รายงานฉบับสมบูรณ์ (ปีที่ 1)

การพัฒนาระบบสำรวจและจัดการข้อมูลปริมาณรังสีแกมมาแบบติดรถยนต์สำหรับการเตรียมพร้อมและตอบสนองต่ออุบัติการณ์ด้านความมั่นคงปลอดภัยทางนิวเคลียร์

Development of Car-Borne Gamma Survey and Data Management Systems for Preparedness and Response to Nuclear Security Incident

โดย

พงษ์แพทย์ เพ่งวาณิชย์

สุพิชชา จันทรโยธา

ชุติมา กรานรอด

งบประมาณสนับสนุนจากงบประมาณแผ่นดินปี 2559

คณะวิศวกรรมศาสตร์

จุฬาลงกรณ์มหาวิทยาลัย

กิตติกรรมประกาศ

คณะผู้วิจัยมีความทราบซึ้งและขอขอบคุณการสนับสนุนจากบุคคลและภาคส่วนต่างๆ ที่ช่วยให้การวิจัยสำเร็จลุล่วงด้วยดี ทั้งด้านงบประมาณจากสำนักงานคณะกรรมการวิจัยแห่งชาติภายใต้งบประมาณแผ่นดินปี 2559 ด้านเครื่องมือและการกระบวนการสำรวจจาก Prof. Dr. Shinji Sugihara และ Dr. Masahiro Hosoda จาก Institute of Radiation Emergency medicine, Hirosaki University, Japan ด้านอิเล็กทรอนิกส์จาก อ. เดโช ทองอร่าม และด้านสถานที่จากภาควิชาวิศวกรรมศาสตร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

คำนำ

รายงานฉบับนี้ เป็นผลการดำเนินการในปีที่ 1 ของงานวิจัยเรื่อง “การพัฒนาระบบสำรวจและจัดการข้อมูลปริมาณรังสีแกมมาแบบติดรถยนต์สำหรับการเตรียมพร้อมและตอบสนองต่ออุบัติการณ์ด้านความมั่นคงปลอดภัยทางนิวเคลียร์” ซึ่งเป็นโครงการวิจัยต่อเนื่องที่มีระยะเวลาดำเนินการ 2 ปี (ตามแผน) ได้รับงบประมาณสนับสนุนจากงบประมาณแผ่นดินปี 2559 โดยในปีที่ 1 เป็นการพัฒนาระบบ และปีที่ 2 จะเป็นการทดสอบระบบในสภาพแวดล้อมจริง แบ่งเนื้อหาของรายงานออกเป็น 4 บท และมีรายละเอียดเพิ่มเติมของผลงานในภาคผนวก

บทคัดย่อ

ปริมาณวัสดุนิวเคลียร์และสารกัมมันตรังสีทั้งในประเทศไทยและต่างประเทศมีเพิ่มสูงขึ้นเรื่อยๆ จากการนำเทคโนโลยีที่เกี่ยวข้องมาใช้ประโยชน์ในด้านต่างๆ ในปัจจุบัน ทำให้ความเสี่ยงที่จะมีการสูญหายของวัสดุนิวเคลียร์และสารกัมมันตรังสีหรือมีผู้นำวัสดุดังกล่าวไปใช้ในทางมิชอบเพิ่มสูงขึ้นด้วย ประกอบกับสถานการณ์ความไม่มั่นคงที่เกิดขึ้นทั่วโลก ทำให้ประเทศต่างๆ ให้ความสำคัญกับความมั่นคงปลอดภัยทางนิวเคลียร์และรังสีเป็นอย่างมาก ซึ่งสำหรับโครงการวิจัยนี้ มีวัตถุประสงค์ในการช่วยเพิ่มศักยภาพในการตรวจหาวัสดุกัมมันตรังสีที่อยู่นอกเหนือการควบคุม (เช่น การสูญหาย การลักลอบขนส่ง หรือมีผู้ไม่หวังดีนำไปวางไว้ในที่สาธารณะ) ซึ่งอาจทำให้สาธารณชนที่อยู่ใกล้เคียงได้รับรังสีโดยไม่ตั้งใจและเป็นอันตรายได้ จึงได้ทำการพัฒนาระบบสำรวจและจัดการข้อมูลปริมาณรังสีแบบติดรถยนต์ขึ้น เพื่อให้สามารถตรวจหาบริเวณที่มีรังสีสูงในบริเวณกว้างได้อย่างรวดเร็ว โดยระบบดังกล่าวประกอบด้วยระบบสำรวจรังสีแกมมาและระบบจัดการข้อมูลอัตโนมัติที่เชื่อมต่อกับฐานข้อมูลซึ่งสามารถค้นหาและเรียกใช้ข้อมูลผ่านระบบอินเตอร์เน็ตเพื่อการวิเคราะห์ภายหลังได้ จึงเหมาะสำหรับประยุกต์ใช้ในด้านการเตรียมการและตอบสนองต่อเหตุการณ์ฉุกเฉินทางนิวเคลียร์และรังสี และสำหรับโปรแกรมควบคุม ได้พัฒนาขึ้นด้วยภาษา Java ซึ่งสามารถใช้งานได้บนระบบปฏิบัติการที่หลากหลาย และใช้มาตรฐานการเชื่อมต่อข้อมูลแบบ RS-232 ซึ่งรองรับอุปกรณ์สำรวจทั่วไปได้ในปัจจุบันได้

The quantity of nuclear and radioactive materials in Thailand and other countries is continuously increasing with today's widespread utilization of related technology. As a result, the risk of losing or misuse of nuclear and radioactive materials also increases. Adding this to the fact that many unstable situations is happening around the world, many countries are stressing the importance of nuclear and radiation security issue. This research project aims to help increase the capability of detection of radioactive materials outside regulatory control (e.g. lost, illegal trafficking, public exposure) that can cause unintentional exposure and danger to the nearby public. Thus, the car-borne radiation survey and data management system has been developed in order to be able to quickly search and identify high radiation level in a wide area. The system consists of a gamma survey system and an automatic data management system that connects to a database which can be queried via internet for later analysis. It is therefore suitable for application in nuclear and radiation emergency preparedness and response. In addition, the control program has been developed in Java programming language which can be used on variety of operating systems, and utilizes the RS-232 standard data connection that is compatible with many survey devices today.

สารบัญ

สารบัญตาราง

สารบัญรูป

บทที่ 1

บทนำ

1.1 ความสำคัญและที่มาของปัญหา

จากการเจริญเติบโตทางเศรษฐกิจและสังคมในภูมิภาค ประกอบกับเสรีในการเดินทางและติดต่อค้าขายระหว่างประเทศ ส่งผลให้ปริมาณและการใช้วัสดุนิวเคลียร์และสารกัมมันตรังสีทั้งในประเทศไทยและประเทศเพื่อนบ้านมีแนวโน้มในทิศทางที่เพิ่มสูงขึ้นเรื่อยๆ ซึ่งนั่นหมายความว่า ความเสี่ยงที่จะมีผู้นำวัสดุนิวเคลียร์และสารกัมมันตรังสีไปใช้ในทางมิชอบก็จะมีเพิ่มขึ้น เพราะแม้ว่าคนส่วนใหญ่จะมุ่งเน้นที่จะนำวัสดุนิวเคลียร์และสารกัมมันตรังสีไปใช้ในทางสันติเพื่อการพัฒนาในด้านต่างๆ (เช่น อุตสาหกรรม การแพทย์ การเกษตร และพลังงาน) แต่จากภาวะสงครามและการก่อการร้าย ที่หลายๆ ประเทศรวมทั้งประเทศไทยใน 3 จังหวัดภาคใต้กำลังเผชิญอยู่ คงไม่สามารถปฏิเสธความเป็นไปได้ที่อาจจะมีผู้ไม่หวังดีบางคนหรือบางกลุ่มนำวัสดุนิวเคลียร์และสารกัมมันตรังสีไปใช้ในทางไม่สันติ

เหตุการณ์การก่อวินาศกรรมในประเทศสหรัฐอเมริกาเมื่อวันที่ 11 กรกฎาคม 2001 ชี้ให้เห็นว่าเทคโนโลยีที่สร้างขึ้นสำหรับพลเรือนหากตกไปอยู่ในมือของผู้ก่อการร้ายก็อาจนำไปสู่หายนะที่ร้ายแรงเทียบเท่าหรือมากกว่าการต่อสู้แบบเผชิญหน้าได้ และจากเหตุการณ์ในครั้งนั้นรวมถึงความพยายามในการก่อการร้ายครั้งต่อๆ มา (เช่น จดหมายแอนแทรกซ์ การใช้สารเคมีในการสู้รบ) จึงทำให้เกิดนโยบายในการตอบโต้การก่อการร้ายทางเคมี (Chemical) ชีวะ (Biological) นิวเคลียร์ (Nuclear) และรังสี (Radiological) หรือ CBRN Counterterrorism ขึ้น ซึ่งเป็นหนึ่งในนโยบายที่ประเทศมหาอำนาจต่างๆ รวมถึงสหภาพยุโรปให้ความสำคัญเป็นอย่างมาก

สาเหตุที่นิวเคลียร์และรังสีถูกรวมอยู่ในนโยบายดังกล่าว มิใช่เพียงเพราะภัยร้ายจากระเบิดปรมาณูเท่านั้น แต่เป็นเพราะความเป็นไปได้ที่จะผสมวัสดุนิวเคลียร์หรือสารกัมมันตรังสีเข้ากับระเบิดทั่วๆ ไป และสร้างสิ่งที่เรียกว่า Dirty Bomb ขึ้น ซึ่งเมื่อเกิดการระเบิด นอกจากจะมีผลกระทบจากแรงระเบิดโดยปกติแล้ว ยังทำให้เกิดการกระจายตัวของสารกัมมันตรังสีไปในบริเวณกว้างได้ หลายประเทศเริ่มตื่นตัวกับเรื่องนี้ เห็นได้จากการประชุมสุดยอดด้านความมั่นคงปลอดภัยทางนิวเคลียร์ (Nuclear Security Summit) ครั้งล่าสุดเมื่อปี 2012 ซึ่งมีผู้นำจาก 53 ประเทศรวมทั้ง ฯพณฯ นายกรัฐมนตรียิ่งลักษณ์ ชินวัตร เข้าร่วม และได้กล่าวถ้อยแถลงกับที่ประชุมว่าจะให้การสนับสนุนการเสริมสร้างความมั่นคงปลอดภัยทางนิวเคลียร์ในภูมิภาค

การเพิ่มขึ้นของวัสดุนิวเคลียร์และสารกัมมันตรังสีทั้งในเชิงปริมาณและชนิดที่นำมาใช้ และในเชิงความหลากหลายของเทคโนโลยีและสถานที่ใช้งาน ทำให้โอกาสที่ผู้ไม่หวังดีจะได้วัสดุนิวเคลียร์หรือสารกัมมันตรังสีมาครอบครองมีมากขึ้น ไม่ว่าวัสดุนิวเคลียร์หรือสารกัมมันตรังสีดังกล่าวจะมาจากในประเทศ ผ่านจุดข้ามแดนเข้ามาหรือออกไปนอกประเทศ หรือเพียงมีการขนส่งผ่านประเทศโดยมิได้รับอนุญาต ก็นับเป็นภัยคุกคามที่ร้ายแรงต่อชีวิตและทรัพย์สินของประชาชนทั้งสิ้น ดังนั้น การตรวจหาและติดตามวัสดุนิวเคลียร์และสารกัมมันตรังสีที่มีผู้ลักลอบขนส่งหรือมีไว้ในครอบครองจึงเป็นหนึ่งในกระบวนการสำคัญที่จะช่วยเสริมสร้างความมั่นคงปลอดภัยทางนิวเคลียร์ของประเทศและของภูมิภาคได้ แต่เนื่องจากวัสดุนิวเคลียร์และสารกัมมันตรังสีดังกล่าวไม่สามารถตรวจวัดได้โดยง่ายและอาจอยู่ในที่ใดก็ได้ การตรวจหาและติดตามจึงต้องสามารถทำได้ในบริเวณกว้าง และ รวดเร็ว จึงเป็นที่มาของงานวิจัยที่ ที่ต้องการพัฒนาระบบสำหรับสำรวจและจัดการข้อมูลปริมาณรังสีแกมมาแบบติดรถยนต์ที่สามารถตรวจหารังสีแกมมาที่ปล่อยออกมาจากวัสดุนิวเคลียร์หรือสาร

กัมมันตรังสีโดยไม่ต้องหยุดสำรวจเป็นจุดๆ ไป ซึ่งจะทำให้สามารถสำรวจบริเวณกว้างได้อย่างรวดเร็ว โดยจะทำการพัฒนาในสองส่วน คือ ระบบสำรวจปริมาณรังสีแกมมาแบบติดรถยนต์ และ ระบบจัดเก็บและแสดงข้อมูลปริมาณรังสีแกมมา

งานวิจัยนี้ นอกจากจะมีประโยชน์ในด้านความมั่นคงปลอดภัยทางนิวเคลียร์แล้ว ยังสามารถประยุกต์ใช้ในด้านการเตรียมการและตอบสนองต่อเหตุการณ์ฉุกเฉินทางนิวเคลียร์และรังสีด้วย ซึ่งอาจนำไปใช้ในการจัดทำข้อมูลพื้นฐานปริมาณรังสีแกมมาที่ประชากรได้รับในบริเวณต่างๆ ของประเทศไทย หรือใช้ในการสำรวจปริมาณรังสีแกมมาเมื่อเกิดเหตุการณ์ฉุกเฉินทางนิวเคลียร์และรังสีเพื่อใช้ประกอบการตัดสินใจเลือกใช้มาตรการตอบสนองได้อย่างเหมาะสม โดยการประยุกต์ใช้ทั้งสองด้าน จะต้องอาศัยระบบตรวจวัดรังสีที่สามารถใช้งานในบริเวณกว้างในเวลาที่รวดเร็วได้ ซึ่งในปัจจุบันยังไม่มีระบบดังกล่าวในประเทศไทย

1.2 วัตถุประสงค์ของโครงการวิจัย

เพื่อพัฒนาระบบสำรวจและจัดเก็บข้อมูลปริมาณรังสีแกมมาแบบติดรถยนต์ที่สามารถใช้ตรวจหาวัสดุนิวเคลียร์และสารกัมมันตรังสีในบริเวณกว้างได้อย่างรวดเร็วเมื่อเกิดอุบัติการณ์ด้านความมั่นคงปลอดภัยทางนิวเคลียร์

1.3 ขอบเขตของโครงการวิจัย

- ออกแบบและพัฒนาระบบสำรวจปริมาณรังสีแกมมาแบบติดรถยนต์ที่สามารถใช้ตรวจหาวัสดุนิวเคลียร์และสารกัมมันตรังสีในบริเวณกว้างได้อย่างรวดเร็วเมื่อเกิดอุบัติการณ์ด้านความมั่นคงปลอดภัยทางนิวเคลียร์
- ออกแบบและพัฒนาระบบจัดเก็บและแสดงข้อมูลปริมาณรังสีแกมมาที่สามารถเรียกใช้งานได้เมื่อมีสัญญาณอินเตอร์เน็ต

1.4 ประโยชน์ที่คาดว่าจะได้รับ

- สามารถเผยแพร่เทคนิคการตรวจหาสามารถนำระบบที่พัฒนาขึ้นไปใช้ตรวจหาและติดตามวัสดุนิวเคลียร์และสารกัมมันตรังสีที่ถูกลักลอบขนส่งหรือโจรกรรม (Orphan Source) ในวารสารได้
- สามารถนำระบบที่พัฒนาขึ้นไปใช้ตรวจหาและติดตามวัสดุนิวเคลียร์และสารกัมมันตรังสีที่ถูกลักลอบขนส่งหรือโจรกรรม (Orphan Source)
- สามารถนำระบบที่พัฒนาขึ้นไปใช้รวบรวมข้อมูลพื้นฐานปริมาณรังสีทั่วประเทศ

1.5 แผนการถ่ายทอดเทคโนโลยีหรือผลการวิจัยสู่กลุ่มเป้าหมาย

- นำเสนอระบบสำรวจให้กับหน่วยงานด้านความมั่นคงของประเทศ
- ระบบที่พัฒนาขึ้นสามารถนำไปใช้สำรวจข้อมูลพื้นฐานปริมาณรังสีของประเทศ ซึ่งเป็นข้อมูลสำคัญที่จะต้องใช้ในการเตรียมพร้อมและตอบสนองต่อเหตุการณ์ฉุกเฉินทางรังสี สำหรับประเทศไทย

1.6 แผนการดำเนินงานตลอดโครงการวิจัย

ระยะเวลาทำการวิจัย คือ 2 ปี มีแผนการดำเนินงานดังนี้

กิจกรรม	ปีที่ 1 (2559)				ปีที่ 2 (2560)			
	1-3	4-6	7-9	10-12	1-3	4-6	7-9	10-12
1. ออกแบบและพัฒนาระบบวัดปริมาณรังสีแกมมาแบบติดรถยนต์	←	→						
2. ออกแบบและพัฒนาระบบจัดการข้อมูลปริมาณรังสีแกมมา	←	→						
3. พัฒนาโปรแกรมเชื่อมต่ออัตโนมัติระหว่างระบบสำรวจและระบบจัดการข้อมูล		←	→					
4. ศึกษาปัจจัยที่มีผลต่อการวัด และวิธีการแก้ค่าต่างๆ			←	→				
5. ทดลองวัดจริงในสิ่งแวดล้อม				←	→			
6. วิเคราะห์ เปรียบเทียบ แก้ค่า และปรับแก้ระบบและวิธีการ				←	→			
7. ทดลองวัดกับ Orphan Source ในสิ่งแวดล้อม						←	→	
8. ปรับแก้วิธีการตรวจหา Orphan Source ในสิ่งแวดล้อม						←	→	
9. สรุปผลการศึกษาและจัดทำรายงาน								↔

1.7 ผลสำเร็จและความคุ้มค่าของการวิจัยที่คาดว่าจะได้รับ

ผลผลิตในแต่ละปีมีดังนี้

<u>ปีที่ 1</u>

- ระบบวัดปริมาณรังสีแกมมาแบบติดรถยนต์
- ระบบจัดการข้อมูลปริมาณรังสีแกมมา
- โปรแกรมเชื่อมต่ออัตโนมัติระหว่างระบบสำรวจและระบบจัดการข้อมูล

<u>ปีที่ 2</u>

- ระบบสำรวจและจัดการข้อมูลปริมาณรังสีแกมมาแบบติดรถยนต์ที่สามารถนำไปตรวจหา Orphan Source ได้

บทที่ 2

ทฤษฎีและวรรณกรรมที่เกี่ยวข้อง

2.1 กรอบแนวคิดของโครงการวิจัย

ระบบวัดรังสีแกมมาที่ใช้อยู่ในปัจจุบัน ได้ผ่านการพัฒนามาเป็นระยะเวลายาวนาน จนมีมาตรฐานและความถูกต้องแม่นยำที่ค่อนข้างสูง ได้รับการยอมรับในวงการนิวเคลียร์เป็นอย่างดี อย่างไรก็ตาม ระบบวัดส่วนใหญ่ได้รับการออกแบบให้วัดอยู่กับที่ ซึ่งอาจไม่เหมาะกับการสำรวจในวงกว้างที่บางครั้งต้องมีการแข่งขันกับเวลา เช่น ในการตรวจหาและติดตามวัสดุนิวเคลียร์และสารกัมมันตรังสีที่มีการลักลอบขนส่งโดยมิได้รับอนุญาตหรือถูกโจรกรรม (Orphan Source) เมื่อเกิดเหตุการณ์ฉุกเฉินทางนิวเคลียร์และรังสีอันนำไปสู่การกระจายตัวของสารกัมมันตรังสีในวงกว้างอย่างรวดเร็วและมีความจำเป็นที่จะต้องเลือกใช้มาตรการตอบสนองฉุกเฉินอย่างเหมาะสม หรือในสถานการณ์ปกติที่ต้องการประเมินปริมาณรังสีที่ประชากรได้รับในแต่ละพื้นที่ของประเทศ

ด้วยเหตุนี้ ในการสำรวจปริมาณรังสีแกมมาในวงกว้าง จึงมักติดตั้งระบบวัดเข้ากับยานพาหนะต่างๆ เช่น เครื่องบิน รถยนต์ และเรือ เพื่อให้สามารถครอบคลุมพื้นที่ในบริเวณกว้าง และเข้าถึงพื้นที่ที่อาจยากต่อการเข้าถึงได้อย่าง อย่างไรก็ตาม การวัดแบบเคลื่อนที่จะต้องคำนึงถึงปัจจัยต่างๆ หลายปัจจัยที่อาจมีผลต่อค่าที่วัดได้ ซึ่งรวมถึง

- หัววัดที่เลือกใช้
- ตำแหน่งและทิศทางของหัววัด
- เวลาของการวัด
- ช่วงการเก็บข้อมูล
- ยานพาหนะที่ใช้
- ความเร็วของการเคลื่อนที่
- ลักษณะภูมิประเทศในบริเวณโดยรอบ
- การกำบังรังสีของยานพาหนะ

ปัจจัยต่างๆ เหล่านี้ จะถูกนำไปผนวกเข้ากับการออกแบบและพัฒนาระบบวัด รวมถึงการวิเคราะห์ผลการวัด เพื่อให้สามารถนำผลที่ได้ไปใช้ได้อย่างเหมาะสม

สำหรับระบบสำรวจและติดตามวัสดุนิวเคลียร์และสารกัมมันตรังสี ปัญหาที่สำคัญอีกปัญหาหนึ่งคือการขาดข้อมูลที่ชัดเจนเกี่ยวกับชนิด ปริมาณ และตำแหน่งของวัสดุนิวเคลียร์หรือสารกัมมันตรังสีที่ต้องการตรวจหา ซึ่งอาจทำให้บางครั้งไม่สามารถแยกแยะสัญญาณที่เกิดจากวัสดุนิวเคลียร์หรือสารกัมมันตรังสีที่ต้องการหาออกจากรังสีพื้นหลัง (Background Radiation) ได้ ดังนั้น เทคนิคการวัดและประเมิณผลจึงเป็นสิ่งสำคัญและจะทำการศึกษาในงานวิจัยนี้ด้วย

2.2 วรรณกรรมที่เกี่ยวข้อง

การใช้ระบบวัดรังสีแกมมาแบบติดรถยนต์ได้รับการพัฒนาอย่างต่อเนื่อง และได้ถูกนำไปใช้ในการสำรวจปริมาณรังสีภาคพื้นดินเพื่อทำแผนที่รังสีในประเทศต่างๆ หลายประเทศ โดยเห็นได้จากผลงานวิจัยหลายชิ้นที่มีการตีพิมพ์ออกมา เช่น

ในปี 2013 M. Tanigaki และคณะได้ตีพิมพ์ผลการพัฒนาและทดสอบระบบ KURAMA (Kyoto University Radiation Mapping system) ซึ่งเป็นระบบสำรวจรังสีแกมมาแบบติดรถยนต์ที่พัฒนาขึ้นเพื่อตอบสนองต่ออุบัติเหตุทางนิวเคลียร์และรังสีที่เกิดขึ้น ณ โรงไฟฟ้าฟุกิชิมะ ประเทศญี่ปุ่น ระบบ KURAMA ได้นำระบบ GPS และเทคโนโลยีการเชื่อมต่อมาใช้ในการรวบรวมข้อมูลปริมาณรังสีจากสถานีตรวจวัดเคลื่อนที่ (รถยนต์) พร้อมกันหลายสถานีตามเวลาจริง ในส่วนของการวัดทีมงานได้เสนอแนะวิธีการปรับเทียบและแก้ค่าต่างๆ ซึ่งรวมถึงผลจากการกำบังรังสีของตัวรถ ระบบนี้ได้รับการยอมรับจากรัฐบาลท้องถิ่นของเมืองฟุกุชิมะและถูกนำไปใช้สำรวจปริมาณรังสีแกมมาในเมืองฟุกุชิมะและเมืองใกล้เคียงเพื่อรวบรวมข้อมูลสำหรับจัดทำแผนที่รังสีและช่วยในการหา Hot Spot ในบริเวณ ผลการตรวจวัดพบว่ามีความสอดคล้องกับค่าที่ได้จากการวัดจากเครื่องบินซึ่งรวบรวมโดยกระทรวงวิทยาศาสตร์และเทคโนโลยีของญี่ปุ่น

S. Sukihara และคณะ ได้ทำการศึกษาการกระจายตัวของอัตราปริมาณรังสีแกมมาในเมืองฟุกุชิมะโดยใช้การสำรวจทางรถยนต์ระหว่างปี 2011-2012 หัววัดที่ใช้เป็นชนิด NaI(Tl) และทำการวัดทั้งในและนอกรถเพื่อหาตัวปรับค่าจากการกำบังรังสีของตัวรถ ซึ่งจากการทดลองพบว่า อัตราปริมาณรังสีที่วัดได้ภายในและภายนอกรถมีความสัมพันธ์ที่เป็นเส้นตรง

ในระหว่างปี 1994-1996 T. Nagato ได้ทำการศึกษาอัตราปริมาณรังสีแกมมาในเกาะ Hokkaido โดยใช้การวัดบนรถยนต์ ในส่วนของการวัด หัววัดที่ใช้เป็นชนิด Pure Germanium Gamma Spectrometer ความเร็วของรถยนต์ที่ใช้อยู่ระหว่าง 30-60 กิโลเมตรต่อชั่วโมง โดยใช้ระยะเวลาในการวัด 900 วินาที ผลที่ได้ถูกนำมาใช้โดย M. Furukawa ในการหาค่าเฉลี่ยปริมาณรังสีบนพื้นดินในประเทศญี่ปุ่นก่อนเกิดแผ่นดินไหวครั้งใหญ่ในปี 2011

ในส่วนของการใช้ระบบวัดรังสีแกมมาในการตรวจหา Orphan Source ก็รับความสนใจเพิ่มขึ้นในช่วงเริ่มต้นศตวรรษที่ 21 เนื่องจากเริ่มมีกรณีการสูญหาญและโจรกรรมสารกัมมันตรังสีเพิ่มมากขึ้น

H.K. Aage และ U. Korsbech ได้ทำการเปรียบเทียบข้อดีข้อเสียระหว่างการใช้ระบบสำรวจแบบติดรถยนต์และติดเครื่องบิน รวมถึงปัญหาและวิธีการแก้ไขการใช้ระบบสำรวจกับต้นกำเนิดรังสีที่มีความแรงต่ำ

ในปี 2012 P. Kock ได้ตีพิมพ์วิทยานิพนธ์ซึ่งมีหัวข้อว่า Orphan source detection in mobile gamma-ray spectrometry: Improved techniques for background assessment โดยได้บรรยายถึงการตรวจหา Orphan Source โดยใช้ระบบวัดเคลื่อนที่ ซึ่งในกรณีนี้คือระบบติดเครื่องบิน

2.3 ทฤษฎีที่เกี่ยวข้อง

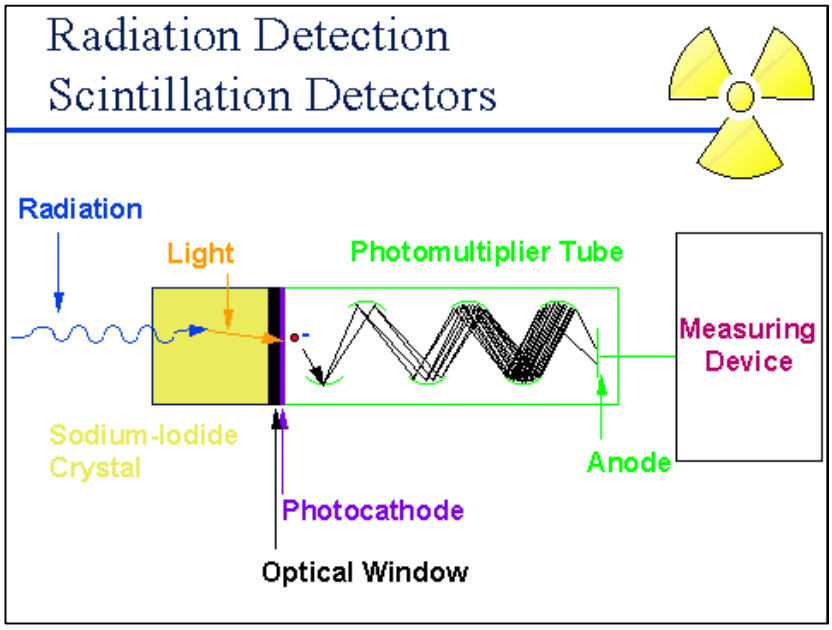
<u>หัววัดรังสี NaI</u>

หัววัดรังสี NaI จัดเป็นหัววัดรังสีแบบ Scintillation ชนิดหนึ่งซึ่งมีความไวต่อรังสีแกมมา แต่ไม่ไวต่อรังสีเบต้าและแอลฟามากนัก จึงเหมาะสำหรับใช้ในสถานการณ์ที่ต้องการตรวจวัดรังสีแกมมาเป็นหลัก เช่น ในการสำรวจบนรถยนต์หรือพาหนะที่เคลื่อนที่ตลอดเวลาและครอบคลุมบริเวณกว้างซึ่งมักจะไม่พบรังสีเบต้าหรือแอลฟาที่มีความสามารถในการทะลุทะลวงและระยะการเคลื่อนที่ในอากาศที่ต่ำกว่ารังสีแกมมามากนัก ข้อดีของหัววัดรังสี NaI ซึ่งทำให้เป็นที่นิยมสำหรับงานสำรวจคือสามารถใช้งานได้ในช่วงอุณหภูมิที่ค่อนข้างกว้างโดยที่ยังมี efficiency และ resolution อยู่ในระดับที่ยอมรับได้ ซึ่งหากต้องการข้อมูลที่แม่นยำมากยิ่งขึ้นใน

ภายหลังก็อาจทำการตรวจซ้ำ ณ จุดที่ต้องการโดยใช้หัววัดรังสีชนิดอื่น เช่น High-Purity Germanium (HPGe) ที่มี efficiency และ resolution สูงกว่าแต่จำเป็นต้องใช้ในสภาวะ cryogenic ได้

การทำงานของหัววัดรังสี NaI (และของหัววัด Scintillation ชนิดอื่น) อาศัยคุณสมบัติ Luminescence ของผลึก NaI ซึ่งปลดปล่อยแสงเมื่อมีรังสีก่อไอออน (หรือในที่นี้คือรังสีแกมมา) ตกกระทบ โดยความเข้มข้นของแสงที่ได้จะแปรผันตามพลังงานของรังสี

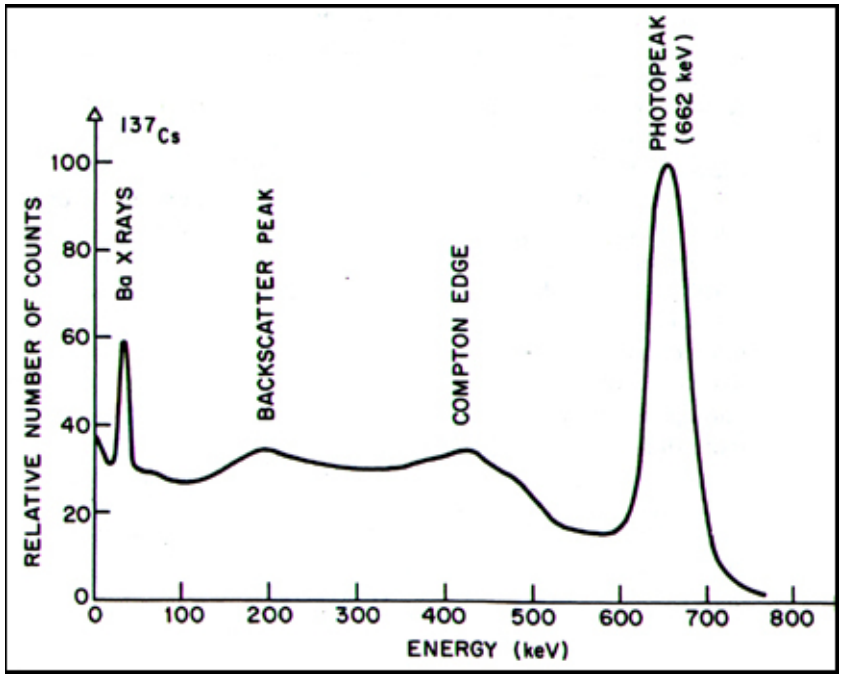
โดยทั่วไปในการวัดรังสี หัววัดรังสี NaI จะถูกต่อเข้ากับ Photomultiplier ดังรูปที่ 2.1 ซึ่งประกอบด้วย Photocathode ที่อาศัยกระบวนการ Photoelectric ในการเปลี่ยนแสงที่ผลิตจากหัววัดเป็นอิเล็กตรอน (กระแส) และ dynode ที่อาศัยกระบวนการ Secondary Emission ในการเพิ่มปริมาณอิเล็กตรอน (เพิ่มกระแส)



รูปที่ 2.1 การทำงานของหัววัดรังสี NaI (ที่มา http://www.ntanet.net/)

กระแสที่ได้ซึ่งมีลักษณะเป็นสัญญาณ pulse จะส่งต่อไปยังอุปกรณ์นับวัดสัญญาณซึ่งอาจประกอบด้วย Multichannel Analyzer ที่เก็บข้อมูลจำนวน count ของ pulse ที่เกิดขึ้นตามขนาดของ pulse ซึ่งแปรผันตามพลังงานของรังสีที่มักแสดงผลออกมาเป็น Spectrum ระหว่างพลังงานและจำนวน count ดังรูปที่ 2.2 หรือ Single-channel Analyzer ที่เก็บข้อมูลจำนวน count ของ pulse ที่มีขนาดในช่วงที่กำหนด (โดยไม่แบ่งตามขนาด) เท่านั้น

ทุกครั้งที่มีรังสีตกกระทบหัววัด จะมีการสร้างสัญญาณ pulse ขึ้นตามพลังงานของรังสี ดังนั้นจำนวน pulse จึงแปรผันตามความเข้มของรังสีที่พลังงานต่างๆ แต่อย่างไรก็ดี ในระหว่างการแปลงสัญญาณจากรังสีแกมมาเป็นแสงและจากแสงเป็นอิเล็กตรอน จะมีปรากฏการณ์อื่นๆ เกิดขึ้นด้วย ซึ่งที่สำคัญคือ Compton Scattering, Backscattering และ Pair Production ซึ่งทำให้ได้อิเล็กตรอนที่พลังงาน (กระแส) ไม่แปรผันตามพลังงานของรังสีออกมาด้วย ดังปรากฏในรูปที่ 2.2



รูปที่ 2.2 ตัวอย่าง Spectrum ที่เกิดจาก Cs-137 (ที่มา http://www.people.vcu.edu/~mhcrosthwait/clrs322/Pulseanalysis.htm)

<u>ปัจจัยที่มีผลต่อการวัดรังสี</u>

อุณหภูมิ

แม้ว่าหัววัดรังสี NaI มีความสามารถในการวัดรังสีในช่วงอุณหภูมิที่ค่อนข้างกว้าง แต่อุณหภูมิสามารถส่งผลต่อประสิทธิภาพของการวัดรังสีได้ ซึ่ง P. L. Reeder และ D. C. Stromswold ได้ทำการศึกษาประสิทธิภาพของหัววัดรังสี NaI(Tl) ในช่วงอุณหภูมิระหว่าง -50 C และ 60 C สำหรับวัดรังสีแกมมาในช่วงพลังงานระหว่าง 25 keV และ 2500 keV และพบว่าขนาดของ pulse ลดลงและ resolution เพิ่มขึ้นเมื่ออุณหภูมิสูงหรือต่ำกว่าอุณหภูมิห้องมากๆ และพบว่าขนาดของ pulse สูงสุดที่อุณหภูมิ 0 องศา (ซึ่งแตกแต่งจากข้อสรุปของการศึกษาที่ผ่านมาเนื่องจากเงื่อนไขการทดลอง) ที่อุณหภูมิ 60 C พบว่าขนาดของ pulse ลดลง 25% แต่ resolution ไม่เพิ่มขึ้นมากนัก

โดยปกติแล้ว หัววัดรังสีจะได้รับการปรับเทียบที่อุณหภูมิห้อง ดังนั้น จึงควรควบคุมอุณหภูมิในระหว่างการวัดให้ใกล้เคียงอุณหภูมิห้องที่สุด

การกำบังรังสี

แม้รังสีแกมมาจะมีคุณสมบัติในการทะลุทะลวงสูง แต่การวัดรังสีบนรถยนต์จำเป็นต้องคำนึงถึงการกำบังรังสีจากตัวรถและตำแหน่งการตั้งหัววัดในรถ ดังนั้น เพื่อให้ได้ค่าที่แท้จริงจากการวัด จึงจำเป็นต้องทำการแก้ค่าที่วัดได้ด้วย Shielding Factor (f_s) ซึ่งเป็นอัตราส่วนระหว่างค่าที่วัดได้ภายนอกรถ ณ ความสูง 1 เมตรจากพื้นดิน กับ ค่าที่วัดได้ ณ จุดวางหัววัดภายในรถ

สภาพภูมิอากาศ

สภาพภูมิอากาศมีผลต่อความสามารถในการตรวจวัดรังสีในธรรมชาติ ซึ่งรวมถึงอุณหภูมิดังกล่าวไว้ข้างต้น ปริมาณน้ำฝนเป็นอีกปัจจัยหนึ่งที่ลดประสิทธิภาพในการสำรวจอย่างชัดเจน เนื่องจากน้ำมีคุณสมบัติในการกำบังรังสีแกมมา จึงทำให้ปริมาณรังสีที่วัดได้ในระหว่างหรือหลังฝนตกลดลงอย่างเห็นได้ชัด ดังนั้น ในการสำรวจจึงจำเป็นต้องทำการเก็บข้อมูลสภาพภูมิอากาศไว้ด้วย

<u>รังสีแกมมาจากพื้นดิน</u>

ในสภาวะปกติ การสำรวจรังสีด้วยรถยนต์จะวัดพบรังสีแกมมาในธรรมชาติที่มาจากพื้นดิน ซึ่งรวมถึงรังสีที่มาจาก K-40 (peak @ 1.46 MeV), ห่วงโซ่การสลายตัวของ U-238 (โดยเฉพาะ Bi-214, peak @ 1.76 MeV) และห่วงโซ่การสลายตัวของ Th-232 (โดยเฉพาะ Ti-208, peak @ 2.62 MeV) รังสีเหล่านี้มีค่าพลังงานอยู่ในช่วงระหว่าง 0.5 - 3.0 MeV และจะเกิด Compton Scattering ในช่วง 0.1 – 0.6 MeV และ Photoelectric Absorption ในช่วง 0 – 0.2 MeV

