การควบคุมกล้องวีดีโอวงจรปิดผ่านอินเตอร์เน็ต

นายกาญจวัฒน์ จินบุตร

นายวีรวัฒน์ เอี่ยมวิทยา

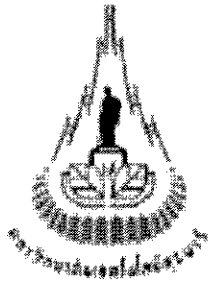
นายวันชัย ล้อถาวรพงษ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม สำนักวิชาวิศวกรรมศาสตร์

มหาวิทยาลัยเทคโนโลยีสุรนารี

ปีการศึกษา 2547



ควบคุมกล้องวีดีโอวงจรปิดผ่านอินเตอร์เน็ต

นายภานุวัฒน์ ดำภูผา
นายวัชรัตน์ เอี่ยมรักษา
นายวันชัย ก่อการุณย์พงศ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต
สาขาวิชาวิศวกรรมโทรคมนาคม สำนักวิชาวิศวกรรมศาสตร์
มหาวิทยาลัยเทคโนโลยีสุรนารี
ปีการศึกษา 2547

โครงงาน ควบคุมกล้องวีดีโอวงจรปิดผ่านอินเตอร์เน็ต
ผู้ดำเนินงาน นายภานุวัฒน์ ดำภูผา B4404538
นายวัชรัตน์ เอี่ยมรักษา B4405450
นายวันชัย ก่อการุณย์พงศ์ B4405498
อาจารย์ที่ปรึกษา อาจารย์ ดร.ชาญชัย ทองโสภา
สาขาวิชา วิศวกรรมโทรคมนาคม
ภาคการศึกษา 2/2547

บทคัดย่อ

ปัจจุบันการติดต่อสื่อสารผ่านเครือข่ายอินเตอร์เน็ตถือว่าเป็นที่นิยมกันอย่างแพร่หลายอย่างมากในยุคปัจจุบัน เนื่องจากมีการใช้งานได้อย่างหลากหลาย ครอบคลุมและกว้างขวาง ที่เห็นได้ชัดก็คือการแลกเปลี่ยนข้อมูลกัน ซึ่งมันจะเป็นการดีที่ผู้คนจากมุมโลกหนึ่งหรือที่ห่างไกลกัน สามารถที่จะรับรู้ข่าวสารจากอีกที่หนึ่งได้อย่างรวดเร็ว การควบคุมระยะไกลก็ได้เข้ามามีบทบาทสำคัญในด้านการควบคุมต่างๆ เพราะไม่จำเป็นที่ผู้ควบคุมจะต้องทำงานอยู่ที่ที่มีเครื่องมือนั้นตั้งอยู่ ถ้าเราสามารถประยุกต์ใช้เครือข่ายอินเตอร์เน็ตเพื่อนำมาใช้ในการควบคุมเครื่องมือนั้น เราก็จะสามารถควบคุมเครื่องมือนั้นที่อยู่ห่างไกลออกไปได้โดยผ่านเครือข่ายอินเตอร์เน็ต ซึ่งนี่ก็เป็นที่มาของโครงงานนี้ คือจะเป็นการสร้างการควบคุมกล้องวีดีโอวงจรปิดระยะไกลโดยผ่านเครือข่ายอินเตอร์เน็ต แนวความคิดเบื้องต้นคือ ต้องการควบคุมการแสดงภาพของกล้องวงจรปิดหลายๆตัว รวมทั้งการหมุนของกล้องโดยใช้อินเตอร์เน็ตเป็นสื่อกลาง และทำการรับข้อมูลภาพที่ได้จากกล้องเหล่านั้น นั่นก็หมายความว่าเราสามารถที่จะตรวจสอบเหตุการณ์ต่างภายในบริเวณนั้น ในขณะที่ผู้ควบคุมอยู่ที่อื่น ซึ่งเราสามารถนำเอาหลักการนี้ไปประยุกต์ใช้ในการรักษาความปลอดภัยแก่สถานที่ต่างๆได้

ปริญญานิพนธ์ปีการศึกษา 2547

สาขาวิชาวิศวกรรมโทรคมนาคม

สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี

เรื่อง **ควบคุมกล้องวีดีโอวงจรปิดผ่านอินเตอร์เน็ต**

ผู้จัดทำ

1. นายภานุวัฒน์ ดำภูผา B4404538

2. นายวัชรัตน์ เอี่ยมรักษา B4405450

3. นายวันชัย ก่อการุณย์พงศ์ B4405498

อาจารย์ที่ปรึกษา

(อาจารย์ ดร.ชาญชัย ทองโสภา)

กิตติกรรมประกาศ

การทำปริญญานิพนธ์นี้สำเร็จได้ด้วยดีเนื่องจากได้รับความอนุเคราะห์ในการให้คำปรึกษาในด้านต่างๆ ในระหว่างการดำเนินการจากบุคคลหลายท่านที่ได้ให้ความช่วยเหลืออย่างดีเสมอมา อันได้แก่

- ผศ.ดร.รังสรรค์ วงศ์สรรค์ หัวหน้าสาขาวิชาวิศวกรรมโทรคมนาคมและอาจารย์ที่ปรึกษาทางด้านวิชาการที่ให้คำปรึกษาในด้านต่างๆ รวมทั้งให้คำเสนอแนะที่เป็นประโยชน์ต่อพวกเราอย่างมาก
- อ.ดร. ชาญชัย ทองโสภา อาจารย์ที่ปรึกษาที่ให้คำปรึกษาในด้านต่างๆทั้งทางวิชาการและการปฏิบัติงาน และคอยดูแล ควบคุมการทำงานอย่างใกล้ชิด และคอยให้กำลังใจเสมอมา
- อาจารย์ปิยาภรณ์ กระฉอดนอก อาจารย์ประจำสาขาวิชาวิศวกรรมโทรคมนาคมที่ให้คำปรึกษาในด้านการส่งข้อมูลผ่านเครือข่ายคอมพิวเตอร์
- คุณประพล จาระตะคุ หัวหน้าอาคารศูนย์เครื่องมือ 3 ที่ช่วยดูแลและติดต่อประสานงานเรื่องเอกสารต่างๆ
- คุณวิชัย ศรีสุรักษ์ และคุณสมิง เติมพรมราช เจ้าหน้าที่เทคนิคห้องปฏิบัติการไมโครโปรเซสเซอร์ที่ให้คำปรึกษาในด้านการสร้างฮาร์ดแวร์
- คุณเด่น กากแก้ว ที่ให้คำปรึกษาในด้านการเขียนโปรแกรมภาษาจาวา
- คุณณรงค์ฤทธิ์ กวางแก้ว ที่ให้คำปรึกษาในด้านการเขียนเว็บเพจ
- บุคลากรสาขาวิชาวิศวกรรมโทรคมนาคมที่สนับสนุนด้านอุปกรณ์และการดำเนินงาน
- พี่น้องชาววิศวกรรมโทรคมนาคมทุกคนซึ่งให้กำลังใจเสมอมา

สุดท้ายนี้คณะผู้จัดทำขอกราบขอบพระคุณบิดา มารดา ที่อบรมเลี้ยงดู และให้โอกาสทางการศึกษา และคอยสนับสนุนด้วยดีตลอดมาอย่างหาที่เปรียบมิได้

จึงเห็นสมควรที่จะมอบคุณคุณความดี และเกียรติคุณเหล่านี้แด่ท่านที่กล่าวมานี้ รวมถึงบุคคลที่มิได้กล่าวนามมา ณ ที่นี้ด้วย

นายภานุวัฒน์ ดำภูผา
นายวัชรัตน์ เอี่ยมรักษา
นายวันชัย ก่อการุณย์พงศ์

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญรูปภาพ

หน้า

สารบัญรูปภาพ (ต่อ)

หน้า

สารบัญตาราง

หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมา

เนื่องด้วยในปัจจุบัน อินเตอร์เน็ตได้เข้ามามีบทบาทสำคัญทางด้านการสื่อสารในชีวิตประจำวันของมนุษย์เรามากขึ้น ทำให้เราเห็นความสะดวกในการใช้งานด้วยระบบอินเตอร์เน็ต ดังนั้นการทำให้อุปกรณ์หรือเครื่องมือใดๆ ที่ใช้ประโยชน์จากการติดต่อผ่านอินเตอร์เน็ต จะได้รับประโยชน์อย่างมากมาย ที่เห็นได้ชัดคือการควบคุมระยะไกล (Remote Control) ซึ่งในปัจจุบันนี้เครือข่ายอินเตอร์เน็ตเป็นเครือข่ายการสื่อสารที่ครอบคลุมไปทั่งโลก และจำนวนผู้ใช้อินเตอร์เน็ตก็เพิ่มขึ้นอย่างรวดเร็ว เราจึงใช้คุณสมบัตินี้มาประยุกต์ใช้ในการควบคุมกล้องวีดีโอวงจรปิดผ่านเครือข่ายอินเตอร์เน็ต อันจะทำให้เราสามารถที่จะสั่งงานจากที่ใดที่หนึ่งในโลกก็ได้

1.2 วัตถุประสงค์

1. สามารถนำความรู้ที่ได้มาจากการศึกษาจากภาคทฤษฎีของวิชาต่างๆ ที่ได้ศึกษามาปฏิบัติและประยุกต์ใช้เพื่อสร้างชิ้นงานขึ้นมาและสามารถนำไปใช้งานจริงได้

2. ศึกษาหาความรู้เพิ่มเติมในเรื่องต่างๆ ที่เกี่ยวข้องกับการทำโครงงาน เช่น ศึกษาการออกแบบแผ่นปริ๊น (PCB) ศึกษาเกี่ยวกับภาษา JAVA เป็นต้น

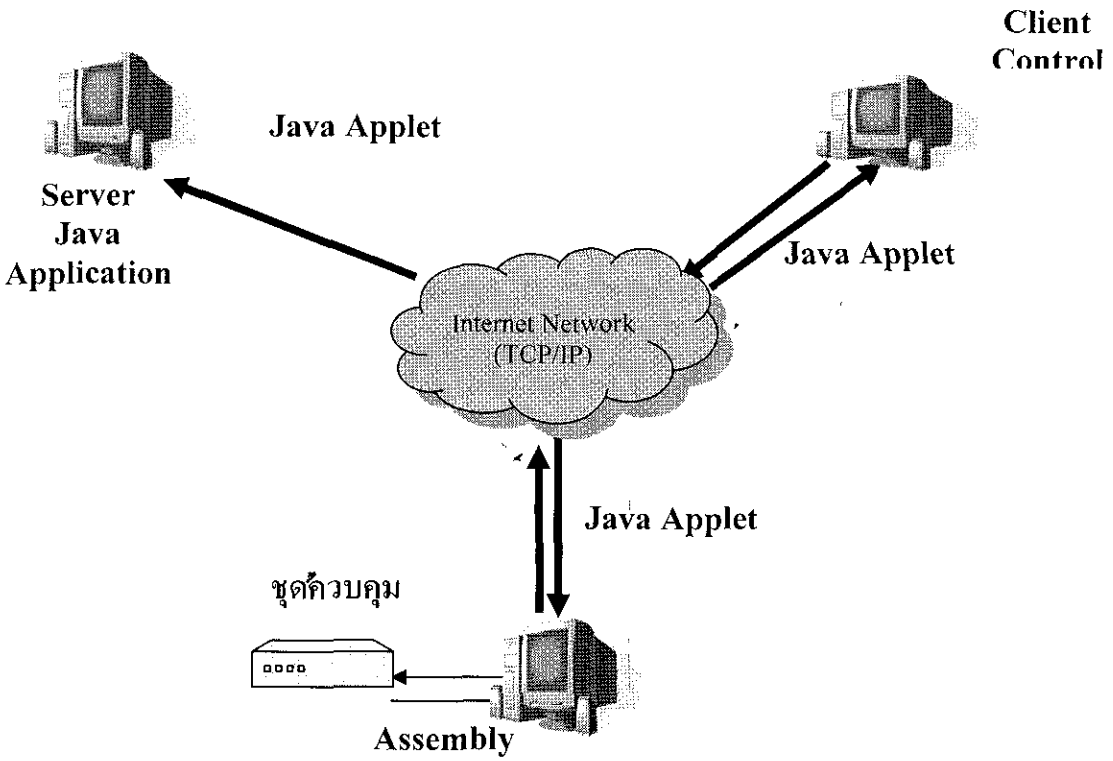
3. สามารถนำโครงงานนี้ไปใช้ในการศึกษาความปลอดภัยและป้องกันการถูกโจรกรรมต่อทรัพย์สินต่างๆ

1.3 ขอบเขตของโครงงาน

1. สร้างเครื่องควบคุมการทำงานของกล้องวงจรปิดจะทำหน้าที่สวิตช์สัญญาณภาพที่ได้จากกล้องวงจรปิดและควบคุมการหมุนของกล้องผ่าน Step Motor ซึ่งการทำงานต่างๆ จะใช้ไมโครคอนโทรลเลอร์ในการควบคุมสั่งการ สำหรับเครื่องควบคุมการทำงานของกล้องวงจรปิดจะถูกสั่งงานผ่านเว็บไซต์

2. สร้างเว็บเพจที่มีฟังก์ชั่นต่างๆ คือมีให้เลือกหลายหมายเลขกล้องที่ต้องการจะให้แสดงภาพ มีฟังก์ชั่นควบคุมการหมุนของกล้อง (ซ้าย-ขวา) และมีปุ่มให้แสดงภาพแบบวนต่อเนื่อง

3. จะต้องทำการเชื่อมต่อสัญญาณภาพระหว่างเครื่องควบคุมการทำงานของกล้องวงจรปิดกับเครื่องคอมพิวเตอร์ และต้องเขียนโปรแกรม JAVA เพื่อใช้ในการรับส่งข้อมูลต่างๆ ระหว่างเครื่องคอมพิวเตอร์ที่ต่อกับเครื่องควบคุมการทำงานของกล้องวงจรปิดกับเครือข่ายอินเตอร์เน็ต (เว็บเพจ)



รูปที่ 1-1 แสดงหลักการทำงานของการรับ-ส่งภาพและ control ผ่านเครือข่ายอินเตอร์เน็ต

จากรูปที่ 1-1 สามารถที่จะอธิบายการทำงานได้ดังนี้

เมื่อเครื่อง Server ทำการรันโปรแกรม Java Application ทำให้มีการเปิด Socket หรือ Port ขึ้นเพื่อเป็นช่องทางการสื่อสารและรอรับการติดต่อจากเครื่อง Client ทั้งสอง เมื่อเครื่อง Client ที่บ้านซึ่งติดกับชุดควบคุมกล้อง ทำการรันโปรแกรม Java Application ขึ้น ทำให้เครื่องที่บ้านจะทำการส่งภาพไปยังเครื่อง Server ในขณะเดียวกันก็จะทำการรอรับการ Control กล้องด้วยเช่นกัน เมื่อเครื่อง Client ที่เป็นผู้ใช้งานซึ่งสามารถอยู่ที่ใดก็ได้บนโลกนี้ ทำการเปิดโปรแกรม Web Browser เพื่อต้องการใช้งาน โปรแกรมก็จะทำการติดต่อไปยังเครื่อง Server เครื่อง Server จะทำการส่งภาพกลับไปยังผู้ใช้ ภาพก็จะปรากฏขึ้นที่หน้าจอผู้ใช้ หากผู้ใช้ต้องการที่จะควบคุมกล้อง ก็จะทำการส่ง Control มายังเครื่อง Server เครื่อง Server ก็จะรับ Control แล้วทำการส่ง Control มายังเครื่อง Client ที่ติดกับชุดควบคุมกล้องซึ่งกำลังรอรับ Control อยู่ เมื่อได้รับ Control ก็จะทำการส่ง Control ผ่านพอร์ตอนุกรมไปยังตัวไมโครคอนโทรลเลอร์ที่ชุดควบคุมกล้อง เพื่อทำการ Control กล้องตามต้องการ

จากการทำงานข้างต้นจะมีโปรแกรมที่เกี่ยวข้องกับการรับ-ส่งภาพและ control ผ่านอินเตอร์เน็ตและสั่งให้ชุดควบคุมกล้องทำงาน ดังนี้

- โปรแกรมที่เครื่อง Server (Java Application) จะใช้ serverclass.jpx

- โปรแกรมที่เครื่อง Client ที่ติดกับชุดควบคุมกล้องวงจรปิด (Java Applet) จะใช้โปรแกรม clientcamsender.jpx
- โปรแกรมที่เครื่อง Client ที่จะดูภาพและ control กล้องผ่านระบบอินเตอร์เน็ต (Java Applet) จะใช้โปรแกรม imagereceive.jpx
- โปรแกรมที่ใช้ควบคุมการทำงานของกล้องวงจรปิด (Assembly) จะใช้โปรแกรม ControlCamera.ASM

1.4 ขั้นตอนการดำเนินงาน

1. ศึกษาตำราและค้นหาข้อมูลเกี่ยวกับการสื่อสารข้อมูลผ่านอินเตอร์เน็ตและการเขียนโปรแกรมภาษา Java

2. ทำ Software ของไมโครคอนโทรลเลอร์เพื่อควบคุมการทำงานของกล้องวงจรปิด

3. ออกแบบฮาร์ดแวร์ของเครื่องควบคุมการทำงานของกล้องวงจรปิด

4. จัดเตรียมวัสดุและอุปกรณ์ต่างๆ

5. ทำ Hardware ของเครื่องควบคุมการทำงานของกล้องวงจรปิด

6. เขียนโปรแกรม JAVA ในการรับส่งข้อมูลระหว่างคอมพิวเตอร์ที่ต่อกับเครื่องควบคุมการทำงานของกล้องวงจรปิดกับเครือข่ายอินเตอร์เน็ต

7. ทดสอบและแก้ไขการทำงานของระบบ

8. จัดทำรายงานโครงการ

9. สรุปและประเมินผล

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. สามารถนำโครงงานนี้ไปใช้ในชีวิตประจำวันเพื่อใช้รักษาความปลอดภัยในชีวิตและทรัพย์สิน

2. ได้รับความรู้มากขึ้นในทางปฏิบัติหลังจากทำการศึกษาทางทฤษฎีมาแล้ว

3. สามารถค้นคว้าหาความรู้เพิ่มเติมนอกเหนือจากหลักสูตรที่เรียนได้

4. สามารถทำงานเป็นทีมได้

บทที่ 2
ทฤษฎีและหลักการ

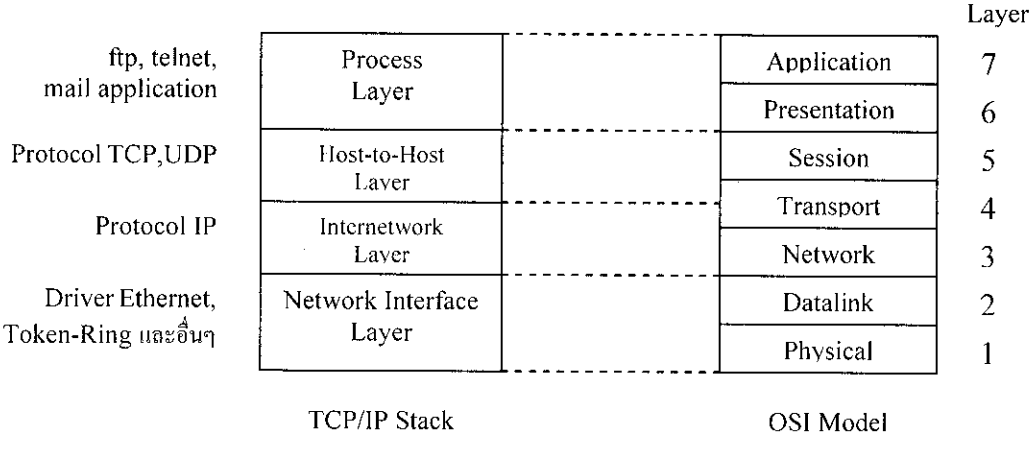
2.1 รูปแบบมาตรฐานโปรโตคอลของอินเตอร์เน็ต

2.1.1 โครงสร้างของโปรโตคอล TCP/IP

Protocol TCP/IP มีการจัดแบ่งกลไกการทำงานออกเป็นชั้นๆ หรือ layer เหมือนกับมาตรฐาน OSI Model และสามารถเทียบเคียงกับมาตรฐานของ OSI Model ได้ ซึ่งในแต่ละ layer ของ Protocol TCP/IP จะประกอบด้วย [8]

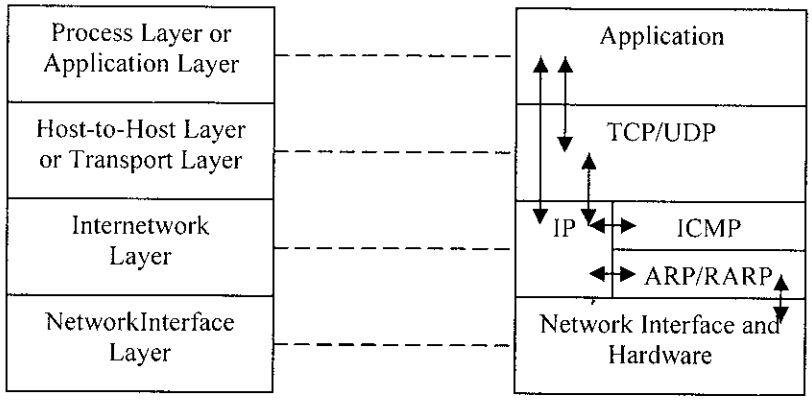
- Process layer หรือ Application layer
- Host-to-Host layer หรือ Transport layer
- Internetwork layer
- Network Interface layer

โดยเมื่อเทียบกับมาตรฐาน OSI Model แล้วจะเป็นดังรูปที่ 2-1 ซึ่งเราจะเห็นว่าบาง Layer ของ Protocol TCP/IP เทียบได้กับมาตรฐาน OSI Model ถึงสอง Layer และบาง Layer ก็จะทำงานคาบเกี่ยวกับหลายๆ Layer ของ OSI Model ตัวอย่างเช่น ในส่วน Network Interface layer ของ Protocol TCP/IP จะเทียบได้กับการนำเอา Datalink layer และ Physical layer ของมาตรฐาน OSI Model มารวมกัน เป็นต้น



รูปที่ 2-1 แสดง TCP/IP stack เปรียบเทียบกับมาตรฐาน OSI

ในแต่ละกลไกของ Protocol TCP/IP จะมี Protocol อื่นๆ ในชุดของ TCP/IP ร่วมทำงานอยู่ด้วย จึงทำให้เป็นที่มาของชื่อเรียก Protocol Stack เนื่องจากมี Protocol ซ้อนทับกันอยู่เพื่อช่วยกันทำงานดังรูปต่อไปนี้

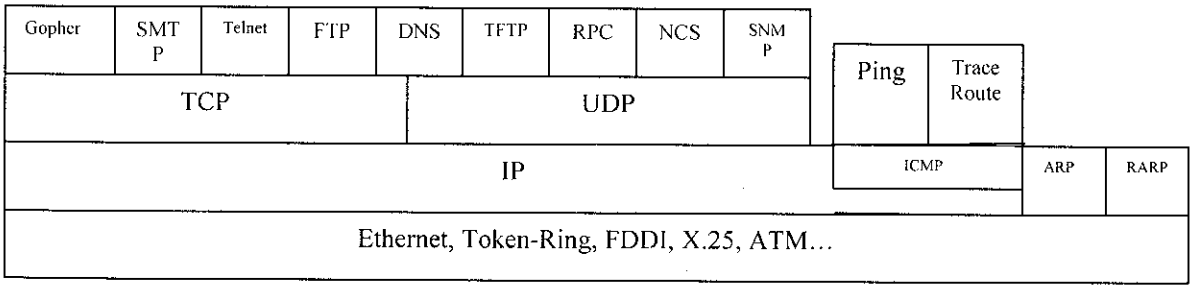


TCP/IP Stack

รูปที่ 2-2 แสดงความสัมพันธ์ระหว่าง Protocol ต่างๆ ของ TCP/IP

จากรูปจะเห็นได้ว่า มี Protocol ในแต่ละระดับซ้อนทับกันอยู่หลายตัวด้วยกัน การซ้อนกันเป็นชั้นๆ หรือแต่ละ Layer นี้หากเป็น OSI Model จะมีข้อความบังคับให้แต่ละชั้นติดต่อได้กับเฉพาะชั้นที่ติดกับตนเองเท่านั้น แต่สำหรับ TCP/IP Stack แล้วจะเห็นว่าบางชั้นสามารถละเลยหรือข้ามไปติดต่อกับชั้นอื่นที่ไม่ติดกับตนได้

1. Process layer หรือ Application Layer



รูปที่ 2-3 แสดง Application ต่างๆ ใน TCP/IP Stack

จากรูปแสดงลำดับชั้นการทำงานของ Protocol TCP/IP เทียบกับมาตรฐาน OSI Model นั้นในชั้นบนสุดเรียกว่า Process Layer ทำงาน 2 หน้าที่เทียบได้กับ Application layer และ Presentation Layer ในชั้นนี้จะรองรับการทำงานของ Application ต่างๆ ที่ทำงานเป็น Process อยู่ในเครื่องเซิร์ฟเวอร์ที่ให้บริการและเครื่องที่ขอใช้บริการ หรือไคลเอนต์ (client) ซึ่งจะติดต่อกันผ่าน Protocol เฉพาะ Application อีกทีหนึ่ง ตัวอย่างเช่น เมื่อผู้ใช้งานอินเตอร์เน็ตต้องการโอนถ่ายไฟล์หรือ download ข้อมูลจากเครื่องเซิร์ฟเวอร์ที่ให้บริการ โดยอาจจะเรียกใช้โปรแกรม ftp client ทั่วไป เช่น โปรแกรม WS_ftp ติดต่อกับ Process ftp ที่กำลังให้บริการอยู่ที่เครื่องเซิร์ฟเวอร์ จากนั้นตัว Process ftp ก็จะเรียกใช้ Protocol FTP (File Transfer Protocol) เพื่อทำการถ่ายโอนไฟล์นี้ หรือ

ถ้าผู้ใช้ต้องการเรียกใช้งานคอมพิวเตอร์เครื่องที่อยู่ห่างไกลออกไปด้วยการใช้โปรแกรม telnet ที่เครื่องเซิร์ฟเวอร์ให้บริการ ตัว Process telnet ที่ทำงานอยู่ก็จะเรียกใช้ Protocol Telnet เพื่อติดต่อกัน หรือในกรณีที่มีการเรียกใช้โปรแกรม web browser เช่น Netscape Navigator เพื่อเรียกดูเว็บเพจในเว็บไซต์ CNN ที่เครื่องซึ่งให้บริการเว็บของ CNN ก็จะมี Process HTTP (HyperText Transfer Protocol) ทำงานอยู่และจะติดต่อกับผู้ใช้ผ่าน Protocol HTTP เป็นต้น

การทำงานของ Application ต่างๆ จะอยู่ที่ Process Layer นี้ และมีการติดต่อกันตามแต่ละ Protocol เฉพาะแล้วแต่ Application ที่ใช้งาน จากการที่ Process Layer ของ TCP/IP รองรับให้ Protocol อื่นทำงานได้หลาย Process และหลาย Protocol ได้พร้อมกันนั้น ทำให้ผู้ใช้สามารถเปิดโปรแกรมใช้งานได้หลายๆ อย่างพร้อมกัน เช่น เปิดโปรแกรม Internet Explorer เพื่อเรียกดูเว็บเพจพร้อมกับใช้งานโปรแกรม Outlook Express เพื่อรับส่งอีเมล์ไปพร้อมกันได้โดยไม่ต้องรอให้ทำงานอย่างหนึ่งอย่างใดเสร็จก่อน หรือในปัจจุบันมีการพัฒนาโปรแกรม web browser ให้สามารถเรียกใช้งาน Protocol อื่นๆ ได้มากขึ้น ทำให้เราสามารถใช้โปรแกรม web browser โอนถ่ายไฟล์ข้อมูลที่ใช้ Protocol FTP ได้โดยไม่ต้องไปหาโปรแกรมอื่นมาใช้

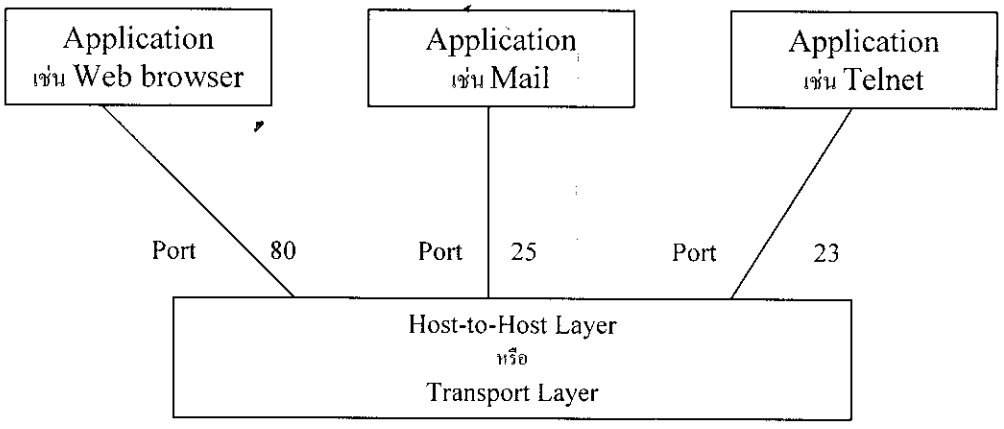
Protocol หลักๆ ที่ทำงานใน Process Layer ซึ่งผู้ใช้มักจะคุ้นเคยกันดีได้แก่ FTP (File Transfer Protocol), Telnet, HTTP (HyperText Transfer Protocol) ซึ่งช่วยในการสร้างรูปแบบหน้าจอขั้นก้าวหน้าซึ่งนิยมใช้ในระบบ WWW [10] และ SMTP (Simple Mail Transfer Protocol) นอกจากนี้ยังมี Protocol อื่นที่อยู่เบื้องหลัง ซึ่งทำงานโดยที่ผู้ใช้ไม่สามารถมองเห็นได้จากโปรแกรมหรือไม่ได้มีการใช้งานโดยตรง เช่น

- Protocol DNS(Domain Name System) ที่ทำหน้าที่แปลงข้อมูลชื่อ domain name หรือชื่อเว็บไซต์ทั้งหลายให้เป็นหมายเลข IP Address
- Protocol SNMP (Simple Network Management Protocol) ใช้ในการควบคุมและตรวจสอบอุปกรณ์ที่อยู่ในเครือข่าย
- Protocol DHCP (Dynamic Host Configuration Protocol) ทำหน้าที่แจกจ่ายข้อมูลพารามิเตอร์ของเครือข่ายให้กับเครื่องลูกข่ายที่เชื่อมต่ออยู่

2. Host-to-Host layer หรือ Transport Layer

ผู้ใช้งานอินเตอร์เน็ตเคยสงสัยหรือไม่ว่าเครื่องเซิร์ฟเวอร์ที่ให้บริการต่างๆ เช่นเว็บเซิร์ฟเวอร์นั้น เมื่อมีผู้เข้ามาเรียกใช้บริการพร้อมกันหลายคน จะมีวิธีการส่งข้อมูลกลับไปยังต้นทางได้อย่างไรโดยไม่ผิดพลาด ซึ่งบางครั้งผู้ใช้รายหนึ่งอาจจะเปิดโปรแกรม web browser ช้อนกันเพื่ออ่านข้อมูลจากเว็บเพจอื่นๆพร้อมกันไปได้ ดังนั้นระบบจะทราบได้อย่างไรถึงการจัดส่งข้อมูลได้อย่างถูกต้อง

การทำงานที่ชั้นของ Host-to-Host Layer นี้จะมีบทบาทในการจัดการต่อจาก Process layer บางครั้งเรามักเรียกชั้น Host-to-Host ว่าเป็น Transport layer ซึ่งไม่ใช่ชั้นของ Transport layer ในมาตรฐาน OSI Model การทำงานของ Host-to-Host layer นี้จะมีการสร้าง connection หรือการเชื่อมต่อกันระหว่าง Application กับ Host-to-Host layer โดยจุดที่เชื่อมกันเพื่อรับส่งข้อมูลนี้เรียกว่า port หรือ socket (คำว่า port ในที่นี้ไม่ได้หมายถึง port ทางฮาร์ดแวร์) และในแต่ละ Application ก็จะสร้างการเชื่อมต่อผ่าน port ได้พร้อมกันหลาย Application ซึ่งการใช้งาน port ของแต่ละ Application ที่อยู่ในชั้น Process layer จะแตกต่างกันตามหมายเลขที่กำหนดไว้ และแต่ละ Protocol จะมีการใช้งาน port หมายเลขต่างๆ ไม่ซ้ำกัน ตามรูปที่ 2-4

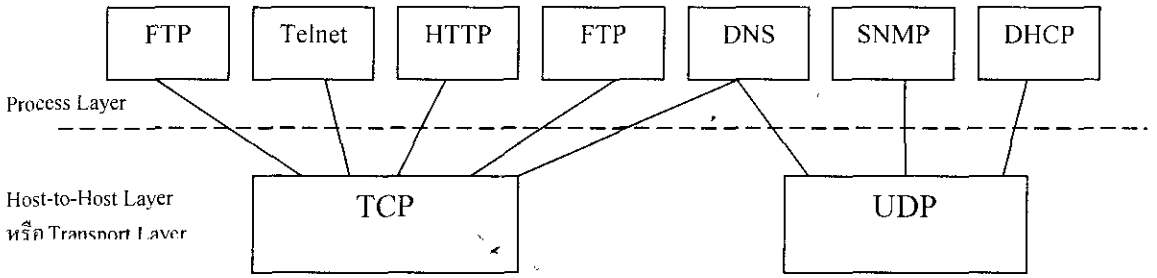


รูปที่ 2-4 Application หรือ Process ต่างๆ สื่อสารกับ Host-to-Host Layer ผ่านจุดเชื่อมต่อหรือ port ส่วนหมายเลขในรูปคือหมายเลข port ที่ Processใช้งาน เช่น เว็บหรือ Process http ใช้งาน port 80 ในการส่งผ่านข้อมูล

เมื่อ Application ทำงานผ่าน Protocol ในชั้น Process layer จะมีการส่งผ่านข้อมูลไปยัง Host-to-Host layer ที่ชั้นนี้จะมีการเชื่อมต่อผ่าน port ที่กำหนด ทำให้การรับส่งข้อมูลในแต่ละ Protocol ทำได้ถูกต้อง ถึงแม้ว่าในเครื่องเซิร์ฟเวอร์ที่ให้บริการจะมีการทำงานอยู่หลาย Process ที่แตกต่างกันก็ตาม หรือมีผู้ใช้บริการเข้ามาใช้งานพร้อมกันจำนวนมากและหลาย Application ในเวลาเดียวกัน ในชั้น Host-to-Host หรือ Transport layer ของ TCP/IP นี้ จะมี Protocol ทำงานอยู่ 2 Protocol ที่แตกต่างกัน คือ Protocol TCP และ Protocol UDP (User Datagram Protocol) ในการส่งผ่านข้อมูลลงไปที่ชั้นถัดๆ ไป เราจะเห็นได้ว่า Protocol TCP และ UDP จะถูกผนึกเข้าไปใน Protocol IP อีกทีหนึ่งและส่งต่อไปยังเครือข่ายอินเตอร์เน็ตต่อไป

ตัว Protocol TCP และ Protocol UDP จะมี Application เฉพาะเพื่อเรียกใช้งานแยกกันคือ Application ที่ใช้ Protocol FTP, Telnet, HTTP และ SMTP จะมีการส่งผ่านข้อมูลโดยเรียกใช้ Protocol TCP ส่วน Application ที่ใช้ Protocol SNMP และ DHCP จะส่งผ่านข้อมูลโดยเรียกใช้

Protocol UDP และสำหรับ Protocol DNS นั้นจะสามารถเรียกใช้งานได้ทั้ง TCP และ UDP ดังรูป 2-5 ซึ่งเหตุผลที่มีการเรียกใช้ Protocol TCP และ UDP แตกต่างกัน ก็เนื่องจากวิธีทำงานของทั้งสอง Protocol ต่างกันนั่นเอง



รูปที่ 2-5 Process ต่างๆที่เรียกใช้ Transport layer เพื่อส่งผ่านข้อมูลโดยอาศัย port ซึ่งในแต่ละ Process จะเรียกใช้งาน port เฉพาะแตกต่างกัน ยกเว้น DNS ที่สามารถใช้งานได้ทั้ง TCP และ UDP

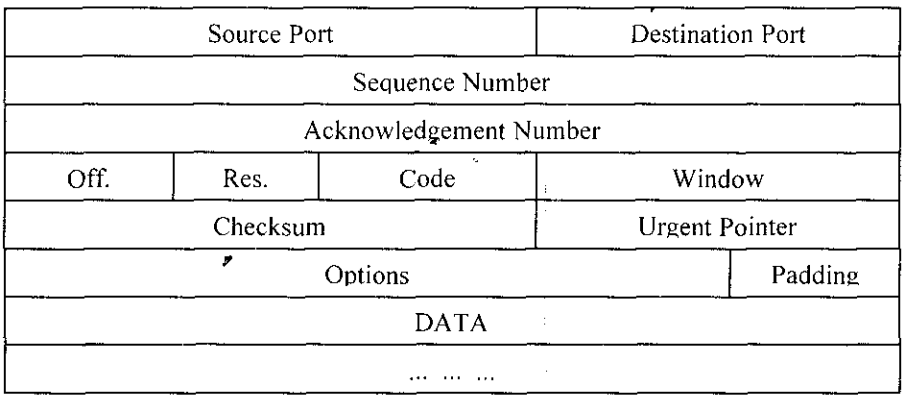
Protocol TCP

Protocol TCP (Transmission Control Protocol) เป็น Protocol ที่มีการรับส่งข้อมูลแบบ stream oriented protocol หมายความว่า การรับส่งข้อมูลจะไม่คำนึงถึงปริมาณข้อมูลที่จะส่งไป แต่จะแบ่งข้อมูลเป็นส่วนย่อยๆ ก่อน แล้วจึงส่งไปยังปลายทางอย่างต่อเนื่องเป็นลำดับข้อมูล ในกรณีที่ข้อมูลส่วนใดส่วนหนึ่งสูญหายไป ก็จะส่งข้อมูลส่วนนั้นใหม่อีกครั้ง สำหรับปลายทางก็จะทำหน้าที่จัดเรียงส่วนของข้อมูล datagram ใหม่ให้ต่อเนื่องและประกอบกลับเป็นข้อมูลทั้งหมดได้ ซึ่งจะแยกข้อมูลที่ไม่ถูกต้องออก ดังนั้น Application หรือ Process ใดที่อาศัยการส่งผ่านข้อมูลด้วย Protocol TCP จะต้องใช้หน่วยความจำและขนาดช่องสัญญาณ (bandwidth) มากกว่า UDP

การติดต่อระหว่างกันจะต้องเป็นแบบ connection-oriented คือต้องมีการสร้างติดต่อกันเป็น session ทั้ง 2 ด้านเสียก่อน แล้วจึงจะรับส่งข้อมูลไปได้พร้อมกัน (full duplex) เหมือนกับการใช้โทรศัพท์ติดต่อกัน เมื่อผู้ติดต่อต้นทางเรียกให้ฝ่ายตรงข้ามรับสายแล้ว จึงเริ่มสนทนา เช่น พูดคำว่า "สวัสดี" หรือ "ฮัลโหล" กันก่อนเพื่อให้แน่ใจว่าฝ่ายตรงข้ามพร้อมจะติดต่อด้วย จากนั้นจึงเริ่มต้นติดต่อกัน และเมื่อต้องการจะเลิกการติดต่อก็จะมีการพูดคำว่า "สวัสดี" ให้ฝ่ายตรงข้ามทราบว่าจะเลิกการติดต่อและวางสายไป ซึ่งในระหว่างการติดต่อกันนั้น แม้ว่าฝ่ายใดฝ่ายหนึ่งหรือทั้งสองฝ่ายจะเงียบไป คือไม่พูดอะไรเป็นเวลานานๆ แต่การเชื่อมโยงระหว่างทั้งสองด้านยังคงมีอยู่ไม่ขาดไปจนกว่าฝ่ายใดฝ่ายหนึ่งจะวางสาย เช่นเดียวกับการติดต่อกันด้วยกลไก Protocol TCP เมื่อ Application ต้องการส่งผ่านข้อมูลจะใช้ Protocol ที่เหมาะสมในชั้น Process layer ติดต่อไปและมีการสร้างช่องส่งข้อมูลผ่าน port ที่กำหนดเพื่อส่งผ่านข้อมูลไปยัง Protocol TCP

ในระหว่างการรับส่งข้อมูลนี้ Protocol TCP จะเพิ่มขบวนการตรวจสอบข้อมูลเพื่อให้ข้อมูลมีความถูกต้องไม่ผิดพลาดไปจากเดิม โดยการส่งสัญญาณตรวจสอบข้อมูล (acknowledgement) และส่งข้อมูลให้ใหม่อีกครั้ง ถ้าปลายทางไม่ได้รับหรือเกิดความผิดพลาดขึ้น

ความน่าเชื่อถือของการส่งผ่านข้อมูลโดย Protocol TCP จะมีมากกว่า แต่ก็ต้องอาศัยทรัพยากรของระบบมากกว่าในการทำงานเช่นกัน



รูปที่ 2-6 รูปแบบของ TCP packet จะเห็นว่ามีฟิลด์ Acknowledgement Number และข้อมูล Checksum เพื่อใช้ตรวจสอบการเดินทางของข้อมูล ส่วน header มีข้อมูลมากทำให้ต้องอาศัยทรัพยากรของระบบทำงานมาก

Protocol UDP

ใน Host-to-Host layer นอกจากจะมี Protocol TCP ทำงานแล้วก็ยังมี Protocol UDP (User Datagram Protocol) ที่มีคุณสมบัติแตกต่างกันอยู่ด้วย ในการรับส่งข้อมูลผ่าน Protocol UDP จะเป็นแบบที่ทั้งสองด้านไม่จำเป็นต้องอาศัยการสร้างช่องทางเชื่อมต่อกัน (connectionless) ระหว่างเครื่องเซิร์ฟเวอร์ให้บริการกับเครื่องที่ขอใช้บริการ (client) โดยไม่ต้องแจ้งให้ฝ่ายรับข้อมูลเตรียมรับข้อมูลเหมือน Protocol TCP และไม่มีการตรวจสอบความถูกต้องครบถ้วนในการรับส่งข้อมูลนั้นๆ ด้วยเนื่องจาก Protocol UDP ไม่มีสัญญาณตรวจสอบข้อมูล (acknowledgement) ในการส่งข้อมูลแต่ละครั้ง และไม่มีการส่งข้อมูลใหม่อีกในกรณีที่เกิดความผิดพลาดของการส่งข้อมูล เมื่อเป็นเช่นนี้ Application หรือ Process ใดที่ต้องอาศัย Protocol UDP ในการส่งผ่านข้อมูลก็อาจจะต้องสร้างขบวนการตรวจสอบข้อมูลขึ้นมาเอง

ตามรูป 2-7 จะเห็นว่า Protocol ชั้นบนขึ้นไป ที่ใช้ในการส่งผ่านข้อมูลโดย Protocol UDP เช่น Protocol SNMP (ใช้ควบคุมและจัดการอุปกรณ์ในเครือข่าย), หรือ Protocol DHCP (ใช้ส่งข้อมูลพารามิเตอร์ของเครือข่ายให้กับเครื่องลูกข่ายได้ใช้งาน) การส่งข้อมูลเหล่านั้นไม่ต้องรับทราบ

หรือตรวจสอบว่าข้อมูลไปถึงปลายทางถูกต้องหรือไม่ แต่กลไกการตรวจสอบข้อมูลที่มีการรับส่งจะไปทำในขั้นตอนของ Protocol ชั้นที่สูงกว่าแทน

ตัวอย่างขั้นตอนกลไกการทำงานโดยใช้ Protocol UDP มีดังต่อไปนี้

1. ในชั้นของ Process layer เมื่อโปรแกรมควบคุมอุปกรณ์เครือข่ายเช่น โปรแกรม Network management ต้องการส่งข้อมูลไปยังอุปกรณ์ที่ต้องการApplication นั้นจะติดต่อผ่าน Protocol SNMP ในชั้น Process layer

2. Protocol SNMP จะติดต่อกับ Protocol UDP ในชั้นถัดไป เพื่อขอติดต่อผ่าน port ที่กำหนด

3. Protocol SNMP เตรียมข้อมูลที่จะส่ง รวมทั้งที่อยู่ปลายทาง

4. Protocol SNMP ส่งผ่านข้อมูลให้ Protocol UDP ที่อยู่ในชั้น Host-to-Host layer

5. Protocol UDP ทำหน้าที่ผนึกข้อมูลหรือ datagram นั้น ไปกับ Protocol IP ในชั้นถัดลงไป เพื่อส่งข้อมูลออกจากเครื่อง

ซึ่งจะเห็นว่ามีกลไกที่ต่างจากการส่งข้อมูลด้วย Protocol TCP ซึ่งจะต้องมีการติดต่อกันก่อนและทั้งสองฝ่ายรับทราบการรับส่งข้อมูลของช่องการส่งข้อมูลนั้น

บิตที่ 0 16 31

Source Port	Destination Port
Length	UDP Checksum
DATA	
... ...	

รูปที่ 2-7 รูปแบบของ UDP packet จะมีฟิลด์ข้อมูลส่วน header น้อยมาก และไม่มีข้อมูล ส่วนการตรวจสอบข้อมูล ทำให้ UDP packet มีขนาดเล็ก และใช้หน่วยความจำหรือทรัพยากรของระบบน้อย

3. Internetwork Layer

ในระดับล่างต่อมาในชั้น Internetwork layer มีหน้าที่ส่งผ่านข้อมูลในระหว่างเครือข่าย โดยมี Protocol ที่ทำงานเป็นกลไกสำคัญในการส่งผ่านข้อมูลไปยังเครือข่ายใดๆ บนอินเตอร์เน็ต คือ Protocol IP (Internet Protocol) นอกจากนี้ในชั้น Internetwork layer ยังมี Protocol ทำงานอยู่ด้วยอีก 2 ชนิดคือ Protocol Internet Control Message Protocol (ICMP) และ Protocol Address Resolution Protocol (ARP)

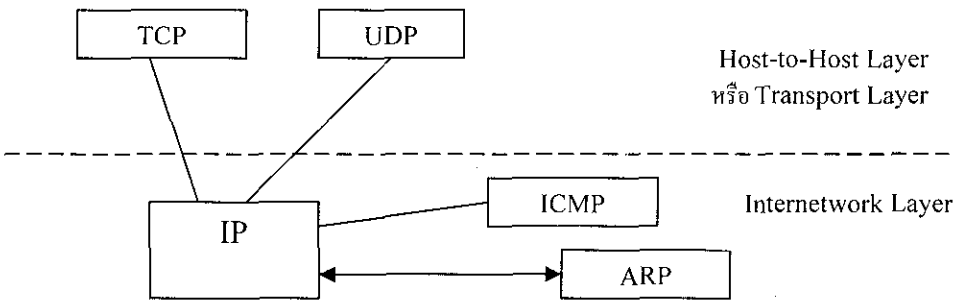
Protocol IP

Protocol IP ทำหน้าที่ให้บริการส่งผ่านข้อมูลที่มาจาก Host-to-Host เพื่อส่งข้ามไปยังเครือข่ายใดๆ ได้อย่างถูกต้อง แม้ว่าจะมีเครือข่ายเชื่อมต่อกันอยู่ในอินเตอร์เน็ตเป็นล้านๆ เครือข่ายก็ตาม

เนื่องจาก Protocol IP มีข้อมูลตำแหน่ง IP ปลายทางที่จะส่งข้อมูลไปให้ โดยทำงานร่วมกับอุปกรณ์ Router เพื่อส่งข้อมูลข้ามเครือข่ายออกไปได้ ตัว Protocol IP จะทำงานแบบ packet switching คือมีการส่งข้อมูลผ่านสวิตช์ (switch) ไปยังปลายทาง โดยข้อมูลจะเดินทางไปยังเครือข่ายต่างๆ ผ่านสวิทช์นี้ไปเรื่อยๆ จนกว่าจะถึงปลายทาง ตัววงจรผ่านหรือ switch นี้อาจเป็น Gateway หรือ Router ในระบบเครือข่ายก็ได้ ซึ่งในข้อมูลของ Protocol IP จะมีข้อมูลหมายเลขของ IP ปลายทางที่จะส่งข้อมูลไปและเมื่อถึงเครือข่ายปลายทางแล้ว จะมีกลไกการแปลงหมายเลข IP ให้เป็นหมายเลขฮาร์ดแวร์ประจำเครื่องที่ถูกต้องอีกทีหนึ่งด้วย Protocol ARP ตามรูปที่ 2-6 ที่จะแสดงการติดต่อกันระหว่าง Protocol ในชั้นของ Host-to-Host layer และ Internetwork layer

Protocol ICMP

หน้าที่หลักของ Protocol ICMP (Internet Control Message Protocol) คือการแจ้งหรือแสดงข้อความจากระบบ เพื่อบอกให้ผู้ใช้ทราบว่าเกิดอะไรขึ้นในการส่งผ่านข้อมูลนั้น ซึ่งปัญหาส่วนมากที่พบคือส่งไปไม่ได้ หรือปลายทางรับข้อมูลไม่ได้ เป็นต้น นอกจากนี้ Protocol ICMP ยังถูกเรียกใช้งานจากเครื่องเซิร์ฟเวอร์และ Router อีกด้วย เพื่อแลกเปลี่ยนข้อมูลที่ใช้ควบคุม ส่วนรูปแบบการทำงานของ Protocol ICMP นั้นจะทำงานคู่กับ Protocol IP ในระดับเดียวกัน และข้อความต่างๆ ที่แจ้งให้ทราบจะถูกผนึกอยู่ภายในข้อมูลของ IP (IP datagram) อีกทีหนึ่ง



รูปที่ 2-8 Protocol TCP และ UDP อาศัย Protocol IP ที่อยู่ชั้นล่างเพื่อส่งผ่านข้อมูลระหว่างเครือข่าย และในชั้น Internetwork Protocol ยังมี Protocol ICMP ทำหน้าที่ส่งข้อความแจ้งเตือนและ Protocol ARP ทำหน้าที่แปลงเลขหมาย IP ไปเป็นเลขหมายของฮาร์ดแวร์จริง

ข้อความที่ Protocol ICMP ส่งนั้นแบ่งออกได้ 2 แบบคือ ICMP error message หรือข้อความแจ้งข้อผิดพลาด และ ICMP query หรือข้อความเรียกขอข้อมูลเพิ่มเติม ตัวอย่างกลไกการทำงานของ Protocol ICMP เช่น เมื่อมีการส่งผ่านข้อมูลจากผู้ใช้ไปยังปลายทางที่ไม่ถูกต้อง หรือขณะนั้นเครื่องปลายทางเกิดปัญหาจนไม่สามารถรับข้อมูลได้ ที่ Router จะส่งข้อความแจ้งเป็น ICMP message ที่ชื่อ destination unreachable ให้กับผู้ส่งข้อมูล นอกจากนี้ตัวข้อมูลที่แจ้งข้อความก็จะมี

ส่วนของข้อมูล IP datagram ที่เกิดปัญหาด้วย ดังนั้นเมื่อผู้ส่งข้อมูลได้รับข้อความแจ้งแล้วก็จะทราบได้ว่าจุดที่เกิดปัญหานั้นอยู่ที่ใด

ดังนั้น Protocol ICMP จึงกลายมาเป็นเครื่องมืออย่างหนึ่งในการช่วยทดสอบเครือข่าย เช่น คำสั่ง ping ที่เรามักใช้ทดสอบว่าเครื่องเซิร์ฟเวอร์ที่ให้บริการหรืออุปกรณ์ที่ต่ออยู่ในเครือข่ายอินเตอร์เน็ตนั้นยังทำงานเป็นปกติหรือไม่ แล้วคำสั่ง ping มีการเรียกใช้งาน Protocol ICMP เป็นข้อความให้ทราบอีกต่อหนึ่ง

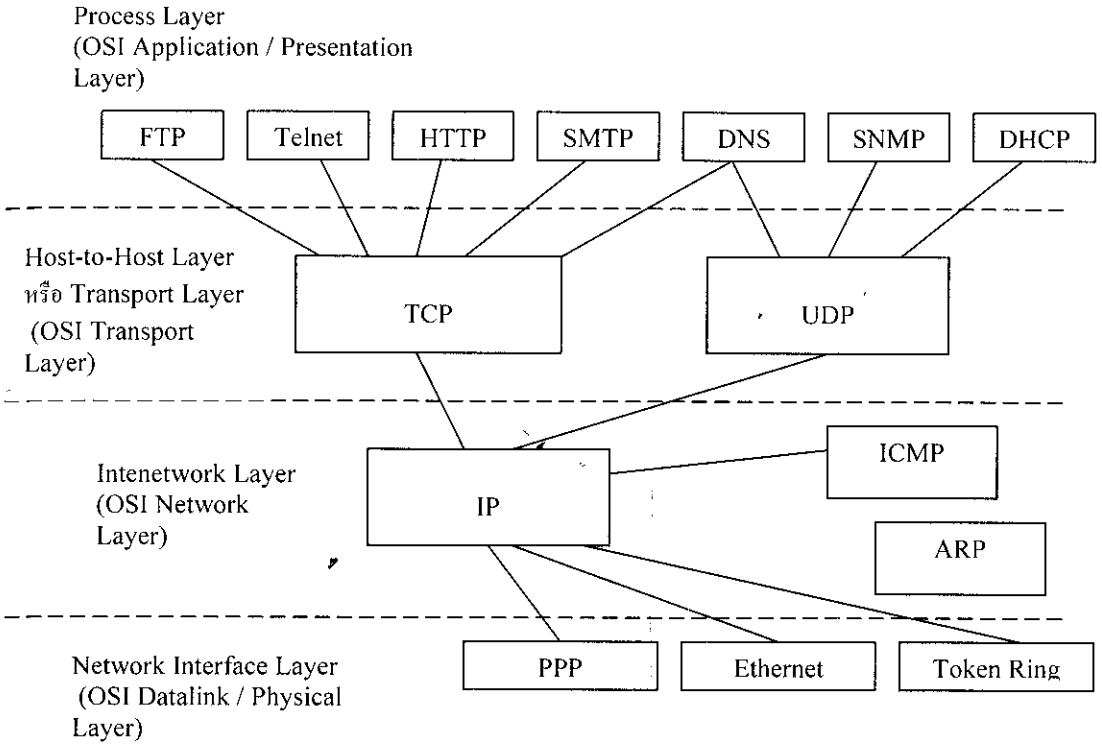
Protocol ARP

Protocol ARP (Address Resolution Protocol) ถูกเรียกใช้งานโดย Protocol IP เพื่อช่วยแปลงหมายเลข IP ไปเป็นหมายเลขฮาร์ดแวร์ปลายทาง ตัวอย่างเช่น เว็บเซิร์ฟเวอร์เครื่องหนึ่งเชื่อมต่ออยู่ในเครือข่ายอินเตอร์เน็ต และในการเชื่อมต่อนี้ต้องอาศัย Network Interface Card (NIC) หรือ LAN card ติดตั้งอยู่ ที่ LAN card นี้เองจะมีหมายเลขเฉพาะประจำฮาร์ดแวร์ที่ไม่ซ้ำกับใคร เพื่อใช้อ้างอิงการส่งข้อมูลในเครือข่าย แต่เมื่อมาใช้งานใน Protocol TCP/IP ก็จะต้องมีการกำหนดหมายเลข IP Address ประจำตัวเพื่อใช้อ้างอิงกัน และ Protocol ARP จะทำหน้าที่แปลงค่าหมายเลข IP ให้เป็นหมายเลขฮาร์ดแวร์จริงให้ในระดับการทำงานที่ Internetwork layer นี้ ซึ่งกลไกการแปลงนี้เรียกว่า Address resolution

4. Network Interface Layer

เนื่องจากในด้านกายภาพของเครือข่ายนั้น มีหลายวิธีการและหลายรูปแบบในการเชื่อมต่อระบบให้เป็นเครือข่าย แต่อย่างไรก็ตามในเครือข่ายอินเตอร์เน็ตนี้ ข้อมูล หรือ IP datagram จะถูกถ่ายทอดและส่งผ่านไปยังปลายทางโดยไม่คำนึงถึงรูปแบบการเชื่อมต่อทางกายภาพ ไม่ว่าจะเป็นการใช้เครือข่ายใยแก้วนำแสงหรือเครือข่ายสาย Unshielded Twist Pair (UTP) เชื่อมต่อเป็นเครือข่าย Ethernet ธรรมดาหรือเครือข่าย Token Ring, ATM, ISDN ฯลฯ ก็ตาม

การทำงานระดับล่างสุดต่อจาก Internetwork layer จะเป็นการแปลงข้อมูล IP datagram ให้อยู่ในรูปที่เหมาะสม และแปลงเป็นสัญญาณไฟฟ้าส่งไปยังเครือข่ายต่อไป ซึ่งในชั้น Network Interface layer นี้เมื่อเทียบกับมาตรฐาน OSI Model แล้วจะเป็นการรวม 2 layer เข้าด้วยกันคือ Datalink layer และ Physical layer กล่าวโดยสรุปคือ การทำงานในชั้นต่างๆ ตามโครงสร้างของ Protocol TCP/IP จะมีลักษณะดังรูปที่ 2-9



รูปที่ 2-9 โครงสร้างของ Protocol TCP/IP ในแต่ละชั้นหรือ layer จะมี Protocol หลักทำหน้าที่ต่างๆ และส่งผ่านข้อมูลไปยังเครือข่ายและออกสู่อินเตอร์เน็ต

ตารางที่ 1 สรุปหมายเลขบางส่วนของ port ที่ใช้งานโดย TCP และ UDP

Protocol ที่ใช้งาน	Port หรือ Socket เชื่อมต่อ (เลขฐาน10)	Protocol ในระดับ Host-to-Host	รายละเอียด
BootP	67	UDP	BOOTstrap Protocol ด้านเซิร์ฟเวอร์
BootP	68	UDP	BOOTstrap Protocol ด้านไคลเอนต์
DHCP	67	UDP	Dynamic Host Configuration Protocol ด้านเซิร์ฟเวอร์
DHCP	68	UDP	Dynamic Host Configuration Protocol ด้านไคลเอนต์
DNS	53	UDP/TCP	Domain Name System
FTP	21	TCP	File Transfer Protocol ด้านเซิร์ฟเวอร์ที่ควบคุม

FTP	20	TCP	File Transfer Protocol ด้านเซิร์ฟเวอร์ที่ส่งข้อมูล
HTTP	80	TCP/UDP	Hyper Text Transfer Protocol ด้านเซิร์ฟเวอร์
NetBT	138	UDP	NetBIOS datagram service
NetBT	139	TCP	NetBIOS session service
SMTP	25	TCP	Sample Mail Transfer Protocol ด้านเซิร์ฟเวอร์
SNMP	161	UDP	Simple Network Management Protocol ด้าน agent
SNMP	162	UDP	SNMP trap manager
Telnet	23	TCP	Teletype Network Protocol
TFTP	69	UDP	Trivial File Transfer Protocol
WINS	137	UDP	Windows Internet Name Service

กล่าวโดยสรุปก็คือ Protocol TCP/IP ทำงานโดยแบ่งเป็นชั้นเทียบกับ OSI Model ได้ กลไกในการทำงานของ Protocol TCP/IP มี 4 ชั้น ซึ่งในชั้นแรกคือ Process Layer ทำหน้าที่ติดต่อกับ Application และ Protocol ที่ Application นั้นๆ ใช้งานและต่อมาส่งให้ชั้น Host-to-Host Layer เพื่อติดต่อกันระหว่างเครื่องเซิร์ฟเวอร์ให้บริการกับเครื่องผู้ขอใช้บริการ ในชั้นนี้จะมีการสร้าง session หรือการเชื่อมต่อระหว่างระบบขึ้นตามแต่ละ Protocol ที่ต้องการ ต่อมาเป็นการผนึกข้อมูลไปเป็น IP datagram ที่ชั้น Internetwork layer โดยอาศัย Protocol IP เพื่อให้สามารถติดต่อส่งข้อมูลข้ามเครือข่ายไปยังเครือข่ายและเครื่องที่ถูกต้องได้ และสุดท้ายการส่งข้อมูลออกสู่โลกภายนอก ต้องอาศัยกลไกในชั้น Network Interface layer เพื่อแปลงข้อมูลใหม่ เพิ่มข้อมูลที่จำเป็นในการอ้างอิงตำแหน่งและแปลงข้อมูลเป็นสัญญาณไฟฟ้าส่งออกไปยังเครือข่าย และอาจจะออกไปยัง Gateway หรือ Router เพื่อข้ามเครือข่ายออกไปยังเส้นทางที่กำหนดไว้ในอินเตอร์เน็ตต่อไป

เราจะเห็นว่าในแต่ละชั้นของโครงสร้าง TCP/IP Stack มีการใช้งาน Protocol ต่างๆ อยู่หนึ่ง Protocol หรือมากกว่า ในแต่ละ Protocol เหล่านี้ก็จะรับผิดชอบทำหน้าที่ของตน เพื่อส่งผ่านข้อมูลลงไปยังระดับล่าง และออกสู่เครือข่ายอินเตอร์เน็ตในที่สุด

2.1.2 กลไกของ Protocol IP

ในการส่งผ่านข้อมูล หรือ IP datagram ไปยังเครือข่ายอินเตอร์เน็ตนั้น Protocol IP จะทำหน้าที่พิจารณาว่าปลายทางในการส่ง IP datagram นั้นจะเป็นภายในเครือข่ายของตนเองหรือจะต้องส่งข้อมูลข้ามเครือข่ายไปอีก โดยการพิจารณานี้ Protocol IP จะตรวจสอบจากค่า IP Address ปลายทางว่าส่วนที่เป็นค่าหมายเลขเครือข่าย (network address) จะเหมือนกับค่าหมายเลขเครือข่ายของ IP Address ต้นทางหรือไม่ ถ้าค่าตรงกันแสดงว่าการส่งข้อมูลอยู่ภายในเครือข่ายเดียวกัน แต่ถ้าค่าต่างกัน แสดงว่าต้องส่งข้อมูลไปยังปลายทางที่อยู่คนละเครือข่ายกัน

