ระบบจัดการพลังงานสำหรับไมโครกริด

MICROGRID ENERGY MANAGEMANT SYSTEM

โดย

นางสาว ยลดา มาศวิจิตรวงศ์
นาย วรพจน์ โรจน์รัตนาวิชัย
นาย วรพจน์ ฮุนสวัสดิกุล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต
สาขาวิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2553

ปีการศึกษา 2553

ระบบจัดการพลังงานสำหรับไมโครกริด

MICROGRID ENERGY MANAGEMANT SYSTEM

โดย

นางสาว ยลดา มาศวิจิตรวงศ์

นาย วรพจน์ โรจน์รัตนาวิชัย

นาย วรพจน์ ฮุนสวัสดิกุล

อาจารย์ที่ปรึกษา

ดร. สมภพ ผลไม้

ปริญญานิพนธ์ปีการศึกษา 2553

สาขาวิชาวิศวกรรมไฟฟ้า

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบจัดการพลังงานสำหรับไมโครกริด

ผู้จัดทำ

1. นางสาว ยลดา มาศวิจิตรวงศ์
2. นาย วรพจน์ โรจน์รัตนาวิชัย
3. นาย วรพจน์ ฮุนสวัสดิกุล

..... อาจารย์ที่ปรึกษา

(ดร. สมภพ ผลไม้)

ระบบจัดการพลังงานสำหรับไมโครกริด

นางสาว ยลดา มาศวิจิตรวงศ์
นาย วรพจน์ โรจน์รัตนาวิชัย
นาย วรพจน์ ฮุนสวัสดิกุล
ดร.สมภพ ผลไม้ อาจารย์ที่ปรึกษา
ปีการศึกษา 2553

บทคัดย่อ

ปริญญานิพนธ์นี้ได้นำเสนอการออกแบบระบบการจัดการพลังงานสำหรับไมโครกริด โดยมีวัตถุประสงค์ในการจัดการพลังงานคือ รักษาสมดุลระหว่างความต้องการกับการผลิตกำลังไฟฟ้า ในปริญญานิพนธ์นี้ได้นำเทคโนโลยีมัลติเอเจนท์มาประยุกต์ใช้สำหรับการควบคุมไมโครกริดในลักษณะที่ไม่เป็นศูนย์กลางโดยใช้ชุดพัฒนาจาวาเอเจนท์ สำหรับไมโครกริดในกรณีศึกษาประกอบด้วยเครื่องกำเนิดไฟฟ้า 3 ตัว และมีภาระไฟฟ้า มีเอเจนท์อยู่ 4 ประเภท ได้แก่ เอเจนท์ของโหลด เอเจนท์ของเครื่องกำเนิดไฟฟ้า เอเจนท์ฐานข้อมูล และเอเจนท์จำลองระบบ จากการจำลองระบบไมโครกริดตามปริมาณความต้องการภาระไฟฟ้าภายใต้สถานการณ์การผลิตที่แตกต่างกันซึ่งได้ถูกออกแบบไว้แล้วนั้น ผลลัพธ์ที่ออกมาแสดงให้เห็นว่าสามารถนำเทคโนโลยีมัลติเอเจนท์มาใช้กับไมโครกริดในการประสานการทำงานสำหรับการควบคุมในลักษณะที่ไม่เป็นศูนย์กลาง

MICROGRID ENERGY MANAGEMENT SYSTEM

Yonlada Masvijitwong
Vorapoj Rojrattanawichai
Woraphot Hunsawatdikun
Dr. Sompob Polmai Supervisor
2010

ABSTRACT

This thesis presents design of energy management system for a microgrid. The purpose of this energy management is to balance the load demand with the generation supply. In this thesis multi-agent technology is adopted for decentralized control of the microgrid using Java Agent DEvelopment Framework (JADE). A simple microgrid consisting of three generators and load is modeled in this study. Four types of agents are defined including load agent, generator agent, database agent and simulation agent. The simulations of the microgrid under pre-defined load profile and generator scenarios are carried out. The simulation results show that integration multi-agent technology with microgrid was successful in co-operation for decentralized control.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้เสร็จสมบูรณ์ไปได้ด้วยดี ในนามของผู้เขียนต้องขอกราบขอบพระคุณบุคคลดังต่อไปนี้

- ขอขอบพระคุณ ดร.สมภพ ผลไม้ ผู้ซึ่งเป็นอาจารย์ที่ปรึกษาที่กรุณาให้คำปรึกษา รวมทั้งแนะนำโครงงานและให้ความช่วยเหลือในทุกๆ ด้าน ตลอดจนอาจารย์ในสาขาวิชาที่ประสิทธิประสาทความรู้ให้กับผู้เขียนในครั้งนี้
- ขอขอบพระคุณสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่คอยส่งเสริมสนับสนุนการศึกษาและการวิจัยของนักศึกษา
- ขอขอบคุณสมาชิกห้องวิจัย PEARL LAB ที่คอยช่วยเหลือด้านต่างๆ รวมทั้งเป็นขวัญและกำลังใจให้กับผู้เขียน
- สุดท้ายนี้ต้องขอขอบพระคุณบุคคลที่สำคัญที่สุดที่ทำให้ข้าพเจ้ามีวันนี้ ก็คือบิดามารดา อันเป็นที่รักยิ่ง ซึ่งได้อบรมเลี้ยงดูข้าพเจ้าเป็นอย่างดี พร้อมทั้งให้โอกาสในการศึกษาอย่างเต็มที่ คอยให้กำลังใจ และคอยเอาใจใส่ในทุกๆ ด้านเสมอ

ข้าพเจ้าขอระลึกคุณอย่างสุดซึ้งและขอกราบขอบพระคุณมา ณ ที่นี้

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญตาราง

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญ

การผลิตไฟฟ้าในประเทศไทยเป็นการผลิตแบบรวมศูนย์ โดยใช้เชื้อเพลิงฟอสซิล เช่น ถ่านหินและก๊าซธรรมชาติเป็นหลัก ถ้าไม่คำนึงถึงข้อเสียด้านมลภาวะจากเชื้อเพลิงฟอสซิล ข้อดีสำคัญของการผลิตไฟฟ้าแบบรวมศูนย์ จะเป็นการผลิตไฟฟ้าปริมาณมากจากโรงไฟฟ้าขนาดใหญ่ จะเป็นการผลิตไฟฟ้าปริมาณมากจากโรงไฟฟ้าขนาดใหญ่ ทาให้มีความคุ้มทุนเชิงเศรษฐศาสตร์และมีเสถียรภาพการผลิตไฟฟ้าสูง

การผลิตและจ่ายไฟฟ้าแบบรวมศูนย์ ซึ่งเป็นการจ่ายไฟฟ้าแบบทางเดียวจากผู้ผลิตสู่ผู้บริโภคนั้นปริมาณไฟฟ้าที่จะผลิตจะต้องสัมพันธ์กับภาระความต้องการไฟฟ้าของผู้ใช้ ซึ่งจะมีความหลากหลายทั้งในแง่ของช่วงเวลาของความต้องการไฟฟ้า หรือตำแหน่งที่อยู่ของผู้ใช้ซึ่งกระจัดกระจายอยู่ทั่วประเทศ ดังนั้น ข้อเสียของการจ่ายไฟฟ้าผ่านเครือข่ายจากโรงจักรไฟฟ้าขนาดใหญ่ คือการสูญเสียพลังงานตามสายส่ง ในกรณีที่ผู้บริโภคอยู่ไกลจากโรงไฟฟ้าและเนื่องจากเป็นระบบผลิตขนาดใหญ่ จึงมีปัญหาด้านอัตราการตอบสนองต่อความต้องการกำลังไฟฟ้าที่เปลี่ยนแปลงของผู้ใช้ ซึ่งโดยปกติแล้วจะมีลักษณะหลากหลายและบ่อยครั้งมักมีการเปลี่ยนแปลงอย่างรวดเร็วในบางช่วงเวลา ระบบผลิตไฟฟ้าขนาดใหญ่ไม่สามารถกักเก็บพลังงานไฟฟ้าได้ ดังนั้น ระบบส่งไฟฟ้าขนาดใหญ่ จึงไม่มีการบริหารจัดการด้านพลังงานสะสมเพื่อรองรับ หรือชดเชยอัตราการการเปลี่ยนแปลงภาระทางไฟฟ้าของผู้ใช้ และข้อเสียประการสุดท้ายที่เห็นได้ชัดคือข้อจำกัดด้านประสิทธิภาพการผลิตไฟฟ้าจากโรงไฟฟ้าขนาดใหญ่จะไม่สูงนัก เนื่องจากพลังงานส่วนใหญ่ของเชื้อเพลิงที่ใช้ในการผลิตไฟฟ้า จะเปลี่ยนรูปไปเป็นความร้อนทิ้งปล่อยออกสู่บรรยากาศ โดยไม่ได้นำกลับมาใช้แต่อย่างใด

ในปัจจุบันเทคโนโลยีการผลิตไฟฟ้าได้พัฒนาจนมีขนาดของระบบเล็กมาก และราคาก็ต่ำลงกว่าในอดีต เช่น กังหันก๊าซขนาดเล็ก หรือเครื่องยนต์สเตอร์ริงซึ่งสามารถเลือกใช้เชื้อเพลิงได้หลากหลายในการผลิตไฟฟ้า และเนื่องจากระบบมีขนาดเล็กมาก ทำให้มีผู้ใช้มีอิสระที่จะเลือกตำแหน่งการติดตั้งระบบผลิตไฟฟ้าที่ใกล้กับภาระความร้อน และสามารถนำความร้อนทิ้งจากระบบผลิตไฟฟ้าขนาดเล็กมากนี้ไปใช้งานได้โดยตรง ทำให้สามารถใช้พลังงานจากแหล่งเชื้อเพลิงได้อย่างมีประสิทธิภาพ คือ ใช้ทั้งผลิตไฟฟ้าและนำความร้อนทิ้งกลับไปใช้ประโยชน์ด้วย

ด้วยการพัฒนาด้านเทคโนโลยีด้านอิเล็กทรอนิกส์กำลัง (Power Electronics) ทำให้เทคโนโลยีการผลิตไฟฟ้าขนาดเล็กมาก เปิดโอกาสในการนาเทคโนโลยีพลังงานหมุนเวียน ซึ่งมักจะเป็นระบบผลิตไฟฟ้ากระแสตรงและมีลักษณะการผลิตที่ไม่สม่ำเสมอขึ้นกับแหล่งกำเนิดพลังงาน ให้สามารถเชื่อมต่อกับระบบกริดไฟฟ้าแบบกระแสสลับได้ การผลิตไฟฟ้าด้วยเทคโนโลยีที่ใช้พลังงานหมุนเวียนมีข้อได้เปรียบเหนือการผลิตไฟฟ้าที่ใช้เชื้อเพลิงฟอสซิลในเชิง

การลดสภาวะโลกร้อนและการปลดปล่อยก๊าซมลภาวะ เทคโนโลยีการผลิตไฟฟ้าขนาดเล็กมาก เหล่านี้สามารถทาให้เกิดไมโครกริด (Microgrid) ซึ่งหมายถึงการเชื่อมต่อระหว่างกลุ่มผู้ผลิตไฟฟ้ารายเล็กกับกลุ่มผู้ใช้ไฟฟ้าที่อยู่ไม่ไกลจากกัน หลักการสำคัญของการผลิตไฟฟ้าด้วยไมโครกริดคือการสร้างสมดุลระหว่างการผลิตไฟฟ้าจากระบบผลิตไฟฟ้าขนาดเล็กมากดังกล่าวข้างต้น ให้พอดีกับภาระทางไฟฟ้าภายในไมโครกริด โดยขนาดของระบบผลิตไฟฟ้ามีขนาดใกล้เคียงกับภาระทางไฟฟ้าของไมโครกริดนั้นๆ แนวคิดดังกล่าว จะอนุญาตให้ผู้ใช้และผู้ผลิตไฟฟ้าภายในไมโครกริดใดๆ สามารถผลิตและใช้ไฟฟ้า ในขณะที่มีการเชื่อมต่อกับกริดไฟฟ้าขนาดใหญ่ หรือไมโครกริดอื่นๆ ได้ ซึ่งไมโครกริดจะมองเห็นการเชื่อมต่อนั้นๆ เป็นแหล่งกำเนิด/ภาระไฟฟ้าอีกแหล่งหนึ่งที่เพิ่มเข้ามา หรือจ่ายออกไปนั้นเอง ความก้าวหน้าของระบบควบคุม ระบบการแปลงไฟฟ้า และระบบเชื่อมต่อไฟฟ้าด้วยวงจรอิเล็กทรอนิกส์ในปัจจุบันทาให้กลุ่มผู้ใช้ไฟฟ้าในระดับครัวเรือน หรือกลุ่มอาคาร หมู่บ้านชุมชน สามารถนาระบบไฟฟ้าขนาดเล็กอื่นๆ เช่น ไมโครเทอร์ไบน์เพื่อผลิตไฟฟ้าและความร้อนจ่ายให้กับภาระไฟฟ้า และความร้อนภายในไมโครกริดนั้นๆ ขณะเดียวกันไมโครกริดเชื่อมต่อกับระบบไฟฟ้าในโครงข่ายไฟฟ้าขนาดใหญ่เดิมได้ด้วย ทั้งนี้ การเปลี่ยนเชื้อเพลิงไปเป็นทั้งพลังงานไฟฟ้าและพลังงานความร้อน ทำให้ประสิทธิภาพกรใช้พลังงานสูงและเมื่อแหล่งผลิตไฟฟ้าขนาดเล็กเป็นเทคโนโลยีที่ใช้พลังงานหมุนเวียน ก็จะเป็นมิตรกับสิ่งแวดล้อมอีกนัยหนึ่งด้วย

เพราะฉะนั้นแล้ว จึงต้องมีระบบการจัดการพลังงานในไมโครกริด เมื่อไมโครกริดอยู่ในสภาวะต่างๆ ให้ระบบมีเสถียรภาพและความน่าเชื่อถือ และไมโครกริดนั้นมีแหล่งกำเนิดพลังงานที่กระจายตัวอยู่จำนวนมาก จึงต้องมีการวิเคราะห์ประมวลผลร่วมกันเพื่อที่จะร่วมกันผลิตพลังงานได้อย่างเหมาะสม จากรูปแบบในการดำเนินงานไมโครกริดจึงได้มีการนำเทคโนโลยีมัลติเอเจนท์ มาประยุกต์ใช้ในการควบคุมชุดของหน่วยผลิตกำลังซึ่งเป็นส่วนหนึ่งในไมโครกริด โดยในการดำเนินการนั้นได้ใช้ชุดพัฒนาจาวาเอเจนท์ (Java Agent Development Framework) ซึ่งมีความเป็นไปได้ที่จะนำระบบนี้ไปประยุกต์ใช้ในระบบไฟฟ้ากำลังที่มีอยู่

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 เพื่อศึกษาโครงสร้างและหลักการของระบบไมโครกริด

1.2.2 เพื่อศึกษาหลักการของระบบมัลติเอเจนท์ รวมไปถึงวิธีการใช้งานชุดพัฒนาจาวาเอเจนท์ (Java Agent Development Framework : JADE)

1.2.3 เพื่อเป็นต้นแบบในการประยุกต์ใช้ระบบมัลติเอเจนท์เข้ามาในระบบจัดการพลังงานของไมโครกริด

1.3 ขอบเขตของการศึกษา

โครงงานนี้เป็นการรวมผลศึกษาเกี่ยวกับโครงสร้างของระบบไมโครกริดและหลักการจัดการพลังงานของไมโครกริด และได้นำเอาเทคโนโลยีของระบบมัลติเอเจนท์ (Multi-Agent System) เข้ามาประยุกต์ใช้โดยการสร้างแบบจำลองพื้นฐานของไมโครกริดและสร้างเอเจนท์ให้ทำงานในส่วนต่างๆ โดยเน้นที่เสถียรภาพในช่วงสภาวะคงตัว (steady-state) ในการรักษาสมดุลของกำลังไฟฟ้าที่ต้องการกับการผลิตอย่างเหมาะสม

1.4 แผนการดำเนินโครงงาน

การดำเนินงาน	ช่วงเวลา								
	มิ.ย.	ก.ค.	ส.ค.	ก.ย.	ต.ค.	พ.ย.	ธ.ค.	ม.ค.	ก.พ.
ศึกษาข้อมูล และงานวิจัยที่เกี่ยวกับไมโครกริด									
ศึกษาโครงสร้างการเขียนโปรแกรมด้วยภาษาจาวา									
ศึกษาระบบ Multi-agent System และ Java Agent Development Framework									
เตรียมนำเสนอและจัดทำเอกสารทางวิชาการผลศึกษาในภาคเรียนที่ 1									
ทดลองสร้างโปรแกรม Agent พื้นฐานและทดสอบคุณสมบัติต่างๆ									
สร้างแบบจำลองพื้นฐานและสร้างโปรแกรมเอเจนท์เพื่อสั่งการ									
ทดสอบโปรแกรมและตรวจสอบแก้ไขข้อผิดพลาด									
เตรียมนำเสนอโครงงาน จัดทำเอกสารทางวิชาการ และปริญญานิพนธ์									

1.5 ประโยชน์ที่คาดว่าจะได้รับ

โครงงานวิศวกรรมนี้ เป็นการนำความรู้จากทฤษฎีที่ได้จากการศึกษามาตลอด 4 ปี โดยเฉพาะที่เกี่ยวกับการวิเคราะห์ระบบไฟฟ้ากำลัง (Power System Analysis) สำหรับการจำลองเพื่อวิเคราะห์การไหลของกำลังไฟฟ้า (Power Flow Analysis) ในสภาวะคงตัว นอกจากนี้ยังต้องศึกษาภาษาจาวาเพิ่มเติมเพื่อนำมาเขียนโปรแกรมสำหรับสั่งการเอเจนท์ด้วยชุดพัฒนาจาวาเอเจนท์ (Java Agent Development Framework) โดยกำหนดให้เอเจนท์แต่ละแบบมีหน้าที่แตกต่างกัน แต่สามารถสื่อสารกันเพื่อวิเคราะห์และประมวลผลร่วมกันได้

1.6 เนื้อหาของปริญญานิพนธ์

บทที่ 2 เป็นการอธิบายทฤษฎีเกี่ยวกับโครงสร้างของไมโครกริด แหล่งพลังงานของไมโครกริด และการจัดการในระบบโครกริด และระบบมัลติเอเจนท์ซึ่งนำมาพัฒนาในส่วนของการบริหารการจัดการส่วนต่างๆ ในไมโครกริด

บทที่ 3 เป็นการอธิบายวิธีการใช้งานการใช้ระบบมัลติเอเจนท์โดยใช้ชุดพัฒนาจาวาเอเจนท์ (Java Agent Development Framework) ซึ่งจะอธิบายตั้งแต่การติดตั้ง วิธีการเขียนโปรแกรม ไปจนถึงวิธีการสื่อสารของเอเจนท์ที่สร้างขึ้น

บทที่ 4 เป็นการจำลองระบบไมโครกริด แล้วทำการจำลองด้วยเอเจนท์ซึ่งทำหน้าที่ต่างๆ กัน ทำการประสานงานกันเพื่อสั่งการจ่ายโหลดให้รักษาสมดุลกับความต้องการ ณ เวลาต่างๆ

บทที่ 5 เป็นการสรุปผลของโครงงานที่ทำมาทั้งหมด ข้อเสนอแนะ ข้อบกพร่องและข้อคิดเห็นสำหรับการนำโครงงานนี้ไปพัฒนาการต่อไป

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

ระบบจัดการพลังงานสำหรับไมโครกริด (Microgrid Energy Management System) เป็นการประยุกต์ใช้ความรู้ทางการวิเคราะห์การไหลของกำลังไฟฟ้าภายในระบบไฟฟ้าขนาดเล็กที่เรียกว่า ไมโครกริด (Microgrid) เข้ากับเทคโนโลยีที่มีชื่อว่า ระบบมัลติเอเจนท์ (Multi – Agent System) โดยทำการสร้างระบบด้วยชุดพัฒนาจาวาเอเจนท์ (Java Agent Development Framework) ระบบซอฟท์แวร์ชุดนี้จะช่วยให้การทำงานของระบบไฟฟ้าขนาดเล็กหรือไมโครกริดทำงานได้อย่างมีเสถียรภาพมากขึ้นและสามารถต่อยอดเพื่อขยายขีดความสามารถในการวิเคราะห์และตัดสินใจสั่งการระบบให้มีความชาญฉลาดมากขึ้นต่อไปในอนาคต

2.1 ระบบไมโครกริด

California Energy Commission ได้ให้นิยามของ "ไมโครกริด" ไว้ว่าไมโครกริดคือการรวมระบบพลังงานซึ่งประกอบด้วยการเชื่อมต่อของโหลดภายในระบบและทรัพยากรพลังงานแบบกระจาย (Distributed energy resources) ซึ่งเป็นระบบที่สามารถปฏิบัติการคู่ขนานไปกับกริดหลัก (Main Grid) ซึ่งเรียกว่า โหมดเชื่อมต่อกริด (Grid connect) หรือในสภาวะที่แยกตัวอิสระจากกริดหลักซึ่งเรียกว่าโหมดแยกอิสระ (Stand alone)

ระบบไฟฟ้ากำลังขนาดเล็กโดยใช้แหล่งกำเนิดไฟฟ้าขนาดเล็ก (Microsource) เช่น ไมโครเทอร์ไบน์ (Microturbine) หรือ เซลล์เชื้อเพลิง (Fuel Cells) รวมทั้งการผลิตไฟฟ้าจากแหล่งพลังงานหมุนเวียน หรือรวมเรียกว่า แหล่งกำเนิดไฟฟ้า ณ จุดใช้งาน (Distributed Energy Resource, DER) เพื่อปรับปรุงความน่าเชื่อถือได้ของระบบไฟฟ้าและตอบสนองการเปลี่ยนแปลงที่ตามมาจากการเปิดเสรีด้านพลังงานไฟฟ้า เรียกว่าแนวคิดไมโครกริด (Microgrid Concept)

2.1.1 ระบบผลิตไฟฟ้าแบบกระจายศูนย์ (Distributed generation : DG)

ระบบไฟฟ้าทั่วโลกเริ่มตระหนักถึงปัญหาการลดลงของปริมาณของทรัพยากรเชื้อเพลิงฟอสซิลและการทำให้พลังงานและสภาพแวดล้อมต่างๆ ค่อยๆ ทรุดโทรมลงไป ปัญหาเหล่านี้ได้นำไปสู่แนวความคิดใหม่ในการผลิตพลังงานภายในพื้นที่ ณ ระดับการกระจายแรงดันต่างๆ โดยการใช้พลังงานทดแทนและพลังงานหมุนเวียน เช่นก๊าซธรรมชาติ ก๊าซชีวภาพ พลังงานลม เซลล์แสงอาทิตย์ เซลล์เชื้อเพลิงความร้อนร่วม (Combined heat and power : CHP) ระบบไมโครเทอร์ไบน์ และเครื่องยนต์สเตอร์ลิง (Stirling engine) ซึ่งการรวมตัวของแหล่งพลังงานเหล่านี้จะเป็นคุณลักษณะที่ประโยชน์ต่อเครือข่ายการจำหน่าย การผลิตไฟฟ้าในรูปแบบนี้จะเรียกว่า ระบบผลิตไฟฟ้าแบบกระจายศูนย์หรือ Distributed Generation (DG) และแหล่งพลังงานเหล่านี้จะเรียกว่า แหล่งพลังงานแบบกระจายศูนย์ หรือ Distributed energy resources

(DER) โดยการผลิตแบบกระจายศูนย์นี้ได้รับการออกแบบให้มีความแตกต่างจากการผลิตไฟฟ้าจากส่วนกลาง (Centralized conventional generation) ในสมัยก่อน และการใช้ระบบการจำหน่ายร่วมกับการรวมตัวของ DG จะทำให้ระบบดังกล่าวเป็นระบบที่มีความรวดเร็วในการจำหน่ายไฟฟ้าไปยังส่วนต่างๆ

ในปลายปี 1990 ได้มีการให้ความสำคัญกับการศึกษาประเด็นหลักที่เกี่ยวข้องกับ DG โดยคณะทำงานของ International Council on Large Electric Systems (CIGRE) และการประชุมนานาชาติและ Exhibition on Electricity Distribution (CIRED) และนำเสนออยู่ในรายงานการประชุม

ข้อกำหนดเกี่ยวกับ DG ที่พบในหลายๆ ประเทศนั้นส่วนใหญ่นั้นจะขึ้นอยู่กับขนาดพิกัดของโรงจักรไฟฟ้า ระดับแรงดันไฟฟ้าในการผลิต ฯลฯ จากการศึกษาและวิจัยได้ข้อสรุปลักษณะทั่วไปของ DG ได้ดังต่อไปนี้

1. ไม่เป็นศูนย์กลางของระบบไฟฟ้าหรือศูนย์กลางการจำหน่ายไฟฟ้า
2. มีขนาดเล็กกว่า 50 เมกะวัตต์
3. แหล่งกำเนิดกำลังไฟฟ้าหรือเครื่องกำเนิดไฟฟ้าแบบกระจายศูนย์มักจะเชื่อมต่อกับระบบจำหน่ายที่มีขนาดของแรงดัน 230/415 V ถึง 145 kV

2.1.2 ข้อดีของระบบผลิตไฟฟ้าแบบกระจายศูนย์

แม้ว่าระบบไฟฟ้าแบบธรรมดาจะมีข้อดีหลายประการ ทั้งข้อดีทางด้านเทนนิค เศรษฐกิจและสิ่งแวดล้อมแต่ก็ยังมีการพัฒนาอย่างค่อยเป็นค่อยไปในการนำการรวมผลิตแบบกระจายศูนย์เข้ากับระบบ ทั้งนี้ด้วยสาเหตุ :

1. เนื่องจากภาระทางไฟฟ้ามีการเจริญเติบโตอย่างรวดเร็ว ความจำเป็นในความต้องการของพลังงานทั่วไปก็เพิ่มขึ้นทำให้เพลิงเชื้อที่ใช้ในการผลิตลดลง ด้วยเหตุนี้เราจึงหาพลังงานทดแทนและพลังงานหมุนเวียนมาเพื่อเป็นพลังงานสำรอง

2. การลดมลภาวะทางสิ่งแวดล้อมและสภาวะโลกร้อนถือเป็นปัจจัยที่ให้ความสำคัญเป็นอันดับแรก ซึ่งเป็นที่น่าสนใจในการใช้ทรัพยากรหมุนเวียนมากกว่าเชื้อเพลิงฟอสซิล ในสนธิสัญญาเกียวโต (Kyoto Protocol), สหภาพยุโรป (EU), สหราชอาณาจักร (UK) และประเทศอื่นๆ ได้มีการวางแผนเพื่อลดก๊าซเรือนกระจก (คาร์บอนและไนโตรเจน) ที่ปล่อยสู่อากาศเพื่อให้ต่อต้านการเปลี่ยนแปลงของสภาพอากาศรวมถึงภาวะโลกร้อน ดังนั้นพวกเขาจะสร้างพลังงานรูปแบบใหม่และการใช้นโยบายเพื่อสนับสนุนการใช้พลังงานที่เหมาะสมเหล่านี้ คาดว่าประโยชน์ของแหล่งพลังงานแบบกระจายศูนย์จะช่วยสร้างสิ่งที่ไม่เป็นอันตรายต่อสิ่งแวดล้อมซึ่งเป็นพลังงานสะอาดที่มีผลกระทบต่อสิ่งแวดล้อมน้อยมาก

3. ระบบผลิตไฟฟ้าแบบกระจายศูนย์จะเป็นแนวทางสำหรับการติดตั้งการร่วมกันผลิต (Co – Generation) การผลิตพลังงานร่วม (tri-generation หรือ CHP) ซึ่งใช้ประโยชน์จากความร้อนเสียของโรงงานอุตสาหกรรมในประเทศการค้าเชิงพานิชย์ได้ดีกว่า ซึ่งจะเพิ่มประสิทธิภาพ

ใช้การใช้พลังงานทั้งหมดของโรงจักรไฟฟ้าและยังช่วยลดมลพิษทางความร้อนของสภาพแวดล้อม

4. เนื่องจากความหนาแน่นทางพลังงานที่ต่ำกว่าและขึ้นกับเงื่อนไขทางกายภาพของภูมิประเทศซึ่งมี DER เป็นหน่วยโมดูล (modular unit) พื้นฐานโดยส่วนมากมีกำลังการผลิตที่ไม่มากนัก โดย DER จะกระจายอยู่ทั่วไปตามพื้นที่และควรอยู่ใกล้กับโหลด จึงกลายมาเป็นข้อกำหนดสำหรับการทำงานของโรงจักรไฟฟ้าให้มีเสถียรภาพและมีความคุ้มทุนเชิงเศรษฐศาสตร์ ตัวอย่างเช่น CHP ต้องอยู่ใกล้กับโหลดเนื่องจากการส่งระยะไกลทำให้เกิดความสิ้นเปลือง ซึ่งจะทำให้ง่ายในการหาค้นหาและช่วยลดในเรื่องของการก่อสร้างและเงินทุน ด้วยระยะทางที่ตั้งที่ไม่ไกลจากโหลดและแหล่งจ่ายและยังช่วยลดการสูญเสียในการส่งและการกระจาย และเนื่องจากกำลังไฟฟ้านั้นถูกผลิตด้วยแรงดันไฟฟ้าต่ำจึงสามารถที่จะเชื่อมต่อ DER โดยให้แยกจากเครือข่ายการจำหน่ายระบบสาธารณูปโภคหรืออาจมีการเชื่อมต่อภายในเป็นรูปแบบของไมโครกริดได้ โดยไมโครกริดนั้นยังสามารถที่จะกลับมาเชื่อมต่อเข้ากับระบบที่แยกออกไปในลักษณะกึ่งอิสระ (semi-autonomous) ได้อีกครั้ง

5. การดำเนินงานในโหมดแยกอิสระ (Stand-alone) และโหมดเชื่อมต่อกับกริด (Grid-connect) ของ DER จะช่วยในการขยายการผลิตและนับเป็นการช่วยปรับปรุงคุณภาพไฟฟ้าและความน่าเชื่อถือโดยรวม นอกจากนี้การที่มันไม่อยู่ภายใต้กฎของสภาวะแวดล้อมรวมถึงการเข้าถึงเครือข่ายการจำหน่ายนั้นนับว่าเป็นการเพิ่มโอกาสสำหรับการรวมระบบเข้ากับDG

2.1.3 โครงข่ายระบบจำหน่ายแบบแอคทีฟ (Active distribution network)

ปัจจุบันโครงข่ายการจำหน่ายไฟฟ้านั้นอยู่ในยุคของการเปลี่ยนแปลงจากโครงข่ายระบบจำหน่ายแบบพาสซีฟ (Passive) ซึ่งส่งกำลังในทิศทางเดียวพัฒนาสู่โครงข่ายระบบจำหน่ายแบบแอคทีฟ (Active) ซึ่งมีการส่งกำลังได้สองทิศทาง โดยโครงข่ายระบบจำหน่ายที่ไม่มี DG นั้นจะเรียกว่าเป็นโครงข่าวระบบจำหน่ายแบบพาสซีฟเนื่องจากกำลังไฟฟ้าจะถูกจ่ายมาจากระบบของเมนกริดให้กับผู้บริโภคที่อยู่ในเครือข่ายระบบจำหน่ายทั้งหมดและจะกลายเป็นระบบแบบแอคทีฟเมื่อมีหน่วย DG เพิ่มเข้าไปในระบบจำหน่าย จึงกลายเป็นการไหลของกำลังไฟฟ้าแบบสองทิศทาง โครงข่ายระบบจำหน่ายแบบแอคทีฟจำเป็นที่จะต้องมีการรวมเอาความยืดหยุ่นและความชาญฉลาดในการควบคุมเข้าไว้ด้วยกัน และเพื่อเป็นการควบคุมพลังงานในเป็นพลังงานที่สะอาดจากการผลิตของ DER ที่เป็นพลังงานหมุนเวียน ดังนั้นโครงข่ายระบบจำหน่ายแบบแอคทีฟจึงควรจะนำมาใช้ในการปฏิบัติงานของโครงข่ายระบบจำหน่ายในอนาคตซึ่งจะนำไปสู่การเป็นกริดอัจฉริยะหรือสมาร์ทกริด (Smartgrid) หรือโครงข่ายไมโครกริด (Microgrid)

ในปัจจุบันกลยุทธ์ที่เรียกว่า 'fit-and-forget' ของ DG ต้องการความเปลี่ยนแปลงในด้านการจัดการโครงข่ายแบบแอคทีฟ ซึ่งได้รวมเอาหน่วยของ DG เข้ากับระบบจำหน่ายรวมถึงการจัดการด้านปริมาณความต้องการเข้าด้วยกัน และได้มีการนำเสนอโดยศูนย์กลางเพื่อการผลิต

แบบกระจายศูนย์และพลังงานไฟฟ้าอย่างยั่งยืนแห่งสหราชอาณาจักร (UK - based Centre for Distributed Generation : www.sedg.ac.uk) ว่าการประยุกต์ใช้วิธีการจัดการโครงข่ายระบบการจำหน่ายแบบแอคทีฟนั้นสามารถรองรับการเชื่อมต่อของ DG หลายๆ ตัวได้ดีกว่าเมื่อเทียบกับโครงข่ายที่ไม่มีการจัดการแบบแอคทีฟนี้

ดังนั้นในการที่จะใช้วิวัฒนาการโครงข่ายระบบจำหน่ายแบบแอคทีฟเพื่อการควบคุมการดำเนินงานที่มีความยืดหยุ่นและสั่งการอย่างชาญฉลาด การวิจัยที่ครอบคลุมจึงเป็นสิ่งจำเป็น โดยการวิจัยจะมุ่งเน้นในประเด็นเหล่านี้เป็นหลักอันได้แก่

- การควบคุมบนพื้นที่บริเวณกว้าง
- ความสามารถในการปรับตัวสำหรับการป้องกันและควบคุม
- อุปกรณ์ในการจัดการโครงข่าย
- การจำลองระบบโครงข่ายแบบเวลาจริง (Realtime)
- เซ็นเซอร์และเครื่องมือวัดระดับสูง
- การสื่อสารอย่างกว้างขวางในระบบการจำหน่าย
- กระบวนการคิดวิเคราะห์ด้วยวิธีอันชาญฉลาด
- การออกแบบระบบการส่งกำลังไฟฟ้ารวมถึงระบบจำหน่ายในรูปแบบใหม่

2.1.4 กรอบความคิดของไมโครกริด

ไมโครกริดนั้นจะมีพิกัดกำลังที่ไม่สูงมากนักและเป็นระบบแรงดันต่ำโดยที่โครงข่ายการรองรับการจ่ายโหลดด้วยระบบการผลิตไฟฟ้าและความร้อนร่วม (Combined heat and power : CHP) ได้ถูกออกแบบมาเพื่อผลิตไฟฟ้าและโหลดความร้อนสำหรับพื้นที่ขนาดเล็ก เช่นบ้านจัดสรร, ชานเมืองท้องถิ่น, หรือพื้นที่สาธารณะ เช่น มหาวิทยาลัยหรือโรงเรียน เขตการค้าอุตสาหกรรมด้านการค้าหรือพื้นที่เทศบาล

โดยส่วนใหญ่แล้วไมโครกริดจะเป็นโครงข่ายระบบจำหน่ายแบบแอคทีฟเสียส่วนใหญ่ ทั้งนี้เนื่องจากแนวคิดของไมโครกริดนั้นได้รวมเอาระบบผลิตไฟฟ้าแบบกระจายศูนย์ (DG) และโหลดที่แตกต่างกัน ณ ระดับแรงดันไฟฟ้าในการจำหน่ายเข้าไว้ด้วยกัน เครื่องกำเนิดไฟฟ้าหรือแหล่งจ่ายขนาดเล็กที่ใช้ในระบบไมโครกริดส่วนมากเป็นพลังงานหมุนเวียนและพลังงานทดแทน ซึ่งแหล่งกำเนิดไฟฟ้า ณ จุดใช้งาน (DER) จะรวมเอาแหล่งพลังงานทั้ง 2 เข้าด้วยกันเพื่อผลิตกำลังไฟฟ้าที่แรงดันของระบบจำหน่าย

