การปรับปรุงวิธีการเลือกเส้นทางจราจรที่ใช้เวลาน้อยที่สุดโดยใช้ข้อมูลการจราจรในอดีต

จรัสศรี เหลือจันทร์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
บัณฑิตวิทยาลัย สาขาเทคโนโลยีสารสนเทศ
สถาบันเทคโนโลยีไทย-ญี่ปุ่น
ปีการศึกษา 2559

IMPROVEMENT OF LEAST TIME-CONSUMING TRAVEL ROUTE SELECTION BY USING HISTORICAL TRAFFIC DATA

Charasri Leuachan

A Thesis Submitted in Partial Fulfillment of Requirements
for The Degree of Master of Science Program in Information Technology
Graduate School
Thai-Nichi Institute of Technology
Academic Year 2016

หัวข้อวิทยานิพนธ์	การปรับปรุงวิธีการเลือกเส้นทางจราจรที่ใช้เวลาน้อยที่สุดโดยใช้ข้อมูลการจราจรในอดีต
โดย	จรัสศรี เหลือจันทร์
สาขาวิชา	เทคโนโลยีสารสนเทศ
อาจารย์ที่ปรึกษาวิทยานิพนธ์	ผู้ช่วยศาสตราจารย์ ดร. วรากร ศรีเชวงทรัพย์

บัณฑิตวิทยาลัย สถาบันเทคโนโลยีไทย-ญี่ปุ่น อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

.....คณบดีบัณฑิตวิทยาลัย

(รองศาสตราจารย์ ดร. พิชิต สุขเจริญพงษ์)

วันที่.....เดือน.....พ.ศ.....

คณะกรรมการสอบวิทยานิพนธ์

.....ประธานกรรมการ

(ดร. เจนจบ วีระพานิชเจริญ)

.....กรรมการ

(ผู้ช่วยศาสตราจารย์ ดร. วิมล แสนอุ้ม)

.....กรรมการ

(ดร. สะพรั่งสิทธิ์ มฤทุสาธร)

.....อาจารย์ที่ปรึกษาวิทยานิพนธ์

(ผู้ช่วยศาสตราจารย์ ดร. วรากร ศรีเชวงทรัพย์)

จรัสศรี เหลือจันทร์ : การปรับปรุงวิธีการเลือกเส้นทางจราจรที่ใช้เวลาน้อยที่สุดโดยใช้ข้อมูลการจราจรในอดีต. อาจารย์ที่ปรึกษา : ผู้ช่วยศาสตราจารย์ ดร. วรากร ศรีเชวงทรัพย์, 89 หน้า.

สภาพการจราจรในปัจจุบันในการสัญจรของผู้คนที่ใช้รถยนต์ในการเดินทาง ทั้งรถยนต์ส่วนบุคคลและรถสาธารณะที่มีเป็นจำนวนมาก ทำให้ท้องถนนเต็มไปด้วยยานพาหนะต่างๆ ซึ่งส่งผลให้เกิดการติดขัดโดยเฉพาะในเวลาที่เร่งด่วนที่ทุกคนต่างออกมาพร้อมกัน และก็ยังมีปัจจัยอื่นที่อาจทำให้การจราจรมีสถานะติดขัดมากกว่าเดิม เช่น สัญญาณไฟจราจรที่การเปลี่ยนแต่ละสัญญาณที่อาจไม่สัมพันธ์กัน, การซ่อมแซมผิวจราจร, เส้นทางที่ถนนมีความคับแคบ หรือการเกิดอุบัติเหตุ เป็นต้น ยิ่งทำให้ไม่สามารถใช้เส้นทางที่เคยใช้ได้ จึงทำให้ต้องเสียเวลาในการเดินทางยาวนานมากขึ้น งานวิจัยนี้ทำการศึกษาและสร้างการค้นหาเส้นทางที่ใช้เวลาน้อยที่สุดเพื่อช่วยในการตัดสินใจการเลือกเส้นทางสำหรับการเดินทาง โดยนำข้อมูลเส้นทางและพิกัดของแต่ละสถานที่มาจาก Google Map และข้อมูลการเดินทางในอดีตที่ผู้ใช้เส้นทางนั้นๆ นำมาประมวลผลเพื่อทำการคาดคะเนเวลาที่จะใช้ในการเดินทางครั้งนั้นให้ความแม่นยำมากขึ้น และนำมาพัฒนาโปรแกรมประยุกต์การหาเส้นทางที่ใช้เวลาในการเดินทางที่น้อยที่สุด บนระบบปฏิบัติการแอนดรอยด์

บัณฑิตวิทยาลัย ลายมือชื่อนักศึกษา .....

สาขาวิชา เทคโนโลยีสารสนเทศ ลายมือชื่ออาจารย์ที่ปรึกษา .....

ปีการศึกษา 2559

CHARASRI LEUACHAN : IMPROVEMENT OF LEAST TIME-CONSUMING TRAVEL ROUTE SELECTION BY USING HISTORICAL TRAFFIC DATA. ADVISOR : ASST. PROF. DR. WARAKORN SRICAHVENGSUP, 89 PP.

The number of both private cars and public vehicles on the road are numerous. This makes the road full of vehicles and results in a jam, especially in the rush hour that everyone comes out at the same time. Moreover, there are other factors that may cause traffic congestion, such as traffic lights that may not work properly, road surface repairs, roads that are cramped or accidents. For these reasons, it is impossible to use the route that was available. So it takes more time to travel. This research offers planning to choose the route to the destination in the quickest and shortest possible time through the mobile application. The mobile application in android platform uses maps that are derived from the google map and past historical data to calculate the time it takes in order to plan a trip, reduce time or avoid routes that may cause travel delay.

Graduate School Student's Signature.....

Field of Information Technology Advisor's Signature.....

Academic Year 2016

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้ที่สามารถสำเร็จได้ด้วยความกรุณาของอาจารย์ ผู้ช่วยศาสตราจารย์ ดร. วรากร ศรีเชวงทรัพย์ อาจารย์ที่ปรึกษาที่กรุณาให้คำปรึกษามาตั้งแต่ต้นตั้งแต่เรียนในวิชาของอาจารย์ที่เข้าใจถึงปัญหาแต่ยังติดตามและให้โอกาส รวมถึงให้กำลังใจ ให้คำแนะนำในหลายๆ ด้าน ให้แนวคิดให้ความรู้สำหรับทุกคำปรึกษา ที่ทำให้ผลงานเสร็จสมบูรณ์ลงได้ผู้วิจัยขอกราบขอบพระคุณท่านอาจารย์อย่างยิ่งยวด

ผู้วิจัยขอกราบขอบพระคุณคณะกรรมการสอบทุกท่าน ประธานกรรมการ ดร. เจนจบ วีระพานิชเจริญ ที่แม้จะต้องเดินทางลำบากเพื่อมาทำหน้าที่ด้วยความปราณีและมาให้ความรู้ให้ข้อแนะนำในห้องสอบ ผู้ช่วยศาสตราจารย์ ดร. วิมล แสนอุ้ม ที่แม้มีภารกิจมากมายแต่ยังเสียสละมาให้คำแนะนำโดยเฉพาะครั้งนี้ทำให้ผู้วิจัยรู้ถึงสิ่งที่ต้องตะหนักเพิ่มขึ้นและมีแนวทางจะทำคือการสื่อสารและการจัดกระบวนการความคิดที่ผู้วิจัยไม่เคยคิดว่ามันต้องแก้ไขแค่ทำงานเสร็จแต่อาจารย์สอนให้เข้าใจว่ามันสำคัญมาก และท่านสุดท้าย คือ ดร. สะพรั่งสิทธิ์ มฤทุสาธร อาจารย์ท่านแรกที่เจอครั้งแรกในสถาบันแห่งนี้และผู้วิจัยรู้สึกได้ว่าความกลัวบางอย่างที่เราไม่ไปพบท่านแต่จริงๆ ทุกครั้งที่ได้สนทนากลับได้การแนะนำที่ดีมาตลอดและอาจารย์ยังสอนให้ก้าวข้ามสิ่งที่กลัวคือภาษาอังกฤษที่ผู้วิจัยคิดว่าต้องผ่านมาให้ได้

ผู้วิจัยขอกราบขอบพระคุณ บิดา มารดา ที่สอนให้เรียนรู้ไม่หยุดนิ่ง ให้ทำทุกสิ่งให้เต็มที่และสามี ลูกชาย ที่ให้กำลังใจอย่างดียอมเสียสละเวลาให้ผู้วิจัยได้ทำหน้าที่มาตลอด พี่ น้อง เพื่อนๆ และกลุ่ม Android Dev Github, Android Droidcon Developer Library Github, เว็บไซต์ Akexorcist ที่ให้คำปรึกษาในการพัฒนาโปรแกรมด้วยดี สุดท้ายไม่กล่าวถึงไม่ได้คือ เพื่อนๆ พี่ๆ น้องๆ MIT รุ่น 2 ที่ฟันฝ่าร่วมแรงร่วมใจเล่าเรียนและให้คำแนะนำต่างๆ จนทำให้มีโอกาสทำให้งานวิจัยชิ้นนี้ให้สำเร็จลุล่วงได้

สุดท้ายนี้ผู้วิจัยหวังว่าวิทยานิพนธ์ฉบับนี้จะเป็นประโยชน์ในการนำไปใช้ในอนาคตต่อไป

ขอขอบคุณอีกครั้ง

จรัสศรี เหลือจันทร์

สารบัญ

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญรูป

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

จากสภาพการจราจรในปัจจุบันที่ผู้คนต่างต้องใช้ถนนหนทางในการเดินทางด้วยจุดประสงค์ต่างๆ ไม่ว่าจะเป็นการเดินทางเพื่อทำงาน ไปเรียน หรือเพื่อการท่องเที่ยว ในการเดินทางจะต้องใช้ยานพาหนะในการเดินทาง ที่มีอยู่หลายรูปแบบไม่ว่าจะเป็นรถนั่งส่วนบุคคลหรือว่าจะเป็นรถที่ให้บริการสาธารณะ ซึ่งความต้องที่จะไปยังจุดหมายปลายทางที่รวดเร็วและระยะทางที่สั้นที่สุด แต่ในบางครั้งอาจไม่เป็นอย่างที่ต้องการได้อันเนื่องจากยวดยานต่าง ๆ ที่คับคั่งเต็มอยู่บนท้องถนน จึงทำให้การขับขี่หรือการเดินทางเป็นไปได้อย่างช้าๆ และอาจหยุดนิ่งอยู่นาน โดยเฉพาะกรณีที่มีอุบัติเหตุเกิดขึ้น หรือการซ่อมผิวการจราจรหรือปิดการใช้ช่องทางไปบางส่วน ยิ่งทำให้ไม่สามารถใช้เส้นทางที่เคยใช้ได้ ก็ยิ่งทำให้การจราจรติดขัดเพิ่มขึ้นใช้เวลาในท้องถนนยาวนาน ทำให้ต้องมีการปรับเปลี่ยนเส้นทางซึ่งอาจมีระยะทางที่ไกลกว่าเดิม อีกทั้งหากเป็นเส้นทางที่รถมีปริมาณการสะสมอยู่มาก สำหรับคนที่ต้องเดินทางจะต้องมีการเผื่อเวลาสำหรับการเดินทางเพิ่มจากเดิมค่อนข้างมาก โดยเฉพาะเวลาเร่งด่วนที่ทุกคนต่างออกมาพร้อมกัน

รูปที่ 1.1 สภาพการจราจรที่ติดขัด

ทั้งนี้ ผู้ขับขี่จึงต้องการหาเส้นทางที่ใช้เวลาที่รวดเร็วและสั้นที่สุดในการเดินทางไปแต่ละจุดหมายปลายทาง เพื่อลดเวลาและต้องการหลีกเลี่ยงเส้นทางที่อาจทำให้การเดินทางเกิดความล่าช้า สำหรับปัจจัยที่ทำให้เกิดความล่าช้านั้นมีอยู่ด้วยกันหลายอย่าง ที่ทำให้การจราจรเกิดความไม่คล่องตัว ซึ่งการวัดว่าสภาพการจราจรที่ผู้ขับขี่อยู่ในเส้นทางนั้นดีกว่าเส้นทางอื่นอยู่แล้วหรือไม่ หรือมีเส้นทาง

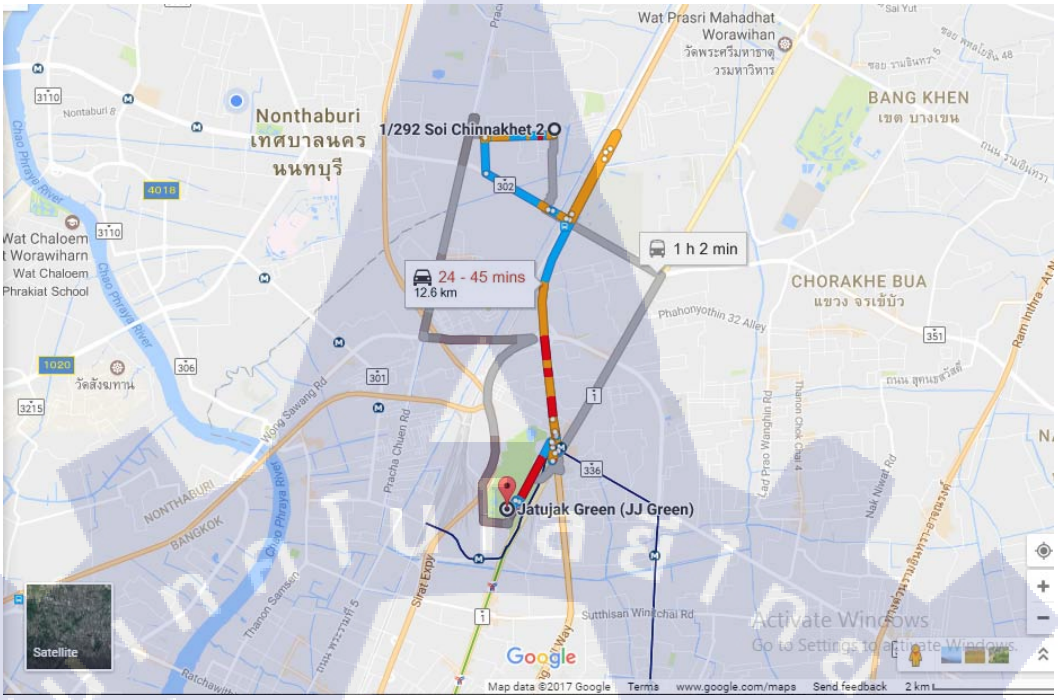
อื่นๆ ที่ดีกว่า จึงเป็นสิ่งที่จำเป็นต้องหาเครื่องมือในการค้นหาเส้นทาง แต่ในส่วนหนึ่งผู้ขับขี่มักจะใช้ประสบการณ์จากการเดินทางตามเส้นทางที่เคยไปตามช่วงเวลาที่เคยใช้ หรือการใช้อุปกรณ์ GPS หรือป้ายไฟจราจรที่จะแสดงสภาพการจราจรตามสีให้ผู้ขับขี่ใช้เป็นจุดสังเกต แต่จากข้อสังเกตที่กล่าวมาก็ไม่สามารถรับประกันได้ว่าสภาพการจราจรหรือเส้นทางที่เคยใช้จะเป็นเหมือนเดิม จึงมีงานวิจัยหลายๆ งานวิจัยที่พยายามที่จะพัฒนาและหาข้อมูลเพื่อนำมาใช้ในการสร้างเครื่องมือสำหรับใช้ในการเดินทางจากจุดหนึ่งไปยังอีกจุดหนึ่ง

การตั้งมาตรฐานการประเมินสภาพการจราจรและประสิทธิภาพของถนนเป็นงานหนึ่งที่ที่ได้มีการดำเนินการจัดทำขึ้นโดยสถาบัน Highway Capacity Manual (HCM) ในปีค.ศ. 1965 โดยแบ่งออกเป็นระดับ (LOS : Level of Service) [1] ซึ่งเป็นมาตรวัดในเชิงคุณภาพ (Qualitative Measure) แบ่งออกเป็น 6 ระดับ ได้แก่ A ,B, C, D, E และ F ค่าแต่ละค่าจะแสดงถึงลักษณะและสภาพการจราจรที่แตกต่างกัน โดยระดับ A (LOS A) เป็นสภาพการจราจรที่ดีที่สุด และ F (LOS F) จะเป็นสภาพการจราจรที่แย่ที่สุด

ตารางที่ 1.1 การกำหนดค่าระดับ LOS : A-F ตามมาตรฐานของ HCM

LOS	Signalized Intersection	Unsignalized Intersection
A	$\leq$ 10 sec.	$\leq$ 10 sec.
B	10-20 sec.	10-15 sec.
C	20-35 sec.	15-25 sec.
D	35-55 sec.	25-35 sec.
E	55-80 sec.	35-50 sec.
F	$\geq$ 80 sec.	$\geq$ 50 sec.

หรือในตัว Application Google Map ที่ใช้กันอย่างแพร่หลายเองก็ยังมีการวัดสภาวะการจราจร โดยการบ่งบอกเวลาโดยประมาณในขณะนั้น และลักษณะของสีตามสภาวะของเส้นทางในแต่ละเส้นทาง ตามที่แสดงในรูปที่ 1.2



รูปที่ 1.2 เส้นทางการจราจรและสถานการณ์จราจรในขณะนั้น

ดังนั้น สำหรับผู้ขับขี่แล้วการมีตัวช่วยในการวางแผนการเดินทางหรือการปรับเปลี่ยนเส้นทางระหว่างการเดินทางจึงมีประโยชน์มาก งานวิจัยฉบับนี้จึงทำการศึกษาและสร้างการค้นหาเส้นทางที่ใช้เวลาน้อยที่สุดเพื่อช่วยในการตัดสินใจการเลือกเส้นทางสำหรับการเดินทาง โดยการศึกษาครั้งนี้จะนำข้อมูลเส้นทางและพิกัดของแต่ละสถานที่มาจาก Google Map และข้อมูลการเดินทางในอดีตที่ผู้ใช้เส้นทางนั้นๆ นำมาประมวลผลเพื่อทำการคาดคะเนเวลาที่จะใช้ในการเดินทางครั้งนั้นให้ความแม่นยำมากขึ้น และนำมาพัฒนาโปรแกรมประยุกต์การหาเส้นทางที่ใช้เวลาในการเดินทางที่น้อยที่สุด บนระบบปฏิบัติการแอนดรอยด์

1.2 วัตถุประสงค์การศึกษา

1.2.1 เพื่อทำการประมวลผลข้อมูลการหาเส้นทางที่ใช้เวลาน้อยที่สุดจากสถานที่หนึ่งไปยังอีกสถานที่หนึ่งโดยใช้ข้อมูลในอดีตในการทำนายเวลาในการเดินทาง

1.2.2 เพื่อศึกษาความสัมพันธ์ของตัวแปรและนำมาสร้างแบบจำลองเพื่อค้นหาเส้นทางที่ใช้เวลาน้อยที่สุดตามจุดหมายปลายทางที่ระบุ

1.3 กรอบแนวคิดการศึกษา

การศึกษาในครั้งนี้เป็นการหาเส้นทางที่ใช้เวลาน้อยที่สุดในการเดินทางโดยการนำข้อมูลประวัติการเดินทางที่ผ่านมาทำการประมวลผลกับแผนที่เส้นทางที่ได้จาก Google Map และหาความสัมพันธ์ของตัวแปรที่จะนำมาสร้างแบบจำลองสำหรับการค้นหาเส้นทางก่อนนำมาสร้างเป็นโปรแกรมประยุกต์บนอุปกรณ์มือถือบนระบบปฏิบัติการแอนดรอยด์ ซึ่งข้อมูลที่จะนำมาใช้ในการค้นหาเส้นทางนั้นสามารถจะมีได้หลากหลายวิธี หนึ่งในนั้นคือการเก็บข้อมูลจากวิดีโอของกล้องวงจรปิดในเส้นทางต่างๆ ที่ผู้วิจัยได้นำเสนอก่อนหน้า แต่จากการประมวลผลพบว่าอาจมีข้อมูลไม่เพียงพอสำหรับใช้ในการวิเคราะห์และเส้นทางที่ใช้ไม่หลากหลาย จึงศึกษาเพิ่มเติมและพบว่าข้อมูลเส้นทางต่างๆ และข้อมูลพื้นฐานที่จะนำไปใช้ สามารถนำมาจาก Google Map ได้ ซึ่งจะทำให้ข้อมูลมีเพียงพอในการวิจัย

1.4 ขอบเขตของการศึกษา

ศึกษาเส้นทางการจราจรในพื้นที่กรุงเทพมหานคร ข้อมูลเส้นทางการจราจรจากพิกัดแผนที่ที่ได้จาก Google Map และนำข้อมูลการเดินทางในอดีตทำการประมวลผลในการหาเส้นทางที่ใช้ระยะเวลาการเดินทางน้อยที่สุด

1.5 ขั้นตอนการวิจัย

1.5.1 ศึกษากระบวนการทำงาน ทฤษฎีด้านการจราจรและ งานวิจัยที่เกี่ยวข้อง

1.5.2 คัดเลือกเส้นทางที่จะทำการศึกษาทดลอง

1.5.3 เก็บข้อมูลถนนที่คัดเลือก เพื่อเก็บข้อมูลการจราจรตามเวลาจริง

1.5.4 ศึกษาวิเคราะห์ตัวแปรต่างๆ ของเส้นทางที่ใช้เวลาน้อยที่สุด ในการสร้างแบบจำลองผ่านโปรแกรมประยุกต์สำหรับการเดินทาง

1.5.5 สรุปผลการทดลอง และอภิปรายผล

1.6 ประโยชน์ที่คาดว่าจะได้รับ

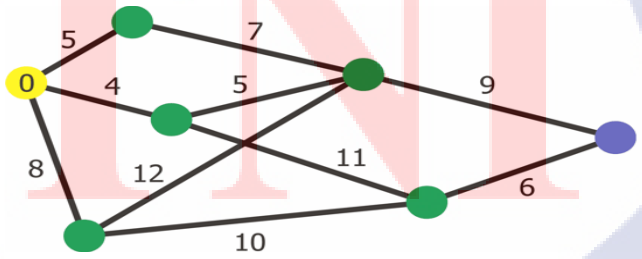
ได้โปรแกรมสำหรับการค้นหาเส้นทางจราจรในการเดินทางจากสถานที่หนึ่งไปยังอีกสถานที่โดยใช้เวลาน้อยที่สุด และเป็นเครื่องมือสำหรับช่วยผู้ขับขี่วางแผนการเดินทางจากสภาพการจราจรที่ใกล้เคียงที่สุดในช่วงเวลาต่างๆ รวมถึงการพัฒนาเพื่อประสิทธิภาพโปรแกรมจากปัจจัยอื่นๆ ที่มีผลกับการเลือกเส้นทาง

บทที่ 2
ทบทวนวรรณกรรม

การศึกษาครั้งนี้ เป็นการศึกษาโดยการทดลองที่นำเอาทฤษฎี และงานวิจัยต่างๆ ที่เกี่ยวข้องกับกรอบแนวความคิด การหาเส้นทางที่ใช้เวลาน้อยที่สุด โดยใช้ข้อมูลเส้นทางแผนที่รวมถึงพิกัดที่ได้จาก Google Map และ Function ในการทำงานที่ทาง Google ได้จัดทำขึ้นสำหรับนักพัฒนาที่จะนำไปจัดทำโปรแกรมประยุกต์ที่จะใช้แผนที่ในการทำงาน รวมกับข้อมูลเส้นทางที่ใช้ในอดีตนำมาประมวลผลสำหรับการหาเส้นทางที่ใช้เวลาน้อยที่สุดในการเดินทางเพื่อวางแผนในการเดินทาง โดย และในงานวิจัยนี้จะนำเอาข้อมูลเส้นทางที่เคยใช้ในอดีตมาทำการประมวลผลเพื่อหาเส้นทางที่ใช้เวลาน้อยที่สัด โดยมีเกณฑ์การประเมินจากเส้นทางที่ได้เทียบกับข้อมูลเส้นทางระยะเวลาที่ได้จาก Google Map มาทำการทดสอบ

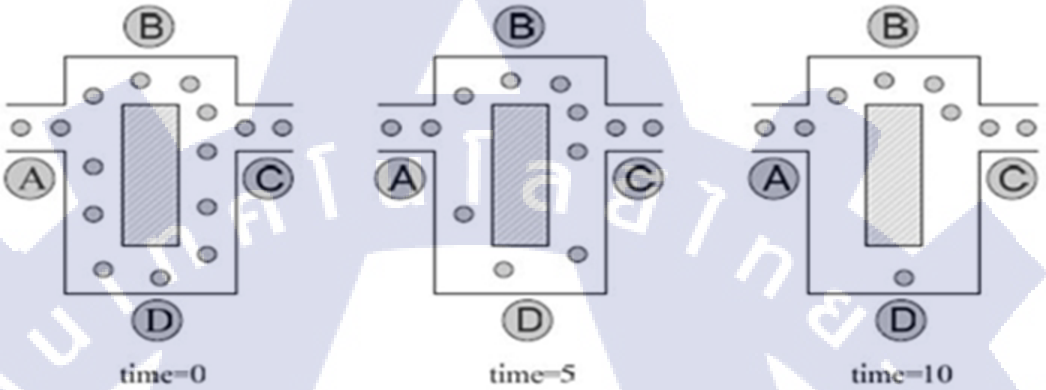
2.1 ทฤษฎีการค้นหาเส้นทาง

1. การหาเส้นทางที่สั้นที่สุด (Shortest Path Algorithm) ทฤษฎีการหาเส้นทางที่สั้นที่สุดเป็นการหาเส้นทางรวมที่มีค่าใช้จ่ายในการดำเนินการที่น้อยสุดที่การทำงานจะเป็นการเปรียบเทียบเส้นทางในแต่ละเส้นทาง โดยการหาเส้นทางอาจวัดจากค่าใช้จ่ายต่างๆ ซึ่งการค้นหาเส้นทางที่สั้นที่สุดซึ่งอัลกอริทึมที่กล่าวถึงและใช้กันโดยมาก คือ Dijkstra's algorithm คิดค้นขึ้นโดยนักวิทยาการคอมพิวเตอร์ชาวดัตช์นามว่า แอ็ดส์เคอร์ ไดก์สตรา(Edsger Dijkstra) เพื่อแก้ไขปัญหาการหาเส้นทางที่สั้นที่สุดจากจุดหนึ่งใดๆ ขั้นตอนวิธี Dijkstra's Algorithm นี้จะหาระยะทางสั้นที่สุดจากจุดหนึ่งไปยังจุดใดๆ ในกราฟโดยจะหาเส้นทางที่สั้นที่สุดไปทีละจุดยอดเรื่อยๆ จนครบตามที่ต้องการ



รูปที่ 2.1 การทำงานโดยใช้ Dijkstra's Algorithm

2. การหาเส้นทางแบบมด (Ant Colony) อัลกอริทึมแบบระบบมด (Ant Colony Algorithm) เป็นอัลกอริทึมที่จำลองหรือได้แนวคิดมาจากการหาอาหารของมด มดจะอาศัยสารเคมีที่เรียกวาฟีโรโมน (Pheromone) ที่มดแต่ละตัว จะปล่อยออกมาเพื่อให้มดตัวหลังได้รู้เส้นทาง ซึ่งมดที่ตามมาก็จะทำการปล่อยฟีโมนเพื่อให้มดตัวต่อๆ ไปได้รับรู้ถึงเส้นทาง เป็นการปรับปรุงฟีโรโมนเส้นทางเหตุนี้เอง ฟีโรโมนจึงเป็นข้อมูลที่สำคัญในการหาเส้นทางจากแหล่งอาหารกลับและย้อนกลับมารัง ดังนี้



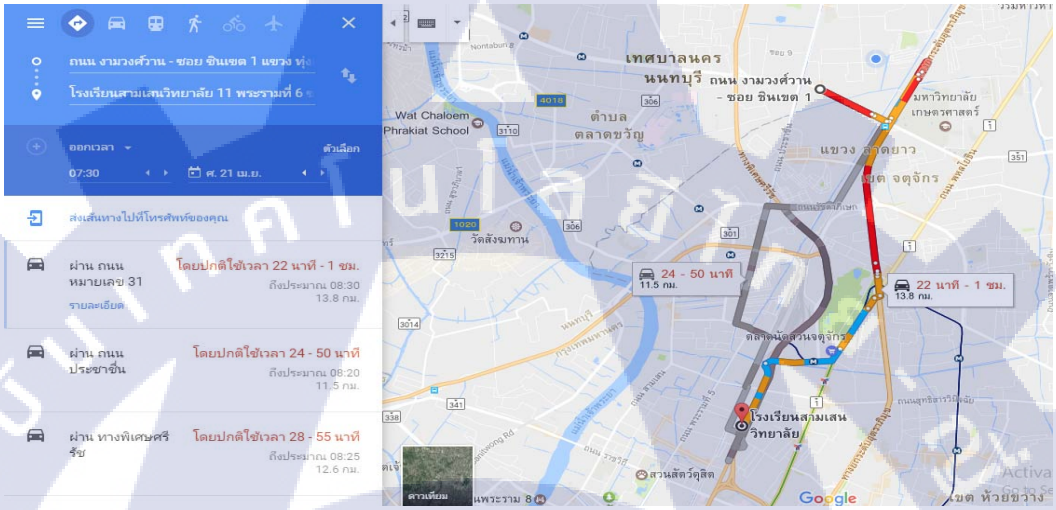
รูปที่ 2.2 วิธีการเดินของมด

อัลกอริทึมวิธีแบบมดจะสามารถหาเส้นทางได้มากกว่า 1 เส้นทาง ทำให้ได้เส้นทางที่หลากหลาย ซึ่งจะต่างจากการหาเส้นทางแบบ Shortest Path Algorithm จะได้เส้นทางที่สั้นที่สุดที่เป็นเส้นทางเดียว แต่ในการเลือกเส้นทางแบบมด การเดินทางจะได้เส้นทางในหลายๆ เส้นทางก็จะสามารถทำการวิเคราะห์โดยไม่ต้องทำการคำนวณเส้นทางใหม่ได้

