การพัฒนาแอพพลิเคชันสำหรับเทคโนโลยีจินี่

JINI TECHNOLOGY APPLICATION DEVELOPMENT

นายปานกริช ขันธสะอาด

นายพนิต เวชศิลป์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

การพัฒนาแอพพลิเคชันสำหรับเทคโนโลยีจินี่

JINI APPLICATION TECHNOLOGY DEVELOPMENT

โดย

นายปานกริช ขันธสะอาด

นายพนิต เวชศิลป์

อาจารย์ที่ปรึกษา

อาจารย์ อภิเนตร อูนากูล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

ปริญญานิพนธ์ปีการศึกษา 2543

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การพัฒนาแอพพลิเคชันสำหรับเทคโนโลยีจินี่

JINI TECHNOLOGY APPLICATION DEVELOPMENT

ผู้จัดทำ

1. นาย ปานกริช ขันธสะอาด รหัสประจำตัว 40010457
2. นาย พนิต เวชศิลป์ รหัสประจำตัว 400010486

_____ อาจารย์ที่ปรึกษา

(อาจารย์อภิเนตร อูนากูล)

การพัฒนาแอพพลิเคชันสำหรับเทคโนโลยีจินี่

นายปานกริช ขันธสะอาด 40010457
นายพนิต เวชศิลป์ 40010486
อาจารย์อภิเนตร อูนากูล อาจารย์ที่ปรึกษา
ปีการศึกษา 2543

บทคัดย่อ

ในปัจจุบันระบบเครือข่ายได้ถูกพัฒนาไปอย่างมาก และมีความซับซ้อนมากขึ้น ซึ่งความซับซ้อนนี้เป็นเรื่องที่ยากเกินไปสำหรับผู้ใช้ทั่วไปในการใช้งานและดูแลรักษา ความต้องการของผู้ใช้ทั่วไปนั้นต้องการเครือข่ายที่ใช้งานง่ายและเป็นอัตโนมัติ

บริษัทซันไมโครซิสเต็มได้นำเอาความต้องการของผู้ใช้ที่ต้องการระบบเครือข่ายที่ง่ายและอัตโนมัติ มารวมกับเทคโนโลยีต่างๆที่มีในปัจจุบันและคาดว่าจะมีในอนาคต เพื่อสร้างเป็นฐานของเทคโนโลยีจินี่

จินี่คือชื่อของระบบการประมวลผลแบบกระจายศูนย์แบบหนึ่งซึ่งแตกต่างจากระบบเครือข่ายทั่วไปตรงที่จินี่นั้นมีความสามารถที่เรียกว่า “เน็ตเวิร์กปลั๊กแอนด์เพลย์” ซึ่งความหมายก็คืออุปกรณ์ต่างๆภายในเครือข่ายนั้นมีความสามารถในการถอดหรือเชื่อมต่อเข้ากับระบบแล้วสามารถทำงานได้ทันทีโดยที่ไม่จำเป็นที่จะต้องมีการปรับแต่งใดๆกับระบบ โดยอุปกรณ์ต่างๆไม่ว่าจะเป็นฮาร์ดแวร์หรือซอฟต์แวร์จะนำเสนอตนเองในรูปของบริการบนเครือข่าย

ปริญญานิพนธ์ฉบับนี้ได้ทำการศึกษาถึงเทคโนโลยีของจินี่ และนำสิ่งที่ได้ศึกษามาประยุกต์ใช้โดยการพัฒนาระบบงานขึ้นมาเป็นระบบตัวอย่างเพื่อแสดงให้เห็นถึงความสามารถของเทคโนโลยีจินี่ ระบบที่พัฒนาขึ้นมานั้นเป็นระบบเครือข่ายที่ได้ใช้เทคโนโลยีจินี่มาช่วยในการติดต่อสื่อสารเพื่อควบคุมหุ่นยนต์ขนาดเล็กซึ่งจำลองให้เป็นอุปกรณ์อิเล็กทรอนิกส์ที่มาเชื่อมต่อกับระบบจินี่ หุ่นยนต์เหล่านั้นสามารถเชื่อมต่อกับระบบได้โดยอัตโนมัติเมื่อเริ่มทำงาน และผู้ใช้งานระบบสามารถใช้เครื่องคอมพิวเตอร์มือถือ (ปาล์มไพลอต) ติดต่อเข้ามายังระบบแล้วเรียกใช้บริการในการควบคุมหุ่นยนต์ที่เชื่อมต่ออยู่ในเครือข่ายขณะนั้นได้อย่างง่ายดายโดยผ่านเทคโนโลยีจินี่

Jini Technology Application Development

Mr. Pankrich	Khanthasa-ard	40010457
Mr. Panit	Wechasil	40010486
Mr. Apinetr	Unakul	Advisor

Abstract

Recently, Network Technology was developed rapidly and made the things more complicate. The complexity made normal people feel more difficult to use and maintain the network system, they need the service that more automatics and easy to use.

Sun Micro System brings the need of the people that want network system more easier to use merge with today and tomorrow technologies such as Java RMI, Distributed Object Technology and Wireless Communication for create the new technology called Jini.

Jini is the name of the new Distributed Processing System. The Difference from the old one is Jini has the ability called "Network Plug and Play", this means that the device on the network can be removed or added to the system and can be ready to use immediately, don't need the configuration from user. The devices on the network both hardware device or software device are introduce themselves as the Jini Services.

This Thesis concern about studying Jini technology and apply it to create the demonstrate system that can show the power of Jini. The system that has been developed in this project is the Jini network system of Micro Robots that act as Jini Service. These Robots can easily connect to the network automatically when they're started. User of this system can use the ability of Jini technology to take control of these Robots by use some handheld computer, PalmPilot, connect to the system.

กิตติกรรมประกาศ

ความสำเร็จของวิทยานิพนธ์ฉบับนี้จะไม่อาจเกิดขึ้นได้ หากขาดการเอาใจใส่ ดูแล ให้คำปรึกษา แนะนำทางสว่าง และความช่วยเหลือจากอาจารย์ที่ปรึกษาวิทยานิพนธ์ผู้ซึ่งเล็งเห็นถึงความสำคัญของวิทยานิพนธ์ อาจารย์อภิเนตร อูนากูล ต้องขอขอบพระคุณเป็นอย่างยิ่ง

ขอขอบคุณพระคุณบิดา มารดาที่ให้ข้าพเจ้าได้มีวันแห่งความสำเร็จนี้ ซึ่งท่านได้ทำการทำนุ ดูแลข้าพเจ้ามาด้วยความรัก ให้กำลังใจข้าพเจ้า โดยไม่หวังสิ่งตอบแทน ถึงแม้ว่าคุณบิดาผู้ล่วงลับไปแล้วจะไม่มีโอกาสได้เห็นความสำเร็จในวันนี้ของข้าพเจ้า แต่ข้าพเจ้าก็ภูมิใจมากยิ่งที่เคยได้รับความรัก ความเอาใจใส่จากคุณบิดา อันมีมากมายเหลือคณานับ และขอให้วิญญาณของคุณบิดาซึ่งสถิตย์อยู่ในสากลโลกอันสงบจงเป็นที่มั่นแห่งพลังในข้าพเจ้าและคุณมารดาต่อไป

สุดท้ายนี้ขอบคุณคู่ร่วมทำงานของข้าพเจ้าและบรรดาเพื่อน อันเป็นสหายร่วมอุปสรรค เคยเรียน เล่น ทำงาน ทำกิจกรรมด้วยกันมาตั้งแต่ครั้งเริ่มต้นเข้ามาเรียน ณ สถาบันแห่งนี้ ซึ่งเป็นกำลังใจในการทำวิทยานิพนธ์มาโดยตลอด น้ำใจของเหล่าเพื่อนไม่เคยแห้งแล้งในถิ่นดินแดนนี้

ปานกริช ขันธสะอาด

พนิต เวชศิลป์

สารบัญ

สารบัญตาราง

สารบัญภาพ

บทที่ 1
บทนำ

1.1ความสำคัญและที่มา

ในปัจจุบัน ระบบเครือข่ายกำลังเข้ามามีบทบาทในทุกที่ไม่ว่าจะเป็นที่บ้าน ที่ทำงาน ในสถานศึกษา หรือองค์กรต่างๆ นอกจากนี้ยังมีซอฟต์แวร์ อุปกรณ์ และแอพพลิเคชั่น ในการจัดการกับเครือข่ายในรูปแบบกระจายศูนย์มากขึ้น การลงทุนในเครือข่ายบรอดแบนด์สูงเพื่อการทำงานในระดับโลกก็มีอย่างต่อเนื่อง แต่ระบบเครือข่ายที่เป็นอยู่ในทุกวันนี้มีความซับซ้อนเกินไปสำหรับผู้ใช้ในการที่จะใช้งานและดูแล ผู้ใช้ทั่วไปต้องการเครือข่ายที่ใช้ง่าย และเชื่อถือได้ เพื่อการทำงานร่วมกันของคอมพิวเตอร์ในการแลกเปลี่ยนข้อมูลข่าวสาร บริการ ทางเครือข่าย

การประมวลผลแบบกระจายศูนย์ (Distributed Computing) เป็นเทคโนโลยีที่ช่วยทำให้โปรแกรมประยุกต์ต่างๆที่อยู่บนเครือข่ายสามารถทำงานร่วมกันได้สะดวกมากขึ้น งานบริการและโปรแกรมประยุกต์เหล่านี้นั้นถูกมองเป็นออบเจ็กต์ โดยการที่ออบเจ็กต์ต่างๆในโปรแกรมเดียวกันจะทำงานร่วมกันหรือมีการติดต่อกันระหว่างออบเจ็กต์ด้วยกันนั้นไม่ได้มีความยุ่งยากซับซ้อนมากนักเพราะทำงานอยู่ในสภาพแวดล้อมเดียวกัน แต่ในระบบออบเจ็กต์แบบกระจาย (Distributed Object) นั้นออบเจ็กต์จะกระจัดกระจายอยู่ทั่วไปในระบบเครือข่ายที่หลากหลายรูปแบบ สิ่งที่ยากลำบากคือการที่จะทำให้โปรแกรมประยุกต์ทั้งหลายนั้นสามารถติดต่อสื่อสารกันได้โดยไม่จำเป็นว่าจะต้องทำงานอยู่ในสภาวะแวดล้อมเดียวกัน บนเครื่องเดียวกัน หรือบนระบบปฏิบัติการเดียวกัน

เทคโนโลยีจาวาได้มาทำให้การประมวลผลแบบกระจายศูนย์นั้นเป็นเรื่องที่ง่ายขึ้น เนื่องจากตัวภาษาจาวาเองนั้นก็สามารถทำงานได้โดยไม่ขึ้นอยู่กับแพลตฟอร์มอยู่แล้ว ซึ่งตรงกับความต้องการของการประมวลผลแบบกระจายศูนย์ โดยทางบริษัทซันไมโครซิสเต็มซึ่งเป็นผู้พัฒนาภาษาจาวา ก็ได้ใส่ความสามารถในการทำงานในลักษณะของการประมวลผลแบบกระจายศูนย์ไว้ในเทคโนโลยีจาวาอาร์เอ็มไอ (Java RMI: Remote Method Invocation) ซึ่งติดมากับชุดพัฒนาของภาษาจาวาด้วย ซึ่งจากความสามารถของภาษาจาวาและแนวคิดของการประมวลผลแบบกระจายศูนย์ ก็ทำให้เกิดแนวความคิดของเทคโนโลยีจินี่ขึ้นมา

1.2 แนวความคิดของเทคโนโลยีจินี่

เป็นเวลาหลายสิบปีมาแล้วที่ผู้นิยมในเทคโนโลยีพยายามสร้างแนวความคิด “บ้านอัจฉริยะ” ให้เป็นจริงขึ้นมา เหมือนกับที่เราเห็นในภาพยนตร์แนววิทยาศาสตร์ ที่อุปกรณ์เครื่องใช้ไฟฟ้าต่างๆ ล้วนแต่รู้หน้าที่ของมันเมื่อได้รับคำสั่ง โทรศัพท์จะติดต่อไปยังเลขหมายที่ต้องการได้เอง มีการค้นหาความผิดปกติในรถยนต์โดยอัตโนมัติ สามารถแสดงรายการอาหารที่ถูกเลือกพร้อมรายงานว่ายังขาดส่วนผสมใดบ้าง

และสั่งซื้อไปยังร้านขายของชำในทันที แต่ที่จะเป็นมากกว่านั้นก็คือ เครื่องใช้ไฟฟ้าแต่ละตัวสามารถหยิบยืมประสิทธิภาพของคอมพิวเตอร์มาใช้เมื่อจำเป็น และจะได้ผลตอบรับในทันที เพราะอาศัยการทำงานของสัญญาณไร้สายจึงเป็นเครือข่ายที่กว้างขวางกว่าในปัจจุบันมากนัก และที่สำคัญคือ จะไม่มีความน่ารำคาญเหมือนดังเช่นอุปกรณ์ต่อเชื่อมคอมพิวเตอร์เดสก์ท้อป (desktop) ในปัจจุบัน หรือถ้าจะลองจินตนาการดูว่า คุณได้ไปร่วมประชุมที่ออฟฟิศของลูกค้า ซึ่งคุณมีหน้าที่ในการนำเสนอข้อมูลให้กับลูกค้า เมื่อคุณต่อเครื่องโน้ตบุ๊กไปยังเครือข่ายที่มีอยู่ภายในห้อง เครื่องโน้ตบุ๊กของคุณก็พบกับ พรินเตอร์ (printer), FAX และ เครื่องฉายภาพเคลื่อนไหว (video projector) ที่อยู่ภายในห้องทันที และสามารถนำมาใช้ได้โดยไม่ต้องกำหนดค่าคอนฟิกูเรชั่น (configuration) ของคอมพิวเตอร์แต่อย่างใด หรือถ้าคุณลืมเอกสารอิเล็กทรอนิกส์ที่สำคัญต่อการประชุมไว้ที่ออฟฟิศของคุณ คุณก็จะสามารถเรียกเอกสารนี้จากเครือข่ายที่ออฟฟิศของคุณผ่านทางอินเตอร์เน็ตได้โดยปลอดภัย และสามารถสั่งพิมพ์ออกมาทางพรินเตอร์ที่อยู่ในห้องประชุมได้ทันที ไม่จำเป็นต้องพึ่ง ไดรเวอร์ (driver), ค่าคอนฟิกูเรชั่น, ไม่มีปัญหาเรื่องความเข้ากันได้ (compatibility) ทุกๆ สิ่งจะทำหน้าที่เป็นบริการที่อยู่ในเครือข่าย เป็นเครือข่ายที่เกิดขึ้นได้โดยตัวมันเองจากอุปกรณ์เครือข่ายที่มีอยู่นับล้านล้านชนิดนั่นเอง พลังประมวลผลของคอมพิวเตอร์กลายเป็นบริการหนึ่งที่คอมพิวเตอร์เครื่องอื่นซึ่งอยู่ในเครือข่ายสามารถหยิบยืมสมองหรือโปรเซสเซอร์ (processor) ของเครื่องอื่นมาใช้ได้ตามหลักการของการประมวลผลแบบกระจายศูนย์ นี่คือแนวคิดของเทคโนโลยี จินี่

1.3 จินี่คืออะไร

จินี่ คือชื่อของระบบการประมวลผลแบบกระจายศูนย์ประเภทหนึ่ง ซึ่งพัฒนาโดยบริษัทซันไมโครซิสเต็ม จินี่นั้นแตกต่างจากระบบเครือข่ายทั่วไปตรงที่ จินี่มีความสามารถที่เรียกว่า “เน็ตเวิร์คปลั๊กแอนด์เพลย์” (Network Plug and Play) ซึ่งมีความหมายคืออุปกรณ์ต่างๆที่อยู่ภายในเครือข่ายของจินี่ นั้นมีความสามารถในการถอดออกจากระบบ หรือเชื่อมต่อกับระบบแล้วสามารถทำงานได้ทันทีโดยที่ไม่ต้องมีการปรับแต่งใดๆ กลุ่มของเซอร์วิสที่อยู่ภายในเครือข่ายจินี่เดียวกันนั้นเราสามารถเรียกรวมได้ว่า สังคมของจินี่ (Jini Community)

ในระบบเครือข่ายของจินี่ บริการทุกๆอย่างในระบบถูกมองเป็นสิ่งที่เรียกว่า “เซอร์วิส” (Service) ไม่ว่าสิ่งเหล่านั้นจะอยู่ในรูปของซอฟต์แวร์ หรือฮาร์ดแวร์ก็ตาม โดยจินี่นั้นออกแบบมาให้เซอร์วิสเหล่านี้สามารถ ทำงานร่วมกันได้อย่างไดนามิกและมีเสถียรภาพสูง โดยความสามารถหลักๆของระบบเครือข่ายจินี่ มี 3 ข้อ คือ

- ไม่จำเป็นต้องมีการปรับแต่งของผู้ใช้เมื่อเซอร์วิสมีการนำมาเชื่อมต่อหรือถอดออกจากระบบ
- สังคมของจินี่นั้น สามารถปรับเปลี่ยนตัวเองได้เมื่อมีเซอร์วิสใหม่มาเชื่อมต่อ หรือว่าเซอร์วิสเก่าถูกถอดออกจากระบบ
- ผู้ใช้เซอร์วิสไม่จำเป็นต้องรู้จักกับตัวเซอร์วิสมาก่อน โดยที่เซอร์วิสจะถูกโหลดมายังผู้ใช้โดยอัตโนมัติ และผู้ใช้จะทำการติดต่อกับเซอร์วิสผ่านทางอินเตอร์เฟสที่เป็นที่รู้จัก เช่น เซอร์วิสของพรินเตอร์ ก็จะมีเมธอด print() ไว้สำหรับผู้ใช้

1.4 วัตถุประสงค์ของปริญญานิพนธ์

1. เพื่อศึกษาเทคโนโลยีจินี่ ว่าทำงานอย่างไร และสามารถนำไปประยุกต์กับงานประเภทใดได้บ้าง
2. ศึกษาเทคโนโลยีที่เกี่ยวข้อง เช่น จาวาอาร์เอ็มไอ และการพัฒนาโปรแกรมประยุกต์ด้วยชุดพัฒนาจาวาทูไมโครเอดิชัน (Java 2 Micro Edition) สำหรับพัฒนาโปรแกรมประยุกต์บนคอมพิวเตอร์มือถือ ปาล์มไพลอต
3. นำเทคโนโลยีจินี่ มาประยุกต์ใช้โดยการสร้างระบบเครือข่ายของจินี่จำลองขึ้นมา ทำการสร้างโปรแกรมประยุกต์ และฮาร์ดแวร์ ซึ่งทำหน้าที่เป็นเซอร์วิสของจินี่ โดยการสร้างนี้จะเน้นที่แสดงให้เห็นถึงความง่าย และคุณสมบัติต่างๆของระบบเครือข่ายของจินี่
4. เพื่อเป็นแนวทางในการพัฒนาระบบที่สามารถนำไปประยุกต์ใช้กับงานจริงได้ต่อไป

1.5 ขอบเขตของปริญญานิพนธ์

จากแนวความคิดของเทคโนโลยีจินี่ ดังที่ได้กล่าวมาข้างต้นแล้ว ปริญญานิพนธ์นี้จึงเป็นปริญญานิพนธ์ที่เกี่ยวกับ การนำเทคโนโลยีจินี่ไปประยุกต์ใช้จริง โดยนำเสนอออกมาในรูปของ ระบบตัวอย่าง ซึ่งเป็นระบบเครือข่ายที่ได้ใช้เทคโนโลยีจินี่ มาใช้ในการติดต่อสื่อสารเพื่อควบคุมหุ่นยนต์ขนาดเล็กซึ่งจำลองให้เป็นอุปกรณ์อิเล็กทรอนิกส์ ซึ่งจะมาเชื่อมต่อกับระบบจินี่ โดยมีเครื่องคอมพิวเตอร์ขนาดฝ่ามือ (Palm Computer) มาจำลองเป็นผู้ใช้บริการในระบบเครือข่ายจำลองของจินี่ โดยที่ผู้ใช้บริการจะสามารถควบคุมหุ่นยนต์ได้โดยผ่านทางเครื่องปาล์มนั่นเอง

ลักษณะของระบบนี้ที่แสดงให้เห็นถึงข้อได้เปรียบของเทคโนโลยีจินี่คือ หุ่นยนต์ขนาดเล็กที่มาทำหน้าที่เป็นเซอร์วิสของระบบนั้น สามารถนำมาเชื่อมต่อกับระบบ หรือจะถอดออกจากระบบได้ตลอดเวลา โดยที่เครื่องปาล์ม ที่ทำหน้าที่เป็นผู้ใช้บริการจะสามารถรับทราบได้ถึงการเปลี่ยนแปลงของหุ่นยนต์ และในส่วนของเซอร์วิส ยังได้มีการจำลองเซอร์วิสที่มีลักษณะเป็นหุ่นยนต์ แต่อยู่ในรูปของโปรแกรมจำลอง โดยที่ผู้ใช้ (โปรแกรมจากเครื่องปาล์ม) ยังสามารถเรียกใช้เซอร์วิสได้ด้วยวิธีเดียวกับการเรียกใช้เซอร์วิสที่เป็นฮาร์ดแวร์นั่นเอง จุดนี้แสดงให้เห็นว่า ผู้ใช้ไม่จำเป็นต้องรู้วิธีการสร้างและวิธีการทำงานของเซอร์วิส รู้เพียงแต่อินเตอร์เฟสของเซอร์วิสที่มีให้กับระบบ ก็สามารถเรียกใช้เซอร์วิสได้แล้ว

ตัวอย่างโครงสร้างของระบบ สามารถแสดงเป็นภาพตัวอย่างคร่าวๆได้ดังนี้

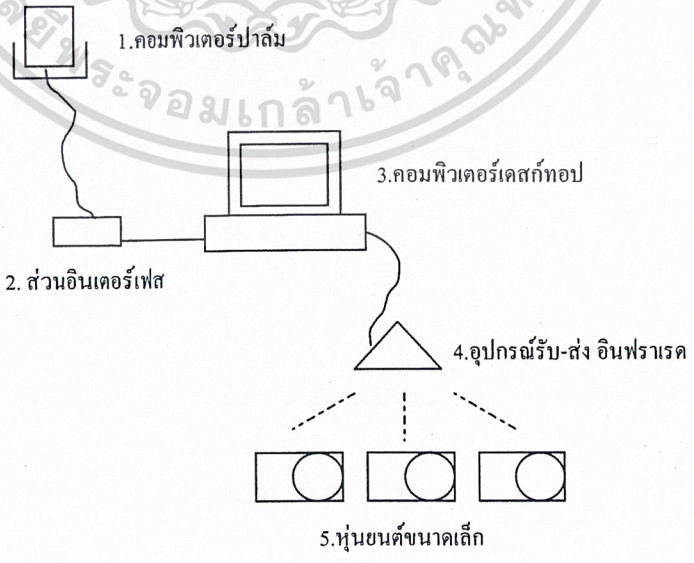
เครื่องปาล์มทำหน้าที่เป็นผู้ใช้ของระบบโดยจะมีโปรแกรมที่มีลักษณะเป็นคันบังคับสามารถใช้บังคับหุ่นได้ เครื่องปาล์มเชื่อมต่อกับคอมพิวเตอร์เดสก์ทอปด้วยอินเตอร์เฟสแบบ RS-232 คอมพิวเตอร์เดสก์ทอปทำหน้าที่เป็นเซิร์ฟเวอร์กลางสำหรับระบบจินี่ และส่วนของหุ่นยนต์ขนาดเล็กนั้นจะติดต่อกับระบบเครือข่ายของจินี่ ผ่านทางอินฟราเรด ซึ่งส่วนนี้อาจจะปรับเปลี่ยนเป็นสื่ออื่นก็สามารถทำได้ หุ่นยนต์ขนาดเล็กนั้นเป็นหุ่นยนต์ที่ควบคุมการทำงานด้วยไมโครคอนโทรลเลอร์ตระกูล 8051 มีล้อซึ่งขับเคลื่อนด้วยสเตปปิ้งมอเตอร์และใช้พลังงานจากแหล่งจ่ายไฟภายนอก

1.6 ขั้นตอนในการดำเนินงาน

การดำเนินงานในปริญญานิพนธ์นี้จะเริ่มจากการศึกษาเทคโนโลยีจินี่และเทคโนโลยีที่เกี่ยวข้อง เช่น จาวาอาร์เอ็มไอ เคเวอร์ชวลแมชีน (KVM: K Virtual Machine) การเขียนโปรแกรมสำหรับปาล์ม เป็นต้น โดยเนื้อหาส่วนใหญ่ที่ต้องศึกษาคือระบบการทำงานและโพรโตคอลของจินี่ เพื่อที่จะใช้ความรู้ที่ได้มาออกแบบโครงสร้างของระบบทั้งหมด

จากนั้นจึงทำการออกแบบระบบตามหลักการและโพรโตคอลของจินี่ ซึ่งการออกแบบและการสร้างระบบนั้น ในชุดพัฒนาจินี่ ได้มี API สำหรับเฟรมเวิร์กของจินี่บางส่วนมาให้แล้วทำให้การออกแบบและพัฒนานั้นง่ายขึ้น โดยส่วนที่ต้องออกแบบคือ

- โครงสร้างของระบบโดยรวม
- ส่วนของการติดต่อกันตามโพรโตคอลของจินี่สำหรับหุ่นยนต์ขนาดเล็กและปาล์ม
- โปรแกรมควบคุมสำหรับปาล์ม โดยพัฒนาด้วยภาษาจาวา ทำงานบน KVM
- โครงสร้างและวงจรต่างๆของหุ่นยนต์ขนาดเล็ก รวมถึงโพรโตคอลในการติดต่อสื่อสาร
- เซอร์วิสจำลองที่สร้างขึ้นเป็นซอฟต์แวร์



รูปที่ 1-1 ภาพรวมของระบบที่จะสร้าง

หลังจากออกแบบเสร็จแล้วก็ทำการพัฒนาส่วนต่างๆของระบบทั้งที่เป็นซอฟต์แวร์และฮาร์ดแวร์ โดยภาษาที่ใช้เป็นหลักในการพัฒนาคือภาษาจาวา ส่วนภาษาที่ใช้ในการโปรแกรมหุ่นยนต์ขนาดเล็ก ใช้ภาษาแอสเซมบลีสำหรับไมโครคอนโทรลเลอร์ตระกูล 8051

บทที่ 2
จินี่

2.1 ภาพรวมของจินี่

จินี่ คือชื่อของระบบออบเจ็กต์แบบกระจายตัวหนึ่งที่มีความสามารถที่เรียกว่า "Network Plug and Play" อุปกรณ์ต่างๆไม่ว่าจะเป็นซอฟต์แวร์หรือฮาร์แวร์สามารถเชื่อมต่อกับระบบแล้วทำการแจ้งให้ระบบทราบถึงการมาของตน ไคลเอ็นต์ที่ต้องการจะเรียกใช้บริการ (Service) ต่างๆบนเครือข่ายสามารถที่จะค้นหาบริการนั้นจากเครือข่ายแล้วทำการเรียกใช้บริการจากผู้ให้บริการนั้นได้ ซึ่งระบบนี้เหมาะกับเครือข่ายที่มีความเปลี่ยนแปลงบ่อย ตัวอย่างสถานการณ์ที่เหมาะสมกับการใช้เครือข่ายจินี่ก็คือ

1) มีเครื่องพรินเตอร์มาเชื่อมต่อกับเครือข่ายและได้แจ้งให้กับระบบรู้ถึงการเชื่อมต่อของตน และประสิทธิภาพของเครื่องที่ทำได้ ซึ่งไคลเอนต์สามารถเรียกใช้เครื่องพรินเตอร์เครื่องนี้ในการพิมพ์งานได้โดยไม่ต้องมีการปรับแต่งอะไรเลย
2) มีกล้องถ่ายรูปแบบดิจิตอลมาเชื่อมต่อกับเครือข่ายแล้วต้องการจะขอใช้บริการการพิมพ์จากเครื่องพรินเตอร์สามารถขอใช้บริการได้เลยโดยที่ไม่ต้องมีการติดตั้ง ไดรเวอร์ของเครื่องพรินเตอร์
3) อุปกรณ์ใหม่ๆสามารถเพิ่มเติมเข้ามาในระบบได้โดยที่ไม่ต้องมีการหยุดการทำงานของระบบและไม่ต้องมีการกลับมาปรับแต่งอะไรกับระบบเครือข่ายอีก
4) ผู้ให้บริการสามารถแจ้งกับระบบได้ถึงการเปลี่ยนแปลงสถานะบางอย่างของระบบ เช่น พรินเตอร์กระดาษหมด โดยที่ไคลเอ็นต์สามารถดักรอการเปลี่ยนแปลงของสถานะต่างๆได้

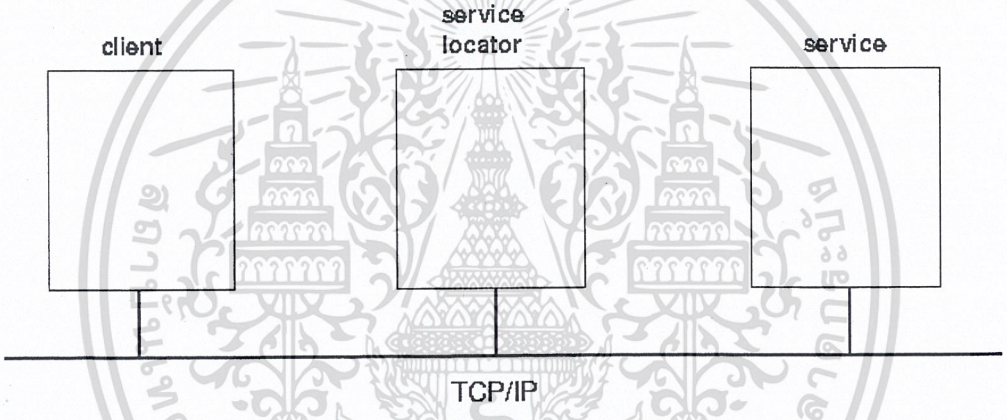
จินี่คือระบบหรือกลุ่มของเซอร์วิสและไคลเอ็นต์ของเซอร์วิสที่มีการติดต่อสื่อสารกันด้วยจินี่โพรโตคอล ซึ่งปรกติแล้วจะประกอบไปด้วย แอพพลิเคชันที่สร้างขึ้นมาด้วยภาษาจาวามีการติดต่อสื่อสารกันด้วยกระบวนการของจาวาอาร์เอ็มไอ แต่ถึงแม้ว่าจินี่จะถูกสร้างขึ้นมาด้วยภาษาจาวาล้วนๆ แต่ว่าไคลเอนต์หรือเซอร์วิสต่างๆที่มาเชื่อมต่อกับเครือข่ายของจินี่ นั้นไม่จำเป็นต้องถูกสร้างขึ้นมาด้วยภาษาจาวาอย่างเดียว อาจจะถูกสร้างขึ้นมาด้วย เนทีฟเมธอด แล้วถูกห่อหุ้มด้วย จาวาออบเจ็กต์หรือแม้กระทั่งเป็นออบเจ็กต์ที่ถูกสร้างขึ้นมาด้วยภาษาอื่นก็สามารถทำได้ ทั้งหมดต้องการเพียงแค่การสื่อสารกันด้วยจินี่โพรโตคอลเท่านั้น

2.2 องค์ประกอบของระบบเครือข่าย จินี่

จินี่เป็นประเภทหนึ่งจากห ลายๆสถาปัตยกรรมของระบบออบเจ็กต์แบบกระจายซึ่งรวมไปถึงระบบที่เป็นมาตรฐานอุตสาหกรรมอย่างเช่น CORBA และ DCOM แต่แตกต่างกันที่จินี่นั้นมีพื้นฐานมาจากภาษาจาวาซึ่งสามารถใช้คุณสมบัติต่างๆของภาษาจาวาได้อย่างครบถ้วน และยังมีเฟรมเวิร์กของจาวาตัวอื่นๆอีกที่ทางซันได้พัฒนาออกมาซึ่งมีส่วนที่คล้ายกับจินี่อยู่บ้าง เช่น เอ็นเตอร์ไพรซ์จาวาบีน

(Enterprise Java Beans) ทำให้จินี่สามารถทำงานร่วมกับเฟรมเวิร์กอื่นๆของซันได้ แต่จะขอไม่กล่าวถึงในที่นี้ว่าทำได้อย่างไรบ้าง

ในระบบของจินี่มีองค์ประกอบอยู่ 3 ส่วนที่เป็นหลักที่สำคัญ ส่วนแรกก็คือเซอร์วิส ตัวอย่างของเซอร์วิสก็เช่น เครื่องพรินเตอร์ เครื่องปิ้งขนมปัง หรือแม้แต่เป็น ซอฟต์แวร์โปรแกรมเช่น โปรแกรมแปลงฟอร์แมตไฟล์ เป็นต้น ส่วนที่สองคือไคลเอนต์ซึ่งมาเรียกใช้บริการของเซอร์วิสต่างๆในระบบ และส่วนสุดท้ายคือ ลุคอัพเซอร์วิส (Lookup Service) ซึ่งทำหน้าที่เป็นตัวกลางระหว่างเซอร์วิสและไคลเอนต์และยังมีส่วนประกอบที่สำคัญอีกตัวหนึ่งคือระบบเครือข่ายที่ทำการเชื่อมองค์ประกอบสำคัญทั้งสามส่วนเข้าด้วยกัน และระบบเครือข่ายที่ใช้สำหรับจินี่นั้นปรกติแล้วจะทำงานอยู่บนเครือข่ายด้วย โพรโตคอล TCP/IP (จินี่นั้นตาม ข้อกำหนดและหลักการนั้นสามารถทำงานได้โดยไม่ขึ้นกับ เครือข่าย โพรโตคอล แต่ในปัจจุบัน เครือข่ายจินี่ที่ถูกสร้างขึ้นนั้น ถูกทำขึ้นสำหรับ TCP/IP เท่านั้น ซึ่งเราสามารถสร้างเครือข่ายจินี่สำหรับ โพรโตคอล อื่นๆก็สามารถทำได้)



รูปที่ 2-1 องค์ประกอบของเครือข่ายจินี่

โค้ดของจาวา สามารถเคลื่อนย้ายระหว่างองค์ประกอบทั้ง 3 ได้ด้วยกระบวนการ ออบเจ็กต์ซีเรียลไลซิง และใช้จาวาซ๊อกเก็ตในการส่งและรับออบเจ็กต์มาจากเครือข่าย ซึ่งออบเจ็กต์ที่อยู่ใน JVM หนึ่งสามารถเรียกใช้ เมธอดของออบเจ็กต์ที่อยู่ในอีก JVM หนึ่งได้โดยกระบวนการของจาวาอาร์เอ็มไอ ถึงแม้ว่าจินี่Specification จะไม่จำเป็นต้องใช้ อาร์เอ็มไอ แต่การเรียกใช้ อาร์เอ็มไอ นั้นทำให้การทำงานของจินี่นั้นง่ายขึ้น

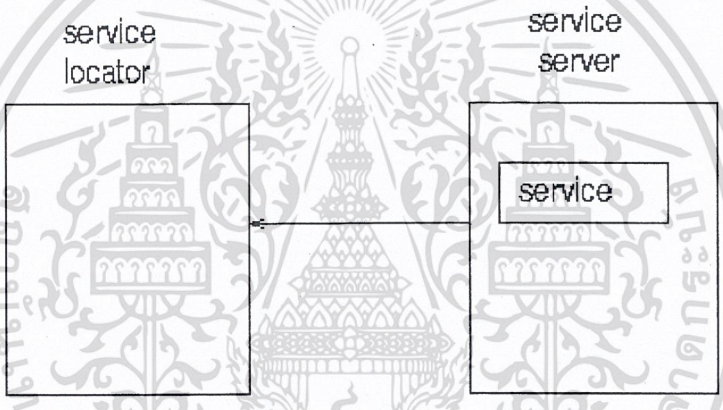
2.3 เซอร์วิสของจินี่

2.3.1 การลงทะเบียน เซอร์วิส

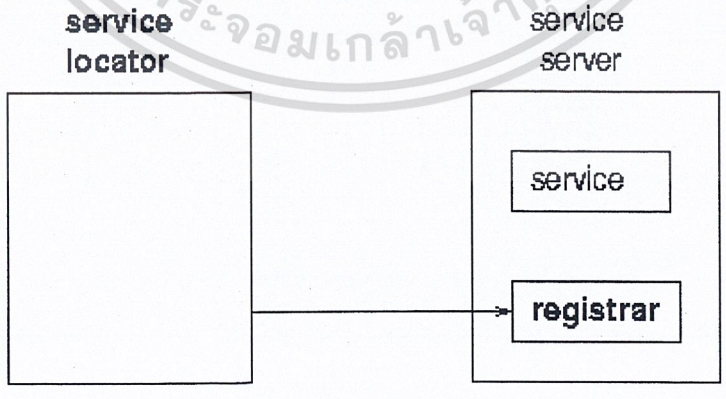
ตัวเซอร์วิสที่แท้จริงของระบบจินี่ก็คือออ บเจ็ กต์หรือกลุ่มของออบเจ็กต์ที่ถูกสร้างขึ้นมาภายในเซิร์ฟเวอร์ นั่นเอง ซึ่งเซิร์ฟเวอร์จะทำหน้าที่ที่สำคัญสำหรับเซอร์วิสคือการลงทะเบียนเซอร์วิสไปยัง ลุคอัพเซอร์วิส (Service Locator) ในกระบวนการการลงทะเบียนนี้ เซิร์ฟเวอร์จะต้องทำการค้นหา ลุคอัพเซอร์วิส เป็นอันดับแรก ซึ่งกระบวนการนี้สามารถทำได้ 2 วิธีคือ ถ้าหากเรารู้ตำแหน่งที่แน่นอน (Address)

ของ ลุคอัพเซอร์วิส แล้วเซิร์ฟเวอร์สามารถใช้การทำ ยูนิคาสต์ (Unicast) ของ TCP ในการเชื่อมต่อโดยตรงไปยัง ลุคอัพเซอร์วิส แต่ถ้าหาก ไม่รู้ว่าลุคอัพเซอร์วิส นั้นทำงานอยู่ที่ใดในเครือข่ายเซิร์ฟเวอร์สามารถทำการส่ง UDP และ ร้องขอแบบมัลติคาสต์ (Multicast Request) ไปยังเครือข่าย เมื่อ ลุคอัพเซอร์วิส ได้รับรีเควสต์ ตัวลุคอัพเซอร์วิสก็จะส่งออบเจ็กต์กลับไปยังเซิร์ฟเวอร์ซึ่งออบเจ็กต์นี้เราเรียกว่า รีจิสทราออบเจ็กต์ (Registrar Object) ที่ทำหน้าที่เป็น พรอกซี่ ไปยังลุคอัพเซอร์วิสการ Request สิ่งต่างๆไปยังลุคอัพเซอร์วิสจะทำผ่านออบเจ็กต์ตัวนี้ ซึ่งกระบวนการที่ รีจิสทราออบเจ็กต์ ทำการติดต่อกลับไปยังลุคอัพเซอร์วิสนั้น สามารถทำผ่าน โพรโตคอล ใดก็ได้ แต่วิธีที่ง่ายที่สุดวิธีหนึ่งก็คือการทำงานผ่านจาวาอาร์เอ็มไอ

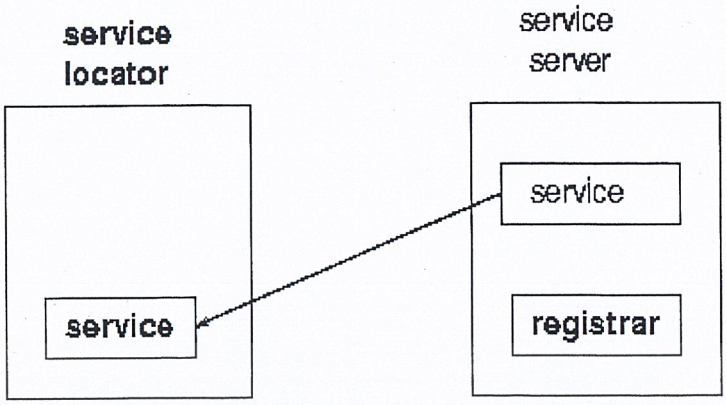
กระบวนการที่เซิร์ฟเวอร์จะทำการลงทะเบียนเซอร์วิสไปยังลุคอัพเซอร์วิสนั้น คือเซิร์ฟเวอร์จะทำการส่งสำเนาของเซอร์วิสObject ไปเก็บไว้ที่ ลุคอัพเซอร์วิส



รูปที่ 2-2 เซอร์วิสทำการค้นหาลุคอัพเซอร์วิส



รูปที่ 2-3 ลุคอัพเซอร์วิสตอบรับด้วย registrar object

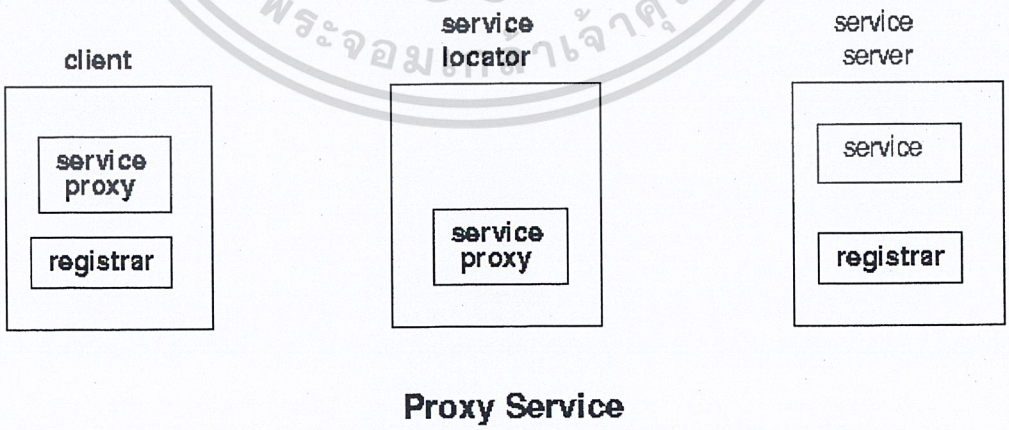


รูปที่ 2-4 เซอร์วิสส่งเซอร์วิสออบเจ็กต์ไปลงทะเบียนที่ลุก อัพเซอร์วิส

จากกระบวนการที่แสดงนี้เซอร์วิสออบเจ็กต์ตัวเดิมก็ยังคงทำงานอยู่ที่เซิร์ฟเวอร์แต่เซอร์วิสออบเจ็กต์ที่เก็บอยู่ที่ลุคอัพเซอร์วิสนั้นเป็นแค่สำเนาของเซอร์วิสออบเจ็กต์ของเซิร์ฟเวอร์เท่านั้น ซึ่งไคลเอนต์ก็จะทำสำเนาของเซอร์วิสออบเจ็กต์นี้ไปจากลุคอัพเซอร์วิสและทำการเรียกใช้บริการของ เซอร์วิสผ่านทางเซอร์วิสออบเจ็กต์ซึ่งอยู่ที่ JVM ของตัวเอง

2.3.2 เซอร์วิสพรอกซี่

ถ้าหากว่าเซอร์วิสนั้นเป็นฮาร์แวร์แทนที่จะเป็น ซอฟต์แวร์ เช่น พรินเตอร์, เครื่องปิ้งขนมปัง ไคลเอนต์ไม่สามารถคุยกับ เซอร์วิสได้โดยตรงเนื่องจากเซอร์วิสนั้นเป็น ฮาร์ดแวร์ ในกรณีนี้ ตัวเซอร์วิสแทนที่จะส่งสำเนาของ ตัวเซอร์วิสเอง(ทำไม่ได้เนื่องจากตัวเซอร์วิสเองเป็น ฮาร์ดแวร์)เซอร์วิสจึงส่งพรอกซี่ออบเจ็กต์ซึ่งทำหน้าที่ติดต่อกลับมายังเซอร์วิสไปแทนโดนอาจจะใช้จาวาอาร์เอ็มไอ ในการส่งการติดต่อ



รูปที่ 2-5 พรอกซี่เซอร์วิส

แต่ก็อาจจะเกิดคำถามมาอีกว่าเนื่องจากพรอกซี่ออบเจ็กต์ไม่ใช่ตัวเซอร์วิสออบเจ็กต์เองพรอกซี่ออบเจ็กต์จะรู้ได้อย่างไรว่าตัวเซอร์วิสนั้นอยู่ที่ไหน ใช้ลุคอัพเซอร์วิสเหมือนกันหรือเปล่า

ไม่ได้มีการกำหนดไว้เป็นวิธีที่ตายตัวว่าจะต้องทำอย่างไร แต่ก็มีหลายวิธีที่จะทำให้พรอกซี่ออบเจ็กต์นั้นสามารถรู้จักกับเซอร์วิสของตัวเอง ตัวอย่างเช่น การใช้กระบวนการของ อาร์เอ็มไอ ในการสร้างพรอกซี่ออบเจ็กต์ก่อนที่จะส่งไป ลงทะเบียน กับลุคอัพเซอร์วิสก็ทำการใส่ Reference ของเซอร์วิสออบเจ็กต์ไปยังพรอกซี่ออบเจ็กต์เพื่อให้พรอกซี่ออบเจ็กต์สามารถมี Reference กลับมายังเซอร์วิสได้ หรือแม้กระทั่งเซอร์วิสทำการใส่ค่าแอดเดรสและพอร์ตของตนเองให้กับพรอกซี่ออบเจ็กต์เพื่อให้พรอกซี่ออบเจ็กต์สามารถสร้าง ซ๊อกเก็ต กลับมายังเซอร์วิสอีกทีก็สามารถทำได้เช่นกัน

2.3.3 โครงสร้างการทำงานของ เซอร์วิส

โครงสร้างของ เซอร์วิสเซิร์ฟเวอร์จะมีกระบวนการภายในซึ่งทำการติดต่อกับลุคอัพเซอร์วิสแล้วทำการลงทะเบียนเซอร์วิสไว้กับลุคอัพเซอร์วิสมีกระบวนการซึ่งแสดงได้ดัง ซูโดโค้ด (Pseudo Code) ต่อไปนี้

- เตรียมตัวในการค้นหา ลุคอัพเซอร์วิส
- ค้นหา ลุคอัพเซอร์วิส
- สร้างข้อมูลเกี่ยวกับเซอร์วิสนั้น (Entry Object)
- ส่งสำเนาของเซอร์วิสออบเจ็กต์ไปยัง ลุคอัพเซอร์วิส
- Renew Lease ตามเวลาที่กำหนด (จะกล่าวถึงในบทที่ 5)

