

2102447 ปฏิบัติการวิศวกรรมอิเล็กทรอนิกส์

Electronic Engineering Laboratory

ผู้สอนประจำวิชา ผศ.ดร.สุริย์ พุ่มรินทร์ และอ.ดร.ณพงศ์ ปณิธานธรรม

ผู้สอนปฏิบัติการ ภัทร เกษมสำราญ (ภาคการศึกษาปลาย 2567)



ภาควิชาวิศวกรรมไฟฟ้า
คณะวิศวกรรมศาสตร์

ก่อตั้ง ๒๔๗๒

จุฬาลงกรณ์มหาวิทยาลัย

การเก็บและนำเสนอภาพข้อมูล IoT ด้วย NETPIE

(IoT Data Collection and Visualization via NETPIE)

วัสดุและอุปกรณ์

1. คอมพิวเตอร์ติดตั้งแอปพลิเคชัน Arduino
2. บอร์ด IOXESP32+ และบอร์ดขยายขา
3. โมดูลจอ OLED 0.96" 128x64 pixel
4. โมดูลวัดอุณหภูมิ ความชื้น และความดัน BME280
5. สาย USB, สายไฟ และอุปกรณ์อื่นๆ ที่เกี่ยวข้อง



"IoT Learning Kit" 2102447 Electronics Engineering Laboratory

ELECTRICAL
ENGINEERING
CHULALONGKORN UNIVERSITY



IOXESP32+ Detail




ESP32 Repository

Components list :

- | | |
|-------------------------------------|---------|
| 1.) IOXESP32+ V1.0 Embedded Board | 1 each |
| 2.) BASE32 Pinout Expansion Board | 1 each |
| 3.) OLED Display 128x64 0.96" | 1 each |
| 4.) OLED Housing Bracket | 1 each |
| 5.) Breadboard / Protoboard | 1 each |
| 6.) 4 SPST Switch Module | 1 each |
| 7.) 8mm LED (R/Y/G) Module | 1 each |
| 8.) Potentiometer Module | 1 each |
| 9.) USB-C to USB-A Cable | 1 each |
| 10.) Jumper Wire (Male to Male) | 10 each |
| 11.) Jumper Wire (Male to Female) | 10 each |
| 12.) Jumper Wire (Female to Female) | 10 each |

Sensor selection list :

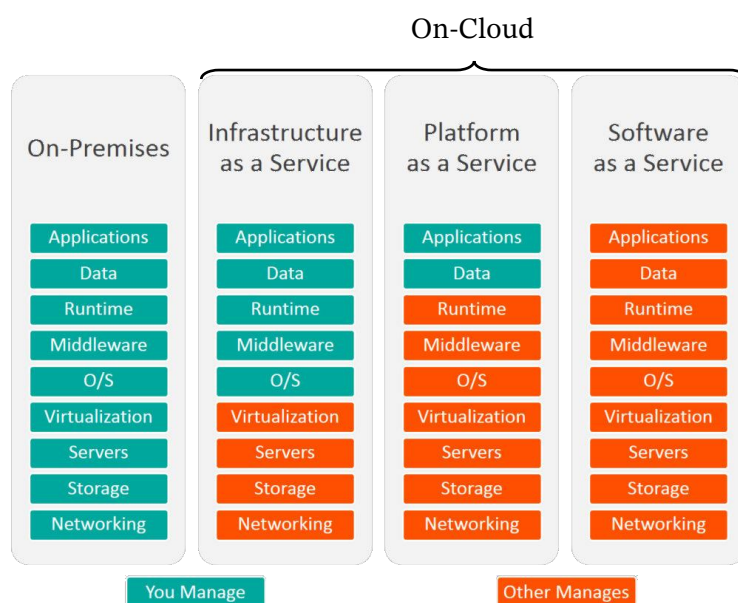
- | | |
|---|---|
| 
TSL2561 Lux sensor | 
BME280 (Temp, Humid, Pressure) |
| 
PIR Motion Sensor | 
DS18B20 with Module |



หมายเหตุ : ในกล่องไม่มีรายการที่ 5.) Breadboard / Protoboard ให้

1. บทนำ

การประมวลผลแบบกลุ่มเมฆ (cloud computing) เกิดขึ้นจากแบบจำลองธุรกิจ (business model) เพื่อการใช้ทรัพยากรอันมีอยู่อย่างจำกัดมาใช้งานร่วมกันให้เกิดประโยชน์สูงสุด โดยผู้ใช้งานไม่จำเป็นต้องเป็นเจ้าของฮาร์ดแวร์ แต่จ่ายค่าบริการให้กับผู้ให้บริการเพื่อใช้บริการทรัพยากรในรูปแบบคล้ายกับการเช่า (subscription) เพื่อใช้งานทรัพยากรหรือซอฟต์แวร์ที่ต้องการ [Gollmann 2005.] โดยผู้ใช้งานนั้นไม่จำเป็นต้องมีความรู้ในเชิงเทคนิค รูปแบบการให้บริการอาจแบ่งตามขอบเขตความรับผิดชอบของผู้ให้บริการดังรูปที่ 1-1



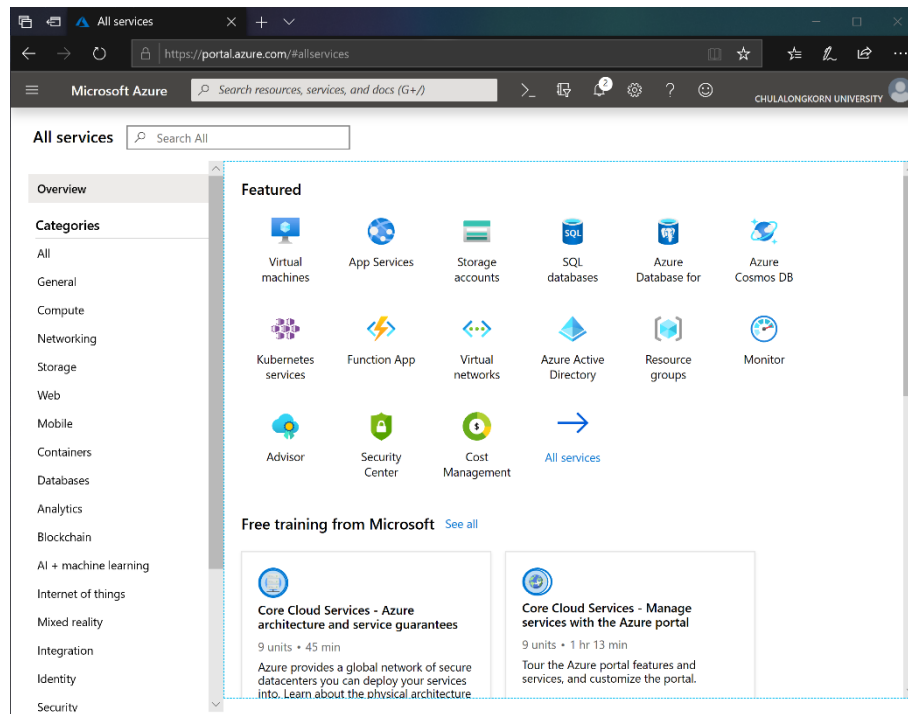
รูปที่ 1-1 ขอบเขตความรับผิดชอบในการบริหารจัดการและการซ่อมบำรุง

1.1 Software-as-a-Service

Software-as-a-Service (SaaS) คือการให้บริการซอฟต์แวร์สำเร็จรูปพร้อมใช้งานได้ทันทีโดยไม่จำเป็นต้องปรับแต่งใด ๆ ข้อดีคือง่ายสำหรับผู้ที่ไม่มีความรู้ทางเทคนิค ข้อจำกัดคือไม่สามารถปรับแต่งหรือพัฒนาส่วนต่อขยายเพื่อเพิ่มฟังก์ชันการทำงานตามที่ต้องการด้วยตัวเองได้ หรือทำได้อย่างจำกัด

