อุปกรณ์แจ้งเตือนอัตโนมัติผ่าน Line , SMS และ E-Mail

ณพวุฒิ โพธิ์หอม

สารนิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์และโทรคมนาคม
วิทยาลัยนวัตกรรมด้านเทคโนโลยีและวิศวกรรมศาสตร์
มหาวิทยาลัยธุรกิจบัณฑิตย์
พ.ศ.2561

Automatically Warning Device via Line SMS and E-Mail

Noppavut Pohom

A Thematic Paper Submitted in Partial Fulfillment of the Requirements

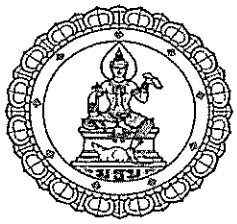
for the Degree of Master of Master of Engineering

Department of Computer and Telecommunication Engineering

College of Innovative Technology And Engineering

Dhurakij Pundit University

2018



ใบรับรองสารนิพนธ์

วิทยาลัยนวัตกรรมด้านเทคโนโลยีและวิศวกรรมศาสตร์

มหาวิทยาลัยธุรกิจบัณฑิตย์

ปริญญา วิศวกรรมศาสตรมหาบัณฑิต

หัวข้อสารนิพนธ์ อุปกรณ์แจ้งเตือนอัตโนมัติผ่าน Line, SMS และ E-Mail

เสนอโดย นายณพวุฒิ โพธิ์หอม

สาขาวิชา วิศวกรรมคอมพิวเตอร์และโทรคมนาคม

อาจารย์ที่ปรึกษาสารนิพนธ์ อาจารย์ ดร.ชัยพร เขมะภาตะพันธ์

ได้พิจารณาเห็นชอบโดยคณะกรรมการสอบสารนิพนธ์แล้ว

.....ประธานกรรมการ

(อาจารย์ ดร.ประศาสน์ จันทราทิพย์)

.....กรรมการและอาจารย์ที่ปรึกษาสารนิพนธ์

(อาจารย์ ดร.ชัยพร เขมะภาตะพันธ์)

.....กรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.ณรงค์เดช กีรติพรานนท์)

วิทยาลัยนวัตกรรมด้านเทคโนโลยีและวิศวกรรมศาสตร์รับรองแล้ว

.....คณบดีวิทยาลัยนวัตกรรมด้านเทคโนโลยีและวิศวกรรมศาสตร์

(ผู้ช่วยศาสตราจารย์ ดร.ณรงค์เดช กีรติพรานนท์)

วันที่ ...24..... เดือน ...เมษายน..... พ.ศ. ...2561...

หัวข้อสารนิพนธ์	อุปกรณ์แจ้งเตือนอัตโนมัติผ่าน Line , SMS และ E-Mail
ชื่อผู้เขียน	ณพวุฒิ โพธิ์หอม
อาจารย์ที่ปรึกษา	อาจารย์ ดร. ชัยพร เขมะภาตะพันธ์
สาขาวิชา	วิศวกรรมคอมพิวเตอร์และโทรคมนาคม
ปีการศึกษา	2560

บทคัดย่อ

การวิจัยครั้งนี้ มีวัตถุประสงค์เพื่อพัฒนาอุปกรณ์แจ้งเตือนผ่าน Line SMS และ E-Mail โดยใช้ RaspberryPi ร่วมกับอุปกรณ์เซ็นเซอร์ต่างๆ พัฒนาด้วยภาษา Python และ Visual Basic.NET เมื่อระบบที่พัฒนาขึ้นสามารถตรวจจับสัญญาณต่างๆได้จากเซ็นเซอร์แล้วจะทำการส่งข้อความแจ้งเตือนไปยังเซิร์ฟเวอร์ซึ่งมีระบบประมวลผลเพื่อวิเคราะห์และแยกแยะประเภทการแจ้งเตือนก่อนตัดสินใจแจ้งเตือนผ่านช่องทางต่างๆ ตามกลุ่มผู้รับที่กำหนดไว้ก่อนหน้าได้

จากการทดลองจะพบว่าระบบสามารถส่งข้อมูลแจ้งเตือนไปยังกลุ่มผู้รับปลายทางได้ครบถ้วนทั้ง 3 ช่องทางตามที่ได้ตั้งค่าไว้ โดยพบว่าการส่งผ่าน Line มีความรวดเร็วมากที่สุด อย่างไรก็ตาม SMS สามารถเข้าถึงผู้รับที่ไม่ได้เชื่อมต่ออินเทอร์เน็ตได้ ในขณะที่การแจ้งเตือนผ่าน E-Mail มีการแสดงรายละเอียดข้อมูลได้มากกว่าวิธีอื่นๆ

Thematic Paper Title	Automatically Warning Device via Line SMS and E-Mail
Author	Noppavut Pohom
Thematic Paper Advisor	Dr. Chaiyaporn Khemapatapan
Department	Computer and Telecommunication Engineering
Academic Year	2017

ABSTRACT

This research aimed to develop automatically warning device via Line, SMS and E-Mail using RaspberryPi working with sensors which is coded by Python and Visual Basic.NET. Developed the system can detect a signal from sensors, a warnning message is directly sent to a computer server which runs a computer program, Analyzer, to analyze the warnning message and decide to send the message via Line and/or SMS and/or E-Mail to a group of user which is predefined.

From testing results, the system can send warning messages via Line, SMS and E-Mail. All users can completely receive all messages. Morever, warning via Line is fastest while SMS users can receive the messages without connection to Internet. However, warning via E-Mail can provide more detailed information.

กิตติกรรมประกาศ

การศึกษาค้วาวิจัยเรื่อง “อุปกรณ์แจ้งเตือนอัตโนมัติผ่าน Line , SMS และ E-Mail” ครั้งนี้สำเร็จลุล่วงไปด้วยดี โดยได้รับความช่วยเหลือและสนับสนุนจากหลายๆ ท่าน โดยเฉพาะอย่างยิ่ง อาจารย์ ดร. ชัยพร เขมะภาตะพันธ์ ซึ่งเป็นอาจารย์ที่ปรึกษาได้ให้คำแนะนำในการคัดเลือกเรื่องที่จัดทำการศึกษาค้นคว้า ให้ความรู้ด้านวิชาการ ด้านเทคนิค และข้อคิดต่างๆ ที่เป็นประโยชน์ต่อการศึกษาค้นคว้าในครั้งนี้ รวมทั้ง ผศ.ดร.มัชฌิกา อ่องแตง ที่ให้คำแนะนำในการแก้ไขปรับปรุง ผู้ทำการศึกษาซาบซึ้งในความกรุณาของท่านเป็นอย่างยิ่ง

การดำเนินการศึกษาครั้งนี้จะสำเร็จลงไม่ได้ หากขาดความร่วมมือจากบริษัท แคท เทเลคอม จำกัด มหาชน สายงานสื่อสารไร้สาย ที่ได้ให้ทดลองใช้บริการของ SMS Gateway เพื่อทดลองส่ง SMS ผ่านเครือข่ายให้บริการ my และระบบ Line Notify , Gmail ที่ให้ใช้บริการต่างๆ ด้วยเช่นกัน

ในสุดท้ายนี้ ต้องขอระลึกถึงความกรุณาของคณาจารย์ทุกท่านที่เป็นผู้ชี้แนะแนวทางการศึกษาในตอนต้น รวมทั้งให้คำแนะนำในการค้นคว้าหาข้อมูลและวิธีการในการดำเนินงานวิจัย พร้อมทั้งวิธีทดสอบต่างๆ ที่เป็นประโยชน์ และขอขอบพระคุณทุกท่านที่มิได้กล่าวนามมา ณ ที่นี้ ที่มีส่วนช่วยเหลือและเป็นกำลังใจรวมทั้งให้การสนับสนุนเป็นอย่างดีจนทำให้การศึกษาครั้งนี้สำเร็จลุล่วงด้วยดี

ณพวุฒิ โพธิ์หอม

สารบัญ

.

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญภาพ

สารบัญภาพ (ต่อ)

ภาพที่ **หน้า**

สารบัญภาพ (ต่อ)

ภาพที่ **หน้า**

รายการสัญลักษณ์

mAh	เป็นหน่วยของแบตเตอรี่ อ่านว่า มิลลิแอมป์
Mhz	เป็นหน่วยของความถี่ อ่านว่า แมกกะเฮิรตซ์
A	เป็นหน่วยวัดกระแสไฟฟ้า อ่านว่า แอมแปร์
µs	เป็นหน่วยพื้นฐานที่ใช้วัดการนำไฟฟ้า อ่านว่า โมโครซีเมนส์

ประมวลศัพท์และคำย่อ

IOT	ย่อมาจาก Internet of Things (IoT)
LoRaWAN หรือ NB-IOT	ย่อมาจาก Narrow Band Internet of Things
API	ย่อมาจาก Application Programming Interface

บทที่ 1
บทนำ

ปัจจุบันเทคโนโลยีต่างๆ พัฒนาไปมากขึ้น ทำให้มีการพัฒนาอุปกรณ์ที่เรียกว่า IOT (internet of things) ขึ้นมา เพื่อนำมาใช้งานเชื่อมต่อให้อุปกรณ์ต่างๆ สามารถใช้งานได้ไร้ขอบเขตมากขึ้น เช่น การนำ IOT ไปติดอุปกรณ์ Sensor ต่างๆ , การนำ IOT ไปติดกับอุปกรณ์ Smart Home เพราะข้อดีของอุปกรณ์ IOT คือ การเชื่อมต่อเครือข่ายอยู่ตลอดเวลา , ใช้พลังงานต่ำ , มีขนาดเล็ก และไม่ต้องการประสิทธิภาพที่สูง ทางผู้จัดทำจึงได้คิดที่จะนำอุปกรณ์ IOT มาพัฒนาเพื่อแก้ไขปัญหากับการทำงานได้

1.1 ที่มาและความสำคัญของปัญหา

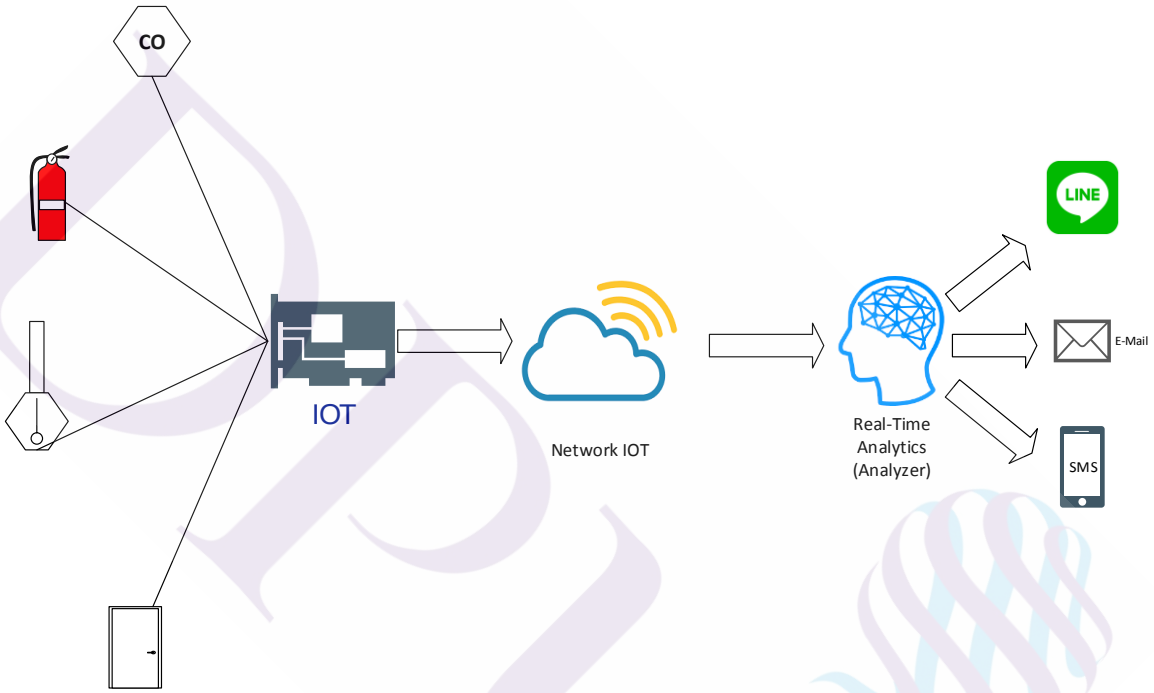
เหตุผลที่ทางผู้จัดทำได้คิดการพัฒนาอุปกรณ์ IOT มาเพื่อแก้ไขกับปัญหากับงานปัจจุบันนั้น เพราะว่าได้เล็งเห็นถึงระบบสิ่งอำนวยความสะดวกพื้นฐานที่ใช้งานอยู่นั้น ไม่สามารถตอบโจทย์กับงานปัจจุบันได้ เช่น ระบบวัดอุณหภูมิในห้อง Server ที่มีตัวเลขดิจิตอลโชว์อยู่กลางห้อง , ระบบตรวจจับควันที่จะมี Alarm เสียงขึ้น ฯลฯ ซึ่งระบบดังกล่าวต้องใช้คนเดินเข้าไปตรวจสอบดู และถ้าไม่มีเจ้าหน้าที่ on-site อยู่จะไม่มีใครที่จะรับรู้ Alarm ได้เลย และที่สำคัญคือมีข้อจำกัดด้านสถานที่ติดตั้งไม่สามารถ integrate ระบบ Sensor ต่างๆ กับ ระบบเดิมที่ติดตั้งอยู่แล้วได้เลย ทำให้ผู้จัดทำจึงคิดโจทย์ที่ว่า “จะหาอุปกรณ์ที่ตรวจจับ Sensor ต่างๆ ที่ติดตั้งได้ง่าย ขนาดเล็ก , ประหยัดพลังงาน และพัฒนาต่อยอดได้ง่าย” จึงเป็นที่มาที่ทางผู้จัดทำจึงได้เลือกหยิบอุปกรณ์ IOT มาใช้ในการพัฒนานี้ร่วมกับการทำ Real-Time Analytics (Kaushik Pal , July 20, 2016. The Advantages of Real-Time Analytics for Enterprise. Presented by Sponsor: Bloor Group. Retrieved from https://www.techopedia.com/2/31433/trends/big-data/advantages-of-real-time-analytics-for-enterprise)

Real-Time Analytics คือการที่หน่วยงานๆ หนึ่งสามารถที่จะนำข้อมูลทั้งหมดในองค์กร มาวิเคราะห์ และจัดการกับข้อมูลต่างๆ เหล่านั้นให้สามารถเพิ่มประสิทธิภาพในการทำงานได้อย่างทันท่วงที โดยที่ไม่จำเป็นต้องใช้ทรัพยากรมนุษย์ ในการดำเนินการ ผู้จัดทำจึงได้พัฒนาระบบ Analyzer โดยอิงรูปแบบการทำงานแบบ Real-Time Analytics ที่จะนำข้อมูลจาก Sensor ต่างๆ รับเข้ามาที่ระบบ Analyzer และทำหน้าที่วิเคราะห์ จัดการ บริหาร เพื่อสร้างเงื่อนไขในการทำงานต่างๆ ให้เป็นประโยชน์ต่อหน่วยงาน สามารถติดตามการเกิดเหตุต่างๆ ได้อย่างทันถ่วงที และทำให้การตัดสินใจในการแก้ไขปัญหาแม่นยำขึ้น พร้อมทั้งสามารถกำหนดผู้ที่สามารถร่วมในการทำงานได้หลายคน โดยระบบที่พัฒนาขึ้นนั้นจะต้องมีรูปแบบในการทำงานดังนี้

1. การสร้างระบบแจ้งเตือนอุปกรณ์จากระบบเดิมจำเป็นต้องเข้าไปเพิ่มอุปกรณ์หรือแก้ไขการตั้งค่าในการส่งข้อมูลโดยใช้การส่งผ่านจากภายในของอุปกรณ์เองซึ่งต้องมีการ Login ระบบเข้าไป ทำให้มีความเสี่ยงอาจมีผลกระทบกับระบบเดิม
2. เมื่อเกิดปัญหาผู้ที่รับผิดชอบสามารถรับรู้ได้ทันท่วงที
3. ผู้รับการแจ้งเตือนแต่ละคนมีช่องทางในการรับข้อมูลไม่เหมือนกัน
4. สามารถติดตามหรือค้นหาประวัติเมื่อเกิดเหตุได้

ประเด็นสำคัญอีกหนึ่งอย่างที่ผู้จัดทำเลือกใช้ IOT คือช่องทางในการติดต่อสื่อสารทั้งในส่วนของโครงสร้างพื้นฐานและการติดต่อสื่อสารกับผู้ใช้งานต้องสามารถมีความยืดหยุ่น ในส่วนของโครงสร้างพื้นฐานคือ ระบบเครือข่ายที่ผู้ใช้เลือกที่จะใช้งานอุปกรณ์ IOT สามารถเลือกการเชื่อมต่อได้หลากหลายทางมาก เช่น 2G , 3G , 4G หรือ Ethernet แต่ก็ไม่แน่ว่าในอนาคตถ้ามีโครงสร้างพื้นฐานที่เป็น Narrow Band ของ IOT ขึ้นมาโดยเฉพาะ เช่น LoRaWAN หรือ NB-IOT ขึ้นมาก็สามารถพัฒนาต่อยอดไปใช้โครงข่ายดังกล่าวได้เมื่อพร้อมใช้งานโดยเพิ่มเพียง Module Communication Board ขึ้นมาเท่านั้น ส่วนด้านการติดต่อสื่อสารกับผู้ใช้งาน ผู้จัดทำได้เล็งเห็น เครือข่ายสังคมออนไลน์ ที่เป็นที่นิยมในประเทศไทย เช่น Line Group เป็นช่องทางการติดต่อสื่อสารบน Internet ที่รวดเร็วที่สุดและมี API พร้อมใช้งานที่มีเสถียรภาพมาก ทางผู้จัดทำจึงได้เลือก Line Notify API มาใช้เป็น 1 ใน 3 ช่องทางในการส่งข้อมูลไปยังผู้ใช้งาน และทีสำคัญสามารถส่งข้อมูลไปยัง ผู้ใช้งานหลายๆ คนได้ในคราวเดียว ซึ่งจะมีโครงข่ายการเชื่อมต่อดังภาพที่ 1.1 ส่วนอีก 2 ช่องทาง คือ SMS และ E-mail เป็นช่องทางการติดต่อสื่อสารพื้นฐานที่ปัจจุบันก็อาจจะมีการใช้งานน้อยลงแต่ก็ยังจำเป็น เช่น SMS ในกรณีที่เครื่องปลายทางไม่สามารถใช้ Internet ได้ SMS ก็เป็นช่องทางที่ดีที่สุดในการส่งข้อมูลไปถึงผู้รับได้ แต่มีข้อจำกัดคือ สามารถส่งข้อความได้ไม่เกิน 140 ตัวอักษร ทำให้ไม่สามารถส่งข้อความ Alarm ยาวๆ ได้

ส่วนกรณีการใช้ E-mail ก็เป็นช่องทางสำหรับผู้ที่ใช้ Internet อยู่ตลอดเวลาเช่นกัน ข้อดีคือสามารถส่งข้อความยาวๆ ได้ แต่ข้อเสีย คือ การรับข้อมูลต่างๆ ผู้ใช้งานต้องทำการเข้ามาตรวจสอบ E-mail เองหรือตั้ง Fetch ข้อมูลอัตโนมัติไว้ ซึ่งส่วนมากจะตั้งไว้ ½ - 1 ชั่วโมง ทำให้บางทีอาจจะไม่เห็น Alarm ได้ทันถ่วงที ถ้าเกิด Alarm ที่สำคัญมาก E-mail อาจจะไม่ใช่ช่องทางที่ดีนัก



ภาพที่ 1.1 โครงข่ายโดยรวมของระบบ IOT และ Real-Time Analytics ที่ออกแบบไว้

1.2 วัตถุประสงค์ของการศึกษา

1. เพื่อนำอุปกรณ์ IOT ติดตั้งไว้ในจุดที่ต้องการและส่งข้อมูลมายังอุปกรณ์ Analyzer ได้
2. เพื่อวิเคราะห์ Alarm ที่ส่งจากอุปกรณ์ IOT ได้ว่าต้องส่งไปที่ผู้ที่รับผิดชอบคนไหน
3. เพื่อส่ง Alarm ออกไปยัง Line Group ที่กำหนดไว้ได้
4. เพื่อส่ง Alarm ออกไปยัง SMS ที่ผู้รับผิดชอบทำการลงทะเบียนเบอร์โทรศัพท์ไว้ได้
5. เพื่อส่ง Alarm ออกไปยัง E-Mail ที่ผู้รับผิดชอบทำการลงทะเบียนเบอร์โทรศัพท์ไว้ได้
6. เก็บรวบรวมประวัติการเกิด Alarm ย้อนหลังไว้ได้

1.3 สมมติฐานของการศึกษา

1. สามารถส่ง Alarm ไปยังผู้ที่รับผิดชอบได้อย่างถูกต้อง
2. สามารถส่ง Alarm ไปจนถึงผู้รับได้อย่างทันท่วงทีภายในเวลาอันรวดเร็ว

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1. สามารถนำอุปกรณ์ IOT ที่พัฒนาขึ้นไปติดตั้งยังจุดที่ต้องการและตรวจสอบเซ็นเซอร์ต่างๆ เมื่อมีความผิดปกติ จะได้รับ Alarm มายังอุปกรณ์ Analyzer
2. ระบบ Analyzer สามารถกรองและวิเคราะห์ข้อมูล Alarm ที่ได้รับเพื่อที่จะส่งไปยังผู้ที่รับผิดชอบได้อย่างถูกต้อง
3. สามารถทำงานร่วมกับ Line API , SMS API และ E-mail SMTP เพื่อส่งข้อมูลได้ทั้ง 3 ช่องทางได้อย่างถูกต้องและรวดเร็ว
4. เมื่อผู้ที่รับผิดชอบได้ข้อมูล Alarm แล้วสามารถเข้าทำงานเพื่อระงับเหตุได้อย่างรวดเร็ว
5. กรณีที่ต้องการแจ้งเตือนในหลายๆ ไซต์งาน อุปกรณ์สามารถเคลื่อนย้ายและเพิ่มจำนวนไซต์งานได้อย่างง่ายดาย เพียงแค่เพิ่มชุดอุปกณ์และเชื่อมต่อเข้ากับ Analyzer
6. สามารถนำอุปกรณ์ IOT ไปพัฒนาต่อยอดเพื่อที่จะใช้ในอุตสาหกรรมอื่นๆได้

บทที่ 2
แนวคิด ทฤษฎี และผลงานวิจัยที่เกี่ยวข้อง

กล่าวนำ

การจัดทำโครงการระบบ Alarm Monitoring with IOT ผู้จัดทำได้เล็งเห็นว่าอุปกรณ์ IOT ได้เข้ามามีบทบาทในชีวิตประจำวันมากขึ้น อีกทั้งในปัจจุบันอุปกรณ์ IOT หาซื้อได้ง่าย มี API และ Sensor ให้เลือกใช้ได้หลากหลายครอบคลุมทุกความต้องการ ทางผู้จัดทำจึงได้มีแรงบันดาลใจศึกษาเอกสารและงานวิจัยต่างๆ ที่เกี่ยวข้อง เพื่อนำข้อมูลทั้งหมดมาสนับสนุนความคิดที่จะนำอุปกรณ์ IOT มาใช้เป็นอุปกรณ์ Alarm Monitoring ในการทำโครงการนี้ให้เกิดขึ้นมา

โดยมีหลายทฤษฎีที่นำมาสนับสนุนดังนี้

2.1 IOT (Internet of Things) คือ อุปกรณ์ทางกายภาพที่เชื่อมต่ออยู่กับระบบอินเทอร์เน็ตตลอดเวลา และอุปกรณ์เหล่านี้จะมีเซ็นเซอร์และกลไกการถ่ายโอนข้อมูลอื่นๆ ซึ่งอุปกรณ์นี้มักจะถูกผนวกเข้ากับหน่วยควบคุมและ/หรือหน่วยประมวลผล IOT อื่นๆ ทางเครือข่ายตลอดเวลา โดยในอนาคตเราอาจจะได้เห็นอุปกรณ์ IOT เป็นรูปเป็นร่างขึ้นเรื่อยๆ มากกว่าปัจจุบันที่เป็นเพียงแค่บอร์ดทดลอง (ทางผู้จัดทำได้ใช้อุปกรณ์ Raspberry Pi ทำงานเปรียบเสมือนเป็น IOT เครื่องหนึ่ง) เช่น เครื่องซักผ้า , หลอดไฟ , กระทะ , แปรงสีฟัน หรือไม่เว้นแม้กระทั่งเสื้อผ้าในอนาคต เพราะปัจจุบันคนเราหันมาดูแลสุขภาพมากขึ้น IOT จึงอาจจะถูกนำมาเป็นอุปกรณ์สวมใส่เพื่อที่จะตรวจสอบและคอยเตือนในการใช้ชีวิตประจำวัน เพราะหลักการของ IOT มีเพียงแค่ เล็ก , ประหยัดไฟ และเชื่อมต่อเครือข่ายอยู่ตลอดเวลาทำให้สามารถตอบโจทย์ดังกล่าวได้อย่างดียิ่ง

ปัจจุบันก็พยายามมีการพัฒนาทั้งอุปกรณ์ IOT และเครือข่าย IOT ขึ้นมาเพื่อให้สามารถตอบโจทย์ต่างๆ มากยิ่งขึ้น ทางผู้จัดทำได้จำแนกให้เห็นภาพดังนี้

2.1.1 ด้านอุปกรณ์ IOT เช่น มีการพัฒนาบอร์ดที่ใช้ในการรับส่งข้อมูลเพื่อที่จะติดต่อกับ Sensor ต่างๆ มากมายเช่น Raspberry Pi , Arduino , Adafruit , Intel Stick และอื่นๆ อีกมากมาย ซึ่งแต่ละบอร์ดก็จะมีความสามารถพิเศษตามโจทย์ที่หลากหลายไป

2.1.2 ด้านเครือข่าย IOT เช่น ปัจจุบันเครือข่ายไร้สายในปัจจุบันได้พัฒนาไปมากทำให้อุปกรณ์ IOT สามารถใช้ประโยชน์ในด้านเครือข่ายได้เป็นอย่างดี เช่น โครงข่าย 3G , 4G และ 5G แต่ยังไม่เพียงเท่านั้นโครงข่ายที่กล่าวมายังมีข้อจำกัดด้านการใช้ทรัพยากรการทำงานที่สูงคือการใช้งานด้านพลังงาน ถ้าใช้โครงข่าย Cellular แล้วแบตเตอรี่ 1000 mAh สามารถใช้ได้เพียงไม่เกิน 2-3 วัน (ยังไม่รวมการบริโภคพลังงานของ Sensor) จึงมีการพัฒนาเครือข่ายของ IOT ขึ้นมา เช่น LoRaWAN และ NB-IOT ซึ่งสองโครงข่ายนี้ได้ทลายข้อจำกัดของการใช้พลังงานจากเครือข่าย Cellular ได้สิ้นเชิง โดย Battery 1000 mAh สามารถใช้งานได้ถึง 10 ปี (ยังไม่รวมการบริโภคพลังงานของ Sensor) ซึ่งในอนาคตถ้าเครือข่าย LoRaWAN หรือ NB-IOT มีการให้บริการ เราน่าจะได้เห็นอุปกรณ์ IOT เป็นที่แพร่หลายมากขึ้น

ที่มา: Lluís Casals ID , Bernat Mir ID , Rafael Vidal ID and Carles Gomez * ID, 2017; Modeling the Energy Performance of LoRaWAN

