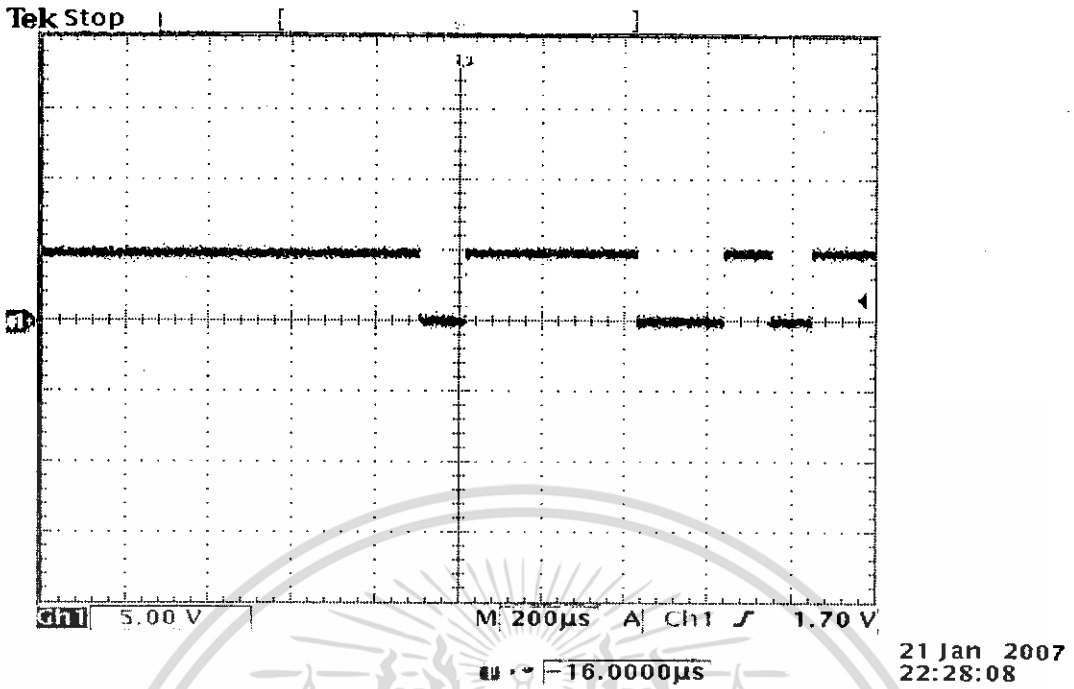


รูปที่ 4.8 ไฟที่หลอดแอลอีดีดับเมื่อกดคีย์บอร์ดด้วยตัวอักษรอื่น ๆ

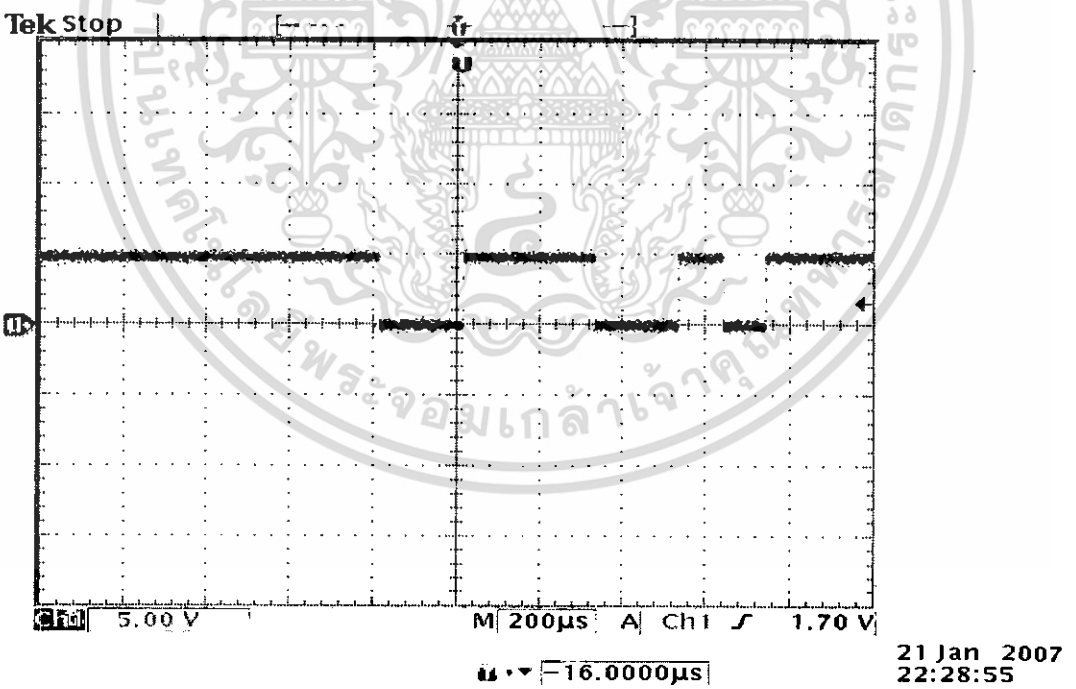
8. ที่หน้าต่างไฮเปอร์เทอร์มินอลจะมีคำว่า OFF ส่งมาจากไมโครคอนโทรลเลอร์



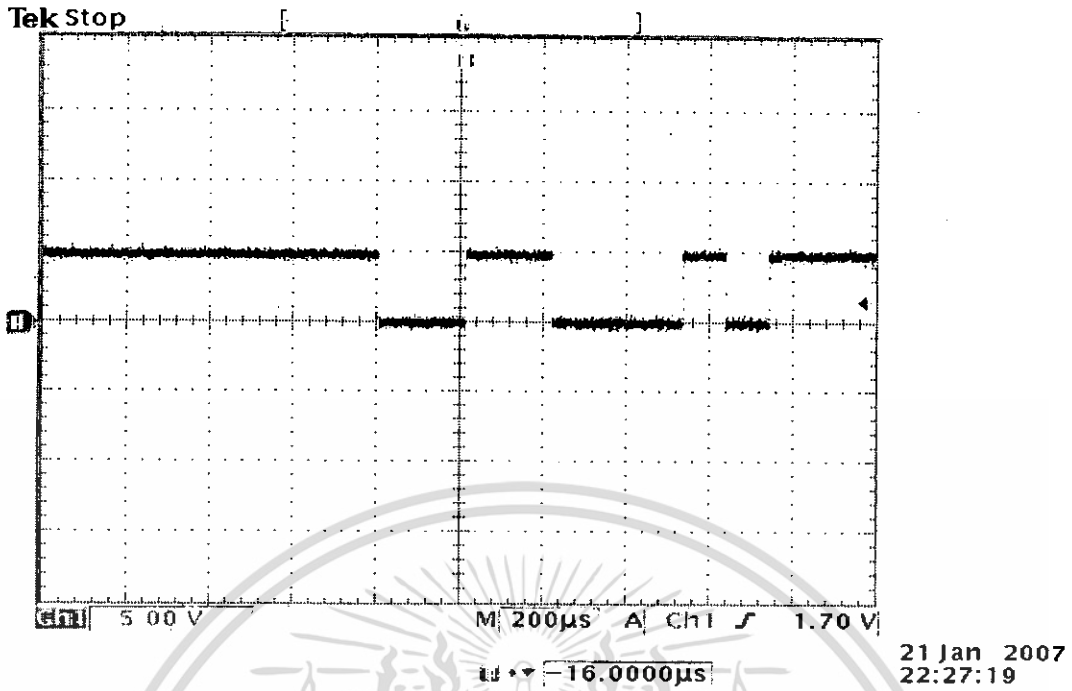
รูปที่ 4.9 ไมโครคอนโทรลเลอร์ส่งคำว่า OFF มายังเครื่องคอมพิวเตอร์แสดงบนไฮเปอร์เทอร์มินอล



รูปที่ 4.10 แสดงสัญญาณที่ไมโครคอนโทรลเลอร์ส่งมาให้กับเครื่องคอมพิวเตอร์
Ch1 คือ ข้อมูลที่เป็นตัวอักษร O



รูปที่ 4.11 แสดงสัญญาณที่ไมโครคอนโทรลเลอร์ส่งมาให้กับเครื่องคอมพิวเตอร์
Ch1 คือ ข้อมูลที่เป็นตัวอักษร N

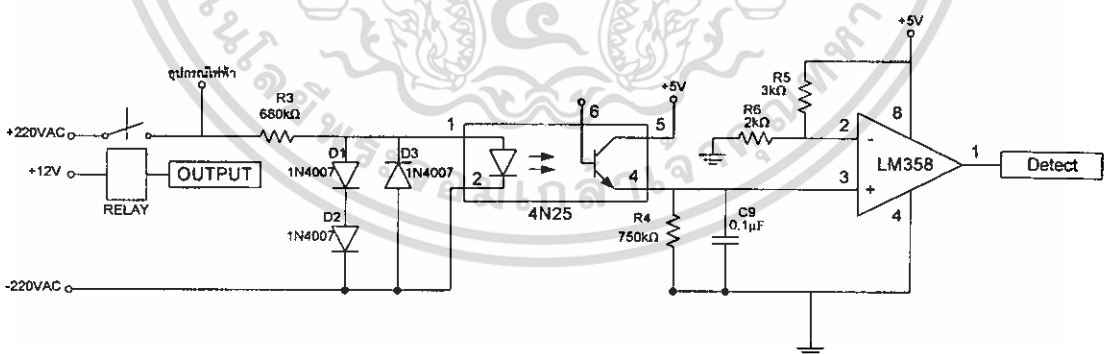


รูปที่ 4.12 แสดงสัญญาณที่ไมโครคอนโทรลเลอร์ส่งมาให้กับเครื่องคอมพิวเตอร์
Ch1 คือ ข้อมูลที่เป็นตัวอักษร F

การทดลองเมื่อกำหนดให้อุปกรณ์ไฟฟ้าสามารถทำงานได้

ขั้นตอนการทดลอง

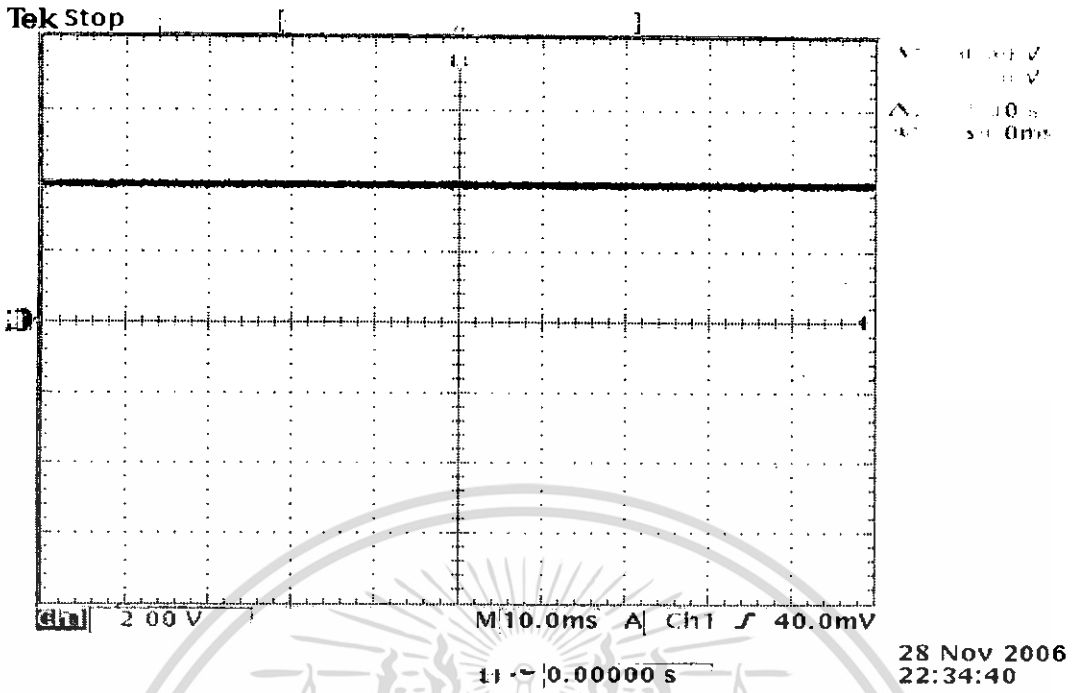
1. ต่อวงจรตามรูปที่ 4.13



รูปที่ 4.13 แสดงวงจรที่ใช้ในการทดลองเมื่อกำหนดให้อุปกรณ์ไฟฟ้าสามารถทำงานได้

3. จ่ายไฟให้กับวงจร

4. จับสัญญาณเอาต์พุตที่ขา 1 ของไอซีเบอร์ LM358

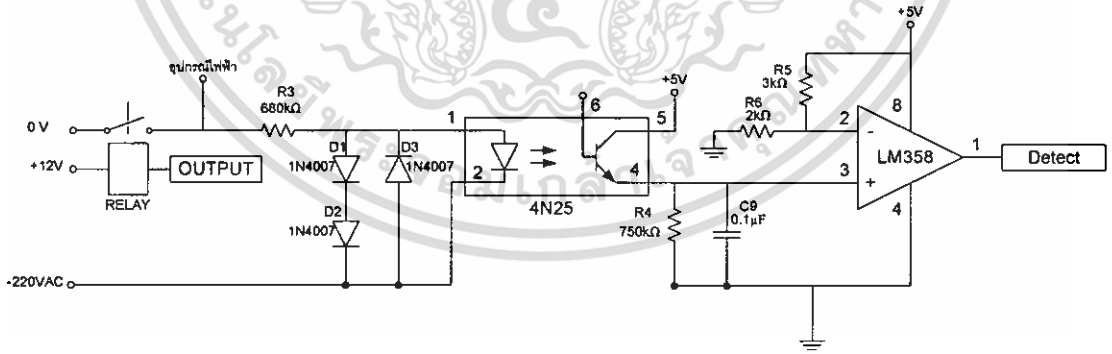


รูปที่ 4.14 แสดงผลการทดลองเมื่อกำหนดให้อุปกรณ์ไฟฟ้าสามารถทำงานได้
Ch1 คือ สัญญาณเอาต์พุตที่ได้จากวงจรเปรียบเทียบแรงดัน

