ระบบแจ้งเตือนอัคคีภัยในครัวเรือนด้วยเทคโนโลยี IoT6

Home Fire Alarm System with IoT6

นายศุภโชค เกียรติกิติกูล

โครงงานนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
ปริญญาวิทยาศาสตรบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
คณะเทคโนโลยีสารสนเทศ
สถาบันเทคโนโลยี ไทย-ญี่ปุ่น
พ.ศ. 2559

ระบบแจ้งเตือนอัคคีภัยในครัวเรือนด้วยเทคโนโลยี IoT6

Home Fire Alarm System with IoT6

นายศุภโชค เกียรติกิติกูล

โครงงานนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
วิทยาศาสตรบัณฑิต สาขาเทคโนโลยีสารสนเทศ
คณะเทคโนโลยีสารสนเทศ
สถาบันเทคโนโลยีไทย - ญี่ปุ่น
ปีการศึกษา 2559

คณะกรรมการสอบ

..... ประธานกรรมการสอบ
(อาจารย์ ดร.ภาสกร อภิรักษ์วรพินิต)

..... กรรมการสอบ
(อาจารย์ อมรพันธ์ ชมกลิ่น)

..... อาจารย์ที่ปรึกษา
(ผู้ช่วยศาสตราจารย์ ดร.อรรณพ หมั่นสกุล)

..... ประธานโครงงาน
(อาจารย์ อมรพันธ์ ชมกลิ่น)

ชื่อโครงงาน/รายงาน	ระบบแจ้งเตือนอัคคีภัยในครัวเรือนด้วยเทคโนโลยี IoT6 Home Fire Alarm System with IoT6
ผู้เขียน	นายศุภโชค เกียรติกิติกูล
คณะวิชา	เทคโนโลยีสารสนเทศ สาขาวิชา เทคโนโลยีสารสนเทศ
อาจารย์ที่ปรึกษา	ผศ.ดร อรรณพ หมั่นสกุล

งานที่ปฏิบัติ (ประกอบไปด้วย รูปแบบ วิธีการดำเนินงาน ข้อมูล และการวิเคราะห์)

นำเทคโนโลยี IoT6 ติดตั้งเซ็นเซอร์แก๊สและเซ็นเซอร์ไฟมาประยุกต์ใช้กับเตาแก๊ส เพื่อเพิ่มความปลอดภัยในครัวเรือน

โดยระบบแจ้งเตือนอัคคีภัยในครัวเรือนด้วยเทคโนโลยี IoT6 สามารถเช็คสถานะว่ามีไฟหรือแก๊สภายในห้องได้ตลอดเวลา ผ่านคลาวด์แพลตฟอร์ม Ubidots โดยเชื่อมต่อด้วยไอพีรุ่นที่หกผ่านการทำ Tunnel Broker พร้อมทั้งสามารถแจ้งเตือนเมื่อเกิดเหตุฉุกเฉินได้

การทำงานของ Ubdots นั้นสามารถทำงานได้ดี สามารถเช็คสถานะของแก๊สและไฟได้ตลอดเวลา โดยเมื่อเกิดเหตุฉุกเฉินจะทำการแจ้งเตือนด้วยอีเมลหรือข้อความมายังผู้ใช้งาน

ผลที่ได้รับจากการดำเนินงานและประโยชน์ที่ได้รับ

สามารถตรวจเช็คสถานะไฟที่เตาแก๊สได้ตลอดเวลาและหากเกิดเหตุการณ์แก๊สรั่วสามารถแจ้งเตือนได้ ซึ่งช่วยเพิ่มความปลอดภัยได้มากยิ่งขึ้น

กิตติกรรมประกาศ

รายงานเล่มนี้สำเร็จลุล่วงไปได้ด้วยดีด้วยความกรุณาจาก ผศ.ดร อรรณพ หมั่นสกุล ที่ได้เขียนโครงการเพื่อขอทุนการศึกษาสำหรับการซื้ออุปกรณ์ Z1 มาให้ทดลองสำหรับค้นคว้าเพื่อใช้ในงานวิจัย คอยให้คำปรึกษา เสนอแนวคิดต่างๆ ช่วยเหลือแก้ไขข้อบกพร่อง และความช่วยเหลือในทุกๆด้าน ซึ่งเป็นประโยชน์ต่องานวิจัยอย่างยิ่ง และขอแสดงความขอบคุณต่อคณะเทคโนโลยีสารสนเทศ สถาบันเทคโนโลยีไทย-ญี่ปุ่นที่เอื้อเฟื้อสถานที่เพื่อใช้ในการวิจัยและทดลองใช้งานอุปกรณ์จนรายงานนี้สำเร็จลุล่วง

สุดท้ายนี้ขอขอบคุณบิดา มารดา ที่สนับสนุนค่าเล่าเรียน อาจารย์ทุกๆท่านและเพื่อนทุกๆคนที่คอยให้ความช่วยเหลือและให้กำลังใจจนสำเร็จการศึกษา

นายศุภโชค เกียรติกิติกูล

ผู้จัดทำ

สารบัญ

สารบัญรูป

รูป **หน้า**

สารบัญตาราง

ตาราง **หน้า**

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบันเทคโนโลยีสารสนเทศกลายมาเป็นส่วนสำคัญในการดำเนินชีวิตของมนุษย์ ซึ่งการพัฒนาของเทคโนโลยีสารสนเทศ ก่อให้เกิดนวัตกรรมใหม่ๆตามมากมาย ซึ่งส่งผลให้ชีวิตประจำวันของมนุษย์เปลี่ยนไป มีความสะดวกสบายและความปลอดภัยที่มากยิ่งขึ้นบุคคลที่อยู่ไกลกันสามารถสื่อสารกันได้และมนุษย์เองสามารถเข้าถึงเทคโนโลยีได้ง่ายขึ้น

เทคโนโลยี IoT6 เป็นอีกเทคโนโลยี ที่มีความทันสมัยและน่าสนใจเพราะในอุปกรณ์ที่มีขนาดเล็กแต่สามารถเชื่อมต่อกับเครือข่ายอินเทอร์เน็ตบนไอพีรุ่นที่หกและส่งข้อมูลที่ได้รับการตรวจวัดจากเซ็นเซอร์ต่างๆมากมายหลายชนิด อัพโหลดขึ้นไปบนอินเทอร์เน็ตได้หลากหลายแพลตฟอร์มโดยสามารถเข้าถึงข้อมูลที่ได้ตรวจวัดได้ทุกที่ ทุกเวลา

ในประเทศไทยครัวเรือนตามบ้านหรือตามร้านอาหารต่างๆ มีการใช้งานแก๊สหุงต้มอยู่มาก ซึ่งหากใช้งานอย่างไม่ระมัดจะวังหรือไม่รอบคอบ จะมีโอกาสที่จะก่อให้เกิดอุบัติเหตุได้บางครั้งอาจลุกลามเป็นไฟไหม้ได้ ซึ่งเป็นอันตรายต่อความปลอดภัยและทรัพย์สิน ทำให้เกิดความคิดที่จะนำอุปกรณ์ Z1 ที่ติดตั้งเซ็นเซอร์ตรวจวัดแก๊สและไฟ เพื่อช่วยคอยดูแลความปลอดภัยของครัวเรือนภายในบ้าน

1.2 วัตถุประสงค์การศึกษา

(1) เพื่อศึกษาการทำงานของ IoT6 (นิยาม)

(2) เพื่อศึกษาอุปกรณ์ที่เหมาะสม เพื่อนำมาใช้เพิ่มความปลอดภัยในครัวเรือนด้วย IoT6 (อุปกรณ์)

(3) เพื่อประยุกต์การใช้เทคโนโลยีสำหรับความปลอดภัยในครัวเรือน

1.3 ขอบเขตของการศึกษา

(1) สามารถรับค่าจากเซ็นเซอร์ และส่งต่อไปยัง Border Router ได้

(2) ใช้ไอพีรุ่นที่หกในการส่งข้อมูลจากเซ็นเซอร์ถึงคลาวด์แพลตฟอร์ม

(3) สามารถแจ้งเตือนผ่านอีเมลและข้อความแจ้งเตือนผู้ใช้งานได้

1.4 นิยามศัพท์เฉพาะ

IoT (Internet of Things) คือ อุปกรณ์ขนาดเล็กที่มีความชาญฉลาดสามารถเชื่อมต่อกับเครือข่าย สามารถส่งข้อมูลที่มีประโยชน์ ซึ่งเมื่อนำไปใช้งานในจำนวนที่มากสามารถนำไปใช้งานให้เกิดประโยชน์หลากหลาย เช่น บ้านอัจฉริยะ อุปกรณ์สวมใส่อัจฉริยะ เมืองอัจฉริยะ ใช้ในอุตสาหกรรมต่างๆ และอื่นๆอีกมากมาย โดยในงานนี้จะเน้นศึกษาบน IoT6 (Internet of Things with IPv6) ซึ่งหมายถึงอุปกรณ์ IoT ที่ทำงานบนอินเทอร์เน็ตโพรโตคอลรุ่นที่หก ซึ่งในสองเทคโนโลยีนี้สามารถนำพาข้อดีของทั้งสองเทคโนโลยีมาไว้ด้วยกัน

Ubidots คือ คลาวด์แพลตฟอร์มที่ใช้สำหรับการรับข้อมูลจากเซ็นเซอร์มาเพื่อแสดงผลให้ผู้ใช้งานสามารถเข้ามาตรวจสอบและมีคุณสมบัติที่จะแจ้งเตือนไปยังผู้ใช้งานได้

Z1 คือ อุปกรณ์ IoT ที่รองรับการทำงานของไอพีรุ่นที่หก สามารถเชื่อมต่อกับเซ็นเซอร์ต่างๆ และทำงานบนระบบปฏิบัติการ Contiki ได้ ซึ่งเป็นอุปกรณ์หลักที่ใช้ในงานนี้

1.5 ประโยชน์ที่คาดว่าจะได้รับ

(1) สามารถนิยาม ทฤษฎี และเทคโนโลยี IoT6 และประยุกต์ใช้

(2) สามารถเลือกใช้อุปกรณ์ที่เหมาะสมสำหรับเทคโนโลยี IoT6

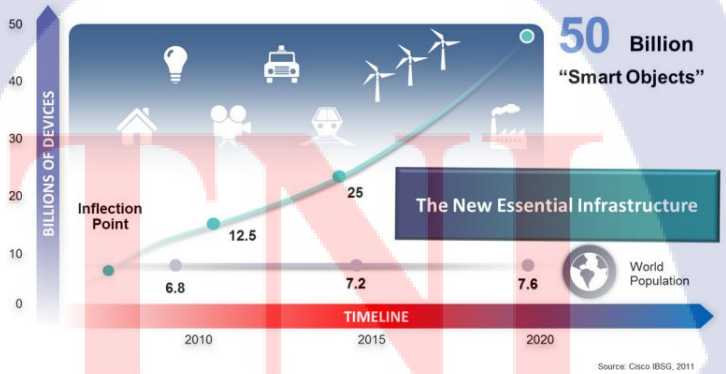
(3) สามารถสร้างระบบการแจ้งเตือนเพื่อเพิ่มความปลอดภัยในครัวเรือนได้

บทที่ 2
ทฤษฎีและเทคโนโลยีที่ใช้ในการปฏิบัติงาน

2.1 อินเทอร์เน็ตสำหรับทุกสิ่งด้วยไอพีรุ่นที่หก

อินเทอร์เน็ตสำหรับทุกสิ่งด้วยไอพีรุ่นที่หก (Internet of Things with IPv6, IoT6) สร้างขึ้นบนโครงข่ายที่ซับซ้อน เชื่อมต่ออุปกรณ์กว่าพันล้านชิ้นและมนุษย์ เพื่อเข้าใช้งานหลากหลายเทคโนโลยี หลายโพรโทคอล หลายแพลตฟอร์ม โดยจุดประสงค์หลักของ IoT คือ การสร้างโลกอัจฉริยะที่ทั้ง กายภาพ ดิจิทัล และโลกเสมือน สร้างสภาพแวดล้อมอัจฉริยะที่จะทำให้เกิดการกระตุ้น ด้านสุขภาพ การขนส่ง เมือง อุตสาหกรรม ตึกอาคาร และสิ่งต่างๆนอกเหนือจากชีวิตประจำวัน

ความคาดหวังต่อการเชื่อมต่อของโครงข่ายอัจฉริยะ เพื่อให้เข้าถึงสารสนเทศได้ไม่ใช่เพียงแค่ทุกเวลาและทุกสถานที่ แต่ยังครอบคลุมถึงทุกสิ่งและทุกคนผ่าน เส้นทาง เครือข่าย บริการต่างๆ ซึ่งจะประสบความสำเร็จโดยมีเป้าหมายที่จะจัดการชีวิตประจำวัน เครื่องมือตรวจวัด ระบุตัวตน อุปกรณ์ระบุตำแหน่ง โดยสร้างขึ้นด้วยที่อยู่ไอพีเพื่อให้กลายเป็นอุปกรณ์อัจฉริยะ ที่สามารถทำการสื่อสารได้ไม่ใช่เพียงอุปกรณ์อื่นๆแต่ยังเชื่อมต่อกับมนุษย์ได้ด้วย โดยคาดหวังที่จะเข้าไปถึงพื้นที่ที่ไม่สามารถเข้าถึงถ้าไม่มีการพัฒนาสร้างการตรวจวัด และการใช้เทคโนโลยีระบุตำแหน่ง



รูปที่ 2.1 วิวัฒนาการของการเชื่อมต่ออินเทอร์เน็ตในอนาคต (ที่มา: Cisco 2011)

2.2 เครือข่ายเซ็นเซอร์ไร้สาย

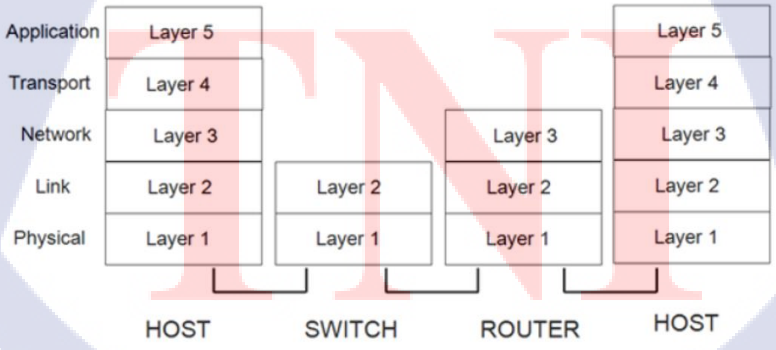
เครือข่ายเซ็นเซอร์ไร้สาย (Wireless Sensors Network, WSN) คือ โหนดเซ็นเซอร์ขนาดเล็ก สามารถตั้งค่าเครือข่ายด้วยตัวเองได้ (เรียกว่า Motes) สื่อสารกันด้วยสัญญาณวิทยุ และถูกนำไปใช้ตรวจจับได้จริง เซ็นเซอร์โหนดเป็นพื้นฐานของคอมพิวเตอร์ขนาดเล็ก ทำงานได้หลากหลาย ประกอบด้วยหน่วยประมวลผลที่มีการประมวลผลที่ถูกจำกัด และหน่วยความที่ถูกจำกัด สื่อสารผ่านอุปกรณ์วิทยุ ใช้พลังงานและเซ็นเซอร์หนึ่งอย่างหรือมากกว่านั้น

โหมด (Motes) จะมีรูปทรงและรูปร่างที่ต่างกันขึ้นอยู่กับการการใช้งานสามารถให้มีขนาดที่เล็กมาก ซึ่งหากถูกปล่อยให้ใช้งานมากจะต้องมีผลกระทบน้อย สามารถที่จะชาร์จพลังงานเพื่อใช้งานในห้องทดลอง หลากหลายอุปกรณ์ขนาดเล็กที่แตกต่างกัน ในสถานการณ์ที่มีความหลากหลายมากที่สุด เพื่อให้แน่ใจหลากหลายของการใช้งาน บางแอปพลิเคชั่นสามารถตรวจสอบ สิ่งแวดล้อม สถาปัตยกรรม สุขภาพและความปลอดภัย

2.3 ไอพีรุ่นที่หก

2.3.1 แนวคิดของไอพีรุ่นที่หก

ไอพีรุ่นที่หก คือ อินเทอร์เน็ตโพรโทคอลรุ่นล่าสุด เพื่อให้เข้าใจว่าทำไมถึงควรที่จะนำมาใช้กับ IoT หรือเกี่ยวข้องกับโพรโทคอล อื่นอย่าง 6LoWPAN โดยจะอธิบายในภายหลัง โดยไอพีรุ่นที่หกนั้นเป็นคนละโพรโทคอลทำงานร่วมกับไอพีรุ่นที่สี่ไม่ได้



รูปที่ 2.2 ชั้นของอินเทอร์เน็ตโพรโทคอล

ไอพีรุ่นที่หก จะทำงานในชั้นที่สาม คือชั้นเน็ตเวิร์ค ส่วนของข้อมูลจะถูกจัดการในชั้นที่สามจะถูกเรียกว่าแพ็กเก็ต อุปกรณ์ที่เชื่อมต่ออินเทอร์เน็ตเป็นได้ทั้งโฮสหรือเราเตอร์ โฮสจะเป็น พีซี โน้ตบุ๊ก หรือเซ็นเซอร์บอร์ดก็ได้ รับและส่งข้อมูลแพ็กเก็ต โดยโฮสจะเป็นต้นทางหรือปลายทางของแพ็กเก็ต เราเตอร์นั้นดูแลการส่งต่อข้อมูล และทำหน้าที่เลือกเราเตอร์ถัดไป เพื่อส่งข้อมูลไปให้ถึงปลายทาง เครือข่ายจะประกอบด้วยการเชื่อมต่อของเราเตอร์มากมาย ที่รับข้อมูลจากช่องทางนึงและส่งไปให้เราเตอร์ถัดไปให้เร็วที่สุดเท่าที่จะเป็นไปได้

2.3.2 แพ็กเก็ตไอพีรุ่นที่หก

ในรูปแบบระดับชั้นก่อนหน้านี้ ทุกชั้นจะมีการแนะนำข้อมูลเลเยอร์ของตัวเองภายในแพ็กเก็ต และใช้ข้อมูลเหล่านั้นประมวลผลชั้นเดียวกันบนอุปกรณ์ไอพีอุปกรณ์อื่น การสนทนาระหว่างระดับชั้นเดียวกันต่างอุปกรณ์จะต้องปฏิบัติตามโพรโทคอล

อินเทอร์เน็ตเลเยอร์มี 5 ชั้นตามลำดับ

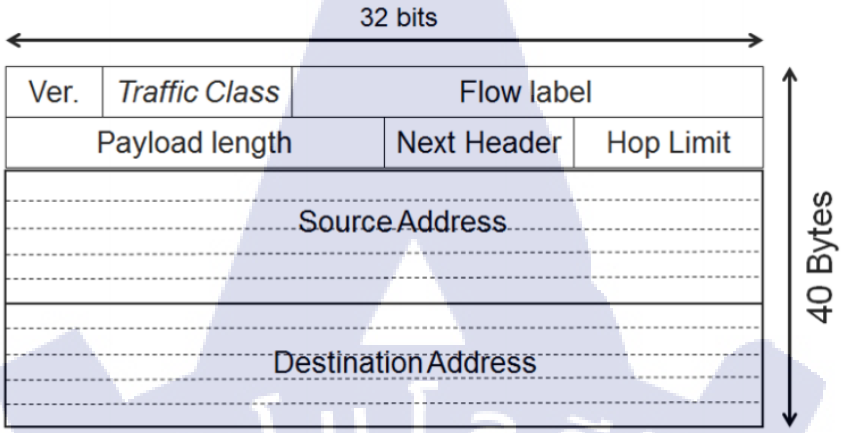
(1) Application เป็นฝั่งของซอฟแวร์ถูกพัฒนาโดยโปรแกรมเมอร์ ที่ใช้กองซ้อนเน็ตเวิร์คเพื่อใช้บริการเครือข่าย ตัวอย่างเช่นเว็บเบราว์เซอร์ที่เปิดการเชื่อมต่อเครือข่ายเพื่อเชื่อมต่อไปที่เว็บเซิร์ฟเวอร์ หรือเว็บเซิร์ฟเวอร์ซอร์ฟแวร์ที่ทำงานในเซิร์ฟเวอร์อยู่ภายในเครือข่ายเพื่อรอตอบคำร้องขอจากเว็บไคลเอนต์ ตัวอย่างแอปพลิเคชันโพรโทคอลเช่น HTTP และ DNS

(2) Transport เป็นชั้นที่ทำงานบนชั้นเน็ตเวิร์คที่คอยบริการเพิ่มเติม เช่น การส่งข้อมูลใหม่จากแพ็กเก็ตที่สูญหายหรือการรับประกันว่าข้อมูลที่ส่งกับข้อมูลที่รับจะต้องเหมือนกัน โดยชั้นนี้แสดงให้เห็นถึงการบริการเครือข่ายไปยังชั้นแอปพลิเคชัน เป็นบริการที่ส่งและรับข้อมูล TCP และ UDP เป็นโพรโทคอลที่ใช้กันทั่วไปภายในอินเทอร์เน็ต

(3) Network ระดับชั้นนี้ดูแลการจัดส่งที่ข้อมูลที่ได้รับจากชั้นการขนส่งข้อมูลไปยังปลายทางให้ถูกต้อง ตลอดจนรับข้อมูลจากชั้นของการเชื่อมต่อที่รับมาจากข้อมูลปลายทางอินเทอร์เน็ตใช้เพียงโพรโทคอลเดียวในชั้นนี้ คือไอพีโดยทั้งต้นทางและปลายทางจะถูกระบุที่อยู่โดยที่อยู่ไอพี

(4) Link ในชั้นนี้ดูแลส่วนของการส่งและรับเฟรม เก็บไบต์ที่ส่งมาจากเน็ตเวิร์คเลเยอร์ภายในขอบเขตของ LAN (Local Area Network) ที่ระบุถึงวิธีการที่จะใช้แบ่งปันการใช้งานของโหนดที่ต่างกัน ในชั้นนี้จะมีวิธีการระบุที่อยู่ของมันเอง ซึ่งจะขึ้นอยู่กับเทคโนโลยีที่ถูกใช้งาน