- การส่งข้อมูลภายในเครือข่ายเดียวกันมีกลไกดังนี้

1. Protocol IP จะเรียกใช้บริการ Protocol ARP (Address Resolution Protocol) เพื่อแปลงหมายเลข IP ปลายทางให้เป็นค่าหมายเลขฮาร์ดแวร์ เช่น MAC address

2. เมื่อ Protocol IP ได้รับค่าหมายเลขฮาร์ดแวร์แล้ว ก็จะส่งข้อมูลนั้นไปยังฮาร์ดแวร์ที่ระบุไว้

- การส่งข้อมูลข้ามเครือข่าย มีกลไกดังนี้

1. Protocol IP ตรวจสอบพบว่าหมายเลข IP Address ปลายทางอยู่คนละเครือข่ายกัน โดย Protocol IP จะอ่านค่า IP Address ของ Router เพื่อเตรียมส่งข้อมูลไปที่ Router แทน ซึ่งในที่นี่จะมีการกำหนดเป็น default router

2. Protocol IP จะเรียกใช้บริการ Protocol ARP เพื่อแปลงค่า IP Address ของ Router ให้เป็นค่าหมายเลขฮาร์ดแวร์

3. Protocol IP ส่งข้อมูล IP datagram ไปยัง Router ที่กำหนดไว้ จากนั้น Router ส่งข้อมูลข้ามเครือข่ายไปตามขั้นตอน

Protocol IP จะรู้ได้อย่างไรว่าเครือข่ายดังกล่าวมีการเชื่อมต่อ Router อยู่และมีค่า IP อะไร ซึ่งในเรื่องนี้ผู้ใช้จะต้องกำหนดค่าที่เรียกว่า default Router หรือ default Gateway เสียก่อน ว่ามีค่า IP Address อะไร โดยสามารถสอบถามได้จากผู้ดูแลระบบ สำหรับกลไกการส่งผ่านข้อมูลต่อจาก Router ไปยังเครื่องปลายทางก็จะมีกลไกเดียวกัน

2.1.3 การกำหนด IP Address ให้กับอุปกรณ์

มีคำถามอยู่ว่าเราจำเป็นต้องกำหนดหมายเลข IP Address ให้กับอุปกรณ์ทุกชิ้นในเครือข่ายหรือไม่ คำตอบคือ ไม่จำเป็นต้องกำหนดทั้งหมดก็ได้ แต่มีหลักอยู่ว่า เราจะต้องกำหนดหมายเลข IP Address ให้กับจุดเชื่อมต่อเข้าสู่เครือข่ายทุกจุด จุดเชื่อมต่อหรือ Interface อาจจะหมายถึง Network Interface Card (การ์ด LAN) ที่ติดตั้งในเครื่องเซิร์ฟเวอร์หรือ WAN port, Ethernet port ที่ Router

ใช้เชื่อมต่อเข้ากับเครือข่ายเป็นต้น การกำหนดหมายเลข IP Address ให้กับจุดเชื่อมต่อนี้ทำให้เราเข้าใจได้ว่าในบางอุปกรณ์มีจุดเชื่อมต่อเข้าเครือข่ายมากกว่าหนึ่งจุด จะต้องกำหนดหมายเลข IP Address ให้ครบ

การกำหนดค่า IP Address ไม่สามารถกำหนดขึ้นได้ตามใจชอบ แต่มีระเบียบวิธีแบ่งและการกำหนดที่ชัดเจนเป็นมาตรฐาน ใน IP Address จะถูกแบ่งเป็น 4 ส่วนโดยคั่นด้วยเครื่องหมายจุด ซึ่งสามารถแยกเป็น 2 ส่วนย่อยคือ ส่วนแรกเป็นหมายเลขของเครือข่าย (Network Address) และส่วนที่สองเป็นหมายเลขของเครื่องลูกข่าย (Host Address) ทั้งนี้การแบ่งส่วนจะเป็นไปตามการแบ่งระดับชั้นของเครือข่ายเรียกว่า Network class ซึ่งการกำหนดให้มี Network class นี้ก็เพื่อให้สามารถแจกจ่าย IP Address ให้กับเครือข่ายต่างๆ ได้อย่างเหมาะสม เพราะในแต่ละเครือข่ายมักจะแตกต่างกันบ้าง บางเครือข่ายก็มีจำนวนเครื่องลูกข่ายมาก บางเครือข่ายมีน้อยแต่มีเครือข่ายย่อยๆ ในเครือข่ายหลักจำนวนมาก ฉะนั้นถ้าไม่มีการจัดลำดับของเครือข่ายให้ดี IP Address ก็จะถูกใช้อย่างสิ้นเปลืองและใช้งานได้ไม่ได้เต็มจำนวนที่มี ลำดับชั้นของเครือข่ายแบ่งได้เป็น 5 ลำดับคือ class A, B, C, D และ E [10]

ในแต่ละ Network class IP Address ทั้ง 32 บิตจะถูกกำหนดเป็นหมายเลขของเครือข่ายและหมายเลขของเครื่องลูกข่าย โดยมีเงื่อนไขดังนี้

- Class A เป็น IP Address ที่มีบิตแรกของไบต์แรกสุดเป็น 0 เสมอ และเป็น IP Address ที่เริ่มตั้งแต่ 0-127 จะกำหนดให้ให้กับเครือข่ายขนาดใหญ่ เพราะ 1 เครือข่ายสามารถมีเครื่องลูกข่ายได้กว่า 16 ล้านเครื่อง IP Address จะเป็นลักษณะ net.host.host.host
- Class B เป็น IP Address ที่มี 2 บิตแรกของไบต์แรกสุดจะเป็น 1 และ 0 เสมอ และเป็น IP Address ที่เริ่มตั้งแต่ 128-191 จะกำหนดให้กับเครือข่ายที่มีขนาดใหญ่เช่นกันแต่เล็กกว่า Class A ในแต่ละเครือข่ายของ Class B สามารถมีเครื่องลูกข่ายได้ 2^{16}-จำนวนแอดเดรสที่ใช้ควบคุมระบบเครือข่าย = 64516 เครื่อง IP Address จะเป็นลักษณะ net.net.host.host
- Class C เป็น IP Address ที่มี 3 บิตแรกของไบต์แรกสุดจะเป็น 1, 1 และ 0 เสมอ และเป็น IP Address ที่เริ่มตั้งแต่ 192-223 จะกำหนดให้กับเครือข่ายที่เป็นองค์กรทั่วไป ซึ่งส่วนใหญ่จะเป็นองค์กรขนาดกลางถึงเล็ก ในแต่ละเครือข่ายมีเครื่องลูกข่ายไม่เกิน 2^{8}-จำนวนแอดเดรสที่ใช้ควบคุมระบบเครือข่าย = 254 เครื่อง IP Address จะเป็นลักษณะ net.net.net.host
- Class D เป็นการกำหนด IP Address สำรองไว้สำหรับส่งข้อมูลแบบ multicast ซึ่งจะไม่มีการแจกจ่ายให้ใช้งานทั่วไป
- Class E เป็น IP Address พิเศษ ที่ใช้สำหรับงานทดสอบและพัฒนา ไม่มีการกำหนดให้ใช้งานทั่วไป

2.1.4 Subnet

ปัจจุบันการกำหนด IP Address ของเครือข่ายประเภท Class A และ Class B นั้น ไม่มีการกำหนดให้แล้ว เนื่องจากแทบไม่มีเครือข่ายใดที่มีความจำเป็นต้องใช้ Address มากขนาดนั้น คงเหลือแต่Class C เท่านั้นที่กำหนดให้แต่ละบริษัทหรือหน่วยงานต่างๆ และถึงแม้ว่าจะเป็นเครือข่ายที่ใช้ IP Address ประเภท Class C ก็ตาม ส่วนมากก็ไม่มีใครเชื่อมต่อเครื่องลูกข่ายมากถึง 254 เครื่องในหนึ่งเครือข่าย หากทุกๆ เครือข่ายมีอุปกรณ์หรือเครื่องติดตั้งใช้งานในเครือข่ายนั้นเพียงไม่กี่เครื่อง แต่จำเป็นต้องกำหนด IP Address ให้ใช้งานในเครือข่ายนั้น 254 เครื่อง จะเกิดปัญหามี IP Address ที่ไม่ได้ถูกใช้งานเป็นจำนวนมาก และไม่สามารถนำ IP Address ที่ไม่ได้ใช้งานนี้ไปให้ผู้อื่นใช้ได้ ดังนั้นการทำ Subnet หรือ Sub Network จึงเกิดขึ้น เพื่อแบ่งเครือข่ายออกเป็นเครือข่ายย่อยๆ และทำให้การกำหนดใช้งาน IP Address ที่ได้รับมาสามารถแบ่งออกเป็นส่วนๆ เหมาะสมกับอุปกรณ์ในแต่ละเครือข่ายได้ ทั้งยังแบ่ง IP Address ส่วนที่ไม่ได้ใช้ให้หน่วยงานอื่นหรือเครือข่ายอื่นได้อีกด้วย

Class C Subnet

เนื่องจาก IP Address ที่เราใช้งานกันอยู่ทุกวันนี้จะเป็นเครือข่าย Class เกือบทั้งหมด ซึ่งหมายถึง เราจะใช้ 8 บิตขวาสุดของ IP Address ในการกำหนด Host Address หรือเครื่องลูกข่าย ดังนั้นการ Subnet ใน Class C ก็คือการแบ่งข้อมูล 8 บิต ของ Host Address นี้ออกเป็น Subnet address และ Host Address ใหม่นั่นเอง ซึ่งใน Class C นี้เราจะแบ่ง Subnet ได้ทั้งหมด 5 แบบ ดังตารางต่อไปนี้

ตารางที่ 2 แสดง Subnet ทั้งหมดของ Class C ซึ่งหากไม่มีการทำ Subnet ค่า Subnet Mask จะเป็น 255.255.255.0

จำนวนเครือข่ายย่อย (Subnet)	จำนวนเครื่องลูกข่าย (Host Address)	Subnet Mask เลขฐานสิบ	จำนวนบิตที่ใช้เป็น Subnet
2	62	255.255.255.192	2
6	30	255.255.255.224	3
14	14	225.255.255.240	4
30	6	255.255.255.248	5
32	2	255.255.255.252	6

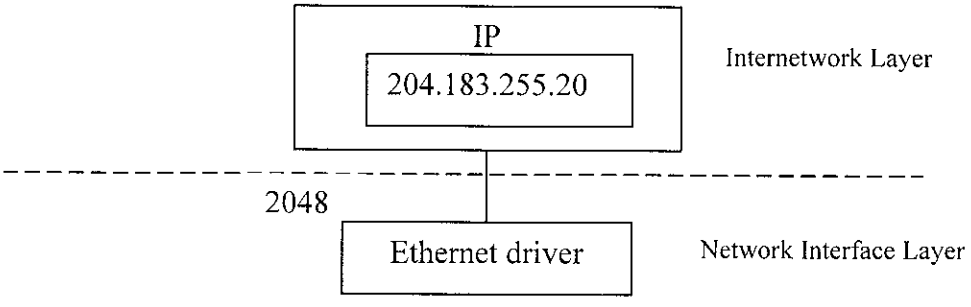
สำหรับ Subnet 0 คือทุกบิตของ Subnet มีค่าเป็น “0” ทั้งหมด และ Subnet “7” ที่ทุกบิตมีค่าเป็น “1” หมดจะถูกจองไว้สำหรับควบคุมระบบเครือข่าย และ Subnet ที่มีจำนวนเครือข่ายย่อยเท่ากับ

หนึ่งจะใช้งานไม่ได้เนื่องจากมีมีจำนวนบิตเหลือให้ไปใช้ทำ Subnet ดังนั้นจาก Host Address ขนาด 8 บิต จึงเหลือนำมาใช้ทำ Subnet ได้เพียง 5 แบบเท่านั้น เพราะถูกหักออกไป 3 Subnet จากเหตุผลดังกล่าว

ในแต่ละ Subnet ก็จะมีการจอง Host address ไว้ 2 ตำแหน่ง คือ Host Address ที่มีค่าเป็น "0" หมดทุกบิต เอาไว้ใช้สำหรับหมายเลขประจำเครือข่าย (Network Address) และ Host Address ที่มีค่าเป็น "1" หมดทุกบิต จะถูกจองไว้สำหรับเป็น Broadcast Address ดังนั้นในทุกๆ แบบของ Subnet จำนวนเครื่องลูกข่ายจึงถูกหักออก 2 Address คือ Address ที่เป็น "0" ทั้งหมดทุกบิต และที่เป็น "1" หมดทุกบิต เหลือจำนวนเครื่องลูกข่ายสูงสุดดังที่แสดงในตารางข้างต้น [8]

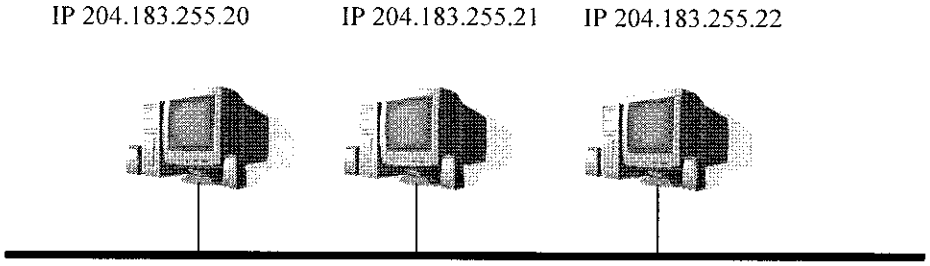
2.1.5 การ Bind IP Address

เมื่อได้กำหนดหมายเลข IP Address ให้กับจุดเชื่อมต่อ เช่น Network card เรียบร้อยแล้ว ที่เครื่อง Server จะต้องมีการ bind หรือ ผนวกค่า IP Address ดังกล่าวเข้ากับ Ethernet driver เพื่ออ้างอิงหมายเลข IP กับฮาร์ดแวร์ ให้ทำหน้าที่ติดต่อส่งข้อมูลในระดับ Network interface ได้ต่อไปดังตัวอย่างในรูปที่ 2-10



รูปที่ 2-10 แสดงการ bind IP Address หมายเลข 204.183.255.20 เข้ากับ Ethernet driver ซึ่งเป็น Network interface driver ทำให้ IP สามารถสื่อสารกับเครือข่ายได้ ส่วนหมายเลข 2048 เป็น interface identifier

จากรูปจะแสดงค่า bind IP Address 204.183.255.20 เข้ากับ Ethernet driver (ในกรณีนี้ใช้เครือข่ายแบบ Ethernet) โปรโตคอล IP จะใช้ค่า IP Address นี้ในการติดต่อกันและผ่านฮาร์ดแวร์ที่ถูก bind ไว้อีกต่อหนึ่ง ค่าหมายเลขฮาร์ดแวร์ก็ได้แก่ MAC Address ที่มีประจำอยู่บน Network card ซึ่งจะไม่ได้ใช้งานอ้างอิงโดยตรง แต่จะผ่านหมายเลข IP Address แทน Ethernet driver



รูปที่ 2-11 แสดงเครื่องคอมพิวเตอร์ในเครือข่าย Ethernet มีหมายเลข IP Address แต่ละเครื่องที่ LAN card เป็น Network interface

จากรูปตัวอย่าง เป็นรูปเครือข่าย Ethernet อย่างง่าย จะเห็นว่ามีการกำหนดหมายเลข IP Address ให้กับ LAN card ที่เป็นจุดเชื่อมต่อเข้ากับเครือข่ายทุกจุด แต่เนื่องจากเครื่องทุกเครื่องมีจุดเชื่อมต่อเพียงจุดเดียว ทำให้เราสามารถอ้างอิงเครื่องกับหมายเลข IP Address นั้นได้ตรงกัน

2.2 ไมโครคอนโทรลเลอร์ MCS-51

ไมโครคอนโทรลเลอร์ (Microcontroller) เป็นอุปกรณ์อิเล็กทรอนิกส์แบบหนึ่งที่รวมเอาหน่วยประมวลผล หน่วยคำนวณทางคณิตศาสตร์และลอจิก วงจรรับสัญญาณอินพุต วงจรขับสัญญาณเอาท์พุต หน่วยความจำ วงจรการกำเนิดสัญญาณนาฬิกาไว้ด้วยกัน ทำให้สามารถนำไปใช้งานแทนวงจรอิเล็กทรอนิกส์ที่ซับซ้อนได้เป็นอย่างดี ช่วยลดจำนวนอุปกรณ์และขนาดของระบบ ในขณะที่มีขีดความสามารถสูงขึ้น ภายใต้งบประมาณที่เหมาะสม

ไมโครคอนโทรลเลอร์มาจากคำ 2 คำรวมกันคือ “ไมโคร” (micro) ซึ่งหมายถึง ไมโครโปรเซสเซอร์ (microprocessor) ซึ่งเป็นอุปกรณ์ประมวลผลข้อมูลขนาดเล็ก ภายในประกอบด้วย หน่วยประมวลผลกลางหรือ ซีพียู (CPU : Central Processing Unit) หน่วยความจำทางคณิตศาสตร์และลอจิก (ALU : Arithmatic Logic Unit) วงจรเชื่อมต่อหน่วยความจำ และวงจรสัญญาณนาฬิกา อีกคำหนึ่งคือคำว่า “คอนโทรลเลอร์”(controller) หมายถึงอุปกรณ์ควบคุม ดังนั้นไมโครคอนโทรลเลอร์จึงเป็นอุปกรณ์ที่ใช้ในการควบคุม โดยที่มารถเขียนโปรแกรมเพื่อกำหนดรูปแบบการควบคุมได้อย่างอิสระ[4]

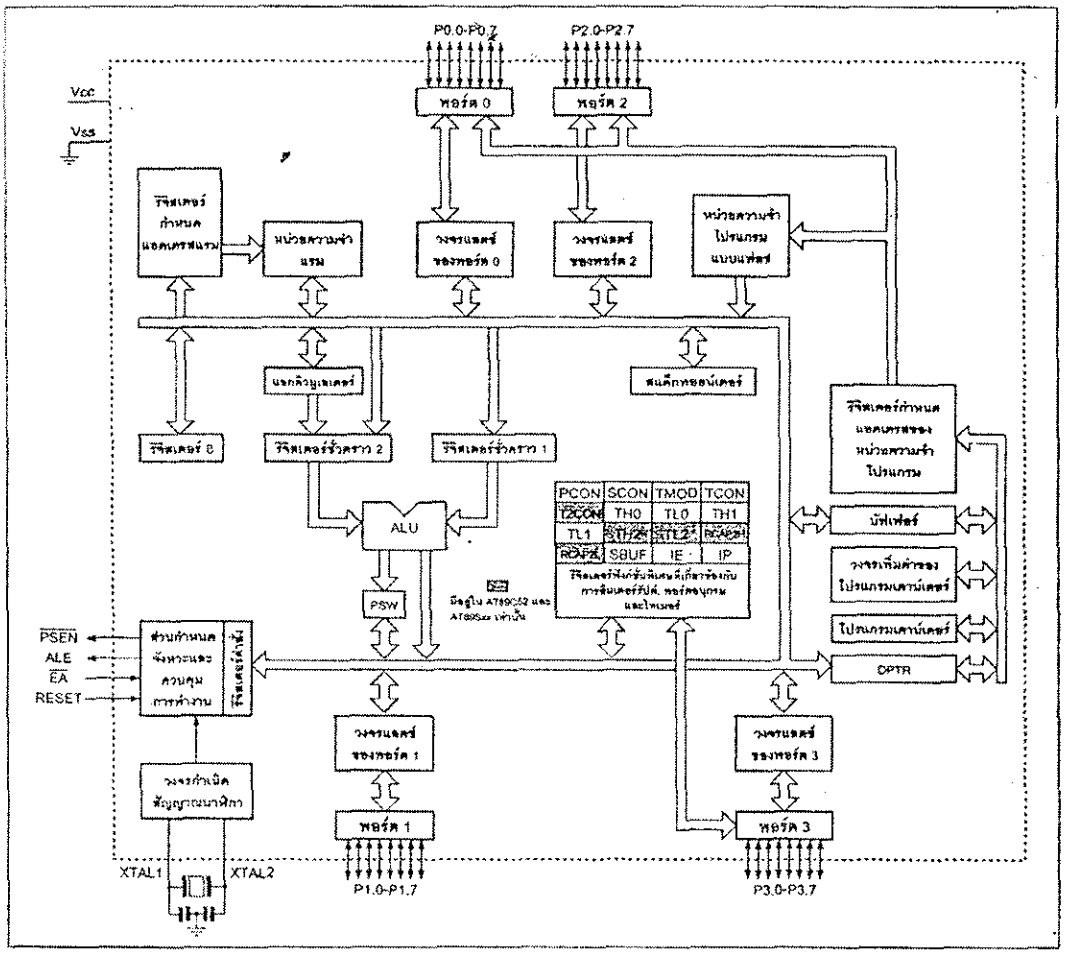
2.2.1 การจัดขาของไมโครคอนโทรลเลอร์ MCS-51

ไมโครคอนโทรลเลอร์ MCS-51 ทุกเบอร์จะมีสถาปัตยกรรมและขาใช้งานพื้นฐานเหมือนกัน ดังแสดงในรูปที่ 2-12 และ 2-13 โดยมีรายละเอียดขั้นต้นดังนี้

ขา **Vcc** ใช้สำหรับต่อไฟเลี้ยง +5V

ขา **GND** เป็นขากราวด์ สำหรับต่อกับกราวด์ของระบบ

ขาพอร์ต 0 **(P0.0-P0.7)** มี 8 ขา แต่ละขาสามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาต์พุต สำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ต 0 ขาใดขาหนึ่งเป็นอินพุตสามารถทำได้โดยการเขียนข้อมูล "1" ไปยังแต่ละบิตของพอร์ตที่ต้องการติดต่อด้วย ส่งผลให้ขาพอร์ตนั้นมีสถานะปล่อยลอย (float) จึงมีอินพุตอิมพีแดนซ์สูงสามารถใช้งานเป็นขาพอร์ตอินพุตได้ นอกจากนั้นขาพอร์ตนี้ยังถูกใช้งานในการติดต่อกับขาแอดเดรสไบต์ต่ำของหน่วยความจำภายนอก (A0-A7) และขาข้อมูล (D0-D7) โดยใช้กระบวนการมัลติเพล็กซ์เข้าช่วย เพื่อสลับการทำงานเป็นได้ทั้งขาติดต่อแอดเดรสและขาข้อมูล



รูปที่ 2-12 รายละเอียดโครงสร้างหลักของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

ขาพอร์ต 1 **(P1.0-P1.7)** มี 8 ขา แต่ละขาสามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาท์พุต สำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ตใดเป็นอินพุต สามารถทำได้โดยการเขียนข้อมูล "1" ไปยังแต่ละบิตของพอร์ตที่ต้องการติดต่อด้วย นอกจากนั้นในอนุกรม AT89Sxx จะใช้ขา P1.0 เป็นขาอินพุตสำหรับนับค่าของไทม์เมอร์ 2 และ P1.1 เป็นขาอินพุตทริกเกอร์ของไทม์เมอร์ 2 ในขณะที่ขา P1.4-P1.7 เป็นขาสำหรับเชื่อมต่อแบบ SPI เพื่อทำการโปรแกรมข้อมูลในระบบ

ขาพอร์ต 2 **(P2.0-P2.7)** มี 8 ขา แต่ละขาสามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาต์พุต สำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ตใดเป็นอินพุตสามารถได้โดยการเขียนข้อมูล "1" ไปยังแต่ละบิตของพอร์ตที่ต้องการติดต่อด้วย ส่งผลให้ขาพอร์ตนั้นมีสถานะปล่อยลอย (float) จึงมีอินพุตอิมพีแดนซ์สูง สามารถใช้งานเป็นขาพอร์ตอินพุตได้ นอกจากนั้นขาพอร์ตนี้ยังถูกใช้งานในการติดต่อกับขาแอดเดรสไบต์สูงของหน่วยความจำภายนอก (A8-A15)

ขาพอร์ต 3 **(P3.0-P3.7)** มี 8 ขา แต่ละขาสามารถกำหนดให้เป็นได้ทั้งอินพุตและเอาต์พุต สำหรับใช้งานทั่วไป ถ้าหากต้องการกำหนดให้ขาพอร์ตใดเป็นอินพุตสามารถได้โดยการเขียนข้อมูล "1" ไปยังแต่ละบิตของพอร์ตที่ต้องการติดต่อด้วย ส่งผลให้ขาพอร์ตนั้นมีสถานะปล่อยลอย (float) จึงมีอินพุตอิมพีแดนซ์สูง สามารถใช้งานเป็นขาพอร์ตอินพุตได้ นอกจากนั้นขาพอร์ต 3 ยังเป็นขาที่มีหน้าที่การใช้งานพิเศษ ดังมีรายละเอียดขั้นต้นต่อไปนี้

P3.0 ใช้เป็นขาอินพุตสำหรับรับข้อมูลจากการสื่อสารแบบอนุกรม หรือขา RxD

P3.1 ใช้เป็นขาอินพุตสำหรับส่งข้อมูลจากการสื่อสารแบบอนุกรม หรือขา TxD

P3.2 ใช้เป็นขาอินพุตรับสัญญาณอินเตอร์รัปจากภายนอกช่อง 0 หรือขา INT0

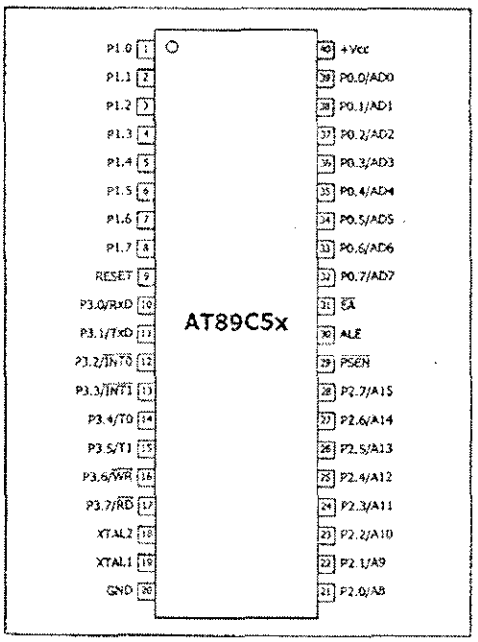
P3.3 ใช้เป็นขาอินพุตรับสัญญาณอินเตอร์รัปจากภายนอกช่อง 1 หรือขา INT1

P3.4 ใช้เป็นขาอินพุตสำหรับสัญญาณ ไทม์เมอร์จากภายนอกช่อง 0 หรือขา T0

P3.5 ใช้เป็นขาอินพุตสำหรับรับสัญญาณอินเตอร์รัปจากภายนอกช่อง 1 หรือขา T1

P3.6 ใช้เป็นขาสัญญาณ WR ในกรณีที่ใช้เชื่อมต่อกับหน่วยความจำภายนอก

P3.7 ใช้เป็นขาสัญญาณ RD ในกรณีที่ใช้เชื่อมต่อกับหน่วยความจำภายนอก



รูปที่ 2-13 การจัดขามาตรฐานของไมโครคอนโทรลเลอร์ MCS-51 ในอนุกรม AT89C5x

ขารีเซต **(Reset)** ใช้ในการรีเซตการทำงานของไมโครคอนโทรลเลอร์ โดยในการป้อนสัญญาณเพื่อรีเซตสถานะที่ขานี้ต้องอยู่ในระดับรีเซตอย่างน้อย 2 แมชีนไซเคิล โดยที่วงจรกำเนิดสัญญาณนาฬิกายังคงทำงานต่อเนื่องไปอย่างเป็นปกติ

ขา $\overline{\textbf{ALE}}$**/PROG** (Address Latch Enable/Prigram pulse input) เป็นขาที่ใช้ในการควบคุมการแลตซ์ของขาพอร์ต 0 เมื่อมีการใช้งานหน่วยความจำภายนอก นอกจากนั้นขานี้ยังใช้เป็นขาสำหรับพัลส์ของการโปรแกรมสำหรับโปรแกรมข้อมูลลงในไมโครคอนโทรลเลอร์ MCS-51 ในรุ่นที่มีหน่วยความจำแบบอีพรอม

ขา $\overline{\textbf{PSEN}}$ (Program Store Enable) ขานี้ใช้ในการส่งสัญญาณเพื่อร้องขอติดต่อกับหน่วยความจำโปรแกรมภายนอก เมื่อไมโครคอนโทรลเลอร์ต้องการอ่านข้อมูลจากหน่วยความจำโปรแกรมภายนอกตัวไมโครคอนโทรลเลอร์ต้องการอ่านข้อมูลจากหน่วยความจำโปรแกรมภายนอกตัวไมโครคอนโทรลเลอร์จะส่งสัญญาณออกมาที่ขานี้ 2 ครั้งในแต่ละแมชีนไซเกิล แต่ถ้าหากติดต่อกับหน่วยความจำข้อมูลภายนอกขานี้จะไม่มีการส่งสัญญาณใด ๆ ออกมา

ขา $\overline{\textbf{EA}}$**/Vpp** (External Access enable/Programming voltage input) ใช้สำหรับเลือกการติดต่อหน่วยความจำโปรแกรมจากภายนอกหรือภายในตัวไมโครคอนโทรลเลอร์ ถ้าหากขานี้เป็น "0" เป็นการเลือกให้ไมโครคอนโทรลเลอร์ติดต่อกับหน่วยความจำโปรแกรมภายนอก แต่ถ้าหากขานี้เป็น "1" เป็นการเลือกให้ไมโครคอนโทรลเลอร์ติดต่อกับหน่วยความจำภายในตัวไมโครคอนโทรลเลอร์ นอกจากนี้ที่ขานี้ยังใช้เป็นขาอินพุตสำหรับรับแรงดันไฟสูงสำหรับการโปรแกรมหน่วยความจำภายในไมโครคอนโทรลเลอร์ สำหรับไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชต้องการแรงดันสำหรับการโปรแกรม +12V

ขา **XTAL1** และ **XTAL2** เป็นขาสำหรับติดต่อคริสตอลเพื่อสร้างสัญญาณนาฬิกาในการกำหนดจังหวะการทำงานของไมโครคอนโทรลเลอร์ [4]

2.2.2 โครงสร้างและการทำงานของพอร์ต

ไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชมีพอร์ตให้ใช้งานทั้งสิ้น 4 พอร์ตคือ พอร์ต 0 ถึง พอร์ต 3 แต่ละพอร์ตมีขนาด 8 บิต เป็นพอร์ตแบบ 2 ทิศทางกล่าวคือ สามารถเป็นได้ทั้งอินพุตสำหรับรับสัญญาณข้อมูลเข้าและเอาต์พุตสำหรับส่งสัญญาณข้อมูลออก ทุกพอร์ตของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชมีวงแลตช์และวงจรขับตลอดจนบัฟเฟอร์อินพุต ดังแสดงให้เห็นสถาปัตยกรรมรูปที่ 2-12

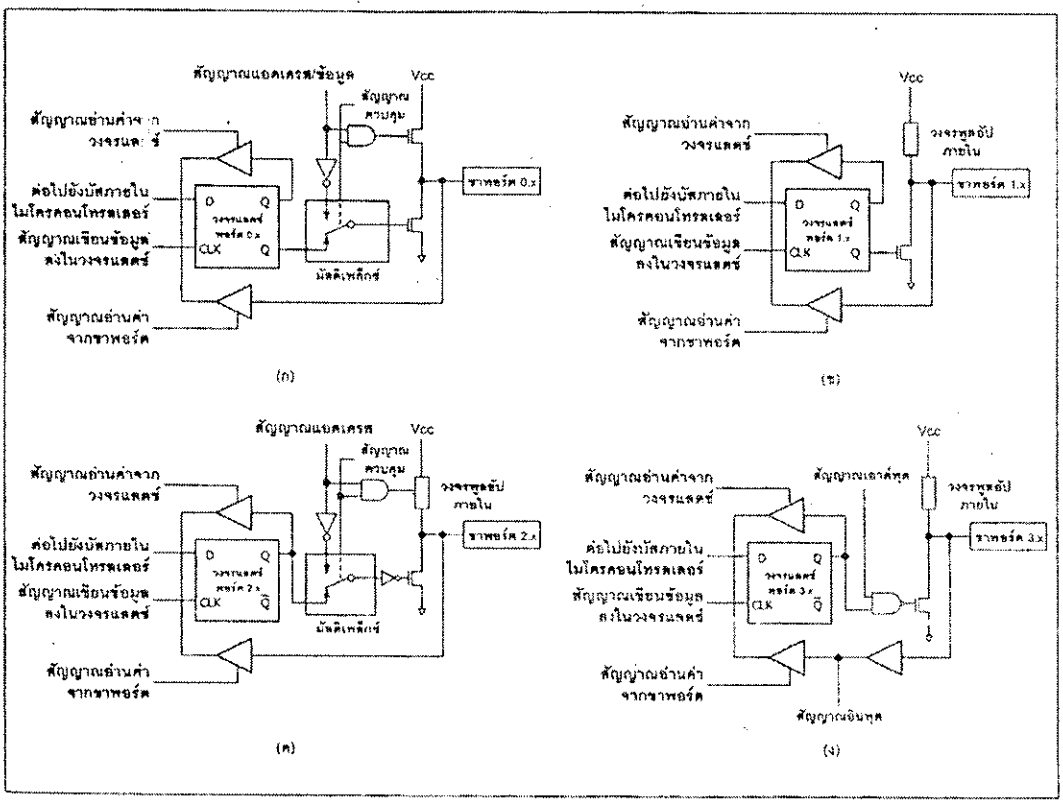
ที่พอร์ต 0 และ พอร์ต 2 จะใช้งานเป็นพอร์ตอินพุตและเอาต์พุตสำหรับงานทั่วไป และใช้ในการติดต่อกับหน่วยความจำภายนอก สำหรับพอร์ต 3 ทั้งพอร์ตและพอร์ต 1 บางขานอกจากจะใช้เป็นขา

พอร์ตอินพุตเอาต์พุตตามปกติแล้ว ยังสามารถใช้งานในหน้าที่พิเศษได้อีก ขึ้นอยู่กับว่าเป็นไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชเบอร์ใด

ในรูปที่ 2-14 แสดงวงจรภายในของแต่ละพอร์ตของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช โดยในรูปที่ 2-14 (ก) เป็นวงจรของพอร์ต 0 วงจรแลตช์ของแต่ละบิตในแต่ละพอร์ตก็คือวงจรดีฟลิฟลอปนั่นเอง การอ่านค่าสถานะของพอร์ตและสถานะของวงจรแลตช์สามารถกระทำได้อย่างอิสระด้วยสัญญาณที่แยกออกจากกัน นั่นคือสัญญาณอ่านข้อมูลจากขาพอร์ต และสัญญาณอ่านข้อมูลจากวงจรแลตช์ ส่วนการเขียนข้อมูลมายังพอร์ตต้องส่งสัญญาณมายังขา CLK ของดีฟลิฟลอปในขณะที่ข้อมูลจะผ่านมาทางขาบัสข้อมูลภายในเข้าสู่ขา D ของดีฟลิฟลอป

ที่พอร์ตนี้มีวงจรมัลติเพล็กซ์สำหรับกำหนดลักษณะการทำงานของพอร์ตว่า ต้องการใช้งานเป็นขาพอร์ตอินพุตเอาต์พุตปกติหรือใช้ในการติดต่อกับหน่วยความจำภายนอกไมโครคอนโทรลเลอร์

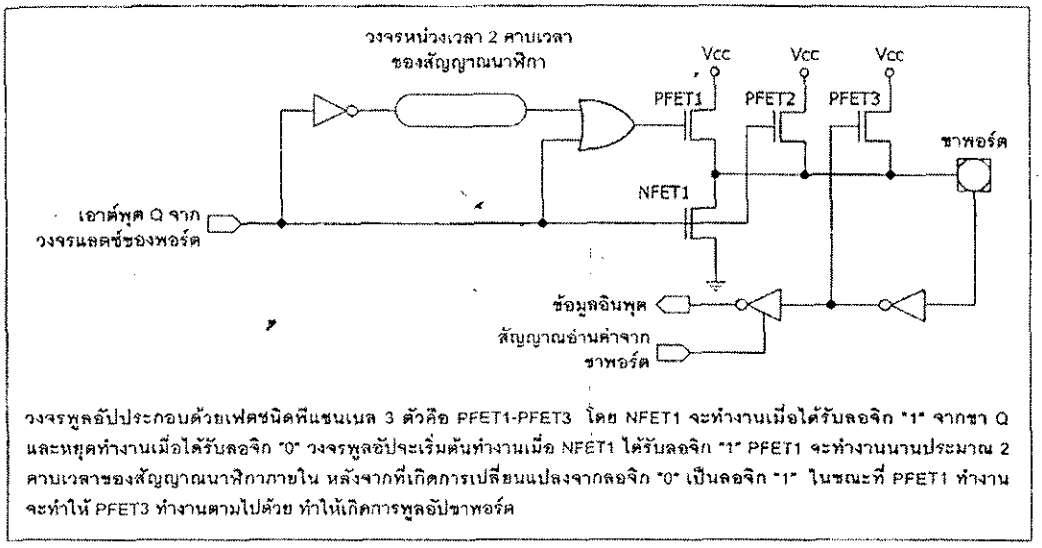
เนื่องจากที่ขาพอร์ต 0 ไม่มีวงจรพูลอัปภายใน หากมีการนำพอร์ต 0 ไปใช้งานเป็นพอร์ตอินพุตจะต้องต่อตัวต้านทานพูลอัปภายนอกเข้าที่ขาพอร์ต 0 ทุกขาด้วย



รูปที่ 2-14 วงจรภายในของพอร์ตทุกพอร์ตในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

ในรูปที่ 2-14 (ข) เป็นวงจรของพอร์ต 1 ซึ่งมีลักษณะโดยทั่วไปคล้ายกับพอร์ต 0 หากแต่ไม่มีวงจรมัลติเพล็กซ์ เนื่องจากพอร์ตนี้จะไม่ใช้ในการติดต่อกับหน่วยความจำภายนอก แต่จะมีวงจรพูลอัปภายในที่แต่ละบิตของพอร์ตนี้แทน สำหรับรายละเอียดของวงจรพูลอัปแสดงในรูปที่ 2-15

ในรูปที่ 2-14 (ค) เป็นวงจรภายในของพอร์ต 2 จะคล้ายกับพอร์ต 0 มาก ต่างกันเพียงมีวงจรพูลอัปเพิ่มเติมเข้ามา ส่วนในรูปที่ 2-14 (ง) เป็นวงจรภายในของพอร์ต 3 จะเห็นได้ว่าคล้ายกับพอร์ต 1 มีการเพิ่มเติมวงจรบัฟเฟอร์และวงจรอินพุตเอาต์พุตเมื่อทำงานในฟังก์ชั่นพิเศษเข้ามา เนื่องจากพอร์ต 3 สามารถนำไปใช้งานในหน้าที่พิเศษได้ทุกขา



รูปที่ 2-15 วงจรพูลอัปภายในพอร์ตของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

2.2.3 พอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 มีวงจรการสื่อสารอนุกรมแบบฟูลดูเพล็กซ์ 1 ชุด (วงจรสื่อสารแบบฟูลดูเพล็กซ์ หมายถึง วงจรสื่อสารที่สามารถทำการรับและส่งข้อมูลในลักษณะ 2 ทิศทางได้ในเวลาเดียวกัน) โดยใช้ขาสัญญาณของพอร์ต 3 คือ ขา P3.0 เป็นขารับข้อมูลเข้าหรือ RxD และขา P3.1 เป็นขาส่งข้อมูลออกหรือ TxD โดยวงจรการสื่อสารข้อมูลแบบอนุกรมของไมโครคอนโทรลเลอร์ตระกูล MCS-51 แบบแฟลชเป็นแบบอะซิงโครนัส ปกติแล้วพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 จะใช้ในการติดต่อสื่อสารกับพอร์ตอนุกรมของคอมพิวเตอร์ โดยใช้มาตรฐานของ RS-232 แต่ในปัจจุบันสามารถสามารถติดต่อกันในมาตรฐาน RS-422 หรือ RS-485 ได้แล้ว โดยใช้ไอซีพิเศษที่ทำหน้าที่ในการแปลงสัญญาณการสื่อสารดังกล่าว

หลักการสื่อสารข้อมูลผ่านพอร์ตอนุกรม

การสื่อสารข้อมูลขนาด 8 บิตของไมโครคอนโทรลเลอร์ MCS-51 ซึ่งสามารถกำหนดค่าอัตราบอดได้จากการตั้งค่า TL1 และ TH1 ของไทม์เมอร์ 1 โดยในการทดลองนี้จะใช้อัตราบอดเท่ากับ 9600 บิตต่อวินาที ซึ่งเป็นรูปแบบมาตรฐานที่ใช้ในการสื่อสารกับเครื่องคอมพิวเตอร์

การสื่อสารข้อมูลผ่านพอร์ตอนุกรมในไมโครคอนโทรลเลอร์นั้น ทำได้ 2 วิธี คือ

1. ใช้อินเตอร์รัปต์ เป็นวิธีที่ให้ผลการทำงานเร็วที่สุด แต่มีความยุ่งยากในการทำงานมากกว่า เนื่องจากตำแหน่งของการอินเตอร์รัปต์ทั้งการรับและการส่งข้อมูลนั้นอยู่ที่ตำแหน่งเดียวกัน ต้องพิจารณาจากแฟลก TI หรือ RI ก่อนว่าเกิดการอินเตอร์รัปต์จากสาเหตุใด และต้องพิจารณาการใช้รีจิสเตอร์ในช่วงเวลานั้น ๆ ด้วยว่ามีโอกาสซ้อนทับกันหรือไม่ ทำให้โปรแกรมของการทำงานในส่วนนี้มีความซับซ้อนมากกว่า

2. วนโปรแกรมตรวจสอบแฟลก เป็นวิธีที่มีความซับซ้อนน้อยกว่า โดยเขียนโปรแกรมให้วนตรวจสอบแฟลกอยู่ตลอดเวลา จนกว่าจะเกิดการเปลี่ยนแปลง ยกตัวอย่าง เมื่อต้องการตรวจสอบการส่งข้อมูล ให้ทำการวนตรวจสอบแฟลก TI ว่าถูกเซตหรือไม่ เมื่อถูกเซต แสดงว่า มีการส่งข้อมูลเกิดขึ้นเรียบร้อยแล้ว จากนั้นให้ทำการเคลียร์แฟลก TI แล้วทำการส่งข้อมูลตัวถัดไป หรือทำงานในคำสั่งต่อไปได้

ในกรณีที่ต้องการตรวจสอบการรับข้อมูล ให้ทำการตรวจสอบแฟลก RI ว่าถูกเซตหรือไม่ เมื่อตรวจสอบได้ว่าถูกเซต แสดงว่า เกิดการรับข้อมูลขึ้น ให้ทำการเคลียร์แฟลก RI แล้วนำค่าจากรีจิสเตอร์ SBUF มาใช้ได้ทันที แต่วิธีการนี้มีข้อเสียตรงที่เป็นการทำงานแบบเรียงลำดับ ทำให้ขั้นตอนในการทำงานช้ากว่าการใช้อินเตอร์รัปต์

การแสดงค่าบนเทอร์มินอลหรือเครื่องคอมพิวเตอร์นั้นจะใช้ข้อมูลรหัส ASCII มาตรฐานในการรับส่งข้อมูลทุกอย่าง เช่น การขึ้นบรรทัดใหม่ใช้ค่า 0AH (ค่า LF) และการเลื่อนตำแหน่งเคอร์เซอร์ไปที่ตำแหน่งซ้ายสุด ให้ใช้ค่า 0DH (ค่า CR) การส่งตัวเลข ตัวอักษร ก็ใช้ค่าตามรหัส ASCII ด้วย

การเขียนโปรแกรมเพื่อส่งข้อมูลจะนำข้อมูลจากค่าที่เก็บอยู่ภายในตัวโปรแกรมเอง จนกว่าจะพบค่า 0FFH จึงจะหยุด ทำให้ไม่มีการจำกัดจำนวนข้อมูลในการส่งแต่ละครั้ง ด้วยวิธีนี้จึงทำให้การส่งข้อมูลมีความยืดหยุ่นในเรื่องขนาดของข้อมูลสูง

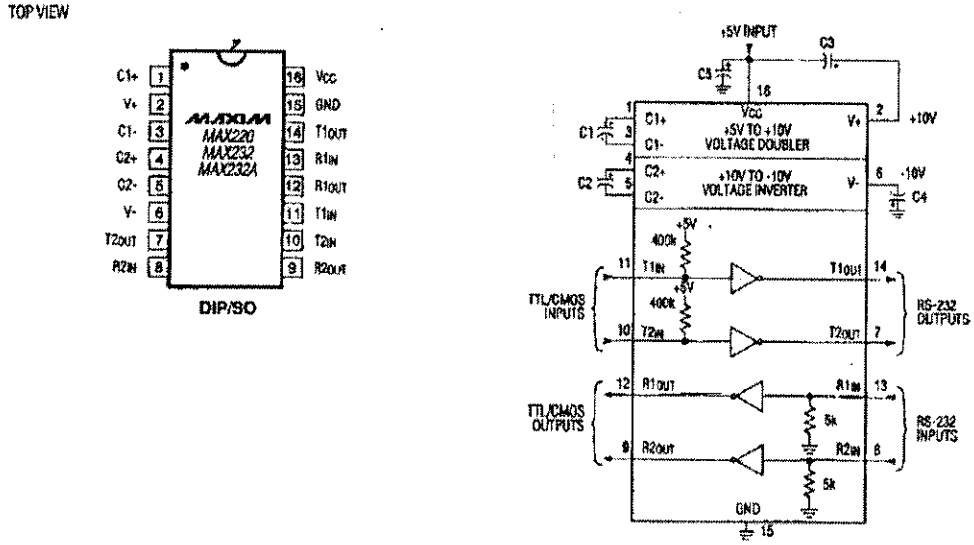
อย่างไรก็ตาม หัวใจสำคัญของการสื่อสารข้อมูลผ่านพอร์ตอนุกรมคือ การกำหนดอัตราบอดและรูปแบบของข้อมูลว่า มีจำนวนบิตเริ่มต้น, บิตของข้อมูล, บิตหยุด หรือว่ามีการตรวจสอบบิตพาริตี้หรือไม่ ถ้าหากข้อกำหนดเหล่านี้ในตัวส่งและตัวรับไม่ตรงกัน จะทำให้การถ่ายทอดข้อมูลเกิดความผิดพลาดได้อย่างง่ายดาย ส่งผลให้การสื่อสารข้อมูลล้มเหลวอย่างสิ้นเชิง

การเชื่อมต่อกับพอร์ตอนุกรมของคอมพิวเตอร์

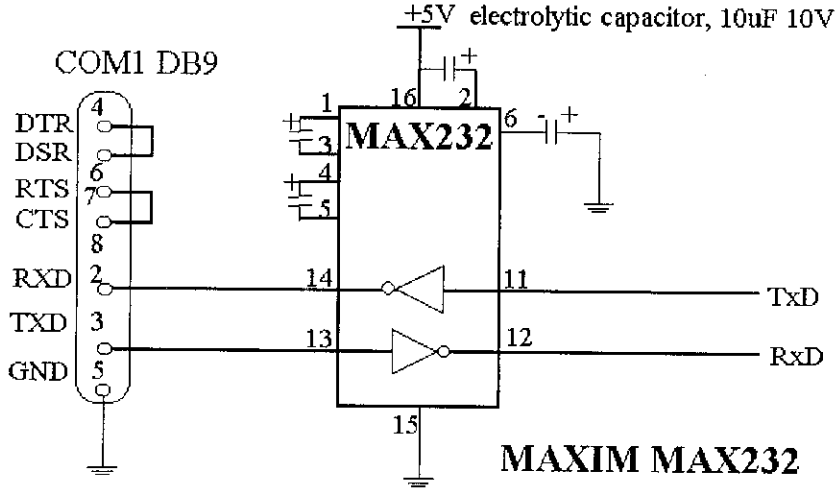
การใช้งานวงจรพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ MCS-51 มักนิยมใช้ในการติดต่อเพื่อแลกเปลี่ยนข้อมูลกับคอมพิวเตอร์ผ่านทางพอร์ตอนุกรมในมาตรฐาน RS-232 เป็นส่วนใหญ่ แต่เนื่องจากระดับสัญญาณของพอร์ตอนุกรม RS-232 มีระดับตั้งแต่ ±3 ถึง ±12V ในขณะที่ระดับสัญญาณของไมโครคอนโทรลเลอร์ MCS-51 อยู่ในระดับทีทีแอล ดังนั้นจึงไม่สามารถเชื่อมต่อพอร์ต

อนุกรมของไมโครคอนโทรลเลอร์ MCS-51 เข้ากับพอร์ตอนุกรมของคอมพิวเตอร์ได้โดยตรง จึงต้องอาศัยการเชื่อมต่อผ่านทางไอซีพิเศษที่ทำหน้าที่ในการแปลงระดับสัญญาณ

ไอซีที่ทำหน้าที่ในการแปลงระดับสัญญาณนี้ต้องทำการแปลงข้อมูลส่งของไมโครคอนโทรลเลอร์ MCS-51 จากระดับทีทีแอลไปเป็นระดับของ RS-232 และทำการแปลงข้อมูลรับจากคอมพิวเตอร์จากระดับของ RS-232 เป็นระดับทีทีแอลเพื่อให้สามารถถ่ายทอดไปยังไมโครคอนโทรลเลอร์ MCS-51 ได้อย่างสมบูรณ์ ไอซีดังกล่าวมีด้วยกันหลายเบอร์จากหลายผู้ผลิต อาทิ MAX232 จาก MAXIM หรือ ICL232 จาก HARRIS เป็นต้น ในรูปที่ 2-16 แสดงการจัดขาของไอซี ICL232 ซึ่งใช้ในการแปลงสัญญาณ RS-232 ส่วนวงจรของการต่อกับไมโครคอนโทรลเลอร์ MCS-51 [9] แสดงในรูปที่ 2-17



รูปที่ 2-16 รายละเอียดเบื้องต้นของไอซีแปลงสัญญาณเพื่อเชื่อมต่อกับพอร์ตอนุกรมของคอมพิวเตอร์



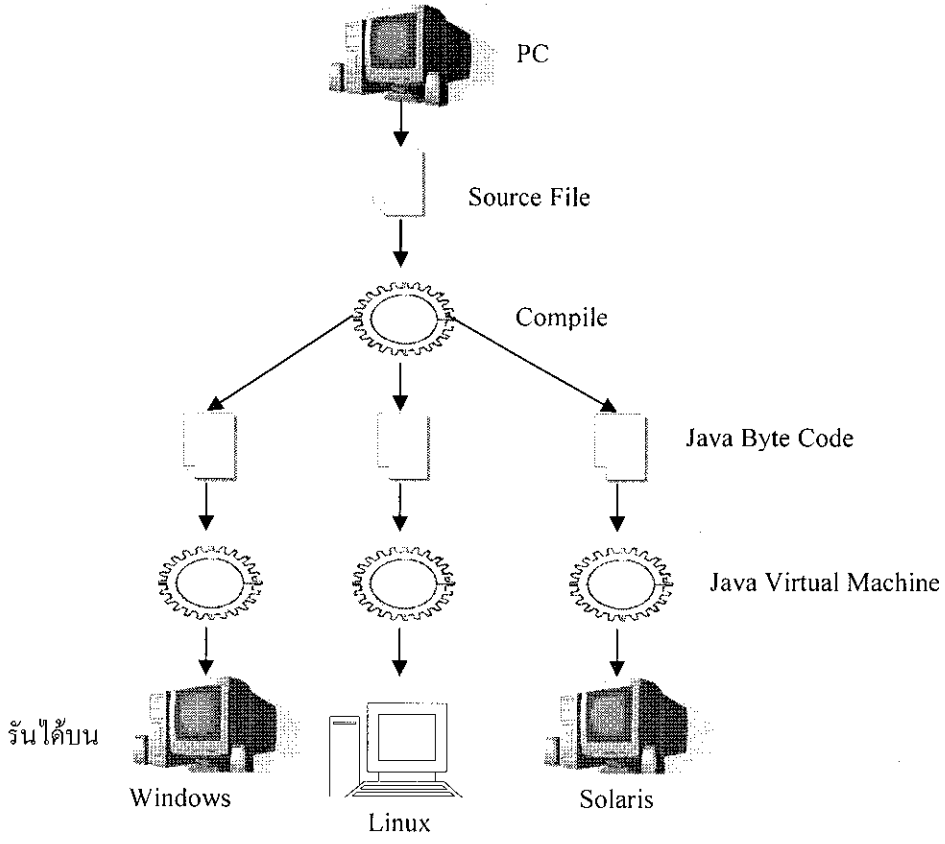
รูปที่ 2-17 วงจรเชื่อมต่อ MAX232 หรือ ICL232 เข้ากับพอร์ตอนุกรมของคอมพิวเตอร์และไมโครคอนโทรลเลอร์ MCS-51

2.3 การใช้โปรแกรมที่สามารถใช้งานผ่านระบบอินเตอร์เน็ต

ในบรรดาภาษาเขียนโปรแกรมนั้นภาษา Java นั้นนับเป็นภาษาที่น่าสนใจมากที่สุดภาษาหนึ่ง เพราะข้อดีที่สำคัญคือ การที่เขียนโปรแกรมครั้งเดียว แต่สามารถรันได้บนเครื่องคอมพิวเตอร์หลายรูปแบบโดยไม่ต้องมาคอมไฟล์ หรือเขียนโปรแกรมใหม่ ทำให้ช่วยย่นระยะเวลาการพัฒนาลงไปได้มาก แต่จริงๆ แล้ว Java ยังมีข้อดีอื่นๆ ซ่อนอยู่มาก

2.3.1 Java คืออะไร

Java เป็นภาษาสำหรับการเขียนโปรแกรมที่ถูกสร้างขึ้นมาเพื่อรองรับการออกแบบซอร์ฟแวร์ที่มีการเชื่อมโยงอินเตอร์เน็ต โดยภาษา Java นั้นสนับสนุนแนวคิดของการเขียนโปรแกรมเชิงวัตถุหรือที่รู้จักกันดีที่เรียกว่า OOP (Object-Oriented Programming) โดยมีความสามารถเฉพาะตัวต่างจากโปรแกรมภาษาชั้นสูงอื่นๆ เช่น C หรือ C++ ในเรื่องของการทำงานข้ามระบบปฏิบัติการ หรือ Platform ได้โดยไม่ต้องมีการคอมไพล์ใหม่



รูปที่ 2-18 แสดงคุณสมบัติของการทำงานข้ามระบบปฏิบัติการได้โดยไม่ต้องมีการคอมไพล์ใหม่

จากรูปที่ 2-18 จะเห็นว่าเราสามารถเขียนโปรแกรมด้วยภาษา Java ครั้งเดียว แต่นำไปใช้งานกับคอมพิวเตอร์ได้หลากหลายรูปแบบทั้ง Windows, ลีนุกส์, ยูนิกส์, Solaris โดยเราจะนำเอาโปรแกรมที่ได้ไปคอมไพล์เป็น Java Byte Code ของคอมพิวเตอร์แบบต่างๆ ซึ่งคอมพิวเตอร์แบบ

ต่างๆ นั้น ก็จะมีสิ่งที่เรียกว่า Java Virtual Machine คอยแปล Java Byte Code นำไปสั่งงานในคอมพิวเตอร์ในแบบของตนเอง [5]

รูปแบบของการเขียนโปรแกรมในภาษา Java

โปรแกรมที่ถูกพัฒนาด้วยภาษา Java ถูกแบ่งเป็น 2 รูปแบบหลักๆ คือ

1. Java Application เป็นโปรแกรม Java ทั่วไปที่ทำงานได้ด้วยตัวเองของมัน (Stand Alone Application เหมือนกับไฟล์ .EXE ใน Windows) ซึ่งสามารถใช้งานได้ทั้งบน Windows ลีนุกส์ และยูนิกส์ตระกูลต่างๆ

2. Java Applet เป็นโปรแกรม Java ขนาดเล็ก ซึ่งไม่สามารถรันได้ด้วยตัวเอง โดยถูกนำไปใช้ทำงานใน Web Browser หรือเว็บเพจทั่วไป

2.3.2 ลักษณะที่โดดเด่นของ Java

ลักษณะที่โดดเด่นอย่างหนึ่งของ Java คือ เป็นภาษาที่ได้รับความนิยมอย่างมาก โดยใช้ระยะเพียงไม่นานเมื่อเทียบกับภาษาอื่นๆ ทั้งที่ Java ไม่ได้เป็นภาษาที่สมบูรณ์แบบมากนัก แต่เนื่องจากโครงสร้างของ Java สามารถรองรับการทำงานกับการพัฒนาในอนาคตได้หลากหลาย บวกกับแนวคิดและวิธีที่ใช้สร้างโปรแกรมของ Java เองรวมถึงคุณสมบัติอื่นๆที่สนับสนุนได้แก่

- ผู้ที่เคยใช้โปรแกรมที่ได้รับความนิยมอื่นๆ มาก่อน เช่น C , C++ และ Smalltalk สามารถใช้โปรแกรม Javaได้อย่างคุ้นเคยเนื่องจาก Java ได้ดึงเอาคุณสมบัติที่ดีของภาษาเหล่านี้มาปรับปรุงให้ใช้งานได้ง่าย และมีประสิทธิภาพมากขึ้น
- มีการทำงานที่ติดต่อกับเว็บเพจอย่างใกล้ชิด จึงทำให้ได้รับความนิยมเป็นอย่างมาก
- สามารถรองรับการทำงานกับภาษาโปรแกรมอื่นๆ ในอนาคตได้ทุกภาษา
- Java ใช้หลักการของ Object Orientation ซึ่งเป็นรูปแบบภาษาการเขียนโปรแกรมแบบใหม่ ที่มีข้อดีมากมาย เช่น จากแนวคิดที่มองทุกอย่างเป็นวัตถุ ทำให้สามารถมองการเขียนโปรแกรมเป็นรูปธรรมมากขึ้น ซึ่งจะช่วยให้เขียนโปรแกรมได้ง่ายขึ้น เป็นต้น
- เขียนโปรแกรมน้อยลง เนื่องจาก Java สนับสนุนคุณสมบัติของการนำกลับมาใช้ใหม่ (Reuse) ทำให้ Application ที่สร้างขึ้นมีขนาดเล็กและสามารถไปใช้ร่วมกับ Application อื่นๆ ได้อย่างเหมาะสม
- ข้อดีอีกอย่างหนึ่งที่สำคัญคือ Java เป็นภาษาที่ใช้งานได้ฟรี และไม่เสียค่าใช้จ่ายใดๆ
- เขียนโปรแกรมเพียงครั้งเดียวสามารถเอาไปใช้ได้ทุกPlatform (ทุกระบบปฏิบัติการ)

ด้วยเหตุผลต่างๆดังที่กล่าวมานี้ รวมถึงคุณสมบัติอื่นๆ ที่ยังไม่ได้กล่าวถึงอีกมากมาย จึงไม่น่าแปลกใจเลยว่า ทำไม Java ถึงเป็นภาษาที่ได้รับความนิยม และเป็นภาษาสมัยใหม่อย่างแท้จริง [1]

2.3.3 รู้จักกับ Java 2 SDK

การใช้ Java และสร้างโปรแกรมจาก Java จำเป็นต้องใส่ชุด Kit ในการพัฒนา ดังนั้น Sun จึงได้เตรียมชุด Kit ไว้ให้กับผู้พัฒนาโปรแกรม คือ "Java 2 Platform System Development Kit (J2SDK)" มีลักษณะการทำงานในแบบ Command Line ซึ่งทำใน DOS นอกจากนี้ยังมีอีกหลายบริษัทที่ได้พัฒนาเครื่องมือสำหรับสร้าง Java Application ที่มีการทำงานในแบบกราฟิก เพื่อให้ใช้งานร่วมกับ J2SDK ได้ง่าย เช่น โปรแกรม JBuilder หรือโปรแกรม IBM VisualAge for Java เป็นต้น

สำหรับSDK จะมีเครื่องมือ และ Library มาตรฐานของ Java ที่ใช้สร้าง Applet และ Application ถ้าต้องการเขียนโปรแกรมแบบ Server-sides หรือ Servlets การใช้ J2SE Platform อย่างเดียวไม่พอ จะต้องดาวน์โหลด Java Platform อื่นเพิ่มเติม คือ Java 2 Enterprise Edition (J2EE) ถ้าต้องการพัฒนาโปรแกรมเกี่ยวกับ Midlets หรือ Wireless ซึ่งเป็นการทำงานเกี่ยวกับการติดต่อแบบไร้สาย จะต้องใช้ Java 2 Micro Edition (J2ME) ทั้ง 3 Platform ของ Java นี้สามารถดาวน์โหลดได้ที่เว็บไซต์ของ java.sun.com

ภายใน SDK จะมี Compiler ที่ใช้สำหรับแปลง Source Code ให้เป็นไฟล์ Byte Code ที่สามารถทำงานได้ทั้งแบบที่เป็น Java Application และ Java Applet ในส่วนของ SDK นอกจากจะมีเครื่องมือที่ใช้ในการ Compiler และ Interpret แล้ว ยังมี JRE และกลุ่มของ Libraries ต่างๆที่ใช้ในโปรแกรมด้วย ทั้ง 2 ส่วนนี้ จะเป็นตัวกำหนด Platform ของ Java ซึ่งส่วนมากการเปลี่ยนเวอร์ชันของ Java จะเปลี่ยนที่องค์ประกอบของ Libraries ต่างๆ เหล่านี้ ซึ่งการติดตั้ง Java 2 SDK สามารถดูได้จากภาคผนวก ง [6]

2.3.4 เครื่องมือของ Java SDK

ตารางที่ 3 แสดงเครื่องมือของ Java SDKซึ่งมีเครื่องมือทั้งหมด 23 ชุด แต่ที่จะกล่าวถึงในขั้นนี้มี 8 ชุด ได้แก่ [6]