1.2 Platform-as-a-Service

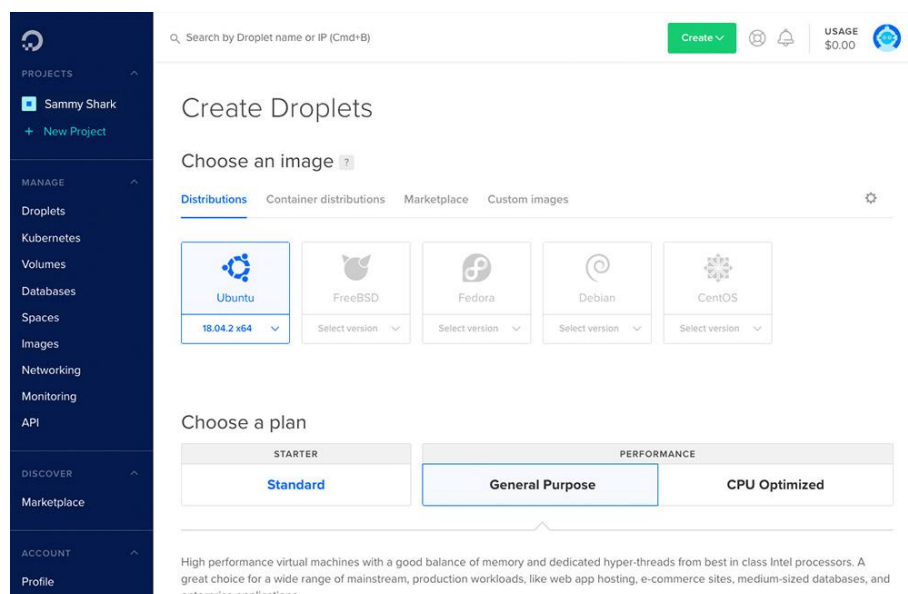
Platform-as-a-Service (PaaS) คือการให้บริการฮาร์ดแวร์และการปรับแต่งที่พร้อมสำหรับการพัฒนาแอปพลิเคชัน ผู้ให้บริการจะติดตั้งคอมไพเลอร์ (compiler) อินเทอร์พรีเตอร์ (interpreter) ระบบจัดการฐานข้อมูล (database management system) เว็บเซิร์ฟเวอร์ (web server) รวมไปถึงแอปพลิเคชันอื่น ๆ ที่จำเป็น เหมาะสมสำหรับนักพัฒนาซอฟต์แวร์เพื่อทำงานเฉพาะอย่าง ข้อดีคือนักพัฒนาไม่จำเป็นต้องเตรียมสภาพแวดล้อมที่เหมาะสมสำหรับการพัฒนาแอปพลิเคชันเอง ตัวอย่าง Microsoft Azure Portal แสดงบริการแบบ PaaS (และ IaaS) ดังรูปที่ 1-2



รูปที่ 1-2 หน้าหลักของ Microsoft Azure Portal

1.3 Infrastructure-as-a-Service

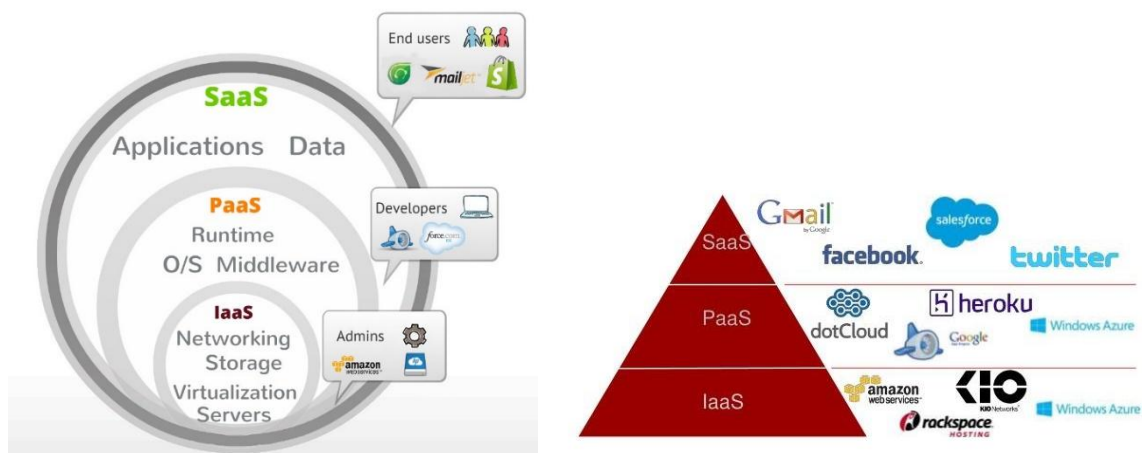
Infrastructure-as-a-Service (IaaS) คือ การเช่าเซิร์ฟเวอร์เสมือน (virtual server) โดยผู้ให้บริการจะแบ่งทรัพยากรฮาร์ดแวร์และเครือข่ายให้ผู้ให้บริการเสมือนได้เซิร์ฟเวอร์ส่วนตัวสำหรับใช้งาน ผู้ใช้สามารถติดตั้งซอฟต์แวร์หรือปรับแต่งสภาพแวดล้อมที่เหมาะสมได้เองดังตัวอย่างในรูปที่ 1-3 การเปรียบเทียบการให้บริการระหว่าง SaaS, PaaS และ IaaS สามารถแสดงได้ดังตารางที่ 1.1 และตัวอย่างผู้ให้บริการบางส่วนดังรูปที่ 1-4 และ 1-5



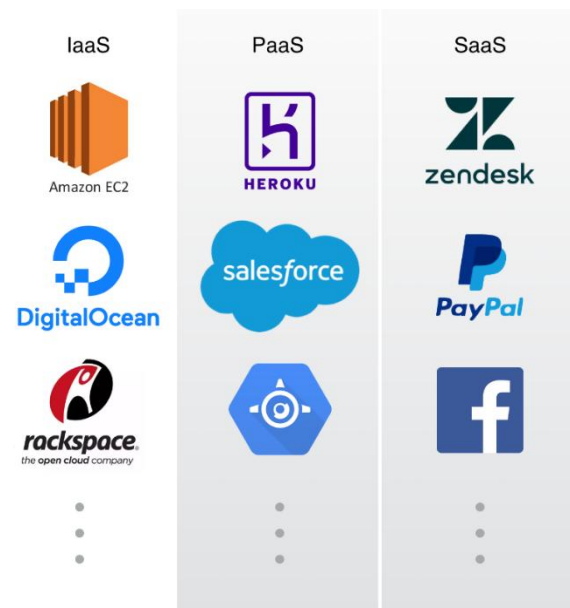
รูปที่ 1-3 หน้าจอหลักของ Digital Ocean ส่วนของผู้ใช้สำหรับจัดการบริการ IaaS

ตารางที่ 1.1 การเปรียบเทียบการให้บริการระหว่าง SaaS, PaaS และ IaaS

Features	IaaS	PaaS	SaaS
What you get	You get the infrastructure & pay accordingly. Freedom to use or install any OS, software or composition	Here you get what you demand. Software, hardware, OS, web environment. You get the platform to use & pay accordingly	Here you don't have to worry about anything. A pre-installed, pre-configured package as per your requirement is given and you only need to pay accordingly.
Importance	The basic layer of Computing	Top of IaaS	It is like a Complete package of services
Technical Difficulties	Technical knowledge required	You get the Basic setup but still the knowledge of subject is required.	No need to worry about technicalities. The SaaS provider company handles everything.
Deals with	Virtual Machines, Storage (Hard Disks), Servers, Network, Load Balancers etc	Runtimes (like java runtimes), Databases (like mySql, Oracle), Web Servers (tomcat etc)	Applications like email (Gmail, Yahoo mail etc), Social Networking sites (Facebook etc)
Popularity Graph	Popular among highly skilled developers, researchers who require custom configuration as per their requirement or field of research.	Most popular among developers as they can focus on the development of their apps or scripts. They don't have to worry about traffic load or server management etc.	Most popular among normal consumers or companies which rely on softwares such as email, file sharing, social networking as they don't have to worry about the technicalities.



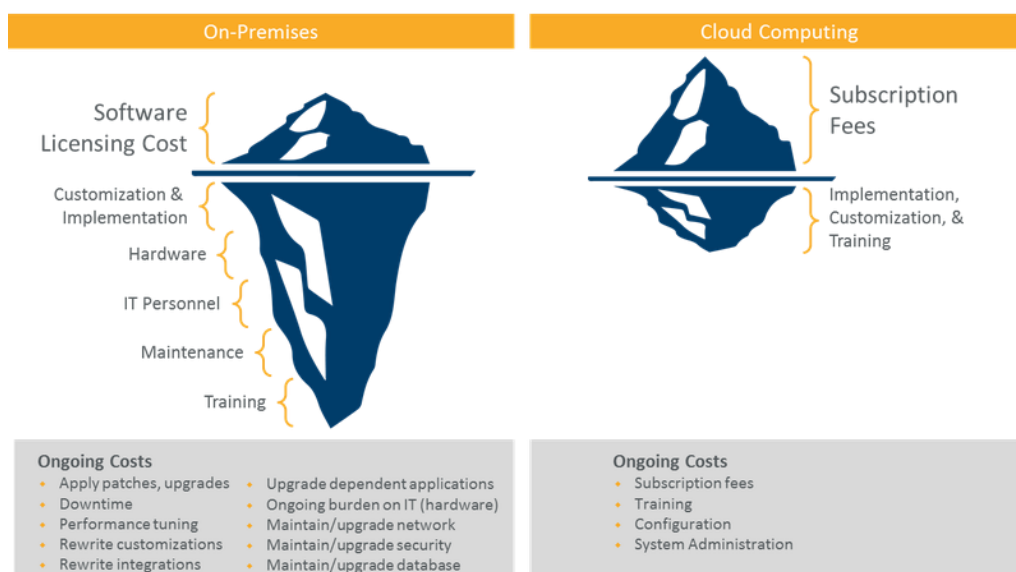
รูปที่ 1-4 ตัวอย่างผู้ให้บริการ SaaS, PaaS และ IaaS