ภาพที่ 2.1 อุปกรณ์ที่ใช้ในการเชื่อมต่อ LoRaWAN

ที่มา: http://shop.allthingstalk.com/product/lora-rapid-development-kit/

ภาพที่ 2.2 อุปกรณ์ที่ใช้ในการเชื่อมต่อ NB-IOT

ที่มา: https://www.blognone.com/node/101079

2.2 Raspberry Pi เปรียบเสมือนเครื่องคอมพิวเตอร์สมัยประมาณ 10 ปีก่อนความเร็วประมาณ 700 – 800 Mhz แต่มีขนาดเล็กเพียงความกว้าง ยาว เท่ากับขนาดบัตรเครดิตเท่านั้น และ ประหยัดพลังงานมาก ราคาไม่แพง และที่สำคัญมี Interface GPIO ที่สามารถนำมาต่อกับ Sensor ต่างๆ ที่มีขายทั่วไปอยู่ในท้องตลาดได้อย่างลงตัว และตัวบอร์ดนั้นก็ต่อเข้ากับจอคอมพิวเตอร์ หรือ จอโทรทัศน์ที่มี Port HDMI ที่ใช้งานกันอย่างแพร่หลายในปัจจุบันได้อย่างง่ายดายโดยที่ไม่จำเป็นต้องตั้งค่าอะไรพิเศษเพิ่มเติม

และนอกจากต่อจอแสดงผลแล้ว อุปกรณ์ Raspberry Pi II ที่ผู้จัดทำเลือกมาใช้ก็สามารถรองรับเมาส์และคีย์บอร์ดผ่าน USB port ตามปกติ ทำให้เราสามารถนำอุปกรณ์พื้นฐานต่างๆ ที่เป็น USB Port มาใช้กับอุปกรณ์ Raspberry Pi II ได้ทันที ส่วนระบบจ่ายไฟของ Raspberry Pi II นั้นก็ใช้เพียงสาย Mini USB กับ Adapter ขนาด 2 A ขึ้นไป (1 A ก็สามารถใช้ได้แต่ถ้ามีอุปกรณ์ USB Port ต่ออยู่เยอะและ CPU ทำงานหนักจะทำให้ไฟเลี้ยงไม่เพียงพอ)

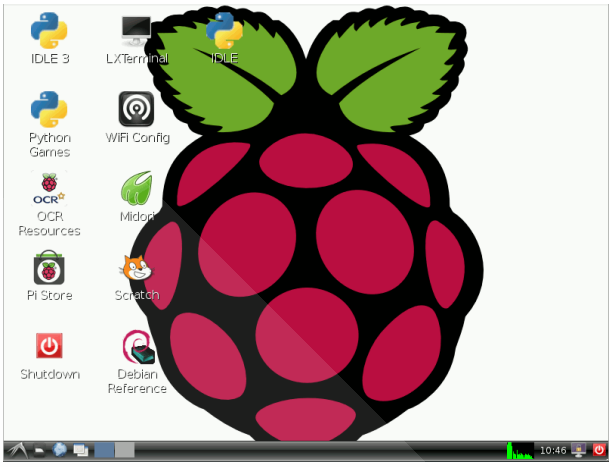
ภาพที่ 2.3 รูปภาพ Raspberry Pi

ที่มา: http://market.samm.com/en/raspberrypi-3

ประวัติโดยสังเขป ของ Raspberry Pi บอร์ด alpha-test ได้ถือเกิดขึ้นเมื่อปี 2549 โดยมีการพัฒนาต่อยอดมากจาก Board Controller ของ Atmel ATmega644 โดยมีผู้ก่อตั้งคือ อีเบน อัพตั้น ซึ่งเป็นนักวิชาการผู้ที่ชื่นชอบการประดิษฐ์ โดยมีแรงบันดาลใจจากที่อยากจะประดิษฐ์ เครื่องคอมพิวเตอร์ขนาดเล็กให้กับเด็กๆ และต่อมาก็ได้มีจุดมุ่งหมายที่จะทำให้เป็นคอมพิวเตอร์ ราคาถูกเพื่อที่ทุกคนจะสามารถเข้าถึงระบบคอมพิวเตอร์ได้ และสามารถศึกษาการทำงานของคอมพิวเตอร์ เขียนโปรแกรมง่ายๆ ได้ทันที โดยได้พัฒนาบนพื้นฐานของการใช้ชิพเซ็ต ARM มาเรื่อยๆ ตั้งแต่ โมเดล a , b และ b+
ที่มา: http://www2.crma.ac.th/itd/Know/RBPI/index.asp

2.2.1 Raspberry Pi ทำอะไรได้บ้าง

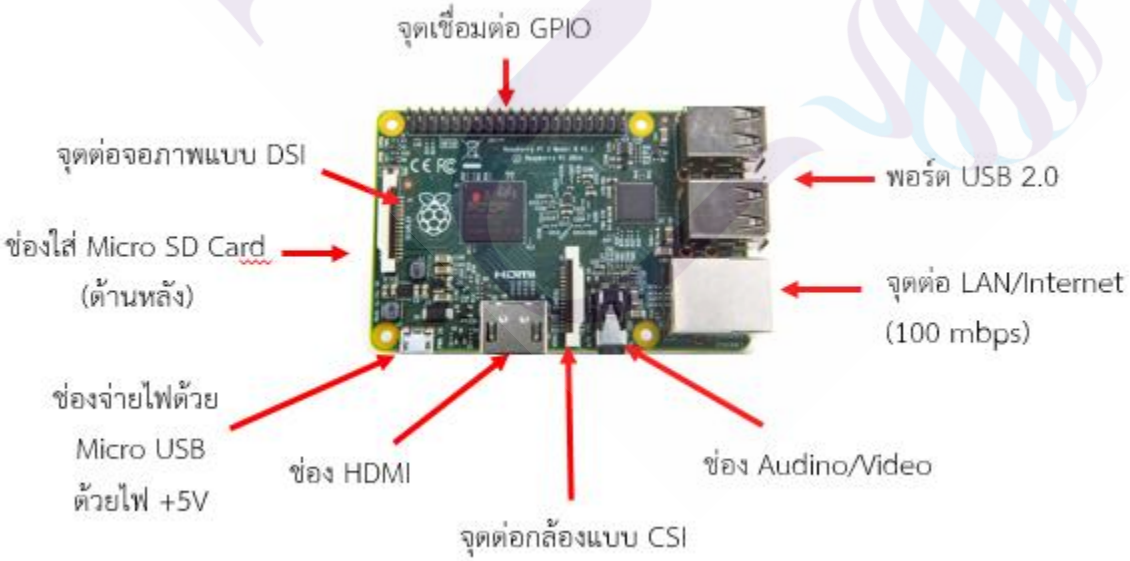
พูดได้เลยว่าในปัจจุบันคอมพิวเตอร์สามารถทำอะไรได้ Raspberry Pi ก็สามารถทำได้เช่นกัน เพียงแต่ความเร็วหรือประสิทธิภาพอาจจะไม่เร็วเท่าคอมพิวเตอร์ในปัจจุบัน แต่ก็แลกมาด้วยการประหยัดพลังงานอย่างมากของ Raspberry Pi , ขนาดของตัวบอร์ดเองที่เล็กมาก แต่ก็มี Port เชื่อมต่ออย่างครบครัน และราคาที่ไม่แพงทำให้ ทุกคนสามารถใช้ Raspberry Pi เป็นเครื่องคอมพิวเตอร์เดสก์ท็อปประจำบ้านอีกเครื่องหนึ่งได้ ส่วนระบบปฏิบัติการพื้นฐานของ Raspberry Pi นั้นคือ Raspian ซึ่งก็จะมีโปรแกรมที่ติดมาทำให้ผู้ใช้งานสามารถเรียนรู้การทำงานของระบบคอมพิวเตอร์ได้เลย รวมทั้งการเขียนโปรแกรมคอมพิวเตอร์ง่ายด้วยอย่างเช่น ภาษา Python , ภาษา C



ภาพที่ 2.4 ตัวอย่างหน้าจอ Raspberry Pi เมื่อนำไปต่อจอ Monitor

ที่มา: http://www2.crma.ac.th/itd/Know/RBPI/index.asp

2.2.2 สเปกและส่วนประกอบของ Raspberry Pi 2 Model B

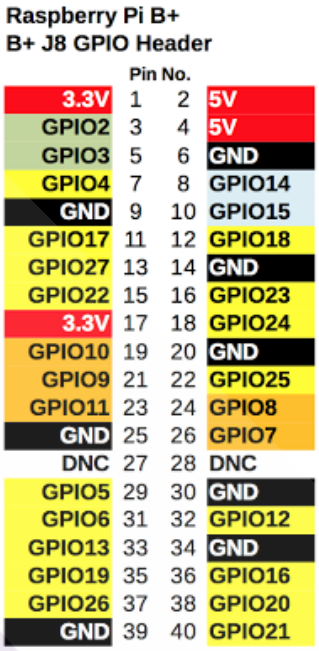


ภาพที่ 2.5 ตัวอย่าง Raspberry Pi และจุดเชื่อมต่อต่างๆ

ที่มา: http://www.homeofmaker.com/?p=891

1. CPU ความเร็ว 900 MHz quad-core ARM Cortex-A7
2. Memory 1 GB
3. USB 2.0 จำนวน 4 พอร์ต
4. GPIO (General Purpose Input/Output) 40 พิน
5. พอร์ต HDMI
6. พอร์ต Ethernet 10/100 Mbps
7. ช่องสัญญาณเสียง และ ภาพ ขนาด 3.5 มม.
8. ช่องต่อกล้องแบบ CSI
9. ช่องต่อสัญญาณภาพแบบ DSI
10. ช่องใส่ Micro SD card
11. หน่วยประมวลผลกราฟฟิค VideoCore IV 3D graphics core

ส่วนที่สำคัญการที่จะนำ Raspberry Pi ไปต่อกับ Sensor ต่างๆ คือ Port GPIO ซึ่งมีไว้สำหรับกำหนดการอ่านค่าแบบ Digital Interface โดยเราสามารถกำหนดโหมดให้เป็น Input หรือ Output ก็ได้ จากภาพที่ 2.6 จะอธิบาย GPIO ของ Raspberry Pi B+ (คือตัวที่ใช้อยู่) โดยมีขาเชื่อมต่อ คือ Power+, GND, I2C, UART, SPI และ GPIO ตามภาพที่ 2.6



ภาพที่ 2.6 ตำแหน่งต่างๆ บน PORT GPIO ของ Raspberry Pi

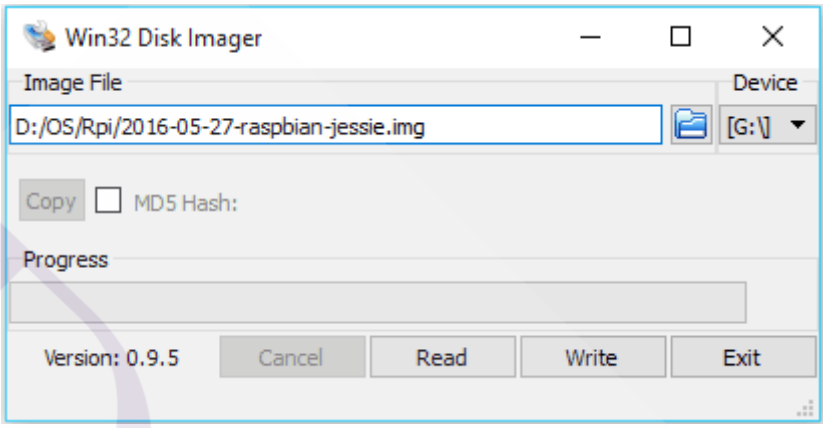
ที่มา: http://thaiopensource.org/มาเล่น-gpio-บน-raspbery-pi-กัน/

2.2.3 วิธีการติดตั้ง Raspberry Pi

เริ่มต้นให้นำ Micro SD (Class 10 ขึ้นไปยิ่งมีความเร็วในการอ่าน-เขียนมากยิ่งดี) มาทำการติดตั้งระบบปฏิบัติการ Raspbian เพราะเป็นระบบปฏิบัตการที่เป็นมาตรฐานการใช้งานของ Raspberry Pi ซึ่งสามารถไปดาวน์โหลดได้ที่ https://www.raspberrypi.org/downloads/ เมื่อดาวน์โหลดเสร็จเรียบร้อยแล้วให้ทำการแตกไฟล์ Raspbian ออกมาจะได้ไฟล์อิมเมจเพื่อที่จะนำไปติดตั้งระบบปฏิบัติการลงบน Micro SD

ทำการ Download โปรแกรมสำหรับเขียน SD card ทั่วไปเช่น Win32DiskImager , Etcher , SD Formatter ซึ่งผู้จัดทำจะยกตัวอย่างโปรแกรม Win32DiskImager ให้ไป Download ที่ http://sourceforge.net/projects/win32diskimager/ ต่อจากนั้นให้สร้าง directory สำหรับเก็บโปรแกรมนี้ เช่น C:\Users\Desktop\Win32DiskImager จากนั้นทำการ Unzip ไฟล์ที่ Download ไปไว้ใน Directory ที่สร้างขึ้น รันโปรแกรม Win32DiskImager.exe ที่ Unzip มา จะได้โปรแกรมตามรูปที่ 2.7 แล้วเลือก

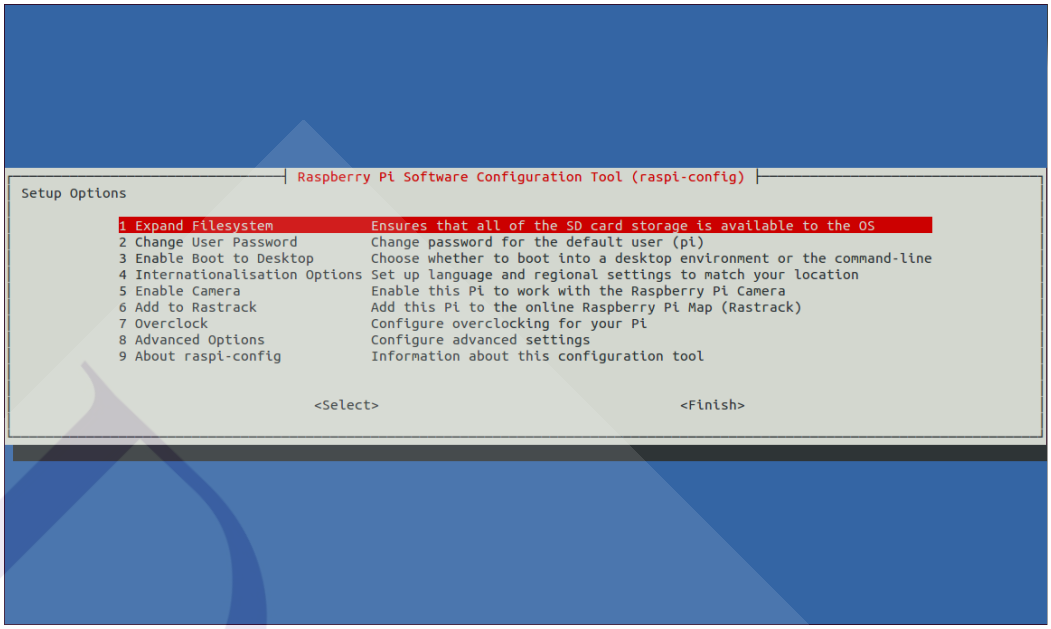
Image File เป็น .img ที่ Unzip ไว้ จากนั้น เลือก Device ไปที่ SD card ที่จะเขียน กด Write แล้วรอจนโปรแกรม Process เสร็จเรียบร้อย จึงจะสามารถดัง Micro SD ออกจาก Card Reader ได้



ภาพที่ 2.7 รูปโปรแกรม Win32DiskImager สำหรับลง Raspbian

ที่มา: https://playelek.com/rpi-os-image/

จากนั้นจึงนำ Micro SD ไปใส่ไว้ใน Slot ของ Micro SD Card บนบอร์ด Raspberry Pi แล้วทำการต่อจอภาพ, ต่อ Keyboard, ต่อสาย Network หรือถ้ามี USB Wifi ก็สามารถนำมาใช้งานได้ จากนั้นก็ต่อสายไฟเข้าที่ช่อง Micro USB เป็นอันพร้อมใช้งานตัว Raspberry Pi จะทำการ Boot ขึ้นมาพร้อมทั้งแสดงผลหน้าจอจนเจอหน้าจอการใช้งานตามภาพที่ 2.8 เพื่อที่จะเริ่มการติดตั้งระบบปฏิบัติการ Raspbian



ภาพที่ 2.8 Raspberry เมื่อมีการ Boot ครั้งแรกเพื่อติดตั้ง

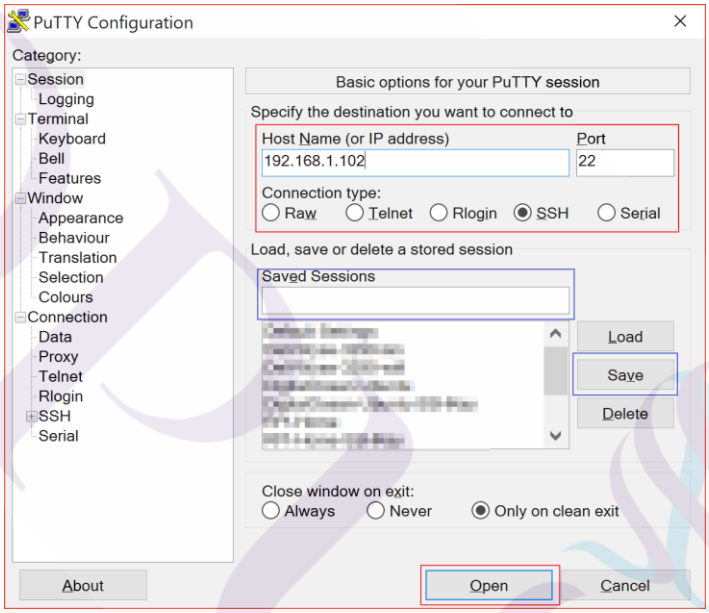
สิ่งสำคัญต่อไปเราจะต้องดำเนินการ Change User Password เพื่อตั้งค่า Default Password ของอุปกรณ์ Raspberry Pi และ Change Time zone ที่ Menu Internationalization Options

ในส่วนของ Application นั้นตัวระบบปฏิบัติการ Raspbian นั้นจะมีการติดตั้ง Application ต่างๆ ที่จำเป็นเบื้องต้นมาอยู่แล้วเช่น โปรแกรมเขียนภาษา Python , โปรแกรม TextEditer ทำให้เราไม่ต้องติดตั้งเพิ่มเติม

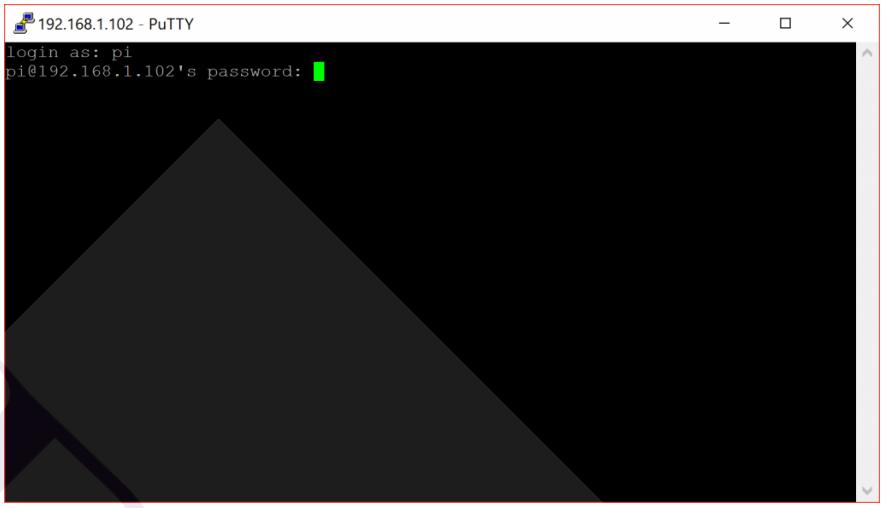
สิ่งที่เราต้องติดตั้งเพิ่มเติมคือ SSH Server เพื่อให้เราสามารถ Remote เข้ามายัง Raspberry Pi ได้ผ่าน Command Line โดยดำเนินการติดตั้งดังนี้ คือ เข้าที่ Terminal ของ Raspberry Pi แล้วพิมพ์ Command “sudo apt-get install openssh-server” ระบบจะดำเนินการติดตั้ง SSH Server ให้ถ้า Raspberry Pi ของเรามีการเชื่อมต่อ Internet แล้ว

ต่อไปคือการที่เราจะ Remote เพื่อเข้าไปจัดการตัวอุปกรณ์ Raspberry Pi นั้นเราต้องมีโปรแกรมจำพวก Secure Shell เพื่อที่จะ Remote เข้าไปได้ เช่น โปรแกรม Putty , SecureCRT , Xterm หรือ PowerShell ตัวอย่างทางผู้จัดทำได้เลือกใช้โปรแกรม Putty เพราะมีข้อดีคือ เป็น Freeware ที่สามารถ Download มาใช้งานได้ทันทีโดยไม่ต้องติดตั้ง (Run EXE ได้เลย) โดยสามารถ Download ใช้งานได้ดังนี้

1. ดาวโหลดโปรแกรม PuTTY > http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html
2. เปิดโปรแกรม PuTTY และพิมพ์ค่าดังต่อไปนี้
3. Host Name : ใส่ IP Address ของ Raspberry Pi (ตัวอย่างเช่น 192.168.1.102)
4. Port : ถ้าไม่ได้เปลี่ยนค่าส่วนมากจะใช้ Port 22
5. Connection type : เลือก SSH
6. กด Open เพื่อทำการเชื่อมต่อ



ภาพที่ 2.9 โปรแกรม Putty ที่ใช้ในการเชื่อมต่อ Raspberry Pi



ภาพที่ 2.10 ใส่ Username และ Password ของ Raspberry Pi ที่ตั้งไว้

2.3 DHT11 Humitdity and Temperature Sensor – Sensor ตรวจสอบอุณหภูมิและความชื้น

ภาพที่ 2.11 DHT11 Sensor

ที่มา: http://www.circuitbasics.com/how-to-set-up-the-dht11-humidity-sensor-on-the-raspberry-pi/

2.3.1 Specification ของ DHT11

ย่านวัดความชื้น 20-90% RH โดยมีค่าความแม่นยำ +- 5% RH ความละเอียดในการวัด 1 % แสดงผลแบบ 8 บิต

ย่านวัดอุณหภูมิ 0 - 50 องศาเซลเซียส โดยมีค่าความแม่นยำ +- 2 องศาเซลเซียส ความละเอียดในการวัด 1 องศาเซลเซียส แสดงผลแบบ 8 บิต

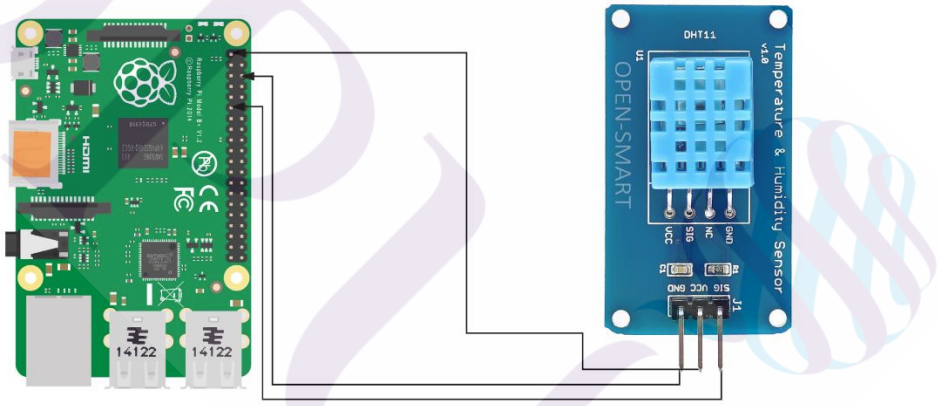
กินกระแส 0.5 - 2.5 mA (ขณะทำการวัดค่า) ที่ระดับแรงดัน 3 - 5.5 VDC อ่านค่าสัญญาณ (Sample Rate) ทุก 1 วินาที

2.3.2 ตัวอย่างการต่อวงจร

DHT11 Pin 1 (VCC) –> Raspberry Pi +5V

DHT11 Pin 2 (DATA) –> GPIO PIN 11 (GPIO17)

DHT11 Pin 3 (GND) –> Raspberry Pi GROUND



ภาพที่ 2.12 DHT11 และตัวอย่างการเชื่อมต่อเข้ากับ Raspberry

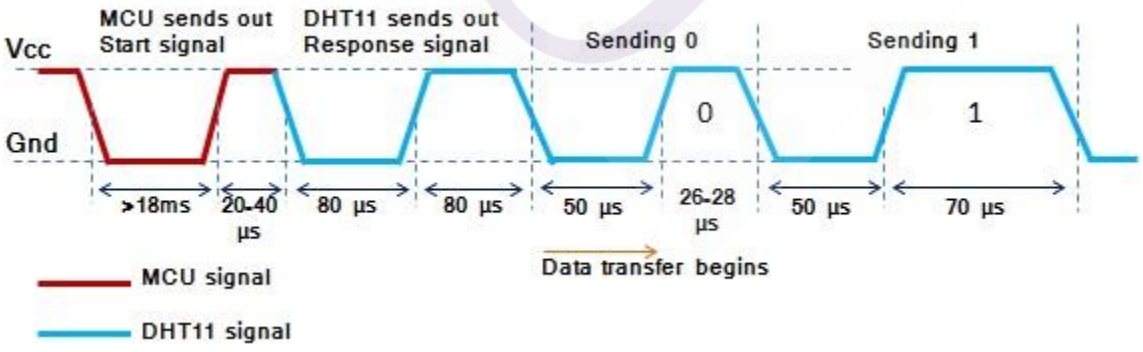
ที่มา: http://issamben.com/how-to-setup-dht11-humidity-temparture-sensor-on-raspberry-pi/

2.3.3 ขั้นตอนการนำมาใช้งาน

1. ทำการติดตั้ง library Adafruit DHT11 library จาก Github ตาม Link ดังต่อไปนี้ https://github.com/adafruit/Adafruit_Python_DHT.git
2. Sample Code การอ่านข้อมูลจาก DHT11 Sensor

```
#!/usr/bin/python
import sys
import Adafruit_DHT
while True:
    humidity, temperature = Adafruit_DHT.read_retry(11, 4)
    print 'Temp: {0:0.1f} C  Humidity: {1:0.1f} %'.format(temperature, humidity)
```

วิธีการติดต่อสื่อสารระหว่าง DHT11 กับ Raspberry Pi ที่ Port Interface GPIO นั้นจะใช้วิธี Single-wire Two-way Serial interface คือ การสื่อสารอนุกรมสองทางโดยใช้สายเส้นเดียว การสื่อสารแบบนี้จะใช้สายสื่อสารเพียงเส้นเดียวและส่งข้อมูลได้ทั้งจาก Raspberry Pi ไปที่ตัว DHT11 และในทางกลับกันจาก DHT11 กลับไป Raspberry Pi ด้วยเช่นกัน โดยในการติดต่อสื่อสารที่ใช้สายเพียงเส้นเดียวนั้น จะใช้การตกลงกันว่าขณะที่ตัวใดตัวหนึ่งส่งข้อมูลนั้นตัวนั้นจะอยู่ในสถานะ Master ส่วนอีกตัวจะอยู่ในสถานะ Slave ทีนี้พอ Master รู้ว่า Slave พร้อม Slave จะส่งแรงดันระดับต่ำกลับไปบ้าง การส่งแรงดันจาก Slave กลับไปจะนาน 80 µs จากนั้นจะรออีก 80 µs ก่อนจึงจะส่งข้อมูลของบิตแรกออกมาที่ Slave โดยจะมีสถานะการทำงานดังรูปที่ 2.13



ภาพที่ 2.13 DHT11 แสดงสถานะในการส่งข้อมูล Two-way

ที่มา:

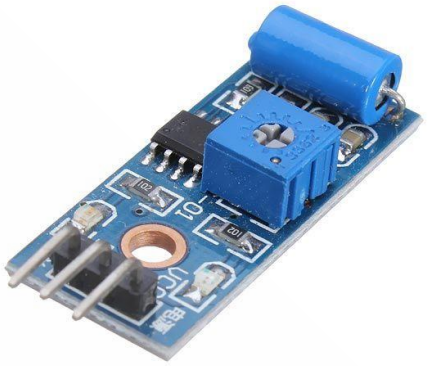
https://www.arduitronics.com/article/13/%E0%B8%81%E0%B8%B2%E0%B8%A3%E0%B9%83%E0%B8%8A%E0%B9%89%E0%B8%87%E0%B8%B2%E0%B8%99-dht11-humitdity-and-temperature-sensor-%E0%B8%81%E0%B8%B1%E0%B8%9A%E0%B8%9A%E0%B8%AD%E0%B8%A3%E0%B9%8C%E0%B8%94-arduino

สำหรับการส่งบิตเป็น "0" ตัว Slave จะดึงระดับแรงดันลงต่ำนาน 50 ไมโครวินาที และปล่อยเป็นระดับ "สูง" นาน 26-28 ไมโครวินาที ดังภาพ 2.13 (ดูช่วง Sending 0) แต่ถ้าเป็นการส่งข้อมูลเป็น "1" ตัวส่งจะดึงสายสัญญาณลงระดับต่ำ 50 ไมโครวินาที และปล่อยให้เป็นระดับสูงนาน 70 ไมโครวินาที (ดูช่วง Sending 1) ทีนี้รู้ว่าแต่ละบิต DH11 ส่งมาเป็นบิต "0" หรือ "1" มา และส่งมาจนครบข้อมูลหนึ่งชุด ในแต่ละชุดของข้อมูลที่ส่งมาจาก DH11 ตัว MCU รับข้อมูลแล้วจะต้องเอามาแปลงต่อว่าข้อมูลที่ส่งมามันแปลว่าอะไร แต่ละชุดข้อมูลจะยาว 40 บิต และใช้เวลาส่งประมาณ 40 มิลลิวินาที ใน 40 บิตที่ส่งมา ประกอบด้วย " 8bit integral RH data + 8bit decimal RH data + 8bit integral T data + 8bit decimal T data + 8bit check sum" ซึ่งมีคนเขียน library ไว้ใช้งานกับ Raspberry board ให้เรียบร้อยแล้ว แค่เอา library มาใส่ให้ถูกแล้วเรียกใช้

ที่มา:

https://www.arduitronics.com/article/13/%E0%B8%81%E0%B8%B2%E0%B8%A3%E0%B9%83%E0%B8%8A%E0%B9%89%E0%B8%87%E0%B8%B2%E0%B8%99-dht11-humitdity-and-temperature-sensor-%E0%B8%81%E0%B8%B1%E0%B8%9A%E0%B8%9A%E0%B8%AD%E0%B8%A3%E0%B9%8C%E0%B8%94-arduino

2.4 SW-420 Vibration Sensor – Sensor ตรวจสอบการสั่นสะเทือน



ภาพที่ 2.14 SW-420 Sensor

2.4.1 Specification ของ SW-420

1. ความดันไฟ 3.3 - 5 V
2. On-board indicator LED to show the results
3. On-board LM393 chip
4. Dimension of the board: 3.2cm x 1.4cm

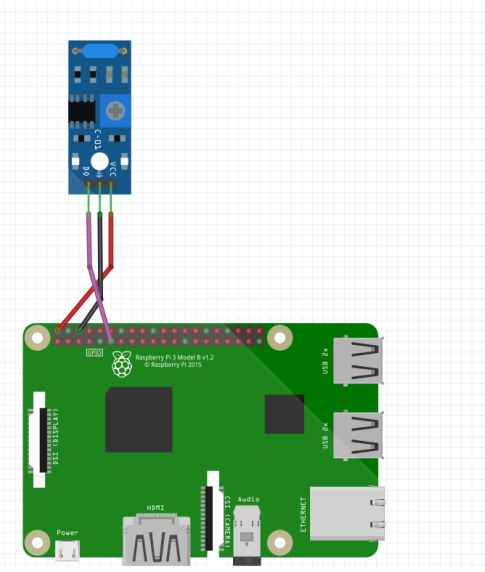
ที่มา: https://www.autobotic.com.my/LM393-Tilt-Sensor-Module

2.4.2 ตัวอย่างการต่อวงจร

SW-420 Pin 1 (VCC) –> Raspberry Pi +5V

SW-420 Pin 2 (DATA) –> GPIO PIN 13 (GPIO27)

SW-420 Pin 3 (GND) –> Raspberry Pi GROUND



ภาพที่ 2.15 SW-420 และตัวอย่างการเชื่อมต่อเข้ากับ Raspberry

ที่มา: https://www.piddlerintheroot.com/vibration-sensor/

2.4.3 ขั้นตอนการทำงาน

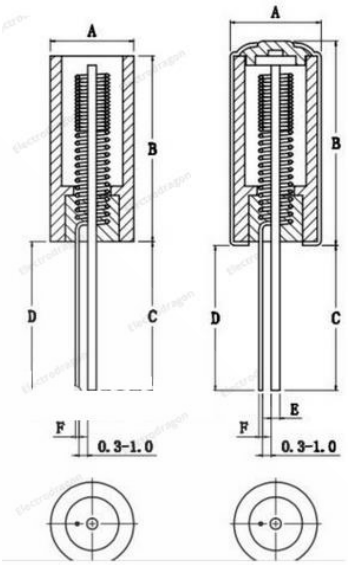
ตัว Sensor ของ SW-420 นั้นเมื่อนำมาต่อตามรูปที่ 2.15 เข้ากับบอร์ด Raspberry Pi แล้วเมื่อจ่ายไฟเข้าในระบบ Sensor สามารถทำงานได้ทันทีโดยสามารถเขียนภาษา Python เพื่อทำการรับค่า Port GPIO ตามที่ต้องการได้ตาม Sample Code ต่อไปนี้

```
#!/usr/bin/python

import RPi.GPIO as GPIO

import time

channel = 17

GPIO.setmode(GPIO.BCM)

GPIO.setup(channel, GPIO.IN)
```

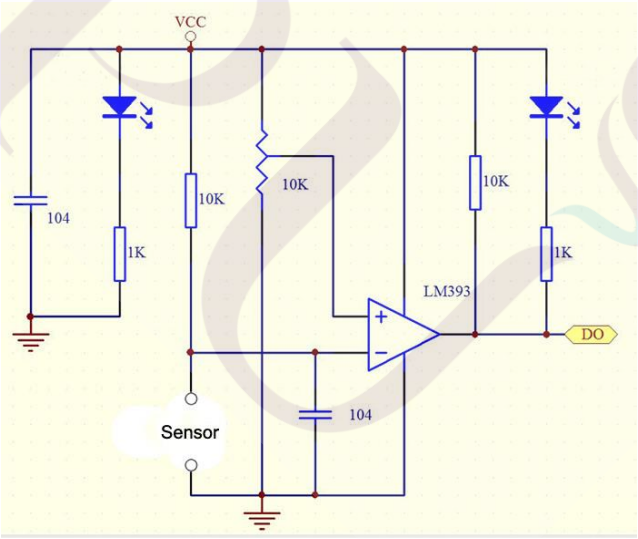
```
def callback(channel):
    if GPIO.input(channel):
        print "Movement Detected!"
    else:
        print "Movement Detected!"
GPIO.add_event_detect(channel, GPIO.BOTH, bouncetime=300)
GPIO.add_event_callback(channel, callback)
while True:
     time.sleep(1)
```

การทำงานของ SW-420 จะทำงานบน Event Call Back โดยจะ Detect สัญญาณจากการสั่น ถ้ามีการสั่นเกิดขึ้น Sensor จะส่งกระแสกลับมาที่ Raspberry Pi และเมื่อ Raspberry Pi ได้รับสัญญาณจาก GPIO ที่ SW-420 ต่ออยู่ตัว Raspberry Pi จะเรียก Event Call Back ขึ้นมาทำงาน

SW-420 เป็นสปริงชนิด Non-direction vibration induced switches และมี Dimension ตามภาพที่ 2.16 เมื่อไม่มีการสั่นเกิดขึ้นสวิทช์ทั้งหมดจะอยู่ใน สถานะ "OFF" เมื่อแรงภายนอกที่จะเข้าถึงการสั่นสะเทือนหรือการเคลื่อนย้ายด้วยความเร็วที่เพียงพอ สื่อกระแสไฟฟ้าจะถูกนำมาใช้ทันทีที่สถานะ "ON" เมื่อแรงภายนอกหายไปสวิทช์จะเปลี่ยนกลับเป็น สถานะ "OFF" และมีวงจร Circuit ตามภาพที่ 2.17



ภาพที่ 2.16 SW-420 แสดงการทำงานภายใน SW-420



ภาพที่ 2.17 วงจร Circuit ภายใน SW-420

2.5 MQ-7 Carbon Sensor – Sensor ตรวจสอบคาร์บอนไดออกไซด์

ภาพที่ 2.18 MQ-7 Sensor

ที่มา: https://tutorials-raspberrypi.com/configure-and-read-out-the-raspberry-pi-gas-sensor-mq-x/

2.5.1 Specification ของ MQ-7

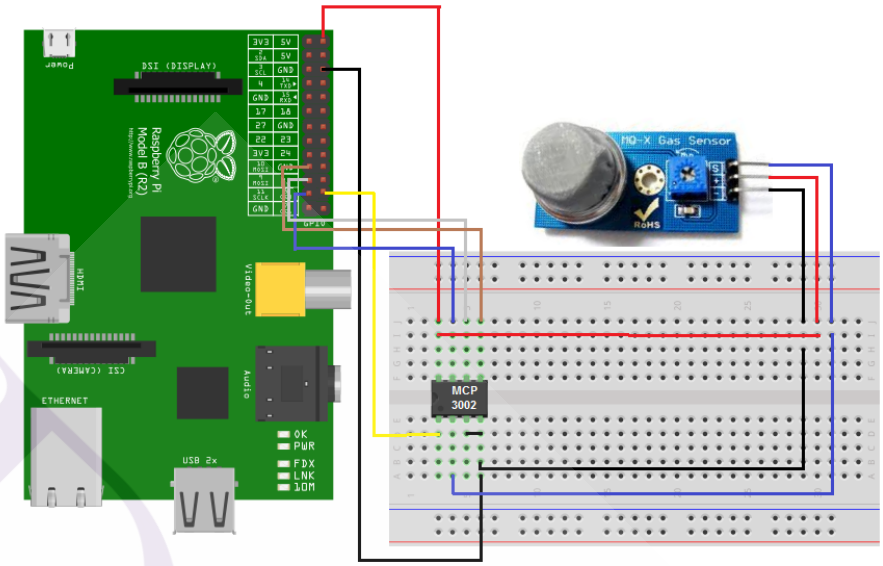
1. ความดันไฟ 3.3 - 5 V
2. Target Gas carbon monoxide
3. Detection range 10 ~ 500ppm CO
4. Sensitivity Rs(in air)/Rs(in 150ppm CO)≥5

2.5.2 ตัวอย่างการต่อวงจร

MQ-7 Pin 1 (VCC) –> Raspberry Pi +5V

MQ-7 Pin 2 (DATA) –> GPIO PIN 15 (GPIO22)

MQ-7 Pin 3 (GND) –> Raspberry Pi GROUND



ภาพที่ 2.19 MQ-7 และตัวอย่างการเชื่อมต่อเข้ากับ Raspberry

ที่มา: http://www.learningaboutelectronics.com/Articles/MQ-2-smoke-sensor-circuit-with-raspberry-pi.php

2.5.3 ขั้นตอนการทำงาน

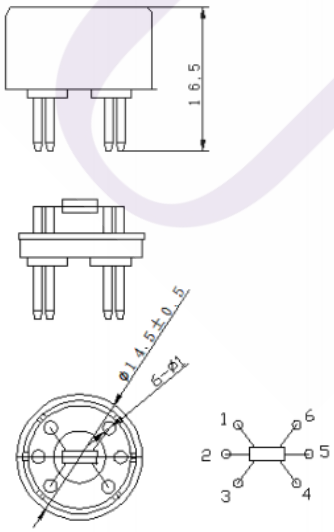
ตัว Sensor ของ MQ-7 นั้นเมื่อนำมาต่อตามรูปที่ 2.19 เข้ากับบอร์ด Raspberry Pi แล้วเมื่อจ่ายไฟเข้าในระบบ Sensor สามารถทำงานได้ทันทีโดยสามารถเขียนภาษา Python เพื่อทำการรับค่า Port GPIO ตามที่ต้องการได้ตาม Sample Code ต่อไปนี้

```
import time, sys
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
GPIO.setup(14, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
def action(pin):
    print('Sensor detected action!')
return
```

```
GPIO.add_event_detect(14, GPIO.RISING)
GPIO.add_event_callback(14, action)
 try:
       while True:
        print('alive')
        time.sleep(0.5)
except KeyboardInterrupt:
GPIO.cleanup()
sys.exit()
```

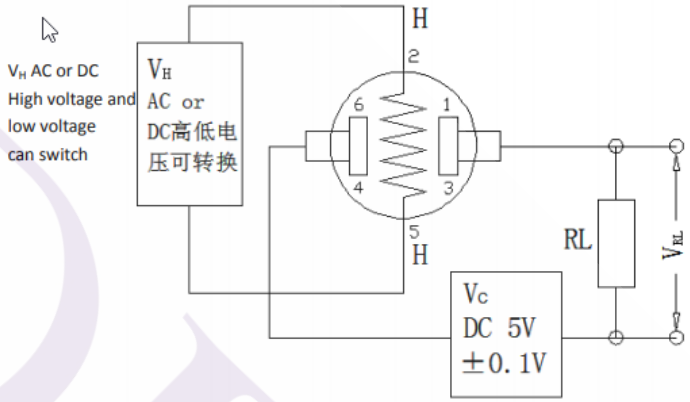
การทำงานของ MQ-7 จะทำงานบน Event Call Back โดยจะ Detect สัญญาณตลอดเวลาเพื่อรอส่งค่า Output เมื่อเกิดค่า Carbon เกิน Threshold ที่ตั้งไว้ตัว Sensor จะส่งกระแสกลับมาที่ Raspberry Pi และเมื่อ Raspberry Pi ได้รับสัญญาณจาก GPIO ที่ MQ-7 ต่ออยู่ตัว Raspberry Pi จะเรียก Event Call Back ขึ้นมาทำงาน

วัสดุที่มีความสำคัญของเซ็นเซอร์ก๊าซ MQ-7 คือ SnO_2 ซึ่งมีการนำไฟฟ้าต่ำในอากาศที่สะอาด ทำให้การตรวจจับโดยวิธีการตรวจจับ CO ที่อุณหภูมิต่ำ ค่าการนำไฟฟ้าของเซนเซอร์สูงขึ้นพร้อมกับ CO ความเข้มข้นของก๊าซเพิ่มขึ้นที่อุณหภูมิสูง ผู้ใช้สามารถแปลงการเปลี่ยนแปลงได้โดยการนำไฟฟ้าของ CO ไปใช้เพื่อให้สอดคล้องกับสัญญาณเอาต์พุตของแก๊สที่มีความเข้มข้นผ่านวงจรง่ายๆ

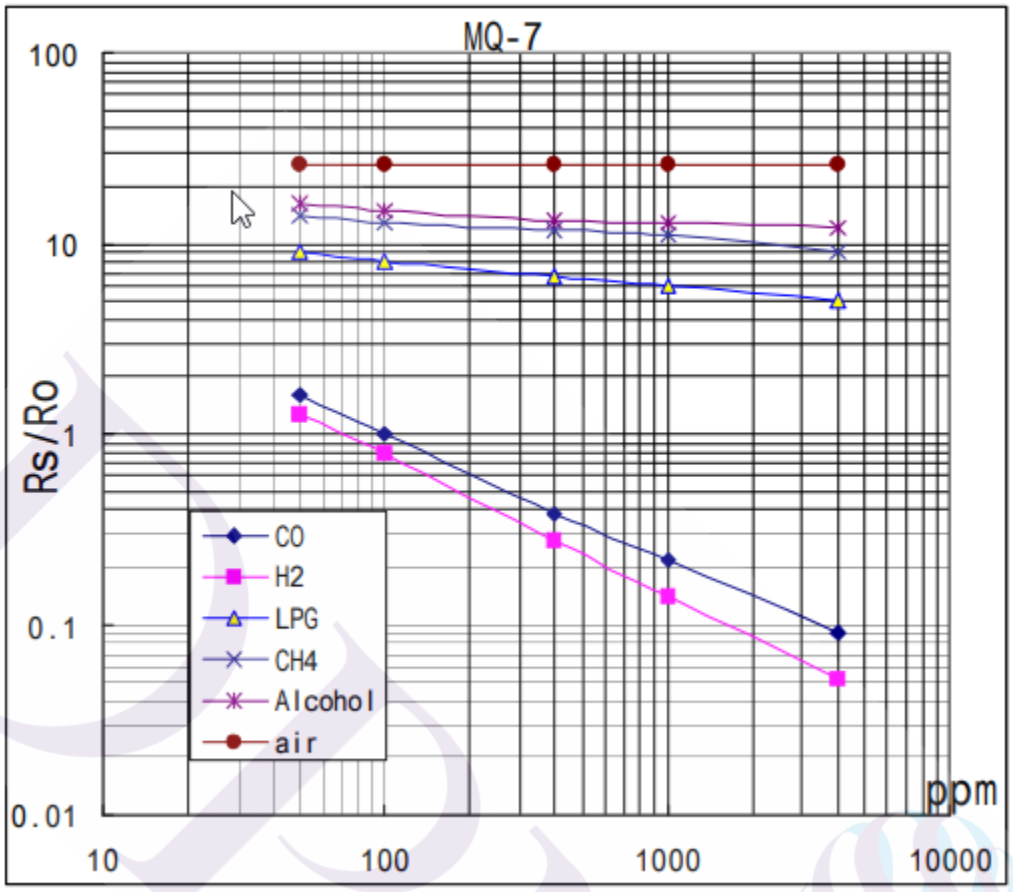


ภาพที่ 2.20 MQ-7 ขนาดและ Dimension

จากภาพที่ 2.21 แสดงวงจรมาตรฐานของ MQ-7 ประกอบด้วย 2 ส่วน คือ เป็นวงจรวัดความร้อนที่มีแรงดันไฟฟ้าสูงและแรงดันไฟฟ้าต่ำ (ในวงกลม) และ วงจรสัญญาณ Output เพื่อที่สามารถส่งข้อมูลที่ถูกต้องออกมาได้ ซึ่งจะแสดงให้เห็นในภาพที่ 2.22 ของความไว MQ-7 ต่อ ก๊าซหลายชนิด ในอุณหภูมิ 20 °C ความชื้น 65% ความเข้มข้นของออกซิเจน 21 %



ภาพที่ 2.21 วงจรเบื้องต้นของ MQ-7



ภาพที่ 2.22 ค่าความสัมพันธ์ระหว่างที่ตรวจพบ CO

ที่มา: https://www.sparkfun.com/datasheets/Sensors/Biometric/MQ-7.pdf

2.6 STM-32 Flame Sensor – Sensor ตรวจสอบเปลวไฟ

ภาพที่ 2.23 STM-32 Sensor

ที่มา: https://www.aliexpress.com/item/Waveshare-Fire-Flame-Sensor-Module-IR-Infrared-Flame-Detection-Sensor-Module-Flame-for-STM32-Raspberry-pi/32644539123.html

2.6.1 Specification ของ STM-32

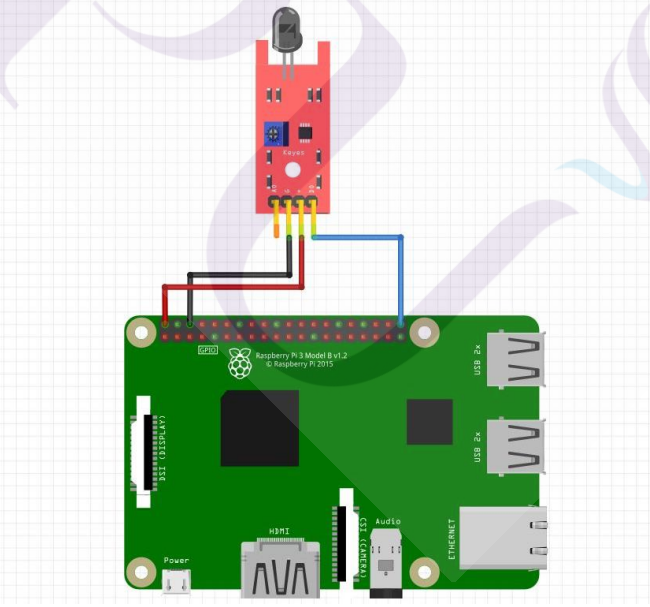
1. ความดันไฟ 3.3 - 5 V
2. ช่วงสเปกตรัม: 760nm ~ 1100nm
3. มุม: 0-60 องศา
4. อุณหภูมิในการทำงาน -25C ~ 85C
5. ขนาด: 27.3 มิลลิเมตร x 15.4 มิลลิเมตร

2.6.2 ตัวอย่างการต่อวงจร

STM-32 Pin 1 (VCC) –> Raspberry Pi +5V

STM-32 Pin 2 (DATA) –> GPIO PIN 13 (GPIO21)

STM-32 Pin 3 (GND) –> Raspberry Pi GROUND



ภาพที่ 2.24 STM-32 และตัวอย่างการเชื่อมต่อเข้ากับ Raspberry

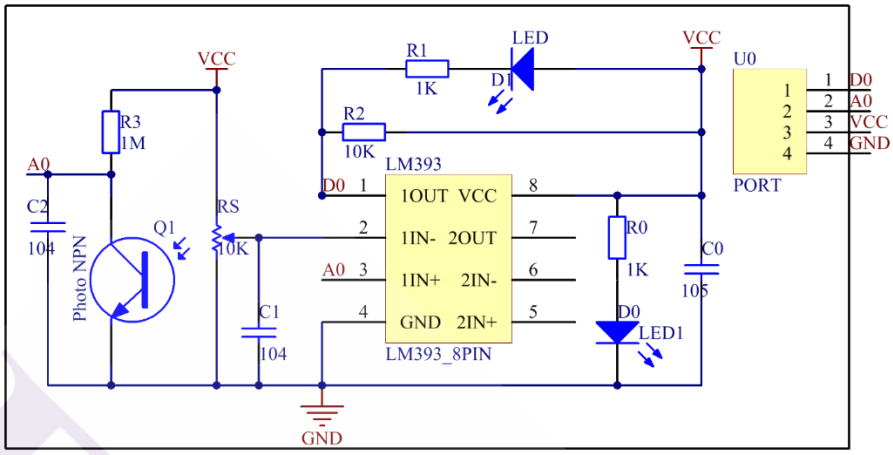
ที่มา: http://www.instructables.com/id/Flame-Sensor-Raspberry-Pi/

2.6.3 ขั้นตอนการทำงาน

ตัว Sensor ของ STM-32 นั้นเมื่อนำมาต่อตามรูปที่ 2.24 เข้ากับบอร์ด Raspberry Pi แล้วเมื่อจ่ายไฟเข้าในระบบ Sensor สามารถทำงานได้ทันทีโดยสามารถเขียนภาษา Python เพื่อทำการรับค่า Port GPIO ตามที่ต้องการได้ตาม Sample Code ต่อไปนี้

```
#!/usr/bin/python
import RPi.GPIO as GPIO
import time
channel = 21
GPIO.setmode(GPIO.BCM)
GPIO.setup(channel, GPIO.IN)
 def callback(channel):
        print("flame detected")
 GPIO.add_event_detect(channel, GPIO.BOTH, bouncetime=300)
GPIO.add_event_callback(channel, callback)
 while True:
          time.sleep(1)
```

การทำงานของ STM-32 จะทำงานคล้าย CO Sensor ซึ่งทำงานบน Event Call Back โดยจะ Detect สัญญาณตลอดเวลาเมื่อมีเปลวไฟเกิดขึ้น เกิน Threshold ที่ตั้งไว้ตัว Sensor จะส่งกระแสกลับมาที่ Raspberry Pi และเมื่อ Raspberry Pi ได้รับสัญญาณจาก GPIO ที่ STM-32 ต่ออยู่ตัว Raspberry Pi จะเรียก Event Call Back ขึ้นมาทำงาน

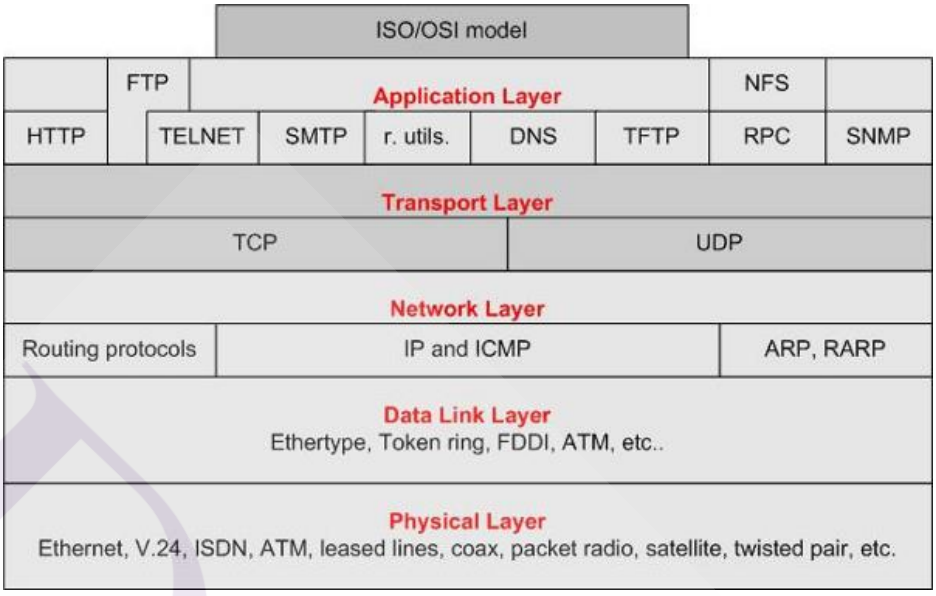


ภาพที่ 2.25 การเชื่อมต่อวงจร Circuit ของ STM-32

ที่มา: http://www.theorycircuit.com/arduino-flame-sensor-interface/

2.7 UDP Protocol – Protocol ที่ใช้ในการเชื่อมต่อและส่งข้อมูล

Protocol UDP นั้นเป็น Internet Protocol ที่ใช้การส่งข้อมูลที่ไม่มีการตอบกลับเพื่อยืนยันการถูกต้องของการรับข้อมูลพูดง่ายๆ คือผู้ส่งจะไม่รู้เลยว่าผู้รับได้รับข้อมูลหรือไม่ ถ้าจะตรวจสอบการถูกต้องของข้อมูลต้องทำการตรวจเช็คข้อมูลเองเช่น เมื่อได้รับแล้วให้ตอบกลับด้วยค่า ack ค่าหนึ่งถ้าไม่ตอบกลับแสดงว่าไม่ได้รับแต่จุดเด่นที่ทางผู้จัดทำเลือกมา Protocol UDP มาใช้คือ มีความรวดเร็วในการส่งข้อมูลสูง และมี Delay ต่ำ และที่สำคัญกับการนำมาใช้กับอุปกรณ์ IOT คือมีค่า Overhead ที่ต่ำทำให้อัตราการใช้ Data Internet ต่ำลงไปด้วย โดย Layer การทำงานอยู่ที่ Layer ที่ 4 (Transport Layer) ตามรูปที่ 2.26



ภาพที่ 2.26 Layer ของ UDP

ที่มา: http://www.voip-sip-sdk.com/p_312-voip-network-communication-protocols-voip.html

2.7.1 โครงสร้างของ packet UDP

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source port																Destination port															
4	32	Length																Checksum															

ภาพที่ 2.27 Packet ของ UDP

ที่มา: https://en.wikipedia.org/wiki/User_Datagram_Protocol

โครงสร้างของ UPD header มีเพียงแค่ 4 field เท่านั้น แต่ละอันมีขนาด 2 bytes (16 bits) ซึ่งประกอบไปด้วย