การทดลองเมื่อกำหนดให้อุปกรณ์ไฟฟ้าไม่ทำงาน

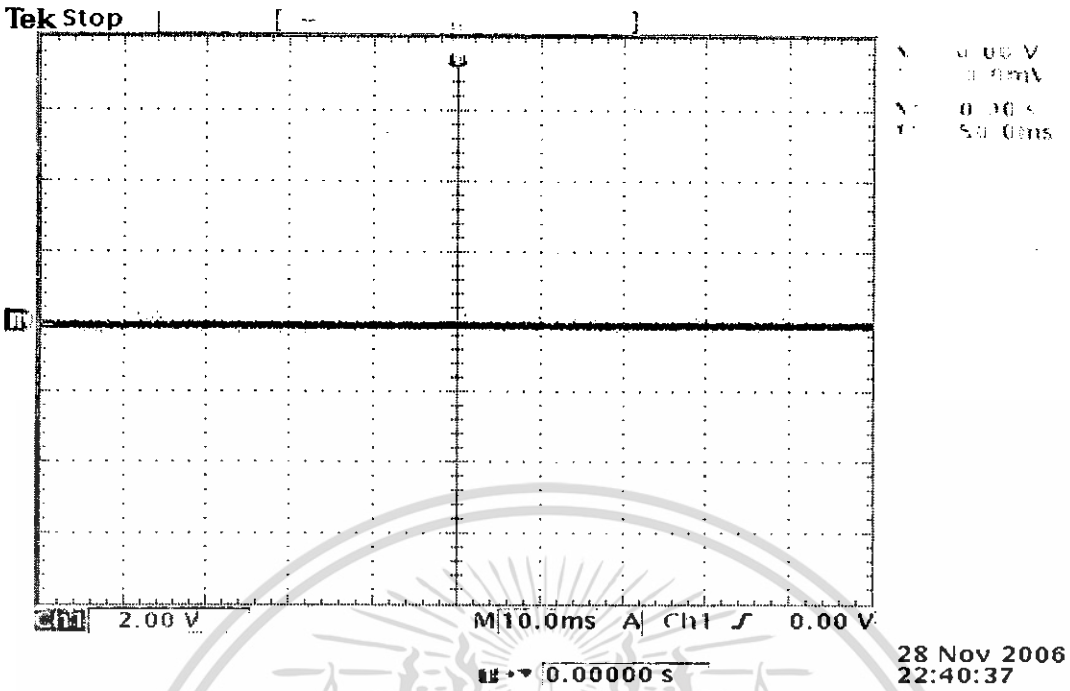
ขั้นตอนการทดลอง

1. ต่อวงจรตามรูปที่ 4.15



รูปที่ 4.15 แสดงวงจรที่ใช้ในการทดลองเมื่อกำหนดให้อุปกรณ์ไฟฟ้าไม่ทำงาน

3. จ่ายไฟให้กับวงจร
4. จับสัญญาณเอาต์พุตที่ขา 1 ของไอซีเบอร์ LM358



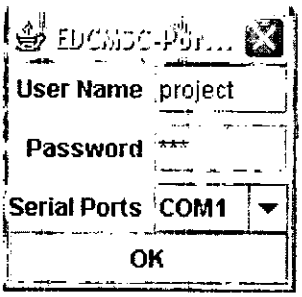
รูปที่ 4.16 แสดงผลการทดลองเมื่อกำหนดให้อุปกรณ์ไฟฟ้าไม่ทำงาน
Ch1 คือ สัญญาณเอาต์พุตที่ได้จากวงจรเปรียบเทียบแรงดัน

4.3 การทดลองโปรแกรมรวมกับวงจรของระบบควบคุมอุปกรณ์ไฟฟ้าและสังเกตการณ์สำหรับห้องเรียน

การทดลองจะเป็นการทดสอบโดยใช้ไอพีแอดเดรสของเครื่องที่เราต้องการติดต่อ โดยจะมีขั้นตอนโปรแกรมดังที่ได้ออกแบบตามที่ได้กล่าวมาแล้วในบทที่ 3 หัวข้อที่ 3.2 โดยจะแบ่งเป็น 2 ฝั่ง คือ ฝั่งเครื่องคอมพิวเตอร์ภายในห้องเรียน และฝั่งเครื่องคอมพิวเตอร์ในเครือข่ายแลนเดียวกันที่เรียกดู โดยเราจะลำดับผลการทดลองตามการทำงานดังรูป

ส่วนที่ 1 เป็นการเชื่อมต่อกับเครื่องคอมพิวเตอร์ภายในห้องเรียน

1. โดยที่ฝั่งเครื่องคอมพิวเตอร์ภายในห้องเรียนจะทำการเปิดพอร์ต
2. ที่คอมพิวเตอร์ในเครือข่ายแลนเดียวกันก็จะทำการเข้ารหัสผ่านไปติดต่อกับเครื่องที่อยู่ภายในห้องเรียน
3. เมื่อสามารถเชื่อมต่อได้แล้วจะสามารถรับชมภาพได้
4. แล้วมีการทำกระบวนการสั่งงานต่อไป

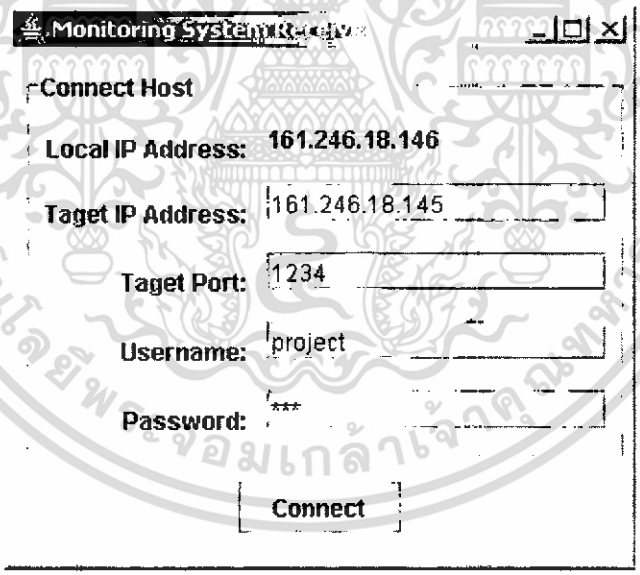


Problems Javadoc Declaration Console

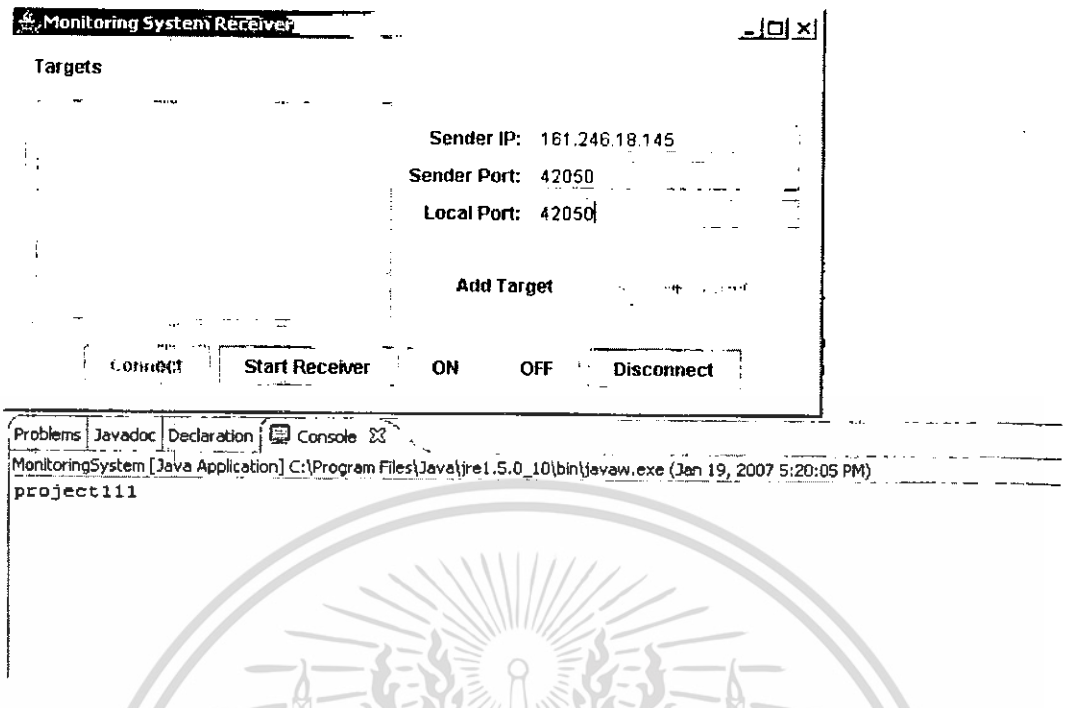
CommPort [Java Application] C:\Program Files\Java\jre1.5.0_09\bin\javaw.exe (19 ม.ค. 2550, 17:15:22)

```
Trying to open COM1...
Ready to read and write port.
Username : project
Password : 111
```

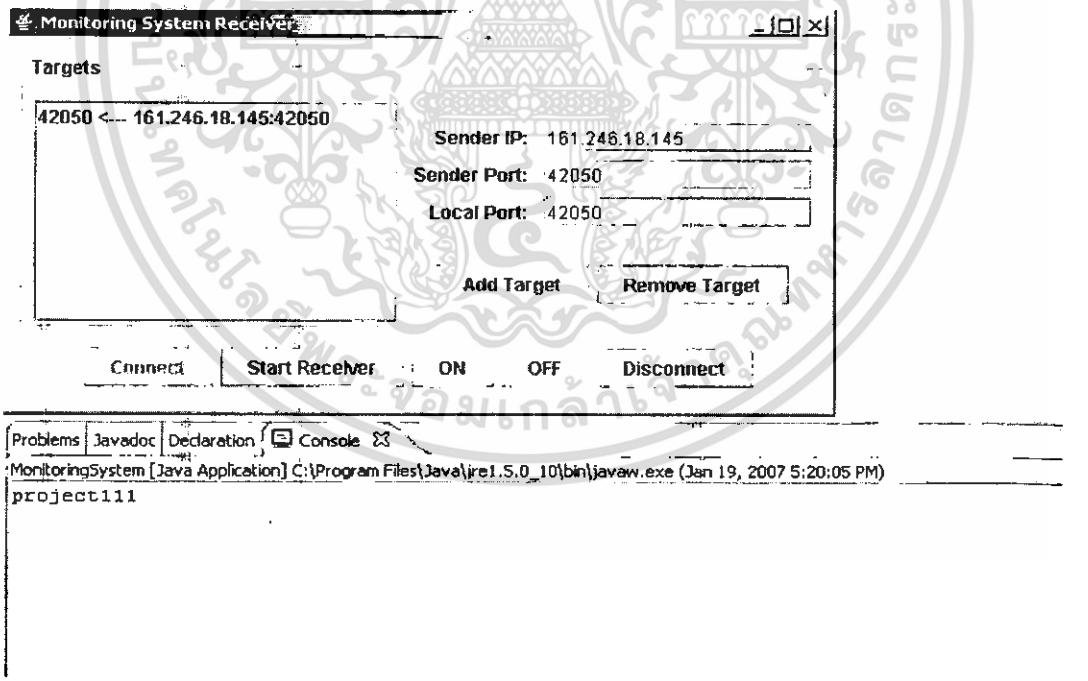
รูปที่ 4.17 ทางฝั่งเครื่องคอมพิวเตอร์ภายในห้องเรียน ทำการตั้งค่า Username Password และเปิดพอร์ตที่ติดต่อกับวงจร



รูปที่ 4.18 ทางฝั่งเครื่องคอมพิวเตอร์ในเครือข่ายแลนเดียวกัน ทำการใส่ไอพีแอดเดรส Username Password ให้ตรงกับทางฝั่งเครื่องคอมพิวเตอร์ภายในห้องเรียนที่จะทำการติดต่อ



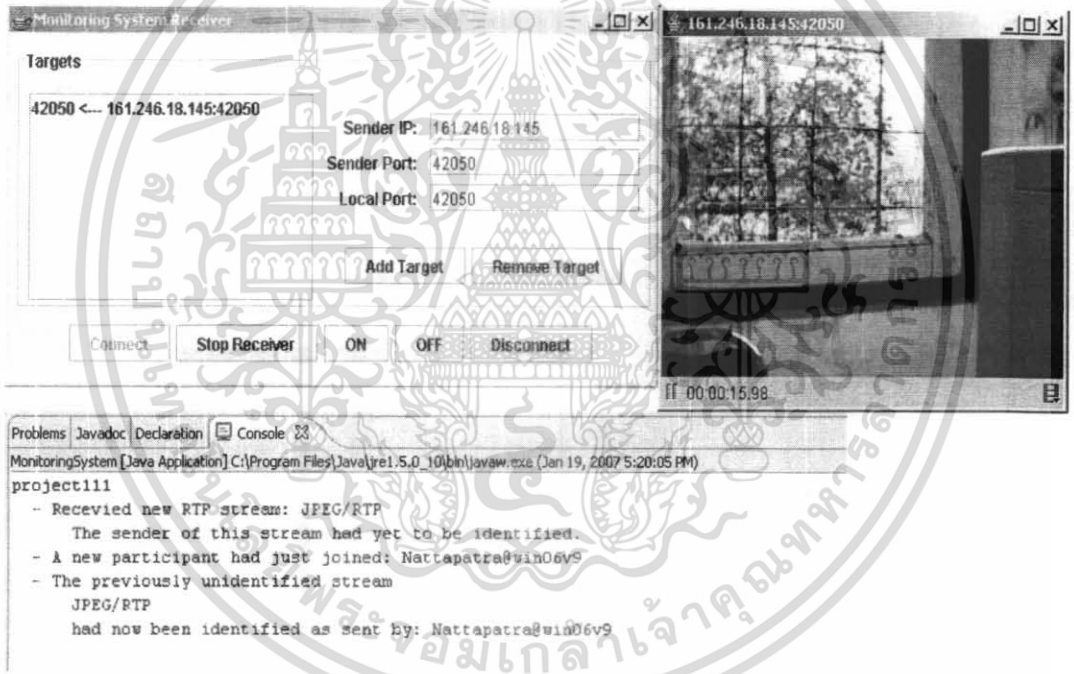
รูปที่ 4.19 ทางฝั่งเครื่องคอมพิวเตอร์ในเครือข่ายแลนเดียวกัน เมื่อกด Connect แล้ว จะทำการใส่ไอพีแอดเดรส และพอร์ตที่กำหนดไว้ให้ตรงกัน



รูปที่ 4.20 ทางฝั่งเครื่องคอมพิวเตอร์ในเครือข่ายแลนเดียวกัน กด Add Target เพื่อจดจำไอพีแอดเดรสในการติดต่อครั้งต่อไป

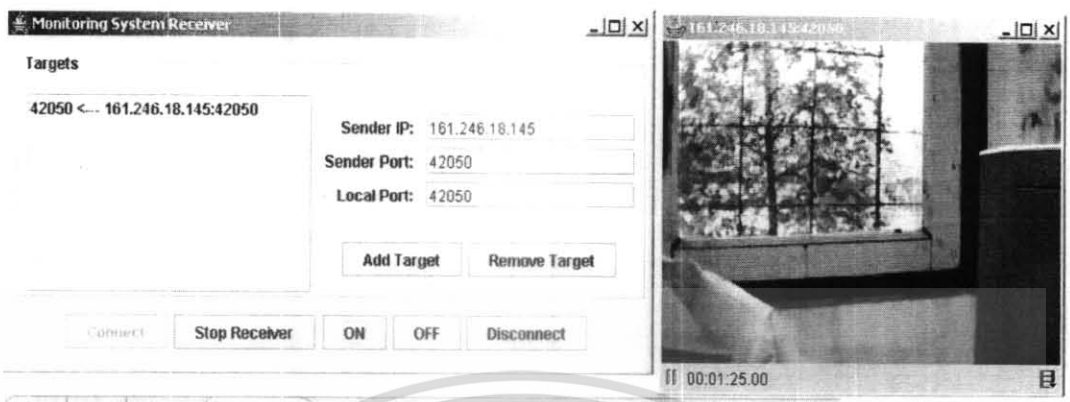
```
Problems Javadoc Declaration Console
CommPort [Java Application] C:\Program Files\Java\jre1.5.0_09\bin\javaw.exe (19 ม.ค. 2550, 17:15:22)
Username : project
Password : 111
161.246.18.146
Video transmitted as:
  JPEG/RTP, 320x240, FrameRate=15.0
streams is [Lcom.sun.media.multiplexer.RawBufferMux$RawBufferSourceStream;@b8f82d : 1
sink: setOutputLocator rtp://161.246.18.146:42050/video
- Setting quality to 0.5 on com.sun.media.codec.video.jpeg.NativeEncoder$1$QCA@192b996
```