รูปที่ 1-5 ตัวอย่างผู้ให้บริการ SaaS, PaaS และ IaaS

1.4 เปรียบเทียบรายการค่าใช้จ่ายระหว่าง On-Cloud และ On-Premises

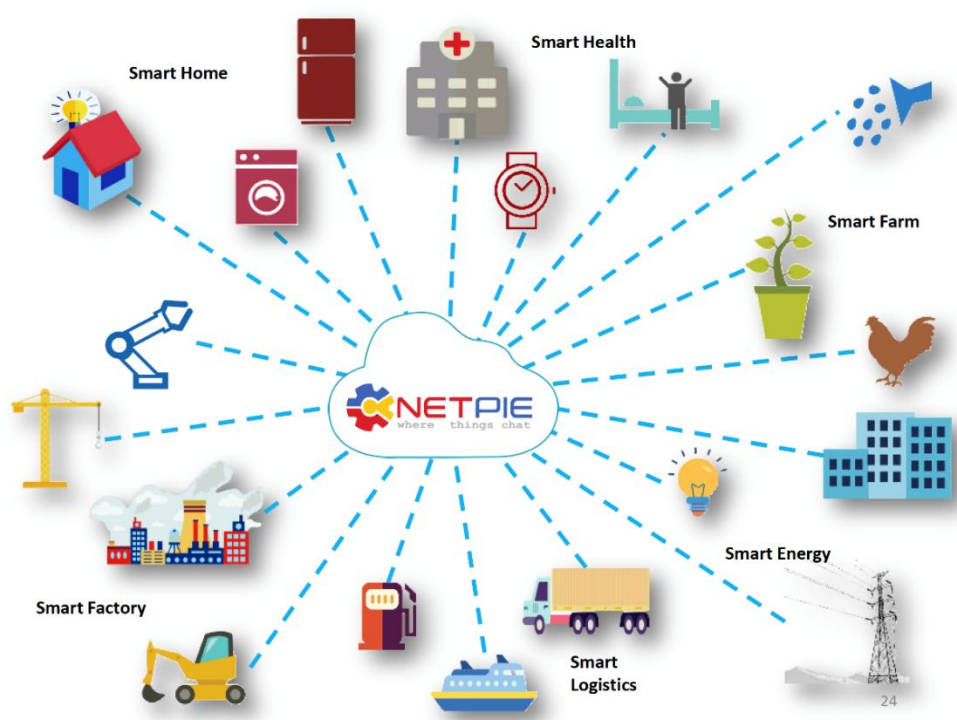
On-Premise คือ ระบบเซิร์ฟเวอร์ที่แบบดั้งเดิม (traditional) ที่ผู้ใช้จัดซื้อ ติดตั้ง บำรุงรักษา และอัปเดตเอง อยู่เสมอ ผู้ใช้อาจว่าจ้างผู้ดูแลด้วยสัญญาบริการดูแลและบำรุงรักษา (Maintenance Service Agreement: MA) ซึ่งมีอายุประมาณ 3-5 ปี และเมื่อครบกำหนด หากธุรกิจของผู้ใช้งานเติบโต ระบบของผู้ใช้งานมีปริมาณงานมากขึ้นหรือต้องการใช้ทรัพยากรมากขึ้น ผู้ใช้มักต้องจัดซื้อฮาร์ดแวร์ใหม่ที่มีประสิทธิภาพมากขึ้นกว่าเดิม การใช้บริการระบบเซิร์ฟเวอร์แบบ On-Cloud เป็นการผลักภาระการบำรุงรักษาไปยังผู้ให้บริการแทนแลกกับการจ่ายค่าบริการ รายการค่าใช้จ่ายระหว่างระบบเซิร์ฟเวอร์ทั้งสองรูปแบบแสดงดังรูปที่ 1-6



รูปที่ 1-6 เปรียบเทียบรายการค่าใช้จ่ายระหว่าง On-Premise กับ On-Cloud

2. NETPIE

NETPIE (Network Platform for Internet of Everything) อ่านว่า เน็ต-พาย เป็นชื่อเครื่องหมายการค้า อินเทอร์เน็ตของสรรพสิ่งที่ให้บริการโพรโทคอล MQTT แบบกระจาย (distributed) ที่สามารถเชื่อมต่ออุปกรณ์ ฮาร์ดแวร์ และแอปพลิเคชันที่เขียนโดยโปรแกรมภาษาใดๆ ได้ เพื่อรองรับการพัฒนานวัตกรรมอัจฉริยะด้านต่าง ๆ ดังรูปที่ 2-1 ในช่วงต้น NETPIE ถูกวิจัยและพัฒนาโดยศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ (NECTEC) ต่อมาได้ถูกถ่ายทอดเทคโนโลยีด้วยการอนุญาตให้ใช้สิทธิ (licensing) แก่บริษัท เน็กซ์พาย จำกัด (NEXPIE) ไปพัฒนาต่อยอดและให้บริการเชิงพาณิชย์ ในขณะเดียวกัน NETPIE ยังคงให้บริการใช้ฟรีสำหรับกลุ่มเมกเกอร์ นักเรียน นักศึกษา นักพัฒนา และอุตสาหกรรม SME ต่อไปภายใต้การสนับสนุนจากคณะกรรมการกิจการกระจายเสียง กิจการโทรทัศน์และกิจการโทรคมนาคมแห่งชาติ หรือ กสทช.



รูปที่ 2-1 การเชื่อมต่ออุปกรณ์ฮาร์ดแวร์ และแอปพลิเคชันต่าง ๆ ด้วย NETPIE

2.1 บัญชีผู้ใช้ NETPIE

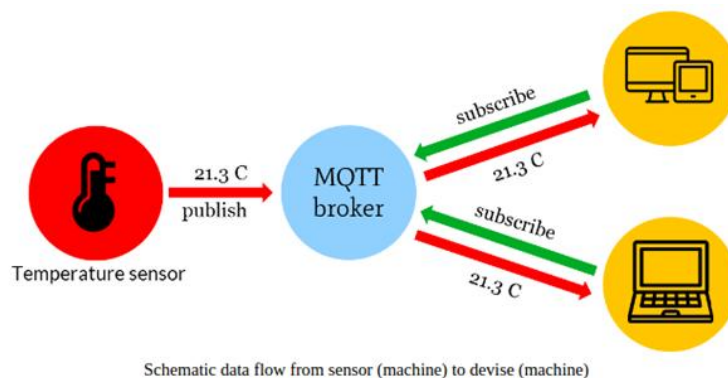
บัญชีผู้ใช้แบบฟรีโควตามีข้อจำกัด ดังนี้

Project	3	Projects
Connected devices	10	Devices
Real-time messages	9,000,000	Messages/month
Time-series data storage	1,000,000	Point-month
Shadow read/write	500,000	Operations/month
API call	800,000	Operations/month
Dashboard	3	Freeboards/project
Freeboard connection	3	Concurrent views

Trigger and action	5,000	Operation/month
--------------------	-------	-----------------

2.2 โพรโทคอล MQTT

MQTT (Message Queuing Telemetry Transport) เป็นโพรโทคอลมีสถาปัตยกรรมแบบแบบไคลเอนต์/เซิร์ฟเวอร์ (client/server) บนชั้นประยุกต์ใช้งานหรือชั้นบนสุดในแบบจำลอง OSI ไคลเอนต์สามารถ Publish ข้อมูลไปยังโบรกเกอร์ด้วยการ Push และสามารถ Subscribe เพื่อรับข้อมูลจากโบรกเกอร์ด้วยการ Request ดังแผนภาพรูปที่ 2-2

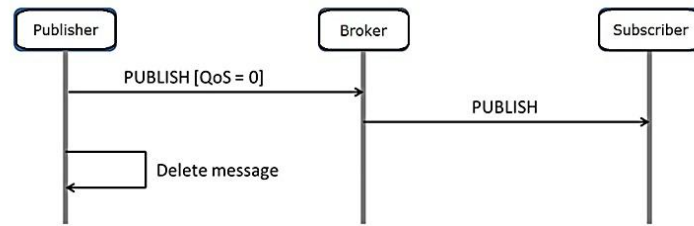


รูปที่ 2-2 แผนภาพแสดงฟังก์ชันการรับ/ส่งของ MQTT ไคลเอนต์กับโบรกเกอร์ (เซิร์ฟเวอร์)

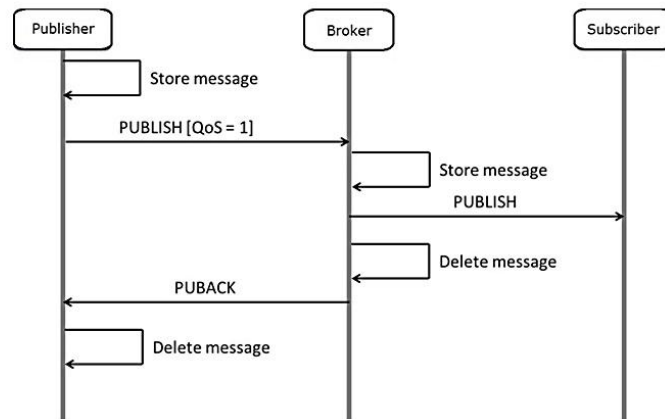
เมื่อเปรียบเทียบ MQTT กับ HTTP (REST) จะพบว่า MQTT มีความได้เปรียบที่โบรกเกอร์สามารถ Push ข้อความ (message) ไปยังไคลเอนต์ได้ตามเหตุการณ์ (event-driven) ในขณะที่เมื่อใช้ HTTP ไคลเอนต์ต้อง Poll ข้อมูลเป็นระยะ ๆ ซึ่งแต่ละครั้งต้องมีการสร้างการเชื่อมต่อขึ้นใหม่และอาจจะไม่มีข้อมูลใหม่ใด ๆ ให้อัปเดต หากต้องการให้ระบบทำงานแบบเวลาจริงย่อมหมายถึงต้องตั้งค่าเวลาการโพลให้สั้น เกิด Overhead ในการรับ/ส่งข้อมูล แต่ครั้งหนึ่งทำให้การใช้งานแบนด์วิดท์มีประสิทธิภาพลดลง จึงเป็นข้อได้เปรียบของโพรโทคอล MQTT เทียบกับ HTTP