(5) Psysical เป็นชั้นที่ดูแลเกี่ยวกับรายละเอียดและสัญญาณไฟฟ้า การเข้ารหัสและอื่นๆ ซึ่งจำเป็นสำหรับข้อมูลที่จะเดินทางจากโหนดนึงไปอีกโหนดสื่อทางกายภาพก็รวมด้วยทั้งมีสายและไร้สาย



รูปที่ 2.3 ส่วนหัวของไอพีรุ่นที่หก

ขั้นแรกส่วนหัวของไอพีรุ่นที่หก ถูกจำกัดขนาดไว้ที่ 40 ไบต์ ตามมาด้วยข้อมูลในชั้นด้านบนและถูกปรับบางส่วนเพิ่มเข้ามาในส่วนหัวซึ่งจะอธิบายในภายหลัง อย่างที่เห็นว่าจะมีหลายพื้นที่ถูกเก็บไว้ในส่วนหัว ซึ่งถูกจัดเตรียมไว้เพิ่มขึ้นเมื่อเปรียบเทียบกับส่วนหัวของไอพีรุ่นที่สี่

(1) จำนวนของตัวเลขถูกลดจาก 12 เป็น 8

(2) ส่วนหัวของไอพีรุ่นที่หก ถูกปรับเป็น 40 ไบต์ และถูกจัดให้เป็นแนวเดียวกัน 64 บิต ซึ่งทำให้เราเตอร์ส่งข้อมูลไปได้เร็วขึ้น

(3) พื้นที่ของที่อยู่เพิ่มจาก 32 เป็น 128 บิต

ส่วนที่สำคัญที่สุดคือต้นทางและปลายทางทุกอุปกรณ์จะต้องมีที่อยู่ไอพี ที่ไม่ซ้ำกันเพื่อระบุที่อยู่บนเครือข่าย โดยใช้ไอพีเพื่อให้เราเตอร์ตัดสินใจในการส่งข้อมูล

ส่วนหัวของไอพีรุ่นที่หกมี 128 บิตสำหรับทุกๆ ไอพีรุ่นที่หก ซึ่งใช้ได้ 2^{128} หรือประมาณ 3.14 คูณด้วย 1038 หรือ 3.4 ตามด้วยศูนย์สามสิบแปดตัว ในทางตรงข้ามไอพีรุ่นที่สี่ใช้เพียง 32 บิตในการเข้ารหัสแต่ละที่อยู่ซึ่งมีให้ใช้ 2^{32} (4,294,967,296)

2.3.3 ที่อยู่ไอพี

การใช้ 128 บิตสำหรับที่อยู่ไอพีนำมาซึ่งข้อดีหลายอย่าง

(1) มีที่อยู่ที่มากตอบสนองความต้องการของอนาคตซึ่งเพียงพอสำหรับนวัตกรรมใหม่ๆ

(2) ทำให้ง่ายขึ้นเมื่ออุปกรณ์สามารถตั้งค่าเครือข่ายได้ด้วยตนเอง

(3) ง่ายต่อการมอบหมาย จัดการที่อยู่

(4) มีพื้นที่สำหรับการจัดลำดับชั้นมากขึ้นและสำหรับการรวมเส้นทาง

(5) ความสามารถ IP Security การทำงานแบบปลายทางไปยังปลายทาง

ที่อยู่ไอพีรุ่นที่หกที่มีอยู่ได้ถูกจัดกลุ่มเป็นหลายประเภทดังนี้ (บางอย่างมีในไอพีรุ่นที่สี่)

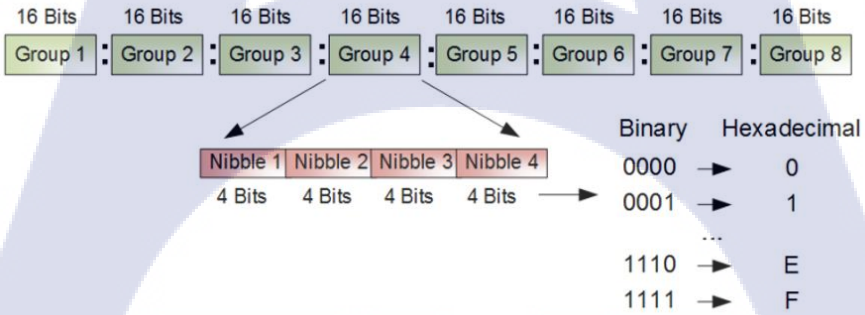
Unicast (one-to-one) ใช้ส่งข้อมูลแพ็กเก็ต จากต้นทางไปยังเป้าหมายเดี่ยว โดยเกิดขึ้นบ่อยมากที่สุด โดยจะกล่าวถึงมันและคลาสย่อยมากขึ้น

Multicast (one-to-many) ใช้ส่งข้อมูลแพ็กเก็ต จากต้นทางไปหลายเป้าหมายที่เป็นไปได้ โดย Multicast จะส่งข้อมูลเดียวกันไปหลายที่

Anycast (one-to-nearest) ใช้ส่งข้อมูลจากต้นทางไปยังกลุ่มปลายทางที่ใกล้ที่สุด

Reserved ที่อยู่หรือกลุ่มพิเศษ ตัวอย่างเช่น ที่อยู่ใช้ในเอกสารหรือตัวอย่าง

2.3.4 กฎของไอพีรุ่นที่หก



รูปที่ 2.4 ที่อยู่ไอพีรุ่นที่หก

ลักษณะของไอพีรุ่นที่หก

(1) 16 บิตจาก 8 กลุ่ม จะถูกแบ่งโดย " : " เลขฐาน 16 จากทุกๆ 4 บิต

(2) ไม่มีการแยกตัวพิมพ์เล็กและตัวพิมพ์ใหญ่

(3) กลุ่มของที่อยู่ (Network Prefix) จะถูกเขียนขนาด Prefix/Prefix Length ซึ่งขนาดของ Prefix จะถูกวัดจากจำนวนบิตของที่อยู่ที่ใช้ทั่วไปในกลุ่ม

(4) เลขศูนย์ซ้ายสุดของแต่ละกลุ่มจะถูกเอาออก

(5) กลุ่มที่มีแต่ 0 หนึ่งกลุ่มหรือมากกว่าจะถูกแทนที่ด้วย “ :: ” ซึ่งทำได้ครั้งเดียว

กฎ 3 อย่างที่บอกถึงลักษณะทั่วไปของ ที่อยู่ไอพีรุ่นที่หกคือ มีลักษณะเป็นเลขฐาน 16 ตัวเลขที่แสดงออกมามี 16 รูปแบบ ระหว่าง 0 ถึง F โดยจะมี 8 กลุ่ม จากเลขฐานสิบหก 4 ตัวอักษรทุกกลุ่มจะถูกแบ่งโดยโคลอน “ : ” อีก 2 กฎที่เหลือจะเทียบให้ดูตามด้านล่าง

ตัวอย่าง

ถ้านำเสนอรูปแบบของบิตที่อยู่จะได้รูปแบบดังตัวอย่าง

2001:0db8:4004:0010:0000:0000:6543:0ffd

ถ้าใส่เครื่องหมายก้ามปูรอบที่อยู่จะได้ที่อยู่ตามจริง

[2001:0db8:4004:0010:0000:0000:6543:0ffd]

ถ้านำกฎลำดับที่สี่มาใช้ จะอนุญาตให้ย่อรูปเลข 0 ด้านซ้ายสุดของทุกกลุ่มออก จะได้

2001:db8:4004:10:0:0:6543:ffd

ถ้านำกฎลำดับที่ห้ามาใช้จะอนุญาตให้ย่อรูปกลุ่มที่มีแต่ 0 และอยู่ติดกันโดยใช้ “ :: ” จะได้

2001:db8:4004:10::6543:ffd

ระวังเวลาใช้การย่อรูปและการขยายรูปของที่อยู่ไอพีรุ่นที่หก การประมวลผลสามารถนำกลับมาเป็นแบบเดิมได้ ซึ่งเป็นปกติที่จะเกิดความผิดพลาด ตัวอย่างเช่น 2001:db8:a:0:0:12:0:80 สามารถย่อรูปโดยการใส่ “ :: ” ได้สองทางเลือก

1) 2001:db8:a::12:0:80

2) 2001:db8:a:0:0:12::80

ทั้งคู่นั้นเป็นที่อยู่ไอพีรุ่นที่หกที่ถูกต้องแต่ที่อยู่ 2001:db8:a::12::80 นั้นผิด ซึ่งไม่ได้ทำตามกฎของการย่อรูปด้านบนปัญหาคือการย่อรูปที่ไม่ดีทำให้เราไม่แน่ใจว่าจะขยายอย่างไร ซึ่งไม่ชัดเจนสามารถขยายแล้วจะเป็น 2001:db8:a:0:12:0:0:80 หรือเป็น 2001:db8:a:0:0:12:0:80

Network prefix ของไอพีรุ่นที่หก

สุดท้ายต้องรู้เกี่ยวกับแนวคิดของ Network Prefix ซึ่งจะบอกถึงบิตที่บังคับและบิตที่ไม่ระบุ ที่สามารถสร้าง Prefix ย่อยใหม่ หรือใช้บอกถึงที่อยู่ไอพีรุ่นที่หกที่มอบให้โฮส

ดังตัวอย่าง

1. เน็ตเวิร์ค Prefix 2001:db8:1::/48(2001:0db8:0001:0000:0000:0000:0000:0000) บอกถึง 48 บิต ที่จะต้องเหมือนเดิม (2001:0db8:0001) ซึ่งสามารถใช้ 80 bits ด้วยอื่นๆได้ ตัวอย่างเช่น เพื่อจัด 2 Prefix ที่เล็กกว่า 2001:db8:1:a::/64 และ 2001:db8:1:b::/64

2. ถ้านำ 1 ใน Prefix เล็กด้านบน 2001:db8:1:b::/64 ที่ 64 บิตแรกจะถูกกำหนดไว้ พวกเราจะมี 64 บิตทางด้านขวาเพื่อกำหนด ตัวอย่างเช่น ไอพีรุ่นที่หกที่แสดงในโฮส 2001:db8:1:b:1:2:3:4 * ตัวอย่างนี้อนุญาตให้เรานำเสนอพื้นฐานของ ไอพีรุ่นที่หก * A/64 มักจะเป็น Prefix ที่ถูกใช้ใน LAN 64 บิตด้านขวา จะถูกเรียกว่า Interface Identifier (IID) เพราะว่ามีเพียงหนึ่งเดียวซึ่งสามารถระบุโฮสใน Local Network ถูกกำหนดโดย /64 จะได้ Prefix ตามรูปภาพนี้

รูปที่ 2.5 เน็ตเวิร์คไอดีและอินเทอร์เฟสไอดี

2.3.5 ใช้ไอพีรุ่นที่หกเพื่ออะไร

อย่างที่เห็นว่าไอพีรุ่นที่หกมีคุณสมบัติที่ช่วยเพิ่มความสะดวก เช่น Global Addressing และ Auto Configuration เพราะไอพีรุ่นที่หก นั้นมีที่อยู่มากมายพอให้ได้ใช้ไปอีกหลายร้อยปี พวกเราสามารถให้ Global unicast address ไอพีรุ่นที่หกกับทุกอุปกรณ์ที่พวกเราจะคิดถึงได้ ซึ่งจะนำกลับมาถึงการเริ่มต้นของรูปแบบอินเทอร์เน็ตที่ทุกๆ อุปกรณ์ไอพี สามารถสื่อสารกันได้. การเชื่อมต่อแบบปลายทางไปยังปลายทางจะอนุญาตให้เชื่อมต่อแบบสองทิศทางบนเครือข่ายระหว่างอุปกรณ์ไอพีใดๆก็ได้ ซึ่งเป็นผลลัพธ์ของแอปพลิเคชันที่ร่วมมือกัน การเก็บข้อมูลรูปแบบใหม่ๆ การส่งและเข้าถึงข้อมูล

ในเนื้อหาจะได้เห็นถึงตัวอย่างการเก็บข้อมูลเซ็นเซอร์ของไอพีรุ่นที่หกรอบโลกการส่งและการเข้าถึงข้อมูลจากสถานที่ต่างๆเพื่อสร้างเครือข่ายของการวัดข้อมูลบนพื้นที่จริง การเก็บข้อมูลและประมวลผล

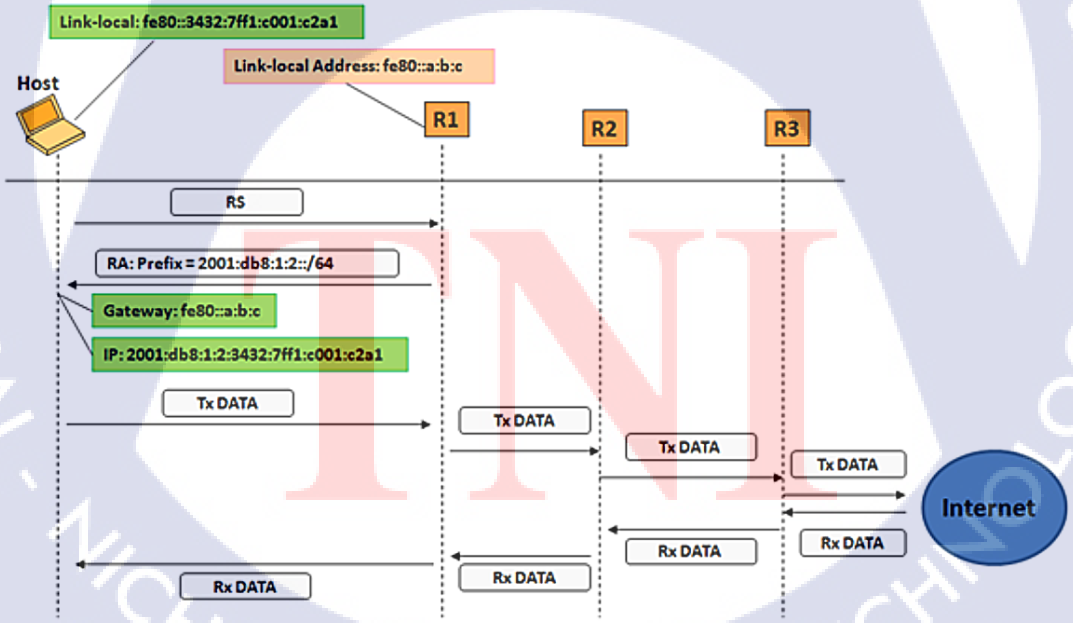
การมีอยู่ของที่อยู่จำนวนมหาศาล ทำให้เกิดวิธีใหม่ที่เรียกว่า Stateless Address Configurataion (SLAAC) ซึ่งไม่มีในไอพีรุ่นที่สี่ซึ่ง โดยจะสรุปวิธีต่างๆที่ใช้ตั้งค่าที่อยู่ของไอพีรุ่นที่หก

Statically คุณสามารถที่จะมอบที่อยู่ให้อุปกรณ์ไอพีจากนั้นตั้งค่าเองในอุปกรณ์โดยใช้วิธีใดก็ได้ เช่น เว็บ บรรทัดคำสั่งและอื่นๆ โดยปกติจะต้องตั้งค่าอื่น เช่น เกตเวย์เพื่อให้ส่งแพ็กเก็ตไปยังนอกเครือข่ายได้

DHCPv6 (Dynamic Host Configuration Protocol for IPv6) [RFC3315] นำมาจากวิธีที่คล้ายกับที่มีอยู่ในไอพีรุ่นที่สี่โดยจำเป็นจะต้องให้เครื่องแม่ข่ายที่ติดต่อเพื่อมอบที่อยู่ให้อุปกรณ์ไอพี DHCPv6 จะอนุญาตให้อุปกรณ์ไอพีตั้งค่าอัตโนมัติจึงเป็นเพราะว่าทำไมถึงชื่อว่า Stateful address Configulation เพราะ DHCPv6 Server จะเป็นคนรักษาสถานะและแจกที่อยู่

SLAAC Stateless Address Auto Configuration [RFC4862] เป็นวิธีใหม่นำเสนอด้วยไอพีรุ่นที่หกที่อนุญาตให้จัดการตั้งค่า ตัวแปรในเครือข่ายบนอุปกรณ์ไอพี โดยให้เราเตอร์มอบการเชื่อมต่อกับเครือข่ายให้

ข้อดีของ SLAAC จะทำให้การตั้งค่าอุปกรณ์จำนวนมากทำได้ง่ายขึ้น เช่น เซ็นเซอร์กล้อง หรืออุปกรณ์พลังงานต่ำอื่นๆ ไม่ต้องตั้งค่าเครือข่ายในอุปกรณ์ไอพีเพื่อ เพียงเสียบปลักและเชื่อมต่ออินเทอร์เน็ตเน็ต ซึ่งง่ายต่อโครงสร้างเครือข่ายที่ใช้ไอพีรุ่นที่หกเพราะไม่จำเป็นต้องเพิ่มอุปกรณ์หรือเครื่องแม่ข่าย ใช้เราเตอร์ตัวเดิมที่ใช้ส่งแพ็กเก็ตไปนอกเครือข่าย เพื่อตั้งค่าอุปกรณ์ไอพีโดยที่เราไม่ต้องเข้าไปถึงรายละเอียดแต่จำเป็นต้องรู้ว่าอยู่ในเครือข่าย LAN เชื่อมต่ออินเทอร์เน็ตโดยใช้เราเตอร์ โดยเราเตอร์จะดูแลส่วนของการส่งข้อมูลที่จำเป็นต้องตั้งค่าไปยังโฮสโดยใช้ข้อความ RA (Router Advertisement) โดยเราเตอร์จะส่ง RAs เป็นช่วงๆเพื่อกระตุ้นการประมวลผล โฮสสามารถส่งข้อความ RS (Router Solicitation) เมื่อเชื่อมต่อกับเครือข่าย เราเตอร์จะส่ง RA ทันทีเพื่อตอบมายัง RS ตามขั้นตอนดังต่อไปนี้



รูปที่ 2.6 การแลกเปลี่ยนแพ็กเก็ตในไอพีรุ่นที่หก

1. R1 เราเตอร์มอบการเชื่อมต่อไปยังโฮสใน LAN และคอยส่ง RAs เป็นช่วงๆ

2. ทั้ง R1 และโฮสจะมี Link-Local address ในการเชื่อมต่อไปยังโฮสของที่อยู่ใน LAN จะตั้งค่าอัตโนมัติเมื่อการเชื่อมต่อพร้อมใช้งาน โฮสจะสร้าง Link-Local address โดยรวม 64 บิตด้านซ้ายของ link-local Prefix (fe80::/64) และ 64 บิตด้านขวาจะสร้าง IID (:3432:7ff1:c001:c2a1) Link-Local address เหล่านี้สามารถใช้ใน LAN เพื่อแลกเปลี่ยนแพ็กเก็ตแต่จะไม่สามารถส่งแพ็กเก็ตไปนอกเครือข่าย LAN ได้

3. โฮสจำเป็นต้องมีพื้นฐาน 2 อย่างเพื่อส่งข้อมูลไปนอกเครือข่ายคือ Global IPv6 Address และที่อยู่ของเกตเวย์ เช่น เราเตอร์ที่จะส่งแพ็กเก็ตจะต้องการที่จะรับการหาเส้นทางเพื่อส่งไปนอกเครือข่าย

4. R1 ส่ง RAs เป็นช่วงๆ (ทุกๆหลายวินาที) เมื่อโฮสรับการเชื่อมต่อและตั้งค่า Link-Local address และส่ง RS ไปยัง R1 และ R1จะตอบด้วย RA ซึ่งบรรจุ 2 อย่างทันที

(1) Global prefix of length 64 บิต เพื่อสำหรับ SLAAC โฮสจะใช้รับ Prefix และมอบ IID เฉพาะมักเป็นส่วนที่ใช้สำหรับ Link-Local address ส่วนทาง Global IPv6 Address จะถูกตั้งค่าในโฮสและสามารถสื่อสารบนเครือข่าย ไอพีรุ่นที่หก

(2) รวมไปถึง Link-local address ของ R1 เพราะเป็นที่อยู่ต้นทางของ RA โฮสจะสามารถใช้ ที่อยู่นี้ตั้งค่าดั่งเดิม เกตเวย์ส่งแพ็กเก็ต ออกไปโดยอัตโนมัติเพื่อไปยังโฮสไอพีรุ่นที่หกบนเครือข่าย

5. ทั้งเกตเวย์และไอพีรุ่นที่หกจะถูกตั้งค่า โฮสจะรับและส่งข้อมูล จากรูปภาพมีบางอย่างส่ง (Tx Data) ไปยังโฮสบนเครือข่าย เพื่อสร้างแพ็กเก็ตไอพีรุ่นที่หกด้วยที่อยู่ปลายทางของผู้รับและต้นทางจะถูก auto-configured global address ที่ส่งมายังเกตเวย์ Link-local address ของ R1 โฮสปลายทางจะสามารถตอบข้อมูลด้วย บางข้อมูล (Rx Data)

2.3.6 ข้อดีของ ไอพีรุ่นที่หก เมื่อทำงานร่วมกับ IoT

2.3.6.1 ไอพีรุ่นที่หกจะถูกยอมรับไปตามเวลา

การเชื่อมต่ออินเทอร์เน็ตจำเป็นต้องมีโพรโทคอลเครือข่ายซึ่งจะบอกที่อยู่ในการส่งข้อมูลจากเว็บ ไอพีรุ่นที่สี่นั้นมีมีจำนวนที่อยู่ไอพี ที่จำกัด ซึ่งจะต้องเปลี่ยนเป็นไอพีรุ่นที่หกอย่างหลีกเลี่ยงไม่ได้ การสำรวจของกูเกิลพบว่าไอพีรุ่นที่หกได้รับการยอมรับเพิ่มขึ้นเป็นเท่าตัวทุกๆ 9 เดือน

2.3.6.2 ขยายเครือข่ายได้ง่าย

ไอพีรุ่นที่หกให้ที่อยู่กับอุปกรณ์ได้มากถึง 2^{128} เลขหมาย ซึ่งเพียงพอต่อปัจจุบันและอนาคตสำหรับอุปกรณ์การสื่อสารทุกชิ้นบนโลก

2.3.6.3 แก้ปัญหา NAT บังไอพีอุปกรณ์นอกเครือข่าย

เมื่อไอพีรุ่นที่สี่ถึงขีดจำกัด การขยายตัวอย่างไม่มีจำกัดอินเทอร์เน็ตจึงต้องยอมรับวิธี NAT ซึ่งจะอนุญาตให้ผู้ใช้งานแบ่งไอพีสาธารณะใช้ ซึ่งมีข้อเสียถึงสองอย่าง