เครื่องมือ	หน้าที่
javac	คำสั่งของคอมไพเลอร์ที่ใช้เปลี่ยนจาก Source Code เป็นByte Code
java	คำสั่งของตัวแปรภาษาที่สั่งให้ Application ทำงาน โดยการอ่านจาก Byte Code สำหรับใช้ใน Java Application
appletviewer	ใช้สำหรับสั่งให้ Java Applet ทำงานเพื่อทดสอบก่อนการนำไปใช้งานจริงผ่านบราวเซอร์

javadoc	เป็นเครื่องมือที่ใช้อธิบายโปรแกรม Java โดยสร้างเอกสาร HTML จากโปรแกรม Java
jdb	เป็นตัวดีบักเกอร์ของ Java ที่ใช้สำหรับตรวจสอบการทำงานของโปรแกรม โดยสั่งให้โปรแกรมหยุดการทำงานตามขั้นตอนที่โปรแกรมเมอร์กำหนดเพื่อตรวจสอบ ตัวแปรและดูลำดับในการทำงานของชุดคำสั่ง
javap	เป็นตัว Java Disassambler ซึ่งใช้ทดสอบ Byte Code ที่เกิดขึ้นแล้วแสดงข้อมูล (Method or Function) ของ Byte Code นั้นถูกเรียกใช้ในกรณีที่มีปัญหาในเรื่องของการเรียกประมวลผลโปรแกรม
jar	คือ ตัวจัดการเกี่ยวกับไฟล์ Jar Archive (JAR) โดยการรวมและบีบอัดไฟล์หลายๆไฟล์ รวมถึงระดับไดเร็กเทอรี่ให้เป็นไฟล์เดียว เพื่อความสะดวกในการใช้งาน
javah	คือ ตัวสร้างไฟล์ Header โดยจะถูกใช้เมื่อมีการทำงานร่วมกับโปรแกรม C หรือ C++ ทำให้ใช้ไฟล์ Java ได้
rmic	ใช้ร่วมกับ Remote Method Invocation (RMI) เพื่อสร้างคลาสในการประมวลผลแบบกระจาย

2.3.5 Compilation และ Interpretation

ในการเขียนโปรแกรมโดยทั่วไปจะต้องมีกระบวนการและเครื่องมือสำหรับสร้างโปรแกรมขึ้นมาซึ่งเครื่องมือพื้นฐานที่ใช้ในโปรแกรมทั่วไปได้แก่ Editor, Compilerและ Interpreter

โดยหลังจากที่สร้างโปรแกรมขึ้นมาแล้ว การนำโปรแกรมที่ได้ไปใช้งานจะต้องผ่านวิธีการคอมไพล์ (Compilation) หรือแปลภาษา (Interpretation) อย่างใดอย่างหนึ่ง เพื่อนำโปรแกรมนั้นไปแปรเป็นภาษาที่เครื่องเข้าใจ (Machine Language) ได้ ซึ่งแต่ละวิธีก็มีข้อดีและข้อเสียแตกต่างกันออกไป ดังนี้

- **วิธีการคอมไพล์ (Compilation)**

เป็นการสร้าง Executable Code หรือโค้ดที่นำไปใช้งานได้จาก Source Code โดยสิ่งที่เรียกว่า **"คอมไพเลอร์ (Compiler)"** วิธีนี้เกิดจากการที่คอมไพเลอร์ จะทำการแปรโปรแกรมทีเดียวทั้งโปรแกรม ซึ่งจะได้ Executable Code หรือไฟล์ .exe ขึ้นมา ดังนั้นคอมไพเลอร์จึงมีโอกาสวิเคราะห์ได้ทั้งโปรแกรม ทำให้การแปรภาษามีประสิทธิภาพ รวดเร็ว และมีขนาดไม่ใหญ่เกินไป ส่วนข้อเสีย คือ ขาดความยืดหยุ่น เนื่องจากการแปรภาษาด้วยวิธีการแปลเพียงครั้งเดียวนี้ จะได้ผลผลิต

ออกมาเป็นไฟล์ .exe ของโปรแกรมทั้งโปรแกรม ดังนั้นในระหว่างที่โปรแกรมทำงานจึงไม่สามารถเปลี่ยนแปลงส่วนใดของโค้ดได้เลย

- **วิธีการแปลภาษา (Interpretation)**

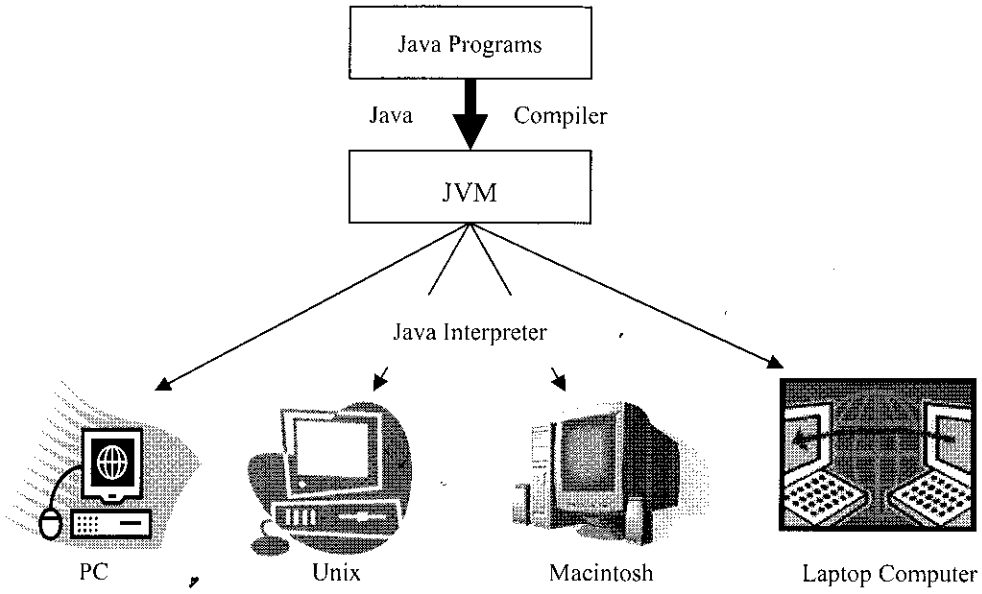
วิธีนี้ใช้เครื่องมือที่เรียกว่า "ตัวแปลภาษา (Interpreter)" แทนคอมไพเลอร์ เป็นวิธีที่แตกต่างจากวิธีการคอมไพล์คือ ใช้การอ่านและการแปลงโค้ดทีละบรรทัดแทน โดยเมื่อแปลงบรรทัดหนึ่งเสร็จแล้วจึงค่อยกลับมาอ่านบรรทัดต่อไปและแปลงโค้ด ทำอย่างนี้ไปเรื่อยๆจนจบโปรแกรม จากจุดนี้จะพบว่าวิธีนี้มีข้อดีคือ ทำได้ง่ายกว่าการคอมไพล์ ทีเดียวทั้งโปรแกรม และมีความยืดหยุ่นในการเขียนโปรแกรมมากกว่าวิธีการคอมไพล์ เนื่องจากในขณะแปลถ้าเกิด Error ที่จุดใดตัวแปรภาษาก็จะฟ้องและอนุญาตให้ผู้พัฒนาสามารถทำการแก้ไขได้ทันที จากนั้นจึงทำงานในบรรทัดต่อไป แต่มีข้อเสีย คือ เนื่องจากต้องทำทีละบรรทัด จึงส่งผลให้ทำงานได้ช้ากว่าวิธีการคอมไพล์

โดยทั่วไปภาษา โปรแกรมจะเลือกใช้วิธีการใดวิธีการหนึ่ง เช่น ภาษา C, Pascal ใช้วิธีการคอมไพล์ ส่วนภาษา Basic, Smalltalk ใช้วิธีการแปลภาษา ซึ่งรูปแบบการทำงานเช่นนี้จะทำให้เกิดผลเสียคือโปรแกรมขาดความสามารถไปการนำไปใช้กับระบบหรือเครื่องที่ต่าง Platform กัน โดยหากต้องการนำโปรแกรมหนึ่งไปใช้กับเครื่องอื่นๆก็ต้องแก้ไข Source Code ใหม่ ทำให้เสียเวลามาก

2.3.6 Java Virtual Machine (JVM)

จากปัญหาที่เกิดขึ้นกับความต้องการที่สามารถนำโค้ดไปใช้งานได้อย่างไม่มีข้อจำกัด ไม่ว่าจะเป็นเครื่องใด หรือระบบปฏิบัติการใดก็ตาม จึงได้มีการพัฒนาทำให้เกิดแนวความคิด Java Virtual Machine (JVM) ขึ้นมามีลักษณะ ดังรูป 2-19

JVM (Java Virtual Machine) เป็นกลไกลเสมือน ซึ่งสร้างขึ้นโดยตัวแปลภาษา (Interpreter) ของ Java โดยมีขั้นตอนการทำงานคือ เริ่มแรกนำ Source Code ที่อยู่ในรูปแบบของโปรแกรม Java มาผ่านการคอมไพล์โดยตัวคอมไพเลอร์ที่อยู่ในเครื่องที่ สร้างโปรแกรมของ Java ซึ่งจะได้ผลลัพธ์ (Interpreter) เพื่อสั่งให้โปรแกรมทำงาน โดยระหว่างแปลภาษานี้ตัว Interpreter ก็จะสร้าง JVM ขึ้นมา เพื่อนำโค้ดที่ได้จากการคอมไพล์มาเข้ากระบวนการที่สร้างขึ้น ซึ่งตัว Interpreter ไม่จำเป็นต้องอยู่ในเครื่องเดียวกับ Source Code ที่สร้างขึ้นก็ได้ โดยในการใช้งานถ้าต้องการเรียกใช้งานโปรแกรม Java ในเครื่อง ใดๆ ก็จะสามารถนำตัว Interpreter ไปติดตั้งไว้ที่เครื่องที่ต้องการได้ทันที โดยไม่ต้องขึ้นกับชนิดของเครื่องหรือระบบปฏิบัติการใดๆ ดังนั้นภาษา Java จึงทำงานได้เร็วและไม่ขึ้นกับระบบ เพราะว่าระบบทั่วๆไปจะลงโปรแกรม Java Interpreter ไว้ และการคอมไพล์ถูกแยกออกจากการ Execution นอกจากนี้การออกแบบคำสั่งของ JVM จะมีความใกล้เคียงกับของหน่วยประมวลผลทั่วไป ซึ่งทำให้ขั้นตอนการทำ Interpreter ทำได้ง่ายและรวดเร็วอีกด้วย



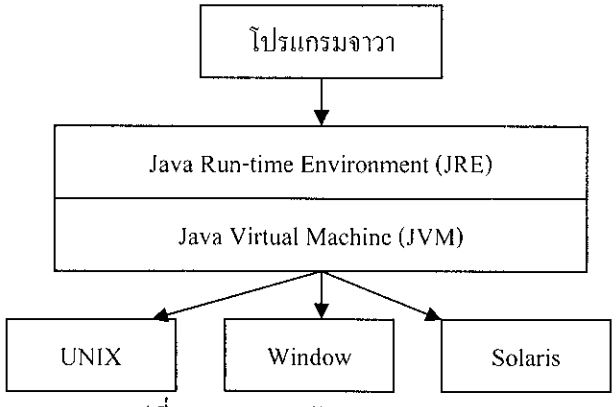
รูปที่ 2-19 แสดงแนวความคิดของ Java Virtual Machine (JVM)

2.3.7 Java Runtime Environment (JRE)

Java ไม่สามารถทำงานตามลำพังได้ แต่จะต้องอาศัยสภาวะแวดล้อมที่สนับสนุนการทำงานด้านต่างๆ เพิ่มเติม เช่น การทำงานของ Java แบบ Applet ที่ต้องอาศัยเว็บบราวเซอร์ในการแสดงผล เป็นต้น

Java ได้จัดเตรียมองค์ประกอบต่างๆ ที่ใช้สำหรับสนับสนุนการทำงานของโปรแกรมที่สร้างขึ้นอยู่ใน "Java Runtime Environment (JRE)"

JRE ประกอบด้วยไฟล์ที่จำเป็นต้องใช้ในการทำงานของ Java เช่น ไฟล์ที่บรรจุ Package ต่างๆ ไฟล์ใช้สำหรับแสดงรูปแบบตัวอักษร เป็นต้น โดยไฟล์ต่างๆเหล่านี้จะถูกติดตั้งพร้อมกับ J2SDK ตั้งแต่ v.1.1 ขึ้นไป ในการพัฒนาโปรแกรมขึ้นด้วยภาษา Java ไปทำงานด้วยนั้น ควรติดตั้ง JRE เวอร์ชันเดียวกันหรือสูงกว่า JRE ของผู้ที่สร้างโปรแกรมขึ้นมา



รูปที่ 2-20 แสดงลักษณะการทำงานของ JRE

2.3.8 Java Class Libraries

โปรแกรม Java ประกอบขึ้นด้วยส่วนต่างๆ ที่เรียกว่า " คลาส (Class) " ภายในคลาสประกอบด้วยส่วนการทำงานที่เรียกว่า "เมธอด (Method)" ซึ่งจะทำหน้าที่แสดงผลของการทำงานและส่งข้อมูลนั้นกลับมาเมื่อทำงานนั้นๆเสร็จ ในการสร้างโปรแกรมจะใช้วิธีเขียนคลาสแยกออกเป็นหนึ่งไฟล์ต่างหาก หรือเขียนทุกคลาสอยู่ในไฟล์เดียวกันก็ได้ แต่โปรแกรมเมอร์ส่วนใหญ่นิยมเขียนให้ทุกคลาสเก็บอยู่ในไฟล์เดียวกัน เพื่อความสะดวกในการคอมไพล์ และทดสอบโปรแกรม นอกจากนี้ Java ยังมีคลาสต่างๆให้โปรแกรม สามารถเรียกใช้งานได้ทันที ไม่จำเป็นต้องเขียนเอง โดยคลาสเหล่านี้จะถูกจัดเป็นกลุ่มๆ ภายใน Java Class Libraries ซึ่งกลุ่มของ Class Libraries เหล่านี้เรียกว่า "Java APIs (Application Programming Interface)" [6] ซึ่งหัวใจของภาษาโปรแกรมของ Java ได้แก่การบรรจุกลุ่มของ packages ที่เรียกว่า Java.lang ซึ่งเป็นส่วนหนึ่งของ API (Application Programming Interface) ของ Java ถึงแม้ว่าแพ็คเกจ (package) java.lang ให้การสนับสนุนภารกิจของภาษาโปรแกรมของ Java ไม่เพียงแต่เฉพาะแพ็คเกจ (package) นี้เท่านั้นที่รวมใน JDK JDK ได้รวมแพ็คเกจต่าง ๆ ต่อไปนี้ Java.applet, java.awt, java.awt.image, java.awt.peer, java.io, java.lang, java.net, และ java.util แพ็คเกจต่าง ๆ ดังกล่าวจะให้การสนับสนุนทุกสิ่งทุกอย่างที่ผู้เขียนโปรแกรมต้องการเริ่มต้น Application ของ Java ที่มีพลังอำนาจอย่างรวดเร็ว โดยสามารถดูโครงสร้างและคลาสต่างๆ ของแพ็คเกจเหล่านี้ได้จากภาคผนวก ก [2]

ตารางที่ 4 แสดงคลาสต่างๆ ของ Java APIs

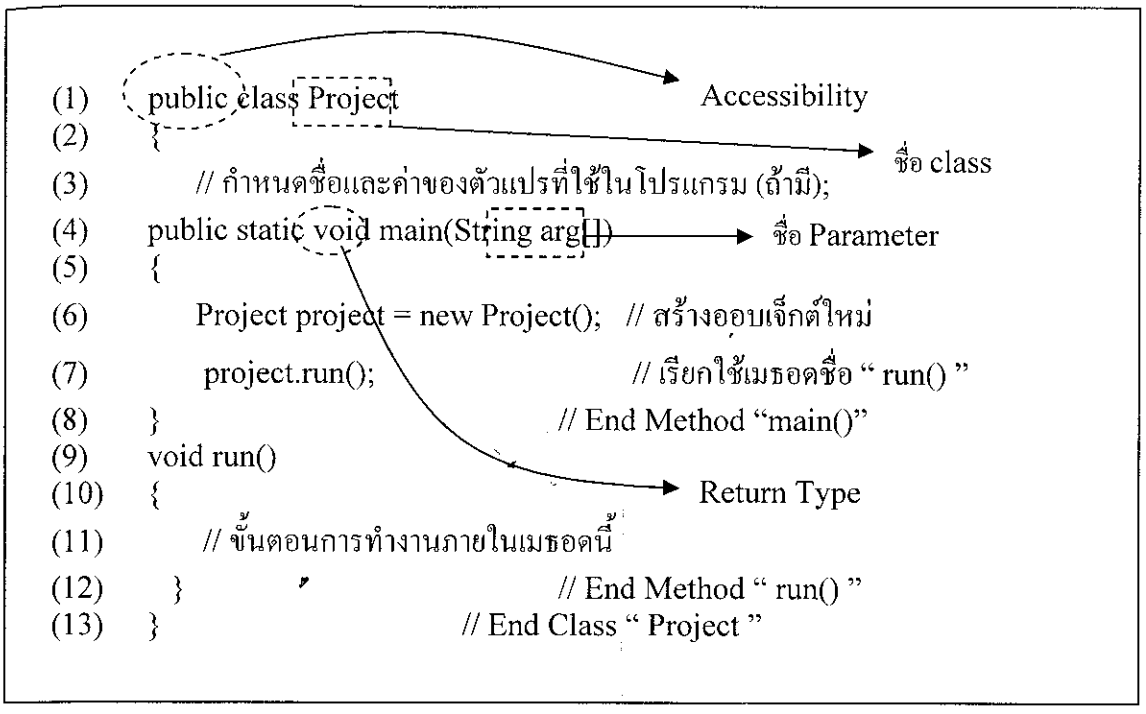
Package	Package	คำอธิบาย
Java.applet	Applet	กลุ่มของคลาสต่าง ๆ ที่เกี่ยวข้องกับสภาวะของ applet และให้การสนับสนุนการสร้าง applet ต่าง ๆ
Java.awt	Abstract Windowing Toolkit	กลุ่มของคลาสต่าง ๆ ที่ให้การสนับสนุนเครื่องมือสำหรับเชื่อมโยงกับกราฟฟิก เช่น ปุ่ม (buttons), scrollbars, controls, และวินโดวส์
Java.awt.image	AWT Image	กลุ่มของคลาสต่าง ๆ ที่เกี่ยวข้องกับการใช้ภาพ (images)
Java.awt.peer	AWT Peer	กลุ่มของคลาสต่าง ๆ สำหรับคลาสต่าง ๆ ของ AWT peer และ methods

Java.sql	Database connectivity	กลุ่มของคลาสต่าง ๆ ที่ช่วยตัวพัฒนาเขียนแอพพลิเคชันของ java เพื่อเข้าถึงฐานข้อมูล
Java.io	I/O	กลุ่มของคลาสต่าง ๆ ที่ให้การสนับสนุนอินพุตและเอาต์พุตมาตรฐานและไฟล์อรรถประโยชน์ I/O
Java.lang	Language	กลุ่มของคลาสต่าง ๆ ที่สำคัญสำหรับภาษา Java ที่ให้การสนับสนุนฟังก์ชันพื้นฐาน เช่น การเชื่อมโยงกับสติงและอะเรย์
Java.net	Network	กลุ่มของคลาสต่าง ๆ ที่ให้การสนับสนุนเครื่องมือสำหรับเข้าถึงเครือข่ายโดย protocols เช่น FTP, Telnet, และ HTTP
Java.util	Utility	กลุ่มของคลาสต่าง ๆ ที่ให้การสนับสนุนฟังก์ชันอรรถประโยชน์ เช่น encoding/ ecoding, hash tables, และ stacks

2.3.9 โปรแกรม Java Application

ในการเขียนโปรแกรมด้วยภาษา Java เพื่อสร้าง Application นั้นจะแบ่งโปรแกรมการทำงานออกเป็นคลาส (Class) โดยภายในหนึ่งโปรแกรมจะต้องมีคลาสอย่างน้อยหนึ่งคลาสเสมอ ผู้เขียนโปรแกรมสามารถกำหนดรูปแบบการเขียนโปรแกรมได้อย่างอิสระ ซึ่งแต่ละคนก็อาจจะเขียนแตกต่างกันไป เช่น บางคนเขียนการทำงานทั้งหมดไว้ในคลาสเดียวโดยไม่มีการสร้างออบเจ็กต์เลย แต่บางคนจะแบ่งการทำงานออกเป็นหลายๆ คลาส แล้วสร้างออบเจ็กต์เพื่อเรียกใช้เมธอดในการเข้าถึงคุณสมบัติบางตัว และในการเขียนคลาสแต่ละคลาสนั้น บางคนอาจเขียนให้หนึ่งคลาสอยู่ในหนึ่งไฟล์ แต่บางคนอาจเขียนหลายๆ คลาสอยู่ในไฟล์เดียวกันก็ได้ เป็นต้น

โครงสร้างการเขียนโปรแกรมแบบมีการสร้างออบเจ็กต์ (Object) ในการเขียนโปรแกรมสำหรับภาษา Java นั้นจะเขียนโปรแกรมเป็นคลาส ซึ่งแต่ละคลาสจะประกอบไปด้วยเมธอด (Method) ตัวแปรและคำสั่งต่างๆ ที่จำเป็นต่อการใช้งาน ภายในโปรแกรมหนึ่งอาจจะประกอบไปด้วยคลาสมากกว่าหนึ่งคลาสก็ได้ ขึ้นอยู่กับผู้ออกแบบโปรแกรม และเมื่อต้องการใช้งานคลาสที่สร้างขึ้นมาจะต้องสร้างออบเจ็กต์จากคลาสดังกล่าวขึ้นมาก่อน แล้วจึงนำออบเจ็กต์ที่สร้างขึ้นไปใช้งานตามต้องการ สำหรับการเขียนโปรแกรมแบบนี้มีโครงสร้างดังนี้ [1]



จากโครงสร้างการเขียนโปรแกรม Java Application ข้างต้น สามารถอธิบายการทำงานดังนี้

บรรทัดที่ (1) เป็น Header ของคลาส คือ เป็นการประกาศว่าคลาสชื่ออะไร และมีชนิด Accessibility เป็นอะไร ในที่นี้คลาส "Project" และมีชนิดของ Accessibility เป็น "public"(ปกติไม่จำเป็นต้องใส่ก็ได้)

บรรทัดที่ (2) เป็นจุดเริ่มต้นการทำงานของคลาส "Project" ซึ่งภายในคลาสและเมธอดจะต้องมีจุดเริ่มต้นและสิ้นสุดการทำงานของคลาสและเมธอดเสมอ โดยเครื่องหมายที่ใช้เป็นจุดเริ่มต้นคือวงเล็บปีกกาเปิด "{"และเครื่องหมายที่ใช้เป็นจุดสิ้นสุดคือ วงเล็บปีกกาปิด "}"

บรรทัดที่ (3) เป็นส่วนที่ใช้ประกาศตัวแปรไว้ใช้ภายในคลาส

บรรทัดที่ (4) เป็นการประกาศ Header ของเมธอด คือ เป็นการประกาศว่าเมธอดชื่ออะไร มีค่า Accessibility เป็นอะไร เมื่อทำงานภายในเมธอดเสร็จจะส่ง (Return Type) ข้อมูลชนิดอะไรกลับมา และมี Parameter กี่ตัว เป็นชนิดอะไรบ้าง ในที่นี้เป็นการประกาศเมธอดชื่อ "main()" มีค่า Accessibility เป็น "public" ไม่มีการส่งค่ากลับมา ("void") และมี Parameter เป็น Array ชื่อ "args" ซึ่งมีชนิดข้อมูลเป็น String โดยเมธอด "main()" เป็นเมธอดหลักที่ใช้เริ่มการทำงานของโปรแกรม รวมถึงการเรียกใช้เมธอดอื่นๆ ด้วย ในภาษา Java มีข้อกำหนดว่าในหนึ่งโปรแกรมจะมีเมธอด "main()" ได้เพียงหนึ่งเมธอดเท่านั้น (กล่าวคือ ไม่สามารถเรียกใช้เมธอด "main()" ของคลาสอื่นได้)

บรรทัดที่ (5) เป็นจุดเริ่มการทำงานของเมธอด "main()"

บรรทัดที่ (6) เป็นการสร้างออบเจ็กต์ชื่อ "project" จากคลาส "Project"

บรรทัดที่ (7) เป็นการเรียกใช้เมธอด “run()” ของออบเจ็กต์

บรรทัดที่ (8) เป็นการสิ้นสุดการทำงานของเมธอด “main()”

บรรทัดที่ (9) เป็นการประกาศเมธอดชื่อ “run()” สังเกตว่าท้ายชื่อเมธอดต้องตามด้วยเครื่องหมายวงเล็บเปิดและปิดด้วย ที่เป็นเช่นนี้เนื่องจากในการประกาศเมธอดนั้นสามารถที่จะมี Parameter ได้ เพื่อนำค่าที่รับมาใช้ในการทำงานภายในเมธอด

บรรทัดที่ (10) เป็นจุดเริ่มการทำงานของเมธอด “run()”

บรรทัดที่ (11) เป็นจุดที่ระบุการทำงานต่างๆ ภายในเมธอด “run()” ซึ่งอาจมีได้มากกว่าหนึ่งคำสั่ง

บรรทัดที่ (12) เป็นจุดสิ้นสุดการทำงานของเมธอด “run()”

บรรทัดที่ (13) เป็นจุดสิ้นสุดการทำงานของคลาส “Project”

2.3.10 โปรแกรม Java Applet

Java Applet คือ โปรแกรมขนาดเล็ก ๆ ในภาษา Java ที่ถูกกำหนดไว้ให้ใช้งานได้บนเว็บเพจ โดยแสดงผลผ่าน Web browser

Applet เป็นโปรแกรมที่ต่างจากโปรแกรมทั่วไปคือ มันไม่สามารถรันได้ด้วยตัวเอง แต่จะต้องถูกเรียกใช้โดย Browser ซึ่งเป็นเหตุผลที่ว่า Applet ต้องเป็นโปรแกรมที่มีขนาดเล็ก เพราะถ้าสร้างให้ Applet มีขนาดใหญ่ก็จะทำให้ต้องใช้ทรัพยากรจำนวนมาก ซึ่งจะทำให้การทำงานช้าลง

Applet ตัวหนึ่งๆ เราสามารถเรียกได้ว่าเป็น GUI หรือ Graphic User Interface ได้ เพราะเมื่อถูกแสดงผลบนหน้าจอวินโดว์ จะมีลักษณะเป็นพื้นที่รูปสี่เหลี่ยม ที่สามารถมีองค์ประกอบอื่นๆ ภายในเช่น ปุ่ม หรือกล่องข้อความ หรือแม้แต่กราฟฟิกที่ให้ความสวยงามจำพวกรูปภาพ รูปทรงต่างๆ เส้นและอื่นๆ ได้อีก โดยนอกจากนั้นแล้วยังสามารถตอบสนองเหตุการณ์ต่างๆ ที่เกิดขึ้นกับหน้าจอ Applet นั้น ได้ด้วย เช่น การคลิกเมาส์ หรือการพิมพ์ที่แป้นของผู้ใช้

กระบวนการหลักๆ ของการสร้าง Applet

Java Applet นั้นเป็นออบเจ็กต์ที่สร้างมาจากคลาส java.applet.Applet หรือเรียกได้ว่าเป็นคลาสย่อยของมันอีกที โดยตัวคลาส Applet แท้ที่จริงมีหน้าที่เพียงแค่ใช้ในการสร้างคลาสย่อย หรือออบเจ็กต์ขึ้นมาไม่ได้ทำอะไรมากกว่านั้น ดังนั้นมันจะเป็นเพียงแค่พื้นที่ว่างๆ บนหน้าจอที่ไม่มีการตอบสนองใดๆ

การจะสร้าง Applet ที่มีประโยชน์ขึ้นมานั้น โปรแกรมเมอร์จะต้องเพิ่มเติมคลาสย่อยซึ่งขยาย (extends) มาจากคลาส Applet มีรูปแบบคือ [5]

```
Public class YourApplet extends java.applet.Applet {
        //ส่วนของโปรแกรม การทำงานของ Applet
}
```

หลังจากที่เพิ่มคลาสย่อยเข้าไปในคลาสที่สร้างจาก Applet แล้ว เราจะได้เมธอดต่างๆ ที่ถูกกำหนดไว้ในคลาส Applet ตามกฎการถ่ายทอดโดยจะมีบางเมธอดที่เราสามารถใส่คำสั่งต่าง ๆ ลงไป ขณะที่บางเมธอดเราก็ไม่ได้ทำอะไรกับมันเลย

จากการเรียนรู้เกี่ยวกับการสร้าง Java Application เรารู้ว่าฟังก์ชั่น main( ) จะถูกเรียกใช้โดยระบบ เพื่อจะรู้ว่าโปรแกรมหรือ Application นั้นๆ มีกระบวนการหรือลำดับการทำงานอย่างไร เมื่อโปรแกรมถูกเรียกใช้หรือประมวลผล แต่เนื่องจาก Applet นั้นไม่ได้เป็นโปรแกรมชนิด stand-alone หรือประมวลผลได้อิสระ เพราะฉะนั้นจึงมีความแตกต่างบ้าง แต่ก็คงยังใช้หลักการเดียวกัน

Applet นั้นมีเมธอดอยู่หลายเมธอดที่ทำหน้าที่เหมือนกับ main( ) ซึ่งเมธอดที่ว่าได้แก่ paint( ), start( ), stop( ), init( ) และ destroy( ) โดยมีรูปแบบทั่วไปดังนี้

```
public class YourApplet extends java.applet.Applet {
        //ส่วนของโปรแกรมการทำงานของ Applet
        public void paint( ) {
        }
        public void start( ) {
        }
        public void stop( ) {
        }
        public void init( ) {
        }
        public void destroy( ) {
        }
}
```

เมธอด paint( )

เป็นเมธอดที่ถูกกำหนดในคลาส Applet แต่ไม่ได้ทำอะไรเลย มีไว้สำหรับเมื่อต้องการให้ Applet แสดงอะไรขึ้นมาบนหน้าจอไม่ว่าจะเป็นข้อความ รูปภาพ รูปทรงต่างๆ หรืออะไรก็แล้วแต่ โดยสามารถเกิดขึ้นกี่ครั้งก็ได้ตลอดช่วงชีวิตของ Applet (applet life cycle)

เมธอด paint( ) นี้จะต้องถูกเขียนใหม่ลงไปที่ส่วนของคลาสย่อย applet ที่สร้างขึ้นเพื่อให้มีอะไรแสดงไปที่หน้าจอ โดยมีรูปแบบดังนี้

```
public void pain(Graphics g) {
        //โปรแกรม สิ่งที่วาดลงไปบน Applet
}
```

พารามิเตอร์ "g" ที่มีชนิดเป็นคลาส Graphics จะถูกกำหนดโดยระบบที่ใช้เรียกใช้ Applet ซึ่งสำหรับการวาดรูป หรือทำกราฟฟิกต่างๆ บนจาวาส่วนใหญ่ จะถูกทำโดยคลาส Graphics กล่าวได้อีกนัยหนึ่งก็คือ เมธอด paint( ) ที่กำหนดใน applet นี้ ไม่ได้ทำการวาดรูปใดๆ บน Applet เลย ไม่ว่าจะเป็นกล่องข้อความหรือรูปทรงต่างๆ แต่ออบเจ็กต์เหล่านั้นทำการวาดรูปมันตัวเองด้วยการใช้เมธอด paint( ) ของมันเองต่างหาก (ตัวอย่างบางอันของ applet ที่ไม่มีเมธอด paint( ))

เมธอด init( )

ปกติแล้วในการสร้างออบเจ็กต์ทั่วไป Constructor จะรับผิดชอบในการกำหนดค่าเริ่มต้นหรือทำการ Initialize ให้กับออบเจ็กต์นั้นๆ เสมอ แต่กับ Applet แล้ว สิ่งที่จะทำการ Initialize Applet กลับไม่ใช่ Constructor แต่เป็นเมธอดที่ชื่อว่า init( ) แทน โดยที่โปรแกรมเมอร์จะต้องเป็นคนเขียนทับลงไปเหมือนกับเมธอด paint( )

เมธอด init( ) เป็นเมธอดที่ถูกอ่านโดยระบบเมื่อ Applet ถูกประมวลผลเช่นเดียวกันกับ paint( ) มีไว้เพื่อบอกกับระบบว่าอะไรจะเกิดขึ้นบ้างในเมื่อ Applet ใดๆ ถูกสร้างขึ้น เช่นต้องการกำหนดสีที่พื้นหลัง ชนิดของฟอนต์ที่ถูกใช้ข้อความขณะที่ Applet กำลัง paint หรือการสร้างปุ่มขึ้นมาบน Applet โดยมีรูปแบบการใช้ดังนี้

```
public void init( ) {
        // ส่วนของโปรแกรมที่ถูกทำเมื่อ applet ถูก load ในตอนแรก
}
```

จริงๆ แล้วบางคนอาจจะเขียนรวมทุกอย่างไว้ในเมธอด paint( ) หมดเลยก็ได้ ถ้าอย่างนั้นแล้ว Java จะกำหนดเมธอด init( ) นี้ขึ้นมาให้ใช้ทำไมกัน คำตอบก็คือภาษา Java กำหนดมันขึ้นมาเพื่อกระทำการกำหนดสิ่งแวดล้อมเริ่มต้นให้กับ applet นั้นๆ ในตอนเริ่มแรกที่มันถูกเรียกขึ้นมาโดยจะถูกทำเพียงครั้งเดียวเท่านั้นในช่วงชีวิตของ applet ต่างกับเมธอด paint( ) ที่สามารถถูกทำใหม่โดยเรียกใช้เมธอด repaint( ) กี่ครั้งก็ได้

เมธอด start( )

ปกติแล้ว Applet จะเริ่มทำงาน (start) หลังจากการที่กำหนดค่าเริ่มต้น (init) แต่การเริ่มของ Applet นั้นสามารถเกิดขึ้นได้อีกในกรณีที่ก่อนหน้านี้มันถูกบังคับให้หยุด หรือผู้ใช้กดลิงค์ทำให้หน้าจอปัจจุบันเปลี่ยนจากการรัน Applet ไปเป็นอย่างอื่น นั่นหมายความว่า เมื่อผู้ใช้กลับมายังหน้า Applet อีกครั้งมันก็จะเริ่มทำงานใหม่อีกครั้งหรือถูก start ขึ้นใหม่นั่นเอง (แสดงให้เห็นว่าการเริ่ม applet สามารถเกิดขึ้นได้มากกว่าหนึ่งครั้งในช่วงชีวิตของ applet)

การเขียนทับเมธอด start( ) ทำได้ดังนี้

```
Public class YourApplet extends Applet {
        Public void start( ) {
        }
}
```

กระบวนการที่จะใส่ไว้ในเมธอด start( ) นี้อาจจะเป็นการส่งข้อความบางอย่าง เพื่อบอกว่า applet นี้ยังคงทำงานอยู่ ถ้าไม่จำเป็นต้องทำก็อาจจะไม่ต้องเขียนลงไปก็ได้

เมธอด stop( )

การหยุดของ applet หมายถึง การที่ผู้ใช้ออกจากหน้าจอปัจจุบันที่กำลังรัน applet อยู่นั้น หรือการที่ applet หยุดตัวมันเองด้วยการเรียกเมธอด stop( ) โดยตรง มีรูปแบบการใช้ดังนี้

```
public class YourApplet extends Applet {
        Public void stop( ) {
        }
}
```

เมธอด destroy( )

เป็นการบอกให้ applet ทำการล้างหน่วยความจำต่างๆ ให้เป็นเหมือนก่อนที่มันจะใช้เมื่อทำการปิดบราวเซอร์หรือใช้ทำลายเธรดต่างๆ ที่ยังคงทำงานค้างอยู่ ซึ่งโดยทั่วไปในการเขียนโปรแกรมในส่วนนี้อาจจะไม่มีความจำเป็นมาก นอกเสียจากจะต้องการปลดปล่อยทรัพยากร (resource) ที่มีจำกัด

```
public class YourApplet extends Applet {
        public void stop ( ) {
        }
}
```

วงจรชีวิตของ Java Applet

เนื่องจากการแสดงผลลัพธ์ของ Applet นั้นได้ทำผ่าน HTML ซึ่งอ่านโดย Browser ดังนั้นการทำงานทั้งหมดจะถูกจัดการโดย Browser [5]

1. เริ่มแรก Browser จะอ่านไฟล์ HTML จนกว่าจะพบ TAG <APPLET> ใดๆ ในไฟล์นั้นๆ

2. จากนั้นก็จะค้นหา CODE, CODEBASE ที่ถูกกำหนดค่าไฟล์ Applet และที่อยู่ของ Applet ใน TAG <APPLET>

3. Browser ทำการดาวน์โหลดไฟล์ (.class) ตามชื่อ ตามที่อยู่ URL ที่เจอในข้อ 2

4. หลังจากการดาวน์โหลดก็จะแปลงข้อมูลไฟล์ที่ดาวน์โหลดมาเป็นคลาสจาวา

5. สร้าง Applet Object

6. เรียกเมธอด init( ) ของ Applet ขึ้นมาทำงาน

7. จากนั้นเรียกเมธอด start( ) และเมธอด paint( ) เพื่อแสดงรายละเอียดหรือผลลัพธ์ของ Applet

8. ขณะที่ Applet ทำงานอยู่ อาจเกิดการคลิกเมาส์กดแป้นพิมพ์หรืออื่นๆ จากผู้ใช้ เพื่อให้เกิดการตอบสนอง Browser จะเรียกใช้เมธอด handleEvent( ) ในการตรวจสอบการกระทำต่างๆ ที่เข้ามาและทำการวาด Applet ใหม่ (repaint)

9. เมื่อ Applet ไม่ใช่จุดสนใจอีกต่อไป (ปิดหน้าจอหรือเปลี่ยนหน้าจอ) Browser จะเรียกเมธอด stop( )

10. สุดท้ายเรียกเมธอด destroy ( ) เป็นการล้างข้อมูลและหน่วยความจำครั้งสุดท้าย

Java Applet กับ HTML

เนื่องจากการใช้งาน Applet นั้นต้องทำงานบนเว็บเพจที่เขียนด้วย HTML ดังนั้นเราน่าจะทราบถึงความเกี่ยวพันของสองสิ่งนี้ [5]

Applet Tag

แท็ก <APPLET> ใช้ในการเพิ่ม Applet ให้กับเว็บเพจหนึ่งๆ ซึ่งจะอยู่คู่กันกับแท็ก </APPLET> เสมอ คำสำคัญ "CODE" ใช้สำหรับการกำหนดชื่อของคลาส Applet ที่ถูกคอมไพล์สมบูรณ์แล้ว (ไฟล์นามสกุล class นั่นเอง) ส่วน "HEIGHT" และ "WIDTH" มีไว้เพื่อกำหนดขนาดของ Applet ที่จะวางบนเว็บเพจนั้นๆ

โดยปกติแล้ว Applet สามารถมีพารามิเตอร์ที่ส่งมาจากเพจ HTML ได้เช่นกันโดยใช้แท็ก <PARAM> ที่จะต้องถูกใช้ภายในแท็ก <APPLET> และ </APPLET> เท่านั้นโดยมีรูปแบบการใช้ดังนี้

```
<APPLET CODE="yourApplet.class" WIDTH=100 HEIGHT=100>
<PARAM NAME="param-name1" VALUE="param-value1">
<PARAM NAME="param-name2" VALUE="param-value2">
.
.
<PARAM NAME="param-nameN" VALUE="param-valueN">
</APPLET>
```

จากรูปแบบการใช้งานที่ผ่านมา คำสำคัญ "NAME" ใช้กำหนดชื่อของพารามิเตอร์หรือกล่าวได้ว่าเป็นการกำหนดชื่อตัวแปรเพิ่มเข้าไปใน Applet และใช้ "VALUE" ในการกำหนดค่าของมัน โดยในแต่ละ Applet นั้นสามารถมีพารามิเตอร์มากเท่าไหร่ก็ได้หรืออาจจะไม่มีเลยก็ไม่เป็นปัญหา

เราสามารถใช้พารานี้ระบุความเป็นไปของ Applet ในตอนที่มันถูกเรียก ซึ่งพารามิเตอร์เหล่านี้สามารถถูกเรียกใช้โดย applet ผ่านเมธอดที่ชื่อว่า getParameter( )

```
String getParameter ( Steing paramName)
```

พารามิเตอร์ “paramName” ในเมธอด getParameter นี้ก็คือชื่อของพารามิเตอร์ที่ระบุไว้ใน <PARAM> หรือก็คือ “param-name” นั่นเอง ซึ่งจะคืนค่าออกมาเป็นค่าที่ระบุไว้ของ “param-name” ซึ่งก็คือ “param-value” นั่นเอง โดยถ้า “paramName” ที่ระบุในเมธอด getParameter ไม่ตรงกับพารามิเตอร์ที่มีอยู่ใน <PARAM> นั้นก็จะทำให้เมธอด getParameter ส่งค่าคืนกลับมาเป็น null

นอกจากนี้การใส่แท็กอื่นๆ นอกเหนือจาก <PARAM> ภายในแท็ก <APPLET></APPLET> นั้นจะไม่มีผลกับบราวเซอร์ที่สนับสนุนจาวา (บราวเซอร์จะไม่อ่านประโยคนั้นๆ) หรือหมายความว่า ถ้าใส่ข้อความว่า “ขออภัยบราวเซอร์นี้ไม่สนับสนุนการใช้จาวา” ภายใต้แท็ก <APPLET> จะถูกแสดงออกทางบราวเซอร์ก็ต่อเมื่อบราวเซอร์นั้นๆ ไม่สนับสนุนการทำงานจาวาจริงๆ

บทที่ 3

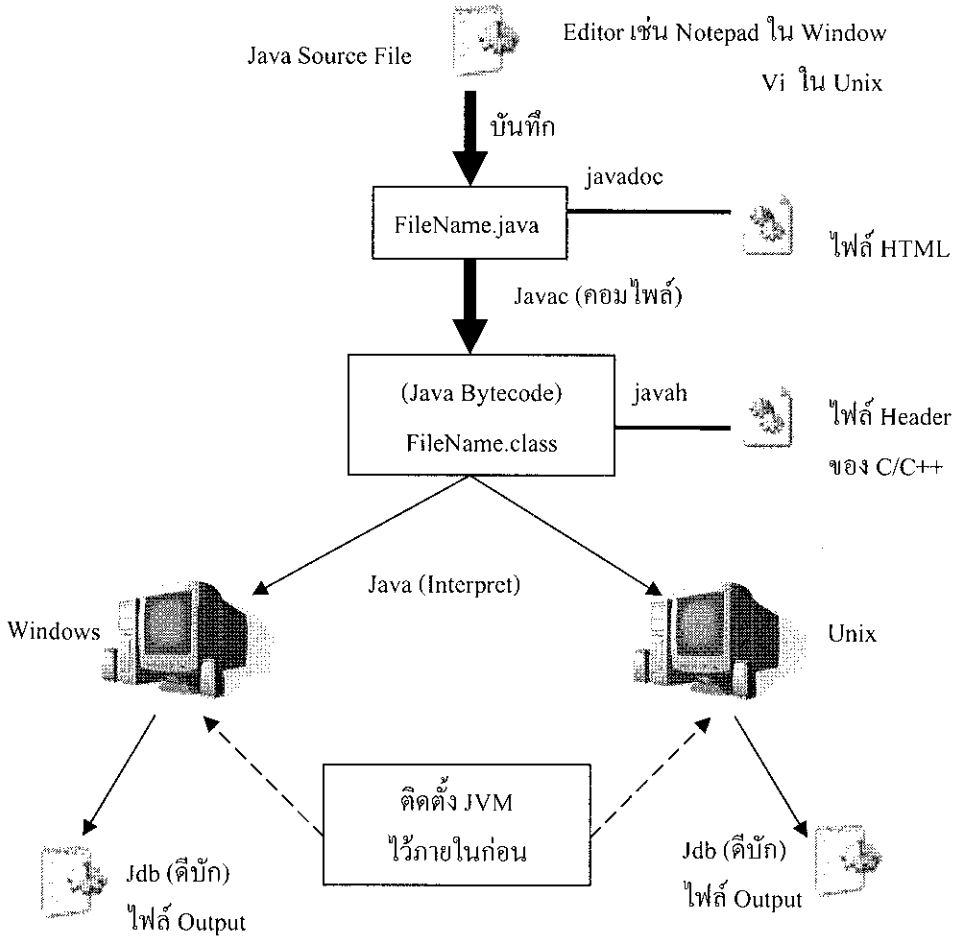
การสร้างและการออกแบบ

3.1 การเขียนโปรแกรมเพื่อรับ-ส่งภาพและ control ระหว่างเครื่อง client กับเครื่อง server

โปรแกรม Java ที่ใช้ในโครงงานนี้มี 2 แบบ คือ Java Application และ Java Applet

- แบบที่ 1 Java Applet เป็นโปรแกรม Java ที่ใช้ทำงานร่วมกับ Browser และภาษา HTML ซึ่งเป็นสภาวะแวดล้อม ในการทำงานบนเว็บเพจ
- แบบที่ 2 Java Application สามารถเรียกใช้งานได้ทันทีโดยไม่ต้องใช้เว็บ Browser

3.1.1 โปรแกรม Java แบบ Application



รูปที่ 3-1 แสดงขั้นตอนการสร้าง Java

การเขียนโปรแกรม Java แบบ Application มีขั้นตอนดังนี้ [1]

1. ในขั้นแรกจะต้องมีเครื่องมือสำหรับการเขียน และ Run โปรแกรมดังนี้

- **Editor Tools** หรือเครื่องมือที่ใช้สำหรับเขียนโปรแกรม เช่น Notepad, Editplus 2 เป็นต้น ปัจจุบันเครื่องมือที่ใช้สำหรับเขียนโปรแกรม Java มีอยู่มากมาย ซึ่งมีอุปกรณ์ช่วยเหลือต่างๆ รวมทั้ง J2SDK ที่มีมาพร้อมกับตัวโปรแกรม
- **J2SDK (Java 2 Software Development Kit)** เป็นเครื่องมือที่มีความจำเป็นสำหรับการเขียนโปรแกรม Java เป็นอย่างมาก เนื่องจากภายในได้บรรจุเครื่องมือที่จำเป็นสำหรับการเขียนโปรแกรมไว้ เช่น JVM, JRE, คอมไพเลอร์(Compiler) และ ตัวแปลภาษา (Interpreter) เป็นต้น สามารถดูขั้นตอนการติดตั้ง J2SDK ได้จากภาคผนวก ง [2]

2. ติดตั้ง J2SDK และ Set ค่าต่างๆให้เรียบร้อย

3. เริ่มต้นการเขียนโปรแกรมโดยใช้ Editor และ บันทึกไฟล์โดยใช้นามสกุลเป็น .java ซึ่งเป็นไฟล์ที่ได้นี้ถ้าใช้คำสั่ง javadoc.exe จะทำให้ได้ไฟล์ HTML ที่อธิบายส่วนประกอบต่างๆ ของไฟล์ java นั้น

4. คอมไพล์โปรแกรมโดยใช้คำสั่ง javac.exe จะได้ไฟล์ที่มีนามสกุล .class

5. หลังจากได้ไฟล์ .class แล้ว อาจจะใช้ javah.exe เพื่อสร้างไฟล์ Header ของ C/C++ ก็ได้

6. เมื่อต้องการให้โปรแกรมทำงานจึงเรียกใช้ java.exe เพื่อ Run โปรแกรม ซึ่งเครื่องมือที่ใช้ Run โปรแกรมนี้ อาจจะเป็น Platform ที่ต่างจากเครื่องที่ใช้สร้าง Code ก็ได้ ถ้าติดตั้ง JVM ไว้

7. ในขณะที่โปรแกรมกำลังทำงานอาจจะเรียกใช้ jdb.exe เพื่อทำการแก้ไขโปรแกรมได้

จากขั้นตอนการเขียนโปรแกรม Java แบบ Application ข้างต้น สามารถที่จะสรุปเป็นขั้นตอนการสร้างโปรแกรม Java Application ได้ ซึ่งดูได้จากภาคผนวก ข [5]

<u>โปรแกรม Java Application ที่ใช้ในโครงงานนี้มีดังนี้</u>

1. โปรแกรม MainClass.java เป็นโปรแกรมทางฝั่งเครื่อง server ที่ทำหน้าที่ติดต่อกับเครื่อง client ซึ่งโปรแกรมนี้จะทำการเรียกไปยังโปรแกรมย่อยต่างๆ เพื่อรับ-ส่งภาพและ control ระหว่างเครื่อง client กับเครื่อง server โดยโปรแกรมย่อยที่โปรแกรมนี้เรียกใช้มีดังนี้

1.1 โปรแกรม ClientReceiveImage.java มีหน้าที่รอรับการติดต่อจากเครื่อง client ที่มีกล้องวงจรปิด โดยจะใช้ port 7001 ในการติดต่อ เมื่อมีการติดต่อจากเครื่อง client แล้วก็จะรับภาพจากเครื่อง client ส่งไปยังโปรแกรมย่อย ClientCamSender.java

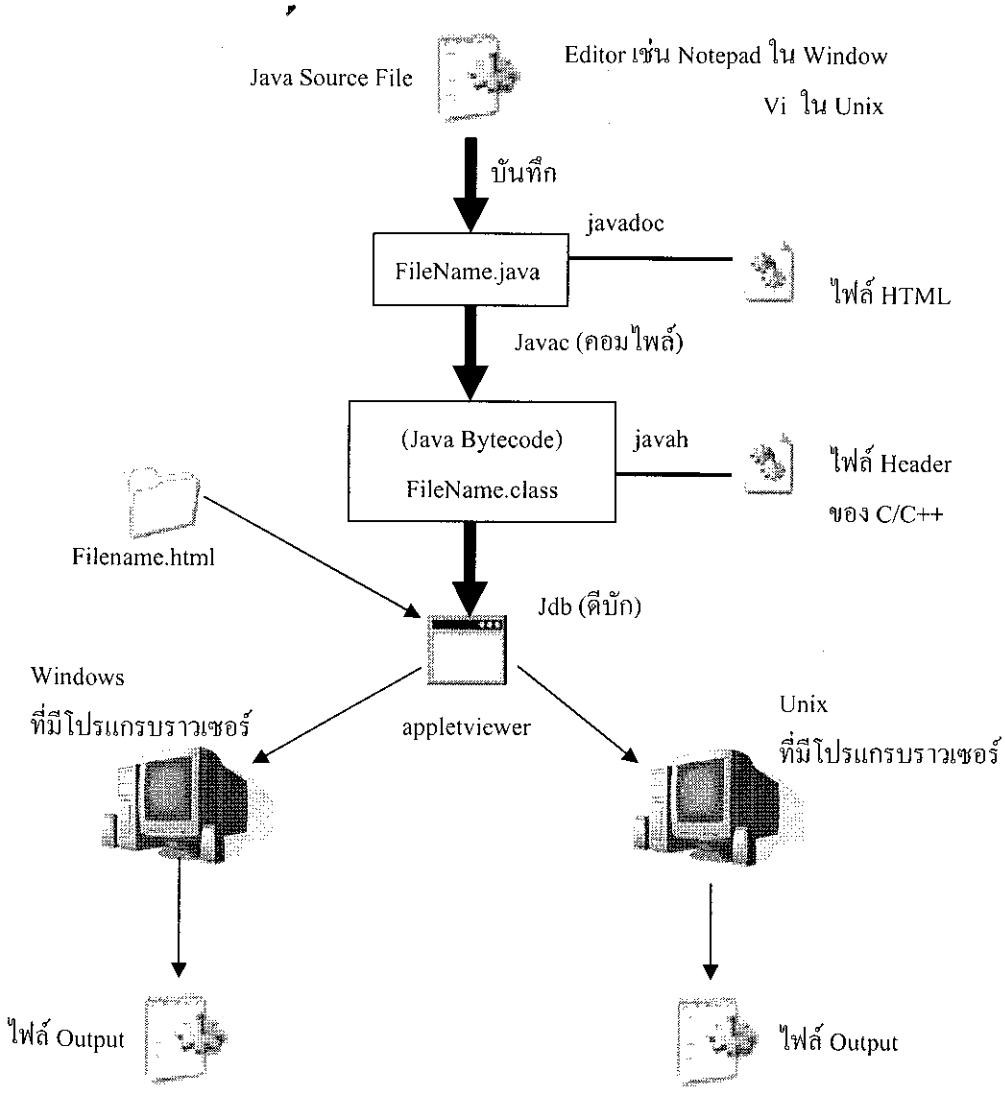
1.2 โปรแกรม ClientCamSender.java มีหน้าที่รอรับการติดต่อจากเครื่อง client ที่ต้องการดูภาพและ control ผ่านอินเตอร์เน็ต โดยจะใช้ port 7002 ในการติดต่อ เมื่อมีการติดต่อจากเครื่อง client แล้ว ก็จะส่งภาพที่ได้จากโปรแกรมย่อย ClientReceiveImage.java ไปยังเครื่อง client

1.3 โปรแกรม ClientReceiveControl.java มีหน้าที่รอรับการติดต่อจากเครื่อง client ที่ต้องการดูภาพและ control ผ่านอินเตอร์เน็ต โดยจะใช้ port 7003 ในการติดต่อ เมื่อมีการติดต่อจากเครื่อง client แล้วก็จะรับ control แล้วส่งไปยังโปรแกรมย่อย ClientControlSender.java

1.4 โปรแกรม ClientControlSender.java มีหน้าที่รอรับการติดต่อจากเครื่อง client ที่มีกล้องวงจรปิด โดยจะใช้ port 7004 ในการติดต่อ เมื่อมีการติดต่อจากเครื่อง client แล้วจะทำการส่ง control ที่รับมาจากโปรแกรมย่อย ClientReceiveControl.java ไปยังเครื่อง client

จากโปรแกรม Java Application ที่ใช้ในโครงงานนี้ข้างต้น สามารถดู source code และ flow chat การทำงานของโปรแกรมต่างๆ ได้จากภาคผนวก ฉ

3.1.2 โปรแกรม Java แบบ Applet



รูปที่ 3-2 แสดงขั้นตอนการสร้าง Java

การเขียนโปรแกรม Java แบบ Applet [1] สามารถทำได้เช่นเดียวกับแบบ Application จนถึงขั้นสร้างไฟล์ .class ซึ่งเมื่อถึงขั้นได้ไฟล์ .class จะแตกต่างกันโดยไม่ต้องเรียกใช้ Interpreter (java.exe) เหมือนกับ Application แต่จะถูกเรียกใช้โดยไฟล์ HTML ซึ่งฝังโค้ดสำหรับเรียกใช้ไฟล์ .class ไว้แล้ว สามารถดูผลงานได้จาก appletviewer.exe ได้ แทนการเรียกจากบราวเซอร์โดยระบุไฟล์ HTML ดังกล่าว จากการเขียนโปรแกรม Java แบบ Applet ข้างต้น ทำให้สามารถสร้าง Java applet ได้จากขั้นตอนการสร้าง Java appletในภาคผนวก ค [5]

โปรแกรม Java Applet ที่ใช้ในโครงงานนี้มีดังนี้

1. โปรแกรม clientcamsender.java เป็นโปรแกรมทางฝั่งเครื่อง client ที่มีกล้องวงจรปิด ซึ่งมีหน้าที่ติดต่อกับเครื่อง server เพื่อส่งภาพจากกล้องวงจรปิดไปยังเครื่อง server รวมทั้งรับ control จากเครื่อง server โดยการทำงานของโปรแกรมนี้จะเป็นการทำงานผ่านโปรแกรมย่อยต่างๆ ดังนี้

1.1 โปรแกรม ClientCam.java มีหน้าที่นำภาพจากกล้องวงจรปิดที่เข้าการ์ด capture มาแสดงบน panel

1.2 โปรแกรม ClientSocket.java มีหน้าที่ติดต่อกับเครื่อง server ผ่าน port 7001 เพื่อส่งภาพไปยังเครื่อง server

1.3 โปรแกรม ClientChat.java มีหน้าที่ติดต่อกับเครื่อง server ผ่าน port 7004 เพื่อรับ control จากเครื่อง server จากนั้นจึงส่ง control ไปยังกล่องควบคุมกล้องโดยผ่าน serial port

2. โปรแกรม Imagereceive.java เป็นโปรแกรมทางฝั่งเครื่อง client ที่ต้องการดูภาพและ control กล้องผ่านระบบอินเตอร์เน็ต มีหน้าที่ติดต่อกับเครื่อง server เพื่อรับภาพจากเครื่อง server แล้วนำไปแสดงที่จอภาพของเครื่อง client รวมทั้งส่ง control ไปยังเครื่อง server การทำงานของโปรแกรมนี้จะเป็นการทำงานผ่านโปรแกรมย่อยต่างๆ ดังนี้

2.1 โปรแกรม Frame1.java มีหน้าที่สร้าง frame สำหรับเขียนภาพลงบน frame

2.2 โปรแกรม Connect.java มีหน้าที่ติดต่อกับเครื่อง server ผ่าน port 7002 เพื่อรับภาพจากเครื่อง server แล้วนำภาพเขียนลงบน frame ที่เตรียมไว้ จากนั้นจึงส่ง frame ที่ได้เขียนภาพเรียบร้อยแล้วไปแสดงบน applet

2.3 โปรแกรม ClientControlSender.java มีหน้าที่ติดต่อกับเครื่อง server ผ่าน port 7003 เพื่อส่ง control ที่รับจากผู้ใช้ไปยังเครื่อง server

จากโปรแกรม Java Applet ที่ใช้ในโครงงานนี้ข้างต้น สามารถดู source code และ flow chat การทำงานของโปรแกรมต่างๆ ได้จากภาคผนวก ฉ

3.2 การเขียนโปรแกรมเพื่อรับ control จากเครื่อง client และควบคุมการทำงานของกล้องวงจรปิด

3.2.1 การอ่านหรือรับข้อมูลจากพอร์ตอนุกรม

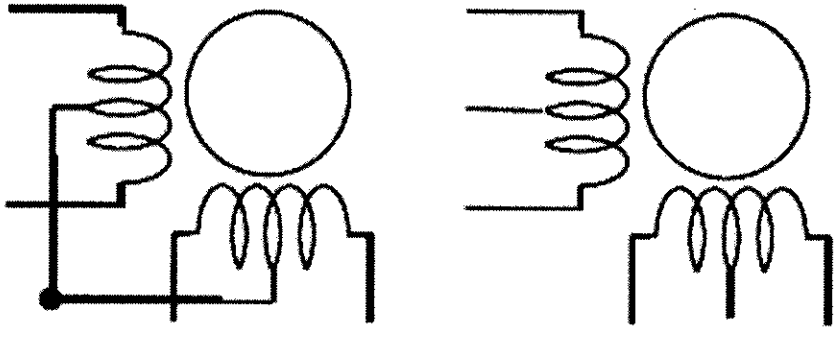
การรับข้อมูลจากพอร์ตอนุกรมสามารถกระทำได้ง่ายมาก เพียงทำการตรวจสอบว่าบิต RI เกิดการเซตขึ้นหรือไม่ ถ้าพบว่ามีการเซตเกิดขึ้นแล้ว ให้ทำการอ่านค่าจากรีจิสเตอร์ SBUF โดยต้องทำการโอนย้ายข้อมูลผ่านทางแอกคิวมูเลเตอร์หรือรีจิสเตอร์ A ดังตัวอย่าง [9]

```
CLR   RI          ; เคลียร์บิต RI เพื่อเตรียมการรับข้อมูล
JNB   RI, $       ; รอคอยการเซตของบิต RI อันเป็นการแจ้งให้ทราบว่า การรับข้อมูล
                    ; เสร็จสมบูรณ์และมีข้อมูลเกิดขึ้นที่รีจิสเตอร์ SBUF
MOV   A, SBUF,    ; อ่านค่าจากรีจิสเตอร์ โดยการโอนย้ายข้อมูลผ่านทางรีจิสเตอร์ A
CLR   RI          ; หลังจากทำการอ่านข้อมูลเรียบร้อยแล้ว ต้องทำการเคลียร์บิต RI
```

3.2.2 การขับสเต็ปมอเตอร์

สเต็ปมอเตอร์ได้รับการพัฒนามาอย่างต่อเนื่อง จนในปัจจุบันสเต็ปมอเตอร์ที่นิยมใช้อย่างแพร่หลายมากที่สุด และหาได้ง่ายคือ สเต็ปมอเตอร์แบบยูนิโพล่าร์ มีลักษณะการพันขดลวดของมอเตอร์แสดงในรูปที่ 3-3

สเต็ปมอเตอร์แบบนี้มีการพันขดลวด 2 ขดในแต่ละขั้วแม่เหล็กของสเตเตอร์ แต่ละขดแบ่งเป็น 2 เฟส รวมมอเตอร์ทั้งตัวจะมี 4 เฟสคือ เฟส 1,2,3 และ 4 มีการต่อสายออกมาจากขดลวดแต่ละขดเพื่อจ่ายไฟเลี้ยง ทำให้สเต็ปมอเตอร์แบบนี้มีทั้งแบบ 5 สายและ 6 สายถ้าเป็นแบบ 5 สาย จะเป็นการนำสายไฟเลี้ยงของขดลวดทั้งสองมาต่อรวมกันเป็นสายเดียว



แบบ 5 สาย 4 เฟส แบบ 6 สาย 4 เฟส

รูปที่ 3-3 โครงสร้างอย่างง่ายของสเต็ปมอเตอร์แบบยูนิโพล่าร์

จากโครงงานเราเลือกใช้สเต็ปมอเตอร์แบบยูนิโพล่าร์ แบบ 6 สาย 4 เฟส มอเตอร์ชนิดนี้จะมีลักษณะการทำงานเหมือนกับสเต็ปมอเตอร์แบบยูนิโพล่าร์ แบบ 5 สาย 4 เฟส แต่จะต่างกันที่แบบ 5 สาย จะเป็นการนำสายไฟเลี้ยงของขดลวดทั้งสองมาต่อรวมกันเป็นสายเดียว ในการใช้งานไม่ว่าจะเป็นแบบ 6 สาย หรือ 5 สาย เราสามารถเลือกใช้แบบใดก็ได้ ก็ขึ้นอยู่กับความเหมาะสมของการใช้งาน