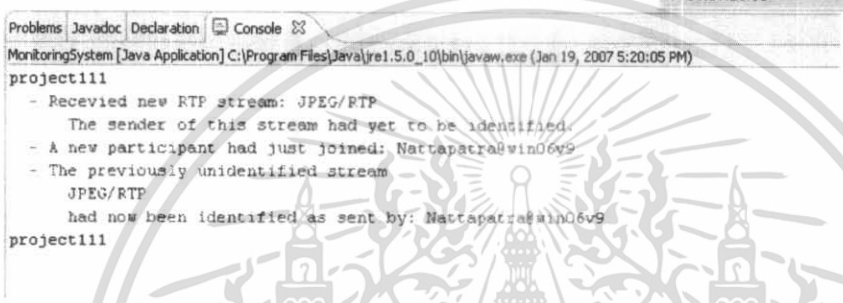
รูปที่ 4.21 ทางฝั่งเครื่องคอมพิวเตอร์ภายในห้องเรียน แสดงรายละเอียดของกล้องเมื่อมีการเริ่มใช้งาน



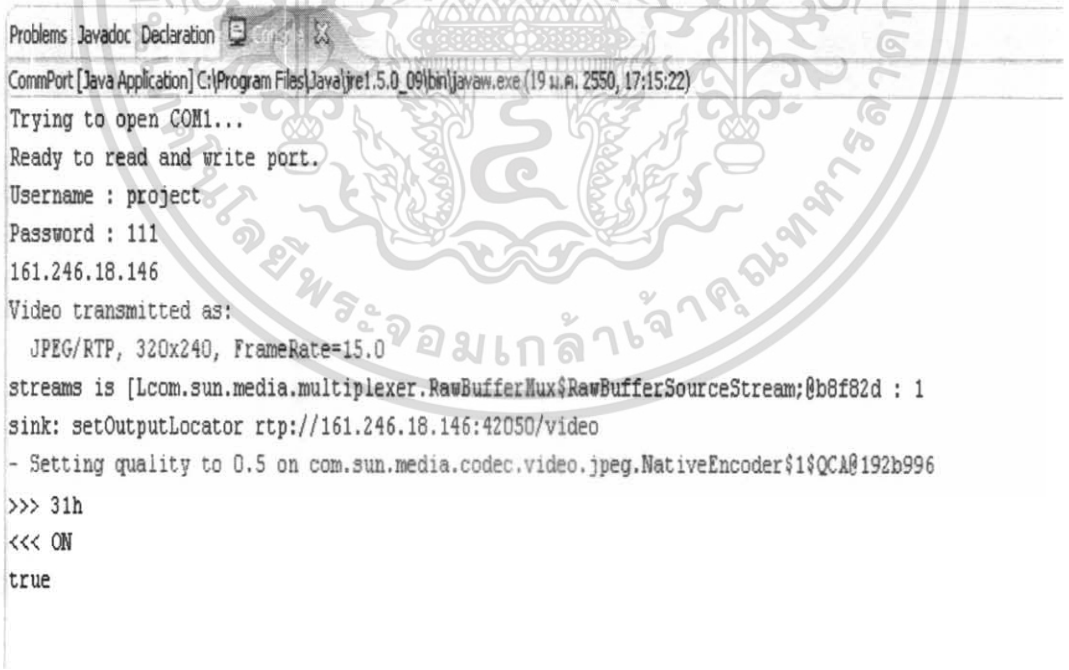
รูปที่ 4.22 ทางฝั่งเครื่องคอมพิวเตอร์ในเครือข่ายแลนเดียวกัน กด Start Receiver จะสามารถรับชมภาพได้

ส่วนที่ 2 จะเป็นส่วนระบบควบคุมวงจรการจ่ายไฟ



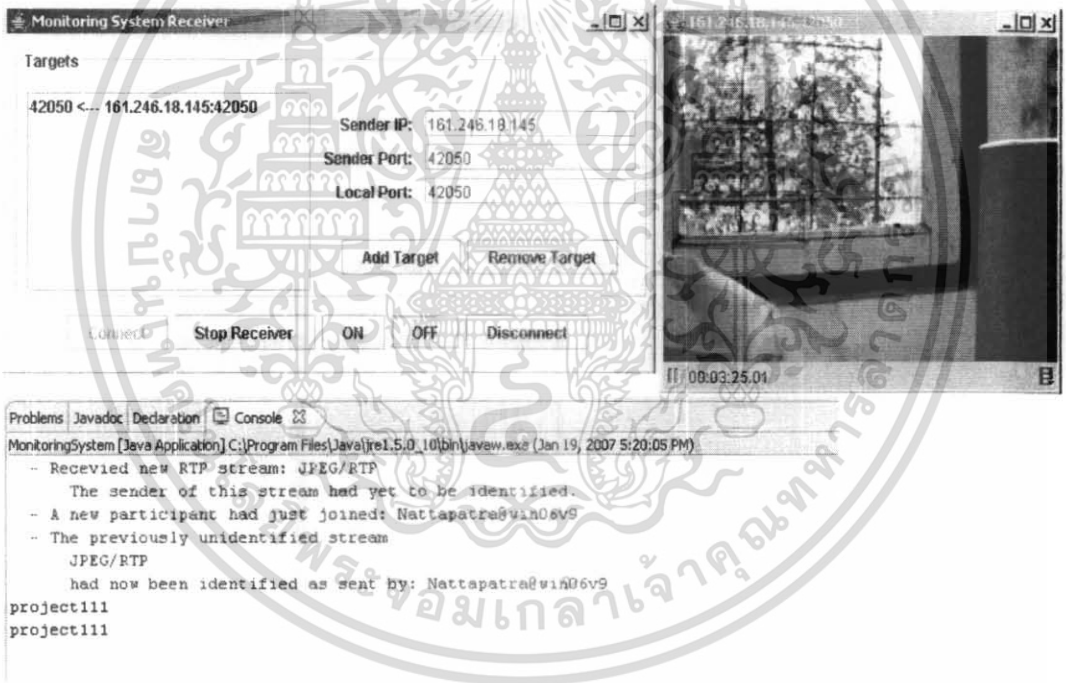


รูปที่ 4.23 ทางฝั่งเครื่องคอมพิวเตอร์ในเครือข่ายแลนเดียวกัน เมื่อทำการกด ON



รูปที่ 4.24 ทางฝั่งเครื่องคอมพิวเตอร์ภายในห้องเรียน ค่าที่ได้จากการกด ON ส่งค่า 1 ที่เป็นรหัสแอสกี้ไป แล้วที่วงจรจะส่งค่า ON กลับมา เมื่อตรงกับโปรแกรมจะแสดงคำว่า true

รูปที่ 4.25 แสดงอุปกรณ์เมื่อมีการกด ON ไฟจะติด

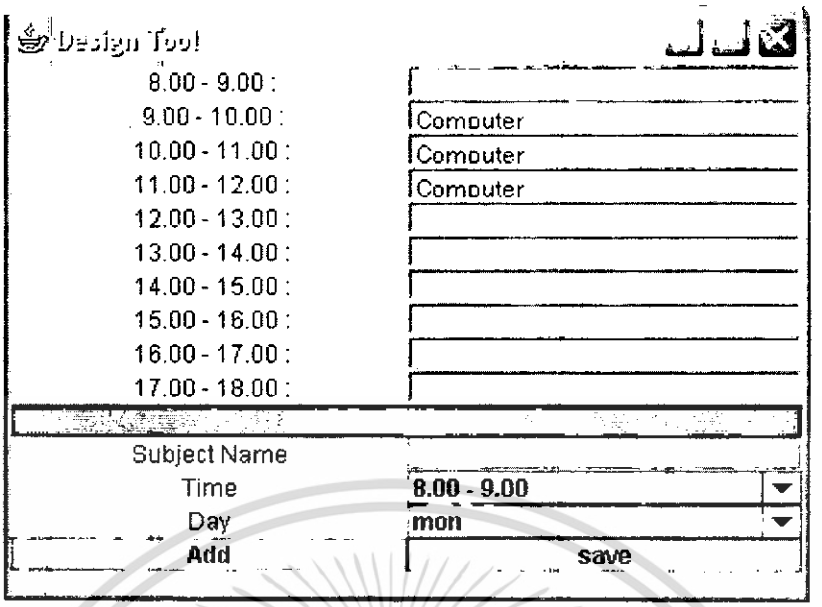


รูปที่ 4.26 ทางฝั่งเครื่องคอมพิวเตอร์ในเครือข่ายแลนเดียวกัน เมื่อทำการกด OFF

```
Problems  Javadoc  Declaration
CommPort [Java Application] C:\Program Files\Java\jre1.5.0_09\bin\javaw.exe (19 ม.ค. 2550, 17:15:22)
Trying to open COM1...
Ready to read and write port.
Username : project
Password : 111
161.246.18.146
Video transmitted as:
  JPEG/RTP, 320x240, FrameRate=15.0
streams is [Lcom.sun.media.multiplexer.RawBufferMux$RawBufferSourceStream;@b8f82d : 1
sink: setOutputLocator rtp://161.246.18.146:42050/video
- Setting quality to 0.5 on com.sun.media.codec.video.jpeg.NativeEncoder$1$QCA@192b996
>>> 31h
<<< ON
true
>>> 0
<<< OFF
true
```

รูปที่ 4.27 ทางฝั่งเครื่องคอมพิวเตอร์ภายในห้องเรียน ค่าที่ได้จากการกด OFF ส่งค่า 0 ที่เป็นรหัสแอสกี้ไป แล้วที่วงจรจะส่งค่า OFF กลับมา เมื่อตรงกับโปรแกรมจะแสดงคำว่า true

รูปที่ 4.28 แสดงอุปกรณ์เมื่อมีการกด OFF ไฟจะดับ



รูปที่ 4.29 แสดงตารางเรียนที่บันทึกไว้

รูปที่ 4.30 แสดงเมื่อถึงเวลาเปิดอุปกรณ์ไฟฟ้า

```
Problems Javadoc Declaration
CommPort (1) [Java Application] C:\Program Files\Java\jre1.5.0_09\bin\javaw.exe (2 มี.ค. 2550, 16:50:49)
Trying to open COM1...
Ready to read and write port.
Username : 1
Password : 1
Wake up! It's 02 มี.ค. 2550 16:51:10.000
>>> 0
<<< OFF
true
```

รูปที่ 4.31 แสดงเมื่อถึงเวลาปิดอุปกรณ์ไฟฟ้า

บทที่ 5
สรุปผลและวิจารณ์

จากการทดลองในแต่ละส่วนของโปรแกรมทั้งการสื่อสารระหว่างคอมพิวเตอร์ 2 เครื่องโดยผ่านแลนและการสื่อสารระหว่างคอมพิวเตอร์กับอุปกรณ์ในบางส่วนโดยผ่านพอร์ตอนุกรมในเทอมแรกนั้น การทดลองในแต่ละส่วนได้ผลตรงตามที่คาดหมายไว้ แต่ยังไม่ได้นำโปรแกรมและวงจรทั้งหมดมาทำงานรวมกัน ดังนั้นทางผู้จัดทำจึงนำโปรแกรมทั้งหมดและอุปกรณ์ทั้งหมดมาทำการทดลองรวมกัน ซึ่งผลการทดลองที่ได้สอดคล้องกันเป็นอย่างดี โดยคอมพิวเตอร์ A สามารถติดต่อกับคอมพิวเตอร์ B ที่ต่ออยู่กับอุปกรณ์ควบคุมการจ่ายไฟภายในห้องเรียนได้ จากนั้นคอมพิวเตอร์ B ส่งภาพเคลื่อนไหวมายังคอมพิวเตอร์ A แม้ว่าภาพจะมีความละเอียดไม่สูงนัก ทั้งนี้ขึ้นอยู่กับโครงข่ายโทรคมนาคมในปัจจุบัน และคอมพิวเตอร์ A สามารถสั่งเปิดและปิดอุปกรณ์ไฟฟ้าที่ต่ออยู่กับอุปกรณ์ควบคุมการจ่ายไฟได้ และเมื่อตั้งเวลาเปิดและปิดอุปกรณ์ไฟฟ้าอัตโนมัติตามตารางเรียนที่ติดตั้งโปรแกรมไว้ในเครื่องคอมพิวเตอร์ B ก็สามารถทำงานได้ตามคำสั่งที่ตั้งเวลาไว้ได้

นอกจากนี้ระบบควบคุมอุปกรณ์ไฟฟ้าและสังเกตการณ์สำหรับห้องเรียนนี้ยังสามารถพัฒนาต่อไปได้อีก เช่น ทำให้คอมพิวเตอร์ B สามารถบันทึกภาพเก็บไว้เมื่อเปิดคอมพิวเตอร์ เพื่อเปิดดูในภายหลังได้ เพิ่มอุปกรณ์ไฟฟ้าในการควบคุมให้เปิดและปิดทีละอย่างหรือเปิดและปิดพร้อมกันทีเดียว เป็นต้น

หนังสืออ้างอิง

[1] วรรณิกา เนตรงาม , "คู่มือการเขียนโปรแกรมภาษาจาวา" , บริษัท เอช เอ็น กรุ๊ป จำกัด ,2545

[2] ดร.วีระศักดิ์ ซึงถาวร , "JAVA Programming VolumeII" ,บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน) ,2548

[3] สุธี พงศาสกุลชัย ,ภาณุวัฒน์ บุญผาสุก , "คัมภีร์ JAVA เล่ม 2" , บริษัท เคทีพี คอมพ์ แอนด์คอล ซัลท์ จำกัด , 2548

[4] ดร.พิพัฒน์ พรหมมี "เอกสารประกอบการสอนวิชา Advanced Network Protocol" ภาควิชาวิศวกรรมโทรคมนาคม คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

[5] วรพจน์ กรแก้ววัฒนกุล ,ชัยวัฒน์ ลิ้มพรจิตรวิไล "เรียนรู้และปฏิบัติการไมโครคอนโทรลเลอร์ MCS-51 ฉบับ AT89C5x/AT89Sxxxx" บริษัท อินโนเวตีฟ เอ็กเพอริเมนต์ จำกัด (ปรับปรุงครั้งที่ 4)

[6] รศ.สมยศ จุณณะปิยะ "การประยุกต์ใช้งานไมโครคอนโทรลเลอร์" ภาควิชาวิศวกรรมโทรคมนาคม คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ภาคผนวก ก