รังสีจากพื้นดิน เป็นส่วนหนึ่งของ Background Radiation ซึ่งจะไม่เท่ากันในแต่ละพื้นที่ ขึ้นอยู่กับลักษณะภูมิประเทศ (เช่น ความสูง ชนิดของชั้นหิน เป็นต้น) และกิจกรรมที่อยู่รอบๆ (เช่น การทำเหมือง อาคาร ถนน เป็นต้น) และสามารถเพิ่มขึ้นหากมีเหตุการณ์ทางรังสีเกิดขึ้น เช่น การระเบิดของโรงไฟฟ้านิวเคลียร์ หรือการลักลอบนำเอาวัสดุกัมมันตรังสีไปทิ้งไว้ในบริเวณ อย่างไรก็ดี มนุษย์ไม่สามารถหลีกเลี่ยงการได้รับรังสีเหล่านี้ได้ ซึ่งจากการศึกษาโดย UNSCEAR ซึ่งเห็นหน่วยงานภายใต้สหประชาชาติ (UN) พบว่า 20% ของปริมาณรังสีที่มนุษย์ได้รับในธรรมชาติมาจากพื้นดิน (ไม่รวมส่วนที่หายใจหรือบริโภคเข้าไป)

ในการสำรวจรังสีด้วยรถยนต์ หัววัดรังสีที่ใช้จึงได้รับการปรับเทียบกับ Isotopes ดังกล่าว (K-40, Bi-214, Ti-208) ก่อนนำไปวัดจริง เพื่อใช้ในการประเมินปริมาณรังสีที่ได้รับ ซึ่งหากมีรังสีที่มาจากธาตุชนิดอื่น เช่น Cs-137 หรือ Co-60 ซึ่งโดยปกติไม่มีอยู่ในธรรมชาติ ก็อาจทำให้ปริมาณรังสีที่ได้รับที่ประเมินได้ไม่ถูกต้อง แต่จะสามารถบอกความผิดปกติในบริเวณได้

บทที่ 3

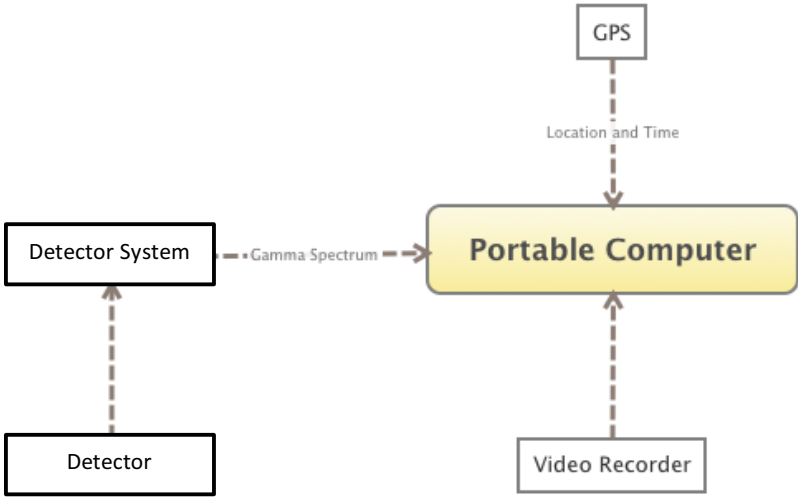
การดำเนินการและผลผลิต

3.1 ออกแบบและพัฒนาระบบวัดปริมาณรังสีแกมมาแบบติดรถยนต์

ระบบวัดรังสีแกมมาแบบติดรถยนต์ได้รับการออกแบบให้มีลักษณะตามรูปที่ 3.1 โดยมีความสามารถในการรับข้อมูลได้ 3 ประเภท คือ

- ข้อมูลรังสี (จำนวน Count ในช่วงเวลาที่กำหนด)
- ข้อมูล GPS (เวลาและตำแหน่ง
- ข้อมูลภาพ (รูปภาพหรือวีดีโอ)

โดยข้อมูลดังกล่าวจะถูกจัดเก็บไว้ในฐานข้อมูลที่ได้รับการพัฒนาขึ้นสำหรับนำไปใช้ในการวิเคราะห์ต่อไป



รูปที่ 3.1 ระบบวัดรังสีแกมมาแบบติดรถยนต์

มีรายละเอียดของอุปกรณ์ที่ใช้ในการวิจัยดังต่อไปนี้

Detector

สำหรับการวิจัยนี้ ได้เลือกใช้หัววัดรังสีแกมมาชนิด NaI ขนาด 1 และ 5 นิ้ว ซึ่งมีคุณลักษณะดังปรากฏในตารางที่ 3.1 และ 3.2 ตามลำดับ

ตารางที่ 3.1 คุณลักษณะของหัววัดรังสีแกมมาชนิด NaI ขนาด 1x1 นิ้ว

Manufacturer	LUDLUM Measurements, Inc.
Model	44-2

Image		
Indicated Use	Low-level, wide-energy gamma detection	
Detector Type	NaI(Tl) scintillator, 2.5 cm (1 in) dia. x 2.5 cm (1 in) thick	
Efficiency (4π)	^{125}I	7%
	^{57}Co	10%
	^{137}Cs	3%
	^{60}Co	3%
Sensitivity (^{137}Cs gamma)	175 cpm per μR/hr (typical)	
Recommended Energy Range	20 keV to 1.5 MeV	
Energy Response	Energy dependent (see Energy Response Curve on Documents tab)	
Background	1800 cpm	
Photomultiplier Tube	2.86 cm (1.123 in) diameter, magnetically shielded	
Operating Voltage	500 to 1200 V	
Temperature Range	-15 to 50 °C (5 to 122 °F) May be certified for -40 to 65 °C (-40 to 150 °F)	
Connector	Series "C"	
Construction	Aluminum with beige powder coat	
Size (D x L)	5.1 x 18.5 cm (2 x 7.3 in)	
Weight	0.5 kg (1 lb)	

ตารางที่ 3.2 คุณลักษณะของหัววัดรังสีแกมมาชนิด NaI ขนาด 5x5 นิ้ว

Manufacturer	BICRON CORP.
Model	5M5/5

Image	
Indicated Use	Low-level, wide-energy gamma detection
Detector Type	NaI(Tl) scintillator, 5 in dia. x 5 in thick

Detector System

ระบบวัดที่ใช้ในการแปรผลสัญญาณจากหัววัด เป็นชนิด Scaler Ratemeter มีคุณลักษณะดังปรากฏในตารางที่ 3.3 สามารถรับส่งข้อมูลกับระบบคอมพิวเตอร์แบบ real-time ด้วยมาตรฐานการสื่อสารแบบ RS-232 โดยวัดปริมาณรังสีเป็นจำนวน Count ในช่วงเวลาที่กำหนด

ตารางที่ 3.3 คุณลักษณะของระบบวัดรังสี

Manufacturer	LUDLUM Measurements, Inc.
Model	2200
Image	
Indicated Use	Single channel analyzing, gross counting

Suggested Detectors	GM, scintillation, proportional
Connector	Series “C” (others available)
Scaler	Six-digit LED display with a range of 0 to 999999 counts
Scaler Linearity	Reading within 2% of true value
Timer	Push-wheel adjustment from 0 to 999 minutes with selectable x0.1 and x1 multipliers
Multipliers	x1, x10, x100, x1000
Ratemeter	0 to 500,000 cpm total range
Ratemeter Linearity	Reading within 10% of true value
Response	Toggle switch for FAST (4 seconds) or SLOW (22 seconds), from 10% to 90% of final reading
Meter Dial	0 to 500 cpm, 0 to 2.5 kV, BAT TEST
Meter	6.4 cm (2.5 in.) arc, 1 mA movement analog type
Zero	Pushbutton to zero meter
High Voltage	Adjustable from 200 to 2500 V (will support 60 MΩ scintillation loads)
Threshold	Adjustable from 1.0 to 10.0
Window	Adjustable from 0 to 10.0 above the threshold setting (can be enabled or disabled)
Discriminator	Adjustable from 2 to 100 mV at threshold setting of 1.00
RS-232	9-pin connector allowing for printer or computer interface
Power	95 to 250 Vac, 50–60 Hz or 4 “D” cell batteries
Battery Life	Typically 120 hours with alkaline batteries (battery condition can be checked on meter)
Temperature Range	-20 to 50 °C (-4 to 122 °F) May be certified to operate from -40 to 65 °C (-40 to 150 °F)
Size (H x W x L)	21.6 x 12.7 x 21.6 cm (8.5 x 5 x 8.5 in.) (excluding handle)
Weight	3.4 kg (7.5 lb), including battery

GPS

GPS receiver ที่ใช้ในงานวิจัยนี้มีคุณลักษณะดังปรากฏในตารางที่ 3.4 สามารถรส่งข้อมูลกับระบบคอมพิวเตอร์แบบ real-time ด้วยมาตรฐาน NMEA

ตารางที่ 3.4 คุณลักษณะของอุปกรณ์รับสัญญาณ GPS

Manufacturer	GlobalSat
Model	BU-353S4

Image	
Frequency	L1, 1575.42 MHz
C/A code	1.023 MHz chip rate
Channels	48 channel all-in-view tracking
Operating temperature	-40 C to +85 C
Output message	NMEA 0183 GGA, GSA, GSV, RMC, VTG, GLL
Dimension	53mm diameter, 19.2mm height
Sensitivity	-163 dBm
Main power input	4.5V – 6.5V DC input
Baud rate	4800 bps
Cable length	1.5m

Video Recorder

อุปกรณ์บันทึกภาพที่ใช้ในการวิจัยนี้มีคุณลักษณะดังปรากฏในตารางที่ 3.5 สามารถบันทึกภาพและวีดีโอมุมกว้าง (122.6 องศา แนวนอน) ได้ แต่มีข้อจำกัดในด้านการสื่อสาร คือ ไม่อนุญาตให้รับส่งข้อมูลแบบ real-time ด้วย software ภายนอกโดยตรงได้

ตารางที่ 3.5 คุณลักษณะของอุปกรณ์บันทึกภาพ

Manufacturer	GoPro
Model	Hero 4 Black
Image	
Widescreen Video Capture	Yes

Optical Sensor Type	CMOS
Wireless Connection	Bluetooth, Wireless LAN
Digital Video Format	H.264
Image Recording Format	JPEG
Max Video Resolution	3840 x 2160
Interfaces Provided	HDMI, composite video/audio
Image Recording Format	JPEG
Max Video Resolution	3840 x 2160
Effective Photo Resolution	12.0 MP
ISO (Max)	6400
Digital Video Format	H.264
Camcorder Sensor Resolution	12.0 MP
Widescreen Video Capture	Yes
Optical Sensor Type	CMOS

Portable Computer

ในงานวิจัยนี้ ได้ทำการพัฒนาโปรแกรมคอมพิวเตอร์ขึ้นโดยเฉพาะสำหรับใช้ในการรับส่งข้อมูลระหว่างระบบสำรวจต่างๆ และระบบคอมพิวเตอร์ ซึ่งโปรแกรมที่ได้พัฒนาขึ้นสามารถใช้ได้ทั้งบนระบบ Windows และระบบ MacOS ดังรายละเอียดที่ปรากฏในหัวข้อที่ 3 ต่อไป

3.2 ออกแบบและพัฒนาระบบจัดการข้อมูลปริมาณรังสีแกมมา

เนื่องจากข้อมูลที่เกิดขึ้นจากระบบสำรวจรังสีแกมมาแบบติดรถยนต์จะมีปริมาณมาก จึงจำเป็นต้องพัฒนาระบบจัดเก็บข้อมูลขึ้นเพื่อให้สามารถสืบค้นและนำไปใช้ในการวิเคราะห์เพิ่มเติมได้ในภายหลัง ในงานวิจัยนี้ได้นำระบบจัดการฐานข้อมูล MySQL มาใช้ เนื่องจากมีความสามารถมาการจัดเก็บข้อมูลปริมาณมาก และสามารถใช้ชุดคำสั่ง SQL ในการสืบค้น (Query) และจัดเรียง (Sort) ข้อมูลได้อย่างรวดเร็ว โดยได้ออกแบบให้ประกอบด้วย 6 ฐานข้อมูลย่อย คือ

- dataset
- detector
- gps
- camera
- dose
- video

ซึ่งแต่ละฐานข้อมูลย่อยใช้สำหรับเก็บข้อมูลดังนี้

dataset

สำหรับจัดเก็บข้อมูลเกี่ยวกับชุดข้อมูล ดังต่อไปนี้

ชื่อข้อมูล	คำอธิบาย	ลักษณะข้อมูล
DataSetID	ID เฉพาะสำหรับชุดข้อมูล	จำนวนเต็ม (ระบบสร้างขึ้น)
SetName	ชื่อชุดข้อมูล	ตัวอักษรไม่เกิน 30 ตัว
SetDescription	คำอธิบายชุดข้อมูล	ตัวอักษรไม่เกิน 255 ตัว
PersonAcquire	ชื่อผู้เกิดข้อมูล	ตัวอักษรไม่เกิน 30 ตัว
PersonUpload	ชื่อผู้นำข้อมูลเข้าฐานข้อมูล	ตัวอักษรไม่เกิน 30 ตัว
DateUpload	วันที่นำข้อมูลเข้าฐานข้อมูล	วันที่
DetectorID	ID สำหรับระบบวัดรังสีที่ใช้	จำนวนเต็ม
GpsID	ID สำหรับระบบ GPS ที่ใช้	จำนวนเต็ม
CameraID	ID สำหรับระบบบันทึกภาพที่ใช้	จำนวนเต็ม

detector

สำหรับจัดเก็บข้อมูลที่เกี่ยวกับระบบวัดรังสี ดังต่อไปนี้

ชื่อข้อมูล	คำอธิบาย	ลักษณะข้อมูล
DetectorID	ID เฉพาะสำหรับระบบวัดรังสี	จำนวนเต็ม (ระบบสร้างขึ้น)
DetectorName	ชื่อระบบวัดรังสี	ตัวอักษรไม่เกิน 30 ตัว
DetectorDescription	คำอธิบายระบบวัดรังสี	ตัวอักษรไม่เกิน 255 ตัว
DetectorManufacturer	ชื่อผู้ผลิตระบบวัดรังสี	ตัวอักษรไม่เกิน 30 ตัว

gps

สำหรับจัดเก็บข้อมูลที่เกี่ยวกับระบบ GPS ดังต่อไปนี้

ชื่อข้อมูล	คำอธิบาย	ลักษณะข้อมูล
GpsID	ID เฉพาะสำหรับระบบ GPS	จำนวนเต็ม (ระบบสร้างขึ้น)
GpsName	ชื่อระบบ GPS	ตัวอักษรไม่เกิน 30 ตัว
GpsDescription	คำอธิบายระบบ GPS	ตัวอักษรไม่เกิน 255 ตัว
GpsManufacturer	ชื่อผู้ผลิตระบบ GPS	ตัวอักษรไม่เกิน 30 ตัว

camera

สำหรับจัดเก็บข้อมูลที่เกี่ยวกับระบบบันทึกภาพ ดังต่อไปนี้

ชื่อข้อมูล	คำอธิบาย	ลักษณะข้อมูล
CameraID	ID เฉพาะสำหรับระบบบันทึกภาพ	จำนวนเต็ม (ระบบสร้างขึ้น)
CameraName	ชื่อระบบบันทึกภาพ	ตัวอักษรไม่เกิน 30 ตัว
CameraDescription	คำอธิบายระบบบันทึกภาพ	ตัวอักษรไม่เกิน 255 ตัว
CameraManufacturer	ชื่อผู้ผลิตระบบบันทึกภาพ	ตัวอักษรไม่เกิน 30 ตัว

dose

สำหรับจัดเก็บข้อมูลจากการวัด ดังต่อไปนี้

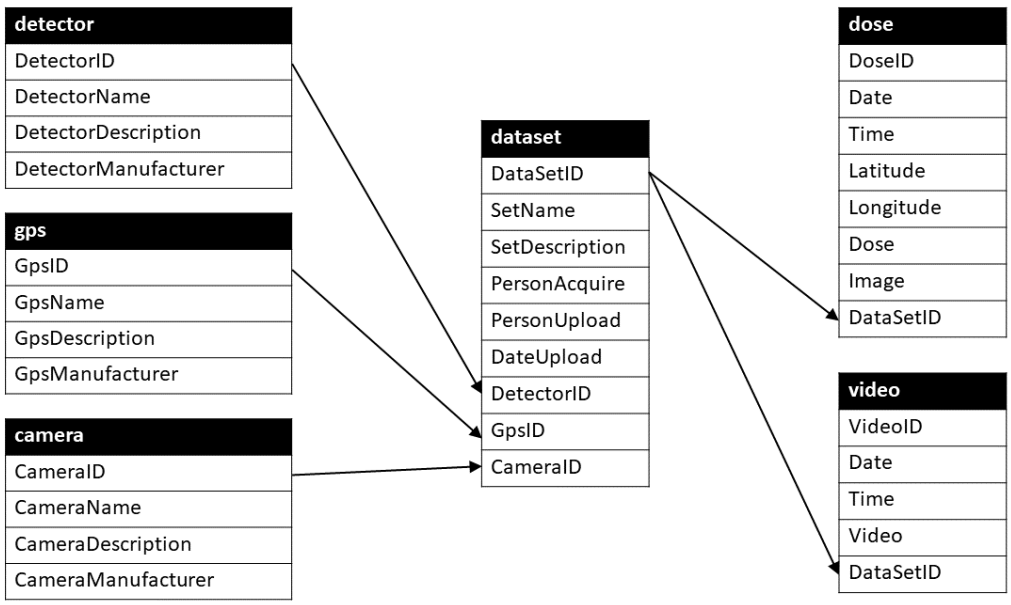
ชื่อข้อมูล	คำอธิบาย	ลักษณะข้อมูล
DoseID	ID เฉพาะสำหรับข้อมูล	จำนวนเต็ม (ระบบสร้างขึ้น)
Date	วันที่เก็บข้อมูล	วันที่
Time	เวลาเก็บข้อมูล	เวลา
Latitude	Latitude ของข้อมูล	จำนวน (Double)
Longitude	Longitude ของข้อมูล	จำนวน (Double)
Dose	ปริมาณรังสี	จำนวน (Double)
Image	ตำแหน่งไฟล์ภาพ	ตัวอักษรไม่เกิน 255 ตัว
DataSetID	ID เฉพาะสำหรับชุดข้อมูล	จำนวนเต็ม

video

สำหรับจัดเก็บข้อมูลจากการวัด ดังต่อไปนี้

ชื่อข้อมูล	คำอธิบาย	ลักษณะข้อมูล
VideoID	ID เฉพาะสำหรับวีดีโอ	จำนวนเต็ม (ระบบสร้างขึ้น)
Date	วันที่เก็บข้อมูล	วันที่
Time	เวลาเก็บข้อมูล	เวลา
Video	ตำแหน่งไฟล์วีดีโอ	ตัวอักษรไม่เกิน 255 ตัว
DataSetID	ID เฉพาะสำหรับชุดข้อมูล	จำนวนเต็ม

ทั้ง 6 ฐานข้อมูลย่อย มีความสัมพันธ์ของข้อมูลดังแสดงในรูปที่ 3.2

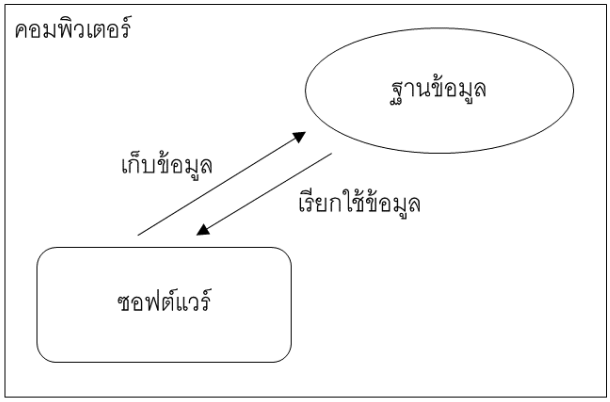


รูปที่ 3.2 ความสัมพันธ์ระหว่างข้อมูลในฐานข้อมูล

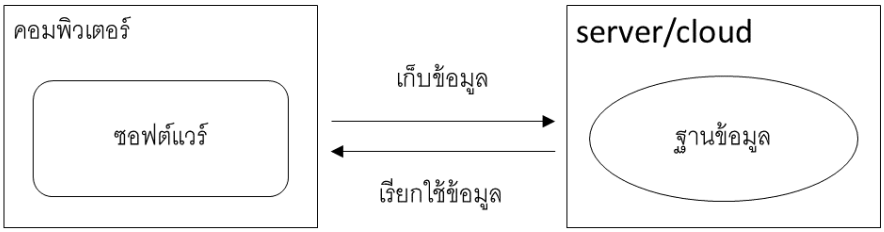
3.3 พัฒนาโปรแกรมเชื่อมต่ออัตโนมัติระหว่างระบบสำรวจและระบบจัดการข้อมูล

เพื่อให้การรวบรวมและประเมินผลข้อมูลสามารถดำเนินการได้อย่างรวดเร็วและมีประสิทธิภาพ จึงได้ทำการพัฒนาโปรแกรมคอมพิวเตอร์ขึ้นเพื่อใช้ในการเรียกเก็บข้อมูลจากระบบวัดต่างๆ ขึ้นไว้ในฐานข้อมูล และนำข้อมูลจากฐานข้อมูลออกมาใช้ในการวิเคราะห์และประเมินผล โดยตัวโปรแกรมเขียนขึ้นโดยใช้ภาษา Java และอาศัย Java Environment (Java8) ในการดำเนินการ (ภาคผนวก ก) ดังนั้น จึงสามารถนำไปใช้งานได้ใน Operating System ที่หลากหลาย รวมถึง Microsoft Windows, Mac OS X, Solaris และ Linux โดยได้ทำการทดสอบใน Microsoft Windows 10 และ Mac OSX 10.10.5

การเชื่อมต่อโปรแกรมและฐานข้อมูลจะดำเนินการผ่านระบบอินเตอร์เน็ตเพื่อให้สามารถใช้งานในลักษณะต่างๆ ดังรูปที่ 3.3 ได้ โดยในแบบ 3.3(ก) โปรแกรมและฐานข้อมูลจะติดตั้งในระบบคอมพิวเตอร์เดียวกัน ส่วนในแบบ 3.3(ข) โปรแกรมและฐานข้อมูลจะติดตั้งไว้แยกกัน ซึ่งวิธีนี้สามารถใช้คอมพิวเตอร์มากกว่าหนึ่งเครื่องในการเข้าถึงข้อมูลได้พร้อมกัน



(ก)



(ข)

รูปที่ 3.3 ลักษณะการเชื่อมต่อระหว่างโปรแกรมและฐานข้อมูล

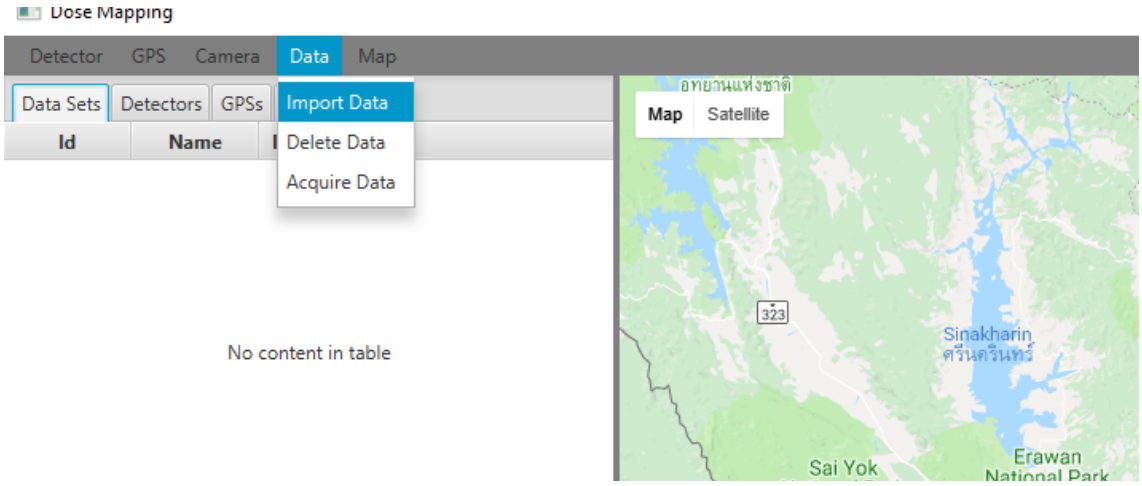
โดยสามารถแบ่งโหมดการทำงานของโปรแกรมออกได้ดังนี้

<u>โหมดเรียกอ่านข้อมูลจากไฟล์ข้อมูลโดยตรง</u>

สามารถเรียกอ่านข้อมูลรังสีโดยตรงจากไฟล์ชนิด Text (ASCII) ที่มีรูปแบบข้อมูลดังต่อไปนี้

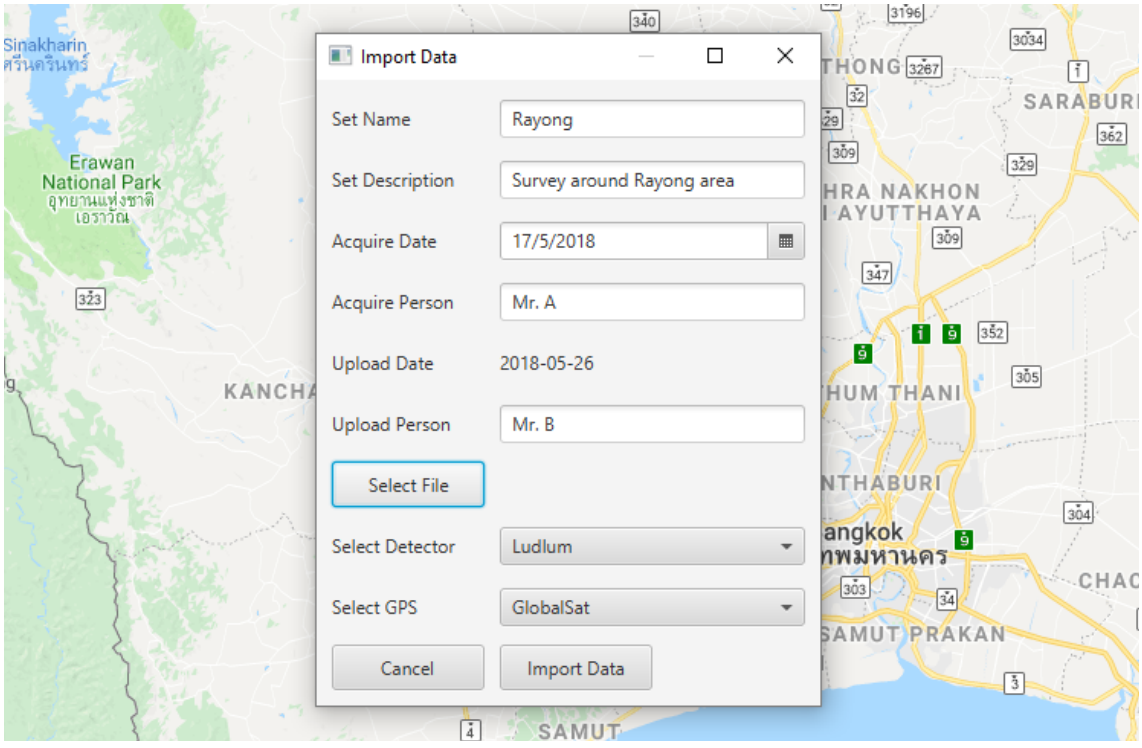
MM/DD/YYYY	HH:MM:SS	LATITUDE	LONGITUDE	DOSE
MM/DD/YYYY	HH:MM:SS	LATITUDE	LONGITUDE	DOSE
...				

การเรียกอ่านข้อมูลสามารถทำได้ได้โดยใช้ฟังค์ชัน Data > Import Data จากเมนูหลัก ดังรูปที่ 3.4

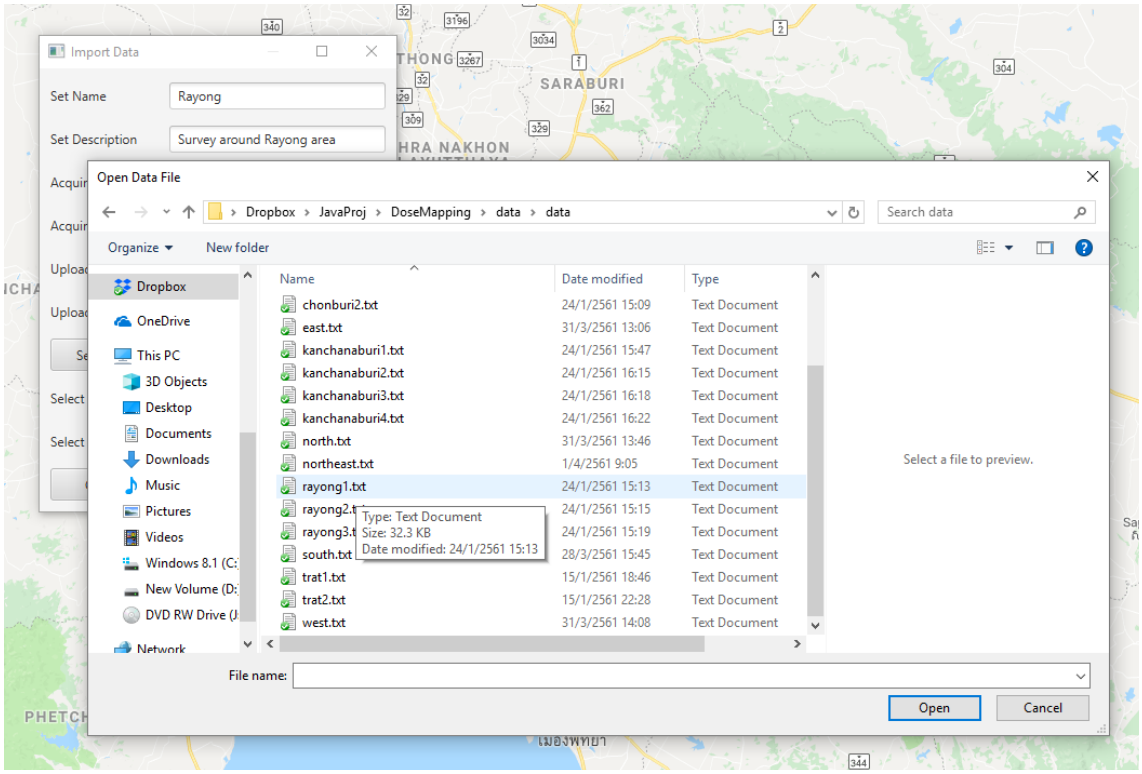


รูปที่ 3.4 ฟังค์ชัน Import Data สำหรับใช้อ่านข้อมูลจากไฟล์โดยตรง

เมื่อเรียกใช้ฟังค์ชัน Import Data จะมีหน้าต่างใหม่แสดงขึ้นดังรูปที่ 3.5 สำหรับกรอกข้อมูลที่จำเป็น และสามารถใช้ปุ่ม Select File ในการโหลดข้อมูลดังรูปที่ 3.6

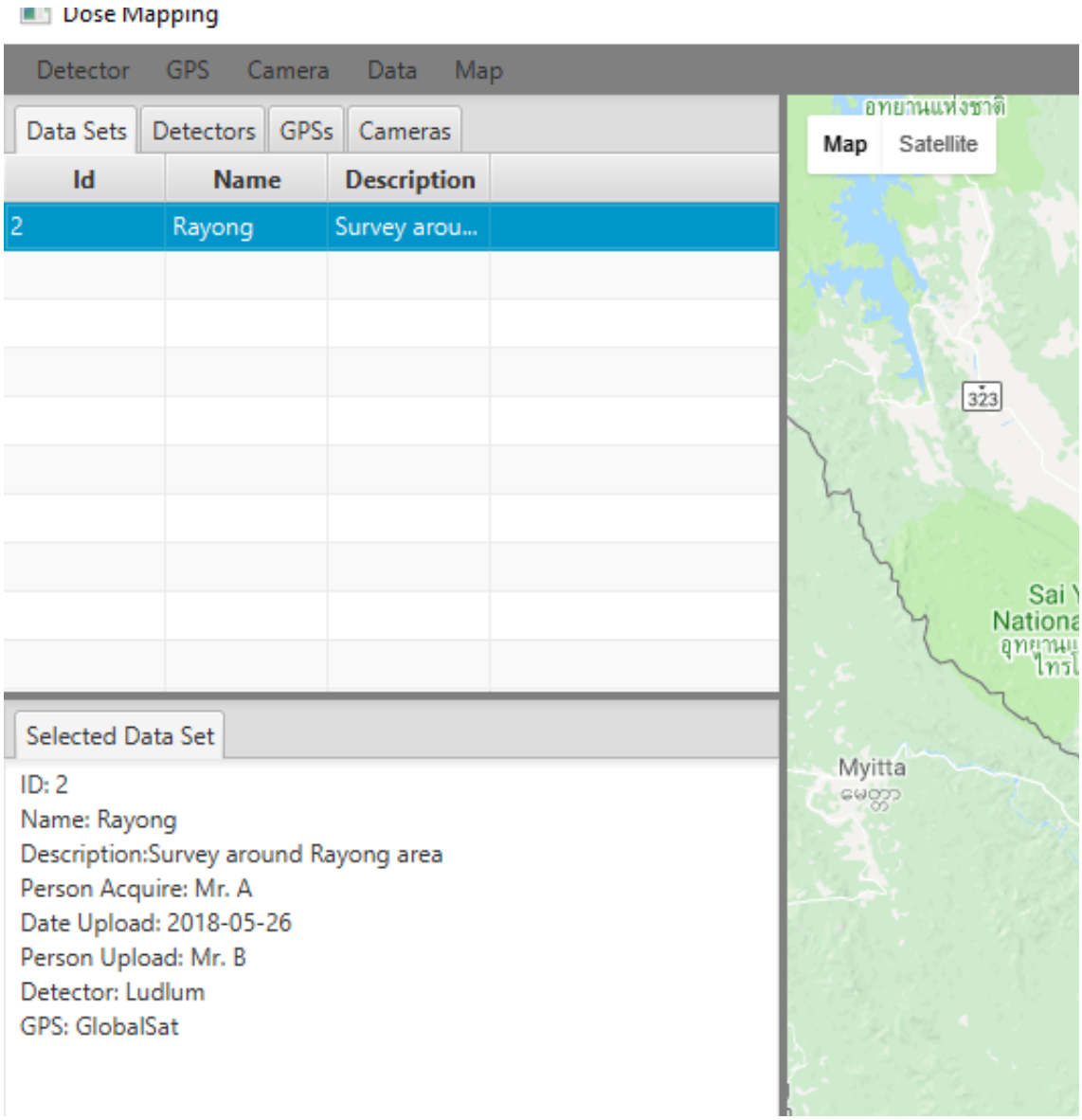


รูปที่ 3.5 หน้าต่างสำหรับกรอกข้อมูลเพื่อ Import



รูปที่ 3.6 หน้าต่างสำหรับเลือกไฟล์ข้อมูลที่ต้องการ Import

เมื่อดำเนินการ Import Data เสร็จสิ้น ข้อมูลที่ถูกอ่านจะถูกเก็บขึ้นไปไว้ในฐานข้อมูล และจะสามารถเรียกใช้ข้อมูลได้จากแท็ป Data Sets ดังรูปที่ 3.7 ซึ่งไม่จำเป็นต้องทำการ Import Data ใหม่เมื่อเริ่มโปรแกรมอีกครั้ง



รูปที่ 3.7 ข้อมูลที่ถูกอ่านและเก็บไว้ในฐานข้อมูลจะถูกแสดงในเมนู Data Sets

<u>โหมดการวัดแบบ real-time</u>

โปรแกรมจะทำการเชื่อมต่อกับเครื่องมือสำรวจทั้ง 3 ชนิด (ระบบวัดรังสี GPS และระบบรับภาพ) และจะทำการเก็บข้อมูลเป็นช่วงๆ (เช่น ทุก 30 วินาที) โดยมีการดำเนินการดังนี้

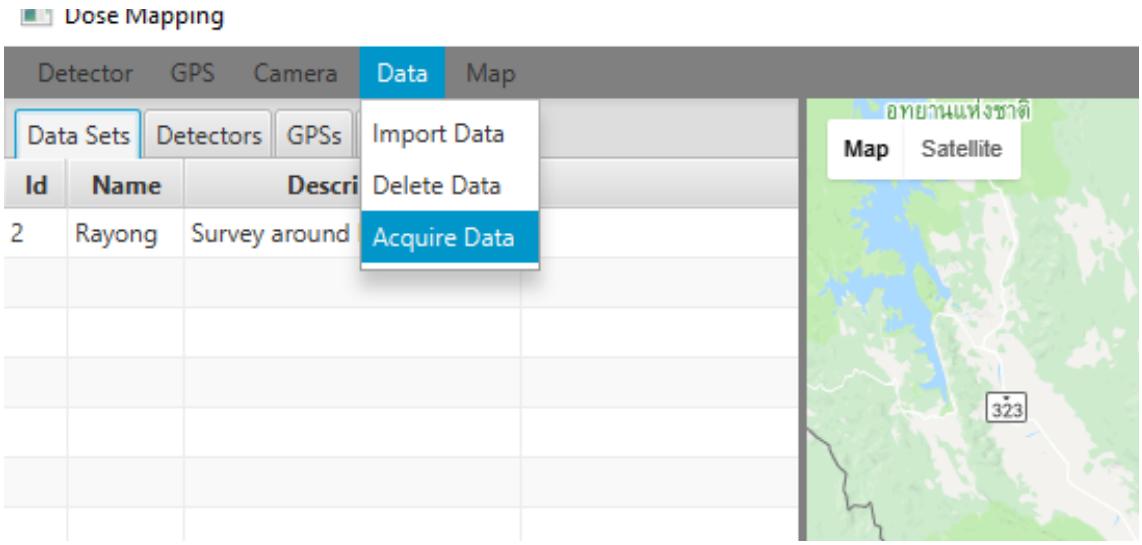
- สั่งให้ระบบวัดรังสีทำการนับวัดในช่วงเวลาตามต้องการ และรายงานข้อมูลการนับวัดเมื่อเสร็จสิ้นในแต่ละช่วง ก่อนจะเริ่มนับวัดในช่วงต่อไป
- รับข้อมูลตำแหน่งจาก GPS
- สั่งให้ระบบรับภาพถ่ายภาพหรือวีดีโอ

ส่วนข้อมูลที่ได้จะถูกประมวลและดำเนินการต่อโดยอัตโนมัติดังนี้

- ข้อมูลรังสีและ gps จะจัดเก็บเป็นไฟล์ชนิด Text (ASCII) ในคอมพิวเตอร์
- ข้อมูลภาพจะจัดเก็บเป็นไฟล์ในคอมพิวเตอร์

