ระบบควบคุมไมโครกริดโดยใช้เอเจนท์

AGENT-BASED MICROGRID CONTROL SYSTEM

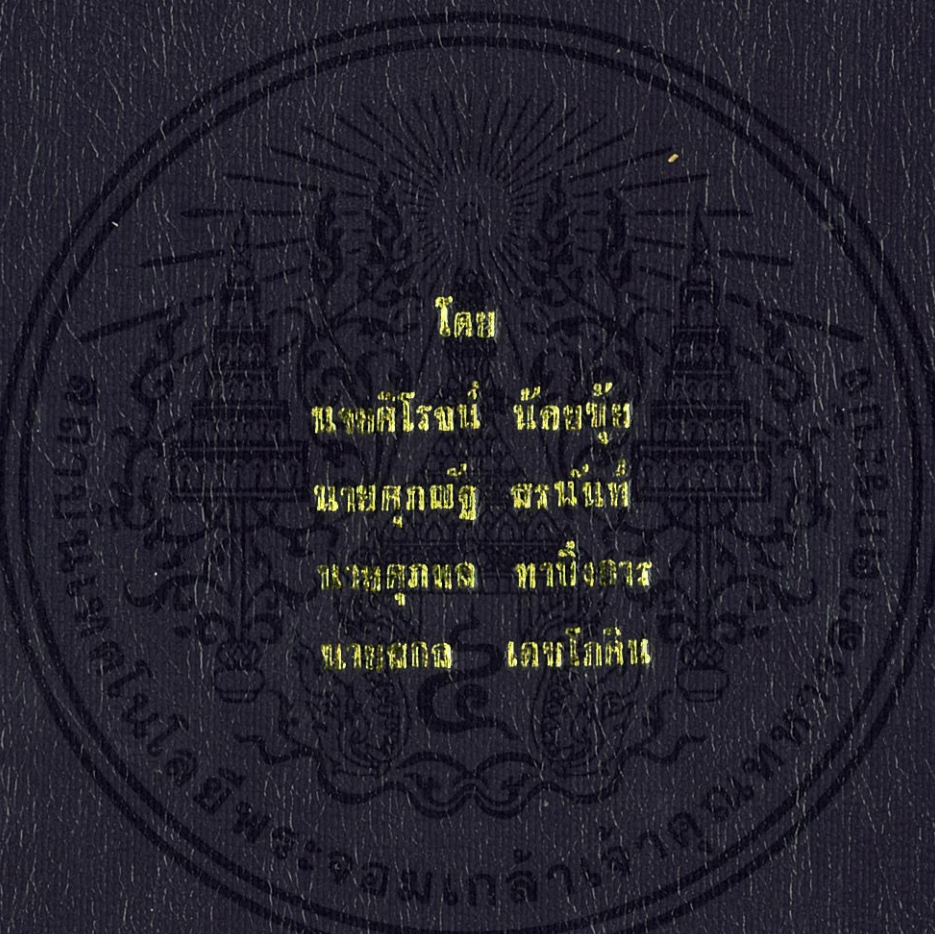
โดย

นายศิโรจน์ นัดขุ้ย

นายสุกฤษฎ์ สรนันท์

นายสุกนต ทาบึงกาฬ

นายสกล เดชโสพิน



ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2557

ระบบควบคุมไมโครกริดโดยใช้เอเจนท์
AGENT-BASED MICROGRID CONTROL SYSTEM

โดย

นายศิโรจน์ น้อยขุ้ย
นายศุภณัฐ สรนันท์
นายศุภพล ทาบึงการ
นายสกล เดชโภคิน

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2557

AGENT-BASED MICROGRID CONTROL SYSTEM

SIROJ NOIKHUI

SUPANUT SAWRANUN

SUPAPOL THABUNGKARN

SAKOL DECHPHOKIN

THIS PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT

FOR THE BACHELOR DEGREE IN ELECTRICAL ENGINEERING

DEPARTMENT OF ELECTRICAL ENGINEERING FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2014

ปีการศึกษา 2557

ระบบควบคุมไมโครกริดโดยใช้เอเจนท์
AGENT-BASED MICROGRID CONTROL SYSTEM

โดย

นายศิโรจน์ น้อยขุ้ย
นายศุภณัฐ สรนันท์
นายศุภพล ทาบึงการ
นายสกล เดชโภคิน

อาจารย์ที่ปรึกษา

ดร.สมภพ ผลไม้

ปริญญานิพนธ์ปีการศึกษา 2557

ภาควิชาวิศวกรรมไฟฟ้า

คณะวิศวกรรมศาสาตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบควบคุมไมโครกริดโดยใช้เอเจนท์

ผู้จัดทำ

1. นาย ศิโรจน์ น้อยขุ้ย
2. นาย ศุภณัฐ สรนันท์
3. นาย ศุภพล ทาบึงการ
4. นาย สกล เดชโภคิน

..... อาจารย์ที่ปรึกษา

(ดร.สมภพ ผลไม้)

ระบบควบคุมไมโครกริดโดยใช้เอเจนท์

นายศิโรจน์ น้อยขุ้ย
นายศุภณัฐ สรนันท์
นายศุภพล ทาบึงการ
นายสกล เดชโภคิน
ดร.สมภพ ผลไม้ อาจารย์ที่ปรึกษา

บทคัดย่อ

โครงงานนี้นำเสนอการพัฒนาระบบการจัดการพลังงานสำหรับไมโครกริดโดยใช้เทคโนโลยีมัลติเอเจนท์ โดยมีวัตถุประสงค์เพื่อรักษาสมดุลระหว่างความต้องการการใช้ไฟฟ้ากับปริมาณการผลิต เพื่อเพิ่มคุณภาพและความน่าเชื่อถือของระบบ โดยไมโครกริดที่จะศึกษาประกอบด้วย เส้นภาระไฟฟ้า และเครื่องกำเนิดไฟฟ้า 3 หน่วยผลิต โดยมีการเขียนโปรแกรมสำหรับการควบคุมเอเจนต์ด้วยโปรแกรม C# เพื่อเชื่อมโยงระหว่างเอเจนท์เครื่องกำเนิดไฟฟ้าทั้ง 3 หน่วยผลิตที่ควบคุมโดย DSP ร่วมกับ Arduino ซึ่งทำหน้าที่ในการควบคุมการเปิด-ปิดภาระทางไฟฟ้าผ่านรีเลย์บอร์ด และมีการแสดงผลการกระทำของภาระทางไฟฟ้าออกทางหน้าจอโปรแกรมที่เขียนขึ้นด้วยโปรแกรม C# โดยการเชื่อมต่อทั้งหมดของระบบจะส่งข้อมูลผ่านทาง CANUSB ซึ่งเป็นอุปกรณ์ที่ใช้ในการรับ-ส่งข้อมูล จากผลการทดลองดังกล่าวได้ผลการทดลองว่า เอเจนท์สามารถจัดสรรพลังงานได้อย่างเหมาะสมและควบคุมให้ระบบดำเนินไปอย่างมีเสถียรภาพอีกด้วย

AGENT-BASED MICROGRID CONTROL SYSTEM

Mr.Siroj Noikhui
Mr.Supanat Sawranan
Mr.Supapol Thabungkarn
Mr.Sakol Detphokin
Dr.Sompob Polmai Advisor

Abstract

This project develops energy management system for a microgrid using agent technology. The objective of the energy management system is to balance the load demand with the generation supply in order to increase quality and reliability of the system. The microgrid in this study consists of load curve and three generators. Using C# for connect the agent of three generators which are controlled by DSP and arduino (controlling turn on and turn off load in relay board) and show the result in the program which appear on The monitor programming by C# and everything in the microgrid system are transmitted and received data by CANUSB. The experimental show the result that agent can manage the energy properly and control system stability properly.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้เสร็จสมบูรณ์ไปได้ด้วยดี ในนามของผู้เขียนต้องขอกราบขอบพระคุณบุคคลดังต่อไปนี้

- ขอขอบพระคุณ ดร.สมภพ ผลไม้ ผู้ซึ่งเป็นอาจารย์ที่ปรึกษาที่กรุณาให้คำปรึกษา รวมทั้งแนะนำโครงงานและให้ความช่วยเหลือในทุกๆ ด้าน ตลอดจนอาจารย์ในสาขาวิชาที่ประสิทธิประสาทความรู้ให้กับผู้เขียนในครั้งนี้
- ขอขอบคุณคุณเพื่อนๆพี่ๆร่วมห้องปฏิบัติการ PEARL โดยเฉพาะพี่ศรายุทธ จิตประสงค์ และพี่โทนี่ ที่คอยช่วยเหลือและให้คำแนะนำมาโดยตลอดในการทำโครงงาน
- ขอขอบคุณ คุณนครศักดิ์ เจ้าหน้าที่ห้องปฏิบัติการที่ให้เบิกใช้เครื่องมือ ตลอดจนคุณปุณยวีร์ ฉายศิริ ที่คอยให้คำแนะนำต่างๆในการทำโครงงาน
- ขอขอบพระคุณสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่คอยส่งเสริมสนับสนุนการศึกษาและการวิจัยของนักศึกษา
- ขอขอบคุณสมาชิกห้องวิจัย PEARL LAB ที่คอยช่วยเหลือด้านต่างๆ รวมทั้งเป็นขวัญและกำลังใจให้กับผู้เขียน
- สุดท้ายนี้ต้องขอขอบพระคุณบุคคลที่สำคัญที่สุดที่ทำให้ข้าพเจ้ามีวันนี้ ก็คือบิดา มารดา อันเป็นที่รักยิ่ง ซึ่งได้อบรมเลี้ยงดูข้าพเจ้าเป็นอย่างดี พร้อมทั้งให้โอกาสในการศึกษาอย่างเต็มที่ คอยให้กำลังใจ และคอยเอาใจใส่ในทุกๆ ด้านเสมอ

ข้าพเจ้าขอระลึกคุณอย่างสุดซึ้งและขอกราบขอบพระคุณมา ณ ที่นี้

สารบัญ

สารบัญ(ต่อ)

สารบัญรูป

สารบัญรูป(ต่อ)

สารบัญตาราง

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบันมีความต้องการในการใช้พลังงานเพิ่มขึ้นอย่างต่อเนื่องทั้งในด้านของภาคธุรกิจ อุตสาหกรรม หรือรวมทั้งในการนำไปใช้ในชีวิตประจำวัน ส่งผลให้ปริมาณเชื้อเพลิงในปัจจุบันมีอัตราลดลงอย่างรวดเร็วอย่างเห็นได้ชัดซึ่งอาจส่งผลกระทบต่อความต้องการเชื้อเพลิงได้ในอนาคต และในทางกลับกันกลับส่งผลให้ราคาของเชื้อเพลิงในการใช้ผลิตพลังงานดังกล่าวนั้นกลับมีอัตราที่เพิ่มขึ้น รวมถึงในปัจจุบันองค์กรณ์ต่างๆได้หันมาให้ความสำคัญกับปัญหาสิ่งแวดล้อม เห็นได้จากพันธกรณีของพิธีสารเกียวโตซึ่งมีเนื้อหาเกี่ยวกับการลดปริมาณการปลดปล่อยก๊าซเรือนกระจกเพื่อป้องกันและบรรเทา ปัญหาโลกร้อน (Global Warming) และแนวโน้มการเปลี่ยนแปลงรูปแบบการผลิตไฟฟ้าจากการเปิดเสรี ทางพลังงาน (Deregulation) ทำให้ความสนใจเกี่ยวกับการผลิตพลังงานไฟฟ้าจากแหล่งพลังงานธรรมชาติหรือ พลังงานหมุนเวียน(Renewable Energy Resources) เช่น พลังงานลม พลังงานแสงอาทิตย์ หรือ พลังงานชีวะมวล เพิ่มขึ้นอย่างเห็นได้ชัด

การผลิตไฟฟ้าในปัจจุบันสามารถแบ่งออกได้เป็นสองประเภทได้แก่ การผลิตไฟฟ้าแบบธรรมดา (Unidirectional) และการผลิตไฟฟ้าแบบมีทิศทางหรือแบบกระจายจากแหล่งพลังงาน (Directional) ในส่วนของการผลิตไฟฟ้าแบบธรรมดา (Unidirectional) จะเป็นลักษณะการผลิตไฟฟ้าปริมาณมากจากโรงไฟฟ้าขนาดใหญ่และจ่ายไฟฟ้าแบบทิศทางเดียว (Unidirectional) จากผู้ผลิตสู่ผู้บริโภคโดยตรงผ่านทางระบบส่ง ระบบจำหน่ายและสถานีย่อยต่างๆ โดยในการผลิตไฟฟ้าจะใช้เชื้อเพลิงจำพวกถ่านหิน น้ำมันดีเซลหรือก๊าซธรรมชาติเป็นหลัก และในปัจจุบันก็ได้มีการนำเทคโนโลยีในด้านของการผลิตไฟฟ้าจากพลังงานนิวเคลียร์เข้ามาใช้ในกระบวนการผลิตไฟฟ้า ซึ่งการผลิตไฟฟ้าแบบธรรมดา (Unidirectional) นี้มีข้อจำกัดอยู่ว่าปริมาณไฟฟ้าที่จะผลิตจะต้องมีความสัมพันธ์กับภาระความต้องการในการใช้พลังงานของผู้บริโภคหรือเส้นโค้งโหลด (Load Curve) ซึ่งโดยทั่วไปแล้วเส้นโค้งของโหลดจะมีแตกต่างกันในแต่ละช่วงเวลา รวมไปถึงความหลากหลายของตำแหน่งสถานที่ของผู้บริโภคนั้นๆ ทำให้การส่งจ่ายกำลังไฟฟ้าจะเป็นลักษณะของการส่งจ่ายกำลังไฟฟ้าไปในระยะทางไกลๆ ซึ่งจะส่งผลให้เกิดการสูญเสียขึ้นในสายส่ง รวมไปถึงปัญหาต่างๆ เช่น ปัญหาทางด้านแรงดันตกที่ปลายสาย ทำให้การผลิตไฟฟ้าแบบทิศทางเดียวที่ได้จากโรงจักรไฟฟ้าขนาดใหญ่จะมีประสิทธิภาพและความเชื่อถือของระบบค่อนข้างต่ำจะส่งผลให้เกิดการสูญเสียในสายส่งรวมไปถึงปัญหาต่างๆ เช่น ปัญหาทางด้านแรงดันตกที่ปลายสาย ทำให้การผลิตไฟฟ้าแบบทิศทางเดียวที่ได้จากโรงจักรไฟฟ้าขนาดใหญ่จะมีประสิทธิภาพและความเชื่อถือของระบบค่อนข้างต่ำ จึงทำให้เกิดแนวคิดการผลิตไฟฟ้าแบบมีทิศทางหรือแบบกระจายจากแหล่งพลังงาน (Directional) คือ การผลิตพลังงานไฟฟ้าแบบกระจายจากแหล่งพลังงานธรรมชาติ หรือพลังงานหมุนเวียน (Renewable Energy Resources) ซึ่งการผลิตไฟฟ้าแบบกระจายนั้นจะเป็นระบบขนาดเล็ก

โดยผู้ใช้มีอิสระที่จะเลือกตำแหน่งติดตั้งที่อยู่ใกล้กับภาระไฟฟ้า ทำให้ระยะห่างระหว่างแหล่งกำเนิดพลังงานไฟฟ้ากับภาระไฟฟ้าที่สั้นมากจึงทำให้ลดการสูญเสียพลังงานตามสายส่งเป็นการเพิ่มประสิทธิภาพรวมของระบบ และลดมลพิษที่ถูกปล่อยสู่บรรยากาศอีกด้วย

การผลิตพลังงานไฟฟ้าจากแหล่งพลังงานธรรมชาติเหล่านี้ โดยทั่วไปแล้วกำลังไฟฟ้าที่ผลิตได้จะมีปริมาณที่ไม่แน่นอนหรือมีความผันผวนเป็นอย่างมาก จึงทำให้เกิดความกังวลในด้านของคุณภาพไฟฟ้าและความเชื่อถือได้ของระบบในกรณีที่มีการนำแหล่งกำเนิดไฟฟ้าเหล่านี้จำนวนมากเข้ามาต่อกับระบบไฟฟ้า (Grid Connected Distributed Generation) จึงได้มีการนำเสนอแนวทางการผลิตและส่งจ่ายไฟฟ้าภายในพื้นที่สำหรับระบบไฟฟ้ากำลังขนาดเล็ก โดยที่ผลิตไฟฟ้าได้ในระดับแรงดันต่ำ จึงทำให้สามารถเชื่อมต่อระบบเข้ากับภาระไฟฟ้า และเชื่อมต่อกับกริดไฟฟ้าหลักได้ เรียกระบบการผลิตไฟฟ้าขนาดเล็กเหล่านี้ว่า “ไมโครกริด” (Microgrid) โดยที่จะเป็นการจ่ายไฟฟ้าให้กับกลุ่มผู้ใช้ขนาดเล็กที่อยู่ไม่ไกลกัน เช่น หมู่บ้าน สถานศึกษา ศูนย์การค้า โรงงานหรือย่านอุตสาหกรรม เป็นต้น โดยขนาดของไมโครกริดจะมีขนาดใกล้เคียงกับภาระทางไฟฟ้าของไมโครกริดนั้น ดังนั้นสิ่งสำคัญของการผลิตไฟฟ้าด้วยไมโครกริด คือการรักษาสมดุลระหว่างความต้องการการใช้ไฟฟ้ากับปริมาณการผลิตกำลังไฟฟ้าเพื่อรักษาคุณภาพ และความน่าเชื่อถือของระบบ

ดังนั้นจึงต้องมีการควบคุม Distributed Generation เหล่านี้ให้มีประสิทธิภาพในการจ่ายโหลดให้เหมาะสมกับความต้องการ และมีคุณภาพทางไฟฟ้าที่ได้มาตรฐานทางด้านพลังงาน ซึ่งนำไปสู่หลักการในการควบคุมระบบการทำงานของไมโครกริดซึ่งต้องมีการทำงานทั้งในแบบ Grid Connected คือเชื่อมต่อกับกริดภายนอก และในแบบ Standalone คือ ไม่มีการเชื่อมต่อกับกริดภายนอก โดยใน Distributed Generation ทำการต่อแบบขนานเข้าสู่ AC Bus ผ่าน Inverter ซึ่งจะนำไปต่อกับ Unity grid ต่อไปในการควบคุมmicro grid เพื่อที่จะแบ่งจ่ายโหลดได้อย่างเหมาะสม รวมไปถึงมีการติดต่อสื่อสารแบบไร้สายกันระหว่าง Distributed Generationซึ่งจะใช้การควบคุมการจ่ายพลังงาน กำลังไฟฟ้าจริงและกำลังไฟฟ้าเสมือน หรือเรียกว่า วิธี Droop Control ซึ่งเป็นวิธีที่มีความแพร่หลายและมีความนิยมในกรณีที่ใช้ในการ โดย Distributed Generationจะจ่ายโหลดให้ได้ตามความต้องการในการใช้พลังงานผ่านการควบคุมความถี่ และ แรงดัน ตามคุณลักษณะ Droop ในส่วนของการควบคุมที่สำคัญของระบบไมโครกริดคือการควบคุมไมโครกริดให้จ่ายแรงดัน,กระแส เมื่อทำการซิงโครไนซ์กับระบบอีกครั้งเพื่อที่จะได้รับ Smooth Reconnection กับ Unity Grid ซึ่งในการทำงานของระบบไม่เพียงแต่จะมีการตัดวงจรจากกริดหลักแล้วแต่ยังต้องสามารถตรวจจับกระแสความผิดพร่อง(Fault) ได้และทำการต่อใหม่กับ Gird เมื่อทำการ Clear Fault เรียบร้อยแล้ว ซึ่งการทำงานในลักษณะนี้จะช่วยป้องกันอันตรายให้กับโหลดต่างๆ ทาง Electronic ซึ่งอาจมีความ Sensitive หรือมีความอ่อนไหวต่อกระแสหรือแรงดันที่มีการกระชากโดยในโครงงานนี้ได้นำเทคนิค Angle Droop และ Voltage Droop มาประยุกต์ใช้โดยศึกษาจากการกำหนดแบบจำลองพื้นฐานระบบไมโครกริด โดยที่เทคนิค Angle Droop และ Voltage Droop ทำให้เรามีความสามารถที่จะเชื่อมต่อ Microsource กับระบบได้ทันที (Plug and Play) โดยอาศัยข้อมูล ณ ตำแหน่งนั้น นั่นก็คือ แรงดัน และความถี่ วิธีการนี้ทำให้เราสามารถต่อขนานระหว่าง Microsource ได้อย่างมีเสถียรภาพทั้งในโหมด Grid Connected และ Standalone เมื่อแหล่งจ่ายภายนอก(Grid) เกิดปัญหา

ถึงแม้ว่าการใช้งานระบบไมโครกริดจะช่วยแก้ไขปัญหาที่เกิดขึ้นในระบบการผลิตไฟฟ้าแบบธรรมดา ทั้งในด้านประสิทธิภาพและความเป็นมิตรกับสิ่งแวดล้อม แต่สิ่งสำคัญที่จำเป็นจะต้องมีคือ ระบบการจัดการพลังงานสำหรับไมโครกริดเมื่อไมโครกริดอยู่ในสภาวะต่างๆ เพื่อให้ระบบมีเสถียรภาพและความน่าเชื่อถือ ซึ่งในความเป็นจริงแหล่งกำเนิดพลังงานของไมโครกริดนั้นอาจจะกระจายตัวอยู่ไกลกันและมีจำนวนมาก จึงจำเป็นต้องมีระบบควบคุมที่สามารถใช้ร่วมกันได้อย่างเหมาะสม ในโครงงานนี้ได้เลือกใช้เทคโนโลยีเอเจนต์ในโปรแกรม C# มาประยุกต์ใช้ในการควบคุมหน่วยการผลิตของไมโครกริด ซึ่งสามารถติดต่อสื่อสารกันได้อย่างอิสระผ่านเครือข่าย CANUSB และตัดสินใจได้ตามโปรแกรมที่เขียนไว้ มาประยุกต์ใช้ในการควบคุมหน่วยการผลิตของไมโครกริด เนื่องด้วยความคล่องตัวของเอเจนท์ซึ่งสามารถติดต่อสื่อสารกันได้อย่างอิสระผ่านระบบเครือข่ายไร้สายและตัดสินใจได้อย่างชาญฉลาดตามโปรแกรมที่เขียนไว้ ทำให้สามารถนำข้อดีตรงจุดนี้มาประยุกต์ใช้ในการรับส่งข้อมูลระหว่างหน่วยควบคุมและหน่วยการผลิตที่อาจจะอยู่ระยะไกลและมีจำนวนมาก

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 เพื่อศึกษาโครงสร้างและหลักการระบบไมโครกริดสาธิต

1.2.2 เพื่อศึกษาหลักการของระบบเอเจนท์ และการทำงานของระบบเอเจนท์บนโปรแกรม Microsoft Visual Studio 2013 ด้วยภาษา C#

1.2.3 เพื่อเป็นต้นแบบในการประยุกต์ใช้ระบบเอเจนท์เข้ามาใช้ควบคุมในระบบจัดการพลังงานของระบบไมโครกริด

1.3 ขอบเขตของการศึกษา

โครงงานนี้แบ่งเป็นสองส่วนคือ ส่วนแรก เป็นการศึกษาการพัฒนาระบบควบคุมไมโครกริดด้วยเทคนิค Angle Droop Voltage Droop และเชื่อมต่อกริด จากการสร้างแบบจำลองพื้นฐานโดยใช้โปรแกรม MATLAB/Simulink เพื่อเป็นการรักษาเสถียรภาพและความน่าเชื่อถือของระบบ รวมถึงการแสดงผลผ่านทางโปรแกรม ส่วนที่สองเป็นการรวบรวมผลศึกษาและทดสอบเกี่ยวกับการทำงานของระบบไมโครกริดและหลักการจัดการพลังงานของไมโครกริด และได้นำเอาเทคโนโลยีของระบบมัลติเอเจนท์ (Multi-Agent System) เข้ามาประยุกต์ใช้โดยการกำหนดแบบจำลองพื้นฐานของไมโครกริดโดยใช้โปรแกรม MATLAB/Simulink สร้างเอเจนท์สำหรับทำงานในส่วนต่างๆโดยชุดพัฒนาจาวาเอเจนท์ (Java Agent Development Framework) และให้เอเจนท์สามารถสื่อสารกันเองผ่านโปรแกรม Micro C เพื่อที่จะสามารถรักษาสมดุลของพลังงานไฟฟ้า รักษาคุณภาพ ความปลอดภัยและความน่าเชื่อถือของระบบไว้ได้

1.4 แผนการดำเนินงาน

การดำเนินงาน	ช่วงเวลา									
	ส.ค.	ก.ย.	ต.ค.	พ.ย.	ธ.ค.	ม.ก.	ก.พ.	มี.ค.	เม.ย.	พ.ค.
ศึกษาระบบการทำงานของไมโครกริด และ เทคนิคการควบคุมแบบ Voltage Droop และแบบ Angle Droop โดยใช้ MATLAB/Simulink										
จำลองผลการใช้ Power Amplifier Model แทน 1 Phase Generator										
เตรียมนำเสนอและจัดทำเอกสารทางวิชาการผลการศึกษาในภาคเรียนที่ 1										
ศึกษาการใช้ Agent-Based										
เขียนโปรแกรมสำหรับการแสดงผล ผ่านคอมพิวเตอร์ โดยใช้โปรแกรม C#										
นำ Agent-Based ไปควบคุมการทำงานของไมโครกริด										
เตรียมนำเสนอและจัดทำเอกสารทางวิชาการผลศึกษาและปริญญานิพนธ์										

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1.5.1 เข้าใจถึงหลักการพื้นฐานและโครงสร้างของระบบไมโครกริด

1.5.2 สามารถออกแบบและติดตั้งระบบไมโครกริดได้

1.5.3 เข้าใจหลักการใช้เทคนิค Angle Droop และ Voltage Droop โดยศึกษาจากแบบจำลองพื้นฐานโดยใช้โปรแกรม MATLAB/Simulink เพื่อพัฒนาระบบควบคุมระบบไมโครกริดสาธิตขนาดได้

1.5.4 สามารถกำหนดให้เอเจนท์แต่ละแบบมีหน้าที่แตกต่างกัน สามารถสื่อสารกันเพื่อวิเคราะห์และประมวลผลร่วมกันได้

1.5.5 สามารถประยุกต์ระบบเอเจนท์เข้ามาใช้พัฒนาระบบการจัดการพลังงานของระบบไมโครกริดให้มีประสิทธิภาพมากขึ้น

1.6 โครงสร้างของเนื้อหาภายในปริญญานิพนธ์

บทที่ 1 กล่าวถึงความเป็นมาและความสำคัญของปัญหา, วัตถุประสงค์ของโครงงาน, ขอบเขต การศึกษา, แผนการดำเนินงาน

บทที่ 2 เป็นการอธิบายทฤษฎีเกี่ยวกับโครงสร้างของไมโครกริด แหล่งพลังงานของไมโครกริด และการจัดการในระบบโครกริด และระบบมัลติเอเจนท์ซึ่งนำมาพัฒนาในส่วนของการบริหารการจัดการส่วนต่างๆ ในไมโครกริด และอธิบายเกี่ยวกับระบบเอเจนท์ว่ามีลักษณะการสื่อสารอย่างไรบ้าง

บทที่ 3 เป็นการอธิบายวิธีการใช้งานการใช้ระบบเอเจนท์โดยโปรแกรม C# ซึ่งจะอธิบายตั้งแต่การติดตั้ง วิธีการเขียนโปรแกรม ไปจนถึงวิธีการสื่อสารของเอเจนท์ที่สร้างขึ้น และการติดตั้ง โปรแกรม C# การตั้งค่าต่างๆเพื่อให้เอเจนต์ทำงานได้ และการทดสอบเพื่อให้โปรแกรมใช้งานได้

บทที่ 4 เป็นการจำลองระบบไมโครกริดด้วยเอเจนท์ซึ่งทำหน้าที่ต่างๆกันเพื่อให้ระบบทำงานได้ภายใต้เงื่อนไขต่างๆ ของระบบที่กำหนดขึ้น

บทที่ 5 เป็นการสรุปผลของโครงงานในภาพรวม ข้อเสนอแนะและข้อคิดเห็นในการนำโครงงานนี้ไปพัฒนาต่อไป

บทที่ 2

ทฤษฎีที่เกี่ยวข้อง

ระบบควบคุมไมโครกริดโดยใช้เอเจนต์(Agent Based Microgrid Control) เป็นการประยุกต์ใช้ความรู้ทางการวิเคราะห์การไหลของกำลังไฟฟ้าภายในระบบไฟฟ้าขนาดเล็กที่เรียกว่าไมโครกริด (Microgrid) เข้ากับเทคโนโลยีที่มีชื่อว่า ระบบเอเจนต์ (Agent System) ซึ่งจะการช่วยให้การทำงานของระบบไฟฟ้าขนาดเล็กหรือไมโครกริดทำงานได้อย่างมีเสถียรภาพมากขึ้น สามารถกำจัดปัญหาความไม่มีเสถียรภาพของระบบ และสามารถต่อยอดเพื่อขยายขีดความสามารถในการวิเคราะห์และตัดสินใจสั่งการระบบให้มีความชาญฉลาดมากขึ้นต่อไปในอนาคต

2.1 ระบบไมโครกริด

2.1.1 ความหมายและความเป็นมาของไมโครกริด

California Energy Commission ได้ให้นิยามของ “ไมโครกริด” ไว้ว่า ไมโครกริดคือการรวมระบบพลังงานซึ่งประกอบด้วยการเชื่อมต่อของโหลดภายในระบบกับทรัพยากรพลังงานแบบกระจาย (Distributed energy resources) หรือจะเรียกได้ว่าเป็น การต่อเชื่อมแหล่ง ผลิตไฟฟ้าขนาดเล็ก ของแต่ละครัวเรือน หรือแต่ละชุมชนเข้าสู่ระบบสายส่งแรงดันต่ำ ซึ่งต่างจากระบบไฟฟ้าที่ใช้อยู่ในปัจจุบันที่ผลิตกระแสไฟฟ้าจากโรงไฟฟ้าขนาดใหญ่ เพื่อจ่ายไฟฟ้าให้แก่ครัวเรือนและชุมชน ซึ่งไมโครกริดเป็นระบบที่สามารถปฏิบัติการคู่ขนานไปกับ กริดหลัก (Main Grid) ซึ่งเรียกว่า โหมดเชื่อมต่อกริด (Grid connect) หรือในสภาวะที่แยกตัวอิสระจากกริดหลักซึ่งเรียกว่าโหมดแยกอิสระ (Stand-alone)

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

แนวคิดไมโครกริด หรือ Microgrid Concept โดยถูกนำเสนอ ในปี 1999 ภายใต้โครงการ CERTS (Consortium for Electric Reliability Technology Solution) ซึ่งเป็นศูนย์วิจัยหลักสังกัดกระทรวงพลังงาน (Department of Energy, DOE) ของสหรัฐอเมริการ่วมกับมหาวิทยาลัย บริษัทผลิตไฟฟ้าองค์กรต่างๆภายใต้แนวคิดนี้ทั้งแหล่งกำเนิดไฟฟ้าขนาดเล็ก และโหลดภายในพื้นที่จะถูกมอง

รวมเป็นระบบอิสระขนาดเล็กระบบหนึ่ง ในสภาวะที่เกิดความผิดปกติขึ้นในระบบไฟฟ้ากำลังหลัก (Grid) ไมโครกริดจะสามารถปลดตัวเองออกจากระบบไฟฟ้ากำลังหลัก และทำงานในแบบไอส์แลนด์ (ผลิตและส่งกำลังไฟฟ้าภายในพื้นที่) ได้อย่างอัตโนมัติ และสามารถเชื่อมต่อกลับเข้าไปกับระบบไฟฟ้ากำลังหลักได้เมื่อความผิดปกติในระบบไฟฟ้ากำลังหลักได้รับการแก้ไขไปแล้ว

โดยแนวคิดไมโครกริดจะถูกมองเป็นระบบควบคุมหน่วยหนึ่งจากระบบไฟฟ้ากำลังหลัก และจะไม่สร้างปัญหาเช่นการกระเพื่อมของแรงดันไฟฟ้า การแกว่งไกวของความถี่ไฟฟ้า เป็นต้น ให้กับระบบไฟฟ้ากำลังหลัก ไมโครกริดที่มีคุณสมบัตินี้ถูกนิยามให้เป็น พลเมืองดี (Good Citizen) ของระบบกำลังไฟฟ้าหลัก ในขณะเดียวกันไมโครกริดยังอาจสามารถช่วยสนับสนุนการทำงานของระบบไฟฟ้ากำลังหลักได้ เช่น การให้บริการคุณภาพไฟฟ้าชั้นเยี่ยม (Premium Power) การรักษาระดับแรงดันไฟฟ้า (Voltage Support) เป็นต้น ไมโครกริดที่มีคุณสมบัตินี้ถูกนิยามให้เป็นพลเมืองตัวอย่าง (Model Citizen) ของระบบไฟฟ้ากำลังหลัก แนวความคิดไมโครกริดนี้ถูกให้ความสำคัญอย่างมากในหลายประเทศ เช่น

สหรัฐอเมริกาเนื่องจากเหตุการณ์ไฟฟ้าดับเป็นบริเวณ กว้างทางฝั่งตะวันตกใน ปี 1996,ฝั่งตะวันออกเฉียงเหนือ ในปี2003ทำให้ความเชื่อถือได้ (Reliability)ของระบบไฟฟ้ากำลัง และความมั่นคงใน การส่งจ่ายกำลังไฟฟ้าอย่างมีเสถียรภาพ เป็นหัวข้อที่สำคัญที่ได้รับความสนใจ สูงมาก นอกจากนี้ค่าใช้จ่ายใน การลงทุน สร้างโรงไฟฟ้าขนาดใหญ่ ระบบส่งกำลังไฟฟ้า อุปกรณ์ไฟฟ้าต่างๆ ตลอดจนความต้องการในการลดค่าใช้จ่าย ด้านพลังงานและ การใช้พลังงานอย่างมีประสิทธิภาพ ปัจจัยเหล่านี้ทำให้เกิดความต้องการไมโครกริดขึ้นมา

สหภาพยุโรปเนื่องจากความใส่ใจของประชาคมยุโรป ต่อสิ่งแวดล้อม เพิ่มสูงขึ้นจึงส่งผลให้มีการติดตั้งแหล่งผลิตกำลังไฟฟ้าด้วยพลังงานหมุนเวียนในระบบไฟฟ้า กำลังเพิ่มขึ้นอย่างมาก นอกจากนี้การส่งจ่ายกำลังไฟฟ้าให้กับเกาะ พื้นที่ห่างไกลเป็นต้น สิ่งเหล่านี้เป็นปัจจัยทำให้เกิดความต้องการระบบไมโครกริดขึ้น

ญี่ปุ่น การเพิ่มขึ้นของแหล่งผลิตกำลังไฟฟ้าด้วยพลัง งานหมุนเวียนในระบบไฟฟ้ากำลังตลอดจนความต้องการ ในด้าน การประหยัดพลังงาน การลดก๊าซเรือนกระจก และ การนำศักยภาพของทรัพยากรในแต่ละท้องถิ่นโดยเฉพาะแหล่งพลังงาน มาใช้ประโยชน์ให้เต็มที่เป็นปัจจัยทำให้เกิดความต้องการไมโครกริดขึ้น

2.1.2 ระบบผลิตไฟฟ้าแบบกระจายศูนย์ Distributed Generation (DG)

ระบบไฟฟ้าทั่วโลกเริ่มตระหนักถึงปัญหาการลดลงของปริมาณของทรัพยากรเชื้อเพลิงฟอสซิลและการทำให้พลังงานและสภาพแวดล้อมต่างๆ ค่อยๆ ทรุดโทรมลงไป ปัญหาเหล่านี้ได้นำไปสู่แนวความคิดใหม่ในการผลิตพลังงานภายในพื้นที่ ณ ระดับการกระจายแรงดันต่างๆ โดย

