ระบบภูมิสารสนเทศเพื่อการบริหารจัดการสมาร์ทมิเตอร์
GIS FOR SMART METER MANAGEMENT

กิตติพงศ์ จิตรศรีศักดา

กิตติวัฒน์ พงษ์ศรีสุภกร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2556

ระบบภูมิสารสนเทศเพื่อการบริหารจัดการสมาร์ทมิเตอร์

GIS FOR SMART METER MANAGEMENT

กิตติพงศ์ จิตรศรีศักดา
กิตติวัฒน์ พงษ์ศรีศุภกร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2556

ปริญญานิพนธ์ปีการศึกษา 2556

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบภูมิสารสนเทศเพื่อการบริหารจัดการสมาร์ทมิเตอร์

GIS FOR SMART METER MANAGEMENT

ผู้จัดทำ

1. นายกิตติพงศ์ จิตรศรีศักดา รหัสนักศึกษา 53010112
2. นายกิตติวัฒน์ พงษ์ศรีศุภกร รหัสนักศึกษา 53010121

..... อาจารย์ที่ปรึกษา

(ผู้ช่วยศาสตราจารย์ ดร. วิศิษฏ์ หิรัญกิตติ)

ระบบภูมิสารสนเทศเพื่อการบริหารจัดการสมาร์ทมิเตอร์

นาย กิตติพงศ์	จิตรศรีศักดา	53010112
นาย กิตติวัฒน์	พงษ์ศรีศุภกร	53010121
ผศ.ดร. วิศิษฏ์	หิรัญกิตติ	อาจารย์ที่ปรึกษา

ปีการศึกษา 2556

บทคัดย่อ

สมาร์ทมิเตอร์เป็นอุปกรณ์ที่ใช้วัดปริมาณการใช้ไฟฟ้าที่จำเป็นในอนาคตอันใกล้เมื่อระบบจ่ายไฟฟ้ากลายเป็น Smart Grid การทำงานของสมาร์ทมิเตอร์ต้องอาศัยโครงข่ายการสื่อสารซึ่งส่งผลต่อกระบวนการติดตั้งและตรวจสอบสถานการณ์ทำงานของสมาร์ทมิเตอร์ ทำให้มีความจำเป็นที่จะต้องมีการนำเอาระบบภูมิสารสนเทศมาใช้ในการบริหารจัดการสมาร์ทมิเตอร์เพื่อช่วยในการติดตั้ง อ่านค่าจากสมาร์ทมิเตอร์ และการระบุตำแหน่งของสมาร์ทมิเตอร์ในโครงข่ายการสื่อสาร

โดยระบบภูมิสารสนเทศที่พัฒนาขึ้นนี้เป็นเว็บแอพพลิเคชั่นซึ่งประกอบด้วย 2 ส่วนคือ Server-side Application และ Client-side Application ซึ่งทั้งคู่พัฒนาด้วยซอร์ฟแวร์โอเพ่นซอร์ส ในส่วนของ Server-side Application อาศัย Mapserver สำหรับการสร้างภาพแผนที่ PostgreSQL และ PostGIS สำหรับการจัดการฐานข้อมูลเชิงพื้นที่ ส่วนที่พัฒนา Web Application ใช้ Python และ Django ในส่วนของ Client-side Application พัฒนาด้วย HTML5, JavaScript และ library ของ JavaScript สำหรับแสดงแผนที่บนหน้าเว็บเบราว์เซอร์ที่ชื่อ OpenLayers

GIS FOR SMART METER MANAGEMENT

Mr. Kittipong	Jitsrisakda	53010112
Ms. Kittiwat	Phongsrisuphakorn	53010121
Asst. Prof. Dr. Visit	Hirankitti	Advisor

Academic Year 2013

ABSTRACT

In the near future a smart meter will be a device for measuring amount of electricity usage when electricity distribution system become a smart grid. Functionality of Smart Meter requires a communications network which affects the installation process and meter monitoring. There is a need to use geographic information system for management of smart meter such as installation, meter reading, and locating of smart meter in communication network.

The geographic information system in this project is developed using open source software. It is a web application which consists of server-side application and client-side application. The server-side application is developed using Mapserver for creating maps, PostgreSQL and PostGIS for managing spatial databases and Python together with Django for creating server-side applications. The client-side application is developed using HTML5, JavaScript and OpenLayers JavaScript library, the latter is used for viewing map on a web browser.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จได้อย่างสมบูรณ์ด้วยการให้คำปรึกษา คำแนะนำและความกรุณาจาก อาจารย์วิศิษฏ์ หิรัญกิตติ อาจารย์ที่ปรึกษาที่คอยให้คำปรึกษา ชี้แนะและให้ความช่วยเหลือในทุกๆ ด้านจนทุกอย่างสำเร็จไปได้ด้วยดี

ขอขอบคุณอาจารย์ภาควิชาวิศวกรรมคอมพิวเตอร์ทุกท่านที่คอยถ่ายทอดความรู้ อบรมสั่งสอนและให้คำปรึกษาในเรื่องต่างๆ

ขอขอบพระคุณบิดา มารดาและครอบครัวที่คอยเป็นกำลังใจและให้การสนับสนุนอย่างเต็มที่ทำให้สามารถทำปริญญานิพนธ์ฉบับนี้สำเร็จไปได้ด้วยดี

ขอขอบคุณเพื่อนๆ ทุกคนที่คอยให้ความช่วยเหลือในหลายๆด้าน ให้กำลังใจตลอดจนคำแนะนำดีๆ เสมอมา

สุดท้ายนี้คุณความดีประโยชน์และคุณค่าที่เกิดจากปริญญานิพนธ์ฉบับนี้ ผู้วิจัยขอมอบแด่บิดา มารดา ครูอาจารย์ ตลอดจนผู้มีพระคุณทุกท่าน

นายกิตติพงศ์ จิตรศรีศักดา

นายกิตติวัฒน์ พงษ์ศรีศุภกร

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญรูป

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของโครงงาน

สมาร์ทมิเตอร์เป็นอุปกรณ์ที่ใช้วัดปริมาณการใช้ไฟฟ้าที่จำเป็นในอนาคตอันใกล้เมื่อระบบจำหน่ายไฟฟ้ากลายเป็น Smart Grid การทำงานของสมาร์ทมิเตอร์ต้องอาศัยโครงข่ายการสื่อสารซึ่งส่งผลต่อกระบวนการติดตั้งและตรวจสอบสถานการณ์ทำงานของสมาร์ทมิเตอร์ ทำให้มีความจำเป็นที่จะต้องมีการนำเอาระบบภูมิสารสนเทศมาใช้ในการบริหารจัดการสมาร์ทมิเตอร์เพื่อช่วยในการติดตั้ง การอ่านค่าไฟฟ้าจากสมาร์ทมิเตอร์ การค้นหามิเตอร์ และการระบุตำแหน่งของสมาร์ทมิเตอร์ในโครงข่ายการสื่อสาร ซึ่งจะช่วยให้ผู้ให้บริการจำหน่ายไฟฟ้าสามารถรับรู้และปรับปรุงประสิทธิภาพการให้บริการได้อย่างรวดเร็ว

โดยระบบภูมิสารสนเทศที่พัฒนาขึ้นนี้เป็นเว็บแอพพลิเคชั่นซึ่งประกอบด้วย 2 ส่วนคือ Server-side Application และ Client-side Application ซึ่งทั้งคู่นั้นพัฒนาโดยใช้ซอร์ฟแวร์โอเพ่นซอร์ส ในส่วนของ Server-side Application ใช้ Mapserver สำหรับการสร้างภาพแผนที่ PostgreSQL และ PostGIS สำหรับการจัดการฐานข้อมูลเชิงภูมิศาสตร์ ส่วนที่พัฒนา Web Application ใช้ Python และ Django ในส่วนของ Client-side Application พัฒนาด้วย HTML5, JavaScript และ library ของ JavaScript สำหรับแสดงแผนที่บนหน้าเว็บที่มีชื่อว่า OpenLayers

1.2 วัตถุประสงค์ของโครงงาน

1) เพื่อพัฒนาระบบภูมิสารสนเทศด้วยซอฟท์แวร์โอเพนซอร์ส
2) เพื่อพัฒนาระบบให้บริการแผนที่ สำหรับเป็นเครื่องมือในการบริหารจัดการสมาร์ทมิเตอร์
3) เพื่อเป็นบริการสำหรับติดตั้งและซ่อมบำรุงสมาร์ทมิเตอร์
4) เพื่อมอนิเตอร์สถานการณ์ทำงานของสมาร์ทมิเตอร์และการระบุตำแหน่งรวมทั้งค้นหาสมาร์ทมิเตอร์
5) เพื่อตรวจสอบปริมาณการใช้พลังงานไฟฟ้าเป็นจุดหรือเป็นพื้นที่บนแผนที่

1.3 ขอบเขตของโครงงาน

1) สร้างแผนที่จากข้อมูลแผนที่
2) นำแผนที่มาช่วยในการติดตั้งสมาร์ทมิเตอร์
3) สร้างระบบค้นหาสถานที่จากแผนที่
4) บอกข้อมูลปริมาณการใช้พลังงานไฟฟ้าจากสมาร์ทมิเตอร์
5) ระบุตำแหน่งปัจจุบันของผู้ใช้งาน

1.4 วิธีการดำเนินการ

1) ศึกษาการสร้างระบบภูมิสารสนเทศ
   - ศึกษาทฤษฎีต่างๆ ที่เกี่ยวข้องกับภูมิสารสนเทศ
   - ศึกษาวิธีการพัฒนา Web application ด้วย HTML5 JavaScript และ CSS
   - ศึกษา Postgresql และ PostGIS สำหรับสร้างฐานข้อมูลเชิงพื้นที่
   - ศึกษา Django เพื่อการจัดการระบบฐานข้อมูลเชิงพื้นที่
   - ศึกษา MapServer เพื่อนำข้อมูลจากฐานข้อมูลมาสร้างรูปภาพแผนที่
   - ศึกษา OpenLayers เพื่อใช้ในการแสดงแผนที่บนเว็บเบราว์เซอร์
   - ศึกษาการส่งข้อมูลจากสมาร์ทมิเตอร์มายังเซิร์ฟเวอร์โดยผ่าน Socket
2) วิเคราะห์และออกแบบระบบระบบภูมิสารสนเทศ
3) วางแผนการพัฒนาระบบภูมิสารสนเทศ
4) ติดตั้งซอฟท์แวร์และพัฒนาระบบที่ได้วิเคราะห์และออกแบบไว้
5) ปรับปรุงแก้ปัญหาส่วนต่างๆ ของระบบ เพื่อทำให้ระบบทั้งหมดสามารถทำงานได้สอดคล้องกันและมีประสิทธิภาพตามวัตถุประสงค์ที่วางไว้

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1) ได้รับความรู้ความเข้าใจถึงการพัฒนาระบบภูมิสารสนเทศ
2) สามารถสร้างเว็บแอพพลิเคชั่น โดยใช้เทคโนโลยี HTML5 และ JavaScript เข้ามาประยุกต์
3) ได้รับความรู้เกี่ยวกับการทำเว็บแอพพลิเคชั่นรวมถึงเครื่องมือต่างๆ ที่นำมาใช้
4) ได้รับความรู้เกี่ยวกับการบริหารจัดการสมาร์ทมิเตอร์ได้แก่ การติดตั้งและการตรวจสอบ
5) เข้าใจถึงกระบวนการพัฒนาระบบ โดยมีการวางแผน การวิเคราะห์ การออกแบบและการพัฒนาโปรแกรม

1.6 ส่วนประกอบของรายงาน

ปริญญานิพนธ์ฉบับนี้แบ่งเนื้อหาออกเป็น 6 บทด้วยกันคือ

บทที่ 1 บทนำ กล่าวถึงความสำคัญและที่มาของโครงงาน วัตถุประสงค์ของโครงงาน ขอบเขตของโครงงาน วิธีการดำเนินการ ประโยชน์ที่คาดว่าจะได้รับ และส่วนประกอบของรายงาน

บทที่ 2 ทฤษฎีที่เกี่ยวข้อง กล่าวถึงทฤษฎีพื้นฐานที่ใช้ในโครงงานประกอบด้วยระบบพิกัด, Projection, ระบบภูมิสารสนเทศ, Mapserver, PostgreSQL+PostGIS, Django+GeoDjango, JavaScript และ OpenLayers

บทที่ 3 การบริการและโครงสร้างของระบบ กล่าวถึงการให้บริการของระบบและโครงสร้างโดยรวมทั้งหมดของระบบ

บทที่ 4 การออกแบบและการพัฒนาระบบ กล่าวถึงรายละเอียดการออกแบบและการพัฒนาโครงงานนี้ โดยอธิบายส่วนต่างๆ ของการทำงานของระบบ

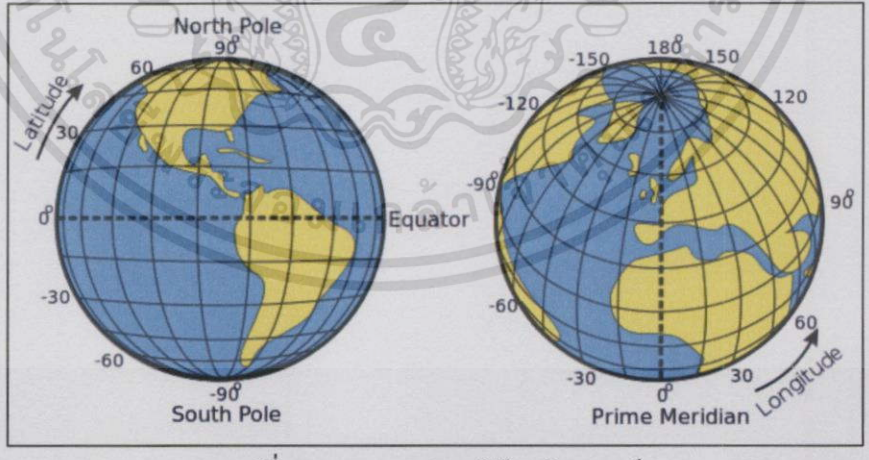
บทที่ 5 ผลการทดลองระบบบริการภูมิสารสนเทศ กล่าวถึงรายละเอียดผลการทดลองระบบ

บทที่ 6 บทสรุป กล่าวถึงบทสรุปของโครงงาน สิ่งที่ได้รับจากโครงงาน ข้อจำกัด รวมถึงปัญหาอุปสรรคต่างๆ ของโครงงานและข้อเสนอแนะสำหรับเป็นแนวทางในการพัฒนาต่อ

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1 ระบบพิกัดภูมิศาสตร์

ระบบพิกัดภูมิศาสตร์ (Geographic Coordinate System) เป็นระบบพิกัดที่กำหนดตำแหน่งต่างๆ บนพื้นโลก ด้วยวิธีการอ้างอิงบอกตำแหน่งเป็นค่าระยะเชิงมุมของละติจูด (Latitude) และลองจิจูด (Longitude) ตามระยะเชิงมุมที่ห่างจากศูนย์กำเนิด (Origin) ของละติจูดและลองจิจูดที่กำหนดขึ้นสำหรับศูนย์กำเนิดของละติจูด (Origin of Latitude) นั้นกำหนดขึ้นจากแนวระดับ ที่ตัดผ่านศูนย์กลางของโลกและตั้งฉากกับแกนหมุน เรียกแนวระนาบศูนย์กำเนิดนั้นว่า เส้นศูนย์สูตร (Equator) ซึ่งแบ่งโลกออกเป็นซีกโลกเหนือและซีกโลกใต้ ฉะนั้นค่าระยะเชิงมุมของละติจูด จะเป็นค่าเชิงมุมที่เกิดจากมุมที่ศูนย์กลางของโลก กับแนวระดับฐานกำเนิดมุมที่เส้นศูนย์สูตรที่วัดค่าของมุมออกไปทั้งซีกโลกเหนือและซีกโลกใต้ ค่าของมุมจะสิ้นสุดที่ขั้วโลกเหนือและขั้วโลกใต้ มีค่าเชิงมุม 90 องศาพอดี ดังนั้นการใช้ค่าระยะเชิงมุมของละติจูดอ้างอิงบอกตำแหน่งต่างๆ นอกจากจะกำหนดเรียกค่าวัดเป็น องศาลิปดา และฟิลิปดา แล้วจะบอกซีกโลกเหนือหรือใต้กำกับด้วยเสมอ

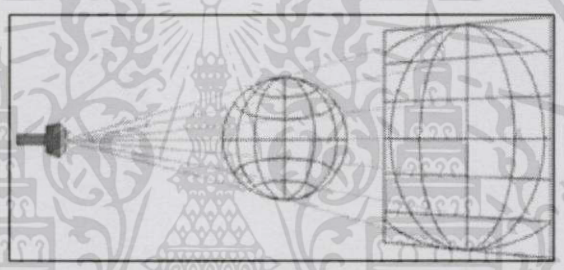


รูปที่ 2.1 แสดงระบบพิกัดภูมิศาสตร์

2.2 Projections

ด้วยเพราะว่าแผนที่นั้นเป็นแผ่นแบนราบแต่โลกมีรูปร่างเป็นแบบสเฟียร์ ทำอย่างไรที่จะทำให้การแสดงผลวัตถุต่าง ๆ ของผิวโลกมีความถูกต้องมากที่สุด วิธีการนำรูปร่างทรงกลมสามมิติของโลกมานำเสนอในแผนที่ที่เป็นสองมิติเรียกว่าการโปรเจคแผนที่ (Projection) วิธีการโปรเจคแผนที่มีอยู่หลายวิธี ซึ่งควรจะทราบคุณลักษณะของการเลือกใช้โปรเจคชั่นและผลกระทบจากการนำมาวิเคราะห์

กระบวนการโปรเจคแผนที่นั้นคิดค้นโดยนักแผนที่โดยอาศัยแนวความคิดว่า ให้มีแหล่งแสงต้นกำเนิดส่องไปยังโลกและมีฉากมารับไว้ด้านหลังวาดรูปตามฉากด้านหลังเป็นภาพสองมิติออกมาเป็นแผนที่ ซึ่งแหล่งแสงต้นกำเนิดอาจมาจากกึ่งกลางโลก ด้านบน หรือแห่งใดแห่งหนึ่งก็ได้ และฉากที่มารับเป็นภาพสองมิติอาจสร้างได้หลายแบบเช่น ฉากทรงกระบอก ฉากตรง หรือฉากรูปกรวย

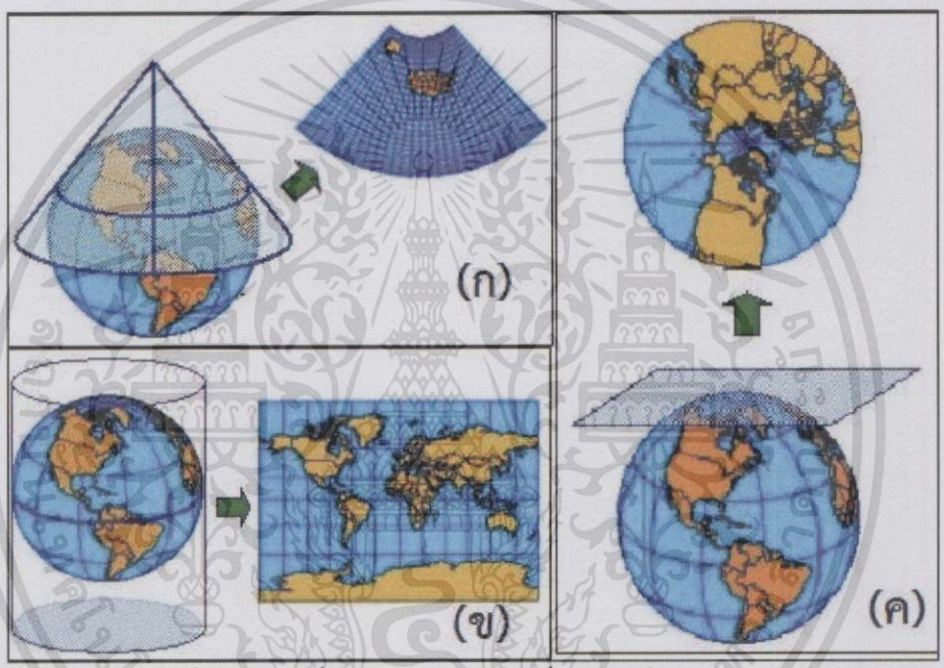


รูปที่ 2.2 แสดงการฉายแสงผ่านโลกมาตกกับฉากด้านหลัง

ถึงแม้ว่าแผนที่ต่าง ๆ นั้นสร้างโดยอาศัยสูตรการคำนวณทางคณิตศาสตร์ โดยไม่ได้เกิดจากการฉายแสงผ่านโลกมาตกกับฉากด้านหลังจริง ๆ แต่การโปรเจคแผนที่ก็อาศัยแนวความคิดนี้มาสร้างเป็นกฎเกณฑ์เพื่อสร้างแผนที่และคำนวณความบิดเบี้ยวของแผนที่ต่าง ๆในปัจจุบันโปรเจคชั่นทุก ๆ โปรเจคชั่นมีสมการในการคำนวณจากระบบพิกัด geographic ไปเป็นโปรเจคชั่นนั้น ๆ ซึ่งโปรเจคชั่นใหม่ ๆ นั้นจะมีรูปแบบหลักอยู่ 4 แบบคือ