ตัวอย่างโปรแกรมด้านล่างนี้เป็นตัวอย่างของเซอร์วิสที่ชื่อ FileClassifier เซอร์วิสซึ่งทำการตรวจสอบว่าไฟล์ที่ไคลเอนต์ส่งมานั้นมีชนิดของไฟล์คืออะไรแล้วทำการส่งผลลัพท์กลับไปยังไคลเอนต์ซึ่งโปรแกรมนี้ไม่สนใจในเรื่องการตรวจสอบความผิดพลาด (Exception) และจะยังไม่มีการอธิบายการทำงานของโปรแกรมนี้ ในบทนี้ เพียงแต่สาธิตการแปลงซูโดโค้ด ด้านบนให้เป็นโปรแกรมเท่านั้น

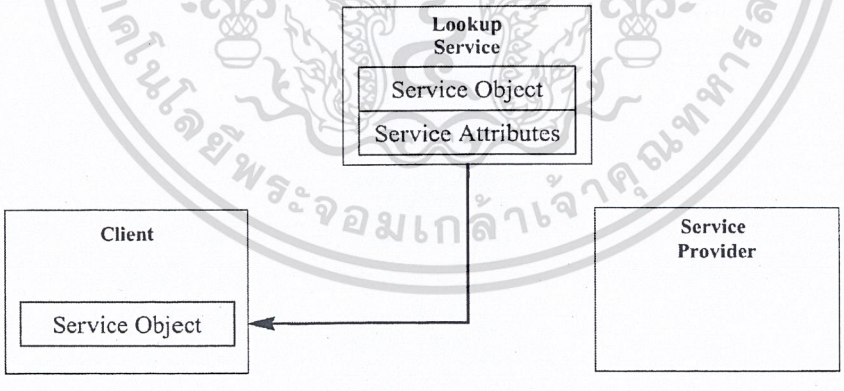
```
public class FileClassifierServer implements DiscoveryListener {
   protected LeaseRenewalManager leaseManager = new LeaseRenewalManager();
   public static void main(String argv[]) {
       new FileClassifierServer();
       // keep server running (almost) forever to
       // - allow time for locator discovery and
       // - keep re-registering the lease
       Thread.currentThread().sleep(Lease.FOREVER);
   }

   public FileClassifierServer() {
       LookupDiscovery discover = null;
       // Prepare for discovery - empty here
       // Discover a lookup service
       // This uses the asynchronous multicast protocol,
       // which calls back into the discovered() method discover = new
       LookupDiscovery(LookupDiscovery.ALL_GROUPS);
       discover.addDiscoveryListener(this);
    }

     public void discovered(DiscoveryEvent evt) {
         ServiceRegistrar registrar = evt.getRegistrars()[0];
```

```
        // At this point we have discovered a lookup service
        // Create information about a service
        ServiceItem item = new ServiceItem(null,
                                new FileClassifierImpl(), null);
        // Export a service
        ServiceRegistration reg = registrar.register(item, Lease.FOREVER);
        // Renew leasing
        leaseManager.renewUntil(reg.getLease(), Lease.FOREVER, this);
    }
} // FileClassifierServer
```

2.4 ไคลเอ็นต์ของจินี่

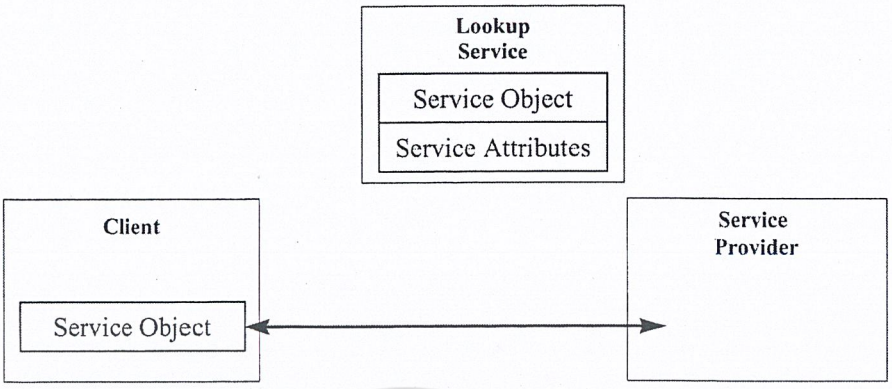
2.4.1 การค้นหาลุคอัพเซอร์วิสของไคลเอ็นต์

กระบวนการที่ไคลเอนต์ทำการค้นหาลุคอัพเซอร์วิสนั้นจะใช้กระบวนการเดียวกับเซิร์ฟเวอร์ในการค้นหาลุคอัพเซอร์วิสโดยที่เมื่อไคลเอนต์สามารถค้นหาลุคอัพเซอร์วิสได้พบแล้ว และต้องการจะเรียกใช้เซอร์วิสจากเครือข่าย จินี่ไคลเอนต์จะต้องทำการค้นหาเซอร์วิสที่ต้องการจากลุคอัพเซอร์วิสโดยไคลเอนต์ต้องทำการสอบถามเข้าไปที่ลุคอัพเซอร์วิสเพื่อค้นหาเซอร์วิสที่ต้องการ การค้นหาเซอร์วิสนั้นไคลเอนต์จะใช้การสร้าง เทมเพลต (Template) ในการค้นหาขึ้นแล้วทำการเรียกใช้เมธอด lookup() ไปยังลุคอัพเซอร์วิสผ่านทางรีจิสทราออบเจ็กต์ที่ได้รับมาโดยส่ง เทมเพลตในการค้นหาไปเป็นพารามิเตอร์และเมื่อลุคอัพเซอร์วิสค้นหาเซอร์วิสที่ไคลเอนต์ต้องการพบก็จะทำการส่งสำเนาของเซอร์วิสออบเจ็กต์ของเซอร์วิสนั้นกลับไปยังไคลเอนต์



รูปที่ 2-6 ไคลเอ็นต์ได้รับเซอร์วิสออบเจ็กต์

เมื่อไคลเอนต์ได้รับเซอร์วิสออบเจ็กต์แล้ว การที่ไคลเอนต์จะเรียกใช้บริการจากเซอร์วิสนั้นไคลเอนต์สามารถทำได้โดยตรงจากการเรียกใช้ไปยังเซอร์วิสออบเจ็กต์โดยที่กระบวนการเรียกใช้เซอร์วิสนี้ไคลเอนต์จะทำงานโดยตรงกับเซอร์วิสโดยที่ไม่จำเป็นต้องเรียกผ่านไปยังลุคอัพเซอร์วิสอีกต่อไป



รูปที่ 2-7 ไคลเอ็นต์ติดต่อกับเซอร์วิสผ่านเซอร์วิสออบเจ็กต์

2.4.2 โครงสร้างการทำงานของไคลเอ็นต์

โครงสร้างของไคลเอนต์ของลุคอัพเซอร์วิสจะมีกระบวนการทำงานภายใน ในการติดต่อกับเครือข่ายของจินี่โดยแสดงให้เป็นเป็น ซูโดโค้ด ดังนี้

- เตรียมตัวในการค้นหา ลุคอัพเซอร์วิส
- ทำการค้นหา ลุคอัพเซอร์วิส
- เตรียมเทมเพลตสำหรับค้นหาเซอร์วิสที่ต้องการใน ลุคอัพเซอร์วิส
- ค้นหาเซอร์วิสใน ลุคอัพเซอร์วิส
- เรียกใช้บริการ เซอร์วิส

ตัวอย่างโปรแกรมด้านล่างนี้เป็น ตัวอย่างของไคลเอนต์ที่ทำการเรียกใช้เซอร์วิสที่ชื่อ FileClassifier โดยที่ไคลเอนต์ทำการค้นหาเซอร์วิสที่ชื่อ FileClassifier แล้วทำการเรียกใช้เมธอดที่ชื่อ getMIMEType() ของเซอร์วิสโดยที่โปรแกรมนี้ไม่สนใจในเรื่องการตรวจสอบความผิดพลาด (Exception) และจะยังไม่มีการอธิบายการทำงานของโปรแกรมนี้ ในบทนี้ โปรแกรมนี้เพียงแต่สาธิตการแปลงซูโดโค้ดด้านบนให้เป็นโปรแกรมเท่านั้น

```
public class TestUnicastFileClassifier {
    public static void main(String argv[]) {
       new TestUnicastFileClassifier();
    }

    public TestUnicastFileClassifier() {
       LookupLocator lookup = null;
       ServiceRegistrar registrar = null;
       FileClassifier classifier = null;

       // Prepare for discovery
       lookup = new LookupLocator("jini://www.all_about_files.com");

       // Discover a lookup service
       // This uses the synchronous unicast protocol
       registrar = lookup.getRegistrar();
```

```
        // Prepare a template for lookup search
        Class[] classes = new Class[] {FileClassifier.class};
        ServiceTemplate template = new ServiceTemplate(null, classes, null);

        // Lookup a service classifier =
                            (FileClassifier)registrar.lookup(template);

        // Call the service
        MIMEType type; type = classifier.getMIMEType("file1.txt");
        System.out.println("Type is " + type.toString());
    }
} // TestUnicastFileClassifier
```

บทที่ 3
จาวาอาร์เอ็มไอ

จาวาอาร์เอ็มไอ (Remote Method Invocation) คือระบบวัตถุแบบกระจาย (Distributed Object System) ซึ่งพัฒนาโดย ซันไมโครซิสเต็ม คำว่า ออบเจ็กต์แบบกระจายนั้น หมายถึงว่า ออบเจ็กต์ที่ทำงานอยู่เป็นระบบหนึ่งนั้น สามารถกระจายอยู่ตามที่ต่างๆบนเครือข่าย (บนจาวาเวอร์ชวลมาชีน ในที่ต่างๆบนระบบเครือข่าย) ซึ่งออบเจ็กต์ที่ถูกสร้างที่อยู่ต่างเวอร์ชวลมาชีนกัน(หรือที่เดียวกันก็ได้) สามารถเรียกใช้เมธอดระหว่างกันได้โดยผ่านทางกระบวนการของ จาวาอาร์เอ็มไอ ซึ่งในฝั่งผู้เรียกใช้ออบเจ็กต์ นั้นจะเรียกใช้ออบเจ็กต์ เหมือนกับว่าออบเจ็กต์ นั้นอยู่ใน แอดเดรสเสปซ เดียวกัน ซึ่งมีลักษณะที่คล้ายกับระบบออบเจ็กต์แบบกระจาย แบบอื่นๆเช่น CORBA หรือ RPCs แต่แตกต่างกันที่อาร์เอ็มไอนั้นถูกพัฒนามาให้ใช้กับภาษาจาวาเท่านั้น แต่ CORBA หรือ RPCs นั้น สามารถใช้กับภาษาใดๆก็ได้

3.1 ข้อแตกต่างระหว่างอาร์เอ็มไอกับ CORBA

1. CORBA เป็นมาตรฐานที่ไม่ขึ้นอยู่กับภาษาใดๆ ทำงานได้กับภาษาเชิงออบเจ็กต์ ได้เกือบจะทุกภาษา
2. CORBA มีกระบวนการต่างๆอีกหลายอย่างที่เป็นมาตรฐาน(เช่น TP Monitor) ซึ่งไม่มีในจาวาอาร์เอ็มไอ
3. ไม่มีการพูดถึงออบเจ็กต์รีเควสต์โบรกเกอร์ (Object Request Broker) ใน อาร์เอ็มไอ

แต่ ในปัจจุบัน อาร์เอ็มไอได้ถูกพัฒนาเพื่อให้เข้ากับมาตรฐานของ CORBA ได้โดยอาร์เอ็มไอจะสามารถทำงานอยู่บนโพรโตคอล IIOP (Internet Inter-ORB) ของ CORBA ได้โดยเรียกชื่อใหม่ว่า RMI/IIOP (RMI Over IIOP)

3.2 สถาปัตยกรรมของอาร์เอ็มไอ

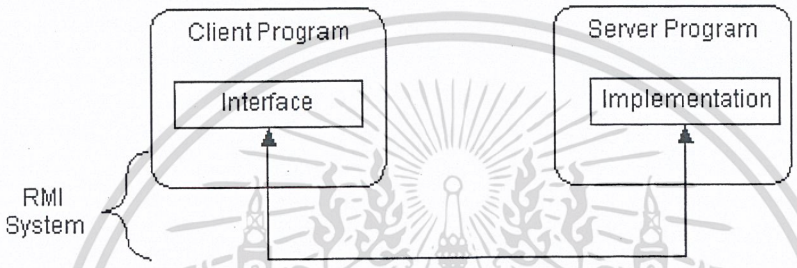
การออกแบบอาร์เอ็มไอนั้นทำขึ้นโดยมีเป้าหมายที่ว่าเพื่อจะสร้างระบบ ออบเจ็กต์แบบกระจายที่สามารถเข้ากันได้กับภาษาจาวา โดยที่ยังคงวิธีการเหมือนกับการเรียกใช้ โลคอลออบเจ็กต์ ซึ่งอาร์เอ็มไอก็สามารถทำได้เป็นอย่างดีโดยส่วนประกอบที่เป็นพื้นฐานของสถาปัตยกรรมของอาร์เอ็มไอมีดังต่อไปนี้

3.2.1 อินเตอร์เฟส : หัวใจหลักของอาร์เอ็มไอ

สถาปัตยกรรมของอาร์เอ็มไอมีหลักการสำคัญอยู่ 1 อย่างก็คือ การสร้างส่วนอธิบายวิธีการปฏิบัติการของออบเจ็กต์ (Object Operation Definition) หรือสามารถเรียกอีกอย่างได้ว่า เมธอดเดฟินิชั่น (Method Definition) ของออบเจ็กต์ และการสร้างวิธีการปฏิบัติการของออบเจ็กต์ (Object Operation Implementation) ซึ่งส่วนที่ใช้อธิบายวิธีการปฏิบัติการของออบเจ็กต์ เราเรียกว่าอินเตอร์เฟสนั่นเอง ซึ่งทั้ง

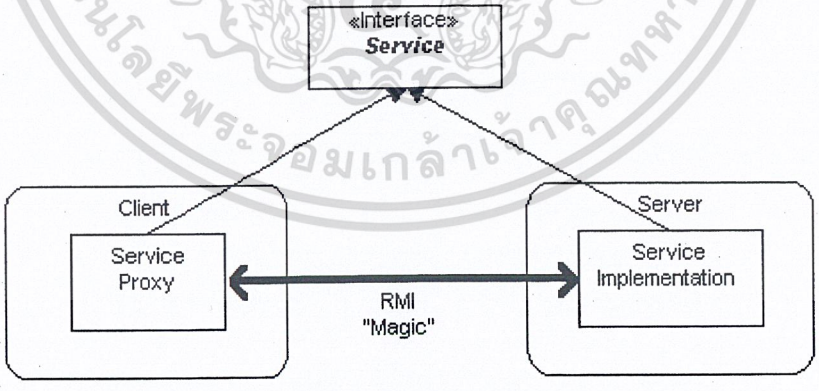
สองส่วนนี้สามารถทำงานแยกกันในต่าง เวอร์ชวลมาชีนได้ ซึ่งหลักการนี้เข้ากันได้ดีกับหลักการของระบบออบเจ็กต์แบบกระจาย คือส่วนของไคลเอ็นต์ ก็เกี่ยวข้องแต่กับส่วนอินเตอร์เฟส ส่วนเซิร์ฟเวอร์ก็มุ่งไปที่การให้บริการออบเจ็กต์

ซึ่งการสร้างส่วนที่อธิบายโอเปอร์เรชันของออบเจ็กต์นั้น ในการเขียนโปรแกรมภาษาจาวาใช้ อินเตอร์เฟสของภาษาจาวาในการสร้าง ส่วนการสร้าง ส่วนปฏิบัติการณ์ของออบเจ็กต์ (Remote Service) นั้นใช้คลาสของจาวาในการสร้าง ดังนั้นหลักการสำคัญของอาร์เอ็มไอที่จะต้องรู้ก็คือ *"อินเตอร์เฟส อธิบายวิธีปฏิบัติการของออบเจ็กต์ ส่วน คลาสอธิบายการสร้างและการทำงานของออบเจ็กต์ โดยไคลเอ็นต์รู้จักเพียงอินเตอร์เฟสส่วนเซิร์ฟเวอร์ทำงานกับ คลาส"*



รูปที่ 3-1 ระบบอินเตอร์เฟสและอิมพลีเมนเทชัน

ส่วนของ โค้ดที่เอ็กซิคิวต์ได้นั้น จะไม่มีอยู่ในส่วนของอินเตอร์เฟส แต่อาร์เอ็มไอจะรองรับ 2 คลาสซึ่ง สร้างจากอินเตอร์เฟสเดียวกันโดย คลาสแรกคลาสที่สร้างจากอินเตอร์เฟสซึ่งทำงานอยู่บนฝั่งเซิร์ฟเวอร์ ส่วนคลาสที่สองทำหน้าที่เป็นพรอกซี่ของรีโมตเซอร์วิส ซึ่งทำงานอยู่ที่ฝั่งไคลเอ็นต์ ดังที่แสดงในรูปต่อไปนี้



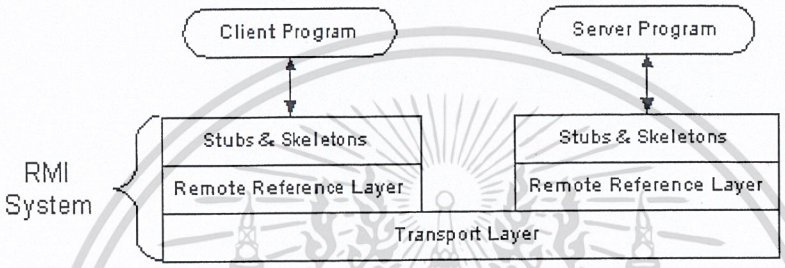
รูปที่ 3-2 การคุยกันระหว่างเซอร์วิสพรอกซี่และตัวเซอร์วิส

โปรแกรมไคลเอ็นต์ จะเรียกใช้ เมธอดไปยัง พรอกซี่ออบเจ็กต์ จากนั้นอาร์เอ็มไอจะทำการส่งคำร้องขอไปยัง ตัวเซอร์วิส ที่ฝั่งเซิร์ฟเวอร์ โดยเซิร์ฟเวอร์จะทำการประมวลผลและถ้าหากมีผลลัพธ์ ผลลัพธ์นั้นก็จะถูกส่งกลับไปให้กับ พรอกซี่ออบเจ็กต์และส่งคืนให้กับโปรแกรมไคลเอ็นต์ ผู้เรียกใช้เมธอดนั้นอีกที

3.2.2 ชั้นของสถาปัตยกรรมของอาร์เอ็มไอ

เมื่อเข้าใจหลักการทำงานของอาร์เอ็มไอในภาพรวมแล้วก็ลองมาดูเบื้องหลังในส่วนที่อธิบายการทำงานของอาร์เอ็มไอว่าทำอย่างไรถึงจะทำเช่นที่กล่าวมาในข้างต้นได้

การทำงานของอาร์เอ็มไอนั้นสร้างขึ้นมาจาก 3 แอ็บสแทร็กต์เลเยอร์ส่วนแรกคือ เลเยอร์ของสตับ (Stub) และสเกลเลตัน (Skeleton) ส่วนที่สองคือ รีโมตรีเฟอร์เรนซ์เลเยอร์ซึ่งอธิบายการเชื่อมต่อของอาร์เอ็มไอและส่วนที่ 3 คือทรานสปอร์ตเลเยอร์ซึ่งสร้างการเชื่อมต่อระหว่าง จาวาเวอร์ชวลมาชีนด้วย TCP/IP ซึ่งเลเยอร์ที่จะพิจารณาเพื่อความเข้าใจ ในโครงการนี้เราจะสนใจแค่เลเยอร์ของสตับและสเกลเลตันเท่านั้น

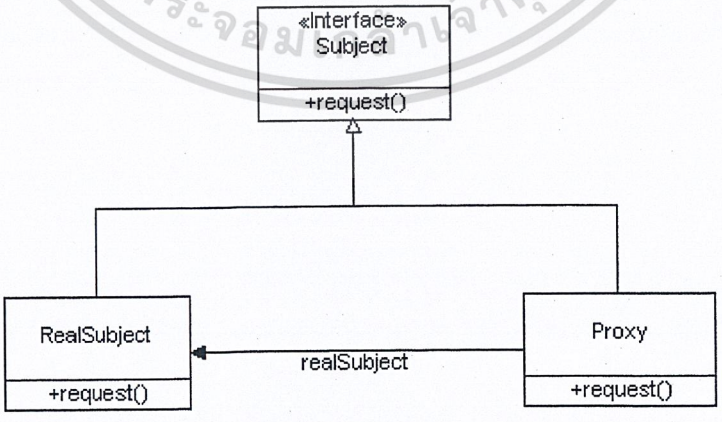


รูปที่ 3-3 เลเยอร์ของสถาปัตยกรรมของอาร์เอ็มไอ

3.2.3 เลเยอร์ของสตับและสเกลเลตัน

การทำงานของเลเยอร์นี้จะเป็นกระบวนการที่ผู้พัฒนาไม่จำเป็นต้องมาเกี่ยวข้อง ซึ่งทำหน้าที่รับการเรียกใช้เมธอดมาจากไคลเอ็นต์และส่งการเรียกนี้ไปยังรีโมตเซอร์วิสที่ฝั่ง เซิร์ฟเวอร์

ในเลเยอร์นี้อาร์เอ็มไอได้ใช้ดีไซน์แพทเทิร์นคือ พรอกซี่แพทเทิร์นซึ่งในพรอกซี่แพทเทิร์นนั้นออบเจ็กต์ที่อยู่ในสิ่งแวดล้อมของระบบหนึ่งสามารถถูกแสดงแทนในอีกระบบหนึ่งด้วยออบเจ็กต์ อีกออบเจ็กต์ หนึ่งที่เรียกว่าพรอกซี่ดังคลาสไดอะแกรมด้านล่างนี้

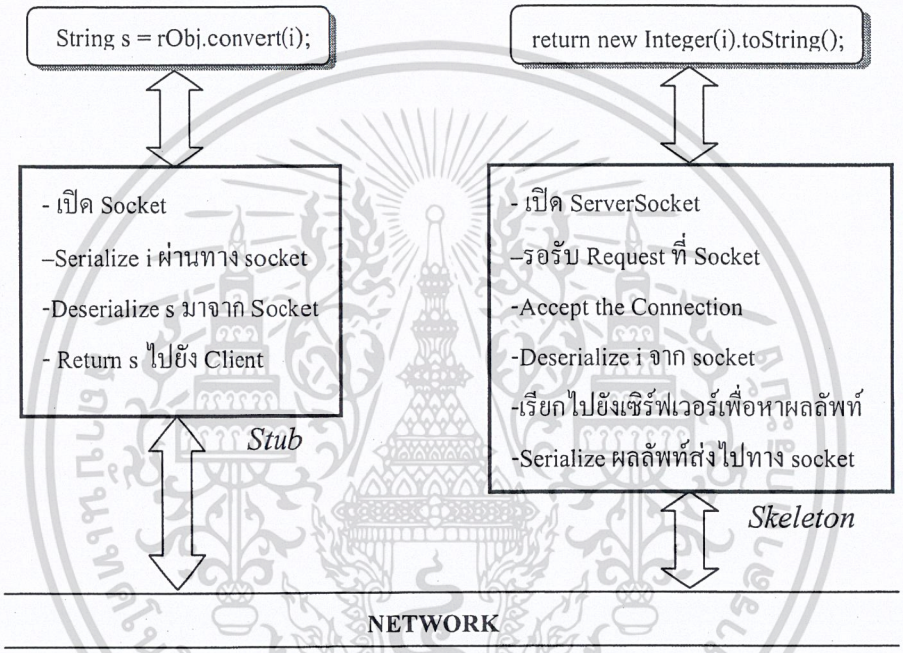


รูปที่ 3-4 ดีไซน์แพทเทิร์นแบบพรอกซี่

สตับและสเกลเลตันเป็นคลาสที่ถูกสร้างขึ้นในอาร์เอ็มไอโดยสร้างจากคลาสที่ทำหน้าที่เป็นรีโมตเซอร์วิสซึ่งวิธีการสร้างนั้นจะกล่าวถึงอีกทีในบทนี้

คลาสของสตับทำหน้าที่เป็น พรอกซี่และ ออบเจ็กต์ที่สร้างเป็นรีโมตเซอร์วิสทำหน้าที่เป็น เรียลซับเจ็กต์ (RealSubject) ส่วนสเกลเลตันนั้นเป็นคลาสช่วยเหลือที่ถูกสร้างขึ้นโดยอาร์เอ็มไอเพื่อที่จะใช้งานในการติดต่อสื่อสารกับ สตับไปบนการเชื่อมต่อของอาร์เอ็มไอโดยสเกลเลตันจะอ่านค่าพารามิเตอร์จากการเรียกใช้เมธอดของสตับแล้วทำการเรียกต่อไปยัง ออบเจ็กต์ที่สร้างเป็นรีโมตเซอร์วิส แล้วรอรับค่าที่ส่งกลับมาจากนั้นจึงส่งค่านั้นกลับไปยังสตับต่อไป

แต่ในชุดพัฒนาของจาวาทูนั้น ทางซันไมโครซิสเต็มได้มีการออกแบบ โพรโตคอลใหม่โดยไม่มีการใช้ สเกลเลตันอีกต่อไป ผู้พัฒนาจึงต้องสนใจสเกลเลตันในกรณีที่พัฒนาแอพพลิเคชันของอาร์เอ็มไอบน JDK 1.0 และ 1.1 เท่านั้น



รูปที่ 3-5 การทำงานของสตับและสเกลเลตัน

3.3 การค้นหารีโมตเซอร์วิส

จากกระบวนการต่างๆที่กล่าวมาในข้างต้น ยังคงมีปัญหาที่ว่า ไคลเอ็นต์นั้น สามารถค้นหา รีโมตเซอร์วิสได้อย่างไร รู้ได้อย่างไรว่า รีโมตเซอร์วิสนั้นอยู่ที่ไหนบนเครือข่าย

ทางแก้ของปัญหานี้ก็คืออาร์เอ็มไอได้มีการใช้ เนมมิ่ง (Naming) และ ไดเร็คทอรี่(Directory) เซอร์วิส (เป็น เซอร์วิส ที่ทำการเก็บรักษาออบเจ็กต์ หรือ ตัวอ้างอิงออบเจ็กต์ไว้โดยการอ้างอิงด้วยชื่อ) ในการเก็บตัวอ้างอิงออบเจ็กต์ของรีโมตเซอร์วิสโดยไคลเอ็นต์นั้นก็จะต้องทำการเรียกใช้ งานเซอร์วิสของเนมมิ่งหรือไดเร็คทอรี่เซอร์วิส นี้ ซึ่งก็ทำให้เป็นปัญหาตามมาอีกว่า ไคลเอ็นต์จะรู้ได้อย่างไรว่า เนมมิ่งหรือไดเร็คทอรี่เซอร์วิสนี้อยู่ที่ใด คำตอบก็คือ เนมมิ่งหรือไดเร็คทอรี่เซอร์วิสนี้จะทำงานอยู่บนโฮสต์และพอร์ตที่เป็นที่รู้จัก หมายความว่า ไคลเอ็นต์จะต้องรู้ว่ามี เนมมิ่งหรือไดเร็คทอรี่เซอร์วิสนั้นทำงานอยู่ที่ไหน และสามารถเรียกใช้บริการของ เนมมิ่งหรือไดเร็คทอรี่เซอร์วิสนั้นได้โดยตรง

การทำงานของ เซิร์ฟเวอร์ผู้ให้บริการรีโมตเซอร์วิสออบเจ็กต์นั้น เซิร์ฟเวอร์จะต้องทำการสร้างโลคอลออบเจ็กต์ จากคลาสซึ่ง สร้างจากรีโมตเซอร์วิสอินเตอร์เฟส (Remote) จากนั้นจึงทำการ เอ็กซ์พอร์ตออบเจ็กต์นั้นออกไปยังอาร์เอ็มไอ(การเอ็กซ์พอร์ตทำได้โดย extends UnicastRemoteObject หรือเรียกใช้เมธอด exportObject()) แล้วจึงทำการลงทะเบียนตัวเองเข้ากับเนมมิ่งเซอร์วิสด้วย ชื่อสาธารณะ (Public Name) (ชื่อที่ไคลเอ็นต์รู้จัก) เพื่อให้ไคลเอ็นต์สามารถค้นหาเซอร์วิสเจอได้ จากนั้นจึงรอให้ ไคลเอ็นต์มาเรียกใช้บริการ

ส่วนทางด้านไคลเอ็นต์นั้น ไคลเอ็นต์สามารถเรียกใช้บริการของ เนมมิ่งเซอร์วิสได้ผ่านทางเมธอด lookup() ซึ่งเป็นสแตติกเมธอดของคลาส Naming ซึ่ง ไคลเอ็นต์จะใช้เมธอดนี้ในการค้นหาไปยัง เนมมิ่งเซอร์วิสว่ามีรีโมตเซอร์วิสที่ต้องการอยู่หรือไม่โดยเมธอดนี้จะอ้างอิง ไปยังเนมมิ่งเซอร์วิสโดยผ่านทางชื่อโฮสต์, พอร์ตและชื่อของ รีโมตเซอร์วิสที่ต้องการค้นหา ถ้าหากค้นหาพบ เมธอดนี้จะคืนค่ารีโมตรีเฟอร์เรนซ์ไปยัง รีโมตเซอร์วิสออบเจ็กต์ซึ่ง ไคลเอ็นต์สามารถ เรียกเมธอดผ่านไปทางรีเฟอร์เรนซ์ นั้น URL ที่ไคลเอ็นต์ใช้ในการติดต่อไปยังอาร์เอ็มไอรีจีสทรี จะอยู่ในรูปของ

rmi://<host_name>[:<name_service_port>]/<service_name>

โดย host_name คือชื่อของเครื่องในระบบ แลนหรือชื่อ DNS ของระบบอินเตอร์เน็ตส่วน name_service_port จำเป็นต้องระบุในกรณีที่ เนมมิ่งเซอร์วิส นั้นทำงานอยู่บนพอร์ตอื่นๆนอกจากพอร์ต 1099 ซึ่งเป็นดีฟอล์ตพอร์ต

3.4 การสร้างระบบของอาร์เอ็มไอ

ในหัวข้อนี้จะแสดงตัวอย่างการสร้างระบบออบเจ็กต์แบบกระจายด้วยอาร์เอ็มไอโดยจะสร้างเป็นระบบตัวอย่างที่ชื่อว่า Remote Converter Service คือเซอร์วิสนี้จะทำการแปลง อินทเจอร์ (Integer) ที่ ไคลเอ็นต์ส่งมาให้ ให้กลายเป็น สตริงก์ (String) แล้วส่งคืนกลับไปยังไคลเอ็นต์และสร้างตัวอย่างของไคลเอ็นต์ในการเรียกใช้ เซอร์วิสดังกล่าว

การสร้างระบบ ออบเจ็กต์แบบกระจายด้วยอาร์เอ็มไอตามระบบตัวอย่างที่กล่าวมา นั้น มีขั้นตอนค่างๆที่ต้องทำดังต่อไปนี้ คือ

1. สร้างอินเตอร์เฟสสำหรับเซอร์วิส
2. สร้าง รีโมตเซอร์วิสจากอินเตอร์เฟสที่สร้างมา
3. สร้าง สตับและสเกลเลตัน
4. สร้าง เซิร์ฟเวอร์สำหรับ รีโมตเซอร์วิส
5. สร้างโปรแกรมสำหรับเป็นไคลเอ็นต์
6. ติดตั้งเซอร์วิสไปยังอาร์เอ็มไอและรันโปรแกรมเพื่อทดสอบ

การเตรียมการสำหรับตัวอย่างนี้คือสร้าง ไดเร็คทอรี่ไว้สอง ไดเร็คทอรี่สำหรับ ซอร์สโค้ดสองส่วน อันแรกสำหรับ เซิร์ฟเวอร์และอันที่สองสำหรับ ไคลเอ็นต์โดยจะมี ไฟล์หนึ่งเป็นอินเตอร์เฟสที่จำเป็นต้องใช้สำหรับทั้งไคลเอ็นต์และเซิร์ฟเวอร์โดยที่เพื่อให้ง่าย ในขั้นตอนนี้เราจะใช้การ ก๊อบปี้ไฟล์ นั้นไปทั้งสองไดเร็คทอรี่

3.5 ตัวอย่าง: การสร้างแอพพลิเคชั่นอาร์เอ็มไอเบื้องต้น

คลาสที่ใช้สำหรับตัวอย่างนี้	
Common Classes	ConvertService
Server Classes	ConvertServiceImpl ConvertServer
Client Classes	ConvertClient

ตารางที่ 3-1 แสดงคลาสที่ใช้สร้างแอพพลิเคชั่นอาร์เอ็มไอเบื้องต้น

3.5.1 รีโมตอินเตอร์เฟส

อินเตอร์เฟสนี้เป็นคอมมอนอินเตอร์เฟสซึ่งทั้ง ไคลเอ็นต์และเซิร์ฟเวอร์จะต้องเรียกใช้ได้ จึงต้องมี คลาสนี้สำหรับ ทั้ง ไคลเอ็นต์และ เซิร์ฟเวอร์โดยที่เซิร์ฟเวอร์ที่ทำการแปลงอินทีเจอร์ เป็น สตริงก์ควรจะมี อินเตอร์เฟสดังนี้

```
import java.rmi.*;

public interface ConvertService extends Remote{
    public String convert(int i) throws RemoteException
}
```

แล้วทำการ คอมไพล์อินเตอร์เฟสนี้ให้เป็น คลาสไฟล์ด้วยคำสั่ง

```
>javac ConvertService.java
```

ใน อินเตอร์เฟสนี้มีสิ่งที่น่าสนใจคือ

- จะต้อง extends java.rmi.Remote interface ซึ่งเป็นอินเตอร์เฟสเปล่าๆ ใช้เพื่อเป็น ตัวบอกชนิดว่าเป็นรีโมทออบเจ็กต์ สำหรับ RMI
- ในทุกๆเมธอดในอินเตอร์เฟสจะต้อง throws java.rmi.RemoteException ด้วย

3.5.2 การสร้างรีโมตเซอร์วิส

รีโมตเซอร์วิสก็คือออบเจ็กต์ ซึ่ง สร้างจากอินเตอร์เฟส Remote ที่สร้างไว้นั่นเอง ซึ่งการแยกส่วนระหว่างอินเตอร์เฟสและส่วนอิมพลีเมนเทชันนั้น สำคัญมากสำหรับทั้งอาร์เอ็มไอและจินี่

มีหลายวิธีที่จะสร้างรีโมตเซอร์วิส แต่วิธีที่ง่ายที่สุดน่าจะเป็นการ extends คลาสนั้นด้วยคลาส java.rmi.server.UnicastRemoteObject ซึ่ง คลาสนี้นั้นจะดูแลเรื่องการส่ง รีเฟอร์เรนซ์ของรีโมตออบเจ็กต์ไปยังเซิร์ฟเวอร์และไคลเอ็นต์ และการเรียกใช้เมธอดระหว่างสองส่วนนั้น หรืออีกวิธีหนึ่งที่สามารถใช้ได้ก็คือการเรียกใช้ สแตติกเมธอดที่ชื่อ exportObject() ของคลาส UnicastRemoteObject ตัวอย่าง การสร้างรีโมตเซอร์วิสคือ

```
import java.rmi.*;
import java.rmi.server.*;

public class ConvertServiceImpl extends UnicastRemoteObject
                                implements ConvertService{

        public ConvertServiceImpl(){
        }

        //Called by the RMI infrastructure when the client
        //invokes the remote method
        public String convert(int i) throws RemoteException{
          return new Integer(i).toString();
        }
}
```

แล้วทำการคอมไพล์คลาสนี้ให้เป็นคลาสไฟล์ด้วยคำสั่ง

```
>javac ConvertServiceImpl.java
```

ในคลาสนี้มีสิ่งที่น่าสนใจคือ

- คอนสตรักเตอร์เมธอดของ UnicastRemoteObject มีการ throws RemoteException ด้วยดังนั้นคอนสตรักเตอร์เมธอดของ ConvertServiceImpl จึงต้อง throw RemoteException ด้วย
- ออบเจ็กต์และเมธอดของคลาสนี้ทำงานอยู่บนเซิร์ฟเวอร์เพราะฉะนั้นรีซอร์สต่างๆที่เซิร์ฟเวอร์ต้องการจะต้องมีให้ด้วย (เช่นคลาสอินเตอร์เฟส)
- หากมีไคลเอ็นต์หลายโปรแกรมที่มาเรียกใช้เมธอด convert() ไคลเอ็นต์แต่ละตัวนั้นจะเชื่อมต่อเข้ามาในเธรดที่แยกจากกันเพราะฉะนั้นเมธอดนี้จะถูกเรียกให้ทำงานพร้อมกันจากหลายไคลเอ็นต์ถ้าหากเซอร์วิสที่ต้องการจะสร้างไม่ต้องการให้มีการทำงานที่ซ้อนกันจะต้องใช้ การทำ ซิงโครไนเซชันเข้ามาช่วย

3.5.3 การสร้างสตับและสเกลเลตัน

การสร้างสตับและสเกลเลตันสำหรับรีโมตเซอร์วิสนี้ทำได้โดยการใช้อาร์เอ็มไอคอมไพเลอร์ ซึ่งมีมาให้กับชุดของ JDK ที่ชื่อ “rmic” เพื่อใช้ในการสร้างสตับและสเกลเลตันไฟล์ซึ่ง คอมไพเลอร์นี้จะใช้ในการ คอมไพล์คลาสไฟล์ของรีโมตเซอร์วิส ด้วยคำสั่ง

```
>rmic ConvertServiceImpl
```

ซึ่งคำสั่งนี้จะทำการสร้างไฟล์ที่ชื่อ ConvertServiceImpl_Stub.class (สตับไฟล์)

และ ConvertServiceImpl_Skel.class (สเกลเลตันไฟล์)

3.5.4 การสร้างเซิร์ฟเวอร์สำหรับรีโมตเซอร์วิส

เซอร์วิสของอาร์เอ็มไอจะต้องถูกสร้างขึ้นมาในโพรเซสของเซิร์ฟเวอร์จากนั้นจึงทำการ ลงทะเบียนเซอร์วิสไปที่เนมมิ่งเซอร์วิสเพื่อให้ไคลเอ็นต์สามารถเรียกใช้ได้ ด้วยสแตติกเมธอด rebind() ของ

คลาส Naming ซึ่งพารามิเตอร์เป็น URL สำหรับเนมมิ่งเซอร์วิสและออบเจ็กต์ที่จะลงทะเบียน ตัวอย่างของเซิร์ฟเวอร์มีดังนี้

```
import java.rmi.*;

public class ConvertServer {
    public ConvertServer(){
      try{
         ConvertService c = new ConvertServiceImpl();
         Naming.rebind("rmi://localhost:1099/ConvertService",c);
      }
      catch(Exception e){
         System.out.println("Troble:"+e);
      }
    }

    public static void main(String[] args){
      new ConvertServer();
    }
}
```

แล้วทำการคอมไพล์คลาสนี้ให้อยู่ในรูปของคลาสไฟล์ด้วยคำสั่ง

```
>javac ConvertService.java
```

3.5.5การสร้างโปรแกรมสำหรับไคลเอ็นต์

สิ่งที่ไคลเอ็นต์โปรแกรมจะต้องทำมีสองอย่างคือ ค้นหาออบเจ็กต์ของรีโมตเซอร์วิสที่ต้องการ แล้วทำการเรียกใช้เมธอดของ รีโมตเซอร์วิสนั้น การค้นหาออบเจ็กต์ของรีโมตเซอร์วิสนั้นสามารถทำได้อย่างง่ายๆโดยเรียกใช้ สแตติกเมธอดที่ชื่อ lookup() ซึ่งเป็นของ คลาส Naming ส่วนการเรียกใช้เมธอดของรีโมตออบเจ็กต์นั้นก็สามารถทำได้เหมือนกับการเรียกใช้เมธอดของโลคอลออบเจ็กต์ธรรมดา ตัวอย่างของไคลเอ็นต์คือ

```
import java.rmi.*;

public class ConvertClient{
    public static void main(String[] args) throws Exception{
    ConvertService cs = (ConvertService)
       Naming.lookup("ConvertService");
    try{
       System.out.println("The result is "+cs.convert(152));
       } catch(RemoteException re){
          System.out.println("Can't convert integer " + re);
       }
    }
}
```

ในการที่ไคลเอ็นต์จะสามารถทำงานกับรีโมตออบเจ็กต์ใดๆไคลเอ็นต์จะเรียกการทำงานผ่านไปทางสตับของ รีโมตออบเจ็กต์นั้น ซึ่งจากตัวอย่าง เราจะต้อง ก๊อบปี้ไฟล์ที่เป็นสตับของ ConvertServiceImpl ที่ชื่อ ConvertServiceImpl_Stub.class มาใส่ไว้ใน classpath ของไคลเอ็นต์ด้วย ไคลเอ็นต์จึงจะทำงานได้ แต่ในส่วนของการคอมไพล์นั้นไคลเอ็นต์ต้องการเพียงอินเตอร์เฟสคลาสในการคอมไพล์เท่านั้น จึงต้องนำคลาสอินเตอร์เฟส มาใส่ไว้ที่ไดเร็คทอรี่ของไคลเอ็นต์ด้วยจากนั้นจึง คอมไพล์ไคลเอ็นต์ด้วยคำสั่ง

>javac ConvertClient.java

3.5.6 การเริ่มการทำงานของระบบ RMI

จากการที่ได้เตรียมการไว้ในตอนแรกว่าให้แยก ไดเรกทอรี่ระหว่างซอร์สโค้ดของไคลเอ็นต์กับซอร์สโค้ดของเซิร์ฟเวอร์ ก็เพื่อจำลองสภาวะแวดล้อมให้เหมือนกับการอยู่ต่างที่กันใน ระบบเครือข่ายโดยที่ในส่วนของเซิร์ฟเวอร์จะต้องมีคลาสดังต่อไปนี้

- ConvertService.class เป็น รีโมตเซอร์วิสอินเตอร์เฟส
- ConvertServiceImpl.class เป็น เป็นตัวรีโมตเซอร์วิส
- ConvertServiceImpl_Stub.class เป็นสตับของรีโมตเซอร์วิส
- ConvertServiceImpl_Skel.class เป็นสเกลเลตันของรีโมตเซอร์วิส
- ConvertServer.class เป็นเซิร์ฟเวอร์ของรีโมตเซอร์วิส

ส่วนในฝั่งของไคลเอ็นต์จะต้องมีคลาสดังต่อไปนี้

- ConvertService.class เป็นรีโมตเซอร์วิสอินเตอร์เฟส
- ConvertServiceImpl_stub.class เป็นสตับของรีโมตเซอร์วิสใช้ในการคุยกับเซอร์วิส
- ConvertClient.class ทำหน้าที่เป็นไคลเอ็นต์ที่เรียกใช้รีโมตเซอร์วิส

วิธีการในการ เริ่มการทำงานของรีโมตเซอร์วิสคือ

1) เริ่มการทำงานของเนมมิ่งเซอร์วิสที่ชื่อ rmiregistry ที่ฝั่งเซิร์ฟเวอร์ที่ดีฟอลต์พอร์ต 1099 ด้วยคำสั่ง

   server> rmiregistry

2) เริ่มการทำงานของเซิร์ฟเวอร์ด้วยคำสั่ง

   server> java ConvertService

3) เริ่มการทำงานของไคลเอ็นต์ด้วยคำสั่ง

   client> java ConvertClient

ถ้าหากการทำงานไม่มีปัญหาอะไรก็จะได้ Output ดังนี้

The result is 152

3.6 การกระจายคลาสไฟล์โดยอัตโนมัติ

จากตัวอย่างที่ได้แสดงไว้ข้างต้นจะเห็นได้ว่า จะต้องมีคลาสที่ใช้สำหรับทั้งไคลเอ็นต์และเซิร์ฟเวอร์และไคลเอ็นต์ก็ยังต้องการ คลาสบาง คลาสที่สร้างโดยเซิร์ฟเวอร์อีกด้วย ซึ่งวิธีการที่ใช้ในตัวอย่างนี้เราใช้การก๊อบปี้คลาสไปยังไดเรกทอรี่ที่ต้องการซึ่งวิธีนี้ก็สามารถทำได้ แต่ว่าในการนำไปใช้จริงคงจะเป็นเรื่องยากที่จะต้องคอย ก๊อบปี้ไฟล์ของเซิร์ฟเวอร์ส่งไปยังไคลเอ็นต์ถ้าหากไคลเอ็นต์กับเซิร์ฟเวอร์

อยู่ห่างไกลกัน ทาง ผู้ออกแบบอาร์เอ็มไอจึงได้ออกแบบส่วนขยายของ การทำ คลาสโหลดดิ้ง โดยเพิ่มส่วนที่ทำให้สามารถ ดาวน์โหลดคลาสผ่านทาง FTP เซิร์ฟเวอร์หรือ HTTP เซิร์ฟเวอร์ได้

RMI สามารถรองรับการทำ การโหลดคลาสระยะไกลโดยผ่านทาง RMIClassLoader ถ้าหากไคลเอ็นต์ทำงานอยู่บนระบบของอาร์เอ็มไอแล้วระบบพบว่าจำเป็นต้องมีการ โหลดคลาสผ่านทาง รีโมตโลเคชัน ระบบก็จะทำการเรียกไปยัง RMIClassLoader เพื่อทำการ โหลดคลาส

การโหลดคลาสของอาร์เอ็มไอจะถูกควบคุมด้วย ค่าพารามิเตอร์จำนวนหนึ่งซึ่ง พารามิเตอร์ต่างๆ เหล่านี้สามารถถูกตั้งค่าได้เมื่อ JVM แต่ละตัวเริ่มทำงานด้วยคำสั่ง

java [-D<PropertyName>=<PropertyValue>] + <ClassFile>

โดยที่ พารามิเตอร์ java.rmi.server.codebase นั้นถูกใช้สำหรับบอก URL ที่ RMIClassLoader ใช้ในการโหลดคลาสซึ่ง URL นี้จะชี้ไปยัง file:, ftp:, http: ที่มีคลาสนั้นให้บริการอยู่ซึ่งออบเจ็กต์ อื่นๆใช้ในการโหลดคลาสสำหรับออบเจ็กต์ที่ได้รับมาจาก JVM นั้น (ใช้กับการทำ คลาสแคสท์ (Class Cast) สำหรับกระบวนการดีซีเรียลไลเซชัน) หรือใช้สำหรับเรียกคลาสที่ต้องการมาใช้ เช่นไคลเอ็นต์จะเรียกใช้สตับคลาสของเซิร์ฟเวอร์ก็สามารถทำโดยผ่าน RMIClassLoader ได้เช่นเดียวกัน