1. Source port number
2. Destination port number
3. Length
4. Checksum

2.8 งานวิจัยที่เกี่ยวข้อง

การพัฒนาระบบแจ้งเตือนระดับอุณหภูมิภายในตู้เย็นเก็บโลหิตผ่านเว็บ เบราว์เซอร์และแอปพลิเคชันบนมือถือ และพบว่าเป็นการนำระบบ IOT และ Sensor มาพัฒนาได้อย่างน่าสนใจโดย การวิจัยครั้งนี้ทีมผู้วิจัยได้ใช้กระบวนการวิจัยและพัฒนา (Research & Development) เพื่อออกแบบและสร้างระบบแจ้งเตือนระดับอุณหภูมิภายในตู้เย็นเก็บโลหิตผ่านเว็บเบราว์เซอร์และแอปพลิเคชันบนมือถือให้ เหมาะสมกับการใช้งานและมีประสิทธิภาพด้านการใช้งานสูงสุด โดยได้ดำเนินการศึกษาตามขั้นตอนดังต่อไปนี้

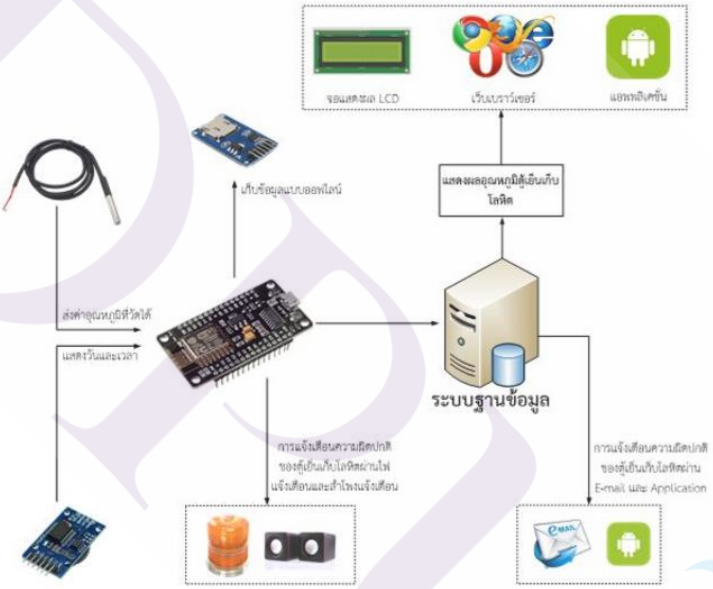
การเก็บรวบรวมข้อมูล การพัฒนาระบบแจ้งเตือนระดับอุณหภูมิภายในตู้เย็นเก็บโลหิตผ่านเว็บเบราว์เซอร์และแอปพลิเคชันบนมือถือ มีแหล่งข้อมูลที่ใช้ในการวิจัยแบ่งได้เป็น 2 ประเภทดังนี้

- ข้อมูลปฐมภูมิ เป็นข้อมูลที่ได้จากเจ้าหน้าที่ธนาคารโลหิตโรงพยาบาล อุตรดิตถ์
- ข้อมูลทุติยภูมิ เป็นการศึกษาค้นคว้ารวบรวมข้อมูลเอกสารวิชาการ รายงานการวิจัย หนังสือ บทความ ตาม อินเตอร์เน็ต เอกสารต่างๆ ที่เกี่ยวข้องกับการวิจัยและพัฒนาระบบแจ้ง เตือนระดับอุณหภูมิในลักษณะต่างๆ โดยมีเครื่องมือและอุปกรณ์การวิจัย ดังต่อไปนี้

1. เซ็นเซอร์วัดระดับอุณหภูมิ DS18B20 1 ตัวต่อ 1 ตู้
2. บอร์ด ESP8266 (NodeMCU)
3. สัญญาณเสียง (Alarm)
4. ระบบปฏิบัติการแอนดรอยด์
5. USB Wifi
6. Infrared Control
7. Micro SD Card Shield Arduino
8. บอร์ดอินเตอร์เฟส และจอแสดงผล (LCD)

9. บอร์ดควบคุม
10. แหล่งจ่ายไฟ
11. โครงสร้างกล่องควบคมุอลูมิเนียม
12. วัสดุเบ็ดเตล็ดอื่นๆ

การออกแบบและพัฒนาระบบแจ้งเตือน มีการทำงานดังนี้



ภาพที่ 2.28 การทำงานของระบบการแจ้งเตือน

และทางผู้วิจัยได้มีการเก็บผลการศึกษาถึงความคลาดเคลื่อนดังนี้

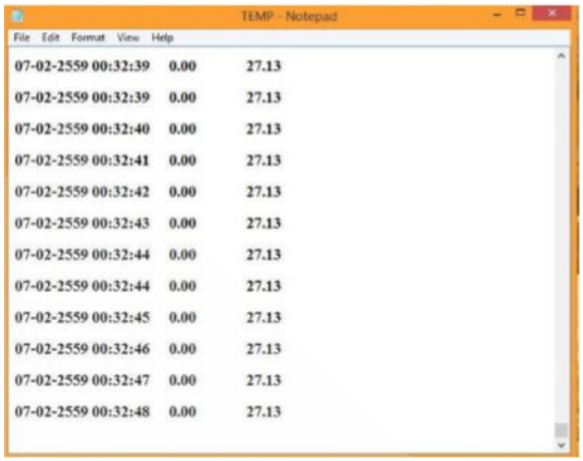
- วิเคราะห์ความแม่นยำในการวัด เนื่องจากข้อมูลที่ได้จากการวัดจะต้องมีการ เปรียบเทียบกับค่ามาตรฐาน ดังนั้นจึงต้องมาวิเคราะห์หาค่าความแม่นย าโดยเปรียบเทียบกับอุณหภูมิที่วัด ได้จากตู้เย็นมาตรฐาน โดยใช้เครื่องวัดอุณหภูมิจากเซ็นเซอร์ DS18B20 และเทอร์โมมิเตอร์ ผลการ วิเคราะห์ดังแสดงในรูปที่ 2.28

ครั้งที่	อุณหภูมิที่วัดได้จากตู้เย็นเก็บโลหิต (องศาเซสเซียส)	อุณหภูมิที่วัดได้จากเซ็นเซอร์ DS18B20 (องศาเซสเซียส)	อุณหภูมิที่ได้จากเทอร์โมมิเตอร์ (องศาเซสเซียส)	ค่าความคลาดเคลื่อนของอุณหภูมิระหว่างตู้เย็นเก็บโลหิตกับเซ็นเซอร์ DS18B20 (องศาเซสเซียส)	ค่าความคลาดเคลื่อนของอุณหภูมิระหว่างตู้เย็นเก็บโลหิตกับเทอร์โมมิเตอร์ (องศาเซสเซียส)
1	6	5.8	5.4	0.2	0.6
2	6	5.8	5.6	0.2	0.4
3	6	5.9	5.2	0.1	0.8
4	6	5.7	5.6	0.3	0.4
5	6	5.9	5.5	0.1	0.5
6	6	5.9	5.4	0.1	0.6
7	6	5.7	5.5	0.3	0.5
8	6	5.7	5.6	0.3	0.4
9	6	5.8	5.4	0.2	0.4
10	6	5.8	5.5	0.2	0.5
ค่าเฉลี่ย	6	5.8	5.47	0.2	0.53

ภาพที่ 2.29 ความคลาดเคลื่อนของระบบ Sensor

- ผลการแจ้งเตือนระดับอุณหภูมิภายในตู้เย็นเก็บโลหิต จากการทดสอบการทำงานของ ระบบสามารถแจ้งเตือนอุณหภูมิภายในตู้เย็นเก็บโลหิตได้ดังต่อไปนี้ คือ 1) หน้าจอแสดงผล (LCD) 2) ไฟแจ้งเตือน 3) สัญญาณเสียง 4) เว็บเบราว์เซอร์โดยใช้ระบบไร้สาย 5) แอปพลิเคชันบนมือถือ 6) การ บันทึกข้อมูลแบบออฟไลน์ใน SD Card เพื่อดูข้อมูลย้อนหลัง

ภาพที่ 2.30 การแจ้งเตือนผ่าน Application



ภาพที่ 2.31 การเก็บค่าย้อนหลังบน Micro SD Card

- ผลการหาความพึงพอใจของระบบแจ้งเตือนระดับอุณหภูมิภายในตู้เย็นเก็บโลหิตผ่านเว็บเบราว์เซอร์และแอปพลิเคชันบนมือถือ เจ้าหน้าที่ผู้ทดลองใช้ระบบและผู้ใช้ระบบทั่วไป จำนวน 10 คน การหา ความพึงพอใจของระบบแจ้งเตือนระดับอุณหภูมิภายในตู้เย็นเก็บโลหิตผ่านเว็บเบราว์เซอร์และแอปพลิเคชัน บนมือถือ ด้วยวิธีการทดลองใช้ระบบกับกลุ่มผู้ใช้ และการสำรวจความพึงพอใจในการใช้ระบบ โดยใช้หลักการ วัดคุณสมบัติที่ดีของระบบสารสนเทศ ซึ่งจะต้องมีลักษณะที่ครอบคลุมมิติ ทั้ง 4 ได้แก่ มิติด้านเวลา (Time) มิติด้านเนื้อหา (Content) มิติด้านรูปแบบ (Format) และมิติด้านกระบวนการ (Process)

ทั้งนี้ทางผู้จัดทำวิจัยยังได้มีข้อเสนอแนะดังนี้

ข้อเสนอแนะการศึกษา การออกแบบและพัฒนาระบบการแจ้งเตือนและควบคุมระดับอุณหภูมิภายในตู้เย็นเก็บโลหิต ผ่านเว็บเบราว์เซอร์และแอปพลิเคชันบนมือถือ สามารถนำไปพัฒนาต่อโดยการประยุกต์ใช้เซ็นเซอร์อื่นๆ ในการวัดอุณหภูมิเพื่อให้ค่าอุณหภูมิที่มีความแม่นยำมากขึ้น ในส่วนของหน้าเว็บเบราว์เซอร์สามารถเพิ่ม ฟังก์ชั่นอื่นๆ ที่ผู้ใช้งานระบบต้องการในการทำงานมากยิ่งขึ้น เช่น ระบบส่งข้อมูลผ่าน SMS (นิธิกร เตโจ; ธีรยุทธ แก้ววงค์หิว; สุภัตรา ปินจันทร์, นเรศวรวิจัย ครั้งที่ 12)

จากงานวิจัยดังกล่าวผู้จัดทำจึงได้ศึกษาเพิ่มเติมเกี่ยวกับงานวิจัยที่เกี่ยวข้องเพื่อหาถึงปัญหาหลายๆ อย่างที่ได้เจอ และได้นำปัญหาต่างๆ มาตั้งเป็นสมมุติฐานตั้งต้น ดังงานวิจัยต่อไปนี้

1. การศึกษาและวิเคราะห์ระบบป้องกัน อัคคีภัยในอาคารขนาดใหญ่พิเศษ กรณีศึกษา: อาคารคุ้มเกล้าโรงพยาบาล ภูมิพลอดุลยเดช ซึ่งได้ศึกษาและวิเคราะห์ เกี่ยวกับ ระบบการป้องกันอัคคีภัย ซึ่งในงานวิจัยได้พูดถึง ประเด็นต่างๆ เช่น มาตรฐานการป้องกันอัคคีภัย , การคำนวณเวลาในการอพยพ , คำนวณสารดับเพลิง ซึ่งผลการวิจัยทางผู้วิจัยพบว่า “ผลการสำรวจแสดงให้ เห็นถึงความไม่ปลอดภัยของอาคาร อันเนื่องมาจาก ข้อบกพร่องในระบบอำนวยความสะดวก เช่น ระบบป้องกันอัคคีภัย ระบบดับเพลิงด้วยน้ำที่ทำงานได้ไม่สมบูรณ์ หรือระบบแจ้งเตือนอัคคีภัยอัตโนมัติที่ไม่สามารถทำงานได้” จะเห็นได้ว่าเมื่อวิเคราะห์ภาพรวมแล้วถ้าระบบถูกวางไว้ให้สามารถแจ้งเตือนได้ทันถ่วงทีจะช่วยให้แผนในการอพยพ และ การป้องกันอัคคีภัยสามารถสำเร็จลุล่วงได้ดี (วิชัย สุขคลีวนัต; อภิชาต แจ้งบำรุง, ฉบับที่ 82 ปีที่ 25 ตุลาคม - ธันวาคม 2555)

2. ระบบรักษาความปลอดภัยในบ้าน ได้ศึกษาและสร้างระบบเตือนภัยภายในบ้านที่ควบคุมด้วยไมโครคอนโทรลเลอร์ โดยมีเซ็นเซอร์ต่างๆ ทำงาน เช่น ตรวจจับควัน , ตรวจจับการทำลายประตู เป็นต้น โดยจะแจ้งเตือนทาง SMS ผ่านระบบโทรศัพท์มือถือ ซึ่งผลการทดลองสามารถทำงานได้อย่างน่าพอใจ (ศิริชัย เติมโชคเกษม; จันทิมา บัวผัน, 2553)

3. ระบบติดตามโรคคนทำงานออฟฟิศโดยใช้กล้อง Kinect สิ่งที่ผู้วิจัยต้องการสร้าง คือ ระบบที่จะช่วยดูแลสุขภาพของคนที่นั่งทำงาน ผ่านการเฝ้าระวังพฤติกรรมการนั่ง โดยระบบที่เป็นมิตรใช้ง่าย เข้าใจง่าย เป็นเหมือนผู้ช่วยหรือคนดูแลส่วนตัวที่สามารถสร้างได้โดยใช้ส่วนประกอบอิเล็กทรอนิกส์ที่มีในประเทศไทยและราคาไม่แพง โดยระบบจะคอย Tracking โดยใช้กล้องตรวจดูการนั่งก้มหน้า หรือ นั่งบิดตัวโดยใช้อุปกรณ์ IOT ส่งข้อมูลไปยังตัวประมวลผลและเก็บข้อมูลอย่าง realtime (นายภูจนา ปาลิยะวรรณ; ผู้ช่วยศาสตราจารย์ ดร. ชาคริดา นุกูลกิจ; ผู้ช่วยศาสตราจารย์ ดร. พรชัย มงคลนาม, 2557)

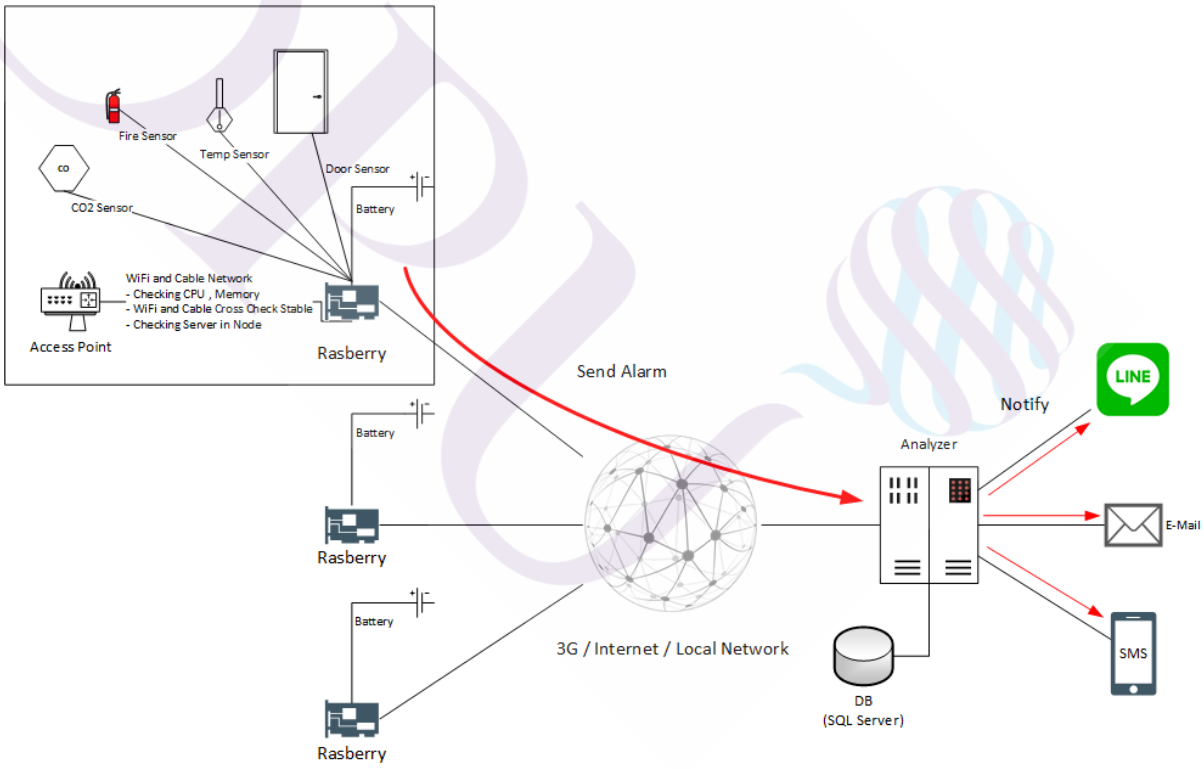
4. Web-based Application on Alarm Box Monitoring System โครงการมีเป้าหมายในการพัฒนาโปรแกรม Web-based Application บนเครื่องเซิร์ฟเวอร์เพื่อทำงานรองรับอุปกรณ์ Alarm Box ซึ่งเป็นอุปกรณ์ปลายทางตรวจวัดสภาพแวดล้อม ณ สถานที่ติดตั้งอุปกรณ์ปลายทาง เช่น ค่าแรงดันไฟฟ้าและกระแสไฟฟ้าของระบบจ่ายกระแสไฟฟ้า อุณหภูมิ เป็นต้น ซึ่งมีส่วนรับและประมวลผลข้อมูล สำหรับรับข้อมูลจากอุปกรณ์ปลายทางทุกๆตัวที่ส่งเข้ามาด้วยวิธี FTP (File Transfer Protocol) ทุกๆ 1 นาทีหรือตามคาบเวลาที่กำหนด ทำการประมวลผลเข้าเก็บในระบบฐานข้อมูล ได้ทดสอบสร้างโปรแกรมจำลองการส่งจากอุปกรณ์ปลายทาง 500-600 ตัว ผลปรากฏว่าเครื่องเซิร์ฟเวอร์ยังคงประมวลผลได้ครบทุกอุปกรณ์ปลายทาง (นายสุชาติ เลื่องยศลือชากุล, 2558)

5. ระบบแจ้งเตือนและควบคุมอุปกรณ์ไฟฟ้าด้วย IVR ผ่านระบบ VOIP โครงการมีวัตถุประสงค์คือศึกษาออกแบบ และพัฒนาเครื่องต้นแบบ ระบบแจ้งเตือนและควบคุมไฟฟ้าด้วยระบบ IVR ผ่านระบบ VOIP โดยประยุกต์ใช้ซอฟต์แวร์ Asterisk (IP-PBX) ซึ่งมีประโยชน์หลายด้าน เช่น ผู้ใช้สามารถทราบถึงเหตุผิดปกติที่เกิดขึ้นกับอุปกรณ์ไฟฟ้าได้อย่างทันท่วงทีและสามารถตรวจสอบสถานะได้ พร้อมทั้งสั่งเปิด - ปิด อุปกรณ์ไฟฟ้าได้อีกด้วย แต่ก็ยังมีข้อจำกัดหลายอย่างเช่น การเพิ่มชุดควบคุมเข้าไปต้องทำโดยผู้ที่ดูแลระบบเท่านั้น โดยผู้วิจัยได้มีข้อเสนอแนะว่าควรจะทำเป็นเว็บเพจ เพื่อสามารถเข้าใช้งานได้ในระบบอินเตอร์เน็ต จะได้ง่ายต่อการใช้งาน (ดุสิต วัฒนเสย, 2555)

ทางผู้จัดทำจึงได้นำการวิจัยทั้งหมดนี้มาเป็นแรงบันดาลใจในการพัฒนาระบบ Alarm Monitoring with IOT นี้ขึ้นมา โดยอาศัยประสบการณ์และแนวคิดจากผู้ที่ถ่ายทอดมาและต่อยอดเพื่อให้การทำงานดียิ่งขึ้น

บทที่ 3
วิธีดำเนินโครงการ

การดำเนินโครงการระบบ Alarm Monitoring with IOT (Internet of Thing) นี้ทางผู้จัดทำได้ดำเนินโครงการตามขั้นตอนต่างๆ ซึ่งมีรูปแบบการทำงานคือ จากอุปกรณ์ IOT ซึ่งรับข้อมูลจาก Sensor ต่างๆ และประมวลผลก่อนส่งต่อข้อมูลไปยัง อุปกรณ์ Analyzer ผ่าน Protocol UDP บนระบบ Network ตามที่อุปกรณ์ IOT ของ Node นั้นๆ จะสามารถเชื่อมต่อได้ โดยจะมีรูปแบบ Diagram ทำงานดังนี้



ภาพที่ 3.1 Diagram ของระบบ Alarm Monitoring with IOT (Internet of Thing)

ภาพที่ 3.1 จะแสดงให้เห็นถึงแผนผังการทำงานของงานวิจัยนี้โดยที่อุปกรณ์ IOT จะทำหน้าที่หลักอยู่ 2 หน้าที่คือ

1. คอยรับค่าจาก Sensor ต่างๆ เช่น CO_2 Sensor , Flame Sensor, Temp Sensor , Humidity Sensor , Vibration Sensor และ Monitoring Sensor

2. ประมวลผลค่าที่ได้รับและส่งค่าออกไปยังอุปกรณ์ Analyzer เพื่อที่อุปกรณ์ Analyzer จะได้ดำเนินการวิเคราะห์ Alarm และส่งต่อไปยังผู้ใช้

โดย Alarm ทั้งหมดจะถูกส่งโดย Protocol UDP ไปยังอุปกรณ์ Analyzer ผ่านระบบ Network ตามที่ได้เซ็ตไว้ 3G , 4G , LAN หรือ Wifi และ Analyzer จะวิเคราะห์ Alarm ว่า Alarm ดังต่อไปนี้จะดำเนินการส่งให้ผู้รับคนไหนต่อไปตามช่องทางต่างๆ ที่ได้เตรียมไว้คือ LINE Notify , E-mail และ SMS

3.1 วัสดุ อุปกรณ์ เครื่องมือหรือโปรแกรมที่ใช้ในการพัฒนา

1. เครื่องคอมพิวเตอร์ พร้อมการเชื่อมต่อเข้าสู่ระบบอินเตอร์เน็ต
2. โปรแกรม Visual Studio เพื่อใช้ในการพัฒนาโปรแกรม Analyzer
3. อุปกรณ์ Raspberry Pi II เพื่อใช้เป็นอุปกรณ์ IOT ในการ Detect Sensor
4. Sensor DHT11 ใช้เชื่อมต่อเข้ากับ Raspberry Pi เพื่อตรวจจับค่าความชื้นและอุณหภูมิ
5. Sensor SW-420 ใช้เชื่อมต่อเข้ากับ Raspberry Pi เพื่อตรวจจับการสั่นสะเทือน
6. Sensor MQ-7 ใช้เชื่อมต่อเข้ากับ Raspberry Pi เพื่อตรวจจับค่า CO
7. Sensor STM-32 ใช้เชื่อมต่อเข้ากับ Raspberry Pi เพื่อตรวจจับเปลวไฟ

3.2 ขั้นตอนการดำเนินงาน

1. ศึกษาและค้นคว้าข้อมูลจากปัญหาที่เจอ ซึ่งนำมาซึ่งวิธีการนำอุปกรณ์ IOT เข้ามาช่วยแก้ไขปัญหา และศึกษาช่องทางเพื่อนำมาประยุกต์ใช้ให้เหมาะสมกับการใช้ชีวิตประจำวัน

2. ศึกษารายละเอียดเกี่ยวกับ API ต่างๆ ที่ต้องนำมาใช้งาน เช่น LINE Notify API , SMS API , SMTP API และ Protocol ที่นำมาใช้ต้องมีความ Simple เพื่อง่ายต่อการพัฒนาต่อยอดในอนาคต

3. ศึกษาภาษาที่มีความยืดหยุ่นที่สามารถใช้เขียนลงอุปกรณ์ IOT เพื่อให้ตรงกับความต้องการการใช้งานของระบบ ซึ่งในโครงงานนี้ใช้ภาษา Python ในการพัฒนาร่วมกับระบบปฏิบัติการ Rasbian Detect ค่า Sensor จาก Port GPIO

4. ศึกษาภาษาที่สามารถเชื่อมต่อกับ API ต่างๆ และสามารถทำงานได้บนอุปกรณ์จำพวก Server และ Cloud ต่างๆ ได้ง่าย ซึ่งในโครงการนี้จะใช้ภาษา .NET ในการพัฒนา

5. จัดทำโครงร่างโครงงานระบบ Alarm Monitoring with IOT เสนอผ่านอาจารย์ที่ปรึกษา

6. นำเสนอรายงานความก้าวหน้าเป็นระยะ

7. ดำเนินการทดสอบนำระบบมา Integrate ทำงานร่วมกันทั้งระบบ Sensor , Analyzer และ API ต่างๆ สามารถทำงานได้อย่างถูกต้อง

8. ทดสอบ Simulation ผลงานโดยการเชื่อมต่อ IOT เข้ากับระบบ Network ที่สามารถเชื่อมต่อเข้ากับระบบ Analyzer ได้ พร้อมทั้ง Allow Port ที่ต้องการใช้งาน และ สามารถเชื่อมต่ออินเตอร์เน็ตได้

9. นำผลที่ได้เสนออาจารย์ที่ปรึกษา

3.3 การวิเคราะห์ข้อมูล

ในการวิเคราะห์ข้อมูลผู้ศึกษาได้แบ่งการวิเคราะห์ข้อมูลแบ่ง 2 ช่วงด้วยกัน

1. ข้อมูลความถูกต้องและการทำงานของอุปกรณ์ IOT

2. ข้อมูลความถูกต้องและการทำงานของระบบ Analyzer โดยต้องทำงานได้ถูกต้องทุก Function และ ทุก Alarm Filter

3. เสถียรภาพของระบบที่สามารถทำงานได้ต่อเนื่องกับทุกๆ API

3.4 เงื่อนไขที่ใช้ในการวิเคราะห์

1. ระบบ IOT สามารถ Detect ข้อมูลตาม Sensor ต่างๆ ได้อย่างถูกต้อง
2. ระบบ IOT สามารถส่ง Trigger ไปยัง Analyzer ได้อย่างถูกต้อง
3. ระบบ Analyzer สามารถส่ง Alarm ไปยัง Line API ได้อย่างถูกต้อง
4. ระบบ Analyzer สามารถส่ง Alarm ไปยัง SMS API โดยจำแนกตาม Filter ต่างๆ ได้
5. ระบบ Analyzer สามารถส่ง Alarm ไปยัง E-Mail โดยจำแนกตาม Filter ต่างๆ ได้

