ชื่อ : นายศิลวัต ลีลาศ

ชื่อโครงงาน : ระบบเก็บข้อมูลเบื้องต้นสำหรับ Digital Twin

สาขาวิชา : เทคโนโลยีสารสนเทศมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

อาจารย์ที่ปรึกษา : ผู้ช่วยศาสตราจารย์ ดร.ขนิษฐา นามี

ปีการศึกษา : 2560

**บทคัดย่อ**

ในปัจจุบันและอนาคตอุปกรณ์ IoT ที่สำคัญจะมีการทำสำเนาข้อมูลของลักษณะทางกายภาพและสถานะต่างๆ หรือโปรไฟล์ขึ้นไปเก็บไว้ในโลกดิจิตอลหรือโลกไซเบอร์ข้อมูลทั้งหลายเหล่านี้จะถูกนำมาใช้งานเพื่อช่วยในการตัดสินใจบรหิหารจัดการดูแลหรือวิเคราะห์ค่าต่างๆ ที่เกี่ยวกับอุปกรณ์ ตัวอย่างเช่นสถานะต่างๆ ราวมทั้งสถาณภาพการทำงานของแต่ละชิ้นส่วนของรถยนต์หรือเครื่องจักรในโรงงานอุตสาหกรรมจะถูกนำมาบันทึกไว้เพื่อประโยชน์ในการวิเคราะห์หาค่าที่เหมาะสมในการใช้งานหรือการแจ้งเตือนก่อนเกิดการชำรุด เสียหาย สำเนาทั้งหลายที่เก็บจึงถูกเรียกว่า Digital Twin หรือคู่แฝดในโลกดิจิตัล

(โครงงานสหกิจศึกษามีจำนวนทั้งสิิ้น XXX หน้า)

คำสำคัญ : Digital Twin, Data format, Metada, IoT, Database, Dashboard IoT

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ อาจารย์ที่ปรึกษาวิทยานิพันธ์หลัก

**กิตติกรรมประกาศ**

โครงงานสหกิจศึกษาเรื่อง ระบบเก็บข้อมูลเบื้องต้นสำหรับ Digital Twin สำเร็จลุล่วงไปได้ด้วยดีเนื่องจากได้รับคำแนะนำจาก ดร.กุลชาติ มีทรัพย์หลาก และขอบคุณอาจารย์ที่ปรึกษาโครงงานสหกิจศึกษา ผศ.ดร.ขนิษฐา นามี ที่คอยติดตามและให้คำแนะนำในการทำโครงงานสหกิจศึกษาในครั้งนี้ขอขอบคุณศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติที่ได้ให้คำแนะนำและให้ความรู้ถึงข้อมูลที่ใช้ในการทำโครงงานสหกิจศึกษาในครั้งนี้ สุดท้ายขอระลึกถึงพระคุณของบิดามารดาครูบาอาจารย์ของข้าพเจ้าที่ได้อบรมสั่งสอนจนถือได้ว่าประสบความสำเร็จไปอีกก้าวหนึ่งของชีวิตขอขอบคุณคำแนะนำและน้ำใจของเหล่าเพื่อนทุกท่านที่ช่วยเหลือและเป็นส่วนหนึ่งที่ทำให้เกิดความสำเร็จของการจัดทำโครงงานสหกิจศึกษาในครั้งนี้

ศิลวัต ลีลาศ

**สารบัญ**

**บทที่ 1**

**บทนำ**

**1.1 ข้อมูลพื้นฐานการปฏบัติงานสหกิจศึกษา**

1.1.1 ชื่อสถานประกอบการ ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ  
ที่อยู่เลขที่ 112 ถนนพหลโยธิน ตำบล/แขวงคลองหนึ่ง อำเภอ/เขตคลองหลวง จังหวัดปทุมธานี 12120 โทรศัพท์ 02-5646900 ต่อ 2566 ชั้น 4 ห้อง 418B

1.1.2 ลักษณะการประกอบการ

1.1.3 ลักษณะงานที่นักศึกษาได้รับหมอบหมายให้รับผิดชอบ

ได้รับงานต่างๆที่ ดร. ต้องการให้ทำและมีประโยชน์ต่อหน่วยงานและได้ใช้จริงและ ทำโครงงานที่ได้รับมอบหมาย

1.1.4 พนักงานที่ปรึกษาและตำแหน่งของพนักงานที่ปรึกษา

ดร.กุลชาติ มีทรัพย์หลาก ตำแหน่งนักวิจัย

1.1.5 ระยะเวลาที่ปฏิบัติงาน

วันที่ 4 มิถุนายน 2561 ถึง 30 พฤษจิกายน 2561

**1.2 ความเป็นมาและปัญหา**

ในปัจจุบันและอนาคตอุปกรณ์ IoT ที่สำคัญจะมีการทำสำเนาข้อมูลของลักษณะทางกายภาพและสถานะต่างๆ หรือโปรไฟล์ขึ้นไปเก็บไว้ในโลกดิจิตอลหรือโลกไซเบอร์ข้อมูลทั้งหลายเหล่านี้จะถูกนำมาใช้งานเพื่อช่วยในการตัดสินใจบรหิหารจัดการดูแลหรือวิเคราะห์ค่าต่างๆ ที่เกี่ยวกับอุปกรณ์ ตัวอย่างเช่นสถานะต่างๆ ราวมทั้งสถาณภาพการทำงานของแต่ละชิ้นส่วนของรถยนต์หรือเครื่องจักรในโรงงานอุตสาหกรรมจะถูกนำมาบันทึกไว้เพื่อประโยชน์ในการวิเคราะห์หาค่าที่เหมาะสมในการใช้งานหรือการแจ้งเตือนก่อนเกิดการชำรุด เสียหาย สำเนาทั้งหลายที่เก็บจึงถูกเรียกว่า Digital Twin หรือคู่แฝดในโลกดิจิตัล

ดังนั้นการสร้างระบบที่รองรับกับ Digital Twin ต้องพิจารณาถึงระบบฐานข้อมูลและปัจจัยอื่นๆ เพื่อบูรณาการข้อมูลจาก IoT เข้าสู่ระบบนอกจากนี้การออกแบบควรพิจารณาถึงรูปแบบ (data format) ของข้อมูลซึ่งต้องอาศัยรายละเอียดฐานข้อมูลหรือ metadata ที่จะต้องจัดเก็บเพื่อการทำงานหรือตีความหมายของอุปกรณ์ต่างๆ เพื่อทำงานร่วมกันอีกด้วย และนอกจากนั้นยังต้องนำข้อมูลมาแสดงผลในรูปแบบแดชบอร์ดเพื่อให้ง่ายต่อการแสดงผลและเข้าใจง่ายต่อผู้ใช้

โครงงานจึงเน้นไปที่การศึกษาและสร้างระบบเก็บฐานข้อมูลบางส่วนสำหรับ Digital twin และการทำหน้าต่างแสดงผลเกี่ยวกับข้อมูล IoT ตามความต้องการดังกล่าวผ่านทางคลาวด์แพลตฟอร์มให้บริการ Network Platform for Internet of Everything (NETPIE) ของศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ

ตัวอย่าง Sensor ที่เลือกมาใช้ในโครงงานนี้มีทั้งหมด 4 ตัวเป็น Senosr ตัวอย่างที่ทำงานเกี่ยวกับ Smart Environment หรือที่ทำงานเพื่อวัดสิ่งแวดล้อมที่เหมาะสมต่อการอยู่อาศัย และยังมีหน้าต่างแสดงผลที่ออกแบบมาเพื่อง่ายต่อผู้ใช้งานที่ดึงข้อมูลมากจาก NETPIE และฐานข้อมูลดังกล่าว

**1.3 วัตถุประสงค์ของโครงาน**

1.3.1 เพื่อศึกษาแนวทางการสร้างและพัฒนาระบบพื้นฐานบางส่วนและจัดเก็บข้อมูลสำหรับ Digital Twin

1.3.2 เพื่อศึกษาความเป็นไปได้ในการบูรณาการข้อมูล IoT ระหว่างอุปกรณ์และเซนเซอร์ต่างๆ และการพัฒนาเบื้องต้น

1.3.3 เพื่อมีการเก็บข้อมูล IoT ที่มากมายและมีการส่งข้อมูลตลอดเวลาให้เกิดเป็นระบบมาตรฐานและสามารถนำข้อมูลมาใช้ให้เกิดประโยชน์ได้

1.3.4 เพื่อนำข้อมูลที่ได้เก็บไว้มาแสดงผลมบนหน้าต่างแสดงผล

**1.4 ขอบเขตของโครงงาน**

1.4.1 Data format ของเซนเซอร์ต่างๆ โดยโครงงานจะยกกรณีตัวอย่าง ทั้งหมด 4 ตัว

1.4.1.1 Data format ของ Sensor DHT22

1.4.1.1.1 มีการเก็บข้อมูลหลักๆ คือ ID

1.4.1.1.2 มีการเก็บข้อมูลหลักๆ คือ สถานะ (state)

1.4.1.1.3 มีการเก็บข้อมูลสิ่งแวดล้อมต่างๆ ของSensor DHT22

1.4.1.1.4 ออกแบบภายใต้เงื่อนไขโครงสร้างของ Metadataเพื่อช่วยให้ จัดการข้อมูล IoT ได้ง่ายขึ้น

1.4.1.2 Data format ของ Sensor MQ-7

1.4.1.2.1 มีการเก็บข้อมูลหลักๆ คือ ID

1.4.1.2.2 มีการเก็บข้อมูลหลักๆ คือ สถานะ (State)

1.4.1.2.3 มีการเก็บข้อมูลสิ่งแวดล้อมต่างๆ ของSensor MQ-7

1.4.1.2.4 ออกแบบภายใต้เงื่อนไขโครงสร้างของMetadata เพื่อช่วยให้ จัดการข้อมูล IoT ได้ง่ายขึ้น

1.4.1.3 Data format ของ Sensor Pulse Heart rate

1.4.1.3.1 มีการเก็บข้อมูลหลักๆ คือ ID

1.4.1.3.2 มีการเก็บข้อมูลหลักๆ คือ สถานะ (State)

1.4.1.3.3 มีการเก็บข้อมูลสิ่งแวดล้อมต่างๆ ของSensor Pulse Heart rate

1.4.1.3.4 ออกแบบภายใต้เงื่อนไขโครงสร้างของMetadata เพื่อช่วยให้ จัดการข้อมูล IoT ได้ง่ายขึ้น

1.4.1.4 Data format ของ Sensor GY-521

1.4.1.4.1 มีการเก็บข้อมูลหลักๆ คือ ID

1.4.1.4.2 มีการเก็บข้อมูลหลักๆ คือ สถานะ (State)

1.4.1.4.3 มีการเก็บข้อมูลสิ่งแวดล้อมต่างๆ ของSensor GY521

1.4.1.1.4 ออกแบบภายใต้เงื่อนไขโครงสร้างของMetadata เพื่อช่วยให้ จัดการข้อมูล IoT ได้ง่ายขึ้น

1.4.2 ฐานข้อมูลเพื่อจัดเก็บข้อมูล IoT สำหรับระบบDigital Twin

1.4.2.1 ใช้ฐานข้อมูล MongoDB

1.4.2.2 รองรับรูปแบบของ Metadata

1.4.2.3 สามารถเก็บข้อมูลในรูปแบบ JSON

1.4.2.4 เก็บข้อมูลต่างๆ ที่ส่งมาได้

1.4.3 การรับข้อมูลและการส่งข้อมูลของ IoT สำหรับ Digital Twin

1.4.3.1 สามารถรับข้อมูลจาก sensor IoT จากคลาวด์แพลตฟอร์ม NETPIE

1.4.3.2 สามารถส่งข้อมูลจาก sensor IoT จากคลาวด์แพลตฟอร์ม NETPIE

1.4.3.3 รองรับรูปแบบของ metadata

1.4.3.4 รองรับรูปแบบของข้อมูลเป็น JSON

1.4.4 REST API

1.4.4.1 สามารถดึงข้อมูลจากฐานข้อมูลได้

1.4.4.2 สามารถบันทึกข้อมูลลงฐานข้อมูลได้

1.4.4.3 สามารถแก้ไขข้อมูลในฐานข้อมูลได้

1.4.4.4 สามารถลบข้อมูลในฐานข้อมูลได้

1.4.4.5 รองรับข้อมูลที่มีโครงสร้างแบบ JSON

1.4.4.6 รองรับข้อมูล metadata

1.4.5 Interface หรือหน้าต่างแสดงผล

1.4.5.1 สามารถแสดงผลข้อมูลเกี่ยวกับ sensor

1.4.5.2 สามารถปรับแต่งการแสดงผลข้อมูลได้

1.4.5.3 ดึงข้อมูลที่ต้องการแสดงจาก NETPIE

1.4.5.4 ใช้ REST API ในการดึงข้อมูลจากฐานข้อมูล

1.4.5.5 แสดงผลในรูปแบบของ Dashboard for IoT

**1.5 วิธีการดำเนินงาน**

1.5.1 ออกแบบ Data format ของเซ็นเซอร์ทั้ง 4 ตัว

1.5.1.1 เซ็นเซอร์ DHT22

1.5.1.1 ศึกษาการทำงานของเซ็นเซอรฺ์ DHT22

1.5.1.1.2 ออกแบบ data format ของเซ็นเซอร์ DHT22

1.5.1.1.3 ศึกษาและออกแบบภายใต้โครงสร้างของ metadata

1.5.1.2 เซ็นเซอร์ MQ-7

1.5.1.2.1 ศึกษาการทำงานของเซ็นเซอรฺ์ MQ-7

1.5.1.2.2 ออกแบบ data format ของเซ็นเซอร์ MQ-7

1.5.1.2.3 ศึกษาและออกแบบภายใต้โครงสร้างของ metadata

1.5.1.3 เซ็นเซอร์ Pulse Heart Rate

1.5.1.2.1 ศึกษาการทำงานของเซ็นเซอรฺ์ Pulse Heart Rate

1.5.1.2.2 ออกแบบ data format ของเซ็นเซอร์ Pulse Heart Rate

1.5.1.2.3 ศึกษาและออกแบบภายใต้โครงสร้างของ metadata

1.5.1.4 เซ็นเซอร์ GY-521

1.5.1.2.1 ศึกษาการทำงานของเซ็นเซอรฺ์ GY-521

1.5.1.2.2 ออกแบบ data format ของเซ็นเซอร์ GY-521

1.5.1.2.3 ศึกษาและออกแบบภายใต้โครงสร้างของ metadata

1.5.2 ฐานข้อมูลเพื่ิอจัดเก็บข้อมูล IoT

1.5.2.1 ศึกษาการใช้งานฐานข้อมูล mongoDB

1.5.2.2 สามารถเก็บบันทึกข้อมูลของเซ็นเซอร์ต่างๆได้

1.5.2.3 สามารถแก้ไชข้อมูลของเซ็นเซอร์ต่างๆได้

1.5.2.4 สามารถลบข้อมูลของเซ็นเซอร์ต่างๆได้

1.5.2.5 สามารถทำให้รองรับรูปแบบ JSON

**1.6 ประโยชน์ที่คาดว่าจะได้รับ**

1.6.1 ได้ Data format ของIoT ที่มี metadata ไว้อธิบายข้อมูลต่างๆ ของเซ็นเซอร์

1.6.2 ได้ระบบฐานข้อมูลที่มีประสิทธิภาพในการเก็บข้อมูลของ IoT อยู่ตลอดเวลาและสามารถนำข้อมูลมาใช้งานเบื้องต้นเกี่ยวกับ Digital Twin