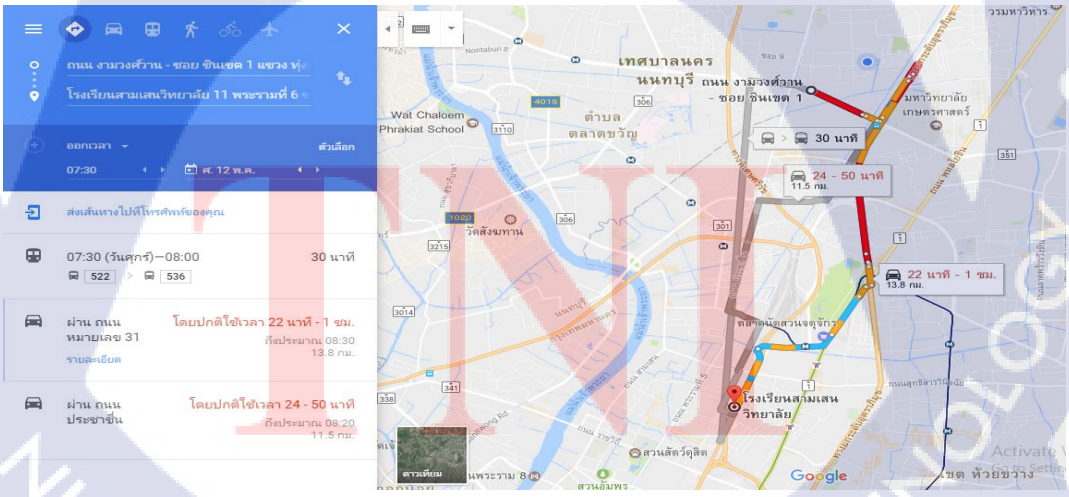
2.2 Google Map, Google API และ Location Base Service

1. Google Maps [2] เป็นบริการของ Google ที่ให้ข้อมูลทางด้านแผนที่และรายละเอียดสถานที่ตั้งต่างๆ โดยระบุพิกัดที่ตั้งทางภูมิศาสตร์ การใช้งานไม่จำเป็นต้องทำการ ลงโปรแกรมใดๆ ซึ่งเราสามารถหาข้อมูลต่างๆ ใน Google Map ได้ เช่น ข้อมูลเกี่ยวกับสถานที่ เส้นทางและใช้การนำทาง เป็นต้น สำหรับการใช้งานทาง Google เองได้จัดให้บริการสำหรับผู้ใช้และสำหรับนักพัฒนาในการนำข้อมูลแผนที่ไปใช้งานต่างๆ ในการนี้ผู้ใช้เพียงแต่ต้องมีการใช้อินเตอร์เน็ตที่สามารถเชื่อมสัญญาณเพื่อติดต่อไปยัง Server ที่อาจจะเป็น WIFI หรือ ผู้ให้บริการสัญญาณเครือข่ายต่างๆ และรวมถึงต้องมีการเปิดใช้บริการการขอเส้นทางด้วย GPS (Global Position System) ซึ่งจะทำการบอกระบุพิกัดที่ต้องของตัวอุปกรณ์ที่ใช้งาน

2. การทำงานของ Google Map จะเป็นการทำงานที่แจ้งเส้นทางที่เป็นไปได้และสั้นที่สุด โดยใช้เวลาเป็นเกณฑ์ในการสร้างเส้นทางในเวลา ณ ขณะนั้นทั้งเส้นทาง เมื่อผู้ใช้มีการร้องขอเส้นทาง และจะทำการสร้างเส้นทางใหม่เมื่อถึงจุดที่อาจพบความผิดปกติหรือเส้นทางเปลี่ยนไปจากเดิม สำหรับการวางแผนการเดินทางล่วงหน้า Google Map จะแสดงเวลาในอดีตของช่วงเวลาที่ผ่านมาซึ่งจะแจ้งระยะเวลาเดียวกับอดีต ณ ช่วงเวลาเดียวกัน



รูปที่ 2.3 การแสดงเส้นทางในช่วงเวลาในอดีต



รูปที่ 2.4 การแจ้งแสดงเส้นทางในช่วงเวลาในอนาคต

2.1 Google Application Programming Interface (Google API) [3] เป็นชุดของโปรแกรมคำสั่งหรือฟังก์ชัน ที่ใช้ในการทำงานสำหรับนักพัฒนา web application และ mobile application (Android, iOS) เพื่อเรียกใช้แผนที่และชุดคำส่งต่างๆ ให้เหมือนกับที่ Google ทำงาน โดยมีบริการและรูปแบบต่างๆ เช่น การควบคุมแผนที่ การคำนวณและการแปลงพิกัดให้เป็นที่ตั้ง มาใช้สำหรับงานของเราเอง เป็นต้น ซึ่งในงานวิจัยนี้จะนำ Google API ที่จะนำมาใช้คือ

2.1.1 Direction API เป็นบริการที่จะส่งค่าชุดเส้นทางและทิศทาง ที่เชื่อมโยงจุดในแต่ละจุด ที่ประกอบไปด้วยวิธีการใช้การขนส่ง จุดเชื่อมและเวลาที่ใช้ในการเดินทางที่เป็นค่าประมาณการที่จะใช้ที่ได้มาจากประวัติเดิม

2.1.2 Geocoding API เป็นชุดคำสั่งการทำพิกัดทางภูมิศาสตร์ของ Google แผนที่เป็นบริการที่มีการระบุพิกัดทางภูมิศาสตร์และการเข้ารหัสของตำแหน่งที่อยู่

2.2 Location Base Service (LBS) เป็นบริการการบอกตำแหน่งที่ใช้ประโยชน์จากเทคโนโลยีเครือข่ายไร้สาย เพื่อวัตถุประสงค์ในการระบุตำแหน่งที่อยู่ของผู้ใช้อุปกรณ์ไร้สายได้อย่างแม่นยำและเป็นปัจจุบัน แบ่งเป็นการให้บริการ 2 แบบ คือ

2.2.1 Pull Services เป็นโปรแกรมการให้บริการต่างๆ ที่ผู้ใช้สามารถดึงมาใช้ได้ตามต้องการ เช่น บริการเรียกรถบริการสาธารณะ เป็นต้น

2.2.2 Push Services ข้อมูลต่างๆ จะถูกส่งโดยมีการร้องขอ หรือ ไม่มีการร้องขอก็ตามจากผู้ใช้บริการ โดยปกติบริการจะเริ่มทำงานเมื่อผู้ใช้เข้าสู่บริเวณที่กำหนด หรือ ตามเวลาที่ตั้งไว้ ตัวอย่างที่เห็นง่ายๆ คือ บริการโฆษณาสินค้าลดราคา โดยองค์ประกอบในการทำงานของ

<u>LBS จะประกอบไปด้วยส่วนที่สำคัญ</u> คือ

1. Mobile Devices คือ อุปกรณ์เคลื่อนที่ที่ใช้ในการสื่อสารและการบอกตำแหน่งของอุปกรณ์เพื่อนำไปประมวลผลในการใช้งานต่างๆ

2. Communication Network การเชื่อมต่อระหว่าง Mobile กับ Data Center ที่มีอยู่หลายแบบ เช่น GPS, WIFI เป็นต้น

3. Positioning Component คือ การให้บริการระบุตำแหน่งอุปกรณ์ Mobile ซึ่งจะต้องใช้ความสามารถของระบบบริหารจัดการเครือข่ายไร้สายที่มีความสามารถสูงในการควบคุมการ Monitor Location

4. Service and Application Provider คือ บริการรับส่งข้อมูลให้กับ Mobile ซึ่งเป็นผู้ใช้ตามเงื่อนไขต่างๆ

5. Data and Content Provider คือ คลังข้อมูลหรือคอนเท็นต์ต่างๆ สำหรับการบริการ เช่น โปรโมชั่น หรืออื่นๆ ที่มีการกำหนดไว้สำหรับ ให้บริการให้กับผู้ใช้ (Mobile)

2.3 ทฤษฎีด้านการจราจร

1. ถนนในช่วงเวลาหนึ่ง ซึ่งจะเป็นตัวชี้วัดสำหรับการนำมาคำนวณความสามารถในการให้บริการของเส้นทางในถนนประเภทต่างๆ เพื่อใช้แบ่งสภาพการจราจรที่ชัดเจน การวิเคราะห์และประเมินการรับรู้สภาพการจราจรที่แตกต่างกันในแต่ละช่วงเวลาจะมีความแตกต่างกัน ทั้งนี้มีการแบ่งระดับการจราจรด้วยความเร็วของยานพาหนะขณะขับขี่ซึ่งมีการจัดทำเป็นคู่มือตัวชี้วัดประสิทธิภาพตาม HCM โดยตรวจวัดจาก อัตราส่วนระหว่างปริมาณ การจราจรกับความสามารถในการให้บริการของถนน (V/C Ratio) ความล่าช้าเฉลี่ยของแต่ละทางแยก และค่าระดับการให้บริการของถนน (Level Of Service, LOS) ด้วยมาตรวัดประสิทธิผลสองตัวประกอบด้วยความเร็วเฉลี่ยในการเดินทาง (Average Travel Speed, ATS) และร้อยละของเวลาที่ใช้ในทางปฏิบัติได้กำหนดระดับบริการออกเป็น 6 ระดับจากระดับ A ถึงระดับ F โดยระดับบริการ A หมายถึงสภาพการใช้งานดีมากและระดับบริการ F เป็นสภาพการใช้งานแย่สุด

2. คุณลักษณะของถนนและสภาพการจราจร จากสภาพของการจราจรมีปัจจัยในหลายๆ ตัวที่ส่งผลกับการใช้งานบนท้องถนนสามารถที่จะเดินทางได้อย่างสะดวก รวดเร็ว หรือเกิดการติดขัดไม่ว่าจะเป็นทางตรงหรือทางอ้อม ซึ่งจะประกอบไปด้วย

2.1 ขนาดของท้องถนน คือ ความกว้างและความยาวของเส้นทางที่ใช้ ที่จะเป็นตัวแปรหนึ่งที่จะทำให้ความจุของถนนที่ใช้มีขนาดเท่าไร

2.2 สภาพของถนนที่ใช้ ลักษณะทางกายภาพของแต่ละช่วงถนน สภาพถนนเป็นปัจจัยอีกส่วนที่มีผลกับการคำนวณเส้นทางและน้ำหนักของเส้นทางเพราะเส้นทางที่มีเป็นปกติ ไม่เป็นพื้นถนนที่เป็นหลุมเป็นบ่อ อยู่ในเขตการซ่อมแซมพื้นผิวถนน หรือถนนใหม่ ย่อมจะทำให้การเดินทางการจราจรเป็นไปอย่างสะดวกและรวดเร็ว หรือในสรูปที่ ถนนลื่นที่ เนื่องมากจากสภาวะอากาศที่มีฝนตกก็เป็นส่วนที่ส่งผลให้การจราจรเดินทางได้ไม่ดีเท่าที่ควรเพราะจะเกิดจากลื่นไถลได้ง่ายเพราะล้อรถยนต์จะเกาะถนนได้ไม่ดีเหมือนในเวลาปกติซึ่งก็อาจ ส่งผลให้เกิดการชนของยานพาหะที่วิ่งกันมาด้วยความเร็ว และอาจเกิดอุบัติเหตุขึ้นได้

2.3 ปัจจัยที่จะทำให้เกิดสภาวะที่รถไม่สามารถเคลื่อนที่ได้

2.3.1 กรณีที่มีสัญญาณไฟจราจร เป็นปัจจัยที่สามารถนำมาประมินำหรับการชี้วัดในโอกาสที่รถยนต์จะติดขัดได้เพราะเป็นเวลาที่สามารถรู้ได้ในแต่ละจุดที่มีสัญญาณไฟ

2.3.2 กรณีที่ไม่มีสัญญาณไฟจราจร เป็นปัจจัยที่ต้องใช้ค่าเฉลี่ยในการประเมินเส้นทาง เนื่องจากโอกาสในการที่รถจะชะลอตัวหรือหยุดนิ่งไม่สามารถวัดได้เนื่องจากมีเวลาที่นำมาใช้ไม่ได้เหมือนจุดที่มีสัญญาณไฟจราจรที่จะมีเวลาที่แน่นอนกว่า

2.3.3 ทางแยกหรือจุดกลับรถ เป็นอีกปัจจัยที่ทำให้เกิดการชะลอตัวของรถ เนื่องจากหากมีจุดกลับรถจำนวนมาก การชะลอตัวก็สูง เนื่องจากรถยนต์แต่ละประเภทแต่ละรุ่นใช้อัตราที่แตกต่างกัน รวมถึงพฤติกรรมของผู้ขับขี่ต่างกัน

2.3.4 การเกิดอุบัติเหตุ ในส่วนนี้เป็นปัจจัยที่ไม่สามารถทำการคาดเดาได้เนื่องจากการเกิดของสาเหตุเป็นเหตุการณ์ที่เกิดในทันทีทันใดของขณะช่วงเวลาหนึ่งเท่านั้น

2.4 งานวิจัยที่เกี่ยวข้อง

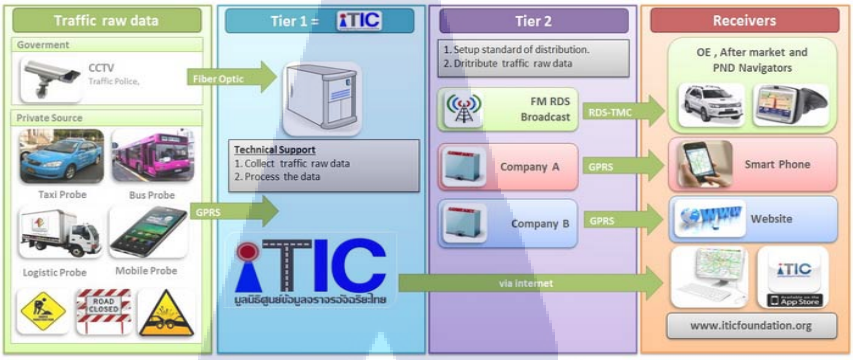
1. งานวิจัยด้านการจราจร

“BMA live Traffic Application” [4] สำหรับการตรวจสอบปัญหาการจราจร ติดขัด ซึ่งเป็นความร่วมมือ ระหว่าง กรุงเทพมหานคร บริษัทรูคอร์เปอร์เรชั่น จำกัด และ มูลนิธิศูนย์ข้อมูลจราจรอัจฉริยะไทย (ITIC) ที่ร่วมมือดำเนินการเชื่อมต่อข้อมูลสัญญาณจราจรจากกล้อง CCTV ในเขตกรุงเทพมหานคร การทำงานจะเป็นเชื่อมต่อสัญญาณจากกล้อง CCTV แบบ Real Time และแบ่งถนนแต่ละเส้น เป็นสี โดยแบ่งออกเป็น 3 สี คือ

- สีแดง สำหรับเส้นทางที่ความเร็วต่ำกว่า 15 กม./ชม. (รถติดหนาแน่น)
- สีเหลือง สำหรับเส้นทางที่ความเร็วต่ำกว่า 25 กม./ชม. (รถติดพอเคลื่อนตัวได้)
- สีเขียว สำหรับเส้นทางมีที่ความเร็วมากว่า 25 กม./ชม. (การจราจรคล่องตัว)

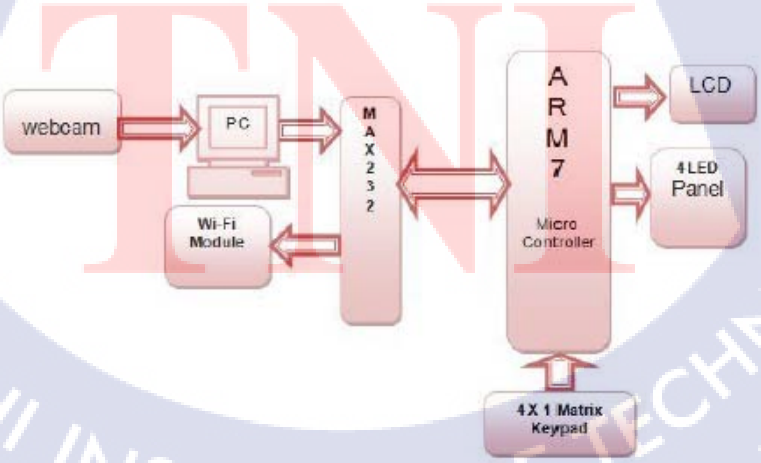
การทำงานจะใช้ Mobile Probe มาเป็นเครื่องตรวจวัดเคลื่อนที่ ในที่นี้ จะเป็นการวัดความเร็ว ตำแหน่ง และ เวลาของรถยนต์ เพื่อนำมาประมวลผล เป็นข้อมูล จราจร (Mobile Probe เป็น Data Input ที่รับจากอุปกรณ์โดยตรงที่ตรวจวัดได้) ข้อดีของ Mobile Probe เมื่อเทียบกับ Fixed Sensor คือ Mobile Probe ครอบคลุมพื้นที่มากกว่า ค่าใช้จ่ายถูกกว่า

Fixed Sensor เพราะโดยปกติ Fixed Sensor จะติดตั้งในบริเวณที่สำคัญ และบริเวณที่การจราจรคับคั่ง ข้อมูลที่ได้จาก Mobile Probe มีความเที่ยงตรงมากกว่า เพราะว่า Mobile Probe แสดงเวลาการเดินทาง ที่ผ่านตามถนนต่างๆ โดยตรงแต่ Fixed Sensor จะแสดงเวลาการเดินทางซึ่งประมาณมาจากความเร็วที่จับได้ขณะที่ผ่าน Sensor



รูปที่ 2.5 กระบวนการรับและส่งข้อมูลจราจรของโปรแกรม BMA Live Traffic

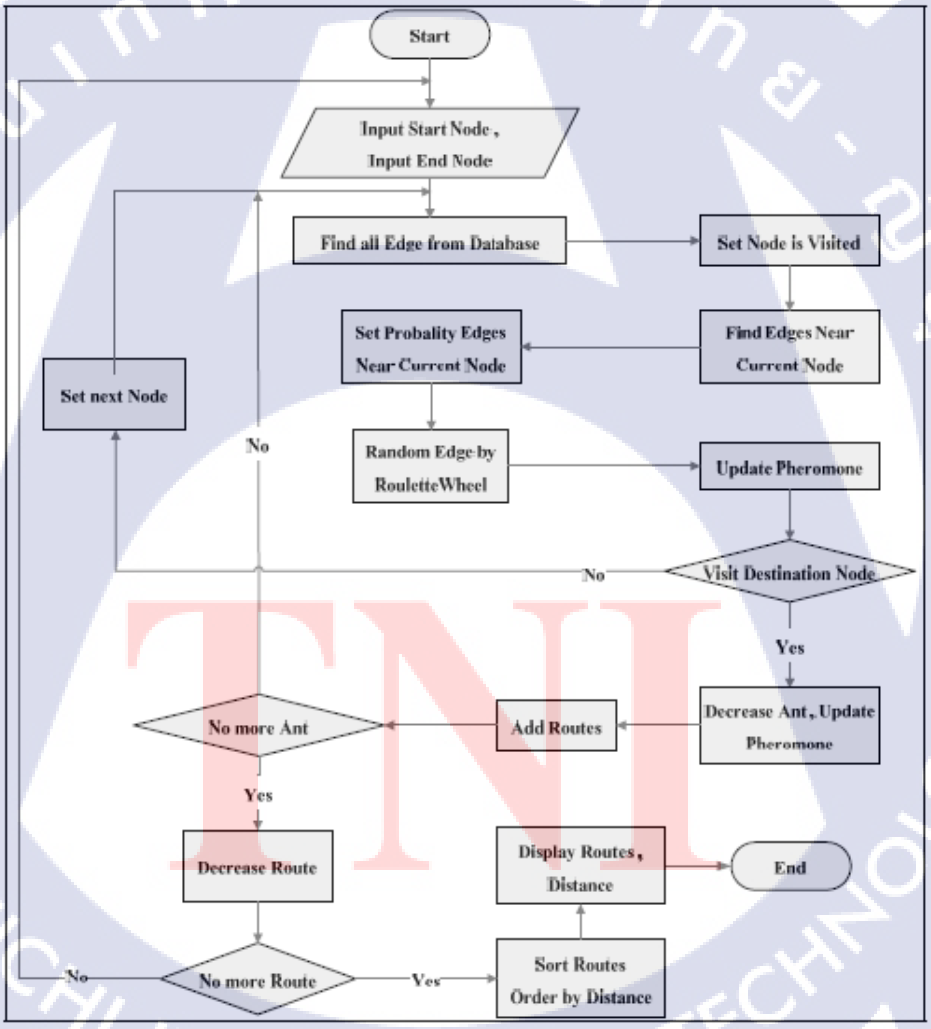
S. Panda et al. [5] นำเสนอการจัดการการจราจร โดยใช้ Swarm Intelligence และการเลือกเส้นทางผ่านโปรแกรมบนระบบปฏิบัติการ แอนดรอยด์ การบริหารการจัดการในเรื่องของสัญญาณไฟจราจร โดยการตรวจวัดความหนาแน่นหนาแน่นของการจราจรและจัดทำระบบโปรแกรมประยุกต์บน Smart Phone ระบบปฏิบัติการแอนดรอยด์และใช้เทคนิดในกลุ่มของ Swam Intelligence ระบบจะตรวจวัดความหนาแน่นการจราจรที่สัญญาณที่ตามแยกต่างๆ โดยใช้การประมวลผลภาพโดยใช้การตรวจจับและการเปลี่ยนแปลงความเพื่อหน่วงเวลาหรับสัญญาณไฟจราจรเพื่อลดความแออัดของแต่ละเส้นทาง คือหากมีปริมาณจำนวน ยานพาหนะมาจะเพิ่มเวลาของสัญญาณไฟเขียวและจะสื่อสารกับสัญญาณทางแยกที่อยู่ติดกัน เพื่อจัดการจราจรซึ่งขึ้นอยู่กับความหนาแน่น ซึ่งจะช่วยค้นหาเส้นทางการคับคั่งให้น้อยที่สุดสำหรับนักเดินทางโดยอิงตามจุดหมายปลายทาง ดังนั้นในทางเดียวกันสัญญาณทั้งหมดจะสื่อสารกับสัญญาณแต่ละจุด กลายระบบการจัดกลุ่ม นอกจากนี้ยังส่งข้อความไประบบการเชื่อมสัญญาณ Wi-Fi



รูปที่ 2.6 Block Diagram of the System at Unit Signal

2. งานวิจัยการค้นหาเส้นทาง

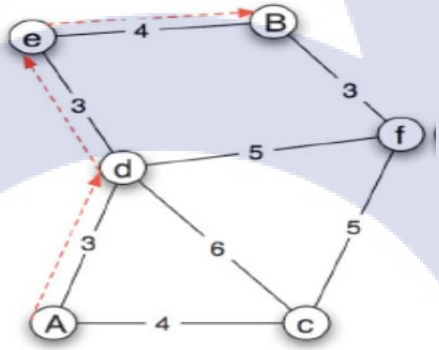
กิตติเดช วงศ์ศักดิ์ และเกียรติศักดิ์ โยชะนัง, “ระบบการค้นหาเส้นทางหลักที่สั้นที่สุดและเส้นทางรองโดยใช้ขั้นตอนวิธีแบบมด” [6] นำเสนอ การนำอัลกอริทึมแบบ Ant Colony มาประยุกต์เพื่อพัฒนา ระบบการค้นหาเส้นทางหลักที่สั้นที่สุดและเส้นทางสำรองที่เหมาะสมกับปัจจุบัน ซึ่งจะทำการหาค่าเฉลี่ยจากต้นทางไปยังปลายทางและนำมาแสดงลงบนแผนที่ใน Google Map และ Link ต่อกับ Web Site และป้ายสัญญาณจราจรอัจฉริยะ โดยหลักการพื้นฐานมาจากพฤติกรรมการหาอาหารของมดและนำมาเข้าสู่กระบวนการการกำหนดเส้นทาง จากกฎความน่าจะเป็นซึ่งจะมีกฎในการเลือกคือ กฎการคัดสรรโดย ใช้วงล้อเสี่ยงทาย (Roulette Wheel Selection Rule) และการหลักเลี่ยงการสะสมโดยใช้อัตราการระเหยของฟีโรโมนมาดำเนินการ เพื่อให้ได้เส้นทางอื่นเพิ่มเติม



รูปที่ 2.7 กระบวนการขั้นตอนวิธีแบบมด

ราชการ ปรึกษาดี และสุนันฑา สดสี [7] “การเปรียบเทียบหาเส้นทางที่เหมาะสม โดยวิธีระบบมดและ Dijkstra’s Algorithm” งานวิจัยนี้นำเสนอการเปรียบเทียบการค้นหาเส้นทางแบบมด กับ Dijkstra’s Algorithm โดยการจำลองสถานการต่างๆ และเปรียบเทียบผลลัพธ์ที่ได้ในแต่ละอัลกอริทึม ซึ่งจากผลลัพธ์ที่ได้จะพบว่า อัลกอริทึม แบบ Dijkstra’s Algorithm จะให้คำตอบเพียงคำตอบเดียวที่ให้ค่าที่ดีที่สุด ซึ่งต่างจากอัลกอริทึมวิธีระบบมด ที่จะให้ค่าคำตอบออกมาหลายค่าที่จะเป็นผลดีในกรณีที่เส้นทางหลักที่ดีที่สุดมีปัญหาจะหาเส้นทางสำรองได้ในทันทีโดยไม่ต้องคำนวณเส้นทางใหม่อีกครั้ง

T. Peng and X. Wang [8] งานวิจัยนี้นำเสนอการคำนวณเวลาที่สั้นที่สุด จากปัจจัยที่ส่งผลกระทบที่จะทำให้การเดินทางล่าช้าคือ จำนวนรถ จำนวนคนเดิน สภาพอากาศและพื้นผิวการจราจร โดยงานวิจัยนี้จะเป็นการวิเคราะห์เชิงพื้นที่โดยการทดลองจะเป็นการวัดจากวันธรรมดาที่เป็นช่วงเวลาทำงานในเมืองใหญ่เท่านั้น เนื่องจากเหตุผลทางด้านเวลาและขนาดของเมืองมีผลกับการจราจร การทดลองจะหาเส้นทางที่จะใช้และให้ค่าน้ำหนักแต่ละเส้น (Path Weight) จากปัจจัยทางด้านการจราจร



Factor / Criteria	Turning	Hospital	School	Resident area	Traffic light
Law	X	X	X	X	X
Non-rush hours	-	-	X	-	-
Driver choice	X	X	X	X	-
Holiday	-	-	X	-	-
Season	-	-	-	X	-
Weather	X	X	X	X	-

รูปที่ 2.8 น้ำหนักเส้นทางและปัจจัยผลกระทบต่างๆ ก่อนนำไปสร้างตัวคูณถ่วงน้ำหนัก

B. T. Morris and M. Manubhai [9] บทความนี่นำเสนอ การวิเคราะห์ พฤติกรรมของผู้ขับขี่ด้วยการตรวจสอบภาพจากกล้องวิดีโอ ใน 2 ส่วนที่สำคัญ ดังนี้ พฤติกรรมผู้ขับขี่จากเส้นทางตามกลุ่มการใช้เส้นทาง เป็นการศึกษาเชิงพฤติกรรมซึ่งมีผู้วิจัย ทำการศึกษาในหลาย รูปแบบที่แตกต่างกันแต่ในเป้าหมายและวัตถุประสงค์เดียวกันคือในเรื่องของเส้นทางตามรูปที่ 2.9 ซึ่งข้อดีในการที่จะสามารถเรียนรู้ได้ในเวลาสั้นๆ ของพฤติกรรมของยานพาหนะและเห็นจุดเด่นที่ชัดเจนทำให้นำไปพัฒนาสำหรับคาดการณ์พฤติกรรมใหม่ๆ ได้แต่ข้อเสียสำหรับวิธีนี้คือมีค่าใช้จ่ายสูงอันเนื่องจากจำนวนข้อมูลที่ศึกษาเป็นข้อมูลวิดีโอทำให้มีข้อมูลปริมาณมากที่ต้องจัดเก็บ และต้องใช้จำนวนของหน่วยความจำในการประมวลผลค่อนข้างสูง รวมถึงข้อจำกัดของข้อมูลที่มีสิ่งรบกวนเข้ามาในภาพ เช่น แสง เงา เป็นต้น

Research study	Classification/ inference	Description
Makris and Ellis[5]	Route envelope	Online adaptive route envelope model between important scene zones.
Piciarelli and Foresti[6]	Probability tree	A tree of clusters is built to share common areas for prediction and anomalous track identification.
Vasquez, Fraichard, and Laugier[7]	Growing HMM	Online adaptive modeling of the different ways to move from scene goals.
Hu et al.[8]	Gaussian chain	Two-level hierarchy to first spatially cluster resampled data and then cluster with zero-padded temporal trajectory with fuzzy c-means.
Morris and Trivedi[3]	HMM	Three-level learning hierarchy accounting for goals, spatial paths, and dynamic behaviors.
Noceti and Odone[9]	Common strings	Image space quantized with Voronoi tessellation to build an alphabet for representing a trajectory as a string.