การใช้พลังงานทดแทนและพลังงานหมุนเวียน เช่นก๊าซธรรมชาติ ก๊าซชีวภาพ พลังงานลม เซลล์แสงอาทิตย์ เซลล์เชื้อเพลิงความร้อนร่วม (Combined heat and power : CHP) ระบบไมโครเทอร์ไบน์ และเครื่องยนต์สเตอร์ลิง (Stirling engine) ซึ่งแหล่งพลังงานเหล่านี้จะเป็นประโยชน์ต่อเครือข่ายการจำหน่าย การผลิตไฟฟ้าในรูปแบบนี้จะเรียกว่า ระบบผลิตไฟฟ้าแบบกระจายศูนย์หรือ Distributed Generation (DG) และแหล่งพลังงานเหล่านี้จะเรียกว่า แหล่งพลังงานแบบกระจายศูนย์หรือ Distributed energy resources (DER) โดยการผลิตแบบกระจายศูนย์นี้ได้รับการออกแบบให้มีความแตกต่างจากการผลิตไฟฟ้าจากส่วนกลาง (Centralized conventional generation) ในสมัยก่อน และการใช้ระบบจำหน่ายร่วมกับการรวมตัวของ 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. มีขนาดเล็กกว่า50 เมกะวัตต์

3.แหล่งกำเนิดกำลังไฟฟ้าหรือเครื่องกำเนิดไฟฟ้าแบบกระจายศูนย์มักจะเชื่อมต่อกับระบบจำหน่ายที่มีขนาดของแรงดัน 230/415 V ถึง 145 kV

เนื่องจากไมโครกริดไม่ได้มุ่งหมายเพื่อการขายไฟฟ้าเข้ากับระบบไฟฟ้ากำลังหลัก หากแต่มุ่งเน้นการผลิตและจำหน่ายไฟฟ้าในพื้นที่ของตัวเอง การเพิ่มขึ้นของไมโครกริด รวมถึง DER จะไม่ทำให้เกิดปัญหากับระบบไฟฟ้ากำลังหลัก ในขณะเดียวกันก็ช่วยเสริมความมั่นคงของการส่งจ่ายไฟฟ้า รวมทั้งความเชื่อถือได้ของระบบไฟฟ้าและคุณภาพไฟฟ้าในพื้นที่ของไมโครกริดความต้องการเสริมความมั่นคงทางพลังงานของประเทศก็จะได้รับการตอบสนอง ขณะที่ผลกระทบต่อสิ่งแวดล้อมจากการผลิตไฟฟ้าก็น้อยลงจากการใช้พลังงานหมุนเวียน และประสิทธิภาพการใช้พลังงานโดยรวมที่เพิ่มขึ้น นอกจากนี้เนื่องจาก DER มีขนาดเล็กการวางแผนและติดตั้งจึงใช้เวลาและค่าใช้จ่ายน้อยกว่าระบบการผลิตขนาดใหญ่ จึงมีส่วนช่วยกระจายความเสี่ยงของการลงทุนด้านการผลิตไฟฟ้าได้อีกทางหนึ่งจากที่กล่าวมาอาจสามารถสรุปความสำคัญของแนวคิดไมโครกริดได้ดังนี้

1. การรักษาคุณภาพกำลังไฟฟ้าและความน่าเชื่อถือได้ของระบบไฟฟ้า

2. การลดระยะเวลาและค่าใช้จ่ายในการติดตั้ง

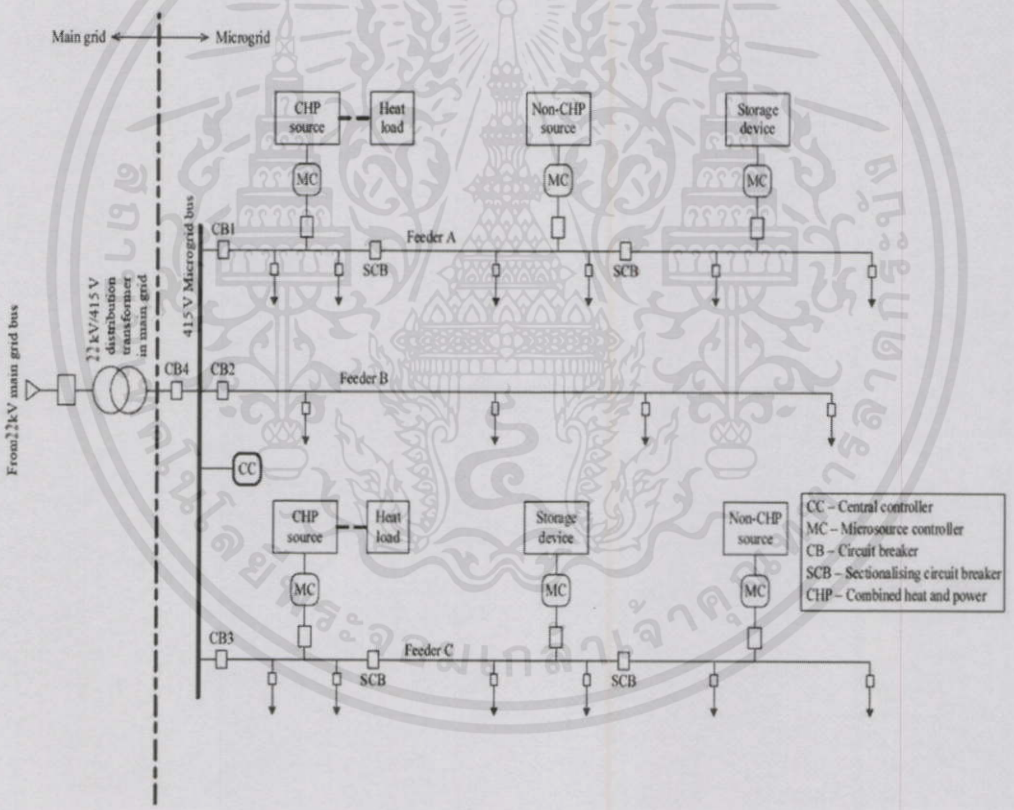
3. การลดค่าใช้จ่ายทางพลังงานหรือค่าเชื้อเพลิงลง

4. การลดผลกระทบต่อสิ่งแวดล้อม

5. การรักษาความมั่นคงทางพลังงาน

2.1.3 ส่วนประกอบต่างๆของไมโครกริด และหน้าที่ของแต่ละส่วนประกอบ

ส่วนประกอบต่างๆของไมโครกริดเป็นไปตามรูปที่ 2.1 จะประกอบด้วยภาระทางไฟฟ้าหรือความร้อน รวมทั้งแหล่งจ่ายขนาดเล็ก (Micro source) โดยมีการเชื่อมต่อในโครงข่ายระบบจำหน่ายแรงดันต่ำ จะเห็นว่าภาระจะถูกวางไว้ใกล้กับแหล่งจ่ายเพื่อลดการสูญเสียโดยเฉพาะภาระทางความร้อน (CHPsource)



รูปที่ 2.1 ส่วนประกอบต่างๆของไมโครกริด [1]

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

ไมโครกริดดำเนินการในสองโหมด : (1) เชื่อมต่อกริด และ (2) ในโหมดแยกอิสระ โดยในโหมดเชื่อมต่อกริดนั้น ไมโครกริดจะยังคงเชื่อมต่อกับกริดหลักทั้งหมดหรือบางส่วนและนำเข้าหรือส่งกำลังไปยังกริดหลัก 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

Micro source Controller (MC) : คือการควบคุมที่เป็นอิสระของการไหลของกำลังไฟฟ้าและแรงดันที่บัสปลายของแหล่งจ่ายขนาดเล็กในการตอบสนองต่อการรบกวนรวมถึงการเปลี่ยนแปลงของภาระ ซึ่งคาดว่าเป็นอิสระในที่นี้หมายถึงการควบคุมโดยไม่มีการสั่งการจากตัวควบคุมส่วนกลาง(CC) นั่นเองและตัว MC นี้จะต้องทำให้มั่นใจได้ว่าแหล่งจ่ายขนาดเล็กแต่ละตัวจะต้องสามารถเพิ่มระดับการผลิตได้อย่างทันท่วงทีเพื่อจ่ายกำลังไฟฟ้ารวมถึงมีการจัดสรรการผลิตให้กับภาระต่างๆในโหมดแยกอิสระและสามารถกลับมาเชื่อมต่อในโหมดของการเชื่อมต่อกริดได้อย่างอัตโนมัติโดยไม่มีการสั่งการจาก CC ซึ่งคุณลักษณะที่สำคัญที่สุดของ MC คือการตอบสนองที่รวดเร็วต่อการตรวจสอบระดับแรงดันและกระแสภายในพื้นที่โดยไม่มีการนำเอาข้อมูลจาก MC ข้างเคียงมาพิจารณา สองคุณลักษณะพิเศษหลักของ MC คือ MC หนึ่งตัวจะทำงานโดยแยกอิสระไม่ขึ้นกับ MC ตัวอื่นๆในไมโครกริดและมันจะสามารถทำการยกเลิกคำสั่งที่มาจาก CC ที่วิเคราะห์แล้วว่าจะก่อให้เกิดอันตรายต่อแหล่งจ่ายขนาดเล็กภายในระบบ

Central controller (CC): จะจัดการควบคุมการดำเนินงานของไมโครกริดและการป้องกันผ่าน MC โดยมีวัตถุประสงค์ (i) เพื่อรักษาระดับแรงดันและความถี่ของภาระปลายทาง ผ่านการควบคุมกำลังไฟฟ้าและความถี่รวมถึงการควบคุมแรงดันไฟฟ้า (ii) เพื่อเพิ่มประสิทธิภาพของพลังงานในไมโครกริดโดย CC ได้ถูกออกแบบมาให้ทำงานในโหมดอัตโนมัติเมื่อมีการแทรกแซงการทำงานของระบบและในยามจำเป็น

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

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

Protection Co-ordination Moduleหรือ PCM จะทำการตอบสนองต่อความผิดพร่องในไมโครกริดรวมถึงกริดหลัก และสภาวการณ์หายไปของกริด (Loss of Grid: LOG) ด้วยคุณลักษณะดังกล่าวจึงทำให้มั่นใจในการทำงานร่วมกันในการป้องกันไมโครกริดได้อย่างถูกต้อง และยังรวมถึงการปรับตัวต่อการเปลี่ยนแปลงต่อระดับกระแสผิดพร่องในระหว่างเกิดการเปลี่ยนสภาวะการทำงานจากโหมดเชื่อมต่อกริดไปเป็นโหมดแยกอิสระ

2.1.4 ตัวควบคุมในระบบไมโครกริด

ในระบบไมโครกริดจะมีตัวควบคุมหลักๆ อยู่ 2 ชนิดคือตัวควบคุมแหล่งจ่ายและตัวควบคุมส่วนกลาง

(1) ตัวควบคุมแหล่งจ่าย (Micro source Controller: MC)

แหล่งจ่ายขนาดเล็ก (Micro source) และอุปกรณ์สำรองพลังงาน (Storage Device) ที่ติดตั้งเข้ากับ MC จะต้องสามารถดำเนินการได้อย่างราบรื่นและมีความยืดหยุ่น เพื่อตอบสนองความต้องการของผู้บริโภคและสาธารณูปโภค การทำงานของ MC นั้นอาจดำเนินการโดยมีการแทรกแซงหรือไม่มีการสั่งการจาก CC ก็ได้ ฟังก์ชันการทำงานของ MC ส่วนมากจะขึ้นกับการเชื่อมต่อด้านอิเล็กทรอนิกส์กำลัง (Power Electronics Interface) ที่ใช้ในแหล่งจ่ายรวมถึงอุปกรณ์สำรองพลังงาน เงื่อนไขการดำเนินงานของ MC 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 ควรตรวจสอบว่าแหล่งจ่ายแต่ละตัวนั้นสามารถรองรับการจัดสรรจ่ายโหลดได้อย่างรวดเร็ว เมื่อไมโครกริดปลดตัวเองออกจากกริดหลัก

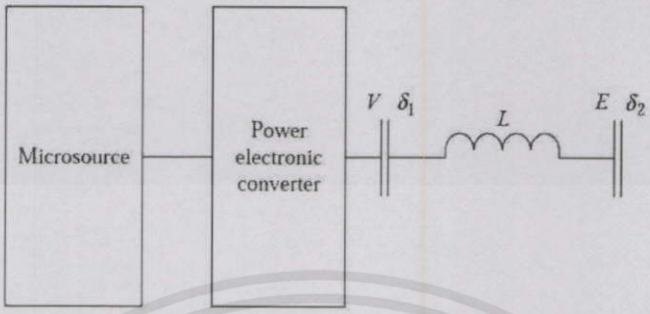
<u>การควบคุมกำลังไฟฟ้าจริงและกำลังไฟฟ้าเสมือน (Active and Reactive Power Control)</u>

แหล่งจ่ายขนาดเล็กที่เรียกว่า Micro source อาจเป็นแหล่งจ่ายแหล่งจ่ายกระแสตรง อย่างเช่น เซลล์แสงอาทิตย์ (PV) เซลล์เชื้อเพลิง (Fuel cell) อุปกรณ์สำรองพลังงาน (Storage Battery) หรือเป็นแหล่งจ่ายกกระแสสลับอย่างพวกกังหันแก๊สขนาดเล็ก (Micro turbine) หรือ กังหันลม สำหรับแหล่งจ่ายกระแสตรงนั้นจะถูกแปลงเป็นกระแสสลับโดยตรงที่ P-f (50/60 Hz) แต่ขณะที่กระแสสลับซึ่งมีความถี่ที่ไม่เป็นที่แน่นอนจะถูกแปลงเป็นกระแสตรงก่อนและจึงแปลงกลับเป็นกระแสสลับอีกครั้งที่ความถี่ปกติด้วยอินเวอร์เตอร์ ซึ่งทั้งสองอย่างต่างก็ต้องใช้ DC/AC Converter

ในรูปที่ 2.2 แสดงโครงสร้างทั่วไปของ MC ซึ่งประกอบด้วยแหล่งจ่ายขนาดเล็ก (Micro source) และอินเวอร์เตอร์อิเล็กทรอนิกส์กำลัง (Power Electronic Converter) โดยอินเวอร์เตอร์ของแหล่งจ่ายแรงดัน ในส่วนของคอนเวอร์เตอร์จะควบคุมขนาดแรงดัน (V) และ มุมเฟส (δ_1) ของแรงดันขาออก $(V\angle\delta_1)$ ที่ปลายบัส 1 (Bus-1) Micro source 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 และ ขนาดแรงดันบัส E ซึ่งเป็นไปตาม สมการที่ 2.1 และ สมการที่ 2.2 ตามลำดับ ดังนี้

$$P = \frac{3EV}{2X}\sin\delta \qquad (2.1)$$

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

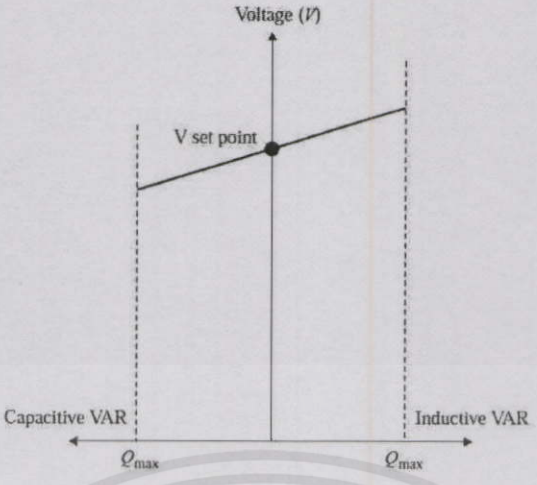


รูปที่ 2.2 รูปแบบทั่วไปของ MC [1]

การควบคุมแรงดัน (Voltage Control)

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

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



รูปที่ 2.3 คุณลักษณะดรูปสำหรับ V-Q droop controller [1]

การควบคุมแรงดันที่บัสของไมโครกริดนั้จำเป็นจะต่อภาพรวมของเสถนียรภาพและความน่าเชื่อถือของไมโครกริด ไมโครกริดที่มีแหล่งจ่ายจำนวนมากอาจมีปัยหาใหเรื่องการแกว่งของกำลังเสมือน (Reactive Power) หากปราศจากการควบคุมแรงดันที่เหมาะสม เช่นเดียวกับเครื่องกำเนิดไฟฟ้าซิงโครนัสขนาดใหญ่ ฟังก์ชันควบคุมแรงดันของ Micro source จะจัดการปัญหากระแสไหลวน (Circulating Reactive Current) ระหว่างสองไมโครกริดให้น้อยลง สำหรับกริดหลักนั้น กระแสไหลวนเหล่านี้โดยทั่วไปจะถูกจำกัดโดยอิมพิแอนซ์ขนาดใหญ่ระหว่างเครื่องกำเนิดไฟฟ้าแต่ในส่วนของไมโครกริดนั้นปัญหาค่อนข้างจะชัดเจน เนื่องจากสายป้อนมักต่อแบบราก(Radial) ซึ่งมีอิมพิแดนซ์ระหว่างแหล่งจ่ายต่ำบางครั้งกระแสไหลวนเหล่านี้อาจเกินค่าพิกัดกระแสของ Micro source แม้ว่าจะมีการเปลี่ยนแปลงที่เล็กน้อยในจุดอ้างอิงแรงดัน ฉะนั้นในไมโครกริดกระแสไหลวนเหล่านี้สามารถถูกควบคุมโดยวิธี (Voltage-Reactive Power V-Q) droop controller ดังกราฟแสดงคุณลักษณะลดลงของแรงดันซึ่งแสดงในรูปที่ 2.3

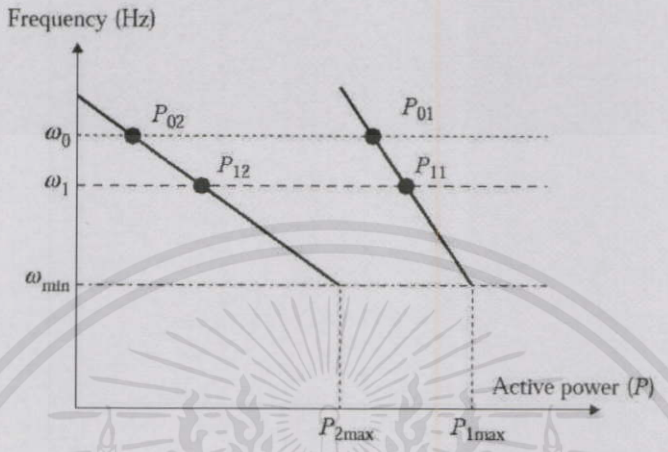
<u>ข้อกำหนดในการสำรองพลังงานเพื่อสมดุลโหลด</u>

สำหรับการทำงานของไมโครกริดโหมดการเชื่อมต่อกริดความสมดุลของกำลังเริ่มต้นขณะที่มีการต่อภาระเพิ่มเข้ามาใหม่จะได้รับการดูแลโดยความเฉื่อยที่มีมากของเครื่องกำเนิดไฟฟ้า แต่สำหรับการดำเนินงานในโหมดแยกอิสระ ไมโครกริดจะต้องมีการรักษาสมดุลของกำลังเริ่มต้นโดยการใช้อุปกรณ์จัดเก็บสำรองพลังงานที่มีประสิทธิภาพพอเป็นความเฉื่อยในระบบไมโครกริด อุปกรณ์สำรองพลังงานกระแสตรงจะถูกเชื่อมต่อที่บัสกระแสตรง (DC bus) ของ แหล่งจ่ายขณะที่อุปกรณ์สำรองพลังงานกระแสสลับจะถูกเชื่อมต่อโดยตรงที่บัสของไมโครกริด ดังนั้น MC จะสร้างความมั่นใจในการใช้อุปกรณ์สำรองพลังงานที่เหมาะสมสำหรับการติดตามโหลดอย่างรวดเร็ว

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

ตัวควบคุมไมโครกริดต้องทำให้แน่ใจว่าเกิดความเรียบและเปลี่ยนแปลงโดยอัตโนมัติจากโหมดเชื่อมต่อกริดเป็นโหมดแยกอิสระ ซึ่งหลักการนี้จะมีความคล้ายกับการดำเนินการของระบบสำรองไฟฟ้า (Uninterrupted Power Supply : UPS) โดยในขณะที่เกิดการเปลี่ยนแปลงเป็นโหมด

แยกอิสระนั้น MC ของแต่ละแหล่งจ่าย จะใช้การควบคุมกำลังและความถี่ เพื่อเปลี่ยนจุดที่ดำเนินการเพื่อให้เกิดความสมดุลของกำลังไฟฟ้าเมื่อปรับเปลี่ยนโหมดใหม่ 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 เมื่อความต้องการโหลดเปลี่ยนไป แหล่งจ่ายจะทำงานที่ความถี่ที่แตกต่างอันเป็นเหตุมาจากการเปลี่ยนแปลงของมุมกำลังและความถี่ในการทำงานที่ลดลงจากปกติกับสัดส่วนความแตกต่างของภาระ ซึ่งจะเกิดเป็นคุณลักษณะของกำลังกับความถี่ (P-f) ดังรูปที่ 2.4 เมื่อความถี่ในไมโครกริดลดลงแล้ว ดังนั้น MC ก็ต้องรวบรวมฟังก์ชันเพื่อฟื้นฟูการทำงานที่ความถี่พิกัดกับการแบ่งสัดส่วนโหลดที่เหมาะสมและควบคุมความถี่ไม่ให้เกินช่วง ±5%

(2) ตัวควบคุมส่วนกลาง (Central Controller: CC)

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

โมดูลการจัดการพลังงาน

โมดูลการจัดการพลังงาน (Energy Management Module : EMM) ประกอบด้วยฟังก์ชันการควบคุมต่างๆ สำหรับการควบคุมพลังงานในไมโครกริดได้อย่างเหมาะสม ในส่วนนี้จะ

อธิบายเกี่ยวกับพื้นฐานของ EMM ที่รวบรวมเอาพื้นฐานฟังก์ชันการควบคุมที่จำเป็นสำหรับการทำงานที่น่าพึงพอใจของไมโครกริด จำนวนฟังก์ชันการควบคุมสามารถเพิ่มได้ เพื่อให้บรรลุถึงรายละเอียดย่อยและควบคุมได้ซับซ้อน

ฟังก์ชันพื้นฐานในการควบคุมแหล่งจ่าย

EMM แบบพื้นฐานจะทำการหาจุดอ้างอิงของกำลังไฟฟ้าจริงและแรงดันของ MC ในขณะที่พื้นฐานควบคุมแหล่งจ่ายจะทำงานผ่าน MC เท่านั้น ค่าแรงดันอ้างอิงจะถูกเก็บในแถบที่ตั้งไว้เพื่อให้ได้แรงดันไฟฟ้าที่เหมาะสมในการรักษาระดับแรงดันในไมโครกริด

<u>การควบคุมแรงดัน (Voltage control)</u>

โดยปกติโหลดและตัวประกอบกำลังในไมโครกริดจะถูกควบคุมโดยการเปลี่ยนขนาดแรงดันและมุมเฟสของแหล่งจ่ายเพื่อหลีกเลี่ยงความซับซ้อนในการควบคุม EMM การควบคุมแรงดันและตัวประกอบกำลังของแหล่งจ่ายจะถูกบังคับผ่าน MC และไม่ผ่าน EMM ซึ่ง EMM จะทำเพียงแค่หาค่าแรงดันอ้างอิงไปยัง MC สำหรับบัสที่สำคัญบางบัสของไมโครกริด เมื่อสายป้อนในการจำหน่ายไม่ได้จ่ายโหลดเต็มพิกัดในไมโครกริด อาจมีแนวโน้มแรงดันเพิ่มขึ้นที่สายป้อน ในการหยุดการเพิ่มของแรงดันนั้น MC จะต้องคอยตรวจสอบแรงดันและจัดการป้อนค่ากลับไปที่ EMM จากนั้น EMM จะนำค่าแรงดันอ้างอิงไปยัง MC เพื่อปรับให้ได้แรงดันที่ต้องการ เป้าหมายของวิธีการควบคุมเช่นนี้คือ เพื่อควบคุมให้ภาระทำงานที่เป็นค่าตัวประกอบกำลังไฟฟ้าเท่ากับ 1 (unity p.f.) และควบคุมให้แรงดันไม่ใช้เกินช่วง ±5% [28]

<u>การควบคุมตัวประกอบกำลัง (Power factor control)</u>

จะไม่เหมือนกับเครื่องกำเนิดไฟฟ้าซิงโครนัสทั่วไปแหล่งจ่ายมักจะไม่มีกาควบคุมตัวประกอบกำลังภายในตัวมันเอง ดังนั้นตัวประกอบกำลังจึงขึ้นอยู่กับโหลด MC ทุกตัวจะมีการควบคุมค่าตัวประกอบกำลัง พร้อมคุณสมบัติฟังก์ชันการติดตามโหลด อย่างไรก็ตามสามารถนำเอาอิเล็กทรอนิกส์กำลังของแหล่งจ่ายบางตัวซึ่งอาจประกอบด้วยการควบคุมตัวประกอบกำลังมาควบคุมมุมเฟสของกระแสและลดฮาร์มอนิกส์ให้น้อยที่สุด คุณลักษณะการควบคุมตัวประกอบกำลังจะถูกจัดรวมไว้ใน MC ดังเช่นนั้นแล้ว จึงไม่ปรากฏคำสั่งใดๆ จาก EMM อีก ยกเว้นค่าแรงดันอ้างอิง

<u>การควบคุมความเร็วต้นกำลัง (Prime mover speed control)</u>

คุณลักษณะนี้จะมีสำหรับแหล่งจ่ายที่มีการหมุนด้วยต้นกำลังเช่น กังหันแก๊สขนาดเล็ก(Micro turbine) หรือ กังหันลม เพื่อรองรับการเปลี่ยนแปลงโหลดของไมโครกริด ต้นกำลังของแหล่งจ่ายจะต้องปรับความเร็วให้สมดุลกับภาวะโหลดใหม่ ในที่นี่ต้นกำลังที่ความเร็วคงที่ควรจะเปลี่ยนเชื้อเพลิงที่ป้อนเข้าไป ซึ่งจะส่งผลต่อประสิทธิภาพของต้นกำลังโดยประสิทธิภาพจะเป็นฟังก์ชันของเชื้อเพลิงที่ใช้และความเร็ว ดังนั้น การควบคุมความเร็วของต้นกำลังควรทำให้เกิดความมั่นใจว่าจะทำให้แหล่งจ่ายเกิดประสิทธิภาพสูงสุด ในการออกแบบเบื้องต้น การควบคุมนี้จะถูกบังคับผ่าน MC

<u>การรักษาระดับความถี่ (Frequency regulation)</u>

ในระบบไฟฟ้ากำลังทั่วไป ความถี่ของแรงดันที่ผลิตออกมาจะขึ้นกับความเร็วของเครื่องกำเนิดไฟฟ้าซิงโครนัส ในทางกลับกัน สำหรับไมโครกริดนั้น สามารถผลิตกำลังได้ตามต้องการด้วยการช่วยจากระบบคอนเวอร์เตอร์อิเล็กทรอนิกส์ (Power electronic converter) ของ MC ของมันเอง ในโหมดเชื่อมต่อกับกริด MC ไม่จำเป็นต้องสั่งการควบคุมกำลังและความถี่ (P-f control) ผ่านคุณลักษณะดรูปของกำลังและความถี่เพราะการเปลี่ยนความถี่จะสามารถดูแลตัวมันเองได้ แต่อย่างไร

ก็ตาม ในโหมดแยกอิสระ MC จำเป็นต้องสั่งการควบคุมกำลังและความถี่ เพื่อรองรับการเปลี่ยนแปลงโหลดในระบบความถี่คงที่ สำหรับทั้งสองโหมดการทำงาน EMM จะไม่รบกวนคุณลักษณะการควบคุมของ MC อย่างไรก็ตาม EMM ก็ยังคอยตรวจสอบความถี่ในไมโครกริดตลอด และเมื่อใดที่ความถี่ตกและไม่ได้รับการฟื้นฟูจาก MC ภายในระยะเวลาที่กำหนด EMM จะทำการปลดโหลดอย่างรวดเร็วในยามฉุกเฉิน เพื่อให้เกิดความสมดุลของกำลังไฟฟ้าและทำให้เกิดเสถียรภาพในไมโครกริด

2.1.5 การดำเนินการของ EMM ในไมโครกริดทั่วไป

เพื่อให้ง่ายต่อการควบคุมไมโครกริด จำนวนฟังก์ชันในการควบคุมใน EMM จะถูกจำกัดเพียงแค่ส่วนที่เป็นพื้นฐานเท่านั้น ทำให้มีสัญญาณป้อนกลับที่ต้องการน้อยที่สุดโดย EMM จาก MC เพื่อส่งคำสั่งที่สำคัญไปยังแหล่งจ่ายวิธีการดำเนินการจะอธิบายดังต่อไปนี้

<u>การทำงานในโหมดเชื่อมต่อกับกริด (Grid-connected operation)</u>

ในโหมดเชื่อมต่อกับกริดสัญญาณควบคุมของ EMM จะถูกจำกัดไปยังกำลังไฟฟ้าจริงและค่าแรงดันอ้างอิงของแหล่งจ่าย การควบคุมแรงดันและตัวประกอบกำลังจะถูกสั่งการโดย MC เพื่อให้ไม-โครกริดสามารถใช้ประโยชน์ในการควบคุมให้โหลดทำงานที่ค่าตัวประกอบกำลังไฟฟ้าเท่ากับ 1 (unity p.f.) ดังนั้น EMM จึงไม่ได้สั่งการควบคุมแรงดันเพิ่มเติมและการขนานคาปาซิเตอร์ของกริดหลักและ MC ในไมโครกริดแต่อย่างใด ถ้าสายป้อนในการจำหน่ายรับภาระโหลดน้อยๆ และอาจทำให้เกิดแรงดันเพิ่มขึ้นจะถูกจัดการโดยตัวควบคุมกริดหลัก (Utility controller) อย่างไรก็ตาม EMM จะสั่งการควบคุมแรงดันของแหล่งจ่ายเพียงบางบัสที่สำคัญของไมโครกริดเท่านั้น

<u>การทำงานในโหมดแยกอิสระ (Stand-alone operation)</u>

ในโหมดแยกอิสระนั้น ฟังก์ชันหลักของ EMM คือหาค่ากำลังไฟฟ้าจริงและค่าแรงดันอ้างอิงของ MC ความถี่และการไหลของกำลังไฟฟ้าเสมือนจะถูกควบคุมโดย MC ผ่านคุณลักษณะดรูปของกำลังจริง-ความถี่ และ กำลังเสมือน-แรงดัน (P-f และ Q-V droop characteristic) EMM จะไม่ติดต่อกับสัญญาณคำสั่งสำหรับควบคุมมุมเฟสและความถี่ไปยัง MC แต่จะคอยตรวจจับความถี่ของไมโครกริดอย่างต่อเนื่องและทำการปลดโหลดอย่างรวดเร็วผ่าน MC กรณีที่ความถี่ไม่ได้รับการฟื้นฟูภายในเวลาที่กำหนด สำหรับระบบที่มั่นใจในเสถียรภาพ สำหรับการดำเนินการแบบแยกอิสระนั้น ฟังก์ชันควบคุมจะตอบสนองให้เกิดความสมดุลของโหลดอย่างรวดเร็ว ทั้งนี้เป็นสาเหตุในส่วนของความจุในการผลิต ไมโครกริดที่แยกออกไปจะไม่มีความเสถียรเท่ากับการทำงานในโหมดที่เชื่อมต่อกับกริด

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

2.1.6 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 ทำให้เกิดผลกระทบในเชิงลึกในรีเลย์ป้องกันกระแสเกิน (overcurrent relay) ทั่วไปที่จะดำเนินการตรวจจับกระแสลัดวงจร คุณลักษณะที่สำคัญของ PCM คือ ความสามารถในการแยกความแตกต่างระหว่างความต้องการในการป้องกันสำหรับทั้งสองโหมดการทำงาน และระบุเหตุที่เกิดอย่างสอดคล้องกัน

<u>การป้องกันในโหมดเชื่อมต่อกับกริด</u>

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-synchronization)

<u>การป้องกันในโหมดแยกอิสระ</u>

เมื่อไมโครกริดดำเนินการในโหมดแยกอิสระระดับการลัดวงจรที่บัสของไมโครกริดจะลงลดอย่างเห็นได้ชัด ทั้งนี้เป็นเพราะแหล่งจ่ายที่มีระบบคอนเวอร์เตอร์อิเล็กทรอนิกส์ (power electronics converter) อาจลดการจ่ายกระแสฟอลต์ขึ้นเป็น 200% ของกระแสโหลด ดังนั้นไมโครกริดในโหมดแยกอิสระจะมีกระแสฟอลต์ที่ต่ำกว่ามากเมื่อเทียบกับในโหมดการเชื่อมต่อกับกริด กระแสฟอลต์ที่ต่ำอาจไม่สามารถตรวจจับด้วยรีเลย์กระแสเกินที่ใช้ในระบบป้องกันดั้งเดิม เพราะฉะนั้นจึงเป็นผลกระทบที่สำคัญในเรื่องของการตรวจจับฟอลต์ของรีเลย์ป้องกันของไมโครกริดที่ตรวจจับด้วยเซนเซอร์ตรวจจับกระแสฟอลต์ (fault current sensing) รีเลย์พื้นฐานอาจใช้เวลานานในการตรวจจับกระแสฟอลต์ หรืออาจตรวจจับไม่ได้เลย ดังนั้นจึงมีวิธีอื่นในการป้องกัน เช่น impedanceprotection,differentialcurrent/voltagerelaying,zerosequencecurrent/voltage relaying, directional overcurrent/earth-fault อาจถูกมาปรับใช้ในไมโครกริดในโหมดแยกอิสระ

2.1.7 การเชื่อมต่อของไมโครกริด

เนื่องจากไมโครกริดถูกออกแบบเพื่อผลิตพลังงานในระบบการจำหน่ายพร้อมกับการใช้ประโยชน์จากความร้อนเหลือทิ้งจึงทำให้เกิดการควบคุมพิกัดการส่งจ่ายพลังงาน ซึ่งความจุสูงสุดของไมโครกริดจะถูกจำกัดโดยปกติที่ประมาณ 10 MVA จากข้อเสนอแนะตาม IEEE ดังนั้นถ้าจะจ่าย

ภาระทางไฟฟ้าขนาดใหญ่แล้วเราสามารถทำได้โดยการรวมเอาการผลิตจากหลายๆไมโครกริดผ่านทางเครือข่ายระบบจำหน่ายกลางได้โดยการแยกกลุ่มของภาระไฟฟ้าภายหน่วยของโหลดที่สามารถควบคุมได้ คู่กับแต่ละหน่วยการผลิตซึ่งรองรับการผลิตด้วยระบบของไมโครกริด ด้วยวิธีนี้ไมโครกริดจึงสามารถเชื่อมโยงไปยังกลุ่มพลังงานที่มีขนาดใหญ่กว่า เพื่อตอบสนองความต้องการไฟฟ้าที่มีจำนวนมาก สำหรับการเชื่อมต่อของไมโครกริดนั้น CC จะต้องดำเนินการในรูปแบบประสานกันกับ CC ที่อยู่ใกล้เคียง ดังนั้นการเชื่อมโยงไมโครกริดจะทำให้มีเสถียรภาพและการควบคุมดีขึ้นได้โดยการจัดการโครงสร้าง นอกจากนี้ยังจะมีความซับซ้อนมากขึ้นเพื่อให้มีความเชื่อมั่นที่ดีขึ้น

ด้วยหลักการดังกล่าว การประสานงานและการสื่อสารแลกเปลี่ยนข้อมูลกันระหว่างหน่วยการผลิตและหน่วยผู้บริโภคจึงเป็นสิ่งสำคัญ

2.1.8 แหล่งพลังงานในไมโครกริด (Resources in microgrid)

พลังงานทดแทนหรือแหล่งกำเนิดไฟฟ้าแบบใหม่ๆ ได้ถูกนำมาใช้ในไมโครกริด ซึ่งถูกเรียกว่า แหล่งกำเนิดไฟฟ้า ณ จุดใช้งาน (distributed energy resource : DER) หรือ แหล่งจ่ายขนาดเล็ก (Micro sources) 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 (co-generation) ที่ผลิตทั้งพลังงานไฟฟ้าและพลังงานความร้อน ซึ่งมีแนวโน้มอย่างมากที่จะนำมาใช้กับไมโครกริด ข้อดีก็คือมีประสิทธิภาพทางพลังงานโดยการใช้ความร้อนทิ้งที่ได้จากครัวเรือนหรือกระบวนการทางอุตสาหกรรม นอกจากนี้ยังสามารถใช้ในการชิลเลอร์สำหรับระบายความร้อนการดูดซึม พร้อมกันของการผลิตไฟฟ้าความร้อนและระบายความร้อนเป็นที่รู้จักกันเป็น tri-generation หรือ poly-generation ระบบ CHP ช่วยให้การใช้พลังงานที่ดีขึ้นกว่ารุ่นเดิมที่อาจถึงประสิทธิภาพมากกว่า 80% เมื่อเทียบกับที่ประมาณ 35% สำหรับโรงไฟฟ้าแบบเดิม สำหรับการผลิตความร้อนร่วมขนาดเล็ก (Micro – CHP) จะถูกติดตั้งในสถานที่มีขนาดเล็กเช่น บ้านหรืออาคารขนาดเล็ก ส่วนใหญ่หน่วย CHP ในอุตสาหกรรมการผลิตไฟฟ้าเป็นผลิตภัณฑ์หลักส่วนความร้อนเป็นผลพลอยได้ ในขณะที่ระบบ Micro CHP การสร้างความร้อนเป็นผลิตภัณฑ์หลักส่วนไฟฟ้าเป็นผลพลอยได้ Micro-CHP มีความเชื่อถือได้ แข็งแกร่งและราคาถูก มีกำลังการผลิตในช่วง 10-100 กิโลวัตต์