ตัวอย่างวิธีการใช้งาน RMIClassLoader เราสามารถทำได้โดยการ

1) เริ่มการทำงานของ เว็บเซิร์ฟเวอร์ (http server) ที่พอร์ตที่ต้องการ เช่น 8080
2) นำ คลาสที่ต้องการจะทำการ โหลดผ่าน RMIClassLoader ไปใส่ไว้ที่ เว็บเซิร์ฟเวอร์
3) การเริ่มการทำงานของ JVM ที่จะต้องการให้ใช้ RMIClassLoader โดยกำหนดพารามิเตอร์ให้กับ JVM ดังนี้

   >java –Djava.rmi.server.codebase=http://your_host:8080 ConvertService

   หากเรานำ คลาส ConvertServiceImpl_Stub.class ไปใส่ไว้ที่ http://your_host เราก็ไม่จำเป็นจะต้องก๊อบปี้คลาสนั้นไปใส่ไว้ในไดเรกทอรี่เดียวกับ ConvertService โดยที่ RMIClassLoader จะทำการโหลดคลาส ConvertServiceImpl_Stub.class มาให้เองโดยอัตโนมัติ

บทที่ 4
การพัฒนาโปรแกรมพื้นฐานของจินี่

4.1 การค้นหาลุคอัพเซอร์วิส

สำหรับ ไคลเอ็นต์ การที่ ไคลเอ็นต์ จะเรียกใช้เซอร์วิสจะต้องทำการค้นหาเซอร์วิสนั้นผ่านทางลุคอัพเซอร์วิสก่อน กระบวนการแรกที่ต้องทำก่อนจะค้นหาเซอร์วิสก็คือการค้นหาลุคอัพเซอร์วิสและสำหรับเซิร์ฟเวอร์การที่จะลงทะเบียนตัวเองเข้ากับลุคอัพเซอร์วิสสิ่งแรกที่ต้องทำก็คือการค้นหาลุคอัพเซอร์วิสนั่นเอง เพราะฉะนั้นกระบวนการค้นหาลุคอัพเซอร์วิสนี้ จึงเป็นกระบวนการที่ใช้ร่วมกันทั้งไคลเอ็นต์และเซิร์ฟเวอร์

การค้นหาลุคอัพเซอร์วิสนั้นสามารถทำได้ทั้ง วิธียูนิคาสท์ (ทราบอยู่แล้วว่าลุคอัพเซอร์วิสอยู่ที่ใดในเครือข่าย) หรือมัลติคาสท์ (ไม่ทราบว่ามีลุคอัพเซอร์วิสทำงานอยู่ที่ไดบ้าง) ซึ่งแท้จริงแล้วลุคอัพเซอร์วิสก็เป็นเซอร์วิสชนิดหนึ่งของจินี่นั่นเอง แต่ว่าจะมีความพิเศษกว่าเซอร์วิสอื่นๆคือทำหน้าที่เก็บรักษาเซอร์วิสอื่นๆ แล้วส่งต่อเซอร์วิสไปยัง ไคลเอ็นต์ ที่ต้องการ

4.1.1 การค้นหาแบบยูนิคาสท์

การค้นหาเซอร์วิสแบบยูนิคาสต์ (Unicast Discovery) จะใช้เมื่อรู้ที่อยู่ของเครื่องที่มีลุคอัพเซอร์วิสทำงานอยู่ จึงสามารถเรียกใช้ลุคอัพเซอร์วิสได้โดยตรง ซึ่งมักจะใช้กับลุคอัพเซอร์วิสที่ทำงานอยู่นอกเครือข่ายภายใน ซึ่งสามารถระบุได้ว่าอยู่ที่ไหน (อาจจะใช้ ไอพีแอดเดรส ในการระบุ)

LookupLocator

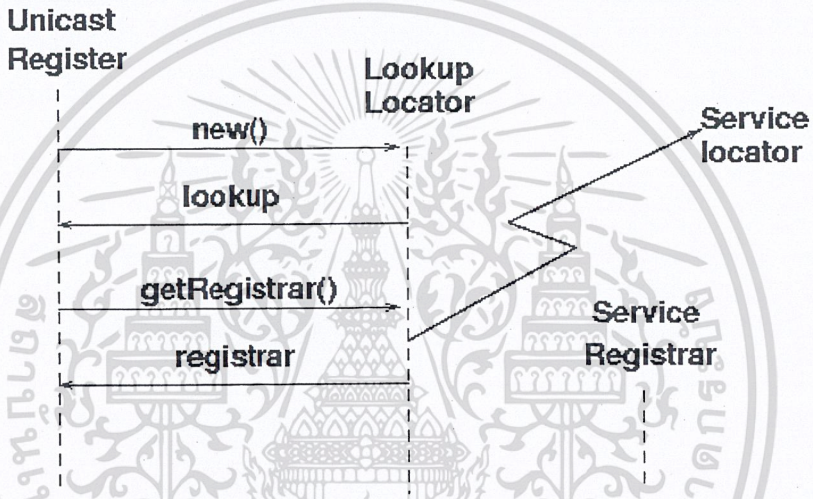
วิธีการที่เราใช้ค้นหาลุคอัพเซอร์วิสโดยการทำ ยูนิคาสท์ ก็คือใช้ออบเจ็กต์ของคลาส LookupLocator ซึ่งเป็นคลาสที่มากับแพคเกจ net.jini.core.discovery ซึ่งคลาส LookupLocator มี คอนสตรัคเตอร์ 2 ตัวคือ

```
package net.jini.core.discovery;
public Class LookupLocator {
    LookupLocator(java.lang.String url) throws
    java.net.MalformedURLException;
    LookupLocator(java.lang.String host,int port);
}
```

สำหรับคอนสตรัคเตอร์ ตัวแรก URL ที่กำหนดจะต้องอยู่ในรูป jini://host หรือ jini://host:port ถ้าไม่มีการกำหนดพอร์ตจะเรียกใช้ พอร์ตที่เป็นค่าดีฟอลต์ คือ 4160 ซึ่งโฮสต์สามารถเป็น localhost หรือเป็น DNS

Name ที่ถูกต้อง ส่วน คอนสตรัคเตอร์ตัวที่สองนั้นจะแยก พอร์ตและ ชื่อโฮสต์ไว้ที่พารามิเตอร์คนละตัวกัน ซึ่งในขั้นตอนนี้จะยังไม่มีการทำยูนิคาสท์

หลังจากที่ทำการสร้าง LookupLocator สำหรับค้นหาลุคอัพเซอร์วิส แล้วการค้นหาลุคอัพเซอร์วิสนั้นจะทำผ่านเมธอด getRegistrar() ซึ่งเป็นของ LookupLocator การเรียกใช้เมธอด getRegistrar() จะทำการค้นหาไปยังเครือข่ายตามแอดเดรสที่ระบุใน คอนสตรัคเตอร์ของ LookupLocator ซึ่ง LookupLocator นั้นจะ คืนค่ามาเป็นออบเจ็กต์ของคลาส ServiceRegistrar ของลุคอัพเซอร์วิสที่ระบุไปมาให้ ออบเจ็กต์นั้นมีชนิดเป็น ServiceRegistrar ซึ่งการนำ ServiceRegistrar ไปใช้อย่างไรนั่นขึ้นอยู่กับว่าเป็นเซิร์ฟเวอร์หรือไคลเอ็นต์ ซึ่งจะกล่าวถึงรายละเอียดของคลาสนี้ในภายหลัง แผนภาพ UML Sequence Diagram ของการค้นหาลุคอัพเซอร์วิสของ LookupLocator คือ



รูปที่ 4-1 การค้นหาเซอร์วิสแบบยูนิคาสท์

ซึ่งตัวอย่างโปรแกรมที่ทำการ ค้นหาลุคอัพเซอร์วิสด้วยวิธียูนิคาสท์แล้วทำการโหลด ServiceRegistrar ออบเจ็กต์มา มีดังต่อไปนี้

```
public class UnicastRegister {
    static public void main(String argv[]) {
       new UnicastRegister();
    }

    public UnicastRegister() {
       LookupLocator lookup = null;
       ServiceRegistrar registrar = null;

       try { lookup = new LookupLocator("jini://localhost"); }
       catch(java.net.MalformedURLException e) {
           System.err.println("Lookup failed: " + e.toString());
           System.exit(1);
       }

       try { registrar = lookup.getRegistrar(); }
       catch (java.io.IOException e) {
           System.err.println("Registrar search failed: " + e.toString());
           System.exit(1);
       } catch (java.lang.ClassNotFoundException e) {
```

```
            System.err.println("Registrar search failed: " + e.toString());
            System.exit(1);
        }

        System.out.println("Registrar found");
        // the code takes separate routes from here for client or service
    }
} // UnicastRegister
```

4.1.2 การค้นหาแบบ Multicast

ถ้าเราไม่สามารถระบุตำแหน่งของลุคอัพเซอร์วิสได้เราจำเป็นที่จะต้องทำการบรอดคาสท์ (Broadcast) เพื่อค้นหาลุคอัพเซอร์วิสการค้นหานั้นอาจจะกระทำผ่านอินเตอร์เน็ต แต่ปรกตินั้นการค้นหาจะถูกจำกัดอยู่ภายใน LAN หรือในเครือข่ายขององค์กร ซึ่งในเครือข่ายขนาดเล็กเช่น เครือข่ายภายในบ้านอาจจะมีลุคอัพเซอร์วิสทำงานอยู่เพียง 1 ตัวเท่านั้น แต่ถ้าเป็นเครือข่ายขนาดใหญ่แล้วมักจะมีมากกว่า 1 ตัว ซึ่งถ้าหากมีลุคอัพเซอร์วิสจำนวนมาก เราสามารถแบ่งลุคอัพเซอร์วิสออกเป็นกลุ่มต่างๆสำหรับประเภทต่างๆของเซอร์วิสที่มาลงทะเบียนกับลุคอัพเซอร์วิสนั้นๆ ตัวอย่างเช่นอาจจะมีลุคอัพเซอร์วิสเฉพาะสำหรับแผนกเอกสารซึ่งทำหน้าที่ดูแล พรินเตอร์เซอร์วิสก็ให้ชื่อกลุ่มว่า “Printer Room” หรือมีลุคอัพเซอร์วิสสำหรับ ห้องประชุมหลายๆห้อง เราอาจะตั้งชื่อกลุ่มของลุคอัพเซอร์วิสสำหรับห้องประชุมว่า “Conference Room” โดยเซอร์วิสจะรู้ว่าตนอยู่ในกลุ่มไหนโดยการกำหนดรายชื่อของกลุ่มให้กับเซอร์วิสโดยกำหนดเป็น อาร์เรย์ของสตริงก์ ดังนี้

```
String[ ] groups = {“Printer Room” , “Conference Room”};
```

LookupDiscovery

การค้นหาลุคอัพเซอร์วิสด้วยวิธีทำมัลติคาสท์ นั้นเราจะใช้คลาส LookupDiscovery มาช่วยในการค้นหาลุคอัพเซอร์วิสซึ่ง LookupDiscovery นั้นเป็น คลาสที่มากับ แพคเกจ net.jini.discovery ซึ่งคลาสนี้มีเพียงคอนสตรัคเตอร์ เดียวคือ

```
LookupDiscovery(java.lang.String[] groups) ;
```

พารามิเตอร์ที่ผ่านให้กับ LookupDiscovery สามารถเป็นได้ 3 กรณีคือ

1) null หรือ LookupDiscovert.ALL_GROUPS หมายความว่าการค้นหาลุคอัพเซอร์วิสนั้นจะค้นหาทุกๆ ลุคอัพเซอร์วิสที่ค้นพบโดยไม่สนใจว่าลุคอัพเซอร์วิสนั้นจะอยู่ในกลุ่มไหน
2) หากเป็น อาร์เรย์เปล่า ของสตริงก์ หรือ LookupDiscovery.NO_GROUPS หมายความว่า ให้สร้างออบเจ็กต์ของ LookupDiscovery ขึ้นมาเฉยๆ แต่ไม่ต้องมีการค้นหาลุคอัพเซอร์วิสซึ่งในกรณีนี้จะใช้เมธอด setGroups() ในการสั่งให้ทำการค้นหา
3) อาร์เรย์ของชื่อกลุ่ม หมายความว่าให้ค้นหาเฉพาะลุคอัพเซอร์วิสอยู่ในกลุ่มที่กำหนดไว้เท่านั้น

Discovery Listener

การทำมัลติคาสท์นั้นเป็นการทำงานบนเครือข่าย ซึ่งคาดว่าจะมีลุคอัพเซอร์วิสตอบกลับมา แต่การที่ ต้องรอให้ LookupDiscovery ค้นหาจนกระทั่งมีการตอบกลับมาจากลุคอัพเซอร์วิสนั้นเป็นการเสียเวลามาก ซึ่งวิธีการจัดการกับความไม่แน่นอนนี้ก็คือ การสร้างออบเจ็กต์ตัวดักฟังมาเพื่อลงทะเบียนกับ LookupDiscovery ซึ่งออบเจ็กต์ตัวดักฟังนั้นจะถูกเรียกให้ทำงานในเวลาที่มีการตอบรับกลับมาจากลุคอัพเซอร์วิสโดยการลงทะเบียนนั้น จะเรียกใช้เมธอด ของ LookupDiscovery คือ

```
public void addDiscoveryListener(DiscoveryListener l);
```

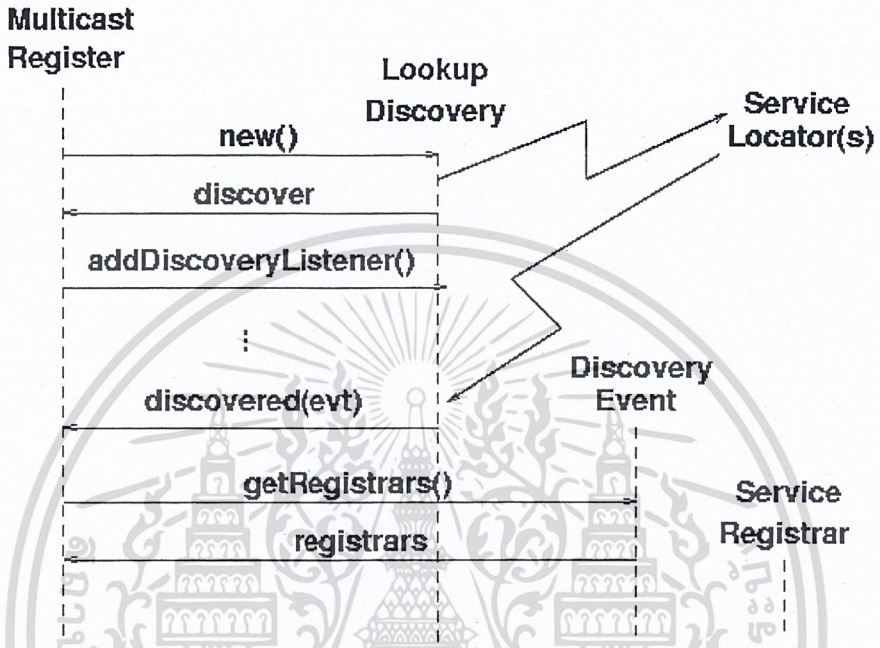
โดยที่ ออบเจ็กต์ตัวดักฟังนั้นจะต้องสร้างตามอินเตอร์เฟส DiscoveryListener ดังนี้

```
package net.jini.discovery;

public abstract interface DiscoveryListener {
    public void discovered(DiscoveryEvent e);
    public void discarded(DiscoveryEvent e);
}
```

ซึ่งเมธอด discovered() จะถูกเรียกให้ทำงานจาก LookupDiscovery เมื่อมีการตอบรับมาจากลุคอัพเซอร์วิสและเมธอด discarded() จะถูกเรียกให้ทำงานเมื่อโปรแกรมต้องการจะเลิกการติดต่อกับลุคอัพเซอร์วิสโดยการเรียกใช้เมธอด discard() ผ่านทาง ออบเจ็กต์ ServiceRegistrar ซึ่งพารามิเตอร์สำหรับเมธอด discovered() คือ DiscoveryEvent ออบเจ็กต์ซึ่งมีอินเตอร์เฟสดังนี้

```
package net.jini.discovery;

public Class DiscoveryEvent {
    public net.jini.core.lookup.ServiceRegistrar[] getRegistrars();
}
```

เมธอด getRegistrars() จะคืนค่ามาเป็น อาร์เรย์ของ ออบเจ็กต์ ServiceRegistrar ซึ่ง ออบเจ็กต์แต่ละตัวนั้นจะเหมือนกัน (มาจากคลาสเดียวกัน) กับออบเจ็กต์ที่คืนมาจากการค้นหาแบบ ยูนิคาสท์ ซึ่งการส่งค่าให้กับ DiscoveryListerner อาจจะมี ServiceRegistrar ส่งมามากกว่า 1 ตัวก็ได้จากการส่งมาครั้งเดียว ซึ่ง UML ของกระบวนการค้นหาลุคอัพเซอร์วิสแบบมัลติคาสท์ คือ



รูปที่ 4-2 การค้นหาเซอร์วิสแบบมัลติคาสท์

ตัวอย่างโปรแกรมที่ทำการค้นหาลุคอัพเซอร์วิสด้วยวิธี มัลติคาสท์ แล้วทำการ โหลดออบเจ็กต์ของ ServiceRegistrar มาจากลุคอัพเซอร์วิสมีดังนี้

```
import net.jini.discovery.LookupDiscovery;
import net.jini.discovery.DiscoveryListener;
import net.jini.discovery.DiscoveryEvent;
import net.jini.core.lookup.ServiceRegistrar;

/* MulticastRegister.java */

public class MulticastRegister implements DiscoveryListener {
    static public void main(String argv[]) {
       new MulticastRegister();
       // stay around long enough to receive replies
       try { Thread.currentThread().sleep(10000L); }
       catch(java.lang.InterruptedException e) { // do nothing }
    }

    public MulticastRegister(){
       System.setSecurityManager(new java.rmi.RMISecurityManager());
       LookupDiscovery discover = null;
       try {
           discover = new LookupDiscovery(LookupDiscovery.ALL_GROUPS);
       } catch(Exception e) {
             System.err.println(e.toString());
             e.printStackTrace(); System.exit(1);
       }
```

```
        discover.addDiscoveryListener(this);
    }

    public void discovered(DiscoveryEvent evt) {
        ServiceRegistrar[] registrars = evt.getRegistrars();
        for (int n = 0; n < registrars.length; n++) {
            ServiceRegistrar registrar = registrars[n];
          // the code takes separate routes from here for client or service
            System.out.println("found a service locator");
        }
    }

    public void discarded(DiscoveryEvent evt) {
    }
} // MulticastRegister
```

4.1.3 ServiceRegistrar

เป็นแอ๊บสแทรกต์คลาส (Abstract Class) ที่ถูกสร้างขึ้นในแต่ละลุคอัพเซอร์วิส หน้าที่ของ ServiceRegistrar คือทำหน้าที่เป็นพรอกซี่ให้กับลุคอัพเซอร์วิสซึ่งตัวพรอกซี่นี้จะทำงานอยู่ที่แอพพลิเคชันทั้งที่ฝั่งไคลเอ็นต์ และฝั่งเซิร์ฟเวอร์

ServiceRegistrar เป็นออบเจ็กต์ตัวแรกที่มีการเคลื่อนย้ายระหว่าง จาวาโพรเซสใน เครือข่ายของจินี่ ServiceRegistrar จะถูกเคลื่อนย้ายจากลุคอัพเซอร์วิสไปยัง แอพพลิเคชันที่ทำการค้นหาลุคอัพเซอร์วิสผ่านทางการเชื่อมต่อแบบ ซ๊อกเก็ต จากนั้น ServiceRegistrar จะทำงานอยู่บนแอพพลิเคชันที่ต้องการติดต่อกับลุคอัพเซอร์วิสโดยที่แอพพลิเคชันนั้นจะเรียกใช้เมธอดของ ServiceRegistrar เพื่อติดต่อไปยังลุคอัพเซอร์วิสซึ่งในตัว ServiceRegistrar จะไม่ได้เก็บข้อมูลใดๆไว้ในฝั่ง แอพพลิเคชันเลยแต่เมื่อ application ต้องการข้อมูลจากลุคอัพเซอร์วิสServiceRegistrar ก็จะไปดึงข้อมูลจากลุคอัพเซอร์วิสมาให้ โดยที่ลุคอัพเซอร์วิสจะมีเมธอด ที่สำคัญอยู่สองส่วนคือ

1) ใช้สำหรับเซอร์วิสในการ ลงทะเบียน ตัวเองเข้ากับลุคอัพเซอร์วิส

```
public ServiceRegistration register(ServiceItem item,
                    long leaseDuration) throws java.rmi.RemoteException
```

2) ใช้สำหรับ ไคลเอ็นต์ ในการค้นหาเซอร์วิสอื่นๆจากลุคอัพเซอร์วิส

```
public java.lang.Object lookup(ServiceTemplate tmpl)
                        throws java.rmi.RemoteException;
public ServiceMatches lookup(ServiceTemplate tmpl, int maxMatches)
                        throws java.rmi.RemoteException;
```

ซึ่งกระบวนการที่เซอร์วิสใช้ ServiceRegistrar ในการลงทะเบียนเซอร์วิสเข้ากับลุคอัพเซอร์วิส และกระบวนการที่ ไคลเอ็นต์ ใช้ในการค้นหาเซอร์วิสอื่นๆในลุคอัพเซอร์วิสจะกล่าวถึงในบทต่อๆไป

4.2 การสร้างเซิร์ฟเวอร์ของเซอร์วิส

4.2.1 ServiceRegistrar

เซิร์ฟเวอร์สำหรับเซอร์วิสสามารถทำการค้นหาลุคอัพเซอร์วิสด้วยวิธี ยูนิคาสท์ ด้วย LookupLocator และวิธีมัลติคาสท์ ด้วย LookupDiscovery ซึ่งผลลัพท์ของทั้งสองวิธีก็คือออบเจ็กต์ของ ServiceRegistrar นั่นเอง ดังที่ได้กล่าวมาแล้ว ServiceRegistrar ทำหน้าที่เป็นพรอกซี่ให้กับ ลุคอัพเซอร์วิส

ซึ่งกระบวนการการ ลงทะเบียนของเซอร์วิสไปยังลุคอัพเซอร์วิส นั้นก็ทำผ่านเมธอด register() ของ ServiceRegistrar นั่นเอง

```
public Class ServiceRegistrar {
    public ServiceRegistration register(ServiceItem item,
            long leaseDuration) throws java.rmi.RemoteException;
}
```

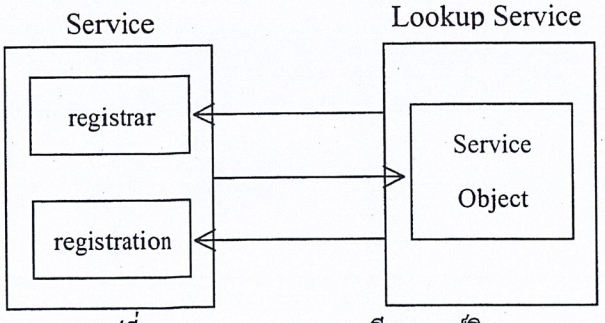
พารามิเตอร์ตัวที่สองคือระยะเวลา (มิลลิวินาที) ที่ ลุคอัพเซอร์วิส จะยังคงให้ เซอร์วิส นั้นลงทะเบียนซึ่ง เซอร์วิส จะต้องแจ้งให้ลุคอัพเซอร์วิสทราบก่อนที่เวลาจะหมดว่าต้องการลงทะเบียนต่อ หากไม่ได้แจ้ง ลุค อัพเซอร์วิส ก็จะทำการยกเลิกการลงทะเบียนของ เซอร์วิส ซึ่งรายละเอียดเกี่ยวกับกระบวนการนี้จะอธิบาย อีกทีในหัวข้อเรื่อง ลีส (Lease) ส่วนพารามิเตอร์ตัวแรกเป็นออบเจ็กต์ซึ่งมีชนิด คือ ServiceItem

```
package net.jini.core.lookup;

public Class ServiceItem {
    public ServiceID serviceID;
    public java.lang.Object service;
    public Entry[] attributeSets;

    public ServiceItem(ServiceID serviceID, java.lang.Object service,
                       Entry[] attrSets);
}
```

4.2.2 ServiceItem

เซิร์ฟเวอร์ จะทำการสร้างออบเจ็กต์ของ ServiceItem และส่งเป็นพารามิเตอร์ให้กับเมธอด register() เพื่อใช้ในการลงทะเบียนเซอร์วิส โดยค่าพารามิเตอร์ที่ส่งผ่านให้กับคอนสตรัคเตอร์ของ ServiceItem ตัวแรกคือ ServiceID ส่งให้เป็น null เมื่อ เซอร์วิส นั้นลงทะเบียนกับ ลุคอัพเซอร์วิส เป็นครั้ง แรก หลังจากลงทะเบียนแล้วลุคอัพเซอร์วิสจะทำการสร้าง ServiceID ให้กับ เซอร์วิส เพื่อใช้เป็น ตัวอ้าง อิงเฉพาะ (Unique Identifier) ให้กับเซอร์วิส เพื่อใช้ต่อไป

พารามิเตอร์ตัวที่สองคือ เซอร์วิสออบเจ็กต์ที่จะใช้ ลงทะเบียน กับ ลุคอัพเซอร์วิส โดยออบเจ็กต์ นี้จะถูกซีเรียลไลซ์ (Serialize) และส่งไปให้กับลุคอัพเซอร์วิส เมื่อมีไคลเอ็นต์ต้องการที่จะเรียกใช้เซอร์วิส นี้ ลุคอัพเซอร์วิสก็จะส่งออบเจ็กต์นี้ให้กับไคลเอ็นต์



รูปที่ 4-3 กระบวนการลงทะเบียนเซอร์วิส

พารามิเตอร์ตัวที่สามคือกลุ่มของข้อมูลเพิ่มเติมเกี่ยวกับเซอร์วิส (Service Attribute) อยู่ในลักษณะของอาร์เรย์ของ Entry ออบเจ็กต์ซึ่งถ้าหากไม่มีข้อมูลเพิ่มเติมก็สามารถส่งค่า null ให้กับ คอนสตรัคเตอร์ได้ วิธีการในการเตรียม เซอร์วิสแอตทริบิวต์ สามารถดูได้ในท้ายบทนี้

4.2.3 ServiceRegistration

หลังจากที่ได้เรียกใช้เมธอดregister() เพื่อลงทะเบียน เซอร์วิส กับ ลุคอัพเซอร์วิส แล้วเมธอด register() จะคืนค่ามาเป็นออบเจ็กต์ที่มี Type คือ ServiceRegistration ซึ่งออบเจ็กต์นี้ทำหน้าที่เป็น Proxy ที่ทำหน้าที่ควบคุมสถานะต่างๆของออบเจ็กต์ที่อยู่ที่ลุคอัพเซอร์วิส

ออบเจ็กต์นี้จะมีข้อมูลต่างๆของ เซอร์วิส ที่ ลุคอัพเซอร์วิส ได้สร้างให้ด้วย ข้อมูลที่สำคัญอันหนึ่งก็คือ ServiceID ซึ่งใช้ในการระบุถึง ออบเจ็กต์ออบเจ็กต์ที่อยู่ที่ ลุคอัพเซอร์วิส ด้วย ServiceID นี้สามารถเรียกดูได้จากเมธอด getServiceID() ของออบเจ็กต์ ServiceRegistration และค่า ServiceID ยังสามารถนำมาใช้ในการลงทะเบียนเซอร์วิสนี้กับลุคอัพเซอร์วิส ตัวอื่นๆได้อีก เพื่อให้ เซอร์วิส นี้มี ID เดียวกันในทุกๆ ลุคอัพเซอร์วิส

ServiceRegistration ยังมีเมธอดอื่นๆอีกที่ใช้ในการปรับเปลี่ยนแอตทริบิวต์ต่าง ของออบเจ็กต์ที่เก็บอยู่ที่ ลุคอัพเซอร์วิส คือ

```
void addAttributes(Entry[] attrSets);
void modifyAttributes(Entry[] attrSetTemplates, Entry[] attrSets);
void setAttributes(Entry[] attrSets);
```

และยังมีเมธอดสุดท้ายที่สำคัญอยู่อีก 1 เมธอดคือ getLease() ซึ่งจะคืนค่ามาเป็น Lease ออบเจ็กต์ของการลงทะเบียน ซึ่งจะกล่าวถึงเรื่อง Lease ออบเจ็กต์โดยรายละเอียดอีกทีหนึ่ง

งานหลักๆของเซิร์ฟเวอร์ก็มีเพียงเท่านี้ หลังจากที่เซิร์ฟเวอร์ได้ส่งเซอร์วิสออบเจ็กต์ไปเก็บไว้ที่ลุคอัพเซอร์วิส แล้วงานที่เหลือของเซิร์ฟเวอร์ก็คือการทำให้เซอร์วิสยังคงลงทะเบียนกับลุคอัพเซอร์วิส อยู่เสมอ เนื่องจากถ้าหากว่าเซอร์วิสออบเจ็กต์ที่ไปฝากไว้ที่ลุคอัพเซอร์วิสนั้นสามารถทำงานทุกอย่างได้เหมือนกับ เซอร์วิสที่เซิร์ฟเวอร์ ตัวเซิร์ฟเวอร์ที่มีเซอร์วิส ทำงานอยู่ก็ไม่จำเป็นต้องคงอยู่อีกต่อไปเนื่องจากการทำงานทั้งหมด เซอร์วิสออบเจ็กต์ที่อยู่ที่ ลุคอัพเซอร์วิส สามารถทำงานแทนได้ แต่ในความจริงไม่ได้เป็นเช่นนั้น เซอร์วิสออบเจ็กต์ที่เซิร์ฟเวอร์ได้ส่งให้กับลุคอัพเซอร์วิส นั้นมักจะทำหน้าที่เป็นแค่พรอกซี่ที่จำเป็นต้องติดต่อกลับมายังเซอร์วิส เพื่อเรียกให้เซอร์วิสทำงาน เพราะฉะนั้นเซอร์วิสจึงต้องยังคงทำงานอยู่เสมอ ซึ่งการลงทะเบียนตัวเองกับลุคอัพเซอร์วิส อยู่เสมอนั้นก็เป็นการยืนยันว่าเซอร์วิสยังคงทำงานอยู่นั่นเอง ซึ่งกระบวนการลงทะเบียนอยู่เสมอนั้นเราจะใช้กระบวนการของลีสซึ่งจะกล่าวถึงรายละเอียดอีกทีในบทที่กล่าวถึงเรื่องลีส

ตัวอย่างโปรแกรมที่ทำหน้าที่เป็นเซิร์ฟเวอร์ของเซอร์วิสแบบง่ายๆ ซึ่งเซอร์วิสออบเจ็กต์ก็คือตัวเซิร์ฟเวอร์เอง และใช้วิธี ยูนิคาสท์ ในการค้นหาลุคอัพเซอร์วิส มีดังนี้

```
import net.jini.core.discovery.LookupLocator;
import net.jini.core.lookup.ServiceRegistrar;
import net.jini.core.lookup.ServiceItem;
import net.jini.core.lookup.ServiceRegistration;
import java.io.Serializable;

/** SimpleService.java */

public class SimpleService implements Serializable {

    static public void main(String argv[]) {
       new SimpleService();
    }

    public SimpleService() {
       LookupLocator lookup = null;
       ServiceRegistrar registrar = null;
       try { lookup = new LookupLocator("jini://localhost"); }
       catch(java.net.MalformedURLException e) {
           System.err.println("Lookup failed: " + e.toString());
           System.exit(1);
       }

       try { registrar = lookup.getRegistrar(); }
       catch (java.io.IOException e) {
           System.err.println("Registrar search failed: " + e.toString());
           System.exit(1);
       }
       catch (java.lang.ClassNotFoundException e) {
           System.err.println("Registrar search failed: " + e.toString());
           System.exit(1);
       }

       // register ourselves as service, with no serviceID
       // or set of attributes
       ServiceItem item = new ServiceItem(null, this, null);
       ServiceRegistration reg = null;

       try { // ask to register for 10,000,000 milliseconds
           reg = registrar.register(item, 10000000L);
       } catch(java.rmi.RemoteException e) {
           System.err.println("Register exception: " + e.toString());
         }

       System.out.println("Service registered");

       // we can exit here if the exported service object can do
       // everything, or we can sleep if it needs to communicate
       // to us or we need to renew a lease later
       //
       // Typically, we will need to renew a lease later
    }
} // SimpleService
```

สิ่งที่น่าสนใจของ โปรแกรมนี้คือ การ implements Serializable ของคลาสนี้ซึ่งการ Implement Serializable หมายความว่าออบเจ็กต์ของคลาสนี้สามารถทำการ ซีเรียลไลซ์ได้ ที่ต้องทำให้ ซีเรียลไลซ์ ได้เพราะว่าคลาสนี้ทำตัวเป็น เซอร์วิสออบเจ็กต์ด้วย และ เซอร์วิสออบเจ็กต์จะต้องถูก ซีเรียลไลซ์ เพื่อส่งตัวออบเจ็กต์ไปยัง ลุคอัพเซอร์วิส เพราะฉะนั้นจึงต้องทำให้คลาสนี้ที่จะเป็น เซอร์วิสออบเจ็กต์นั้น ซีเรียลไลซ์ ได้ด้วยการ Implement Serializable

และคลาสไฟล์ ของ เซอร์วิสออบเจ็กต์จะต้องถูกส่งไปทางเครือข่ายด้วย โดยอาจจะส่งผ่านไปทาง HTTP เซิร์ฟเวอร์ หรือวิธีใดก็ตามไม่จำกัดอยู่เพียงแค่ HTTP

4.2.4 เซอร์วิสแอตทริบิวต์และอินเตอร์เฟส Entry

เซอร์วิสแอตทริบิวต์ใช้ในการระบุถึงรายละเอียดของเซอร์วิส เช่นหากว่าเซอร์วิสนั้นเป็นเซอร์วิสของพรินเตอร์ แอตทริบิวต์ก็มักจะเป็นตำแหน่งที่อยู่ของพรินเตอร์หรือเป็นความสามารถของพรินเตอร์ต่างๆเช่น สามารถพรินต์สีได้หรือไม่ สามารถพิมพ์ที่ความละเอียดสูงสุดเท่าใด หรือแม้แต่เป็นสถานะต่างๆของพรินเตอร์เช่น กระดาษหมด หรือหมึกพิมพ์หมดเป็นต้น แอตทริบิวต์แต่ละตัวที่ใช้ในการระบุถึงรายละเอียดของ จินี่เซอร์วิส นั้นไม่ได้อยู่ในรูปของชื่อ แอตทริบิวต์และค่าพารามิเตอร์โดยตรง แต่ว่ามีลักษณะเป็นออบเจ็กต์ของคลาสที่สร้างตามอินเตอร์เฟส net.jini.core.entry.Entry ตัวอย่างของการสร้าง Entry คลาสมีดังนี้

Entry ออบเจ็กต์นั้นจะต้องเป็นออบเจ็กต์ที่สามารถซีเรียลไลซ์ได้ ซึ่งข้อแตกต่างของแอตทริบิวต์ต่างๆนั้นขึ้นอยู่กับวิธีการ สร้างจาก อินเตอร์เฟส Entry แต่เนื่องจากตัว อินเตอร์เฟส Entry นั้นไม่มีเมธอด มันจึงทำหน้าที่เป็น ตัวแสดงชนิด (Type Identifier) เท่านั้น โดยข้อกำหนดในการสร้างคลาสของ Entry ออบเจ็กต์คือ

- จะต้อง implement interface net.jini.core.entry.Entry และ java.io.Serializable
- ทุกๆ ฟิลด์ ของคลาสจะต้องเป็น Public
- ไม่สามารถมี Instance Variable ของ Primitive Type ได้
- ไม่สามารถอ้างอิงถึงออบเจ็กต์ที่มี ชนิดเป็น static, transient, final หรือตัวที่ไม่ได้เป็น public เนื่องจากว่าออบเจ็กต์ที่มีชนิดเหล่านี้ไม่สามารถทำ ซีเรียลไลซ์ ได้และจะถูก สร้างขึ้นใหม่เมื่อทำ ดีซีเรียลไลซ์
- จะต้องมี ดีฟอลต์คอนสตรัคเตอร์ (ไม่มี อาร์กิวเมนต์)

ตัวอย่างของการสร้างคลาสของออบเจ็กต์ Entry คือ

```
import net.jini.entry.*;
import net.jini.lookup.entry.*;

public class Copyright extends AbstractEntry implements ServiceControlled{

    //The field of Entry -
    //each must be a public, serializable object
    public String owner;
    public String date;
    public String lawfirm;

    public Copyright(){
       this(null,null,null);
    }

    public Copyright(String owner,String date,String lawfirm){
       this.owner = owner;
       this.date = date;
       this.lawfirm = lawfirm
    }
}
```

จากตัวอย่างเป็นการสร้างแอตทริบิวต์ใหม่ชื่อว่า Copyright ซึ่งประกอบด้วย 3 ฟิลด์ซึ่งแสดงชื่อ, วันที่และองค์กรที่ดูแลของ Copyright ซึ่งการสร้างคลาสนี้ทำตามข้อกำหนดด้านบนคือ ทุกฟิลด์เป็น public มี ดีฟอลต์คอนสตรัคเตอร์และ Implement net.jini.core.entry.Entry และ java.io.Serializable ด้วยการอินเฮอริแทนซ์มาจาก net.jini.entry.AbstractEntry ซึ่งได้ implement ทั้งสองอินเตอร์เฟสไว้แล้ว และการ implement net.jini.lookup.entry.ServiceControlled นั้นทำเพื่อเป็นตัวบอกว่าแอตทริบิวต์ของ เซอร์วิสนี้นั้นไม่สามารถแก้ไขได้จากไคลเอ็นต์ ซึ่งไคลเอ็นต์สามารถอ่านได้อย่างเดียวเท่านั้น

และเพื่อความสะดวกในแพคเกจ net.jini.lookup.entry ยังได้มีแอตทริบิวต์คลาสที่สร้างมาพร้อมแล้วโดยสามารถนำมาใช้ได้เลยคือ

- net.jini.lookup.entry.Address : ใช้บอกที่อยู่ของ เซอร์วิส
- net.jini.lookup.entry.Comment : ใช้บอกข้อมูลอื่นๆของ เซอร์วิส
- net.jini.lookup.entry.Location : ใช้บอกตำแหน่งที่เฉพาะของ เซอร์วิส
- net.jini.lookup.entry.Name : ใช้บอกชื่อของ เซอร์วิส
- net.jini.lookup.entry.ServiceInfo : ใช้บอกข้อมูลเบื้องต้นเช่น ผู้ผลิต, เวอร์ชันของ เซอร์วิส
- net.jini.lookup.entry.ServiceType : ใช้บอกประเภทของ เซอร์วิส
- net.jini.lookup.entry.Status : ใช้บอกสถานะปัจจุบันของ เซอร์วิส

ซึ่งรายละเอียดต่างๆและวิธีการใช้คลาสเหล่านี้สามารถดูได้ที่ เอกสารประกอบที่มากับชุดพัฒนา จินี่และตัวอย่างวิธีการสร้าง พารามิเตอร์สำหรับ คอนสตรัคเตอร์ของ ServiceItem คือ

```
Entry[] attributes = new Entry[3];
attributes[0] = new Name("Laser Printer Service");
attributes[1] = new Location("4","4101","Building 4 ");
attributes[2] = new Copyright("AJC Marketing","9/1999","Oaks");
```

4.3 การสร้างไคลเอ็นต์ของเซอร์วิส

4.3.1 ServiceRegistrar

เช่นเดียวกับกระบวนการของเซิร์ฟเวอร์ หลังจากที่ไคลเอ็นต์ค้นหาลุคอัพเซอร์วิส พบแล้วและได้ ServiceRegistrar มาเพื่อใช้เป็นพรอกซี่ไปยังลุคอัพเซอร์วิสแล้วนั้น ไคลเอ็นต์จะเริ่มทำกระบวนการที่แตกต่างจากเซิร์ฟเวอร์ โดยแทนที่จะใช้ ServiceRegistrar ในการลงทะเบียนไปยัง เซิร์ฟเวอร์ กลับใช้ในการค้นหาเซอร์วิสที่ต้องการ (เนื่องจากไคลเอ็นต์ไม่จำเป็นต้องลงทะเบียนกับลุคอัพเซอร์วิส) โดยค้นหาผ่านทางเมธอด lookup() ของ ServiceRegistrar

```
public Class ServiceRegistrar {
    public Object lookup(ServiceTemplate tmpl)
                        throws java.rmi.RemoteException;
    public ServiceMatches lookup(ServiceTemplate tmpl,
                                 int maxMatches)
                        throws java.rmi.RemoteException;
}
```

โดยที่เมธอด lookup() ตัวแรก จะทำเพียงแค่ค้นหาเซอร์วิสที่ต้องการแล้วส่งค่าคืนมาเป็น เซอร์วิสออบเจ็กต์โดย เป็นเซอร์วิสออบเจ็กต์ที่ค้นพบเป็นตัวแรกเท่านั้น ส่วน lookup() ตัวที่สองจะสามารถกำหนดได้ว่าต้องการให้ค้นหาเซอร์วิส มาทั้งหมดไม่เกินกี่ตัวและคืนค่าออกมาเป็นเซอร์วิสออบเจ็กต์ของเซอร์วิสที่ค้นพบทั้งหมดซึ่งอ้างอิงถึงโดยเป็นอาร์เรย์ของ เซอร์วิสออบเจ็กต์อยู่ที่ออบเจ็กต์ServiceMatches.items

วิธีการที่ไคลเอ็นต์ใช้ในการค้นหาเซอร์วิสคือ ทำการสร้างเทมเพลตในการค้นหา โดยใช้วิธีการสร้างออบเจ็กต์ของคลาส net.jini.core.lookup.ServiceTemplate ขึ้นมาเพื่อทำหน้าที่เป็น เทมเพลตออบเจ็กต์

```
public Class ServiceTemplate {
    public ServiceID serviceID;
    public java.lang.Class[] serviceTypes;
    public Entry[] attributeSetTemplates;

    ServiceTemplate(ServiceID serviceID, java.lang.Class[]
                    serviceTypes, Entry[] attrSetTemplates);
}
```

พารามิเตอร์ที่ส่งให้กับ คอนสตรัคเตอร์ของ ServiceTemplate มีดังนี้

1) ServiceID คือออบเจ็กต์ที่เป็น ตัวบ่งชี้เฉพาะของ เซอร์วิส ซึ่งมักจะเป็น null เมื่อมีการค้นหาครั้งแรก และถ้ามีการค้นหาเซอร์วิสเดิมในครั้งต่อไป ก็จะใช้ค่า ID ของเซอร์วิสตัวเดิมจากที่ค้นหาในครั้งแรก
2) Class[ ] ที่จะใช้เป็นต้นแบบของการค้นหา ซึ่งมักจะเป็นออบเจ็กต์ที่มี Type เป็นคลาสซึ่งเป็นอินเตอร์เฟสของ เซอร์วิส ที่ต้องการค้นหา
3) Entry[ ] ซึ่งเป็นแอตทริบิวต์สำหรับเซอร์วิสที่ต้องการ เพื่อใช้ในการค้นหาเซอร์วิสที่ต้องการ ซึ่งวิธีการสร้างเหมือนกับการสร้างแอตทริบิวต์ สำหรับเซอร์วิส

หลังจากกระบวนการค้นหา เซอร์วิส แล้ว หากว่าสามารถค้นหาเซอร์วิสได้พบ ตัว ServiceRegistrar ก็จะส่งเซอร์วิสออบเจ็กต์คืนมาในรูปแบบต่างๆตามรูปแบบของเมธอด lookup() ที่เลือกใช้ หากเลือกใช้ lookup() ตัวแรกก็จะส่งค่าออกมาเป็น เซอร์วิสออบเจ็กต์ตัวแรกที่พบ ซึ่งสามารถนำไปใช้ได้เลย แต่ถ้าหากเลือกใช้ lookup() ตัวที่สองก็จะคืนค่าออกมาเป็นออบเจ็กต์ของคลาส ServiceMatches

```
package net.jini.core.lookup;

public class ServiceMatch{
    public ServiceItem[] items;
    public int totalMatches;
}
```

กระบวนการต่างๆที่กล่าวมาสำหรับทั้ง ไคลเอ็นต์ และ เซอร์วิส เป็นกระบวนการที่มีขั้นตอนเยอะและยุ่งยาก ในชุดพัฒนา จินี่ได้มีคลาสต่างๆที่มาช่วยให้การทำงานตามขั้นตอนด้านบนนั้นง่ายขึ้น เช่น

- net.jini.discovery.LookupLocatorDiscovery
- net.jini.lookup.JoinManager

ซึ่งรายละเอียดเกี่ยวกับวิธีใช้คลาสช่วยเหลือเหล่านี้สามารถดูได้ที่เอกสารประกอบที่ให้มากับชุดพัฒนาจินี่

บทที่ 5
ลีสซิ่งและรีโมตอีเวนต์

5.1 ลีสซิ่ง

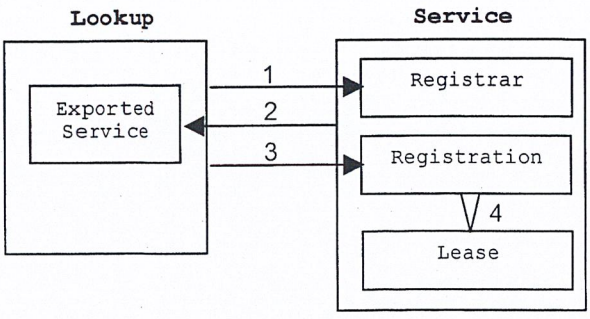
ลีสซิ่ง เป็นกระบวนการที่สำคัญอันหนึ่งของจินี่เฟรมเวิร์ก เพื่อให้เครือข่ายของจินี่ เป็นเครือข่ายที่มีความมั่นคงสูง เนื่องจากในเครือข่ายจินี่ นั้นเซอร์วิสและไคลเอ็นต์มีการเชื่อมต่อและออกจากระบบอยู่ตลอดเวลา ซึ่งจินี่ จะต้องรับรู้ว่าในขณะนั้นมีเซอร์วิสหรือไคลเอ็นต์ใดบ้างที่เชื่อมต่อกับเครือข่ายอยู่และมีใครบ้างที่ออกจากเครือข่ายไปแล้ว ดังนั้นกระบวนการของการทำลีสซิ่ง ได้ถูกสร้างขึ้นเพื่อใช้ในการที่เซอร์วิสของจินี่ จะติดตามว่าไคลเอ็นต์ของเซอร์วิสในขณะนั้นยังคงทำงานอยู่หรือไม่ซึ่งกระบวนการของการทำลีสซิ่ง นั้นสามารถใช้ได้กับทั้งระหว่างลุคอัพเซอร์วิสและ เซอร์วิสของจินี่ (เซอร์วิสของจินี่ ถือเป็นไคลเอ็นต์ของ ลุคอัพเซอร์วิส) หรือระหว่าง เซอร์วิสของจินี่ และไคลเอ็นต์ก็ได้