รูปที่ 2.9 การศึกษาพฤติกรรมการใช้เส้นทาง

แบบจำลองพฤติกรรม เป็นการศึกษาจากการเรียนรู้พฤติกรรมจากการนำวิดีโอมาสร้างเป็นแบบจำลองโดยแบ่งส่วนของวิดีโอเป็นส่วนย่อย เพื่อศึกษาการเปลี่ยนแปลงและพฤติกรรมในรูปแบบต่างๆ และเรียนรู้ถึงพฤติกรรมที่เกิดขึ้นทั้งที่ค่อยๆ เป็นไปและการเปลี่ยนแปลงแบบชั่วขณะซึ่งมีการต่อยอดขยายเพิ่มเติมการเรียนรู้และปรับปรุงเพื่อการติดตามเฝ้าระวัง

Research study	Input data	Classification/inference	Description
Wang et al.[16]	Trajectory	Dynamic dual-HDP	Dynamic dual-HDP nonparametric Bayesian model to automatically model activity categories and semantic regions without specifying the number of topics and with online update of model.
Song et al.[17]	Optical flow	LDA	Two-level LDA topic model learned first for single-agent motion, which is input to second level LDA for multiagent interactions.
Fowlkes et al.[18]	Foreground pixels	DPMM	Fast rank-1 robust PCA used for foreground detection with counts of pixels in blocks used as input for DPMM learning, which enables incremental learning and inference.
Fu et al.[19]	Optical flow	LDA	Sparse topical coding used for efficient learning and representation of the topic model.
Hu et al.[20]	Trajectory	tDPMM	Dirichlet process mixture model is used for unsupervised clustering and modified to handle temporal structure and ordering inherent in a trajectory sequence.

รูปที่ 2.10 การศึกษาแบบจำลองพฤติกรรม

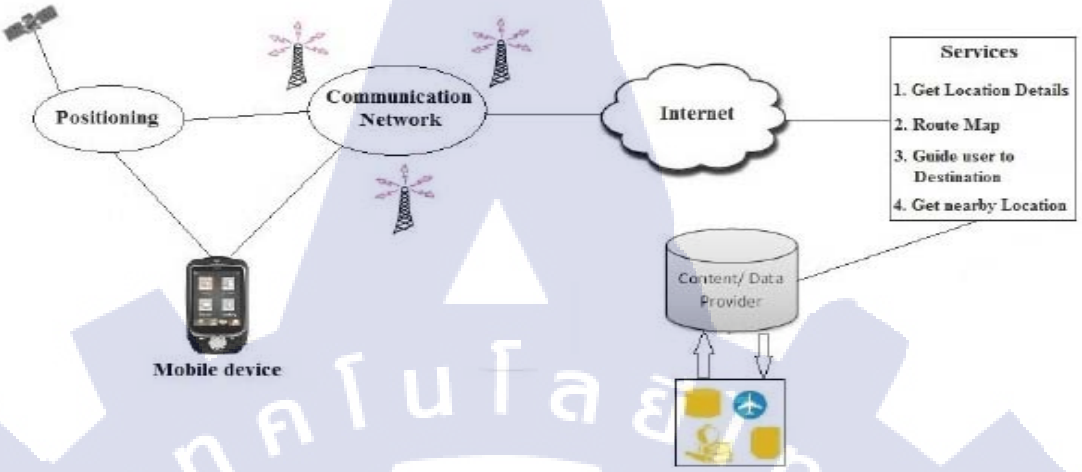
3. งานวิจัยที่เกี่ยวกับการจัดทำโปรแกรมและการใช้บริการของ Google Map

P. Doshi et al. “Location Based Services and Integration of Google Maps in Android” [10] งานวิจัยฉบับนี้เป็นการศึกษาและอธิบายการใช้งาน โดยการอ้างอิงถึงบริการของพิกัดแผนที่และบริการของ Google ในที่นี้ คือ Google Map และ Location Base Service (LBS) ในรายละเอียดจะระบุองค์ประกอบที่สำคัญของ LBS สำหรับการให้บริการนี้แก่ผู้ใช้บนแพลตฟอร์มแอนดรอยด์ นอกจากนี้ยังอธิบายการใช้งาน Google Map และ API ของ Google ในการรับข้อมูลตามตำแหน่งต่างๆ บน Android ซึ่งบริการ LBS ได้รับการสนับสนุนจาก องค์ประกอบโครงสร้างพื้นฐานบางอย่างที่ระบุไว้คือ

1. อุปกรณ์เคลื่อนที่ : การเข้าถึงบริการ LBS จะได้รับความช่วยเหลือจากอุปกรณ์เคลื่อนที่เพื่อส่งคำขอและเรียกดูผลการค้นหาสำหรับผู้ใช้ โทรศัพท์มือถือนี้สามารถอยู่ในรูปแบบของโทรศัพท์มือถือ แท็บเล็ต อุปกรณ์นำทางแบบพกพา (PNDs) เป็นต้น

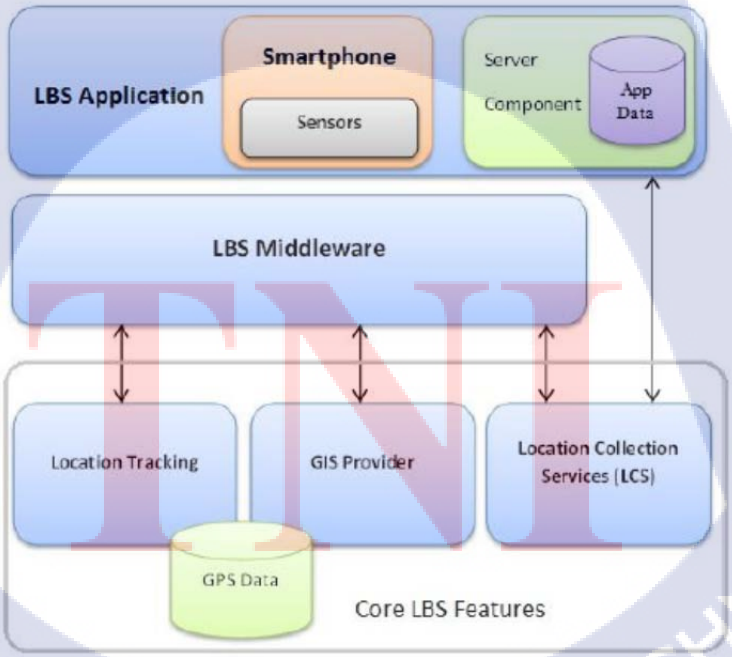
2. โปรแกรมประยุกต์: โปรแกรมประยุกต์ไม่มีอะไรเลยนอกจากซอฟต์แวร์ที่พัฒนาโดยนักพัฒนาโปรแกรมประยุกต์หรือผู้ให้บริการซึ่งดาวน์โหลดและติดตั้งลงในโทรศัพท์มือถือของผู้ใช้ซึ่งมีส่วนติดต่อเพื่อเข้าถึงบริการ LBS โดยจะกล่าวถึงองค์ประกอบหลักต่างๆ ด้วยกันคือ

2.1 LBS Components and Service Process



รูปที่ 2.11 Location Base Service (LBS) Process

สถาปัตยกรรมของ LBS จะประกอบด้วยส่วนต่างๆ ที่สำคัญดังนี้



รูปที่ 2.12 LBS Architecture

2.2 LBS Application ประกอบด้วยแอ็พพลิเคชัน การค้นหาบนแผนที่ ประกอบด้วยสมาร์ทโฟนที่มีจำนวนเซ็นเซอร์และส่วนประกอบ เซิร์ฟเวอร์ แอ็พพลิเคชัน และฐานข้อมูลเชิงพื้นที่ เป็นศูนย์กลางการประมวลผลสำหรับแพลตฟอร์ม LBS ที่จัดการฟังก์ชันอินเทอร์เฟซสำหรับผู้ใช้และสื่อสารกับฐานข้อมูลเชิงพื้นที่

2.3 LBS Middleware คือ โปรโตคอลและเทคโนโลยีเครือข่ายด้วยเทคโนโลยีไร้สายและอินเทอร์เน็ต มีการใช้งานภายในเครือข่ายของผู้ให้บริการระบบเครือข่าย

2.4 Core LBS Features ประกอบด้วย

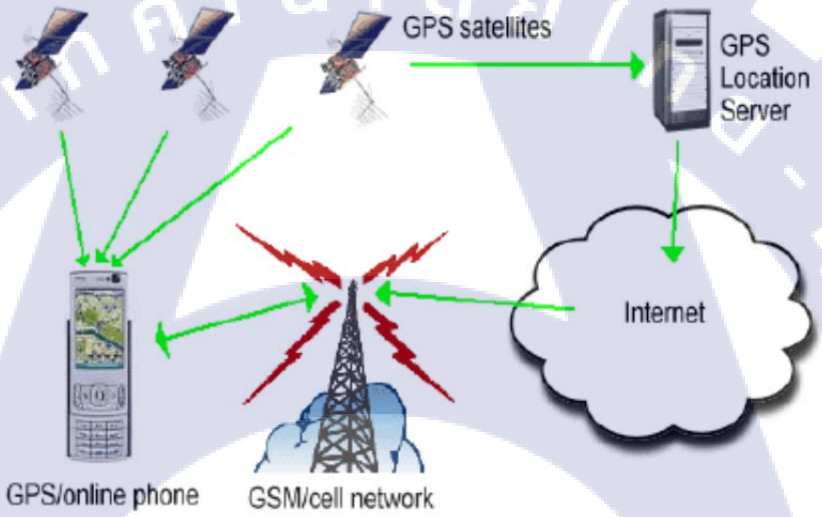
1. Location Tracking เป็นบริการสำหรับโทรศัพท์การระบุตำแหน่ง Global
2. Positioning System (GPS) ที่ผู้ใช้สามารถใช้เพื่อติดตามการเคลื่อนไหว
3. GIS Provider อำนวยความสะดวกที่เฉพาะเจาะจงทางภูมิศาสตร์สำหรับแอพพลิเคชัน LBS รวมทั้งข้อมูลแผนที่การแสดงแผนที่และบริการไดเรกทอรี
4. Location Collection Service องค์ประกอบนี้มุ่งเน้นไปที่พิกัดตำแหน่งของผู้ใช้ (ละติจูดและลองจิจูด)

2.5 Android Location API คือ การกำหนดตำแหน่งทางภูมิศาสตร์ในปัจจุบัน โดยมีแพ็คเกจการใช้งานตามตารางด้านล่าง

ตารางที่ 2.1 Classes in Android Location Package

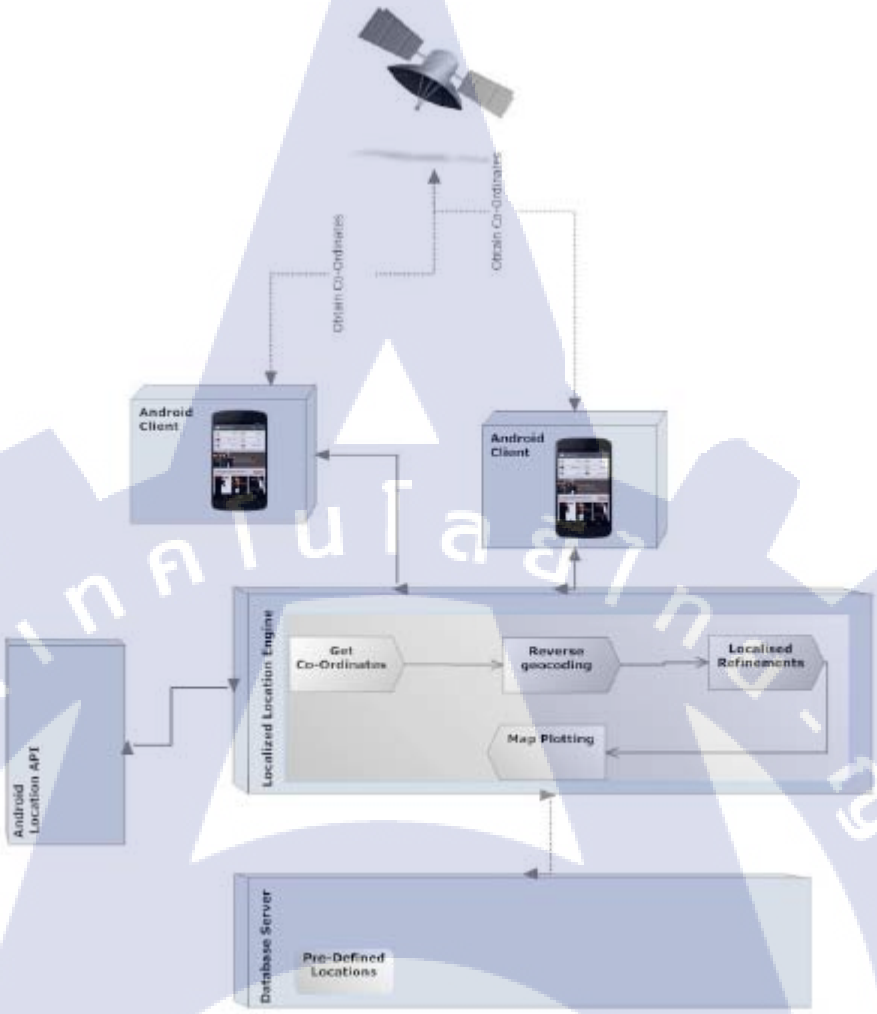
Address	It is a class representing an Address,which is a set of Strings describing a location.
Criteria	It is a class indicating the application criteria for selecting a location provider.
Geocoder	It is a class for handling geocoding and reverse geocoding.
GpsSatellite	This class represents the current state of a GPS satellite.
GpsStatus	This class represents the current state of the GPS engine.
Location	This is a data class representing a geographic location.
LocationManager	It is a class which provides access to the system location services.
LocationProvider	It is an abstract superclass for location providers.

M. Singhal and A. Shukla,“Implementation of Location Based Services in Android using GPS and Web Services” [11] นำเสนอ การใช้บริการตาม ตำแหน่งผ่าน Google Web Services และ API .และการเปลี่ยนเส้นทางจากการเดิน โดยผู้วิจัยแสดงให้เห็นถึงบริการของ API ที่รวมเข้ากับแผนที่เส้นทางที่จะทำการดึงข้อมูลเส้นทาง ผู้วิจัยทำการ ทดสอบระบบโดยพัฒนาโปรแกรมประยุกต์บนอุปกรณ์เคลื่อนที่บนระบบปฏิบัติการแอนด์ดรอยด์ที่ครอบคลุม APIs ที่ Google ให้บริการและอุปกรณ์เคลื่อนที่ที่ใช้งาน A-GPS เป็นเทคโนโลยีที่ผสานรวมเครือข่ายโทรศัพท์มือถือเข้ากับ GPS เพื่อให้มีความแม่นยำสูงเมตร การแก้ไขตำแหน่งภายในไม่กี่ วินาทีมีความครอบคลุมที่ดีขึ้นและในบางกรณีอาจใช้ภายในอาคารใช้พลังงานแบตเตอรี่น้อยลงและต้องใช้ดาวเทียมน้อยลง



รูปที่ 2.13 Location Base Service (LBS) Process

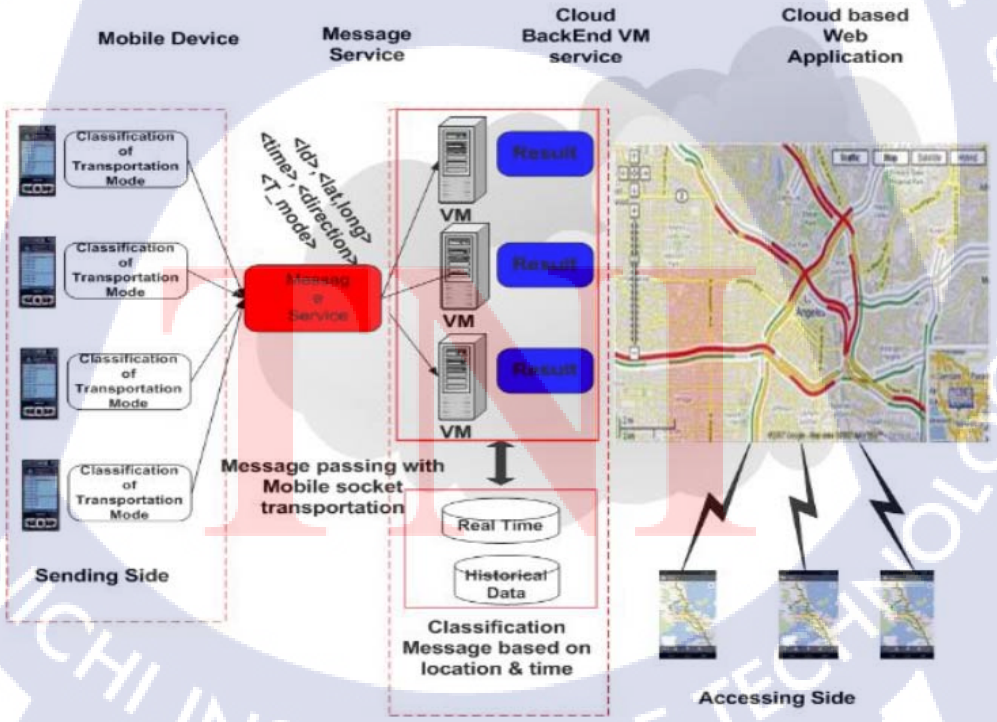
S. Bhatia and S. Hilal, “A New Approach for Location Based Tracking” [12] บทความนี้นำเสนอการเปรียบเทียบเทคนิคต่างๆ และการประยุกต์ใช้ การติดตามตำแหน่งพร้อมกับแนวทางใหม่ๆ ในการติดตามตำแหน่ง โครงร่างที่เสนอขึ้นบน อัลกอริทึมที่เรียกว่า"Localize Intelligence Algorithm" ซึ่งเป็นพื้นที่เฉพาะที่กำหนดขนาดเล็กและตั้งชื่อตามความแตกต่างของละติจูดและลองจิจูด ช่วยเพิ่มความแม่นยำในตำแหน่งของผู้ใช้และอัลกอริทึม โดย Localize Intelligence Algorithm เมื่อเกิดการทำงานจะได้รับสัญญาณและพิกัดจากดาวเทียม และออกแบบสถานที่ต้นทางและแจ้งสถานที่ตามพิกัดจากบริการ Google API



รูปที่ 2.14 UML Package Diagram for Location Intelligence Algorithm

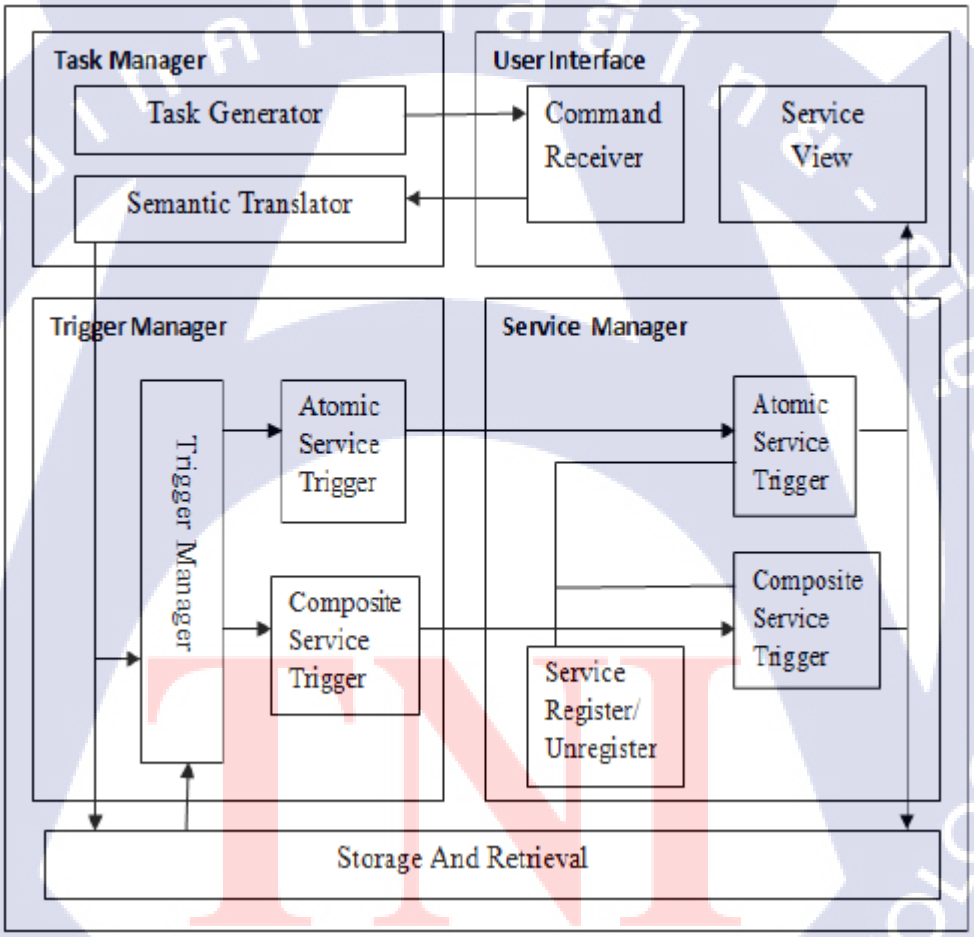
D. Suvarna et al. [13] บทความนี้นำเสนอการแก้ปัญหาการจราจรโดยใช้การเลือกเส้นทางที่ขึ้นอยู่กับพารามิเตอร์ต่างๆ เช่น อัตราการใช้ความเร็วของยานพาหนะ เส้นทางที่สั้นที่สุดและเวลาที่ใช้ การเลือกเส้นทางแบบพลวัติจะดำเนินการโดยใช้อัลกอริทึมของ Kruskal ตามพารามิเตอร์ที่ต่างกันของผู้ใช้ในการระบุตำแหน่งของผู้ใช้ไปยังปลายทางซึ่งจะช่วยในการค้นหายานพาหนะที่ตนอยู่หรือตำแหน่งของบุคคล โดยจะทำการรวมกลุ่มของผู้ใช้ในการให้เส้นทาง เช่น อยู่ในระแวกเดียวกันหรือเป็นทางเดียวกันระบบจะให้เส้นทางที่เหมือนกันแต่หากไม่ใช่และไม่มีผู้ใดเคยใช้เส้นทางระบบจะทำการคำนวณเส้นทางใหม่และจัดเก็บเพื่อเป็นข้อมูลในการให้เส้นทางกับผู้อื่นต่อๆ ไป กระบวนการตัดสินใจเส้นทาง แบ่งการทำงานออกเป็น 2 ส่วน คือ ขั้นตอนก่อนที่จะออกเดินทาง พิกัดจากบริการ “พบว่ามีค่าของความคลาด เคลื่อนจะคาดการณ์เส้นทางการขับรถและขณะเดินทางใช้ข้อมูลในเวลาจริงในการคำนวณเส้นทางตามฐานข้อมูลการจราจรที่มีขณะนั้น

S. S. Aung and T. T. Naing [14] นำเสนอการตรวจสอบสภาพการจราจรโดยให้ความสำคัญกับการวิเคราะห์พฤติกรรมของยานพาหนะจากโทรศัพท์เคลื่อนที่โดยใช้ข้อมูลจากจีพีเอสและข้อมูลทางสถิติที่ผ่านา ระบบถูกสร้างขึ้นเป็นสองส่วนคือ Client และ Cloud Server ด้านไคลเอ็นต์ระบบจะแยกแยะว่าผู้ให้บริการโทรศัพท์กำลังเดินทางโดยรถหรือเดิน เมื่อต้องการวิเคราะห์สถานการณ์นี้ใช้วิธีการกรองการย้ายเฉลี่ย ในฝั่งเซิร์ฟเวอร์ระบบจะตรวจจับสถานการณ์รับส่งข้อมูลโดยยึดตามการตรวจสอบพฤติกรรมของยานพาหนะ และติดตามผลของลูกค้าโดยใช้ Bayes Classifier โดยรวมของระบบ ระบบสามารถแยกออกเป็นสามส่วน ได้แก่ ฝั่งไคลเอ็นต์ Cloud Computation และ Cloud Storage การจัดประเภทโหมดการคมนาคมเสร็จสมบูรณ์บนโทรศัพท์มือถือ ในการจัดหมวดหมู่นี้การระบุรายละเอียดขอโหมดการเดินทาง เช่น การเดินหรือการขี่จักรยาน หรือ การขับขี่ หรือ รถบัสไม่จำเป็นต้องเป็นข้อมูลการจราจร ด้วยเหตุนี้จึงมีการระบุประเภทของโหมด (การเดินหรือยานพาหนะ) ไว้ในงานนี้ ผลของการกรองที่ได้รับจากอุปกรณ์เคลื่อนที่แต่ละเครื่องคำนวณจาก Cloud โดยการรวมข้อมูลประวัติของสถานที่สำคัญ ผลลัพธ์สุดท้ายจะถูกจัดเก็บในที่เก็บข้อมูล Cloud เป็นข้อมูลประวัติสำหรับการคาดการณ์ในอนาคต ระบบตรวจจับการเข้าชมมีวัตถุประสงค์เพื่อใช้กับ Google Cloud และสามารถเข้าถึงแอปพลิเคชันได้จากอุปกรณ์เคลื่อนที่ผ่านเครือข่าย



รูปที่ 2.15 ภาพรวมการทำงานของระบบ

P. Shah et al. “Location Based Reminder Using GPS For Mobile (Android)” [15] วัตถุประสงค์หลักของบริการตามตำแหน่งคือการให้บริการแก่ลูกค้าตามความรู้เกี่ยวกับสถานที่ นำเสนอการแจ้งเตือนตามสถานที่เกิดเหตุและอธิบายวิธีการที่ผลเหล่านี้มีอิทธิพลต่อการออกแบบอย่างต่อเนื่อง ระบบเตือนความจำที่อิงกับตำแหน่งที่ครอบคลุมมากขึ้น โดยใช้สถาปัตยกรรมของบริการระบุตำแหน่งที่ตั้งด้วย Global Positioning System (GPS) สำหรับการจัดการกลไกและการเรียกใช้บริการ ประกอบด้วยข้อมูลการจราจรแบบเรียลไทม์บริการแผนที่ดิจิทัลซึ่งจัดส่งไปยังอุปกรณ์เคลื่อนที่ตามสถานที่ของผู้ใช้เพื่อลดการรับส่งข้อมูลและให้บริการคำแนะนำแบบพลวัตตามสถานที่ของผู้ใช้และสภาพการจราจรในปัจจุบัน



รูปที่ 2.16 การทำงานของระบบ (System Architecture)

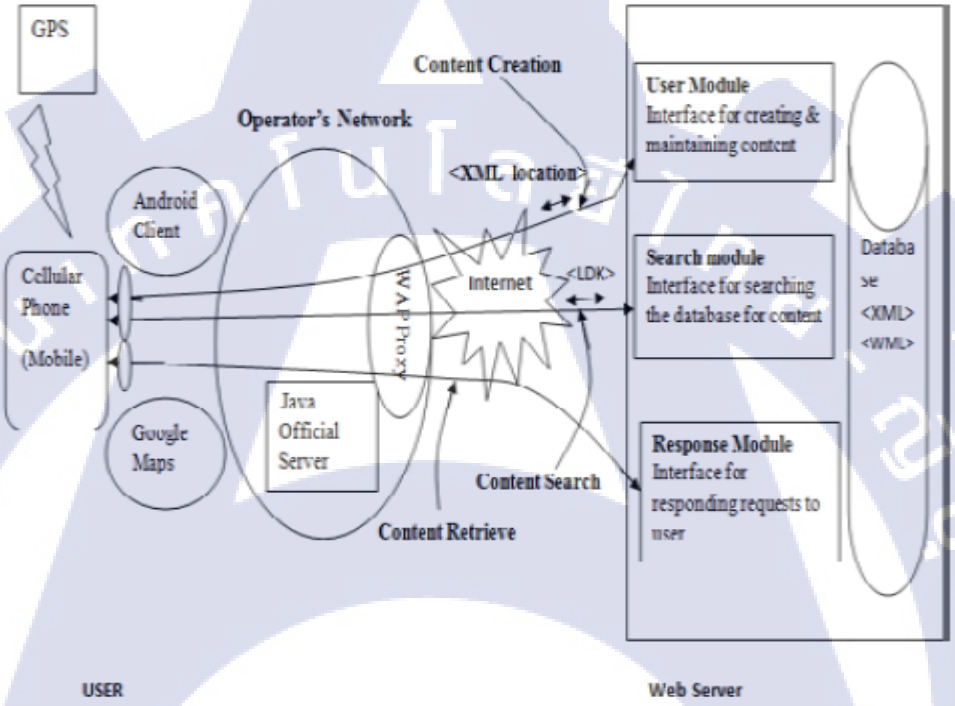
โดยนำหลักการและทำการออกแบบมาใช้ ดังต่อไปนี้:

(1) การสร้างและแก้ไขเนื้อหาเหตุการณ์

(2) ระบบทำการแจ้งข้อมูลโดยระบุตำแหน่ง

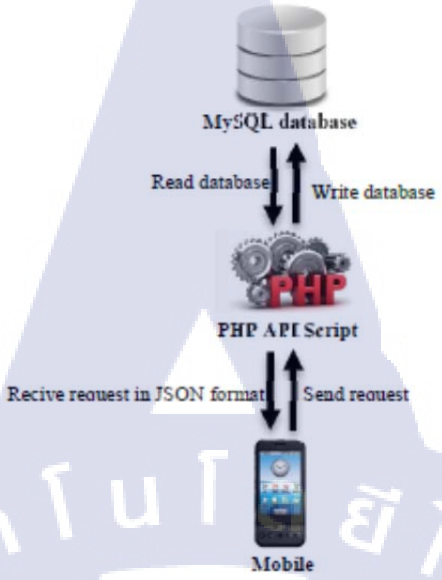
(3) ฟังก์ชันการค้นหาโดยใช้ข้อมูลตำแหน่งของผู้ใช้และ

(4) แสดงผลการค้นหาบนหน้าจอ



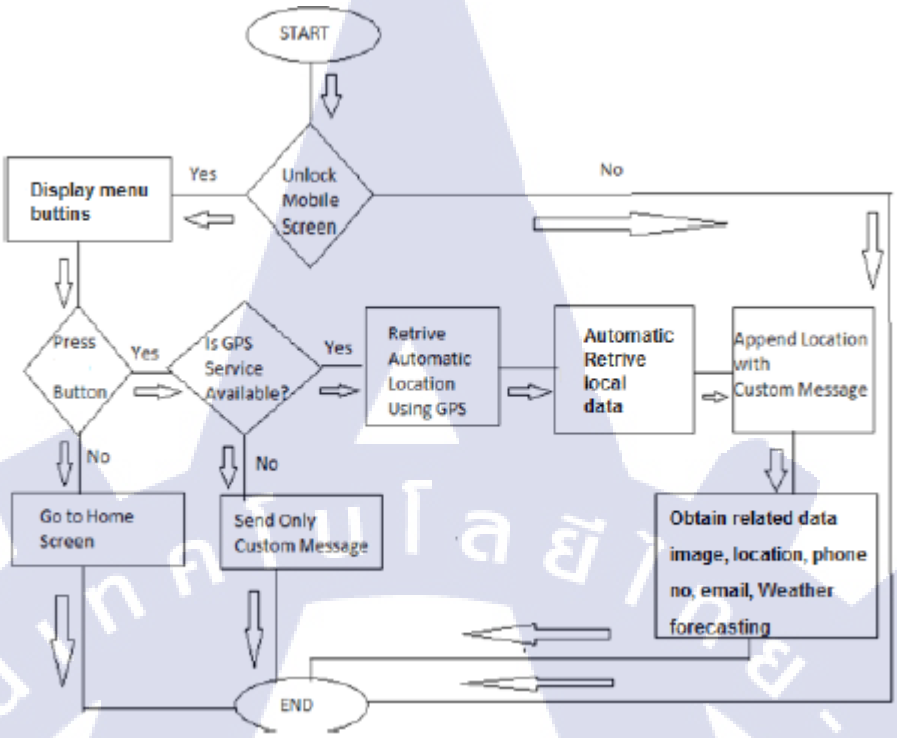
รูปที่ 2.17 Overview and Structure of the Proposed System

O. A. Ibrahim and K. J. Mohsen [16] บทความนำเสนอ แอปพลิเคชันสำหรับมือถือบนระบบปฏิบัติการแอนดรอยด์ โดยผู้ใช้สามารถเพิ่มลบและตรวจสอบตำแหน่งที่ตั้งเฉพาะบนแผนที่ออนไลน์ได้ และโปรแกรมจะแจ้งการนำทางพื้นฐานที่เป็นการแจ้งทิศทางพร้อมเส้นทางที่ดีที่สุดระหว่างต้นทางและจุดหมายปลายทางพร้อม คำนวณระยะทางและเวลาในการขับขี่ที่คาดหวัง ซึ่งระบบนี้จะใช้บริการของ Google Maps APIs, Google Direction APIs, JSON, PHP และ MySQL ในการดำเนินการเชื่อมต่ออินเทอร์เน็ตและฐานข้อมูลส่วนกลาง



รูปที่ 2.18 Structure of the Application

V. Thakare and S. Honale “Enhanced Functionality for Tracing Places, Weather Forecasting and Geo Fencing Using Android” [17] บทความนี้แสดงถึงความต้องการของผู้ใช้เกี่ยวกับการแจ้งเตือนการจราจร ในระบบที่เสนอนำสามารถใช้โปรแกรมประยุกต์บนมือถือโดยมีลักษณะพิเศษ เช่น การค้นหาตำแหน่งและพิกัดทางภูมิศาสตร์ การติดตามจะใช้จากตำแหน่งของยายพาหนะหรือบุคคลที่เคลื่อนที่โดยทั่วไป และใช้งาน Global Positioning System (GPS) เพื่อตรวจสอบและติดตามตำแหน่งที่แม่นยำของผู้ให้บริการในช่วงเวลาต่างๆ ตามรูปที่ 2.19



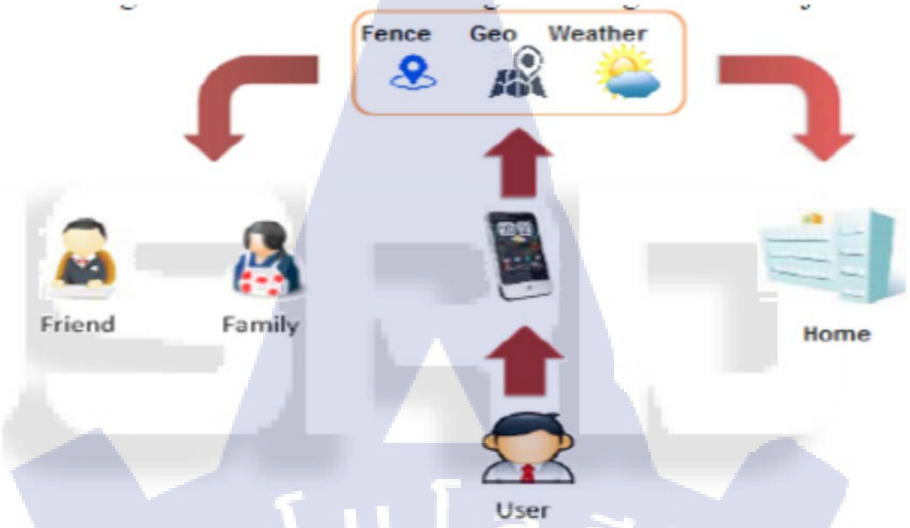
รูปที่ 2.19 Flow Chart Showing Working of this Project

เป็นการทำงานร่วมกันระหว่าง Google และ Google API โดย Google ให้ข้อมูลเกี่ยวกับตำแหน่งและทำให้ข้อมูลอยู่ในรูปแบบที่อ่านง่ายและมีประโยชน์ในชีวิตประจำวัน การติดตามตำแหน่งสถานที่แบบเรียลไทม์จากตำแหน่งทางภูมิศาสตร์ในปัจจุบันของผู้ใช้คุคือสถานที่ตั้งอุปกรณ์ของคุณอุปกรณ์เคลื่อนที่ Android จะเก็บข้อมูลทางภูมิศาสตร์ในรูปแบบต่างๆ (รูปที่ 2.20) ดังนี้

1) GPS (ออฟไลน์หรือฮาร์ดแวร์) ไม่ถูกต้อง

2) ผู้ให้บริการเครือข่าย (ให้บริการเครือข่ายมือถือ) มีความถูกต้องปานกลาง

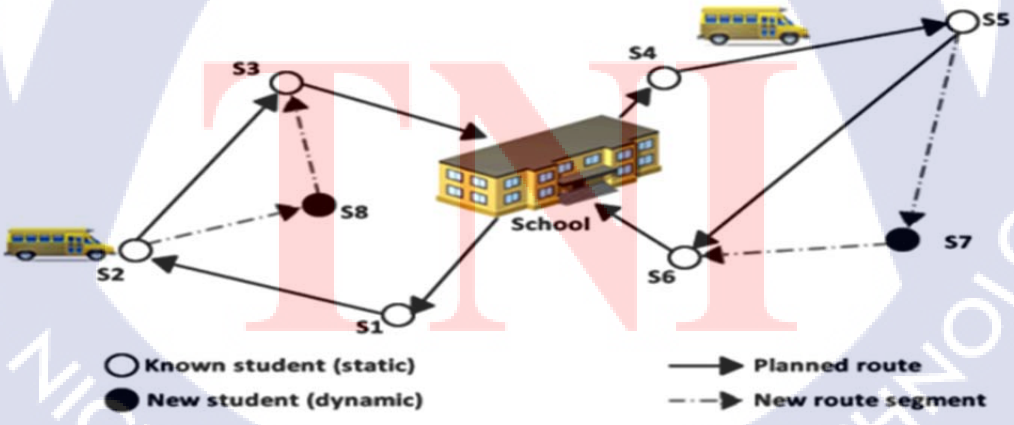
3) ผู้ให้บริการแบบ Passive (ให้บริการโดยอินเทอร์เน็ต) มีความถูกต้องมากขึ้น



รูปที่ 2.20 Flow Chart Showing Functionality of this Project

โดยการติดตามตำแหน่งในแบบเรียลไทม์จะให้ข้อมูลรายละเอียดเกี่ยวกับสถานที่ใดแห่งหนึ่งที่ช่วยในการติดตาม และการะบุตำแหน่งซึ่งมีในบริการของ Google API คือ Location Base System (LBS)

T. Yigit and O. Unsal [18] นำเสนอการปรับปรุงเส้นทางในปัจจุบันและการพัฒนาเส้นทางออนไลน์โดยใช้ GPSแบบ Real Time โดยใช้ค่าคงที่ของตำแหน่งของนักเรียนเพื่อช่วยลดการใช้เวลาในการค้นหาและแรงงานที่ใช้ โดยขั้นตอนและวิธีจะศึกษาจากข้อมูลดังนี้



รูปที่ 2.21 A Dynamic School Bus Routing Case

1. ปัญหาเรื่องเส้นทางการเดินรถโรงเรียน (School Bus Routing Problem) ซึ่งการออกแบบเส้นทางที่ไม่เหมาะสมจึงเกิดปัญหาในด้านการกระจายการขนส่งมีปัญหา เป้าหมายหลักของการแก้ไข คือการให้วิธีการกระจายเส้นทางภายใต้เส้นทางที่ยานพาหนะต้องหยุดที่บ้านของนักเรียนแต่ละรายไปยังปลายทางเพื่อหลีกเลี่ยงเวลาที่ใช้ในการขนส่ง การรวบรวมและการกำหนดเส้นทางรถโรงเรียน จึงมีข้อจำกัด ในการสร้างอัลกอริทึมที่ใช้เพื่อสร้างโซลูชันให้เหมาะสม การกำหนดเส้นทางที่สั้นที่สุดสำหรับรถโรงเรียนเพื่อรับนักเรียนทั้งหมดจากเพื่อจะพาพวกเขาไปโรงเรียน และออกจากโรงเรียนกลับบ้าน

2. การศึกษาขั้นตอนการเลือกเส้นทางแบบมด (Ant Colony Optimization) ซึ่งเป็นการลอกเลียนแบบพฤติกรรมการหาอาหารของมดที่พวกมันมีความสามารถของการหาเส้นทางที่สั้นที่สุดจากแหล่งอาหาร

3. การศึกษาขั้นตอนของปัญหาเส้นทางของรถส่งนักเรียนร่วมกับการวิธีการแก้ไขปัญหาแบบอาณาจักรมด (ACO Algorithm for Dynamic School Bus Routing Problem) และสร้างอัลกอริทึม

Algorithm 1. Ant colony.

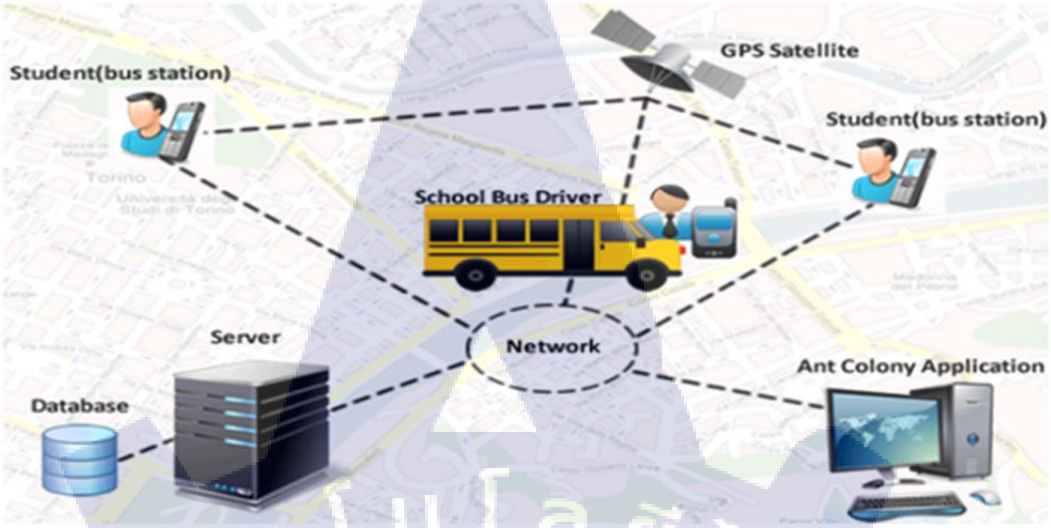
Begin Define number of iteration, school buses (ants).

1. Define the parameters α, β, ρ. Define a random starting city for each school bus.

2. For each iteration. For each school bus.

3. Define the next student stop by using Equation 5.

4. Define the iteration length. Update pheromone by using Equation 3 and 4. End.



รูปที่ 2.22 Block Diagram of Developed Application

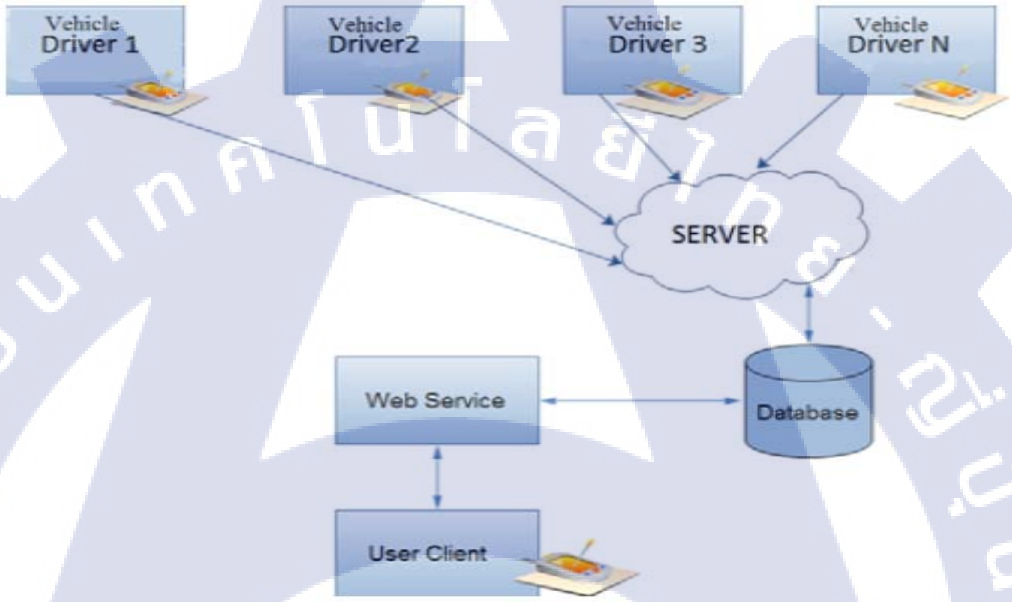
R. Shinde et al. [19] ออกแบบแอปพลิเคชัน Android ที่มีชื่อว่า CLICKn SAVE เพื่อใช้ในการช่วยเหลือผู้ประสบอุบัติหตุที่จะใช้ภาพของฉากอุบัติเหตุโดยเฉพาะ จากนั้นก็จะแนะนำโรงพยาบาลใกล้เคียงหรือสถานีตำรวจให้โดย โปรแกรมนี้จะแสดงโรงพยาบาลที่ใกล้ที่สุด 5 รายการและสถานีตำรวจที่มีระยะทางกับสถานที่จากสถานที่เกิดอุบัติเหตุ จากนั้นผู้ใช้ต้องเลือกสิ่งที่ดีที่สุดเลือกโรงพยาบาลและรอการตอบกลับตามเวลาที่กำหนดหากไม่ตอบกลับจะย้ายไปยังโรงพยาบาลใกล้เคียงและส่งข้อความไปที่สถานีตำรวจซึ่งจะมีขั้นตอน ดังนี้

ขั้นตอนการตรวจสอบและดำเนินการนำส่งผู้บาดเจ็บ

1. จับรูปภาพโดยอุปกรณ์ในระบบปฏิบัติการ Android
2. หาตำแหน่งโดยใช้อัลกอริทึม
3. ใช้ตำแหน่งเมืองจาก Google API
4. เรียงลำดับจากน้อยไปหามากโดยใช้อัลกอริทึม

A. Renugambal and V. A. Kameswari [20] นำเสนอแนวทางในการค้นหาเส้นทางที่ดีที่สุดบน GPS มีการพัฒนาระบบ GPS-equipped เพื่อใช้แท็กซี่เป็นอุปกรณ์ในการนำส่งตำแหน่งและส่งกลับมายังศูนย์กลางในการให้บริการ การปรับปรุงและการค้นหาเส้นทางที่สามารถใช้ความเร็วเส้นทางที่มีการใช้น้อยที่สุดตามที่ผู้ใช้ข้อความค้นหา เส้นทางที่เร็วที่สุดและสั้นที่สุดสำหรับผู้ใช้ปลายทาง ผ่าน Google, Yahoo, Bing แผนที่เราสามารถวิเคราะห์วันที่ที่เราสามารถทำได้ หาเส้นทางที่สั้นที่สุด และอิงกับระบบคลาวด์สำหรับการคำนวณทางปฏิบัติที่รวดเร็ว สำหรับผู้ใช้เฉพาะ

ใช้รถแท็กซี่ที่มี GPS และผู้ใช้โทรศัพท์ที่รองรับ GPS เพื่อลดปัญหาการเข้าใช้แอปพลิเคชั่นประเภทต่างๆ GPS, Wi-Fi เป็นต้น ทำให้สามารถหลีกเลี่ยงปัญหาการจราจรและได้เส้นทางที่เร็วที่สุด ขั้นตอนการกำหนดเส้นทาง ใช้สำหรับเส้นทางที่ระบุสำหรับกราฟหลักจากบางส่วนโดยใช้อัลกอริทึมการกำหนดเส้นทางตามเวลาและเส้นทางการกลั่นจะใช้ในการหาเส้นทางที่เร็วที่สุดสำหรับเชื่อมต่อสถานที่สำคัญต่อเนื่องใช้ข้อ จำกัด ความเร็วและการเขียนโปรแกรมแบบพลวัตในการค้นหาขนาดเล็กแลทำการคำนวณไปพร้อมๆ กัน หาข้อมูลที่กระจัดกระจายเพื่อระบุทางเลือกที่พบความแออัด



รูปที่ 2.23 System Architecture

<u>รายละเอียดของ Framework :</u>

1. งานนี้มุ่งเน้นที่การระบุเส้นทางที่ดีและที่สั้นที่สุดจากข้อมูลตำแหน่งที่เก็บรวบรวมโดยผู้ใช้จำนวนมาก
2. ข้อมูลจะถูกรวบรวมจากผู้ใช้รายอื่นและเก็บไว้ในฐานข้อมูล
3. การประมวลผลข้อมูลที่ได้และใช้เทคนิคการทำเหมืองข้อมูลค้นหาแผนที่เส้นทางที่ดีที่สุด (ข้อมูล)
4. เส้นทางที่สั้นที่สุดสามารถระบุโดยใช้วิธีการต่างๆ
5. ค้นหาเส้นทางของผู้ใช้แต่ละรายและทุกรายโดยใช้ข้อมูลตำแหน่ง
6. ค้นหาเส้นทางทั่วไปของผู้ใช้ทั้งหมดตามเวลา
7. ดึงเวลาการเดินทางจากจุด/เส้นทางไปยังจุด/เส้นทางอื่น

8. ดึงข้อมูลเส้นทางและเวลาจากการประมวลผลด้านบนและอัพเดตด้วย DB
9. ใช้สภาพอากาศสำหรับเส้นทางคำนวณ

ตารางที่ 2.2 สรุปการดำเนินการของงานวิจัยที่ศึกษา

งานวิจัยเรื่อง	**วิธีการดำเนินการ**
Google Map [2]	- หาระยะทางที่สั้นที่สุดเทียบจากเวลาเป็นเกณฑ์
การค้นหาเส้นทางหลักที่สั้นที่สุดและเส้นทางรอง [6]	- ขั้นตอนวิธีแบบมด (Ant Colony) - กฎการคัดสรรโดย ใช้วงล้อเสี่ยงทาย (Roulette Wheel Selection Rule)
การเปรียบเทียบหาเส้นทางที่เหมาะสม โดยวิธีระบบมดและ Dijkstra's Algorithm [7]	- Dijkstra's Algorithm - ขั้นตอนวิธีแบบมด (Ant Colony)
การคำนวณเวลาที่สั้นที่สุด [8]	- ปัจจัยที่ส่งผลกระทบที่จะทำให้การเดินทางล่าช้าคือ จำนวนรถ จำนวนคนเดิน สภาพอากาศและพื้นผิวการจราจร โดยงานวิจัยนี้จะเป็นการวิเคราะห์เชิงพื้นที่โดยการทดลองจะเป็นการวัดจากวันธรรมดาที่เป็นช่วงเวลาทำงาน - หาเส้นทางที่จะใช้และให้ค่าน้ำหนักแต่ละเส้น (Path Weight) จากปัจจัยทางด้านการจราจร
การวิเคราะห์ พฤติกรรมของผู้ขับขี่ด้วยการตรวจสอบภาพจากกล้องวิดีโอ [9]	- เพื่อศึกษาการเปลี่ยนแปลงและพฤติกรรมในรูปแบบต่างๆ และเรียนรู้ถึงพฤติกรรมที่เกิดขึ้นทั้งที่ค่อยๆ เป็นไปและการเปลี่ยนแปลงแบบชั่วขณะ
การศึกษาและอธิบายการใช้งาน Google [10]	- เป็นการศึกษาการอ้างอิงถึงบริการของ Google Map และของพิกัดแผนที่ คือ Google Map, Google API และ Location Base Service (LBS) [10]

ตารางที่ 2.2 สรุปการดำเนินการของงานวิจัยที่ศึกษา (ต่อ)

งานวิจัยเรื่อง	วิธีการดำเนินการ
การใช้บริการตามตำแหน่งผ่าน Google Web Services และ API และการเปลี่ยนเส้นทางจากการเดิน [11]	- ศึกษาการทำงานของ APIs ที่ Google ให้บริการและอุปกรณ์เคลื่อนที่ที่ใช้งาน A-GPS เป็นเทคโนโลยีที่ผสานรวมเครือข่ายโทรศัพท์มือถือเข้ากับ GPS
A New Approach for Location based Tracking [12]	- บทความนี้นำเสนอการเปรียบเทียบเทคนิคต่างๆ และการประยุกต์ใช้ การติดตามตำแหน่งพิกัด - Localize Intelligence Algorithm - ได้รับสัญญาณและพิกัดจากดาวเทียม และออกแบบสถานที่ต้นทางและแจ้งสถานที่ตามพิกัดจากบริการ Google API
การแก้ปัญหาการจราจรโดยใช้การเลือกเส้นทางที่ขึ้นอยู่กับพารามิเตอร์ [13]	- อัลกอริทึมของ Kruskal ตามพารามิเตอร์ที่ต่างกัน - การทำงานออกเป็น 2 ส่วนคือ ขั้นตอนก่อนที่จะออกเดินทางที่จะคาดการณ์เส้นทางการขับรถ และขณะเดินทางใช้ข้อมูลในเวลาจริงในการคำนวณเส้นทางตามฐานข้อมูลการจราจรที่มีขณะนั้น
Location Based Reminder Using GPS For Mobile (Android) [15]	- วัตถุประสงค์หลักของบริการตามตำแหน่งคือการให้บริการแก่ลูกค้าตามความรู้เกี่ยวกับสถานที่ - นำเสนอการแจ้งเตือนตามสถานที่เกิดเหตุและอธิบายวิธีการ - ใช้สถาปัตยกรรมของบริการระบุตำแหน่งที่ตั้งด้วย Global Positioning System (GPS)
O. A. Ibrahim and K. J. Mohsen [16] - นำเสนอแอปพลิเคชันสำหรับมือถือบนระบบปฏิบัติการแอนดรอยด์	- โปรแกรมจะแจ้งการนำทางพื้นฐานที่เป็นการแจ้งทิศทางพร้อมเส้นทางที่ดีที่สุดระหว่างต้นทางและจุดหมายปลายทาง พร้อมคำนวณระยะทางและเวลาในการขับขี่ที่คาดหวัง

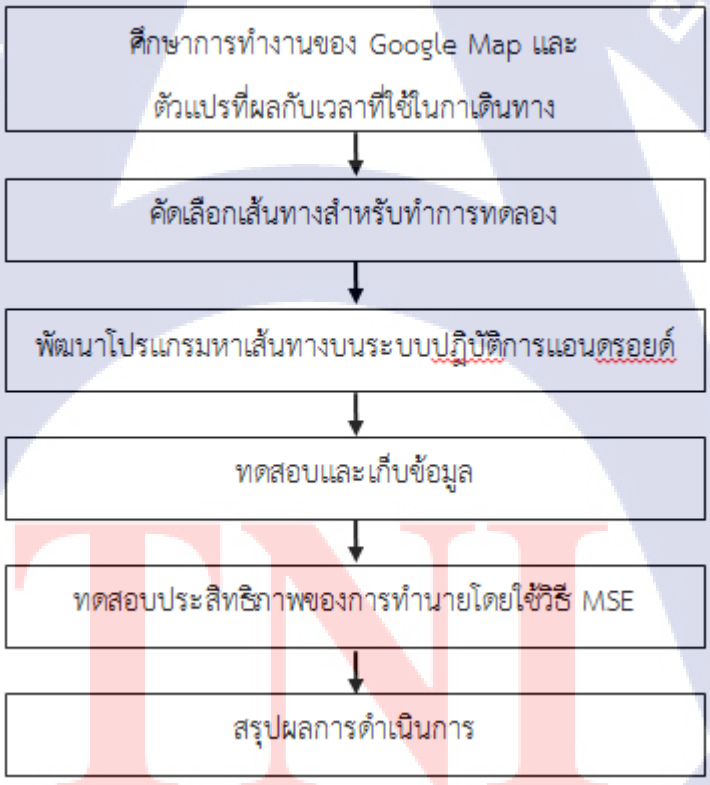
ตารางที่ 2.2 สรุปการดำเนินการของงานวิจัยที่ศึกษา (ต่อ)

งานวิจัยเรื่อง	วิธีการดำเนินการ
การปรับปรุงเส้นทาง การทำนายสภาพภูมิอากาศและพิกัดที่ใช้ผ่านอุปกรณ์ในระบบปฏิบัติการแอนดรอยด์ [17]	- การแจ้งเตือนการจราจร ใช้โปรแกรมบนมือถือโดยมีลักษณะพิเศษ เช่น การค้นหาตำแหน่งและพิกัดทางภูมิศาสตร์ การติดตามจะใช้จากตำแหน่งของยานพาหนะหรือบุคคลที่เคลื่อนที่โดยทั่วไป และใช้งาน Global Positioning System (GPS) เพื่อตรวจสอบและติดตามตำแหน่งที่แม่นยำของผู้ให้บริการในช่วงเวลาต่างๆ
พัฒนาเส้นทางออนไลน์โดยใช้ GPS แบบ Real Time [18]	-การเลือกเส้นทางแบบมด (Ant Colony Optimization)
พัฒนาระบบ CLICKnSAVE [19]	-หาระยะทางที่สั้นที่สุดเทียบจากระยะทางเป็นเกณฑ์ โดยการเรียงลำดับเส้นทางจากน้อยไปหามาก
พัฒนาระบบ GPS-equipped [20]	-หาระยะทางที่สั้นที่สุดเทียบจากเวลาเป็นเกณฑ์ โดยมุ่งเน้นการใช้บริการของ LBS เป็นหลัก

จากการศึกษาบทความวิจัยที่ผ่านมาตามตารางข้างต้นจะพบว่าในงานวิจัยโดยส่วนใหญ่จะพยายามหาเส้นทางที่สั้นที่สุดโดยจะใช้เวลาหรือระยะทางเป็นเกณฑ์ ซึ่งการใช้พารามิเตอร์ที่แตกต่างกันออกไป แต่สุดท้ายก็จะมีการอ้างอิงเวลาและเส้นทางที่ Google Map แจ้งไว้ทั้งหมด ในงานวิจัยนี้จะเป็นการนำเสนอการค้นหาเส้นทางที่เป็นการทำนายเวลาที่จะต้องใช้ในแต่ละเส้นทาง โดยนำที่ผ่านมาเพื่อปรับปรุงเส้นทางที่จะเดินทางให้ได้เวลาที่สั้นที่สุดและแม่นยำมากขึ้น