(1) ผู้ใช้งาน NAT จะยืมและแบ่ง ที่อยู่ไอพีกับคนอื่น ส่งผลให้ไม่มีที่อยู่ไอพีสาธารณะ ไอพีจะกลายมาเป็นผู้ใช้งานอินเทอร์เน็ตที่ไม่มีไอพีจริง โดยสามารถใช้งานอินเทอร์เน็ตได้แต่ไม่สามารถถูกเข้าถึงจากอินเทอร์เน็ตได้

(2) ทำให้การเชื่อมต่อแบบปลายทางไปยังปลายทางโดยตรงไม่ได้ และช้าลง มีความปลอดภัยต่ำ

2.3.6.4 มีความปลอดภัยมาก

ไอพีรุ่นที่หกครอบคลุมการเชื่อมต่อปลายทางไปยังปลายทางซึ่งมีหลายกลไกการหาเส้นทาง และไอพีรุ่นที่หกรองรับชุมชนผู้ใช้งานจำนวนมาก ผู้วิจัยที่จะช่วยเพิ่มความปลอดภัย รวมไปถึง IP Sec

2.3.6.5 กองซ้อนมีขนาดเล็ก

ไอพีรุ่นที่หกและ IoT ถูกวิจัยมาหลายปี ผู้พัฒนาได้พัฒนาการย่อขนาดไอพีรุ่นที่หกลงซึ่งก็คือ 6LoWPAN ซึ่งง่ายและมีประสิทธิภาพต่อกลไกเพื่อลดขนาดที่อยู่ลงเพื่ออุปกรณ์ที่มีข้อจำกัด อย่างไรก็ตามขอบของเครือข่ายจะต้องแปลงการบีบอัดให้กลับเป็นไอพีรุ่นที่หก กองซ้อนขนาดเล็กถูกพัฒนาเช่นเดียวกับ Contiki ซึ่งกินขนาดน้อยกว่า 11.5 กิโลไบต์

2.3.6.6 เครือข่ายของอุปกรณ์

ไอพีรุ่นที่หกมีที่จำนวนโพรโทคอลเครือข่ายมาก สามารถให้บริการทุกๆ อุปกรณ์ได้ การทดลองพิสูจน์ได้ว่าไอพีรุ่นที่หกมีมากพอสำหรับเซ็นเซอร์บ้านอัจฉริยะ เมืองอัจฉริยะ COAP เป็นโพรโทคอลที่อนุญาต ให้บริการเว็บเซอร์วิสได้ง่ายที่จะทำตาม สถาปัตยกรรม REST

2.3.6.7 ความรวดเร็ว

ไอพีรุ่นที่หกรองรับการทำงานที่รวดเร็วของโหนดปลายทางและการหาเส้นทางของโหนดที่รวดเร็ว

2.3.6.8 การตั้งค่าที่อยู่ด้วยตัวเอง

ไอพีรุ่นที่หก รองรับการตั้งค่าอุปกรณ์ด้วยตัวเอง โดยโหนดสามารถกำหนดที่อยู่ของตนเองได้ ซึ่งจะช่วยลดค่าใช้จ่ายในการตั้งค่าได้

2.3.6.9 ทำตามอินเทอร์เน็ตอย่างสมบูรณ์

ไอพีรุ่นที่หกส่งผลให้พัฒนาโครงข่าย อุปกรณ์สามารถทำงานร่วมกับอุปกรณ์ได้อย่างแท้จริงครอบคลุมทั่วโลก

ทั้ง 9 ปัจจัยนี้ทำให้ไอพีรุ่นที่หกมีความจำเป็นต่อ IoT เป็นอย่างมากแต่การจะนำ IoT มาใช้งานร่วมกับไอพีรุ่นที่หกเลยเป็นเรื่องยากเพราะไอพีรุ่นที่หกเองยังใช้พลังงานที่สูงและ IoT จะต้องใช้สถาปัตยกรรมซอฟต์แวร์ที่มีความสามารถที่จะจัดการกับข้อมูลจำนวนมาก การจัดการ การคำนวณ การประมวลผลใหม่ๆ การถ่ายทอดสด การกรองข้อมูล การรวมข้อมูล และทำเหมืองข้อมูล ทั้งหมดนี้ถูกทำงานบน HTTP และไอพีซึ่งแตกต่างจากธรรมชาติของ IoT ที่ใช้พลังงานน้อยและต้องการสื่ออื่นๆเพื่อให้ต่ออินเทอร์เน็ตได้ขณะใช้งานแบตเตอรี่พลังงานจะถูกใช้กับการส่งข้อมูลที่ไม่จำเป็น หลายโพรโทคอลไม่เหมาะกับรูปแบบการสื่อสาร ซึ่งจำเป็นจะต้องเชื่อมต่ออินเทอร์เน็ต อินเทอร์เน็ตโพรโทคอลที่มีอยู่เช่น HTTP TCP ไม่เหมาะกับการสื่อสารที่ใช้งานพลังงานต่ำ เนื่องจากทั้งข้อมูลและส่วนหัวต้องการความน่าเชื่อถือของแพ็กเก็ตในขั้นที่สูงขึ้น ซึ่งเป็นอุปสรรคต่อการปรับโพรโทคอลที่มีอยู่ให้เข้ากับการทำงานของเครือข่ายนั้นๆ เพื่อที่จะเชื่อต่อกัน เช่น เดี่ยวกับอินเทอร์เน็ตเชื่อมต่อหลายอุปกรณ์ IoT เช่น RFID Sensor และอื่นๆ ซึ่งใช้พลังงานน้อย มีความน่าเชื่อถือ และเข้าถึงอินเทอร์เน็ตได้

2.4 เครือข่ายส่วนบุคคลไร้สายที่ใช้พลังงานต่ำสำหรับไอพีรุ่นที่หก

เครือข่ายส่วนบุคคลไร้สายที่ใช้พลังงานต่ำสำหรับไอพีรุ่นที่หก (IPv6 over Low power Wireless Personal Area Networks, 6LoWPAN) คือ ส่วนขับเคลื่อนของ IoT ซึ่งจะเชื่อมต่อกันผ่านเครือข่ายไร้สาย โดยสร้างช่องทางการเชื่อมต่อและรับส่งข้อมูลกัน การนำเทคโนโลยีเครือข่ายไร้สายมาใช้จะเพิ่มอุปกรณ์ที่เชื่อมต่อกัน ส่งผลให้เกิดข้อจำกัดในเรื่องของการใช้งาน อายุแบตเตอรี่ การใช้พลังงาน และระยะห่างระหว่างอุปกรณ์ เทคโนโลยีใหม่และโพรโทคอลควรรับมือสภาวะใหม่ถูกเรียกว่า Low power and Lossy network (LLNs) โดยมีลักษณะดังต่อไปนี้

1. จำเป็นต้องรองรับอุปกรณ์มากกว่าที่อยู่บน LAN ในปัจจุบัน
2. จำกัดโค้ดและการใช้แรมในอุปกรณ์
3. ข้อจำกัดในเรื่องของสื่อสารด้านระยะห่างของเครือข่าย พลังงาน และการประมวลผล
4. ทุกองค์ประกอบควรทำงานด้วยกันและใช้พลังงานและแบนด์วิธน์เหมาะสม

ปัจจัยอื่นที่จะถูกนำมาใช้ภายใน IoT คือการใช้ไอพีเป็นโพรโทคอล การใช้ไอพีมีข้อดีหลายอย่าง เพราะเป็นมาตรฐานที่เปิดให้ใช้อย่างแพร่หลาย นำไปใช้ได้ง่าย ทำงานร่วมกันได้ดีและง่ายต่อการนำไปพัฒนา การใช้มาตรฐานที่เหมือนกัน เช่น การเชื่อมต่อจากปลายทางไปยังปลายทางบนพื้นฐานไอพีซึ่งแก้ปัญหาการทำงานร่วมกันไม่ได้

สำหรับเทคโนโลยีการสื่อสารแบบไร้สาย IEEE 802.15.4 เป็นความหวังของเลเยอร์ชั้นด้านล่าง (ระดับชั้น Data Link และระดับชั้น Physical Layer) แม้ว่าจะมีการใช้ ทางเลือกดีๆเช่น Low Power WiFi, Bluetooth พลังงานต่ำ, DECT Ultra Low Energy, ITU-T G.9959 และ NFC(Near Field Communication)

IETF แตงต่างจาก Working groups (WGs) ที่จะคอยพัฒนามาตรฐานเพื่อ WSN

1. 6LoWPAN เป็นมาตรฐานสำหรับไอพีรุ่นที่หกสื่อสารบน IEEE 802.15.4 เทคโนโลยีเครือข่ายไร้สาย 6LoWPAN ทำหน้าที่ระดับชั้นในการปรับตัวระหว่างมาตรฐานของไอพีรุ่นที่หกสื่อสารด้วยพลังงานต่ำและเครือข่ายไร้สายที่ถูกบีบอัด เป็นสื่อกลางโดย IEEE 802.15.4 ซึ่งสามารถใช้งานกับแค่ไอพีรุ่นที่หกเท่านั้นไม่รองรับไอพีรุ่นที่สี่

2. Roll (Routing Over Low power and Lossy networks) จำเป็นในการใช้วิธีเส้นทางที่เหมาะสม เพราะไม่สามารถใช้โพรโทคอลในการหาเส้นทางแบบที่มีอยู่ได้ WGs เน้นไปที่การแก้ปัญหาการหาเส้นทางทุกแอปพลิเคชันที่ที่เป็นไปได้ของ LLNs (อุตสาหกรรม บ้าน สิ่งก่อสร้างและเครือข่ายเซ็นเซอร์ในเมือง)

และโพรโทคอลถูกออกแบบให้เหมาะสมกับแอปพลิเคชัน WGs เน้นไปที่กรอบความคิดของสถาปัตยกรรม การหาเส้นทางบนไอพีรุ่นที่หก

3. 6lo (IPv6 over Networks of Resource-Constrained Nodes) WGs จัดการกับไอพีรุ่นที่หกให้เชื่อมต่อกับโหนดบนเครือข่ายที่มีข้อจำกัด ด้วยการขยายการทำงานของ 6LoWPAN WGs นิยาม IPv6-Over-Foo ที่ในชั้นของการปรับตัวใช้ 6LoWPAN สำหรับชั้น Data Linkในโหนดบนเครือข่ายที่มีข้อจำกัด

จากที่กล่าวมา 6LoWPAN เป็นจุดเริ่มต้นของงานที่ดำเนินการตามมาตรฐานของ IETF เพื่อสื่อสารบนโหนดที่มีข้อจำกัดด้านทรัพยากรใน LLNs โดยใช้ไอพีรุ่นที่หก การทำงานบน 6LoWPAN สำเร็จเรียบร้อยและสำเร็จมากยิ่งขึ้นด้วย Roll WGs ตอบสนองความต้องการของการหาเส้นทางและ 6lo เพื่อขยายมาตรฐาน 6LoWPAN ไปยังระกับชั้นในการเชื่อมต่ออื่น ตามมาด้วยรายละเอียดเพิ่มเติมที่เกี่ยวกับ 6LoWPAN ขั้นตอนแรกคือไอพีรุ่นที่หกบน WSN/IoT 6LoWPAN และมาตรฐานที่ใกล้เคียงต่างเป็นกังวลเกี่ยวกับการนำไอพีเพื่อเชื่อมต่ออุปกรณ์ซึ่งไม่เหมาะสมกับระดับชั้นด้านบน ยกเว้นเฉพาะ UDP ในชั้นการขนส่งข้อมูล

2.4.1 ภาพรวมของ LoWPAN

Low-power and lossy Network (LLNs) โดยทั่วไปจะหมายถึงเครือข่ายที่สร้างจากโหนดที่มีข้อจำกัดมาก (หน่วยประมวลผลที่มีข้อจำกัด หน่วยความจำ พลังงาน) เชื่อมต่อกันด้วยลักษณะ “ลดทอน” ผ่านคลื่นวิทยุพลังงานต่ำ มีลักษณะความเร็วต่ำ ประสิทธิภาพต่ำ มีราคาถูก และมีการเชื่อมต่อที่ไม่เสถียร

LoWPAN เฉพาะกรณีของ LLNs นั้นสร้างขึ้นโดยอุปกรณ์ที่ทำตามมาตรฐาน IEEE 804.15.4

ตัวอย่างคุณลักษณะของอุปกรณ์ LoWPAN

(1) มีความสามารถในการประมวลผลที่จำกัด ต่างรุ่นก็มีความเร็วสัญญาณนาฬิกาที่ต่างกัน โดยเริ่มจาก 8 บิต

(2) มีหน่วยความจำขนาดเล็ก แรมและรอมมีขนาดเล็ก ซึ่งคาดว่าจะมากกว่านี้ในอนาคต จึงต้องพยายามใช้น้อยที่สุดเท่าที่จำเป็น

(3) มีพลังงานต่ำ: เพื่อการใช้พลังงานที่ประหยัด

(4) มีระยะสั้น POS (The Personal Operating Space) ได้นิยามโดย IEEE 802.15.4 ว่าสามารถใช้ได้ในระยะ 10 เมตร สำหรับการใช้งานจริงและมากกว่า 100 เมตรสำหรับทางที่ไม่มีสิ่งกีดขวาง

(5) มีต้นทุนที่ต่ำขับเคลื่อนไปด้วยลักษณะ เช่น การประมวลผลที่ช้า หน่วยความจำต่ำ และอื่นๆ

ทุกข้อจำกัดบนโหนดจะเปลี่ยนในอนาคตเหมือนเทคโนโลยีที่พัฒนาขึ้น แต่เปรียบกับปัจจัยอื่น คาดว่า LoWPANs นั้นสามารถใช้งานได้ในอุปกรณ์ที่มีข้อจำกัดอยู่มาก เพื่อรองรับอุปกรณ์ราคาถูกใช้ได้นานและข้อจำกัดสมบัติอื่นๆด้วย

โดยปกติ LoWPAN จะรวมไปถึงอุปกรณ์ที่ทำงานร่วมกันเพื่อเชื่อมต่อสภาพแวดล้อมจริงไปยังแอปพลิเคชันที่ใช้จริง เช่น เซ็นเซอร์ไร้สาย แม้ว่า LoWPAN ไม่จำเป็นต้องประกอบด้วยโหนดเซ็นเซอร์เพียงอย่างเดียว ซึ่งอาจจะมีตัวกระตุ้นอื่นๆ

นอกจากนี้สิ่งสำคัญของลักษณะของ LoWPAN เพราะเป็นข้อจำกัดที่จะชี้นำทางในด้านเทคนิค

(1) แพ็กเก็ตขนาดเล็ก ขนาดเฟรมที่สูงสุดของ physical layer คือ 127 ไบต์ ผลลัพธ์สูงสุดในเฟรมที่ระดับชั้น MAC (Media Access Control) คือ 102 Octet Link-layer มีความปลอดภัยที่เพิ่มขึ้น ที่เพิ่มเข้ามาสำหรับข้อมูลแพ็กเก็ต สูงสุด 81 Octet

(2) IEEE 802.15.4 นิยามที่อยู่ไว้หลายรูปแบบที่อนุญาตให้ใช้อย่างใดอย่างหนึ่ง IEEE 64 บิต ถูกเสริมเข้ามา หรือหลังจากการรวมตัวที่อยู่ 16 บิต เฉพาะเพื่อใช้ใน PAN (Personal Area Network)

(3) แบนด์วิดธ์ อัตราการส่งข้อมูลขนาด 2510 Kbps, 40 Kbps , และ 20 Kbps ที่ระดับชั้น Physical กำหนดไว้คือ (2.4 GHz, 915 MHz และ 868 MHz ตามลำดับ)

(4) โทโพโลยีแบบดาวและตาข่าย

(5) อุปกรณ์จำนวนมากที่จะถูกปล่อยออกไปตลอดการใช้งาน ตำแหน่งของอุปกรณ์ไม่ต้องถูกกำหนดล่วงหน้า มักถูกใช้งานในรูปแบบ ad-hoc บางครั้งอุปกรณ์อาจถูกวางไว้ในจุดที่เข้าถึงยากหรือง่ายต่อการเคลื่อนย้ายก็ได้

(6) อุปกรณ์ภายใน LoWPAN มีแนวโน้มที่จะไม่น่าเชื่อถือหลากหลายเหตุผล การเชื่อมต่อด้วยคลื่นวิทยุที่ไม่แน่นอน การใช้พลังงาน อุปกรณ์ค้าง การโดนปลอมข้อมูล และอื่นๆ

(7) โหมดนิทรา อุปกรณ์อาจจะพักเป็นช่วงเวลานานเพื่อประหยัดพลังงานทำให้ไม่สามารถสื่อสารได้ขณะพัก

2.4.2 การใช้ไอพีบน 6LoWPAN

จากที่ได้กล่าวมาแล้วดูเหมือนว่าการใช้ไอพีโดยเฉพาะไอพีรุ่นที่หก ถูกปรับใช้ไปในวงกว้างเพราะว่ามีข้อดีที่หลากหลาย 6LoWPAN ก็คือไอพีรุ่นที่หกบนเครือข่าย 6LoWPAN ในส่วนที่จะพูดถึงประโยชน์และปัญหาที่เพิ่มเข้ามาเมื่อใช้ไอพีบน 6LoWPAN แอปพลิเคชันของเทคโนโลยีไอพีโดยเฉพาะเครือข่ายไอพีรุ่นที่หกถูกยอมรับว่ามีประโยชน์ต่อ 6LoWPAN ตามนี้

(1) ธรรมชาติของไอพีคือการมีอยู่ที่แพร่หลายโดยใช้ประโยชน์จาก โครงสร้างที่มีอยู่

(2) เทคโนโลยีบนไอพีมีอยู่แล้วอย่างที่รู้กันดีว่าถูกใช้งานอย่างแพร่หลาย ซึ่งช่วยให้ใช้งานได้ง่าย ง่ายต่อการปรับปรุงทำงานร่วมกันได้ดีและยังง่ายต่อการพัฒนาแอปพลิเคชัน

(3) เทคโนโลยีเครือข่ายไอพีถูกเปิดให้ใช้งานอย่างอิสระตามข้อกำหนด ซึ่งผู้คนสามารถเข้าใจได้ง่ายกว่าเลือกวิธีที่มีเจ้าของกรรมสิทธิ์

(4) เครื่องมือสำหรับไอพีนั้นมีอยู่แล้ว

(5) อุปกรณ์ที่ใช้ไอพีสามารถเชื่อมต่อไปยังเครือข่ายอื่นได้ง่ายโดยที่ไม่จำเป็นต้องผ่านตัวกลาง เช่น ผ่านการแปลงโพรโทคอล เกตเวย์ หรือพร็อกซี

(6) การใช้ไอพีรุ่นที่หกช่วยให้มีที่อยู่ใช้งานเป็นจำนวนมากซึ่งง่ายต่อการตั้งค่าตัวแปรด้วย การตั้งค่าอัตโนมัติ (SLAAC) ซึ่งเป็นเรื่องสำคัญสำหรับ 6LoWPAN ที่มีรองรับการทำงานของอุปกรณ์จำนวนมาก

ในทางกลับกันการใช้ไอพีเพื่อสื่อสารบน 6LoWPAN นำมาซึ่งหลายปัญหาที่ต้องจำไว้

(1) เชื่อมต่อด้วยไอพี หนึ่งในลักษณะของ 6LoWPAN คือมีขนาดแพ็กเก็ตที่จำกัดซึ่งส่วนที่เพิ่มเข้ามาสำหรับส่วนหัวของไอพีรุ่นที่หก และเลเยอร์ด้านบนจะต้องถูกบีบอัดเท่าที่เป็นไปได้

(2) โทโพโลยี 6LoWPAN รองรับหลากหลายโทโพโลยีรวมไปถึงแบบตาข่ายและแบบดาว โทโพโลยีแบบตาข่ายจะใช้ การกระโดดข้าม (multi-hop) เพื่อไปยังปลายทางตามที่ต้องการ ดังตัวอย่างอุปกรณ์ตัวกลางจะทำหน้าที่ส่งต่อแพ็กเก็ตที่ระดับชั้น Data Link โทโพโลยีแบบดาวจะจัดการเตรียมอุปกรณ์ที่มีความสามารถในการส่งต่อข้อมูล ซึ่งเพิ่มเข้ามาใน IEEE 802.15.4 โดยอุปกรณ์เหล่านี้ใช้วิธีการอื่นในการเชื่อมต่อเครือข่าย เช่น อีเทอร์เน็ตหรือ 802.11 โดยเป้าหมายคือการรวมเครือข่ายบนเทคโนโลยีที่แตกต่างกันได้อย่างลงตัว ซึ่งแน่นอนว่าเป็นแรงขับเคลื่อนสำคัญที่เริ่มต้นด้วยไอพี

(3) การจำกัดขนาดแพ็กเก็ต แอปพลิเคชันใน 6LoWPAN นั้นคาดหวังให้มีขนาดแพ็กเก็ตที่เล็ก ชั้นที่เพิ่มเข้ามาสำหรับไอพีจะส่งผ่านในเฟรมเดียวโดยที่ไม่ต้องแบ่งส่วนและประกอบใหม่

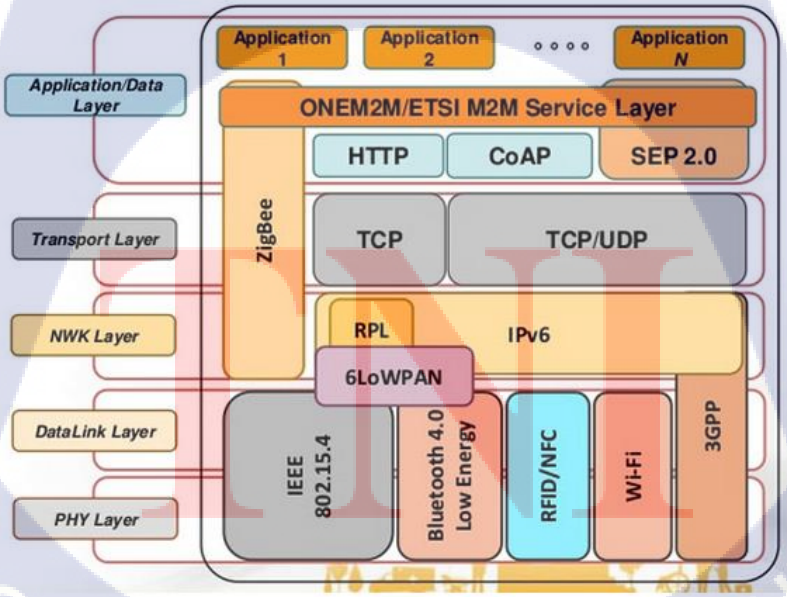
มากเกินไป นอกจากนี้โพรโทคอลจะต้องถูกออกแบบและเป็นส่วนตัว “Control/Protocol packet” ให้พอดีใน 802.15.4 เฟรมเดียว