Micro CHP สามารถพัฒนาจากเทคโนโลยีหลายเทคโนโลยี เช่น เครื่องยนต์สเตอรลิง (Stirling engines) Reciprocating engines (เครื่องยนต์สันดาปภายใน, Internal Combustion engine) และเซลล์เชื้อเพลิง (Fuel Cell) โดยที่เทคโนโลยีที่กล่าวถึงจะทำหน้าที่ในการผลิตกระแสไฟฟ้าและนำความร้อนเหลือใช้ในการผลิตกระแสไฟฟ้าไปใช้ในการทำความร้อน และถ้าจะ

ผลิตความเย็นก็จะใช้ความร้อนที่ได้นี้ไปเป็นพลังงานสำหรับเครื่องทำความเย็นแบบดูดซึม (absorption) เพื่อผลิตความเย็นต่อไป

Wind Energy Conversion

เป็นการเปลี่ยนรูปพลังงานลมเป็นพลังงานไฟฟ้า โดยมีหลักการพื้นฐานโดยใช้กังหันลมต่อควบกับเครื่องกำเนิดไฟฟ้าผ่านระบบเกียร์ (gearbox) ซึ่งเครื่องกำเนิดไฟฟ้าที่ใช้จะเป็นเครื่องกำเนิดไฟฟ้าเหนี่ยวนำ กังหันลมจะจับพลังงานจลน์จากการไหลของลมผ่านใบพัดโรเตอร์และส่งพลังงานไปยังเครื่องกำเนิดไฟฟ้าเหนี่ยวนำผ่านระบบเกียร์เพลาของเครื่องกำเนิดไฟฟ้าจะถูกขับด้วยกังหันเปลี่ยนเป็นพลังงานไฟฟ้า สำหรับระบบเกียร์นั้นมีไว้เพื่อช่วยปรับระดับความเร็วของโรเตอร์ก่อนที่จะเข้าไปยังเครื่องกำเนิดไฟฟ้ากังหันลมที่ใช้มีทั้งแบบลักษณะการหมุนแนวตั้ง (Horizontal axis) และหมุนในแนวนอน (Vertical axis) ขนาดกำลังการผลิตนั้นปัจจุบันสามารถผลิตได้สูงถึง 5 เมกะวัตต์ กำลังการผลิตนั้นมีปัจจัยขึ้นอยู่กับสภาพอากาศ โดยจะแปรผันตรงกับความเร็วลม ขนาดใบพัด และความหนาแน่นของอากาศ

Solar Photovoltaic

พลังงานแสงอาทิตย์ เกิดจากปฏิกิริยาฟิวชั่นของดวงอาทิตย์ จะปล่อยพลังงานออกมาในรูป คลื่นแม่เหล็กไฟฟ้า ที่เรียกว่า รังสีแสงอาทิตย์ (Solar Radiation) รังสีนี้จะแพร่กระจายออกทุกทิศทุกทาง โลกของเราก็ได้รับอิทธิพลของรังสีนี้ โดยมีความเข้มของรังสีที่ตกลงบนผิวโลกประมาณ 961 ถึง 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 เนื่องจากการใช้เชื้อเพลิงฟอสซิลในการผลิตพลังงาน จึงหันมาใช้เซลล์แสงอาทิตย์เพิ่มขึ้นเรื่อย ๆ

Small-scale hydroelectric

เป็นการกำเนิดไฟฟ้าโดยใช้กังหันน้ำขนาดเล็กในการขับเคลื่อนเครื่องกำเนิดไฟฟ้า โดยมีหลักการโดยใช้ประโยชน์จากการไหลของน้ำจากแหล่งน้ำขนาดเล็กในการขับเคลื่อนกันหัน กำลังการผลิตนั้นแปรผันโดยตรงกับอัตราการไกลของน้ำ ความหนาแน่นของน้ำ และระดับความสูงของหัวน้ำ

Storage Device

เป็นอุปกรณ์ในการเก็บสะสมพลังงานสำหรับจ่ายพลังงานฉุกเฉินในไมโคร กริด ซึ่งสามารถจ่ายพลังงานในช่วงเวลาสั้นๆ อุปกรณ์ในการเก็บพลังงานมีดังนี้

- Storage Battery
- Flywheels
- Ultra Capacitor

2.1.9 การจัดการและปัญหาการดำเนินงานของไมโครกริด

การส่งเสริมไมโครกริดให้ดีขึ้นนั้นนอกจากจะส่งผลดีต่อการจ่ายกำลังไฟฟ้าอย่างมีเสถียรภาพแล้วยังมีประโยชน์ในด้านต่างๆ ดังนี้

ด้านสิ่งแวดล้อม

ไม่จำเป็นต้องพูดเลยว่าไมโครกริดนั้นสร้างผลกระทบให้กับสิ่งแวดล้อมน้อยกว่าสถานีไฟฟ้าพลังงานความร้อนมากๆแต่อย่างไรก็ตามการอ้างถึงความสำเร็จในการส่งเสริมของการควบคุมคาร์บอนและเก็บโครงการของโรงไฟฟ้าพลังงานความร้อนไว้เพราะจะช่วยลดผลกระทบต่อสิ่งแวดล้อมได้อย่างมาก แต่กระนั้นผลประโยชน์บางอย่างของไมโครกริดก็ควรจะมีดังนี้

- การลดการใช้เชื้อเพลิงแก๊สให้น้อยที่สุดและลดการแพร่ของ CO2 ให้น้อยที่สุด ลดการกำหนดใช้ในวงจำกัดโดยควรจะเป็นกระบวนการเผาไหม้ในขั้นสุดท้ายเพื่อช่วยระบบทั้งหมด เมื่อแหล่งจ่ายงานอื่นทำงานได้ไม่เพียงพอ

- การใช้ไมโครกริดและแหล่งผลิตพลังงานย่อยควรช่วยกันเพิ่มความตระหนัก ของการใช้พลังงานไปในทางที่เหมาะสม

ด้านการใช้งานและการเงิน

ไมโครกริดนั้นทำให้มีลดระยะทางระหว่างแหล่งผลิตพลังงานย่อยกับโหลดได้ดังนี้

- ช่วยส่งเสริมสนับสนุนระบบทั้งหมดด้วยการเพิ่มระดับของแรงดันไฟฟ้าระบบ
- การลดความแออัดของระยะสายป้อนในการส่งและจำหน่ายกำลังไฟฟ้า
- การลดค่าความสูญเสียทางกำลังไฟฟ้าในการส่งและการจำหน่ายกำลังไฟฟ้า
- ช่วยลดการลงทุนสร้างการขยายของระบบส่งจ่ายกำลังไฟฟ้าและระบบผลิตพลังงานไฟฟ้า โดยจัดการใช้ทรัพย์สินในวงจำกัด

ด้านคุณภาพของกำลังไฟฟ้า

ส่งเสริมให้ดีขึ้นในคุณภาพของกำลังไฟฟ้าและความมั่งคงของระบบได้ ได้ดังนี้

- การจ่ายพลังงานไฟฟ้าจากศูนย์กลางไปยังพื้นที่ต่างๆ
- มีความสอดคล้องกันอย่างดีระหว่างการจ่ายพลังงานไฟฟ้าพอเพียงพอต่อความต้องการใช้พลังงานไฟฟ้า
- การลดผลกระทบของ ค่ากำลังไฟฟ้า, แรงดันไฟฟ้า, ความถี่ในการใช้งานได้ เมื่อระบบส่งจ่ายและผลิตกำลังไฟฟ้าชองส่วนกลางขาดหายไป
- ลดเวลาของการไม่ทำงานของเครื่องจักรกลไฟฟ้าให้น้อยที่สุด และ เพิ่มการซ่อมแซมให้สำเร็จตลอดการกลับมาเริ่มต้นใช้งานของแหล่งผลิตพลังงานไฟฟ้าย่อย

การประหยัดค่าใช้จ่าย

- การประหยัดค่าใช้จ่ายต่อไปนี้จะประสบความสำเร็จใน ไมโครกริด: การประหยัดอย่างมีนัยสำคัญมาจากการใช้ประโยชน์จากความร้อนเหลือทิ้งในโหมดของการทำงานของ CHP นอกจากนั้นที่แหล่งจ่าย CHP ยังตั้งอยู่ใกล้กับโหลดผู้ใช้งานที่ไม่มีโครงสร้างพื้นฐานที่สำคัญเป็นสิ่งจำเป็นสำหรับการส่งผ่านความร้อนนี้ให้ประสิทธิภาพการใช้พลังงานรวมกว่า 80% เมื่อเทียบกับจำนวนสูงสุดที่ 40% สำหรับระบบไฟฟ้าทั่วไป

- การประหยัดค่าใช้จ่ายจะมีผลโดยตลอดการรวมของหลายแหล่งผลิตพลังงานไฟฟ้าย่อยขณะที่พวกเขาจะอยู่ในประเทศในโหมด plug-and-play, ค่าใช้จ่ายที่ส่งจ่ายกำลังไฟฟ้ามีการ

ลดลงอย่างมากหรือตัดออก เมื่อรวมกันเป็นไมโครกริดไฟฟ้าที่ผลิตสามารถใช้ร่วมกันในกลุ่มผู้ใช้งานในประเทศ ซึ่งนอกจากนั้นการลดลงจำเป็นต้องนำเข้า / ส่งออกพลังงานไป / กลับจากกริดหลักที่มีสายป้อนขนาดยาว

ปัญหาการตลาด

ข้อดีต่อไปนี้จะบรรลุในกรณีของตลาดการมีส่วนร่วม:

- การพัฒนาตลาดที่ขับเคลื่อนด้วยขั้นตอนการดำเนินงานของไมโครกริดจะนำไปสู่การลดลงอย่างมีนัยสำคัญจากอำนาจตลาดที่จัดตั้งขึ้นโดยการกระทำของบริษัท ผลิต

- การประยุกต์ใช้อย่างกว้างขวางจาก Plug-and-play modular micro sources อาจนำไปสู่การลดราคาพลังงานในตลาดพลังงาน

- สมดุลทางเศรษฐกิจที่เหมาะสมระหว่างเครือข่ายการลงทุนและการผลิตไฟฟ้า แบบกระจายศูนย์การใช้มีแนวโน้มที่จะลดระยะยาวราคาของผู้ใช้งานโดยการผลิตไฟฟ้าประมาณ 10%

2.2 ระบบมัลติเอเจนท์ (Multi-Agent System)

Object Management Group (OMG) ซึ่งเป็นองค์กรสากลที่ก่อตั้งขึ้นในปี ค.ศ. 1989 ประกอบด้วยสมาชิกที่เป็นทั้งตัวแทนจำหน่ายระบบสารสนเทศ นักพัฒนาซอฟท์แวร์และผู้ใช้งาน 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

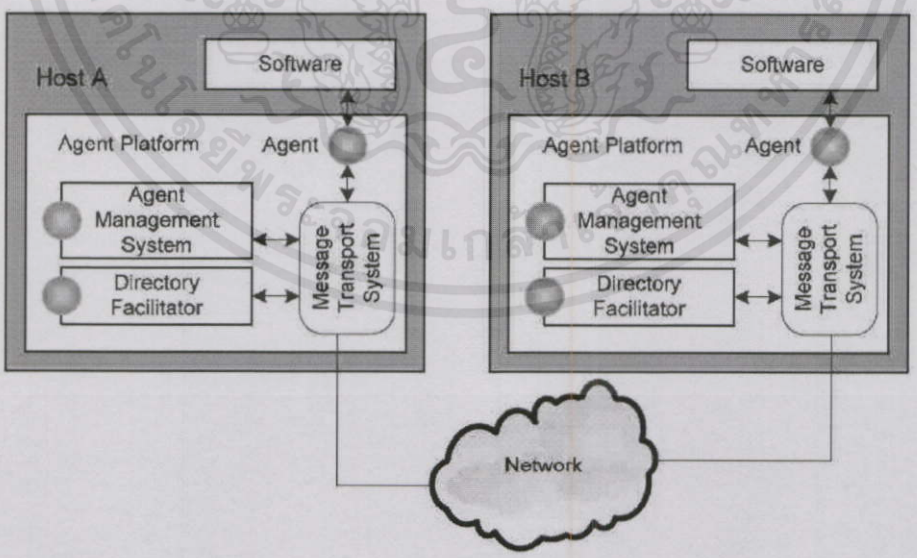
2.2.2 มาตรฐานของระบบเอเจนท์

มาตรฐานของเอเจนท์ที่ใช้อ้างอิงในการทำโครงงานในครั้งนี้ผู้ทำวิจัยได้อ้างอิงตามมาตรฐานของ FIPA ในการศึกษาระบบเอเจนท์

FIPA หรือ The Foundation For Intelligent Physical Agents ก่อตั้งขึ้นในปี ค.ศ. 1996 เพื่อสร้างมาตรฐานซอฟต์แวร์สำหรับระบบเอเจนท์ และในปี ค.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 2.5 ซึ่งแบบจำลองนี้ได้ถูกนำไปพัฒนาเป็นเอเจนท์แพลตฟอร์ม (Agent Platform ; AP) หรือระบบเอเจนท์นั่นเอง



รูปที่ 2.5 แบบจำลองระบบเอเจนท์ตามมาตรฐาน FIPA (FIPA 2002) [1]

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 แต่เอเจนท์อาจจะปฏิเสธการให้บริการได้หลังจากที่ทำการลงทะเบียนไว้แล้วดังนั้น DF จึงไม่สามารถรับรองความเป็นปัจจุบันของข้อมูลที่ลงทะเบียนไว้ นอกจากนี้เอเจนท์อาจทำการถอนการลงทะเบียนที่ DF หรือแก้ไขรายละเอียดเอเจนท์ได้ในภายหลัง

Message Transport System (MTS)

MTS ทำหน้าที่ในการส่งข้อความระหว่างเอเจนท์โดยการรับส่งข้อความระหว่างเอเจนท์จะทำผ่าน MTS เท่านั้น

ดังนั้นเอเจนท์จึงเป็นโปรแกรมคอมพิวเตอร์ที่สามารถทำงานแบบอัตโนมัติและติดต่อสื่อสารกับโปรแกรมประยุกต์เพื่อทำงานตามที่กำหนด เอเจนท์ติดต่อสื่อสารกันโดยใช้ภาษาในการติดต่อสื่อสารของเอเจนท์ (Agent Communication Language หรือ ACL) และอาจมีบริการสำหรับให้เอเจนท์อื่นสามารถเรียกใช้งานได้ นอกจากนี้เอเจนท์ต้องมีเจ้าของเอเจนท์ (Owner) ซึ่งอาจเป็นคนหรือองค์กร และมีชื่อเอเจนท์เพื่อใช้ในการอ้างถึง (Agent Identifier หรือ AID) และติดต่อกับเอเจนท์อื่น

2.2.4 การติดต่อสื่อสารระหว่างเอเจนท์

เอเจนท์จะมีการติดต่อสื่อสารกันด้วยการส่งข้อความเพื่อทำงานร่วมกันหรือทำการเจรจาต่อรองกัน เนื่องจากเอเจนท์อาจสร้างมาจากต่างระบบกันจึงต้องมีการกำหนดมาตรฐานโครงสร้างข้อความระหว่างเอเจนท์ ซึ่งเกี่ยวข้องกับโดเมนที่เอเจนท์นั้นทำงานเพื่อให้ทุกเอเจนท์เข้าใจตรงกัน และแม้ว่าจะไม่เข้าใจ

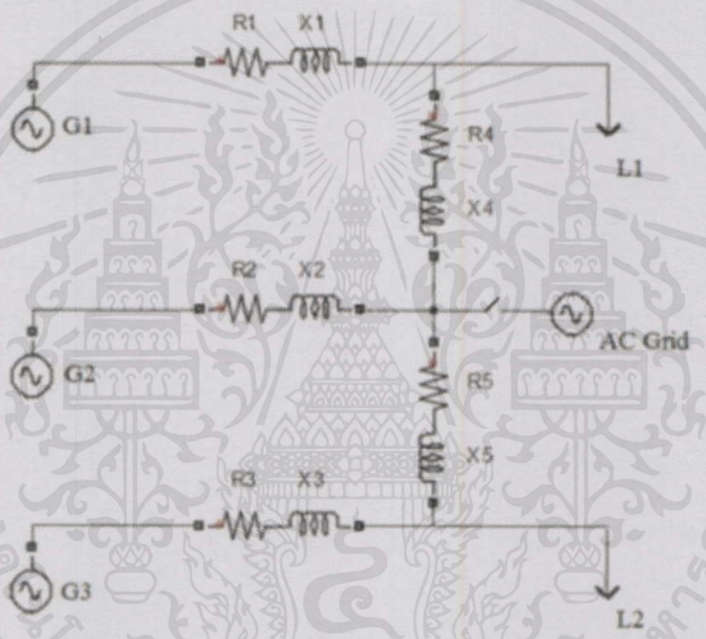
เนื้อหาแต่ก็ควรเข้าใจโครงสร้างของข้อความที่เป็นมาตรฐาน โครงสร้างนี้เรียกว่า Agent Communication Languages

บทที่ 3
การออกแบบวงจรควบคุมระบบผลิตไฟฟ้าสำหรับไมโครกริดและการใช้งานเอเจนต์

3.1 การออกแบบวงจรควบคุมสำหรับไมโครกริดระบบ 1 เฟส

3.1.1 ค่าพารามิเตอร์และวงจรสมมูล

โดยค่า R1,X1 เป็นค่าอิมพีแดนซ์สายของบัส 1 R2,X2 เป็นค่าอิมพีแดนซ์สายของบัส2 และ R3,X3 เป็นค่าอิมพีแดนซ์สายของบัส3 ซึ่งทั้งสามวงจรจะไปเชื่อมต่อกันที่จุดต่อร่วม (PCC) ซึ่งมีค่าอิมพีแดนซ์คือ R4, X4, R5, และ X5 ดังรูป



รูปที่ 3.1 วงจรสมมูลเฟสเดียวของระบบที่ใช้ในการทดลอง

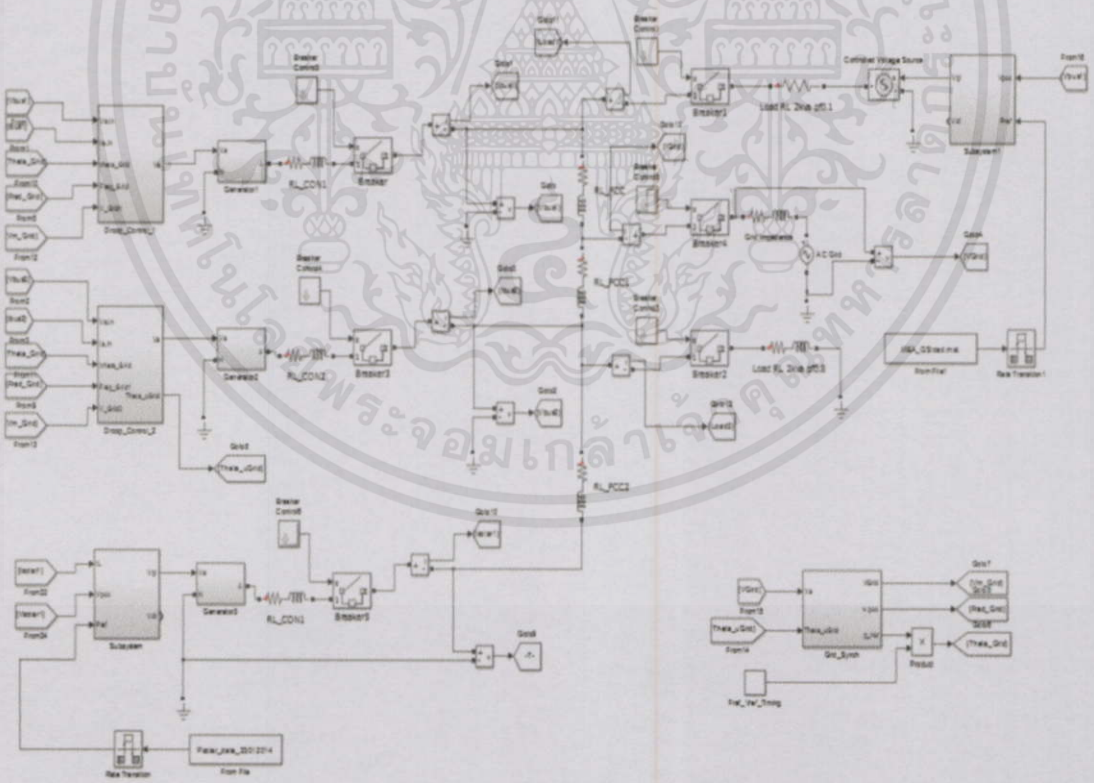
ตารางที่ 3.1 ค่าพารามิเตอร์ของวงจรสมมูลที่ใช้ในการทดลอง

อุปกรณ์	ค่าพารามิเตอร์
G1	Power rating 2kVA , Designed for p.f. = 0.8
G2	Power rating 2kVA , Designed for p.f. = 0.8
G3	Power rating 3kVA , Designed for p.f. = 1
R1	0.1446Ω
X1	4.633e-3 H
R2	0.1446Ω
X2	4.633e-3 H

R3	0.25Ω
X3	100e-6 H
R4	0.25Ω
X4	100e-6 H
R5	0.25Ω
X5	100e-6 H
L1	48.4Ω
L2 (R+jXl)	Case load RL at P.F.=0.8 Z = 19.36+14.52jΩ

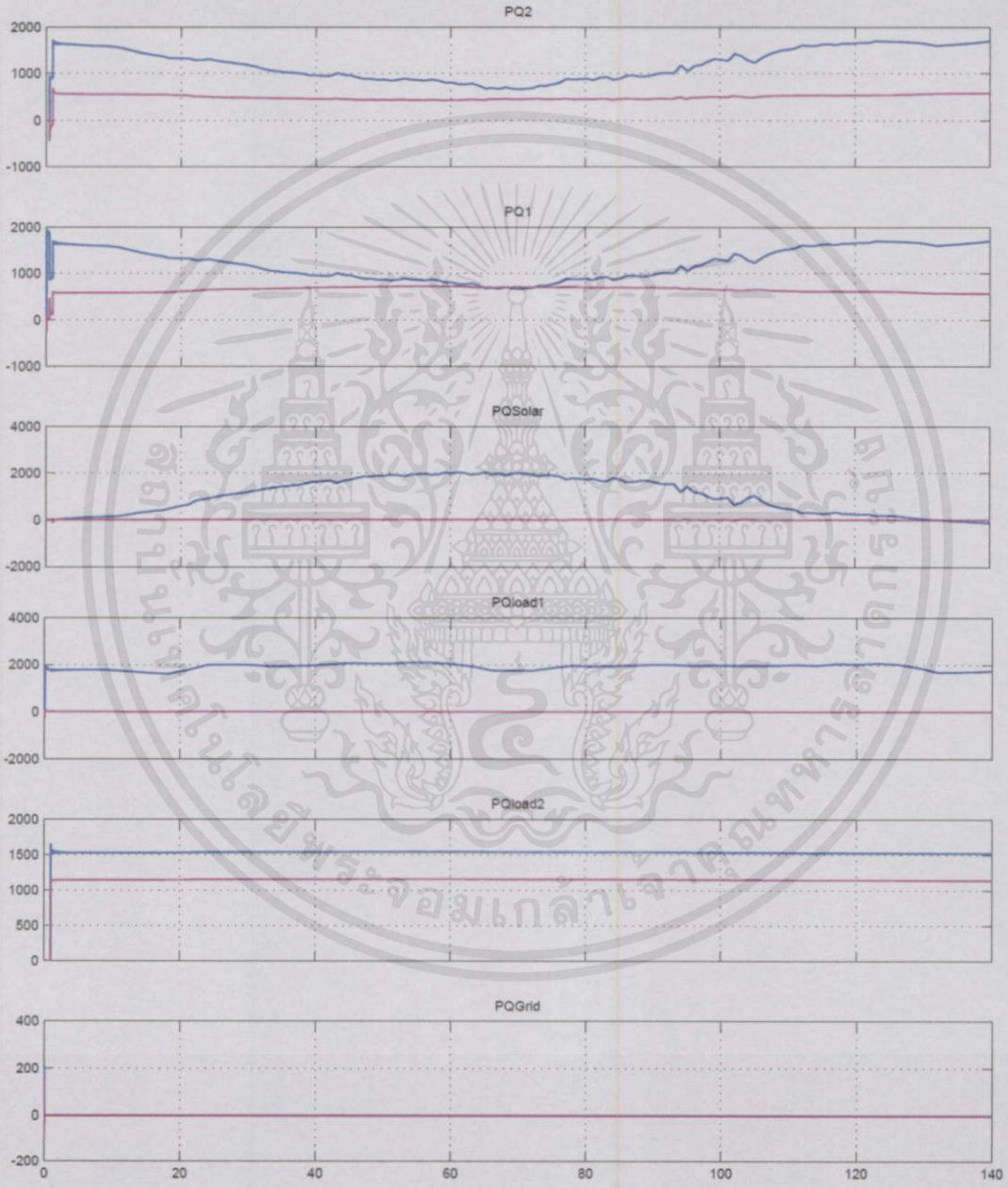
3.1.2 การออกแบบแบบจำลองไมโครกริดในโหมด Stand-alone และโหมด Grid connected ของระบบ 1 เฟส

ระบบไมโครกริด (Microgrid System) จะมี Converter 3 ตัวคือ Converter1, Converter2 และ Converter3 ในการทำหน้าที่ช่วยกันจ่ายโหลดตาม Droop Coefficient ตามรูปที่ 3.2

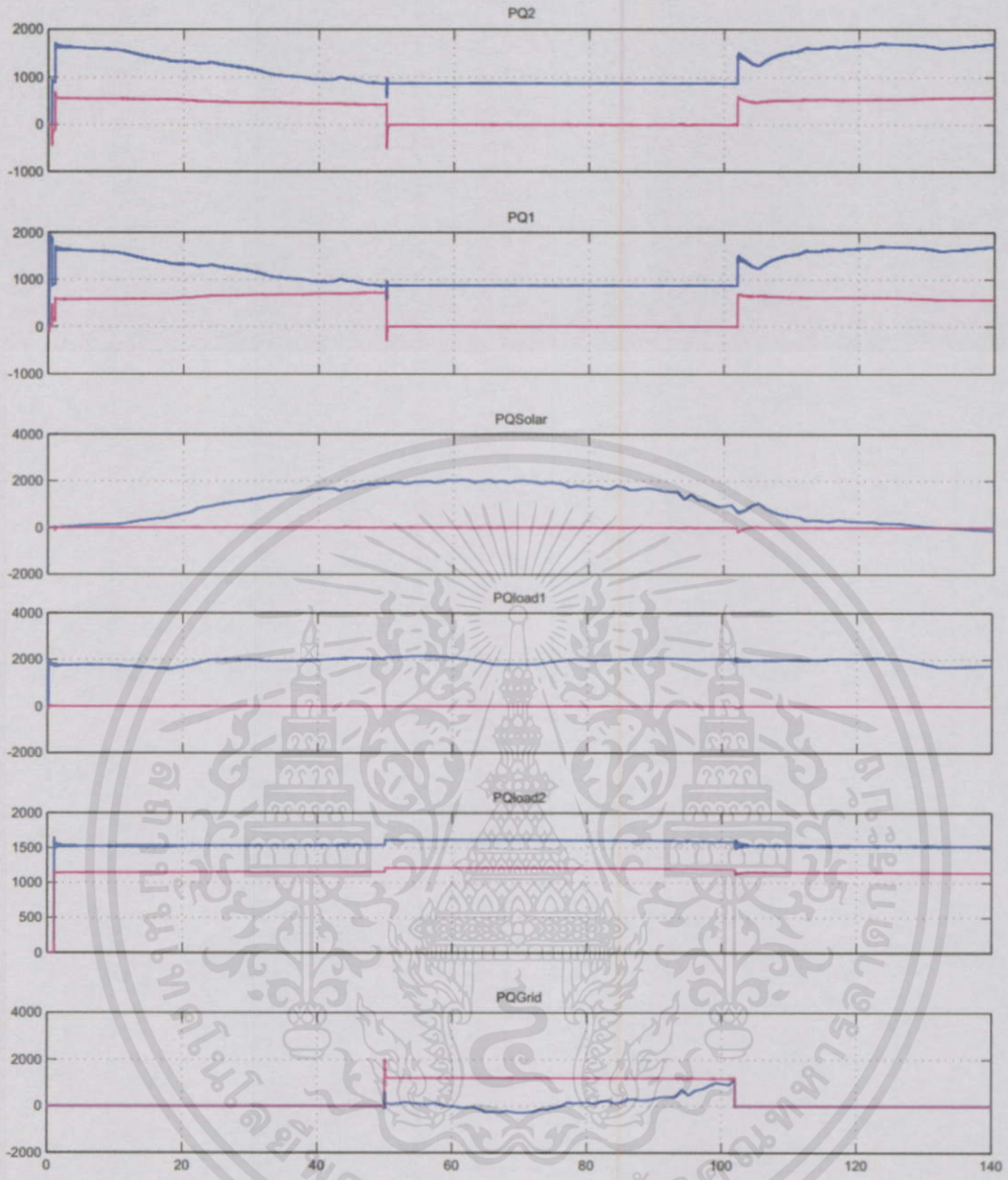


รูปที่ 3.2 แบบจำลองไมโครกริดในโหมด Stand-alone และ Grid-Connected ของระบบ 1 เฟส

และจากโมเดลดังกล่าวรวมทั้งค่าพารามิเตอร์ในตารางที่ 3.1 นำไปทำการจำลองผ่านโปรแกรม Matlab ในช่วงเวลา 1 วัน โดยในการจำลองนี้เรากำหนดให้ 1 วินาทีของการจำลองเท่ากับ 5 นาทีในเวลาจริงจะได้ใน 1 วันของเวลาจริงจะใช้เวลาในการจำลองเท่ากับ 140s ทำให้ได้กราฟของ P,Q ของเครื่องกำเนิดไฟฟ้าเครื่องที่ 1 2 3 และกริดหลักตามลำดับ และในการจำลองจะมีการแบ่งการจำลองออกเป็น 2 โหมด คือ โหมด Stand-alone ในรูปที่ 3.3 และ โหมด Grid-connect ในรูปที่ 3.4



รูปที่ 3.3 กราฟแสดงความสัมพันธ์ PและQ ในโหมด Stand alone

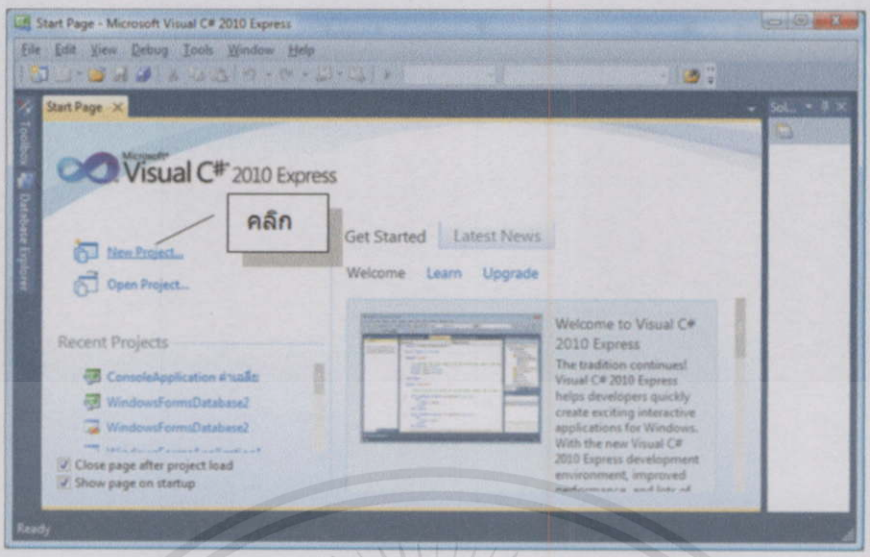


รูปที่ 3.4 กราฟแสดงความสัมพันธ์ PและQ ในโหมด Grid-connect

3.2 แนะนำโปรแกรม Microsoft visual c# 2010 เบื้องต้น

3.2.1 วิธีการเรียกใช้โปรแกรม Visual C#

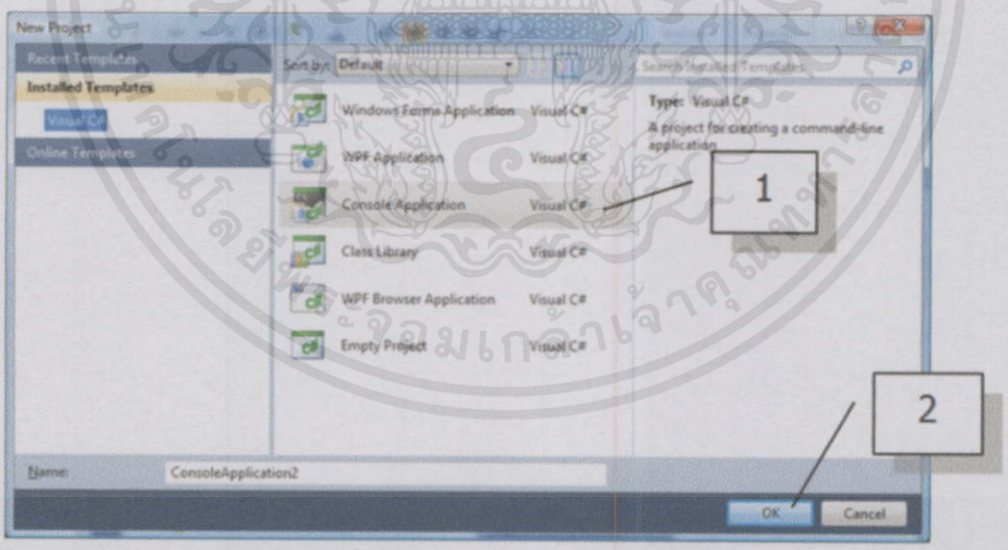
เมื่อเข้าสู่โปรแกรม C# จะพบหน้าต่าง Start Page ดังรูปต่อไปนี้



รูปที่3.5 Start Page ของ Visual C#

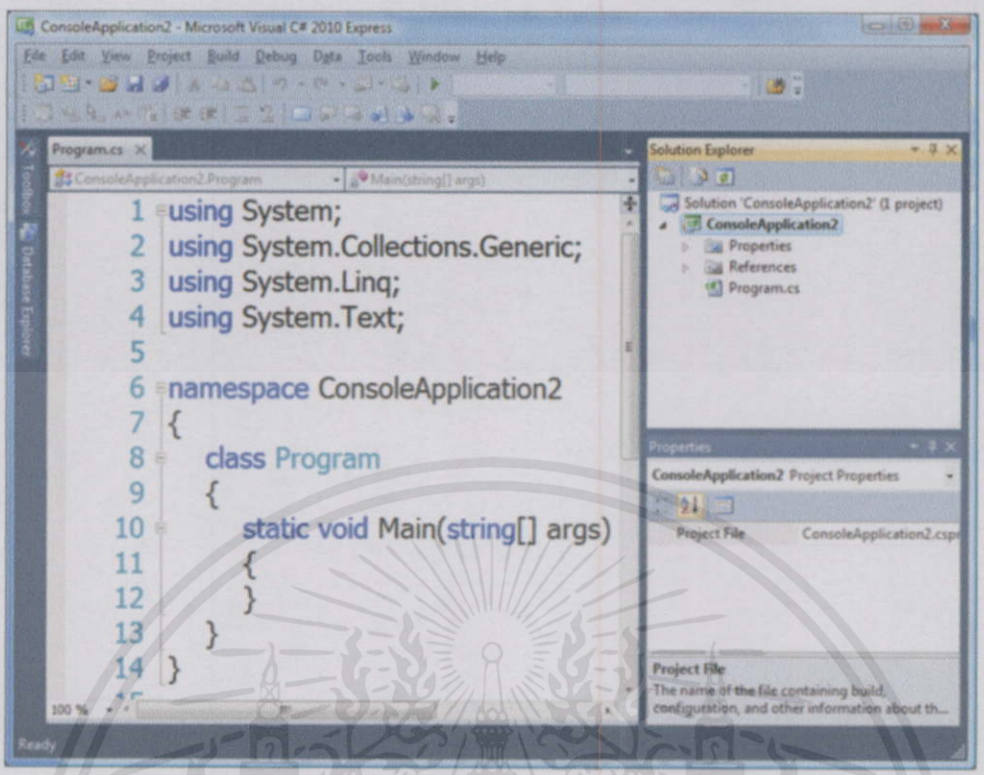
การเขียนโปรแกรมจะต้องเริ่มต้นจากการสร้างโปรเจ็ค (Project) เสียก่อน โดยการคลิกที่ลิงค์ New Project ดังรูป หรือคลิกที่เมนู File แล้วเลือก New Project ก็ได้