5.1.1 การใช้ ลีสระหว่างลุคอัพเซอร์วิสและ เซอร์วิสของจินี่

เมื่อเซอร์วิสได้ลงทะเบียนตัวเองกับลุคอัพเซอร์วิสสถานะของการลงทะเบียนนั้นจะคงอยู่ได้ระยะเวลาหนึ่งโดยที่จินี่ ได้ใช้อายุของลีส (ลีสเป็นออบเจ็กต์ตัวหนึ่ง) เพื่อแทนถึงช่วงเวลานั้นเซอร์วิสจะต้องทำการ รีนิว (renew) ลีสภายในช่วงเวลาอายุขัยของลีสนั้น หากอายุขัยของลีสของลีสนั้นหมดลงเซอร์วิสนั้นจะถูก ยกเลิกการลงทะเบียนโดยอัตโนมัติ เพราะฉะนั้นจึงเป็นหน้าที่ของเซอร์วิสที่จะต้องทำการ รีนิว ลีสก่อนที่ลีสจะหมดอายุ ซึ่งจุดประสงค์ของการใช้กระบวนการนี้ก็คือทำให้ระบบเครือข่ายของจินี นั้นมีความมั่นคงมากขึ้น ถ้าหากเซิร์ฟเวอร์สำหรับเซอร์วิสนั้นหยุดทำงาน (ไม่สามารถ รีนิวลีสได้อีกต่อไป) ลุคอัพเซอร์วิสจะสามารถทราบได้ และสามารถนำเซอร์วิสออบเจ็กต์ ที่มีอยู่ของเซอร์วิสนั้นออกไปจากระบบ ทำให้ไคลเอ็นต์ไม่ต้องเสียเวลาในการพยายามที่จะติดต่อกับเซอร์วิสนั้นหากได้เซอร์วิสออบเจ็กต์ไปแล้ว

สำหรับเซอร์วิสของจินี่ ลีสของลุคอัพเซอร์วิสนั้นสามารถได้มาจากกระบวนการลงทะเบียนของเซอร์วิสโดยที่ช่วงอายุของลีสนั้นสามารถขอไปได้โดยผ่านทางเมธอด register() ซึ่งในเมธอด นี้นั้นมีพารามิเตอร์สองตัวที่ส่งไปให้ คือเซอร์วิสออบเจ็กต์ และ ช่วงอายุของลีสซึ่งค่าอายุของลีสที่เป็นไปได้ในการส่งไปมี 3 แบบคือ

1) มีชนิดเป็น long ซึ่งหมายถึงหน่วยเป็นมิลลิวินาที เช่น 1000 หมายถึง 1 วินาที
2) Lease.ANY หมายถึง ให้ลุคอัพเซอร์วิสเป็นผู้ตัดสินใจเองว่าจะให้เวลาเท่าใด
3) Lease.FOREVER หมายถึง ขอลีสที่ไม่มีเวลาหมดอายุ



รูปที่ 5-1 เซอร์วิสได้รับลีสจากลุคอัพเซอร์วิส

ลุคอัพเซอร์วิสทำหน้าที่เป็นผู้กำหนดเวลาให้กับลีสว่าจะให้มีอายุเท่าใดซึ่งไม่จำเป็นว่าจะต้องเท่ากับเวลาที่เซอร์วิสขอมาโดยส่วนมากจะน้อยกว่าหรือเท่ากับ โดยที่ลีสที่ส่งไปให้กับเซอร์วิสนั้นเซอร์วิสสามารถรับมาได้จากเมธอด getLease() ของ ServiceRegistration ออบเจ็กต์ โดยโครงสร้างของลีสนั้นมีเมธอด ต่างๆดังนี้

```
package net.jini.core;

public interface Lease {
    void cancel();
    long getExpiration();
    void renew(long duration);
}
```

เวลาหมดอายุของลีสสามารถดูได้จากเมธอด getExpiration() ซึ่งคืนค่าออกมาเป็นเวลาในหน่วยมิลลิวินาที ซึ่งมีลักษณะเดียวกับเมธอด System.currentTimeMillis() เราจึงสามารถหาเวลาที่เหลือของลีสได้จาก

```
long duration = lease.getExpiration() - System.currentTimeMillis();
```

เมื่อลีสถูกทำให้หมดอายุโดยลุคอัพเซอร์วิสนั้นลุคอัพเซอร์วิสจะไม่ได้แจ้งกลับไปยังผู้ที่ถือลีสว่าขณะนี้ลีสได้หมดอายุแล้ว ซึ่งจะต้องเป็นหน้าที่ของเซอร์วิสที่จะต้องทำการ รีนิว ก่อนที่ลีสจะหมดอายุเพื่อที่จะต่ออายุของลีสซึ่งบางทีก็เป็นเรื่องที่ยุ่งยากของโปรแกรมเมอร์ในการที่จะต้องมาดูอายุของลีสว่าหมดอายุหรือหากใกล้หมดอายุก็ต้องทำการ รีนิว แล้วบางทียังอาจจะมีการผิดพลาดเนื่องจากความล่าช้าของเครือข่ายก็ได้ ในชุดพัฒนาจินี่ จึงได้มีคลาสที่มาช่วยเหลือในกระบวนการนี้ คือ LeaseRenewalManager

LeaseRenewalManager

LeaseRenewalManager จะคอยช่วยกระบวนการดูแลไม่ให้ลีสหมดอายุ ซึ่งจะทำงานโดยอัตโนมัติเมื่อถึงเวลาโครงสร้างของ LeaseRenewalManager มีดังนี้

```
package net.jini.lease;

public Class LeaseRenewalManager {
    public LeaseRenewalManager();
    public LeaseRenewalManager(Lease lease, long expiration,
                               LeaseListener listener);
```

```
    public void renewFor(Lease lease, long duration,
                         LeaseListener listener);
    public void renewUntil(Lease lease, long expiration,
                           LeaseListener listener);
    // etc
}
```

LeaseRenewalManager สามารถดูแลลีสได้ทีละหลายๆตัวโดยที่การส่งลีสให้ LeaseRenewalManager ดูแลนั้นสามารถส่งไปทาง คอนสตรัคเตอร์หรือส่งโดยเรียกใช้เมธอด renewFor() และ renewUntil() เวลาที่ร้องขอไปจะเป็นหน่วยมิลลิวินาทีเหมือนกันในทุก เมธอด โดย expiration หมายความว่าคือเวลาที่ลีสจะหมดอายุ ส่วน duration หมายความว่า เวลานับจากนี้ไปเท่าใด

แต่อย่างไรก็ตามหาก LeaseRenewalManager ไม่สามารถ รีนิวลีสได้ด้วยสาเหตุต่างๆเช่นลีสหมดอายุไปแล้วหรือเครือข่ายขัดข้อง ก็จะมี Exception ส่งออกมาซึ่งจะถูกจับไว้ด้วย LeaseRenewalManager แล้วทำการเรียกใช้เมธอด notify() ของ LeaseListener ที่ลงทะเบียนไว้เมื่อตอนสร้าง LeaseRenewalManager

5.1.2 การใช้ลีสระหว่าง เซอร์วิสของจินี่ และ Client

กระบวนการการใช้ลีสในการดูแลไคลเอ็นต์ว่ายังคงเชื่อมต่อ อยู่หรือไม่นั้นสามารถนำมาใช้กับเซอร์วิสใดๆก็ได้ ซึ่งการที่เซอร์วิสจะใช้กระบวนการ ลีสซิ่ง สำหรับ ดูแลไคลเอ็นต์ของตนนั้นเซอร์วิสจะต้องสร้างกระบวนการ ลีสซิ่ง ขึ้นมาเอง เนื่องจากว่าตัวกระบวนการนั้นไม่ได้เป็นส่วนหนึ่งของจินี่ API ตัวชุดพัฒนาจินี่นั้นมีเพียง อินเตอร์เฟสของคลาส ต่างๆที่จำเป็นต้องใช้มาให้เท่านั้น ส่วนลีสและผู้ดูแลลีสนั้นจึงต้องสร้างขึ้นมาเองโดย สร้างตามอินเตอร์เฟสที่มีมาให้ คือ AbstractLease และ LandLord package

5.1.2.1AbstractLease

AbstractLease คือ แอ็บสแทรกต์คลาส ซึ่ง สร้างตามอินเตอร์เฟส Leaseใช้ในการสร้างออบเจ็กต์ของ Lease โดยวิธีการใช้จะต้องสร้างซับคลาสของ AbstractLease อีกทีหนึ่งเนื่องจากคลาสนี้เป็นแอ็บสแทรกต์คลาส

```
package com.sun.jini.lease;

public abstract class AbstractLease implements Lease,java.io.Serializable {
    protected AbstractLease(long expiration);
    public long getExpiration();
    public int getSerialFormat();
    public void setSerialFormat(int format);
    public void renew(long duration);
    protected abstract long doRenew(long duration);
}
```

คลาสนี้ทำการสร้างตามอินเตอร์เฟส Lease โดยตรงโดยวิธีการนำไปใช้มีเงื่อนไข 3 อย่างคือ

1) คอนสตรัคเตอร์ของคลาส นี้ประกาศว่าเป็นแบบ protected เพราะฉะนั้นการตั้งอายุของลีสนั้นสามารถตั้งได้โดย คลาส ที่เป็น ซับคลาสของคลาสนี้เท่านั้น

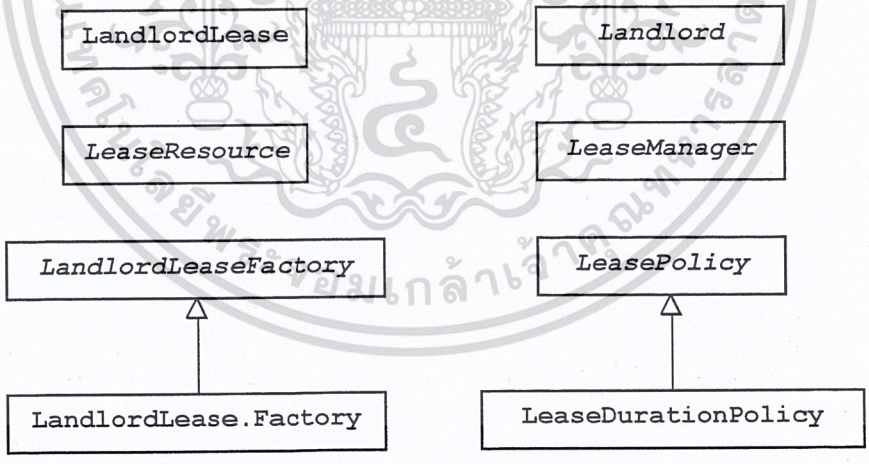
2) เนื่องจากเมธอด renew() จะทำการเรียกต่อไปยังเมธอด doRenew() เพื่อทำการ รีนิว ซึ่ง doRenew() เป็น แอ็บสแทรกต์ เมธอด ดังนั้นจึงเป็นการบังคับให้ซับคลาสจะต้องสร้างวิธีการในการ รีนิว ด้วย การโอเวอร์ไรด์เมธอดนี้
3) AbstractLease ไม่ได้ ไม่ได้มีการสร้างเมธอด cancel() ของอินเตอร์เฟส Lease เหลือไว้ให้ ซับคลาส เป็นผู้ สร้าง

5.1.2.2 แพคเกจ LandLord

แพคเกจนี้สร้างขึ้นเพื่อช่วยในการสร้างระบบลีสที่ซับซ้อนมากขึ้น โดยแพคเกจนี้ประกอบด้วย คลาส และอินเตอร์เฟสโดยที่แพคเกจนี้ยังไม่สมบูรณ์ในตัวเอง แต่เหลือบางส่วนไว้เป็นอินเตอร์เฟสไว้ สำหรับแอพพลิเคชันนำไปสร้างเป็นระบบของตัวเอง

อินเตอร์เฟส LandLord จะใช้สำหรับการสร้างออบเจ็กต์ ซึ่งทำหน้าที่ดูแลและจัดการลีสได้หลายๆตัวโดยลีสแต่ละตัวที่ LandLord สร้างขึ้นจะถูกอ้างอิงโดย ออบเจ็กต์ ซึ่งลีสแต่ละตัวจะมี ออบเจ็กต์ที่ใช้อ้างอิงโดยไม่ซ้ำกันในแต่ละ LandLord โดยปรกติแล้วมักจะเป็น ออบเจ็กต์ของคลาส Integer

เมื่อมีการร้องขอลีสตัวใหม่ Landlord จะใช้ landlord lease factory เป็นตัวสร้างลีส (LandLord สามารถสร้างเองก็ได้ แล้วแต่วิธีการสร้าง LandLord ว่าจะให้ทำอย่างไร) เมื่อลีสต้องการจะ แคนเซิล หรือ รีนิว ก็จะให้ Landlord เป็นผู้จัดการให้ โดยที่ไคลเอ็นต์ไม่จำเป็นต้องคุยกับ LandLord โดยตรง แต่กระทำผ่านลีสออบเจ็กต์



รูปที่ 5-2 คลาสและอินเตอร์เฟสบางส่วนในแพคเกจ Landlord

ด้านบนนี้คือแผนภาพคลาสและอินเตอร์เฟสต่างๆที่อยู่ใน แพคเกจ นี้ โดยอินเตอร์เฟสจะแสดงโดยตัวอักษรเอียง ส่วนคลาสจะแสดงโดยตัวอักษรธรรมดา โดยรายละเอียดของคลาสหรืออินเตอร์เฟสเหล่านี้สามารถดูได้จากคู่มือ API ของจินี่

com.sun.jini.lease.landlord.LeaseResource (อินเตอร์เฟส)

อินเตอร์เฟส นี้จะถูกนำไปสร้างสำหรับ คลาส ที่ทำการเก็บข้อมูลของลีสต่างๆ (เช่น session data ออบเจ็กต์) สร้างขึ้นเพื่อให้สามารถรวบรวมข้อมูลต่างๆของลีสไว้เป็น ออบเจ็กต์ เดียว ซึ่ง คลาส นี้จะจำเป็นต้องใช้เมื่อมาการใช้ LeaseManager หรือ LeasePolicy

com.sun.jini.lease.landlord.LeasePolicy (อินเตอร์เฟส)

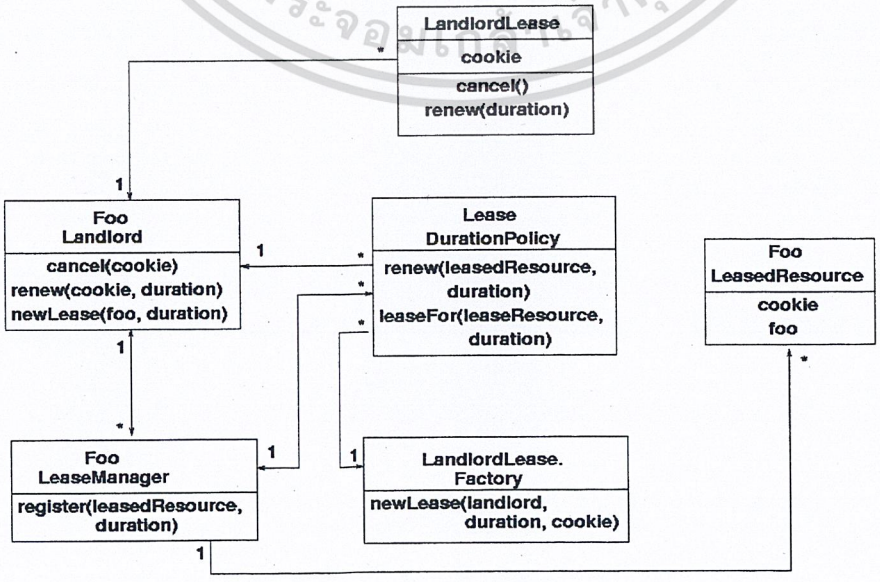
อินเตอร์เฟส นี้จะถูกนำไปสร้าง สำหรับ คลาส ที่ทำการจัดการกับ LeaseResource ออบเจ็กต์ โดยที่จะเก็บวิธีการรายละเอียดในการ รีนิว หรือแคนเซิล Lease สำหรับ LeaseResource ต่างๆที่ส่งมาให้จัดการ ซึ่งกระบวนการหลักๆที่คลาสนี้จัดการนั้นก็คือการแก้ไขเวลาหมดอายุของลีสที่อยู่ภายใน LeaseResource ที่ส่งมาให้มันทำงาน

com.sun.jini.lease.landlord.LeaseManager(อินเตอร์เฟส)

อินเตอร์เฟส นี้จะถูกนำไปสร้าง สำหรับคลาสที่ทำการจัดการกระบวนการหลังจากที่ระบบลีสซิ่งได้มีการทำงานต่างๆ เช่น มีการสร้างลีสใหม่ หรือลีสที่มีอยู่แล้วถูก รีนิว ซึ่ง LeaseManager นี้จะใช้จัดกระกระบวนการเช่น เก็บรักษา LeaseResource ที่เกิดใหม่ทั้งหมด ทำการทำลาย LeaseResource ที่หมดอายุแล้ว และกระบวนการต่างๆอีกมากมาย แล้วแต่จะสร้างขึ้นอย่างไรก็ได้ โดยจะมีเมธอด register() และ renew() สำหรับรับการแจ้งเตือนว่ามีลีสที่เกิดใหม่หรือมีลีสที่ถูกรีนิว ซึ่ง LeaseManager นี้อาจจะถูกแจ้งเตือนโดย Landlord ก็ได้

com.sun.jini.lease.landlord.LeaseDurationPolicy (คลาส)

คลาส นี้สร้างขึ้นโดยสร้างมาจากอินเตอร์เฟส LeasePolicy ซึ่งออบเจ็กต์นี้จะถูกใช้โดย Landlord ในการสร้างลีสใหม่ หรือทำการ รีนิวลีสโดยที่สามารถส่ง LandlordLeaseFactory และ LeaseManager ให้เป็นพารามิเตอร์สำหรับคอนสตรัคเตอร์ได้เพื่อให้ไว้ใช้สำหรับการสร้างลีสใหม่ และจัดการลีสที่เกิดขึ้น จากแผนภาพ UML ด้านล่างนี้ แสดงตัวอย่างการสร้างระบบลีสซิ่งด้วย แพคเกจ Landlord



รูปที่ 5-3 ตัวอย่างการออกแบบกระบวนการลีสซิ่ง

- Landlord (FooLandlord) สามารถจัดการกับลีสหลายๆตัว แต่ว่าสามารถอ้างอิงถึงลีส(LandlordLease) แต่ละตัวได้จาก คุกกี้ (Cookie) ซึ่งลีสแต่ละตัวนั้นจะมีตัวอ้างอิงไปยัง Landlord ของตัวเอง และเรียกใช้เมธอดของ Landlord โดยใช้คุ๊กกี้ของตัวเอง
- การที่ไคลเอ็นต์จะกระทำใดๆกับลีส (เช่น รีนิวหรือแคนเซิล) การร้องขอนั้นจะถูกส่งต่อไปยัง Landlord (FooLandlord) เนื่องมาจากการจะจัดการกับลีสอย่างไรนั้นจะต้องขึ้นอยู่กับ วิธีการและการตัดสินใจของ Landlord โดยที่ Landlord จะส่งไปให้กับ LeaseDurationPolicy เป็นตัวจัดการ
- LeaseFactory สามารถถูกใช้ได้จากหลายๆ LeasePolicy
- Landlord หนึ่งสามารถมีได้หลาย LeasePolicy และสามารถเลือกได้ว่าจะใช้ Policy ไหน
- LeaseManager หนึ่งสามารถถูกใช้ได้จาก หลายๆ LeasePolicy โดยส่งผ่านเป็นคอนสตรัคเตอร์ให้กับ LeaseDurationPolicy

5.2 ระบบรีโมตอีเวนต์

แนวคิดของการใช้อีเวนต์นั้นเป็นแนวคิดที่มีมานานแล้ว และใช้อยู่ในภาษาทั่วไป พื้นฐานของแนวความคิดนี้คือ ออบเจ็กต์หนึ่งสามารถที่จะแจ้งให้กับ ออบเจ็กต์อื่นๆทราบถึงการเปลี่ยนแปลงบางอย่างกับตัวมันได้ ซึ่งวิธีการนี้ทำให้ไคลเอ็นต์ไม่จำเป็นต้องคอยไปตรวจสอบที่เซอร์วิสอยู่ตลอดเวลาว่ามีเหตุการณ์อะไรเกิดขึ้นหรือไม่ ทำให้ช่วยลดการใช้ ซีพียู ลง ลดปริมาณข้อมูลที่จะต้องใช้ในการสื่อสารลง

แนวความคิดของอีเวนต์เริ่มใช้ครั้งแรกใน ข้อกำหนดของจาวาบีน (JavaBeans Specification) ซึ่งใน ข้อกำหนดนี้อีเวนต์ออบเจ็กต์ ทั้งหมดจะอินเฮอริตมาจากคลาส java.util.EventObject แล้วถูกส่งไปให้กับออบเจ็กต์ที่ทำหน้าที่เฝ้าดู (Listener) ซึ่งถูกสร้างจากอินเตอร์เฟส java.util.EventListener ซึ่งใน ข้อกำหนดของจาวาบีน ผู้ที่จะต้องการรับอีเวนต์จะต้องสร้างออบเจ็กต์ที่เป็นตัวดักฟัง (Listener) แล้วส่งออบเจ็กต์นี้ให้กับออบเจ็กต์ที่ส่งอีเวนต์ออกมาซึ่งออบเจ็กต์นั้นจะเรียกใช้ เมธอด ของ ออบเจ็กต์ตัวดักฟังที่เก็บรักษาไว้อยู่

จินี่ได้ขยายหลักการของอีเวนต์นี้ออกไปโดยที่ ออบเจ็กต์ตัวดักฟังนั้นได้กลายเป็น รีโมตออบเจ็กต์ ซึ่งทำให้อีเวนต์นั้นสามารถถูกส่งผ่านข้าม จาวาเวอร์ชวลมาชีนได้ โดยที่ออบเจ็กต์ที่เป็นตัวดักฟังในโมเดลใหม่นี้นั้นจะต้องสร้างขึ้นตามอินเตอร์เฟส net.jini.core.event.RemoteEventListener และตัวอีเวนต์ออบเจ็กต์ นั้นจะต้องอินเฮอร์ริตมาจากคลาส net.jini.core.event.RemoteEvent โดยที่กระบวนการในการส่งอีเวนต์นั้นก็ได้ใช้อาร์เอ็มไอ เป็นหลักโดยที่ RemoteEventListener ที่อยู่ที่ฝั่งผู้ส่งอีเวนต์นั้นจะเป็นสตับของอาร์เอ็มไอ ทำให้สามารถสามารถส่งอีเวนต์กลับไปยังผู้รับอีเวนต์ได้

5.2.1 RemoteEvent

ในอีเวนต์โมเดลของ AWT นั้น มีการใช้อีเวนต์แบบต่างๆมากมาย แต่ว่าใน จินี นั้นจะใช้อีเวนต์ชนิดเดียวคือ RemoteEvent และมี ซับคลาสเพียงเล็กน้อยเท่านั้น ซึ่ง RemoteEvent มี เมธอด ต่างๆดังนี้

```
package net.jini.core.event;

public class RemoteEvent {
    public long getID();
    public long getSequenceNumber();
    public java.rmi.MarshalledObject getRegistrationObject();
}
```

ซึ่งอีเวนต์ใน AWT จะมีข้อมูลของสถานะของผู้ส่งอีเวนต์ที่ซับซ้อนมาก แต่ใน จินี กลับหลีกเลี่ยงโดยส่งข้อมูลผ่านไปทางอีเวนต์แค่ที่จำเป็นเท่านั้น ทำให้อีเวนต์ออบเจ็กต์ นั้นมีขนาดเล็กและถูกส่งผ่านไปทางเครือข่ายได้ง่ายขึ้น

5.2.2 การลงทะเบียน Event

ใน จินี นั้นไม่ได้มีการกำหนดเรื่องของวิธีการในการลงทะเบียนอีเวนต์สำหรับ ออบเจ็กต์ ซึ่งแตกต่างจากอีเวนต์โมเดลอื่นๆของ จาวาซึ่งออบเจ็กต์ที่เป็นต้นกำเนิดอีเวนต์มักจะเตรียมเมธอดสำหรับออบเจ็กต์ที่ต้องการดักจับอีเวนต์เช่น

```
public void addActionListener(ActionListener listener);
```

ซึ่งผู้พัฒนาจินี จำเป็นต้องสร้างเมธอดในลักษณะนี้เองสำหรับเซอร์วิสที่ต้องการให้ไคลเอ็นต์สามารถได้รับอีเวนต์ได้โดยไคลเอ็นต์ จะต้องสร้างออบเจ็กต์ที่ทำหน้าที่ดักจับอีเวนต์แล้วส่งไปลงทะเบียนให้กับเซอร์วิส โดยที่จินีได้มีออบเจ็กต์ที่มาช่วยในการลงทะเบียนซึ่งมีชนิดเป็น EventRegistration โดยออบเจ็กต์นี้จะถูกส่งคืนให้กับออบเจ็กต์ที่ทำการลงทะเบียนอีเวนต์ทำให้เมธอด ที่ใช้ในการลงทะเบียนอีเวนต์สำหรับจินีจะมีลักษณะดังนี้

```
public RventRegistration registerForEvent(RemoteEventListener listener);
```

ซึ่งพารามิเตอร์สำหรับเมธอด นี้ก็คือ RemoteEventListener ซึ่งทำหน้าที่เป็นตัวดักจับอีเวนต์นั่นเองสำหรับโครงสร้างของ EventRegistration คือ

```
package net.jini.core.event;
import net.jini.core.lease.Lease;

public class EventRegistration implements java.io.Serializable {
    public EventRegistration(long eventID, Object source, Lease lease
                             ,long seqNum);
    public long getID();
    public Object getSource();
    public Lease getLease();
    public long getSequenceNumber();
}
```

ในการลงทะเบียนในแต่ละครั้งสำหรับจินี่จะมีอายุเพียงช่วงหนึ่งเท่านั้น เนื่องจากเหตุผลเดียวกับการลงทะเบียนเซอร์วิสที่ลุคอัพเซอร์วิสเพราะฉะนั้นจึงมีการทำกระบวนการของลีส มาใช้ในที่นี้ด้วยเช่นกัน โดยลีสสำหรับกระบวนการลงทะเบียนอีเวนต์นี้จะถูกส่งมาให้ผ่านทางออบเจ็กต์ EventRegistration ซึ่งออบเจ็กต์ที่ทำการลงทะเบียนอีเวนต์สามารถรับลีสมาจากเมธอด getLease() ของออบเจ็กต์ EventRegistration เพื่อนำไปส่งต่อให้กับ LeaseRenewalManager เพื่อดูแลและทำการรีนิวต่อไป

5.2.3 RemoteEventListener

การสร้างออบเจ็กต์ที่ทำการดักจับรีโม ตอีเวนต์จะต้องทำการสร้างโดยจะต้องสร้างตามอินเตอร์เฟสที่ชื่อ RemoteEventListener

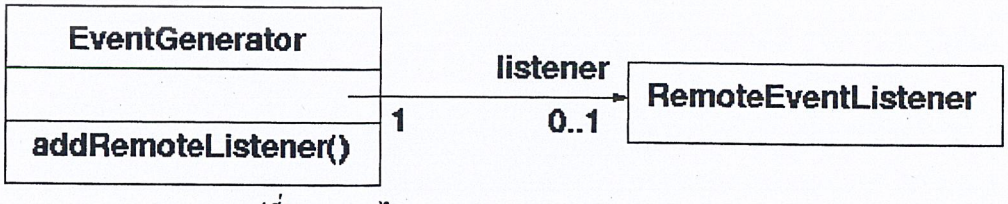
```
public interface RemoteEventListener
                extends java.rmi.Remote, java.util.EventListener {
    public void notify(RemoteEvent theEvent)
                    throws UnknownEventException,java.rmi.RemoteException;
}
```

เนื่องจากอินเตอร์เฟสนี้อินเฮอริตมาจากอินเตอร์เฟส Remote ทำให้ออบเจ็กต์ที่ดักจับรีโมตอีเวนต์ซึ่ง สร้างตามอินเตอร์เฟสนี้ทำตัวเป็น รีโมตออบเจ็กต์และจึงต้องสร้าง อาร์เอ็มไอสตับสำหรับรีโมตออบเจ็กต์ ด้วย โดยที่ออบเจ็กต์ที่เป็นต้นกำเนิดของอีเวนต์จะเรียกใช้เมธอด notify() ของ RemoteEventListener เพื่อส่งออบเจ็กต์ RemoteEvent กลับไปยังผู้ที่ต้องการรับอีเวนต์

ในการออกแบบระบบอีเวนต์สำหรับเราสามารถออกแบบให้ ออบเจ็กต์ ที่ให้กำเนิดอีเวนต์นั้นสามารถรองรับผู้ดักฟังเพียงตัวเดียว หรือจะออกแบบให้รองรับผู้ดักฟังหลายตัวก็ได้โดยมีหลักการออกแบบดังนี้

5.2.4 Single Listener

หากมีออบเจ็กต์ผู้ดักฟังเพียงตัวเดียวเราสามารถใช้ตัวแปรที่มีค่าเดียว (Single-valued variable) มารองรับออบเจ็กต์ผู้ดักฟังอีเวนต์ได้



รูปที่ 5-4 คลาสไดอะแกรมของระบบ Single Listener

โดยตัวอย่าง เมธอด ที่ใช้ทำการลงทะเบียนดักฟังอีเวนต์คือ

```
//Use landlord for create new lease
protected ServerLandlord landlord = new ServerLandlord()
protected final long defaultDuration = 10*1000  //10 Seconds
protected RemoteEventListener listener = null;
```

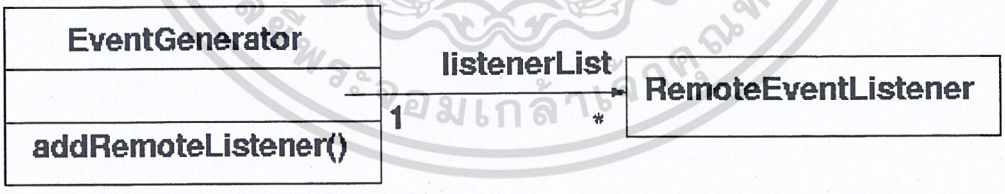
```
public EventRegistration addRemoteListener(RemoteEventListener listener)
                          throws java.util.TooManyListenersException {
    if (this.listener == null){
       this.listener = listener;
    } else { throw new java.util.TooManyListenersException(); }

    //Create new Lease for EventRefgsitration
    Lease l = landlord.newLease(defaultDuration);
    return new EventRegistration(0L, this, l, 0L);
}
```

ซึ่งลีสที่ส่งให้กับไคลเอ็นต์ Landlord จะเป็นผู้สร้าง แต่ว่าในตัวอย่างนี้ยังไม่ได้มีการจัดการกับลีสออบเจ็กต์ที่หมดอายุ และถ้าหากมีอีเวนต์เกิดขึ้น เซอร์วิส สามารถส่งอีเวนต์ให้กับไคลเอ็นต์ได้โดยส่งอีเวนต์ให้กับตัวดักฟังอีเวนต์ผ่านทาง เมธอด notify() โดยมีตัวอย่างดังต่อไปนี้

```
public void eventNotify(long eventID, long seqNum){
    if(listener == null){
       return;
    }
    //Create Remote event and send event to listener object
    RemoteEvent remoteEvent = new RemoteEvent(this, eventID, seqNum, null);
    Listener.notify(remoteEvent);
}
```

5.2.5 Multiple Listener

ในกรณีที่จะต้องรองรับออบเจ็กต์ที่ทำการดักฟังอีเวนต์หลายตัวนั้น ในการออกแบบอาจจะใช้ออบเจ็กต์ประเภท คอนเทนเนอร์มาช่วยในการเก็บรักษาออบเจ็กต์ตัวดักฟังอีเวนต์เช่น Vector, Hashtable หรืออาจจะใช้ javax.swing.event.EventListenerList มาช่วยก็สามารถทำได้



รูปที่ 5-5 คลาสไดอะแกรมของระบบ Multiple Listener

โดยตัวอย่าง เมธอด ที่ใช้ทำการลงทะเบียนดักฟังอีเวนต์ซึ่งยังไม่ได้จัดการกับลีสที่หมดอายุแล้วคือ

```
//Use landlord for create new lease
protected ServerLandlord landlord = new ServerLandlord()
protected final long defaultDuration = 10*1000  //10 Seconds
protected Vector listeners = new Vector();

public EventRegistration addRemoteListener(RemoteEventListener listener)
                          throws java.util.TooManyListenersException {
    //Add listener to Vector
    listeners.addElement(listener);

    //Create new Lease for EventRefgsitration
    Lease l = landlord.newLease(defaultDuration);
```

```
    return new EventRegistration(0L, this, 1, 0L);
}
```

และตัวอย่าง เมธอด ที่ทำการส่งอีเวนต์ให้กับตัวดักฟังคือ

```
public void eventNotify(long eventID, long seqNum){
    if(listeners.isEmpty()){
       return;
    }
    //Create Remote event object
    RemoteEvent remoteEvent = new RemoteEvent(this, eventID, seqNum, null);

    //Send RemoteEvent to all listeners
    RemoteEventListener listener = null;
    For(Enumeration e = listeners.elements();e.hasMoreElements();){
       listener = (RemoteEventListener)e.nextElement();
       Listener.notify(remoteEvent);
    }
}
```

5.3 รีโมตอีเวนต์สำหรับลุคอัพเซอร์วิส

ในกรณีที่ ไคลเอ็นต์ กำลังทำงานอยู่ ไคลเอ็นต์สามารถได้รับข้อมูลต่างๆจากลุคอัพเซอร์วิสได้ เช่น มีเซอร์วิส ที่ต้องการมาลงทะเบียนใหม่ เซอร์วิสที่กำลังใช้อยู่ได้ออกจากระบบจินีไปแล้ว หรือเซอร์วิสที่มีอยู่มีการเปลี่ยนแปลงแอตทริบิวต์

ซึ่งกระบวนการที่กล่าวมานี้นั้นลุคอัพเซอร์วิส จะใช้หลักการของ รีโมตอีเวนต์ เพื่อส่งอีเวนต์ไปบอกยังไคลเอ็นต์ของลุคอัพเซอร์วิส กระบวนการที่จะทำให้ ไคลเอ็นต์ สามารถรับอีเวนต์จากลุคอัพเซอร์วิสได้ก็เหมือนกับกระบวนการของ รีโมตอีเวนต์ดังที่กล่าวมาแล้วข้างต้น ก็คือ

- สร้างออบเจ็กต์สำหรับดักฟังอีเวนต์
- ลงทะเบียนออบเจ็กต์นั้นกับ ต้นกำเนิดของอีเวนต์ซึ่งกรณีนี้ก็คือ ลุคอัพเซอร์วิสนั่นเอง

ตัวอย่างของคลาสสำหรับออบเจ็กต์ตัวดักฟังคือ

```
import java.rmi.*;
import java.rmi.server.*;
import net.jini.core.lookup.*;
import net.jini.core.event.*;

public class LookupServiceListener extends UnicastRemoteObject
                                   implements RemoteEventListener {

    public void notify(RemoteEvent event) throws UnknownEventException
                                                 RemoteException{
       if(!(event instanceof ServiceEvent)){
           Throw new UnknownEventException("LookupServiceListener");
       }

       ServiceEvent sevent = (ServiceEvent)event;
       ServiceItem item = sevent.getServiceItem();
       if(sevent.getTransition()==ServiceRegistrar.TRANSITION_NOMATH_MATCH){
           System.out.println("Found new Service");
       }
    }
}
```

สำหรับอีเวนต์ที่ได้รับมานั้น ไคลเอ็นต์ สามารถรู้ถึงเหตุการณ์ที่ ลุคอัพเซอร์วิส ส่งมาได้ว่าคืออะไรโดยดึงข้อมูลมาจากเมธอด getTransition() ของออบเจ็กต์ ServiceEvent ซึ่งข้อมูลที่ได้เป็นข้อมูลที่แสดงเหตุการณ์สำหรับ ลุคอัพเซอร์วิส มีข้อมูลต่างๆดังนี้

TRANSITION_NOMATH_MATH

หมายความว่ามี เซอร์วิส ซึ่งเหมือนกับเทมเพลตที่ต้องการมาลงทะเบียนใหม่

TRANSITION_MATH_NOMATH

หมายความว่า เซอร์วิส ที่เคยค้นหาพบได้ถูกนำออกไปจากระบบแล้ว หรือมีการเปลี่ยนแอตทริบิวต์ของเซอร์วิส นั้นจนทำให้ไม่เหมือนกับเทมเพลตที่ต้องการอีกต่อไป

TRANSITION_MATH_MATH

หมายความว่า เซอร์วิสได้มีการเปลี่ยนแปลงแอตทริบิวต์แต่ว่ายังคงเข้ากันได้กับเทมเพลตที่ต้องการ

ขั้นตอนในการลงทะเบียนออบเจ็กต์ผู้ดักฟัง จะกระทำผ่านเมธอด notify() ของ ServiceRegistrar ออบเจ็กต์ที่ได้มาจากลุคอัพเซอร์วิส โดยที่เมธอดนี้ต้องการพารามิเตอร์ 5 ตัวสำหรับเป็นข้อมูลของเซอร์วิสที่ต้องการค้นหา คือ

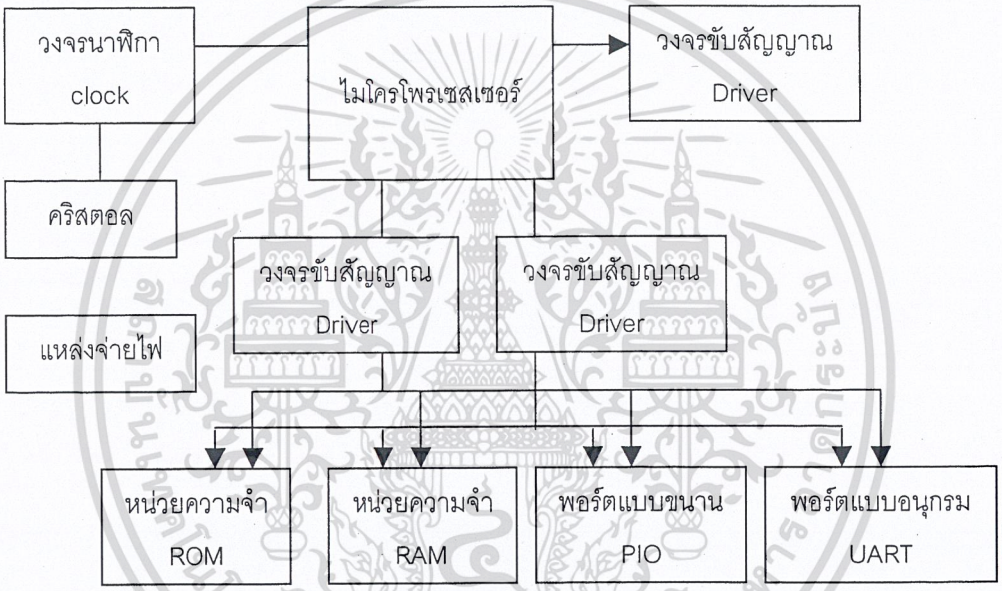
- เทมเพลตออบเจ็กต์ สำหรับ เซอร์วิส ที่ต้องการค้นหา
- ลักษณะการเปลี่ยนแปลงที่ต้องการจะรู้
- RemoteEventListener
- มาแชลด์ออบเจ็กต์ (Marshaled Object)
- ระยะเวลาที่ต้องการจะรับอีเวนต์

ซึ่งหลังจากเรียกใช้เมธอดนี้แล้ว ก็จะคืนค่าออกมาเป็นออบเจ็กต์ของคลาส EventRegistration ซึ่งจะนำมาใช้เช่นเดียวกับกระบวนการของ RemoteEvent ทั่วไปคือนำลีสที่ได้จาก EventRegistration ออบเจ็กต์ ไปส่งให้กับ LeaseRenewalManager เพื่อดูแลต่อไป

บทที่ 6
ทฤษฎีของฮาร์ดแวร์

6.1 ไมโครคอนโทรลเลอร์ ในตระกูล 8051

ไมโครคอนโทรลเลอร์ในตระกูล 8051 เป็นไมโครคอนโทรลเลอร์ที่นิยมใช้อย่างแพร่หลาย ในงานควบคุมเครื่องจักร และควบคุมอุปกรณ์เพื่อการอินเตอร์เฟสกับคอมพิวเตอร์ เพราะมีการใช้งานที่ง่าย แผนภาพของ หน่วยการทำงานดังนี้



รูปที่ 6-1 แสดงหน่วยการทำงานทั่วไปของระบบไมโครโพรเซสเซอร์

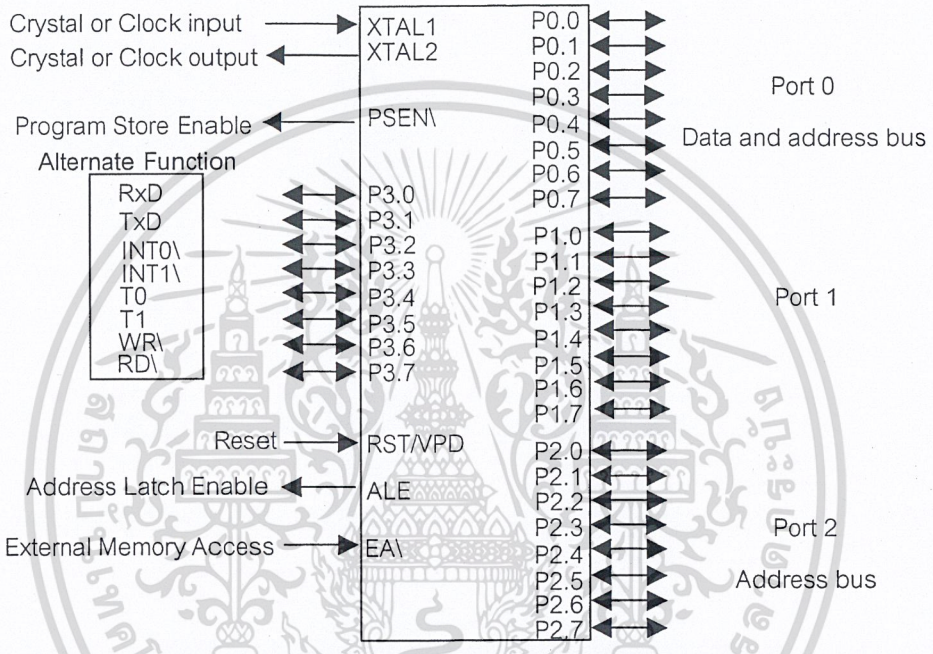
คุณลักษณะของไมโครคอนโทรลเลอร์ 8051

- หน่วยประมวลผลกลาง 8 บิต
- หน่วยประมวลผลสำหรับข้อมูลแบบบิต
- ความสามารถในการอ้างตำแหน่งของหน่วยความจำโปรแกรม 64 กิโลไบต์
- ความสามารถในการอ้างอิงตำแหน่งความจำข้อมูล 64 กิโลไบต์
- หน่วยความจำโปรแกรมภายในขนาด 4 กิโลไบต์แบบ EPROM (เบอร์ 8751) หรือแบบ ROM(เบอร์ 8051)
- หน่วยความจำแบบ RAM ภายในจำนวน 128 ไบต์
- พอร์ตอินพุต/เอาท์พุทแบบขนานจำนวน 32 เส้น ซึ่งสามารถแยกทำงานได้อย่างอิสระ
- วงจรนับ/จับเวลาขนาด 16 บิต จำนวนสองวงจร
- วงจรสื่อสารแบบอนุกรมแบบฟูลดูเพล็กซ์ (Full Duplex)

- วงจรควบคุมการอินเทอร์รัปท์จากแหล่งสัญญาณ 6 ประเภท พร้อมการกำหนดลำดับความสำคัญได้สองระดับ
- วงจรออสซิลเลเตอร์ภายใน

6.1.1 ลักษณะของไมโครคอนโทรลเลอร์ 8051

โดยทั่วไป ไมโครคอนโทรลเลอร์ ในตระกูลนี้จะมีรูปร่างไอซีแบบ ดูอัล อินไลน์ เพกเกจ (DIP) ขนาด 40 ขาดังแสดงในภาพ



รูปที่ 6-2 แสดงการกำหนดหน้าที่ขาสัญญาณของไอซี 8051

แต่ละขาสัญญาณจะมีหน้าที่ชัดเจน มีบางขาอาจมีหน้าที่มากกว่าหนึ่งอย่าง เช่นในพอร์ต P3 ซึ่งจะทำหน้าที่ในหน้าที่หนึ่งได้เพียงอย่างเดียวขึ้นกับการเชื่อมต่อวงจร

6.1.2 การใช้งานพอร์ตของไมโครคอนโทรลเลอร์ 8051

6.1.2.1 การใช้งานพอร์ตเป็นการอินพุท

ต้องเริ่มด้วยการส่งข้อมูลที่มีค่าเป็น 1 ออกมาทางบิตของพอร์ตนั้นก่อนเป็นอันดับแรก เพื่อหยุดการทำงานของทราสซิสเตอร์ที่ทำหน้าที่ขับสัญญาณเอาท์พุทของบิตนั้น ทำให้ขาสัญญาณของบิตถูกต่อเข้ากับตัวต้านทานซึ่งทำหน้าที่ พูลอัพ(Pull-up) ภายในซึ่งมีผลให้บิตนั้น ๆ ของพอร์ต 1 , 2 และ3 เป็นภาวะของลอจิกสูง ตัวต้านทานนี้มีค่าประมาณ 50 กิโลโอห์ม ซึ่งเป็นค่าที่สูงมาก และทำให้อุปกรณ์ภายนอกสามารถขับสัญญาณของพอร์ตเหล่านี้เป็นลอจิกต่ำได้ง่าย สำหรับบิตของพอร์ต 0 นั้นแม้ว่าจะมีหลักการทำงาน คล้ายคลึงกับบิตของพอร์ตอื่น ๆ แต่เนื่องจากการที่ไม่มีตัวต้านทาน ทำหน้าที่พูลอัพ ภายในไว้