การกระตุ้นและควบคุมการหมุนของสเต็ปมอเตอร์

การกระตุ้นและควบคุมการหมุนของมอเตอร์ให้เคลื่อนที่ไปแต่ละสเต็ปทำได้โดยจ่ายกำลังไฟฟ้าไปยังขดลวดแต่ละขดบนสเตเตอร์ ซึ่งต้องป้อนเป็นแบบซีเควนเชียลในรูปแบบที่ถูกต้องด้วย สามารถแบ่งได้เป็น 3 รูปแบบคือ แบบฟูลสเต็ปหนึ่งเฟส, แบบฟูลสเต็ป 2 เฟส และแบบฮาล์ฟสเต็ป (half step) [4]

แบบฟูลสเต็ปหนึ่งเฟส (full step) เป็นการกระตุ้นที่มีรูปแบบง่ายที่สุด โดยทำการกระตุ้นขดลวดทีละขดในเวลาหนึ่งไล่เรียงถัดกันไป เช่น เริ่มต้นที่ขดที่ 1, 2, 3, 4 แล้ววนกลับมาขดที่ 1 แล้ววนไปเรื่อย ๆ หรือเริ่มที่ขดที่ 1 แล้วย้อนไปยังขดที่ 4, 3, 2 แล้ววนมายังขดที่ 1 อีกครั้ง ซึ่งทำให้ทิศทางการหมุนสวนกัน ในการกระตุ้นรูปแบบนี้จึงมีขดลวดเพียงขดเดียวในเวลาหนึ่งที่ถูกกระตุ้นเท่านั้น วงจรกระตุ้นแบบเวฟจึงมีราคาถูกและง่าย ขั้นตอนการทำงานต่างๆ แสดงดังตารางที่ 5

สเต็ปที่ 1	เฟสที่ 1	เฟสที่ 2	เฟสที่ 3	เฟสที่ 4
1	ทำงาน	-	-	-
2	-	ทำงาน	-	-
3	-	-	ทำงาน	-
4	-	-	-	ทำงาน

ตารางที่ 5 แสดงลำดับการทำงานของขดลวดในแต่ละเฟสของมอเตอร์ เมื่อได้รับการกระตุ้นแบบฟูลสเต็ปหนึ่งเฟส

แบบฟูลสเต็ปสองเฟสเป็นการกระตุ้นซึ่งคล้ายกับแบบหนึ่งเฟส แต่การกระตุ้นแบบนี้จะทำการกระตุ้นโดยจ่ายกำลังไฟฟ้าไปที่ขดลวด 2 ขด ที่อยู่ใกล้กันในเวลาเดียวกัน และเรียงถัดกันไปเช่นเดียวกันกับแบบหนึ่งเฟส ดังตัวอย่าง ขดลวดชุดแรกที่ถูกกระตุ้นจะเป็นขดที่ 1 และ 2 ตามด้วยการกระตุ้นขดที่ 2 และ 3 ต่อไปเป็นขดที่ 3 และ 4 ถัดไปเป็นขดที่ 4 และ 1 แล้ววนกลับมาที่ขดที่ 1 และ 2 วนไปตามลำดับเช่นนี้ หรือเริ่มที่ขดที่ 1 และ 4 ตามด้วยขดที่ 4 และ 3 ถัดไปเป็นขดที่ 3 และ 2 ต่อไปเป็นขดที่ 2 และ 1 แล้ววนกลับมาที่ขดที่ 1 และ 4 ทิศทางการหมุนจะสวนทางกัน การกระตุ้นสเต็ปมอเตอร์แบบนี้สามารถเพิ่มแรงบิดได้มากกว่าแบบเวฟ โรเตอร์จะเคลื่อนที่ด้วยแรงดึงอย่างเต็มแรงจาก 2 ขดลวดที่ถูกกระตุ้น

พร้อมกัน และต่อไปด้วยแรงดึงจากอีก 2 ขดลวดถัดไป สำหรับข้อเสียคือการกระตุ้นแบบนี้จะต้องใช้กำลังไฟฟ้ามากขึ้น ขั้นตอนการทำงานต่าง ๆ แสดงดังในตารางที่ 6

สเต็ปที่ 1	เฟสที่ 1	เฟสที่ 2	เฟสที่ 3	เฟสที่ 4
1	ทำงาน	ทำงาน	-	-
2	-	ทำงาน	ทำงาน	-
3	-	-	ทำงาน	ทำงาน
4	ทำงาน	-	-	ทำงาน

ตารางที่ 6 แสดงลำดับการทำงานของขดลวดในแต่ละเฟสของมอเตอร์ เมื่อได้รับการกระตุ้นแบบฟูลสเต็ปสองเฟส

แบบฮาล์ฟสเต็ปเป็นรูปแบบที่ผสมผสานระหว่างการกระตุ้นแบบฟูลสเต็ปหนึ่งเฟส และ 2 เฟส เพื่อเพิ่มจำนวนของสเต็ปต่อรอบอีกเท่าตัวหนึ่ง ในระบบนี้จะทำการกระตุ้นขดลวดเรียงกันไปเป็นลำดับเป็นดังนี้ เริ่มจากขดลวดที่ 1, 1 และ 2, 2, 2 และ 3, 3, 3 และ 4, 4, 4 และ 1 แล้ววนกลับมายังขดลวดที่ 1 แรงบิดที่ได้จากการกระตุ้นแบบนี้จะเพิ่มมากขึ้นอีก เพราะช่วงสเต็ปมีระยะสั้นลง แต่ละสเต็ปเกิด แรงเกิดแรงดึงจากขดลวด 2 ขดที่ถูกกระตุ้นพร้อมกัน ความถูกต้องของตำแหน่งมีเพิ่มมากขึ้น แต่ต้องพึงระวังไว้อีกประการหนึ่งว่า เมื่อกระตุ้นให้ทำงานในรูปแบบนี้จะต้องทำการหมุนถึง 2 สเต็ป จึงจะได้เท่ากับระยะเท่ากับ 1 สเต็ปเต็มของการควบคุมใน 2 แบบแรก สำหรับแหล่งจ่ายกำลังไฟฟ้าต้องใช้ขนาดเท่ากับแบบ 2 เฟสเป็นอย่างน้อย จึงจะเพียงพอ ขั้นตอนการทำงานต่าง ๆ แสดงดังในตารางที่ 7

สเต็ปที่ 1	เฟสที่ 1	เฟสที่ 2	เฟสที่ 3	เฟสที่ 4
1	ทำงาน	-	-	-
2	ทำงาน	ทำงาน	-	-
3	-	ทำงาน	-	ทำงาน
4	-	ทำงาน	ทำงาน	ทำงาน
5	-	-	ทำงาน	-
6	-	-	ทำงาน	ทำงาน
7	-	-	-	ทำงาน
8	ทำงาน	-	-	ทำงาน

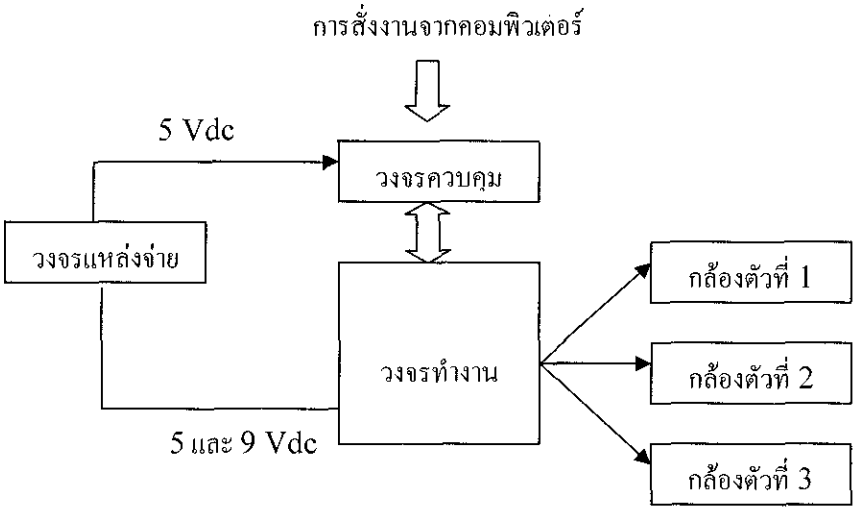
ตารางที่ 7 แสดงลำดับการทำงานของขดลวดในแต่ละเฟสของมอเตอร์ เมื่อได้รับการกระตุ้นแบบฮาล์ฟสเต็ป

จากโครงงานเราใช้การกระตุ้นแบบฟลูสเต็ปหนึ่งเฟสให้กับมอเตอร์ เพราะว่าเป็นการกระตุ้นที่มีรูปแบบที่ง่าย และไม่ต้องใช้กำลังไฟมากนัก เนื่องจากกล้องที่ติดกับมอเตอร์มีน้ำหนักไม่มากและ ไม่จำเป็นจะต้องใช้แรงบิดมาก การกระตุ้นแบบฟลูสเต็ปหนึ่งเฟสจึงเหมาะสมกับการทำงานมากกว่าการกระตุ้นแบบอื่น

3.3 การสร้างอุปกรณ์สำหรับควบคุมกล้องวงจรปิด

3.3.1 ขอบเขตความสามารถของอุปกรณ์

อุปกรณ์ชุดนี้จัดทำขึ้นเพื่อรองรับการสั่งการจากคอมพิวเตอร์ ซึ่งอุปกรณ์ชุดนี้สามารถรับการสั่งการทางอินเตอร์เน็ตโดยผ่านเว็บบราวเซอร์ (Web Browser) ซึ่งจะทำให้ผู้ใช้สามารถเข้าถึงการทำงานของอุปกรณ์ได้ โดยในโครงงานนี้จะเลือกกล่าวถึงอุปกรณ์เพียง 4 อุปกรณ์ซึ่งจะอยู่ใน 3 ส่วนหลักๆ คือ ส่วนของวงจรแหล่งจ่าย ส่วนของวงจรควบคุม และ ส่วนของวงจรทำงาน ซึ่งสามารถแสดงเป็นแผนภาพการทำงานของอุปกรณ์ได้ดังนี้



รูปที่ 3-4 แสดงอุปกรณ์

ในวงจรแหล่งจ่ายจะประกอบไปด้วย

1. วงจรเรียงกระแสแบบ Full Wave ซึ่งไดโอดทำหน้าที่เรียงกระแสที่ถูกแปลงแรงดันจากหม้อแปลง 220/9 V แล้วทำให้ได้แรงดันกระแสตรงซึ่งนำมาจ่ายให้กับ Regulator
2. Regulator 2 ตัว คือ 5 V และ 9 V เพื่อนำไปเป็นไฟเลี้ยงให้กับวงจรควบคุมและวงจรทำงาน

ในวงจรควบคุมประกอบไปด้วย

1. ไมโครคอนโทรลเลอร์ เบอร์ AT89S8252 ซึ่งมีหน้าที่เอาสัญญาณที่เข้ามาจากพอร์ต Serial แล้วทำการเลือกค่าลอจิก

2. MAX 232 เพื่อทำการเปลี่ยนระดับแรงดันจาก ± 15 V เป็นระดับของสัญญาณ TTL หรือ ± 5 V

ในวงจรทำงานประกอบไปด้วย

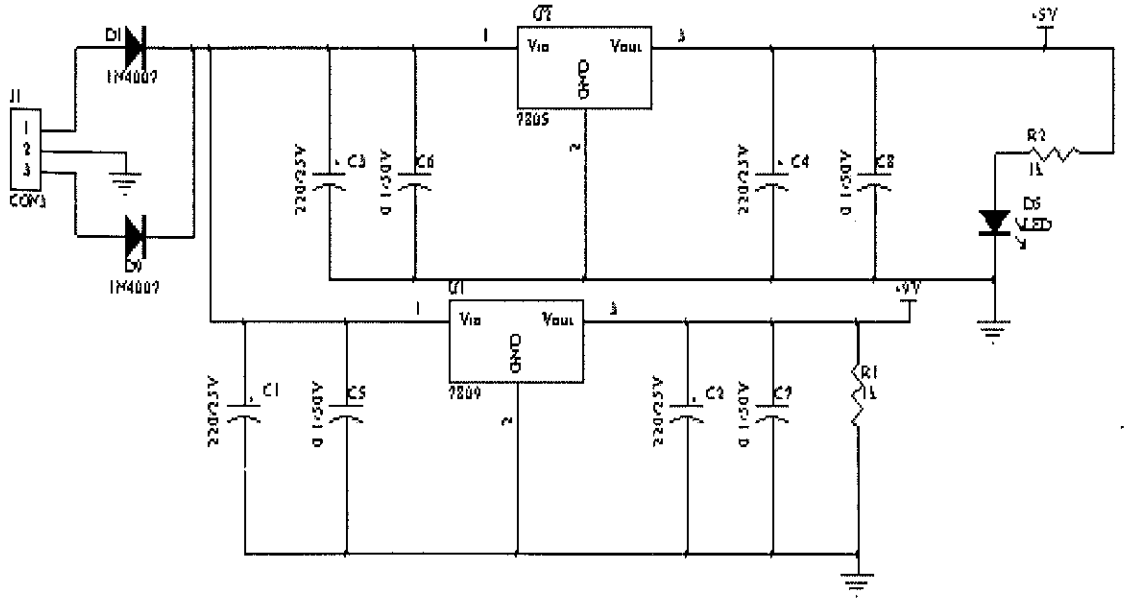
1. วงจรขับรีเลย์แบบใช้ทรานซิสเตอร์ เพื่อทำหน้าที่เป็น Switch
2. ULN2003 ใช้ในการขับโหลด (Stepping motor)

3.3.2 อธิบายการทำงานของวงจร

จากภาพวงจรที่ได้กล่าวไว้ในข้างต้น ซึ่งสามารถแบ่งออกเป็น 3 ส่วน คือ

1. ส่วนของวงจรจ่ายไฟ

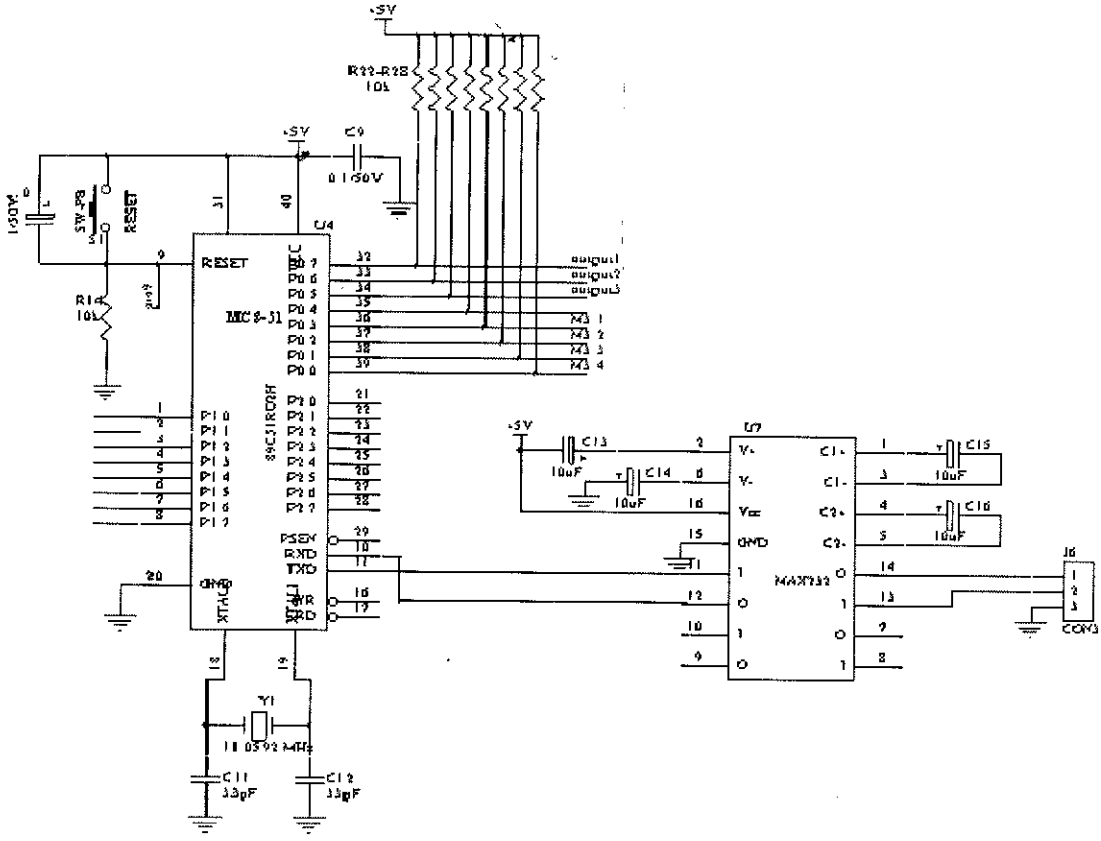
วงจรในส่วนนี้มีหน้าที่ในการจ่ายแรงดันเพื่อเป็นไฟเลี้ยงในวงจรและเป็นแหล่งจ่ายให้กับอุปกรณ์ภายนอก ส่วนประกอบในวงจรมีหม้อแปลงทำหน้าที่แปลงแรงดันไฟฟ้าจาก 220 โวลต์ มาเป็น 9 โวลต์ แล้วผ่านวงจรเรียงกระแสโดยใช้ไดโอด 2 ตัว เพื่อให้แรงดันมีความคงที่มากขึ้น แล้วส่งแรงดันผ่านมาให้กับ Regulator เบอร์ 7805 และ Regulator เบอร์ 7809 โดยที่ Regulator เบอร์ 7805 จะทำหน้าที่แปลงระดับแรงดันแล้วจ่ายแรงดัน 5 โวลต์ ออกมาเพื่อเป็นไฟเลี้ยงไอซีต่างๆ ส่วน Regulator เบอร์ 7809 จะทำหน้าที่แปลงระดับแรงดันแล้วจ่ายแรงดัน 9โวลต์ ออกมาเพื่อเป็นไฟเลี้ยง Stepping Motor [3]



รูปที่ 3-5 แสดงวงจรจ่ายไฟ

2. ส่วนของวงจรควบคุม

ส่วนนี้จะมีวงจรที่ทำหน้าที่รับค่าอินพุตจากพอร์ต Serial เพื่อนำค่าอินพุตที่เข้ามาไปส่งให้รีเลย์ทำงาน ในส่วนนี้จะประกอบไปด้วย ไมโครคอนโทรลเลอร์ เบอร์ AT89S8252 ซึ่งมีหน้าที่เอาสัญญาณที่เข้ามาจากพอร์ต Serial แล้วทำการเลือกค่าลอจิก แล้วส่งไปให้ ไอซี เบอร์ 2N3904 ทำหน้าที่เป็นตัวกำหนดให้ตัวอุปกรณ์ ทำงานหรือไม่ทำงานโดยอาศัยค่าลอจิก ค่าลอจิกที่ได้รับจะมาจ่ายให้กับวงจรขับรีเลย์ ซึ่งจะทำให้อุปกรณ์ทำงาน

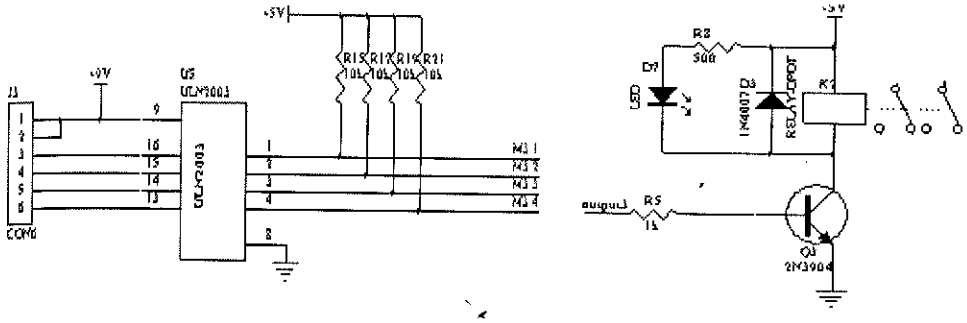


รูปที่ 3-6 แสดงวงจรควบคุม

3. ส่วนของวงจรทำงาน

ในส่วนของวงจรทำงานจะมีอยู่ด้วยกัน 2 อุปกรณ์ด้วยกันคือ ส่วนของรีเลย์ และ ส่วนของการขับ Stepping Motor ซึ่งรีเลย์ทำหน้าที่เป็นสวิตซ์เปิดปิด เพื่อจ่ายกระแสไฟฟ้า ให้อุปกรณ์ทำงาน รีเลย์ที่ใช้เป็นชนิด 5 โวลต์กระแสตรงซึ่งมีวงจรขับรีเลย์เป็นตัวส่งให้รีเลย์ทำงาน วงจรขับรีเลย์ประกอบด้วยทรานซิสเตอร์ที่ใช้รับสัญญาณจากวงจรขาเบส ทำให้รีเลย์มีการสวิตซ์ [3] เมื่อรีเลย์ทำงานก็จะทำให้

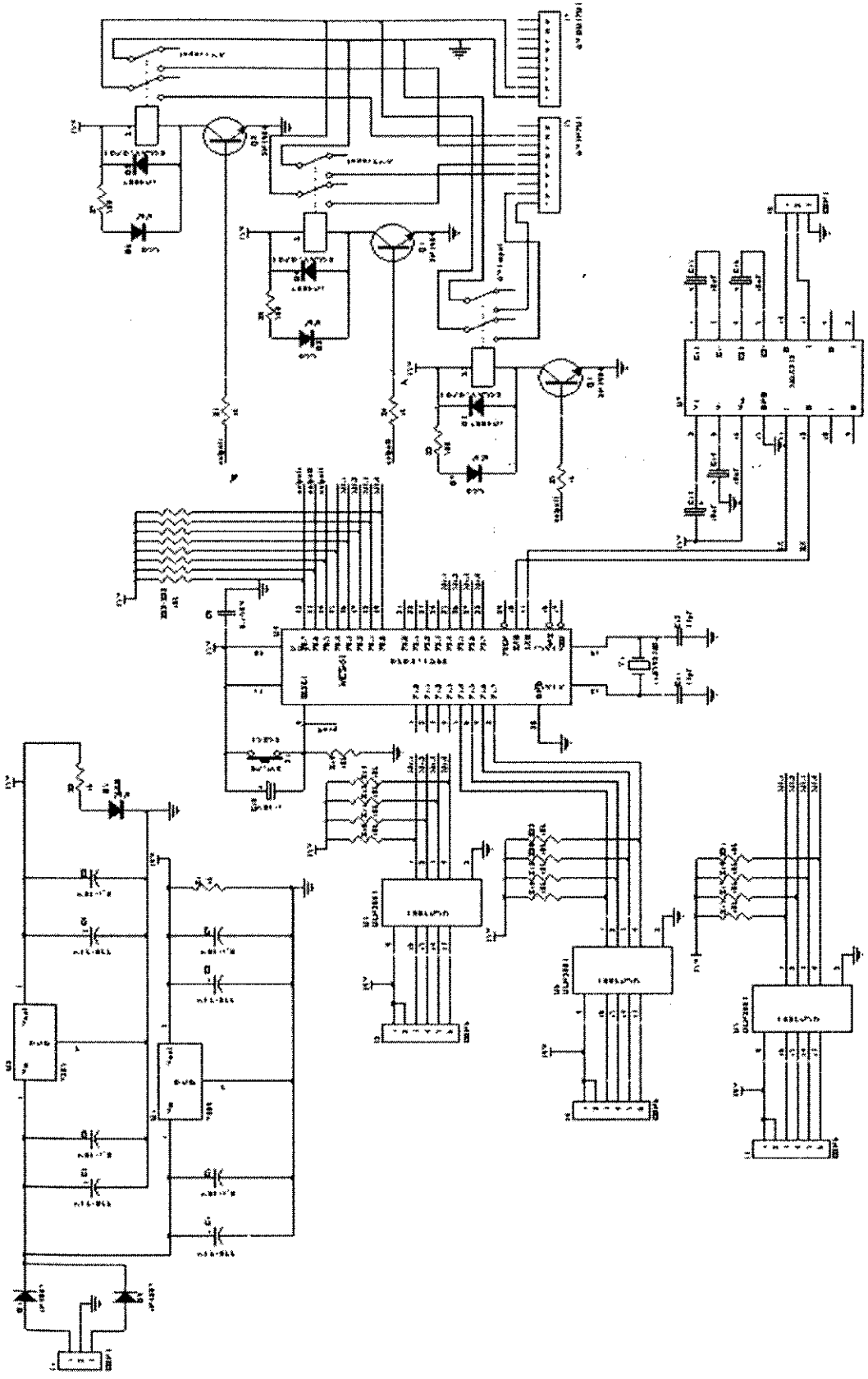
กล้องทำงาน ส่วน ULN2003 จะทำการขับมอเตอร์ที่กินกระแสไม่มากนัก ซึ่งเป็นตัวขับกระแสให้ Stepping Motor ซึ่งมีไดโอดต่อป้องกันไว้ภายในสำหรับขับรีเลย์ได้เลย



รูปที่ 3-7 แสดงวงจรทำงาน

4. วงจรรวม

ในส่วนนี้จะแสดงรูปของวงจรรวมทั้งหมดมาใช้ร่วมกัน โดยมีวงจรจ่ายไฟฟ้าเป็นตัวจ่ายไฟเลี้ยงให้กับวงจรทั้งหมด วงจรควบคุมทำหน้าที่ตัดสินใจการสั่งงาน และวงจรการทำงานทำหน้าที่เป็นสวิตซ์และควบคุมการทำงานของมอเตอร์ ซึ่งวงจรควบคุมการทำงานของกล้องวงจรปิดทั้งระบบสามารถดูรูปที่ 3-8



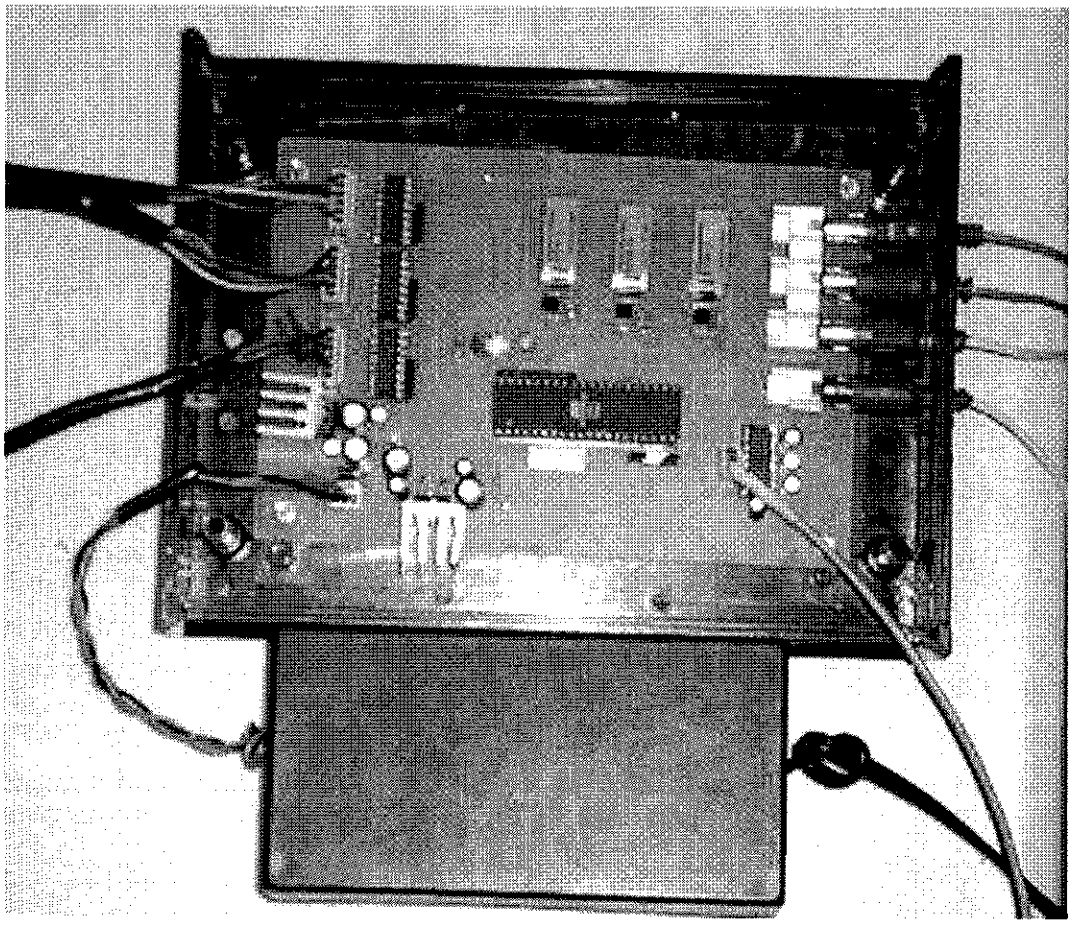
รูปที่ 3-8 แสดงวงจรรวมของชุดควบคุมกล้องวงจรปิด

บทที่ 4

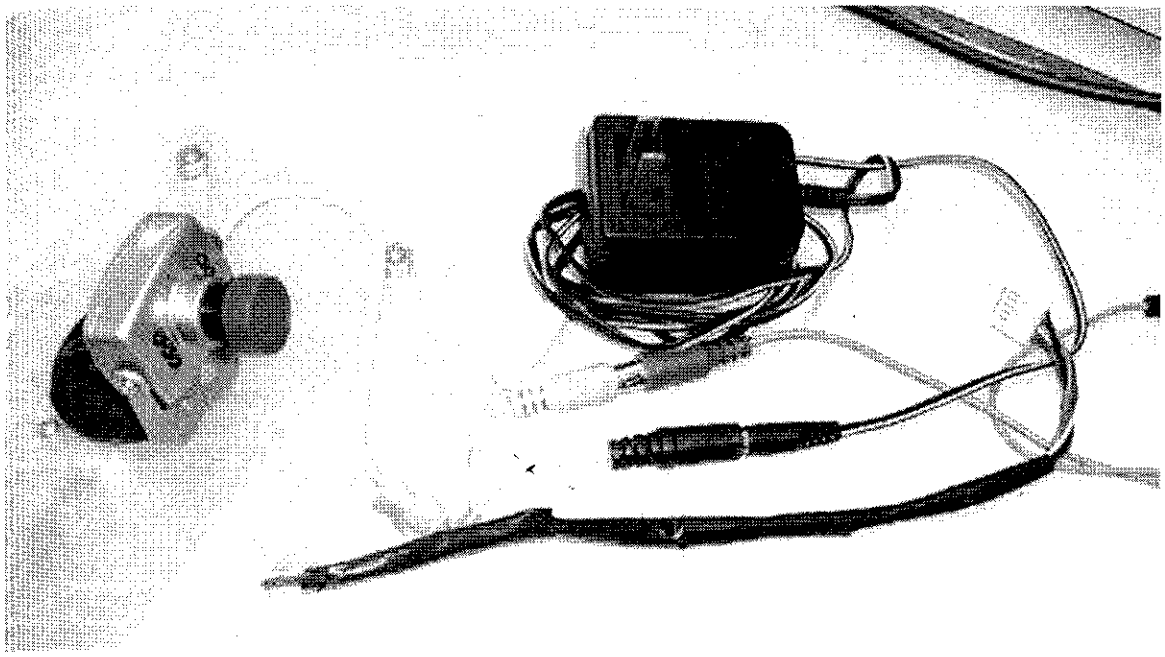
การทดลอง

4.1 การต่ออุปกรณ์

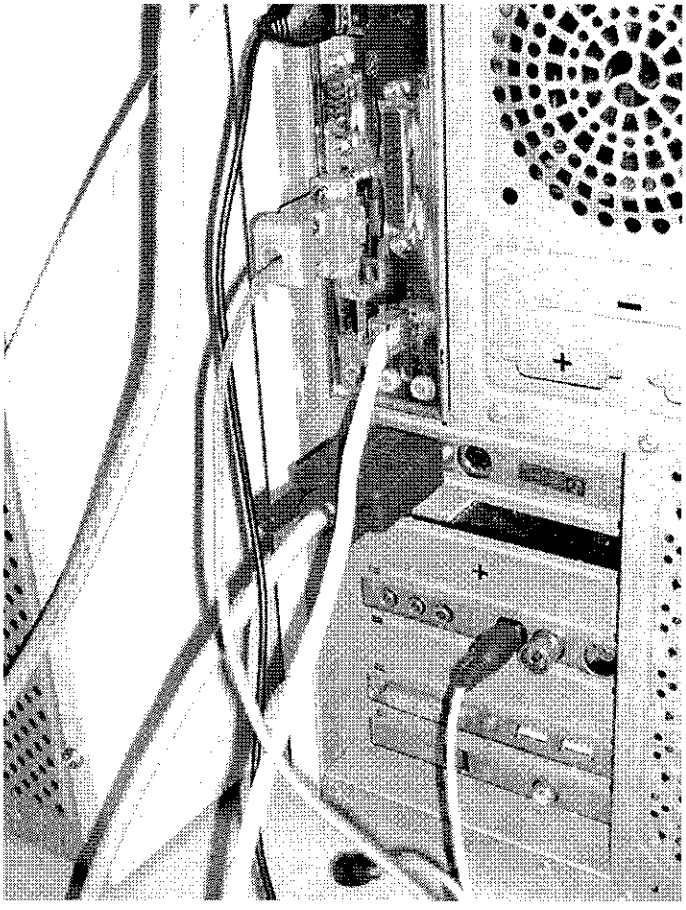
นำกล่องควบคุมกล้องวงจรปิดมาต่อเข้ากับคอมพิวเตอร์ โดยนำพอร์ตอนุกรมต่อเข้ากับพอร์ตอนุกรมของคอมพิวเตอร์ (com1 หรือ com2 ขึ้นอยู่กับโปรแกรม java ของคอมพิวเตอร์ที่ต่ออยู่กับคล้องวงจรปิด) แล้วนำปลายสายอีกข้างหนึ่งต่อเข้ากับ connector ที่ใช้ต่อกับพอร์ตอนุกรมภายในกล่องควบคุม จากนั้นจึงนำ connector ของสเต็ปปิ้งมอเตอร์ทั้ง 3 มาต่อเข้ากับ connector ของสเต็ปปิ้งมอเตอร์ที่อยู่ภายในกล่องควบคุมกล้อง แล้วนำสาย AV 3 เส้นต่อเข้ากับกล้องวงจรปิด (connector สีเหลือง) ส่วนปลายสายอีกด้านหนึ่งต่อเข้ากับ connector ของสัญญาณ AV ที่เป็นอินพุท นำสาย AV อีกหนึ่งเส้นต่อเข้ากับ connector ของสัญญาณ AV ที่ติดอยู่กับการ์ด capture แล้วนำปลายสายอีกด้านต่อเข้าที่ connector ของสัญญาณ AV ที่เป็นเอาท์พุท ต่อสาย adapter เข้ากับกล้องวงจรปิดทั้งสาม และนำ connector ของหม้อแปลงต่อเข้ากับ connector ของหม้อแปลงที่อยู่ภายในกล่องควบคุมกล้อง จากนั้นจึงเสียบปลั๊กของหม้อแปลง และ adapter ของกล้องวงจรปิด



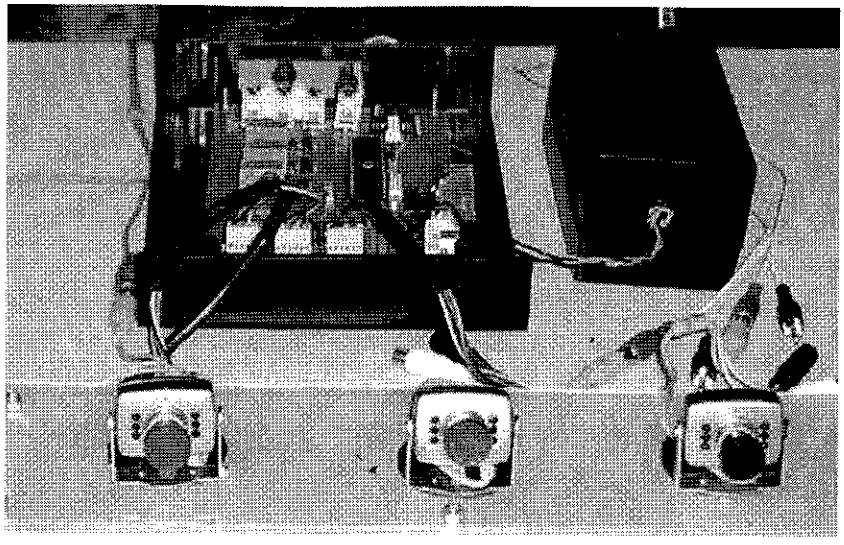
(ก)



(ข)



(ค)



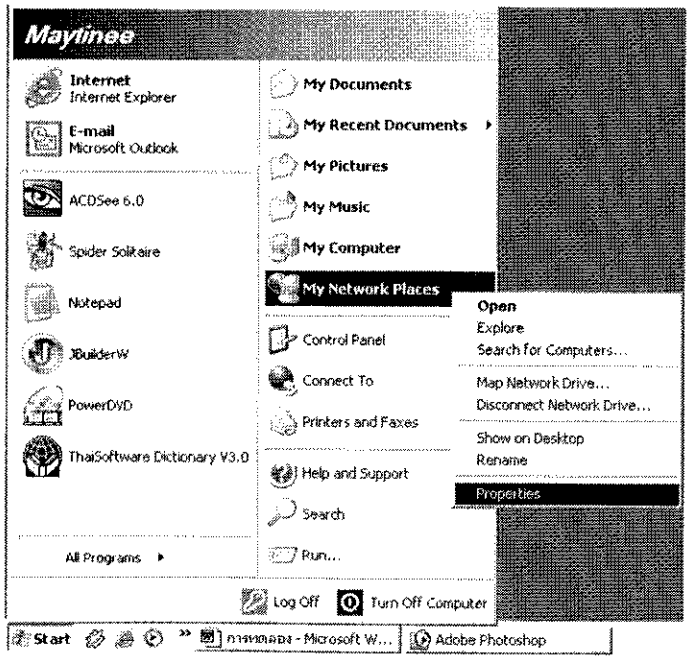
(ง)

รูปที่ 4-1 การต่ออุปกรณ์

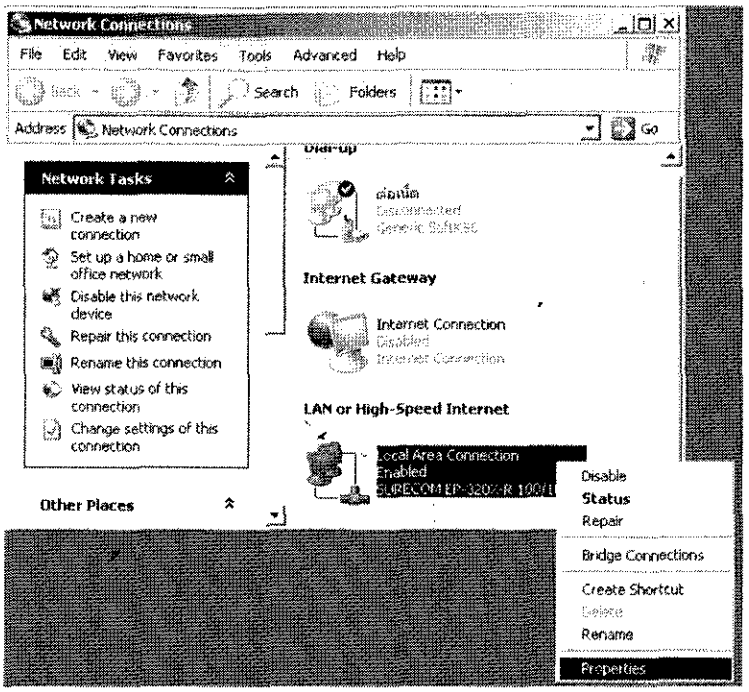
4.2 การใช้โปรแกรมเพื่อเริ่มทำงานของคอมพิวเตอร์ฝั่ง server

หลังจากที่ทำการติดตั้งโปรแกรมต่างๆ ที่ทำให้คอมพิวเตอร์ทำงานเป็นเครื่อง server และนำโปรแกรมทั้งหมดใส่ในเครื่อง server เรียบร้อยแล้ว

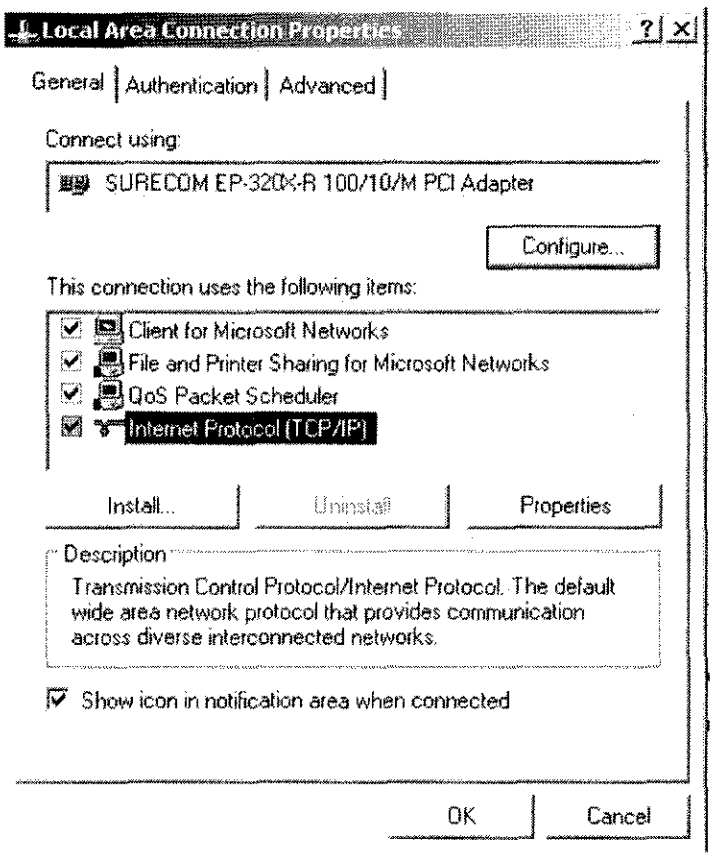
1. ทำการตั้ง IP address ของคอมพิวเตอร์เป็น 192.168.0.1 โดยคลิ๊กขวาที่ My Network Places เลือก Properties จากนั้นคลิ๊กขวาที่ Local Area Connection เลือก Properties แล้วเลือก Internet Protocol และคลิ๊กที่ Properties จะปรากฏหน้าต่างดังรูปที่ 4-2 (ง) จากนั้นจึงตั้งค่าต่างๆ ดังรูป แล้วคลิ๊ก OK



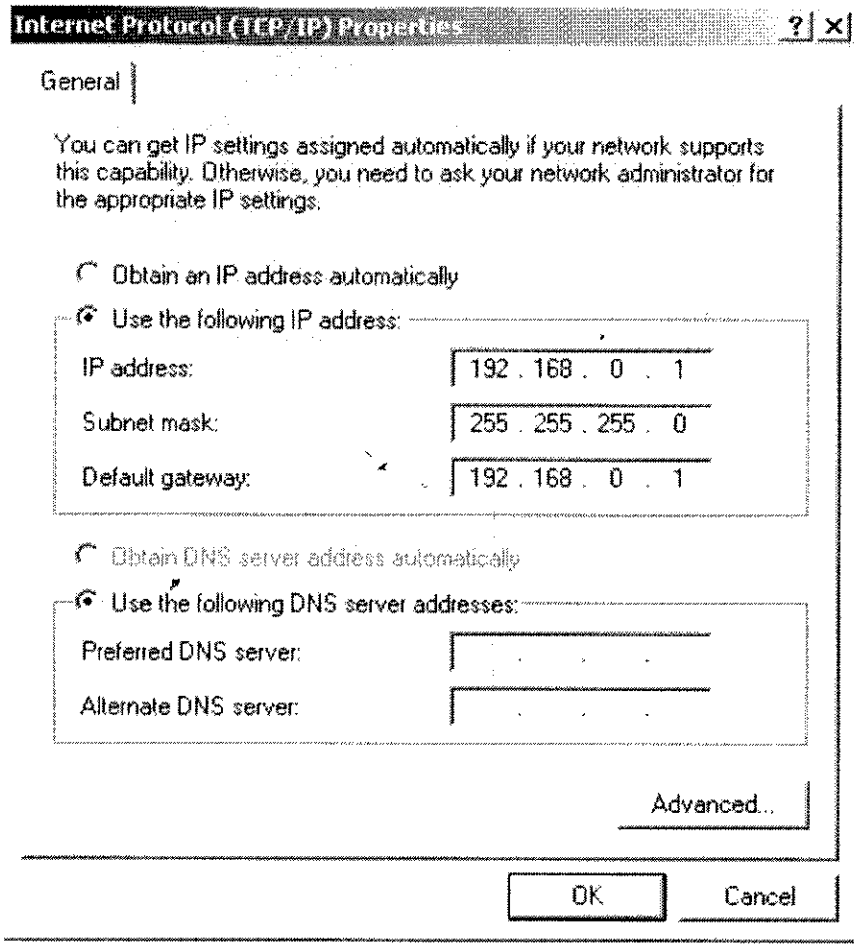
(ก)


Network Connections
File Edit View Favorites Tools Advanced Help
Search Folders
Address Network Connections
Go
Network Tasks
Create a new connection
Set up a home or small office network
Disable this network device
Repair this connection
Rename this connection
View status of this connection
Change settings of this connection
Other Places
Internet Gateway
Internet Connection
Disabled
LAN or High-Speed Internet
Local Area Connection
Enabled
Disable
Status
Repair
Bridge Connections
Create Shortcut
Delete
Rename
Properties

(ข)


Local Area Connection Properties
General Authentication Advanced
Connect using:
SURECOM EP-320X-R 100/10/M PCI Adapter
Configure...
This connection uses the following items:
Client for Microsoft Networks
File and Printer Sharing for Microsoft Networks
QoS Packet Scheduler
Internet Protocol (TCP/IP)
Install...
Uninstall
Properties
Description
Transmission Control Protocol/Internet Protocol. The default wide area network protocol that provides communication across diverse interconnected networks.
Show icon in notification area when connected
OK
Cancel

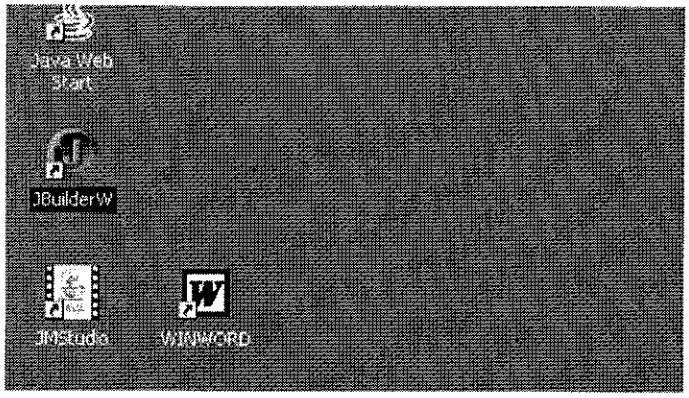
(ค)



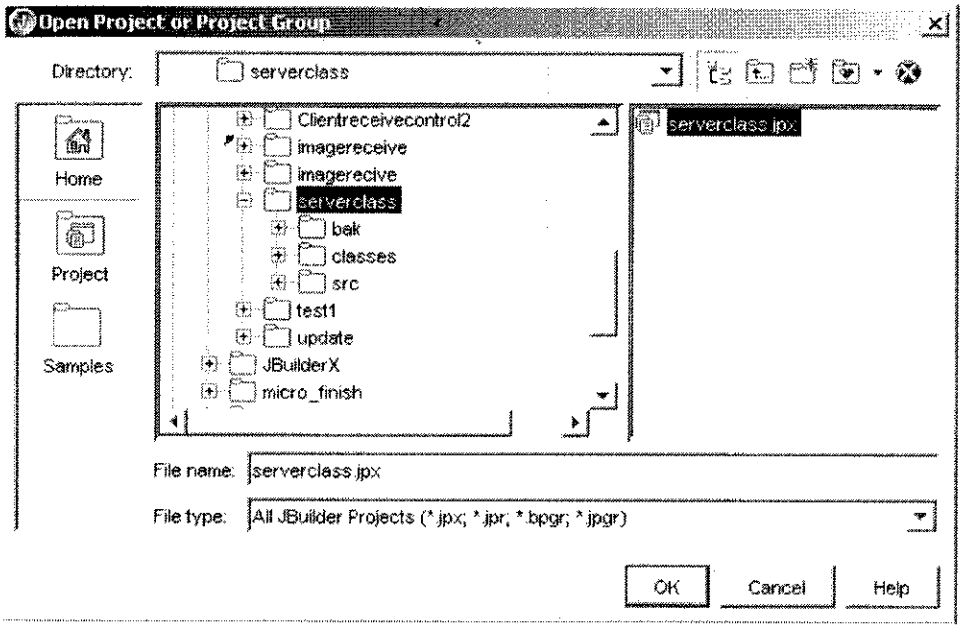
(ง)

รูปที่ 4-2 การตั้ง IP address

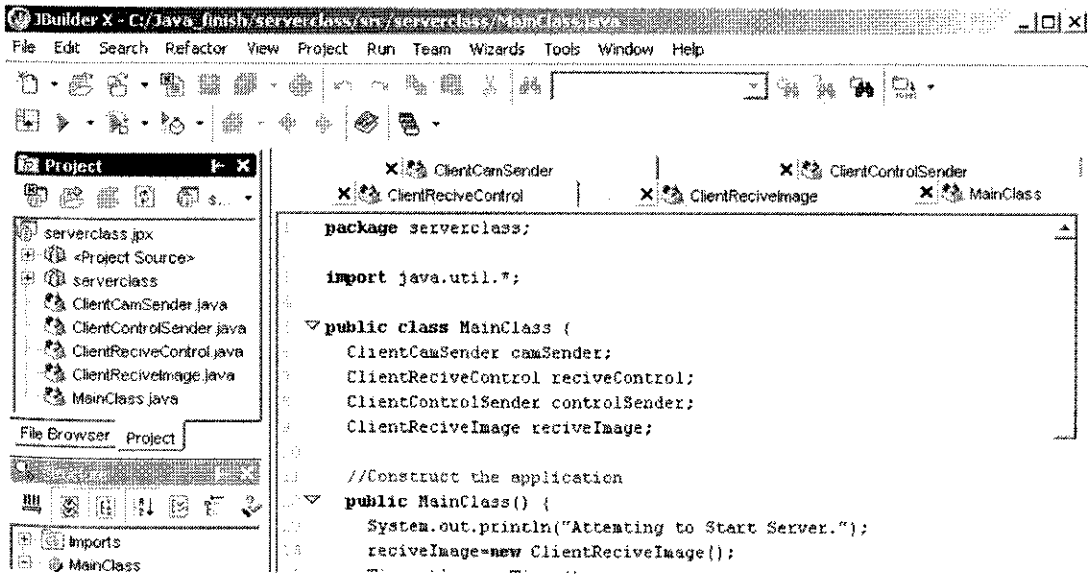
2. ทำการเปิดโปรแกรม JBuilder โดยดับเบิ้ลคลิ๊กที่ไอคอน JBuilderW จากนั้นคลิ๊ก Flie และเลือก Open Project เปิดไฟล์ชื่อ serverclass.jpx แล้วคลิ๊ก OK จะปรากฏโปรแกรมดังรูป 4-3 (ค) จากนั้นทำการ run โปรแกรมโดยคลิ๊กที่ run เลือก run Project หรือกด F9 จะได้ผลดังรูปที่ 4-3 (ง) ซึ่งแสดงว่า run โปรแกรมเรียบร้อยแล้ว ถ้าต้องการหยุด run โปรแกรมให้คลิ๊กที่เครื่องหมายกากบาท จะปรากฏหน้าต่างดังรูปที่ 4-3 (จ) ให้คลิ๊กที่ OK จะได้ผลเป็นดังรูปที่ 4-3 (ค) แสดงว่าโปรแกรมหยุด run เรียบร้อยแล้ว


Java Web Start
JBuilderW
JMStudio
WINWORD

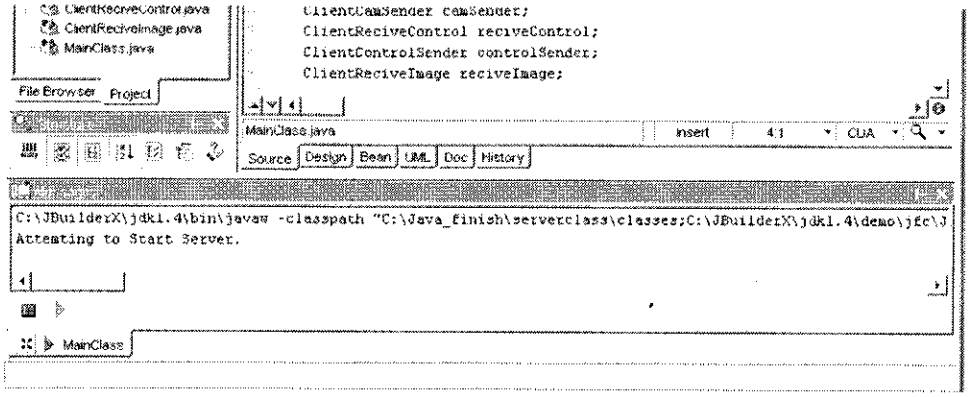
(ก)


Open Project or Project Group
Directory:
serverclass
Home
Project
Samples
Clientreceivecontrol2
imagereceive
imagerecive
serverclass
bak
classes
src
test1
update
JBuilderX
micro_finish
serverclass.jpx
File name:
serverclass.jpx
File type:
All JBuilder Projects (*.jpx; *.jpr; *.bpgr; *.jpgr)
OK
Cancel
Help

(ข)


JBuilder X - C:/Java_finish/serverclass/src/serverclass/MainClass.java
File Edit Search Refactor View Project Run Team Wizards Tools Window Help
Project
serverclass.jpx
<Project Source>
serverclass
ClientCamSender.java
ClientControlSender.java
ClientReciveControl.java
ClientReciveImage.java
MainClass.java
File Browser
Project
Imports
MainClass
ClientCamSender
ClientControlSender
ClientReciveControl
ClientReciveImage
MainClass
package serverclass;
import java.util.*;
public class MainClass {
ClientCamSender camSender;
ClientReciveControl reciveControl;
ClientControlSender controlSender;
ClientReciveImage reciveImage;
//Construct the application
public MainClass() {
System.out.println("Attemting to Start Server.");
reciveImage=new ClientReciveImage();

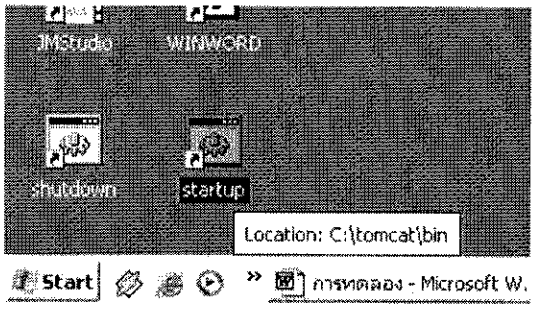
(ค)



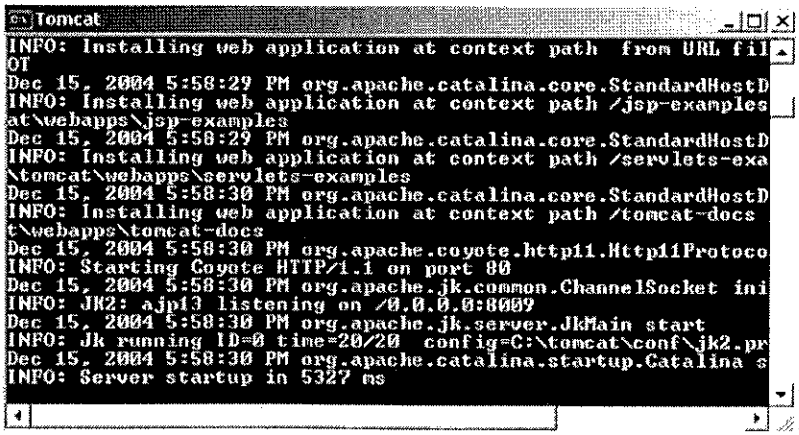
(ง)

รูปที่ 4-3 การ Runโปรแกรม serverclass.jpx

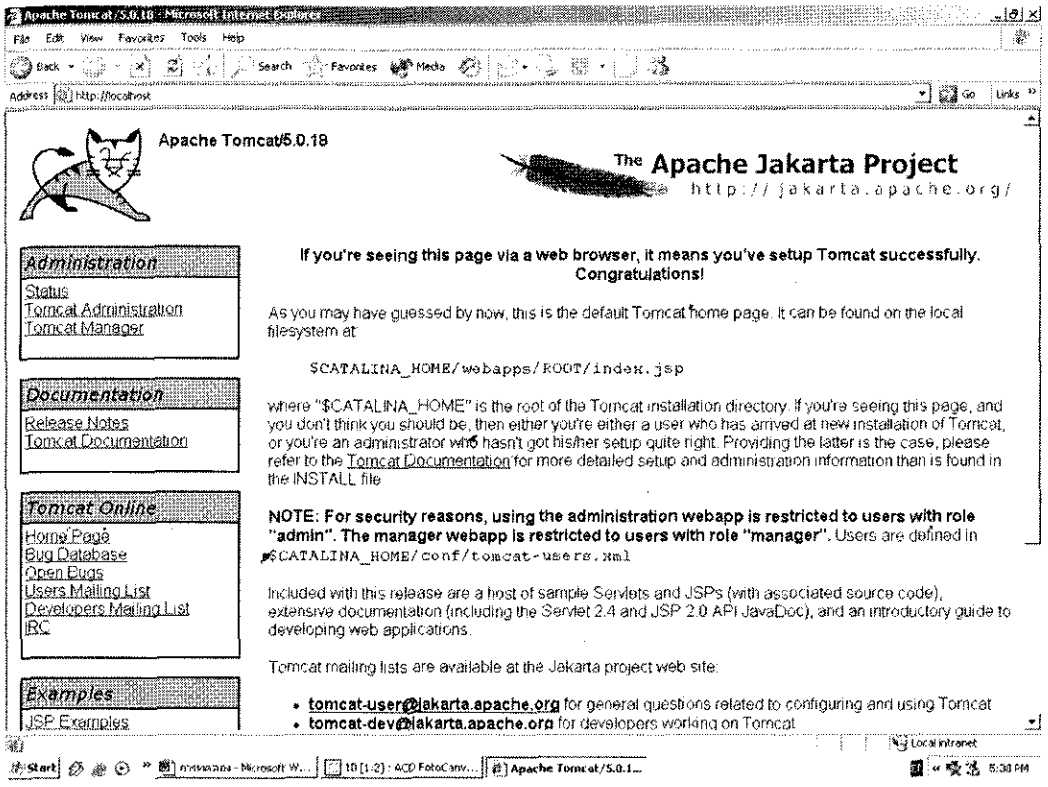
3. ทำการ run โปรแกรม jakarta-tomcat-5.0.18 (การติดตั้งดูได้จากภาคผนวก จ [7])โดยดับเบิ้ลคลิ๊กที่ไอคอน startup จะได้ผลดังรูปที่ 4-4 (ข) แสดงว่าเครื่องคอมพิวเตอร์เป็น server เรียบร้อยแล้ว พร้อมที่จะให้บริการแก่เครื่อง client ซึ่งเราสามารถทดสอบว่าโปรแกรม jakarta-tomcat run เรียบร้อยจริงหรือไม่ โดยการเปิดโปรแกรม Internet Explorer แล้วพิมพ์ว่า http://localhost ที่ Address Bar แล้ว Enter จะได้ผลดังรูปที่ 4-4 (ค) แสดงว่า run jakarta-tomcat-5.0.18 เรียบร้อยแล้ว เมื่อต้องการที่จะหยุด run โปรแกรมให้ทำการดับเบิ้ลคลิ๊กที่ไอคอน shutdown



(ก)



(ข)



(ค)

รูปที่ 4-4 การ Run โปรแกรม jakarta-tomcat-5.0.18

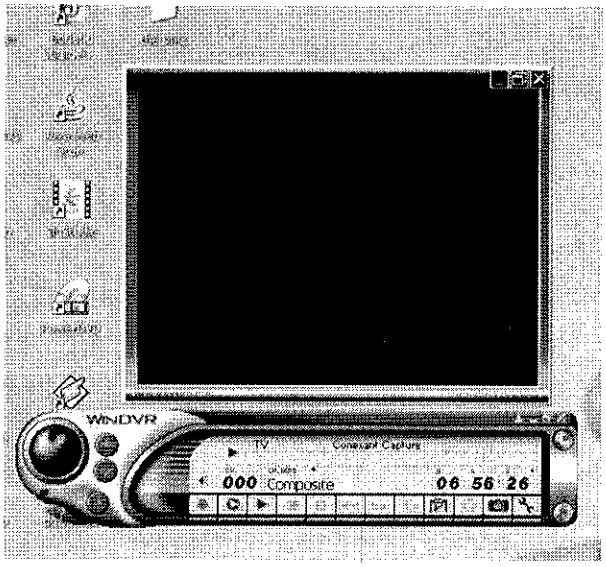
เมื่อทำการตั้งค่าและประมวลผลโปรแกรมตามขั้นตอนทั้ง 3 เรียบร้อยแล้ว จะทำให้เครื่องคอมพิวเตอร์เป็นเครื่อง server ที่พร้อมจะให้บริการต่างๆ แก่เครื่อง client ที่ทำการเชื่อมต่อมายังเครื่อง server โดยเครื่อง server จะทำการเปิด port เบอร์ 7001 เพื่อใช้ในการรับภาพจากเครื่อง client ที่มีกล้องวงจรปิด port เบอร์ 7002 ใช้ส่งภาพไปยังเครื่อง client ที่ควบคุมกล้องวงจรปิดผ่านเครือข่ายอินเตอร์เน็ต port เบอร์ 7003 ใช้รับ control จากเครื่อง client ที่ควบคุมกล้องวงจรปิดผ่านเครือข่ายอินเตอร์เน็ต และ port เบอร์ 7004 ใช้ส่ง control ไปยังเครื่อง client ที่มีกล้องวงจรปิด