//CommPort.java//

```
package sender;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;

import javax.comm.NoSuchPortException;
import javax.comm.PortInUseException;
import javax.comm.UnsupportedCommOperationException;
import javax.media.MediaLocator;

public class CommPort {

      public static void main(String[] ap) throws IOException,
NoSuchPortException, PortInUseException,
UnsupportedCommOperationException {
            CommPortOpen cp;
            VideoTransmit vt=null;
            String data = "Hello My name is CommPort";
            String host = "localhost";
            int port = 1234;
            boolean isfirst = false;
            try {
                  cp = new CommPortOpen();
                  cp.converse();
                  AlarmClock alarmClock = new AlarmClock(8, 14, 0);
                  alarmClock.setCommPort(cp);
                  alarmClock.start();
                  DesignTool designTool = new DesignTool();
                  // นำ server มาใส่เพื่อใช้ในการติดต่อกับอีกเครื่องหนึ่ง
                  while (true) {
                        ServerSocket serverSocket = new
                        ServerSocket(port);
                        Socket socket = serverSocket.accept();
                        data =
socket.toString().substring(socket.toString().indexOf("/")+1,socket.t
oString().indexOf(","));
                        host = data;

PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
                        data = cp.getUserName()+cp.getPassword();
                        out.print(data);
                        out.close();
                        socket.close();
                        serverSocket.close();
                        Socket skt = new Socket(host, 1234);
                        BufferedReader in = new BufferedReader(new
InputStreamReader(skt.getInputStream()));
                        while (!in.ready()) {
                        }
                        String inText = in.readLine();
                        in.close();
                        skt.close();
                        if
                  (inText.equals(cp.getUserName()+cp.getPassword()))
                        {

                              if (!isfirst) {
                                    //vfw://0
                                    System.out.println(host);
                                    vt = new VideoTransmit(new
                        MediaLocator("vfw://0"), host, "42050");
```

```
                        vt.start();
                    }
                    data = "connected";
                } else if (inText.equals("disconnect")) {
                    if (vt!=null) {
                        vt.stop();
                    }
                    data = "Hello My name is CommPort";
                }else if (inText.equals("on")){
                    cp.send("31h");
                    System.out.println(cp.expect("ON"));
                }else if (inText.equals("off")){
                    cp.send("0");
                    System.out.println(cp.expect("OFF"));
                }


                }
            } catch (Exception e) {
                System.err.println("You lose!");
                System.err.println(e);
            }
        }
        public CommPort() throws IOException, NoSuchPortException,
PortInUseException, UnsupportedCommOperationException {


        }
}
```

```
//CommPortOpen.java//

package sender;

import java.io.DataInputStream;
import java.io.IOException;
import java.io.PrintStream;

import javax.comm.CommPort;
import javax.comm.CommPortIdentifier;
import javax.comm.NoSuchPortException;
import javax.comm.PortInUseException;
import javax.comm.SerialPort;
import javax.comm.UnsupportedCommOperationException;

import com.sun.java_cup.internal.internal_error;
/**
 * ใช้สำหรับเปิด port serial เพื่อการติดต่อ
 *
 */
public class CommPortOpen {
        /** ใช้กำหนด timeout สำหรับการรอ */
        public static final int TIMEOUTSECONDS = 30;
        /** ใช้กำหนด baud rate ที่ใช้ */
        public static final int BAUD = 9600;
        /** The input stream */
        protected DataInputStream is;
        /** The output stream */
        protected PrintStream os;
        /** The last line read from the serial port. */
        protected String response;
        /** A flag to control debugging output. */
        protected boolean debug = true;
        /** port ที่เราเลือก */
        CommPortIdentifier thePortID;
        CommPort thePort;
        String userName = null;
```

```
        String password = null;

        /* Constructor */
        public CommPortOpen() throws IOException, NoSuchPortException,
PortInUseException, UnsupportedCommOperationException {
                // เลือก port ที่ต้องการจะใช้จาก PortChooserGUI
                PortChooserGUI chooser = new PortChooserGUI(null);
                String portName = null;

                do {
                        chooser.setVisible(true);
                        // get ชื่อ port ที่เลือกแล้ว
                        portName = chooser.getSelectedName();
                        userName = chooser.getUserName();
                        password = chooser.getPassword();
                        if (portName == null)
                                System.out.println("No port selected. Try
again.\n");
                } while (portName == null);
                // get port ที่เลือกออกมา
                thePortID = chooser.getSelectedIdentifier();
                // เริ่มทำการเปิด port
                // จะใช้ชื่อและ timeout
                //
                System.out.println("Trying to open " +
thePortID.getName() + "...");
                if (thePortID.getPortType() ==
CommPortIdentifier.PORT_SERIAL) {
                        thePort = thePortID.open("EDCMSC DataComm",
TIMEOUTSECONDS * 1000);
                        SerialPort myPort = (SerialPort) thePort;
                        // set up the serial port
                        myPort.setSerialPortParams(BAUD,
SerialPort.DATABITS_8, SerialPort.STOPBITS_1,
SerialPort.PARITY_NONE);
                }
                // Get the input and output streams ที่จะใช้ในการ read write
                // Printers can be write-only
                try {
                        is = new DataInputStream(thePort.getInputStream());
                } catch (IOException e) {
                        System.err.println("Can't open input stream: write-
only");
                        is = null;
                }
                os = new PrintStream(thePort.getOutputStream(), true);
        }

        protected void converse() throws IOException {
                System.out.println("Ready to read and write port.");
                System.out.println("Username : " + userName);
                System.out.println("Password : " + password);
                // Finally, clean up.
                //if (is != null)
                        //is.close();
                //os.close();
        }
        /* The public "getter" ใช้เรียก port ที่เลือกด้วย ชื่อ */
        public String getUserName() {
                return userName;
        }
        /* The public "getter" ใช้เรียก port ที่เลือกด้วย ชื่อ */
        public String getPassword() {
                return password;
        }
```

```
        /*
         * Send a line to a PC-style modem. Send \r\n, regardless of
what platform
         * we're on, instead of using println().
         *
         */
        protected void send(String s) throws IOException {
                if (debug) {
                        System.out.print(">>> ");
                        System.out.print(s);
                        System.out.println();
                }
                os.print(s);
                os.print("\r\n");
        }
        /*
         * Read a line, saving it in "response".
         */
        protected boolean expect(String exp) throws IOException {
                response = is.readLine();
                if (debug) {
                        System.out.print("<<< ");
                        System.out.print(response);
                        System.out.println();
                }
                return response.indexOf(exp) >= 0;
        }
        protected void closeIO()throws IOException{
                os.close();
                is.close();
        }
}
```

```
                        //PortChooserGUI.java//

package sender;

import java.awt.BorderLayout;
import java.awt.Container;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.util.Enumeration;
import java.util.HashMap;

import javax.comm.CommPortIdentifier;
import javax.comm.SerialPort;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JPasswordField;
import javax.swing.JTextField;
/*
 *  class นี้ใช้สร้าง GUI ที่จะใช้ในการเลือก port serial ที่เราต้องการ
 */
public class PortChooserGUI extends JDialog implements ItemListener {
        /* ใช้สำหรับ mapping ชื่อ port ของ gui กับ CommPortIdentifier(port จริง) */
        protected HashMap map = new HashMap();
        /* ใช้แทนชื่อ port ที่จะใช้ในการเลือก */
        protected String selectedPortName;
        /* ใช้แทน port จริงที่เลือก */
        protected CommPortIdentifier selectedPortIdentifier;
        /* ใช้แทนชื่อ list port สำหรับให้เลือก */
```

```
        protected JComboBox serialPortsChoice;
        /* serail port ที่เราจะนำไปใช้งานจริง(serail port object) */
        protected SerialPort ttya;

        public JTextField user;
        public JPasswordField pass;
        public String username = "";
        public String password = "";

        public void itemStateChanged(ItemEvent e) {
                // Get the name
                selectedPortName = (String) ((JComboBox)
e.getSource()).getSelectedItem();
                // Get the given CommPortIdentifier
                selectedPortIdentifier = (CommPortIdentifier)
map.get(selectedPortName);
                username = user.getText();
                password = pass.getText();
        }
        /* The public "getter" ใช้เรียก port ที่เลือกด้วย ชื่อ */
        public String getSelectedName() {
                return selectedPortName;
        }
        /* The public "getter" ใช้เรียก port ที่เลือกด้วย ชื่อ */
        public String getUserName() {
                return username;
        }
        /* The public "getter" ใช้เรียก port ที่เลือกด้วย ชื่อ */
        public String getPassword() {
                return password;
        }
        /* The public "getter" ใช้เรียก port ที่เลือกด้วย CommPortIdentifier */
        public CommPortIdentifier getSelectedIdentifier() {
                return selectedPortIdentifier;
        }

        /**
         * Construct a PortChooser -- ใช้เรียกส่วนสร้าง GUI และใส่ port
         * ที่มีเพื่อให้ user เลือก
         */
        public PortChooserGUI(JFrame parent) {
                super(parent, "EDCMSC-Port Chooser", true);
                makeGUI();
                populate();
                finishGUI();
        }
        /**
         * Method นี้ใช้สร้าง GUI
         */
        protected void makeGUI() {
                Container cp = getContentPane();
                JPanel centerPanel = new JPanel();
                cp.add(BorderLayout.CENTER, centerPanel);
                centerPanel.setLayout(new GridLayout(0, 2, 5, 5));
                centerPanel.add(new JLabel("User Name", JLabel.RIGHT));
                user = new JTextField();
                centerPanel.add(user);
                centerPanel.add(new JLabel("Password", JLabel.RIGHT));
                pass = new JPasswordField();
                centerPanel.add(pass);
                centerPanel.add(new JLabel("Serial Ports",
JLabel.RIGHT));
                serialPortsChoice = new JComboBox();
                centerPanel.add(serialPortsChoice);
                serialPortsChoice.setEnabled(true);
                JButton okButton;
```

```
            cp.add(BorderLayout.SOUTH, okButton = new JButton("OK"));
            okButton.addActionListener(new ActionListener() {
                  public void actionPerformed(ActionEvent e) {
                        PortChooserGUI.this.dispose();
                  }
            });
      }
      /**
       * ใช้ในการอ่าน port จริงแล้วใส่เข้าไปใน combobox
       */
      protected void populate() {

            // get list ของport ที่มีอยู่จริงในคอมที่รัน//
            Enumeration pList =
CommPortIdentifier.getPortIdentifiers();
            // Process the list, putting serial and parallel into
ComboBoxes//
            while (pList.hasMoreElements()) {
                  CommPortIdentifier cpi = (CommPortIdentifier)
pList.nextElement();
                  map.put(cpi.getName(), cpi);
                  if (cpi.getPortType() ==
CommPortIdentifier.PORT_SERIAL) {
                        serialPortsChoice.setEnabled(true);
                        serialPortsChoice.addItem(cpi.getName());
                  }
            }
            serialPortsChoice.setSelectedIndex(-1);
      }
      protected void finishGUI() {
            serialPortsChoice.addItemListener(this);
            pack();
      }
}
```

```
                        //VideoTransmit.java//

package sender;

import java.awt.*;
import javax.media.*;
import javax.media.protocol.*;
import javax.media.protocol.DataSource;
import javax.media.format.*;
import javax.media.control.TrackControl;
import javax.media.control.QualityControl;
import java.io.*;

public class VideoTransmit {

    // Input MediaLocator
    // Can be a file or http or capture source
    private MediaLocator locator;
    private String ipAddress;
    private String port;

    private Processor processor = null;
    private DataSink  rtptransmitter = null;
    private DataSource dataOutput = null;

    public VideoTransmit(MediaLocator locator,
                  String ipAddress,
                  String port) {

      this.locator = locator;
      this.ipAddress = ipAddress;
      this.port = port;
    }
```

```
/** Starts the transmission. Returns null if transmission started ok.
  * Otherwise it returns a string with the reason why the setup
failed.*/
    public synchronized String start() {
      String result;

      // Create a processor for the specified media locator
      // and program it to output JPEG/RTP
      result = createProcessor();
      if (result != null)
          return result;

      // Create an RTP session to transmit the output of the
      // processor to the specified IP address and port no.
      result = createTransmitter();
      if (result != null) {
          processor.close();
          processor = null;
          return result;
      }

      // Start the transmission
      processor.start();

      return null;
    }

    /**
     * Stops the transmission if already started
     */
    public void stop() {
      synchronized (this) {
          if (processor != null) {
            processor.stop();
            processor.close();
            processor = null;
            rtptransmitter.close();
            rtptransmitter = null;
          }
      }
    }

    private String createProcessor() {
      if (locator == null)
          return "Locator is null";

      DataSource ds;
      DataSource clone;

      try {
          ds = Manager.createDataSource(locator);
      } catch (Exception e) {
          return "Couldn't create DataSource";
      }

      // Try to create a processor to handle the input media locator
      try {
          processor = Manager.createProcessor(ds);
      } catch (NoProcessorException npe) {
          return "Couldn't create processor";
      } catch (IOException ioe) {
          return "IOException creating processor";
      }

      // Wait for it to configure
      boolean result = waitForState(processor, Processor.Configured);
      if (result == false)
          return "Couldn't configure processor";
```

```
        // Get the tracks from the processor
        TrackControl [] tracks = processor.getTrackControls();

        // Do we have atleast one track?
        if (tracks == null || tracks.length < 1)
            return "Couldn't find tracks in processor";

        boolean programmed = false;

        // Search through the tracks for a video track
        for (int i = 0; i < tracks.length; i++) {
            Format format = tracks[i].getFormat();
            if (  tracks[i].isEnabled() &&
                format instanceof VideoFormat &&
                !programmed) {

              // Found a video track. Try to program it to output
JPEG/RTP
              // Make sure the sizes are multiple of 8's.
              Dimension size = ((VideoFormat)format).getSize();
              float frameRate = ((VideoFormat)format).getFrameRate();
              int w = (size.width % 8 == 0 ? size.width :
                        (int)(size.width / 8) * 8);
              int h = (size.height % 8 == 0 ? size.height :
                        (int)(size.height / 8) * 8);
              VideoFormat jpegFormat = new
VideoFormat(VideoFormat.JPEG_RTP,
                                          new Dimension(w, h),
                                          Format.NOT_SPECIFIED,
                                          Format.byteArray,
                                          frameRate);
              tracks[i].setFormat(jpegFormat);
              System.err.println("Video transmitted as:");
              System.err.println("  " + jpegFormat);
              // Assume succesful
              programmed = true;
            } else
              tracks[i].setEnabled(false);
        }

        if (!programmed)
            return "Couldn't find video track";

        // Set the output content descriptor to RAW_RTP
        ContentDescriptor cd = new
ContentDescriptor(ContentDescriptor.RAW_RTP);
        processor.setContentDescriptor(cd);

        // Realize the processor. This will internally create a flow
        // graph and attempt to create an output datasource for
JPEG/RTP
        // video frames.
        result = waitForState(processor, Controller.Realized);
        if (result == false)
            return "Couldn't realize processor";

        // Set the JPEG quality to .5.
        setJPEGQuality(processor, 0.5f);

        // Get the output data source of the processor
        dataOutput = processor.getDataOutput();
        return null;
    }

    // Creates an RTP transmit data sink. This is the easiest way to
create
    // an RTP transmitter. The other way is to use the
RTPSessionManager API.
```

```
    // Using an RTP session manager gives you more control if you
wish to
    // fine tune your transmission and set other parameters.
    private String createTransmitter() {
      // Create a media locator for the RTP data sink.
      // For example:
      //    rtp://129.130.131.132:42050/video
      String rtpURL = "rtp://" + ipAddress + ":" + port + "/video";
      MediaLocator outputLocator = new MediaLocator(rtpURL);

      // Create a data sink, open it and start transmission. It will
wait
      // for the processor to start sending data. So we need to start
the
      // output data source of the processor. We also need to start
the
      // processor itself, which is done after this method returns.
      try {
          rtptransmitter = Manager.createDataSink(dataOutput,
outputLocator);
          //datasink = Manager.createDataSink(dataOutput, ml);
          rtptransmitter.open();
          rtptransmitter.start();
          dataOutput.start();
      } catch (MediaException me) {
          return "Couldn't create RTP data sink";
      } catch (IOException ioe) {
          return "Couldn't create RTP data sink";
      }
      try {


      } catch (Exception e) {
          System.err.println(e);
      }
      return null;
    }


    /**
     * Setting the encoding quality to the specified value on the
JPEG encoder.
     * 0.5 is a good default.
     */
    void setJPEGQuality(Player p, float val) {

      Control cs[] = p.getControls();
      QualityControl qc = null;
      VideoFormat jpegFmt = new VideoFormat(VideoFormat.JPEG);

      // Loop through the controls to find the Quality control for
      // the JPEG encoder.
      for (int i = 0; i < cs.length; i++) {

          if (cs[i] instanceof QualityControl &&
            cs[i] instanceof Owned) {
            Object owner = ((Owned)cs[i]).getOwner();

            // Check to see if the owner is a Codec.
            // Then check for the output format.
            if (owner instanceof Codec) {
                Format fmts[] =
((Codec)owner).getSupportedOutputFormats(null);
                for (int j = 0; j < fmts.length; j++) {
                  if (fmts[j].matches(jpegFmt)) {
                      qc = (QualityControl)cs[i];
                      qc.setQuality(val);
                      System.err.println("- Setting quality to " +
```

```
                                    val + " on " + qc);
                            break;
                        }
                    }
                }
                if (qc != null)
                    break;
            }
        }
    }

    /**********************************************************************
     * Convenience methods to handle processor's state changes.
**********************************************************************/

    private Integer stateLock = new Integer(0);
    private boolean failed = false;

    Integer getStateLock() {
      return stateLock;
    }

    void setFailed() {
      failed = true;
    }

    private synchronized boolean waitForState(Processor p, int state)
{
      p.addControllerListener(new StateListener());
      failed = false;

      // Call the required method on the processor
      if (state == Processor.Configured) {
          p.configure();
      } else if (state == Processor.Realized) {
          p.realize();
      }

      // Wait until we get an event that confirms the
      // success of the method, or a failure event.
      // See StateListener inner class
      while (p.getState() < state && !failed) {
          synchronized (getStateLock()) {
            try {
                getStateLock().wait();
            } catch (InterruptedException ie) {
                return false;
            }
          }
      }

      if (failed)
          return false;
      else
          return true;
    }

    /**********************************************************************
     * Inner Classes
**********************************************************************/

    class StateListener implements ControllerListener {

      public void controllerUpdate(ControllerEvent ce) {

          // If there was an error during configure or
```

```
            // realize, the processor will be closed
            if (ce instanceof ControllerClosedEvent)
              setFailed();

            // All controller events, send a notification
            // to the waiting thread in waitForState method.
            if (ce instanceof ControllerEvent) {
              synchronized (getStateLock()) {
                  getStateLock().notifyAll();
              }
            }
        }
    }


    /************************************************************************
     * Sample Usage for VideoTransmit class
*************************************************************************/

    public static void main(String [] args) {
      // We need three parameters to do the transmission
      // For example,
      //   java VideoTransmit file:/C:/media/test.mov
129.130.131.132 42050

      if (args.length < 3) {
          System.err.println("Usage: VideoTransmit <sourceURL>
<destIP> <destPort>");
          System.exit(-1);
      }

      // Create a video transmit object with the specified params.
      VideoTransmit vt = new VideoTransmit(new MediaLocator(args[0]),
                                           args[1],
                                           args[2]);
      // Start the transmission
      String result = vt.start();

      // result will be non-null if there was an error. The return
      // value is a String describing the possible error. Print it.
      if (result != null) {
          System.err.println("Error : " + result);
          System.exit(0);
      }

      System.err.println("Start transmission for 60 seconds...");

      // Transmit for 60 seconds and then close the processor
      // This is a safeguard when using a capture data source
      // so that the capture device will be properly released
      // before quitting.
      // The right thing to do would be to have a GUI with a
      // "Stop" button that would call stop on VideoTransmit
      try {
          Thread.currentThread().sleep(60000);
      } catch (InterruptedException ie) {
      }

      // Stop the transmission
      vt.stop();

      System.err.println("...transmission ended.");

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