(4) ข้อจำกัดใสการตั้งค่าและการจัดการ อุปกรณ์ใน LoWPAN คาดว่าจะถูกปล่อยใช้งานอย่างมากมาย นอกจากนี้คาดว่าจะมีข้อจำกัดในเรื่องของการแสดงผลและการรองรับกระแสไฟฟ้า นอกจากนี้สถานที่ตั้งของอุปกรณ์อาจยากที่จะเข้าถึงตามมาด้วยโพรโทคอลที่ใช้ใน LoWPANs ควรตั้งค่าง่ายโดยเฉพาะ “out of the box” ง่ายต่อการออกแบบ อุปกรณ์เหล่าเมื่อเข้าถึงเครือข่ายจะมีลักษณะที่สามารถรักษาตนเองเมื่อได้รับการสืบทอดที่ผิดพลาด

(5) การค้นหาบริการ LoWPANs จำเป็นต้องใช้ Simple Service Discovery Protocol เพื่อค้นหาและควบคุมจัดการบริการโดยอุปกรณ์

(6) ความปลอดภัย IEEE 802.15.4 บังคับให้ Data Link layer รักษาความปลอดภัยบน AES โดยที่จะยกเว้นรายละเอียดเกี่ยวกับ bootstrapping, key management และความปลอดภัยในระดับชั้นที่สูงกว่า ซึ่งแน่นอนว่าเป็นทางออกของการรักษาความปลอดภัยสำหรับอุปกรณ์ LoWPAN ซึ่งจะต้องเลือกแอปพลิเคชั่นอย่างระมัดระวัง

2.4.3 ชั้นการทำงานของ 6LoWPAN



รูปที่ 2.7 ชั้นการทำงานของ 6LoWPAN

จากที่กล่าวมาจะเห็นถึงลักษณะของ 6LoWPAN ไปแล้วโดยการทำงานของ 6LoWPAN จะอยู่ที่ชั้น Data Link และชั้น Network ซึ่งจะทำงานร่วมกับ RPL โพรโตคอลซึ่งจะกล่าวถึงภายหลัง

2.5 ระบบปฏิบัติการ Contiki

Contiki เป็นระบบปฏิบัติการแบบโอเพนซอร์สสำหรับ Internet of Things เพื่อเชื่อมต่ออุปกรณ์ขนาดเล็กราคาถูก หรือไมโครคอนโทรลเลอร์ที่มีพลังงานต่ำไปยังอินเทอร์เน็ต

Contiki รองรับการใช้พลังงานต่ำเพื่อสื่อสารบนอินเทอร์เน็ต รองรับมาตรฐานไอพีรุ่นที่สี่และไอพีรุ่นที่หก พร้อมทั้งมาตรฐานเครือข่ายไร้สายพลังงานต่ำเช่น 6LoWPAN RPL Coap ContikiMAC ของ Contiki และ sleepy routers แม้ว่าทำงานเป็นเราเตอร์ไร้สายสามารถใช้งานได้ด้วยแบตเตอร์รี่

ด้วย Contiki ง่ายต่อการพัฒนาเพราะแอปพลิเคชันถูกเขียนด้วยภาษาซีและมี Cooja simulator ซึ่งสามารถจำลองเครือข่ายก่อนติดตั้งบนอุปกรณ์จริงและ Instant Contiki ที่สามารถเข้าถึงการพัฒนา ด้วยการดาวโหลดเพียงครั้งเดียว

2.5.1 การติดตั้ง Contiki

มีหลายวิธีในการติดตั้ง Contiki โดยติดตั้งจากซอร์สโค้ดหรือใช้การจำลองระบบปฏิบัติการก็ได้ขึ้นอยู่กับความชื่นชอบและเวลาที่มี เพื่อใช้งาน Contiki นั้นจำเป็นต้องมี 3 อย่าง

1.ซอร์สโค้ดของ Contiki

2.แพลตฟอร์มเป้าหมาย (เป็นการจำลองแพลตฟอร์มหรืออุปกรณ์จริงก็ได้)

3.เครื่องมือสำหรับแปลซอร์สโค้ดสำหรับแพลตฟอร์มเป้าหมาย

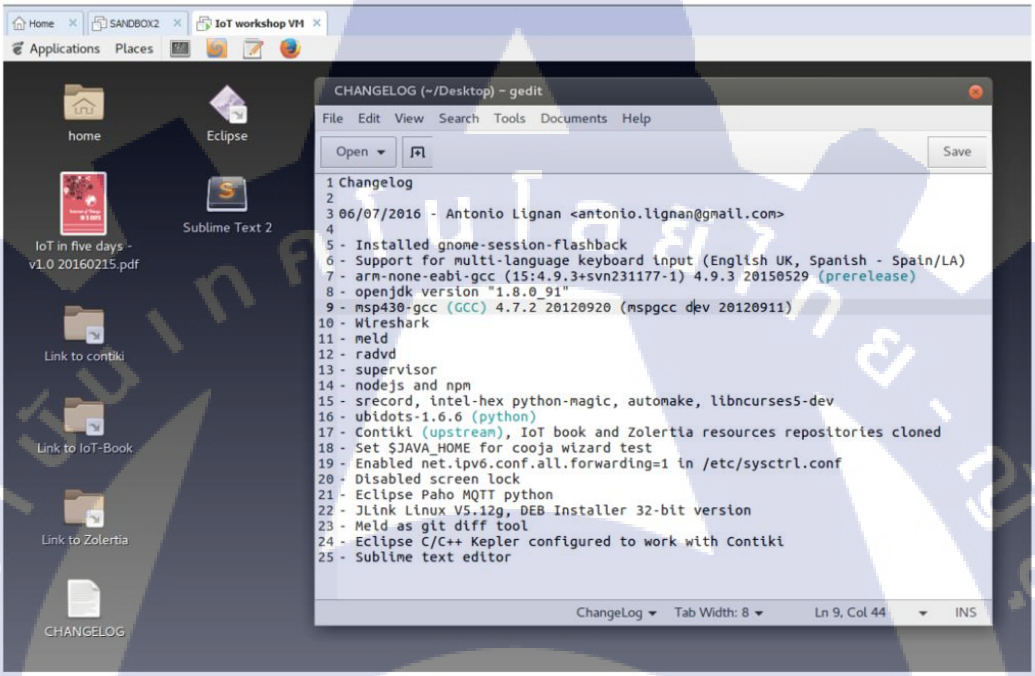
2.5.2 เครื่องมือจำลอง Instant contiki

Instant Contiki นั้นรวมสิ่งจำเป็นทั้งหมดของ Contiki เพื่อใช้พัฒนาในการดาวโหลดเพียงครั้งเดียว ทำงานบนเครื่องจำลอง Ubuntu Linux และมีระบบปฏิบัติการ Contiki พร้อมทั้งเครื่องมือที่ใช้พัฒนา เครื่องมือที่ใช้แปลโปรแกรม และตัวจำลองถูกติดตั้งไว้เรียบร้อยแล้ว สามารถดาวน์โหลด Instant Contiki ได้จากเว็บไซต์ Contiki ได้โดยตรง

โดยใช้ VMware เปิดไปที่ไฟล์ Instant_Contiki_Ubuntu_12.04_32-bit.vmx ถ้ามีการแจ้งเตือนเกี่ยวกับ VMware ให้เลือก I copied It และรอการจำลองของ Ubuntu linux ตื่นขึ้น เข้าสู่ระบบ

Instant Contiki โดยผู้ใช้งานและรหัสผ่านคือ user และอย่าเพิ่งปรับรุ่นในตอนนี้ให้อัพเดท Contiki ก่อนแล้วค่อยปรับรุ่น

เครื่องมือจำลองอย่างเป็นทางการของ IoT 5 days คือ IoT Workshop โดยจะมีทุกอย่างที่มีใน Instant Contiki แต่จะสร้างขึ้นบน Ubuntu Server 16.04 LTS (xenial) แทนเพิ่มเติมด้วยหลายโปรแกรมสำเร็จรูป เพื่อใช้พัฒนาได้ง่ายรวมไปถึง Eclipse IDE Workspace ที่ตั้งค่าไว้เรียบร้อยแล้ว



รูปที่ 2.8 IoT workshop VM

Contiki มีโครงสร้างตามนี้

Folder	Description	Zolertia files
examples	Ready to build examples	examples/zolertia, examples/cc2538-common
app	Contiki applications	-
cpu	Specific MCU files	msp430, cc2538
dev	External chip and devices	cc2420, cc1200
platform	Specific files and platform drivers	z1, zoul
core	Contiki core files and libraries	-
tools	Tools for flashing, debuging, simulating, etc.	zolertia, sky
doc	Self-generated doxygen documentation	-
regression-tests	**nightly regression tests**	**-**

รูปที่ 2.9 โครงสร้าง Contiki

ถ้าหากว่าดึงข้อมูลเลียนแบบมาจากกิ่งของ IoT-Workshop (หรือว่าใช้เครื่องจำลองของ IoT in five days) จะพบแฟ้มนี้ contiki/examples/zolertia/tutorial เนื้อหาของ Tutorial จะมี

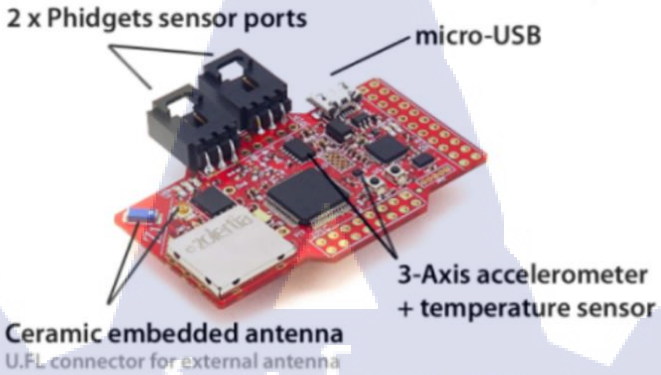
01-basics พื้นฐานของ Contiki (เวลา, GPIO, LEDs)

02-ipv6 เครือข่ายไร้สาย, พื้นฐานสัญญาณวิทยุ, การใช้ IPv6/6LoWPAN(UDP/TCP) และขอบของเครือข่าย

03-coap ตัวอย่างการใช้งาน CoAP server

04-mqtt ตัวอย่างการใช้งาน MQTT client

2.6 Zolertia Z1



รูปที่ 2.10 Z1 motes

Z1 แพลตฟอร์มมีเป้าหมายเพื่อพัฒนาแพลตฟอร์มสำหรับเครือข่ายเซ็นเซอร์ไร้สาย (WSN) ออกแบบมาสำหรับนักวิจัย นักพัฒนา ผู้ที่สนใจหรือเป็นงานอดิเรก แพลตฟอร์มนี้สามารถทำงานร่วมกับ Tmote™ family motes ได้อย่างสมบูรณ์ ด้วยบางอย่างที่ถูกปรับให้ดีขึ้นจึงทำให้มีประสิทธิภาพในบางด้านสูงกว่าเดิมประมาณ 2 เท่า

Z1 motes ใช้รุ่นที่สองของ MSP430F2617 ใช้พลังงานต่ำ 16 บิต RISC หน่วยประมวลผล 16 MHz MCU, มีแรมขนาด 8 kB และรอมขนาด 92 kB และใช้ CC2420 อุปกรณ์รับส่งข้อมูลยอดนิยม ทำงานร่วมกับ IEEE 802.15.4 ได้ ซึ่งทำงานบนคลื่น 2.4 GHz ส่งข้อมูลได้ถึง 250 kbps

Zolertia Z1 สามารถทำงานบนระบบปฏิบัติการ Tiny, ระบบปฏิบัติการ Contiki, ระบบปฏิบัติการ OpenWSN และระบบปฏิบัติการ RIOT ถูกนำมาใช้งานอย่างต่อเนื่องกว่า 5 ปีในมหาวิทยาลัย การวิจัย และศูนย์พัฒนา เป็นสินค้าในเชิงพาณิชย์มากกว่า 43 ประเทศ และมีงานตีพิมพ์ทางวิทยาศาสตร์ออกมากว่า 50 งาน

ลักษณะของ Z1 แพลตฟอร์ม เป็นไมโครคอนโทรลเลอร์ที่ใช้พลังงานต่ำมากและตัวรับส่งสัญญาณ 2.4GHz มี 2 ดิจิทัลเซ็นเซอร์พร้อมใช้งาน (อุณหภูมิและความเร่ง 3 แกน) ใช้ USB สำหรับติดตั้งโปรแกรมได้ง่าย จ่ายพลังงานได้หลายวิธี สามารถใช้ถ่าน (2xAA, 2xAAA) ถ่านนาฬิกา (ได้ถึง 3.6V) ผ่าน USB ต่อสายสองเส้นโดยตรงมายังแหล่งพลังงานเลย USB VCC และ GND ดิจิทัลสามารถต่อพินขยายช่องได้ สามารถต่อไปยังพินแหล่งพลังงานจาก 4V ไป 5.25V ซึ่งจะถูกปรับเป็น 5V และ 3V

2.7 เซ็นเซอร์

เซ็นเซอร์คือตัวแปลงสัญญาณโดยมีวัตถุประสงค์เพื่อตรวจจับหรือวิเคราะห์ลักษณะของสภาพแวดล้อมโดยให้ผลลัพธ์ที่สอดคล้องกัน โดยทั่วไปจะเป็นสัญญาณไฟฟ้าที่เกี่ยวข้องกับตัวแปรที่วัด

2.7.1 อนาล็อกเซ็นเซอร์

อนาล็อกเซ็นเซอร์โดยทั่วไปจะเชื่อมต่อไปยัง ADC (Analogue to Digital Converter) เพื่อแปลงสัญญาณอนาล็อกที่ต่อเนื่องให้อ่านค่าเป็นดิจิทัลในมิลลิโวลต์ คุณภาพและผลลัพธ์ในการวัดจะขึ้นอยู่กับทั้ง ADC (ได้ถึง 12 บิต ใน Z1 และ Re-mote) และ Sampling Frequency

Z1 mote และ RE-Mote เซ็นเซอร์แรงดันไฟฟ้าให้เป็นช่องรับสัญญาณ ADC เตรียมไว้แล้ว ตลอดจนนำไปใช้ในการอ่านค่าอนาล็อกเซ็นเซอร์ภายนอกได้

อนาล็อกเซ็นเซอร์นั้นสามารถเชื่อมต่อ Z1 ผ่านพอร์ทอนาล็อก

อนาล็อกเซ็นเซอร์ที่ทดลองแล้วสามารถใช้งานได้

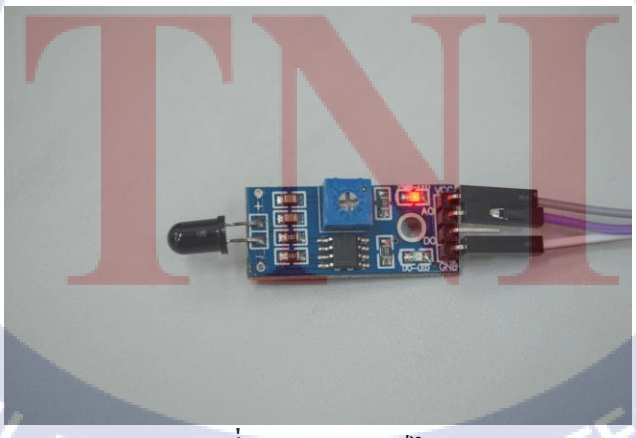
PIR Motion Sensor ใช้ในการตรวจสอบการเคลื่อนไหว

Grove - Light Sensor ใช้ในการวัดค่าของแสง

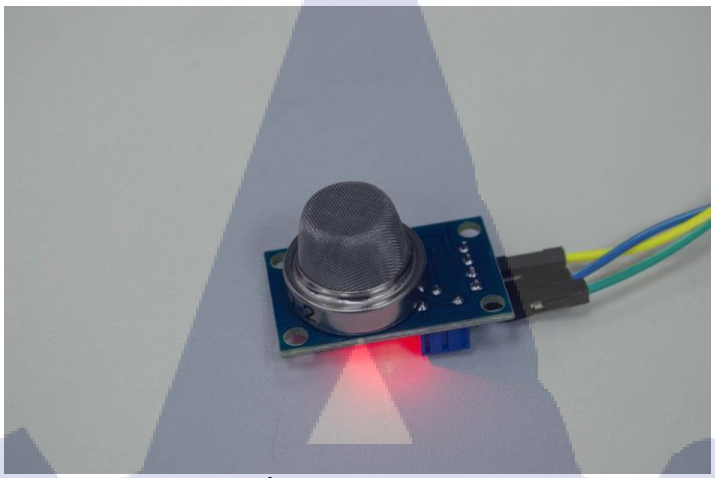
Grove - Rotary Angle Sensor ใช้วัดค่าของการหมุน

Grove - Loudness Sensor ใช้วัดความดังของเสียง

โดยเซ็นเซอร์ไฟและแก๊สที่นำมาใช้ในงานนี้มีลักษณะเป็นอนาล็อกเซ็นเซอร์โดยค่าที่แสดงออกมานั้นมีลักษณะที่ต่างกัน คือ ค่าปกติของเซ็นเซอร์ไฟจะอยู่ที่ประมาณ 4000 และหากตรวจพบค่าจะลดลงมา ส่วนค่าปกติของเซ็นเซอร์แก๊สจะอยู่ที่ 300 หากตรวจพบแก๊ส จะขึ้นไปถึงประมาณ 3000



รูปที่ 2.11 เซ็นเซอร์ไฟ



รูปที่ 2.11 เซ็นเซอร์แก๊ส

2.7.2 ดิจิทัลเซ็นเซอร์

ดิจิทัลเซ็นเซอร์โดยทั่วไปจะจัดการผ่าน Digital communication protocol เช่น I2C, SPI, 1-Wire, Serial หรือขึ้นอยู่กับผู้ผลิต โพรโตคอลที่เป็นกรรมสิทธิ์ โดยทั่วไปตามสัญญาณนาฬิกา ดิจิทัลเซ็นเซอร์นั้นจะมีคำสั่งที่เพิ่มเข้ามา เช่น เปิด ปิด หรือจัดการข้อผิดพลาด

Z1 mote มีดิจิทัลเซ็นเซอร์ 2 อย่างติดตั้งมาแล้วคืออุณหภูมิและความเร่ง 3 แกน

ADXL345 ใช้วัดความเร่งมีตัวอย่างที่ชื่อ test-adxl345.c ใน examples/zolertia/z1 ที่แสดงถึงการทำงานของเซ็นเซอร์

TMP102 เซ็นเซอร์อุณหภูมิแบบดิจิทัล ซึ่งติดตั้งมาบน Z1 แพล็ตฟอร์มมีความแม่นยำ +/-5 เซลเซียส และสามารถวัดอุณหภูมิได้ตั้งแต่จาก -25 เซลเซียส ถึง 85 เซลเซียส ซึ่งตัวอย่าง examples/zolertia/z1/test-102.c จะแสดงถึงการทำงานของเซ็นเซอร์

External digital sensor สามารถเชื่อมต่อกับ Z1 ได้

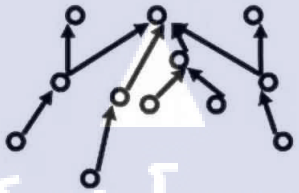
SHT25 Humidity and Temperature ใช้วัดอุณหภูมิและความชื้นด้วยเซ็นเซอร์ตัวเดียว

2.8 โพรโทคอลหาเส้นทางสำหรับอุปกรณ์พลังงานต่ำและลดทอนเครือข่าย

โพรโทคอลหาเส้นทางสำหรับอุปกรณ์พลังงานต่ำและลดทอนเครือข่าย (Routing Protocol for Low power and Lossy Networks , RPL) คือโปรโตคอลหาระยะของเส้นทาง (Distance vector routing protocol) ที่หาเส้นทางบน Destination or DoDAGs

ก่อนที่จะทำความเข้าใจเกี่ยวกับ RPL ต้องรู้ก่อนว่า DODAGs นั้นมีรูปแบบเป็นอย่างไร ต้องเข้าใจถึงศัพท์เฉพาะทางและอะไรคือหัวใจหลักของโพรโตคอลการหาเส้นทางนี้

DAG (Directed Acylic Graph) เป็นกราฟที่กระบวนการไม่สมบูรณ์พวกเราจะเห็นกราฟจำพวกนี้ใน Spanning tree ตามรูปภาพที่ 2.13



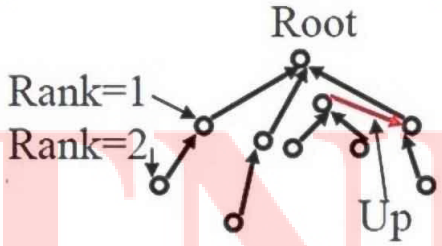
รูปที่ 2.13 DAG

Root เป็นปลายทางของโหนดใน DAG จะไม่มีขาออก

Up เป็นขอบใดก็ได้ที่ส่งข้อมูลโดยตรงไปยัง Root

Down เป็นขอบใดก็ได้ที่ส่งข้อมูลออกจาก Root

Destination Oriented DAG (DODAG) เป็น DAG ชนิดพิเศษที่ทุกโหนดจะต้องส่งไปให้ถึงปลายทางเดียวกันตามาภาพที่ 2.14