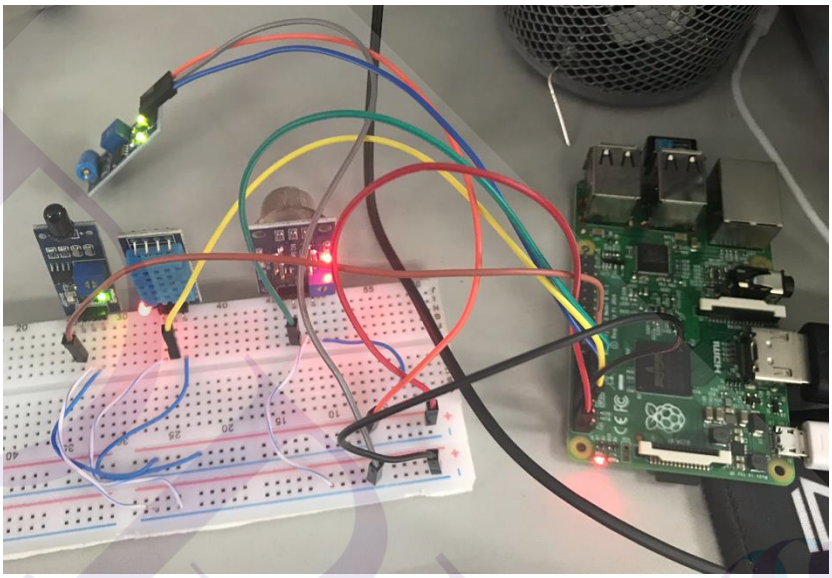
3.5 ระยะเวลาที่ใช้ในการดำเนินการ

1. ศึกษาหาความเป็นไปได้ของโครงการ 2 เดือน
2. พัฒนาระบบ IOT 1 เดือน
3. พัฒนาระบบ Analyzer 2 เดือน
4. ดำเนินการวัดผลและสรุปผลพร้อมจัดทำรายงาน 1 เดือน

3.6 วิธีดำเนินงาน

1. ทำการจัดเตรียม Raspberry Pi และติดตั้งระบบปฏิบัติการต่างๆ พร้อมทั้ง Compilers ที่จำเป็นให้ครบถ้วน

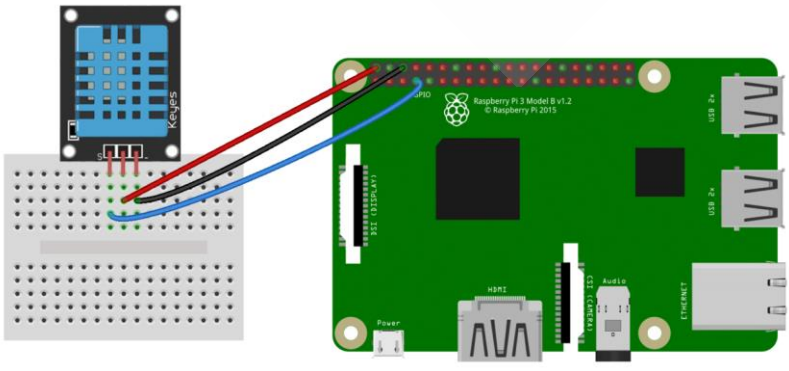
2. เมื่อเราติดตั้งและตั้งค่าระบบปฏิบัติการเรียบร้อย ดำเนินการเชื่อมต่อ Sensor ต่างๆ (DHT11,SW-420,MQ-7,STM-32) เข้ากับ Raspberry Pi ที่ Port GPIO ตาม Diagram ที่ Plan ไว้



ภาพที่ 3.2 เชื่อมต่อ Sensor ต่างๆ เข้ากับ Port GPIO

จากภาพที่ 3.2 ทางผู้จัดทำได้ทำการเชื่อมต่อ Sensor ดังนี้

a. DHT11 จะมี 3 Port ในการเชื่อมต่อ VCC , DATA , GND ซึ่งให้เชื่อมต่อ VCC เข้าที่ GPIO Port 2 VCC 5 V และ GND เข้ากับ GPIO Port 6 ส่วน DATA INPUT ทางผู้จัดได้เลือก Port GPIO 7 ใช้ในการรับข้อมูล Temperature และ Humidity จาก DHT11



ภาพที่ 3.3 การเชื่อมต่อ DHT11 เข้ากับ Raspberry PI

และเขียน Code การทำงานแบ่งตามส่วนต่างๆ ดังนี้

ตั้งค่า Threshold ของอุณหภูมิและความชื้น

```
TEMPCRITICAL = 50
TEMPMAJOR = 40
TEMPNORMAL = 30
HUMUCRITICAL = 90
HUMUMAJOR = 70
HUMUNORMAL = 50
```

ดำเนินการรับค่าจาก Sensor DHT11

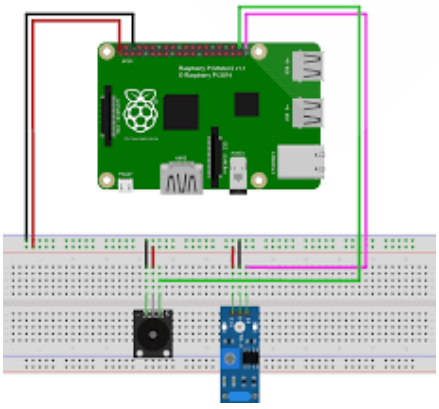
```
humidity, temperature = Adafruit_DHT.read_retry(11, 4)
```

เช็คค่า Humidity และ Temperature ว่าเกินค่า Threshold ไหม ถ้าเกินก็ส่งค่าออกไปยังอุปกรณ์ Analyzer

```
if temperature>=TEMPCRITICAL and TEMPSTAT<>"Abnormal-2":
        MESSAGE = socket.gethostname() + "|801|" + ALARMDATE +
        "|Critical|Room High Temperature|Temperature :" + str(temperature) + "|Env"
        sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
        sock.sendto(MESSAGE, (UDP_IP, UDP_PORT))
elif temperature>=TEMPNORMAL and temperature<=TEMPMAJOR and
TEMPSTAT<>"Abnormal-1":
          MESSAGE = socket.gethostname() + "|802|" + ALARMDATE +
        "|Major|Room High Temperature|Temperature :" + str(temperature) + "|Env"
          sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
          sock.sendto(MESSAGE, (UDP_IP, UDP_PORT))
elif temperature<TEMPNORMAL and TEMPSTAT<>"Normal":
        MESSAGE = socket.gethostname() + "|803|" + ALARMDATE + "|Clear|Room
        Normal Temperature|Temperature :" + str(temperature) + "|Env"
        sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
        sock.sendto(MESSAGE, (UDP_IP, UDP_PORT))
```

```
        if humidity>=HUMUCRITICAL and HUMUSTAT<>"Abnormal-2":
                MESSAGE = socket.gethostname() + "|811|" + ALARMDATE +
                "|Critical|Room High Humidity|Humidity :" + str(humidity) + "|Env"
                sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
                sock.sendto(MESSAGE, (UDP_IP, UDP_PORT))
    elif humidity>=HUMUNORMAL and humidity<=HUMUMAJOR and
HUMUSTAT<>"Abnormal-1":
                MESSAGE = socket.gethostname() + "|812|" + ALARMDATE + "|Major|Room
                High Humidity|Humidity :" + str(humidity) + "|Env"
                sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
                sock.sendto(MESSAGE, (UDP_IP, UDP_PORT))
                HUMUSTAT = "Abnormal-1"
    elif humidity<HUMUNORMAL and HUMUSTAT<>"Normal":
                MESSAGE = socket.gethostname() + "|813|" + ALARMDATE + "|Clear|Room
                Normal Humidity|Humidity :" + str(humidity) + "|Env"
                sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
                sock.sendto(MESSAGE, (UDP_IP, UDP_PORT))
                HUMUSTAT = "Normal"
```

b. SW-420 จะมี 3 Port ในการเชื่อมต่อ VCC , DATA , GND ซึ่งให้เชื่อมต่อ VCC เข้าที่ GPIO Port 2 VCC 5 V และ GND เข้ากับ GPIO Port 6 ส่วน DATA INPUT ทางผู้จัดทำได้เลือก Port GPIO 11 ใช้ในการรับข้อมูล Trigger Vibration จาก Sensor



ภาพที่ 3.4 การเชื่อมต่อ SW-420 เข้ากับ Raspberry Pi

และเขียน Code การทำงานแบ่งตามส่วนต่างๆ ดังนี้

ตั้งค่า Channel ในการเชื่อมต่อและกำหนดค่าต่างๆ

```
channel = 11
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(channel, GPIO.IN)
```

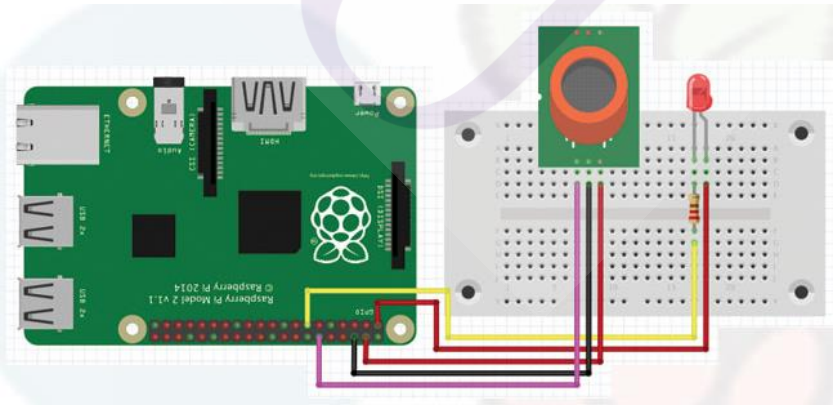
เขียน Sub ในการทำงานเมื่อเกิด Event ของอุปกรณ์เซ็นเซอร์ขึ้น

```
def callback(channel):
        MESSAGE = socket.gethostname() + "|201|" + ALARMDATE + "|Critical|Door
        Opening|Door Alarm|Env"
        sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
        sock.sendto(MESSAGE, (UDP_IP, UDP_PORT))
```

ตั้งค่าให้อุปกรณ์ Sensor ทำงานเมื่อมี Input เข้ามา

```
GPIO.add_event_detect(channel, GPIO.BOTH, bouncetime=500)
GPIO.add_event_callback(channel, callback)
```

c. MQ-7 จะมี 3 Port ในการเชื่อมต่อ VCC , DATA , GND ซึ่งให้เชื่อมต่อ VCC เข้าที่ GPIO Port 2 VCC 5 V และ GND เข้ากับ GPIO Port 6 ส่วน DATA INPUT ทางผู้จัดได้เลือก Port GPIO 13 ใช้ในการรับข้อมูล Trigger GAS CO_2 จาก Sensor



ภาพที่ 3.5 การเชื่อมต่อ MQ-7 เข้ากับ Raspberry Pi

และเขียน Code การทำงานแบ่งตามส่วนต่างๆ ดังนี้

ตั้งค่า Channel ในการเชื่อมต่อและกำหนดค่าต่างๆ

```
channel = 13
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(channel, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
```

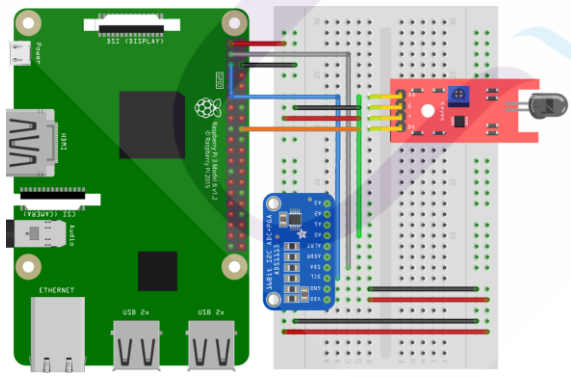
เขียน Sub ในการทำงานเมื่อเกิด Event ของอุปกรณ์เซ็นเซอร์ขึ้น

```
def action(pin):
        MESSAGE = socket.gethostname() + "|701|" + ALARMDATE + "|Critical|CO Senser
        Detected|Smoke Alarm|Env"
        sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
        sock.sendto(MESSAGE, (UDP_IP, UDP_PORT))
```

ตั้งค่าให้อุปกรณ์ Sensor ทำงานเมื่อมี Input เข้ามา

```
GPIO.add_event_detect(channel, GPIO.RISING, bouncetime=500)
GPIO.add_event_callback(channel, action)
```

d. STM-32 จะมี 3 Port ในการเชื่อมต่อ VCC , DATA , GND ซึ่งให้เชื่อมต่อ VCC เข้าที่ GPIO Port 2 VCC 5 V และ GND เข้ากับ GPIO Port 6 ส่วน DATA INPUT ทางผู้จัดได้เลือก Port GPIO 15 ใช้ในการรับข้อมูล Trigger Flame (Fire) จาก Sensor



ภาพที่ 3.6 การเชื่อมต่อ STM-32 เข้ากับ Raspberry Pi

และเขียน Code การทำงานแบ่งตามส่วนต่างๆ ดังนี้

ตั้งค่า Channel ในการเชื่อมต่อและกำหนดค่าต่างๆ

```
channel = 15
GPIO.setwarnings(False)
```

```
GPIO.setmode(GPIO.BOARD)
GPIO.setup(channel, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
```

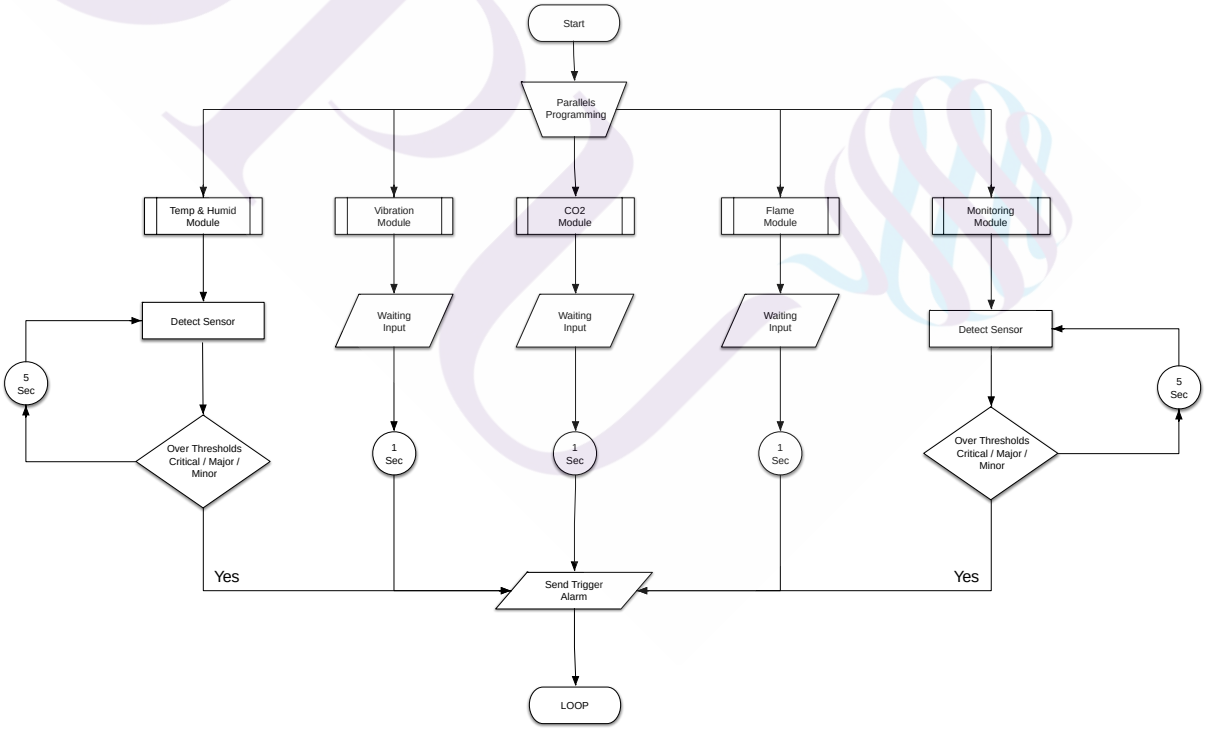
เขียน Sub ในการทำงานเมื่อเกิด Event ของอุปกรณ์เซ็นเซอร์ขึ้น

```
def action(pin):
        MESSAGE = socket.gethostname() + "|601|" + ALARMDATE + "|Critical|Flame Senser
        Detected|Fire Alarm|Env"
        sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
        sock.sendto(MESSAGE, (UDP_IP, UDP_PORT))
```

ตั้งค่าให้อุปกรณ์ Sensor ทำงานเมื่อมี Input เข้ามา

```
GPIO.add_event_detect(channel, GPIO.RISING, bouncetime=500)
GPIO.add_event_callback(channel, action)
```

โดยเมื่อ Sensor ต่างๆ มาทำงานร่วมกันระบบทั้งหมดจะทำงานพร้อมกันแบบ Parallels และจะทำงานไปเรื่อยดังขั้นตอนการดำเนินงานเป็น Flow Chart ดังภาพที่ 3.7



ภาพที่ 3.7 Flowchart การทำงานของ Raspberry Pi ที่ทำหน้าที่ส่ง Alarm

3. ดำเนินการเขียน Code Analyzer บนระบบปฏิบัติการ Windows ซึ่งทางผู้จัดทำให้ได้ใช้ภาษา Visual Basic .NET ในการพัฒนา และพัฒนาด้วย IDE Visual Studio 2013 บน .Net Framework 4.0

โดยหลักการทำงานของโปรแกรมจะแบ่งเป็น 3 Module หลักๆ คือ

a. Receive Alarm Module

จะทำหน้าที่รับ Alarm เข้ามาในระบบผ่าน Protocol UDP ตาม Port ที่ได้ Configuration ไว้เมื่อรับ Alarm เข้ามาแล้วจะทำการเก็บไว้ในระบบตาม เวลาที่ได้รับโดยมี Code การทำงานตามนี้

```
Private Sub bgRecieve_DoWork
    Dim receivingUdpClient As New UdpClient(SocketNO
    Dim RemoteIpEndPoint As New IPEndPoint(IPAddress.Any, 0)
      Try
        Dim receiveBytes As [Byte]() = receivingUdpClient.Receive(RemoteIpEndPoint)
        Dim returnData As String = Encoding.ASCII.GetString(receiveBytes
        AnalyzerAlarm(returnData.ToString(), RemoteIpEndPoint.Address.ToString(),
RemoteIpEndPoint.Port.ToString())
      Catch ex As Exception
      End Try
    Loop
    StatusRecieve = False
End Sub
Private Sub AnalyzerAlarm(ByVal Alarm As String, ByVal RetIP As String, ByVal RetPort As
String)
      Dim AlarmLine() As String = Alarm.Split("|")
      Dim strConn As String = "Server=.;Database=db_Analyzer;Trusted_Connection=True;"
      Dim Conn As New SqlConnection
      Dim Insql As String
      Dim sqltoCMD As SqlCommand = New SqlCommand
 Insql = "INSERT INTO [tb_accuringalarm]([hostname],[alarmID],[alarmdatetime],[alarmsever
ity],[alarmname],[alarmdescription],[alarmtype],[remoteip],[remoteport]) VALUES ('" & Alarm
```

```
Line(0) & "','" & AlarmLine(1) & "','" & AlarmLine(2) & "','" & AlarmLine(3) & "','" & Alarm
Line(4) & "','" & AlarmLine(5) & "','" & AlarmLine(6) & "','" & RetIP & "','" & RetPort & "')"
        With sqltoCMD  .CommandType = CommandType.Text
          .CommandText = Insql.ToString
          .Connection = Conn
          .ExecuteNonQuery()
        End With
      End If
    End Sub
```

b. Filter Alarm Module

ทำหน้าที่ Process Alarm ที่เข้ามาเป็น Alarm ประเภทอะไร และต้องส่งให้ใครบ้าง โดยสามารถจำแนกตาม Severity , Alarm ID , Alarm Name และเลือกส่งได้ทั้งเป็น Group (Line) และส่งได้ทั้งรายบุคคลตามหน้าที่ที่รับผิดชอบ เมื่อ Filter แล้วจะนำ Alarm ที่ต้องส่งทั้งหมดไปเก็บไว้ในถังข้อมูลของแต่ละ API ที่ต้องทำการจัดส่งไปยังปลายทาง ซึ่งจะมี Code การทำงานแบ่งตามปลายทางที่ต้องส่งดังนี้ดังนี้

- LINE Notify

Query ข้อมูลของ Group Line ที่ต้องการรับ Alarm นี้

```
Insql = "SELECT [LineGroup] FROM [tb_Bearer] WHERE [AlarmID] LIKE '%" &
AlarmID & "|%' AND [AlarmHost] LIKE '%" & AlarmHost & "|%' AND
[AlarmSeverity] LIKE '%" & AlarmSeverity & "|%'"
```

ดำเนินการ Insert ข้อมูลลง Table [tb_ LineSending]

```
For j As Integer = 0 To ds.Tables("Group").Rows.Count - 1
        Dim GroupName As String = ds.Tables("Group").Rows(j).Item("LineGroup")
        SqlExcuteCmd = "INSERT INTO
        [tb_LineSending]([LineMessage],[LineGroup], [LineDateTime]) VALUES ('" &
        MessageSend & "','" & GroupName & "','" & DateTime.Now & "')"
Next
```

- SMS Gateway

Query ข้อมูลของเบอร์ที่ต้องการรับ Alarm นี้

```
Insql = "SELECT [Member] FROM [tb_SMSBearer] WHERE [AlarmID] LIKE '%" &
AlarmID & "|%' AND [AlarmHost] LIKE '%" & AlarmHost & "|%' AND
[AlarmSeverity] LIKE '%" & AlarmSeverity & "|%'"
```

ดำเนินการ Insert ข้อมูลลง Table [tb_ SMSSending]

```
SqlExcuteCmd = "INSERT INTO [tb_SMSending]([SendOriginate],[SendMSISDN],
[SendMessage],[SendLang],[SendDatetime],[SendLongSMS]) VALUES ('เบอร์ต้น
ทาง','" & ds.Tables("SMS").Rows(j).Item("Member") & "','" & MessageSend &
"','ENGLISH','" & DateTime.Now & "','0')"
```

- Mail Gateway

Query ข้อมูลของ E-Mail ที่ต้องการรับ Alarm นี้

```
Insql = "SELECT [MailAddress] FROM [tb_MailBearer] WHERE [AlarmID] LIKE '%"
& AlarmID & "|%' AND [AlarmHost] LIKE '%" & AlarmHost & "|%' AND
[AlarmSeverity] LIKE '%" & AlarmSeverity & "|%'"
```

ดำเนินการ Insert ข้อมูลลง Table [tb_MailSending]

```
SqlExcuteCmd = "INSERT INTO [tb_MailSending]([MailDestination],[MailSubject]
,[MailBody],[MailDatetime]) VALUES ('" & ds.Tables("Mail").Rows(j).Item("Mail
Address") & "','" & MailSubject & "','" & MailBody & "','" & DateTime.Now & "')"
```

c. Sending API Module

ทำหน้าที่ติดต่อกับ API ภายนอก เช่น Line Notify API , SMS Gateway และ Mail Server โดยการทำงานคือ อ่านข้อมูล Alarm ทั้งหมดที่ต้องส่งไปยังปลายทางในถังข้อมูลของ API นั้นๆ และเมื่อได้ข้อมูลมาแล้ว ก็จะทำการจัดส่งให้ปลายทางตาม API ที่ติดตั้งไว้แบ่งตาม API ดังนี้

LINE NOTIFY API แสดง Code ที่ใช้ในการส่ง Alarm ออกไปยัง Line Notify API การเรียกใช้ Line Notify API

ACT_SendMessage(Messages, bearerToken)

ประกาศค่าตัวแปร Line Notify API

```
Dim requestData = DirectCast(WebRequest.Create("https://notify-api.line.me/api/notify"),
HttpWebRequest)
    Dim postData = String.Format("message=" & lineMessage)
    Dim data = Encoding.UTF8.GetBytes(postData)
    requestData.Method = "POST"
    requestData.ContentType = "application/x-www-form-urlencoded"
    requestData.ContentLength = data.Length
    requestData.Headers.Add("Authorization", "Bearer " & BearerToken)
    Using stream = requestData.GetRequestStream()
    stream.Write(data, 0, data.Length)
    End Using
    Dim response = DirectCast(requestData.GetResponse(), HttpWebResponse)
    Dim responseString = New StreamReader(response.GetResponseStream()).ReadToEnd()
```

SMS API แสดง Code ที่ใช้ในการส่ง Alarm ออกไปยัง SMS Gateway API

การเรียกใช้งาน Web Services SMS API

```
mySendSMS.SendSMS(username,password,source,destination,message,langague,Long Message)
```

Add References Web Services SMS API

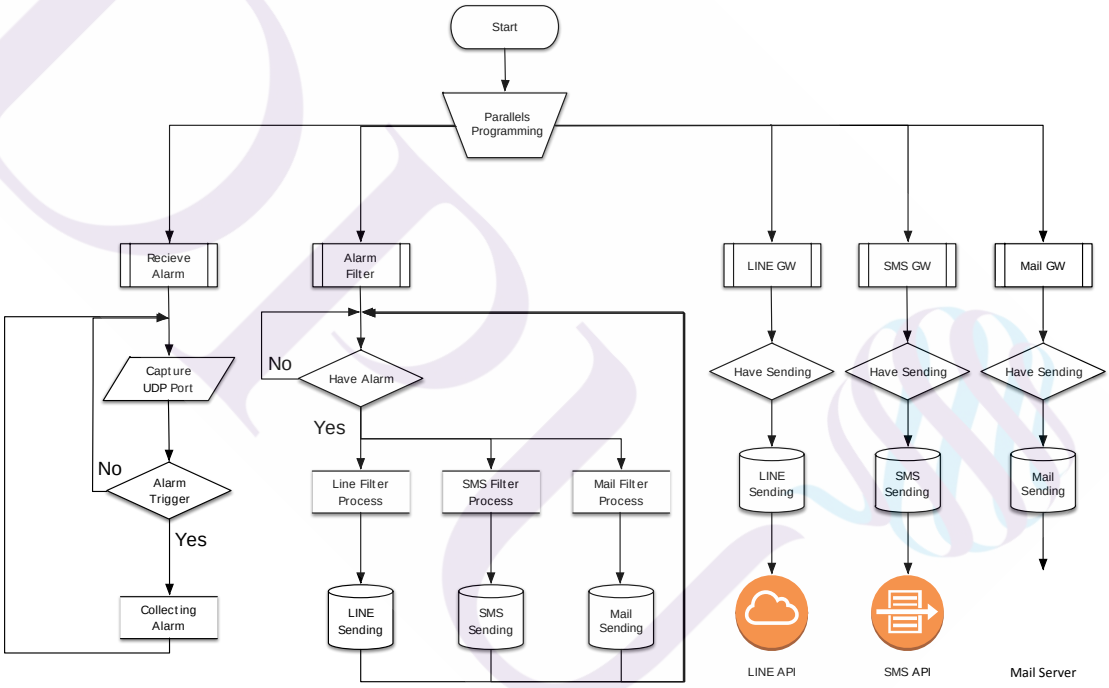
```
    <client>
        <endpoint address="http://myservice.mybycat.com/myiService/sms.asmx"
        binding="basicHttpBinding" bindingConfiguration="smsSoap"
        contract="SMS.smsSoap" name="smsSoap" />
    </client>
```

Mail API แสดง Code ที่ใช้ในการส่ง Alarm ออกไปยัง Mail SMTP API

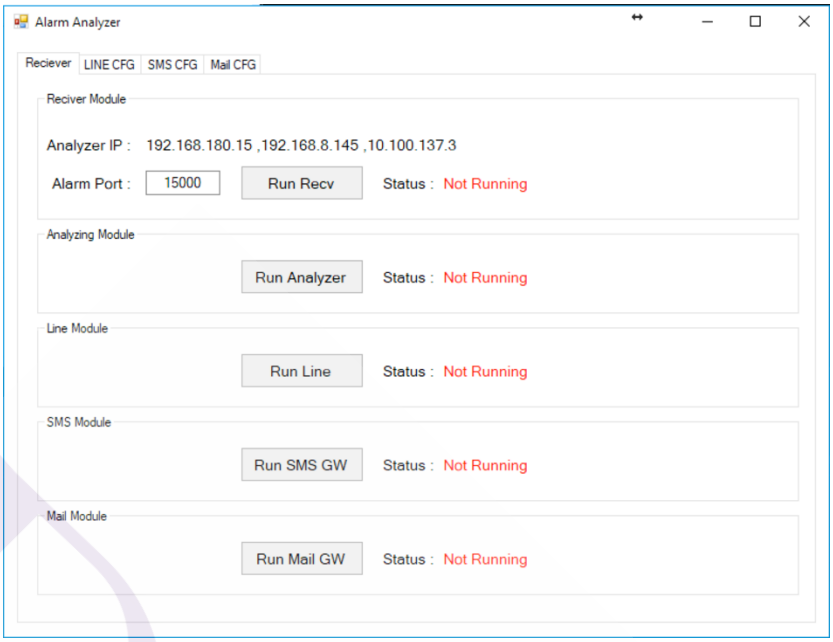
แสดง Code ที่ใช้เชื่อมต่อกับ Mail Server

```
    Dim Mail As New MailMessage
    Dim SMTP As New SmtpClient("smtp.gmail.com")
```

```
Mail.Subject = ds.Tables("MailSending").Rows(i).Item("MailSubject")
Mail.From = New MailAddress("NineAnaly121@gmail.com")
SMTP.Credentials = New  System.Net.NetworkCredential
("NineAnaly121@gmail.com", "Nine@myPro") '<-- Password Here
Mail.To.Add(ds.Tables("MailSending").Rows(i).Item("MailDestination"))
Mail.IsBodyHtml = True
Mail.Body = ds.Tables("MailSending").Rows(i).Item("MailBody")
SMTP.EnableSsl = True
SMTP.Port = "587"
SMTP.Send(Mail)
```