ความแตกต่างระหว่างไมโครกริด และโรงไฟฟ้าทั่วไปมีดังต่อไปนี้

- แหล่งจ่ายขนาดเล็กนั้นมีพิกัดกำลังการผลิตที่เล็กมากถ้าเทียบกับเครื่องกำเนิดไฟฟ้าขนาดใหญ่ในโรงไฟฟ้าทั่วไป
- กำลังไฟฟ้าที่ถูกสร้างขึ้นที่ระดับแรงดันของระบบจำหน่ายนั้นสามารถที่จะป้อนให้กับระบบจำหน่ายของสาธารณูปโภคได้โดยตรง

- แหล่งจ่ายขนาดเล็กโดยปกติจะติดตั้งใกล้กับสถานที่ของลูกค้าเพื่อจ่ายไฟฟ้าให้แก่ผู้บริโภคได้อย่างมีประสิทธิภาพด้วยระดับแรงดันไฟฟ้าและความที่มีเสถียรภาพ และยังเป็นการลดการสูญเสียในสายส่งให้มีค่าน้อยลง

ในมุมของของกริดหลัก (main grid) ข้อดีหลักของไมโครกริดคือการที่มันสามารถควบคุมตัวเองได้ภายในระบบไฟฟ้า มันจึงสามารถมองรวมเป็นภาระทางไฟฟ้าได้เลย ซึ่งจะทำให้ง่ายต่อการควบคุมและปฏิบัติตามกฎของกริดหลักรวมถึงสามารถรักษาระดับแรงดันโดยไม่ส่งผลกระทบหรือขัดขวางต่อความน่าเชื่อถือและความปลอดภัยในการใช้ประโยชน์จากกำลังไฟฟ้า

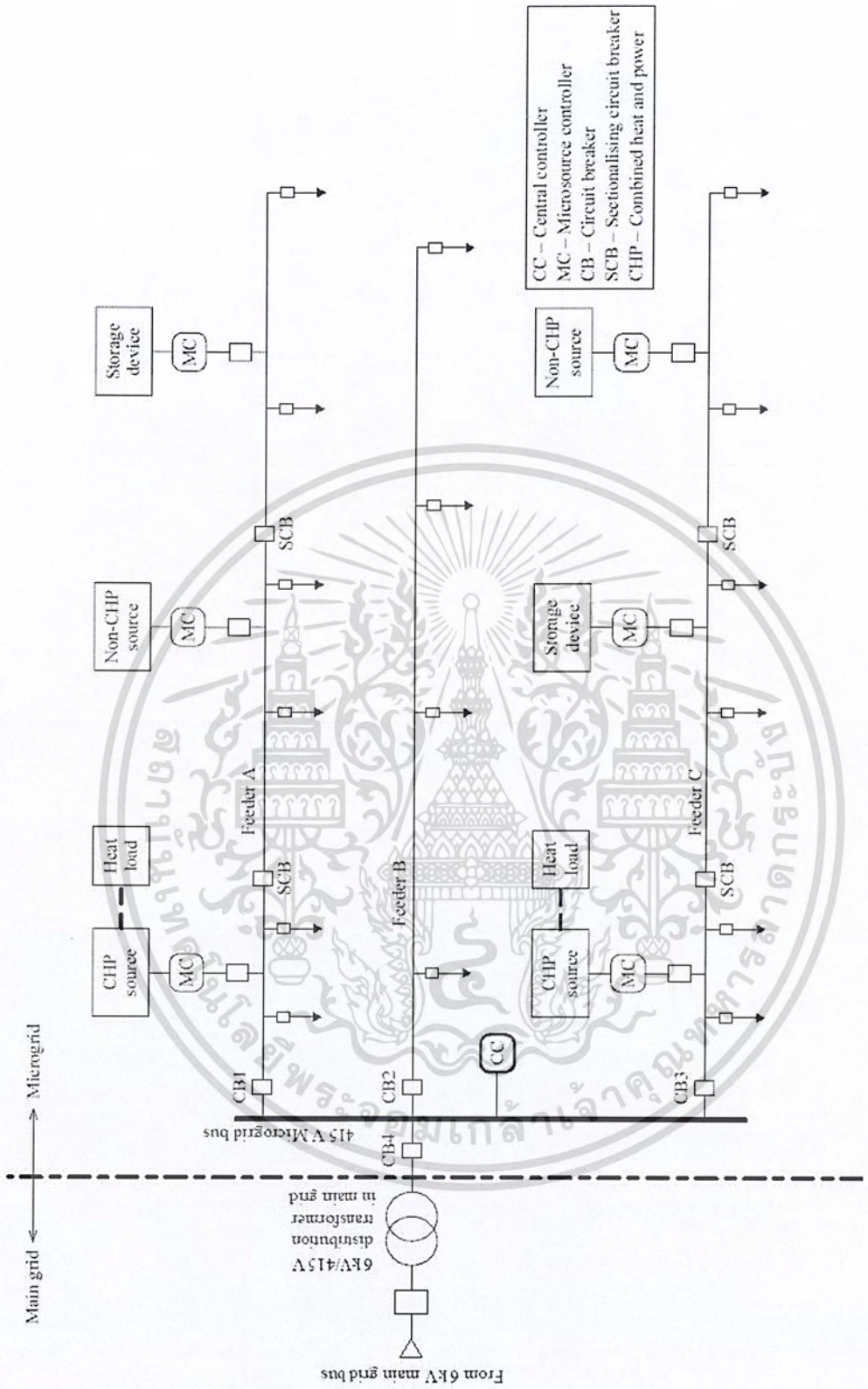
ในมุมมองจากผู้บริโภค ไมโครกริดนั้นจะมีประโยชน์มากในด้านการรองรับในช่วงไฟดับ ปรับปรุงความน่าเชื่อถือของระบบไฟฟ้า ช่วยลดกำลังสูญเสียบนสายส่งและมีการรองรับระดับแรงดันภายในพื้นที่อีกด้วย

ในมุมมองทางด้านสิ่งแวดล้อม ไมโครกริดนั้นจะลดมลภาวะทางสิ่งแวดล้อมและภาวะโลกร้อนผ่านการใช้ประโยชน์จากเทคโนโลยีการลดคาร์บอน (low-carbon technology)

2.1.5 องค์ประกอบทั่วไปของไมโครกริด

องค์ประกอบทั่วไปของไมโครกริดเป็นไปตามรูปที่ 2.1 ประกอบด้วยภาระทางไฟฟ้าหรือความร้อนรวมทั้งแหล่งจ่ายขนาดเล็ก (microsource) โดยมีเชื่อมต่อในโครงข่ายระบบจำหน่ายแรงดันต่ำ จะสังเกตว่าภาระ (โดยเฉพาะภาระทางความร้อน) จะถูกวางไว้ใกล้กับแหล่งจ่ายเพื่อลดการสูญเสีย

ไมโครกริดประกอบด้วยสายป้อนตามแนวรัศมีสามตัว คือ A B และ C เพื่อจ่ายให้กับภาระ นอกจากนี้ยังมี CHP 2 ตัว แหล่งจ่ายขนาดเล็กที่ไม่เป็น CHP 2 ตัวและอุปกรณ์กักเก็บพลังงาน โดยแหล่งจ่ายขนาดเล็กและอุปกรณ์กักเก็บพลังงานจะเชื่อมต่ออยู่ที่สายป้อน A และ C ผ่านตัวควบคุมแหล่งจ่าย (Microsource Controller : MC) โดยที่ต่อเข้าที่สายป้อน A และ C เท่านั้นเนื่องจากบางภาระบนสายป้อน A และ C นั้นสมมติให้เป็นโหลดที่มีความสำคัญ (ต้องจ่ายไฟเลี้ยงตลอดเวลา อาทิเช่นระบบแจ้งเหตุเพลิงไหม้) และบางส่วนอาจเป็นภาระที่ไม่ใช่ตัวที่ต้องให้ความสำคัญสามารถตัดออกจากระบบได้ซึ่งสมมุติให้ภาระชนิดนี้อยู่บนสายป้อน B



รูปที่ 2.1 องค์ประกอบทั่วไปของ Microgrid [1]

ไมโครกริดนั้นจะเชื่อมต่อกับกริดหลัก (main grid) แรงดันระดับกลางผ่านเซอร์กิตเบรคเกอร์ซึ่งเป็นจุดต่อร่วม (Point of Common Coupling : PCC) หรือ CB4 ซึ่งทำหน้าที่เชื่อมต่อและตัดตอนไมโครกริด ทั้งหมดจากกริดหลักตามโหมดที่เลือกการดำเนินงานนั่นคือถ้า CB4 ปิดแสดงว่าไมโครกริดทำงานอยู่ในโหมดเชื่อมต่อกับกริด (grid connect) แต่ถ้า CB4 เปิดแสดงว่าไมโครกริดทำงานอยู่ในโหมดแยกอิสระ (stand-alone)

ไมโครกริดดำเนินการในสองโหมด : (1) เชื่อมต่อกริด และ (2) ในโหมดแยกอิสระ โดยในโหมดเชื่อมต่อกริดนั้นไมโครกริดจะยังคงเชื่อมต่อกับกริดหลักทั้งหมดหรือบางส่วนและนำเข้าหรือส่งกำลังไปยังกริดหลัก ในกรณีที่เกิดสัญญาณรบกวนในกริดหลัก ไมโครกริดจะสลับไปยังโหมดแยกอิสระในขณะที่ยังคงจ่ายกำลังให้กับภาระที่สำคัญเหมือนเดิม โหมดแยกอิสระนี้สามารถทำได้โดย

(i) ตัดการเชื่อมต่อ Microgrid ทั้งหมดโดยเปิด CB4

(ii) ตัดการเชื่อมต่อ สายป้อน A และ C โดยเปิด CB1 และ CB3

สำหรับตัวเลือก (i) ไมโครกริดจะทำงานเป็นระบบอัตโนมัติที่มีทั้งหมดแหล่งจ่ายขนาดเล็กจะจ่ายกำลังให้กับภาระทั้งหมดในสายป้อน A, B และ C

ในขณะที่ตัวเลือก (ii) สายป้อน A และ C จะจ่ายเพียงโหลดสำคัญในขณะที่สายป้อน B จะถูกตัดทิ้งไปเนื่องจากเป็นโหลดไม่สำคัญ

การดำเนินการและการจัดการของไมโครกริดในโหมดต่างๆจะถูกควบคุมและมีการทำงานร่วมกัน (co – ordinate)ผ่านตัวควบคุมแหล่งจ่าย (MC) และตัวควบคุมส่วนกลาง (CC) ที่มีฟังก์ชั่นการทำงานดังต่อไปนี้ :

Microsource controller (MC) : หน้าที่หลักของ MC คือการควบคุมที่เป็นอิสระของการไหลกำลังไฟฟ้าและแรงดันที่บัสปลายของแหล่งจ่ายขนาดเล็กในการตอบสนองต่อการรบกวนรวมถึงการเปลี่ยนแปลงของภาระ ซึ่งคำว่าเป็นอิสระในที่นี้หมายถึงการควบคุมโดยไม่มีการสั่งการจากตัวควบคุมส่วนกลาง (CC) นั่นเองและตัว MC นี้จะต้องทำให้มั่นใจได้ว่าแหล่งจ่ายขนาดเล็กแต่ละตัวนั้นจะต้องสามารถเพิ่มระดับการผลิตได้อย่างทันท่วงทีเพื่อจ่ายกำลังไฟฟ้ารวมถึงมีการจัดสรรการผลิตให้กับภาระต่างๆในโหมดแยกอิสระและสามารถกลับมาเชื่อมต่อในโหมดของการเชื่อมต่อกริดได้อย่างอัตโนมัติโดยไม่มีการสั่งการจาก CC ซึ่งคุณลักษณะที่สำคัญที่สุดของ MC คือการตอบสนองที่รวดเร็วต่อการตรวจสอบระดับแรงดันและกระแสภายในพื้นที่โดยไม่มีการนำเอาข้อมูลจาก MC ข้างเคียงมาพิจารณา

สองคุณลักษณะพิเศษหลักของ MC คือ MC หนึ่งตัวจะทำงานโดยแยกอิสระ ไม่ขึ้นกับ MC ตัวอื่นๆในไมโครกริด และ มันจะสามารถทำการยกเลิกคำสั่งที่มาจาก CC ที่วิเคราะห์แล้วว่าจะก่อให้เกิดอันตรายต่อแหล่งจ่ายขนาดเล็กภายในระบบ

Central controller (CC) : จะจัดการควบคุมการดำเนินงานของไมโครกริดและการป้องกันผ่านMC โดยมีวัตถุประสงค์เพื่อ (i) เพื่อรักษาระดับแรงดันและความถี่ของภาระปลายทาง ผ่านการควบคุมกำลังไฟฟ้าและความถี่รวมถึงการควบคุมแรงดันไฟฟ้า (ii) เพื่อเพิ่ม

ประสิทธิภาพของพลังงานในไมโครกริดโดยCCได้ถูกออกแบบมาให้ทำงานในโหมดอัตโนมัติเมื่อมีการแทรกแซงการทำงานของระบบและในยามจำเป็น สองโมดูลการทำงานหลักของ CC คือโมดูลการจัดการพลังงาน(Energy Management Module) และ โมดูลป้องกันในการทำงานร่วมกัน (Protection Co-ordination Module)

Energy Management Module หรือ EMM จะทำการส่งค่าของจุดอ้างอิงของกำลังไฟฟ้าจริง (active power) และกำลังไฟฟ้าเสมือน (reactive power) แรงดันไฟฟ้าและความถี่ไปยังแต่ละ MC เพื่อใช้เป็นจุดอ้างอิงในการดำเนินงาน

Protection Co-ordination Module หรือ PCM จะทำการตอบสนองต่อความผิดพร่องในไมโครกริดรวมถึงกริดหลัก และภาวการณ์หายไปของกริด (Loss Of Grid : LOG) ด้วยคุณลักษณะดังกล่างจึงทำให้มั่นใจในการทำงานร่วมกันในการป้องกันไมโครกริดได้อย่างถูกต้องและยังรวมถึงการปรับตัวต่อการเปลี่ยนแปลงต่อระดับกระแสผิดพร่องในระหว่างเกิดการเปลี่ยนสภาวะการทำงานจากโหมดเชื่อมต่อกริดไปเป็นโหมดแยกอิสระ

2.1.6 ตัวควบคุมในระบบไมโครกริด

ในระบบไมโครกริจจะมีตัวควบคุมหลักๆ อยู่ 2 ชนิดคือตัวควบคุมแหล่งจ่ายและตัวควบคุมส่วนกลาง

(1) ตัวควบคุมแหล่งจ่าย (Microsource Controller : MC)

แหล่งจ่ายขนาดเล็ก (microsource) และอุปกรณ์สำรองพลังงาน (Storage Device) ที่ติดตั้งเข้ากับ MC จะต้องสามารถดำเนินการได้อย่างราบรื่นและมีความยืดหยุ่น เพื่อตอบสนองความต้องการของผู้บริโภคและสาธารณูปโภค การทำงานของMCนั้นอาจดำเนินการโดยมีการแทรกแซงหรือไม่มีการสั่งการจาก CC ก็ได้ ฟังก์ชันการทำงานของ MC ส่วนมากจะขึ้นกับการเชื่อมต่อด้านอิเลคทรอนิกส์กำลัง (Power Electronics Interface) ที่ใช้ในแหล่งจ่ายรวมถึงอุปกรณ์สำรองพลังงาน เงื่อนไขการดำเนินงานของMC มีดังนี้ (i) การเพิ่มแหล่งจ่ายตัวใหม่เข้าไปในระบบโดยไม่มีการเปลี่ยนแปลงโครงสร้างเดิม (ii) ไมโครกริดสามารถสับ/ปลด โดยตัวเองได้อย่างรวดเร็ว (iii) สามารถควบคุมกำลังจริงและกำลังเสมือนได้อย่างอิสระ (iv) สามารถแก้ไขแรงดันดันตก (Voltage Sag) และความไม่สมดุลของระบบจะได้ (v) สามารถจัดการกับความผิดพร่อง (fault) ต่างๆได้โดยไม่สูญเสียความมั่นคงของระบบ และ (vi) ไมโครกริดสามารถตอบสนองความต้องการของพลศาสตร์ของภาระได้

กุญแจสำคัญในการออกแบบ MC มีดังนี้

ไม่มีขอบเขตสำหรับปฏิสัมพันธ์ระหว่างแหล่งจ่ายโดยปราศจากการแทรกแซงจาก CC ซึ่งจะช่วยให้แต่ละ MC มีการดำเนินงานที่ประสิทธิภาพ เพื่อตอบสนองการเปลี่ยนแปลงของระบบโดยไม่ต้องใช้ข้อมูลจาก MC อื่นๆ หรือแหล่งข้อมูล

MC จะถูกออกแบบ เพื่อติดต่อสื่อการกับ CC และกระทำตามคำสั่งของตัวมันเอง มันสามารถแทนที่คำสั่งของ CC ที่แหล่งจ่ายไม่สามารถยอมรับได้

ฟังก์ชันการควบคุมของตัวควบคุมแหล่งจ่าย ประกอบด้วย

- การควบคุมกำลังไฟฟ้าจริงและกำลังไฟฟ้าเสมือน (Active and Reactive Power Control)
- การควบคุมแรงดัน (Voltage Control)
- ความต้องการในการจัดเก็บพลังงานสำหรับการติดตามภาระอย่างรวดเร็ว
- การจัดสรรภาระผ่านการควบคุมกำลังและความถี่(P-f Control)

MC ควรตรวจสอบว่าแหล่งจ่ายแต่ละตัวนั้นสามารถรอบรับการจัดสรรจ่ายโหลดได้อย่างรวดเร็ว เมื่อไมโครกริดปลดตัวเองออกจากกริดหลัก

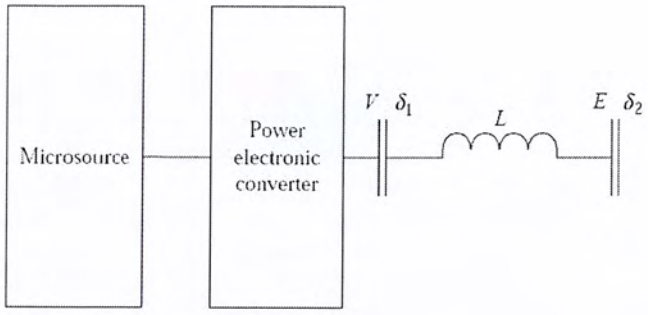
<u>การควบคุมกำลังไฟฟ้าจริงและกำลังไฟฟ้าเสมือน (Active and Reactive Power Control)</u>

แหล่งจ่ายขนาดเล็กที่เรียกว่า Microsource อาจเป็นแหล่งจ่ายแหล่งจ่ายกระแสตรง อย่างเช่น เซลล์แสงอาทิตย์ (PV) เซลล์เชื้อเพลิง (Fuel cell) อุปกรณ์สำรองพลังงาน (Storage Battery) หรือเป็นแหล่งจ่ายกกระแสสลับอย่างพวกกังหันแก๊สขนาดเล็ก (Microturbine) หรือกังหันลม สำหรับแหล่งจ่ายกระแสตรงนั้นจะถูกแปลงเป็นกระแสสลับโดยตรงที่ P-f (50/60 Hz) แต่ขณะที่กระแสสลับซึ่งมีความถี่ที่ไม่เป็นที่แน่นอนจะถูกแปลงเป็นกระแสตรงก่อนและจึงแปลงกลับเป็นกระแสสลับอีกครั้งที่ความถี่ปกติด้วยอินเวอร์เตอร์ ซึ่งทั้งสองอย่างต่างก็ต้องใช้ DC/AC Converter

ในรูปที่ 2.2 แสดงโครงสร้างทั่วไปของ MC ซึ่งประกอบด้วยแหล่งจ่ายขนาดเล็ก(Microsource) และอินเวอร์เตอร์อิเลคทรอนิกส์กำลัง(Power Electronic Converter) โดยอินเวอร์เตอร์ของแหล่งจ่ายแรงดัน ในส่วนของคอนเวอร์เตอร์จะควบคุมขนาดแรงดัน (V) และมุมเฟส (δ_1) ของแรงดันขาออก $(V\angle\delta_1)$ ที่ปลายบัส 1 (Bus-1) Microsource จะควบคุมการจ่ายกำลังที่บัส 2 (Bus-2) ที่แรงดัน $(E\angle\delta_2)$ ผ่านค่าความเหนี่ยวนำที่มีค่ารีแอกแตนซ์ X โดยปกติ $(V\angle\delta_1)$ จะนำหน้า $(E\angle\delta_2)$ อยู่ที่มุมกำลัง δ โดยที่ $\delta=\delta_1-\delta_2$ การไหลของกำลังไฟฟ้าจริง (P) จะถูกควบคุมโดยการควบคุม δ ส่วนกำลังไฟฟ้าเสมือน จะถูกควบคุมโดยการควบคุม V ตัวควบคุมจะอยู่บนฐานของการควบคุมลูปปิด (Feedback loop) ของกำลังขาออก P และขนาดแรงดันบัส E ซึ่งเป็นไปตาม สมการที่ 2.1 และ สมการที่ 2.2 ตามลำดับ ดังนี้

$$P=\frac{3EV}{2X}\sin\delta \tag{2.1}$$

$$Q=\frac{3EV}{2X}\left(V-E\cos\delta\right) \tag{2.2}$$

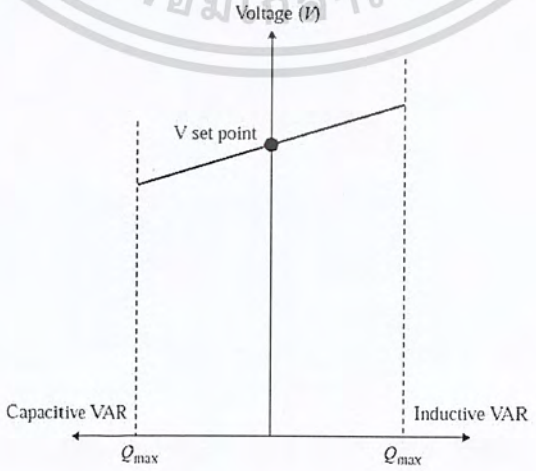


รูปที่ 2.2 รูปแบบทั่วไปของ MC [1]

การควบคุมแรงดัน (Voltage Control)

นอกจากการควบคุมกำลังไฟฟ้าจริง (P) และกำลังไฟฟ้าเสมือน (Q) แล้ว การควบคุมแรงดันที่บัสของไมโครกริดจะมีความสำคัญต่อเสถียรภาพและความน่าเชื่อถือของไมโครกริดอีกด้วย ไมโครกริดที่มีแหล่งจ่ายจำนวนมาก อาจมีปัญหาในเรื่องของการแกว่งของกำลังเสมือน (Reactive Power) ซึ่งเกิดจากไม่มีการควบคุมแรงดันที่บัสให้เหมาะสม คล้ายกับเครื่องกำเนิดไฟฟ้าซิงโครนัสขนาดใหญ่ ฟังก์ชันควบคุมแรงดันของ MC จะจัดการปัญหากระแสรีแอคทีฟไหลวนระหว่างไมโครกริดให้บรรเทาลง สำหรับกริดหลักนั้นกระแสไหลวนเหล่านี้โดยทั่วไปจะถูกยับยั้งโดยอิมพิแดนซ์ขนาดใหญ่ระหว่างเครื่องกำเนิดไฟฟ้าปัญหาขึ้นมองเห็นได้ไม่ชัดเจนนัก แต่ในส่วนของไมโครกริดนั้นยังมีปัญหาค่อนข้างชัดเจน เนื่องจากสายป้อนมักเป็นจุดกระจายรัศมีซึ่งมีอิมพิแดนซ์ระหว่างแหล่งจ่ายต่ำ

บางครั้งกระแสไหลวนเหล่านี้อาจทำให้กระแสของแหล่งจ่ายมีค่าเกินพิกัด แม้ว่าจุดอ้างอิงแรงดันจะมีค่าต่างกันเพียงเล็กน้อย กระแสไหลวนเหล่านี้สามารถถูกควบคุมโดยใช้ตัวควบคุมดรูปแรงดันรีแอคทีฟ (Voltage-Reactive Power (V-Q) droop controller) กับคุณลักษณะการลดตัวซึ่งแสดงในรูปที่ 2.3



รูปที่ 2.3 คุณลักษณะดรูปสำหรับ V-Q droop controller [1]

ฟังก์ชันการควบคุม คือ การเพิ่มค่าจุดอ้างอิงของแรงดัน (Voltage Set Point) เมื่อกระแสรีแอคทีฟของแหล่งจ่ายมีลักษณะเป็นไปทางความเหนี่ยวนำ (Inductive) และลดค่าจุดอ้างอิงของแรงดัน เมื่อกระแสรีแอคทีฟเป็นไปทางความจุไฟฟ้า (Capacitive) ค่าจำกัดของกำลังไฟฟ้าเสมือนนั้นจะถูกกำหนดด้วยพิกัดของกำลังปรากฏ (VA Rating (VAR; *S*)) ของอินเวอร์เตอร์ และกำลังไฟฟ้าจริง (P) ขาออกของแหล่งจ่าย ดังความสัมพันธ์ต่อไปนี้

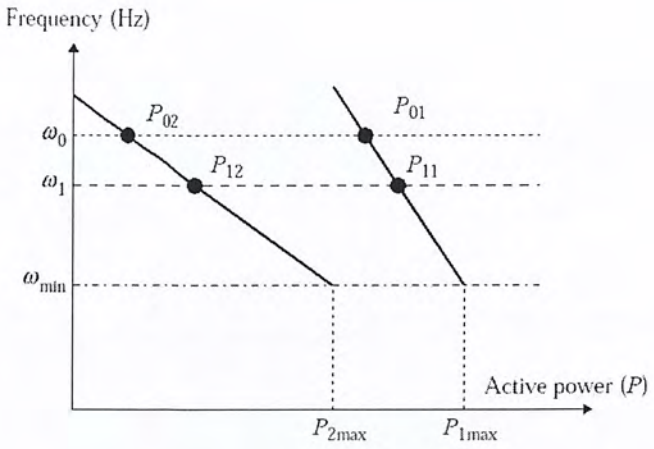
$$Q_{\max} = \sqrt{\left(S^2 - P^2\right)} \tag{2.3}$$

<u>ข้อกำหนดในการสำรองพลังงานเพื่อสมดุลโหลด</u>

สำหรับการทำงานของไมโครกริดโหมดการเชื่อมต่อกริดความสมดุลของกำลังเริ่มต้นขณะที่มีการต่อภาระเพิ่มเข้ามาใหม่จะได้รับการดูแลโดยความเฉื่อยที่มีมากของเครื่องกำเนิดไฟฟ้า แต่สำหรับการดำเนินงานในโหมดแยกอิสระ ไมโครกริดจะต้องมีการรักษาสมดุลของกำลังเริ่มต้นโดยการใช้อุปกรณ์จัดเก็บสำรองพลังงานที่มีประสิทธิภาพพอเป็นความเฉื่อยในระบบไมโครกริด อุปกรณ์สำรองพลังงานกระแสตรงจะถูกเชื่อมต่อที่บัสกระแสตรง (DC bus) ของแหล่งจ่ายขณะที่อุปกรณ์สำรองพลังงานกระแสสลับจะถูกเชื่อมต่อโดยตรงที่บัสของไมโครกริด ดังนั้น MC จะสร้างความมั่นใจในการใช้อุปกรณ์สำรองพลังงานที่เหมาะสมสำหรับการติดตามโหลดอย่างรวดเร็ว

<u>การจัดสรรภาระผ่านการควบคุมกำลังและความถี่ (P-f Control)</u>

ตัวควบคุมไมโครกริดต้องทำให้แน่ใจว่าเกิดความเรียบและเปลี่ยนแปลงโดยอัตโนมัติจากโหมดเชื่อมต่อกริดเป็นโหลดแยกอิสระ ซึ่งหลักการนี้จะมีความคล้ายกับการดำเนินการของระบบสำรองไฟฟ้า (Uninterrupt Power Supply : UPS) โดยในขณะที่เกิดการเปลี่ยนแปลงเป็นโหมดแยกอิสระนั้น MC ของแต่ละแหล่งจ่าย จะใช้การควบคุมกำลังและความถี่ เพื่อเปลี่ยนจุดที่ดำเนินการ เพื่อให้เกิดความสมดุลของกำลังไฟฟ้าเมื่อปรับเปลี่ยนโหลดใหม่ ตัวควบคุมจะทำการดำเนินงานด้วยตนเองหลังจากการติดตามโหลดโดยไม่ต้องรอคำสั่งจาก CC หรือ MC ตัวอื่น รูปที่ 2.4 แสดงคุณลักษณะของกำลังและความถี่ที่ใช้ใน MC สำหรับการควบคุมกำลังและความถี่ (P-f Control)



รูปที่ 2.4 คุณลักษณะดรูปของกำลังจริงเทียบกับความถี่ [1]

ในโหมดแยกอิสระโหลดของไมโครกริดจะได้รับพลังงานจากทั้งกริดหลักและจากแหล่งจ่ายขึ้นอยู่กับความต้องการของผู้บริโภค เมื่อแหล่งจ่ายจากกริดหลักเกิดการหยุดชะงัก (interrupt) เนื่องจากสาเหตุต่างๆ ไมโครกริดจะสับเปลี่ยนเป็นโหมดแยกอิสระ ในขณะที่เกิดการเปลี่ยนแปลงโหมดการทำงานมุมเฟสของแรงดันที่แหล่งจ่ายจะเปลี่ยนแปลงตาม นำไปสู่เกิดการตกของกำลังไฟฟ้าขาออกได้อย่างชัดเจน ทำให้ความถี่เปลี่ยนแปลง โดยในกรณีนี้แหล่งจ่ายแต่ละตัวจะมีการจัดสรรภาระเสียใหม่โดยไม่มีการส่งกำลังไฟฟ้ามาจาก CC ยกตัวอย่างเช่น กำหนดให้ แหล่งจ่าย 2 แหล่งดำเนินการที่ความถี่ปกติต่ำสุดที่ความจุ P_{1max} และ P_{2max} ในโหมดเชื่อมต่อกับกริด นั้นจะทำงานที่ความถี่พื้นฐานในการส่งกำลัง P_{01} และ P_{02} ตามลำดับ เมื่อความต้องการโหลดเปลี่ยนไปแหล่งจ่ายจะทำงานที่ความถี่ที่แตกต่างอันเป็นเหตุมาจากการเปลี่ยนแปลงของมุมกำลังและความถี่ในการทำงานที่ลดลงจากปกติกับสัดส่วนความแตกต่างของภาระ ซึ่งจะเกิดเป็นคุณลักษณะของกำลังกับความถี่ (P-f) ดังรูปที่ 2.4 เมื่อความถี่ในไมโครกริดลดลงแล้ว ดังนั้น MC ก็ต้องรวบรวมฟังก์ชันเพื่อฟื้นฟูการทำงานที่ความถี่พิกัดกับการแบ่งสัดส่วนโหลดที่เหมาะสม

(2) ตัวควบคุมส่วนกลาง (Central Controller : CC)

CC มีอำนาจควบคุมผ่านโมดูลพื้นฐานสองอย่างได้แก่ โมดูลการจัดการพลังงาน (Energy Management Module) และโมดูลป้องกันในการทำงานร่วมกัน (Protection Co-ordination Module)

โมดูลการจัดการพลังงาน

โมดูลการจัดการพลังงาน (Energy Management Mudule : EMM) ประกอบด้วยฟังก์ชันการควบคุมต่างๆ สำหรับการควบคุมพลังงานในไมโครกริดได้อย่างเหมาะสม ในส่วนนี้จะอธิบายเกี่ยวกับพื้นฐานของ EMM ที่รวบรวมเอาพื้นฐานฟังก์ชันการควบคุมที่จำเป็นสำหรับ

การทำงานที่น่าพึงพอใจของไมโครกริด จำนวนฟังก์ชันการควบคุมสามารถเพิ่มได้ เพื่อให้บรรลุถึงรายละเอียดย่อยและควบคุมได้ซับซ้อน

ฟังก์ชันพื้นฐานในการควบคุมแหล่งจ่าย

EMM แบบพื้นฐานจะทำการหาจุดอ้างอิงของกำลังไฟฟ้าจริงและแรงดันของ MC ในขณะที่พื้นฐานควบคุมแหล่งจ่ายจะทำงานผ่าน MC เท่านั้น ค่าแรงดันอ้างอิงจะถูกเก็บในแถบที่ตั้งไว้เพื่อให้ได้แรงดันไฟฟ้าที่เหมาะสมในการรักษาระดับแรงดันในไมโครกริด

การควบคุมแรงดัน (Voltage control)

โดยปกติ โหลดและตัวประกอบกำลังในไมโครกริดจะถูกควบคุมโดยการเปลี่ยนขนาดแรงดันและมุมเฟสของแหล่งจ่ายเพื่อหลีกเลี่ยงความซับซ้อนในการควบคุม EMM การควบคุมแรงดันและตัวประกอบกำลังของแหล่งจ่ายจะถูกบังคับผ่าน MC และไม่ผ่าน EMM ซึ่ง EMM จะทำเพียงแค่หาค่าแรงดันอ้างอิงไปยัง MC สำหรับบัสที่สำคัญบางบัสของไมโครกริด เมื่อสายป้อนในการจำหน่ายไม่ได้จ่ายโหลดเต็มพิกัดในไมโครกริด อาจมีแนวโน้มแรงดันเพิ่มขึ้นที่สายป้อนในการหยุดการเพิ่มของแรงดันนั้น MC จะต้องคอยตรวจสอบแรงดันและจัดการป้อนค่ากลับไปที่ EMM จากนั้น EMM จะนำค่าแรงดันอ้างอิงไปยัง MC เพื่อปรับให้ได้แรงดันที่ต้องการ เป้าหมายของวิธีการควบคุมเช่นนี้คือ เพื่อควบคุมให้ภาระทำงานที่เป็นค่าตัวประกอบกำลังไฟฟ้าเท่ากับ 1 (unity p.f.)