1.6.3 ได้ระบบ API ที่ทำงานโดยตรงกับระบบฐานข้อมูล

1.6.4 ได้หน้าต่างแสดงผลข้อมูลของ IoT ที่มีความสวยงามและง่ายต่อผู้ใช้

**บทที่ 2**

**เอกสารและงานวิจัยที่เกี่ยวข้อง**

**2.1 Digital Twin**

เมื่อ Internet of things เข้ามาสู้สื่อกระแสหลักและกลายเป็นคำที่ใช้กันกว้างขวางทั้งในและนอกวงการไอที หลายคนคงจินตนาการได้ถึงอนาคตที่รอบตัวเราจะมีเซ็นเซอร์หลายร้อยหลายพันตัว Gatner ถึงขนาดพยากรณ์ว่าจะมีเซ็นเซอร์มากกว่า 500 ตัวในบ้าน 1 หลังในปี 2022 หลังจากที่ IoT เข้ามาเป็นส่วนหนึ่งของสภาพแวดล้อมแล้วชีวิตเราจะเป็นอย่างไรต่อคำศัพท์ใหม่ที่เริ่มเป็นที่กล่าวถึงในปีที่ผ่านมาคือ Digital Twin ซึ่งหมานถึงการเก็บสำเนาของทรัพย์สินสิ่งของอาคารหรือเครื่องจักรในรูปแบบดิจทัล Digital Twin ไม่ได้หมายถึงแค่การสร้างโมเดลจำลองของสิ่งของในแบบ 3 มิติ แต่รวมการบันทึกข้อมูลสถานะทุกอย่างของของชิ้นนั้นเพื่อตรวจสอบการทำงานเพื่อคาดการณ์ความผิดปกติล่วงหน้าเพื่อหาต้นตอของปัญหาและหลีกเลี่ยงปัญหาในอนาคต ฯลฯ Digital Twin ต้องอาศัยการติดเซนเซอร์จำนวนมากเพื่อได้สำเนาที่ใกล้เคียงความจริงในปัจจุบันที่สุดข้อมูลมหาศาลจากเซนเซอร์จะถูกส่งขึ้นคลาวด์เพื่อจัดเก็บข้อมูลและประมวลผลตามบริบทของการใช้งาน เช่น ถ้าจะสร้าง Digital Twin ของสะพานก็ต้องมีข้อมูลการสั่นสะเทือนแรงกดในทุกจุดบนสะพาน ใต้สะพาน บนเสา เพื่อประเมินสภาพหรือคาดการณ์การร้าวหรือทรุดตัวของสะพานเป็นต้นอาจกล่าวได้ว่า Digital Twin จะเกิดขึ้นได้ต้องอาศัยหลายเทคโนโลยี ที่เป็น buzzword ของยุคนี้ไม่ว่าจะเป็น IoT, Cloud, Big Data, Machine Learning รวมถึง 3D modeling Digital Twin จึงได้รับการคาดหวังสูงและติดโผเป็น 1 ใน 10 เทคโนโลยีที่น่าจับตามองของปี 2017 ความจริงแล้วการสร้างระบบเสมือนหรือสำเนาของระบบหรือเครื่องจักรมีมาตั้งแต่ยุคก่อนอินเทอร์เน็ต ผู้บุกเบิกก็คือองค์การบริหารการบินและอวกาศแห่งชาติสหรัฐฯ หรือ NASA ในภารกิจส่งยานอวกาศออกไปนอกโลก NASA จำเป็นต้องควบคุมหรือซ่อมบำรุงยานให้ได้จากทางไกลเมื่อเกิดเหตุขัดข้องอย่างกรณีของ Apollo 13 ระบบเสมือนที่อยู่บนโลกช่วยให้วิศวกรวิเคราะห์ปัญหาและกู้ระบบคืนได้จากระยะไกล ปัจจุบันเริ่มมีการประยุกต์ใช้ Digital Twin ในหลายด้านแล้ว เช่นในการผลิตและซ่อมบำรุงเครื่องยนต์เจ็ตที่ใช้บนเครื่องบิน บริษัท General Electric ติดตั้งเซนเซอร์กว่า 100 จุดบนเครื่องยนต์เพื่อสร้างระบบจำลองของเครื่องยนต์เจ็ตทุกตัวที่บริษัทขายเพื่อตรวจสอบสถานะระหว่างการบินคงไม่ยากที่จะจินตนาการว่าอีกหน่อยรถยนต์ทุกคันก็จะมี Digital Twin ศูนย์บริการจะรู้ว่ารถคันไหนใช้งานมากน้อยแค่ไหน สถานะของแบตเตอรี่ ลมยาง น้ำมันหล่อลื่น ผ้าเบรคเป็นอย่างไร เจ้าของรถเพียงแค่รอศูนย์โทรมาเรียกให้นำรถเข้าไปซ่อมบำรุงแทนการนัดหมายทุก 6 เดือน ผู้ที่ใช้ประโยชน์อาจไม่ใช่แค่ศูนย์บริการหากรถนั้นเป็นรถสาธารณะผู้ใช้บริการสามารถตรวจสอบสภาพของรถก่อนเลือกใช้บริการได้ ไม่เพียงแค่เครื่องจักร อีกหน่อยคงมีสำเนาดิจิทัลของคน สัตว์และสิ่งของรอบตัวเผลอๆ โรงพยาบาลอาจโทรมาเรียกเราไปหาหมอก่อนที่ตัวเราจะรู้สึกตัวว่าป่วยก็ได้

“สักวาฝาแฝดดิจิทัล

ทุกสิ่งสรรพ์มีสำเนาบนโลกเสมือน

ทุกพฤติกรรมบันทึกไว้ไม่บิดเบือน

หลักฐานเตือนให้ตั้งมั่นหมั่นทำดี

พูดคิดดีมีสติมิให้ขาด

เพียงเผลอพลาดพลั้งผิดมิตรหลีกหนี

โลภโกรธหลงประสงค์ร้ายคุมให้ดี

อย่าให้มีสำเนาสีเทาเอย”

**2.2 Internet of Things**

"**อินเทอร์เน็ตของสรรพสิ่ง**" **เชื่อมต่ออุปกรณ์และพาหนะด้วยเซ็นเซอร์อิเล็กทรอนิกส์และอินเทอร์เน็ตอินเทอร์เน็ตของสรรพสิ่ง** (**อังกฤษ**: Internet of Things) **หรือไอโอที** (IoT) **หมายถึงเครือข่ายของวัตถุอุปกรณ์ พาหนะ สิ่งปลูกสร้าง และสิ่งของอื่นๆ ที่มีวงจรอิเล็กทรอนิกส์ ซอฟต์แวร์ เซ็นเซอร์และการเชื่อมต่อกับเครือข่ายฝังตัวอยู่และทำให้วัตถุเหล่านั้นสามารถเก็บบันทึกและแลกเปลี่ยนข้อมูลได้**

อินเทอร์เน็ตของสรรพสิ่งทำให้วัตถุสามารถรับรู้สภาพแวดล้อมและถูกควบคุมได้จากระยะไกลผ่านโครงสร้างพื้นฐานเครือข่ายที่มีอยู่แล้วทำให้เราสามารถผสานโลกกายภาพเข้ากับระบบคอมพิวเตอร์ได้แนบแน่นมากขึ้นผลที่ตามมาคือประสิทธิภาพความแม่นยำและประโยชน์ทางเศรษฐกิจที่เพิ่มมากขึ้นเมื่อ IoT ถูกเสริมด้วยเซ็นเซอร์และแอคชูเอเตอร์ซึ่งสามารถเปลี่ยนลักษณะทางกลได้ตามการกระตุ้นก็จะกลายเป็นระบบที่ถูกจัดประเภทโดยทั่วไปว่าระบบไซเบอร์-กายภาพ (cyber-physical system) ซึ่งรวมถึงเทคโนโลยีอย่างกริดไฟฟ้าอัจริยะ (สมาร์ตกริด) บ้านอัจฉริยะ (สมาร์ตโฮม) ระบบ ขนส่งอัจฉริยะ(อินเทลลิเจนต์ทรานสปอร์ต) และเมืองอัจฉริยะ (สมาร์ตซิตี้) วัตถุแต่ละชิ้นสามารถถูกระบุได้โดยไม่ซ้ำกันผ่านระบบคอมพิวเตอร์ฝังตัวและสามารถทำงานร่วมกันได้บนโครงสร้างพื้นฐานอินเทอร์เน็ตที่มีอยู่แล้วในปัจจุบัน ผู้เชี่ยวชาญประเมินว่าเครือข่ายของสรรพสิ่งจะมีวัตถุเกือบ 50,000 ล้านชิ้นภายในปี 2020 มูลค่าตลาดคาดการณ์ไว้ที่ 80 พันล้านเหรียญ "สรรพสิ่ง" ในความหมายของ IoT สามารถหมายถึงอุปกรณ์ที่แตกต่างหลากหลาย เช่น อุปกรณ์วัดอัตราหัวใจแบบฝังใน ร่างกายแท็กไบโอชิปที่ติดกับปศุสัตว์ ยานยนต์ที่มีเซ็นเซอร์ในตัว อุปกรณ์วิเคราะห์ดีเอ็นเอในสิ่งแวดล้อมหรือ อาหารหรืออุปกรณ์ภาคสนามที่ช่วยในการทำงานของนักผจญเพลิงในภารกิจค้นหาและช่วยเหลืออุปกรณ์เหล่านี้ จะจัดเก็บข้อมูลที่เป็นประโยชน์ด้วยการใช้เทคโนโลยีหลากหลายชนิดและจากส่งต่อข้อมูลระหว่างอุปกรณ์อื่นๆ โดยอัตโนมัติ ตัวอย่างในตลาดขณะนี้ เช่น เทอร์โมสตัตอัจฉริยะ และ เครื่องซักผ้า-อบผ้าที่ต่อกับเครือข่ายไวไฟเพื่อให้สามารถดูสถานะจากระยะไกลได้

**2.3 MongoDB Datbase**

MongoDB **เป็น** database **แบบ** NoSQL **ตัวหนึ่งซึ่งแน่นอนว่าไม่เหมือนกับ** MySQL **แน่ๆ แต่สิ่งที่** MongoDB **หยิบยื่นให้แตกต่างจาก** NoSQL **ตัวอื่นบางตัวก็คือการที่มีบริษัทเอกชนเป็น** back **หลังให้อยู่ ซึ่งคือบริษัท** 10gen Inc (**หลายตัวก็มีบริษัทสนับสนุนด้วยเหมือนกัน**) **เนื่องมาจาก** product **อะไรก็ตามที่เป็น** Open Source **แบบคนธรรมดาทำจะมีความเสี่ยงอย่างหนึ่งก็คือใช้ๆไปแล้วเกิดคนที่พัฒนาเบื่อหรือเปลี่ยนความสนใจหรือมีเหตุที่จะไม่สามารถทำต่อได้อีก อนาคตเราก็ดับวูบไปเลยแต่ว่าหากมีบริษัทเป็น** Backup **อย่างน้อยก็มั่นใจได้ว่ามันจะยังไปต่อได้ ตราบใดที่มีบริษัทสนับสนุนและเมื่อถึงวันที่มันแข็งแกร่งพอก็จะเกิด** version community **ตามออกมาเองคราวนี้ก็จะเหมือนดาวค้างฟ้า เป็น** Open Source **คงกระพันเหมือนอย่างที่** MySQL **เคยเป็นมาก่อนนั่นแหล่ะอีกเรื่องที่เป็นความแตกต่างก็คือ** MongoDB **ไม่ได้เรียบง่ายมากถึงขนาดกับเป็น** database **ที่เก็บได้แค่** key **กับ** value **ที่ทำเงื่อนไขอะไรอย่างอื่นไม่ได้แต่ว่ามันสามารถทำงานที่มีความซับซ้อนและเงื่อนไขเพิ่มขึ้นได้มากกว่า**database**ตัวอื่นโดยตัว** MongoDB **เองชูจุดเด่นเอาไว้ดังนี้**

* เก็บข้อมูลแบบ Document - คือการเก็บข้อมูลในรูปแบบที่เป็น

Pattern แบบมีโครงสร้าง (จะไม่แบนๆ แบบ MySQL ที่ใน table จะมี field หลายๆ field) โดยมีโครงสร้างแบบทั้งลึกและกว้าง ในแต่ละ record หากนึกไม่ออกลองนึกถึง array แบบหลายมิติครับ นั่นล่ะครับ 1 record สามารถเก็บเป็นแบบ array หลายมิติได้ ไม่แบนราบเหมือน MySQL 1 record ที่เมื่อแปลงเป็น Array ก็ได้แค่มิติเดียวเท่านั้น

* รองรับการทำ Full Index - มีข้อดีในการ search หาได้อย่างรวดเร็วกับข้อมูลที่มีปริมาณมหาศาล (เรียกว่ามหาศาลเพราะเยอะกว่าปกติ) และ search ได้จากข้อมูลในส่วนใดก็ได้
* รองรับการขยายขนาดและรองรับการทำงานหนักๆ - เพราะว่าเน้นรองรับงานหนักและ ปริมาณข้อมูลมากๆ สามารถขยายขนาดได้อย่างรวดเร็วลดข้อจำกัดต่างๆ ลง
* ทำระบบสำรองได้ง่าย - เราสามารถเพิ่มระบบเพื่อทำงานเป็นตัวหลักตัวรองหรือว่าเป็นหลายๆตัว ช่วยกันทำงานได้อย่างง่ายๆ ไม่ต้องตั้งค่าอะไรเยอะแยะ
* การเรียกข้อมูลมากแสดง - อย่างที่บอกว่าเป็นการเก็บข้อมูลแบบโครงสร้างดังนั้นเวลาเรียกข้อมูลมาแสดงก็จะได้ทั้งโครงสร้างของข้อมูล ออกมาเลย
* แก้ไขข้อมูลได้รวดเร็ว - หากเราใช้ MySQL แล้วศึกษาลึกๆ จะพบว่าการ query update จะทำให้ตารางนั้น lock จังหวะที่ update แต่ว่า MongoDB ไม่เป็นอย่างนั้น
* เขียนชุดคำสั่งการทำงานได้ - หากเรามีการทำงานหลายขั้นตอนแบบซ้ำๆ เมื่อเกิดการทำงานในลักษณะแบบใดเราก็จัดกลุ่มคำสั่งที่ทำซ้ำแล้วเขียนเหมือนเป็น script เอาไว้เลยเวลานันก็รันทั้งก้อนนี้เลย
* เก็บข้อมูลด้วยระบบ GridFS – เป็นระบบการเก็บไฟล์บนพื้นที่ Harddisk ที่เก็บข้อมูลเป็นก้อนๆ และรองรับการเพิ่มหรือลดของปริมาณข้อมูลได้
* มีบริการสอบถามและดูแลเป็นพิเศษ - มี service ดูแลอย่างดีเป็นพิเศษให้คำปรึกษาพร้อมดูแลอย่างใกล้ชิด โดยบริษัท 10gen, Inc แต่ก็แน่นอนว่าไม่ฟรีนะสำหรับการบริการพิเศษแบบนี้และอีกเรื่องที่ทำให้มันน่าสนใจก็คือ Performance ถึงแม้ว่าจะช้ากว่า NoSQL ด้วย กันเองแต่มันก็เร็วกว่า MySQL มากๆ และข้อมูลที่เก็บก็มีความน่าเชื่อถือไม่สูญหายดีในระดับหนึ่งเพราะว่า NoSQL หลายตัวที่ชูเรื่องความแรงมากๆแต่ถ้าเครื่องดับโดยไม่คาดฝันข้อมูลสูญหายหรือเสียหายไปเลยก็มี