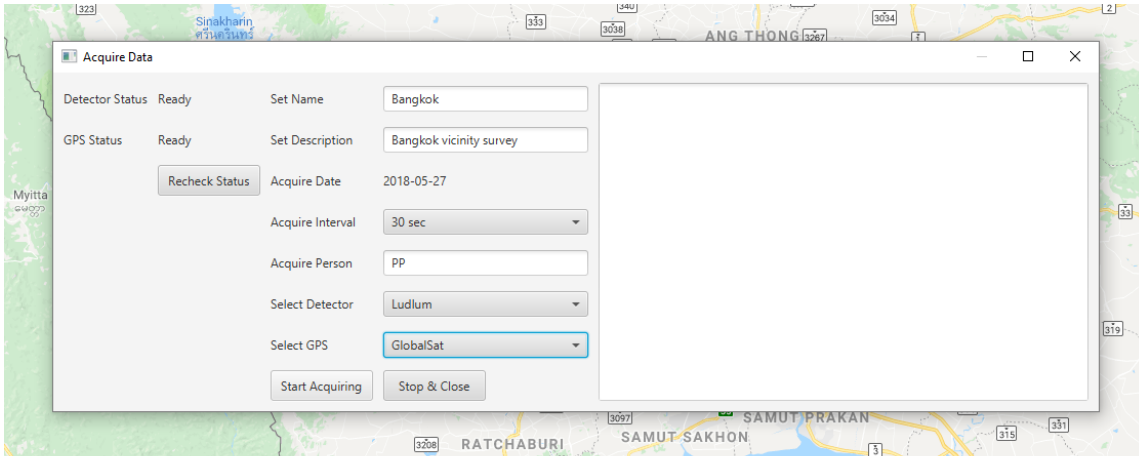
- ข้อมูลวีดีโอจะจัดเก็บเป็นไฟล์ในคอมพิวเตอร์
- ข้อมูลรังสี gps ตำแหน่งไฟล์ภาพ ตำแหน่งไฟล์วีดีโอจะจัดเก็บขึ้นฐานข้อมูล (ข้อมูลภาพ และ วีดีโอ ยังไม่มีวิธีการจัดเก็บไปยังเซิร์ฟเวอร์โดยตรง จึงยังไม่สามารถดำเนินการเชื่อมต่อตามรูปที่ 3.3(ข) ได้เต็มที่)

สามารถเริ่มโหมดการวัดแบบ real-time ได้โดยใช้ฟังค์ชัน Data > Acquire Data ดังรูปที่ 3.8



รูปที่ 3.8 การเริ่มโหมดการวัดแบบ real-time ด้วยฟังค์ชัน Acquire Data

โดยจะมีหน้าต่างใหม่ขึ้นดังรูปที่ 3.9 เพื่อแสดงความพร้อมของอุปกรณ์สำรวจ (ระบบวัดรังสี และ gps) และช่องสำหรับกรอกข้อมูลต่างๆ ที่จำเป็น



รูปที่ 3.9 หน้าต่างควบคุมการวัดแบบ real-time

การวัดจะเริ่มขึ้นเมื่อกดปุ่ม Start Acquiring โดยใช้ช่วงการวัดแต่ละช่วงเท่ากับระยะเวลาที่กำหนดไว้ใน Acquire Interval ซึ่งสามารถหยุดวัดชั่วคราวได้โดยกดปุ่ม Pause Acquiring (ปุ่มเดียวกับ Start Acquiring) และหยุดการวัดโดยกดปุ่ม Stop & Close

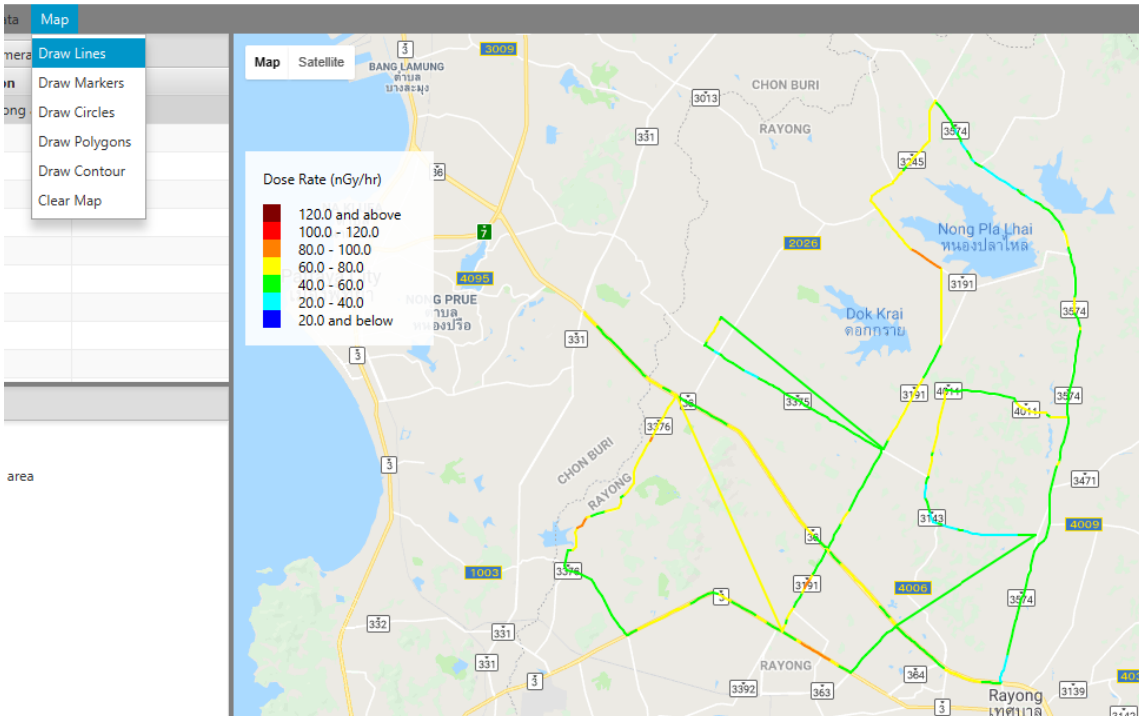
เวลาที่บันทึกเป็นข้อมูลของแต่ละจุด จะใช้เวลาที่เริ่มวัด ส่วนตำแหน่ง gps จะใช้ตำแหน่งที่กึ่งกลางของช่วงที่วัด เช่น หากกำหนด Acquire Interval เป็น 30 วินาที จะเก็บข้อมูลเวลาที่วินาทีที่ 0 ข้อมูลรังสีที่นับได้ระหว่าง 0-30 วินาที และข้อมูล gps ที่วินาทีที่ 15 เป็นต้น

โหมดการแสดงผลข้อมูล

สามารถเรียกข้อมูลที่จัดเก็บไว้ในฐานข้อมูลมาแสดงผลได้ในรูปแบบต่างๆ บน Google Map ดังนี้

แบบ Line

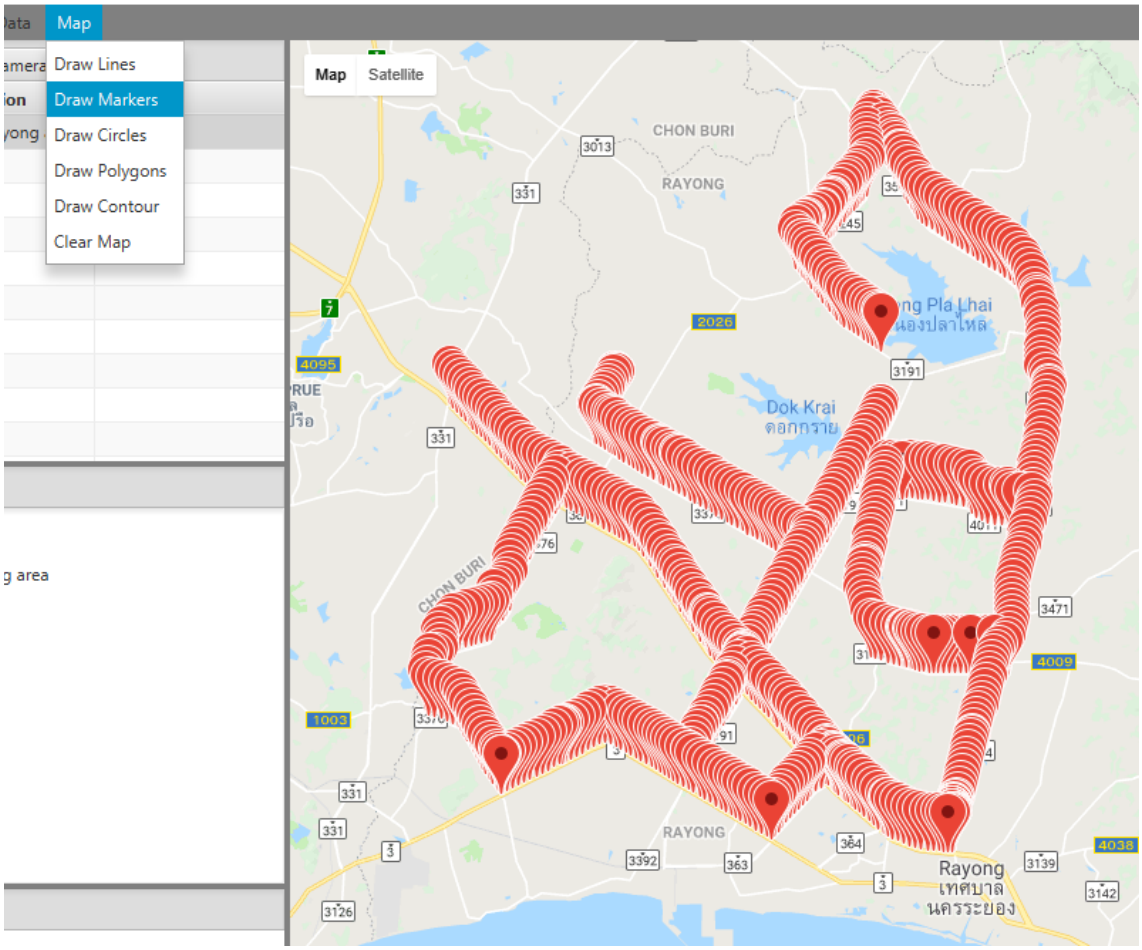
สามารถทำได้โดยเลือกข้อมูลที่ต้องการแสดงในแท็ป Data Sets และเลือกฟังค์ชัน Map > Draw Lines ดังรูปที่ 3.10 โดยการแสดงผลจะใช้ค่าเฉลี่ยอัตราปริมาณรังสีระหว่างจุดวัด 2 จุดที่ติดกันเป็นค่าตัวแทนของอัตราปริมาณรังสีบนเส้นทางระหว่าง 2 จุดนั้น



รูปที่ 3.10 การแสดงผลแบบ Line

แบบ Marker

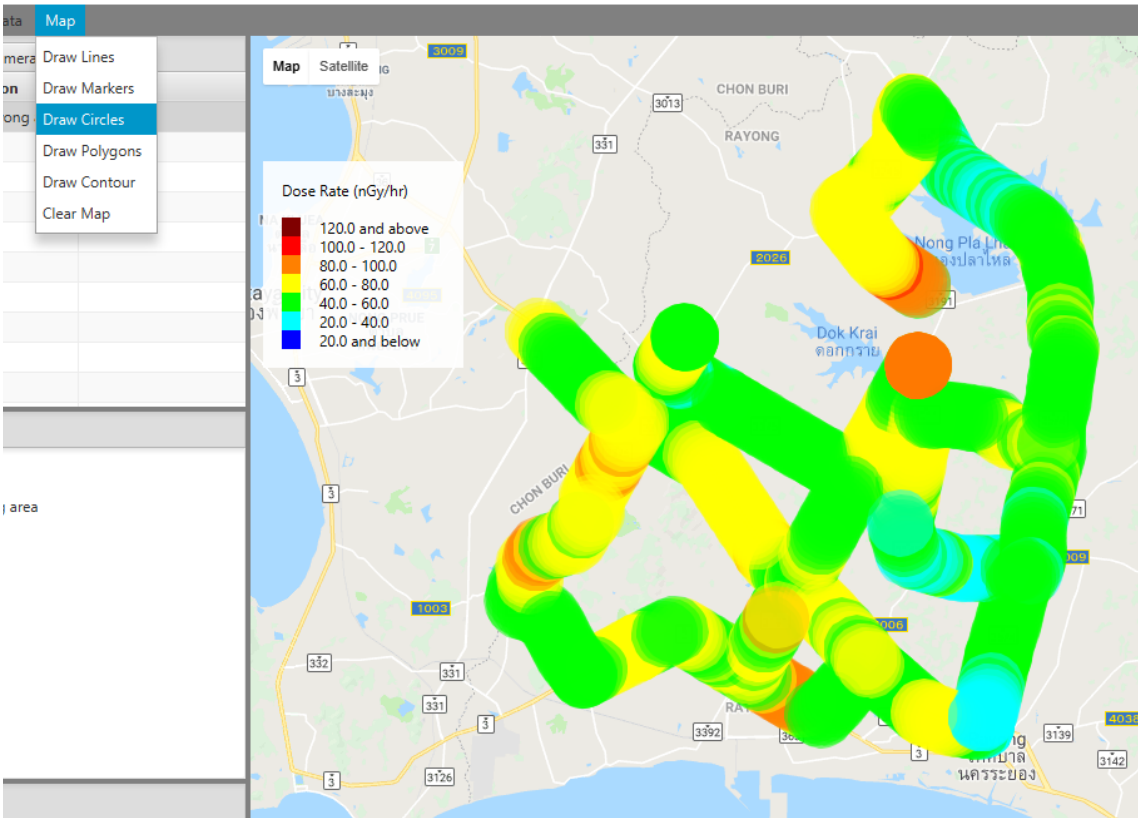
สามารถทำได้โดยเลือกข้อมูลที่ต้องการแสดงในแท็ป Data Sets และเลือกฟังค์ชัน Map > Draw Markers ดังรูปที่ 3.11 โดยเป็นการแสดงตำแหน่งการเก็บข้อมูลเท่านั้น



รูปที่ 3.11 การแสดงผลแบบ Marker

แบบ Circle

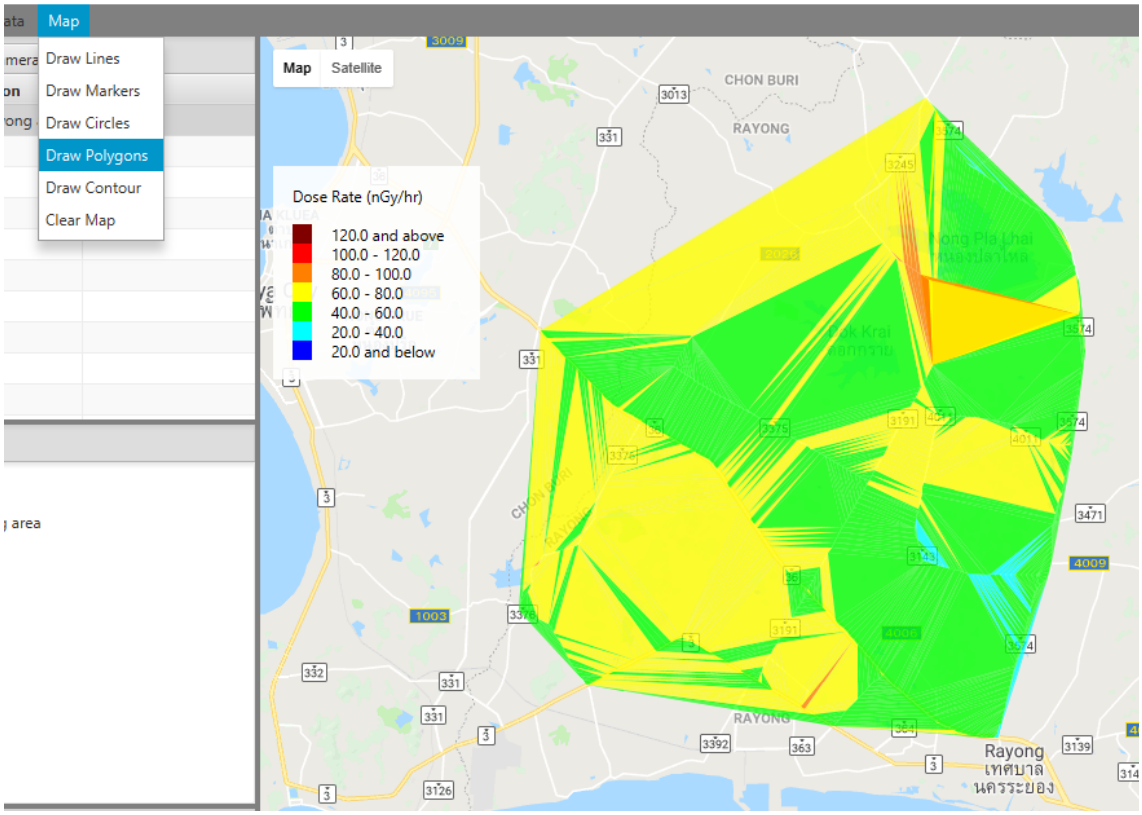
สามารถทำได้โดยเลือกข้อมูลที่ต้องการแสดงในแท็ป Data Sets และเลือกฟังค์ชัน Map > Draw Circles ดังรูปที่ 3.12 โดยเป็นการแสดงอัตราปริมาณรังสี ณ จุดที่วัดโดยตรง



รูปที่ 3.12 การแสดงผลแบบ Circle

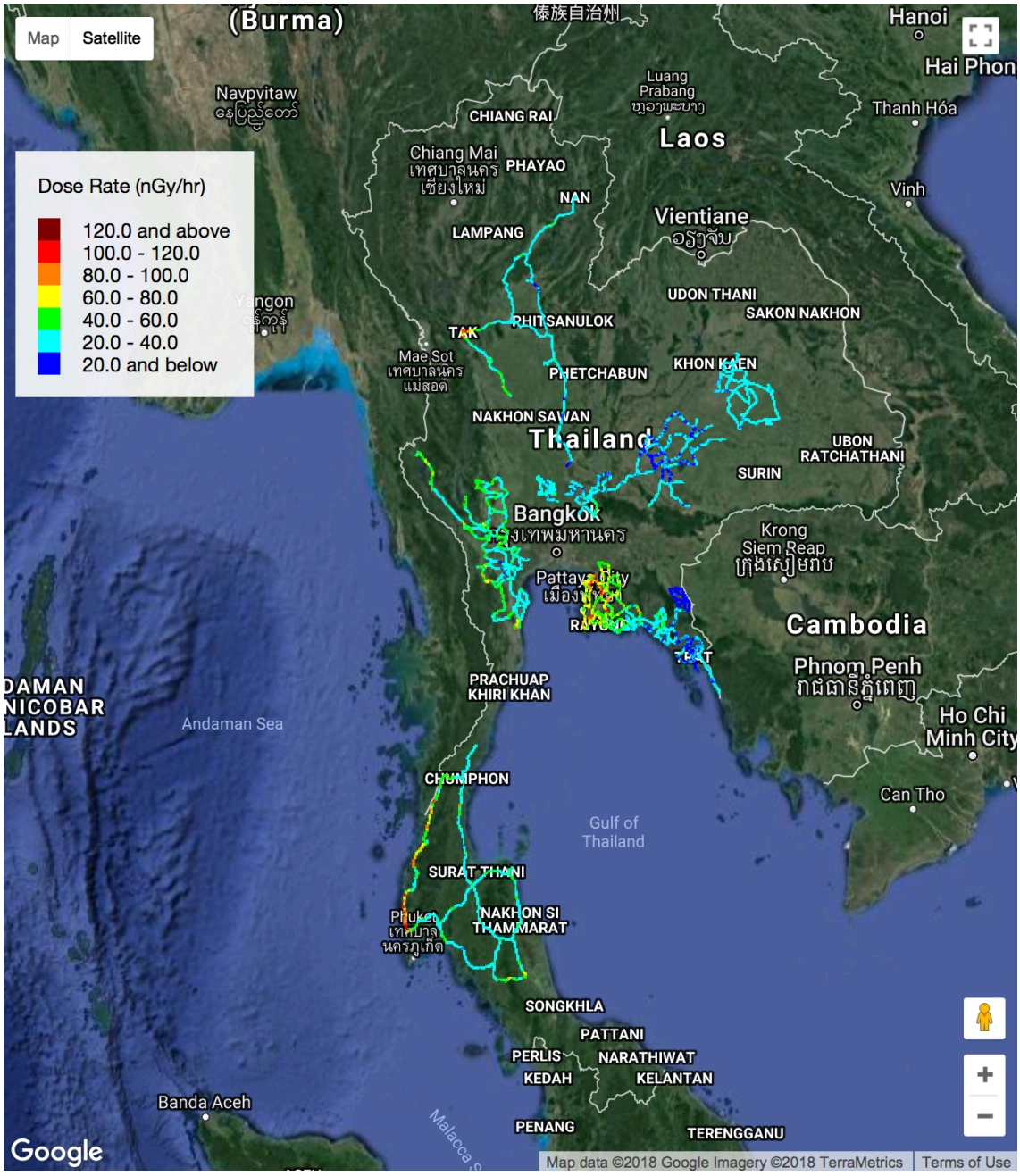
แบบ Polygon

สามารถทำได้โดยเลือกข้อมูลที่ต้องการแสดงในแท็ป Data Sets และเลือกฟังค์ชัน Map > Draw Polygons ดังรูปที่ 3.13 โดยได้ใช้เทคนิค Delaunay Triangulation ในการสร้างสามเหลี่ยมจากจุดวัดเพื่อประกอบเป็น polygon ขึ้น และอาศัยค่าเฉลี่ยของอัตราปริมาณรังสีที่วัดได้ ณ มุมทั้งสามของแต่ละสามเหลี่ยมเป็นค่าตัวแทนของอัตราปริมาณรังสีภายในพื้นที่ของสามเหลี่ยมนั้น ซึ่งวิธีนี้จะยิ่งมีความถูกต้องมากขึ้นหากจุดวัดมีระยะห่างระหว่างกันลดลง



รูปที่ 3.13 การแสดงผลแบบ Polygon

สามารถเรียกข้อมูลหลายชุดมาแสดงผลได้พร้อมกันเพื่อใช้ในการวิเคราะห์ผลในบริเวณที่กว้างขึ้น ดังตัวอย่างในรูปที่ 3.14



รูปที่ 3.14 การแสดงผลที่ประกอบด้วยข้อมูลหลายชุดรวมกัน

3.4 ศึกษาปัจจัยที่มีผลต่อการวัด และวิธีการแก้ค่าต่างๆ

การวัดปริมาณรังสีด้วยระบบติดรถยนต์ มีปัจจัยต่างๆ ที่มีผลต่อการวัด รวมถึง ขนาดของหัววัด ความเร็วของรถยนต์ การกำบังรังสีของรถยนต์ และช่วงระยะเวลาที่ใช้ในการเก็บข้อมูล ซึ่งจะ*ดำเนินการศึกษารายละเอียดในปีที่ 2 ของโครงการ*

<u>ความสัมพันธ์ระหว่างขนาดของหัววัด ความเร็วของรถยนต์ ช่วงระยะเวลาที่ใช้ในการเก็บข้อมูล</u>

ในการวิจัยนี้ ได้ทำการทดสอบระบบโดยใช้หัววัด NaI 3 ขนาด คือ 1 และ 5 นิ้ว (ตาราง 3.1 และ 3.2) และได้ทำการเปรียบเทียบกับระบบลักษณะใกล้เคียงกันที่ออกแบบใช้กับหัววัดขนาด 3 นิ้ว (EMF-221, EMF Japan Co.) ในเบื้องต้นพบว่าหัววัดที่มีขนาดใหญ่กว่าสามารถเก็บสัญญาณ (นับวัด) ได้มากกว่าในช่วงเวลาเก็บเท่าๆ

กัน เนื่องจากมีพื้นที่ผิวในการรับสัญญาณมากกว่า ซึ่งเป็นประโยชน์สำหรับการวัดรังสีในธรรมชาติที่ต้องมีการเคลื่อนที่ไปด้วย เพราะหากต้องการลดค่าความผิดพลาดทางสถิติในการนับวัด จะต้องใช้จำนวนนับวัดเพิ่มขึ้น เช่น ต้องใช้จำนวนนับวัด 10,000 counts เพื่อให้ค่าความผิดพลาดทางสถิติลดลงน้อยกว่า 1% ดังนั้น ในทางปฏิบัติหากต้องการเพิ่มจำนวนนับวัด จะสามารถกระทำได้ 2 วิธี คือ

- ลดความเร็วของรถยนต์ที่ใช้ มีข้อเสียคือ ใช้เวลาเพิ่มขึ้นในการเก็บข้อมูลบริเวณกว้าง
- เพิ่มระยะเวลาในการเก็บข้อมูล มีข้อเสียคือ เพิ่มความห่างระหว่างจุดการเก็บข้อมูล ทำให้ความละเอียดของการสำรวจลดลง

ดังนั้น ในการออกแบบกระบวนการเก็บข้อมูล จึงต้องคำนึงถึงทั้ง 3 ปัจจัยนี้ เพื่อให้ความผิดพลาดทางสถิติในการวัดอยู่ในระดับที่ยอมรับได้

<u>การกำบังรังสีของรถยนต์</u>

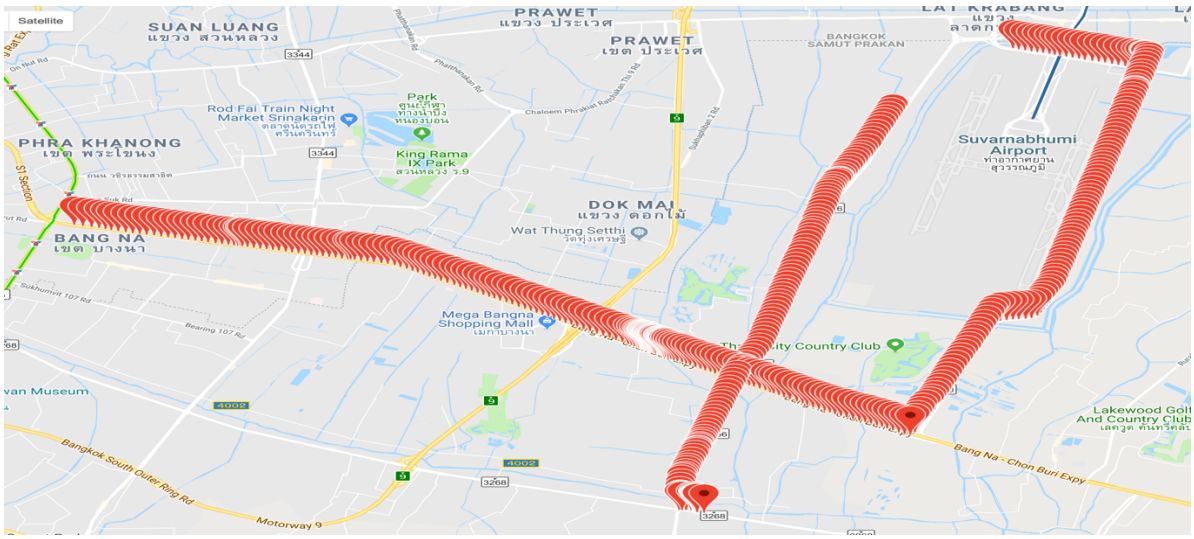
ในการนำระบบไปใช้งานจริงซึ่งจะติดตั้งภายในรถยนต์ จะต้องทำการปรับเทียบก่อนเสมอ เนื่องจากรถยนต์มีความสามารถในการกำบังรังสีปริมาณหนึ่ง ซึ่งการปรับเทียบจะทำโดยการนับวัด 2 จุด คือ

- บริเวณนอกรถที่ความสูง 1 เมตร (ความสูงมาตรฐานที่ใช้ในการสำรวจปริมาณรังสี)
- บริเวณในรถที่จุดติดตั้งระบบ

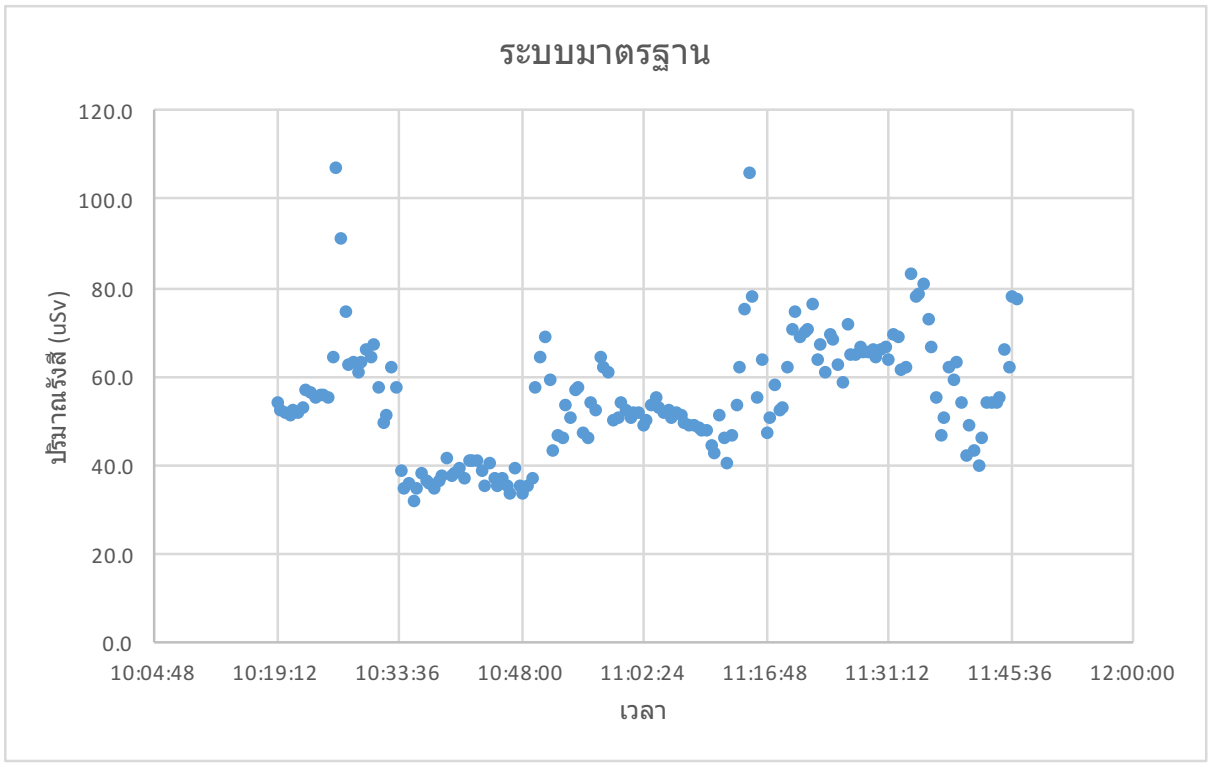
โดยค่าการนับวัดที่ได้จากทั้ง 2 จุดนำไปหา Shielding Factor เพื่อแก้ค่าที่นับวัดได้ในการสำรวจต่อไป

3.5 การทดลองวัดจริงในสิ่งแวดล้อม

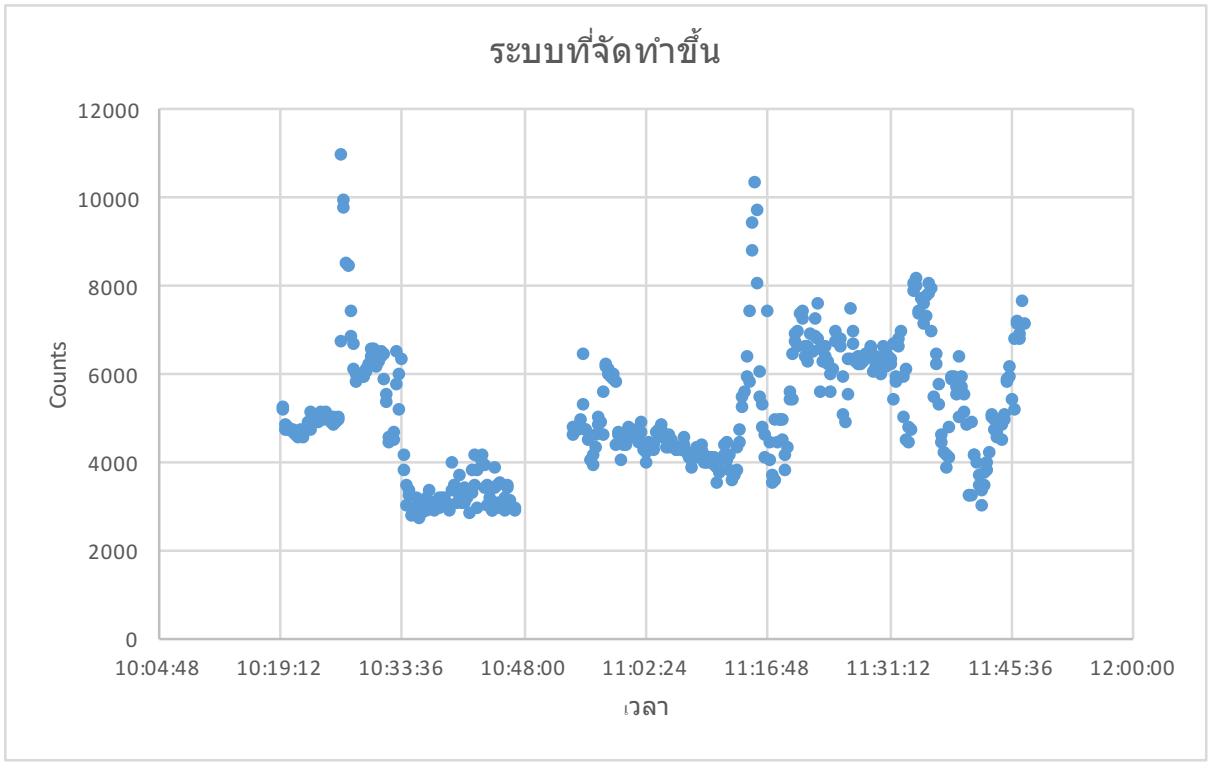
ได้ดำเนินการทดสอบระบบวัดในสิ่งแวดล้อมโดยทำการวัดในบริเวณกรุงเทพและปริมลฑล และทำการเปรียบเทียบข้อมูลที่ได้กับระบบวัดมาตรฐานที่มีลักษณะใกล้เคียงกัน (EMF-221, EMF Japan Co.) พบว่าระบบที่จัดทำขึ้นยังไม่สามารถดำเนินการได้อย่างต่อเนื่องเป็นระยะเวลานาน เนื่องจากการ sync ของเครื่องมือทำให้ internal clock เกิดความผิดพลาดสะสมขึ้นจนเป็นสาเหตุให้โปรแกรมหยุดทำงาน ซึ่งจะดำเนินการแก้ไขต่อไป



รูปที่ 3.15 เส้นทางการทดสอบวัด



(ก)



(ข)

รูปที่ 3.16 เปรียบเทียบผลการวัดปริมาณรังสีระหว่าง (ก) ระบบมาตรฐาน และ (ข) ระบบที่จัดทำขึ้น

อย่างไรก็ดีในช่วงระยะสั้นๆ ที่สามารถเก็บข้อมูลได้ จากการนำข้อมูลตามเส้นทางที่แสดงในรูปที่ 3.15 มาวิเคราะห์เบื้องต้นพบว่า

- ระบบ GPS ที่ใช้ของทั้ง 2 ระบบ ให้ผลไม่แตกต่างกัน และใช้เวลาในการค้นหา GPS ใกล้เคียงกัน
- ผลที่ได้จากการวัดรังสีของทั้งสองระบบเป็นไปในทิศทางกัน ดังแสดงในรูปที่ 3.16(ก) และ (ข) ซึ่งแสดงให้เห็นว่า ระบบที่จัดทำขึ้นมีความถูกต้องเพียงพอสำหรับการสำรวจเบื้องต้น แม้ว่าจะเป็นระบบที่วัดค่า count รวมซึ่งไม่สามารถจำแนกพลังงานตาม channel เหมือนระบบมาตรฐาน และมีข้อได้เปรียบคือ สามารถเก็บข้อมูลได้รวดเร็วกว่า เนื่องจากสามารถเลือกใช้หัววัดที่มีขนาดใหญ่กว่าได้ ซึ่งหากต้องการความถูกต้องทางสถิติที่เท่ากัน จะทำให้สามารถเคลื่อนที่ได้รวดเร็วกว่า และครอบคลุมพื้นที่ได้กว้างกว่าด้วย

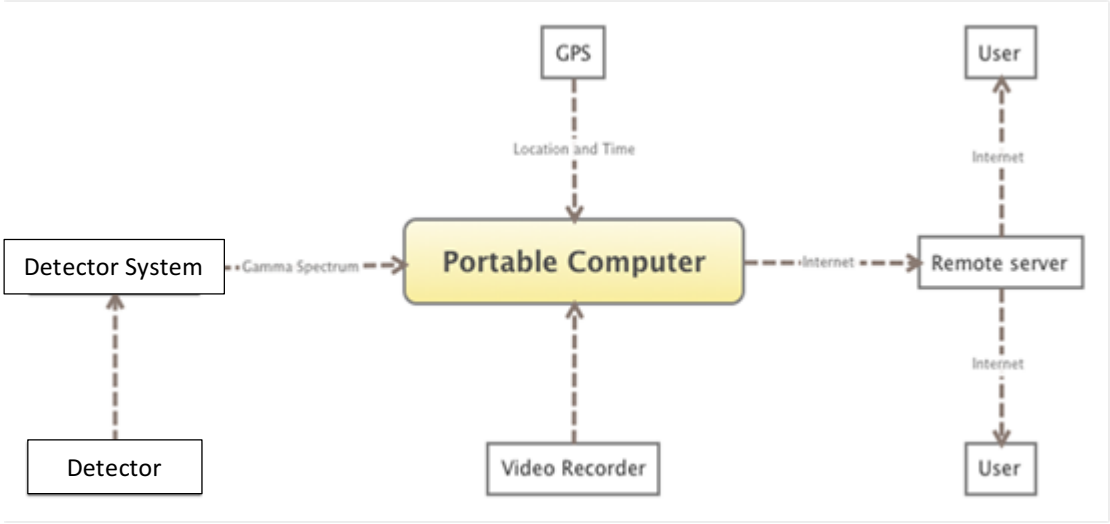
สำหรับในปีที่ 2 ของโครงการ จะดำเนินการแก้ไขปัญหาที่พบ และศึกษารายละเอียดเพิ่มเติมพร้อมแก้ข้อจำกัดของระบบที่มีต่อไป