จากนั้นโปรแกรมจะเปิดหน้าต่าง New Project ขึ้นมาให้เลือกรูปแบบของโปรแกรมที่ต้องการ เลือก Console Application แล้วคลิกตกลง ดังรูป



รูปที่ 3.6 การสร้างหน้าต่าง Console Application

จากนั้นจะได้ Project และไฟล์ที่ใช้ในการเขียนรหัสคำสั่งชื่อ Program.cs โดยไฟล์นี้โปรแกรม C# จะสร้างรหัสโปรแกรมมาให้ส่วนหนึ่ง เพื่อช่วยให้เราเขียนโปรแกรมได้เร็วขึ้น



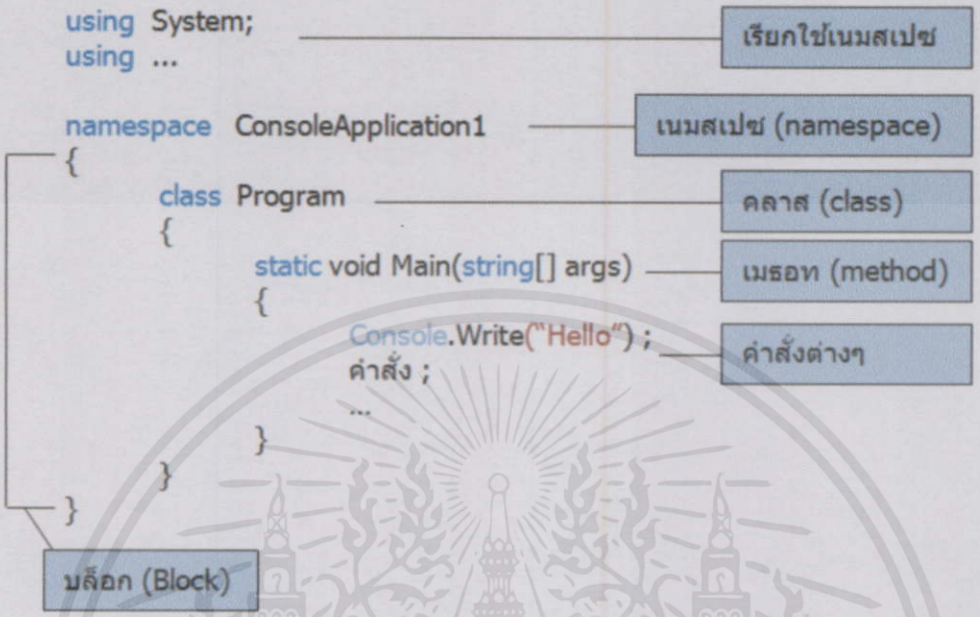
รูปที่ 3.7 ตัวอย่างการเขียนโปรแกรม

เมื่อพิมพ์คำสั่งเสร็จแล้ว กด F5 บนแป้นพิมพ์ เพื่อดูผลลัพธ์ของโปรแกรม ซึ่งต่อไปเราจะเรียกการเรียกดูผลลัพธ์ว่า การรันโปรแกรม (ใน C# เรียกว่า Debugging) ผลลัพธ์ที่ได้หลังจากการรันก็คือ

Hello World

3.2.2 โครงสร้างคำสั่งโปรแกรมภาษา C#

โปรแกรมภาษา C# มีรูปแบบโครงสร้างคำสั่งดังนี้



รูปที่ 3.8 โครงสร้างคำสั่งโปรแกรมภาษา C#

3.2.3 รูปแบบคำสั่งของภาษา C#

โปรแกรมภาษา C# เป็นโปรแกรมภาษารุ่นล่าสุดของบริษัทไมโครซอฟต์ พัฒนามาจากภาษา C++ และ Java จุดประสงค์เพื่อให้สามารถเขียนโปรแกรมได้อย่างมีประสิทธิภาพ จึงทำให้ C# สามารถเขียนโปรแกรมได้หลายรูปแบบ มีคำสั่งต่างๆ มากมายที่ให้เราเรียกใช้ และมีคำศัพท์ที่เกี่ยวข้องมาก เช่น namespace, class, method, property, statement เป็นต้น บางครั้งจึงดูเหมือนยุ่งยากและซับซ้อนในการเขียนโปรแกรม ดังนั้นการเขียนโปรแกรมเบื้องต้นเราจะเรียนรู้เพียงบางส่วนเท่านั้น ขอกล่าวโดยสรุปเกี่ยวกับสีของคำสั่ง ดังนี้

สีน้ำเงิน หมายถึงคำสั่งที่เป็นคำสงวนของโปรแกรม C# ได้แก่คำว่า string, int, if, else คำเหล่านี้เป็นคำสั่งสำเร็จของ C# ที่เรานำมาใช้เพื่อจุดประสงค์อย่างหนึ่งอย่างใด ซึ่งเราจะไม่สามารถตั้งชื่อตัวแปรซ้ำกับคำเหล่านี้ได้ string กับ int เราใช้สำหรับประกาศตัวแปรให้เป็นชนิดข้อความและตัวเลขจำนวนเต็มตามลำดับ ส่วนคำสั่ง if...else ใช้สำหรับการตรวจสอบเงื่อนไขในการเขียนโปรแกรมแบบทางเลือก ข้อสังเกตก็คือ คำสั่งเหล่านี้จะอยู่แบบเดี่ยวๆ ไม่ต้องมีจุดต่อท้ายเหมือนคำสั่งอื่นๆ และอักษรตัวแรกของคำสั่งจะเป็นตัวพิมพ์เล็กเสมอ

สีฟ้า เป็นการบ่งบอกว่าคำเหล่านี้เป็นชื่อของคลาสที่เรานำมาใช้ในการเขียนโปรแกรม ซึ่งคลาสก็คือที่ที่เก็บเมธอดซึ่งเป็นคำสั่งเพื่อให้โปรแกรมทำงานอย่างหนึ่งอย่างใด การที่เราจะเรียกใช้เมธอดใดก็ตามเราจะต้องระบุชื่อของคลาสเสียก่อนแล้วตามด้วยเครื่องหมายจุดแล้วจึงระบุชื่อของเมธอดอีกที จากโปรแกรมเราจะเรียกใช้คลาส Console ข้อสังเกตก็คือคำสั่งที่เป็นคลาสเหล่านี้มักจะมีจุดต่อท้ายแล้วตามด้วยคำสั่งอื่นๆ อีกเสมอ และอักษรตัวแรกของคำสั่งจะเป็นตัวพิมพ์ใหญ่เสมอ

สีดำ สีนี้จะนำมาใช้กับคำสั่งหลายๆ ชนิด เช่น ชื่อของตัวแปร ได้แก่ name และ size (ชื่อของตัวแปรเราจะเป็นผู้ตั้งชื่อเอง) ชื่อของเมธอด ได้แก่ Write และ ReadKey ชื่อของพร็อพเพอตี้ ได้แก่ Length เครื่องหมายต่างๆ ได้แก่ = > . ( ) และ ; ใช้กับค่าคงที่ที่เป็นตัวเลขได้แก่ 10 ดังนั้นสีดำ จึงถูกนำมาใช้มากที่สุด แสดงว่าสีดำเป็นสีที่ไม่ต้องการเน้นเป็นพิเศษนั่นเอง ข้อสังเกตที่สำคัญก็คือ ถ้าเป็นชื่อของเมธอดจะพิมพ์ต่อเนื่องมาจากคลาสอีกทีโดยมีเครื่องหมายจุดเป็นตัวคั่นหรือแบ่งแยกระหว่างคลาสกับเมธอด นอกจากนั้นเมธอดจะมีวงเล็บต่อท้ายเสมอและอักษรตัวแรกจะขึ้นต้นด้วยอักษรพิมพ์ใหญ่เสมอ

ส่วนพร็อพเพอตี้นั้นอักษรตัวแรกจะขึ้นต้นด้วยตัวพิมพ์ใหญ่เสมอเช่นกัน แต่จะไม่มีวงเล็บต่อท้ายเหมือนเมธอดและพร็อพเพอตี้อาจจะอยู่ต่อท้ายของตัวแปร หรือเมธอดก็ได้

ไม่ว่าจะเป็นคลาส เมธอด หรือพร็อพเพอตี้จะเป็นคำสั่งสำเร็จรูปที่ C# เตรียมไว้ให้เรานำมาใช้ได้เลย ดังนั้นเราจะพิมพ์ชื่อผิดเพี้ยนไม่ได้ โปรแกรมจะแจ้งข้อผิดพลาดทันที เช่น คำสั่ง ReadKey เราจะพิมพ์เป็น readkey หรือ Readkey หรือ readKey ไม่ได้เด็ดขาด

สีแดง เป็นสีที่นำมาใช้ค่าคงที่ที่เป็นข้อความ (string) หรือ ตัวอักขระ (char) ที่อยู่ภายใต้เครื่องหมาย "..." และ '...'เท่านั้น

สีเขียว ใช้กับคำอธิบายโปรแกรม ซึ่งจะไม่มีผลอะไรกับการทำงานของโปรแกรม

3.2.4 องค์ประกอบพื้นฐานที่ควรรู้จัก

มีองค์ประกอบพื้นฐานที่สำคัญบางอย่างตามรูปแบบของ C# ที่เราควรรู้จักมีดังนี้

3.2.4.1 บล็อก { ... }

รูปแบบของภาษา ก็เช่นเดียวกับ c / c ++ คือจะใช้บล็อก { ... } ในการกำหนดจุดเริ่มต้นและจุดสิ้นสุดของการทำงานในแต่ละส่วน ซึ่งภายในบล็อกอาจมีบล็อกย่อยๆ ซ้อนลงไปได้อีก ตามลักษณะของงาน เช่น

```
if (.....)
    {
    for (.....)
      {
          .....
          .....
      }
    .....
    .....
  }
```

ทั้งนี้เครื่องหมาย " { " (open brace) กับ " } '' (close brace) นั้นต้องครบคู่กันพอดี และโดยทั่วไปเรานิยมวาง " { " ไว้ให้ตรงกับคำสั่งเริ่มต้นบล็อก เช่น จากรหัสคำสั่งตัวอย่างที่เราวางไว้ตรงกับคำสั่ง if หรือ for เป็นต้น ทั้งนี้เพื่อให้พิจารณาขอบเขตของบล็อกได้ง่าย แต่ไม่ใช่กฎข้อบังคับแต่อย่างใด เราอาจเลือกวางในแบบที่เราถนัดก็ได้

3.2.4.2 เครื่องหมายสิ้นสุดคำสั่ง (;)

ใน C#เราจะใช้เครื่องหมายเซมิโคลอน (semicolon) " ; '' เป็นตัวแสดงจุดสิ้นสุดของแต่ละคำสั่ง หากเราไม่ใส่เครื่องหมายนี้เพื่อคั่นระหว่างคำสั่ง โปรแกรมจะถือว่าเป็นคำสั่งเดียวกันไปตลอดแม้ว่าจะอยู่คนละบรรทัดก็ตาม เช่น จากตัวอย่างมีคำสั่งทั้งหมด 3 คำสั่ง

```
x = 10;
y = "xxx";
z = x + 10;
```

เมื่อเครื่องหมาย ; เป็นตัวบ่งบอกจุดสิ้นสุดของคำสั่ง จึงสามารถมีคำสั่งอยู่ในบรรทัดเดียวกันมากกว่า 1 คำสั่งก็ได้ เช่น จากตัวอย่างมีคำสั่งทั้งหมด 3 คำสั่ง

```
x = 10; y = "xxx"; z = x + 10;
```

แต่การเขียนรหัสคำสั่งในลักษณะนี้ ไม่ค่อยนิยมทำกัน เนื่องจากจะทำให้เราโปรแกรมได้ยาก รหัสโปรแกรมดูไม่เป็นระเบียบ แต่ก็อาจนำมาใช้ในบางกรณีได้เช่นกัน

บางคำสั่งไม่สามารถทำให้จบภายในบรรทัดเดียวได้ เพราะบางคำสั่งจะมีคำสั่งย่อยๆ อยู่ในภายอีกก็ได้ ดังนั้นคำสั่งเหล่านี้จะมีบล็อกของตนเอง เช่น คำสั่ง if คำสั่ง for คำสั่ง while เป็นต้น เมื่อคำสั่งใดมีบล็อกอยู่แล้วก็ไม่จำเป็นต้องใส่เครื่องหมาย ; ต่อท้ายเครื่องหมายบล็อกอีก แต่คำสั่งภายในต้องมีเครื่องหมาย ; ตามปกติ เช่น

if (เงื่อนไข)

{

คำสั่ง A;

คำสั่ง B;

}

นักเรียนควรจดจำไว้ว่า **จะไม่มี** ; อยู่ทั้งก่อนหน้า “{” และหลัง “}”

3.2.4.3 การเขียนคำอธิบายประกอบแทรกไว้ในรหัสโปรแกรม

คำอธิบาย (Comment) หมายถึง การเขียนข้อความใดๆ ที่ไม่ใช่คำสั่งปะปนกันไปกับ คำสั่งอื่นๆ เพื่ออธิบายเรื่องใดเรื่องหนึ่งเอาไว้เพื่อความเข้าใจในรหัสโปรแกรมตรงส่วนนั้น ดังนั้นเพื่อให้โปรแกรมไม่สับสนว่าส่วนใดเป็นคำสั่ง ส่วนใดเป็นเพียงคำอธิบายไม่เกี่ยวข้องกับการทำงานของโปรแกรม จึงต้องใช้เครื่องหมายมาเป็นตัวช่วยในการแยกแยะ การแทรกคำอธิบายสามารถทำได้ 2 ลักษณะดังนี้คือ

รูปแบบ // คำอธิบาย และ

รูปแบบ /* คำอธิบาย */

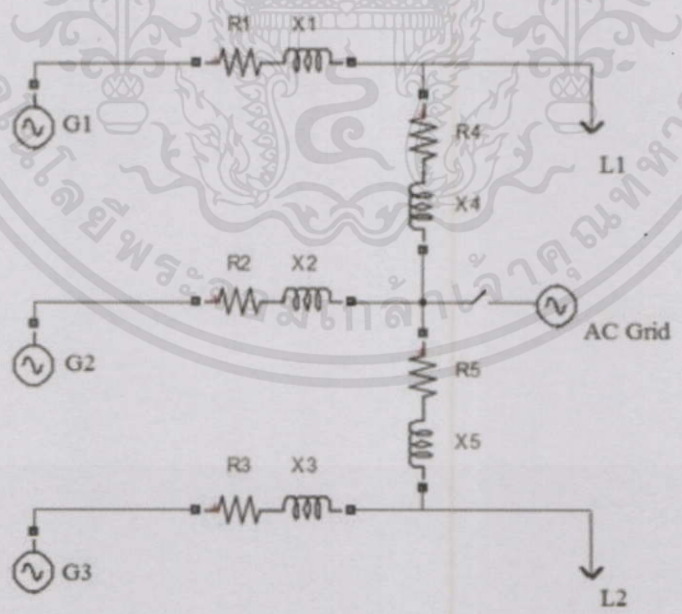
บทที่ 4

การทดลองและผลการทดลอง

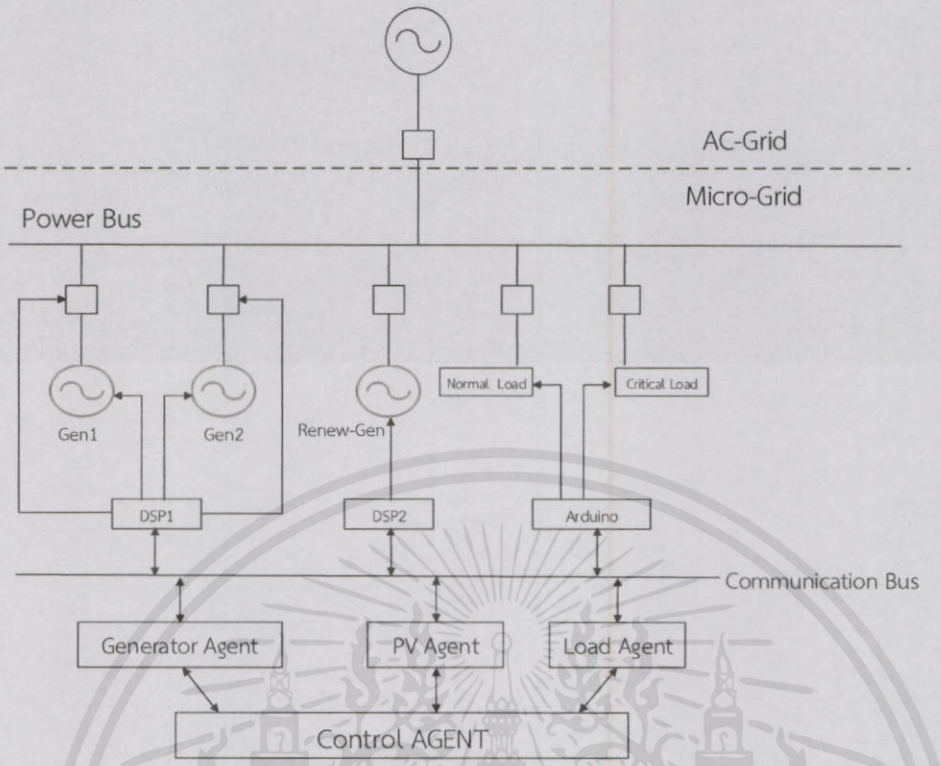
ในบทที่ 4 นี้ จะกล่าวถึงวิธีการจำลองระบบไมโครกริดโดยมีเอเจนท์ควบคุมและตัดสินใจ ซึ่งแต่ละเอเจนท์มีหน้าที่แตกต่างกันซึ่งในบทนี้ได้อธิบายกระบวนการตัดสินใจของเอเจนท์เพื่อสั่งการให้แหล่งกำเนิดไฟฟ้านั้นตอบสนองภาระทางไฟฟ้าที่มีการเปลี่ยนแปลงในแต่ละช่วงเวลาโดยการจัดสรรกำลังไฟฟ้าได้อย่างเหมาะสมคอยรักษาระดับแรงดันด้านโหลดและความถี่ของระบบให้คงที่ อีกทั้งยังสามารถรักษาเสถียรภาพของระบบเมื่อเกิดการขาดหายไปของกำลังการผลิตได้อีกด้วย และในส่วนท้ายได้แสดงผลการจำลองระบบเพื่อดูพฤติกรรมของเอเจนท์ในการเข้ามาควบคุมระบบไมโครกริดในเหตุการณ์ต่างๆด้วย

4.1 แบบจำลองระบบไมโครกริดและเอเจนท์

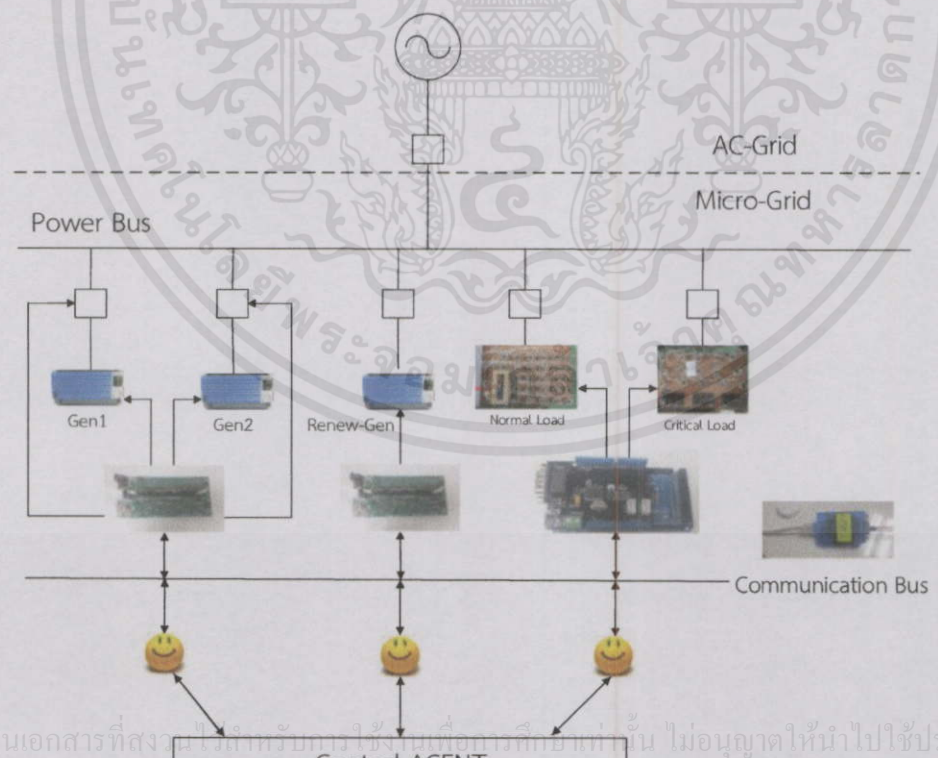
ระบบไมโครกริดที่จำลองเป็นระบบแบบแยกอิสระ(Stand-alone) และระบบแบบต่อกริดหลัก(Grid-connected) แรงดันต่ำ 380V ที่เป็นลักษณะแบบ Network ผ่าน CANUSB ดังรูปที่ 4.1 และลักษณะของการเชื่อมต่อต่างของการทดลองดังรูปที่ 4.2 และ รูปที่ 4.3



รูปที่ 4.1 แบบจำลองไมโครกริด



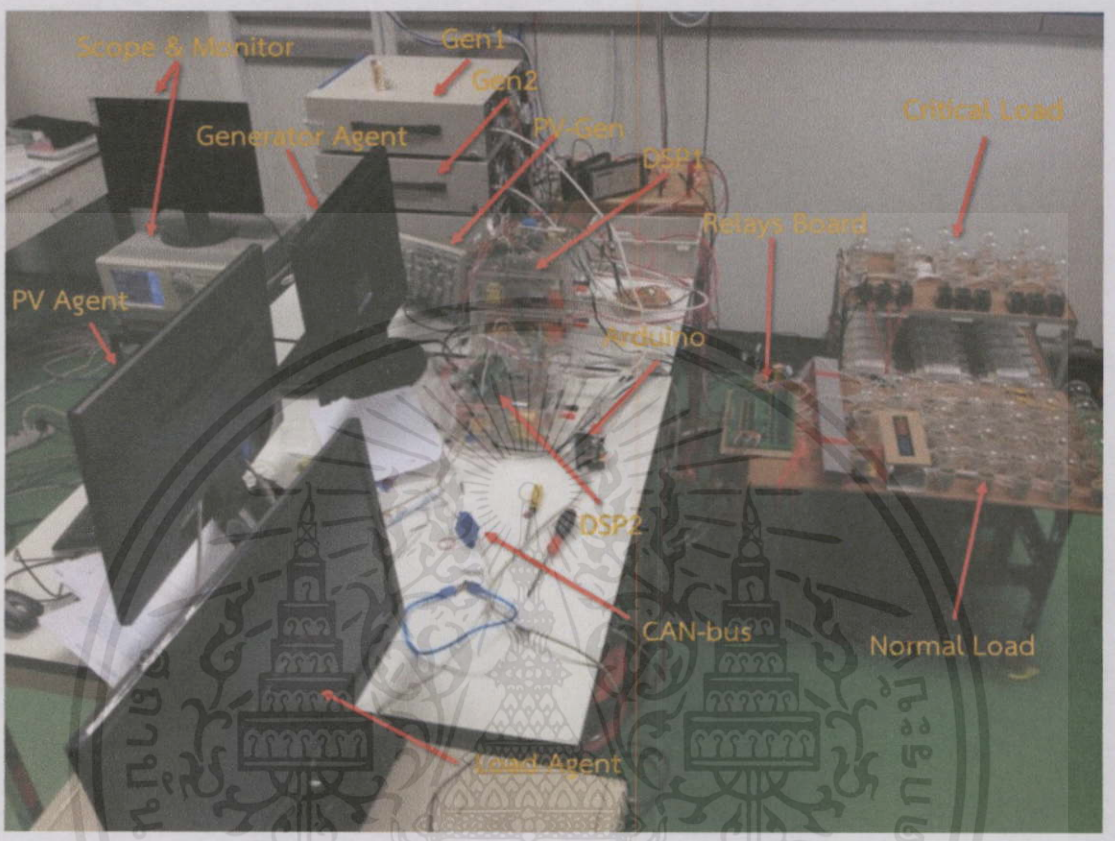
รูปที่ 4.2 การทำงานของระบบไมโครกริด



รูปที่ 4.3 ภาพรวมของการทดลอง

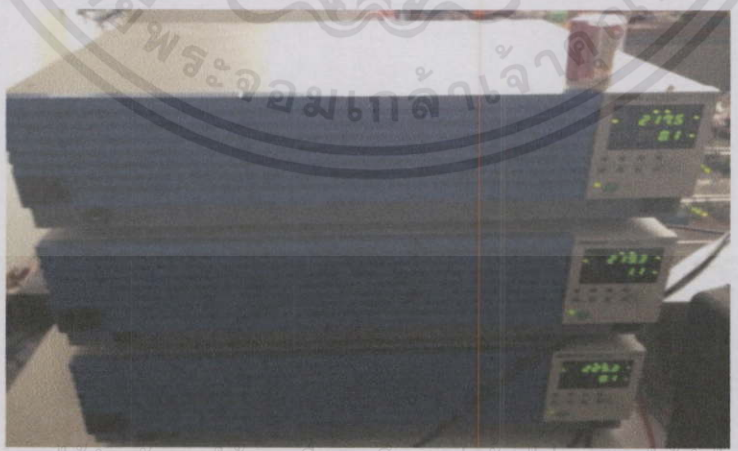
4.2 อุปกรณ์ที่ใช้ในโครงงาน

อุปกรณ์ที่ใช้ในโครงงานมีดังนี้



รูปที่ 4.4 ภาพรวมของอุปกรณ์ที่ใช้ทำการทดลอง

- แหล่งกำเนิดไฟฟ้าพลังงานทางเลือก และแหล่งกำเนิดกำลังไฟฟ้า 1,2 ทำหน้าที่จ่ายกำลังไฟฟ้าให้กับโหลด ดังรูปที่ 4.5

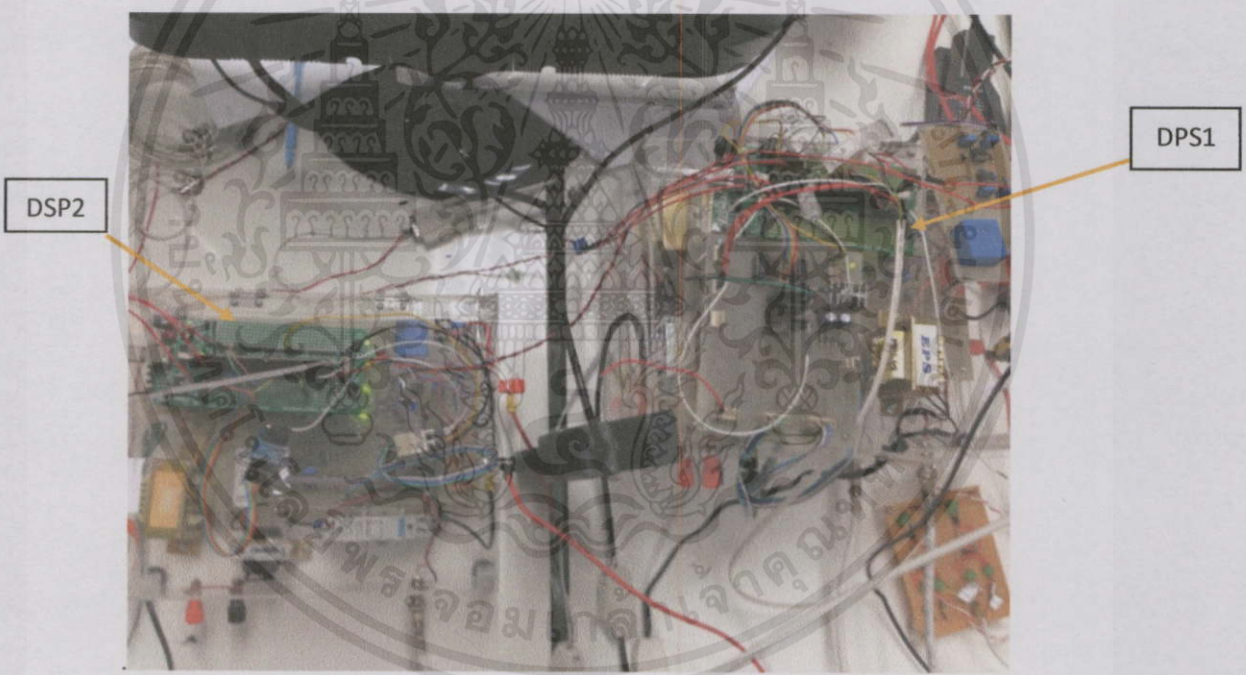


รูปที่ 4.5 แหล่งกำเนิดไฟฟ้าพลังงานทางเลือก และแหล่งกำเนิดกำลังไฟฟ้า 1,2 ตามลำดับ

ตารางที่ 4.1 ข้อมูลของเครื่องกำเนิดไฟฟ้าในไมโครกริด

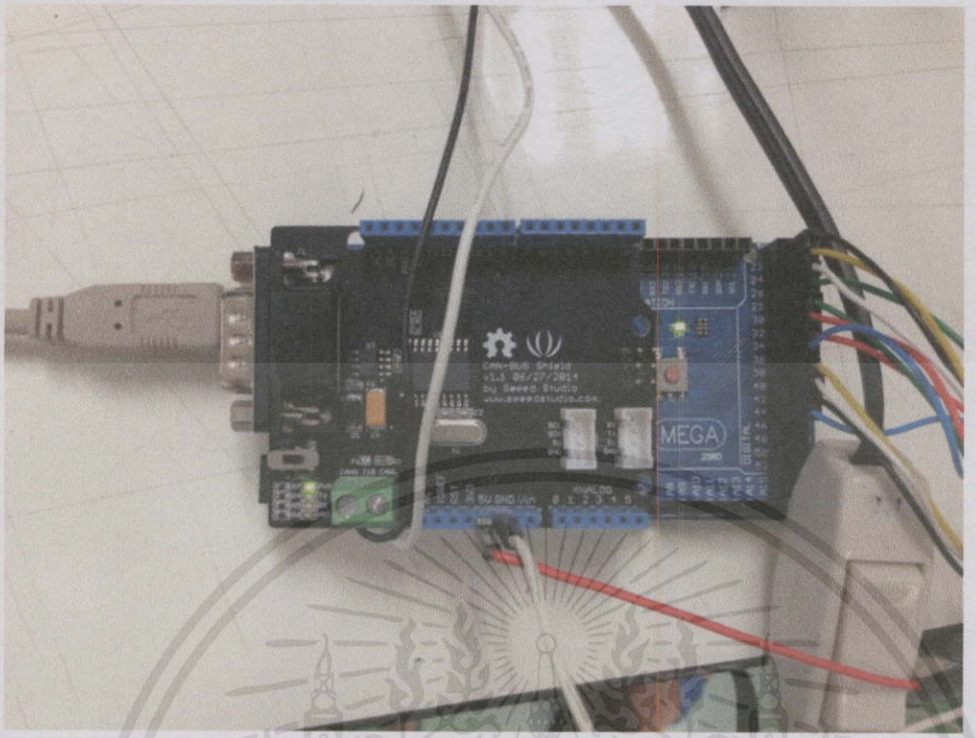
Generator Type	Capacity	จำนวน
	KVA	(เครื่อง)
Generator 1	2	1
Generator 2	2	1
Solar Photovoltaic (PV)	2	1

- DSP (Digital signal processor) ในโครงงานนี้จะใช้ DSP จำนวน 2 ตัว ซึ่งตัวที่ 1 จะเป็นตัวควบคุมแหล่งกำเนิดกำลังไฟฟ้าทั้ง 2 ตัว ให้สามารถจ่ายโหลด และควบคุมการ Droop-control ของระบบไมโครกริดให้ แหล่งกำเนิดกำลังไฟฟ้าทั้ง 2 ตัวที่ขนานกันสามารถทำงานร่วมกันได้ และ DSP2 ทำหน้าที่ควบคุมแหล่งกำเนิดไฟฟ้าพลังงานทางเลือก ดังรูปที่ 4.6



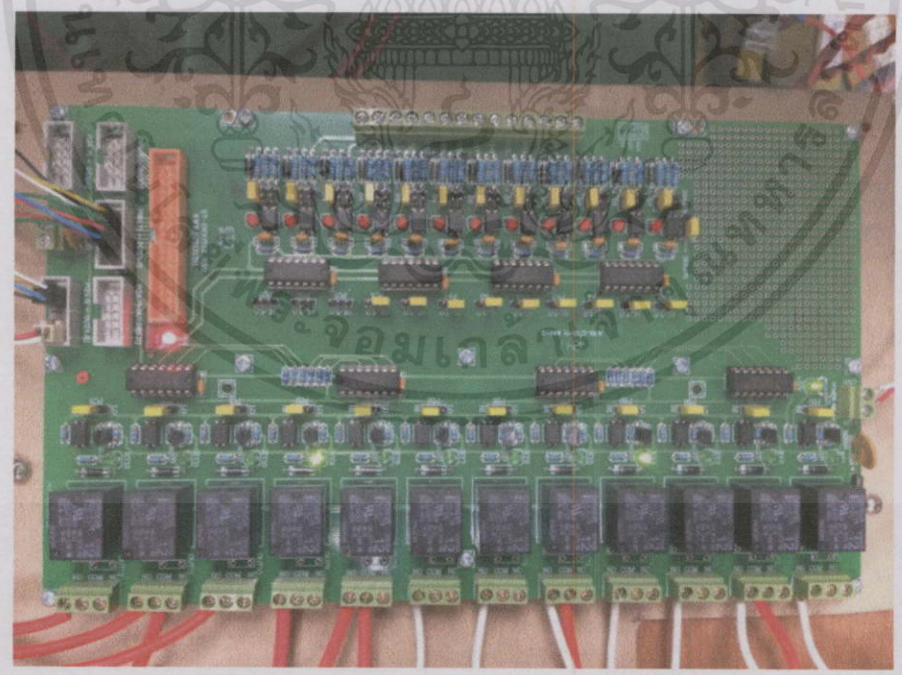
รูปที่ 4.6 DSP (Digital signal processor)

- Arduino ทำหน้าที่ควบคุมรีเลย์บอร์ดโดยจะรับคำสั่งมาจาก Load Agent ในการเปิดปิดโหลด โดยตัว Arduino เองก็ต้องได้รับการลงโปรแกรมเพื่อใช้ในการทำงานด้วย ดังรูปที่ 4.7



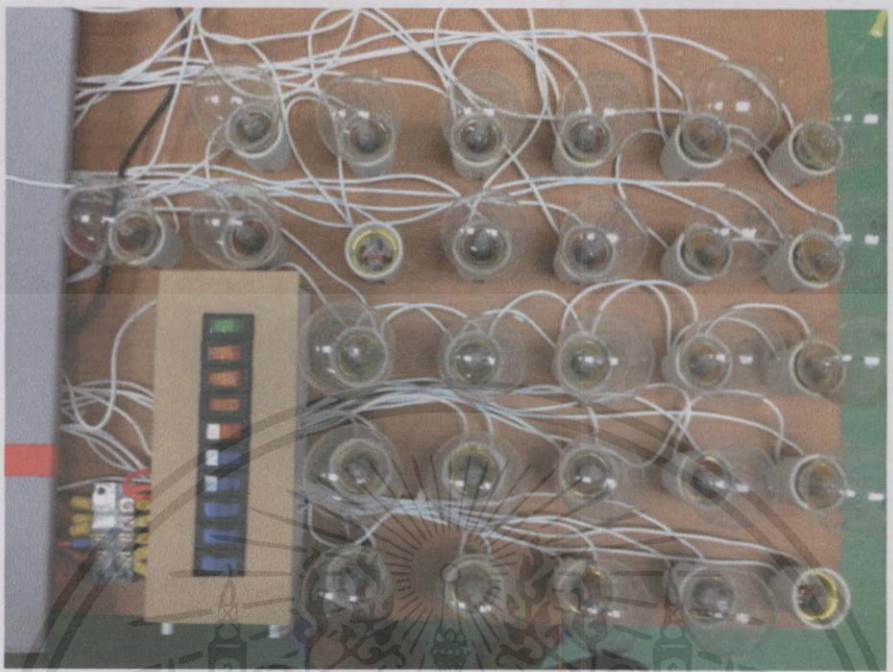
รูปที่ 4.7 Arduino

- รีเลย์บอร์ด ทำหน้าที่ในการควบคุมการเปิด-ปิดโหลด โดยจะรับคำสั่งมาจาก Aduino อีกที ดังรูปที่ 4.8