บทที่ 3
วิธีดำเนินการศึกษา

งานวิจัยนี้เป็นการศึกษาและหาวิธีการคำนวณให้ได้มาซึ่งเส้นทางที่ใช้เวลาที่น้อยที่สุดสำหรับการเดินทาง โดยการศึกษาจากเส้นทางการเดินรถจากจุดหนึ่งไปยังอีกจุดหนึ่ง ซึ่งมีเส้นทางเชื่อมต่อและสภาวะของการจราจรในเวลาที่แตกต่างกัน ในการค้นหาเส้นทางการประมวลผลข้อมูลจะทำการค้นหาเส้นทางจากจุดเริ่มต้นไปยังจุดหมายปลายทางโดยจะนำข้อมูลจากแผนที่ใน Google Map และข้อมูลการเดินทางในอดีต นำมาคำนวณเพื่อหาเส้นทางที่ใช้เวลาน้อยที่สุด โดยในส่วนของ Google Map , Google API และ Location Base Services (LBS) [2, 3] จะแสดงแผนที่และพิกัดแสดงเส้นทาง ซึ่งขั้นตอนและกระบวนการทำการวิจัยจะมีขั้นตอนการดำเนินการตามขั้นตอนต่อไปนี้

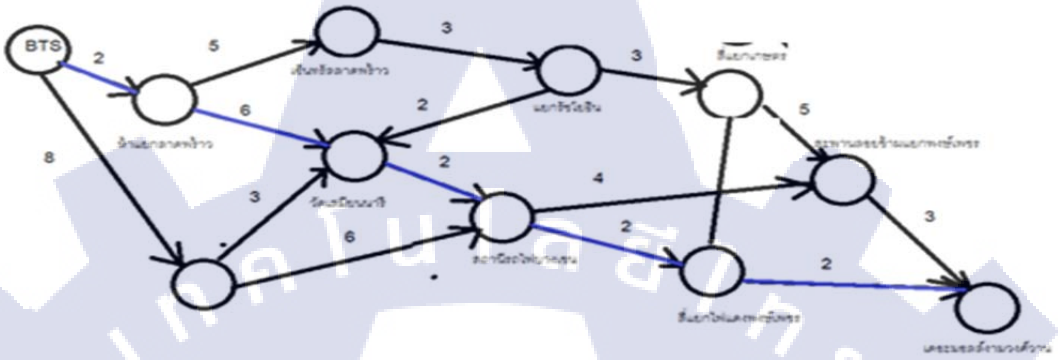


รูปที่ 3.1 ภาพรวมขั้นตอนการดำเนินการทำงาน

ในภาพรวมของการทำงานจะเป็นการนำเสนอในส่วนของการค้นหาเส้นทางเท่านั้นจะไม่มีในส่วนของการบำรุงรักษาระบบโดยเฉพาะในเรื่องของฐานข้อมูลเส้นทางต่างๆ จะเป็นการใช้บริการจากทางบริการที่ Google มีเท่านั้น

3.1 ศึกษาการทำงานและการเก็บข้อมูลที่ประกอบด้วย

1. ศึกษากระบวนการหาเส้นทางที่สั้นที่สุด ในแต่ละจุดใน Google Map จะมีการคำนวณเส้นทางที่มีทั้งระยะทางและเวลาที่จะต้องใช้โดยการประมาณการ โดยในรายงานฉบับนี้จะใช้ระยะทางที่ได้นำมาหาเส้นทางที่ใช้เวลาน้อยที่สุด [6] ,[7]



รูปที่ 3.2 ตัวอย่างการประมวลผลเส้นทางที่ใช้สั้นที่สุด

2. ศึกษารูปแบบการแสดงผลข้อมูลดัชนีรถติดที่แสดงสถานะการจราจร และการทำงานของ Google Map API กับ Global Positioning System (GPS) และ Location Base System (LBS) [8]

2.1 การทำงานของ Google Map และ Google API การใช้บริการแผนที่และฟังก์ชั่นการทำงานเพื่ออ้างอิงและการขอข้อมูลเส้นทางทั้งหมดที่ระบุต้นทางและปลายทาง

2.2 การทำงานของ GPS ในการใช้บอกพิกัดของอุปกรณ์หรือการขอเส้นทาง

2.3 การทำงานของ LBS เป็นการทำงานร่วมกันในหลายๆ ส่วนเพื่อรองรับการรองรับการเรียกและค้นหาพิกัดสถานที่ตั้งตามที่ผู้ใช้เรียก ซึ่งจะมีการประมวลผลในด้านฐานข้อมูลที่ประกอบไปด้วย Application และ Server การประมวลผลข้อมูล ด้านเครือข่ายที่จะต้องมีระบบการเชื่อมต่อการตั้งข้อกำหนดของการใช้งาน รวมทั้งตัวอุปกรณ์ที่ใช้ ที่ทั้งหมดต้องมีการประสานกันเพื่อรองรับการทำงานทั้งหมด

3. คัดเลือกช่วงถนน และเก็บข้อมูลถนนที่คัดเลือก

การคัดเลือกช่วงถนนที่จะใช้มีผลกับการวิจัยมากเพราะการคัดเลือกหากเลือกช่วงถนนที่จำนวนรถยนต์ไม่มากเพียงพอก็จะทำให้ผลการวิจัยมีผลลัพธ์ซึ่งไม่แตกต่างกับการใช้งานได้ปกติ ดังนั้นในการทดลองนี้จะเลือกช่วงถนนเพื่อเก็บข้อมูลอยู่ในช่วงของถนนงามวงศ์วานเป็นจุดเริ่มต้นและจุดสิ้นสุดที่ถึงรถไฟฟ้าใต้ดินสถานีพหลโยธินในช่วงวันทำงาน โดยจะแบ่งเวลาออกเป็นช่วงเช้า

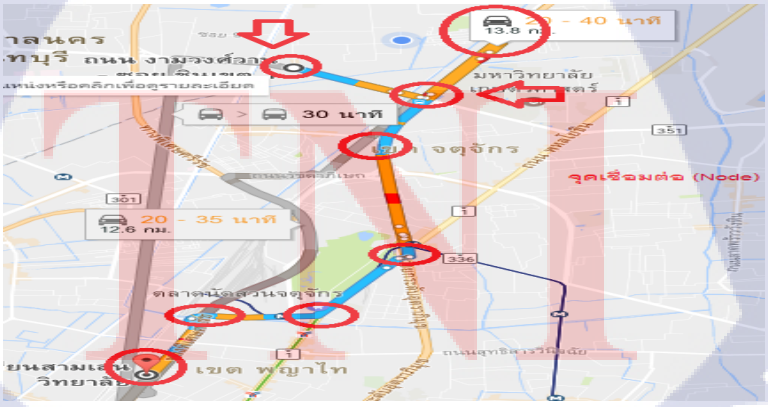
และเย็นของวันจันทร์ วันศุกร์และวันเสาร์ ในระยะเวลา 1 เดือน และในวันเวลาทำงานโดยปกติและวันสุดสัปดาห์ซึ่งจะแบ่งเป็นช่วงเวลาดังนี้

ตารางที่ 3.1 การเก็บข้อมูลในช่วงวัน เวลา ต่างๆ

วันที่ทำการทดสอบ	เช้า	เย็น
วันจันทร์	07.00-08.30	17.30-19.00
วันศุกร์	07.00-08.30	17.30-19.00
วันเสาร์	07.00-08.30	17.30-19.00

โดยการที่เลือกเส้นทางนี้เนื่องจากเป็นเส้นทางที่ผ่านในหลายจุดและมีทางแยกซึ่งเป็นเส้นทางที่สามารถไปได้หลายเส้นทาง และไม่มีจุดในการสร้างทางรถไฟฟ้าในปัจจุบัน เพื่อลดความคลาดเคลื่อนที่อาจเกิดการชะลอตัวนาน และในเส้นทางที่ใช้ก็มีข้อมูลที่มีปัจจัยต่างๆ ที่มีผลต่อการจราจรปกติ (เนื่องจากในปัจจุบันทางรัฐบาลและองค์การขนส่งมีการสร้างเส้นทางรถไฟฟ้าอยู่หลายเส้นทางซึ่งล้วนเป็นจุดที่มีความติดขัดผิดปกติจึงไม่นำมาใช้ในการทดสอบนี้)

จากเส้นทางที่จะทำการสำรวจจะพบปัจจัยที่ทำให้ช่องผิวจราจรมีปัจจัยที่อาจทำให้เส้นทางเกิดการชะลอตัวที่จะทำให้เกิดความหนาแน่นและติดขัดและแต่ละเส้นทางก็มีการเชื่อมต่อของเส้นทางในหลายส่วน โดยเส้นทางที่จุดเชื่อมต่อหรือโนด (Node) อาจจะเป็นสัญญาณไฟจราจรหรือจุดกลับรถ เป็นต้น ตามรูปที่ 3.3 ซึ่งก็จะมีเวลาที่ใช้ในแต่ละโนดที่เชื่อมต่อกัน



รูปที่ 3.3 เส้นทางที่มีจุดการเชื่อมต่อในแต่ละเส้นทาง (Node)

เมื่อ T = ระยะเวลาทั้งหมดที่ใช้ในการเดินจากจุดเริ่มต้นถึงปลายทาง

D = ระยะทั้งหมดวัดจากจุดเริ่มต้นถึงปลายทาง

S = ความเร็วของยานพาหนะ

$$T = D/S \tag{3.1}$$

จากรูปที่ 3.3 จะพบว่าแต่ละเส้นทางจะมีจุดและระยะทางที่เชื่อมโยงไปยังจุดถัดไปจนถึงปลายทางที่จะมีเวลาที่เดินทางจากจุดที่ 1 ไปจุดที่ 2

เมื่อ t_i คือ เวลาแต่ละจุด ,

d_i คือ ระยะทางแต่ละโนดที่เชื่อมต่อกัน ,

s_i คือ ความเร็วที่ใช้ในแต่ละโนดที่เชื่อมต่อกัน,

และ i คือ จุดเชื่อมต่อจากของโนด

$$t_i = \frac{d_i}{s_i} \tag{3.2}$$

ดังนั้น เวลาทั้งจึงเป็นผลรวมของเวลาที่ใช้ในการเดินทางแต่ละจุดและระยะทางที่เชื่อมโยง

$$T = \frac{d_1}{s_1} + \frac{d_2}{s_2} + \cdots + \frac{d_{n-1}}{s_{n-1}} + \frac{d_n}{s_n} \tag{3.3}$$

หรือ

$$T = \sum_1^n t_i \tag{3.4}$$

จากสมการที่ (3.1)-(3.4) เมื่อต้องการทำการเดินทางและตรวจสอบการเดินทางผ่านตัวโปรแกรม Google Map จะได้ค่าของ t_i ที่ใช้ d_i และ s_i ณ เวลาที่ผู้ใช้เรียกดูข้อมูลที่เวลาเดียวกันทุกจุด เช่น เส้นทางจาก ม.ธุรกิจบัณฑิตย์ ไปห้างเดอะมอลล์งามวงศ์วานเวลา 8.00 น. และมีโนดเส้นทางทั้งหมด 5 จุด ข้อมูลของ t_i ก็จะเป็นข้อมูลที่เวลา 8.00 น. ทุกจุด ข้อมูลจึงยังคลาดเคลื่อนเพราะผู้เดินทางไม่ได้อยู่ในแต่ละโนด d_i ที่เวลาเดียวกัน แต่ในงานวิจัยนี้จะทำการคำนวณหา ค่า t_i โดยใช้ d_i และ s_i ที่แตกต่างกัน ตามตัวอย่างในตารางที่ 3.3 , 3.4 และตารางที่ 3.5 เส้นทางจาก ม.ธุรกิจบัณฑิตย์ (DPU) ไปห้างเดอะมอลล์งามวงศ์วานเวลา 8.00 น.

ตารางที่ 3.2 รายละเอียดเส้นทางจาก ม.ธุรกิจบัณฑิตย์ (DPU) ไปห้างเดอะมอลล์งามวงศ์วาน

เส้นทาง	รายละเอียด
จุดที 1	DPU ถึง ซอย ประชาชื่น 12 แยก 1
จุดที 2	ซอย ประชาชื่น 12 แยก 1 ถึง ถนน ประชาชื่น
จุดที 3	ถนน ประชาชื่น ถึง ถนน งามวงศ์วาน
จุดที 4	ถนน งามวงศ์วาน เข้าสู่ ซอย งามวงศ์วาน 18/ซอย จุฬ้าเกษม
จุดที 5	เข้าสู่ ซอย งามวงศ์วาน 18/ซอย จุฬ้าเกษม ถึง เดอะมอลล์ งามวงศ์วาน

ณ จุดเริ่มต้น Google Map จะพิจารณาสภาพจราจรทุกโนดที่เวลาเดียวกันเป็นจุดเริ่มต้น คือ 8.00 ประเมิน เวลาตามตารางที่ 3.3 แต่งานวิจัยที่นำเสนอแจ้งเวลาที่ใช้ในแต่ละจุดที่เวลาแตกต่างกันในเวลาที่ถึงในแต่ละจุด ตามตารางที่ 3.4 และ เวลาที่เดินทางจริงใช้เวลา 30 นาที เมื่อเทียบกันเวลาที่งานวิจัยนำเสนอสามารถให้เวลาที่ดีกว่าในการคาดคะเนการเดินทางได้ตามตารางที่ 3.5

ตารางที่ 3.3 ข้อมูลเวลาที่ Google Map แจ้ง

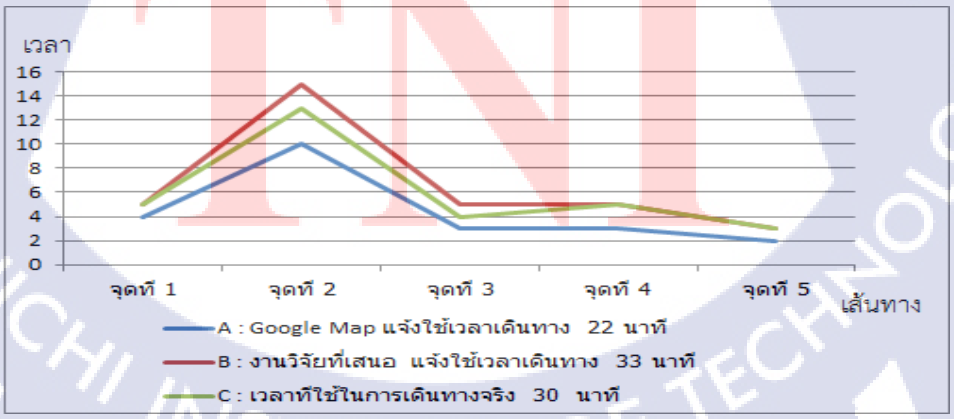
ที่เวลา	8.00 น.	8.00 น.	8.00 น.	8.00 น.	8.00 น.
เส้นทาง	จุดที 1	จุดที 2	จุดที 3	จุดที 4	จุดที 5
Google Map แจ้งใช้เวลาเดินทาง 22 นาที	4	10	3	3	2

ตารางที่ 3.4 ข้อมูลเวลาทีงานวิจัยที่นำเสนอแจ้ง

ที่เวลา	8.00 น.	8.05 น.	8.20 น.	8.25 น.	8.30 น.
เส้นทาง	จุดที 1	จุดที 2	จุดที 3	จุดที 4	จุดที 5
งานวิจัยที่เสนอ แจ้งใช้เวลาเดินทาง 33 นาที	5	15	5	5	3

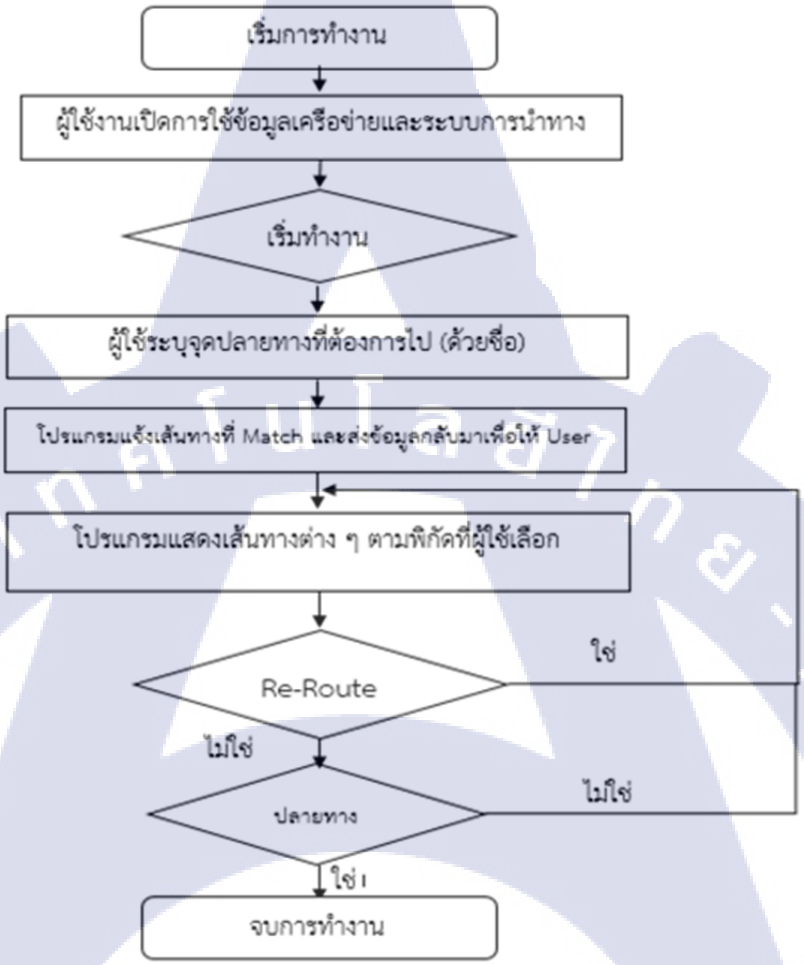
ตารางที่ 3.5 เปรียบเทียบเวลาที่เดินทางจริง เทียบกับ Google Map และ งานวิจัยที่นำเสนอ

เส้นทาง	จุดที 1	จุดที 2	จุดที 3	จุดที 4	จุดที 5
A : Google Map แจ้งใช้เวลาเดินทาง 22 นาที	4	10	3	3	2
B : งานวิจัยที่เสนอ แจ้งใช้เวลาเดินทาง 33 นาที	5	15	5	5	3
C : เวลาที่ใช้ในการเดินทางจริง 30 นาที	5	13	4	5	3



รูปที่ 3.4 เปรียบเทียบเวลาที่เดินทางจริงใช้เวลาทั้งหมดกับเวลาของ Google และงานที่นำเสนอ

3.2 การประมวลผลข้อมูล



รูปที่ 3.5 ขั้นตอนการประมวลผล

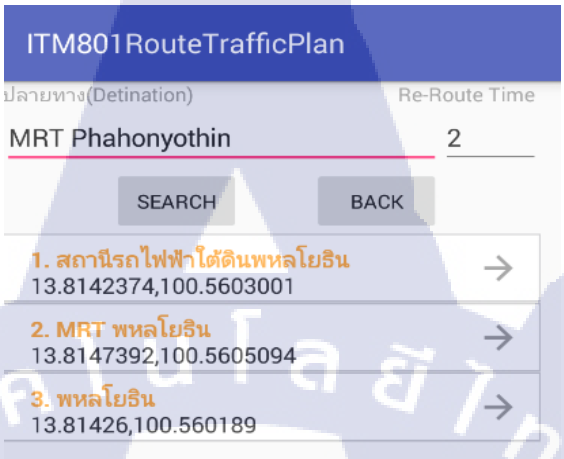
ขั้นตอนการประมวลผลการทำงาน จะประกอบไปด้วย

1. ผู้ใช้งานเปิดการใช้ข้อมูลเครือข่าย ระบบการนำทาง และโปรแกรมที่พัฒนาขึ้น และเริ่มทำงาน

2. ผู้ใช้ระบุจุดปลายทางที่ต้องการไป (ด้วยชื่อ) เพื่อให้โปรแกรมตรวจสอบและดึงข้อมูลของชื่อสถานที่ที่ต้องการไป Match กับข้อมูลของ Google Map โดยจะเรียกใช้คำสั่ง GeoCode ดังนี้

```
new Geocoder(getApplicationContext(),new Locale("TH","TH"));
geoCoder.getFromLocationName(DestinationEditText.getText().toString(),LIST);
```

3. โปรแกรมทำการดึงข้อมูลจะต้องกำหนดข้อมูลว่าเป็นของประเทศใด และให้แสดงใน ListView สำหรับให้ผู้ใช้เลือกสถานที่อีกครั้ง โดยผลที่ได้จะได้ข้อมูลดังนี้



รูปที่ 3.6 รายการเส้นทางที่เป็นไปได้ตามชื่อที่ผู้ใช้ค้นหา

และโปรแกรมจะทำการสร้างเส้นทางโดยแยกออกเป็นสี ดังนี้

```
String color = colors[i % colors.length];
ArrayList<LatLng>directionPositionList=leg.getDirectionPoint();
polylines[j]=  mMapView.addPolyline(DirectionConverter.createPolyline(this,
directionPositionList, 10, Color.parseColor(color)));
```

รูปที่ 3.7 เส้นทางหลังผู้ใช้เลือกสถานที่ต้องการไป

4. โปรแกรมจะทำการตรวจสอบและทำการค้นหาเส้นทางใหม่อีกครั้งและแสดงเส้นทางต่างๆ แยกตามสีต่างๆ ที่เป็นเส้นทางที่สามารถเดินทางได้ ในกรณีดังนี้

4.1 โปรแกรมตรวจสอบพบว่ามีพิกัดเปลี่ยนไป โดยคำสั่งที่ตรวจสอบในการหาเส้นทางใหม่ เมื่อ

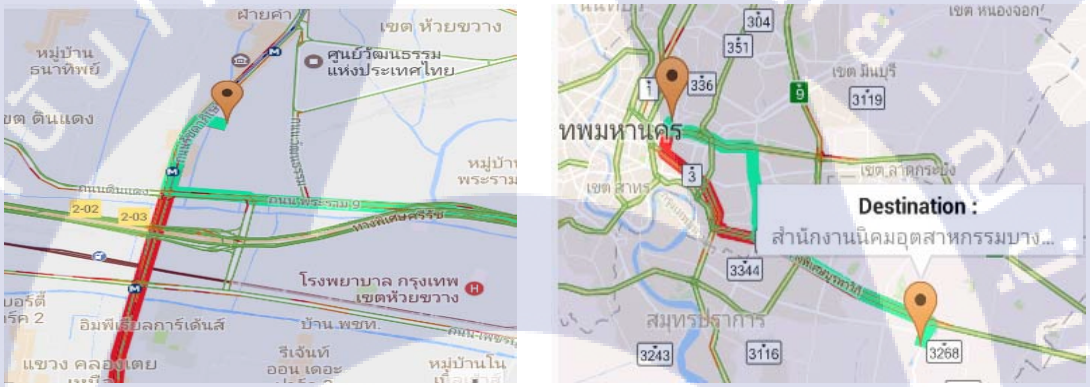
4.1.1 Location คือ ที่ตั้งที่เปลี่ยน

onLocationUpdated(Location location)

4.1.2 ครบระยะเวลาที่ต้องการให้โปรแกรมทำการ Re-Route เส้นทางที่ผู้ใช้กำหนดตั้งแต่เริ่มต้น โดยคำสั่งระยะเวลาที่ต้องการให้คำนวณเส้นทางใหม่

เมื่อ UPDATE_INTERVAL คือ เวลาที่ต้องการให้คำนวณเส้นทางใหม่

GoogleMap.setInterval(UPDATE_INTERVAL);



(a) พิกัดมีการเปลี่ยนแปลง (b) ครบเวลาที่ตั้ง (Interval)

รูปที่ 3.8 การปรับปรุงเส้นทางเมื่อพิกัดมีการเปลี่ยนแปลงหรือครบเวลาที่ตั้ง

5. โปรแกรมตรวจสอบกรณีถึงปลายทางจะแจ้งให้ทราบว่าถึงพิกัดที่กำหนด และสิ้นสุดการทำงาน

3.3 การสร้างโปรแกรมค้นหาเส้นทาง

งานวิจัยนี้เป็นการพัฒนาระบบขึ้นเพื่อใช้ตรวจสอบและหาเส้นทางการเดินทางที่ใช้เวลาน้อยที่สุด การสำรวจเส้นทางการใช้งานจึงเป็นการเก็บข้อมูลเพื่อใช้ในการเปรียบเทียบเส้นทางการทดสอบ เมื่อระบบทำการดึงจากฐานข้อมูลของทาง Google เท่านั้นไม่มีการเก็บข้อมูล ซึ่งการทดสอบหากไม่พบเส้นทางใน Google แสดงออกมาเป็นเส้นทางในแผนที่ การทดสอบในเส้นทางที่ใช้จริงจะ

ทำการทดสอบการค้นหาเส้นทางใหม่ จากระยะเวลาหรือจากเส้นทางที่เปลี่ยนไปเมื่อไม่ได้อยู่ในเส้นทางที่กำหนด โดยโปรแกรมจะมีการดำเนินการดังนี้

1. ศึกษาความต้องการและความเป็นไปได้ของโปรแกรมที่ต้องการพัฒนาเส้นทาง โดยสำหรับ ความต้องการของโปรแกรมนี้ก็คือโปรแกรมการค้นหาเส้นทางที่สั้นที่สุดและใช้เวลาที่น้อยที่สุดเพื่อใช้ในการช่วยตัดสินใจในการเดินทาง ซึ่งจะต้องแสดงเส้นทางที่เป็นไปได้ที่มีอยู่โดยรอบเส้นทาง และรายละเอียดการเดินทาง รวมถึงการประมาณระยะเวลาที่ใช้ในเส้นทางนั้น โดยปัญหาจากบทที่ 1 และ ทฤษฎีและงานวิจัยที่ศึกษาจากบทที่ 2

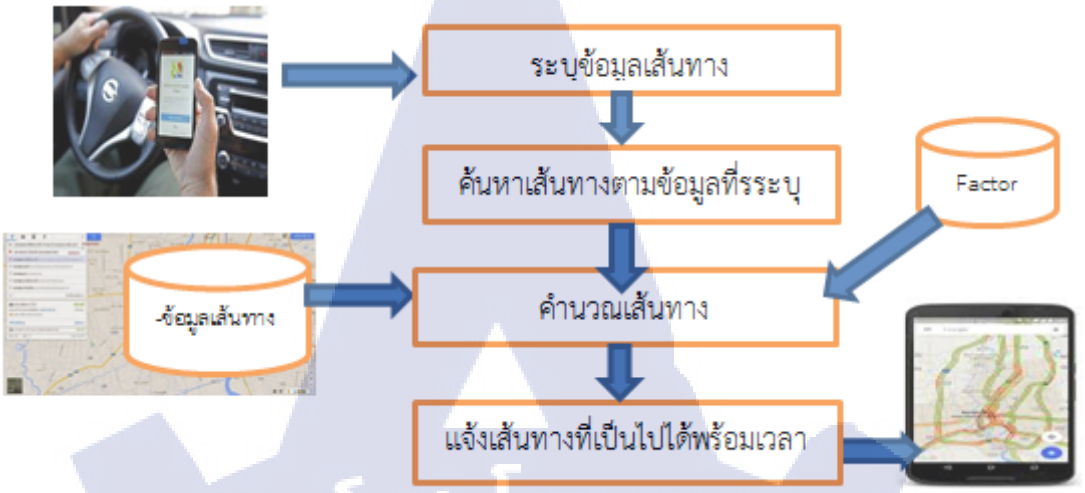
2. โปรแกรมที่ใช้ในการพัฒนา ในงานวิจัยนี้เลือกโปรแกรมในระบบปฏิบัติการระบบ Android มาใช้ในการพัฒนา เนื่องจากเป็นโปรแกรมที่เป็นระบบเปิดสามารถทำพัฒนาและการทดลองได้ง่าย รวมถึงไลบรารี่ที่มีในตัวของระบบปฏิบัติการของแอนด์ดรอยด์เอง และการนำไลบรารีการทำงานอื่นๆ ที่ไม่ได้มีตั้งต้นไว้แต่แรกทำการเพิ่มเข้ามาเพื่อง่ายต่อการทำงานในกรณีที่อาจต้องพัฒนาคำสั่งที่มีความสลับซับซ้อนมาก โดยนักพัฒนาด้วยกันเองจะมีการแบ่งปันเพื่อให้นักพัฒนาอื่นๆ ได้นำมาประยุกต์ให้เข้ากับงานของตนเอง

3. โปรแกรมที่ได้จะทำการประมวลผลจะแบ่งข้อมูลเป็น 2 ส่วนคือ ผลลัพธ์ในส่วนของการเส้นทางการเดินทางและเวลาตามเส้นทางจากจุดเริ่มต้นปกติ ส่วนที่ 2 ทดสอบการ Reroute .ในช่วงเวลาที่ทางผู้ใช้งานกำหนดและการเปลี่ยนเส้นทางที่มีการออกนอกพิกัดเส้นทาง เพื่อทดสอบการทำงานในสถานะเส้นทางมีการเปลี่ยนไป