**2.4 Metadata**

ข้อมูลที่ใช้กำกับและอธิบายข้อมูลหลักหรือกลุ่มข้อมูลอื่นหรือเรียกว่า เมทาดาทา อธิบายรายละเอียดของชุดข้อมูลภูมิสารสนเทศอาทิพื้นที่ ครอบคลุม คุณภาพ เงื่อนไข ใคร อะไร และเมื่อไหร่ ต่างๆ ซึ่งสามารถนำมาพิจารณาความเหมาะสมในการนำข้อมูลมาใช้โดยสามารถบอกได้ว่าข้อมูลเริ่มขึ้นที่ใด ขั้นตอนการผลิต ข้อมูลอรรถาอธิบายประกอบด้วยระบบโครงข่ายแผนที่ข้อมูลครอบคลุมพื้นที่บริเวณใดจะเอาข้อมูลมาได้อย่างไรจะต้องมีค่าใช้จ่ายในการได้ข้อมูลหรือไม่จะติดต่อใครเพื่อขอรายละเอียด เพิ่มเติม เป็นต้น

เมทาดาทา หมายถึงสารสนเทศที่เครื่องสามารถเข้าใจได้ขณะที่บางแห่งใช้ในการอธิบายทรัพยากรสารสนเทศอิเล็กทรอนิกส์หรือใช้เมทาดาทาในการอธิบายให้ทราบรายละเอียดของข้อมูลในเว็บเช่นเดียวกับการทำรายการ (Catalog) ของหนังสือการอธิบายเว็บเพจว่าใครเป็นเจ้าของงานนั้น งานนั้นชื่ออะไร มีหัวเรื่องคำสำคัญอย่างไรเพื่อให้สามารถสืบค้นข้อมูลสาระที่ต้องการได้โดยสะดวก และมีลักษณะเป็นอะไรส่วนใหญ่เป็นไฮเปอร์เท็กซ์ (Hypertext) ที่จะคลิกเข้าสู่เนื้อหาของข้อมูลโดยที่เนื้อหาต่างๆ เหล่านั้นอยู่ในเว็บนั่นเอง

ความหมายของ เมทาดาทา อีกความหมายหนึ่งค่อนข้างจะกว้างและหลากหลายกว่าการทำรายการทั่วไปเพราะว่าสามารถทำรายการทรัพยากรสารสนเทศที่มีอยู่ในรูปของมัลติมีเดีย และรูปภาพต่างๆ ได้ด้วย รวมทั้งสามารถทำดัชนีให้กับสื่อทุกรูปแบบโดยมีการจัดข้อมูลซึ่งใช้ระบบที่มีโครงสร้างที่เป็นมาตรฐานเพื่อที่จะให้สืบค้นได้โดยระบุจากเขตข้อมูลหรือประเด็นที่ต้องการจะค้นและสามารถเรียกดูสารสนเทศที่อ้างถึงในแต่ละเรื่องที่ค้นมาได้ซึ่งจะสามารถเชื่อมโยงไปดูสารสนเทศนั้นได้ที่สำคัญในตัวเมทาดาทาเป็นแหล่งอ้างอิงเหมือนกับเป็นบรรณานุกรมของข้อมูลที่อยู่ในอินเทอร์เน็ต แท้จริงแล้วเมทาดาทาเป็นเรื่องที่วงการบรรณารักษศาสตร์คุ้นเคยเป็นอย่างดีในส่วนของการลงรายการของทรัพยากรสารสนเทศของห้องสมุดทั้งที่เป็นสารสนเทศประเภทสิ่งพิมพ์หรือสารสนเทศดิจิทัล โดยมีการลงรายการแบบมาร์ก 21 (MARC 21) และรูปแบบการลงรายการแบบแองโกลอเมริกันฉบับพิมพ์ครั้งที่ 2 (Anglo- Ameri can Cataloging Rules II – AACR2) เป็นมาตรฐานในการลงรายการเมทาดาทาจึงไม่จำกัดเฉพาะอยู่แต่กับสารสนเทศที่เป็นดิจิทัลเท่านั้นไม่ว่าจะอยู่ในสื่อ รูปแบบใดแต่เมทาดาทาจะต้องมีคุณสมบัติ 2 ประการ คือประการที่หนึ่งเป็นข้อมูลที่มีโครงสร้างและประการที่สองเมทาดาทา ต้องอธิบายทรัพยากรสารสนเทศนั้นๆเมทาดาทาเป็นองค์ประกอบที่สำคัญในการพัฒนาระบบ GIS และประยุกต์การใช้งานให้ประสบผลสำเร็จสามารถแสดงได้ใน 2 ลักษณะเมทาดาทาป้องกันองค์กรในการลงทุนเรื่องข้อมูลเมทาดาทาเป็นเหมือนบัตรรายการเกี่ยวกับห้องสมุด ข้อมูล GIS สำหรับองค์กรบุคลากรภายในองค์กรสามารถมองเห็นได้ว่าข้อมูลใดที่เป็นประโยชน์และจะเอามาใช้ได้อย่างไรการหลีกเลี่ยงการสูญเสียของข้อมูลหรือการผลิตข้อมูลที่มีอยู่แล้วให้น้อยที่สุดเมทาดาทาช่วยสนับสนุนการแบ่งปันข้อมูลเมทาดาทาช่วยบุคลากรในการแบ่งปันข้อมูลด้วยการเตรียมการแสดงตำแหน่งประมาณการและการได้มาของข้อมูลที่บุคลากรต้องการ การจัดสร้างเมทาดาทาต้องทำตามมาตรฐานซึ่งได้มีการกำหนดความหมายของคำศัพท์และข้อมูลผู้ใช้สามารถอ่านข้อมูลอย่างง่ายและเข้าใจความหมายในทิศทางเดียวกันมาตรฐานที่สำคัญคือ ISO/TC211 (ISO1911 5:2003) Geographic Information-Metadata, ISO19115-2 Part 2: Extension for Imagery and Gridded Data และ FGDC Content Standard for Digital Geosaptial Metadata (CSDGM) มาตรฐานดัง กล่าวใช้ภาษา Extensible Markup Language: XML ในการจัดเก็บข้อมูลสำหรับการเผยแพร่และกำหนดการจัดเก็บข้อมูลเป็นส่วนต่างๆ โดยแต่ละส่วนมีรายการที่บังคับต้องใช้และไม่บังคับซึ่งขึ้นอยู่กับรายละเอียดที่ผู้จัดทำต้องการเหตุที่มีการตื่นตัวเรื่องเมทาดาทากันมากเนื่องจากในช่วงดังกล่าวมีสารสนเทศจำนวนมหาศาลในเวิล์ดไวด์เว็บ (World Wide Web; WWW) ทั้งที่เป็นข้อมูล รูปภาพ เสียง วีดิทัศน์ หลากหลายทั้งรูปแบบและภาษาที่แตกต่างกันซึ่งเป็นเอกสารที่เจ้าของผลงานผลิตขึ้นเองโดยใช้เพียงภาษาเอชทีเอ็มแอลในการกำหนดรูปแบบการแสดงผลและเชื่อมโยงข้อมูลเท่านั้นไม่มีโครงสร้างมาตรฐานสำหรับสืบค้นที่ระบุเขตข้อมูลทำให้เสิร์ชเอ็นจิ้น เช่น ยาฮู (Yahoo) อัลตาวิสต้า (Alta Vista) สามารถสืบค้นให้แบบระดับกว้างเท่านั้น และไม่ตรงกับความต้องการเท่าใดนักจึงเกิดความเคลื่อนไหวในเรื่องเมทาดาทาขึ้นคำว่าเมทาดาทามีการใช้กันอย่างหลากหลายตามความแตกต่างของสถานที่นำไปใช้บางแห่งใช้เมทาดาทาเพื่อหมายถึงสารสนเทศที่เครื่องสามารถเข้าใจได้ขณะที่บางแห่งใช้ในการอธิบายทรัพยากรสารสนเทศอิเล็กทรอนิกส์หรือใช้เมทาดาทาในการอธิบาย ให้ทราบรายละเอียดของข้อมูลในเว็บ เช่นเดียวกับการทำรายการ (Catalog) ของหนังสือการอธิบายเว็บเพจ ว่าใครเป็นเจ้าของงานนั้นงานนั้นชื่อ อะไรมีหัวเรื่องความสำคัญอย่างไรเพื่อให้สามารถสืบค้นข้อมูลสาระที่ต้องการได้โดยสะดวกและมีลักษณะเป็นอะไร ส่วนใหญ่เป็นไฮเปอร์เท็กซ์ (Hyper text) ที่จะคลิกเข้าสู่เนื้อหาของข้อมูลโดยที่เนื้อหาต่างๆ เหล่านั้นอยู่ในเว็บนั่นเอง

ความจำเป็นที่ต้องมีเมทาดาทา

เกิดขึ้นเนื่องจากสารสนเทศที่สร้างขึ้นประกอบด้วยองค์ประกอบ 3 ลักษณะ คือ

1. เนื้อหา (Content) ของงานเกี่ยวกับชื่อเรื่อง หัวเรื่อง ต้นฉบับ (แหล่งที่มา) ภาษา เรื่องที่เกี่ยวข้องและขอบเขต

2. บริบท (Context) ของสารสนเทศเกี่ยวกับทรัพย์สินทางปัญญาของงาน เช่น ผู้เขียน ผู้สร้างสรรค์ผลงาน สำนักพิมพ์ผู้มีส่วนร่วมในผลงาน และสิทธิในงานนั้นๆ

3. โครงสร้าง (Structure) ของข้อมูลเกี่ยวกับ วันเดือนปีที่สร้างผลงาน ประเภทของเนื้อหารูปแบบการนำเสนอผลงานและตัวบุ่งชี้หรือตัวระบุถึงทรัพยากร

**2.5 เซ็นเซอร์ที่เป็นตัวอย่างในโครงงาน**

**2.5.1 Sensor DHT22**

เป็นเซ็นเซอร์ที่ใช้ในการตรวจวัดอุณหภูมิและควมาชื้นสัมพัทธ์ มีการรักษาความน่าเชื่อถือและเสถียรภาพของตัวเครื่อง เซ็นเซอร์รุ่นนี้มีการเชื่อมต่อกับคอมพิวเตอร์แบบชิพแบบ 8 บิต เซ็นเซอร์ของรุ่นนี้ได้รับการชดเชยอุณหภูมิและเปรียบเทียบในห้องสอบเทียบที่ถูกต้องและค่าสัมประสิทธิ์การวัดจะถูกบันทึกไว้ในรูปแบบของโปรแกรมหน่วยความจำ มีระยะในการวัดได้ถึง 20 เมตรทำให้ DHT22 สามารถใช้งานได้ทุกรูปแบบ แถวเดียวบรรจุด้วยสี่หมุดทำให้การเชื่อมต่อสะดวก

**2.5.2 Sensor MQ-7**

เป็นเซ็นเซอร์ชนิดตรวจสอบคาร์บอนมอนนอกไซต์ในอากาศที่ใช้งานง่ายและเหมาะสำหรับการตรวจจับความเข้มข้นของคาร์บอนมอนนอกไซต์ในอากาศ MQ-7 สามารถตรวจับความเข้มได้ตั้งแต่ 20 ถึง 2000 ppm และยังมีความไวสูงตอบสนองได้อย่างรวดเร็ว

**2.5.3 Sensor Pulse Heart Rate**

เซ็นเซอร์ตรวจจับอัตราการเต้นของหัวใจโดยการใช้นิ้วแตะที่เซ็นเซอร์ ข้อมูลอัตราการเต้นของหัวใจเป็นประโยชน์อย่างแท้จริงไม่ว่าคุณจะออกแบบกิจกรรมการออกกำลังกายศึกษากิจกรรมระบบดับความวิตกกังวลของคุณ เซ็นเซอร์ตัวนี้เป็นการวัดอัตราการเต้นของหัวใจแบบพลักแอนด์เพลย์สำหรับ Arduino

**2.5.4 Sensor GY-521**

เป็นโมดูล Accelerometers and Gyroscope ซึ่งสามารถทำงานได้ทั้ง 2 อย่างในเวลาเดียวกันใช้ในการตรวจสอบทิศทางการเคลื่อนที่และสามารถใช้ในการตรวจสอบความเร็วในการเปลี่ยนแปลงทิศทางของแกน xyz ได้ยกตัวอย่างถ้าวัตถุเกิดการเคลื่อนที่หรือเอียง output ของ Accelometer จะบอกค่าของการเอียงว่าสถานะปัจจุบันค่า xyz อยู่ที่เท่าไหร่ Gyroscope จะว่าค่าได้ตอนที่กำลังเอียงหรือตอนกำลังเคลื่อนไหวเท่านั้นเมื่อวัตถุหยุดนิ่งค่าของ Gyroscope จะวัดไม่ได้เพราะไม่มีการเคลื่อนไหว

**2.6 JSON**

**เจสัน เป็นรูปแบบสายอักขระ** (String) **ชนิดหนึ่งที่ถูกจัดเก็บในรูปแบบที่สามารถอ่านและเข้าใจได้ง่าย ถ้าจะให้เข้าใจง่าย ๆ ก็คือเจสันเปรียบเสมือนรูปแบบของ อาเรย์** (Array) **ชนิดหนึ่งที่ใช้รับส่งข้อมูลผ่านอาแจ็กซ์เพราะซึ่งปกติแล้วถ้าเราต้องการรับ**-**ส่งข้อมูลผ่านอาแจ็กซ์ต่าง ๆ นั้น จะต้องรับ**-**ส่งมาในรูปแบบของสายอักขระทั้งก้อน และเมื่อฝั่งอาแจ็กซ์ทำการรับค่าที่ทำการส่งค่ากลับมาจากเซิร์ฟเวอร์ ก็จะต้องนำสายอักขระ เหล่านั้นมาตัดตำแหน่งที่ต้องการ เพื่อเอาสายอักขระตัวที่ต้องการมาใช้ แต่สำหรับเจสันแล้ว สามารถรับส่งชุดค่าตัวแปรได้ทั้งฝั่งไคลเอนต์** (Client) **และฝั่งเซิร์ฟเวอร์** (Server) **โดยทั้ง** 2 **ฝั่งสามารถทำการเข้ารหัสและถอดรหัสโดยใช้เจสันเอนโค้ด** (Json Encode) **และ เจสันดีโค้ด** (Json Decode) **เพื่ออ่านค่าตัวแปรเหล่านั้น และจะเรียกใช้งานมันได้อย่างไร ซึ่งปกติแล้วจะอยู่ในรูปแบบของอาเรย์และสำหรับตัวแปรเจสันนั้นไม่จำกัดแค่รับส่งข้อมูลผ่านเว็บเบราว์เซอร์เท่านั้น แต่ยังสามารถนำเจสันไปประยุกต์กับการรับส่งข้อมูลในรูปแบบอื่น ๆ ได้ เช่นการจับเก็บข้อมูลในรูปแบบของ สายอักขระในข้อความหรือการรับส่งผ่านตัวให้บริการเว็บไซต์** (Web Service) **ก็สามารถทำได้เช่นเดียวกัน**  
JSON นั้นใช้ความสัมพันธ์ของภาษาจาวาสคริปต์ แต่ไม่ถูกมองว่าเป็นภาษาโปรแกรม กลับถูกมองว่าเป็นภาษาในการแลกเปลี่ยนข้อมูลมากกว่า ในปัจจุบันมีไลบรารีของภาษาโปรแกรมอื่นๆ ที่ใช้ประมวลผลข้อมูลในรูปแบบ JSON มากมาย