การควบคุมตัวประกอบกำลัง (Power factor control)

จะไม่เหมือนกับเครื่องกำเนิดไฟฟ้าซิงโครนัสทั่วไปแหล่งจ่ายมักจะไม่มีกาควบคุมตัวประกอบกำลังภายในตัวมันเอง ดังนั้นตัวประกอบกำลังจึงขึ้นอยู่กับโหลด MC ทุกตัวจะมีการควบุคมค่าตัวประกอบกำลัง พร้อมคุณสมบัติฟังก์ชันการติดตามโหลด อย่างไรก็ตามสามารถนำเอาอิเลคทรอนิกส์กำลังของแหล่งจ่ายบางตัวซึ่งอาจประกอบด้วยการควบคุมตัวประกอบกำลังมาควบคุมมุมเฟสของกระแสและลดฮาร์มอนิกส์ให้น้อยที่สุด คุณลักษณะการควบคุมตัวประกอบกำลังจะถูกจัดรวมไว้ใน MC ดังเช่นนั้นแล้ว จึงไม่ปรากฏคำสั่งใดๆ จาก EMM อีก ยกเว้นค่าแรงดันอ้างอิง

การควบคุมความเร็วต้นกำลัง (Prime mover speed control)

คุณลักษณะนี้จะมีสำหรับแหล่งจ่ายที่มีการหมุนด้วยต้นกำลังเช่น กังหันแก๊สขนาดเล็ก (Microturbine) หรือ กังหันลม เพื่อรองรับการเปลี่ยนแปลงโหลดของไมโครกริด ต้นกำลังของแหล่งจ่ายจะต้องปรับความเร็วให้สมดุลกับภาวะโหลดใหม่ ในที่นี่ต้นกำลังที่ความเร็วคงที่ควรจะเปลี่ยนเชื้อเพลิงที่ป้อนเข้าไป ซึ่งจะส่งผลต่อประสิทธิภาพของต้นกำลังโดยประสิทธิภาพจะเป็นฟังก์ชันของเชื้อเพลิงที่ใช้และความเร็ว ดังนั้น การควบคุมความเร็วของต้นกำลังควรทำให้เกิด

ความมั่นใจว่าจะทำให้แหล่งจ่ายเกิดประสิทธิภาพสูงสุด ในการออกแบบเบื้องต้น การควบคุมนี้จะถูกบังคับผ่าน MC

การรักษาระดับแรงดัน (Frequency regulation)

ในระบบไฟฟ้ากำลังทั่วไป ความถี่ของแรงดันที่ผลิตออกมาจะขึ้นกับความเร็วของเครื่องกำเนิดไฟฟ้าซิงโครนัส ในทางกลับกัน สำหรับไมโครกริดนั้น สามารถผลิตกำลังได้ตามต้องการด้วยการช่วยจากระบบคอนเวอร์เตอร์อิเลคทรอนิกส์ (Power electronic converter) ของ MC ของมันเอง ในโหมดเชื่อต่อกับกริด MC ไม่จำเป็นต้องสั่งการควบคุมกำลังและความถี่ (P-f control) ผ่านคุณลักษณะดรูปของกำลังและความถี่เพราะการเปลี่ยนความถี่จะสามารถดูแลตัวมันเองได้ แต่อย่างไรก็ตาม ในโหมดแยกอิสระ MC จำเป็นต้องสั่งการควบคุมกำลังและความถี่เพื่อรองรับการเปลี่ยนแปลงโหลดในระบบความถี่คงที่ สำหรับทั้งสองโหมดการทำงาน EMM จะไม่รบกวนคุณลักษณะการควบคุมของ MC อย่างไรก็ตาม EMM ก็ยังคอยตรวจสอบความถี่ในไมโครกริดตลอด และเมื่อใดที่ความถี่ตกและไม่ได้รับการฟื้นฟูจาก MC ภายในระยะเวลาที่กำหนด EMM จะทำการปลดโหลดอย่างรวดเร็วในยามฉุกเฉิน เพื่อให้เกิดความสมดุลของกำลังไฟฟ้าและทำให้เกิดเสถียรภาพในไมโครกริด

การดำเนินการของ EMM ในไมโครกริดทั่วไป

เพื่อให้ง่ายต่อการควบคุมไมโครกริด จำนวนฟังก์ชันในการควบคุมใน EMM จะถูกจำกัดเพียงแค่ส่วนที่เป็นพื้นฐานเท่านั้น ทำให้มีสัญญาณป้อนกลับที่ต้องการน้อยสุดโดย EMM จาก MC เพื่อส่งคำสั่งที่สำคัญไปยังแหล่งจ่ายวิธีการดำเนินการจะอธิบายดังต่อไปนี้

การทำงานในโหมดเชื่อมต่อกับกริด (Grid-connected operation)

ในโหมดเชื่อมต่อกับกริดสัญญาณควบคุมของ EMM จะถูกจำกัดไปยังกำลังไฟฟ้าจริงและค่าแรงดันอ้างอิงของแหล่งจ่าย การควบคุมแรงดันและตัวประกอบกำลังจะถูกสั่งการโดย MC เพื่อให้ไมโครกริดสามารถใช้ประโยชน์ในการควบคุมให้โหลดทำงานที่ค่าตัวประกอบกำลังไฟฟ้าเท่ากับ 1 (unity p.f.) ดังนั้น EMM จึงไม่ได้สั่งการควบคุมแรงดันเพิ่มเติมและการขนานคาปาซิเตอร์ของกริดหลักและ MC ในไมโครกริดแต่อย่างใด ถ้าสายป้อนในการจำหน่ายรับภาระโหลดน้อยๆ และอาจทำให้เกิดแรงดันเพิ่มขึ้นจะถูกจัดการโดยตัวควบคุมกริดหลัก(utility controller) อย่างไรก็ตาม EMM จะสั่งการควบคุมแรงดันของแหล่งจ่ายเพียงบางบัสที่สำคัญของไมโครกริดเท่านั้น

การทำงานในโหมดแยกอิสระ (Stand-alone operation)

ในโหมดแยกอิสระนั้น ฟังก์ชันหลักของ EMM คือหาค่ากำลังไฟฟ้าจริงและค่าแรงดันอ้างอิงของ MC ความถี่และการไหลของกำลังไฟฟ้าเสมือนจะถูกควบคุมโดย MC ผ่าน

คุณลักษณะดรูปของกำลังจริง-ความถี่ และ กำลังเสมือน-แรงดัน (P-f และ Q-V droop characteristic) EMM จะไม่ติดต่อกับสัญญาณคำสั่งสำหรับควบคุมมุมเฟสและความถี่ไปยัง MC แต่จะคอยตรวจจับความถี่ของไมโครกริดอย่างต่อเนื่องและทำการปลดโหลดอย่างรวดเร็วผ่าน MC กรณีที่ความถี่ไม่ได้รับการฟื้นฟูภายในเวลาที่กำหนด สำหรับระบบที่มั่นใจในเสถียรภาพ สำหรับการดำเนินการแบบแยกอิสระนั้นฟังก์ชันควบคุมจะตอบสนองให้เกิดความสมดุลของโหลดอย่างรวดเร็ว ทั้งนี้เป็นสาเหตุในส่วนของความจุในการผลิต ไมโครกริดที่แยกออกไปจะไม่มีความเสถียรเท่ากับการทำงานในโหมดที่เชื่อมต่อกับกริด

ในการจัดการพลังงานอย่างเหมาะสมเพื่อประสิทธิภาพสูงสุดนั้น เมื่อไมโครกริดหลายแห่งต้องการกำลังไฟฟ้าส่วนกลางขนาดใหญ่ พวกมันจำเป็นที่จะต้องเชื่อมต่อกัน ในกรณีนี้ EMM ของไมโครกริดอื่นควรนำการควบคุมสำหรับบรรลุเป้าหมายการเพิ่มประสิทธิภาพพลังงานขั้นพื้นฐานในระบบเชื่อมโยงทั้งหมด เพื่อที่จะได้จุดที่ทำให้เกิดประสิทธิภาพสูงสุดของแหล่งจ่ายนั้น EMM ควรจะเดินเครื่องแหล่งจ่ายในจำนวนที่เหมาะสม (โดนเฉพาะกังหันแก๊สที่เป็น microturbine) ให้เข้าใกล้ค่าพิกัดขณะโหลดน้อยๆ แทนที่จะเดินเครื่องทั้งหมดแล้วแยกกันจ่าย การควบคุมแบบนี้จะทำให้ประสบผลสำเร็จสูงสุดโดย EMM เหตุเนื่องมาจากการรับรู้เข้าใจล่วงหน้าของเงื่อนไขกระบวนการ ตัวแปรทางภูมิอากาศ กำหนดการผลิตของแหล่งจ่ายและข้อมูลของเชื้อเพลิง เช่น ราคา และรูปแบบการบริโภค

โมดูลป้องกันระบบในการทำงานร่วมกัน

โมดูลป้องกันระบบในการทำงานร่วมกัน (Protection co-ordination module : PCM) จะกำกับดูแลการป้องกันโดยรวมของไมโครกริด หลักการในการป้องกันของไมโครกริด จะแตกต่างจากระบบระบบโครงข่ายระบบจำหน่ายแบบทั่วไปด้วยวิธีระบบตามแนวรัศมี(radial system) เหตุผลของความแตกต่าง มีดังนี้

1.) ไมโครกริดมีทั้งเครื่องกำเนิดไฟฟ้า และ ภาระที่การไหลของกำลังไฟฟ้ามีทิศการไหลทั้งสองทางผ่านอุปกรณ์ป้องกันในระบบการจำหน่ายตามแนวรัศมี (radial system)

2.) โครงข่ายระบบจำหน่ายแบบแพสซีฟ จะเปลี่ยนเป็นแบบแอคทีฟ เนื่องจากการประพฤติตัวของแหล่งจ่ายขนาดเล็ก

3.) ไมโครกริดจะประสบกับการเปลี่ยนแปลงที่สำคัญ ในทางความสามารถในการลัดวงจร เมื่อมีการเปลี่ยนจากโหมดเชื่อมต่อกับกริดไปเป็นโหมดแยกอิสระ ทำให้เกิดผลกระทบในเชิงลึกในรีเลย์กระแสเกิน (overcurrent relay) ทั่วไป ที่จะดำเนินการตรวจจับกระแสลัดวงจร

คุณลักษณะที่สำคัญของ PCM คือ ความสามารถในการแยกความแตกต่างระหว่างความต้องการในการป้องกันสำหรับทั้งสองโหมดการทำงาน และระบุเหตุที่เกิดอย่างสอดคล้องกัน

การป้องกันในโหมดเชื่อมต่อกับกริด

ในโหมดเชื่อมต่อกับกริดนั้น PCM จะตรวจจับและกระทำใน 5 เหตุการณ์ที่จะกล่าวต่อไปนี้ PCM จะคิดคำนึงถึงเวลาตอบสนองของแต่ละแหล่งจ่ายตลอดจนจุดต่อร่วม (Point of common coupling : PCC)

- สภาวะปกติ (Normal Condition)
- สภาวะเกิดความผิดพร่องในสายป้อนของไมโครกริด (Microgrid feeder fault)
- สภาวะเกิดความผิดพร่องบนกริดหลัก (Utility fault)
- สภาวะเกิดความผิดพร่องบนบัสของไมโครกริด(Microgrid bus fault)
- สภาวะการเกิดรีซิงโครไนซ์เซชั่น (Re-synchronisation)

การป้องกันในโหมดแยกอิสระ

เมื่อไมโครกริดดำเนินการในโหมดแยกอิสระระดับการลัดวงจรที่บัสของไมโครกริดจะลงลดอย่างเห็นได้ชัด ทั้งนี้เป็นเพราะแหล่งจ่ายที่มีระบบคอนเวอร์เตอร์อิเลคทรอนิกส์ (power electronics converter) อาจลดการจ่ายกระแสฟอลต์ขึ้นเป็น 200% ของกระแสโหลด ดังนั้นไมโครกริดในโหมดแยกอิสระจะมีกระแสฟอลต์ที่ต่ำกว่ามากเมื่อเทียบกับในโหมดการเชื่อมต่อกับกริด กระแสฟอลต์ที่ต่ำอาจไม่สามารถตรวจจับด้วยรีเลย์กระแสเกินที่ใช้ในระบบป้องกันดั้งเดิม เพราะฉะนั้นจึงเป็นผลกระทบที่สำคัญในเรื่องของการตรวจจับฟอลต์ของรีเลย์ป้องกันของไมโครกริดที่ตรวจจับด้วยเซอร์ตรวจจับกระแสฟอลต์ (fault current sensing) รีเลย์พื้นฐานอาจใช้เวลานานในการตรวจจับกระแสฟอลต์ หรืออาจตรวจจับไม่ได้เลย ดังนั้นจึงมีวิธีอื่นในการป้องกัน เช่น impedance protection, differential current/voltage relaying, zero sequence current/voltage relaying, directional overcurrent/earth-fault อาจถูกมาปรับใช้ในไมโครกริดในโหมดแยกอิสระ

2.1.7 การเชื่อมต่อของไมโครกริด

เนื่องจากไมโครกริดถูกออกแบบเพื่อผลิตพลังงานในระบบการจำหน่ายพร้อมกับการใช้ประโยชน์จากความร้อนเหลือทิ้งจึงทำให้เกิดการควบคุมพิกัดการส่งจ่ายพลังงาน ซึ่งความจุสูงสุดของไมโครกริดจะถูกจำกัดโดยปกติที่ประมาณ 10 MVA จากข้อเสนอแนะตาม IEEE ดังนั้นถ้าจะจ่ายภาระทางไฟฟ้าขนาดใหญ่แล้วเราสามารถทำได้โดยการรวมเอาการผลิตจากหลายๆ ไมโครกริดผ่านทางเครือข่ายระบบจำหน่ายกลางได้โดยการแยกกลุ่มของภาระไฟฟ้าภายหน่วยของโหลดที่สามารถควบคุมได้ คู่กับแต่ละหน่วยการผลิตซึ่งรองรับการผลิตด้วยระบบของไมโครกริด ด้วยวิธีนี้ไมโครกริดจึงสามารถเชื่อมโยงไปยังกลุ่มพลังงานที่มีขนาดใหญ่กว่าเพื่อตอบสนองความต้องการไฟฟ้าที่มีจำนวนมาก สำหรับ การเชื่อมต่อของไมโครกริดนั้น CC แต่จะต้องดำเนินการในรูปแบบประสานกันกับ CC ที่อยู่ใกล้เคียง ดังนั้นจึงเชื่อมโยงไมโครกริด

จะทำให้มีเสถียรภาพและควบคุมดีขึ้นได้โดยการจัดการโครงสร้าง นอกจากนี้ยังจะมีความซ้ำซ้อนมากขึ้นเพื่อให้มีความเชื่อมั่นที่ดีขึ้น

ด้วยหลักการดังกล่าว การประสานงานและการสื่อสารแลกเปลี่ยนข้อมูลกันระหว่างหน่วยการผลิตและหน่วยผู้บริโภคจึงเป็นสิ่งสำคัญ

2.1.8 ข้อเสียของการพัฒนาไมโครกริด

แม้การพัฒนาใช้ระบบไมโครกริดจะดูเหมือนมีคุณประโยชน์แต่ก็ยังมีข้อเสียอยู่หลายประการ ในที่นี้จะขอกล่าวถึงข้อเสียเชิงเทคนิคของระบบเท่านั้น ประเด็นแรกจะเป็นปัญหาทางการขาดประสบการณ์ทางเทคนิคในการควบคุมแหล่งจ่ายจำนวนมาก ในแง่มุมนี้ต้องการขยายเวลาและปิดระบบ (off-line) เพื่อทำการการวิจัยการจัดการ ด้านการป้องกันและควบคุมไมโครกริดรวมถึงในการเลือกขนาดและตำแหน่งของแหล่งจ่าย โดยเฉพาะโครงสร้างพื้นฐานด้านโทรคมนาคมและโปรโตคอลการสื่อสารจะต้องพัฒนาในพื้นที่นี้ การวิจัยเป็นไปในการดำเนินการและการนำเสนอของมาตรฐาน IEC 61850 ในการสื่อสารสำหรับไมโครกริดและเครือข่ายการกระจายระบบอย่างไรก็ตามในชนบทขาดโครงสร้างพื้นฐานการสื่อสารที่เหมาะสมเป็นอุปสรรคที่ในการดำเนินการไมโครกริด และอีกประเด็นหนึ่งคือการขาดมาตรฐาน เนื่องจากไมโครกริดเป็นแนวคิดแบบใหม่ มาตรฐานยังไม่พร้อมที่จัดการการดำเนินงานและปัญหาด้านการป้องกัน ควรจะประกาศข้อมูลคุณภาพกำลังสำหรับประเภทของแหล่งจ่าย มาตรฐานและโปรโตคอลสำหรับการรวมแหล่งจ่ายรวมถึงการลดระดับกำลังการผลิต (deregulated power) และแนวทางการป้องกันความปลอดภัยและอื่นๆ

2.1.9 การจัดการและปัญหาการดำเนินงานของไมโครกริด

การจัดการและปัญหาการดำเนินงานที่เกี่ยวข้องกับไมโครกริด มีดังนี้

1.) สำหรับการบำรุงรักษาคุณภาพกำลังจะต้องมีการสมดุลกำลังไฟฟ้าจริงและกำลังไฟฟ้าเสมือนภายในไมโครกริดบนพื้นฐานระยะสั้น

2.) ไมโครกริดควรทำงานในโหมดแยกอิสระบนพื้นที่ซึ่งไม่สามารถจัดหาสาธารณูปโภคหรือในโหมดการเชื่อมต่อกับกริด ภายในเครือข่ายกระจายสาธารณูปโภคขนาดใหญ่ การดำเนินการของไมโครกริดควรจะสามารถเลือกโหมดการดำเนินการภายใต้กรอบระเบียบที่เหมาะสม

3.) การสร้างแหล่งจ่ายและแหล่งสำรองพลังงานต้องมีการวางแผนอย่างเหมาะสมตามความต้องการโหลดในไมโครกริดรวมถึงการสมดุลพลังงานในระยะยาว

4.) การควบคุมกำกับดูแลและจัดหาข้อมูล (SCADA) พื้นฐาน ฟังก์ชั่นรการควบคุมและป้องกันควรจะรวมอยู่ใน CC และ MC ของไมโครกริด รวมถึงข้อกำหนดจะต้องได้รับการบัญญัติขึ้นเพื่อการวิเคราะห์ระบบผ่านฟังก์ชั่นการประเมินสถานะของระบบในขณะนั้น

5.) ระบบรักษาความปลอดภัยจะต้องรักษาด้วยการวิเคราะห์ฉุกเฉินและการดำเนินงานฉุกเฉิน (เช่นโหลดความต้องการด้านการจัดการ, โหลดแสงสว่าง, การแยกตัวทำงานอย่างอิสระหรือ การดับของหน่วยการผลิตใดๆ) และการจัดสรรค์การผลิตควรจะทำการดูแลระบบของภาระไฟฟ้าและแรงดันไฟฟ้า/ ความถี่ที่บัสปลายด้วย

6.) ความคาดเคลื่อนชั่วขณะระหว่างการผลิตกับปริมาณความต้องการจะถูกทำให้ลดลงด้วยการทำนายโหลดที่เหมาะสมและการจัดการด้านความต้องการ การเลื่อนของโหลดจะช่วยให้กราฟโหลดราบเรียบเพื่อลดพิกัดการสำรองพลังงาน

7.) โครงสร้างพื้นฐานที่เหมาะสมของระบบสื่อสารและการสื่อสารโปรโตคอลต้องได้รับการนำมาใช้ในทุกการจัดการพลังงานรวมถึงการควบคุมและการป้องกันระบบซึ่งมาตรฐานการสื่อสาร IEC 61850 ที่จะใช้เป็นโครงสร้างการสื่อสารพื้นฐานเป็นส่วนใหญ่จะถูกนำมาใช้ในการอ้างอิง

2.1.10 แหล่งพลังงานในไมโครกริด

Resouces in microgrid

พลังงานทดแทนหรือแหล่งกำเนิดไฟฟ้าแบบใหม่ๆ ได้ถูกนำมาใช้ในไมโครกริดซึ่งถูกเรียกว่า แหล่งกำเนิดไฟฟ้า ณ จุดใช้งาน (distributed energy resource : DER) หรือ แหล่งจ่ายขนาดเล็ก (microsources) มีวัตถุประสงค์เพื่อรวมนำข้อดีของแหล่งกำเนิดพลังงานที่ทำให้คาร์บอนไดออกไซด์ต่ำและแหล่งกำเนิดพลังงานไฟฟ้าร่วมกับความร้อน (Combined Heat and Power : CHP) ที่มีประสิทธิภาพสูง

DER ที่ใช้ในไมโครกริดมีดังนี้

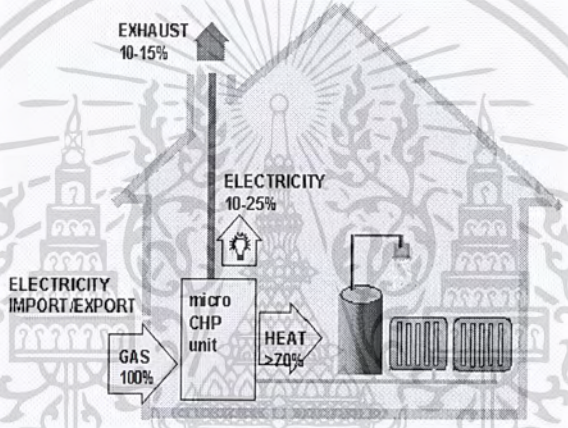
- Combined Heat and Power
- Wind Energy Conversion
- Solar Photovoltaic
- Small-scale hydroelectric
- Storage Device

Combined Heat and Power (CHP)

CHP นั้นเป็นแหล่งผลิตที่เป็นการผลิตพลังงานร่วม (cogeneration) ที่ผลิตทั้งพลังงานไฟฟ้าและพลังงานความร้อน ซึ่งมีแนวโน้มอย่างมากที่จะนำมาใช้กับไมโครกริด ข้อดีก็คือมีประสิทธิภาพทางพลังงานโดยการใช้ความร้อนทิ้งที่ได้จากครัวเรือนหรือกระบวนการทางอุตสาหกรรมนอกจากนี้ยังสามารถใช้ในการชิลเลอร์สำหรับระบายความร้อนการดูดซึม พร้อมกันของการผลิตไฟฟ้าความร้อนและระบายความร้อนเป็นที่รู้จักกันเป็น trigeneration หรือ polygeneration ระบบ CHP ช่วยให้การใช้พลังงานที่ดีขึ้นกว่ารุ่นเดิมที่อาจถึงประสิทธิภาพมากกว่า 80% เมื่อเทียบกับที่ประมาณ 35% สำหรับโรงไฟฟ้าแบบเดิม

สำหรับการผลิตความร้อนร่วมขนาดเล็ก (Micro – CHP) จะถูกติดตั้งในสถานที่มีขนาดเล็กมักจะชอบบ้านหรือขนาดเล็กอาคาร ส่วนใหญ่หน่วย CHP ในอุตสาหกรรมการผลิตไฟฟ้าเป็นผลิตภัณฑ์หลักด้วยความร้อนเป็นที่สองในขณะที่ระบบ Micro CHP สร้างความร้อนเป็นหลักกับไฟฟ้าเป็นผลพลอยได้ Micro - CHP มีความเชื่อถือได้ แข็งแกร่งและราคาถูก มีกำลังการผลิตในช่วง 10-100 กิโลวัตต์

Micro CHP สามารถพัฒนาจากเทคโนโลยีหลายเทคโนโลยี เช่น เครื่องยนต์สเตอร์ลิง (Stirling engines) Reciprocating engines (หรือ เครื่องยนต์สันดาปภายใน, Internal Combustion engine) และ เซลล์เชื้อเพลิง (Fuel Cell) โดยที่เทคโนโลยีที่กล่าวถึงจะทำหน้าที่ในการผลิตกระแสไฟฟ้าและนำความร้อนเหลือใช้ในการผลิตกระแสไฟฟ้าไปใช้ในการทำความร้อน และถ้าจะผลิตความเย็นก็จะใช้ความร้อนที่ได้นี้ไปเป็นพลังงานสำหรับเครื่องทำความเย็นแบบดูดซึมเพื่อผลิตความเย็นต่อไป



รูปที่ 2.5 Micro-CHP

Wind Energy Conversion

เป็นการเปลี่ยนรูปพลังงานลมเป็นพลังงานไฟฟ้า โดยมีหลักการพื้นฐานโดยใช้กังหันลมต่อควบกับเครื่องกำเนิดไฟฟ้าผ่านระบบเกียร์ (gearbox) ซึ่งเครื่องกำเนิดไฟฟ้าที่ใช้จะเป็นเครื่องกำเนิดไฟฟ้าเหนี่ยวนำ กังหันลมจะจับพลังงานจลน์จากการไหลของลมผ่านใบพัดโรเตอร์และส่งพลังงานไปยังเครื่องกำเนิดไฟฟ้าเหนี่ยวนำผ่านระบบเกียร์เพลาของเครื่องกำเนิดไฟฟ้าจะถูกขับด้วยกังหันเปลี่ยนเป็นพลังงานไฟฟ้า สำหรับระบบเกียร์นั้นมีไว้เพื่อช่วยปรับระดับความเร็วของโรเตอร์ก่อนที่จะเข้าไปยังเครื่องกำเนิดไฟฟ้า

กังหันลมที่ใช้มีทั้งแบบลักษณะการหมุนแนวตั้ง (Herizontal axis) และหมุนในแนวนอน (Vertical axis) ขนาดกำลังการผลิตนั้นปัจจุบันสามารถผลิตได้สูงถึง 5 MW กำลังการผลิตนั้นมีปัจจัยขึ้นอยู่กับสภาพอากาศ โดยจะแปรผันตรงกับความเร็วลม ขนาดใบพัด และความหนาแน่นของอากาศ

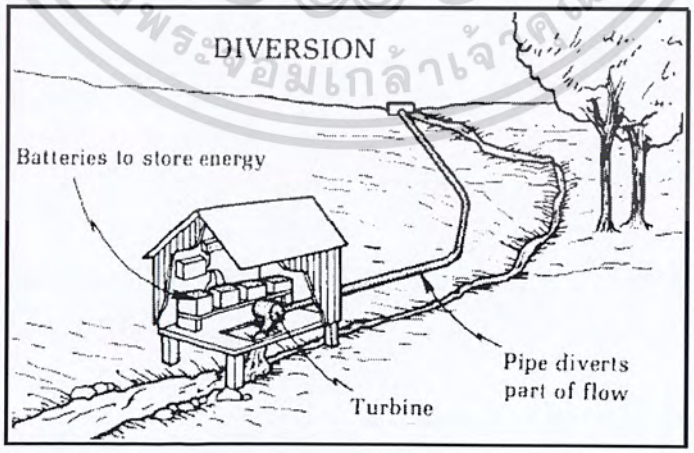
Solar Photovoltaic

พลังงานแสงอาทิตย์ เกิดจากปฏิกริยาฟิวชั่นของดวงอาทิตย์ จะปล่อยพลังงานออกมาในรูป คลื่นแม่เหล็กไฟฟ้า ที่เรียกว่า รังสีแสงอาทิตย์ (Solar Radiation) รังสีนี้จะแพร่กระจายออกทุกทิศทุกทาง โลกของเราก็ได้รับอิทธิพลของรังสีนี้ โดยมีความเข้มของรังสีที่ตกลงบนผิวโลกประมาณ 961-1,191 วัตต์ต่อตารางเมตร หรือคิดเป็นพลังงานประมาณ 2,000-2,500 กิโลวัตต์ชั่วโมงต่อตารางเมตรต่อปี

สำหรับประเทศไทย พื้นที่เกือบทั้งหมดสามารถรับพลังงานจากแสงอาทิตย์เฉลี่ยประมาณ 4.5 กิโลวัตต์-ชั่วโมง/ตารางเมตร/วัน ดังนั้นในพื้นที่ 1 ตางรางกิโลเมตร สามารถติดตั้งระบบผลิตไฟฟ้าจากเซลล์แสงอาทิตย์ขนาด 33 เมกะวัตต์ หรือ 165,000 กิโลวัตต์-ชั่วโมง/ตารางเมตร/วัน ในปัจจุบันความต้องการพลังงานไฟฟ้าของประเทศประมาณวันละ 250 ล้านกิโลวัตต์-ชั่วโมง ถ้าต้องการผลิตจากพลังงานแสงอาทิตย์ทั้งหมด จำเป็นต้องใช้พื้นที่ประมาณ 1,500 ตารางกิโลเมตร หรือคิดเป็นพื้นที่ประมาณ 0.3% ของประเทศเท่านั้น ในอดีตการผลิตไฟฟ้าจากเซลล์แสงอาทิตย์มีราคาแพงมาก แต่เนื่องจากปัจจุบันราคาของเซลล์แสงอาทิตย์ได้ลดลงมาอย่างมาก และมีแนวโน้มว่าจะลดลงอีกเรื่อย ๆ เพราะประชาชนโดยทั่วไปได้ตระหนักถึงสภาวะแวดล้อมเป็นพิษ เนื่องจากการใช้ เชื้อเพลิงบรรพชีวินในการผลิตพลังงาน จึงหันมาใช้เซลล์แสงอาทิตย์เพิ่มขึ้นเรื่อย ๆ

Small-scale hydroelectric

เป็นการกำเนิดไฟฟ้าโดยใช้กังหันน้ำขนาดเล็กในการขับเคลื่อนเครื่องกำเนิดไฟฟ้า โดยมีหลักการโดยใช้ประโยชน์จากการไหลของน้ำจากแหล่งน้ำขนาดเล็กในการขับเคลื่อนกันหันกำลังการผลิตนั้นแปรผันโดยตรงกับอัตราการไกลของน้ำ ความหนาแน่นของน้ำ และระดับความสูงของหัวน้ำ



รูปที่ 2.6 การกำเนิดไฟฟ้าจากพลังงานน้ำขนาดเล็ก

Storage Device

เป็นอุปกรณ์ในการเก็บสะสมพลังงานสำหรับจ่ายพลังงานฉุกเฉินในไมโครกริด ซึ่งสามารถจ่ายพลังงานในช่วงเวลาสั้นๆ อุปกรณ์ในการเก็บพลังงานมีดังนี้

- Storage Battery
- Flywheels
- Ultra Capacitor

2.2 ระบบมัลติเอเจนท์ (Multi-Agent System)

Object Management Group (OMG) ซึ่งเป็นองค์กรสากลที่ก่อตั้งขึ้นในปี ค.ศ.1989 ประกอบด้วยสมาชิกที่เป็นทั้งตัวแทนจำหน่ายระบบสารสนเทศ นักพัฒนาซอฟท์แวร์ และผู้ใช้งาน ได้เสนอทฤษฎีและแนวปฏิบัติสำหรับการพัฒนาซอฟต์แวร์ด้วยเทคโนโลยีเชิงวัตถุ ได้ให้คำจำกัดความของคำที่เกี่ยวข้องกับเอเจนท์ไว้ใน Mobile Agent Facility Specification Version 1.0 (OMG 2000) ไว้ดังนี้

เอเจนท์ (Agent) คือ โปรแกรมคอมพิวเตอร์ซึ่งทำงานอัตโนมัติแทนคนหรือองค์กร โดยแต่ละเอเจนท์จะมีเธรด (Thread) การประมวลผลของตนเองซึ่งสามารถเริ่มการทำงานเองได้ ทั้งนี้ OMG ได้แบ่งประเภทของเอเจนท์ไว้เป็น 2 ประเภท คือ

- เอเจนท์แบบสเตชันนารี (Stationary Agent)
- เอเจนท์แบบเคลื่อนที่ (Mobile Agent)

เอเจนท์แบบสเตชันนารี

เอเจนท์แบบสเตชันนารี คือ เอเจนท์ที่จะทำการประมวลผลได้เฉพาะบนระบบที่สร้างเอเจนท์นั้น หากเอเจนท์ต้องการติดต่อกับเอเจนท์ที่อยู่ในระบบอื่นก็จะต้องทำการติดต่อผ่านกลไกการติดต่อสื่อสารด้วยการเรียกใช้ส่วนการทำงานระยะไกล (Remote Procedure Call หรือ RPC)

เอเจนท์แบบเคลื่อนที่

เอเจนท์แบบเคลื่อนที่ คือ เอเจนท์ที่ไม่ถูกผูกติดกับระบบที่เอเจนท์เริ่มต้นทำการประมวลผล โดยสามารถเคลื่อนที่ตัวมันเองไปยังระบบอื่นในเครือข่ายได้ ความสามารถในการเคลื่อนที่นี้ทำให้เอเจนท์เคลื่อนย้ายไปยังระบบเอเจนท์ปลายทางที่มีวัตถุที่เอเจนท์ต้องการติดต่อด้วยได้ และเอเจนท์อาจใช้บริการของวัตถุนั้นได้

2.2.1 คุณสมบัติพื้นฐานของเอเจนท์

(Wooldridge and Jenning 1995) กล่าวถึงคุณสมบัติพื้นฐานของเอเจนท์ไว้ดังนี้

- Autonomy เอเจนท์สามารถทำงานได้อัตโนมัติโดยไม่จำเป็นต้องติดต่อโดยตรงกับผู้ใช้ตลอดเวลาและมีกลไกบางอย่างในการควบคุมการกระทำและสถานะของเอเจนท์
- Social ability เอเจนท์ติดต่อกับเอเจนท์อื่นผ่านภาษาในการติดต่อสื่อสารระหว่างเอเจนท์
- Reactivity เอเจนท์สามารถทำการตอบสนองเมื่อมีการเปลี่ยนแปลงเกิดขึ้น
- Pro–activeness เอเจนท์สามารถริเริ่มแสดงพฤติกรรมเพื่อทำงานตามเป้าหมาย

2.2.2 มาตรฐานของระบบเอเจนท์

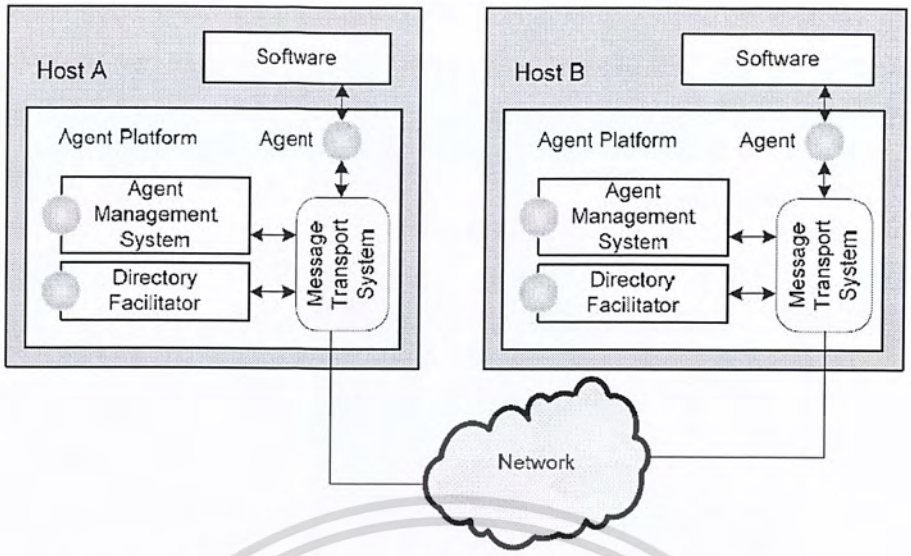
มาตรฐานของเอเจน์ที่ใช้อ้างอิงในการทำโครงงานในครั้งนี้ผู้ทำวิจัยได้อ้างอิงตามมาตรฐานของ FIPA ในการศึกษาระบบเอเจนท์

FIPA หรือ The Foundation For Intelligent Physical Agents ก่อตั้งขึ้นในปี ค.ศ. 1996 เพื่อสร้างมาตรฐานซอฟต์แวร์สำหรับระบบเอเจนท์ และในปีค.ศ. 2005 องค์กรมาตรฐาน IEEE Computer Society ได้มีมติรับ FIPA เป็นหนึ่งในคณะกรรมการมาตรฐานของ IEEE โดย FIPA ได้เสนอข้อกำหนด Foundation For Intelligent Physical Agents Specification (FIPA 2002) ซึ่งเป็นข้อกำหนดที่เกี่ยวข้องกับการติดต่อสื่อสารระหว่างเอเจนท์ การส่งข้อความระหว่างเอเจนท์ การจัดการเอเจนท์ แนวคิดสำหรับสถาปัตยกรรมและโปรแกรมประยุกต์สำหรับระบบเอเจนท์

2.2.3 แบบจำลองการจัดการเอเจนท์

Agent management Reference model

องค์ประกอบระบบเอเจนท์เคลื่อนที่เสนอในข้อกำหนดตามมาตรฐานของ FIPA แสดงด้วยแบบจำลองการจัดการเอเจนท์ (Agent Management Reference Model) ดังรูปที่ 2.7 ซึ่งแบบจำลองนี้ได้ถูกนำไปพัฒนาเป็นเอเจนท์แพลตฟอร์ม (Agent Platform :AP) หรือระบบเอเจนท์นั่นเอง



รูปที่ 2.7 แบบจำลองระบบเอเจนท์ตามมาตรฐาน FIPA (FIPA 2002) [2]

องค์ประกอบของระบบเอเจนท์ในแบบจำลองการจัดการเอเจนท์ตามมาตรฐาน FIPA ประกอบด้วย

Agent Platform (AP)

AP เป็นระบบเอเจนท์ที่มีโครงสร้างพื้นฐานทางกายภาพที่เอเจนท์สามารถทำงานได้ โดยระบบเอเจนท์ประกอบด้วยเครื่องคอมพิวเตอร์ ระบบปฏิบัติการซอฟท์แวร์สนับสนุนเอเจนท์ องค์ประกอบในการจัดการเอเจนท์และเอเจนท์ การออกแบบภายในระบบเอเจนท์เป็นประเด็นสำหรับผู้พัฒนาระบบเอเจนท์และไม่ได้กำหนดไว้เป็นมาตรฐานใน FIPA โดยเอเจนท์ที่อยู่ในระบบเอเจนท์เดียวกันไม่จำเป็นต้องอยู่บนคอมพิวเตอร์เครื่องเดียวกันก็ได้ ทั้งนี้แต่ละระบบเอเจนท์จะทำงานโดยใช้หนึ่ง Process และแต่ละเอเจนท์จะทำงานโดยใช้เธรด