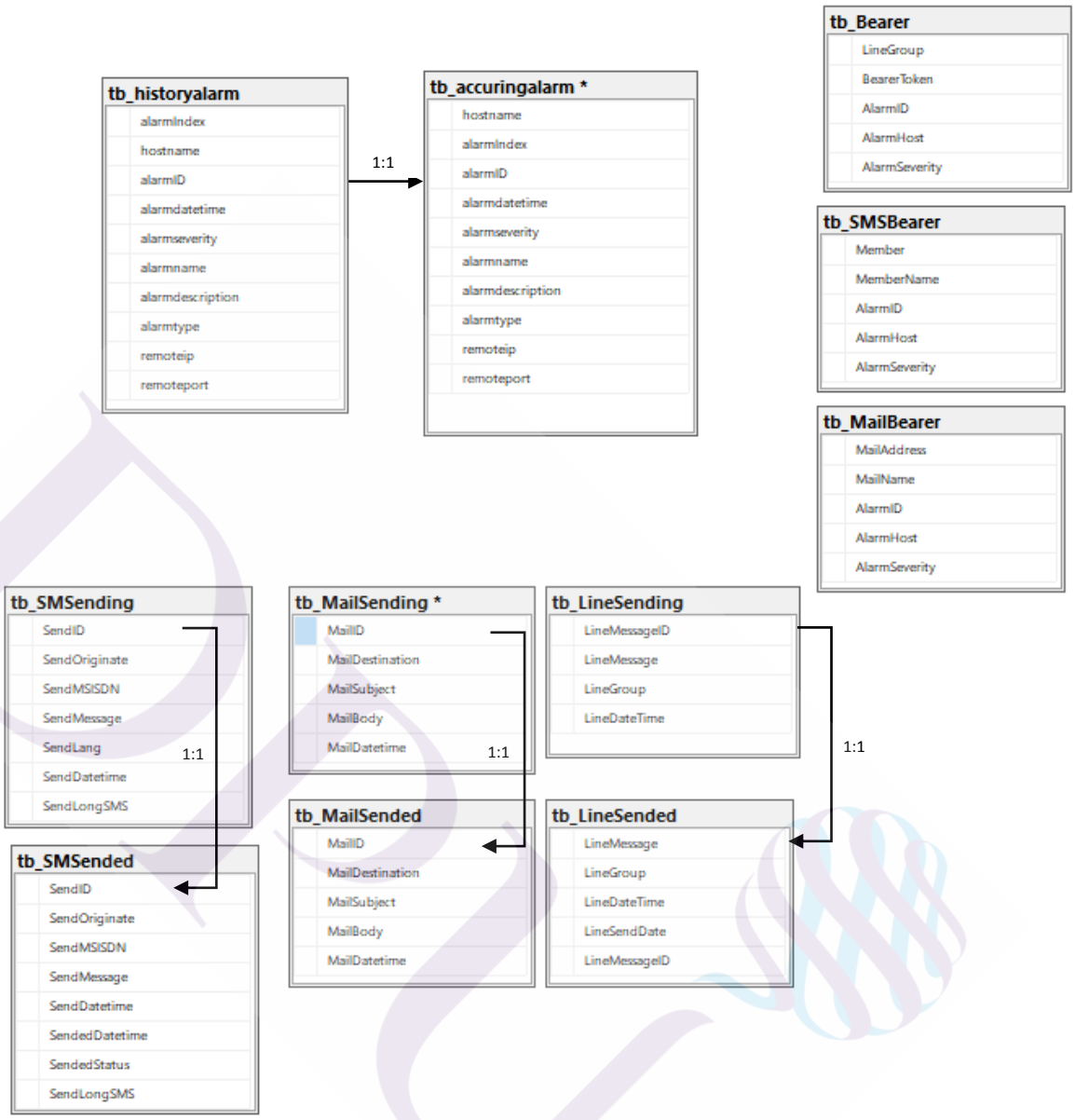


ภาพที่ 3.8 Flowchart การทำงานของโปรแกรม Analyzer



ภาพที่ 3.9 โปรแกรม Analyzer ที่พัฒนาขึ้นเพื่อรับข้อมูล Alarm และส่งต่อ

4. ฐานข้อมูลที่ผู้จัดทำนำมาใช้คือ SQL Server 2016 Express Edition ซึ่งมีการออกแบบฐานข้อมูลดังภาพต่อไปนี้



ภาพที่ 3.10 Diagram ของฐานข้อมูลที่นำมาใช้

- tb_accuringalarm เป็น Table ที่ใช้เก็บข้อมูล Alarm ที่รับเข้ามาทั้งหมดก่อนที่จะดำเนินการเข้า Process ของ Analyzer เพื่อ Filter และจัดการส่งให้แต่ละ API

- tb_historyalarm เป็น Table ที่ใช้เก็บข้อมูล Alarm ที่หลังจากการดำเนินการ Process เรียบร้อยแล้ว

- tb_Bearer เป็น Table ที่ใช้เก็บ Token ของ Line Group ID เพื่อใช้ในการ Register กับ Line Notity API ส่ง Line Message และสามารถจำแนก Filter ว่า Line Group นี้จะรับ Alarm ID , AlarmHost , AlarmSeverity อะไรบ้าง

- tb_SMSBearer เป็น Table ที่ใช้เก็บ Profile เบอร์โทรที่ต้องการส่ง SMS ของบุคคลที่ต้องการรับ Alarm และ Filter ว่าบุคคลนั้นต้องการที่จะรับ Alarm ID , AlarmHost , AlarmSeverity อะไรบ้าง

- tb_MailBearer เป็น Table ที่ใช้เก็บ Profile E-mail ที่ต้องการส่ง Mail ของบุคคลที่ต้องการรับ Alarm และ Filter ว่าบุคคลนั้นต้องการที่จะรับ Alarm ID , AlarmHost , AlarmSeverity อะไรบ้าง

- tb_LineSending,tb_SMSSending,tb_MailSending เป็น Table ที่เป็น Buffer หลังที่ Analyzer Process แล้วว่าต้องส่ง LINE , SMS หรือ E-mail ไปที่ไหนบ้างเพื่อรอ Module Line GW , SMS GW , Mail GW มาดึงข้อมูลใน Table นี้ไปส่งข้อมูล

- tb_LineSended,tb_SMSSended,tb_MailSended เป็น Table ที่หลังจาก API ต่างๆ ดำเนินการส่ง LINE , SMS หรือ E-mail เรียบร้อยแล้วจะนำข้อมูล History มาเก็บไว้บน Table นี้ เพื่อสามารถใช้ดู Log ย้อนหลังได้

บทที่ 4
ผลการดำเนินการและวิเคราะห์ข้อมูล

การจัดทำโครงการ Alarm Monitoring with IOT มีวัตถุประสงค์เพื่อแก้ไขปัญหาที่เกิดขึ้นจากการที่ทางผู้จัดทำต้องการที่จะ Monitoring สภาพแวดล้อมของอุปกรณ์ที่ต้องดูแลให้สามารถรับรู้ได้ทันท่วงทีที่เกิดปัญหาขึ้น จำได้ดำเนินการเข้าไปแก้ปัญหาได้ทันทีและส่งผลกระทบต่อผู้ใช้บริการน้อยที่สุด

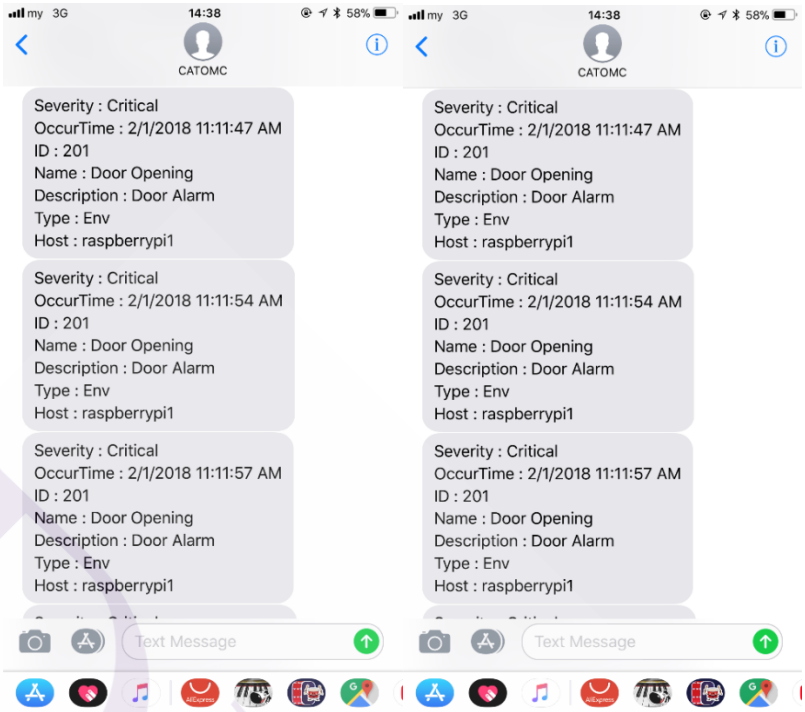
4.1 ผลการพัฒนาระบบ IOT

การพัฒนาระบบ IOT ซึ่งทางผู้จัดทำได้ใช้อุปกรณ์ Raspberry Pi II พร้อมกับระบบปฏิบัติการ Raspbian นั้น ได้ดำเนินการตามขั้นตอนการดำเนินงานที่เสนอในบทที่ 3 แล้ว

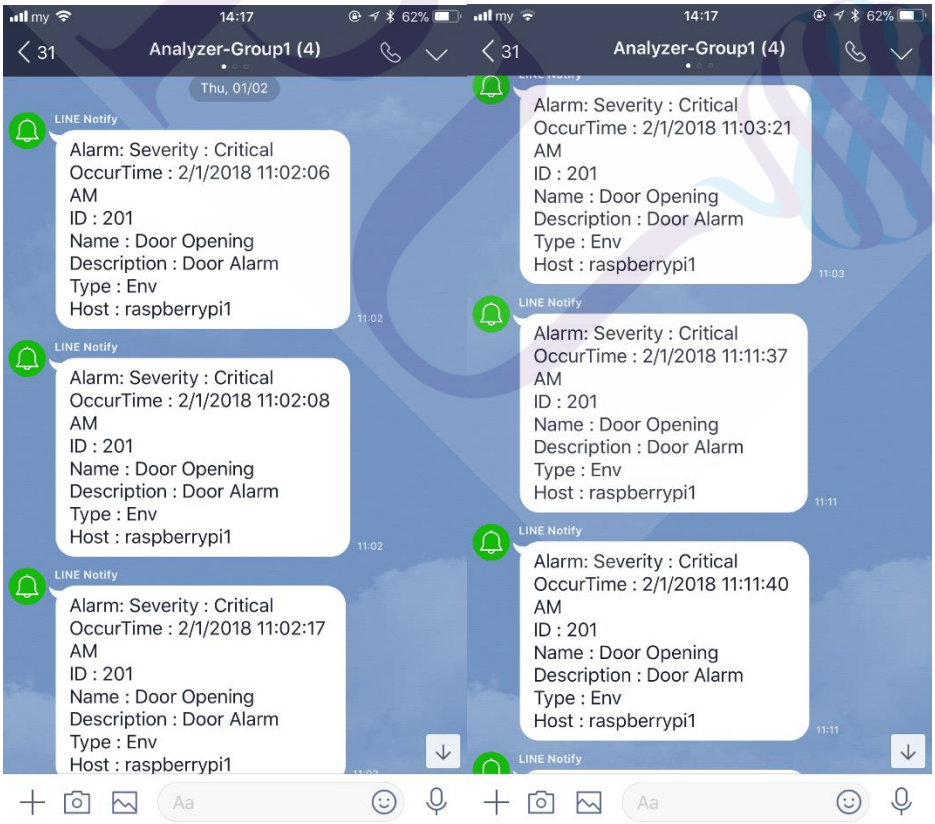
โดยผู้ใช้จะต้องดำเนินการ Configuration Hostname , IP และ Port ของอุปกรณ์ Analyzer ปลายทางเพื่อให้ระบบ IOT สามารถส่งข้อมูลไปหา Analyzer ได้อย่างถูกต้อง

การทำงานของระบบ IOT เมื่อผู้ใช้งานเปิดอุปกรณ์ IOT และได้ตั้งค่าอุปกรณ์ทั้งหมดแล้ว Process ทั้งหมดจะ Active ขึ้นมาทำงานโดยอัตโนมัติจาก Crontab Script ที่ตั้งไว้เมื่อทำการ Reboot ระบบ แต่ถ้าระบบทำงานผิดพลาด ผู้ใช้งานสามารถเข้าไปดำเนินการ Stop และ Start จาก Process Script ที่ได้สร้างไว้ (launcher_start.sh , launcher_stop.sh)

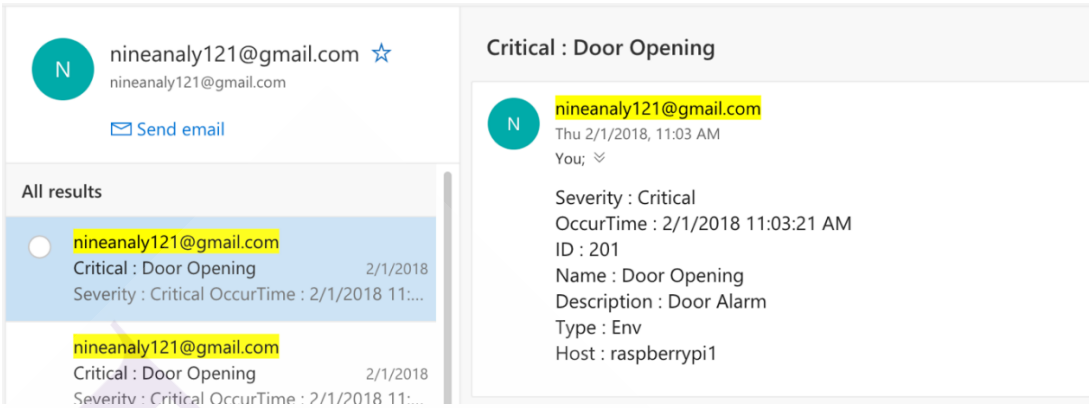
ระบบสามารถส่งข้อมูลไปยังปลายทางได้อย่างถูกต้อง



ภาพที่ 4.1 ระบบสามารถส่ง SMS ไปตามข้อมูลที่ Filter ได้อย่างถูกต้อง

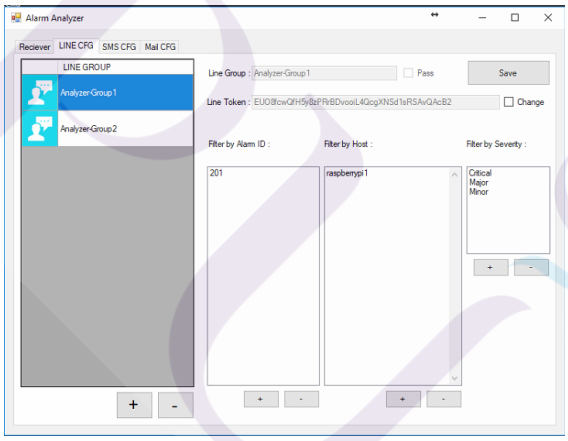


ภาพที่ 4.2 การส่งข้อมูลผ่าน LINE Notify API ที่ Filter Alarm โดย LINE Group

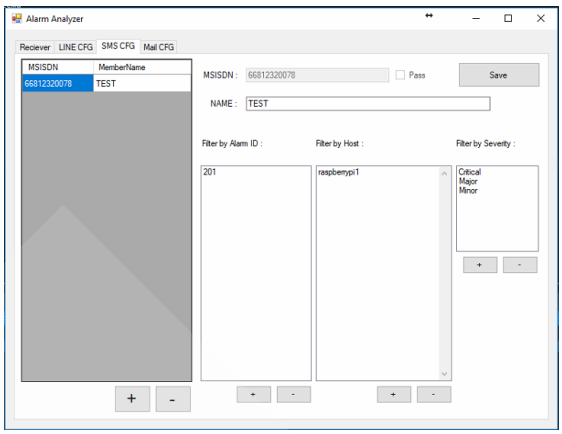


ภาพที่ 4.3 การส่งข้อมูลผ่าน SMTP (E-Mail) ตามข้อมูลที่ Filter

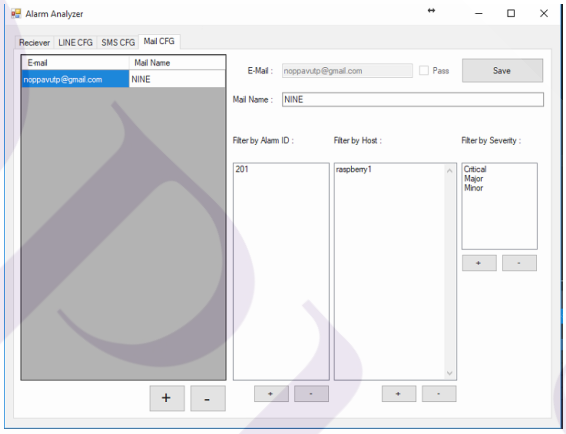
โดยทางผู้จัดทำได้จำแนก Filter การรับ Alarm ได้ผ่านทาง Native Application ได้ทันที



ภาพที่ 4.4 การ Filter ข้อมูลที่จะส่งไปตาม LINE Group ต่างๆ



ภาพที่ 4.5 การ Filter ข้อมูลที่จะส่ง SMS ไปตามเบอร์ที่ตั้งค่าไว้



ภาพที่ 4.6 การ Filter ข้อมูลที่จะส่ง E-mail ไปตาม Address ที่ตั้งค่าไว้

จากการจำลองการทำงานของระบบ ผลการทำงานตามตารางต่อไปนี้

ตารางที่ 4.1 การรับข้อมูลจาก Sensor และส่งไปยัง API

Sensor / Alarm	Local	DHT11	SW-420	MQ-7	STM-32
Line	✓	✓	✓	✓	✓
SMS	✓	✓	✓	✓	✓
E-Mail	✓	✓	✓	✓	✓

ตารางแสดงการได้รับ Alarm Monitoring ผ่านระบบ Line , SMS และ E-mail

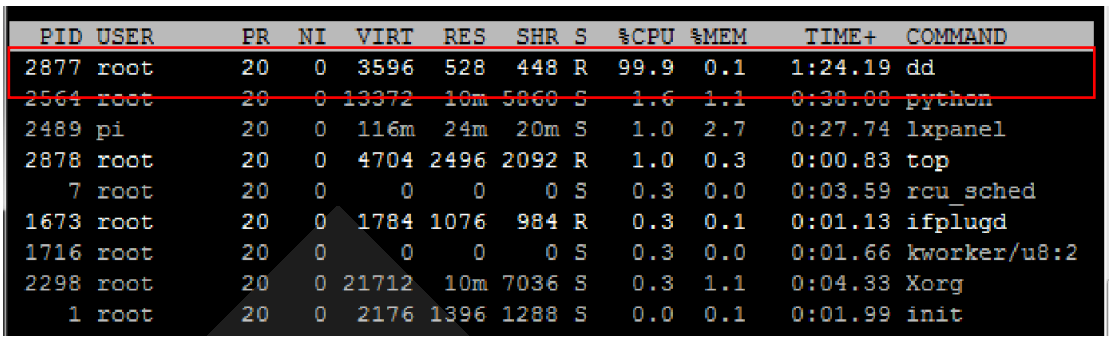
จากผลการทดลองเบื้องต้นพบว่าระบบ Alarm Monitoring สามารถรับค่าจาก Sensor ต่างๆ (Local Environment, DHT11 , SW-420 , MQ-7 และ STM-32) และสามารถส่งข้อมูลไปยังปลายทางตามที่ตั้งค่าไว้ได้ และจากผลการทดลองเบื้องต้น ทางผู้จัดทำจึงจำแนกการทดลองไปที่แต่ละตัว Sensor โดยทำการทดลองดังนี้

1. ทดลองส่ง Alarm CPU Load ของอุปกรณ์ Raspberry Pi ไปยังปลายทางทั้ง 3 ช่องพร้อมกันและจำแนกทีละช่องทาง
2. ทดลองส่ง Alarm จาก Sensor DHT11 ของอุปกรณ์ Raspberry Pi ไปยังปลายทางทั้ง 3 ช่องพร้อมกันและจำแนกทีละช่องทาง
3. ทดลองส่ง Alarm จาก Sensor SW-420 ของอุปกรณ์ Raspberry Pi ไปยังปลายทางทั้ง 3 ช่องพร้อมกันและจำแนกทีละช่องทาง
4. ทดลองส่ง Alarm จาก Sensor MQ-7 ของอุปกรณ์ Raspberry Pi ไปยังปลายทางทั้ง 3 ช่องพร้อมกันและจำแนกทีละช่องทาง
5. ทดลองส่ง Alarm จาก Sensor STM-32 ของอุปกรณ์ Raspberry Pi ไปยังปลายทางทั้ง 3 ช่องพร้อมกันและจำแนกทีละช่องทาง

การทดลองที่ 1

ทดลอง Run Command “dd if=/dev/zero of=/dev/null” บนบอร์ด Raspberry Pi ที่เชื่อมต่ออยู่กับระบบ Analyzer และเมื่ออุปกรณ์ Raspberry Pi มี CPU Usage สูงขึ้น จนถึงค่า Threshold ที่ตั้งไว้ อุปกรณ์ Raspberry Pi จะทำการส่ง Alarm Data ไปยังระบบ Analyzer แล้วสังเกต Alarm ที่ได้รับ

ภาพที่ 4.7 จำลองการทดลอง CPU Load เพื่อให้ระบบส่ง Alarm มาที่ Analyzer



PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
2877	root	20	0	3596	528	448	R	99.9	0.1	1:24.19	dd
2564	root	20	0	13372	10m	5860	S	1.6	1.1	0:38.08	python
2489	pi	20	0	116m	24m	20m	S	1.0	2.7	0:27.74	lxpanel
2878	root	20	0	4704	2496	2092	R	1.0	0.3	0:00.83	top
7	root	20	0	0	0	0	S	0.3	0.0	0:03.59	rcu_sched
1673	root	20	0	1784	1076	984	R	0.3	0.1	0:01.13	ifplugd
1716	root	20	0	0	0	0	S	0.3	0.0	0:01.66	kworker/u8:2
2298	root	20	0	21712	10m	7036	S	0.3	1.1	0:04.33	Xorg
1	root	20	0	2176	1396	1288	S	0.0	0.1	0:01.99	init

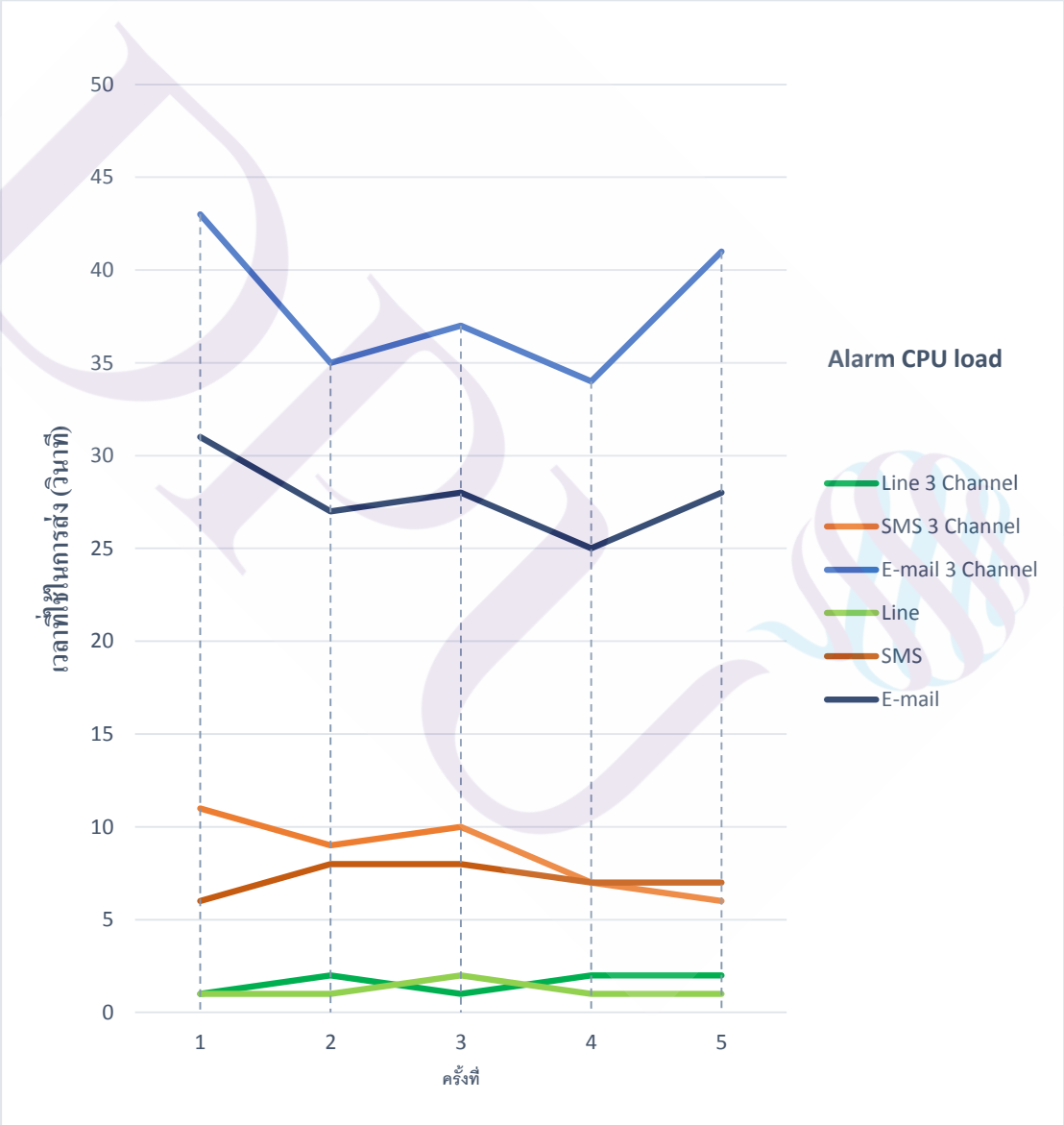
ภาพที่ 4.8 จำลองเมื่อสถานะ CPU Load สูงขึ้น

ตัวอย่างวันที่ 20 มกราคม 2561

ตารางที่ 4.2 เวลาที่จำลองส่งข้อมูล CPU Load ที่เกิดขึ้น ในช่วงเวลาตั้งแต่ที่เกิด Alarm ไปยังปลายทางทั้ง 3 ช่องทางและจำแนกทีละช่องทาง