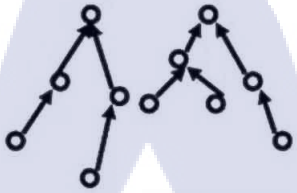


รูปที่ 2.14 DODAG

Objective Function จะช่วยตัดสินใจว่าอยู่ใกล้หรือไกลจาก Root Objective function จะตัดสินใจตามโปรแกรมหรือคนออกแบบเป็นสิ่งที่ต้องการให้มีขนาดเล็ก สามารถเป็นพลังงานหรือแฝงอยู่และถ้าตัดสินใจว่าต้องการจะย่อให้เล็กลง ซึ่งจะมอบหมายเลขลำดับให้

Rank ตามรูปภาพที่ 2.15 แสดงระยะห่างจาก Root

RPL instance เมื่อมี 1 หรือมากกว่า 1 DODAGs ทุกๆ DODAGs จะเป็น instance ตามรูปภาพที่ 2.13 แสดงถึง Two RPL Instances



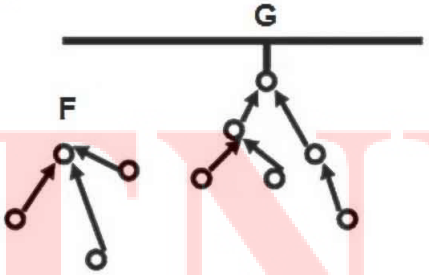
รูปที่ 2.15 Two RPL Instances

DODAG ID ทุกๆ DODAG จะมี IPv6ID 128 บิต จะถูกมอบจาก Root เท่านั้นและตราบที่ Root ไม่เปลี่ยน ID ก็จะไม่เปลี่ยน

DODAG version ทุกๆการเปลี่ยนรูปของ DODAG หมายถึงการเปลี่ยนรูปแบบ

GOAL คือที่ DODAG ต้องการจะไปให้ถึง สามารถเป็นเครือข่ายแบบมีสาย Goal จะเปลี่ยนไปตาม Objective Function ใน Objective Function เล็กไปถึงการทำให้มีขนาดเล็ก อย่างไรก็ตาม GOAL นั้นหมายถึงที่ๆต้องการจะไป

Floating เมื่อ DODAG ไม่เชื่อมต่อหรือยังไม่ถึง Goal จะถูกเรียกว่า Floating ในรูปภาพที่ 2.16 แสดงถึง Floating DODAG



รูปที่ 2.16 Floating และ Grounded

Parent คือที่ที่ลูกศรชี้ไปและ Child คือที่ๆลูกศรออกมาโดย Parents สามารถมีได้หลาย Childs เช่นเดียวกับ Child ที่สามารถมีได้หลาย Parents

Sub-DODAG เป็น Subtree ใดก็ได้ที่ถูกมอบโดย DODAG

Storing โหนดจะเก็บตารางการหาเส้นทางไว้ทั้งหมด โดยจะรู้การเดินทางจากโหนดนึงไปยังโหนดอื่นๆ

Non-Storing โหนดที่ไม่รู้ตารางการหาเส้นทางโดยจะรู้แค่ Parent

DODAG ที่สมบูรณ์จะถูกยกเว้นจาก Root เพื่อรักษารูปร่างโดยจะต้องมีทั้ง Storing และ Non-Storing โดย Root จะต้องเป็น Storing เสมอ

RPL Control-messages มี Control Message 5 รูปแบบใน Spanning tree

1. DODAG information Object (DIO) ข้อความนี้จะมัลติแคสลงด้านล่าง จะให้โหนดใน DODAG มัลติแคสข้อความนี้เพื่อให้โหนดอื่นๆได้รู้ ไม่ว่าจะเป็น Grounded หรือไม่ หรือเป็น Storing หรือ non-storing ก็ตามโดยจะประกาศไปยังโหนดอื่นๆว่า “ถ้าสนใจจะเข้าร่วมให้ตอบกลับมา”

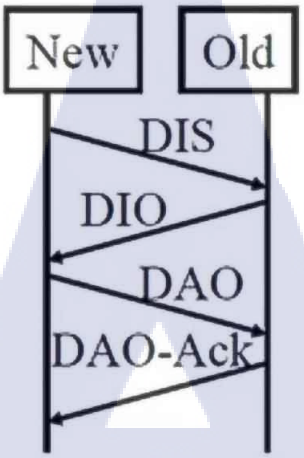
2. DODAG Information Solicitation (DIS) เมื่อไม่ได้รับการประกาศมา ถ้าโหนดต้องการเข้าร่วม DODAG จะส่ง Control Messasge เพื่อรู้การมีอยู่ของ DODAG ข้อความที่ส่งมาจะมีความหมายว่า “มี DODAG อยู่ไหม”

3. DODAG Advertisement Object (DAO) เป็นการร้องขอที่ส่งจาก Child ไปยัง Parent หรือ Root เป็นข้อความเพื่อขออนุญาตจาก Child เข้าร่วม DODAG

4. DAO-ACK เป็นการตอบกลับจาก Root ไป Parent หรือ Child โดยคำตอบเป็นได้ทั้งได้และไม่ได้

5. Consistency check เกี่ยวกับเรื่องของความปลอดภัย

ในภาพที่ 2.17 แสดงถึง Control Message รูปแบบต่างๆ



รูปที่ 2.17 Control Message

รูปแบบของ DODAG

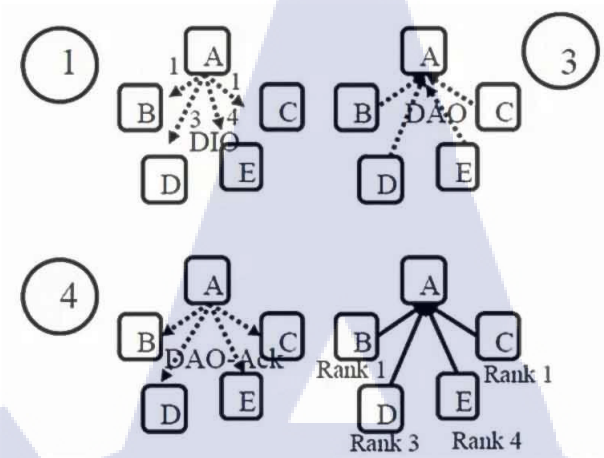
Root ใน DODAG นั้นเป็นโหนดชนิดพิเศษ โหนดทุกตัวไม่สามารถเป็น ROOT ได้ ซึ่งจะต้องถูกโปรแกรมเฉพาะทางไว้สมมุติว่ามี 5 โหนด A, B, C, D, E โดยต้องการจะสร้าง DODAG ตามรูปแบบนี้

1. โหนดA มัลติแคส DIO ออกไป

2. โหนดทั้งหมดได้รับ DIO (B,C,D,E) จะพยายามเข้าร่วมโดยไม่คำนึงถึงระยะห่าง จากนั้นจะรู้ระยะห่างจากโหนด A ว่าเป็น 1,1,3,4 ตามลำดับ

3. โหนด B, C, D, E จะส่ง DAOs มาที่โหนด A

4. โหนด A ยอมรับและตอบกลับด้วย DAO-ACK



รูปที่ 2.18 รูปแบบ DODAG

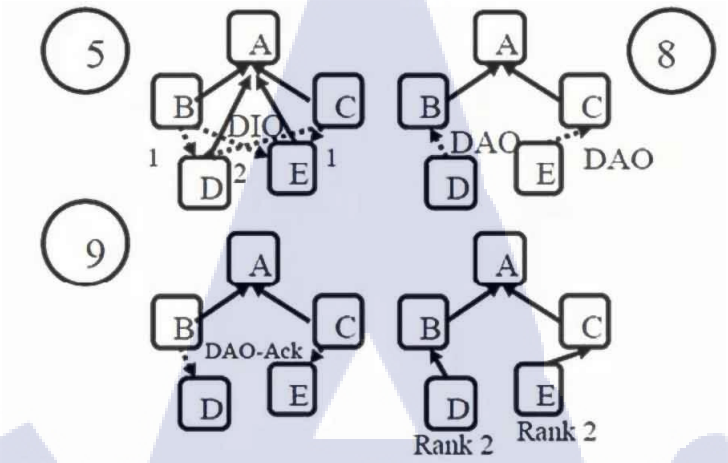
5. เริ่มรอบใหม่ โดยเริ่มจากโหนดที่ใกล้ที่สุดของ Root ตั้งแต่โหนด B และโหนด C ที่อยู่ Rank 1 จะส่ง DIOs

6. โหนด D รับ DIOs และคำนวณระยะห่างจากทางโหนด B และโหนด C คือ 2 และ 1 ตามลำดับ

7. เช่นเดียวกับโหนด E ที่รับ DIOs มาและคำนวณระยะทางจากโหนด B และโหนด C คือ 2 และ 1 ตามลำดับ

8. โหนด D ที่อยู่ใกล้โหนด B มากกว่าและโหนด E ที่อยู่ใกล้โหนด C มากกว่าเพราะฉะนั้น โหนด D จะส่ง DAO กลับไปยังโหนด B และโหนด E จะส่ง DAO กลับไปยังโหนด C

9. เพื่อให้ DODAG สมบูรณ์โหนด C และโหนด B ตอบ DAO-ACK กลับไปยังโหนด D และโหนด E ตามรูป 2.19



รูปที่ 2.19 รูปแบบ DODAG (2)

2.9 IPv6 Tunnel Broker

Zolertia Z1 เป็นอุปกรณ์ที่ทำการสื่อสารบนไอพีรุ่นที่หกซึ่งภายในห้องทดลองของสถาบันเทคโนโลยีไทย-ญี่ปุ่นยังใช้ไอพีรุ่นที่สี่จึงจำเป็นต้องใช้บริการของ Tunnel Broker ในการแปลงเป็นไอพีรุ่นที่หกเพื่อการสื่อสารโดยในการใช้งานมี 2 เงื่อนไขสำคัญคือ

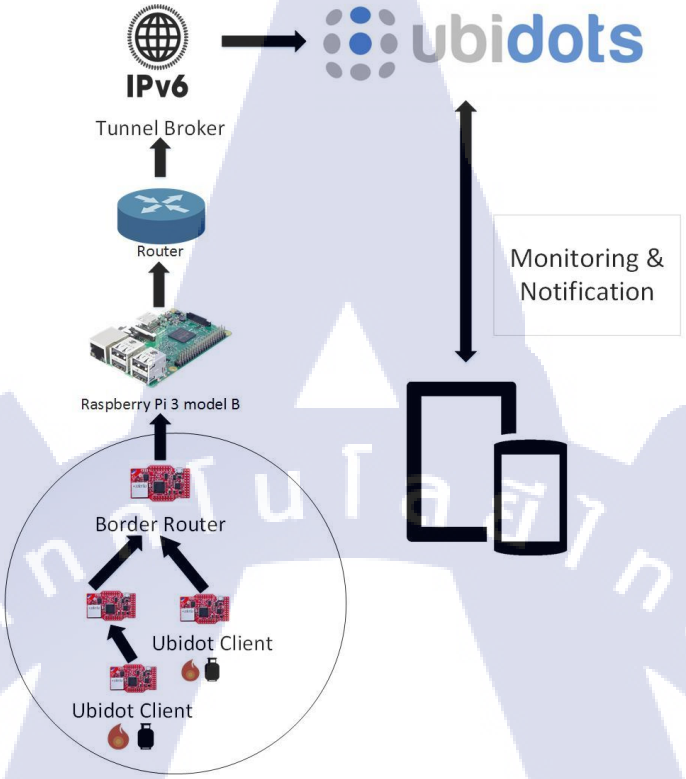
1.ต้องเปิดพอร์ทสำหรับ ICMP เพื่อให้ทางเซิร์ฟเวอร์ตอบกลับมาได้

2.เปิดไอพีโพรโตคอล 41 สำหรับการเพิ่มพื้นที่บนส่วนหัวของไอพีรุ่นที่สี่เพื่อห่อหุ้มแพ็กเก็ต ไอพีรุ่นที่หกเข้าไป

เมื่อทำตาม 2 ข้อนี้จะทำให้สามารถใช้บริการ Tunnel Broker เพื่อใช้งานไอพีรุ่นที่หกได้

2.10 Ubidots

Ubidots คือ IoT คลาวด์แพลตฟอร์มที่จะช่วยในการสร้างแอปพลิเคชั่นซึ่งเก็บข้อมูลจากการตรวจวัดมาใช้ให้เกิดประโยชน์โดยมีคุณสมบัติที่หลากหลาย



รูปที่ 2.20 ตัวอย่างการทำงาน

รูปภาพที่ 2.20 แสดงถึงการทำงานโดยเริ่มจากเมื่อ Z1 ที่ติดตั้งโปรแกรม Ubidots Client ไว้รับค่าจากเซ็นเซอร์ส่งค่ามายัง

Z1 ที่ติดตั้งโปรแกรม Border Router ซึ่งจะส่งค่าผ่าน USB มายังคอมพิวเตอร์โดยจากคอมพิวเตอร์จะส่งออกไปยังเราเตอร์ด้วยไอพีรุ่นที่สี่โดยส่งค่าไปยัง Tunnel Broker เพื่อใช้บริการ Tunnel แปลงเป็นไอพีรุ่นที่หกและส่งค่าไปยัง Ubidots โดยสามารถดูจากโทรศัพท์หรือเมื่อเกิดเหตุการณ์จะสามารถแจ้งเตือนมายังอีเมลหรือเบอร์โทรศัพท์ได้

บทที่ 3
แผนงานการปฏิบัติงานและขั้นตอนการดำเนินงาน

3.1 แผนการทำงาน

ตารางที่ 3.1 แผนการทำงาน

หัวข้อ	เดือน 1				เดือน 2				เดือน 3				เดือน 4			
ศึกษาการทำงานของ Z1 กับเซ็นเซอร์																
ศึกษาเกี่ยวกับการทำงานของเซ็นเซอร์																
ทดสอบการทำงานของเซ็นเซอร์																
ปรับปรุงแก้ไขโค้ดให้เหมาะสม																
ทดลอง																
เขียนรูปเล่ม																

3.1.1 ศึกษาการทำงานของ Zolertia Z1 กับเซ็นเซอร์

ในขั้นตอนนี้ตั้งเป้าหมายไปที่การใช้ศึกษาพอร์ทของ Zolertia Z1 ว่าแต่ละพอร์ทมีลักษณะอย่างไร และสามารถใช้กับเซ็นเซอร์ที่มีได้หรือไม่

3.1.2 ศึกษาเกี่ยวกับการทำงานของเซ็นเซอร์

ศึกษาการทำงานของเซ็นเซอร์ที่มีอยู่หลายชนิด หลายผู้ผลิตโดยแต่ละชนิดเองจะมีลักษณะเพื่อวัดค่าที่ต่างกัน การใช้พลังงานที่ต่างกันส่งผลให้ค่าที่ออกมาต่างกันระยะที่วัดได้ต่างกัน

3.1.3 ทดสอบการทำงานของเซ็นเซอร์

ปรับโปรแกรมเพื่อใช้งานเซ็นเซอร์ที่มีเพื่อให้สามารถทดสอบการทำงานของเซ็นเซอร์ได้และทดลองแต่ละเซ็นเซอร์ว่ามีลักษณะและผลลัพธ์ต่างกันอย่างไร

3.1.4 ปรับปรุงแก้ไขโค้ดให้เหมาะสม

ปรับปรุงแก้ไขโค้ดให้ส่งค่าขึ้น Ubidots โดยจากเดิมใช้ส่งได้แค่เพียงค่าอุณหภูมิและความชื้นเท่านั้น ให้ส่งค่าจากเซ็นเซอร์อื่นๆทั้ง 3.3V และ 5V ส่งค่าให้ขึ้น Ubidots ได้

3.1.5 ทดลอง

ทดลองใช้งานเซ็นเซอร์ไฟและเซ็นเซอร์แก๊สด้วยไฟแช็คภายในห้องทดลอง โดยอัพโหลดค่าจากเซ็นเซอร์ทั้งสองขึ้นไปยัง Ubidots และสามารถแจ้งเตือนมายังข้อความหรืออีเมลได้

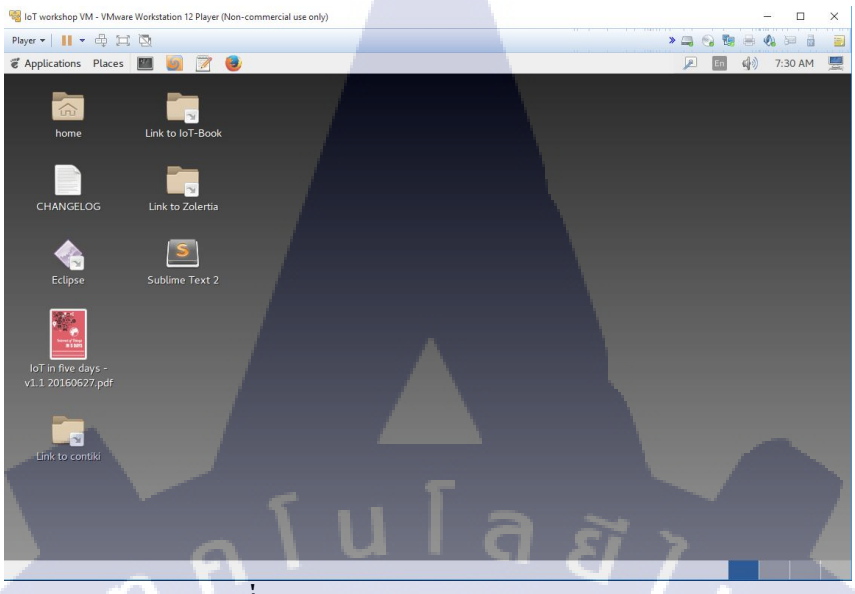
3.1.6 เขียนรูปเล่ม

สรุปงานทดลองและเขียนรูปเล่ม

3.2 ขั้นตอนการดำเนินงาน

3.2.1 ติดตั้ง VMware และ Contiki

สำหรับกรณีที่ใช้งานระบบปฏิบัติการวินโดวส์ ทำการติดตั้ง VMware Workstation player โดยสามารถดาวน์โหลดมาใช้งานและติดตั้งได้ฟรี จากนั้นดาวน์โหลด IoT Workshop VM เพื่อใช้งาน Contiki OS



รูปที่ 3.1 IoT Workshop และ VMware

สำหรับระบบปฏิบัติการ Linux ให้โหลด Contiki ผ่าน Terminal ด้วยคำสั่ง

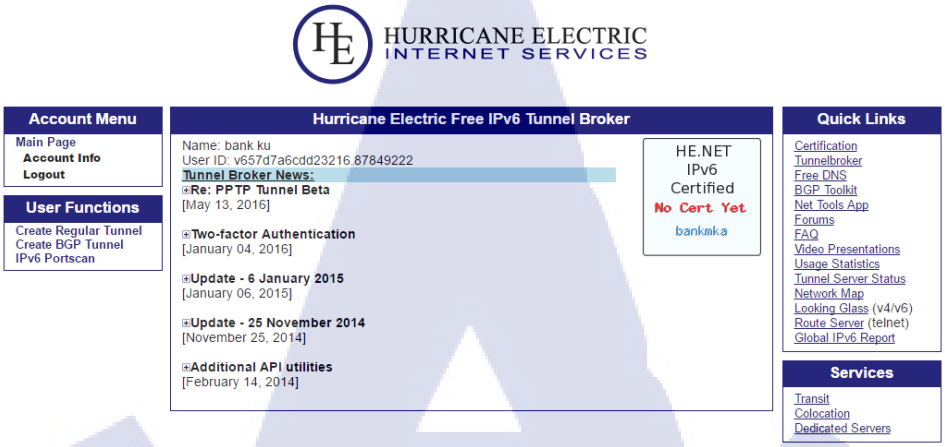
```
cd /home/user/contiki
git remote add iot-workshop https://github.com/alignan/contiki
git fetch iot-workshop
git checkout iot-workshop
```

เพื่อใช้ในการติดตั้ง Contiki จากนั้นให้ดาวน์โหลดเครื่องมือสำหรับแปลซอร์สโค้ดสำหรับ Z1 ด้วยคำสั่ง

```
sudo apt-get -y install gcc gcc-msp430
```

3.2.2 ทำ Tunnel จากไอพีรุ่นที่สี่เพื่อใช้ไอพีรุ่นที่หก

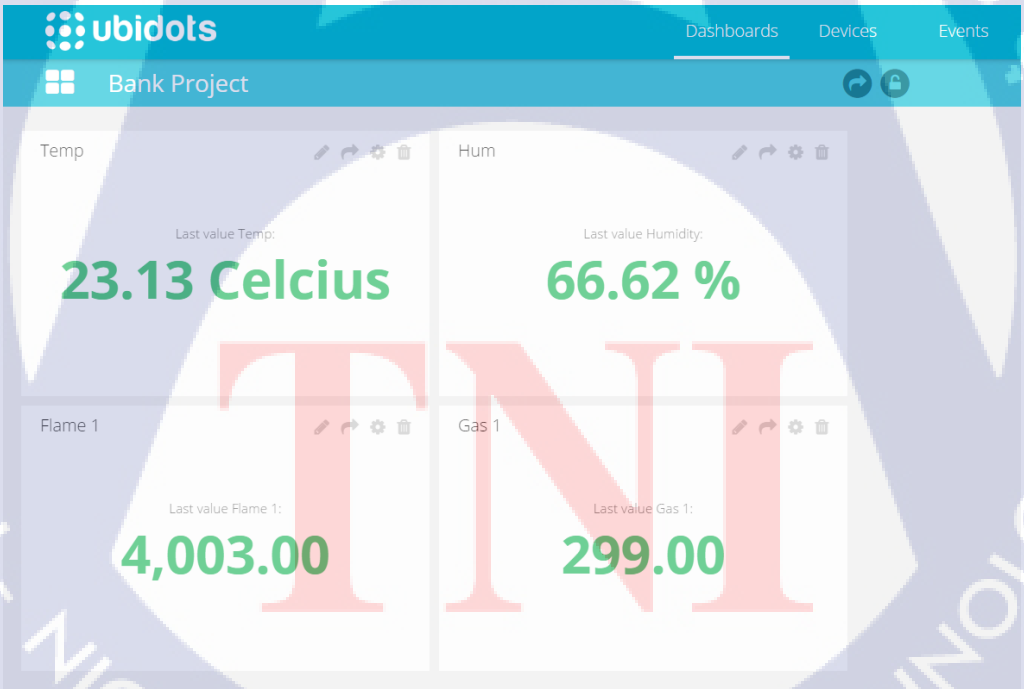
สมัครใช้งานเว็บ Hurricane Tunnel Broker เพื่อขอใช้งานไอพีรุ่นที่หกผ่านไอพีสาธารณะรุ่นที่สี่ที่ใช้งานอยู่ซึ่งจำเป็นต้องเปิดพอร์ท ICMP และโพรโตคอล 41 เพื่อให้แพ็กเก็ตที่ออกไปสามารถทำการ Tunnel เพื่อให้ใช้งานไอพีรุ่นที่หก ได้



รูปที่ 3.2 Tunnel Broker

3.2.3 สมัครใช้งานและตั้งค่า Ubidots เพื่อรับค่าและแจ้งเตือน

สมัคร Ubidots คลาวด์แพลตฟอร์มและตั้งค่าสำหรับเตรียมรับข้อมูลจากเซ็นเซอร์ โดยวิธีการสมัครใช้งานและวิธีการตั้งค่าสามารถดูได้ที่ภาคผนวก ก.