Agent Management System (AMS)

AMS เป็นเอเจนท์ที่ทำการควบคุมการเข้าถึงและการใช้งานระบบเอเจนท์โดยภายในหนึ่งระบบเอเจนท์จะมีได้เพียงหนึ่ง AMS เท่านั้น หน้าที่ของ AMS ได้แก่ การเก็บรายการ Agent Identifiers (AID) ของเอเจนท์ที่ลงทะเบียนในระบบ การจัดการการดำเนินการเกี่ยวกับเอเจนท์ เช่น การสร้างเอเจนท์ การลบเอเจนท์และการย้ายเอเจนท์เข้ามาหรือออกจากระบบเอเจนท์อื่น การสอบถามรายละเอียดของระบบเอเจนท์อื่นก่อนที่จะดำเนินการย้ายเอเจนท์ และการดูแลวงจรชีวิตของเอเจนท์ที่ลงทะเบียนไว้กับระบบเอเจนท์

Directory Facilitator (DF)

DF เป็นเอเจนท์ที่ให้บริการไดเร็คทอรี (Directory) สำหรับการสืบค้นบริการของเอเจนท์ ทำหน้าที่ในการเก็บรักษารายการบริการของเอเจนท์และให้บริการข้อมูลเกี่ยวกับเอเจนท์ที่อยู่ในไดเร็คทอรีแก่เอเจนท์อย่างเท่าเทียมกันโดยเอเจนท์สามารถลงทะเบียนบริการของตนไว้กับ DF หรือสอบถาม DF ว่าเอเจนท์อื่นมีบริการอะไรบ้าง ซึ่งแต่ละระบบเอเจนท์อาจมีหลาย DF ร่วมกันทำงาน เอเจนท์ที่ต้องการให้บริการแก่เอเจนท์อื่นต้องลงทะเบียนรายละเอียดบริการของเอเจนท์ไว้กับ DF แต่เอเจนท์อาจจะปฏิเสธการให้บริการได้หลังจากที่ทำการลงทะเบียนไว้แล้ว ดังนั้น DF จึงไม่สามารถรับรองความเป็นปัจจุบันของข้อมูลที่ลงทะเบียนไว้ นอกจากนี้เอเจนท์อาจทำการถอนการลงทะเบียนที่ DF หรือแก้ไขรายละเอียดเอเจนท์ได้ในภายหลัง

Message Transport System (MTS)

MTS ทำหน้าที่ในการส่งข้อความระหว่างเอเจนท์โดยการรับส่งข้อความระหว่างเอเจนท์จะทำผ่าน MTS เท่านั้น

ดังนั้น Agent จึงเป็นโปรแกรมคอมพิวเตอร์ที่สามารถทำงานแบบอัตโนมัติและติดต่อสื่อสารกับโปรแกรมประยุกต์เพื่อทำงานตามที่กำหนด เอเจนท์ติดต่อสื่อสารกันโดยใช้ภาษาในการติดต่อสื่อสารของเอเจนท์ (Agent Communication Language หรือ ACL) และอาจมีบริการสำหรับให้เอเจนท์อื่นสามารถเรียกใช้งานได้ นอกจากนี้เอเจนท์ต้องมีเจ้าของเอเจนท์ (owner) ซึ่งอาจเป็นคนหรือองค์กร และมีชื่อเอเจนท์เพื่อใช้ในการอ้างถึง(Agent Identifier หรือ AID) และติดต่อกับเอเจนท์อื่น

2.2.4 การกำหนดชื่อเอเจนท์

เอเจนท์ต้องมีการกำหนดชื่อที่ไม่ซ้ำกันเพื่อให้สามารถถูกระบุหรือเรียกใช้งานได้เมื่อต้องการ เช่น เจ้าของเอเจนท์สามารถติดต่อหรือควบคุมเอเจนท์ได้เมื่อมีการส่งของ FIPA นั้นมีรูปแบบเป็นคู่ของพารามิเตอร์- ค่า (parameter-value pairs) ที่เรียกว่า Agent Identifier (AID) ซึ่ง AID นี้จะไม่สามารถแก้ไขได้ตลอดระยะเวลาที่เอเจนท์ยังมีชีวิตอยู่ พารามิเตอร์สำหรับการกำหนดชื่อเอเจนท์แสดงดังตารางที่ 2.1

ตารางที่ 2.1 พารามิเตอร์สำหรับการกำหนดชื่อเอเจนท์ (FIPA 2002) [2]

พารามิเตอร์	คำอธิบาย
name	ค่าบ่งบอกที่ไม่ซ้ำ (Unique identifier)ซึ่งสามารถอ้างอิงได้ กลไกที่ง่ายที่สุดคือสร้างจากชื่อจริง (actual name) ของเอเจนท์และ home agent platform (HAP) address หรือที่อยู่ของเอเจนท์แพลตฟอร์มที่สร้างเอเจนท์นั้นโดยแยกด้วยอักขระ @ เช่น Name agent-b@bar.com
addresses	รายการของที่อยู่ทางกายภาพที่เอเจนท์สามารถใช้ในการติดต่อสื่อสารกัน (transport addresses) เช่น addresses (sequence iiop://bar.com/acc)
resolvers	รายการของ name resolution service addresses เช่น (sequence (agent-identifier :name ams@foo.com :addresses(sequence iiop://foo.com/acc)))

ตัวอย่าง AID ที่สอดคล้องกับตาราง 2.1 เช่น

```
(agent-identifier
   :name agent-b@bar.com
   addresses (sequence iiop://bar.com/acc)
   :resolvers (sequence
      (agent-identifier
       :name ams@foo.com
       :addresses (sequence iiop://foo.com/acc))))
```

2.2.5 กระบวนการใช้ชื่อโดเมนสืบค้น

Name resolution เป็นกลไกที่ระบบใช้เพื่อหาตำแหน่งปัจจุบันของวัตถุ (Karnik and Tripathi 1998)โดยในระบบเอเจนท์แบบเคลื่อนที่นั้น name resolution เป็นบริการของ AMS ซึ่งเอเจนท์อื่นสามารถร้องขอให้ AMS ค้นหา ตัวอย่างรูปแบบของ name resolution เป็นดังนี้

1) เอเจนท์ a ต้องการส่งข้อความให้เอเจนท์ b ซึ่ง AID เป็นดังนี้

```
(agent-identifier
      :name agent-b@bar.com
      :resolvers (sequence
      (agent-identifier
            :name ams@foo.com
            :addresses(sequence iiop://foo.com/acc))))
```

และเอเจนท์ a ต้องการรู้ที่อยู่ของเอเจนท์ b

2) ดังนั้นเอเจนท์ a สามารถส่งการร้องขอการค้นหาไปยังเอเจนท์แรกในพารามิเตอร์ resolvers ซึ่งปกติคือ AMS ซึ่งในตัวอย่างนี้คือ AMS ที่อยู่ใน foo.com

3) ถ้า AMS ที่ foo.com มีเอเจนท์ b ลงทะเบียนอยู่ก็จะส่งข้อความที่มีที่อยู่เอเจนท์ b กลับไปให้เอเจนท์ a แต่ถ้าไม่มีเอเจนท์ b ลงทะเบียนอยู่ก็จะส่งข้อความกลับไปให้เอเจนท์ a ว่าไม่มี

4) เมื่อเอเจนท์ a ได้รับข้อความกลับมา ก็จะแยกข้อความออกมาซึ่งจะทำให้ทราบที่อยู่ของเอเจนท์ b

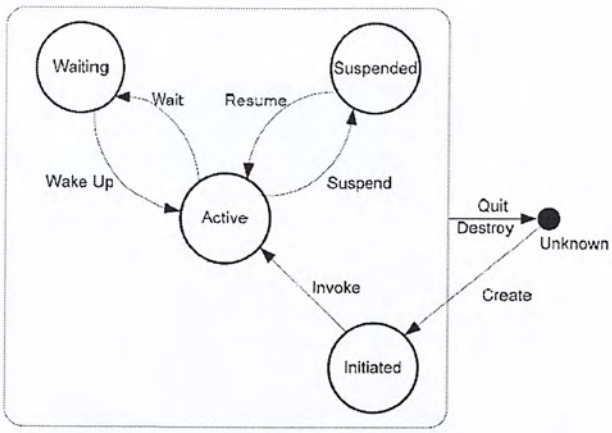
5) เอเจนท์ a สามารถส่งข้อความให้แก่เอเจนท์ b ได้โดยการเพิ่มพารามิเตอร์ address ใน AID ของเอเจนท์ b

2.2.6 วงจรชีวิตของเอเจนท์

เมื่อเอเจนท์ถูกสร้างขึ้นมาอยู่ในระบบเอเจนท์และใช้บริการต่างๆของระบบเอเจนท์ ทั้งนี้เอเจนท์ซึ่งเป็นโปรเซสของซอร์ฟแวร์จะมีวงจรชีวิตที่ถูกจัดการโดยระบบเอเจนท์ตามกฎต่อไปนี้

- **AP Bounded** : เอเจนท์จะถูกผูกติดกับระบบเอเจนท์ โดยเอเจนท์จะถูกจัดการทางกายภาพภายในระบบเอเจนท์หนึ่งเสมอ และวงจรชีวิตของเอเจนท์จะผูกติดกับระบบเอเจนท์นั้น
- **Application Independent** : แบบจำลองวงจรชีวิตของเอเจนท์(agent life cycle model) เป็นอิสระจากระบบโปรแกรมประยุกต์และถูกมองเห็นเฉพาะสถานะและการเปลี่ยนสถานะของเอเจนท์ในวงจรชีวิตของเอเจนท์เท่านั้น
- **Instance-Oriented** : เอเจนท์ที่ถูกอธิบายในแบบจำลองวงจรชีวิตของเอเจนท์จะถูกมองว่าเป็นวัตถุที่ถูกสร้างขึ้นซึ่งจะมีชื่อที่ไม่ซ้ำกันและทำการประมวลผลอย่างอิสระ
- **Unique** : แต่ละเอเจนท์มีสถานะของวงจรชีวิตได้เพียงหนึ่งสถานะ ณ ขณะเวลาหนึ่งและภายในหนึ่ง AP

วงจรชีวิตของเอเจนท์ตามมาตรฐาน FIPA แสดงได้ดัง รูปที่ 2.8



รูปที่ 2.8 วงจรชีวิตของเอเจนท์ตามมาตรฐาน FIPA (FIPA 2002) [2]

จากรูปที่ 2.8 วงจรชีวิตของเอเจนท์จะเริ่มขึ้นเมื่อเอเจนท์ถูกสร้าง (Create) ขึ้นโดยระบบเอเจนท์เริ่มต้นเอเจนท์จะมีสถานะเป็น Initiated จนกว่าระบบเอเจนท์จะทำการ Invoke จึงจะเข้าสู่สถานะ Suspended Waiting และ Transit ตามลำดับ หลังจากนั้นเอเจนท์สามารถกลับสู่สถานะ Active ได้อีกครั้งด้วยการ Resume Wake up และ Execute ตามลำดับ ทั้งนี้การทำงานของเอเจนท์จะสิ้นสุดลงเมื่อระบบเอเจนท์สั่ง Quit หรือ Destroy

ตารางที่ 2.2 สรุปการเปลี่ยนสถานะของเอเจนท์ [2]

การดำเนินการเพื่อเปลี่ยนสถานะ	คำอธิบาย
Create	การสร้างหรือติดตั้งเอเจนท์ใหม่
Invoke	การปลุกให้เอเจนท์ทำงาน
Destroy	การบังคับให้เอเจนท์หยุดการทำงานซึ่งเฉพาะ AMS เท่านั้นที่ทำได้และเอเจนท์ไม่สามารถปฏิเสธได้
Quit	การขอให้เอเจนท์หยุดการทำงานแต่เอเจนท์อาจไม่ทำก็ได้
Suspend	การทำให้เอเจนท์เข้าสู่สถานะ Suspended โดยเอเจนท์หรือ AMS
Resume	ทำให้เอเจนท์ออกจากสถานะ suspended ทำได้โดย AMS เท่านั้น
Wait	การทำให้เอเจนท์เข้าสู่สถานะรอ ทำได้โดยเอเจนท์
Wake up	ทำให้เอเจนท์ออกจากสถานะรอ ทำได้โดย AMS เท่านั้น

เมื่อเอเจนท์มีการเปลี่ยนสถานะ การส่งข้อความให้แก่เอเจนท์ผ่านทาง MTS ในแต่ละสถานะของวงจรชีวิตของเอเจนท์จะถูกจัดการโดย AMS ซึ่งการข้อความในแต่ละสถานะของเอเจนท์แสดงดังตารางที่ 2.3

ตารางที่ 2.3 การส่งข้อความในแต่ละสถานะของเอเจนท์ [2]

สถานะ	การส่งข้อความ
Active	MTS ส่งข้อความให้แก่เอเจนท์แบบปกติ
Initiated/Waiting/Suspended	MTS ทำการเก็บข้อความไว้จนกว่าเอเจนท์จะเปลี่ยนสถานะเป็น Active หรือทำการส่งต่อข้อความไปยังตำแหน่งใหม่หากว่ามีการระบุว่าให้ส่งต่อให้เอเจนท์
Transit	MTS ทำการเก็บข้อความไว้จนกว่าเอเจนท์จะเปลี่ยนสถานะเป็น Active ซึ่งอาจเกิดจากการย้ายการทำงานของเอเจนท์ ลัมเหลวที่ระบบเอเจนท์ต้นทางหรือเอเจนท์เริ่มย้ายการทำงานไปที่ระบบเอเจนท์ปลายทางแล้ว หรือทำการส่งต่อข้อความไปยังตำแหน่งใหม่หากว่ามีการระบุว่าให้ส่งต่อให้เอเจนท์ สถานะ Transit นี้จะเกิดขึ้นกับเอเจนท์แบบเคลื่อนที่เท่านั้น
Unknow	MTS ทำการเก็บข้อความไว้หรือไม่รับข้อความนั้นขึ้นอยู่กับนโยบาย MTS และข้อกำหนดในข้อความนั้น

2.2.7 การติดต่อสื่อสารระหว่างเอเจนท์

เอเจนท์จะมีการติดต่อสื่อสารกันด้วยการส่งข้อความเพื่อทำงานร่วมกันหรือทำการเจรจาต่อรองกัน เนื่องจากเอเจนท์อาจสร้างมาจากต่างระบบกันจึงต้องมีการกำหนดมาตรฐานโครงสร้างข้อความระหว่างเอเจนท์ ซึ่งเกี่ยวข้องกับโดเมนที่เอเจนท์นั้นทำงานเพื่อให้ทุกเอเจนท์เข้าใจตรงกัน และแม้ว่าจะไม่เข้าใจเนื้อหาแต่ก็ควรเข้าใจโครงสร้างของข้อความที่เป็นมาตรฐานโครงสร้างนี้เรียกว่า Agent Communication Languages

ตารางที่ 2.4 Agent Communication Languages [2]

โครงสร้าง	คำอธิบาย
performative	เป็นคำ 1 คำที่จะอธิบายจุดประสงค์ของข้อความ เช่น แจ้งเพื่อทราบ(tell), ยกเลิก(cancel),ลงทะเบียน(register) ,ตอบกลับ (reply)
sender	ชื่อและที่อยู่ของเอเจนท์ที่เป็นผู้ส่ง
receiver	ชื่อและที่อยู่ของเอเจนท์ที่เป็นผู้รับ
language	ภาษาที่ใช้ในเนื้อหาของข้อความ
ontology	ออนโทโลยีหรือคำศัพท์เฉพาะในการแปลข้อความ
content	เนื้อหาข้อความที่ต้องการส่ง

2.3 ชุดพัฒนาจาวาเอเจนท์

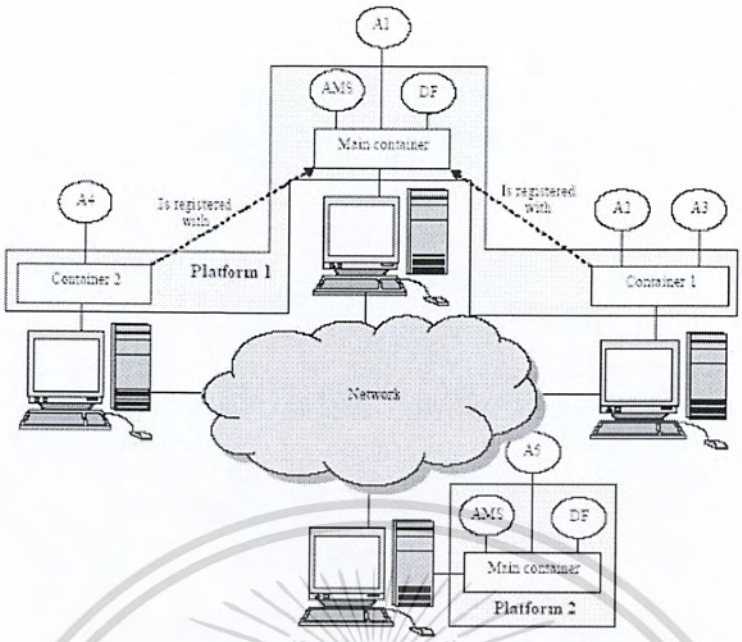
Java Agent Development Framework (JADE) เป็นโครงสร้างการทำงานสำหรับการพัฒนาระบบมัลติเอเจนท์และงานประยุกต์ตามมาตรฐาน FIPA ซึ่งพัฒนาด้วยภาษาจาวา JADE ประกอบด้วย 2 ส่วนหลักคือ JADE Platform เป็นระบบเอเจนท์ตามมาตรฐาน FIPA (FIPA – compliant agent platform) และ JADE Agent เป็นชุดคำสั่งซึ่งเป็นคลาสสำหรับพัฒนาเอเจนท์

2.3.1 JADE Platform

JADE Platform สามารถทำงานได้บนเครื่องคอมพิวเตอร์ที่มีการติดตั้ง Java Virtual Machine (JVM) โดยเมื่อมีการเรียกใช้ JADE Platform ขึ้นมาก็จะมีการสร้างคอนเทนเนอร์ซึ่งจะใช้เป็นพื้นที่หลักเพื่อให้เอเจนท์ทำงานเรียกว่า main – container รวมทั้งมีการสร้างเอเจนท์หลักเพื่อควบคุมการทำงานของแพลตฟอร์มและสร้างคอมโพเน็นท์ในการควบคุมการสื่อสารของเอเจนท์ ได้แก่

- AMS (Agent Management System) เป็นเอเจนท์ที่ทำหน้าที่ควบคุมการเข้าถึงและการใช้งานแพลตฟอร์มรวมถึงการจัดการเอเจนท์ที่ถูกสร้างขึ้นมาใหม่ แต่ละแพลตฟอร์มจะมีเพียง 1 AMS เพื่อจัดการรายการเอเจนท์ในแพลตฟอร์มดูแลวงจรชีวิตของเอเจนท์และสถานะของเอเจนท์ โดยทุกเอเจนท์จะต้องลงทะเบียนกับ AMS เพื่อรับ AID
- DF(Directory Facilitator) เป็นเอเจนท์ที่ทำหน้าที่บริการรายชื่อบริการต่างๆ ของเอเจนท์ที่อยู่ในแพลตฟอร์ม
- Massage Transport System หรือ Agent Communication Channel (ACC) เป็นส่วนการทำงานที่ทำหน้าที่ควบคุมการแลกเปลี่ยน ข้อความระหว่างเอเจนท์ทั้งภายในแพลตฟอร์มและระหว่างแพลตฟอร์ม

เอเจนท์ที่ถูกสร้างขึ้นมาใช้งานสามารถทำงานอยู่ใน main – container หรือคอมเทนเนอร์อื่นในแพลตฟอร์มก็ได้และแพลตฟอร์มของ JADE สามารถกระจายอยู่บนคอมพิวเตอร์หลายเครื่องได้โดยแต่ละเครื่องต่างก็มี JVM ของตนเอง คอนเทนเนอร์ที่สร้างใหม่จะถูกควบคุมโดย AMS และ DF จาก main – container แสดงดังรูปที่ 2.9



รูปที่ 2.9 แสดงแพลตฟอร์มของ JADE แบบกระจาย [9]

2.3.2 JADE Agent

JADE มีคลาส Agent ซึ่งเป็นคลาสพื้นฐานเพื่อใช้ในการสร้างเอเจนท์ที่สามารถทำงานได้ใน JADE Platform ประกอบด้วยเมธอดด้วยๆ ที่จำเป็นต่อการทำงานของเอจเนท์ ซเน การรับส่งข้อความและการเปลี่ยนถานะของเอเจนท์ เป็นต้น

2.4 การวิเคราะห์การไหลของกำลังไฟฟ้า

การวิเคราะห์การไหลของกำลังไฟฟ้านั้นมีวัตถุประสงค์เพื่อศึกษาการไหลของกำลังไฟฟ้าที่จุดต่างๆในระบบไฟฟ้ากำลังในสภาวะการทำงานปกติ เพื่อศึกษาถึงพฤติกรรมของระบบไฟฟ้ากำลังในขณะที่ความต้องการไฟฟ้าเปลี่ยนไป ซึ่งจะทำการศึกษาในสภาวะคงตัว (steady-state) ของระบบไฟฟ้ากำลังเพื่อหาปริมาณต่างๆ อาทิเช่น

- Voltage Profile
- Power Flows
- Current Flows
- Power Factors
- Transformer LTC Setting
- Voltage Drops
- Generator's Mvar Damand (Q_{max} & Q_{min})
- Total Generation & Power Demand
- MW & Mvar losses

บัสในระบบไฟฟ้ากำลังแบ่งออกเป็น 3 ประเภท ได้แก่

1) Slack bus หรือ Reference bus หรือ Floating bus หรือ Swing bus คือ bus ที่ใช้เป็น reference ของมุมของ voltage ในระบบทั้งหมด โดยส่วนใหญ่จะเป็น bus ที่มีgenerator ขนาดใหญ่ที่สุด และกำหนดให้ Slack bus เป็น node ที่ 1 ดังนั้นจะได้ $\delta_1 = 0$ และจะทราบค่า V_1

2) Voltage Control bus หรือ Generator bus หรือ P-V bus คือ bus ที่มีGenerator ต่ออยู่ทำให้ทราบค่า P กับ V ส่วนตัวแปรที่ไม่ทราบค่าคือ δ และ Q

3) Load bus หรือ P-Q bus

ในการหาผลเฉลยของการวิเคราะห์การไหลของระบบไฟฟ้ากำลังนั้นมีอยู่ 3 วิธีหลักๆ ได้แก่ [3]

1. Gauss-Seidel Power Flow Solution
2. Newton-Raphson Power Flow Solution
3. Fast Decoupled Power Flow Solution

สำหรับโครงงานนี้ได้ใช้วิธี Newton-Raphson Power Flow Solution ในการหาผลเฉลย ซึ่งมีรายละเอียดดังนี้

กำหนดให้บัสที่ 1 เป็น Slack bus สำหรับโหลดบัสนั้นจะต้องสมมุติค่า δ และ V เริ่มต้น ส่วน PV bus นั้นสมมุติค่า δ เริ่มต้นเพียงค่าเดียว โดยทั่วไปจะสมมุติค่าแรงดันเริ่มต้นเท่ากับ 1.0 pu และมุมเริ่มต้นเท่ากับ 0 เรเดียน

ความสัมพันธ์ระหว่างแรงดัน กระแส และ Y-Bus Matrix เป็นดังสมการที่ 2.4

$$\mathbf{I}_i = \sum_{j=1}^{n} \mathbf{Y}_{ij} \mathbf{V}_j \tag{2.4}$$

จาก สมการที่ 2.4 เขียนเป็นแบบโพลาร์ได้ดังสมการที่ 2.5

$$\mathbf{I}_i = \sum_{j=1}^{n} |Y_{ij}||V_j| \angle \theta_{ij} + \delta_j \tag{2.5}$$

กำลังไฟฟ้าเชิงซ้อนของบัส i คือ

$$P_i - jQ_i = \mathbf{V}_\mathbf{i}^* \mathbf{I}_\mathbf{i} \tag{2.6}$$

นำสมการที่ 2.5 แทนค่า $\mathbf{I}_i$ ลงในสมการที่ 2.6 จะได้ว่า

$$P_i - jQ_i = |V_i| \angle -\delta_i \sum_{j=1}^{n} |Y_{ij}||V_j| \angle \theta_{ij} + \delta_j \tag{2.7}$$

แยกออกเป็นเทอมที่เป็นส่วนจริงและส่วนจินตภาพจะได้ว่า

$$P_i = \sum_{j=1}^{n} |V_i||V_j||Y_{ij}| \cos\left(\theta_{ij} - \delta_i + \delta_j\right) \tag{2.8}$$

$$Q_i = -\sum_{j=1}^{n} |V_i||V_j||Y_{ij}| \sin\left(\theta_{ij} - \delta_i + \delta_j\right) \tag{2.9}$$

สมการทั้งสองเป็นสมการที่เป็น nonlinear จากวิธีการของ Newton-Raphson เขียนสมการความสัมพันธ์ของ Jacobian Matrix ได้เป็นดังนี้

$$\begin{bmatrix} \Delta P_2^{(k)} \\ \vdots \\ \Delta P_n^{(k)} \\ \Delta Q_2^{(k)} \\ \vdots \\ \Delta Q_n^{(k)} \end{bmatrix} = \left[\begin{array}{ccc|ccc} \frac{\partial P_2}{\partial \delta_2}^{(k)} & \cdots & \frac{\partial P_2}{\partial \delta_n}^{(k)} & \frac{\partial P_2}{\partial |V_2|}^{(k)} & \cdots & \frac{\partial P_2}{\partial |V_n|}^{(k)} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial P_n}{\partial \delta_2}^{(k)} & \cdots & \frac{\partial P_n}{\partial \delta_n}^{(k)} & \frac{\partial P_n}{\partial |V_2|}^{(k)} & \cdots & \frac{\partial P_n}{\partial |V_n|}^{(k)} \\ \hline \frac{\partial Q_2}{\partial \delta_2}^{(k)} & \cdots & \frac{\partial Q_2}{\partial \delta_n}^{(k)} & \frac{\partial Q_2}{\partial |V_2|}^{(k)} & \cdots & \frac{\partial Q_2}{\partial |V_n|}^{(k)} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial Q_n}{\partial \delta_2}^{(k)} & \cdots & \frac{\partial Q_n}{\partial \delta_n}^{(k)} & \frac{\partial Q_n}{\partial |V_2|}^{(k)} & \cdots & \frac{\partial Q_n}{\partial |V_n|}^{(k)} \end{array}\right] \begin{bmatrix} \Delta \delta_2^{(k)} \\ \vdots \\ \Delta \delta_n^{(k)} \\ \Delta |V_2^{(k)}| \\ \vdots \\ \Delta |V_n^{(k)}| \end{bmatrix} \tag{2.10}$$

จากสมการที่ 2.10 สามารถเขียนเป็นรูปอย่างย่อได้ดังนี้

$$\begin{bmatrix} \Delta \mathbf{P} \\ \Delta \mathbf{Q} \end{bmatrix} = \begin{bmatrix} \mathbf{J}_1 & \mathbf{J}_2 \\ \mathbf{J}_3 & \mathbf{J}_4 \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{\delta} \\ \Delta |\mathbf{V}| \end{bmatrix} \tag{2.11}$$

สมาชิกใน $\mathbf{J}_1$ มีดังนี้

$$\frac{\partial P_i}{\partial \delta_i} = \sum_{j \neq i} |V_i||V_j||Y_{ij}| \sin\left(\theta_{ij} - \delta_i + \delta_j\right) \tag{2.12}$$

$$\frac{\partial P_i}{\partial \delta_j} = -|V_i||V_j||Y_{ij}| \sin\left(\theta_{ij} - \delta_i + \delta_j\right) \qquad j \neq i \tag{2.13}$$

สมาชิกใน $\mathbf{J}_2$ มัดังนี้

$$\frac{\partial P_i}{\partial |V_i|} = 2|V_i||Y_{ii}|\cos(\theta_{ii}) + \sum_{j\neq i}|V_j||Y_{ij}|\cos(\theta_{ij} - \delta_i + \delta_j) \tag{2.14}$$

$$\frac{\partial P_i}{\partial |V_j|} = |V_i||Y_{ij}|\cos(\theta_{ij} - \delta_i + \delta_j) \qquad j \neq i \tag{2.15}$$

สมาชิกใน $\mathbf{J}_3$ มัดังนี้

$$\frac{\partial Q_i}{\partial \delta_i} = \sum_{j\neq i}|V_i||V_j||Y_{ij}|\cos(\theta_{ij} - \delta_i + \delta_j) \tag{2.16}$$

$$\frac{\partial Q_i}{\partial \delta_j} = -|V_i||V_j||Y_{ij}|\cos(\theta_{ij} - \delta_i + \delta_j) \qquad j \neq i \tag{2.17}$$

สมาชิกใน $\mathbf{J}_4$ มัดังนี้

$$\frac{\partial Q_i}{\partial |V_i|} = -2|V_i||Y_{ii}|\sin(\theta_{ii}) - \sum_{j\neq i}|V_j||Y_{ij}|\sin(\theta_{ij} - \delta_i + \delta_j) \tag{2.18}$$

$$\frac{\partial Q_i}{\partial |V_j|} = -|V_i||Y_{ij}|\sin(\theta_{ij} - \delta_i + \delta_j) \qquad j \neq i \tag{2.19}$$

ในเทอมของ $\Delta P_i^{(k)}$ และ $\Delta Q_i^{(k)}$ นั้นเป็นผลต่างระหว่างค่าจริงที่กำหนดไว้ (sch) ซึ่งเป็นข้อมูลที่ได้จากโหลดบัสหรือบัสควบคุมแรงดัน กับค่าที่ได้จากการคำนวณ

$$\Delta P_i^{(k)} = P_i^{(sch)} - P_i^{(k)} \tag{2.20}$$

$$\Delta Q_i^{(k)} = Q_i^{(sch)} - Q_i^{(k)} \tag{2.21}$$

จากสมการที่ 2.10 เมื่อทราบค่าทั้งหมดแล้ว สามารถทำการหาค่า $\Delta\delta_i^{(k)}$ และ $\Delta|V_i^{(k)}|$ ได้ เพราะฉะนั้น เราสามารถหาค่าประมาณของ δ และ V ค่าใหม่ได้ดังนี้

$$\delta_i^{(k+1)} = \delta_i^{(k)} + \Delta\delta_i^{(k)} \tag{2.22}$$

$$|V_i^{(k+1)}| = |V_i^{(k)}| + \Delta|V_i^{(k)}| \tag{2.23}$$

เมื่อนำค่าประมาณค่าใหม่ไปทำการคำนวณอีกครั้ง จะทำให้ผลต่างของผลเฉลยออกมานั้นมีค่าความแตกต่างจากครั้งแรก ดังนั้นเมื่อทำมากครั้ง ผลต่างก็จะน้อยลงจนถึงค่าผลต่างที่สามารถยอมรับได้ คำตอบสุดท้ายที่ได้ก็คือค่า δ และ V ของบัสนั้นๆ เมื่อทราบค่า δ และ V ของทุกบัสแล้วก็สามารถหาการไหลของกระแสกำลังไฟฟ้าได้ดังนี้

จากบัส $i \longrightarrow j$

$$\mathbf{I}_{ij} = \mathbf{y}_{ij}\left(\mathbf{V}_i - \mathbf{V}_j\right) \tag{2.24}$$

$$\mathbf{I}_{ij} = y_{ij}\angle\gamma_{ij}\left(V_i\angle\delta_i - V_j\angle\delta_j\right) \tag{2.25}$$

เมื่อ $\mathbf{y}_{ij}$ คือ admittance ระหว่างบัส i กับ j

สมการการไหลของกำลังไฟฟ้า

$$\mathbf{S}_{ij} = \mathbf{V}_i\mathbf{I}_{ij}^{*} \tag{2.26}$$

จากสมการที่ 2.25 จะได้

$$\mathbf{I}_{ij}^{*} = y_{ij}\angle-\gamma_{ij}\left(V_i\angle-\delta_i - V_j\angle-\delta_j\right) \tag{2.27}$$

แทนค่าลงในสมการที่ 2.26 จะได้ว่า

$$\mathbf{S}_{ij} = V_i\angle\delta_i\left[y_{ij}\angle-\gamma_{ij}\left(V_i\angle-\delta_i - V_j\angle-\delta_j\right)\right] \tag{2.28}$$

$$\mathbf{S}_{ij} = \left(V_i^2\angle-\gamma_{ij}\right) - \left(V_iV_j\angle\delta_i - \delta_j - \gamma_{ij}\right) \tag{2.29}$$

เมื่อแยกเป็นส่วนจริงและส่วนจินตภาพจะได้ว่า

$$P_{ij} = \left(V_i^2\cos\left(-\gamma_{ij}\right)\right) - \left(V_iV_j\cos\left(\delta_i - \delta_j - \gamma_{ij}\right)\right) \tag{2.30}$$

$$Q_{ij} = \left(V_i^2\sin\left(-\gamma_{ij}\right)\right) - \left(V_iV_j\sin\left(\delta_i - \delta_j - \gamma_{ij}\right)\right) \tag{2.31}$$

กำลังสูญเสียในสายระหว่างบัส i กับ j คือ

$$P_{L,ij} = P_{ij} + P_{ji} \tag{2.32}$$

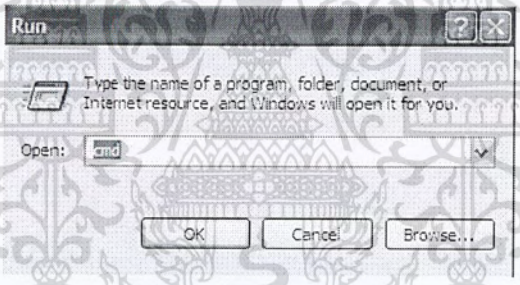
$$Q_{L,ij} = Q_{ij} + Q_{ji} \tag{2.33}$$

บทที่ 3
การใช้งานชุดพัฒนาจาวาเอเจนท์

ในบทที่ 3 นี้ จะกล่าวถึงวิธีการใช้งานชุดพัฒนาจาวาเอเจนท์ (Java Agent Development Framework : JADE) เริ่มตั้งแต่การตั้งค่าต่างๆ การนำเมธอดมาใช้ซึ่งมีเมธอดที่หลากหลาย ดังนั้นในบทนี้จะกล่าวถึงเมธอดพื้นฐานของ JADE เช่นการกำหนดพฤติกรรมของเอเจนท์แบบต่างๆ การสื่อสารของเอเจนท์ด้วยการรับส่งข้อความ ACL Message รวมไปถึงการสื่อสารระหว่าง Platform และการนำเอเจนท์ที่มีอยู่ในโปรแกรม JADE เช่น Dummy Agent และ Sniffer Agent มาใช้งานร่วมกัน

3.1 การใช้งาน command prompt

ในการคอมไพล์และรันโปรแกรมนั้นจะใช้ command prompt บนระบบปฏิบัติการณ์ Dos ดำเนินการทั้งหมด ในการเรียก command prompt ให้คลิกที่ Start แล้วคลิกที่ Run ในช่อง Open ให้พิมพ์ว่า cmd แสดงดังรูปที่ 3.1



รูปที่ 3.1 การเรียกหน้าต่าง command prompt

คำสั่งพื้นฐานของ command prompt มีดังต่อไปนี้

cd <directory name>	เข้าถึงไดเร็กทอรี่ที่ระบุ

ตัวอย่างที่ 3.1

```
C:\>cd jade          // เข้าถึงไดเร็กทอรี่โฟลเดอร์ที่ชื่อว่า jade ขณะอยู่ที่ c:\
C:\jade>
```

cd..	ออกจากไดเร็กทอรี่ 1 ขั้น

ตัวอย่างที่ 3.2

```
C:\Program Files\Java>cd..
C:\Progeam Files>cd..
C:\>
```

cd\	ออกจากไดเร็กทอรี่ไปยังไดร์ฟเริ่มต้น

ตัวอย่างที่ 3.3

```
C:\Program Files\Java>cd\
C:\>
```

d:	ไปยังไดร์ฟ D (หรืออาจเป็นไดร์ฟอื่นๆ)