บทที่ 4

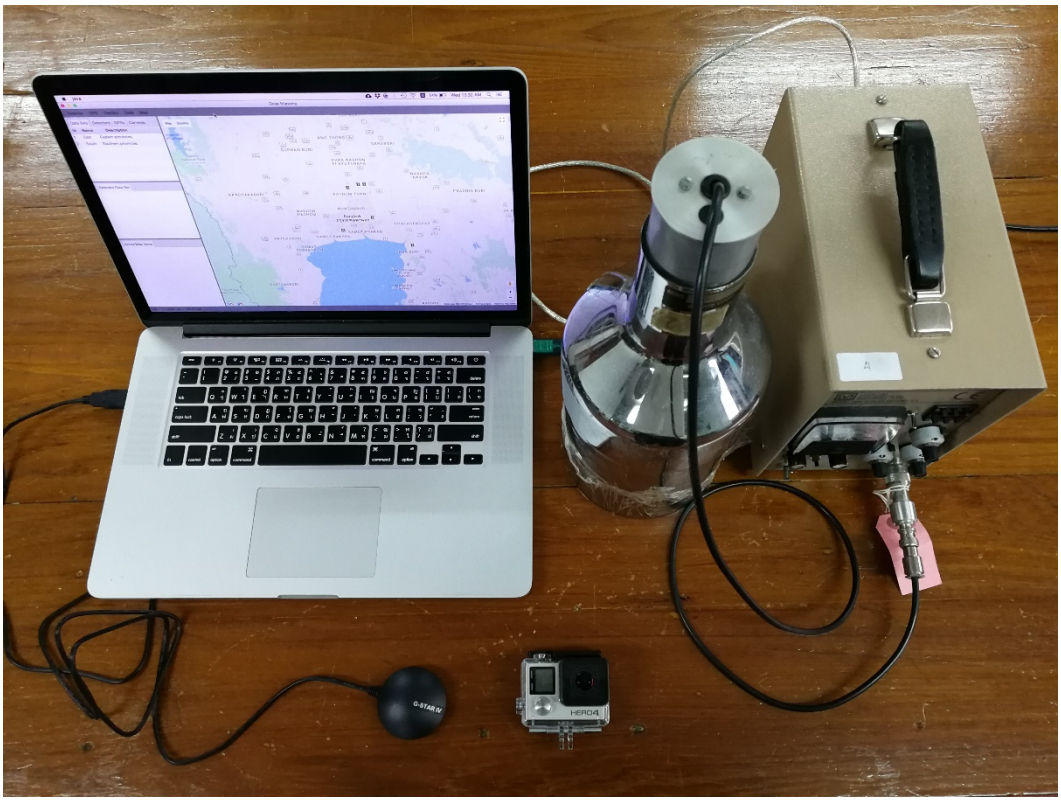
สรุปผลการดำเนินการปีที่ 1 และแผนการดำเนินงานปีที่ 2

4.1 สรุปผลการดำเนินงานปีที่ 1

ในช่วงระยะเวลา 1 ปีของการดำเนินงาน ทีมวิจัยได้พัฒนาระบบสำรวจและจัดการข้อมูลปริมาณรังสีแกมมาแบบติดรถยนต์ขึ้นตามแผน ทั้งในส่วนของการเชื่อมต่อระบบวัด ระบบ GPS และระบบบันทึกภาพ การเก็บข้อมูล ฐานข้อมูล และโปรแกรมคอมพิวเตอร์ที่ใช้เชื่อมต่อทุกส่วนเข้าด้วยกันดังรูปที่ 4.1 และ 4.2



รูปที่ 4.1 การเชื่อมต่อของระบบสำรวจและจัดการข้อมูลปริมาณรังสีแบบติดรถยนต์



รูปที่ 4.2 การเชื่อมต่อของระบบ

การเชื่อมต่อระหว่างระบบวัด ใช้มาตรฐานการเชื่อมต่อข้อมูลแบบ RS-232 ซึ่งรองรับอุปกรณ์สำรวจทั่วไปที่ใช้งานอยู่ในปัจจุบัน ตัวโปรแกรมควบคุมได้รับการออกแบบให้มีความเป็น Object Oriented ซึ่งสามารถเพิ่มเติมอุปกรณ์ได้เมื่อมีความต้องการ และพัฒนาขึ้นด้วยภาษา Java ซึ่งสามารถใช้งานได้บนระบบปฏิบัติการที่หลากหลาย รวมถึง ระบบ Microsoft Windows (ทดสอบด้วยเวอร์ชัน 10) และระบบ Mac OSX (ทดสอบด้วยเวอร์ชัน 10.10.5)

การเก็บข้อมูลจากการอุปกรณ์วัดสามารถทำได้ทั้งในแบบเรียกอ่านข้อมูลจากไฟล์หรือแบบ Real-time ซึ่งข้อมูลจะถูกจัดเก็บไปยังระบบจัดการฐานข้อมูล MySQL ซึ่งสามารถค้นหาและเรียกใช้ข้อมูลผ่านระบบอินเตอร์เน็ตเพื่อการวิเคราะห์ภายหลังได้ จึงเหมาะสำหรับประยุกต์ใช้ในด้านการเตรียมการและตอบสนองต่อเหตุการณ์ฉุกเฉินทางนิวเคลียร์และรังสี

4.2 แผนการดำเนินงานปีที่ 2

ในส่วนของการดำเนินงานปีที่ 2 จะเป็นการดำเนินการที่ต่อเนื่องจากปีที่ 1 ซึ่งจะเน้นการนำระบบไปใช้จริงและการปรับปรุงระบบทั้งในส่วนของ hardware และ software ให้เหมาะสม โดยมีกิจกรรมดังต่อไปนี้

- ศึกษาปัจจัยที่มีผลต่อการวัด และวิธีการแก้ค่าต่างๆ
- ทดลองวัดจริงในสิ่งแวดล้อม
- วิเคราะห์ เปรียบเทียบ แก้ค่า และปรับแก้ระบบและวิธีการ
- ทดลองวัดกับ Orphan Source ในสิ่งแวดล้อม
- ปรับแก้วิธีการตรวจหา Orphan Source ในสิ่งแวดล้อม
- สรุปผลการศึกษาและจัดทำรายงาน

เอกสารอ้างอิง

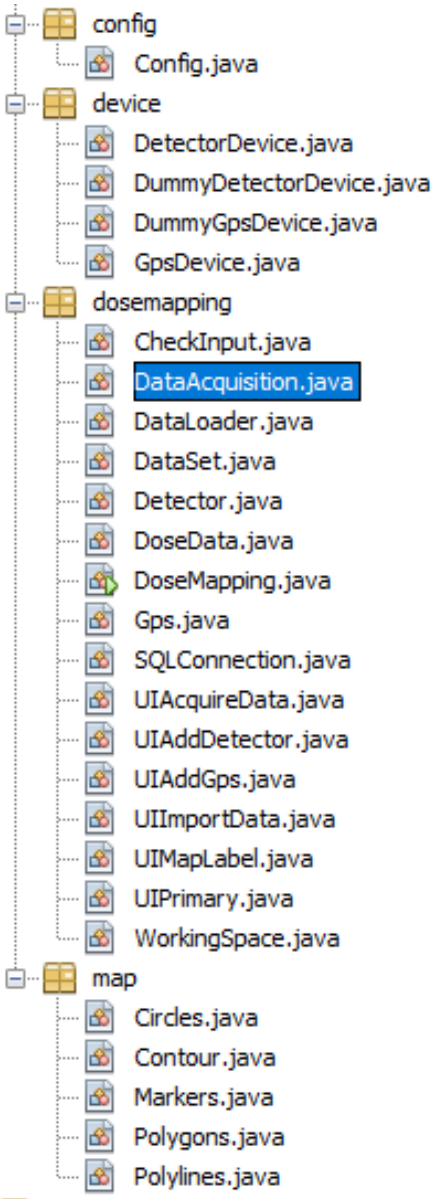
- M. Tanigaki et al., Developmentofacar-borne γ ray surveysystem, KURAMA. Nuclear Instruments and Methods in Physics Research A. 726 (2013), p 162–168.
- S. Sukihara et al., Distribution of gamma-ray dose rate in Fukushima prefecture by a car-borne survey method. Proceedings of the International Symposium on Environmental monitoring and dose estimation of residents after accident of TEPCO's Fukushima Daiichi Nuclear Power Stations. Shiran Hall, Kyoto, Japan, December 14, 2012.
- T. Nagato et al., A car-borne survey of environmental gamma rays in Hokkaido, Japan. Japanese Journal of Health Physics. 32 (1997) 3, pp 295-304.
- M. Furukawa and R. Shingaki, Terrestrial gamma radiation dose rate in Japan estimated before the 2011 Great East Japan Earthquake. Radiation Emergency Medicine. 1 (2012) 1-2, pp 11-16.
- M. Dowdall et al., Car-borne gamma spectrometry: a virtual exercise in emergency response. Journal of Environmental Radioactivity. 107 (2012), pp 68-77.
- J. Tulyatid, Airborne radiometric data interpretation as an aid to granitic terrain mapping: a case study for Hua Hin - Pran Buri area, south central Thailand. National Conference on Geologic Resources of Thailand: Polential for Future Development. Department of Mineral Resources, Bangkok, Thailand, 17-24 November, 1992, p 86.
- S. Chanyotha et. al., Terrestrial gamma radiation in Phuket Island, Thailand. Engineering Journal. 14 (2011) 4.
- International Atomic Energy Agency, Guidelines for Radioelement Mapping Using Gamma Ray Spectrometry Data. TECDOC-1363. July, 2003.
- P. Kock, Comparison of airborne and terrestrial gamma spectrometry measurements - evaluation of three areas in southern Sweden. Journal of Environmental Radioactivity. 102 (2011), pp 605-613.
- K. K. Lai, Terrestrial gamma ray dose rates of Brunei Darussalam. Applied Radiation and Isotopes. 50 (1999), pp 599-608.
- H. K. Aage and U. Korsbech, Search for lost or orphan radioactive sources based on NaI gamma spectrometry. Applied Radiation and Isotopes. 58 (2003) pp 103–113.
- P. Kock, Orphan source detection in mobile gamma-ray spectrometry: Improved techniques for background assessment. Doctoral thesis: Department of Medical Radiation Physics, Lund University, Sweden. 2012.
- P. L. Reeder and D. C. Stromswold, Performance of Large NaI(Tl) Gamma-Ray Detectors Over Temperature -50°C to +60°C. Pacific Northwest Laboratory Report, USA. June 2004.

ภาคผนวก ก

โปรแกรม DoseMapping

โปรแกรมคอมพิวเตอร์ที่ได้พัฒนาขึ้นในงานวิจัยนี้เขียนขึ้นด้วยภาษา Java มีโครงสร้างดังแสดงในรูปที่ ก-1 และเรียกใช้ Library จากภายนอก 3 ตัว คือ

- GMapsFX เวอร์ชัน 2.12.2 สำหรับเชื่อมต่อกับ Google Map พัฒนาขึ้นโดย Rob Terpilowski
- DelaunayTriangulator เวอร์ชัน 1.0.3 (The MIT License (MIT) Copyright 2015 Johannes Diemke)
- jSSC เวอร์ชัน 2.8.0 สำหรับติดต่อกับระบบสำรวจผ่าน Serial Port พัฒนาขึ้นโดย Sokolov Alexey



รูปที่ ก-1

โดยมี Source Code ดังนี้

1. config > Config.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package config;


import java.text.DecimalFormat;


/**
 *
 * @author ppengvan
 */
public class Config {
    // Database
    public static final String CONNECTION = "jdbc:mysql://localhost:3306/carborne?zeroDateTimeBehavior=convertToNull";
    public static final String USERNAME = "root";
    public static final String PASSWORD = "lopburi2430";
////    public static final String CONNECTION =
"jdbc:mysql://sql12.freemysqlhosting.net:3306/sql12220111?zeroDateTimeBehavior=convertToNull";
////    public static final String USERNAME = "sql12220111";
////    public static final String PASSWORD = "yPYfENfbDe";


    // Input Limit
    public static final int MAX_NAME_LENGTH = 30;
    public static final int MAX_DESCRIPTION_LENGTH = 255;


    // Format
    public static final String DECIMAL_FORMAT = "###.00000";


    // Ports
```

```
    public static final String DETECTOR_PORT = "/dev/tty.wchusbserial1410";

    public static final String GPS_PORT = "/dev/tty.usbserial";


    // Map

    public static final double DOSE_RATE_RANGE = 20;


    // Diretory

    public static final String DEFAULT_PATH = "/Users/ppengvan/Dropbox/JavaProj/DoseMapping/data";

}
```

2. device > DetectorDevice.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package device;

import java.nio.charset.StandardCharsets;
import java.sql.Time;
import java.time.LocalTime;
import java.util.logging.Level;
import java.util.logging.Logger;
import jssc.SerialPort;
import jssc.SerialPortException;

/**
 *
 * @author ppengvan
 */
public class DetectorDevice implements Runnable {
    private SerialPort port;
    private Thread self;
    private boolean threadIsRunning = false;
    private boolean isReady = false;
    private boolean isAcquiringData = false;
    private int interval = 0;
    private int count = 0;

    public void start(SerialPort serialPort) {
        self = new Thread(this);
        threadIsRunning = true;
        port = serialPort;
        self.start();
    }
```

```
    public void stop() {
        threadIsRunning = false;
    }

    @Override
    public void run() {
        System.out.println("Start thread: " + self.toString());
        connect();

        while (threadIsRunning) {
            try {
                if (isReady & isAcquiringData) {
                    synchronized(this) {
                        readData();
                        notify();
                    }
                } else {
                    Thread.sleep(1);
                }
            } catch (InterruptedException ex) {
                Logger.getLogger(DetectorDevice.class.getName()).log(Level.SEVERE, null, ex);
            }
        }

        disconnect();

        System.out.println("Stop thread: " + self.toString());
    }

    public void connect() {
        try {
            port.openPort(); //Open serial port
            port.setParams(2400, 8, 1, 0); //Set params.

            isReady = port.isOpened();

//            if (isReady) {
              System.out.println("Connect detector...success");
//            } else {
////                port.closePort();
//              System.out.println("Connect detector...failed (signal)");
//            }
        }
        catch (SerialPortException ex) {
            System.out.println("Connect detector...failed (port)");
        }
    }

    public void disconnect() {
        if (port.isOpened()) {
```

```
            try {
                port.closePort();
                System.out.println("Disconnect detector...success");
            } catch (SerialPortException ex) {
                System.out.println("Disconnect detector...failed (port)");
            }
        }
    }

    public boolean isReady() {
//        isReady = true;  // for testing only
        return isReady;
    }

    public void acquire(int inval) {
        interval = inval;
        isAcquiringData = true;
    }

    public void readData() {
//        Time time;

        try {
            sendCommand("SO\n", "Start counting");
//            System.out.println(Thread.currentThread().getName());
            Thread.sleep(interval);
            sendCommand("SS\n", "Stop counting");
//            time = Time.valueOf(LocalTime.now());
            count = requestData("RS\n");
            System.out.print(LocalTime.now().toString() + "\t");
            System.out.println("Count: " + count);
            isAcquiringData = false;
        } catch (SerialPortException ex) {
            System.out.println(ex);
        } catch (InterruptedException ex) {
            Logger.getLogger(DetectorDevice.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    private void sendCommand(String s, String m) throws SerialPortException {
        byte[] buffer = s.getBytes(StandardCharsets.UTF_8);

        port.writeBytes(buffer);

        System.out.print(LocalTime.now().toString() + "\t");
        System.out.println(m);
    }

    private int requestData(String s) throws SerialPortException {
        byte[] buffer = s.getBytes(StandardCharsets.UTF_8);
        int val;
```

```
        port.writeBytes(buffer);
        byte[] b = port.readBytes(8);

        String v = new String(b, StandardCharsets.UTF_8);

        try {
            val = Integer.parseInt(v.trim());
        } catch (Exception ex) {
            System.out.println("Detector value error: " + v);
            return 0;
        }

        return val;
    }

    public int getCount() {
        return count;
    }
}
```

3. device > DummyDetectorDevice.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package device;

import java.nio.charset.StandardCharsets;
import java.sql.Time;
import java.time.LocalTime;
import java.util.logging.Level;
import java.util.logging.Logger;
import jssc.SerialPort;
import jssc.SerialPortException;

/**
 *
 * @author ppengvan
 */
public class DummyDetectorDevice implements Runnable {
    private SerialPort port;
    private Thread self;
    private boolean threadIsRunning = false;
    private boolean isReady = false;
    private boolean isAcquiringData = false;
    private int interval = 0;
    private int count = 0;
```

```
    public void start(SerialPort serialPort) {
        self = new Thread(this);
        threadIsRunning = true;
        port = serialPort;
        self.start();
    }

    public void stop() {
        threadIsRunning = false;
    }

    @Override
    public void run() {
        System.out.println("Start thread: " + self.toString());
        connect();

        while (threadIsRunning) {
            try {
                if (isReady & isAcquiringData) {
                    synchronized(this) {
                        readData();
                        notify();
                    }
                } else {
                    Thread.sleep(1);
                }
            } catch (InterruptedException ex) {
                Logger.getLogger(DummyDetectorDevice.class.getName()).log(Level.SEVERE, null, ex);
            }
        }

        disconnect();

        System.out.println("Stop thread: " + self.toString());
    }

    public void connect() {
        isReady = true;
        System.out.println("Connect detector...success");
    }

    public void disconnect() {
        System.out.println("Disconnect detector...success");
    }

    public boolean isReady() {
        return isReady;
    }

    public void acquire(int inval) {
        interval = inval;
```

```
            isAcquiringData = true;
        }

        public void readData() {
            try {
                sendCommand("SO\n", "Start counting");
    //              System.out.println(Thread.currentThread().getName());
                Thread.sleep(interval);
                sendCommand("SS\n", "Stop counting");
    //              time = Time.valueOf(LocalTime.now());
                count = requestData("RS\n");
                System.out.print(LocalTime.now().toString() + "\t");
                System.out.println("Count: " + count);
                isAcquiringData = false;
            } catch (InterruptedException ex) {
                Logger.getLogger(DummyDetectorDevice.class.getName()).log(Level.SEVERE, null, ex);
            } catch (SerialPortException ex) {
                Logger.getLogger(DummyDetectorDevice.class.getName()).log(Level.SEVERE, null, ex);
            }
            count = 0;
            System.out.print(LocalTime.now().toString() + "\t");
            System.out.println("Count: " + count);
            isAcquiringData = false;
        }

        private void sendCommand(String s, String m) throws SerialPortException {
            System.out.print(LocalTime.now().toString() + "\t");
            System.out.println(m);
        }

        private int requestData(String s) throws SerialPortException {
            return 0;
        }

        public int getCount() {
            return count;
        }
    }
```

4. device > DummyGpsDevice.java

```
    /*
     * To change this license header, choose License Headers in Project Properties.
     * To change this template file, choose Tools | Templates
     * and open the template in the editor.
     */
    package device;

    import java.nio.charset.StandardCharsets;
    import java.text.DecimalFormat;
    import java.time.LocalTime;
```

```
import java.util.logging.Level;
import java.util.logging.Logger;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import jssc.SerialPort;
import jssc.SerialPortException;

/**
 *
 * @author ppengvan
 */
public class DummyGpsDevice implements Runnable {
    private SerialPort port;
    private Thread self;
    private boolean threadIsRunning = false;
    private boolean isReady = false;
    private boolean isAcquiringData = false;
    private boolean foundGpsSignal = false;
    private int delay = 0;
    private String data = "";
    private double lat = 0;
    private double lon = 0;

    Pattern p = Pattern.compile("(\\d*)(\\d\\d.\\d\\d\\d\\d\\d)$");
    DecimalFormat f = new DecimalFormat("#.######");

    public void start(SerialPort serialPort) {
        self = new Thread(this);
        threadIsRunning = true;
        port = serialPort;
        self.start();
    }

    public void stop() {
        threadIsRunning = false;
    }

    @Override
    public void run() {
        System.out.println("Start thread: " + self.toString());
        connect();
        checkGpsSignal();

        while (threadIsRunning) {
            try {
                if (isReady) {
                    readData();

                    if (isAcquiringData) {
                        Thread.sleep(delay);
                        getData();
```

```
                }
            } else {
                Thread.sleep(1);
            }
        } catch (InterruptedException ex) {
            Logger.getLogger(DummyGpsDevice.class.getName()).log(Level.SEVERE, null, ex);
        } catch (SerialPortException ex) {
            Logger.getLogger(DummyGpsDevice.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    disconnect();
    System.out.println("Stop thread: " + self.toString());
}

public void connect() {
    isReady = true;
    System.out.println("Connect GPS...success");
}

public void disconnect() {
    System.out.println("Disconnect GPS...success");
}

public boolean isReady() {
    isReady = isReady & foundGpsSignal;
    return isReady;
}

public void acquire(int del) {
    delay = del;
    isAcquiringData = true;
}

public void getData() {
    isAcquiringData = false;
}

public void readData() throws SerialPortException {
    try {
        Thread.sleep(1);
    } catch (InterruptedException ex) {
        Logger.getLogger(DummyGpsDevice.class.getName()).log(Level.SEVERE, null, ex);
    }
}

public void checkGpsSignal() {
    if (isReady) {
        foundGpsSignal = true;
    }
}
```

```
    public void calLat(String latVal, String direction) {
        if ((latVal.length() >= 8) & (direction.length() == 1)) {
            System.out.println(latVal);
            Matcher m = p.matcher(latVal);
            if (!m.matches()) {
                System.out.println("No match found");
            } else {
                double dd = Integer.parseInt(m.group(1));
                double mm = Double.parseDouble(m.group(2)) / 60;
                double newLat = Double.valueOf(f.format(dd + mm));
                if (direction == "S") {
                    newLat = newLat * -1;
                }
                System.out.println(newLat);
                lat = newLat;
            }
        }
    }

    public void calLon(String lonVal, String direction) {
        if ((lonVal.length() >= 8) & (direction.length() == 1)) {
            System.out.println(lonVal);
            Matcher m = p.matcher(lonVal);
            if (!m.matches()) {
                System.out.println("No match found");
            } else {
                double dd = Integer.parseInt(m.group(1));
                double mm = Double.parseDouble(m.group(2)) / 60;
                double newLon = Double.valueOf(f.format(dd + mm));
                if (direction == "S") {
                    newLon = newLon * -1;
                }
                System.out.println(newLon);
                lon = newLon;
            }
        }
    }

    public double getLat() {
        return lat;
    }

    public double getLon() {
        return lon;
    }
}
```

5. device > GpsDevice.java

```
/*
```

```
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package device;

import java.nio.charset.StandardCharsets;
import java.text.DecimalFormat;
import java.time.LocalTime;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import jssc.SerialPort;
import jssc.SerialPortException;

/**
 *
 * @author ppengvan
 */
public class GpsDevice implements Runnable {
    private SerialPort port;
    private Thread self;
    private boolean threadIsRunning = false;
    private boolean isReady = false;
    private boolean isAcquiringData = false;
    private boolean foundGpsSignal = false;
    private int delay = 0;
    private String data = "";
    private double lat = 0;
    private double lon = 0;

    Pattern p = Pattern.compile("(\\d*)(\\d\\d.\\d\\d\\d\\d\\d)$");
    DecimalFormat f = new DecimalFormat("#.######");

    public void start(SerialPort serialPort) {
        self = new Thread(this);
        threadIsRunning = true;
        port = serialPort;
        self.start();
    }

    public void stop() {
        threadIsRunning = false;
    }

    @Override
    public void run() {
        System.out.println("Start thread: " + self.toString());
        connect();
        checkGpsSignal();
```

```
        while (threadIsRunning) {
            try {
                if (isReady) {
                    readData();

                    if (isAcquiringData) {
                        Thread.sleep(delay);
                        getData();
                    }
                } else {
                    Thread.sleep(1);
                }
            } catch (InterruptedException ex) {
                Logger.getLogger(GpsDevice.class.getName()).log(Level.SEVERE, null, ex);
            } catch (SerialPortException ex) {
                Logger.getLogger(GpsDevice.class.getName()).log(Level.SEVERE, null, ex);
            }
        }

        disconnect();
        System.out.println("Stop thread: " + self.toString());
    }

    public void connect() {
        try {
            port.openPort(); //Open serial port
            port.setParams(4800, 8, 1, 0); //Set params.

            isReady = port.isOpened();

//            System.out.println(isReady);

//            if (isReady) {
              System.out.println("Connect GPS...success");
//            } else {
//                port.closePort();
//                System.out.println("Connect GPS...failed (signal)");
//            }
        }
        catch (SerialPortException ex) {
            System.out.println("Connect GPS...failed (port)");
        }
    }

    public void disconnect() {
        if (port.isOpened()) {
            try {
                port.closePort();
                System.out.println("Disconnect GPS...success");
            } catch (SerialPortException ex) {
```

```
                System.out.println("Disconnect GPS...failed (port)");
            }
        }
    }

    public boolean isReady() {
//        checkGpsSignal();
        isReady = isReady & foundGpsSignal;

//        isReady = true;  // for testing only
        return isReady;
    }

    public void acquire(int del) {
        delay = del;
        isAcquiringData = true;
    }

    public void getData() {
        boolean found = false;

        while (!found) {
            try {
                if (data.startsWith("$GPGGA")) {
                    found = true;
                } else {
                    readData();
                }
            } catch (SerialPortException ex) {
                Logger.getLogger(GpsDevice.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
        String[] terms = data.split(",");
        System.out.print(LocalTime.now().toString() + "\t");
        System.out.println(data.trim());
        calLat(terms[2],terms[3]);
        calLon(terms[4],terms[5]);
        isAcquiringData = false;
    }

    public void readData() throws SerialPortException {
        boolean foundEOL = false;
        String line = "";

        while (!foundEOL) {
            byte[] b = port.readBytes(1);

            if (b[0] == '\n') {
                foundEOL = true;
            }
```

```
            line = line + (new String(b, StandardCharsets.UTF_8));
        }

        data = line;
    }

    public void checkGpsSignal() {
        boolean found = false;

        if (isReady) {
            while (!found) {
                try {
                    readData();
                    if (data.contains("$GPGGA")) {
                        found = true;
                    }
                } catch (SerialPortException ex) {
                    Logger.getLogger(GpsDevice.class.getName()).log(Level.SEVERE, null, ex);
                }
            }
            String[] terms = data.split(",");
            if (terms[6].equals("0")) {
                foundGpsSignal = false; // actual code
////                foundGpsSignal = true;  // for testing purpose
            } else {
                foundGpsSignal = true;
            }
        }
    }

    public void calLat(String latVal, String direction) {
        if ((latVal.length() >= 8) & (direction.length() == 1)) {
            System.out.println(latVal);
            Matcher m = p.matcher(latVal);
            if (!m.matches()) {
                System.out.println("No match found");
            } else {
                double dd = Integer.parseInt(m.group(1));
                double mm = Double.parseDouble(m.group(2)) / 60;
                double newLat = Double.valueOf(f.format(dd + mm));
                if (direction == "S") {
                    newLat = newLat * -1;
                }
                System.out.println(newLat);
                lat = newLat;
            }
        }
    }

    public void calLon(String lonVal, String direction) {
        if ((lonVal.length() >= 8) & (direction.length() == 1)) {
```

```
            System.out.println(lonVal);
            Matcher m = p.matcher(lonVal);
            if (!m.matches()) {
                System.out.println("No match found");
            } else {
                double dd = Integer.parseInt(m.group(1));
                double mm = Double.parseDouble(m.group(2)) / 60;
                double newLon = Double.valueOf(f.format(dd + mm));
                if (direction == "S") {
                    newLon = newLon * -1;
                }
                System.out.println(newLon);
                lon = newLon;
            }
        }
    }

    public double getLat() {
        return lat;
    }

    public double getLon() {
        return lon;
    }
}
```

6. dosemapping > CheckInput.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package dosemapping;

import config.Config;
import java.time.LocalDate;

/**
 *
 * @author Administrator
 */
public class CheckInput {
    public static boolean isValidName(String txt) {
        if (txt.length() == 0) {
            return false;
        }
        else if (txt.length() > Config.MAX_NAME_LENGTH) {
            return false;
        }
        else if (txt.matches("^\\S.*")) {
```

```
            return true;
        }
        else {
            return false;
        }
    }

    public static boolean isValidDescription(String txt) {
        if (txt.length() == 0) {
            return false;
        }
        else if (txt.length() > Config.MAX_DESCRIPTION_LENGTH) {
            return false;
        }
        else if (txt.matches("^\\S.*")) {
            return true;
        }
        else {
            return false;
        }
    }

    public static boolean isValidDate(LocalDate date) {
        if (date != null) {
            return true;
        }
        else {
            return false;
        }
    }
}
```

7. dosemapping > DataAcquisition.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package dosemapping;

import config.Config;
import device.DetectorDevice;
import device.DummyDetectorDevice;
import device.DummyGpsDevice;
import device.GpsDevice;
import java.io.BufferedWriter;
import java.io.IOException;
import java.nio.charset.Charset;
import java.nio.file.Files;
import java.nio.file.Path;
```

```
import java.nio.file.Paths;
import java.nio.file.StandardOpenOption;
import java.sql.Date;
import java.sql.Time;
import java.time.LocalDate;
import java.time.LocalTime;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import jssc.SerialPort;

/**
 *
 * @author Administrator
 */
public class DataAcquisition implements Runnable {
    private Thread self;
    private boolean threadIsRunning = false;
    private boolean isReady = false;
    private boolean isAcquiringData = false;
    private int interval;
    private List<DoseData> doseDatas = new ArrayList<>();
    private UIAcquireData ui;

    // Device parameter
    DetectorDevice detector = new DetectorDevice();
    GpsDevice gps = new GpsDevice();
    private boolean detectorIsReady = false;
    private boolean gpsIsReady = false;

    public void start(UIAcquireData uiVal) {
        self = new Thread(this);
        ui = uiVal;
        threadIsRunning = true;
        self.start();
    }

    public void stop() {
        threadIsRunning = false;
    }

    @Override
    public void run() {
        Time t;
        Date d;

        System.out.println("Start thread: " + self.toString());

        detector.start(new SerialPort((Config.DETECTOR_PORT)));
        gps.start(new SerialPort(Config.GPS_PORT));
```

```
        try {
            System.out.println(Thread.currentThread().getName());
            Thread.sleep(1000);
            checkDeviceReadiness();
        } catch (InterruptedException ex) {
            Logger.getLogger(DataAcquisition.class.getName()).log(Level.SEVERE, null, ex);
        }

        while (threadIsRunning) {
            try {
                if (isReady & isAcquiringData) {
                    synchronized(detector) {
                        t = Time.valueOf(LocalTime.now());
                        d = Date.valueOf(LocalDate.now());
                        detector.acquire(interval);
                        gps.acquire(interval/2);
                        detector.wait();
                    }
                    doseDatas.add(new DoseData(d, t, gps.getLat(), gps.getLon(), detector.getCount()));
                    ui.taData.appendText("Date: " + d + ", Time: " + t + ", Lat: " + gps.getLat() + ", Lon: " + gps.getLon() + ",
Dose: " + detector.getCount() + "\n");
                    saveData(d + "\t" + t + "\t" + gps.getLat() + "\t" + gps.getLon() + "\t" + detector.getCount() + "\n");
//////                    ui.taData.appendText("Date: " + d + ", Time: " + t + ", Dose: " + detector.getCount() + "\n");
//////                    saveData(d + "\t" + t + "\t" + detector.getCount() + "\n");
                } else {
                    Thread.sleep(1);
                }
            } catch (InterruptedException ex) {
                Logger.getLogger(DataAcquisition.class.getName()).log(Level.SEVERE, null, ex);
            }
        }

        detector.stop();
        gps.stop();
        System.out.println("Stop thread: " + self.toString());
    }

    public void startAcquiring() {
        if (threadIsRunning) {
            isAcquiringData = true;
        } else {
            isAcquiringData = false;
        }
    }

    public void pauseAcquiring() {
        if (threadIsRunning) {
            isAcquiringData = false;
            System.out.println("Pause acquiring");
        } else {
```

```
            isAcquiringData = false;
        }
    }

    public boolean isReady() {
        return isReady;
    }

    public boolean detectorIsReady() {
        return detectorIsReady;
    }

    public boolean gpsIsReady() {
        return gpsIsReady;
    }

    public boolean isAcquiringData() {
        return isAcquiringData;
    }

    public List<DoseData> getDoseDatas() {
        return doseDatas;
    }

    public void setInterval(int val) {
        interval = val;
    }

    public void checkDeviceReadiness() {
       detectorIsReady = detector.isReady();
       gpsIsReady = gps.isReady();

       isReady = detectorIsReady & gpsIsReady;
//////       isReady = detector.isReady();
    }

    public void saveData(String s) {
        Path floc = Paths.get(Config.DEFAULT_PATH + "/acquiredData.txt");
        Charset charset = Charset.forName("US-ASCII");
        try {
            if (!Files.exists(floc)) {
                Files.createFile(floc);
            }
            BufferedWriter out = Files.newBufferedWriter(floc, charset, StandardOpenOption.APPEND);
            System.out.print("Writing to file: " + s);
            out.write(s);
            out.close();
        } catch (IOException ex) {
            Logger.getLogger(DataAcquisition.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
```

```
}
```

8. dosemapping > DataLoader.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package dosemapping;

import java.io.BufferedReader;
import java.io.File;
import java.nio.charset.Charset;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.sql.Date;
import java.sql.Time;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
import java.util.regex.MatchResult;

/**
 *
 * @author ppengvan
 */
public class DataLoader {
//    public List<DoseData> getDataFromGPX(File file) throws Exception {
//        List<DoseData> doseData = new ArrayList<>();
//        GPSCtrl gps = new GPSCtrl();
//        gps.gpxParser(file.getPath());
//
//        for (int i = 0; i < gps.gData.size(); i++) {
//            DoseData data = new DoseData();
//            data.setLat(gps.gData.get(i).getLat());
//            data.setLon(gps.gData.get(i).getLon());
//            data.setDose(gps.gData.get(i).getEle()); // Temperary use due to no dose data
//            doseData.add(data);
//        }
//
//        return doseData;
//    }

    public void getDataFromTXT(List<DoseData> doseData, File file) throws Exception {
        Path floc = Paths.get(file.getPath());
        Charset charset = Charset.forName("US-ASCII");
        int year;
        int month;
        int day;
```

```
            int hour;
            int minute;
            int second;
            int i = 0;

            System.out.println("Openning data file.");

            try (BufferedReader in = Files.newBufferedReader(floc, charset)) {
                System.out.println("Importing data from file.");
                Scanner sc = new Scanner(in);
                while (sc.hasNext()) {
                    DoseData data = new DoseData();

                    if(sc.hasNext("(\\d+)/(\\d+)/(\\d+)")) {
                        sc.next("(\\d+)/(\\d+)/(\\d+)");
                    } else if(sc.hasNext("(\\d+)-(\\d+)-(\\d+)")) {
                        sc.next("(\\d+)-(\\d+)-(\\d+)");
                    }
                    MatchResult date = sc.match();
                    month = Integer.parseInt(date.group(1));
                    day = Integer.parseInt(date.group(2));
                    year = Integer.parseInt(date.group(3));

                    sc.next("(\\d+):(\\d+):(\\d+)");
                    MatchResult time = sc.match();
                    hour = Integer.parseInt(time.group(1));
                    minute = Integer.parseInt(time.group(2));
                    second = Integer.parseInt(time.group(3));

                    data.setDate(new Date(year, month, day));
                    data.setTime(new Time(hour, minute, second));
                    data.setLat(sc.nextDouble());
                    data.setLon(sc.nextDouble());
                    data.setDose(sc.nextDouble());
                    doseData.add(data);
                    i++;
                    System.out.println("Importing " + i + " data points.");
                }
            }

            System.out.println("Done importing data from file.");
        }
    }
```

9. dosemapping > DataSet.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
```

```
package dosemapping;

import javafx.beans.property.SimpleStringProperty;

/**
 *
 * @author Administrator
 */
public class DataSet {
    private final SimpleStringProperty setId;
    private final SimpleStringProperty setName;
    private final SimpleStringProperty setDescription;
    private final SimpleStringProperty dateUpload;
    private final SimpleStringProperty personAcquire;
    private final SimpleStringProperty personUpload;
    private final SimpleStringProperty detectorId;
    private final SimpleStringProperty gpsId;

    public DataSet(String idVal, String setNameVal, String setDescriptionVal, String personAcquiredVal, String
dateUploadedVal, String personUploadedVal, String detectorIdVal, String gpsIdVal) {
        this.setId = new SimpleStringProperty(idVal);
        this.setName = new SimpleStringProperty(setNameVal);
        this.setDescription = new SimpleStringProperty(setDescriptionVal);
        this.personAcquire = new SimpleStringProperty(personAcquiredVal);
        this.dateUpload = new SimpleStringProperty(dateUploadedVal);
        this.personUpload = new SimpleStringProperty(personUploadedVal);
        this.detectorId = new SimpleStringProperty(detectorIdVal);
        this.gpsId = new SimpleStringProperty(gpsIdVal);
    }

    public String getSetId() {
        return setId.get();
    }

    public String getSetName() {
        return setName.get();
    }

    public String getSetDescription() {
        return setDescription.get();
    }

    public String getPersonAcquire() {
        return personAcquire.get();
    }

    public String getDateUpload() {
        return dateUpload.get();
    }

    public String getPersonUpload() {
```

```
        return personUpload.get();
    }

    public String getDetectorId() {
        return detectorId.get();
    }

    public String getGpsId() {
        return gpsId.get();
    }
}
```

10. dosemapping > Detector.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package dosemapping;

import javafx.beans.property.SimpleStringProperty;

/**
 *
 * @author Administrator
 */
public class Detector {
    private final SimpleStringProperty detectorId;
    private final SimpleStringProperty detectorName;
    private final SimpleStringProperty detectorDescription;
    private final SimpleStringProperty detectorManufacturer;

    public Detector(String idVal, String nameVal, String descriptionVal, String manufacturerVal) {
        this.detectorId = new SimpleStringProperty(idVal);
        this.detectorName = new SimpleStringProperty(nameVal);
        this.detectorDescription = new SimpleStringProperty(descriptionVal);
        this.detectorManufacturer = new SimpleStringProperty(manufacturerVal);
    }

    public String getDetectorId() {
        return detectorId.get();
    }

    public String getDetectorName() {
        return detectorName.get();
    }

    public String getDetectorDescription() {
        return detectorDescription.get();
    }
```

```
    public String getDetectorManufacturer() {
        return detectorManufacturer.get();
    }
}
```

11. dosemapping > DoseData.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package dosemapping;

import java.sql.Date;
import java.sql.Time;

/**
 *
 * @author ppengvan
 */
public class DoseData {
    private Date date;
    private Time time;
    private double lat;
    private double lon;
    private double dose;

    public DoseData() {}

    public DoseData(Date dval, Time tval, double latval, double lonval, double doseval) {
        date = dval;
        time = tval;
        lat = latval;
        lon = lonval;
        dose = doseval;
    }

    public void setDate(Date val){
        date = val;
    }

    public void setTime(Time val){
        time = val;
    }
    public void setLat(double val){
        lat = val;
    }
    public void setLon(double val){
        lon = val;
```