1) รูปทรงกระบอก (Cylinder) แนวความคิดของโปรเจคชั่นแบบทรงกระบอกคือให้ต้นกำเนิดแสงอยู่กึ่งกลางโลกส่องแสงออกมาภายนอกโลก โดยมีฉากรูปทรงกระบอกห่อรอบโลกอยู่ การโปรเจคแบบนี้บริเวณที่ฉากสัมผัสกับผิวโลกจะมีความบิดเบี้ยวน้อย แต่บริเวณที่ห่างจากจุดสัมผัสมากขึ้นก็มีความบิดเบี้ยวมากขึ้น
2) รูปทรงระนาบ (Plane) แนวความคิดการโปรเจคแบบ plane นั้นให้ต้นกำเนิดแสงอยู่กึ่งกลางโลกโดยมีฉากแบบราบสัมผัสกับผิวโลกจุดใดจุดหนึ่ง จากตัวอย่างด้านบนนี้ฉากจะสัมผัสกับผิวโลกที่ขั้วโลกเหนือ จากการโปรเจคชั่นแบบนี้จะทำให้เส้นรุ้งปรากฏออกมาเป็นเส้นวงกลมส่วนเส้นแวงจะปรากฏออกมาเป็นเส้นตรงพุ่งมาเป็นรัศมีคล้ายซี่ล้อรถจักรยาน

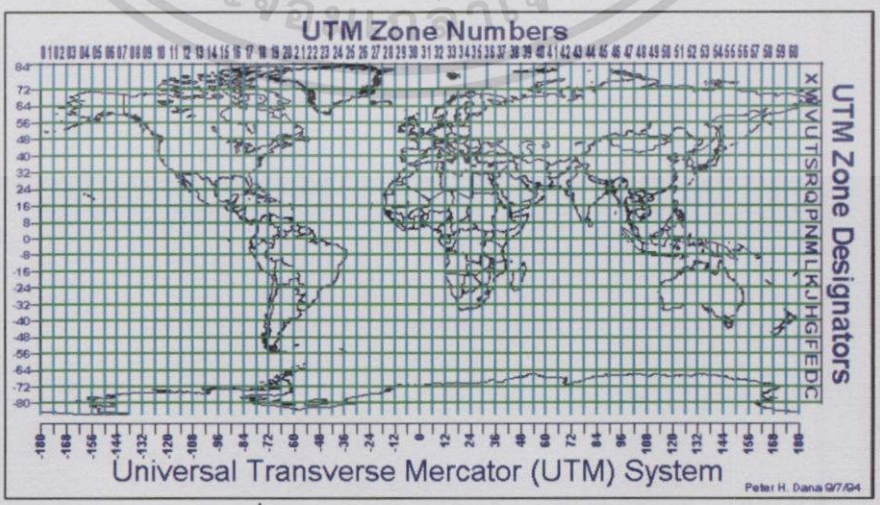
3) รูปทรงกรวย (Cone) ผลจากการโปรเจคแบบรูปทรงกรวยนี้จะพบว่าเส้นรุ้งจะเป็นเส้นโค้งซ้อน ๆ กัน และเส้นแวงจะเป็นเส้นตรงที่เบนด้านบนเข้าหากัน การโปรเจคแบบรูปทรงกรวยนี้ไม่เหมาะสมนำมาใช้งานกับบริเวณเส้นศูนย์สูตรเพราะการโปรเจคชั่นแบบนี้ความคลาดเคลื่อนด้านล่างจะเพิ่มขึ้นมากเรื่อย ๆ แต่การโปรเจคชั่นแบบนี้น่าจะเหมาะกับพื้นที่บริเวณซีกโลกเหนือหรือซีกโลกใต้ และมีรูปร่างกระจายไปทางด้านตะวันออกและตะวันตก ดังเช่นประเทศสหรัฐอเมริกา



รูปที่ 2.3 แสดงโปรเจคชั่นแบบต่างๆ
(ก) รูปทรงกรวย
(ข) รูปกระบอก
(ค) รูปทรงระนาบ

4) Projection แบบ UTM ระบบพิกัดกริดแบบ UTM (Universal Transverse Mercator co-ordinate System) พิกัดกริด UTM (Universal Transverse Marcator) เป็นการฉายภาพลูกโลกบนพื้นระนาบ 2 มิติแบบหนึ่งซึ่งเป็นระบบตารางกริดที่ใช้ช่วยในการกำหนดตำแหน่งและใช้อ้างอิงในการบอกตำแหน่ง ที่นิยมใช้กับแผนที่ในกิจการทหารของประเทศ ต่าง ๆ เกือบทั่วโลกในปัจจุบัน เพราะเป็นระบบตารางกริดที่มีขนาดรูปร่างเท่ากันทุกตารางและมีวิธีการกำหนดบอกค่าพิกัดที่ง่ายและถูกต้องเป็นระบบ กริดที่นำเอาเส้นโครงแผนที่แบบ Universal Transverse Mercator Projection ของ Gauss -Krueger มาใช้ดัดแปลงการ

ถ่ายทอดรายละเอียดของพื้นผิวโลกให้รูปทรงกระบอก Mercator Projection อยู่ในตำแหน่ง Mercator Projection (แกนของรูปทรงกระบอกจะทับกับแนวเส้นอิเควเตอร์ และตั้งฉากกับแนวแกนของขั้วโลก) ประเทศไทยเราได้นำเอาเส้นโครงแผนที่แบบ UTM นี้มาใช้กับการทำแผนที่ เป็นชุด L 7017 ที่ใช้ในปัจจุบันแผนที่ระบบพิกัดกริด ที่ใช้เส้นโครงแผนที่แบบ UTM เป็นระบบเส้นโครงชนิดหนึ่งที่ใช้ผิวรูปทรงกระบอกเป็นผิวแสดงเส้นเมริเดียน (หรือเส้นลองจิจูด) และเส้นละติจูดของโลก โดยใช้ทรงกระบอกตัดโลกระหว่างละติจูด 84 องศาเหนือ และ 80 องศาใต้ในลักษณะแกนรูปทรงกระบอกแล้วทำมุมกับแกนโลก 90 องศา รอบโลก แบ่งออกเป็น 60 โซนๆ ละ 6 องศา โซนที่ 1 อยู่ระหว่าง 180 องศา กับ 174 องศา ตะวันตก และมีลองจิจูด 177 องศาตะวันตกเป็นเมริเดียนย่านกลาง (Central Meridian) มีเลขกำกับแต่ละโซนจาก 1 ถึง 60 โดย นับจากซ้ายไปทางขวาระหว่างละติจูด 84 องศาเหนือ 80 องศาใต้ แบ่งออกเป็น 2 ช่อง ช่องละ 8 องศา ยกเว้นช่องสุดท้าย เป็น 12 องศา โดยเริ่มนับตั้งแต่ละติจูด 80 องศาใต้ ขึ้นไป ทางเหนือให้ช่องแรกเป็นอักษร C และช่องสุดท้ายเป็นอักษร X (ยกเว้น I และ O) จากการแบ่งตามที่กล่าวแล้วจะเห็นพื้นที่ในเขตลองจิจูด 180 องศาตะวันตก ถึง 180 องศาตะวันออกและละติจูด 80 องศาใต้ ถึง 84 องศาเหนือ จะถูกแบ่งออกเป็นรูปสี่เหลี่ยมผืนผ้า 1,200 รูป แต่ละรูปมีขนาดกว้างยาว 6 องศา x 8 องศา จำนวน 1,140 รูป และกว้างยาว 6 องศา x 12 องศา จำนวน 60 รูป รูปสี่เหลี่ยมนี้เรียกว่า Grid Zone Designation (GZD) การเรียกชื่อ Grid Zone Designation ประเทศไทยมีพื้นที่อยู่ ระหว่างละติจูด 5 องศา 30 ลิปดา เหนือ ถึง 20 องศา 30 ลิปดา เหนือและลองจิจูดประมาณ 97 องศา 30 ลิปดา ตะวันออก ถึง 105 องศา 30 ลิปดา ตะวันออก ดังนั้น ประเทศไทยจึงตกอยู่ในGZD 47N 47P 47Q 48N 48P และ 48 Q



รูปที่ 2.4 แสดงระบบพิกัดกริดแบบ UTM

2.3 ระบบภูมิสารสนเทศ

2.3.1 ความหมายของภูมิสารสนเทศ

ระบบภูมิสารสนเทศ หรือ Geographic Information System : GIS คือระบบสารสนเทศที่เกี่ยวข้องกับข้อมูลเชิงพื้นที่ เช่น ที่อยู่ บ้านเลขที่ เป็นระบบข้อมูลสารสนเทศที่อยู่ในรูปของตารางข้อมูล และฐานข้อมูลที่มีส่วนสัมพันธ์กับข้อมูลเชิงพื้นที่ (Spatial Data) ซึ่งรูปแบบและความสัมพันธ์ของข้อมูลเชิงพื้นที่ทั้งหลาย จะสามารถนำมาวิเคราะห์และทำให้สื่อความหมายในเรื่องการเปลี่ยนแปลงที่สัมพันธ์กับเวลาได้ เมื่อปรากฏบนแผนที่ทำให้สามารถแปลและสื่อความหมาย ใช้งานได้ง่าย

GIS เป็นสารสนเทศที่สามารถแปลความหมายข้อมูลเชื่อมโยงกับสภาพภูมิศาสตร์ต่างๆ สภาพท้องที่ การทำงานของระบบสัมพันธ์กับสัดส่วนระยะทางและพื้นที่จริงบนแผนที่ ข้อแตกต่างระหว่าง GIS กับ MIS นั้นสามารถพิจารณาได้จากลักษณะของข้อมูล คือ ข้อมูลที่จัดเก็บใน GIS มีลักษณะเป็นข้อมูลเชิงพื้นที่ (Spatial Data) ที่แสดงในรูปของภาพ (graphic) แผนที่ (map) ที่เชื่อมโยงกับข้อมูลเชิงบรรยาย (Attribute Data) หรือฐานข้อมูล (Database)การเชื่อมโยงข้อมูลทั้งสองประเภทเข้าด้วยกัน จะทำให้ผู้ใช้สามารถที่จะแสดงข้อมูลทั้งสองประเภทได้พร้อมๆ กัน เช่น สามารถจะค้นหาตำแหน่งของจุดตรวจวัดควันดำ - ควันขาวได้โดยการระบุชื่อจุดตรวจ หรือในทางตรงกันข้าม สามารถที่จะสอบถามรายละเอียดของ จุดตรวจจากตำแหน่งที่เลือกขึ้นมา ซึ่งจะต่างจาก MIS ที่แสดงภาพได้เพียงอย่างเดียว โดยจะขาดการเชื่อมโยงกับฐานข้อมูลที่เชื่อมโยงกับรูปภาพนั้น เช่นใน CAD (Computer Aid Design) จะเป็นภาพเพียงอย่างเดียว แต่แผนที่ใน GIS จะมีความสัมพันธ์กับตำแหน่งในเชิงพื้นที่ทางภูมิศาสตร์ คือค่าพิกัดที่แน่นอน ข้อมูลใน GIS ทั้งข้อมูลเชิงพื้นที่และข้อมูลเชิงบรรยาย สามารถอ้างอิงถึงตำแหน่งที่มีอยู่จริงบนพื้นโลกได้โดยอาศัยระบบพิกัดทางภูมิศาสตร์ (Geocode) ซึ่งจะสามารถอ้างอิงได้ทั้งทางตรงและทางอ้อม ข้อมูลใน GIS ที่อ้างอิงกับพื้นผิวโลกโดยตรง หมายถึง ข้อมูลที่มีค่าพิกัดหรือมีตำแหน่งจริงบนพื้นโลกหรือในแผนที่ เช่น ตำแหน่งอาคาร ถนน ฯลฯ สำหรับข้อมูล GIS ที่จะอ้างอิงกับข้อมูลบนพื้นโลกได้โดยทางอ้อมได้แก่ ข้อมูลของบ้าน(รวมถึงบ้านเลขที่ ซอย เขต แขวง จังหวัด และรหัสไปรษณีย์) โดยจากข้อมูลที่อยู่ เราสามารถทราบได้ว่าบ้านหลังนี้มีตำแหน่งอยู่ ณ ที่ใดบนพื้นโลก เนื่องจากบ้านทุกหลังจะมีที่อยู่ไม่ซ้ำกัน

2.3.2 ข้อมูลภูมิสารสนเทศ

ข้อมูลภูมิสารสนเทศแบ่งออกได้เป็น 2 ประเภท คือ ข้อมูลเชิงพื้นที่หรือเชิงตำแหน่ง (Spatial Data) และข้อมูลเชิงคุณลักษณะ (Attribute)

ข้อมูลเชิงตำแหน่งประกอบด้วยข้อมูล 3 ชนิดด้วยกัน คือ

1) จุด (points)

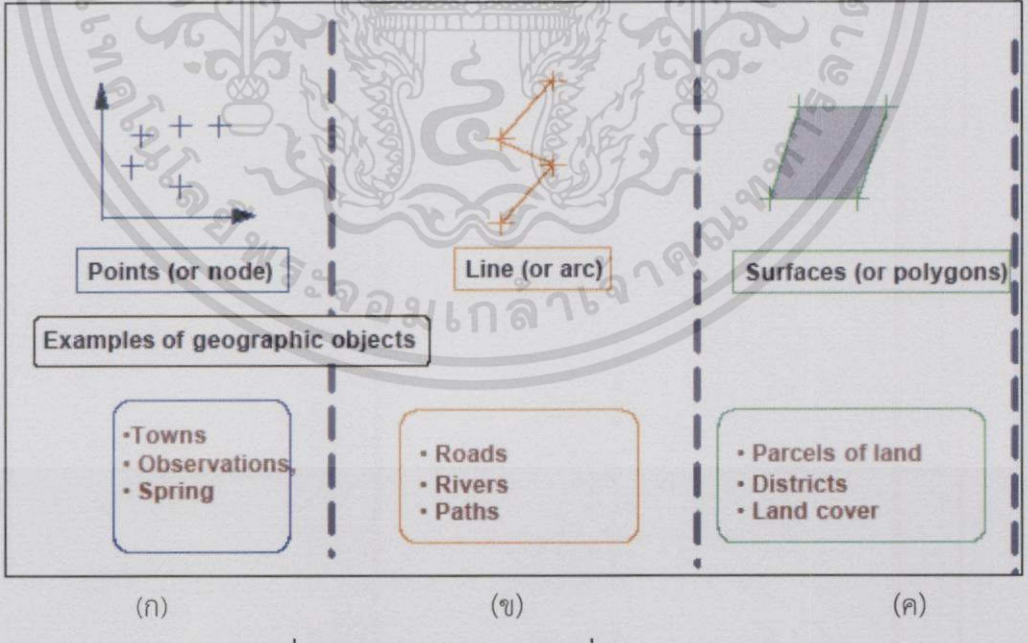
ข้อมูลจุด เป็นรูปแบบของข้อมูลเชิงตำแหน่งที่มีขนาด แต่ไม่มีระยะทาง หรือ ความยาว มีมิติเป็น0 (Zero Dimensional Objects) แสดงตำแหน่งในพื้นที่ เช่น ที่ตั้งของเมือง จุดสังเกต และตำแหน่งของน้ำพุ เป็นต้น

2) เส้น (lines)

ข้อมูลเส้น หมายถึงวัตถุที่มีหนึ่งมิติ (One Dimension) มีตำแหน่ง (position)ในพื้นที่ มีความยาว แต่ไม่มีความกว้าง เช่น ถนน แม่น้ำ และทางเดิน เป็นต้น

3) พื้นที่ (Polygons)

ข้อมูลพื้นที่ หมายถึงวัตถุที่มีสองมิติ (Two-Dimensional Spatial Objects) ซึ่งนอกจากจะมีตำแหน่งแล้ว ยังมีความยาว (length) และความกว้าง (width) อีกด้วย เช่น ขอบเขตของพื้นที่ เขตการปกครอง เป็นต้น



รูปที่ 2.5 ข้อมูลเชิงตำแหน่ง ที่มา: GDTA (2001)

1) แสดงภาพข้อมูลจุด

2) แสดงภาพข้อมูลเส้น

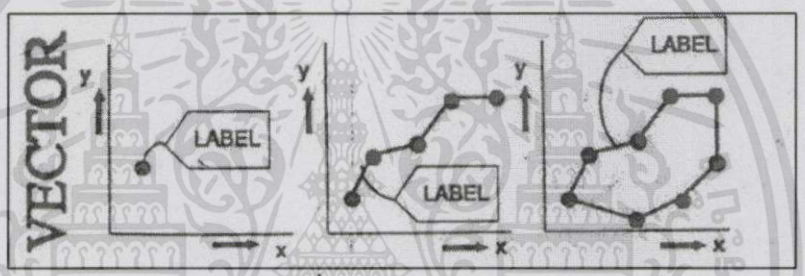
3) แสดงภาพข้อมูลพื้นที่

ข้อมูลแผนที่แบ่งออกเป็น 2 แบบ คือ เวกเตอร์โมเดล (Vector Model) และ แรสเตอร์โมเดล (Raster Model) ดังนี้

1) เวกเตอร์โมเดล (Vector Model)

โครงสร้างชนิดนี้ ตำแหน่งของจุด เส้น และพื้นที่ตามที่ได้อธิบายไปแล้ว ถูกกำหนดโดยใช้พิกัดซึ่งมีตำแหน่งค่อนข้างแน่นอน

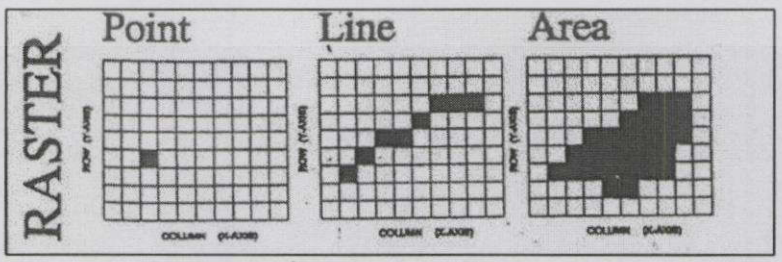
- จุด ถูกกำหนดโดยพิกัด X,Y เพียงคู่เดียว และมีชื่อกำกับ
- เส้น ถูกกำหนดโดยพิกัด X,Y มากกว่า 1 คู่ ขึ้นไป และมีชื่อกำกับ
- พื้นที่ หรือ Polygon ถูกกำหนดโดยกลุ่มของระบบพิกัดและมีชื่อประกอบโดยที่จุดเริ่มต้น และจุดสุดท้ายของพิกัดจะมีค่าเหมือนกัน



รูปที่ 2.6 เวกเตอร์โมเดล

2) แรสเตอร์โมเดล (Raster Model)

เป็นข้อมูลแผนที่ที่ประกอบไปด้วยจุดภาพจำนวนมากมาย จัดเรียงตัวกันอยู่ตามแนวนอน (row) และตามแนวตั้ง (column) โครงสร้างของข้อมูลแรสเตอร์ ชนิดจุดจะแสดงด้วย จุดภาพ 1จุด ในขณะที่ เส้น แสดงด้วยจุดภาพที่เรียงต่อกันเป็นแนวเข้าด้วยกัน (String Of Pixels) ส่วนพื้นที่จะแสดงด้วยกลุ่มของจุดภาพ (Group of Pixels) อาจจะกล่าวได้ว่าโครงสร้างแบบแรสเตอร์นี้ไม่เหมาะสมในการใช้แสดงข้อมูลที่เป็น เส้น หรือข้อมูล Graphic



รูปที่ 2.7 แรสเตอร์โมเดล

องค์ประกอบพื้นฐานของข้อมูลแรสเตอร์ คือ "จุดภาพ" ซึ่งเป็นค่าที่แสดงคุณสมบัติของพื้นที่หนึ่งเช่น ชนิดของดิน (Soil Type) ความสูง (Altitude) ค่าสะท้อนแสง (Reflectance Value) จากข้อมูลดาวเทียมหรือ ลักษณะอื่น ๆ ซึ่งคุณสมบัติต่าง ๆ เหล่านี้จะแทนด้วยค่า จุดภาพ ซึ่งเป็นตัวเลข (Numerical Properties) เช่น ความสูง ความลึกของดิน ทิศด้านลาด ฯลฯ หรืออาจแสดงเป็นการผสมระหว่างตัวเลขและตัวอักษร (Alphanumeric Data) เช่น ชนิดของดิน ซึ่งเชื่อมโยงกับตารางที่อธิบายเกี่ยวกับชนิดของดินนั้น เป็นต้น

จุดภาพ 1 จุด จะแทนด้วยค่า 1 ค่า ซึ่งหมายความว่ารายละเอียดต่างๆ นั้นได้มาจากพื้นที่ที่ถูกครอบคลุมด้วยจุดภาพนั้น ในระบบ GIS เกือบทั้งหมดจุดภาพจะมีลักษณะเป็นแบบสี่เหลี่ยม ขนาดของจุดภาพจะถูกกำหนดโดยขอบเขตของหน่วยภูมิประเทศที่เกี่ยวข้อง เช่น ถ้าขนาดของจุดภาพมีค่า = 10 เมตร หมายความว่าแต่ละจุดภาพในแผนที่จะประกอบด้วยค่าซึ่งเป็นตัวแทนของหน่วยพื้นที่ที่มีขนาด 100 ตารางเมตร