4.3 การใช้โปรแกรมเพื่อส่งภาพจากกล้องวงจรปิดและรับ control ซึ่งติดต่อกับเครื่อง server โดยคอมพิวเตอร์ฝั่ง client ที่มีกล้องวงจรปิด

การติดต่อกับเครื่อง server เพื่อส่งภาพจากกล้องวงจรปิดและรอรับ control มีขั้นตอนการใช้งานดังนี้

1. ทำการ run โปรแกรม InterVideoWinDVR , TV Capture หรือโปรแกรมอื่นๆ ที่ติดมากับการ์ด capture เพื่อทำให้ Windows ได้รู้จักและพบ component ของการ์ด capture โดยดับเบิ้ลคลิ๊กที่ไอคอนของโปรแกรมข้างต้น แต่ในการทดลองนี้จะใช้โปรแกรม InterVideoWinDVR ดังรูปที่

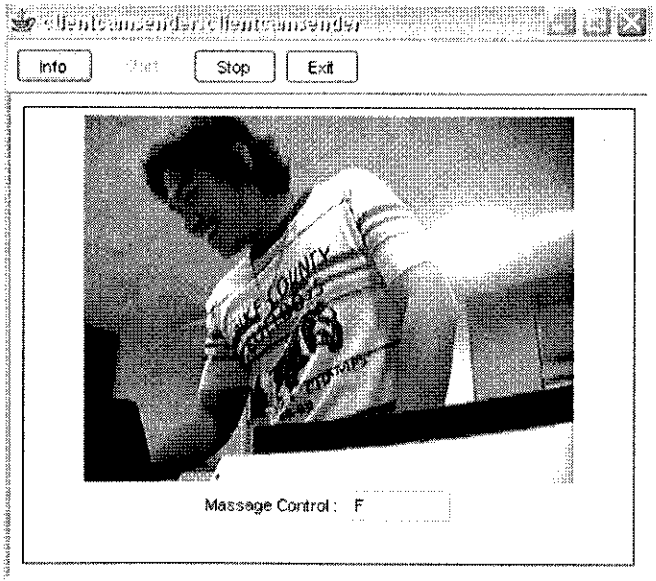
4-5 เมื่อ run โปรแกรมเรียบร้อยแล้ว ปล่อยทิ้งไว้สักครู่หนึ่ง แล้วทำการปิดโปรแกรม InterVideoWinDVR



รูปที่ 4-5 แสดงการ run โปรแกรม InterVideoWinDVR

2. ตั้ง IP Address ของคอมพิวเตอร์เป็น 192.168.0.2 (อาจจะใช้ IP Address อื่นก็ได้ แต่ต้องอยู่ในคลาสเดียวกัน) โดยทำตามขั้นตอนเหมือนกับข้อที่ 1 ในหัวข้อการใช้โปรแกรมเพื่อเริ่มทำงานของคอมพิวเตอร์ฝั่ง server

3. เปิดโปรแกรม JBuilder แล้ว run ไฟล์ชื่อ clientcamsender.jpx โดยทำตามขั้นตอนเหมือนกับข้อที่ 2 ในหัวข้อการใช้โปรแกรมเพื่อเริ่มทำงานของคอมพิวเตอร์ฝั่ง server จะปรากฏหน้าต่างขึ้นดังรูปที่ 4-6



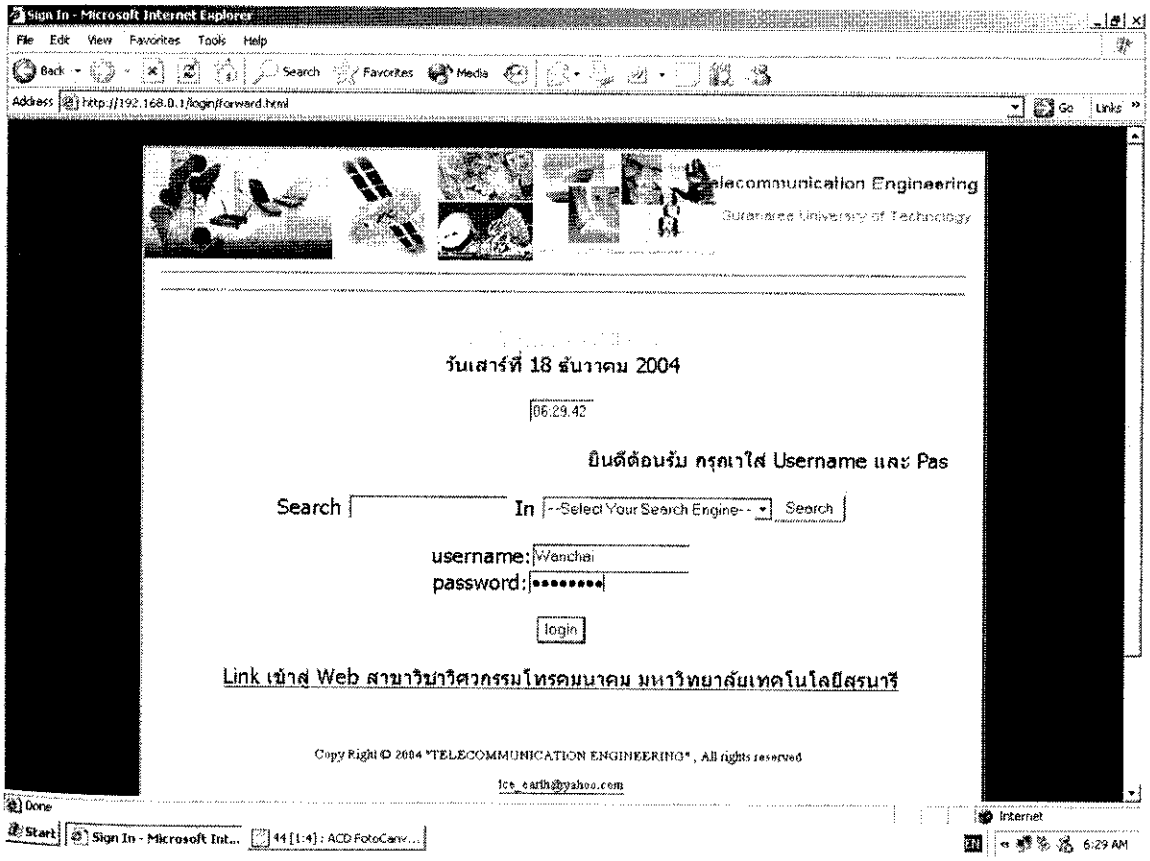
รูปที่ 4-6 แสดงผลการ runไฟล์ชื่อ clientcamsender.jpx

4.4 การใช้โปรแกรมเพื่อรับภาพจากกล้องวงจรปิดและควบคุมการทำงานของกล้องวงจรปิดผ่านระบบอินเตอร์เน็ต

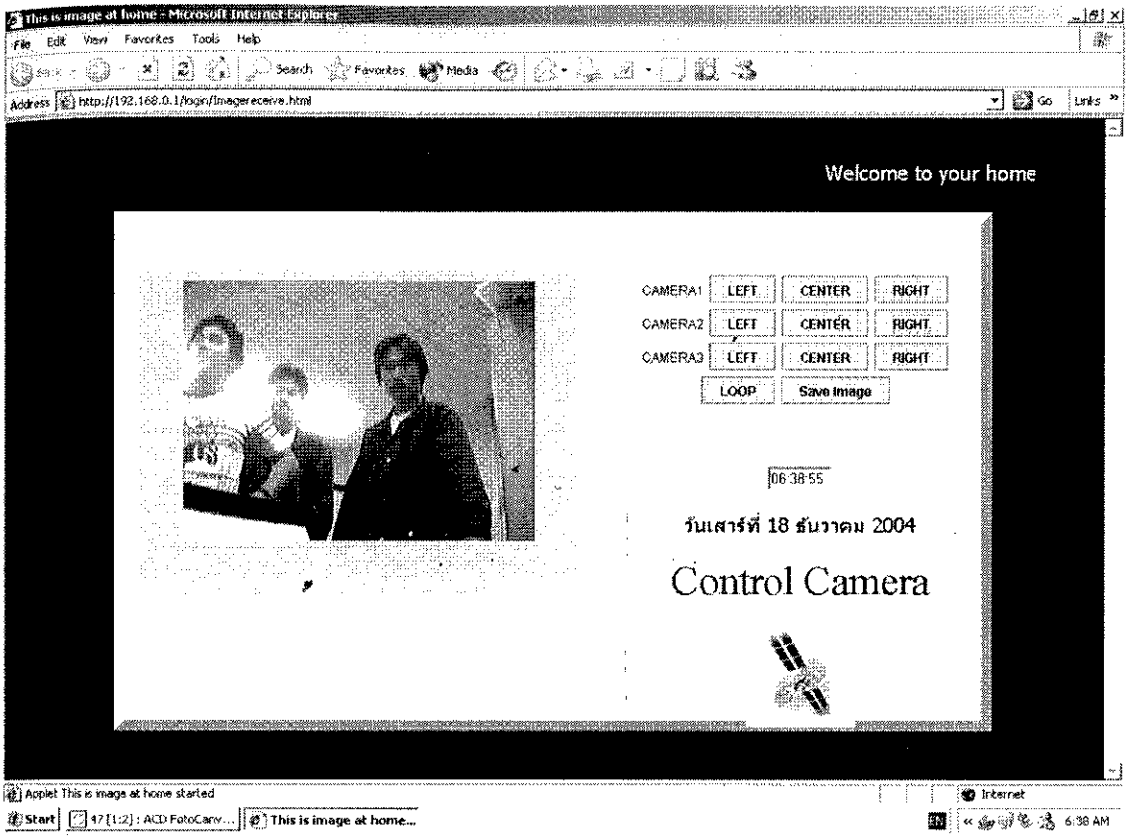
การรับภาพจากกล้องวงจรปิดและควบคุมกล้องวงจรปิดมีขั้นตอนการใช้งานดังนี้

1. ตั้ง IP Address ของคอมพิวเตอร์เป็น 192.168.0.3 (อาจจะใช้ IP Address อื่นก็ได้ แต่ต้องอยู่ในคลาสเดียวกัน) โดยทำตามขั้นตอนเหมือนกับข้อที่ 1 ในหัวข้อการใช้โปรแกรมเพื่อเริ่มทำงานของคอมพิวเตอร์ฝั่ง server

2. เปิดโปรแกรม Internet Explorer จากนั้นพิมพ์ http://192.168.0.1/login/forward.html ในช่อง Address Bar แล้วกด Enter เพื่อเปิด webpage ที่อยู่ภายเครื่อง server จะปรากฏหน้าของ webpage ดังรูปที่ 4-7 (ก) จากนั้นให้ใส่ Username และ Password เพื่อ login เข้าสู่ระบบ แล้วคลิ๊กที่ปุ่ม login ถ้า login ไม่สำเร็จจะปรากฏ webpage ดังรูปที่ ซึ่งอาจจะเกิดจากการใส่ Username หรือ Password ผิด แต่ถ้าทำการ login สำเร็จจะปรากฏหน้า webpage ดังรูปที่ 4-7 (ข) ซึ่งแสดงว่าพร้อมที่จะให้ผู้ใช้ควบคุมกล้องวงจรปิดผ่านระบบอินเตอร์เน็ตตามต้องการ



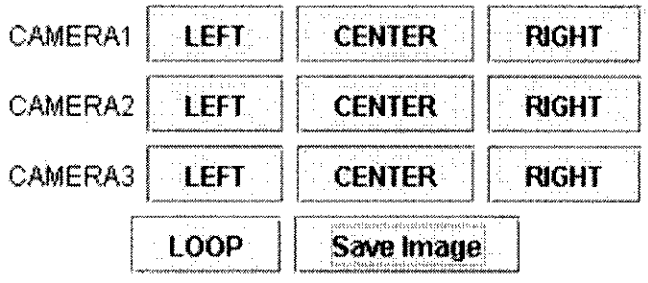
(ก)



(ข)

รูปที่ 4-7 แสดงการเรียก web browser เพื่อดูภาพและสั่งการผ่านอินเตอร์เน็ต

3. ทำการควบคุมกล้องวงจรปิดตามความต้องการ โดยจะมีฟังก์ชั่นต่างๆ ดังรูปที่ 4-8



รูปที่ 4-8 แสดงปุ่มสำหรับควบคุมการทำงานของกล้องวงจรปิด

สำหรับปุ่ม control ต่างๆ มีฟังก์ชั่นการทำงานดังนี้

Camera1 LEFT ให้กล้องวงจรปิดตัวที่ 1 หมุนไปทางซ้าย 30 องศา

Camera1 RIGHT ให้กล้องวงจรปิดตัวที่ 1 หมุนไปทางขวา 30 องศา

Camera1 CENTER ให้กล้องวงจรปิดตัวที่ 1 หมุนกลับมายังจุดศูนย์กลาง

Camera2 LEFT ให้กล้องวงจรปิดตัวที่ 2 หมุนไปทางซ้าย 30 องศา

Camera2 RIGHT ให้กล้องวงจรปิดตัวที่ 2 หมุนไปทางขวา 30 องศา

Camera2 CENTER ให้กล้องวงจรปิดตัวที่ 2 หมุนกลับมายังจุดศูนย์กลาง

Camera3 LEFT ให้กล้องวงจรปิดตัวที่ 3 หมุนไปทางซ้าย 30 องศา

Camera3 RIGHT ให้กล้องวงจรปิดตัวที่ 3 หมุนไปทางขวา 30 องศา

Camera3 CENTER ให้กล้องวงจรปิดตัวที่ 3 หมุนกลับมายังจุดศูนย์กลาง

LOOP ให้แสดงภาพจากกล้องวงจรปิดตัวที่ 1 เป็นเวลา 5 วินาทีแล้วหยุดแสดงภาพ จากนั้นแสดงภาพจากกล้องวงจรปิดตัวที่ 2 เป็นเวลา 5 วินาทีแล้วหยุดแสดงภาพ แล้วแสดงภาพจากกล้องวงจรปิดตัวที่ 3 เป็นเวลา 5 วินาทีแล้วหยุดแสดงภาพ จากนั้นจึงวนกลับไปแสดงภาพจากกล้องวงจรปิดตัวที่ 1 อีก ซึ่งจะแสดงวนเช่นนี้ตลอดเวลา

Save Image จะ save ภาพที่แสดงบน webpage โดยภาพจะถูกจัดเก็บที่โฟลเดอร์ C:\SaveImage และมีไฟล์ภาพชื่อ Image?.jpg ภาพทั้งหมดจะถูกจัดเก็บไว้ภายในคอมพิวเตอร์ที่มีกล้องวงจรปิดอยู่ (เครื่องที่บ้าน)

เมื่อทำการดูภาพและควบคุมกล้องวงจรปิดตามต้องการเรียบร้อยแล้ว ถ้าต้องการจะออกจาก webpage ให้ทำการปิดหน้าต่างของ Internet Explorer สำหรับเครื่องคอมพิวเตอร์ที่มีกล้องวงจรปิดอยู่ (เครื่องที่บ้าน) ถ้าต้องการหยุด run โปรแกรม clientcamsender.jpx ให้ทำตามขั้นตอนข้อที่ 2 ในหัวข้อการใช้โปรแกรมเพื่อเริ่มทำงานของคอมพิวเตอร์ฝั่ง server แล้วถ้า run โปรแกรม clientcamsender.jpx ขึ้นมาใหม่ และผู้ใช้ทำการ save ภาพภาพจะถูก save ทับภาพเดิมที่มีอยู่ ดังนั้นผู้ใช้ควรที่จะย้ายที่เก็บภาพไปไว้ที่โฟลเดอร์อื่น ถ้าผู้ใช้ต้องการที่จะเก็บภาพเดิมเอาไว้ ก่อนที่จะ run โปรแกรมใหม่อีกครั้ง

บทที่ 5

สรุปผลการทดลองและข้อเสนอแนะ

5.1 สิ่งที่ได้จากจากโครงงาน

- ได้รับความรู้ในการใช้เขียนโปรแกรมภาษา Java มากขึ้น ซึ่งไม่เคยศึกษามาก่อนที่จำทำโครงงานนี้ โดยในโครงงานนี้ใช้การเขียนโปรแกรมภาษา Java ในการรับ-ส่งภาพและ control ระหว่างเครื่อง client กับเครื่อง server
- ได้รับความรู้ในการใช้โปรแกรมภาษา Assembly มากขึ้น ซึ่งในโครงงานนี้ใช้ในการควบคุมการทำงานของกล้องวงจรปิด
- ได้รับความรู้ในการใช้งานโปรแกรม 2 โปรแกรมร่วมกัน คือ โปรแกรมภาษา Java และโปรแกรมภาษา Assembly
- ได้รับความรู้เกี่ยวกับการเขียนโปรแกรมภาษา HTML มากขึ้น รวมทั้งสามารถนำภาษา HTML มาใช้งานร่วมกับภาษา Java ได้ ซึ่งเรียกว่า JSP
- ได้รับความรู้เรื่องการรับ-ส่งข้อมูลบนเครือข่ายอินเตอร์เน็ตมากขึ้น
- ได้รับความรู้เกี่ยวกับการทำงานของอุปกรณ์อิเล็กทรอนิกส์ต่างๆ
- ได้รับความรู้ในการประกอบอุปกรณ์ต่างๆ และสามารถเลือกใช้อุปกรณ์ที่เหมาะสมในโครงงาน
- ทำให้สามารถนำความรู้ที่ได้รับจากการเรียนทฤษฎีมาใช้ในการปฏิบัติจริง
- ทำให้รู้จักการทำงานร่วมกับผู้อื่น
- สามารถนำความรู้ที่ได้จากโครงงานมาประยุกต์ใช้งานได้

5.2 ปัญหาและอุปสรรค

- มีความรู้ในการเขียนโปรแกรมภาษา Java น้อย จึงต้องใช้เวลาในการศึกษาและทำความเข้าใจมากรวมทั้งต้องอาศัยผู้ที่มีความเชี่ยวชาญช่วยให้คำแนะนำ ทำให้เสียเวลาในส่วนนี้มาก
- ใช้เวลานานในการศึกษาการใช้โปรแกรมร่วมกันระหว่างภาษา Java และ Assembly
- มีความรู้เรื่องการเขียนโปรแกรมภาษา HTML น้อย ทำให้ต้องใช้เวลาในการศึกษามาก
- มีความรู้ในการเลือกใช้อุปกรณ์น้อย จึงต้องอาศัยผู้ที่มีความเชี่ยวชาญช่วยให้คำแนะนำ
- มีความรู้ในการประกอบวงจรบนแผ่น PCB น้อย ทำให้วงจรมีความเสียหายบ่อย และใช้งานไม่ได้ ทำให้ต้องสร้างวงจรหลายครั้งจึงจะสามารถนำมาใช้งานได้

- มีความรู้เรื่องการรับ-ส่งข้อมูลผ่านระบบอินเตอร์เน็ตน้อย เนื่องจากการเรียนทฤษฎีในวิชา Data Communication ที่เกี่ยวข้องกับเรื่องนี้ได้เรียนตอนปลายเทอม ทำให้มีเวลาในการศึกษาน้อย

5.3 ข้อจำกัดของโครงงาน

- การนำโครงงานนี้มาใช้งานจริง จะต้องมีเช่าเครื่อง server, หรือเป็นเจ้าของเครื่อง server และจดทะเบียน Domain name ของ website ที่จะให้บริการ เนื่องจากการรับ-ส่งภาพและ control ต้องเปิด port หลาย port ซึ่งเครื่อง server ส่วนใหญ่จะไม่เปิดให้บริการ เพราะถ้าเครื่อง server ยิ่งเปิด port มาก จะทำให้ถูก Hack ข้อมูลได้ง่าย
- เครื่องคอมพิวเตอร์ที่บ้านต้องติดต่อกับระบบอินเตอร์เน็ตตลอดเวลาจึงจะใช้งานได้
- การใช้งานกล้องจะต้องมีการ์ด Video Capture ติดตั้งไว้ที่เครื่องคอมพิวเตอร์ที่ต่อกับกล้องวงจรปิดเสมอ
- การติดตั้งกล้องในที่ไกลๆ เช่นนอกอาคาร ทำได้ค่อนข้างลำบาก เนื่องจากกล้องต้องต่อกับ supply , motor และสาย AV เสมอ
- การบันทึกไฟล์ภาพสามารถเก็บไฟล์ภาพได้เฉพาะบนเครื่องคอมพิวเตอร์ที่ต่อกับกล้องวงจรปิด เนื่องจากข้อจำกัดของ Java Applet ที่ไม่สามารถเขียนไฟล์บนเครื่อง client ได้

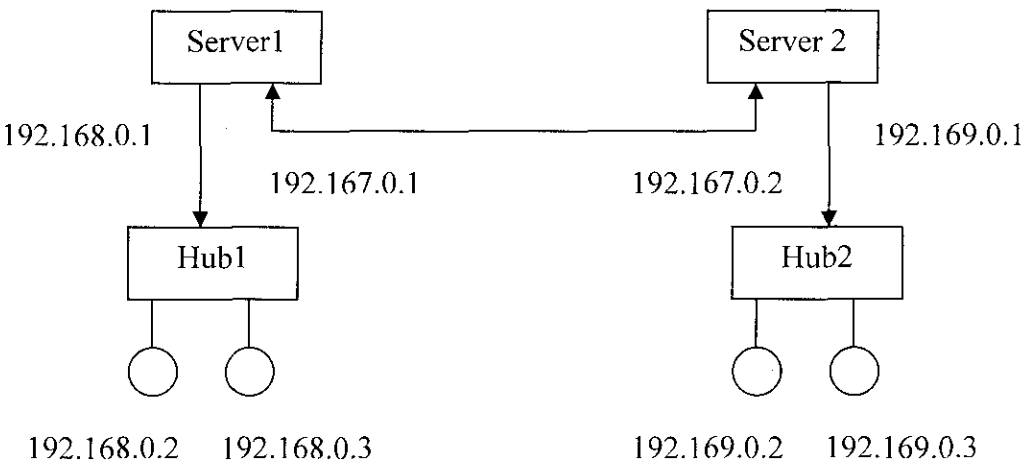
5.4 ข้อเสนอแนะ

โครงงานนี้มีลักษณะงานที่เป็นการควบคุมอุปกรณ์ในระยะไกลผ่านเครือข่ายอินเตอร์เน็ต ทำให้สามารถที่จะนำโครงงานนี้ไปพัฒนาต่อได้อีกหลายด้านดังนี้

- ทำการส่งภาพจากเครื่อง client ไปยังเครื่อง server เป็นแบบ Stream โดยใช้ RTP (Real-time Transfer Protocol) ซึ่งจะทำให้เครื่อง client ที่ต้องการดูภาพผ่านระบบอินเตอร์เน็ตได้รับภาพที่เป็น real-time และต่อเนื่องมากขึ้น เนื่องจากโครงงานนี้มีการส่งภาพจากเครื่อง client ที่มีกล้องวงจรปิดไปยังเครื่อง server จากนั้นเครื่อง client ที่ต้องการดูภาพผ่านระบบอินเตอร์เน็ตจะนำภาพจากเครื่อง server มาแสดงบนหน้าจอคอมพิวเตอร์ของเครื่อง client จะต้องใช้ระยะเวลาในการส่งภาพพอสมควร รวมทั้งโครงงานนี้ใช้การส่งภาพแบบส่งเป็น frame ของข้อมูล ทำให้ภาพที่ทางฝั่ง client ได้รับมีการ delay และเกิดการกระพริบ
- เพิ่มเติมในส่วนของคุณสมบัติของการใช้งานให้แก่เครื่อง client ที่ต้องการดูภาพและสั่งการผ่านระบบอินเตอร์เน็ต เพื่อที่ผู้ใช้งานจะได้รับความสะดวกและประโยชน์สูงสุดในการใช้งาน เช่น เพิ่มระบบบันทึกภาพอัตโนมัติทุกกี่วินาที สามารถบันทึกไฟล์ภาพที่เครื่อง client ที่ดูภาพ

และสั่งการผ่านระบบอินเตอร์เน็ต มีการรายงานสถานะการใช้งาน มีการแสดงภาพได้ทีละหลายๆ กล้องพร้อมๆ กัน และสามารถบันทึกภาพเป็นไฟล์ Video ได้ เป็นต้น

- พัฒนาในส่วนของอุปกรณ์ควบคุมกล้องให้สามารถที่จะติดตั้งได้สะดวกและง่ายขึ้น เช่น ใช้กล้องวงจรปิดไร้สายแทนแบบมีสาย เป็นต้น
- การทดลองในสภาวะที่เครื่อง server และเครื่อง client ทั้งสองเครื่องอยู่ต่าง subnet กัน ทำโดยเพิ่มเครือข่ายขึ้นจากเดิมที่มีเพียงเครือข่ายเดียว ให้เป็นสองเครือข่าย แล้วทำการเชื่อมต่อสองเครือข่ายเข้าด้วยกัน โดยเครื่อง server ของแต่ละเครือข่ายต้องมี Card LAN 2 ตัว ซึ่งนำ Card LAN 1 ตัวของแต่ละเครือข่ายเชื่อมต่อกันผ่านสาย UTP ซึ่งเครือข่ายแรกจะตั้งIP Address ขึ้นเป็น 192.167.0.1 ส่วนอีกเครือข่ายหนึ่งจะตั้ง IP Address เป็น 192.167.0.2 แล้วนำ Card LAN อีก 1 ตัวของแต่ละเครือข่ายจะต่อเข้ากับ Hub จะทำการตั้ง IP Address ของ Card LAN ของเครือข่ายหนึ่งเป็น 192.168.0.1 ส่วนเครื่องลูกข่ายที่ต่อกับ Hub ที่ออกจาก Card LAN ตัวนี้จะใช้ IP Address เป็น 192.168.0.2, 192.168.0.3, ... การติดตั้ง IP Address ของเครือข่ายที่สอง จะทำเหมือนกับการติดตั้ง IP Address ของเครือข่ายแรก โดยจะใช้ IP Address 192.169.0.1, 192.169.0.2, ... สามารถแสดงเป็นรูปที่ 5-1



รูปที่ 5-1 แสดงการเชื่อมต่อระหว่างเครื่อง server กับเครื่อง client ที่อยู่ต่าง subnet กัน

- การออกแบบเพื่อพัฒนาโครงงานนี้จะต้องมีการใช้ port สำหรับรับ-ส่งภาพและ control ให้น้อยลง และสามารถรองรับผู้ใช้ได้มากขึ้นภายใต้ port ที่สร้างขึ้น โดยการเขียนโปรแกรมเพิ่มในส่วนของเครื่อง server ซึ่งจะเช็คว่าเครื่องที่เชื่อมต่อมาที่เครื่อง server เป็นใคร ขอรับบริการอะไรจากเครื่อง server เช่น รับภาพหรือส่งภาพ รับ control หรือส่ง control เป็นต้น เมื่อเช็คได้แล้วว่าเครื่อง client ต้องการบริการอะไรจากเครื่อง server จากนั้นเครื่อง server จึง

ให้บริการนั้นๆ แก่เครื่อง client ซึ่งวิธีนี้จะทำให้ port ที่เครื่อง server จะให้บริการแก่เครื่อง client ลดลง

- สามารถที่จะนำหลักการทำงานของโครงงานนี้มาสร้างโครงงานหรือพัฒนาโครงงานอื่นๆ ได้ เช่น สามารถนำหลักการรับ-ส่งข้อมูลผ่านระบบอินเตอร์เน็ตมาใช้ในการสั่งการอุปกรณ์ไฟฟ้าผ่านเครือข่ายอินเตอร์เน็ต การควบคุมเซ็นเซอร์การเปิดปิดวาล์วน้ำผ่านอินเตอร์เน็ต และสั่งการหุ่นยนต์ผ่านระบบอินเตอร์เน็ต เป็นต้น

บรรณานุกรม

[1] กิตติ ภักดีวัฒนะกุล. (2546). คัมภีร์ JAVA เล่ม 1. พิมพ์ครั้งที่2. กรุงเทพฯ: เคทีพี คอมพ์ แอนด์ คอนซัลท์

[2] เจนวิทย์ เหลืองอร่าม. (2539). การเขียนโปรแกรมสำหรับ Applications และ Applets ด้วย JAVA. กรุงเทพฯ: ธรรมสาร

[3] บุญสืบ โพธิ์ศรี, สมหมาย ปานเขียว และโกมล ศิริสมบูรณ์เวช. (2546). ไฟฟ้าและอิเล็กทรอนิกส์เบื้องต้น. กรุงเทพฯ: ศูนย์ส่งเสริมอาชีวะ

[4] วรพจน์ กรแก้ววัฒนกุล และชัยวัฒน์ ลิ้มพรจิตรวิไล. (n.d.). เรียนรู้และปฏิบัติการไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช. กรุงเทพฯ: อินโนเวตีฟ เอ็กเพอริเมนต์

[5] วรรณิกา เนตรงาม. (2545). คู่มือการเขียนโปรแกรมภาษา JAVA ฉบับผู้เริ่มต้น. กรุงเทพฯ: เอช เอ็น กรุ๊ป

[6] วีระศักดิ์ ซึงถาวร. (2547). Java Programming Volume III. กรุงเทพฯ: ซีเอ็ดยูเคชั่น

[7] สาธิต ชัยวิวัฒน์ตระกูล. (2545). เก่ง JSP ให้ครบสูตร. กรุงเทพฯ: H.N. Group

[8] สุวัฒน์ ปุณณชัยยะ, ต้น ตัณฑ์สุทธิวงศ์ และสุพจน์ ปุณณชัยยะ. (2545). เปิดโลก TCP/IP และโปรโตคอลของอินเตอร์เน็ต. พิมพ์ครั้งที่2. กรุงเทพฯ: โปรวิชั่น

[9] A.M. Muhammad. (2000). THE 8051 MICROCONTROLLER AND EMBEDDED SYSTEMS. United States of America: Prentice-Hall

[10] S.T. Andrew. (2004). Computer Network. Bangkok: Pearson Education Indochaina

ภาคผนวก

ภาคผนวก ก

คลาสต่างๆ ของ Java API Package

java.lang

ชื่อคลาส (Class Name)	คำอธิบาย
Boolean	ตัวห่อหุ้มคลาสสำหรับชนิดข้อมูล boolean ดั้งเดิม
Character	ตัวห่อหุ้มคลาสสำหรับชนิดข้อมูลอักขระ ดั้งเดิม
Class	บรรจุช่วงเวลารันโปรแกรม เป็นตัวแทนของข้อมูลชนิด class ทั้งหมด
ClassLoader	คลาสประเภท abstract ที่ระบุวิธีโหลดคลาสในช่วงเวลารัน
Compile	ช่วยให้เข้าถึงคอมไพเลอร์ของ Java
Double	ตัวห่อหุ้มคลาสสำหรับชนิดข้อมูล double ดั้งเดิม
Float	ตัวห่อหุ้มคลาสสำหรับชนิดข้อมูล float ดั้งเดิม
Interger	ตัวห่อหุ้มคลาสสำหรับชนิดข้อมูล interger ดั้งเดิม
Long	ตัวห่อหุ้มคลาสสำหรับชนิดข้อมูล long ดั้งเดิม
Math	libraries ของฟังก์ชั่นมาตรฐานทางคณิตศาสตร์
Number	superclass ประเภท abstract ให้ชนิดข้อมูลเป็นจำนวนเต็ม และจำนวนทศนิยมทั้งหมด
Object	superclass ของคลาส java ทั้งหมด และช่วยเหลือ method ต่างๆ ที่สืบทอดโดยคลาสต่างๆ ของ java ทั้งหมด
Process	libraries ของกระบวนการควบคุม method ต่างๆ
Runtime	libraries ของช่วงเวลารันในการเข้าถึง method ต่างๆ
SecurityManager	คลาสประเภท abstract ที่บรรจุเทมเพลต (template) ต่างๆ ของ method สำหรับนโยบายรักษาความปลอดภัย
String	superclass หรือคลาสแม่สำหรับออบเจ็กต์ต่างๆ ของสตริงทั้งหมด
StringBuffer	superclass หรือคลาสแม่สำหรับออบเจ็กต์ต่างๆ ของสตริงทั้งหมดที่มีขนาดใหญ่
System	ช่วยให้เข้าถึงระบบอย่างอิสระ เพื่อให้ได้ใช้ทรัพยากรของระบบที่สำคัญ เช่น stdin, stdout, และ stderr
Thread	Superclass สำหรับคลาสและ method ต่างๆ ของ thread ทั้งหมด ช่วยให้โปรแกรมสามารถทำงานได้หลายอย่างในคราวเดียวกัน

ThreadGroup	Superclass หรือคลาสแม่สำหรับจัดกลุ่มของ thread เชิงซ้อน (multiple thread) ต่างๆ เข้าด้วยกัน
Throwable	Superclass หรือคลาสแม่สำหรับออบเจ็กต์และ method ต่างๆ เชื่อมโยงกับ exception ต่างๆ และข้อผิดพลาดของ java ทั้งหมด

java.util

ชื่อคลาส (Class Name)	คำอธิบาย
BitSet	libraries สำหรับจัดบิต (bit)
Date	libraries สำหรับจัดการเกี่ยวกับ วันและเวลา
Dictionary	ใช้สร้างออบเจ็กต์ต่างๆ เป็น container ของข้อมูล ซึ่งช่วยให้คีย์ต่างๆ ที่เกี่ยวข้องสามารถเข้าถึงค่าต่างๆ ของข้อมูล
Hashtable	subclass ของคลาส Dictionary ที่อนุญาตให้ออบเจ็กต์ต่างๆ ที่รวบรวมไว้ สามารถเข้าถึงโดยค่าของโค้ด hash
Observable	ช่วยออบเจ็กต์ต่างๆ ที่ถูกสร้างขึ้นเป็นตัวเฝ้าดูออบเจ็กต์ต่างๆ ที่ได้รับการปรับปรุง และได้ให้การสนับสนุนอินเตอร์เฟซ Observer
Properties	subclass ของคลาส Hashtable ที่สามารถบันทึก หรือโหลด สตรีม (stream)
Random	ช่วยผลิตเลขสุ่ม (random number)
Stack	ช่วยสร้างสแต็ก (stack) ของออบเจ็กต์ต่างๆ
StringTokenizer	ช่วยผ่านออบเจ็กต์ของคลาส String เข้าไปในกลุ่มของ token
Vector	ช่วยเหลืออะเรย์ขนาดใหญ่

java.io

ชื่อคลาส (Class Name)	คำอธิบาย
BufferedInputStream	สนับสนุนความสามารถในการสร้างบัฟเฟอร์ (buffer) สำหรับออบเจ็กต์ของ InputStream ช่วยอ่านได้เร็วขึ้น
BufferedOutputStream	สนับสนุนความสามารถในการสร้างบัฟเฟอร์ (buffer) สำหรับออบเจ็กต์ของ OutputStream ช่วยเขียนได้เร็วขึ้น
ByteArrayInputStream	ช่วยแปลงอะเรย์ของไบต์ (byte) เข้าไปในออบเจ็กต์ของคลาส InputStream
ByteArrayOutputStream	ช่วยแปลงอะเรย์ของไบต์ (byte) เข้าไปในออบเจ็กต์ของคลาส

	OutputStream
DataInputStream	สนับสนุนความสามารถในการอ่านชนิดข้อมูลดั้งเดิมและออบเจ็กต์ต่างๆ จากออบเจ็กต์ของคลาส InputStream
DataOutputStream	สนับสนุนความสามารถในการเขียนชนิดข้อมูลดั้งเดิมและออบเจ็กต์ต่างๆ จากออบเจ็กต์ของคลาส OutputStream
File	สนับสนุนการเข้าถึงไฟล์ที่เป็นอิสระต่อระบบ หรือไดแรกทอรีบนระบบของ host
FileDescriptor	สตรีมอ่านจากตัวอธิบายของไฟล์
FileInputStream	อนุญาตออบเจ็กต์ของคลาส File ใช้เป็นพื้นฐานสำหรับสร้างออบเจ็กต์ของคลาส InputStream
FileOutputStream	อนุญาตออบเจ็กต์ของคลาส File ใช้เป็นพื้นฐานสำหรับสร้างออบเจ็กต์ของคลาส OutputStream
FilterInputStream	superclass ของคลาสทั้งหมดที่ช่วยเหลือฟิลเตอร์ (filter) สำหรับสตรีมของอินพุต
FilterOutputStream	superclass ของคลาสทั้งหมดที่ช่วยเหลือฟิลเตอร์ (filter) สำหรับสตรีมของเอาท์พุต
InputStream	superclass ของคลาสต่างๆ ของสตรีมสำหรับอินพุตทั้งหมด มันสนับสนุน method ต่างๆ ที่ต้องการสตรีมของอินพุตเป็นไบต์
LineNumberInputStream	ใช้เฝ้าดูหมายเลขบรรทัดที่เกี่ยวข้องกับออบเจ็กต์ของคลาส InputStream
OutputStream	superclass ของคลาสต่างๆ ของสตรีมสำหรับเอาท์พุต มันสนับสนุน method ต่างๆ ที่ต้องการสตรีมของเอาท์พุตเป็นไบต์
PipedInputStream	ให้การสนับสนุนสตรีมของอินพุตต่อ thread จนกระทั่งมันสามารถอ่านข้อมูลที่เขียนให้กับออบเจ็กต์ของคลาส PipedOutputStream โดย thread อื่น
PipedOutputStream	สตรีมสำหรับเขียนข้อมูลให้กระบวนการอื่น
PointStream	สตรีมของเอาท์พุตสำหรับข้อมูลตามรูปแบบที่กำหนดไว้ไปออกอุปกรณ์เอาท์พุต
PushbackInputStream	สตรีมของอินพุตที่อนุญาตให้ไบต์เดียวถูกนำกลับไปบนสตรีมจนกระทั่งสามารถอ่านมันได้อีกครั้ง

RandomAccessFile	สตรีมของอินพุตที่อนุญาตเข้าถึงไฟล์โดยการสุ่ม
SequenceInputStream	สตรีมของอินพุตที่อนุญาตสตรีมของอินพุตมากกว่าหนึ่งที่จะอ่านแบบเรียงลำดับ
StreamTokenizer	method ต่างๆ สำหรับแปลงสตรีมของอินพุตให้เป็น token ต่างๆ
StringBufferInputStream	สตรีมอ่านจากออบเจ็กต์ของคลาส StringBuffer

java.net

ชื่อคลาส (Class Name)	คำอธิบาย
ContentHandler	คลาสสำหรับสร้างออปเจ็คจาก URLs
DatagramPacket	ห่อหุ้มออปเจ็ค datagram ที่อ่านหรือเขียนจาก socket ของ UDP
InetAddress	ดำเนินการเกี่ยวกับ host ของอินเตอร์เน็ต และแอดเดรสของ IP (Internet Protocol)
ServerSocket	ช่วยสร้าง socket ของ TCP ที่สามารถถูกใช้ให้สนับสนุนแอพพลิเคชันของคอมพิวเตอร์แม่ข่าย (server)
Socket	ให้การสนับสนุน socket ที่ใช้โดยโปรแกรมในคอมพิวเตอร์ลูกข่าย (client)
SocketImpl	Superclass ประเภท abstract สำหรับคลาสต่างๆของ socket ทั้งหมด
URL	ห่อหุ้ม URLs และให้การสนับสนุนของกลุ่ม method ต่างๆสำหรับเข้าถึงทรัพยากรของเครือข่ายที่อ้างถึงโดย URL
URLConnection	จัดการเกี่ยวกับการเชื่อมต่อของ HTTP ที่สร้างด้วยทรัพยากรตามที่ URL ระบุ
URLEncoder	ถูกใช้ให้ลงโค้ดสารสนเทศในรูปแบบสำหรับเชื่อมต่อทาง URL
URLStreamHandler	ให้การสนับสนุน URLConnection สำหรับชนิดโปรโตคอล (protocol) ต่างกัน

java.awt

ชื่อคลาส (Class Name)	คำอธิบาย
BorderLayout	lay out ออบเจ็กต์ต่างๆ ของ GUI ที่บรรจุภายในออบเจ็กต์ของคลาส Container มันช่วยจัด (layout) ออบเจ็กต์ต่างๆ ของคลาส Window, Frame, และ Dialog

Button	method ต่างๆ สำหรับดำเนินการเกี่ยวกับออบเจ็กต์ต่างๆ ของคลาส Button
Canvas	ให้การสนับสนุนออบเจ็กต์ของ GUI ที่ช่วยเหลือการเขียนภาพ การเขียนภาพจะไม่ได้รับการสนับสนุนบน canvas ของมันเอง แต่อยู่บนออบเจ็กต์ของคลาส Graphics ที่จัดให้โดย canvas
CardLayout	layout ออบเจ็กต์ต่างๆ ใน container ในลักษณะคล้ายกับสำรับไพ่
Checkbox	ให้การสนับสนุน controls ของ GUI สำหรับ checkbox และ ปุ่มวิทยุ (radio button)
CheckboxGroup	ใช้ร่วมกับคลาส Checkbox เพื่อสนับสนุนปุ่มวิทยุต่างๆ
CheckboxMenuItem	ให้การสนับสนุนตัวเลือก หรือรายการต่างๆ ของเมนูที่สามารถตรวจสอบการเลือกได้
Choice	ให้การสนับสนุนตัวเลือก หรือรายการต่างๆ แบบ pulldown ที่สามารถจัดลงในพื้นที่หลักของวินโดว์
Color	ให้การสนับสนุนสีที่เป็นอิสระต่อระบบ และได้กำหนดค่าคงที่ของสีไว้มากมาย
Component	superclass ของ controls ของ GUI ทั้งหมด มันช่วยเตรียมกลุ่มของ method ต่างๆ ที่ให้การสนับสนุน การจัดส่วนประกอบต่างๆ (component) การแสดง และการเชื่อมโยงเหตุการณ์ต่างๆ
Container	superclass ของคลาสต่างๆ ของวินโดว์ที่บรรจุออบเจ็กต์ต่างๆ และได้จัดทำ method ต่างๆ เพื่อช่วยจัดส่วนประกอบต่างๆ และการแสดงออบเจ็กต์ต่างๆ ที่บรรจุอยู่
Dialog	ให้การสนับสนุนวินโดว์ของไดอะล็อกบ๊อกซ์ (dialog box)
Dimension	ใช้แทนความกว้างและความสูงของออบเจ็กต์ขนาด 2 มิติ
Event	คลาส Even ให้ห่อหุ้มเหตุการณ์ต่างๆ ไว้ทั้งหมดที่จะดำเนินการโดยโปรแกรมวินโดว์ของ Java
FileDialog	ใช้สร้างไดอะล็อกบ๊อกซ์ที่ช่วยเหลือเลือกไฟล์ต่างๆ สำหรับดำเนินการอินพุต และเอาท์พุต
FlowLayout	layout ออบเจ็กต์ต่างๆ ของคลาส Container ของวินโดว์ มันยังช่วย Layout โดยอัตโนมัติที่ใช้กับคลาส Panel
Font	ให้การสนับสนุนกลุ่มของฟอนต์ (font) ต่างๆ ที่เป็นอิสระต่อระบบ ซึ่ง

	ควบคุมการแสดงเท็กซ์
FontMetrics	ช่วยให้เข้าถึงการแสดงคุณลักษณะของคลาส Font โดยเฉพาะ
Frame	ใช้สร้างและควบคุมวินโดว์หลักของโปรแกรมวินโดว์ของ Java ที่ใช้โดดเดี่ยว
Graphics	ช่วยเขียนออบเจ็กต์ต่างๆ ของกราฟฟิก และเท็กซ์ภายในวินโดว์
GridBagConstraints	ใช้จำแนกตำแหน่งของพารามิเตอร์ของส่วนประกอบต่างๆ (component) ที่บรรจุอยู่ภายในออบเจ็กต์ที่ layout
GridBagLayout	อนุญาตออบเจ็กต์ของคลาส Container เพื่อ layout ออบเจ็กต์ต่างๆ ของ component ที่มากกว่า 1 แถว 1 คอลัมน์ ตามสไตล์ของกริด (grid)
GridLayout	ใช้ lay out ออบเจ็กต์ของคลาส Container ในลักษณะของกริด (grid) ซึ่งสมาชิกต่างๆ ของกริด (grid) ทั้งหมดมีขนาดเดียวกัน
Image	กลไกที่เป็นอิสระต่อ content ไว้ให้สนับสนุนภาพกราฟฟิก
Insets	ใช้ระบุขอบ (border) ที่ต้องการ ซึ่งเกี่ยวข้องกับออบเจ็กต์ของ GUI
Label	ใช้แสดงฉลาก (label) ต่างๆ ของเท็กซ์ภายในวินโดว์หรือ container ของGUI อื่นๆ
List	ให้การสนับสนุน controls ของ GUI สำหรับตัวเลือกที่มีหนึ่งเดียวหรือมากกว่าหนึ่ง
MediaTracker	เตรียมกลุ่ม method ต่างๆ ไว้ให้สำหรับจัดการภาพต่างๆ ที่ใช้สนับสนุนออบเจ็กต์ต่างๆ ของมัลติมีเดีย
Menu	ให้การสนับสนุนเมนูแบบ pull-down เดียวซึ่งติดเข้ากับเมนูบาร์ (menu bar) หรือเมนูอื่น
MenuBar	ให้การสนับสนุนเมนูบาร์ (menu bar) ซึ่งติดเข้ากับออบเจ็กต์ของ Frame ของโปรแกรมวินโดว์
MenuComponent	Superclass ของคลาสต่างๆ ที่เกี่ยวข้องกับเมนูทั้งหมดและช่วยเตรียมกลุ่มของ method ต่าง ๆ ไว้ใช้กับ subclass ของมัน
MenuItem	ให้การสนับสนุนตัวเลือก หรือคำสั่งต่าง ๆ ที่สามารถเลือกจากเมนูแบบ pull-down มันจะสืบทอดไปยังคลาส Menu และ CheckboxMenuItem
Panel	ใช้ในลักษณะ container เพื่อจัดส่วนประกอบต่าง ๆ ของ GUI ภายใน

	วินโดว์ มันจะกำหนด layout ของมันเป็นแบบ FlowLayout ให้โดยอัตโนมัติ
Point	ใช้แทนโดยทั่วไป สำหรับพิกัด x, y ขนาด 2 มิติ
Polygon	ใช้แทนรูปหลายเหลี่ยมตามรายการของพิกัด x, y ซึ่งจำแนกจุดต่าง ๆ ตรงมุมของรูปหลายเหลี่ยม
Rectangle	ใช้แทนรูปสี่เหลี่ยมมุมฉาก ที่ใช้พิกัด x, y ของมุมบนซ้ายสุด และความกว้าง และความสูงของมัน
Scrollbar	ใช้สนับสนุน scrollbar สำหรับเลื่อนจอภาพในแนวระดับและแนวดิ่ง
TextArea	Method ต่าง ๆ สำหรับดำเนินการกับพื้นที่เท็กซ์ (textarea)
TextComponent	Superclass ของออบเจ็กต์ต่าง ๆ สำหรับเท็กซ์ทั้งหมด
TextField	Method ต่าง ๆ สำหรับดำเนินการกับบรรทัดเดียวของเท็กซ์
Toolkit	ช่วยเชื่อมโยงระหว่าง AWAT ธรรมดาที่ Java ให้ความช่วยเหลือ และ AWAT ที่ขึ้นกับแพล็ตฟอร์ม (platform)
Window	Superclass ของคลาสต่าง ๆ ที่เกี่ยวข้องกับวินโดว์ทั้งหมดและช่วยเตรียมการกลุ่มของ method ต่าง ๆ สำหรับจัดวินโดว์และแสดงวินโดว์ต่าง ๆ

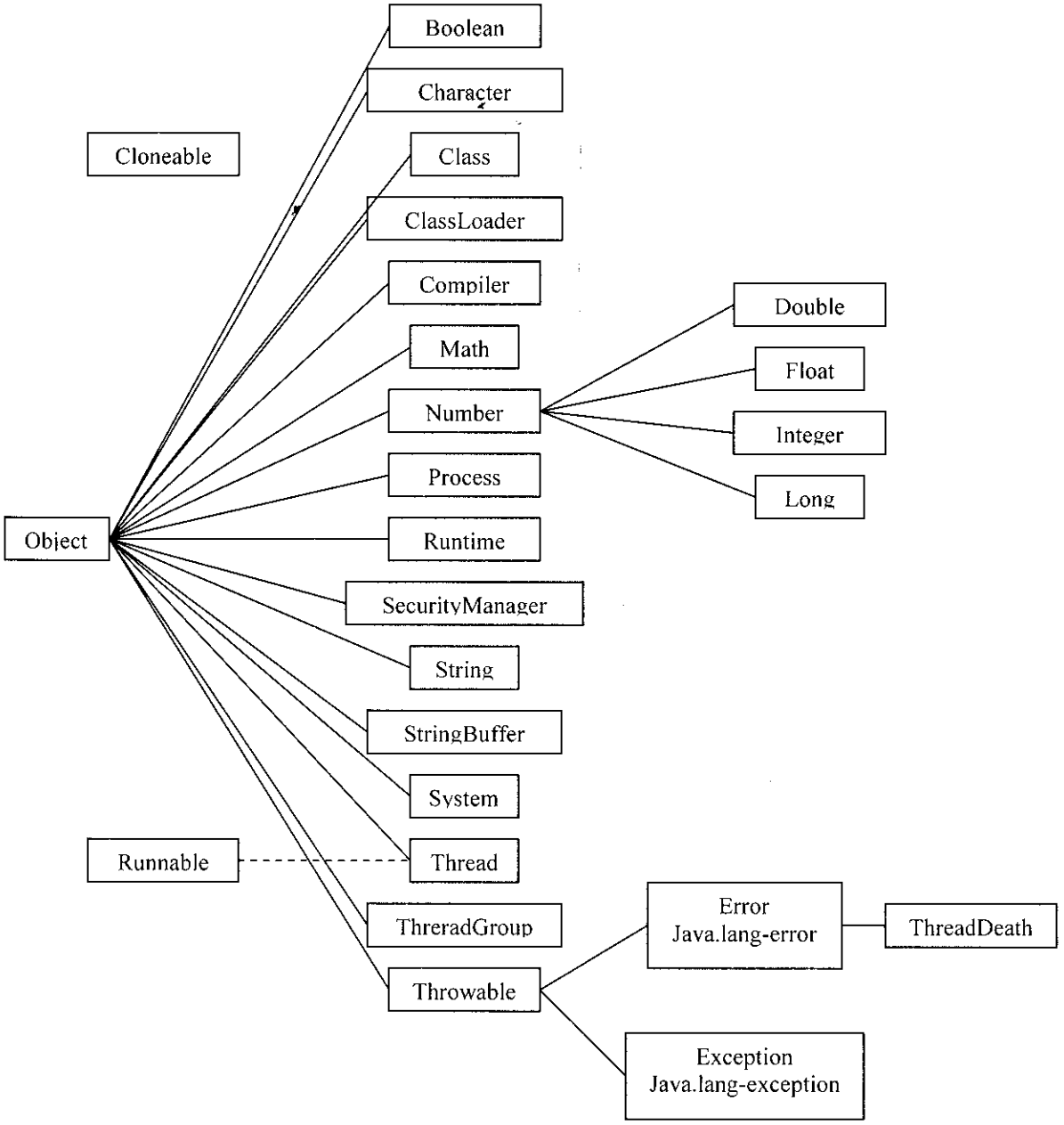
java.awt.image

ชื่อคลาส (Class Name)	คำอธิบาย
ColorModel	คลาสประเภท abstract ช่วยแปลงระหว่างโมเดล (model) ของสี
CropImageFilter	Method ต่าง ๆ สำหรับใช้ crop ภาพต่าง ๆ ให้พื้นที่ที่ต้องการ
DirectColorModel	ใช้ให้เข้าถึงค่าต่าง ๆ ของสีของจุด (pixel) โดยตรง
FilteredImageSource	ให้การสนับสนุนความสามารถในการฟิลเตอร์ (filter) ภาพที่ใช้ออบเจ็กต์ของคลาส ImageFilter
ImageFilter	ให้การสนับสนุนกลุ่มของ method ต่าง ๆ สำหรับให้การสนับสนุนฟิลเตอร์ (filter) ของภาพ
IndexColorModel	subclass ของคลาส ColorModel ที่ทำการแปลค่าของจุด (pixel) ของ colormap ไปเป็นสีต่าง ๆ ของส่วนประกอบของ RGB
MemoryImageSource	ช่วยสร้างภาพที่ใช้อะเรย์ของค่าต่าง ๆ ของจุด (pixel)
PixelGrabber	ช่วยจับจุด (pixel) ต่าง ๆ ของภาพ และบรรจุมันไว้ในอะเรย์

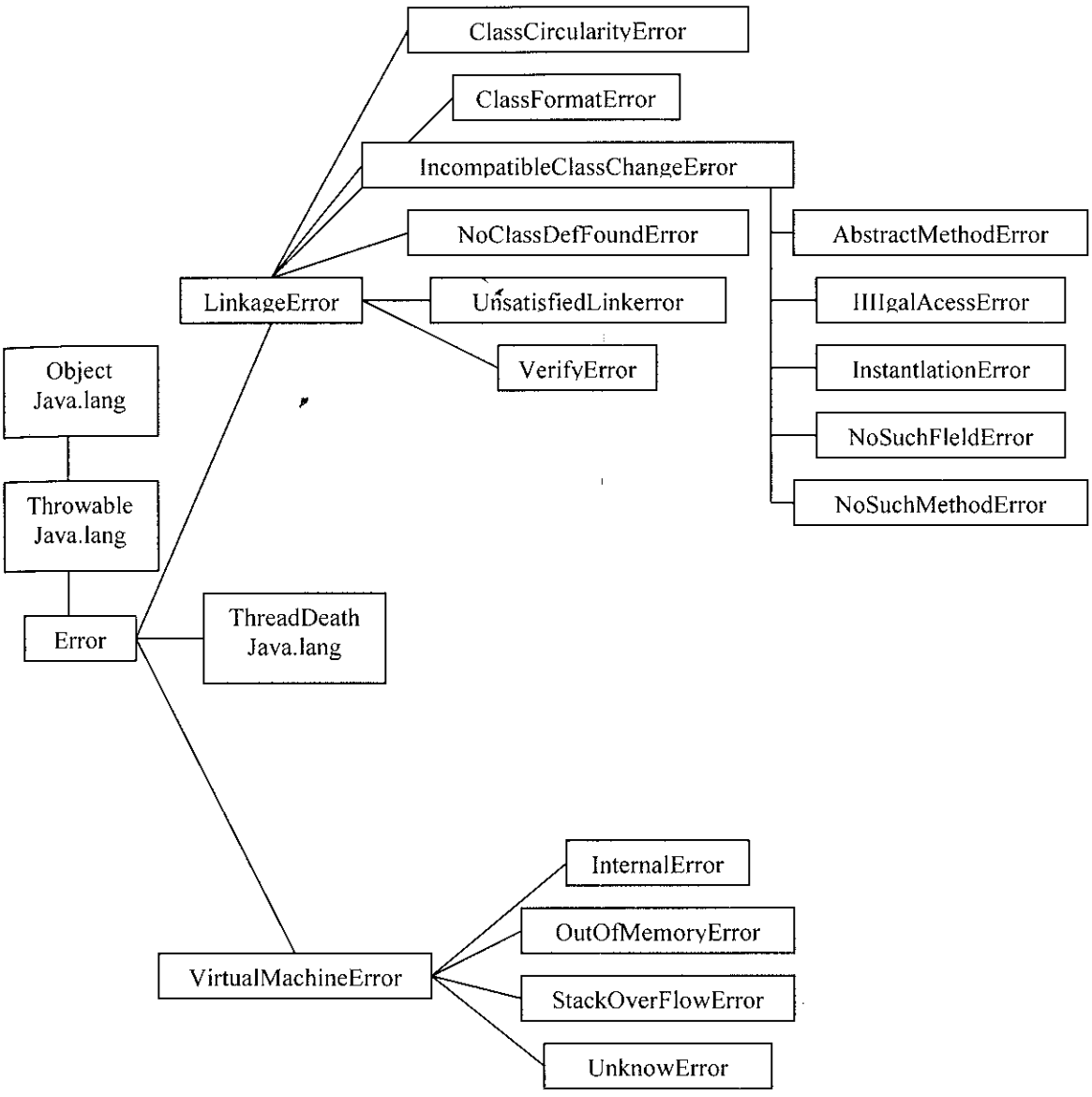
RGBImageFilter	ช่วยสร้างฟิลเตอร์ (filter) ของภาพที่ปรับปรุงจุด (pixel) ต่าง ๆ ของโมเดลของสีของ RGB โดยอัตโนมัติ