```
    }
    public void setDose(double val){
        dose = val;
    }

    public Date getDate(){
        return date;
    }
    public Time getTime(){
        return time;
    }
    public double getLat(){
        return lat;
    }
    public double getLon(){
        return lon;
    }
    public double getDose(){
        return dose;
    }
}
```

12. dosemapping > DoseMapping.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package dosemapping;

import com.lynden.gmapsfx.MapComponentInitializedListener;
import com.lynden.gmapsfx.javascript.event.GMapMouseEvent;
import com.lynden.gmapsfx.javascript.event.UIEventType;
import com.lynden.gmapsfx.javascript.object.LatLong;
import com.lynden.gmapsfx.javascript.object.LatLongBounds;
import com.lynden.gmapsfx.javascript.object.MVCArray;
import com.lynden.gmapsfx.javascript.object.MapOptions;
import com.lynden.gmapsfx.javascript.object.MapTypeIdEnum;
import com.lynden.gmapsfx.javascript.object.Marker;
import com.lynden.gmapsfx.javascript.object.MarkerOptions;
import com.lynden.gmapsfx.shapes.Circle;
import com.lynden.gmapsfx.shapes.CircleOptions;
import com.lynden.gmapsfx.shapes.Polygon;
import com.lynden.gmapsfx.shapes.PolygonOptions;
import com.lynden.gmapsfx.shapes.Polyline;
import com.lynden.gmapsfx.shapes.PolylineOptions;
import config.Config;
import io.github.jdiemke.triangulation.DelaunayTriangulator;
import io.github.jdiemke.triangulation.Edge2D;
import io.github.jdiemke.triangulation.NotEnoughPointsException;
```

```
import io.github.jdiemke.triangulation.Triangle2D;
import io.github.jdiemke.triangulation.Vector2D;
import java.io.File;
import java.sql.Date;
import java.text.DecimalFormat;
import java.time.LocalDate;
import java.util.ArrayList;
import java.util.List;
import java.util.Optional;
import java.util.logging.Level;
import java.util.logging.Logger;
import javafx.application.Application;
import javafx.beans.value.ChangeListener;
import javafx.beans.value.ObservableValue;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.geometry.Rectangle2D;
import javafx.scene.Node;
import javafx.scene.Scene;
import javafx.scene.control.Alert;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.control.ButtonType;
import javafx.scene.control.CheckBox;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.stage.FileChooser;
import javafx.stage.Modality;
import javafx.stage.Screen;
import javafx.stage.Stage;
import javafx.stage.WindowEvent;
import map.Circles;
import map.Contour;
import map.Markers;
import map.Polygons;
import map.Polylines;

/**
 *
 * @author ppengvan
 */
public class DoseMapping extends Application implements MapComponentInitializedListener {

    private UIPrimary ui = new UIPrimary();
    private WorkingSpace ws = new WorkingSpace();
    private SQLConnection connection = new SQLConnection();
    private DataLoader dataLoader = new DataLoader();

    private DecimalFormat formatter = new DecimalFormat(Config.DECIMAL_FORMAT);

    @Override
    public void start(Stage primaryStage) throws Exception {
```

```
        // Set UI behavior
            ui.miAddDetector.setOnAction(new EventHandler<ActionEvent>() {
                @Override
                public void handle(ActionEvent event) {
                    try {
                        Stage addDetectorStage = new Stage();
                        setAddDetectorStage(addDetectorStage);
                        addDetectorStage.initModality(Modality.WINDOW_MODAL);
                        addDetectorStage.initOwner(primaryStage);
                        addDetectorStage.showAndWait();
                    } catch (Exception ex) {
                        Logger.getLogger(DoseMapping.class.getName()).log(Level.SEVERE, null, ex);
                    }
                }
            });

            ui.miDeleteDetector.setOnAction(new EventHandler<ActionEvent>() {
                @Override
                public void handle(ActionEvent event) {
                    try {
                        if (!ui.tvDetector.getSelectionModel().isEmpty()) {
                            Alert alert = new Alert(AlertType.CONFIRMATION);
                            alert.setTitle("Confirmation Dialog");
                            alert.setHeaderText("Deletion Confirmation");
                            alert.setContentText("Are you sure that you want to delete this detector?");

                            Optional<ButtonType> result = alert.showAndWait();
                            if (result.get() == ButtonType.OK){
                                connection.deleteDetector(ws.detectorList, ui.tvDetector.getSelectionModel().getSelectedItem());
                            }
                        }
                    } catch (Exception ex) {
                        Logger.getLogger(DoseMapping.class.getName()).log(Level.SEVERE, null, ex);
                    }
                }
            });

            ui.miAddGps.setOnAction(new EventHandler<ActionEvent>() {
                @Override
                public void handle(ActionEvent event) {
                    try {
                        Stage addGpsStage = new Stage();
                        setAddGpsStage(addGpsStage);
                        addGpsStage.initModality(Modality.WINDOW_MODAL);
                        addGpsStage.initOwner(primaryStage);
                        addGpsStage.showAndWait();
                    } catch (Exception ex) {
                        Logger.getLogger(DoseMapping.class.getName()).log(Level.SEVERE, null, ex);
                    }
                }
            });
```

```
ui.miDeleteGps.setOnAction(new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent event) {
        try {
            if (!ui.tvGps.getSelectionModel().isEmpty()) {
                Alert alert = new Alert(AlertType.CONFIRMATION);
                alert.setTitle("Confirmation Dialog");
                alert.setHeaderText("Deletion Confirmation");
                alert.setContentText("Are you sure that you want to delete this gps?");

                Optional<ButtonType> result = alert.showAndWait();
                if (result.get() == ButtonType.OK){
                    connection.deleteGps(ws.gpsList, ui.tvGps.getSelectionModel().getSelectedItem());
                }
            }
        } catch (Exception ex) {
            Logger.getLogger(DoseMapping.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
});

ui.miImportData.setOnAction(new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent event) {
        try {
            Stage addDataSetStage = new Stage();
            setAddDataStage(addDataSetStage);
            addDataSetStage.initModality(Modality.WINDOW_MODAL);
            addDataSetStage.initOwner(primaryStage);
            addDataSetStage.showAndWait();
        } catch (Exception ex) {
            Logger.getLogger(DoseMapping.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
});

ui.miDeleteData.setOnAction(new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent event) {
        try {
            if (!ui.tvDataSet.getSelectionModel().isEmpty()) {
                Alert alert = new Alert(AlertType.CONFIRMATION);
                alert.setTitle("Confirmation Dialog");
                alert.setHeaderText("Deletion Confirmation");
                alert.setContentText("Are you sure that you want to delete this data set?");

                Optional<ButtonType> result = alert.showAndWait();
                if (result.get() == ButtonType.OK){
                    connection.deleteDataSet(ws.dataSetList, ui.tvDataSet.getSelectionModel().getSelectedItem());
                }
```

```
                }
            } catch (Exception ex) {
                Logger.getLogger(DoseMapping.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
    });

    ui.miAcquireData.setOnAction(new EventHandler<ActionEvent>() {
        @Override
        public void handle(ActionEvent event) {
            try {
                Stage acquireDataSetStage = new Stage();
                setAcquireDataStage(acquireDataSetStage);
                acquireDataSetStage.initModality(Modality.WINDOW_MODAL);
                acquireDataSetStage.initOwner(primaryStage);
                acquireDataSetStage.showAndWait();
            } catch (Exception ex) {
                Logger.getLogger(DoseMapping.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
    });

    ui.miDrawLines.setOnAction(new EventHandler<ActionEvent>() {
        @Override
        public void handle(ActionEvent event) {
            try {
                if (!ui.tvDataSet.getSelectionModel().isEmpty()) {
                    drawLines();
                }
            } catch (Exception ex) {
                Logger.getLogger(DoseMapping.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
    });

    ui.miDrawMarkers.setOnAction(new EventHandler<ActionEvent>() {
        @Override
        public void handle(ActionEvent event) {
            try {
                if (!ui.tvDataSet.getSelectionModel().isEmpty()) {
                    drawMarkers();
                }
            } catch (Exception ex) {
                Logger.getLogger(DoseMapping.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
    });

    ui.miDrawCircles.setOnAction(new EventHandler<ActionEvent>() {
        @Override
        public void handle(ActionEvent event) {
```

```
            try {
                if (!ui.tvDataSet.getSelectionModel().isEmpty()) {
                    drawCircles();
                }
            } catch (Exception ex) {
                Logger.getLogger(DoseMapping.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
    });

    ui.miDrawPolygons.setOnAction(new EventHandler<ActionEvent>() {
        @Override
        public void handle(ActionEvent event) {
            try {
                if (!ui.tvDataSet.getSelectionModel().isEmpty()) {
                    drawPolygons();
                }
            } catch (Exception ex) {
                Logger.getLogger(DoseMapping.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
    });

    ui.miDrawContour.setOnAction(new EventHandler<ActionEvent>() {
        @Override
        public void handle(ActionEvent event) {
            try {
                if (!ui.tvDataSet.getSelectionModel().isEmpty()) {
                    drawContour();
                }
            } catch (Exception ex) {
                Logger.getLogger(DoseMapping.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
    });

    ui.miClearMap.setOnAction(new EventHandler<ActionEvent>() {
        @Override
        public void handle(ActionEvent event) {
            try {
                clearMap();
                for (CheckBox item : ws.activeMapItemsList) {
                    item.setSelected(false);
                }
            } catch (Exception ex) {
                Logger.getLogger(DoseMapping.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
    });

// Link UI with observable data
```

```
        connection.getTableDataSets(ws.dataSetList);
        connection.getTableDetectors(ws.detectorList);
        connection.getTableGpss(ws.gpsList);

        ui.tvDataSet.setItems(ws.dataSetList);
        ui.tvDataSet.getColumns().addAll(ui.tcDataSetId, ui.tcDataSetName, ui.tcDataSetDescription);
        ui.tcDataSetId.setCellValueFactory(new PropertyValueFactory<DataSet,String>("setId"));
        ui.tcDataSetName.setCellValueFactory(new PropertyValueFactory<DataSet,String>("setName"));
        ui.tcDataSetDescription.setCellValueFactory(new PropertyValueFactory<DataSet,String>("setDescription"));
//        ui.tcDataSetDate.setCellValueFactory(new PropertyValueFactory<DataSet,String>("dateAcquire"));

        ui.tvDataSet.getSelectionModel().selectedItemProperty().addListener((obs, oldSelection, newSelection) -> {
            if (newSelection != null) {
                String desc = "ID: " + ui.tvDataSet.getSelectionModel().getSelectedItem().getSetId()+ "\n";
                desc += "Name: " + ui.tvDataSet.getSelectionModel().getSelectedItem().getSetName() + "\n";
                desc += "Description:" + ui.tvDataSet.getSelectionModel().getSelectedItem().getSetDescription() + "\n";
//                desc += "Date Acquire: " + ui.tvDataSet.getSelectionModel().getSelectedItem().getDateAcquire() + "\n";
                desc += "Person Acquire: " + ui.tvDataSet.getSelectionModel().getSelectedItem().getPersonAcquire() +
"\n";
                desc += "Date Upload: " + ui.tvDataSet.getSelectionModel().getSelectedItem().getDateUpload() + "\n";
                desc += "Person Upload: " + ui.tvDataSet.getSelectionModel().getSelectedItem().getPersonUpload() +
"\n";
                desc += "Detector: " + connection.getDetectorByID(ws.detectorList,
ui.tvDataSet.getSelectionModel().getSelectedItem().getDetectorId()).getDetectorName() + "\n";
                desc += "GPS: " + connection.getGpsByID(ws.gpsList,
ui.tvDataSet.getSelectionModel().getSelectedItem().getGpsId()).getGpsName();
                ui.taDescription.setText(desc);
            }
        });

        ui.tvDetector.setItems(ws.detectorList);
        ui.tvDetector.getColumns().addAll(ui.tcDetectorId, ui.tcDetectorName, ui.tcDetectorDescription);
        ui.tcDetectorId.setCellValueFactory(new PropertyValueFactory<Detector,String>("detectorId"));
        ui.tcDetectorName.setCellValueFactory(new PropertyValueFactory<Detector,String>("detectorName"));
        ui.tcDetectorDescription.setCellValueFactory(new
PropertyValueFactory<Detector,String>("detectorDescription"));

        ui.tvDetector.getSelectionModel().selectedItemProperty().addListener((obs, oldSelection, newSelection) -> {
            if (newSelection != null) {
                String desc = "ID: " + ui.tvDetector.getSelectionModel().getSelectedItem().getDetectorId()+ "\n";
                desc += "Name: " + ui.tvDetector.getSelectionModel().getSelectedItem().getDetectorName() + "\n";
                desc += "Description: " + ui.tvDetector.getSelectionModel().getSelectedItem().getDetectorDescription() +
"\n";
                desc += "Manufacturer: " +
ui.tvDetector.getSelectionModel().getSelectedItem().getDetectorManufacturer();
                ui.taDescription.setText(desc);
            }
        });

        ui.tvGps.setItems(ws.gpsList);
        ui.tvGps.getColumns().addAll(ui.tcGpsId, ui.tcGpsName, ui.tcGpsDescription);
```

```
        ui.tcGpsId.setCellValueFactory(new PropertyValueFactory<Gps,String>("gpsId"));
        ui.tcGpsName.setCellValueFactory(new PropertyValueFactory<Gps,String>("gpsName"));
        ui.tcGpsDescription.setCellValueFactory(new PropertyValueFactory<Gps,String>("gpsDescription"));

        ui.tvGps.getSelectionModel().selectedItemProperty().addListener((obs, oldSelection, newSelection) -> {
            if (newSelection != null) {
                String desc = "ID: " + ui.tvGps.getSelectionModel().getSelectedItem().getGpsId()+ "\n";
                desc += "Name: " + ui.tvGps.getSelectionModel().getSelectedItem().getGpsName() + "\n";
                desc += "Description: " + ui.tvGps.getSelectionModel().getSelectedItem().getGpsDescription() + "\n";
                desc += "Manufacturer: " + ui.tvGps.getSelectionModel().getSelectedItem().getGpsManufacturer();
                ui.taDescription.setText(desc);
            }
        });

        ui.tpTable.getSelectionModel().selectedItemProperty().addListener((obj, oldSelection, newSelection) -> {
            if (newSelection != null) {
                ui.tvDataSet.getSelectionModel().clearSelection();
                ui.tvDetector.getSelectionModel().clearSelection();
            }
        });

        ui.checkList.setItems(ws.activeMapItemsList);
//        activeMapList.addAll(ui.cbPath, ui.cbMarker, ui.cbCircle, ui.cbDelaunayTriangle);

        ui.checkList.getSelectionModel().selectedItemProperty().addListener((obs, oldSelection, newSelection) -> {
            double minDose = 0;
            double maxDose = 0;
            double minLat = 0;
            double maxLat = 0;
            double minLon = 0;
            double maxLon = 0;

            if (newSelection != null) {
                String name = ui.checkList.getSelectionModel().getSelectedItem().getText();
                if (ui.checkList.getSelectionModel().getSelectedItem().getUserData().getClass().isInstance(new Polylines())) {
                    minDose = ((Polylines)ui.checkList.getSelectionModel().getSelectedItem().getUserData()).getMinDose();
                    maxDose = ((Polylines)ui.checkList.getSelectionModel().getSelectedItem().getUserData()).getMaxDose();
                    minLat = ((Polylines)ui.checkList.getSelectionModel().getSelectedItem().getUserData()).getMinLat();
                    maxLat = ((Polylines)ui.checkList.getSelectionModel().getSelectedItem().getUserData()).getMaxLat();
                    minLon = ((Polylines)ui.checkList.getSelectionModel().getSelectedItem().getUserData()).getMinLon();
                    maxLon = ((Polylines)ui.checkList.getSelectionModel().getSelectedItem().getUserData()).getMaxLon();
                    resizeMap(maxLat, minLat, maxLon, minLon);
                    ui.mapLabel.setLabel("polylines", name, minDose, maxDose);
                } else if (ui.checkList.getSelectionModel().getSelectedItem().getUserData().getClass().isInstance(new
Markers())) {
                    minDose = ((Markers)ui.checkList.getSelectionModel().getSelectedItem().getUserData()).getMinDose();
                    maxDose = ((Markers)ui.checkList.getSelectionModel().getSelectedItem().getUserData()).getMaxDose();
                    minLat = ((Markers)ui.checkList.getSelectionModel().getSelectedItem().getUserData()).getMinLat();
                    maxLat = ((Markers)ui.checkList.getSelectionModel().getSelectedItem().getUserData()).getMaxLat();
                    minLon = ((Markers)ui.checkList.getSelectionModel().getSelectedItem().getUserData()).getMinLon();
                    maxLon = ((Markers)ui.checkList.getSelectionModel().getSelectedItem().getUserData()).getMaxLon();
```

```
                    resizeMap(maxLat, minLat, maxLon, minLon);
                    ui.mapLabel.setLabel("markers", name, minDose, maxDose);
                } else if (ui.checkList.getSelectionModel().getSelectedItem().getUserData().getClass().isInstance(new
Circles())) {
                    minDose = ((Circles)ui.checkList.getSelectionModel().getSelectedItem().getUserData()).getMinDose();
                    maxDose = ((Circles)ui.checkList.getSelectionModel().getSelectedItem().getUserData()).getMaxDose();
                    minLat = ((Circles)ui.checkList.getSelectionModel().getSelectedItem().getUserData()).getMinLat();
                    maxLat = ((Circles)ui.checkList.getSelectionModel().getSelectedItem().getUserData()).getMaxLat();
                    minLon = ((Circles)ui.checkList.getSelectionModel().getSelectedItem().getUserData()).getMinLon();
                    maxLon = ((Circles)ui.checkList.getSelectionModel().getSelectedItem().getUserData()).getMaxLon();
                    resizeMap(maxLat, minLat, maxLon, minLon);
                    ui.mapLabel.setLabel("circles", name, minDose, maxDose);
                } else if (ui.checkList.getSelectionModel().getSelectedItem().getUserData().getClass().isInstance(new
Polygons())) {
                    minDose = ((Polygons)ui.checkList.getSelectionModel().getSelectedItem().getUserData()).getMinDose();
                    maxDose = ((Polygons)ui.checkList.getSelectionModel().getSelectedItem().getUserData()).getMaxDose();
                    minLat = ((Polygons)ui.checkList.getSelectionModel().getSelectedItem().getUserData()).getMinLat();
                    maxLat = ((Polygons)ui.checkList.getSelectionModel().getSelectedItem().getUserData()).getMaxLat();
                    minLon = ((Polygons)ui.checkList.getSelectionModel().getSelectedItem().getUserData()).getMinLon();
                    maxLon = ((Polygons)ui.checkList.getSelectionModel().getSelectedItem().getUserData()).getMaxLon();
                    resizeMap(maxLat, minLat, maxLon, minLon);
                    ui.mapLabel.setLabel("polygons", name, minDose, maxDose);
                } else if (ui.checkList.getSelectionModel().getSelectedItem().getUserData().getClass().isInstance(new
Contour())) {
                    minDose = ((Contour)ui.checkList.getSelectionModel().getSelectedItem().getUserData()).getMinDose();
                    maxDose = ((Contour)ui.checkList.getSelectionModel().getSelectedItem().getUserData()).getMaxDose();
                    minLat = ((Contour)ui.checkList.getSelectionModel().getSelectedItem().getUserData()).getMinLat();
                    maxLat = ((Contour)ui.checkList.getSelectionModel().getSelectedItem().getUserData()).getMaxLat();
                    minLon = ((Contour)ui.checkList.getSelectionModel().getSelectedItem().getUserData()).getMinLon();
                    maxLon = ((Contour)ui.checkList.getSelectionModel().getSelectedItem().getUserData()).getMaxLon();
                    resizeMap(maxLat, minLat, maxLon, minLon);
                    ui.mapLabel.setLabel("contour", name, minDose, maxDose);
                }
            }
        });
    // Add listener
        ui.mapComponent.addMapInitializedListener(this);

    // Create primary stage
        Rectangle2D visualBounds = Screen.getPrimary().getVisualBounds();
        Scene scene = new Scene(ui.bp, visualBounds.getWidth(), visualBounds.getHeight());
        scene.getStylesheets().add("dosemappingstyle.css");
        primaryStage.setTitle("Dose Mapping");
        primaryStage.setScene(scene);
        primaryStage.show();
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
```

```
        launch(args);
    }

    @Override
    public void mapInitialized() {
        MapOptions mapOptions = new MapOptions();

        mapOptions.center(new LatLong(13.7420, 100.5140))
                .mapType(MapTypeIdEnum.ROADMAP)
                .zoom(9);
        ui.map = ui.mapComponent.createMap(mapOptions, false);

        ui.lbLatitude.setText("Lat = " + formatter.format(ui.map.getCenter().getLatitude()));
        ui.lbLongitude.setText("Lon = " + formatter.format(ui.map.getCenter().getLongitude()));

        ui.map.addMouseEventHandler(UIEventType.click, (GMapMouseEvent event) -> {
            LatLong latLong = event.getLatLong();
            ui.lbLatitude.setText("Lat = " + formatter.format(latLong.getLatitude()));
            ui.lbLongitude.setText("Lon = " + formatter.format(latLong.getLongitude()));
        });
    }

    public void drawLines() {
        double maxLat;
        double minLat;
        double maxLon;
        double minLon;
        double maxDose;
        double minDose;

        Polylines mapPolylines = new Polylines();
        List<DoseData> doseDatas = new ArrayList<>();
        connection.getDoseData(doseDatas, ui.tvDataSet.getSelectionModel().getSelectedItem());

        if (doseDatas.size() > 1) {
            maxLat = doseDatas.get(0).getLat();
            minLat = doseDatas.get(0).getLat();
            maxLon = doseDatas.get(0).getLon();
            minLon = doseDatas.get(0).getLon();
            maxDose = doseDatas.get(0).getDose();
            minDose = doseDatas.get(0).getDose();

            for (int i = 0; i < doseDatas.size(); i++) {
                double dose = doseDatas.get(i).getDose();
                if (dose >= maxDose) {
                    maxDose = dose;
                }
                if (dose <= minDose) {
                    minDose = dose;
                }
            }
```

```
            mapPolylines.setMinDose(minDose);
            mapPolylines.setMaxDose(maxDose);

            for (int i = 0; i < doseDatas.size()-1; i++){
                LatLong l1 = new LatLong(doseDatas.get(i).getLat(), doseDatas.get(i).getLon());
                LatLong l2 = new LatLong(doseDatas.get(i+1).getLat(), doseDatas.get(i+1).getLon());
                LatLong[] ary = new LatLong[]{l1, l2};
                MVCArray mvc = new MVCArray(ary);

                double average = (doseDatas.get(i).getDose() + doseDatas.get(i+1).getDose()) / 2;

                int bcolor = (int) ((average - minDose) / (maxDose - minDose) * 255);
                int rgcolor = 255 - bcolor;

                PolylineOptions polyOpts = new PolylineOptions()
                        .path(mvc)
    //                    .strokeColor("rgb(255," + rgcolor + "," + rgcolor + ")")
                        .strokeColor(getColor(average))
                        .strokeWeight(2);

                Polyline polyline = new Polyline(polyOpts);
                mapPolylines.setPolyline(polyline);
                ui.map.addMapShape(polyline);

                if (l1.getLatitude() > maxLat) {
                    maxLat = l1.getLatitude();
                }
                if (l1.getLatitude() < minLat) {
                    minLat = l1.getLatitude();
                }
                if (l1.getLongitude() > maxLon) {
                    maxLon = l1.getLongitude();
                }
                if (l1.getLongitude() < minLon) {
                    minLon = l1.getLongitude();
                }
                if (doseDatas.get(i).getDose() > maxDose) {
                    maxDose = doseDatas.get(i).getDose();
                }
                if (doseDatas.get(i).getDose() < minDose) {
                    minDose = doseDatas.get(i).getDose();
                }
            }
            mapPolylines.setMinLat(minLat);
            mapPolylines.setMaxLat(maxLat);
            mapPolylines.setMinLon(minLon);
            mapPolylines.setMaxLon(maxLon);

            resizeMap(maxLat, minLat, maxLon, minLon);
            ws.mapPolylinesGroup.add(mapPolylines);
            CheckBox ch = new CheckBox("(" + ui.tvDataSet.getSelectionModel().getSelectedItem().getSetId() + ") Lines");
```

```
            ui.mapLabel.setLabel("polylines");
            ch.setUserData(mapPolylines);
            setMapLinesCheckBox(ch);
            ch.setSelected(true);
            ws.activeMapItemsList.add(ch);
        }
        doseDatas.clear();
    }

    public void drawMarkers() {
        double maxLat;
        double minLat;
        double maxLon;
        double minLon;
        double maxDose;
        double minDose;

        Markers mapMarkers = new Markers();
        List<DoseData> doseDatas = new ArrayList<>();
        connection.getDoseData(doseDatas, ui.tvDataSet.getSelectionModel().getSelectedItem());

        if (doseDatas.size() > 0) {
            maxLat = doseDatas.get(0).getLat();
            minLat = doseDatas.get(0).getLat();
            maxLon = doseDatas.get(0).getLon();
            minLon = doseDatas.get(0).getLon();
            maxDose = doseDatas.get(0).getDose();
            minDose = doseDatas.get(0).getDose();

            for (int i = 0; i < doseDatas.size(); i++) {
                double dose = doseDatas.get(i).getDose();
                if (dose >= maxDose) {
                    maxDose = dose;
                }
                if (dose <= minDose) {
                    minDose = dose;
                }
            }
            mapMarkers.setMinDose(minDose);
            mapMarkers.setMaxDose(maxDose);

            for (int i = 0; i < doseDatas.size(); i++){
                MarkerOptions markerOptions = new MarkerOptions();
                LatLong markerLatLong = new LatLong(doseDatas.get(i).getLat(), doseDatas.get(i).getLon());
                markerOptions.position(markerLatLong).visible(true);

                Marker myMarker = new Marker(markerOptions);
                mapMarkers.setMarker(myMarker);
                ui.map.addMarker(myMarker);

                if (markerLatLong.getLatitude() > maxLat) {
```

```
                    maxLat = markerLatLong.getLatitude();
                }
                if (markerLatLong.getLatitude() < minLat) {
                    minLat = markerLatLong.getLatitude();
                }
                if (markerLatLong.getLongitude() > maxLon) {
                    maxLon = markerLatLong.getLongitude();
                }
                if (markerLatLong.getLongitude() < minLon) {
                    minLon = markerLatLong.getLongitude();
                }
            }
            mapMarkers.setMinLat(minLat);
            mapMarkers.setMaxLat(maxLat);
            mapMarkers.setMinLon(minLon);
            mapMarkers.setMaxLon(maxLon);

            resizeMap(maxLat, minLat, maxLon, minLon);
            ws.mapMarkersGroup.add(mapMarkers);
            CheckBox ch = new CheckBox("(" + ui.tvDataSet.getSelectionModel().getSelectedItem().getSetId() + ")
Markers");
            ui.mapLabel.setLabel("markers", ch.getText(), minDose, maxDose);
            ch.setUserData(mapMarkers);
            setMapMarkersCheckBox(ch);
            ch.setSelected(true);
            ws.activeMapItemsList.add(ch);
        }
        doseDatas.clear();
    }

    public void drawCircles() {
        double maxLat;
        double minLat;
        double maxLon;
        double minLon;
        double maxDose;
        double minDose;

        Circles mapCircles = new Circles();
        List<DoseData> doseDatas = new ArrayList<>();
        connection.getDoseData(doseDatas, ui.tvDataSet.getSelectionModel().getSelectedItem());

        if (doseDatas.size() > 0) {
            maxLat = doseDatas.get(0).getLat();
            minLat = doseDatas.get(0).getLat();
            maxLon = doseDatas.get(0).getLon();
            minLon = doseDatas.get(0).getLon();
            maxDose = doseDatas.get(0).getDose();
            minDose = doseDatas.get(0).getDose();

            for (int i = 0; i < doseDatas.size(); i++) {
```

```
                double dose = doseDatas.get(i).getDose();
                if (dose >= maxDose) {
                    maxDose = dose;
                }
                if (dose <= minDose) {
                    minDose = dose;
                }
            }

            mapCircles.setMinDose(minDose);
            mapCircles.setMaxDose(maxDose);

            for (int i = 0; i < doseDatas.size(); i++){
                LatLong markerLatLong = new LatLong(doseDatas.get(i).getLat(), doseDatas.get(i).getLon());

    //              int bcolor = (int) ((doseDatas.get(i).getDose() - minDose) / (maxDose - minDose) * 255);
    //              int rgcolor = 255 - bcolor;

                CircleOptions circleOptions = new CircleOptions();
                circleOptions.center(markerLatLong).visible(true);
                circleOptions.radius(2000)
                        .strokeWeight(0)
    //                    .fillColor("rgb(" + rgcolor + ",255," + rgcolor + ")")
                        .fillColor(getColor(doseDatas.get(i).getDose()))
                        .fillOpacity(0.5);
                Circle myCircle = new Circle(circleOptions);
                mapCircles.setCircle(myCircle);
                ui.map.addMapShape(myCircle);

                if (markerLatLong.getLatitude() > maxLat) {
                    maxLat = markerLatLong.getLatitude();
                }
                if (markerLatLong.getLatitude() < minLat) {
                    minLat = markerLatLong.getLatitude();
                }
                if (markerLatLong.getLongitude() > maxLon) {
                    maxLon = markerLatLong.getLongitude();
                }
                if (markerLatLong.getLongitude() < minLon) {
                    minLon = markerLatLong.getLongitude();
                }
                if (doseDatas.get(i).getDose() > maxDose) {
                    maxDose = doseDatas.get(i).getDose();
                }
                if (doseDatas.get(i).getDose() < minDose) {
                    minDose = doseDatas.get(i).getDose();
                }
            }
            mapCircles.setMinLat(minLat);
            mapCircles.setMaxLat(maxLat);
            mapCircles.setMinLon(minLon);
```

```
            mapCircles.setMaxLon(maxLon);

            resizeMap(maxLat, minLat, maxLon, minLon);
            ws.mapCirclesGroup.add(mapCircles);
            CheckBox ch = new CheckBox("(" + ui.tvDataSet.getSelectionModel().getSelectedItem().getSetId() + ")
Circles");
//              ui.mapLabel.setLabel("circles", ch.getText(), minDose, maxDose);
            ui.mapLabel.setLabel("circles");
            ch.setUserData(mapCircles);
            setMapCirclesCheckBox(ch);
            ch.setSelected(true);
            ws.activeMapItemsList.add(ch);
        }
        doseDatas.clear();
    }

    public void drawPolygons() {
        double maxLat;
        double minLat;
        double maxLon;
        double minLon;
        double maxDose;
        double minDose;

        DelaunayTriangulator dt;
        List<Vector2D> points = new ArrayList<>();
        List<Triangle2D> triangles = new ArrayList<>();

        Polygons mapPolygons = new Polygons();
        List<DoseData> doseDatas = new ArrayList<>();
        connection.getDoseData(doseDatas, ui.tvDataSet.getSelectionModel().getSelectedItem());

        if (doseDatas.size() > 2) {
            maxLat = doseDatas.get(0).getLat();
            minLat = doseDatas.get(0).getLat();
            maxLon = doseDatas.get(0).getLon();
            minLon = doseDatas.get(0).getLon();
            maxDose = doseDatas.get(0).getDose();
            minDose = doseDatas.get(0).getDose();

            for (int i = 0; i < doseDatas.size(); i++) {
                double dose = doseDatas.get(i).getDose();
                if (dose >= maxDose) {
                    maxDose = dose;
                }
                if (dose <= minDose) {
                    minDose = dose;
                }
            }

            mapPolygons.setMinDose(minDose);
```

```
            mapPolygons.setMaxDose(maxDose);

            for (int i = 0; i < doseDatas.size(); i++){
                points.add(new Vector2D(doseDatas.get(i).getLon(), doseDatas.get(i).getLat()));

                if (doseDatas.get(i).getLat() > maxLat) {
                    maxLat = doseDatas.get(i).getLat();
                }
                if (doseDatas.get(i).getLat() < minLat) {
                    minLat = doseDatas.get(i).getLat();
                }
                if (doseDatas.get(i).getLon() > maxLon) {
                    maxLon = doseDatas.get(i).getLon();
                }
                if (doseDatas.get(i).getLon() < minLon) {
                    minLon = doseDatas.get(i).getLon();
                }
                if (doseDatas.get(i).getDose() > maxDose) {
                    maxDose = doseDatas.get(i).getDose();
                }
                if (doseDatas.get(i).getDose() < minDose) {
                    minDose = doseDatas.get(i).getDose();
                }
            }
            mapPolygons.setMinLat(minLat);
            mapPolygons.setMaxLat(maxLat);
            mapPolygons.setMinLon(minLon);
            mapPolygons.setMaxLon(maxLon);

            dt = new DelaunayTriangulator(points);
            try {
                dt.triangulate();
            } catch (NotEnoughPointsException ex) {
                Logger.getLogger(DoseMapping.class.getName()).log(Level.SEVERE, null, ex);
            }
            triangles = dt.getTriangles();

            for (int i = 0; i < triangles.size(); i++){
                LatLong poly1 = new LatLong(triangles.get(i).a.y, triangles.get(i).a.x);
                LatLong poly2 = new LatLong(triangles.get(i).b.y, triangles.get(i).b.x);
                LatLong poly3 = new LatLong(triangles.get(i).c.y, triangles.get(i).c.x);
                LatLong[] pAry = new LatLong[]{poly1, poly2, poly3};
                MVCArray pmvc = new MVCArray(pAry);

                double average = findAverage(triangles.get(i), points, doseDatas);

//                int bcolor = (int) ((average - minDose) / (maxDose - minDose) * 255);
//                int rgcolor = 255 - bcolor;
                PolygonOptions polygOpts = new PolygonOptions()
                        .paths(pmvc)
                        .strokeColor("blue")
```

```
                    .strokeWeight(0)
                    .editable(false)
//                    .fillColor("rgb(" + rgcolor + "," + rgcolor + ",255)")
                    .fillColor(getColor(average))
                    .fillOpacity(0.8);

            Polygon pg = new Polygon(polygOpts);
            mapPolygons.setPolygon(pg);
            ui.map.addMapShape(pg);
        }

        resizeMap(maxLat, minLat, maxLon, minLon);
        ws.mapPolygonsGroup.add(mapPolygons);
        CheckBox ch = new CheckBox("(" + ui.tvDataSet.getSelectionModel().getSelectedItem().getSetId() + ")
Polygons");
//          ui.mapLabel.setLabel("polygons", ch.getText(), minDose, maxDose);
        ui.mapLabel.setLabel("polygons");
        ch.setUserData(mapPolygons);
        setMapPolygonsCheckBox(ch);
        ch.setSelected(true);
        ws.activeMapItemsList.add(ch);
      }
      doseDatas.clear();
   }

   public void drawContour() {
      double maxLat;
      double minLat;
      double maxLon;
      double minLon;
      double maxDose;
      double minDose;
//
      DelaunayTriangulator dt;
      List<Vector2D> points = new ArrayList<>();
      List<Triangle2D> triangles = new ArrayList<>();
      String[] label = new String[7];
//
      Contour mapContour = new Contour();
      List<DoseData> doseDatas = new ArrayList<>();
      connection.getDoseData(doseDatas, ui.tvDataSet.getSelectionModel().getSelectedItem());
//
      if (doseDatas.size() > 2) {
         maxLat = doseDatas.get(0).getLat();
         minLat = doseDatas.get(0).getLat();
         maxLon = doseDatas.get(0).getLon();
         minLon = doseDatas.get(0).getLon();
         maxDose = doseDatas.get(0).getDose();
         minDose = doseDatas.get(0).getDose();

         for (int i = 0; i < doseDatas.size(); i++) {
```

```
                double dose = doseDatas.get(i).getDose();
                if (dose >= maxDose) {
                    maxDose = dose;
                }
                if (dose <= minDose) {
                    minDose = dose;
                }
            }

            mapContour.setMinDose(minDose);
            mapContour.setMaxDose(maxDose);

            for (int i = 0; i < doseDatas.size(); i++){
                points.add(new Vector2D(doseDatas.get(i).getLon(), doseDatas.get(i).getLat()));

                if (doseDatas.get(i).getLat() > maxLat) {
                    maxLat = doseDatas.get(i).getLat();
                }
                if (doseDatas.get(i).getLat() < minLat) {
                    minLat = doseDatas.get(i).getLat();
                }
                if (doseDatas.get(i).getLon() > maxLon) {
                    maxLon = doseDatas.get(i).getLon();
                }
                if (doseDatas.get(i).getLon() < minLon) {
                    minLon = doseDatas.get(i).getLon();
                }
                if (doseDatas.get(i).getDose() > maxDose) {
                    maxDose = doseDatas.get(i).getDose();
                }
                if (doseDatas.get(i).getDose() < minDose) {
                    minDose = doseDatas.get(i).getDose();
                }
            }
            mapContour.setMinLat(minLat);
            mapContour.setMaxLat(maxLat);
            mapContour.setMinLon(minLon);
            mapContour.setMaxLon(maxLon);

            dt = new DelaunayTriangulator(points);
            try {
                dt.triangulate();
            } catch (NotEnoughPointsException ex) {
                Logger.getLogger(DoseMapping.class.getName()).log(Level.SEVERE, null, ex);
            }
            triangles = dt.getTriangles();

    //          for (int i = 0; i < triangles.size(); i++){
    //              LatLong poly1 = new LatLong(triangles.get(i).a.y, triangles.get(i).a.x);
    //              LatLong poly2 = new LatLong(triangles.get(i).b.y, triangles.get(i).b.x);
    //              LatLong poly3 = new LatLong(triangles.get(i).c.y, triangles.get(i).c.x);
```

```
//              LatLong[] pAry = new LatLong[]{poly1, poly2, poly3};
//              MVCArray pmvc = new MVCArray(pAry);
//
//              PolygonOptions polygOpts = new PolygonOptions()
//                      .paths(pmvc)
//                      .strokeColor("blue")
//                      .strokeWeight(0)
//                      .editable(false)
//                      .fillColor("white")
//                      .fillOpacity(0.8);
//
//              Polygon pg = new Polygon(polygOpts);
//              mapContour.setContour(pg);
//              ui.map.addMapShape(pg);
//          }

            double dDose = maxDose - minDose;

            for (int i = 1; i < 8; i++) {
//              double dose = minDose + (i/8.0) * dDose;
                double dose = Config.DOSE_RATE_RANGE * i;
                double[] locs = {0, 0, 0, 0};
                label[i-1] = String.format("%.2f", dose);

                for (Triangle2D tri : triangles) {
                    if (isInside(tri, dose, points, doseDatas, locs)) {
//                      System.out.println(locs[0] + " " + locs[1] + " " + locs[2] + " " + locs[3]);
                        LatLong l1 = new LatLong(locs[1], locs[0]);
                        LatLong l2 = new LatLong(locs[3], locs[2]);
                        LatLong[] ary = new LatLong[]{l1, l2};
                        MVCArray mvc = new MVCArray(ary);

//                      int bcolor = (int) ((dose - minDose) / (maxDose - minDose) * 255);
//                      int rgcolor = 255 - bcolor;

                        PolylineOptions polyOpts = new PolylineOptions()
                                .path(mvc)
                                .strokeColor(getColor("level" + i))
                                .strokeWeight(2);

                        Polyline polyline = new Polyline(polyOpts);
                        mapContour.setContour(polyline);
                        ui.map.addMapShape(polyline);
                    }
                }
            }

// Test creating polygon
//          double dose = 20;
//          double[] locs = {0, 0, 0, 0};
//          LatLong l1;
```

```
//            LatLong l2;
//            List<LatLong> lcw = new ArrayList<>();
//            List<LatLong> lccw = new ArrayList<>();
//            int[] mark = new int[triangles.size()];
//            boolean foundAll = false;
//
//            for (int i = 0; i < triangles.size(); i++) {
//                mark[i] = 0;
//            }
//            for (int ti = 0; ti < triangles.size(); ti++) {
//                if (mark[ti] == 0) {
//                    foundAll = false;
//                    if (isInside(triangles.get(ti), dose, points, doseDatas, locs)) {
//                        l1 = new LatLong(locs[1], locs[0]);
//                        l2 = new LatLong(locs[3], locs[2]);
//                        mark[ti] = 1;
//                        int curenti = ti;
//                        while (!foundAll) {
//                            boolean found = false;
//                            for (int tj = 0; tj < triangles.size(); tj++) {
//                                if (isInside(triangles.get(tj), dose, points, doseDatas, locs)) {
//
//                                }
//                            }
//                        }
//                    } else {
//                        mark[ti] = 1;
//                    }
//                }
//            }

// End test creating polygon

            resizeMap(maxLat, minLat, maxLon, minLon);
            ws.mapContourGroup.add(mapContour);
            CheckBox ch = new CheckBox("(" + ui.tvDataSet.getSelectionModel().getSelectedItem().getSetId() + ")
Contour");
//              ui.mapLabel.setLabel("contour", ch.getText(), minDose, maxDose);
            ui.mapLabel.setLabel(label);
            ch.setUserData(mapContour);
            setMapContourCheckBox(ch);
            ch.setSelected(true);
            ws.activeMapItemsList.add(ch);
        }
        doseDatas.clear();
    }

    public void clearAllPolylines() {
        for (int j = ws.mapPolylinesGroup.size()-1; j >= 0; j--) {
            for (int i = 0; i < ws.mapPolylinesGroup.get(j).getPolylines().size(); i++) {
                ui.map.removeMapShape(ws.mapPolylinesGroup.get(j).getPolylines().get(i));
```

```
            }
            ws.mapPolylinesGroup.remove(ws.mapPolylinesGroup.get(j));
        }

        ws.mapPolylinesGroup.clear();
    }

    public void clearAllMarkers() {
        for (int j = ws.mapMarkersGroup.size()-1; j >= 0; j--) {
            for (int i = 0; i < ws.mapMarkersGroup.get(j).getMarkers().size(); i++) {
                ui.map.removeMarker(ws.mapMarkersGroup.get(j).getMarkers().get(i));
            }
            ws.mapMarkersGroup.remove(ws.mapMarkersGroup.get(j));
        }

        ws.mapMarkersGroup.clear();
    }

    public void clearAllCircles() {
        for (int j = ws.mapCirclesGroup.size()-1; j >= 0; j--) {
            for (int i = 0; i < ws.mapCirclesGroup.get(j).getCircles().size(); i++) {
                ui.map.removeMapShape(ws.mapCirclesGroup.get(j).getCircles().get(i));
            }
            ws.mapCirclesGroup.remove(ws.mapCirclesGroup.get(j));
        }

        ws.mapCirclesGroup.clear();
    }

    public void clearAllPolygons() {
        for (int j = ws.mapPolygonsGroup.size()-1; j >= 0; j--) {
            for (int i = 0; i < ws.mapPolygonsGroup.get(j).getPolygons().size(); i++) {
                ui.map.removeMapShape(ws.mapPolygonsGroup.get(j).getPolygons().get(i));
            }
            ws.mapPolygonsGroup.remove(ws.mapPolygonsGroup.get(j));
        }

        ws.mapPolygonsGroup.clear();
    }

    public void clearAllContour() {
        for (int j = ws.mapContourGroup.size()-1; j >= 0; j--) {
            for (int i = 0; i < ws.mapContourGroup.get(j).getContourPolygons().size(); i++) {
                ui.map.removeMapShape(ws.mapContourGroup.get(j).getContourPolygons().get(i));
            }
            for (int i = 0; i < ws.mapContourGroup.get(j).getContourPolylines().size(); i++) {
                ui.map.removeMapShape(ws.mapContourGroup.get(j).getContourPolylines().get(i));
            }
            ws.mapContourGroup.remove(ws.mapContourGroup.get(j));
        }
```

```
        ws.mapContourGroup.clear();
    }

    public void clearMap() {
        clearAllPolylines();
        clearAllMarkers();
        clearAllCircles();
        clearAllPolygons();
        clearAllContour();
        ws.activeMapItemsList.clear();
        ui.mapLabel.hideLabel();
    }

    public void resizeMap(double maxLat, double minLat, double maxLon, double minLon) {
        LatLongBounds bound = new LatLongBounds(new LatLong(minLat, minLon), new LatLong(maxLat, maxLon));
        ui.map.fitBounds(bound);
    }

    public double findAverage(Triangle2D triangle, List<Vector2D> points, List<DoseData> doseData) {
        List<Double> ele = new ArrayList<>();
        int found = 0;
        int i = 0;
        double average = 0;

        for (Vector2D vector : points) {
            if (triangle.hasVertex(vector)) {
                ele.add(doseData.get(i).getDose());
                found++;
            }
            if (found == 3) {
                break;
            }
            i++;
        }

        if (found != 3) {
            System.out.println("error");

        } else {
            average = (ele.get(0) + ele.get(1) + ele.get(2)) / 3;
        }

        return average;
    }

    public boolean isInside(Triangle2D triangle, double dose, List<Vector2D> points, List<DoseData> doseData,
double[] locs) {
        List<Double> ele = new ArrayList<>();
        List<Vector2D> vertex = new ArrayList<>();
        List<Double> x = new ArrayList<>();
        List<Double> y = new ArrayList<>();
```

```
        int found = 0;
        int i = 0;
        double maxDose = 0;
        double minDose = 0;
        boolean isInside = false;

        for (Vector2D vector : points) {
            if (triangle.hasVertex(vector)) {
                ele.add(doseData.get(i).getDose());
                vertex.add(vector);
                found++;
            }
            if (found == 3) {
                break;
            }
            i++;
        }

        if (found != 3) {
            System.out.println("error");
        } else {
            minDose = ele.get(0);
            maxDose = ele.get(0);

            if (minDose > ele.get(1)) { minDose = ele.get(1); }
            if (minDose > ele.get(2)) { minDose = ele.get(2); }
            if (maxDose < ele.get(1)) { maxDose = ele.get(1); }
            if (maxDose < ele.get(2)) { maxDose = ele.get(2); }

            if ((dose >= minDose) & (dose <= maxDose)) {
//                System.out.println(dose + " " + maxDose + " " + minDose + " " + ele.get(0) + " " + ele.get(1) + " " +
ele.get(2));
                isInside = true;
                if ((dose-ele.get(0))*(dose-ele.get(1)) <= 0) {
                    x.add(vertex.get(0).x + (dose-ele.get(0))/(ele.get(1)-ele.get(0))*(vertex.get(1).x-vertex.get(0).x));
                    y.add(vertex.get(0).y + (dose-ele.get(0))/(ele.get(1)-ele.get(0))*(vertex.get(1).y-vertex.get(0).y));
                }
                if ((dose-ele.get(1))*(dose-ele.get(2)) <= 0) {
                    x.add(vertex.get(1).x + (dose-ele.get(1))/(ele.get(2)-ele.get(1))*(vertex.get(2).x-vertex.get(1).x));
                    y.add(vertex.get(1).y + (dose-ele.get(1))/(ele.get(2)-ele.get(1))*(vertex.get(2).y-vertex.get(1).y));
                }
                if ((dose-ele.get(0))*(dose-ele.get(2)) <= 0) {
                    x.add(vertex.get(0).x + (dose-ele.get(0))/(ele.get(2)-ele.get(0))*(vertex.get(2).x-vertex.get(0).x));
                    y.add(vertex.get(0).y + (dose-ele.get(0))/(ele.get(2)-ele.get(0))*(vertex.get(2).y-vertex.get(0).y));
                }
                locs[0] = x.get(0);
                locs[1] = y.get(0);
                locs[2] = x.get(1);
                locs[3] = y.get(1);
            }
        }
```

```
        return isInside;
    }

    public void setAddDetectorStage(Stage stage) {

        UIAddDetector uiAddDetector = new UIAddDetector();

    // Set UI behavior
        uiAddDetector.btCancel.setOnAction(new EventHandler<ActionEvent>() {
            @Override
            public void handle(ActionEvent event) {
                stage.close();
            }
        });

        uiAddDetector.btAdd.setOnAction(new EventHandler<ActionEvent>() {
            @Override
            public void handle(ActionEvent event) {
                try {
                    if (CheckInput.isValidName(uiAddDetector.tfName.getText())
                        & CheckInput.isValidDescription(uiAddDetector.tfDescription.getText())
                        & CheckInput.isValidName(uiAddDetector.tfManufacturer.getText())) {

                        connection.addDetector(ws.detectorList, uiAddDetector.tfName.getText(),
uiAddDetector.tfDescription.getText(), uiAddDetector.tfManufacturer.getText());
                    }
                } catch (Exception ex) {
                    System.out.println("Incomplete information. Cannot add this detector.");
                }
                stage.close();
            }
        });

    // Create add set stage
        Scene scene = new Scene(uiAddDetector.gp);
        stage.setScene(scene);
        stage.setTitle("Add Detector");
    }

    public void setAddGpsStage(Stage stage) {

        UIAddGps uiAddGps = new UIAddGps();

    // Set UI behavior
        uiAddGps.btCancel.setOnAction(new EventHandler<ActionEvent>() {
            @Override
            public void handle(ActionEvent event) {
                stage.close();
            }
        });
```

```
        uiAddGps.btAdd.setOnAction(new EventHandler<ActionEvent>() {
            @Override
            public void handle(ActionEvent event) {
                try {
                    if (CheckInput.isValidName(uiAddGps.tfName.getText())
                        & CheckInput.isValidDescription(uiAddGps.tfDescription.getText())
                        & CheckInput.isValidName(uiAddGps.tfManufacturer.getText())) {

                        connection.addGps(ws.gpsList, uiAddGps.tfName.getText(), uiAddGps.tfDescription.getText(),
uiAddGps.tfManufacturer.getText());
                    }
                } catch (Exception ex) {
                    System.out.println("Incomplete information. Cannot add this gps.");
                }
                stage.close();
            }
        });

    // Create add set stage
        Scene scene = new Scene(uiAddGps.gp);
        stage.setScene(scene);
        stage.setTitle("Add GPS");
    }

    public void setAddDataStage(Stage stage) {

        UIImportData uiImportData = new UIImportData(ws);

    // Set UI behavior
        uiImportData.btSelectDataFile.setOnAction(new EventHandler<ActionEvent>() {
            @Override
            public void handle(ActionEvent event) {
                Node node = (Node) event.getSource();
                FileChooser fChooser = new FileChooser();
                fChooser.setTitle("Open Data File");
                File file = fChooser.showOpenDialog(stage);

                try {
                    if (file != null) {
                        uiImportData.lbFileName.setText(file.getPath());
                        ws.currentFile = file;
                    }
                } catch (Exception ex) {
                    Logger.getLogger(DoseMapping.class.getName()).log(Level.SEVERE, null, ex);
                }
            }
        });

        uiImportData.btCancel.setOnAction(new EventHandler<ActionEvent>() {
            @Override
```

```
            public void handle(ActionEvent event) {
                stage.close();
            }
        });

        uiImportData.btImportSet.setOnAction(new EventHandler<ActionEvent>() {
            @Override
            public void handle(ActionEvent event) {
                try {
                    if (CheckInput.isValidName(uiImportData.tfSetName.getText())
                        & CheckInput.isValidDescription(uiImportData.tfSetDescription.getText())
                        & CheckInput.isValidDate(uiImportData.dpAcquireDate.getValue())
                        & CheckInput.isValidName(uiImportData.tfAcquirePerson.getText())
//                        & CheckInput.isValidDate(uiImportData.dpUploadDate.getValue())
                        & CheckInput.isValidName(uiImportData.tfUploadPerson.getText())
                        & ws.currentFile != null) {

                        Date acquireDate = new Date(uiImportData.dpAcquireDate.getValue().getYear()-1900,
uiImportData.dpAcquireDate.getValue().getMonthValue()-1, uiImportData.dpAcquireDate.getValue().getDayOfMonth());
                        Date uploadDate = Date.valueOf(LocalDate.now());

                        List<DoseData> doseDatas = new ArrayList<>();
                //        doseDatas = dataLoader.getDataFromGPX(currentFile);
                        dataLoader.getDataFromTXT(doseDatas, ws.currentFile);

                        int did =
Integer.parseInt(ws.detectorList.get(uiImportData.cbDetector.getSelectionModel().getSelectedIndex()).getDetectorId());
                        int gid =
Integer.parseInt(ws.gpsList.get(uiImportData.cbGps.getSelectionModel().getSelectedIndex()).getGpsId());

                        if (doseDatas.size() > 0) {
                            connection.addDataSet(ws.dataSetList, doseDatas, uiImportData.tfSetName.getText(),
uiImportData.tfSetDescription.getText(), acquireDate, uiImportData.tfAcquirePerson.getText(), uploadDate,
uiImportData.tfUploadPerson.getText(), did, gid);
                        }
                    }
                } catch (Exception ex) {
                    System.out.println("Incomplete information. Cannot add this data set.");
                }
                ws.currentFile = null;
                stage.close();
            }
        });

    // Create add set stage
        Scene scene = new Scene(uiImportData.gp);
        stage.setScene(scene);
        stage.setTitle("Import Data");
    }

    public void setAcquireDataStage(Stage stage) throws InterruptedException {
```

```
        UIAcquireData uiAcquireData = new UIAcquireData(ws);

        DataAcquisition dataAcquisition = new DataAcquisition();
        dataAcquisition.start(uiAcquireData);

        Thread.sleep(2000);  // Add delay to check connection status in dataAcquisition thread

        if (!dataAcquisition.isReady()) {
            uiAcquireData.disableAll();
        } else {
            uiAcquireData.enableAll();
        }
        if (dataAcquisition.detectorIsReady()) {
            uiAcquireData.lbDetectorStatusValue.setText("Ready");
        } else {
            uiAcquireData.lbDetectorStatusValue.setText("Not ready");
        }
        if (dataAcquisition.gpsIsReady()) {
            uiAcquireData.lbGpsStatusValue.setText("Ready");
        } else {
            uiAcquireData.lbGpsStatusValue.setText("Not ready");
        }

    // Set UI behavior
        uiAcquireData.btRecheckStatus.setOnAction(new EventHandler<ActionEvent>() {
            @Override
            public void handle(ActionEvent event) {
                dataAcquisition.checkDeviceReadiness();

                if (!dataAcquisition.isReady()) {
                    uiAcquireData.disableAll();
                } else {
                    uiAcquireData.enableAll();
                }
                if (dataAcquisition.detectorIsReady()) {
                    uiAcquireData.lbDetectorStatusValue.setText("Ready");
                } else {
                    uiAcquireData.lbDetectorStatusValue.setText("Not ready");
                }
                if (dataAcquisition.gpsIsReady()) {
                    uiAcquireData.lbGpsStatusValue.setText("Ready");
                } else {
                    uiAcquireData.lbGpsStatusValue.setText("Not ready");
                }
            }
        });

        uiAcquireData.btStartPause.setOnAction(new EventHandler<ActionEvent>() {
            @Override
            public void handle(ActionEvent event) {
                if (CheckInput.isValidName(uiAcquireData.tfName.getText())
```

```
                    & CheckInput.isValidDescription(uiAcquireData.tfDescription.getText())
                    & (boolean)uiAcquireData.btStartPause.getUserData()
                    & !dataAcquisition.isAcquiringData()) {

                uiAcquireData.btStartPause.setUserData(false);
                uiAcquireData.disableAllInput();
                uiAcquireData.btStartPause.setText("Pause Acquiring");

                dataAcquisition.setInterval(uiAcquireData.getInterval());
                dataAcquisition.startAcquiring();
//                  System.out.println("2 " + dataAcquisition.toString());
            } else if (dataAcquisition.isAcquiringData()) {
                dataAcquisition.pauseAcquiring();

                uiAcquireData.btStartPause.setUserData(true);
                uiAcquireData.btStartPause.setText("Start Acquiring");
            }
        }
    });

    uiAcquireData.btStopClose.setOnAction(new EventHandler<ActionEvent>() {
        @Override
        public void handle(ActionEvent event) {
            dataAcquisition.stop();

            if (dataAcquisition.getDoseDatas().size() > 0) {
                Date day = Date.valueOf(LocalDate.now());
                int did =
Integer.parseInt(ws.detectorList.get(uiAcquireData.cbDetector.getSelectionModel().getSelectedIndex()).getDetectorId()
);
                int gid =
Integer.parseInt(ws.gpsList.get(uiAcquireData.cbGps.getSelectionModel().getSelectedIndex()).getGpsId());

                connection.addDataSet(ws.dataSetList, dataAcquisition.getDoseDatas(), uiAcquireData.tfName.getText(),
uiAcquireData.tfDescription.getText(), day, uiAcquireData.tfPerson.getText(), day, uiAcquireData.tfPerson.getText(), did,
gid);
            }

            stage.close();
        }
    });

    stage.setOnCloseRequest(new EventHandler<WindowEvent>() {
        @Override
        public void handle(WindowEvent event) {
            dataAcquisition.stop();
        }
    });

  // Create add set stage
    Scene scene = new Scene(uiAcquireData.gp);
```

```
        stage.setScene(scene);
        stage.setTitle("Acquire Data");
    }

    public void setMapLinesCheckBox(CheckBox ch) {
        ch.selectedProperty().addListener(new ChangeListener<Boolean>() {
            @Override
            public void changed(ObservableValue cv, Boolean ov, Boolean nv) {
                if (ch.selectedProperty().getValue()) {
                    for (int i = 0; i < ((Polylines)ch.getUserData()).getPolylines().size(); i++){
                        ui.map.addMapShape(((Polylines)ch.getUserData()).getPolylines().get(i));
                    }
                    ws.mapPolylinesGroup.add((Polylines)ch.getUserData());
                } else {
                    for (int i = 0; i < ((Polylines)ch.getUserData()).getPolylines().size(); i++){
                        ui.map.removeMapShape(((Polylines)ch.getUserData()).getPolylines().get(i));
                    }
                    ws.mapPolylinesGroup.remove((Polylines)ch.getUserData());
                }
            }
        });

    }

    public void setMapMarkersCheckBox(CheckBox ch) {
        ch.selectedProperty().addListener(new ChangeListener<Boolean>() {
            @Override
            public void changed(ObservableValue cv, Boolean ov, Boolean nv) {
                if (ch.selectedProperty().getValue()) {
                    for (int i = 0; i < ((Markers)ch.getUserData()).getMarkers().size(); i++){
                        ui.map.addMarker(((Markers)ch.getUserData()).getMarkers().get(i));
                    }
                    ws.mapMarkersGroup.add((Markers)ch.getUserData());
                } else {
                    for (int i = 0; i < ((Markers)ch.getUserData()).getMarkers().size(); i++){
                        ui.map.removeMarker(((Markers)ch.getUserData()).getMarkers().get(i));
                    }
                    ws.mapMarkersGroup.remove((Markers)ch.getUserData());
                }
            }
        });
    }

    public void setMapCirclesCheckBox(CheckBox ch) {
        ch.selectedProperty().addListener(new ChangeListener<Boolean>() {
            @Override
            public void changed(ObservableValue cv, Boolean ov, Boolean nv) {
                if (ch.selectedProperty().getValue()) {
                    for (int i = 0; i < ((Circles)ch.getUserData()).getCircles().size(); i++){
                        ui.map.addMapShape(((Circles)ch.getUserData()).getCircles().get(i));
                    }
```

```
                ws.mapCirclesGroup.add((Circles)ch.getUserData());
            } else {
                for (int i = 0; i < ((Circles)ch.getUserData()).getCircles().size(); i++){
                    ui.map.removeMapShape(((Circles)ch.getUserData()).getCircles().get(i));
                }
                ws.mapCirclesGroup.remove((Circles)ch.getUserData());
            }
        }
    });
}

public void setMapPolygonsCheckBox(CheckBox ch) {
    ch.selectedProperty().addListener(new ChangeListener<Boolean>() {
        @Override
        public void changed(ObservableValue cv, Boolean ov, Boolean nv) {
            if (ch.selectedProperty().getValue()) {
                for (int i = 0; i < ((Polygons)ch.getUserData()).getPolygons().size(); i++){
                    ui.map.addMapShape(((Polygons)ch.getUserData()).getPolygons().get(i));
                }
                ws.mapPolygonsGroup.add((Polygons)ch.getUserData());
            } else {
                for (int i = 0; i < ((Polygons)ch.getUserData()).getPolygons().size(); i++){
                    ui.map.removeMapShape(((Polygons)ch.getUserData()).getPolygons().get(i));
                }
                ws.mapPolygonsGroup.remove((Polygons)ch.getUserData());
            }
        }
    });
}

public void setMapContourCheckBox(CheckBox ch) {
    ch.selectedProperty().addListener(new ChangeListener<Boolean>() {
        @Override
        public void changed(ObservableValue cv, Boolean ov, Boolean nv) {
            if (ch.selectedProperty().getValue()) {
                for (int i = 0; i < ((Contour)ch.getUserData()).getContourPolygons().size(); i++){
                    ui.map.addMapShape(((Contour)ch.getUserData()).getContourPolygons().get(i));
                }
                for (int i = 0; i < ((Contour)ch.getUserData()).getContourPolylines().size(); i++){
                    ui.map.addMapShape(((Contour)ch.getUserData()).getContourPolylines().get(i));
                }
                ws.mapContourGroup.add((Contour)ch.getUserData());
            } else {
                for (int i = 0; i < ((Contour)ch.getUserData()).getContourPolygons().size(); i++){
                    ui.map.removeMapShape(((Contour)ch.getUserData()).getContourPolygons().get(i));
                }
                for (int i = 0; i < ((Contour)ch.getUserData()).getContourPolylines().size(); i++){
                    ui.map.removeMapShape(((Contour)ch.getUserData()).getContourPolylines().get(i));
                }
                ws.mapContourGroup.remove((Contour)ch.getUserData());
            }
```

```
            }
        });
    }

    public String getColor(double val) {
        double range = Config.DOSE_RATE_RANGE;
        String color = "";

        if (val <= range) {
            color = "rgb(0,0,255)";
        } else if (val <= (range * 2)) {
            color = "rgb(0,255,255)";
        } else if (val <= (range * 3)) {
            color = "rgb(0,255,0)";
        } else if (val <= (range * 4)) {
            color = "rgb(255,255,0)";
        } else if (val <= (range * 5)) {
            color = "rgb(255,128,0)";
        } else if (val <= (range * 6)) {
            color = "rgb(255,0,0)";
        } else {
            color = "rgb(128,0,0)";
        }

        return color;
    }

    public String getColor(String val) {
        String color = "";

        if (val.equals("level1")) {
            color = "rgb(0,0,255)";
        } else if (val.equals("level2")) {
            color = "rgb(0,255,255)";
        } else if (val.equals("level3")) {
            color = "rgb(0,255,0)";
        } else if (val.equals("level4")) {
            color = "rgb(255,255,0)";
        } else if (val.equals("level5")) {
            color = "rgb(255,128,0)";
        } else if (val.equals("level6")) {
            color = "rgb(255,0,0)";
        } else {
            color = "rgb(128,0,0)";
        }

        return color;
    }
}
```