```
                                // DesignTool.java//
package sender;

import java.awt.Color;
import java.awt.GridLayout;
import java.awt.Label;
import java.awt.TextField;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Date;

import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextField;

public class DesignTool extends JFrame {
      public static PrintWriter pw;
      public DesignTool() {

            super("Design Tool");
            getContentPane().add(new CADPaintPanel());
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            setSize(400, 300);
            setVisible(true);
      }
      public static void main(String[] args) {
            new DesignTool();
      }

      protected class CADPaintPanel extends JPanel implements
ActionListener {
            // ประกาศ object ที่จะใช้เป็นปุ่มทั้งสองปุ่ม
            private JButton genButton = new JButton("Add");
            private JButton clearButton = new JButton("save");
            private String operatorName = "";
            private String dayName = "";

            /*
             * ประกาศตัวแปรที่จะใช้แสดงผลลัพธ์ของ code ที่ได้จากการ Generate
             * โดยจะใช้เป็น TextField ขนาด 30 ตัวอักษร
             */
            private TextField text1 = new TextField(30);
            private TextField text2 = new TextField(30);
            private TextField text3 = new TextField(30);
            private TextField text4 = new TextField(30);
            private TextField text5 = new TextField(30);
            private TextField text6 = new TextField(30);
            private TextField text7 = new TextField(30);
            private TextField text8 = new TextField(30);
            private TextField text9 = new TextField(30);
            private TextField text10 = new TextField(30);

            /* Labels for operation and color fields */
            private Label label1 = new Label(" 8.00 - 9.00 :",
Label.CENTER);
            private Label label2 = new Label(" 9.00 - 10.00 :",
Label.CENTER);
            private Label label3 = new Label(" 10.00 - 11.00 :",
Label.CENTER);
```

```
            private Label label4 = new Label(" 11.00 - 12.00 :",
Label.CENTER);
            private Label label5 = new Label(" 12.00 - 13.00 :",
Label.CENTER);
            private Label label6 = new Label(" 13.00 - 14.00 :",
Label.CENTER);
            private Label label7 = new Label(" 14.00 - 15.00 :",
Label.CENTER);
            private Label label8 = new Label(" 15.00 - 16.00 :",
Label.CENTER);
            private Label label9 = new Label(" 16.00 - 17.00 :",
Label.CENTER);
            private Label label10 = new Label(" 17.00 - 18.00 :",
Label.CENTER);
            private Label controlLable = new Label("");
            private Label controlLable2 = new Label("");
            private Label inputlLable = new Label("Subject Name",
Label.CENTER);
            private Label inputlLable2 = new Label("Day",
Label.CENTER);
            private Label operationLable = new Label("Time",
Label.CENTER);

            // ประกาศ array ของตัวอักษรที่จะให้เลือก operation และ input
            String[] operationArray = {"8.00 - 9.00", "9.00 - 10.00",
"10.00 - 11.00", "11.00 - 12.00", "12.00 - 13.00", "13.00 - 14.00",
"14.00 - 15.00", "15.00 - 16.00", "16.00 - 17.00", "17.00 - 18.00"};
            String[] dayArray = {"sun",
"mon","tue","wed","thu","fri","sat"};
            // ประกาศ ComboBox ที่จะให้สำหรับให้ ผู้ใช้เลือก operation และ input
            private JTextField input = new JTextField("");
            private JComboBox operator = new
JComboBox(operationArray);
            private JComboBox day = new JComboBox(dayArray);
            //
            public CADPaintPanel() {

                setLayout(new GridLayout(15, 2));
                /* Add label and color text field */
                add(label1);
                add(text1);

                /* Add label and operation text field */
                add(label2);
                add(text2);

                /* Add label and cursor text field */
                add(label3);
                add(text3);

                add(label4);
                add(text4);

                add(label5);
                add(text5);



                add(label6);
                add(text6);

                add(label7);
                add(text7);

                add(label8);
                add(text8);
                add(label9);
                add(text9);
                add(label10);
```

```
                    add(text10);

                    operator.setSelectedIndex(-1);
                    Date date = new Date();
                    day.setSelectedIndex(date.getDay());
                    controlLable.setBackground(new Color(100, 100,
100));
                    controlLable2.setBackground(new Color(100, 100,
100));
                    add(controlLable);
                    add(controlLable2);
                    add(inputlLable);
                    add(input);
                    add(operationLable);
                    add(operator);
                    add(inputlLable2);
                    add(day);
                    add(genButton);
                    add(clearButton);

                    /* Setup action listener */
                    clearButton.addActionListener(this);
                    genButton.addActionListener(this);
                    input.addActionListener(this);
                    operator.addActionListener(this);
                    day.addActionListener(this);
            }
      private void clearText(){

                    text1.setText("");
                    text2.setText("");
                    text3.setText("");
                    text4.setText("");
                    text5.setText("");
                    text6.setText("");
                    text7.setText("");
                    text8.setText("");
                    text9.setText("");
                    text10.setText("");

      }

            private void setText(){
                    try {
                            FileReader fis = new FileReader("subject-
"+dayName+".txt");
                            BufferedReader br = new BufferedReader(fis);

                            String pref = br.readLine();
                            text1.setText(pref);
                            pref = br.readLine();
                            text2.setText(pref);
                            pref = br.readLine();
                            text3.setText(pref);
                            pref = br.readLine();
                            text4.setText(pref);
                            pref = br.readLine();
                            text5.setText(pref);
                            pref = br.readLine();
                            text6.setText(pref);
                            pref = br.readLine();
                            text7.setText(pref);
                            pref = br.readLine();
                            text8.setText(pref);
                            pref = br.readLine();
                            text9.setText(pref);
                            pref = br.readLine();
                            text10.setText(pref);
                            br.close();
```

```
                } catch (Exception IOE) {
                }
            }
            /*
             * method นี้เป็น method ที่ใช้ประมวลผลของการ click mouse (action event)
             * โดยจะนิยามไว้ตาม ActionListener ที่ได้มีการ add ไว้ในส่วนแสดงผล
             *
             */
            public void actionPerformed(ActionEvent e) {
                // นิยามไว้เพื่อรองรับการกด operator combobox
                if (e.getSource() == operator) {
                    operatorName = (String) ((JComboBox)
e.getSource()).getSelectedItem();
                }
                if (e.getSource() == day) {
                    dayName = (String) ((JComboBox)
e.getSource()).getSelectedItem();
                    clearText();
                    setText();
                }
                // นิยามไว้เพื่อรองรับการกด ปุ่ม add
                if (e.getSource() == genButton) {
                    System.out.println("Add record");
                    if (!input.getText().equals("")) {
                        if(operatorName.equals("8.00 - 9.00")){
                            text1.setText(input.getText());
                        }

if(operatorName.equals("9.00 - 10.00")){
text2.setText(input.getText());
                        }

if(operatorName.equals("10.00 - 11.00")){
text3.setText(input.getText());
                        }

if(operatorName.equals("11.00 - 12.00")){
text4.setText(input.getText());
                        }

if(operatorName.equals("12.00 - 13.00")){
text5.setText(input.getText());
                        }

if(operatorName.equals("13.00 - 14.00")){
text6.setText(input.getText());
                        }

if(operatorName.equals("14.00 - 15.00")){
text7.setText(input.getText());
                        }

if(operatorName.equals("15.00 - 16.00")){
text8.setText(input.getText());
                        }

if(operatorName.equals("16.00 - 17.00")){
text9.setText(input.getText());
                        }

if(operatorName.equals("17.00 - 18.00")){
text10.setText(input.getText());
                        }

                    }

                }
```

```
                    // นิยามไว้เพื่อรองรับการกด ปุ่ม save
                    if (e.getSource() == clearButton) {
                          operator.setSelectedIndex(-1);
                          //day.setSelectedIndex(-1);
                          System.out.println("Save");

                          try {
                                pw = new PrintWriter(new
FileWriter("timer-"+dayName+".txt"));
                                if (text1.getText().equals("")) {
                                      pw.println(0);
                                }else {
                                      pw.println(1);
                                }
                                if (text2.getText().equals("")) {
                                      pw.println(0);
                                }else {
                                      pw.println(1);
                                }
                                if (text3.getText().equals("")) {
                                      pw.println(0);
                                }else {
                                      pw.println(1);
                                }
                                if (text4.getText().equals("")) {
                                      pw.println(0);
                                }else {
                                      pw.println(1);
                                }
                                if (text5.getText().equals("")) {
                                      pw.println(0);
                                }else {
                                      pw.println(1);
                                }
                                if (text6.getText().equals("")) {
                                      pw.println(0);
                                }else {
                                      pw.println(1);
                                }
                                if (text7.getText().equals("")) {
                                      pw.println(0);
                                }else {
                                      pw.println(1);
                                }
                                if (text8.getText().equals("")) {
                                      pw.println(0);
                                }else {
                                      pw.println(1);
                                }
                                if (text9.getText().equals("")) {
                                      pw.println(0);
                                }else {
                                      pw.println(1);
                                }
                                if (text10.getText().equals("")) {
                                      pw.println(0);
                                }else {
                                      pw.println(1);
                                }

                                pw.flush();
                                pw.close();
                          } catch (IOException e1) {
                                JOptionPane.showMessageDialog(null,
"ERROR", "ERROR", JOptionPane.ERROR_MESSAGE);
                                e1.printStackTrace();
                          }
                          try {
```

```
                                    pw = new PrintWriter(new
FileWriter("subject-"+dayName+".txt"));
                                    pw.println(text1.getText());
                                    pw.println(text2.getText());
                                    pw.println(text3.getText());
                                    pw.println(text4.getText());
                                    pw.println(text5.getText());
                                    pw.println(text6.getText());
                                    pw.println(text6.getText());
                                    pw.println(text7.getText());
                                    pw.println(text8.getText());
                                    pw.println(text9.getText());
                                    pw.println(text10.getText());

                                    pw.flush();
                                    pw.close();
                              } catch (IOException e1) {
                                    JOptionPane.showMessageDialog(null,
"ERROR", "ERROR", JOptionPane.ERROR_MESSAGE);
                                    e1.printStackTrace();
                              }

                        }

                  }
            }


}

                              //Alarmclock.java//

package sender;

import java.io.BufferedReader;
import java.io.FileReader;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import java.util.Timer;
import java.util.TimerTask;

public class AlarmClock {
      private final Scheduler scheduler = new Scheduler();
      private final SimpleDateFormat dateFormat = new
SimpleDateFormat("dd MMM yyyy HH:mm:ss.SSS");
      private final int hourOfDay, minute, second;
      public CommPortOpen cp;
      String[] dayArray = {"sun",
"mon","tue","wed","thu","fri","sat"};
      Date date = new Date();

      public AlarmClock(int hourOfDay, int minute, int second) {
            this.hourOfDay = hourOfDay;
            this.minute = minute;
            this.second = second;
      }

      public void start() {
            scheduler.schedule(new SchedulerTask() {
                  public void run() {
                        soundAlarm();
                  }
                  private void soundAlarm() {
                        System.out.println("Wake up! " + "It's " +
dateFormat.format(new Date()));
                        try {
                              cp.send("0");
```

```
                                    System.out.println(cp.expect("OFF"));
                                    String[] timeTest = new String[10];
                                    String pref;
                                    try {
                                          FileReader fis = new
FileReader("subject"+dayArray[date.getDay()]+".txt");
                                          BufferedReader br = new
BufferedReader(fis);


                                          for (int i = 0; i <
timeTest.length; i++) {
                                                pref = br.readLine();
                                                timeTest[i] = pref;
                                          }
                                          br.close();
                                    } catch (Exception IOE) {
                                    }
                                    Date date = new Date();

      System.out.println(timeTest[date.getHours()-8]);
                                    if (timeTest[date.getHours()-8]!="") {
                                          cp.send("1");

      System.out.println(cp.expect("ON"));
                                    }
                                    /*else{ cp.send("0");
                                    System.out.println(cp.expect("OFF"));
                                    }*/