4. อุปกรณ์ที่ใช้ในการทดสอบและลงโปรแกรมที่จัดทำขึ้นเป็นหลัก คือ Smart Phone ยี่ห้อ Lava Grand Pro 5.5 โดยจะมีหน้าจอการทำงานหลักตามรูปที่ 3.4

5. การทำงานเมื่อต้องการการใช้งานผู้ใช้ต้องเชื่อมต่อกับผู้ให้บริการเครือข่ายและเปิดระบบการนำทาง GPS เพื่อเชื่อมต่อสัญญาณ โดยในโปรแกรมที่จัดทำขึ้นสามารถเชื่อมต่อได้ทั้งบริการของ GSM และบริการ WI-FI แต่ความละเอียดของพิกัดจะไม่ได้ดีเทียบเท่า เพราะระบบ GSM เป็นระบบสัญญาณที่เชื่อมต่อกับดาวเทียมโดยตรงกับอุปกรณ์

ลำดับขั้นตอนการทำงานจะมีภาพรวมตามรูปที่ 3.8 และในหน้าจอหลักการทำงานตามรูปที่ 3.9

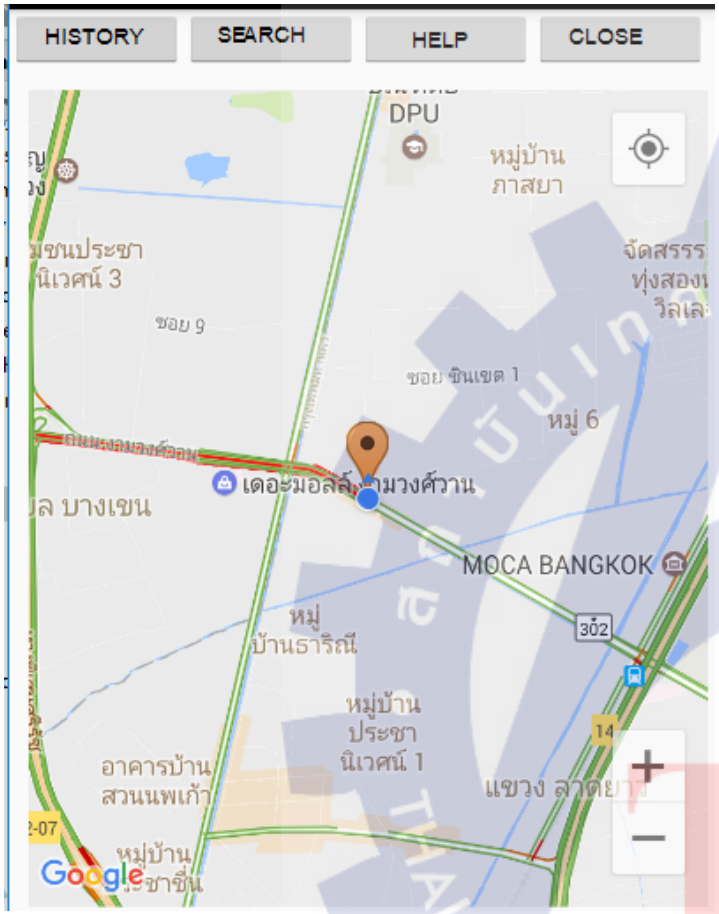


รูปที่ 3.9 การทำงานของโปรแกรมการค้นหาเส้นทาง

ขั้นตอนการทำงาน

1. ผู้ใช้จะทำการป้อนข้อมูลปลายทางที่จะทำการเดินทาง
2. โปรแกรมจะทำการดึงข้อมูลเส้นทางที่ต้องการไปตามชื่อที่สถานที่ผู้ใช้ระบุที่เป็นไปได้ที่ในฐานข้อมูลทั้งหมดและนำมาแสดงให้ผู้ใช้เลือกก่อนที่จะแสดงเส้นทางทั้งหมดที่เป็นไปได้ตามพิกัดที่ได้เลือกมา
3. แจ้งผลเส้นทางที่ค้นหาไปยังหน้าจอมือถือ/แท็บเล็ต

รูปที่ 3.10 หน้าจอการแสดงผล

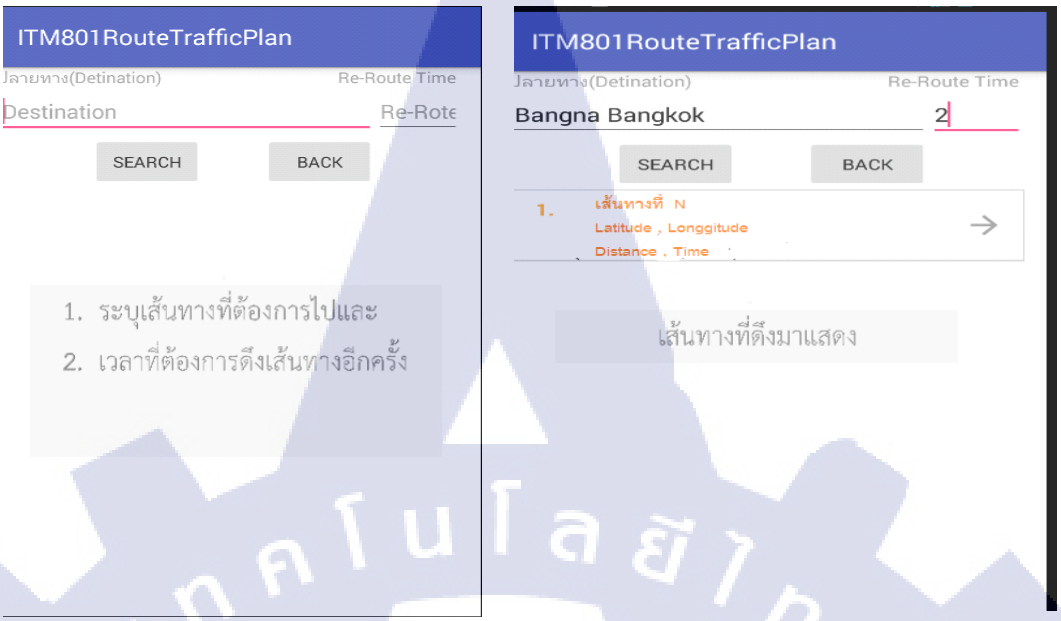


รูปที่ 3.11 หน้าจอการแสดงผลหน้าแรก

หน้าจอหลักการทำงาน จะประกอบไปด้วย

1. History เป็นหน้าจอการเก็บบันทึกเส้นทางที่ User เคยเดินทาง
2. Search เป็นหน้าจอการค้นหาเส้นทางที่ User ต้องทำการระบุ
3. Help เป็นหน้าจอแนะนำระบบการทำงาน
4. Close ออกจากระบบเมื่อทำการเลือกที่ปุ่มหรือเมื่อกดย้อนกลับ (Back Press )

หน้าจอการแสดงแผนที่โดยเริ่มต้นระบบจะแจ้งสถานที่ตั้งของอุกรณ์ และเมื่อเลือกเส้นทางจากหน้าจอการค้นหาจะแสดงเส้นทางทั้งหมดที่เป็นไปได้ในการเดินทาง



(a) ผู้ใช้ระบุชื่อเส้นทางที่จะไป (b) รายการเส้นทางหลังกดปุ่มค้นหา

รูปที่ 3.12 หน้าจอการแสดงผลเมื่อกดปุ่มค้นหา (Search)

หน้าจอการค้นหาโดยรูปทางซ้ายมือผู้ใช้จะมีอยู่ 2 ส่วน คือ

1. ระบุเส้นทางที่เป็นปลายทางที่ต้องการเดินทาง
2. ระบุเวลาที่ต้องการให้ระบบทำการค้นหาเส้นทางใหม่เพื่อทำการปรุงปรุงข้อมูลเส้นทางในขณะเดินทาง
3. การทำงานเมื่อผู้ใช้ระบุเส้นทางและกดปุ่ม SEARCH ระบบจะดึงเส้นทางทั้งหมดที่เป็นไปได้ให้ผู้ใช้ได้เลือก

3.4 ทดสอบและเก็บข้อมูล

ข้อมูลจากขั้นตอนนี้จะประกอบไปด้วย 2 ส่วนคือ ส่วนที่ 1 ข้อมูลจากการทำนายเวลาของ Google Map และข้อมูลจากการทำนายเวลาของของโปแกรมที่พัฒนาขึ้นในแต่ละโนด และส่วนที่ 2 คือข้อมูลที่ได้จากเวลาจริง ที่จะนำมาใช้ในการเปรียบเทียบเวลาที่ได้

3.5 ทดสอบประสิทธิภาพของการทำนายโดยใช้วิธี MSE

การทดสอบประสิทธิภาพของการทำนาย จะเป็นวิธีการหนึ่งจากหลายๆ วิธีการในการประเมินค่าการทำนายในสิ่งที่เป็นอนาคตและยังไม่เกิดขึ้นในการนำมาใช้ประโยชน์ โดยในงานวิจัยนี้จะนำการวัดประสิทธิภาพโดยการใช้ค่าการคำนวณ หาความผิดพลาดของการพยากรณ์ จากค่าเฉลี่ยความผิดพลาดยกกำลังสอง (Mean Squared Error : MSE) ซึ่งเป็นค่าเฉลี่ยความแตกต่างระหว่างการพยากรณ์ และค่าการสังเกตจำนวน n ครั้ง ยกกำลังสอง มาทำการตรวจสอบจากการวัดข้อมูลจริงที่เกิดขึ้นกับค่าของการทำนายเวลาของ Google Map กับค่าการทำนายเวลาของโปรแกรมที่พัฒนาขึ้น กับข้อมูลจำนวน n ครั้งที่จะใช้หาความแม่นยำ แต่ตัววัดเหล่านี้จะเป็นเพียงความแตกต่างของการพยากรณ์ที่เกิดขึ้นจริง ไม่ได้กำหนดถึงมาตรฐานของความผิดพลาดที่ยอมรับได้

3.6 สรุปผลการดำเนินการ

การวิเคราะห์และสรุปการวิจัยจะใช้ผลของการทดลองในการเก็บค่าข้อมูลจากเวลาที่ใช้จริงในการเดินทางและประเมินผลลัพธ์ตรวจสอบค่าการทำนายและการตรวจสอบประสิทธิภาพ

บทที่ 4
ผลการทดลอง

การทดลองจะแบ่งออกเป็น 2 ส่วน คือ ส่วนของการเก็บข้อมูลและการทดสอบการการใช้แผนที่เส้นทางที่ได้จากโปรแกรมประยุกต์ที่ทำการพัฒนาขึ้นและทำการเปรียบเทียบผลที่ได้กับโปรแกรม Google Map เพื่อวัดระยะเวลาที่ได้จากการคำนวณข้อมูลในอดีต กับเวลาที่ Google ได้แสดงในแผนที่

4.1 การเก็บข้อมูลเพื่อทำการทดลอง

งานวิจัยฉบับนี้จะเกี่ยวข้องกับแผนที่ ที่มีการวัดและการคำนวณจากระยะทางและเวลาก่อนสร้างแบบจำลองเพื่อพัฒนาโปรแกรมทำนายเวลาในการเดินทาง โดยระยะทางที่ได้จะเป็นระยะทางที่ได้จากเส้นทางที่ใช้อ้างอิงตามพิกัดทางภูมิศาสตร์ที่ Google มีบริการ และในส่วนของเวลาที่จะเป็นการส่งค่ากลับมาเพื่อให้ผู้ใช้ได้เลือกเส้นทางที่อาจมีเส้นทางในหลายๆ เส้นทางที่จะแสดงผลเมื่อมีการเคลื่อนที่และตำแหน่งของพิกัดเปลี่ยน หรือ การปรับปรุงเส้นทางตามการกำหนดเวลาที่ตั้งไว้เพื่อให้ได้เส้นทางที่เป็นไปได้ หากเกิดการเปลี่ยนแปลงที่อาจคาดเดาไม่ได้ เช่น การเกิดอุบัติเหตุ เป็นต้น

การทดสอบนี้จะทำการทดสอบและเก็บข้อมูลใน 2 ส่วนคือ

1. ข้อมูลที่ Google และโปรแกรมที่พัฒนาขึ้นแสดงเส้นทางและเวลาที่ใช้ก่อนการเดินทาง จะทำการเก็บข้อมูลแต่ละจุดเพื่อทำการเปรียบเทียบเวลาที่ใช้ก่อนการเดินทางจริงจาก Google Map และจากโปรแกรมที่สร้างขึ้น เพื่อตรวจสอบความแตกต่างของเวลา เมื่อจุดที่ 1-23 จะหมายถึงเส้นทางของจุดที่เชื่อมโยงกันรายละเอียดตามตารางที่ 4.2

ตารางที่ 4.1 ข้อมูลก่อนการเดินทางจริงจาก Google Map และจากโปรแกรมที่สร้างขึ้น

หน่วยการวัดเวลาเป็น นาที																							
เวลาที่ใช้	เวลาที่ใช้ไน	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
	A : GoogleMap	0.16	0.22	0.23	0.06	1.25	0.22	0.07	0.20	0.08	5.56	0.29	1.49	2.48	1.14	3.43	0.13	10.00	1.20	0.17	0.08	1.29	0.41	0.20
ในแต่ละจุด	B : ITM801	0.65	1.30	1.37	0.23	5.21	1.30	0.29	1.23	0.30	4.30	0.47	2.38	4.00	2.00	4.29	0.16	12.45	1.00	0.21	1.00	1.12	0.52	0.25

โดยเส้นทางในแผนที่จะมีรายละเอียดของโนดต่างๆ หรือจุดที่เชื่อมโยงกัน การเก็บข้อมูลในการทดลองจะพบว่าเวลาที่ใช้ของโปรแกรมทั้ง 2 โปรแกรมจะมีความใกล้เคียงกัน มีเฉพาะบางจุดที่พัฒนาขึ้นอาจมีข้อมูลที่มากกว่าอันเนื่องจากการคำนวณค่าตามรายละเอียดของจุดตามตารางที่ 4.2

ตารางที่ 4.2 รายละเอียดโนดตามเส้นทางจาก1/292 ซ.ชินเขต ถึงรถไฟฟ้าใต้ดินสถานีพหลโยธิน

เส้นทาง	รายละเอียด
1. ใช้ ซอย ชินเขต 2/6 แยก 2 ไปทาง ถนนหมายเลข 302 มุ่งไป แขวง ลาดยาว 5 มีระยะทาง 1.4 กม.	1. มุ่งหน้าทางใต้ ไปตามซอย งามวงศ์วาน 43 แยก 10-7 เข้าสู่ซอย งามวงศ์วาน 47 แยก 12 ระยะทาง 94 เมตร
	2. ขับต่อไปยัง ซอย ชินเขต 1/36 130 เมตร
	3. ขับต่อไปยัง ซอย งามวงศ์วาน 47 แยก 6-7 ระยะทาง 140 เมตร
	4. เลี้ยวขวา เข้าสู่ ซอย งามวงศ์วาน 47 แยก ระยะทาง 6 33 เมตร
	5. เลี้ยวซ้าย เข้าสู่ ซอย ชินเขต 2/6 แยก 2 ระยะทาง 750 เมตร
	6. เลี้ยวขวา ไปยัง ถนนหมายเลข 302 130 ระยะทาง เมตร
	7. เลี้ยวขวา ไปยัง ถนนหมายเลข 302 42 ระยะทาง เมตร
	8. เลี้ยวซ้าย ไปยัง ถนนหมายเลข 302 120 ระยะทาง เมตร
2. ใช้ ถนนหมายเลข 302 ระยะทาง 1 กม.	9. ใช้ 2 ช่องทางขวา เพื่อเลี้ยวซ้ายเข้าสู่ ถนนหมายเลข 302 15 เมตร
	10. ชิดขวา เพื่อวิ่งบน ถนนหมายเลข 302
3. ใช้ ซอย จันทรสถิตย์ และ ซอย สุธรรมอารีกุล ไปทาง ทางคู่ขนานถนนวิภาวดีรังสิต ระยะทาง 1 กม.	11.เลี้ยวซ้าย ไปยัง ซอย จันทรสถิตย์ ระยะทาง 59 เมตร
	12.เลี้ยวซ้าย เข้าสู่ ซอย จันทรสถิตย์ ระยะทาง 300 เมตร
	13.ขับต่อไปยัง ซอย สุธรรมอารีกุล ระยะทาง 500 เมตร
	14. เลี้ยวซ้าย ที่ ซอย ไพฑูรย์อิงคสุวรรณ ระยะทาง 150 เมตร
4. เดินทางต่อไปบน ทางคู่ขนานถนนวิภาวดีรังสิต ไปยังจุดหมายที่แขวงจตุจักร	15.เลี้ยวซ้าย เข้าสู่ ทางคู่ขนานถนนวิภาวดีรังสิต ระยะทาง 1.0 กม.
	16.ใช้ช่องทางขวา เพื่อใช้ทางลาดไป ดินแดง ทางพิเศษ ระยะทาง 38 เมตร
	17. ตัดเข้าไปยัง ถนนหมายเลข 31 ระยะทาง 2.9 กม.
	18.ใช้ช่องทางซ้าย เพื่อใช้ทางออก 11 ไปทาง ลาดพร้าว/สะพานควาย ระยะทาง 210 เมตร
	19.ใช้ช่องทางซ้าย เพื่อตัดเข้าสู่ ทางคู่ขนานถนนวิภาวดีรังสิต ระยะทาง50 เมตร
	20. ใช้ช่องทางซ้าย เพื่ออยู่บน ทางคู่ขนานถนนวิภาวดีรังสิต ระยะทาง 22 เมตร
	21.ใช้ช่องทางซ้าย เพื่อใช้ทางลาดไป ลาดพร้าว ระยะทาง 260 เมตร
	22. ชิดซ้ายตรงทางแยก ตามป้ายบอกทาง Kamphaeng Phet Road และตัดเข้าสู่ ถนนหมายเลข 1 ระยะทาง 120 เมตร
	23.ใช้ทางออก ลาดพร้าว เข้าสู่ บางกะปิ

2. ข้อมูลการใช้เวลาจริงในการเดินทางเพื่อใช้ในการเปรียบเทียบผลของการทดลองที่ได้การคำนวณเวลาที่ใช้ในแต่ละเส้นทางจากพิกัดปกติที่ดึงได้จากแผนที่ตามพิกัดที่ได้จาก Google จะมีการปรับปรุงเส้นทางภายใต้เงื่อนไขคือเมื่อพิกัดเปลี่ยนหรือถึงเวลาที่ตั้งไว้ที่โปรแกรมจะต้องทำการค้นหาเส้นทางใหม่ โดยการทดสอบจะตรวจสอบการทำงานกรณีการปรับปรุงเส้นทางใหม่ที่จะส่งผลให้การเดินทางได้เร็วขึ้น เมื่อเทียบกับเส้นทางที่ดึงได้จากการระบุเส้นทางในครั้งแรก

ดังนั้น เส้นทางที่จะทำการทดสอบจะต้องมีเส้นทางที่หลากหลายในการเดินทางเพื่อเก็บข้อมูลได้เพียงพอ และเพื่อตรวจสอบการทำงานเมื่อมีการเปลี่ยนแปลงเส้นทางอีกครั้ง จึงได้เลือกเส้นทางจาก ซอยชินเขต 1/55 ถนนงามวงศ์วาน ไปยังสถานีรถไฟฟ้าใต้ดินสถานีพหลโยธิน .เนื่องจากเป็นเส้นทางที่มีระยะทางที่มีเส้นทางที่มีข้อมูลที่ต้องการเพียงพอ และจากการใช้เส้นทางลัดที่ผู้ทำการทดสอบเคยเปลี่ยนเส้นทางเองจากการใช้งานเป็นประจำ

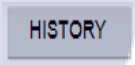
4.2 การเก็บข้อมูลและสร้างโปรแกรมค้นหาเส้นทาง

1. ศึกษาความต้องการและความเป็นไปได้ของโปรแกรมที่พัฒนา ระบบทำการดึงแผนที่จาก Google ไม่มีการเก็บข้อมูล และใช้ข้อมูลในอดีตทำการแจ้งเวลาที่ดีที่สุดใจการวางแผนเส้นทางบนระบบปฏิบัติการระบบ Android มาใช้ในการพัฒนา เนื่องจากเป็นโปรแกรมที่เป็นระบบเปิดสามารถทำพัฒนาและการทดลองได้ง่าย

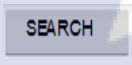
2. โปรแกรมที่ได้จะทำการประมวลผลจะเก็บข้อมูลและแบ่งเป็น 2 ส่วนคือ ผลลัพธ์ในส่วนของเวลาก่อนการเดินทางและเวลาที่ใช้ตามจริงในการเดินทาง

3. อุปกรณ์ที่ใช้ในการทดสอบและลงโปรแกรมที่จัดทำขึ้นเป็นหลัก คือ Smart Phone ยี่ห้อ Lava Grand Pro 5.5 การทำงานผู้ใช้ต้องเชื่อมต่อกับผู้ให้บริการเครือข่ายและเปิดระบบการนำทาง GPS เพื่อขอเส้นทาง สามารถใช้บริการของ GSM และบริการ WI-FI

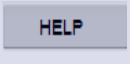
4. ขั้นตอนการประมวลผลการทำงาน จะประกอบไปด้วยเริ่มทำงาน ผู้ใช้งานเปิดการใช้ข้อมูลเครือข่าย ระบบการนำทาง และโปรแกรมที่พัฒนาขึ้นโดยมีหน้าจอหลักการทำงาน จะประกอบไปด้วย

4.1 

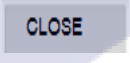
เรียกหน้าจอการเก็บบันทึกเส้นทางที่ User เคยเดินทาง

4.2 

เรียกหน้าจอการค้นหาเส้นทางที่ User ต้องทำการระบุ

4.3 

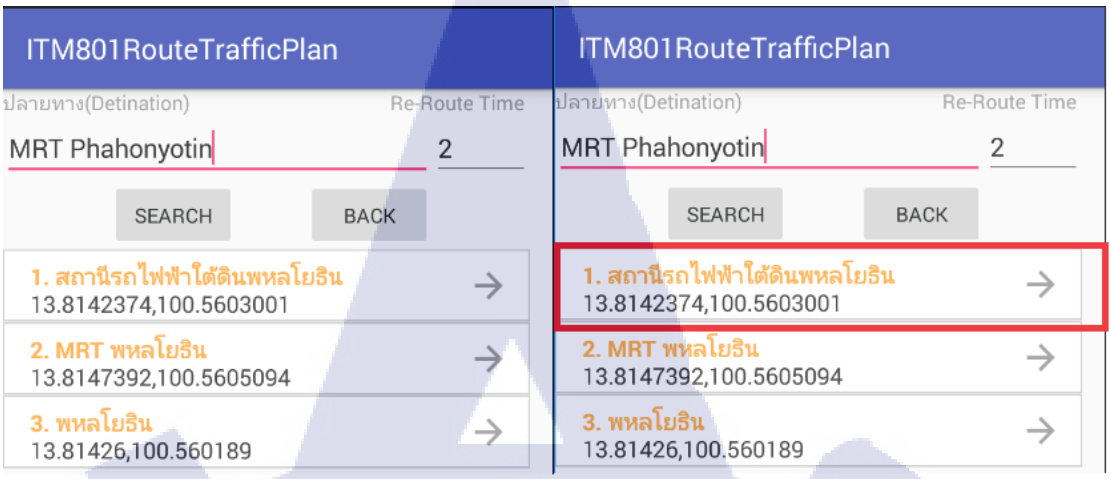
เรียกหน้าจอแนะนำระบบการทำงาน

4.4 

ออกจากระบบเมื่อทำการเลือกที่ปุ่มหรือเมื่อกดย้อนกลับ ( )

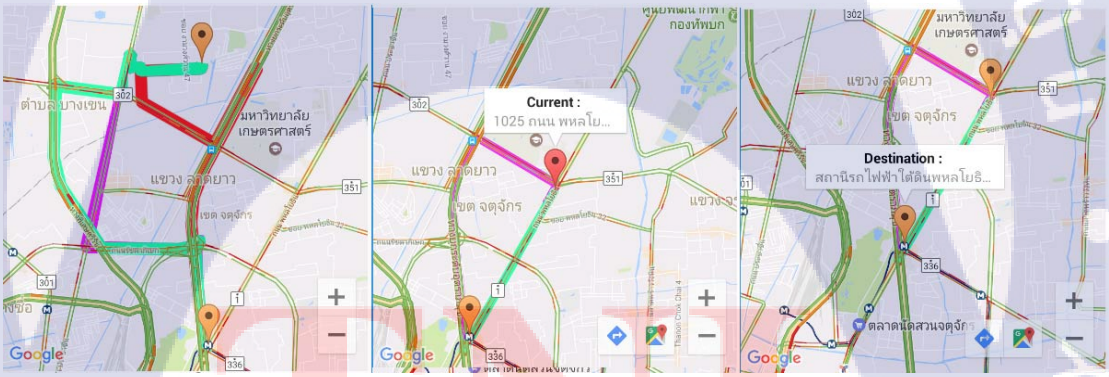
4.5 หน้าจอแสดงแผนที่ในการเดินทาง

4.5.1 ผู้ใช้ระบุจุดปลายทางที่ต้องการไป (ด้วยชื่อ) เพื่อให้โปรแกรมตรวจสอบและดึงข้อมูลของชื่อสถานที่ที่ต้องการไป Match กับข้อมูลของ Google Map สำหรับให้ผู้ใช้เลือกสถานที่อีกครั้ง และระบุระยะเวลาหน่วยเป็นนาที เพื่อกำหนดให้โปรแกรมทำการค้นหาเส้นทางใหม่ตามเวลาที่กำหนด ตามรูปด้านล่าง



รูปที่ 4.1 รายการเส้นทางที่เป็นไปได้ตามชื่อที่ผู้ใช้ค้นหา

4.5.2 เมื่อผู้ใช้เลือกเส้นทางที่ต้องการแล้ว โปรแกรมจะทำการสร้างและแสดงเส้นทางต่างๆ ตามพิกัดที่ผู้ใช้เลือก แสดงรายการแยกตามสีต่าง และทำการค้นหาเส้นทางใหม่อีกครั้งเมื่อพบว่ามีพิกัดเปลี่ยนไปหรือครบระยะเวลาที่ต้องการให้โปรแกรมหาเส้นทาง



รูปที่ 4.2 เส้นทางหลังผู้ใช้เลือกสถานที่ต้องการไปและการปรับปรุงเส้นทาง

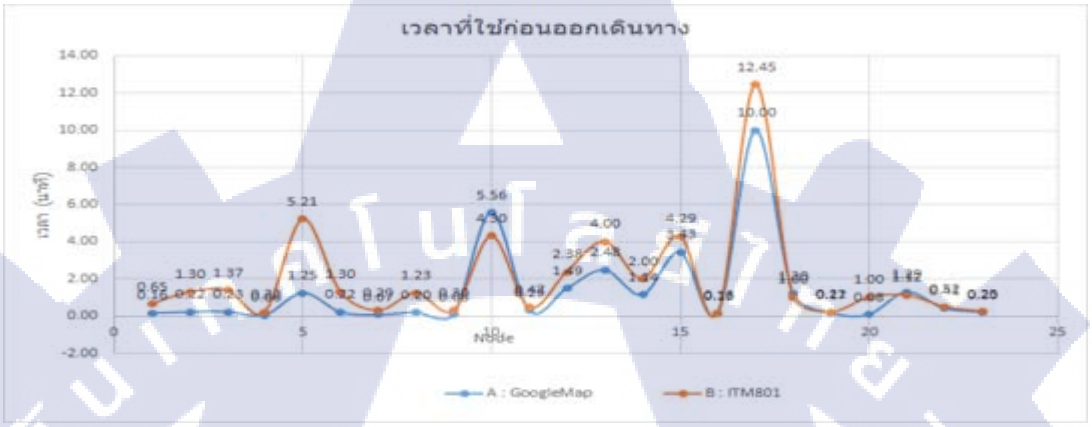
4.3 ผลการทดลองการเก็บข้อมูล

1. การเก็บข้อมูลเวลาก่อนการเดินทางจริงแต่ละโนดหรือจุด นำมาเปรียบเทียบก่อนการทดสอบจากการเก็บเวลาจริงที่ใช้ในการเดินทางผลที่ได้ดังนี้

ตารางที่ 4.3 เวลาที่ใช้ในการเดินทางก่อนเดินทางจริง (จุด 1-23 คือ จุดตามเส้นทางตามตารางที่ 4.2)

หน่วยการวัดเวลาเป็น นาที

เวลาที่ใช้	เวลาที่ใช้ใน	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
	A : GoogleMap	0.16	0.22	0.23	0.06	1.25	0.22	0.07	0.20	0.08	5.56	0.29	1.49	2.48	1.14	3.43	0.13	10.00	1.20	0.17	0.08	1.29	0.41	0.20
ในแต่ละจุด	B : ITM801	0.65	1.30	1.37	0.23	5.21	1.30	0.29	1.23	0.30	4.30	0.47	2.38	4.00	2.00	4.29	0.16	12.45	1.00	0.21	1.00	1.12	0.52	0.25