โครงสร้างของ Java API Package

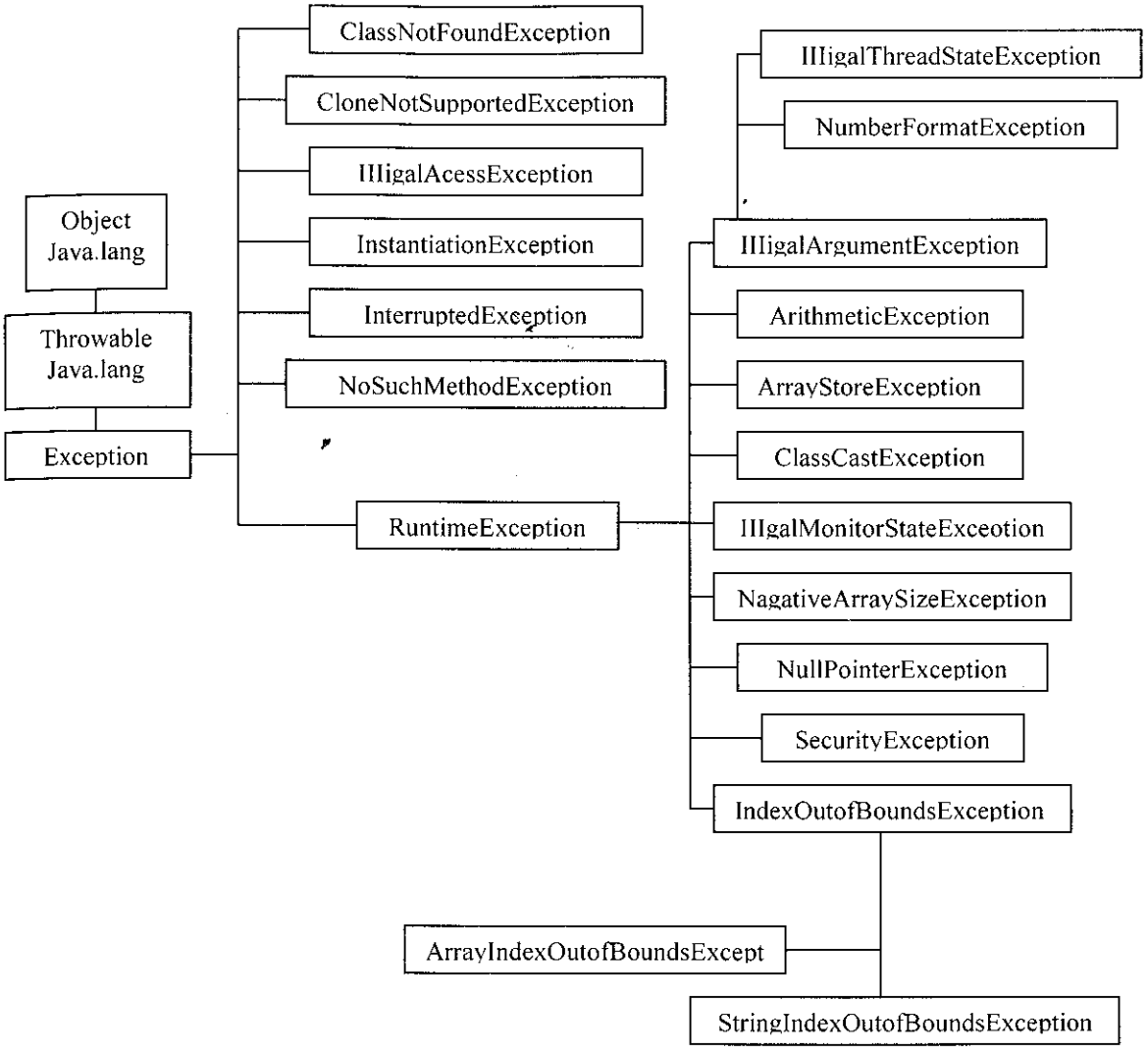
java.lang



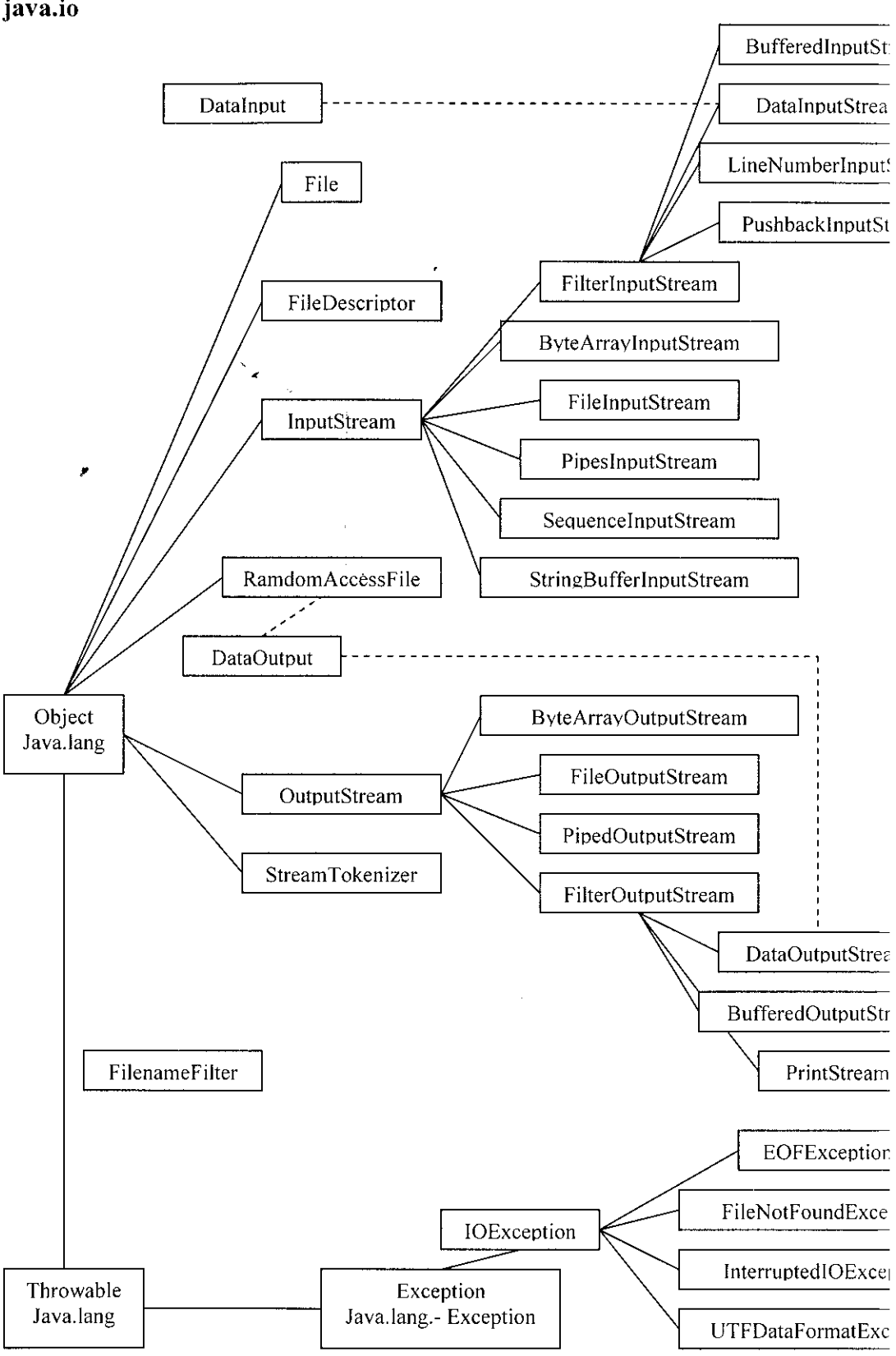
java.lang-errors



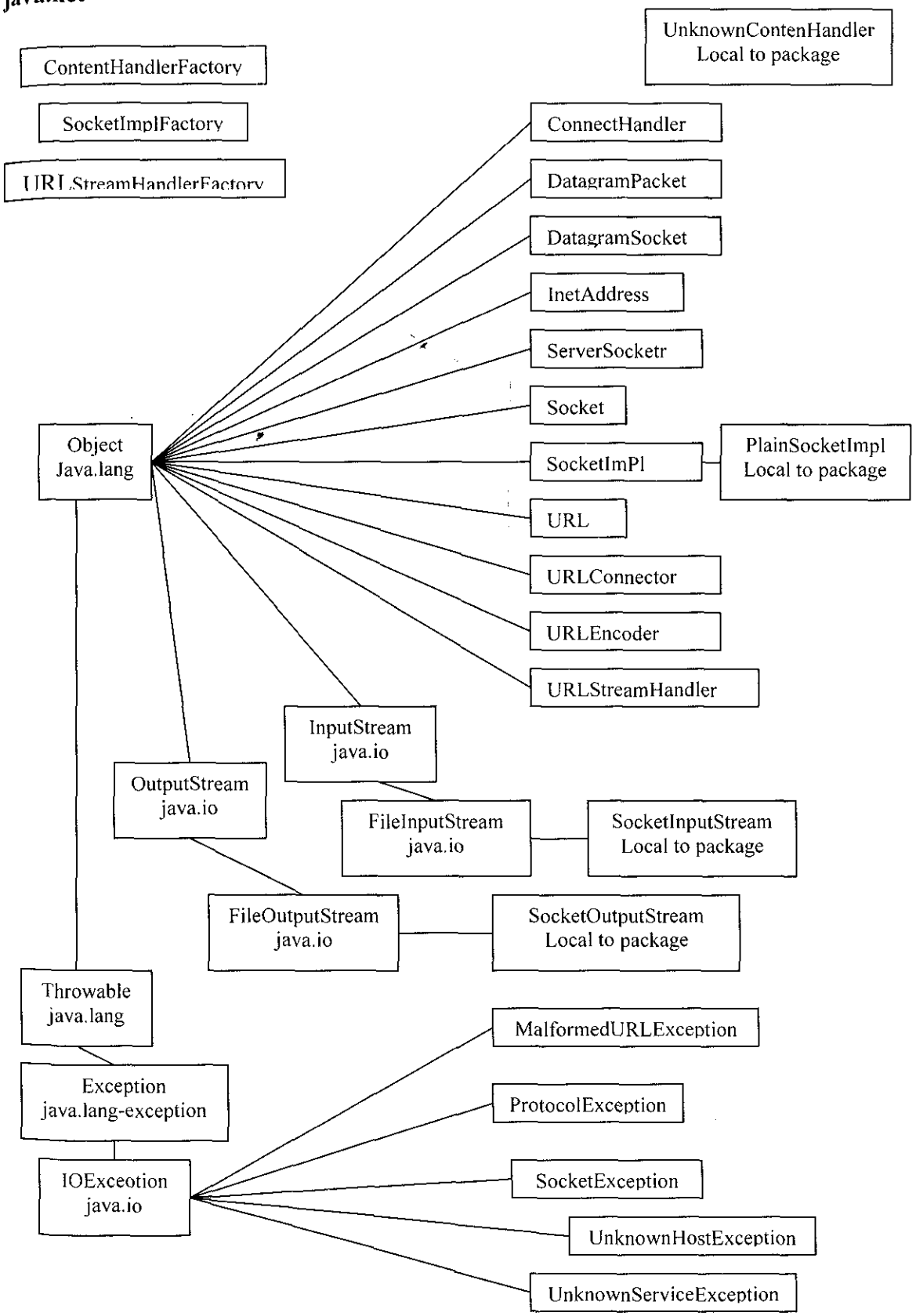
java.lang-exceptions



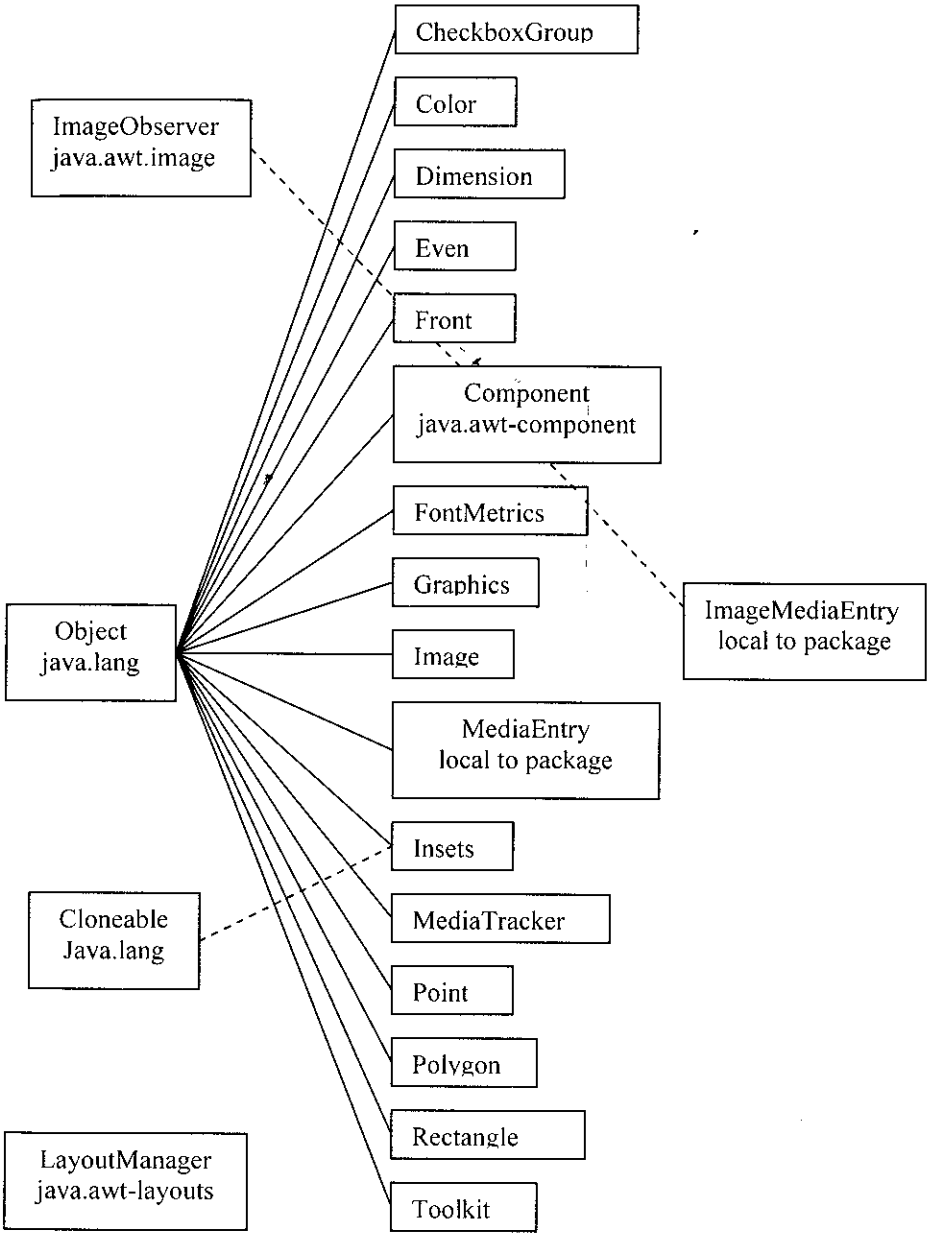
java.io


BufferedInputSt
DataInput
DataInputStrea
LineNumberInput
File
PushbackInputSt
FilterInputStream
FileDescriptor
ByteArrayInputStream
InputStream
FileInputStream
PipesInputStream
SequenceInputStream
RamdomAccessFile
StringBufferInputStream
DataOutput
Object
Java.lang
ByteArrayOutputStream
OutputStream
FileOutputStream
StreamTokenizer
PipedOutputStream
FilterOutputStream
DataOutputStrea
BufferedOutputStr
FilenameFilter
PrintStream
EOFException
IOException
FileNotFoundExce
InterruptedIOExce
Throwable
Java.lang
Exception
Java.lang.- Exception
UTFDataFormatExc

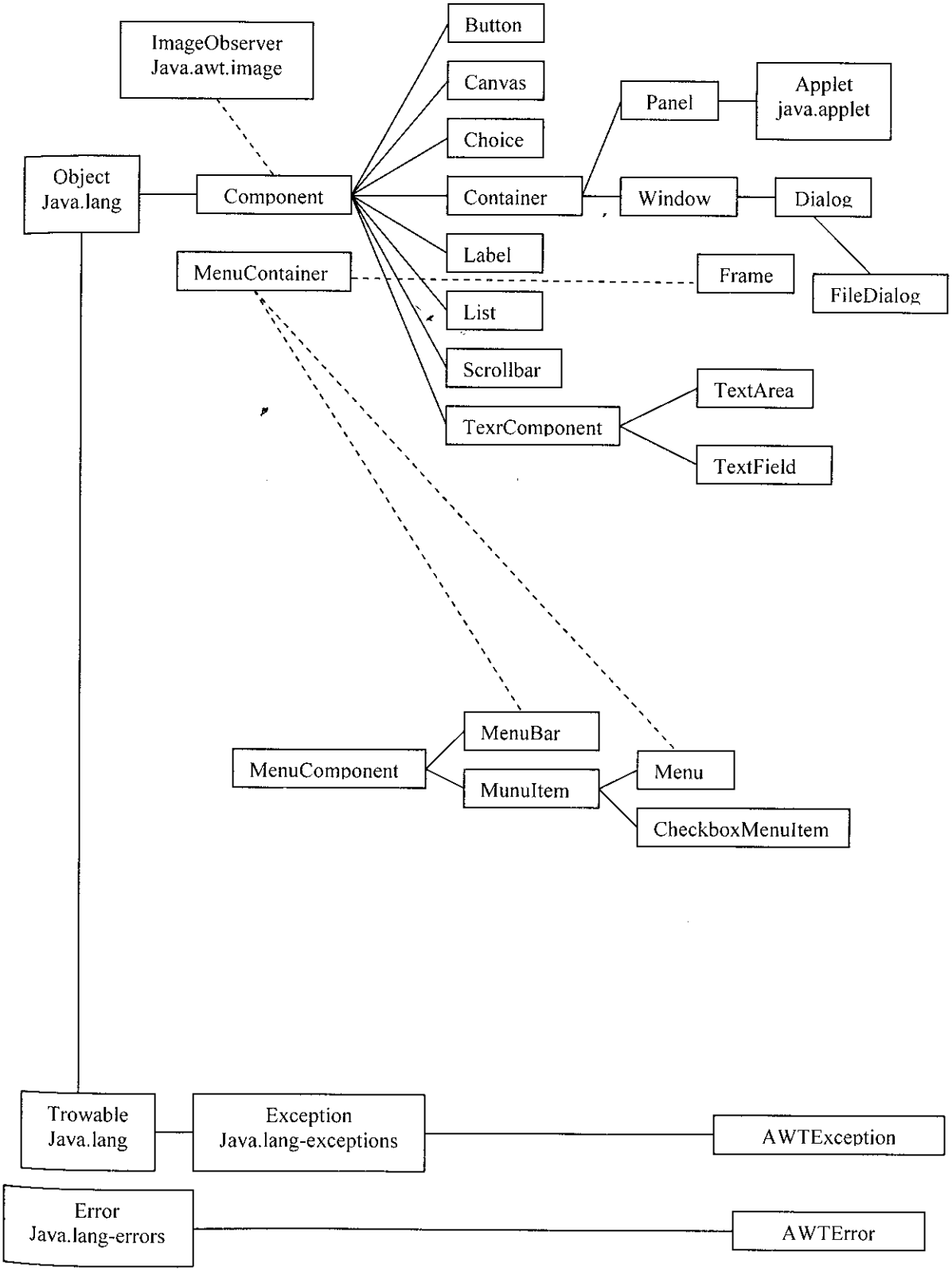
java.net


UnknownContenHandler
Local to package
ContentHandlerFactory
SocketImplFactory
URLStreamHandlerFactory
ConnectHandler
DatagramPacket
DatagramSocket
InetAddress
ServerSocketr
Socket
Object
Java.lang
SocketImPl
PlainSocketImpl
Local to package
URL
URLConnector
URLEncoder
URLStreamHandler
InputStream
java.io
OutputStream
java.io
FileInputStream
java.io
SocketInputStream
Local to package
FileOutputStream
java.io
SocketOutputStream
Local to package
Throwable
java.lang
MalformedURLException
Exception
java.lang-exception
ProtocolException
IOExceotion
java.io
SocketException
UnknownHostException
UnknownServiceException

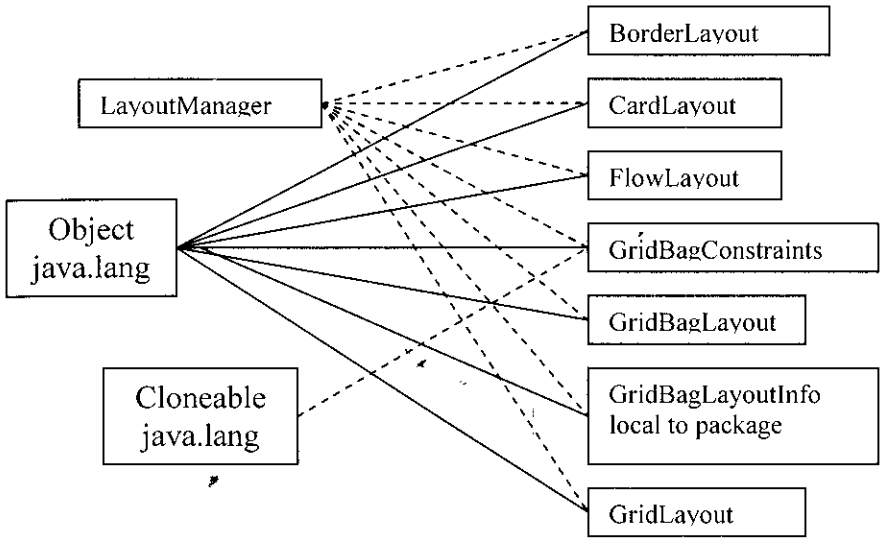
Java.awt


CheckboxGroup
Color
Dimension
Even
Front
ImageObserver
java.awt.image
Component
java.awt-component
FontMetrics
Graphics
Image
Object
java.lang
ImageMediaEntry
local to package
MediaEntry
local to package
Insets
Cloneable
Java.lang
MediaTracker
Point
Polygon
Rectangle
LayoutManager
java.awt-layouts
Toolkit

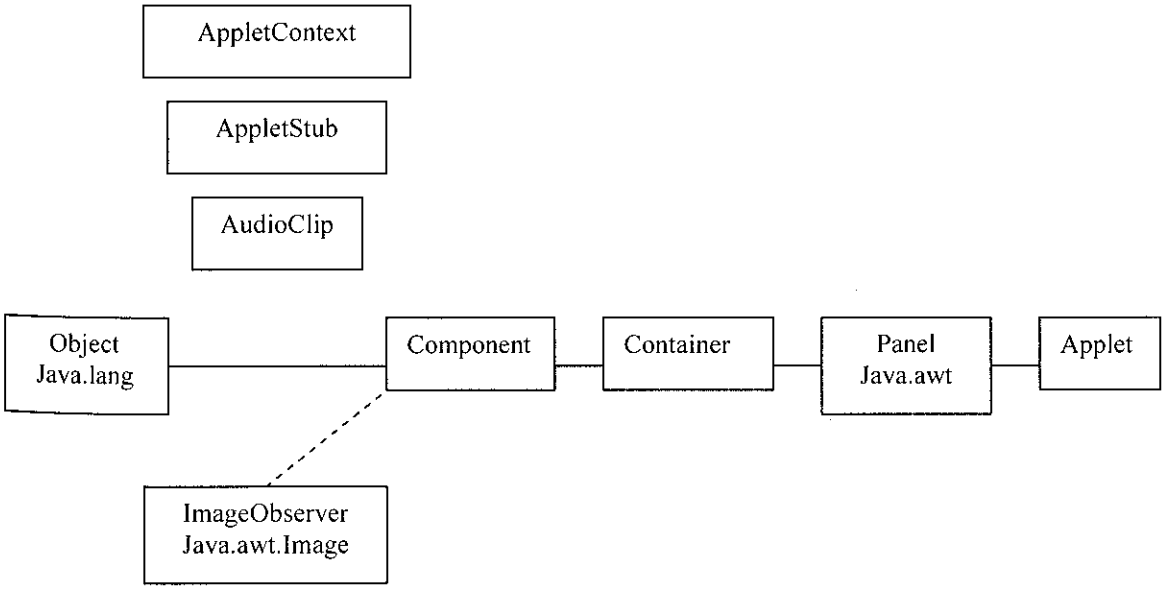
java.awt-conponent



java.awt-layouts



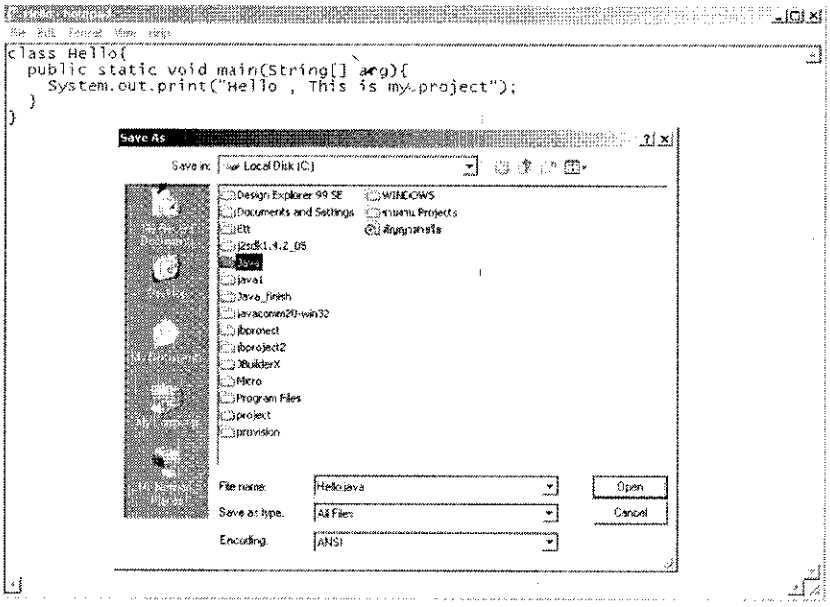
java.applet

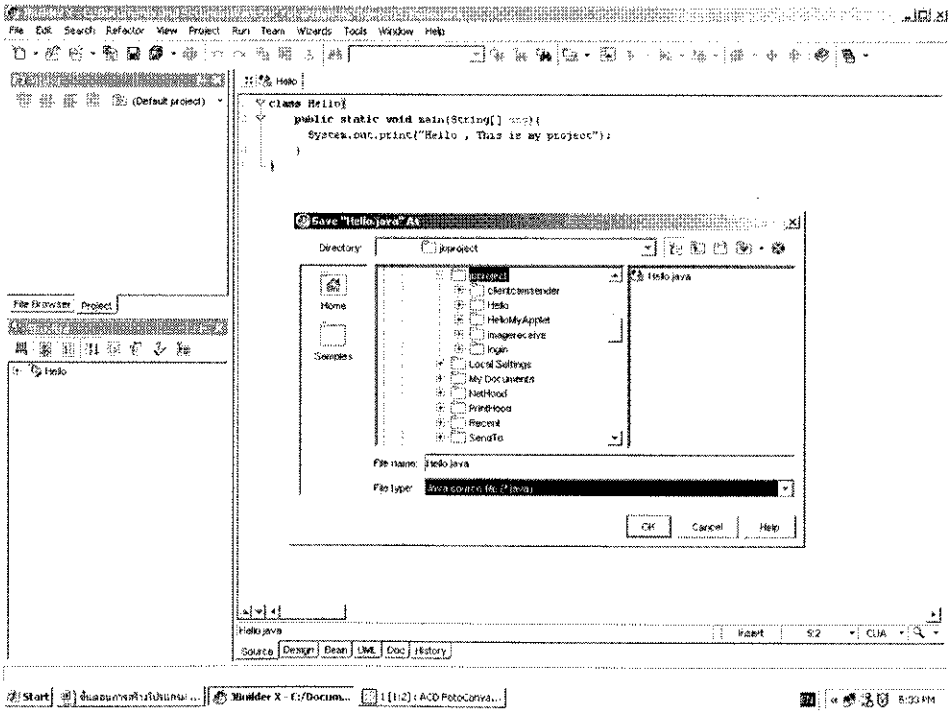


ภาคผนวก ข

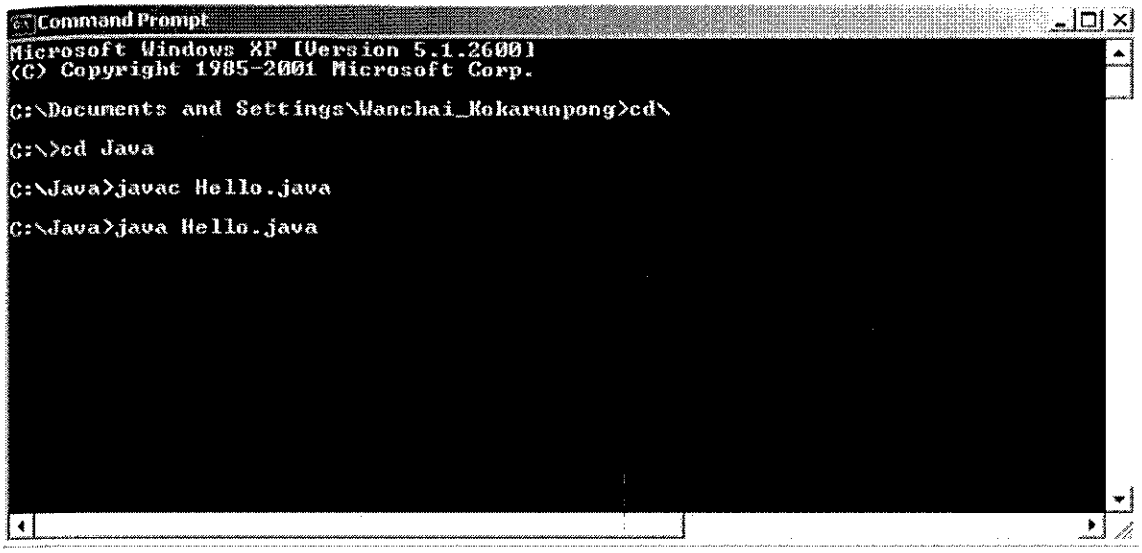
ขั้นตอนการสร้างโปรแกรม Java application

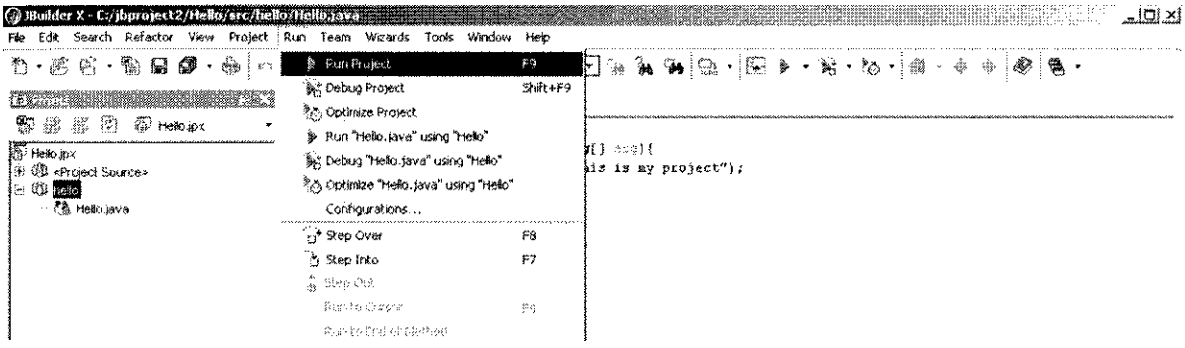
1. การสร้าง Java source code โดยพิมพ์โปรแกรมลงบน Editor ของ Java ไม่ว่าจะเป็น Notepad , Wordpad แต่เพื่อลดข้อผิดพลาดในการใช้คำสั่งควรใช้ Editplus หรือ JBulider เนื่องจาก Editor เหล่านี้มีการใช้สีแยกความแตกต่างของคำสั่งต่างๆ ในโปรแกรม ซึ่งช่วยให้ง่ายต่อการอ่านโปรแกรมมากขึ้น (ในที่นี้จะใช้โปรแกรม JBuliderX) จากนั้นบันทึกไฟล์นั้นด้วยนามสกุล Java (.java)



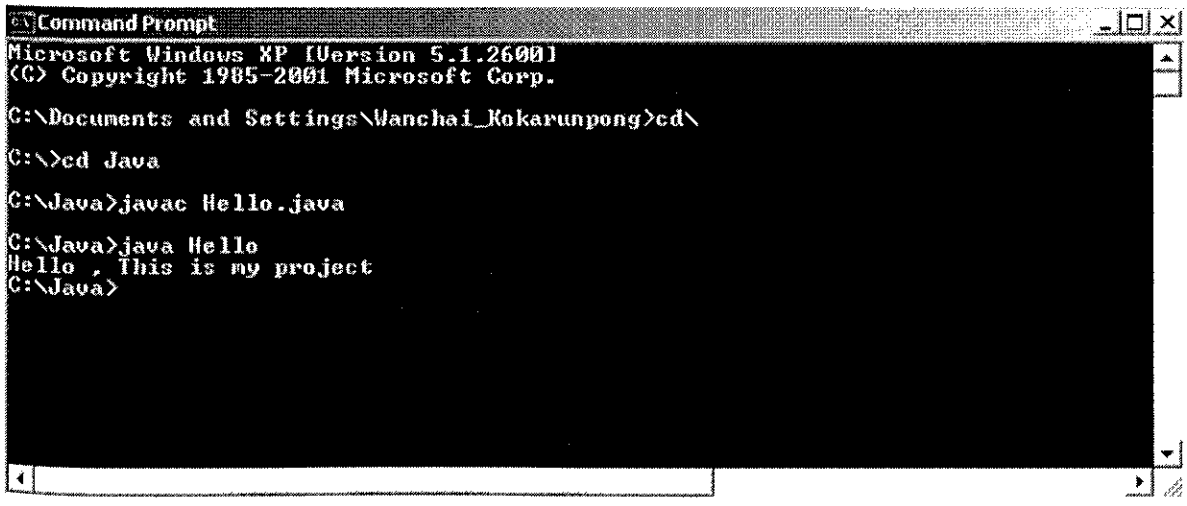


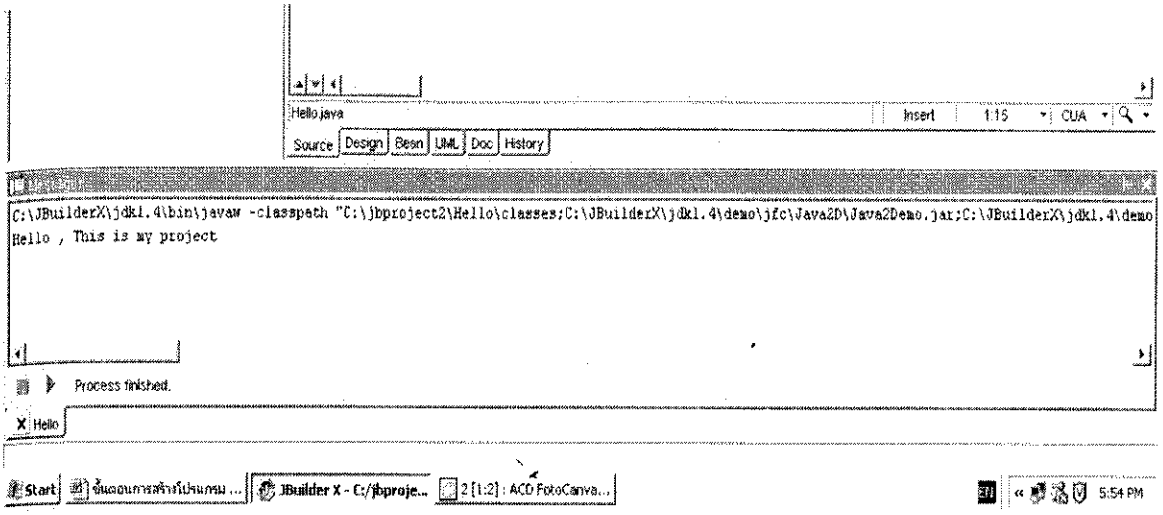
2. ทำการคอมไพล์ด้วยคำสั่ง javac Hello.java จากนั้นจึงใช้ Java Interpreter รันแอพพลิเคชัน โดยใช้คำสั่ง java Hello





3. แสดงผลการรันแอพพลิเคชัน ซึ่งจะแสดงข้อความออกมาทางจอภาพ

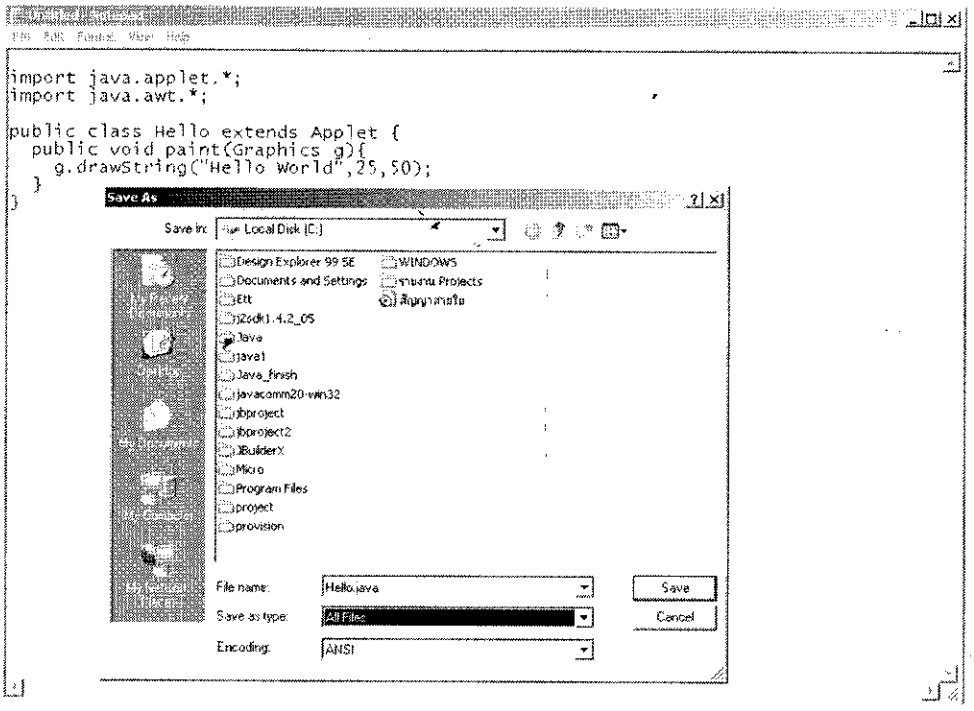


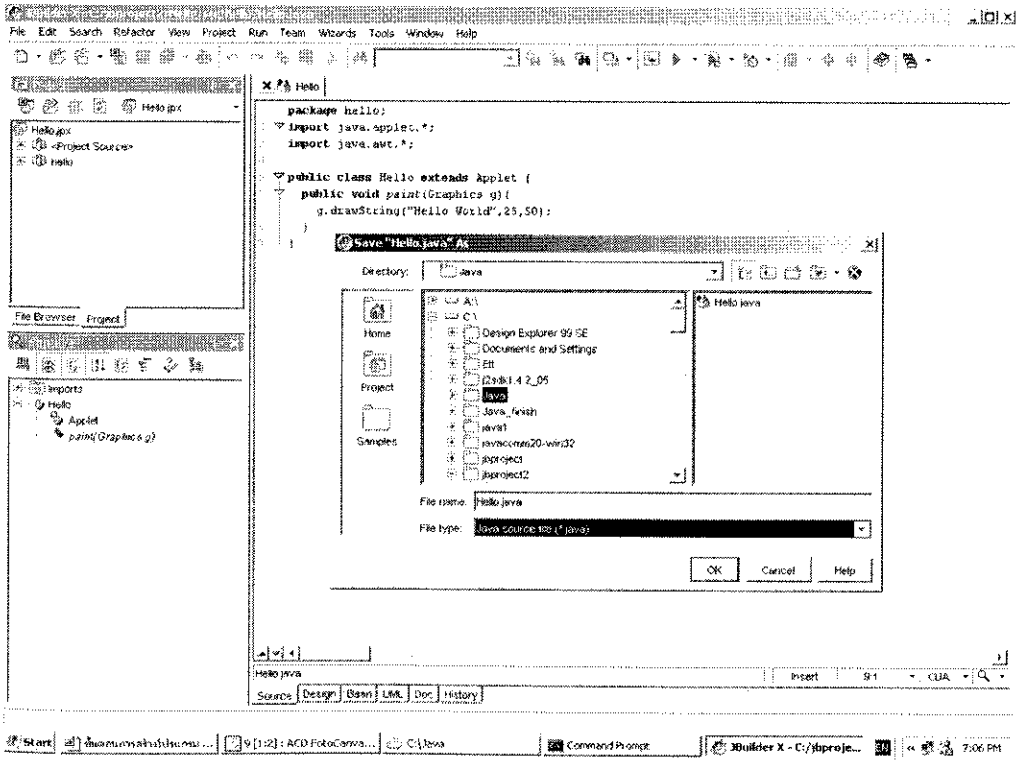

Hello.java
Insert
1:15
CUA
Source
Design
Besn
UML
Doc
History
C:\JBuilderX\jdk1.4\bin\javaw -classpath "C:\jbproject2\Hello\classes;C:\JBuilderX\jdk1.4\demo\jfc\Java2D\Java2Demo.jar;C:\JBuilderX\jdk1.4\demo
Hello , This is my project
Process finished.
Hello
Start
JBuilder X - C:/jbproje...
2 [1:2] : ACD FotoCanva...
5:54 PM

ภาคผนวก ค

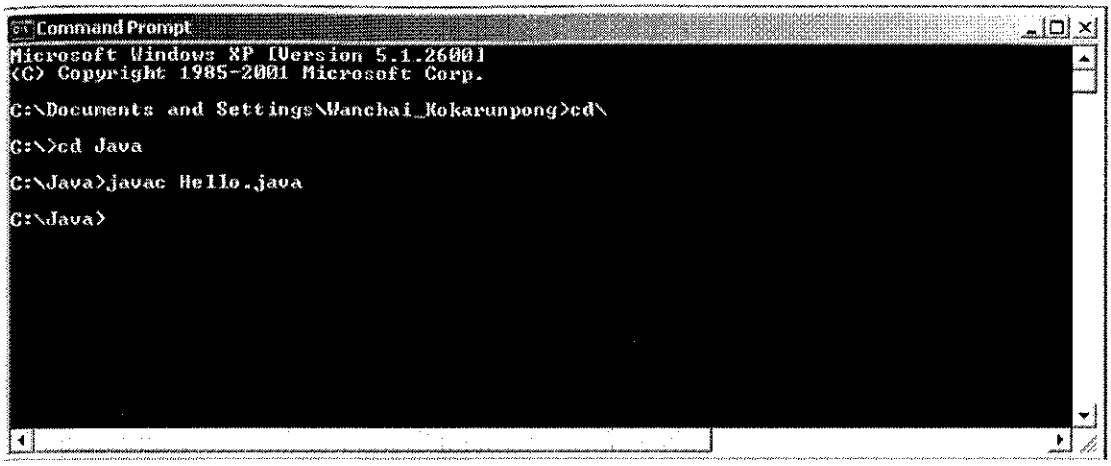
ขั้นตอนการสร้างโปรแกรมภาษา Java Applet

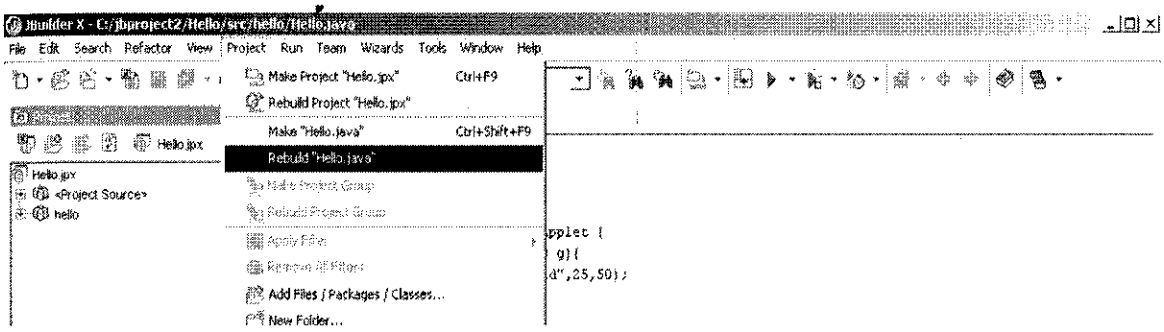
1. การสร้าง Java source code โดยพิมพ์โปรแกรมลงบน Editor โดยทำเหมือนการสร้างโปรแกรม Java application



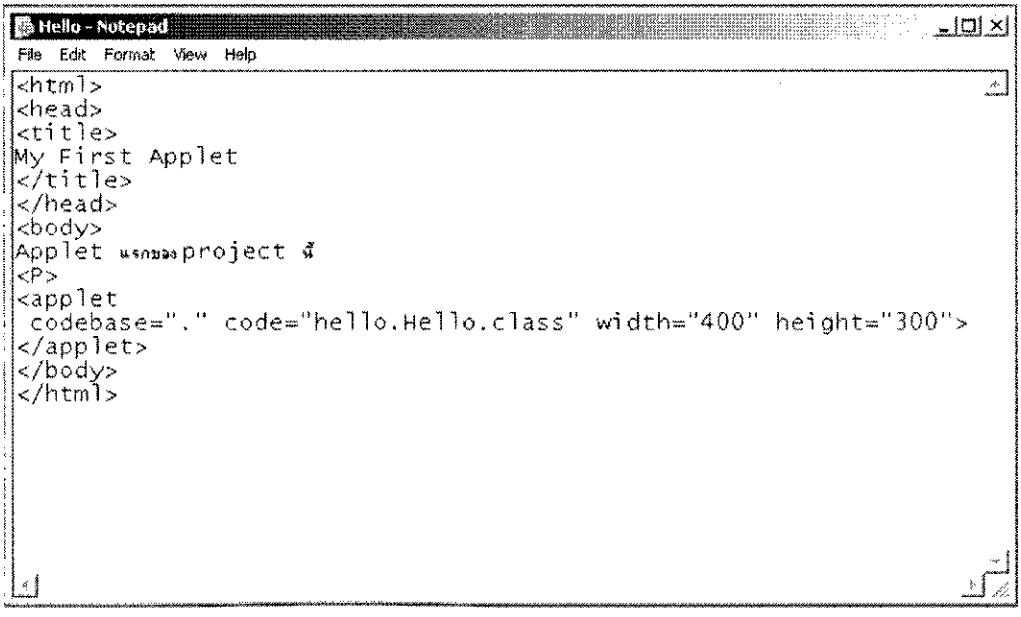


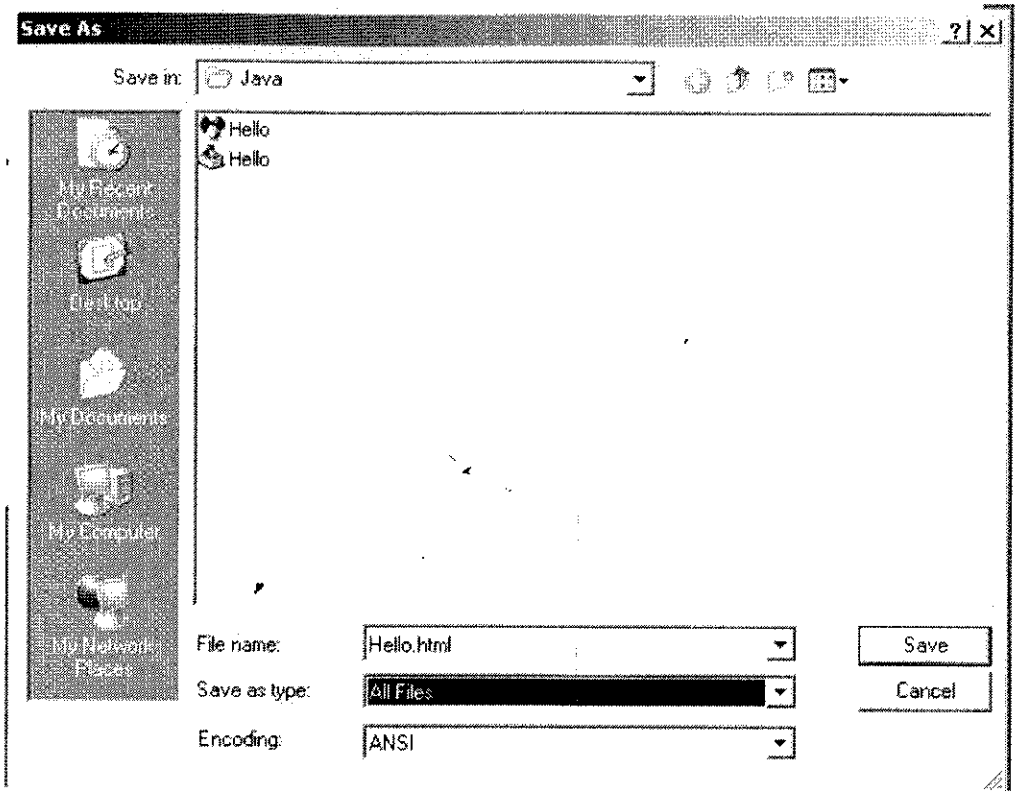
2. ทำการคอมไพล์ด้วยคำสั่ง javac Hello.java



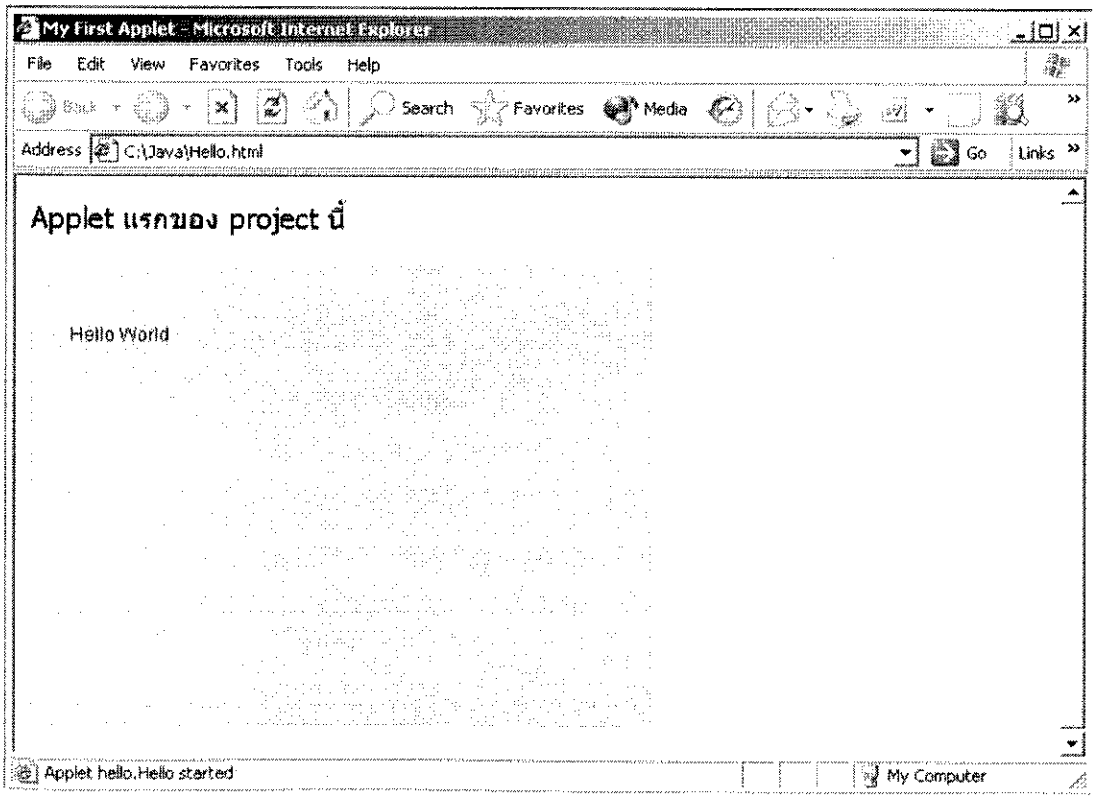


3. นำไฟล์คลาสที่ได้ไปเก็บไว้ในเอกสาร HTML แล้วให้ web browser เป็นตัวเรียกอ่านไฟล์คลาสนั้นๆ โดยเอกสาร HTML จะเก็บ applet เอาไว้โดยใช้แท็ก <APPLET></APPLET> จากนั้นบันทึกเป็นไฟล์ HTML (.html)





4. เรียกดู applet โดยใช้ web browser เช่น Internet Explorer , Netscape เป็นต้น โดยเรียกไฟล์ชื่อ Hello.html ขึ้นมา

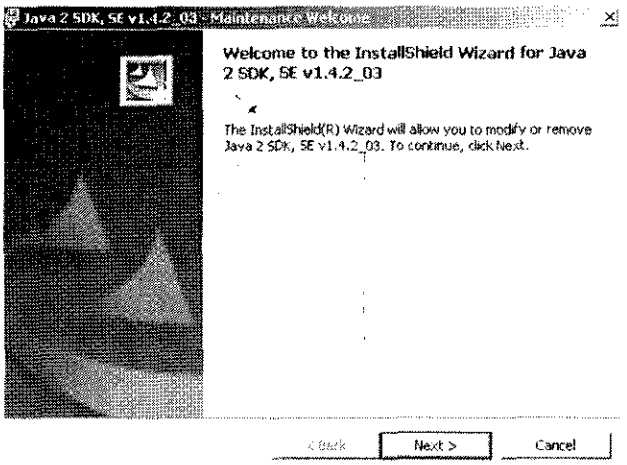


ภาคผนวก ง

การติดตั้ง J2SE

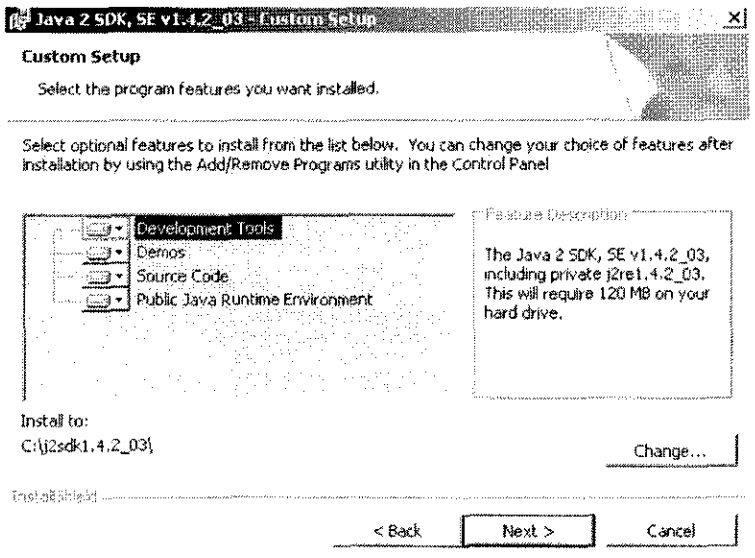
การติดตั้ง J2SE มีขั้นตอนดังนี้

1. เริ่มต้นโดยดับเบิ้ลคลิกที่ไฟล์ j2sdk-1_4_2_03-windows-i586-pที่เราดาวน์โหลดมา
2. โปรแกรมจะ extract ไฟล์เพื่อเตรียมการติดตั้ง
3. จากนั้นจึงเข้าสู่หน้าจอแสดงข้อความต้อนรับการติดตั้งโปรแกรม



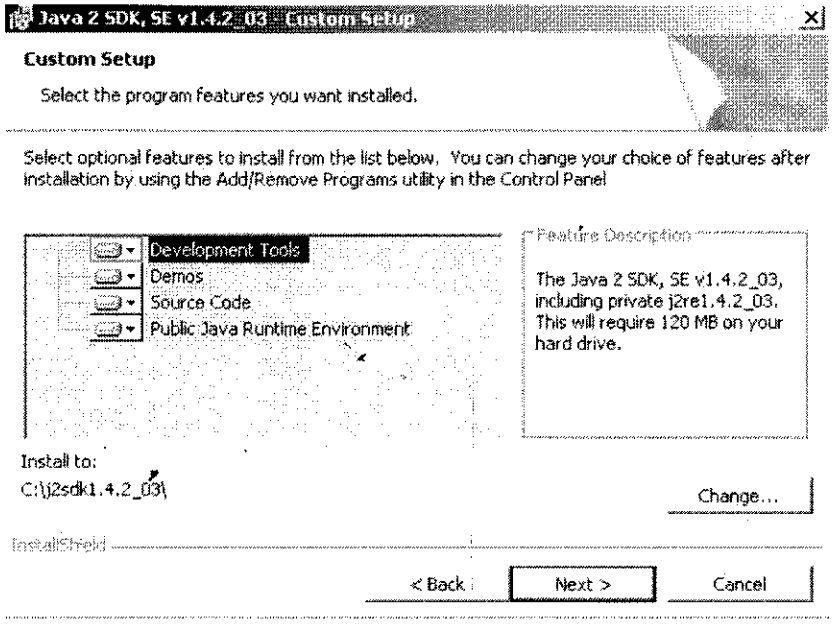
4. เมื่อคลิกปุ่ม Next ก็จะเข้าสู่หน้าจอแสดงรายละเอียดเงื่อนไขข้อตกลงเกี่ยวกับการนำโปรแกรมไปใช้งาน หรือที่เรียกว่า License Agreement ตามธรรมเนียม

5. พอคลิกปุ่ม Yes ยอมรับเงื่อนไข หน้าต่างถัดไปจะให้เลือกไดเรกทอรีที่ต้องการติดตั้ง J2SE ลงไป ดังรูป ในที่นี้จะติดตั้งในไดเรกทอรีที่กำหนดเป็นดีฟอลต์ (default) ไว้ล่วงหน้าแล้วคือ C:\ j2sdk1.4.2_03แต่ถ้าต้องการติดตั้งในไดเรกทอรีอื่นๆ ก็สามารถเปลี่ยนได้โดยคลิกปุ่ม Browse…



ภาพการเลือกไดเรกทอรีที่ต้องการติดตั้ง

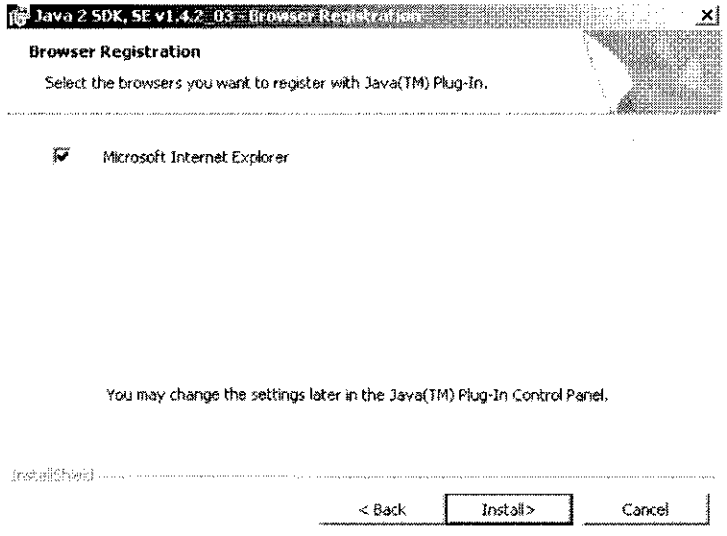
6. เมื่อเลือกไดเรกทอรีและคลิกปุ่ม Next ก็จะไปที่หน้าต่างดังภาพ เพื่อให้เลือกส่วนประกอบที่ต้องการติดตั้ง ในที่นี้ขอให้เลือกหมดทุกอย่างเลย แล้วคลิกปุ่ม Next



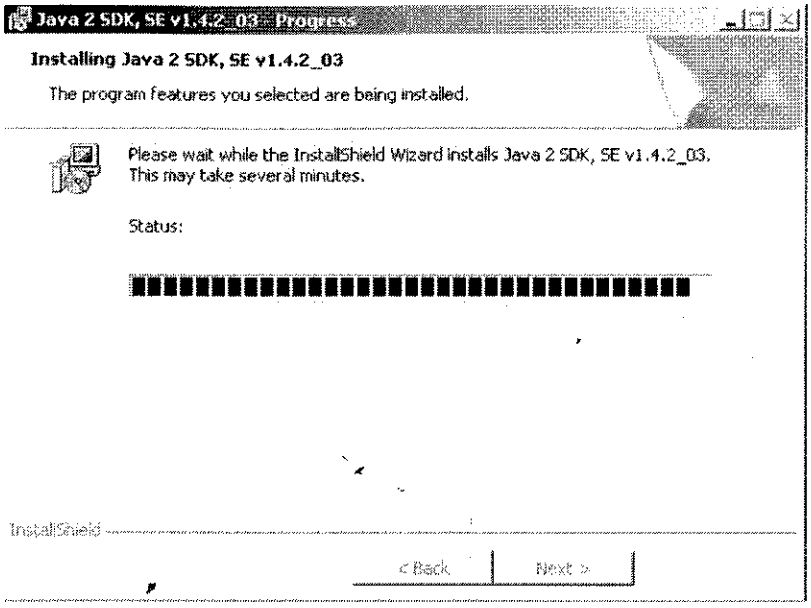
ภาพการเลือกcomponent ที่ต้องการติดตั้ง

7. ต่อจากนั้น จะไปยังหน้าจอดังภาพ เพื่ออับเกรด Java Virtual Machine (JVM) ที่มีอยู่ ในโปรแกรมบราวเซอร์ภายในเครื่องของเรา โดย JVM ทำหน้าที่เป็นตัวช่วยรันโค้ดของ Java

หน้าจอนี้จะให้เราระบุว่าต้องการอับเกรด JVM ในโปรแกรม IE หรือ Netscape เราก็กาเครื่องหมาย √ เข้าไปตามต้องการ แล้วคลิกปุ่ม Next เช่นเคย



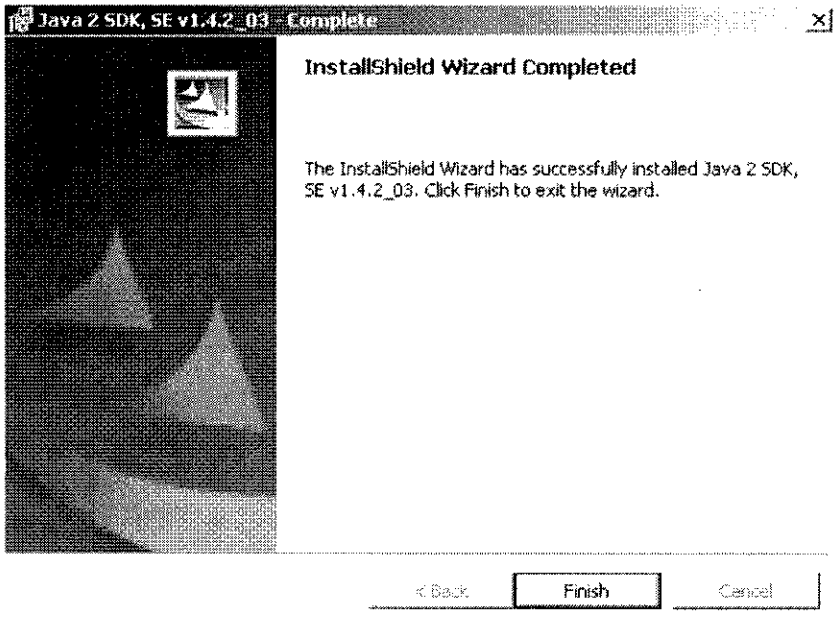
ภาพการ อับเกรด Java Virtual Machine ในโปรแกรมบราวเซอร์



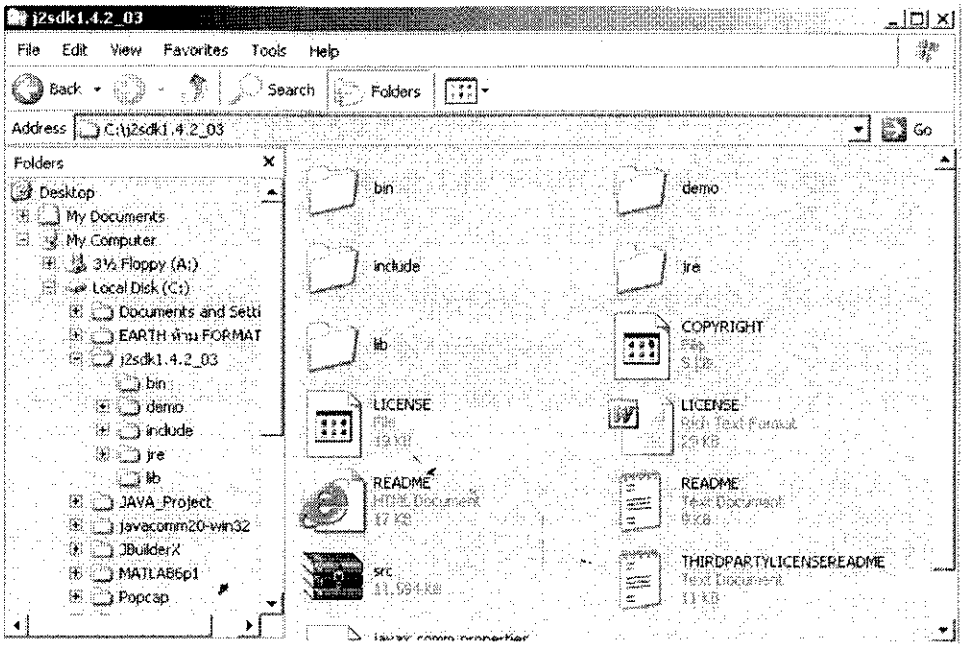
ภาพการกำลังติดตั้ง

8. จากนั้นโปรแกรมจะเริ่มติดตั้ง พร้อมทั้งแสดงเปอร์เซ็นต์ความคืบหน้า ดังภาพ

9. ถ้าการติดตั้งเสร็จสมบูรณ์แล้ว หน้าจอดังภาพ จะปรากฏออกมา เมื่อถึงตอนนี้ ให้คลิก Finish ได้เลย



ภาพการติดตั้งเสร็จสมบูรณ์



ภาพไฟล์และไดเรกทอรีย่อยภายในไดเรกทอรีที่ติดตั้ง J2SE

แล้วเราก็จะได้โปรแกรม J2SE ที่ติดตั้งอยู่ในไดเรกทอ j2sdk1.4.2_03 ซึ่งประกอบด้วยไดเรกทอรีย่อยต่างๆ ดังภาพ

การกำหนดตัวแปร PATH เพื่อให้เรียกใช้สะดวก

หลังจากติดตั้ง J2SE เสร็จเรียบร้อย เราก็สามารถเขียนโปรแกรมด้วยภาษา Java ได้แล้ว โดยโปรแกรมหรือคำสั่งต่างๆ ที่สามารถใช้งานได้ เช่น javac.exe หรือ java.exe มีอยู่ในไดเรกทอรี bin (คำสั่ง javac ทำหน้าที่คอมไพล์ไฟล์ .java ให้กลายเป็นไฟล์ .class ส่วนคำสั่ง java ทำหน้าที่เอาไฟล์ .class ซึ่งคอมไพล์จากคำสั่ง javac แล้วมารันให้ทำงาน)