การจัดเก็บข้อมูลแรสเตอร์ ขึ้นอยู่กับขนาดพื้นที่ ขนาดของจุดภาพ และชนิดของข้อมูลที่เก็บด้วย โดยทั่วไปแล้วแผนที่แรสเตอร์ แบ่งออกได้ 2 ชนิด คือ

1) 1 -bit map: แต่ละจุดภาพ จะประกอบไปด้วยค่าเพียง 2 ค่า เป็นอย่างมาก คือ 1 หรือ 0 ต้องการพื้นที่ในการจัดเก็บไม่มากนัก เพราะแต่ละจุดภาพต้องการเนื้อที่เพียง 1 Bit เท่านั้น
2) 8-bit map: หรือ 2-byte word ใน 1 จุดภาพสามารถเก็บโดยใช้เนื้อที่บนคอมพิวเตอร์ 1 Byte ใน 1 Byte จะประกอบด้วยหน่วยย่อย 8 bit ซึ่งสามารถที่จะแสดงค่าได้ถึง 256 ระดับ (0-255) จอภาพสีสามารถแสดงรายละเอียดได้เพียง 256 ระดับ ดังนั้น 8-bit map จึงสามารถใช้ในการเก็บ แผนที่เฉพาะเรื่อง (Thematic Map) ซึ่งมีค่าไม่เกิน 256 ค่า เช่นข้อมูลดาวเทียม หรือข้อมูลภาพถ่ายทางอากาศที่ได้มาจากการแสกน

โครงสร้างแบบเวกเตอร์และแบบแรสเตอร์ มีข้อได้เปรียบและเสียเปรียบ ในการใช้งาน ดังได้สรุปการเลือกใช้โครงสร้างชนิดใดชนิดหนึ่ง จะขึ้นอยู่กับสถานการณ์ที่เหมาะสม เช่นโครงสร้างแบบแรสเตอร์จะมีความเหมาะสมเกี่ยวกับการศึกษาความผันแปรของปรากฏการณ์ในพื้นที่ ส่วนโครงสร้างเวกเตอร์จะมีความเหมาะสมสำหรับวิเคราะห์โครงข่าย (Network Analysis) เป็นต้น

ตารางที่ 2.1 ข้อได้เปรียบระหว่างโครงสร้างแบบแรสเตอร์ และแบบเวกเตอร์

ลักษณะการทำงาน	แรสเตอร์	เวคเตอร์
- การรวบรวมข้อมูล	เร็ว	ช้า
- ความคมชัด	ปานกลาง	ดี
- โครงสร้างข้อมูล	ง่าย	ซับซ้อน
- ความละเอียดเชิงเรขาคณิต	ต่ำ	สูง
- การวิเคราะห์แบบโครงข่าย	ไม่ดี	ดี
- การวิเคราะห์เชิงพื้นที่	ดี	ปานกลาง

หลังจากที่ได้อธิบายทฤษฎีพื้นฐานเกี่ยวกับระบบภูมิสารสนเทศไปแล้วต่อไปจะเป็นการอธิบายถึงทฤษของเครื่องมือ open source ที่ใช้พัฒนา

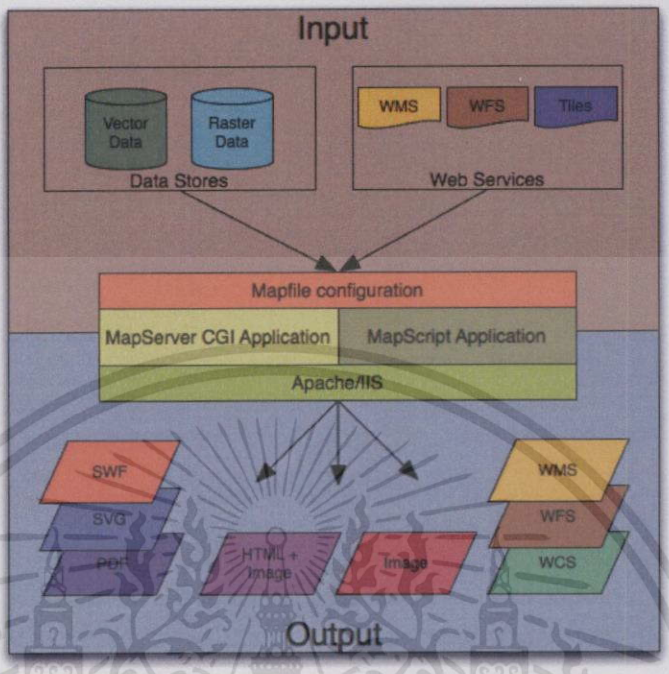
2.4 MapServer

MapServer เป็นโอเพนซอร์สสำหรับแสดงข้อมูลเชิงพื้นที่และติดต่อกับ Web Mapping Application โดยข้อมูลที่ได้มาจากฐานข้อมูล ซึ่งถูกเก็บอยู่ในรูปตารางฐานข้อมูล แล้วนำข้อมูลที่ได้ไปแสดงผลเป็นรูปภาพ

คุณสมบัติหลักของ MapServer คือ

- สนับสนุกการแสดงผลของแรสเตอร์, เวคเตอร์ และรูปแบบดาต้าเบสจำนวนมาก ๆ
- สามารถทำงานได้หลายระบบปฏิบัติการ (Windows, Linux, Mac OS X และอื่นๆ)
- สนับสนุนหลายภาษา (PHP, Python, Perl, Ruby, Java, .NET)
- การแสดงผลคุณภาพสูง
- พร้อมใช้งานในหลาย ๆ โอเพนซอร์ส

รูปแบบที่สำคัญของ MapServer คือ เมื่อการร้องขอถูกส่งไปยัง MapServer มันจะส่งข้อมูล URL และ Mapfile ไปสร้างรูปภาพแผนที่ที่ร้องขอมา การร้องขออาจจะส่งรูปภาพกลับมาสำหรับคำบรรยาย, แถบสเกล, แผนที่อ้างอิง และค่าต่าง ๆ ผ่านตัวแปร CGI



รูปที่ 2.8 การทำงานของ MapServer

โครงสร้างของ MapServer ประกอบด้วย

1) Mapfile เป็นไฟล์ที่กำหนดโครงสร้างการทำงานของ MapServer เช่น กำหนดขอบเขตของแผนที่, ชั้นข้อมูลของแผนที่, แหล่งข้อมูล, โปรเจ็คชัน, สัญลักษณ์ต่าง ๆ ที่ใช้แสดง โดยจะต้องมีนามสกุลเป็น .map
2) Geographic Data คือ ข้อมูลเชิงภูมิศาสตร์ ซึ่ง Mapserver รองรับรูปแบบข้อมูลที่หลากหลาย รวมถึง shape file และ postgis
3) HTML Pages เป็นส่วนที่ติดต่อระหว่างผู้ใช้งานกับ MapServer
4) MapServer CGI เป็นไบนารี่หรือไฟล์ที่ใช้ในการร้องขอและส่งกลับข้อมูลต่าง ๆ จาก MapServer
5) HTTP Server เป็นเว็บเซิร์ฟเวอร์ให้บริการ HTML page เช่น apache, Microsoft IIS, Nginx เป็นต้น

2.5 PostgreSQL+PostGIS

PostgreSQL เป็นระบบจัดการฐานข้อมูลเชิงวัตถุ-สัมพันธ์ (Object-Relational DataBase Management System หรือ ORDBMS) เป็นโอเพ่นซอร์สที่สนับสนุนมาตรฐานเอสคิวแอลทั่วไป (SQL Standard) และสนับสนุนคุณสมบัติใหม่ ๆ เช่น สามารถใช้ได้ทั้ง Windows, Free BSD, Solaris, Linux และ Mac OS เพราะเป็นซอฟท์แวร์แบบ cross-platform

คุณสมบัติใหม่ที่เพิ่มเข้ามาของ PostgreSQL

- complex queries
- foreign keys
- triggers
- views
- transactional integrity
- multiversion concurrency control

PostgreSQL สามารถขยายได้หลายทางโดยผู้ใช้ เช่น

- data types
- function
- operators
- aggregate functions
- index methods
- procedural language

PostGIS เป็นตัวเสริมฐานข้อมูลของ PostgreSQL เพื่อสนับสนุนข้อมูลเชิงภูมิศาสตร์ สามารถรองรับข้อมูลด้านสารภูมิสนเทศ (GIS) คือสนับสนุนข้อมูลที่สัมพันธ์เชิงพื้นที่ (Spatial) PostGIS ได้เพิ่มข้อมูลชนิดพิเศษ (เรขาคณิต, ภูมิศาสตร์, แรสเตอร์ และอื่น ๆ) ไปยัง PostgreSQL นอกจากนี้ยังเพิ่มฟังก์ชัน, ตัวกระทำ และปรับปรุงอินเดกให้ประยุกต์เข้ากับข้อมูลเชิงพื้นที่ การเพิ่มเหล่านี้ช่วยให้แกนหลักของ PostgreSQL DBMS เร็วขึ้น, มีคุณสมบัติเพิ่มขึ้น และช่วยให้มีประสิทธิภาพในการจัดการเชิงพื้นที่

PostGIS 2 มีคุณสมบัติดังนี้

- การประมวลผลและฟังก์ชั่นการวิเคราะห์ทั้งข้อมูลเวกเตอร์และแรสเตอร์สำหรับการประกบ, การตัดเป็นสี่เหลี่ยม, การเปลี่ยนจากรูปหนึ่งไปอีกรูปหนึ่ง, การจัดแบ่งประเภทและการจัดเก็บ / การรวม ขึ้นอยู่กับตัว SQL
- มีพีชคณิตแผนที่เชิงภาพสำหรับการประมวลผลเชิงภาพที่มีความละเอียด
- Spatial reprojection SQL สามารถไปเรียกฟังก์ชันได้ทั้งสำหรับข้อมูลเวคเตอร์และแรสเตอร์
- สนับสนุกการอิมพอร์ต/เอกพอร์ต ข้อมูลเวคเตอร์ ESRI shapefile ผ่านทางคอมมานไลน์
- แพคเกจคอมมานไลน์เพื่อการอิมพอร์ตข้อมูลแรสเตอร์จากหลายรูปแบบ : GeoTiff, NetCDF, PNG, JPG
- การแสดงผลและนำเข้าข้อมูลเวคเตอร์สนับสนุนฟังก์ชันรูปแบบต้นฉบับมาตรฐาน เช่น KML, GML, GeoJSON, GeoHash และ WKT โดยใช้ SQL
- สามารถเรียกฟังก์ชันเพื่อการอัดขึ้นรูปของค่าพิกเซลโดยขอบเขตเรขาคณิต, ตัดแรสเตอร์โดยรูปทรงเราขาคณิต,
- สนับสนุวัตถุ 3 มิติ, อินเดกเชิงพื้นที่
- สนับสนุนโครงสร้างเครือข่าย

Spatial Database เป็นฐานข้อมูลที่กำหนดคอลัมน์ชนิดของข้อมูลแบบพิเศษเพื่อเก็บวัตถุในพื้นที่ซึ่งสามารถเพิ่มลงไปในตารางฐานข้อมูลได้ ข้อมูลที่เก็บไว้โดยปกติจะเป็นข้อมูลทางภูมิศาสตร์ในธรรมชาติเช่นจุดที่ตั้งหรือขอบเขตของทะเลสาบ โดยมีการแบ่งฟังก์ชันและอินเด็กซ์สำหรับการคิวรี่และการจัดการข้อมูลที่สามารถเรียกจาก query language เช่น Structured Query Language (SQL) ส่วนมากฐานข้อมูลเชิงพื้นที่จะใช้เก็บส่วนประกอบของข้อมูลเชิงพื้นที่

ฐานข้อมูลเชิงพื้นที่ได้ให้เครื่องมือในการเก็บ, การวิเคราะห์และการจัดการรวมอยู่ในอันเดียวกัน

2.6 Django+GeoDjango

Django เป็น web application framework ของภาษา Python เพื่อใช้ในการพัฒนาเว็บแอพพลิเคชันและเว็บเซอร์วิสต่างๆ โดยใช้หลักของเอ็มวีซี (MVC : Model-View-Controller) ทำให้เว็บมีความเป็นระเบียบ

คุณสมบัติของ Django Framework

1) Object-relational mapper คือ การกำหนด Data Model ในภาษา Python เพื่อการทำงานด้านข้อมูล และสนับสนุน dynamic database-access API
2) Automatic admin interface คือ ส่วนของการสร้าง Interface อัตโนมัติสำหรับการ add, edit , delete และ search ด้วย Django Framework
3) Elegant URL design คือ การทำให้ URL มีความสวยงาม สั้น กระชับ และสื่อความหมายของหน้านั้น ๆ ได้อย่างชัดเจน เหมาะสมกับการทำ SEO ในปัจจุบัน
4) Template system คือ Django นั้นมีการออกแบบ Template Language เพื่อการเขียนแยกส่วนระหว่าง Design และ Business Logic
5) Cache system คือ ส่วนของการบันทึก หรือจัดการข้อมูลที่มีการดาวน์โหลดไปแล้ว เพื่อเพิ่มประสิทธิภาพการทำงานของเว็บไซต์ด้านความเร็ว และด้านอื่น ๆ
6) Internationalization คือ Django สนับสนุน Application ที่มีความหลากหลายด้านภาษาในการแสดงผล

GeoDjango เป็นโมดูลสำหรับ Django โดยมุ่งเน้นเพื่อทำให้ง่ายต่อการสร้างเว็บแอพพลิเคชั่นทางภูมิศาสตร์ โดยมีส่วนรองรับ spatial database เช่น Postgresql+Postgis, Oracal และ Mysql และรองรับการทำงานตามมาตรฐานของ OGC นอกจากนี้ยังรองรับการทำงานกับรูปแบบการจัดเก็บข้อมูล GIS

2.7 JavaScript

JavaScript เป็นภาษาโปรแกรมที่เรียกกันว่าสคริปต์ (script) ซึ่งมีวิธีการทำงานในลักษณะแปลความและดำเนินงานไปทีละคำสั่ง (interpret) ได้รับการพัฒนาเพื่อที่จะช่วยให้เว็บเพจสามารถแสดงเนื้อหาที่มีการเปลี่ยนแปลงไปได้ ตามเงื่อนไขหรือสภาพแวดล้อมต่างๆกัน หรือสามารถโต้ตอบกับผู้ใช้ได้มากขึ้น ทั้งนี้ภาษา HTML แต่เดิมนั้นเหมาะสำหรับใช้แสดงเอกสารที่มีเนื้อหาคงที่แน่นอนและไม่มีลูกเล่นอะไรมากมาย

การทำงานของ JavaScript จะต้องมีการแปลความคำสั่งซึ่งขั้นตอนนี้จะถูกจัดการโดยบราวเซอร์ ดังนั้น JavaScript จึงสามารถทำงานได้เฉพาะบนบราวเซอร์ที่สนับสนุน ซึ่งปัจจุบันบราวเซอร์เกือบทั้งหมดก็สนับสนุน JavaScript

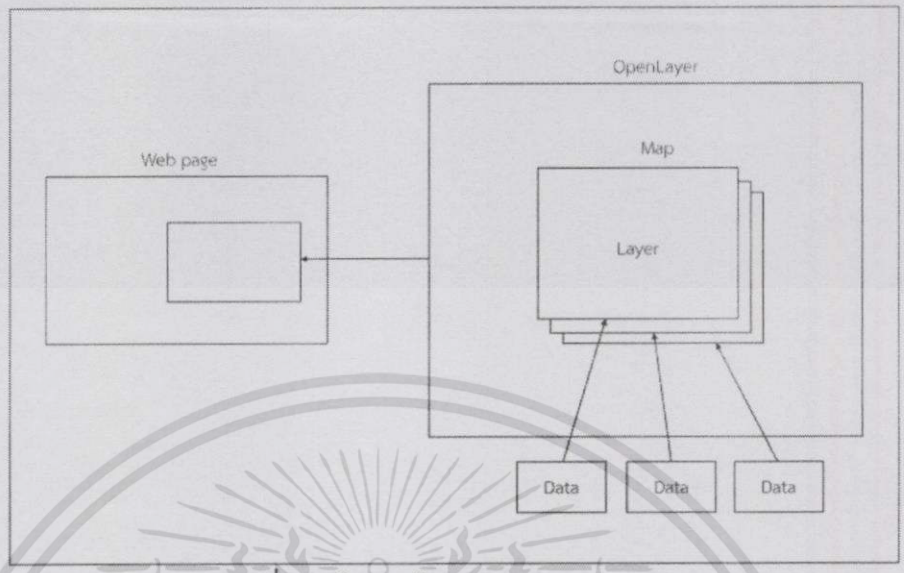
การทำงานของ JavaScript เกิดขึ้นบนบราวเซอร์ (client-side script) ไม่ว่าจะใช้เซิร์ฟเวอร์อะไรหรือที่ไหน ก็ยังคงสามารถใช้ JavaScript ในเว็บเพจได้ ต่างกับภาษาสคริปต์อื่น เช่น Perl, PHP หรือ ASP ซึ่งต้องแปลความและทำงานที่ตัวเครื่องเซิร์ฟเวอร์ (server-side script) ดังนั้นจึงต้องใช้บนเซิร์ฟเวอร์ที่สนับสนุนภาษาเหล่านี้เท่านั้น จากลักษณะดังกล่าวก็ทำให้ JavaScript มีข้อจำกัด คือไม่สามารถรับและส่งข้อมูลต่างๆ กับเซิร์ฟเวอร์โดยตรง เช่น การอ่านไฟล์จากเซิร์ฟเวอร์เพื่อนำมาแสดงบนเว็บเพจ หรือรับข้อมูลจากผู้ชมเพื่อนำไปเก็บบนเซิร์ฟเวอร์ เป็นต้น

2.8 OpenLayers

OpenLayers เป็น JavaScript Library ในการพัฒนา Web Mapping Aplication ขั้นสูงสำหรับแสดงข้อมูลแผนที่ผ่านทางเว็บบราวเซอร์ซึ่งไม่ขึ้นกับ server-side โดย Openlayers นั้นรองรับการแสดงผลแผนที่ที่ใช้เทคนิคแบบ Tile Caching มีความยืดหยุ่นในการพัฒนาเพราะ core โมดูลส่วนใหญ่เป็น API ที่เรียกใช้งานได้ง่าย และสามารถขยาย Class หลัก ๆ ให้รองรับงานได้

OpenLayers ได้เตรียม JavaScript API ไว้สำหรับการสร้างเว็บเบสแอพพลิเคชั่นทางภูมิศาสตร์ คล้าย ๆ กับ Google Maps และ MSN Virtual Earth APIs

OpenLayers มี 2 แนวคิดหลัก ๆ ซึ่งเป็นสิ่งสำคัญที่ต้องเข้าใจก็คือ แผนที่และชั้นข้อมูล โดยตัวแผนที่จะเก็บข้อมูลโปรเจคชัน ขอบเขต และส่วนอื่น ๆ ของแผนที่ ภายในแผนที่ข้อมูลถูกส่งผ่านไปยังชั้นข้อมูล ชั้นข้อมูลเป็นแหล่งข้อมูลว่าจะร้องขอข้อมูลหรือแสดงข้อมูล



รูปที่ 2.9 โครงสร้างของ OpenLayers

Layer เป็นชั้นข้อมูลที่นำมาแสดงผลโดยมีข้อมูลอยู่ภายใน แต่ละชั้นข้อมูลสามารถนำมาซ้อนกันได้เพื่อประกอบกันเป็นข้อมูลเดียว

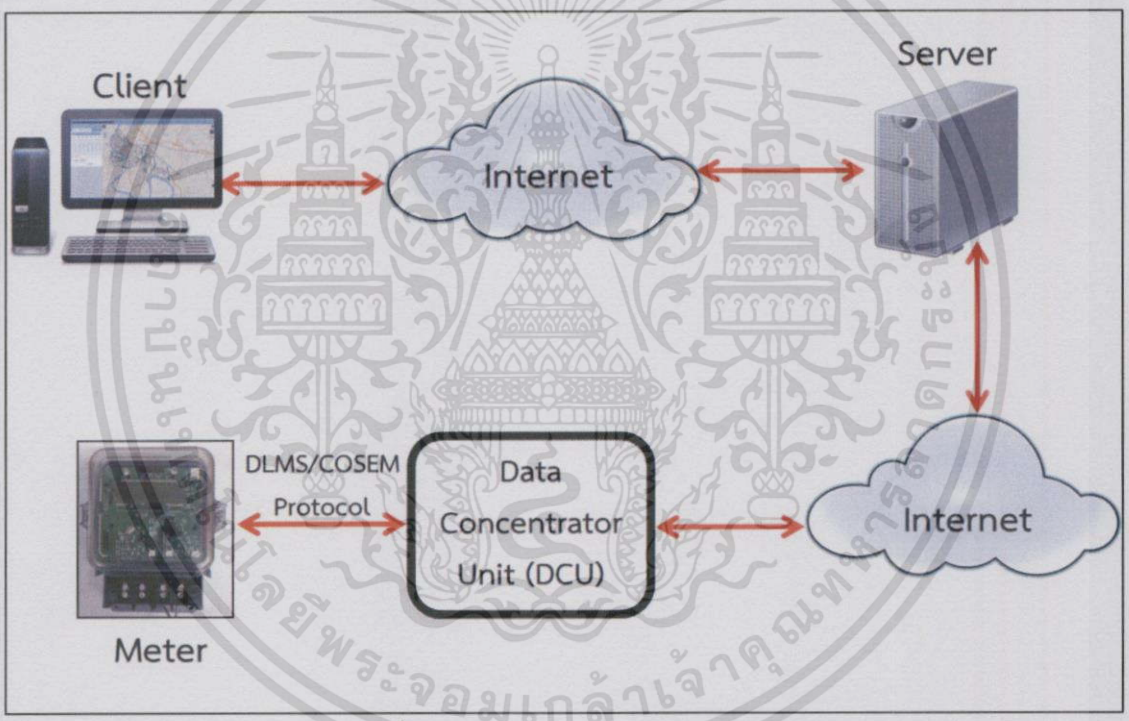
Base layer เป็นชั้นข้อมูลชั้นฐานล่างสุด จะวางไว้ก่อนชั้นข้อมูลอื่น ๆ โดยภายในจะมีข้อมูลทางภูมิศาสตร์เป็นฐานแสดงผล แล้วจึงนำชั้นข้อมูลอื่น ๆ มาซ้อนทับ