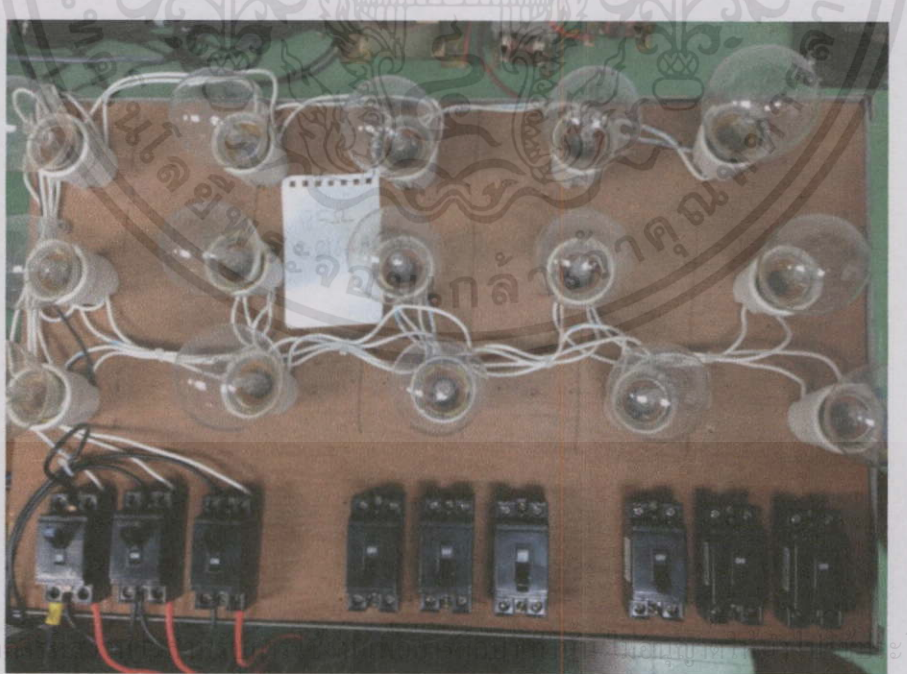
รูปที่ 4.8 รีเลย์บอร์ด

- โหลดความต้านทาน(R) โดยจะรับคำสั่งมาจาก Arduino ดังรูป 4.9



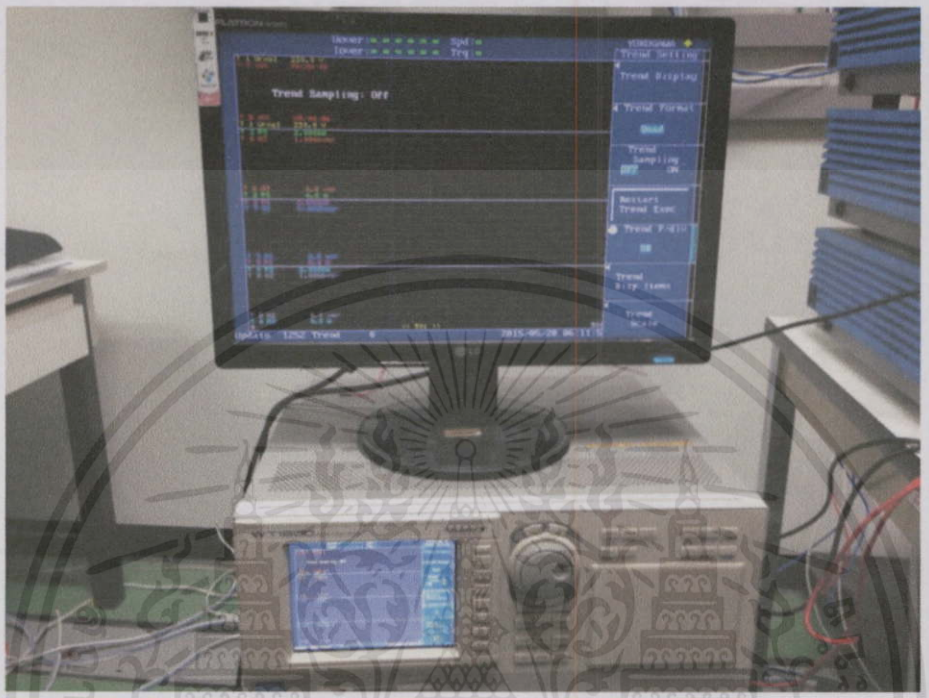
รูปที่ 4.9 โหลดความต้านทาน(R)

- โหลดความต้านทานและความเหนี่ยวนำ(R-L) โดยจะรับคำสั่งมาจาก Arduino ดังรูปที่ 4.10



รูปที่ 4.10 โหลดความต้านทานและความเหนี่ยวนำ(R-L)

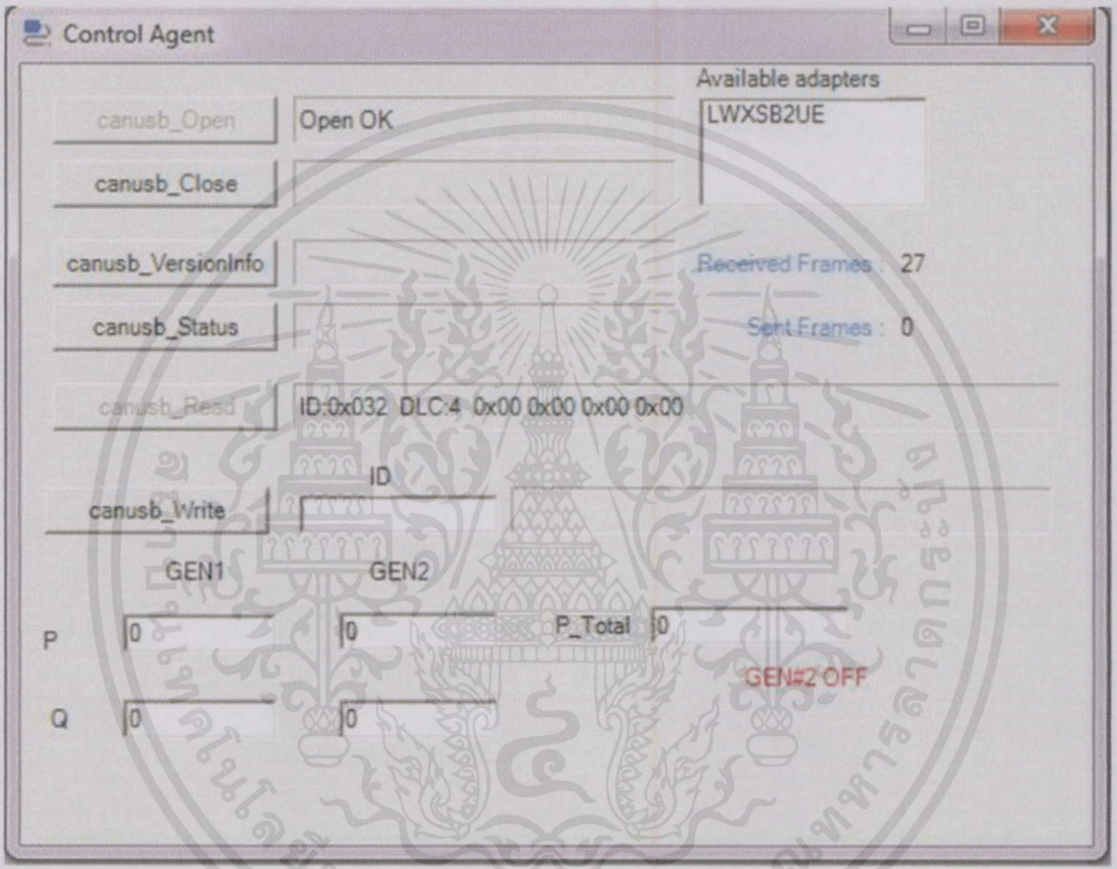
- Scope และ จอแสดงผลจาก Scope จะแสดงรูปกราฟของผลลัพธ์ต่างๆ ของการทดลอง ดังรูปที่ 4.11



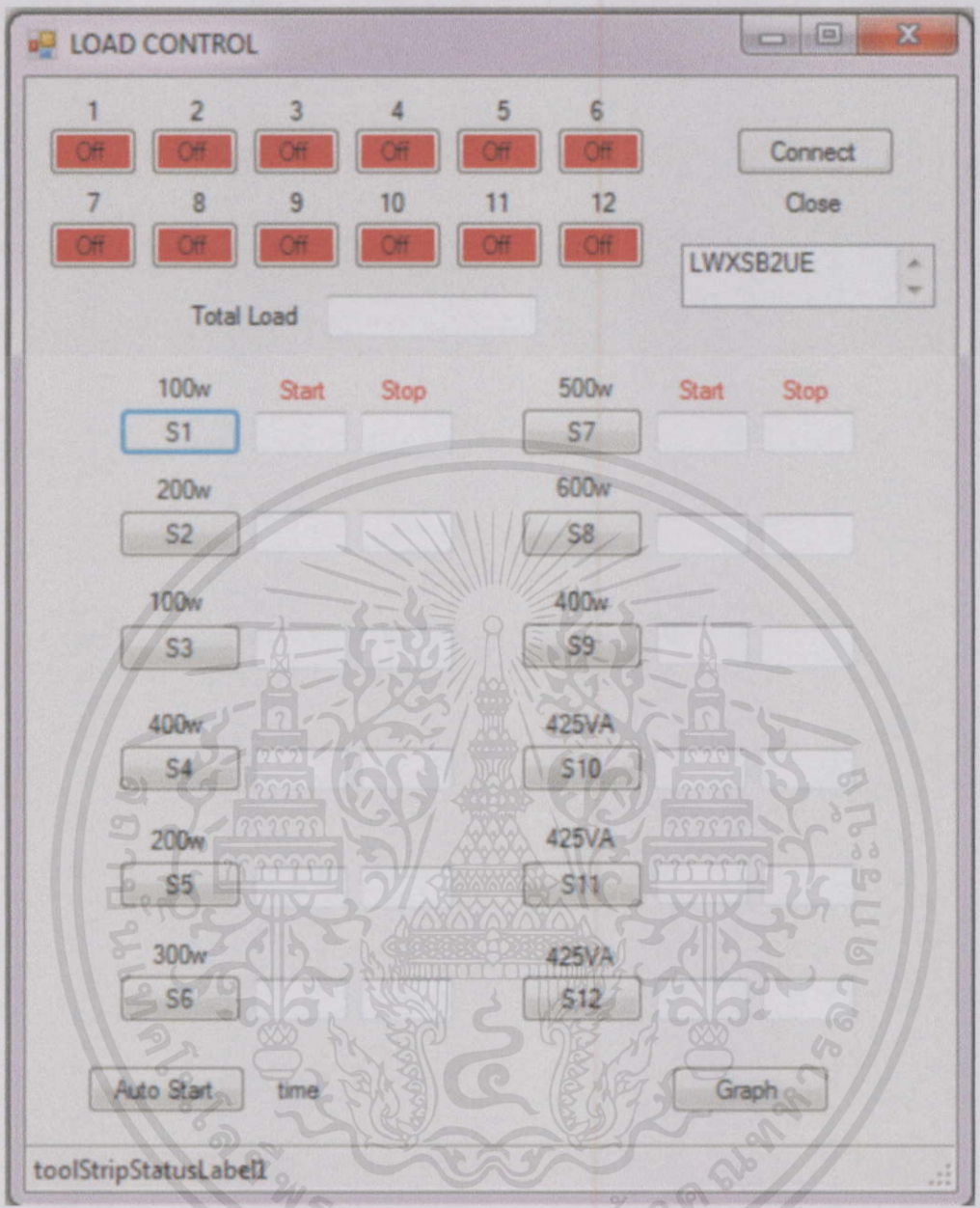
รูปที่ 4.11 Scope และ จอแสดงผลจาก Scope

4.3 Software ที่ใช้ในการทดลอง

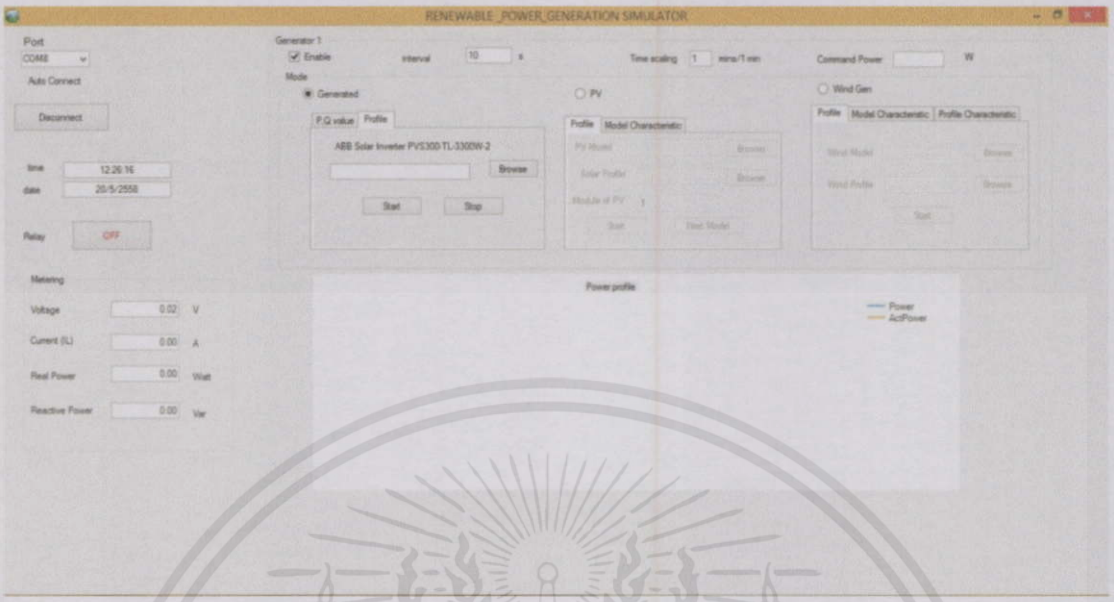
Software ต่างๆที่ใช้ในการทดลองนั้นจะทำการเขียนโปรแกรมด้วยภาษา C# ผ่านโปรแกรม Microsoft Visual Studio (Version 2013) ซึ่งวิธีการเขียนโปรแกรมและโค้ดของโปรแกรมนั้นจะแสดงไว้ในภาคผนวก



รูปที่ 4.12 Agent Control ที่สร้างโดย C#



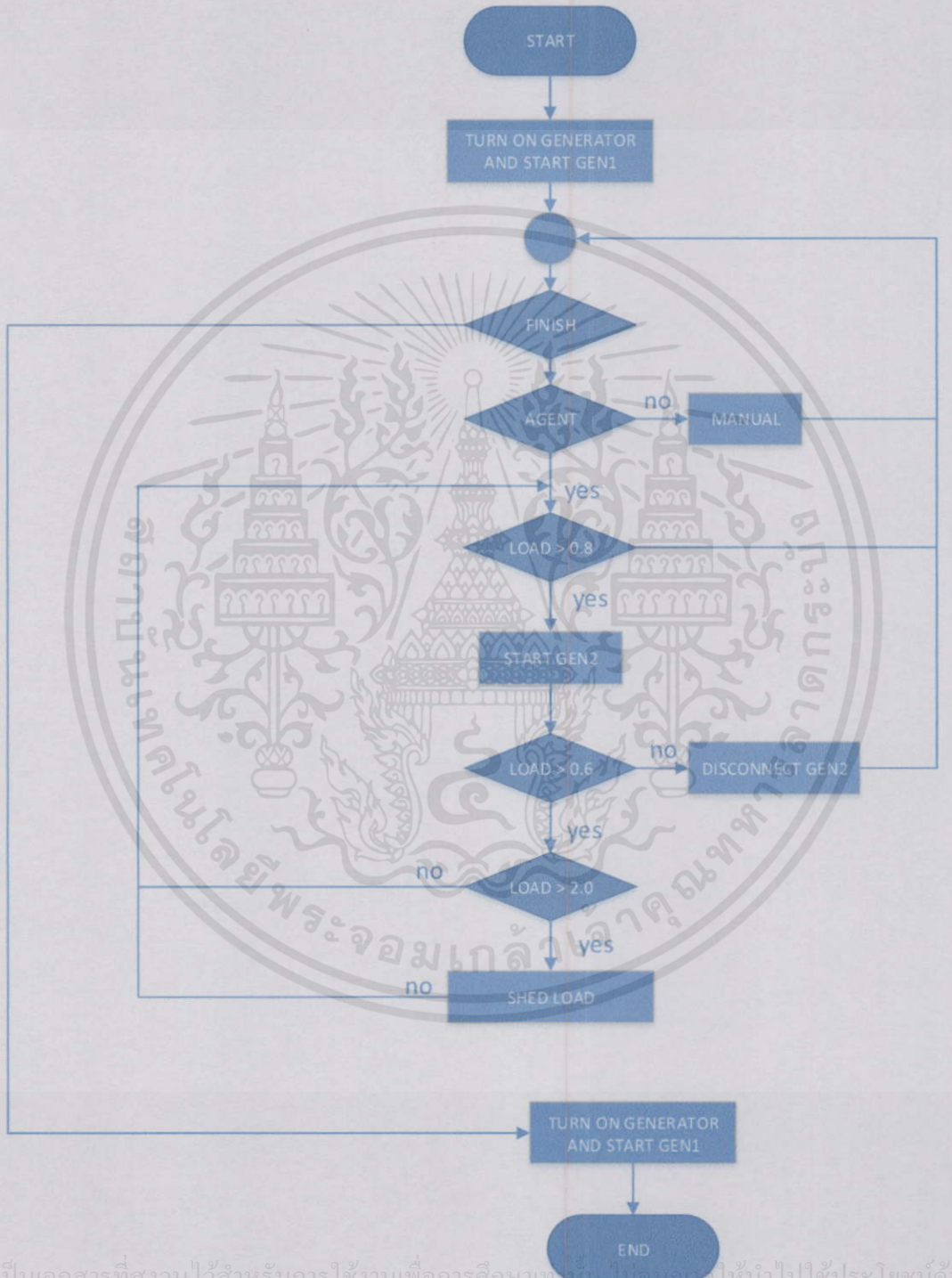
รูปที่ 4.13 ชุดควบคุมโหลดที่สร้างโดย C#



รูปที่ 4.14 ชุดควบคุม Renewable Energy ระบบพลังงานแสงอาทิตย์

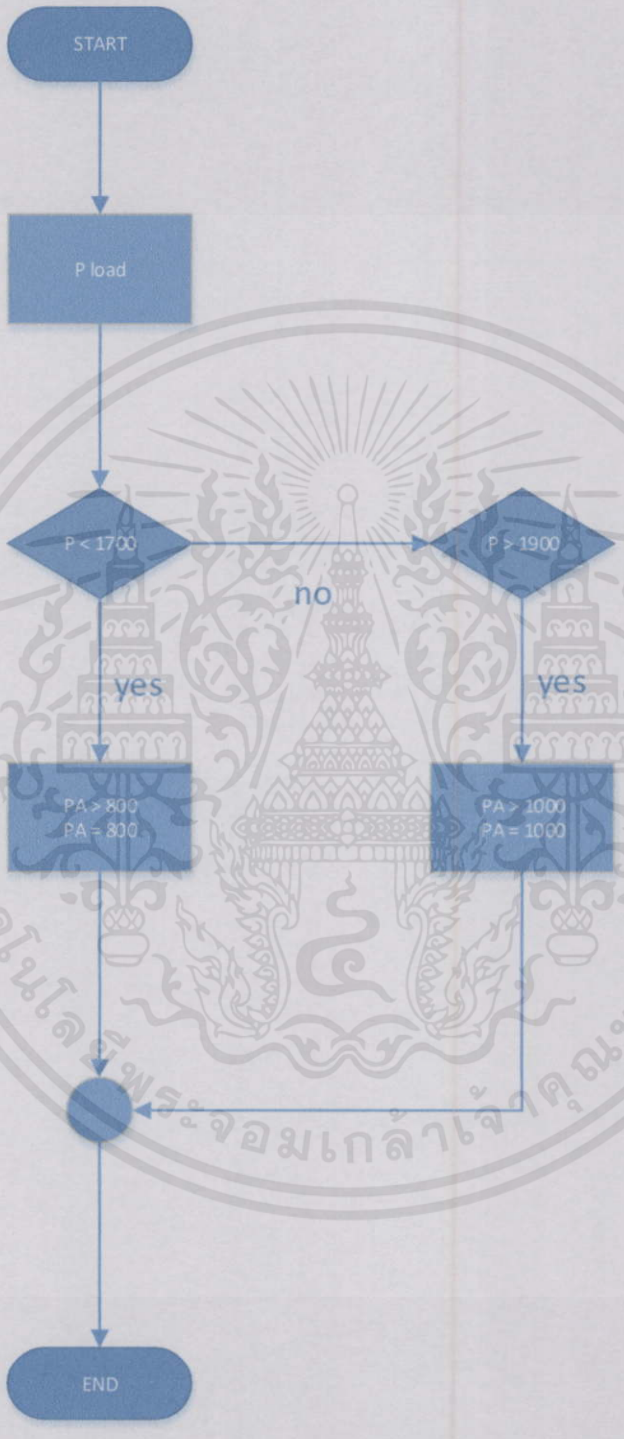
4.4 Flowchart และ State diagram แสดงขั้นตอนการทำงานของระบบต่างๆ

4.4.1 Flowchart of Agent System



รูปที่ 4.15 Flowchart of Agent System

4.4.2 Flowchart of renewable generator

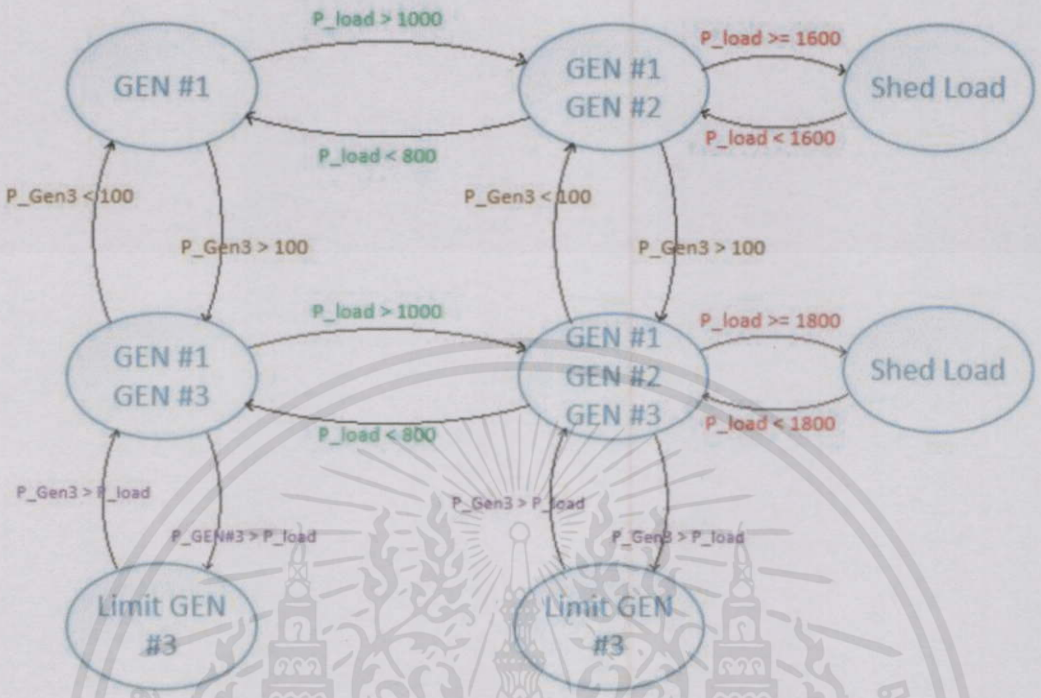


รูปที่ 4.16 Flowchart of renewable generator

4.4.3 Shading load flowchart

รูปที่ 4.17 Shading load flowchart

4.4.4 State diagram ของระบบ



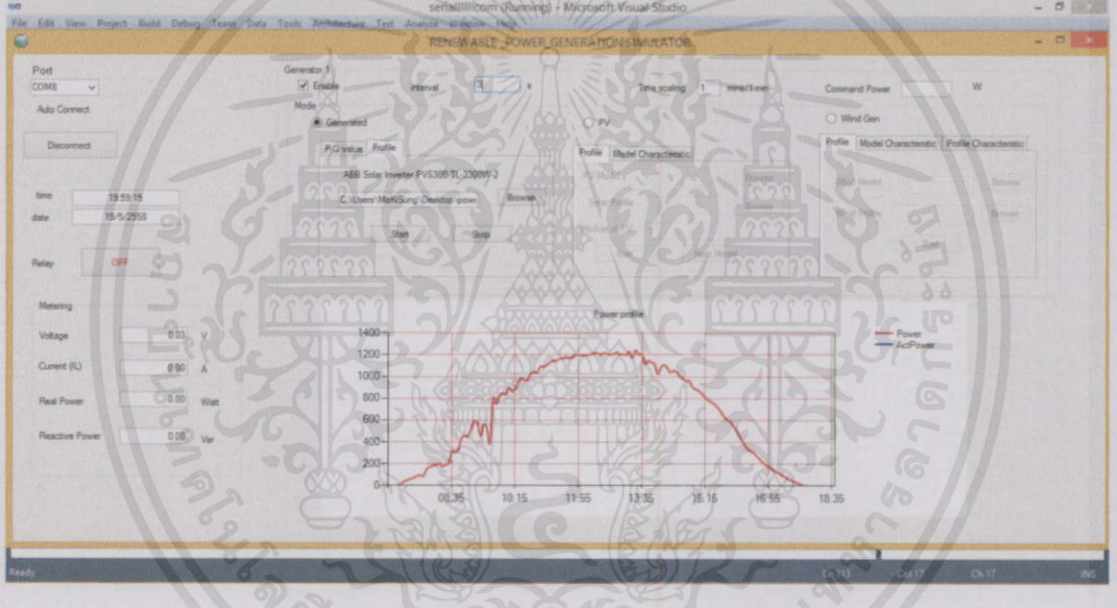
รูปที่ 4.18 State Diagram แสดงขั้นตอนการทำงานของ Agent

4.5 วิธีการทดลอง

4.5.1 ในการทดลองจะเชื่อมต่อ Current Probe เข้ากับ WT1600 Power Meter และแสดงผลออกมาที่จอ Scope ดังรูปที่ 4.11

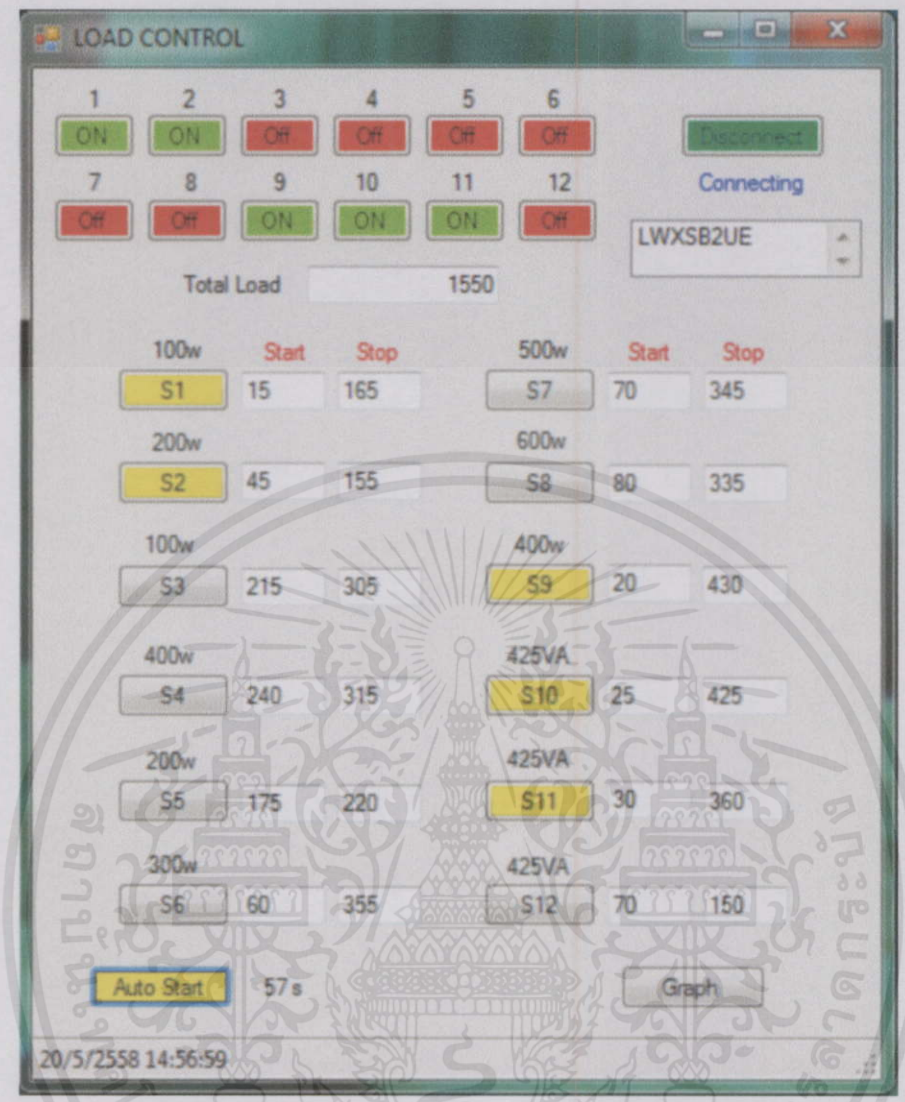
4.5.2 จากนั้น เปิดเซอร์กิตเบรกเกอร์ เปิดเครื่องกำเนิดไฟฟ้าทั้งสามตัว แล้ว Run โปรแกรม CCS ของ Droop Control 1 และ 2 หลังจากนั้น ก็ทำการ Run ผ่าน โปรแกรม CCS ของ Renewable Generator

4.5.3 นำเข้าโปรไฟล์ที่จะใช้ทดลองชุดควบคุม Renewable Energy จากนั้นกด On Relay แล้วกด Start ดังรูปที่ 4.19



รูปที่4.19 การนำเข้าโปรไฟล์ที่จะใช้ทดลองชุดควบคุม Renewable Energy

4.5.4 เปิดชุดควบคุมโหลด ใส่ค่า Time start และ Time stop จากนั้นกด Auto Start จะได้ดังรูปที่ 4.20



รูปที่ 4.20 การใช้งาน Load control ในโหมด Auto start

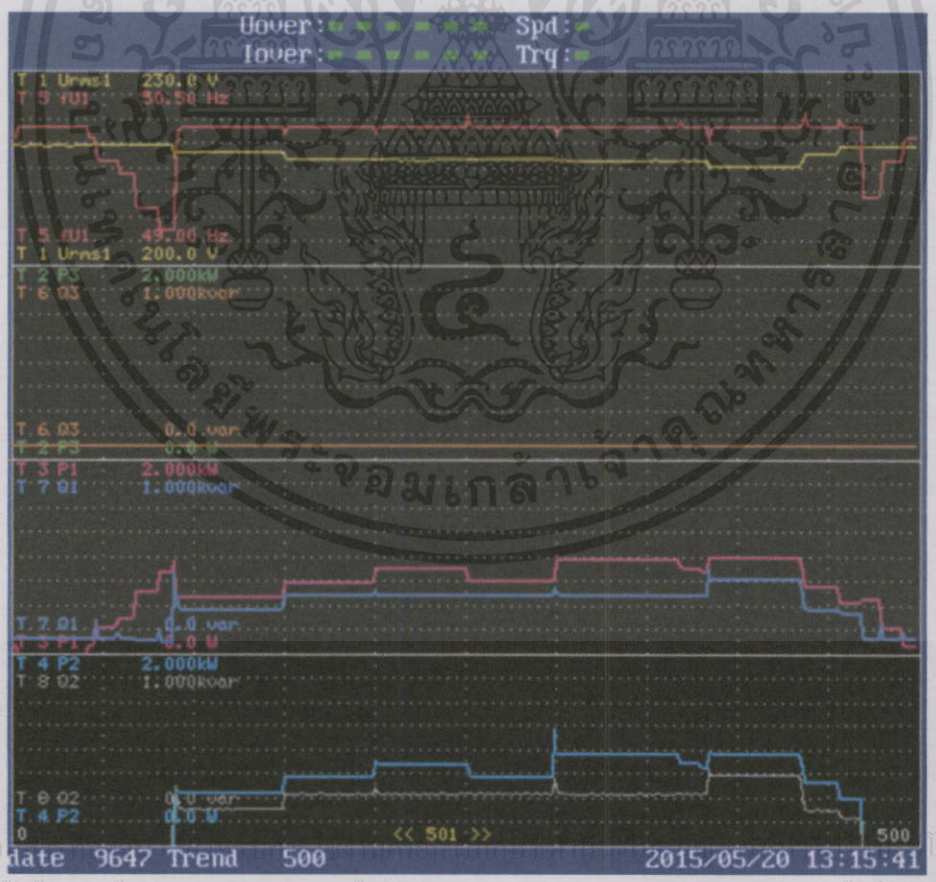
4.5.5 เก็บข้อมูลที่ได้จากการทดลอง ซึ่งในการทดลองของโครงงานนี้จะแบ่งการทดลองออกเป็น 3 กรณี โดยแต่ละการทดลองจะใช้ Agent เป็นตัวออกคำสั่งในการควบคุมอุปกรณ์ต่างๆของทั้งระบบ

4.6 ผลการทดลอง

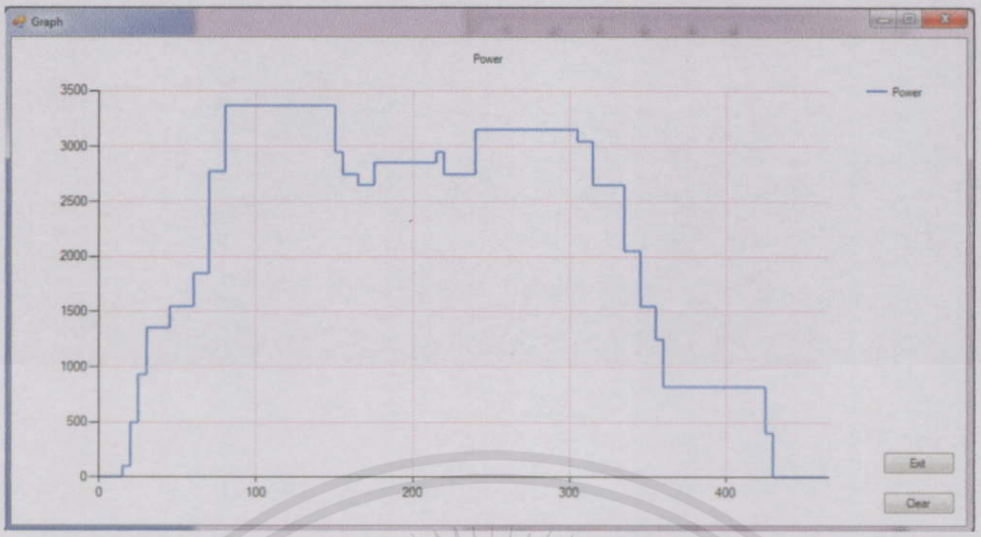
ในการทดลองของโครงงานนี้จะแบ่งออกเป็น 3 กรณีดังนี้