2.3 การรับประกันคุณภาพของโพรโทคอล MQTT

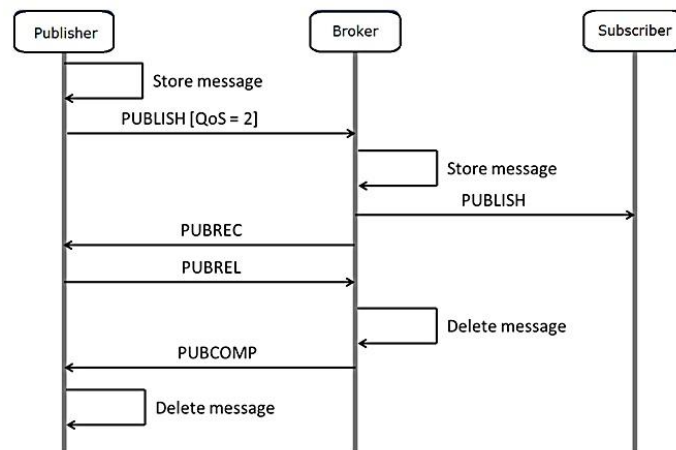
การรับประกันคุณภาพ (Quality of Service: QoS) การจัดการช่องทางแบนด์วิดท์ของระบบเครือข่าย สำหรับโพรโทคอล MQTT ไคลเอนต์จะเป็นผู้กำหนดระดับของบริการส่งและรับข้อความหรือ QoS ที่ต้องการในแต่ละหัวข้อ (topic) ที่ Publish หรือ Subscribe และโบรกเกอร์จะตอบสนองด้วย QoS ระดับเดียวกันสำหรับหัวข้อนั้น ๆ QoS ใน MQTT แบ่งได้เป็น 3 ระดับ คือ อย่างมากหนึ่งครั้ง (At Most Once: QoS 0), อย่างน้อยหนึ่งครั้ง (At Least Once: QoS 1) และ หนึ่งครั้งเท่านั้น (Exactly Once: QoS 2) แสดงการรับ/ส่งข้อมูลได้ดังรูปที่ 2-3, 2-4 และ 2-5 ซึ่งสามารถเรียงการใช้แบนด์วิดท์จากน้อยไปมากได้แก่ QoS 0, QoS 1 และ QoS 2 ตามลำดับ



รูปที่ 2-3 แผนผังการสื่อสารเพื่อ Publish ข้อความด้วย QoS 0



รูปที่ 2-4 แผนผังการสื่อสารเพื่อ Publish ข้อความด้วย QoS 1



รูปที่ 2-5 แผนผังการสื่อสารเพื่อ Publish ข้อความด้วย QoS 2

2.4 Device บน NETPIE (<https://docs.netpie.io/device-config.html>)

ไคลเอนต์หรือ Device บน NETPIE มีโครงสร้างแบบ (configuration) อยู่ 4 ส่วน คือ Shadow, Schema, Trigger and Event Hook, และ Feed

Shadow คือ ฐานข้อมูลเสมือนของ Device เป็นฐานข้อมูลเล็ก ๆ ที่มีคู่อยู่ Device ทุกตัว ใช้สำหรับเก็บข้อมูลต่าง ๆ เกี่ยวกับอุปกรณ์นั้น ๆ (shadow data) เช่น ข้อมูลที่เกิดจากเซนเซอร์ ข้อมูลการกำหนดองค์ประกอบต่าง ๆ เป็นต้น

Schema คือ แผงผังข้อมูลที่กำหนดไว้เพื่อใช้กำกับ Shadow สำหรับ Device ที่ต้องมีการจัดการข้อมูลควรสร้าง Schema ของข้อมูลเตรียมไว้ เสมือนเป็น Template ทำให้เซิร์ฟเวอร์สามารถตรวจสอบชนิดข้อมูล (data validation) แปลงข้อมูล (data transformation) เช่น เปลี่ยนหน่วยของข้อมูล เป็นต้น และ เก็บข้อมูลลงในฐานข้อมูลอนุกรมเวลา (timeseries database)

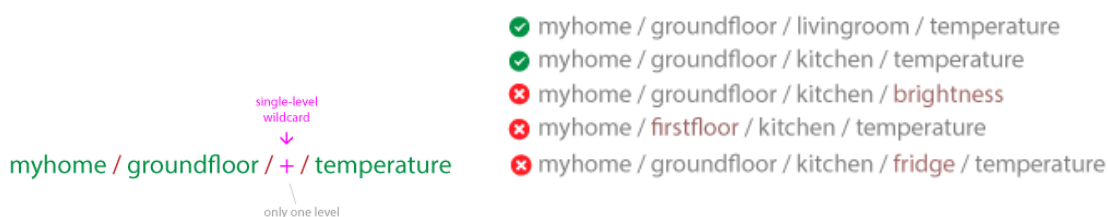
Trigger and Event Hook เป็นระบบที่ผูกการเปลี่ยนแปลงข้อมูลของ Shadow ตามเงื่อนไขที่ถูกตั้งค่าไว้ (trigger) เข้ากับการกระทำภายนอก (event hook) เช่น การตั้งค่าแจ้งเตือนตามสถานะต่าง ๆ

Feed เป็นระบบจัดการและดูข้อมูลใน Timeseries Data เบื้องต้นของแต่ละ Device ซึ่งจะแสดงในรูปแบบของกราฟเส้น แยกตามฟิลด์ที่กำหนดใน Schema และยังสามารถดาวน์โหลดข้อมูลออกมาเป็นไฟล์ .csv ได้

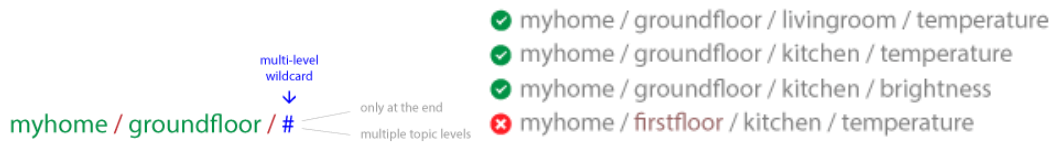
2.4 Topic บน MQTT และบน NETPIE

Topic บน MQTT เป็นข้อมูลประเภท UTF-8 String ซึ่งมีรูปแบบเดียวกับพาท (path) หรือเส้นทางการเข้าถึงข้อมูลของระบบจัดเก็บไฟล์ คือสามารถจัดเป็นลำดับชั้นได้ด้วยการขึ้นด้วย “/” ตัวอย่างเช่น myhome/groundfloor/livingroom/temperature เป็นต้น ชื่อของหัวข้อที่นำไปใช้ได้ต้องขึ้นต้นด้วยตัวอักษรใดก็ได้หนึ่งตัวอักษร ยกเว้น “\$” เนื่องจากเป็นหัวข้อสงวนสำหรับใช้ภายในโบรกเกอร์เท่านั้น

ไคลเอนต์สามารถเลือก Publish หรือ Subscribe เฉพาะ Topic หรือหลาย Topic พร้อมกันโดยใช้ อักขระตัวแทน (wildcard character) โดยที่เครื่องหมาย “+” หมายถึงการระบุแบบหนึ่งระดับ (single-level) ส่วนเครื่องหมาย “#” หมายถึงการระบุแบบหลายระดับ (multi-level) ตัวอย่างเช่น กรณี myhome/groundfloor/+ / temperature หมายถึง Topic ที่ขึ้นต้นด้วย myhome/groundfloor ลำดับชั้นถัดไป 1 ลำดับเป็นอะไรก็ได้ และลำดับชั้นสุดท้ายเป็น temperature ดังรูปที่ 2-6 กรณี myhome/groundfloor/# หมายถึง Topic ที่ขึ้นต้นด้วย myhome/groundfloor ทั้งหมด ดังรูปที่ 2-7



รูปที่ 2-6 การระบุแบบหนึ่งระดับ



รูปที่ 2-7 การระบุแบบหลายระดับ

Topic บน NETPIE มีการกำหนดเพิ่มเติมโดยแบ่งออกเป็น Message Topic และ Shadow Topic

Message Topic ใช้สำหรับ Publish หรือ Subscribe ข้อมูลเพื่อสื่อสารระหว่าง Device ที่อยู่ภายใน Group เดียวกัน Topic จะขึ้นต้นด้วย @msg เป็นลำดับขั้นแรก