ในการทำงานของ OpenLayers นั้น OpenLayers จะทำการร้องขอแผนที่ไปยังเซิร์ฟเวอร์ WMS เมื่อเซิร์ฟเวอร์ได้รับแล้วก็จะสร้างแผนที่ตามที่ร้องขอและส่งกลับมาในรูปของกระเบื้องแผนที่ จากนั้น OpenLayers จะนำภาพกระเบื้องแผนที่ที่ได้มาแสดงผ่านทางหน้าเว็บโดยต่อเป็นแผนที่เดียว

บทที่ 3
การวิเคราะห์และการออกแบบ

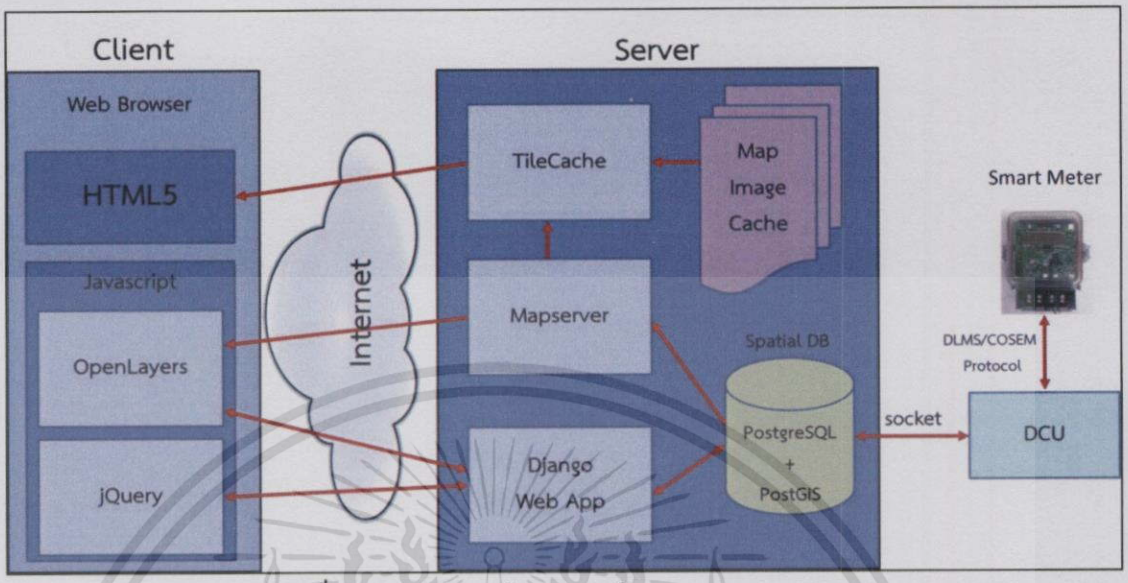
3.1 โครงสร้างของระบบโดยรวม

โครงสร้างของระบบสามารถแบ่งออกได้เป็น 3 ส่วน คือ สมาร์ทมิเตอร์ (Smart Meter) เซิร์ฟเวอร์ (Server Side) และไคลเอนต์ (Client Side) แสดงได้ดังรูปภาพด้านล่าง



รูปที่ 3.1 แสดงภาพรวมโครงสร้างของระบบ

การทำงานของระบบจะมีเครือข่ายของสมาร์ทมิเตอร์ที่จะมีการส่งข้อมูลค่าไฟฟ้ามายังเว็บเซิร์ฟเวอร์เพื่อการจัดเก็บข้อมูลและเว็บเซิร์ฟเวอร์จะคอยให้บริการข้อมูลแก่เว็บไคลเอนต์ และเมื่อเว็บไคลเอนต์ร้องขอบริการเข้ามาจะตอบกลับการร้องขอกลับไปยังไคลเอนต์



รูปที่ 3.2 แสดงสถาปัตยกรรมโดยรวมของระบบ

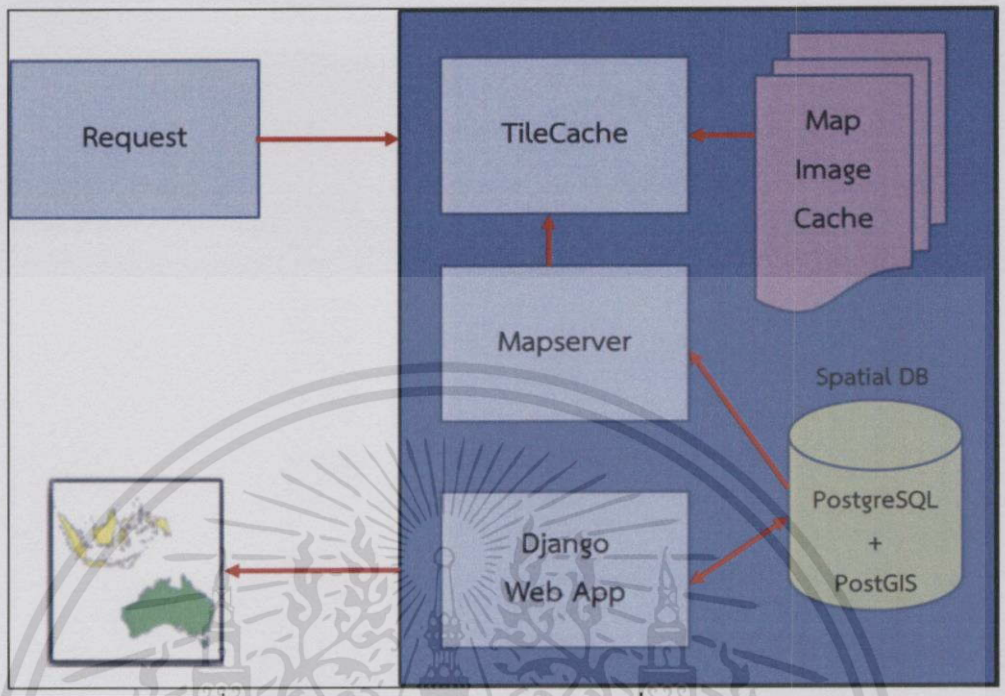
3.1.1 สมาร์ทมิเตอร์

ฝั่งของสมาร์ทมิเตอร์จะมีการส่งข้อมูลด้วยโปรโตคอล DLMS/COSEM ไปยัง DCU หลังจากนั้น DCU จะส่งข้อมูลผ่าน Socket เข้าไปยังเซิร์ฟเวอร์เพื่อเก็บข้อมูลลงดาต้าเบส สมาร์ทมิเตอร์จะสามารถอ่านค่าปริมาณการใช้ไฟฟ้าของผู้บริโภคและค่าพารามิเตอร์อื่นๆของไฟฟ้าที่จ่ายให้กับผู้บริโภค โดยรวมแล้วฝั่งสมาร์ทมิเตอร์จะมีหน้าที่ส่งข้อมูลมาจัดเก็บบนฐานข้อมูล

3.1.2 การออกแบบฝั่งของเซิร์ฟเวอร์

ฝั่งของเซิร์ฟเวอร์จะมีหน้าเว็บเซิร์ฟเวอร์ทำหน้าที่ให้บริการข้อมูลภูมิสารสนเทศ เพื่อรายงานการใช้พลังงานไฟฟ้าให้กับผู้ใช้งาน ผู้ใช้งานจะร้องขอบริการจากทางฝั่งไคลเอนต์ มาประมวลผลและส่งข้อมูลผลลัพธ์กลับไป ซึ่งการทำงานดังกล่าวจะมีเครือข่ายอินเตอร์เน็ตในการติดต่อสื่อสารทำให้สามารถเข้าถึงบริการเว็บเซิร์ฟเวอร์ได้ในทุกๆที่ที่มีอินเตอร์เน็ตใช้งานได้ ส่วนการทำงานภายในของเว็บเซิร์ฟเวอร์จะประกอบไปด้วย

1) เว็บแอพพลิเคชั่นจะประกอบไปด้วย (1) MapServer ทำหน้าที่ให้บริการภาพแผนที่ โดยสร้างจากฐานข้อมูลเชิงพื้นที่ (2) TileCache สำหรับเก็บภาพแผนที่ที่สร้างขึ้นเอาไว้ในแคช เมื่อมีการเรียกภาพแผนที่ที่เคยเรียกใช้งานแล้วก็สามารถนำภาพที่มีอยู่ในแคชไปแสดงผลได้ทันที (3) Web Django เป็น web application framework เพื่อใช้ในการพัฒนาเว็บแอพพลิเคชันและเว็บเซอร์วิสต่างๆ
2) ระบบจัดการฐานข้อมูลเชิงพื้นที่ จัดเก็บทั้งข้อมูลปกติและข้อมูลเชิงพื้นที่



รูปที่ 3.3 แสดงโครงสร้างการทำงานในฝั่งของเซิร์ฟเวอร์

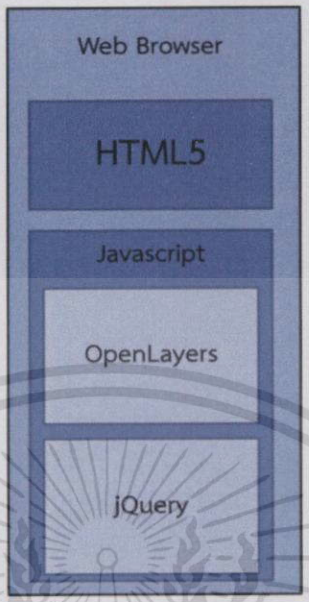
โดยรวมแล้วฝั่งของเซิร์ฟเวอร์มีหน้าที่ดังต่อไปนี้

1) รอรับการร้องบริการ (Request) จากฝั่งของไคลเอนต์
2) นำคำร้องขอมาประมวลผล
3) ติดต่อหรือดึงข้อมูลจากฐานข้อมูล
4) Web Map Service สร้างภาพแผนที่ (Render)
5) ส่งข้อมูลหรือผลลัพธ์ให้กับฝั่งของไคลเอนต์

3.1.3 การออกแบบฝั่งไคลเอนต์

ฝั่งไคลเอนต์ทำหน้าที่รับบริการข้อมูลภูมิสารสนเทศ เพื่อรายงานการใช้พลังงานไฟฟ้าให้กลับผู้ใช้งาน โดยฝั่งไคลเอนต์จะมีส่วนที่จะติดต่อรับส่งข้อมูลระหว่างผู้ใช้งานผ่านทางหน้าเว็บเบราว์เซอร์ของผู้ใช้งานในรูปแบบ HTML ซึ่งในการติดต่อกับฝั่งของเซิร์ฟเวอร์กระทำผ่านเครือข่ายอินเตอร์เน็ต โดยรวมแล้วฝั่งของไคลเอนต์มีหน้าที่ดังต่อไปนี้

1) รับคำสั่งจากผู้ใช้ผ่านเบราว์เซอร์
2) ส่งการร้องขอข้อมูลไปยังฝั่งเซิร์ฟเวอร์
3) รับข้อมูลหรือผลลัพธ์จากฝั่งเซิร์ฟเวอร์
4) นำข้อมูลแสดงผลแก่ผู้ใช้ผ่านเบราว์เซอร์

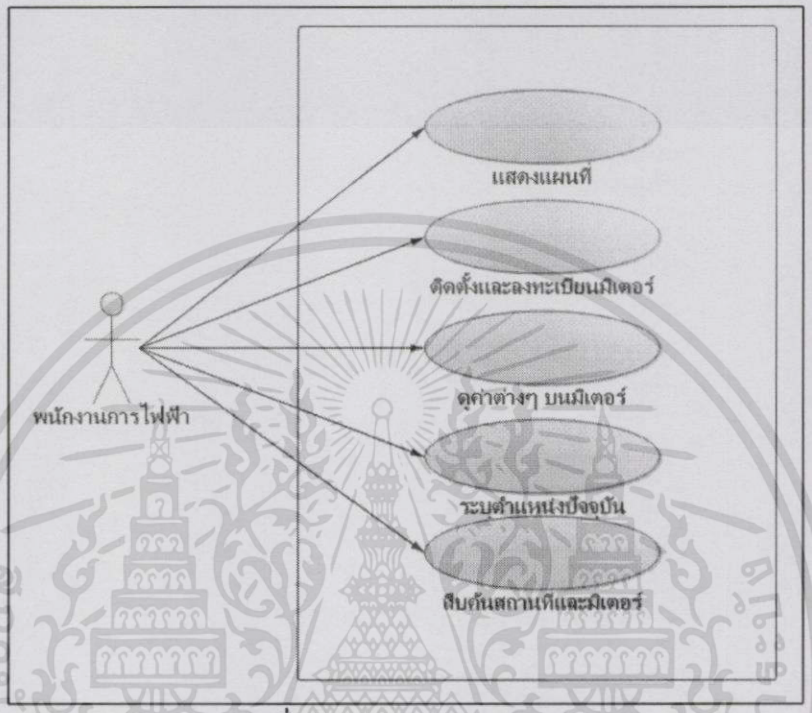


รูปที่ 3.4 แสดงโครงสร้างการออกแบบฝั่งไคลเอนต์

3.2 ความต้องการของระบบ

1) บริการข้อมูลทางภูมิสารสนเทศ
   - บริการเว็บแผนที่
   - บันทึก แก้ไข ข้อมูลแผนที่
   - ย่อ ขยาย เลื่อนภาพแผนที่
   - เปิด/ปิด ชั้นข้อมูลแผนที่
   - ค้นหาข้อมูลสถานที่
   - แสดงผลการใช้ไฟฟ้าของสมาร์ทมิเตอร์
   - แสดงตำแหน่งสมาร์ทมิเตอร์
   - แสดงสถานะสมาร์ทมิเตอร์
2) ฐานข้อมูล
   - สามารถเก็บข้อมูลเชิงพื้นที่ (Spatial Data)
   - มีฟังก์ชันคำสั่งที่เกี่ยวข้องกับการจัดการข้อมูลเชิงพื้นที่
   - มีการเก็บข้อมูลการใช้พลังงานไฟฟ้า
3) การรับข้อมูลจาก Smart Meter
   - รับข้อมูลผ่านทาง Socket
   - สามารถเพิ่มจำนวน Smart Meter ในแผนที่ได้

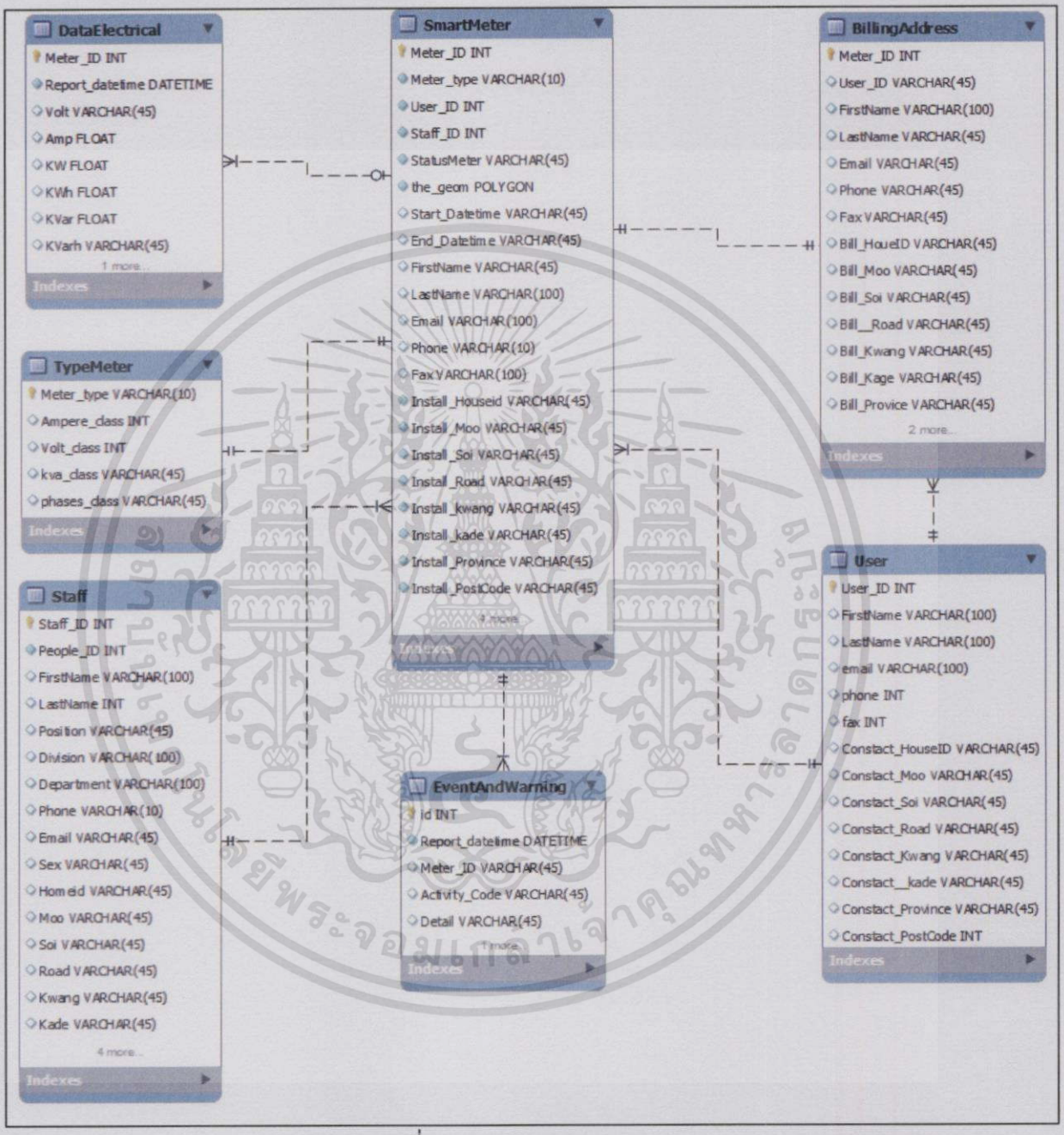
3.3 Use Case Diagram



รูปที่ 3.5 แสดง Use Case Diagram

1) แสดงแผนที่ เป็นการดูชั้นข้อมูลแผนที่ต่างๆ การเลื่อนดูขอบเขตการแสดงผลของแผนที่ การย่อ ขยายภาพแผนที่
2) ติดตั้งและลงทะเบียนมิเตอร์ เป็นการเพิ่มตัวสมาร์ทมิเตอร์ลงไปบนแผนที่โดยมีการกำหนดค่าต่างๆ ที่จำเป็น
3) ดูค่าต่างๆ บนมิเตอร์ เป็นการแสดงค่าต่างๆ ของสมาร์ทมิเตอร์ที่แสดงบนแผนที่ โดยที่สามารถแสดงผลตามที่ได้รับการรายงานการใช้พลังงานไฟฟ้า
4) ระบุตำแหน่งปัจจุบัน เป็นการระบุพิกัดของบุคคล อาจจะเป็นพนักงานติดตั้งตัวมิเตอร์ เพื่อช่วยในการระบุตำแหน่งของมิเตอร์บนแผนที่
5) สืบค้นสถานที่และมิเตอร์ แบ่งเป็นการค้นหาสถานที่และการค้นหาสมาร์ทมิเตอร์บนแผนที่