4.6.1 กรณีที่1 Droop Control

Agent จะสั่งให้ Generator1 และ Generator2 ทำการ Droop control เมื่อมี Load มากกว่า 1000W โดยในตอนแรก Generator1 จะทำการจายโหลดเพียงตัวเดียว และหลังจากที่โหลดเพิ่มขึ้นจนถึง 1000W ขึ้นไป Agent จะสั่งการให้ Generator2 ทำงานแบ่งการจ่าย Load ช่วย Generator1 ซึ่งข้อกำหนดในการจ่าย Load ร่วมกันของ Generator ทั้ง 2 ตัวนั้นจะต้งมีข้อกำหนดคือ มีแรงดันเท่ากัน และมีความถี่เดียวกัน ซึ่งข้อกำหนดดังกล่าวจะถูกควบคุมโดยการ Voltage Droop Control ที่ 220V และ Frequency Droop Control ที่ 50Hz จากโปรแกรม CCS ได้ผลการทดลองดังรูปที่ 4.20 โดยเป็นผลการทดลองการจ่ายกำลังไฟฟ้าของแหล่งกำเนิดไฟฟ้าทั้ง 3 ตัวในระบบไมโครกริด โดยมีทั้งกราฟแสดง Real Power กับ Reactive Power และ Voltage กับ Frequency ของแหล่งจ่ายในรูปที่ 4.20 และแสดงกราฟแสดงค่า Power ของโหลดที่ใช้ในแต่ละวันในรูปที่ 4.21



รูปที่ 4.21 กราฟแสดงการดรูประหว่าง Generator1กับ Generator2 ตามลำดับในกรณีการ Droop Control

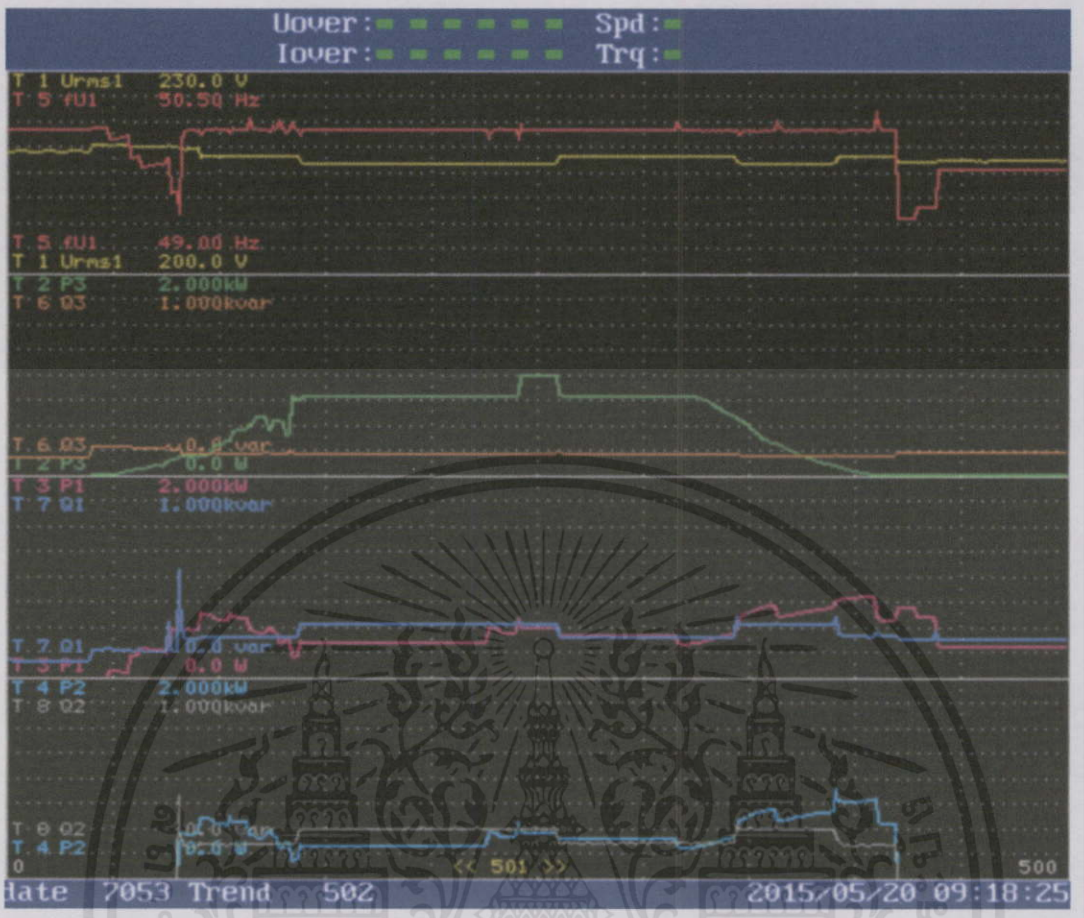


รูปที่ 4.22 กราฟแสดงค่า Power ของโหลดที่ใช้ในแต่ละวัน

4.6.2 กรณีที่ 2 จำกัดการจ่ายโหลดของ PV-gen (Capture PV-gen)

กรณีที่2 จะมีการต่อ Renewable Generator(PV-gen) เข้ามาช่วยจ่าย Load ให้กับระบบและ Agent จะสั่งให้จำกัดการจ่าย Load ของ Renewable Generator เมื่อเกิดช่วงที่ Load มีความต้องการไฟฟ้าน้อย Generator1 และ Generator2 มีความสามารถในการจ่าย Load เพียงพออยู่แล้วเพื่อป้องกันการเกิดกำลังไฟฟ้าไหลเข้าสู่ Generator1 และ Generator2 ในรูปที่ 4.22 จะเห็นได้ว่า Renewable Generator ช่วยจ่ายกำลังไฟฟ้าที่เข้ามาในระบบมโครกริด โดยในกรณีที่มีการจ่ายกำลังไฟฟ้ามากในช่วงเวลาใดเวลาหนึ่ง Renewable Generator จะสามารถทำหน้าที่จำกัดกำลังไฟฟ้าให้ไม่เกินค่าๆหนึ่งเพราะอาจจะเกิดเหตุการณ์ที่กำลังไฟฟ้านั้นไหลย้อนกลับเข้าสู่แหล่งจ่ายทำให้เกิดความเสียหายได้ และภาพรวมของการจ่ายกำลังไฟฟ้าของ Generator ทั้ง 3 ตัว จะแสดงในรูปที่ 4.24

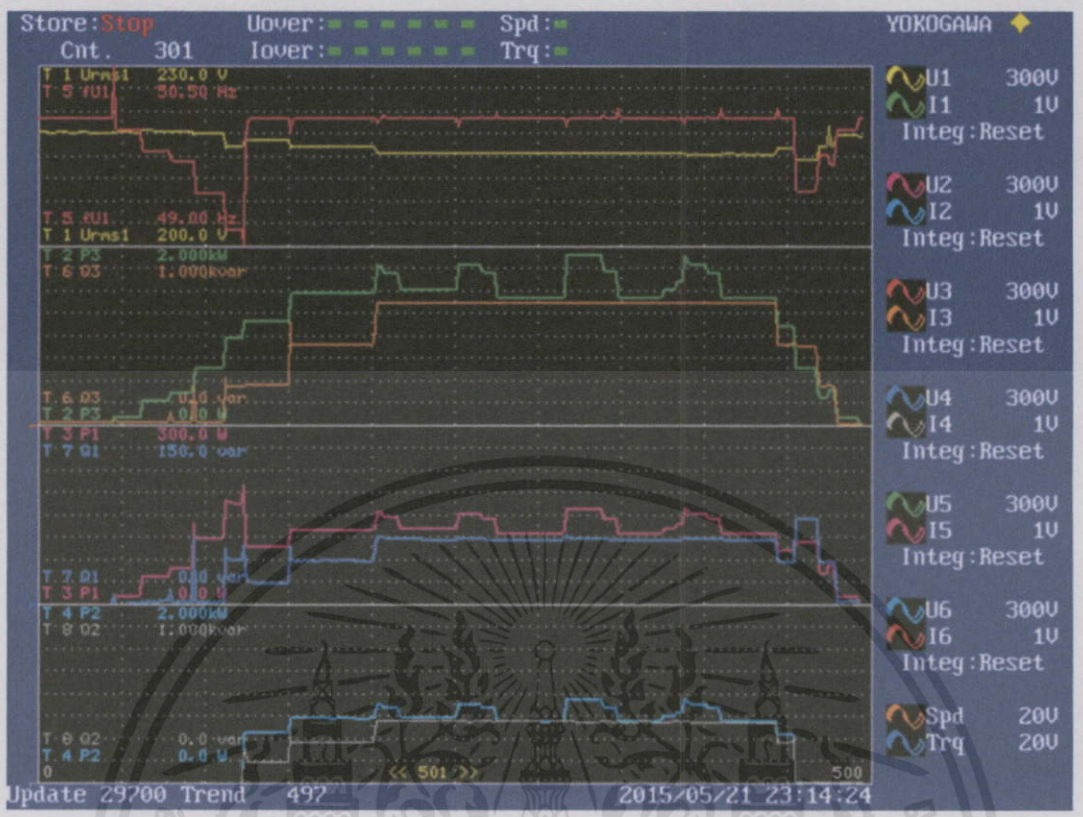
รูปที่4.23 รูปแสดงการจำกัดกำลังไฟฟ้าของ Renewable Generator(PV-gen) ที่จ่ายให้กับโหลด



รูปที่ 4.24 กราฟแสดง Voltage, Frequency, การจ่ายกำลังไฟฟ้าของ Renewable Generator, Genterator1 และ Geanerator2 ตามลำดับ ในกรณีการ Capture PV-gen

4.6.3 กรณีที่ 3 การตัดและเชื่อมต่อโหลดโดยอัตโนมัติ (Shading Load)

ในกรณีนี้ Agent จะสั่งให้ตัด Load ออกจากระบบ เมื่อ Generator ทั้ง 2 ตัว ไม่สามารถจ่ายโหลดได้อย่างเพียงพอ ซึ่ง Agent จะทำการตัดสินใจว่าจะตัดโหลดที่ไม่จำเป็น(Non-Critical Load) ออกจากระบบ โดยจะยังคงโหลดที่จำเป็น(Critical Load) ไว้อยู่ เพื่อรักษาเสถียรภาพของระบบและจะทำการต่อโหลดที่ตัดออกไปเข้ามาในระบบอีกครั้งเมื่อ Generator ทั้ง 2 ตัว มีความสามารถในการจ่ายโหลดมากขึ้นจนเพียงพอในการจ่ายให้กับโหลดเหล่านั้น โดยในรูปที่ 4.25 จะเป็นรูปที่แสดงการใช้กำลังไฟฟ้าของโหลด และกำลังไฟฟ้าที่จ่ายออกโดย Generator1 กับ Generator2



รูปที่ 4.25 กราฟแสดง Voltage, Frequency, การใช้กำลังไฟฟ้าของโหลด, การจ่ายกำลังไฟฟ้าของ Generator1 และ Generator2 ตามลำดับ ของกรณีการ Shading Load

บทที่ 5

บทสรุปและข้อเสนอแนะ

5.1 สรุปผลการวิจัย

โครงงานนี้ถูกนำมาวิจัยเพื่อศึกษาและพัฒนาในเรื่องของเทคนิคการควบคุมในการทำงานของระบบไมโครกริด ซึ่งจะเป็นแนวโน้มใหม่ของเครือข่ายของระบบไฟฟ้าในอนาคต ทั้งนี้ไมโครกริดที่มีระบบควบคุมจะสามารถรักษาเสถียรภาพของตนเองในการที่จะจ่ายโหลดได้อย่างมีประสิทธิภาพ โดยปราศจากปัญหาในโหมด Stand-alone ในยามที่แหล่งจ่ายการไฟฟ้าเกิดปัญหาขึ้น โดยโครงงานนี้จะแบ่งการศึกษาออกเป็นสองส่วนหลักๆ คือ การจำลองระบบไมโครกริดในโหมด Stand-alone และการเขียนโปรแกรมให้ Agent ทำงานแทนคนในการดูแลระบบไมโครกริดให้มีความฉลาดและเสถียรภาพมากที่สุด ผ่านโปรแกรม Microsoft Visual Studio 2013 ด้วยภาษา C# ซึ่งในการทดลองจะแบ่งออกเป็น 3 กรณีด้วยกันคือ กรณีที่1 Droop Control กรณีที่2 จำกัดการจ่ายโหลดของ PV-gen (Capture PV-gen) และ กรณีที่ 3 การตัดและเชื่อมต่อโหลดโดยอัตโนมัติ (Shading Load)

สำหรับการศึกษาและพัฒนาการจำลองระบบไมโครกริดในโหมด Stand-alone ของระบบไฟฟ้าสามเฟสและเฟสเดียวของแหล่งจ่ายสองตัวที่มีพารามิเตอร์เดียวกันนั้นจะทำการศึกษาโดยใช้โปรแกรม Matlab/Simulink เพื่อศึกษาดูว่าระบบไมโครกริดนั้นมีลักษณะอย่างไร และทำงานอย่างไร เพื่อที่จะต่อยอดไปสู่การพัฒนาให้ระบบไมโครกริดธรรมดากลายไปเป็นระบบสมาร์ทไมโครกริดโดยใช้ Agent ในการควบคุมแบบอัตโนมัติ

สำหรับผลการทดลองนั้นจะมี 3 กรณีคือ กรณีที่1 Droop Control พบว่าผลลัพธ์ที่ออกมานั้นค่อนข้างเป็นที่น่าพอใจ กล่าวคือเมื่อเรากำหนดสัดส่วนการแชร์โหลด Droop Coefficient ของมุมแรงดันและแรงดันไว้เท่ากันที่ 1:1 แล้ว พบว่ามีการแชร์อัตราส่วนกำลังไฟฟ้าจริง และกำลังไฟฟ้ารีแอคทีฟ ที่อัตราส่วนประมาณ 1:1 ภายใต้ระยะเวลาเข้าสู่สภาวะคงตัวที่เหมาะสม และมีและมีการคงแรงดันไว้ที่ 220V และการดึงความถี่กลับเข้าสู่ 50 Hz ซึ่งเป็นการควบคุมแบบ Secondary Control ของระบบไมโครกริดในการรักษาเสถียรภาพของการทำงาน

กรณีที่2 จำกัดการจ่ายโหลดของ PV-gen (Capture PV-gen) พบว่าการทำงานของ Agent นั้นยังมีความฉลาดที่จะจำกัดการจ่ายกำลังไฟฟ้าของ PV-gen เข้ามาในระบบไมโครกริดในช่วงที่มีการใช้โหลดน้อย เพื่อป้องกันการเกิดกำลังไฟฟ้าเกินความต้องการของโหลดและป้องกันกำลังไฟฟ้านั้นไหลกลับเข้าสู่ Generator1 และ Generator2 เพราะจะเป็นสาเหตุให้ Generator1 และ Generator2 เสียหายได้

กรณีที่ 3 การตัดและเชื่อมต่อโหลดโดยอัตโนมัติ (Shading Load) พบว่า Agent สามารถสั่งให้มีการตัดโหลดที่ไม่จำเป็น(Non-Critical Load) ออกจากระบบ โดยจะยังคงโหลดที่จำเป็น(Critical Load) ไว้อยู่ เพื่อรักษาเสถียรภาพของระบบและจะทำการต่อโหลดที่ตัดออกไปเข้ามาในระบบอีกครั้งเมื่อ Generator ทั้ง 2 ตัว มีความสามารถในการจ่ายโหลดมากขึ้นจนเพียงพอในการจ่ายให้กับโหลดเหล่านั้น ได้อย่างมีประสิทธิภาพอีกเช่นกัน

ข้อดีของระบบไมโครกริดที่ใช้ Agent เป็นตัวควบคุม คือ Agent จะมีความยืดหยุ่นในการทำงานมากโดยจะขึ้นอยู่กับการโปรแกรมโดยผู้เขียนเองว่าต้องการให้ Agent ทำงานในกรณีใดบ้าง และ Agent เองก็มีประสิทธิภาพและเสถียรภาพมากด้วยเช่นกัน จึงทำให้ระบบไมโครกริดมีเสถียรภาพมากยิ่งขึ้น

5.2 ข้อเสนอแนะ

ระบบไมโครกริดที่ทำการจำลองนั้นเป็นระบบจำลองอย่างง่ายเพื่อต้องการให้เกิดความเข้าใจในการใช้งานในส่วนโปรแกรมและการประสานงานตามรูปแบบการทำงานของไมโครกริดแต่ระบบ

ซึ่งในทางปฏิบัติแล้วอาจมีความหลากหลายและซับซ้อนมากกว่า เช่น ระบบตัดสินใจการทำงานในโหมดแบบแยกอิสระและแบบเชื่อมต่อกับกริดหลัก ตามเงื่อนไขต่างๆ เช่น เมื่อเครื่องกำเนิดไฟฟ้าถูกตัดออกจากระบบ เกิดความผิดพร่องในระบบ หรือเกิดความผิดพร่องในกริดหลัก เป็นต้น อาจมีการนำอุปกรณ์จำพวกอุปกรณ์สำรองพลังงานเข้ามาในระบบและมีระบบการปลดโหลดที่ไม่สำคัญออกในกรณีเกิดเหตุความผิดพลาดต่างๆ จึงสามารถนำโปรแกรมนี้ไปประยุกต์กับระบบจริงๆได้ โดยตัวโปรแกรมสามารถเชื่อมต่อกับระบบเครือข่าย สามารถนำค่าตัวแปรต่างๆเพื่อไปควบคุมเครื่องกำเนิดไฟฟ้าที่อยู่ในระยะไกลได้ ซึ่งผู้เขียนหวังว่าจะมีผู้ที่สนใจนำไปพัฒนาให้ดีขึ้นต่อไปในอนาคตได้

เอกสารอ้างอิง

[1] นายจิราวุฒิ ภู่สาร, นายณัฐพล นันทวิชาวุธสาร, นายณัฐพล ป้อมป้องกัน, “ระบบจัดการพลังงานสำหรับไมโครกริดโดยเอเจนท์”, ปริญญานิพนธ์วิศวกรรมศาสตรบัณฑิต, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, ปีการศึกษา 2554

[2] นายพยุงศักดิ์ แซ่หลิ่ว, นายพงศ์ภรณ์ เตชะชลประเสริฐ, นายไพโรจน์ คอนเอม, นายมหาราชราชสีห์, “โครงการสาธิตไมโครกริดของศูนย์นวัตกรรมพลังงาน (ClnES Microgrid Demonstration Operation and Control)”, ปริญญานิพนธ์วิศวกรรมศาสตรบัณฑิต, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, ปีการศึกษา 2555

[3] นายธนกฤต กิตติวรารัตน์, นายปณต เชิดชูเหล่า, นายพัฒนกิต ชูติกุล, “ชุดจำลองการกำเนิดกำลังไฟฟ้าพลังงานทดแทน”, ปริญญานิพนธ์วิศวกรรมศาสตรบัณฑิต, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, ปีการศึกษา 2555

[4] นายวัจเรจ หน่อเทพ, นายศรายุทธ จิตประสงค์, นายศันสนะ ตรีอาทิตย์โยธิน, นายศิรสิทธิ์ พิริยะคุณธร“ชุดจำลองการกำเนิดกำลังไฟฟ้าพลังงานทดแทน”, ปริญญานิพนธ์วิศวกรรมศาสตรบัณฑิต, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, ปีการศึกษา 2556

[5] นายพสิษฐ์ มณีทิพย์, นายพัทธพล เรืองเดชวรชัย, นายพิรุณ แสงกุล, นายภูชกร เปลี่ยนเชาว์ “ชุดจำลองการกำเนิดกำลังไฟฟ้าพลังงานทดแทน”, ปริญญานิพนธ์วิศวกรรมศาสตรบัณฑิต, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, ปีการศึกษา 2556

ภาคผนวก

ภาคผนวก ก

ภาษา C/C# เบื้องต้น

ก.1 ชนิดของตัวแปรและการตั้งชื่อตัวแปร

ชนิดของตัวแปรในภาษา C/C# แบ่งออกได้ดังตารางที่ ก.1

ตารางที่ ก.1 ชนิดของตัวแปรในภาษา C/C#

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 ตัวแปร จำนวนเต็ม และตัวแปรทศนิยม มีดังนี้

ตารางที่ ก.3 ตัวดำเนินการทางคณิตศาสตร์

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

ก.4 การดำเนินการทางตรรกะ

ตารางที่ ก.4 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.5 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 ซึ่งสมาชิกของอาเรย์อาจเป็นตัวแปรพื้นฐานหรือตัวแปรอ้างอิงก็ได้ จำนวนสมาชิกของอะเรย์มีขนาดแน่นอน และสมาชิกของอาเรย์แต่ละตัวจะมีลำดับประจำตัวอยู่ ตัวอย่างประกาศตัวแปรอาเรย์ 1 มิติ เป็นดังนี้

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

ดังนั้น ตัวแปรที่เป็นสมาชิกของอาเรย์ n ได้แก่ n[0],n[1],n[2], ... ,n[10]

ภาคผนวก ข
การสร้าง Load Agent โดยใช้ภาษา C#

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO.Ports;
using System.Threading;
using System.Timers;
using Graph = System.Windows.Forms.DataVisualization.Charting;
using Lawicel;

namespace aaa
{
    public partial class From1 : Form
    {
        Form2 Form2 = new Form2();
        Int32 Pload;
        static uint handle;
        //      static uint sentframes;
        //      static uint receivedframes;

        int sum;
        int count2 = 0;

        int[] v = new int[4];
        Int32 number1, number2, number3, number4, number5, number6, number7,
number8,
            number9, number10, number11, number12, number13, number14,
number15, number16,
            number17, number18, number19, number20, number21, number22,
number23, number24;
        int second = 0;

        public From1()
```

```
{
    InitializeComponent();

}

int st_can = 0;
private void button13_Click(object sender, EventArgs e)
{

    if (st_can == 0)
    {
        st_can = 1;

        if (listBox1.SelectedIndex == -1)
        {
            handle = CANUSB.canusb_Open(IntPtr.Zero,
                                        CANUSB.CAN_BAUD_1M,
                                        CANUSB.CANUSB_ACCEPTANCE_CODE_ALL,
                                        CANUSB.CANUSB_ACCEPTANCE_MASK_ALL,
                                        CANUSB.CANUSB_FLAG_TIMESTAMP);
        }
        else
        {
            string device = listBox1.Items[listBox1.SelectedIndex].ToString();
            handle = CANUSB.canusb_Open(device,
                                        CANUSB.CAN_BAUD_1M,
                                        CANUSB.CANUSB_ACCEPTANCE_CODE_ALL,
                                        CANUSB.CANUSB_ACCEPTANCE_MASK_ALL,
                                        CANUSB.CANUSB_FLAG_TIMESTAMP);
        }

        if (handle > 0)
        {
            labelp1.Text = "Connecting";
            labelp1.ForeColor = Color.Blue;
            button13.Text = "Disconnect";
            button13.BackColor = Color.SpringGreen;
            timer5.Enabled = true;
```

```
            /*                  OpenLabel.Text = "Open OK";
                                OpenButton.Enabled = false;
                                StatusButton.Enabled = true;
                                WriteButton.Enabled = true;
                                CloseButton.Enabled = true;
                                CloseLabel.Text = "";
                                timer1.Enabled = true;
                                VersionInfoButton.Enabled = true;
                                VersionLabel.Text = "";
            */
        }
        else
        {
            labelp1.Text = "Failed to Open";
            labelp1.ForeColor = Color.Brown;

        }

    }
    else
    {

        { labelp1.Text = "Closing"; }
        // port2.Open();
        //   if (port2.IsOpen) { lapelp2.Text = "Opening"; }
        st_can = 0;
        button13.Text = "Connect";
        button13.BackColor = Color.Empty;
        int res = CANUSB.canusb_Close(handle);
    }


}
int old;
public void DataReceivedHandler_1(object sender, SerialDataReceivedEventArgs
e)// from arduino
{
```

```
Thread.Sleep(30);
byte[] buf1 = new byte[2];
int[] s = new int[16];
int n0, n1, n3;
int powof2 = 2;
//          port1.Read(buf1, 0, 2);
n0 = Convert.ToInt16(buf1[0]);
n1 = Convert.ToInt16(buf1[1]);
n3 = n1 * 256 + n0;
for (int i = 0; i < 12; i++)
{
    powof2 = powof2 * 2;
    if (i == 0) { powof2 = 1; }
    s[i] = (n3 / powof2) % 2;
}

if (old != n3)
{
    if (s[0] == 1)
    {
        button15.BackColor = Color.GreenYellow;
        button15.Text = "ON";
        button1.BackColor = Color.Yellow;
        y[1] = 100;
        n[1] = 1;

    }
    else //(s[0] == 0)
    {
        button1.BackColor = Color.Empty;
        n[1] = 0;
        y[1] = 0;
        button15.BackColor = Color.Red;
        button15.Text = "Off";
    }
    if (s[1] == 1)
    {
        button16.BackColor = Color.GreenYellow;
```

```
        button16.Text = "ON";
        button2.BackColor = Color.Yellow;
        y[2] = 200;
        n[2] = 1;
    }
    else //(s[1] == 0)
    {
        button2.BackColor = Color.Empty;
        y[2] = 0;
        n[2] = 0;
        button16.BackColor = Color.Red;
        button16.Text = "Off";
    }
    if (s[2] == 1)
    {
        button17.BackColor = Color.GreenYellow;
        button17.Text = "ON";
        button3.BackColor = Color.Yellow;
        y[3] = 100;
        n[3] = 1;

    }
    else //(s[2] == 0)
    {
        button3.BackColor = Color.Empty;
        n[3] = 0;
        y[3] = 0;
        button17.BackColor = Color.Red;
        button17.Text = "Off";
    }
    if (s[3] == 1)
    {
        button18.BackColor = Color.GreenYellow;
        button18.Text = "ON";
        button4.BackColor = Color.Yellow;
        y[4] = 400;
        n[4] = 1;
```

```
}
else //(s[3] == 0)
{

    button4.BackColor = Color.Empty;
    y[4] = 0;
    n[4] = 0;
    button18.BackColor = Color.Red;
    button18.Text = "Off";
}
if (s[4] == 1)
{
    button19.BackColor = Color.GreenYellow;
    button19.Text = "ON";
    button5.BackColor = Color.Yellow;
    y[5] = 200;
    n[5] = 1;
}
else //(s[4] == 0)
{
    button5.BackColor = Color.Empty;
    y[5] = 0;
    n[5] = 0;
    button19.BackColor = Color.Red;
    button19.Text = "Off";

}
if (s[5] == 1)
{
    button20.BackColor = Color.GreenYellow;
    button20.Text = "ON";
    button6.BackColor = Color.Yellow;
    y[6] = 300;
    n[6] = 1;
}
else //(s[5] == 0)
{
    button6.BackColor = Color.Empty;
```

```
        y[6] = 0;
        n[6] = 0;
        button20.BackColor = Color.Red;
        button20.Text = "Off";
    }
    if (s[6] == 1)
    {
        button21.BackColor = Color.GreenYellow;
        button21.Text = "ON";
        button7.BackColor = Color.Yellow;
        y[7] = 500;
        n[7] = 1;
    }
    else //(s[6] == 0)
    {
        button7.BackColor = Color.Empty;
        y[7] = 0;
        n[7] = 0;
        button21.BackColor = Color.Red;
        button21.Text = "Off";
    }
    if (s[7] == 1)
    {
        button22.BackColor = Color.GreenYellow;
        button22.Text = "ON";
        button8.BackColor = Color.Yellow;
        y[8] = 600;
        n[8] = 1;
    }
    else //(s[7] == 0)
    {
        button8.BackColor = Color.Empty;
        y[8] = 0;
        n[8] = 0;
        button22.BackColor = Color.Red;
        button22.Text = "Off";
    }
    if (s[8] == 1)
```

```
{
    button23.BackColor = Color.GreenYellow;
    button23.Text = "ON";
    button9.BackColor = Color.Yellow;
    y[9] = 400;
    n[9] = 1;
}
else //(s[8] == 0)
{
    button9.BackColor = Color.Empty;
    y[9] = 0;
    n[9] = 0;
    button23.BackColor = Color.Red;
    button23.Text = "Off";
}
if (s[9] == 1)
{
    button24.BackColor = Color.GreenYellow;
    button24.Text = "ON";
    button10.BackColor = Color.Yellow;
    y[10] = 425;
    n[10] = 1;
}
else //(s[9] == 0)
{
    button10.BackColor = Color.Empty;
    y[10] = 0;
    n[10] = 0;
    button24.BackColor = Color.Red;
    button24.Text = "Off";
}
if (s[10] == 1)
{
    button25.BackColor = Color.GreenYellow;
    button25.Text = "ON";
    button11.BackColor = Color.Yellow;
    y[11] = 425;
    n[11] = 1;
```

```
            }
            else //(s[10] == 0)
            {
                button11.BackColor = Color.Empty;
                y[11] = 0;
                n[11] = 0;
                button25.BackColor = Color.Red;
                button25.Text = "Off";
            }
            if (s[11] == 1)
            {
                button26.BackColor = Color.GreenYellow;
                button26.Text = "ON";
                button12.BackColor = Color.Yellow;
                y[12] = 425;
                n[12] = 1;
            }
            else //(s[11] == 0)
            {
                button12.BackColor = Color.Empty;
                y[12] = 0;
                n[12] = 0;
                button26.BackColor = Color.Red;
                button26.Text = "Off";
            }
            this.Invoke((MethodInvoker)delegate
            {
                sumpower(y);
            });
        }
        old = n3;
    }
    public void DataReceivedHandler_2(object sender, SerialDataReceivedEventArgs
e)// from agent
    {
        Thread.Sleep(20);
        byte[] buf2 = new byte[1];
```

```
}
byte[] byte_send = new byte[15];
int j;
int[] n = new int[13];
int[] y = new int[13];
ulong buffold;
private void CAN_WRITE(ulong data)
{
    if (buffold != data)
    {
    CANUSB.CANMsg msg = new CANUSB.CANMsg();
    msg.id = 0x30;
    msg.len = 1;
    msg.flags = 0;
    msg.data = data;
    int rv = CANUSB.canusb_Write(handle, ref msg);
    buffold = data;
    }
}

//        byte[] byte_send1 = new byte[1];
//        byte[] byte_send = new byte[15];
//        int j = 0;
//        int[] k = byte_send[0];
private void button1_Click(object sender, EventArgs e)
{
    j = 1;
    if (n[1] == 0)
    {
        byte_send[j] = 65; //on
        button1.BackColor = Color.Yellow;
        button15.BackColor = Color.GreenYellow;
        button15.Text = "ON";
        y[1] = 100;
        n[1] = 1;
    }
    else
    {
```

```
        byte_send[j] = 66;  //off
        n[1] = 0;
        y[1] = 0;
        button1.BackColor = Color.Empty;
        button15.BackColor = Color.Red;
        button15.Text = "Off";
    }
    CAN_WRITE(byte_send[j]);
    sumpower(y);
}

private void button2_Click(object sender, EventArgs e)
{
    j = 2;
    if (n[2] == 0)
    {
        byte_send[j] = 67; //on
        y[2] = 200;
        n[2] = 1;
        button16.BackColor = Color.GreenYellow;
        button16.Text = "ON";
        //button2.Text = "200W";
        button2.BackColor = Color.Yellow;
    }
    else
    {
        byte_send[j] = 68;  //off
        y[2] = 0;
        n[2] = 0;
        // button2.Text = "S2";
        button16.BackColor = Color.Red;
        button16.Text = "Off";
        button2.BackColor = Color.Empty;
    }
    CAN_WRITE(byte_send[j]);
    sumpower(y);

}
```

```
private void button3_Click(object sender, EventArgs e)
{
    j = 3;
    if (n[3] == 0)
    {
        byte_send[j] = 69; //on
        y[3] = 100;
        n[3] = 1;
        button17.BackColor = Color.GreenYellow;
        button17.Text = "ON";
        button3.BackColor = Color.Yellow;
    }
    else
    {
        byte_send[j] = 70;  //off
        n[3] = 0;
        y[3] = 0;
        button17.BackColor = Color.Red;
        button17.Text = "Off";
        button3.BackColor = Color.Empty;
    }
    CAN_WRITE(byte_send[j]);
    sumpower(y);
}

private void button4_Click(object sender, EventArgs e)
{
    j = 4;
    if (n[4] == 0)
    {
        byte_send[j] = 71; //on
        y[4] = 400;
        n[4] = 1;
        button18.BackColor = Color.GreenYellow;
        button18.Text = "ON";
        button4.BackColor = Color.Yellow;
    }
```

```
    else
    {
        byte_send[j] = 72;  //off
        n[4] = 0;
        y[4] = 0;
        button18.BackColor = Color.Red;
        button18.Text = "Off";
        button4.BackColor = Color.Empty;
    }
    CAN_WRITE(byte_send[j]);
    sumpower(y);
}

private void button5_Click(object sender, EventArgs e)
{
    if (n[5] == 0)
    {
        byte_send[j] = 73; //on
        y[5] = 200;
        n[5] = 1;
        button19.BackColor = Color.GreenYellow;
        button19.Text = "ON";
        button5.BackColor = Color.Yellow;
    }
    else
    {
        byte_send[j] = 74;  //off
        y[5] = 0;
        n[5] = 0;
        button19.BackColor = Color.Red;
        button19.Text = "Off";
        button5.BackColor = Color.Empty;
    }
    CAN_WRITE(byte_send[j]);
    sumpower(y);
}

private void button6_Click(object sender, EventArgs e)
```

```
{
    if (n[6] == 0)
    {
        byte_send[j] = 75; //on
        y[6] = 300;
        n[6] = 1;
        button20.BackColor = Color.GreenYellow;
        button20.Text = "ON";
        button6.BackColor = Color.Yellow;
    }
    else
    {
        byte_send[j] = 76;  //off
        y[6] = 0;
        n[6] = 0;
        button20.BackColor = Color.Red;
        button20.Text = "Off";
        button6.BackColor = Color.Empty;
    }
    CAN_WRITE(byte_send[j]);
    sumpower(y);
}

private void button7_Click(object sender, EventArgs e)
{
    if (n[7] == 0)
    {
        byte_send[j] = 77; //on
        y[7] = 500;
        n[7] = 1;
        button21.BackColor = Color.GreenYellow;
        button21.Text = "ON";
        button7.BackColor = Color.Yellow;
    }
    else
    {
        byte_send[j] = 78;  //off
        y[7] = 0;
```

```
            n[7] = 0;
            button21.BackColor = Color.Red;
            button21.Text = "Off";
            button7.BackColor = Color.Empty;
        }
        CAN_WRITE(byte_send[j]);
        sumpower(y);
    }

    private void button8_Click(object sender, EventArgs e)
    {
        if (n[8] == 0)
        {
            byte_send[j] = 79; //on
            y[8] = 600;
            n[8] = 1;
            button22.BackColor = Color.GreenYellow;
            button22.Text = "ON";
            button8.BackColor = Color.Yellow;
        }
        else
        {
            byte_send[j] = 80;  //off
            y[8] = 0;
            n[8] = 0;
            button22.BackColor = Color.Red;
            button22.Text = "Off";
            button8.BackColor = Color.Empty;
        }
        CAN_WRITE(byte_send[j]);
        sumpower(y);
    }

    private void button9_Click(object sender, EventArgs e)
    {
        if (n[9] == 0)
        {
            byte_send[j] = 81; //on
```

```
            y[9] = 400;
            n[9] = 1;
            button23.BackColor = Color.GreenYellow;
            button23.Text = "ON";
            button9.BackColor = Color.Yellow;
        }
        else
        {
            byte_send[j] = 82;  //off
            y[9] = 0;
            n[9] = 0;
            button9.BackColor = Color.Empty;
            button23.BackColor = Color.Red;
            button23.Text = "Off";
        }
        CAN_WRITE(byte_send[j]);
        sumpower(y);
    }

    private void button10_Click(object sender, EventArgs e)
    {
        if (n[10] == 0)
        {
            byte_send[j] = 83; //on
            y[10] = 425;
            n[10] = 1;
            button24.BackColor = Color.GreenYellow;
            button10.BackColor = Color.Yellow;
        }
        else
        {
            byte_send[j] = 84;  //off
            y[10] = 0;
            n[10] = 0;
            button24.BackColor = Color.Red;
            button24.Text = "Off";
            button10.BackColor = Color.Empty;
        }
```

```
    CAN_WRITE(byte_send[j]);
    sumpower(y);
}

private void button11_Click(object sender, EventArgs e)
{
    if (n[11] == 0)
    {

        byte_send[j] = 85; //on
        y[11] = 425;
        n[11] = 1;
        button25.BackColor = Color.GreenYellow;
        button25.Text = "ON";
        button11.BackColor = Color.Yellow;
    }
    else
    {

        byte_send[j] = 86;  //off
        n[11] = 0;
        y[11] = 0;
        button25.BackColor = Color.Red;
        button25.Text = "Off";
        button11.BackColor = Color.Empty;
    }
    CAN_WRITE(byte_send[j]);
    sumpower(y);
}

private void button12_Click(object sender, EventArgs e)
{
    if (n[12] == 0)
    {
        byte_send[j] = 87; //on
        y[12] = 425;
        n[12] = 1;
        button26.BackColor = Color.GreenYellow;
```