Shadow Topic ใช้สำหรับจัดการ Shadow Data ของ Device มี Topic การ Publish และ Subscribe ดังนี้

Publish Topic	
@shadow/data/update	อัปเดตค่าใน Shadow Data โดยส่ง Payload ดังนี้ { "data":{ "field name 1": value 1, "field name 2": value 2, ..., "field name n": value n }}
@shadow/batch/update	อัปเดตค่าใน Shadow แบบเป็นชุดข้อมูล (shadow batch update)
@shadow/data/get	ร้องขอ Shadow Data ของตัวเองแบบทั้งหมด รอรับข้อมูลโดย Subscribe Topic @private/# หรือ @private/shadow/data/get/response

Subscribe Topic	
@private/shadow/data/get/response	รอรับ Shadow Data เมื่อมีการร้องขอข้อมูลไป
@private/shadow/batch/update/response	รอรับข้อความตอบกลับการอัปเดต Shadow แบบเป็นชุดข้อมูล
@private/#	รอรับทุกข้อมูล Topic ที่ขึ้นต้นด้วย @private/ รวมถึงข่าวสารต่างๆ ที่ Platform ต้องการแจ้งก็จะถูก Publish มาที่ Topic นี้

2.5 ไลบรารี PubSubClient (<https://github.com/knolleary/pubsubclient>)

PubSubClient คือซอฟต์แวร์ไลบรารีที่ทำงานเป็นไคลเอนต์สำหรับการส่งรับ Message อย่างง่ายผ่านการ Publish หรือ Subscribe กับเซิร์ฟเวอร์ที่รองรับ MQTT โดยสามารถ Publish ที่ QoS 0 และ Subscribe ที่ QoS 0 หรือ 1 ฟังก์ชันบางส่วนในไลบรารี PubSubClient แสดงในตารางที่ 2.1

ตารางที่ 2.1 ฟังก์ชันบางส่วนในไลบรารี PubSubClient

ฟังก์ชัน	ต้นแบบฟังก์ชันและหน้าที่
constructor	PubSubClient(Client& client); ฟังก์ชันคอนสตรัคเตอร์ (constructor) สร้างวัตถุจากคลาส PubSubClient
setServer	PubSubClient& setServer(const char * domain, uint16_t port); กำหนดเซิร์ฟเวอร์ที่ต้องการเชื่อมต่อ
connect	bool connect(const char* clientId, const char* username, const char* password); เชื่อมต่อกับเซิร์ฟเวอร์
connected	bool connected(); ตรวจสอบสถานะการเชื่อมต่อกับเซิร์ฟเวอร์
disconnect	void disconnect(); ยกเลิกการเชื่อมต่อกับเซิร์ฟเวอร์
setCallback	PubSubClient& setCallback(MQTT_CALLBACK_SIGNATURE); ใช้กำหนดฟังก์ชันที่จะถูกเรียกเมื่อไคลเอนต์ได้รับ Message ใหม่
publish	bool publish(const char* topic, const char* payload); ส่ง Message ไปยัง Topic ที่กำหนด
subscribe	bool subscribe(const char* topic, uint8_t qos); สมัครติดตามการรับ Message ที่เกิดขึ้นใน Topic นั้น ๆ
unsubscribe	bool unsubscribe(const char* topic); ยกเลิกการรับ Message ที่เกิดขึ้นใน Topic นั้น ๆ
loop	bool loop(); เรียกใช้อย่างสม่ำเสมอเพื่อให้ไคลเอนต์ประมวลผล Message ที่เข้ามาและยืนยันการเชื่อมต่อกับเซิร์ฟเวอร์

2.6 Freeboard บน NETPIE

Freeboard คือส่วนการนำเสนอภาพข้อมูล (data visualization) ของ Device อย่างง่ายบน NETPIE สามารถแสดงผลได้หลากหลาย ทั้งแบบ Text, Gauge, Indicator Light, Button, Toggle, FeedView, Map เป็นต้น และใส่คำสั่ง Javascript สำหรับ Action ต่าง ๆ ได้

2.7 RESTful API บน NETPIE

RESTful API เป็นช่องทางสำหรับให้ Device เรียกใช้บริการด้วย HTTP Protocol ซึ่งเหมาะสำหรับใช้เป็นช่องทางในการผสมรวม (integration) ระบบต่างๆ ทั้งที่มีอยู่แล้วหรือกำลังจะพัฒนาขึ้นมาใหม่ โดยไม่จำกัดว่าจะต้องพัฒนาจากภาษาโปรแกรมใด (ทดสอบการทำงานของ API ได้ที่ <https://trial-api.netpie.io>)

3. Multitasking

Multitasking คือการทำงานมากกว่าหนึ่งงาน (task) พร้อมกัน โดยแนวคิดคือทุกงานจะต้องทำพร้อม ๆ กัน ในเวลาเดียวกัน (parallelism) แต่ในความเป็นจริงนั้นหน่วยประมวลผลหรือ CPU จะสลับการทำงานไปมาระหว่างงาน (context switching) อย่างรวดเร็วจนดูเหมือนสามารถทำงานหลายงานได้ภายในเวลาเดียวกัน สำหรับการทำงานแบบ Multitasking บน Arduino IDE สามารถทำได้ดังนี้

3.1 Multitasking โดยใช้ millis()

Multitasking โดยใช้ millis()ทำได้โดยกำหนดให้ทุกงานอยู่ภายใต้ loop() และอาศัยฟังก์ชัน millis() เพื่ออ่านค่าเวลาและตรวจสอบว่าแต่ละงานถึงกำหนดเวลาในการทำงานอีกครั้งหนึ่งแล้วหรือไม่ ดังรูป 3-1 โดยแต่ละงานควรใช้เวลาสั้น ๆ และไม่ควรมีการวนลูปรอที่ใช้เวลานาน ๆ

```
void loop(){
  /* 250ms Task */
  currentMillis = millis();
  if(currentMillis - previousMillis1 > 250){
    previousMillis1 = currentMillis;
    /* Task 1 code here */
  }

  /* 500ms Task */
  currentMillis = millis();
  if(currentMillis - previousMillis2 > 500){
    previousMillis2 = currentMillis;
    /* Task 2 code here */
  }

  /* 1000ms Task */
  currentMillis = millis();
  if(currentMillis - previousMillis3 > 1000){
    previousMillis3 = currentMillis;
    /* Task 3 code here */
  }
}
```

รูปที่ 3-1 ตัวอย่าง Multitasking โดยใช้ millis() ทุก ๆ 250, 500 และ 1000 ms

3.2 Multitasking โดยใช้ FreeRTOS

FreeRTOS เป็นระบบปฏิบัติการสำหรับอุปกรณ์ Embedded ใช้ได้ในหลากหลายไมโครคอนโทรลเลอร์ ซึ่งทาง Espressif ได้นำ FreeRTOS มาใส่ในชุดพัฒนาของ ESP32 ทำให้สามารถใช้งานใน Arduino IDE โดยตรงได้เลย การสร้าง Task สามารถทำได้โดยใช้ฟังก์ชัน xTaskCreate() ซึ่งมีรูปแบบการใช้งานดังนี้

```
void xTaskCreate(TaskFunction_t pvTaskCode,    // ฟังก์ชันที่จะไปเรียกทำงานในรูปแบบ Task
                const char *pcName,           // ชื่อของ Task ที่สร้าง
                int usStackDepth,              // พื้นที่บนแรมสำหรับตัวแปรแบบ Local ใน Task
                void *pvParameters,            // ข้อมูลที่ต้องการส่งเข้าไปในฟังก์ชัน Task
                int uxPriority,                  // ระดับความสำคัญของ Task
                TaskHandle_t *pxCreatedTask);  // ตัวแปรพอยเตอร์ที่จะนำ Task ไปควบคุมในขั้นตอนต่อไป
```

ภายใน Task ที่สร้างจะต้องมีการเรียกใช้ฟังก์ชันหน่วงเวลา vTaskDelay() เสมอเพื่อเปิดโอกาสให้ FreeRTOS ได้นำเวลาที่ถูกหน่วงไปทำงานใน Task อื่น ๆ ที่มีระดับความสำคัญรองลงมา หากภายใน Task ไม่มีการเรียกใช้ฟังก์ชันหน่วงเวลาอยู่เลยจะทำให้ Task นั้นไม่ทำงาน FreeRTOS เปิดให้สามารถสร้าง Task ได้ไม่จำกัด แต่จะถูกจำกัดจำนวนพื้นที่แรม โดยจะใช้ได้เท่าที่จองไว้เท่านั้น