                              } catch (Exception e) {
                                    System.err.println("You lose!");
                                    System.err.println(e);
                              }
                              // Start a new thread to sound an alarm...
                        }
                  }, new DailyIterator(hourOfDay, minute, second));
      }
      public void setCommPort(CommPortOpen cp) {
            this.cp = cp;
      }

      public static void main(String[] args) {
            AlarmClock alarmClock = new AlarmClock(8, 0, 0);
            alarmClock.start();
      }

}

abstract class SchedulerTask implements Runnable {

      final Object lock = new Object();

      int state = VIRGIN;
      static final int VIRGIN = 0;
      static final int SCHEDULED = 1;
      static final int CANCELLED = 2;

      TimerTask timerTask;

      protected SchedulerTask() {
      }

      public abstract void run();

      public boolean cancel() {
            synchronized (lock) {
                  if (timerTask != null) {
                        timerTask.cancel();
```

```
                }
                boolean result = (state == SCHEDULED);
                state = CANCELLED;
                return result;
            }
        }

        public long scheduledExecutionTime() {
            synchronized (lock) {
                return timerTask == null ? 0 :
timerTask.scheduledExecutionTime();
            }
        }

}

class Scheduler {

        class SchedulerTimerTask extends TimerTask {
            private SchedulerTask schedulerTask;
            private ScheduleIterator iterator;
            public SchedulerTimerTask(SchedulerTask schedulerTask,
ScheduleIterator iterator) {
                this.schedulerTask = schedulerTask;
                this.iterator = iterator;
            }
            public void run() {
                schedulerTask.run();
                reschedule(schedulerTask, iterator);
            }
        }

        private final Timer timer = new Timer();

        public Scheduler() {
        }

        public void cancel() {
            timer.cancel();
        }

        public void schedule(SchedulerTask schedulerTask,
ScheduleIterator iterator) {

            Date time = iterator.next();
            if (time == null) {
                schedulerTask.cancel();
            } else {
                synchronized (schedulerTask.lock) {
                    if (schedulerTask.state !=
SchedulerTask.VIRGIN) {
                        throw new IllegalStateException("Task
already scheduled " + "or cancelled");
                    }
                    schedulerTask.state =
SchedulerTask.SCHEDULED;
                    schedulerTask.timerTask = new
SchedulerTimerTask(schedulerTask, iterator);
                    timer.schedule(schedulerTask.timerTask,
time);
                }
            }
        }

        private void reschedule(SchedulerTask schedulerTask,
ScheduleIterator iterator) {

            Date time = iterator.next();
            if (time == null) {
```

```
                    schedulerTask.cancel();
                } else {
                    synchronized (schedulerTask.lock) {
                        if (schedulerTask.state !=
SchedulerTask.CANCELLED) {
                            schedulerTask.timerTask = new
SchedulerTimerTask(schedulerTask, iterator);
                            timer.schedule(schedulerTask.timerTask,
time);
                        }
                    }
                }
            }

}
class DailyIterator implements ScheduleIterator {
        private final int hourOfDay, minute, second;
        private final Calendar calendar = Calendar.getInstance();

        public DailyIterator(int hourOfDay, int minute, int second) {
                this(hourOfDay, minute, second, new Date());
        }

        public DailyIterator(int hourOfDay, int minute, int second,
Date date) {
                this.hourOfDay = hourOfDay;
                this.minute = minute;
                this.second = second;
                calendar.setTime(date);
                calendar.set(Calendar.HOUR_OF_DAY, hourOfDay);
                calendar.set(Calendar.MINUTE, minute);
                calendar.set(Calendar.SECOND, second);
                calendar.set(Calendar.MILLISECOND, 0);
                if (!calendar.getTime().before(date)) {
                    calendar.add(Calendar.HOUR, -1);
                }
        }

        public Date next() {
                calendar.add(Calendar.HOUR, 1);
                return calendar.getTime();
        }

}

interface ScheduleIterator {
        public Date next();
}
```

//MonitoringSystem.java//

```
package reciever;

import java.io.*;
import java.awt.*;
import java.awt.event.*;
import java.net.*;
import java.util.*;

import javax.media.rtp.*;

import javax.swing.*;
import javax.swing.event.*;
import javax.swing.border.*;

public class MonitoringSystem extends JFrame implements
ActionListener, KeyListener, MouseListener, WindowListener {
      Vector targets;
      JList list;
      JButton connect;
      JButton disconnect;
      JButton on;
      JButton off;
      JButton rtcp;
      JButton startRx;
      JButton expiration;
      JButton statistics;
      JButton addTarget;
      JButton removeTarget;
      JTextField tf_remote_address;
      JTextField tf_remote_data_port;
      JTextField tf_local_data_port;
      JTextField tf_media_file;
      JTextField tf_taget_ip;
      JTextField tf_taget_port;
      JTextField tf_username;
      JPasswordField tf_password;
      TargetListModel listModel;
      JCheckBox cb_loop;
      Config config;
      AVReceiver avReceiver;
      JPanel p1;
      JPanel p2;
      JPanel buttonPanel;
      JPanel targetPanel;

      public boolean isConnected = false;

      public MonitoringSystem() {
            setTitle("Monitoring System Receiver");
            config = new Config();
            GridBagLayout gridBagLayout = new GridBagLayout();
            GridBagConstraints gbc;
            p1 = new JPanel();
            p1.setLayout(gridBagLayout);
            JPanel localPanel = createLocalPanel();
            gbc = new GridBagConstraints();
            gbc.gridx = 0;
            gbc.gridy = 0;
            gbc.gridwidth = 2;
            gbc.anchor = GridBagConstraints.CENTER;
            gbc.fill = GridBagConstraints.BOTH;
            gbc.insets = new Insets(10, 5, 0, 0);
            ((GridBagLayout)
p1.getLayout()).setConstraints(localPanel, gbc);
            p1.add(localPanel);
```

```
            targetPanel = createTargetPanel();

            gbc = new GridBagConstraints();
            gbc.gridx = 1;
            gbc.gridy = 1;
            gbc.weightx = 1.0;
            gbc.weighty = 1.0;
            gbc.anchor = GridBagConstraints.CENTER;
            gbc.fill = GridBagConstraints.BOTH;
            gbc.insets = new Insets(10, 5, 0, 0);
            ((GridBagLayout)
p1.getLayout()).setConstraints(targetPanel, gbc);
            // p.add(targetPanel);

            buttonPanel = new JPanel();
            connect = new JButton("Connect");
            disconnect = new JButton("Disconnect");
            on = new JButton("ON");
            off = new JButton("OFF");
            rtcp = new JButton("RTCP Monitor");
            startRx = new JButton("Start Receiver");

            connect.addActionListener(this);
            disconnect.addActionListener(this);
            on.addActionListener(this);
            off.addActionListener(this);
            rtcp.addActionListener(this);
            startRx.addActionListener(this);
            buttonPanel.add(connect);
            //buttonPanel.add(rtcp);
            //buttonPanel.add(startRx);

            gbc = new GridBagConstraints();
            gbc.gridx = 0;
            gbc.gridy = 3;
            gbc.gridwidth = 2;
            gbc.weightx = 1.0;
            gbc.weighty = 0.0;
            gbc.anchor = GridBagConstraints.CENTER;
            gbc.fill = GridBagConstraints.HORIZONTAL;
            gbc.insets = new Insets(5, 5, 10, 5);
            ((GridBagLayout)
p1.getLayout()).setConstraints(buttonPanel, gbc);
            p1.add(buttonPanel);

            getContentPane().add(p1);

            list.addMouseListener(this);

            addWindowListener(this);

            pack();

            setVisible(true);
      }

      private JPanel createTargetPanel() {
            JPanel p = new JPanel();

            GridBagLayout gridBagLayout = new GridBagLayout();

            GridBagConstraints gbc;

            p.setLayout(gridBagLayout);

            targets = config.targets;

            listModel = new TargetListModel(targets);
```

```
            list = new JList(listModel);

            list.addKeyListener(this);


      list.setPrototypeCellValue("xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx");

            JScrollPane scrollPane = new JScrollPane(list,
ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED,
ScrollPaneConstants.HORIZONTAL_SCROLLBAR_NEVER);

            gbc = new GridBagConstraints();
            gbc.gridx = 0;
            gbc.gridy = 0;
            gbc.weightx = 1.0;
            gbc.weighty = 1.0;
            gbc.anchor = GridBagConstraints.CENTER;
            gbc.fill = GridBagConstraints.BOTH;
            gbc.insets = new Insets(10, 5, 0, 0);
            ((GridBagLayout)
p.getLayout()).setConstraints(scrollPane, gbc);
            p.add(scrollPane);

            JPanel p1 = new JPanel();

            p1.setLayout(gridBagLayout);

            JLabel label = new JLabel("Sender IP:");

            gbc = new GridBagConstraints();
            gbc.gridx = 0;
            gbc.gridy = 0;
            gbc.weightx = 0.0;
            gbc.weighty = 0.0;
            gbc.anchor = GridBagConstraints.EAST;
            gbc.fill = GridBagConstraints.NONE;
            gbc.insets = new Insets(5, 5, 0, 5);
            ((GridBagLayout) p1.getLayout()).setConstraints(label,
gbc);
            p1.add(label);

            tf_remote_address = new JTextField(15);

            gbc = new GridBagConstraints();
            gbc.gridx = 1;
            gbc.gridy = 0;
            gbc.weightx = 0.0;
            gbc.weighty = 0.0;
            gbc.anchor = GridBagConstraints.WEST;
            gbc.fill = GridBagConstraints.NONE;
            gbc.insets = new Insets(5, 5, 0, 5);
            ((GridBagLayout)
p1.getLayout()).setConstraints(tf_remote_address, gbc);
            p1.add(tf_remote_address);

            label = new JLabel("Sender Port:");

            gbc = new GridBagConstraints();
            gbc.gridx = 0;
            gbc.gridy = 1;
            gbc.weightx = 0.0;
            gbc.weighty = 0.0;
            gbc.anchor = GridBagConstraints.EAST;
            gbc.fill = GridBagConstraints.NONE;
            gbc.insets = new Insets(5, 5, 0, 5);
            ((GridBagLayout) p1.getLayout()).setConstraints(label,
gbc);
            p1.add(label);
```

```
            tf_remote_data_port = new JTextField(15);

            gbc = new GridBagConstraints();
            gbc.gridx = 1;
            gbc.gridy = 1;
            gbc.weightx = 0.0;
            gbc.weighty = 0.0;
            gbc.anchor = GridBagConstraints.WEST;
            gbc.fill = GridBagConstraints.NONE;
            gbc.insets = new Insets(5, 5, 0, 5);
            ((GridBagLayout)
p1.getLayout()).setConstraints(tf_remote_data_port, gbc);
            p1.add(tf_remote_data_port);

            label = new JLabel("Local Port:");

            gbc = new GridBagConstraints();
            gbc.gridx = 0;
            gbc.gridy = 2;
            gbc.weightx = 0.0;
            gbc.weighty = 0.0;
            gbc.anchor = GridBagConstraints.EAST;
            gbc.fill = GridBagConstraints.NONE;
            gbc.insets = new Insets(5, 5, 0, 5);
            ((GridBagLayout) p1.getLayout()).setConstraints(label,
gbc);
            p1.add(label);

            tf_local_data_port = new JTextField(15);

            gbc = new GridBagConstraints();
            gbc.gridx = 1;
            gbc.gridy = 2;
            gbc.weightx = 0.0;
            gbc.weighty = 0.0;
            gbc.anchor = GridBagConstraints.WEST;
            gbc.fill = GridBagConstraints.NONE;
            gbc.insets = new Insets(5, 5, 0, 5);
            ((GridBagLayout)
p1.getLayout()).setConstraints(tf_local_data_port, gbc);
            p1.add(tf_local_data_port);

            JPanel p2 = new JPanel();

            addTarget = new JButton("Add Target");
            removeTarget = new JButton("Remove Target");

            p2.add(addTarget);
            p2.add(removeTarget);

            addTarget.addActionListener(this);
            removeTarget.addActionListener(this);

            gbc = new GridBagConstraints();
            gbc.gridx = 0;
            gbc.gridy = 3;
            gbc.weightx = 1.0;
            gbc.weighty = 0.0;
            gbc.gridwidth = 2;
            gbc.anchor = GridBagConstraints.CENTER;
            gbc.fill = GridBagConstraints.HORIZONTAL;
            gbc.insets = new Insets(20, 5, 0, 5);
            ((GridBagLayout) p1.getLayout()).setConstraints(p2, gbc);
            p1.add(p2);

            gbc = new GridBagConstraints();
            gbc.gridx = 1;
            gbc.gridy = 0;
```

```
            gbc.weightx = 1.0;
            gbc.weighty = 1.0;
            gbc.anchor = GridBagConstraints.CENTER;
            gbc.fill = GridBagConstraints.BOTH;
            gbc.insets = new Insets(10, 5, 0, 0);
            ((GridBagLayout) p.getLayout()).setConstraints(p1, gbc);
            p.add(p1);

            TitledBorder titledBorder = new TitledBorder(new
EtchedBorder(), "Targets");

            p.setBorder(titledBorder);

            if (targets.size() > 0) {
                  removeTarget.setEnabled(true);
            } else {
                  removeTarget.setEnabled(false);
            }

            return p;
      }

      /*
       * method นี้ใช้เพื่อสร้าง local panel ซึ่งจะประกอบไปด้วย 1. local ip 2.
       * username 3.password 4. ปุ่ม connect
       */
      private JPanel createLocalPanel() {
            p2 = new JPanel();

            GridBagLayout gridBagLayout = new GridBagLayout();

            GridBagConstraints gbc;

            p2.setLayout(gridBagLayout);
            // สร้าง local ip และ add ลง panel
            JLabel label = new JLabel("Local IP Address:");

            gbc = new GridBagConstraints();
            gbc.gridx = 0;
            gbc.gridy = 0;
            gbc.weightx = 0.0;
            gbc.weighty = 0.0;
            gbc.anchor = GridBagConstraints.EAST;
            gbc.fill = GridBagConstraints.NONE;
            gbc.insets = new Insets(5, 5, 0, 5);
            ((GridBagLayout) p2.getLayout()).setConstraints(label,
gbc);
            p2.add(label);

            JLabel tf_local_host = new JLabel();

            gbc = new GridBagConstraints();
            gbc.gridx = 1;
            gbc.gridy = 0;
            gbc.weightx = 0.0;
            gbc.weighty = 0.0;
            gbc.anchor = GridBagConstraints.WEST;
            gbc.fill = GridBagConstraints.NONE;
            gbc.insets = new Insets(5, 5, 10, 5);
            ((GridBagLayout)
p2.getLayout()).setConstraints(tf_local_host, gbc);
            p2.add(tf_local_host);