**2.7 REST API**

Representational state transfer หรือ REST คือ การสร้าง Webservice**ชนิดหนึ่งที่ใช้สื่อสารกันบน** Internet **ใช้หลักการแบบ** stateless **คือไม่มี** session **ซึ่งต่างจาก** webservice**แบบอื่นเช่น**[WSDL](https://saixiii.com/what-is-wsdl/)**และ**[SOAP](https://saixiii.com/what-is-soap/)**การทำงานของ** RESTful[Webservice](https://saixiii.com/what-is-webservice/)**จะอาศัย** URI/URL **ของ** request **เพื่อค้นหาและประมวลผลแล้วตอบกลับไปในรูป**[XML](https://saixiii.com/what-is-xml/), HTML,[JSON](https://saixiii.com/what-is-json/)**โดย** response **ที่ตอบกลับจะเป็นการยืนยันผลของคำสั่งที่ส่งมา และสามารถพัฒนาด้วยภาษา** programming **ได้หลากหลายคำสั่งก็จะมีตาม** HTTP verbs **ซึ่งก็คือ**

* GET ทำกการดึงข้อมูลภายใน URI ที่กำหนด
* POST สำหรับสร้างข้อมูล
* PUT ใช้แก้ไขข้อมูล
* DELETE สำหรับลบข้อมูล

REST ถูกตั้งขึ้นโดย Roy Fielding ในปี 2000 ที่University of California, Irvine ซึ่งได้ทำการพัฒนาขึ้นมาควบคู่กับ HTTP1.1 และUniform Resource Identifiers (URI)

คุณสมบัติของ REST

เป็น[API](https://saixiii.com/what-is-api/)อย่างหนึ่ง ซึ่งทุกๆ system ต่างใช้ resource ซึ่งเป็นได้ทั้ง image, video, web page หรือข้อมูลทางธรุกิจ ก็ได้ที่สามารถแสดงบนระบบ computer วัตถุประสงค์เพื่อให้ user สามารถเข้าถึง, ติดตั้ง, ปรับแต่ง, ขยาย resource เหล่านี้ได้ง่าย ซึ่งทาง RESTful ได้ออกแบบมาให้มีคุณสมบัติต่อไปนี้ แสดงผล, เก็บข้อมูล, มี URIs, Stateless ทำงานโดยไม่ต้องมี session, เชื่อมต่อระหว่าง web service, Caching

**2.8 ReactJS**

React เป็น Javascript Library ที่เอาไว้สำหรับทำ UI ถูกพัฒนาขึ้นมาโดยทาง Facebook และเปิดให้คนทั่วไปนำมาใช้ฟรี ซึ่งเว็บไซต์ในปัจจุบันของ Facebook.com ก็ใช้ React อยู่เช่นกัน  
3 องค์ประกอบใน React ได้แก่   
1. Component คือส่วนต่างๆ ในเว็บจะถูกแบ่งเป็นส่วนๆ ไว้แล้วนำมารวมกันไว้เป็น Website  
2. State คือข้อมูลที่อยู่ใน Component แต่ละส่วนจะถูกเรียกว่า state  
3. Props คือข้อมูลที่ถูกส่งต่อจาก Component จากชั้นบนลงไปชั้นล่างเรียกว่า Props (Properties)

**2.9 NETPIE**

NET-PIE เป็น Platform as a Service บริการการเชื่อมต่อข้อมูลและแลกเปลี่ยนข้อมูลระหว่างอุปกรณ์ต่าง ๆ ได้ เหมาะสำหรับ Internet of Things ซึ่งรูปแบบการใช้งานจะเหมือนกับ MQTT พ่วงความสามารถด้าน Authentication และ Token เพิ่มความปลอดภัยในการสื่อสาร เราสามารถเชื่อมต่อกับ NET-PIE ได้หลายช่องทาง ทั้งอุปกรณ์ IoT อย่าง Arduino, Raspberry Pi, ESP8266 Dev Kit, HTML5 หรือแม้กระทั่ง Node.js Application ดังนั้นหากคุณใช้ ESP8266 Dev Kit ของค่ายใดอยู่ก็ตามสามารถใช้งานได้เลย ดู Library ของอุปกรณ์ต่าง ๆ ได้ที่หน้าโครงการใน GitHub NET-PIE ให้บริการโดยศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ (NECTEC)

**2.10 NETPIE FEED**

NETPIE FEED **ข้อมูลที่เกิดจากการอ่านค่าของอุปกรณ์เซนเซอร์ในบทที่แล้ว สามารถนามาใช้ให้เกิดประโยชน์ได้อย่างหลากหลาย เช่น ใช้สาหรับการตรวจสอบ** (Monitoring) **หรือการแสดงผลของข้อมูล** (Data Visualization) **โดยเฉพาะอย่างยิ่งข้อมูลแบบทันเวลา** (Real-time Data) **ซึ่งจะต้องเก็บรวบรวมด้วยอัตราความถี่ที่เหมาะสมเพื่อให้การตรวจสอบหรือการแสดงผลตรงตามความต้องการ ในปัจจุบันแพลตฟอร์ม** NETPIE **มีบริการที่สามารถเก็บข้อมูลและแสดงผลที่เรียกว่า** Feed **ซึ่งทาหน้าที่เสมือนถังเก็บข้อมูล ประเภท** Time-Series **กล่าวคือ เป็นชุดข้อมูลหรือค่าตัวแปร ณ เวลาต่างๆ เช่นอุณหภูมิอากาศในช่วงเวลาต่างๆ ของวัน เป็นต้น ข้อมูลเหล่านี้จะถูกเก็บแบบต่อเนื่องสะสมกันไปตลอด และสามารถเรียกออกมาดูในช่วงเวลาใดก็ได้ ในบทนี้จะอธิบายรายละเอียดการใช้งาน** NETPIE Feed **ตั้งแต่การตั้งค่าการเก็บข้อมูลจาก** Datasource **ไปจนถึงการแสดงผล**

**2.11 NodeMCU**

NodeMCU (**โหนด เอ็มซียู**) **คือ บอร์ดคล้าย**[Arduino](https://poundxi.com/arduino-คืออะไร)**ที่สามารถเชื่อมต่อกับ** WiFi **ได้**,**สามารถเขียนโปรแกรมด้วย** Arduino IDE**ได้เช่นเดียวกับ** Arduino **และบอร์ดก็มีราคาถูกมากๆ เหมาะแก่ผู้ที่คิดจะเริ่มต้นศึกษา หรือทดลองใช้งานเกี่ยวกับ** Arduino, IoT, **อิเล็กทรอนิกส์ หรือแม้แต่การนำไปใช้จริงในโปรเจคต่างๆ ก็ตาม เพราะราคาไม่แพง**

ภายในบอร์ดของ NodeMCU ประกอบไปด้วย ESP8266 (ไมโครคอนโทรลเลอร์ที่สามารถเชื่อมต่อ WiFi ได้) พร้อมอุปกรณ์อำนวยความสะดวกต่างๆ เช่น พอร์ต micro USB สำหรับจ่ายไฟ/อัปโหลดโปรแกรม, ชิพสำหรับอัปโหลดโปรแกรมผ่านสาย USB, ชิพแปลงแรงดันไฟฟ้า และขาสำหรับเชื่อมต่ออุปกรณ์ภายนอก เป็นต้น

**บทที่ 3**

**ขั้นตอนและวิธีการดำเนินงาน**

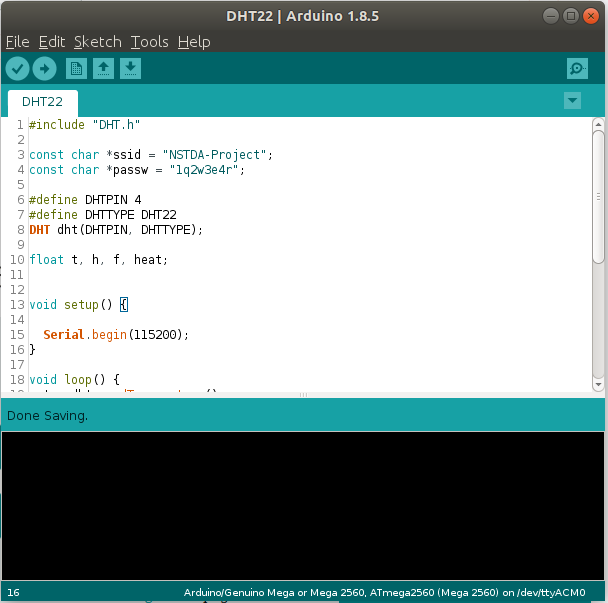
3.1 ออกแบบ Data format ของเซ็นเซอร์ต่างๆ

3.1.1 เซ็นเซอร์ DHT22

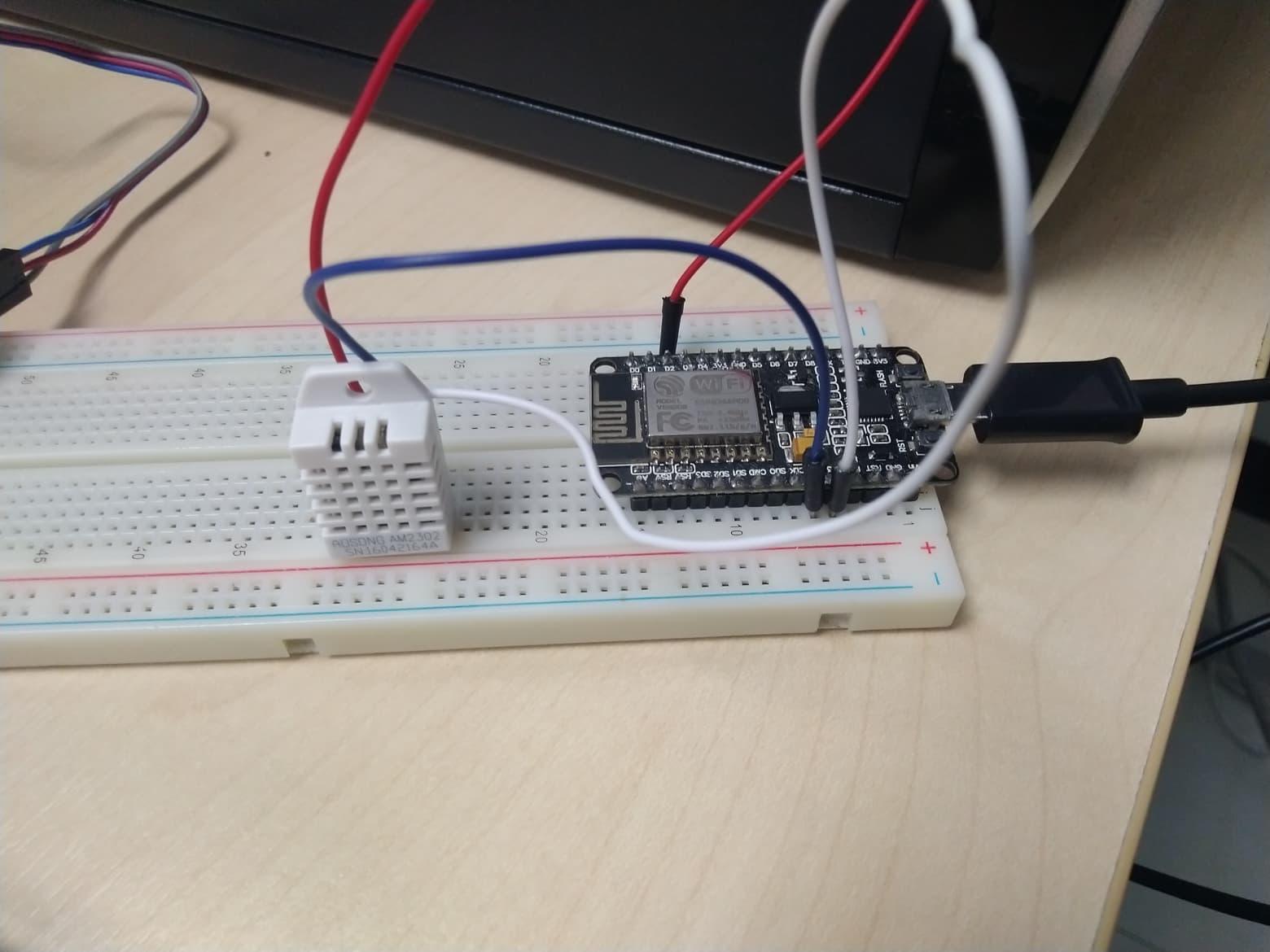
1. ศึกษาการทำงานของเซ็นเซอร์ DHT22

DHT22 เป็นโมดูลวัดอุณหภูมิและความชื้นในตัวเดียว มีความแม่นยำสูงมีตัว ต้านทานและสามารถต่อขาได้สะดวกและง่ายมาก

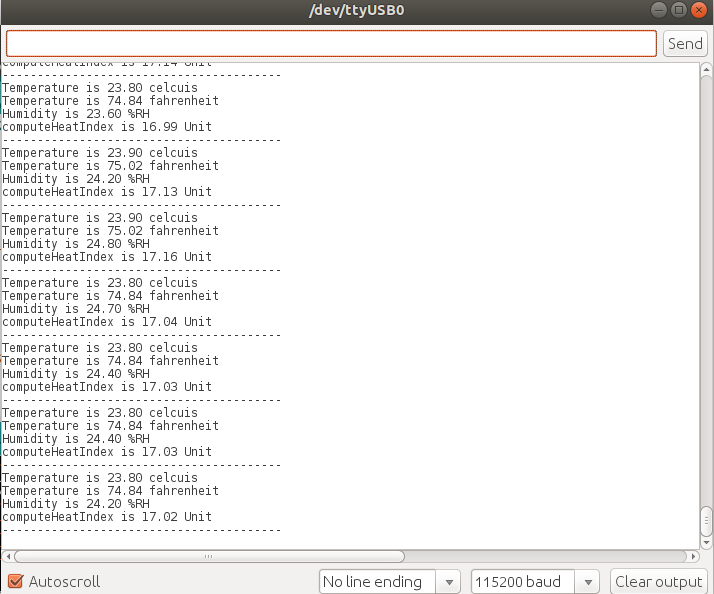
ศึกษาการทำงานของ DHT22 โดยใช้โปรแกรม Arduino IDE ในการทดลอง

  
ภาพที่ 3 - 1 : Program Arduino IDE

2. ทำการต่อวงจรระหว่าง DHT22 กับ ESP8266NodeMCUโดยการต่อวงจรนั้นจะอ้างอิงขาเอ้าพุทอินพุทตามที่เราเขียนโค้ดไว้

  
ภาพที่ 3 - 2 : ภาพแสดงการต่อวงจรระหว่าง ESP8266 กับ DHT22

3. ทำการเขียนโค้ดเพื่อให้โมดูลของ DHT22 ทำงานและศึกษาการทำงานของ DHT22 เพื่อดูว่ามีการตรวจจับค่าอะไรบ้างและสามารถส่งค่าอะไรได้บ้างเพื่อนำมาออกแบบ Data format