ตัวอย่างที่ 3.4

```
C:\Program Files\Java>d:
D:\>
```

cls	เคลียร์หน้าจอ

ตัวอย่างที่ 3.5

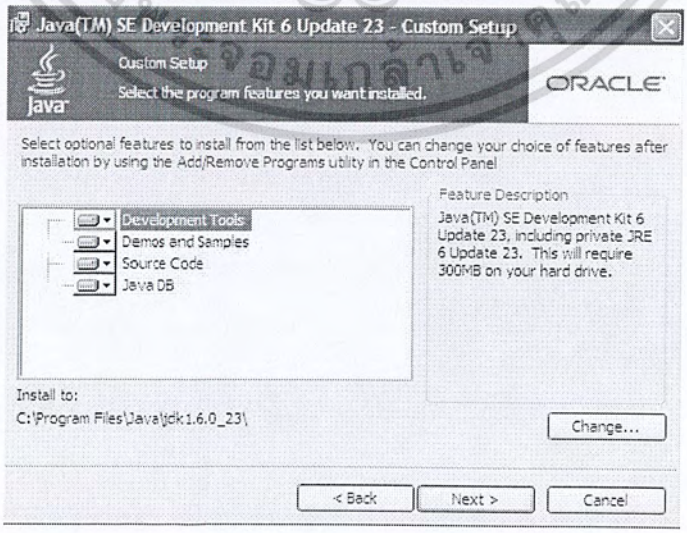
```
C:\>cls
```

3.2 การติดตั้ง

การติดตั้งนั้นจะแบ่งออกเป็น 2 ส่วนคือการติดตั้งส่วนพัฒนาภาษาจาวา และส่วนของ JADE โดยมีขั้นตอนดังต่อไปนี้

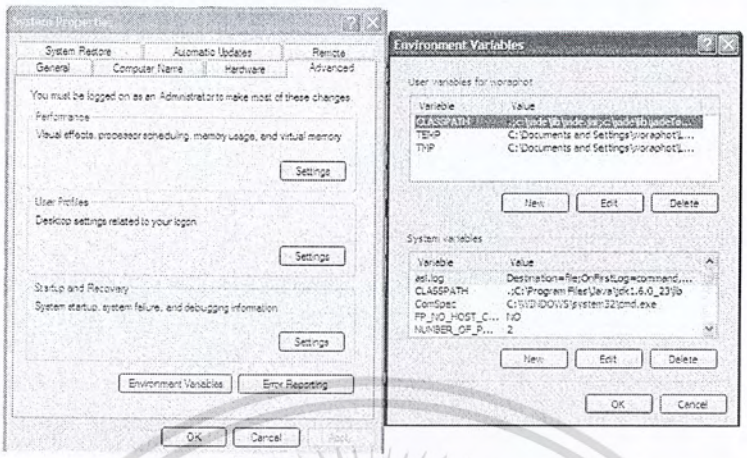
3.2.1 การติดตั้งชุดพัฒนาจาวา

ขั้นตอนที่ 1 ดาวน์โหลดชุดพัฒนาภาษาจาวาจากเว็บไซต์ http://www.oracle.com/technetwork/java/javase/downloads/index.html โดยดาวโหลดไฟล์ JDK (Java Development Kit) สำหรับโครงงานนี้ใช้ JDK เวอร์ชั่น 6 update 23 แล้วทำการติดตั้งดังรูปที่ 3.2



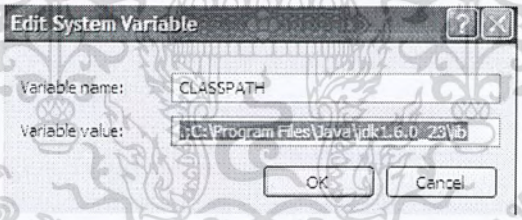
รูปที่ 3.2 การติดตั้งชุดพัฒนาจาวา JDK 6 update 23

ขั้นตอนที่ 2 คลิกขวาที่ My Computer เลือก Properties จากนั้นเลือกที่แท็บ Advanced แล้วคลิกที่ Environment Variables เพื่อจะไปตั้งค่าต่างๆ



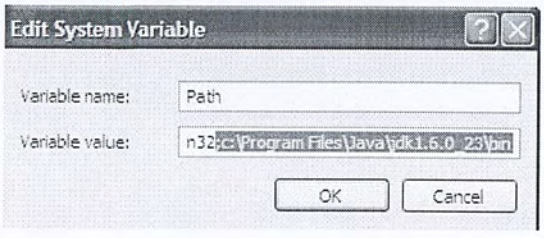
รูปที่ 3.3 การเข้าสู่การตั้งค่าชุดพัฒนาจาวา

ขั้นตอนที่ 3 ตั้งค่า CLASSPATH ของชุดพัฒนาจาวาโดยในส่วนของ System Variables ในตัวแปร CLASSPATH คลิกที่ Edit เข้าไปแล้วในส่วนของ Variable Value ให้พิมพ์ว่า **.;C:\Program Files\Java\jdk1.6.0_23\lib** แสดงดังรูปที่ 3.4



รูปที่ 3.4 การตั้งค่า CLASSPATH ของชุดพัฒนาจาวา

ขั้นตอนที่ 4 ตั้งค่า Path ของชุดพัฒนาจาวาโดยในส่วนของ System Variables ในตัวแปร Path คลิกที่ Edit เข้าไปแล้วในส่วนของ Variable value ให้พิมพ์ต่อท้ายว่า **;c:\Program Files\Java\jdk1.6.0_23\bin** แสดงดังรูปที่ 3.5



รูปที่ 3.5 การตั้งค่า Path ของชุดพัฒนาจาวา

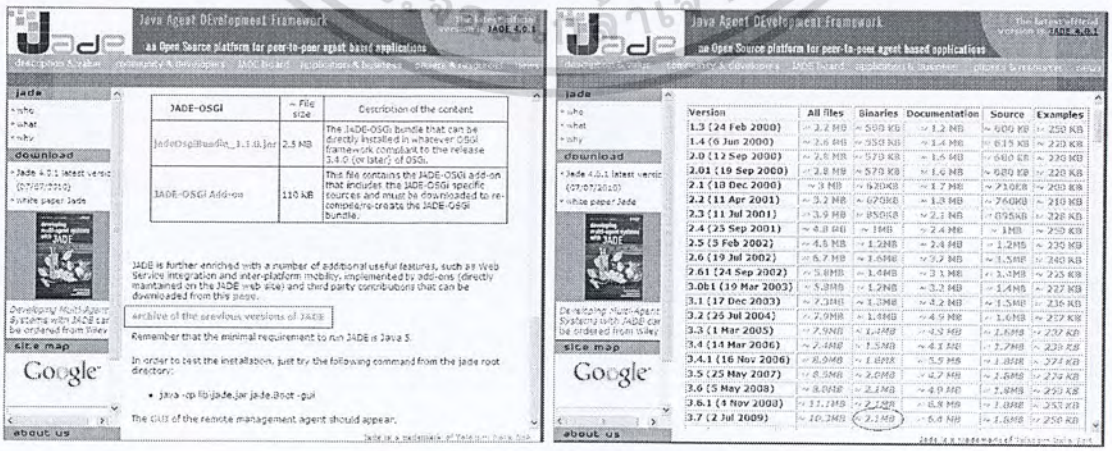
ขั้นตอนที่ 5 ไปที่หน้า command prompt พิมพ์คำสั่งว่า javac จะปรากฏข้อความดังรูปที่ 3.6 แสดงว่าการติดตั้งเสร็จสมบูรณ์

รูปที่ 3.6 การตรวจสอบการตั้งค่าชุดพัฒนาจาวา

3.2.2 การติดตั้งชุดพัฒนาจาวาเอเจนท์

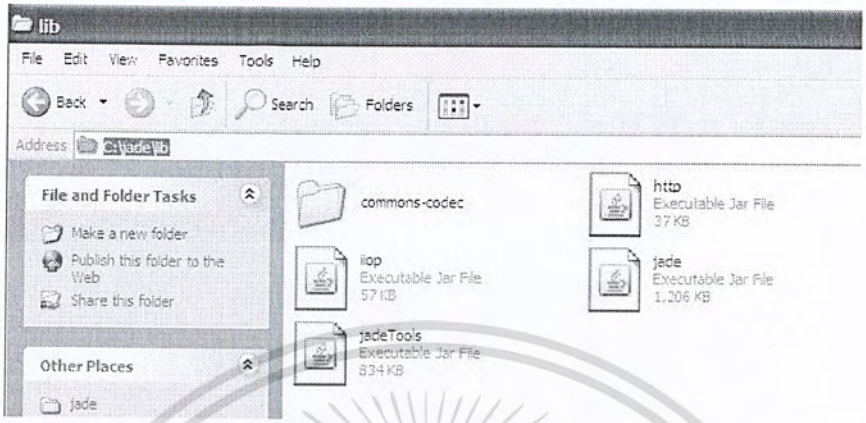
ขั้นตอนที่ 1 ไปที่เว็บไซต์ http://jade.tilab.com/ คลิกที่ Download แต่จะต้องทำการล้อกอิน ให้ทำการลงทะเบียนให้เรียบร้อย

ขั้นตอนที่ 2 ทำการล้อกอินเข้าไปแล้วจะมีตัวเลือกให้ดาวน์โหลดไฟล์ สำหรับโครงงานนี้ได้ใช้ **JADE 3.7** จากนั้นให้ทำการดาวน์โหลดไฟล์ Binaries จะได้ไฟล์ชื่อว่า **jade-bin-3.7.zip** (ส่วนไฟล์อื่นๆ นั้นเป็นไฟล์ข้อมูลต่างๆ รวมทั้งตัวอย่างประกอบ ถ้าต้องการศึกษาเพิ่มเติมก็สามารถดาวน์โหลดมาได้)



รูปที่ 3.7 การดาวน์โหลด JADE

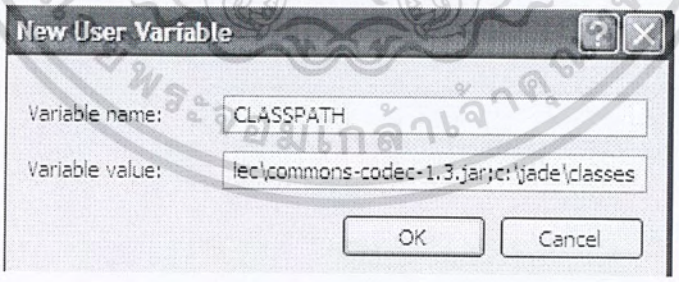
ขั้นตอนที่ 3 เมื่อได้ไฟล์ดังกล่าวแล้วให้ทำการ extract โฟลเดอร์ที่ชื่อว่า jade ไว้ที่ c:\ เมื่อทำการ extract แล้ว ให้เข้าไปดูในไดเร็กทอรี่ C:\jade\lib แล้วตรวจสอบว่ามีไฟล์ครบดังรูปที่ 3.8



รูปที่ 3.8 ไฟล์ที่ได้จากการ extract จากไฟล์ jade-bin-3.7.zip

ขั้นตอนที่ 4 ทำการตั้งค่า CLASSPATH ของ JADE โดยเข้าไปที่ Environment Variables เช่นเดียวกับการตั้งค่าในชุดพัฒนาจาวา ในส่วนของ user variables นั้น ให้คลิกที่ New จากนั้นในช่อง Variable name ให้พิมพ์คำว่า **CLASSPATH** และVariable valueให้พิมพ์ดังนี้

.;c:\jade\lib\jade.jar;c:\jade\lib\jadeTools.jar;c:\jade\lib\iiop.jar;
c:\jade\lib\http.jar;c:\jade\lib\commons-codec\commons-codec-
1.3.jar;c:\jade\classes



รูปที่ 3.9 การตั้งค่า CLASSPATH ของ JADE

3.3 การสร้างและเรียกใช้เอเจนท์

ในการเขียนโปรแกรมสำหรับสร้างเอเจนท์นั้นสามารถใช้ Notepad แล้วบันทึกไฟล์เป็น .java หรืออาจจะใช้โปรแกรมอื่นที่สามารถใช้เขียนโค้ดภาษาจาวาเช่น Edit Plus หรือ JCreator เป็นต้น ซึ่งจะสะดวกมากขึ้น สำหรับโครงงานนี้จะใช้ JCreator 4.5 ซึ่งโปรแกรมนั้นติดตั้งง่าย ใช้งานง่าย และโปรแกรมมีขนาดเล็ก

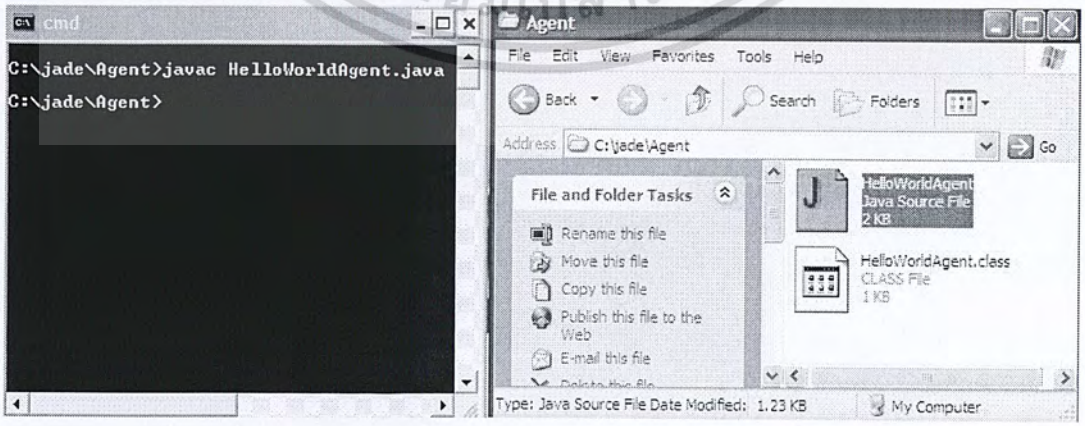
สิ่งแรกที่จะต้องเขียนในโปรแกรมคือการ import ไลบรารี่เอเจนท์คือ import jade.core.Agent; และชื่อไฟล์จะต้องเหมือนกับชื่อคลาส และคลาสจะต้องมีการสืบทอดจากคลาสของเอเจนท์ด้วยดังตัวอย่างต่อไปนี้ [2]

```
import jade.core.Agent;

public class HelloWorldAgent extends Agent {

  protected void setup() {
      System.out.println("Hello World! My name is "+getLocalName());

  }
}
```

เมื่อทำการบันทึกไฟล์แล้วจะต้องทำการคอมไพล์โดยใช้ command prompt แล้วเข้าไปยังไดเร็กทอรี่ที่เก็บไฟล์ดังกล่าวจากนั้นคอมไพล์โดยใช้คำสั่ง javac <ชื่อไฟล์.java> ดังตัวอย่างต่อไปนี้

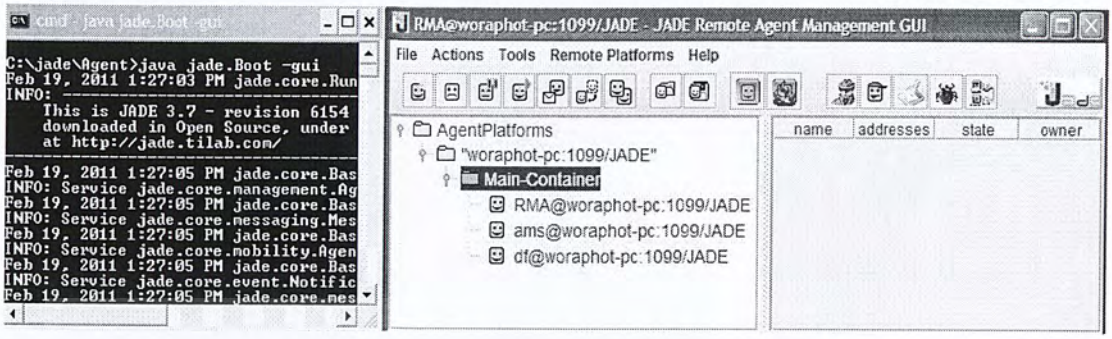
```
>javac HelloWorldAgent.java
```

เมื่อคอมไพล์แล้ว หากมีข้อความเกิดขึ้นแสดงว่าโปรแกรมมีปัญหาที่โค้ดของโปรแกรมที่ตำแหน่งต่างๆ ต้องดำเนินการแก้ไข ถ้าคอมไพล์สำเร็จจะไม่ปรากฏข้อความใดๆ และจะได้คลาสไฟล์ที่ไดเร็กทอรี่ที่เก็บโค้ดไว้ แสดงดังรูปที่ 3.10



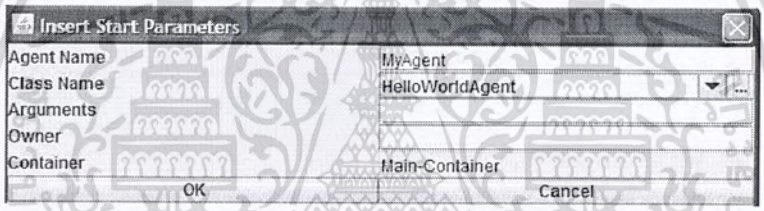
รูปที่ 3.10 การคอมไพล์โปรแกรมเอเจนท์

จากนั้นทำการเรียกส่วนติดต่อผู้ใช้ (Graphic User Interface : GUI) ของ JADE โดยใช้คำสั่ง java jade.Boot –gui ที่ไดเร็กทอรี่ที่มีคลาสไฟล์ แสดงดังรูปที่ 3.11



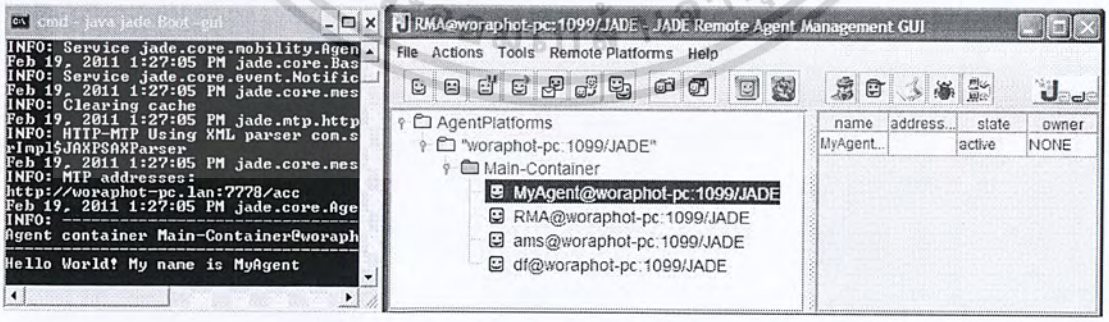
รูปที่ 3.11 การเรียกส่วนติดต่อผู้ใช้ของ JADE

จากนั้นทำการเรียกใช้งานเอเจนท์โดยคลิกขวาที่ Container แล้วคลิกที่ Start New Agent จากนั้นใส่ชื่อเอเจนท์และคลาสไฟล์ของเอเจนท์ดังรูปที่ 3.12



รูปที่ 3.12 การเรียกใช้งานเอเจนท์

จะเห็นว่า เอเจนท์ถูกเรียกใช้งานแล้ว และทำงานตามคำสั่งที่ได้ระบุไว้คือพิมพ์ ข้อความว่า Hello World! My name is <ชื่อเอเจนท์>



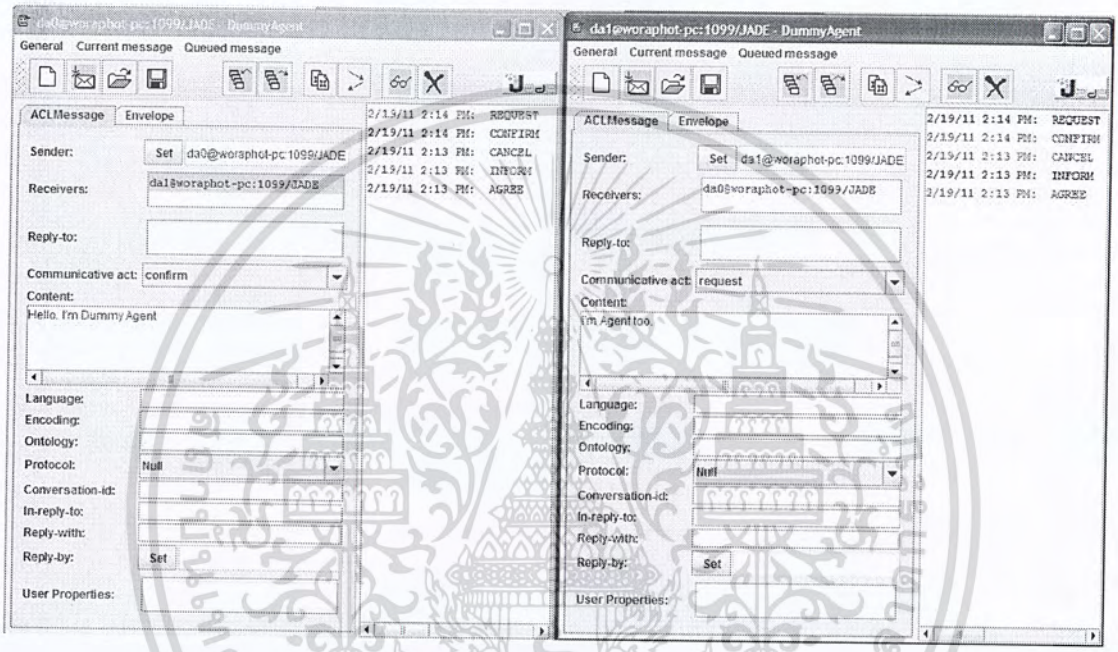
รูปที่ 3.13 ผลลัพธ์จากการเรียกเอเจนท์

เราสามารถเรียกเอเจนท์ที่มีคลาสเดียวกันได้อีกเรื่อยๆ แต่ชื่อของเอเจนท์นั้นจะต้องไม่ซ้ำกัน

3.4 เอเจนท์สำเร็จรูป

3.4.1 Dummy Agent

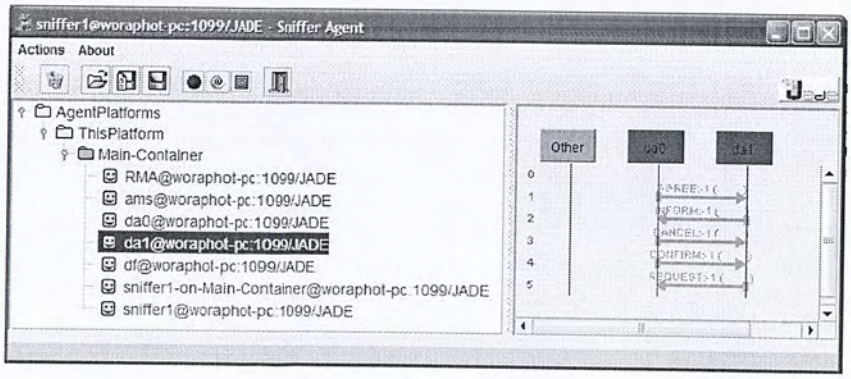
Dummy Agent [2] เป็นเครื่องมืออย่างง่ายสำหรับใช้ส่งข้อความ ACL Message ที่กำหนดได้เอง ซึ่งใช้ประโยชน์ในการทดสอบพฤติกรรมของเอเจนท์เมื่อเอเจนท์ได้รับข้อความ ACL Message โดยเอเจนท์นี้มีลักษณะเป็น GUI ดังรูปที่ 3.14 แสดง Dummy Agent ที่มีการตอบโต้การสนทนากัน



รูปที่ 3.14 Dummy Agent

3.4.2 Sniffer Agent

Sniffer Agent [2] เป็นเครื่องมือที่ใช้สำหรับดูการโต้ตอบของ ACL Message ระหว่างเอเจนท์ จากรูปที่ 3.14 นั้นสามารถแสดงทิศทางการตอบโต้กันแสดงดังรูปที่ 3.15



รูปที่ 3.15 Sniffer Agent

3.5 ชนิดและการทำงานของ Behaviour

พฤติกรรมที่เอเจนท์จะต้องทำจะถูกกำหนดไว้ใน behaviour [2] ซึ่งแต่ละ behaviour จะใช้เธรดในการประมวลผลแยกจากกัน เอเจนท์สามารถจัดการและใช้งาน behaviour ได้โดยการ import คลาส jade.core.behaviours และกำหนดงานภายในเมธอด action() ชนิดของ behaviour แบ่งออกเป็นดังนี้

1. One-Shot behaviour เป็น behaviour ที่มีการประมวลผลทันทีและเมธอด action() จะถูกประมวลผลเพียงครั้งเดียว การเรียก One-Shot behaviour ทำได้โดยการขยายคลาส jade.core.behaviour.OneShotBehaviour

```
public class MyOneShotBehaviour extends OneShotBehaviour {
      public void action() {
            // perform operation X
      }
}
```

จากตัวอย่างนี้ operation X จะดำเนินการเพียง 1 ครั้ง

2. Cyclic behaviour เป็น behaviour ที่มีการทำงานในเมธอด action() วนซ้ำไม่สิ้นสุด โดยเรียก behaviour แบบนี้ได้โดยการขยายคลาส jade.core.behaviour.CyclicBehaviour

```
public class MyCyclicBehaviour extends CyclicBehaviour {
      public void action() {
            // perform operation Y
      }
}
```

จากตัวอย่างนี้ operation Y จะทำงานเรื่อยๆ จนกว่าเอเจนท์ที่มี behaviour นี้จะถูกทำลาย

3. Generic behaviour เป็น behaviour ที่มีการกำหนดสถานะไว้ และจะทำงานตามสถานะที่กำหนดจนเสร็จตามเงื่อนไข โดยเรียก behaviour แบบนี้ได้โดยการขยายคลาส jade.core.behaviour.Behaviour

```
public class ThreeStepBehaviour extends Behaviour {
      private int step = 0;
            public void action() {
            switch (step) {
                  case 0:
                        // perform operation X
                        step++;
                        break;
                  case 1:
                        // perform operation Y
                        step++;
                        break;
                  case 2:
                        // perform operation Z
```

```
                        step++;
                        break;
                    }
                }
        public boolean done() {
                return step == 3;
        }
}
```

X, Y, Z จะถูกทำงานอย่างละครั้งตามลำดับแล้ว behaviour จึงจะทำงานเสร็จ

ตัวอย่างที่ 3.6

```
import jade.core.Agent;
import jade.core.behaviours.CyclicBehaviour;

public class AgentCyclic extends Agent{

        protected void setup(){
                System.out.println(getAID());
                addBehaviour(new Receiver());
        }

        private class Receiver extends CyclicBehaviour{
                public void action(){
                        System.out.println("Received");
                }
        }
}
```

เมื่อเรียกเอเจนท์นี้มาแล้วจะพิมพ์ข้อความ Received เรื่อยๆ จนกว่าเอเจนท์นี้จะถูกทำลาย

JADE มี behaviour สำหรับใช้กำหนดให้มีการดำเนินการได้ตามเวลาที่กำหนด ซึ่งทำได้โดยเรียกคลาสดังต่อไปนี้

4. Waker behaviour เป็น behaviour ที่มีการทำงานโดยระบุเวลาที่จะให้ดำเนินการเมื่อถึงเวลาที่ระบุไว้เป็นหน่วยมิลลิวินาที โดยเรียก behaviour แบบนี้ได้โดยการขยายคลาส jade.core.behaviour.WakerBehaviour

```
public class MyAgent extends Agent {
        protected void setup() {
                System.out.println("Adding waker behaviour");

        addBehaviour(new WakerBehaviour(this, 10000) {
                protected void onWake() {
                        // perform operation X
                        }
                } );
        }
}
```

Operation X จะทำงานหลังจากพิมพ์ข้อความ Adding waker behaviour ไปแล้ว 10 วินาที

5. Ticker behaviour เป็น behaviour ที่มีการทำงานให้ดำเนินการวนซ้ำตามคาบเวลาที่ระบุไว้ซึ่งเวลามีหน่วยเป็นวินาที เรียก behaviour นี้ได้โดยการขยายคลาส jade.core.behaviour.TickerBehaviour

```
public class MyAgent extends Agent {
      protected void setup() {

      addBehaviour(new TickerBehaviour(this, 10000) {
            protected void onTick() {
                  // perform operation Y
            }
      } );
      }
}
```

Operation Y จะทำซ้ำเรื่อยๆ คาบละ 10 วินาที

ตัวอย่างที่ 3.7

```
import jade.core.Agent;
import jade.core.behaviours.TickerBehaviour;

public class AgentBehave extends Agent {

    public void setup() {

      addBehaviour(new TickerBehaviour(this,10000) {
            public void onTick() {
                  System.out.println("Cycling");
            }
      });
    }
}
```

เมื่อเรียกเอเจนท์นี้แล้วจะพิมพ์ข้อความ Cycling ออกมาทุกๆ 10 วินาที

3.6 การสื่อสารด้วย ACL Message

JADE Agent ติดต่อสื่อสารกันโดยใช้การส่งข้อความโดยแต่ละเอเจนท์มีคิวสำหรับเก็บข้อความ (message queue) [2] เมื่อมีเอเจนท์อื่นส่งข้อความมาให้ ข้อความจะถูกเก็บอยู่ในคิวของเอเจนท์ผู้รับจนกว่าผู้รับจะดึงข้อความออกจากคิว ข้อความทุกข้อความนั้นจะต้องเป็นแบบตัวแปร String ดังนั้นในการรับส่งข้อความจะต้องมีการแปลงชนิดตัวแปรเมื่อจำเป็น เช่น ถ้ามีส่วนของการคำนวณจะต้องแปลงตัวแปรระหว่าง String กับ Integer หรือ Double เป็นต้น ซึ่งจะกล่าวรายละเอียดไว้ในส่วนของภาคผนวก ก

3.6.1 การส่งข้อความ

JADE มีคลาสในการส่งข้อความคือ jade.lang.acl.ACLMessage การส่งข้อความทำได้โดยการระบุฟิลด์ของอ็อบเจค ACL Message แล้วเรียกเมธอด send ของคลาสเอเจนท์ ตัวอย่างรหัสคำสั่งในการส่งข้อความแจ้งผู้รับให้แก่เอเจนท์ที่ชื่อ Peter มีข้อความว่า Today it's raining

```
ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
msg.addReceiver(new AID("Peter", AID.ISLOCALNAME));
msg.setLanguage("English");
msg.setOntology("Weather-forecast-ontology");
msg.setContent("Today it's raining");
send(msg);
```

3.6.2 การรับข้อความ

ข้อความจะถูกส่งไปยังคิวของข้อความอัตโนมัติ เมื่อมีข้อความอยู่ในคิวแล้วเอเจนท์สามารถดึงข้อความออกจากคิวโดยใช้เมธอด receive ซึ่งจะส่งข้อความแรกในคิวออกมาและลบข้อความออกจากคิว ตัวอย่างคำสั่งการรับข้อความเป็นดังนี้

```
ACLMessage msg = receive();
      if (msg != null) {
      // Process the message
}
```

เมื่อได้รับข้อความมาแล้วก็สามารถตอบกลับด้วย ACL Message ได้เช่นกันดังนี้

```
ACLMessage msg = receive();
      if (msg != null) {
            ACLMessage reply = msg.createReply();
            reply.setLanguage("English");
            reply.setContent("Yes, I'm Peter");
            send(reply);
      }
```

ตัวอย่างที่ 3.8

ทำการสร้างเอเจนท์ Sender และ Receiver ให้เอเจนท์ทั้งสองสื่อสารกันดังต่อไปนี้

- เอเจนท์ส่งข้อความ

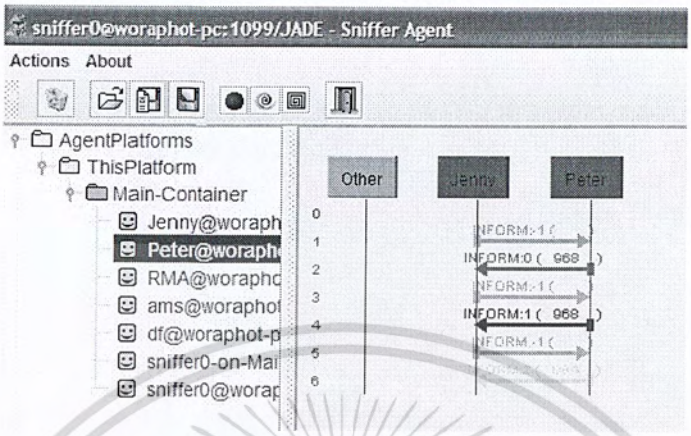
```
//--------------------Sender-------------------------

import jade.core.Agent;
import jade.core.behaviours.TickerBehaviour;
import jade.core.AID;
import jade.lang.acl.*;

public class Sender extends Agent {
```

```
    public void setup() {

      System.out.println("Hello. My name is "+getLocalName());

      addBehaviour(new TickerBehaviour(this,10000){
            public void onTick() {

            ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
            msg.addReceiver(new AID("Peter", AID.ISLOCALNAME));
            msg.setLanguage("English");
            msg.setContent("Are you Peter?");
            send(msg);;
            }
      });
    }
}

//--------------------end-------------------------
```

- เอเจนท์รับข้อความ

```
//----------------------Receiver-------------------------

import jade.core.AID;
import jade.core.Agent;
import jade.lang.acl.*;
import jade.core.behaviours.*;
import jade.lang.acl.MessageTemplate;


public class Receiver extends Agent{

      protected void setup(){
      System.out.println("Generator Agent " +getAID().getName());
      addBehaviour(new Receive());

      }

private class Receive extends CyclicBehaviour {

      public void action(){

            ACLMessage msg = receive();

            if ((msg!=null)) {
                  System.out.println("You says "+msg.getContent());
                  ACLMessage reply = msg.createReply();
                  reply.setLanguage("English");
                  reply.setContent("Yes, I'm Peter");
                  send(reply);

                  }
            else block();

            }
      }
}

//--------------------end-------------------------
```

เมื่อทำการเรียกเอเจนท์ทั้งสองแล้วจะพบว่าเมื่อ Sender ส่งข้อความไปแล้ว Receiver จะมีการตอบสนองโดยมีการส่งข้อความตอบกลับมา สามารถดูได้จาก Sniffer Agent ดังรูปที่ 3.16



รูปที่ 3.16 การตอบสนองของ Sender และ Receiver

เอเจนท์ที่มีหน้าที่รับข้อความก็สามารถเลือกข้อความได้ตามเงื่อนไขที่ระบุโดยการขยายคลาส jade.lang.acl.MessageTemplate โดยมีรูปแบบคำสั่งดังนี้

```
private MessageTemplate mt =
  MessageTemplate.MatchPerformative(ACLMessage.CFP);

      public void action() {
            ACLMessage msg = receive(mt);
            if (msg != null) {
                //Process it
                }
            else {
            block();
      }
}
```

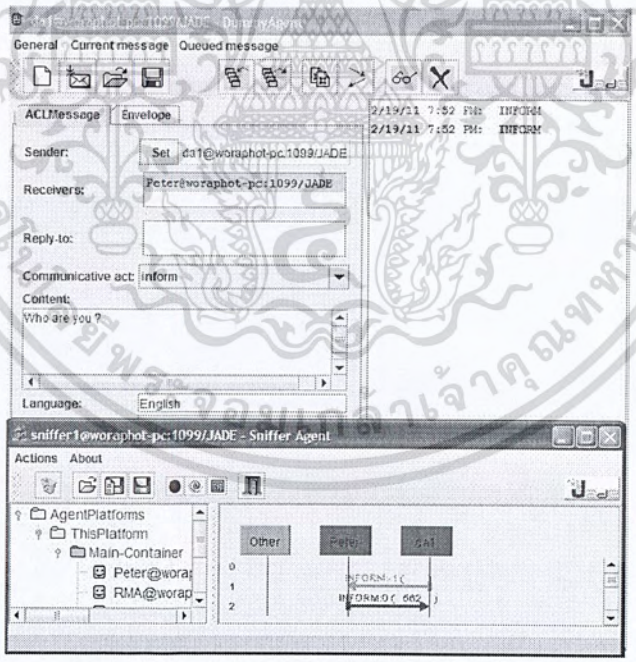
ตัวอย่างที่ 3.9

สร้างเอเจนท์รับข้อความโดยจะรับข้อความที่มี Language = English เท่านั้น

```
import jade.core.AID;
import jade.core.Agent;
import jade.lang.acl.*;
import jade.core.behaviours.*;
import jade.lang.acl.MessageTemplate;


public class Receiver extends Agent{

      protected void setup(){
      addBehaviour(new Receive());
      }
```