ทำให้เมื่อทรานซิสเตอร์ที่ทำหน้าที่ขับสัญญาณเอาท์พุทนั้นหยุดทำงาน ก็จะเป็นผลให้ขาสัญญาณนี้อยู่ในสภาวะอิมพีแดนซ์สูงแทน

6.1.1.2 การใช้งานพอร์ตเป็นการเอาท์พุท

เมื่อมีการส่งข้อมูลที่มีค่าเป็น 0 ให้กับแต่ละบิตของพอร์ตทุกพอร์ต ข้อมูลนี้จะถูกส่งให้กับ ฟลิปฟลอป(Flip-Flop) ซึ่งจะค้างค่านี้ไว้ และมีผลทำให้ทรานซิสเตอร์ที่ทำหน้าที่ขับสัญญาณเอาท์พุทนั้นทำงานดังนั้นขาสัญญาณก็จะมีสภาวะลอจิกเป็นต่ำด้วย

ส่วนการส่งข้อมูลที่มีค่าเป็น 1 ออกมานั้น ในกรณีนี่เป็นการทำงานในแต่ละบิตของพอร์ต 1 2 หรือ 3 จะทำให้ทรานซิสเตอร์ที่ทำหน้าที่ ขับสัญญาณเอาท์พุทนั้นหยุดทำงาน มีผลทำให้ขาของสัญญาณเป็นลอจิกสูงตัวต้านทานที่พูลอัพ อยู่ภายในนั้น แต่สำหรับการทำงานในแต่ละบิตของพอร์ต 0 นั้น จะมีผลที่แตกต่างกันออกไป โดยขาสัญญาณจะเป็นสภาวะอิมพีแดนซ์สูงแทน เนื่องจากไม่มีตัวต้านทานภายในเชื่อมต่ออยู่นั่นเอง ดังนั้นการใช้งานพอร์ต 0 เป็นการเอาท์พุทข้อมูล จึงจำเป็นต้องใช้ตัวต้านทานภายนอกพูลอัพ สัญญาณไว้กับลอจิกสูงแทน

6.1.3 ระบบอินเทอร์รัปท์ของ 8051

ใน 8051 มีอินเทอร์รัปท์อยู่ 2 ประเภทคือ อินเทอร์รัปท์ภายใน และภายนอก โดยอินเทอร์รัปท์ภายในจะเกิดภายในตัว 8051 เอง ได้แก่สัญญาณจาก ไทเมอร์แฟลก 0 (TF0) ไทเมอร์แฟลก 1(TF1) และพอร์ตอนุกรม สำหรับอินเทอร์รัปท์ภายนอกเกิดจากสัญญาณที่กระตุ้นเข้ามาที่ขาสัญญาณ INT0 และ INT1 ส่วนการที่ตัวไมโครคอนโทรลเลอรร์จะยอมให้มีการทำงานเมื่อเกิดอินเทอร์รัปท์หรือไม่ ต้องโปรแกรมไปที่ รีจิสเตอร์อินเทอร์รัปท์ อีนาเบิ้ล (IE) ถ้ามีอินเทอร์รัปท์จากหลาย ๆ ทางเข้ามาพร้อมกันสามารถจัดลำดับความสำคัญของอินเทอร์รัปท์ต่าง ๆ โดยโปรแกรมไปที่ อินเทอร์รัปท์ไพออริตี้(IP) มีรายละเอียดดังนี้

6.1.3.1 อินเตอร์รัปท์ อีนาเบิ้ล

เป็นรีจิสเตอร์ที่สามารถเข้าถึงข้อมูลระดับบิต ใช้สำหรับกำหนดค่าว่าถ้าเกิดการอินเทอร์รัปท์จากแหล่งต่าง ๆ จะทำอินเทอร์รัปท์เหล่านั้นหรือไม่

บิต 7							บิต 0
EA	X	ET2	ES	ET2	EX1	ET0	EX0

รูปที่ 6-3 แสดงบิตต่าง ๆ ของรีจิสเตอร์อินเตอร์รัปท์ อีนาเบิ้ล

บิต	ชื่อบิต	ตำแหน่งบิต	รายละเอียด
IE.7	EA	AFH	ถ้าเซตยอมให้มีการอินเทอร์รัปท์
IE.6	-	AEH	ไม่ใช้งาน
IE.5	ET2	ADH	อีนาเบิ้ล อินเทอร์รัปท์จากไทเมอร์2 (8052)
IE.4	ES	ACH	อีนาเบิ้ล อินเทอร์รัปท์จากพอร์ตอนุกรม
IE.3	ET1	ABH	อีนาเบิ้ล อินเทอร์รัปท์จากไทเมอร์ 1
IE.2	EX1	AAH	อีนาเบิ้ล อินเทอร์รัปท์จากขาสัญญาณ INT0
IE.1	ET0	A9H	อีนาเบิ้ล อินเทอร์รัปท์จากไทเมอร์ 0
IE.0	EX0	A8H	อีนาเบิ้ล อินเทอร์รัปท์จากขาสัญญาณ INT0

ตารางที่ 6-1 แสดงรายละเอียดของรีจิสเตอร์อินเทอร์รัปท์อีนาเบิ้ลในบิตต่าง

6.1.3.2 รีจิสเตอร์อินเทอร์รัปท์ไพออริตี้

เป็นรีจิสเตอร์ที่สามารถเข้าถึงข้อมูลระดับบิตใช้จัดลำดับความสำคัญของการอินเทอร์รัปท์ซึ่งสามารถจัดได้สองระดับ ถ้าเป็น 1 หมายความว่ามีความสำคัญสูงสุด ถ้าเป็น 0 หมายความว่ามีลำดับความสำคัญต่ำสุด

ลำดับ	อินเทอร์รัปท์
1 (สูงสุด)	IE0
2	TF0
3	IE1
4 (ต่ำสุด)	พอร์ตอนุกรม

ตารางที่ 6-2 แสดงลำดับการตั้งค่าความสำคัญของอินเทอร์รัปต์

บิต	ชื่อบิต	ตำแหน่งบิต	รายละเอียด
IP.7	-	-	ไม่ใช้งาน
IP.6	-	0BDH	ไม่ใช้งาน
IP.5	PT2	0BCH	ใช้กับ ไทเมอร์2(8052)
IP.4	PS	0BBH	ใช้กับพอร์ตอนุกรม
IP.3	PT1	0BAH	ใช้กับไทเมอร์1
IP.2	PX1	0B9H	ใช้กับอินเทอร์รัปท์จาก INT1
IP.1	PT0	0B8H	ใช้กับไทเมอร์0
IP.0	PX0	0B7H	ใช้กับอินเทอร์รัปท์จาก INT0

ตารางที่ 6-3 แสดงบิตและหน้าที่ต่าง ๆ ของรีตจิสเตอร์อินเทอร์รัปต์ไพออริตี้

จากตารางแสดงการอินเทอร์รัปท์จากแหล่งต่าง ๆ ที่มีผลกับไมโครคอนโทรลเลอร์ ตัวไมโครคอนโทรลเลอร์จะต้องได้รับการเซตค่า บิตต่าง ๆ ในรีจิสเตอร์อินเทอร์รัปท์ อีนาเบิ้ล เป็นาการกำหนดว่าจะให้อินเทอร์รัปท์ใดเกิดขึ้นได้บ้าง จากตารางยังแสดงให้เห็นอีกว่าเมื่อมีการอินเทอร์รัปท์พเข้ามาจะมีผลต่อแฟลกใด เช่นถ้า สัญญาณ INT0 เป็น 1 บิต IE0 จะเป็น 1 หมายความว่าถูกอินเทอร์รัปท์ โดยแฟลกต่าง ๆ ที่มีผลต่อการถูกอินเทอร์รัปท์แสดงให้เห็นได้ดังตาราง

อินเทอร์รัปท์	แฟลก	ประกอบอยู่ในรีจิสเตอร์
เอ็กเตอร์นอล 0	IE0	TCON.1
เอ็กเตอร์นอล 1	IE1	TCON.3
ไทเมอร์ 1	TF1	TCON.7
ไทเมอร์ 0	TF0	TCON.5
พอร์ตอนุกรม	TI	SCON.0
พอร์ตอนุกรม	RI	SCON.1
ไทเมอร์ 2	TF2	T2CON.7 (8052)
ไทเมอร์ 2	EXF2	T2CON.6 (8052)

ตารางที่ 6-4 แสดงแฟลกที่จะทำงานเมื่อถูกอินเทอร์รัปต์

จากตารางจะเห็นว่า ถ้ามีการอินเทอร์รัปท์จากภายนอกเข้ามา ตัวที่จะอินเทอร์รัปท์ 8051 คือบิตแฟลก IE0 ซึ่งอยู่ในรีจิสเตอร์ TCON ถ้ามีการสื่อสารแบบอนุกรม เมื่อข้อมูลถูกส่งไปหมดแล้วจะอินเทอร์รัปท์ 8051 ทางแฟลก TI ถ้ารับข้อมูลหมดแล้วจะอินเทอร์รัปท์ 8051 ทางบิตแฟลก RI ซึ่งอยู่ในรีจิสเตอร์ SCON และถ้าใช้ ไทเมอร์ 0 ในการนับเมื่อเกิดโอเวอร์โฟลว สามารถอินเทอร์รัปท์ 8051 ได้ทางบิต TF0

6.1.3.3 การทำงานของระบบหลังจากถูกอินเทอร์รัปท์

เมื่อไมโครคอนโทรลเลอร์ 8051 ถูกอินเทอร์รัปท์ จะต้องกระโดดไปทำโปรแกรมตอบสนองการอินเทอร์รัปท์โดยตำแหน่งที่จะกระโดดไปเรียกว่า อินเทอร์รัปท์เวกเตอร์ (Interrupt Vector) เมื่อทำโปรแกรมตอบสนองการอินเทอร์รัปท์เรียบร้อยแล้ว ไมโครคอนโทรลเลอร์ 8051 จะกระโดดกลับมายังตำแหน่งเดิม โดยก่อนที่จะกระโดดไปทำโปรแกรมตอบสนองการอินเทอร์รัปท์จะต้องเก็บค่าตำแหน่งเดิมไว้ โดยเก็บค่า โปรแกรม เคาท์เตอร์ (Program Counter) ลงหน่วยความจำที่ถูกชี้โดยรีจิสเตอร์ สแตกพอยท์เตอร์ (Stack Pointer) เมื่อทำโปรแกรมตอบสนองการอินเทอร์รัปท์เสร็จแล้วจะคืนค่าในหน่วยความจำในสแตค ให้โปรแกรมเคาท์เตอร์ ตามเดิม ค่าอินเทอร์รัปท์ของไมโครคอนโทลเลอร์ 8051 แสดงในตาราง

อินเทอร์รัปท์	อินเทอร์รัปท์เวกเตอร์
ซิสเตม รีเซต	0000H
เอ็กเตอร์นอล 0	0003H
ไทเมอร์ 0	00BH
เอ็กเตอร์นอล 1	0013H
ไทเมอร์ 1	001BH
พอร์ตอนุกรม	0023H
ไทเมอร์ 2	002BH

ตารางที่ 6-5 แสดง อินเทอร์รัปต์เวกเตอร์ ของอินเทอร์รัปต์ต่าง ๆ

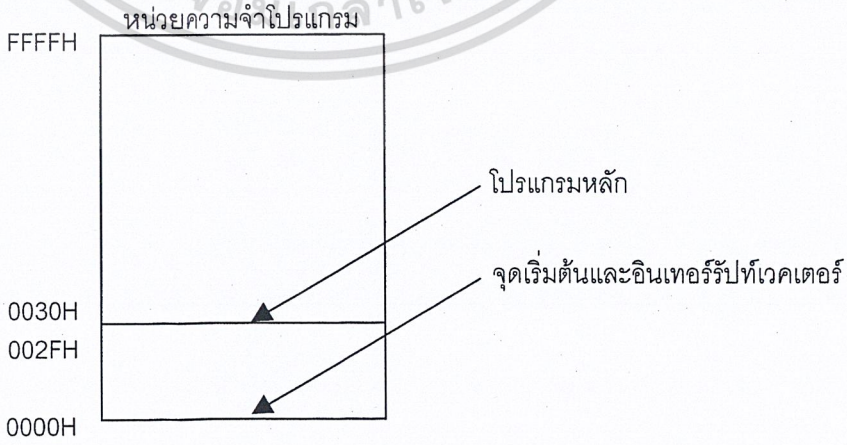
6.1.3.4 การเขียนโปรแกรมอินเทอร์รัปท์

ในการเขียนโปรแกรมหลักต้องกำหนดว่าจะให้ไมโครคอนโทลเลอร์ 8051 ถูกอินเทอร์รัปท์ด้วยอะไร และจะให้ ไมโครคอนโทลเลอร์ถูกอินเทอร์รัปท์หรือไม่ โดยการโปรแกรมค่าต่าง ๆ ในรีจิสเตอร์อินเทอร์รัปท์ อีนาเบิ้ล ถ้ามีการอินเทอร์รัปท์จากสองแหล่งขึ้นไปควรมีการจัดลำดับความสำคัญใน รีจิสเตอร์อินเทอร์รัปท์ ไพออริตี้ ดังนั้นโปรแกรมหลักจะต้องมีโปรแกรมต่อไปนี้

- ส่วนการโปรแกรมค่าในรีจิสเตอร์ อินเทอร์รัปท์ อีนาเบิ้ล
- ส่วนการโปรแกรมค่าในรีจิสเตอร์ อินเตอร์รัปไพออริตี้

สำหรับโปรแกรมตอบสนองการอินเทอร์รัปท์ถือว่าเป็นโปรแกรมย่อยโปรแกรมหนึ่งและต้องมีการจบโปรแกรมด้วยค่า RETI (Return from Interrupt)

การเขียนโปรแกรมต้องทำการเริ่มต้นโปรแกรมหลักให้พ้นจากหน่วยความจำซึ่งเก็บอินเทอร์รัปท์เวกเตอร์ เพราะโปรแกรมหลักอาจจะยาวจนทับส่วนอินเทอร์รัปท์เวกเตอร์ทำให้เกิดความผิดพลาดในการทำงานได้

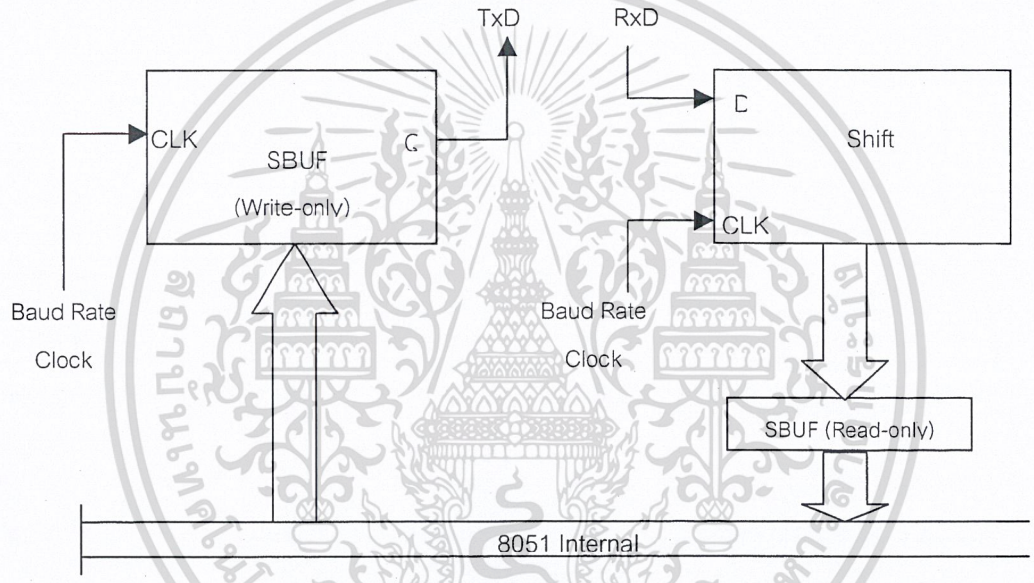


รูปที่ 6-4 แสดงการจัดตำแหน่งโปรแกรมในหน่วยความจำ

6.1.4 การส่งข้อมูลแบบอนุกรมของ 8051

ไมโครคอนโทลเลอร์ 8051 มีพอร์ตอนุกรมเพื่อการติดต่อเป็นพอร์ตอินพุท/เอาท์พุท แบบอนุกรม โดยใช้ที่พอร์ต 3 ที่ขาสัญญาณ 10 (RxD) และขา 11 (TxD) โดยการส่งข้อมูลนั้นจะส่งข้อมูลทีละบิตจนครบไบต์ ซึ่งเมื่อเปรียบเทียบกับการส่งข้อมูลแบบขนานแล้วจะช้ากว่าแต่ส่งได้ไกลกว่า

พอร์ตอนุกรมของไมโครคอนโทลเลอร์ 8051 ทำงานแบบฟูลดูเพล็กซ์ คือสามารถรับและส่งข้อมูลได้ในเวลาเดียวกัน ซึ่งจะมีบัฟเฟอร์สำหรับเก็บข้อมูลให้ใช้คือ SBUF และ SCON ซึ่งเป็นรีจิสเตอร์ที่อยู่ในรีจิสเตอร์ฟังก์ชันพิเศษ (Special Function Register) โดยรีจิสเตอร์ บัฟเฟอร์พอร์ตอนุกรม(SBUF) จะอยู่ในตำแหน่ง 99H ถ้าเขียนข้อมูลไปที่ตำแหน่งนี้จะเป็นการส่งข้อมูลออกทางพอร์ตอนุกรม และถ้าอ่านข้อมูลที่ตำแหน่งนี้ จะเป็นการอ่านข้อมูลทางพอร์ตอนุกรม



รูปที่ 6-5 แสดงการรับส่งข้อมูลระหว่างรีจิสเตอร์กับบัสภายใน

รีจิสเตอร์ที่ทำหน้าที่ความคุมการส่งข้อมูลแบบอนุกรมคือ รีจิสเตอร์ความคุมพอร์ตอนุกรม(Serail Port Control Register) อยู่ที่ตำแหน่ง 98H เป็นรีจิสเตอร์ที่เข้าถึงข้อมูลระดับบิตได้ มีรายละเอียดดังนี้

บิต	ชื่อ	ตำแหน่ง	ความหมาย
SCON.7	SM0	9FH	บิตเลือกโหมดการทำงานบิต 0
SCON.6	SM1	9EH	บิตเลือกโหมดการทำงานบิต 1
SCON.5	SM2	9DH	บิตเลือกโหมดการทำงานบิต 2
SCON.4	REN	9CH	บิตแฟลกกำหนดยอมให้มีการรับข้อมูล
SCON.3	TB8	9BH	ค่าของบิต 9 สำหรับการส่งข้อมูลในโหมด 2 และ 3 สามารถเซต และ เคลียร์โดยซอฟแวร์
SCON.2	RB8	9AH	ค่าของบิตที่ 9 เมื่อรับข้อมูลเข้ามา
SCON.1	TI	99H	บิตแฟลกแสดงการอินเตอร์รัปท์ภายหลังการส่งข้อมูลออกไปโดยจะ เซต เมื่อส่งข้อมูลไปหมดแล้วและสามารถ เคลียร์โดยซอฟแวร์
SCON.0	RI	98H	แฟลกแสดงการอินเทอร์รัปท์ภายหลังการรับส่งข้อมูลเข้ามาสามารถ เคลียร์โดยซอฟแวร์

ตารางที่ 6-6 แสดงบิตต่าง ๆ ของรีจิสเตอร์ SCON

SM0	SM1	Mode	ความหมาย	Baud Rate
0	0	0	Shift Register	เปลี่ยนแปลงไม่ได้(Oscillator Frequency / 12)
0	1	1	8-bit UART	สามารถเปลี่ยนแปลงได้โดยกำหนดจาก ไทเมอร์
1	0	2	9-bit UART	เปลี่ยนแปลงไม่ได้(Oscillator Frequency /12 หรือ 64)
1	1	3	9-bit UART	สามารถเปลี่ยนแปลงได้โดยกำหนดจาก ไทเมอร์

ตารางที่ 6-7 แสดงโหมดต่าง ๆ ของการรับส่งข้อมูลแบบอนุกรม

โหมดของการทำงาน

8-บิต ชิฟรีจิสเตอร์ (โหมด 0)

ใช้ขารับRxD ในการรับส่งข้อมูลโดยต่อกับ ชิฟรีจิสเตอร์ ภายนอก ส่วนขา TxD จะเป็นเอาท์พุท ชิฟรีจิสเตอร์ คลอก เพื่อกระตุ้นรีจิสเตอร์ภายนอกให้เลื่อนบิตถ้ามีการส่งข้อมูลหรือรับข้อมูล 8 บิต จะเริ่มที่บิตต่ำสุดก่อน โดยมีค่าอัตราการส่ง(Baud Rate) เท่ากับ 1/12 ของความถี่ที่ใช้บนชิพ

ในการส่งข้อมูลจะทำโดยเขียนข้อมูลไปที่รีจิสเตอร์ SBUF ข้อมูลจะถูกส่งออกมาทางขา RxD โดยจะสอดคล้องกับสัญญาณที่ออกมาทางขา TxD ซึ่งสัญญาณของขา TxD จะถูกส่งออกมาทุก ๆ รอบของแมชชีน

8-บิต UART สามารถเปลี่ยนแปลงอัตราการรับส่งข้อมูล(โหมด 1)

ในโหมดนี้จะส่งข้อมูลแบบ 10 บิตซึ่งประกอบด้วยบิตเริ่มต้น (เป็น 0) ข้อมูล 8 บิต และบิตจบ (เป็น 1) นอกจากนั้นยังสามารถกำหนดค่า อัตราการรับส่งข้อมูลได้โดยค่าอัตราการส่งนี้จะแปรตามตัวจับเวลาตัวที่ 1 ในโหมดนี้ จะส่งข้อมูลออกทาง TxD และรับข้อมูลทาง RxD ถ้าเป็นการรับข้อมูลเข้าตัวบิตหยุด จะเข้ามาทาง RB8 ใน SCON

ค่าอัตราการรับส่งข้อมูลที่ใช้รับส่งจะถูกกำหนดด้วย ไทเมอร์ 1 หลังจากโปรแกรมไปในตัวไทเมอร์ 1 แล้วสามารถเลือกค่า อัตราการรับส่งข้อมูลได้อีกสองค่าคือ จากไทเมอร์ 1 โอเวอร์โฟลว หาร 32 กับค่าไทเมอร์ 1 โอเวอร์โฟลว หาร16

การส่งข้อมูลจะทำการเขียนข้อมูล 8 บิตไปที่ SBUF โดยบิตที่ 9 (บิตหยุด) ให้เขียนลงใน TB8 ใน SCON จากนั้นข้อมูลจะถูกส่งออกมาทางขา TxD โดยส่ง บิตเริ่ม ออกมาก่อนตามด้วยข้อมูล 8 บิตและจบด้วยบิตจบ เมื่อข้อมูลถูกส่งออกไปหมดแล้ว อินเทอร์รัปท์แฟลก (TI) จะเป็น 1 ดังนั้นการเขียนข้อมูลลงใหม่จะต้องตรวจสอบที่บิตนี้

การรับข้อมูลจะเริ่มจากเมื่อมีการเปลี่ยนแปลงลอจิกจาก 1 เป็น 0 ทางขา RxD หมายความว่า เริ่มรับบิตเริ่มต้น จากนั้นเมื่อข้อมูลอีก 8 บิตจะถูกเก็บลงใน SBUF และบิตหยุด จะถูกเก็บในบิต RB8 ของรีจิสเตอร์ SCON เมื่อข้อมูลเข้ามาครบแล้วบิต อินเทอร์รัปท์แฟลก จะถูกเซต ดังนั้นในการอ่านข้อมูลจะอ่านได้เมื่อบิต RI ถูกเซตแล้ว เมื่ออ่านข้อมูลไปแล้วจะต้องเคลียร์บิตนี้

9-บิต UART อัตราการรับส่งข้อมูลเปลี่ยนแปลงไม่ได้ (โหมด 2)

การทำงานในโหมดนี้ไม่สามารถกำหนดค่า อัตราการรับส่งข้อมูลได้ ซึ่งค่า อัตราการรับส่งข้อมูลจะมีสองค่าคือ 1/16 และ 1/32 ของสัญญาณนาฬิกาบนชิพ การรับส่งข้อมูลจะเป็นข้อมูล 9 บิต บิตเริ่มต้นและบิตหยุด รวมทั้งหมด 11 บิต โดยข้อมูล 9 บิตจะเป็นจำนวนข้อมูล 8 บิต และบิตที่โปรแกรมได้อีก 1 บิต ซึ่งบิตนี้จะใช้เป็นพาริตี้บิต ในการส่งข้อมูลจะต้องเขียนไปที่บิต TB8 ในรีจิสเตอร์ SCON สำหรับการรับข้อมูลบิตที่ 9 จะถูกเก็บในบิต RB8

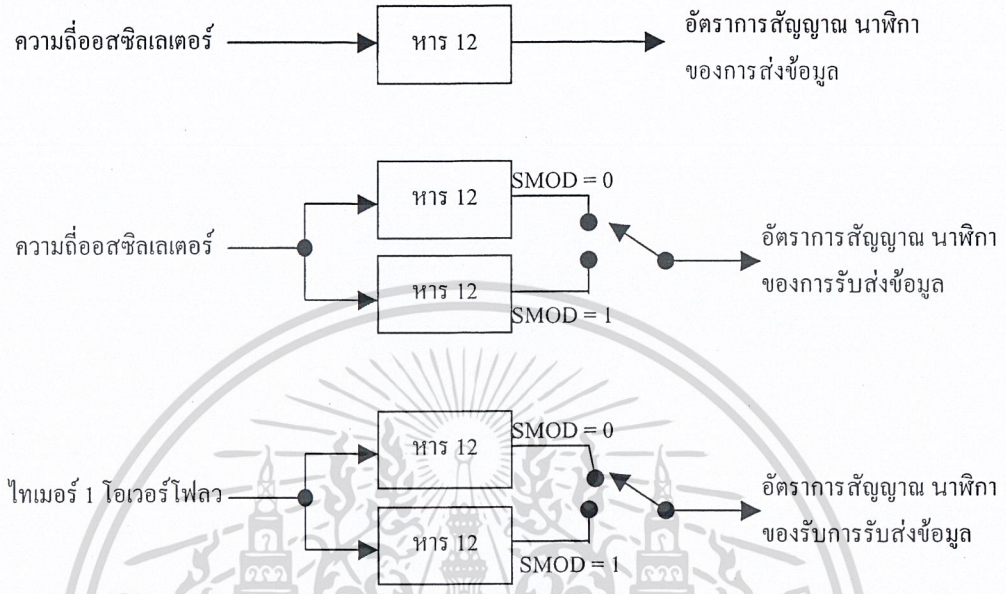
9-บิต UART สามารถเปลี่ยนแปลงอัตราการรับส่งข้อมูล(โหมด 3)

คล้ายกับโหมด 2 แต่สามารถกำหนดค่า อัตราการรับส่งข้อมูลได้ โดยการโปรแกรมไปที่ไทเมอร์ 1 หลังจากโปรแกรมแล้วยังสามารถเลือกได้อีก 2 ค่าคือ ความถี่การ โอเวอร์โฟลว ของไทเมอร์ 1 หารด้วย 16 และหารด้วย 32

การกำหนดค่าอัตราการรับส่งข้อมูลของพอร์ตอนุกรม

ในโหมด 0 และ 2 ไม่สามารถกำหนดค่าอัตราการส่งเองได้ โดยในโหมด 0 ค่าอัตราการรับส่งข้อมูลเป็น ความถี่ออสซิลเลเตอร์ หารด้วย 12 ในโหมด 2 จะมีสองค่าคือ ความถี่ออสซิลเลเตอร์หารด้วย 16 และ 64 สองค่านี้เรียกว่า SMOD 0 และ SMOD 1 ซึ่งสามารถกำหนดได้ในรีจิสเตอร์ PCON บิตที่ 7 ใน

รีจิสเตอร์ PCON นี้ไม่สามาถเข้าถึงข้อมูลระดับบิตได้ การเขียนข้อมูลลงไปทีละบิตจะต้องใช้วิธีที่เรียกว่า อ่าน-เปลี่ยนแปลง-เขียน (Read-Modify-Write) หรืออ่านค่าขึ้นมาแล้วเขียนลงไปใหม่ โหมด 1 และโหมด 3 กำหนดค่าอัตราการรับส่งข้อมูลได้ดังรูป



รูปที่ 6-6 แสดงการกำหนดอัตราการรับส่งข้อมูลในโหมดต่าง ๆ

การใช้ค่าไทเมอร์ 1 กำหนดค่า อัตราการรับส่งข้อมูล ทำได้โดยกำหนดค่าลงใน ไทเมอร์ 1 ทำได้โดยการโปรแกรมที่ TMOD ให้ทำงานแบบ 8 บิต รีโหลดอัตโนมัติ (Auto Reload) (โหมด 2) โดยเขียนค่าไปที่ TH1 ถ้าต้องการอัตราการรับส่งข้อมูล ต่ำ ๆ สามารถใช้ โหมด 16 บิต ได้โดยโปรแกรมเป็น TMOD = 0001xxxxB ค่าอัตราการรับส่งข้อมูล ที่ส่งออกมาจะมีค่าเท่ากับ ความถี่ของไทเมอร์ 1 โอเวอร์โฟลว หารด้วย 32 (หรือหารด้วย 16 ถ้า SMOD=1)

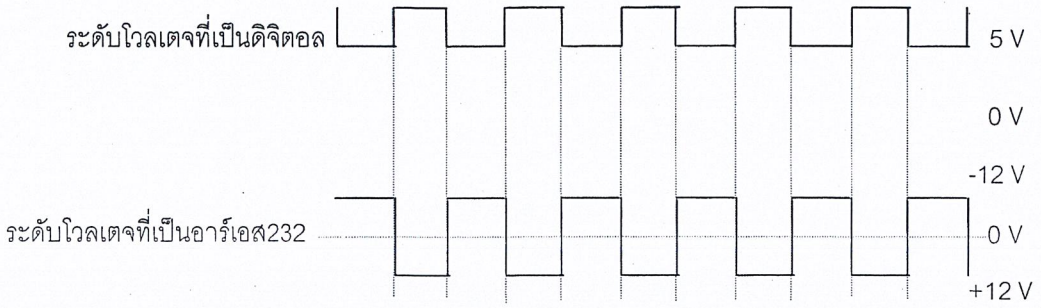
6.2 การสื่อสารในรูปแบบพอร์ตอนุกรมโดยใช้อินฟาเรด

ในการสื่อสารระหว่างคอมพิวเตอร์กับ ไมโครคอนโทรลเลอร์แบบอนุกรมนั้น มีข้อดี คือระยะของการสื่อสารมากกว่าแบบขนาน โดยแบบอนุกรมจะส่งข้อมูลทีละบิต แล้วไปเข้า UART จัดการให้เป็นข้อมูลแบบขนานอีกครั้ง โดยต้องมีการกำหนดอัตราการรับส่งข้อมูลให้ตรงกัน ระหว่างตัวไมโครคอนโทรลเลอร์ และ อัตราการรับส่งข้อมูลของพอร์ตอนุกรม ของคอมพิวเตอร์มาตรฐานที่ใช้มีสองแบบ คือ อาร์เอส232(RS232)และ อาร์เอส232(RS485) ในที่นี้จะกล่าวเฉพาะ RS232

6.2.1 มาตรฐานการสื่อสาร แบบ อาร์เอส232

เป็นการสื่อสารแบบอนุกรม ซึ่งระดับโวลเตจแตกต่างจากที่ใช้ในระบบดิจิทอลทั่วไปโดย ระดับสัญญาณของ อาร์เอส232 เป็นแบบไบโพลาร์ (Bipolar) จะแทนระดับลอจิกด้วยไฟฟ้าสองขั้ว ระดับ

โวลเตจทางด้านลบช่วง –3V ถึง 20V แทนค่าลอจิก 1 และโวลเตจทางด้านช่วงบวกช่วง +3 V ถึง +12V แทนค่าลอจิก 0



รูปที่ 6-7 แสดงการเปรียบเทียบระดับโวลเตจของระบบดิจิตอลกับการสื่อสารโดยอาร์เอส232

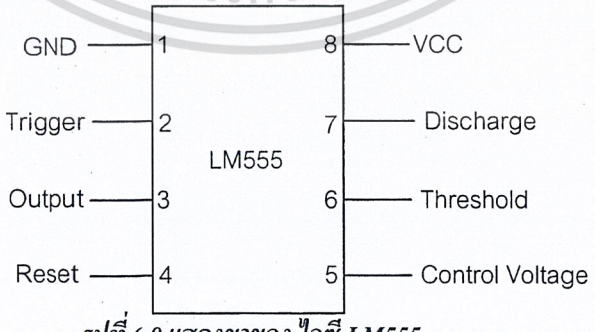
ดังนั้นในการใช้งานถ้าจะส่งข้อมูลที่เป็นดิจิตอลเช่นพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ติดต่อกับคอมพิวเตอร์โดยใช้มาตรฐานอาร์เอส232 ต้องมีอุปกรณ์หรือวงจรพิเศษเพื่อทำการแปลงระดับโวลเตจให้ถูกต้องตามมาตรฐาน

6.2.2 วงจรกำเนิดสัญญาณนาฬิกา

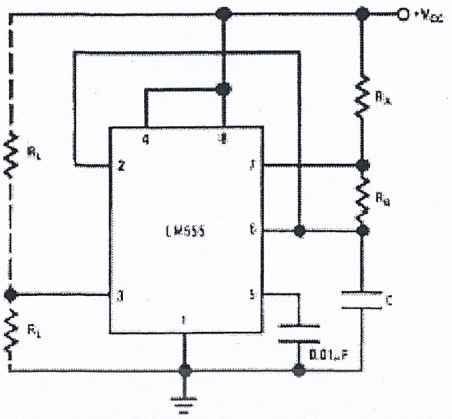
เป็นวงจรที่ให้กำเนิดสัญญาณนาฬิกาในรูปแบบ ของคลื่นต่อเนื่องในระดับโวลเตจทีทีแอล(TTL) หรือสัญญาณต่อเนื่องแบบดิจิตอล

รูปที่ 6-8 แสดงสัญญาณที่ได้จากวงจรกำเนิดสัญญาณนาฬิกา

ในการให้กำเนิดสัญญาณนาฬิกานั้นให้ไอซีเบอร์ 555 โดยมีลักษณะดังรูป



รูปที่ 6-9 แสดงขาของ ไอซี LM555



รูปที่ 6-10 แสดงการเชื่อมต่อวงจรเพื่อสร้างสัญญาณนาฬิกา

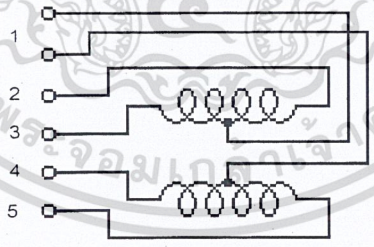
การคำนวณความถี่ต้องใช้ค่าความต้านทาน และค่าตัวเก็บประจุ R_A, R_B และ C ตามลำดับ โดยสูตรเป็นดังนี้

$$F = 1.44/(R_A + 2R_B)C$$

6.3 สเต็ปปิ้งมอเตอร์และการขับสเต็ปปิ้งมอเตอร์

สเต็ปปิ้งมอเตอร์

เป็นมอเตอร์ที่มีการแยกการจ่ายไฟให้ขดลวดเป็นหลาย ๆ ขดเพื่อให้แต่ละการจ่ายไฟเข้าแต่ละขดทำให้เกิดการเหนี่ยวนำทำให้ตัวแม่เหล็กซึ่งเป็นแกน หมุนไปทีละสเตป ใช้ในงานที่ต้องการความละเอียดซึ่งต่างจากดีซีมอเตอร์ หรือมอเตอร์กระแสตรงธรรมดา ที่มีขดลวดเพียงขดเดียว ดังรูป



รูปที่ 6-11 แสดงวงจรภายในสเตปปิ้งมอเตอร์

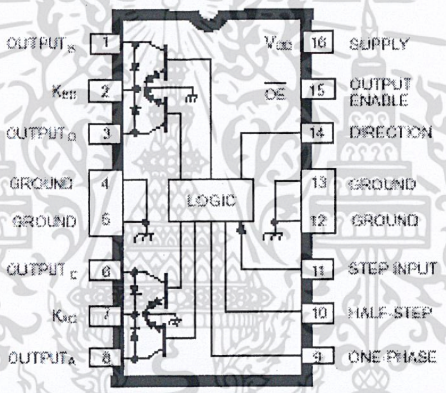
จากรูปจะเห็นว่า การที่จะทำให้มอเตอร์หมุนไปทีละสเตปต้องจ่ายขั้วที่ขั้วเบอร์ 1 และที่เหลือต้องจ่ายไฟขั้งตรงข้ามและต้องจ่ายให้ถูกจังหวะเพื่อการหมุนจะได้เป็นสเตปต่อเนื่องกันไป

จากวงจรภายในสเตปปิ้งมอเตอร์ ในขณะที่มอเตอร์อยู่นิ่งจะกินกระแสมากกว่าในขณะที่หมุน เพราะไฟฟ้าไหลได้ครบวงจร แต่ทอร์กอันเนื่องมากจากการเหนี่ยวนำของขดลวดเมื่อจ่ายกระแสไฟ ของสองขดมีขนาดเท่ากัน แต่กระทำในทิศทางตรงข้าม ส่งผลให้มอเตอร์ร้อนเนื่องจากมีกระแสไหลผ่าน

6.3.1 การขับสเตปปิ้งมอเตอร์

ในการขับสเตปปิ้งมอเตอร์ต้องจ่ายไฟให้เป็นจังหวะต่อเนื่องกันไป โดยการจ่ายไฟไปที่ขดลวดแต่ละขดต้องถูกต้องเพราะถ้าจ่ายผิดลำดับของขดแล้วอาจทำให้ตัวแกนไม่หมุนไปให้เลย หรืออาจทำให้กระแสไหลภายในขดมากเกินไปเป็นเหตุให้มอเตอร์เสียหายได้

การขับโดยมากจะใช้ไอซีเข้ามาช่วยเพราะไม่ต้องทำการกำหนดจังหวะการจ่ายไฟเอง และทำให้ง่ายต่อการทำงาน โดยใช้ไอซีเบอร์ 5804 เป็นไอซีขับสเตปปิ้งมอเตอร์ที่สะดวกเพราะการควบคุมใช้ระดับโวลเตจของดิจิตอล ไม่ต้องทำการกำหนดจังหวะเอง เพียงจ่ายสัญญาณนาฬิกา ซึ่งเป็นความถี่ของการหมุน ความเร็วของการหมุนจึงขึ้นอยู่กับความถี่นี้ ซึ่งถ้าจ่ายด้วยความถี่สูงเกินไป ตัวแกนจะไม่สามารถหมุนได้เพราะไม่สามารถเอาชนะแรงเฉื่อยของแกนมอเตอร์ได้ และถ้าจ่ายด้วยความถี่น้อยเกินไปจะทำให้ตัวสเตปปิ้งมอเตอร์ร้อนเพราะจ่ายไฟตรงให้ขดลวดแต่ละขดนานเกินไปทำให้กระแสวิ่งผ่านมากอาจทำให้สเตปปิ้งมอเตอร์เสียหายได้ การจ่ายด้วยความถี่มาก จะได้ความเร็วมากกว่า จ่ายด้วยความถี่น้อยกว่าแต่จะได้ทอร์กสูงกว่า



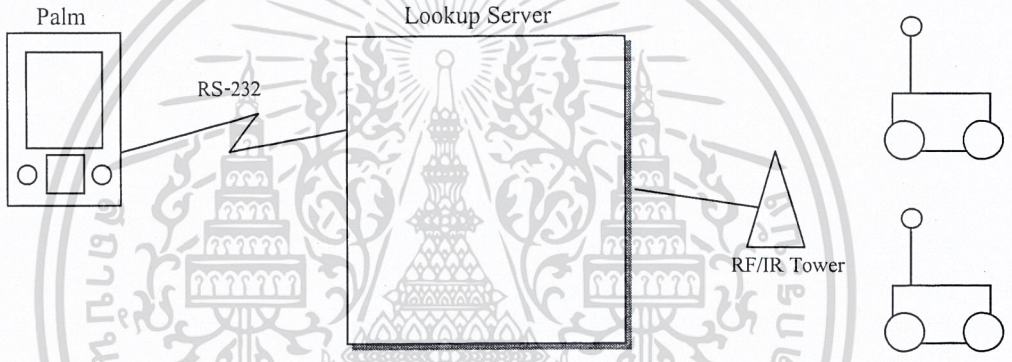
รูปที่ 6-12 แสดงรายละเอียดของไอซี 5804

จากรูปสัญญาณควบคุมต่างใช้ระดับโวลเตจ ทีทีแอลส่วนไฟที่จ่ายให้สเตปปิ้งมอเตอร์จะออกมาเป็นจังหวะตามที่กำหนดไว้ที่ขาโดยการขับแบ่งเป็น 3 โหมดคือ

- เวฟไดรฟว์ ไฟที่ออกจากเอาท์พุทเป็นจังหวะ A-B-C-D ซึ่งเป็นโหมดที่ประหยัดพลังงานและได้ความแม่นยำในเรื่องของทางหรือการวัดค่า
- สองเฟส ไฟที่ออกจากเอาท์พุทเป็นจังหวะ AB-BC-CD-DA โหมดนี้จะได้ความเร็วกับทอร์กที่ดี
- ครึ่งสเตป ไฟที่ออกจากเอาท์พุท จะเป็นจังหวะ A-AB-B-BC-C-CD-D-DA ซึ่งมีแปดช่วง สเตป

บทที่ 7
การออกแบบระบบจินี่

จากแนวความคิดของจินี่ โครงงานนี้ได้นำแนวคิดและทฤษฎีที่กล่าวมาจากบทข้างต้นมาประยุกต์ใช้ในการสร้างระบบตัวอย่าง ซึ่งระบบนี้จากที่ได้กล่าวไว้ในบทนำ เป็นระบบเครือข่ายของหุ่นยนต์ที่ทำหน้าที่เป็นเซอร์วิสของจินี่ หุ่นยนต์สามารถเชื่อมต่อเข้ากับระบบหรือถอดออกจากระบบได้ตลอดเวลา และสามารถมีหุ่นยนต์ในระบบเพิ่มกี่ตัวก็ได้โดยที่ไม่จำเป็นต้องปรับแต่งระบบ



รูปที่ 7-1 ภาพรวมของการออกแบบระบบ

การควบคุมหุ่นยนต์จะควบคุมโดยเครื่องคอมพิวเตอร์มือถือ ปาล์มไพลอต ซึ่งจะทำหน้าที่เป็นผู้ใช้งานเซอร์วิสของเครือข่ายจินี่ ซึ่งความสามารถของหุ่นยนต์ที่ถูกควบคุมได้คือ การเดินหน้า ถอยหลัง เลี้ยวซ้าย เลี้ยวขวา และสั่งหยุดการเคลื่อนไหว โดยที่ตัวหุ่นจะมีเซ็นเซอร์ สองชนิด ชนิดละ 1 ตัวติดอยู่ คือ อินฟราเรดเซ็นเซอร์ สำหรับเตือนว่ามีวัตถุขวางอยู่ด้านหน้า และ คอนแท็คสวิตช์ สำหรับแจ้งเตือนการชน หุ่นยนต์จะติดต่อกับเซิร์ฟเวอร์ผ่านทางคลื่นอินฟราเรด ซึ่งส่วนที่ใช้ส่งและรับข้อมูลมีอินเตอร์เฟสแบบ RS-232 ทำให้สามารถใช้สื่อสัญญาณอื่นที่มีอินเตอร์เฟสแบบ RS-232 แทนอินฟราเรดก็ได้ เช่นสายเคเบิล หรือ คลื่นวิทยุ

เนื่องจากในระบบของจินี่นั้น จะมองบริการทุกอย่างเป็นเซอร์วิสเพราะฉะนั้นในการออกแบบจึงต้องมองภาพของหุ่นยนต์ขนาดเล็กให้อยู่ในรูปของเซอร์วิส ซึ่งจากการออกแบบ ระบบตัวอย่างนี้จะมองกลุ่มของหุ่นยนต์ขนาดเล็กเป็น 1 เซอร์วิส และรวมเรียกเซอร์วิสนี้ว่า RobotRoomService (ห้องของหุ่นยนต์) โดยแค่ละกลุ่มก็จะมีโปรแกรมที่ทำการสร้างเซอร์วิสออบเจ็กต์ และนำไปลงทะเบียนกับ ลุคอัพเซอร์วิส ของจินี่ให้

การออกแบบระบบนี้นั้นสามารถแบ่งการออกแบบ ออกเป็นส่วนต่างๆได้หลักๆ 3 ส่วนคือ ส่วนของปาล์ม ส่วนของเซอร์วิส (Robot Room Service) และส่วนของฮาร์ดแวร์และซอฟต์แวร์ของหุ่นยนต์

โดยในบทนี้จะกล่าวถึงส่วนของปาล์มและส่วนของเซอร์วิส ในส่วนของฮาร์ดแวร์และซอฟต์แวร์ของหุ่นยนต์นั้นจะกล่าวถึงในบทถัดไป

7.1 การออกแบบส่วนของเซอร์วิส

เซอร์วิสที่ใช้ในระบบตัวอย่างนี้นั้น แบ่งออกเป็นสองส่วนคือ เซอร์วิสที่เป็นฮาร์ดแวร์ และเซอร์วิสที่เป็นซอฟต์แวร์ ซึ่งมีลักษณะที่แตกต่างกันคือ

- ซอฟต์แวร์เซอร์วิส สร้างเป็นโปรแกรมกราฟฟิกซึ่งแสดงห้องของหุ่นยนต์ และมีหุ่นยนต์อยู่ภายในห้อง โดยที่ผู้ใช้สามารถควบคุมหุ่นที่อยู่ในห้องนั้นให้เคลื่อนไหวได้
- ฮาร์ดแวร์เซอร์วิส สร้างเป็นหุ่นยนต์ขนาดเล็กควบคุมด้วยไมโครคอนโทรลเลอร์ ผู้ใช้สามารถควบคุมหุ่นให้เคลื่อนที่ได้ และสามารถรับรู้ถึงสถานการณ์ต่างๆที่เกิดขึ้นจากหุ่นยนต์ จากเซ็นเซอร์ที่ติดอยู่กับตัวหุ่น