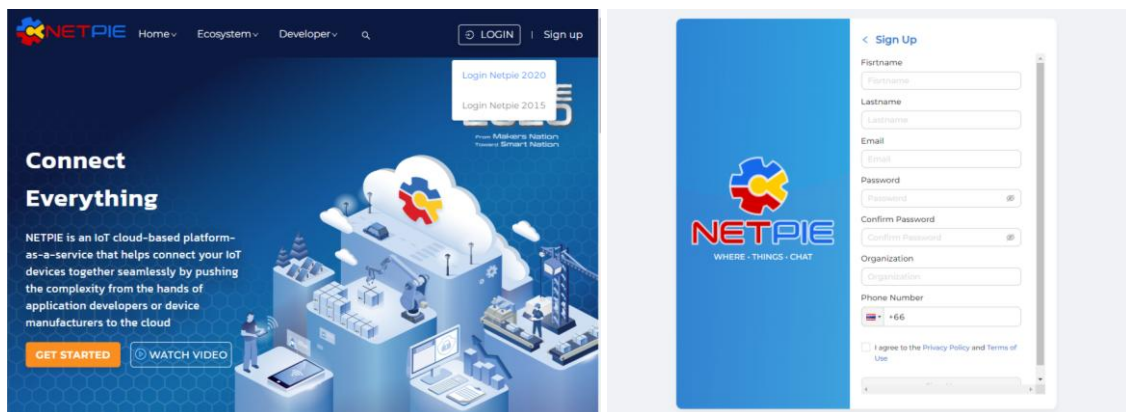
การทดลองที่ 0 | การลงทะเบียนบัญชีผู้ใช้ NETPIE และการติดตั้งไลบรารี PubSubClient บน Arduino IDE

สิ่งที่ต้องส่ง

ไม่มี

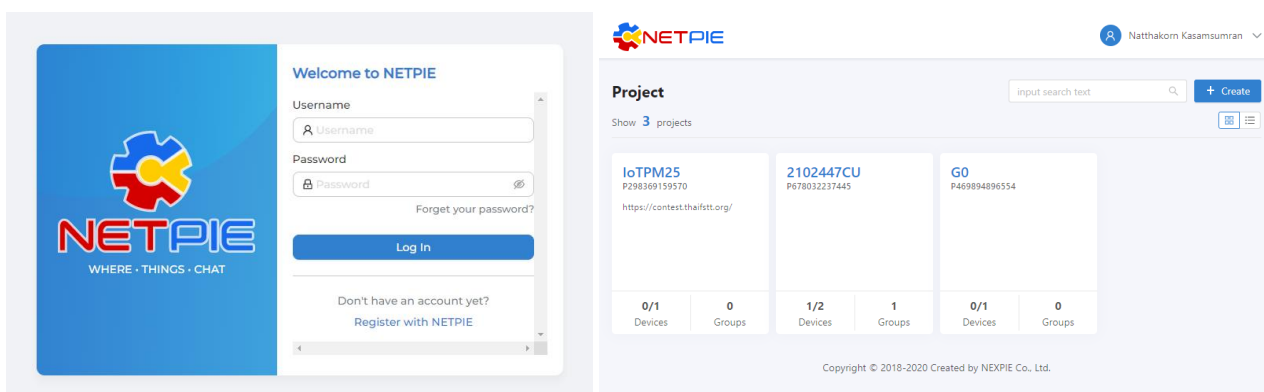
ขั้นตอนปฏิบัติ

1. ไปที่เว็บไซต์ <https://netpie.io> จากนั้นเลือก Sign up ดังรูปที่ L0-1 กรอกข้อมูลให้ครบถ้วน (email กับ phone number ไม่สามารถแก้ไขภายหลังได้) จากนั้นรอรับ password ทาง email



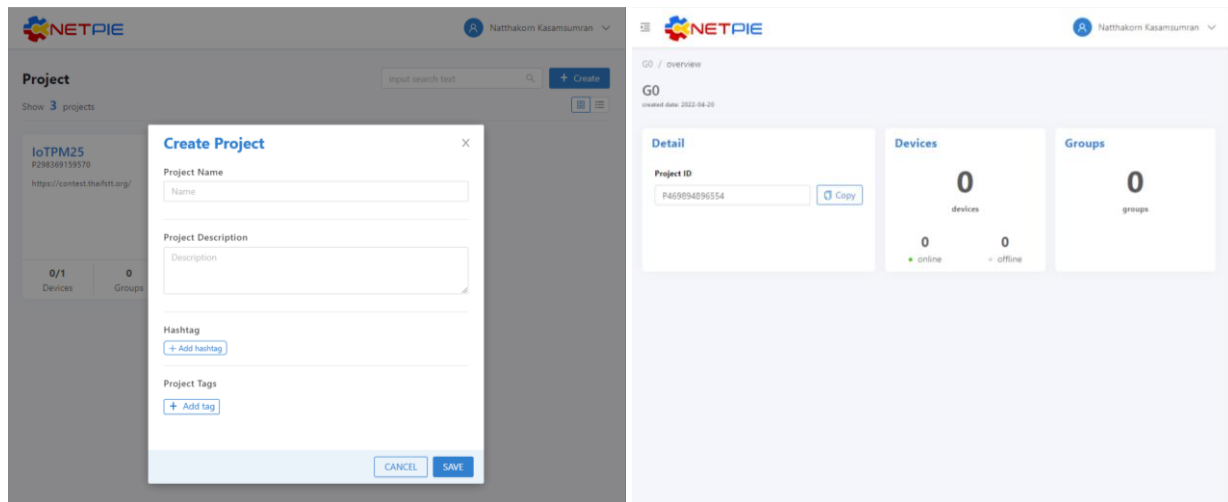
รูปที่ L0-1 หน้าของเว็บไซต์ netpie.io และหน้า Sign up

2. ไปที่เว็บไซต์ <https://netpie.io> จากนั้นเลือก LOGIN เพื่อเข้าสู่ระบบ ดังรูปที่ L0-2 สามารถเปลี่ยน password ได้ โดยกดชื่อบัญชีด้านขวาบนแล้วเลือก profile



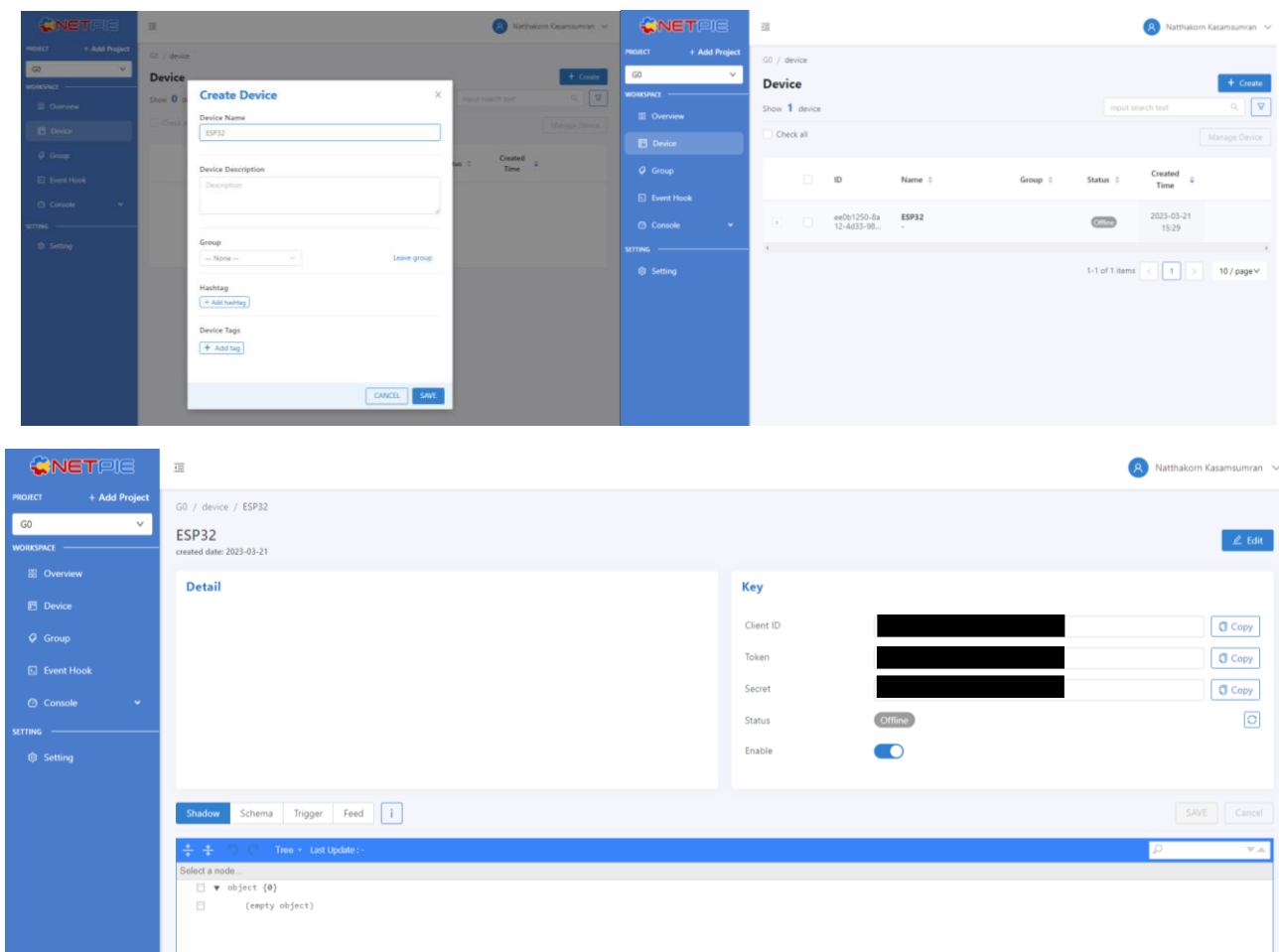
รูปที่ L0-2 หน้า LOGIN และหน้าหลักเมื่อเข้าสู่ระบบ