3.4 ER Diagram



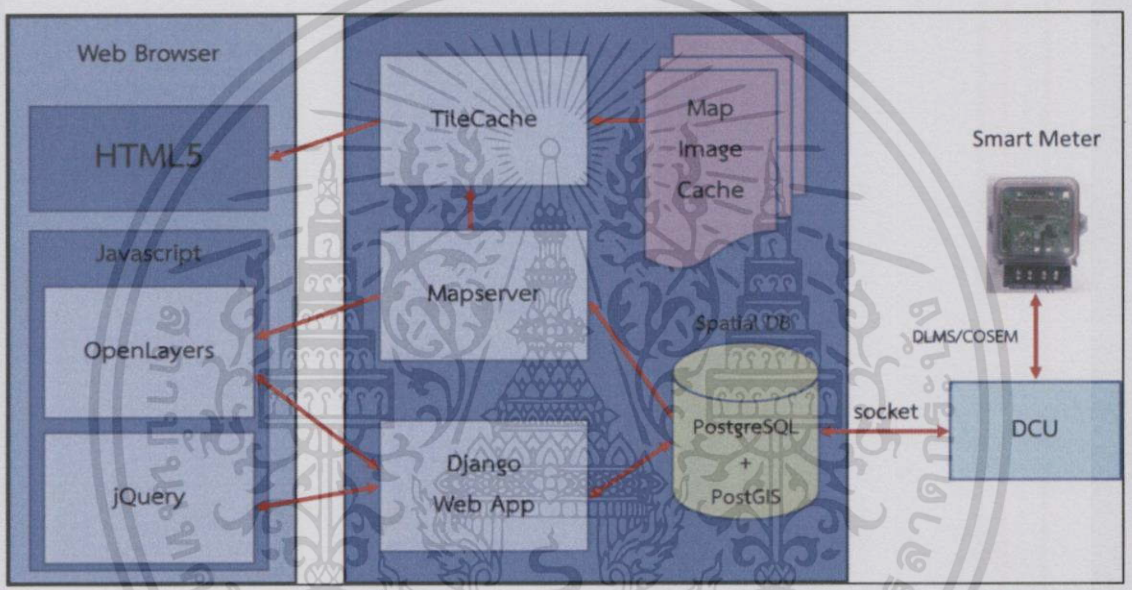
รูปที่ 3.6 แสดง ER Diagram

ในส่วนของการออกแบบฐานข้อมูลได้ทำการออกแบบฐานข้อมูลออกเป็น 7 ตารางดังนี้

1) ตาราง Smart Meter เป็นตารางที่เก็บข้อมูลของตัวสมาร์ทมิเตอร์ เช่น หมายเลขของสมาร์ทมิเตอร์, ตำแหน่งที่ติดตั้ง เป็นต้น
2) ตาราง DataElectrical เป็นตารางที่เก็บข้อมูลค่าไฟฟ้าที่ได้จากสมาร์ทมิเตอร์ เช่น หมายเลขของสมาร์ทมิเตอร์, ค่า volt, ค่า ampere, ค่า kW เป็นต้น โดยตาราง DataElectrical มีความสัมพันธ์กับตาราง SmartMeter แบบ many to one
3) ตาราง EventAndWarning เป็นตารางที่เก็บข้อมูลเหตุการณ์ที่เกิดขึ้นกับสมาร์ทมิเตอร์ เช่น วันและเวลาที่เกิดเหตุการณ์, โค๊ดของเหตุการณ์, รายละเอียดของเหตุการณ์ เป็นต้น โดยตาราง EventAndWarning มีความสัมพันธ์กับตาราง SmartMeter แบบ many to one
4) ตาราง BillingAddress เป็นตารางที่เก็บข้อมูลใบเสร็จชำระค่าไฟฟ้า เช่น ชื่อผู้ขอใช้ไฟฟ้า, ที่อยู่, เบอร์โทรศัพท์, ค่าไฟฟ้าที่ใช้, จำนวนเงินที่ต้องชำระ เป็นต้น โดยตาราง BillingAddress มีความสัมพันธ์กับตาราง SmartMeter แบบ one to one
5) ตาราง TypeMeter เป็นตารางที่เก็บข้อมูลประเภทของสมาร์ทมิเตอร์ เช่น ประเภทของมิเตอร์, ประเภทของ volt, ประเภทของ ampere เป็นต้น โดยตาราง TypeMeter มีความสัมพันธ์กับตาราง SmartMeter แบบ one to one
6) ตาราง User เป็นตารางที่เก็บข้อมูลของผู้ขอใช้ไฟฟ้า เช่น ชื่อ-นามสกุล, หมายเลขโทรศัพท์, ที่อยู่ เป็นต้น โดยตาราง User มีความสัมพันธ์กับตาราง SmartMeter แบบ one to many และตาราง User มีความสัมพันธ์กับตาราง BillingAddress แบบ one to many
7) ตาราง Staff เป็นตารางที่เก็บข้อมูลของพนักงานที่ติดตั้งตัวสมาร์ทมิเตอร์ เช่น รหัสพนักงาน, ชื่อ-นามสกุล, ที่อยู่, เบอร์โทรศัพท์ เป็นต้น โดยตาราง Staff มีความสัมพันธ์กับตาราง SmartMeter แบบ one to many

บทที่ 4
การพัฒนาระบบ

จากที่ได้อธิบายส่วนประกอบ การออกแบบโครงสร้าง และการทำงานของระบบในบทที่ 3 สามารถอธิบายขั้นตอนการพัฒนาและเทคโนโลยีที่ใช้ได้ดังต่อไปนี้



รูปที่ 4.1 แสดงสถาปัตยกรรมโดยรวมของระบบ

1) การนำเข้าและการสร้างฐานข้อมูลแผนที่
2) การสร้าง Map script เพื่อให้ Map Server สร้างภาพแผนที่ตามต้องการ
3) การสร้าง JavaScript เพื่อให้ OpenLayers แสดงภาพแผนที่บนหน้าเว็บเบราว์เซอร์
4) การสร้างฐานข้อมูลโดยใช้ Django
5) ในส่วนของ Web Application จะแบ่งเป็น
   5.1) การสืบค้นโดยใช้ JavaScript และ Django
   5.2) การติดต่อกับสมาร์ทมิเตอร์ผ่านทาง socket โดยใช้ Python

4.1 องค์ประกอบและเทคโนโลยีต่างๆ ในการพัฒนาระบบ

ในการพัฒนาระบบภูมิสารสนเทศเพื่อการบริหารจัดการสมาร์ทมิเตอร์มีการเลือกซอฟต์แวร์และระบบปฏิบัติการโอเพนซอร์สดังนี้

ตารางที่ 4.1 องค์ประกอบและเทคโนโลยีต่างที่ใช้งานในระบบ

องค์ประกอบ	เทคโนโลยีที่ใช้
ระบบปฏิบัติการ	Ubuntu Server 12.04 LTS
ส่วนจัดการฐานข้อมูล	PostgreSQL 9.1 PostGIS 2.0.4
ส่วนการสร้างภาพแผนที่	Mapserver 6.0.1
ส่วนการแสดงผลแผนที่	OpenLayers 2.13.1
ส่วนจัดการเว็บแอพพลิเคชั่น	Python 2.7 Django 1.5.4 Geodjango
เว็บเซิร์ฟเวอร์	Apache 2.0 HTML 5 JavaScript jQuery

4.2 การนำเข้าและการสร้างฐานข้อมูลแผนที่

ในการจัดเก็บข้อมูลสารสนเทศเชิงภูมิศาสตร์โดยใช้ซอฟต์แวร์ PostgreSQL และติดตั้งส่วนเสริม PostGIS ซึ่งมีขั้นตอนดังต่อไปนี้

1) การสร้างฐานข้อมูลชื่อ "meagisdb" ใช้คำสั่ง

```
#su postgres
$createdb meagisdb
```

2) สร้างฐานข้อมูลที่รองรับข้อมูลเชิงภูมิศาสตร์โดยใช้คำสั่ง

```
$createlang –d meagisdb plpgsql
$psql –d meagisdb –f postgis.sql
$psql –d meagisdb –f spatial_ref_sys.sql
$psql –d meagisdb –f legacy.sql
```

เมื่อได้ฐานข้อมูลที่สามารถรองรับข้อมูลภูมิสารสนเทศได้แล้ว ขั้นตอนต่อไปจะเป็นการตั้งค่าระบบจัดการฐานข้อมูล PostgreSQL ให้สามารถเรียกใช้งานได้จากผู้ใช้และจากไอพีแอดเดรสที่กำหนด โดยสร้าง Group role แล้วค่อยมอบสิทธิ์ให้ user

3) สร้าง group role ที่ชื่อ gisgroup ขึ้นมา จำกัดสิทธิ์ให้สร้าง Database ได้เท่านั้น

```
#CREATE ROLE gisgroup NOSUPERUSER NOINHERIT CREATEDB NOCREATEROLE;
```

4) สร้าง user ชื่อ "gis" ด้วยพาสเวิร์ด 'passw0rd'

```
#CREATE ROLE gis LOGIN PASSWORD 'passw0rd' NOINHERIT;
```

5) แล้วมอบสิทธิ์หรือบทบาทจาก gisgroup ให้ด้วยคำสั่ง GRANT

```
#GRANT gisgroup TO gis;
```

6) ตั้งค่าให้ PostgreSQL Server ให้รองรับการใช้งานจากทุกไอพีแอดเดรส โดยเข้าไปแก้ไขไฟล์ที่ /etc/postgresql/9.1/main/pg_hba.conf โดยแก้ไขจาก

```
Listen_addresses='localhost'
```

แก้ไขเป็น

```
Listen_addresses='*'
```

เมื่อได้ฐานข้อมูลที่สามารถรองรับข้อมูลภูมิสารสนเทศได้ตั้งค่าผู้ใช้งานได้แล้ว ขั้นตอนต่อไปจะเป็นการนำเข้าข้อมูลทางภูมิสารสนเทศที่ได้เลือกใช้ข้อมูลแผนที่ที่ได้เป็นตัวอย่างจากการไฟฟ้านครหลวง อยู่ในรูปแบบไฟล์ ESRI Shape File (“.shp”) เข้าสู่ฐานข้อมูล meagisdb ที่ได้สร้างไว้ในขั้นตอนก่อนหน้า

7) ตัวอย่างการใช้คำสั่ง shp2pgsql เพื่อนำเข้าข้อมูลเชิงพื้นที่

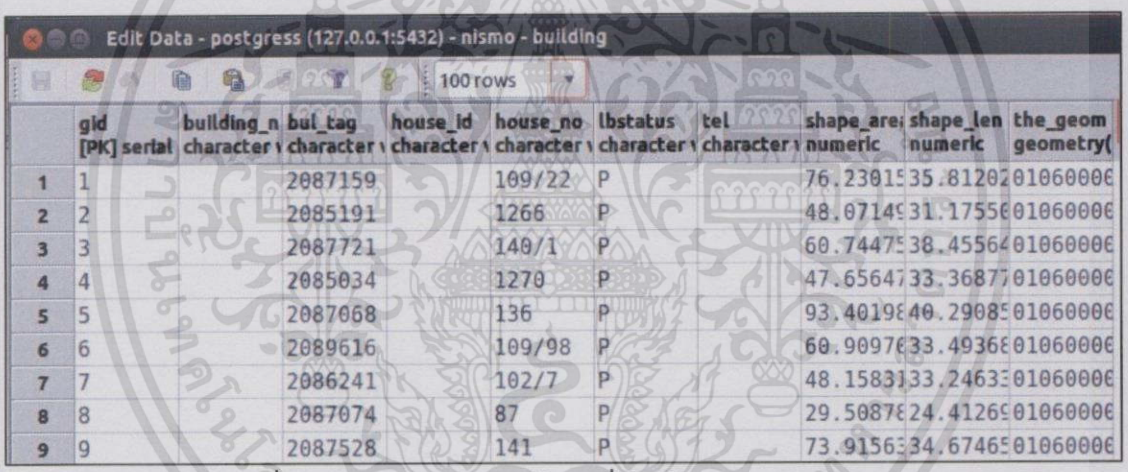
```
Shp2pgql –s 32647 –W windows-874 –g the_geom Building.shp Building | psql
–h 127.0.0.1 –d meagisdb –U pis
```

สามารถอธิบายคำสั่งได้ดังนี้

Shp2pgql	คือ คำสั่งแปลงไฟล์ ESRI Shape file เป็น SQL
–s 32647	คือ SRID เป็น EPSG:32647
–W windows-874	คือ specify encoding เป็น windows-874
–g the_geom	คือ คำสั่งในการกำหนดแอททริบิวท์ Geometry ให้เป็นชื่อ ‘the_geom’
Building.shp	คือ ไฟล์อินพุต
Building	คือ ชื่อตารางเอาท์พุต

| (pipe)	คือ การนำเอาท์พุตคำสั่งข้างซ้าย ไปเป็นอินพุตคำสั่งข้างขวา
psql	คือ psql เป็นคำสั่งติดต่อฐานข้อมูล PostgreSQL
–h 127.0.0.1	คือ แอดเดรสของโฮสฐานข้อมูล
–d meagisdb	คือ ชื่อฐานข้อมูล
–U pis	คือ ชื่อ User

เมื่อนำข้อมูลเข้าสู่ตารางแล้ว เราจะสามารถเปิดดูข้อมูลได้จากโปรแกรม pgAdmin3 ซึ่งจะแสดงแอททริบิวท์ทั้งหมด โดยมีแอททริบิวท์ “gid” เป็นคีย์หลักและมีแอททริบิวท์ที่สำคัญคือ แอททริบิวท์ “the_geom” เก็บข้อมูลทางภูมิสารสนเทศ


Edit Data - postgress (127.0.0.1:5432) - nismo - building

100 rows

	gid [PK] serial	building_n character v	bul_tag character v	house_id character v	house_no character v	lbstatus character v	tel character v	shape_are numeric	shape_len numeric	the_geom geometry(
1	1		2087159		109/22	P		76.2301	35.8120	0106000(
2	2		2085191		1266	P		48.0714	31.1755	0106000(
3	3		2087721		140/1	P		60.7447	38.4556	0106000(
4	4		2085034		1270	P		47.6564	33.3687	0106000(
5	5		2087068		136	P		93.4019	40.2908	0106000(
6	6		2089616		109/98	P		60.9097	33.4936	0106000(
7	7		2086241		102/7	P		48.1583	33.2463	0106000(
8	8		2087074		87	P		29.5087	24.4126	0106000(
9	9		2087528		141	P		73.9156	34.6746	0106000(

รูปที่ 4.2 แสดงตารางฐานข้อมูลที่นำเข้ามาจาก ESRI Shape File

4.3 การสร้าง Map script เพื่อให้ Map Server สร้างรูปแผนที่ตามต้องการ

การนำข้อมูลภูมิสารสนเทศมาวาดเป็นรูปภาพแผนที่โดยใช้ซอฟท์แวร์ MapServer สามารถที่จะใช้แหล่งข้อมูลที่มีความแตกต่างกันได้ ทั้งชนิดของข้อมูลและแหล่งที่มา ซึ่งก็อาจจะใช้การตั้งค่าที่แตกต่างกันไปด้วย แต่สิ่งหนึ่งที่ต้องคำนึงถึงก็คือ ในแต่ละ mapfile ถ้าจะนามาใช้ร่วมกัน จำเป็นต้องใช้รหัสอ้างอิงทางภูมิสารสนเทศเดียวกันจึงจะสามารถนามาซ้อนทับกันได้อย่างถูกต้องแม่นยำ โดย mapfile ที่ใช้ได้มีการโปรแกรมในส่วนหัวไว้เหมือนกันดังนี้

```
MAP #
    NAME "WMS_server"
    FONTSET "fonts.list"
    SYMBOLSET "symbol.list"
    EXTENT 660664.575039127 1514276.3509819 673102.579
    #EXTENT 660669.88 1514879.38 672683.25 1523011.12
    UNITS METERS
    STATUS ON
    PROJECTION
        "init=epsg:32647"
    END
    IMAGETYPE PNG

    OUTPUTFORMAT
        NAME 'AGGA'
        MIMETYPE "image/png"
        DRIVER AGG/PNG
        IMAGEMODE RGBA
        FORMATOPTION "QUANTIZE_FORCE=on"
        FORMATOPTION "COMPRESSION=9"
    END
    WEB
        METADATA
            "wms_enable_request"          "*"
            "wfs_title"                   "WFS Demo Ser
            "wfs_onlineresource"          "192.168.70.1
            "wfs_encoding"                "UTF-8"
            "wms_feature_info_mime_type" "text/html"
            "wms_srs"                     "EPSG:900913"
            #"wms_format"                 "image/png"
        END
    END
```

กำหนดค่าต่างให้กับแผนที่ให้ทุกเลเยอร์ใช้รวมกัน เช่น ชื่อแผนที่ พิกัดขอบเขตแผนที่หน่วยวัด (scale) Status projection เป็นต้น

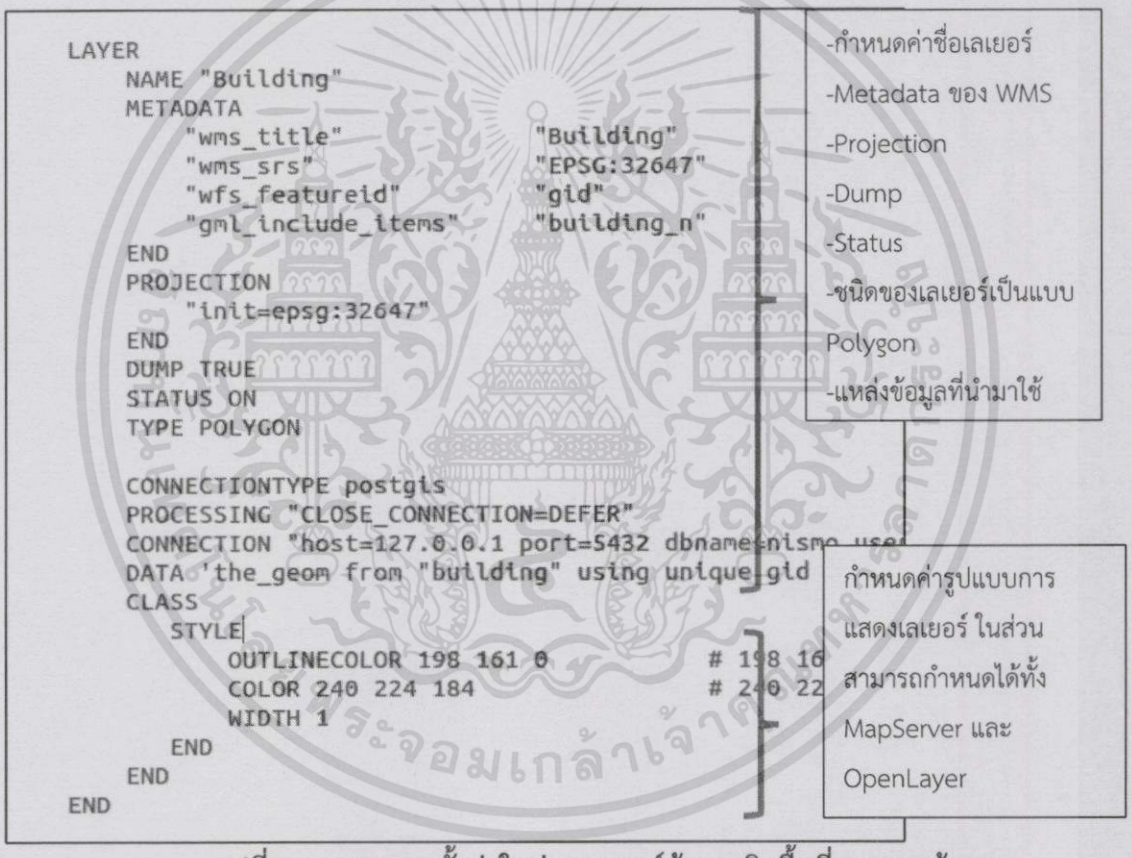
กำหนดค่ารูปแบบของไฟล์ที่ได้จาก MapServer ของแต่ละชนิดไฟล์

กำหนดค่า metadata ของ WMS ในการเรียกใช้หรือเข้าถึง Mapfile

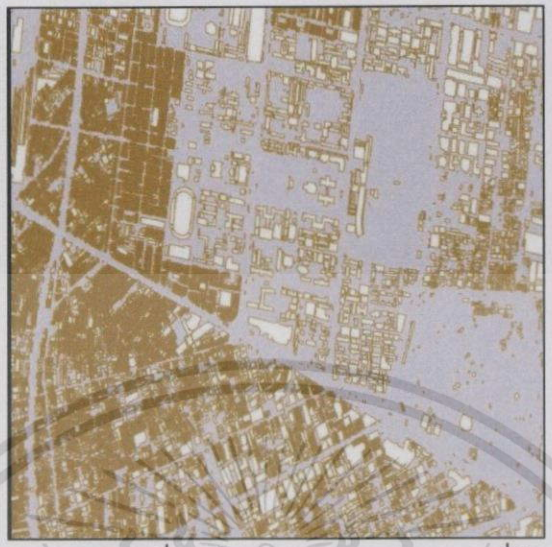
รูปที่ 4.3 แสดงการตั้งค่าส่วนหัวของ mapfile

ต่อไปจะกล่าวถึงการโปรแกรมในส่วนของเลเยอร์จะนำมาใช้จะถูกเก็บอยู่ในฐานข้อมูล PostGIS ดังนั้นจึงต้องกำหนดค่า CONNECTIONTYPE ใน LAYER OBJECT เป็น postgis และกำหนดค่า CONNECTION เป็นชื่อและที่อยู่ของฐานข้อมูลรวมถึงชื่อผู้ใช้และรหัสผ่านที่สามารถทำให้เข้าถึงฐานข้อมูลได้ และที่สำคัญที่สุด คือ DATA ที่ต้องระบุคอลัมน์และตารางที่ต้องการด้วยรหัส SRID ที่ถูกต้องด้วย อธิบายการเขียนแต่ละชนิดเลเยอร์ที่นำมาใช้ทั้ง ข้อมูลเชิงพื้นที่ ข้อมูลเชิงเส้น ข้อมูลแบบจุด ได้ดังนี้