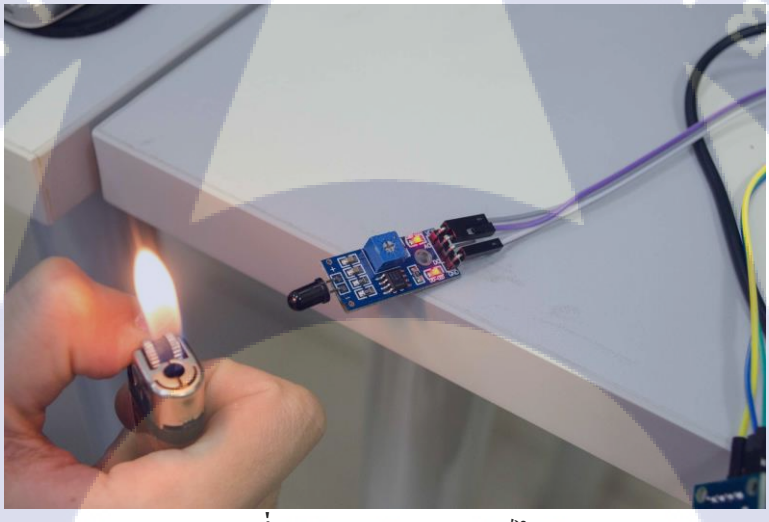
รูปที่ 3.3 Ubidots

3.2.4 ติดตั้งโปรแกรมบนอุปกรณ์ด้วยโปรแกรม Border Router และ Ubidots client บน z1 ที่ต่อกับเซ็นเซอร์แก๊สและเซ็นเซอร์ไฟ

เตรียมอุปกรณ์ Z1 ที่ติดตั้งโปรแกรม Ubidots client ที่แก้ไขโค้ดให้ส่งค่าจากอนาล็อกเซ็นเซอร์โดยเชื่อมต่อกับเซ็นเซอร์ไฟและแก๊สไว้ ส่วน Z1 อีกหนึ่งอันติดตั้งโปรแกรม Border Router เพื่อเป็นทางผ่านข้อมูลโดยเชื่อมต่อกับ Raspberry pi

3.2.5 จุดไฟแช็คให้ติดไฟหรือแก๊สเพื่อทดลองกับเซ็นเซอร์

จุดไฟจากไฟแช็คขึ้นมาโดยสมมุติเหตุการณ์กรณีมีเปลวไฟภายในห้อง ซึ่งหากพบไฟภายในห้องเซ็นเซอร์จะแสดงสีแดงขึ้นมาดังภาพ 3.4 โดยใช้งานร่วมกับเซ็นเซอร์ไฟพร้อมทั้งส่งค่าให้ขึ้น Ubidots ได้ซึ่งจะต้องมีค่าแตกต่างกันเมื่อจุดไฟและไม่จุดไฟ



รูปที่ 3.4 ทดสอบเซ็นเซอร์ไฟ

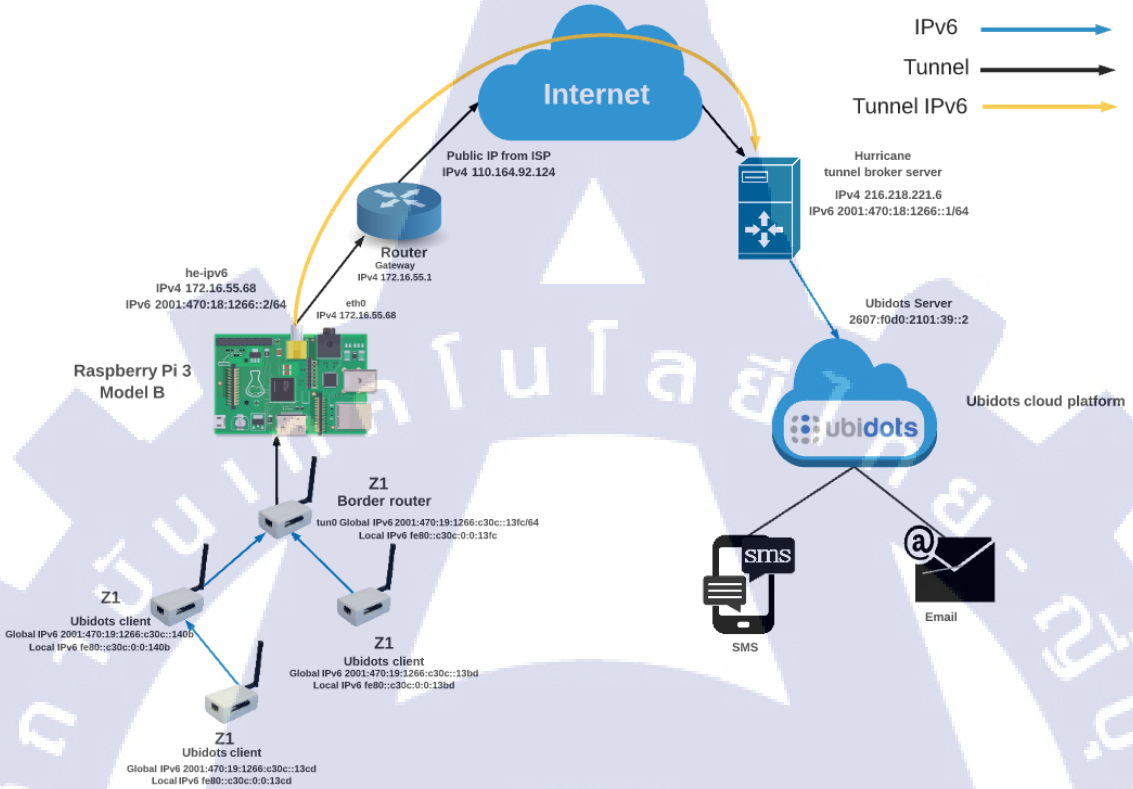
จุดแก๊สจากไฟแช็คโดยจ่อไปที่เซ็นเซอร์เพื่อทดลองกรณีเกิดแก๊สรั่ว ซึ่งจะมีสีเหลือแสดงขึ้นมาบนเซ็นเซอร์ดังภาพ 3.5 โดยค่าที่แสดงบนคลาวด์จะต่างกันกรณีที่มีแก๊สและไม่มีแก๊ส

รูปที่ 3.5 ทดสอบเซ็นเซอร์แก๊ส

3.2.6 Ubidots แจ้งเตือนมายังโทรศัพท์ของผู้ใช้งาน

เมื่อตรวจพบแก๊สหรือไฟจะส่งข้อมูลมายังอีเมลและ SMS ที่ได้ตั้งค่าไว้เพื่อแจ้งเตือนผู้ใช้งาน เพื่อจำลองสถานการณ์ไฟไหมจริง

3.3 ขั้นตอนการทำงาน



รูปที่ 3.6 ขั้นตอนการทำงาน

3.3.1 การตั้งค่าจาก Ubidot Clients (Z1) เพื่อส่งข้อมูลไปยัง Border Router (Z1)

เริ่มจากทำการติดตั้ง Ubidot Clients บนอุปกรณ์ที่ต้องการรับค่าโดยโค้ดที่ทำการติดตั้งถูกปรับปรุงจากเดิมที่สามารถรับค่าได้เพียงจากเซ็นเซอร์พอร์ทดิจิทัลและมีเวลาประมาณ 1 นาทีต่อการอัพเดทหนึ่งครั้งให้ส่งได้เร็วขึ้นประมาณ 3 วินาทีต่อ 1 ค่า และทำให้สามารถรับค่าจากเซ็นเซอร์พอร์ท Piglet ได้ และเปลี่ยนค่าของ VARIEABLE KEY ให้ตรงกับที่ได้ตั้งค่าไว้ใน Ubidots ในทุกๆอุปกรณ์ที่จะทำการติดตั้งโดยสามารถดูโค้ดได้ที่ภาคผนวก ก.

Z1 นั้นเป็นอุปกรณ์ที่ทำงานบนไอพีรุ่นที่หก จึงจำเป็นต้องใช้งานไอพีรุ่นที่หกเพื่อมอบไอพีให้กับอุปกรณ์แต่ในห้องแลปนั้นไม่สามารถใช้งานไอพีรุ่นที่หกได้ แต่สามารถงานไอพีรุ่นที่สี่แบบสาธารณะได้

ทำการติดตั้ง Border Router ด้วยคำสั่ง Make border-router.upload && make connect-router PREFIX=2001:470:19:1266::/64 ซึ่งเป็นไอพีที่ได้รับมาจากการทำ tunnel และติดตั้ง Ubidots Client ด้วยคำสั่ง make ubidots-client.upload ซึ่งจะต้องทำการปรับโค้ดในส่วนของ VARIEABLE KEY ทุกครั้งในทุกอุปกรณ์

เมื่อติดตั้งโปรแกรมเสร็จเรียบร้อยต้องรอเวลาซักครู่ เพื่อทำการแจกไอพีและอัพเดทตารางการของอุปกรณ์ด้วยวิธีการของ RPL

3.3.2 การส่งข้อมูลจาก Border Router (Z1) ไปยัง Raspberry pi

เมื่อติดตั้ง Border Router โปรแกรมจะเชื่อมต่อไปยัง Raspberry pi โดยเชื่อมต่อออกไปยังเครือข่ายและส่งข้อมูลผ่านพอร์ท USB บนอินเทอร์เฟส tun0 ซึ่งเชื่อมต่อกันระหว่าง Border Router กับ Raspberry pi โดย Border Router จะรู้ไอพีรุ่นที่หกของตนจากการกำหนด Prefix ซึ่งในกรณีนี้ Prefix จะได้มาจากการทำ Tunnel Broker เพื่อใช้งานไอพีรุ่นที่หก

3.3.3 การตั้งค่าบน Raspberry pi เพื่อส่งต่อข้อมูลไปยังเราเตอร์

บน Raspberry pi นั้นจะทำหน้าที่เป็นทางผ่านของข้อมูลจาก Zolertia Z1 ที่ติดตั้งโปรแกรม Border Router โดยก่อนอื่นนั้นต้องการทำการตั้งค่าบน Raspberry pi โดยบน Raspberry pi จะมี 3 อินเทอร์เฟสดังนี้

1.อินเทอร์เฟส eth0 ใช้สำหรับรับไอพี ที่ทางสถาบันฯทำการ NAT กับไอพีรุ่นที่สี่แบบสาธารณะ เพื่อใช้ออกสู่อินเทอร์เน็ต

2.อินเทอร์เฟสเสมือน tun0 ใช้สำหรับเชื่อมต่อเครือข่ายไอพีรุ่นที่หกบนเครือข่ายบุคคลไร้สายพลังงานต่ำของอุปกรณ์ Zolertia Z1 ที่ทำหน้าที่เป็นขอบอุปกรณ์จัดเส้นทาง เพื่อเชื่อมต่อกับอินเทอร์เฟส eth0

3.อินเทอร์เฟสเสมือน he-ipv6 ใช้สำหรับเชื่อมโยงเครือข่ายไอพีรุ่นที่หกที่วิ่งผ่านเครือข่ายไอพีรุ่นที่สี่จากเว็บไซต์ tunnelbroker.net เพื่อให้ Raspberry pi สามารถออกสู่อินเทอร์เน็ตผ่านระบบเครือข่ายไอพีรุ่นที่หกได้

จากนั้นให้ตั้งค่าตารางไอพีเพื่อให้การส่งกลุ่มข้อมูลสามารถวิ่งไปถึงปลายทางที่กำหนดได้ โดยการตั้งค่าให้ ยอมรับ ในทุกช่องทาง ทั้งการนำเข้า การส่งต่อ และการนำออกของกลุ่มข้อมูล ตั้งค่าให้อินเทอร์เฟส eth0 สามารถส่งต่อกลุ่มข้อมูลไปที่อินเทอร์เฟสเสมือน tun0 และตั้งค่าให้อินเทอร์เฟสเสมือน

tun0 ส่งต่อกลุ่มข้อมูลกลับไปที่อินเทอร์เฟส eth0 ได้เช่นกัน ตั้งค่าให้อินเทอร์เฟส eth0 ทำหน้าที่จัดหาเส้นทางให้กับเครือข่ายภายใน โดยการพิมพ์บรรทัดคำสั่งลงบนโปรแกรมคำสั่งงานดังนี้

```
iptables -I INPUT -j ACCEPT
iptables -F
iptables -X
iptables -t nat -F
iptables -t nat -X
iptables -t mangle -F
iptables -t mangle -X
iptables -P INPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -P OUTPUT ACCEPT
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
iptables -A FORWARD -i eth0 -o tun0 -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -A FORWARD -i tun0 -o eth0 -m state --state RELATED,ESTABLISHED -j ACCEPT
```

ทำการเปิดทางให้ไอพีรุ่นที่สี่และไอพีรุ่นที่หกสามารถส่งต่อกลุ่มข้อมูลได้ โดยการเข้าถึงแฟ้มข้อมูล /etc/sysctl.conf และลบเครื่องหมาย หมายเหตุ ที่อยู่ข้างหน้าบรรทัดด้านล่างนี้

```
net.ipv4.ip_forward = 1 และ
net.ipv6.conf.default.forwarding=1
```

ตั้งค่าอินเทอร์เฟส eth0 ให้เป็นค่าคงที่เพื่อให้เส้นทางของกลุ่มข้อมูลสามารถส่งได้ถูกต้อง โดยการเข้าถึงแฟ้มข้อมูล /etc/dhcpcd.conf และทำการเพิ่มบรรทัดคำสั่งด้านล่างนี้

```
interface eth0
static ip_adress=172.16.55.68/24
static routers=172.16.55.1
static domain_name_servers=172.16.10.10
```

สร้างอินเทอร์เฟสเสมือน he-ipv6 เพื่อให้ Raspberry pi สามารถใช้งานอินเทอร์เน็ตผ่านไอพีรุ่นที่หกได้ โดยการเพิ่มคำสั่งดังนี้

```
auto he-ipv6
iface he-ipv6 inet6 v4tunnel
    address 2001:470:18:126::2
    netmask 64
    endpoint 216.218.221.6
    local 172.16.55.68
    ttl 255
    gateway 2001:470:18:1266::1
```

เมื่อทำการตามขั้นตอนนี้แล้วจะทำให้ Raspberry pi สามารถเป็นทางผ่านข้อมูลให้กับ Border Router ผ่าน tun0 และสามารถส่งข้อมูลต่อได้เพราะตั้งค่าบน Raspberry pi แล้ว โดยบนอินเทอร์เฟส eth0 นั้นสามารถใช้งานไอพีสาธารณะทำให้สามารถใช้งานไอพีรุ่นที่หกได้ผ่านการทำ tunnel ผ่าน tunnel broker บนอินเทอร์เฟสซึ่งตั้งค่าไว้ที่อินเทอร์เฟส he-ipv6

3.3.4 การตั้งค่าบนเราเตอร์ เพื่อส่งข้อมูลไปยัง Tunnel Broker

จากเราเตอร์ไปยัง Tunnel Broker นั้นจะเชื่อมต่อไปยังอินเทอร์เน็ตเครือข่ายภายนอก โดยต้องตั้งค่าที่เราเตอร์ให้เปิดใช้งานสองส่วนคือ

1. โพรโตคอล 41 ไว้เพื่อเป็นการเปิดอนุญาตสำหรับการเพิ่มพื้นที่บนส่วนหัวของไอพีรุ่นที่สี่เพื่อห่อหุ้มแพ็กเก็ตไอพีรุ่นที่หกเข้าไป

2. พอร์ท ICMP เพื่อให้ทางเซิฟเวอร์สามารถตอบกลับมายัง Raspberry pi ได้

ซึ่งเมื่อทำตามทั้ง 2 อย่างนี้จะทำให้เพ็กเก็ตที่ออกไปสามารถใช้บริการของ Tunnel broker ได้และจะได้รับไอพีรุ่นที่หกมาใช้งานซึ่งสามารถใช้เชื่อมต่อกับทุกเว็บ

3.3.5 จาก Tunnel broker ไปยัง Ubidots

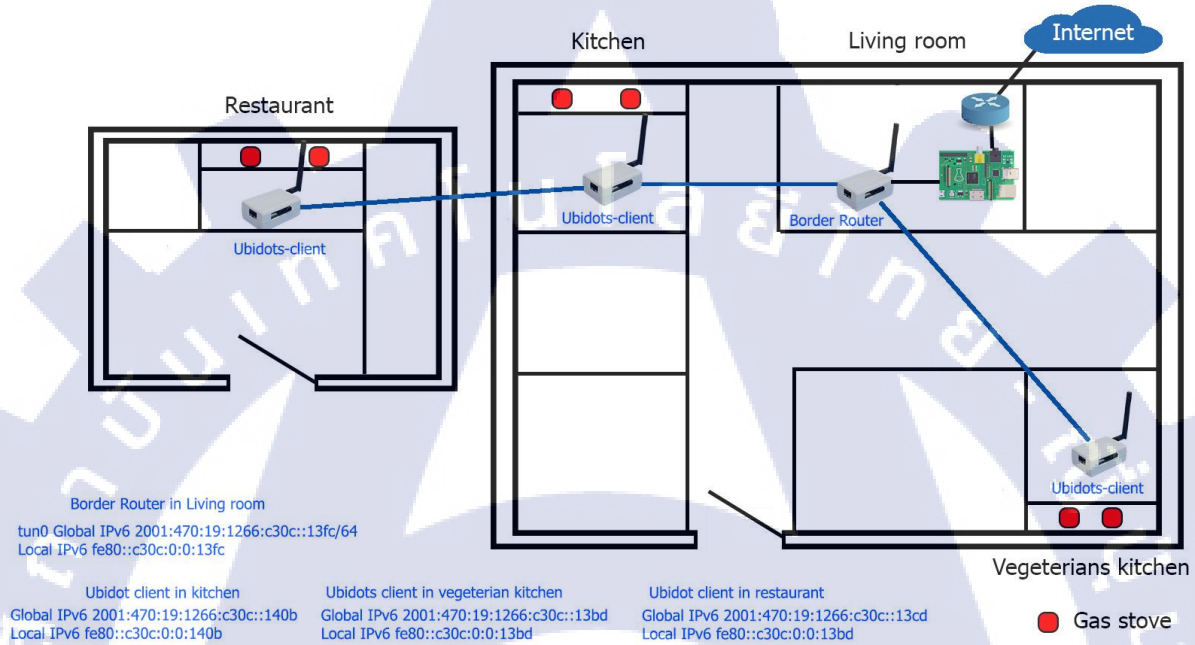
ข้อมูลจาก Tunnel Broker นั้นเมื่อเชื่อมต่อไป Ubidots จะเชื่อมต่อไปด้วยไอพีรุ่นที่หกออกมาจาก Tunnel broker ซึ่งข้อมูลของแพ็กเก็ตจาก Border Router จะถูกคลายออกในขั้นตอนนี้และส่งต่อไปด้วยไอพีรุ่นที่หก

3.3.6 จาก Ubidots แจ้งเตือนไปยังผู้ใช้งาน

บน Ubidots สามารถตั้งค่าการแจ้งเตือนผู้ใช้งานได้โดยมีหลายฟังชันก์ให้เลือกแต่ที่ง่ายที่สุดคือแจ้งผ่านข้อความหรืออีเมลโดยตั้งเงื่อนไขให้กับค่าที่ได้รับมาและเมื่อค่าเป็นไปตามเงื่อนไขจะทำการส่งข้อมูล

บทที่ 4
ผลการทดลอง การวิเคราะห์และสรุปผลต่าง ๆ

4.1 ผลการทดลอง

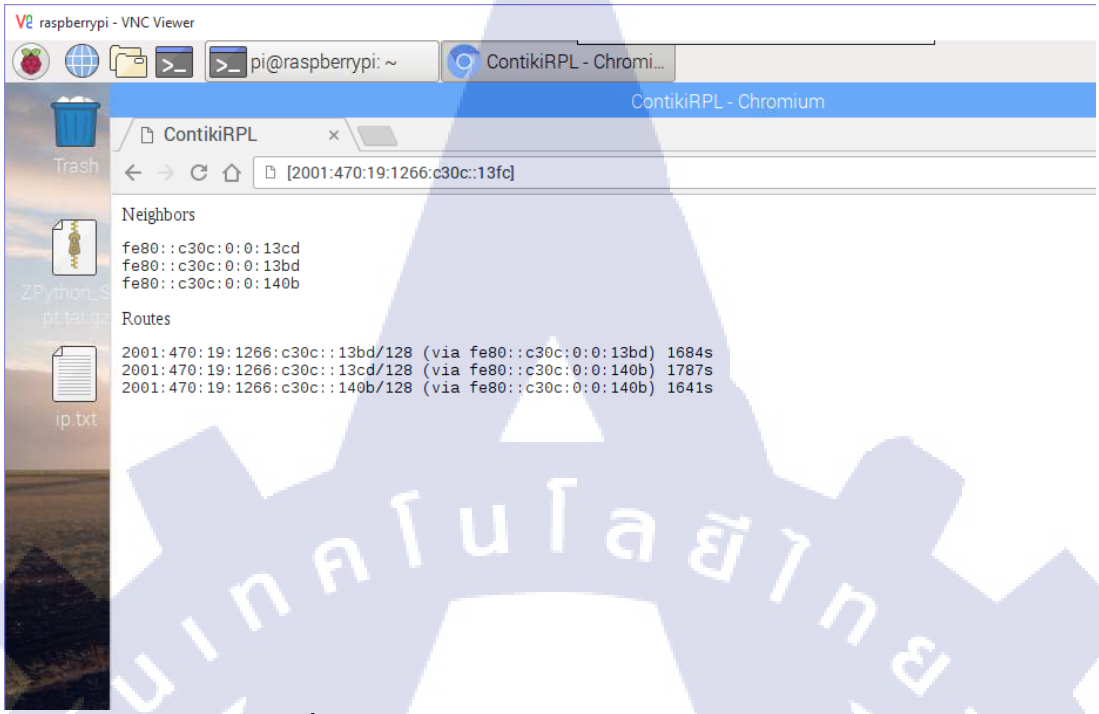


รูปที่ 4.1 กรณีศึกษาสถานการจำลองระบบแจ้งเตือนอัคคีภัยในครัวเรือน

การนำระบบแจ้งเตือนอัคคีภัยในครัวเรือนด้วยเทคโนโลยี IoT6 มาติดตั้งในบ้าน โดยจากรูป 4.1 แสดงถึงบ้านขนาดใหญ่มีหลายห้อง มีสองห้องครัวโดยห้องครัวแรกเป็นห้องครัวสำหรับอาหารมังสวิรัติ และห้องที่สองเป็นห้องครัวสำหรับอาหารทั่วไป โดยมีพื้นที่ขนาดเล็กข้างบ้านเปิดกิจการเป็นร้านอาหารขนาดเล็กซึ่งทั้งสามห้องนั้นมีการใช้งานเตาแก๊ส