โดยเซอร์วิสทั้งสองรูปแบบนี้ ถึงแม้ว่าจะมีลักษณะที่แตกต่างกัน แต่ว่าถูกสร้างขึ้นมาด้วยอินเตอร์เฟสเดียวกัน เพราะฉะนั้น การควบคุมจากฝั่งของปาล์มก็จะสามารถทำได้โดยผู้ใช้ไม่รู้สึกถึงความแตกต่างและไม่จำเป็นต้องปรับแต่งในฝั่งปาล์มแต่อย่างใด

7.1.1 การออกแบบเซอร์วิสของจินี่

กระบวนการในการจะออกแบบและสร้างเซอร์วิสของจินี่ ไม่ว่าจะเป็นฮาร์ดแวร์เซอร์วิส หรือซอฟต์แวร์เซอร์วิส จะมีกระบวนการสร้าง และโครงสร้างหลักของโปรแกรมที่คล้ายกัน จะแตกต่างกันที่รายละเอียด โครงสร้างหลักของเซอร์วิสสามารถแบ่งได้เป็น 2 ส่วนหลักๆตามหลักการของการสร้างเซอร์วิสสำหรับเครือข่ายจินี่ คือ *เซอร์วิสออบเจ็กต์* และ*ตัวเซอร์วิส*

การออกแบบโครงสร้างการทำงานของเซอร์วิสและเซอร์วิสออบเจ็กต์ ผู้พัฒนาระบบนั้นสามารถออกแบบได้หลายแบบ แต่การออกแบบที่ใช้ในโครงงานนี้นั้น ตัวเซอร์วิสจะทำการ สร้างเซอร์วิสออบเจ็กต์ขึ้นมา แล้วทำการลงทะเบียนเซอร์วิสออบเจ็กต์นั้นกับ ลุคอัพเซอร์วิส ซึ่งเมื่อผู้ใช้มาค้นหาเซอร์วิสก็จะได้รับเซอร์วิสออบเจ็กต์เพื่อไปทำหน้าที่เป็นพรอกซี่กลับมายังตัวเซอร์วิส

การสร้าง RobotRoom Service

การสร้างตัวเซอร์วิส ในโครงงานนี้ตัวเซอร์วิสคือกลุ่มของหุ่นยนต์ซึ่งเราเรียกว่า Room เพราะฉะนั้นจึงสร้างออบเจ็กต์ขึ้นมาเพื่อทำหน้าที่เป็นห้องของหุ่นยนต์ โดยกำหนดอินเตอร์เฟสให้กับออบเจ็กต์ที่จะมาสร้างเป็น RoomService เพื่อในกรณีที่ต้องการพัฒนาเซอร์วิสในรูปแบบต่างๆหลายรูปแบบจะได้สามารถนำเซอร์วิสแต่ละอันไปใช้ร่วมกันได้โดยผ่านอินเตอร์เฟสเดียวกัน

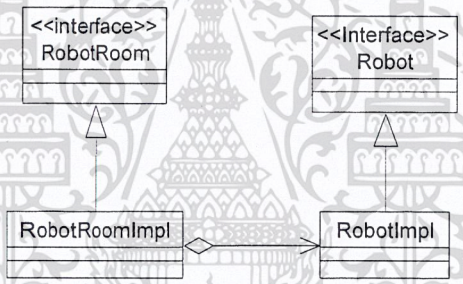
```
public interface RobotRoom {
        public int[] getAllRobotID();
        public Robot getRobot(int ID);
        public void setSender(ServerDelivery s);
        public void sendEvent(int type,int fromRobot);
        public void display(String s);
```

```
}
```

ออบเจ็กต์ที่ถูกสร้างเป็นเซอร์วิสนี้นั้นจะทำหน้าที่เป็นผู้ดูแลหุ่นยนต์ที่อยู่ภายในห้อง โดย เมธอดต่างๆจะถูกเรียกใช้จากผู้ใช้ ผ่านทางเซอร์วิสออบเจ็กต์ ออบเจ็กต์ของหุ่นยนต์นั้นก็เช่นกัน จะถูกสร้างตามอินเตอร์เฟสที่ชื่อ Robot

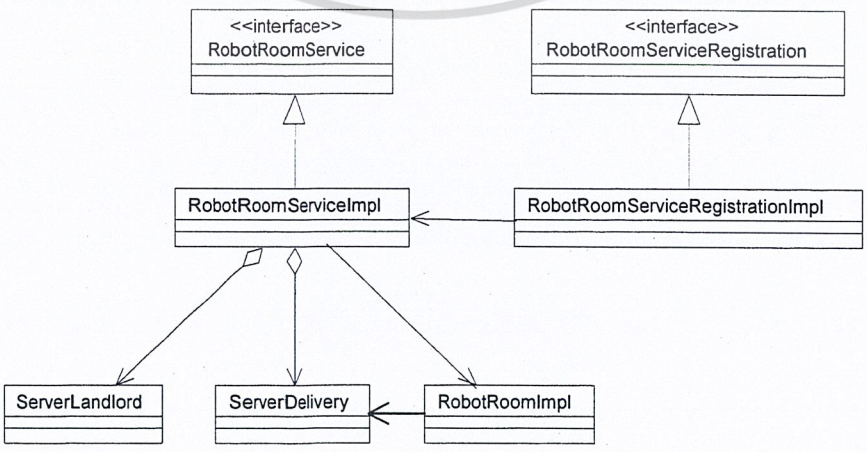
```
public interface Robot{
        public int getID();
        public void forward();
        public void backward();
        public void turnLeft();
        public void turnRight();
        public void turnOn();
        public void turnOff();
        public void stop_move();
}
```

ส่วนการสร้างรายละเอียดของหุ่นยนต์และตัวเซอร์วิสจะสร้างอย่างไรนั้น ก็ขึ้นอยู่กับว่าต้องการให้เซอร์วิสออกมาในลักษณะไหน ซึ่งสามารถดูตัวอย่างการสร้างได้ในหัวข้อต่อไป



รูปที่ 7-2 คลาสไดอะแกรม อินเตอร์เฟสหลักของระบบ

เซอร์วิสออบเจ็กต์ สามารถกล่าวได้ว่าเป็นส่วนที่สำคัญที่สุดของเซอร์วิสก็ได้ เนื่องจากการติดต่อกับผู้ใช้เซอร์วิสนั้นจะต้องทำผ่านเซอร์วิสออบเจ็กต์ เพราะฉะนั้นเซอร์วิสออบเจ็กต์ที่ออกแบบ ควรจะออกแบบให้ใช้ง่ายสำหรับผู้ใช้



รูปที่ 7-3 คลาสไดอะแกรมของเซอร์วิสออบเจ็กต์

จากแผนภาพ UML คลาสที่ทำตัวเป็นเซอร์วิสออบเจ็กต์ก็คือ RobotRoomServiceImpl ซึ่งสร้างมาจากอินเตอร์เฟส RobotRoomService ซึ่งอินเตอร์เฟสนี้มีลักษณะดังนี้

```
import java.rmi.*;
import net.jini.core.event.*;

public interface RobotRoomService {
    public RobotRoomServiceRegistration getInstance(long duration)
                                        throws RemoteException;
    public EventRegistration trackEvents(long duration,
                                         RemoteEventListener rel,
                                         MarshalledObject key)
                                         throws RemoteException;
}
```

เมทตอด getInstance() ใช้สำหรับผู้ใช้ทำการลงทะเบียนเพื่อจะขอใช้เซอร์วิส โดยถ้าหากลงทะเบียนสำเร็จ จะคืนค่าออกมาเป็นออบเจ็กต์ชนิด RobotRoomServiceRegistration ซึ่งการที่ผู้ใช้เซอร์วิสจะเรียกใช้เซอร์วิส ก็จะเรียกผ่านออบเจ็กต์นี้

เมทตอด trackEvents() ใช้สำหรับผู้ใช้ทำการลงทะเบียนเพื่อขอรับฟัง Event ที่เกิดขึ้นในระบบ โดยผู้ใช้จะส่งออบเจ็กต์ที่ทำหน้าที่เป็นตัวดักฟัง Event มาเป็นพารามิเตอร์

สำหรับโครงสร้างของ RobotRoomServiceRegistration คือ

```
import java.rmi.*;
import net.jini.core.lease.*;

public interface RobotRoomServiceRegistration{
       //Get all RobotID on the room for request control
       public int[] getRobotID() throws RemoteException;

       //Get Robot controller for the specific RobotID
       public RobotController getController(int ID) throws RemoteException;

       //Get the Lease for renewal perpose
       public Lease getLease()  throws RemoteException;
}
```

เมทตอดที่มีอยู่ 3 อันนั้นมีไว้สำหรับผู้ใช้เพื่อใช้งานเซอร์วิส เมทตอด getRobotID() ไว้สำหรับเรียกดูว่ามีหุ่นยนต์อยู่กี่ตัวในเซอร์วิส และมี ID อะไรบ้าง เมทตอด getController ทำหน้าที่ร้องขอออบเจ็กต์ที่ทำหน้าที่ควบคุมหุ่นเมื่อผู้ใช้ต้องการควบคุมหุ่น และเมทตอด getLease() จะคืนค่าเป็น Lease ออบเจ็กต์สำหรับผู้ใช้เพื่อดูแลต่อไป

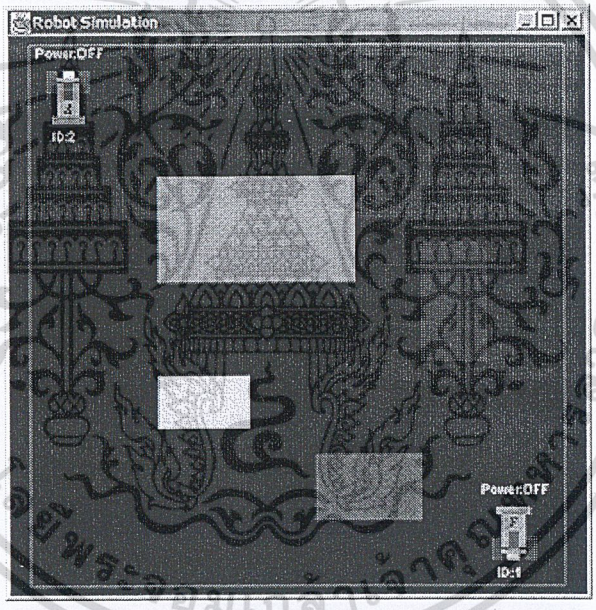
ในเซอร์วิสออบเจ็กต์ (RobotRoomServiceImpl) นั้นจะสร้างออบเจ็กต์ที่ช่วยในการจัดการขึ้นมาอีก 2 ตัวคือ ServerLandlord และ ServerDelivery โดย ServerLandlordสร้างจากอินเตอร์เฟส Landlord จากแพคเกจ com.sun.jini.lease.landlord ของชุดพัฒนาจินี่ ซึ่ง ServerLandlord ทำหน้าที่เป็นตัวสร้างและดูแล Lease ออบเจ็กต์ของผู้ใช้ คอยดูแลว่า Lease ใดไม่ได้ถูกผู้ใช้ Renew ตามเวลาที่กำหนด ก็จะทำให้ Lease นั้นหมดอายุและถือว่าผู้ใช้ที่เป็นเจ้าของ Lease นั้นได้ออกจากระบบไปแล้ว

เซอร์วิสออบเจ็กต์ที่สร้างขึ้นมานี้สามารถนำไปใช้ได้กับทั้งฮาร์ดแวร์เซอร์วิสและซอฟต์แวร์เซอร์วิส เนื่องจากเซอร์วิสออบเจ็กต์นี้นั้นติดต่อกลับมายังตัวเซอร์วิสผ่านอินเตอร์เฟส RobotRoom และตัวเซอร์วิสทั้งฮาร์ดแวร์เซอร์วิสและซอฟต์แวร์เซอร์วิส ต่างก็ต้องสร้างมาจากอินเตอร์เฟสนี้ด้วย จึงทำให้สามารถใช้เซอร์วิสออบเจ็กต์นี้ร่วมกันได้

ServerDelivery ทำหน้าที่ส่ง Event ไปยังผู้ใช้ที่มาลงทะเบียนรับฟัง Event ไว้โดยออบเจ็กต์นี้จะถูกเรียกใช้จาก RobotRoom เพื่อส่ง Event ที่เกิดจากหุ่นยนต์

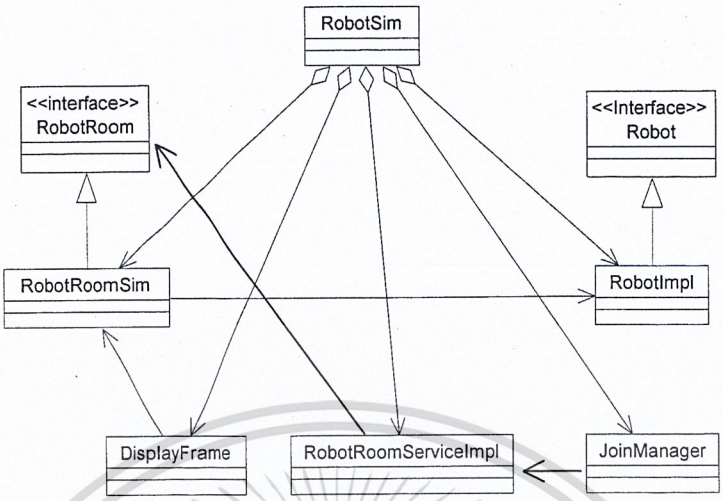
7.1.2 การออกแบบซอฟต์แวร์เซอร์วิส (Robot Simulator)

การออกแบบโปรแกรมโรบอตซิมมูเลเตอร์ หรือเรียกย่อๆว่า RobotSim นั้นจะออกแบบโปรแกรมในลักษณะโปรแกรมกราฟฟิกที่แสดงห้องของหุ่นยนต์ โดยห้องนั้นมีกำแพงล้อมรอบ และมีวัตถุเสมือนวางอยู่ภายในห้อง โปรแกรมนี้จะลงทะเบียนตัวเองเป็นซอฟต์แวร์เซอร์วิสของจินี่



รูปที่ 7-4 ตัวอย่างหน้าจอของโปรแกรมโรบอตซิมมูเลเตอร์

จากที่ได้กล่าวมาข้างต้น การสร้างเซอร์วิสที่แตกต่างกัน สามารถทำได้โดยการสร้างจากอินเตอร์เฟสเดียวกัน คือ RobotRoom และ Robot ซึ่งผู้พัฒนาสามารถเพิ่มส่วนที่เป็นรายละเอียดเข้าไปได้ สำหรับโรบอตซิมมูเลเตอร์ มีการออกแบบคร่าวๆดังนี้



รูปที่ 7-5 คลาสไดอะแกรมของโปรแกรมโรบอตซิมมูเลเตอร์

คลาส RobotSim ซึ่งเป็นคลาสหลักของเซอร์วิสนี้ ทำการสร้างออบเจ็กต์ RobotRoomSim (สร้างจากอินเตอร์เฟส RobotRoom) เพื่อเป็นตัวเซอร์วิส และสร้าง RobotImpl (สร้างจากอินเตอร์เฟส Robot) ซึ่งแทนตัวหุ่นยนต์จำลองในระบบนี้ โดยการแสดงผลห้องของหุ่นยนต์ที่เป็นกราฟฟิกนั้นจะแสดงผลโดยออบเจ็กต์ของคลาส DisplayFrame ซึ่ง Inherit มาจากคลาส Frame DisplayFrame นั้นจะทำการอ่านค่าตำแหน่งของหุ่นยนต์จำลองที่อยู่ภายในห้อง ตำแหน่งของออบเจ็กต์ต่างๆภายในห้อง แล้วทำการนำมาแสดงผลในรูปแบบของกราฟฟิก

ออบเจ็กต์ RobotRoomServiceImpl สร้างขึ้นมาโดยกำหนดให้เป็น ออบเจ็กต์ระยะไกล เพื่อทำหน้าที่เป็นเซอร์วิสออบเจ็กต์ RobotSim จะทำการลงทะเบียนเซอร์วิสออบเจ็กต์นี้กับ ลุคอัพเซอร์วิส โดยออบเจ็กต์ของคลาส JoinManager ซึ่งเป็นส่วนหนึ่งของชุดพัฒนาจินี่

7.1.3 การออกแบบฮาร์ดแวร์เซอร์วิส

ฮาร์ดแวร์เซอวิสในโครงงานนี้นั้นจะสร้างเป็นตัวหุ่นยนต์ขนาดเล็กซึ่งควบคุมด้วยไมโครคอนโทรลเลอร์ตระกูล 8051 การเขียนโปรแกรมควบคุมหุ่นยนต์นั้นใช้ภาษาแอสเซมบลีสำหรับ 8051 ในการเขียน ทำให้หุ่นยนต์ที่สร้างขึ้นยังไม่มีความสามารถที่จะติดต่อกับเครือข่ายของจินี่ได้โดยตรง จึงจำเป็นต้องมีการพัฒนาโปรแกรมเพิ่มเติม เพื่อเป็นตัวกลางในการติดต่อระหว่างหุ่นยนต์และเครือข่ายของจินี่ โดยให้ชื่อโปรแกรมว่า โรบอตเดมอน (Robot Daemond)

เนื่องจากตามหลักการและแนวคิดของจินี่ ที่ทำให้อุปกรณ์ต่างๆต่อเชื่อมเข้ากับเครือข่ายได้อย่างอัตโนมัตินั้น การพัฒนาโปรแกรมโรบอตเดมอนนั้นจึงสร้างให้มีความอัตโนมัติมากที่สุด โดยถึงแม้ว่าจะไม่ได้ใช้ความสามารถของจินี่ แต่ก็ใช้หลักการเดียวกันในการสร้าง

การออกแบบฮาร์ดแวร์เซอร์วิสนี้นั้นจะแบ่งการออกแบบเป็นสองส่วน คือส่วนของโปรแกรมโรบอตเดมอน และส่วนของตัวฮาร์ดแวร์ของหุ่นยนต์ ซึ่งส่วนของโปรแกรมโรบอตเดมอนนี้นั้นจะกล่าวถึงในบทนี้ ส่วนตัวฮาร์ดแวร์ของหุ่นยนต์นั้น จะกล่าวถึงในบทถัดไป

การติดต่อกันระหว่างฮาร์ดแวร์ของหุ่นยนต์กับโปรแกรมโรบอตเดมอนนั้น จะสร้างโพรโตคอลในการคุยกัน ดังนี้

- การส่งข้อมูลในแต่ละครั้งระหว่างโรบอตเดมอนไปยังหุ่นยนต์ และระหว่างหุ่นยนต์มายังโรบอตเดมอนจะส่งข้อมูลทีละ 1 ไบต์ โดยลักษณะของ แมสเซจและการแบ่งเป็น message ประเภทต่างๆ ดังนี้

Message จากหุ่นยนต์ → โรบอตเดมอน

PING	มีโครงสร้างคือ	1	1	0	0	0	x	x	x
ALERT_HIT	มีโครงสร้างคือ	1	1	1	1	1	x	x	x
HIT_UNKNOWN	มีโครงสร้างคือ	1	1	1	1	0	x	x	x

Message จากโรบอตเดมอน → หุ่นยนต์

ROBOT_FORWARD	มีโครงสร้างคือ	0	1	0	0	1	x	x	x
ROBOT_BACKWARD	มีโครงสร้างคือ	0	1	0	1	0	x	x	x
ROBOT_LEFT	มีโครงสร้างคือ	0	1	0	1	1	x	x	x
ROBOT_RIGHT	มีโครงสร้างคือ	0	1	1	0	0	x	x	x
ROBOT_STOP	มีโครงสร้างคือ	0	1	1	0	1	x	x	x

ใน 3 บิตหลังจะแทนค่าด้วยหมายเลข ID ของหุ่นยนต์ที่เกี่ยวข้อง เป็นเลขฐานสอง ซึ่งแสดงว่าระบบนี้สามารถมีหุ่นยนต์ได้พร้อมกันทีละ 16 ตัวโดยมี ID คือ 0-15 หากต้องการพัฒนาให้รองรับหุ่นยนต์ให้ได้มากกว่า 16 ตัวจะต้องมีการแก้ไขในส่วนโพรโตคอลการสื่อสารใหม่

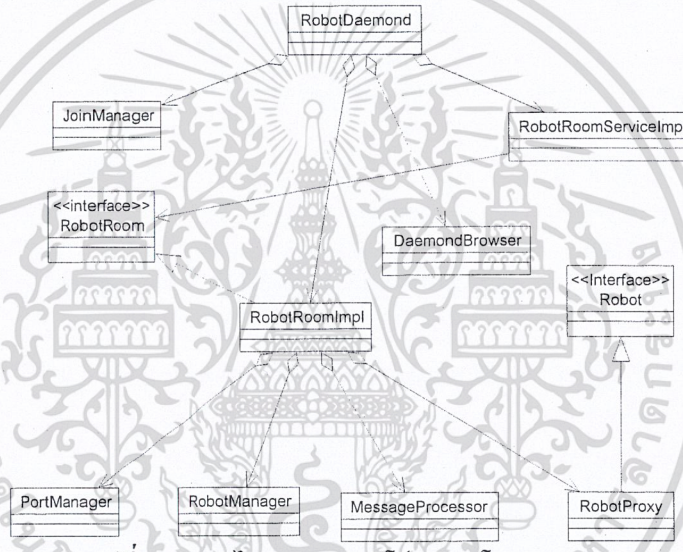
- หุ่นยนต์จะใช้ message PING ในการบอกกับระบบว่าตอนนี้ได้เชื่อมต่อกับระบบแล้ว
- หุ่นยนต์จะต้องส่ง message PING มายังโรบอตเดมอนภายในเวลาที่กำหนด มิเช่นนั้นโรบอตเดมอนจะถือว่าหุ่นยนต์ตัวนั้นออกจากระบบไปแล้ว (ไทม์เอ้าท์)
- เมื่อเกิดเหตุการณ์อะไรขึ้นกับหุ่นยนต์ หุ่นยนต์จะแจ้งให้กับระบบทราบโดยใช้ message ALERT_HIT และ HIT_UNKNOWN
- โรบอตเดมอนจะส่งสัญญาณควบคุมเพื่อควบคุมไปยังหุ่นด้วยสัญญาณ ROBOT_FORWARD ROBOT_BACKWARD,ROBOT_LEFT,ROBOT_RIGHT,ROBOT_LEFT และ ROBOT_STOP และใส่หมายเลข ID ของหุ่นที่ต้องการควบคุมไปยัง 3 บิตหลัง
- เมื่อโรบอตเดมอนได้รับสัญญาณปิงในครั้งแรกจากหุ่นยนต์ตัวใด จะถือว่าหุ่นยนต์ตัวนั้นได้เข้าสู่ระบบ หุ่นยนต์ตัวนั้นจะต้องส่งสัญญาณ PING มาเรื่อยๆเพื่อยืนยันการคงอยู่ของตนเอง ถ้าหากหุ่นยนต์ไม่ได้ส่งสัญญาณ PING กลับมายังระบบภายในเวลาที่กำหนดไว้ จะถือว่าหุ่นยนต์ตัวนั้นได้ออกจากระบบแล้ว ซึ่งแนวคิดนี้ได้นำมาจากกระบวนการ Leasing ของจินี่ การ PING ก็เปรียบเสมือนการ renew Lease นั่นเอง

7.1.4 การออกแบบโปรแกรมโรบอตเดมอน

โปรแกรมนี้จะสร้างเป็นโปรแกรมที่ใช้ควบคุมหุ่นยนต์ที่เป็นฮาร์ดแวร์โดยการส่งสัญญาณการควบคุมผ่านไปทางพอร์ต COM ซึ่งการติดต่อไปยังพอร์ตการสื่อสารนั้น จะใช้ชุด API พิเศษของภาษาจาวาชื่อว่า Java Communication API ซึ่งสามารถใช้ในการติดต่อกับพอร์ตการสื่อสารของคอมพิวเตอร์ได้

โรบอตเดมอนสามารถติดต่อกับหุ่นยนต์ได้ทีละหลายตัวผ่านทางพอร์ตอนุกรมเพียงพอร์ตเดียว ด้วยการเปลี่ยนสื่อสำหรับการส่งข้อมูลออกจากพอร์ตอนุกรมไปยังหุ่นยนต์ ซึ่งในโครงงานนี้ได้ใช้การสื่อสารด้วยอินฟราเรด มาเป็นตัวกลางในการส่งข้อมูลระหว่างคอมพิวเตอร์และหุ่นยนต์ ทำให้สามารถกระจายสัญญาณจากตัวโรบอตเดมอนไปยังหุ่นยนต์ได้ทีละหลายตัว

การออกแบบโปรแกรมสามารถแสดงได้ด้วยแผนภาพ UML อย่างคร่าวๆได้ดังนี้



รูปที่ 7-6 คลาสไดอะแกรมของโปรแกรมโรบอตเดมอน

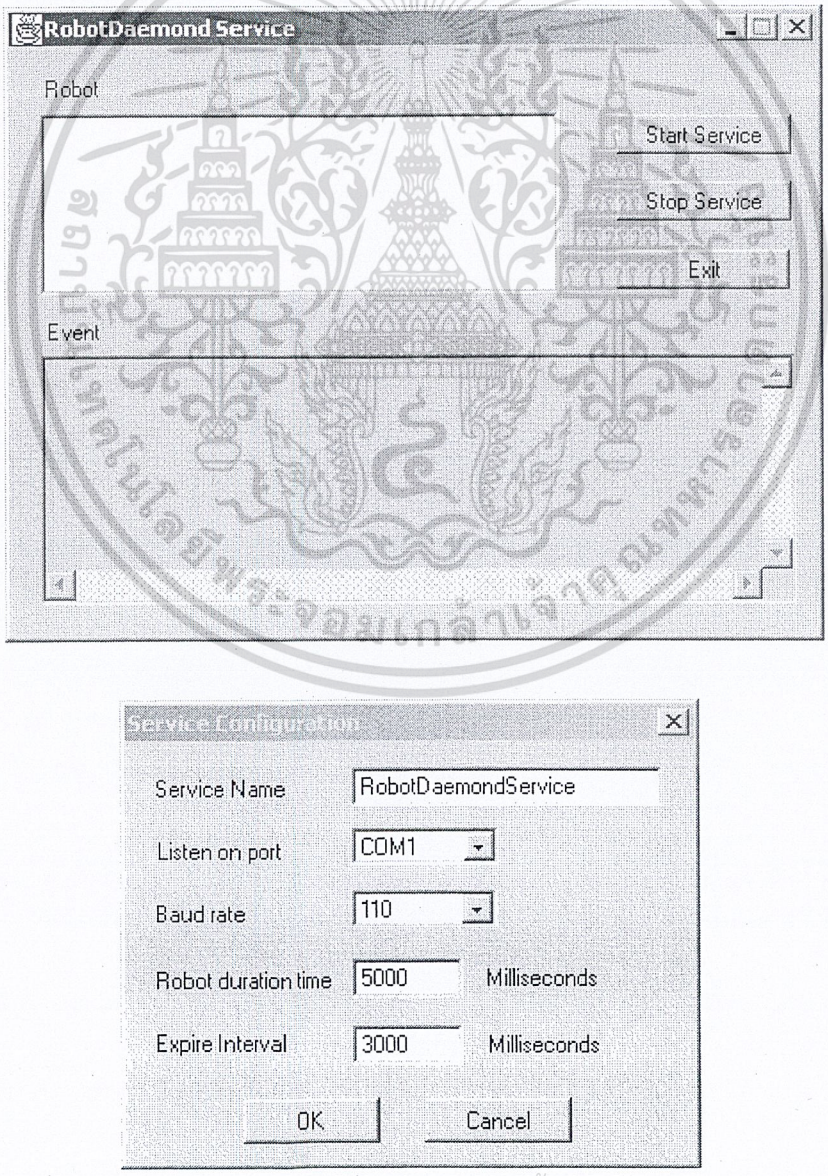
ตัวเซอร์วิสนั้นคือออบเจ็กต์ที่ทำหน้าที่เป็นห้องเสมือนของหุ่นยนต์ คือ RobotRoomImpl ซึ่งสร้างมาจากอินเตอร์เฟส RobotRoom ส่วนเซอร์วิสออบเจ็กต์นั้นสามารถใช้ออบเจ็กต์ชนิดเดียวกับที่เป็นซอฟต์แวร์เซอร์วิส คือ RobotRoomServiceImpl ได้เลยเนื่องจากว่าเซอร์วิสออบเจ็กต์นี้นั้นติดต่อกลับมายังตัวเซอร์วิสผ่านอินเตอร์เฟส RobotRoom และตัวเซอร์วิสสำหรับฮาร์ดแวร์เซอร์วิสนี้นั้นก็สร้างมาจากอินเตอร์เฟสนี้ด้วย จึงทำให้สามารถใช้ร่วมกันได้

RobotRoomImpl ซึ่งเป็นตัวเซอร์วิสนั้นจะสร้างออบเจ็กต์ที่มาช่วยในการจัดการหุ่นอีก 3 ตัวคือ PortManager ทำหน้าที่ดูแลรับและส่งข้อมูลจากพอร์ตสื่อสาร MessageProcessor ทำหน้าที่ประมวลผลข้อมูลที่รับมาจาก PortManager และสุดท้าย RobotManager ทำหน้าที่เป็นตัวจัดการกับหุ่นยนต์แต่ละตัว โดยจะทำงานกับหุ่นยนต์เสมือน ที่สร้างขึ้นจากออบเจ็กต์ RobotProxy RobotProxy นั้นถูกสร้างขึ้นมาตามอินเตอร์เฟส Robot ทำให้สามารถทำงานร่วมกับเซอร์วิสออบเจ็กต์ได้เป็นอย่างดี

เมื่อมีสัญญาณ PING เข้ามายังพอร์ต MessageProcessor จะแจ้งให้กับ RobotManager ทราบว่ามีสัญญาณ PING มาจากหุ่นยนต์ ID ใด เมื่อ RobotManager ทราบก็จะไปตรวจสอบดูว่ามี RobotProxy ถูกสร้างขึ้นมาสำหรับหุ่นยนต์ ID นั้นหรือยัง ถ้ายัง ก็จะสร้าง RobotProxy สำหรับหุ่นยนต์ ID นั้นขึ้นเพื่อแสดงว่าหุ่นยนต์ที่มี ID ที่ส่งมานั้นได้เชื่อมต่อกับระบบแล้ว ถ้าหากว่ามี RobotProxy สำหรับหุ่นยนต์ ID นั้นอยู่แล้ว RobotManager ก็จะถือว่าเป็นสัญญาณการ renew ของ RobotProxy

และ RobotManager ยังสร้างเธรดขึ้นมาอีกเธรดหนึ่งเพื่อเป็นตัวตรวจสอบว่า RobotProxy ตัวใดหมดอายุแล้ว ถ้าหากตัวใดหมดอายุ ก็จะถอดหุ่นยนต์ตัวนั้นออกจากระบบ

สำหรับโปรแกรมนี้ได้มีการสร้าง GUI ขึ้นมาด้วยเพื่อให้ผู้ใช้งานระบบได้ใช้ในการปรับแต่งค่าต่างๆของเซอร์วิส เนื่องจากการสื่อสารผ่านพอร์ตอนุกรมจะต้องมีการกำหนดค่าพารามิเตอร์ต่างๆเพื่อจะเปิดพอร์ตในการติดต่อสื่อสาร



รูปที่ 7-7 ตัวอย่างหน้าจอของโปรแกรมโรยอตเดมอน

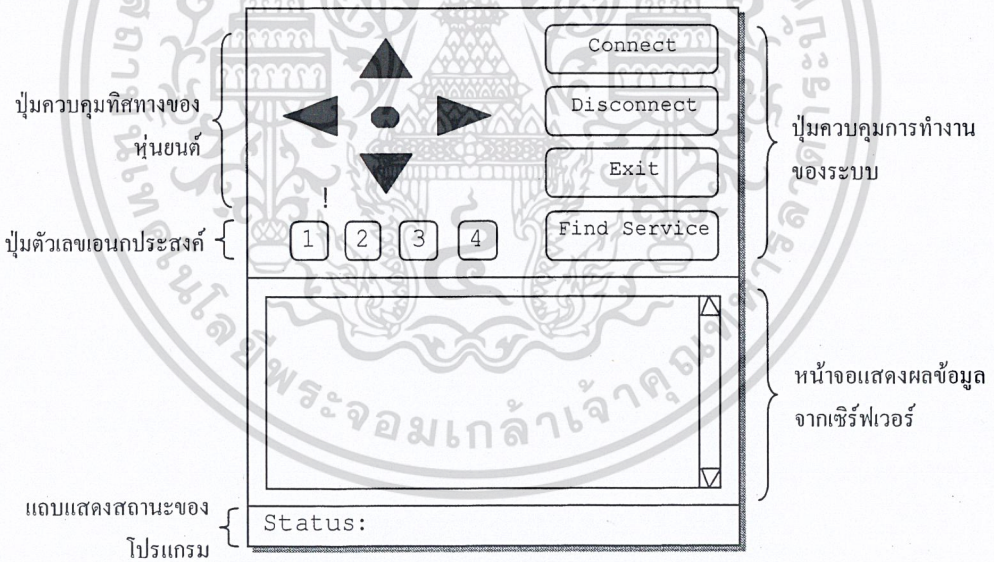
ค่าที่สามารถปรับแต่งได้สำหรับเซอร์วิส คือ

- ชื่อของเซอร์วิสที่ลงทะเบียนกับจินี่
- ต้องการรับข้อมูลที่พอร์ตไหน
- มี baud rate เท่าใด
- ระยะเวลาช่วงอายุของ RobotProxy ถ้าไม่ได้รับการ รีนิว
- ช่วงเวลาการตรวจสอบหา RobotProxy ของหุ่นยนต์ที่หมดอายุ

7.2 การออกแบบในส่วนของผู้ใช้

7.2.1 การออกแบบโปรแกรมบนปาล์ม

ผู้ใช้ของเซอร์วิสนั้นจะควบคุมหุ่นยนต์โดยผ่านเครื่องปาล์ม ซึ่งเครื่องปาล์มนั้นจะมีโปรแกรมที่ทำหน้าที่เป็นเหมือนคันบังคับของหุ่นอยู่ โปรแกรมนี้จะถูกเขียนขึ้นด้วยภาษาจาวาและทำงานอยู่บนเวอร์ชวลมาชีน สำหรับเครื่องปาล์มโดยเฉพาะที่ชื่อว่า KVM ย่อมาจาก Kilobyte Virtual Machine ซึ่งเป็นส่วนหนึ่งของ จาวาไมโครอดิชันเวอร์ชัน 2 ซึ่งรายละเอียดเกี่ยวกับ KVM รวมถึงวิธีการเขียนโปรแกรมสำหรับปาล์มนี้สามารถดูได้ในภาคผนวก



รูปที่ 7-8 ตัวอย่างหน้าจอของโปรแกรมโรยอตไคลเอ็นต์

ด้านบนนี้คือภาพหน้าจอของโปรแกรมควบคุมหุ่นยนต์สำหรับปาล์ม ซึ่งประกอบด้วยส่วนต่างๆ ที่ใช้ในการควบคุมหุ่นยนต์ดังนี้

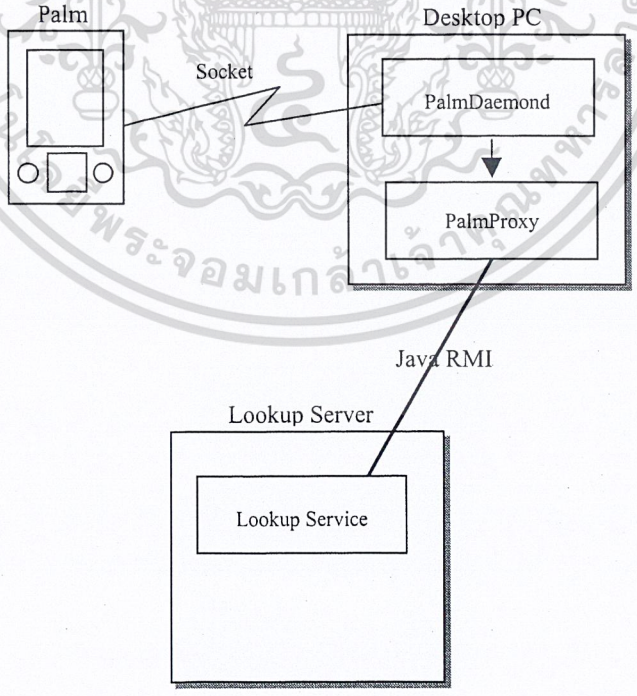
- ปุ่มควบคุมการทำงานของระบบ มีปุ่ม Connect ใช้ในการเชื่อมต่อเข้ากับ PalmDaemond ปุ่ม Disconnect ใช้เลิกการติดต่อกับ PalmDaemond ปุ่ม Exit ใช้ออกจากโปรแกรม และปุ่ม Find Service ใช้ค้นหา Service เมื่อเชื่อมต่อกับระบบอยู่
- ปุ่มควบคุมทิศทางของหุ่นยนต์ ใช้สำหรับควบคุมทิศทางของหุ่น เมื่อเลือกตัวหุ่นที่จะบังคับแล้ว

- ปุ่มตัวเลขเอนกประสงค์ ใช้ช่วยในสถานะต่างๆของโปรแกรม เช่น ในเวลาการค้นหา Service ก็จะใช้เลือก Service ที่ต้องการเมื่อค้นหาพบ
- หน้าจอแสดงผลข้อมูลจากเซิร์ฟเวอร์ แสดงผลข้อมูลและข้อความที่ส่งมาจากเซิร์ฟเวอร์
- แถบแสดงสถานะของโปรแกรม แสดงสถานะของโปรแกรมในขณะนั้น

แต่จากการศึกษาพบว่า ไม่สามารถให้โปรแกรมสำหรับปาล์มเชื่อมต่อกับระบบของจินี่ได้โดยตรงเนื่องมาจาก ชุด API ของ Java2MicroEdition ซึ่งเป็น API สำหรับปาล์มนั้นไม่มีส่วนของ Java RMI ซึ่งเป็นส่วนประกอบสำคัญที่ใช้ในการติดต่อกับระบบเครือข่ายของจินี่ ทำให้ต้องมีการพัฒนาโปรแกรมเพิ่มเติม เพื่อทำให้ปาล์มสามารถทำงานกับเครือข่ายของจินี่ได้ โดยที่โปรแกรมที่พัฒนาเพิ่มเติมนั้น จะทำการสร้างออบเจ็กต์ที่ทำตัวเป็นเหมือนพรอกซี่ (Proxy) ของปาล์มแต่ละเครื่องที่เชื่อมต่อเข้ามาในระบบ ซึ่งพรอกซี่นั้นทำหน้าที่ในการเชื่อมต่อไปยังเครือข่ายของจินี่ โปรแกรมนี้ชื่อว่าปาล์มเดมอน (PalmDaemon)

7.2.2 การออกแบบปาล์มเดมอน

เครื่องปาล์มจะเชื่อมต่อกับปาล์มเดมอนผ่านทางโพรโตคอล TCP/IP โดยสร้างซ๊อกเก็ตมายังโปรแกรมปาล์มเดมอน ซึ่งนั้นจะทำงานอยู่บนเครื่องพีซีที่มีพลังการประมวลผลสูงกว่า ทำให้สามารถใช้ Java RMI ของ Java 2 Standard Edition ได้ โดยปาล์มเดมอนนั้นจะทำการสร้างออบเจ็กต์ ที่มีลักษณะเป็นพรอกซี่ซึ่งเป็นเหมือนตัวแทนของปาล์มในการเชื่อมต่อไปยังระบบเครือข่ายของจินี่



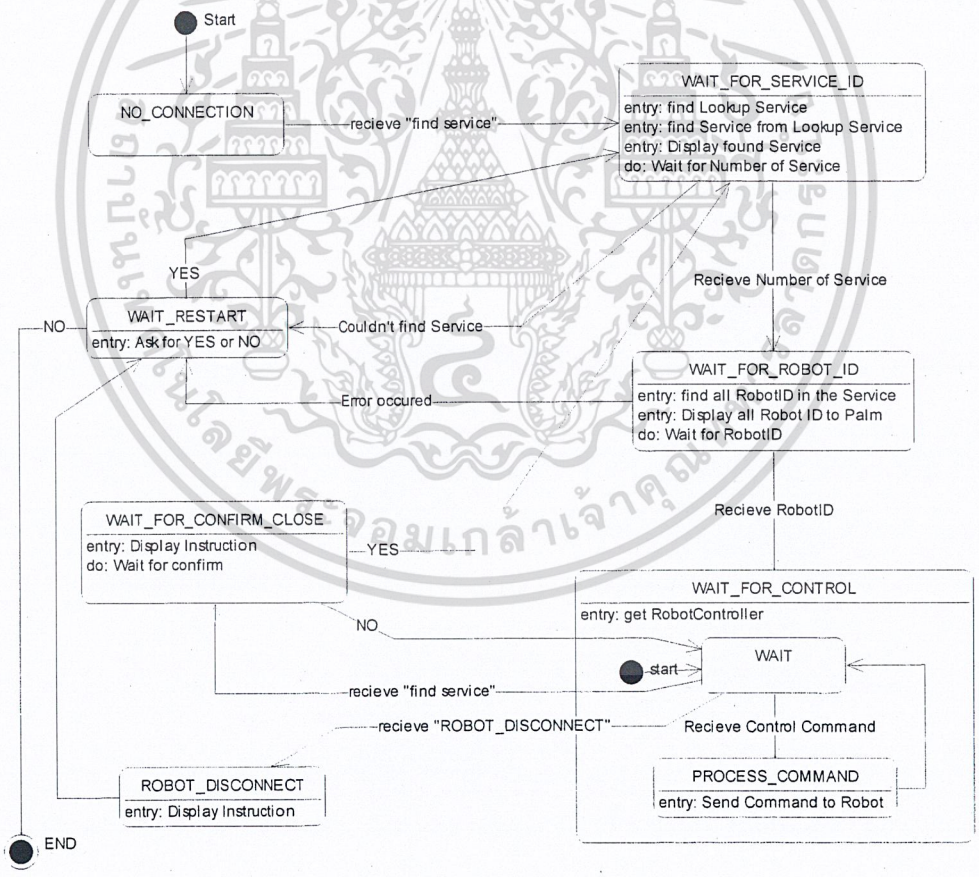
รูปที่ 7-9 การออกแบบโปรแกรมปาล์มเดมอน

การคุยกันระหว่างปาล์มและปาล์มเดมอน หลังจากที่ปาล์มเดมอนทำการสร้างพรอกซี่ออบเจ็กต์ที่ชื่อปาล์มพรอกซี่ (PalmProxy) แล้ว ซอกเก็ตของปาล์มจะถูกโอนมาให้ปาล์มพรอกซี่ดูแลต่อ การทำงาน

ของปาล์มพรอกซี่จะรับข้อมูลคำสั่งมาจากปาล์มโดยเมื่อมีการกดปุ่มใดๆบนหน้าจอของปาล์มจะมีการส่งคำสั่งของปุ่มนั้นมาเป็นสตริง ให้กับปาล์มพรอกซี่ โดยที่ปาล์มพรอกซี่จะนำคำสั่งที่ส่งมามาประมวลผลตามสเตทต่างๆของปาล์มพรอกซี่ คำสั่งต่างๆที่ส่งมาจากการกดปุ่มคือ

ลูกศรบังคับทิศทาง ขึ้น	ส่งข้อมูล	“up”
ลูกศรบังคับทิศทาง ลง	ส่งข้อมูล	“down”
ลูกศรบังคับทิศทาง ซ้าย	ส่งข้อมูล	“left”
ลูกศรบังคับทิศทาง ขวา	ส่งข้อมูล	“right”
ปุ่มวงกลมตรงกลาง	ส่งข้อมูล	“stop”
ปุ่ม Find Service	ส่งข้อมูล	“find service”
ปุ่ม !	ส่งข้อมูล	“turn on”
ปุ่ม ?	ส่งข้อมูล	“turn off”

เสตทต่างๆของปาล์มพรอกซี่ สามารถเขียนอธิบายได้ด้วย State Diagram ของ UML ได้คือ



รูปที่ 7-10 สเตทไดอะแกรมของออบเจ็กต์ปาล์มพรอกซี่

NO_CONNECTION เป็นเสตทแรกเมื่อปาล์มทำการติดต่อมายังระบบของปาล์มเดมอนแล้วและปาล์มพรอกซี่ถูกสร้างขึ้น จากนั้นซ๊อกเก็ตของปาล์มจะถูกส่งมาให้กับปาล์มพรอกซี่ดูแล เมื่อผู้ใช้กดปุ่ม

Find service คำสั่ง “find service” ก็จะถูกส่งไปยังปาล์มพรอกซี่ ปาล์มพรอกซี่จะสร้างออบเจ็กต์ ServiceFinder ขึ้นมา โดย ServiceFinder จะทำการค้นหา ลุคอัพเซอร์วิส จากในเครือข่ายด้วยวิธีการทำ Multicasting เมื่อ ServiceFinder ค้นพบ ลุคอัพเซอร์วิส แล้วก็จะทำการค้นหาเซอร์วิสที่มีอินเตอร์เฟส ตรงกับเซอร์วิสที่ปาล์มต้องการคือ RobotRoomService ซึ่งมีอินเตอร์เฟสดังนี้

```
import java.rmi.*;
import net.jini.core.event.*;

public interface RobotRoomService {
    public RobotRoomServiceRegistration getInstance(long duration)
                                        throws RemoteException;
    public EventRegistration trackEvents(long duration,
                                        RemoteEventListener rel,
                                        MarshalledObject key)
                                        throws RemoteException;
}
```

ServiceFinder จะคืนค่าจากการค้นหาเซอร์วิส เป็นอาร์เรย์ของเซอร์วิสออบเจ็กต์ ของเซอร์วิสที่ตรงตามความต้องการ

เมื่อปาล์มพรอกซี่ค้นหาเซอร์วิสที่ตรงกับความต้องการพบ ก็จะแจ้งให้กับปาล์มทราบว่าพบเซอร์วิสจำนวนกี่ตัวและมีชื่อว่าอะไรบ้าง โดยอ่านค่าชื่อของเซอร์วิสมาจากเซอร์วิสออบเจ็กต์ จากนั้นปาล์มพรอกซี่ก็จะเปลี่ยนเสตทไปเป็น WAIT_FOR_SERVICE_ID เพื่อที่จะรอให้ผู้ใช้กดปุ่มตัวเลขเพื่อเลือกเซอร์วิสที่ต้องการ

เมื่อผู้ใช้เลือกเซอร์วิสที่ต้อง การแล้วปาล์มพรอกซี่ก็จะทำการติดต่อกับตัวเซอร์วิสโดยตรงโดยทำผ่านเซอร์วิสออบเจ็กต์ที่มีอยู่ กระบวนการที่ปาล์มพรอกซี่ติดต่อกับเซอร์วิสเพื่อจะเตรียมการในการบังคับหุ่นยนต์มีดังนี้