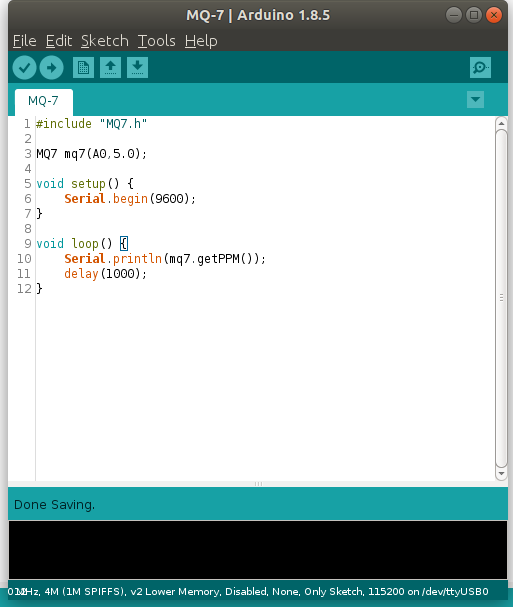
  
ภาพที่ 3 - 3 : ภาพแสดงผลลัพธ์ของ DHT22

4. เมื่อได้ศึกษาการเก็บค่าให้นำมาคิดออกแบบ Data format ของ DHT22

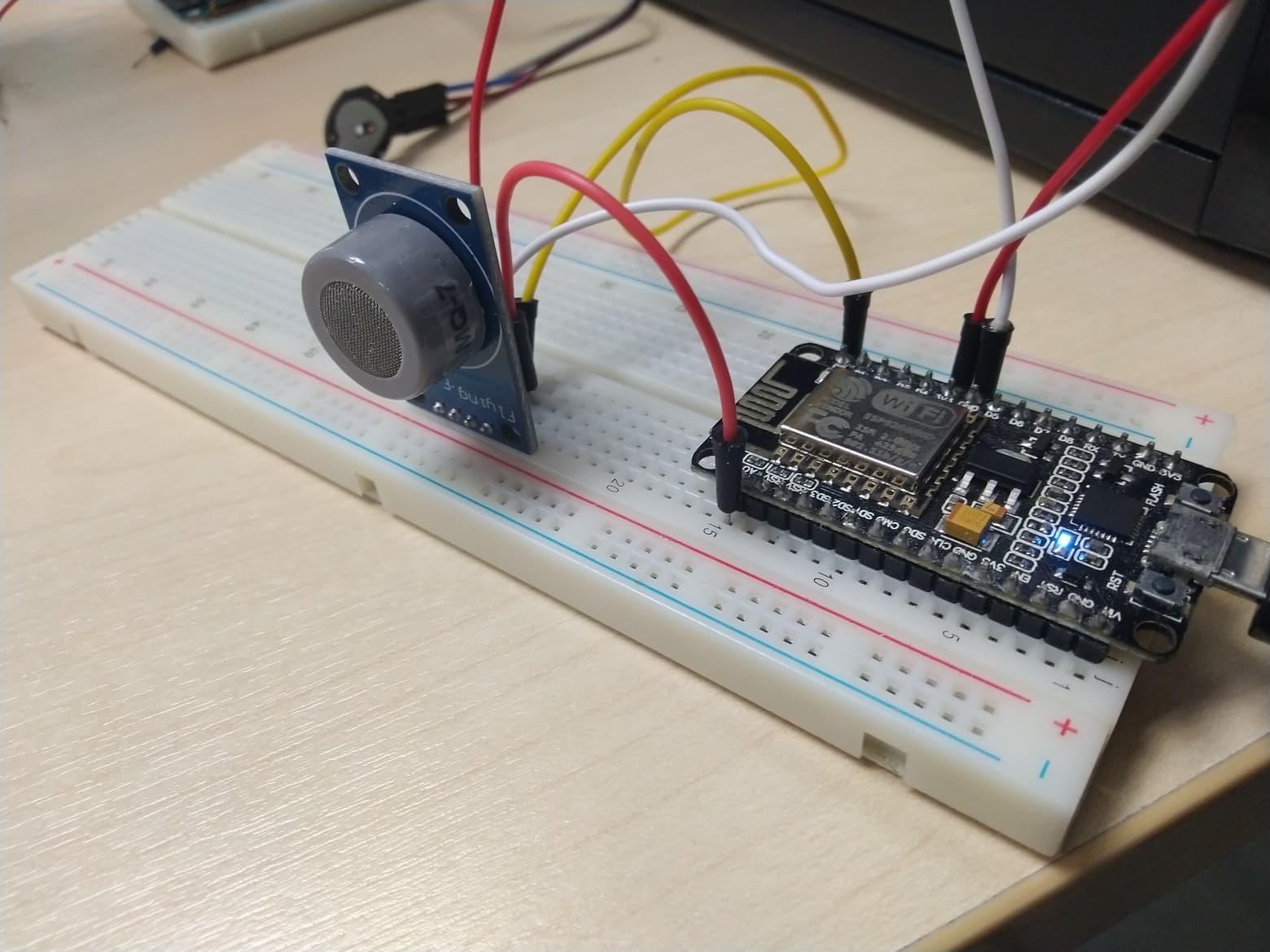
3.1.2 เซ็นเซอร์ MQ-7

1. ศึกษาการทำงานของเซ็นเซอร์ MQ-7

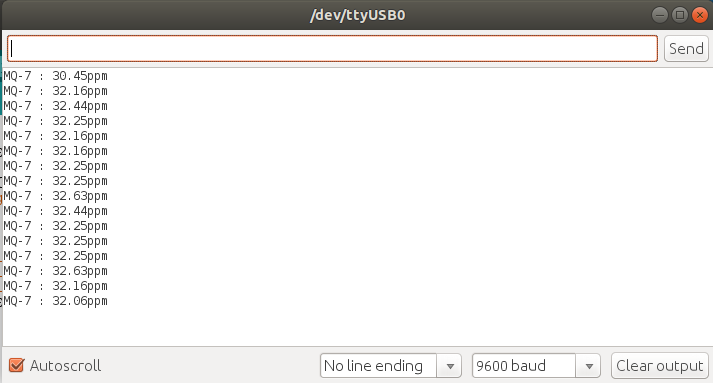
MQ-7 เป็นชุดเซ็นเซอร์ตรวจวัดความหนาแน่นของก๊าซคาร์บอนมอนน็อกไซต์ ในอากาศโดยสามารถวัดได้ความละเอียดอยู่ในช่วง 20 – 2000 ppm โมดูลทำงานที่ 5V-DC

  
ภาพที่ 3 - 4 : ภาพแสดงโค้ดของ MQ-7 บนโปรแกรม Arduino IDE

2. ทำการต่อวงจรระหว่าง MQ-7 กับ ESP8266NodeMCU

  
ภาพที่ 3 - 5 : ภาพแสดงการต่อวงจรระหว่าง ESP8266 กับ MQ-7

3. ทำการเขียนโค้ดเพื่อให้โมดูลของ MQ-7 ทำงานและศึกษาการทำงานของ MQ-7 เพื่อดูว่ามีการตรวจจับค่าอะไรบ้างและสามารถส่งค่าอะไรได้บ้างเพื่อนำมาออกแบบ Data format

  
ภาพที่ 3 - 6 : ภาพแสดงผลลัพธ์ที่ได้ของ MQ-7

4. เมื่อได้ศึกษาการเก็บค่าที่ได้ให้นำมาคิดออกแบบ Data format ของ MQ-7

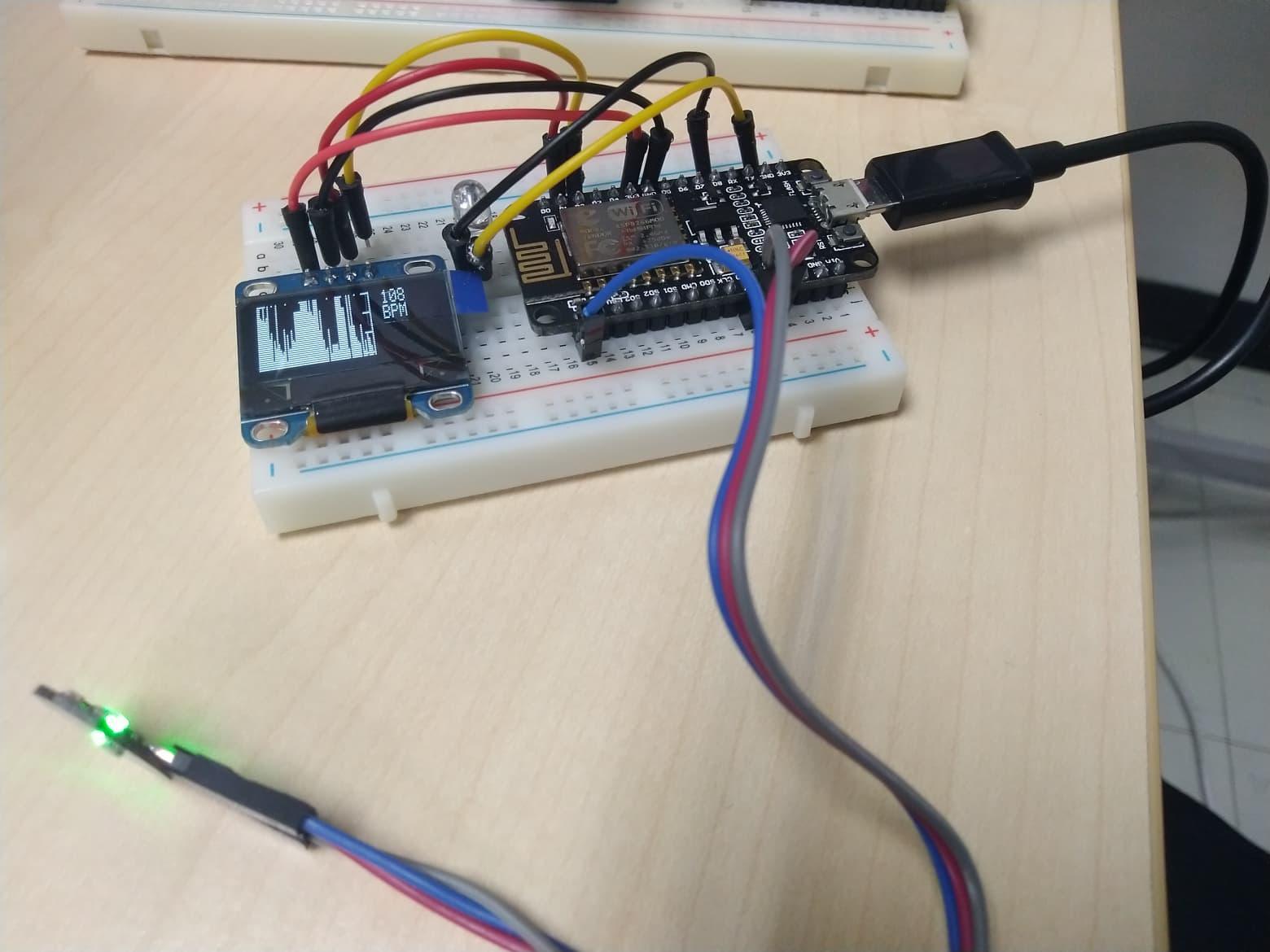
3.1.3 เซ็นเซอร์ Pulse Heart Rate

1. ศึกษาการทำงานของเซ็นเซอร์ Pulse Heart Rate

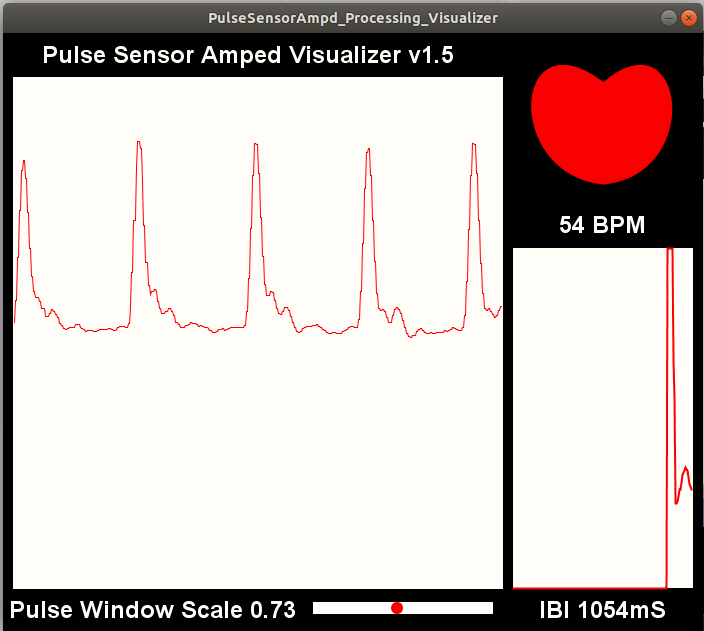
Pulse Heart Rate เป็นโมดูลไว้ทำงานเกี่ยวกับการตรวจจับชีพจรหัวใจและวัดค่าการเต้นของหัวใจว่าได้กี่ครั้งต่อนาทีและแสดงผลคลื่นหัวใจเป็นกราฟ

  
ภาพที่ 3 - 7 : ภาพแสดงโค้ดของ Pulse Heart Rate บนโปรแกรม Arduino

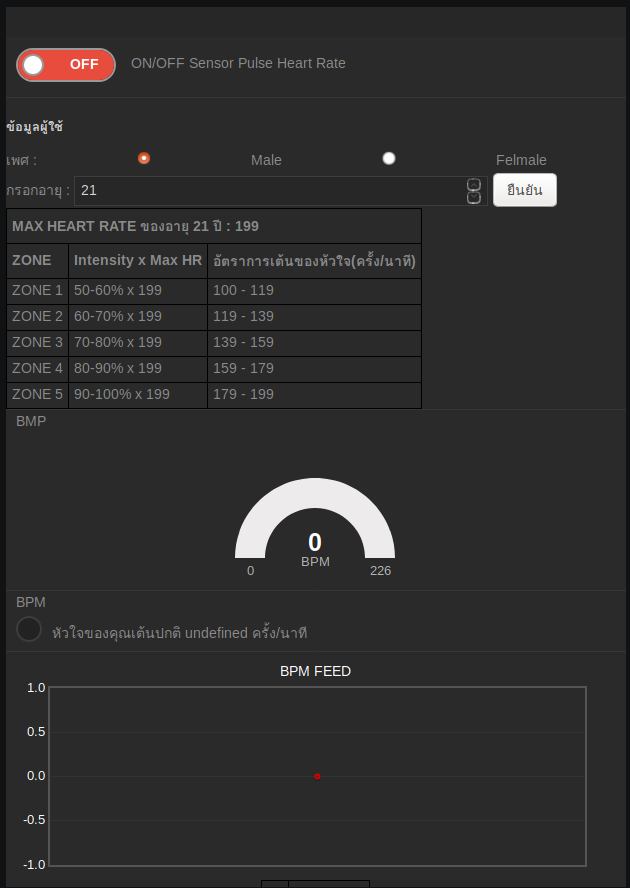
2. ทำการต่อวงจรระหว่าง Pulse Heart Rate กับ ESP8266NodeMCU

  
ภาพที่ 3 - 8 : ภาพแสดงการต่อวงจรระหว่าง Pulse Heart Rate กับ ESP8266

3. ทำการเขียนโค้ดเพื่อให้โมดูลของ Pulse Heart Rate ทำงานและศึกษาการทำงานของ Pulse Heart Rate เพื่อดูว่ามีการตรวจจับค่าอะไรบ้างและสามารถส่งค่าอะไรได้บ้างเพื่อนำมาออกแบบ Data format

  
ภาพที่ 3 - 9 : ภาพแสดงการทำงานของเซ็นเซอร์ Pulse Heart Rate

4. ทำการเขียนโค้ดเพื่อทดลองส่งค่าไปยัง NETPIE และสั่งการ Pulse Heart Rate ผ่าน NETPIE และต้องการจะทำให้เซ็นเซอร์ทำเป็น Simple wearable