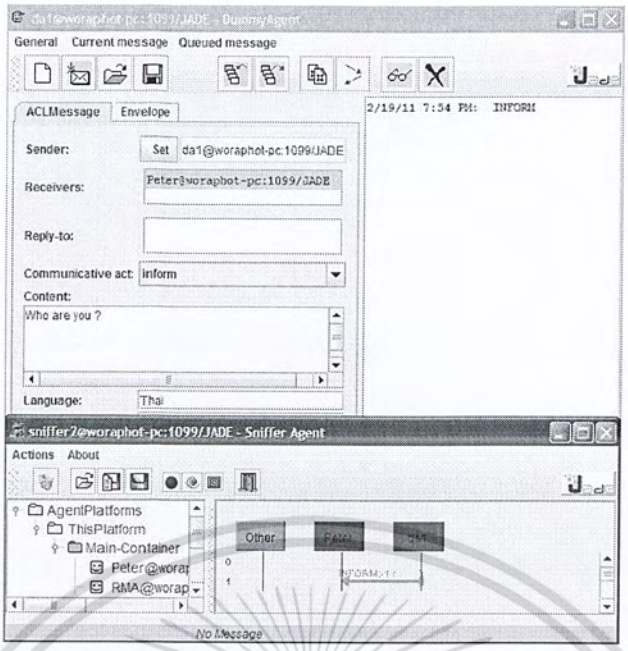
```
private class Receive extends CyclicBehaviour {

      private MessageTemplate mt =
            MessageTemplate.MatchLanguage("English");

      public void action(){

            ACLMessage msg = receive();

            if ((msg!=null)) {
                  System.out.println("You says "+msg.getContent());
                  ACLMessage reply = msg.createReply();
                  reply.setLanguage("English");
                  reply.setContent("Yes, I'm Peter");
                  send(reply);

                  }
            else block();

            }
      }
}
```

ทำการส่งข้อความโดยใช้ Dummy Agent กำหนดให้ Language = English ผลที่ได้ออกมาคือเอเจนท์ตัวรับข้อความจะส่งข้อความกลับมายังตัว Dummy Agent ที่เป็นตัวส่งข้อความ แสดงได้ดังรูปที่ 3.17



รูปที่ 3.17 การตอบสนองของเอเจนท์ เมื่อกำหนดให้ข้อความที่ส่งไปมี Language = English

ทำการส่งข้อความโดยใช้ Dummy Agent กำหนดให้ Language = Thai ผลที่ได้ออกมาคือข้อความจะถูกส่งไปแต่เอเจนท์จะไม่ยอมรับข้อความนั้นจึงไม่มีการส่งข้อความกลับมา แสดงดังรูปที่ 3.18



รูปที่ 3.18 การตอบสนองของเอเจนท์ เมื่อกำหนดให้ข้อความที่ส่งไปมี Language = Thai

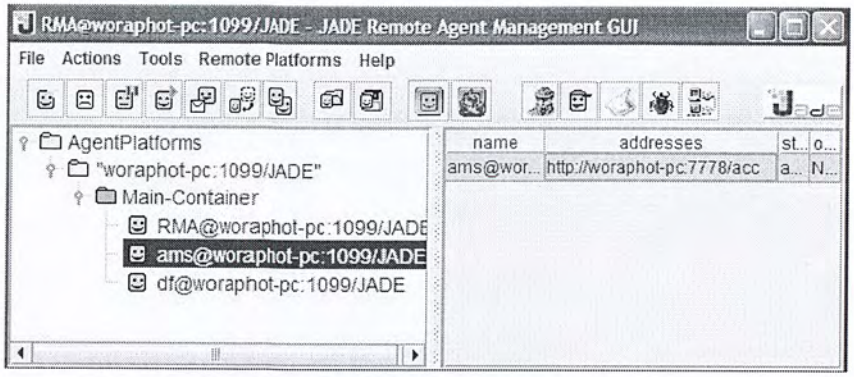
3.7 การสื่อสารระหว่าง Platform

เอเจนท์นั้นสามารถสื่อสารกันระหว่างแพลตฟอร์ม หรืออาจกล่าวได้ว่าสามารถสื่อสารระหว่างคอมพิวเตอร์ได้โดยผ่านอินเตอร์เน็ท มีหลักการเช่นเดียวกับที่ผ่านมาแต่จะมีความซับซ้อนในเรื่องของการกำหนดที่อยู่ผู้รับข้อความ โดยจะต้องระบุชื่อของเอเจนท์ให้ชัดเจนรวมไปถึงแอดเดรสของแพลตฟอร์มที่เอเจนท์นั้นอยู่ด้วยดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 3.10 กำหนดเอเจนท์และที่อยู่ของเอเจนท์เพื่อส่ง ACL Message

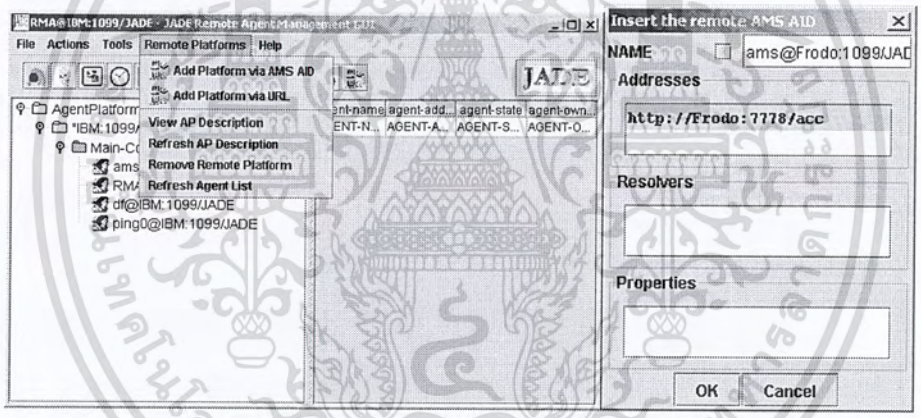
```
AID AD = new AID("loaddata@woraphot-pc:1099/JADE",AID.ISGUID);
AD.addAddresses("http://woraphot-pc:7778/acc");
ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
msg.addReceiver(AD);
```

จากตัวอย่างเป็นการกำหนดชื่อและ AID ของเอเจนท์ คือ loaddata@woraphot-pc:1099/JADE และกำหนดแอดเดรสของแพลตฟอร์มคือ http://woraphot-pc:7778/acc สำหรับแอดเดรสของแพลตฟอร์มใดๆ นั้นสามารถดูได้จากเอเจนท์ ams ของแพลตฟอร์มนั้น แสดงดังรูปที่ 3.19

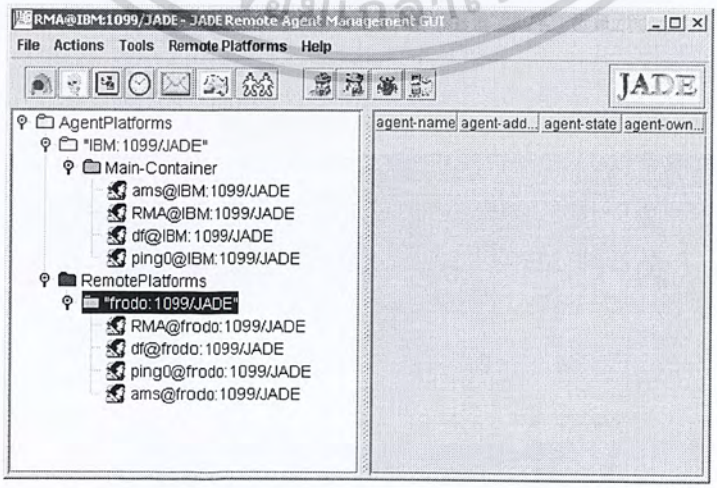


รูปที่ 3.19 แอดเดรสของแพลตฟอร์ม

นอกจากการสื่อสารแล้ว ยังสามารถทำการ Remote เพื่อดูสภาวะของเอเจนท์ที่แพลตฟอร์มอื่นได้โดยใช้ ams AID และ แอดเดรสของแพลตฟอร์ม แสดงดังรูปที่ 3.20 จากนั้นจะเห็นว่ามีโฟลเดอร์อีกแพลตฟอร์มหนึ่งเพิ่มขึ้นมาซึ่งก็คือแพลตฟอร์มที่เราได้กำหนดไว้และจะเห็นรายชื่อของเอเจนท์ของแพลตฟอร์มนั้นด้วย แสดงดังรูปที่ 3.21



รูปที่ 3.20 ขั้นตอนการ Remote Platform



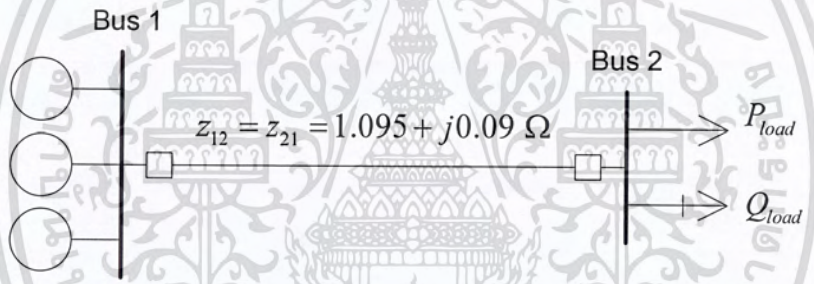
รูปที่ 3.21 ผลของการ Remote Platform

บทที่ 4
การทดลองและผลการทดลอง

ในบทที่ 4 นี้ จะกล่าวถึงวิธีการจำลองระบบไมโครกริดโดยมีเอเจนท์ควบคุมและตัดสินใจ ซึ่งแต่ละเอเจนท์มีหน้าที่แตกต่างกัน ซึ่งในบทนี้ได้อธิบายกระบวนการตัดสินใจของเอเจนท์เพื่อสั่งการให้แหล่งกำเนิดไฟฟ้านั้นตอบสนองภาระทางไฟฟ้าที่มีการเปลี่ยนแปลงในแต่ละช่วงเวลาโดยการรักษาสมดุลของกำลังไฟฟ้าได้อย่างเหมาะสม และคอยรักษาระดับแรงดันด้านโหลดให้คงที่ และในช่วงท้ายได้แสดงผลการจำลองระบบ ณ เวลาต่างๆ เพื่อดูพฤติกรรมการผลิตกำลังไฟฟ้าของแหล่งกำเนิดไฟฟ้าในช่วงเวลาต่างๆ

4.1 แบบจำลองระบบไมโครกริดและเอเจนท์

ระบบไมโครกริดที่จำลองเป็นระบบแบบแยกอิสระ (Stand-alone) แรงดันต่ำ 400 V ที่เป็นลักษณะแบบเรเดียล ดังรูปที่ 4.1



รูปที่ 4.1 แบบจำลองไมโครกริด

ข้อมูล Bus 1 เป็นส่วนของการผลิตกำลังไฟฟ้า ดังตารางที่ 4.1

ตารางที่ 4.1 ข้อมูลของเครื่องกำเนิดไฟฟ้าในไมโครกริด

Generator Type	Capacity		จำนวน (เครื่อง)
	kW	kVAR	
Diesel Generator	12.8	9.6	2
Micro-Hydroelectric Generator	2	-	1

ข้อมูล Bus 2 เป็นส่วนของโหลดทั้งหมด ซึ่งโหลดมีการเปลี่ยนแปลง ณ เวลาต่างๆ ตามตารางที่ 4.2

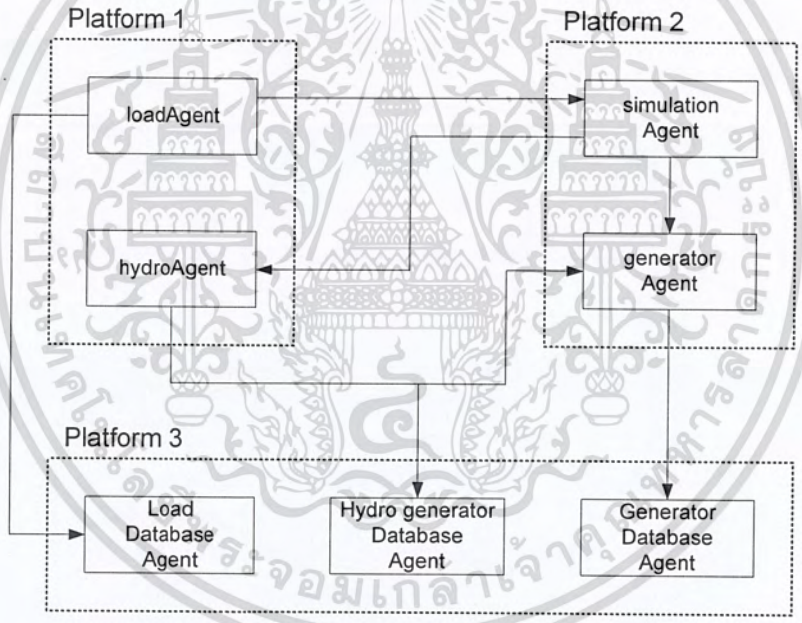
ตารางที่ 4.2 ข้อมูลของโหลดในช่วงเวลาต่างๆ

Time	Load		Time	Load		Time	Load	
	kW	kVAR		kW	kVAR		kW	kVAR
0:00	3.6	1.74	8:00	8.2	3.97	16:00	9.5	4.60
0:15	3.6	1.74	8:15	8.2	3.97	16:15	9.5	4.60
0:30	3.6	1.74	8:30	8.2	3.97	16:30	9.5	4.60
0:45	3.6	1.74	8:45	8.6	4.17	16:45	9.5	4.60
1:00	2.4	1.16	9:00	8.6	4.17	17:00	9.2	4.46
1:15	2.4	1.16	9:15	8.6	4.17	17:15	9.2	4.46
1:30	2.4	1.16	9:30	8.6	4.17	17:30	9.2	4.46
1:45	2.4	1.16	9:45	9.2	4.46	17:45	9.2	4.46
2:00	2.2	1.07	10:00	9.2	4.46	18:00	8.8	4.26
2:15	2.2	1.07	10:15	9.2	4.46	18:15	8.8	4.26
2:30	2.2	1.07	10:30	9.2	4.46	18:30	8.8	4.26
2:45	2.2	1.07	10:45	9.4	4.55	18:45	8.8	4.26
3:00	2.2	1.07	11:00	9.4	4.55	19:00	8.4	4.07
3:15	2.2	1.07	11:15	9.4	4.55	19:15	8.4	4.07
3:30	2.2	1.07	11:30	9.4	4.55	19:30	8.4	4.07
3:45	2.6	1.26	11:45	9.4	4.55	19:45	8.7	4.21
4:00	2.6	1.26	12:00	9.3	4.50	20:00	8.7	4.21
4:15	2.6	1.26	12:15	9.3	4.50	20:15	8.7	4.21
4:30	2.6	1.26	12:30	9.3	4.50	20:30	8.7	4.21
4:45	3.8	1.84	12:45	9.3	4.50	20:45	8.7	4.21
5:00	3.8	1.84	13:00	9.3	4.50	21:00	8.6	4.17
5:15	3.8	1.84	13:15	9.3	4.50	21:15	8.6	4.17
5:30	3.8	1.84	13:30	9.3	4.50	21:30	8.6	4.17
5:45	5.6	2.71	13:45	9.3	4.50	21:45	8.6	4.17
6:00	5.6	2.71	14:00	9.2	4.46	22:00	7.6	3.68
6:15	5.6	2.71	14:15	9.2	4.46	22:15	7.6	3.68
6:30	5.6	2.71	14:30	9.2	4.46	22:30	7.6	3.68
6:45	7.2	3.49	14:45	9.3	4.50	22:45	7.6	3.68
7:00	7.2	3.49	15:00	9.3	4.50	23:00	5.8	2.81
7:15	7.2	3.49	15:15	9.3	4.50	23:15	5.8	2.81
7:30	7.2	3.49	15:30	9.3	4.50	23:30	5.8	2.81
7:45	8.2	3.97	15:45	9.5	4.60	23:45	5.8	2.81

เอเจนท์สำหรับระบบจำลอง มีดังนี้

- **loadAgent** : รับค่าปริมาณความต้องการทางภาระไฟฟ้าที่ต้องการ ณ ขณะนั้น
- **hydroAgent** : ตัดสินใจสั่งการผลิตกำลังของ Small-Hydroelectric Generator
- **generatorAgent** : สั่งการผลิตกำลังไฟฟ้าด้วย Diesel Generator ทั้งสองและตัดสินใจสัดส่วนการผลิต
- **simulationAgent** : รับค่าปริมาณต่างๆ ของโหลดเพื่อวิเคราะห์การไหลของกำลังไฟฟ้าของระบบ
- **databaseAgent** : รับข้อมูลจากเอเจนท์อื่นๆ ได้แก่ loadAgent hydroAgent และ generatorAgent แล้วนำมาเก็บเป็นข้อมูลทางสถิติเพื่อนำไปวิเคราะห์ต่อไป

เอเจนท์เหล่านี้ถูกวางไว้ในแพลตฟอร์มและมีการสื่อสารระหว่างกัน แสดงดังรูปที่ 4.2 สำหรับวิธีการคิดและตัดสินใจของเอเจนท์จะอธิบายในหัวข้อที่ 4.3



รูปที่ 4.2 หน้าที่และการสื่อสารของเอเจนท์

4.2 การวิเคราะห์การไหลของกำลังไฟฟ้าในไมโครกริด

จากรูปที่ 4.1 บัสในระบบมีจำนวน 2 บัส โดยจัดให้บัสที่ 1 เป็น Slack Bus และบัสที่ 2 เป็น Load Bus

จากกระบวนการ Newton-Raphson Power Flow Solution นั้น Slack bus จะมีค่า $\delta_1 = 0$ และจะทราบค่า V_1 และจะต้องทำการสมมุติตัวแปรที่ยังไม่ทราบค่าได้แก่ V_2 และ δ_2 โดยปกติจะสมมุติให้ $V_2^{(0)} = 1.00$ pu. และ $\delta_2^{(0)} = 0.00$ rad

จากสมการที่ (2.8) และ (2.9) สมการกำลังไฟฟ้าในแต่ละบัสของไมโครกริดเป็นดังนี้

$$P_1 = |V_1|^2 |Y_{11}| \cos(\theta_{11}) + |V_1||V_2||Y_{12}| \cos(\theta_{12} - \delta_1 + \delta_2) \quad (4.1)$$

$$P_2 = |V_2||V_1||Y_{21}| \cos(\theta_{21} - \delta_2 + \delta_1) + |V_2|^2 |Y_{22}| \cos(\theta_{22}) \quad (4.2)$$

$$Q_1 = -|V_1|^2 |Y_{11}| \sin(\theta_{11}) - |V_1||V_2||Y_{12}| \sin(\theta_{12} - \delta_1 + \delta_2) \quad (4.3)$$

$$Q_2 = -|V_2||V_1||Y_{21}| \sin(\theta_{21} - \delta_2 + \delta_1) - |V_2|^2 |Y_{22}| \sin(\theta_{22}) \quad (4.4)$$

จากสมการที่ (2.12) ถึงสมการที่ (2.19) ได้สมการ Jacobian ได้ดังนี้

$$\frac{\partial P_2}{\partial \delta_2} = |V_2||V_1||Y_{21}| \sin(\theta_{21} - \delta_2 + \delta_1) \quad (4.5)$$

$$\frac{\partial Q_2}{\partial \delta_2} = |V_2||V_1||Y_{21}| \cos(\theta_{21} - \delta_2 + \delta_1) \quad (4.6)$$

$$\frac{\partial P_2}{\partial |V_2|} = |V_1||Y_{21}| \cos(\theta_{21} - \delta_2 + \delta_1) + 2|V_2||Y_{22}| \cos(\theta_{22}) \quad (4.7)$$

$$\frac{\partial Q_2}{\partial |V_2|} = -|V_1||Y_{21}| \cos(\theta_{21} - \delta_2 + \delta_1) - 2|V_2||Y_{22}| \cos(\theta_{22}) \quad (4.8)$$

P_{sch} และ Q_{sch} ได้จากปริมาณโหลด P_{load} และ Q_{load} โดยที่ $P_{sch} = -P_{load}$ และ $Q_{sch} = -Q_{load}$ จากสมการที่ (2.20) และ (2.21) จะได้สมการดังนี้

$$\Delta P_2^{(k)} = P_2^{(sch)} - P_2^{(k)} \quad (4.9)$$

$$\Delta Q_2^{(k)} = Q_2^{(sch)} - Q_2^{(k)} \quad (4.10)$$

จากสมการที่ (2.7) เขียนเป็นความสัมพันธ์ได้ดังนี้

$$\begin{bmatrix} \Delta P_2^{(k)} \\ \Delta Q_2^{(k)} \end{bmatrix} = \begin{bmatrix} \frac{\partial P_2}{\partial \delta_2}^{(k)} & \frac{\partial P_2}{\partial |V_2|}^{(k)} \\ \frac{\partial Q_2}{\partial \delta_2}^{(k)} & \frac{\partial Q_2}{\partial |V_2|}^{(k)} \end{bmatrix} \begin{bmatrix} \Delta \delta_2^{(k)} \\ \Delta |V_2|^{(k)} \end{bmatrix} \quad (4.11)$$

จากสมการที่ (4.11) แก้สมการหาค่า $\Delta \delta_2^{(k)}$ และ $\Delta |V_2^{(k)}|$ แล้วทำการหาค่าแรงดันและมุมกำลังที่บัส 2 ได้ดังนี้

$$\delta_2^{(k+1)} = \delta_2^{(k)} + \Delta \delta_2^{(k)} \quad (4.12)$$

$$|V_2^{(k+1)}| = |V_2^{(k)}| + \Delta |V_2^{(k)}| \quad (4.13)$$

จากทั้งหมดนี้ คำนวณซ้ำเป็นจำนวน 100 ครั้ง จะได้ค่าที่เข้าใกล้คำตอบมากที่สุด จากนั้นทำการหาค่ากำลังการผลิตจากสมการที่ (4.1) และ (4.3) และจากสมการที่ (2.27) และ (2.28) จะได้สมการการไหลของกำลังไฟฟ้าได้ดังนี้

$$P_{12} = \left(V_1^2 \cos\left(-\gamma_{12}\right)\right) - \left(V_1 V_2 \cos\left(\delta_1 - \delta_2 - \gamma_{12}\right)\right) \quad (4.14)$$

$$P_{21} = \left(V_2^2 \cos\left(-\gamma_{21}\right)\right) - \left(V_2 V_1 \cos\left(\delta_2 - \delta_1 - \gamma_{21}\right)\right) \quad (4.15)$$

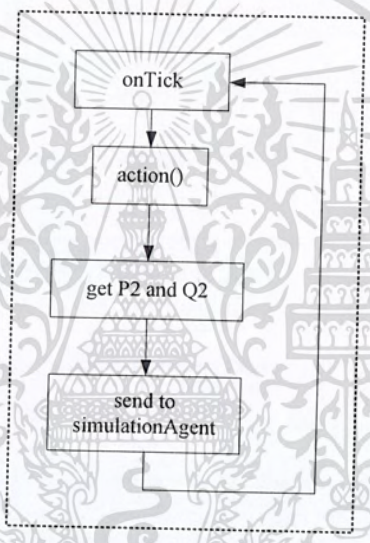
$$Q_{12} = \left(V_1^2 \sin\left(-\gamma_{12}\right)\right) - \left(V_1 V_2 \sin\left(\delta_1 - \delta_2 - \gamma_{12}\right)\right) \quad (4.16)$$

$$Q_{21} = \left(V_2^2 \sin\left(-\gamma_{21}\right)\right) - \left(V_2 V_1 \sin\left(\delta_2 - \delta_1 - \gamma_{21}\right)\right) \quad (4.17)$$

4.3 กระบวนการตัดสินใจของเอเจนท์

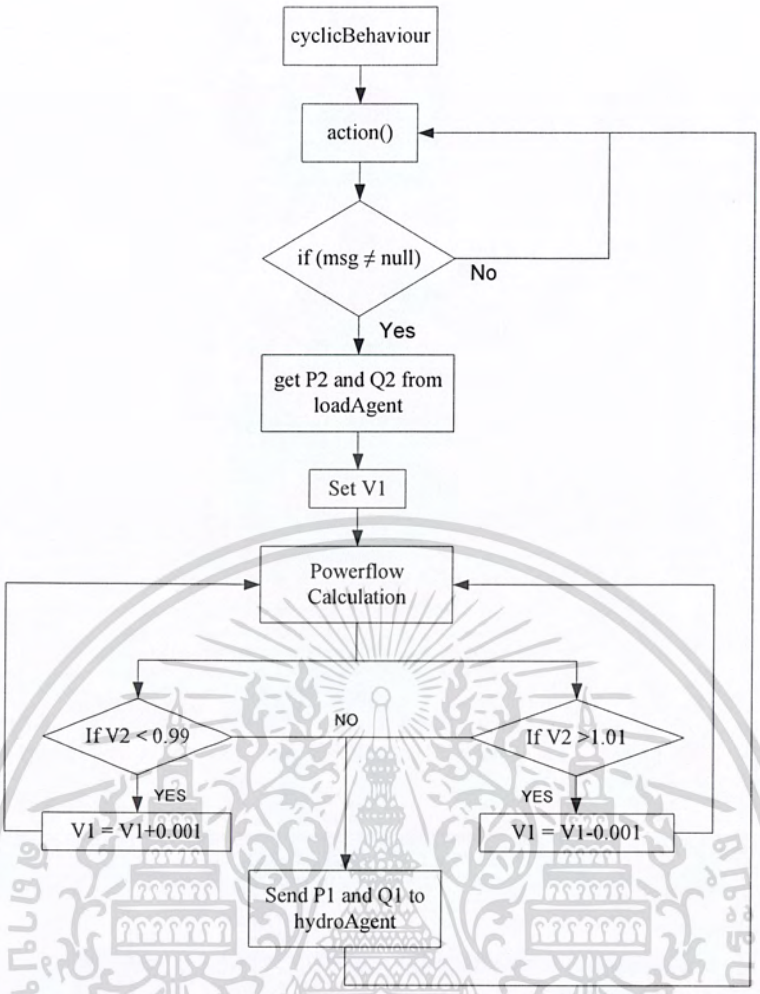
จากรูปที่ 4.2 ได้แสดงลำดับการสื่อสารของเอเจนท์ โดยเอเจนท์แต่ละตัวจะมีกระบวนการทำงานดังนี้

- loadAgent จะรับค่าโหลดที่ต้องการ ณ เวลานั้นดังที่แสดงในตารางที่ 4.2 และส่งข้อมูลโหลดไปยัง simulationAgent แสดงเป็น Flow Chart ดังรูปที่ 4.3



รูปที่ 4.3 Flow Chart การทำงานของ loadAgent

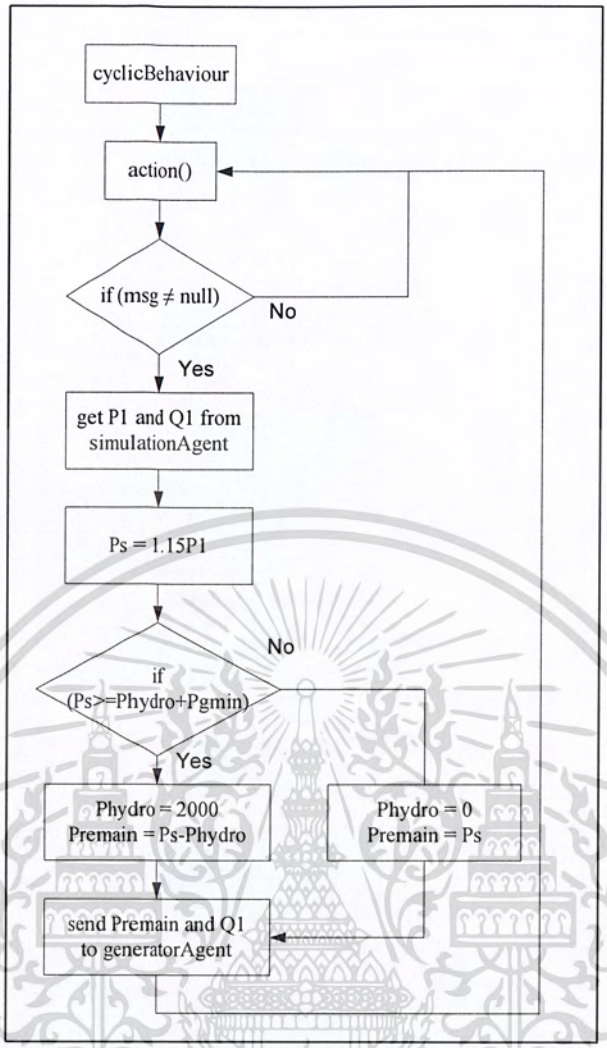
- simulationAgent จะรับค่าโหลดจาก loadAgent ทำการวิเคราะห์การไหลกำลังไฟฟ้า จะเห็นว่าได้ผลเฉลยกำลังไฟฟ้าที่บัส 1 แต่จะเห็นได้อีกว่าแรงดันปลายทางจะเกิดการแกว่ง ถ้าต้องการรักษาระดับแรงดันคงที่ที่โหลดบัส 1.00 pu. จะต้องมีการปรับแต่งแรงดันที่บัส 1 (Voltage set-point) โดยเมื่อแรงดันลดก็จะต้องปรับแรงดันต้นทางให้สูงขึ้นหรือถ้าแรงดันเพิ่มขึ้นก็จะต้องปรับแรงดันต้นทางให้ลดลงเพื่อให้แรงดันปลายทางนั้นมีการแกว่งเล็กน้อยประมาณ 1% ผลเฉลยของกำลังไฟฟ้าที่บัส 1 ที่เป็นส่วนของการผลิตกำลังไฟฟ้านั้นจะถูกส่งไปที่ hydroAgent และ generatorAgent ตามลำดับ แสดงเป็น Flow Chart ดังรูปที่ 4.4



รูปที่ 4.4 Flow Chart การทำงานของ simulationAgent

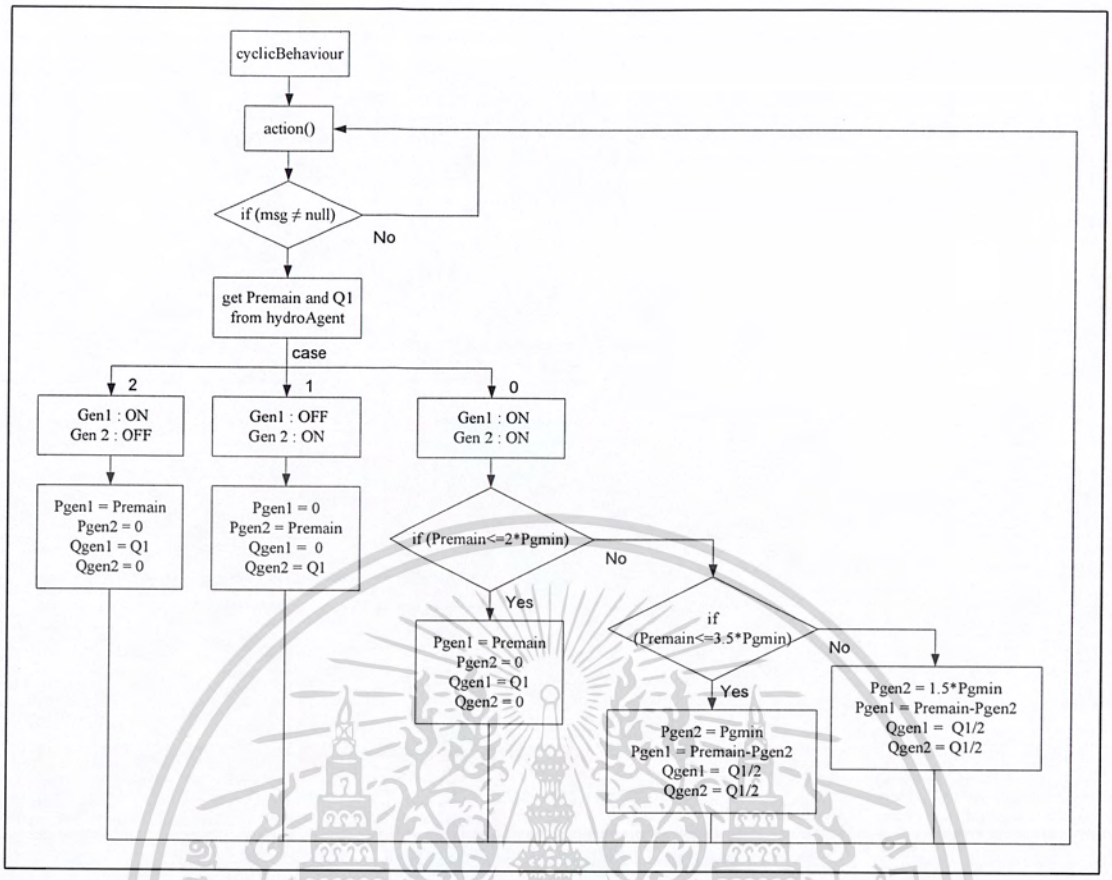
ในส่วนของการผลิตกำลังไฟฟ้านั้น เมื่อ Micro-Hydroelectric Generator เดินเครื่อง จะเดินเครื่องที่พิกัดกำลัง ส่วน Diesel Generator นั้น เครื่องที่ 1 จะเดินเครื่องตามกำลังที่ต้องการ ณ เวลานั้น ส่วนเครื่องที่ 2 นั้นเดินเครื่องในลักษณะเป็นกำลังฐาน (Base Load) ในระดับหนึ่ง ส่วนกำลังไฟฟ้ารีแอคทีฟนั้น Diesel Generator จะผลิตทั้งหมด หน้าที่ของเอเจนท์ในส่วนของการผลิตกำลังต่างๆ มีดังนี้

- hydroAgent จะรับค่ากำลังการผลิตทั้งหมดเพื่อที่จะสั่งการจาก simulationAgent และจะต้องมีการเผื่อกำลังผลิต (Spinning reserve) ประมาณ 15% ของกำลังการผลิตที่คำนวณได้จาก simulationAgent ลำดับการตัดสินใจเดินเครื่องของ Micro-Hydroelectric Generator นั้นเป็นไปตาม Flow Chart ดังรูปที่ 4.5 จากนั้นส่วนค่ากำลังการผลิตที่ต้องผลิตอีกนั้นจะถูกส่งไปที่ generatorAgent เพื่อสั่งการต่อไป