สมมติว่าเขียนโปรแกรมขึ้นมา 1 โปรแกรม คือไฟล์ Hello.java และเก็บบันทึกไฟล์นี้ไว้ในไดเรกทอรี C:\javaprogram และถ้าเราต้องการคอมไพล์โปรแกรมนี้ให้เป็นไฟล์ .class จะต้องใช้คำสั่ง javac ดังนี้

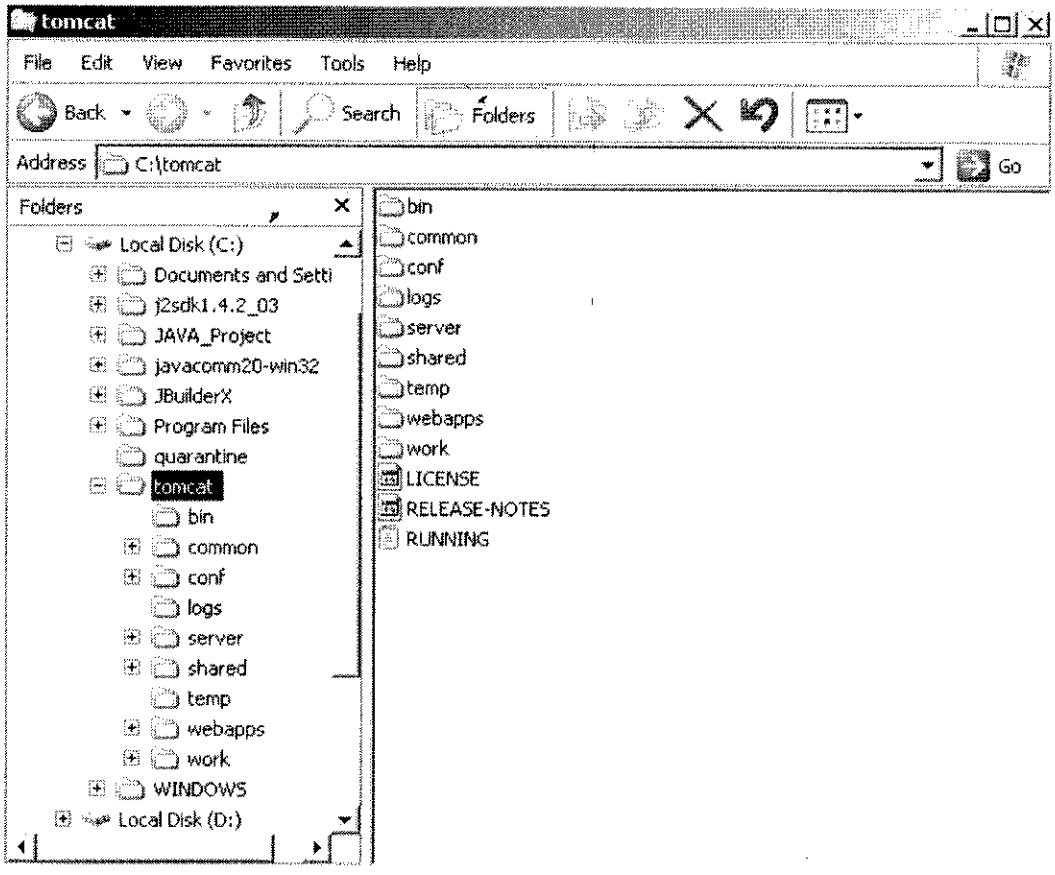
```
C:\javaprogram> C:\ j2sdk1.4.2_03\bin\javac Hello.java
```

จะเห็นได้ว่าเวลาใช้คำสั่ง javac เราจะต้องระบุพาธที่เก็บไฟล์ javac.exe นี้ไว้จริงๆ ด้วย นั่นคือ C:\j2sdk1.4.2_03\bin\javac ซึ่งทำให้เกิดความยุ่งยากในการป้อนคำสั่งคอมไพล์แต่ละครั้ง ดังนั้นเราควรกำหนดตัวแปร PATH ขึ้นมาเพื่อให้สามารถเรียกคำสั่ง javac โดยไม่จำเป็นต้องระบุพาธของคำสั่งนี้เต็มๆ

ภาคผนวก จ

การติดตั้ง Tomcat เป็นเว็บเซิร์ฟเวอร์

การติดตั้งเว็บเซิร์ฟเวอร์ Tomcat นั้น เริ่มต้นด้วยการ unzip ไฟล์ jakarta-tomcat-5.0.18.zip ไปเก็บไว้ในไดรฟ์ C: ซึ่งมันจะสร้างไดเรกทอรีชื่อ jakarta-tomcat-5.0.18 ขึ้นมาโดยอัตโนมัติ แต่เราควรเปลี่ยนชื่อไดเรกทอรีนี้ให้สั้นและกระชับเป็น tomcat เพื่อให้ดูง่ายในการใช้งาน ภายในไดเรกทอรี tomcat จะมีไฟล์และไดเรกทอรีย่อยมากมายดังแสดงในภาพ



หน้าที่ของไดเรกทอรีย่อยแต่ละไดเรกทอรี มีดังนี้

- **webapps** เป็นไดเรกทอรีหลักที่ใช้เก็บไฟล์ภายในเว็บไซต์ของเราเอง โดยไฟล์สำคัญๆ เช่นไฟล์ .jsp หรือไฟล์ .html จะต้องเก็บอยู่ในไดเรกทอรี ROOT ซึ่งซ้อนอยู่ภายในไดเรกทอรีนี้อีกชั้น ไดเรกทอรี webapps\ROOT จึงมีสถานะเป็น default directory คือเมื่อมีการเรียกเปิดไฟล์ต่างๆ ทางเบราเซอร์ เว็บเซิร์ฟเวอร์ Tomcat ก็จะมองหาไฟล์ใน webapps\ROOT เป็นหลัก ดังนั้นสคริปต์ JSP ที่เราจะเขียนขึ้นมา ต้องเก็บไว้ในไดเรกทอรีนี้

แต่ถ้าต้องการให้ไดเรกทอรีใดมีสถานะเป็น default directory เพื่อจะได้เก็บไฟล์สคริปต์ JSP ไว้ในไดเรกทอรีนั้นแทน webapps\ROOT ก็สามารถทำได้ เพียงแต่จะต้องมีการแก้ไขอะไรบางอย่าง

- **bin** เป็นไดเรกทอรีสำหรับเก็บไฟล์ที่เกี่ยวข้องกับการทำงานของ Tomcat เช่น การ start โปรแกรม หรือการ shutdown โปรแกรม Tomcat เป็นต้น
- **classes** และ **lib** เป็นไดเรกทอรีสำหรับเก็บคลาสต่างๆ ที่นำมาใช้จากภายนอก เช่น คลาสที่เขียนขึ้นเองหรือคลาสที่ดาวน์โหลดมาจากเว็บไซต์ เป็นต้น โดยถ้าเป็นไฟล์ .class ต้องเก็บไว้ในไดเรกทอรี classes แต่ถ้าเป็นไฟล์คลาสหลายๆ ไฟล์ที่นำมาบีบอัดรวมกันเป็นไฟล์ .jar ต้องเก็บไว้ในไดเรกทอรี lib
- **common** และ **server** เป็นไดเรกทอรีสำหรับเก็บคลาสมาตรฐานต่างๆ ของ JSP และ Servlet
- **conf** เป็นไดเรกทอรีสำหรับเก็บไฟล์ที่เก็บค่าจำพวก configuration ซึ่งเกี่ยวข้องกับการทำงานของ Tomcat เช่น ไฟล์ server.xml หรือ ไฟล์ web.xml เป็นต้น
- **logs** เป็นไดเรกทอรีสำหรับเก็บไฟล์ .log ของ Tomcat โดยไฟล์จำพวกนี้ทำหน้าที่เก็บบันทึกสถิติต่างๆ ในลักษณะ text file ธรรมดา เช่นผู้ที่เข้ามาชมเว็บไซต์ของเรามาจาก IP address ไหน เปิดเข้าชมเว็บเพจไหน หรือไฟล์ไหนในวันและเวลาใด เป็นต้น
- **temp** สำหรับเก็บไฟล์ชั่วคราวต่างๆ ของ Tomcat
- **work** เป็นไดเรกทอรีสำหรับเก็บไฟล์ .java และ .class ซึ่ง JSP Container แปลงมาจากไฟล์ .jsp ที่เราเขียนขึ้น

การปรับแต่งให้เรียกใช้ **Tomcat** สะดวกเช่นกัน

หลังจากติดตั้ง Tomcat เรียบร้อยแล้ว จะต้องเพิ่มตัวแปรเข้าไปในระบบเหมือนกับตัวแปร PATH ซึ่งขั้นตอนการกำหนดตัวแปรก็จะคล้ายๆ กัน นั่นคือ ในกรณีของ Windows 95, 98 ให้เพิ่มบรรทัดนี้ลงไปในไฟล์ C:\autoexec.bat (ลักษณะเดียวกับการเพิ่มตัวแปร PATH ลงในไฟล์นี้)

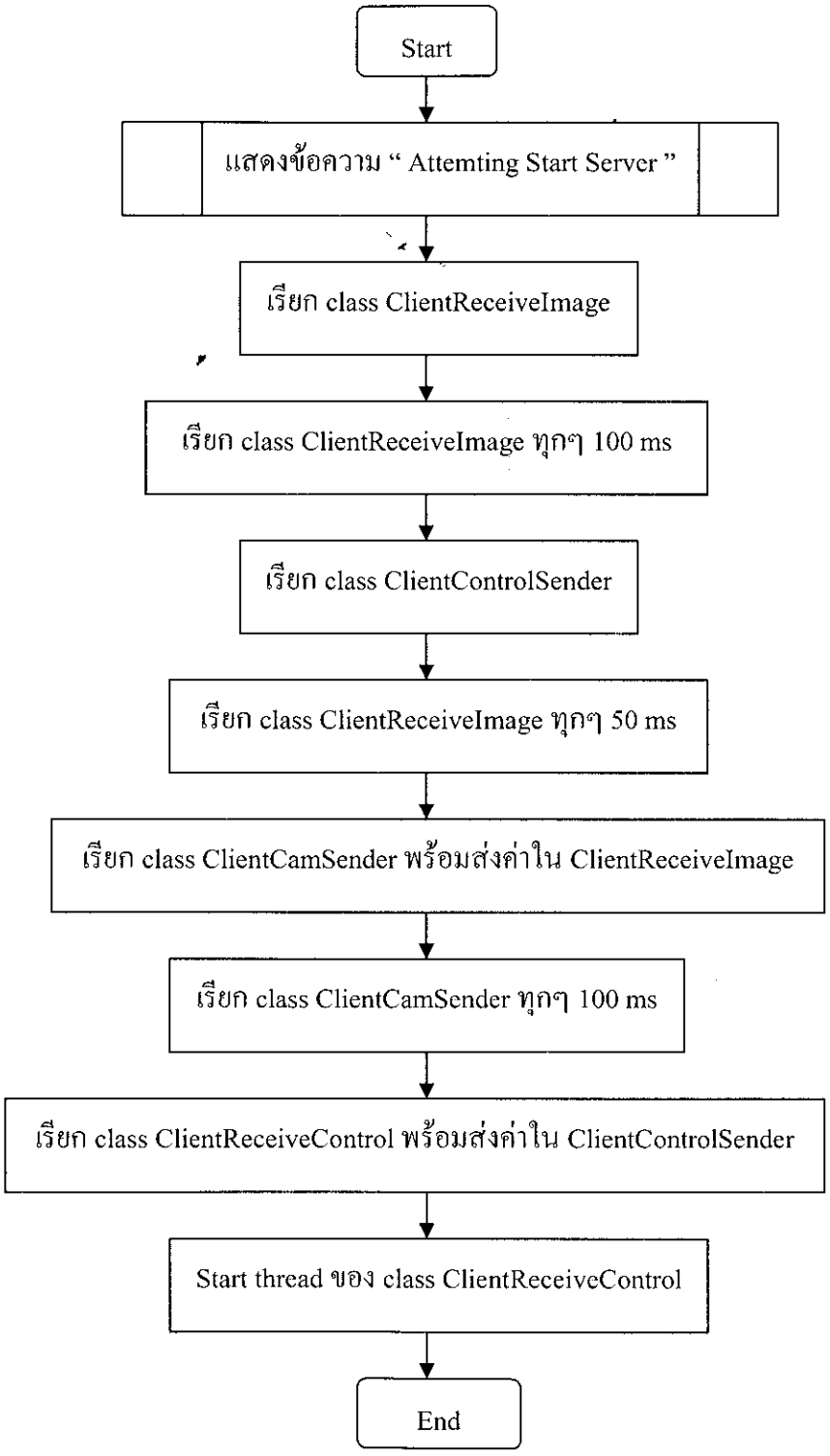
```
set JAVA_HOME=C:\ j2sdk1.4.2_03
set CATALINA  HOME=C:\tomcat
```

ส่วน Windows Me, NT, 2000, XP จะเป็นหน้าต่างให้กรอกข้อมูล โดยกำหนดตัวแปรใหม่ชื่อ JAVA_HOME มีค่าเป็น C:\ j2sdk1.4.2_03และตัวแปรอีกตัวหนึ่ง ชื่อ CATALINA_HOME มีค่าเป็น C:\tomcat หน้าที่ของ CATALINA_HOME เป็นตัวบ่งบอกให้ระบบรู้ว่าไดเรกทอรีที่ติดตั้ง Tomcat คือไดเรกทอรีไหน ส่วนตัวแปร JAVA_HOME ก็เป็นตัวบ่งบอกให้ระบบรู้ว่า ไดเรกทอรีที่ติดตั้ง J2SE คือ ไดเรกทอรีไหน

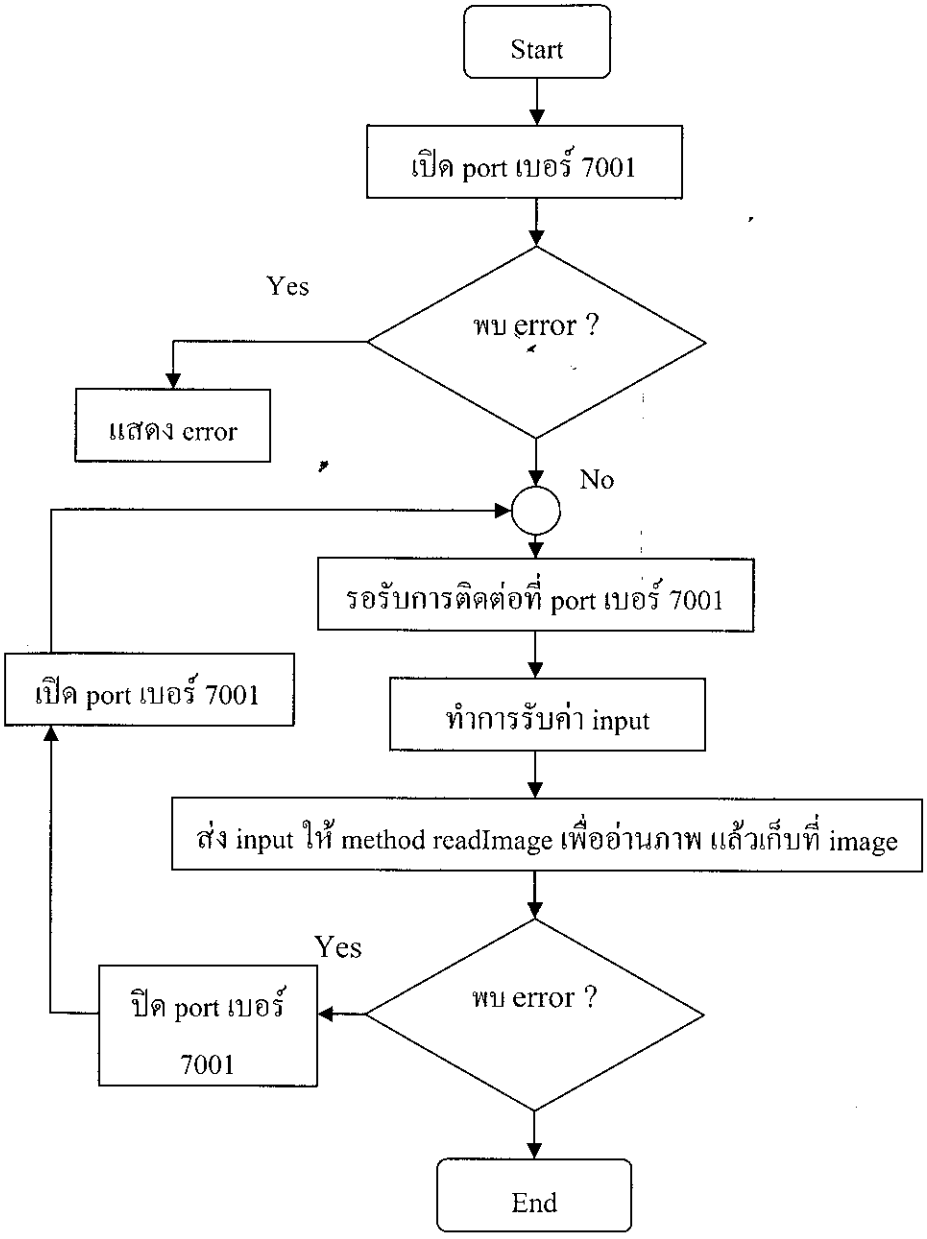
ภาคผนวก ฉ

Flow chat Package serverclass

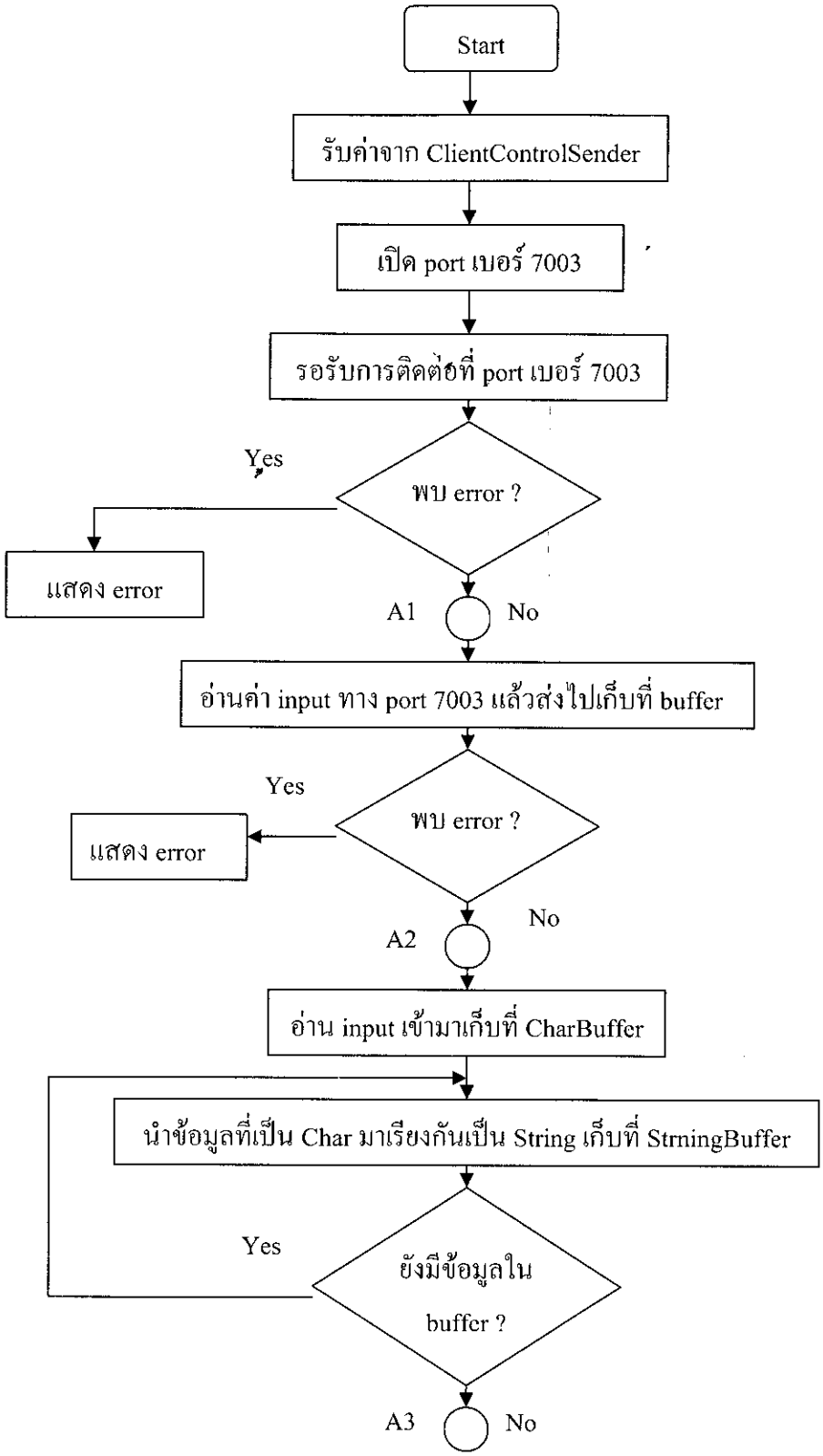
Class MainClass

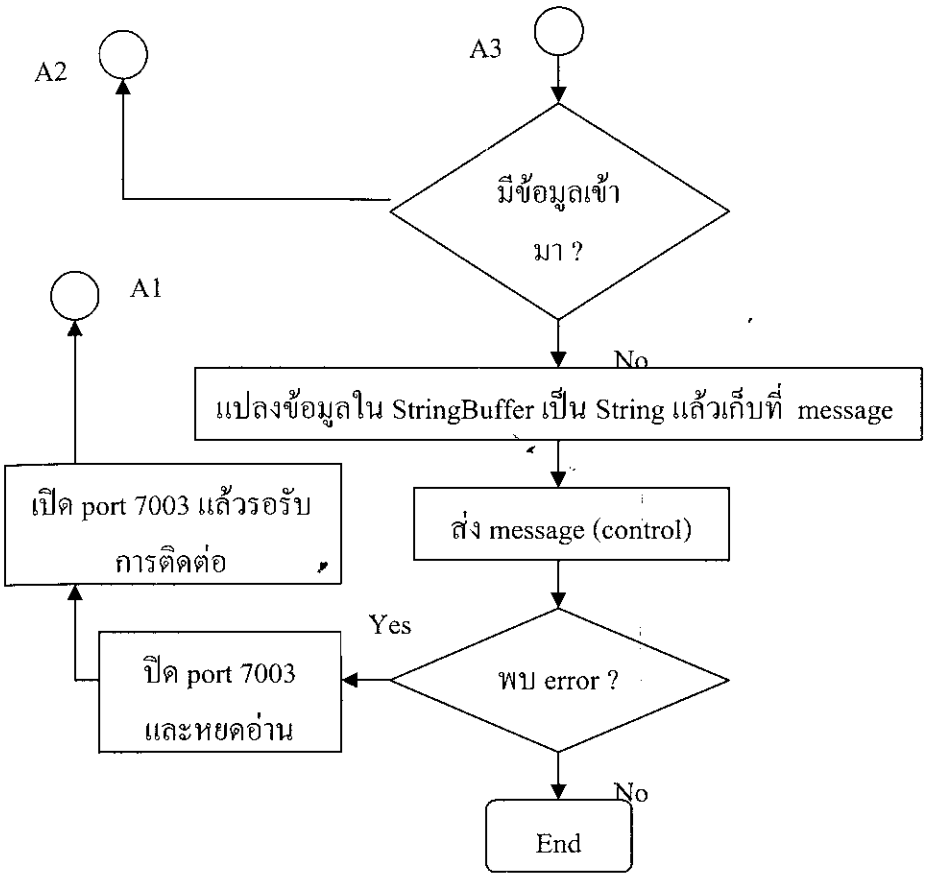


Class ClientReceiveImage

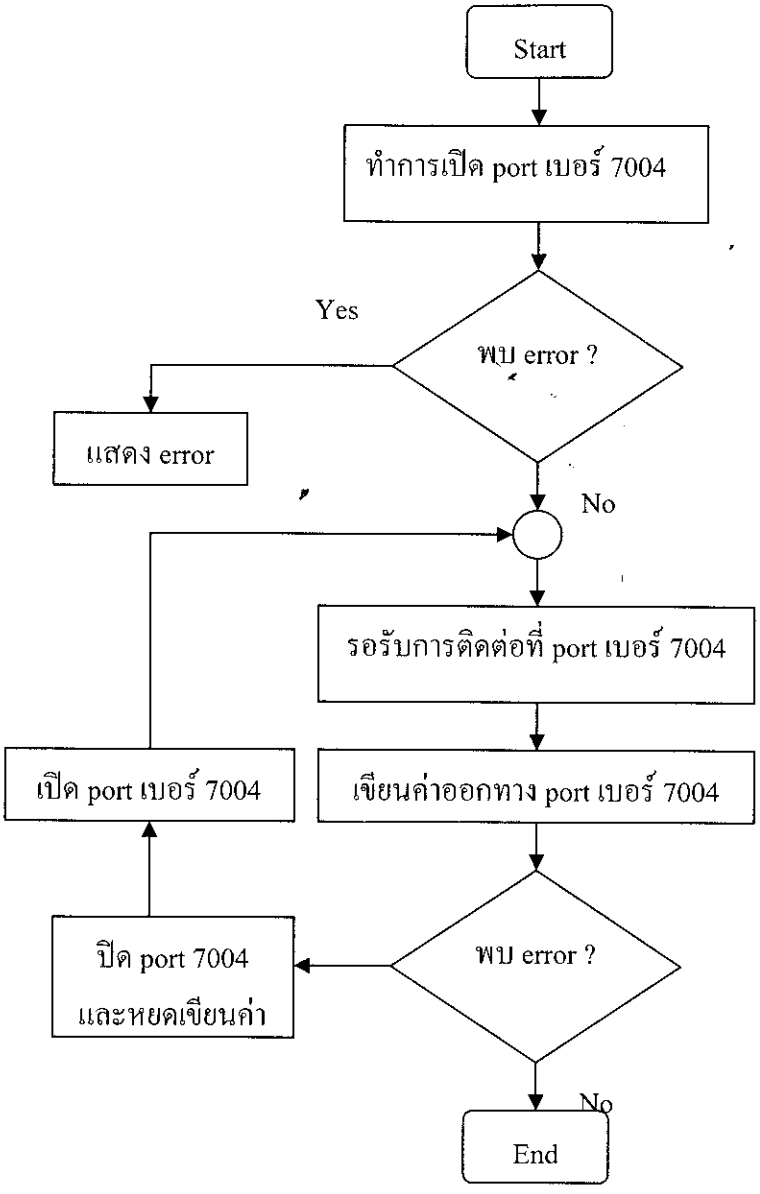


Class ClientReceiveControl

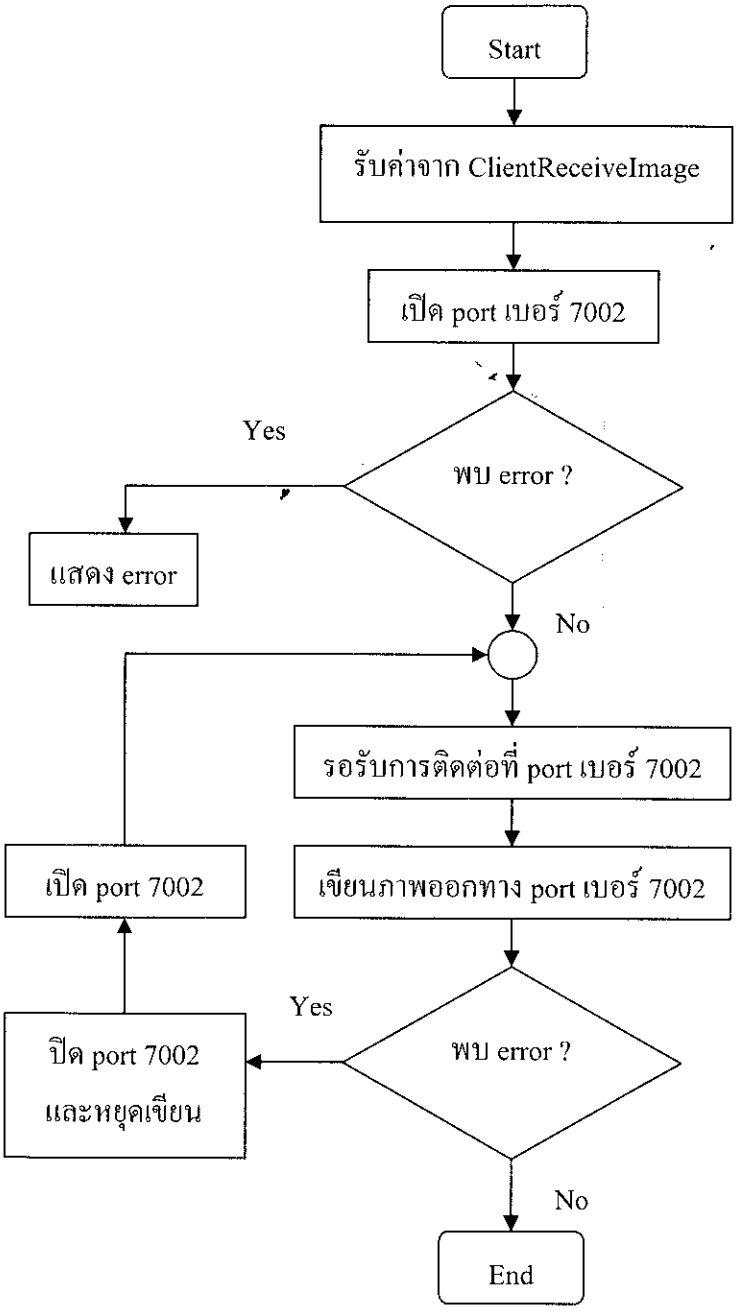



A2
A3
มีข้อมูลเข้า
มา ?
A1
No
แปลงข้อมูลใน StringBuffer เป็น String แล้วเก็บที่ message
เปิด port 7003 แล้วรอรับ
การติดต่อ
ส่ง message (control)
Yes
ปิด port 7003
และหยุดอ่าน
พบ error ?
No
End

Class ClientControlSender

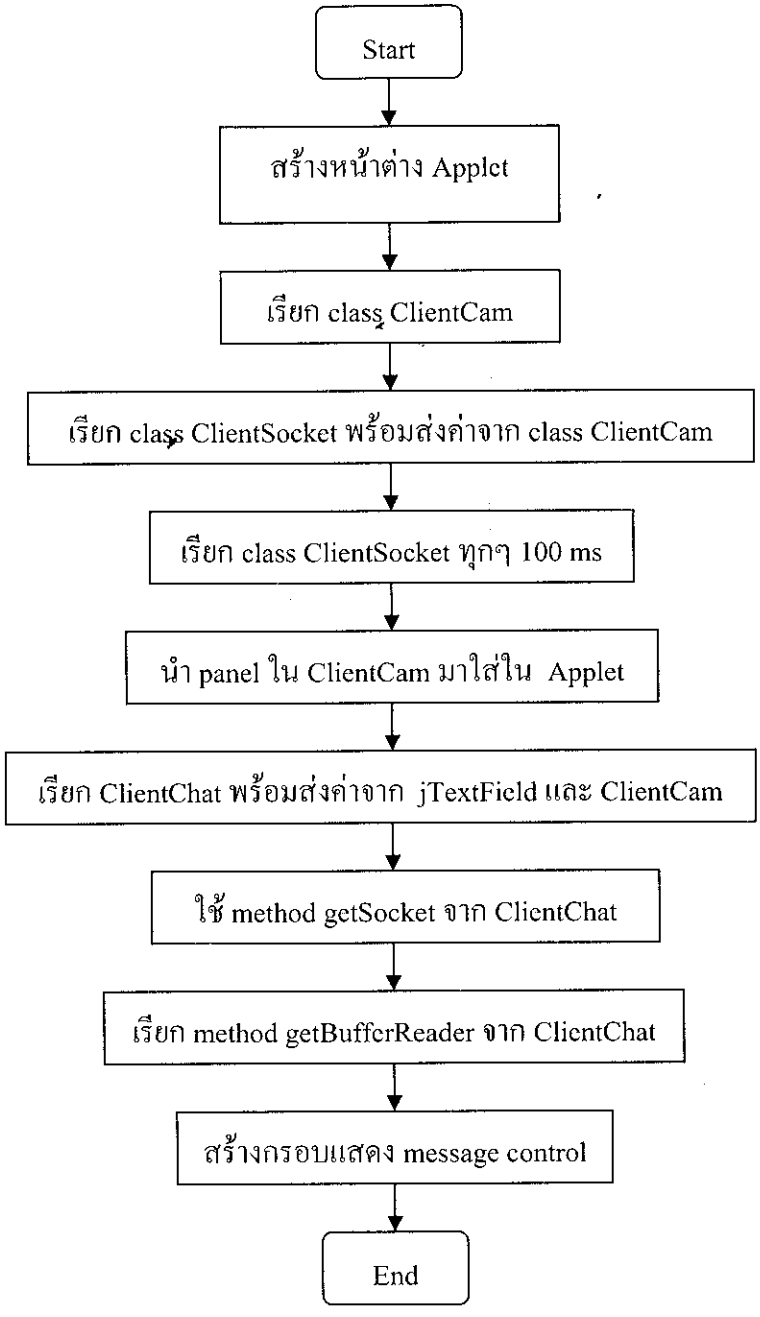


Class ClientCamSender

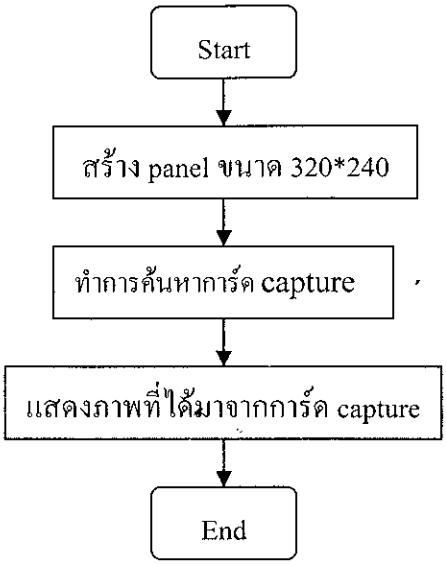


Flow chat Package clientcamsender

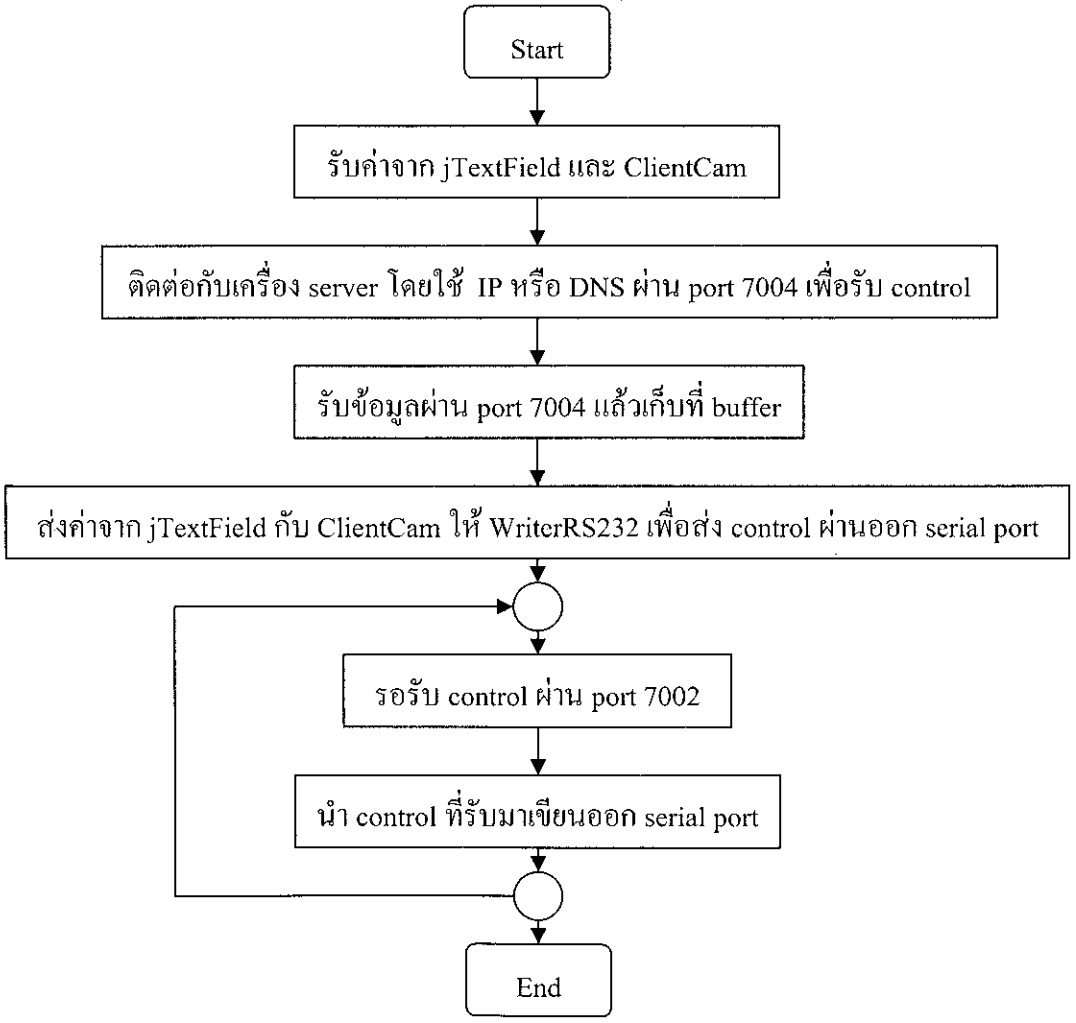
Class clientcamsender



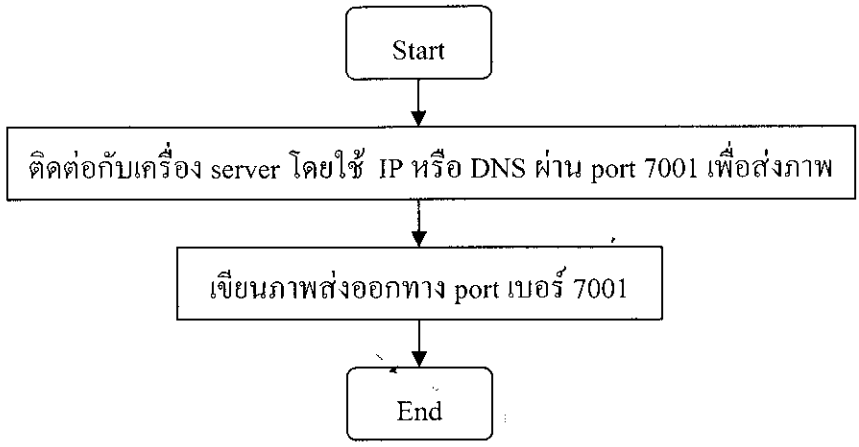
Class ClientCam



Class ClientChat

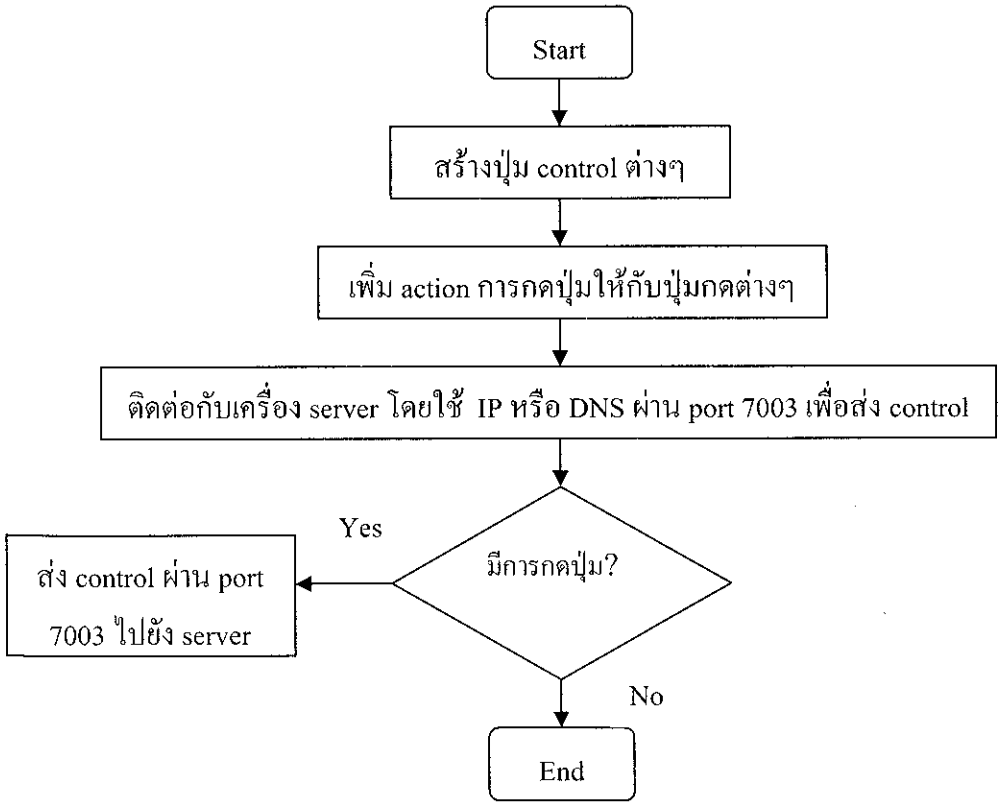


Class ClientSocket

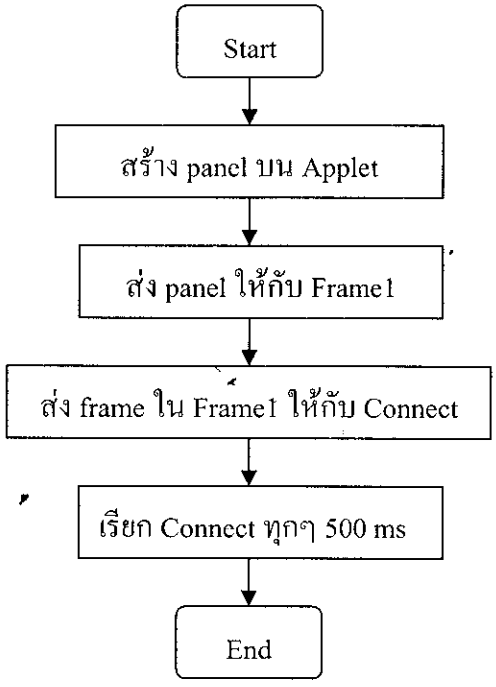


Flow chat Package imagereceive

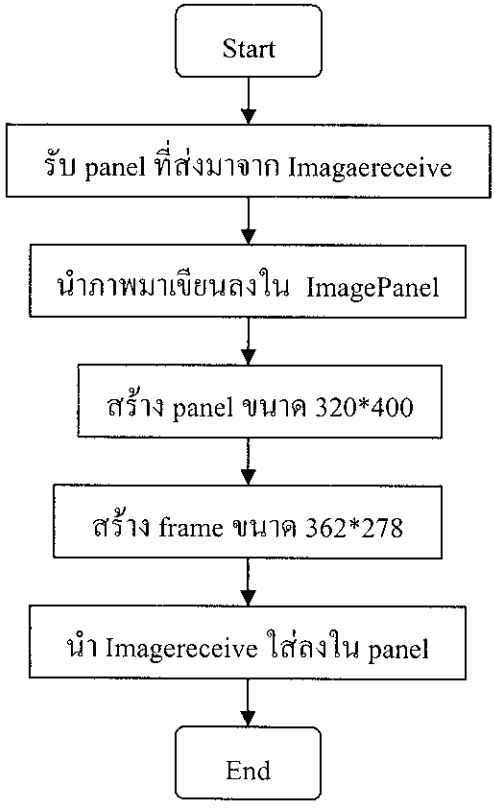
Class ClientControlSender



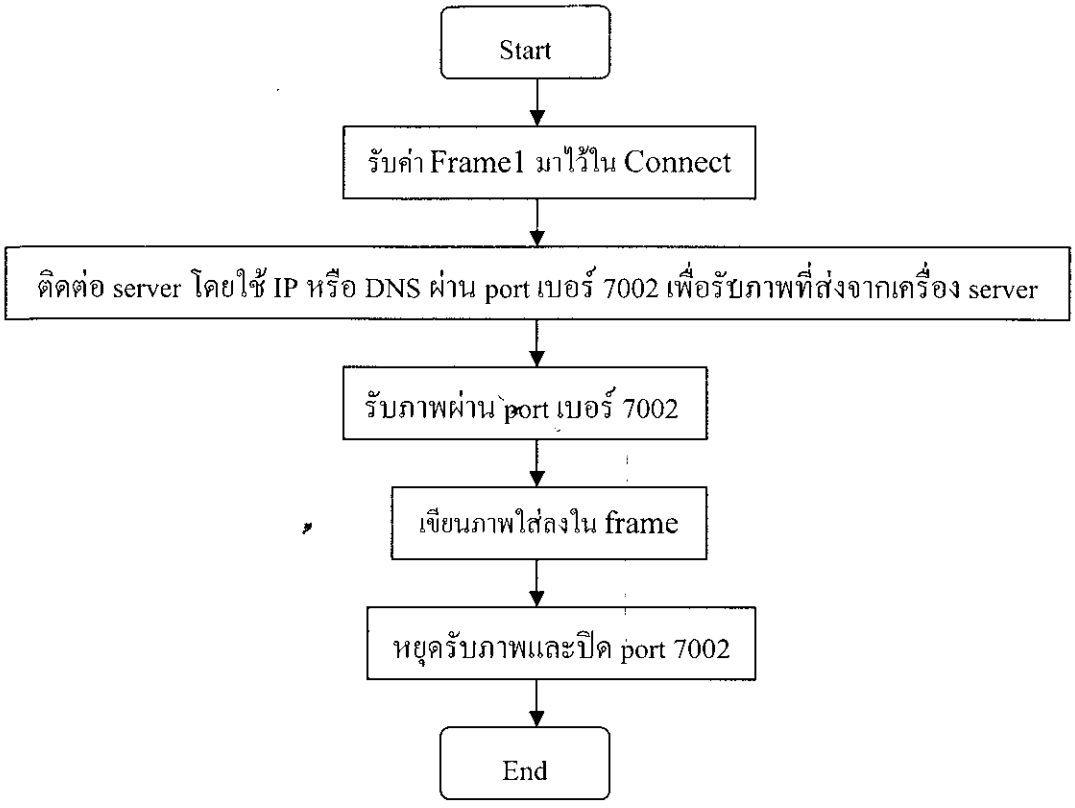
Class Imagereceive



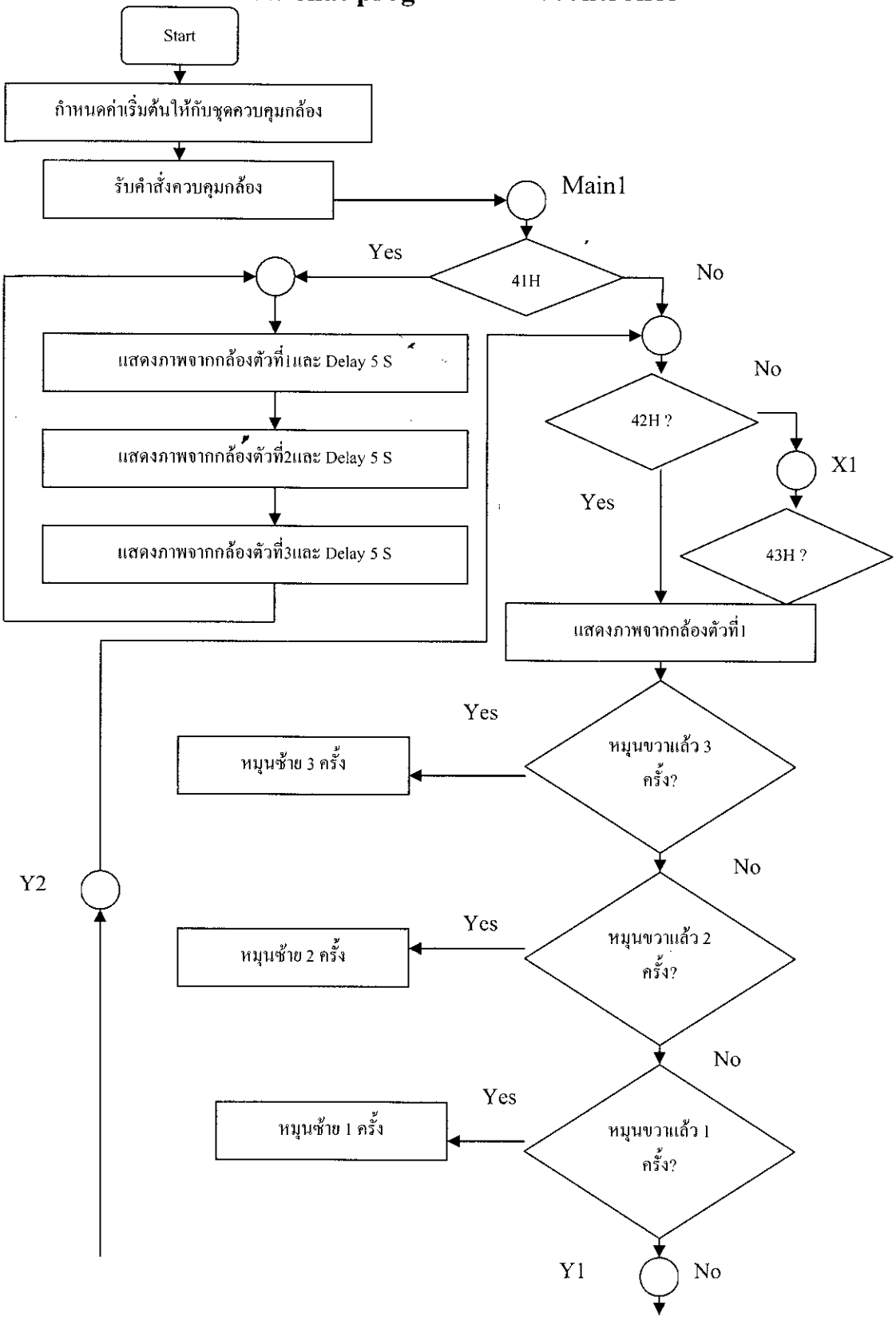
Class Frame1

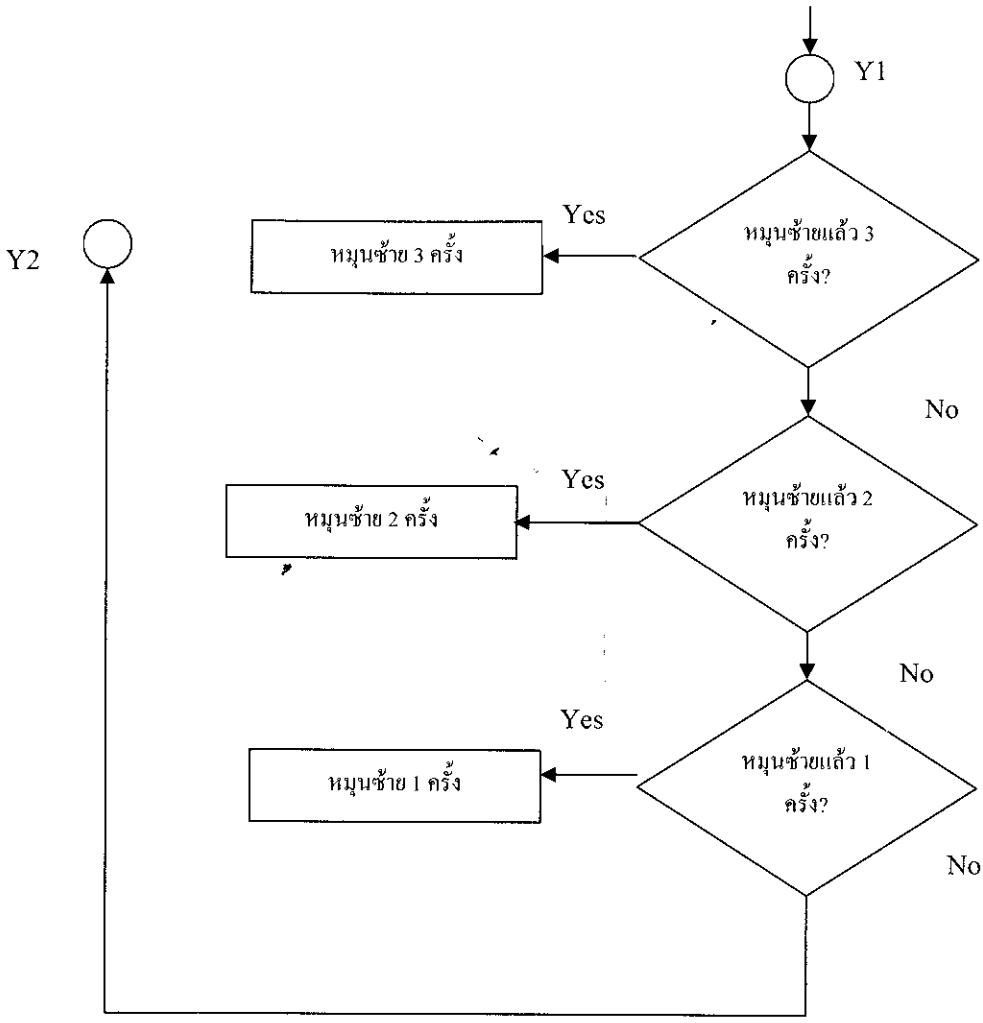


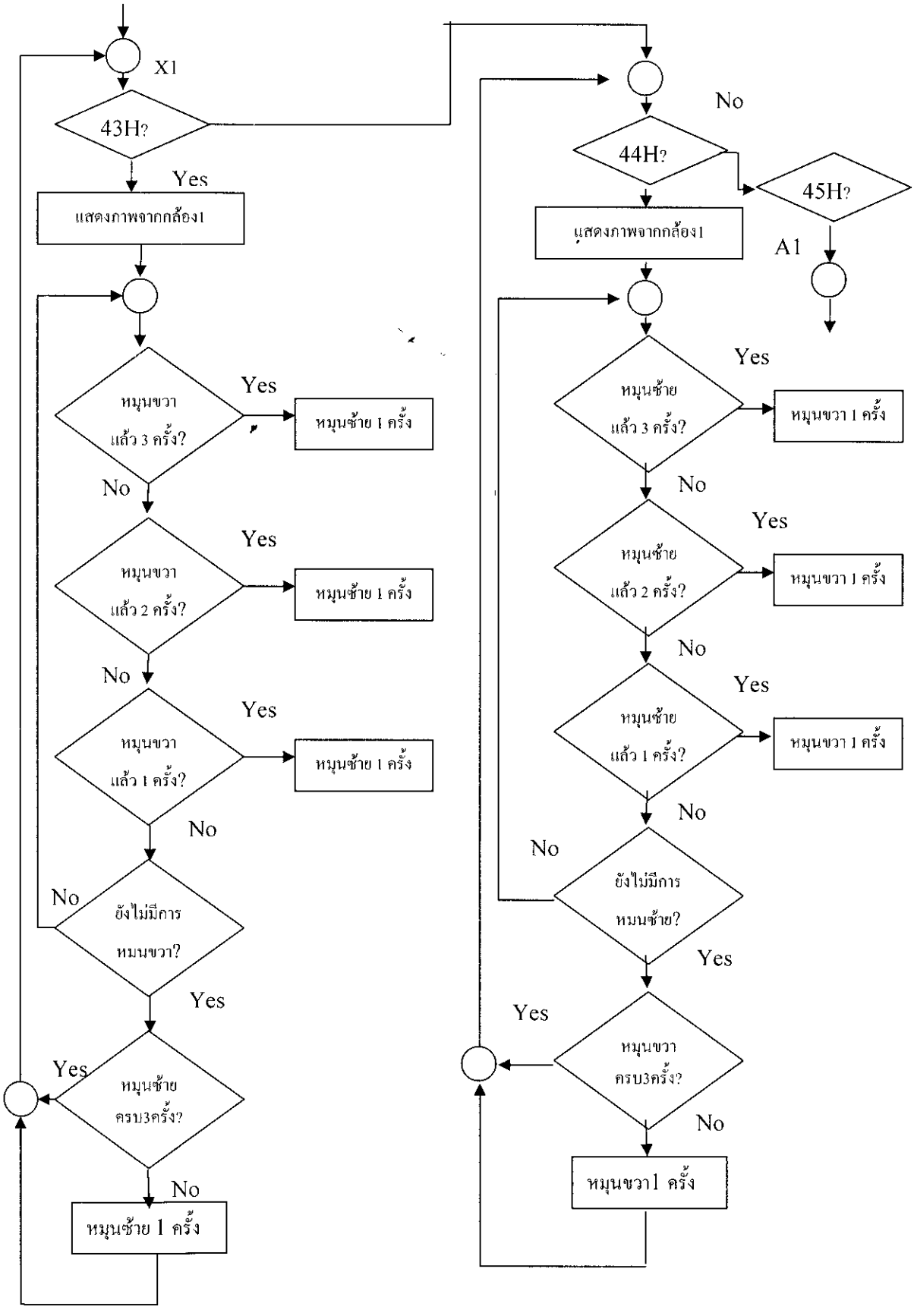
Class Connect

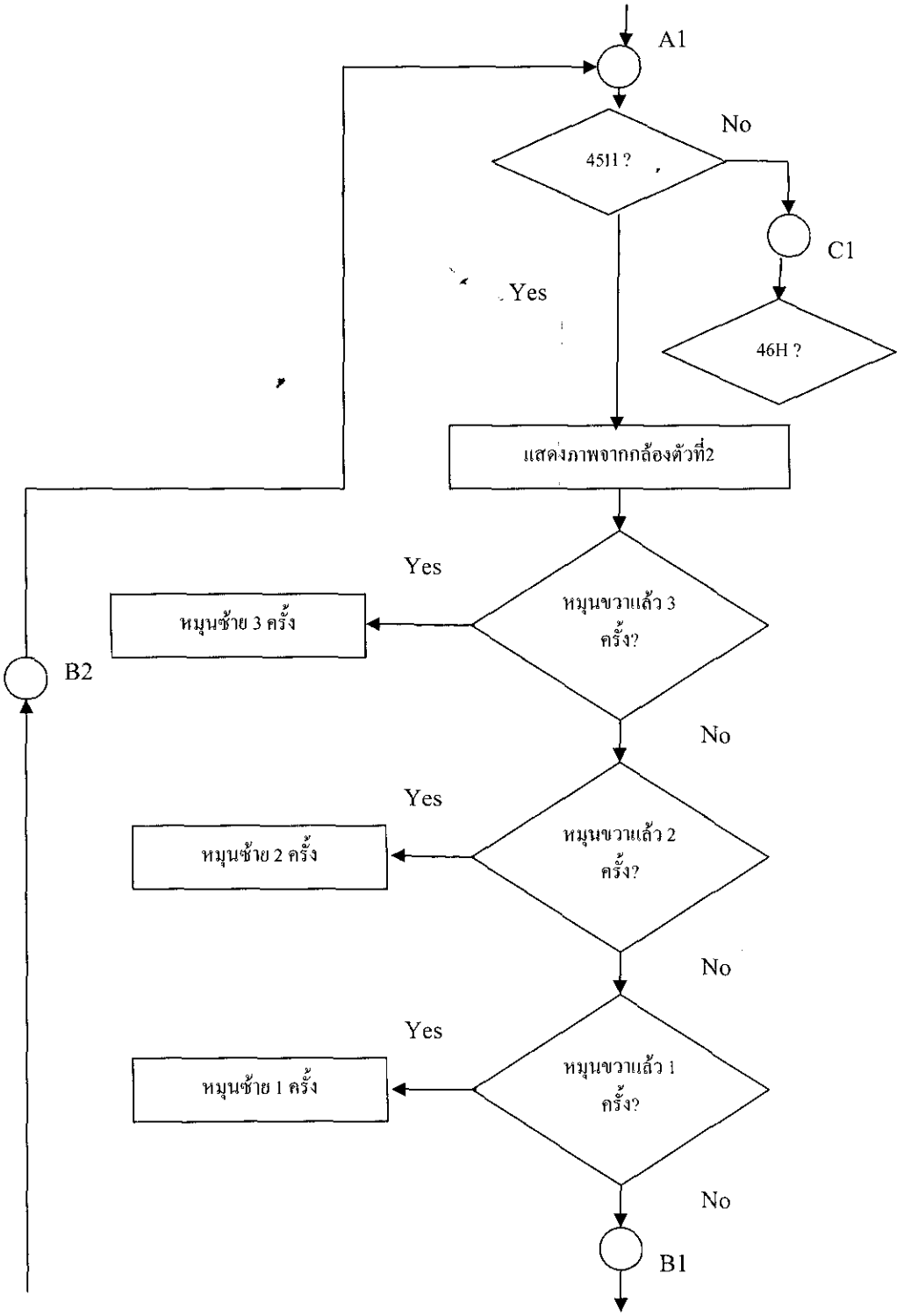


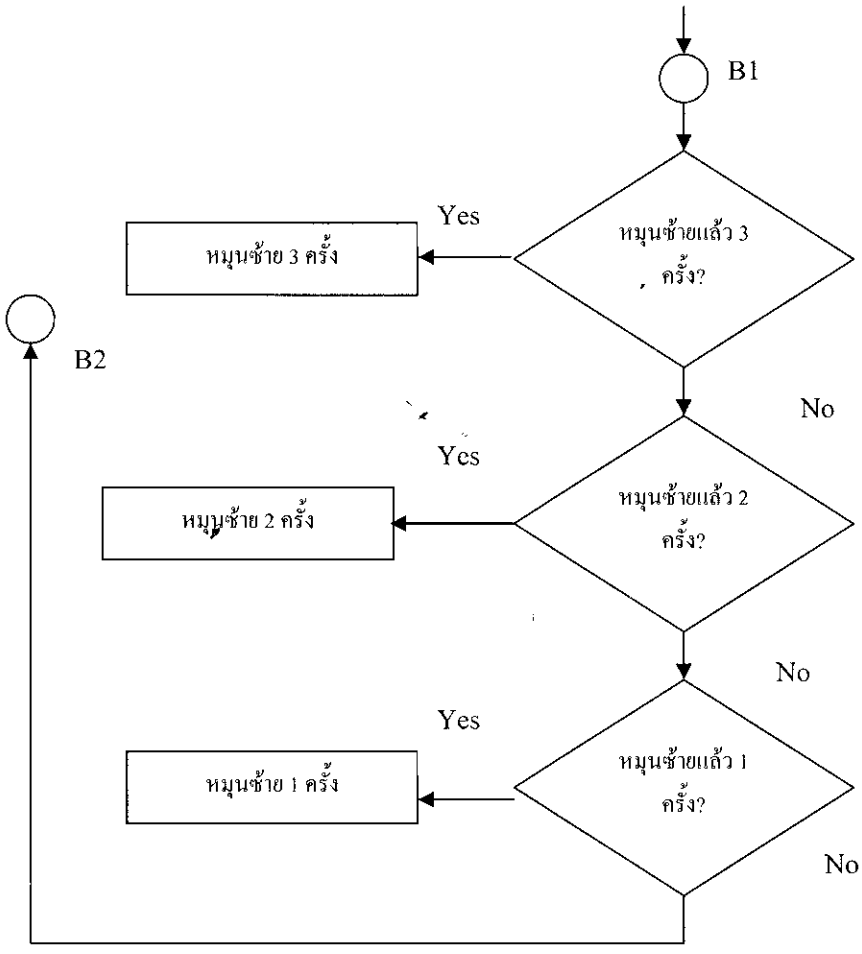
Flow chat program microcontrollor

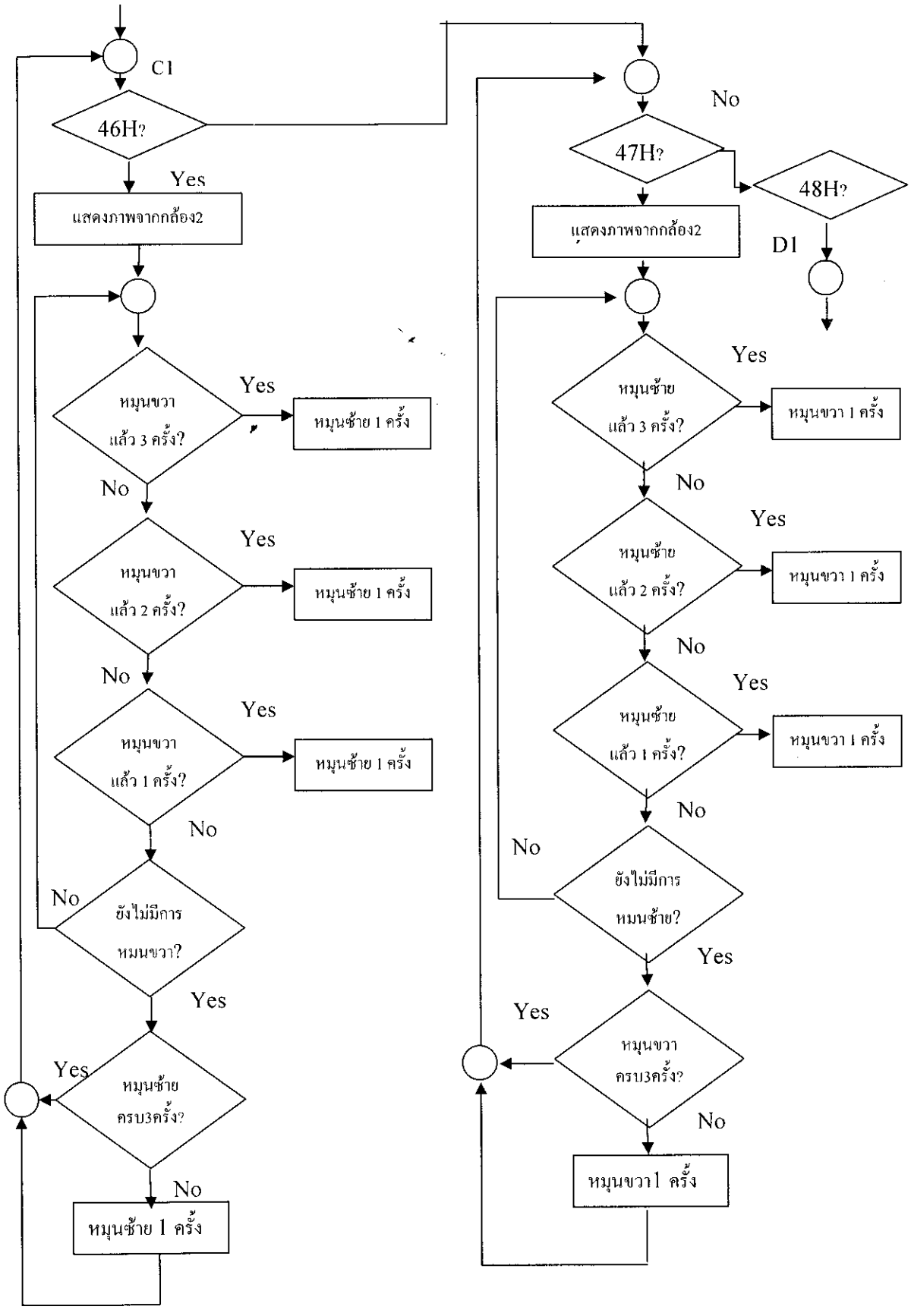


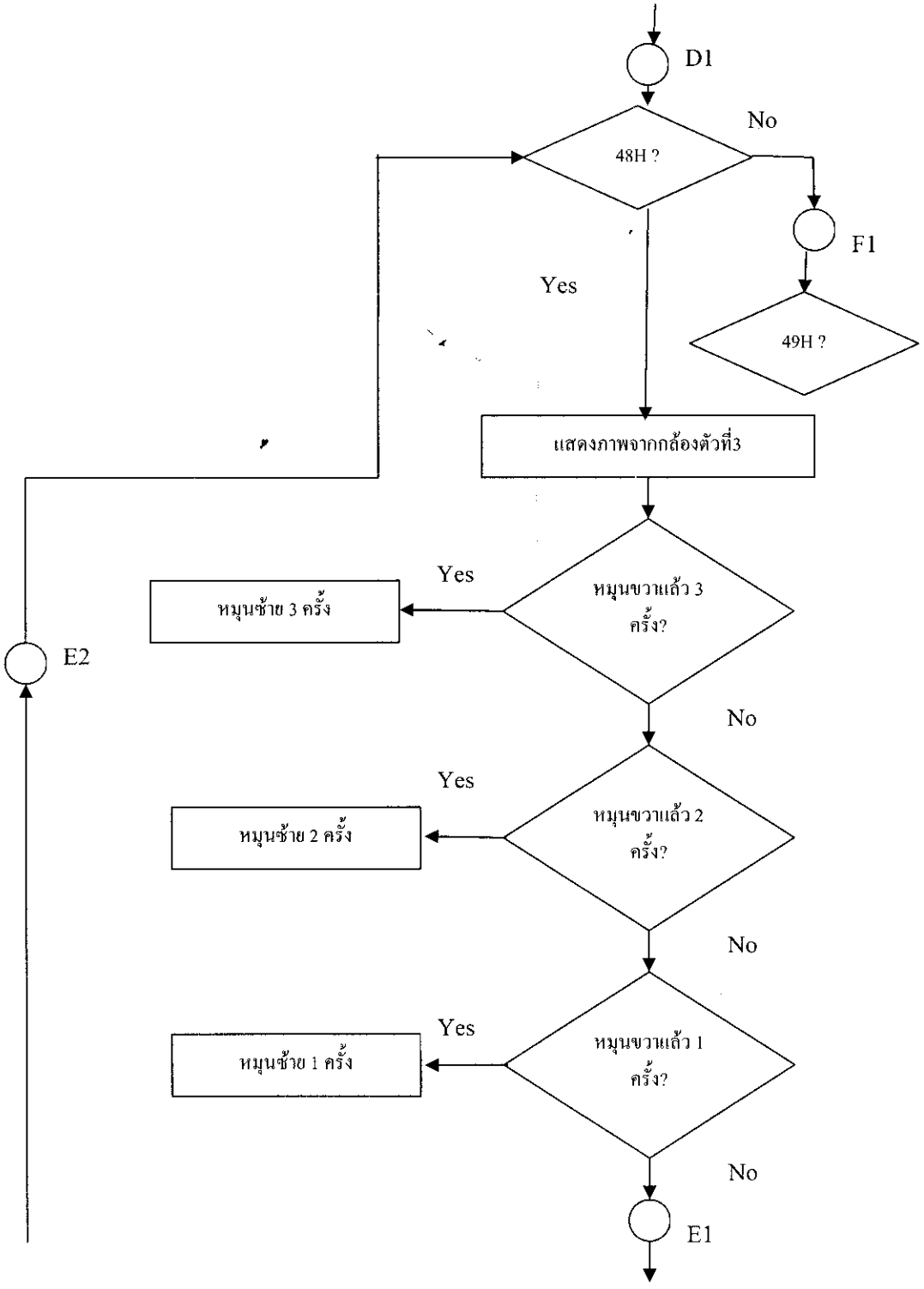

Y1
หมุนซ้ายแล้ว 3 ครั้ง?
Yes
หมุนซ้าย 3 ครั้ง
No
หมุนซ้ายแล้ว 2 ครั้ง?
Yes
หมุนซ้าย 2 ครั้ง
No
หมุนซ้ายแล้ว 1 ครั้ง?
Yes
หมุนซ้าย 1 ครั้ง
No
Y2