รูปที่ 4.3 เวลาที่ใช้ก่อนการเดินทางจริงจาก Google Map และจากโปรแกรมที่สร้างขึ้น

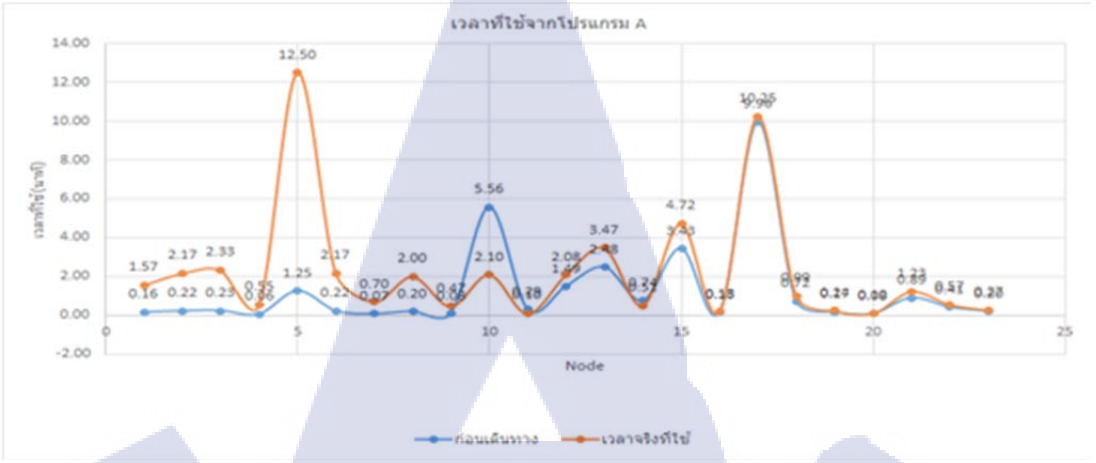
2. การเก็บข้อมูลการทดสอบในเวลาจริงในขณะเดินทาง เพื่อเปรียบเทียบข้อมูลเวลาการเดินทางที่ได้จริงเทียบกันของ Google Map และ โปรแกรมที่สร้างขึ้น

2.1 เปรียบเทียบข้อมูลโปรแกรม A : Google Map ก่อนเดินทางกับเวลาจริง ตามจุดที่เชื่อมโยงกันตั้งแต่จุดที่ 1-23 รายละเอียดแต่ละจุดตามตารางที่ 4.2

ตารางที่ 4.4 ข้อมูลเวลาของ Google Map ที่แจ้งก่อนเดินทางและข้อมูลเวลาในการเดินทางจริง

หน่วยการวัดเวลาเป็น นาที

เวลาจาก	เวลาที่ใช้ใน	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
โปรแกรม A : Google Map	แต่ละจุด																							
	ก่อนเดินทาง	0.16	0.22	0.23	0.06	1.25	0.22	0.07	0.20	0.08	5.56	0.29	1.49	2.48	0.74	3.43	0.13	9.96	0.72	0.17	0.08	0.89	0.41	0.20
	เวลาจริงที่ใช้	1.57	2.17	2.33	0.55	12.50	2.17	0.70	2.00	0.47	2.10	0.10	2.08	3.47	0.51	4.72	0.18	10.25	0.99	0.24	0.10	1.23	0.57	0.27



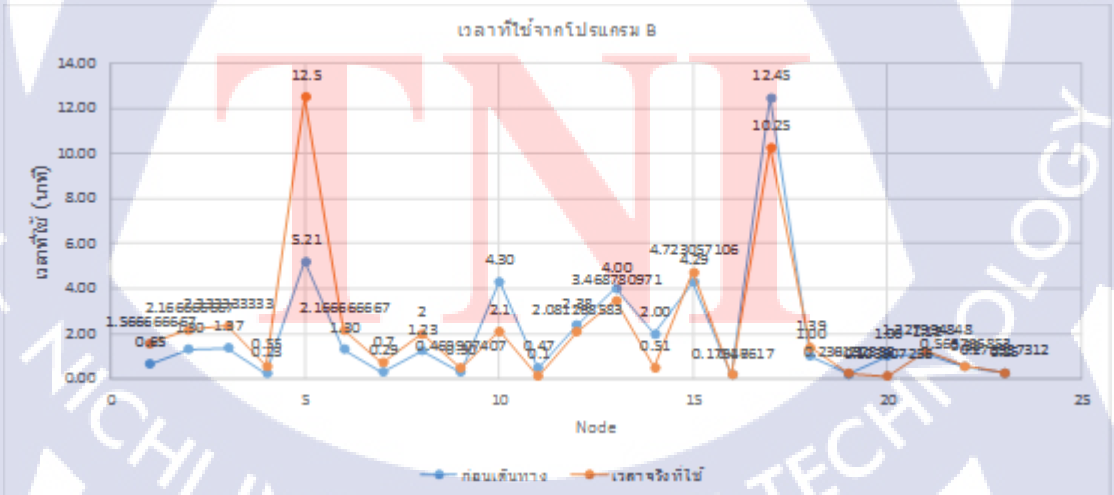
รูปที่ 4.4 เวลาที่ Google Map แสดงก่อนการเดินทางกับเวลาที่ใช้จริงของ Google Map

2.2 เปรียบเทียบข้อมูลเวลาจากโปรแกรม B : ITM801 ก่อนเดินทางกับเวลาจริง ตามจุดที่เชื่อมโยงกันตั้งแต่จุดที่ 1-23 (รายละเอียดแต่ละจุดตามตารางที่ 4.2)

ตารางที่ 4.5 ข้อมูลเวลาของโปรแกรมที่พัฒนาขึ้นที่แจ้งก่อนเดินทางและข้อมูลเวลาในการเดินทางจริง

หน่วยการวัดเวลาเป็น นาที

เวลาที่ใช้	เวลาที่ใช้ใน	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
เวลาจากโปรแกรม B : ITM801 (นาที)	ก่อนเดินทาง	0.65	1.30	1.37	0.23	5.21	1.30	0.29	1.23	0.30	4.30	0.47	2.38	4.00	2.00	4.29	0.16	12.45	1.00	0.21	1.00	1.12	0.52	0.25
	เวลาจริงที่ใช้	1.57	2.17	2.33	0.55	12.5	2.17	0.7	2	0.47	2.1	0.1	2.08	3.47	0.51	4.723	0.18	10.3	1.39	0.24	0.1	1.23	0.57	0.27



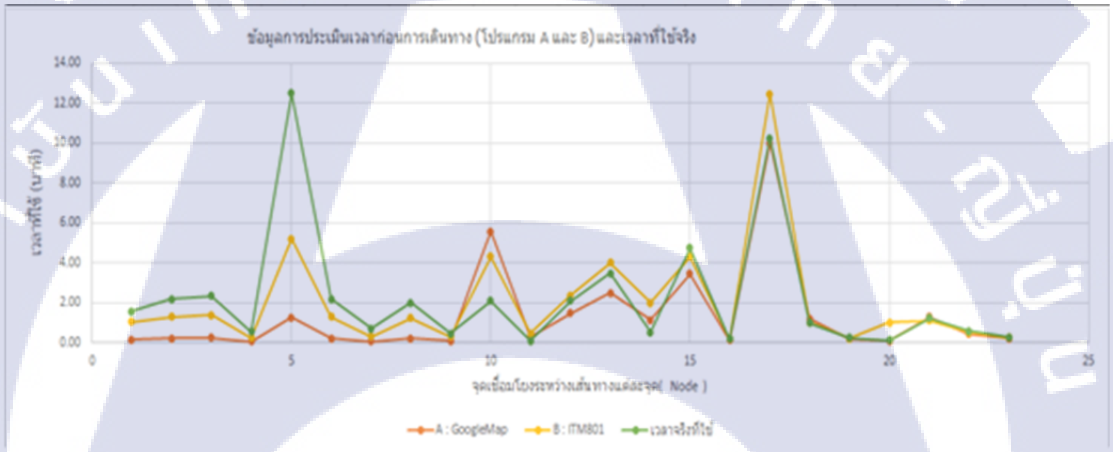
รูปที่ 4.5 เวลาจากโปรแกรมที่พัฒนาที่แสดงก่อนการเดินทางกับเวลาที่ใช้จริง

2.3 เปรียบเทียบข้อมูลเวลาจากโปรแกรม A: Google Map, B : ITM801 ที่จัดเก็บก่อนเดินทางกับข้อมูลตามเวลาจริงในแต่ละจุดตามตารางที่ 4.6 โดยแสดงการใช้เวลาในแต่ละจุดตั้งแต่จุดที่ 1-23 (รายละเอียดแต่ละจุดตามตารางที่ 4.2)

ตารางที่ 4.6 ข้อมูลเวลาที่ได้จากโปรแกรม A: Google Map , B : ITM801 และเวลาที่ใช้ในการเดินทางจริง

หน่วยการวัดเวลาเป็น นาที

เวลาที่ใช้ในแต่ละจุด	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
A : GoogleMap	0.16	0.22	0.23	0.06	1.25	0.22	0.07	0.20	0.08	5.56	0.29	1.49	2.48	1.14	3.43	0.13	10.00	1.20	0.17	0.08	1.29	0.41	0.20
B : ITM801	1.05	1.30	1.37	0.23	5.21	1.30	0.29	1.23	0.30	4.30	0.47	2.38	4.00	2.00	4.29	0.16	12.45	1.00	0.21	1.00	1.12	0.52	0.25
เวลาจริงที่ใช้	1.57	2.17	2.33	0.55	12.50	2.17	0.70	2.00	0.47	2.10	0.10	2.08	3.47	0.51	4.72	0.18	10.25	0.99	0.24	0.10	1.23	0.57	0.27



รูปที่ 4.6 การเปรียบเทียบเวลาที่คาดการณ์ก่อนการเดินทางจากโปรแกรม A และ B และเวลาที่ใช้ในการเดินทางจริง

จะพบว่าข้อมูลเวลาที่คาดการณ์ก่อนการเดินทางทั้งโปรแกรม A และ โปรแกรม B ต่างมีความคลาดเคลื่อนซึ่งจะทำการตรวจสอบโดย การคำนวณหาค่าความผิดพลาดของการพยากรณ์ จากค่าเฉลี่ยความผิดพลาดยกกำลังสอง) Mean Squared Error : MSE) ซึ่งเป็นค่าเฉลี่ยของความแตกต่างระหว่างการพยากรณ์ และค่าการสังเกต n กำลังสอง

$$MSE = \sum \frac{\text{ค่าความคลาดเคลื่อน}^2}{n}$$

ตารางที่ 4.7 การตรวจสอบความคลาดเคลื่อนของการพยากรณ์เวลาที่ใช้เทียบกับเวลาที่ใช้จริง

จุดเชื่อมโยง (Node)	เวลาที่ใช้ จริง	ค่าพยากรณ์ A : GoogleMap	ค่าพยากรณ์ B : ITM801	ค่าความคลาดเคลื่อน^2 A : GoogleMap	ค่าความคลาดเคลื่อน^2 B : ITM801
1	1.57	0.16	1.05	1.99	0.27
2	2.17	0.22	1.30	3.80	0.75
3	2.33	0.23	1.37	4.41	0.93
4	0.55	0.06	0.23	0.25	0.10
5	12.50	1.25	5.21	126.56	53.12
6	2.17	0.22	1.30	3.80	0.75
7	0.70	0.07	0.29	0.40	0.17
8	2.00	0.20	1.23	3.24	0.59
9	0.47	0.08	0.30	0.15	0.03
10	2.10	5.56	4.30	11.94	4.84
11	0.10	0.29	0.47	0.04	0.14
12	2.08	1.49	2.38	0.35	0.09
13	3.47	2.48	4.00	0.98	0.28
14	0.51	1.14	2.00	0.40	2.22
15	4.72	3.43	4.29	1.66	0.18
16	0.18	0.13	0.16	0.00	0.00
17	10.25	10.00	12.45	0.06	4.85
18	0.99	1.20	1.00	0.04	0.00
19	0.24	0.17	0.21	0.00	0.00
20	0.10	0.08	1.00	0.00	0.80
21	1.23	1.29	1.12	0.00	0.01
22	0.57	0.41	0.52	0.02	0.00
23	0.27	0.20	0.25	0.01	0.00
ผลรวมของค่าคลาดเคลื่อนยกกำลัง 2		$\frac{\sum(\text{ความคลาดเคลื่อน})^2}{n}$		6.96	3.05

จากผลการทดลองพบว่าโปรแกรมสามารถให้เส้นทางที่หลากหลายและมีเส้นทางที่มีระยะทางและเวลาที่สั้นที่สุดมาแสดงให้ผู้ใช้งานสามารถตัดสินใจได้ว่าต้องการจะใช้เส้นทางใดในการเดินทางครั้งนั้น และให้เวลาก่อนการเดินทางและเวลาที่ใช้จริงได้ใกล้เคียงกว่าการใช้ Google Map โดยการหาค่าความผิดพลาดที่ได้จากการพยากรณ์ของ Google Map และ โปรแกรมที่พัฒนาขึ้นมาเทียบกับเวลาจริงที่ใช้ในการเดินทางพบว่ามีค่าของความคลาดเคลื่อนที่น้อยกว่า Google Map จึงทำให้สามารถวางแผนการเดินทางได้ดีกว่าในการวางแผนการเดินทางล่วงหน้าเพราะมีการใช้ข้อมูลในอดีตมาทำการประเมินเส้นทางไว้ก่อน

บทที่ 5
การวิเคราะห์ผลการทดลอง สรุปผลงานวิจัย และข้อเสนอแนะ

5.1 วิเคราะห์ผลการทดลอง และสรุปผลงานวิจัย

วิทยานิพนธ์นี้นำเสนอการค้นหาเส้นทางที่เป็นการทำนายเวลาในการเดินทางที่สั้นที่สุดโดยใช้เวลาในอดีตมาทำการคำนวณเวลาในการเดินทางและหลังจากการทำการทดลองโดยการเก็บข้อมูลเพื่อเปรียบเทียบกับใน Google Map แล้วจะทำการตรวจสอบประสิทธิภาพของการทำนายว่ามีประสิทธิภาพและความแม่นยำในการดำเนินการหรือไม่ โดยในงานวิจัยนี้จะนำการวัดประสิทธิภาพโดยการใช้ค่าการคำนวณ หาความผิดพลาดของการพยากรณ์ จากค่าเฉลี่ยความผิดพลาดยกกำลังสอง (Mean Squared Error : MSE) ซึ่งเป็นค่าเฉลี่ยความแตกต่างระหว่างการพยากรณ์ และค่าการสังเกตจำนวน n ครั้ง ยกกำลังสอง มาทำการตรวจสอบอีกครั้ง ดังนี้

เมื่อ ค่าในเวลาจริง หมายถึง ข้อมูลจริงที่เกิดขึ้น

ค่าการพยากรณ์ หมายถึงข้อมูลที่เกิดขึ้นจากการคำนวณเพื่อนำมาพยากรณ์

และ n หมายถึง จำนวนของข้อมูลที่นำมาใช้วัดความแม่นยำ

แต่ตัววัดเหล่านี้เพียงแต่บอกถึงความแตกต่างของการพยากรณ์ที่เกิดขึ้นจริง ไม่ได้กำหนดถึงมาตรฐานของความผิดพลาดที่ยอมรับได้และการตรวจสอบหาข้อผิดพลาดในการพยากรณ์ค่าที่ได้ต้องไม่มากไปกว่างานเดิมที่มีก่อนหน้าจึงจะสามารถการนำไปใช้งานจริง สำหรับค่าเวลาที่ได้จากการพยากรณ์เป็นแค่ค่าที่เกิดจากการคำนวณและการคาดคะเนจึงควรมีความถูกต้องแม่นยำเพียงพอ ถึงแม้ว่าความถูกต้องของการพยากรณ์จะขึ้นอยู่กับการใช้งานที่แท้จริงของผู้ใช้งานและปัจจัยภายนอกที่ไม่สามารถควบคุมได้ แต่สามารถควบคุมกระบวนการในการพยากรณ์ให้มีประสิทธิภาพนั้นได้

สำหรับผลการทดลองใช้ในการเดินทางบนท้องถนน จะพบว่ามีหลายปัจจัยที่ส่งผลกับการเดินทางไม่ใช่เพียงแต่ระยะเวลา ความเร็วการเคลื่อนที่ของรถยนต์ หรือระยะทางเท่านั้น ในเรื่องของเส้นทาง สภาพของถนน และเหตุการณ์ต่างๆ เช่น การเกิดอุบัติเหตุ หรือ เส้นทางที่มีห้างสรรพสินค้าที่มักติดขัดมากในช่วงวันหยุด เป็นต้น

การเก็บข้อมูลเส้นทางแผนที่และระยะเวลาสำหรับการทดลองนี้ สามารถดำเนินการเก็บข้อมูลได้ครบถ้วนเนื่องจากเส้นทางที่ใช้มีปัจจัยต่างๆ สำหรับการทดสอบซึ่งผลของการบันทึกได้ก็เห็นถึงข้อแตกต่างในผลลัพธ์ที่ได้ โดยในงานนี้การใช้บริการของ Google และการให้บริการทางด้าน

เครือข่ายของอุปกรณ์ที่ใช้ทดสอบที่มีและเรานำมาใช้ นั่นก็คือ บริการการแจ้งตำแหน่งทางภูมิศาสตร์ ในส่วนของข้อมูลในการเรียกใช้อาจมีคลาดเคลื่อนเพียงอันเนื่องมาจากชื่อสถานที่ตั้งที่ระบุไม่ชัดเจน

การประมวลผลข้อมูลที่ได้มาทำการทดสอบและทำการเปรียบเทียบเส้นทางในแต่ละเส้น การค้นหาเส้นทางใหม่จากเงื่อนไขที่ตั้งขึ้นจากการตรวจสอบพิกัดใหม่ที่เป็นต้นทางเมื่อมีการเปลี่ยนแปลงเส้นทางหรือระยะเวลาที่มีการตั้งให้โปรแกรมทำการปรับปรุง ซึ่งโดยปกติจะมีดำเนินการทุก 5 นาที ในการปรับปรุงเส้นทางใหม่ พบว่าเส้นทางที่ได้มาอาจมีการเปลี่ยนเส้นทางไปจากเดิม คือ ไม่เป็นแม้แต่จุดเชื่อมต่อตั้งแต่แรกที่โปรแกรมดึงเส้นทางให้ในครั้งแรก แต่โดยมากจะเป็นเส้นทางที่เป็นจุดเชื่อมต่อเดิม แต่เมื่อพิกัดของผู้ใช้เปลี่ยนจึงทำให้เส้นทางนี้กลับขึ้นมาเป็นเส้นทางแนะนำ า ร คำนวณและการสร้างเส้นทางที่ต้องพิจารณาและตรวจสอบปัจจัยที่ส่งผลกระทบกับเส้นทางจึงต้องให้ความสำคัญเป็นอย่างมาก เนื่องจากการทดสอบจะเห็นว่าในบางเวลาเส้นทางที่เคยใช้เวลาไม่มากนักกลับกลายเป็นเส้นทางที่ใช้เวลานานทั้งที่เป็นวันหยุด ซึ่งเมื่อตรวจสอบจะเป็นว่าเส้นทางนั้นมีห้างสรรพสินค้าใหญ่อย่างห้างเซ็นทรัลลาดพร้าว ยูเนี่ยนมอลล์และทางไปสวนจตุจักร ที่ผู้คนจะให้ความสนใจไปในวันหยุดที่อาจส่งผลให้รถที่จราจรมีมากขึ้น

ดังนั้นจากผลของการเก็บข้อมูลและการทดลอง มาทำการเปรียบเทียบ ระบบที่พัฒนาภายใต้ระบบปฏิบัติการแอนด์ดรอยด์ สามารถให้เส้นทางที่หลากหลายแต่มีเส้นทางที่มีระยะทางและเวลาที่สั้นที่สุดมาแสดงให้ผู้ใช้งานสามารถตัดสินใจได้ว่าต้องการจะใช้เส้นทางใดในการเดินทางครั้งนั้น และสามารถทำนายเวลาที่ใช้ในการเดินทางได้ดีกว่าการคาดคะเนเวลาที่ได้จาก Google ที่จุดเริ่มต้นการเดินทางโดยทำการตรวจสอบจาก MSE (Mean Squared Error : ค่าเฉลี่ยความผิดพลาดยกกำลังสอง) ที่พบว่ามีความคลาดเคลื่อนน้อยกว่าเมื่อเทียบกับ MSE ของ Google Map ถึงแม้ว่าระยะทางจริงจะมีความคลาดเคลื่อนออกไปบ้างจากพิกัดทางภูมิศาสตร์ ที่เกิดจากการป้อนข้อมูลของผู้ใช้แต่การคัดกรองเพิ่มเติมจากการดึงลิสต์รายการให้ผู้ใช้ตรวจสอบก่อนเพื่อแจ้งปลายทางที่ชัดเจนขึ้นเป็นการกรองเส้นทางเบื้องต้นก่อนแสดงเส้นทางบนแผนที่ที่เป็นไปได้โดยแยกแต่ละเส้นทางด้วยสีอย่างชัดเจน ทำให้ผลลัพธ์การเดินทางมีการคัดเลือกและกลั่นกรองก่อนนำไปใช้และยังเพิ่มทางเลือกให้กับผู้ใช้งานพิจารณาเส้นทางเพราะการเดินทางครั้งนั้นเส้นทางที่ผู้ใช้อาจไม่ใช่เส้นทางที่สั้นที่สุดแต่แรกแต่สามารถนำที่จะได้รับเป็นเส้นทางที่ดีที่สุดสำหรับการเดินทางในครั้งนั้น

5.2 ข้อเสนอแนะ

สำหรับงานวิจัยในอนาคตควรมีข้อเสนอแนะด้วยกัน 3 เรื่อง

1. ข้อมูลของเส้นทางและพิกัดทางภูมิศาสตร์ที่ควรต้องเพิ่มเติมในส่วนของฐานข้อมูลเพื่อเปรียบเทียบกับข้อมูลที่ดึงมาได้จากบริการของทาง Google เพราะในบางพื้นที่อาจมีการตัดถนนเพิ่มหรือการปรับสถานที่ซึ่งทำให้เส้นทางการเดินรถและในบางครั้งเมื่อเราใช้ข้อมูลทางพิกัดที่ได้รับอาจมี

ความคลาดเคลื่อน ในเรื่องของชื่อสถานที่ โดยอาจทำการเพิ่มโปรแกรมในส่วนของการดูแลรักษาเพื่อให้ผู้ใช้งานแจ้งชื่อสถานที่เองและอ่านพิกัดจากเครื่องของผู้ใช้จนกว่าเมื่อในแผนที่เส้นทางที่ทาง Google จะมีและทางโปรแกรมที่พัฒนาจะทำการปรับปรุงข้อมูลให้เป็นปัจจุบัน การเพิ่มการเชื่อมโยงและการจัดเก็บข้อมูลเพื่อให้ได้เส้นทางที่มากที่สุดในการคำนวณเพื่อทำนายการเดินทางที่เม่นยำมากขึ้นและนำไปใช้ได้กับผู้ใช่อื่นๆ ที่ใช้ในเส้นทางเดียวกัน

2. การหาค่าเส้นทางและเวลาที่อาจต้องทำการผนวกในเรื่องการวิเคราะห์สถานการณ์หรือเหตุการณ์ใดๆ ที่จะทำให้การจราจรในแต่ละจุดเกิดปัญหา เพื่อที่จะได้นำมาผนวกกับการเพิ่มค่าปัจจัยอื่นๆ ก็จะทำให้สามารถนำค่าปัจจัยต่างๆ มาทำการให้น้ำหนักในการคำนวณเวลาได้อย่างแม่นยำหรือมีการผิดพลาดทางด้านการคำนวณไม่มาก

3. การเก็บข้อมูลและการสำรวจเส้นทางจากการใช้โดยจากผู้ใช้จริงจะเป็นการบันทึกข้อมูลจากสภาพการใช้งานจริงๆ และพฤติกรรมการเดินทางของผู้ใช้ ที่สามารถนำมาทำการวิเคราะห์เพิ่มเติมสำหรับการแนะนำเส้นทาง ที่สามารถนำมาสร้างกลุ่มของผู้ใช้ก่อนการให้เส้นทาง และมีความน่าเชื่อถือเหมาะกับสภาพการสัญจรและพฤติการณ์การขับขี่ของผู้ใช้

บรรณานุกรม

บรรณานุกรม

[1] Transportation Research Board, *Highway Capacity Manual (HCM2000)*, The National Academy of Sciences, Washington D.C. : National Research Council, 2000.

[2] Google Map About, *"Google Maps,"* [Online]. Available : https://www.google.com/maps/about/. [Accessed : November 20, 2016].

[3] Google Direction API, *"Google Direction,"* [Online]. Available : https://developers.google.com/maps/documentation/geolocation/intro. [Accessed : November 10, 2016].

[4] มูลนิธิศูนย์ข้อมูลจราจรอัจฉริยะไทย (ITICFoundtion), *"กระบวนการรับและส่งข้อมูลจราจร,"* [Online]. Available : http://www.iticfoundation.org/whyitic. [Accessed : 10 กันยายน 2558].

[5] S. Panda et al., "Traffic management using swarm intelligence and route selection using android application," *International Journal of Engineering and Innovative Technology,* vol. 5, no. 1, pp. 59-63, December 2015.

[6] กิตติเดช วงศ์ศักดิ์ และเกียรติศักดิ์ โยชะนัง, "ระบบการค้นหาเส้นทางหลักที่สั้นที่สุดและเส้นทางรองโดยใช้ขั้นตอนวิธีแบบมด," *The 6th National Conference On Computing And Information Technology,* NCCIT2010, กรุงเทพฯ, 17 พฤษภาคม 2553, หน้า 161-166.

[7] ราชการ ปรึกษาดี และสุนันฑา สดสี, "การเปรียบเทียบหาเส้นทางที่เหมาะสมโดยวิธีระบบมดและ Dijkstra's Algorithm," *The National Conference on Computing and Information Technology,* NCCIT'08, กรุงเทพฯ, 20 กรกฎาคม 2555, หน้า 572-577.

[8] T. Peng and X. Wang, *"A Mobile-based Navigation Web Application : Finding the Shortest-Time Path based on Factor Analysis,"* B.S. Thesis (Geomatics), University of Gävle, Gävle, Sweden, 2012.

[9] B. T. Morris and M. Manubhai, "Understatnding vehicular traffic behavior from video : a survey of unsupervised approaches," *Journal of Electronic Imaging,* vol. 22, no. 4, pp. 1-15, Octerber-December 2013.

[10] P. Doshi et al., "Location based services and integration of google maps in android," *International Journal of Engineering And Computer Science,* vol. 3, no. 3, pp. 5,072-5,077, March 2014.

[11] M. Singhal and A. Shukla, "Implementation of location based services in android using GPS and web services," *IJCSI International Journal of Computer Science Issues,* vol. 9, no. 2, pp. 237-242, January 2012.

[12] S. Bhatia and S. Hilal, "A new approach for location based tracking," *International Journal of Computer Science Issues,* vol. 10, no. 1, pp. 73-77, May 2013.

[13] D. Suvarna et al., "Traffic reduction using android application," *International Journal on Advanced Computer Theory and Engineering, IJACTE,* vol. 2, no. 3, pp. 2,319-2,526, December 2014.

[14] S. S. Aung and T. T. Naing, *"Naïve bayes classifier based traffic detection system on cloud infrastructure,"* 6th International Conference on Intelligent Systems, Modelling and Simulation, Intelligent Systems, Modelling and Simulation, ISMS, Kuala Lumpur, Malaysia, 9-12 Februuuary 2015, pp. 193-198.

[15] P. Shah et al., "Location based reminder using GPS for mobile (android)," *ARPN Journal of Science and Technology,* vol. 2, no. 4, pp. 377-380, May 2012.

[16] O. A. Ibrahim and K. J. Mohsen, "Design and implementation an online location based services using google maps for android mobile," *International Journal of Computer Networks and Communications Security,* vol. 2, no. 3, pp. 113-118, March 2014.

[17] V. Thakare and S. Honale, "Enhanced functionality for tracing places, weather forecasting and geo fencing using android," *IJSRD-International Journal for Scientific Research & Development,* vol. 3, no. 4, pp. 814-817, May 2016.

[18] T. Yigit and O. Unsal, "Using the ant colony algorithm for real-time automatic route of school buses," *The International Arab Journal of Information Technology,* vol. 13, no. 5, pp. 559-565, September 2016.

[19] R. Shinde et al., "CLICKnSAVE-AN android application," *International Research Journal of Engineering and Technology, IRJET,* vol. 3, no. 10, pp. 1,356-1,360, November 2016.

[20] A. Renugambal and V. A. Kameswari, “Finding optimal vehicular route based on GPS,” *(IJCSIT) International Journal of Computer Science and Information Technologies,* vol. 5, no. 2, pp. 1,330-1,332, January 2014.