3. กดปุ่ม “+” เพื่อสร้าง Project ตั้งชื่อ Project กดปุ่ม Create แล้วกดเลือก Project ที่สร้างไว้ ดังรูปที่ L0-3



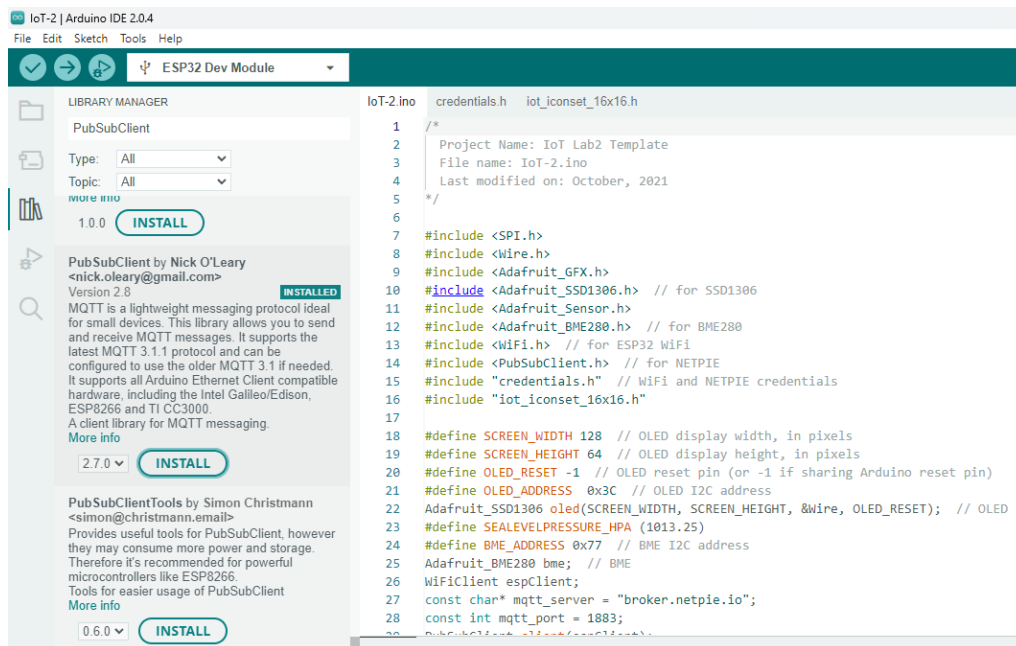
รูปที่ L0-3 การสร้าง Project บน NETPIE

4. กดเลือก Device List จากนั้นกดปุ่ม “+ Create” เพื่อสร้าง Device ตั้งชื่อ Device กดปุ่ม Create แล้วกดเลือก Device ที่สร้างไว้ ดังรูปที่ L0-4 โดยแต่ละ Device จะมี Key คือ ClientID, Token, และ Secret ไว้ใช้สำหรับการเชื่อมต่อ



รูปที่ L0-4 การสร้าง Device ใน Project

5. ติดตั้งไลบรารี PubSubClient บน Arduino IDE โดยไปที่เมนู Sketch → Include Library → Manage Libraries ค้นหาคำว่า PubSubClient แล้วกดปุ่ม Install เพื่อติดตั้ง



รูปที่ L0-5 การติดตั้งไลบรารีด้วย Manage Libraries บน Arduino IDE 2.0



รูปที่ L0-6 การติดตั้งไลบรารี PubSubClient

*** แนะนำให้ใช้ PubSubClient by Nick O'Leary <nick.oleary@gmail.com> ***

เนื่องจากทดสอบแล้วสามารถสื่อสารกับ NETPIE 2020 ได้ปกติ หากนิสิตอยากทดสอบ Library อื่นๆ ได้

☆ จบการทดลองที่ 0 ☆

การทดลองที่ 1 | การการเก็บและนำเสนอภาพข้อมูลด้วย NETPIE

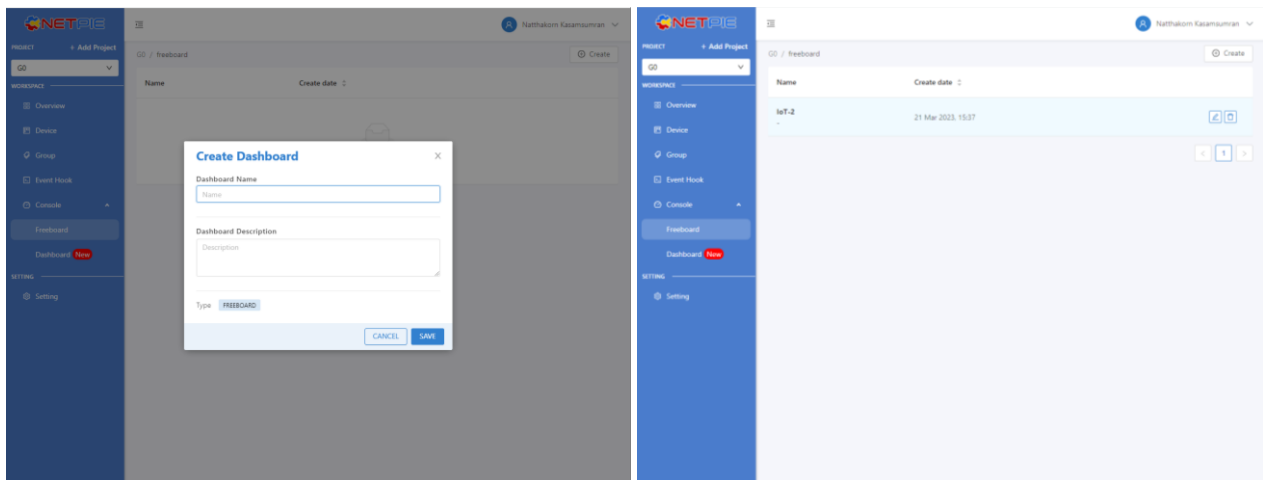
สิ่งที่ต้องส่ง

มีสิ่งที่ต้องส่ง 2 รายการ คือ

1. ไฟล์ IoT-2_1.zip โดยประกอบด้วยไฟล์ต่าง ๆ ที่แก้ไขยกเว้น credentials.h และ iot_iconset_16x16.h อย่าลืมแก้ไขไฟล์ schema.txt ด้วย โดยส่งใน attachment slot บน mycourseville
2. วิดีโอแสดงการทำงานของอุปกรณ์ โดยส่ง URL ของวิดีโอใน textbox บน mycourseville