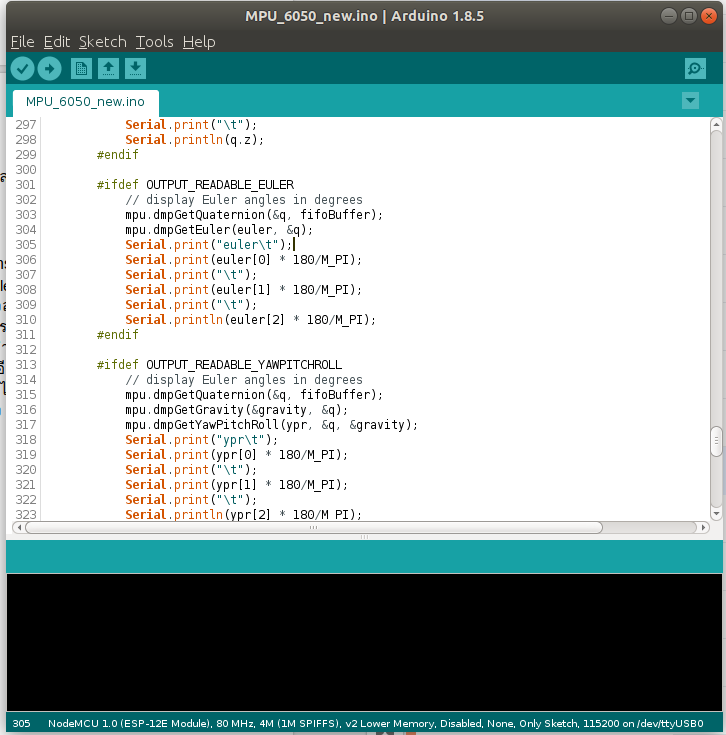
  
ภาพที่ 3 - 10 : ภาพแสดงการทำงานบน NETPIE Freeboard

5. เมื่อได้ศึกษาการเก็บค่าที่ได้ให้นำมาคิดออกแบบ Data format ของ Pulse Heart Rate

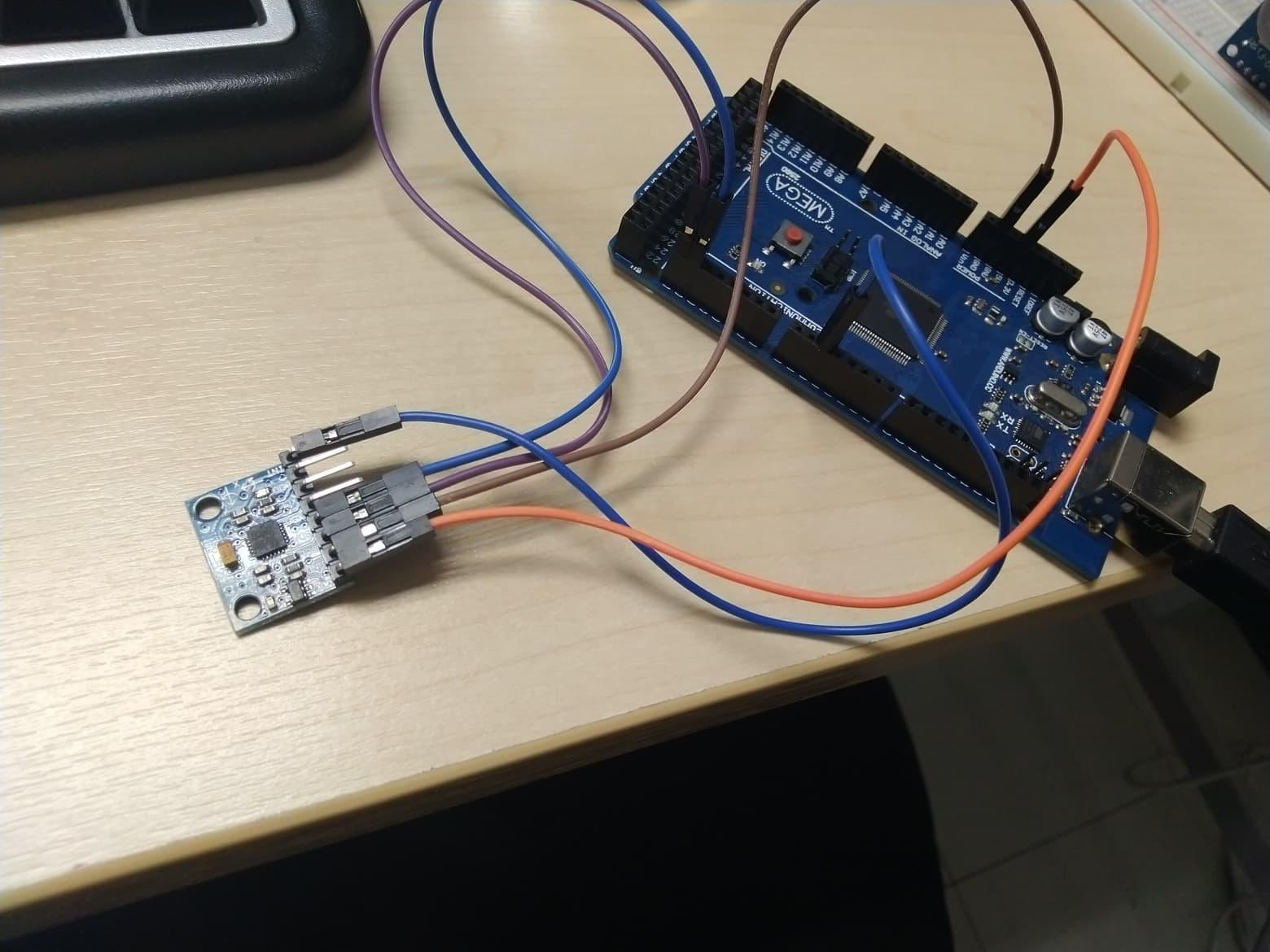
3.1.4 เซ็นเซอร์ GY-521

1. ศึกษาการทำงานของเซ็นเซอร์ GY-521

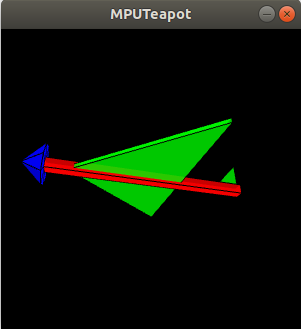
GY-521 เป็นเซ็นเซอร์ชนิดตรวจสอบคาร์บอนมอนนอกไซต์ในอากาศที่ใช้งานง่ายและเหมาะสำหรับการตรวจจับความเข้มข้นของคาร์บอนมอนนอกไซต์ในอากาศ MQ-7 สามารถตรวจับความเข้มได้ตั้งแต่ 20 ถึง 2000 ppm และยังมีความไวสูงตอบสนองได้อย่างรวดเร็ว

  
ภาพที่ 3 - 11 : ภาพแสดงโค้ดของ GY-521 บนโปรแกรม Arduino

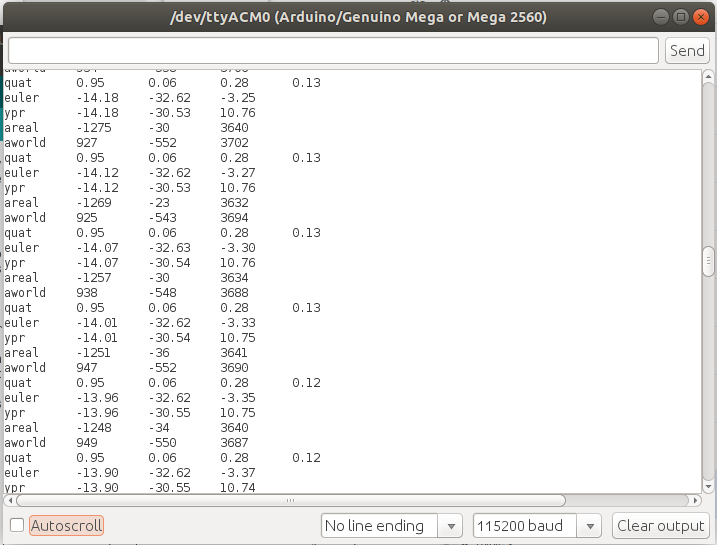
2. ทำการต่อวงจรระหว่าง GY-521 กับ ESP8266NodeMCU

  
ภาพที่ 3 - 12 : ภาพแสดงการต่อวงจรระหว่าง GY-521 กับ Arduino MEGA

3. ทำการเขียนโค้ดเพื่อให้โมดูลของ GY-521 ทำงานและศึกษาการทำงานของ GY-521 เพื่อดูผลลัพธ์โดยใช้ Programd Processing IDE เพื่อแสดงหน้า Interface เพื่อง่ายต่อการศึกษาการใช้งานเซ็นเซอร์

  
ภาพที่ 3 - 13 : ภาพแสดงผลลัพธ์ของเซ็นเซอร์แบบหน้าจอ Interface

4. ทำการเขียนโค้ดเพื่อให้แสดงผลอีกแบบก็คือแสดงเป็นค่าตัวเลข เพื่อง่ายต่อการเก็บข้อมูล

  
ภาพที่ 3 - 14 : ภาพแสดงผลลัพธ์ของเซ็นเซอร์ GY-521

5. เมื่อได้ศึกษาการเก็บค่าที่ได้ให้นำมาคิดออกแบบ Data format ของ GY-521

3.2 ระบบฐานข้อมูล mongoDB

3.2.1 ศึกษาข้อมูลเบื้องต้นของระบบฐานข้อมูล mongoDB

3.2.2 ศึกษาโครงสร้างข้อมูลของ JSON เพื่อนำมาเก็บในฐานข้อมูล

3.2.3 ทำการติดตั้งระบบฐานข้อมูล mongoDB ลงเครื่องคอมพิวเตอร์

ขั้นตอนการติดตั้ง mongoDB บน Ubuntu 18.04

1. import public key ของแพ็คเกจโดยใช้คำสั่งด้านล่างนี้

“ $ sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 2930ADAE8CAF5059EE73BB4B58712A2291FA4AD5 ”

2. ทำการสร้าง source list เพื่อบอกให้ระบบรู้จัก mongoDB ด้วยคำสั่ง

“ $ echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu xenial/mongodb-org/3.6 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-3.6.list ”

3. ทำการอัพเดทระบบปฏิบัติการ ด้วยคำสั่ง

“ $ sudo apt-get update ”

4. ติดตั้ง mongoDB ด้วคำสั่ง

“ $ sudo apt-get install -y mongodb-org ”

5. คำสั่งเปิดใช้งานเซอร์วิส mongoDB

$ sudo service mongod start // ทำการ Start

$ sudo service mongod stop // ทำการปิด Service

$ sudo service mongod restart // ทำการ Restart Service

3.2.4 ทดลองเชื่อมต่อกับฐานข้อมูล mongoDB

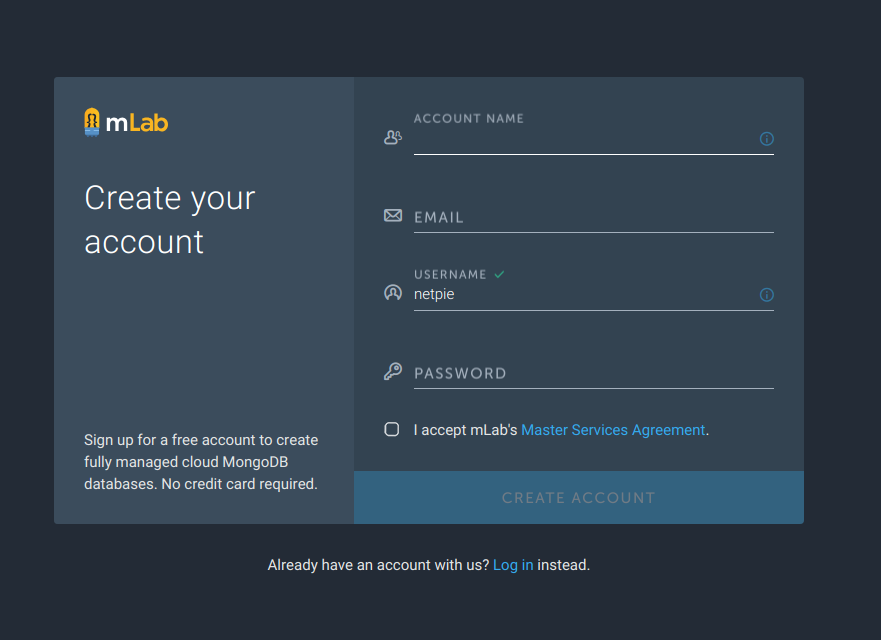
3.2.5 ทดลองเพิ่ม ลบและแก้ไขข้อมูลลงไปในฐานข้อมูล

3.2.6 เป็นการไปใช้ mongoDB ออนไลน์โดยใช้บริการของ mLab

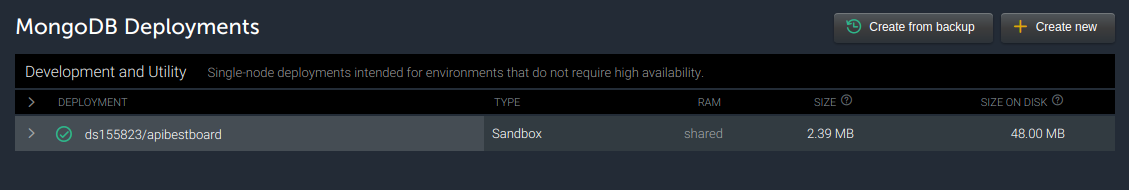
ขั้นตอนแรกให้ทำการสมัครมาชิกเพื่อล็อกอินเข้าใช้งาน mLab ที่ลิ้งด้านล่าง

<https://mlab.com/> สมัครสมาชิก

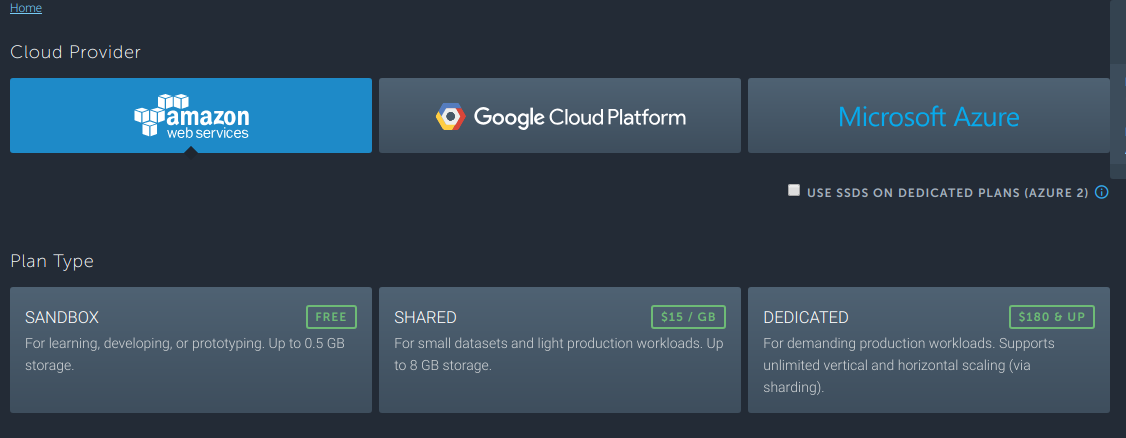
เมื่อสมัครเสร็จแล้วให้ทำการล๊อคอินเข้ามาเพื่อมาสร้างฐานข้อมูล mongoDB โดย การกดปุ่ม Create new บนขวา

  
ภาพที่ 3 - 15 : หน้าสมัครสมาชิกของ mLab

จากนั้นทำการเลือกบริการของแต่ละค่ายที่เราต้องการในกรณีศึกษานี้จะใช ของAWS Amazon web Service เพราะว่าใช้ฟรี 500MB

  
ภาพที่ 3 - 16 : สร้างฐานข้อมูล mongoDB

ขั้นต่อไปก็เลือกภูมิภาคของ server ของเราโดยของ Amazon ฟรีจะมีแค่ใน Europe และUS East เท่านั้น แนะนำให้เลือก US East