1) ข้อมูลเชิงพื้นที่ของอาคารสถานที่



รูปที่ 4.4 แสดงการตั้งค่าในส่วนเลเยอร์ข้อมูลเชิงพื้นที่จากฐานข้อมูล



รูปที่ 4.5 แสดงภาพแผนที่จากโปรแกรม MapServer ที่มีเลเยอร์ข้อมูลเชิงพื้นที่

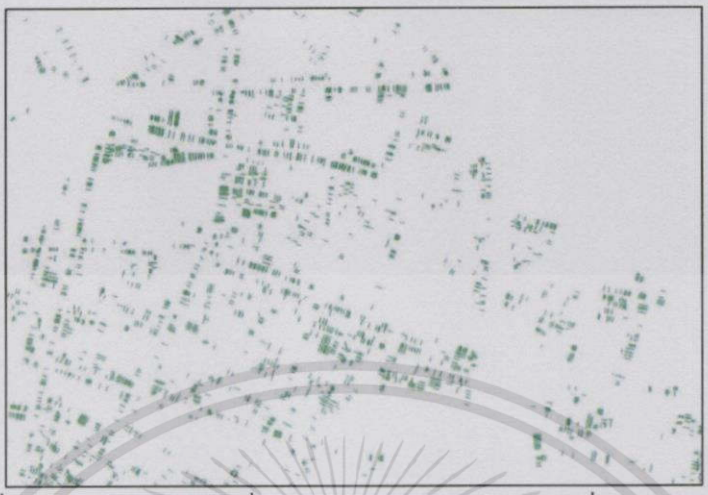
2) ข้อมูลเชิงเส้นของขอบของอาคารสถานที่

```
LAYER
    NAME "BuildingLine"
    METADATA
        "wfs_title"             "BuildingLine"
        "wfs_srs"               "EPSG:32647"
        "wfs_featureid"         "gid"
        "gml_include_items"     "all"

    END
    PROJECTION
        "init=epsg:32647"
    END
    DUMP TRUE
    STATUS ON
    TYPE line

    CONNECTIONTYPE postgis
    PROCESSING "CLOSE_CONNECTION=DEFER"
    CONNECTION "host=127.0.0.1 port=5432 dbname=nismo
er=postgres password=passw0rd"
    DATA 'the_geom from "buildingline" using unique g
ID=32647'
    CLASS
        OVERLAYSYMBOL "DashLine10_4"
        OVERLAYSIZE 1
        OVERLAYCOLOR 0 0 255
    END
```

-กำหนดค่าชื่อเลเยอร์
-Metadata ของ WMS
-Projection
-Dump
-Status
-ชนิดของเลเยอร์เป็นแบบเส้น(Line)
-แหล่งข้อมูลที่นำมาใช้

กำหนดค่ารูปแบบการแสดงเลเยอร์ ในส่วนสามารถกำหนดได้ทั้ง MapServer และ OpenLayer

รูปที่ 4.6 แสดงการตั้งค่าในส่วนเลเยอร์ข้อมูลเชิงเส้นจากฐานข้อมูล



รูปที่ 4.7 แสดงภาพแผนที่จากโปรแกรม MapServer ที่มีเลเยอร์ข้อมูลเชิงเส้น

3) ข้อมูลเชิงจุดของจุดที่มีการปักหมุด

```
    LAYER
        NAME "AnchorGuy"
        METADATA
            "wfs_title"             "AnchorGuy"
            "wfs_srs"               "EPSG:32647"
            "gml_include_items"     "all"
            #"gml_name_alias"       "all"
        END
        PROJECTION
            "init=epsg:32647"
        END
        DUMP TRUE
        STATUS ON
        TYPE POINT

        CONNECTIONTYPE postgis
        PROCESSING "CLOSE_CONNECTION=DEFER"
        CONNECTION "host=127.0.0.1 port=5432 dbname=nis
user=postgres password=passw0rd"
        DATA 'the_geom from "anchorguy" using unique gi
SRID=32647'
        CLASS
            OVERLAYSYMBOL "AnchorGuySymbol"
            OVERLAYSIZE 20
        END
    END
```

-กำหนดค่าชื่อเลเยอร์
-Metadata ของ WMS
-Projection
-Dump
-Status
-ชนิดของเลเยอร์เป็นแบบจุด(Point)
-แหล่งข้อมูลที่นำมาใช้

กำหนดค่ารูปแบบการแสดงเลเยอร์ ในส่วนสามารถกำหนดได้ทั้ง MapServer และ OpenLayer

รูปที่ 4.8 แสดงการตั้งค่าในส่วนเลเยอร์ข้อมูลแบบจุดจากฐานข้อมูล

รูปที่ 4.9 แสดงภาพแผนที่จากโปรแกรม MapServer ที่มีเลเยอร์ข้อมูลแบบจุด

4.4 การสร้าง JavaScripy เพื่อให้ OpenLayers แสดงภาพแผนที่บนหน้าเว็บเบราว์เซอร์

OpenLayers เป็นส่วนหลักในการแสดงผลแผนที่ โดยกำหนดค่าแท็กดิพ ให้กับไฟล์ HTML แล้วจึงสร้างแผนที่ด้วยจาวาสคริปต์ซึ่งมีขั้นตอนดังนี้

1) ใส่ฟังก์ชัน init(); ให้ทำการเรียกใช้ตอนหน้าเพจถูกโหลดขึ้นมา
2) ในฟังก์ชัน init(); มีการสร้างตัวแปร map ให้เป็น OpenLayers.map()
3) กำหนด option ให้กับ map

```
var map;

function init() {
        map = new OpenLayers.Map(
                'map_element', {
                displayprojection: 'EPSG:4326',
                numZoomLevels: 20,
                minscale:1128,
                maxScale:591658711,
                unit: 'm'
                });
```

ตั้งค่า option ให้กับตัวแปร map โดยมีการกำหนด projection ขนาดสเกลในการย่อ/ขยาย

รูปที่ 4.10 แสดงการกำหนดค่าให้กับ map เพื่อใช้งาน OpenLayers

4) สร้างตัวแปรและกำหนดเลเยอร์ต่างๆ

```
var wms = new OpenLayers.Layer.WMS(
'building',
'http://www.nismo.net/cgi-bin/meagis.cgi',
{layers: "building", map: "/home/nismo/www/demo.map"},
{}
);
```

รูปที่ 4.11 แสดงการกำหนดค่าให้เลเยอร์ต่าง ๆ ในการใช้งาน OpenLayers

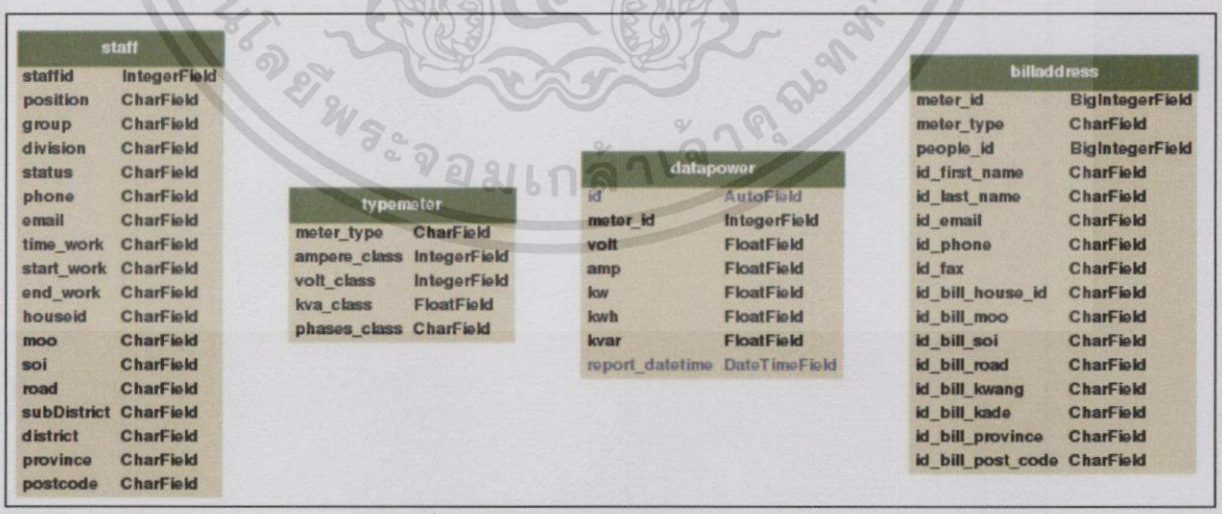
5) เพิ่มเลเยอร์ที่ประกาศไว้และคอนโทรลต่างๆ ให้กับ map

```
map.addLayer(wms);
if(!map.getCenter()){
map.zoomToMaxExtent();
}
}
```

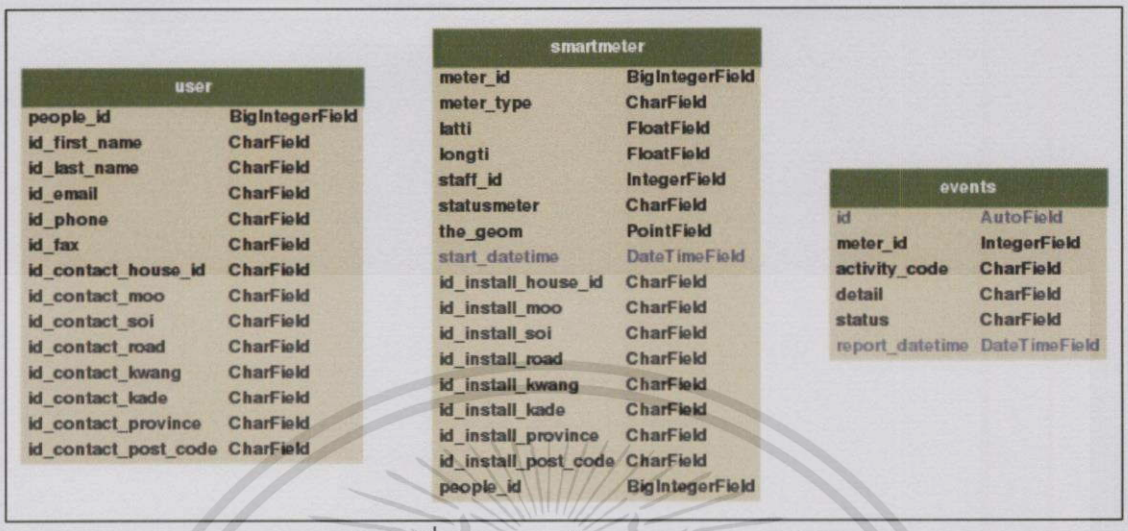
รูปที่ 4.12 การเพิ่มเลเยอร์และคอนโทรล

4.5 การสร้างฐานข้อมูลโดยใช้ Django

ในส่วนของการพัฒนาฐานข้อมูลนั้นได้พัฒนาโดยใช้ Django เขียนเป็นคลาสของฐานข้อมูลและกำหนด attribute ต่างๆ



รูปที่ 4.13 ฐานข้อมูลจาก Django (1)



user	
people_id	BigIntegerField
id_first_name	CharField
id_last_name	CharField
id_email	CharField
id_phone	CharField
id_fax	CharField
id_contact_house_id	CharField
id_contact_moo	CharField
id_contact_soi	CharField
id_contact_road	CharField
id_contact_kwang	CharField
id_contact_kade	CharField
id_contact_province	CharField
id_contact_post_code	CharField

smartmeter	
meter_id	BigIntegerField
meter_type	CharField
latti	FloatField
longti	FloatField
staff_id	IntegerField
statusmeter	CharField
the_geom	PointField
start_datetime	DateTimeField
id_install_house_id	CharField
id_install_moo	CharField
id_install_soi	CharField
id_install_road	CharField
id_install_kwang	CharField
id_install_kade	CharField
id_install_province	CharField
id_install_post_code	CharField
people_id	BigIntegerField

events	
id	AutoField
meter_id	IntegerField
activity_code	CharField
detail	CharField
status	CharField
report_datetime	DateTimeField

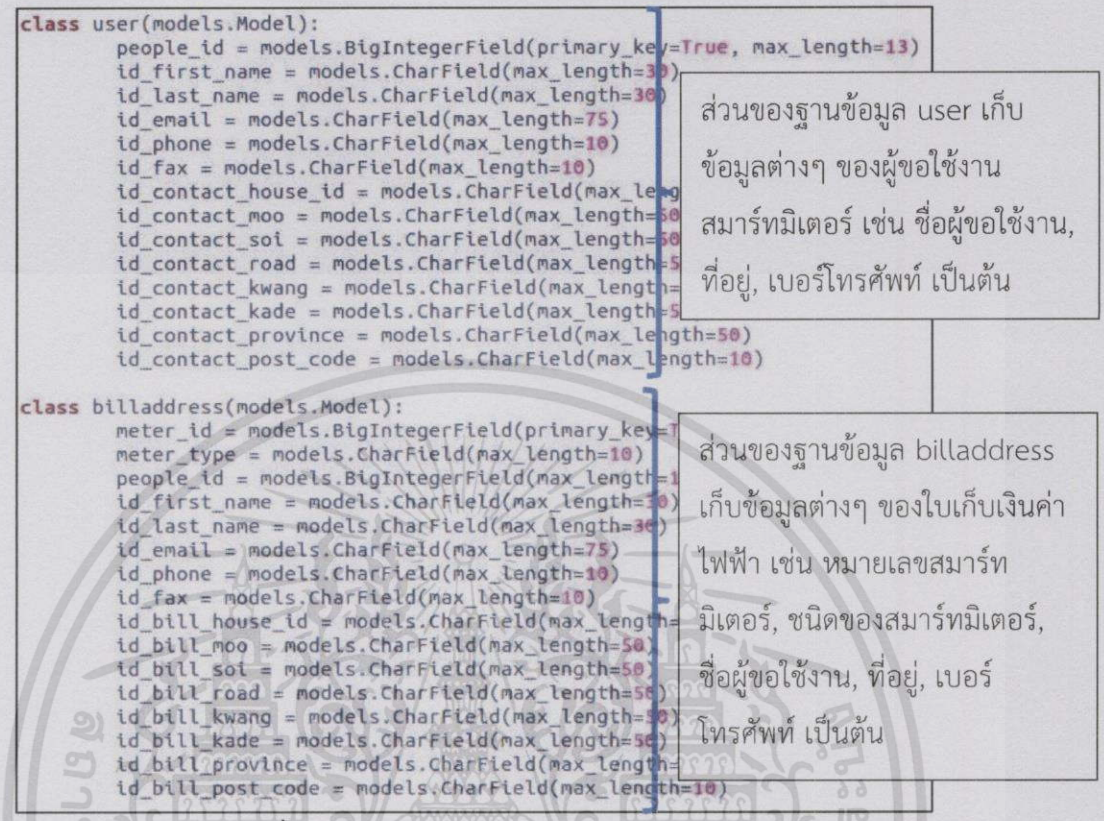
รูปที่ 4.14 ฐานข้อมูลจาก Django (2)

```
# _*_ encoding: utf-8 _*_
from django.db import models
from django.contrib.gis.db import models

class events(models.Model):
        meter_id = models.IntegerField()
        activity_code = models.CharField(max_length=10)
        detail = models.CharField(max_length=100)
        status = models.CharField(max_length=15)
        report_datetime = models.DateTimeField(auto_now=True)

class smartmeter(models.Model):
        meter_id = models.BigIntegerField(primary_key=True, max_length=10)
        meter_type = models.CharField(max_length=10)
        latti = models.FloatField()
        longti = models.FloatField()
        staff_id = models.IntegerField(max_length=10)
        statusmeter = models.CharField(max_length=15)
        the_geom = models.PointField(srid=32647)
        start_datetime = models.DateTimeField(auto_now=True)
        id_install_house_id = models.CharField(max_length=10)
        id_install_moo = models.CharField(max_length=10)
        id_install_soi = models.CharField(max_length=10)
        id_install_road = models.CharField(max_length=10)
        id_install_kwang = models.CharField(max_length=10)
        id_install_kade = models.CharField(max_length=10)
#       subDistrict = models.CharField(max_length=50)
#       district = models.CharField(max_length=50)
        id_install_province = models.CharField(max_length=10)
        id_install_post_code = models.CharField(max_length=10)
        people_id = models.BigIntegerField(max_length=13)
        def __unicode__(self):
                return self.the_geom
```

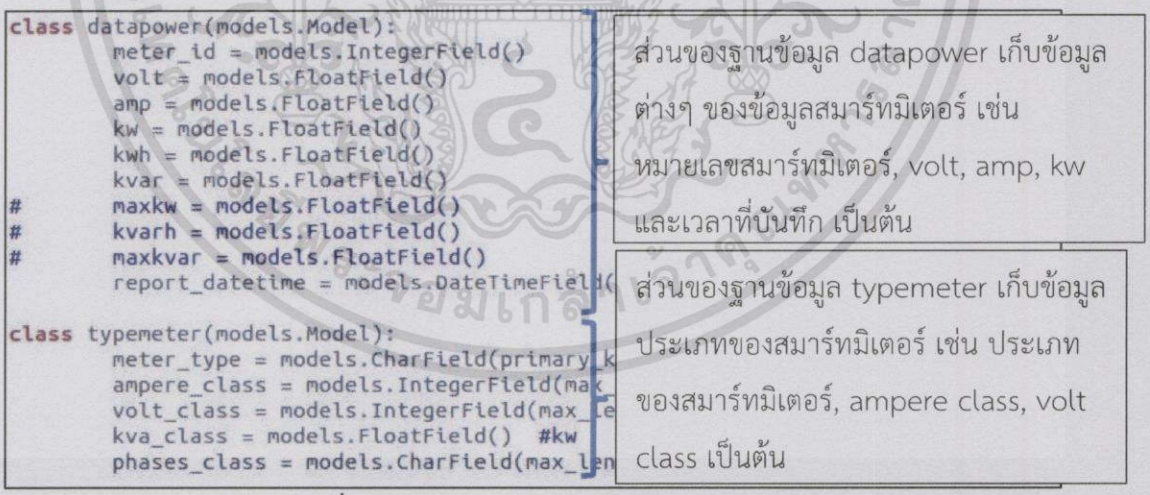
ส่วนของฐานข้อมูล event เก็บสถานะต่างๆ ของสมาร์ทมิเตอร์

ส่วนของฐานข้อมูล smartmeter เก็บข้อมูลต่างๆ ของสมาร์ทมิเตอร์ เช่น หมายเลขสมาร์ทมิเตอร์, ชนิดของสมาร์ทมิเตอร์, ตำแหน่งของสมาร์ทมิเตอร์, หมายเลขผู้ติดตั้ง เป็นต้น

รูปที่ 4.15 การสร้างฐานข้อมูลโดยใช้ Django (1)



รูปที่ 4.16 การสร้างฐานข้อมูลโดยใช้ Django (2)



รูปที่ 4.17 การสร้างฐานข้อมูลโดยใช้ Django (3)

```
class staff(models.Model):
        staffid = models.IntegerField(primary_key=True)
        position =  models.CharField(max_length=50)
        group =  models.CharField(max_length=50)
        division =  models.CharField(max_length=50)
        status =  models.CharField(max_leng
        phone =  models.CharField(max_lengt
        email =  models.CharField(max_lengt
        time_work =  models.CharField(max_le
        start_work =  models.CharField(max_
        end_work = models.CharField(max_len
        houseid = models.CharField(max_leng
        moo = models.CharField(max_length=1
        soi = models.CharField(max_length=1
        road = models.CharField(max_length=20)
        subDistrict = models.CharField(max_length=20)
        district = models.CharField(max_length=20)
        province = models.CharField(max_length=20)
        postcode = models.CharField(max_length=10)
```

ส่วนของฐานข้อมูล staff เก็บข้อมูลของพนักงานที่ทำการติดตั้ง เช่น หมายเลขพนักงาน, ตำแหน่ง, เบอร์โทรศัพท์, เวลาทำงาน, ที่อยู่ เป็นต้น

รูปที่ 4.18 การสร้างฐานข้อมูลโดยใช้ Django (4)

4.6 การสืบค้นโดยใช้ JavaScript และ Django

ในส่วนของการสืบค้นข้อมูลต่างๆ ได้พัฒนาโดยใช้ JavaScript และ Django โดยในส่วนของ JavaScript จะเป็นการรับค่าจากผู้ใช้งานแล้วส่งค่าไปยัง Django เพื่อไปคิวรี่ข้อมูลจากฐานข้อมูล จากนั้นส่งค่ากลับมายัง JavaScript เพื่อแสดงผลลัพธ์ที่ค้นหา

```
function nSearch()
{
        var choose = document.getElementById("idsearch0").value;
        var Item = document.getElementById("nsearch").value;
        if (Item == ""){
                alert("Please fill in Search!");
        }else if(choose =="Meter"){
                var req = new XMLHttpRequest();
                req.open('GET', '/metersearch?meterid='+Item+'', fa
                req.send(null);
                if(req.status == 0){
                        dump(req.responseText);
                }
                obj = JSON.parse(req.responseText);
                if(obj.status == true){
                        move_to(obj.lat, obj.lon);
                }else if(obj.status ==false){
                        alert("Meter ID Not Found!");
                }else{
                        alert("Error Search Engine");
                }
        }else if(choose == "Building" || choose =="Centerline"){        //search building & centerline
                var req = new XMLHttpRequest();
                req.open('GET', '/metersearch?choose='+choose+'&search='+Item+'', false);
                req.send(null);
                if(req.status == 0){
                        dump(req.responseText);
                }
                obj = JSON.parse(req.responseText);
                if(obj.status == true){
                        move_to(obj.lat, obj.lon);
                }else if(obj.status == false){
                        alert("Not Found!");
                }else{
                        alert("Error Search Engine");
```