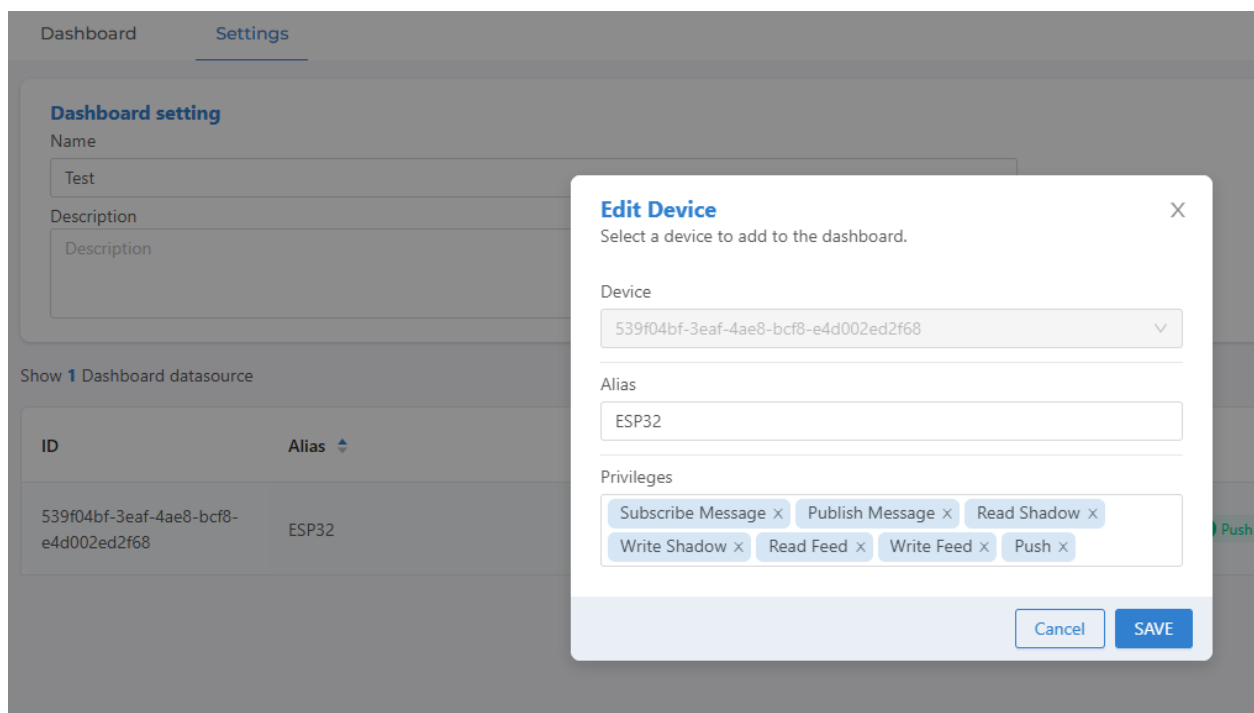
ขั้นตอนปฏิบัติ

1. ดาวน์โหลดไฟล์ IoT-2.zip โดยประกอบด้วย 6 ไฟล์ ได้แก่
 - a. Lab447-IoT-2.pdf ชีทแล็บนี้
 - b. IoT-2.ino โค้ดหลัก
 - c. credentials.h เก็บรหัสผ่าน WiFi และ NETPIE
 - d. iot_iconset_16x16.h เก็บไอคอน
 - e. schema.txt โค้ด Schema
 - f. MTask/Mtask.ino โค้ดตัวอย่าง Multitasking โดยใช้ FreeRTOS
2. ให้นิสิตต่อโมดูล OLED และโมดูล BME280 เข้ากับบอร์ด IOXESP32+ ผ่านทาง I2C และแก้ไข I2C Address ในไฟล์ IoT-2.ino (แนะนำควรใช้ I2C scanner ในการหา Address)
3. จากนั้นแก้ไข SSID และ password ที่มาจาก WiFi Hotspot ในส่วนของ /* WiFi*/ ของนิสิต และใน /*NETPIE*/ ให้นำ ClientID, Token, และ Secret จากรูปที่ L0-4 มาใส่ในตัวแปรในไฟล์ credentials.h แนะนำให้ใช้ WiFi Hotspot เพื่อความสะดวกในการทดลองเปิดปิด WiFi (ถ้าเป็น iPhone ให้เปิด Maximum compatibility)
4. จากนั้นเข้าไปที่ Schema ของ Device ใน NETPIE เปลี่ยนมุมมองจาก Tree เป็น Code จากนั้น Copy โค้ดในไฟล์ schema.txt ไปใส่ในกล่องแล้วกด Save
5. Upload โค้ดในไฟล์ IoT-2.ino ไปยังบอร์ด IOXESP32+ แล้วดูผลที่จอ OLED และส่วน Shadow กับ Feed ของ Device ใน NETPIE
6. กดเลือก Dashboard จากนั้นกดปุ่ม “+ Create” เพื่อสร้าง Dashboard ตั้งชื่อ Dashboard กดปุ่ม Save แล้วกดเลือก Dashboard ที่สร้างไว้ ดังรูปที่ L1-1



รูปที่ L1-1 การสร้าง Dashboard ใน Project ของนิสิต

7. เมื่อนิสิตเข้ามาในหน้า Dashboard แล้วให้เลือก Setting จากนั้นกดปุ่ม “+ Add device” และเลือก Device ที่ได้สร้างไว้ก่อนหน้านี้ ส่วนของ Privileges ให้เลือก Subscribe Message, Publish Message, Read Shadow, Write Shadow, Read Feed, Write Feed และ Push แล้วจึงกด Save ดังรูปที่ L1-2



รูปที่ L1-2 การกำหนดสิทธิ์การเข้าถึงของอุปกรณ์

8. กลับมาที่หน้า Dashboard ให้กดปุ่ม Edit แล้วจึงกดปุ่ม “+ Add Panel” นิสิตสามารถปรับขนาดความกว้างของ Panel ได้โดยการกดไอคอนรูปประแจ (Config) และกำหนดชื่อ Panel ใน Title และกำหนดความกว้างด้วยการเลื่อน Columns จากนั้นกดปุ่ม Done และจึงกดปุ่ม Save (คำเตือน ให้กดปุ่ม Save ทุกครั้งที่มีการแก้ไข Dashboard ไม่เช่นนั้นแล้วสิ่งที่ทำจะไม่ถูกบันทึก)

9. จากนั้นให้ทำการเพิ่ม Widget โดยการกดไอคอน + เลือก Type = Gauge ให้กรอกรายละเอียดดังนี้ Title = Altitude, ให้นิสิตกดปุ่ม “+ Device” ใน Value แล้วจะได้ #[“ชื่อ Device ของนิสิต”] แล้วกรอกให้เป็น Value = #[“ชื่อ Device ของนิสิต”][“shadow”][“altitude”].toFixed(2), Units = m, Minimum = 0, Maximum = 200 ปรับ Human friendly number เป็น No แล้วกด Done แล้วจึงกด Save
10. ให้นิสิตสร้าง Panel เพิ่ม กำหนดให้ Columns = 3 จากนั้นให้ทำการเพิ่ม Widget โดยการกดไอคอน + เลือก Type = Chart, Data Source = #[“ชื่อ Device ของนิสิต”][“feed”], Filter = altitude แล้วกด Done
11. กด Save ที่ Dashboard
12. ให้นิสิตแก้ไขโค้ดในไฟล์ IoT-2.ino และ Schema ของ Device เพื่อรองรับอุณหภูมิ (Temperature) และความชื้น (Humidity) หน่วยเป็น %Rh (relative humidity) ของโมดูล BME280 เพิ่มเติม โดยทำการอ่านค่าแสดงไอคอนและค่าบนจอ OLED (อุณหภูมิหน่วยเป็นองศาเซลเซียส °C) ส่งค่าไปยัง NETPIE และแสดงผลบน Dashboard (อุณหภูมิหน่วยเป็นองศาฟาเรนไฮต์ °F) ทั้งแบบ Gauge และ Chart แสดงคล้ายกับ Feed ในหน้า Device ของ NETPIE
13. หลังจากเก็บค่ามาประมาณสัก 10 นาทีเป็นอย่างน้อย บันทึกวิดีโอการแสดงผลบนจอ OLED และแสดงผลบน Chart กับ Feed และการแสดงผลบนจอ OLED เมื่อปิด WiFi ไปสัก 20 วินาที และเปิด WiFi Hotspot กลับ ซึ่งชุดคำสั่งที่นิสิตปรับปรุงเพิ่มจะต้องสามารถเชื่อมต่อ WiFi ได้อย่างอัตโนมัติ และสามารถส่งค่าไปยัง NETPIE ได้หลังจากเชื่อมต่ออินเทอร์เน็ตได้โดยสมบูรณ์

☆ จบการทดลองที่ 1 ☆

*** ถ้าจอ OLED แสดงผลผิดเพียง ให้ลองเอานิ้วแตะทางด้าน SPI ของโมดูล BME280 ไว้ ***

การทดลองที่ 2 | การประยุกต์ใช้ **Multitasking**

สิ่งที่ต้องส่ง

มีสิ่งที่ต้องส่ง 2 รายการ คือ

1. ไฟล์ IoT-2_2.zip โดยประกอบด้วยไฟล์ต่าง ๆ ที่แก้ไขยกเว้น credentials.h และ iot_iconset_16x16.h โดยส่งใน attachment slot บน mycourseville
2. วิดีโอแสดงการทำงานของอุปกรณ์ โดยส่ง URL ของวิดีโอใน textbox บน mycourseville

ขั้นตอนปฏิบัติ

1. ให้นักศึกษาทดลอง Upload โค้ดในไฟล์ MTask.ino ไปยังบอร์ด IOXESP32+ แล้วสังเกตการทำงานของหน้าจอ OLED และ LED บนบอร์ด IOXESP32+ (LED onboard at GPIO 5)
2. ให้นักศึกษาแก้ไขชุดคำสั่งจากการทดลองที่ 1 ให้เป็นแบบ Multitasking โดยใช้ FreeRTOS กำหนดให้
 - Task 1 ทำการอ่านค่าความดัน (Pressure) และความสูง (Altitude) และส่งค่าความดัน ณ ขณะนั้น และค่าความสูงเท่ากับ 0 ไปยัง NETPIE ทุก ๆ 3 วินาที โดยตรวจสอบการเชื่อมต่อ NETPIE ว่ายังอยู่ก่อนส่งค่าไปยัง NETPIE
 - Task 2 ทำเช่นเดียวกับ Task1 แต่กับค่าอุณหภูมิ (Temperature) ทุก ๆ 4 วินาที
 - Task 3 ทำเช่นเดียวกับ Task1 แต่กับค่าความชื้น (Humidity) ทุก ๆ 5 วินาที
 - Task 4 ตรวจสอบการเชื่อมต่อ NETPIE และ WiFi ทุก ๆ 7 วินาที
 - Task 5 แสดงไอคอนและค่าต่าง ๆ และการเชื่อมต่อ WiFi ทุก ๆ 1 วินาที
3. บันทึกการแสดงผลบนจอ OLED และส่วน Feed ของ Device เมื่อปิด WiFi Hotspot ไปสัก 20 วินาที และเปิด WiFi Hotspot ให้ IOXESP32+ เชื่อมต่ออินเทอร์เน็ต และสังเกตลักษณะ Feed

★ จบการทดลองที่ 2 ★