เวลาที่เกิด Alarm	เวลาที่ได้รับ Alarm			หมายเหตุ
	Line	SMS	E-mail	
09:10:14	09:10:15	09:10:25	09:10:57	ส่งผ่าน Line , SMS , E-mail
09:15:08	09:15:10	09:15:17	09:15:43	ส่งผ่าน Line , SMS , E-mail
09:20:17	09:20:18	09:20:27	09:20:54	ส่งผ่าน Line , SMS , E-mail
09:25:13	09:25:15	09:25:20	09:25:47	ส่งผ่าน Line , SMS , E-mail
09:30:08	09:30:10	09:30:14	09:30:59	ส่งผ่าน Line , SMS , E-mail
09:40:10	09:40:11	-	-	ส่งผ่าน Line
09:45:08	09:45:09	-	-	ส่งผ่าน Line
09:50:11	09:50:13	-	-	ส่งผ่าน Line
09:55:10	09:55:11	-	-	ส่งผ่าน Line
10:00:14	10:00:15	-	-	ส่งผ่าน Line
10:20:09	-	10:20:15	-	ส่งผ่าน SMS
10:25:12	-	10:25:20	-	ส่งผ่าน SMS
10:30:14	-	10:30:22	-	ส่งผ่าน SMS
10:35:17	-	10:35:24	-	ส่งผ่าน SMS
10:40:13	-	10:40:20	-	ส่งผ่าน SMS

10:50:12	-	-	10:50:43	ส่งผ่าน E-Mail
10:55:08	-	-	10:55:35	ส่งผ่าน E-Mail
11:00:11	-	-	11:00:39	ส่งผ่าน E-Mail
11:05:12	-	-	11:05:37	ส่งผ่าน E-Mail
11:10:10	-	-	11:10:39	ส่งผ่าน E-Mail



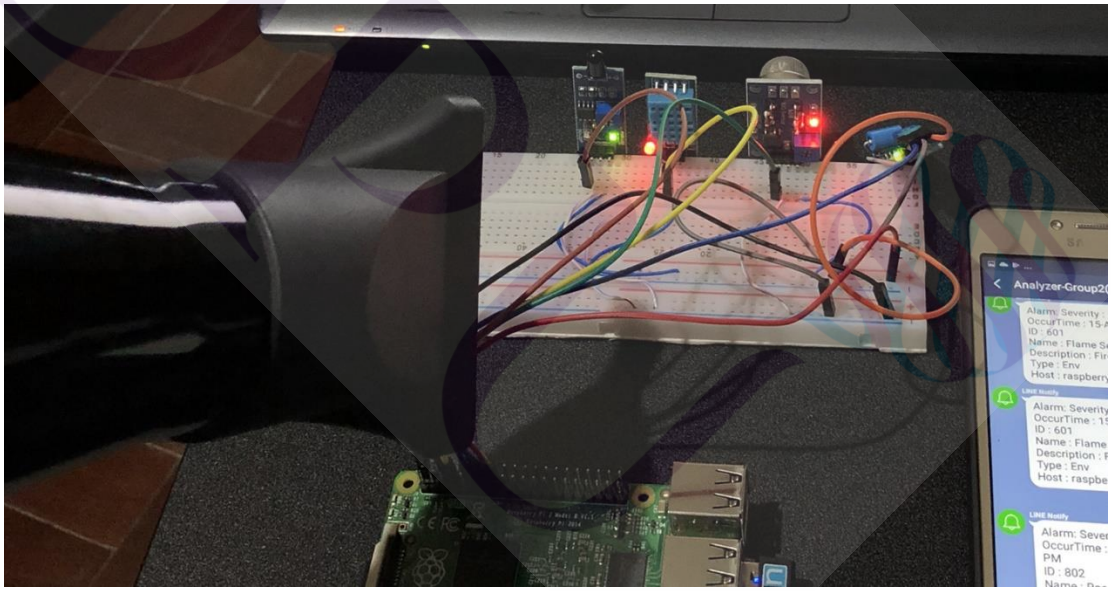
ภาพที่ 4.9 เวลาในการส่งข้อมูล CPU Load ตั้งแต่เกิด Alarm จนถึงผู้รับปลายทาง

สรุปผลการทดลองที่ 1

เมื่อทำการทดลอง Run Command “dd if=/dev/zero of=/dev/null” บนบอร์ด Raspberry Pi แล้วรอให้ CPU ค่อยๆ ทำงาน Load มากขึ้นเรื่อย จะพบว่าเมื่อถึง Threshold ที่ตั้งไว้ Raspberry Pi จะส่ง Alarm Data (Packet UDP) ออกมายัง ระบบ Analyzer และส่งจนถึงปลายทางจะพบว่าการการส่งข้อมูลผ่านทางช่องทาง Line Group (Line Notify) จะสามารถส่งข้อมูลได้ถึงปลายทางเร็วที่สุด และต่อมาคือ SMS และ E-mail ตามลำดับ

การทดลองที่ 2

ทดลองส่ง Alarm จาก Sensor DHT11 ของอุปกรณ์ Raspberry Pi ไปยังปลายทางทั้ง 3 ช่องพร้อมกันและจำแนกทีละช่องทาง โดยการทดลองจะทำการติดตั้ง Sensor DHT11 เข้ากับ Raspberry Pi และใช้อุปกรณ์ทำความร้อน (ไดร์เป่าผม) เป่าจนมีอุณหภูมิที่สูงขึ้นแล้วสังเกต Alarm ที่ได้รับ

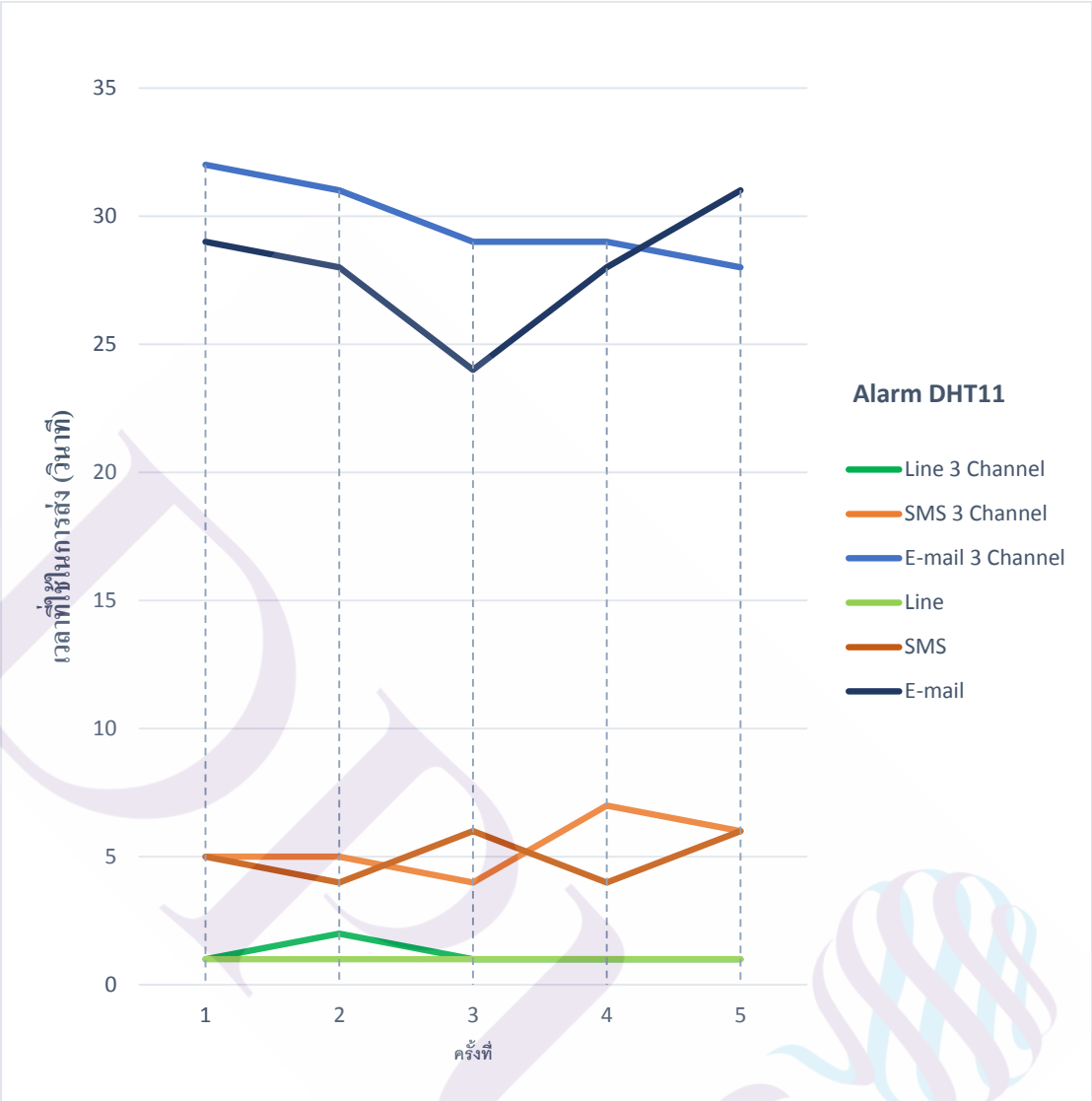


ภาพที่ 4.10 จำลองขณะทำการทดลองเพิ่มอุณหภูมิกับ Sensor DHT11

ตัวอย่างวันที่ 20 มกราคม 2561

ตารางที่ 4.3 เวลาที่จำลองส่งข้อมูล Temp ที่เกิดขึ้น ตั้งแต่ที่เกิด Alarm ไปยังปลายทางทั้ง 3 ช่องทางและจำแนกทีละช่องทาง

เวลาที่เกิด Alarm	เวลาที่ได้รับ Alarm			หมายเหตุ
	Line	SMS	E-mail	
11:30:27	11:30:28	11:30:32	11:30:59	ส่งผ่าน Line , SMS , E-mail
11:35:26	11:35:28	11:35:31	11:35:57	ส่งผ่าน Line , SMS , E-mail
11:40:27	11:40:28	11:40:31	11:40:56	ส่งผ่าน Line , SMS , E-mail
11:45:21	11:45:22	11:45:28	11:45:50	ส่งผ่าน Line , SMS , E-mail
11:50:24	11:50:25	11:50:30	11:50:52	ส่งผ่าน Line , SMS , E-mail
12:00:23	12:00:24	-	-	ส่งผ่าน Line
12:05:21	12:05:22	-	-	ส่งผ่าน Line
12:10:24	12:10:25	-	-	ส่งผ่าน Line
12:15:21	12:15:22	-	-	ส่งผ่าน Line
12:20:22	12:20:23	-	-	ส่งผ่าน Line
12:30:23	-	12:30:28	-	ส่งผ่าน SMS
12:35:27	-	12:35:31	-	ส่งผ่าน SMS
12:40:21	-	12:40:27	-	ส่งผ่าน SMS
12:45:23	-	12:45:27	-	ส่งผ่าน SMS
12:50:20	-	12:50:26	-	ส่งผ่าน SMS
13:00:22	-	-	13:00:51	ส่งผ่าน E-Mail
13:05:21	-	-	13:05:49	ส่งผ่าน E-Mail
13:10:23	-	-	13:10:47	ส่งผ่าน E-Mail
13:15:21	-	-	13:15:49	ส่งผ่าน E-Mail
13:20:19	-	-	13:20:50	ส่งผ่าน E-Mail



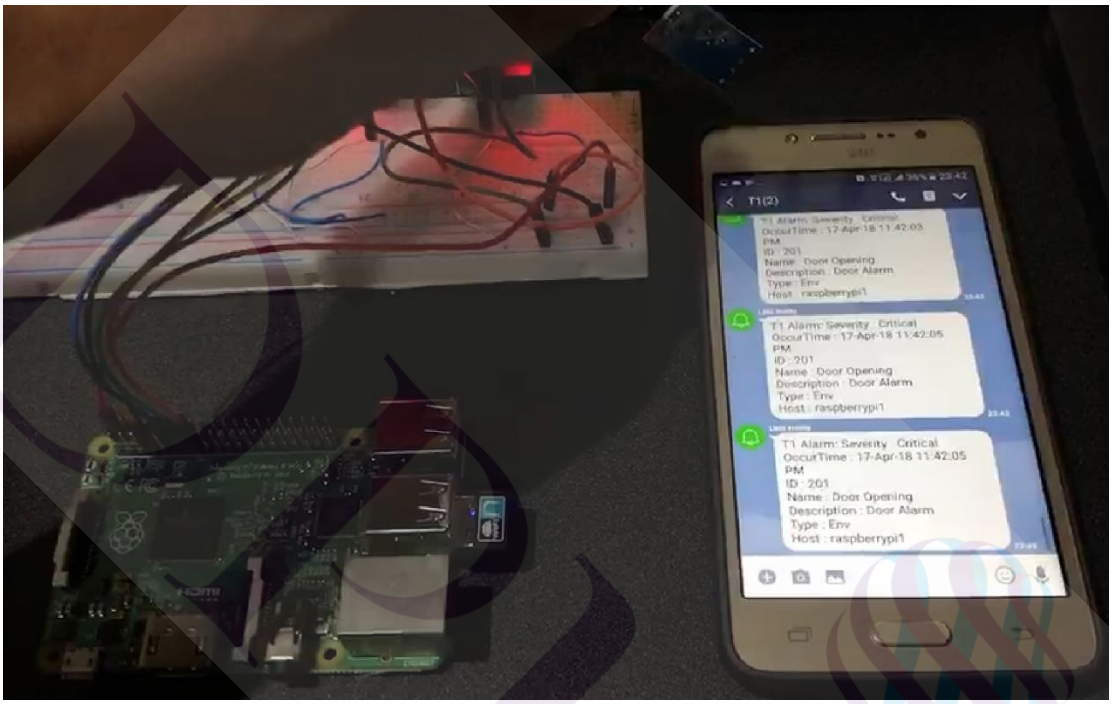
ภาพที่ 4.11 เวลาในการส่งข้อมูล Sensor DHT11 ตั้งแต่เกิด Alarm จนถึงผู้รับปลายทาง

สรุปผลการทดลองที่ 2

เมื่อทำการทดลองเพิ่มอุณหภูมิของ Sensor DHT11 จะเห็นว่าอุปกรณ์ Raspberry pi จะส่งข้อมูล Alarm Data Temperature High Temp ออกมาเมื่อค่า Temperature ถึงค่า Threshold ที่ตั้งค่าไว้บน Raspberry Pi และส่งออกมายัง ออกมายัง ระบบ Analyzer จนถึงปลายทางจะพบว่าการส่งข้อมูลผ่านทางช่องทาง Line Group (Line Notify) จะสามารถส่งข้อมูลได้ถึงปลายทางเร็วที่สุด และต่อมาคือ SMS และ E-mail ตามลำดับ

การทดลองที่ 3

ทดลองส่ง Alarm จาก Sensor SW-420 ของอุปกรณ์ Raspberry Pi ไปยังปลายทางทั้ง 3 ช่องพร้อมกันและจำแนกทีละช่องทาง โดยการทดลองจะทำการติดตั้ง Sensor SW-420 เข้ากับ Raspberry Pi และทำการเขย่าตัว Sensor เพื่อทดลองว่ามีการเปิดประตูเข้ามาแล้วสังเกต Alarm ที่ได้รับ

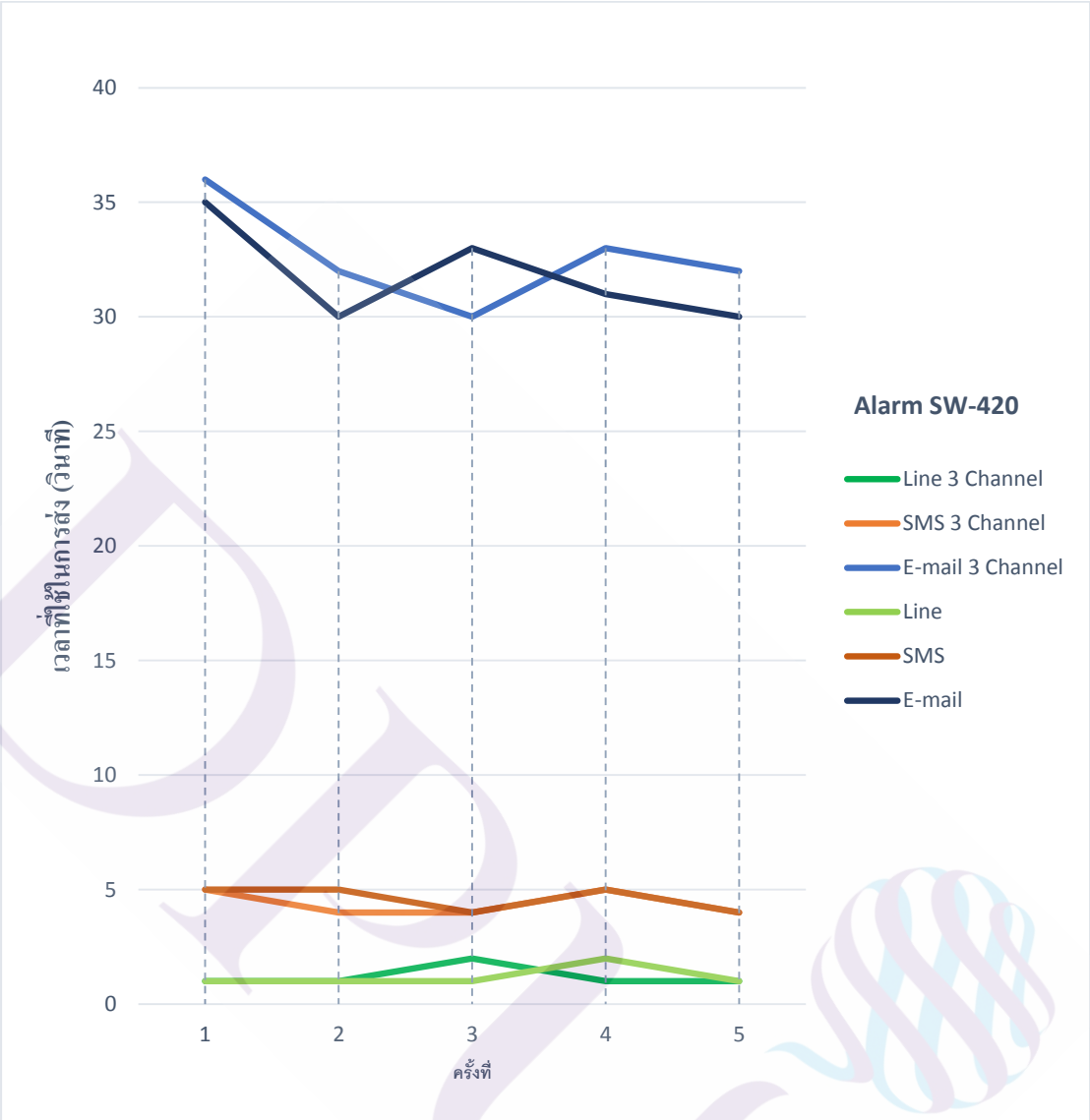


ภาพที่ 4.12 จำลองขณะทำการทดลองเขย่า Sensor SW-420

ตัวอย่างวันที่ 20 มกราคม 2561

ตารางที่ 4.4 เวลาที่จำลองส่งข้อมูล Door Alarm (Vibration Sensor) ที่เกิดขึ้นในช่วงเวลาตั้งแต่ที่เกิด Alarm ไปยังปลายทางทั้ง 3 ช่องทางและจำแนกทีละช่องทาง

เวลาที่เกิด Alarm	เวลาที่ได้รับ Alarm			หมายเหตุ
	Line	SMS	E-mail	
13:40:02	13:40:03	13:40:07	13:40:38	ส่งผ่าน Line , SMS , E-mail
13:42:02	13:42:03	13:42:06	13:42:34	ส่งผ่าน Line , SMS , E-mail
13:44:02	13:44:04	13:44:06	13:44:32	ส่งผ่าน Line , SMS , E-mail
13:46:02	13:46:03	13:46:07	13:46:35	ส่งผ่าน Line , SMS , E-mail
13:48:02	13:48:03	13:48:06	13:48:34	ส่งผ่าน Line , SMS , E-mail
13:55:02	13:55:03	-	-	ส่งผ่าน Line
13:57:03	13:57:04	-	-	ส่งผ่าน Line
13:59:02	13:59:03	-	-	ส่งผ่าน Line
14:01:02	14:01:04	-	-	ส่งผ่าน Line
14:03:03	14:03:04	-	-	ส่งผ่าน Line
14:10:02	-	14:10:07	-	ส่งผ่าน SMS
14:12:02	-	14:12:07	-	ส่งผ่าน SMS
14:14:02	-	14:14:06	-	ส่งผ่าน SMS
14:16:02	-	14:16:07	-	ส่งผ่าน SMS
14:18:03	-	14:18:07	-	ส่งผ่าน SMS
14:25:02	-	-	14:25:37	ส่งผ่าน E-Mail
14:27:02	-	-	14:27:32	ส่งผ่าน E-Mail
14:29:02	-	-	14:29:35	ส่งผ่าน E-Mail
14:31:03	-	-	14:31:34	ส่งผ่าน E-Mail
14:33:02	-	-	14:33:32	ส่งผ่าน E-Mail



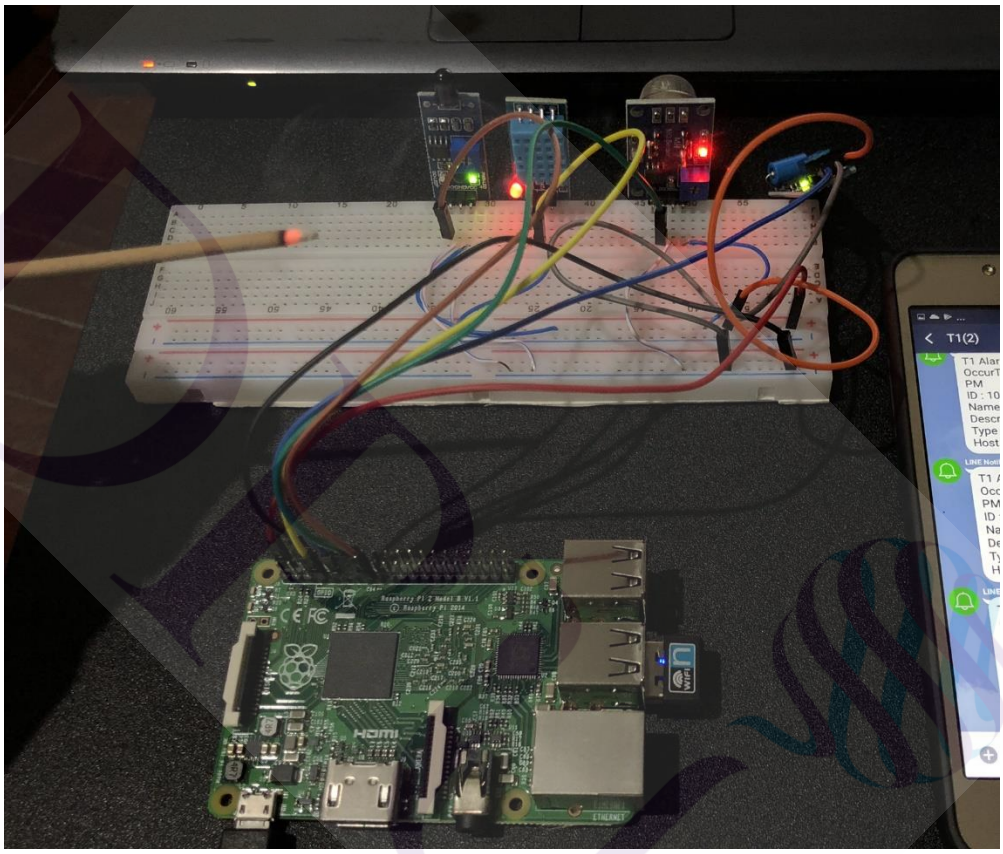
ภาพที่ 4.13 เวลาในการส่งข้อมูลจาก Sensor SW-420 ตั้งแต่เกิด Alarm จนถึงผู้รับปลายทาง

สรุปผลการทดลองที่ 3

เมื่อทำการทดลองเขย่า Sensor SW-420 จะเห็นว่าอุปกรณ์ Raspberry pi จะส่งข้อมูล Alarm Data Temperature Door Alarm ออกมายัง ระบบ Analyzer จนถึงปลายทางจะพบว่าการส่งข้อมูลผ่านทางช่องทาง Line Group (Line Notify) จะสามารถส่งข้อมูลได้ถึงปลายทางเร็วที่สุด และต่อมาคือ SMS และ E-mail ตามลำดับ

การทดลองที่ 4

ทดลองส่ง Alarm จาก Sensor MQ-7 ของอุปกรณ์ Raspberry Pi ไปยังปลายทางทั้ง 3 ช่องพร้อมกันและจำแนกทีละช่องทาง โดยการทดลองจะทำการติดตั้ง Sensor MQ-7 เข้ากับ Raspberry Pi และทำการจำลองให้ระบบตรวจจับควันหรือก๊าซ CO_2 (ซึ่งจากการทดลองจะใช้ควันจากธูป) เพื่อทดลองเหมือนว่ามีควันแล้วสังเกต Alarm ที่ได้รับ



ภาพที่ 4.14 จำลองขณะทำการทดลองนำธูปมาเพื่อทำให้เกิดควันกับ Sensor MQ-7

ตัวอย่างวันที่ 20 มกราคม 2561

ตารางที่ 4.5 เวลาที่จำลองส่งข้อมูล Smoke Alarm (CO_2 Sensor) ที่เกิดขึ้น ในช่วงเวลาตั้งแต่ที่เกิด Alarm ไปยังปลายทางทั้ง 3 ช่องทางและจำแนกทีละช่องทาง

เวลาที่เกิด Alarm	เวลาที่ได้รับ Alarm			หมายเหตุ
	Line	SMS	E-mail	
15:00:38	15:00:39	15:00:42	15:00:58	ส่งผ่าน Line , SMS , E-mail
15:05:21	15:05:22	15:05:25	15:05:51	ส่งผ่าน Line , SMS , E-mail
15:10:30	15:10:31	15:10:34	15:10:57	ส่งผ่าน Line , SMS , E-mail
15:15:19	15:15:20	15:15:23	15:15:43	ส่งผ่าน Line , SMS , E-mail
15:20:26	15:20:27	15:20:30	15:20:49	ส่งผ่าน Line , SMS , E-mail
15:30:24	15:30:25	-	-	ส่งผ่าน Line
15:35:26	15:35:28	-	-	ส่งผ่าน Line
15:40:18	15:40:20	-	-	ส่งผ่าน Line
15:45:21	15:45:22	-	-	ส่งผ่าน Line
15:50:21	15:50:22	-	-	ส่งผ่าน Line
16:00:18	-	16:00:22	-	ส่งผ่าน SMS
16:05:19	-	16:05:22	-	ส่งผ่าน SMS
16:10:21	-	16:10:26	-	ส่งผ่าน SMS
16:15:18	-	16:15:25	-	ส่งผ่าน SMS
16:20:19	-	16:20:24	-	ส่งผ่าน SMS
16:30:20	-	-	16:30:48	ส่งผ่าน E-Mail
16:35:19	-	-	16:35:43	ส่งผ่าน E-Mail
16:40:20	-	-	16:40:46	ส่งผ่าน E-Mail
16:45:19	-	-	16:45:41	ส่งผ่าน E-Mail
16:50:20	-	-	16:50:42	ส่งผ่าน E-Mail