ส่วนของการสืบค้นสมาร์ทมิเตอร์ โดยค่าที่รับเข้ามาจะเป็นหมายเลขของสมาร์ทมิเตอร์

ส่วนของการสืบค้นสถานที่ ถนน และตัวอาคาร

รูปที่ 4.19 การสืบค้นโดยใช้ JavaScript

```
def metersearch(request):
    result = {'status' : False}
    if request.GET.has_key('meterid'):
        if request.GET.get('meterid')!=None:
                meterid = request.GET.get('meterid')
                for d in smartmeter.objects.raw("select meter_id from meter_smartmeter where mete
                        result['status'] = True
                        result['meterid'] = d.meter_id
                        result['lon'] = d.longti
                        result['lat'] = d.latti
                return HttpResponse(simplejson.dumps(result), mimetype='application/json')
    elif request.GET.has_key('choose'):
        if request.GET.get('choose')!=None:
                if request.GET.get('search')!=None:
                        choose = request.GET.get('choose')
                        search = request.GET.get('search')
                        if choose=="Building":
                            for d in Building.objects.raw("select gi
(ST_AsText(ST_Transform(ST_Centroid(the_geom),4326))) as y  from bui
                                result['status'] = True
                                result['gid'] = d.gid
                                result['lon'] = d.x
                                result['lat'] = d.y
                        elif choose=="Centerline":
                            for d in Centerline.objects.raw("select gid, ST_X(ST_AsText(ST_Transfo
(ST_AsText(ST_Transform(ST_Centroid(the_geom),4326))) as y from centerline where name='"+search+"
                                result['status'] = True
                                result['gid'] = d.gid
                                result['lon'] = d.x
                                result['lat'] = d.y
                        elif choose=="Location":
                            for d in LandmarkPoint.objects.raw("select gid, ST_X(ST_AsText(ST_Tra
ST_Y(ST_AsText(ST_Transform(ST_Centroid(the_geom),4326))) as y  from landmarkpoint where location
                                result['status'] = True
                                result['gid'] = d.gid
```

ส่วนของ Django จะทำการการรับค่าจาก JavaScript แล้วนำไปคิวรี่ในฐานข้อมูล จากนั้นก็ส่งกลับไปให้ JavaScript

รูปที่ 4.20 การสืบค้นโดยใช้ Django

4.7 การติดต่อกับสมาร์ทมิเตอร์ผ่านทาง Socket โดยใช้ Python

การติดต่อกับสมาร์ทมิเตอร์จะใช้ Socket ซึ่งพัฒนาโดยใช้ภาษา Python

```
#!/usr/bin/python           # This is client.py file
# -*- coding: utf-8 -*-
import socket               # Import socket module
import threading
import time
import psycopg2
import psycopg2.extras
import sys
import datetime

current_time = datetime.datetime.now()
current_time1 = str(current_time).split(".")

class countThread (threading.Thread):
    def __init__(self,o = '',n = 1):
        threading.Thread.__init__(self)
        self.n = n
        self.o = o

    def run(self):
        time.sleep(self.n)
        self.o.stopIt()

class clientTread (threading.Thread):
    def __init__(self):
        threading.Thread.__init__(self)
        self.s = socket.socket()        # Create a socket object
        self.host = '192.168.1.90' # Get local machine name
        self.port = 12345               # Reserve a port for your servi
        self.play = True
```

ส่วนที่ import library ที่ต้องใช้งาน

ส่วนที่สร้างการเชื่อมต่อกับฝั่งของสมาร์ทมิเตอร์

รูปที่ 4.21 การติดต่อกับสมาร์ทมิเตอร์ผ่านทาง Socket โดยใช้ Python (1)

```
    def run(self):
        self.s.connect((self.host, self.port))
        print self.s.recv(1024)
        self.play = True
        k = 'read'
        while self.play:
            self.s.send(k)
            data = self.s.recv(1024)
            print data
            data = data.split(",")
            conn_string = "host='localhost' dbname='nismo' user='postgres' pa
            conn = psycopg2.connect(conn_string)
            cursor = conn.cursor()
            cursor.execute("INSERT INTO meter_datapower(meter_id,volt,amp,kw,
[1])+", "+str(data[2])+", "+str(data[3])+", "+str(data[4])+", to_timestamp("
            conn.commit()
            time.sleep(2)
        self.s.send('end')
        self.s.close

    def stopIt(self):
        self.play = False

def main():
    cliT = clientThread()
    couT = countThread(cliT,30)
    cliT.start()
    couT.start()


if __name__ == "__main__":
    main()
```

ส่วนที่รับค่าและเพิ่มเข้าไปในฐานข้อมูล

รูปที่ 4.22 การติดต่อกับสมาร์ทมิเตอร์ผ่านทาง Socket โดยใช้ Python (2)

บทที่ 5
การทดลองและผลการทดลอง

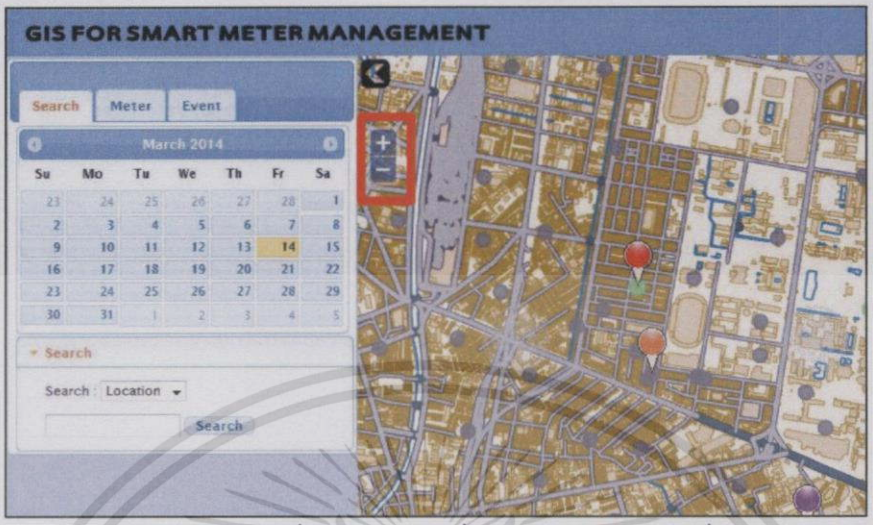
หลังจากที่ได้มีการติดตั้งระบบและพัฒนาโปรแกรมต่างๆ แล้วได้มีการทดลองในฝั่งของเซิร์ฟเวอร์โดยใช้เครื่องเซิร์ฟเวอร์รุ่น Dell PowerEdge 2950 ในส่วนของฝั่งไคลเอนต์ได้ใช้เว็บเบราว์เซอร์ Mozilla FireFox เวอร์ชัน 27.0.1 ซึ่งเว็บแอพพลิเคชั่นที่ได้จัดทำขึ้นสามารถให้บริการได้ดังนี้

- ย่อ ขยาย เลื่อนภาพแผนที่
- เปิด/ปิด ชั้นข้อมูลแผนที่
- ค้นหาข้อมูลสถานที่และสมาร์ทมิเตอร์
- ติดตั้งและลงทะเบียนสมาร์ทมิเตอร์
- การขอข้อมูลปริมาณการใช้ไฟฟ้าจากสมาร์ทมิเตอร์
- แสดงสถานะสมาร์ทมิเตอร์
- ระบุตำแหน่งปัจจุบัน

5.1 บริการแผนที่

5.1.1 ย่อ ขยาย เลื่อนแผนที่

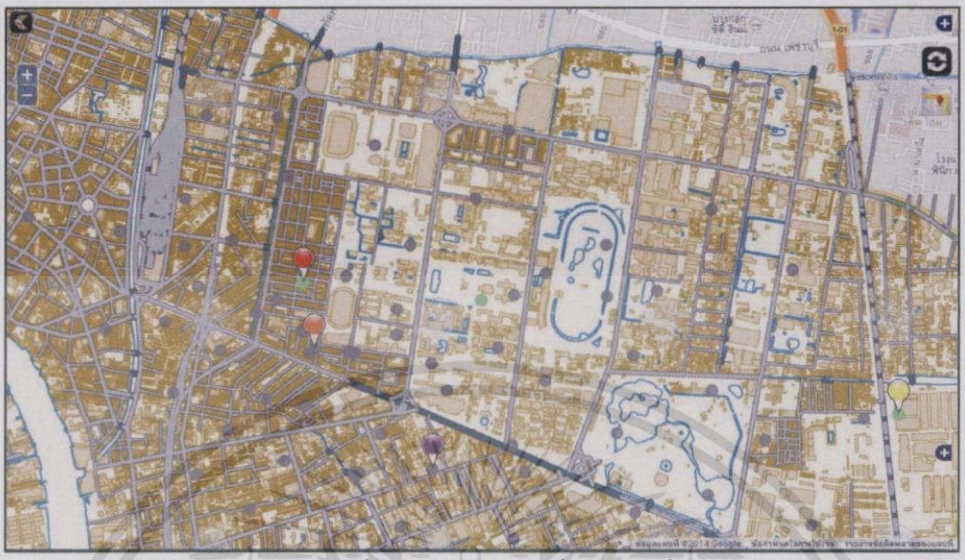
ประกอบด้วยการย่อ ขยาย และการเลื่อนแผนที่ ซึ่งแผนที่สามารถทำการย่อ ขยายได้ 15 ระดับ โดยมีเครื่องมือในการย่อ ขยายแผนที่บนหน้าเว็บและสามารถใช้เมาส์เพื่อการย่อ ขยายได้เช่นกันเพียงแค่วาง cursor ไว้บนแผนที่แล้วเลื่อน scroll ขึ้นหรือลงเพื่อ ย่อ ขยายแผนที่ หรือทาการ double click เพื่อการขยายแผนที่ 1 ระดับ



รูปที่ 5.1 แสดงเครื่องมือย่อ ขยายแผนที่

ในส่วนของการเลื่อนแผนที่สามารถทาการเลื่อนตำแหน่งการแสดงผลของภาพแผนที่ได้โดยการกดเมาส์ซ้ายค้างไว้บนหน้าแสดงแผนที่แล้วทาการเลื่อนเมาส์ไปมาตามตำแหน่งที่ต้องการ แผนที่จะเปลี่ยนขอบเขตการแสดงผลไปตามตำแหน่งที่เลื่อนเมาส์ไป

รูปที่ 5.2 แสดงการเลื่อนแผนที่



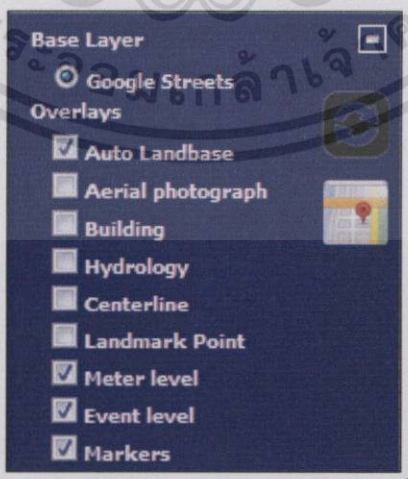
รูปที่ 5.3 แสดงการเลื่อนแผนที่ (ต่อ)

5.1.2 เปิด/ปิด ชั้นข้อมูลแผนที่

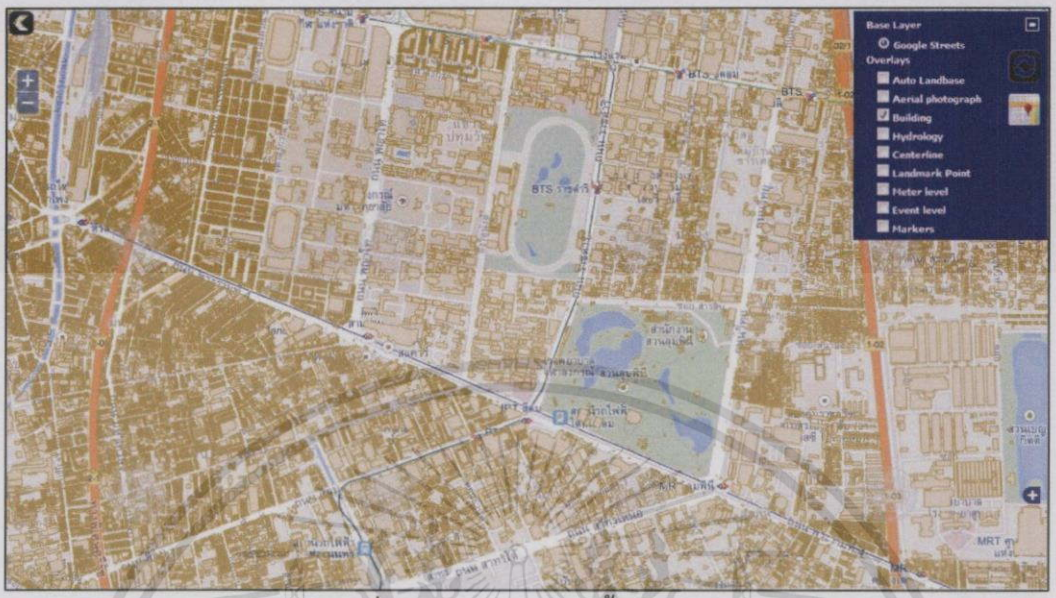
ในการเปิด/ปิด ชั้นข้อมูลแผนที่จะเครื่องมือในการจัดการแสดงผลเพื่อเปิดและปิดชั้นข้อมูลต่างๆ ประกอบด้วย 2 ส่วน

- Base Layers เป็นชั้นข้อมูลหลักสำหรับแสดงผลโดยใช้แผนที่ของ Google เป็นพื้นหลังเพื่อความสวยงามของแผนที่

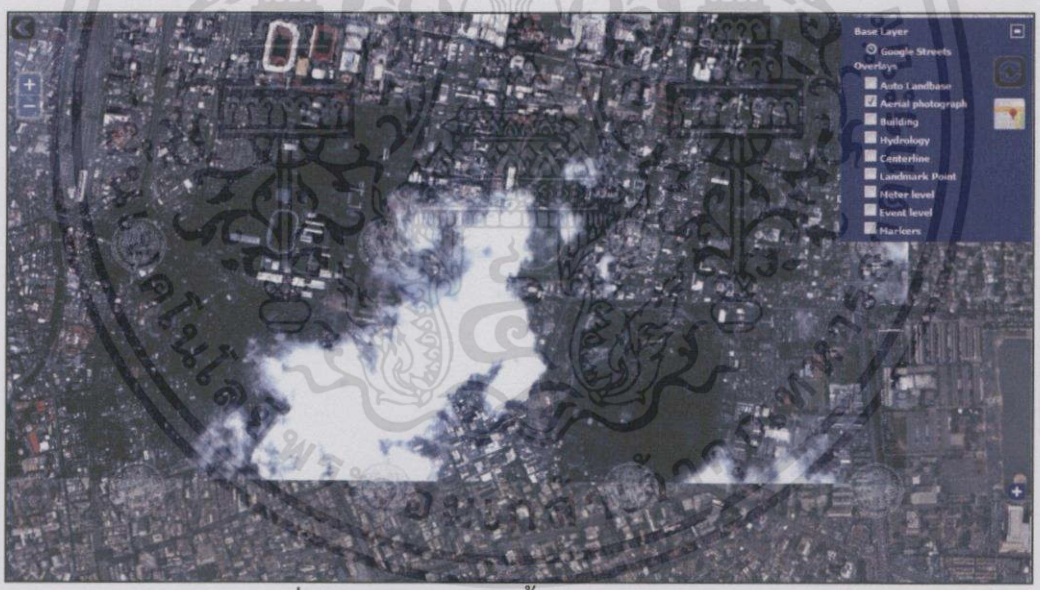
- Overlays จะเป็นส่วนเลือกการแสดงผลของชั้นข้อมูลที่ได้ทำการพัฒนาซึ่งประกอบด้วยชั้นข้อมูลต่างๆ สามารถเลือกเปิดปิดการแสดงผลโดยให้เลเยอร์ต่างๆ ซ้อนทับกันได้



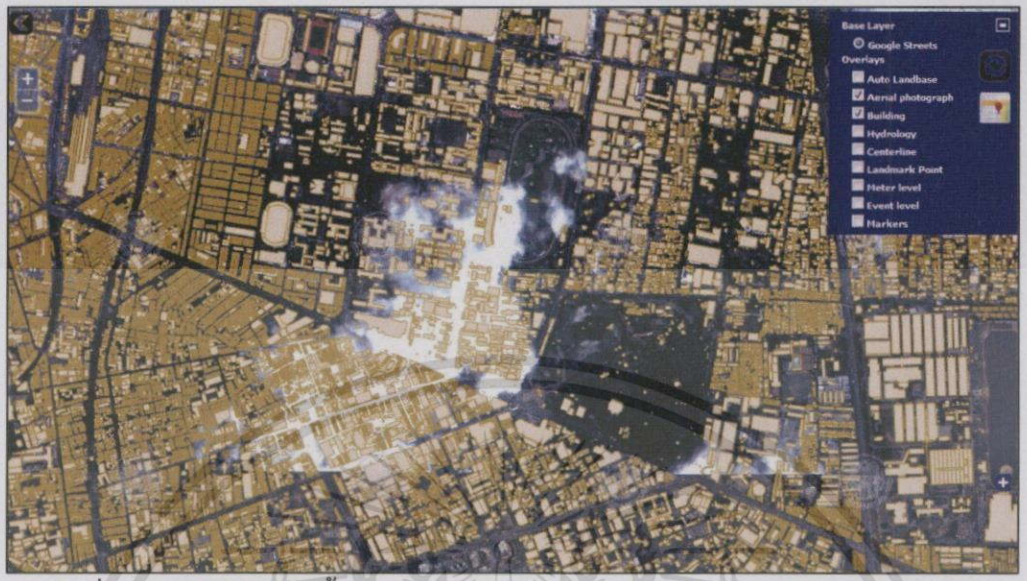
รูปที่ 5.4 แสดงเครื่องมือในการเปิด/ปิด ชั้นข้อมูลต่างๆ



รูปที่ 5.5 แสดงการเปิดชั้นข้อมูล Building



รูปที่ 5.6 แสดงการเปิดชั้นข้อมูล Aerial photograph

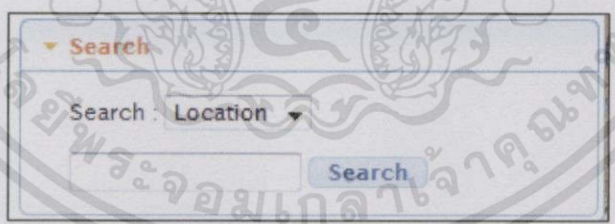


รูปที่ 5.7 แสดงการเปิดชั้นข้อมูล Aerial photograph และ Building พร้อมกัน

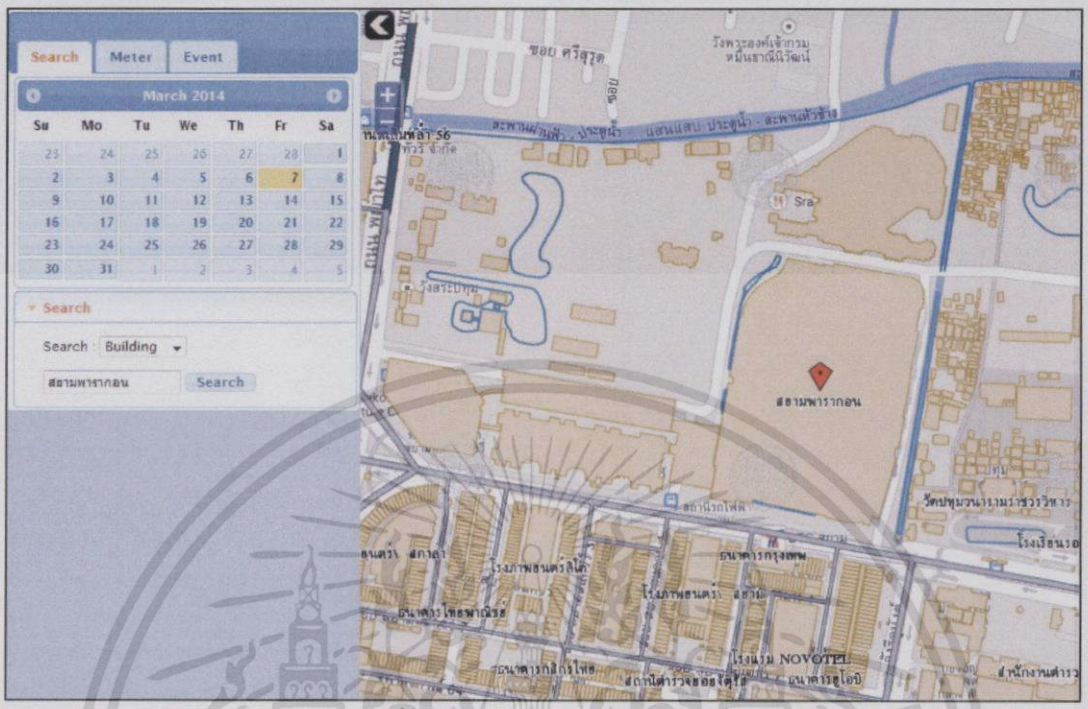
5.2 การค้นหาข้อมูลสถานที่และสมาร์ทมิเตอร์