13. dosemapping > Gps.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package dosemapping;

import javafx.beans.property.SimpleStringProperty;

/**
 *
 * @author Administrator
 */
public class Gps {
    private final SimpleStringProperty gpsId;
    private final SimpleStringProperty gpsName;
    private final SimpleStringProperty gpsDescription;
    private final SimpleStringProperty gpsManufacturer;

    public Gps(String idVal, String nameVal, String descriptionVal, String manufacturerVal) {
        this.gpsId = new SimpleStringProperty(idVal);
        this.gpsName = new SimpleStringProperty(nameVal);
        this.gpsDescription = new SimpleStringProperty(descriptionVal);
        this.gpsManufacturer = new SimpleStringProperty(manufacturerVal);
    }

    public String getGpsId() {
        return gpsId.get();
    }

    public String getGpsName() {
        return gpsName.get();
    }

    public String getGpsDescription() {
        return gpsDescription.get();
    }

    public String getGpsManufacturer() {
        return gpsManufacturer.get();
    }
}
```

14. dosemapping > SQLConnection.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package dosemapping;
```

```
import config.Config;
import java.sql.Connection;
import java.sql.Date;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import javafx.collections.ObservableList;

/**
 *
 * @author ppengvan
 */
public class SQLConnection {
    private Connection connection = null;
    private PreparedStatement query = null;
    private String queryStatement = null;
    private ResultSet queryResult = null;

    private List<String> dbTables = new ArrayList<>();
    private List<String> dataSets = new ArrayList<>();

    boolean detectorTableExist = false;
    boolean gpsTableExist = false;
    boolean dataSetTableExist = false;
    boolean doseTableExist = false;

//    ObservableList<DataSet> dataSetList = FXCollections.observableArrayList();
//    ObservableList<String> items =FXCollections.observableArrayList();

    public SQLConnection() {
        try {
            establishSQLConnection();
            createSQLDatabase();
        } catch (SQLException ex) {
            Logger.getLogger(SQLConnection.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    public void establishSQLConnection() {
        System.out.print("Establish database connection...");
        try {
            Class.forName("com.mysql.jdbc.Driver");
            String connectionStatement = Config.CONNECTION + "&user=" + Config.USERNAME + "&password=" +
Config.PASSWORD;
//            System.out.println(connectionStatement);
```

```
            connection = DriverManager.getConnection(connectionStatement);
        }
        catch (Exception ex) {
            System.out.println("Error with database connection.");
        }
        System.out.println("Done.");
    }

    public void testSQLConnection() throws SQLException {
        if (connection == null) {
            System.out.println("Please establish SQL connection first.");
        }
        else {
//            query = connection.prepareStatement("insert into dosemap.DataSet values (default,?,?)");
//            query.setString(1, "test");
//            query.setString(2, "this is a test");
//            query.executeUpdate();
        }
    }

    public void createSQLDatabase() throws SQLException {
        System.out.println("Check/create database:");
        if (connection == null) {
            System.out.println("Please establish SQL connection first.");
        }
        else {
            dbTables.clear();
            queryResult = connection.getMetaData().getTables(null, null, "%", null);
            while (queryResult.next()) {
                dbTables.add(queryResult.getString(3));
            }
            for (int i = 0; i < dbTables.size(); i++) {
                switch (dbTables.get(i)) {
                    case "detector":    detectorTableExist = true;
                                        break;
                    case "gps":         gpsTableExist = true;
                                        break;
                    case "dataset":     dataSetTableExist = true;
                                        break;
                    case "dose":        doseTableExist = true;
                                        break;
                }
            }

            createDetectorTable();
            createGpsTable();
            createDataSetTable();
            createDoseTable();
        }
    }
```

```
    private void createDetectorTable() throws SQLException {
        System.out.print("     Checking if Detector table already existed...");
        if (detectorTableExist) {
            System.out.println("Detector table already existed.");
        }
        else {
            queryStatement = "CREATE TABLE detector ("
                    + "DetectorID SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY, "
                    + "DetectorName VARCHAR(30), "
                    + "DetectorDescription VARCHAR(255), "
                    + "DetectorManufacturer VARCHAR(30)"
                    + ")";
            query = connection.prepareStatement(queryStatement);
            query.executeUpdate();

            queryStatement = "INSERT INTO detector (DetectorName, DetectorDescription, DetectorManufacturer)
values (?, ?, ?)";
            query = connection.prepareStatement(queryStatement);
            query.setString(1, "Dummy detector");
            query.setString(2, "Dummy description");
            query.setString(3, "Dummy manufacturer");
            query.execute();

            System.out.println("Detector table created.");
        }
    }

    private void createGpsTable() throws SQLException {
        System.out.print("     Checking if GPS table already existed...");
        if (gpsTableExist) {
            System.out.println("GPS table already existed.");
        }
        else {
            queryStatement = "CREATE TABLE gps ("
                    + "GpsID SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY, "
                    + "GpsName VARCHAR(30), "
                    + "GpsDescription VARCHAR(255), "
                    + "GpsManufacturer VARCHAR(30)"
                    + ")";
            query = connection.prepareStatement(queryStatement);
            query.executeUpdate();

            queryStatement = "INSERT INTO gps (GpsName, GpsDescription, GpsManufacturer) values (?, ?, ?)";
            query = connection.prepareStatement(queryStatement);
            query.setString(1, "Dummy GPS");
            query.setString(2, "Dummy description");
            query.setString(3, "Dummy manufacturer");
            query.execute();

            System.out.println("GPS table created.");
        }
```

```
    }

    private void createDataSetTable() throws SQLException {
        System.out.print("     Checking if DataSet table already existed...");
        if (dataSetTableExist) {
            System.out.println("DataSet table already existed.");
        }
        else {
            queryStatement = "CREATE TABLE dataset ("
                    + "DataSetID SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY, "
                    + "SetName VARCHAR(30), "
                    + "SetDescription VARCHAR(255), "
                    + "PersonAcquire VARCHAR(30), "
                    + "DateUpload DATE, "
                    + "PersonUpload VARCHAR(30), "
                    + "DetectorID SMALLINT UNSIGNED,"
                    + "GpsID SMALLINT UNSIGNED,"
                    + "FOREIGN KEY (DetectorID) "
                    + "REFERENCES detector(DetectorID) "
                    + "ON UPDATE CASCADE "
                    + "ON DELETE RESTRICT,"
                    + "FOREIGN KEY (GpsID) "
                    + "REFERENCES gps(GpsID) "
                    + "ON UPDATE CASCADE "
                    + "ON DELETE RESTRICT"
                    + ")";
            query = connection.prepareStatement(queryStatement);
            query.executeUpdate();
            System.out.println("DataSet table created.");
        }
    }

    private void createDoseTable() throws SQLException {
        System.out.print("     Checking if Dose table already existed...");
        if (doseTableExist) {
            System.out.println("Dose table already existed.");
        }
        else {
            queryStatement = "CREATE TABLE dose ("
                    + "DoseID SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY, "
                    + "Date DATE, "
                    + "Time TIME, "
                    + "Latitude DOUBLE, "
                    + "Longitude DOUBLE, "
                    + "Dose DOUBLE, "
                    + "DataSetID SMALLINT UNSIGNED,"
                    + "FOREIGN KEY (DataSetID) "
                    + "REFERENCES dataset(DataSetID) "
                    + "ON UPDATE CASCADE "
                    + "ON DELETE RESTRICT"
                    + ")";
```

```
            query = connection.prepareStatement(queryStatement);
            query.executeUpdate();
            System.out.println("Dose table created.");
        }
    }


    public Connection getSQLConnection() {
        return connection;
    }

//    public void getListDataSets(ListView<String> list) throws SQLException {
//        queryStatement = "SELECT Name FROM dataset";
//        query = connection.prepareStatement(queryStatement);
//        queryResult = query.executeQuery();
//
//        while (queryResult.next()) {
//            items.add(queryResult.getString(1));
//        }
//        list.setItems(items);
//    }

    public void getTableDetectors(ObservableList<Detector> detectorList) throws SQLException {
        Detector dat;

        queryStatement = "SELECT * FROM detector";
        query = connection.prepareStatement(queryStatement);
        queryResult = query.executeQuery();

        while (queryResult.next()) {
            dat = new Detector(Integer.toString(queryResult.getInt(1)), queryResult.getString(2), queryResult.getString(3),
queryResult.getString(4));
            detectorList.add(dat);
        }
    }

    public void getTableGpss(ObservableList<Gps> gpsList) throws SQLException {
        Gps dat;

        queryStatement = "SELECT * FROM gps";
        query = connection.prepareStatement(queryStatement);
        queryResult = query.executeQuery();

        while (queryResult.next()) {
            dat = new Gps(Integer.toString(queryResult.getInt(1)), queryResult.getString(2), queryResult.getString(3),
queryResult.getString(4));
            gpsList.add(dat);
        }
    }

    public void getTableDataSets(ObservableList<DataSet> dataSetList) throws SQLException {
```

```
        DataSet dat;

        queryStatement = "SELECT * FROM dataset";
        query = connection.prepareStatement(queryStatement);
        queryResult = query.executeQuery();
        while (queryResult.next()) {
            dat = new DataSet(Integer.toString(queryResult.getInt(1)), queryResult.getString(2), queryResult.getString(3),
queryResult.getString(4), queryResult.getDate(5).toString(), queryResult.getString(6),
Integer.toString(queryResult.getInt(7)),  Integer.toString(queryResult.getInt(8)));
            dataSetList.add(dat);
        }
    }

    public void addDetector(ObservableList<Detector> detectorList, String detectorName, String detectorDescription,
String detectorManufacturer) {
        int id = 0;

        try {
            queryStatement = "INSERT INTO detector (DetectorName, DetectorDescription, DetectorManufacturer)
values (?, ?, ?)";
            query = connection.prepareStatement(queryStatement);
            query.setString(1, detectorName);
            query.setString(2, detectorDescription);
            query.setString(3, detectorManufacturer);
            query.execute();

            queryStatement = "SELECT LAST_INSERT_ID()";
            query = connection.prepareStatement(queryStatement);
            queryResult = query.executeQuery();
            while (queryResult.next()) {
                id = queryResult.getInt(1);
                Detector dat = new Detector(Integer.toString(id), detectorName, detectorDescription,
detectorManufacturer);
                detectorList.add(dat);
            }
        } catch (SQLException e) {
            System.out.println("Cannot add the data provided.");
        }
    }

    public void addGps(ObservableList<Gps> gpsList, String gpsName, String gpsDescription, String gpsManufacturer) {
        int id = 0;

        try {
            queryStatement = "INSERT INTO gps (GpsName, GpsDescription, GpsManufacturer) values (?, ?, ?)";
            query = connection.prepareStatement(queryStatement);
            query.setString(1, gpsName);
            query.setString(2, gpsDescription);
            query.setString(3, gpsManufacturer);
            query.execute();
```

```
            queryStatement = "SELECT LAST_INSERT_ID()";
            query = connection.prepareStatement(queryStatement);
            queryResult = query.executeQuery();
            while (queryResult.next()) {
                id = queryResult.getInt(1);
                Gps dat = new Gps(Integer.toString(id), gpsName, gpsDescription, gpsManufacturer);
                gpsList.add(dat);
            }
        } catch (SQLException e) {
            System.out.println("Cannot add the data provided.");
        }
    }

    public void addDataSet(ObservableList<DataSet> dataSetList, List<DoseData> doseData, String setName, String setDescription, Date dateAcquire, String personAcquire, Date dateUpload, String personUpload, int detectorID, int gpsID) {
        int id = 0;

        try {
            queryStatement = "INSERT INTO dataset (SetName, SetDescription, PersonAcquire, DateUpload, PersonUpload, DetectorID, GpsID) values (?, ?, ?, ?, ?, ?, ?)";
            query = connection.prepareStatement(queryStatement);
            query.setString(1, setName);
            query.setString(2, setDescription);
//            query.setDate(3, dateAcquire);
            query.setString(3, personAcquire);
            query.setDate(4, dateUpload);
            query.setString(5, personUpload);
            query.setInt(6, detectorID);
            query.setInt(7, gpsID);
            query.execute();

            queryStatement = "SELECT LAST_INSERT_ID()";
            query = connection.prepareStatement(queryStatement);
            queryResult = query.executeQuery();
            while (queryResult.next()) {
                id = queryResult.getInt(1);
                DataSet dat = new DataSet(Integer.toString(id), setName, setDescription, personAcquire, dateUpload.toString(), personUpload, Integer.toString(detectorID), Integer.toString(gpsID));
                dataSetList.add(dat);
            }

            for (int i = 0; i < doseData.size(); i++) {
                queryStatement = "INSERT INTO dose (Date, Time, Latitude, Longitude, Dose, DataSetID) values (?, ?, ?, ?, ?, ?)";
                query = connection.prepareStatement(queryStatement);
                query.setDate(1, doseData.get(i).getDate());
                query.setTime(2, doseData.get(i).getTime());
                query.setDouble(3, doseData.get(i).getLat());
                query.setDouble(4, doseData.get(i).getLon());
                query.setDouble(5, doseData.get(i).getDose());
```

```
                query.setInt(6, id);
                query.execute();
            }
        } catch (SQLException e) {
            System.out.println("Cannot add the data provided.");
        }
    }

    public void deleteDataSet(ObservableList<DataSet> dataSetList, DataSet set) {
        try {
            queryStatement = "DELETE FROM dose WHERE DataSetID=" + set.getSetId();
            query = connection.prepareStatement(queryStatement);
            query.execute();

            queryStatement = "DELETE FROM dataset WHERE DataSetID=" + set.getSetId();
            query = connection.prepareStatement(queryStatement);
            query.execute();

            int found = -1;
            for (int i = 0; i < dataSetList.size(); i++) {
                if (dataSetList.get(i).getSetId() == set.getSetId()) {
                    found = i;
                    break;
                }
            }

            dataSetList.remove(found);
        } catch (SQLException e) {
            System.out.println("Cannot delete the data selected.");
        }
    }

    public void deleteDetector(ObservableList<Detector> detectorList, Detector set) {
        try {
            queryStatement = "DELETE FROM detector WHERE DetectorID=" + set.getDetectorId();
            query = connection.prepareStatement(queryStatement);
            query.execute();

            int found = -1;
            for (int i = 0; i < detectorList.size(); i++) {
                if (detectorList.get(i).getDetectorId() == set.getDetectorId()) {
                    found = i;
                    break;
                }
            }

            detectorList.remove(found);
        } catch (SQLException e) {
            System.out.println("Cannot delete the data selected.");
        }
    }
```

```
    public void deleteGps(ObservableList<Gps> gpsList, Gps set) {
        try {
            queryStatement = "DELETE FROM gps WHERE GpsID=" + set.getGpsId();
            query = connection.prepareStatement(queryStatement);
            query.execute();

            int found = -1;
            for (int i = 0; i < gpsList.size(); i++) {
                if (gpsList.get(i).getGpsId() == set.getGpsId()) {
                    found = i;
                    break;
                }
            }

            gpsList.remove(found);
        } catch (SQLException e) {
            System.out.println("Cannot delete the data selected.");
        }
    }

    public void getDoseData(List<DoseData> doseData, DataSet set) {
        try {
            queryStatement = "SELECT * FROM dose WHERE DataSetID=" + set.getSetId() + " ORDER BY Date, Time";
            query = connection.prepareStatement(queryStatement);
            queryResult = query.executeQuery();
            while (queryResult.next()) {
                DoseData dose = new DoseData(queryResult.getDate(2), queryResult.getTime(3),
queryResult.getDouble(4), queryResult.getDouble(5), queryResult.getDouble(6));
                doseData.add(dose);
            }
        } catch (SQLException ex) {
            Logger.getLogger(SQLConnection.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    public Detector getDetectorByID(ObservableList<Detector> detectorList, String id) {
        for (Detector detector : detectorList) {
            if (detector.getDetectorId().equals(id)) {
                return detector;
            }
        }
        return null;
    }

    public Gps getGpsByID(ObservableList<Gps> gpsList, String id) {
        for (Gps gps : gpsList) {
            if (gps.getGpsId().equals(id)) {
                return gps;
            }
        }
```

```
            return null;
        }
    }
```