ในส่วนของห้องนั่งเล่นนั้นเป็นตำแหน่งที่ใช้วางเราเตอร์และเชื่อมต่อกับ Raspberry pi 3 โดยติดตั้ง Border Router บน Z1 ในห้องนั่งเล่น

ในส่วนของห้องห้องครัวสำหรับอาหารมังสวิรัติ ห้องครัวสำหรับอาหารทั่วไป และร้านอาหารข้างๆ นำ Z1 ที่ติดตั้งโปรแกรม Ubidots Client พร้อมติดเซ็นเซอร์ไฟและแก๊สไว้



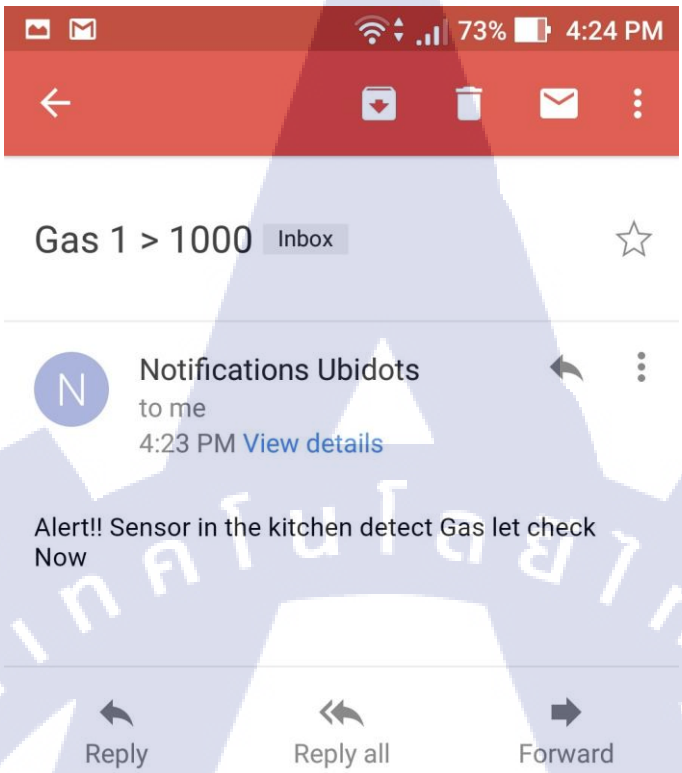
รูปที่ 4.2 ตารางการหาเส้นทางของ Border Router

โดยเมื่อวางอุปกรณ์ตามรูปภาพ 4.1 จากนั้นให้เปิดเว็บเบราว์เซอร์บน Raspberry pi จากนั้นใส่หมายเลขไอพีเข้าไปที่ช่อง URL จะพบดังรูปภาพ 4.2 แสดงถึงตารางการหาเส้นทางบน Border Router โดย Border Router จะถูกตั้งในห้องนั่งเล่นและเชื่อมต่อกับ Z1 ในห้องสำหรับครัวทั่วไปและ Z1 ในห้องครัวสำหรับอาหารมังสวิรัติได้โดยตรง โดยดูจากไอพี ::13bd และ ::140b ว่าเชื่อมต่อผ่านไอพีภายในตัว Z1 ของตัวเอง แต่สำหรับ Z1 ในร้านอาหารข้างๆบ้านนั้นเชื่อมต่อมายัง Border Router ผ่าน Z1 ในห้องครัวทั่วไป โดยดูจากไอพี ::13cd ที่เชื่อมต่อผ่าน ::140b

จากการทดลองใช้งาน Z1 เซ็นเซอร์ไฟและแก๊สสามารถทำงานร่วมกันได้ โดยสามารถแจ้งเตือนไปที่ยังผู้ใช้งานได้โดยจะแบ่งออกเป็นกรณีดังนี้

4.1.1 ห้องครัวสำหรับอาหารมังสวิรัติเกิดแก๊สรั่ว

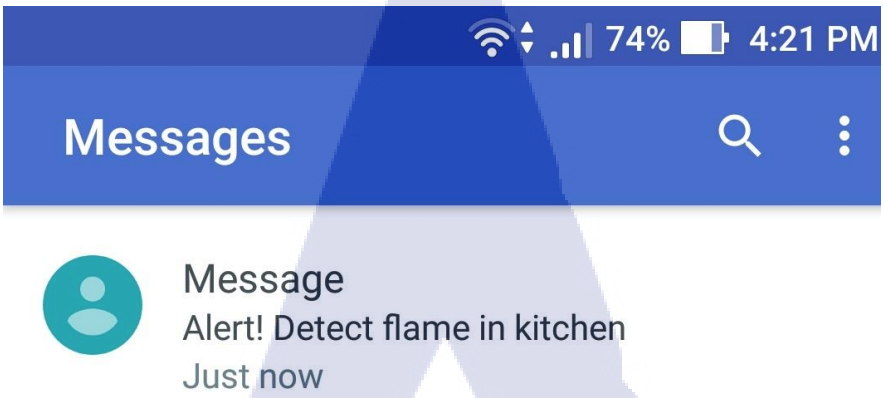
เมื่อเกิดกรณีแก๊สรั่วออกมาจากถังแก๊ส เซ็นเซอร์แก๊สในห้องครัวสำหรับอาหารมังสวิรัติจะตรวจพบแก๊สรั่วออกมาจากถังแก๊สและส่งค่าไปยัง Border Router ในห้องนั่งเล่นและส่งค่าผ่าน Raspberry pi และเราเตอร์ไปยัง Ubidots โดยค่าที่ถูกส่งไปจะถูกแสดงใน Ubidots และเมื่อค่าเกิดการเปลี่ยนแปลงซึ่งเป็นสัญญาณว่าตรวจพบแก๊สจะทำการแจ้งเตือนมายังผู้ใช้งานด้วยเมลและข้อความมายังเบอร์โทรศัพท์ที่ได้ตั้งค่าไว้ใน Ubidots



รูปที่ 4.3 เมลแจ้งเตือนจาก Ubidots

4.1.2 ห้องครัวสำหรับอาหารทั่วไปตรวจพบไฟ

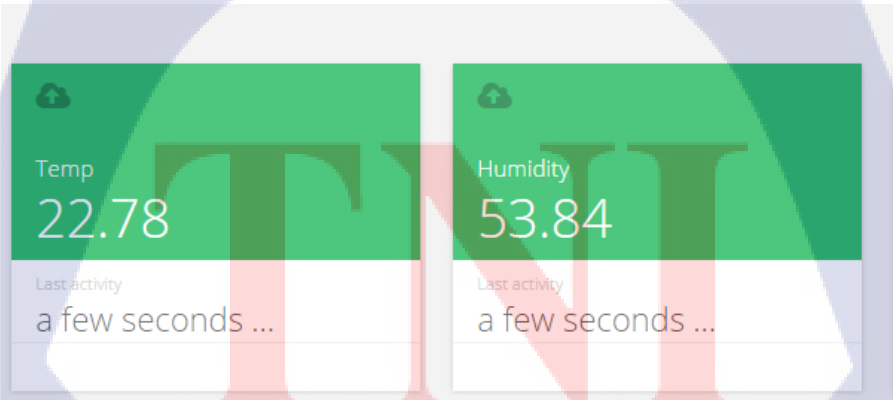
เมื่อเกิดกรณีเกิดเหตุเปลวไฟในห้องอันเนื่องมาจากเหตุผลใดๆ เซ็นเซอร์ไฟในห้องครัวสำหรับอาหารทั่วไปจะตรวจพบไฟในห้อง และจะส่งค่าไฟที่รับมาไปยัง Border Router ในห้องนั่งเล่นและส่งค่าผ่าน Raspberry pi และเราเตอร์ไปยัง Ubidots โดยค่าที่ถูกส่งไปจะถูกแสดงใน Ubidots และเมื่อค่าเกิดการเปลี่ยนแปลงซึ่งเป็นสัญญาณว่าตรวจพบไฟจะทำการแจ้งเตือนมายังผู้ใช้งานด้วยข้อความมายังเบอร์โทรศัพท์ที่ได้ตั้งค่าไว้ใน Ubidots



รูปที่ 4.4 ข้อความแจ้งเตือนจาก Ubidots

4.1.3 ตรวจสอบค่าความชื้นและอุณหภูมิในร้านอาหารข้างบ้าน

เมื่อเกิดกรณีแก๊สรั่วออกมาจากถังแก๊ส เซ็นเซอร์แก๊สในร้านอาหารข้างบ้านจะตรวจพบแก๊สรั่วออกมาจากถังแก๊สและส่งค่าไปยัง Border Router ในห้องนั่งเล่น ผ่าน Z1 ในห้องครัวสำหรับอาหารทั่วไป ซึ่ง Z1 ในห้องครัวสำหรับอาหารทั่วไปจะทำการส่งต่อข้อมูลไปให้ Border Router และส่งค่าผ่าน Raspberry pi และเราเตอร์ไปยัง Ubidots โดยค่าที่ถูกส่งไปจะถูกแสดงใน Ubidots และเมื่อค่าเกิดการเปลี่ยนแปลงซึ่งเป็นสัญญาณว่าตรวจพบแก๊สจะทำการแจ้งเตือนมายังผู้ใช้งานด้วยเมลและข้อความมายังเบอร์โทรศัพท์ที่ได้ตั้งค่าไว้ใน Ubidots



รูปที่ 4.5 การแสดงข้อมูลบน Ubidots

บทที่ 5
บทสรุปและข้อเสนอแนะ

5.1 สรุปผลการดำเนินงาน

อุปกรณ์สามารถทำงานร่วมกับเซ็นเซอร์ได้ดี สามารถส่งค่าขึ้น Ubidots ได้และแจ้งเตือนมายังสมาร์ทโฟนได้ตลอดเวลาตามวัตถุประสงค์ ซึ่งภายในห้องทดลองสามารถทำงานได้ดี แต่การจะนำไปใช้งานจริงจำเป็นต้องมีไอพีรุ่นที่สี่แบบสาธารณะในการใช้งานซึ่งการขอตามบ้านเรือนอาจจะเป็นไปได้ยาก และยังต้องเปิดพอร์ท ICMP และโพรโทคอล 41 หากต้องการทำ Tunnel จึงทำให้ในตอนนี้สามารถทำการทดลองได้เพียงภายในแลปของสถาบันเทคโนโลยีไทย-ญี่ปุ่น เท่านั้น

5.2 ปัญหา

5.2.1 ไอพีรุ่นที่สี่แบบสาธารณะ

ในห้องทดลองสามารถใช้งานไอพีรุ่นที่สี่แบบสาธารณะได้ได้แต่ในการใช้งานตามบ้านเรือนจริงนั้นยังเป็นปัญหาอยู่ซึ่งหากทางตามบ้านเรือนสามารถใช้งานไอพีรุ่นที่สี่แบบสาธารณะ หรือใช้งานไอพีรุ่นที่หกได้จะสามารถแก้ปัญหาในส่วนนี้ได้

5.2.2 Z1 ต้องเชื่อมต่อ Micro USB

Z1 มีพอร์ทเชื่อมต่อกับเซ็นเซอร์แบบ 5V และ 3.3V ซึ่งหากเชื่อมต่อกับเซ็นเซอร์ด้วยพอร์ท 5V เพื่อใช้งานจะต้องเชื่อมต่อพอร์ท Micro USB เพื่อใช้ให้พลังงานเพียงพอ ซึ่งเซ็นเซอร์แก๊สนั้นจำเป็นต้องใช้เชื่อมต่อกับพอร์ท 5V หากต่อกับพอร์ท 3.3V จะส่งค่าผิด ดังนั้นในการทดลองจึงจำเป็นต้องต่อกับ Micro USB เพื่อทดลอง

5.2.3 ปัญหาที่ Ubidots

Ubidots เป็นบริการคลาวด์แพลตฟอร์มซึ่งในบางครั้งย่อมมีการปรับปรุงแก้ไขซึ่งหากเซิฟเวอร์ทำการปิดบริการชั่วคราวจะส่งผลให้การส่งค่าขึ้นไปบน Ubidots จะหยุดชะงักไปชั่วคราว

5.3 ข้อเสนอแนะ

IoT6 ยังเป็นเทคโนโลยีที่ใหม่สามารถนำไปพัฒนาต่อไปได้เหมาะแก่การศึกษาเพื่อทดลองทำการวิจัยอย่างยิ่ง หรือนำไปใช้แก้ไขปัญหาอื่น ปัญหาในชีวิตประจำวันที่ใช้งานได้จริง

5.4 แผนการทำงานในอนาคต

5.4.1 พัฒนาคลาวด์แพลตฟอร์มของตัวเอง

ในการใช้งาน Ubidots จะมีบางข้อจำกัดในการใช้งานบางคุณสมบัติหรือพบปัญหามาจาก Ubidots ซึ่งไม่สามารถแก้ไขปัญหาได้ในทันที การสร้างคลาวด์แพลตฟอร์มของตัวเองจึงเป็นอีกหนึ่งในแผนการทำงานในอนาคต

5.4.2 เพิ่มเซ็นเซอร์

นำเซ็นเซอร์ เช่น เซ็นเซอร์แสง เซ็นเซอร์เสียง หรือเซ็นเซอร์อื่นๆ เข้ามาในระบบเพื่อเพิ่มคุณสมบัติการทำงานให้มากขึ้น

เอกสารอ้างอิง

"marcozennaro/IPv6-WSN-Book." *GitHub*. Accessed February 5, 2017. https://github.com/marcozennaro/IPv6-WSN-book.

"Zolertia/Resources." *GitHub*. Accessed February 5, 2017. https://github.com/Zolertia/Resources.

"IoT Platform | Internet of Things | Ubidots." Accessed February 5, 2017. https://ubidots.com/.

"Hurricane Electric Free IPv6 Tunnel Broker." Accessed February 5, 2017. https://tunnelbroker.net/.

"IPv6 for IoT | IoT6.eu." Accessed February 5, 2017. http://iot6.eu/ipv6_for_iot.

"AKIF." *AKIF*. Accessed February 5, 2017. https://zaidmufti.wordpress.com/.

"IoT Standards & Protocols Guide | 2017 Comparisons on Network, Wireless Comms, Security, Industrial." Accessed February 16, 2017. /internet-of-things-protocols/.

ภาคผนวก

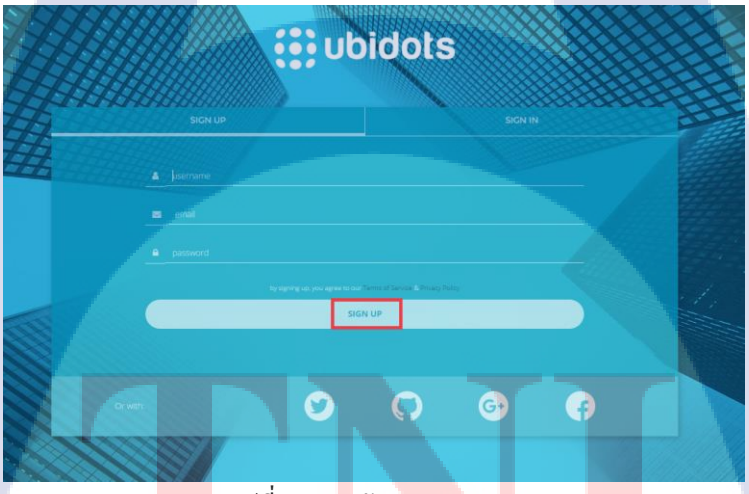
ภาคผนวก ก.

การสมัครและตั้งค่าเพื่อใช้งาน Ubidots

การสมัครใช้งาน Ubidots

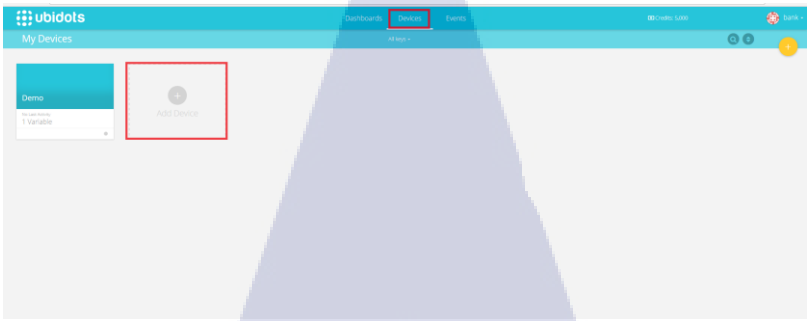
รูปที่ 1 การสมัคร Ubidots (1)

1. เปิดเว็บเบราว์เซอร์และพิมพ์ https://ubidots.com/ ในช่อง URL จากนั้นเลือกไปที่ Sign up ด้านบนขวา



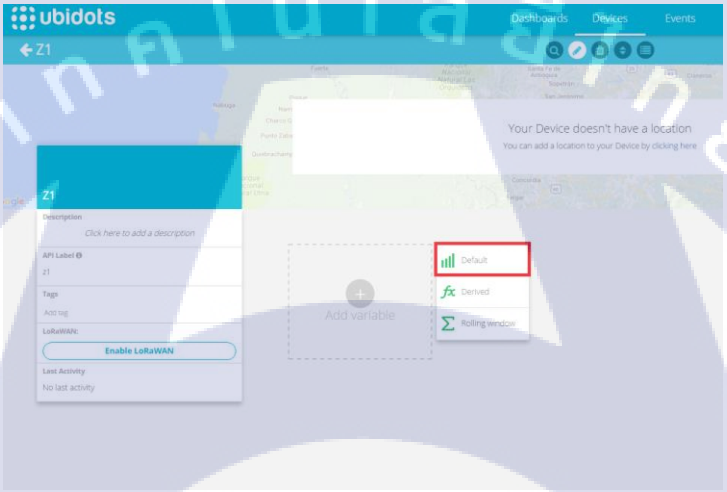
รูปที่ 2 การสมัคร Ubidots (2)

2.จะเข้ามาสู่หน้าลงทะเบียนใช้งาน ให้ใส่ชื่อผู้ใช้งาน รหัส และอีเมล จากนั้นกด Sign up



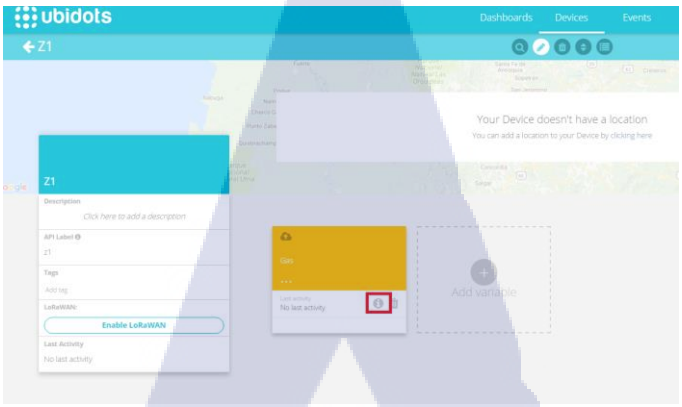
รูปที่ 3 การเพิ่มตัวแปรและอุปกรณ์ (1)

3. เลือกที่ Devices จากนั้นเลือกที่ Add Devices จากนั้นให้ใส่ชื่ออุปกรณ์และเลือกที่อุปกรณ์นั้น

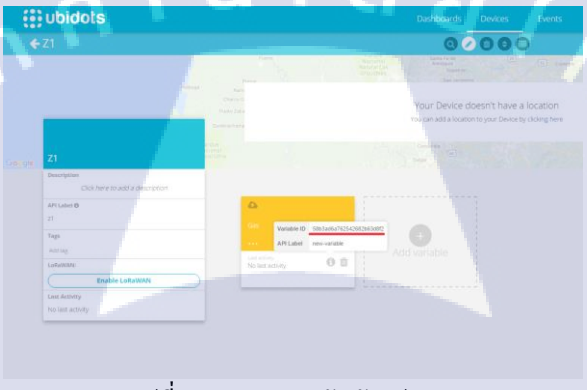


รูปที่ 4 การเพิ่มตัวแปรและอุปกรณ์ (2)

4. เมื่อเข้ามาแล้วเลือกที่ Add variable จากนั้นเลือกไปที่ Default แล้วตั้งชื่อให้อุปกรณ์นั้น



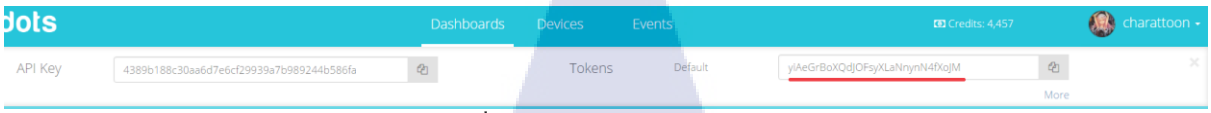
รูปที่ 5 ตรวจสอบรหัสตัวแปร (1)



รูปที่ 6 ตรวจสอบรหัสตัวแปร (2)