1) ร้องขอ Registeration ออบเจ็กต์จากเซอร์วิส ด้วย เมธอด getInstance() ของเซอร์วิสออบเจ็กต์ โดยส่ง duration time ของ Lease ที่ต้องการให้กับ เมธอด นี้ ออบเจ็กต์ที่คืนค่ามาจะมีชนิดเป็น RobotRoomServiceRegistration ซึ่งจะมี Lease ของเซอร์วิสนี้ติดมากับ Registration ออบเจ็กต์ด้วย สามารถเรียกใช้ได้จาก เมธอด getLease() โดยปาล์มพรอกซี่ จะนำ Lease ที่ได้จาก RobotRoomServiceResistration ไปส่งให้ LeaseRenewalManager เป็นผู้ดูแล
2) ลงทะเบียนขอรับ Event จากเซอร์วิสโดย เมธอด trackEvents() ของเซอร์วิสออบเจ็กต์ โดยส่งพารามิเตอร์ให้กับ เมธอด นี้ 3 ตัวคือ duration time ของ Lease ของการรับฟัง Event ที่ต้องการ, ออบเจ็กต์ที่ทำหน้าที่เป็นตัวดักฟังเหตุการณ์ทางไกล (Remote Event Listener) และ Marshalled Object ซึ่งอาจส่งค่าเป็น null ให้ไปก็ได้ การลงทะเบียนนี้จะคืนค่ามาเป็นออบเจ็กต์ที่มีชนิดเป็น EventRegistration ซึ่งจะมี Lease ของการรับฟัง Event ติดมาด้วย โดยสามารถเรียกใช้ได้จาก เมธอด getLease() และส่งต่อให้กับ LeaseRenewalManager เป็นผู้ดูแล
3) ทำการตรวจดูว่ามีหุ่นยนต์ลงทะเบียนอยู่กี่ตัวในเซอร์วิสนั้นด้วย เมธอด getRobotID() ของ RobotRoomServiceRegistration ออบเจ็กต์ ซึ่งจะคืนค่ามาเป็นอาร์เรย์ของ ID ของหุ่นยนต์ทั้งหมดในเซอร์วิส แล้วทำการแสดงผลให้กับผู้ใช้ทราบ

4) เปลี่ยนเสตทของปาล์มพรอกซี่เป็น WAIT_FOR_ROBOT_ID เพื่อรอรับการเลือกหุ่นที่จะบังคับจากผู้ใช้
5) เมื่อผู้ใช้กดปุ่มตัวเลขเพื่อเลือกว่าจะควบคุมหุ่นยนต์ตัวไหนแล้ว ปาล์มพรอกซี่ก็จะทำการร้องขอออบเจ็กต์ที่ใช้ในการควบคุมหุ่น (RobotController) โดยใช้ เมธอด getController() จาก RobotRoomServiceRegistration โดยส่งพารามิเตอร์ให้คือ ID ของหุ่นเพื่อจะเลือกว่าจะควบคุมหุ่นตัวใด ออบเจ็กต์นี้เป็นออบเจ็กต์ระยะไกล ซึ่งตัวออบเจ็กต์จริงๆนั้นจะทำงานอยู่ที่ฝั่งเซอร์วิส หลังจากที่ได้ออบเจ็กต์นี้มาแล้ว ปาล์มพรอกซี่ก็จะเปลี่ยนสถานะเป็น WAIT_FOR_CONTROL เพื่อรอรับการกดปุ่มทิศทางจากฝั่งของปาล์ม
6) การบังคับหุ่นยนต์จะใช้การเรียกใช้ เมธอด ของโรบอตคอนโทรลเลอร์ เช่นเมื่อฝั่งปาล์มกดปุ่มลูกศรขึ้น ปาล์มก็จะส่ง “up” มาให้กับปาล์มพรอกซี่ เมื่อปาล์มพรอกซี่รับคำสั่งมาแล้วก็จะมาแปลว่าเป็นคำสั่งอะไร ถ้าหากพบว่าเป็นคำสั่ง “up” ก็จะทำการเรียกใช้ เมธอด forward() ของโรบอตคอนโทรลเลอร์ เพื่อสั่งให้หุ่นยนต์เดินไปข้างหน้า

บทที่ 8
การออกแบบฮาร์ดแวร์ของหุ่นยนต์

ไมโครโรบอตเป็นอุปกรณ์ที่ใช้ร่วมกับระบบจีนี่ เป็นตัวจำลองการบริการซึ่งมีในระบบ จะถูกใช้โดยคอมพิวเตอร์ปาล์ม ไมโครโรบอตจะจำลองการบริการบังคับให้กับระบบ คอมพิวเตอร์ปาล์ม

โดยทั่วไปไมโครโรบอตเป็นโรบอตที่ใช้ไมโครคอนโทรลเลอร์ตระกูล 8051 ใช้การเคลื่อนที่โดยสเตปปิ้งมอเตอร์ การติดต่อกับคอมพิวเตอร์ใช้พอร์ทอนุกรม การสื่อสารแบบโดยผ่านทางอินฟราเรด ที่ไมโครโรบอตมีเขียนโปรแกรมตอบสนองต่อสัญญาณบังคับที่มาจากโรบอตเดมอน ซึ่งเป็นภาษาแอสเซมบลีของไมโครคอนโทรลเลอร์ 8051 มีการส่งสัญญาณปิง สัญญาณเตือนการชน และสัญญาณบอกโรบอตเดมอนว่าเกิดการชนกับวัตถุ

ไมโครโรบอตเริ่มแรกต้องทำการส่งสัญญาณปิงของตนเอง คือสัญญาณปิงที่มีไอดีของตัวเอง ไปให้โรบอตเดมอน ในระบบจีนี่ก็จะเพิ่มเซอร์วิสการบังคับของโรบอตเข้าไป จากนั้นเมื่อคอมพิวเตอร์ปาล์มติดต่อเข้ามา ก็จะแจ้งเซอร์วิสของ ไมโครโรบอตในแต่ละไอดีของไมโครโรบอตก็เป็นแต่ละเซอร์วิส คอมพิวเตอร์ปาล์มสามารถเลือกในการบังคับได้ว่าจะบังคับตัวใด และสามารถเลิกการบังคับได้

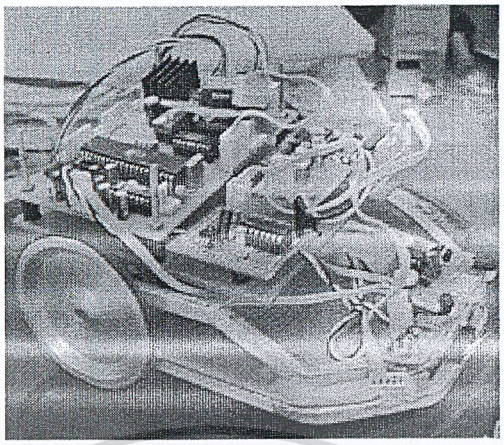
เมื่อไมโครโรบอตได้ตรวจจับว่ามีสิ่งกีดขวางทางจะทำการแจ้งไปให้โรบอตเดมอน ซึ่งแสดงผลไปที่ปาล์มคอมพิวเตอร์เพื่อให้ผู้บังคับทำการตัดสินใจในการบังคับ และเมื่อไมโครโรบอตเกิดการชนกับสิ่งกีดขวาง จะส่งสัญญาณไปแจ้งเช่นกัน และในโปรแกรมแอสเซมบลีของไมโครโรบอตได้ใส่รูทีนเพื่อทำการตีจากสิ่งกีดขวางที่ชนไว้เป็นระบบอัตโนมัติที่จะกระทำเมื่อเกิดการชน

การส่งสัญญาณและการเคลื่อนไหวอัตโนมัตินี้ได้ใส่ไว้ในอินเทอร์รัปต์รูทีน มีข้อดีคือไม่ต้องทำการตรวจสอบด้วยการเขียนโปรแกรมเอง แต่ตัวไมโครคอนโทรลเลอร์จะทำการตรวจสอบให้เองโดยอัตโนมัติ ในแต่ละมาชีนไซเคิล

8.1 โครงสร้างฮาร์ดแวร์ของไมโครโรบอต

8.1.1 โครงของไมโครโรบอต

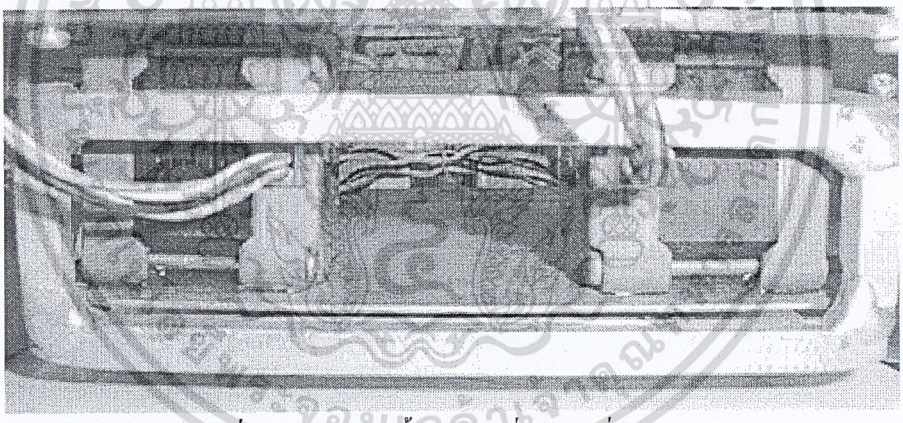
โครงของไมโครโรบอตทำมาจากอลูมิเนียมเส้นเชื่อมต่อกันขึ้นเป็นโครง ทำให้มีน้ำหนักเบา แต่ไม่สามารถรับแรงกดได้มากนัก แต่ก็มีความแข็งแรงเพียงพอในการใช้งานในโครงงาน



รูปที่ 8-1 แสดงโครงสร้างไมโครโรบอตโดยรวม

8.1.2 มอเตอร์ขับเคลื่อนไมโครโรบอต

ไมโครโรบอตใช้สเตปปิ้งมอเตอร์ในการขับเคลื่อน โดยแรงดันที่ใช้จ่ายมอเตอร์ควรจะมากกว่า 16 โวลท์ เพื่อจะได้ทอร์กที่เพียงพอในการขับเคลื่อน

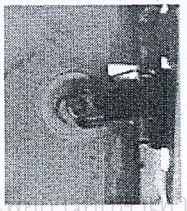


รูปที่ 8-2 แสดงสเตปปิ้งมอเตอร์ที่ใช้ขับเคลื่อน

8.1.3 ล้อของไมโครโรบอต

8.1.3.1 ล้อหน้า

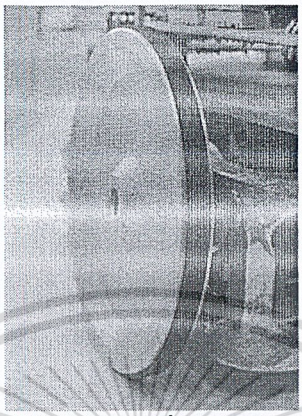
ใช้ลูกกลิ้งที่ใช้กับเครื่องเล่นเทป ทำการเจาะรูให้ได้ขนาดแล้วขันน็อตยาว แล้วใช้น็อตขันสองตัวให้ติดอยู่กับฐานอลูมิเนี่ยม โดยให้สามารถหมุดได้อย่างอิสระ เพื่อให้การเลี้ยวของไมโครโรบอตเป็นไปอย่างราบรื่นไม่ติดขัด



รูปที่ 8-3 แสดงล้อหน้า

8.1.3.2 ล้อขับเคลื่อน

ล้อที่ใช้ขับเคลื่อนของไมโครโรบอตเป็นล้ออลูมิเนียมกัด ซึ่งเป็นชนิดเดียวกับล้อของไมโครเมาส์ ยึดติดโดยตรงกับมอเตอร์ โดยไม่มีเพลา ซึ่งลักษณะนี้คล้ายกับไมโครเมาส์



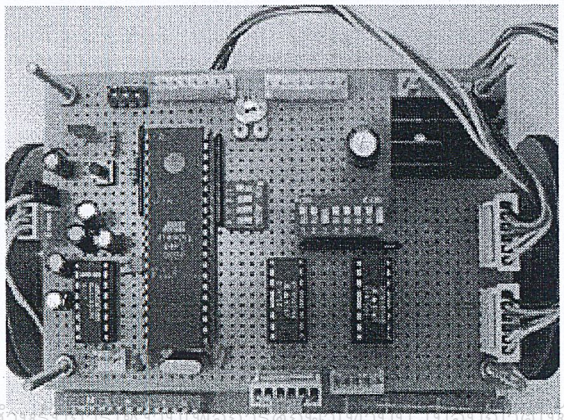
รูปที่ 8-4 แสดงล้อขับเคลื่อนของไมโครบอต

8.2 วงจรอิเล็คทรอนิกส์

8.2.1 วงจรควบคุมหลัก

ในแผงควบคุมหลักจะประกอบด้วย 3 วงจรคือ วงจรควบคุมหลัก วงจรพาวเวอร์ซัพพลายหลัก และวงจรขับสเตปปิ้งมอเตอร์ ส่วนของวงจรควบคุมหลักของไมโครโรบอตเป็นส่วนที่ใช้ควบคุมการทำงานของไมโครโรบอตซึ่งประกอบไปด้วยไมโครโปรเซสเซอร์ 8051 และอุปกรณ์รอบข้าง รวมถึงวงจรที่ใช้ติดต่อกับพอร์ตอนุกรมซึ่งเป็นมาตรฐานอาร์เอส232 ใช้ไอซีแมกซ์232 และดิพสวิตซ์ ซึ่งใช้ในการตั้งหมายเลขของไมโครโรบอต ส่วนของวงจรควบคุมหลักต้องการไฟ 5 โวลท์มาจากวงจรพาวเวอร์ซัพพลายหลัก

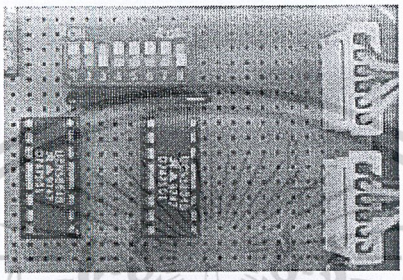
วงจรควบคุมหลักซึ่งใช้ไมโครคอนโทรลเลอร์ 8051 เพียงตัวเดียว ไม่จำเป็นต้องใช้หน่วยความจำภายนอกเพราะโปรแกรมที่รันนั้นไม่ต้องการทรัพยากรมาก ในส่วนของผังไฟฟ้าของวงจรหลักอยู่ในหน้า sch1



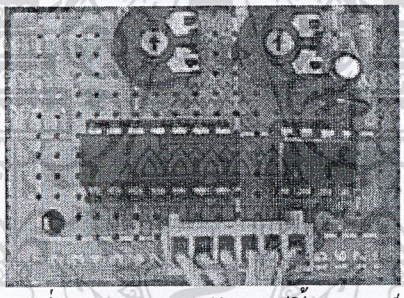
รูปที่ 8-5 แสดงวงจรหลักของไมโครโรบอต

8.2.2 วงจรขับสเตปปิ้งมอเตอร์

วงจรขับสเตปปิ้งมอเตอร์ใช้ไอ ซียูซีเอ็น5804 ซึ่งรับเพียงสัญญาณนาฬิกาเพื่อเป็นจังหวะในการจ่ายกระแสไฟให้สเตปปิ้งมอเตอร์ ใช้ไฟเลี้ยง 5 โวลท์จากวงจรพาวเวอร์ซัพพลายหลักและรับไฟขับ สเตปปิ้งมอเตอร์ ซึ่งต่อมาจากพาวเวอร์ซัพพลายโดยตรงประมาณ 15 โวลท์ขึ้นไปเพื่อทอร์กในการขับเกลื่อนไมโครโรบอต ดังที่กล่าวไปแล้วว่าวงจรขับสเต็ปปิ้งมอเตอร์ต้องใช้สัญญาณนาฬิกาดังนั้นต้องใช้วงจรสร้างสัญญาณนาฬิกาด้วย



รูปที่ 8-6 แสดงวงจรสร้างสัญญาณนาฬิกาสำหรับสเตปปิ้งมอเตอร์



รูปที่ 8-7 แสดงวงจรขับสเตปปิ้งมอเตอร์

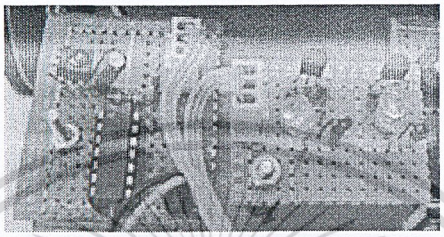
วงจรขับสเต็ปปิ้งมอเตอร์สามารถควบคุมทิศทางและการหมุนของมอเตอร์โดยระดับโวลท์เตจทีทีแอ็ล และสามารถเลือกโหมดในการขับสเต็ปปิ้งมอเตอร์ด้วยดิพสวิตซ์ แผนผังไฟฟ้าของวงจรขับ สเตปปิ้งมอเตอร์อยู่ในหน้า sch3

8.2.3 วงจรพาวเวอร์ซัพพลายหลัก

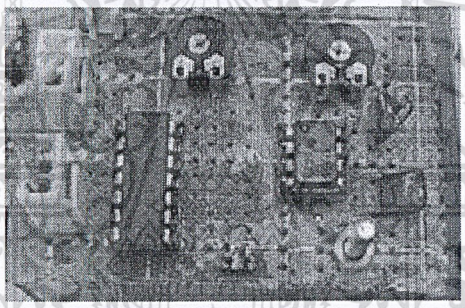
วงจรพาวเวอร์ซัพพลายหลักทำหน้าที่เรกูเลตไฟที่มาจากพาวเวอร์ซัพพลายและจ่ายไฟเลี้ยงวงจรทั้งหมดของไมโครโรบอตซึ่งต้องการไฟเลี้ยง 5 โวลท์ ใช้ไอซี 7805 ในการแปลงไฟที่มาจากพาวเวอร์ซัพพลายให้เท่ากับ 5 โวลท์ ใช้คาพาซิสเตอร์ต่อคร่อมให้กระแสเรียบยิ่งขึ้น แผนผังไฟฟ้าของวงจรพาวเวอร์ซัพพลายหลักอยู่ในหน้า sch2

8.2.4 วงจรสื่อสารโดยใช้อินฟาเรด

ใช้สื่อสารกับโรบอตเดมอน โดยใช้อินฟราเรด ซึ่งใช้การส่งข้อมูลแบบอนุกรมเป็นหลักนำสัญญาณมารวมเข้ากับคลื่นพาหะ แล้วส่งไป ทางฝั่งรับก็นำสัญญาณที่ได้รับมาถอดเอาแต่สัญญาณที่ส่งมา สำหรับการรวมคลื่นใช้การแอนด์บิต คลื่นพาหะมีลักษณะเป็นสัญญาณนาฬิกาที่ความถี่ 38 กิโลเฮิร์ต



รูปที่ 8-8 แสดงวงจรสื่อสารด้วยอินฟรานรด



รูปที่ 8-9แสดงวงจรกำเนิดคลื่นพาหะและม็อดความถี่

8.2.4.1 วงจรรับ

ใช้ไอซี ทีเอ็สโอพี 7438 ซึ่งจะรับเพียงสัญญาณอินฟราเรดที่มีความถี่ 38 เฮิร์ตเท่านั้น เป็นไอซีที่ใช้รับสัญญาณรีโมตคอนโทรลของโทรทัศน์ วีดีโอ หรือเครื่องเสียงที่มีรีโมตคอนโทรล เมื่อมีสัญญาณเข้ามาที่ขาสัญญาณเอาท์พุทของไอซีจะเปลี่ยนเป็นลอจิกต่ำ ทำให้การในการใช้งานจริงต้องใช้ไอซี น็อตเกท เพื่อกลับสัญญาณให้ถูกต้อง

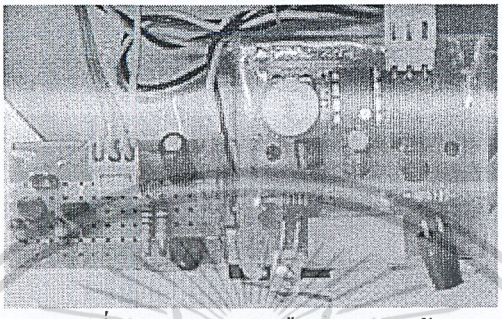
8.2.4.2 วงจรส่ง

วงจรส่งใช้คลื่นพาหะเป็นสัญญาณนาฬิกาที่มีความถี่ 38 เฮิร์ต แอนด์กับสัญญาณจากพอร์ตอนุกรมของไมโครโพรเซสเซอร์ 8051 ได้สัญญาณแล้วส่งเป็นอินฟราเรด โดยในการส่งต้องมีการขยายสัญญาณให้มีความแรงเพียงพอที่จะขับแอลอีดี โดยใช้วงจรขยายสัญญาณเพื่อลดความเพี้ยนของข้อมูลที่ส่ง และทำให้ส่งได้ไกลกว่าเดิม

วงจรขยายสัญญาณทำได้โดยใช้ทรานซิสเตอร์ความถี่สูงเบอร์ บีซี 557 สามารถรองรับความถี่ได้ 200 เมกกะเฮิร์ต ซึ่งเพียงพอต่อการใช้งานแผนผังไฟฟ้าของวงจรสื่อสารด้วยอินฟราเรดซึ่งประกอบด้วยส่วนรับ,ส่วนส่ง และวงจรขยายสัญญาณอยู่หน้าที่ sch 4

8.2.5 วงจรเซนเซอร์กันชนหน้า

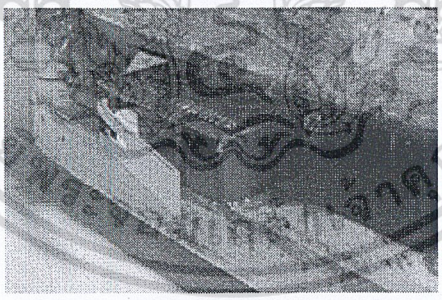
หลักการของวงจรคล้ายกับ วงจรสื่อสารโดยอินฟราเรด เพียงแต่ให้ส่งสัญญาณนาฬิกาความถี่ 38 กิโลเฮิร์ต ตลอด ถ้ามีวัตถุเข้าใกล้ในระยะที่สะท้อนได้ สัญญาณอินฟราเรดจะสะท้อนจากวัตถุมาตกกระทบที่ไอซี ทีเอ็สโอพี 7438 ทำให้สามารถตรวจสอบได้ว่าจะมีการชน แผนผังไฟฟ้าของวงจรเซนเซอร์กันชนหน้าอยู่ในหน้า sch 5



รูปที่ 8-10 แสดงวงจรเตือนการชนหน้า

8.2.6 วงจรลิมิตสวิตซ์ตรวจจับการชน

เพื่อตรวจจับ และส่งสัญญาณว่าเกิดการชนไปที่โรบอตเคมอน เป็นระบบลิมิตสวิตซ์ซึ่งต่อเข้ากับขาอินเตอร์รัปท์ภายนอกของไมโครคอนโทรลเลอร์ 8051 (ขา int0) จะถือว่ามีความสำคัญที่สุดในบรรดาอินเตอร์รัปท์ที่โปรแกรมไว้แผนผังไฟฟ้าแสดงการเชื่อมต่อของลิมิตสวิตซ์ตรวจจับการชนอยู่ในหน้า sch6



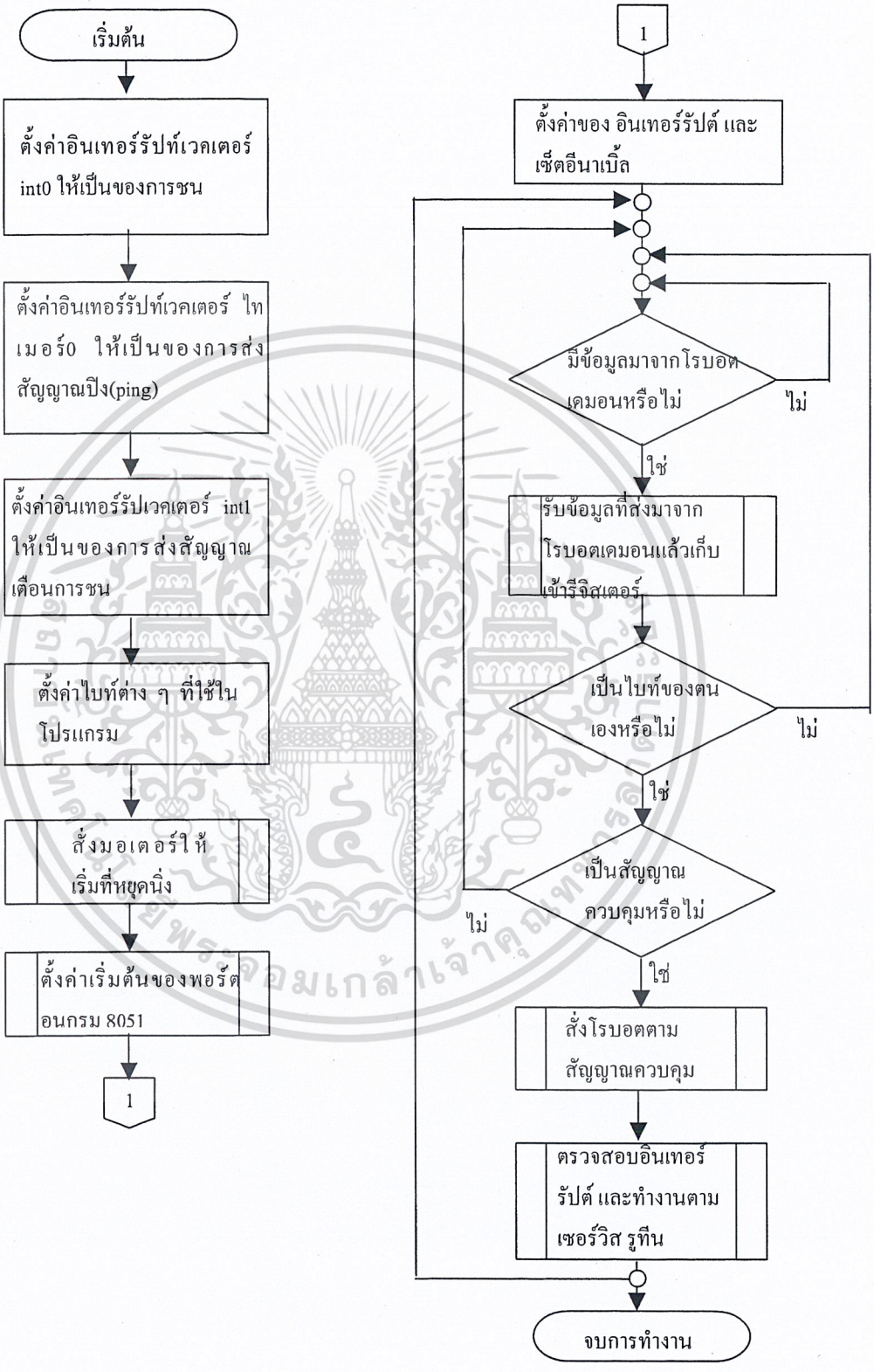
รูปที่ 8-11แสดงลิมิตสวิตซ์ตรวจสอบการชน

8.3 การออกแบบโปรแกรมควบคุม

โปรแกรมควบคุมเขียนด้วยโปรแกรมภาษาแอสเซมบลีสำหรับ 8051 มีการออกแบบโปรแกรมและขั้นตอนการทำงานของโปรแกรม ซึ่งสามารถแบ่งเป็นโมดูลต่างๆได้ดังนี้

8.3.1 ส่วนของโปรแกรมหลัก

การทำงานของโปรแกรมหลักสามารถเขียนเป็นโฟลวชาร์ทได้ดังนี้



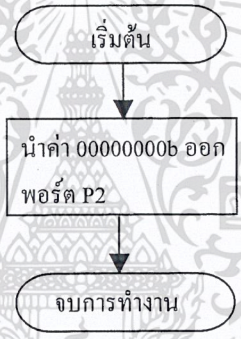
รูปที่ 8-12 แสดงโฟลวชาร์ตของการทำงานของโปรแกรม

สามารถอธิบายคร่าวๆได้คือเริ่มต้นโปรแกรม จะทำการเซ็ตค่าต่าง ๆ ที่จำเป็นคือ เซ็ตค่าพอร์ตอนุกรม เซ็ตส่วนของอินเทอร์รัปต์ จากนั้นก็จะรอรับสัญญาณจากโรบอตเดมอนเพียงอย่างเดียว ถ้ามีอินเทอร์รัปต์ที่มาจากภายนอก ก็คือสัญญาณเตือนการชน และสัญญาณการชน ไมโครโรบอตจะส่งสัญญาณแจ้งให้โรบอตเดมอนทราบ และถ้าถึงระยะเวลาที่กำหนด ไทเมอร์ อินเทอร์รัปต์จะทำงาน ก็คือจะส่งสัญญาณปิงไปแจ้งแก่โรบอตเดมอนทราบว่าตัวไมโครโรบอตยังคงอยู่ ส่วนในกรณีที่มีสัญญาณจากโรบอตเดมอน ไมโครโรบอตจะทำการประมวลผล ตรวจสอบ ว่าเป็นสัญญาณของตัวเองหรือไม่ และเป็นคำสั่งควบคุมหรือไม่ แล้วก็ทำตามคำสั่ง จากนั้นก็จะรอคำสั่งอีกครั้ง

8.3.2 ส่วนของโปรแกรมย่อย

8.3.2.1 สั่งให้มอเตอร์หยุด

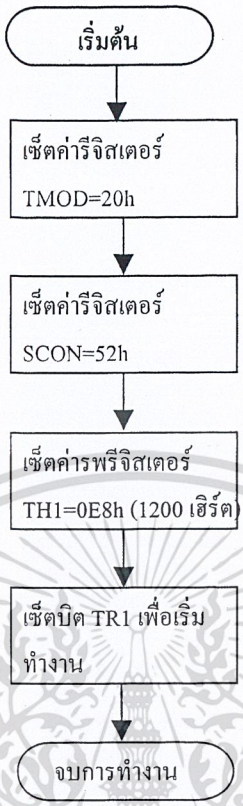
เป็นการเริ่มต้นการทำงานของไมโครโรบอต โดยสั่งให้ไมโครโรบอตหยุดนิ่ง สามารถเขียนโฟลวชาร์ทได้ดังนี้



รูปที่ 8-13แสดงโฟลวชาร์ทในการสั่งให้ไมโครโรบอตหยุด

8.3.2.2 ตั้งค่าเริ่มต้นกับพอร์ตอนุกรม 8051

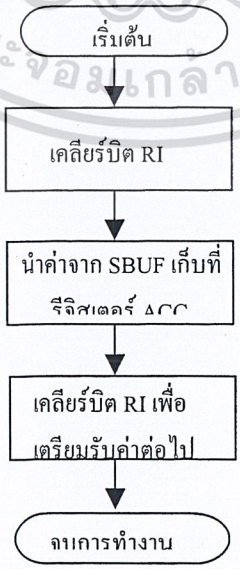
เป็นการตั้งค่าโหมด และอัตราการส่งข้อมูล(บอร์ดเรท)ให้กับไมโครโรบอต ที่จะใช้ติดต่อกับโรบอตเดมอนซึ่งต้องมีค่าเท่ากัน ในการส่งและรับ เพราะเป็นการส่งข้อมูลแบบอนุกรม จังหวะที่ใช้รับข้อมูลเป็นต้องเท่ากัน



รูปที่ 8-14 แสดงโฟลวชาร์ทในการเซ็คค่าของพอร์ตอนุกรม 8501

8.3.2.3 รับข้อมูลที่ส่งมาจากโรบอตเดมอนแล้วเก็บเข้ารีจิสเตอร์

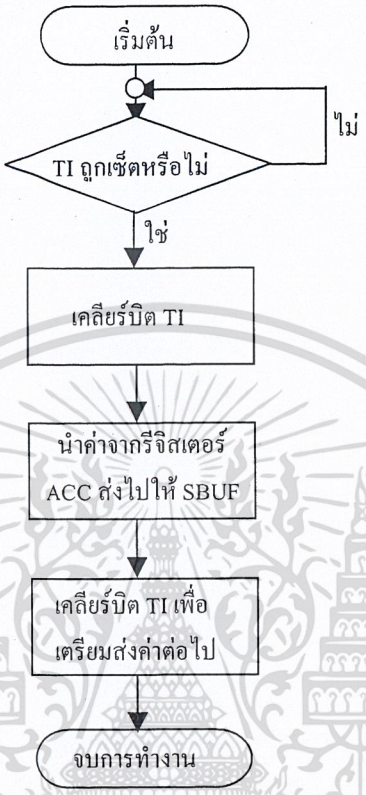
เป็นการรับค่าแบบอนุกรม แล้วเก็บข้อมูลซึ่งมีลักษณะเป็นไบท์จากบัฟเฟอร์ของพอร์ตอนุกรม เข้าเก็บในรีจิสเตอร์แอคคิวมูเลเตอร์



รูปที่ 8-15 แสดงโฟลวชาร์ทในการรับค่าจากโรบอตเดมอนทางพอร์ทอนุกรม

8.3.2.4 ส่งข้อมูลจากโรบอตไปที่โรบอตเดมอน

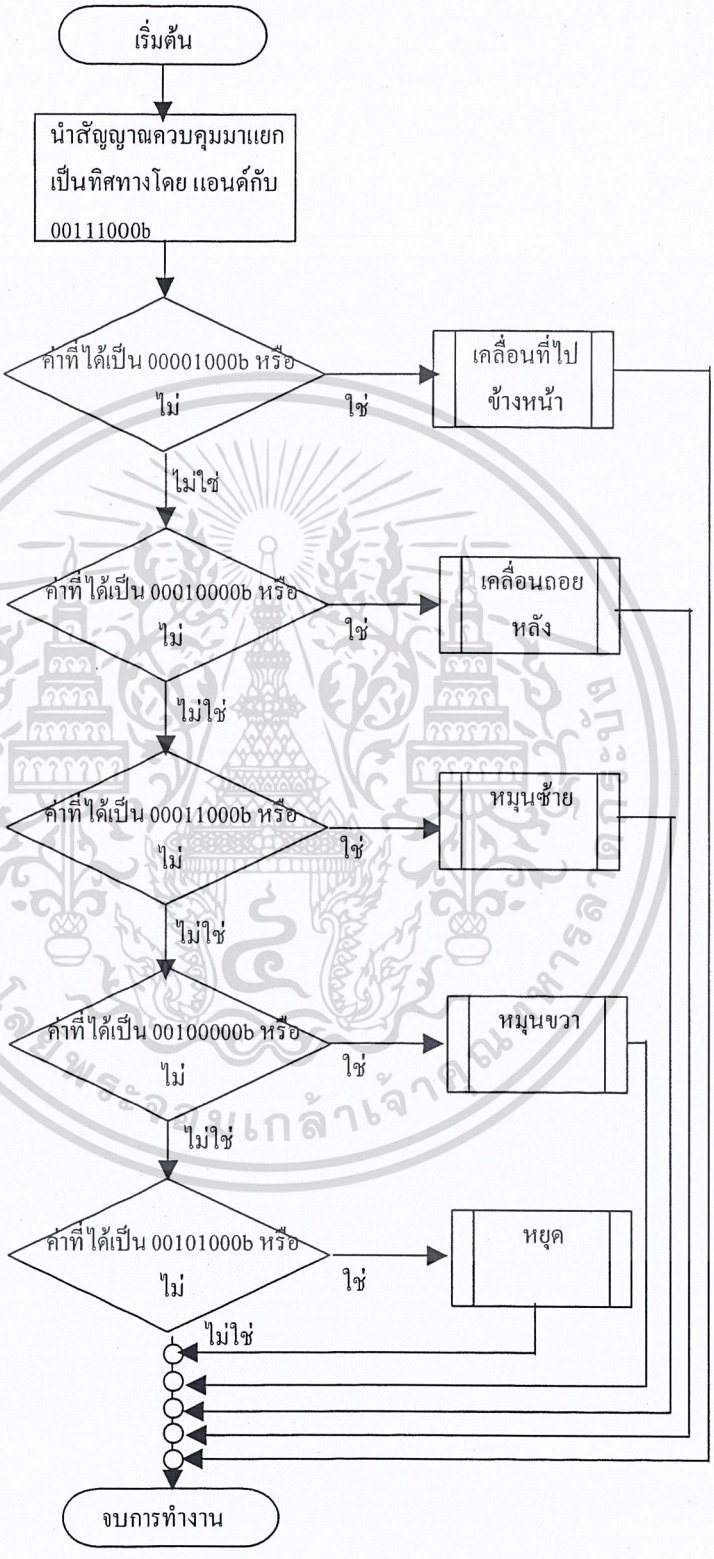
เป็นการส่งข้อมูลที่เป็นไบท์ ไปให้โรบอตเดมอนผ่านทางพอร์ตอนุกรม



รูปที่ 8-16 แสดงโฟลวชาร์ทการส่งข้อมูลโดยพอร์ทอนุกรม

8.3.2.5 สั่งโรบอตตามสัญญาณควบคุม

เป็นการทำสัญญาณควบคุมที่ได้รับ มาแปลความแล้วสั่งให้ไมโครโรบอตทำงานตามที่สั่ง



รูปที่ 8-17 แสดงโฟลวชาร์ทของการสั่งให้ไมโครบอตปฏิบัติตามสัญญาณควบคุม

8.3.2.6 ส่วนสั่งการเคลื่อนที่ของไมโครโรบอต

เป็นรูทีนการเคลื่อนไหวของไมโครโรบอต ภายในรูทีนจะทำการสั่งไมโครโรบอตให้เคลื่อนโดยใช้การเขียนค่าพอร์ท P2 ด้วยค่าต่าง ๆ ซึ่งค่าต่าง ๆ เหล่านี้จะถูกส่งให้วงจรขับสเตปปิ้งมอเตอร์ ทำให้ได้การเคลื่อนไหวตามต้องการ การเคลื่อนไหวหลัก ๆ ได้แก่ เคลื่อนที่ไปข้างหน้า , เคลื่อนถอยหลัง , หมุนซ้าย (ทวนเข็มนาฬิกา) , หมุนขวา (ตามเข็มนาฬิกา) และหยุด

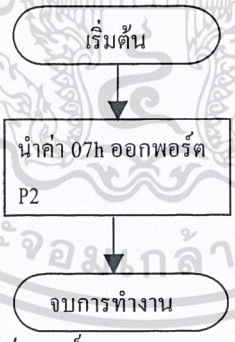
เคลื่อนที่ไปข้างหน้า

สามารถแสดงการทำงานด้วยโฟลวชาร์ทดังนี้

รูปที่ 8-18 แสดงโฟลวชาร์ทการของการเคลื่อนที่ไปข้าหน้า

เคลื่อนถอยหลัง

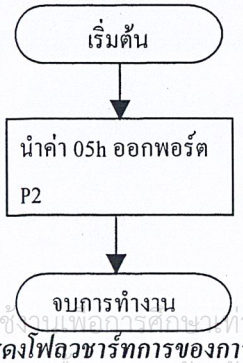
สามารถแสดงการทำงานด้ายโฟลวชาร์ทดังนี้



รูปที่ 8-19 แสดงโฟลวชาร์ทการของการเคลื่อนที่ถอยหลัง

หมุนซ้าย

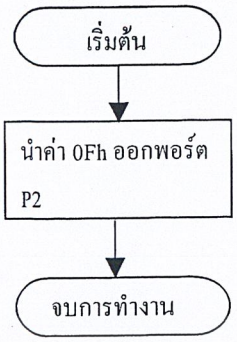
สามารถแสดงการทำงานด้ายโฟลวชาร์ทดังนี้



รูปที่ 8-20 แสดงโฟลวชาร์ทการของการหมุนซ้าย

หมุนขวา

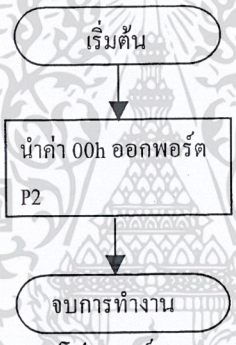
สามารถแสดงการทำงานด้วยโฟลวชาร์ทดังนี้



รูปที่ 8-21 แสดงโฟลวชาร์ทการของการหมุนขวา

หยุด

สามารถแสดงการทำงานด้วยโฟลวชาร์ทดังนี้



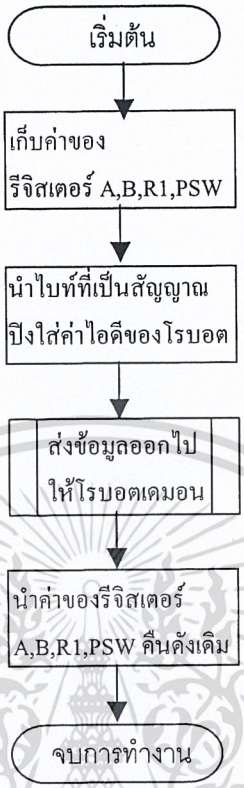
รูปที่ 8-22แสดงโฟลวชาร์ทการของการหยุด

8.3.2.7 ส่วนโปรแกรมอินเทอร์รัปท์

เป็นโปรแกรมซึ่งฝังเป็นอินเทอร์รัปท์ เป็นโปรแกรมที่จะทำเมื่อมีอีเว็นท์ที่สำคัญของไมโครโรบอตเกิดขึ้นที่ทั้งหมด 3 อีเว็นท์ที่สำคัญคือ ส่งสัญญาณปิง,เตือนการชน และส่งสัญญาณบอกการชน

ส่งสัญญาณปิง

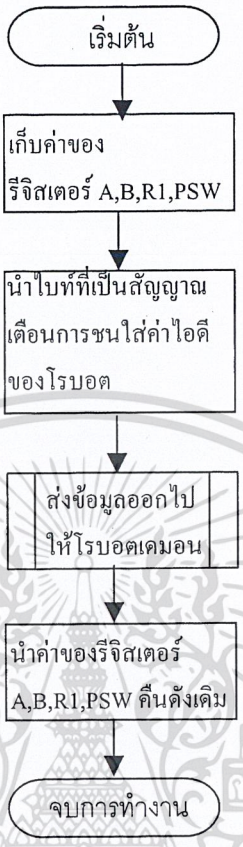
เป็นการส่งสัญญาณบอกโรบอตเดมอนว่า ไมโครโรบอตยังคงอยู่ในระบบซึ่งต้องมีการส่งตลอดเวลา หรือต้องส่งก่อนที่ตัวโรบอตเดมอนจะทำการตัดสินว่าขาดการติดต่อ หรือไทม์เอาท์ ทำการตั้งให้เป็นอินเทอร์รัปท์ไทม์เมอร์ 0 เพราะต้องมากระการทำตลอดเวลาซึ่งอยู่ในเซอร์วิส รูทีนที่ชื่อ pinging



รูปที่ 8-23 แสดงโฟลวชาร์ทอินเทอร์รัปท์รูทีนในการส่งสัญญาณปิง

เตือนการชน

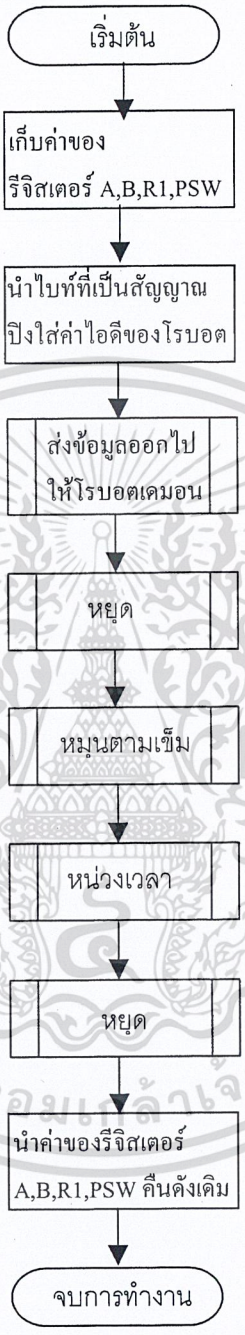
เป็นการส่งสัญญาณเตือนว่าจะมีการชน หรือเจอวัตถุที่ขวางทางไมโครโรบอตไปยังโรบอตเดมอน เป็นสัญญาณที่มาจากเซ็นเซอร์หน้าส่งเข้ามาให้กับไมโครคอนโทรลเลอร์ ที่ขาสัญญาณ int1 เป็นขาที่รับสัญญาณอินเทอร์รัปต์ภายนอก เซอร์วิส รูทีนชื่อว่า alerting



รูปที่ 8-24 แสดงโฟลวชาร์ทของอินเทอร์รัปท์รูทีนในการส่งสัญญาณเตือนการชน

การชน

เป็นการส่งสัญญาณว่าไมโครโรบอตเกิดการชนกับวัตถุซึ่งขวางอยู่ไปยังโรบอตเดมอน เป็นขาสัญญาณที่มาจากลิมิทสวิตซ์ที่อยู่หน้าสุด ส่งเข้ามาให้กับไมโครคอนโทรลเลอร์ ที่ขาสัญญาณ int 0 เป็นขาที่รับสัญญาณอินเทอร์รัปต์ภายนอก เซอร์วิส รูทีนชื่อว่า crashing โดยภายในเซอร์วิส รูทีนนั้นมีการสั่งไมโครโรบอตให้เคลื่อนไหวเพื่อหลบหลีกสิ่งกีดขวางโดยอัตโนมัติ



รูปที่ 8-25 แสดงโฟลวชาร์ทของอินเทอร์รัปท์รูทีนในการส่งสัญญาณเมื่อเกิดการชน

บทที่ 9
สรุปและวิจารณ์

9.1 วิเคราะห์ผลจากการออกแบบและสร้างระบบ

ผลของการสร้างระบบจำลองเครือข่ายของจินี่ จากจุดประสงค์ของโครงการนี้ที่ว่าด้วยการแสดงให้เห็นถึงความง่ายและคุณสมบัติต่างๆของจินี่ แต่จากระบบที่สร้างขึ้นมานั้น พบว่าการพัฒนาเซอร์วิสสำหรับจินี่ในปัจจุบันนี้นั้น ยังไม่สามารถจะสร้างเซอร์วิสได้ตามความสามารถทั้งหมดของจินี่ที่กล่าวไว้ในเอกสารของจินี่ เนื่องจากเทคโนโลยีที่เกี่ยวข้อง เช่น j2me หรือการสื่อสารแบบไร้สาย นั้นยังพัฒนาได้ไม่เพียงพอที่จะรองรับกับความสามารถของจินี่ทำให้ ระบบที่สร้างขึ้นมานั้น แสดงความสามารถของจินี่ได้ไม่เต็มที่ และยังคงมีจุดอ่อนในบางจุด เช่น การเรียกใช้เซอร์วิสจากปาล์ม ชุด API สำหรับปาล์มนั้นยังไม่มีส่วนของ Java RMI ซึ่งเป็นส่วนที่สำคัญสำหรับเครือข่ายของจินี่ ทำให้ต้องมีการสร้างโปรแกรมปาล์มเดมอนที่ทำงานอยู่บน PC เพื่อเป็นพรอกซี่ในการติดต่อไปยังจินี่และเซอร์วิส