15. dosemapping > UIAcquireData.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package dosemapping;

import java.sql.Date;
import java.time.LocalDate;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.geometry.Insets;
import javafx.scene.control.Button;
import javafx.scene.control.ComboBox;
import javafx.scene.control.Label;
import javafx.scene.control.TextArea;
import javafx.scene.control.TextField;
import javafx.scene.layout.GridPane;

/**
 *
 * @author Administrator
 */
public class UIAcquireData {

    protected GridPane gp = new GridPane();

    protected Label lbDetectorStatus = new Label("Detector Status");
    protected Label lbDetectorStatusValue = new Label("Not ready");

    protected Label lbGpsStatus = new Label("GPS Status");
    protected Label lbGpsStatusValue = new Label("Not ready");

    protected Button btRecheckStatus = new Button("Recheck Status");

    protected Label lbName = new Label("Set Name");
    protected TextField tfName = new TextField();

    protected Label lbDescription = new Label("Set Description");
    protected TextField tfDescription = new TextField();

    protected Label lbDate = new Label("Acquire Date");
    protected Label lbDateValue = new Label(Date.valueOf(LocalDate.now()).toString());

    protected Label lbInterval = new Label("Acquire Interval");
```

```
    protected ComboBox cbInterval = new ComboBox();

    protected Label lbPerson= new Label("Acquire Person");
    protected TextField tfPerson = new TextField();

    protected Label lbSelectDetector = new Label("Select Detector");
    protected ComboBox cbDetector = new ComboBox();

    protected Label lbSelectGps = new Label("Select GPS");
    protected ComboBox cbGps = new ComboBox();

    protected Button btStartPause = new Button("Start Acquiring");
    protected Button btStopClose = new Button("Stop & Close");

    protected TextArea taData = new TextArea();

    protected ObservableList<String> interval = FXCollections.observableArrayList("5 sec", "10 sec", "15 sec", "20 sec",
"30 sec");
    protected ObservableList<String> detector = FXCollections.observableArrayList();
    protected ObservableList<String> gps = FXCollections.observableArrayList();

    public UIAcquireData(WorkingSpace ws) {

    // Set UI properties
        gp.setPadding(new Insets(10));
        gp.setHgap(10);
        gp.setVgap(10);

        lbDetectorStatus.setMinHeight(30);
        lbDetectorStatusValue.setMinWidth(100);

        lbGpsStatus.setMinHeight(30);
        lbGpsStatusValue.setMinWidth(100);

        btRecheckStatus.setMinWidth(100);
        btRecheckStatus.setMinHeight(30);

        lbName.setMinHeight(30);
        tfName.setMinWidth(200);

        lbDescription.setMinHeight(30);
        tfDescription.setMinWidth(200);

        lbDate.setMinHeight(30);
        lbDateValue.setMinWidth(200);

        lbInterval.setMinHeight(30);
        cbInterval.setMinWidth(200);
        cbInterval.setItems(interval);
        cbInterval.setValue(interval.get(0));
```

```
        lbPerson.setMinHeight(30);
        tfPerson.setMinWidth(200);

        for (Detector dt : ws.detectorList) {
            detector.add(dt.getDetectorName());
        }

        lbSelectDetector.setMinHeight(30);
        cbDetector.setMinWidth(200);
        cbDetector.setItems(detector);
        cbDetector.setValue(detector.get(0));

        for (Gps dt : ws.gpsList) {
            gps.add(dt.getGpsName());
        }

        lbSelectGps.setMinHeight(30);
        cbGps.setMinWidth(200);
        cbGps.setItems(gps);
        cbGps.setValue(gps.get(0));

        btStartPause.setMinWidth(100);
        btStartPause.setMinHeight(30);
        btStartPause.setUserData(true);

        btStopClose.setMinWidth(100);
        btStopClose.setMinHeight(30);

        taData.setMinWidth(200);
        taData.setMinHeight(150);
        taData.setEditable(false);

    // UI construction
        gp.add(lbDetectorStatus, 0, 0);
        gp.add(lbDetectorStatusValue, 1, 0);
        gp.add(lbGpsStatus, 0, 1);
        gp.add(lbGpsStatusValue, 1, 1);
        gp.add(btRecheckStatus, 1, 2);
        gp.add(lbName, 2, 0);
        gp.add(tfName, 3, 0);
        gp.add(lbDescription, 2, 1);
        gp.add(tfDescription, 3, 1);
        gp.add(lbDate, 2, 2);
        gp.add(lbDateValue, 3, 2);
        gp.add(lbInterval, 2, 3);
        gp.add(cbInterval, 3, 3);
        gp.add(lbPerson, 2, 4);
        gp.add(tfPerson, 3, 4);
        gp.add(lbSelectDetector, 2, 5);
        gp.add(cbDetector, 3, 5);
        gp.add(lbSelectGps, 2, 6);
```

```
        gp.add(cbGps, 3, 6);
        gp.add(btStartPause, 2, 7);
        gp.add(btStopClose, 3, 7);
        gp.add(taData, 4, 0, 1, 8);
    }

    public void disableAllInput() {
        tfName.setDisable(true);
        tfDescription.setDisable(true);
        cbInterval.setDisable(true);
        cbDetector.setDisable(true);
        cbGps.setDisable(true);
        tfPerson.setDisable(true);
    }

    public void disableAll() {
        tfName.setDisable(true);
        tfDescription.setDisable(true);
        cbInterval.setDisable(true);
        cbDetector.setDisable(false);
        cbGps.setDisable(false);
        tfPerson.setDisable(true);
        btStartPause.setDisable(true);
    }

    public void enableAll() {
        tfName.setDisable(false);
        tfDescription.setDisable(false);
        cbInterval.setDisable(false);
        cbDetector.setDisable(false);
        cbGps.setDisable(false);
        tfPerson.setDisable(false);
        btStartPause.setDisable(false);
    }

    public int getInterval() {
        String s = cbInterval.getValue().toString();
        int interval = 0;

        switch (s) {
            case "5 sec"    :   interval = 5000;
                                break;
            case "10 sec"   :   interval = 10000;
                                break;
            case "15 sec"   :   interval = 15000;
                                break;
            case "20 sec"   :   interval = 20000;
                                break;
            case "30 sec"   :   interval = 30000;
                                break;
        }
```

```
        return interval;
    }
}
```

16. dosemapping > UIAddDetector.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package dosemapping;

import javafx.geometry.Insets;
import javafx.scene.control.Button;
import javafx.scene.control.DatePicker;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.GridPane;

/**
 *
 * @author ppengvan
 */
public class UIAddDetector {
    protected GridPane gp = new GridPane();

    protected Label lbName = new Label("Name");
    protected TextField tfName = new TextField();

    protected Label lbDescription = new Label("Description");
    protected TextField tfDescription = new TextField();

    protected Label lbManufacturer = new Label("Manufacturer");
    protected TextField tfManufacturer = new TextField();

    protected Button btCancel = new Button("Cancel");
    protected Button btAdd = new Button("Add Detector");

    public UIAddDetector() {

    // Set UI properties
        gp.setPadding(new Insets(10));
        gp.setHgap(10);
        gp.setVgap(10);

        lbName.setMinHeight(30);
        tfName.setMinWidth(200);

        lbDescription.setMinHeight(30);
```

```
        tfDescription.setMinWidth(200);

        lbManufacturer.setMinHeight(30);
        tfManufacturer.setMinWidth(200);

        btCancel.setMinWidth(100);
        btCancel.setMinHeight(30);

        btAdd.setMinWidth(100);
        btAdd.setMinHeight(30);

    // UI construction
        gp.add(lbName, 0, 0);
        gp.add(tfName, 1, 0);
        gp.add(lbDescription, 0, 1);
        gp.add(tfDescription, 1, 1);
        gp.add(lbManufacturer, 0, 2);
        gp.add(tfManufacturer, 1, 2);
        gp.add(btCancel, 0, 3);
        gp.add(btAdd, 1, 3);
    }
}
```

17. dosemapping > UIAddGps.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package dosemapping;

import javafx.geometry.Insets;
import javafx.scene.control.Button;
import javafx.scene.control.DatePicker;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.GridPane;

/**
 *
 * @author ppengvan
 */
public class UIAddGps {
    protected GridPane gp = new GridPane();

    protected Label lbName = new Label("Name");
    protected TextField tfName = new TextField();

    protected Label lbDescription = new Label("Description");
    protected TextField tfDescription = new TextField();
```

```
    protected Label lbManufacturer = new Label("Manufacturer");
    protected TextField tfManufacturer = new TextField();

    protected Button btCancel = new Button("Cancel");
    protected Button btAdd = new Button("Add GPS");

    public UIAddGps() {

    // Set UI properties
        gp.setPadding(new Insets(10));
        gp.setHgap(10);
        gp.setVgap(10);

        lbName.setMinHeight(30);
        tfName.setMinWidth(200);

        lbDescription.setMinHeight(30);
        tfDescription.setMinWidth(200);

        lbManufacturer.setMinHeight(30);
        tfManufacturer.setMinWidth(200);

        btCancel.setMinWidth(100);
        btCancel.setMinHeight(30);

        btAdd.setMinWidth(100);
        btAdd.setMinHeight(30);

    // UI construction
        gp.add(lbName, 0, 0);
        gp.add(tfName, 1, 0);
        gp.add(lbDescription, 0, 1);
        gp.add(tfDescription, 1, 1);
        gp.add(lbManufacturer, 0, 2);
        gp.add(tfManufacturer, 1, 2);
        gp.add(btCancel, 0, 3);
        gp.add(btAdd, 1, 3);
    }
}
```

18. dosemapping > UIImportData.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package dosemapping;

import java.sql.Date;
```

```
import java.time.LocalDate;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.geometry.Insets;
import javafx.scene.control.Button;
import javafx.scene.control.ComboBox;
import javafx.scene.control.DatePicker;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.GridPane;

/**
 *
 * @author Administrator
 */
public class UIImportData {

    protected GridPane gp = new GridPane();

    protected Label lbSetName = new Label("Set Name");
    protected TextField tfSetName = new TextField();

    protected Label lbSetDescription = new Label("Set Description");
    protected TextField tfSetDescription = new TextField();

    protected Label lbAcquireDate = new Label("Acquire Date");
    protected DatePicker dpAcquireDate = new DatePicker();

    protected Label lbAcquirePerson = new Label("Acquire Person");
    protected TextField tfAcquirePerson = new TextField();

    protected Label lbUploadDate = new Label("Upload Date");
    protected Label lbUploadDateValue = new Label(Date.valueOf(LocalDate.now()).toString());
//    protected DatePicker dpUploadDate = new DatePicker();

    protected Label lbUploadPerson = new Label("Upload Person");
    protected TextField tfUploadPerson = new TextField();

    protected Button btSelectDataFile = new Button("Select File");
    protected Label lbFileName = new Label("");

    protected Label lbSelectDetector = new Label("Select Detector");
    protected ComboBox cbDetector = new ComboBox();

    protected Label lbSelectGps = new Label("Select GPS");
    protected ComboBox cbGps = new ComboBox();

    protected Button btCancel = new Button("Cancel");
    protected Button btImportSet = new Button("Import Data");

    protected ObservableList<String> detector = FXCollections.observableArrayList();
```

```
    protected ObservableList<String> gps = FXCollections.observableArrayList();

    public UIImportData(WorkingSpace ws) {

    // Set UI properties
        gp.setPadding(new Insets(10));
        gp.setHgap(10);
        gp.setVgap(10);

        lbSetName.setMinHeight(30);
        tfSetName.setMinWidth(200);

        lbSetDescription.setMinHeight(30);
        tfSetDescription.setMinWidth(200);

        lbAcquireDate.setMinHeight(30);
        dpAcquireDate.setMinWidth(200);

        lbAcquirePerson.setMinHeight(30);
        tfAcquirePerson.setMinWidth(200);

        lbUploadDate.setMinHeight(30);
//        dpUploadDate.setMinWidth(200);
        lbUploadDateValue.setMinWidth(200);

        lbUploadPerson.setMinHeight(30);
        tfUploadPerson.setMinWidth(200);

        lbFileName.setMinWidth(200);
        btSelectDataFile.setMinWidth(100);
        btSelectDataFile.setMinHeight(30);

        for (Detector dt : ws.detectorList) {
            detector.add(dt.getDetectorName());
        }

        lbSelectDetector.setMinHeight(30);
        cbDetector.setMinWidth(200);
        cbDetector.setItems(detector);
        cbDetector.setValue(detector.get(0));

        for (Gps dt : ws.gpsList) {
            gps.add(dt.getGpsName());
        }

        lbSelectGps.setMinHeight(30);
        cbGps.setMinWidth(200);
        cbGps.setItems(gps);
        cbGps.setValue(gps.get(0));

        btCancel.setMinWidth(100);
```

```
        btCancel.setMinHeight(30);

        btImportSet.setMinWidth(100);
        btImportSet.setMinHeight(30);

    // UI construction
        gp.add(lbSetName, 0, 0);
        gp.add(tfSetName, 1, 0);
        gp.add(lbSetDescription, 0, 1);
        gp.add(tfSetDescription, 1, 1);
        gp.add(lbAcquireDate, 0, 2);
        gp.add(dpAcquireDate, 1, 2);
        gp.add(lbAcquirePerson, 0, 3);
        gp.add(tfAcquirePerson, 1, 3);
        gp.add(lbUploadDate, 0, 4);
//        gp.add(dpUploadDate, 1, 4);
        gp.add(lbUploadDateValue, 1, 4);
        gp.add(lbUploadPerson, 0, 5);
        gp.add(tfUploadPerson, 1, 5);
        gp.add(btSelectDataFile, 0, 6);
        gp.add(lbFileName, 1, 6);
        gp.add(lbSelectDetector, 0, 7);
        gp.add(cbDetector, 1, 7);
        gp.add(lbSelectGps, 0, 8);
        gp.add(cbGps, 1, 8);
        gp.add(btCancel, 0, 9);
        gp.add(btImportSet, 1, 9);
    }
}
```

19. dosemapping > UIMapLabel.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package dosemapping;

import config.Config;
import javafx.geometry.Pos;
import javafx.scene.control.Label;
import javafx.scene.layout.StackPane;
import javafx.scene.paint.Color;
import javafx.scene.shape.Rectangle;

/**
 *
 * @author ppengvan
 */
public class UIMapLabel extends StackPane {
```

```
    protected Rectangle bg = new Rectangle(160, 165);
    protected Rectangle[] r = new Rectangle[7];
    protected Label[] l = new Label[7];

    protected Label lt = new Label();
//    protected Label l1 = new Label();
//    protected Label l2 = new Label();

    public UIMapLabel() {
        super();
        super.setMinSize(100, 165);
        super.setMaxSize(160, 165);
        super.setAlignment(Pos.TOP_LEFT);

        bg.setFill(Color.WHITE);
        bg.setOpacity(0);

        for (int i = 0; i < 7; i++) {
            r[i] = new Rectangle(15, 15);
            r[i].setTranslateX(15);
            r[i].setTranslateY(45+(i*15));
            r[i].setFill(Color.BLUE);
            r[i].setOpacity(0);

            l[i] = new Label();
            l[i].setTranslateX(45);
            l[i].setTranslateY(45+(i*15));
            l[i].setTextFill(Color.BLACK);
            l[i].setOpacity(0);
        }

        lt.setTranslateX(15);
        lt.setTranslateY(15);
        lt.setTextFill(Color.BLACK);

//        l1.setTranslateX(45);
//        l1.setTranslateY(45);
//        l1.setTextFill(Color.BLACK);

//        l2.setTranslateX(45);
//        l2.setTranslateY(180);
//        l2.setTextFill(Color.BLACK);

//        super.getChildren().addAll(bg, lt, l1, l2);
        super.getChildren().addAll(bg, lt);

        for (int i = 0; i < 7; i++) {
            super.getChildren().add(r[i]);
            super.getChildren().add(l[i]);
        }
    }
```

```
    public void setLabel(String type, String name, double min, double max) {
        String color = "";
        if (type == "markers") {
            hideLabel();
        } else {
            bg.setOpacity(0.9);

            for (int i = 0; i < 7; i++) {
                int bcolor = (int) ((6-i)/6.0*255);
                int rgcolor = 255 - bcolor;

                switch (type) {
                    case "polylines" :  color = "rgb(255," + rgcolor + "," + rgcolor + ")";
                                        break;
//                    case "circles"   :  color = "rgb(" + rgcolor + ",255," + rgcolor + ")";
//                                        break;
                    case "polygons"  :  color = "rgb(" + rgcolor + "," + rgcolor + ",255)";
                                        break;
                    case "contour"   :  color = "rgb(" + rgcolor + ",255," + rgcolor + ")";
                                        break;
                }

                r[i].setFill(Color.valueOf(color));
                r[i].setOpacity(1);
                l[i].setOpacity(1);
            }
            lt.setText("Dose Rate (nGy/hr)");
            l[0].setText(String.format("%.2f", max));
            l[1].setText(String.format("%.2f", min + (5.0/6)*(max-min)));
            l[2].setText(String.format("%.2f", min + (4.0/6)*(max-min)));
            l[3].setText(String.format("%.2f", min + (3.0/6)*(max-min)));
            l[4].setText(String.format("%.2f", min + (2.0/6)*(max-min)));
            l[5].setText(String.format("%.2f", min + (1.0/6)*(max-min)));
            l[6].setText(String.format("%.2f", min));
//            l1.setText(Double.toString(max));
//            l2.setText(Double.toString(min));
        }
    }

    public void setLabel(String type) {
        double range = Config.DOSE_RATE_RANGE;
        if (type == "markers") {
            bg.setOpacity(0);

            for (int i = 0; i < 7; i++) {
                r[i].setOpacity(0);
            }
            lt.setText("");
//            l1.setText("");
//            l2.setText("");
```

```
        } else {
            bg.setOpacity(0.9);
            lt.setText("Dose Rate (nGy/hr)");
            r[0].setFill(Color.valueOf("rgb(128,0,0)"));
            r[1].setFill(Color.valueOf("rgb(255,0,0)"));
            r[2].setFill(Color.valueOf("rgb(255,128,0)"));
            r[3].setFill(Color.valueOf("rgb(255,255,0)"));
            r[4].setFill(Color.valueOf("rgb(0,255,0)"));
            r[5].setFill(Color.valueOf("rgb(0,255,255)"));
            r[6].setFill(Color.valueOf("rgb(0,0,255)"));
            l[0].setText(Double.toString(range*6) + " and above");
            l[1].setText(Double.toString(range*5) + " - " + Double.toString(range*6));
            l[2].setText(Double.toString(range*4) + " - " + Double.toString(range*5));
            l[3].setText(Double.toString(range*3) + " - " + Double.toString(range*4));
            l[4].setText(Double.toString(range*2) + " - " + Double.toString(range*3));
            l[5].setText(Double.toString(range*1) + " - " + Double.toString(range*2));
            l[6].setText(Double.toString(range*1) + " and below");

            for (int i = 0; i < 7; i++) {
                r[i].setOpacity(1);
                l[i].setOpacity(1);
            }

        }
    }

    public void setLabel(String[] label) {
        bg.setOpacity(0.9);
        lt.setText("Dose Rate (nGy/hr)");
        r[0].setFill(Color.valueOf("rgb(128,0,0)"));
        r[1].setFill(Color.valueOf("rgb(255,0,0)"));
        r[2].setFill(Color.valueOf("rgb(255,128,0)"));
        r[3].setFill(Color.valueOf("rgb(255,255,0)"));
        r[4].setFill(Color.valueOf("rgb(0,255,0)"));
        r[5].setFill(Color.valueOf("rgb(0,255,255)"));
        r[6].setFill(Color.valueOf("rgb(0,0,255)"));
        l[0].setText(label[6]);
        l[1].setText(label[5]);
        l[2].setText(label[4]);
        l[3].setText(label[3]);
        l[4].setText(label[2]);
        l[5].setText(label[1]);
        l[6].setText(label[0]);

        for (int i = 0; i < 7; i++) {
            r[i].setOpacity(1);
            l[i].setOpacity(1);
        }
    }

    public void hideLabel() {
```

```
            bg.setOpacity(0);
            for (int i = 0; i < 7; i++) {
                r[i].setOpacity(0);
                l[i].setOpacity(0);
            }
            lt.setText("");
    //        l1.setText("");
    //        l2.setText("");
        }
    }
```

20. dosemapping > UIPrimary.java

```
    /*
     * To change this license header, choose License Headers in Project Properties.
     * To change this template file, choose Tools | Templates
     * and open the template in the editor.
     */
    package dosemapping;

    import com.lynden.gmapsfx.GoogleMapView;
    import com.lynden.gmapsfx.javascript.object.GoogleMap;
    import java.util.Locale;
    import javafx.geometry.Orientation;
    import javafx.geometry.Pos;
    import javafx.scene.control.Button;
    import javafx.scene.control.CheckBox;
    import javafx.scene.control.Label;
    import javafx.scene.control.ListView;
    import javafx.scene.control.Menu;
    import javafx.scene.control.MenuBar;
    import javafx.scene.control.MenuItem;
    import javafx.scene.control.Separator;
    import javafx.scene.control.SplitPane;
    import javafx.scene.control.SplitPane.Divider;
    import javafx.scene.control.Tab;
    import javafx.scene.control.TabPane;
    import javafx.scene.control.TableColumn;
    import javafx.scene.control.TableView;
    import javafx.scene.control.TextArea;
    import javafx.scene.control.ToolBar;
    import javafx.scene.layout.BorderPane;
    import javafx.scene.layout.StackPane;

    /**
     *
     * @author Administrator
     */
    public class UIPrimary {

        protected GoogleMapView mapComponent = new GoogleMapView(Locale.getDefault().getLanguage(), null);;
```

```
    protected GoogleMap map;

    protected UIMapLabel mapLabel = new UIMapLabel();

    protected Label lbLatitude = new Label();
    protected Label lbLongitude = new Label();

    protected TableView<DataSet> tvDataSet = new TableView<DataSet>();
    protected TableColumn<DataSet,String> tcDataSetId = new TableColumn<DataSet,String>("Id");
    protected TableColumn<DataSet,String> tcDataSetName = new TableColumn<DataSet,String>("Name");
    protected TableColumn<DataSet,String> tcDataSetDescription = new
TableColumn<DataSet,String>("Description");
    protected TableColumn<DataSet,String> tcDataSetDate = new TableColumn<DataSet,String>("Date");

    protected TableView<Detector> tvDetector = new TableView<Detector>();
    protected TableColumn<Detector,String> tcDetectorId = new TableColumn<Detector,String>("Id");
    protected TableColumn<Detector,String> tcDetectorName = new TableColumn<Detector,String>("Name");
    protected TableColumn<Detector,String> tcDetectorDescription = new
TableColumn<Detector,String>("Description");

    protected TableView<Gps> tvGps = new TableView<Gps>();
    protected TableColumn<Gps,String> tcGpsId = new TableColumn<Gps,String>("Id");
    protected TableColumn<Gps,String> tcGpsName = new TableColumn<Gps,String>("Name");
    protected TableColumn<Gps,String> tcGpsDescription = new TableColumn<Gps,String>("Description");

    protected BorderPane bp = new BorderPane();
    protected ToolBar tbt = new ToolBar();
    protected MenuBar mbt = new MenuBar();
    protected ToolBar tbb = new ToolBar();
    protected SplitPane sph = new SplitPane();
    protected SplitPane spv = new SplitPane();

    protected Menu mnDetector = new Menu("Detector");
    protected Menu mnGps = new Menu("GPS");
    protected Menu mnCamera = new Menu("Camera");
    protected Menu mnData = new Menu("Data");
    protected Menu mnMap = new Menu("Map");

    protected MenuItem miAddDetector = new MenuItem("Add Detector");
    protected MenuItem miDeleteDetector = new MenuItem("Delete Detector");

    protected MenuItem miAddGps = new MenuItem("Add GPS");
    protected MenuItem miDeleteGps = new MenuItem("Delete GPS");

    protected MenuItem miImportData = new MenuItem("Import Data");
    protected MenuItem miDeleteData = new MenuItem("Delete Data");
    protected MenuItem miAcquireData = new MenuItem("Acquire Data");

    protected MenuItem miDrawLines = new MenuItem("Draw Lines");
    protected MenuItem miDrawMarkers = new MenuItem("Draw Markers");
    protected MenuItem miDrawCircles = new MenuItem("Draw Circles");
```

```
    protected MenuItem miDrawPolygons = new MenuItem("Draw Polygons");
    protected MenuItem miDrawContour =  new MenuItem("Draw Contour");
    protected MenuItem miClearMap = new MenuItem("Clear Map");

//    protected Button btAddDetector = new Button("Add Detector");
//    protected Button btDeleteDetector = new Button("Delete Detector");
//    protected Button btAddGps = new Button("Add GPS");
//    protected Button btDeleteGps = new Button("Delete GPS");
//    protected Button btImportDataSet = new Button("Import Data");
//    protected Button btDeleteDataSet = new Button("Delete Data");
//    protected Button btAcquireRealTimeData = new Button("Acquire Data");
//    protected Button btDrawLines = new Button("Draw Lines");
//    protected Button btDrawMarkers = new Button("Draw Markers");
//    protected Button btDrawCircles = new Button("Draw Circles");
//    protected Button btDrawPolygons = new Button("Draw Polygons");
//    protected Button btClearMap = new Button("Clear Map");

    protected TabPane tpTable = new TabPane();
    protected Tab tbDataSet = new Tab("Data Sets");
    protected Tab tbDetector = new Tab("Detectors");
    protected Tab tbGps = new Tab("GPSs");
    protected Tab tbCamera = new Tab("Cameras");

    protected TabPane tpDescription = new TabPane();
    protected Tab tbDescription = new Tab("Selected Data Set");

    protected TabPane tpList = new TabPane();
    protected Tab tbList = new Tab("Active Map Items");

    protected TextArea taDescription = new TextArea();

    protected ListView<CheckBox> checkList = new ListView<CheckBox>();

    protected StackPane mapElem = new StackPane();

    public UIPrimary() {

    // Set UI properties
        tbDataSet.closableProperty().setValue(false);
        tbDetector.closableProperty().setValue(false);
        tbGps.closableProperty().setValue(false);
        tbCamera.closableProperty().setValue(false);
        tbDescription.closableProperty().setValue(false);
        tbList.closableProperty().setValue(false);

        taDescription.autosize();
        taDescription.setEditable(false);

        sph.setOrientation(Orientation.HORIZONTAL);
        sph.setDividerPositions(0.2);
```

```
        spv.setMinWidth(300);
        spv.setOrientation(Orientation.VERTICAL);
        spv.setDividerPositions(0.3, 0.6);

    // UI construction
        tbDataSet.setContent(tvDataSet);
        tbDetector.setContent(tvDetector);
        tbGps.setContent(tvGps);
        tpTable.getTabs().addAll(tbDataSet, tbDetector, tbGps, tbCamera);

        tbDescription.setContent(taDescription);
        tpDescription.getTabs().add(tbDescription);

        tbList.setContent(checkList);
        tpList.getTabs().add(tbList);

        mapElem.getChildren().addAll(mapComponent, mapLabel);
        mapElem.setAlignment(mapLabel, Pos.TOP_LEFT);
        mapLabel.setTranslateX(10);
        mapLabel.setTranslateY(100);

        sph.getItems().addAll(spv, mapElem);
        spv.getItems().addAll(tpTable, tpDescription, tpList);

        mnDetector.getItems().addAll(miAddDetector, miDeleteDetector);
        mnGps.getItems().addAll(miAddGps, miDeleteGps);
        mnData.getItems().addAll(miImportData, miDeleteData, miAcquireData);
        mnMap.getItems().addAll(miDrawLines, miDrawMarkers, miDrawCircles, miDrawPolygons, miDrawContour,
miClearMap);

        mbt.getMenus().addAll(mnDetector, mnGps, mnCamera, mnData, mnMap);

//        tbt.getItems().addAll(btAddGps, btDeleteGps, new Separator(), btAddDetector, btDeleteDetector, new
Separator(), btImportDataSet, btDeleteDataSet, btAcquireRealTimeData, new Separator(), btDrawLines,
btDrawMarkers, btDrawCircles, btDrawPolygons, btClearMap);
        tbb.getItems().addAll(lbLatitude, lbLongitude);

        bp.setTop(mbt);
//        bp.setTop(tbt);
        bp.setCenter(sph);
        bp.setBottom(tbb);
    }
}
```

21. dosemapping > WorkingSpace.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
```

```
package dosemapping;

import java.io.File;
import java.util.ArrayList;
import java.util.List;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.scene.control.CheckBox;
import map.Circles;
import map.Contour;
import map.Markers;
import map.Polygons;
import map.Polylines;

/**
 *
 * @author Administrator
 */
public class WorkingSpace {
    protected List<Polylines> mapPolylinesGroup = new ArrayList<>();
    protected List<Markers> mapMarkersGroup = new ArrayList<>();
    protected List<Circles> mapCirclesGroup = new ArrayList<>();
    protected List<Polygons> mapPolygonsGroup = new ArrayList<>();
    protected List<Contour> mapContourGroup = new ArrayList<>();

    protected File currentFile = null;

    protected ObservableList<Detector> detectorList = FXCollections.observableArrayList();
    protected ObservableList<Gps> gpsList = FXCollections.observableArrayList();
    protected ObservableList<DataSet> dataSetList = FXCollections.observableArrayList();
    protected ObservableList<CheckBox> activeMapItemsList = FXCollections.observableArrayList();
}
```

22. map > Circles.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package map;

import com.lynden.gmapsfx.shapes.Circle;
import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author ppengvan
 */
public class Circles {
```

```
    private List<Circle> circles = new ArrayList<>();
    private double minDose;
    private double maxDose;
    private double minLat;
    private double maxLat;
    private double minLon;
    private double maxLon;

    public void setCircle(Circle circle) {
        circles.add(circle);
    }
    public void setMinDose(double val) {
        minDose = val;
    }
    public void setMaxDose(double val) {
        maxDose = val;
    }
    public void setMinLat(double val) {
        minLat = val;
    }
    public void setMaxLat(double val) {
        maxLat = val;
    }
    public void setMinLon(double val) {
        minLon = val;
    }
    public void setMaxLon(double val) {
        maxLon = val;
    }
    public List<Circle> getCircles() {
        return circles;
    }
    public double getMaxDose() {
        return maxDose;
    }
    public double getMinDose() {
        return minDose;
    }
    public double getMinLat() {
        return minLat;
    }
    public double getMaxLat() {
        return maxLat;
    }
    public double getMinLon() {
        return minLon;
    }
    public double getMaxLon() {
        return maxLon;
    }
}
```

23. map > Contour.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package map;

import com.lynden.gmapsfx.shapes.Polygon;
import com.lynden.gmapsfx.shapes.Polyline;
import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author ppengvan
 */
public class Contour {
    private List<Polyline> polylines = new ArrayList<>();
    private List<Polygon> polygons = new ArrayList<>();
    private double minDose;
    private double maxDose;
    private double minLat;
    private double maxLat;
    private double minLon;
    private double maxLon;

    public void setContour(Polyline polyline) {
        polylines.add(polyline);
    }
    public void setContour(Polygon polygon) {
        polygons.add(polygon);
    }
    public void setMinDose(double val) {
        minDose = val;
    }
    public void setMaxDose(double val) {
        maxDose = val;
    }
    public void setMinLat(double val) {
        minLat = val;
    }
    public void setMaxLat(double val) {
        maxLat = val;
    }
    public void setMinLon(double val) {
        minLon = val;
    }
    public void setMaxLon(double val) {
        maxLon = val;
```

```
    }
    public List<Polyline> getContourPolylines() {
        return polylines;
    }
    public List<Polygon> getContourPolygons() {
        return polygons;
    }
    public double getMaxDose() {
        return maxDose;
    }
    public double getMinDose() {
        return minDose;
    }
    public double getMinLat() {
        return minLat;
    }
    public double getMaxLat() {
        return maxLat;
    }
    public double getMinLon() {
        return minLon;
    }
    public double getMaxLon() {
        return maxLon;
    }
}
```

24. map > Markers.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package map;

import com.lynden.gmapsfx.javascript.object.Marker;
import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author ppengvan
 */
public class Markers {
    private List<Marker> markers = new ArrayList<>();
    private double minDose;
    private double maxDose;
    private double minLat;
    private double maxLat;
    private double minLon;
```

```
    private double maxLon;

    public void setMarker(Marker marker) {
        markers.add(marker);
    }
    public void setMinDose(double val) {
        minDose = val;
    }
    public void setMaxDose(double val) {
        maxDose = val;
    }
    public void setMinLat(double val) {
        minLat = val;
    }
    public void setMaxLat(double val) {
        maxLat = val;
    }
    public void setMinLon(double val) {
        minLon = val;
    }
    public void setMaxLon(double val) {
        maxLon = val;
    }
    public List<Marker> getMarkers() {
        return markers;
    }
    public double getMaxDose() {
        return maxDose;
    }
    public double getMinDose() {
        return minDose;
    }
    public double getMinLat() {
        return minLat;
    }
    public double getMaxLat() {
        return maxLat;
    }
    public double getMinLon() {
        return minLon;
    }
    public double getMaxLon() {
        return maxLon;
    }
}
```

25. map > Polygons.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
```

```
 * and open the template in the editor.
 */
package map;

import com.lynden.gmapsfx.shapes.Polygon;
import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author ppengvan
 */
public class Polygons {
    private List<Polygon> polygons = new ArrayList<>();
    private double minDose;
    private double maxDose;
    private double minLat;
    private double maxLat;
    private double minLon;
    private double maxLon;

    public void setPolygon(Polygon polygon) {
        polygons.add(polygon);
    }
    public void setMinDose(double val) {
        minDose = val;
    }
    public void setMaxDose(double val) {
        maxDose = val;
    }
    public void setMinLat(double val) {
        minLat = val;
    }
    public void setMaxLat(double val) {
        maxLat = val;
    }
    public void setMinLon(double val) {
        minLon = val;
    }
    public void setMaxLon(double val) {
        maxLon = val;
    }
    public List<Polygon> getPolygons() {
        return polygons;
    }
    public double getMaxDose() {
        return maxDose;
    }
    public double getMinDose() {
        return minDose;
    }
```

```
    public double getMinLat() {
        return minLat;
    }
    public double getMaxLat() {
        return maxLat;
    }
    public double getMinLon() {
        return minLon;
    }
    public double getMaxLon() {
        return maxLon;
    }
}
```

26. map > Polylines.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package map;

import com.lynden.gmapsfx.shapes.Polyline;
import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author ppengvan
 */
public class Polylines {
    private List<Polyline> polylines = new ArrayList<>();
    private double minDose;
    private double maxDose;
    private double minLat;
    private double maxLat;
    private double minLon;
    private double maxLon;

    public void setPolyline(Polyline polyline) {
        polylines.add(polyline);
    }
    public void setMinDose(double val) {
        minDose = val;
    }
    public void setMaxDose(double val) {
        maxDose = val;
    }
    public void setMinLat(double val) {
        minLat = val;
```