5.2.1 การค้นหาข้อมูลสถานที่

การค้นหาข้อมูลสถานที่จะมีเครื่องมือให้ค้นหาโดยการเลือกชั้นข้อมูลที่ต้องการค้นหาและกรอกข้อมูลที่ต้องการค้นหาลงไป (ข้อมูลที่ต้องการค้นหาต้องมีเก็บอยู่ในฐานข้อมูล)



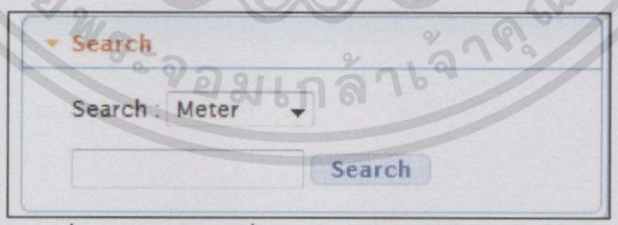
รูปที่ 5.8 แสดงเครื่องมือที่ใช้ในการค้นหาสถานที่



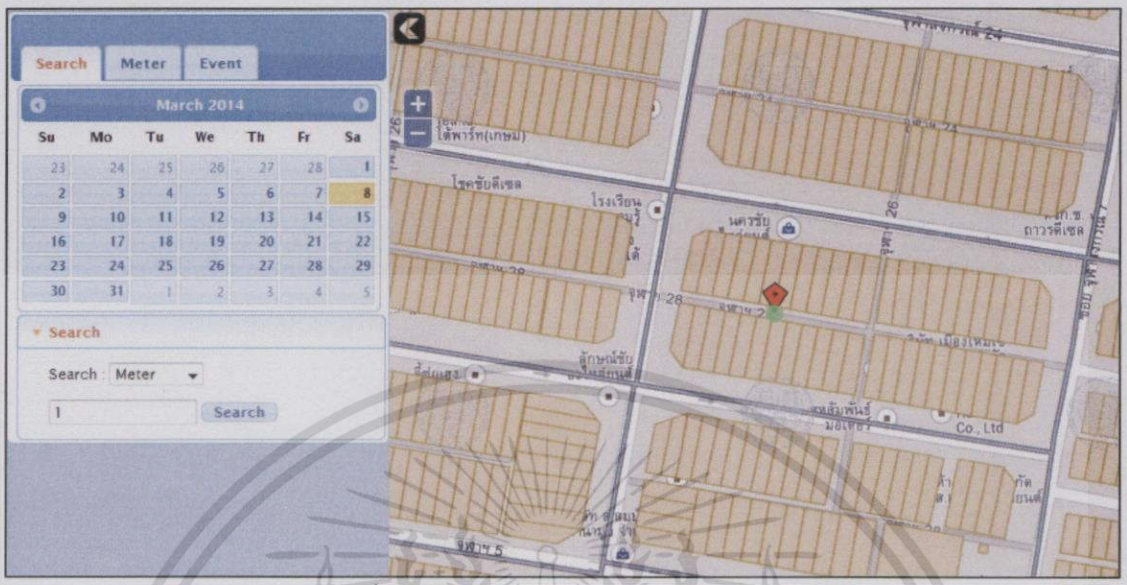
รูปที่ 5.9 แสดงการค้นหาสถานที่

5.2.2 การค้นหาสมาร์ทมิเตอร์

การค้นหาสมาร์ทมิเตอร์จะมีเครื่องมือให้ค้นหาโดยการเลือกชั้นข้อมูลไปที่ Meter และกรอกหมายเลขสมาร์ทมิเตอร์ที่ต้องการค้นหาลงไป (ข้อมูลหมายเลขสมาร์ทมิเตอร์ที่ต้องการค้นหาต้องมีเก็บอยู่ในฐานข้อมูล)



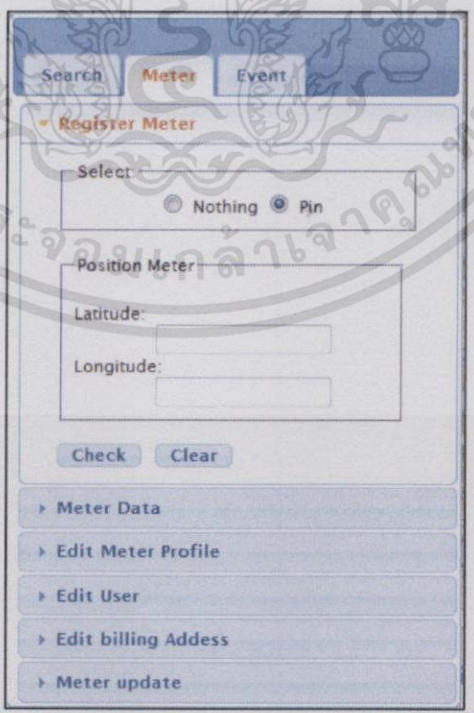
รูปที่ 5.10 แสดงเครื่องมือการค้นหาสมาร์ทมิเตอร์



รูปที่ 5.11 แสดงการค้นหาสมาร์ทมิเตอร์

5.3 ติดตั้งและลงทะเบียนสมาร์ทมิเตอร์

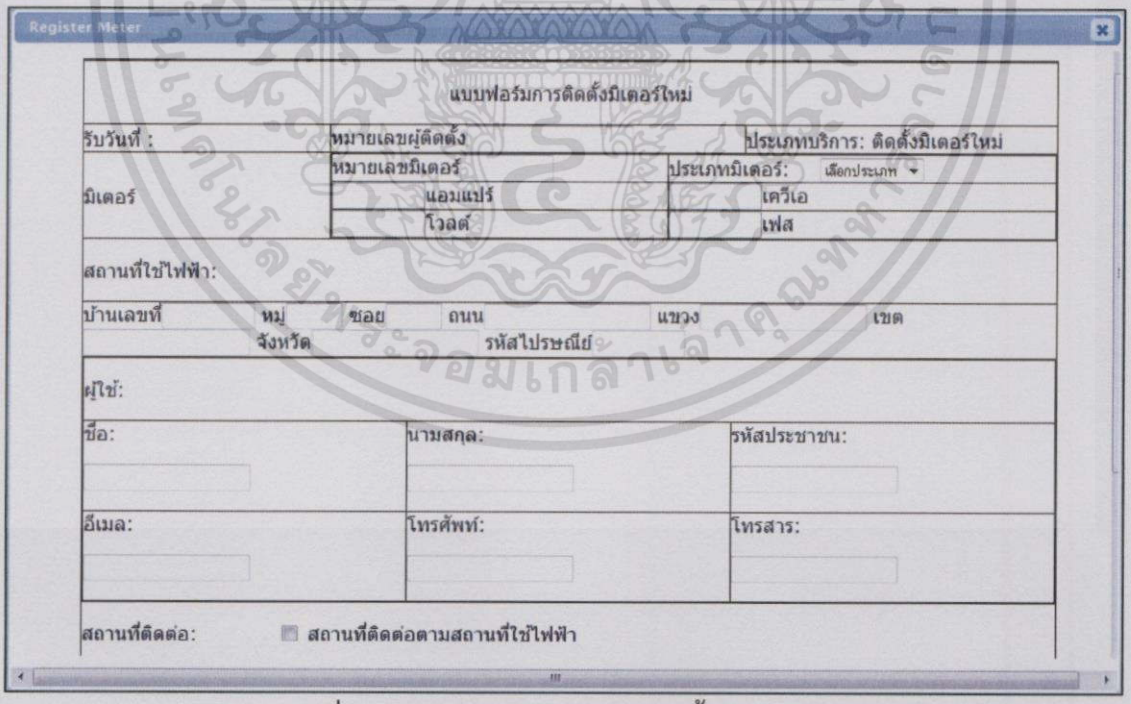
การเพิ่มสมาร์ทมิเตอร์ลงไปบนแผนที่สามารถทำได้โดยการเลือกเครื่องมือไปที่ Meter จากนั้นทำการเลือก Pin แล้วเลือกตำแหน่งบนแผนที่ที่ต้องการวางสมาร์ทมิเตอร์ลงไป



รูปที่ 5.12 แสดงเครื่องมือการเพิ่มสมาร์ทมิเตอร์

รูปที่ 5.13 แสดงตำแหน่งที่ต้องการวางสมาร์ทมิเตอร์

จากนั้นจะมีแบบฟอร์มการติดตั้งสมาร์ทมิเตอร์ใหม่ให้กรอกข้อมูล โดยสามารถกรอกข้อมูลลงในช่องแล้วกด Register Meter ก็จะปรากฏสมาร์ทมิเตอร์บนแผนที่



รูปที่ 5.14 แสดงแบบฟอร์มการติดตั้งสมาร์ทมิเตอร์ใหม่

รูปที่ 5.15 แสดงตำแหน่งของสมาร์ทมิเตอร์ที่วางลงไป

5.4 การขอข้อมูลปริมาณการใช้ไฟฟ้าและสถานะจากสมาร์ทมิเตอร์

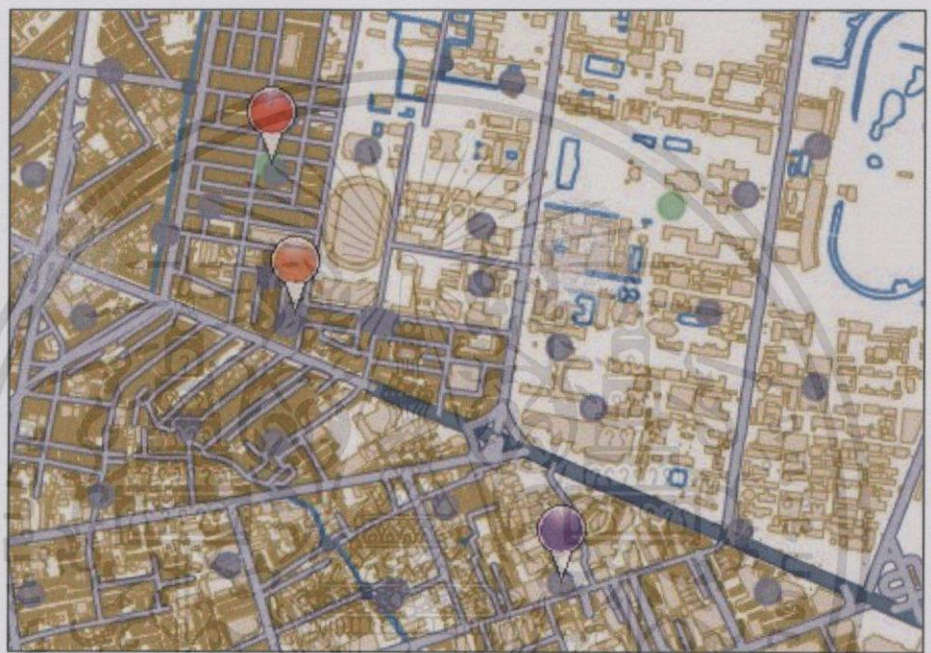
5.4.1 การขอข้อมูลปริมาณการใช้ไฟฟ้าจากสมาร์ทมิเตอร์

การขอข้อมูลการใช้ไฟฟ้าจากสมาร์ทมิเตอร์สามารถคลิกไปบนสมาร์ทมิเตอร์ที่ต้องการดูค่า จากนั้นจะมีป๊อบอัพแสดงข้อมูลขึ้นมา โดยข้อมูลที่นำมาแสดงมี Meter ID, ประเภทมิเตอร์, ผู้ใช้, การใช้ไฟฟ้า, การใช้ไฟฟ้าสูงสุดของวันและอัพเดทล่าสุด

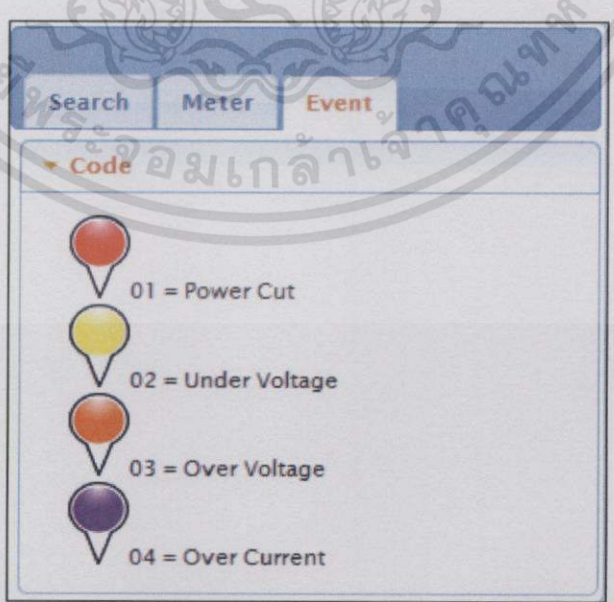
รูปที่ 5.16 แสดงผลการใช้ไฟฟ้าของสมาร์ทมิเตอร์

5.4.2 การแสดงผลสถานะของสมาร์ทมิเตอร์

การแสดงผลสถานะของสมาร์ตมิเตอร์สามารถดูได้จากบนแผนที่ซึ่งถ้าสมาร์ทมิเตอร์มีสถานะเกิดขึ้นจะแสดงสัญลักษณ์บนสมาร์ทมิเตอร์ตัวนั้น โดยสามารดูสัญลักษณ์ของแต่ละเหตุการณ์ได้ที่แถบเครื่องมือ event



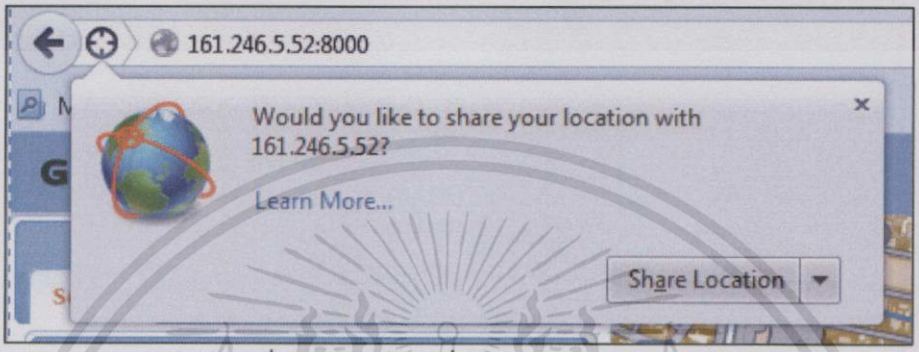
รูปที่ 5.17 แสดงสถานะของสมาร์ทมิเตอร์ที่เกิดขึ้น



รูปที่ 5.18 แสดงสัญลักษณ์ของสถานะและเหตุการณ์ที่เกิดขึ้น

5.5 การแสดงตำแหน่งปัจจุบัน

ระบบสามารถค้นหาตำแหน่งที่เราอยู่ปัจจุบันเพื่อช่วยในการติดตั้งสมาร์ทมิเตอร์ โดยเลือกไปที่ ⊕ จากนั้นกด Share Location แผนที่ก็จะแสดงตำแหน่งที่เราอยู่ปัจจุบัน



รูปที่ 5.19 แสดงเครื่องมือการหาตำแหน่งปัจจุบัน

รูปที่ 5.20 แสดงตำแหน่งปัจจุบัน

บทที่ 6
บทสรุป

6.1 บทสรุป

ระบบภูมิสารสนเทศเพื่อการบริหารจัดการสมาร์ทมิเตอร์เป็นเว็บแอพพลิเคชั่นที่ใช้แสดงข้อมูลและจัดการสมาร์ทมิเตอร์โดยใช้โอเพ่นซอร์สทั้งหมด โดยตัวเว็บไซต์สามารถดูข้อมูลบนตัวสมาร์ทมิเตอร์, เพิ่มสมาร์ทมิเตอร์ลงบนแผนที่, ค้นหาสมาร์ทมิเตอร์และสถานที่ และค้นหาตำแหน่งปัจจุบันได้ ซึ่งจะช่วยในการบริหารจัดการสมาร์ทมิเตอร์ให้มีประสิทธิภาพเพื่อการใช้ทรัพยากรที่มีอยู่ให้เกิดประโยชน์

ระบบภูมิสารสนเทศเพื่อการบริหารจัดการสมาร์ทมิเตอร์ที่ได้พัฒนาขึ้นมีข้อดีคือสามารถนำใช้งานในการบริหารจัดการสมาร์ทมิเตอร์ได้ แต่มีข้อเสียคือเมื่อมีการใช้มิเตอร์จำนวนมากระบบอาจจะต้องมีการรองรับข้อมูลจำนวนมากที่ได้จากสมาร์ทมิเตอร์ โดยถ้ามีการนำไปพัฒนาต่อในด้านต่างๆ เช่น ด้านของแผนที่เพื่อขยายพื้นที่ของแผนที่ที่มีอยู่อย่างจำกัด ในด้านของการจัดเก็บข้อมูลเพื่อรองรับข้อมูลจำนวนมากที่ได้จากสมาร์ทมิเตอร์ ในด้านของการเชื่อมต่อกับสมาร์ทมิเตอร์หลายๆตัว ก็จะทำให้ระบบมีความสมบูรณ์มากยิ่งขึ้น

6.2 สิ่งที่ได้รับ

1) ได้รับความรู้ความเข้าใจถึงโครงสร้างและการทำงานระบบภูมิสารสนเทศ
2) สามารถสร้างเว็บแอพพลิเคชั่น โดยใช้เทคโนโลยี HTML5 เข้ามาประยุกต์
3) ได้รับความรู้เกี่ยวกับการทำเว็บแอพพลิเคชั่นรวมถึงเครื่องมือต่างๆ
4) ได้รับความรู้เกี่ยวกับการบริหารจัดการสมาร์ทมิเตอร์, การติดตั้งและการตรวจสอบ
5) มีความเข้าใจถึงปัญหาและความสำคัญของการพัฒนาระบบพลังงานไฟฟ้าในอนาคต
6) เข้าใจถึงกระบวนการพัฒนาระบบ ทั้งการวางแผน การวิเคราะห์ การออกแบบ การพัฒนาโปรแกรม

6.3 ปัญหาอุปสรรคและการแก้ไข

1) ลง library ไม่ครบทำให้บางโปรแกรมไม่สามารถแสดงผลลัพธ์ออกมาได้ จึงทำการติดตั้ง library ตามเอกสารที่แนะนำ
2) ไฟล์ map ไม่ทำงานทำให้ต้องลง mapserver ใหม่
3) มีปัญหา cache ของเว็บเบราว์เซอร์ทำให้ค้างค่าเก่าอยู่ จึงไปทำการลบ cache ที่เว็บเบราว์เซอร์

6.3 ปัญหาอุปสรรคและการแก้ไข

1) ลง library ไม่ครบทำให้บางโปรแกรมไม่สามารถแสดงผลลัพธ์ออกมาได้ จึงทำการติดตั้ง library ตามเอกสารที่แนะนำ
2) ไฟล์ map ไม่ทำงานทำให้ต้องลง mapserver ใหม่
3) มีปัญหา cache ของเว็บเบราว์เซอร์ทำให้ค้างค่าเก่าอยู่ จึงไปทำการลบ cache ที่เว็บเบราว์เซอร์

6.4 แนวทางในการพัฒนาต่อ

1) เพิ่มประสิทธิภาพการดึงข้อมูลจากเว็บแมพเซอร์วิส
2) พัฒนาแอพพลิเคชั่นแสดงบนสมาร์ทโฟน
3) พัฒนาเรื่องความปลอดภัยของระบบ
4) พัฒนาเรื่องการรับค่าและจัดเก็บข้อมูลจำนวนมาก

บรรณานุกรม

[1] Erik Hazzard. **OpenLayers 2.10. Birmingham : Packt Publishing Ltd, 2011.**

[2] Regina O. Obe. Leo S. Hsu. **PostGIS in Ation**. Stamford : Manning Publication Co, 2011.

[3] สุริยา เกิดสำราญ. อนันต์ ธารีวิบูลย์. **"ระบบแผนที่จีไอเอสบนคลาวด์"** ปริญญานิพนธ์วิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง. 2555.

[4] โครงสร้างข้อมูลแรสเตอร์และเวคเตอร์. **"บทที่ 1 แผนที่ตัวเลข (Digital Map)."**[Online]. Available : http://resgat.net/digital/digimap.html, 2013.

[5] ระบบสารสนเทศภูมิศาสตร์. **"Learnig GIS."**[Online]. Available : http://www.gisthai.org/about-gis/gis.html, 2013.

[6] Django. **"Django documentation."**[Online]. Available : https://docs.djangoproject.com/en/1.5/, 2013.

[7] MapServers. **"MapServer 6.4.0 Documentation."**[Online]. Available : http://mapserver.org/documentation.html, 2013.

[8] PostGIS. **"PostGIS Document."**[Online]. Available : http://postgis.net/documentation, 2013.

[9] PostgreSQL. **"Documentation."**[Online]. Available : http://www.postgresql.org/docs, 2013.

[10] SMART CITY. **"SMART GRID, SMART CITY สู่เมืองอัจฉริยะแห่งอนาคต."**[Online]. Available : http://www.measwatch.org/writing/3205, 2013.

[11] OpenLayers. **"Documentation."**[Online]. Available : http://trac.osgeo.org/openlayers/wiki/Documentation, 2013.

[12] W3schools. **"HTML5."**[Online]. Available : http://www.w3schools.com/html/html5_intro.asp, 2013.

[13] W3schools. **"JavaScript."**[Online]. Available : http://www.w3schools.com/js/default.asp, 2013.