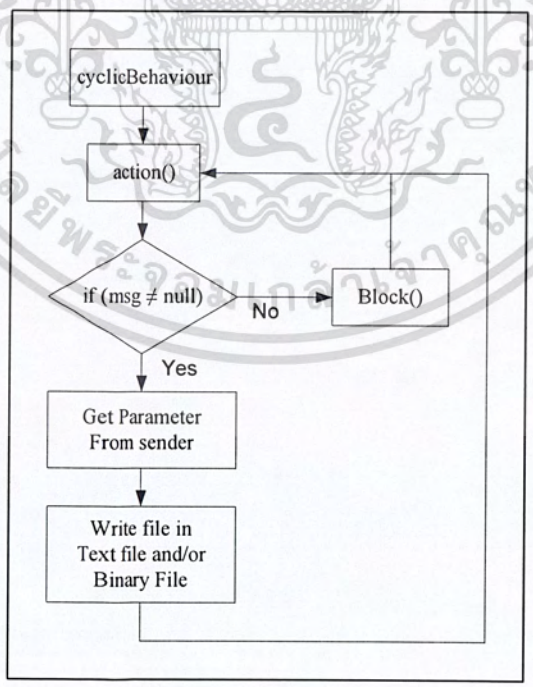


รูปที่ 4.5 Flow Chart การทำงานของ hydroAgent

- generatorAgent จะรับค่ากำลังการผลิตจาก hydroAgent และสั่งการเดินเครื่อง Diesel Generator โดยแบ่งเงื่อนไขแบ่งออกไปคือ กรณีที่สามารถเดินเครื่องได้ทั้งสองเครื่อง และกรณีเครื่องใดเครื่องหนึ่งไม่สามารถเดินเครื่องได้ ลำดับการตัดสินใจการเดินเครื่องนั้นได้แสดงใน Flow Chart ดังรูปที่ 4.6
- databaseAgent จะรับค่าพารามิเตอร์ต่างๆ ได้แก่ค่าโหลดและค่ากำลังการผลิตของเครื่องกำเนิดไฟฟ้าต่างๆ จากเอเจนท์อื่นๆ แบ่งออกเป็น loaddatabaseAgent hydrodatabaseAgent และ generatordatabaseAgent เพื่อทำการเก็บข้อมูลโดยมีลำดับการทำงานเป็นไปตาม Flow Chart ดังรูปที่ 4.7



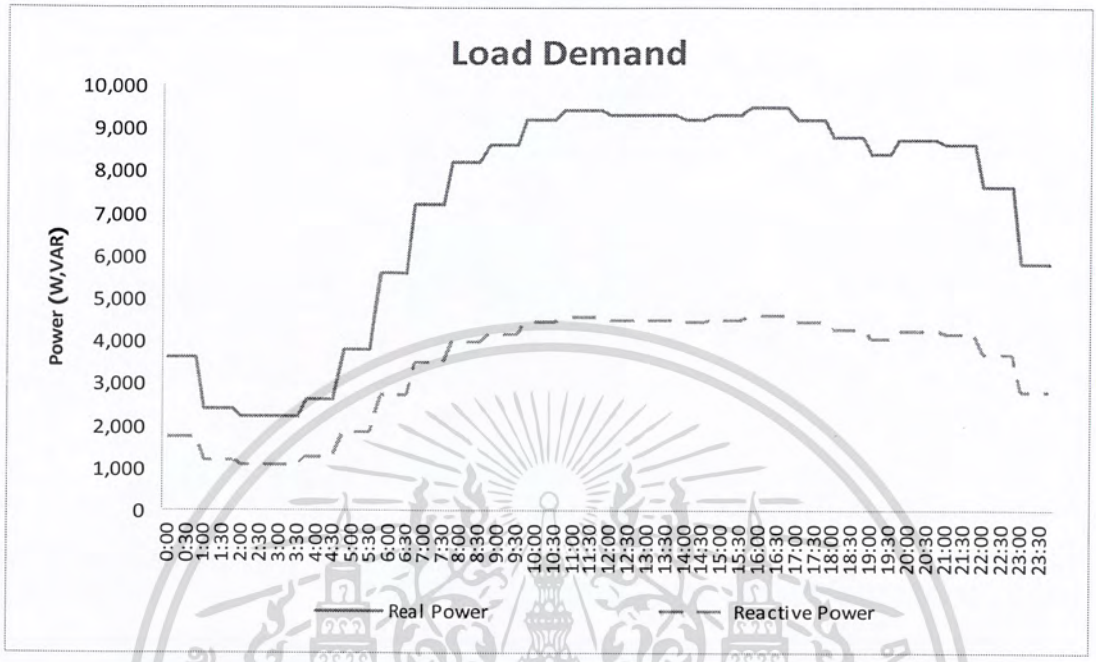
รูปที่ 4.6 Flow Chart การทำงานของ generatorAgent



รูปที่ 4.7 Flow Chart การทำงานของ databaseAgent

4.4 ผลการจำลองระบบ

จากการจำลองระบบนั้น ได้ความสัมพันธ์ระหว่างความต้องการกำลังไฟฟ้ากับเวลาในช่วงต่างๆ ดังรูปที่ 4.8



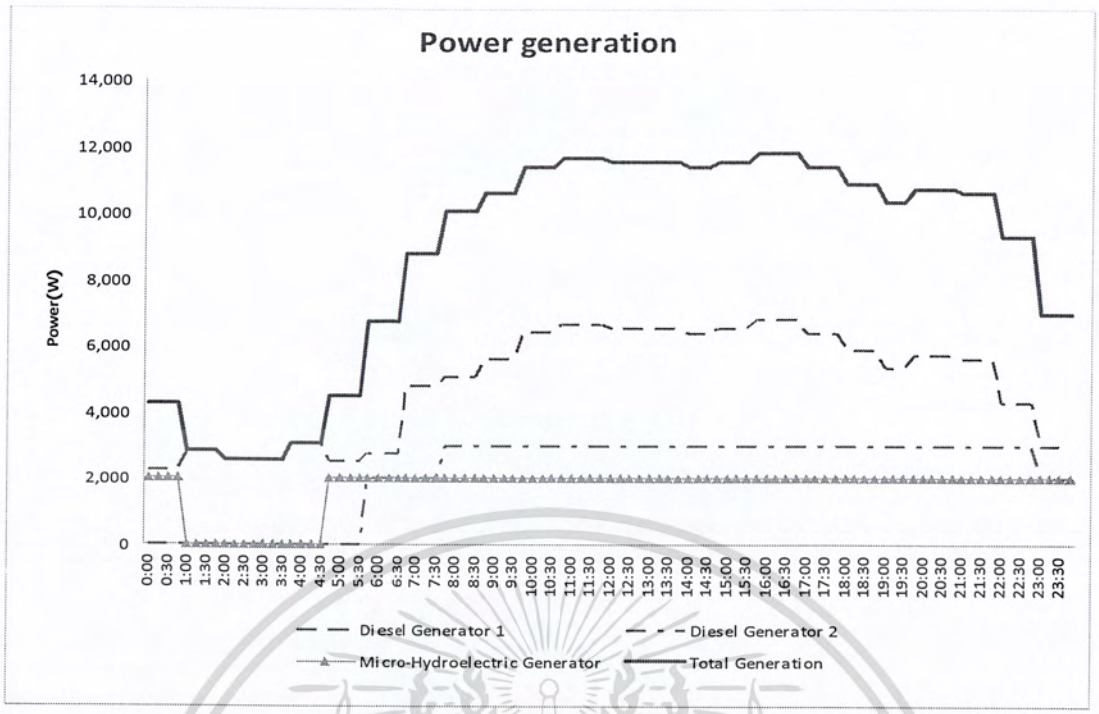
รูปที่ 4.8 ความต้องการกำลังไฟฟ้าในแต่ละช่วงเวลาที่ได้จากการจำลองระบบ

กรณีศึกษาการจำลองได้แบ่งออกเป็น 3 กรณี ดังต่อไปนี้

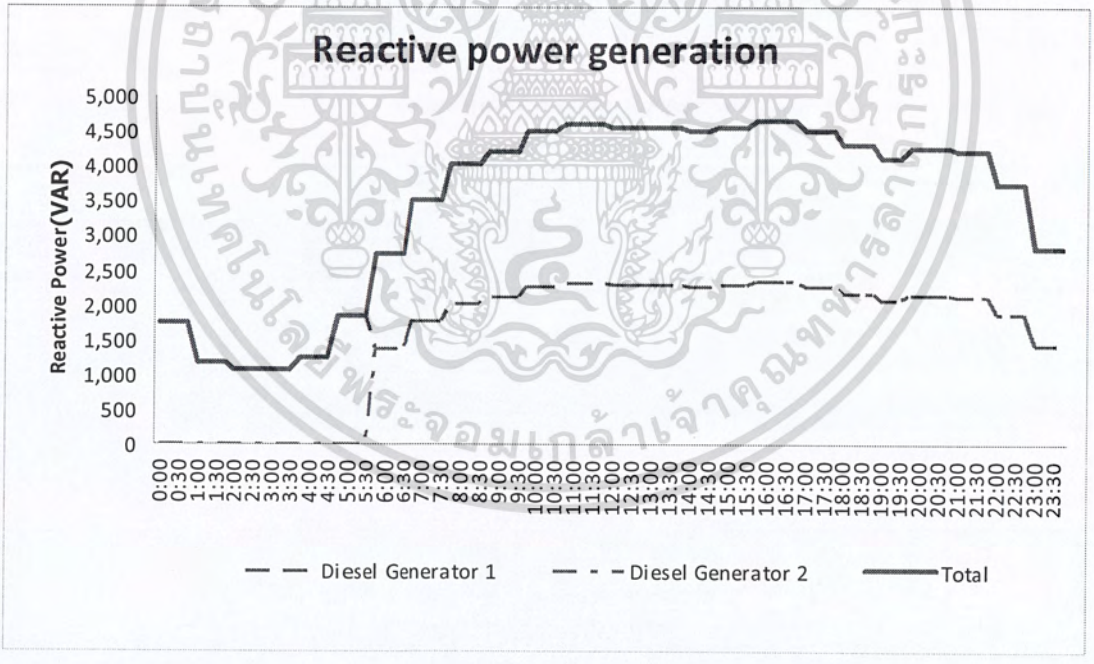
1. เครื่องกำเนิดไฟฟ้าสามารถเดินเครื่องได้ทั้งหมด
2. Diesel Generator 1 ไม่สามารถเดินเครื่องได้
3. Diesel Generator 2 ไม่สามารถเดินเครื่องได้

4.4.1 เครื่องกำเนิดไฟฟ้าสามารถเดินเครื่องได้ทั้งหมด

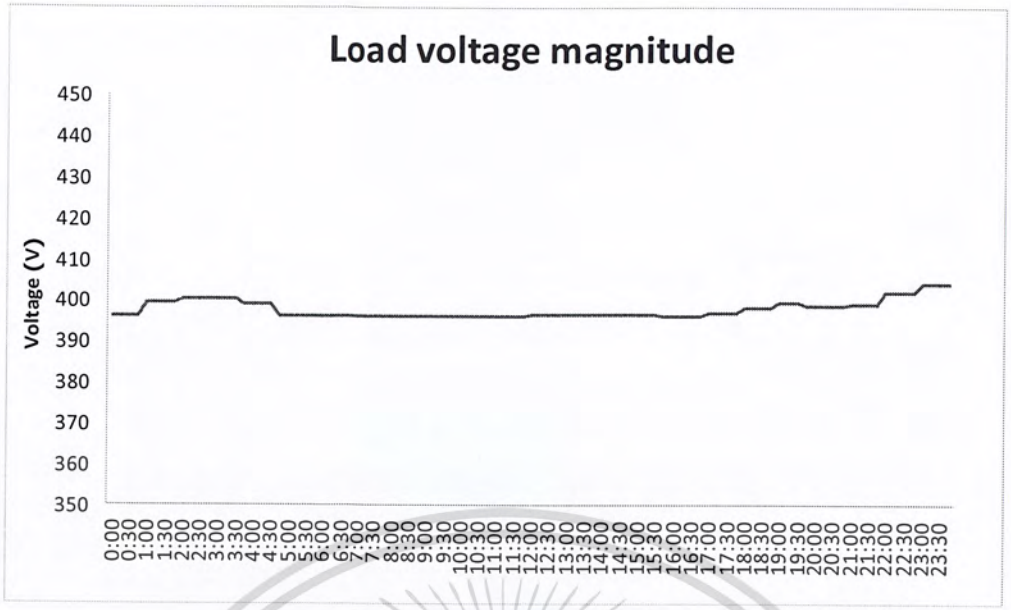
เครื่องกำเนิดไฟฟ้าทุกตัวสามรถเดินเครื่องได้ โดยมีเครื่องกำเนิดไฟฟ้า Diesel Generator 1 เดินเครื่องตลอดและเป็นเครื่องกำเนิดไฟฟ้าที่เป็นตัวปรับตามโหลด ส่วนเครื่องกำเนิดไฟฟ้า Diesel Generator 2 และ Micro-Hydroelectric Generator จะผลิตกำลังในลักษณะเป็นกำลังการผลิตฐาน (Base load) ตามเงื่อนไขที่ได้กำหนดไว้ ส่วนกำลังไฟฟ้ารีแอคทีฟนั้นจะมีเพียง Diesel Generator ทั้งสองที่สามารถผลิตออกมาได้ โดยผลิตออกมาตามเงื่อนไขที่ได้กำหนดไว้ ผลการจำลองการผลิตกำลังไฟฟ้าจริง กำลังไฟฟ้ารีแอคทีฟ และแรงดันที่โหลดบัส เป็นไปตามรูปที่ 4.9 4.10 และ 4.11 ตามลำดับ



รูปที่ 4.9 การผลิตกำลังไฟฟ้าจริง กรณีเครื่องกำเนิดไฟฟ้าสามารถเดินเครื่องได้ทั้งหมด



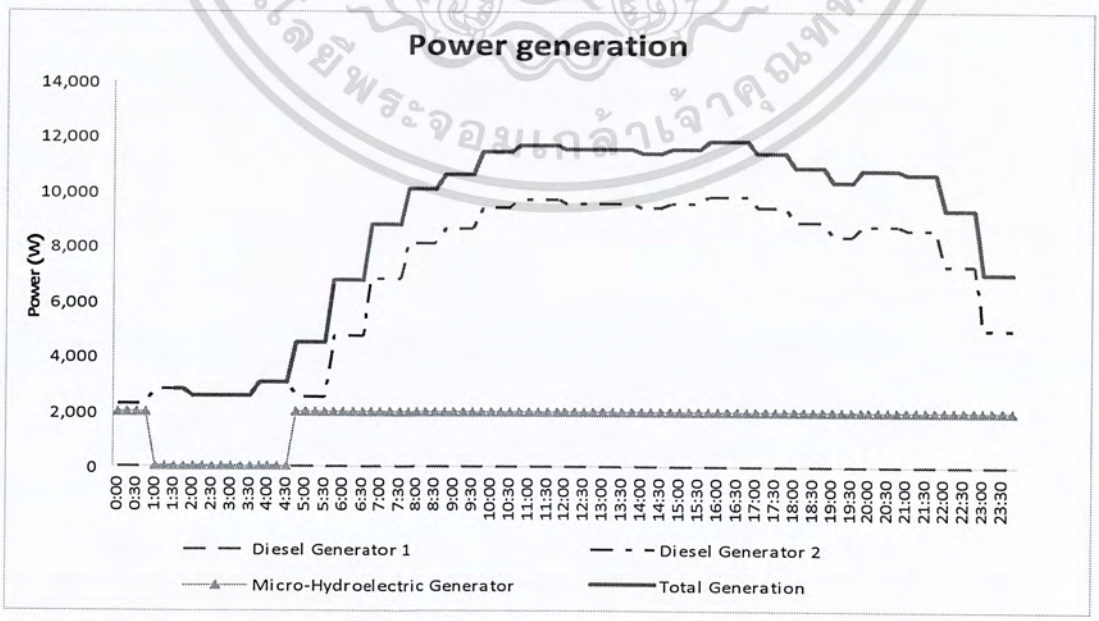
รูปที่ 4.10 การผลิตกำลังไฟฟ้ารีแอคทีฟ กรณีเครื่องกำเนิดไฟฟ้าสามารถเดินเครื่องได้ทั้งหมด



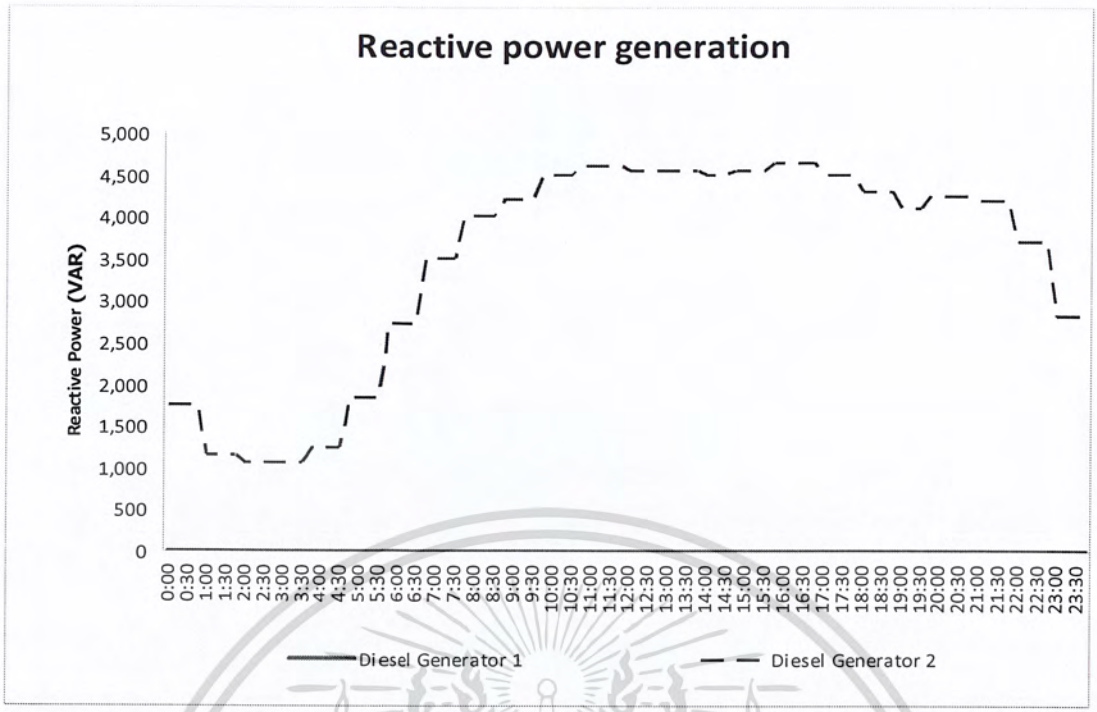
รูปที่ 4.11 ขนาดแรงดันที่โหลดบัส กรณีเครื่องกำเนิดไฟฟ้าสามารถเดินเครื่องได้ทั้งหมด

4.4.2 Diesel Generator 1 ไม่สามารถเดินเครื่องได้

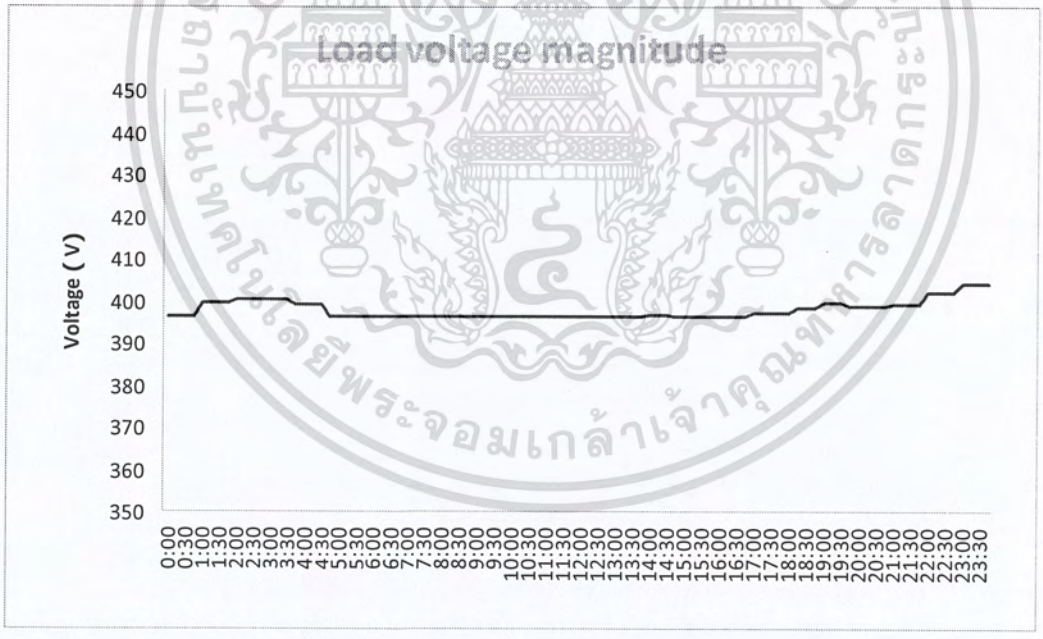
เครื่องกำเนิดไฟฟ้า Diesel Generator 1 จะถูกปิดการเดินเครื่องตลอด และ Diesel Generator 2 จะเดินเครื่องแทนและทำหน้าที่เป็นเครื่องกำเนิดไฟฟ้าที่เป็นตัวปรับตามโหลด ส่วนเครื่องกำเนิดไฟฟ้า Micro-Hydroelectric Generator จะผลิตกำลังในลักษณะเป็นกำลังการผลิตฐาน (Base load) ตามเงื่อนไขที่กำหนด และส่วนกำลังไฟฟ้ารีแอคทีฟจะถูกผลิตออกมาโดย Diesel Generator 2 เท่านั้น ผลการจำลองการผลิตกำลังไฟฟ้าจริง กำลังไฟฟ้ารีแอคทีฟ และแรงดันที่โหลดบัส เป็นไปตามรูปที่ 4.12 4.13 และ 4.14 ตามลำดับ



รูปที่ 4.12 การผลิตกำลังไฟฟ้าจริง กรณี Diesel Generator 1 ไม่สามารถเดินเครื่องได้



รูปที่ 4.13 การผลิตกำลังไฟฟ้ารีแอคทีฟ กรณี Diesel Generator 1 ไม่สามารถเดินเครื่องได้

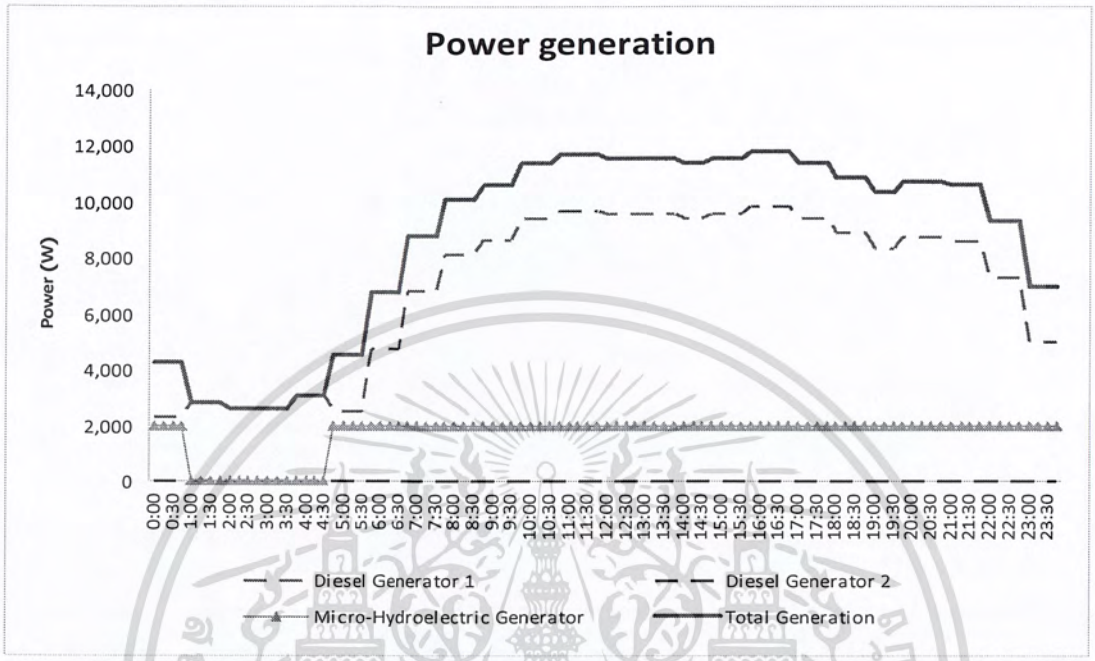


รูปที่ 4.14 ขนาดแรงดันที่โหลดบัส กรณี Diesel Generator 1 ไม่สามารถเดินเครื่องได้

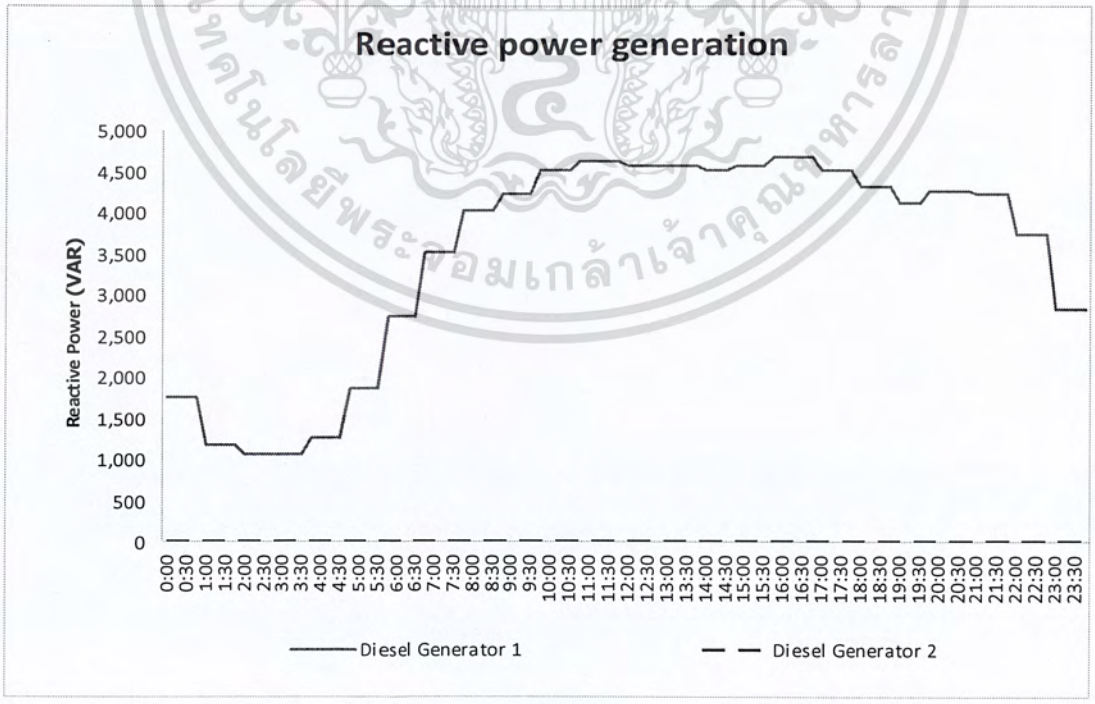
4.4.3 Diesel Generator 2 ไม่สามารถเดินเครื่องได้

เครื่องกำเนิดไฟฟ้า Diesel Generator 2 จะถูกปิดการเดินเครื่องตลอด และ Diesel Generator 1 จะเดินเครื่องแทนและทำหน้าที่เป็นเครื่องกำเนิดไฟฟ้าที่เป็นตัวปรับตามโหลด ส่วนเครื่องกำเนิดไฟฟ้า Micro-Hydroelectric Generator จะผลิตกำลังในลักษณะเป็นกำลังการ

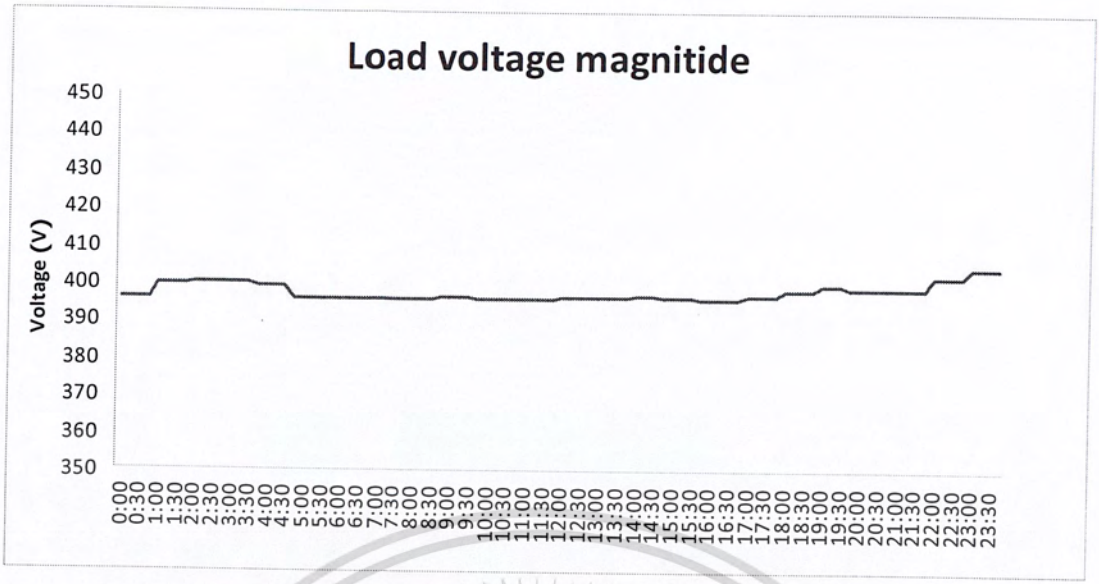
ผลิตฐาน (Base load) ตามเงื่อนไขที่กำหนด และส่วนกำลังไฟฟ้ารีแอคทีฟจะถูกผลิตออกมาโดย Diesel Generator 1 เท่านั้น ผลการจำลองการผลิตกำลังไฟฟ้าจริง กำลังไฟฟ้ารีแอคทีฟ และแรงดันที่โหลดบัส เป็นไปตามรูปที่ 4.15 4.16 และ 4.17 ตามลำดับ



รูปที่ 4.15 การผลิตกำลังไฟฟ้าจริง กรณี Diesel Generator 2 ไม่สามารถเดินเครื่องได้



รูปที่ 4.16 การผลิตกำลังไฟฟ้ารีแอคทีฟ กรณี Diesel Generator 2 ไม่สามารถเดินเครื่องได้



รูปที่ 4.17 ขนาดแรงดันที่โหลดบัส กรณี Diesel Generator 2 ไม่สามารถเดินเครื่องได้

4.5 สรุปผลการจำลอง

จากการจำลองระบบทั้ง 3 กรณีจะพบว่าโปรแกรมเอเจนท์สามารถตัดสินใจในการดำเนินการสั่งจ่ายกำลังการผลิตได้อย่างเพียงพอตามเงื่อนไขที่ได้กำหนดไว้ในโปรแกรมเอเจนท์ และมีการรักษาระดับแรงดันที่บัสของโหลดให้คงที่ตลอดเวลา

บทที่ 5
บทสรุปและข้อเสนอแนะ

ไมโครกริด (Microgrid) เป็นระบบไฟฟ้าทางเลือกขนาดเล็กระดับระบบจำหน่ายโดยใช้แหล่งกำเนิดพลังงานขนาดเล็ก (Distributed Energy Resource : DER) ซึ่งใช้แหล่งพลังงานทางเลือกต่างๆ ที่มีตั้งแต่ ระบบความร้อนร่วม (Combined Heat and Power : CHP),ระบบไม่ใช้ความร้อนร่วม (Non-Combined Heat and Power) และอุปกรณ์สำรองพลังงาน (Storage Device) โดยส่งจ่ายทั้งพลังงานไฟฟ้าและพลังงานความร้อนให้กับผู้บริโภค การเชื่อมต่อของไมโครกริดมีทั้งแบบแยกอิสระ (Stand-alone หรือ Islanded) และแบบเชื่อมต่อกับกริดหลัก (Grid-connected) ระบบไมโครกริดนั้นเป็นระบบที่มีเสถียรภาพ มีความน่าเชื่อถือ มีความยืดหยุ่นและมีความปลอดภัย ระบบส่งจ่ายกำลังไฟฟ้ามีระยะทางไม่ไกลมาก จึงมีกำลังการสูญเสียในสายส่งต่ำ และไมโครกริดสามารถดำเนินการได้อย่างอิสระจากการควบคุมของกริดหลัก

การควบคุมส่วนต่างๆ ในไมโครกริด เช่น ส่วนของการผลิต ส่วนของผู้บริโภค จะควบคุมให้แยกอิสระจากกัน แต่ละส่วนสามารถตัดสินใจได้ด้วยตนเองโดยไม่ขึ้นกับคำสั่งจากส่วนอื่นๆ และจะต้องมีการแลกเปลี่ยนข้อมูลสื่อสารกันเพื่อประสานงานกันในระบบ ยกตัวอย่าง เช่น แหล่งกำเนิดไฟฟ้าขนาดเล็ก (Microsource) จะมีส่วนสำหรับควบคุมอยู่เรียกว่า Microsource Controller (MC) โดยแหล่งกำเนิดไฟฟ้าขนาดเล็กนี้จะรับคำสั่งจาก MC นี้เท่านั้น แต่ MC นี้สามารถสื่อสารกับ MC ของแหล่งกำเนิดไฟฟ้าขนาดเล็กอื่นๆได้ เพื่อประสานการทำงานในการสั่งจ่ายกำลังไฟฟ้าได้อย่างเหมาะสม นอกจากนี้ยังมีการควบคุมส่วนกลางหรือ Central Controller (CC) เป็นส่วนที่ควบคุมภาพรวมในไมโครกริด คอยดูแลตรวจสอบข้อมูลจากแหล่งกำเนิดไฟฟ้าและโหลด ควบคุมในส่วนโหมดการทำงานแบบแยกอิสระและแบบเชื่อมต่อกับกริดหลัก เป็นต้น

ในการควบคุมสั่งการส่วนต่างๆ ของไมโครกริดนั้นได้นำเทคโนโลยีระบบมัลติเอเจนท์ (Multi-Agent System) ซึ่งมีลักษณะการประมวลผลแบบกระจาย โดยภายในระบบจะมีซอฟท์แวร์เอเจนท์ ซึ่งเป็นเสมือนตัวแทนผู้ใช้งานซึ่งมีความเป็นอิสระ สามารถตัดสินใจด้วยตนเองได้โดยไม่ขึ้นกับคำสั่งอื่นใดและมีความสามารถสื่อสารกันระหว่างเอเจนท์ เพื่อให้สามารถสั่งการและดำเนินการร่วมกันรวมถึงมีการรับรู้และตอบสนองต่อสิ่งแวดล้อมและแสดงพฤติกรรมออกมาเพื่อให้บรรลุตามวัตถุประสงค์ หากเปรียบเทียบกับมนุษย์แล้วเอเจนท์ก็เหมือนกับมนุษย์คนหนึ่ง และส่วนของโปรแกรมเอเจนท์ก็เหมือนกับสมองของมนุษย์ เพื่อคิดสั่งการให้เอเจนท์แสดงพฤติกรรมต่างๆ อาจเรียกได้ว่าเอเจนท์เป็นโปรแกรมที่มีลักษณะเป็นเหมือนสังคม ซึ่งสามารถติดต่อสื่อสารเพื่อประสานงานได้เหมือนกับมนุษย์ช่วยกันทำงานเป็นกลุ่ม ดังนั้น การที่เอเจนท์จะทำงานแทนผู้ใช้ได้ดีหรือไม่นั้น ขึ้นอยู่กับความฉลาดของโปรแกรมเอเจนท์เอง การพัฒนาโปรแกรมเอเจนท์ต้องเขียนโปรแกรมให้มีฉลาดพอที่จะทำงานแทนผู้ใช้ได้

ชุดโปรแกรมที่ใช้พัฒนาระบบมัลติเอเจนท์นั้นได้ใช้ชุดพัฒนาจาวาเอเจนท์ (Java Agent Development : JADE) ซึ่งเป็นชุดพัฒนาซอฟท์แวร์โดย Telecom Italia (TILAB) ซึ่งเป็นตัวพัฒนาระบบมัลติเอเจนท์ตามมาตรฐานของ FIPA (THE FOUNDATION FOR INTELLIGENT, PHYSICAL AGENTS) ชุดพัฒนานี้ได้พัฒนาโดยใช้ภาษาจาวาซึ่งมีลักษณะรูปแบบโปรแกรมเชิงวัตถุ (Object-Oriented-Programming : OOP) โดยจะต้องสร้างรูปแบบของวัตถุ (Object) ที่เรียกว่าคลาส (Class) ของเอเจนท์ขึ้นมา แล้วทำการเรียกวัตถุ ซึ่งในที่นี้ก็คือเอเจนท์ให้ออกมาทำงานตามโปรแกรมที่ได้เขียนไว้