  
ภาพที่ 3 - 17 : เลือกบริการของแต่ละค่าย

จากนั้นทำการตั้งชื่อของฐานข้อมูลที่เราต้องการแล้วกดยืนยันออเดอร์เป็นอันเสร็จ

3.3 การรับข้อมูลและการส่งข้อมูลของอุปกรณ์ IoT สำหรับระบบ Digital Twin

3.3.1 ศึกษาการทำงานของ NETPIE

3.3.2 ทดลองใช้งานบริการของ NETPIE ทั้ง 3 อย่าง

1. NETPIE Application

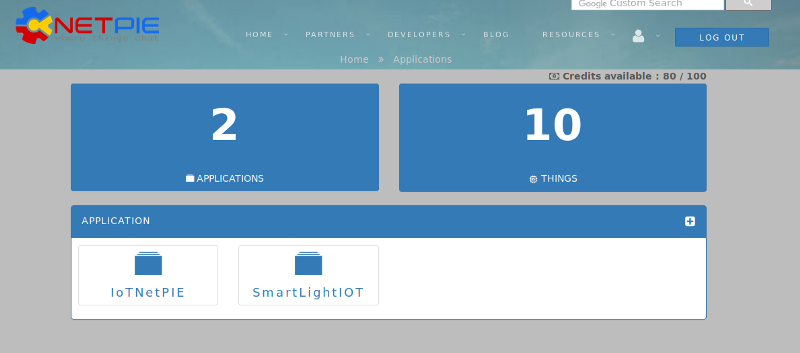
สร้าง Application ขึ้นมาเพื่อใช้งานบริการและรับส่งข้อมูลไปยัง NETPIE

ขั้นตอนแรกล๊อคอินเข้าไปใช้ในระบบของ NETPIE ที่ลิ้งด้านล่างนี้

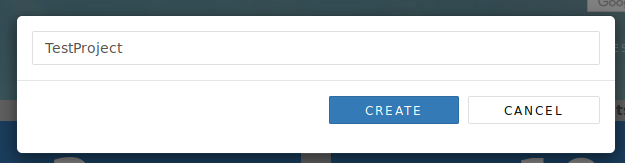
<https://netpie.io/login> ลิ้งล็อกอิน

เมื่อล๊อกอืนเข้ามาแล้วให้ทำการ้าสร้าง Application โดยกดที่ปุ่ม Resouce บนขวา แล้วเลือก Application (Resource → Application)

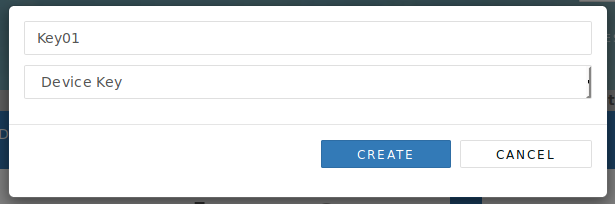
จากนั้นทำการสร้าง Application และตั้งชื่อตามต้องการโดยการกดปุ่ม + ทางขวา

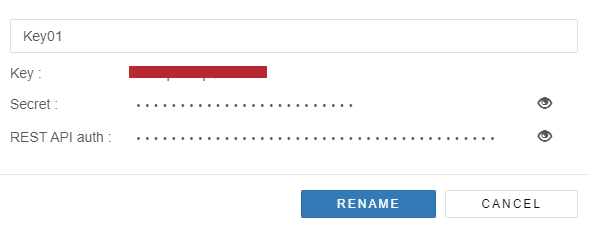
ภาพที่ 3 - 18 : แสดง Application NETPIE

ขั้นตอนต่อไปจะเป็นการสร้าง Key เพื่อไปต่อกับอุปกรณต่างๆที่เราต้องการ

ภาพที่ 3 - 19 : ตั้งชื่อ Application

เมื่อสร้างเสร็จแล้วจะได้ Key และ Secret Key ของ NETPIE ไปใช้เพื่อไปเชื่อมต่อ กับอุปกรณ์

ภาพที่ 3 - 20 : สร้าง Key ให้กับ Application

  
ภาพที่ 3 - 21 : แสดง Key และ Secret Key ที่ได้มา

2. NETPIE Freeboard

สร้าง Freeboard ขึ้นมาเพื่อมาใช้งานในการแสดงผลและควบคุมเซ็นเซอร์และ ทำการศึกษาการทำงานต่างๆของ NETPIE

3. NETPIE Feed

สร้าง Feed เพื่อนำมาใช้เก็บข้อมูลและนำมาวาดกราฟเพื่อง่ายต่อการแสดงผล

3.3.3 นำ NETPIE มาใช้ในงานเพื่อนำมาสร้าง Dashboard IoT

3.4 REST API

3.4.1 ทำการศึกษาเกี่ยวกับ REST API เพื่อนำมาใช้งานกับฐานข้อมูล mongoDB

3.4.2 ทำการเขียนการทำงานให้กับ API โดยใช้ express บน nodeJS

express เป็น web application framework บน NodeJS ที่ได้รับความนิยมมากๆ ซึ่งมีความยืดหยุ่นสูงมาก สามารถเอาไว้สร้างเว็บไซต์ธรรมดาก็ได้ หรือจะใช้ทำ Single Page Application โดยใช้ตัว Express เป็นตัว API Server คอยรับส่งข้อมูลผ่าน RESTFul ก็ได้เช่นกัน

3.4.3ทำให้ API ที่เขียนสามารถเชื่อมต่อได้กับฐานข้อมูล mongoDB โดยใช้ library mongoose เพื่อง่ายต่อการเชื่อมต่อ

mongoose เป็น ODM (Object Document Mapping) ตัวหนึ่งของ mongoDB มีหน้าที่ในการเปลี่ยนแปลง mongoDB Object mapping เป็น JavaScript เพื่อใช้ใน Application

3.4.4 เชื่อมต่อ API เข้ากับเซ็นเซอร์เพื่อให้เซ็นเซอร์สามารถบันทึกข้อมูลลงในฐานข้อมูลได้

3.4.5 นำข้อมูลที่ได้มาไปใช้งานบนแดชบอร์ดที่จะนำไปใช้ในการแสดงผล

3.5 Interface Dashboard IoT

3.5.1 ศึกษาการทำงานของ Dashboard IoT โดยดูตัวอย่างจากที่อื่น เช่น

1. Freeboard : https://freeboard.io/

2. Grafana : https://grafana.com/

3. ThingBoard : https://thingsboard.io/

เพื่อนำมาใช้ในการออกแบบ Interface

3.5.2 ศึกษาการทำงานและการใช้งานของ ReactJS เพราะจะนำ ReactJS มาใช้ในงาน Dashboard IoT

3.5.3 ออกแบบ Interface เพื่อนำมาสร้างจริงโดยใช้ตัวอย่างจากข้างต้น

1. ออกแบบหน้าต่างหลักหรือหน้าโฮมเพจ

2. ออกแบบแทบเมนูในการใช้งาน

3. ออกแบบหน้าการทำงานหลัก

4. ออกแบบวิดเจ็ตที่ไว้สำหรับแสดงค่าต่างๆของเซ็นเซอร์

3.5.4 เริ่มทำการเขียนโค้ดในส่วนของหน้า front end

โดยเริ่มจากการเขียนส่วน front end ที่ออกแบบไว้ข้างต้นโดยใช้ภาษา ReacJS ก่อนจะเริ่มก็ต้องลง webpack ก่อนโดยใช้คำสั่งดังนี้

$ create react-app name-project

จากนั้นเริ่มเขียนโค้ดตามที่ออกแบบไว้เลย

3.5.5นำส่วนของ API มาเชื่อมต่อกับหน้าต่างแดชบอร์ดเพื่อให้สามารถเชื่อมต่อกับฐานข้อมูลได้

นำส่วนของระบบทีี่เขียนไว้ไปเชื่อมต่อกับระบบ API ที่ทำไว้โดยใช้ Library axios ในการเรียกใช้งาน วิธิติดตั้ง axios สามารถใช้คำสั่งดังนี้

$ npm install axios

3.5.6 เชื่อมต่อการทำงานของ NETPIE เข้ากับหน้าต่างแดชบอร์ด

นำระบบ Datsource ที่เขียนไว้มาเชื่อมต่อกับ NETPIE โดยใช้ Library Microgear ในการจะเรียกใช้งานต้องทำการติดตั้งก่อน ติดตั้งได้โดยคำสั่ง

$ npm install microgear

เริ่มเขียนโค้ดเชื่อมต่อการทำงานต่างๆได้เลย

3.5.7 ทดลองใช้งานระบบที่ทำสำเร็จแล้ว

3.5.8 ทำ Documentation สำหรับสอนการใช้งานให้กับผู้ใช้

**บทที่ 4**

**ผลการดำเนินงาน**

4.1 ผลการออกแบบ data format ของเซ็นเซอร์ทั้ง 4 ตัว

ทั้ง 4 ตัวจะออกแบบในโครงสร้างเหมือนกันแต่ในส่วนของค่าที่ได้จากเซ็นเซอร์จะต่างกันไปตามค่าที่ได้จากเซนเซอร์

4.1.1อธิบายโครงสร้างที่ออกแบบ data format ของเซ็นเซอร์

\* W3C WOT : W3C

\* Microsoft IoT Device : Microsoft

\* Device Shadow for AWS IoT : AWS

\* Google : Google Cloud IoT

\* Terbine

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| แอททริบิวต์ | ความหมาย | แหล่งอ้างอิง | | | | |
| W3C | Microsoft | AWS | Terbine | Google |
| **sensorId** | เป็น Field ที่ใช้ในการเก็บข้อมูล Id หรือ Primary Key ของเซ็นเซอร์เพื่อบ่งบอกว่าคือเซ็นเซอร์ตัวไหน | \* | \* | \* | \* | \* |
| **desired** | เป็น Field ที่ใช้ในการเก็บข้อมูลต่างๆ ที่ได้จากเซ็นเซอร์และจะมีการเก็บค่า value และ unit ของค่านั้นๆ |  | \* | \* |  |  |
| **status** | เป็น Field ที่ใช้ในการเก็บค่าสถานะของเซ็นเซอร์ว่าเปิดหรือปิดใช้งานอยู่ | \* | \* | \* | \* | \* |
| **timestamp** | เป็น Field ที่ใช้บอกเวลาล่าสุดที่มีการอัพเดท ค่าต่างๆของเซ็นเซอร์ | \* | \* | \* | \* | \* |
| **metada** | เป็น Field ที่ใช้บอกข้อมูล metadata ทั้งหมดของเซ็นเซอร์และมี Field  ย่อยในการบอกรายละเอียดต่างๆ | \* |  | \* |  | \* |
| **thingId** | เป็น Field ที่ใช้บอกข้อมูลเกี่ยวกับId ของอุปกรณ์ว่าอยู่ที่อุปกรณ์ไหน | \* | \* |  | \* |  |
| **name** | ไว้บอกชื่อว่าของเซ็นเซอร์ว่าชื่ออะไร | \* |  | \* |  |  |
| **description** | ไว้บอกรายละเอียดต่างๆและลักษณะการทำงานของเซ็นเซอร์ | \* |  | \* |  |  |
| **language** | ไว้บอกโค้ดภาษาที่ใช้ในการเก็บข้อมูลว่าเป็นภาษาอะไร |  |  |  | \* |  |
| **sensorInfo.type** | ไว้บอกชนิดของเซ็นเซอร์ว่าเป็นชนิดอะไรทำงานอะไร | \* | \* |  | \* |  |
| **sensorInfo.model** | ไว้บอกโมเดลหรือบอกรุ่นของเซ็นเซอร์ | \* |  |  |  | \* |
| **categories** | บอกประเภทการทำงานของเซ็นเซอร์ว่าเกี่ยวกับอะไร |  | \* |  |  |  |
| **location.type** | เป็น Field ที่ใช้บอกชนิดของ location |  |  |  | \* | \* |
| **freetext** | ไว้บอกข้อมูลเพิ่มเติมต่างๆ เกี่ยวกับที่อยู่ไว้ข้อมูลเพิ่มเติม |  |  |  |  | \* |
| **latitude** | ไว้บอกข้อมูล latitude (ถ้ามี) |  |  |  | \* |  |
| **longitude** | ไว้บอกข้อมูล longitude (ถ้ามี) |  |  |  | \* |  |
| **version** | ไว้บอกเวอร์ชันการเก็บข้อมูลของเซ็นเซอร์ | \* |  |  |  | \* |

4.1.2 ผลลัพธ์การออกแบบของเซ็นเซอร์

รูปแบบ Schema mongoDB model ในรูปแบบ JSON

{

**sensorId**: { type: String, required: true },

**desired**: { type: Object, required: true },

**status**: { type: String, default: 'enabled' },

**timestamp**: { type: Date, default: Date.now },

**metadata**: {

**thingId**: { type: String, required: true },

**name**: { type: String, required: true},

**description**: { type: String },

**language**: { type: String, default: 'en-US' },

**sensorInfo**: {

**type**: { type: String, required: true },

**model**: { type: String, required: true }

},

**categories**: [{ type: Object }],

**location**: {

**type**: { type: String, default: 'Fixed' },

**address**: { type: String },

**city**: { type: String },

**county**: { type: String },

**country**: { type: String },

**postalCode**: { type: String },

**freeText**: { type: String },

**gps**: {

**latitude**: { type: Number },

**longitude**: { type: Number }

}

},

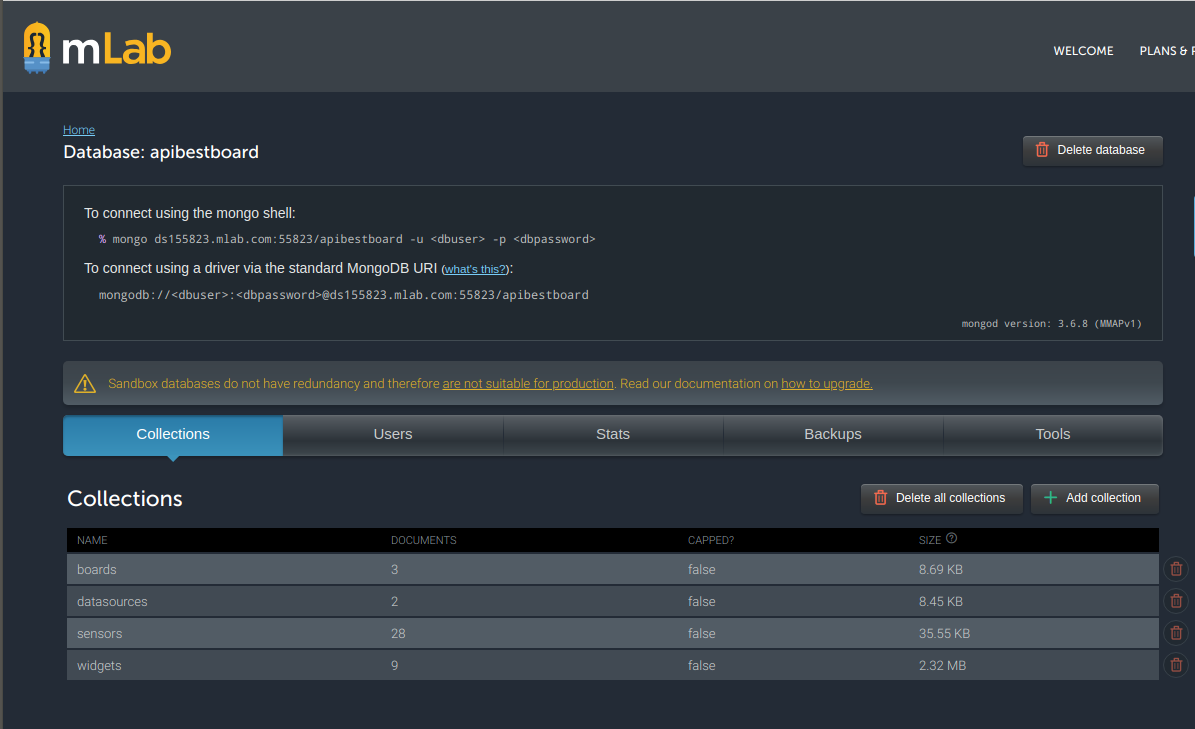
**version**: { type: String, default: '0.1' }