```
    }
    public void setMaxLat(double val) {
        maxLat = val;
    }
    public void setMinLon(double val) {
        minLon = val;
    }
    public void setMaxLon(double val) {
        maxLon = val;
    }
    public List<Polyline> getPolylines() {
        return polylines;
    }
    public double getMaxDose() {
        return maxDose;
    }
    public double getMinDose() {
        return minDose;
    }
    public double getMinLat() {
        return minLat;
    }
    public double getMaxLat() {
        return maxLat;
    }
    public double getMinLon() {
        return minLon;
    }
    public double getMaxLon() {
        return maxLon;
    }
}
```

ประวัตินักวิจัยและคณะ

ประวัติผู้วิจัย

ชื่อ-นามสกุล	(ภาษาไทย)	นายพงษ์แพทย์ เพ่งวาณิชย์
	(ภาษาอังกฤษ)	Mr.Phongphaeth Pengvanich
เพศ ชาย	วันเดือนปีเกิด	30/01/2524
เลขที่บัตรประชาชน		3619900088459
ตำแหน่งปัจจุบัน		อาจารย์
สถานที่ติดต่อ (ที่ทำงาน)		ภาควิชาวิศวกรรมนิวเคลียร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ปทุมวัน กรุงเทพ 10330
โทรศัพท์		02-2186770
โทรสาร		02-2186780
E-mail		phongphaeth.p@chula.ac.th
ที่อยู่ (ที่บ้าน)		92/41 ถ. สุขาภิบาล 1 ต. บางศรีเมือง อ. เมือง จ. นนทบุรี 11000
โทรศัพท์/โทรสาร		080-558-6931

ประวัติการศึกษา (ปริญญาตรี – เอก ; สาขา และสถาบัน)

ปริญญาเอก	Doctor of Philosophy (Nuclear Engineering) University of Michigan, Ann Arbor, Michigan, USA
ปริญญาโท	Master of Science in Engineering (Nuclear Engineering) University of Michigan, Ann Arbor, Michigan, USA
ปริญญาตรี	Bachelor of Science in Engineering (Nuclear Engineering) University of Michigan, Ann Arbor, Michigan, USA

ผลงานวิจัย

ก. ผลงานวิจัยที่ตีพิมพ์ในวารสารระดับชาติและนานาชาติ

1. P. Pengvanich, Y. Y. Lau, E. Cruz, R. M. Gilgenbach, B. Hoff, and J. W. Luginsland, “Analysis of peer-to-peer locking of magnetrons,” Physics of Plasmas, 15, 103104, October 2008.P.
2. Pengvanich, Y. Y. Lau, J. W. Luginsland, R. M. Gilgenbach, E. Cruz, and E. Schamiloglu, “Effects of frequency chirp on magnetron injection locking,” Physics of Plasmas, 15, 073110, July 2008.
3. P. Pengvanich, D. Chernin, Y. Y. Lau, J. W. Luginsland, and R. M. Gilgenbach, “Effect of random circuit fabrication errors on small signal gain and phase in traveling wave tubes,” IEEE Transactions on Electron Devices, 55, 3, March 2008.

4. P. Pengvanich, V. B. Neculaes, Y. Y. Lau, R. M. Gilgenbach, M. C. Jones, W. M. White, and R. D. Kowalczyk, “Modeling and experimental studies of magnetron injection locking,” Journal of Applied Physics, 98, 114903, December 2005.
5. V. B. Neculaes, **P. Pengvanich**, Y. Hidaka, Y. Y. Lau, R. M. Gilgenbach, W. M. White, M. C. Jones, H. L. Bosman, and J. W. Luginsland, “Rapid Kinematic Bunching and Parametric Instability in a Crossed-Field Gap With a Periodic Magnetic Field,” IEEE Transactions on Plasma Science, 33, 2, April 2005.
6. E. J. Cruz, B. W. Hoff, **P. Pengvanich**, Y. Y. Lau, R. M. Gilgenbach, and J. W. Luginsland, "Experiments on Peer-to-Peer Locking of Magnetrons,” Applied Physics Letters, 95, 191503, 2009.
7. N. M. Jordan, Y. Y Lau, D. M. French, R. M. Gilgenbach, and **P. Pengvanich**, “Electric field and electron orbits near a triple point,” Journal of Applied Physics, 102, 033301, 2007.
8. V.B Neculaes, M.C. Jones, R.M. Gilgenbach, Y.Y Lau, J.W. Luginsland, W. White, N.M. Jordan, **P. Pengvanich**, Y. Hidaka and H. Bosman, "Magnetic Priming Effects on Noise, Startup and Mode Competition in Magnetrons", IEEE Transactions on Plasma Science, Special Issue of Invited Papers from ICOPS 2004, 2005.
9. V. B. Neculaes, M. C. Jones, R. M. Gilgenbach, Y. Y. Lau, J. W. Luginsland, B. W. Hoff, W. M. White, N. M. Jordan, **P. Pengvanich**, Y. Hidaka, H. L. Bosman**, “Magnetic perturbation effects on noise and startup in DC-operating oven magnetrons,”** IEEE Transactions on Electron Devices, Special Issue of IVEC papers, **52, 5, May 2005.**
10. M. C. Jones, V. B. Neculaes, W. M. White, Y. Y. Lau, R. M. Gilgenbach, J. W. Luginsland, **P. Pengvanich**, N. M. Jordan, Y. Hidaka, H. L. Bosman, “**Simulations of magnetic priming in a relativistic magnetron,”** IEEE Transactions on Electron Devices, Special Issue of IVEC papers, **52, 5, May 2005.**

ข. ผลงานวิจัยที่สามารถนำไปใช้ประโยชน์ได้

-

ค. ผลงานอื่นๆ เช่น ตำรา บทความ สิทธิบัตร ฯลฯ

Conference Records :

1. C. Thanasupsombat, **P. Pengvanich**, S. Aootaphao, and S. S. Thongvigitmanee, “Monte Carlo method for characterization of x-ray scattering in a cone beam computed tomography,” 10th International Conference on Nuclear Analytical Methods in the Life Sciences, Bangkok, Thailand January 2012.

2. S. Boriboon, **P. Pengvanich**, and S. Laoharojanapan, "Instrumental neutron activation analysis of aromatic rice based on the K-0 method," 10th International Conference on Nuclear Analytical Methods in the Life Sciences, Bangkok, Thailand, January 2012.
3. L. Mitrayon, **P. Pengvanich**, and S. Punnachaiya, "Development of low-temperature atmospheric dielectric barrier discharge plasma source," 12th Thailand Nuclear Science and Technology Conference, 2011.
4. **P. Pengvanich**, Y. Y. Lau, R. M. Gilgenbach, D. Chernin, and J. W. Luginsland, "Effects of random circuit fabrication errors on small signal gain and phase in traveling wave tubes," 35th IEEE International Conference on Plasma Science, June 2008.
5. **P. Pengvanich**, Y. Y. Lau, R. M. Gilgenbach, E. J. Cruz, J. W. Luginsland, E. Schamiloglu, "Magnetron Phase Locking: Effects of Frequency Chirp and Locking of Multiple Magnetrons," 35th IEEE International Conference on Plasma Science, June 2008.
6. E. Cruz, **P. Pengvanich**, Y. Y. Lau, R. Gilgenbach, J. Luginsland, E. Schamiloglu, "Recent advance in magnetron phase locking: effects of frequency chirps and locking of multiple magnetrons, "American Physical Society, 49th Annual Meeting of the Division of Plasma Physics, November 2007.
7. **P. Pengvanich**, D. P. Chernin, Y. Y. Lau, J. W. Luginsland, and R. M. Gilgenbach, "Effect of random circuit fabrication errors on small signal gain and phase in helix traveling wave tubes," American Physical Society, 49th Annual Meeting of the Division of Plasma Physics, November 2007.
8. **P. Pengvanich**, Y. Y. Lau, D. Chernin, J. W. Luginsland, and R. M. Gilgenbach, "Effects of circuit manufacturing errors on small signal gain and phase in a traveling wave tube," 34th IEEE International Conference on Plasma Science, June 2007.
9. **P. Pengvanich**, Y. Y. Lau, R. M. Gilgenbach, E. J. Cruz, J. W. Luginsland, and E. Schamiloglu, "Recent advances in magnetron phase locking: effects of frequency chirps and locking of multiple magnetrons," 34th IEEE International Conference on Plasma Science, June 2007.
10. **P. Pengvanich**, Y.Y. Lau, R.M. Gilgenbach, and J.W. Luginsland, "Modeling and magnetron injection locking characteristics," American Physical Society, 48th Annual Meeting of the Division of Plasma Physics, October 2006.
11. **P. Pengvanich**, Y. Y. Lau, D. Chernin, J. W. Luginsland, and R. M. Gilgenbach, "Effect of random geometric perturbations on slow wave devices," 33th IEEE International Conference on Plasma Science, June 2006.

12. **P. Pengvanich**, Y.Y. Lau, R.M. Gilgenbach, V.B. Neculaes, M.C. Jones, W.M. White, R.D. Kowalczyk, "Modeling and magnetron injection locking characteristics," 33th IEEE International Conference on Plasma Science, June 2006.
13. **P. Pengvanich**, D. Chernin, Y. Y. Lau, R. M. Gilgenbach, and D. Dialetis, "Effects of Random Geometrical Perturbations in Slow Wave Devices," American Physical Society, 47th Annual Meeting of the Division of Plasma Physics, October 2005.
14. **P. Pengvanich**, V. B. Neculaes, Y. Y. Lau, R. M. Gilgenbach, M. C. Jones, W. M. White, and R. D. Kowalczyk, "Modeling and Experimental Studies of Magnetron Injection Locking," 32nd IEEE International Conference on Plasma Science, June 2005.
15. **P. Pengvanich**, V. B. Neculaes, Y. Hidaka, Y. Y. Lau, R. M. Gilgenbach, W. White, M. C. Jones, H. Bosman, J. W. Luginsland, "Rapid Kinematic Bunching and Parametric Instability in a Crossed-Field Gap with a Periodic Magnetic Field," American Physical Society, 46th Annual Meeting of the Division of Plasma Physics, November 2004.
16. N. M. Jordan, R. M. Gilgenbach, Y. Y. Lau, B. W. Hoff, D. M. French, and **P. Pengvanich**, "Metal-oxide-junction, triple-point cathodes for high current vacuum electron devices," American Physical Society, 49th Annual Meeting of the Division of Plasma Physics, November 2007.
17. D. M. French, N. M. Jordan, Y. Y. Lau, R. M. Gilgenbach, and **P. Pengvanich**, "Electric field and electron orbits near a triple point," American Physical Society, 49th Annual Meeting of the Division of Plasma Physics, November 2007.
18. N. M. Jordan, R. M. Gilgenbach, Y. Y. Lau, B. W. Hoff, E. J. Cruz, D. M. French, M. R. Gomez, **P. Pengvanich**, J. Zier, and M. C. Jones, "Metal-oxide-junction, triple-point cathodes for high current vacuum electron devices," 34th IEEE International Conference on Plasma Science, June 2007.
19. R.M. Gilgenbach, B.W. Hoff, Y.Y. Lau, N.M. Jordan, E. Cruz, **P. Pengvanich**, W. White, T.A. Spencer, D. Price, "Enhanced performance of a relativistic magnetron by magnetic priming," American Physical Society, 48th Annual Meeting of the Division of Plasma Physics, October 2006.
20. W. M. White, R. M. Gilgenbach, M. C. Jones, V. B. Neculaes, Y. Y. Lau, N. Jordan, **P. Pengvanich**, R. Edgar, B. Hoff, T. A. Spencer, D. Price, "RF Priming Experiments and Simulations of Magnetic Priming in Relativistic Magnetrons," American Physical Society, 46th Annual Meeting of the Division of Plasma Physics, November 2004.
21. V. Neculaes, R. M. Gilgenbach, Y. Y. Lau, M. C. Jones, J. Luginsland, W. White, **P. Pengvanich**, N. M. Jordan, Y. Hidaka, H. Bosman, "Magnetron microwave noise

reduction and magnetic priming by azimuthally varying axial magnetic fields," **5th IEEE International Vacuum Electronics Conference, April 2004**.

22. M. C. Jones, V. B. Neculaes, W. White, Y.Y. Lau, R. M. Gilgenbach, J. Luginsland, **P. Pengvanich**, N.M. Jordan, Y. Hidaka, H. Bosman, "Simulation of rapid startup in microwave magnetrons with azimuthally-varying axial magnetic fields," **5th IEEE International Vacuum Electronics Conference, April 2004**.
23. V. B. Neculaes, R. M. Gilgenbach, M. Lopez, Y. Y. Lau, M. Jones, W. White, **P. Pengvanich**, M. D. Johnson, T. Strickler, T. A. Spencer, J. Luginsland, M. Haworth, K. Cartwright, P. Mardahl, T. Murphy, and D. Price, "Mode Competition in Relativistic Magnetron and Injection Locking in KW Magnetrons," American Institute of Physics Conference Proceeding, Vol. 691, Issue 1, p301, December 2003.
24. V. B. Neculaes, R.M. Gilgenbach, Y.Y. Lau, M.C. Jones, W.M. White and **P. Pengvanich**, "Microwave Noise Reduction Experiments in kW Magnetrons," American Physical Society, 45th Annual Meeting of the Division of Plasma Physics, 2003.
25. R. M. Gilgenbach, V. B. Neculaes, M. Lopez, M. Jones, W. White, Y. Y. Lau, **P. Pengvanich**, M. D. Johnston, T. Strickler, T. A. Spencer, J. Luginsland, M. Haworth, K. Cartwright, P. Mardahl, T. Murphy, D. Price, **4th IEEE International Vacuum Electronics Conference, May 2003.**

ง. รางวัลผลงานวิจัยที่เคยได้รับ

2007:	IEEE Nuclear and Plasma Sciences Society Graduate Scholarship Award
2006 – 2007:	Rackham Pre-doctoral Fellowship
2002:	Rackham Graduate School Recruitment Fellowship
2002:	James B. Angell Scholar, University of Michigan
2000 – 2001:	University Honors
2000:	Class Honors
1999:	Tau Beta Pi
1998 – 2002:	Dean's List
1997 – 2008:	Royal Thai Scholarship (BSE through Ph.D.), Thai Government

จ. สาขาวิชาที่เชี่ยวชาญ (สามารถตอบได้มากกว่า 1 สาขา)

- Theoretical modeling and simulation for high power microwave and millimeter wave sources, specifically magnetrons, klystrons, and traveling wave amplifiers
- General theory of oscillators
- Generation of terahertz radiation using vacuum electronics.
- Processing of plasmas for industrial and medical applications, specifically low temperature atmospheric plasmas

- Nuclear nonproliferation
- Nuclear security

ฉ. ภาระงานในปัจจุบัน

1. งานประจำ

อาจารย์ประจำภาควิชา วิศวกรรมนิวเคลียร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

2. งานวิจัยที่รับผิดชอบในปัจจุบัน

- การพัฒนาซอฟแวร์สำหรับจำลองพลาสมาในบรรยากาศ
- การผลิตก๊าซไฮโดรเจนและมีเทนจากเอทานอลโดยใช้ไดอิเล็กทริกแบริเออร์พลาสมา
- การผลิตไดอิเล็กทริกแบริเออร์พลาสมาความหนาแน่นสูงสำหรับฆ่าเชื้อ
- การผลิตเครื่องอิเล็กตรอนไซโคลตรอนเรโซแนนซ์พลาสมาสำหรับเคลือบฟิล์มบนพื้นผิว
- การจำลองการกระจายของรังสีเอกซ์ในเนื้อเยื่อจากเครื่องเอกซ์เรย์คอมพิวเตอร์สำหรับทันตกรรม
- โครงการ : การศึกษาสถานภาพเทคโนโลยีเครื่องปฏิกรณ์นิวเคลียร์ขนาดเล็กแบบโมดูลาร์” โครงการร่วมสนับสนุนทุนวิจัยและพัฒนา กฟผ. – สกว. มกราคม 2556 - เมษายน 2557

ผู้ร่วมวิจัย คนที่ 1

1. ข้อมูลส่วนตัว ชื่อ-นามสกุล (ภาษาไทย) นางสาว ชุติมา กรานรอด

(ภาษาอังกฤษ) Miss Chutima Kranrod

2. เลขบัตรประจำตัวประชาชน 3 1101 00708 76 9

3. ตำแหน่งปัจจุบัน นักวิจัย

4. สถานที่ทำงานปัจจุบัน

ภาควิชา วิศวกรรมนิวเคลียร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ถนนพญาไท เขตปทุมวัน กรุงเทพมหานคร
รหัสไปรษณีย์ 10330โทรศัพท์ 02-218-6784 โทรศัพท์มือถือ 081-692-3466
โทรสาร 02-218-6780
E-mail: ckranrod@gmail.com

5. ประวัติการศึกษา

การศึกษาระดับปริญญาตรี	คุณวุฒิ วท.บ. (ชีววิทยา) ปีที่จบ พ.ศ. 2540 ภาควิชา รังสีประยุกต์และไอโซโทป มหาวิทยาลัยเกษตรศาสตร์
การศึกษาระดับปริญญาโท	คุณวุฒิ วท.ม (นิวเคลียร์เทคโนโลยี) ปีที่จบ พ.ศ. 2545 ภาควิชา นิวเคลียร์เทคโนโลยี จุฬาลงกรณ์มหาวิทยาลัย
การศึกษาระดับปริญญาเอก	คุณวุฒิ วศ.ด. (วิศวกรรมนิวเคลียร์) ปีที่จบ พ.ศ. 2554 ภาควิชานิวเคลียร์เทคโนโลยี จุฬาลงกรณ์มหาวิทยาลัย

6. สาขาวิชาการที่มีความชำนาญพิเศษ

การตรวจวัดรังสีในสิ่งแวดล้อม เน้นการตรวจวัดก๊าซเรดอน โทรอน และธาตุลูกหลานเรดอนและโทรอน

7. ประสบการณ์การทำงานวิจัย

<u>ต่างประเทศ</u>

ปี พ.ศ. 2551 ถึง 2553 ตำแหน่ง นักวัยจัยฝ่ายเทคนิค ชื่อหน่วยงาน Research Center for Radiation Protection, National Institute of Radiological Sciences (NIRS) ที่ตั้ง เมืองชิบะ ประเทศญี่ปุ่น

ปี พ.ศ 2547 (4เดือน) และ 2550 (11เดือน) ตำแหน่ง นักวิจัยแลกเปลี่ยนโครงการ Nuclear Researchers Exchange Program (MEXT/NSRA) ชื่อหน่วยงาน National Institute of Radiological Science (NIRS) ที่ตั้ง เมืองชิบะ ประเทศญี่ปุ่น

ภายในประเทศ

ปี พ.ศ.2548 ถึง ปัจจุบัน ตำแหน่ง นักวิจัยโครงการวิจัยร่วมภาครัฐกับเอกชน ชื่อหน่วยงาน ภาควิชาวิศวกรรมนิวเคลียร์ ที่ตั้ง คณะวิศวกรรมศาสตร์จุฬาลงกรณ์มหาวิทยาลัย กรุงเทพฯ

งานวิจัยที่เกี่ยวข้อง

- งานสื่อสารสร้างความรู้ความเข้าใจโครงการโรงไฟฟ้าใหม่ผ่านเว็บไซต์และ SOCIAL MEDIA, การไฟฟ้าฝ่ายผลิตแห่งประเทศไทยม ม.ค.2555-ม.ค.2558
- การหาปริมาณธาตุกัมมันตรังสีตามธรรมชาติในเครื่องมือและกากจากอุตสาหกรรมปิโตรเลียม บริษัท ปตท.สผ. จากัด (มหาชน) ม.ค.2555-ม.ค.2558
- การจัดทำข้อมูลรังสีพื้นหลังของปริมาณธาตุรังสีตามธรรมชาติใน พืช และผลไม้ท้องถิ่นบางชนิดในพื้นที่ที่มีศักยภาพเป็นที่ตั้งโครงการโรงไฟฟ้าพลังงานนิวเคลียร์ งบประมาณแผ่นดิน 2555-2556
- โครงการ การพัฒนาระบบปรับเทียบก๊าซเรดอนและโทรอน (พ.ศ. 2553 -2554)
- โครงการ การการพัฒนาระบบการวัดการกระจายขนาดเพื่อประเมินอัตราการได้รับรังสีจากละอองฝุ่นขนาดนาโนเมตรของลูกหลานเรดอนและโทรอนในอุตสาหกรรมแร่ (พ.ศ. 2551 -2552)
- โครงการ การศึกษาปริมาณก๊าซเรดอนตามแนวรอยเลื่อนองครักษ์บริเวณศูนย์วิจัยนิวเคลียร์องครักษ์ (พ.ศ.2548-2549)
- โครงการการหาปริมาณธาตุกัมมันตรังสีตามธรรมชาติในผลิตภัณฑ์ ผลิตภัณฑ์พลอยได้และกากที่ได้จากอุตสาหกรรมเคมี (พ.ศ.2546-2548)

งานวิจัยที่กำลังทำ

- งานสื่อสารสร้างความรู้ความเข้าใจโครงการโรงไฟฟ้าใหม่ผ่านเว็บไซต์และ SOCIAL MEDIA, การไฟฟ้าฝ่ายผลิตแห่งประเทศไทยม ม.ค.2555-ม.ค.2558 (แล้วเสร็จ 75%)
- การหาปริมาณธาตุกัมมันตรังสีตามธรรมชาติในเครื่องมือและกากจากอุตสาหกรรมปิโตรเลียม บริษัท ปตท.สผ. จากัด (มหาชน) ม.ค.2555-ม.ค.2558 (แล้วเสร็จ 75%)

8. ผลงานทางวิชาการ

บทความทางวิชาการในวารสารต่างประเทศ

8.2 Chanyotha, S., Kranrod, C., Burnett, W.C.,“Assessing Diffusive Fluxes and Pore Water Radon Activities via a Single Automated Experiment,. Journal of Radioanalytical and Nuclear Chemistry Volume 301, Issue 2, pp 581-588, August 2014.

8.3 Supitcha Chanyotha, C. Kranrod, W. C. Burnett, D. Lane-Smith, J. Simko, “Prospecting for Groundwater Discharge in the Canals of Bangkok via Natural Radon and Thoron”. Journal Of Hydrology., submitted

8.4 S. Chanyotha, C. Kranrod, N. Chankow, R. Kritsananuwat, P. Sriploy and K. Pangza. “Natural Radionuclide Concentrations in Processed Materials from Thai Mineral Industries”, Radiation Protection Dosimetry Vol. 152, No. 1–3, pp. 71–75, (2012)

8.5 C. Kranrod, S. Chanyotha, N.Chankow, T. Ishikawa, S. Tokonami, “Measurement of Radon and Thoron Progeny Size Distributions and Dose Assessments at the Mineral Treatment Industry in Thailand”, Journal of Radioanalytical and Nuclear Chemistry , ISSN 0236-5731, (2012)

8.6 N.M. Hassana, T. Ishikawa, M. Hosoda, K. Iwaoka, A. Sorimachi, S.K. Sahoo, M. Janik, **C.Kranrod**, H. Yonehara, M. Fukushi, S. Tokonami, *The effect of water content on the radon emanation coefficient for some building materials used in Japan.* Radiation Measurements, Volume 46, Issue 2, February 2011, Pages 232-237.

8.7 V.W.Y. Choi, C. K. M. Ng, R. K. K. Lam, M. Janik, A. Sorimachi, **C Kranrod**, D. Nikezic, S. Tokonami, K. N. Yu, *Long-term determination of airborne radon progeny concentrations using LR 115 detectors and the effects of thoron*. Radiat Prot Dosimetry, Volume 141, 2010, pages 404-407.

8.8 K. Sahoo, T. Ishikawa, S. Tokonami, A. Sorimachi, **C. Kranrod**, M. Janik, M. Hosoda, N. M. Hassan, S. Chanyotha, V. K. Parami, H. Yonehara, R. C. Ramola, *A comparative study of thorium activity in NORM and high background radiation area.* Radiat Prot Dosimetry, Volume 141, 2010, pages 416-419.

8.9 **C. Kranrod**, T. Ishikawa, S. Tokonami, A. Sorimachi, S. Chanyotha, N. Chankow, *Comparative Dosimetry of Radon and Thoron.* Radiat Prot Dosimetry, Volume 141, 2010, pages 424-427.

8.10 M. Janik, S. Tokonami, **C. Kranrod**, A. Sorimachi, T. Ishikawa, N. M. Hassan, *International intercomparisons of integrating radon/thoron detectors with the NIRS radon/thoron chambers.* Radiat Prot Dosimetry, Volume 141, 2010, pages 436-439.

8.11 A. Sorimachi, **C. Kranrod**, S. Tokonami, T. Ishikawa, M. Hosoda, M. Janik,, R. Shingaki, M. Furukawa, *Anomalously High Radon Concentrations in a Dwelling in Okinawa, Japan.* Radioisotopes, Volume 58, Issue 12, 2009, Pages 807-813.

8.12 M. Janik, S. Tokonami, T. Kovács, N. Kávási, **C. Kranrod**, A. Sorimachi, H. Takahashi, N. Miyahara, T. Ishikawa, *International intercomparisons of integrating radon detectors in the NIRS radon chamber.* Applied Radiation and Isotopes, Volume 67, Issue 9, September 2009, Pages 1691-1696.

8.13 **C. Kranrod**, S. Tokonami, T. Ishikawa, A. Sorimachi, M. Janik, R. Shingaki, M. Furukawa, S. Chanyotha, N. Chankow, *Mitigation of the effective dose of radon decay products through the use of an air cleaner in a dwelling in Okinawa, Japan.* Applied Radiation and Isotopes, Volume 67, Issue 6, June 2009, Pages 1127-1132.

8.14 **C. Kranrod**, S. Chanyotha, N. Chankow, S. Tokonami, T. Ishikawa, S.K. Sahoo, *Simple Technique for Determining the Equilibrium Equivalent Thoron Concentration Using a*

CR-39 Detector: Application in Mineral Treatment Industry. Radioprotection, Volume 44, Number 5, June 2009, Pages 301-304.

ผู้ร่วมวิจัย (คนที่ 2)

1. ชื่อ นางสุพิชชา จันทรโยธา

Ms. Supitcha Chanyotha

2. เลขหมายบัตรประจำตัวประชาชน 3101701487466

3. ตำแหน่งปัจจุบัน รองศาสตราจารย์ ระดับ 9

4. หน่วยงาน

ภาควิชาวิศวกรรมนิวเคลียร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
โทรศัพท์ 081-484-5164
โทรสาร 02-218-6780
E-mail: Supitcha.C@Chula.ac.th

5. ประวัติการศึกษา

มหาวิทยาลัย	ปริญญา	สาขา	ปีที่ได้รับ
มหาวิทยาลัยขอนแก่น	ตรี	เคมี	2521
จุฬาลงกรณ์มหาวิทยาลัย	โท	นิวเคลียร์เทคโนโลยี	2524
University of Arizona	เอก	Nuclear Engineering	2534

6. สาขาวิชาการที่มีความชำนาญ (แตกต่างจากวุฒิการศึกษา) ระบุสาขาวิชา

- การจัดการกากกัมมันตรังสี
- การสำรวจและตรวจวัดรังสีในสิ่งแวดล้อม
- บรรจุภัณฑ์สำหรับเภสัชรังสี

7. ประสบการณ์ที่เกี่ยวข้องกับการบริหารงานวิจัยทั้งภายในและภายนอก

7.1 ผู้อำนวยการแผนงานวิจัย : ชื่อแผนงานวิจัย

- การวิจัยเชิงนโยบายเพื่อสนับสนุนโครงการโรงไฟฟ้าพลังนิวเคลียร์ของประเทศไทยสำนักงานคณะกรรมการวิจัยแห่งชาติ 2552

7.2 หัวหน้าโครงการวิจัย:

7.2.1 งานสื่อสารสร้างความรู้ความเข้าใจโครงการโรงไฟฟ้าใหม่ผ่านเว็บไซต์และ SOCIAL MEDIA, การไฟฟ้าฝ่ายผลิตแห่งประเทศไทยม ม.ค.2555-ม.ค.2558

7.2.2 การหาปริมาณธาตุกัมมันตรังสีตามธรรมชาติในเครื่องมือและกากจากอุตสาหกรรมปิโตรเลียม บริษัท ปตท.สผ. จากัด (มหาชน) ม.ค.2555-ม.ค.2558

7.2.3 การศึกษาสถานภาพเทคโนโลยีเครื่องปฏิกรณ์นิวเคลียร์ขนาดเล็กแบบโมดูลาร์" โครงการร่วมสนับสนุนทุนวิจัยและพัฒนา กฟผ. – สกว. มกราคม 2556 – กรกฎาคม 2557

7.2.4 การจัดทำข้อมูลรังสีพื้นหลังของปริมาณธาตุรังสีตามธรรมชาติใน พืช และผลไม้ท้องถิ่นบางชนิดในพื้นที่ที่มีศักยภาพเป็นที่ตั้งโครงการโรงไฟฟ้าพลังงานนิวเคลียร์ งบประมาณแผ่นดิน 2555-2556

7.2.5 โครงการศึกษาและจัดทำแผนงานด้านโครงสร้างอุตสาหกรรมและมาตรฐานอุตสาหกรรมและการพาณิชย์" สำนักพัฒนาโครงการโรงไฟฟ้าพลังงานนิวเคลียร์ มิถุนายน 2553

7.2.6 โครงการพัฒนาระบบการวัดการกระจายขนาดเพื่อประเมินอัตราการได้รับรังสีจากละอองฝุ่นขนาดนาโนเมตรของลูกหลานเรดอนและโทรอน โครงการวิจัยร่วมภาครัฐกับเอกชน 2551-2552

7.2.7 โครงการพัฒนาระบบการวัดการกระจายขนาดเพื่อประเมินอัตราการได้รับรังสีจากละอองฝุ่นขนาดนาโนเมตรของลูกหลานเรดอนและโทรอน โครงการวิจัยร่วมภาครัฐกับเอกชน 2551-2552

7.2.8 การวิจัยเชิงนโยบายเพื่อสนับสนุนโครงการโรงไฟฟ้าพลังนิวเคลียร์ของประเทศไทย (ระยะที่ 1) สำนักงานคณะกรรมการวิจัยแห่งชาติ 2551

7.2.9 โครงการ การศึกษาเพิ่มเติมรอยเลื่อนมีพลัง โครงการอ่างเก็บน้ำแม่สะป๊วด (อันเนื่องมาจากพระราชดำริ) จ.ลำพูน โดยใช้เทคนิคการศึกษาปริมาณความเข้มข้นของก๊าซเรดอนตามแนวรอยเลื่อนมีพลังแม่ทา สำนักบริการวิชาการแห่งจุฬาลงกรณ์มหาวิทยาลัย ดำเนินงานศึกษาให้กรมชลประทาน ว่าจ้างโดยบริษัท ครีเอทีฟ เทคโนโลยี เมษายน 2551

7.2.10 การสำรวจและตรวจวัดปริมาณธาตุกัมมันตรังสีในสิ่งแวดล้อมของบริษัท ปตท. สผ. จำกัด (มหาชน) โครงการวิจัยร่วมภาครัฐกับเอกชน 2545-2548

7.2.11 โครงการวิจัยการสำรวจและตรวจวัดปริมาณธาตุกัมมันตรังสีในสิ่งแวดล้อมของบริษัท ไทยแลนด์สเมลติ้งแอนด์รีไฟนิ่ง จำกัด คณะวิศวกรรมศาสตร์ ดำเนินการให้กับบริษัท ไทยแลนด์สเมลติ้งแอนด์รีไฟนิ่ง จำกัด โครงการวิจัยร่วมภาครัฐกับเอกชน 2548-2549.

7.2.12 การสำรวจและตรวจวัดปริมาณธาตุกัมมันตรังสีในสิ่งแวดล้อมของบริษัท ปตท. สผ. สยาม จำกัด โครงการวิจัยร่วมภาครัฐกับเอกชน 2549-2550

7.3 งานวิจัยที่ทำเสร็จแล้ว:

- ทุกโครงการในข้อ 7.1 ยกเว้น ข้อ 7.1.1 (แล้วเสร็จ 75%) และ 7.1.2 (แล้วเสร็จ 75%)
- โครงการวิจัยการใช้ธาตุเรดอน และ เรเดียม เป็นตัวติดตามในการซึมผ่านของน้ำใต้ดินตื้นเข้าสู่น้ำผิวดิน (Utility of Radon and Radium Isotopes to Trace Shallow Groundwater Seepage in to Surface water) ร่วมวิจัยกับ Florida State University 2006 – 2009
- W.C Burnett, Wattayakorn, G and **S. Chanyotha** , "Assessing the Groundwater Nutrient Pump of Tonle Sap Lake, Cambodia." , ทุนจาก National Geographic Society, 2009-2010.

7.3.1 ผลงานตีพิมพ์และประชุมวิชาการ

1. **Supitcha Chanyotha**, C. Kranrod, W. C. Burnett, D. Lane-Smith, J. Simko, "*Prospecting for Groundwater Discharge in the Canals of Bangkok via Natural Radon and Thoron*". Journal of Hydrology, submitted.
2. R. Kritsananuwat, S. K. Sahoo, M. Fukushi, K. Pangza, **S. Chanyotha** "Radiological risk assessment of ^{238}U, ^{232}Th and ^{40}K in Thailand coastal sediments at selected

areas proposed for nuclear power plant sites". J Radioanal Nucl Chem, DOI 10.1007/s10967-014-3376-7

3. **Chanyotha, S.**, Kranrod, C., Burnett, W.C.,*"Assessing Diffusive Fluxes and Pore Water Radon Activities via a Single Automated Experiment,.* **Journal of Radioanalytical and Nuclear Chemistry** Volume 301, Issue 2, pp 581-588, August 2014.
4. C. Kranrod, **S. Chanyotha,** N.Chankow, T. Ishikawa, S. Tokonami, **"***Measurement of Radon and Thoron Progeny Size Distributions and Dose Assessments at the Mineral Treatment Industry in Thailand"*, **Journal of Radioanalytical and Nuclear Chemistry** , doi 10.1007/s10967-012-2151-x, ISSN 0236-5731**,** 2012
5. W.C.Burnett, R.N.Peterson, S.Chanyotha, G. Wattayakorn, B.Ryan, *"Using High-Resolution In-Situ Radon Measurements to Determine Groundwater Discharge At A Remote Location: Tonle Sap Lake, Cambodia",* **Journal of Radioanalytical and Nuclear Chemistry** , doi 10.1007/s10967-012-1914-8, 2012
6. S. Chanyotha; C. Kranrod; N. Chankow; R. Kritsananuwat; P. Sriploy; K. Pangza, Natural Radionuclide Concentrations In Processed Materials From Thai Mineral Industries, **Radiation Protection Dosimetry,** doi: 10.1093/rpd/ncs185, 2012
7. **S. Chanyotha**, C. Kranrod, S. Tokonami, N. Suwankot, T. Pangza**,** C. Pornnumpa *"Terrestrial Gamma Radiation in Phuket Island*, Thailand"**, Engineering Journal** (ISSN 8281-0125, Vol.15 Issue 4p 76-65, 2011
8. C. Kranrod, T. Ishikawa, S. Tokonami, A. Sorimachi, **S. Chanyotha** and N. Chankow , *"Comparative dosimetry of radon and thoron",* **Radiation Protection Dosimetry**, September 9, 2010
9. **S. Chanyotha**, W. C. Burnett, M. Taniguchi, R. Kritsananuwat and P. Sriploy, "*Experience in using radon and thoron data to solve Environmental and water problems"*, **Radiation Protection Dosimetry**, September 23, 2010
10. S. K. Sahoo, T. Ishikawa1, S. Tokonami1, A. Sorimachi, C. Kranrod, M. Janik, M. Hosoda1, N. Hassan, **S. Chanyotha**, V. K. Parami, H. Yonehara1 and R. C. Ramola, *"A comparative study of thorium activity in norm and high background radiation area"*, **Radiation Protection Dosimetry**, September 16, 2010
11. **Chanyotha S.,** Taniguchi M, and Burnett WC., "*Detecting Groundwater Inputs into Bangkok Canals via Radon and Thoron Measurements"*. **Springer book**, Groundwater and Subsurface Environment. (2010)
12. C. Kranrod, S.Tokonami, T.Ishikawa, A. Sorimachi, M. Janik, R. Shingaki, M. Furukawa, **S. Chanyotha,** N. Chankow, "*Mitigation Of The Effective Dose Of Radon Decay Products Through The Use Of An Cleaner In A Dwelling in Okinawa"*, Japan. **Applied Radiation and Isotopes**, 67 (2009) 1127-1132.

13. Chutima Kranrod , **S. Chanyotha,** N. Chankow ,and Shinji Tokonami. "*Simple Technique for Determining the Equilibrium Equivalent Thoron Concentration Using a CR-39 Detector*", Scientific **Journal International (SJI), Radioprotection. Radioprotection,** Vol.44, n° (2009) 301-304, ©EDP Sciences, 2009
14. W.C. Burnett, **S. Chanyotha**, G. Wattayakorn, M. Taniguchi, Y. Umezawa, and T. Ishitobi, "*Underground Sources Of Nutrient Contamination To Surface Waters In Bangkok, Thailand*" **Science of the Total Environment** 407, 3198-3207, (2009).
15. Burnett, W.C., Peter, R., Wattayakorn, G., and **Supitcha Chanyotha.** "*Importance of groundwater discharge in the developing urban centers of southeast Asia*". **From Headwater to the Ocean- : Hydrological Change and watershed Management**. p.289-294, Taylor&francis Group, London, ISBN 978-0-415-47279-1. 2009

Select Conference

1. C. Kranrod, **S. Chanyotha,** N.Chankow, T. Ishikawa, S. Tokonami, **"***Measurement of Radon and Thoron Progeny Size Distributions and Dose Assessments at the Mineral Treatment Industry in Thailand"*, **The Ninth International Conference on Methods and Applications of Radioanalytical Chemistry (MARC IX),** 25-30 March 2012,Kona, Hawii, USA
2. W.C.Burnett, R.N.Peterson, **S.Chanyotha**, G. Wattayakorn, B.Ryan, *"Using high-resolution in-situ radon measurements to determine groundwater discharge at a remote location: Tonle Sap Lake, Cambodia",* **The Ninth International Conference on Methods and Applications of Radioanalytical Chemistry (MARC IX),** 25-30 March 2012,Kona, Hawii, USA
3. **S. Chanyotha**; C. Kranrod; N. Chankow; R. Kritsananuwat; P. Sriploy; K. Pangza, Natural Radionuclide Concentrations In Processed Materials From Thai Mineral Industries **, NARE Symposium,** 29 Feb- 3 Mar, 2012, Hirosaki University, Japan
4. **S. Chanyotha**, C. Kranrod, S. Tokonami, N. Suwankot, T. Pangza, C. Pornnumpa *"Terrestrial Gamma Radiation Dose Rate in Phuket Island"* Thailand**,** The 7th **International Conference on high levels of Natural Radiation and Radon Areas**, 26-24 November 2010, Navi Mumbai, India.
5. **S. Chanyotha**, P.Quinrum and W. C. Burnett. "*A Radon Monitoring Technique for Determinating Areas of Shallow Groundwater Seepage into Waterway*", **International Symposium on Efficient Groundwater Resources Management,** "The Challenge of Quality and Quality for Sustainable Future", Bangkok, Thailand. 16-21 February 2009.
6. Sorimachi A., Kranrod C., **Chanyotha S.**, Tokonami S., "*Development and Evaluation of an Impactor Sampler for Radioactive Aerosol Particles*". **12th**

International Congress of the International Radiation Protection Association-IRPA12, 19-24 October, Buenos Aires, Argentina. 2008.

7. W.C. Burnett, **S. Chanyotha**, G. Wattayakorn, M. Taniguchi, Y. Umezawa, and T. Ishitobi, "*Groundwater as a Pathway for Nutrient Contamination in Bangkok, Thailand*" **Asia Oceania Geosciences Society (AOGS 2007) 4th Annual Meeting**. 30 July - 4 August. Bangkok Thailand (2007).
8. K.Limbanyen, T. Soponkanabhorn, **S. Chanyotha**, N. Chankow and P. Krobbuaban. "*Application of Systematic Technique for the Characterization of Naturally Occurring Radioactive Material (NORM) at Bongkot Field and Songkla Petroleum Development Support Base*". **Asia Pacific Health, Safety, Security and Environment Conference and Exhibition**. Page SPE 108873:1-6, 10-20 September Bangkok Thailand (2007).
9. **S. Chanyotha,** O. Jiradttrakul and C. Yenchai. "*Determination of Naturally Occurring Radioactive Elements in Portland Cement Used for Building Construction*". **Proceedings of the 10th Conference on Nuclear Science and Technology**. Page NE05:1- 12, 16-17 August Bangkok Thailand (2007).

7.4 งานวิจัยที่กำลังทำ:

7.4.1 งานสื่อสารสร้างความรู้ความเข้าใจโครงการโรงไฟฟ้าใหม่ผ่านเว็บไซต์และ SOCIAL MEDIA, การไฟฟ้าฝ่ายผลิตแห่งประเทศไทยม ม.ค.2555-ม.ค.2558

7.1.2 การหาปริมาณธาตุกัมมันตรังสีตามธรรมชาติในเครื่องมือและกากจากอุตสาหกรรมปิโตรเลียม บริษัท ปตท.สผ. จากัด (มหาชน) ม.ค.2555-ม.ค.2558