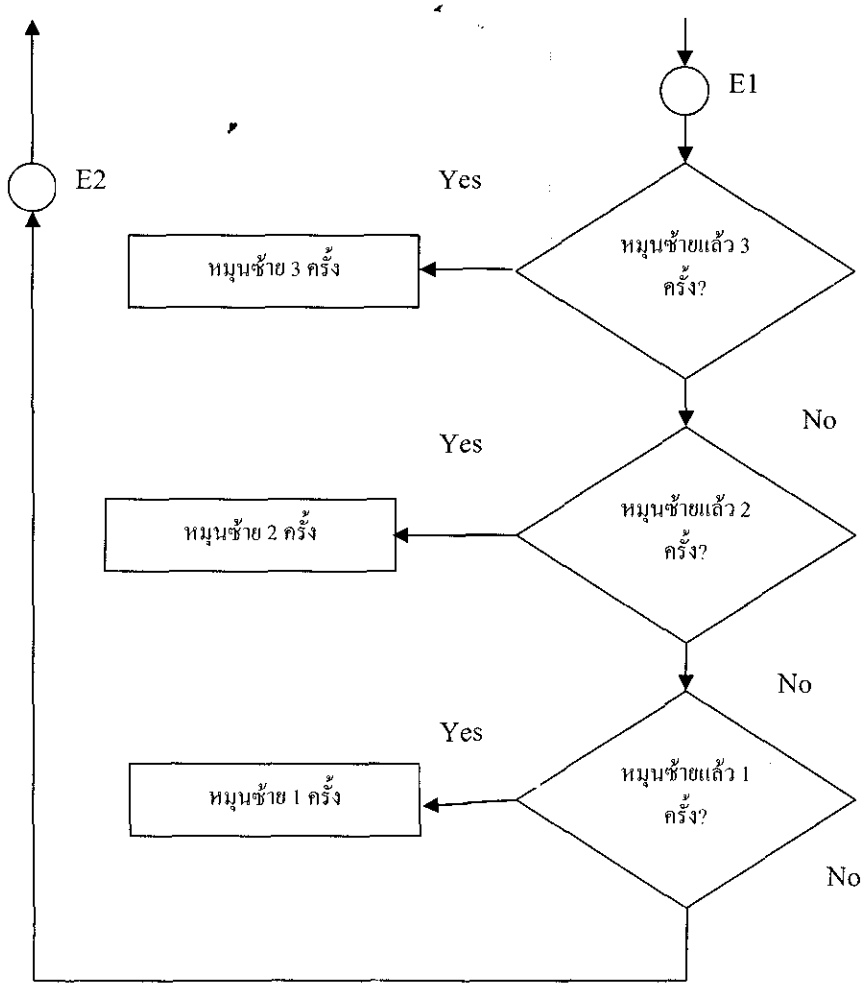

X1
43H?
Yes
แสดงภาพจากกล้อง1
หมุนขวา
แล้ว 3 ครั้ง?
Yes
หมุนซ้าย 1 ครั้ง
No
หมุนขวา
แล้ว 2 ครั้ง?
Yes
หมุนซ้าย 1 ครั้ง
No
หมุนขวา
แล้ว 1 ครั้ง?
Yes
หมุนซ้าย 1 ครั้ง
No
ยังไม่มีการ
หมนขวา?
No
Yes
หมุนซ้าย
ครบ3ครั้ง?
Yes
No
หมุนซ้าย 1 ครั้ง
No
44H?
45H?
A1
แสดงภาพจากกล้อง1
หมุนซ้าย
แล้ว 3 ครั้ง?
Yes
หมุนขวา 1 ครั้ง
No
หมุนซ้าย
แล้ว 2 ครั้ง?
Yes
หมุนขวา 1 ครั้ง
No
หมุนซ้าย
แล้ว 1 ครั้ง?
Yes
หมุนขวา 1 ครั้ง
No
ยังไม่มีการ
หมนซ้าย?
No
Yes
หมุนขวา
ครบ3ครั้ง?
Yes
No
หมุนขวา1 ครั้ง

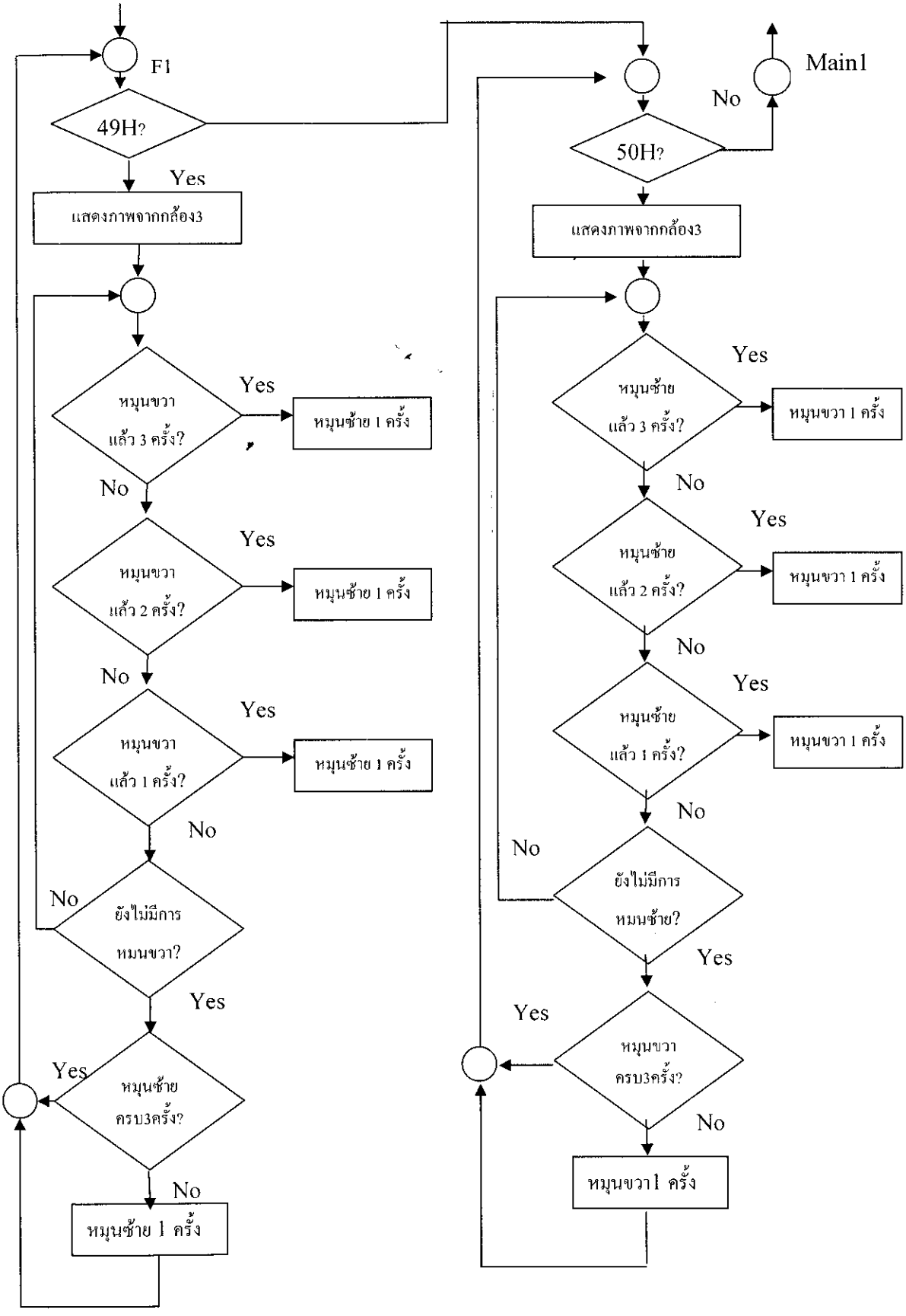

A1
4511 ?
No
C1
Yes
46H ?
แสดงภาพจากกล้องตัวที่2
Yes
หมุนขวาแล้ว 3 ครั้ง?
หมุนซ้าย 3 ครั้ง
B2
No
Yes
หมุนขวาแล้ว 2 ครั้ง?
หมุนซ้าย 2 ครั้ง
No
Yes
หมุนขวาแล้ว 1 ครั้ง?
หมุนซ้าย 1 ครั้ง
No
B1


B1
หมุนซ้ายแล้ว 3 ครั้ง?
Yes
หมุนซ้าย 3 ครั้ง
No
หมุนซ้ายแล้ว 2 ครั้ง?
Yes
หมุนซ้าย 2 ครั้ง
No
หมุนซ้ายแล้ว 1 ครั้ง?
Yes
หมุนซ้าย 1 ครั้ง
No
B2


C1
46H?
Yes
แสดงภาพจากกล้อง2
หมุนขวา แล้ว 3 ครั้ง?
Yes
หมุนซ้าย 1 ครั้ง
No
หมุนขวา แล้ว 2 ครั้ง?
Yes
หมุนซ้าย 1 ครั้ง
No
หมุนขวา แล้ว 1 ครั้ง?
Yes
หมุนซ้าย 1 ครั้ง
No
ยังไม่มีการ หมุนขวา?
No
Yes
หมุนซ้าย ครบ3ครั้ง?
Yes
No
หมุนซ้าย 1 ครั้ง
No
47H?
48H?
แสดงภาพจากกล้อง2
D1
หมุนซ้าย แล้ว 3 ครั้ง?
Yes
หมุนขวา 1 ครั้ง
No
หมุนซ้าย แล้ว 2 ครั้ง?
Yes
หมุนขวา 1 ครั้ง
No
หมุนซ้าย แล้ว 1 ครั้ง?
Yes
หมุนขวา 1 ครั้ง
No
ยังไม่มีการ หมุนซ้าย?
No
Yes
หมุนขวา ครบ3ครั้ง?
Yes
No
หมุนขวา1 ครั้ง


D1
48H ?
No
F1
49H ?
Yes
แสดงภาพจากกล้องตัวที่3
หมุนขวาแล้ว 3 ครั้ง?
Yes
หมุนซ้าย 3 ครั้ง
E2
No
หมุนขวาแล้ว 2 ครั้ง?
Yes
หมุนซ้าย 2 ครั้ง
No
หมุนขวาแล้ว 1 ครั้ง?
Yes
หมุนซ้าย 1 ครั้ง
No
E1


E1
E2
Yes
หมุนซ้ายแล้ว 3 ครั้ง?
หมุนซ้าย 3 ครั้ง
No
Yes
หมุนซ้ายแล้ว 2 ครั้ง?
หมุนซ้าย 2 ครั้ง
No
Yes
หมุนซ้ายแล้ว 1 ครั้ง?
หมุนซ้าย 1 ครั้ง
No


F1
49H?
Yes
แสดงภาพจากกล้อง3
หมุนขวา
แล้ว 3 ครั้ง?
Yes
หมุนซ้าย 1 ครั้ง
No
หมุนขวา
แล้ว 2 ครั้ง?
Yes
หมุนซ้าย 1 ครั้ง
No
หมุนขวา
แล้ว 1 ครั้ง?
Yes
หมุนซ้าย 1 ครั้ง
No
ยังไม่มีการ
หมุนขวา?
No
Yes
หมุนซ้าย
ครบ3ครั้ง?
Yes
No
หมุนซ้าย 1 ครั้ง
Main1
No
50H?
แสดงภาพจากกล้อง3
หมุนซ้าย
แล้ว 3 ครั้ง?
Yes
หมุนขวา 1 ครั้ง
No
หมุนซ้าย
แล้ว 2 ครั้ง?
Yes
หมุนขวา 1 ครั้ง
No
หมุนซ้าย
แล้ว 1 ครั้ง?
Yes
หมุนขวา 1 ครั้ง
No
ยังไม่มีการ
หมุนซ้าย?
No
Yes
หมุนขวา
ครบ3ครั้ง?
Yes
No
หมุนขวา1 ครั้ง

Source code package serverclass

class MainClass

```
package serverclass;

import java.util.*;

public class MainClass {
  ClientCamSender camSender;
  ClientReciveControl reciveControl;
  ClientControlSender controlSender;
  ClientReciveImage reciveImage;

 //Construct the application
 public MainClass() {
    System.out.println("Attemting to Start Server.");
    reciveImage=new ClientReciveImage();
    Timer t1 =new Timer();
    t1.scheduleAtFixedRate(reciveImage,100,100);
    controlSender=new ClientControlSender();
    Timer t2=new Timer();
    t2.scheduleAtFixedRate(controlSender,50,50);
    camSender=new ClientCamSender(reciveImage);
    Timer t3=new Timer();
    t3.scheduleAtFixedRate(camSender,100,100);
    reciveControl=new ClientReciveControl(controlSender);
    reciveControl.start();
  }

  //Main method
  public static void main(String[] args) {
    new MainClass();
  }
}
```

class ClientReciveControl

```
package serverclass;

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

public class ClientReciveControl extends Thread{
  private Socket socket;
  private BufferedReader inMsg;
  private String message;
```

```
private ServerSocket serverSocket;
private ClientControlSender controlSender;

public ClientReciveControl(ClientControlSender controlSender) {
 this.controlSender=controlSender;
 try {
  serverSocket = new ServerSocket(7003);
  socket = serverSocket.accept();
 }
 catch (IOException ex) {
  ex.printStackTrace();
 }
}

public String getMassage() {
 return message;
}

public void run() {
 try {
  inMsg = new BufferedReader(new InputStreamReader(socket.getInputStream()));
 }
 catch (IOException ex) {
  ex.printStackTrace();
 }
 while (true){
 try {
  char charBuffer[] = new char[1];

  while (inMsg.read(charBuffer, 0, 1) != -1) {
   StringBuffer stringBuffer = new StringBuffer(1024);

   while (charBuffer[0] != '\0') {
    stringBuffer.append(charBuffer[0]);
    inMsg.read(charBuffer, 0, 1);
   }
   message = stringBuffer.toString();
   controlSender.send(message);
  }
 }
 catch (Exception e) {
  try {
   serverSocket.close();
   socket.close();
   inMsg.close();
   serverSocket = new ServerSocket(7003);
   socket = serverSocket.accept();
```

```
      this.run();
     }
     catch (IOException ex1) {
      }
     }
    }
  }
}
```

class ClientCamSender

```
package serverclass;

import java.net.Socket;
import java.io.OutputStream;
import java.awt.Image;
import java.awt.image.PixelGrabber;
import java.io.IOException;
import java.io.ObjectOutputStream;
import java.net.ServerSocket;
import java.util.TimerTask;

public class ClientCamSender extends TimerTask{
  Socket socket;
  OutputStream outputStream;
  ServerSocket serverSocket;
  ClientReciveImage clientReciveImage;

  public ClientCamSender(ClientReciveImage clientReciveImage) {
    this.clientReciveImage=clientReciveImage;
    try {
      serverSocket = new ServerSocket(7002);
    }
    catch (IOException ex) {
      ex.printStackTrace();
    }
  }

  public int [] grabPixels(Image img){
    int width = img.getWidth(null);
    int height = img.getHeight(null);
    int [] pixels = new int[width*height];
    PixelGrabber grabber = new PixelGrabber(img,0,0,width,height,pixels,0,width);
    try{
      if (!grabber.grabPixels())
        return null;
    }
    catch (InterruptedException e){
```

```
    return null;
   }
   return pixels;
  }

  public void writeImage(Image img, OutputStream out) throws IOException{
   int [] pixelValues = grabPixels(img);
   ObjectOutputStream objectOut = new ObjectOutputStream(out);
   objectOut.writeObject(pixelValues);
   objectOut.flush();
  }

  public void run(){
   try {
    socket = serverSocket.accept();
    outputStream=socket.getOutputStream();
    writeImage(clientReciveImage.getImage(),outputStream);
   }
   catch (IOException ex) {
      try {
          serverSocket.close();
          outputStream.close();
          socket.close();
          serverSocket = new ServerSocket(7002);
          this.run();
      }
      catch (IOException ex2) {
      }
   }

   try {
     outputStream.close();
     socket.close();
   }
   catch (IOException ex1) {
    ex1.printStackTrace();
   }
  }
}
```

class ClientControlSender

```
package serverclass;

import java.io.PrintWriter;
import java.net.Socket;
import java.net.ServerSocket;
import java.io.IOException;
```

```
import java.util.TimerTask;

public class ClientControlSender extends TimerTask{
  private Socket socket;
  private PrintWriter outMsg;
  private ServerSocket serverSocket;

  public ClientControlSender() {
    try {
      serverSocket = new ServerSocket(7004);
    }
    catch (IOException ex) {
    }
  }
  public void run() {
    try {
      socket = serverSocket.accept();
      outMsg = new PrintWriter(socket.getOutputStream());
    }
    catch (IOException ex) {
       try {
         serverSocket.close();
         socket.close();
         outMsg.close();
         serverSocket = new ServerSocket(7004);
         this.run();
       }
       catch (IOException ex1) {
       }
     }
    }

 public void send(String Message) {
       outMsg.print(Message);
       outMsg.print("\0");
       if (outMsg.checkError()) {
         System.out.println("Send Message Error!");
       }
    }
}
```

class ClientReciveImage

```
package serverclass;

import java.net.Socket;
import java.io.InputStream;
import java.io.BufferedInputStream;
```

```
import java.net.ServerSocket;
import java.awt.Image;
import java.awt.image.MemoryImageSource;
import java.awt.Toolkit;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.util.TimerTask;

public class ClientReciveImage extends TimerTask{
 Socket socket;
 InputStream in;
 BufferedInputStream bufferStream;
 ServerSocket serverSocket;
 private Image image;

 public ClientReciveImage() {
  try {
   serverSocket = new ServerSocket(7001);
  }
  catch (IOException ex) {
   ex.printStackTrace();
  }
 }

 public Image createImage(int [] pixels, int width, int height){
  MemoryImageSource imageSource = new
MemoryImageSource(width,height,pixels,0,width);
  return Toolkit.getDefaultToolkit().createImage(imageSource);
 }

 public Image readImage(InputStream in) throws IOException{
  int [] pixelValues=null;
  ObjectInputStream objectIn = new ObjectInputStream(in);
  try{
   pixelValues = (int [])objectIn.readObject();
  }
  catch (ClassNotFoundException e){
  }
  return createImage(pixelValues,320,240);
 }

 public Image getImage(){
  return image;
 }

 public void run() {
  try {
```

```
    socket = serverSocket.accept();
    in = socket.getInputStream();
    image=readImage(in);
   }
   catch (IOException ex) {
       try {
        serverSocket.close();
        socket.close();
        serverSocket = new ServerSocket(7001);
        this.run();
        }
        catch (IOException ex2) {
       }
   }
   try {
    socket.close();
    in.close();
   }
   catch (IOException ex1) {
      try {
        serverSocket.close();
        socket.close();
        serverSocket = new ServerSocket(7001);
        this.run();
       }
       catch (IOException ex2) {
       }
   }
  }
}
```

Source code package imagereceive

class Imagereceive

```
package imagereceive;

import java.applet.*;
import java.util.Timer;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.net.Socket;
import javax.swing.*;
import java.io.OutputStream;
import java.awt.event.ActionEvent;
import java.io.PrintWriter;

public class Imagereceive extends Applet  {
```

```
private boolean isStandalone = false;
Connect connect;
Frame1 frame;
JPanel jPanel = new JPanel();

//Get a parameter value
public String getParameter(String key, String def) {
  return isStandalone ? System.getProperty(key, def) :
    (getParameter(key) != null ? getParameter(key) : def);
}

//Construct the applet
public Imagereceive() {
}

//Initialize the applet
public void init() {
  try {
    jbInit();
  }
  catch(Exception e) {
    e.printStackTrace();
  }
}

//Component initialization
private void jbInit() throws Exception {
  this.add(jPanel, BorderLayout.SOUTH);
  frame =new Frame1(jPanel);
  connect=new Connect(frame);
  Timer t=new Timer();
  t.schedule(connect,500,1000);
}

//Get Applet information
public String getAppletInfo() {
  return "Applet Information";
}

//Get parameter info
public String[][] getParameterInfo() {
  return null;
}
}
```

class Frame1

```
package imagereceive;
```

```
import javax.swing.*;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.awt.*;

public class Frame1 extends JFrame {
  ImagePanel imagePanel;
  JPanel jPanel;
  public Frame1(JPanel jPanel) {
    this.jPanel=jPanel;
    imagePanel=new ImagePanel();
    try {
      jbInit();
    }
    catch(Exception e) {
      e.printStackTrace();
    }
  }

  public void setImage(Image img){
    imagePanel.setImage(img);
  }

 public  class ImagePanel extends Panel {
  public Image myimg = null;

  public ImagePanel() {
    setLayout(null);
    setSize(320, 240);
  }

  public void setImage(Image img) {
    this.myimg = img;
    repaint();
  }

  public void paint(Graphics g) {
    if (myimg != null) {
      g.drawImage(myimg, 0, 0, this);
    }
  }
}
  private void jbInit() throws Exception {
    jPanel.setSize(320, 240);
    this.setSize(new Dimension(362, 278));
    jPanel.add(imagePanel);
```

```
  }
}
```

class Connect

```
package imagereceive;

import java.util.*;
import java.awt.*;
import java.awt.image.*;
import java.io.*;
import java.net.*;
import com.sun.image.codec.jpeg.*;
import javax.swing.*;

public class Connect extends TimerTask{
  Socket socket;
  InputStream in;
  BufferedInputStream bufferStream;
  private Frame1 frame;
  Image img;

  public Connect(Frame1 frame) {
   this.frame=frame;
  }

  public static Image createImage(int [] pixels, int width, int height){
    MemoryImageSource imageSource = new
MemoryImageSource(width,height,pixels,0,width);
    return Toolkit.getDefaultToolkit().createImage(imageSource);
  }

  public static Image readImage(InputStream in) throws IOException{
    int [] pixelValues=null;
    ObjectInputStream objectIn = new ObjectInputStream(in);
    try{
      pixelValues = (int [])objectIn.readObject();
    }
    catch (ClassNotFoundException e){
    }
    return createImage(pixelValues,320, 240);

  }

  public void run() {
    try {
      while (true) {
        socket = new Socket("192.168.0.1",7002);
```

```
      in = socket.getInputStream();
      try {
        img = readImage(in);
        frame.setImage(img);
      }
      catch (IOException ex) {
        ex.printStackTrace();
      }
      in.close();
      socket.close();
    }
   }
   catch (Exception ex1) {
     ex1.printStackTrace();
   }
  }
}
```

class Clientcontrolsender

```
package imagereceive;

import java.awt.*;
import java.awt.event.*;
import java.applet.*;
import java.net.Socket;
import javax.swing.*;
import java.io.OutputStream;
import java.awt.event.ActionEvent;
import java.io.PrintWriter;
import com.borland.dbswing.*;

public class Clientcontrolsender extends Applet implements ActionListener{
  private boolean isStandalone = false;
  public Socket socket;
  private PrintWriter outMsg;
  JButton CAM1_CENTER = new JButton();
  JButton CAM1_LEFT = new JButton();
  JButton CAM1_RIGHT = new JButton();
  JTextArea jTextArea1 = new JTextArea();
  JTextArea jTextArea2 = new JTextArea();
  JButton CAM2_LEFT = new JButton();
  JButton CAM2_CENTER = new JButton();
  JButton CAM2_RIGHT = new JButton();
  JTextArea jTextArea3 = new JTextArea();
  JButton CAM3_LEFT = new JButton();
  JButton CAM3_CENTER = new JButton();
  JButton CAM3_RIGHT = new JButton();
```

```
JButton LOOP = new JButton();
JButton Capture = new JButton();

//Get a parameter value
public String getParameter(String key, String def) {
 return isStandalone ? System.getProperty(key, def) :
    (getParameter(key) != null ? getParameter(key) : def);
}

//Construct the applet
public Clientcontrolsender() {
}

//Initialize the applet
public void init() {
 try {
  jbInit();
 }
 catch (Exception e) {
  e.printStackTrace();
 }
}

//Component initialization
private void jbInit() throws Exception {
 CAM1_LEFT.setVerifyInputWhenFocusTarget(true);
 CAM1_LEFT.setActionCommand("CAM1_LEFT");
 CAM1_LEFT.setText("LEFT");
 CAM1_RIGHT.setActionCommand("CAM1_RIGHT");
 CAM1_RIGHT.setText("RIGHT");
 CAM1_CENTER.setActionCommand("CAM1_CENTER");
 CAM1_CENTER.setText("CENTER");
 this.setBackground(SystemColor.control);
 this.setEnabled(true);
 this.setFont(new java.awt.Font("Dialog", 1, 14));
 jTextArea1.setBackground(UIManager.getColor("info"));
 jTextArea1.setText("CAMERA1");
 jTextArea2.setBackground(UIManager.getColor("info"));
 jTextArea2.setText("CAMERA2");
 CAM2_LEFT.setActionCommand("CAM2_LEFT");
 CAM2_LEFT.setText("LEFT");
 CAM2_CENTER.setActionCommand("CAM2_CENTER");
 CAM2_CENTER.setRolloverEnabled(false);
 CAM2_CENTER.setText("CENTER");
 CAM2_RIGHT.setActionCommand("CAM2_RIGHT");
 CAM2_RIGHT.setText("RIGHT");
 jTextArea3.setBackground(UIManager.getColor("info"));
```

```
jTextArea3.setText("CAMERA3");
CAM3_LEFT.setActionCommand("CAM3_LEFT");
CAM3_LEFT.setText("LEFT");
CAM3_CENTER.setActionCommand("CAM3_CENTER");
CAM3_CENTER.setText("CENTER");
CAM3_RIGHT.setVerifyInputWhenFocusTarget(true);
CAM3_RIGHT.setActionCommand("CAM3_RIGHT");
CAM3_RIGHT.setText("RIGHT");
LOOP.setText("LOOP");
Capture.setActionCommand("Capture");
Capture.setText("Save Image");
this.add(jTextArea1, null);
this.add(CAM1_LEFT, null);
this.add(CAM1_CENTER, null);
this.add(CAM1_RIGHT, null);
this.add(jTextArea2, null);
this.add(CAM2_LEFT, null);
this.add(CAM2_CENTER, null);
this.add(CAM2_RIGHT, null);
this.add(jTextArea3, null);
this.add(CAM3_LEFT, null);
this.add(CAM3_CENTER, null);
this.add(CAM3_RIGHT, null);
this.add(LOOP, null);
this.add(Capture, null);
CAM1_CENTER.addActionListener(this);
CAM1_LEFT.addActionListener(this);
CAM1_RIGHT.addActionListener(this);
CAM2_CENTER.addActionListener(this);
CAM2_LEFT.addActionListener(this);
CAM2_RIGHT.addActionListener(this);
CAM3_CENTER.addActionListener(this);
CAM3_LEFT.addActionListener(this);
CAM3_RIGHT.addActionListener(this);
LOOP.addActionListener(this);
Capture.addActionListener(this);

socket = new Socket("192.168.0.1", 7003);
outMsg = new PrintWriter(socket.getOutputStream());
}
//Get Applet information
public String getAppletInfo() {
 return "Applet Information";
}

//Get parameter info
public String[][] getParameterInfo() {
```

```
  return null;
 }

 public void actionPerformed(ActionEvent e) {
  if (e.getActionCommand().endsWith("CAM1_LEFT")) {
   send("C");
  }
  if (e.getActionCommand().endsWith("CAM1_CENTER")) {
   send("B");
  }
  if (e.getActionCommand().endsWith("CAM1_RIGHT")) {
   send("D");
  }
  if (e.getActionCommand().endsWith("CAM2_LEFT")) {
   send("F");
  }
  if (e.getActionCommand().endsWith("CAM2_CENTER")) {
   send("E");
  }
  if (e.getActionCommand().endsWith("CAM2_RIGHT")) {
   send("G");
  }
  if (e.getActionCommand().endsWith("CAM3_LEFT")) {
   send("I");
  }
  if (e.getActionCommand().endsWith("CAM3_CENTER")) {
   send("H");
  }
  if (e.getActionCommand().endsWith("CAM3_RIGHT")) {
   send("P");
  }
  if (e.getActionCommand().endsWith("LOOP")) {
   send("A");
  }
  if (e.getActionCommand().endsWith("Capture")) {
     send("S");
  }
 }

 private void send(String string) {
  outMsg.print(string);
  outMsg.print("\0");
  if(outMsg.checkError()){
   System.out.println("Message Error");
  }
 }
}
```

Source code package clientcamsender

class clientcamsender

```
package clientcamsender;

import java.awt.*;
import java.awt.event.*;
import java.applet.*;
import java.util.Timer;
import java.net.Socket;
import java.io.BufferedReader;
import java.io.IOException;
import javax.swing.*;

public class clientcamsender extends Applet {
  private boolean isStandalone = false;
  Timer t,t2;
  ClientSocket soc;
  ClientChat chat;
  Socket socket;
  BufferedReader inMsg;
  JTextField jTextField1 = new JTextField();
  JLabel jLabel1 = new JLabel();
  private ClientCam cam;

  public String getParameter(String key, String def) {
    return isStandalone ? System.getProperty(key, def) :
      (getParameter(key) != null ? getParameter(key) : def);
  }

  public clientcamsender() {
  }
  public void init() {
    try {
      jbInit();
    }
    catch(Exception e) {
      e.printStackTrace();
    }
    chat.start();
  }
  private void jbInit() throws Exception {
    this.setSize(550,400);
    cam=new ClientCam();
    soc=new ClientSocket(cam);
    t2=new Timer();
    t=new Timer();
    t.schedule(soc,500,250);
```

```
  this.add(cam);
  chat=new ClientChat(jTextField1,cam);
  socket=chat.getSocket();
  inMsg=chat.getBufferedReader();
  jLabel1.setText("Massage Control :");
  jTextField1.setText("                ");
  this.setEnabled(true);
  this.add(jLabel1,BorderLayout.CENTER);
  this.add(jTextField1,BorderLayout.CENTER);
 }

 //Get Applet information
 public String getAppletInfo() {
    return "Applet Information";
 }

 //Get parameter info
 public String[][] getParameterInfo() {
    return null;
 }

 public void start(){
 }

 public void stop(){
 }

 public void destroy(){
  cam.closed();
  try {
   socket.close();
   inMsg.close();
  }
  catch (IOException ex) {
   ex.printStackTrace();
  }
 }
}
```

class ClientSocket

```
package clientcamsender;

import java.net.*;
import java.io.*;
import java.io.DataOutputStream;
import java.io.DataInputStream;
import java.io.OutputStream;
```

```
import java.awt.Image;
import javax.media.Buffer;
import java.awt.image.BufferedImage;
import java.awt.image.PixelGrabber;
import java.util.TimerTask;

public class ClientSocket extends TimerTask{
 Socket socket;
 OutputStream outputStream;
 ClientCam cam;
 public ClientSocket(ClientCam cam) {
  this.cam=cam;
 }

 public int [] grabPixels(Image img){
  int width = img.getWidth(null);
  int height = img.getHeight(null);
  int [] pixels = new int[width*height];
  PixelGrabber grabber = new PixelGrabber(img,0,0,width,height,pixels,0,width);
  try{
   if (!grabber.grabPixels())
    return null;
  }
  catch (InterruptedException e){
   return null;
  }
  return pixels;
 }

 public void writeImage(Image img, OutputStream out) throws IOException{
  int [] pixelValues = grabPixels(img);
  ObjectOutputStream objectOut = new ObjectOutputStream(out);
  objectOut.writeObject(pixelValues);
  objectOut.flush();
 }

 public void run() {
  try {
   Thread.sleep(200);
  }
  catch (InterruptedException ex1) {
   ex1.printStackTrace();
  }
  try {
   socket = new Socket("192.168.0.1", 7001);
   outputStream = socket.getOutputStream();
   writeImage(cam.capture(), outputStream);
```

```
    }
    catch (IOException ex) {
      ex.printStackTrace();
    }
    try {
      outputStream.close();
      socket.close();
    }
    catch (IOException ex2) {
      ex2.printStackTrace();
    }
  }
}
```

class ClientCam

```
package clientcamsender;

import javax.swing.*;
import javax.swing.event.*;
import java.io.*;
import javax.media.*;
import javax.media.format.*;
import javax.media.util.*;
import javax.media.control.*;
import javax.media.protocol.*;
import java.util.*;
import java.awt.*;
import java.awt.image.*;
import java.awt.event.*;
import com.sun.image.codec.jpeg.*;

public class ClientCam extends Panel{
  private Player player = null;
  private CaptureDeviceInfo di = null;
  private MediaLocator ml = null;
  private Buffer buf = null;
  public Image img = null;
  private VideoFormat vf = null;
  private BufferToImage btoi = null;
  int tce=0;
  String s = "C:\\SaveImage\\Image";

  public ClientCam() {
    setLayout(new BorderLayout());
    setSize(320, 240);

    String str1 = "vfw:Microsoft WDM Image Capture (Win32):0";
```

```
  di = CaptureDeviceManager.getDevice(str1);
  ml = di.getLocator();

  try {
   player = Manager.createRealizedPlayer(ml);
   player.start();
   Component comp;

   if ( (comp = player.getVisualComponent()) != null) {
    add(comp, BorderLayout.CENTER);
   }
  }
  catch (Exception e) {
   e.printStackTrace();
  }
 }

 public void saveJPG(Image img) {
  BufferedImage bi = new BufferedImage(img.getWidth(null), img.getHeight(null),
                          BufferedImage.TYPE_INT_RGB);
  Graphics2D g2 = bi.createGraphics();
  g2.drawImage(img, null, null);

  FileOutputStream out = null;
  try {
     tce=tce+1;
     out = new FileOutputStream(s.toString()+tce+".jpg");
  }
  catch (java.io.FileNotFoundException io) {
  System.out.println("File Not Found");
  }

  JPEGImageEncoder encoder = JPEGCodec.createJPEGEncoder(out);
  JPEGEncodeParam param = encoder.getDefaultJPEGEncodeParam(bi);
  param.setQuality(0.5f, false);
  encoder.setJPEGEncodeParam(param);

  try {
   encoder.encode(bi);
   out.close();
  }
  catch (java.io.IOException io) {
    System.out.println("IOException");
  }
}

 public Image capture() {
```

```
// Grab a frame
   FrameGrabbingControl fgc = (FrameGrabbingControl)
      player.getControl("javax.media.control.FrameGrabbingControl");
   buf = fgc.grabFrame();

// Convert it to an image
   btoi = new BufferToImage( (VideoFormat) buf.getFormat());
   img = btoi.createImage(buf);
   return img;
  }

  public void closed(){
   player.close();
  }
}
```

class ClientChat

```
package clientcamsender;

import java.util.*;
import java.net.*;
import java.io.*;
import javax.swing.JTextField;

public class ClientChat extends Thread{
  private Socket socket;
  private BufferedReader inMsg;
  ClientCam cam;
  WriterRS232 writeSerial;
  private String message;

  public ClientChat(JTextField jTextField1,ClientCam cam) {
    this.cam=cam;
    try {
      socket = new Socket("192.168.0.1", 7004);
      inMsg=new BufferedReader(new InputStreamReader(socket.getInputStream()));
      writeSerial =new WriterRS232(jTextField1,cam);
    }
    catch (IOException ex) {
      ex.printStackTrace();
    }
  }

  public Socket getSocket(){
    return socket;
  }
```

```
 public BufferedReader getBufferedReader(){
  return inMsg;
 }

 public String getMassage(){
  return message;
 }

 public void run() {
  while(true){
  try{
   char charBuffer[]=new char[1];

   while(inMsg.read(charBuffer,0,1)!=-1){
    StringBuffer stringBuffer = new StringBuffer(1024);

    while(charBuffer[0] !='\0'){
     stringBuffer.append(charBuffer[0]);
     inMsg.read(charBuffer,0,1);
    }
    message=stringBuffer.toString();
    writeSerial.write(message);
   }
  }
  catch(Exception e){
   e.printStackTrace();
   }
  }
 }
}
```

class WriterRS232

```
package clientcamsender;

import javax.swing.JTextField;
import java.io.IOException;
import javax.comm.*;
import java.io.*;
import java.util.*;

public class WriterRS232 {
 JTextField jTextField1;
 Enumeration portList;
 CommPortIdentifier portId;
 SerialPort serialPort;
 OutputStream outputStream;
 ClientCam cam;
```

```
boolean outputBufferEmptyFlag = false;

public WriterRS232(JTextField jTextField1,ClientCam cam) {
  this.cam=cam;
  this.jTextField1 = jTextField1;
  boolean portFound = false;
  String defaultPort = "COM2";
  portList = CommPortIdentifier.getPortIdentifiers();
if (portList.hasMoreElements()) {
   portId = (CommPortIdentifier) portList.nextElement();

  if (portId.getPortType() == CommPortIdentifier.PORT_SERIAL) {

    if (portId.getName().equals(defaultPort)) {
      portFound = true;

    try {
     serialPort = (SerialPort) portId.open("SimpleWrite", 2000);
    }
    catch (PortInUseException e) {
     System.out.println("Port in use.");
     System.exit(1);
    }

    try {
     outputStream = serialPort.getOutputStream();
    }
    catch (IOException e) {}

    try {
     serialPort.setSerialPortParams(9600,
                        SerialPort.DATABITS_8,
                        SerialPort.STOPBITS_1,
                        SerialPort.PARITY_NONE);
    }
    catch (UnsupportedCommOperationException e) {
     e.printStackTrace();
    }

    try {
     serialPort.notifyOnOutputEmpty(true);
    }
    catch (Exception e) {
     System.out.println("Error setting event notification");
     System.out.println(e.toString());
     System.exit( -1);
     }
```

```
    }
   }
  }
}

  //write message
  public void write(String message) {
   jTextField1.setText(message);
   if(message.endsWith("S")){
    cam.saveJPG(cam.capture());
   }

   try {
     outputStream.write(message.getBytes());
   }
   catch (IOException e) {
     e.printStackTrace();
   }
  }
}
```

class Clientreceivecontrol

```
package clientcamsender;

import java.awt.*;
import java.awt.event.*;
import java.applet.*;
import java.net.*;
import java.io.BufferedReader;
import java.io.*;
import javax.swing.*;

public class Clientreceivecontrol extends Applet {
  private boolean isStandalone = false;
  ClientChat chat;
  Socket socket;
  BufferedReader inMsg;
  JTextField jTextField1 = new JTextField();
  JLabel jLabel1 = new JLabel();
  ClientCam cam=null;

  //Get a parameter value
  public String getParameter(String key, String def) {
   return isStandalone ? System.getProperty(key, def) :
     (getParameter(key) != null ? getParameter(key) : def);
  }
```

```
//Construct the applet
public Clientreceivecontrol () {
}

//Initialize the applet
public void init() {
  try {
    jbInit();
  }
  catch(Exception e) {
    e.printStackTrace();
  }
  chat.start();
}

//Component initialization
private void jbInit() throws Exception {
  chat=new ClientChat(jTextField1,cam);
  socket=chat.getSocket();
  inMsg=chat.getBufferedReader();
  jLabel1.setText("Massage:");
  jTextField1.setText("                ");
  this.setEnabled(true);
  this.add(jLabel1, null);
  this.add(jTextField1, null);
}

//Get Applet information
public String getAppletInfo() {
  return "Applet Information";
}

//Get parameter info
public String[][] getParameterInfo() {
  return null;
}

public void start(){
}

public void stop(){
}

public void destory(){
  try {
    socket.close();
    inMsg.close();
```

```
    }
    catch (IOException ex) {
      ex.printStackTrace();
    }
  }
}
```

Source code microcontroller สำหรับควบคุมกล้อง

```
;Program Control Camera
;*****************************************
;MAIN PROGRAM
;*****************************************
                ORG         0000H
                LJMP        MAIN

                ORG         0023H
                LJMP        SER_INT

MAIN:           MOV         TMOD,#021H
                MOV         TH1,#0FDH

                MOV         IE,#10010000B
                SETB        TR1

                MOV         SCON,#050H
                CLR         A
                MOV         P0,#00H
                MOV         P1,#00H
                MOV         R0,#00H
                MOV         R1,#00H
                MOV         R2,#00H
                MOV         R3,#00H
                MOV         R4,#00H
                MOV         R5,#00H

MAIN1:          CJNE        A,#41H,CAMERA1      ;A
                JMP         LOOP

LOOP:           MOV         P0,#20H
                CALL        DELAY2
                MOV         P0,#00H
                CJNE        A,#41H,CAMERA1

                MOV         P0,#40H
                CALL        DELAY2
                MOV         P0,#00H
                CJNE        A,#41H,CAMERA1
```

```
                MOV     P0,#80H
                CALL    DELAY2
                MOV     P0,#00H
                CJNE    A,#41H,CAMERA1
                JMP     LOOP

CAMERA1:        CJNE    A,#42H,CAMERA1_L    ;B
                MOV     P0,#20H
CK1_3:          CJNE    R4,#3,CK1_2
TT1:            LCALL   R_1
                DJNZ    R4,TT1
CK1_2:          CJNE    R4,#2,CK1_1
TT2:            LCALL   R_1
                DJNZ    R4,TT2
CK1_1:          CJNE    R4,#1,CK1_0
TT3:            LCALL   R_1
                DJNZ    R4,TT3
CK1_0:          CJNE    R4,#0,CK1_3
                JMP     CK11_3
CK11_3:         CJNE    R5,#3,CK11_2
TT4:            LCALL   L_1
                DJNZ    R5,TT4
CK11_2:         CJNE    R5,#2,CK11_1
TT5:            LCALL   L_1
                DJNZ    R5,TT5
CK11_1:         CJNE    R5,#1,CK11_0
TT6:            LCALL   L_1
                DJNZ    R5,TT6
CK11_0:         CJNE    R5,#0,CK11_3
                JMP     CAMERA1


CAMERA1_L:CJNE  A,#43H,CAMERA1_R    ;C
                MOV     P0,#20H
Y3:             CJNE    R5,#3,Y2
                LCALL   L_1
                DEC     R5
                CLR     A
                JMP     CAMERA1_L
Y2:             CJNE    R5,#2,Y1
                LCALL   L_1
                DEC     R5
                CLR     A
                JMP     CAMERA1_L
Y1:             CJNE    R5,#1,Y0
                LCALL   L_1
```

```
            DEC     R5
            CLR     A
            JMP     CAMERA1_L
Y0:         CJNE    R5,#0,Y3
            CJNE    R4,#3,F34
            JMP     CAMERA1_L
F34:        LCALL   L_1
            CLR     A
            INC     R4
            JMP     CAMERA1_L

CAMERA1_R:CJNE      A,#44H,CAMERA   ;D
            MOV     P0,#20H
D3:         CJNE    R4,#3,D2
            LCALL   R_1
            DEC     R4
            CLR     A
            JMP     CAMERA1_R
D2:         CJNE    R4,#2,D1
            LCALL   R_1
            DEC     R4
            CLR     A
            JMP     CAMERA1_R
D1:         CJNE    R4,#1,D0
            LCALL   R_1
            DEC     R4
            CLR     A
            JMP     CAMERA1_R
D0:         CJNE    R4,#0,D3
            CJNE    R5,#3,F35
            JMP     CAMERA1_R
F35:        LCALL   R_1
            CLR     A
            INC     R5
            JMP     CAMERA1_R

WW0:        LJMP    MAIN1

CAMERA2:    CJNE    A,#45H,CAMERA2_L    ;E
            MOV     P0,#40H
CK2_3:      CJNE    R2,#3,CK2_2
VV1:        LCALL   R_2
            DJNZ    R2,VV1
CK2_2:      CJNE    R2,#2,CK2_1
VV2:        LCALL   R_2
            DJNZ    R2,VV2
CK2_1:      CJNE    R2,#1,CK2_0
```

```
VV3:        LCALL     R_2
            DJNZ      R2,VV3
CK2_0:      CJNE      R2,#0,CK2_3
            JMP       CK22_3
CK22_3:     CJNE      R3,#3,CK22_2
VV4:        LCALL     L_2
            DJNZ      R3,VV4
CK22_2:     CJNE      R3,#2,CK22_1
VV5:        LCALL     L_2
            DJNZ      R3,VV5
CK22_1:     CJNE      R3,#1,CK22_0
VV6:        LCALL     L_2
            DJNZ      R3,VV6
CK22_0:     CJNE      R3,#0,CK22_3
            JMP       CAMERA2

CAMERA2_L:CJNE        A,#46H,CAMERA2_R   ;F
            MOV       P0,#40H
U3:         CJNE      R3,#3,U2
            LCALL     L_2
            DEC       R3
            CLR       A
            JMP       CAMERA2_L
U2:         CJNE      R3,#2,U1
            LCALL     L_2
            DEC       R3
            CLR       A
            JMP       CAMERA2_L
U1:         CJNE      R3,#1,U0
            LCALL     L_2
            DEC       R3
            CLR       A
            JMP       CAMERA2_L
U0:         CJNE      R3,#0,U3
            CJNE      R2,#3,PP
            JMP       CAMERA2_L
PP:         LCALL     L_2
            CLR       A
            INC       R2
            JMP       CAMERA2_L

CAMERA2_R:CJNE        A,#47H,CAMERA3   ;G
            MOV       P0,#40H
L3:         CJNE      R2,#3,L2
            LCALL     R_2
            DEC       R2
            CLR       A
```

```
        JMP     CAMERA2_R
L2:     CJNE    R2,#2,L1
        LCALL   R_2
        DEC     R2
        CLR     A
        JMP     CAMERA2_R
L1:     CJNE    R2,#1,L0
        LCALL   R_2
        DEC     R2
        CLR     A
        JMP     CAMERA2_R
L0:     CJNE    R2,#0,L3
        CJNE    R3,#3,MM
        JMP     CAMERA2_R
MM:     LCALL   R_2
        CLR     A
        INC     R3
        JMP     CAMERA2_R

WW1:    LJMP    WW0

CAMERA3:CJNE    A,#48H,CAMERA3_L   ;H
        MOV     P0,#80H
CK3_3:  CJNE    R0,#3,CK3_2
CC1:    LCALL   R_3
        DJNZ    R0,CC1
CK3_2:  CJNE    R0,#2,CK3_1
CC2:    LCALL   R_3
        DJNZ    R0,CC2
CK3_1:  CJNE    R0,#1,CK3_0
CC3:    LCALL   R_3
        DJNZ    R0,CC3
CK3_0:  CJNE    R0,#0,CK3_3
        JMP     CK33_3
CK33_3: CJNE    R1,#3,CK33_2
CC4:    LCALL   L_3
        DJNZ    R1,CC4
CK33_2: CJNE    R1,#2,CK33_1
CC5:    LCALL   L_3
        DJNZ    R1,CC5
CK33_1: CJNE    R1,#1,CK33_0
CC6:    LCALL   L_3
        DJNZ    R1,CC6
CK33_0: CJNE    R1,#0,CK33_3
        JMP     CAMERA3

CAMERA3_L:CJNE  A,#49H,CAMERA3_R   ;I
```

```
        MOV     P0,#80H
Q3:     CJNE    R1,#3,Q2
        LCALL   L_3
        DEC     R1
        CLR     A
        JMP     CAMERA3_L
Q2:     CJNE    R1,#2,Q1
        LCALL   L_3
        DEC     R1
        CLR     A
        JMP     CAMERA3_L
Q1:     CJNE    R1,#1,Q0
        LCALL   L_3
        DEC     R1
        CLR     A
        JMP     CAMERA3_L
Q0:     CJNE    R1,#0,Q3
        CJNE    R0,#3,AA
        JMP     CAMERA3_L
AA:     LCALL   L_3
        CLR     A
        INC     R0
        JMP     CAMERA3_L

CAMERA3_R:CJNE  A,#50H,WW1  ;P
        MOV     P0,#80H
W3:     CJNE    R0,#3,W2
        LCALL   R_3
        DEC     R0
        CLR     A
        JMP     CAMERA3_R
W2:     CJNE    R0,#2,W1
        LCALL   R_3
        DEC     R0
        CLR     A
        JMP     CAMERA3_R
W1:     CJNE    R0,#1,W0
        LCALL   R_3
        DEC     R0
        CLR     A
        JMP     CAMERA3_R
W0:     CJNE    R0,#0,W3
        CJNE    R1,#3,SS
        JMP     CAMERA3_R
SS:     LCALL   R_3
        CLR     A
        INC     R1
```

```
        JMP     CAMERA3_R

L_1:    MOV     P2,#00H
EE1:    MOV     P2,#10H
        CALL    DELAY1
        MOV     P2,#20H
        CALL    DELAY1
        MOV     P2,#40H
        CALL    DELAY1
        MOV     P2,#80H
        CALL    DELAY1
        RET

R_1:    MOV     P2,#00H
QQ1:    MOV     P2,#80H
        CALL    DELAY1
        MOV     P2,#40H
        CALL    DELAY1
        MOV     P2,#20H
        CALL    DELAY1
        MOV     P2,#10H
        CALL    DELAY1
        RET

L_2:    MOV     P1,#00H
EE2:    MOV     P1,#10H
        CALL    DELAY1
        MOV     P1,#20H
        CALL    DELAY1
        MOV     P1,#40H
        CALL    DELAY1
        MOV     P1,#80H
        CALL    DELAY1
        RET

R_2:    MOV     P1,#00H
QQ2:    MOV     P1,#80H
        CALL    DELAY1
        MOV     P1,#40H
        CALL    DELAY1
        MOV     P1,#20H
        CALL    DELAY1
        MOV     P1,#10H
        CALL    DELAY1
        RET

L_3:    MOV     P0,#00H
```

```
EE3:        MOV         P0,#81H
            CALL        DELAY1
            MOV         P0,#82H
            CALL        DELAY1
            MOV         P0,#84H
            CALL        DELAY1
            MOV         P0,#88H
            CALL        DELAY1
            RET

R_3:        MOV         P0,#00H
QQ3:        MOV         P0,#88H
            CALL        DELAY1
            MOV         P0,#84H
            CALL        DELAY1
            MOV         P0,#82H
            CALL        DELAY1
            MOV         P0,#81H
            CALL        DELAY1
            RET
;*****************************************
;RECEIVE DATA FROM CLIENT
;*****************************************
SER_INT:    JB          RI,REC
TRANS:      CLR         TI
            RETI

REC:        MOV         A,SBUF
            CLR         RI
            RETI

;*****************************************
;DELAY
;*****************************************
DELAY1:     MOV         R7,#10
AGAIN:      MOV         TH0,#4BH
            MOV         TL0,#0FEH
            SETB        TR0
DELAY_50MS:JNB          TF0,DELAY_50MS
            CLR         TR0
            CLR         TF0
            DJNZ        R7,AGAIN
            RET

DELAY2:     MOV         R7,#200
AGAIN2:     MOV         TH0,#4BH
            MOV         TL0,#0FEH
```

```
                SETB        TR0
                CJNE        A,#41H,RET
DELAY_50MS2:JNB             TF0,DELAY_50MS2
                CLR         TR0
                CLR         TF0
                CJNE        A,#41H,RET
                DJNZ        R7,AGAIN2
RET:            RET
                SJMP        $

                END
```

ประวัติผู้เขียน

นายภานุวัฒน์ ดำภูผา เกิดวันที่ 8 กรกฎาคม พ.ศ. 2525 ภูมิลำเนาอยู่บ้านเลขที่ 105 หมู่ที่ 3 ถนนเพชรเกษม ตำบลทรัพย์อนันต์ อำเภอท่าแซะ จังหวัดชุมพร จบการศึกษาระดับมัธยมศึกษาตอนปลายจากโรงเรียนท่าแซะรัชดาภิเษก จังหวัดชุมพร ปีการศึกษา 2543 ปัจจุบันกำลังศึกษาอยู่ชั้นปีที่ 4 สาขาวิชาวิศวกรรมโทรคมนาคม สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี จังหวัดนครราชสีมา

นายวัชรัตน์ เอี่ยมรักษา เกิดวันที่ 5 มีนาคม พ.ศ. 2526 ภูมิลำเนาอยู่บ้านเลขที่ 122 หมู่ที่ 8 ตำบลตาคลี อำเภอตาคลี จังหวัดนครสวรรค์ จบการศึกษาระดับมัธยมศึกษาตอนปลายจากโรงเรียนตาคลีประชาสรรค์ จังหวัดนครสวรรค์ ปีการศึกษา 2543 ปัจจุบันกำลังศึกษาอยู่ชั้นปีที่ 4 สาขาวิชาวิศวกรรมโทรคมนาคม สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี จังหวัดนครราชสีมา

นายวันชัย ก่อการุณย์พงศ์ เกิดวันที่ 12 ธันวาคม พ.ศ. 2525 ภูมิลำเนาอยู่บ้านเลขที่ 243 หมู่ที่ 6 ถนนมิตรภาพ ตำบลหนองกอมเกาะ อำเภอเมือง จังหวัดหนองคาย จบการศึกษาระดับมัธยมศึกษาตอนปลายจากโรงเรียนปทุมเทพวิทยาคาร จังหวัดหนองคาย ปีการศึกษา 2543 ปัจจุบันกำลังศึกษาอยู่ชั้นปีที่ 4 สาขาวิชาวิศวกรรมโทรคมนาคม สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี จังหวัดนครราชสีมา