}

}

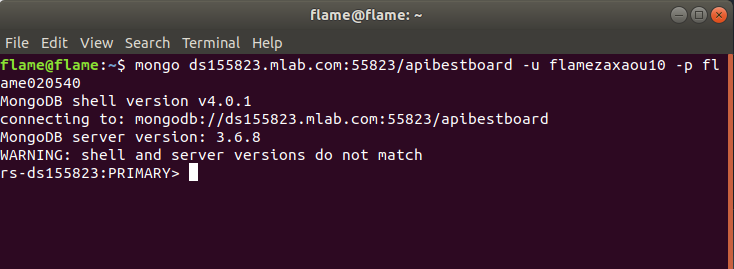
4.2 ผลลัพธ์ของฐานข้อมูล mongoDB

4.2.1 ใช้ mLab mongoDB ในการเก็บฐานข้อมูลเพื่อจะให้ฐานข้อมูลออนไลน์ตลอดเวลา

  
ภาพที่ 4 - 1 : ภาพแสดงฐานข้อมูลใน mongoDB

คำสั่งใช้งานเบื้องต้นของ mongoDB

$ mongo ds155823.mlab.com:55823/apibestboard -u <dbuser> -p <dbpassword>

  
ภาพที่ 4 - 2 : ภาพแสดงการเชื่อมต่อเข้ากับฐานข้อมูล

ฐานข้อมูล mongoDB สามารถทำงานได้แล้ว

4.3 ผลลัพธ์การศึกษาการรับส่งข้อมูลของอุปกรณ์ IoT ผ่านคลาวด์แพลตฟอร์ม NETPIE

4.3.1 สร้าง NETPIE Application เพื่อมาใช้งานเชื่อมต่อกับอุปกรนณ์ IoT

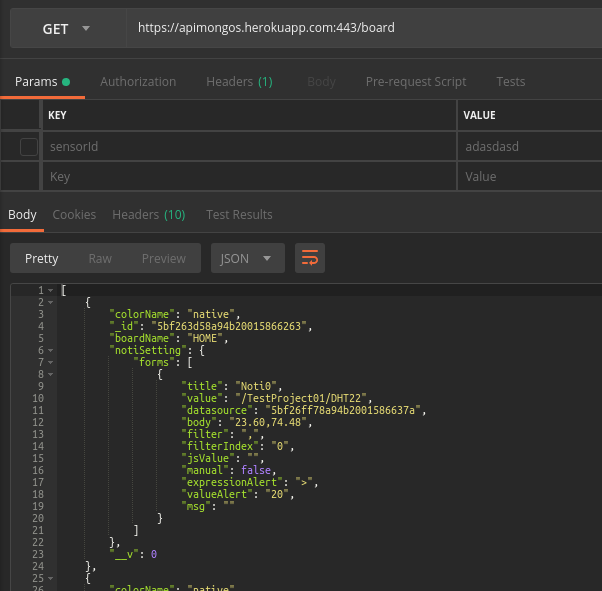
4.3.2 สร้าง NETPIE Freeboard เพื่อนำค่าที่ส่งมาจากเซ็นเซอร์ต่างๆมาแสดง

4.3.3 สร้าง NETPIE Feed เพื่อเขียนข้อมูลลงไปในกราฟเพื่อง่ายต่อการอ่านค่าย้อนหลัง

4.4 ผลลัพธ์ REST API

4.4.1 สามารถบันทึก แก้ไข และลบข้อมูลในฐานข้อมูลผ่าน API ได้ทันที

**API สามารถ GET ค่าได้ตามนี้หรือก็คือการดึงค่ามาแสดงในรูปแบบ JSON**

  
ภาพที่ 4-3 : แสดงการ GET ค่าของ API โดยใช้โปรแกรม POSTMAN

- GET /sensor : เป็นการดึงค่าทั้งหมดที่บันทึกข้อมูลลงไปในฐานข้อมูล

- GET /sensor/:sensorId : เป็นการดึงค่าทั้งหมดตาม sensorId ที่กำหนด

- GET /sensor/:sensorId/lastdata : เป็นการดึงค่าล่าสุดของ sensorId ที่กำหนด

- GET /sensor/:sensorId/day=:day : เป็นการดึงค่าล่าสุดตามช่วงเวลาหน่วยเป็น วันตาม sensorId ที่กำหนด

- GET /sensor/:sensorId/hour=:hour : เป็นการดึงค่าล่าสุดตามช่วงเวลาหน่วย เป็นชั่วโมงตาม sensorId ที่กำหนด

- GET /sensor/:sensorId/minute=:minute : เป็นการดึงค่าตามช่วงเวลาหน่วย เป็นนาทีตาม sensorId ที่กำหนด

- GET /board : เป็นการดึงค่าทั้งหมดของ board หรือpage ในระบบ Dashboard IoT ของผม

- GET /board/:boardId : เป็นการดึงค่า boardตาม boardId ที่กำหนด

- GET /datasource : เป็นการดึงค่าทั้งหมดของ datasource ในระบบ Dashboard IoT

- GET /datasource/:datasourceId : เป็นการดึงค่า datasource ตาม datasourceId ที่กำหนด

- GET /widget : เป็นการดึงค่าทั้งหมดของ widget ในระบบ Dashboard IoT

- GET /widget/board/:boardId : เป็นการดึงค่า widget ทั้งหมดที่อยู่ใน boardนั้นตามboardId ที่กำหนด

**API สามารถ POST ค่าได้ตามนี้หรือก็คือการเขียนค่าใหม่ลงไปในฐานข้อมูล**

- POST /sensor : เป็นการเขียนค่าของ sensor ใหม่ลงไปในฐานข้อมูล

- POST /board : เป็นการเขียนค่าของ boardใหม่ลงไปในฐานข้อมูล

- POST /datasource : เป็นการเขียนค่าของ datasource ใหม่ลงไปในฐานข้อมูล

- POST /widget : เป็นการเขียนค่าของ widget ใหม่ลงไปในฐานข้อมูล

**API สามารถ PUT ค่าได้ตามนี้หรือก็คือการแก้ไขข้อมูลนในฐานข้อมูล**

- PUT /sensor/:sensorId : เป็นการแก้ไขข้อมูลในฐานข้อมูลของ sensorตาม sensorId ที่กำหนด

- PUT /board/:boardId : เป็นการแก้ไขข้อมูลในฐานข้อมูลของ board ตาม boardId ที่กำหนด

- PUT /datasource/:datasourceId : เป็นการแก้ไขข้อมูลในฐานข้อมูลของ datasource ตาม datasourceId ที่กำหนด

- PUT /widget/:widgetId : เป็นการแก้ไขข้อมูลในฐานข้อมูลของ widget ตาม widgetId ที่กำหนด

**API ที่สามารถ DELETE ค่าได้ตามนี้หรือคือการลบข้อมูลในฐานข้อมูล**

- DELETE /sensor/:sensorId : เป็การลบค่าในฐานข้อมูลของ sensor ตาม sensorId ที่กำหนด

- DELETE /board/:boardId : เป็การลบค่าในฐานข้อมูลของ board ตาม boardId ที่กำหนด

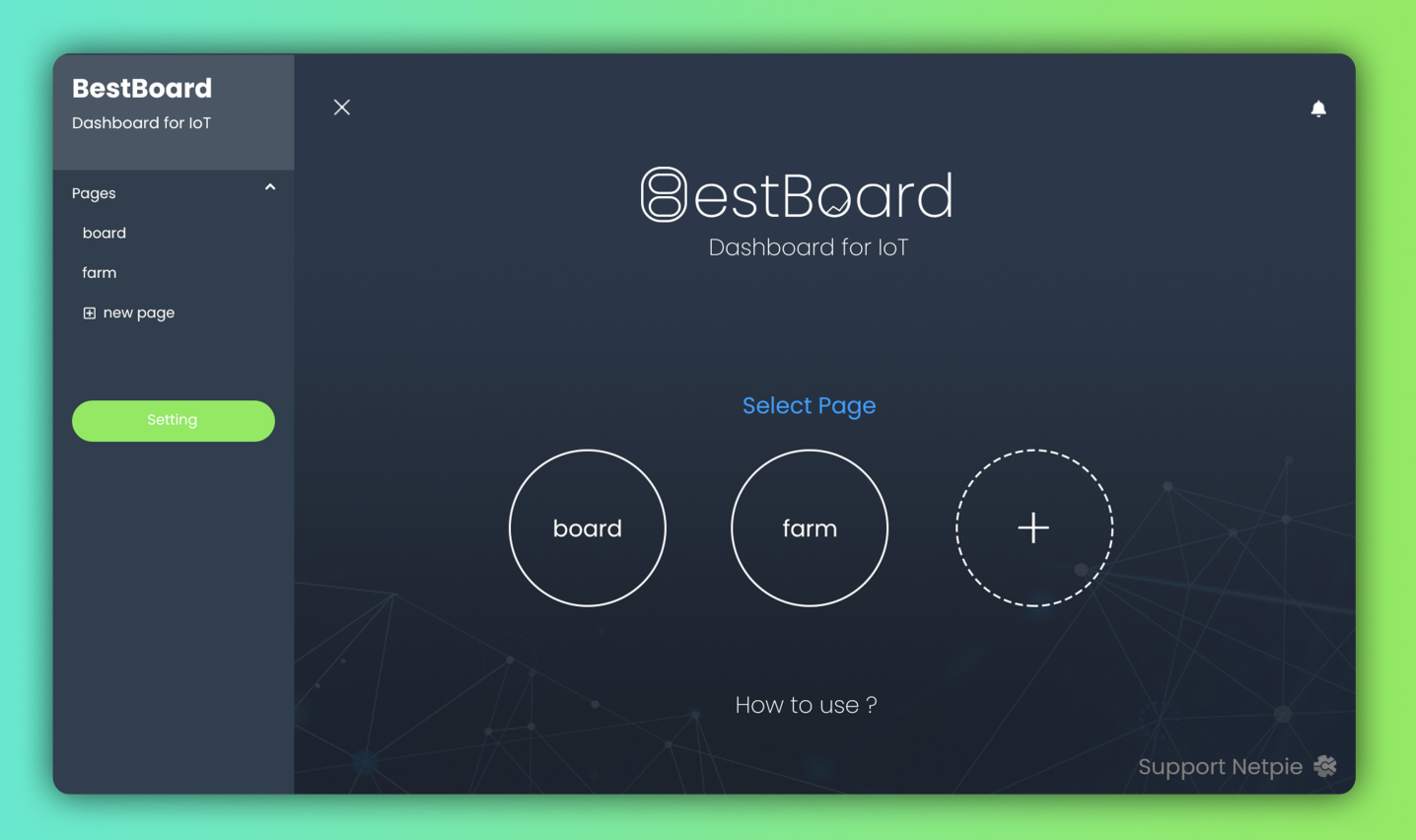
- DELETE /datasource/:sensorId : เป็การลบค่าในฐานข้อมูลของ datasource ตามdatasourceId ที่กำหนด

- DELETE /widget/: widgetId : เป็การลบค่าในฐานข้อมูลของ widget ตาม widgetId ที่กำหนด

4.5 ผลลัพธ์ของ Interface Dashboard IoT

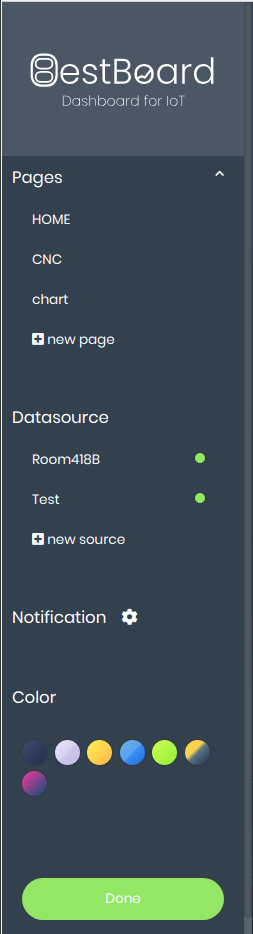
4.5.1 ผลลัพธ์การออกแบบหน้าหลัก

โดยหน้าหลักจะประกอบไปด้วยเพจเพื่อง่ายต่อการเข้าถึงและมีเมนูทางด้านซ้ายไว้สำหรับการทำงานต่างๆ

  
ภาพที่ 4 - 4 : ภาพแสดงการออกแบบหน้าหลัก

4.5.2 ผลลัพธ์การออกแบบหน้าเมนูและระบบ

ระบบ Page ที่ทำหน้าที่ไว้แยกหน้าเป็นหน้าในแต่ละงาน สามารถเพิ่มได้หลาย หน้า

**  
ภาพที่ 4 - 6 : แสดงเมนู sidebar และระบบต่างๆ

ระบบ Datasource สามารถเพิ่มลบและแก้ไข Datasource ตามต้องการ Datasource ในที่นีี้หมายถึงแหล่งข้อมูลที่เราจะเอามาแสดงใน Dashboard โดยโครงงานจะ เป็นการเชื่อมต่อกับ NETPIE

ระบบ Notifaication ไว้สำหรับแจ้งเตือนตามเงื่อนไขที่เรากำหนด

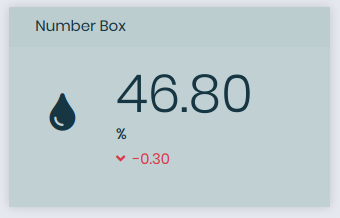
ระบบ Color หรือธีม ไว้สำหรับเปลี่ยนธีมตามที่เราต้องการ

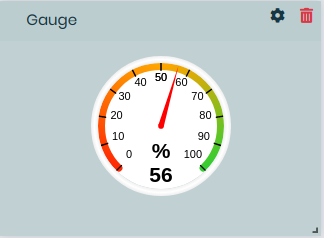
4.5.3 ผลลัพธ์การออกแบบวิดเจ็ตเพื่อแสดงค่าต่างๆที่ถูกส่งมายังระบบ

วิดเจ็ตต่างๆ ไว้สำหรับการทำงานต่างๆ บนแดชบอร์ดแบ่งเป็นสามประเภทได้แก่

ประเภทแสดงผล ประเภทควมคุมการทำงาน และประเภทอื่นๆ ไว้ใช้ให้เหมาะกับงาน

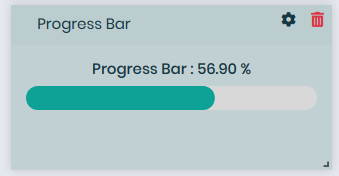
Number Box ไว้สำหรับแสดงผลค่าตัวเลขต่างๆที่ ได้มาจาก sensor

  
ภาพที่ 4 - 7 : Number Box Widget

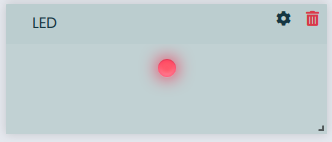
  
ภาพที่ 4 - 8 : Gauge Widget

Gauge : ไว้สำหรับแสดงค่าตัวเลขต่างๆ ในรูปแบบ Gauge

Progress Bar : ไว้สำหรับแสดงค่าเป็นหลอดโหลดเพื่อง่านต่อการใช้งาน

  
ภาพที่ 4 - 9 : Progress Bar

LED : เป็นการเอาค่ามาเช็คว่าไฟติดหรือไม่ตามที่ต้องการ

ภาพที่ 4 - 10 : LED Widget