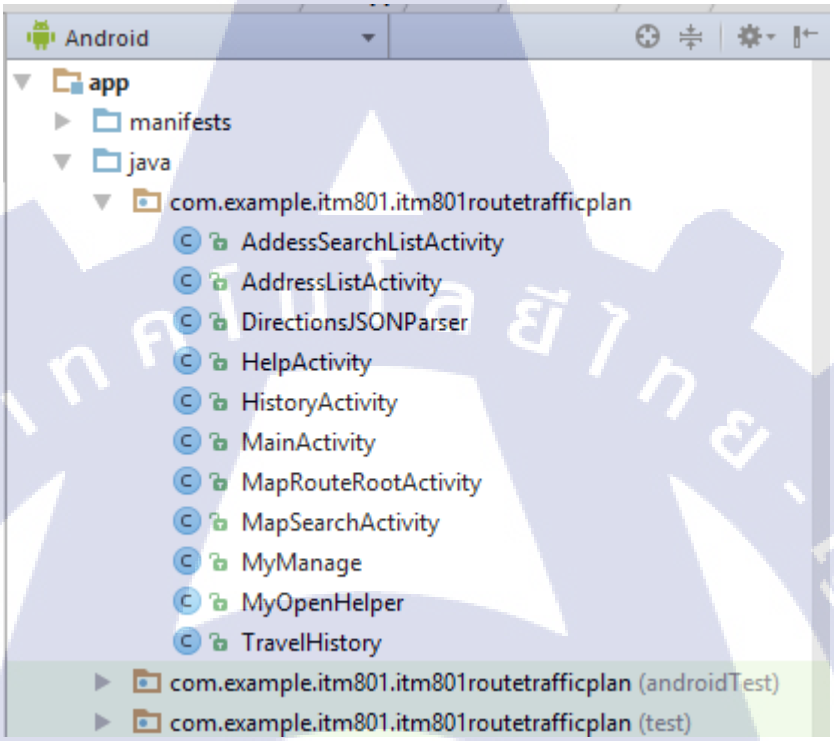
ภาคผนวก

ภาคผนวก ก.

ซอร์สโค้ดโปรแกรมบนสมาร์ทโฟน Android

ซอร์สโค้ดโปรแกรมบนสมาร์ทโฟน Android

ส่วนของซอร์สโค้ดโปรแกรมบนสมาร์ทโฟน Android มีโครงสร้าง ดังนี้



รูปที่ ก.1 โครงสร้าง Code ของระบบ

1. MainActivity.java หน้าจอหลักในการทำงานและแสดงแผนที่ที่ทำการค้นหาและเส้นทางที่สร้างขึ้น

```
import android.content.Intent;
import android.content.pm.PackageManager;
import android.location.Location;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.v4.app.ActivityCompat;
import android.support.v4.app.FragmentActivity;
```

```
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.ImageButton;
import android.widget.Toast;
import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.api.GoogleApiClient;
import com.google.android.gms.location.LocationCallback;
import com.google.android.gms.location.LocationListener;
import com.google.android.gms.location.LocationRequest;
import com.google.android.gms.location.LocationServices;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;
public class MainActivity extends FragmentActivity implements
View.OnClickListener, OnMapReadyCallback {
    private GoogleApiClient googleApiClient;
    private ImageButton HistImgeBtn;
    private ImageButton SearchImgeBtn;
    private ImageButton HelpImgeBtn;
    private ImageButton CloseImgeBtn;
    private LocationRequest mLocationRequest;
    private static final long UPDATE_INTERVAL = 5000;
    private static final long FASTEST_INTERVAL = 1000;
    private GoogleMap mMapView;
    private Location location;
    private double startLat, statLng;
    private String provider;
```

```
private LatLng origin;
private double startLatADouble, startLngADouble;
private double altitude;
private float accuracy;
private float bearing;
private float speed;
private long time;
public static int REQ_GEO_CODING_SEARCH = 1;
@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);
  SetBindWidget();
  googleApiClient = new GoogleApiClient.Builder(this)
        .addApi(LocationServices.API)
        .addConnectionCallbacks(new
              GoogleApiClient.ConnectionCallbacks() {
                @Override
                public void onConnected(@Nullable Bundle bundle)
  {                  mLocationRequest = LocationRequest.create();
                  mLocationRequest
  setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
                  mLocationRequest.setInterval(UPDATE_INTERVAL);
  LocationRequest.setFastestInterval(FASTEST_INTERVAL);
                  if
  (ActivityCompat.checkSelfPermission(MainActivity.this,
  android.Manifest.permission.ACCESS_FINE_LOCATION) !=
  PackageManager.PERMISSION_GRANTED &&
  ActivityCompat.checkSelfPermission(MainActivity.this,
  android.Manifest.permission.ACCESS_COARSE_LOCATION) !=
  PackageManager.PERMISSION_GRANTED) {
```

```
                    return;         }
LocationServices.FusedLocationApi.requestLocationUpdates(google
ApiClient, mLocationRequest, new LocationListener() {
                    @Override
                    public void onLocationChanged(Location
location) {
                        startLat = location.getLatitude();
                        statLng = location.getLongitude();
                        startLatADouble = startLat;
                        startLngADouble = statLng;
                        LatLng latLng = new LatLng(startLat,
statLng);
mMapView.addMarker(new
MarkerOptions().position(latLng).icon(BitmapDescriptorFactory.fromR
esource(R.drawable.pin)));

mMapView.animateCamera(CameraUpdateFactory.newLatLngZoom
(latLng, 17));
mMapView.getUiSettings().setRotateGesturesEnabled(true);
                    }   });
                } //onConnected
                @Override
                public void onConnectionSuspended(int i) {
                    Toast.makeText(getApplicationContext(),
"Connection is onConnectionSuspended!",
Toast.LENGTH_LONG).show();
                }   })
            .addOnConnectionFailedListener(new
GoogleApiClient.OnConnectionFailedListener() {
                @Override
                public void onConnectionFailed(@NonNull
```

```
ConnectionResult connectionResult) {
                    Toast.makeText(getApplicationContext(),
"Connection is onConnectionFailed!", Toast.LENGTH_LONG).show();
                }    }).build();   } //onCreate
    @Override
    protected void onStart() {
        super.onStart();
        googleApiClient.connect();
    }
    @Override
    protected void onStop() {
        super.onStop();
        googleApiClient.disconnect();
    }
    private void SetBindWidget() {
        HistImgeBtn = (ImageButton) findViewById(R.id.HistImgeBtn);
        SearchImgeBtn = (ImageButton)
findViewById(R.id.SearchImgeBtn);
        HelpImgeBtn = (ImageButton) findViewById(R.id.HelpImgeBtn);
        CloseImgeBtn = (ImageButton)
findViewById(R.id.CloseImgeBtn);
        HistImgeBtn.setOnClickListener(this);
        SearchImgeBtn.setOnClickListener(this);
        HelpImgeBtn.setOnClickListener(this);
        CloseImgeBtn.setOnClickListener(this);
        SupportMapFragment MainMapfragment =
(SupportMapFragment)
getSupportFragmentManager().findFragmentById(R.id.MainMapfragm
ent);
        MainMapfragment.getMapAsync(this);   } //SetBindWidget
    @Override
```

```
    public void onClick(View v) {
        Integer i = v.getId();
        Intent intent;
        if (i == R.id.HistImgeBtn) {
            intent = new Intent(getApplicationContext(),
HistoryActivity.class);
            startActivity(intent);
        } else if (i == R.id.SearchImgeBtn) {
            Toast.makeText(getApplicationContext(),
"MapSearchActivity LAT : " + startLatADouble + "  START LNG : " +
startLngADouble, Toast.LENGTH_LONG).show();
            intent = new Intent(getApplicationContext(),
MapSearchActivity.class);
            intent.putExtra("LAT", startLatADouble);
            intent.putExtra("LNG", startLngADouble);
            startActivity(intent);
        } else if (i == R.id.HelpImgeBtn) {
            intent = new Intent(getApplicationContext(),
HelpActivity.class);
            startActivity(intent);
        } else if (i == R.id.CloseImgeBtn) {
            System.exit(0);
        }    } //onClick
    @Overrde
    public void onMapReady(GoogleMap googleMap) {
        mMapView = googleMap;
        mMapView.setMapType(GoogleMap.MAP_TYPE_HYBRID);
        mMapView.setTrafficEnabled(true);
        mMapView.setBuildingsEnabled(true);
        mMapView.setIndoorEnabled(true);
        if (ActivityCompat.checkSelfPermission(this,
```

```
android.Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(this,
android.Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
        return;        }
    mMapView.setMyLocationEnabled(true);
    mMapView.getUiSettings().setZoomControlsEnabled(true);
} //onMapReady}
```

1. MapSearchAcitivity.java ซอร์สโค้ดที่ทำการค้นหาข้อมูลสถานที่ตั้งพิกัดทางภูมิศาสต์ส่งไปยังหน้า MainActivity.java ก่อนนำไปสร้างเส้นทางบนสมาร์ทโฟนบนหน้าจอของโปรแกรม

```
import android.app.ListActivity;
import android.content.Context;
import android.content.Intent;
import android.location.Address;
import android.location.Criteria;
import android.location.Geocoder;
import android.location.Location;
import android.location.LocationManager;
import android.os.AsyncTask;
import android.os.Handler;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.text.Html;
import android.util.Log;
import android.view.LayoutInflater;
```

```
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.BaseAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;
import com.akexorcist.googledirection.DirectionCallback;
import com.akexorcist.googledirection.GoogleDirection;
import com.akexorcist.googledirection.constant.TransportMode;
import com.akexorcist.googledirection.model.Direction;
import com.akexorcist.googledirection.model.Info;
import com.akexorcist.googledirection.model.Leg;
import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.api.GoogleApiClient;
import com.google.android.gms.location.LocationListener;
import com.google.android.gms.location.LocationServices;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;
import java.io.IOException;
import java.nio.DoubleBuffer;
import java.util.List;
import java.util.Locale;
public class MapSearchActivity extends AppCompatActivity implements
View.OnClickListener {
```

```
 private Button BtnSearch, BtnBack;
private EditText StartEditText, DestinationEditText;
private String mStartAddress;
 private String mDestinationAddress;
 private String iServerKey =
"AIzaSyB5QerS1CewwM0fq4mmolEbNAfSzYnIQhE";
 private List<Address> addresses;
  private String[] locationName;
 private GoogleMap mMapView;
   private LatLng latLng;
   private MarkerOptions markerOptions;
   private List<Address> addressList;
   private Double mStartLat, mStartLng, mDestLat, mDestLng;
   private ListView listRouteLatLng;
   private Button BtnCnt;
   private LatLng mStartLatLng;
   private LatLng mDestLatLng;
   private Leg leg;
   private String _destDistance;
   private String _destDuration;
   private LocationManager locationManager;
   private Criteria criteria;
   private MyManage myManage;
   private int addressRowCnt;
   private EditText DurationEditText;
   private GoogleApiClient googleApiClient;
   private Integer mReRoute;
   private String mDistanceDuration;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_map_search);
        Toast.makeText(getApplicationContext(), "MapSearchActivity LAT : ",
Toast.LENGTH_LONG).show();
        LocationListener listener = new LocationListener() {
            @Override
            public void onLocationChanged(Location location) {
            } } ;
        Intent i = getIntent();
        mStartLat = i.getDoubleExtra("LAT", 0);
        mStartLng = i.getDoubleExtra("LNG", 0);
        mStartLatLng = new LatLng(mStartLat, mStartLng);
       DurationEditText = (EditText) findViewById(R.id.DurationEditText);
        listRouteLatLng = (ListView) findViewById(R.id.list) ;
        DestinationEditText = (EditText)
findViewById(R.id.DestinationEditText);
        BtnSearch = (Button) findViewById(R.id.BtnSearch);
        BtnBack = (Button) findViewById(R.id.BtnBack);
        BtnSearch.setOnClickListener(this);
        BtnBack.setOnClickListener(this);
        listRouteLatLng.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int
selectedIndex, long id) { int maxAddressLines =
addressList.get(selectedIndex).getMaxAddressLineIndex();
                String addressTitleAndDescription = "";
                for (int i = 0; i < maxAddressLines; i++) {
                    addressTitleAndDescription = addressTitleAndDescription +
addressList.get(selectedIndex).getAddressLine(i) + "\n";                }
                double foundLat = addressList.get(selectedIndex).getLatitude();
                double foundLng =
```

```
addressList.get(selectedIndex).getLongitude();
                mReRoute =
Integer.valueOf(((DurationEditText.getText().toString())));
                Intent intent = new Intent(getApplicationContext(),
MapRouteRootActivity.class);
                intent.putExtra("START_LAT", mStartLat);
                intent.putExtra("START_LNG", mStartLng);
                intent.putExtra("END_LAT", foundLat);
                intent.putExtra("END_LNG", foundLng);
                intent.putExtra("TITLE", addressTitleAndDescription);
                intent.putExtra("REROUTE",mReRoute);
                intent.putExtra("DESTINATION_NAME",
DestinationEditText.getText().toString());
                startActivity(intent);
            }     });
    } //onCreate
    @Override
    public void onClick(View v) {
        if (v.getId()==R.id.BtnBack) {
            finish();
        } else if (v.getId()==R.id.BtnSearch){
            runGeoCodingSearch();         }    }
    private void runGeoCodingSearch() {
        final String dest = DestinationEditText.getText().toString();
        final Handler handler = new Handler();
        new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    Geocoder geoCoder = new
Geocoder(getApplicationContext(),new Locale("TH","TH"));
```

```
                addressList
=geoCoder.getFromLocationName(DestinationEditText.getText().toString(),3
0);                handler.post(new Runnable() {
                    @Override
                    public void run() {
                        listRouteLatLng.setAdapter(new
EfficientAdapter(getApplicationContext()));
                    } });          }catch (Exception e){   }     }
        }).start();    }
        public class EfficientAdapter extends BaseAdapter {
        private LayoutInflater mInflater;
        private ViewHolder holder;
        public EfficientAdapter(Context context){
            mInflater = LayoutInflater.from(context);
        }
        @Override
        public int getCount() {
            return addressList.size();
        }
        @Override
        public Object getItem(int position) {
            return null;
        }
        @Override
        public long getItemId(int position) {
            return 0;
        }
        @Override
        public View getView(int position, View convertView, ViewGroup
parent) {
            ViewHolder holder = null;
```

```
            if (convertView == null){
                convertView = mInflater.inflate(R.layout.address_list_item,
null);
                holder = new ViewHolder();
                holder.titleTextView = (TextView)
convertView.findViewById(R.id.addressTitleTextView);
                convertView.setTag(holder);
            }else{
                holder = (ViewHolder) convertView.getTag();
            }
            String _addressTitle = addressList.get(position).getAddressLine(0);
            String _addressLat =
String.valueOf(addressList.get(position).getLatitude());
            String _addressLng =
String.valueOf(addressList.get(position).getLongitude());
            Double _DestaddressLat = addressList.get(position).getLatitude();
            Double _DestaddressLng =
addressList.get(position).getLongitude();
            mDestLatLng = new LatLng(_DestaddressLat,_DestaddressLng);
            mDistanceDuration =
getDistanceDuration(mStartLatLng,mDestLatLng);
            holder.titleTextView.setText(Html.fromHtml("<b><font
color='#FEAA3F'>"
                    + String.valueOf(position + 1) + ". " + _addressTitle
                    + "  </font></b><br/>" + _addressLat + ","  + _addressLng//
                    + "  </font></b><br/>" + _destDistance + "     (Distance) ,"
+ _destDuration+"      (Time)"
            ));
            return convertView;
        } //getView
        } //EfficientAdapter
```

```
    public class ViewHolder{
        TextView titleTextView;
    } //ViewHolder
    private String getDistanceDuration(LatLng mStartLatLng, LatLng
mDestLatLng) {
        return null;    }
```

apRouteRootActivity.jva หน้าจอแสดงแผนที่เส้นทาง

```
import android.content.Intent;
import android.graphics.Color;
import android.location.Address;
import android.location.Geocoder;
import android.location.Location;
import android.os.AsyncTask;
import android.os.Bundle;
import android.support.v4.app.FragmentActivity;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.RadioButton;
import android.widget.Toast;
import com.akexorcist.googledirection.DirectionCallback;
import com.akexorcist.googledirection.GoogleDirection;
import com.akexorcist.googledirection.constant.TransportMode;
import com.akexorcist.googledirection.model.Direction;
import com.google.android.gms.location.LocationListener;
import com.google.android.gms.location.LocationRequest;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
```

```
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.android.gms.maps.model.PolylineOptions;
import org.json.JSONObject;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
public class MapRouteRootActivity extends FragmentActivity implements
OnMapReadyCallback, DirectionCallback {
    // Latitude & Longitude
    private Double Latitude ;// = 13.855467;
    private Double Longitude ;// = 100.5421912;
    // Google Map
    private GoogleMap googleMap;
    // RadioButton
    RadioButton rdoNormal, rdoHybrid, rdoSatellite, rdoTerrain;
    LatLng latLng;
    MarkerOptions markerOptions;
private double mStartLat;
private double mStartLng;
private LatLng mStartLatLng;
privatedoublemEndLat;
privatedoublemEndLng;
```

```
privateLatLngmENDLatLng;
privateStringmTitle;
privateIntegermReRoute;
privateLatLngorigin;
privatedoubleendLatADouble,endLngADouble;
privateLatLngdestination;
privateStringiServerKey="AIzaSyB5QerS1CewwM0fq4mmolEbNAfSzYnIQhE";
privateLatLngdest;
privateIntegerUPDATE_INTERVAL;
privatestaticfinallongFASTEST_INTERVAL=1000;
privateLocationListenermyLocation;
protectedvoidonCreate(BundlesavedInstanceState){
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_map_route_root);
Intenti=getIntent();
mTitle=i.getStringExtra("TITLE");
mReRoute=i.getIntExtra("REROUTE",0);
UPDATE_INTERVAL=mReRoute*60*1000;
mStartLat=i.getDoubleExtra("START_LAT",0);
mStartLng=i.getDoubleExtra("START_LNG",0);
mStartLatLng=newLatLng(mStartLat,mStartLng);
mEndLat=i.getDoubleExtra("END_LAT",0);
mEndLng=i.getDoubleExtra("END_LNG",0);
mENDLatLng=newLatLng(mEndLat,mEndLng);
Toast.makeText(getApplicationContext(),"AddPolyLineSTART"+mStartLatLn
g+"END:"+mENDLatLng,Toast.LENGTH_LONG).show();
SupportMapFragmentmMap=(SupportMapFragment)getSupportFragmentM
anager().findFragmentById(R.id.MapRoutefragment);
mMap.getMapAsync(this);
finalEditTexttxtKeyword=(EditText)findViewById(R.id.txtKeyword);
txtKeyword.setText(mTitle);
```

```
//***ButtonSearch
ImageButtonbtnSearch=(ImageButton)findViewById(R.id.imgBtnSearch);
btnSearch.setOnClickListener(newView.OnClickListener(){
publicvoidonClick(Viewv){
newGeocoderTask().execute(txtKeyword.getText().toString());}});
LocationRequestmLocationRequest=newLocationRequest();
mLocationRequest
.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
mLocationRequest.setInterval(UPDATE_INTERVAL);
mLocationRequest.setFastestInterval(FASTEST_INTERVAL);
CheckInterval();}
privatevoidSetOnMarker(){
googleMap.clear();
createMarkerUser();//refreshmarker
createMakerDestination(mENDLatLng);
googleMap.animateCamera(CameraUpdateFactory.newLatLngZoom(mEND
LatLng,17));
googleMap.getUiSettings().setRotateGesturesEnabled(true);}
publicvoidrequestDirection(){
GoogleDirection.withServerKey(iServerKey)
.from(mStartLatLng)
.to(mENDLatLng)
.alternativeRoute(true)
.execute(this);}
privatevoidcreateMakerDestination(LatLngmENDLatLng){
googleMap.addMarker(newMarkerOptions()
.position(latLng).icon(BitmapDescriptorFactory.defaultMarker(BitmapDescri
ptorFactory.HUE_ORANGE)));;
endLatADouble=latLng.latitude;
endLngADouble=latLng.longitude;
destination=newLatLng(endLatADouble,endLngADouble);}
```

```
privatevoidcreateMarkerUser(){origin=newLatLng(mStartLat,mStartLng);
googleMap.addMarker(newMarkerOptions().position(origin));}
@Override
publicvoidonMapReady(GoogleMapgoogleMap){
this.googleMap=googleMap;
googleMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);
googleMap.setTrafficEnabled(true);
googleMap.setBuildingsEnabled(true);
googleMap.setIndoorEnabled(true);
LatLngcoordinate=newLatLng(mStartLat,mStartLng);
//***Focus&Zoom
googleMap.setMapType(GoogleMap.MAP_TYPE_NORMAL);
googleMap.animateCamera(CameraUpdateFactory.newLatLngZoom(coordi
nate,10));
//***ZoomControl
googleMap.getUiSettings().setRotateGesturesEnabled(true);
googleMap.getUiSettings().setZoomControlsEnabled(true);
origin=newLatLng(mStartLat,mStartLng);
googleMap.addMarker(newMarkerOptions()
.position(mStartLatLng).icon(BitmapDescriptorFactory.defaultMarker(Bitma
pDescriptorFactory.HUE_RED)));
dest=newLatLng(mEndLat,mEndLng);
googleMap.addMarker(newMarkerOptions()
.position(dest)
.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE
_YELLOW)));
Stringurl=getDirectionsUrl(origin,dest);
DownloadTaskdownloadTask=newDownloadTask();
downloadTask.execute(url);}
privatevoidCheckInterval(){LocationListenerlistener=newLocationListener(){
@Override
```

```
publicvoidonLocationChanged(Locationlocation){
Stringprovider=location.getProvider();
doublelatitude=location.getLatitude();
doublelongitude=location.getLongitude();
doublealtitude=location.getAltitude();
floataccuracy=location.getAccuracy();
floatbearing=location.getBearing();
floatspeed=location.getSpeed();
longtime=location.getTime();
mStartLat=latitude;
mStartLng=longitude;}};}
privateclassDownloadTaskextendsAsyncTask<String,Void,String>{

@Override
protectedvoidonPreExecute(){
super.onPreExecute();}
@Override
protectedStringdoInBackground(String...params){
Stringdata="";
try{data=downloadUrl(params[0]);
Log.i("codemobiles",data);
}catch(Exceptione){}
returndata;}
@Override
protectedvoidonPostExecute(Strings){
super.onPostExecute(s);
newParserTask().execute(s);
}}//DownloadTask

privateclassParserTaskextendsAsyncTask<String,Void,List<List<HashMap<St
ring,String>>>>{
```

```
@Override
protectedList<List<HashMap<String,String>>>doInBackground(String...jsonD
ata){JSONObjectjObject;
List<List<HashMap<String,String>>>routes=null;
try{jObject=newJSONObject(jsonData[0]);
DirectionsJSONParserparser=newDirectionsJSONParser();
routes=parser.parse(jObject);
}catch(Exceptione){e.printStackTrace();}
returnroutes;}
@Override
protectedvoidonPostExecute(List<List<HashMap<String,String>>>result){
super.onPostExecute(result);
ArrayList<LatLng>points=null;
PolylineOptionslineOptions=null;
MarkerOptionsmarkerOptions=newMarkerOptions();
for(inti=0;i<result.size();i++){
points=newArrayList<LatLng>();
lineOptions=newPolylineOptions();
List<HashMap<String,String>>path=result.get(i);
for(intj=0;j<path.size();j++){
HashMap<String,String>point=path.get(j);
doublelat=Double.parseDouble(point.get("lat"));
doublelng=Double.parseDouble(point.get("lng"));
LatLngposition=newLatLng(lat,lng);
point.add(position);}
//AddingallthepointsintheroutetoLineOptions
lineOptions.addAll(points);
lineOptions.width(8);
lineOptions.color(Color.parseColor("#CC181E"));}
try{googleMap.addPolyline(lineOptions);
}catch(Exceptione){
```

```
Toast.makeText(getApplicationContext(),"Can'tgetroutedirection",Toast.LEN
GTH_SHORT).show();
}}}//ParserTask
privateStringdownloadUrl(StringstrUrl)throwsIOException{
Stringdata="";
InputStreamiStream=null;
HttpURLConnectionurlConnection=null;
try{
URLurl=newURL(strUrl);
urlConnection=(HttpURLConnection)url.openConnection();
urlConnection.connect();
iStream=urlConnection.getInputStream();
BufferedReaderbr=newBufferedReader(newInputStreamReader(iStream));
StringBuffersb=newStringBuffer();
Stringline="";
while((line=br.readLine())!=null){
sb.append(line);}
data=sb.toString();
br.close();
}catch(Exceptione){
Log.d("Exceptionwhiledownloadingurl",e.toString());
}finally{
iStream.close();
urlConnection.disconnect();
}
returndata;}//downloadUrl
privateStriggetDirectionsUrl(LatLngorigin,LatLngdest){
//Originofroute
Stringstr_origin="origin="+origin.latitude+","+origin.longitude;
//Destinationofroute
Stringstr_dest="destination="+dest.latitude+","+dest.longitude;
```

```
//Sensorenabled
Stringsensor="sensor=false";
//Buildingtheparameterstothewebservice
Stringparameters=str_origin+"&"+str_dest+"&"+sensor;
//Outputformat
Stringoutput="json";
//Buildingtheurltothewebservice
Stringurl="https://maps.googleapis.com/maps/api/directions/"+output+"?"+
parameters;
Log.i("codemobiles","webservicerequesturl:"+url);
returnurl;

}//getDirectionsUrl
@Override
publicvoidonDirectionSuccess(Directiondirection,StringrawBody){}
@Override
publicvoidonDirectionFailure(Throwablet){}
//***AnAsyncTaskBackgroundProcess
privateclassGeocoderTaskextendsAsyncTask<String,Void,List<Address>>{
@Override
protectedList<Address>doInBackground(String...locationName){
Geocodergeocoder=newGeocoder(getBaseContext());
List<Address>addresses=null;
try{addresses=geocoder.getFromLocationName(locationName[0],30);
}catch(IOExceptione){e.printStackTrace();}
returnaddresses;}
@Override
protectedvoidonPostExecute(List<Address>addresses){
if(addresses==null||addresses.size()==0){
Toast.makeText(getBaseContext(),"NoLocationfound",Toast.LENGTH_SHORT
).show();}
```

```
googleMap.clear();
origin=newLatLng(mStartLat,mStartLng);
googleMap.addMarker(newMarkerOptions()
.position(mStartLatLng)
.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE
_RED)));

for(inti=0;i<addresses.size();i++){
Addressaddress=(Address)addresses.get(i);
//CreatinganinstanceofGeoPoint,todisplayinGoogleMap
latLng=newLatLng(address.getLatitude(),address.getLongitude());
StringaddressText=String.format("%s,%s",
address.getMaxAddressLineIndex()>0?address.getAddressLine(0):"",
address.getCountryName());
markerOptions=newMarkerOptions();
markerOptions.position(latLng);
markerOptions.title(addressText);
markerOptions.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescript
orFactory.HUE_ORANGE));
googleMap.addMarker(markerOptions);
Stringurl=getDirectionsUrl(origin,latLng);
DownloadTaskdownloadTask=newDownloadTask();
downloadTask.execute(url);
if(i==0)
{googleMap.animateCamera(CameraUpdateFactory.newLatLng(latLng));
}}}}}
```

2. DirectionJSONParser.java

```
packagecom.example.itm801.itm801routetrafficplan;
importcom.google.android.gms.maps.model.LatLng;
importorg.json.JSONArray;
importorg.json.JSONException;
```

```
importorg.json.JSONObject;
importjava.util.ArrayList;
importjava.util.HashMap;
importjava.util.List;
publicclassDirectionsJSONParser{
/**ReceivesaJSONObjectandreturnsalistoflistscontaininglatitudeandlongitu
de*/
publicList<List<HashMap<String,String>>>parse(JSONObjectjObject){
List<List<HashMap<String,String>>>routes=newArrayList<List<HashMap<St
ring,String>>>();
JSONArrayjRoutes=null;
JSONArrayjLegs=null;
JSONArrayjSteps=null;
JSONArrayjDistance=null;
JSONArrayjDuration=null;
try{
jRoutes=jObject.getJSONArray("routes");
/**Traversingallroutes*/
for(inti=0;i<jRoutes.length();i++){
jLegs=((JSONObject)jRoutes.get(i)).getJSONArray("legs");
//jDistance=(((JSONObject)jRoutes.get(i)).getJSONArray("legs")).getJSONArray
("distance");
Listpath=newArrayList<HashMap<String,String>>();
/**Traversingalllegs*/
for(intj=0;j<jLegs.length();j++){
jSteps=((JSONObject)jLegs.get(j)).getJSONArray("steps");
/**Traversingallsteps*/
for(intk=0;k<jSteps.length();k++){
Stringpolyline="";
polyline=(String)((JSONObject)((JSONObject)jSteps.get(k)).get("polyline")).get
("points");
```

```
List<LatLng>list=decodePoly(polyline);
for(intl=0;l<list.size();l++){
HashMap<String,String>hm=newHashMap<String,String>();
hm.put("lat",Double.toString(((LatLng)list.get(l)).latitude));
hm.put("lng",Double.toString(((LatLng)list.get(l)).longitude));path.add(hm);
} }
routes.add(path);}}
}catch(JSONExceptione){
e.printStackTrace();
}catch(Exceptione){}
returnroutes;}
privateList<LatLng>decodePoly(Stringencoded){
List<LatLng>poly=newArrayList<LatLng>();
intindex=0,len=encoded.length();
intlat=0,lng=0;
while(index<len){intb,shift=0,result=0;
do{
b=encoded.charAt(index++)-63;
result|=(b&0x1f)<<shift;
shift+=5;
}while(b>=0x20);
intdlat=((result&1)!=0?~(result>>1):(result>>1));
lat+=dlat;
shift=0;
result=0;
do{b=encoded.charAt(index++)-63;
result|=(b&0x1f)<<shift;
shift+=5;
}while(b>=0x20);
intdlng=((result&1)!=0?~(result>>1):(result>>1));
lng+=dlng;
```

```
LatLngp=newLatLng((((double)lat/1E5)),
(((double)lng/1E5)));
poly.add(p);}
returnpoly;}}
```