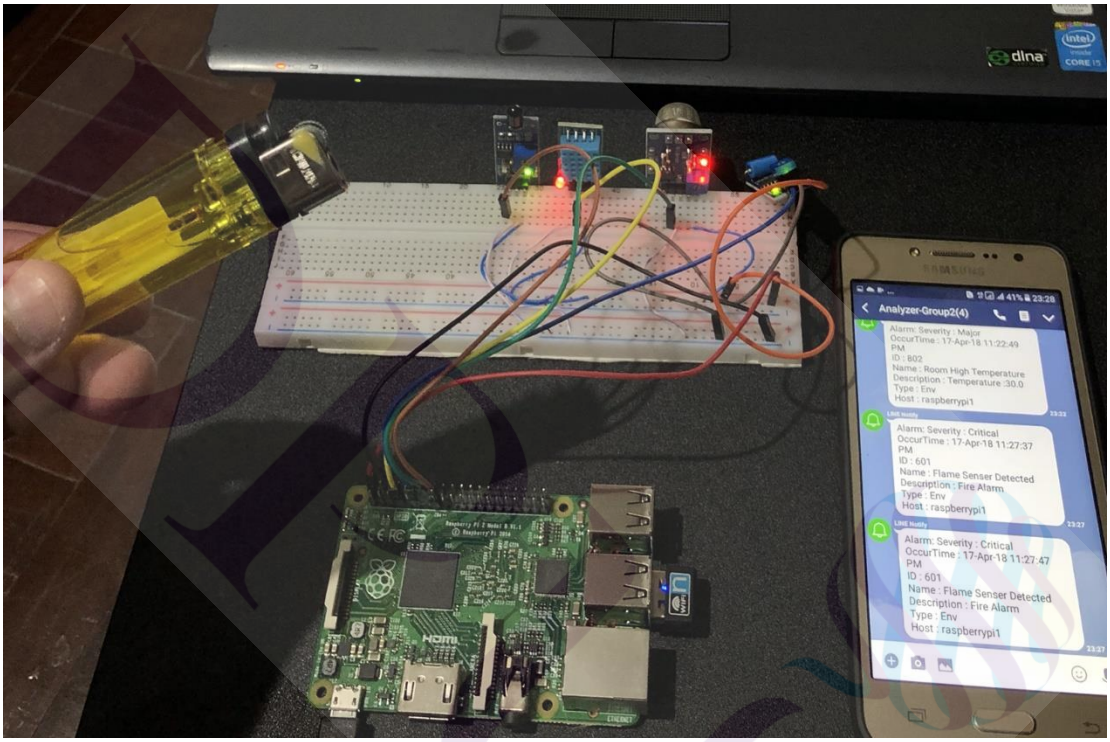
ภาพที่ 4.15 เวลาในการส่งข้อมูลจาก Sensor MQ-7 ตั้งแต่เกิด Alarm จนถึงผู้รับปลายทาง

สรุปผลการทดลองที่ 4

เมื่อทำการทดลองนำธูปไปวางไว้ใกล้ Sensor MQ-7 จะเห็นว่าอุปกรณ์ Raspberry pi จะส่งข้อมูล Alarm Data Smoke Alarm ออกมายัง ระบบ Analyzer จนถึงปลายทางจะพบว่าการส่งข้อมูลผ่านทางช่องทาง Line Group (Line Notify) จะสามารถส่งข้อมูลได้ถึงปลายทางเร็วที่สุด และต่อมาคือ SMS และ E-mail ตามลำดับ

การทดลองที่ 5

ทดลองส่ง Alarm จาก Sensor STM-32 ของอุปกรณ์ Raspberry Pi ไปยังปลายทางทั้ง 3 ช่องพร้อมกันและจำแนกทีละช่องทาง โดยการทดลองจะทำการติดตั้ง Sensor STM-32 กับ Raspberry Pi และทำการจำลองให้ระบบตรวจจับเมื่อมีเปลวไฟอยู่ในรัศมีใกล้เคียง (ซึ่งจากการทดลองจะไฟแช็ค) เพื่อทดลองเหมือนว่ามีเหตุการณ์วางเพลิงหรือไฟไหม้แล้วสังเกต Alarm ที่ได้รับ

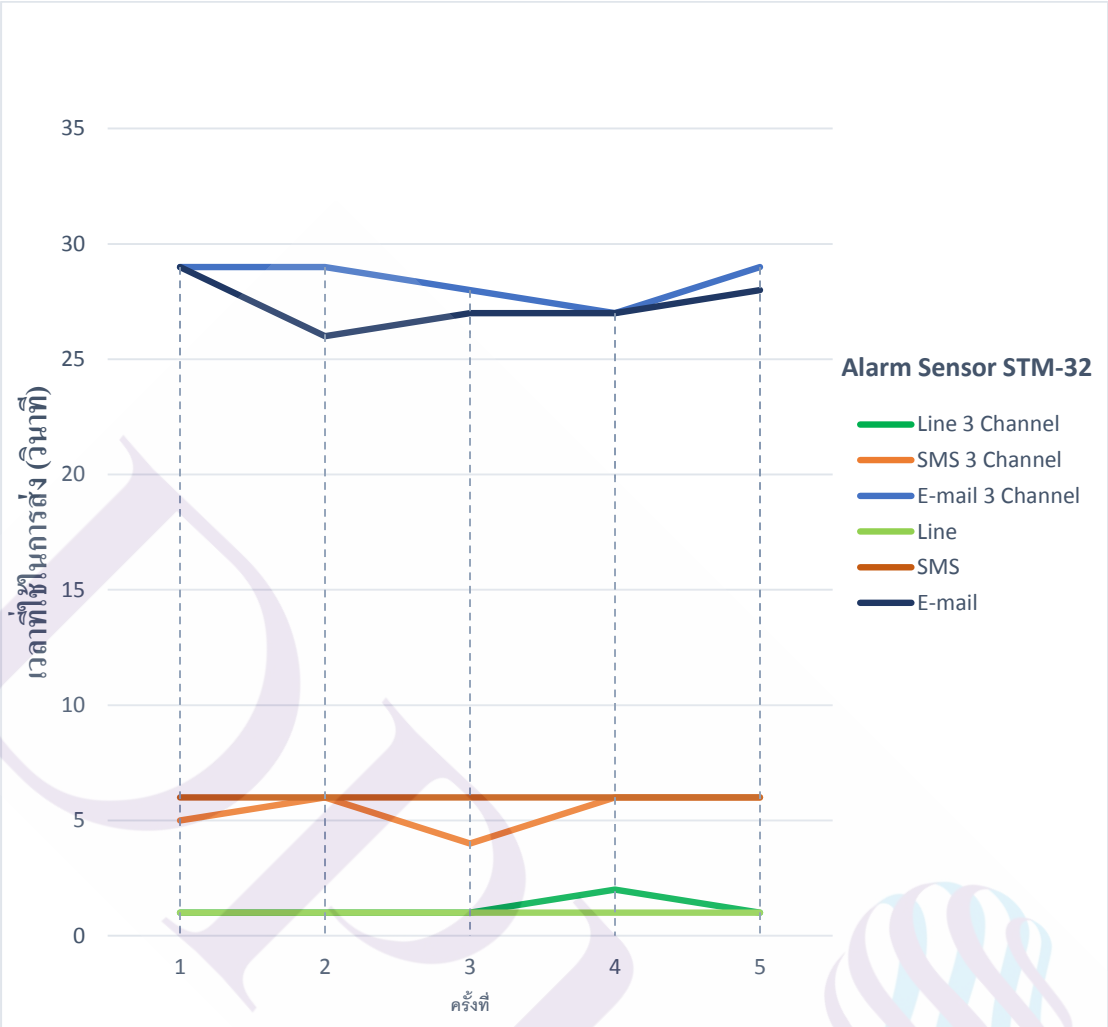


ภาพที่ 4.16 จำลองขณะทำการทดลองนำเปลวไฟมาอยู่ในรัศมี Sensor STM-32

ตัวอย่างวันที่ 20 มกราคม 2561

ตารางที่ 4.6 เวลาที่จำลองส่งข้อมูล Fire Alarm (STM-32 Sensor) ที่เกิดขึ้น ในช่วงเวลาตั้งแต่ที่เกิด Alarm ไปยังปลายทางทั้ง 3 ช่องทางและจำแนกทีละช่องทาง

เวลาที่เกิด Alarm	เวลาที่ได้รับ Alarm			หมายเหตุ
	Line	SMS	E-mail	
17:20:03	17:20:04	17:20:08	17:20:32	ส่งผ่าน Line , SMS , E-mail
17:22:02	17:22:03	17:22:08	17:22:31	ส่งผ่าน Line , SMS , E-mail
17:24:03	17:24:04	17:24:07	17:24:31	ส่งผ่าน Line , SMS , E-mail
17:26:02	17:26:04	17:26:08	17:26:29	ส่งผ่าน Line , SMS , E-mail
17:28:02	17:28:03	17:28:08	17:28:31	ส่งผ่าน Line , SMS , E-mail
17:40:02	17:40:03	-	-	ส่งผ่าน Line
17:42:02	17:42:03	-	-	ส่งผ่าน Line
17:44:03	17:44:04	-	-	ส่งผ่าน Line
17:46:02	17:46:03	-	-	ส่งผ่าน Line
17:48:02	17:48:03	-	-	ส่งผ่าน Line
18:10:02	-	18:10:07	-	ส่งผ่าน SMS
18:12:02	-	18:12:07	-	ส่งผ่าน SMS
18:14:02	-	18:14:07	-	ส่งผ่าน SMS
18:16:02	-	18:16:07	-	ส่งผ่าน SMS
18:18:03	-	18:18:08	-	ส่งผ่าน SMS
18:30:02	-	-	18:30:31	ส่งผ่าน E-Mail
18:32:02	-	-	18:32:28	ส่งผ่าน E-Mail
18:34:03	-	-	18:34:28	ส่งผ่าน E-Mail
18:36:02	-	-	18:36:29	ส่งผ่าน E-Mail
18:38:02	-	-	18:38:30	ส่งผ่าน E-Mail



ภาพที่ 4.17 เวลาในการส่งข้อมูลจาก Sensor STM-32 ตั้งแต่เกิด Alarm จนถึงผู้รับปลายทาง

สรุปผลการทดลองที่ 5

เมื่อทำการทดลองนำเปลวไฟไปใกล้ Sensor STM-32 จะเห็นว่าอุปกรณ์ Raspberry pi จะส่งข้อมูล Alarm Data Fire Alarm ออกมายัง ระบบ Analyzer จนถึงปลายทางจะพบว่าการส่งข้อมูลผ่านทางช่องทาง Line Group (Line Notify) จะสามารถส่งข้อมูลได้ถึงปลายทางเร็วที่สุด และต่อมาคือ SMS และ E-mail ตามลำดับ แล้วจากตารางที่ 4.6 จะเห็นว่า Sensor STM-32 มีความเร็วต่อรัศมีของเปลวไฟค่อนข้างมากทำให้ค่า Event ของการเกิด Alarm แทบจะเกิดทันทีที่เปลวไฟถูกจุดติดขึ้นมา

4.2 สรุปผลการดำเนินการ

จะเห็นได้ว่าทั้ง 5 การทดลองข้างต้นช่องทางในการส่งข้อมูล Alarm ทั้งหมดช่อง Line เป็นช่องทางที่ถึงปลายทางเร็วที่สุดและตามมาด้วย SMS และ E-mail แต่ถ้าตรวจสอบดูค่านัยสำคัญแล้วต่างกันอยู่แค่ประมาณ 4-6 วินาที และ 25-40 วินาทีตามลำดับ ซึ่งไม่น่าจะมีผลกับการที่เมื่อปลายทางได้รับ Alarm แล้วลงมือดำเนินการตามหน้าที่ที่รับผิดชอบ

แต่ยังมีอีก 1 ประเด็นที่น่าสนใจคือ ความรวดเร็วในการตรวจจับของ Sensor ต่าง ซึ่งจากการทดลองพบว่า Sensor STM-32 (Flame Sensor) เป็น Sensor ที่มีความรวดเร็วในการตรวจจับสูงมากเพราะเมื่อมีเปลวไฟเกิดขึ้นตัว Sensor STM-32 จะมี Output ส่งไปยัง Raspberry Pi ทันทีในเวลาไม่ถึง 1 – 2 วินาที ต่างกับ Sensor อื่นซึ่งจะมี Delay ระดับหนึ่งกว่าจะส่ง Output ออกไปยัง Raspberry Pi โดยเรียงลำดับตามนี้ STM-32 > SW-420 > DHT11 > MQ-7

จากนั้นได้ทดลองนำไปใช้จริงโดยให้ Active ครบทุก Sensor และตั้งให้บาง Alarm ส่งเฉพาะ Line, E-mail กับ บาง Alarm ที่สำคัญส่ง Line , SMS (เพราะการใช้ทุก SMS มีค่าใช้จ่าย 1 บาท/ข้อความ) จะได้ผลลัพธ์ตามตารางที่ 4.7

ตัวอย่างวันที่ 1 กุมภาพันธ์ 2561

ตารางที่ 4.7 เวลาที่ส่งข้อมูลตั้งแต่ที่เกิด Alarm ไปยังปลายทาง

เวลาที่เกิด Alarm	เวลาที่ได้รับ Alarm			หมายเหตุ
	Line	SMS	E-mail	
20:00:18	20:00:19	-	20:00:41	ส่งผ่าน Line , E-mail
20:05:02	20:05:03	-	20:05:29	ส่งผ่าน Line , E-mail
20:10:15	20:10:17	-	20:10:33	ส่งผ่าน Line , E-mail
20:15:03	20:15:04	-	20:15:30	ส่งผ่าน Line , E-mail
20:20:21	20:20:22	20:20:25	-	ส่งผ่าน Line , SMS
20:25:02	20:25:03	20:25:07	-	ส่งผ่าน Line , SMS
20:30:02	20:30:03	20:30:08	-	ส่งผ่าน Line , SMS
20:35:19	20:35:20	20:35:22	-	ส่งผ่าน Line , SMS

จากการทดลองทั้งหมดจะเห็นว่า ระบบสามารถส่งข้อมูลไปยังแต่ละ API ได้โดยไม่มีข้อผิดพลาดตามตารางที่ 4.1 ถึงตารางที่ 4.7 และจะเห็นว่าใช้เวลาในการ Process ตั้งแต่รับ Alarm จนถึงส่งไปยังผู้ที่รับผิดชอบใช้เวลาไม่ถึง 1 นาที (ค่าประมาณจากทุกๆ API รวมกัน)

บทที่ 5
สรุปผลการศึกษาและข้อเสนอแนะ

การจัดทำโครงการ Alarm Monitoring with IOT ทางผู้จัดทำได้นำปัญหาที่เกิดขึ้นทั้งหมดมารวบรวมเพื่อหาสาเหตุและวิธีเพิ่มขีดความสามารถในการให้บริการ โดยการนำอุปกรณ์ IOT มาช่วย Monitoring ปัญหาให้สามารถรับรู้ได้ทันถ่วงที ซึ่งทางผู้จัดทำก็หวังว่าโครงการนี้จะช่วยให้สามารถต่อยอดในการบริการได้ โดยที่โครงงานนี้มีประโยชน์ที่ได้รับดังนี้

1. Monitoring อุปกรณ์ IOT
2. Monitoring อุณหภูมิ
3. Monitoring ความชื้น
4. Monitoring Carbon
5. Monitoring Vibration (สั่นสะเทือน)
6. Monitoring Flame

5.1 สรุปผลการวิจัย

การนำอุปกรณ์ Raspberry Pi นำมาใช้เป็นอุปกรณ์ IOT เพื่อต่อกับ Sensor ต่างๆ นั้นสามารถทำงานได้อย่างครบถ้วนตาม วัตถุประสงค์ของการวิจัย โดยดูจากผลสรุปของการดำเนินการที่แบ่งการทดลองออกเป็น 5 การทดลองคือ

1. ทดลองส่ง Alarm CPU Load ของอุปกรณ์ Raspberry Pi ไปยังปลายทางทั้ง 3 ช่องพร้อมกันและจำแนกทีละช่องทาง
2. ทดลองส่ง Alarm จาก Sensor DHT11 ของอุปกรณ์ Raspberry Pi ไปยังปลายทางทั้ง 3 ช่องพร้อมกันและจำแนกทีละช่องทาง
3. ทดลองส่ง Alarm จาก Sensor SW-420 ของอุปกรณ์ Raspberry Pi ไปยังปลายทางทั้ง 3 ช่องพร้อมกันและจำแนกทีละช่องทาง
4. ทดลองส่ง Alarm จาก Sensor MQ-7 ของอุปกรณ์ Raspberry Pi ไปยังปลายทางทั้ง 3 ช่องพร้อมกันและจำแนกทีละช่องทาง

5. ทดลองส่ง Alarm จาก Sensor STM-32 ของอุปกรณ์ Raspberry Pi ไปยังปลายทางทั้ง

3 ช่องพร้อมกันและจำแนกทีละช่องทาง

พบว่าในแต่ละ Sensor อาจจะมีค่า Delay Time อยู่บ้างแต่ก็เรียกได้ว่า น้อยมากจนไม่เป็นค่านัยสำคัญ และการรับค่าของ Sensor ต่างๆ สามารถทำงานได้อย่างมีเสถียรภาพ ส่วน API ที่ใช้เป็นจุดที่สื่อสารกับผู้ใช้งานปลายทาง เช่น Line Notify API , SMS API และ E-mail Server ที่ทางผู้จัดทำเลือกใช้ก็สามารถทำงานได้ครบถ้วน ดูจากผลการทดลองที่ 1 – 5 ในบทที่ 4 จะเห็นได้ว่าการส่งผ่าน Line Notify API จะใช้เวลาน้อยที่สุด และไม่มีค่า Loss เลยเช่นเดียวกับการส่งผ่าน SMS API แต่บางช่วงเวลาในการส่ง SMS เช่น ช่วงเย็น - หัวค่ำ คาดว่าน่าจะมีการใช้งานเครือข่ายที่สูงทำให้ได้รับ SMS ช้าในช่วงเวลาดังกล่าว ส่วนการส่งผ่าน E-mail Server จะพบว่าการส่งทั้งหมดประมาณ 55 ครั้งจะมีอยู่ 1 ครั้งที่ไม่ได้รับ E-Mail คิดเป็น 2 เปอร์เซ็นที่ Loss ไป คาดว่าน่าจะเกิดจากมีการส่ง E-mail ในจำนวนครั้งที่ถี่ไป ทำให้ระบบกรองมองว่าเป็น Spam Mail

5.2 อภิปรายผล

จากการทำโครงการเรื่อง Alarm Monitoring with IOT ผลการศึกษาและการทดลองอยู่ในเกณฑ์ที่ดี และมีความสำเร็จและยืดหยุ่นพอที่จะนำไปต่อยอดเพื่อใช้แก้ปัญหาตามผู้จัดทำต้องการได้ และเมื่อพิจารณารายละเอียดหลายๆ ด้านแล้วคิดว่า ในอนาคตนั้นอุปกรณ์ IOT จะเข้ามามีบทบาทกับชีวิตประจำวันมากขึ้นและสามารถช่วยอำนวยความสะดวกต่างๆ และสามารถทำงานแทนบุคคลได้เลยทีเดียว และโครงการนี้ได้รับคำแนะนำจากอาจารย์และผู้มีประสบการณ์หลายๆ ท่านว่าจะเป็นโครงการที่เป็นประโยชน์ สามารถนำไปประยุกต์ใช้ได้ทันที พร้อมทั้งเป็นอุปกรณ์ IOT ที่มีคนนิยมใช้มากจึงง่ายต่อการพัฒนาเป็นอย่างมาก

5.3 ข้อเสนอแนะ

1. ปัจจุบันผู้จัดทำได้ใช้อุปกรณ์ IOT (Raspberry Pi) ที่ยังต้องอาศัยการเชื่อมต่อกับ Wi-Fi หรือ RJ45 ทำให้ยังมีข้อจำกัดในการใช้งานในจุดที่ไม่มี Wi-Fi หรือ RJ45 ซึ่งสามารถพัฒนาต่อยอดไปใช้อุปกรณ์ 3G , 4G หรือ สัญญาณ LoRaWAN หรือ NB-IOT ในอนาคตได้

2. ในโครงการนี้ยังไม่ได้ดำเนินการออกแบบในส่วนของ Prototype ซึ่งสามารถพัฒนาใส่อุปกรณ์ Battery + 3G เพื่อสามารถทำงานได้ในเวลาที่ไม่มี Power Source จ่ายเข้ามาในระบบ

บรรณานุกรม

บรรณานุกรม

ภาษาไทย

ชัชวาน สุขสุทธิ Chutchavan Suksutthi. **การพัฒนาโปรแกรมบน Raspberry Pi ด้วย Qt**. สืบค้นจาก http://www2.crma.ac.th/itd/Know/RBPI/index.asp

ดุสิต วัฒนเสย, (2555). **ระบบแจ้งเตือนและควบคุมอุปกรณ์ไฟฟ้าด้วย IVR ผ่านระบบ VOIP.** กรุงเทพฯ: ผู้แต่ง.

นิธิกร เตโจ; ธีรยุทธ แก้ววงค์หิว; สุภัตรา ปินจันทร์, (นเรศวรวิจัย ครั้งที่ 12). **วิจัยและนวัตกรรมกับการพัฒนาประเทศ.** สืบค้นจาก http://conference.nu.ac.th/nrc12/

ภูจนา ปาลิยะวรรณ; ผู้ช่วยศาสตราจารย์ ดร. ชาคริดา นุกูลกิจ; ผู้ช่วยศาสตราจารย์ ดร. พรชัย มงคลนาม, (2557). **ระบบติดตามโรคคนทำงานออฟฟิศโดยใช้กล้อง Kinect.** สืบค้นจาก https://dlab.sit.kmutt.ac.th/index.php/kinect-health-monitoring-technology/

วิชัย สุขคลีวนัต; อภิชาต แจ้งบำรุง, (ฉบับที่ 82 ปีที่ 25 ตุลาคม - ธันวาคม 2555). **การศึกษาและวิเคราะห์ระบบป้องกัน อัคคีภัยในอาคารขนาดใหญ่พิเศษ** สืบค้นจาก www.eng.ku.ac.th/e-journal_th

ศิริชัย เติมโชคเกษม; จันทิมา บัวผัน, (2553). **ระบบรักษาความปลอดภัยในบ้าน Executive Journal.** สืบค้นจาก http://www.bu.ac.th/knowledgecenter/executive_journal/july_sep_10/

สถาบัน Home of Maker. **การเรียนรู้ Raspberry Pi .** สืบค้นจาก http://www.homeofmaker.com/

สุชาติ เลื่องยศลือชากุล ฝ่ายวิจัยและพัฒนา บมจ. กสท โทรคมนาคม, (2558). **Web-based Application on Alarm Box Monitoring System.** สืบค้นจาก http://www.rdcat.cathosting.in.th/proj_op_research01.htm

อ.นพ มหิยานนท์, (2560). **Raspberry Pi PROJECTS. (พิมพ์ครั้งที่ 1).** นนทบุรี: ผู้แต่ง.

บรรณานุกรม (ต่อ)

ภาษาต่างประเทศ

Lluís Casals ID , Bernat Mir ID , Rafael Vidal ID and Carles Gomez * ID. (2017). **Modeling the Energy.** Performance of LoRaWAN.

Kaushik Pal ,(July 20, 2016). The Advantages of Real-Time Analytics for Enterprise. Presented by Sponsor: Bloor Group. Retrieved from https://www.techopedia.com/2/31433/trends/big-data/advantages-of-real-time-analytics-for-enterprise

ภาคผนวก

ภาคผนวก

ที่มาของเนื้อหาในบทที่ 2

สืบค้นเมื่อวันที่ 14 เมษายน 2561,จากhttp://shop.allthingstalk.com/product/lora-rapid-development-kit/

สืบค้นเมื่อวันที่ 14 เมษายน 2561,จาก https://www.blognone.com/node/101079

สืบค้นเมื่อวันที่ 14 เมษายน 2561,จาก http://market.samm.com/en/raspberrypi-3

สืบค้นเมื่อวันที่ 14 เมษายน 2561,จาก http://www2.crma.ac.th/itd/Know/RBPI/index.asp

สืบค้นเมื่อวันที่ 14 เมษายน 2561,จาก http://www2.crma.ac.th/itd/Know/RBPI/index.asp

สืบค้นเมื่อวันที่ 14 เมษายน 2561,จาก http://www.homeofmaker.com/?p=891

สืบค้นเมื่อวันที่ 14 เมษายน 2561,จาก http://thaiopensource.org/มาเล่น-gpio-บน-raspbery-pi-กัน/

สืบค้นเมื่อวันที่ 14 เมษายน 2561,จาก https://playelek.com/rpi-os-image/

สืบค้นเมื่อวันที่ 14 เมษายน 2561,จาก http://www.circuitbasics.com/how-to-set-up-the-dht11-humidity-sensor-on-the-raspberry-pi/

สืบค้นเมื่อวันที่ 14 เมษายน 2561,จาก http://issamben.com/how-to-setup-dht11-humidity-temparture-sensor-on-raspberry-pi/

สืบค้นเมื่อวันที่ 14 เมษายน 2561,จาก https://www.arduitronics.com/article/13/%E0%B8%81%E0%B8%B2%E0%B8%A3%E0%B9%83%E0%B8%8A%E0%B9%89%E0%B8%87%E0%B8%B2%E0%B8%99-dht11-humitdity-and-temperature-sensor-%E0%B8%81%E0%B8%B1%E0%B8%9A%E0%B8%9A%E0%B8%AD%E0%B8%A3%E0%B9%8C%E0%B8%94-arduino

สืบค้นเมื่อวันที่ 14 เมษายน 2561,จาก https://www.arduitronics.com/article/13/%E0%B8%81%E0%B8%B2%E0%B8%A3%E0%B9%83%E0%B8%8A%E0%B9%89%E0%B8%87%E0%B8%B2%E0%B8%99-dht11-humitdity-and-temperature-sensor-%E0%B8%81%E0%B8%B1%E0%B8%9A%E0%B8%9A%E0%B8%AD%E0%B8%A3%E0%B9%8C%E0%B8%94-arduino

สืบค้นเมื่อวันที่ 14 เมษายน 2561,จาก https://www.autobotic.com.my/LM393-Tilt-Sensor-Module

สืบค้นเมื่อวันที่ 14 เมษายน 2561,จาก https://www.piddlerintheroot.com/vibration-sensor/

สืบค้นเมื่อวันที่ 14 เมษายน 2561,จากhttps://tutorials-raspberrypi.com/configure-and-read-out-the-raspberry-pi-gas-sensor-mq-x/

สืบค้นเมื่อวันที่ 14 เมษายน 2561,จาก http://www.learningaboutelectronics.com/Articles/MQ-2-smoke-sensor-circuit-with-raspberry-pi.php

สืบค้นเมื่อวันที่ 14 เมษายน 2561,จาก https://www.sparkfun.com/datasheets/Sensors/Biometric/MQ-7.pdf

สืบค้นเมื่อวันที่ 14 เมษายน 2561,จาก https://www.aliexpress.com/item/Waveshare-Fire-Flame-Sensor-Module-IR-Infrared-Flame-Detection-Sensor-Module-Flame-for-STM32-Raspberry-pi/32644539123.html

สืบค้นเมื่อวันที่ 14 เมษายน 2561,จาก http://www.instructables.com/id/Flame-Sensor-Raspberry-Pi/

สืบค้นเมื่อวันที่ 14 เมษายน 2561,จาก http://www.theorycircuit.com/arduino-flame-sensor-interface/

สืบค้นเมื่อวันที่ 14 เมษายน 2561,จาก http://www.voip-sip-sdk.com/p_312-voip-network-communication-protocols-voip.html

สืบค้นเมื่อวันที่ 14 เมษายน 2561,จาก https://en.wikipedia.org/wiki/User_Datagram_Protocol

ประวัติผู้เขียน

ชื่อ-นามสกุล	นายณพวุฒิ โพธิ์หอม
ประวัติการศึกษา	วิศวกรรมศาสตร์บัณฑิต (วิศกรรมคอมพิวเตอร์) มหาวิทยาลัยรังสิต ปี 2551
ตำแหน่งและสถานที่ทำงานปัจจุบัน	วิศวกรรถไฟฟ้า ระบบอาณัติสัญญาณ บริษัท เอ็มเอชพีเอ็ม จำกัด 128/128 อาคารพญาไทพลาซ่า ชั้น 12 ห้อง ดี ถนนพญาไท แขวงทุ่งพญาไท เขตราชเทวี กรุงเทพมหานคร 10400