            // สร้าง Username และ add ลง panel
            JLabel label2 = new JLabel("Username:");

            gbc = new GridBagConstraints();
            gbc.gridx = 0;
```

```
            gbc.gridy = 4;
            gbc.weightx = 0.0;
            gbc.weighty = 0.0;
            gbc.anchor = GridBagConstraints.EAST;
            gbc.fill = GridBagConstraints.NONE;
            gbc.insets = new Insets(5, 5, 0, 5);
            ((GridBagLayout) p2.getLayout()).setConstraints(label2,
gbc);
            p2.add(label2);

            tf_username = new JTextField(15);

            gbc = new GridBagConstraints();
            gbc.gridx = 1;
            gbc.gridy = 4;
            gbc.weightx = 0.0;
            gbc.weighty = 0.0;
            gbc.anchor = GridBagConstraints.WEST;
            gbc.fill = GridBagConstraints.NONE;
            gbc.insets = new Insets(5, 5, 10, 5);
            ((GridBagLayout)
p2.getLayout()).setConstraints(tf_username, gbc);
            p2.add(tf_username);
            // สร้าง Password และ add ลง panel
            JLabel label3 = new JLabel("Password:");

            gbc = new GridBagConstraints();
            gbc.gridx = 0;
            gbc.gridy = 5;
            gbc.weightx = 0.0;
            gbc.weighty = 0.0;
            gbc.anchor = GridBagConstraints.EAST;
            gbc.fill = GridBagConstraints.NONE;
            gbc.insets = new Insets(5, 5, 0, 5);
            ((GridBagLayout) p2.getLayout()).setConstraints(label3,
gbc);
            p2.add(label3);

            tf_password = new JPasswordField(15);

            gbc = new GridBagConstraints();
            gbc.gridx = 1;
            gbc.gridy = 5;
            gbc.weightx = 0.0;
            gbc.weighty = 0.0;
            gbc.anchor = GridBagConstraints.WEST;
            gbc.fill = GridBagConstraints.NONE;
            gbc.insets = new Insets(5, 5, 10, 5);
            ((GridBagLayout)
p2.getLayout()).setConstraints(tf_password, gbc);
            p2.add(tf_password);

            // สร้าง Taget ip และ add ลง panel
            JLabel label4 = new JLabel("Taget IP Address:");

            gbc = new GridBagConstraints();
            gbc.gridx = 0;
            gbc.gridy = 2;
            gbc.weightx = 0.0;
            gbc.weighty = 0.0;
            gbc.anchor = GridBagConstraints.EAST;
            gbc.fill = GridBagConstraints.NONE;
            gbc.insets = new Insets(5, 5, 0, 5);
            ((GridBagLayout) p2.getLayout()).setConstraints(label4,
gbc);
            p2.add(label4);

            tf_taget_ip = new JTextField(15);
```

```
            gbc = new GridBagConstraints();
            gbc.gridx = 1;
            gbc.gridy = 2;
            gbc.weightx = 0.0;
            gbc.weighty = 0.0;
            gbc.anchor = GridBagConstraints.WEST;
            gbc.fill = GridBagConstraints.NONE;
            gbc.insets = new Insets(5, 5, 10, 5);
            ((GridBagLayout)
p2.getLayout()).setConstraints(tf_taget_ip, gbc);
            p2.add(tf_taget_ip);

            // สร้าง Taget Port และ add ลง panel
            JLabel label5 = new JLabel("Taget Port:");

            gbc = new GridBagConstraints();
            gbc.gridx = 0;
            gbc.gridy = 3;
            gbc.weightx = 0.0;
            gbc.weighty = 0.0;
            gbc.anchor = GridBagConstraints.EAST;
            gbc.fill = GridBagConstraints.NONE;
            gbc.insets = new Insets(5, 5, 0, 5);
            ((GridBagLayout) p2.getLayout()).setConstraints(label5,
gbc);
            p2.add(label5);

            tf_taget_port = new JTextField(15);

            gbc = new GridBagConstraints();
            gbc.gridx = 1;
            gbc.gridy = 3;
            gbc.weightx = 0.0;
            gbc.weighty = 0.0;
            gbc.anchor = GridBagConstraints.WEST;
            gbc.fill = GridBagConstraints.NONE;
            gbc.insets = new Insets(5, 5, 10, 5);
            ((GridBagLayout)
p2.getLayout()).setConstraints(tf_taget_port, gbc);
            p2.add(tf_taget_port);

            try {
                  String host =
InetAddress.getLocalHost().getHostAddress();
                  tf_local_host.setText(host);
            } catch (UnknownHostException e) {
            }

            TitledBorder titledBorder = new TitledBorder(new
EtchedBorder(), "Connect Host");

            p2.setBorder(titledBorder);

            return p2;
      }

      public void actionPerformed(ActionEvent event) {
            Object source = event.getSource();

            if (source == addTarget) {
                  String ip = tf_remote_address.getText().trim();
                  String port = tf_remote_data_port.getText().trim();
                  String localPort =
tf_local_data_port.getText().trim();

                  if (avReceiver != null) {
                        avReceiver.addTarget(ip, port, localPort);
                  }
```

```
                addTargetToList(localPort, ip, port);
            } else if (source == removeTarget) {
                String ip = tf_remote_address.getText().trim();
                String port = tf_remote_data_port.getText().trim();

                int index = list.getSelectedIndex();

                if (index != -1) {
                    Target target = (Target)
targets.elementAt(index);

                    if (avReceiver != null) {
                        avReceiver.removeTarget(ip, port);
                    }

                    targets.removeElement(target);
                    listModel.setData(targets);

                    if (targets.size() == 0) {
                        removeTarget.setEnabled(false);
                    }

                    if (targets.size() > 0) {
                        if (index > 0) {
                            index--;
                        } else {
                            index = 0;
                        }

                        list.setSelectedIndex(index);

                        setTargetFields();
                    } else {
                        list.setSelectedIndex(-1);
                    }
                }
            } else if (source == startRx) {
                if (startRx.getLabel().equals("Start Receiver")) {
                    avReceiver = new AVReceiver(this, targets);
                    startRx.setLabel("Stop Receiver");
                } else {
                    avReceiver.close();
                    avReceiver = null;

                    startRx.setLabel("Start Receiver");
                }
            }else if (source==connect) {
                String tmp =
tf_username.getText()+tf_password.getText();

      sendRequest(tf_taget_ip.getText(),Integer.parseInt(tf_taget_por
t.getText()),tmp);
                if (true) {
                p2.show(false);
                p1.add(targetPanel);
                p1.remove(1);
                GridBagConstraints gbc = new GridBagConstraints();
                gbc.gridx = 0;
                gbc.gridy = 3;
                gbc.gridwidth = 2;
                gbc.weightx = 1.0;
                gbc.weighty = 0.0;
                gbc.anchor = GridBagConstraints.CENTER;
                gbc.fill = GridBagConstraints.HORIZONTAL;
                gbc.insets = new Insets(5, 5, 10, 5);
                ((GridBagLayout)
p1.getLayout()).setConstraints(buttonPanel, gbc);
                buttonPanel.add(startRx);
                buttonPanel.add(on);
```

```
                    buttonPanel.add(off);
                    buttonPanel.add(disconnect);
                    connect.setEnabled(false);
                    disconnect.show(true);
                    startRx.show(true);
                    on.show(true);
                    off.show(true);
                    p1.add(buttonPanel);
                    pack();
                    }else{
                          JOptionPane.showMessageDialog(p1,"Connect
fail pleace try again","",JOptionPane.WARNING_MESSAGE);
                    }

              }else if (source == disconnect) {

      sendRequest(tf_taget_ip.getText(),Integer.parseInt(tf_taget_por
t.getText()),"disconnect");
                    p2.show(true);
                    p1.remove(1);
                    connect.setEnabled(true);
                    disconnect.show(false);
                    startRx.show(false);
                    on.show(false);
                    off.show(false);
                    pack();
              }else if (source==off) {

      sendRequest(tf_taget_ip.getText(),Integer.parseInt(tf_taget_por
t.getText()),"off");
              }else if (source==on) {

      sendRequest(tf_taget_ip.getText(),Integer.parseInt(tf_taget_por
t.getText()),"on");
              }
      }

      synchronized public void addTargetToList(String localPort,
String ip, String port) {
              ListUpdater listUpdater = new ListUpdater(localPort, ip,
port, listModel, targets, removeTarget);

              SwingUtilities.invokeLater(listUpdater);
      }

      public void windowClosing(WindowEvent event) {
              config.write();

              System.exit(0);
      }

      public void windowClosed(WindowEvent event) {
      }

      public void windowDeiconified(WindowEvent event) {
      }

      public void windowIconified(WindowEvent event) {
      }

      public void windowActivated(WindowEvent event) {
      }

      public void windowDeactivated(WindowEvent event) {
      }

      public void windowOpened(WindowEvent event) {
      }
```

```
	public void keyPressed(KeyEvent event) {
	}

	public void keyReleased(KeyEvent event) {
		Object source = event.getSource();

		if (source == list) {
			int index = list.getSelectedIndex();
		}
	}

	public void keyTyped(KeyEvent event) {
	}

	public void mousePressed(MouseEvent e) {
	}

	public void mouseReleased(MouseEvent e) {
	}

	public void mouseEntered(MouseEvent e) {
	}

	public void mouseExited(MouseEvent e) {
	}

	public void mouseClicked(MouseEvent e) {
		Object source = e.getSource();

		if (source == list) {
			setTargetFields();
		}
	}

	public void setTargetFields() {
		int index = list.getSelectedIndex();

		if (index != -1) {
			Target target = (Target) targets.elementAt(index);

			tf_remote_address.setText(target.ip);
			tf_remote_data_port.setText(target.port);
			tf_local_data_port.setText(target.localPort);
		}
	}

	/*
	 * Method นี้จะเป็น Method ที่เปิด socket
	 * แล้วทำการส่งข้อมูลไปยังอีกฝั่งหนึ่งเลยโดยไม่ต้องเปิด serversocket
	 * รอรับข้อมูล เพราะฝั่งนี้ต้องการจะทำการส่งข้อมูลเลย
	 */
	public void sendRequest(String host, int port, String data) {
		try {
			Socket skt = new Socket(host, port);
			BufferedReader in = new BufferedReader(new
InputStreamReader(skt.getInputStream()));
			while (!in.ready()) {
			}
			String inText = in.readLine();
			//System.out.println(inText);
			in.close();

			ServerSocket serverSocket = new ServerSocket(1234);
			Socket socket = serverSocket.accept();
			PrintWriter out = new
PrintWriter(socket.getOutputStream(), true);
			out.print(data);
			out.close();
```

```
                        socket.close();
                        serverSocket.close();
                        if
(inText.equals(tf_username.getText()+tf_password.getText())) {
                              isConnected=true;
                        }
                  } catch (Exception e) {
                        System.out.print("Sorry! It didn't work!\n");
                        isConnected = false;
                  }
            }
            public static void main(String[] args) {
                  new MonitoringSystem();
            }
}

class TargetListModel extends AbstractListModel {
            private Vector options;

            public TargetListModel(Vector options) {
                  this.options = options;
            }

            public int getSize() {
                  int size;

                  if (options == null) {
                        size = 0;
                  } else {
                        size = options.size();
                  }

                  return size;
            }

            public Object getElementAt(int index) {
                  String name;

                  if (index < getSize()) {
                        Target o = (Target) options.elementAt(index);

                        name = o.localPort + " <--- " + o.ip + ":" +
o.port;
                  } else {
                        name = null;
                  }

                  return name;
            }

            public void setData(Vector data) {
                  options = data;

                  fireContentsChanged(this, 0, data.size());
            }

}

class ListUpdater implements Runnable {
            String localPort, ip, port;
            TargetListModel listModel;
            Vector targets;
            JButton removeTarget;

            public ListUpdater(String localPort, String ip, String port,
TargetListModel listModel, Vector targets, JButton removeTarget) {
                  this.localPort = localPort;
                  this.ip = ip;
                  this.port = port;
```

```
            this.listModel = listModel;
            this.targets = targets;
            this.removeTarget = removeTarget;
      }

      public void run() {
            Target target = new Target(localPort, ip, port);

            if (!targetExists(localPort, ip, port)) {
                  targets.addElement(target);
                  listModel.setData(targets);
                  removeTarget.setEnabled(true);
            }
      }

      public boolean targetExists(String localPort, String ip, String
port) {
            boolean exists = false;

            for (int i = 0; i < targets.size(); i++) {
                  Target target = (Target) targets.elementAt(i);

                  if (target.localPort.equals(localPort) &&
target.ip.equals(ip) && target.port.equals(port)) {
                        exists = true;
                        break;
                  }
            }

            return exists;
      }

}
```

//AVReceiver.java//

```
package reciever;

import java.io.*;
import java.awt.*;
import java.net.*;
import java.awt.event.*;
import java.util.*;

import javax.media.*;
import javax.media.rtp.*;
import javax.media.rtp.event.*;
import javax.media.rtp.rtcp.*;
import javax.media.protocol.*;
import javax.media.format.AudioFormat;
import javax.media.format.VideoFormat;
import javax.media.Format;
import javax.media.format.FormatChangeEvent;
import javax.media.control.BufferControl;
import com.sun.media.rtp.RTPSessionMgr;

public class AVReceiver implements ReceiveStreamListener,
SessionListener,
      ControllerListener, RemoteListener
{
    Vector mgrs;
    Vector playerWindows = null;

    boolean dataReceived = false;
    Object dataSync = new Object();

    Vector targets;
    MonitoringSystem rx;
```

```
    public AVReceiver( MonitoringSystem rx, Vector targets) {
      this.rx= rx;
      this.targets = targets;

      initialize();
    }

    private void initialize() {
      mgrs = new Vector( targets.size());

        playerWindows = new Vector();

        // Open the RTP sessions.
        for( int i= 0; i < targets.size(); i++) {
                Target target= (Target) targets.elementAt( i);

          addTarget( target.ip, target.port, target.localPort);

      }
    }

    public boolean isDone() {
      return playerWindows.size() == 0;
    }

    public void addTarget( String senderAddress,
                           String senderPort,
                           String localPort) {
        try {
          InetAddress ipAddr;
          SessionAddress localAddr;
          SessionAddress destAddr;

          RTPManager mgr = RTPManager.newInstance();
          mgr.addSessionListener(this);
          mgr.addReceiveStreamListener(this);
          mgr.addRemoteListener( this);

          ipAddr = InetAddress.getByName( senderAddress);

          int local_port= new Integer( localPort).intValue();

          if( ipAddr.isMulticastAddress()) {
            // local and remote address pairs are identical:
            localAddr= new SessionAddress( ipAddr,
                                           local_port,
                                           6);

            destAddr = new SessionAddress( ipAddr,
                                           local_port,
                                           6);
          } else {
              localAddr= new SessionAddress(
InetAddress.getLocalHost(),
                                                 local_port);

            int remotePort= new Integer( senderPort).intValue();

                destAddr = new SessionAddress( ipAddr, remotePort);
          }

          mgr.initialize( localAddr);

          // You can try out some other buffer size to see
          // if you can get better smoothness.
          BufferControl bc =
(BufferControl)mgr.getControl("javax.media.control.BufferControl");
          if (bc != null) {
              bc.setBufferLength(350);
```

```
            }

            mgr.addTarget(destAddr);

            mgrs.addElement( mgr);
          } catch (Exception e){
            System.err.println("Cannot create the RTP Session: " +
e.getMessage());
            return;
          }
      }

      public void removeTarget( String address, String port) {
        for( int i= 0; i < playerWindows.size(); i++) {
            PlayerWindow pw= (PlayerWindow)playerWindows.elementAt( i);

            System.out.println( pw.senderAddress + ":" +
pw.senderPort);

            if( address.equals( pw.senderAddress) && port.equals(
pw.senderPort)) {
              try {
                pw.close();
              } catch (Exception e) {
              }

              playerWindows.removeElement( pw);

              break;
            }
        }

        for (int i = 0; i < mgrs.size(); i++) {
            RTPSessionMgr mgr= (RTPSessionMgr) mgrs.elementAt( i);

            SessionAddress addr= mgr.getRemoteSessionAddress();

            String dataPort= addr.getDataPort() + "";

             if( addr.getDataHostAddress().equals( address)
             && port.equals( dataPort)) {

               mgr.removeTarget( addr, "Closing session from
AVReceiver");
               mgr.dispose();

              mgrs.removeElement( mgr);

              break;
            }
        }
      }

      /**
       * Close the players and the rtp  managers.
       */
      protected void close() {
        for( int i= 0; i < playerWindows.size(); i++) {
            try {
              ((PlayerWindow)playerWindows.elementAt(i)).close();
            } catch (Exception e) {}
        }

        playerWindows.removeAllElements();

        // close the RTP session.
        for (int i = 0; i < mgrs.size(); i++) {
            RTPManager mgr= (RTPManager) mgrs.elementAt( i);
```

```
            mgr.removeTargets( "Closing session from AVReceiver");
            mgr.dispose();
          mgr = null;
        }
      }

      PlayerWindow find(Player p) {
        for (int i = 0; i < playerWindows.size(); i++) {
          PlayerWindow pw = (PlayerWindow)playerWindows.elementAt(i);

          if (pw.player == p) {
            return pw;
          }
        }
        return null;
      }


      PlayerWindow find(ReceiveStream strm) {
        for (int i = 0; i < playerWindows.size(); i++) {
          PlayerWindow pw = (PlayerWindow)playerWindows.elementAt(i);
          if (pw.stream == strm)
            return pw;
        }
        return null;
      }


      /**
       * SessionListener.
       */
      public synchronized void update(SessionEvent evt) {
        if (evt instanceof NewParticipantEvent) {
          Participant p =
((NewParticipantEvent)evt).getParticipant();
          System.err.println("  - A new participant had just joined:
" + p.getCNAME());
        }
      }


      /**
       * ReceiveStreamListener
       */
      public synchronized void update( ReceiveStreamEvent evt) {
        RTPManager mgr = (RTPManager)evt.getSource();

        Participant participant = evt.getParticipant(); // could be
null.
        ReceiveStream stream = evt.getReceiveStream();  // could be
null.