5.สามารถตรวจเช็ค Variable ID โดยเลือกที่ตัว I ด้านล่างขวาของ Variable จะเห็น Varia bleID แสดงขึ้นมา

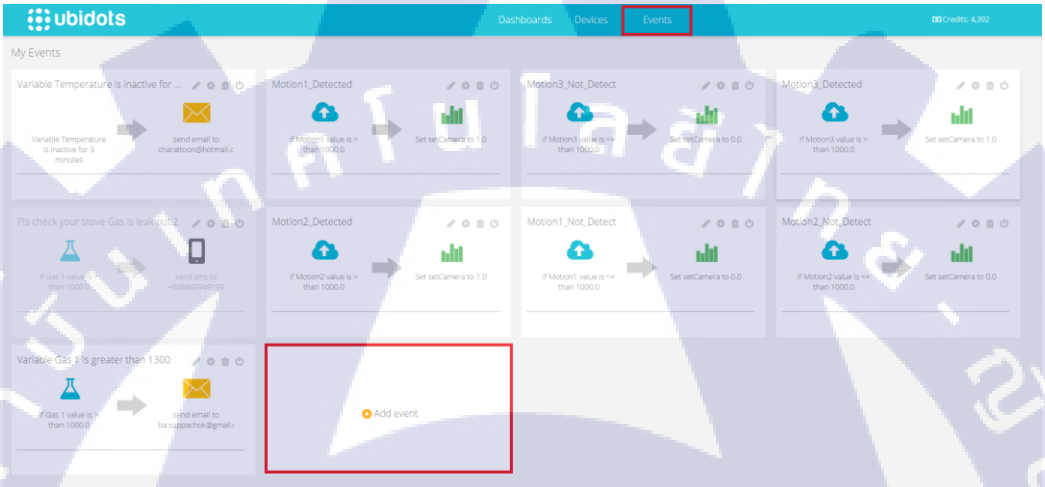
รูปที่ 7 ตรวจสอบ Token Key (1)



รูปที่ 8 ตรวจสอบ Token Key (2)

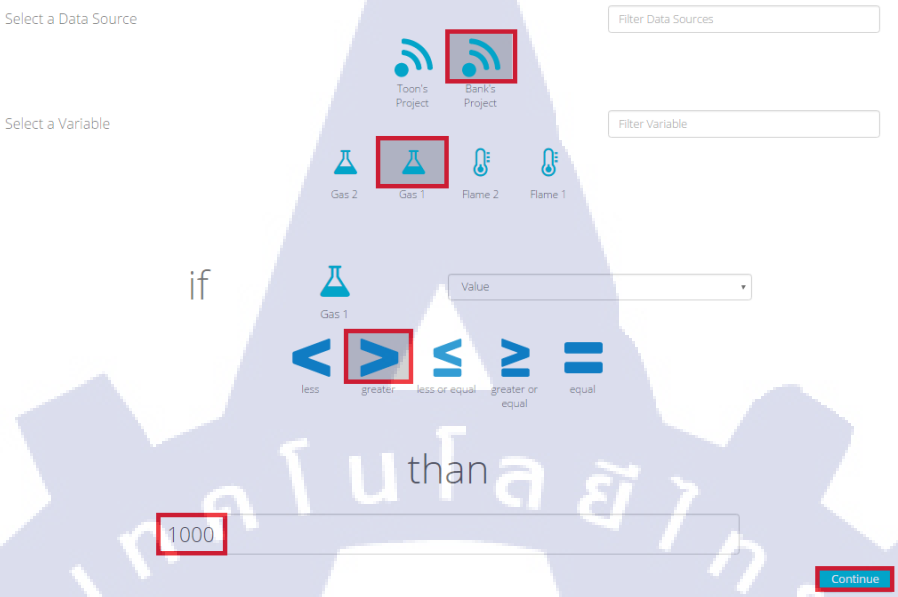
6. สามารถเช็ค Token Key โดยเลือกที่ปุ่มลงด้านบนขวาจากนั้นเลือก Token Key จะแสดงขึ้นมา

การตั้งค่าเหตุการณ์เพื่อส่งข้อความหรืออีเมล



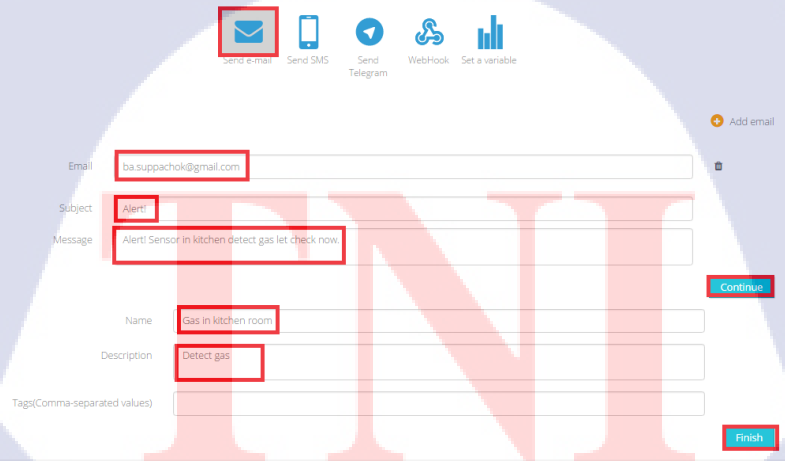
รูปที่ 9 การเพิ่มเหตุการณ์

1. ไปยังแถบ Events ด้านบนจากนั้นเลือก Add event



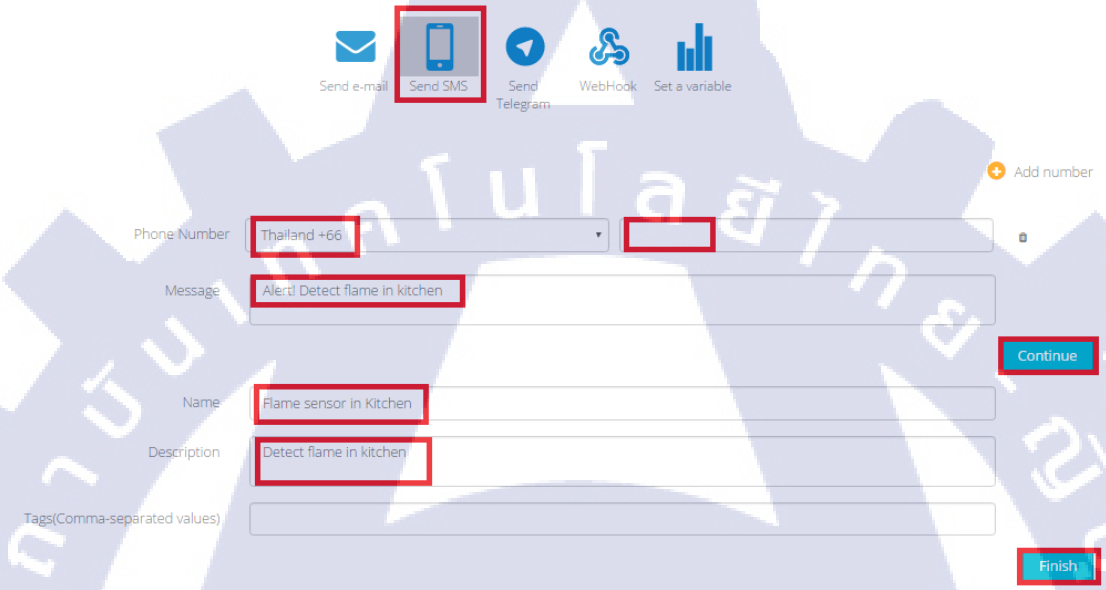
รูปที่ 10 การเพิ่มเหตุการณ์ (2)

2. เลือกอุปกรณ์ที่ต้องการจะใช้ จากนั้นเลือกไปที่ตัวแปร และเลือกเงื่อนไขจากนั้นใส่ค่าเข้าไปและกด Continue



รูปที่ 11 การส่งค่าไปยังอีเมล

3.ในส่วนของการส่งอีเมลให้เลือก Send e-mail ใส่ที่อยู่อีเมลที่ต้องการจะส่ง หัวข้อและข้อความ จากนั้นกด Continue และใส่ชื่อและรายละเอียดที่ต้องการจะแสดงบน Ubidots และกด Finish



รูปที่ 12 การส่งค่าไปยังข้อความโทรศัพท์

4.ในส่วนของการส่งข้อความให้เลือก Send sms ใส่ตำแหน่งประเทศและเบอร์โทรศัพท์ที่ต้องการจะส่งและข้อความ จากนั้นกด Continue และใส่ชื่อและรายละเอียดที่ต้องการจะแสดงบน Ubidots และกด Finish

ปรับปรุงโค้ดเพื่อส่งค่าจากอนาล็อกเซ็นเซอร์ไปยัง Ubidots

ในการใช้งาน Ubidots ให้ส่งค่านั้นได้ทำการปรับปรุงโค้ด 2 ส่วน คือ Ubidots-client.c จะปรับปรุงโค้ดให้ส่งค่าจากเดิมที่ส่งได้เพียงเซ็นเซอร์จากพอร์ท Ziglet ให้ส่งจาก Phidget ได้ และ project-conf.h ให้เปลี่ยนในส่วนของ VARIABLE_KEY ให้ตรงกับตัวแปรที่ตั้งไว้บน Ubidots จะสามารถส่งค่าได้ตรง

โค้ดดั่งเดิมของ Ubidots

Ubidots-client.c

```
#include "contiki-net.h"
#include "dev/leds.h"
#include "rpl.h"
#include "ubidots.h"
#include "dev/button-sensor.h"
#include "dev/sht25.h"
#include "ip64-addr.h"
#include <stdio.h>
/*---------------------------------------------------------------------------*/
PROCESS(ubidots_example_process, "Ubidots example");
AUTOSTART_PROCESSES(&ubidots_example_process);
/*---------------------------------------------------------------------------*/
static const char *headers[] = {
 "Vary",
 NULL
};
/*---------------------------------------------------------------------------*/
static void
post_collection(void)
{
 uint16_t aux;
 char variable_buffer[VARIABLE_BUF_LEN];
 if(ubidots_prepare_post(NULL) == UBIDOTS_ERROR) {
  printf("post_collection: ubidots_prepare_post failed\n");
   return;
 }
 aux = sht25.value(SHT25_VAL_TEMP);
 memset(variable_buffer, 0, VARIABLE_BUF_LEN);
 snprintf(variable_buffer, VARIABLE_BUF_LEN, "%u", aux);
 if(ubidots_enqueue_value(VARKEY_VARIABLE_ONE, variable_buffer) == UBIDOTS_ERROR) {
  printf("post_collection: ubidots_prepare_post failed\n");
 }
 aux = sht25.value(SHT25_VAL_HUM);
 memset(variable_buffer, 0, VARIABLE_BUF_LEN);
 snprintf(variable_buffer, VARIABLE_BUF_LEN, "%u", aux);
 if(ubidots_enqueue_value(VARKEY_VARIABLE_TWO, variable_buffer) == UBIDOTS_ERROR) {
  printf("post_collection: ubidots_prepare_post failed\n");
 }
 if(ubidots_post() == UBIDOTS_ERROR) {
  printf("post_collection: ubidots_prepare_post failed\n");
 }
```

```
}
/*---------------------------------------------------------------------------*/
static void
print_reply(ubidots_reply_part_t *r)
{
 switch(r->type) {
 case UBIDOTS_REPLY_TYPE_HTTP_STATUS:
  printf("HTTP Status: %ld\n", *((long int *)r->content));
  break;
 case UBIDOTS_REPLY_TYPE_HTTP_HEADER:
  printf("H: '%s'\n", (char *)r->content);
  break;
 case UBIDOTS_REPLY_TYPE_PAYLOAD:
  printf("P: '%s'\n", (char *)r->content);
  break;
 default:
  printf("Unknown reply type\n");
  break;
 }
}
/*---------------------------------------------------------------------------*/
PROCESS_THREAD(ubidots_example_process, ev, data)
{
 static struct etimer et;
 uip_ip4addr_t ip4addr;
 uip_ip6addr_t ip6addr;
 PROCESS_BEGIN();
 SENSORS_ACTIVATE(sht25);
 ubidots_init(&ubidots_example_process, headers);
 uip_ipaddr(&ip4addr, 8, 8, 8, 8);
 ip64_addr_4to6(&ip4addr, &ip6addr);
 uip_nameserver_update(&ip6addr, UIP_NAMESERVER_INFINITE_LIFETIME);
 while(1) {
  PROCESS_YIELD();
  if(ev == ubidots_event_established ||
    (ev == PROCESS_EVENT_TIMER && data == &et)) {
   leds_on(LEDS_GREEN);
   post_collection();
  } else if(ev == ubidots_event_post_sent) {
   leds_off(LEDS_GREEN);
   etimer_set(&et, POST_PERIOD);
  } else if(ev == ubidots_event_post_reply_received) {
```

```
    print_reply((ubidots_reply_part_t *)data);
   }
  }
  PROCESS_END();
}
/*---------------------------------------------------------------------------*/
```

project-conf.c

```
/*---------------------------------------------------------------------------*/
#ifndef PROJECT_CONF_H_
#define PROJECT_CONF_H_
/*---------------------------------------------------------------------------*/
/* User configuration */
#define POST_PERIOD                    (CLOCK_SECOND * 40)
#define VARIABLE_BUF_LEN               16
#define UBIDOTS_CONF_AUTH_TOKEN        ""
#define VARKEY_VARIABLE_ONE            ""
#define VARKEY_VARIABLE_TWO            ""
#define UBIDOTS_CONF_IN_BUFFER_SIZE    64
/*---------------------------------------------------------------------------*/
/* IPv6 address of things.ubidots.com is "2607:f0d0:2101:39::2", leave
 * commented to resolve the host name.  The NAT64 address is "::ffff:3217:7c44"
 */
#define UBIDOTS_CONF_REMOTE_HOST       "::ffff:3217:7c44"
/*---------------------------------------------------------------------------*/
#undef  NETSTACK_CONF_RADIO
#define NETSTACK_CONF_RADIO            cc2538_rf_driver
#define ANTENNA_SW_SELECT_DEF_CONF     ANTENNA_SW_SELECT_2_4GHZ
#define NETSTACK_CONF_RDC              nullrdc_driver
#undef IEEE802154_CONF_PANID
#define IEEE802154_CONF_PANID          0xABCD
/*---------------------------------------------------------------------------*/
/* The following are Z1 specific */
#undef RF_CHANNEL
#define RF_CHANNEL                     26
#undef CC2420_CONF_CHANNEL
#define CC2420_CONF_CHANNEL            26
/* The following are Zoul (RE-Mote, etc) specific */
#undef CC2538_RF_CONF_CHANNEL
#define CC2538_RF_CONF_CHANNEL         26
```

```
/*---------------------------------------------------------------------------*/
#define RESOLV_CONF_SUPPORTS_MDNS       0
#define UIP_CONF_MAX_ROUTES             3
#define NBR_TABLE_CONF_MAX_NEIGHBORS    3
/*---------------------------------------------------------------------------*/
#endif /* PROJECT_CONF_H_ */
/*---------------------------------------------------------------------------*/
```

โค้ดที่ถูกปรับปรุงของ Ubidots client โดยมี 2 ไฟล์

ubidots-test.c

```
/*---------------------------------------------------------------------------*/
#include "contiki-net.h"
#include "dev/leds.h"
#include "rpl.h"
#include "ubidots.h"
#include "dev/button-sensor.h"
#include "dev/z1-phidgets.h"
#include "ip64-addr.h"
#include <stdio.h>
/*---------------------------------------------------------------------------*/
PROCESS(ubidots_example_process, "Ubidots example");
AUTOSTART_PROCESSES(&ubidots_example_process);
/*---------------------------------------------------------------------------*/
static const char *headers[] = {
 "Vary",
 NULL
};
/*---------------------------------------------------------------------------*/
static void
post_collection(void)
{
 uint16_t aux;
 char variable_buffer[VARIABLE_BUF_LEN];
 if(ubidots_prepare_post(NULL) == UBIDOTS_ERROR) {
  printf("post_collection: ubidots_prepare_post failed\n");
  return;
 }
 /*
  aux = phidgets.value(PHIDGET5V_1);
```

```
  printf("Phidget 5V 1:%d\n", aux);
  memset(variable_buffer, 0, VARIABLE_BUF_LEN);
  snprintf(variable_buffer, VARIABLE_BUF_LEN, "%u", aux);

  if(ubidots_enqueue_value(VARKEY_VARIABLE_ONE, variable_buffer) == UBIDOTS_ERROR) {
   printf("post_collection: ubidots_prepare_post failed\n");
  }
*/
  aux = phidgets.value(PHIDGET3V_2);
  printf("Phidget 3V 2:%d\n", aux);
  memset(variable_buffer, 0, VARIABLE_BUF_LEN);
  snprintf(variable_buffer, VARIABLE_BUF_LEN, "%u", aux);
  if(ubidots_enqueue_value(VARKEY_VARIABLE_TWO, variable_buffer) == UBIDOTS_ERROR) {
   printf("post_collection: ubidots_prepare_post failed\n");
  }
  if(ubidots_post() == UBIDOTS_ERROR) {
   printf("post_collection: ubidots_prepare_post failed\n");
  }
}
/*---------------------------------------------------------------------------*/
static void
print_reply(ubidots_reply_part_t *r)
{
  switch(r->type) {
  case UBIDOTS_REPLY_TYPE_HTTP_STATUS:
    printf("HTTP Status: %ld\n", *((long int *)r->content));
    break;
  case UBIDOTS_REPLY_TYPE_HTTP_HEADER:
    printf("H: '%s'\n", (char *)r->content);
    break;
  case UBIDOTS_REPLY_TYPE_PAYLOAD:
    printf("P: '%s'\n", (char *)r->content);
    break;
  default:
    printf("Unknown reply type\n");
    break;
  }
}
/*---------------------------------------------------------------------------*/
PROCESS_THREAD(ubidots_example_process, ev, data)
{
  static struct etimer et;
```

```
uip_ip4addr_t ip4addr;
uip_ip6addr_t ip6addr;
PROCESS_BEGIN();
SENSORS_ACTIVATE(phidgets);
ubidots_init(&ubidots_example_process, headers);
uip_ipaddr(&ip4addr, 8, 8, 8, 8);
ip64_addr_4to6(&ip4addr, &ip6addr);
uip_nameserver_update(&ip6addr, UIP_NAMESERVER_INFINITE_LIFETIME);
while(1) {
 PROCESS_YIELD();
 if(ev == ubidots_event_established ||
    (ev == PROCESS_EVENT_TIMER && data == &et)) {
  leds_on(LEDS_GREEN);
  post_collection();
 } else if(ev == ubidots_event_post_sent) {
  leds_off(LEDS_GREEN);
  etimer_set(&et, POST_PERIOD);
 } else if(ev == ubidots_event_post_reply_received) {
  print_reply((ubidots_reply_part_t *)data);
 }
}
PROCESS_END();
}
/*---------------------------------------------------------------------------*/
```

project-conf.h

```
/*---------------------------------------------------------------------------*/
#ifndef PROJECT_CONF_H_
#define PROJECT_CONF_H_
/*---------------------------------------------------------------------------*/
/* User configuration */
#define POST_PERIOD                 (CLOCK_SECOND)
#define VARIABLE_BUF_LEN            16
#define UBIDOTS_CONF_AUTH_TOKEN     "yiAeGrBoXQdJOFsyXLaNnynN4fXoxx"
//1
//#define VARKEY_VARIABLE_ONE           "5896c7887625424b92a0b5xx"
//#define VARKEY_VARIABLE_TWO           "5896c7777625424b94a961xx"
//2
//#define VARKEY_VARIABLE_ONE           "5896c78f7625424b94a962xx"
//#define VARKEY_VARIABLE_TWO           "5896c7837625424b902b29xx"
```

```
#define UBIDOTS_CONF_IN_BUFFER_SIZE     64
/*---------------------------------------------------------------------------*/
/* IPv6 address of things.ubidots.com is "2607:f0d0:2101:39::2", leave
 * commented to resolve the host name.  The NAT64 address is "::ffff:3217:7c44"
 */
#define UBIDOTS_CONF_REMOTE_HOST        "2607:f0d0:2101:39::2"
/*---------------------------------------------------------------------------*/
#undef  NETSTACK_CONF_RADIO
#define NETSTACK_CONF_RADIO             cc2538_rf_driver
#define ANTENNA_SW_SELECT_DEF_CONF      ANTENNA_SW_SELECT_2_4GHZ
#define NETSTACK_CONF_RDC               nullrdc_driver
#undef IEEE802154_CONF_PANID
#define IEEE802154_CONF_PANID           0xABCD
/*---------------------------------------------------------------------------*/
/* The following are Z1 specific */
#undef RF_CHANNEL
#define RF_CHANNEL                  26
#undef CC2420_CONF_CHANNEL
#define CC2420_CONF_CHANNEL             26
/* The following are Zoul (RE-Mote, etc) specific */
#undef CC2538_RF_CONF_CHANNEL
#define CC2538_RF_CONF_CHANNEL          26
/*---------------------------------------------------------------------------*/
#define RESOLV_CONF_SUPPORTS_MDNS       0
#define UIP_CONF_MAX_ROUTES             3
#define NBR_TABLE_CONF_MAX_NEIGHBORS    3
/*---------------------------------------------------------------------------*/
#endif /* PROJECT_CONF_H_ */
/*---------------------------------------------------------------------------*/
```

ประวัติผู้จัดทำโครงงาน

ชื่อ – สกุล นายศุภโชค เกียรติกิติกูล

วัน เดือน ปีเกิด 29 เมษายน พ.ศ. 2538

ประวัติการศึกษา

ระดับประถมศึกษา โรงเรียนพระหฤทัยนนทบุรี

ระดับมัธยมศึกษา โรงเรียนหอวัง

ระดับอุดมศึกษา สถาบันเทคโนโลยีไทย-ญี่ปุ่น

ทุนการศึกษา ทุนการศึกษาประเภทที่ 3 ของสถาบันเทคโนโลยีไทยญี่ปุ่น
ทุน JASSO เพื่อไปวิจัยที่ National Institute of Technology, Tsuruoka College
เป็นเวลา 3 เดือน ปีการศึกษา 2559

ประวัติการฝึกอบรม - ไม่มี -

ผลงานที่ได้รับการตีพิมพ์ - ไม่มี -