```
            button26.Text = "ON";
            button12.BackColor = Color.Yellow;
        }
        else
        {
            byte_send[j] = 88;  //off
            y[12] = 0;
            n[12] = 0;
            button26.BackColor = Color.Red;
            button26.Text = "Off";
            button12.BackColor = Color.Empty;

        }
        CAN_WRITE(byte_send[j]);
        sumpower(y);
    }



    private void Form1_Load(object sender, EventArgs e)
    {

    }

    private int sumpower(int[] p)
    {

        sum = (p[1] + p[2] + p[3] + p[4] + p[5] + p[6] + p[7] + p[8] + p[9] + p[10] +
p[11] + p[12]);
        sumP.Text = sum.ToString();
        return (sum);
    }
    int second1 = 0;
    int sec = 0;
    int second12 = 0;
    private void timer1_Tick(object sender, EventArgs e)
    {
        toolStripStatusLabel1.Text = DateTime.Now.ToString();
```

```
        second12 = second12 + 1;
        sec = sec + 1;
        byte[] t = new byte[6];
        labeltimer.Text = sec + " s";
        /*      t[0]=71;
              t[1] = 67;
              t[2] = 73;
              t[3] = 85;
              t[4]=  86;
              t[5] = 74;

              if(sec==0)
              {
                  port1.Write(t, 0, 2);//[1]send to arduino

              }
              if (sec == 90) { port1.Write(t, 2, 1); }
              if (sec == 180) { port1.Write(t, 3, 1); }
              if (sec == 270) { port1.Write(t, 4, 2); }
              */


}


private void From1_Load(object sender, EventArgs e)
{
    this.Form2.chart1.Series["Power"].Points.AddXY(0, 0);


    StringBuilder buf = new StringBuilder(32);
    int i, cnt;
    // Fill the listbox with data
    cnt = CANUSB.canusb_getFirstAdapter(buf, 32);
    listBox1.Items.Add(buf);
```

```
    for (i = 1; i < cnt; i++)
    {
        if (CANUSB.canusb_getNextAdapter(buf, 32) > 0)
        {
            listBox1.Items.Add(buf);
        }
    }



    this.Form2.chart1.Titles.Add("Power");
    timer1.Interval = 1000;
    timer2.Interval = 1000;

}


private void button14_Click(object sender, EventArgs e)
{


    if (string.IsNullOrEmpty(textBox1.Text)) { textBox1.Text = "0"; }
    if (string.IsNullOrEmpty(textBox2.Text)) { textBox2.Text = "0"; }
    if (string.IsNullOrEmpty(textBox3.Text)) { textBox3.Text = "0"; }
    if (string.IsNullOrEmpty(textBox4.Text)) { textBox4.Text = "0"; }
    if (string.IsNullOrEmpty(textBox5.Text)) { textBox5.Text = "0"; }
    if (string.IsNullOrEmpty(textBox6.Text)) { textBox6.Text = "0"; }
    if (string.IsNullOrEmpty(textBox7.Text)) { textBox7.Text = "0"; }
    if (string.IsNullOrEmpty(textBox8.Text)) { textBox8.Text = "0"; }
    if (string.IsNullOrEmpty(textBox9.Text)) { textBox9.Text = "0"; }
    if (string.IsNullOrEmpty(textBox10.Text)) { textBox10.Text = "0"; }
    if (string.IsNullOrEmpty(textBox11.Text)) { textBox11.Text = "0"; }
    if (string.IsNullOrEmpty(textBox12.Text)) { textBox12.Text = "0"; }
    if (string.IsNullOrEmpty(textBox13.Text)) { textBox13.Text = "0"; }
    if (string.IsNullOrEmpty(textBox14.Text)) { textBox14.Text = "0"; }
    if (string.IsNullOrEmpty(textBox15.Text)) { textBox15.Text = "0"; }
    if (string.IsNullOrEmpty(textBox16.Text)) { textBox16.Text = "0"; }
    if (string.IsNullOrEmpty(textBox17.Text)) { textBox17.Text = "0"; }
```

```
if (string.IsNullOrEmpty(textBox18.Text)) { textBox18.Text = "0"; }
if (string.IsNullOrEmpty(textBox19.Text)) { textBox19.Text = "0"; }
if (string.IsNullOrEmpty(textBox20.Text)) { textBox20.Text = "0"; }
if (string.IsNullOrEmpty(textBox21.Text)) { textBox21.Text = "0"; }
if (string.IsNullOrEmpty(textBox22.Text)) { textBox22.Text = "0"; }
if (string.IsNullOrEmpty(textBox23.Text)) { textBox23.Text = "0"; }
if (string.IsNullOrEmpty(textBox24.Text)) { textBox24.Text = "0"; }

number1 = Convert.ToInt32(textBox1.Text);
number2 = Convert.ToInt32(textBox2.Text);
number3 = Convert.ToInt32(textBox3.Text);
number4 = Convert.ToInt32(textBox4.Text);
number5 = Convert.ToInt32(textBox5.Text);
number6 = Convert.ToInt32(textBox6.Text);
number7 = Convert.ToInt32(textBox7.Text);
number8 = Convert.ToInt32(textBox8.Text);
number9 = Convert.ToInt32(textBox9.Text);
number10 = Convert.ToInt32(textBox10.Text);
number11 = Convert.ToInt32(textBox11.Text);
number12 = Convert.ToInt32(textBox12.Text);
number13 = Convert.ToInt32(textBox13.Text);
number14 = Convert.ToInt32(textBox14.Text);
number15 = Convert.ToInt32(textBox15.Text);
number16 = Convert.ToInt32(textBox16.Text);
number17 = Convert.ToInt32(textBox17.Text);
number18 = Convert.ToInt32(textBox18.Text);
number19 = Convert.ToInt32(textBox19.Text);
number20 = Convert.ToInt32(textBox20.Text);
number21 = Convert.ToInt32(textBox21.Text);
number22 = Convert.ToInt32(textBox22.Text);
number23 = Convert.ToInt32(textBox23.Text);
number24 = Convert.ToInt32(textBox24.Text);

{
    if (v[1] == 0)
    {
        timer1.Start();
        timer2.Start();
```

```
            timer3.Start();
            v[1] = 1;
            button14.BackColor = Color.Yellow;
            button14.Text = "Auto Start";
        }
        else
        {
            timer1.Stop();
            timer2.Stop();
            timer3.Stop();
            second = 0;
            second1 = 0;
            count2 = 0;



            sec = 0;
            button14.BackColor = Color.Empty;
            button14.Text = "Auto Start";
            v[1] = 0;
        }
    }
}

private void timer2_Tick(object sender, EventArgs e)
{
    second = second + 1;
    second1 = second1 + 1;
    count2 = count2 + 1;

    if (second == number1)
    {

        byte_send[j] = 65; //on
        CAN_WRITE(byte_send[j]);
        y[1] = 100;
        n[1] = 1;
        button15.BackColor = Color.GreenYellow;
```

```
    button15.Text = "ON";
    button1.BackColor = Color.Yellow;
    button1.Text = "S1";


}
if (second1 == number2)
{

    byte_send[j] = 66; //off
    CAN_WRITE(byte_send[j]);
    y[1] = 0;
    n[1] = 0;
    button15.BackColor = Color.Red;
    button15.Text = "Off";
    button1.BackColor = Color.Empty;
    button1.Text = "S1";

}
if (second == number3)
{

    byte_send[j] = 67; //on
    CAN_WRITE(byte_send[j]);
    y[2] = 200;
    n[2] = 1;
    button16.BackColor = Color.GreenYellow;
    button16.Text = "ON";
    button2.BackColor = Color.Yellow;
    button2.Text = "S2";
}
if (second1 == number4)
{
```



```
    byte_send[j] = 68; //off
    CAN_WRITE(byte_send[j]);
    y[2] = 0;
```

```
    n[2] = 0;
    button16.BackColor = Color.Red;
    button16.Text = "Off";
    button2.BackColor = Color.Empty;
    button2.Text = "S2";
}


if (second == number5)
{

    byte_send[j] = 69; //on
    CAN_WRITE(byte_send[j]);
    y[3] = 100;
    n[3] = 1;
    button17.BackColor = Color.GreenYellow;
    button17.Text = "ON";
    button3.BackColor = Color.Yellow;
    button3.Text = "S3";
}
if (second1 == number6)
{

    byte_send[j] = 70; //off
    CAN_WRITE(byte_send[j]);
    y[3] = 0;
    n[3] = 0;
    button17.BackColor = Color.Red;
    button17.Text = "Off";
    button3.BackColor = Color.Empty;
    button3.Text = "S3";
}


if (second == number7)
{

    byte_send[j] = 71; //on
```

```
    CAN_WRITE(byte_send[j]);
    y[4] = 400;
    n[4] = 1;
    button18.BackColor = Color.GreenYellow;
    button18.Text = "ON";
    button4.BackColor = Color.Yellow;
    button4.Text = "S4";
}

if (second1 == number8)
{

    byte_send[j] = 72; //off
    CAN_WRITE(byte_send[j]);
    y[4] = 0;
    n[4] = 0;
    button18.BackColor = Color.Red;
    button18.Text = "Off";
    button4.BackColor = Color.Empty;
    button4.Text = "S4";
}

if (second == number9)
{

    byte_send[j] = 73; //on
    CAN_WRITE(byte_send[j]);
    y[5] = 200;
    n[5] = 1;
    button19.BackColor = Color.GreenYellow;
    button19.Text = "ON";
    button5.BackColor = Color.Yellow;
    button5.Text = "S5";
}
if (second1 == number10)
{

    byte_send[j] = 74; //off
```

```
    CAN_WRITE(byte_send[j]);
    y[5] = 0;
    n[5] = 0;
    button19.BackColor = Color.Red;
    button19.Text = "Off";
    button5.BackColor = Color.Empty;
    button5.Text = "S5";
}

if (second == number11)
{

    byte_send[j] = 75; //on
    CAN_WRITE(byte_send[j]);
    y[6] = 300;
    n[6] = 1;
    button20.BackColor = Color.GreenYellow;
    button20.Text = "ON";
    button6.BackColor = Color.Yellow;
    button6.Text = "S6";
}
if (second1 == number12)
{

    byte_send[j] = 76; //off
    CAN_WRITE(byte_send[j]);
    y[6] = 0;
    n[6] = 0;
    button20.BackColor = Color.Red;
    button20.Text = "Off";
    button6.BackColor = Color.Empty;
    button6.Text = "S6";
}

if (second == number13)
{

    byte_send[j] = 77; //on
```

```
    CAN_WRITE(byte_send[j]);
    y[7] = 500;
    n[7] = 1;
    button21.BackColor = Color.GreenYellow;
    button21.Text = "ON";
    button7.BackColor = Color.Yellow;
    button7.Text = "S7";
}
if (second1 == number14)
{

    byte_send[j] = 78; //off
    CAN_WRITE(byte_send[j]);
    y[7] = 0;
    n[7] = 0;
    button21.BackColor = Color.Red;
    button21.Text = "Off";
    button7.BackColor = Color.Empty;
    button7.Text = "S7";
}

if (second == number15)
{

    byte_send[j] = 79; //on
    CAN_WRITE(byte_send[j]);
    y[8] = 600;
    n[8] = 1;
    button22.BackColor = Color.GreenYellow;
    button22.Text = "ON";
    button8.BackColor = Color.Yellow;
    button8.Text = "S8";
}
if (second1 == number16)
{
```



```
    byte_send[j] = 80; //off
    CAN_WRITE(byte_send[j]);
```

```
    y[8] = 0;
    n[8] = 0;
    button22.BackColor = Color.Red;
    button22.Text = "Off";
    button8.BackColor = Color.Empty;
    button8.Text = "S8";
}

if (second == number17)
{

    byte_send[j] = 81; //on
    CAN_WRITE(byte_send[j]);
    y[9] = 400;
    n[9] = 1;
    button23.BackColor = Color.GreenYellow;
    button23.Text = "ON";
    button9.BackColor = Color.Yellow;
    button9.Text = "S9";
}
if (second1 == number18)
{

    byte_send[j] = 82; //off
    CAN_WRITE(byte_send[j]);
    y[9] = 0;
    n[9] = 0;
    button23.BackColor = Color.Red;
    button23.Text = "Off";
    button9.BackColor = Color.Empty;
    button9.Text = "S9";
}

if (second == number19)
{
```



```
    byte_send[j] = 83; //on
    CAN_WRITE(byte_send[j]);
```

```
    y[10] = 425;
    n[10] = 1;
    button24.BackColor = Color.GreenYellow;
    button24.Text = "ON";
    button10.BackColor = Color.Yellow;
    button10.Text = "S10";
}
if (second1 == number20)
{

    byte_send[j] = 84; //off
    CAN_WRITE(byte_send[j]);
    y[10] = 0;
    n[10] = 0;
    button24.BackColor = Color.Red;
    button24.Text = "Off";
    button10.BackColor = Color.Empty;
    button10.Text = "S10";
}

if (second == number21)
{

    byte_send[j] = 85; //on
    CAN_WRITE(byte_send[j]);
    y[11] = 425;
    n[11] = 1;
    button25.BackColor = Color.GreenYellow;
    button25.Text = "ON";
    button11.BackColor = Color.Yellow;
    button11.Text = "S11";
}
if (second1 == number22)
{
```



```
    byte_send[j] = 86; //off
    CAN_WRITE(byte_send[j]);
    y[11] = 0;
```

```
            n[11] = 0;
            button25.BackColor = Color.Red;
            button25.Text = "Off";
            button11.BackColor = Color.Empty;
            button11.Text = "S11";
        }

        if (second == number23)
        {

            byte_send[j] = 87; //on
            CAN_WRITE(byte_send[j]);
            y[12] = 425;
            n[12] = 1;
            button26.BackColor = Color.GreenYellow;
            button26.Text = "ON";
            button12.BackColor = Color.Yellow;
            button12.Text = "S12";
        }
        if (second1 == number24)
        {

            byte_send[j] = 88; //off
            CAN_WRITE(byte_send[j]);
            y[12] = 0;
            n[12] = 0;
            button26.BackColor = Color.Red;
            button26.Text = "Off";
            button12.BackColor = Color.Empty;
            button12.Text = "S12";
        }
        sumpower(y);
    }

    private void textBox1_TextChanged(object sender, EventArgs e)
    {
        if (string.IsNullOrEmpty(textBox1.Text)) { textBox1.Text = "0"; }
        number1 = Convert.ToInt32(textBox1.Text);
```

```
}

private void textBox2_TextChanged(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(textBox2.Text)) { textBox2.Text = "0"; }
    number2 = Convert.ToInt32(textBox2.Text);
}

private void textBox3_TextChanged(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(textBox3.Text)) { textBox3.Text = "0"; }
    number3 = Convert.ToInt32(textBox3.Text);
}

private void textBox4_TextChanged(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(textBox4.Text)) { textBox4.Text = "0"; }
    number4 = Convert.ToInt32(textBox4.Text);
}

private void textBox5_TextChanged(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(textBox5.Text)) { textBox5.Text = "0"; }
    number5 = Convert.ToInt32(textBox5.Text);
}

private void textBox6_TextChanged(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(textBox6.Text)) { textBox6.Text = "0"; }
    number6 = Convert.ToInt32(textBox6.Text);
}

private void textBox7_TextChanged(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(textBox7.Text)) { textBox7.Text = "0"; }
    number7 = Convert.ToInt32(textBox7.Text);
}
```

```
private void textBox8_TextChanged(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(textBox8.Text)) { textBox8.Text = "0"; }
    number8 = Convert.ToInt32(textBox8.Text);
}

private void textBox9_TextChanged(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(textBox9.Text)) { textBox9.Text = "0"; }
    number9 = Convert.ToInt32(textBox9.Text);

}

private void textBox10_TextChanged(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(textBox10.Text)) { textBox10.Text = "0"; }
    number10 = Convert.ToInt32(textBox10.Text);

}

private void textBox11_TextChanged(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(textBox11.Text)) { textBox11.Text = "0"; }
    number11 = Convert.ToInt32(textBox11.Text);

}

private void textBox12_TextChanged(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(textBox12.Text)) { textBox12.Text = "0"; }
    number12 = Convert.ToInt32(textBox12.Text);

}

private void textBox13_TextChanged(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(textBox13.Text)) { textBox13.Text = "0"; }
    number13 = Convert.ToInt32(textBox13.Text);
```

```
}

private void textBox14_TextChanged(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(textBox14.Text)) { textBox14.Text = "0"; }
    number14 = Convert.ToInt32(textBox14.Text);

}

private void textBox15_TextChanged(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(textBox15.Text)) { textBox15.Text = "0"; }
    number15 = Convert.ToInt32(textBox15.Text);

}

private void textBox16_TextChanged(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(textBox16.Text)) { textBox16.Text = "0"; }
    number16 = Convert.ToInt32(textBox16.Text);

}

private void textBox17_TextChanged(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(textBox17.Text)) { textBox17.Text = "0"; }
    number17 = Convert.ToInt32(textBox17.Text);

}

private void textBox18_TextChanged(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(textBox18.Text)) { textBox18.Text = "0"; }
    number18 = Convert.ToInt32(textBox18.Text);
}
```

```
private void textBox19_TextChanged(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(textBox19.Text)) { textBox19.Text = "0"; }
    number19 = Convert.ToInt32(textBox19.Text);

}

private void textBox20_TextChanged(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(textBox20.Text)) { textBox20.Text = "0"; }
    number20 = Convert.ToInt32(textBox20.Text);

}

private void textBox21_TextChanged(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(textBox21.Text)) { textBox21.Text = "0"; }
    number21 = Convert.ToInt32(textBox21.Text);

}

private void textBox22_TextChanged(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(textBox22.Text)) { textBox22.Text = "0"; }
    number22 = Convert.ToInt32(textBox22.Text);

}

private void textBox23_TextChanged(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(textBox23.Text)) { textBox23.Text = "0"; }
    number23 = Convert.ToInt32(textBox23.Text);

}
```



```
private void textBox24_TextChanged(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(textBox24.Text)) { textBox20.Text = "0"; }
```

```
    number24 = Convert.ToInt32(textBox24.Text);

}


//UInt16[] off = new UInt16[12];


private void timer3_Tick(object sender, EventArgs e)
{


    this.Form2.chart1.Series["Power"].Points.AddXY(count2, sum);

}


private void button28_Click(object sender, EventArgs e)
{
    Form2.Show();
}
int secc = 0;
int Time;
int P_dif;

private void timer4_Tick(object sender, EventArgs e)
{

    P_dif = Pload - 1600;
    Time = (800 / P_dif);
    //Time = 1;

    secc = secc + 1;
    if (secc == Time)
    {
        CAN_WRITE(66); //s1
```

```
}


if (secc == Time + 5)
{

    CAN_WRITE(68);//S2

}

if (secc == Time + 10)
{
    CAN_WRITE(70);//S3

}


if (secc == Time + 15)
{

    CAN_WRITE(72);//S4

}


if (secc == Time + 20)
{

    CAN_WRITE(74);//S5

}


if (secc == Time + 25)
{

    CAN_WRITE(76);//S6
```

```
        }


        if (secc == Time + 30)
        {

            CAN_WRITE(78);//S7

        }


    }
    int[] bufpq = new int[8];
    Int32 Shading;
    private void timer5_Tick(object sender, EventArgs e)
    {
        int iCont = 1;
        // Read one CAN message
        CANUSB.CANMsg msg = new CANUSB.CANMsg();

        while (iCont == 1)
        {
            int rv = CANUSB.canusb_Read(handle, out msg);
            if (CANUSB.ERROR_CANUSB_OK == rv)
            {
                String str = "";
                if (0 != (msg.flags & CANUSB.CANMSG_EXTENDED))
                {
                    str += "EID:0x" + msg.id.ToString("X8") + "  DLC:" + msg.len.ToString()
+ "  ";
                }
                else
                {
                    str += "ID:0x" + msg.id.ToString("X3") + "  DLC:" + msg.len.ToString() +
"  ";
                }
                if (0 != (msg.flags & CANUSB.CANMSG_RTR))
```

```
    {
        str += "RTR";// Remote Request Frame
    }
    else
    {
        for (int i = 0; i < msg.len; i++) // Data Frame
        {
            str += "0x" + ((byte)(msg.data >> i * 8)).ToString("X2") + " ";
            bufpq[i] = (byte)(msg.data >> i * 8);
        }
    }


    if (msg.id == 0x080)
    {
        Pload = Convert.ToInt16(bufpq[1] * 256 + bufpq[0]);
    }
    if (msg.id == 0x088)
    {
        Shading = Convert.ToInt16(bufpq[1] * 256 + bufpq[0]);
    }
    if (Shading == 90)
    {
        timer4.Start();
    }
    if (Shading == 91)
    {
        timer4.Stop();
        secc = 0;
    }


    ReadLabel.Text = str;
}
else if (CANUSB.ERROR_CANUSB_NO_MESSAGE == rv)
{
    iCont = 0;
}
else
```

```
                {
                    iCont = 0;
                    ReadLabel.Text = "Failed to read message.";
                }


            }
        }
    }
}
```

ภาคผนวก ค

การสร้าง Agent โดยใช้ภาษา C#

```
using System;
using System.Windows.Forms;
using System.Text;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using Lawicel;

namespace Lawicel_canusb_Demo
{
    /// <summary>
    /// Summary description for Form1.
    /// </summary>
    public class Form1 : System.Windows.Forms.Form
    {
        static uint handle;
        static uint sentframes;
        static uint receivedframes;
        int[] k = new int[10];
        //ulong data;



        private System.Windows.Forms.Button VersionInfoButton;
        private System.Windows.Forms.Button CloseButton;
        private System.Windows.Forms.Button WriteButton;
        private System.Windows.Forms.Button ReadButton;
        private System.Windows.Forms.Label CloseLabel;
        private System.Windows.Forms.Label WriteLabel;
        private System.Windows.Forms.Label ReadLabel;
        private System.Windows.Forms.Label VersionLabel;
        private System.Windows.Forms.Button StatusButton;
        private System.Windows.Forms.Label StatusLabel;
        private System.Windows.Forms.Timer timer1;
        private System.Windows.Forms.Button OpenButton;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.Label lblsentframes;
```

```
private System.Windows.Forms.Label lblreceivedframes;
private System.Windows.Forms.Label OpenLabel;
private ListBox listBox1;
private Label label3;
private TextBox textBoxID;
private Label label4;
private TextBox P1;
private TextBox Q1;
private TextBox P2;
private TextBox Q2;
private Label label5;
private Label label6;
private Label label7;
private Label label8;
private Label gen2lb;
private TextBox textBox1;
private Label label9;
private Label label10;
private System.ComponentModel.IContainer components;

public Form1()
{
    //
    // Required for Windows Form Designer support
    //
    InitializeComponent();
    //
    // TODO: Add any constructor code after InitializeComponent call
    //
}

/// <summary>
/// Clean up any resources being used.
/// </summary>
protected override void Dispose(bool disposing)
{
    if (disposing)
    {
```

```
            if (components != null)
            {
                components.Dispose();
            }
        }
        base.Dispose(disposing);
    }

    #region Windows Form Designer generated code
    /// <summary>
    /// Required method for Designer support - do not modify
    /// the contents of this method with the code editor.
    /// </summary>
    private void InitializeComponent()
    {
        this.components = new System.ComponentModel.Container();
        System.ComponentModel.ComponentResourceManager resources = new
System.ComponentModel.ComponentResourceManager(typeof(Form1));
        this.VersionInfoButton = new System.Windows.Forms.Button();
        this.OpenButton = new System.Windows.Forms.Button();
        this.OpenLabel = new System.Windows.Forms.Label();
        this.CloseButton = new System.Windows.Forms.Button();
        this.CloseLabel = new System.Windows.Forms.Label();
        this.WriteButton = new System.Windows.Forms.Button();
        this.WriteLabel = new System.Windows.Forms.Label();
        this.ReadButton = new System.Windows.Forms.Button();
        this.ReadLabel = new System.Windows.Forms.Label();
        this.VersionLabel = new System.Windows.Forms.Label();
        this.StatusButton = new System.Windows.Forms.Button();
        this.StatusLabel = new System.Windows.Forms.Label();
        this.timer1 = new System.Windows.Forms.Timer(this.components);
        this.label1 = new System.Windows.Forms.Label();
        this.label2 = new System.Windows.Forms.Label();
        this.lblsentframes = new System.Windows.Forms.Label();
        this.lblreceivedframes = new System.Windows.Forms.Label();
        this.listBox1 = new System.Windows.Forms.ListBox();
        this.label3 = new System.Windows.Forms.Label();
        this.textBoxID = new System.Windows.Forms.TextBox();
```

```
            this.label4 = new System.Windows.Forms.Label();
            this.P1 = new System.Windows.Forms.TextBox();
            this.Q1 = new System.Windows.Forms.TextBox();
            this.P2 = new System.Windows.Forms.TextBox();
            this.Q2 = new System.Windows.Forms.TextBox();
            this.label5 = new System.Windows.Forms.Label();
            this.label6 = new System.Windows.Forms.Label();
            this.label7 = new System.Windows.Forms.Label();
            this.label8 = new System.Windows.Forms.Label();
            this.gen2lb = new System.Windows.Forms.Label();
            this.textBox1 = new System.Windows.Forms.TextBox();
            this.label9 = new System.Windows.Forms.Label();
            this.label10 = new System.Windows.Forms.Label();
            this.SuspendLayout();
            //
            // VersionInfoButton
            //
            this.VersionInfoButton.Enabled = false;
            this.VersionInfoButton.Location = new System.Drawing.Point(16, 88);
            this.VersionInfoButton.Name = "VersionInfoButton";
            this.VersionInfoButton.Size = new System.Drawing.Size(112, 24);
            this.VersionInfoButton.TabIndex = 0;
            this.VersionInfoButton.Text = "canusb_VersionInfo";
            this.VersionInfoButton.Click += new
System.EventHandler(this.VersionInfoButton_Click);
            //
            // OpenButton
            //
            this.OpenButton.Location = new System.Drawing.Point(16, 16);
            this.OpenButton.Name = "OpenButton";
            this.OpenButton.Size = new System.Drawing.Size(112, 24);
            this.OpenButton.TabIndex = 2;
            this.OpenButton.Text = "canusb_Open";
            this.OpenButton.Click += new System.EventHandler(this.OpenButton_Click);
            //
            // OpenLabel
            //
            this.OpenLabel.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;
```

```
this.OpenLabel.Location = new System.Drawing.Point(136, 16);
this.OpenLabel.Name = "OpenLabel";
this.OpenLabel.Size = new System.Drawing.Size(192, 24);
this.OpenLabel.TabIndex = 3;
this.OpenLabel.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
//
// CloseButton
//
this.CloseButton.Enabled = false;
this.CloseButton.Location = new System.Drawing.Point(16, 48);
this.CloseButton.Name = "CloseButton";
this.CloseButton.Size = new System.Drawing.Size(112, 24);
this.CloseButton.TabIndex = 10;
this.CloseButton.Text = "canusb_Close";
this.CloseButton.Click += new System.EventHandler(this.CloseButton_Click);
//
// CloseLabel
//
this.CloseLabel.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;
this.CloseLabel.Location = new System.Drawing.Point(136, 48);
this.CloseLabel.Name = "CloseLabel";
this.CloseLabel.Size = new System.Drawing.Size(192, 24);
this.CloseLabel.TabIndex = 11;
this.CloseLabel.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
//
// WriteButton
//
this.WriteButton.Enabled = false;
this.WriteButton.Location = new System.Drawing.Point(12, 212);
this.WriteButton.Name = "WriteButton";
this.WriteButton.Size = new System.Drawing.Size(112, 24);
this.WriteButton.TabIndex = 6;
this.WriteButton.Text = "canusb_Write";
this.WriteButton.Click += new System.EventHandler(this.WriteButton_Click);
//
// WriteLabel
//
this.WriteLabel.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;
```

```
this.WriteLabel.Location = new System.Drawing.Point(245, 212);
this.WriteLabel.Name = "WriteLabel";
this.WriteLabel.Size = new System.Drawing.Size(271, 24);
this.WriteLabel.TabIndex = 7;
this.WriteLabel.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
//
// ReadButton
//
this.ReadButton.Enabled = false;
this.ReadButton.Location = new System.Drawing.Point(16, 160);
this.ReadButton.Name = "ReadButton";
this.ReadButton.Size = new System.Drawing.Size(112, 24);
this.ReadButton.TabIndex = 8;
this.ReadButton.Text = "canusb_Read";
//
// ReadLabel
//
this.ReadLabel.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;
this.ReadLabel.Location = new System.Drawing.Point(136, 160);
this.ReadLabel.Name = "ReadLabel";
this.ReadLabel.Size = new System.Drawing.Size(384, 24);
this.ReadLabel.TabIndex = 9;
this.ReadLabel.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
//
// VersionLabel
//
this.VersionLabel.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;
this.VersionLabel.Location = new System.Drawing.Point(136, 88);
this.VersionLabel.Name = "VersionLabel";
this.VersionLabel.Size = new System.Drawing.Size(192, 24);
this.VersionLabel.TabIndex = 1;
this.VersionLabel.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
//
// StatusButton
//
this.StatusButton.Enabled = false;
this.StatusButton.Location = new System.Drawing.Point(16, 120);
this.StatusButton.Name = "StatusButton";
```

```
this.StatusButton.Size = new System.Drawing.Size(112, 24);
this.StatusButton.TabIndex = 4;
this.StatusButton.Text = "canusb_Status";
this.StatusButton.Click += new System.EventHandler(this.StatusButton_Click);
//
// StatusLabel
//
this.StatusLabel.BorderStyle = System.Windows.Forms.BorderStyle.Fixed3D;
this.StatusLabel.Location = new System.Drawing.Point(136, 120);
this.StatusLabel.Name = "StatusLabel";
this.StatusLabel.Size = new System.Drawing.Size(192, 24);
this.StatusLabel.TabIndex = 5;
this.StatusLabel.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
//
// timer1
//
this.timer1.Interval = 10;
this.timer1.Tick += new System.EventHandler(this.timer1_Tick);
//
// label1
//
this.label1.ForeColor = System.Drawing.Color.RoyalBlue;
this.label1.Location = new System.Drawing.Point(334, 124);
this.label1.Name = "label1";
this.label1.Size = new System.Drawing.Size(100, 16);
this.label1.TabIndex = 12;
this.label1.Text = "Sent Frames :";
this.label1.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
//
// label2
//
this.label2.ForeColor = System.Drawing.Color.RoyalBlue;
this.label2.Location = new System.Drawing.Point(334, 92);
this.label2.Name = "label2";
this.label2.Size = new System.Drawing.Size(100, 16);
this.label2.TabIndex = 13;
this.label2.Text = "Received Frames :";
this.label2.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
```

```
            //
            // lblsentframes
            //
            this.lblsentframes.Location = new System.Drawing.Point(438, 124);
            this.lblsentframes.Name = "lblsentframes";
            this.lblsentframes.Size = new System.Drawing.Size(64, 16);
            this.lblsentframes.TabIndex = 14;
            this.lblsentframes.Text = "0";
            this.lblsentframes.TextAlign = System.Drawing.ContentAlignment.MiddleLeft;
            //
            // lblreceivedframes
            //
            this.lblreceivedframes.Location = new System.Drawing.Point(438, 92);
            this.lblreceivedframes.Name = "lblreceivedframes";
            this.lblreceivedframes.Size = new System.Drawing.Size(64, 16);
            this.lblreceivedframes.TabIndex = 15;
            this.lblreceivedframes.Text = "0";
            this.lblreceivedframes.TextAlign =
System.Drawing.ContentAlignment.MiddleLeft;
            //
            // listBox1
            //
            this.listBox1.FormattingEnabled = true;
            this.listBox1.Location = new System.Drawing.Point(339, 16);
            this.listBox1.Name = "listBox1";
            this.listBox1.Size = new System.Drawing.Size(115, 56);
            this.listBox1.TabIndex = 16;
            //
            // label3
            //
            this.label3.AutoSize = true;
            this.label3.Location = new System.Drawing.Point(336, 0);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(94, 13);
            this.label3.TabIndex = 17;
            this.label3.Text = "Available adapters";
            //
            // textBoxID
```

```
//
this.textBoxID.Location = new System.Drawing.Point(139, 216);
this.textBoxID.Name = "textBoxID";
this.textBoxID.Size = new System.Drawing.Size(100, 20);
this.textBoxID.TabIndex = 18;
//
// label4
//
this.label4.AutoSize = true;
this.label4.Location = new System.Drawing.Point(172, 200);
this.label4.Name = "label4";
this.label4.Size = new System.Drawing.Size(18, 13);
this.label4.TabIndex = 19;
this.label4.Text = "ID";
//
// P1
//
this.P1.Location = new System.Drawing.Point(51, 276);
this.P1.Name = "P1";
this.P1.Size = new System.Drawing.Size(77, 20);
this.P1.TabIndex = 20;
//
// Q1
//
this.Q1.Location = new System.Drawing.Point(51, 319);
this.Q1.Name = "Q1";
this.Q1.Size = new System.Drawing.Size(77, 20);
this.Q1.TabIndex = 21;
//
// P2
//
this.P2.Location = new System.Drawing.Point(159, 275);
this.P2.Name = "P2";
this.P2.Size = new System.Drawing.Size(80, 20);
this.P2.TabIndex = 22;
//
// Q2
//
```

```
this.Q2.Location = new System.Drawing.Point(159, 319);
this.Q2.Name = "Q2";
this.Q2.Size = new System.Drawing.Size(80, 20);
this.Q2.TabIndex = 23;
//
// label5
//
this.label5.AutoSize = true;
this.label5.Location = new System.Drawing.Point(70, 248);
this.label5.Name = "label5";
this.label5.Size = new System.Drawing.Size(36, 13);
this.label5.TabIndex = 24;
this.label5.Text = "GEN1";
//
// label6
//
this.label6.AutoSize = true;
this.label6.Location = new System.Drawing.Point(172, 248);
this.label6.Name = "label6";
this.label6.Size = new System.Drawing.Size(36, 13);
this.label6.TabIndex = 25;
this.label6.Text = "GEN2";
//
// label7
//
this.label7.AutoSize = true;
this.label7.Location = new System.Drawing.Point(9, 283);
this.label7.Name = "label7";
this.label7.Size = new System.Drawing.Size(14, 13);
this.label7.TabIndex = 26;
this.label7.Text = "P";
//
// label8
//
this.label8.AutoSize = true;
this.label8.Location = new System.Drawing.Point(13, 322);
this.label8.Name = "label8";
this.label8.Size = new System.Drawing.Size(15, 13);
```

```
this.label8.TabIndex = 27;
this.label8.Text = "Q";
//
// gen2lb
//
this.gen2lb.AutoSize = true;
this.gen2lb.ForeColor = System.Drawing.Color.Red;
this.gen2lb.Location = new System.Drawing.Point(360, 301);
this.gen2lb.Name = "gen2lb";
this.gen2lb.Size = new System.Drawing.Size(66, 13);
this.gen2lb.TabIndex = 28;
this.gen2lb.Text = "GEN#2 OFF";
//
// textBox1
//
this.textBox1.Location = new System.Drawing.Point(315, 272);
this.textBox1.Name = "textBox1";
this.textBox1.Size = new System.Drawing.Size(100, 20);
this.textBox1.TabIndex = 29;
//
// label9
//
this.label9.AutoSize = true;
this.label9.Location = new System.Drawing.Point(482, 350);
this.label9.Name = "label9";
this.label9.Size = new System.Drawing.Size(0, 13);
this.label9.TabIndex = 30;
//
// label10
//
this.label10.AutoSize = true;
this.label10.Location = new System.Drawing.Point(265, 275);
this.label10.Name = "label10";
this.label10.Size = new System.Drawing.Size(44, 13);
this.label10.TabIndex = 31;
this.label10.Text = "P_Total";
//
// Form1
```