        String timestamp= getTimestamp();

        if (evt instanceof RemotePayloadChangeEvent) {

          System.err.println("  - Received an RTP
PayloadChangeEvent.");
          System.err.println("Sorry, cannot handle payload change.");
          System.exit(0);

        } else if (evt instanceof NewReceiveStreamEvent) {
          try {
            stream = ((NewReceiveStreamEvent)evt).getReceiveStream();
            DataSource ds = stream.getDataSource();

            // Find out the formats.
            RTPControl ctl =
(RTPControl)ds.getControl("javax.media.rtp.RTPControl");
```

```
            if (ctl != null) {
                System.err.println("  - Recevied new RTP stream: " +
ctl.getFormat());
            } else {
                System.err.println("  - Recevied new RTP stream");
            }

            if (participant == null) {
                System.err.println("      The sender of this stream
had yet to be identified.");
            } else {
                System.err.println("      The stream comes from: " +
participant.getCNAME());
            }

            // create a player by passing datasource to the Media
Manager
            Player p = javax.media.Manager.createPlayer(ds);

            if (p == null) {
                return;
            }

            p.addControllerListener(this);
            p.realize();
            PlayerWindow pw = new PlayerWindow(p, stream);
            playerWindows.addElement(pw);

            // Notify intialize() that a new stream had arrived.
            synchronized (dataSync) {
                dataReceived = true;
                dataSync.notifyAll();
            }

        } catch (Exception e) {
            System.err.println("NewReceiveStreamEvent exception " +
e.getMessage());
            return;
        }
    } else if (evt instanceof StreamMappedEvent) {
        if (stream != null && stream.getDataSource() != null) {
            DataSource ds = stream.getDataSource();
            // Find out the formats.
            RTPControl ctl =
(RTPControl)ds.getControl("javax.media.rtp.RTPControl");
            System.err.println("  - The previously unidentified
stream ");


            if (ctl != null) {
                System.err.println("      " + ctl.getFormat());
            }

            System.err.println("      had now been identified as sent
by: " + participant.getCNAME());

            RTPSessionMgr rtpManager= (RTPSessionMgr)
evt.getSessionManager();

            SessionAddress addr=
rtpManager.getRemoteSessionAddress();

            if( addr != null) {
                PlayerWindow pw = find(stream);

                pw.setTitle( addr.getDataHostAddress(),
                             addr.getDataPort());
            }
        }
```

```
        } else if (evt instanceof ByeEvent) {
            StringBuffer sb= new StringBuffer();

            sb.append( timestamp + " BYE");

            String reason= ((ByeEvent) evt).getReason();

            sb.append( " from " + participant.getCNAME());
            sb.append( " ssrc=" + stream.getSSRC());
            sb.append( " reason='" + reason + "'");

            //rx.rtcpReport( sb.toString());

            PlayerWindow pw = find(stream);

            if (pw != null) {
              pw.close();
              playerWindows.removeElement(pw);
            }
        }
    }

    public void update( RemoteEvent event) {
      String timestamp= getTimestamp();

      if( event instanceof ReceiverReportEvent) {
          ReceiverReport rr= ((ReceiverReportEvent)
event).getReport();

          StringBuffer sb= new StringBuffer();

          sb.append( timestamp + " RR");

          if( rr != null) {
            Participant participant= rr.getParticipant();

            if( participant != null) {
                sb.append( " from " + participant.getCNAME());
                sb.append( " ssrc=" + rr.getSSRC());
            } else {
                sb.append( " ssrc=" + rr.getSSRC());
            }

            //rx.rtcpReport( sb.toString());
          }
      } else if( event instanceof SenderReportEvent) {
          SenderReport sr= ((SenderReportEvent) event).getReport();

          StringBuffer sb= new StringBuffer();

          sb.append( timestamp + " SR");

          if( sr != null) {
            Participant participant= sr.getParticipant();

            if( participant != null) {
                sb.append( " from " + participant.getCNAME());
                sb.append( " ssrc=" + sr.getSSRC());
            } else {
                sb.append( " ssrc=" + sr.getSSRC());
            }

            //rx.rtcpReport( sb.toString());
          }
      } else {
          System.out.println( "RemoteEvent: " + event);
      }
    }
```

```
private String getTimestamp() {
  String timestamp;

  Calendar calendar= Calendar.getInstance();

  int hour= calendar.get( Calendar.HOUR_OF_DAY);

  String hourStr= formatTime( hour);

  int minute= calendar.get( Calendar.MINUTE);

  String minuteStr= formatTime( minute);

  int second= calendar.get( Calendar.SECOND);

  String secondStr= formatTime( second);

  timestamp= hourStr + ":" + minuteStr + ":" + secondStr;

  return timestamp;
}

private String formatTime( int time) {
  String timeStr;

  if( time < 10) {
      timeStr= "0" + time;
  } else {
      timeStr= "" + time;
  }

  return timeStr;
}

/**
 * ControllerListener for the Players.
 */
public synchronized void controllerUpdate(ControllerEvent ce) {

  Player p = (Player)ce.getSourceController();

  if (p == null)
      return;

  // Get this when the internal players are realized.
  if (ce instanceof RealizeCompleteEvent) {
      PlayerWindow pw = find(p);
      if (pw == null) {
        // Some strange happened.
        System.err.println("Internal error!");
        System.exit(-1);
      }
      pw.initialize();
      pw.setVisible(true);
      p.start();
  }

  if (ce instanceof ControllerErrorEvent) {
      p.removeControllerListener(this);
      PlayerWindow pw = find(p);
      if (pw != null) {
        pw.close();
        playerWindows.removeElement(pw);
      }

      System.err.println("AVReceiver internal error: " + ce);
  }
}
```

```
/**
 * GUI classes for the Player.
 */
class PlayerWindow extends Frame {
  Player player;
  ReceiveStream stream;
  String senderPort;
  String senderAddress;

  PlayerWindow(Player p, ReceiveStream strm) {
      player = p;
      stream = strm;

      super.setTitle( "Unkown Sender");
  }

  public void setTitle( String address,
                        int    port) {
      senderAddress= address;
      senderPort= port + "";

      String title= senderAddress + ":" + senderPort;

      super.setTitle( title);
  }

  public void initialize() {
      add(new PlayerPanel(player));
  }

  public void close() {
      player.close();
      setVisible(false);
      dispose();
  }

  public void addNotify() {
      super.addNotify();
      pack();
  }
}

/**
 * GUI classes for the Player.
 */
class PlayerPanel extends Panel {

  Component vc, cc;

  PlayerPanel(Player p) {
      setLayout(new BorderLayout());
      if ((vc = p.getVisualComponent()) != null)
        add("Center", vc);
      if ((cc = p.getControlPanelComponent()) != null)
        add("South", cc);
  }

  public Dimension getPreferredSize() {
      int w = 0, h = 0;
      if (vc != null) {
        Dimension size = vc.getPreferredSize();
        w = size.width;
        h = size.height;
      }
      if (cc != null) {
        Dimension size = cc.getPreferredSize();
        if (w == 0)
            w = size.width;
```

```
            h += size.height;
          }
          if (w < 160)
            w = 160;
          return new Dimension(w, h);
        }
      }
}
```

//Config.java//

```
package reciever;

import java.io.*;
import java.util.*;

/*
 *  คลาสนี้เป็นคลาสที่ใช้จัดการเกี่ยวกับ config ของการเชื่อมต่อด้วย rtp ซึ่งเป็นการส่งข้อมูลภาพและเสียง
 *  โดยจะมีสองงานด้วยกันคือ read config และ write config โดยจะปรากฏอยู่ครึ่งส่วน target panel ใน
หน้า
 *  GUI ของโปรแกรม ซึ่งจะช่วยให้สามารถจดจำค่าต่าง ๆ ของ เครื่องที่มีอยู่ได้(เก็บได้หลาย ๆ เครื่อง)
 */

public class Config {
      private String pathPrefix;

      public Vector targets;

      public Config() {
            // หา path ของ home ของเครื่องที่รันแล้วก็ทำการ read config
            // ออกมาก่อนถ้าไม่มีก็จะสร้างใหม่ขึ้นมา
            pathPrefix = System.getProperty("user.home") +
File.separator;
            read();
      }

      // เปิด file แล้วอ่านข้อมูลของ target ที่จดจำไว้มาใส่เก็บไว้ใน vector
      // เพื่อนำไปเป็น list ให้เลือก
      // โดยอ่านผ่านทาง method readString()
      public void read() {
            targets = new Vector();

            try {
                  String path = pathPrefix + "rx.dat";

                  FileInputStream fin = new FileInputStream(path);
                  BufferedInputStream bin = new
BufferedInputStream(fin);
                  DataInputStream din = new DataInputStream(bin);

                  int n_targets = din.readInt();

                  for (int i = 0; i < n_targets; i++) {
                        String localPort = readString(din);
                        String ip = readString(din);
                        String port = readString(din);

                        targets.addElement(new Target(localPort, ip,
port));
                  }

                  fin.close();
            } catch (IOException e) {
                  System.out.println("rx.dat file missing!");
            }
      }
```

```
	// เปิด file แล้วนำ list ของ target มาวน loop เขียนข้อมูลเพื่อช่วยจำ target
	//โดยผ่านทางการเรียกใช้ method writeString()
	public void write() {
		try {
			String path = pathPrefix + "rx.dat";

			FileOutputStream fout = new FileOutputStream(path);
			BufferedOutputStream bout = new
BufferedOutputStream(fout);
			DataOutputStream dout = new DataOutputStream(bout);

			dout.writeInt(targets.size());

			for (int i = 0; i < targets.size(); i++) {
				Target target = (Target)
targets.elementAt(i);

				writeString(dout, target.localPort);
				writeString(dout, target.ip);
				writeString(dout, target.port);
			}

			dout.flush();
			dout.close();
			fout.close();
		} catch (IOException e) {
			System.out.println("Error writing rx.dat!");
		}
	}
	// ใช้อ่านจาก DataInputStream
	public String readString(DataInputStream din) {
		String s = null;

		try {
			short length = din.readShort();

			if (length > 0) {
				byte buf[] = new byte[length];

				din.read(buf, 0, length);

				s = new String(buf);
			}
		} catch (IOException e) {
			System.err.println(e);
		}

		return s;
	}
	// ใช้เขียนลง DataOutputStream
	public void writeString(DataOutputStream dout, String str) {
		try {
			if (str != null) {
				dout.writeShort(str.length());
				dout.writeBytes(str);
			} else {
				dout.writeShort(0);
			}
		} catch (Exception e) {
			e.printStackTrace();
		}
	}
}
```

//Target.java//

```
package reciever;
/*
 * คลาสนี้เป็นคลาสที่นิยามขึ้นเพื่อ เก็บค่าต่าง ๆ ของปลายทางที่ต้องการทำการติดกันโดย rtp
 */
public class Target {
    String localPort;
    String ip;
    String port;
    /*
     * การทำงานก็คือเมื่อมีการนิยาม target เกิดขึ้นในที่ใด ๆ ก็ตามจะเซต ค่าต่าง ๆ ของปลายทางเก็บไว้
     * และสามารถนำไปใช้เมื่อไรก็ได้
     */
    public Target( String localPort,
                String ip,
                String port) {
      this.localPort= localPort;
      this.ip= ip;
      this.port= port;
    }
}
```

//MyDataSinkListener.java//

```
package reciever;
import javax.media.datasink.*;


public class MyDataSinkListener implements DataSinkListener
{
      boolean endOfStream = false;

      public void dataSinkUpdate(DataSinkEvent event)
      {
            if (event instanceof
javax.media.datasink.EndOfStreamEvent)
                  endOfStream = true;
      }

      public void waitEndOfStream(long checkTimeMs)
      {
            while (! endOfStream)
            {
                  System.out.println("datasink: waiting for end of
stream ...");
                  try { Thread.currentThread().sleep(checkTimeMs); }
catch (InterruptedException ie) {}
            }
            System.out.println("datasink: ... end of stream
reached.");
      }
}
```

****

ภาคผนวก ข

```
                ORG     0000H

MAIN:           MOV     SCON,#50H
                MOV     TMOD,#021H
                MOV     TH1,#0FDH
                MOV     TL1,#0FDH
                SETB    TR1

                MOV     P1,#00H
GET:            JNB     RI,$
                MOV     A,SBUF
                CLR     RI

                CJNE    A,#031H,PW_OFF
                MOV     P1,#00000100B
                AJMP    FEEDBACK_ON

FEEDBACK_ON:            JB      P1.7,SEND_ON
SEND_ON:        MOV     DPTR,#SEND_TEXT_ON
                ACALL   TX_TEXT
                AJMP    GET
        ;-----------------------------------------
TX_TEXT:        CLR     TI
TX_LOOP:        CLR     A
                MOVC    A,@A+DPTR
                INC     DPTR
                CJNE    A,#0,TX_CHAR
                RET

TX_CHAR:        MOV     SBUF,A
                JNB     TI,$
                CLR     TI

                ACALL   DELAY_100ms
                AJMP    TX_LOOP
        ;-----------------------------------------
DELAY_100ms:            MOV     R7,#100
DELAY_100ms_1:          MOV     R6,#0E6H
DELAY_100ms_2:          NOP
                NOP
                DJNZ    R6,DELAY_100ms_2
                DJNZ    R7,DELAY_100ms_1
                RET
        ;-----------------------------------------
NULL            EQU     0
SEND_TEXT_ON:           DB      'ON',00DH,NULL
;-------------------------------------------------

PW_OFF:                 MOV     P1,#00000000B
                AJMP    FEEDBACK_OFF

FEEDBACK_OFF:           JNB     P1.7,SEND_OFF
SEND_OFF:       MOV     DPTR,#SEND_TEXT_OFF
                ACALL   TX_TEXT
                AJMP    GET
SEND_TEXT_OFF:          DB      'OFF',00DH,NULL
;-------------------------------------------------
        END
```