เซอร์วิสที่สร้างนี้นั้นไม่สามารถสร้างให้ใช้ความสามารถของจินี่ได้อย่างครบถ้วน แต่เลือกมาเพียงบางส่วนเพื่อให้เหมาะสมกับตัวเซอร์วิสเท่านั้น และคุณสมบัติบางอย่างนั้น ก็ไม่มีความจำเป็นที่จะต้องใช้ และบางส่วนก็ไม่สามารถพัฒนาได้ทันตามกำหนด

การพัฒนาเซอร์วิสให้สามารถใช้งานได้ในสถานการณ์จริงนั้น ยังคงต้องรอให้เทคโนโลยีที่เกี่ยวข้อง พัฒนาได้มากกว่านี้ เพราะถ้าขาดเทคโนโลยีเหล่านี้จินี่ก็อาจจะไม่สามารถอำนวยความสะดวกให้และอาจจะทำให้ระบบนั้นยุ่งยากขึ้นด้วย

9.2 ปัญหาและอุปสรรค

สามารถสรุปปัญหาที่เกิดจากการทำโครงการได้ดังนี้

1) เนื่องจากจินี่เป็นเทคโนโลยีใหม่ ยังมีแหล่งข้อมูลอยู่น้อย จึงใช้เวลาในการศึกษาค่อนข้างมาก
2) ชุดพัฒนาของเทคโนโลยี j2me นั้นยังไม่สมบูรณ์ ทำให้การพัฒนาหุ่นยนต์และเซอร์วิสไม่ได้ตามแผนที่วางไว้ ซึ่งแผนเดิมคือการให้ตัวหุ่นยนต์ (เซอร์วิส) และตัวปาล์ม (ผู้ใช้เซอร์วิส) ติดต่อกับเครือข่ายของจินี่ได้โดยตรง จะทำให้ระบบมีความอัตโนมัติสูง แต่เนื่องจากไม่สามารถทำได้จึงต้องออกแบบระบบใหม่ให้ทำงานคล้ายระบบเดิม แต่ก็ลดความเป็นอัตโนมัติลงบ้าง
3) การพัฒนาการติดต่อสื่อสารระหว่างหุ่นยนต์และเซิร์ฟเวอร์ด้วยอินฟราเรดนั้น พบว่าสามารถทำได้ แต่ประสิทธิภาพที่ได้ ไม่ดีเท่าที่ควรเนื่องจากข้อจำกัดของสัญญาณ

อินฟราเรดที่ต้องส่งสัญญาณเป็นแนวตรง ทำให้ไม่เหมาะกับหุ่นยนต์ที่ต้องเคลื่อนที่ตลอดเวลา

4) การสร้างหุ่นยนต์สำหรับระบบนี้นั้น อุปกรณ์แต่ละชิ้นมีราคาแพง เนื่องจากงบประมาณที่จำกัดทำให้สามารถพัฒนาฮาร์ดแวร์ที่เป็นหุ่นยนต์ได้เพียงตัวเดียว
5) เครื่องปาล์มไพลอต ซึ่งใช้เป็นตัวควบคุมเซอร์วิสนั้นมีราคาแพง ในการพัฒนาจึงจำเป็นต้องใช้โปรแกรมจำลองเครื่องปาล์มไพลอตในการทดสอบแทน ทำให้ไม่ได้เห็นการทำงานของเครื่องปาล์มจริง ว่ามีปัญหาอะไรหรือไม่

9.3 ข้อดีและข้อเสียของระบบ

ข้อดี

- ระบบนี้ออกแบบด้วยออบเจ็กต์โอเรียนเต็ด ทำให้สามารถนำส่วนต่างๆมาใช้ใหม่ (Reuse) ได้ง่าย และได้ระบบที่มีประสิทธิภาพ และสามารถประยุกต์เอา Design Pattern แบบต่างๆมาใช้ในระบบได้
- เป็นระบบออบเจ็กต์แบบกระจาย ออบเจ็กต์ต่างๆสามารถทำงานร่วมกันได้ ถึงแม้ว่าจะอยู่ในต่างแพลตฟอร์ม และต่างสถานที่กัน
- มีการติดต่อกันโดยอัตโนมัติระหว่างผู้ใช้และเซอร์วิส โดยที่ผู้ใช้ไม่จำเป็นต้องรู้ตำแหน่งที่อยู่ของเซอร์วิสมาก่อน
- ผู้ใช้ไม่จำเป็นต้องรู้จัก ตัวเซอร์วิสมาก่อน เนื่องจากผู้ใช้ติดต่อกับเซอร์วิสผ่านทางอินเตอร์เฟส จึงไม่จำเป็นต้องรู้ถึงรายละเอียดวิธีการสร้างเซอร์วิส
- สามารถขยายระบบได้ง่ายโดยไม่ก่อให้เกิดความซับซ้อนมากนัก เนื่องจาก
- ง่ายต่อผู้ใช้เนื่องจากมีอินเตอร์เฟสที่ชัดเจน ในการที่มีเซอร์วิสอยู่หลายรูปแบบ และผู้ใช้ไม่จำเป็นต้องเปลี่ยนหรือปรับแต่งระบบของตัวเอง ในกรณีที่มีเซอร์วิสใหม่เกิดขึ้น

ข้อเสีย

- ทำงานได้ช้าหากเปรียบเทียบกับระบบแบบเดียวกันที่สร้างจากภาษาอื่นเช่น ภาษาซี
- เทคโนโลยีถูกจำกัดด้วยภาษาจาวา การจะพัฒนาระบบของจินี่ด้วยภาษาอื่นนั้นสามารถทำได้แต่ว่ายุ่งยากกว่า
- การออกแบบระบบด้วยออบเจ็กต์โอเรียนเต็ดทำได้ยาก

9.4 แนวทางในการพัฒนาต่อไป

1) สามารถสร้างเซอร์วิสแบบใหม่ๆ ด้วยอินเตอร์เฟสเดิม ทำให้ผู้ใช้สามารถใช้โปรแกรม Client กับเซอร์วิสได้หลายแบบ
2) สามารถพัฒนาระบบการสื่อสารระหว่างอุปกรณ์กับเซิร์ฟเวอร์ให้ดีมากขึ้น เช่น ใช้การสื่อสารแบบไร้สายด้วยคลื่นวิทยุ

3) ในอนาคตถ้าเครื่องปาล์มมีพลังในการประมวลผลมากขึ้น อาจจะพัฒนาให้ปาล์ม สามารถติดต่อกับจิ๋นี่ได้โดยตรงโดยไม่ต้องสร้างโปรแกรมมาทำหน้าที่เป็นพรอกซี่ให้ยุ่งยาก
4) สามารถพัฒนาหุ่นยนต์ที่ทำหน้าที่เป็นเซอร์วิสให้มีความสามารถมากขึ้น เช่น ติดกล้อง เพื่อส่งสัญญาณภาพมายังผู้ใช้สำหรับการควบคุมระยะไกล หรือพัฒนาให้มีความฉลาดมากขึ้นและโต้ตอบกับผู้ใช้ได้มากขึ้น

9.5 สรุป

ถึงแม้ว่าเทคโนโลยีจิ๋นี่จะเป็นเทคโนโลยีใหม่ แต่ถ้าหากดูถึงรายละเอียดภายในแล้ว เทคโนโลยีที่ใช้นั้นก็เป็นเทคโนโลยีพื้นฐาน มิได้มีอะไรที่แปลกใหม่ แต่จากการนำเทคโนโลยีพื้นฐานต่างๆมาใช้ร่วมกันอย่างเหมาะสม ก็สามารถสร้างเป็นเทคโนโลยีใหม่ที่มีประโยชน์ต่อชีวิตประจำวัน และอาจจะนำไปพัฒนาต่อในเชิงพาณิชย์ได้

นี่จึงเป็นสิ่งที่แสดงให้เห็นว่า เทคโนโลยีที่มีประโยชน์และสามารถประสบความสำเร็จในเชิงพาณิชย์นั้นไม่จำเป็นต้องเป็นเทคโนโลยีใหม่เสมอไป แต่อาจจะเป็นการนำเทคโนโลยีที่มีอยู่แล้วมาประยุกต์ใช้ให้เกิดประโยชน์มากที่สุด เทคโนโลยีจิ๋นี่จะดำเนินต่อไปทางใดนั้น ก็ขึ้นอยู่กับผู้ที่นำไปใช้ ว่าสามารถนำไปพัฒนาต่อและประยุกต์ให้เกิดประโยชน์ได้มากเพียงใด

ภาคผนวก ก.
ความรู้เบื้องต้นเกี่ยวกับ J2ME

ขนาดของสิ่งใด ๆ ก็ตามถ้ามีเพียงขนาดเดียวย่อมไม่เหมาะกับทุกสถานการณ์ เช่นเดียวกับขนาดของ จาวาเทคโนโลยีซึ่งแบ่งออกได้เป็นสามเอดิชั่น เพื่อความเหมาะสมในการใช้งานซึ่งแต่ละเอดิชันจะเหมาะกับตลาดระดับต่าง ๆ กันดังนี้

Java 2 Enterprise Edition สำหรับความต้องการทางด้านเป็นเซิร์ฟเวอร์ซึ่งให้บริการด้านธุรกิจ ทางอินเตอร์เน็ต จะได้ข้อดีของความมั่นคงในการทำงาน ความสมบูรณ์ และขยายขนาดได้ ในการให้บริการกับลูกค้าหรือผู้ขาย

Java 2 Standard Edition สำหรับการสร้างแอพพลิเคชันมาตรฐาน และสำหรับผู้ใช้ทั่วไป

Java2MicroEdition สำหรับอุปกรณ์ขนาดเล็กงานเอมเบ็ดเด็ดซึ่ง เป็นอุปกรณ์ที่มีขนาดของทรัพยากรจำกัด

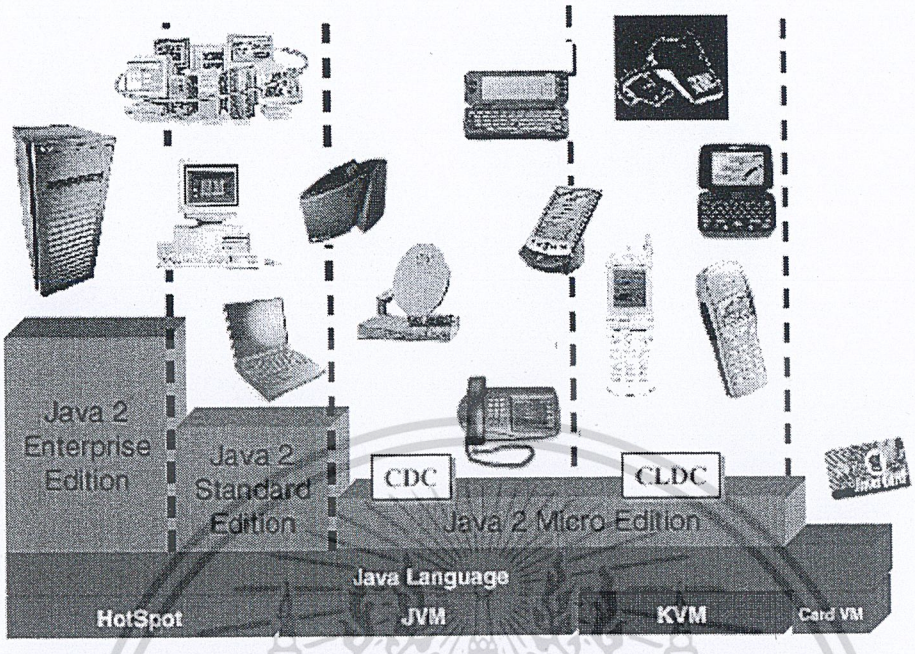
ในแต่ละเอดิชันจะมีเซ็ตของทูลและสิ่งที่จำเป็นต่อการพัฒนาซึ่งจะแตกต่างกันแต่ละเอดิชันดังนี้

- จาวาเวอร์ชวลมาชีน
- ไลบรารี และ เอพีไอ
- ทูลเพื่อการดีพลอย

Java 2 Micro Edition ครอบคลุมการพัฒนาแอพพลิเคชันบนอุปกรณ์ขนาดเล็ก ซึ่งแอพพลิเคชันเหล่านี้มีการพัฒนาเป็นไปอย่างรวดเร็ว เทียบเท่าความก้าวหน้าของตัวอุปกรณ์ จาวาสำหรับอุปกรณ์ขนาดเล็กนั้นสามารถนำมาใช้ในการพัฒนาให้ได้แอปพลิเคชั่น ที่มีความสามารถเทียบเท่าแอปพลิเคชั่นที่พัฒนาด้วยจาวาเอดิชั่นมาตรฐานบนเดสก์ทอปคอมพิวเตอร์ ดังเช่น

- สามารถนำแอปพลิเคชั่นไปรันได้บนอุปกรณ์หลาย ๆ อุปกรณ์
- ใช้ภาษาระดับสูงและการเขียนโปรแกรมเชิงวัตถุ
- โค้ดสามารถพอร์ตไปยังหลายแพลตฟอร์มได้
- มีการจัดการด้านเน็ตเวิร์คที่ปลอดภัย
- สามารถขยายขนาดขึ้นไปหา จาวาเอดิชันมาตรฐาน หรือ เอ็นเตอร์ไพรส์ได้

จาวาเอดิชันขนาดเล็กเหมาะจะพัฒนาแอปพลิเคชั่นที่ มีความขยายตัวแบบไดนามิค สามารถใช้กับเน็ตเวิร์ค เพื่อตลาดของอุปกรณ์เอมเบ็ดเด็ด และนำเอาแอปพลิเคชั่นที่พัฒนาขึ้นมาไปใช้กับอุปกรณ์ได้หลากหลาย



รูปที่ ก-1 แสดงเทคโนโลยีจาวาและกลุ่มเป้าหมาย

จากรูปสามารถแบ่งกลุ่มเป้าหมายของจาวาเอดิชั่นขนาดเล็กได้สองกลุ่มคือ

- อุปกรณ์พวกพกพาส่วนบุคคล เช่น โทรศัพท์มือถือ , เพจเจอร์หรือพวกออแกนไนเซอร์ จะเห็นว่าอุปกรณ์ในกลุ่มนี้ยูสเซอร์อินเทอร์เฟส มีความซับซ้อนน้อยกว่าคอมพิวเตอร์เดสก์ทอป มีขนาดของทรัพยากรน้อยกว่า มีหน่วยความจำประมาณ 128-512 กิโลไบท์ มีแบนด์วิธ ในการสื่อสารน้อย ซึ่งเน็ตเวิร์คของอุปกรณ์กลุ่มนี้ไม่จำเป็นต้องเป็นเน็ตเวิร์ค ทีซีพี/ไอพี
- อุปกรณ์แชร์หรือมีการเชื่อมต่อตายตัว เช่น อินเตอร์เน็ททีวี , โทรศัพท์เห็นภาพผ่านอินเตอร์เน็ต , ระบบการนำทางหรือเพื่อความบันเทิงบนรถ ซึ่งจะมีความซับซ้อนของยูสเซอร์อินเทอร์เฟสมากขึ้น มีทรัพยากรมากขึ้น มีหน่วยความจำประมาณ 2-16 เมกกะไบท์ มีแบนด์วิธ ในการติดต่อสูงกว่า และใช้เน็ตเวิร์คทีซีพี/ไอพี ในการเชื่อมต่อ

J2ME configurations and profiles

อุปกรณ์ที่เป็นกลุ่มเป้าหมายของจาวาเอดิชั่นมาตรฐานต่างก็มีส่วนคล้ายกัน คือสามารถแบ่งแยกออกมาจากอีกอันได้จากฟังก์ชันและฟีเจอร์ในการทำงานซึ่งจะเห็นว่าจาวาเอดิชันขนาดเล็กไม่เพียงจะมีขนาดเล็กแต่ยังสามารถแบ่งเป็นโมดูลและมีความคัสตอมไมซ์

จาวาเอดิชันขนาดเล็กสำหรับโปรดักต์ จะมีคอนฟิกูเรชั่นของเวอร์ชวลมาชีน และจาวาเอพีไอซึ่งจะเลือกเอาเฉพาะที่จำเป็นใช้มาจากจาวาเอพีไอตัวเต็มเพื่อพัฒนาอุปกรณ์ขนาดเล็ก และอาจจะมีจาวาเอพีไอบางตัวเพิ่มเข้าไปเพื่อความเหมาะสมกับในการพัฒนาอุปกรณ์ขนาดเล็ก เป็นส่วนเอพีไอที่นอกจาก จาวาเอพีไอของตัวเต็ม

ความสามารถในการพัฒนาแอปพลิเคชันซึ่งคัสตอมไมซ์ และขยายขนาดได้โดยอิสระมีคอนเซ็ปที่สำคัญอยู่สองประการ

- คอนฟิกูเรชั่น เป็นการกำหนดความต้องการอย่างน้อยในการพัฒนาของแพล็ตฟอร์ม ต่าง ๆ เป็นข้อกำหนดทางด้านแนวนอนซึ่งแต่ละอุปกรณ์จะมีเช่นเดียวกัน ดังเช่น หน่วยความจำที่ต้องใช้งาน พลังของโพรเซสเซอร์ ตัวคอนฟิกูเรชั่น จะกำหนดภาษาจาวา ตัวจาวาเวอร์ชวลมาชีน และไลบรารีอย่างน้อยที่ในการพัฒนา ซึ่งอุปกรณ์ในคลาสเดียวกันซึ่งมาจากหลาย ๆ ผู้ผลิตสามารถใช้ได้เหมือนกัน
- โพรไฟล์ เป็นตัวกำหนดความต้องการแบบเฉพาะเจาะจงของแต่ละอุปกรณ์ ซึ่งเป็นข้อกำหนดทางด้านแนวตั้ง ตัวโพรไฟล์นี้จะต่างกันแต่ละอุปกรณ์เพื่อความเหมาะสม อาจจะเป็นไลบรารี่ที่เฉพาะเจาะจงของแต่ละแพล็ตฟอร์ม เพิ่มเติมขึ้นไปจากคอนฟิกูเรชั่น

คอนฟิกูเรชั่น

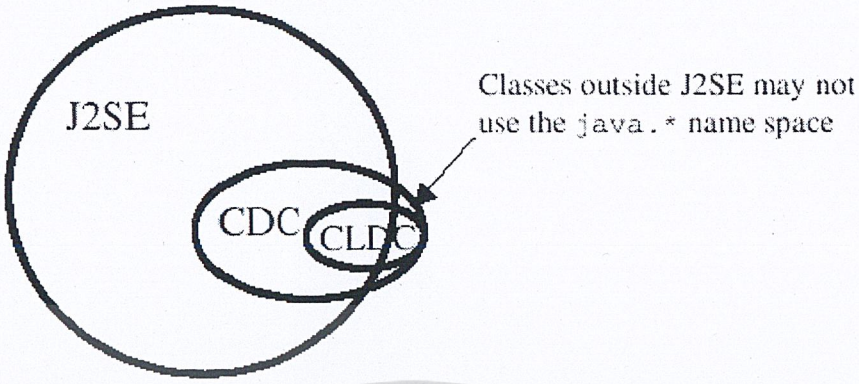
จาวาเอดิชันขนาดเล็กสามารถมีคอนฟิกูเรชันหลาย ๆ แบบ คอนฟิกูเรชันดังที่กล่าวไปแล้วว่าเป็นข้อกำหนดแนวนอน มีสิ่งที่เจาะจงมากกว่านั้นคือ

- ความเจาะจงของฟีเจอร์ภาษาจาวาที่สามารถรองรับได้
- ความเจาะจงของฟีเจอร์จาวาเวอร์ชวลมาชีนที่สามารถรองรับได้
- ความเจาะจงของฟีเจอร์ไลบรารีของจาวาที่สามารถรองรับได้

ในแต่ละคอนฟิกูเรชั่นจะมีฟีเจอร์ของจาวาเวอร์ชวลมาชีนและเซ็ตของเอพีไอที่ผู้ผลิตอุปกรณ์จะยอมรับในใช้ได้กับอุปกรณ์ตัวเอง ผู้พัฒนาแอปพลิเคชั่นต้องทำการออกแบบโค้ดของตนเองให้อยู่ในกรอบของฟีเจอร์ที่จำกัดของจาวาเวอร์ชวลมาชีนและเอพีไอที่กำหนดในคอนฟิกูเรชั่นนั้นๆ

คอนฟิกูเรชั่นของจาวาเอดิชั่นขนาดเล็กมีด้วยกันสองแบบคือ

- Connected, Limited Device Configuration (CLDC). เป็นคอนฟิกูเรชั่นที่ใช้กับอุปกรณ์พวกโทรศัพท์มือถือ หรือพวกที่พกติดตามตัว คอนฟิกูเรชั่นนี้รวมไปถึงคลาสของไลบรารีที่ออกแบบมาให้เหมาะกับอุปกรณ์ขนาดเล็กซึ่งไม่ได้มาจาก จาวาเอดิชั่น มาตรฐาน
- Connected, Device Configuration (CDC). เป็นคอนฟิกูเรชั่นที่ใช้กับอุปกรณ์พวกที่แชร์หรือมีการเชื่อต่อตายตัว ซึ่งจะเป็นซูเปอร์เซ็ตของ CLDC



รูปที่ ก-2 แสดงความสัมพันธ์ของจาวาเอดิชันขนาดเล็กและเอดิชั่นมาตรฐาน

จากรูปจะเห็นว่า CLDC จะเป็นซับเซ็ตของ CDC และทั้งคู่จะอินเฮอริท จากจาวาเอดิชั่นมาตรฐาน แต่ก็จะมีเอพีไอที่ไม่มีใน เอดิชั่นมาตรฐานซึ่งได้ออกแบบมาเฉพาะสำหรับอุปกรณ์ขนาดเล็ก ซึ่งรายละเอียดเต็ม ๆ สามารถศึกษาได้จาก Configurations and Profiles Architecture Specification, Java™ 2 Platform Micro Edition (J2ME), Sun Microsystems, Inc.

โพรไฟล์

แอปพลิเคชั่นทีสามารถพอร์ทได้นั้นถือเป็นสิ่งที่เป็นผลกำไรจากการพัฒนา แต่อย่างไรก็ตาม อุปกรณ์แต่ละผู้ผลิต มีความแตกต่างกันมากทำให้เป็นการยากยิ่งในการรองรับการสร้างแอปพลิเคชั่นบนอุปกรณ์หลาย ๆ ความแตกต่างด้วยทางออกเพียงทางเดียว

โดยทั่วไปผู้บริโภคไม่ได้ต้องการแอปพลิเคชั่นซึ่งเป็นอันเดียวกันหมดทั้งหมดอันเกิดมากจากการพอร์ท แต่จะเป็นไปแค่อุปกรณ์ชนิดเดียวกันเท่านั้นซึ่งต้องการให้เกิดการพอร์ทได้ 100% ดังนั้นการที่จะทำให้เกิดความเหมาะสมนี้ต้องมีการกำหนด จาวาแพล็ตฟอร์ม ในทางแนวตั้ง ก็คือจะมีคอนเซ็ปของโพรไฟล์เข้ามาเกี่ยวข้อง ทำให้เกิดแบ่งส่วนของอุปกรณ์ซึ่งแต่ละส่วนจะพอร์ทแอปพลิเคชั่นถึงกันได้ ตัวโพรไฟล์สามารถใช้กับตัวความต้องการสองอันโดย

- โพรไฟล์จะมีชุดเครื่องมือเพื่อนำแอปพลิเคชั่นไปใช้ในอุปกรณ์ชนิดต่าง ๆ กัน
- โพรไฟล์อาจจะถูกสร้างขึ้นมาเพื่อรองรับกลุ่มของแอปพลิเคชั่นซึ่งใช้อยู่บนอุปกรณ์ต่าง ๆ ชนิดกัน

ในระดับของการอิมพลีเมนท์ โพรไฟล์จะเป็นกลุ่มของจาวาเอพีไอและคลาสไลบรารีซึ่งอยู่บนคอนฟิกูเรชั่นที่เจาะจงอีกชั้นหนึ่ง เป็นการเพิ่มความสามารถของอุปกรณ์ซึ่งจะไม่เหมือนกันในแต่ละกลุ่มของตลาด

ความรู้เบื้องต้นเกี่ยวกับ KVM

CLDC นั้นใช้เวอร์ชวลมาชีนที่ชื่อว่า K Virtual Machine (KVM) เป็นเวอร์ชวลมาชีนที่มีขนาดเล็ก เป็นเวอร์ชวลมาชีนที่พอร์ทมาจากเอดิชั่นมาตรฐานเหมาะกับอุปกรณ์ที่มีทรัพยากรจำกัด จุดมุ่งหมายของเวอร์ชวลมาชีนตัวนี้คือ มีความสามารถเท่าเทียมกับเอดิชั่นมารตฐาน ซึ่งสามารถรองรับฟีเจอร์ต่าง ๆ ของภาษาจาวา ได้เหมือนกับเอดิชั่นมารตฐาน แต่ทำงานในอุปกรณ์ที่มีทรัพยากรจำกัด

ลักษณะเฉพาะของ KVM

- มีขนาดเล็ก ใช้หน่วยความจำน้อยในการทำงาน อยู่ในช่วงประมาณ 40 – 80 กิโลไบท์
- ตัวโคดสามารถพอร์ทได้อย่างง่ายดาย
- มีลักษณะเป็นโมดูล และคัสตอมไมซ์
- มีความสมบูรณ์ และความเร็วเท่าที่เป็นไปได้โดยไม่เสียข้อดีอื่นๆ

ตัว K ในชื่อ KVM หมายถึง กิโล KVM เหมาะกับโพรเซสเซอร์ ทั้ง 16 หรือ 32 บิท และ CISC หรือ RISC ซึ่งมีขนาดหน่วยความนำรวมแล้วไม่เกิน 128 กิโลไบท์ โดยทั่วไปอุปกรณ์พวกนี้คือ โทรศัพท์มือถือ เพจเจอร์ ออร์แกไนเซอร์ส่วนตัว เป็นต้น

หน่วยความจำที่จำเป็นใช้สำหรับ KVM อย่างน้อยประมาณ 128 กิโลไบท์ จะรวมตัวเวอร์ชวลมาชีน จาวาไลบรารี และหน่วยความจำฮีปเพื่อการรันแอปพลิเคชั่น การอิมพลีเมนท์การใช้งานของ KVM อื่น ๆ เช่น มีหน่วยความจำให้ใช้ทั้งหมด 256 กิโลไบท์ แบ่งครึ่งหนึ่งเป็นหน่วยความจำฮีป 40-80 กิโลไบท์ แบ่งให้กับตัวเวอร์ชวลมาชีนและที่เหลือเป็นส่วนของไลบรารีต่าง ๆ อัตราส่วนของหน่วยความจำแบบรอมกับแรมก็ขึ้นอยู่กับการอิมพลีเมนท์ คือ KVM ที่ไม่มี คลาสซึ่งยังไม่ได้ลิงค์เพื่อทำการเอ็กซีคิวท์ จะต้องการแรมมากว่า KVM ที่อิมพลีเมนท์ด้วยคลาสพรีโหลด ซึ่งได้โหลดคลาสที่จำเป็นใช้มาเก็บในอุปกรณ์แล้ว

สามารถหาข้อมูลเพิ่มเติมซึ่งมากกว่านี้ได้จาก

http://java.sun.com/products/kvm

แปลและเรียบเรียงมาจากเอกสารของชุดพัฒนา J2ME_CLDC ในภาคผนวกที่ 1 Introduction to Java 2 Micro Edition and KVM ของ Sun Microsystem,Inc.

ภาคผนวก ข.
การเขียนโปรแกรมสำหรับปาล์ม

การพัฒนาโปรแกรมสำหรับปาล์มด้วยจาวา

เตรียมอุปกรณ์ในการพัฒนา

- สำหรับการทดสอบในโครงงานนี้ได้ใช้โปรแกรมปาล์มอีมูเลเตอร์และรอม สำหรับเครื่องปาล์ม ซึ่งสามารถดาวน์โหลดได้จาก http://www.palmdino.com
- ดาวน์โหลดไฟล์ "j2me-cldc-1_0-src-winsol.zip" และ "j2me-cldc-1_0-src-palm-overlay.zip" มา ซึ่งไฟล์แรกนั้นประกอบไปด้วย J2ME API, K Virtual Machine (KVM), และ Tools ต่างๆเช่น JAM, JCC ซึ่ง KVM ในไฟล์แรกนี้เป็น KVM ซึ่งเป็น Common KVM หมายถึง KVM ที่ยังไม่ได้พอร์ตไปยังอุปกรณ์ใดส่วนไฟล์ที่สองนั้น ประกอบด้วย KVM ซึ่งทำพอร์ตไปยังปาล์มแล้ว พร้อมทั้งมีตัวอย่างโปรแกรมอีกจำนวนหนึ่งด้วย
- ซึ่ง KVM และตัวอย่างโปรแกรมใน ไฟล์ ที่สองนั้น ไฟล์จะอยู่ในรูปของ .prc ซึ่งเป็น execute file ใน ฟอร์แมตของ Palm

การโหลด KVM ลงบนปาล์ม

- จะต้องใช้ไฟล์ 2 ไฟล์ในการโหลด KVM ลงในปาล์มคือ "KVM.prc" และ "KVMutil.prc" ซึ่งวิธีการทำงานของสองไฟล์นี้คือไฟล์แรกจะเป็น KVM ส่วนไฟล์ที่สองเป็นชุด API ของ CLDC
- เมื่อโหลดไฟล์ทั้งสองลงไปแล้วก็สามารถโหลดคลาสไฟล์ของภาษาจาวาซึ่งแปลงให้อยู่ในรูปของ .prc ลงไปยังปาล์ม ได้เลย
- เมื่อสั่งให้จาวาแอพพลิเคชันเริ่มทำงาน แอพพลิเคชันนั้นก็จะเรียกใช้ KVM ขึ้นมาทำงานเองโดยอัตโนมัติ

ขั้นตอนในการคอมไพล์และสร้างโปรแกรมสำหรับปาล์ม

การเขียนแอพพลิเคชันสำหรับปาล์มด้วยภาษา จาวาซึ่งนำไปรันบน KVM สำหรับปาล์มโดยทำการทดสอบด้วย Palm OS Emulator ในขั้นตอนการเขียนโปรแกรม และการคอมไพล์โปรแกรม มีขั้นตอนการทำงานดังนี้

1. คอมไพล์ทูลส์สำหรับใช้ในการสร้างไฟล์ .prc โดยเข้าไปที่ไดเร็คทอรี่ที่ลงชุดพัฒนา J2ME ไว้

   c:\j2me_cldc\tools\palm\src\palm\database แล้ว Compile ด้วยคำสั่ง

   javac -d c:\j2me_cldc\bin\api\classes\ *.java

   ก๊อบปี้ไฟล์ Wrapper.prc และ DefaultTiny.bmp จาก

c:\j2me_cldc\tools\palm\src\palm\database ไปยัง

c:\j2me_cldc\bin\api\classes\palm\database

2. ทำการคอมไพล์ไฟล์ภาษา Java ที่เขียนขึ้น ด้วย java compiler (javac) มาตรฐานจากชุด J2SE

Example : javac –g:none –d tmp –classpath ../lib/classes -bootclasspath ../lib/classes HelloWorld.java

Flag –bootclasspath เป็นการบังคับให้ javac ใช้ API Class เฉพาะของ j2me ก่อนโดยไม่ใช้ของ j2se ซึ่งเป็น Default และ ผลของการ Compile จะเก็บอยู่ใน directory tmp

3. ทำการ preverify คลาสของภาษาจาวาซึ่งได้มาจากข้อ 1 ซึ่ง preverify เป็น Tool ที่มากับชุดพัฒนา j2me_cldc

Example : preverify –d classes –classpath ../lib/classes tmp

ซึ่งผลลัพธ์ของการ preverify จะเก็บอยู่ใน ไดเรคทอรี่ classes

4. นำคลาสทั้งหมดที่ คอมไพล์ได้ (หากมีหลายคลาส) มารวมกันสร้างเป็น jar file

Example : jar HelloWorld.class MyTextBox.class ..more class..

5. นำ jar file มาแปลงเป็นไฟล์สำหรับปาล์ม (.prc) โดย Tools MakePalmApp ซึ่งมากับชุดพัฒนา j2me_cldc palm overlay

Example : java –classpath ../lib/classes palm.database.MakePalmApp –v –version "1.0" -icon test.bmp –bootclasspath ../lib/classes classes Helloworld

ความรู้พื้นฐานในการเขียนโปรแกรม

แอปพลิเคชั่นสำหรับปาล์มโดยปรกตินั้นจะอินเฮอริท มาจากคลาสที่ชื่อว่า Spotlet ซึ่งอยู่ในแพคเกจ com.sun.kjava ในชุด API ของ j2me ซึ่งคลาสนี้จะช่วยในการดักอีเวนต์และทำหน้าที่เป็น หน้าจอหลักของแอพพลิเคชัน โดยที่แอพพลิเคชันตัวหนึ่งนั้นสามารถสร้างออบเจ็กต์ของคลาส Spotlet ได้หลายออบเจ็กต์ แต่ว่าในเวลาหนึ่งนั้น จะมีหน้าจอที่ถูกโฟกัสได้เพียงหน้าจอเดียวเท่านั้น

ออบเจ็กต์ของ Spotlet เป็นออบเจ็กต์ที่เป็นกราฟฟิก แต่แตกต่างจากออบเจ็กต์ของ AWT หรือ Swing ตรงที่ออบเจ็กต์ของ Spotlet นั้นไม่มีความสามารถสำหรับงานในระดับสูง เช่น ไม่มี ตัวจัดการเลย์เอ้าท์ (Layout Manager) ไม่มีตัวจัดการการวาดหน้าจอใหม่ (Repaint Manager) ไม่มีระบบจัดการEvent ที่แท้จริง

Simple Spotlet

โปรแกรมด้านล่างนี้เป็นตัวอย่างการสร้างแอปพลิเคชั่นของ Spotlet โดยยังไม่มีระบบรองรับอีเวนต์ซึ่งทำให้ยังไม่สามารถทำได้แม้กระทั่งการปิดโปรแกรม

```
import com.sun.kjava.*;

/** Most basic of all examples, we extend a Spotlet, and merely draw
 *  some text on the screen.  Note that there is no event handling of
 *  any kind, including exiting the Spotlet.
 */
public class SimpleSpotlet extends Spotlet {

    // handle on the singleton Graphics object
    static Graphics g = Graphics.getGraphics();

    /**
     * The main method simply creates a Scribble spotlet and
     * registers its event handlers.
     */
    public static void main(String[] args) {
        SimpleSpotlet testSpotlet = new SimpleSpotlet();
       testSpotlet.register(NO_EVENT_OPTIONS);
    }

    /**
     * Default constructor creates the GUI components and draws them.
     */
    public SimpleSpotlet() {
       // paint the text
       paint();
    }

    /**
     * Draw the screen
     */
    private void paint() {
       g.clearScreen();
       g.drawString("Simple Spotlet",60, 80);
    }
}
```

- คลาสนี้อินเฮอริแตนซ์มาจากคลาส Spotlet ซึ่งจะมีกระบวนการคอลแบ็ค สำหรับรองรับ อีเวนต์โดยที่เมื่อมี อีเวนต์เกิดขึ้นเมธอดที่มีชื่อเฉพาะจะถูกเรียกใช้โดยอัตโนมัติ เช่น เมื่อมีการกดสไตลัส ลงบนหน้าจอ เมธอด penDown(int x,int y) ก็จะถูกเรียก โดยส่งค่าตำแหน่งที่เกิดการกดมาเป็นพารามิเตอร์
- เพื่อให้มีการโฟกัสมาที่หน้าจอของ Spotlet ออบเจ็กต์นั้น Spotlet จะต้องเรียกใช้ method register() ซึ่งจะทำให้มีการโฟกัสมายังออบเจ็กต์ที่เรียกใช้ method
- การเขียนตัวอักษร “Hello World” ลงไปบนหน้าจอ ทำโดยการโอเวอร์ไรด์ method paint() ของ Spotlet

```
public SimpleSpotlet() {
    paint();
}

private void paint() {
    g.clearScreen();
```

```
        g.drawString("Simple Spotlet",60, 80);
    }
```

- จากโค้ดข้างต้นจะเห็นว่า เมื่อเริ่มต้น ต้องมีการเรียกใช้ method paint() ใน Constructor
- method paint() นั้นไม่จำเป็นต้อง ส่งพารามิเตอร์เป็น Graphics Context เหมือนกับใน AWT แต่ใช้ Graphic Context เดียวตลอดทั้งโปรแกรม ซึ่งได้มาจาก การเรียกใช้ method ของคลาส Graphics.getGraphics()
- การเคลียร์หน้าจอจะต้องทำด้วยตัวเอง ระหว่างอยู่ในกระบวนการของ method paint()

การดักฟัง Event

การลงทะเบียนขอรับ Event ของออบเจ็กต์ของ Spotlet เราสามารถเรียกใช้ method register() ของ ออบเจ็กต์ของ Spotlet เพื่อลงทะเบียนรับ Event โดยกระบวนการของ method นี้คือ จะทำให้ออบเจ็กต์ Spotlet ที่ถูกเรียกใช้ method นี้นั้น ถูกโฟกัสเพื่อรอรับ Event ทั้งหมดและออบเจ็กต์ที่ได้ลงทะเบียนไว้ก่อนหน้านี้จะถูกยกเลิก วิธีการที่ Spotlet ใช้ในการรับ Event คือวิธีการคอลแบ็ค โดยการทำการโอเวอร์ไรด์ method ที่มีชื่อเฉพาะ ที่จะถูกเรียกโดยอัตโนมัติ (คอลแบ็ค) เมื่อเกิด Event ต่างๆ

จำนวน Event ที่รับได้ขึ้นอยู่กับค่าพารามิเตอร์ที่ส่งให้กับ method register() ซึ่งเป็นได้สองกรณี คือ NO_EVENT_OPTIONS และ WANT_SYSTEM_KEY

ถ้ามีการเรียกใช้ method register(WANT_SYSTEM_KEY) จะทำให้ Spotlet ออบเจ็กต์นี้นั้นสามารถรับ Event ทั้งหมดได้ และรวมถึง System key ซึ่งเป็นปุ่มที่อยู่ที่เครื่องปาล์มด้วย Event ที่รับได้คือ

- Pen Down events
- Pen Up events
- Pen Move events
- การเขียนตัวอักษรด้วย กราฟฟิตี้
- การกดปุ่มต่างๆบนเครื่องปาล์ม (Hard Key)
- การกดปุ่ม 4 ปุ่มรอบๆเขตของกราฟฟิตี้ หรือเขต 123 หรือ abc ภายในเขตของกราฟฟิตี้
- การเคาะปุ่ม Menu

ส่วนถ้ามีการลงทะเบียนด้วย method register(NO_EVENT_OPTION) ก็จะได้รับเพียง Event เหล่านี้เท่านั้น

- Pen Down events
- Pen Up events
- Pen Move events
- การเขียนตัวอักษรด้วย กราฟฟิตี้
- การเคาะปุ่ม Menu

ด้านล่างนี้เป็นตัวอย่างโปรแกรมที่ใช้จัดการกับ Event ต่างๆ

```
import com.sun.kjava.*;

/** An example showing event handling options for your basic Spotlet.
 * @author Robert Evans, rbevans@akane.jhuapl.edu
 */
public class EventHandling extends Spotlet {

    // First some header strings to be used later on
    final String eventHeader = "Event:  ";
    final String locationHeader = "Located at ";
    final String keyHeader = "Key pressed was [";

    // we will have one line of text showing the event
    String eventLine = eventHeader + "<None>";
    // and one line of text showing info about the event
    String infoLine = locationHeader + "<N/A>";
    // I use a flag to keep track if we are continuing a penMove
    // event...this lets me update the graphics quicker
    boolean penMoving = false;

    // Where I start the event and info lines
    int leftMargin = 15;
    // spacing between the two lines
    int line = 15;
    // the Y-coord of the first (eventLine) line
    int firstLine = 60;

    // The EXIT GUI button
    Button exitButton;

    // handle on the singleton Graphics object
    static Graphics g = Graphics.getGraphics();

    /**
     * The main method simply creates a Scribble spotlet and
     * registers its event handlers.
     */
    public static void main(String[] args) {
       EventHandling eh = new EventHandling();
       eh.register(WANT_SYSTEM_KEYS);
//     eh.register(NO_EVENT_OPTIONS);
    }

    /**
     * Default constructor creates the GUI components and draws them.
     */
    public EventHandling() {
       exitButton = new Button("Exit",139,145);
       // paint the screen
       paint();
    }

    /**
     * Draw the screen
     */
    private void paint() {
       g.clearScreen();
       g.drawString(" EventHandling Demo ", 34, 0, Graphics.INVERT);
       g.drawString(eventLine, leftMargin, firstLine);
       g.drawString(infoLine, leftMargin, firstLine + line);
       exitButton.paint();
    }
    /**
```

```
 * Handle a pen down event. If it is in our exit button, we will
 * exit the application, otherwise we will print the data to the screen
 */
public void penDown(int x, int y) {
   if (exitButton.pressed(x,y)) {
       // The following is used with the KVM_EA1 version
       System.exit(0);
       // The following is used with the KVM_J1 version
       // Runtime.exit();
   } else {
       eventLine = eventHeader + "penDown";
       infoLine = locationHeader + "(" + x + "," + y + ")";
       paint();
   }
}

/**
 * Handle a pen up event.
 */
public void penUp(int x, int y) {
   penMoving = false;
   eventLine = eventHeader + "penUp";
   infoLine = locationHeader + "(" + x + "," + y + ")";
   paint();
}

/**
 * Handle a pen move event.
 */
public void penMove(int x, int y) {
   eventLine = eventHeader + "penMove";
   // if the pen is still moving, we only need to update the
   // location line
   if (penMoving) {
       // note the trailing spaces, this "clears" extra
       // characters from the last write operation.  Not elegant,
       // but it works, and it is fast and flicker free...
       infoLine = locationHeader + "(" + x + "," + y + ")    ";
       g.drawString(infoLine, leftMargin, firstLine + line);
   } else {
       // if this is a new event, we want a full paint/clear cycle
       infoLine = locationHeader + "(" + x + "," + y + ")";
       paint();
       penMoving = true;
   }

}

/**
 * Handle a key down event. A keyDown event *may* involve
 *  - Text Entry using Graffiti
 *  - Pushing the hard buttons at the bottom of the unit
 *  - Touching the four icons to the sides of the Graffiti area
 *    or touching the "abc" or "123" areas within the Graffiti area
 *
 * If the Spotlet was registered with
 * WANT_SYSTEM_KEYS, all three of the above key down
 * events are trapped and handled by the application.
 *
 * If the Spotlet was registered with the
 * NO_EVENT_OPTIONS, the only keys the application
 * will react to are:
 *
 *  - Text Entry using Graffiti
 *  - Tapping the Menu Icon
 *
 * All other keys will perform their normal PalmOS function, with
 * the exception of the "abc" and "123" areas in the Graffiti area,
```

```
     * which now have no effect whatsoever.
     */
    public void keyDown(int keyCode) {
       eventLine = eventHeader + "keyDown";
       switch(keyCode) {
       case KEY_HARD1:
           infoLine = keyHeader + "Hard Key 1]";
           break;
       case KEY_HARD2:
           infoLine = keyHeader + "Hard Key 2]";
           break;
       case KEY_HARD3:
           infoLine = keyHeader + "Hard Key 3]";
           break;
       case KEY_HARD4:
           infoLine = keyHeader + "Hard Key 4]";
           break;
       case CALCICON:
           infoLine = keyHeader + "Calculator Icon]";
           break;
       case PAGEDOWN:
           infoLine = keyHeader + "Page Down Key]";
           break;
       case PAGEUP:
           infoLine = keyHeader + "Page Up Key]";
           break;
       case MENUICON:
           infoLine = keyHeader + "Menu Icon]";
           break;
       case 264:
           infoLine = keyHeader + "Home Icon]";
           break;
       case 266:
           infoLine = keyHeader + "Find Icon]";
           break;
       case 272:
           infoLine = keyHeader + "abc Area]";
           break;
       case 273:
           infoLine = keyHeader + "123 Area]";
           break;
       default:
           char tmpChar = (char)keyCode;
           infoLine = keyHeader + tmpChar + "]";
       }
       paint();
    }

}
```

จากหัวข้อที่กล่าวมาแล้วคงจะพอเป็นแนวทางในการพัฒนาโปรแกรมสำหรับปาล์มด้วยภาษาจาวาได้ ซึ่งที่กล่าวมานั้นยังเป็นเรื่องพื้นฐาน ยังขาดเรื่องสำคัญอีกคือการใช้ แพคเกจ com.sun.kjava ซึ่งเป็นการสร้างกราฟฟิกยูสเซอร์อินเตอร์เฟส รายละเอียดในเรื่องนี้สามารถศึกษาเพิ่มเติมได้ที่ http://webdev.apl.jhu.edu/~rbe/kvm/

บรรณานุกรม

เอกสารอ้างอิง

[1] Scott Oaks, Henry Wong : *"Jini in a Nutshell "*, O'Reilly & Associates, Inc. United State of America. 2000.

[2] ดร.วีระศักดิ์ ซึงถาวร : *"Fundamental of JAVA Programming Volume 1"*, Sum Publishing Department, Sum System Company Limited. 1998.

[3] ดร.วีระศักดิ์ ซึงถาวร : *"Fundamental of JAVA Programming Volume 2"*, Se-Education Public Company Limited. 2000.

[4] Bruce Powel Douglass: *"Real-Time UML: Developing Efficient Objects for Embedded Systems"*, Addison-Wesley Longman Inc., Massachusetts, 1998.

[5] Qusay Mahmoud :*" Setting up the KVM on your Palm and Getting Started with the KVM "* , Java.sun.com articles, Sun microsystem Inc, November, 2000

เว็บไซท์อ้างอิง

[1] http://www.sun.com/jini

[2] http://www.java.sun.com/j2me

[3] http://www.cswl.com/whiteppr/tutorials/jini.html

[4] http://webdev.apl.jhu.edu/~rbe/kvm/index.html

[5] http://www.kvmworld.com

[6] http://www.javacentrix.com

[7] http://www.8052.com

[8] http://www.national.com