```
//
this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
this.ClientSize = new System.Drawing.Size(536, 392);
this.Controls.Add(this.label10);
this.Controls.Add(this.label9);
this.Controls.Add(this.textBox1);
this.Controls.Add(this.gen2lb);
this.Controls.Add(this.label8);
this.Controls.Add(this.label7);
this.Controls.Add(this.label6);
this.Controls.Add(this.label5);
this.Controls.Add(this.Q2);
this.Controls.Add(this.P2);
this.Controls.Add(this.Q1);
this.Controls.Add(this.P1);
this.Controls.Add(this.label4);
this.Controls.Add(this.textBoxID);
this.Controls.Add(this.label3);
this.Controls.Add(this.listBox1);
this.Controls.Add(this.lblreceivedframes);
this.Controls.Add(this.lblsentframes);
this.Controls.Add(this.label2);
this.Controls.Add(this.label1);
this.Controls.Add(this.StatusLabel);
this.Controls.Add(this.StatusButton);
this.Controls.Add(this.VersionLabel);
this.Controls.Add(this.ReadLabel);
this.Controls.Add(this.ReadButton);
this.Controls.Add(this.WriteLabel);
this.Controls.Add(this.WriteButton);
this.Controls.Add(this.CloseLabel);
this.Controls.Add(this.CloseButton);
this.Controls.Add(this.OpenLabel);
this.Controls.Add(this.OpenButton);
this.Controls.Add(this.VersionInfoButton);
this.Icon = ((System.Drawing.Icon)(resources.GetObject("$this.Icon")));
this.Name = "Form1";
this.Text = "Control Agent";
```

```
    this.Load += new System.EventHandler(this.Form1_Load);
    this.ResumeLayout(false);
    this.PerformLayout();

}
#endregion

/// <summary>
/// The main entry point for the application.
/// </summary>
[STAThread]
static void Main()
{
    Application.Run(new Form1());
}



private void OpenButton_Click(object sender, System.EventArgs e)
{
    //
    //      To test a diffrent Mask/Filter, change argument 3 & 4 to these instead:
    //
    //      0x1FBD1FFD Acceptance code for accepting only 0x5E8 and 0x7F8
    //      0x1F001F00 Acceptance mask for accepting only 0x5E8 and 0x7F8
    //

    // Open channel at 500 kbps, no filter
    if (listBox1.SelectedIndex == -1)
    {
        handle = CANUSB.canusb_Open(IntPtr.Zero,
                                    CANUSB.CAN_BAUD_1M,
                                    CANUSB.CANUSB_ACCEPTANCE_CODE_ALL,
                                    CANUSB.CANUSB_ACCEPTANCE_MASK_ALL,
                                    CANUSB.CANUSB_FLAG_TIMESTAMP);
    }
    else
    {
```

```
        string device = listBox1.Items[listBox1.SelectedIndex].ToString();
        handle = CANUSB.canusb_Open(device,
                                    CANUSB.CAN_BAUD_1M,
                                    CANUSB.CANUSB_ACCEPTANCE_CODE_ALL,
                                    CANUSB.CANUSB_ACCEPTANCE_MASK_ALL,
                                    CANUSB.CANUSB_FLAG_TIMESTAMP);
    }

    if (handle > 0)
    {
        OpenLabel.Text = "Open OK";
        OpenButton.Enabled = false;
        StatusButton.Enabled = true;
        WriteButton.Enabled = true;
        CloseButton.Enabled = true;
        CloseLabel.Text = "";
        timer1.Enabled = true;

        VersionInfoButton.Enabled = true;
        VersionLabel.Text = "";
    }
    else
    {
        OpenLabel.Text = "Failed to Open CANUSB";
    }
}

private void CloseButton_Click(object sender, System.EventArgs e)
{
    // Close the CANUSB
    int res = CANUSB.canusb_Close(handle);
    if (CANUSB.ERROR_CANUSB_OK == res)
    {
        CloseLabel.Text = "Closed OK";
    }
    else
    {
        CloseLabel.Text = "Failed to Close CANUSB";
```

```
    }

    timer1.Enabled = false;
    OpenButton.Enabled = true;
    OpenLabel.Text = "";
    StatusButton.Enabled = false;
    StatusLabel.Text = "";
    WriteButton.Enabled = false;
    WriteLabel.Text = "";
    ReadButton.Enabled = false;
    ReadLabel.Text = "";
    CloseButton.Enabled = false;
    VersionLabel.Text = "";
    VersionInfoButton.Enabled = false;

}

ulong d = 0;
int n = 0;

private void WriteButton_Click(object sender, System.EventArgs e)
{
    // Send one message with extended id
    CANUSB.CANMsg msg = new CANUSB.CANMsg();
    uint ID, h0, h1, h2;
    ID = Convert.ToUInt32(textBoxID.Text);
    h0 = ID % 10;
    h1 = (ID % 100) / 10;

    h2 = (ID % 1000) / 100;
    msg.id = h2 * 16 * 16 + h1 * 16 + h0;
    msg.len = 2;
    msg.flags = 0x00;
    if (n == 0) { msg.data = 64; n = 1; }
    else if (n == 1) { msg.data = 63; n = 0; }
    d++;
```

```
    int rv = CANUSB.canusb_Write(handle, ref msg);
    if (CANUSB.ERROR_CANUSB_OK == rv)
    {
        WriteLabel.Text = "Message send successfully";
        sentframes++;
    }
    else if (CANUSB.ERROR_CANUSB_TX_FIFO_FULL == rv)
    {
        WriteLabel.Text = "FIFO full. Can't send message.";
    }
    else
    {
        WriteLabel.Text = "Failed to send message.";
    }

}

private void StatusButton_Click(object sender, System.EventArgs e)
{
    int status = CANUSB.canusb_Status(handle);
    StatusLabel.Text = status.ToString();
}

private void VersionInfoButton_Click(object sender, System.EventArgs e)
{
    // Get version info
    StringBuilder buf = new StringBuilder(256);
    int rv = CANUSB.canusb_VersionInfo(handle, buf);
    if (CANUSB.ERROR_CANUSB_OK == rv)
    {
        VersionLabel.Text = buf.ToString();
    }
    else
    {
        VersionLabel.Text = "Failed to get version information";
    }

}
```

```
Int32 P_1, P_2, Q_1, Q_2, Pload, P_3, Q_3;
Int32[] P = new Int32[20];
int con = 0;
int aaa = 0;
int x;
UInt16[] off = new UInt16[6];

UInt16 P_Tot;
private void timer1_Tick(object sender, System.EventArgs e)
{

    int iCont = 1;
    // Read one CAN message
    CANUSB.CANMsg msg = new CANUSB.CANMsg();
    int[] bufpq = new int[8];//buffer for PQ

    while (iCont == 1)
    {
        int rv = CANUSB.canusb_Read(handle, out msg);
        if (CANUSB.ERROR_CANUSB_OK == rv)
        {
            String str = "";
            if (0 != (msg.flags & CANUSB.CANMSG_EXTENDED))
            {
                str += "EID:0x" + msg.id.ToString("X8") + "  DLC:" + msg.len.ToString()
+ "  ";
            }
            else
            {
                str += "ID:0x" + msg.id.ToString("X3") + "  DLC:" + msg.len.ToString() +
"  ";
            }


            if (0 != (msg.flags & CANUSB.CANMSG_RTR))
            {
                str += "RTR";// Remote Request Frame
            }
```

```
                else
                {
                    for (int i = 0; i < msg.len; i++) // Data Frame
                    {
                        str += "0x" + ((byte)(msg.data >> i * 8)).ToString("X2") + " ";
                        bufpq[i] = (byte)(msg.data >> i * 8);
                    }
                }
                if (msg.id == 0x032)
                {
                    P_3 = Convert.ToInt16(bufpq[1] * 256 + bufpq[0]);
                    Q_3 = Convert.ToInt16(bufpq[3] * 256 + bufpq[2]);
                }
                if (msg.id == 0x010)
                {
                    P_1 = Convert.ToInt16(bufpq[1] * 256 + bufpq[0]);
                    Q_1 = Convert.ToInt16(bufpq[3] * 256 + bufpq[2]);
                    P_2 = Convert.ToInt16(bufpq[5] * 256 + bufpq[4]);
                    Q_2 = Convert.ToInt16(bufpq[7] * 256 + bufpq[6]);
                    // Pload = x;
                     Pload = P_1+P_2+P_3;
                     P_Tot = Convert.ToUInt16(Pload);
                     VPload(P_Tot);

                    // P_Tot = Pload - k[0] - k[1] - k[2] - k[3] - k[4]-k[5]-k[6];
                    textBox1.Text = Pload.ToString();



                    if (Pload >= 1000 && con == 0)
                    {
                        can_send2gen(64); con = 1; gen2lb.Text = "GEN#2 ON";
gen2lb.ForeColor = Color.Green;
                    }
                    if (Pload <= 800 && con == 1)
                    {
                        can_send2gen(63); con = 0; gen2lb.Text = "GEN#2 OFF";
gen2lb.ForeColor = Color.Red;
```

```
        }
        if (Pload <= 1600 && aaa == 1)// RPGS
        {

            CAP_RENEW(61);
            aaa = 0;
        }

        if (Pload >= 1800 && aaa == 0)
        {

            CAP_RENEW(62);
            aaa = 1;
        }
        if (Pload > 1600)
        {

            Shading_Load(90);
        }
        if (Pload <= 1600)
        {
            Shading_Load(91);
        }

        P1.Text = (P_1).ToString("N0");
        Q1.Text = (Q_1).ToString("N0");
        P2.Text = (P_2).ToString("N0");
        Q2.Text = (Q_2).ToString("N0");
    }
    ReadLabel.Text = str;
    receivedframes++;
    if (receivedframes > 59999)
        receivedframes = 0;
}
else if (CANUSB.ERROR_CANUSB_NO_MESSAGE == rv)
{
    iCont = 0;
}
```

```
        else
        {
            iCont = 0;
            ReadLabel.Text = "Failed to read message.";
        }
        lblsentframes.Text = sentframes.ToString();
        lblreceivedframes.Text = receivedframes.ToString();

    }
}
ulong buffold;
private void CAN_WRITE(ulong data)
{
    if (buffold != data)
    {

        CANUSB.CANMsg msg = new CANUSB.CANMsg();
        msg.id = 0x030;
        msg.len = 1;
        msg.flags = 0;
        msg.data = data;
        int rv = CANUSB.canusb_Write(handle, ref msg);
        buffold = data;
    }

}
private void CAP_RENEW(ulong data)
{
    CANUSB.CANMsg msg = new CANUSB.CANMsg();
    msg.id = 0x023;
    msg.len = 1;
    msg.flags = 0;
    msg.data = data;
    int rv = CANUSB.canusb_Write(handle, ref msg);
}
private void can_send2gen(uint data)
{                       // Send one message with extended id
```

```
    CANUSB.CANMsg msg2 = new CANUSB.CANMsg();

    msg2.id = 0x050;     //send to gen
    msg2.len = 1;
    msg2.flags = 0x00;
    msg2.data = data;
    int rv = CANUSB.canusb_Write(handle, ref msg2);

}
private void Shading_Load(ulong data)
{
    CANUSB.CANMsg msg = new CANUSB.CANMsg();
    msg.id = 0x088;
    msg.len = 2;
    msg.flags = 0;
    msg.data = data;
    int rv = CANUSB.canusb_Write(handle, ref msg);
}
private void VPload(ulong data)
{
    CANUSB.CANMsg msg = new CANUSB.CANMsg();
    msg.id = 0x080;
    msg.len = 2;
    msg.flags = 0;
    msg.data = data;
    int rv = CANUSB.canusb_Write(handle, ref msg);
}
private void Form1_Load(object sender, EventArgs e)
{
    StringBuilder buf = new StringBuilder(32);
    int i, cnt;


    // Fill the listbox with data
    cnt = CANUSB.canusb_getFirstAdapter(buf, 32);
    listBox1.Items.Add(buf);
    for (i = 1; i < cnt; i++)
    {
```

```
            if (CANUSB.canusb_getNextAdapter(buf, 32) > 0)
            {
                listBox1.Items.Add(buf);
            }
        }
    }
}
```

ภาคผนวก ง

บทความทางวิชาการ

ระบบควบคุมไมโครกริดโดยใช้มัลติเอเจนต์

AGENT-BASED MICROGRID CONTROL SYSTEM

ศิโรจน์ น้อยชุ้ย ศุภณัฐ สรนันท์ ศุภพล ทาบึงการ และสกล เดชโภคิน

ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ถนนฉลองกรุง แขวงลำปลาทิว เขตลาดกระบัง กรุงเทพมหานคร 10520 โทรศัพท์ 0-2739-2478

บทคัดย่อ

โครงงานนี้นำเสนอการพัฒนาระบบการจัดการพลังงานสำหรับไมโครกริดโดยใช้เอเจนต์ โดยมีวัตถุประสงค์เพื่อรักษาสมดุลระหว่างความต้องการการใช้ไฟฟ้ากับปริมาณการผลิต เพื่อเพิ่มคุณภาพและความน่าเชื่อถือของระบบ โดยไมโครกริดที่จะศึกษาประกอบด้วย เส้นภาระไฟฟ้า และเครื่องกำเนิดไฟฟ้า 3 หน่วยผลิต โดยมีการเขียนโปรแกรมสำหรับการควบคุมเอเจนต์ด้วยโปรแกรม C# เพื่อเชื่อมโยงระหว่างเอเจนต์เครื่องกำเนิดไฟฟ้าทั้ง 3 หน่วยผลิตที่ควบคุมโดย DSP ร่วมกับ Arduino ซึ่งทำหน้าที่ในการควบคุมการเปิด-ปิดภาระทางไฟฟ้าผ่านรีเลย์บอร์ด และมีการแสดงผลการกระทำของภาระทางไฟฟ้าออกทางหน้าจอโปรแกรมที่เขียนขึ้นด้วยโปรแกรม C# โดยการเชื่อมต่อทั้งหมดของระบบจะส่งข้อมูลผ่านทาง CANUSB ซึ่งเป็นอุปกรณ์ที่ใช้ในการรับ-ส่งข้อมูล จากผลการทดลองดังกล่าวได้ผลการทดลองว่า เอเจนต์สามารถจัดสรรพลังงานได้อย่างเหมาะสมและควบคุมให้ระบบดำเนินไปอย่างมีเสถียรภาพอีกด้วย

คำสำคัญ:เอเจนต์, ไมโครกริด, C#

Abstract

This project develops energy management system for a microgrid using agent technology. The objective of the energy management system is to balance the load demand with the generation supply in order to increase quality and reliability of the system. The microgrid in this study consists of load curve and three generators. Using C# for connect the agent of three generators which are controlled by DSP and adruino (controlling turn on and turn off load in relay board) and show the result in the program which appear on The monitor programming by C# and everything in the microgrid system are transmitted and received data by CANUSB. The experimental show the result that agent can manage the energy properly and control system stability properly.

Keyword: Agent, Microgrid, C#

1.บทนำ

ในปัจจุบันเทคโนโลยีการผลิตไฟฟ้าได้พัฒนาไปมาก มีการผลิตไฟฟ้าแบบกระจายโดยใช้แหล่งพลังงานหมุนเวียนซึ่งเป็นระบบที่มีขนาดเล็ก เรียกว่า ไมโครกริด (Microgrid) โดยขนาดของไมโครกริดจะมีขนาดใกล้เคียงกับภาระทางไฟฟ้าของไมโครกริดนั้น จึงจำเป็นต้องมีระบบการจัดการพลังงานสำหรับไมโครกริดเมื่อไมโครกริดอยู่ในสภาวะต่างๆ เพื่อให้ระบบมีเสถียรภาพและความน่าเชื่อถือ ในความเป็นจริงแหล่งกำเนิดพลังงานของไมโครกริดนั้นอาจจะกระจายตัวอยู่ไกลกันและมีจำนวนมาก จึงจำเป็นต้องมีระบบควบคุมที่สามารถใช้งานร่วมกันได้อย่างเหมาะสม

จากรูปแบบการดำเนินงานไมโครกริด จึงได้มีการนำเทคโนโลยีเอเจนต์ในโปรแกรม C# มาประยุกต์ใช้ในการควบคุมหน่วยการผลิตของไมโครกริด ซึ่งสามารถติดต่อสื่อสารกันได้อย่างอิสระผ่านเครือข่าย CANUSB และตัดสินใจได้ตามโปรแกรมที่เขียนไว้

2. โครงสร้างและหลักการของไมโครกริด

การผลิตพลังงานไฟฟ้าจากแหล่งพลังงานหมุนเวียนกำลังไฟฟ้าที่ผลิตได้จะมีปริมาณที่ผันผวนเป็นอย่างมาก จึงทำให้เกิดความกังวลเกี่ยวกับผลกระทบในแง่ลบทั้งในด้านคุณภาพไฟฟ้าและความน่าเชื่อถือของระบบในกรณีที่มีการนำแหล่งกำเนิดไฟฟ้าเหล่านี้จำนวนมากเข้ามาต่อกับระบบไฟฟ้า (Grid Connected Distributed Generation) จึงได้มีการนำเสนอแนวทางการผลิตและส่งจ่ายไฟฟ้าภายในพื้นที่สำหรับระบบไฟฟ้ากำลังขนาดเล็กโดยใช้แหล่งกำเนิดไฟฟ้าขนาดเล็ก เรียกว่า แหล่งกำเนิดไฟฟ้า ณ จุดใช้งาน (Distributed Energy Resource: DER) เพื่อปรับปรุงความน่าเชื่อถือของระบบและตอบสนองการเปลี่ยนแปลงที่ตามมาจากการเปิดเสรีด้านพลังงานไฟฟ้า เรียกว่าแนวคิดไมโครกริด

โดยแนวคิดไมโครกริดจะถูกมองเป็นระบบควบคุมหน่วยหนึ่งจากระบบไฟฟ้ากำลังหลัก และจะไม่สร้างปัญหาเช่นการกระเพื่อมของแรงดันไฟฟ้า การแกว่งไกวของความถี่ไฟฟ้า แถมยังสามารถช่วยปรับปรุงคุณภาพของระบบเช่น การให้บริการคุณภาพ

ไฟฟ้าชั้นเยี่ยม (Premium Power) การรักษาระดับแรงดันไฟฟ้า (Voltage Support) เป็นต้น

3. เอเจนต์ (Agent System)

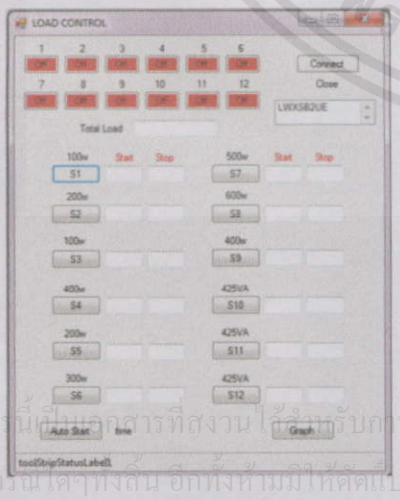
เอเจนต์ (Agent) เป็นระบบซอฟแวร์ที่มีอิสระในการทำงาน สามารถประมวลผลและสื่อสารให้เป็นไปอย่างอัตโนมัติ สามารถรับรู้และตอบสนองต่อสิ่งแวดล้อมได้ เอเจนต์เปรียบเสมือนเป็นตัวแทนผู้ใช้งาน ซึ่งจะสามารถทำแทนได้ดีหรือไม่ ขึ้นอยู่กับความฉลาดของโปรแกรมเอเจนต์ ดังนั้น การพัฒนาโปรแกรมเอเจนต์จะต้องมีการออกแบบให้โปรแกรมมีความฉลาดพอที่จะทำหน้าที่แทนผู้ใช้ได้อย่างมีประสิทธิภาพ

คุณสมบัติพื้นฐานของเอเจนต์ มีดังนี้

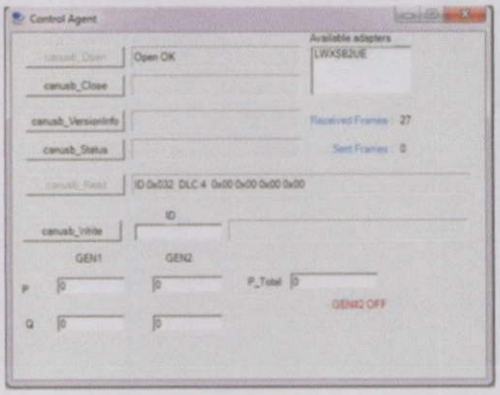
1. สามารถทำงานได้อัตโนมัติโดยไม่จำเป็นต้องติดต่อโดยตรงกับผู้ใช้ตลอดเวลาและมีกลไกบางอย่างในการควบคุมการกระทำและสถานะของเอเจนต์
2. ติดต่อกับเอเจนต์อื่นผ่านภาษาในการติดต่อสื่อสารระหว่างเอเจนต์
3. สามารถทำการตอบสนองเมื่อมีการเปลี่ยนแปลงเกิดขึ้น
4. สามารถริเริ่มแสดงพฤติกรรมเพื่อทำงานตามเป้าหมาย

4. C#

ภาษาC# เป็นภาษาโปรแกรมแบบหลายโมเดล ที่ใช้ระบบชนิดข้อมูลแบบรัดกุมและสนับสนุนการเขียนโปรแกรมเชิงคำสั่ง การเขียนโปรแกรมเชิงประกาศ การเขียนโปรแกรมเชิงฟังก์ชัน การเขียนโปรแกรมเชิงกระบวนการ การเขียนโปรแกรมเชิงวัตถุ และการเขียนโปรแกรมเชิงส่วนประกอบ พัฒนาเริ่มแรกโดยบริษัทไมโครซอฟท์ เพื่อทำงานบน .NET Framework และมีรากฐานมาจากภาษา C++ ภาษา Java และภาษา Eiffel โดยมีจุดมุ่งหมายให้เป็นภาษาสมัยใหม่ที่ไม่ซับซ้อน ใช้งานได้ทั่วไป และเป็นเชิงวัตถุเป็นหลัก



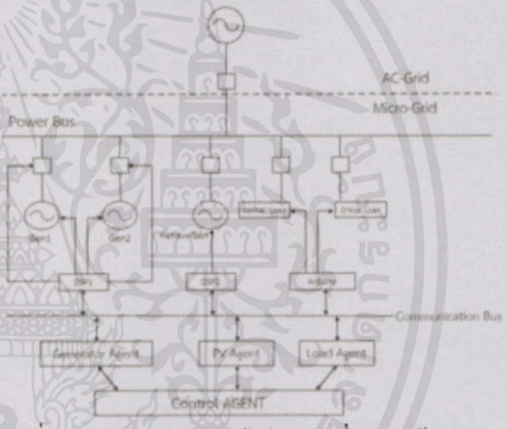
รูปที่ 1 ชุดควบคุมโหลดที่สร้างโดย C#



รูปที่ 2 Agent ที่สร้างโดย C#

5. การจำลองระบบไมโครกริด

ระบบไมโครกริดที่จำลองเป็นระบบแบบแยกอิสระ(Stand-alone) แรงดันต่ำ 380V เป็นลักษณะแบบ Network ที่ควบคุมด้วย Agent ผ่าน CANUSB ดังรูปที่ 2



รูปที่ 3 แบบจำลองของระบบไมโครกริดที่ควบคุมด้วย Agent

ตารางที่ 1 ข้อมูลของเครื่องกำเนิดไฟฟ้าในไมโครกริด

Generator Type	Capacity KVA	จำนวน (เครื่อง)
Generator 1	2	1
Generator 2	2	1
Solar Photovoltaic (PV)	2	1

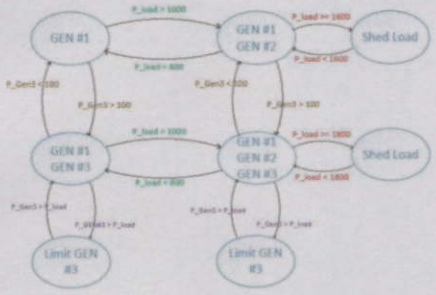
โดยพิจารณาเหตุการณ์ ดังต่อไปนี้

1. เมื่อต้องการดูการ Droop-control ของแหล่งกำเนิดที่ขนานกัน

2. เมื่อต้องการให้เอเจนต์ตัดสินใจว่าควรทำอย่างไรไม่ให้กำลังไฟฟ้าไหลย้อนกลับเข้าสู่แหล่งกำเนิด เมื่อเกิดกรณีแหล่งกำเนิดผลิตกำลังไฟฟ้าเกินกว่าความต้องการของโหลด

3. เมื่อต้องการตัดและเชื่อมต่อโหลดโดยอัตโนมัติ (Shading Load)

6. ผลการจำลอง



รูปที่ 4 State Diagram แสดงขั้นตอนการทำงานของ Agent

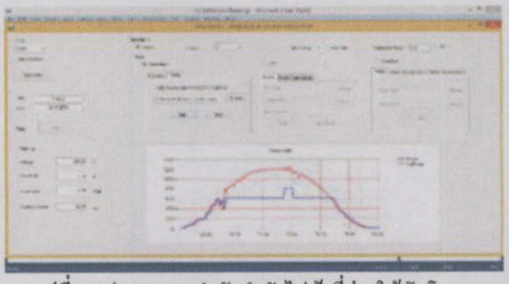
<u>กรณีที่ 1</u> เมื่อต้องการดูการ Droop-control ของแหล่งกำเนิดที่ขนานกัน

รูปที่ 5 กราฟแสดงการดรูประหว่าง Generator1 กับ Generator2

รูปที่ 6 กราฟแสดงค่า Power ของโหลดที่ใช้ในแต่ละวัน

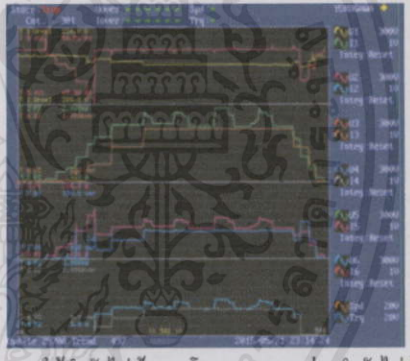
<u>กรณีที่ 2</u> เมื่อต้องการให้เอเจนต์ตัดสินใจว่าควรทำอย่างไรไม่ให้กำลังไฟฟ้าไหลย้อนกลับเข้าสู่แหล่งกำเนิด เมื่อเกิดกรณีแหล่งกำเนิดผลิตกำลังไฟฟ้าเกินกว่าความต้องการของโหลด

รูปที่ 7 การจ่ายกำลังไฟฟ้าของ Generator ทั้ง 3 ตัว



รูปที่ 8 รูปแสดงการจำกัดกำลังไฟฟ้าที่จ่ายให้กับโหลด

<u>กรณีที่ 3</u> การตัดและเชื่อมต่อโหลดโดยอัตโนมัติ (Shading Load) ในกรณีนี้ Agent จะสั่งให้ตัด Load ออกจากระบบ เมื่อ Generator ทั้ง 2 ตัว ไม่สามารถจ่ายโหลดได้อย่างเพียงพอ ซึ่ง Agent จะทำการตัดสินใจว่าจะตัดโหลดที่ไม่จำเป็น(Non-Critical Load) ออกจากระบบ โดยจะยังคงโหลดที่จำเป็น(Critical Load) ไว้อยู่ เพื่อรักษาเสถียรภาพของระบบและจะทำการต่อโหลดที่ตัดออกไปเข้ามาในระบบอีกครั้งเมื่อ Generator ทั้ง 2 ตัว มีความสามารถในการจ่ายโหลดมากขึ้นจนเพียงพอในการจ่ายให้กับโหลดเหล่านั้น



รูปที่ 9 การใช้กำลังไฟฟ้าของโหลดและการจ่ายกำลังไฟฟ้าของ Generator1 กับ Generator2

7. สรุป

โครงงานนี้ถูกนำมาวิจัยเพื่อศึกษาและพัฒนาในเรื่องของเทคนิคการควบคุมในการทำงานของระบบไมโครกริด ซึ่งจะเป็นแนวโน้มใหม่ของเครือข่ายของระบบไฟฟ้าในอนาคต ทั้งนี้ไมโครกริดที่มีระบบควบคุมจะสามารถรักษาเสถียรภาพของตนเองในการที่จะจ่ายโหลดได้อย่างมีประสิทธิภาพ โดยปราศจากปัญหาในโหมด Stand-alone ในยามที่แหล่งจ่ายการไฟฟ้าเกิดปัญหาขึ้น โดยโครงงานนี้จะแบ่งการศึกษาออกเป็นสองส่วนหลักๆ คือ การจำลองระบบไมโครกริดในโหมด Stand-alone และการเขียนโปรแกรมให้ Agent ทำงานแทนคนในการดูแลระบบไมโครกริดให้มีความฉลาดและเสถียรภาพมากที่สุด ผ่านโปรแกรม Microsoft Visual Studio 2013 ด้วยภาษา C# ซึ่งในการทดลองจะแบ่งออกเป็น 3 กรณีด้วยกันคือ กรณีที่1 Droop Control กรณีที่2 จำกัดการจ่ายโหลดของ PV-gen (Capture

PV-gen) และ กรณีที่ 3 การตัดและเชื่อมต่อโหลดโดยอัตโนมัติ (Shading Load)

สำหรับการศึกษาและพัฒนาการจำลองระบบไมโครกริดในโหมด Stand-alone ของระบบไฟฟ้าสามเฟสและเฟสเดียวของแหล่งจ่ายสองตัวที่มีพารามิเตอร์เดียวกันนั้นจะทำการศึกษาโดยใช้โปรแกรม Matlab/Simulink เพื่อศึกษาดูว่าระบบไมโครกริดนั้นมีลักษณะอย่างไร และทำงานอย่างไร เพื่อที่จะต่อยอดไปสู่การพัฒนาให้ระบบไมโครกริดธรรมดากลายไปเป็นระบบสมาร์ทไมโครกริดโดยใช้ Agent ในการควบคุมแบบอัตโนมัติ

สำหรับผลการทดลองนั้นจะมี 3 กรณีคือ กรณีที่1 Droop Control พบว่าผลลัพธ์ที่ออกมานั้นค่อนข้างเป็นที่น่าพอใจ กล่าวคือ เมื่อเรากำหนดสัดส่วนการแชร์โหลด Droop Coefficient ของมุมแรงดันและแรงดันไว้เท่ากันที่ 1:1 แล้ว พบว่ามีการแชร์อัตราส่วนกำลังไฟฟ้าจริง และ กำลังไฟฟ้ารีแอคทีฟ ที่อัตราส่วนประมาณ 1:1 ภายใต้ระยะเวลาเข้าสู่สภาวะคงตัวที่เหมาะสม และมีและมีการคงแรงดันไว้ที่ 220V และการดึงความถี่กลับเข้าสู่ 50 Hz ซึ่งเป็นการควบคุมแบบ Secondary Control ของระบบไมโครกริดในการรักษาเสถียรภาพของการทำงาน

กรณีที่2 จำกัดการจ่ายโหลดของ PV-gen (Capture PV-gen) พบว่าการทำงานของ Agent นั้นยังมีความฉลาดที่จะจำกัดการจ่ายกำลังไฟฟ้าของ PV-gen เข้ามาในระบบไมโครกริดในช่วงที่มีการใช้โหลดน้อย เพื่อป้องกันการเกิดกำลังไฟฟ้าเกินความต้องการของโหลด และป้องกันกำลังไฟฟ้านั้นไหลกลับเข้าสู่ Generator1 และ Generator2 เพราะจะเป็นสาเหตุให้ Generator1 และ Generator2 เสียหายได้

กรณีที่ 3 การตัดและเชื่อมต่อโหลดโดยอัตโนมัติ (Shading Load) พบว่า Agent สามารถสั่งให้มีการตัดโหลดที่ไม่จำเป็น(Non-Critical Load) ออกจากระบบ โดยจะยังคงโหลดที่จำเป็น(Critical Load) ไว้อยู่ เพื่อรักษาเสถียรภาพของระบบและจะทำการต่อโหลดที่ตัดออกไปเข้ามาในระบบอีกครั้งเมื่อ Generator ทั้ง 3 ตัว มีความสามารถในการจ่ายโหลดมากขึ้นจนเพียงพอในการจ่ายให้กับโหลดเหล่านั้น ได้อย่างมีประสิทธิภาพอีกเช่นกัน

เอกสารอ้างอิง

[1] นายจิราวุฒิ กู่สาร, นายณัฐพล นันทวิชาวุธสาร, นายณัฐพล ป้อมป้องกัน, "ระบบจัดการพลังงานสำหรับไมโครกริดโดยเอเจนท์", ปริญญานิพนธ์วิศวกรรมศาสตรบัณฑิต, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, ปีการศึกษา 2554

[2] นายพยุงศักดิ์ แซ่หลิ่ว, นายพงศ์ภรณ์ เตชะชลประเสริฐ, นายไพโรจน์ คอนเอม, นายมหาราช ราชสีห์, "โครงการสาธิตไมโครกริดของศูนย์นวัตกรรมพลังงาน (CInES Microgrid Demonstration Operation and Control)", ปริญญานิพนธ์วิศวกรรมศาสตรบัณฑิต, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, ปีการศึกษา 2555

[3] นายธนกฤต กิตติวรารัตน์, นายปณต เชิดชูเหล่า, นายพัฒนกิต ชูติกุล, "ชุดจำลองการกำเนิด กำลังไฟฟ้าพลังงานทดแทน", ปริญญานิพนธ์วิศวกรรมศาสตรบัณฑิต, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, ปีการศึกษา 2555

[4] นายวัจเรจ หน่อเทพ, นายศรายุทธ จิตประสงค์, นายศันสนะ ตรีอาทิตย์โยธิน, นายศิรสิทธิ์ พิริยะคุณธร"ชุดจำลองการกำเนิดกำลังไฟฟ้าพลังงานทดแทน", ปริญญานิพนธ์วิศวกรรมศาสตรบัณฑิต, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, ปีการศึกษา 2556

[5] นายพสิษฐ์ มณีทิพย์, นายพัทธพล เรืองเดชวรชัย, นายพิรุณ แสงกุล, นายภูชกร เปลี่ยนเชาว์"ชุดจำลองการกำเนิดกำลังไฟฟ้าพลังงานทดแทน", ปริญญานิพนธ์วิศวกรรมศาสตรบัณฑิต, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, ปีการศึกษา 2556

ประวัติผู้เขียน

นาย ศิโรจน์ น้อยชุ้ย
วัน/เดือน/ปีเกิด 17 มกราคม 2535
ที่อยู่ 73 หมู่1 ต.ฝายนาแซง อ.หล่มสัก จ. เพชรบูรณ์ 67110
Email: boonsong966@gmail.com

นาย ศุภณัฐ สรนันท์
วัน/ เดือน/ ปีเกิด 11 มีนาคม 2536
ที่อยู่ 203/7 ซอยลาดพร้าว 41 เขตห้วยขวาง แขวง สามเสนนอก กทม 10310
Email: princezaa_guy@hotmail.com

นาย ศุภพล ทาบึงการ
วัน/ เดือน/ ปีเกิด 24 ธันวาคม 2535
ที่อยู่ 69 หมู่4 ต.ดอนแตง อ.ขาณุวรลักษบุรี จ.กำแพงเพชร 62140
Email: Supapol33@gmail.com

นาย สกล เดชโภคิน
วัน/เดือน/ปีเกิด 30 มีนาคม 2535
ที่อยู่ 49/452 หมู่4 แขวงคลองกุ่ม เขตบึงกุ่ม กรุงเทพฯ 10240
Email: sakol3488@gmail.com

ประวัติผู้เขียน

นาย ศิโรจน์ น้อยขุ้ย
วัน/เดือน/ปีเกิด 17 มกราคม 2535
ที่อยู่ 73 หมู่1 ต.ฝายนาแซง อ.หล่มสัก จ.เพชรบูรณ์ 67110
สำเร็จการศึกษาระดับมัธยมศึกษาตอนปลายจากโรงเรียนหล่มสักวิทยาคม
Email: boonsong966@gmail.com

นาย ศุภณัฐ สรนันท์
วัน/ เดือน/ ปีเกิด 11 มีนาคม 2536
ที่อยู่ 203/7 ซอยลาดพร้าว41 เขตห้วยขวาง แขวงสามเสนนอก กทม. 10310
สำเร็จการศึกษาระดับมัธยมศึกษาตอนปลายจากโรงเรียนนวมินทราชินูทิศบดินทรเดชา
Email: princezaa_guy@hotmail.com

นาย ศุภพล ทาบึงการ
วัน/ เดือน/ ปีเกิด 24 ธันวาคม 2535
ที่อยู่ 69 หมู่4 ต.ดอนแตง อ.ขาณุวรลักษบุรี จ.กำแพงเพชร 62140
สำเร็จการศึกษาระดับมัธยมศึกษาตอนปลายจากโรงเรียนขาณุวิทยา
Email: Supapol33@gmail.com

นาย สกล เดชโภคิน
วัน/เดือน/ปีเกิด 30 มีนาคม 2335
ที่อยู่ 49/452 หมู่4 แขวงคลองกุ่ม เขตบึงกุ่ม กรุงเทพฯ 10240
สำเร็จการศึกษาระดับมัธยมศึกษาตอนปลายจากโรงเรียนสุขุมนวพันธ์อุปถัมภ์
Email: sakol3488@gmail.com