จากการจำลองการทำงานระบบไมโครกริดนั้น ได้ทำการจำลองระบบไมโครกริดในโหมดแยกอิสระ (Stand-alone) โดยมีแหล่งกำเนิดไฟฟ้าชนิดต่างๆ และโหลดอยู่ภายในระบบ โดยมุ่งเน้นให้ระบบมีเสถียรภาพในสภาวะคงตัว (Steady-State) โดยแต่ละส่วนจะถูกควบคุมสั่งการโดยเอเจนท์ต่างๆ และสื่อสารประสานงานกันโดยมีหลักการโดยสรุปดังนี้

1. เอเจนท์ที่ควบคุมโหลดจะรับข้อมูลปริมาณความต้องการทางภาระไฟฟ้า ณ ขณะนั้น และทำการส่งข้อมูลของโหลดยังเอเจนท์สำหรับการจำลอง (Simulation Agent)
2. เมื่อเอเจนต์สำหรับการจำลองได้รับค่าปริมาณความต้องการทางภาระไฟฟ้า แล้วจะทำการจำลองการไหลของกำลังไฟฟ้า (Power flow solution) ทำให้ทราบค่าพารามิเตอร์ในแต่ละบัส และการไหลของกำลังไฟฟ้าในสายส่ง ถ้าพารามิเตอร์บางตัวมีค่าไม่เหมาะสม เช่น เกิดแรงดันตกที่บัสปลายทางหรือกระแสรีแอคทีฟไหลวนมากเกินไป จะต้องมีการปรับค่าพารามิเตอร์บางตัวใหม่เพื่อให้ค่าพารามิเตอร์นั้นๆ มีค่าที่เหมาะสม จากการวิเคราะห์ด้วยกระบวนการดังกล่าวจะได้ค่ากำลังไฟฟ้าที่ต้องทำการผลิตทั้งหมดเพื่อรองรับกับความต้องการทางภาระไฟฟ้า และค่ากำลังไฟฟ้าที่ผลิตนี้จะถูกส่งไปยังเอเจนท์ที่ควบคุมหน่วยผลิต
3. เมื่อเอเจนท์ที่ควบคุมส่วนการผลิต ได้รับค่ากำลังไฟฟ้าที่ต้องผลิตมาแล้ว จะต้องมีส่วนเผื่อการผลิต (Spinning reserve) อีกประมาณ 15% เพื่อสำรองการเปลี่ยนแปลงของภาระทางไฟฟ้า และเอเจนต์แต่ละตัวจะต้องสื่อสารกันว่าจะต้องแบ่งสัดส่วนการผลิตอย่างไร โดยมีเงื่อนไขว่า
   - จะต้องมีเครื่องกำเนิดไฟฟ้าดีเซลที่เดินเครื่องตลอดเวลาเพื่อรองรับการจ่ายกำลังไฟฟ้ารีแอคทีฟ
   - เมื่อเดินเครื่องกำเนิดไฟฟ้าพลังงานน้ำขนาดเล็กจะต้องจ่ายกำลังไฟฟ้าที่พิกัด
   - ให้เครื่องกำเนิดไฟฟ้าดีเซลอย่างน้อย 1 เครื่อง เดินเครื่องตามสภาวะที่ภาระทางไฟฟ้ามีการเปลี่ยนแปลง
4. จากข้อมูลภาระทางไฟฟ้าและข้อมูลการผลิตของแต่ละหน่วยการผลิตจะถูกส่งมาที่ส่วนของฐานเก็บข้อมูล โดยสร้างเป็นเอเจนท์สำหรับรับค่าและเก็บข้อมูลต่างๆ ในแต่ละช่วงเวลาไว้ในที่เดียวกัน

เมื่อทำการรันโปรแกรมเอเจนท์แล้วจะเห็นว่าเมื่อเอเจนท์ของโหลดรับค่าปริมาณความต้องการทางภาระไฟฟ้ามาแล้ว เอเจนท์ตัวอื่นๆ จะมีการตอบสนองต่อการเปลี่ยนแปลงทันทีโดยแสดงพฤติกรรมที่ได้ระบุไว้ในเอเจนท์แต่ละตัว เมื่อนำผลลัพธ์มาวิเคราะห์จะพบว่าการผลิตนั้นสามารถรองรับกับความต้องการกำลังไฟฟ้าได้อย่างเพียงพอและเป็นไปตามเงื่อนไขที่ได้กำหนดไว้

ข้อเสนอแนะ

ระบบไมโครกริดที่ทำการจำลองนั้นเป็นระบบจำลองอย่างง่ายเพื่อต้องการให้เกิดความเข้าใจในการใช้งานในส่วนโปรแกรมและการประสานงานตามรูปแบบการทำงานของไมโครกริดแต่ระบบ ซึ่งในทางปฏิบัติแล้วอาจมีความหลากหลายและซับซ้อนมากกว่า เช่น ระบบตัดสินใจการทำงานในโหมดแบบแยกอิสระและแบบเชื่อมต่อกับกริดหลัก มีการนำอุปกรณ์จำพวกอุปกรณ์สำรองพลังงานเข้ามาในระบบ และระบบการปลดโหลดที่ไม่สำคัญออก กรณีเกิดเหตุความผิดพลาดต่างๆ ซึ่งผู้เขียนหวังว่าจะมีผู้ที่สนใจนำไปพัฒนาให้ดีขึ้นต่อไป

บรรณานุกรม

[1] S. Chowdhury, S.P.Chowdhury and P.Crossley. **Microgrids and Active Distribution Networks**. Institution of Engineering and Technology, London, United Kingdom, 2009

[2] Fabio Bellifemine, Giovanni Claire, Domimic Greenwood. **Developing Multi-Agent System with JADE.** John Wiley & Sons, 2007

[3] Hadi Saadat. **Power System Analysis 2nd Edition.** McGraw-Hill, 2004

[4] Hassan Feroze. **Multi-Agent Systems in Microgrids : Design and Implementation.** Thesis submitted to the Faculty of the Virgenia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering, 2009

[5] จามิกร หิรัญรัตน์. **โครงสร้างการทำงานโดยอาศัยเทคโนโลยีเอเจนท์แบบเคลื่อนที่สำหรับตลาดอิเล็กทรอนิกส์**. วิทยานิพนธ์วิทยาศาสตร์มหาบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์ ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยสงขลานครินทร์, 2547

[6] จตุรพิธ เกราะแก้ว. **ระบบควบคุมไฟจราจรชาญฉลาดด้วยมัลติเอเจนท์**. วิทยานิพนธ์วิศวกรรมศาสตร์มหาบัณฑิต, สาขาวิชาวิศวกรรมคอมพิวเตอร์ บัณฑิตวิทยาลัย สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, 2550

[7] A. L. Dimeas N. D. Hatziargyriou. **Agent based Control for Microgrids**. National Technical University of Athens, June 2007

[8] Shuichiro Yamazaki, Motohisa Ishikawa, Junichi Arai, Takaya Shono. **Application of Mobile Agent Technology for Control of Power Sources in Isolated Microgrid**. Kogakuin University and Toshiba corporation

[9] จามิกรณ์ หิรัญรัตน์ และ อำนาจ เปาะทอง. **การประยุกต์เอเจนท์แบบเคลื่อนที่สำหรับการประมูลอิเล็กทรอนิกส์**. ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยสงขลานครินทร์

ภาคผนวก

ภาคผนวก ก
ภาษาจาวาเบื้องต้น

ภาษาจาวา (อังกฤษ: Java programming language) เป็นภาษาโปรแกรมเชิงวัตถุ (อังกฤษ: Object Oriented Programming) พัฒนาโดย เจมส์ กอสลิง และวิศวกรคนอื่นๆ ที่ ซัน ไมโครซิสเต็มส์ ภาษาจาวาถูกพัฒนาขึ้นในปี พ.ศ. 2534 (ค.ศ. 1991) โดยเป็นส่วนหนึ่งของ โครงการกรีน (the Green Project) และสำเร็จออกสู่สาธารณะในปี พ.ศ. 2538 (ค.ศ. 1995) ซึ่งภาษานี้มีจุดประสงค์เพื่อใช้แทนภาษาซีพลัสพลัส (C++) โดยรูปแบบที่เพิ่มเติมขึ้นคล้ายกับภาษาอ็อบเจกต์ทีฟซี (Objective-C) แต่เดิมภาษานี้เรียกว่า ภาษาโอ๊ก (Oak) ซึ่งตั้งชื่อตามต้นโอ๊กใกล้ที่ทำงานของ เจมส์ กอสลิง แต่ว่ามีปัญหาทางลิขสิทธิ์ จึงเปลี่ยนไปใช้ชื่อ "จาวา" ซึ่งเป็นชื่อกาแฟแทน และแม้ว่าจะมีชื่อคล้ายกัน แต่ภาษาจาวาไม่มีความเกี่ยวข้องใด ๆ กับภาษาจาวาสคริปต์ (JavaScript) ปัจจุบันมาตรฐานของภาษาจาวาดูแลโดย Java Community Process ซึ่งเป็นกระบวนการอย่างเป็นทางการ ที่อนุญาตให้ผู้ที่สนใจเข้าร่วมกำหนดความสามารถในจาวาแพลตฟอร์มได้

จุดมุ่งหมายหลัก 4 ประการ ในการพัฒนาจาวา คือ

1. ใช้ภาษาโปรแกรมเชิงวัตถุ
2. ไม่ขึ้นกับแพลตฟอร์ม (สถาปัตยกรรม และ ระบบปฏิบัติการ)
3. เหมาะกับการใช้ในระบบเครือข่าย พร้อมมีไลบรารีสนับสนุน
4. เรียกใช้งานจากระยะไกลได้อย่างปลอดภัย

ก.1 ชนิดของตัวแปรและการตั้งชื่อตัวแปร

ชนิดของตัวแปรในภาษาจาวาแบ่งออกได้ดังตารางที่ ก.1

ตารางที่ ก.1 ชนิดของตัวแปรในภาษาจาวา

Keyword	Description	Size
`byte`	Byte-size integer	8-bit
`short`	Short integer	16-bit
`int`	Integer	32-bit
`long`	Long integer	64-bit
`char`	Single character	16-bit
`float`	Single-precision floating point	32-bit
`double`	Double-precision floating point	64-bit
`boolean`	True of false	1-bit

การตั้งชื่อตัวแปรนั้นจะใช้ชื่ออะไรก็ได้ ตัวอักษรที่ประกอบเป็นชื่อจะเป็นตัวอักษรภาษาอังกฤษตัวพิมพ์เล็ก ตัวพิมพ์ใหญ่ ตัวเลข เครื่องหมายสกุลเงินต่างๆ อักษรโรมัน รวมทั้งเครื่องหมาย _ มีข้อแม้คือชื่อของตัวแปรห้ามขึ้นต้นด้วยตัวเลข และตัวพิมพ์ใหญ่กับตัวพิมพ์เล็กถือว่าเป็นคนละตัวอักษร นอกจากนี้ยังห้ามตั้งชื่อด้วยคำสงวน ซึ่งมีดังต่อไปนี้

abstract	do	import	public	throws
boolean	double	instanceof	return	transient
break	else	int	short	try
byte	extends	interface	static	void
case	final	long	strictfp	volatile
catch	finally	native	super	while
char	float	new	switch	null
class	for	package	synchronized	goto
continue	if	private	this	const

นอกจากนี้ยังมีตัวแปรประเภทสตริง (String) ที่เก็บข้อมูลประเภทข้อความ ในการเขียนโปรแกรมบางครั้งจำเป็นที่จะต้องเปลี่ยนชนิดของตัวแปรเช่นเมธอดต่อไปนี้

```
//How to Convert from String to Number

Integer.parseInt(i);              // Convert String to Integer
Float.parseFloat(i);              // Convert String to Float
Double.parseDouble(i);            // Convert String to Double


//How to Convert from Number to String
String.valueOf(i);                // Convert number to String
```

ก.2 คำสั่งแสดงข้อความ

คำสั่งในการแสดงข้อความมีดังนี้

```
System.out.print("Message");
System.out.println("Message");
```

ทั้งสองคำสั่งจะมีความแตกต่างกันคือ `System.out.println` จะแสดงข้อความโดยจะจองพื้นที่ไว้ 1 บรรทัด

ในบางครั้งบางอักขระไม่สามารถเขียนได้โดยตรงเช่น \ ‘ หรือ “ เป็นต้น ดังนั้นจึงมีวิธีเขียนโดยใส่เป็นรหัสคำสั่งดังตารางที่ ก.2

ตารางที่ ก.2 รหัสอักขระพิเศษต่างๆ

Escape code	Interpretation
\'	Single quotation mark
\"	Double quotation mark
\b	Backspace
\f	From feed (page break)
\n	Newline character
\r	Carriage return
\t	Tab
\\	Backslash

ในการคอมเมนท์โปรแกรม สามารถทำได้ 2 วิธีคือ

1. การใช้เครื่องหมาย /* และ */ คั่นระหว่างข้อความของคุณ เมื่อคอมไพล์เลอร์เจอเครื่องหมายนี้ มันจะข้ามข้อความที่อยู่ภายในเครื่องหมายนี้ทั้งหมดไปเลย ดังนั้นจึงสามารถเขียนอะไรลงไปก็ได้โดยไม่ทำให้โปรแกรมผิดเพี้ยน

2. ใช้เครื่องหมาย // ในการบอกคอมไพล์เลอร์ให้ข้ามสิ่งที่อยู่ต่อท้ายเครื่องหมายไปเลยจนจบบรรทัด กรณีนี้ไม่ต้องปิดข้อความด้วยเครื่องหมายใดๆเพราะคอมไพล์เลอร์จะดูจากการขึ้นบรรทัดใหม่เป็นการบอกว่าข้อความส่วนตัวสิ้นสุดแล้ว

ก.3 การดำเนินการทางคณิตศาสตร์

เครื่องหมายคณิตศาสตร์สามารถใช้ได้กับ ค่าคงตัวจำนวนเต็ม ค่าคงตัวทศนิยม ตัวแปรจำนวนเต็ม และตัวแปรทศนิยม มีดังนี้

Operator	Description	Example	Result
+	Addition	5+2	7
-	Subtraction	5-2	3
*	Multiplication	5*2	10
/	Division	5/2	2
%	Modulus (division remainder)	5%2	1

ตารางที่ ก.3 ตัวดำเนินการทางคณิตศาสตร์

นอกจากนี้ยังมีคลาสที่มีเมธอดทางคณิตศาสตร์อื่นๆ อีก เช่น ตรีโกณมิติ เอ็กซ์โพเนนเชียน ลอการิทึม แสดงดังตารางที่ ก.4

ตารางที่ ก.4 Math class Methods

Math Method	Description
`Math.abs(int n)`	Absolute value (*n* or 0-*n*, whichever is greater)
`Math.acos(double d)`	Arc cosine of *d*
`Math.asin(double d)`	Arc sine of *d*
`Math.atan(double d)`	Arc tangent of *d*
`Math.ceil(double d)`	Ceiling (smallest value not less than *d* that is an integer)
`Math.cos(double d)`	Cosine of *d*
`Math.exp(double d)`	(ed, where e=2.718...)
`Math.floor(double d)`	Floor (highest value not greater than *d* that is an integer)
`Math.log(double d)`	Natural logarithm of *d*
`Math.pow(double a, double b)`	a^b
`Math.random()`	Generates a random number between `0.0` and `1.0`
`Math.round(float f)`	Rounds *f* to the nearest `int` value
`Math.round(double d)`	Rounds *d* to the nearest `long` value
`Math.sin(double d)`	Sine of *d*
`Math.sqrt(double d)`	Square root of *d*
`Math.tan(double d)`	Tangent of *d*
`Math.toDegrees(double d)`	Converts *d* (in radians) to degrees
`Math.toRadians(double d)`	Converts *d* (in degrees) to radians

ก.4 การดำเนินการทางตรรกะ

ตารางที่ ก.5 ตัวดำเนินการทางตรรกะ

รูปแบบคำสั่ง	ความหมาย
`!x`	ค่าความจริงที่ตรงข้ามกับ x
`x & y`	ค่าความจริงของ x AND y
`x | y`	ค่าความจริงของ x OR y
`x ^ y`	ค่าความจริงของ x XOR y
`x && y`	ค่าความจริงของ x AND y
`x || y`	ค่าความจริงของ x OR y

ตารางที่ ก.6 ตารางค่าความจริง

x	y	!x	x & y	x | y	x ^ y	x && y	x || y
True	True	False	True	True	False	True	True
True	False	False	False	True	True	False	True
False	True	True	False	True	True	False	True
False	False	True	False	False	False	False	False

ก.5 โครงสร้างแบบเลือกทำ

- คำสั่ง if – else

```
if (logical expression) {
      //true statements
}

else {
      //false statements
}
```

- คำสั่ง switch

```
switch (expression) {
      case value 1 : //statements 1
      break;

      case value 2 : //statements 2
      break;
            : :
      case value N : //statements N
      break;

      default : statements N+1
      break;
}
```

ก.6 โครงสร้างแบบทำซ้ำ

- คำสั่ง while

```
initial statements
      while (logical expression){
            //statements
            //update statements
      }
```

- คำสั่ง do while

```
initial statements
      do {
            //statements
            //update statements
      }

      while(logical expression);
```

- คำสั่ง for

```
for(initial statements; expression; update statements) {
      //statements
}
```

ก.7 ตัวแปรอาเรย์

อาเรย์ คือ เซตของตัวแปรชนิดเดียวกัน ซึ่งสมาชิกของอาเรย์อาจเป็นตัวแปรพื้นฐาน หรือตัวแปรอ้างอิงก็ได้ จำนวนสมาชิกของอะเรย์มีขนาดแน่นอน และสมาชิกของอาเรย์แต่ละตัวจะมีลำดับประจำตัวอยู่ ตัวอย่างประกาศตัวแปรอาเรย์ 1 มิติ เป็นดังนี้

```
int[] n = new n[10];
```

ดังนั้น ตัวแปรที่เป็นสมาชิกของอาเรย์ n ได้แก่ `n[0],n[1],n[2], … ,n[10]`

ก.8 การอ่านและเขียนเท็กซ์ไฟล์

ในการอ่านและเขียนเท็กซ์ไฟล์นั้นจำเป็นที่จะต้องขยายคลาสของ io ดังคำสั่งต่อไปนี้ import java.io.*; และจะต้องมรการดักจับข้อผิดพลาดโดยใช้ try-catch ด้วย

- **การเขียนเท็กซ์ไฟล์** จากตัวอย่างต่อไปนี้จะเป็นคำสั่งการเขียนไฟล์โดยชื่อไฟล์คือ Data test.txt และไฟล์นั้นพิมพ์ข้อความออกมาว่า 50011365

```
try {
    FileWriter fw = new FileWriter("Data test.txt");
    PrintWriter opf = new PrintWriter(fw);
    opf.println("50011365");
    opf.close();
    }

catch(IOException e){
   System.out.println("Error");
   }
```

- **การอ่านเท็กซ์ไฟล์** จากตัวอย่างต่อไปนี้จะเป็นคำสั่งการอ่านไฟล์โดยชื่อไฟล์คือ Data test.txt โดยไฟล์ที่อ่านนั้นจะเป็นไฟล์ประเภทสตริง

```
try {
   fw = new FileReader("Data test.txt");
   inp = new BufferedReader(fw);
   String m = inp.readLine();
   inp.close();
   }

catch(IOException e){
   System.out.println("Error");
   }
```

คลาสและเมธอดของ JADE

ข.1 AID

Field Summary	
static java.lang.String	**AGENT_CLASSNAME** Key to retrieve the agent class name as a user defined slot of the AID included in the AMSAgentDescription registered with the AMS.
static boolean	**ISGUID** constant to be used in the constructor of the AID
static boolean	**ISLOCALNAME** constant to be used in the constructor of the AID

Constructor Summary
AID() Constructs an Agent-Identifier whose slot name is set to an empty string
AID(java.lang.String guid) **Deprecated.** *This constructor might generate a wrong AID, if the passed parameter is not a guid (globally unique identifier), but the local name of an agent (e.g. "da0").*
AID(java.lang.String name, boolean isGUID) Constructor for an Agent-identifier

Method Summary	
void	**addAddresses**(java.lang.String url) This method permits to add a transport address where the agent can be contacted.
void	**addResolvers**(AID aid)

	This method permits to add the AID of a resolver (an agent where name resolution services for the agent can be contacted)
void	**addUserDefinedSlot**(java.lang.String key, java.lang.String value) To add a user defined slot (a pair key, value).
void	**clearAllAddresses**() To remove all addresses of the agent
void	**clearAllResolvers**() To remove all resolvers.
java.lang.Object	**clone**() Clone the AID object.
int	**compareTo**(java.lang.Object o) Comparison operation.
boolean	**equals**(java.lang.Object o) Equality operation.
java.lang.String[]	**getAddressesArray**() Returns an array of string containing all the addresses of the agent
Iterator	**getAllAddresses**() Returns an iterator of all the addresses of the agent.
Iterator	**getAllResolvers**() Returns an iterator of all the resolvers.
Properties	**getAllUserDefinedSlot**() Returns the user-defined slots as properties.
java.lang.String	**getHap**() Returns the HAP of the agent or null if the GUID of this AID is not of the form @
java.lang.String	**getLocalName**() Returns the local name of the agent (without the HAP).
java.lang.String	**getName**()

	This method returns the name of the agent.
AID[]	**getResolversArray**() Returns an array containing all the AIDs of the resolvers.
int	**hashCode**() Hash code.
boolean	**removeAddresses**(java.lang.String url) To remove a transport address.
boolean	**removeResolvers**(AID aid) To remove a resolver.
boolean	**removeUserDefinedSlot**(java.lang.String key) To remove a user defined slot.
void	**setLocalName**(java.lang.String n) This method permits to set the symbolic name of an agent.
void	**setName**(java.lang.String n) This method permits to set the symbolic name of an agent.
java.lang.String	**toString**() Converts this agent identifier into a readable string.

ข.2 ACL Message

Field Summary	
static int	**ACCEPT_PROPOSAL** constant identifying the FIPA performative
static int	**AGREE** constant identifying the FIPA performative
static java.lang.String	**AMS_FAILURE_AGENT_NOT_FOUND** AMS failure reasons
static java.lang.String	**AMS_FAILURE_AGENT_UNREACHABLE**
static java.lang.String	**AMS_FAILURE_FOREIGN_AGENT_NO_ADDRESS**
static java.lang.String	**AMS_FAILURE_FOREIGN_AGENT_UNREACHABLE**
static java.lang.String	**AMS_FAILURE_SERVICE_ERROR**
static java.lang.String	**AMS_FAILURE_UNAUTHORIZED**
static java.lang.String	**AMS_FAILURE_UNEXPECTED_ERROR**
static int	**CANCEL** constant identifying the FIPA performative
static int	**CFP** constant identifying the FIPA performative
static int	**CONFIRM** constant identifying the FIPA performative

static int	**DISCONFIRM** constant identifying the FIPA performative
static int	**FAILURE** constant identifying the FIPA performative
static java.lang.String	**IGNORE_FAILURE** User defined parameter key specifying, when set to "true", that if the delivery of a message fails, no FAILURE notification has to be sent back to the sender.
static int	**INFORM** constant identifying the FIPA performative
static int	**INFORM_IF** constant identifying the FIPA performative
static int	**INFORM_REF** constant identifying the FIPA performative
static int	**NOT_UNDERSTOOD** constant identifying the FIPA performative
static java.lang.String	**POST_TIME_STAMP** User defined parameter key specifying that the corresponding value must be replaced by the JADE runtime by an ISO8601 encoded time-stamp at posting time.
static int	**PROPAGATE** constant identifying the FIPA performative
static int	**PROPOSE** constant identifying the FIPA performative
static int	**PROXY** constant identifying the FIPA performative
static int	**QUERY_IF** constant identifying the FIPA performative
static int	**QUERY_REF** constant identifying the FIPA performative

static int	**REFUSE** constant identifying the FIPA performative
static int	**REJECT_PROPOSAL** constant identifying the FIPA performative
static int	**REQUEST** constant identifying the FIPA performative
static int	**REQUEST_WHEN** constant identifying the FIPA performative
static int	**REQUEST_WHENEVER** constant identifying the FIPA performative
static int	**SUBSCRIBE** constant identifying the FIPA performative
static java.lang.String	**TRACE** User defined parameter key specifying that the JADE tracing mechanism should be activated for this message.
static int	**UNKNOWN** constant identifying an unknown performative

Constructor Summary
ACLMessage() **Deprecated.** *Since every ACL Message must have a message type, you should use the new constructor which gets a message type as a parameter. To avoid problems, now this constructor silently sets the message type to not-understood.*
ACLMessage(int perf) This constructor creates an ACL message object with the specified performative.

Method Summary	
void	**addReceiver**(AID r) Adds a value to :receiver slot.

void	**addReplyTo**(AID dest) Adds a value to :reply-to slot.
void	**addUserDefinedParameter**(java.lang.String key, java.lang.String value) Add a new user defined parameter to this ACLMessage.
void	**clearAllReceiver**() Removes all values from :receiver slot.
void	**clearAllReplyTo**() Removes all values from :reply_to slot.
java.lang.Object	**clone**() Clone an ACLMessage object.
ACLMessage	**createReply**() create a new ACLMessage that is a reply to this message.
Iterator	**getAllIntendedReceiver**() retrieve the whole list of intended receivers for this message.
static java.lang.String[]	**getAllPerformativeNames**() Returns the list of the communicative acts as an array of String.
Iterator	**getAllReceiver**() Reads :receiver slot.
Iterator	**getAllReplyTo**() Reads :reply_to slot.
Properties	**getAllUserDefinedParameters**() get a clone of the data structure with all the user defined parameters
byte[]	**getByteSequenceContent**() Reads :content slot.

java.lang.String	**getContent**() Reads :content slot.
java.io.Serializable	**getContentObject**() This method returns the content of this ACLMessage when they have been written via the method setContentObject.
java.lang.String	**getConversationId**() Reads :conversation-id slot.
java.lang.String	**getEncoding**() Reads :encoding slot.
Envelope	**getEnvelope**() Reads the envelope attached to this message, if any.
java.lang.String	**getInReplyTo**() Reads :reply-to slot.
static int	**getInteger**(java.lang.String perf) Returns the integer corresponding to the performative
java.lang.String	**getLanguage**() Reads :language slot.
java.lang.String	**getOntology**() Reads :ontology slot.
int	**getPerformative**() return the integer representing the performative of this object
static java.lang.String	**getPerformative**(int perf) Returns the string corresponding to the integer for the performative
java.lang.String	**getProtocol**() Reads :protocol slot.
java.lang.String	**getReplyBy**() **Deprecated.** *Since the value of this slot is a Date by*

	definition, then the getReplyByDate should be used that returns a Date
java.util.Date	**getReplyByDate**() Reads :reply-by slot.
java.lang.String	**getReplyWith**() Reads :reply-with slot.
AID	**getSender**() Reads :sender slot.
java.lang.String	**getUserDefinedParameter**(java.lang.String key) Searches for the user defined parameter with the specified key.
boolean	**hasByteSequenceContent**() This method allows to check if the content of this ACLMessage is a byteSequence or a String
boolean	**removeReceiver**(AID r) Removes a value from :receiver slot.
boolean	**removeReplyTo**(AID dest) Removes a value from :reply_to slot.
boolean	**removeUserDefinedParameter**(java.lang.String key) Removes the key and its corresponding value from the list of user defined parameters in this ACLMessage.
void	**reset**() Resets all the message slots.
void	**setByteSequenceContent**(byte[] content) Writes the :content slot.
void	**setContent**(java.lang.String content) Writes the :content slot.
void	**setContentObject**(java.io.Serializable s) This method sets the content of this ACLMessage to a Java object.

void	**setConversationId**(java.lang.String str) Writes the :conversation-id slot.
void	**setDefaultEnvelope**() Writes the message envelope for this message, using the :sender and :receiver message slots to fill in the envelope.
void	**setEncoding**(java.lang.String str) Writes the :encoding slot.
void	**setEnvelope**(Envelope e) Attaches an envelope to this message.
void	**setInReplyTo**(java.lang.String reply) Writes the :in-reply-to slot.
void	**setLanguage**(java.lang.String str) Writes the :language slot.
void	**setOntology**(java.lang.String str) Writes the :ontology slot.
void	**setPerformative**(int perf) set the performative of this ACL message object to the passed constant.
void	**setProtocol**(java.lang.String str) Writes the :protocol slot.
void	**setReplyByDate**(java.util.Date date) Writes the :reply-by slot.
void	**setReplyWith**(java.lang.String reply) Writes the :reply-with slot.
void	**setSender**(AID s) Writes the :sender slot.
java.lang.String	**toString**() Convert an ACL message to its string representation.

ข.3 Message Template

Method Summary	
static MessageTemplate	**and**(MessageTemplate op1, MessageTemplate op2) Logical **and** between two MessageTemplate objects.
boolean	**match**(ACLMessage msg) Matches an ACL message against this MessageTemplate object.
static MessageTemplate	**MatchAll**() This *Factory Method* returns a message template that matches any message.
static MessageTemplate	**MatchContent**(java.lang.String value) This *Factory Method* returns a message template that matches any message with a given :content slot.
static MessageTemplate	**MatchConversationId**(java.lang.String value) This *Factory Method* returns a message template that matches any message with a given :conversation-id slot.
static MessageTemplate	**MatchCustom**(ACLMessage msg, boolean matchPerformative) This *Factory Method* returns a message template that matches ACL messages against a given one, passed as parameter.
static MessageTemplate	**MatchEncoding**(java.lang.String value) This *Factory Method* returns a message template that matches any message with a given :encoding slot.
static MessageTemplate	**MatchInReplyTo**(java.lang.String value) This *Factory Method* returns a message template that matches any message with a given :in-reply-to slot.
static MessageTemplate	**MatchLanguage**(java.lang.String value) This *Factory Method* returns a message template that matches any message with a given :language slot.

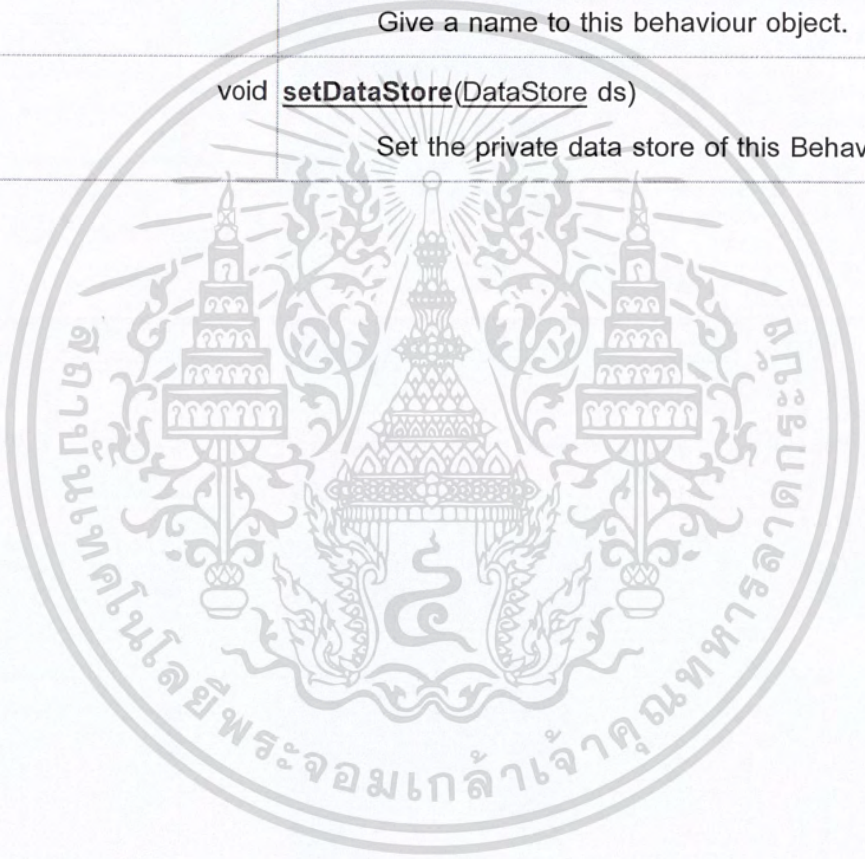
static MessageTemplate	**MatchOntology**(java.lang.String value) This *Factory Method* returns a message template that matches any message with a given :ontology slot.
static MessageTemplate	**MatchPerformative**(int value) This *Factory Method* returns a message template that matches any message with a given performative.
static MessageTemplate	**MatchProtocol**(java.lang.String value) This *Factory Method* returns a message template that matches any message with a given :protocol slot.
static MessageTemplate	**MatchReceiver**(AID[] values) This *Factory Method* returns a message template that matches any message with a given :receiver slot.
static MessageTemplate	**MatchReplyByDate**(java.util.Date value) This *Factory Method* returns a message template that matches any message with a given :reply-by slot.
static MessageTemplate	**MatchReplyTo**(AID[] values) This *Factory Method* returns a message template that matches any message with a given :reply-to slot.
static MessageTemplate	**MatchReplyWith**(java.lang.String value) This *Factory Method* returns a message template that matches any message with a given :reply-with slot.
static MessageTemplate	**MatchSender**(AID value) This *Factory Method* returns a message template that matches any message with a given :sender slot.
static MessageTemplate	**MatchTopic**(AID topic) This *Factory Method* returns a message template that matches any message about a given topic.
static MessageTemplate	**not**(MessageTemplate op) Logical **not** of a MessageTemplate object.

static MessageTemplate	**or**(MessageTemplate op1, MessageTemplate op2) Logical **or** between two MessageTemplate objects.
java.lang.String	**toString**() Retrieve a string representation of this message template.

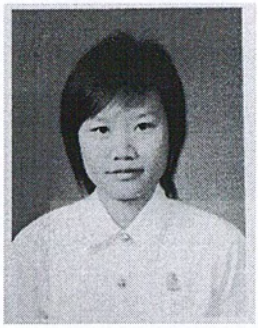
ข.4 Behaviours

Method Summary	
abstract void	**action**() Runs the behaviour.
void	**block**() Blocks this behaviour.
void	**block**(long millis) Blocks this behaviour for a specified amount of time.
abstract boolean	**done**() Check if this behaviour is done.
java.lang.String	**getBehaviourName**() Retrieve the name of this behaviour object.
DataStore	**getDataStore**() Return the private data store of this Behaviour.
protected CompositeBehaviour	**getParent**() Retrieve the enclosing CompositeBehaviour (if present).
boolean	**isRunnable**() Returns whether this Behaviour object is blocked or not.
int	**onEnd**() This method is just an empty placeholder for subclasses.
void	**onStart**() This method is just an empty placeholders for subclasses.
void	**reset**() Restores behaviour initial state.

void	**restart**() Restarts a blocked behaviour.
Behaviour	**root**() Returns the root for this Behaviour object.
void	**setAgent**(Agent a) Associates this behaviour with the agent it belongs to.
void	**setBehaviourName**(java.lang.String name) Give a name to this behaviour object.
void	**setDataStore**(DataStore ds) Set the private data store of this Behaviour



ประวัติผู้เขียน



นางสาว ยลดา มาศวิจิตรวงศ์
เกิดวันที่ 16 เมษายน 2532
ภูมิลำเนา เพชรบุรี
การศึกษา
- มัธยมศึกษา โรงเรียนพรหมานุสรณ์ จังหวัดเพชรบุรี
- ปริญญาตรี สาขาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

นาย วรพจน์ โรจน์รัตนาวิชัย
เกิดวันที่ 20 มีนาคม 2532
ภูมิลำเนา กรุงเทพมหานคร
การศึกษา
- มัธยมศึกษา โรงเรียนนวมินทราชินูทิศ เตรียมอุดมศึกษาน้อมเกล้า
- ปริญญาตรี สาขาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

นาย วรพจน์ ฮุนสวัสดิกุล
เกิดวันที่ 12 พฤศจิกายน 2531
ภูมิลำเนา พระนครศรีอยุธยา
การศึกษา
- มัธยมศึกษา โรงเรียนอยุธยาวิทยาลัย
- ปริญญาตรี สาขาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง