ใช้งาน PIEBOX ด้วย

Flow Engine

แก้ไขล่าสุดเมื่อ 2 กันยายน พ.ศ.2563 15:59 น.

บทนำ

ภาษาของการเขียนโปรแกรมในยุคปัจจุบันมีความเปลี่ยนแปลงอย่างรวดเร็วและหลากหลายมากขึ้น นำมาซึ่งการเรียนรู้ของผู้ใช้งานที่ต้องทำความเข้าใจเกี่ยวกับประโยชน์ของแต่ละภาษาในการสร้างโปรแกรมหนึ่งขึ้นมาใช้งาน

Flow Programming ถูกออกแบบขึ้นมาเพื่อลดความยุ่งยากและซับซ้อนของการเขียนโปรแกรมให้น้อยลง โดยใช้ลักษณะการเชื่อมต่อการทำงานแต่ละส่วนเข้าด้วยกัน และทำการตั้งค่าภายในการทำงานแต่ละส่วนลงไปโดยที่ผู้ใช้งานไม่จำเป็นต้องรู้วิธีในการเขียนโปรแกรมภาษาต่าง ๆ หรือต้องการเพียงความรู้พื้นฐานเบื้องต้นเท่านั้น

PIEBOX ใช้ Flow Programming ในลักษณะการอนุญาตให้ผู้ใช้งานสามารถสร้างโปรแกรมขึ้นสามารถสร้างโปรแกรมขึ้นมาได้เสมือนการวางโฟลว์

สารบัญ

PIEBOX

PIEBOX จัดว่าเป็น IoT Edge Gateway ขนาดเล็กที่ถูกติดตั้งโปรแกรมเอาไว้ภายในพร้อมรองรับการเชื่อมต่อกับแพลตฟอร์ม IoT อีกทั้งยังรองรับการเชื่อมต่อเข้ากับ Serial Port ต่าง ๆ รวมทั้งโปรโตคอล Modbus, HTTP, และ MQTT อีกด้วย

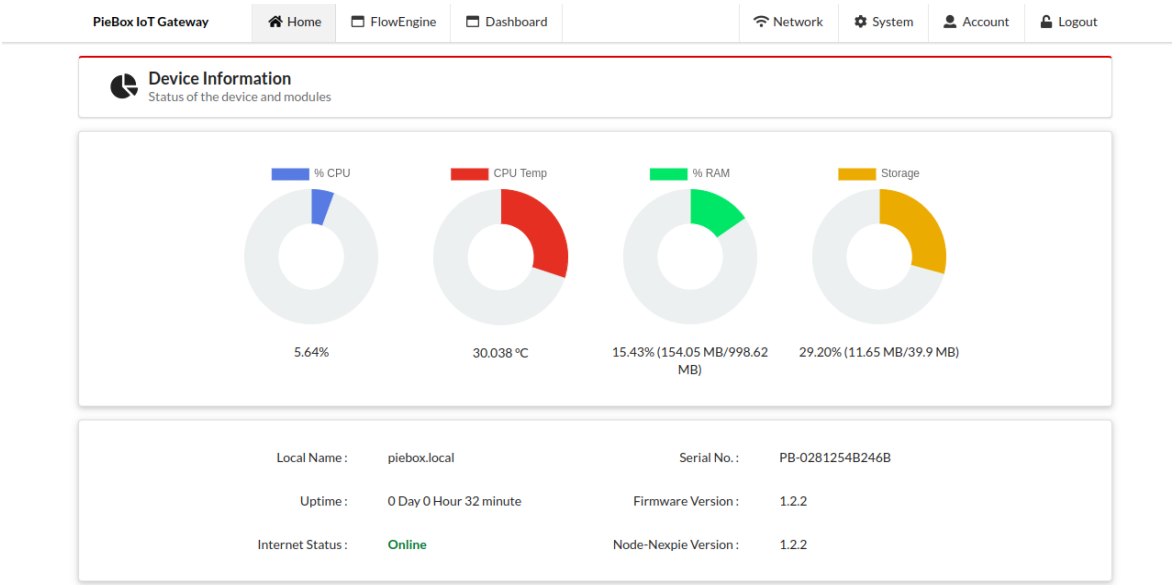
ในการใช้งาน Flow Engine ที่อยู่ภายใน PIEBOX นั้น ผู้ใช้งานจำเป็นต้องมีตัวกล่อง PIEBOX เสียก่อนจึงจะสามารถใช้งานได้ ซึ่งตัวกล่องเองก็จำเป็นที่จะต้องเชื่อมต่อเข้ากับอินเตอร์เน็ตเพื่อเปิดหน้าต่างโปรแกรมนี้ขึ้นมา โดยการเชื่อมต่ออินเตอร์เน็ตจากกล่องสามารถทำได้ 2 วิธี ทั้งการเชื่อมต่อแบบมีสายและการเชื่อมต่อแบบไร้สาย

การเชื่อมต่อแบบใช้สาย

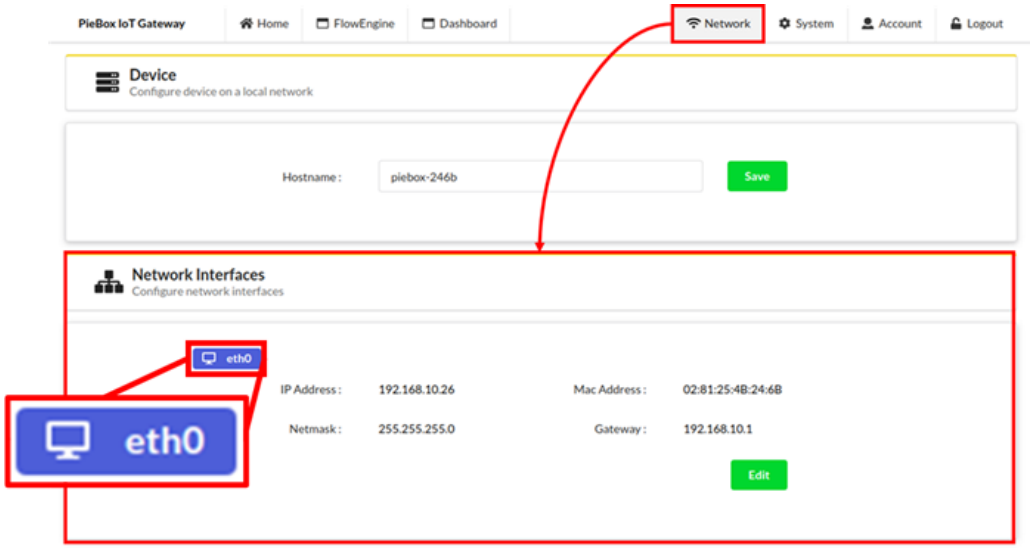
การเชื่อมต่อเข้ากับกล่องแบบใช้สายนั้น ผู้ใช้งานจำเป็นจะต้องมีอุปกรณ์กระจายสัญญาณ (Router) เพื่อทำหน้าที่เป็นตัวกลางในการต่ออินเตอร์เน็ต โดยใช้ช่อง WAN ของตัวกล่อง PIEBOX เชื่อมต่อเข้ากับช่อง LAN ของอุปกรณ์กระจายสัญญาณ จากนั้นจึงเชื่อมสัญญาณจากอุปกรณ์กระจายสัญญาณเข้าสู่คอมพิวเตอร์ผ่านทาง WiFi หรือสาย LAN ตามที่ผู้ใช้งานต้องการได้ทันที

เมื่อผู้ใช้งานสามารถเชื่อมต่ออุปกรณ์ได้เรียบร้อยแล้ว ให้ผู้ใช้งานทำการเปิดเว็บเบราเซอร์ขึ้นมาแล้วไปยัง http://piebox.local เพื่อเข้าสู่หน้าต่างการใช้งานภายในกล่อง PIEBOX

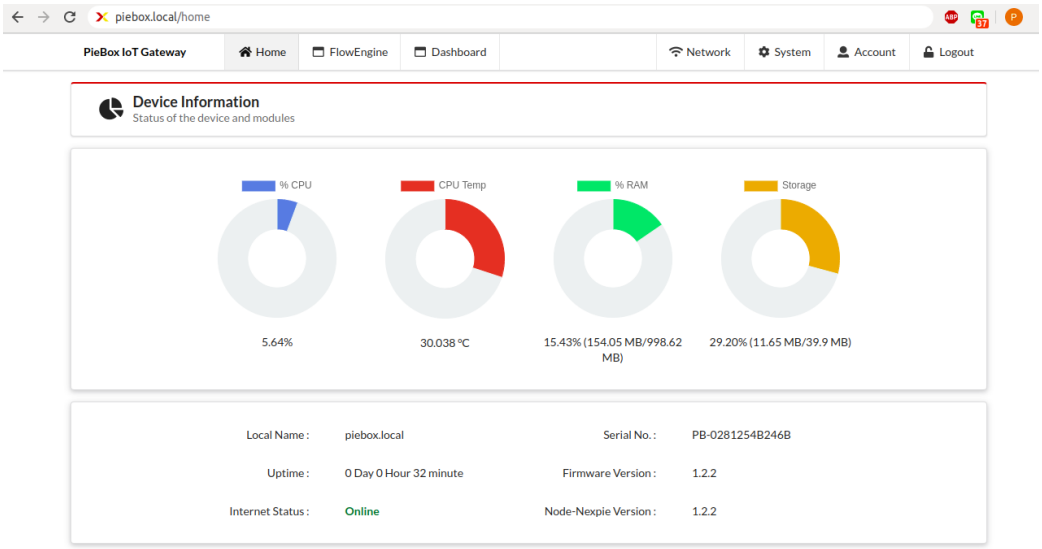
ผู้ใช้งานทำการ login เข้าไปโดยการใช้ Username และ Password ที่ตั้งค่าเอาไว้ ซึ่งค่าเริ่มต้นของทั้งสองตัวแปรนี้คือ ‘admin’ โดยหากการเข้าสู่ระบบนั้นสามารถทำได้สำเร็จ หน้าต่างจะปรากฏขึ้นมาดังรูปด้านล่างนี้



อย่างไรเสีย นี่ยังไม่ใช่บทสรุปที่แท้จริงของการเชื่อมต่ออินเตอร์เน็ต ผู้ใช้งานยังจำเป็นต้องไปที่เมนู Network ในหน้าโปรแกรมเพื่อทำการระบุวิธีการตั้งค่าที่ผู้ใช้งานเลือกอีกเล็กน้อย โดยในกรณีที่ผู้ใช้งานทำการเชื่อมต่อแบบมีสาย ให้ทำการตรวจสอบว่า ภายในแถบ ‘Network Interfaces’ ในหน้าโปรแกรมนั้นแสดงข้อมูลการเชื่อมต่อกับอุปกรณ์กระจายสัญญาณได้ถูกต้องหรือไม่ ซึ่งจะแสดงรายละเอียดอยู่ภายในการ์ด `eth0`



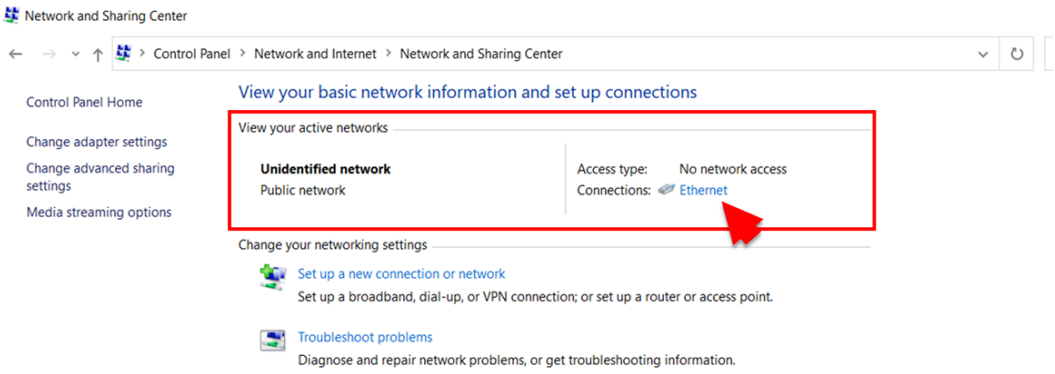
หากการเชื่อมต่อเข้ากับอินเตอร์เน็ตสำเร็จ หน้าต่างแรกของโปรแกรม (Home) จะปรากฏสถานะ ‘Online’ ขึ้นที่ ‘Internet Status’ เพียงเท่านี้ กล่อง PIEBOX ก็พร้อมที่จะนำไปใช้งานในส่วนอื่น ๆ ต่อได้แล้ว



การเชื่อมต่อแบบไร้สาย

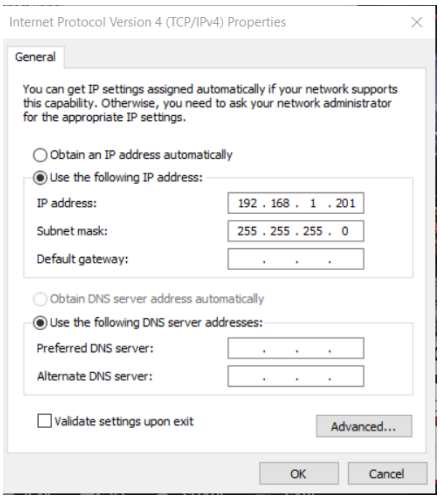
ในการเชื่อมต่อคอมพิวเตอร์เข้ากับกล่อง PIEBOX แบบไร้สายนั้น ผู้ใช้งานสามารถเชื่อมต่อ PIEBOX เข้ากับอุปกรณ์ที่ต้องการจากช่อง LAN ของตัวกล่องเข้าสู่ช่อง Ethernet ของคอมพิวเตอร์ได้โดยตรง จากนั้นให้ทำการแก้ไขค่า IP Address ของคอทพิวเตอร์เพื่อให้สามารถรับสัญญาณจากกล่องได้ตามขั้นตอนต่อไปนี้

1. ไปยัง Control Panel > Netwrok and Internet > Network and Sharing Center จากนั้นให้ทำการคลิกที่ ‘Ethernet’ ในส่วนของ View your active networks



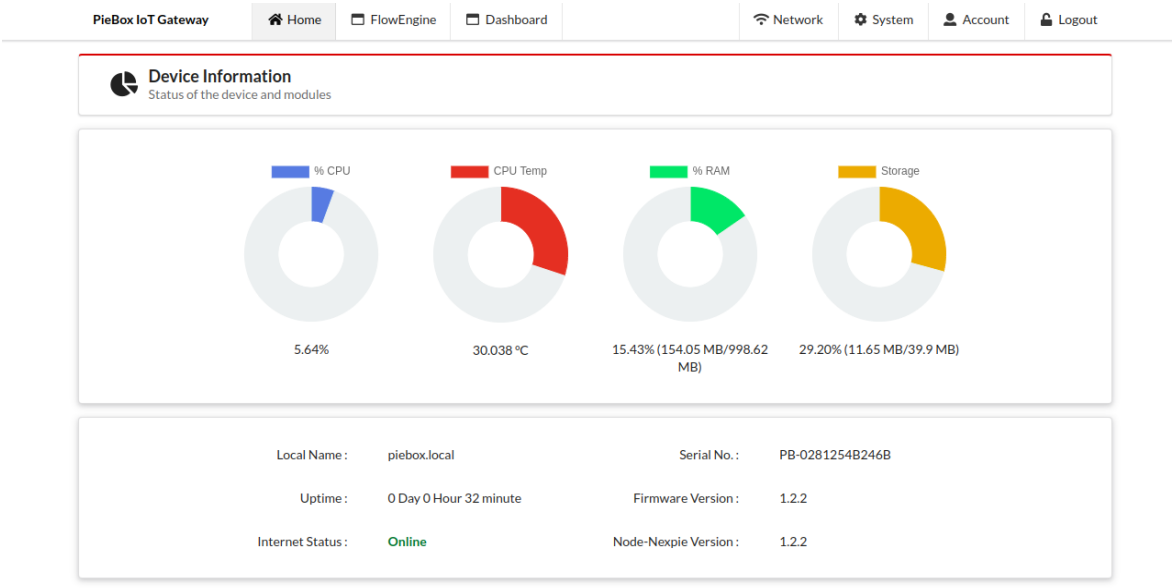
2. หน้าต่าง Eternet Status จะปรากฏขึ้นมา ให้ผู้ใช้งานทำการคลิกที่ Properties
3. เลือก *Internet Protocol Version 4 (TCP/IP)* จากนั้นคลิก Properties

4. เลือก *Use the following IP address* แล้วตั้งค่า IP ให้อยู่ในเครือข่ายเดียวกันกับกล่อง PIEBOX โดย IP ที่ตั้งค่าต้องไม่ซ้ำกับ IP Address เริ่มต้นของกล่องคือ 192.168.1.200 ดังนั้นผู้ใช้งานสามารถตั้งค่า IP ให้กลายเป็น 192.168.1.xxx ที่ ‘xxx’ นั้นไม่ใช่ 200 ได้นั่นเอง
5. กด OK เพื่อทำการบันทึกค่า IP ที่ต้องการแก้ไข



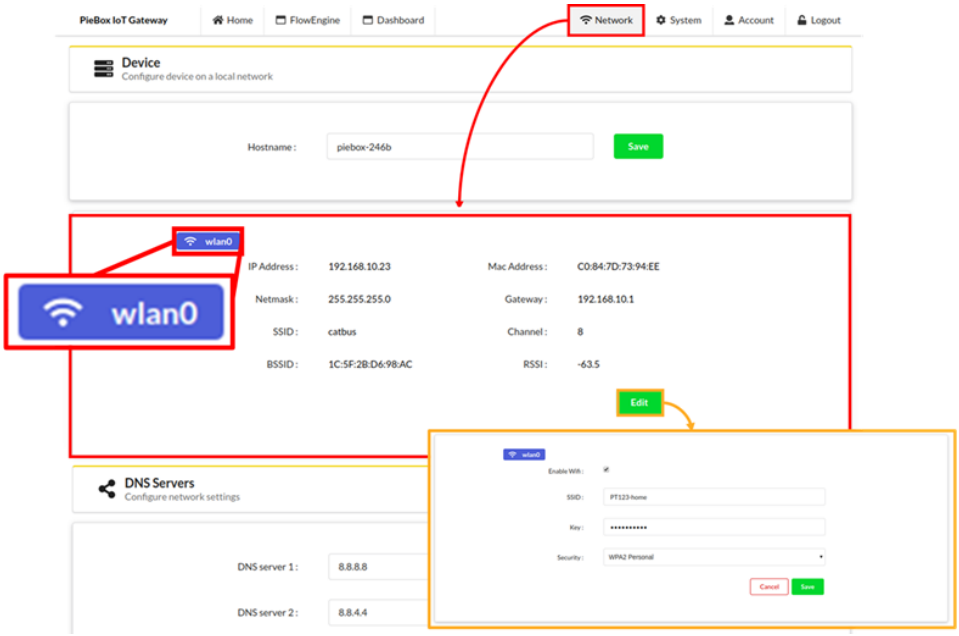
หลังจากทำการเชื่อมต่อกับกล่อง PIEBOX และทำการแก้ไข IP ของคอมพิวเตอร์เป็นที่เรียบร้อยแล้ว ก่อนการเข้าใช้งานกล่องผ่าน URL นั้น หากคอมพิวเตอร์ที่ใช้งานมีการเชื่อมต่อ WiFi อยู่ ผู้ใช้งานต้องทำการตัดการเชื่อมต่อกับ WiFi นั้นเสียก่อนเพื่อให้เหลือเฉพาะการเชื่อมต่อระหว่าง LAN ของตัวกล่อง PIEBOX กับ Ethernet ของคอมพิวเตอร์เท่านั้น จากนั้นจึงเข้าใช้งานตัวกล่องผ่านเว็บเบราเซอร์ http://piebox.local

เมื่อไปยัง URL นี้แล้ว หน้าต่างจะปรากฏหน้าการเข้าสู่ระบบขึ้นมา โดยในเบื้องต้น ให้ผู้ใช้งานแทนค่าทั้ง Username และ Password ด้วยคำว่า ‘admin’ เพื่อเริ่มต้นการใช้งานในครั้งแรก ซึ่งหากการเข้าสู่ระบบนั้นสำเร็จ หน้าต่างจะปรากฏขึ้นมาดังรูปด้านล่างนี้

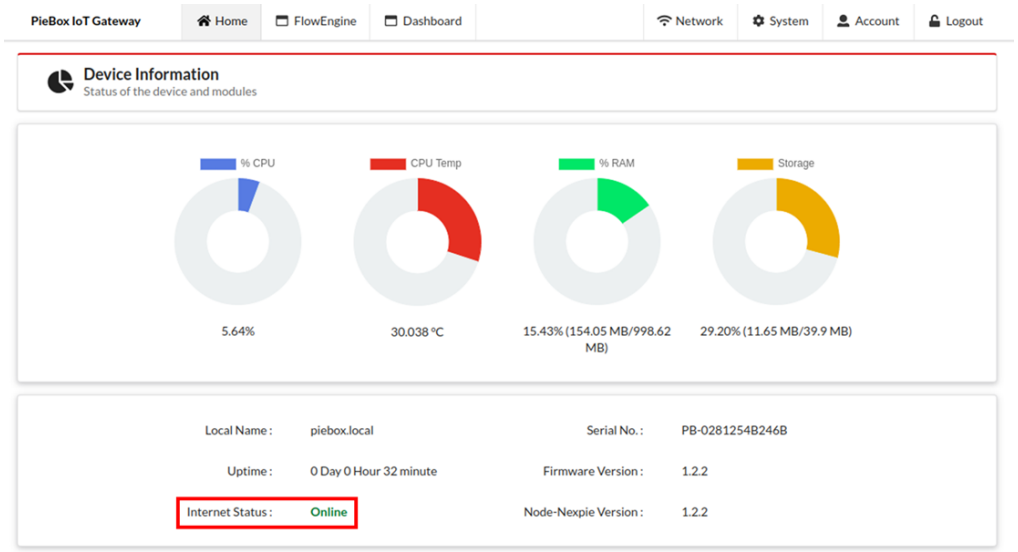


ในการเลือกเชื่อมต่อกล่อง PIEBOX แบบไร้สายนั้น ให้ผู้ใช้งานค้นหาการ์ด `wlan0` ในหน้าโปรแกรม จากนั้นให้ทำการคลิกที่ปุ่ม ‘Edit’ เพื่อทำการกรอกข้อมูลของ WiFi ที่ต้องการเชื่อมต่อลงไป จากนั้นจึงทำการคลิกปุ่ม ‘Save’ เพื่อบันทึกการเปลี่ยนแปลงนั้น และรอจนกว่าที่ PIEBOX จะ

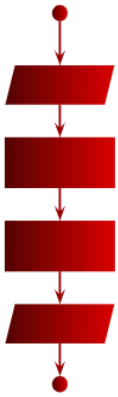
สามารถเชื่อมต่อกับ WiFi ได้สำเร็จ หากยังไม่แสดงข้อมูล ให้ลองกดปุ่ม F5 บนแป้นคีย์บอร์ดเพื่อ Refresh หน้าต่างเบราเซอร์หนึ่งครั้ง



หากการเชื่อมต่อเข้ากับอินเตอร์เน็ตสำเร็จ หน้าต่างแรกของโปรแกรม (Home) จะปรากฏสถานะ ‘Online’ ขึ้นที่ ‘Internet Status’ เพียงเท่านี้ กล่อง PIEBOX ก็พร้อมที่จะนำไปใช้งานในส่วนอื่น ๆ ต่อได้แล้ว

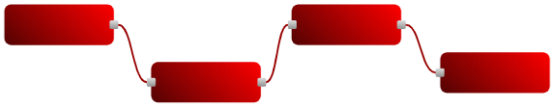


Flow Engine



ในปัจจุบันที่ภาษาในการเขียนโปรแกรมนั้นมีจำนวนมากขึ้น การเรียนรู้ภาษาใหม่ ๆ เหล่านั้นของผู้ใช้งานอาจไม่ใช่เรื่องง่ายสักเท่าไหร่ ทั้งภาษาที่มีการเปลี่ยนแปลงอยู่ตลอดเวลา รวมไปถึงลักษณะการเขียนที่ต้องใช้เวลาในการทำความเข้าใจ การสร้าง Flow Programming ขึ้นมาจึงเป็นเครื่องมือที่ช่วยลดความยุ่งยากในการทำความรู้จักภาษาโปรแกรมต่าง ๆ ให้น้อยลง และช่วยให้ผู้ใช้งานที่แม้จะไม่เคยเขียนโปรแกรมใด ๆ มาก่อนก็สามารถเริ่มต้นการใช้งานได้อย่างง่ายดาย

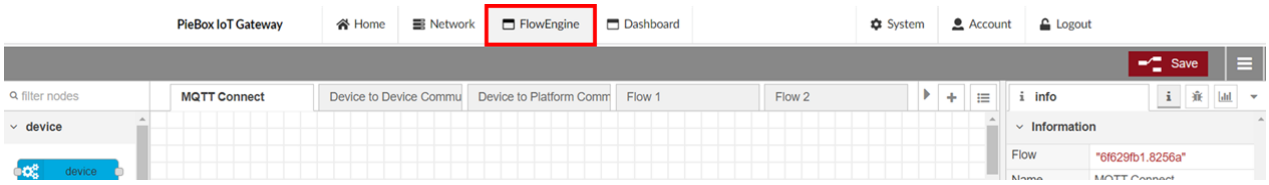
จุดเด่นของ Flow Programming คือการตัดส่วนของการเขียนโปรแกรมด้วยภาษาต่าง ๆ ทิ้งออกไป แล้วแทนที่ส่วนต่าง ๆ ที่ต้องเขียนด้วยโหนดการทำงานแต่ละส่วน เช่น หากต้องการให้โปรแกรมพิมพ์ข้อความใดข้อความหนึ่งออกมา แทนการที่ผู้ใช้งานต้องเขียนโปรแกรมเพื่อให้แสดงผลค่าข้อความนั้น ผู้ใช้งานสามารถเลือกใช้โหนดแสดงผลข้อความมาใช้งานได้ทันที เป็นต้น



Flow Engine คือเครื่องมือในลักษณะของ Flow Programming ที่ทาง NEXPIE ได้นำมาใช้เพื่อให้ผู้ใช้งานสามารถสร้างโปรแกรมที่สามารถติดต่อสื่อสารกับอุปกรณ์ได้ง่ายดายมากขึ้น ผ่านการลากโหนดต่าง ๆ มาเชื่อมต่อกันเพื่อกลายเป็นโฟลว์ขนาดใหญ่โฟลว์หนึ่งที่คล้ายคลึงกับการสร้างผังงาน (Flowchart) ที่ทุกคนน่าจะรู้จักกันเป็นอย่างดี

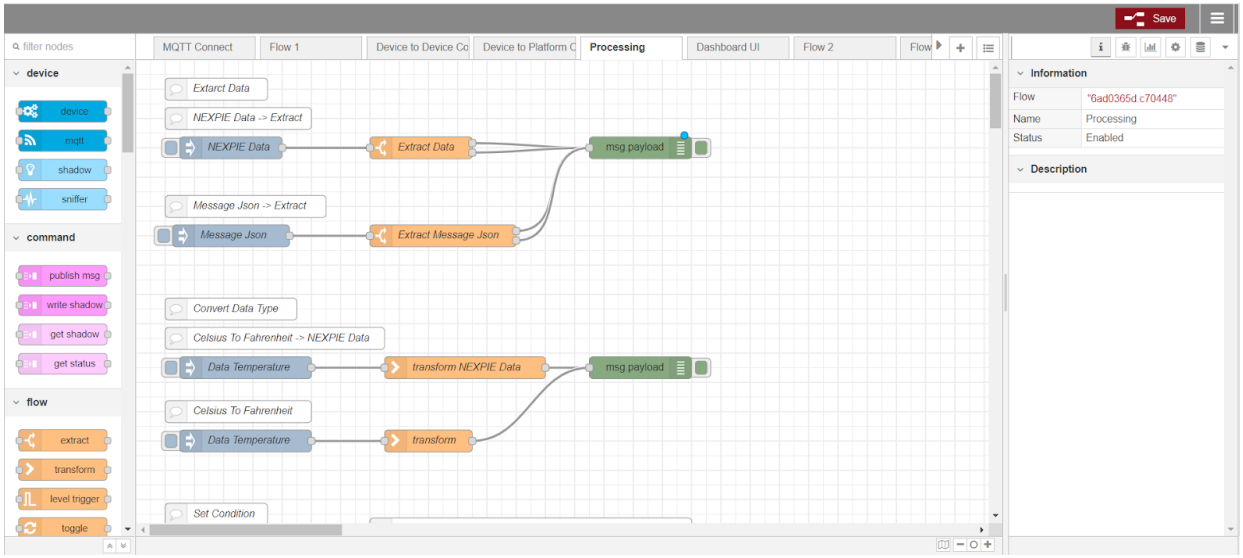
Flow Engine ใน PIEBOX

ในการเข้าใช้งาน Flow Engine ให้ผู้ใช้งานเชื่อมต่อ PIEBOX เข้ากับคอมพิวเตอร์แล้วเข้าไปยัง http://piebox.local โดยการใช้ Username และ Password ที่ทำการลงทะเบียนไว้กับ NEXPIE (โดยเบื้องต้นคือให้กรอกว่า ‘admin’ ทั้ง Username และ Password) จากนั้นจึงเลือกไปยังแถบเมนู Flow Engine เพื่อเข้าสู่โฟลว์



ภาพรวมการสร้างโฟลว์

การสร้างโฟลว์ขึ้นมานั้น มีส่วนประกอบหลัก ๆ 4 อย่างที่ผู้ใช้งานมักจะต้องใช้ตลอดเวลา ได้แก่ แถบเมนูด้านบน (Header) ที่ใช้ในการบันทึกการเปลี่ยนแปลงภายในโฟลว์ และรวมเมนูการตั้งค่าหลักของโฟลว์เอาไว้ แพลิตต์ (Palette) ที่รวบรวมโหนดต่าง ๆ ไว้ทางด้านซ้ายของหน้าต่างโปรแกรม เวิร์คสเปซ (Workspace) ที่ใช้ในการสร้างโฟลว์ขึ้นมา และแถบเมนูข้าง (Sidebar) ทางด้านขวาของโปรแกรมที่ใช้เพื่ออธิบายลักษณะของโฟลว์หรือโหนดต่าง ๆ ภายในโปรแกรม รวมไปถึงการแสดงข้อความและการตั้งค่าการแสดงผลต่าง ๆ ภายในโปรแกรมด้วย



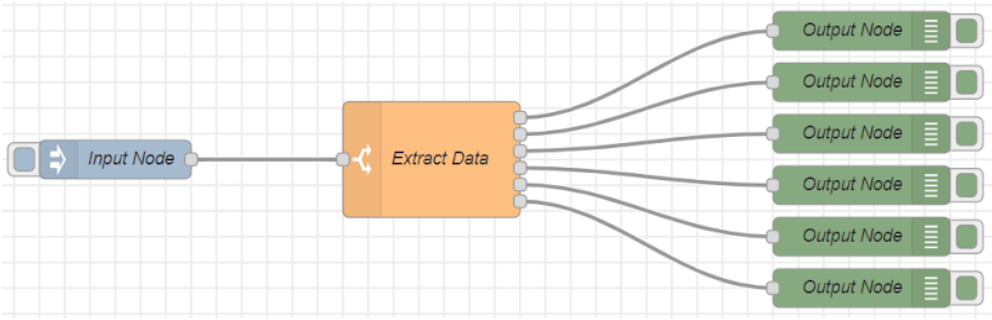
เพื่อให้เข้าใจถึงองค์ประกอบและการใช้งานต่าง ๆ ของ Flow Engine ผู้ใช้งานอาจเริ่มจากการเข้าใจถึงแนวคิดขั้นพื้นฐานในความหมายของสิ่งที่อาจจะถูกพูดถึงในภายหลัง ซึ่งจะช่วยให้ผู้ใช้งานสามารถคุ้นชินกับคำศัพท์และเห็นภาพถึงการใช้งานได้มากขึ้น

โหนด

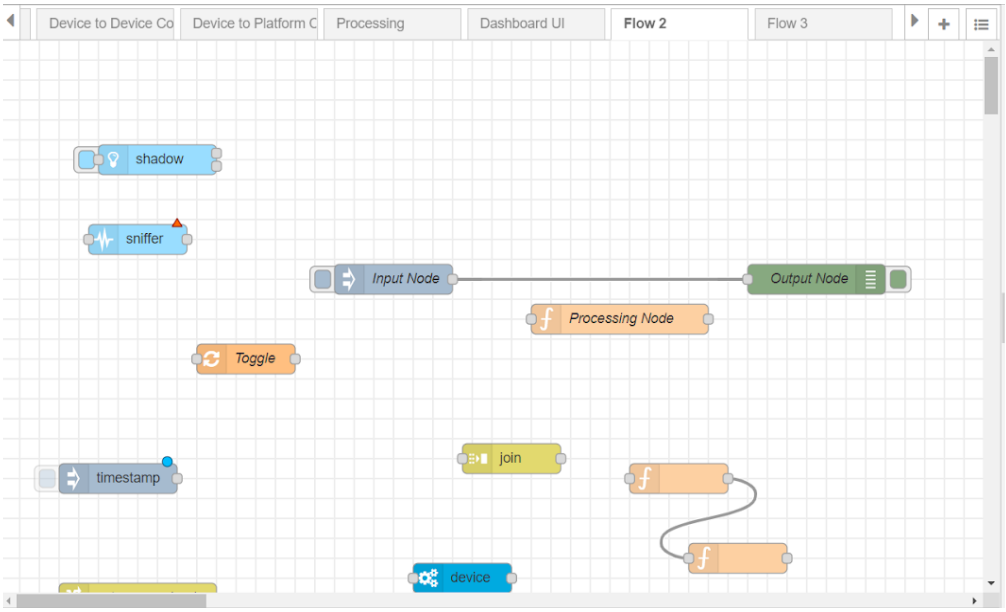
โหนด (Node) เป็นรากฐานการใช้งานพื้นฐานของโฟลว์ ซึ่งโหนดจะมีการกระตุ้นให้เกิดการใช้งานขึ้นเมื่อได้รับข้อความที่ถูกส่งมาจากโหนดก่อนหน้าของโฟลว์ หรือมีเหตุการณ์บางอย่างจากภายนอกมากระตุ้นให้เกิดการเปลี่ยนแปลง เช่น HTTP Request การตั้งเวลา หรือเหตุการณ์อื่น ๆ ซึ่ง

เมื่อเข้ามาในโหนดนั้นแล้ว อาจทำให้เกิดการทำงานบางอย่างและส่งข้อความไปยังโหนดอื่น ๆ ที่เชื่อมต่ออยู่ในโฟลว์เดียวกันได้อีกด้วย

โหนดแต่ละโหนดสามารถมีจุดเชื่อมต่อขาเข้าได้สูงสุดเพียงหนึ่งจุดเท่านั้น แต่สามารถมีจุดเชื่อมต่อขาออกจากโหนดได้มากกว่าหนึ่งจุด



โฟลว์



โฟลว์ (Flow) คือคำที่ใช้อธิบายกลุ่มของโหนดที่เชื่อมต่อกันหนึ่งกลุ่ม ซึ่งภายในหนึ่งโฟลว์ (แถบหน้าโฟลว์) สามารถมีโฟลว์ (กลุ่มการเชื่อมต่อของโหนด) ได้มากกว่าหนึ่งชุด

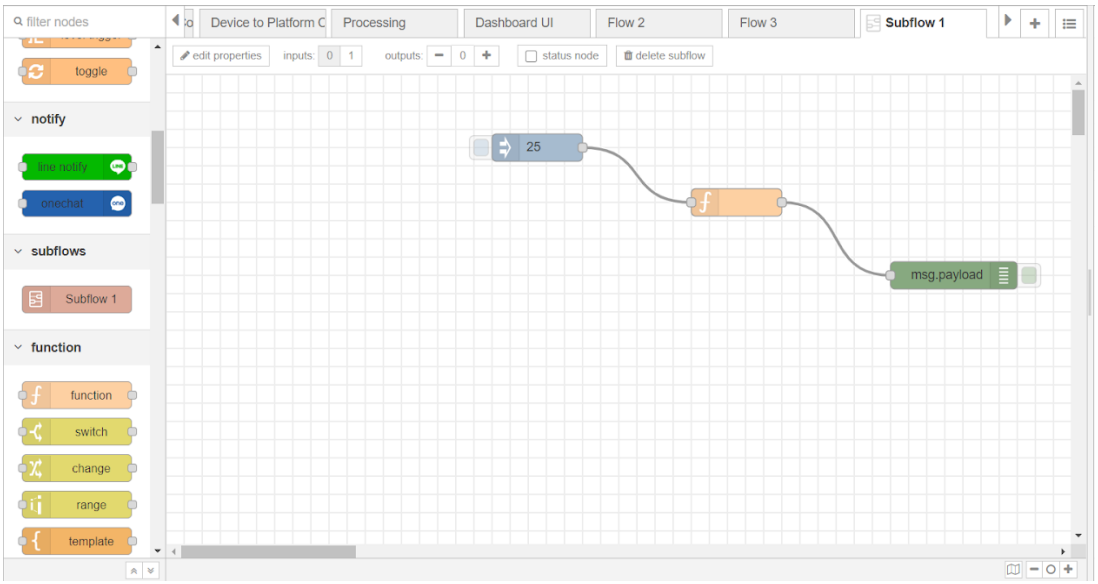
โฟลว์แต่ละโฟลว์จะมีชื่อเรียกเป็นของตัวเอง เช่นเดียวกับคำอธิบายของโฟลว์นั้นซึ่งจะปรากฏขึ้นบนแถบเมนูรายละเอียดด้านขวาของหน้าต่างโปรแกรม

ข้อความ

ข้อความ (Message) จะถูกส่งผ่านกันระหว่างโหนดแต่ละโหนดภายในโฟลว์ โดยจะอยู่ในลักษณะของ JavaScript Object ที่สามารถรวบรวมค่าคุณสมบัติต่าง ๆ ของข้อความนั้นเอาไว้ภายในได้ ซึ่งมักจะใช้ตัวแปร `msg` ในการอ้างอิงถึงข้อความต่าง ๆ ภายในโฟลว์

ซึ่งโดยปกติแล้ว ภายในข้อความเหล่านี้ที่ได้ถูกใช้งานจะมีค่าตัวแปรที่ชื่อว่า `payload` ซึ่งจะทำการรวบรวมข้อมูลสำคัญเอาไว้บรรจุอยู่ภายในด้วยเสมอ

ซับโฟลว์



ซับโฟลว์ (Subflow) คือพื้นที่ที่ใช้รวบรวมส่วนหนึ่งของโฟลว์เอาไว้ให้กลายเป็นโหนดใหม่อีกโหนดหนึ่งที่สามารถนำไปใช้งานได้ในตำแหน่งอื่น ๆ ของเวิร์คสเปซ

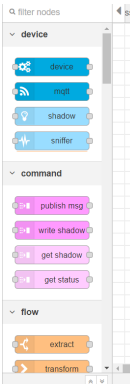
ประโยชน์หลัก ๆ ของการสร้างซับโฟลว์ขึ้นมาในโฟลว์คือจะช่วยลดความยุ่งเหยิงและดูยากของโฟลว์ให้เล็กลง อีกทั้งยังมีประโยชน์ต่อกลุ่มการทำงานของบางโหนดที่อาจสามารถนำไปใช้ซ้ำในตำแหน่งอื่น ๆ ของโฟลว์ได้อีกด้วย

เส้นเชื่อมต่อ

การสร้างโฟลว์นั้นจำเป็นต้องใช้เส้นเชื่อมต่อ (Wire) ในการใช้เชื่อมต่อระหว่างโหนดหลาย ๆ โหนดเข้าด้วยกัน ซึ่งจะช่วยในการอธิบายถึงการไหลของข้อความว่าเดินทางจากโหนดใดไปสู่โหนดใดบ้าง

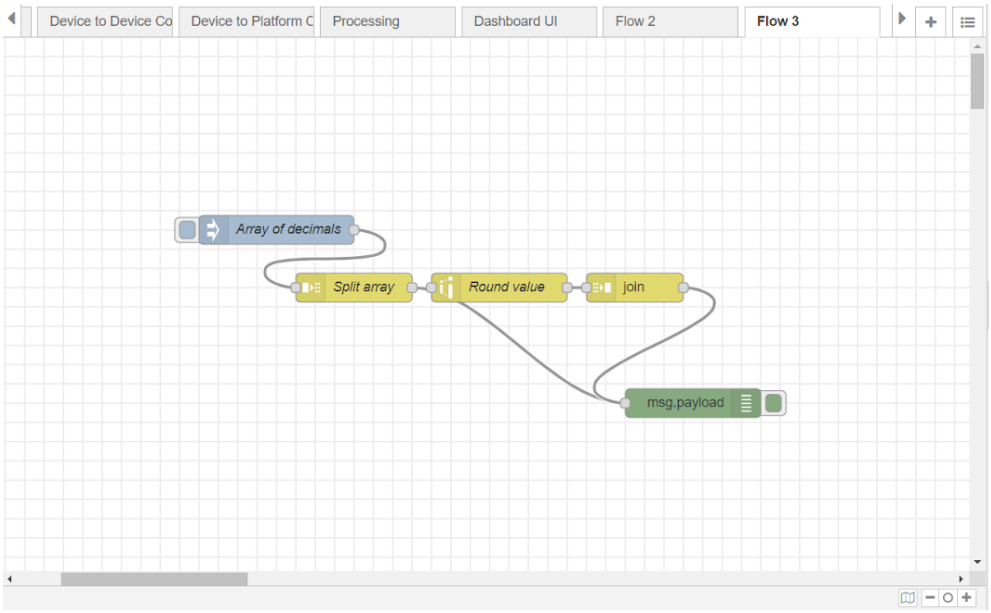
การลากเส้นเชื่อมต่อนี้ ผู้ใช้งานสามารถทำได้โดยการลากจากจุดสีเทาของโหนดหนึ่งไปเชื่อมต่อกับอีกโหนดหนึ่ง โดยมีเงื่อนไขว่า ต้องเชื่อมต่อจุดสีเทาทางด้านขวาของโหนดเข้ากับทางด้านซ้ายของโหนดเท่านั้น

แพลิตต์



แพลิตต์จะปรากฏที่ด้านซ้ายของหน้าต่างโปรแกรม ซึ่งจะรวบรวมโหนดต่าง ๆ ที่สามารถใช้งานได้ภายในโฟลว์นั้น โดยแต่ละโหนดที่อยู่ภายในแพลิตต์จะถูกแบ่งออกเป็นหมวดหมู่ย่อยต่าง ๆ อิงตามประเภทการใช้งานของโหนดเหล่านั้น เช่น การใช้งานทั่วไป (Common) ฟังก์ชัน (Function) การจัดเก็บ (Storage) เป็นต้น ซึ่งจะช่วยให้ง่ายต่อการค้นหาและนำมาใช้งานวางลงบนโฟลว์

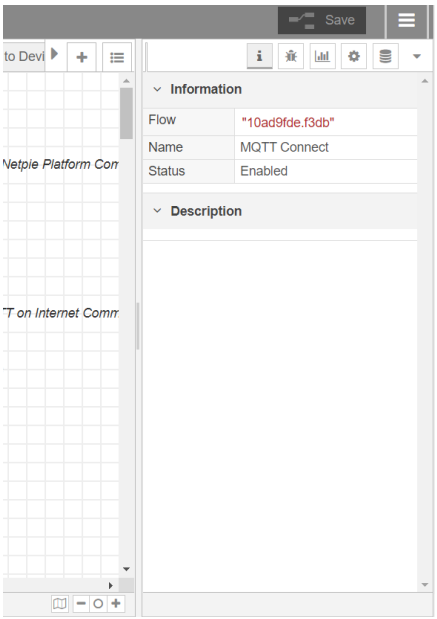
เวิร์คสเปซ



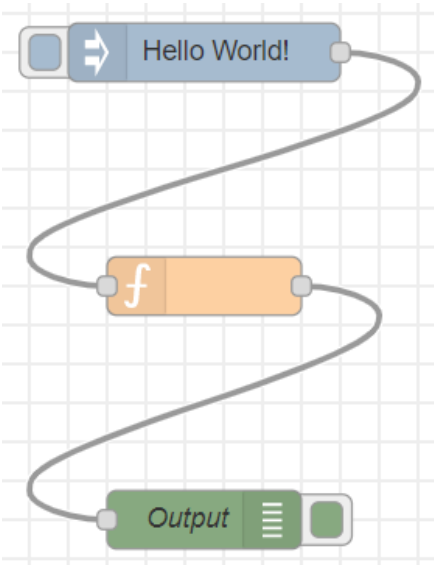
เวิร์คสเปซคือพื้นที่หลักที่โฟลว์จะถูกสร้างขึ้นจากการนำโหนดซึ่งอยู่ภายในแพลิตต์ลงมาวางลงไปแล้วเชื่อมต่อโหนดแต่ละโหนดนั้นเข้าด้วยกัน เวิร์คสเปซจะปรากฏแถบรายชื่อของโฟลว์และซับโฟลว์ที่กำลังเปิดใช้งานอยู่ด้านบน

แถบเมนูข้าง

แถบเมนูข้างจะประกอบไปด้วยเมนูต่าง ๆ ที่ได้ทำการรวบรวมเครื่องมือที่มีประโยชน์ต่อการใช้งานเอาไว้ ซึ่งจะรวมไปถึงการดูข้อมูลต่าง ๆ และความช่วยเหลือในการใช้งานโหนดที่ผู้ใช้งานเลือก ดูข้อความที่ถูกส่งผ่านระหว่างกันของโหนดต่าง ๆ เป็นต้น



โฟลว์



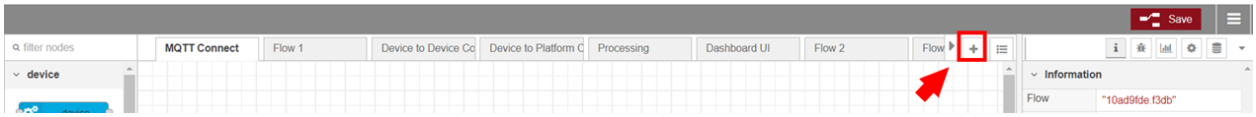
โฟลว์ (Flows) ในลักษณะของ Flow Engine คือพื้นที่ที่อนุญาตให้ผู้ใช้งานเชื่อมต่อการทำงานในแต่ละส่วนเข้าด้วยกันราวกับสายไฟที่เชื่อมต่อระหว่างสวิตช์ไปยังหลอดไฟต่าง ๆ โดยเปรียบโหนด (Nodes) แต่ละโหนดเป็นอุปกรณ์แต่ละตัว และใช้เส้นการเชื่อมต่อ (Wires) แทนสายไฟที่เชื่อมโยงแต่ละอุปกรณ์เข้าด้วยกัน

โฟลว์ คือการประกอบร่างรวมกันระหว่างโหนดแต่ละประเภท ซึ่งลักษณะของโฟลว์ที่ถูกสร้างขึ้นมานั้นจะมีลักษณะคล้ายคลึงกับผังงาน (Flowchart) ที่ใช้โหนดแทนสัญลักษณ์ต่างของผังงาน และใช้เส้นที่เชื่อมโยงระหว่างแต่ละโหนดแทนลูกศรที่ถูกชี้ไประหว่างสัญลักษณ์แต่ละอย่าง ซึ่งจะเป็นการอธิบายว่าเมื่อการทำงานของโหนดนี้สิ้นสุดลงจะส่งผลลัพธ์หรือต้องการให้โหนดโหนดใดทำงานในลำดับถัดไป

สร้างโฟลว์ใหม่

โฟลว์แต่ละผังจะถูกแยกออกจากกันโดยแถบเมนูด้านบนของเวิร์คสเปซ ซึ่งผู้ใช้งานสามารถสร้างโฟลว์เพิ่มเติมได้ ในกรณีที่ต้องการแยกประเภทของโฟลว์แต่ละผังออกจากกันและเพื่อจัดการโหนดต่าง ๆ ให้เป็นระเบียบเรียบร้อยมากขึ้น โดยโฟลว์แต่ละโฟลว์จะมีชื่อและคำอธิบายปรากฏที่แถบเมนูรายละเอียดด้านขวามือ

การสร้างโฟลว์ใหม่ขึ้นมานั้น ผู้ใช้งานสามารถทำได้โดยการคลิกที่ปุ่ม **‘+’** บนแถบรายชื่อโฟลว์ หรือทำการ double-click บนพื้นที่เปล่าที่แถบรายชื่อโฟลว์นั้น

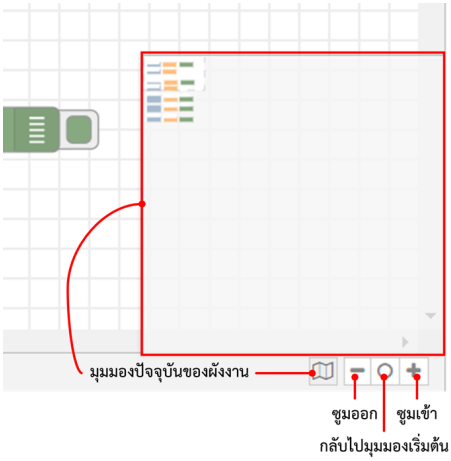


โฟลว์ใหม่ที่ถูกสร้างขึ้นมาจะถูกตั้งชื่อว่า ‘**Flow #**’ ซึ่ง # คือลำดับของโฟลว์ที่ถูกสร้างขึ้นมาล่าสุดที่ยังไม่ได้ทำการตั้งชื่อ ในกรณีที่ผู้ใช้งานต้องการเปลี่ยนชื่อหรือเพิ่มคำอธิบายให้กับโฟลว์นี้ ให้ผู้ใช้งานทำการกด double-click ที่ชื่อของโฟลว์นี้ที่แถบรายชื่อโฟลว์เพื่อเปิดหน้าต่างแก้ไขออกมาทางด้านขวาของหน้าต่างโปรแกรม ซึ่งเมื่อทำการแก้ไขเสร็จเรียบร้อยแล้ว ชื่อใหม่จะปรากฏขึ้นบน

แถบโฟลว์ของโฟลว์นี้ และจะปรากฏทั้งชื่อและคำอธิบายใหม่ที่สร้างขึ้นมาบนแถบเมนูข้อมูลรายละเอียดด้านขวามือของหน้าต่างโปรแกรม

นอกจากคำอธิบายโดยปกติแล้ว ผู้ใช้งานยังสามารถใช้การเขียนในลักษณะ Markdown ลงไปในส่วนของคำอธิบายได้อีกด้วย ผู้ใช้งานสามารถศึกษาวิธีการเขียน Markdown ลงไปได้ผ่านทางเว็บไซต์ https://www.markdownguide.org/basic-syntax/ หรือใช้เครื่องมือที่ปรากฏให้บนแถบเครื่องมือของแถบหน้าต่างแก้ไขนั้นในการเขียนก็ได้เช่นกัน

การปรับมุมมองโฟลว์



ด้านล่างของโฟลว์จะประกอบไปด้วยปุ่มที่ใช้ในการปรับลด-ขยายมุมมองของโฟลว์ได้ เช่นเดียวกับการกลับไปยังค่าเริ่มต้นของการปรับลด-ขยายมุมมอง อีกทั้งยังมีปุ่มที่จะแสดงให้เห็นกรอบมุมมองปัจจุบัน (View Navigator) ของโฟลว์ที่ใช้งานอยู่นี้อีกด้วย

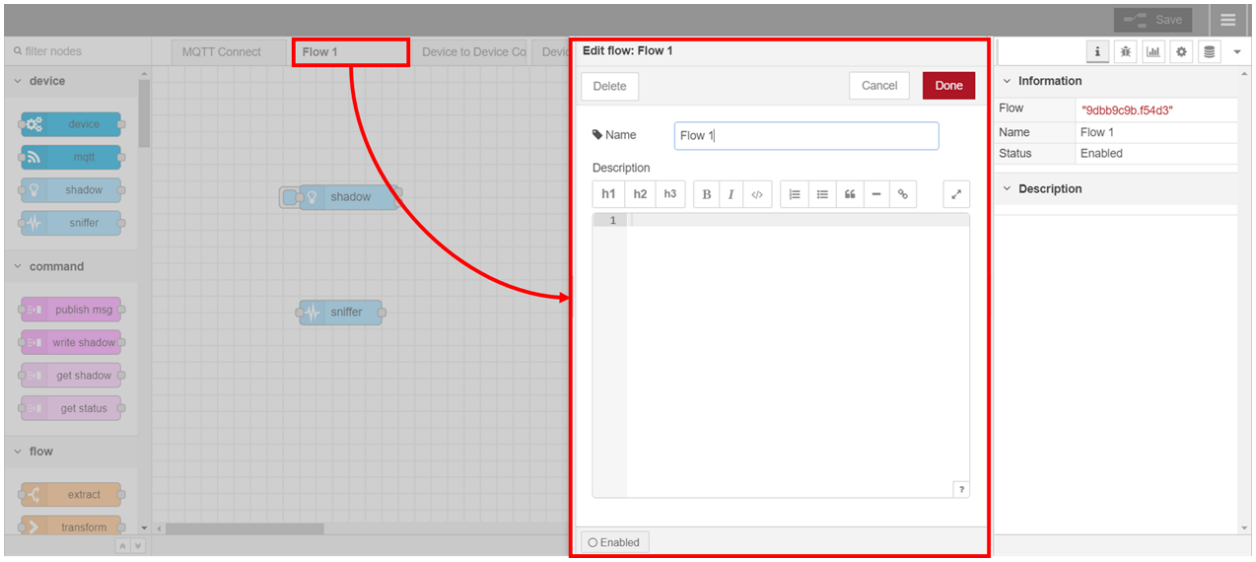
ภาพมุมมองปัจจุบันที่ปรากฏขึ้นมานี้จะช่วยให้ผู้ใช้งานรู้ว่าตอนนี้โฟลว์ปัจจุบันแสดงโฟลว์ในตำแหน่งใดให้เห็นอยู่ ซึ่งจะมีประโยชน์ต่อโฟลว์ขนาดใหญ่ที่ผู้ใช้งานทำการสร้างเอาไว้ โดยจะปรากฏขึ้นเป็นหน้าต่างเล็ก ๆ ทางขวามือด้านล่างของโฟลว์ ซึ่งมุมมองปัจจุบันจะปรากฏภายในกรอบสีขาวในหน้าต่างนั้น

หากผู้ใช้งานต้องการจัดการตั้งค่าเกี่ยวกับมุมมองของ ให้ทำการไปยังหน้าการตั้งค่าโดยกดปุ่ม `Ctrl/⌘` + `,` เพื่อเปิดหน้าต่างการตั้งค่าขึ้นมา ซึ่งภายในหน้าต่างการตั้งค่านั้น ผู้ใช้งานสามารถตั้งค่าได้ตั้งแต่ภาษาที่ใช้งาน ลักษณะของตารางที่ปรากฏในโฟลว์ ลักษณะของโหนดในโฟลว์ รวมไปถึงการปรากฏตัวช่วย (Tips) ขึ้นมาด้วย

การตั้งค่าโฟลว์

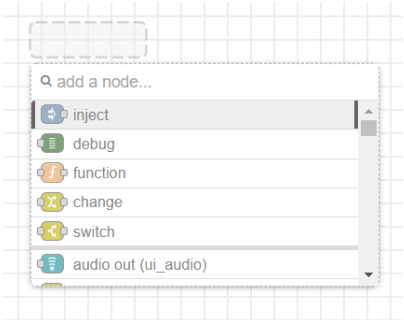
ดังที่กล่าวไปในหัวข้อก่อนหน้า ผู้ใช้งานสามารถแก้ไขชื่อและคำอธิบายของโฟลว์แต่ละหน้าได้โดยการ double-click ที่ชื่อของโฟลว์บนแถบรายชื่อโฟลว์นั้น นอกเหนือจากนั้น ผู้ใช้งานยังสามารถเลือกได้ว่าต้องการให้โหนดต่าง ๆ ภายในโฟลว์ทั้งหมดนั้นทำงานหรือไม่ โดยคลิกที่ปุ่ม ‘Enabled/Disabled’ ที่อยู่ด้านล่างของแถบเมนูแก้ไขทางด้านขวาเพื่อเปิด/ปิดการใช้งานของโฟลว์

นั้น ซึ่งการปิดการใช้งานโฟลว์จะไม่ทำการลบสิ่งที่ผู้ใช้งานสร้างเอาไว้ภายในโฟลว์ แต่จะเป็นการไม่ทำให้โหนดต่าง ๆ เหล่านั้นที่ถูกสร้างเอาไว้ได้ออกมาแสดงผลให้ผู้ใช้งานเห็น



หากผู้ใช้งานต้องการลบโฟลว์นั้นทิ้ง ให้ทำการคลิกที่ปุ่ม ‘Delete’ ด้านซ้ายบนของแถบเมนูแก้ไขเพื่อลบโฟลว์นั้นทิ้ง

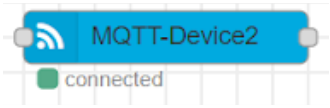
โหนด



เพื่อสร้างโฟลว์ขึ้นมาภายในโฟลว์ ผู้ใช้งานจำเป็นต้องนำโหนดซึ่งอยู่ภายในแพลิตต์ทางด้านซ้ายของโปรแกรมลงมาวางในโฟลว์ ซึ่งผู้ใช้งานสามารถลากโหนดเหล่านั้นลงมาได้โดยตรง หรือทำการกดปุ่ม `Ctrl/⌘` ค้างไว้แล้วทำการ double-click บนเวิร์คสเปซซึ่งจะปรากฏตัวเลือกโหนด (Quick-Add Dialog) ขึ้นมาให้ผู้ใช้งานสามารถเลือกโหนดที่จะปรากฏบนโฟลว์นี้ได้ทันที โดยจะปรากฏโหนดหลัก 5 ประเภท ได้แก่ Inject, Debug, Function, Change, และ Switch อยู่ด้านบนสุดของตัวเลือกนี้ ตามด้วยโหนดที่ถูกเพิ่มไปในโฟลว์ล่าสุด และปิดท้ายด้วยโหนดอื่น ๆ ที่เหลือ เรียงตามลำดับตัวอักษร

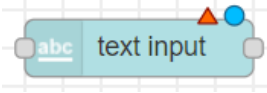
โหนดแต่ละโหนดจะถูกเชื่อมเข้าด้วยกันจากการลากเส้นเชื่อมต่อจากจุดเชื่อมของโหนดเหล่านั้น ซึ่งโหนดแต่ละโหนดจะมีจุดเชื่อมขาเข้าไปในโหนดได้เพียงจุดเดียว แต่สามารถส่งข้อมูลออกจากโหนดนั้นเป็นหลายจุดการเชื่อมต่อได้ โดยหาก

ผู้ใช้งานนำเมาส์ไปลอยเหนือจุดเชื่อมต่อขาออกของแต่ละโหนด จุดเชื่อมต่อนั้นจะปรากฏข้อความช่วยเหลือเพื่อแสดงว่าค่าใดที่จะกลายเป็นผลลัพธ์ออกไปจากจุดการเชื่อมต่อของโหนดนั้น



บางโหนดอาจปรากฏข้อความสถานะอยู่ด้านล่างเพื่อใช้ในการบอกถึงสถานะปัจจุบันของโหนดนั้นว่ายังมีการเชื่อมต่ออยู่หรือไม่ เช่น อาจใช้ประโยชน์ในกรณีที่ผู้ใช้งานต้องการทราบสถานะการเชื่อมต่อปัจจุบันของ MQTT ว่ายังเชื่อมต่ออยู่หรือไม่ในโหนด **‘MQTT’** เป็นต้น

ในกรณีที่โหนดนั้นมีการเปลี่ยนแปลงเกิดขึ้นและยังไม่ได้ทำการบันทึกการเปลี่ยนแปลงที่เกิดขึ้นกับโฟลว์ โหนดจะปรากฏวงกลมสีฟ้าขึ้นเล็ก ๆ ที่ด้านบนของโหนด ในขณะที่โหนดที่เกิดความผิดพลาดในการตั้งค่าภายในจะปรากฏเป็นสามเหลี่ยมสีแดงขึ้นบนโหนดแทน



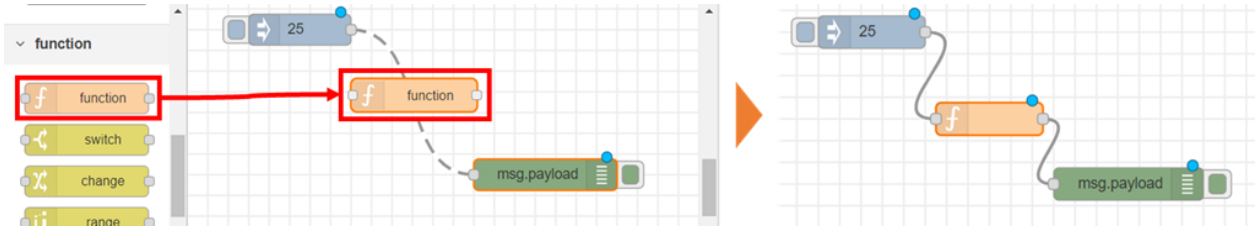
โหนดบางโหนดอาจปรากฏปุ่มขึ้นที่ด้านซ้ายหรือขวาของโหนดนั้นเพื่อใช้ในการมีปฏิสัมพันธ์กับโหนดนั้นได้โดยตรง โดยโหนดหลักที่มีปุ่มการเหล่านี้ปรากฏขึ้น ได้แก่โหนด Inject และ Debug นั่นเอง

เส้นเชื่อมต่อ

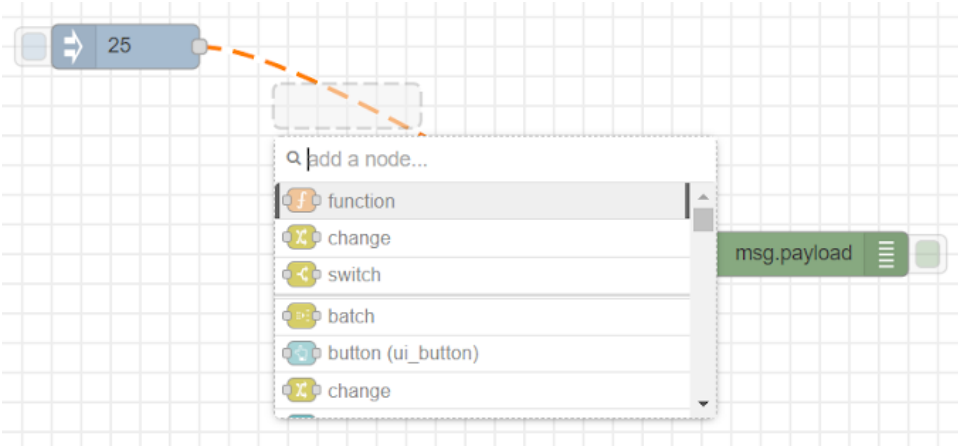
โหนดแต่ละโหนดจะถูกเชื่อมเข้าด้วยกันโดยการคลิกเมาส์ซ้ายค้างบนจุดเชื่อมต่อของโหนดหนึ่งแล้วลากไปโยงกับอีกโหนดหนึ่งแล้วปล่อยเมาส์ การทำเช่นนี้จะทำให้ปรากฏเส้นเชื่อมต่อ (Wire) ขึ้นระหว่างโหนดที่เลือกทั้งสองโหนดนั้น โดยมีกฎเพียงข้อเดียวว่า เส้นที่ลากออกจากโหนดนั้นต้องลากออกจากทางด้านขวาของโหนด และลากเข้าทางด้านซ้ายของอีกโหนดหนึ่งเท่านั้น

อีกวิธีหนึ่งที่สามารถทำได้คือการคลิกที่ปุ่ม `Ctrl`/`Command` ค้างบนแป้นคีย์บอร์ดแล้วทำการนำเมาส์ซ้ายไปคลิกบนจุดการเชื่อมต่อทางออกของโหนดหนึ่ง (โดยไม่ต้องลากเส้นออกมา) จากนั้นผู้ใช้งานสามารถทำการคลิกที่จุดเชื่อมต่อบนอีกโหนดหนึ่งได้ทันที ซึ่งจะปรากฏเป็นเส้นเชื่อมต่อเช่นเดียวกับการลากเชื่อมธรรมดา แต่การทำเช่นนี้จะอนุญาตให้ผู้ใช้งานสามารถลากเส้นเชื่อมต่อเพิ่มเติมจากโหนดล่าสุดที่คลิกไป เพื่อลากไปยังโหนดอื่น ๆ อีกได้เช่นกัน ซึ่งจะมีประโยชน์ในกรณีที่ผู้ใช้งานต้องการเชื่อมต่อโหนดจำนวนมากเข้าด้วยกันเป็นโฟลว์ขนาดใหญ่โฟลว์หนึ่ง

ยิ่งไปกว่านั้น หากผู้ใช้งานนำโหนดที่สามารถทำหน้าที่ได้ทั้งรับและส่งค่าข้อมูล (โหนดที่มีจุดสีเทาอยู่ทั้งสองข้างของโหนด) มาวางลงไปบนเส้นเชื่อมต่อที่เชื่อมระหว่างโหนดอื่น ๆ สองโหนดอยู่ เส้นเชื่อมต่อจะกลายเป็นเส้นประ และเมื่อผู้ใช้งานปล่อยเมาส์ เส้นเชื่อมต่อนั้นจะเชื่อมโหนดนั้นล่าสุดเข้ากับอีกสองโหนดที่เหลือทันทีโดยที่ผู้ใช้งานไม่จำเป็นต้องลากการเชื่อมต่อใหม่อีกต่อไป



อีกกรณีที่สามารถทำได้คือ หากผู้ใช้งานกดปุ่ม `Ctrl` ค้างเอาไว้แล้วนำเมาส์ซ้ายไปคลิกบนเส้นเชื่อมต่อที่ถูกลากเชื่อมระหว่างโหนดสองโหนด เส้นเชื่อมต่อนั้นจะกลายเป็นเส้นประและปรากฏตัวเลือกโหนดขึ้นมาให้ผู้ใช้งานสามารถเลือกเพื่อนำโหนดนั้นมาคั่นกลางระหว่างสองโหนดที่มีเส้นเชื่อมต่อนั้นเชื่อมถึงกันได้ทันที

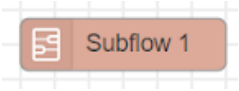


ในกรณีที่ผู้ใช้งานต้องการเปลี่ยนตำแหน่งของการเชื่อมต่อของเส้นเชื่อมต่อ ให้ผู้ใช้งานทำการกดปุ่ม `Shift` บนแป้นคีย์บอร์ดค้างเอาไว้ จากนำเมาส์ซ้ายไปคลิกที่เส้นบนจุดการเชื่อมต่อที่ต้องการย้ายสายการเชื่อมต่อค้างไว้เพื่อลากเส้นนั้นไปต่อเชื่อมยังโหนดตำแหน่งอื่นในโฟลว์ แต่หากผู้ใช้งานปล่อยสายนั้นออกไปก่อนที่จะได้เชื่อมต่อกับจุดเชื่อมใหม่ เส้นเชื่อมต่อนั้นจะถูกลบทิ้งไปโดยปริยาย

การลบเส้นเชื่อมต่อทิ้งนั้นสามารถทำได้อีกวิธี คือการคลิกที่เส้นเชื่อมต่อที่ต้องการลบทิ้ง จากนั้นคลิกที่ปุ่ม `←Backspace` หรือ `Del` บนแป้นคีย์บอร์ดนั้นทิ้งไปก็ทำได้เช่นกัน

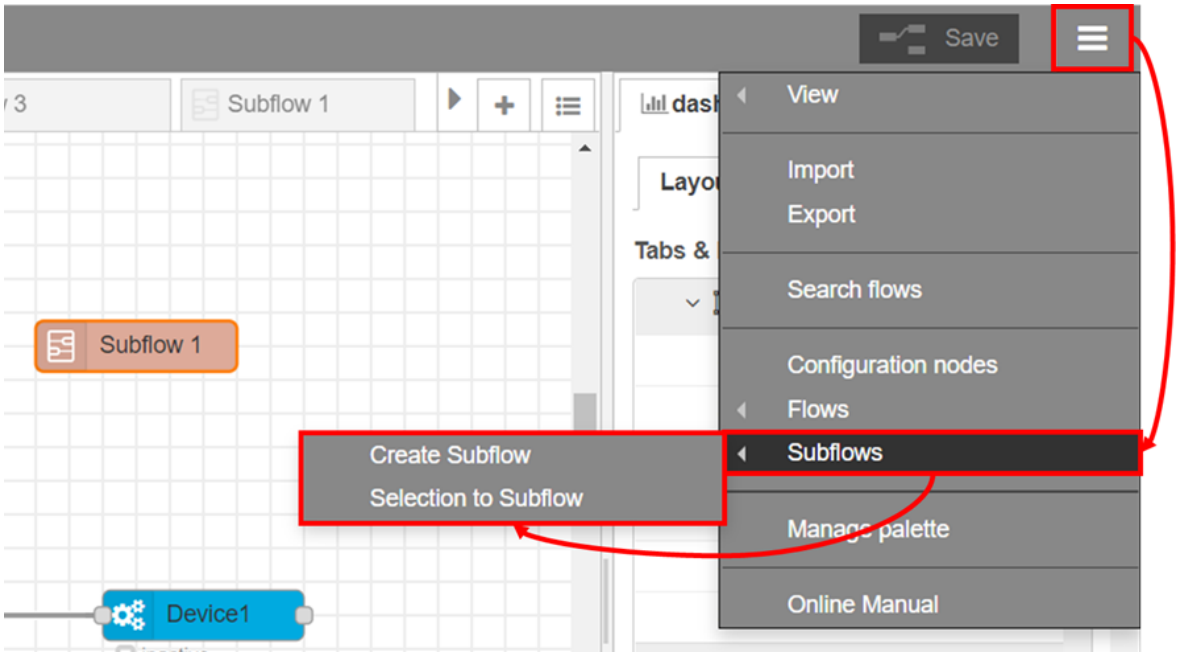
ซับโฟลว์

ในบางครั้ง การสร้างโฟลว์ขนาดใหญ่โฟลว์หนึ่งอาจประกอบไปด้วยส่วนของการทำงานในหลาย ๆ ส่วนที่ถูกเชื่อมต่ออยู่ภายในโฟลว์เดียวกัน บางโฟลว์อาจเป็นโฟลว์ที่ถูกใช้อยู่บ่อยครั้ง หรือบางโฟลว์นั้นทำให้โฟลว์โดยรวมดูใหญ่จนยุ่งเหยิงและยากต่อการจัดการ การสร้างซับโฟลว์ (Subflows) ขึ้นมาจึงจะช่วยสร้างกลุ่มให้แก่โหนดต่าง ๆ ภายในโฟลว์ให้อยู่รวมกันได้ การทำเช่นนี้จะลดความยุ่งเหยิงของการเชื่อมต่อลง อีกทั้งยังมีประโยชน์สำหรับกลุ่มของโหนดที่อาจจะนำไปใช้ในตำแหน่งอื่น ๆ ของโฟลว์อีกด้วย



การสร้างซับโฟลว์ขึ้นมาจะเปรียบเสมือนการสร้างโหนดใหม่ขึ้นมาด้วย ซึ่งเมื่อการสร้างซับโฟลว์นี้ถูกสร้างได้สำเร็จ ซับโฟลว์นี้จะปรากฏเป็นอีกโหนดหนึ่งบนแพลิตต์ด้านซ้ายมือของหน้าต่างโปรแกรม อย่างไรก็ดี การสร้างซับโฟลว์นั้นไม่สามารถสร้างโดยมีซับโฟลว์ตัวเองอยู่ภายในนั้นได้ ไม่ว่ากรณีใดก็ตาม

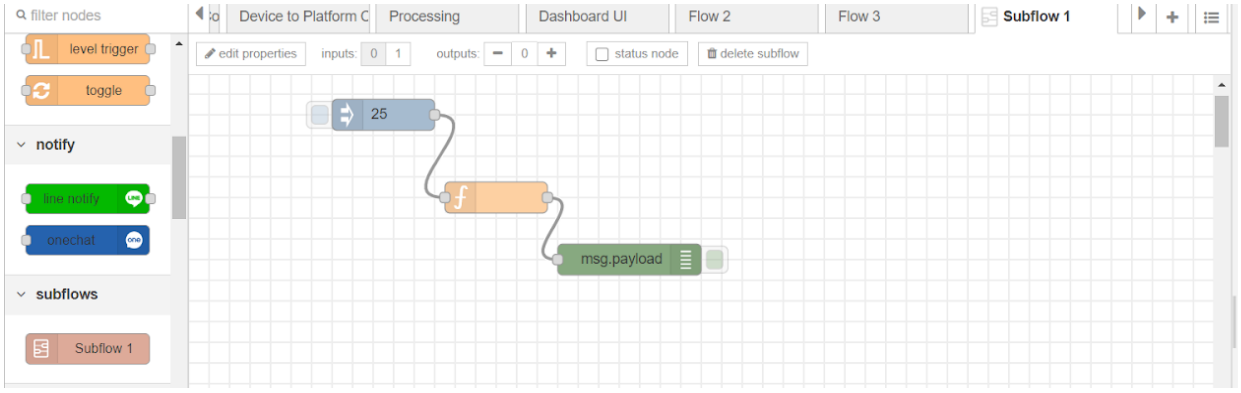
การสร้างซับโฟลว์สามารถทำได้ 2 วิธี



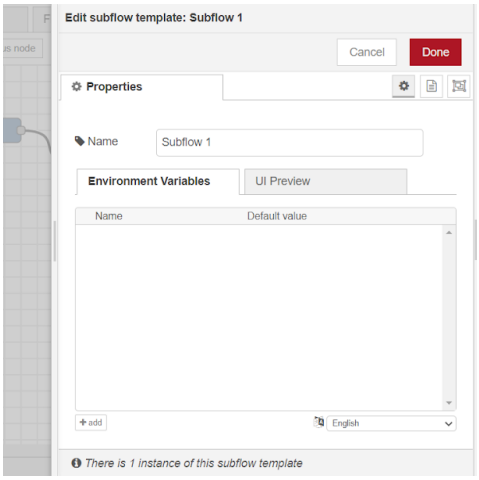
1. การสร้างซับโฟลว์ใหม่ขึ้นมา ให้ทำการไปยังเมนู **‘Subflows’** บนแถบเมนูด้านบน แล้วเลือก **‘Create Subflow’** ซึ่งโปรแกรมจะปรากฏหน้าต่างซับโฟลว์ใหม่ขึ้นมาให้ผู้ใช้งานสามารถสร้างซับโฟลว์ภายในหน้านั้นได้

2. การสร้างซับโฟลว์จากโหนดที่เลือกไว้ในหน้าโฟลว์ ให้ผู้ใช้งานคลิกเมาส์ซ้ายแล้วลากคลุมโหนดที่ต้องการสร้างเป็นซับโฟลว์ จากนั้นไปยังเมนู **‘Subflows’** บนแถบเมนูด้านบนแล้วเลือก **‘Selection to Subflow’** เพื่อเปลี่ยนกลุ่มของโหนดที่เลือกให้กลายเป็นซับโฟลว์ใหม่ทันที ซึ่งการเช่นนี้จะสามารถทำได้ในกรณีที่เส้นเชื่อมต่อขาเข้าไปในซับโฟลว์นั้นเชื่อมต่อไปยังโหนดเพียงโหนดเดียวเท่านั้น

เมื่อซับโฟลว์ถูกสร้างขึ้นมาแล้ว ในกรณีที่ผู้ใช้งานต้องการแก้ไขโฟลว์ภายในหน้าโฟลว์นั้น ผู้ใช้งานสามารถทำได้โดยการ double-click ที่โหนดของซับโฟลว์ที่ต้องการแก้ไขในแพลิตต์ หรือคลิกที่ปุ่ม ‘Edit flow template’ เมื่อทำการ double-click ที่โหนดของซับโฟลว์นี้ที่วางอยู่บนหน้าโฟลว์ ซึ่งจะทำการเปิดโฟลว์ใหม่ขึ้นมาในลักษณะเดียวกันกับโฟลว์ทั่ว ๆ ไป นั่นหมายความว่า ผู้ใช้งานสามารถแก้ไขหรือใช้งานซับโฟลว์นั้นได้เช่นเดียวกับโฟลว์ปกติอื่น ๆ และสามารถปิดแถบซับโฟลว์นั้นเพื่อซ่อนซับโฟลว์ลงไปได้เช่นกัน



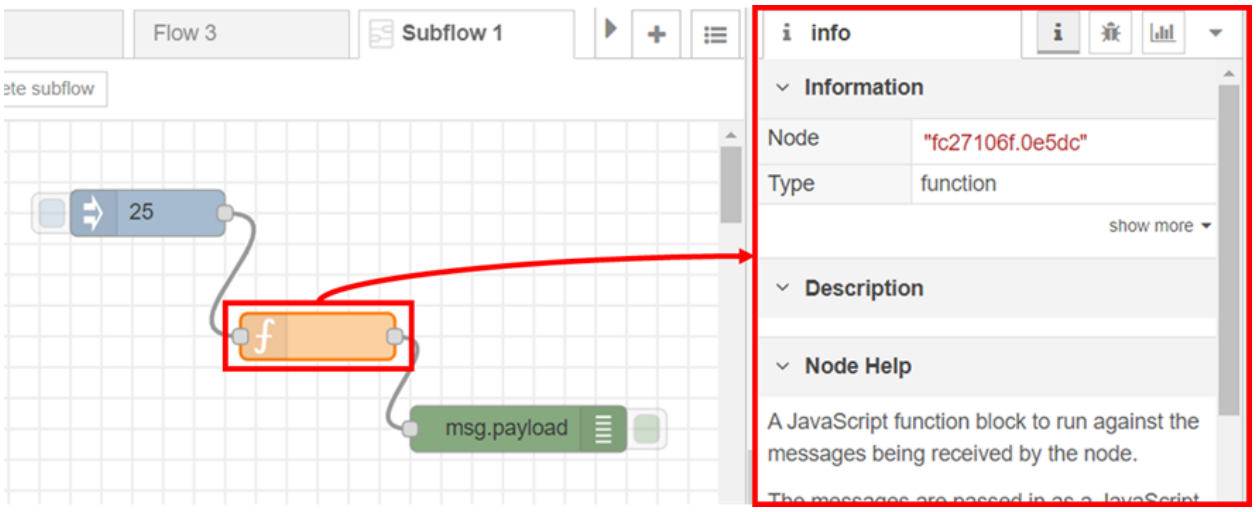
ลักษณะของโหนดแบบซับโฟลว์เป็นเช่นเดียวกับโหนดทั่ว ๆ ไป คือจะปรากฏจุดการเชื่อมต่อสีเทาอยู่ด้านซ้ายและขวาของโหนดตามกฎเดิมเช่นกัน (มีจุดขาเข้าได้เพียงจุดเดียว แต่สามารถมีจุดการเชื่อมต่อขาออกได้มากกว่าหนึ่งจุด อีกทั้งยังสามารถแก้ไขข้อมูลทั้งชื่อ คำอธิบาย สัญลักษณ์ หรือสีก็ได้เช่นกัน



อย่างไรก็ดี สิ่งหนึ่งที่แตกต่างจากโหนดปกติคือการปรากฏแถบแก้ไขส่วน Environment Varibale ขึ้นมาเพื่อให้ผู้ใช้งานสามารถสร้างตัวแปรเพิ่มเติมขึ้นมาใช้งานภายในโฟลว์ได้ โดยตัวแปรเหล่านี้จะเป็นตัวแปรที่เฉพาะโหนดที่ถูกใช้งานภายในซับโฟลว์นี้เท่านั้นจึงจะสามารถใช้ค่าตัวแปรที่สร้างมาเหล่านั้นได้

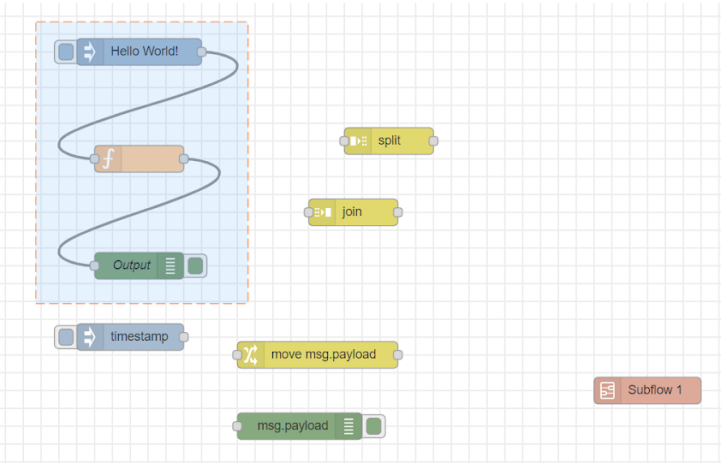
การเลือกโหนด

ในการเลือกโหนดที่วางอยู่แล้วในโฟลว์ ผู้ใช้งานสามารถคลิกลงไปบนโหนดนั้นได้โดยตรง เช่นเดียวกับการยกเลิกการเลือกโหนดนั้น โดยโหนดที่ถูกเลือกจะถูกล้อมไว้ด้วยกรอบสีส้ม และจะปรากฏรายละเอียดขึ้นในแถบเมนูข้างทางด้านขวาของหน้าต่างโปรแกรม



หากปุ่ม `Ctrl/⌘` ถูกกดค้างเอาไว้บนแป้นคีย์บอร์ดในขณะที่กดบนโหนดต่าง ๆ จะเป็นการเพิ่มโหนดภายในโฟลว์นั้นเข้าไปอยู่ในชุดของโหนดที่เลือกทีละโหนด แต่หากเป็น `Shift` ที่ถูกกดค้างไว้บนแป้นคีย์บอร์ด โหนดที่เลือกและทุกโหนดที่เชื่อมอยู่กับโหนดที่เลือกนั้นจะถูกเพิ่มเข้าไปอยู่ในชุดของโหนดที่เลือก

หรืออีกกรณีคือให้ผู้ใช้งานทำการใช้เมาส์ลากคลุมโหนดที่ต้องการเลือกได้โดยตรงผ่านเครื่องมือการเลือกแบบอิสระ (Lasso Tool) ซึ่งจะช่วยให้ผู้ใช้งานสามารถเลือกโหนดหลาย ๆ โหนดได้ภายในการลากคลุมเพียงครั้งเดียว

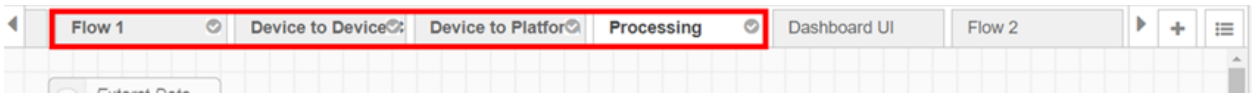


ในทางกลับกัน เส้นเชื่อมต่อระหว่างแต่ละโหนดจะสามารถถูกเลือกได้เพียงทีละเส้นเท่านั้น

โหนดที่ถูกเลือกต่าง ๆ สามารถคัดลอก ตัด และวางบนโฟลว์ต่าง ๆ ได้ โดยใช้คลิปบอร์ดภายใน (Internal Clipboard) ในการวางโหนดที่ถูกคัดลอกหรือตัด

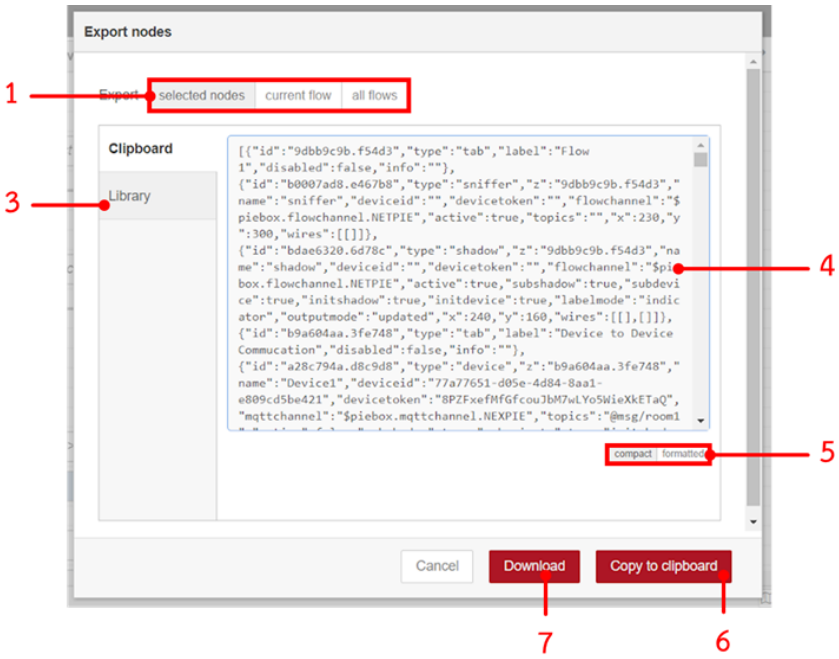
การเลือกโฟลว์

ผู้ใช้งานสามารถเลือกโฟลว์หลาย ๆ หน้าได้ด้วยการกดปุ่ม `Ctrl` หรือ `Command` ค้างไว้แล้วกดเลือกโฟลว์ที่ต้องการบนแถบโฟลว์ด้านบนได้เช่นเดียวกับการเลือกโหนด ซึ่งโฟลว์ที่ถูกเลือกด้วยวิธีนี้จะปรากฏสัญลักษณ์เครื่องหมายถูกขึ้นมาบนแถบชื่อของโฟลว์นั้น นอกเหนือจากนี้ ผู้ใช้งานยังสามารถคัดลอก ตัด และวางหน้าโฟลว์เหล่านี้ได้เช่นเดียวกับการเลือกโหนด



การส่งออกโฟลว์

ผู้ใช้งานสามารถนำโฟลว์ที่ถูกสร้างเอาไว้ออกไปวางยังโฟลว์ของผู้ใช้งานคนอื่น ๆ ได้ในรูปแบบของไฟล์ JSON โดยให้ทำการไปเมนูด้านขวามือบนแถบเมนูด้านบนแล้วคลิกเลือกที่ **‘Export’** ซึ่งผู้ใช้งานสามารถเลือกที่จะส่งออกโฟลว์หมดทั้งหน้า หรือส่งออกเฉพาะโฟลว์ที่ทำการเลือกไว้ก็ได้เช่นกัน



1. เลือกกลุ่มของโหนดที่ต้องการส่งออกจากโฟลว์นี้ โดยสามารถเลือกได้สามลักษณะ ได้แก่
    a. กลุ่มของโหนดที่ถูกเลือกภายในโฟลว์ (Selected nodes)
    b. หน้าของโฟลว์ปัจจุบันที่ทำการเปิดใช้งานอยู่ (Current flow)
    c. ทุกโฟลว์ที่อยู่ภายในโปรแกรมนี้ (All flows)
2. ผู้ใช้งานสามารถเลือกที่จะคัดลอกค่าของหน้าโฟลว์ที่ต้องการส่งออกนี้ให้ไปวางไว้ในหน้าคลิปบอร์ด (Clipboard) ได้
3. ผู้ใช้งานสามารถเลือกจัดเก็บโฟลว์ทั้งหมดให้อยู่ในลักษณะของ library ที่สามารถนำไปใช้งานในภายหลังได้ ซึ่ง library ที่ถูกจัดเก็บนั้นจะถูกใช้งานอยู่ภายในโฟลว์นั้นและถูกจัดเก็บในลักษณะของไฟล์ JSON
4. หน้าต่างนี้จะปรากฏค่า JSON ของโฟลว์ที่ผู้ใช้งานต้องการส่งออกขึ้นมา
5. ผู้ใช้งานสามารถจัดลักษณะของการแสดงผลได้ว่าต้องการให้แสดงออกมาในลักษณะใด ระหว่างย่อค่าข้อมูลทั้งหมดเข้าด้วยกัน (Compact) หรือจัดรูปให้อยู่ในลักษณะที่ดูง่าย (Formatted)

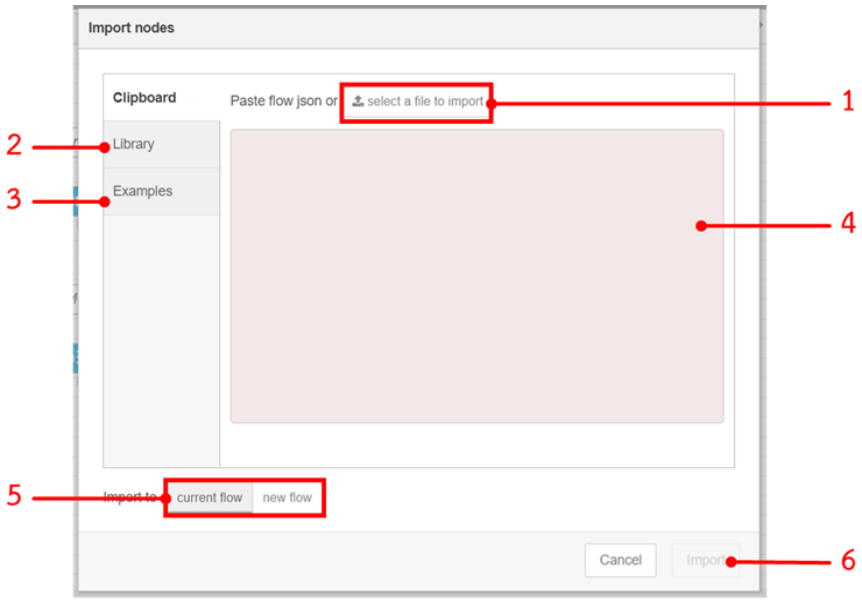
6. การคัดลอกค่าข้อมูล JSON ที่ปรากฏข้างต้นไปวางลงบนคลิปบอร์ด
7. ผู้ใช้งานสามารถดาวน์โหลดค่าข้อมูลของโฟลว์ให้ออกมาเป็นไฟล์ JSON ได้โดยตรง

การนำเข้าโฟลว์

ผู้ใช้งานสามารถเพิ่มโฟลว์เข้ามายังโฟลว์ปัจจุบันได้ ไม่ว่าจะเป็นการวางค่า JSON ที่ได้รับมาลงไปโดยตรง การเพิ่มไฟล์ JSON ที่ผู้ใช้งานมีอยู่เข้าไปในโฟลว์ การใช้งานจาก Flow library ที่อยู่ในพื้นที่ปัจจุบัน หรือการใช้งานโฟลว์ตัวอย่างที่จัดการโดยโหนดที่ถูกติดตั้งเอาไว้ และไม่ว่าจะนำเข้าโฟลว์ด้วยวิธีใด ผู้ใช้งานสามารถเลือกได้ว่าจะนำโฟลว์ที่นำเข้ามานั้นวางลงในโฟลว์ปัจจุบันหรือสร้าง

เป็นโฟลว์ใหม่ก็ได้เช่นกัน

การนำเข้าโฟลว์นั้น ให้ผู้ใช้งานไปที่แถบเมนูด้านขวาบนแถบเมนูด้านบนแล้วคลิกเลือกที่ **‘Import’** เพื่อนำเข้าโฟลว์ที่ทำการส่งออก (Export) มาจากโฟลว์อื่น ๆ



1. ผู้ใช้งานสามารถนำไฟล์ JSON ของโหนดที่ถูกส่งออกมาก่อนหน้านี้มาวางลงไปในหน้าโฟลว์นี้เพื่อทำการนำเข้าโฟลว์นั้นเข้ามาในโปรแกรมได้
2. หากผู้ใช้งานทำการบันทึกค่าของโฟลว์นั้นใน library ผู้ใช้งานสามารถใช้งานมันได้โดยการเปิดแถบ library ขึ้นมาแล้วเลือกโฟลว์ที่ต้องการ
3. ในบางครั้ง อาจมีโฟลว์ตัวอย่างที่ถูกสร้างเอาไว้เพื่อเป็นแบบแปลนในการสร้างโฟลว์อื่น ๆ ที่เกี่ยวข้องกับโฟลว์ตัวอย่างนั้นอยู่ด้วย
4. ผู้ใช้งานสามารถวาง (Paste) ค่า JSON ที่ทำการคัดลอกลงบนคลิปบอร์ดไว้ก่อนหน้านี้ลงไปในพื้นที่ตรงกลางได้โดยตรง
5. ผู้ใช้งานสามารถเลือกได้ว่าต้องการจะนำโฟลว์ที่นำเข้ามานั้นใส่ลงไปในหน้าโฟลว์ที่กำลังเปิดใช้งานอยู่ (Current flow) หรือสร้างเป็นโฟลว์ใหม่อีกโฟลว์หนึ่งไปเลย (New flow)
6. เมื่อทำการเลือกโฟลว์ที่ต้องการจะนำเข้าและตั้งค่าการนำเข้าของโฟลว์นั้นไว้เรียบร้อยแล้ว ให้ผู้ใช้งานทำการคลิกที่ปุ่ม ‘Import’ เพื่อนำเข้าโฟลว์นั้น โดยโฟลว์ที่นำเข้าจะเคลื่อนที่ตามตำแหน่งที่เมาส์ขยับไป จนกว่าผู้ใช้งานจะคลิกลงบนเวิร์คสเปซในตำแหน่งที่ต้องการนำโฟลว์นั้นไปวางลง

ในเอกสารฉบับนี้ได้ทำการเขียนตัวอย่างของโฟลว์บางส่วนเอาไว้ให้ผู้ใช้งานสามารถคัดลอกไปวางในโฟลว์ของผู้ใช้งานเองได้ โดยจะปรากฏในลักษณะของค่า JSON ที่ถูกวางเอาไว้ในกรอบสี่เหลี่ยมที่ล้อมรอบด้วยเส้นประ (⬚) เช่น

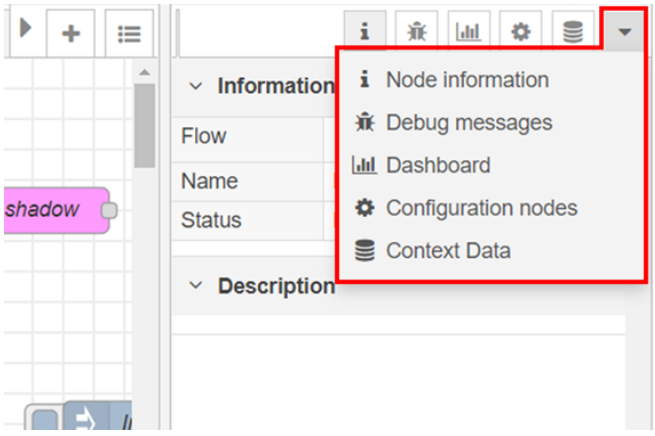
```
[{"id":"4f19b295.ce21dc","type":"inject","z":"8cfb1028.a0d6d","name":"","topic":"","payload":"Hello World!","payloadType":"str","repeat":"","crontab":"","once":false,"onceDelay":0.1,"x":370,"y":960,"wires":[["67bbe1fd.ec7f1"]]},{"id":"67bbe1fd.ec7f1","type":"debug","z":"8cfb1028.a0d6d","name":"","active":true,"tosidebar":true,"console":false,"tostatus":false,"complete":"payload","targetType":"msg","x":590,"y":960,"wires":[]}]
```

โค้ดด้านบนนี้คือตัวอย่างของโฟลว์ที่ใช้ในการส่งค่าข้อความ ‘Hello World!’ ออกไปแสดงผลในแถบเมนูข้างดีบัก ซึ่งเมื่อนำโค้ดด้านบนไปวางลงในโฟลว์ของผู้ใช้งาน จะปรากฏโฟลว์ดังรูปขึ้น

ในกรณีที่บางโหนดจำเป็นต้องให้ผู้ใช้งานเติมค่าบางอย่างลงไปเอง จะปรากฏเป็น <> เพื่อให้ผู้ใช้งานแก้ไขในส่วนนั้นก่อนการนำไปใช้งาน โดยให้ใส่ค่าตามที่ระบุเอาไว้ในวงเล็บนั้นเสียก่อน จึงจะสามารถนำไปใช้งานในโฟลว์ได้

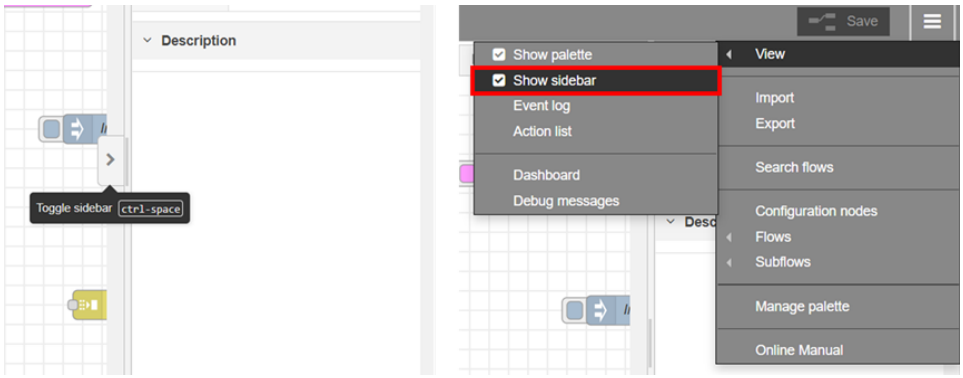
แถบเมนูข้าง

แถบเมนูข้าง (Sidebar) ทางด้านขวาของหน้าต่างโปรแกรมจะมีเครื่องมือช่วยเหลือให้สำหรับผู้ใช้งานสามารถนำไปใช้งานได้ เช่น แถบเมนูรายละเอียด (Info) ที่จะแสดงรายละเอียดของโหนดนั้น แถบเมนู Debug ที่จะแสดงข้อความที่ถูกส่งภายในโฟลว์ แถบเมนูตั้งค่าโหนด (Config) ที่ใช้ในการแก้ไขตั้งค่าการทำงานของโหนดต่าง ๆ เป็นต้น โดยบางโหนดก็อาจจะมีแถบเมนูข้างนี้เป็นของตัวเองก็เป็นได้ เช่น โหนดประเภทแดชบอร์ด (Dashboard) ที่มีแถบเมนูในการแก้ไขจัดการการแสดงผลของโหนดต่าง ๆ เป็นต้น ซึ่งผู้ใช้งานสามารถเปลี่ยนไปหน้าเมนูต่าง ๆ ได้โดยการคลิกที่ปุ่มลูกศรมุมขวาบนของแถบเมนูข้างนี้แล้วเลือกเมนูที่ต้องการเปิดใช้งาน

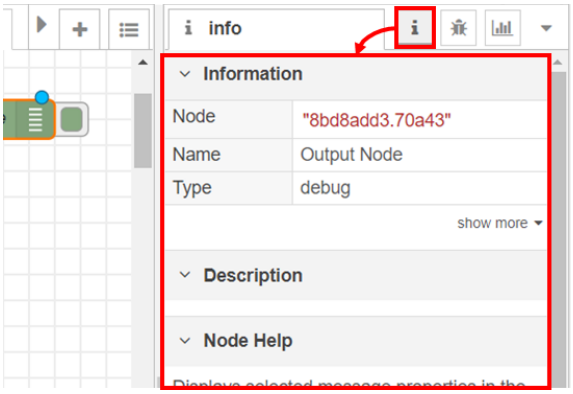


ผู้ใช้งานสามารถย่อขยายแถบเมนูข้างนี้ได้โดยการลากขอบของแถบเมนูข้างย่อเข้า-ขยายออกบนเวิร์คสเปซ แต่หากผู้ใช้งานลากจนเข้าใกล้ขอบของหน้าโปรแกรมทางด้านขวา แถบเมนูข้าง

จะถูกซ่อนลงไป ซึ่งผู้ใช้งานสามารถเปิดมันขึ้นมาได้อีกครั้งโดยการนำเมาส์ไปลอยเหนือขอบหน้าต่างโปรแกรมด้านขวามือให้ปรากฏลูกศรออกมาแล้วคลิกที่ลูกศรนั้น หรือกดเลือกที่ **‘Show sidebar’** ในเมนู **‘View’** ซึ่งอยู่ภายในเมนูด้านขวาที่อยู่บนแถบเมนูด้านบน



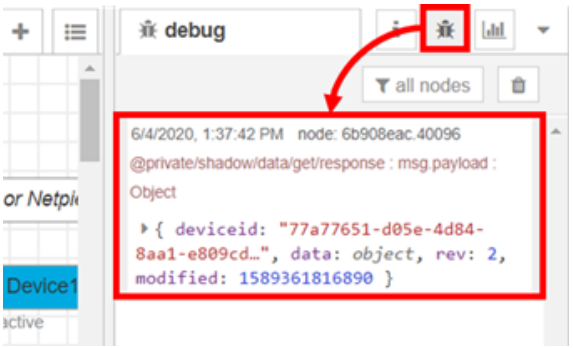
Nodes Information Sidebar



แถบเมนูแสดงรายละเอียดโหนด (Nodes Information Sidebar) จะแสดงให้เห็นถึงรายละเอียดของโหนดที่เลือกอยู่ในปัจจุบันขึ้นมา ไม่ว่าจะเป็นค่าคุณสมบัติของโหนดนั้น หรือข้อความตัวช่วยสำหรับโหนดนั้นก็ตาม

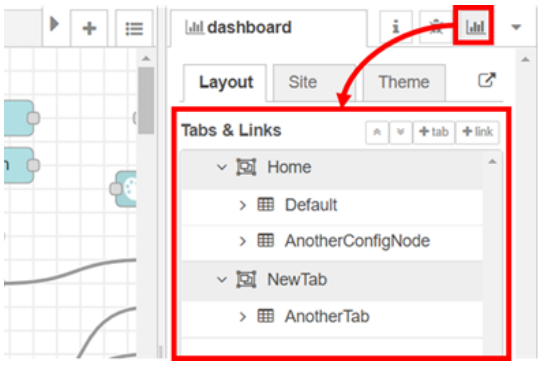
แต่หากไม่ได้มีโหนดใดที่ถูกเลือกอยู่ภายในโฟลว์ แถบเมนูนี้จะแสดงรายละเอียดของโฟลว์นั้นขึ้นมาแทน ซึ่งเป็นรายละเอียดที่ผู้ใช้งานได้ทำการแก้ไขไว้เมื่อทำการแก้ไขโฟลว์

Debug Messages Sidebar



แถบหน้าต่าง Debug นี้จะใช้แสดงผลข้อความที่ถูกส่งเข้าไปในโหนด **‘Debug’** ภายในโฟลว์นั้นในลักษณะของ log หรือบันทึกเหตุการณ์พร้อมช่วงเวลาที่ข้อมูลนั้นได้ทำการถูกส่งเข้าไป ซึ่งรายละเอียดวิธีการใช้งานแถบเมนูนี้จะถูกอธิบายอีกครั้งเมื่อมีการกล่าวถึงการใช้งานของข้อความที่ถูกส่งอยู่ภายในโฟลว์

Dashboard Sidebar



โหนดบางประเภทจะมีแถบเมนูเป็นของตัวเอง หนึ่งในนั้นคือโหนดในหมวดหมู่ของแดชบอร์ด (Dashboard) ซึ่งจะมีแถบเมนูที่ปรากฏรายละเอียดของวิดเจ็ทต่าง ๆ ที่ถูกสร้างเอาไว้

ภายในแถบเมนูนี้ จะประกอบไปด้วยรายละเอียดของโหนดของหน้าต่างการแสดงผลต่าง ๆ ว่าถูกจัดกลุ่มไว้ให้อยู่ในกลุ่มใด (Group) ในแถบหน้าต่างใด (Tab) ของแดชบอร์ด

ค่าข้อมูลและข้อความต่างที่ถูกส่งผ่านแต่ละโหนดข้อมูลนั้นยังคงเชื่อมต่อกันตามลักษณะของโฟลว์เช่นเดิม แม้แต่ละโหนดจะถูกจัดแยกอยู่ในต่างกลุ่มและแถบหน้าต่างก็ตาม

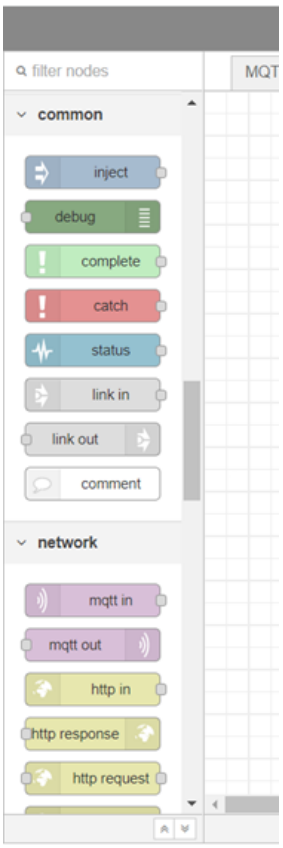
คีย์ลัดโฟลว์

ภายในโฟลว์นี้ ผู้ใช้งานสามารถใช้คีย์ลัดบนคีย์บอร์ดในการเข้าถึงหรือใช้งานส่วนต่าง ๆ ภายในโฟลว์ได้ หรือการใช้แถบเมนูที่อยู่ขวาสุดบนแถบด้านบนของโปรแกรมเพื่อเข้าถึงส่วนต่าง ๆ ของโปรแกรมได้ โดยด้านล่างคือรายการคีย์ลัดภายในโฟลว์นี้

การทำงาน	คีย์ลัด
ซูมเข้าโฟลว์ (Zoom In)	`Ctrl/⌘ + =`
ซูมออกโฟลว์ (Zoom Out)	`Ctrl/⌘ + -`
รีเซ็ตการซูมของโฟลว์กลับไปเป็นปกติ (Reset Zoom)	`Ctrl/⌘ + 0`
เลือกโหนดทั้งหมดในโฟลว์ (Select all)	`Ctrl/⌘ + a`
คัดลอกผังของโหนดที่เลือก (Copy to Clipboard)	`Ctrl/⌘ + c`
ตัดผังของโหนดที่เลือก (Cut to Clipboard)	`Ctrl/⌘ + x`
วางผังของโหนดที่คัดลอกหรือตัดเอาไว้ (Paste from Clipboard)	`Ctrl/⌘ + v`
นำเข้าหน้าโฟลว์ (Import)	`Ctrl/⌘ + i`
ส่งออกโฟลว์ (Export)	`Ctrl/⌘ + e`
ค้นหาโหนด (Search Flows)	`Ctrl/⌘ + f`
เปิดและปิดแพลิตต์ (Toggle Palette)	`Ctrl/⌘ + p`

เปิดและปิดแถบเมนูข้าง (Toggle Sidebar)	`Ctrl`/⌘ + `Space`
ล้างข้อความทั้งหมดในแถบเมนูข้าง Debug	`Ctrl`/⌘ + `Alt` + `l`

โหนด



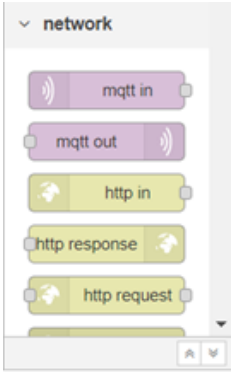
หากเปรียบการใช้งาน Flow Engine เป็นดั่งการสร้างผังงานทั่วไป (Flowchart) โหนดแต่ละโหนดจะเปรียบได้ดั่งสัญลักษณ์ต่าง ๆ ภายในหน้าผังงานที่มีหน้าที่แตกต่างกัน โดยจะเริ่มต้นจากกล่องหรือสัญลักษณ์ที่อยู่ลำดับบนสุดของผังงาน แล้วไล่ไปตามเส้นทางของผังงานนั้นจนถึงปลายสาย

ในแถบด้านซ้ายมือของหน้าต่างปรากฏแพลิตต์ (Palette) ซึ่งรวบรวมโหนดต่าง ๆ เอาไว้ให้ผู้ใช้งานนำไปวางลงในเวิร์คสเปซเพื่อสร้างโปรแกรมใหม่ขึ้นมา ซึ่งโหนดแต่ละโหนดนั้นจะมีการใช้งานและการตั้งค่าที่แตกต่างกันไป ซึ่งจะถูกจัดเรียงอยู่ในหมวดหมู่ต่าง ๆ เป็นการแยกประเภทการใช้งานหลักของแต่ละโหนดให้โหนดประเภทที่คล้ายคลึงกันอยู่ด้วยกัน เช่น โหนดทั่วไป (Common) ซึ่งรวบรวมโหนดพื้นฐานเอาไว้ โหนดเกี่ยวกับเครือข่าย (Network) ซึ่งใช้ในการรับ-ส่งค่าข้อมูลผ่านทางเครือข่ายต่าง ๆ หรือโหนดของแดชบอร์ด (Dashbord) ซึ่งจะช่วยในการสร้างแดชบอร์ดขึ้นจากการประกอบร่างของโฟลว์ที่สร้างไว้ เป็นต้น

ผู้ใช้งานสามารถลากโหนดจากแพลิตต์มือลงไปวางในโฟลว์ได้ ซึ่งเมื่อลากโหนดจากแพลิตต์ด้านซ้ายมือลงมาใช้งานแล้ว คำที่ปรากฏอยู่บนโหนดในโฟลว์นั้นอาจจะมีการเปลี่ยนแปลงไป

แพลิตต์

แพลิตต์ (Palette) ทางด้านซ้ายของหน้าต่างโปรแกรมได้ทำการรวบรวมโหนดต่าง ๆ ซึ่งถูกติดตั้งและพร้อมสำหรับการใช้งาน โดยจะถูกจัดแบ่งเอาไว้ตามหมวดหมู่ต่าง ๆ เช่น ข้อมูลขาเข้า (Input) ข้อมูลขาออก (Output) หรือฟังก์ชัน (Function) เป็นต้น รวมไปถึงแยกซับโฟลว์ (Subflow) ที่ผู้ใช้งานสร้างเอาไว้เป็นอีกหนึ่งหมวดหมู่ด้วยเช่นกัน

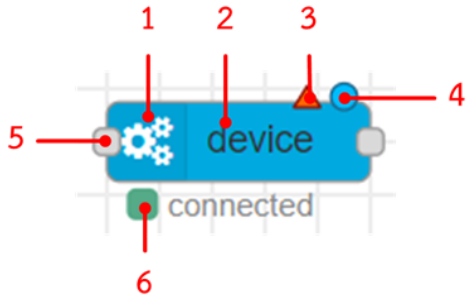


ผู้ใช้งานสามารถย่อ-ขยายดูโหนดภายในแต่ละหมวดหมู่ได้โดยการคลิกที่ชื่อของหมวดหมู่ที่ต้องการย่อ-ขยายนั้น แต่หากผู้ใช้งานต้องการย่อหรือขยายหมวดหมู่ทั้งหมดภายในแถบโหนด ผู้ใช้งานสามารถใช้บริการปุ่มกดที่อยู่ด้านล่างของแพลิตต์เพื่อย่อทุกหมวดหมู่ (⌃) และขยายทุกหมวดหมู่ (⌄) ได้ภายในการคลิกเพียงครั้งเดียว

ในกรณีที่ผู้ใช้งานต้องการค้นหาโหนดที่อยู่ภายในแพลิตต์ ผู้ใช้งานสามารถพิมพ์ลงไปในช่องค้นหาด้านบนของแพลิตต์นั้นได้

หากผู้ใช้งานต้องการจะซ่อนแถบเมนูนี้ ผู้ใช้งานก็สามารถซ่อนได้ด้วยการคลิกที่ปุ่มซ่อนที่จะปรากฏขึ้นเมื่อนำเมาส์ไปลอยเหนือขอบด้านขวาของแพลิตต์ ซึ่งจะแสดงมาในลักษณะของลูกศรชี้ไปทางด้านซ้าย

ส่วนประกอบของโหนด



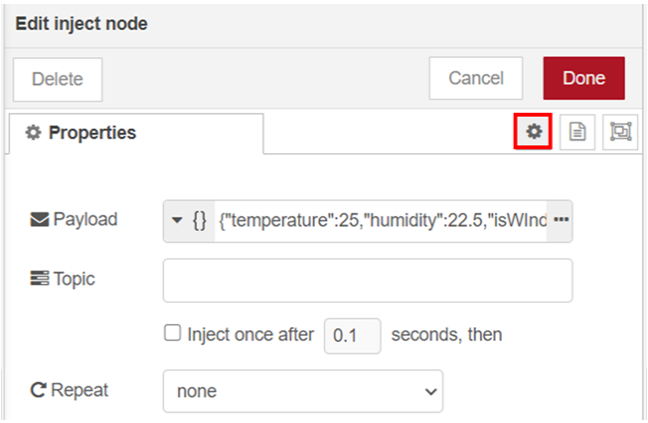
1. สัญลักษณ์ (Icon) ของโหนดนี้
2. ชื่อ (Name) ของโหนดนี้ หรือในบางโหนดอาจบอกถึงข้อมูลที่บรรจุอยู่ภายในโหนดนั้น
3. หากมีสัญลักษณ์สามเหลี่ยมสีแดงปรากฏขึ้นบนโหนด นั่นหมายถึงมีข้อผิดพลาดบางอย่างที่เกิดขึ้นภายในการตั้งค่าขอโหนดนี้
4. หากมีสัญลักษณ์วงกลมสีฟ้าปรากฏขึ้นบนโหนด นั่นหมายถึงมีการเปลี่ยนแปลงที่เกิดขึ้นกับโหนดนี้แต่ยังไม่ได้ทำการบันทึกการเปลี่ยนแปลงนั้น
5. แต่ละโหนดจะปรากฏจุดสีเทาขึ้นที่ด้านข้างของโหนดซึ่งเป็นจุดเชื่อมต่อที่ใช้ในการเชื่อมต่อระหว่างแต่ละโหนดให้เข้าด้วยกัน ซึ่งจุดสีเทาในแต่ละฝั่งของโหนดนั้นมีการใช้งานและข้อจำกัดที่แตกต่างกัน ดังนี้
    a. จุดสีเทาที่อยู่ทางด้านซ้ายของโหนด หมายถึงการรับค่าข้อมูลจากโหนดอื่นเข้ามาใช้งาน ซึ่งแต่ละโหนดจะสามารถมีจุดสีเทาด้านซ้ายได้เพียงจุดเดียวเท่านั้น
    b. จุดสีเทาที่อยู่ทางด้านขวาของโหนด หมายถึงการส่งค่าข้อมูลที่อยู่ภายในโหนดนี้ออกไปยังโหนดอื่น ๆ ได้ ซึ่งบางโหนดอาจมีจุดเชื่อมต่อขาออกได้มากกว่าหนึ่งจุด
6. บางโหนดอาจปรากฏค่าสถานะปัจจุบันของโหนดนั้นไว้ด้านล่างของโหนด เช่น โหนดที่ต้องการแสดงสถานะการเชื่อมต่อ เป็นต้น

ในบางโหนดอาจจะมีปุ่มกดที่อยู่ทางด้านซ้ายหรือขวาของโหนดที่ใช้ในการกระตุ้นการทำงานบางอย่างของโหนด เช่น กระตุ้นให้มีการส่งข้อความออกไปจากโหนดนั้น เป็นต้น

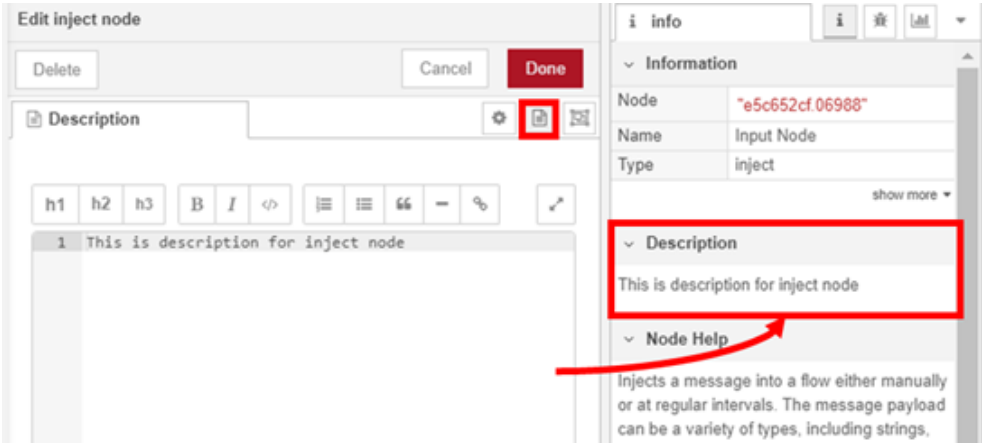
การตั้งค่า

ผู้ใช้งานสามารถแก้ไขการตั้งค่าของโหนดแต่ละโหนดได้ด้วยการ double-click บนโหนดนั้น ๆ ที่ต้องการ หรือทำการกดเลือกตัวโหนดที่ต้องการแก้ไข จากนั้นกดปุ่ม `Enter` บนแป้นคีย์บอร์ด โดยการทำเช่นนี้จะเป็นการเปิดหน้าต่างแก้ไขด้านขวามือขึ้นมาเพื่อให้ผู้ใช้งานสามารถแก้ไขได้ ซึ่งการตั้งค่าแก้ไขนี้จะประกอบไปด้วยสามส่วนด้วยกัน ตามแถบเมนูในหน้าการตั้งค่า

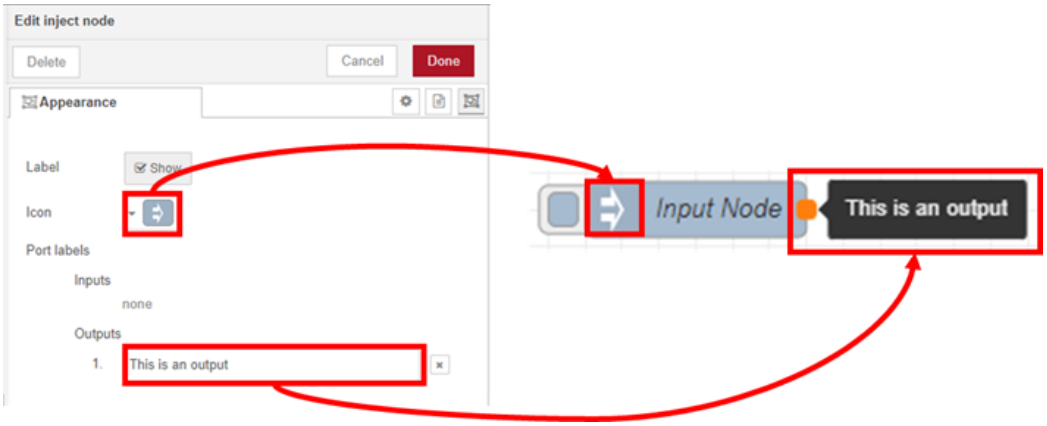
- **ค่าคุณสมบัติของโหนด (Properties)** ซึ่งจะเปลี่ยนแปลงหน้าตาและลักษณะในการตั้งค่าต่าง ๆ ไปตามประเภทของโหนดนั้น ๆ



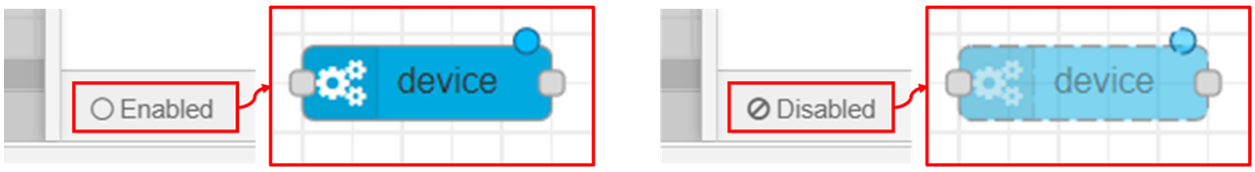
- **คำอธิบาย (Description)** คือการเพิ่มคำอธิบายให้โหนดนั้นในลักษณะของคู่มือ (Documentation) ที่ถูกเขียนขึ้นในลักษณะของ Markdown โดยจะปรากฏเป็นข้อความขึ้นที่แถบเมนูรายละเอียด (Information Sidebar) ที่ด้านขวามือของโปรแกรม



- **ลักษณะการแสดงผล (Appearance)** คือการตั้งค่าลักษณะของโหนดนั้นที่จะแสดงผลอยู่บนโฟลว์ เช่น ต้องการให้ปรากฏชื่อ (Label) ประกอบโหนดนั้นหรือไม่ สัญลักษณ์ (Icon) ที่ต้องการใช้งานเป็นแบบใด และสามารถเพิ่มคำอธิบายประกอบการจุดเชื่อมต่อ (Port Labels) ทั้งจุดเชื่อมต่อทางเข้าและทางออกของโหนดนั้นได้อีกด้วย



นอกเหนือจากนั้น หากผู้ใช้งานไม่ต้องการให้โหนดใดมีการทำงานอยู่ภายในโฟลว์ ผู้ใช้งานสามารถคลิกที่ปุ่ม ‘Enabled’ ด้านล่างของหน้าต่างแก้ไขให้กลายเป็น ‘Disabled’ เพื่อปิดการใช้งานโหนดนั้นได้ โดยโหนดและเส้นการเชื่อมต่อไปยังโหนดนั้นจะกลายเป็นเส้นประที่ไม่เกิดการทำงานขึ้น ในทางกลับกัน หากผู้ใช้งานต้องการกลับมาใช้โหนดนั้นอีก ให้ทำการคลิกที่ปุ่มเดิมเพื่อเปลี่ยนจาก ‘Disabled’ ให้กลายเป็น ‘Enabled’ เพื่อใช้งานได้ตามปกติ



โหนดที่ถูกปิดการทำงานเอาไว้จะไม่มีข้อความใด ๆ ที่สามารถไหลผ่านเข้าไปภายในโหนดนั้นได้ และจะไม่มีข้อความใดไหลผ่านได้หากโหนดนั้นถูกลากเชื่อมต่อระหว่างโหนดอื่น ๆ อีกสองโหนด

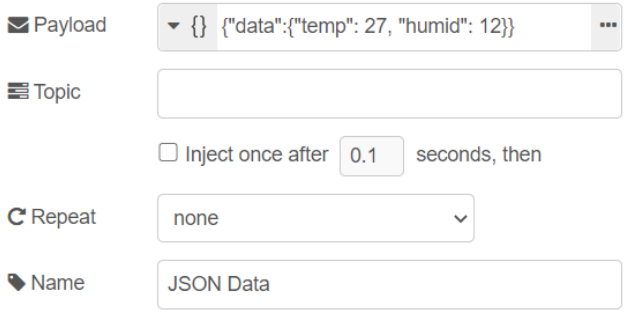
โหนดหลัก

โดยทั่วไปแล้ว จะมีโหนดบางประเภทที่มักถูกนำมาใช้งานอยู่ภายในโฟลว์อยู่เสมอ เปรียบเสมือนโหนดหลักของ Flow Engine ที่ผู้ใช้ควรรู้จักถึงประโยชน์และลักษณะในการใช้งานโหนดเหล่านี้เอาไว้ เช่น โหนดที่ทำการสร้างข้อความขึ้นมาอย่างโหนด **‘Inject’** หรือโหนดที่ทำการรับข้อความที่ส่งออกจากโหนดต่าง ๆ มาแสดงบนแถบเมนูดีบักด้านขวาของโปรแกรมอย่างโหนด **‘Debug’** เป็นต้น โดยโหนดหลัก ๆ เหล่านั้นมีดังต่อไปนี้

Inject

Inject คือโหนดที่อนุญาตให้ผู้ใช้งานสามารถสร้างข้อมูลสมมติขึ้นมาภายในโฟลว์ได้ตามประเภทของค่าข้อมูลที่ผู้ใช้งานต้องการ เช่น ตัวเลข ข้อความ หรืออ็อบเจกต์ เป็นต้น โดยจะมีปุ่มทางด้านซ้ายของโหนดที่ใช้ในการส่งข้อความหรือข้อมูลภายในโหนดนี้ออกไปยังโหนดที่เชื่อมต่อกัน

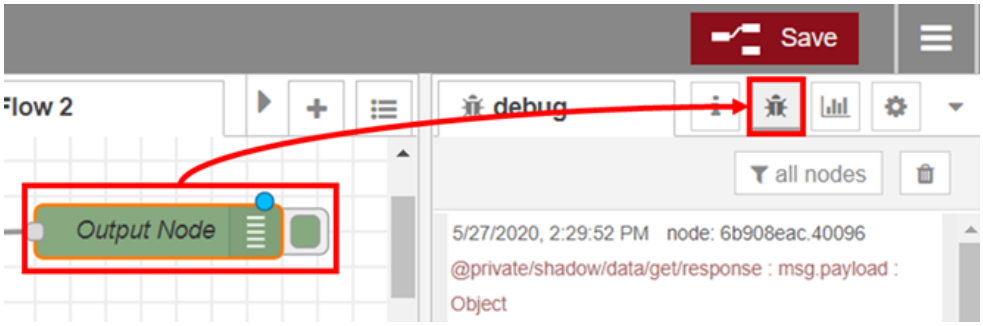
ในขณะเดียวกัน โหนดนี้ก็มีส่วนการตั้งค่าที่อนุญาตให้ตัวมันเองสามารถส่งข้อความออกไปจากโหนดได้ตามระยะเวลาที่ถูกกำหนดเอาไว้ เช่น ให้ส่งค่าข้อมูลออกไปจากโหนดนี้ทุก ๆ 5 นาที หรือส่งค่าข้อมูลทุก ๆ หกโมงเช้าของวันจันทร์ หรือแม้แต่ให้ส่งค่าข้อมูลออกจากโหนดนี้ทุก 1 นาที ระหว่างช่วงเวลาหกโมงเช้าถึงเจ็ดโมงเช้าของวันจันทร์ เป็นต้น



ข้อความที่ถูกส่งไปจากโหนดนี้นั้นจะประกอบไปด้วยตัวแปรสองค่าอันได้แก่ `payload` ซึ่งเป็นข้อมูลของโหนดนี้ซึ่งสามารถเป็นได้ทั้งข้อความ ตัวเลข ค่าความจริง หรือข้อมูลประเภทอื่น ๆ ตามที่ผู้ใช้งานกำหนด ควบคู่กับค่า `topic` ที่เป็นข้อความส่งไปคู่กัน ซึ่ง `topic` ที่ถูกส่งออกไปนั้นมักจะถูกใช้เป็นชื่อตัวแปรของค่า `payload` ที่ถูกส่งออกไปจากโหนดนี้

Debug

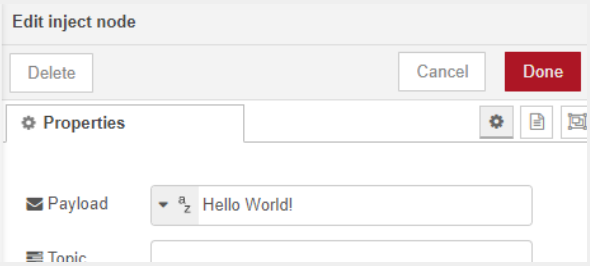
Debug คือโหนดที่ใช้ในการแสดงผลข้อความที่ได้รับมาจากโหนดอื่น ๆ ขึ้นมาปรากฏบนแถบเมนูข้าง ‘Debug’ ทางด้านขวาของโปรแกรม



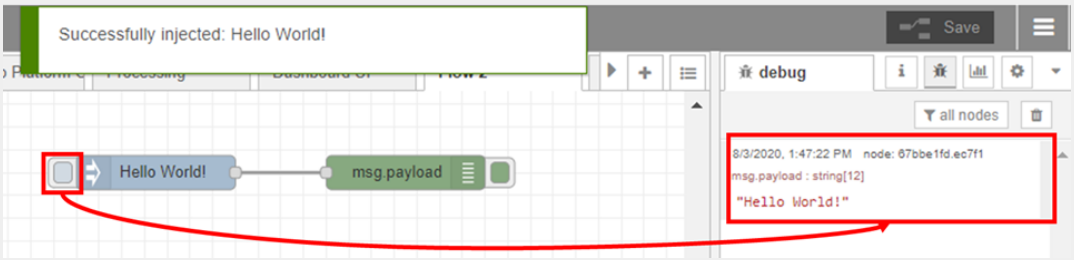
ปุ่มที่ปรากฏอยู่ทางด้านขวาของโหนดนี้จะเป็นการเปิดหรือปิดการแสดงผลลัพธ์ของตัวมันเองเอาไว้ว่าต้องการให้แสดงผลของข้อมูลที่เข้ามาภายในโหนดนี้หรือไม่ ซึ่งประเด็นในการปิดการทำงานของโหนดนี้ในกรณีที่ไม่ได้ถูกใช้งานนั้นเป็นสิ่งที่แนะนำให้ทำอย่างยิ่ง เพราะมันจะทำให้แถบเมนูด้านข้างไม่แสดงข้อความที่ไม่เกี่ยวข้องมากเกินความจำเป็น เช่นเดียวกับการลบโหนดนี้ออกไปจากหน้าโฟลว์ในกรณีที่ไม่ได้ต้องการให้แสดงผลค่าข้อมูลของโหนดที่ส่งมาอีกต่อไป

ตัวอย่าง เพื่อแสดงผลข้อความว่า ‘Hello World’ ขึ้นมาในแถบเมนูข้างดีบัก ให้ผู้ใช้งานทำการนำโหนด **‘Inject’** มาวางลงบนโฟลว์ เช่นเดียวกับโหนด **‘Debug’** จากนั้นให้ลากเส้นเชื่อมต่อออกมาจากโหนด **‘Inject’** เข้าสู่จุดเชื่อมต่อทางด้านซ้ายของโหนด **‘Debug’**

ภายในโหนด **‘Inject’** นั้น ให้ผู้ใช้งานทำการตั้งค่า ‘payload’ เป็นประเภทของข้อความแล้วพิมพ์คำว่า ‘Hello World’ ลงไป จากนั้นจึงคลิกที่ปุ่ม ‘Done’ เพื่อยืนยันการเปลี่ยนแปลง

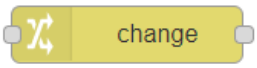


เมื่อทำการบันทึกโฟลว์นี้เรียบร้อยแล้ว ทุกครั้งที่ผู้ใช้งานทำการคลิกบนปุ่มที่อยู่ทางด้านขวาของโหนด **‘Inject’** ค่าข้อความที่อยู่ภายในโหนดจะถูกส่งไปแสดงผลบนแถบเมนูข้างดีบักทันที



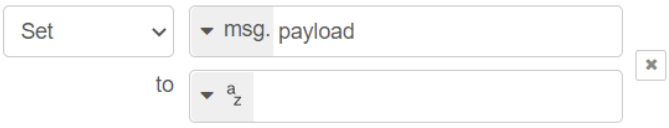
```
[{"id":"4f19b295.ce21dc","type":"inject","z":"8cfb1028.a0d6d","name":"","topic":"","payload":"Hello
World!","payloadType":"str","repeat":"","crontab":"","once":false,"onceDelay":0.1,"x":370,"y":960,"wi
res":[["67bbe1fd.ec7f1"]]},{"id":"67bbe1fd.ec7f1","type":"debug","z":"8cfb1028.a0d6d","name":"","acti
ve":true,"tosidebar":true,"console":false,"tostatus":false,"complete":"payload","targetType":"msg","x
":590,"y":960,"wires":[]}]
```


Change

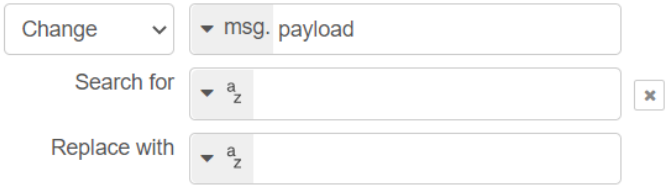


Change คือโหนดที่อนุญาตผู้ใช้งานสามารถแก้ไขหรือเปลี่ยนแปลงค่าข้อมูลที่รับเข้ามาจากโหนดอื่น ๆ ได้ เช่น การตั้งค่าข้อมูลเป็นค่าที่กำหนด การเปลี่ยนแปลงข้อมูลบางส่วน การย้ายค่าข้อมูลหนึ่งจากตัวแปรหนึ่งไปยังอีกตัวแปรหนึ่ง หรือการลบค่าข้อมูลนั้นออกไป ซึ่งโหนดประเภทนี้แต่ละโหนดนั้นสามารถมีค่าคำสั่งมากกว่าหนึ่งคำสั่ง ซึ่งจะเรียงลำดับการทำงานเหล่านั้นจากบนลงล่างตามที่ผู้ใช้งานกำหนดเอาไว้

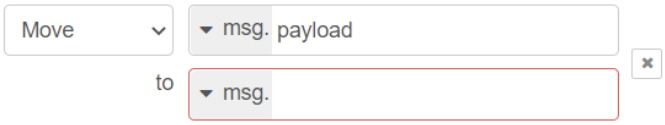
- **Set** คือการเปลี่ยนแปลงค่าข้อมูลที่เข้ามาให้กลายเป็นค่าข้อมูลที่ผู้ใช้งานกำหนดขึ้นมาเองได้



- **Change** คือการเปลี่ยนแปลงค่าข้อมูลที่มีอยู่แล้วด้วยการแทนที่ด้วยผค่าใหม่ที่ผู้ใช้งานกำหนดขึ้นมาลงไป มักใช้ในกรณีที่ค่าข้อมูลเข้ามาในลักษณะของข้อความ โดยจะเป็นการเปลี่ยนตัวอักษรหรือข้อความบางส่วนที่กำหนดให้กลายเป็นค่าอื่นได้

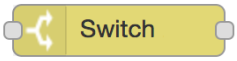


- **Move** จะเป็นการย้ายของมูลจากตัวแปรหนึ่งไปยังอีกตัวแปรหนึ่ง



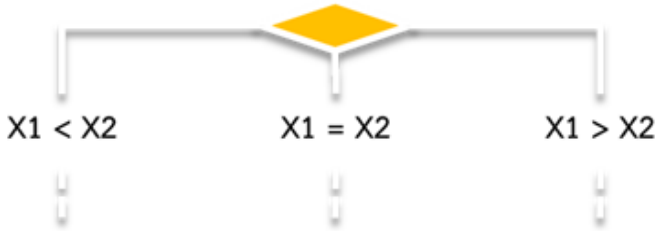
- **Delete** จะเป็นการลบค่าข้อมูลบางส่วนออกตามค่าตัวแปรที่ผู้ใช้งานกำหนด

Switch



Switch คือโหนดที่อนุญาตให้ผู้ใช้งานสามารถตั้งเงื่อนไขหรือกฎให้กับค่าข้อมูลที่เข้ามาได้ โดยจะส่งผลให้เกิดหลายทางของผลลัพธ์โดยอ้างอิงจากเงื่อนไขของค่าที่เข้ามา ซึ่งชื่อของโหนดนี้มีที่มาจากฟังก์ชันพื้นฐานของการเขียนโปรแกรมที่ใช้ในการเลือกค่าข้อมูลที่ต้องการใช้เป็นผลลัพธ์ (Switch-case statement) ไม่ใช่สวิตช์ที่เปิด-ปิดในชีวิตจริงแต่อย่างใด

ตัวอย่างการใช้งาน เช่น การเปรียบเทียบค่าข้อมูลสองค่า หากค่าข้อมูลแรกน้อยกว่าค่าข้อมูลที่สอง ผู้ใช้งานอาจตั้งค่าให้โฟลว์ทำงานอย่างหนึ่ง และในกรณีที่ค่าข้อมูลที่สองน้อยกว่าค่าข้อมูลแรก ผู้ใช้งานอาจตั้งค่าให้มีการทำงานอีกอย่างหนึ่งได้เช่นกัน เป็นต้น



โดยภายในโหนดนี้จะทำการตรวจสอบค่า `msg.payload` ที่เข้ามาเป็นค่าเริ่มต้น (ผู้ใช้งานสามารถแก้ไขค่าที่ต้องการตรวจสอบให้เป็นค่าอื่นได้ เช่น การให้ตรวจสอบค่า `msg.topic` แทน เป็นต้น) ว่าตรงกับเงื่อนไขที่กำหนดเอาไว้หรือไม่ ซึ่งเงื่อนไขที่สามารถตั้งได้ในโหนดนี้มีอยู่ด้วยกัน 4 ประเภท ได้แก่

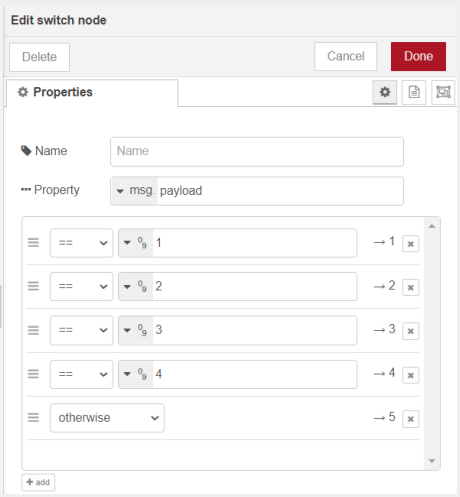
1. *ค่าข้อมูล*นั้นตรงกับเงื่อนไขของค่าข้อมูลที่กำหนดเอาไว้หรือไม่ เช่น มีค่าเท่ากัน มากกว่า หรือน้อยกว่าค่าที่กำหนด เป็นต้น
2. ตำแหน่งของค่าต่าง ๆ ที่อยู่ในข้อมูลที่เป็น*ลำดับรายการ*ว่าต้องการค่าข้อมูลส่วนต้น ส่วนท้าย หรือระหว่างตำแหน่งของค่าข้อมูลสองค่า
3. ค่าข้อมูลที่เข้ามานั้นตรงกับเงื่อนไขที่กำหนดโดยสมการที่ถูกเขียนขึ้นในลักษณะของค่า *JSON*

4. *ถ้ามิเช่นนั้น (Otherwise)* จะเป็นเพียงเงื่อนไขเดียวที่ผู้ใช้งานไม่จำเป็นต้องกำหนดค่าใด ๆ แต่หากค่าข้อมูลที่ได้รับเข้ามาไม่ตรงกับเงื่อนไขใด ๆ ก่อนหน้าเลยแม้แต่เงื่อนไขเดียว เงื่อนไขนี้จะทำการรองรับทุกค่าข้อมูลที่ไม่ตรงกับเงื่อนไขเหล่านั้นทั้งหมด

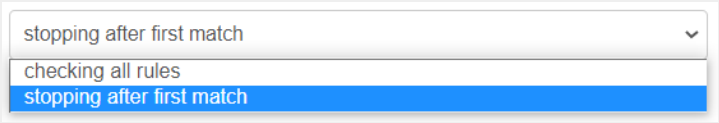
นอกเหนือไปกว่านี้คือ ผู้ใช้งานยังสามารถกำหนดได้อีกด้วยว่าต้องการให้ตรวจสอบครบทุกเงื่อนไขที่ทำการกำหนดเอาไว้ (Checking all rules) หรือให้หยุดการตรวจสอบเปรียบเทียบทันทีที่มีเงื่อนไขใดตรงกับค่าข้อมูลที่เข้ามา (Stop after first match) ซึ่งการใช้วิธีตรวจสอบทั้งสองแบบนั้นขึ้นอยู่กับลักษณะของโปรแกรมที่ผู้ใช้งานต้องการว่าควรเป็นเช่นไร เช่น หากทุกเงื่อนไขนั้นต่างกันหมด และค่าข้อมูลที่เข้ามามีสิทธิ์ตรงกับค่าข้อมูลเพียงค่าเดียวอยู่แล้วเท่านั้น การใช้การตรวจสอบแบบหยุดทันทีเมื่อพบเงื่อนไขที่ตรงจึงเหมาะสมมากกว่า อีกทั้งยังช่วยลดระยะเวลาในการตรวจสอบเงื่อนไขต่าง ๆ ลงไปด้วย เป็นต้น

ตัวอย่าง ผู้ใช้งานต้องการตรวจสอบว่า ค่าข้อมูลที่ได้รับเข้ามาจากโหนดก่อนหน้านั้นเท่ากับจำนวนเต็ม 1, 2, 3, และ 4 หรือไม่ โดยให้แสดงผลสรุปสุดท้ายออกมาในลักษณะของคำในภาษาอังกฤษของตัวเลขนั้น ๆ (เช่น หากค่าข้อมูลเป็น 1 จะให้แสดงผลลัพธ์ออกมาว่า ‘One’ เป็นต้น) ซึ่งหากใช่เลขเหล่านี้ จะให้โหนด **‘Switch’** นี้แยกค่าข้อมูลทั้งสี่ค่าออกเป็นคนละจุดกัน แต่หากค่าข้อมูลที่เข้ามาไม่ตรงกับค่าใดเลย จะให้แยกออกมาเป็นอีกจุดเชื่อมต่อขาออกอีกจุดหนึ่ง ซึ่งหมายถึงโหนดนี้จะมีจุดเชื่อมต่อขาออกถึง 5 จุดนั่นเอง

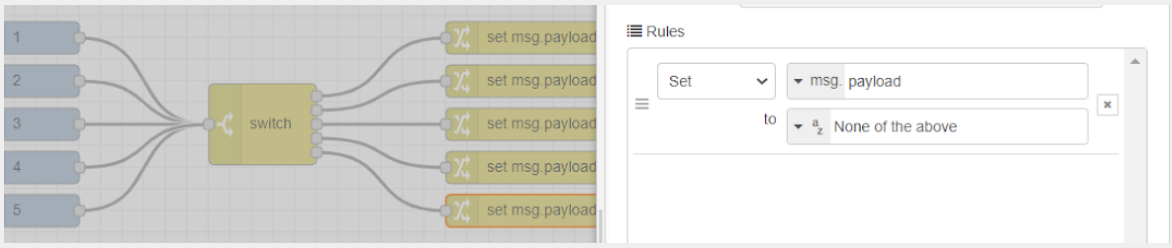
ผู้ใช้งานทำการลากโหนดของข้อมูลที่ต้องการตรวจสอบเข้ากับจุดเชื่อมต่อขาเข้าของโหนดนี้ จากนั้นจึงเปิดหน้าแก้ไขของโหนด **‘Switch’** ขึ้น โดยให้ทำการเพิ่มเงื่อนไขที่ต้องการตรวจสอบให้ครบ 5 เงื่อนไขตามที่กำหนดไว้ข้างต้นโดยการคลิกที่ปุ่ม ‘+ add’ จากนั้นจึงทำการตั้ง 4 เงื่อนไขแรกให้เป็นประเภทของตัวเลข และใส่ค่าที่ต้องการตรวจสอบไล่ตั้งแต่ 1 ถึง 4 ตามลำดับจากบนลงล่าง โดยใช้เงื่อนไขเป็น ‘==’ เพื่อเปรียบเทียบว่าเท่ากับค่าข้อมูลนั้นหรือไม่ ในขณะที่เงื่อนไขสุดท้ายให้ใช้เป็น ‘Otherwise’ เพื่อใช้ในกรณีที่ค่าข้อมูลที่เข้ามานั้นไม่ตรงกับค่าข้อมูลใด ๆ ด้านบนเลย



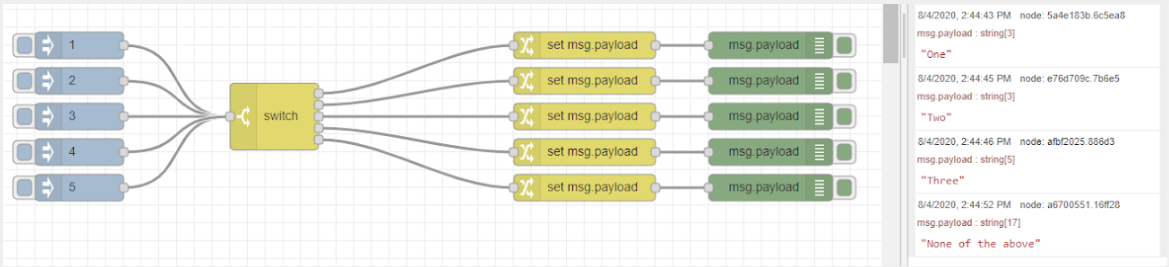
เพื่อลดระยะเวลาในการค้นหาเงื่อนไขที่ตรงกับค่าที่กำหนด ให้ผู้ใช้งานทำการเลือก drop-down ที่อยู่ด้านล่างของกล่องเงื่อนไขให้เป็น ‘stop after first match’ เพื่อให้ทำการหยุดเปรียบเทียบกับเงื่อนไขต่าง ๆ ทันทีที่ตรงกับเงื่อนไขใดเงื่อนหนึ่งด้านบนไปแล้ว



ผู้ใช้งานสามารถเปลี่ยนแปลงผลของค่าข้อมูลที่เข้ามาให้กลายเป็นตัวอักษรได้โดยการใช้โหนด **‘Change’** ในการเปลี่ยนแปลงตัวเลขเหล่านั้นโดยใช้การ ‘Set’ เพื่อเปลี่ยนแปลงค่าข้อมูล payload ให้เป็นค่าที่กำหนดในแต่ละเส้นเชื่อมต่อขาออก และเชื่อมต่อเข้ากับโหนด **‘Debug’** เพื่อแสดงผลข้อมูลเหล่านั้น ในขณะที่ค่าข้อมูลที่ไม่ตรงกับเงื่อนไขใด ๆ เลย (เส้นเชื่อมต่อจากจุดเชื่อมต่อขาออกสุดท้าย) ให้แสดงผลข้อความออกมาว่า ‘None of the above’

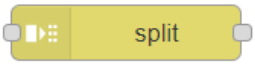


เมื่อทำการแก้ไขจนเรียบร้อยและบันทึกโฟลว์นี้แล้ว หากผู้ใช้งานลองทำการเปลี่ยนแปลงค่าข้อมูลค่าเข้าให้เป็นค่าต่าง ๆ จะพบว่า ค่าข้อมูลที่แสดงผล ณ แถบเมนูข้างจะปรากฏออกมาเป็นข้อความตามที่กำหนดเอาไว้



```
[{"id":"6e6b719d.c403d","type":"switch","z":"8cfb1028.a0d6d","name":"","property":"payload","property
Type":"msg","rules":[{"t":"eq","v":"1","vt":"num"},{"t":"eq","v":"2","vt":"num"},{"t":"eq","v":"3","v
t":"num"},{"t":"eq","v":"4","vt":"num"},{"t":"else"}],"checkall":"false","repair":false,"outputs":5,"
x":390,"y":2500,"wires":[["fc4fc7c7.382a98"],["415dc39a.f32fec"],["a0851495.504328"],["66bb150f.60aa6
c"],["3fab71f3.ea7b5e"]]},{"id":"5a4e183b.6c5ea8","type":"debug","z":"8cfb1028.a0d6d","name":"","acti
ve":true,"tosidebar":true,"console":false,"tostatus":false,"complete":"false","x":850,"y":2420,"wires
":[]},{"id":"e76d709c.7b6e5","type":"debug","z":"8cfb1028.a0d6d","name":"","active":true,"tosidebar":
true,"console":false,"tostatus":false,"complete":"false","x":850,"y":2460,"wires":[]},{"id":"afbf2025
.886d3","type":"debug","z":"8cfb1028.a0d6d","name":"","active":true,"tosidebar":true,"console":false,
"tostatus":false,"complete":"false","x":850,"y":2500,"wires":[]},{"id":"a19195d2.e630b8","type":"debu
g","z":"8cfb1028.a0d6d","name":"","active":true,"tosidebar":true,"console":false,"tostatus":false,"co
mplete":"false","x":850,"y":2540,"wires":[]},{"id":"a6700551.16ff28","type":"debug","z":"8cfb1028.a0d
6d","name":"","active":true,"tosidebar":true,"console":false,"tostatus":false,"complete":"false","x":
850,"y":2580,"wires":[]},{"id":"fc4fc7c7.382a98","type":"change","z":"8cfb1028.a0d6d","name":"","rule
s":[{"t":"set","p":"payload","pt":"msg","to":"One","tot":"str"}],"action":"","property":"","from":"",
"to":"","reg":false,"x":640,"y":2420,"wires":[["5a4e183b.6c5ea8"]]},{"id":"415dc39a.f32fec","type":"c
hange","z":"8cfb1028.a0d6d","name":"","rules":[{"t":"set","p":"payload","pt":"msg","to":"Two","tot":"
str"}],"action":"","property":"","from":"","to":"","reg":false,"x":640,"y":2460,"wires":[["e76d709c.7
b6e5"]]},{"id":"a0851495.504328","type":"change","z":"8cfb1028.a0d6d","name":"","rules":[{"t":"set","
p":"payload","pt":"msg","to":"Three","tot":"str"}],"action":"","property":"","from":"","to":"","reg":
false,"x":640,"y":2500,"wires":[["afbf2025.886d3"]]},{"id":"66bb150f.60aa6c","type":"change","z":"8cf
b1028.a0d6d","name":"","rules":[{"t":"set","p":"payload","pt":"msg","to":"Four","tot":"str"}],"action
":"","property":"","from":"","to":"","reg":false,"x":640,"y":2540,"wires":[["a19195d2.e630b8"]]},{"id
":"3fab71f3.ea7b5e","type":"change","z":"8cfb1028.a0d6d","name":"","rules":[{"t":"set","p":"payload",
"pt":"msg","to":"None of the
above","tot":"str"}],"action":"","property":"","from":"","to":"","reg":false,"x":640,"y":2580,"wires"
:[["a6700551.16ff28"]]}]
```

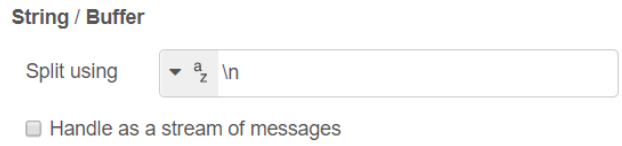

Split



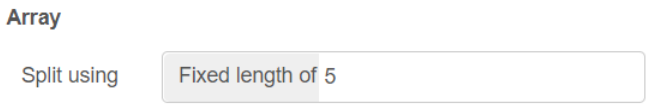
Split คือโหนดที่อนุญาตให้ผู้ใช้งานสามารถแยกค่าข้อมูลออกจากกันได้ ซึ่งโหนดนี้มักใช้คู่กับโหนด **‘Join’** ที่จะรวบรวมแต่ละค่าข้อมูลเข้าด้วยกันให้กลายเป็นค่าข้อมูลเดียว

โหนดนี้จะทำการแยกค่าข้อมูล `msg.payload` ออกจากกันให้กลายเป็นค่าข้อมูลหลายค่า ซึ่งโหนดการแยกค่าข้อมูลนี้รองรับการแยกค่าข้อมูล 3 ประเภท ได้แก่ ข้อความ (String) รายการ (Array) และอ็อบเจกต์ (Object) โดยในการรับค่าข้อมูลแต่ละค่ามาทำการแยกข้อมูลออกจากกันจะมีการตั้งค่าที่แตกต่างกัน ดังนี้

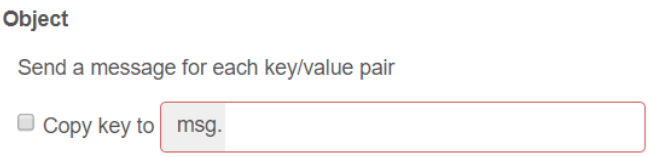
- **String/buffer** คือค่าข้อมูลที่เข้ามาในรูปแบบของข้อความ ซึ่งจะทำการตัดแบ่งข้อความออกเป็นส่วน ๆ โดยอ้างอิงจากตัวอักษรพิเศษที่ต้องการใช้ในการแบ่ง (ค่าเริ่มต้นคือ ‘\n’ หรือการเคาะขึ้นบรรทัดใหม่นั่นเอง) หรือแบ่งข้อความออกเป็นขนาดที่กำหนดไว้ (Fixed length)



- **Array** คือค่าข้อมูลที่เข้ามาในลักษณะรายการของข้อมูลประเภทเดียวกันที่เรียงเป็นลำดับ (เช่น `[1, 2, 3, 4, 5]` เป็นต้น) ซึ่งโหนดนี้จะทำการตัดแบ่งข้อมูลประเภทนี้โดยใช้ขนาดของรายการของข้อมูลนั้นในการแบ่งแต่ละข้อมูลให้แยกออกจากกันเป็นรายการของข้อมูลที่มีขนาดเล็กลงมากว่าเดิม

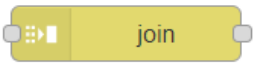


- **Object** คือโหนดที่ค่าข้อมูลที่ถูกส่งเข้ามาเป็นคู่ ๆ ระหว่างค่าตัวแปร (Key) และค่าข้อมูลของตัวแปร (Value) นั้นที่ถูกเรียกว่าอ็อบเจกต์



ซึ่งผลลัพธ์ที่ออกมาจากโหนดนี้นั้น จะออกมาในลักษณะของรายการของค่าข้อมูลที่จะประกอบไปด้วยค่าข้อมูลต่าง ๆ ที่รวบรวมข้อมูลที่เกี่ยวกับข้อความที่ถูกตัดแบ่งออกไปว่ามีลักษณะอย่างไร โดยหากผู้ใช้งานนำค่านี้ส่งไปให้ยังโหนด **‘Join’** โหนดนั้นจะสามารถรวมค่าข้อมูลที่ถูกตัดแบ่งเข้าด้วยกันจนกลายเป็นค่าข้อความค่าเดียวได้

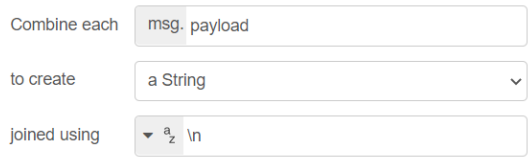
Join



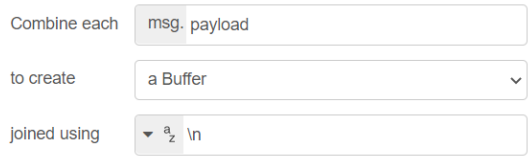
โหนด **‘Join’** ใช้ในการรวบรวมค่าข้อมูลต่าง ๆ เข้าด้วยกันให้กลายเป็นค่าข้อมูลหรือข้อความเพียงค่าเดียวได้ โดยผู้ใช้งานสามารถเลือกได้ว่าต้องการให้ผลลัพธ์ที่ออกมานั้นอยู่ในลักษณะใด ระหว่างข้อความ รายการของข้อมูล หรืออ็อบเจกต์ ซึ่งโหนดนี้เป็นโหนดที่ทำหน้าที่ตรงกันข้ามกับโหนด **‘Split’** ที่ทำการแยกข้อมูลที่เข้ามาภายในโหนดออกจากกัน

โหนดนี้มีการตั้งค่าทั้งหมด 3 รูปแบบ ได้แก่

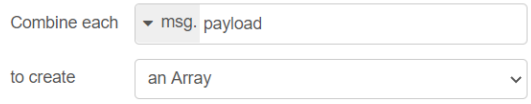
- **Automatic** เมื่อใช้งานคู่กับโหนด **‘Split’** กล่องนี้จะทำการรวมข้อมูลทั้งหมดเข้าด้วยกันเพื่อย้อนคืนสิ่งโหนดนั้นเคยทำ
- **Manual** ใช้ในการรวบรวมค่าข้อมูลเข้าด้วยกันตามที่ผู้ใช้งานกำหนดว่าต้องการให้ผลลัพธ์ออกมาในลักษณะใด และรวบรวมค่าข้อมูลประเภทนั้นอย่างไร
    - **String** คือต้องการให้ผลลัพธ์ออกมาในลักษณะของข้อความ โดยใช้ตัวอักษรที่กำหนดในการเชื่อมต่อค่าข้อความแต่ละค่าเข้าด้วยกัน ซึ่งค่าเริ่มต้นของการเชื่อมข้อความนี้คือการขึ้นบรรทัดใหม่ (End-line; \n)



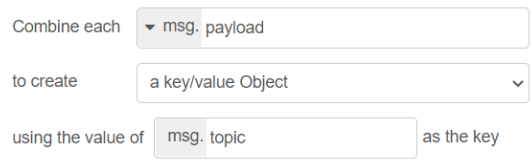
    - **Buffer** คือต้องการให้ผลลัพธ์ออกมาในลักษณะของบัฟเฟอร์ โดยใช้ตัวอักษรที่กำหนดในการเชื่อมต่อค่าข้อความแต่ละค่าเข้าด้วยกัน ซึ่งค่าเริ่มต้นของการเชื่อมข้อความนี้คือการขึ้นบรรทัดใหม่ (End-line; \n)



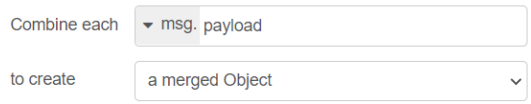
    - **Array** คือการรวมค่าข้อมูลต่าง ๆ ที่เข้ามาภายในโหนดเข้าด้วยกันจนกลายเป็นรายการของข้อมูล



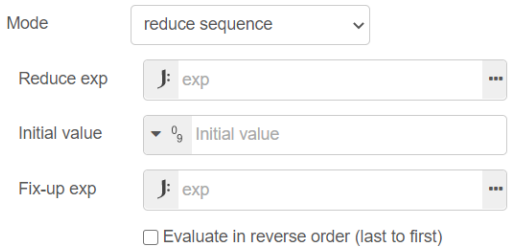
    - **Key/Value Object** คือการสร้างอ็อบเจกต์ใหม่ขึ้นมา โดยผู้ใช้งานสามารถกำหนดได้ว่าต้องการใช้สิ่งใดเป็นตัวแปร (Key) เพื่อเข้าคู่กับค่าข้อมูลนั้น ซึ่งค่าเริ่มต้นคือการใช้ `msg.topic` เป็นเสมือน key ของค่าข้อมูล



    - **Merged Object** จะทำหน้าที่รวมคุณสมบัติของแต่ละข้อความเข้าด้วยกันจนกลายเป็นอ็อบเจกต์เพียงอ็อบเจกต์เดียว



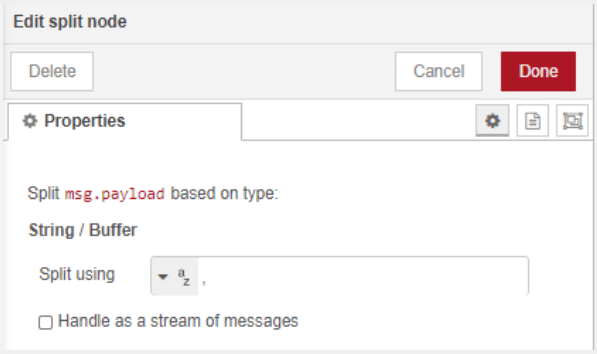
- **Reduce Sequence** จะทำการใส่ค่าสมการให้กับทุกข้อความในลำดับของข้อมูล (Sequence) นั้นเพื่อแปลงให้กลายเป็นค่าข้อมูลเดียว โดยการใช้งานในโหมดนี้จะเป็นการกำหนดว่าโหนดนี้ถูกใช้งานเชื่อมต่อมาจากโหนด **'Split'** หรือได้รับข้อความที่มีตัวแปร `msg.parts` อยู่ภายในโหนดนั้น



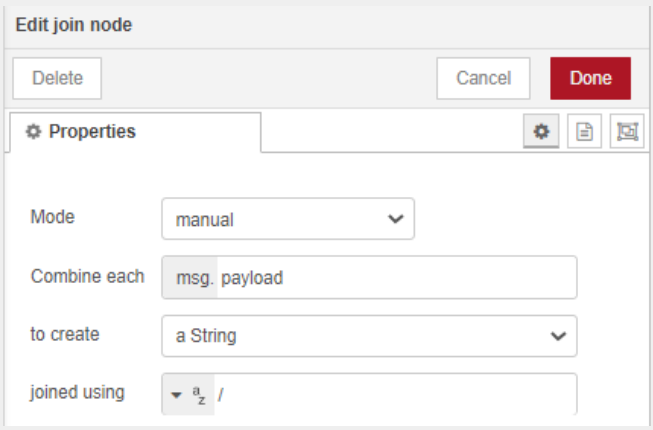
ค่าข้อมูลที่ถูกส่งเข้ามาในโหนดนี้จะทำการรวมกันกลายเป็นอีกค่าหนึ่งก็ต่อเมื่อมีจำนวนของข้อความที่เข้ามาตามที่กำหนด หรือถูกส่งเข้ามาอยู่ภายในช่วงเวลาในหน่วยวินาทีที่กำหนดหลังจากข้อมูลแรกถูกส่งเข้ามาภายในโหนด (Timeout) หรือมีการตั้งค่าตัวแปร `msg.complete` มาจากโหนดก่อนหน้า

ตัวอย่าง ค่าข้อมูล "1, 2, 3" ถูกส่งออกไปจากโหนด **'Inject'** และต้องการเปลี่ยนแปลงเพื่อให้ผลสุดท้ายของค่าข้อมูลเป็น "1/2/3" ผู้ใช้งานสามารถทำได้โดยการแยกค่าข้อความที่เข้ามาออกจากกันโดยใช้โหนด **'Split'** เข้าเชื่อมต่อกับโหนด **'Inject'** นั้น แล้วทำการลากผลลัพธ์เข้ากับโหนด **'Join'**

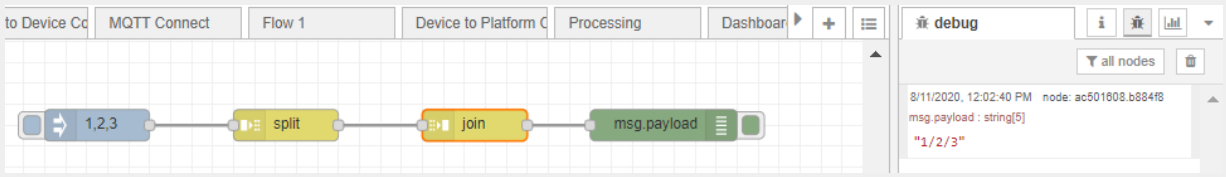
ทำการตั้งค่าภายในโหนด **'Split'** ในส่วนของ String/buffer ให้เป็นการแยกค่าข้อมูลโดยใช้ ','



ทำการตั้งค่าภายในโหนด **'Join'** โดยให้เลือกเป็นหมวด manual เพื่อทำการสร้างข้อความ (String) ที่ถูกเชื่อมต่อเข้าด้วยกันด้วยอักษร '/'



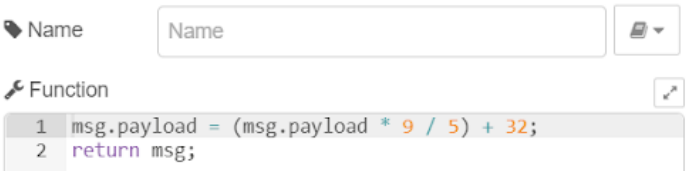
เมื่อผู้ใช้งานลองทำการคลิกที่ปุ่มบนโหนด **‘Inject’** จะพบว่า จากค่าข้อมูลที่ถูกตั้งไว้ตอนแรก ได้ถูกเปลี่ยนจนกลายเป็น “1/2/3” ดังที่กำหนดเอาไว้ในที่สุด




```
[{"id":"ac501608.b884f8","type":"debug","z":"8cfb1028.a0d6d","name":"","active":true,"tosidebar":true
,"console":false,"tostatus":false,"complete":"false","x":810,"y":3480,"wires":[]},{"id":"2a2ef889.5f1
b48","type":"join","z":"8cfb1028.a0d6d","name":"","mode":"custom","build":"string","property":"payloa
d","propertyType":"msg","key":"topic","joiner":"/","joinerType":"str","accumulate":false,"timeout":""
,"count":"","reduceRight":false,"reduceExp":"$abs","reduceInit":"0","reduceInitType":"num","reduceFix
up":"","x":630,"y":3480,"wires":[["ac501608.b884f8"]]},{"id":"be4110af.804a4","type":"split","z":"8cf
b1028.a0d6d","name":"","splt":",","spltType":"str","arraySplt":1,"arraySpltType":"len","stream":false
,"addname":"","x":450,"y":3480,"wires":[["2a2ef889.5f1b48"]]},{"id":"3afcef5d.eb9dd","type":"inject",
"z":"8cfb1028.a0d6d","name":"","topic":"","payload":"1,2,3","payloadType":"str","repeat":"","crontab"
:"","once":false,"onceDelay":0.1,"x":270,"y":3480,"wires":[["be4110af.804a4"]]}]
```


Function

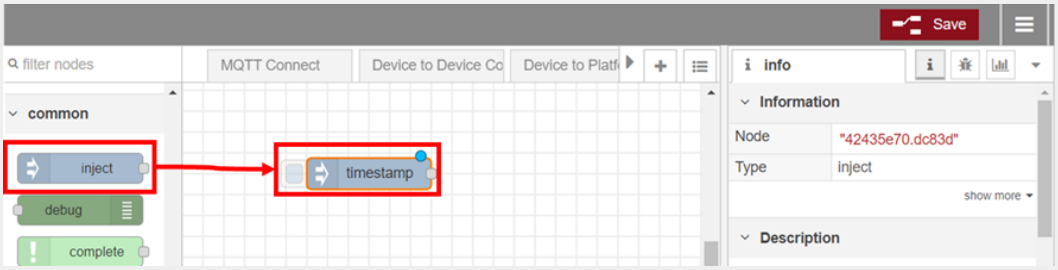
ในกรณีที่ผู้ใช้งานไม่พบโหนดที่เหมาะสมต่อการแก้ไขค่าข้อมูลต่าง ๆ **‘Function’** คือโหนดที่อนุญาตให้ผู้ใช้งานสามารถเขียนโปรแกรมภาษา JavaScript เพื่อเปลี่ยนแปลงค่าข้อความหรือจัดการกับค่าข้อมูลที่ไหลผ่านเข้ามาภายในโหนดนี้ได้



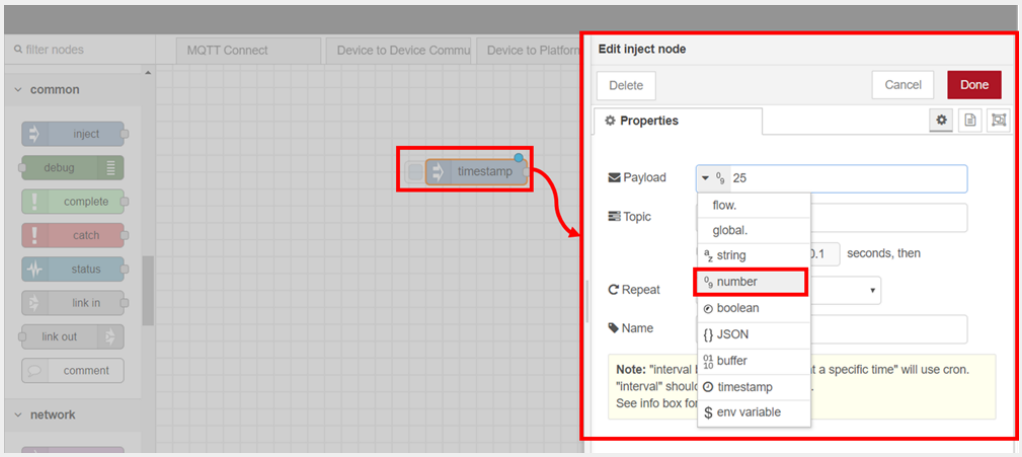
ในการเข้าถึงค่าข้อมูลที่ถูกส่งมาจากโหนดก่อนหน้า ผู้ใช้งานสามารถใช้ตัวแปร `msg` เพื่อแทนค่าข้อความนั้นได้ ซึ่งลักษณะของข้อความที่ถูกส่งนั้นจะถูกกล่าวอีกครั้งในหัวข้อการส่งข้อความ

ตัวอย่าง การสร้างผังงานเริ่มต้นที่จะส่งอุณหภูมิแบบเซลเซียส (°C) เพื่อไปแสดงผลในลักษณะของอุณหภูมิแบบฟาเรนไฮต์ (°F) ณ กล่องรับข้อมูล

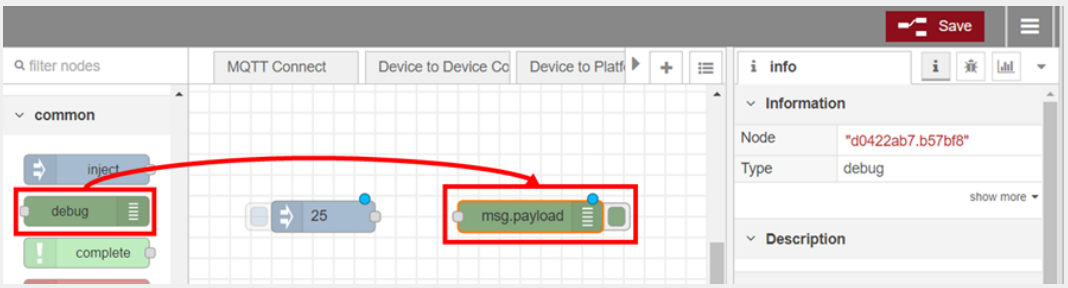
1. ผู้ใช้งานทำการนำกล่องฟังก์ชั่น **'Inject'** ที่อยู่ในหมวดหมู่ ***'Common'*** ลงมาวางในพื้นที่ผังงาน



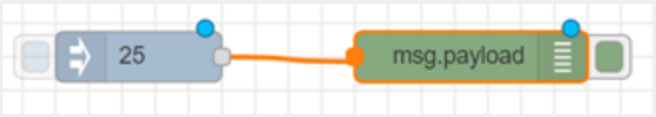
2. คลิกที่ตัวกล่อง **'Inject'** เพื่อทำการแก้ไขข้อมูลภายใน โดยให้เปลี่ยนประเภทของข้อมูลที่จะส่งเป็นตัวเลขในกล่อง **'Payload'** ซึ่งปัจจุบันยังคงเป็นการส่งตราประทับเวลา (Timestamp) อยู่ โดยให้ทำการคลิกที่ drop-down แล้วเลือกเป็นตัวเลข (Number) จากนั้นจึงใส่ค่าอุณหภูมิที่ต้องการลงไป (ในที่นี้จะใช้ 25 องศา)



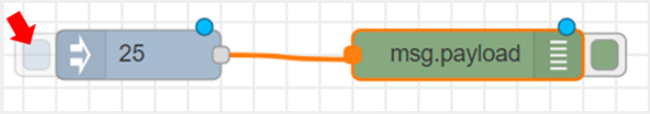
3. นำกล่องฟังก์ชั่น **'Debug'** ที่อยู่ในหมวดหมู่ ***'Common'*** ลงมาวางในพื้นที่ผังงาน



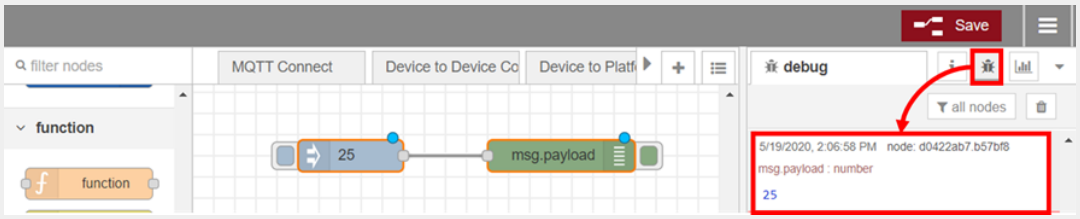
4. เชื่อมต่อกล่องฟังก์ชั่นทั้งสองกล่องเข้าด้วยกัน โดยให้ทำการลากเชื่อมจากจุดสีเทาด้านขวาของกล่องฟังก์ชั่น **‘Inject’** ไปยังจุดสีเทาที่อยู่ด้านซ้ายของกล่อง **‘Debug’** เพื่อให้ข้อมูลอุณหภูมิที่กำหนดไว้ในกล่อง **‘Inject’** นั้นส่งไปแสดงผลยังกล่องฟังก์ชั่น **‘Debug’**



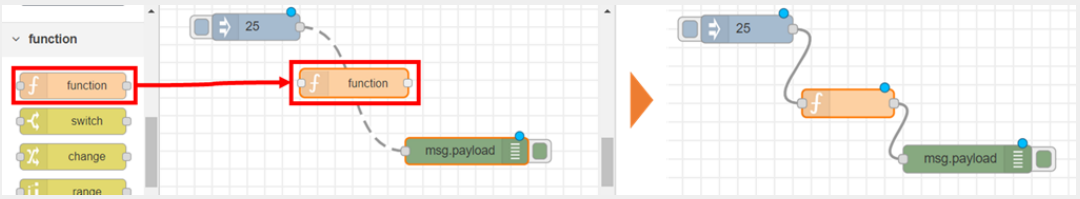
5. บันทึกหน้าผังงานนี้โดยการกดที่ปุ่ม ‘Save’ ด้านบนของโปรแกรม จากนั้นจึงทำการคลิกที่ปุ่มที่อยู่ข้าง ๆ กล่อง **‘Inject’** เพื่อเป็นการทดลองส่งข้อมูลออกไป



6. ผู้ใช้งานทำการเปิดหน้าต่าง debug ด้านขวาขึ้นมา จะพบว่าข้อมูลอุณหภูมิแบบเซลเซียสที่กำหนดไว้นั้นถูกส่งมาเรียบร้อย อย่างไรก็ตาม ค่าข้อมูลที่ถูกส่งมานั้นยังไม่ถูกเปลี่ยนแปลงหน่วยเป็นฟาเรนไฮต์ดังที่หนดเอาไว้ จึงจำเป็นต้องมีการแก้ไขบางส่วนเพิ่มเติมเพื่อให้ข้อมูลถูกต้อง

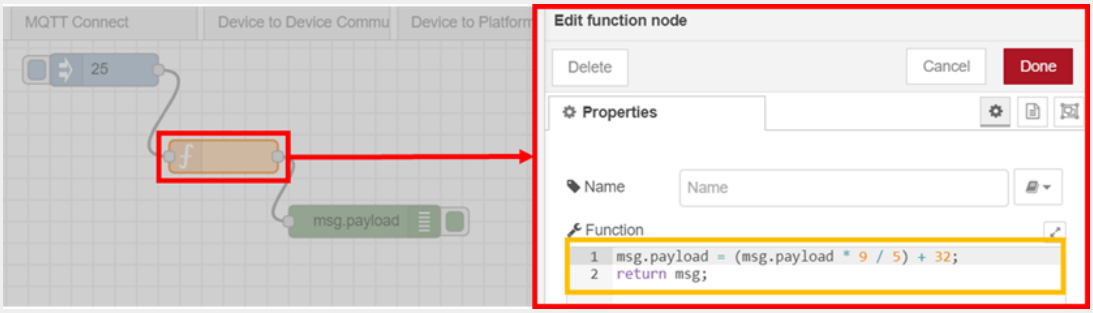


7. ทำการลากกล่อง **‘Function’** ในหมวดหมู่ ***‘Function’*** ลงมา โดยให้ทำการวางแทรกกลางไว้ระหว่างกล่องฟังก์ชั่นเดิมทั้งสองกล่องหรือทับบนเส้นการเชื่อมต่อเดิมที่มีอยู่ โดยจะทำให้เส้นการเชื่อมต่อนั้นกลายเป็นเส้นประ และเมื่อวางสำเร็จ กล่องฟังก์ชั่นทั้งสามกล่องจะเชื่อมต่อกันโดยอัตโนมัติ



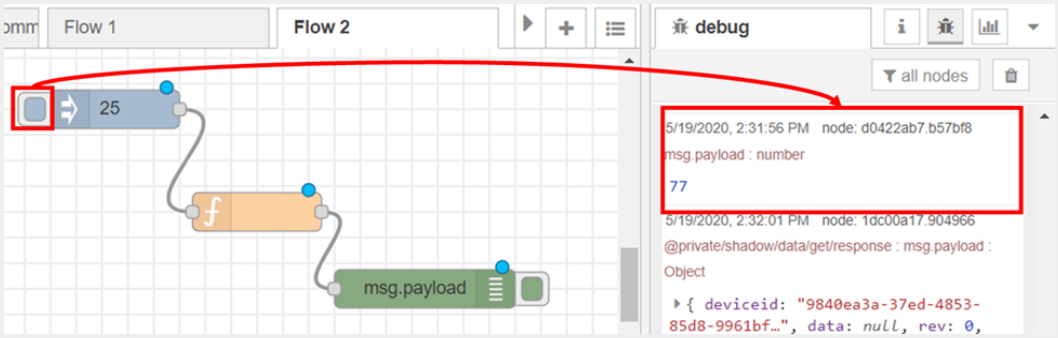
8. คลิกที่กล่อง **‘Function’** เพื่อเปิดการตั้งค่าด้านในขึ้นมา โดยหน้าต่างจะปรากฏพื้นที่ให้ผู้ใช้งานเขียนโค้ดเพิ่มเติมเพื่อเปลี่ยนแปลงค่าจากกล่องฟังก์ชั่นก่อนหน้าและคืนค่าเป็นค่าใหม่ได้ (ซึ่งในที่นี้ใช้สูตรในการ

คำนวณเปลี่ยนแปลงอุณหภูมิจากองศาเซลเซียสให้กลายเป็นองศาฟาเรนไฮต์) จากนั้นจึงกด **'Done'** เมื่อทำการเปลี่ยนแปลงสำเร็จ



<u>ข้อควรรู้</u> ข้อมูลที่ถูกส่งเข้ามาจากกล่องฟังก์ชั่นก่อนหน้านั้นถูกบรรจุเป็นค่าตัวแปรที่ชื่อว่า *'Payload'* อยู่ใน Object ที่ชื่อว่า *'msg'* ซึ่งหากผู้ใช้งานต้องการเข้าถึงข้อมูลตัวแปรนี้ ผู้ใช้งานสามารถเขียน ***'msg.payload'*** เพื่อเข้าถึงค่าข้อมูลตัวแปรนั้นได้

9. ทำการบันทึกหน้าผังงานนี้ จากนั้นจึงกดปุ่มที่กล่องฟังก์ชั่น **'Inject'** อีกครั้ง แล้วตรวจสอบค่าข้อมูลที่ถูกส่งออกไปในหน้าต่าง debug จะพบว่าค่าข้อมูลที่เข้ามานั้นอยู่เปลี่ยนไปตามสูตรที่กำหนดไว้เรียบร้อย




```
[{"id":"42435e70.dc83d","type":"inject","z":"b97e3fb1.484b","name":"","topic":"","payload":"25","payl
oadType":"num","repeat":"","crontab":"","once":false,"onceDelay":0.1,"x":610,"y":1380,"wires":[["3254
da6e.2910e6"]]},{"id":"d0422ab7.b57bf8","type":"debug","z":"b97e3fb1.484b","name":"","active":true,"t
osidebar":true,"console":false,"tostatus":false,"complete":"false","x":850,"y":1520,"wires":[]},{"id"
:"3254da6e.2910e6","type":"function","z":"b97e3fb1.484b","name":"","func":"msg.payload = (msg.payload
* 9 / 5) + 32;\nreturn msg;","outputs":1,"noerr":0,"x":720,"y":1460,"wires":[["d0422ab7.b57bf8"]]}]
```


Comment

ในโฟลว์ที่มีความซับซ้อนหรือมีความยุ่งเหยิงจากการเชื่อมต่อของโหนดมากมายเข้าด้วยกัน ผู้ใช้งานสามารถใส่โหนดที่ไม่ได้รับหรือส่งค่าข้อมูลใด ๆ แต่ต้องการใส่คำอธิบายเพิ่มเติมลงไปใน

โฟลว์เพื่อความเป็นระเบียบเรียบร้อยและเพื่อสร้างความเข้าใจให้กับการอ่านโฟลว์ในภายหลังได้ โดยให้ผู้ใช้งานนำโหนด **'Comment'** ในหมวดหมู่ ***'Common'*** ลงมาใช้งาน

การส่งผ่านข้อความ

การทำงานโฟลว์ภายในโฟลว์นั้น จะใช้วิธีการส่งข้อความถึงกันระหว่างแต่ละโหนด โดยค่าข้อความเหล่านั้นที่ส่งไปจะออกไปในลักษณะของอ็อบเจกต์ (Object) อย่างง่ายของ JavaScript ซึ่งจะประกอบไปด้วยตัวแปรต่าง ๆ ภายใน ซึ่งค่าเริ่มต้นของข้อความที่ถูกส่งออกไปนี้จะถูกเรียกว่า `msg` ซึ่งภายในอ็อบเจกต์ `msg` จะประกอบไปด้วยตัวแปรหลัก ๆ ชื่อ `payload` ซึ่งเป็นตัวแปรเริ่มต้นที่โหนดส่วนใหญ่ในโฟลว์จะนำไปใช้ นอกเหนือจากนั้นยังมีตัวแปรชื่อ `_msgid` ที่ใช้ระบุตัวตนของข้อความเพื่อใช้ในการติดตามการเคลื่อนไหวของโฟลว์ได้อีกด้วย

```
{ "_msgid": "12345", "payload": "..." }
```

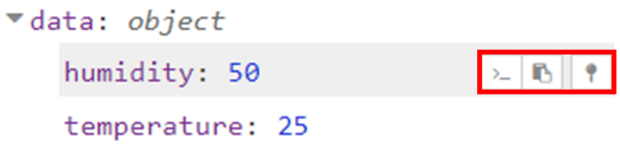
ในการเข้าถึงค่าตัวแปรเหล่านี้ที่อยู่ภายในอ็อบเจกต์ `msg` ผู้ใช้งานสามารถทำได้โดยการใช้ . ต่อท้ายอ็อบเจกต์ `msg` เพื่อเรียกชื่อตัวแปรที่อยู่ภายใน `msg` มาใช้งาน ยกตัวอย่างเช่น หากต้องการเรียกดูค่าข้อมูล `payload` ที่อยู่ภายในอ็อบเจกต์ `msg` ผู้ใช้งานสามารถใช้ `msg.payload` ในการเรียกค่าข้อมูลนั้นออกมาดูได้ เป็นต้น

ค่าข้อมูลที่ถูกส่งควบคู่ไปกับค่าตัวแปรต่าง ๆ นั้นล้วนแต่ต้องเป็นค่าตัวแปรที่สามารถยอมรับได้โดยภาษา Javascript ทั้งสิ้น เช่น ค่าความจริง (Boolean; `true`, `false`) ตัวเลข (Number) ค่าข้อความ (String) รายการข้อมูล (Array; เช่น `[1,2,3,4]` เป็นต้น) อ็อบเจกต์ (Object; เช่น `{"a": 1, "b": 2}` เป็นต้น) หรือค่าข้อมูลที่ไม่มีข้อมูล (Null) เป็นต้น

ในการดูข้อความที่ถูกส่งไปเหล่านี้ ผู้ใช้งานสามารถตรวจสอบได้โดยการส่งข้อความไปยังโหนด **'Debug'** และดูค่าข้อมูลในแถบเมนู Debug (🐞) ด้านขวามือของหน้าต่างโปรแกรม

- ด้านบนสุดจะแสดงชื่อของตัวแปรที่ถูกส่งไป ซึ่งค่าเริ่มต้น `msg.payload` จะถูกนำมาใช้งานในส่วนนี้
- ถัดจากชื่อของตัวแปรคือประเภทของค่าที่ถูกส่งไป เช่น Object, String, Array เป็นต้น
- ถัดลงมาด้านล่างจะแสดงผลค่าข้อมูลต่าง ๆ ที่ถูกนำส่งไปซึ่งผู้ใช้งานสามารถย่อ-ขยายการแสดงค่าข้อมูลได้โดยการคลิกที่ลูกศรที่อยู่ด้านข้างของค่าข้อมูลนั้น ๆ

หากผู้ใช้งานทำการนำเมาส์ไปวางเหนือค่าข้อมูลใด ๆ (ปรากฏเป็นแถบสีเทาที่ค่าข้อมูลนั้น) จะแสดงให้เห็นปุ่มเล็ก ๆ ที่อยู่ด้านบนขวาของข้อมูล:



- '>_' จะเป็นการคัดลองที่อยู่ของค่าข้อมูลนี้ออกไปใช้งาน ซึ่งในรูปจะเป็นการคัดลอก `payload.data.humidity` ออกมา

  ข้อสังเกต ข้อความที่ถูกส่งอยู่ภายในโฟลว์นั้นมักจะถูกบรรจุอยู่ใน `msg.payload` เสมอ ดังนั้นเมื่อผู้ใช้งานต้องการเรียกค่าข้อมูลที่อยู่ในข้อความที่ส่งไป จึงมักจะต้องมีการอ้างอิงถึง `payload` ดังตัวอย่างข้างต้น

- ' ' จะเป็นการคัดลอกค่าข้อมูลนั้นออกมาใช้งานโดยตรง เช่น ในรูปจะทำการนำ '50' คัดลอกออกมา เป็นต้น
- ' ' จะเป็นการปักหมุดข้อมูลไว้ให้แสดงผลทุกครั้งที่มีการรับค่าข้อมูลนี้เข้ามาภายในโหนด **'Debug'** เดิมที่ใช้งานอยู่ โดยโปรแกรมจะทำการขยายค่าข้อมูลและแสดงให้เห็นค่าข้อมูลในตัวแปรนี้ที่ถูกปักหมุดเอาไว้ออกมาเสมอ

การใช้งานโหนด Function

'Function' คือโหนดที่จะใช้ในการเปลี่ยนแปลงหรือแก้ไขข้อมูลหรือข้อความที่ถูกส่งเข้ามาภายในโหนดนี้ผ่านการเขียนโปรแกรมในภาษา JavaScript ซึ่งข้อความนั้นมักจะถูกบรรจุมาในตัวแปรที่ชื่อ `msg` และภายในตัวแปรนี้ก็มักจะมี `payload` ซึ่งเป็นค่าข้อมูลหลักที่จะถูกนำมาใช้งานเสมอ โดยผู้ใช้งานสามารถเข้าถึงค่านี้ได้โดยการเรียกผ่านตัวแปรหลักว่า `msg.payload`

การเขียนฟังก์ชัน

โปรแกรมที่ถูกเขียนบรรจุเข้าไปในโหนดนี้จะถูกเรียกว่า body ซึ่งโดยพื้นฐานแล้ว โหนดนี้ได้เขียนโปรแกรมขั้นต้นเอาไว้ดังตัวอย่างของชุดคำสั่งนี้

```
return msg;
```

ชุดคำสั่งเริ่มต้นนี้จะทำการรับค่าข้อมูลที่เข้ามาภายในโหนดแล้วส่งต่อออกไปโดยไม่ได้เปลี่ยนแปลงค่าใด ๆ ค่าข้อมูลอย่างไรก็ออกไปอย่างนั้น ฉะนั้นหากโหนดนี้ส่งค่า null ออกไปจากโหนด นั่นหมายถึงไม่มีค่าข้อมูลใด ๆ ส่งผ่านเข้ามาในโหนดนี้

โหนดฟังก์ชันนั้นจำเป็นต้องส่งค่า `msg` ออกมาเสมอ อย่างไรก็ดี ผู้ใช้งานสามารถเปลี่ยนแปลงหรือแก้ไขข้อมูลเหล่านั้นก่อนที่จะส่งต่อไปให้ยังโหนดอื่น ๆ ได้ โดยสุดท้ายแล้วข้อความที่ออกไปนั้นไม่จำเป็นต้องเป็นค่าเดียวกันกับค่าที่เข้ามาภายในโหนดเลยด้วยซ้ำ นั่นหมายถึงอ็อบเจกต์ที่เข้ามาภายในโหนดนี้อาจจะมีค่าคุณสมบัติภายในที่แตกต่างจากอ็อบเจกต์สุดท้ายที่ถูกสร้างขึ้นมาภายในโหนดฟังก์ชัน

```
var newMsg = { payload: msg.payload.length }
return newMsg;
```

สิ่งเดียวที่ผู้ใช้งานควรระวังในการเปลี่ยนแปลงค่าข้อมูลเมื่อจะส่งออกไปคือ มันอาจเป็นการทำลายบางโฟลว์ให้มีข้อผิดพลาดเกิดขึ้นได้ ฉะนั้นแล้วโหนดฟังก์ชันจึงควรจะส่งค่าข้อมูลที่มีตัวแปรเดิมออกไปจากโหนดฟังก์ชันนี้ด้วย

ส่งออกข้อมูลหลายค่าในครั้งเดียว

โหนดฟังก์ชันนั้นนอกเหนือจากจะสามารถเปลี่ยนแปลงค่าข้อมูลที่ไหลผ่านเข้ามาได้แล้วนั้น ยังสามารถส่งค่าข้อมูลออกจากโหนดนี้ได้มากกว่าหนึ่งค่าอีกด้วย ซึ่งจำนวนข้อมูลที่ผู้ใช้งานต้องการให้ส่งออกไป สามารถตั้งค่าได้ที่โดยการเปิดหน้าการตั้งค่าของโหนดนี้ขึ้นมาแล้วเปลี่ยนตัวเลขที่ ‘Output’ การส่งออกข้อมูลหลายค่านี้มีประโยชน์ในบางกรณีที่ต้องการจะส่งค่าข้อมูลเดียวกันจากการแปลงผลที่แตกต่างกันไปให้ยังโหนดที่แตกต่างกัน ตัวอย่างเช่น ผู้ใช้งานอาจต้องการส่งค่าที่ได้รับ `topic` มาว่า ‘banana’ ไปให้ยังหัวข้อส่งออกที่สองแทนจุดเชื่อมต่อส่งออกแรกของโหนด

```
if (msg.topic == “banana”) {
      return [ null, msg ];
} else {
      return [ msg, null ];
}
```

หรืออีกตัวอย่างในการส่งค่าข้อมูลเดิมไปยังจุดเชื่อมต่อขาออกแรก และความยาวของค่า `payload` ไปยังจุดเชื่อมต่อขาออกที่สอง

```
var newMsg = { payload: msg.payload.length };
```

```
return [msg, newMsg];
```

Flow Engine และ NEXPIE

NEXPIE ได้ทำการนำ Flow Engine เข้ามาใช้งานเพื่อให้ผู้ใช้งานสามารถสื่อสารและควบคุมอุปกรณ์ได้ผ่านหน้าต่างโปรแกรม Flow Engine โดยการตั้งค่าให้มีการรับค่าข้อมูลจากอุปกรณ์ที่สร้างเอาไว้ในหน้า https://portal.nexpie.io ซึ่งการเชื่อมต่ออุปกรณ์เข้ามาใช้งานในหน้า Flow Engine นั้นสามารถทำได้ 2 วิธี ได้แก่ การเชื่อมต่ออุปกรณ์เข้ากับแพลตฟอร์ม NETPIE/NEXPIE และการเชื่อมต่ออุปกรณ์ผ่าน MQTT Broker โดยทั้ง 2 วิธีต่างก็นำโหนดที่อยู่ในหมวดหมู่ ***'Device'*** มาใช้งาน

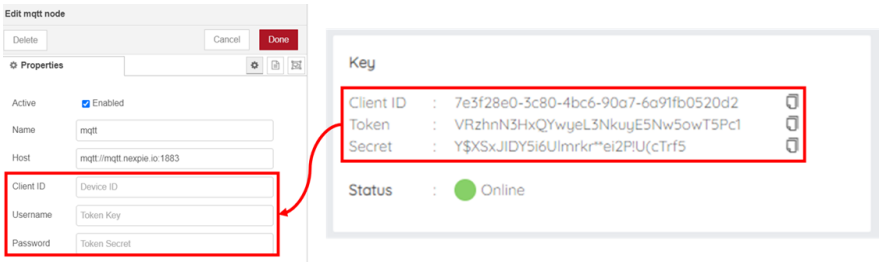
การสร้างอุปกรณ์ใหม่

ภายในหน้าต่าง Flow Engine ผู้ใช้งานสามารถสร้างอุปกรณ์เสมือนขึ้นมาเพื่อใช้ในการรับและส่งค่าข้อมูลภายในโฟลว์ได้ โดยใช้หลักการเดียวกันกับการเชื่อมต่อกับอุปกรณ์ที่มีอยู่จริง ซึ่งผู้ใช้งานจำเป็นต้องใช้ค่า key จากอุปกรณ์ที่ถูกสร้างไว้ในหน้า https://portal.nexpie.io มาใส่ลงไปในโหนดที่ผู้ใช้งานเลือกภายในโฟลว์ ซึ่งผู้ใช้งานสามารถสร้างอุปกรณ์ใหม่ขึ้นมาได้ 2 วิธี

MQTT

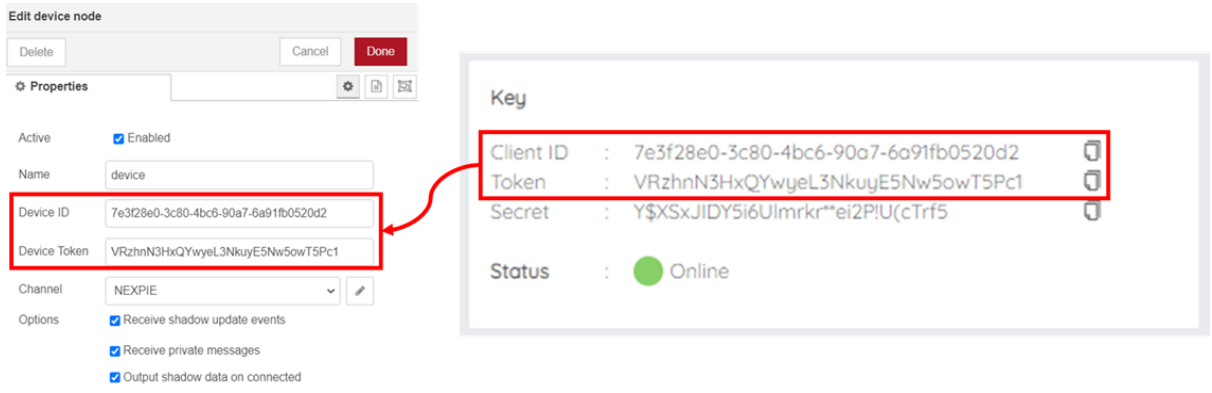
'MQTT' คือโหนดที่จะช่วยให้ผู้ใช้งานสามารถสร้างอุปกรณ์ใหม่ในโฟลว์ได้โดยใช้การเชื่อมต่อในลักษณะของ MQTT ซึ่งถูกออกแบบมาให้อุปกรณ์สามารถสื่อสารและเชื่อมต่อระหว่างกันได้โดยมีคนกลาง (Broker) ที่ทำหน้าที่รับข้อมูลจากอุปกรณ์หนึ่ง เพื่อส่งออกไปให้ยังอีกอุปกรณ์ที่อยู่ภายในหัวข้อการสนทนาเดียวกันได้อย่าง Real-time

ผู้ใช้งานต้องนำค่า Client ID, Token, และ Secret จากอุปกรณ์ที่ถูกสร้างเอาไว้ภายใน https://portal.nexpie.io มากรอกลงไปในโหนดนี้ โดยให้ใช้ Client ID แทนค่า Client ID, ใช้ Token แทนค่า Username, และใช้ Secret แทนค่า Password ซึ่งการใช้งานโหนดนี้นั้น ผู้ใช้งานจำเป็นต้องกำหนดตำแหน่งของ Host ที่ต้องการให้โหนดเชื่อมต่อด้วยตัวเอง ดังในรูปด้านล่างจะเป็นการเชื่อมต่อกับ NEXPIE ผ่านตำแหน่งของ Host ที่ mqtt://mqtt.nexpie.io:1883



Device

เช่นเดียวกับ MQTT โหนด **‘Device’** อนุญาตให้ผู้ใช้งานสร้างอุปกรณ์เสมือนใหม่ขึ้นมาในโฟลว์ได้ ซึ่งค่า key ที่นำมาใส่ลงไปในโหนดนี้นั้นคือค่า Client ID และ Token จากอุปกรณ์ที่ถูกสร้างเอาไว้ภายใน https://portal.nexpie.io ซึ่งผู้ใช้งานต้องนำค่าเหล่านี้มากรอกลงไปในโหนดนี้ โดยให้ใช้ Client ID แทนค่า Device ID และใช้ Token แทนค่า Device Token จากนั้นให้ผู้ใช้งานกำหนดว่าค่าของอุปกรณ์นั้น ผู้ใช้งานได้มาจากอุปกรณ์ที่สร้างไว้ภายใน NETPIE หรือ NEXPIE หรือผู้ใช้งานสามารถกำหนดค่า MQTT Channel ที่นำมาใช้งานได้ด้วยตัวเอง



สถานะการเชื่อมต่อ

หากอุปกรณ์ที่สร้างเอาไว้สามารถเชื่อมต่อได้สำเร็จ โหนด **‘Device’** และ **‘MQTT’** จะขึ้นสถานะ *‘Connected’* อยู่ด้านล่างของโหนดนั้นพร้อมสัญญาณสีเขียว เช่นเดียวกับอุปกรณ์ที่จะขึ้นสถานะ *‘Online’* พร้อมสัญญาณไฟสีเขียวในหน้า https://portal.nexpie.io ในขณะที่อุปกรณ์ที่ไม่ได้ถูกทำให้เชื่อมต่อไว้จะปรากฏเป็นสถานะ *‘Inactive’* ที่ตัวโหนดนั้นแทนเพื่อแสดงให้เห็นว่ายังไม่สามารถเชื่อมต่ออุปกรณ์เข้ากับโหนดนั้นในโฟลว์ได้

ส่วนอุปกรณ์ที่มีข้อผิดพลาดเกิดขึ้นมาระหว่างการสร้างโฟลว์ สถานะของโหนดนั้นภายในโฟลว์จะถูกแสดงขึ้นมาว่า *'Disconnected'* ปรากฏขึ้นที่ด้านล่างของโหนดนั้น

ข้อจำกัดในการสร้างอุปกรณ์ใหม่

อย่างไรก็ดี ในการสร้างอุปกรณ์ขึ้นมาใหม่นั้น ผู้ใช้งานควรระวังไม่สร้างอุปกรณ์ที่ใช้ค่า key เดียวกันมากกว่าหนึ่งตัว ไม่ว่าอุปกรณ์นั้นจะเป็นอุปกรณ์เสมือนที่ถูกสร้างขึ้นในโฟลว์หรือเป็นอุปกรณ์ที่มีอยู่จริงก็ตาม ทั้งนี้เพื่อป้องกันไม่ให้เกิดปัญหาในการเชื่อมต่อของอุปกรณ์ต่าง ๆ ที่เมื่อต้องทำการพร้อมกันทั้งสองต่อ ระบบจะทำการผลักหนึ่งในอุปกรณ์ออกจากการเชื่อมต่อทันทีเพื่อให้มีอุปกรณ์เพียงตัวเดียวที่เชื่อมต่อกับแพลตฟอร์มโดยใช้ค่า key นั้น ซึ่งอาจจะก่อให้เกิดปัญหาถึงข้อมูลจากอุปกรณ์ที่ถูกส่งไปยังแพลตฟอร์มมีความผิดเพี้ยนไปได้

อีกกรณีหนึ่งคือ หากอุปกรณ์ใดอุปกรณ์หนึ่งของผู้ใช้งานถูกเจาะระบบเข้ามาจนค้นพบค่า key ของอุปกรณ์นั้น อาจส่งผลให้ทุกอุปกรณ์ที่ใช้ key ถูกขโมยข้อมูลไปอย่างง่ายดายจากผู้ไม่หวังดีที่พึงประสงค์จะเจาะระบบเข้ามาดึงข้อมูลเหล่านี้ไปได้อีกด้วย

เพราะฉะนั้นแล้ว หากผู้ใช้งานจำเป็นต้องเชื่อมต่อกับอุปกรณ์ที่ถูกนำ key ของเขาไปใช้ในพื้นที่อื่นอยู่แล้ว ผู้ใช้งานอาจใช้โหนด **'Shadow'** หรือ **'Sniffer'** ในการเข้าถึงอุปกรณ์ต่าง ๆ เหล่านั้นแทนได้

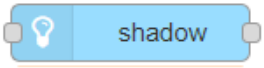
เข้าถึงข้อมูลจากอุปกรณ์ที่มีอยู่

โดยปกติแล้ว ค่า key ของอุปกรณ์ที่ผู้ใช้งานสร้างเอาไว้ใน https://portal.nexpie.io นั้นควรใช้ควบคู่กับอุปกรณ์เพียงตัวเดียวเท่านั้นเพื่อป้องกันปัญหาของการเชื่อมต่อระหว่างอุปกรณ์หลาย ๆ

ชิ้นพร้อม ๆ กัน การเข้าถึงอุปกรณ์ที่ผู้ใช้งานเคยสร้างเอาไว้แล้วนั้นจึงไม่ควรเป็นการนำค่า key ของอุปกรณ์นั้นมาใส่ในโหนด **‘Device’** หรือ **‘MQTT’** ตรง ๆ เพื่อรับหรือส่งข้อมูลให้กับอุปกรณ์นั้น

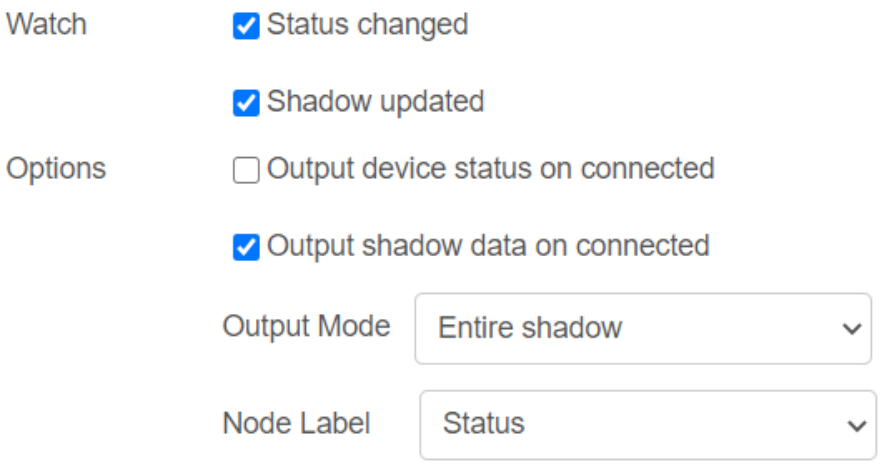
ในทางกลับกัน ผู้ใช้งานสามารถใช้โหนด **‘Shadow’** เพื่อรับและส่งข้อมูลให้กับอุปกรณ์ของผู้ใช้งาน และ **‘Sniffer’** เพื่อรับและส่งข้อความของอุปกรณ์นั้นแทนได้ ซึ่งจะช่วยให้ผู้ใช้งานยังคงสามารถมีปฏิสัมพันธ์กับโหนดเดิมได้โดยไม่ก่อให้เกิดปัญหาการเชื่อมต่อในภายหลัง

Shadow



‘Shadow’ คือโหนดที่อนุญาตให้ผู้ใช้งานสามารถดึงข้อมูล Shadow จากอุปกรณ์หรือนำค่า Shadow ใหม่ที่ถูกสร้างขึ้นมาภายในโฟลว์ส่งไปให้ยังอุปกรณ์ที่มีอยู่แล้วได้ โดยการใช้โหนดนี้จะไม่ใช่การเชื่อมต่อกับอุปกรณ์นั้นโดยตรง แต่จะมีตัวกลาง (Broker) ที่คอยทำการเชื่อมต่อทั้งสองเข้าด้วยกัน ซึ่งจะช่วยในการตัดทอนปัญหาที่เกิดขึ้นจากการเชื่อมต่อหลาย ๆ อุปกรณ์ด้วยค่า key เดียวกันออกไป

ค่าของข้อมูลภายในโหนดนี้จะผูกพันกับค่าที่ได้รับจากอุปกรณ์จริง ซึ่งประโยชน์ของโหนดนี้คือจะทำให้ผู้ใช้งานสามารถจัดการกับอุปกรณ์อื่น ๆ ภายนอกได้โดยตรง ทั้งการรับผลค่าข้อมูลต่าง ๆ ของอุปกรณ์มาใช้งาน รวมไปถึงการสร้างค่าข้อมูลใหม่แล้วส่งไปยังอุปกรณ์นั้นก็ทำได้เช่นกัน



สืบเนื่องมาจากโหนด **‘Shadow’** นี้เปรียบเสมือนตัวแทนของค่าอุปกรณ์ที่ใช้งาน key ต่าง ๆ ที่ตั้งค่าไว้ภายในนี้อยู่ ผู้ใช้งานจึงสามารถตั้งค่าได้ว่าต้องการให้โหนดนี้คอยเฝ้าระแวดระวังหรือตรวจ

สอบค่าข้อมูลที่มีการเปลี่ยนแปลงใดของโหนดนี้บ้างระหว่างสถานะของอุปกรณ์หรือค่าข้อมูล Shadow หรือให้ตรวจสอบค่าข้อมูลทั้งสองค่า

ในขณะเดียวกัน ผู้ใช้งานยังสามารถเลือกได้ว่าต้องการให้โหนดนี้แสดงผลสถานะของอุปกรณ์หรือค่าข้อมูล Shadow เมื่อมีการเชื่อมต่อเกิดขึ้นหรือไม่ และยังสามารถเลือกได้อีกว่าต้องการให้แสดงผลค่าข้อมูลออกมาเฉพาะค่าตัวแปรที่มีการเปลี่ยนแปลงเท่านั้น หรือต้องการให้แสดงผลข้อมูล Shadow ทั้งหมดออกมา

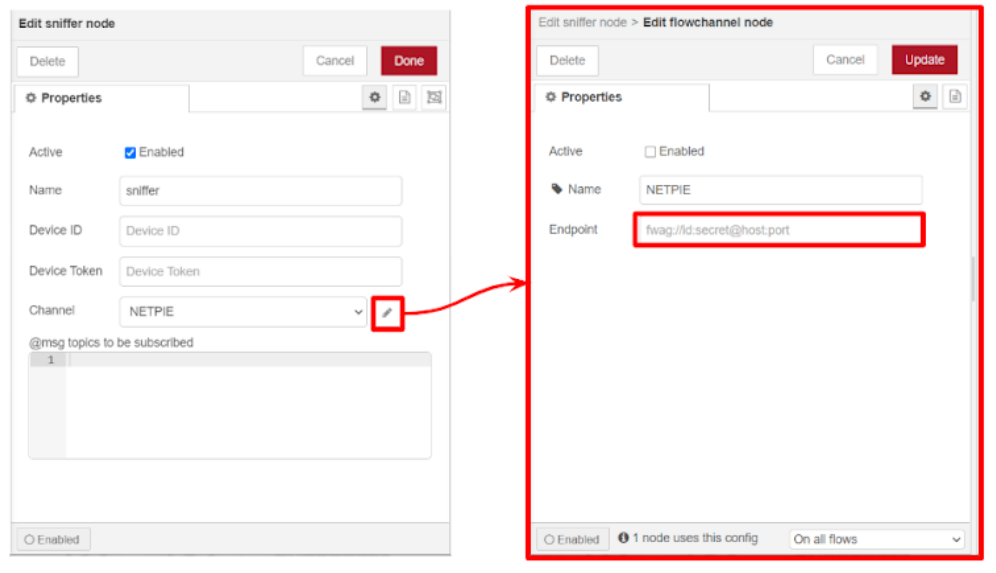
Sniffer

‘Sniffer’ คือโหนดที่อนุญาตให้ผู้ใช้งานสามารถติดตามข้อความที่ถูกส่งมาจากอุปกรณ์อื่น ๆ ได้ในลักษณะของการดักฟัง โดยใช้ Device ID และ Token ของอุปกรณ์ที่ต้องการดักฟังมากรอกใส่ลงไป เช่นเดียวกับหัวข้อ (Topic) ของข้อความที่ต้องการจะดักฟัง ซึ่งลักษณะของการทำงานในการรับ-ส่งข้อความระหว่างอุปกรณ์นั้นจะถูกอธิบายในหัวข้อถัดไป

อย่างไรก็ดี นอกเหนือจากการใส่ค่า key ต่าง ๆ ลงไปแล้ว ผู้ใช้งานยังจำเป็นต้องกำหนด Endpoint ของตัวอุปกรณ์ที่ต้องการให้เชื่อมต่อลงไปด้วย

ข้อจำกัดในการใช้งาน

การใช้งานโหนด **‘Shadow’** และ **‘Sniffer’** นั้นมีข้อจำกัดอยู่เล็กน้อยคือ ผู้ใช้งานจำเป็นต้องทำการกำหนดค่า Endpoint ลงไปในโหนดทั้งสองประเภทเสียก่อนจึงจะสามารถใช้งานโหนดเหล่านี้ได้ โดยวิธีการใส่ค่า Endpoint ลงไปนั้น ให้ผู้ใช้งาน double-click เข้าไปในโหนดเพื่อเปิดหน้าการแก้ไขขึ้นมา แล้วคลิกที่ปุ่มดินสอข้าง ๆ ส่วนการตั้งค่า Channel เพื่อทำการเปิดหน้าต่างที่ใช้ในการใส่ค่านี้ขึ้นมา จากนั้นทำการคลิกที่ปุ่ม ‘Update’ หลังจากใส่ค่า Endpoint เรียบร้อยแล้ว เพียงเท่านี้โหนดทั้งสองก็จะสามารถใช้งานได้แล้ว



การส่งข้อความระหว่างอุปกรณ์

อุปกรณ์แต่ละตัวจะส่งข้อความถึงกันและกันโดยยึดหลักการส่งข้อความในลักษณะของ MQTT ซึ่งจะเป็นการกระจายข้อความ (Publish) และติดตามข้อความ (Subscribe) ที่ถูกส่งไปยังหัวข้อต่าง ๆ (Topic) ที่ถูกกำหนดขึ้น โดยผู้รับสารที่ติดตามหัวข้อเดียวกันกับหัวข้อที่ผู้กระจายข้อความส่งมาเท่านั้นที่จะได้รับค่าข้อความ หลักการนี้คล้ายคลึงกับการส่งข้อความภายในกลุ่มสนทนาในแอพลิเคชั่นต่าง ๆ โดยหากเมื่อผู้ส่งสารส่งข้อความไปในกลุ่มการสนทนาใด จะมีเพียงผู้ที่อยู่ภายในกลุ่มสนทนานั้นเท่านั้นที่ได้รับข้อความดังกล่าว

ในการรับส่งข้อความด้วยหลักการนี้นั้นจะใช้หัวข้อ (Topic) แทนชื่อของกลุ่มการสนทนา ซึ่งภายในหัวข้อจะต้องเริ่มต้นด้วย ‘@msg’ ตามด้วยโครงสร้างของชื่อหัวข้อที่ต้องการ ประกอบด้วยคำต่าง ๆ ที่ถูกแยกด้วย ‘/’ ยกตัวอย่างเช่น ‘@msg/home/temp’ เป็นต้น

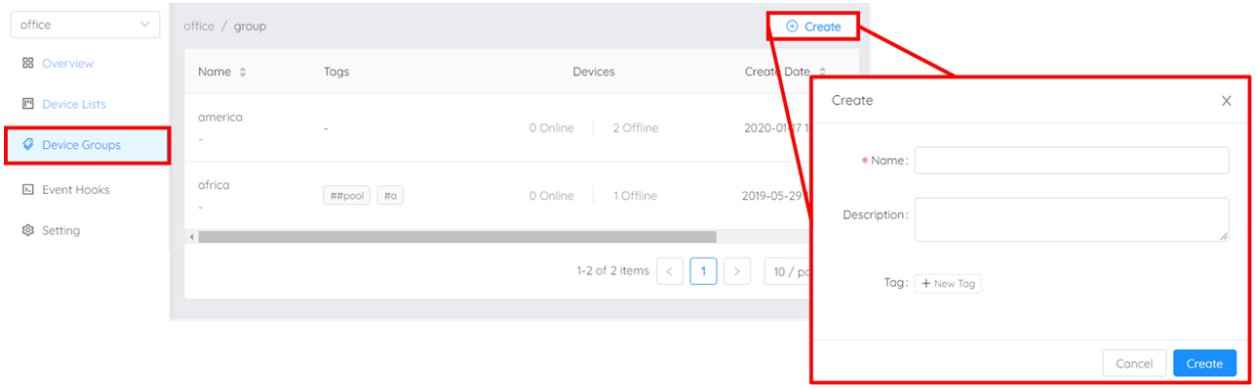
อย่างไรก็ตาม หากผู้ใช้งานต้องการส่งข้อความไปให้ยังอุปกรณ์อื่น ๆ จำนวนมากที่ทำการ subscribe อยู่ในหัวข้อที่แตกต่างกัน ผู้ใช้งานสามารถนำ Wildcard มาช่วยในการเพิ่มความสะดวกสบายในการรับและส่งข้อความได้ โดยสามารถใช้ **'#'** เพื่อใช้แทนค่าคำทุกคำที่อยู่หลัง '/' และใช้ **'+'** เพื่อแทนค่าคำหนึ่งคำที่อยู่ระหว่าง '/' โดยผู้ใช้งานสามารถใช้ค่าทั้งสองค่าร่วมกันได้ แต่การใช้ '#' จำเป็นต้องวางไว้หลังสุดเสมอ เช่น '/+/+/#' เป็นต้น แต่ไม่สามารถใช้ '/#/+' ได้

Subscribe at	Publish to `@msg/home/ground/#`	Publish to `@msg/home/ground/+/temp`
`@msg/home/ground/kitchen`	✔	✕
`@msg/home/ground/kitchen/temp`	✔	✔
`@msg/home/upstairs/bedroom/temp`	✕	✕
`@msg/home/ground`	✔	✕

การส่งข้อความจากโฟลว์

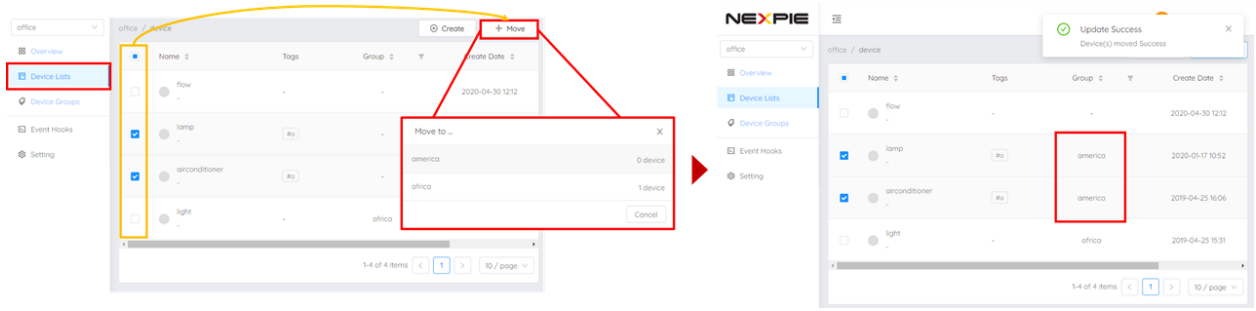
หากผู้ใช้งานต้องการส่งข้อความจากอุปกรณ์ที่อยู่ภายในโฟลว์ไปยังอุปกรณ์อื่น ๆ ผู้ใช้งานสามารถทำได้โดยการนำโหนด **'Publish msg'** ที่อยู่ภายในหมวดหมู่ ***'Command'*** ลงมาใช้งาน

ก่อนอื่น ก่อนการที่อุปกรณ์จะสามารถคุยกันและส่งข้อความหากันได้ อุปกรณ์แต่ละตัวนั้นจะต้องอยู่ภายในกลุ่มการสื่อสารเดียวกันเสียก่อน ซึ่งวิธีการทำให้อุปกรณ์แต่ละตัวอยู่ในกลุ่มการสื่อสารเดียวกันนั้น ผู้ใช้งานต้องไปที่หน้า https://portal.nexpie.io แล้วทำการสร้างกลุ่มที่ต้องการให้อุปกรณ์เหล่านี้ไปอยู่รวมกันขึ้นมา โดยการคลิกที่ปุ่ม '⊕ Create' ภายในหน้า 'Device Groups' เพื่อทำการสร้างกลุ่มใหม่ขึ้น



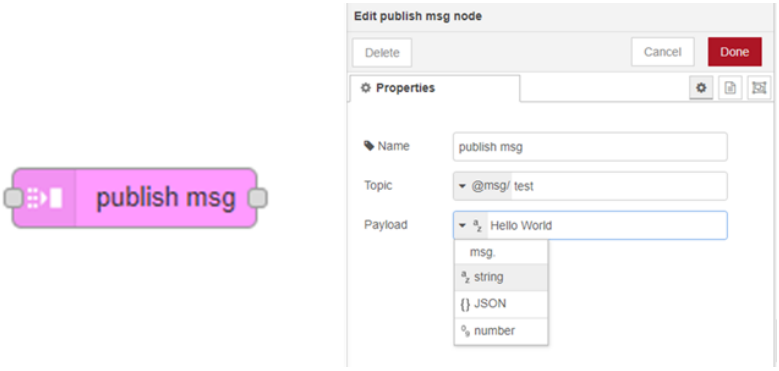
หรือหากในกรณีที่มีกลุ่มที่ถูกสร้างไว้อยู่แล้ว ให้ผู้ใช้งานทำการรวมกลุ่มของอุปกรณ์เหล่านี้เข้าด้วยกัน โดยการคลิกเลือกอุปกรณ์ที่ต้องการรวมไว้ให้อยู่ในกลุ่มเดียวกันจาก checkbox ของ

อุปกรณ์นั้นภายในหน้า ‘Device Lists’ แล้วทำการคลิกที่ปุ่ม ‘✢ Move’ เพื่อให้ปรากฏรายชื่อของกลุ่มที่ผู้ใช้งานเคยสร้างเอาไว้ภายใน portal ขึ้นมา จากนั้นทำการคลิกเลือกกลุ่มที่ต้องการให้อุปกรณ์นี้ไปอยู่ ซึ่งเมื่อทำเช่นนี้เรียบร้อยแล้ว อุปกรณ์นั้นจะปรากฏชื่อของกลุ่มขึ้นมาบนแถบรายละเอียดของอุปกรณ์นั้นในหน้า ‘Device Lists’



เงื่อนไขหลัก ๆ ของการรวมกลุ่มอุปกรณ์เข้าด้วยกันคือ อุปกรณ์แต่ละชิ้นสามารถถูกบรรจุอยู่ได้เพียงในกลุ่มเดียวกันเท่านั้นอุปกรณ์แต่ละชิ้นสามารถบรรจุอยู่ในกลุ่มได้เพียงกลุ่มเดียว และจะมีเพียงอุปกรณ์ที่อยู่ในกลุ่มเดียวกันเท่านั้นที่จะสามารถส่งข้อความเพื่อคุยกันได้

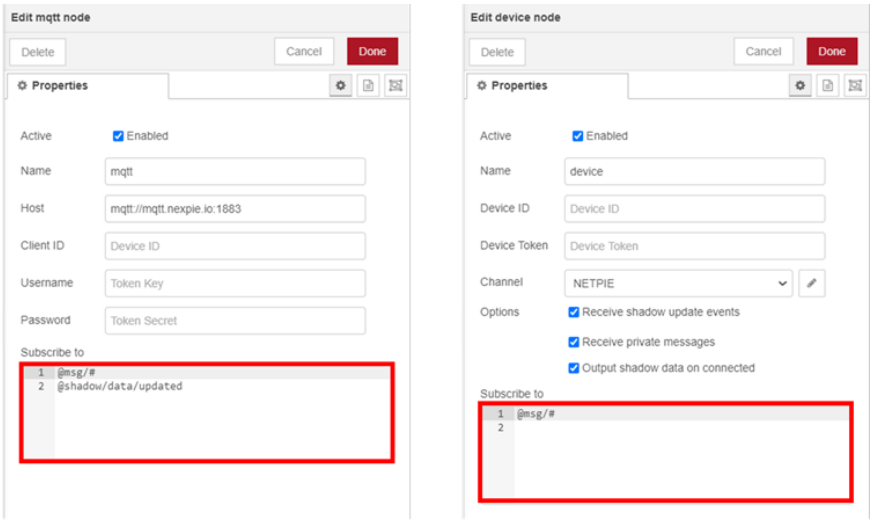
เมื่อทำการจัดกลุ่มให้อุปกรณ์เรียบร้อยแล้ว ในหน้าโฟลว์ ผู้ใช้งานสามารถนำโหนด **‘Publish msg’** มารับค่าข้อความที่ต้องการจะส่งออกไปได้ ซึ่งโหนดนี้นั้นสามารถทำหน้าที่ได้ทั้งเป็นตัวกลางในการรับค่าข้อความจาก `msg` ที่ถูกส่งมาจากโหนดอื่น ๆ หรือสร้างข้อความขึ้นมาเองในลักษณะของ JSON ข้อความ หรือตัวเลขก็ได้เช่นกัน



อย่างไรก็ตาม การตั้งค่าในหมวด ‘Topic’ ของโหนดนี้จะแตกต่างจากคำว่า ‘`topic`’ ในโหนดอื่น ๆ เช่น โหนด **‘Inject’** อยู่พอสมควร ในขณะที่คำว่า `topic` ในโหนดเหล่านั้นคือการตั้งค่าข้อมูลเพิ่มเติมหรือค่าตัวแปรของข้อมูลที่จะถูกส่งไป แต่สำหรับการตั้งค่า Topic ในโหนดนี้คือชื่อของการสื่อสารที่จะทำการส่งออกไปจากโหนดนี้ ดังที่ได้อธิบายไปในหัวข้อของ*การส่งข้อความระหว่างอุปกรณ์* ยกตัวอย่างเช่นดังรูปคือการส่งข้อความออกไปยังอุปกรณ์ที่ทำการติดตามหัวข้อของข้อความที่ชื่อว่า ‘`@msg/test`’ เป็นต้น

การติดตามข้อความภายในโหนด

ในส่วนท้ายภายในโหนด **‘Device’** และ **‘MQTT’** มีช่องการตั้งค่าที่ตั้งชื่อว่า ‘Subscribe to’ เพื่อให้ผู้ใช้งานกำหนด topic ที่ต้องการให้อุปกรณ์เหล่านี้ไปติดตามข้อมูลที่ถูกส่งมาจากอุปกรณ์ต่าง ๆ ผ่าน topic นั้นเอาไว้อยู่ โดยค่าเริ่มต้นที่ทั้งสองโหนดได้ทำการติดตามเอาไว้อยู่คือ ‘`@msg/#`’ ซึ่งหมายถึงการติดตามทุกข้อความที่ถูกส่งผ่านมาในโหนดนี้ ในขณะที่โหนด **‘MQTT’** จะมีการติดตามเพิ่มอีกหนึ่ง topic ได้แก่ ‘`@shadow/data/updated`’ ซึ่งเป็นการติดตามค่าข้อมูลประเภท Shadow ที่จะถูกส่งเข้ามาภายในโหนดนี้แยกไว้อีกโหนดหนึ่งด้วย ซึ่งค่าข้อมูลประเภทนี้จะถูกอธิบายในหัวข้อถัดไป



MQTT Device

การส่งข้อมูลจากอุปกรณ์ไปยังแพลตฟอร์ม

เพื่อนำค่าข้อมูลที่ได้รับจากอุปกรณ์ไปใช้งาน หรือทำการกำหนดค่าใหม่ลงไปในตัวอุปกรณ์นั้น ผู้ใช้งานสามารถตั้งค่าให้อุปกรณ์ผูกเข้ากับแพลตฟอร์มเพื่อรับส่งค่าข้อมูลของกันและกันได้ผ่านตัวแปรที่เรียกว่า *‘Shadow’* ซึ่งจะเป็นตัวกำหนดลักษณะของค่าข้อมูลที่ได้รับและส่งออก

Shadow

Shadow คือฐานข้อมูลเสมือนเล็ก ๆ ของอุปกรณ์ที่อยู่คู่กับอุปกรณ์ทุกตัว ซึ่งใช้ในการเก็บค่าข้อมูล 2 ประเภท ได้แก่ ข้อมูลที่ระบุถึงคุณสมบัติของอุปกรณ์ (Device Shadow Data) เช่น ข้อมูลที่

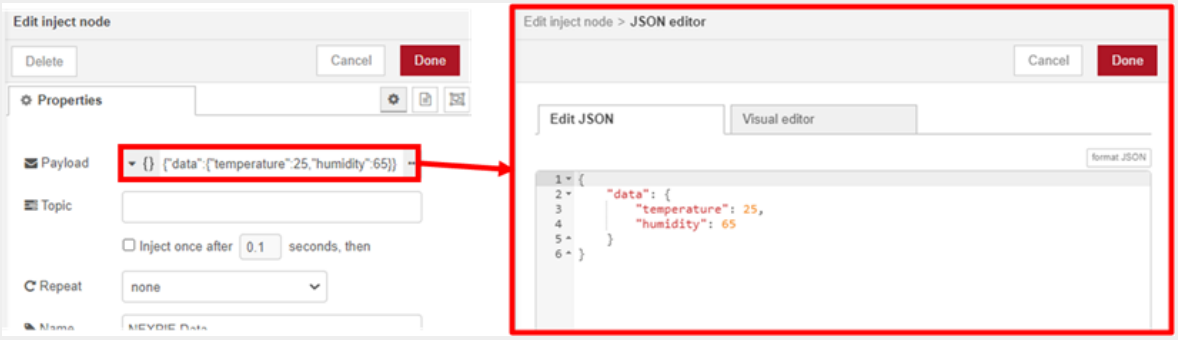
ได้รับเข้ามาจากเซนเซอร์ของอุปกรณ์ หรือข้อมูลการกำหนดค่าต่าง ๆ ของอุปกรณ์ เป็นต้น และข้อมูลที่บอกถึงสถานะของอุปกรณ์ (Device Shadow State) เช่น สถานะเปิด/ปิดของอุปกรณ์ เป็นต้น โดยค่าข้อมูลที่เข้ามาใน Shadow นั้นจะเป็นค่าของอุปกรณ์ ณ เวลาใดเวลาหนึ่งที่จะไม่มีการเก็บค่าย้อนหลังของอุปกรณ์เอาไว้เป็นอันขาด

ลักษณะของค่าข้อมูลที่เข้ามาในรูปแบบ Shadow นั้นจะถูกห่อหุ้มไว้ด้วยค่าตัวแปร data คือ

`{“data”: {“field1”: “value1”, “field2”: “value2”, …}}`

โดย `field` จะหมายถึงชื่อของตัวแปรแต่ละตัวที่อยู่ภายใน Shadow ที่ส่งมา (เช่น ‘*temperature*’ เพื่อใช้แทนค่าอุณหภูมิ หรือ ‘*humidity*’ เพื่อใช้แทนค่าความชื้น เป็นต้น) และ `value` คือค่าข้อมูลของแต่ละตัวแปรนั้น

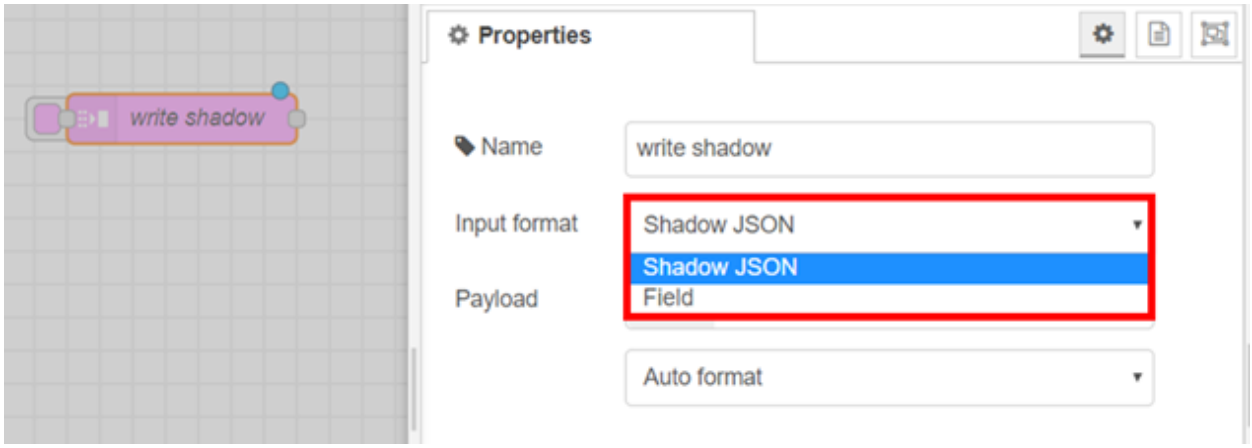
ตัวอย่าง ผู้ใช้งานสามารถสร้างค่าข้อมูล Shadow ขึ้นมาในโฟลว์ได้ผ่านโหนด **‘Inject’** โดยให้กำหนดประเภทของ payload เป็น JSON เพื่อใส่ค่า Shadow ลงไป เช่น หากต้องการให้ค่าข้อมูลที่ใส่ลงไปมีค่าตัวแปร *‘temperature’* เพื่อแทนอุณหภูมิและ *‘humidity’* เพื่อแทนความชื้น ผู้ใช้งานสามารถเขียนลงไปในโหนด **‘Inject’** ได้ดังนี้




```
[{"id":"4a4c9ed5.231f6","type":"inject","z":"8cfb1028.a0d6d","name":"NEXPIE
Data","topic":"","payload":"{\"data\":{\"temperature\":25,\"humidity\":65}}","payloadType":"json","re
peat":"","crontab":"","once":false,"onceDelay":0.1,"x":171.00000762939453,"y":1742.0000505447388,"wir
es":[["c075cb29.fb8738"]]}]
```

ส่งค่าข้อมูล Shadow ไปยังแพลตฟอร์ม

ผู้ใช้งานสามารถเขียนค่าข้อมูล Shadow ไปยังแพลตฟอร์มได้โดยการนำโหนด **‘Write Shadow’** ที่อยู่ในหมวดหมู่ ***‘Command’*** มาใช้งาน โหนดนี้จะทำการนำค่าข้อมูลที่ได้รับมาแปลงให้กลายเป็นค่า Shadow เสียก่อนการจะนำเอาไปใช้งานต่อไป ซึ่งโหนดนี้สามารถรับค่าข้อมูลที่เข้ามาได้สองประเภท ได้แก่



1. **Shadow JSON** คือการรับค่าข้อมูลจากโหนดก่อนหน้าที่ถูกส่งมาในลักษณะของ Shadow JSON อยู่แล้วเพื่อไปใช้งานโดยตรง

   `{"data": {"field1": "value1", "field2": "value2", ...}}`

2. **Field** คือรับค่าข้อมูลจากโหนดหลาย ๆ โหนดมารวมกันเพื่อสร้าง Shadow ใหม่ขึ้นมา จากนั้นจึงสามารถส่งค่าข้อมูลนั้นไปใช้งานอื่นต่อไป โดยการสร้างค่า Shadow นั้นจะอ้างอิงจากค่าข้อมูลในตัวแปรต่าง ๆ ที่อยู่ภายในโหนดก่อนหน้า

   `{"field1": "value1", "field2": "value2", ...}`

หากผู้ใช้งานต้องการเลือกที่จะรับค่าข้อมูลในลักษณะ Field มาใช้งาน ค่าข้อมูลเหล่านั้นจะถูกแปลงให้กลายเป็นค่าข้อมูลในลักษณะ Shadow เสียก่อนเพื่อนำค่า Shadow ที่ได้ส่งไปยังแพลตฟอร์ม ซึ่งจะปรากฏการตั้งค่าเพิ่มเติมขึ้นมาดังนี้

- **Combine each** จะช่วยในการผนวกรวมค่าข้อมูลเข้าด้วยกันเพื่อแปลงค่าข้อมูลเหล่านั้นให้กลายเป็นค่าข้อมูล Shadow ซึ่งจะแปลงเป็นค่าในลักษณะนี้

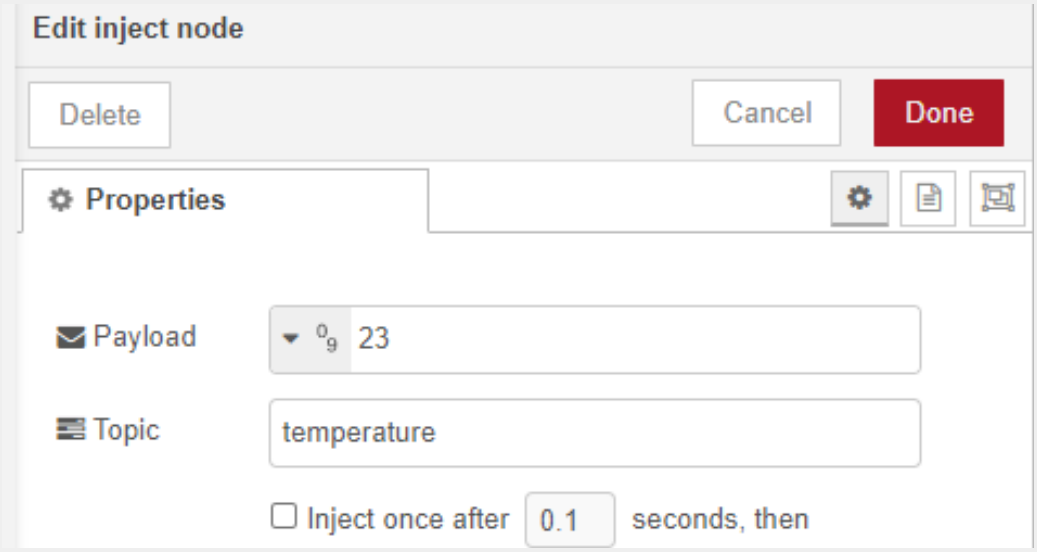
  `{"data": {"key1": "value1", "key2": "value2", ...}}`

  - ***Key*** คือชื่อของตัวแปรหรือ field ที่จะถูกนำไปใส่ใน Shadow โดยค่าเริ่มต้นของค่านี้จะนำ `topic` ที่ถูกส่งมาจากโหนดก่อนหน้ามาใช้งาน
  - ***Value*** คือค่าของตัวแปรนั้นที่จะถูกนำไปใส่ใน Shadow ซึ่งค่าเริ่มต้นของค่านี้คือค่าข้อมูลจาก `payload` ที่ส่งมาจากโหนดก่อนหน้า

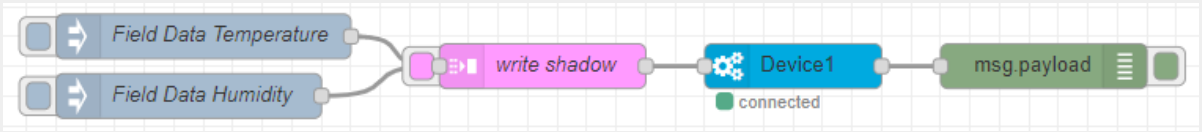
- **Write shadow** คือส่วนการตั้งค่าเพิ่มเติมเกี่ยวกับการเขียนข้อมูลในลักษณะ Shadow ว่าจะมีการเขียนค่าข้อมูล Shadow นี้เมื่อตรงเงื่อนไขของเหตุการณ์ใดบ้าง ซึ่งประกอบไปด้วย 3 เหตุการณ์ ได้แก่

- ***After a number of message parts*** คือการกำหนดว่าให้เขียน Shadow เมื่อมีได้รับข้อความมาทั้งหมดกี่ค่าแล้ว
- ***After timeout following the first message*** คือการกำหนดระยะเวลาสิ้นสุด (timeout) หลังจากที่ได้รับข้อความแรกเข้ามา
- ***After a message with the msg.complete property set*** จะไม่ได้ต้องการให้ผู้ใช้งานกำหนดค่าใด ๆ แต่เป็นการย้ำเตือนว่าจะทำการเขียนข้อมูล Shadow ลงไปในอุปกรณ์เมื่อค่าข้อความที่มีคุณสมบัติ `msg.complete` ได้ถูกตั้งค่าเอาไว้เรียบร้อยแล้ว

ตัวอย่าง ผู้ใช้งานทำการสร้างโหนดที่จะส่งค่าอุณหภูมิและความชื้นขึ้นมาด้วยโหนด 'Inject' ซึ่งในที่นี้จะทำการแบ่งค่าข้อมูลทั้งสองตัวแปรนั้นออกเป็นสองโหนดที่แตกต่างกัน โดยให้กำหนดค่า payload เป็นค่าข้อมูล และใส่กำกับชื่อตัวแปรของค่าข้อมูลนั้นเอาไว้ที่ topic

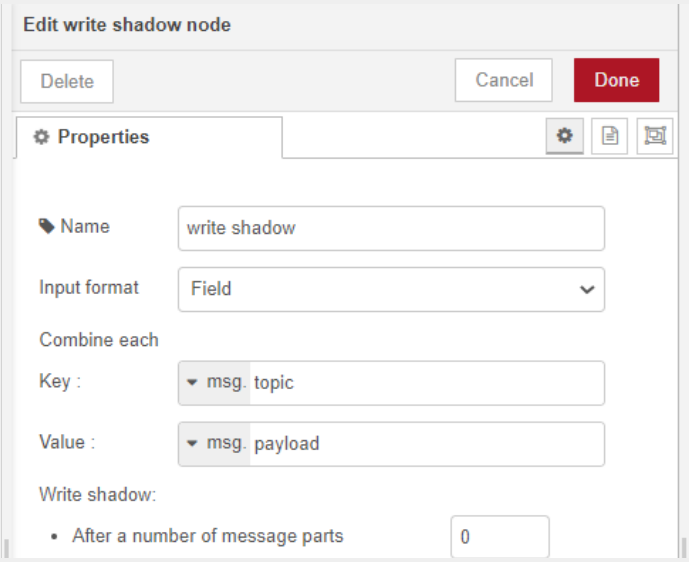


ผู้ใช้งานทำการนำโหนด **'Inject'** ลากเชื่อมเข้ากับโหนด **'Write Shadow'** เพื่อรอรับค่าข้อมูลที่ออกมาและแปลงค่าเหล่านั้นที่เข้ามาให้อยู่ในลักษณะของ Shadow เพื่อเตรียมส่งค่าข้อมูลสุดท้ายเข้าไปยังอุปกรณ์ โดยให้ทำการลากเส้นจากโหนด **'Write Shadow'** เข้าไปในโหนด **'Device'** อีกที ในกรณีที่ต้องการตรวจสอบผลของค่าข้อมูลนั้นว่าเป็นไปตามที่คาดหวังเช่นไร ผู้ใช้งานสามารถนำโหนด **'Debug'** ไปเชื่อมต่อท้ายของโฟลว์นี้อีกทีได้



ภายในโหนด **'Write Shadow'** นั้น เนื่องจากค่าข้อมูลที่เข้ามาภายในโหนดนี้ยังไม่ใช่ค่า Shadow แต่เป็นค่าข้อมูลที่กำกับตัวแปรเอาไว้และส่งมาจากโหนดที่แตกต่างกัน ผู้ใช้งานจึงจำเป็นต้องเลือก Input format ภายในโหนดนี้ว่าเป็น 'Field' ซึ่งมีค่า key คือ topic ซึ่งเป็นชื่อของตัวแปรที่กำหนดเอาไว้ก่อนหน้าและ value คือ payload ซึ่งเป็นค่าข้อมูล

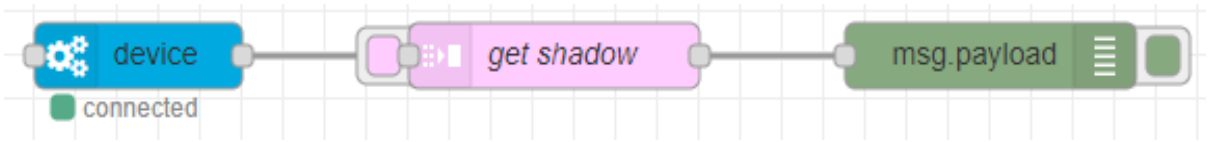
ของตัวแปรนี้นั่นเอง จากนั้นจึงทำการคลิกที่ปุ่ม ‘Done’ เพื่อบันทึกการเปลี่ยนแปลงนั้น




```
[{"id":"58c7f8c.d03f608","type":"write shadow","z":"8cfb1028.a0d6d","name":"write
shadow","shadowproperty":"payload","shadowpropertyType":"msg","shadowformatmode":"auto","fieldkeyprop
erty":"topic","fieldkeypropertyType":"msg","fieldvalueproperty":"payload","fieldvaluepropertyType":"m
sg","timeout":0.2,"count":0,"inputformat":"json","x":370,"y":3340,"wires":[["adee5717.14da38"]]},{"id
":"adee5717.14da38","type":"device","z":"8cfb1028.a0d6d","name":"Device1","deviceid":"<device_id>","d
evicetoken":"<device_token>","mqttchannel":"$piebox.mqttchannel.NEXPIE","topics":"@msg/room1","active
":true,"subshadow":true,"subprivate":true,"initshadow":true,"x":540,"y":3340,"wires":[["ac501608.b884
f8"]]},{"id":"ac501608.b884f8","type":"debug","z":"8cfb1028.a0d6d","name":"","active":true,"tosidebar
":true,"console":false,"tostatus":false,"complete":"false","x":710,"y":3340,"wires":[]},{"id":"bc5b14
3c.9f9428","type":"inject","z":"8cfb1028.a0d6d","name":"Field Data
Temperature","topic":"temperature","payload":"23","payloadType":"num","repeat":"","crontab":"","once"
:false,"onceDelay":0.1,"x":140,"y":3320,"wires":[["58c7f8c.d03f608"]]},{"id":"$piebox.mqttchannel.NEX
PIE","type":"mqttchannel","name":"NEXPIE","endpoint":"mqtt://mqtt.nexpie.io:1883"}]
```

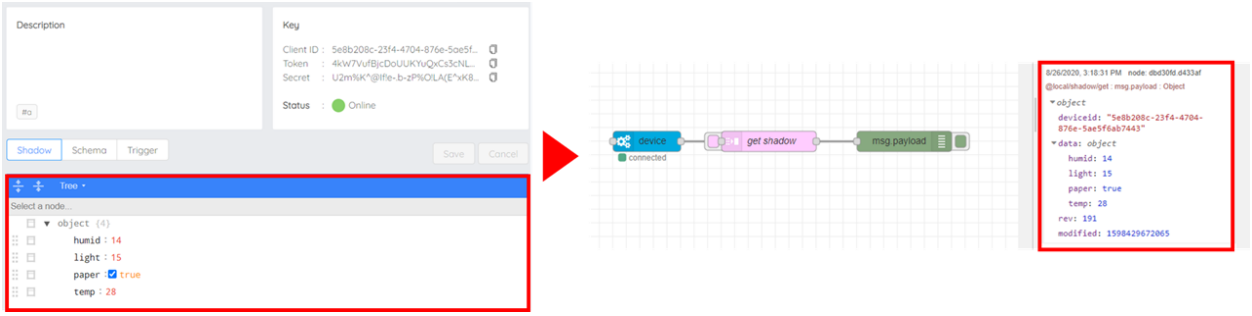

เรียกดูค่าข้อมูลจากแพลตฟอร์ม

ผู้ใช้งานสามารถดูค่าข้อมูลปัจจุบันของอุปกรณ์ (Shadow) โดยการใช้โหนด **‘Get Shadow’** ที่อยู่ในหมวดหมู่ ***‘Command’***



ผู้ใช้งานไม่จำเป็นต้องทำการตั้งค่าใด ๆ ภายในโหนดนี้ แต่ให้ทำการนำโหนดนี้เชื่อมต่อเข้าด้านหลังของโหนดอุปกรณ์ที่ผู้ใช้งานต้องการจะรับค่าข้อมูลดังรูปด้านบน เมื่อค่า Shadow ของอุปกรณ์ที่โหนดนี้ได้ไปทำการเชื่อมต่อมีการเปลี่ยนแปลงข้อมูลที่อยู่ด้านใน โหนด **‘Get Shadow’** จะทำการดึงค่าข้อมูล Shadow ของอุปกรณ์นั้นออกมาใช้งานในทันที

ในทางกลับกัน หากผู้ใช้งานทำการย้ายโหนดนี้ไปไว้หน้าโหนดของอุปกรณ์ดังรูปด้านบน ผู้ใช้งานสามารถผลักค่า Shadow ของอุปกรณ์ให้ออกมาแสดงผลได้โดยการคลิกบนปุ่มที่อยู่บนโหนดนี้ เปรียบเสมือนการกระทุ้งค่าข้อมูลให้ออกมาจากโหนดของอุปกรณ์นั้นนั่นเอง โดยการทำเช่นนี้จะทำให้ผู้ใช้งานสามารถดูค่าข้อมูล Shadow ปัจจุบันของอุปกรณ์ได้โดยไม่จำเป็นต้องรอให้มีการเปลี่ยนแปลงของค่า Shadow ของอุปกรณ์นั้น



เรียกดูสถานะของอุปกรณ์

ผู้ใช้งานสามารถดูค่าสถานะปัจจุบันของอุปกรณ์ได้ว่ายังคงทำงานอยู่ (On) หรือยุติการทำงานไปเรียบร้อยแล้ว (Off) โดยการใช้โหนด **‘Get Status’** ที่อยู่ใหมวดหมู่ ***‘Command’***

ข้อแตกต่างระหว่างโหนด **‘Get Status’** และโหนด **‘Get shadow’** นั้นคือ โหนดนี้จำเป็นต้องทำการเชื่อมต่อไว้ทางด้านหน้าของอุปกรณ์ที่ต้องการตรวจสอบสถานะการเชื่อมต่อเท่านั้น โดยวิธีการใช้งานของโหนดนี้นั้นจะเปรียบเสมือนการกระทุ้งค่าสถานะปัจจุบันให้ออกมาจากอุปกรณ์ที่ต้องการ ซึ่งผู้ใช้งานไม่จำเป็นต้องตั้งค่าให้กับโหนดนี้เช่นกันกับโหนด ‘Get Shadow’ แต่ให้นำโหนดนี้เชื่อมต่อเข้ากับจุดเชื่อมต่อขาเข้ากับอุปกรณ์ที่ต้องการนำค่าข้อมูลออกมา

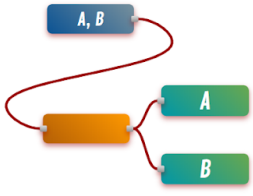
ในทุกครั้งที่ผู้ใช้งานทำการกดที่ปุ่มของโหนดนี้ จะเป็นการกระทุ้งค่าข้อมูลให้ออกมาจากโหนดของอุปกรณ์ที่โหนดนี้ไปเชื่อมต่ออยู่ โดยในที่นี้คือการกระทุ้งค่าข้อมูลสถานะปัจจุบันของอุปกรณ์ที่ถูกเชื่อมต่ออยู่ให้ออกมาแสดงผล โดยผลลัพธ์ที่ออกมานั้นจะอยู่ในลักษณะของค่า 0 และ

1 ซึ่งค่า 0 นั้นหมายถึงอุปกรณ์อยู่ในสถานะไม่พร้อมใช้งาน ในขณะที่ 1 หมายถึงอุปกรณ์ได้รับการเชื่อมต่อและอยู่ระหว่างการใช้งานนั่นเอง โดยรูปแบบของผลลัพธ์ที่ออกมาจะปรากฏว่า

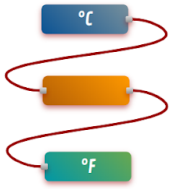
```
{“status”: 1}
```

การจัดการข้อมูล

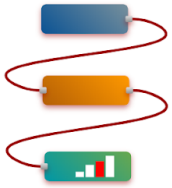
ในบางกรณี ข้อมูลที่รับหรือส่งมาจากอุปกรณ์อาจจำเป็นต้องมีการตั้งค่าและ/หรือแก้ไขบางอย่างเพื่อความเหมาะสมของการใช้งาน โดย NEXPIE นั้นได้ทำการสร้างโหนดขึ้นมา 4 ชนิด ในหมวดหมู่ ***‘Flow’*** เพื่อใช้ควบคู่กับการตั้งค่าข้อมูลต่าง ๆ ที่ได้รับ ได้แก่ การคัดแยกค่าข้อมูล (Extract) การเปลี่ยนแปลงค่าข้อมูล (Transform) การแปลงข้อมูลที่ได้รับเมื่อถึงระดับที่กำหนด (Level Trigger) และการสลับผล (Toggle)



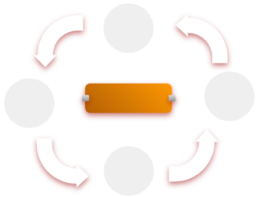
การคัดแยกค่าข้อมูล (Extract)



การเปลี่ยนแปลงค่าข้อมูล (Transform)



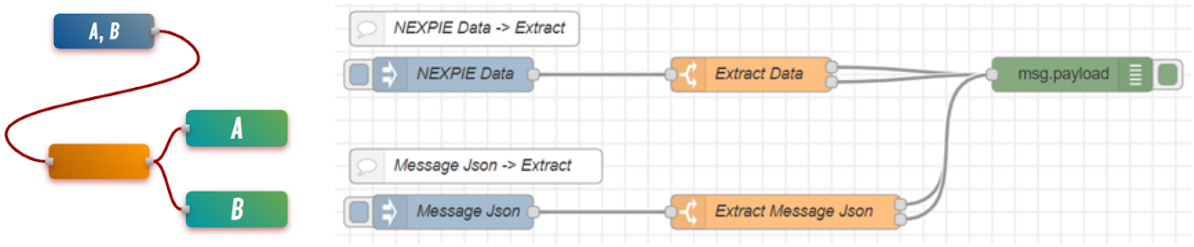
การส่งค่าข้อมูลใหม่เมื่อมีการเปลี่ยนแปลงของระดับการทำงาน (Level trigger)



การสลับค่าสถานะที่ต้องการส่งออก (Toggle)

การคัดแยกค่าข้อมูล

ในกรณีที่ค่าข้อมูลถูกส่งเข้ามาในลักษณะของ JSON หรือค่าข้อมูลในลักษณะ Shadow ผู้ใช้งานสามารถทำการคัดแยกข้อมูลเหล่านั้นออกจากกันได้ด้วยการใช้โหนดชื่อว่า **‘Extract’** ในการดึงค่าข้อมูลจากแต่ละ field ออกมา โดยจะส่งออกเป็นค่าข้อมูลหลายค่า (One-to-many) ออกมาจากโหนดนี้ ตามจำนวนที่ผู้ใช้งานกำหนดเอาไว้



ข้อมูลที่สามารถส่งเข้ามาภายในโหนดนี้ได้นั้นประกอบด้วยกัน 2 รูปแบบ

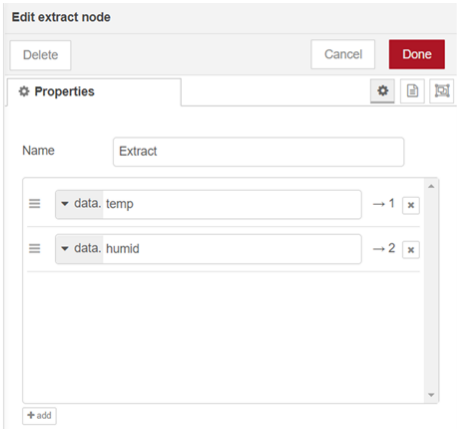
- **NEXPIE Data** หรือค่าข้อมูลในลักษณะ Shadow ซึ่งค่าข้อมูล payload ที่ถูกส่งเข้ามาภายในโหนดนี้นั้นต้องอยู่ในลักษณะ

  `{“data”: {“field1”: “value1”, “field2”: “value2”, …}}`

  โดย field หมายถึงค่าตัวแปรต่าง ๆ และ value คือค่าข้อมูลของตัวแปรนั้น ๆ การใช้งานประเภทนี้เหมาะกับข้อมูล Shadow ที่ได้รับมาจากอุปกรณ์โดยตรง
- **JSON Data** หรือค่าข้อมูลที่ถูกส่งมาในลักษณะของ JSON โดยตรงโดยต้องไม่จำเป็นต้องนำ ‘data’ มาคลุมค่าข้อมูลทั้งหมดเอาไว้อีกทีเหมือน Shadow

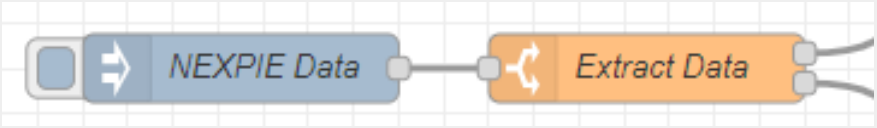
  `{“field1”: “value1”, “field2”: “value2”, …}`

ซึ่งในการรับค่าข้อมูลของโหนดนี้ ให้ผู้ใช้งานทำการเปลี่ยน drop-down ไปที่ ‘*data.*’ ในกรณีที่ค่าข้อมูลเป็น NEXPIE Data หรือ ‘*json path*’ ในกรณีที่ค่าข้อมูลเป็น JSON Data จากนั้นจึงเติมชื่อตัวแปรที่ได้รับมาจากค่าข้อมูลก่อนหน้าเพื่อดึงค่าข้อมูลนั้นออกมา

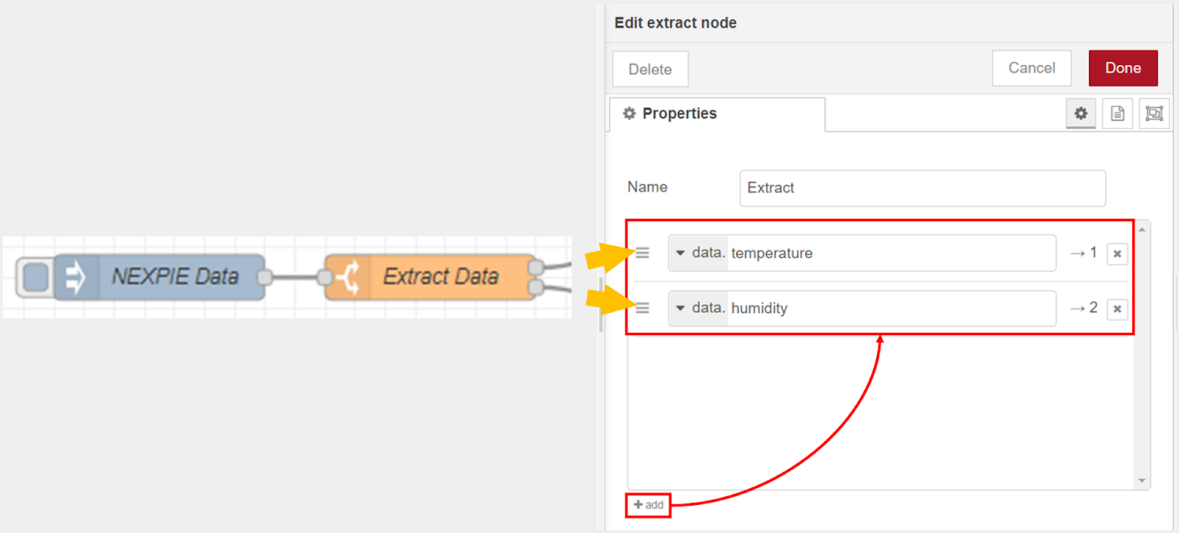


จำนวนจุดสีเทาที่อยู่ด้านขวาจะถึงอยู่กับจำนวนค่าตัวแปรที่ผู้ใช้งานกำหนดและจะลำดับจากบนลงล่างตามลำดับของตัวแปรที่ตั้งค่าไว้ในโหนด ซึ่งผู้ใช้งานสามารถเพิ่มค่าตัวแปรอื่น ๆ ได้ด้วยการกดที่ปุ่ม ‘+ add’ ด้านล่าง และสามารถลบตัวแปรที่ไม่ต้องการดึงออกมาใช้งานได้ด้วยกดที่ปุ่มกากบาทบนแถบข้อมูลของตัวแปรนั้น

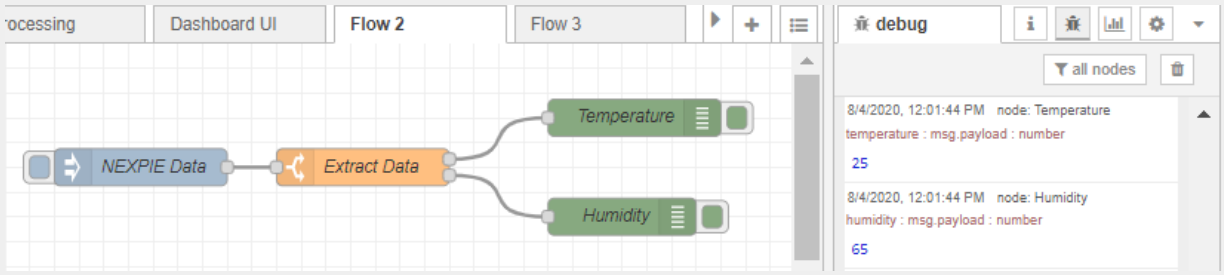
ตัวอย่าง ข้อมูลจากโหนดก่อนหน้าถูกส่งเข้ามาในลักษณะของค่า Shadow ที่ประกอบไปด้วยตัวแปร *'temperature'* และ *'humidity'* ผู้ใช้งานต้องการแยกค่านี้ออกเป็นสองเส้นเพื่อนำไปส่งให้แก่อุปกรณ์ที่แตกต่างกัน โดยสามารถทำได้จากการลากข้อมูลจากโหนดที่ส่งค่า Shadow ออกมาเข้ากับโหนด **'Extract'** และเปิดหน้าการตั้งค่าของโหนดขึ้นมา



ในหน้าการตั้งค่า ผู้ใช้งานทำการคลิกที่ปุ่ม '+ add' เพื่อเพิ่มตัวแปรที่ต้องการคัดแยกออกจาก Shadow ขึ้นมาอีกค่าหนึ่ง จากนั้นทำการกรอกชื่อตัวแปรของค่าข้อมูลนั้นลงไป โดยตัวเลขทางด้านขวาที่กำกับชื่อตัวแปรที่ใส่ลงไป จะบอกถึงตำแหน่งของจุดเชื่อมต่อขาออกของโหนดนี้จากบนลงล่าง เริ่มต้นที่เลข 1 หลังจากนั้นให้ทำการคลิกที่ปุ่ม 'Done' เพื่อยืนยันการเปลี่ยนแปลง



เมื่อทำการบันทึกการเปลี่ยนแปลงของหน้าโฟลว์เรียบร้อยแล้ว หากลองนำโหนด **'Debug'** มาเชื่อมต่อกับแต่ละจุดเชื่อมต่อขาออกของโหนดนี้ จะพบว่าค่าข้อมูลถูกส่งออกแยกเป็นสองค่าโดยไม่เหลือเค้าโครงของ Shadow อีกต่อไป




```
[{"id":"f4fb3b18.7352b8","type":"inject","z":"8cfb1028.a0d6d","name":"NEXPIE
Data","topic":"","payload":"{\"data\":{\"temperature\":25,\"humidity\":65}}","payloadType":"json","re
peat":"","crontab":"","once":false,"onceDelay":0.1,"x":630,"y":140,"wires":[["5506dd32.71d9a4"]]},{"i
d":"5506dd32.71d9a4","type":"extract","z":"8cfb1028.a0d6d","name":"Extract
Data","property":"payload","propertyType":"msg","rules":[{"v":"temperature","vt":"data.","t":"eq"},{"
v":"humidity","vt":"data.","t":"eq"}],"outputs":2,"x":810,"y":140,"wires":[["82c0b54.9b4f148"],["d88d
090b.7756e8"]]},{"id":"82c0b54.9b4f148","type":"debug","z":"8cfb1028.a0d6d","name":"Temperature","act
ive":true,"tosidebar":true,"console":false,"tostatus":false,"complete":"payload","targetType":"msg","
x":1030,"y":100,"wires":[]},{"id":"d88d090b.7756e8","type":"debug","z":"8cfb1028.a0d6d","name":"Humid
```

```
ity","active":true,"tosidebar":true,"console":false,"tostatus":false,"complete":"payload","targetType
":"msg","x":1020,"y":180,"wires":[]}]
```

ในการคัดแยกค่าข้อมูลจากค่า JSON ที่ไม่ใช่ Shadow ผู้ใช้งานสามารถทำได้โดยการเปลี่ยนจาก '*data.*' ให้กลายเป็น '*json path*' แล้วกรอกชื่อของตัวแปรลงไปตามเดิม

การเปลี่ยนแปลงค่าข้อมูล

โหนด **'Transform'** จะเปลี่ยนแปลงค่าข้อมูลที่ได้รับเข้ามา ผ่านกระบวนการทางคณิตศาสตร์หรือสมการอื่น ๆ ที่สามารถเขียนขึ้นได้ในรูปแบบของ JSONata ซึ่งผู้ใช้งานสามารถกำหนดขึ้นมาเองได้ มักใช้ในกรณีที่ต้องการเปลี่ยนแปลงหน่วยของค่าข้อมูลต่าง ๆ หรือการเปลี่ยนแปลงลักษณะการแสดงผลของข้อมูลต่าง ๆ ให้เป็นอีกลักษณะเพื่อนำไปใช้งานในส่วนอื่น ๆ ต่อได้ เช่น การเปลี่ยนแปลงจากค่าอุณหภูมิจากแบบองศาเซลเซียสเป็นอุณหภูมิแบบองศาฟาเรนไฮต์ หรือเปลี่ยนแปลงจากหน่วยเวลานาทีให้กลายเป็นหน่วยเวลาแบบวินาที เป็นต้น

การตั้งค่าของโหนดนี้นั้นประกอบไปด้วย 5 ส่วน ดังนี้

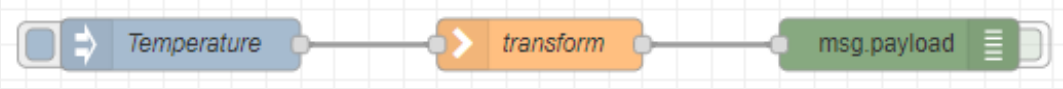
- **Name** คือการตั้งชื่อให้กับโหนดนี้
- **Input** คือการเลือกประเภทของข้อมูลที่โหนดนี้ได้รับเข้ามา ซึ่งสามารถเลือกได้ 3 ประเภท ได้แก่
  - ***Raw input*** คือค่าข้อมูลที่เข้ามาในลักษณะข้อมูลดิบหรือค่าตัวเลขเพียงค่าเดียวโดยตรง
  - ***NEXPIE Data*** คือค่าข้อมูลที่เข้ามาในรูปแบบของ shadow ซึ่งผู้ใช้งานสามารถเข้าถึงข้อมูลของแต่ละตัวแปรด้วยการเลือก drop-down ไปที่ '*data.*' แล้วทำการพิมพ์ชื่อตัวแปรที่ต้องการดึงค่ามาใช้งานลงไป
  - ***JSON Path*** คือค่าข้อมูลที่เข้ามาในรูปแบบ JSON โดยการเลือกที่ตัวเลือก drop-down '*json path*' โดยผู้ใช้งานสามารถพิมพ์ชื่อของค่าตัวแปรที่ต้องการดึงออกมาจากค่าข้อความที่ถูกส่งเข้ามาไปใช้งานต่อได้โดยตรง
- **Expression** คือการตั้งค่าสมการทางคณิตศาสตร์ โดยผู้ใช้งานสามารถใช้ '`$value`' เพื่อเป็นตัวแปรแทนค่าข้อมูลที่เข้ามาภายในโหนดได้ เช่น หากผู้ใช้งานต้องตั้งสมการเพื่อ

เปลี่ยนค่าข้อมูลจากองศาเซลเซียส (℃) ให้กลายเป็งองศาฟาเรนไฮต์ (℉) ผู้ใช้งานสามารถเขียนค่าสมการลงไปได้ว่า

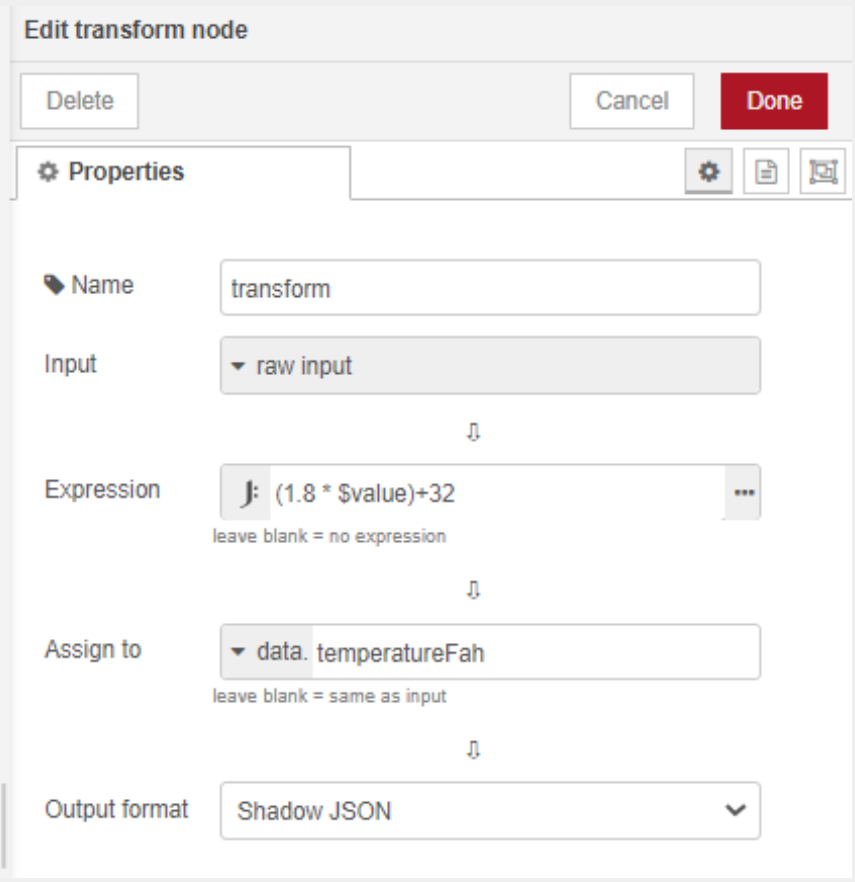
```
(1.8 * $value) + 32
```

- **Assign to** คือการตั้งค่าว่าผลลัพธ์ที่ออกไปนั้นจะออกไปยังตัวแปรใด โดยสามารถเลือกได้ว่าจะให้ข้อมูลออกไปในลักษณะของข้อมูลดิบ (Raw data) หรือในลักษณะของ shadow ก็ได้เช่นกัน
- **Output format** คือการเลือกลักษณะของผลลัพธ์
  - ***Shadow JSON*** คือการส่งข้อมูลออกไปในลักษณะของ shadow
  - ***Field*** จะเป็นการส่งค่าข้อมูลในลักษณะของ payload ออกไป โดยภายในโหนดปลายทางจะได้รับข้อมูลในลักษณะของ ‘msg’

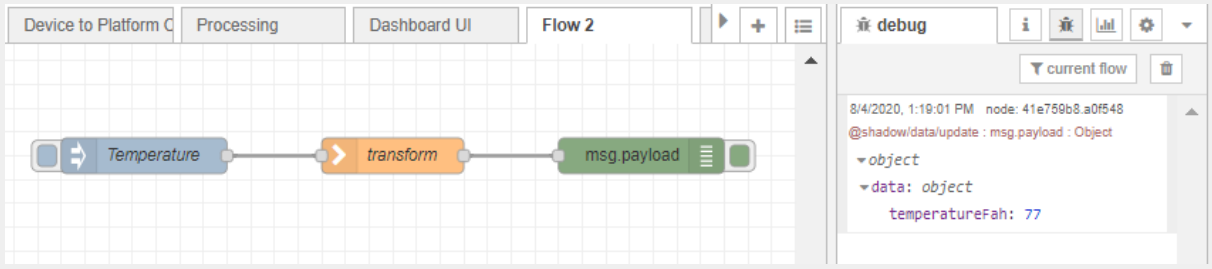
ตัวอย่าง ค่าข้อมูลที่ถูกส่งเข้ามาในรูปแบบองศาเซลเซียส ผู้ใช้งานต้องการเปลี่ยนค่าข้อมูลนั้นให้อยู่ในรูปแบบขององศาฟาเรนไฮต์ โดยทำการนำโหนด **‘Transform’** มาลากต่อจากโหนดที่ทำการส่งข้อมูลอุณหภูมินั้นมา พร้อมด้วยโหนด ‘Debug’ ที่จะทำการตรวจสอบค่าข้อมูลที่ออกมาจากโหนด ‘Transform’ ว่าเป็นไปดังที่คาดการณ์หรือไม่ จากนั้นจึงทำการเปิดหน้าแก้ไขของโหนด **‘Transform’** ขึ้นมา



ข้อมูลที่ถูกส่งเข้ามาในโหนดนี้เป็นค่าข้อมูลดิบ ดังนั้น Input ของโหนดนี้จึงต้องเลือกเป็น ‘raw input’ จากนั้นทำการตั้งค่าสมการของการเปลี่ยนแปลงค่าข้อมูลนี้ว่า `(1.8 * $value) + 32` ซึ่งอ้างอิงมาจากค่าสมการ $F = \frac{C \times 9}{5} + 32$ โดยให้ `$value` แทนค่าข้อมูลในหน่วยองศาเซลเซียสที่ถูกส่งมาจากโหนดก่อนหน้า จากนั้นจึงทำการตั้งค่าให้เก็บโหนดนี้ในลักษณะของ Shadow ในตัวแปรที่ชื่อ `temperatureFah` โดยให้กำหนดลักษณะของค่าข้อมูลขาออกใน Output format ว่าเป็น ‘Shadow JSON’ จากนั้นตั้งชื่อตัวแปรของค่าข้อมูลขาออกให้กำกับโดยมี ‘*data.*’ อยู่ด้านหน้าชื่อของตัวแปรนั้น ทำการคลิกที่ปุ่ม ‘Done’ เมื่อทำทุกอย่างเรียบร้อยแล้ว

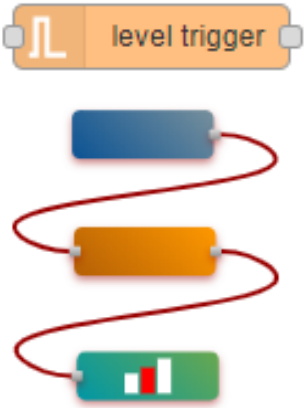


ทำการบันทึกหน้าโฟลว์หลังการเปลี่ยนแปลงสำเร็จ เมื่อลองทำการส่งค่าข้อมูล 25 องศาเซลเซียสจากโหนดที่สร้างข้อมูลเข้าไปยังโหนดนี้ จะพบว่าข้อมูลที่ออกมานั้นมีค่าเท่ากับ 77 องศาฟาเรนไฮต์ ซึ่งค่าข้อมูลที่ออกมานั้นจะอยู่ในรูปลักษณะของต่า Shadow ที่มีตัวแปรชื่อ `temperatureFah` กำกับค่านี้เอาไว้



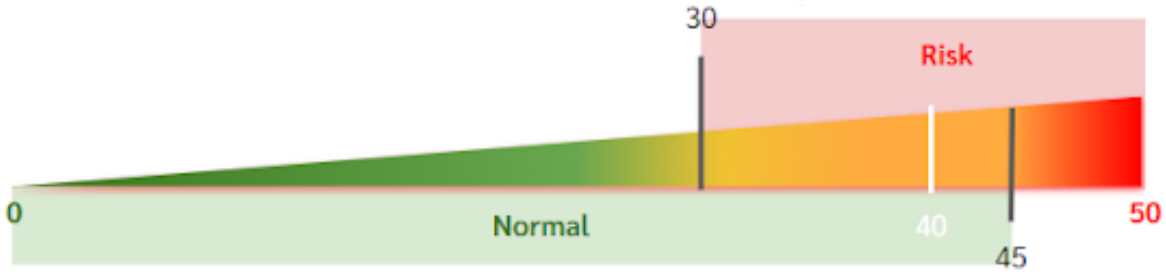

```
[{"id":"88719d44.72ca5","type":"inject","z":"8cfb1028.a0d6d","name":"Temperature","topic":"temperatur
e","payload":"25","payloadType":"json","repeat":"","crontab":"","once":false,"onceDelay":0.1,"x":450,
"y":2140,"wires":[["1267e7af.3407b8"]]},{"id":"1267e7af.3407b8","type":"transform","z":"8cfb1028.a0d6
d","name":"transform","shadowproperty":"payload","shadowpropertyType":"msg","inputfieldproperty":"","
inputfieldpropertyType":"raw","expression":"(1.8*$value)+32","outputfieldproperty":"temperatureFah","
outputfieldpropertyType":"data.","outputformat":"json","x":660,"y":2140,"wires":[["41e759b8.a0f548"]]
},{"id":"41e759b8.a0f548","type":"debug","z":"8cfb1028.a0d6d","name":"","active":true,"tosidebar":tru
e,"console":false,"tostatus":false,"complete":"false","x":870,"y":2140,"wires":[]}]
```

การแปลงข้อมูลที่ได้รับเมื่อถึงระดับที่กำหนด



หลายครั้งที่ผู้ใช้งานไม่ได้ต้องการจะรู้ค่าข้อมูลที่ได้รับจากอุปกรณ์นั้นโดยตรง แต่ต้องการจะรู้เพียงช่วงของค่าข้อมูลที่ส่งเข้ามาเสียมากกว่าว่ากำลังอยู่ในระดับใด เช่น ค่าข้อมูลที่ได้รับมานั้นสูงเกินไป หรืออยู่ต่ำกว่าค่ามาตรฐานที่ได้ทำการกำหนดเอาไว้

โหนด **‘Level Trigger’** ถูกสร้างขึ้นมาเพื่อตอบสนองความต้องการนี้ของผู้ใช้งาน โดยการแบ่งค่าข้อมูลออกเป็นสองระดับที่มีค่าขอบบนและขอบล่างของค่ามาตรฐานเป็นเกณฑ์ในการแบ่งระดับของค่าข้อมูลเหล่านั้น ซึ่งค่าขอบบนและขอบล่างของค่ามาตรฐานนี้คือการกำหนดขอบเขตการเบี่ยงเบนของค่าข้อมูลที่อาจจะเกิดขึ้น เช่น ต้องการบอกว่า 25 องศาเซลเซียสของเครื่องปรับอากาศคือค่าของอุณหภูมิกลางที่ใช้แบ่งระหว่างความเย็นและความร้อน ค่าขอบเขตนี้จึงเป็นการกำหนดความคลาดเคลื่อนที่อาจเป็นไปได้ของอุปกรณ์นั้นนั่นเอง



เมื่อมีข้อมูลที่ไหลผ่านเข้ามาภายในโหนด โหนดนี้จะทำการตัดสินใจว่าค่าข้อมูลนี้ควรอยู่ในระดับใดที่กำหนดไว้ และจะส่งออกค่าข้อมูลที่ถูกตั้งเอาไว้ในระดับนั้นออกไป เช่น การส่งข้อความออกไปจากโหนดว่า ‘High’ ในกรณีที่ค่าข้อมูลที่เข้ามาในโหนดมีค่าสูงกว่าค่ามาตรฐานที่ตั้งเอาไว้ก่อนหน้า เป็นต้น โดยค่าที่ถูกส่งออกไปจากโหนดนี้จะยังคงเป็นผลลัพธ์เดิมไว้เสมอ จนกว่าจะมีค่าข้อมูลใดที่ลดลงจนต่ำกว่าขอบล่างของข้อมูล หรือเพิ่มสูงขึ้นจนสูงกว่าขอบบนของข้อมูล

ตัวอย่าง หากผู้ใช้งานกำหนดให้ 25 องศาเซลเซียสคือค่ามาตรฐานโดยมีค่าความคลาดเคลื่อนของอุปกรณ์อยู่ที่ ±5 องศาเซลเซียส นั่นหมายถึง ผู้ใช้งานต้องกำหนดค่าขอบบนไว้ที่ 30 องศาเซลเซียสเพื่อแสดงผล **‘High’** และกำหนดค่าขอบล่างไว้ที่ 20 องศาเซลเซียสเพื่อแสดงผล **‘Low’** หลังจากนั้นจึงทำการส่งค่าของข้อมูลเข้ามายังโหนดนี้เรื่อย ๆ โดยให้มีการเปลี่ยนแปลงค่าของข้อมูลไปในแต่ละครั้งที่มีการส่งข้อมูลเข้ามา ซึ่งผลลัพธ์จะปรากฏดังตารางด้านล่าง

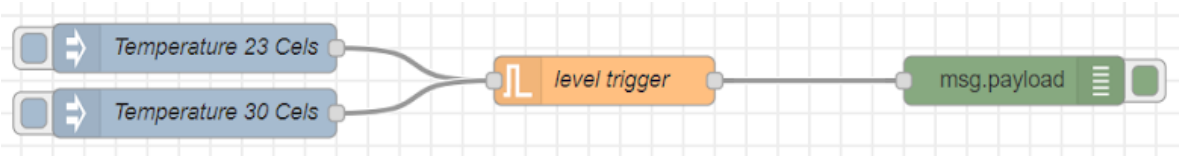
ครั้งที่	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
อุณหภูมิ (°C)	32	30	28	25	23	21	27	31	33

ผลลัพธ์	High	High	High	High	Low	Low	Low	High	High

การส่งค่าข้อมูลใหม่เมื่อมีการเปลี่ยนแปลงของระดับการทำงานนั้นจะเป็นการตั้งค่าขอบล่างและขอบบนของการเปลี่ยนแปลง โดยค่าข้อมูลจะทำการเปลี่ยนแปลงไปใช้งานค่าข้อมูลในระดับบน (High Value) เมื่อค่าข้อมูลเพิ่มขึ้นจนเกินขอบบนไป (High Trigger) และใช้งานในระดับล่าง (Low Valu) เมื่อค่าข้อมูลลดลงต่ำกว่าค่าระดับล่างที่ตั้งไว้ (Low Trigger)

การเปลี่ยนแปลงของข้อมูลในลักษณะนี้ จะช่วยให้ค่าข้อมูลที่มีความกวัดแกว่งนั้นสามารถเปลี่ยนระดับของข้อมูลได้ดีขึ้นกว่าการตั้งค่ากลางเพียงค่าเดียว เพราะมีช่วงเผื่อของค่าข้อมูลเอาไว้ในกรณีที่ค่าข้อมูลยังลงไม่ถึงขอบล่างหรือขึ้นไม่ถึงขอบบน

ตัวอย่างการใช้งาน เช่น การตั้งค่าให้เครื่องปรับอากาศปิดการทำงานเมื่ออุณหภูมิลดลงต่ำกว่า 25 องศาเซลเซียส หรือการตั้งค่าให้ไฟปิดเมื่อพระอาทิตย์ขึ้น และเปิดอีกครั้งเมื่อไม่มีแสงจากภายนอกหรือพระอาทิตย์ เป็นต้น

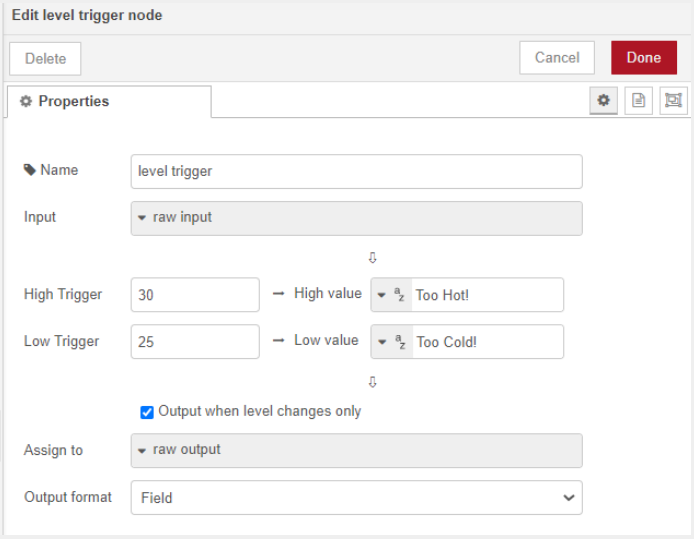


ภายในโหนดนี้ มีการตั้งค่าประกอบไปด้วย 6 ส่วน ดังนี้

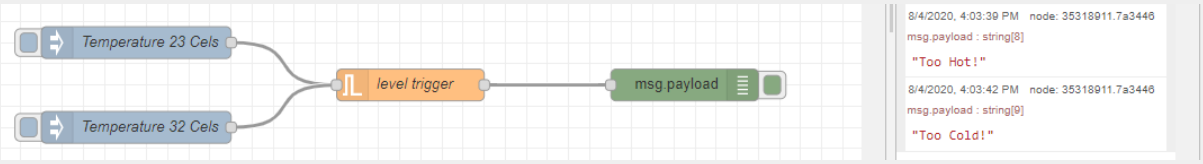
- **Name** คือการตั้งชื่อให้กับโหนดนี้
- **Input** คือการกำหนดลักษณะของค่าข้อมูลที่เข้ามาว่าต้องการรับค่าข้อมูลในลักษณะใด ระหว่างค่าข้อมูล shadow ด้วยการเลือก ‘*data.*’ หรือค่าข้อมูลที่มาจากไฟล์ JSON ด้วยการเลือก ‘*json path*’ หรือค่าข้อมูลดิบของตัวเลข (Raw data)
- **High/Low Trigger** คือการตั้งค่าขอบบน (High Trigger) และขอบล่าง (Low Trigger) ของค่าข้อมูลที่จะทำให้มีการเปลี่ยนแปลงของผลลัพธ์ เช่น การตั้งค่าขอบบนไว้ที่ 80 หมายถึง หากค่าข้อมูลที่ได้รับเข้ามาเกิน 80 ค่าที่ตั้งไว้ในขอบบนนี้จะถูกนำไปใช้งาน
- **High/Low Value** คือการตั้งค่าผลลัพธ์เมื่อค่าข้อมูลที่ได้รับเข้ามาสูงกว่าค่าขอบบนหรือลดต่ำกว่าค่าขอบล่างที่ตั้งเอาไว้ตามลำดับ เช่น เมื่อค่าข้อมูลที่ได้รับเข้ามาสูงเกินค่าขอบบนที่ตั้งเอาไว้ จะส่งค่าข้อมูลออกมาว่า ‘High’ เพื่อบอกว่าค่าข้อมูลยั้นสูงเกินไป เป็นต้น
- **Assign to** คือการเลือกว่าผลลัพธ์ที่ได้คำนวณออกมานั้น จะถูกส่งออกไปยังค่าตัวแปรใดระหว่างค่า shadow หรือออกไปในลักษณะ JSON หรือออกไปเป็นค่าข้อมูลดิบ (Raw data)

- **Output format** คือการเลือกลักษณะของผลลัพธ์ที่ออกไป
  - ***Shadow JSON*** คือการส่งข้อมูลออกไปในลักษณะของ shadow
  - ***Field*** จะเป็นการส่งค่าข้อมูล payload ออกไป

ตัวอย่าง เครื่องปรับอากาศถูกกำหนดให้มีการกำหนดระดับความเย็นไว้เมื่ออุณหภูมิลดลงถึง 25 องศาเซลเซียสว่าเป็นความเย็นจนเกินไป และอุณหภูมิที่สูงกว่า 30 องศาเซลเซียสว่าเป็นความร้อนจนมากเกินไป ซึ่งผู้ใช้งานสามารถตั้งค่าเงื่อนไขนี้ได้ โดยการกำหนด High Trigger เท่ากับ 30 และกำหนด Low Trigger ให้เท่ากับ 25 โดยหากค่าข้อมูลที่เข้ามานั้นสูงกว่าค่า High Trigger ที่ตั้งว่า ให้ตั้งค่าข้อความผลลัพธ์ใน High Value ออกมาว่า ‘Too Hot!’ ในทางกลับกันที่ข้อมูลต่ำกว่าค่า Low Trigger ที่ตั้งไว้ ให้ตั้งค่าข้อความใน Low Value ว่า ‘Too Cold!’



เมื่อทำการบันทึกหน้าโฟลว์นี้หลังจากเชื่อมต่อโหนดข้อมูลขาเข้าและโหนด **‘Debug’** ประกบกับโหนดนี้แล้ว เมื่อทำการส่งข้อมูลเข้ามาภายในโหนดนี้จะพบว่า หากข้อมูลที่ส่งเข้ามาภายในโหนดนี้อยู่สูงกว่ากว่า High Trigger ผลลัพธ์ที่แสดงออกมาจะเป็น ‘Too Hot!’ และจะแสดงผลว่า ‘Too Cold!’ ในกรณีที่ค่าข้อมูลที่ส่งมาอยู่ต่ำกว่า Low Trigger




```
[{"id":"85b44dfd.a9351","type":"level
trigger","z":"6ad0365d.c70448","name":"","shadowproperty":"payload","shadowpropertyType":"msg","hight
rigger":"30","lowtrigger":"25","outputformat":"field","inputfieldproperty":"","inputfieldpropertyType
":"raw","outputfieldproperty":"","outputfieldpropertyType":"raw","lowoutputproperty":"Too
Cold!","lowoutputpropertyType":"str","highoutputproperty":"Too
Hot!","highoutputpropertyType":"str","triggeronchange":true,"x":430,"y":1160,"wires":[["35318911.7a34
46"]]},{"id":"35318911.7a3446","type":"debug","z":"6ad0365d.c70448","name":"","active":true,"tosideba
r":true,"console":false,"tostatus":false,"complete":"false","x":690,"y":1160,"wires":[]},{"id":"1f838
35e.ecbc6d","type":"inject","z":"6ad0365d.c70448","name":"Temperature 23
Cels","topic":"temperature","payload":"23","payloadType":"num","repeat":"","crontab":"","once":false,
"onceDelay":0.1,"x":170,"y":1120,"wires":[["85b44dfd.a9351"]]},{"id":"959e547e.d74c48","type":"inject
","z":"6ad0365d.c70448","name":"Temperature 32
```

```
Cels","topic":"temperature","payload":"32","payloadType":"num","repeat":"","crontab":"","once":false,
"onceDelay":0.1,"x":170,"y":1200,"wires":[["85b44dfd.a9351"]]}]
```

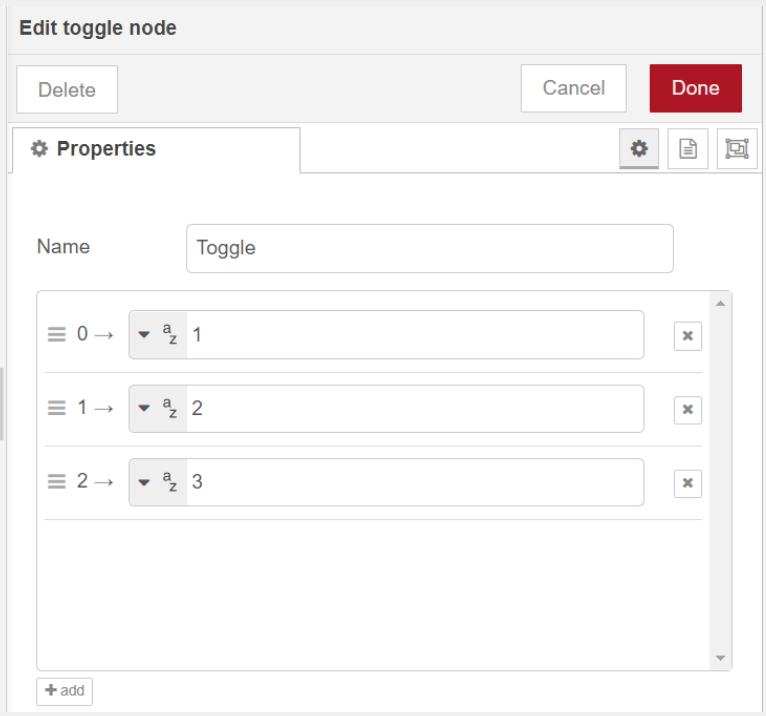
การสลับผล

การสลับค่าสถานะที่ต้องการส่งออก (Toggle) เช่น การสลับจากสถานะปิดและเปิดของสวิตช์ เป็นต้น โดยสถานะจะเปลี่ยนไปในแต่ละครั้งที่มีการส่งค่าข้อมูลใด ๆ ก็ตามเข้ามา

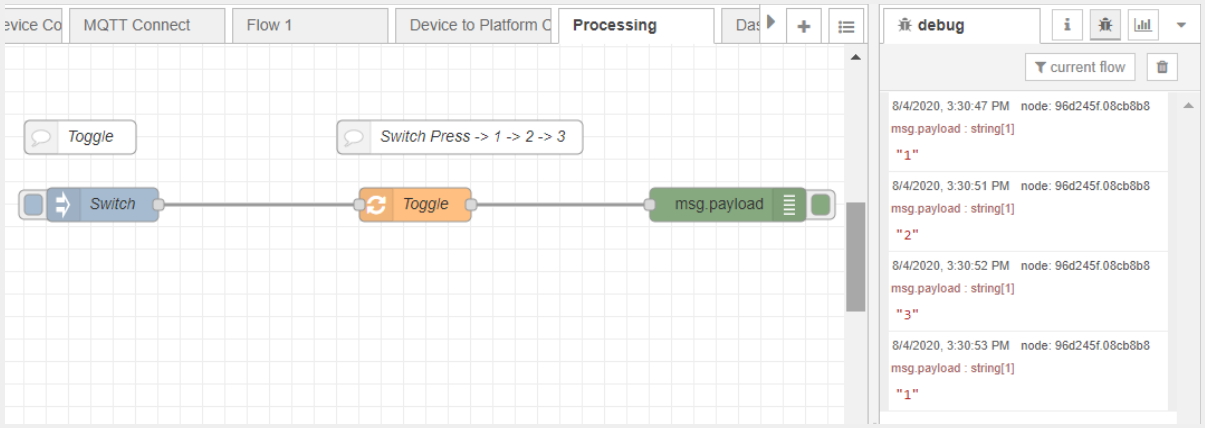
โหนดนี้ไม่จำเป็นต้องรับค่าข้อมูลใด ๆ เข้ามา แต่ให้ทำการตั้งค่าข้อมูลในแต่ละสถานะในโหนดนี้โดยตรง ซึ่งจะใช้การกระทุ้งของข้อมูลจากโหนดก่อนหน้าในการกระทุ้งและเปลี่ยนค่าข้อมูลแต่ละสถานะในกล่องข้อมูลนี้แทน

ตัวอย่าง เมื่อมีค่าข้อมูลเข้ามาแต่ละครั้ง จะแสดงผลค่าข้อมูลเป็น 1, 2, และ 3 ตามจำนวนครั้งที่มีการเข้ามาของข้อมูล และจะวนกลับไปที่เลข 1 อีกครั้งเมื่อมีข้อมูลเข้ามาเป็นครั้งที่ 4 และจะวนเช่นนี้ไปเรื่อย ๆ โดยไม่สนใจว่าค่าข้อมูลขาเข้าที่เข้ามาภายในโหนดนี้คือค่าใด

ภายในโหนด **‘Toggle’** ให้ทำการกำหนดแต่ละสถานะที่จะมีการเปลี่ยนแปลงเข้าไป โดยให้คลิกที่ปุ่ม ‘+ add’ เพื่อเพิ่มค่าสถานะต่าง ๆ เพิ่มเติม จากนั้นทำการคลิกที่ปุ่ม ‘Done’ เพื่อบันทึกการตั้งค่าของโหนดนี้



หลังจากเชื่อมต่อโหนดข้อความขาเข้าเข้ากับโหนดนี้ และเชื่อมต่อข้อมูลขาออกออกจากโหนดนี้ไปยังโหนด **‘Debug’** และทำการบันทึกหน้าโฟลว์นี้เรียบร้อยแล้ว เมื่อลองทำการส่งข้อความจากโหนดที่ทำการส่งข้อมูลเข้าไปทีละครั้ง จะพบว่าผลลัพธ์ที่ออกมาจะค่อย ๆ เปลี่ยนไปทีละเลข และวนกลับมาที่เลข 1 อีกครั้งเมื่อผ่านเลข 3 ไป

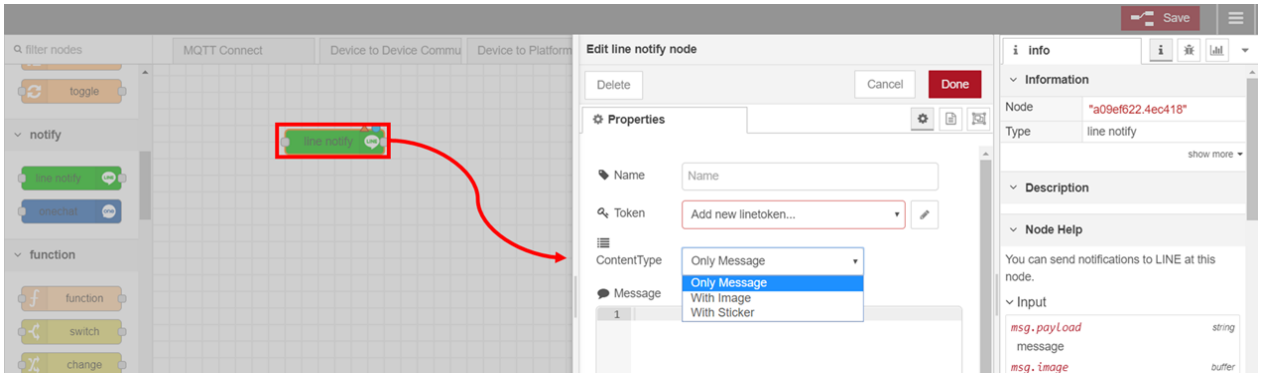



```
[{"id":"7b838286.4cb8dc","type":"toggle","z":"6ad0365d.c70448","name":"Toggle","rules":[{"v":"1","vt"
:"str","t":"eq"},{"v":"2","vt":"str","t":"eq"},{"v":"3","vt":"str","t":"eq"}],"x":390,"y":1040,"wires
":[["96d245f.08cb8b8"]]},{"id":"848209cd.6a5f58","type":"inject","z":"6ad0365d.c70448","name":"Switch
","topic":"","payload":"","payloadType":"str","repeat":"","crontab":"","once":false,"onceDelay":0.1,"
x":110,"y":1040,"wires":[["7b838286.4cb8dc"]]},{"id":"96d245f.08cb8b8","type":"debug","z":"6ad0365d.c
70448","name":"","active":true,"tosidebar":true,"console":false,"tostatus":false,"complete":"false","
x":670,"y":1040,"wires":[]}]
```

การแจ้งเตือน

นอกเหนือจากการใช้งานเพื่อสร้างโปรแกรมตามปกติแล้วนั้น ผู้ใช้งานยังสามารถเชื่อมต่อโฟลว์เข้ากับแอปพลิเคชั่นที่ผู้ใช้งานมีเพื่อทำการส่งการแจ้งเตือนของข้อมูลไปยังอุปกรณ์อื่น ๆ ของผู้ใช้งานได้อีกด้วย ซึ่งแอปพลิเคชั่นหลัก ๆ ที่มักจะถูกนำมาใช้งานก็คือ LINE นั่นเอง

ผู้ใช้งานสามารถนำโหนด **'LINE Notify'** ในหมวด ***'Notify'*** ลงมาวางในโฟลว์แล้วเชื่อมต่อกับ LINE โดยการนำ token ที่ได้รับมาจาก https://notify-bot.line.me/ มาใส่ลงไปในช่อง token โดยการคลิกที่รูปดินสอข้างกล่องการตั้งค่านี้เพื่อให้อุปกรณ์สามารถเชื่อมต่อกับบริการของ LINE ได้ การทำเช่นนี้จะสามารถทำให้ผู้ใช้งานสามารถส่งข้อความแจ้งเตือน พร้อมแนบรูปหรือสติกเกอร์ควบคู่ไปกับข้อความนั้นได้อีกด้วย

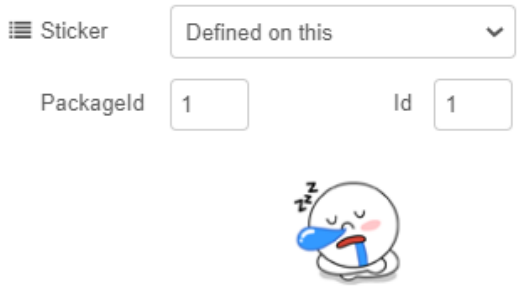


ในการส่งข้อความไปพร้อมการแนบรูปไปนั้น ผู้ใช้งานจำเป็นต้องนำรูปนั้นอัปโหลดขึ้นบนเว็บไซต์ฝากรูปใดก็ได้ก่อน จากนั้นจึงนำ url ที่อ้างอิงถึงตำแหน่งของรูปภาพที่ผู้ใช้งานอัปโหลดขึ้นไปนั้นมากรอกลงไปบนช่องการตั้งค่าเพิ่มเติมด้านล่างของช่องกรอกข้อความ โดยรูปที่ใช้นั้นมีอยู่ด้วยกันสองประเภท คือรูปขนาดเต็ม (Full Size) ขนาดไม่เกิน 2048 x 2048 พิกเซล และรูปขนาดย่อ (Thumbnail) ขนาดไม่เกิน 240 x 240 พิกเซล ซึ่งถ้าหากไม่มีการกำหนดรูปขนาดย่อเอาไว้ รูปขนาดเต็มที่ระบุไว้จะถูกนำมาใช้งานในส่วนนี้แทน

ส่วนสติกเกอร์นั้น ผู้ใช้งานสามารถกำหนดได้ว่าต้องการให้ใช้สติกเกอร์ที่กำหนดภายในโหนดนี้หรือจากค่าข้อมูลที่ถูกส่งมาจากโหนดก่อนหน้าที่มีตัวแปร `msg.stickerImageId` และ `msg.stickerId` ส่งควบคู่มากับค่าข้อมูลนั้นด้วย ซึ่งหากผู้ใช้งานต้องการให้ข้อความแนบค่าตัวแปรเหล่านี้มาด้วย ให้ทำการเลือก drop-down ในหมวดการตั้งค่า Sticker ให้กลายเป็น 'Design on

msg variable’ เพื่อให้ลักษณะการแสดงผลของสติกเกอร์นั้นขึ้นอยู่กับข้อความที่ถูกส่งมาจากโหนดก่อนหน้านั่นเอง

สติกเกอร์ที่จะปรากฏขึ้นมานั้น ขึ้นอยู่กับค่า Package ID และ ID ที่ผู้ใช้งานสามารถกำหนดได้เองโดยอ้างอิงจากตัวเลขในตารางรายการสติกเกอร์ของไลน์ที่ https://devdocs.line.me/files/sticker_list.pdf

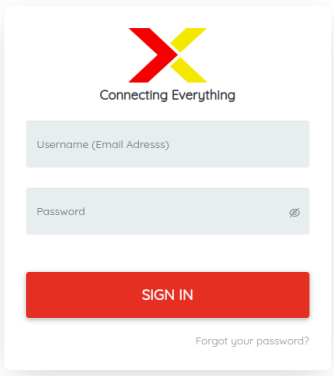


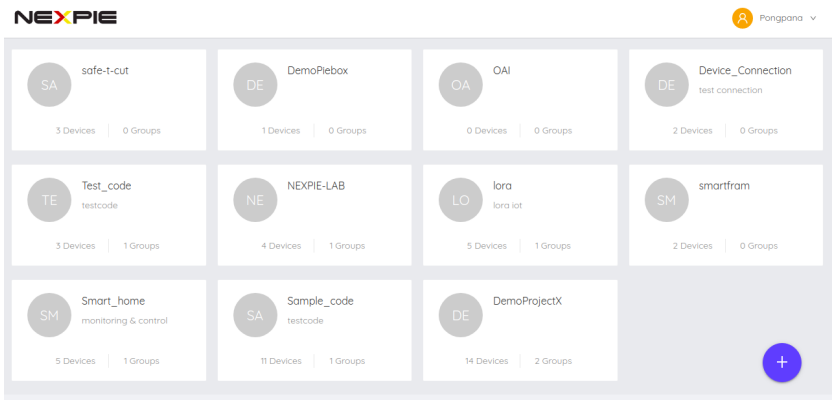
อ่านเซนเซอร์อุณหภูมิด้วย PIEBOX

เพื่อให้เข้าใจถึงการใช้งานตัวกล่อง PIEBOX เข้าร่วมกับโปรแกรม Flow Engine มากขึ้น นี่คือตัวอย่างการใช้งานกล่อง PIEBOX ร่วมกับเซนเซอร์วัดอุณหภูมิและความชื้นแบบ RS485 Modbus RTU โดยจะนำข้อมูลที่ได้รับจากอุปกรณ์นี้เชื่อมต่อไปยัง NEXPIE Platform ซึ่งการทำงานของตัวอย่างนี้จะประกอบไปด้วยสามขั้นตอน โดยเริ่มต้นจากการสร้างโปรเจกต์ใหม่ขึ้นมาบนแพลตฟอร์มของ NEXPIE จากนั้นจึงทำการเชื่อมต่อเข้ากับกล่อง PIEBOX และทำการนำข้อมูลที่อ่านได้จากเซนเซอร์ส่งไปแสดงผลยังหน้าแพลตฟอร์ม

สร้างโปรเจกต์บน NEXPIE Platform

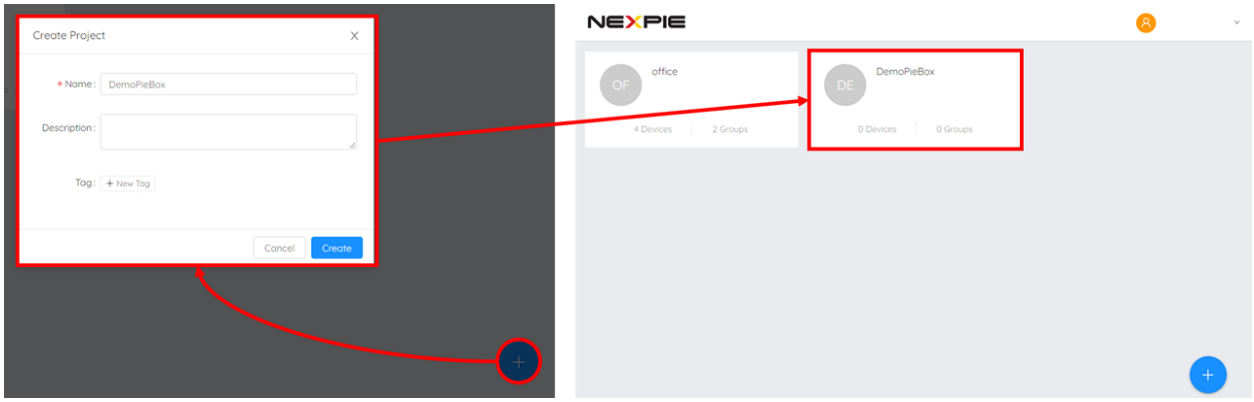
ในการสร้างโปรเจกต์ขึ้นมาบนหน้า NEXPIE Platform นั้น ผู้ใช้งานจำเป็นต้องมีบัญชีผู้ใช้ของ NEXPIE เสียก่อน จากนั้นจึงนำ Username และ Password ที่ได้ทำการสมัครไว้นั้นไปเข้าสู่ระบบที่ https://portal.nexpie.io ซึ่งหน้านี้จะถูกเรียกว่าหน้า *portal* ที่จะรวบรวมโปรเจกต์ต่าง ๆ ที่ผู้ใช้งานสร้างเอาไว้ภายในนั้น โดยหากผู้ใช้งานเข้าสู่ระบบได้สำเร็จ หน้าต่างโปรแกรมจะปรากฏเป็นดังภาพด้านล่าง



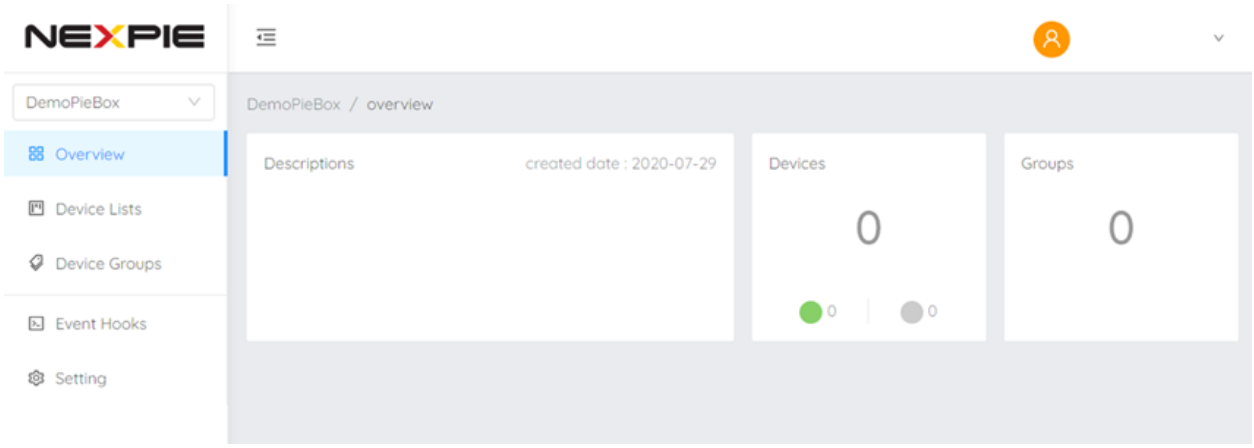


อย่าแปลกใจหากหน้าต่างโปรแกรมของคุณอาจไม่ได้ตรงกับภาพที่เห็นทุกประการ จากภาพนี้เป็นภาพหน้าต่าง portal ของผู้ใช้งานที่ทำการสร้างโปรเจกต์ไว้แล้วบางส่วน ซึ่งแต่ละกล่องสีขาวที่ปรากฏในหน้านี้นั้นแทนค่าหนึ่งโปรเจกต์ที่ผู้ใช้งานได้สร้างขึ้นมา ในกรณีที่ผู้ใช้งานต้องการสร้างโปรเจกต์ใหม่ ให้ทำการคลิกที่ปุ่ม **‘+’** สีฟ้าด้านล่างขวาของจอเพื่อเปิดหน้าต่างสร้างโปรเจกต์ใหม่ขึ้นมา

โดยให้ผู้ใช้งานทำการกรอกรายละเอียดของข้อมูลที่ระบุเอาไว้ด้วย * ให้ครบถ้วน จากนั้นจึงคลิกที่ปุ่ม ‘Create’ เพื่อทำการยืนยันการสร้างโปรเจกต์

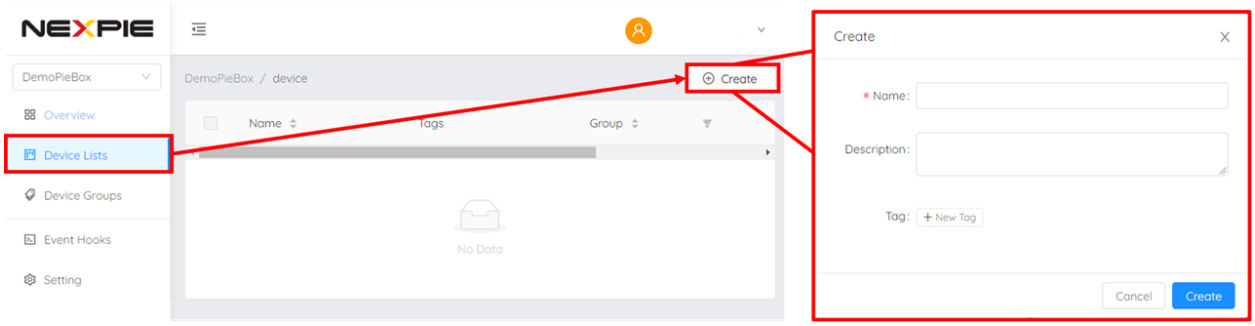


โปรเจกต์ที่ถูกสร้างขึ้นมาจะขึ้นมาปรากฏบนหน้าต่างโปรแกรมในลักษณะของกล่องสีขาวที่มีชื่อโปรเจกต์ตามที่ตั้งไว้เมื่อสักครู่ ผู้ใช้งานสามารถคลิกเข้าไปที่กล่องของโปรเจกต์นั้นเพื่อทำการเปิดหน้าโปรเจกต์ที่สร้างขึ้นมานี้ได้ โดยจะปรากฏเป็นหน้าต่างในลักษณะด้านล่างขึ้นมา

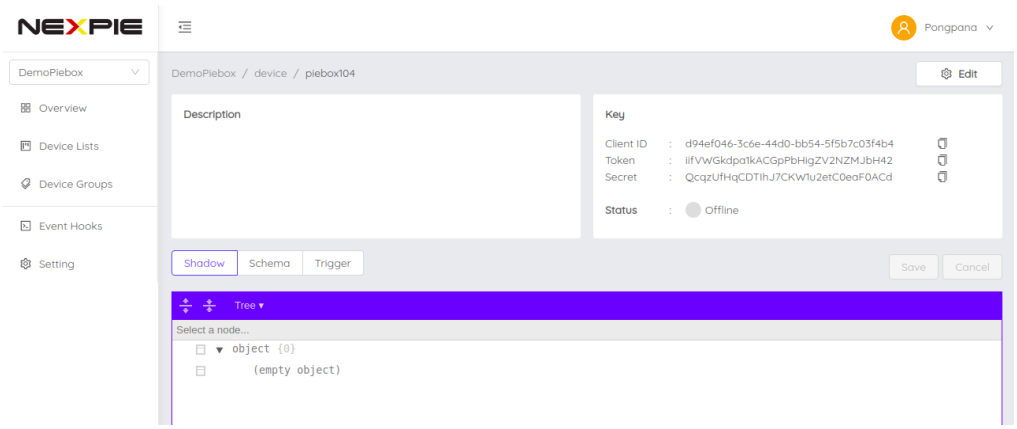


หน้าต่างนี้จะแสดงภาพรวมของโปรเจกต์นี้ว่าเป็นเช่นไรในขณะนี้ เช่น มีคำอธิบายเกี่ยวกับโปรเจกต์นี้ว่าอย่างไร มีอุปกรณ์ที่เชื่อมต่อกับหน้าโปรเจกต์นี้ทั้งหมดกี่ชิ้น และอยู่ในสถานะ *‘online’* และ *‘offline’* อย่างละกี่อุปกรณ์ เป็นต้น

ในขั้นตอนนี้ ผู้ใช้งานจำเป็นต้องสร้างอุปกรณ์ขึ้นมาภายในโปรเจกต์เพื่อให้ PIEBOX สามารถเชื่อมต่อมายังแพลตฟอร์มนี้ได้ โดยให้ทำการไปที่เมนู ‘Device Lists’ จากนั้นคลิกที่ปุ่ม ‘+ Create’ ที่อยู่บนหน้านั้นเพื่อทำการสร้างอุปกรณ์ใหม่ขึ้นมา โดยให้กรอกรายละเอียดตามที่ระบุไว้ด้วย * ให้ครบถ้วนและคลิก ‘Create’ เพื่อยืนยันการสร้างอุปกรณ์นั้น โดยอุปกรณ์ที่ถูกสร้างขึ้นมาจะปรากฏอยู่บนหน้าต่างโปรแกรมทันทีที่การสร้างนั้นเสร็จสมบูรณ์

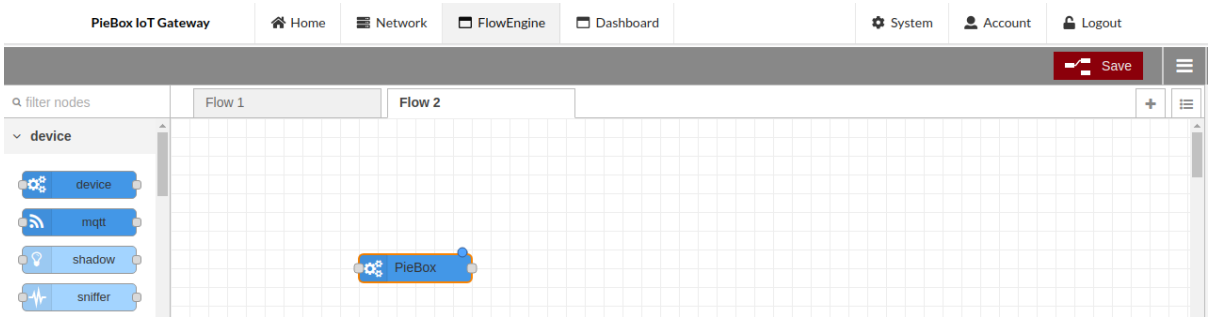


เมื่อผู้ใช้งานทำการ double-click เข้าไปที่ตัวอุปกรณ์ที่สร้างขึ้นมา ผู้ใช้งานจะพบกับหน้าต่างใหม่ที่แสดงรายละเอียดของอุปกรณ์นั้น รวมไปถึงค่า Client ID, Token และ Secret ที่ผู้ใช้งานสามารถนำไปใช้เพื่อให้อุปกรณ์อื่น ๆ สามารถเชื่อมต่อเข้ามายังแพลตฟอร์มนี้ได้ในช่อง key

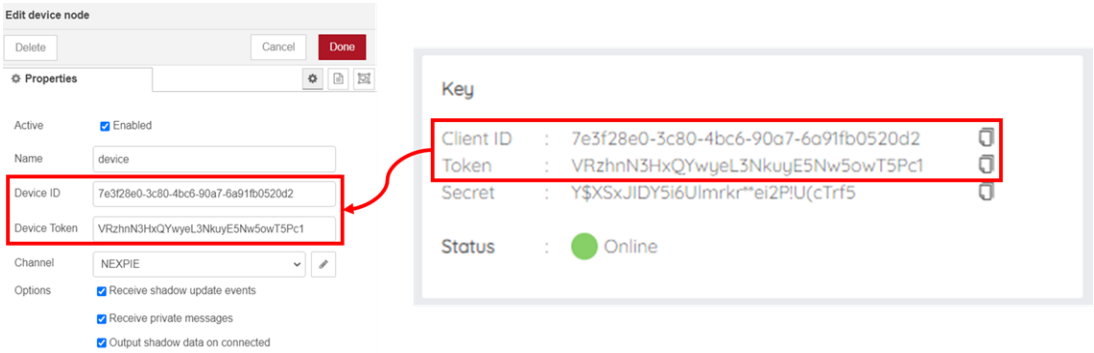


เชื่อมต่อ PIEBOX เข้ากับแพลตฟอร์ม

เมื่อผู้ใช้งานทำการเชื่อมต่อ PIEBOX เข้ากับคอมพิวเตอร์ได้เรียบร้อยแล้ว ให้ผู้ใช้งานทำการเข้าสู่ระบบที่ http://piebox.local (ค่าเริ่มต้นคือคำว่า ‘admin’ ทั้ง Username และ Password) และไปยังแถบเมนู Flow Engine เพื่อเปิดโฟลว์ขึ้นมา จากนั้นให้ทำการลากโหนด **‘Device’** ในหมวดหมู่ ***‘Device’*** ออกมาจากแพลิตต์แล้ววางลงบนโฟลว์



การที่โหนด **‘Device’** นี้จะทำการเชื่อมต่อเข้ากับอุปกรณ์ที่ถูกสร้างไว้ในหน้า *portal* ได้นั้น ผู้ใช้งานต้องทำการตั้งค่าโหนดนี้โดยการนำค่า Client ID และ Token มาระบุลงไปในช่อง Device ID และ Device Token ของโหนด **‘Device’** ตามลำดับ โดยให้เปลี่ยน Channel ของโหนดเป็น ‘NEXPIE’ จากนั้นจึงทำการบันทึกการตั้งค่าของโหนดโดยการคลิกที่ปุ่ม ‘Done’ เพื่อยืนยันการเปลี่ยนแปลง

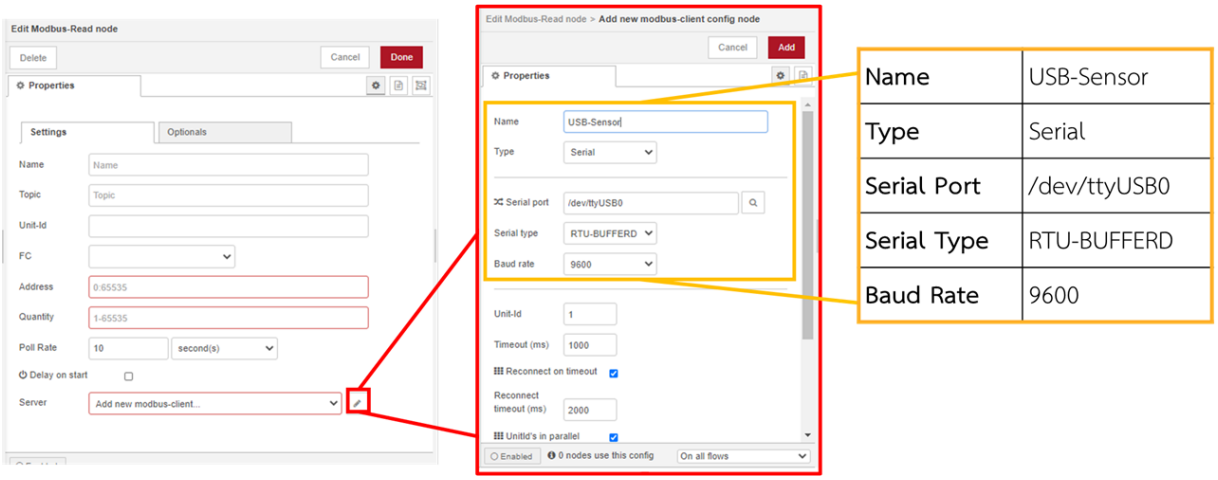


ทำการบันทึกหน้าโฟลว์โดยการคลิกที่ปุ่ม ‘Save’ ด้านบนของหน้าต่างโปรแกรม หากอุปกรณ์ที่อยู่บนหน้าโฟลว์สามารถเชื่อมต่อเข้ากับแพลตฟอร์มได้สำเร็จ โหนดนี้จะปรากฏจะสถานะสีเขียวพร้อมคำว่า *‘Connected’* ขึ้นทางด้านล่างของโหนด ซึ่งหมายถึงอุปกรณ์นี้พร้อมที่จะส่งข้อมูลเซนเซอร์ที่ได้รับแล้ว แต่หากการเชื่อมต่อนั้นยังไม่สำเร็จ ให้ผู้ใช้งานลองตรวจสอบค่าข้อมูลที่กรอกลงไปในโหนดนี้ว่าถูกต้องหรือไม่ หรือ PIEBOX นั้นยังทำการเชื่อมต่ออินเตอร์เน็ตอยู่หรือไม่

อ่านข้อมูลเซนเซอร์ด้วย Modbus RTU

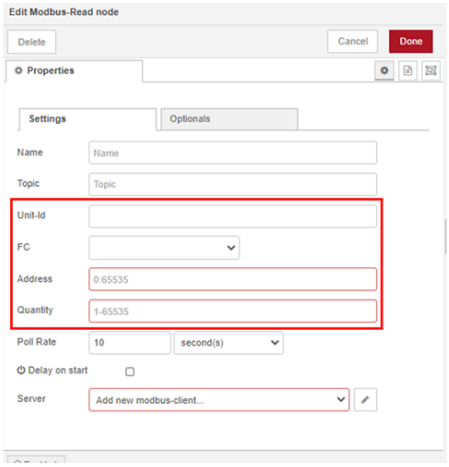
Modbus RTU คือเซนเซอร์ที่ใช้ในการวัดค่าอุณหภูมิและ/หรือความชื้นที่เกิดขึ้นใกล้ ๆ ตัวอุปกรณ์ ซึ่งผู้ใช้งานสามารถอ่านค่าข้อมูลจากเซนเซอร์ได้ภายในโฟลว์ของ Flow Engine โดยการนำโหนด **‘Modbus Read’** ในหมวดหมู่ ***‘Modbus’*** จากแพลิตต์ลากลงมาใช้งานในโฟลว์ จากนั้นให้ทำการ double-click เข้าไปที่โหนดนี้เพื่อทำการระบุพอร์ตที่ใช้ในการสื่อสารกับเซนเซอร์ลงไป โดยให้เลื่อนหาการตั้งค่าในส่วน ‘Server’ แล้วเลือกเป็น ‘Add new modbus-client...’ จากนั้นจึงทำการ

คลิกที่ปุ่มดินสอข้าง ๆ ส่วนการตั้งค่านั้นเพื่อระบุพอร์ตลงไปตามที่ปรากฏในตาราง จากนั้นจึงกด ‘Add’ เพื่อยืนยันการระบุพอร์ตนั้น



ทำการระบุค่าข้อมูลที่ต้องการอ่านจากเซนเซอร์ว่าต้องการอ่านเฉพาะค่าอุณหภูมิ เฉพาะค่าความชื้น หรืออ่านทั้งสองค่า โดยใช้ตารางด้านล่างในการระบุประเภทของค่าที่ต้องการอ่านจากเซนเซอร์ Modbus แบบ RS485 รุ่นโมดูล xy-md02 โดยให้นำค่าเหล่านี้กรอกลงไปในช่องการตั้งค่าต่าง ๆ ของโหนดนี้

Data	Unit-Id	Function code (FC)	Address	Quantity
เฉพาะอุณหภูมิ	1	4	1	1
เฉพาะความชื้น	1	4	2	1
อุณหภูมิและความชื้น	1	4	1	2

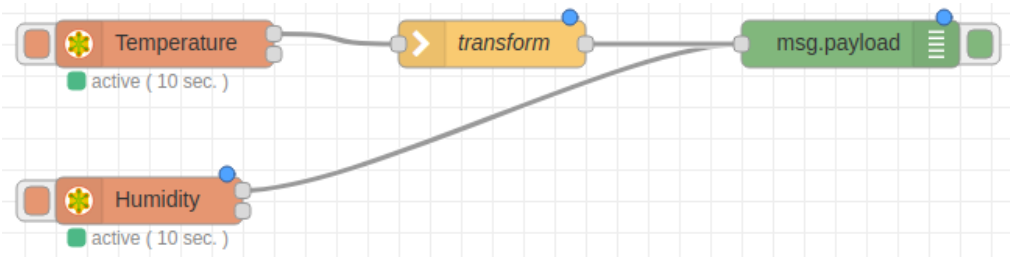


ในที่นี้ ผู้เขียนได้ทำการสร้างโหนดของเซนเซอร์ขึ้นมาสองโหนด โดยให้โหนดหนึ่งอ่านค่าอุณหภูมิ (Temperature) และอีกโหนดหนึ่งอ่านค่าความชื้น (Humidity) ซึ่งจะแสดงผลค่าข้อมูลที่ได้รับจากทั้งสองโหนดนี้ผ่านการลากเชื่อมต่อเข้ากับโหนด **‘Debug’** เพื่อให้ข้อมูลปรากฏขึ้นบนแถบหน้าต่างดีบักด้านขวามือของหน้าต่างโปรแกรม

Modbus RTU สามารถอ่านค่าข้อมูลอุณหภูมิได้ก็จริง แต่ค่าอุณหภูมิที่เข้ามานั้นยังไม่อยู่ในรูปที่สามารถนำไปใช้งานได้จริง เนื่องจากค่าที่ถูกส่งเข้ามายังสูงเกินกว่าความเป็นจริงอยู่ถึง 10 เท่า ดัง

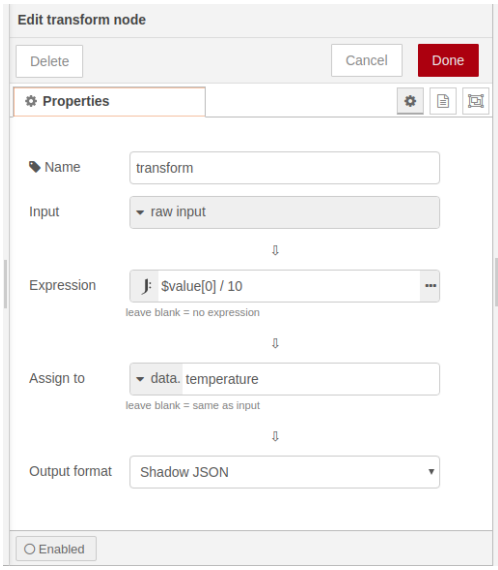
นั้นผู้ใช้งานจึงอาจจะต้องหารค่าข้อมูลที่ได้รับด้วย 10 เสียก่อน เพื่อให้เซนเซอร์สามารถส่งผลของข้อมูลออกมาได้อย่างถูกต้องตามหน่วยที่มันควรจะเป็น

โหนด **‘Transform’** นั้นสามารถนำมาใช้ประโยชน์ได้ในส่วนนี้ โดยให้ทำการลากวางโหนดนี้ไว้บนเส้นเชื่อมต่อระหว่างโหนด Modbus ที่ทำการอ่านค่าอุณหภูมิและโหนด Debug จากนั้นให้ทำการ double-click ที่ตัวโหนดเพื่อเปิดแถบหน้าต่างแก้ไขด้านขวาขึ้นมา



ภายในหน้าต่างแก้ไขของโหนดนี้ ให้ผู้ใช้งานทำการระบุว่าข้อมูลที่ได้รับเข้ามาจากเซนเซอร์นั้นเป็น ‘ค่าข้อมูลดิบ’ (Raw Input) จากนั้นให้ทำการตั้งสมการ (Expression) โดยการใช้ตัวแปร `$value[0]` เพื่อแทนค่าเซนเซอร์วัดอุณหภูมิของอุปกรณ์ที่ได้รับลงไป แล้วหารค่านี้ด้วย 10 ซึ่งจะเขียนสมการได้ ดังนี้

`$value[0] / 10`

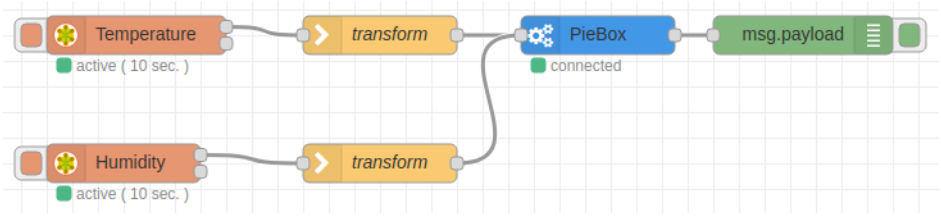


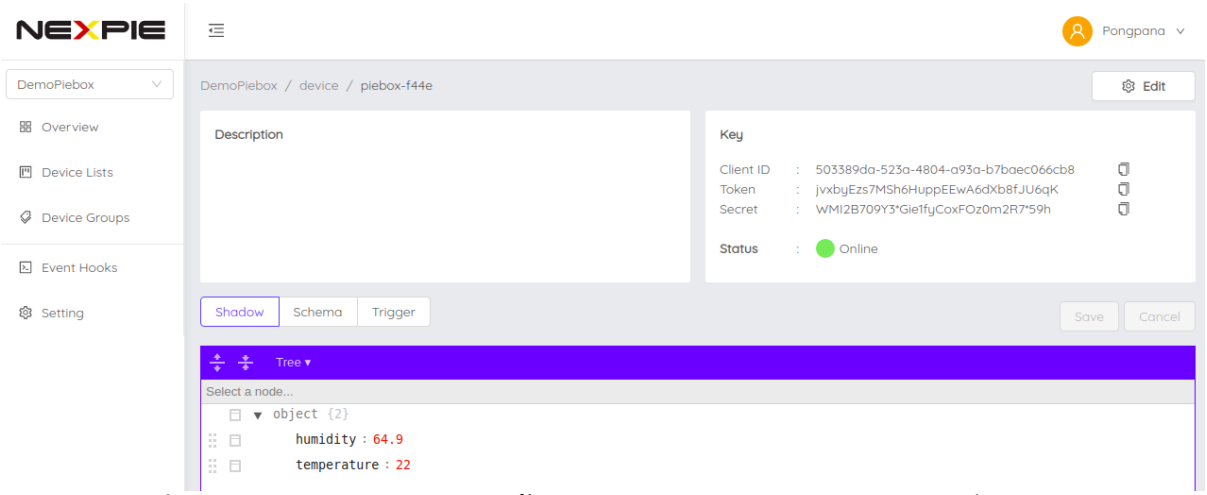
เมื่อทำการกำหนดสมการเรียบร้อยแล้ว หากต้องการให้ค่าข้อมูลที่ถูกคำนวณแล้วสามารถส่งไปแสดงผลยังแพลตฟอร์มได้ ผู้ใช้งานต้องทำการแปลงค่าที่ได้รับให้อยู่ในรูปแบบของ Shadow เสียก่อน ซึ่งผู้ใช้งานสามารถแปลงลักษณะของข้อมูลได้โดยตรงในโหนดนี้ โดยการตั้งค่า ‘Assign to’ ไปที่ *‘data.’* จากนั้นจึงเขียนชื่อของตัวแปร (Field) ที่ต้องการใช้ลงไป โดยในที่นี้จะตั้งชื่อว่า *‘temperature’* ตามลักษณะของค่าข้อมูลที่เป็นการอ่านค่าอุณหภูมิ และให้ตั้ง ‘Output format’ เป็น ‘Shadow JSON’ เพื่อทำการจัดรูปแบบของข้อมูลที่ออกไปในรูปแบบของ Shadow แล้วทำการคลิกที่ปุ่ม ‘Done’ เพื่อยืนยันการตั้งค่านี้

เมื่อผู้ใช้งานบันทึกหน้าโฟลว์นี้หลังการตั้งค่าเรียบร้อยแล้ว ข้อมูลที่ปรากฏบนแถบหน้าต่างดีบักจะขึ้นเป็นค่าข้อมูลในลักษณะ Shadow ซึ่งพร้อมต่อการนำส่งข้อมูลนั้นไปยังแพลตฟอร์ม NEXPIE โดยผู้ใช้งานสามารถทำเช่นนี้กับโหนดเซนเซอร์วัดค่าความชื้นได้เช่นเดียวกัน

ส่งข้อมูลไปยังแพลตฟอร์ม

หลังจากเปลี่ยนแปลงค่าข้อมูลจนเรียบร้อยแล้ว ผู้ใช้งานสามารถลากเชื่อมต่อโหนด **‘Transform’** ที่เปลี่ยนแปลงค่าข้อมูลของเซนเซอร์เหล่านั้นเข้ากับโหนด **‘Device’** เพื่อส่งค่าข้อมูลสุดท้ายไปยังแพลตฟอร์มได้ ซึ่งผลลัพธ์ของค่าข้อมูลนี้จะปรากฏขึ้นบนหน้าแพลตฟอร์ม ดังรูป



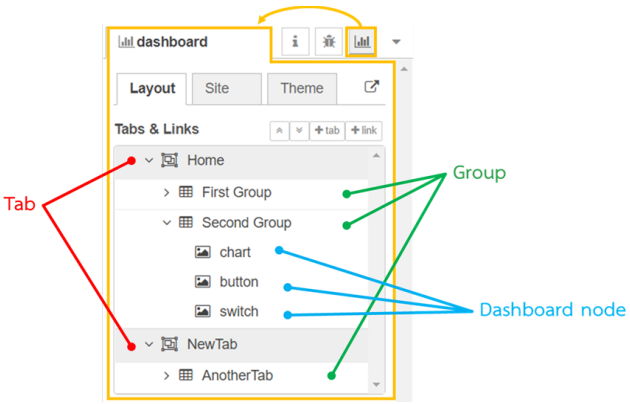


ข้อมูลที่ถูกส่งเข้าไปยังแพลตฟอร์มนั้นจะแสดงผลเฉพาะค่าข้อมูลล่าสุดที่ถูกส่งเข้าไปในแพลตฟอร์มเท่านั้น หากผู้ใช้งานต้องการเก็บค่าข้อมูลย้อนหลัง ผู้ใช้งานสามารถทำได้โดยการกำหนดรูปแบบของข้อมูลที่ต้องการจัดเก็บในสคีมา (Schema) ซึ่งวิธีการในการเขียนสคีมานั้น ผู้ใช้งานสามารถอ่านเพิ่มเติมได้ที่ https://docs.nexpie.io/device-config.html#device-schema

แดชบอร์ด

นอกเหนือจากการสร้างหน้าโปรแกรมผ่านโฟลว์ตามปกติแล้ว ผู้ใช้งานยังสามารถสร้างแดชบอร์ด (Dashboard) ขึ้นมาได้อีกด้วย ซึ่งแดชบอร์ดนี้จะปรากฏให้ผู้ใช้งานเห็นและควบคุมมันได้ในแถบเมนู **‘Dashboard’** ภายในหน้าโปรแกรมของ PIEBOX

แดชบอร์ดจะเป็นตัวช่วยให้ผู้ใช้งานสามารถเข้าใจถึงค่าข้อมูลในตัวแปรต่าง ๆ ที่ได้รับในรูปแบบของแผงหน้าปัดประเภทต่าง ๆ เช่น กราฟ มาตรวัด หลอดไฟ เป็นต้น อีกทั้งยังอนุญาตให้ผู้ใช้งานสามารถมีส่วนร่วมหรือสร้างค่าข้อมูลใหม่ ๆ โดย ตรงผ่านแดชบอร์ดได้ เช่น การกดปุ่มต่าง ๆ การพิมพ์ข้อความใส่ลงไป เป็นต้น

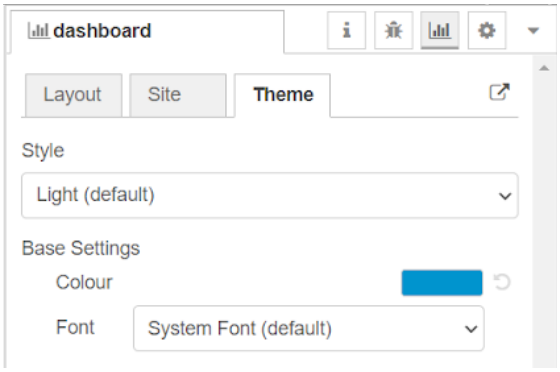


ในการใช้งานแดชบอร์ดนั้น ผู้ใช้งานจำเป็นต้องใช้ควบคู่กับแถบเมนูแดชบอร์ด (Dashboard) ที่อยู่ด้านขวามือของโปรแกรม ซึ่งภายภายในแถบเมนูนี้ จะประกอบไปด้วยรายละเอียดเกี่ยวกับโหนดของหน้าต่างการแสดงผลต่าง ๆ ว่าถูกจัดกลุ่มไว้ให้อยู่ในกลุ่มใด (Group) ในแถบหน้าต่างใด (Tab) ของแดชบอร์ด อีกทั้งยังอนุญาตให้ผู้ใช้งานเพิ่มหรือแก้ไขแถบการแสดงผลและกลุ่มต่าง ๆ จากตรงนี้ได้โดยตรงอีกด้วย

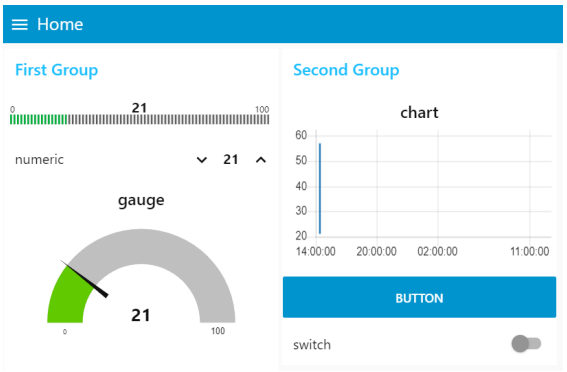
โหนดของแดชบอร์ดแต่ละโหนดนั้น เมื่อนำมาใช้งานจะถูกเรียกว่าเป็น ‘วิดเจ็ท’ (Widget) ซึ่ง Flow Engine มีวิดเจ็ทหลายประเภทที่อนุญาตให้ผู้ใช้งานนำมาสร้างหน้าแดชบอร์ดได้ ทั้งวิดเจ็ทที่ใช้ในการแสดงผลเพียงอย่างเดียว เช่น มาตรวัด (Gauge) หรือขีดระดับ (Level) เป็นต้น หรือวิดเจ็ทที่สามารถสร้างค่าข้อมูลใหม่ขึ้นมาโดยตัวมันเองได้ เช่น แถบเลื่อนเส้นจำนวน (Slider) หรือตัวเลือกตัวเลข (Numeric) เป็นต้น

การตั้งค่าลักษณะของแดชบอร์ด

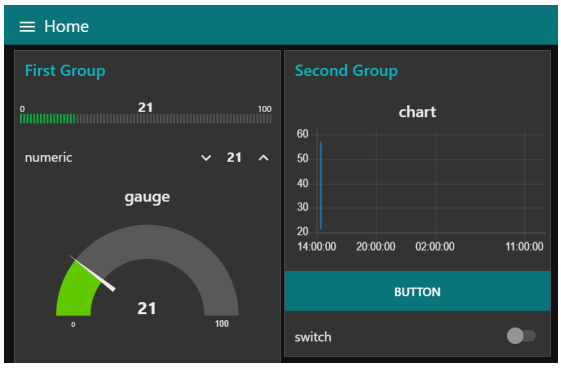
ผู้ใช้งานสามารถตั้งค่าลักษณะการแสดงผลของหน้าต่างการแสดงผลได้โดยไปยังแถบ ‘Theme’ ภายในแถบเมนูข้าง ‘Dashboard’ ซึ่งการตั้งค่าจะประกอบไปด้วยส่วนต่าง ๆ ดังนี้



- **Style** คือลักษณะของการแสดงผล มีให้เลือก 3 รูปแบบ ได้แก่
    - ***Light*** เป็นค่าเริ่มต้นของการแสดงผล โดยจะแสดงผลแบบสว่าง

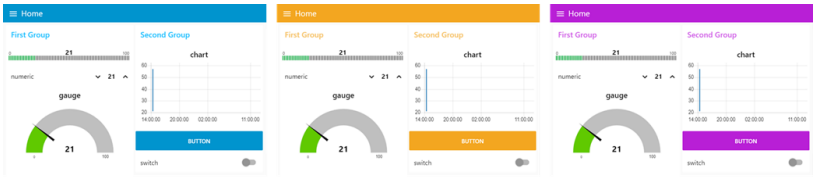


    - ***Dark*** คือลักษณะการแสดงผลที่จะแสดงในลักษณะที่มืด



    - ***Custom*** จะอนุญาตให้ผู้ใช้งานสามารถตั้งค่ารูปแบบในการแสดงผลได้เอง

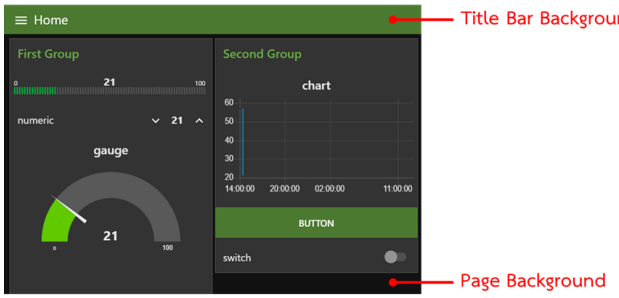
- **Base Settings** คือการตั้งค่าการแสดงผลพื้นฐานอันได้แก่
    - ***Colour*** คือสีที่จะใช้โดยรวมภายในแดชบอร์ดนั้น ซึ่งค่าเริ่มต้นของสีที่ใช้งานคือ ■ #0094ce โดยผู้ใช้งานสามารถกลับไปยังสีเริ่มต้นนี้ได้ด้วยการคลิกที่ปุ่มคืนค่า (Refresh; ↻) ที่อยู่ด้านหลังกล่องเลือกสีนั้น



#0094ce (ค่าเริ่มต้น) #f3a620 #b71fd6

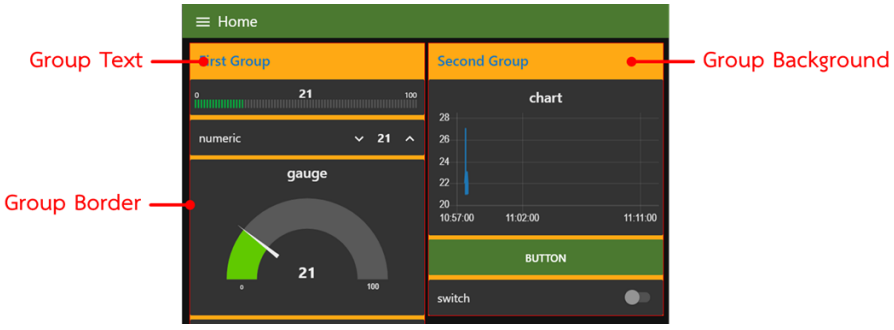
- ***Font*** คือรูปแบบอักษรที่ต้องการใช้ภายในแดชบอร์ด

- **Page Settings** จะปรากฏให้ตั้งค่าหากเลือกการตั้งค่ารูปแบบในการแสดงผลเอง (Custom) โดยจะเป็นการตั้งค่าสีที่จะปรากฏในแต่ละแถบการแสดงผล



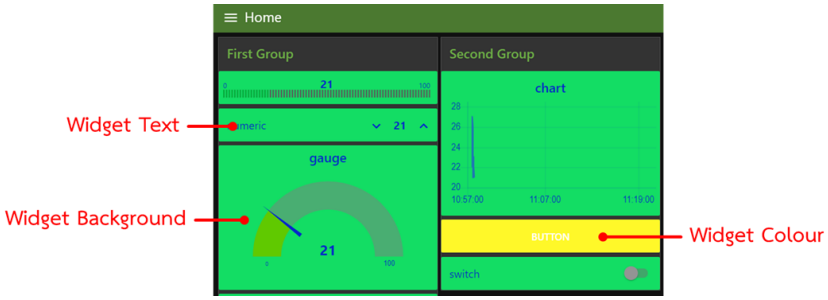
  - ***Title Bar Background*** คือการกำหนดสีของพื้นหลังชื่อของแถบแดชบอร์ด
  - ***Page Background*** คือการกำหนดสีพื้นหลังของแดชบอร์ด
  - ***Side Bar Background*** คือการกำหนดสีพื้นหลังของแถบด้านข้างในแดชบอร์ด

- **Group Settings** จะปรากฏให้ตั้งค่าหากเลือกการตั้งค่ารูปแบบในการแสดงผลเอง (Custom) โดยจะเป็นการตั้งค่าเกี่ยวกับกลุ่มการแสดงผลต่าง ๆ



  - ***Group Text*** คือการตั้งค่าสีข้อความของชื่อกลุ่ม
  - ***Group Border*** คือการตั้งค่าขอบของกลุ่ม
  - ***Group Background*** คือการตั้งค่าพื้นหลังของกลุ่ม

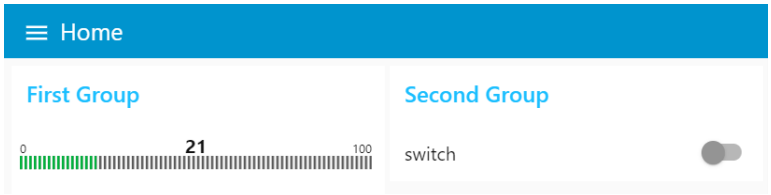
- **Widget Settings** จะปรากฏให้ตั้งค่าหากเลือกการตั้งค่ารูปแบบในการแสดงผลเอง (Custom) โดยจะเป็นการตั้งค่าเกี่ยวกับวิดเจ็ทต่าง ๆ



  - ***Widget Text*** จะเป็นการตั้งสีของข้อความภายในวิดเจ็ทต่าง ๆ
  - ***Widget Colour*** เป็นการตั้งค่าสีหลักของวิดเจ็ทนั้น
  - ***Widget Background*** คือการตั้งค่าพื้นหลังของแต่ละวิดเจ็ท

กลุ่มของวิดเจ็ท

กลุ่ม (Group) คือการจัดหมวดหมู่ของโหนดแต่ละโหนดที่อยู่ภายในหน้าเดียวกันให้แยกออกจากกันเพื่อความสะดวกต่อการอ่านผลข้อมูลของผู้ใช้งาน

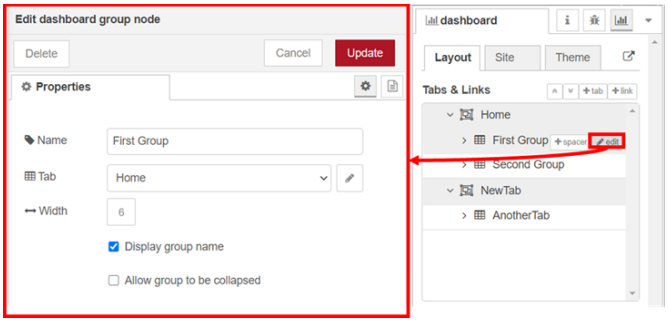


ค่าข้อมูลและข้อความต่างที่ถูกส่งผ่านแต่ละโหนดข้อมูลนั้นยังคงเชื่อมต่อกันตามลักษณะของโฟลว์เช่นเดิม แม้แต่ละโหนดจะถูกจัดแยกอยู่ในต่างกลุ่มกันก็ตาม

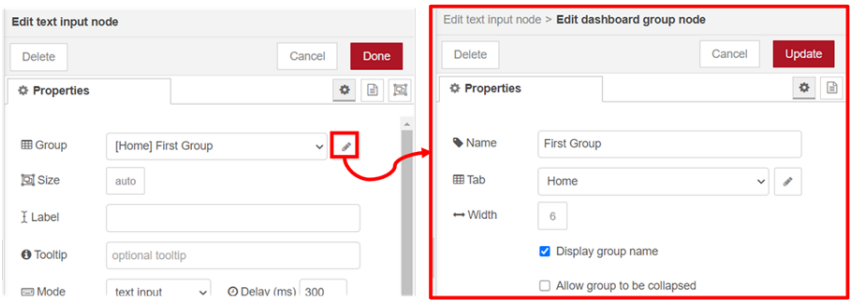
การแก้ไขกลุ่มของวิดเจ็ท

ผู้ใช้งานสามารถแก้ไขกลุ่มของวิดเจ็ทนี้ได้ 2 วิธี

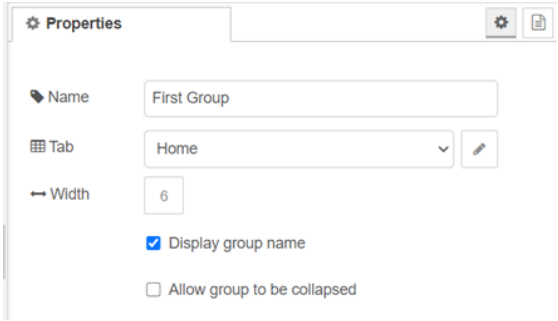
- นำเมาส์ไปลอยเหนือกลุ่มของวิดเจ็ทที่ต้องการแก้ไขในแถบเมนู ‘Dashboard’ จากนั้นคลิกที่ปุ่ม ‘Edit’ เพื่อให้ปรากฏหน้าต่างแก้ไขขึ้นมา



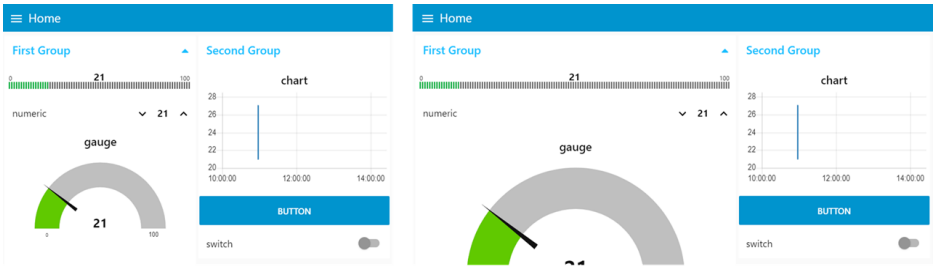
- คลิกที่ปุ่มดินสอ ( ) ที่ข้าง drop-down ของ ‘Group’ ภายในโหนดที่อยู่ภายใต้แถบหน้าต่างที่ต้องการแก้ไขนั้นเพื่อเปิดหน้าต่างแก้ไขขึ้นมา



ภายในหน้าต่างแก้ไขนั้น ผู้ใช้งานสามารถแก้ไขกลุ่มที่โหนดนั้นไปอยู่ได้ โดยมีส่วนในการแก้ไขกลุ่มทั้งหมด 5 ส่วน

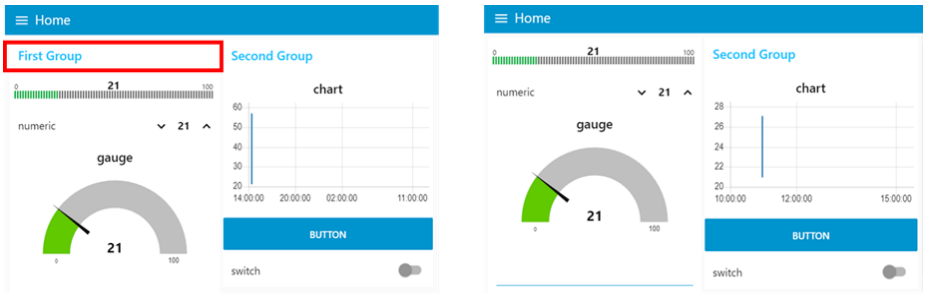


- **Name** คือชื่อกลุ่มของวิดเจ็ทเหล่านี้
- **Tab** คือการแถบแดชบอร์ดที่กลุ่มของวิดเจ็ทเหล่านี้จะไปปรากฏอยู่ ซึ่งผู้ใช้งานสามารถเปลี่ยนแปลงได้โดยคลิกที่ drop-down เพื่อเลือกแถบหน้าต่างอื่นที่ต้องการให้กลุ่มนี้ไปอยู่
- **Width** คือการกำหนดขนาดความกว้างของกลุ่มที่อยู่ภายในแดชบอร์ดนั้น โดยผู้ใช้งานสามารถทำการเปลี่ยนแปลงขนาดได้โดยการลากขยายขนาดที่ต้องการเมื่อคลิกที่ปุ่มนี้ขึ้นมา (ค่าเริ่มต้นอยู่ที่ 6 หน่วย)



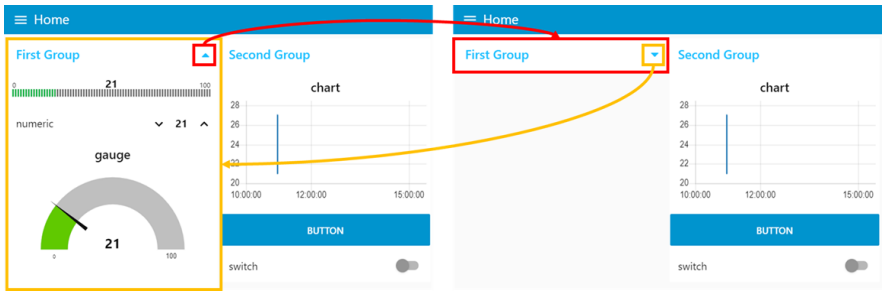
'First Group' Width = 6 'First Group' Width = 10

- **Display group name** คือการกำหนดว่าต้องการให้มีการแสดงชื่อกลุ่มในแดชบอร์ดหรือไม่ โดยหากปิดการใช้งาน ชื่อของกลุ่มนั้นจะไม่ปรากฏขึ้นในแดชบอร์ด



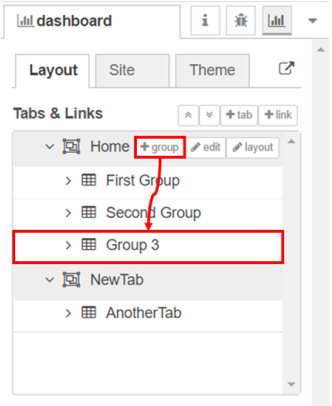
Display *'First Group'* name Disable display *'First Group'* name

- **Allow group to be collapse** จะปรากฏให้ตั้งค่าหากเปิดการใช้งาน '*Display group name*' โดยจะเป็นการกำหนดว่าต้องการให้กลุ่มของวิดเจ็ทนั้นสามารถย่อและขยายการแสดงผลได้หรือไม่ ซึ่งจะปรากฏในลักษณะลูกศรที่อยู่ข้างชื่อกลุ่มภายในแดชบอร์ดนั้น ๆ



การสร้างกลุ่มใหม่

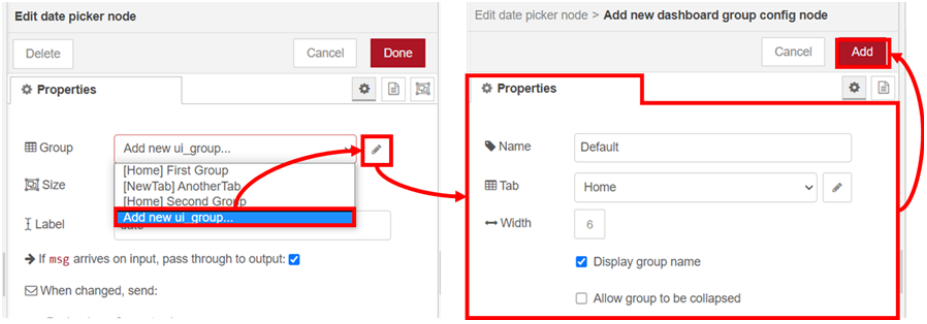
ในการสร้างกลุ่มฟังก์ชันการแสดงผลใหม่ขึ้นมานั้น ให้ผู้ใช้งานนำเมาส์ไปลอยเหนือแถบแดชบอร์ดที่ต้องการสร้างกลุ่มฟังก์ชันใหม่ขึ้นมา แล้วทำการคลิกที่ **'+ group'** เพื่อสร้างกลุ่มฟังก์ชันใหม่ขึ้นมา



วิดเจ็ทแต่ละวิดเจ็ทจำเป็นต้องมีการกำหนดกลุ่มที่ต้องการให้วิดเจ็ทนั้นไปอยู่ โดยวิดเจ็ทหนึ่งวิดเจ็ทสามารถอยู่ได้เพียงกลุ่มเดียวเท่านั้น ในขณะที่ภายในหนึ่งกลุ่มสามารถมีวิดเจ็ทอยู่ภายในนั้นได้มากกว่า 1 วิดเจ็ท

การสร้างกลุ่มใหม่จากภายในโหนด

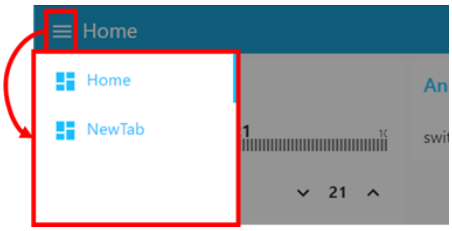
ผู้ใช้งานสามารถกลุ่มให้แก่โหนดที่เพิ่งลากวางลงมาในโฟลว์ใหม่ได้โดยตรงในการตั้งค่าภายในโหนดนั้น โดยให้ทำการคลิกที่โหนดเพื่อเปิดการตั้งค่าขึ้นมา แล้วเลือก drop-down ภายในแถบ ‘Group’ ไปที่ ‘*Add new_ui group…*’ แล้วคลิกที่ปุ่มดินสอ ( ) ที่อยู่ด้านข้างแถบนั้นเพื่อทำการเข้าไปแก้ไขค่าข้อมูลของกลุ่มนั้นด้านใน จากนั้นจึงทำการกดที่ปุ่ม ‘Add’ เพื่อยืนยันการสร้างกลุ่มนั้นขึ้นมา โดยโหนดที่ใช้ในการแก้ไขนั้นจะถูกจัดให้อยู่ในกลุ่มนั้นโดยอัตโนมัติ



อย่างไรก็ตาม การสร้างกลุ่มของโหนดการแสดงผลขึ้นมานั้นจำเป็นต้องมีการสร้างแถบของแดชบอร์ดที่ต้องการให้กลุ่มนั้นไปอยู่ขึ้นมาเสียก่อน จึงจะสามารถสร้างกลุ่มของวิดเจ็ทขึ้นมาได้

แถบหน้าต่าง

แถบหน้าต่าง (Tab) จะเป็นการแบ่งหน้าต่างของโหนดให้แยกออกจากกันเป็นคนละหน้า โดยผู้ใช้งานสามารถเข้าไปยังหน้าต่างอื่น ๆ ของแดชบอร์ดได้ด้วยการเปิดเมนูด้านซ้ายขึ้นมา แล้วเลือกไปยังแถบหน้าต่างที่ต้องการให้แสดงผล

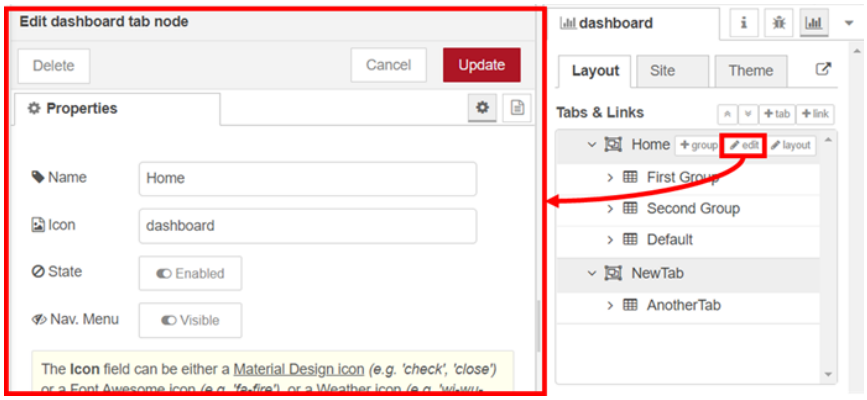


ค่าข้อมูลและข้อความต่างที่ถูกส่งผ่านแต่ละโหนดข้อมูลนั้นยังคงเชื่อมต่อกันตามลักษณะของโฟลว์เช่นเดิม แม้แต่ละโหนดจะถูกจัดแยกอยู่ในต่างแถบแดชบอร์ดกันก็ตาม

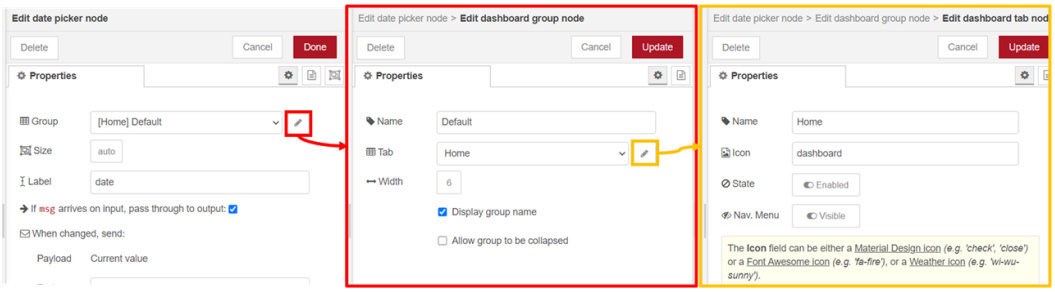
การแก้ไขแถบหน้าต่าง

ผู้ใช้งานสามารถแก้ไขแถบหน้าต่างนี้ได้ 2 วิธี

- นำเมาส์ไปลอยเหนือแถบหน้าต่างที่ต้องการแก้ไขในแถบเมนู ‘Dashboard’ จากนั้นคลิกที่ปุ่ม ‘Edit’ เพื่อให้ปรากฏหน้าต่างแก้ไขขึ้นมา



- คลิกที่ปุ่มดินสอ ( ) ที่ข้าง drop-down ของ ‘Group’ ภายในโหนดที่อยู่ภายใต้แถบหน้าต่างที่ต้องการแก้ไขนั้นเพื่อเปิดหน้าต่างแก้ไขกลุ่มของวิดเจ็ทขึ้นมา จากนั้นคลิกที่ปุ่มดินสอ ( ) ที่อยู่ด้านข้าง ‘Tab’ ภายในหน้าการตั้งค่ากลุ่มนั้นอีกทีเพื่อเปิดหน้าแก้ไขแถบแดชบอร์ดขึ้นมา

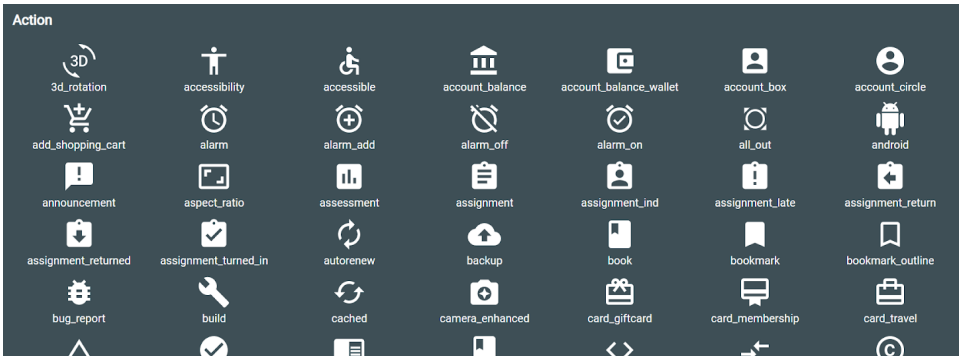


เมื่อผู้ใช้งานทำการแก้ไขแถบแดชบอร์ดนั้นเรียบร้อยแล้ว ให้ทำการคลิกที่ปุ่ม ‘Update’ เพื่อยืนยันการแก้ไข

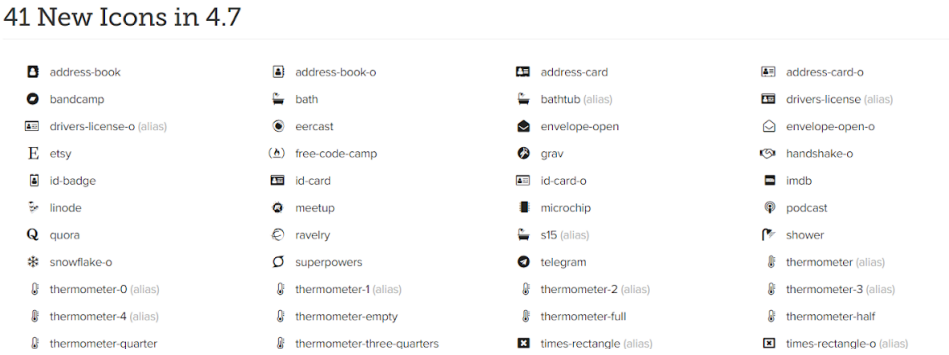
ภายในหน้าต่างแก้ไขนั้น ผู้ใช้งานสามารถแก้ไขได้ทั้งหมด 4 ส่วน ได้แก่

- **Name** คือชื่อของแถบแดชบอร์ดนั้น ๆ

- **Icon** คือสัญลักษณ์ของแถบแดชบอร์ดนั้น ๆ (ค่าเริ่มต้นคือ ‘dashboard’) ซึ่งผู้ใช้งานสามารถเปลี่ยนสัญลักษณ์นี้ได้โดยทำการพิมพ์ชื่อของสัญลักษณ์ที่ต้องการลงไป ซึ่งประเภทของสัญลักษณ์ที่สามารถนำมาใช้งานได้นั้น ได้แก่
    - ***Material Design*** หรือลักษณะการแสดงผลของสัญลักษณ์ทั่วไป (https://klarsys.github.io/angular-material-icons/)

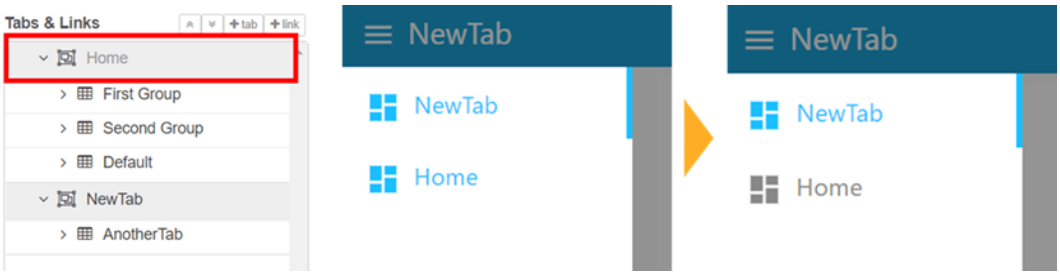


    - ***Font Awesome*** คือสัญลักษณ์ที่ถูกสร้างขึ้นโดยเว็บไซต์ Font Awesome (https://fontawesome.com/v4.7.0/icons/)

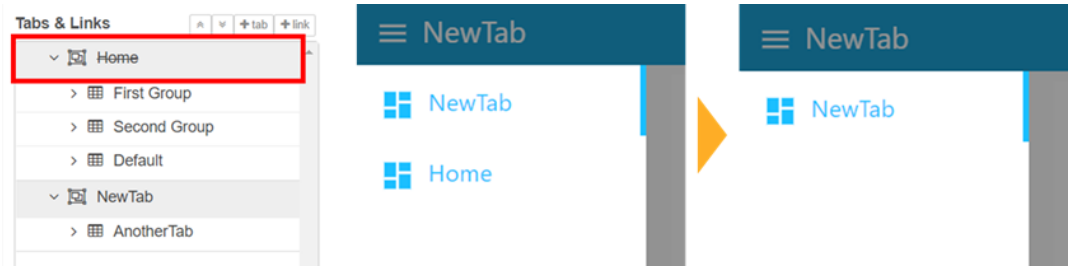


    - ***Weather*** หรือการแสดงผลสัญลักษณ์ที่เกี่ยวกับสภาพภูมิอากาศต่าง ๆ (https://github.com/Paul-Reed/weather-icons-lite/blob/master/css_mappings.md)

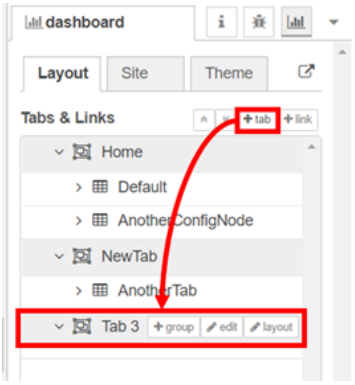
- **State** คือการปิดไม่ให้ผู้ใช้งานสามารถเข้าไปดูแดชบอร์ดนั้นได้หากตั้งค่าให้เป็น ‘Disabled’ (ชื่อของแถบแดชบอร์ดนั้นจะกลายเป็นสีเทาและไม่อนุญาตให้กด) และให้กลับมาเข้าถึงได้ตามปกติหากตั้งค่าเป็น ‘Enabled’



- **Nav. menu** คือการซ่อนแถบหน้าต่างนี้ไม่ให้แสดงผลขึ้นมาในแถบเมนูด้านข้างภายในแดชบอร์ดหากตั้งค่าเป็น ‘Hidden’ (ชื่อของแถบแดชบอร์ดนั้นจะถูกขีดฆ่าเมื่อตรวจสอบในแถบเมนู ‘Dashboard’ เช่น ~~Home~~ เป็นต้น) และตั้งค่าให้กลับมาแสดงผลดังเดิม หากตั้งค่ากลับมาให้เป็น ‘Visible’



การสร้างแถบหน้าต่างใหม่

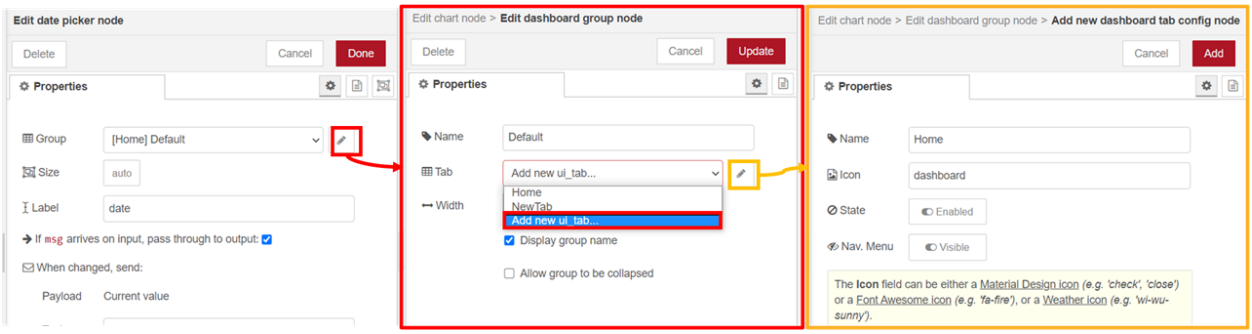


ในการสร้างแถบหน้าต่างใหม่ขึ้นมานั้น ให้ผู้ใช้งานทำการไปยังแถบเมนูแดชบอร์ด จากนั้นไปยังเมนู ‘Layout’ แล้วทำการคลิกที่ ‘**+ tab**’ เพื่อสร้างแถบหน้าต่างใหม่ขึ้นมา

วิดเจ็ทแต่ละวิดเจ็ทจำเป็นต้องมีการกำหนดแถบหน้าต่างที่ต้องการให้วิดเจ็ทนั้นไปอยู่ โดยวิดเจ็ทหนึ่งวิดเจ็ทสามารถอยู่ได้บนแถบแดชบอร์ดเดียวเท่านั้น ในขณะที่ภายในหนึ่งแดชบอร์ดสามารถมีวิดเจ็ทอยู่ภายในนั้นได้มากกว่า 1 วิดเจ็ท

การสร้างแถบหน้าต่างใหม่จากภายในโหนด

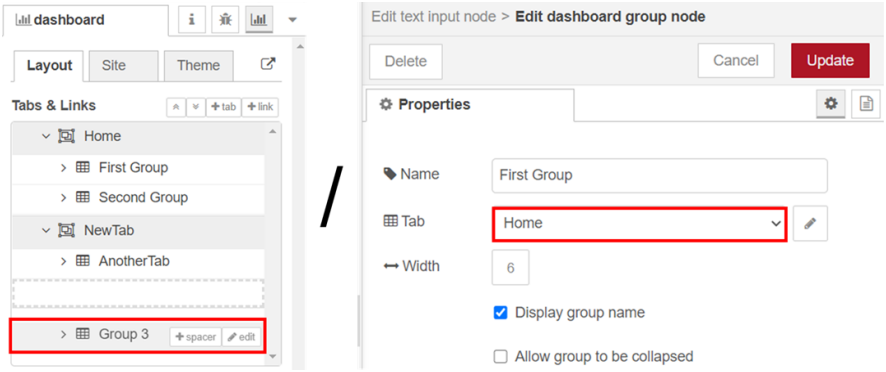
ผู้ใช้งานสามารถสร้างแถบแดชบอร์ดขึ้นมาใหม่ได้จากภายในโหนดโดยตรง โดยให้ทำการคลิกที่โหนดที่ต้องการขึ้นมา แล้วคลิกเข้าไปที่ปุ่มดินสอข้างแถบของ ‘Group’ จากนั้นทำการคลิกเลือกที่ ‘*Add new_ui tab...*’ ในแถบ ‘Tab’ แล้วทำการคลิกที่ปุ่มดินสอข้างแถบนั้นอีกทีเพื่อเปิดหน้าการตั้งค่าแถบแดชบอร์ดขึ้นมา โดยให้ทำการแก้ไขข้อมูลภายในนั้นให้เรียบร้อย จากนั้นจึงทำการกดที่ปุ่ม ‘Add’ เพื่อยืนยันการสร้างแถบแดชบอร์ดนั้นขึ้นมา



โหนดที่ใช้ในการสร้างแถบแดชบอร์ดนั้นขึ้นมาจะถูกจัดอยู่ในแดชบอร์ดนั้นในทันที

การย้ายกลุ่มของโหนด

ผู้ใช้งานสามารถเคลื่อนย้ายกลุ่มของฟังก์ชันการแสดงผลภายในแต่ละแถบแดชบอร์ดไปยังแถบหน้าต่างอื่น ๆ หรือเรียงลำดับการแสดงผลของกลุ่มได้โดยการคลิกเมาส์ค้างที่กลุ่มที่ต้องการเคลื่อนย้ายแล้วลากไปวางยังพื้นที่อื่นที่ต้องการ ซึ่งหน้าต่างปรากฏแถบเส้นประที่กลุ่มนั้นจะไปอยู่ขึ้นมา หรือเปิดหน้าการแก้ไขของกลุ่มของฟังก์ชันนั้นขึ้นมาแล้วทำการเลือกแถบหน้าต่างอื่นที่ต้องการให้กลุ่มของโหนดนี้ไปอยู่ผ่าน drop-down ในแถบ ‘Tab’

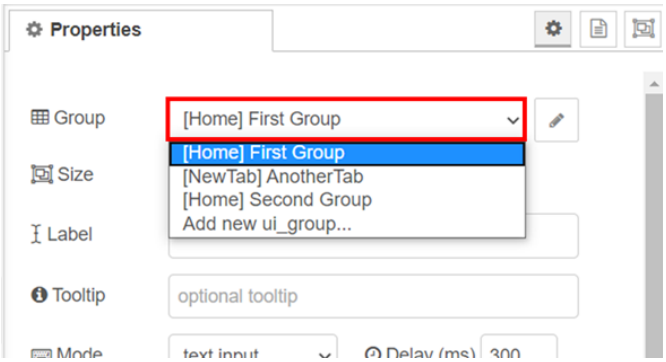


การตั้งค่าพื้นฐานภายในวิดเจ็ท

โหนดของวิดเจ็ทแต่ละโหนดนั้นมีการตั้งค่าที่แตกต่างกันซึ่งขึ้นอยู่กับลักษณะของการใช้งานวิดเจ็ทแต่ละประเภทนั้น ๆ โดยวิดเจ็ทส่วนใหญ่จะมีการตั้งค่าบางส่วนที่คล้ายคลึงกัน เช่น การจัดกลุ่มของวิดเจ็ท ขนาดของวิดเจ็ทต่าง ๆ เป็นต้น

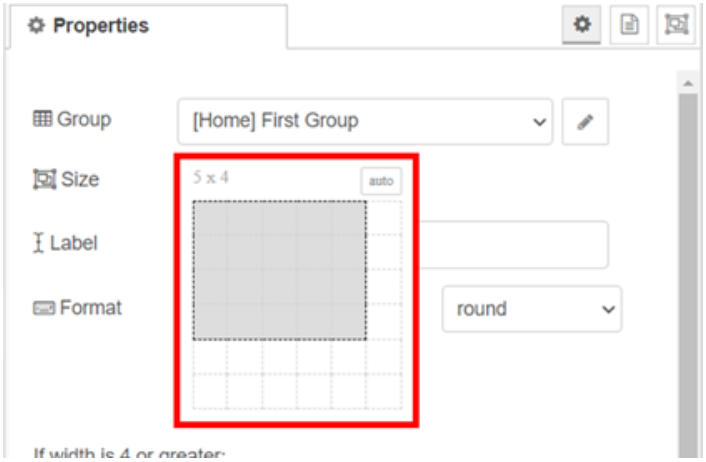
การจัดกลุ่มวิดเจ็ท

ผู้ใช้งานจำเป็นต้องกำหนดกลุ่มที่โหนดนั้นควรไปอยู่เพื่อให้โหนดนั้นสามารถใช้งานแสดงผลข้อมูลต่าง ๆ ออกมาได้ โดยให้ทำการคลิกที่ตัวโหนดเพื่อแสดงหน้าต่างตั้งค่าขึ้นมา แล้วเลือกกลุ่มที่ต้องการถูกสร้างเอาไว้แล้วก่อนหน้าใน drop-down ของ ‘Group’ การทำเช่นนี้จะเป็นการจัดกลุ่มของการแสดงผลต่าง ๆ ให้อยู่ด้วยกันตามความคล้ายคลึงกันของการใช้งานโหนดนั้น ๆ หรือจัดกลุ่มวิดเจ็ทเดียวกันให้อยู่ในหมวดหมู่เดียวกัน



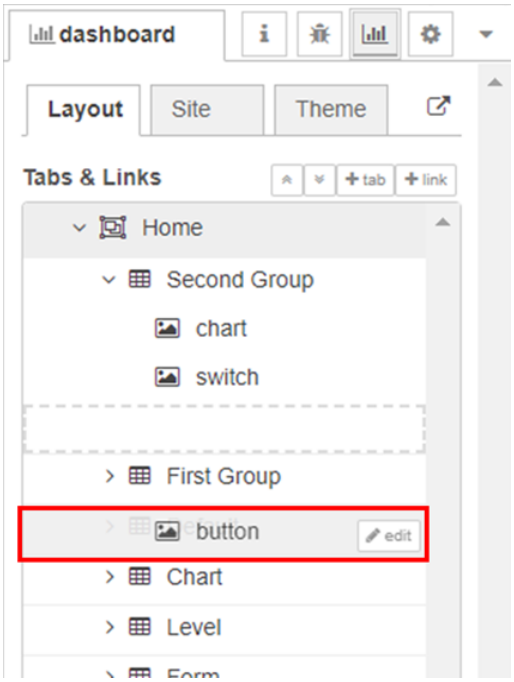
การตั้งค่าขนาดของวิดเจ็ท

ผู้ใช้งานสามารถกำหนดขนาดของการแสดงผลโหนดนั้นได้โดยการลากคลุมขนาดที่ต้องการให้แสดงผลใน ‘Size’ ซึ่งหากผู้ใช้งานไม่ได้ทำการตั้งค่าขนาดที่ต้องการให้แสดงผลในโหนดนี้ โหนดจะทำการตั้งขนาดของวิดเจ็ทนั้นเองโดยอัตโนมัติ โดยความกว้างสูงสุดของแต่ละโหนดถูกกำหนดไว้ที่ขนาดของกลุ่มที่วิดเจ็ทนั้นอาศัยอยู่



หากผู้ใช้งานทำการคลิกที่ปุ่ม ‘auto’ บนการตั้งค่าขนาดการแสดงผลของวิดเจ็ท หน้าต่างโปรแกรมจะทำการแสดงผลของวิดเจ็ทนั้นออกมาตามความเหมาะสมของวิดเจ็ทที่ถูกวางเอาไว้บนหน้าต่างแดชบอร์ด

การลำดับการแสดงผล



ในกรณีที่ผู้ใช้งานต้องการเรียงลำดับการแสดงผลของวิดเจ็ทที่อยู่ภายในแต่ละกลุ่มจากบนลงล่างใหม่ หรือเรียงลำดับการแสดงผลของแต่ละกลุ่มจากซ้ายไปขวา หรือแม้กระทั่งการเรียงลำดับการแสดงผลของแถบหน้าต่างแดชบอร์ดก็ตาม ผู้ใช้งานสามารถทำได้โดยการไปที่แถบเมนูข้างแดชบอร์ด แล้วทำการลากวิดเจ็ท กลุ่ม หรือแถบหน้าต่างที่ต้องการจะเรียงลำดับใหม่ขึ้นหรือลงไปอยู่ ณ ตำแหน่งที่ต้องการให้แสดงผล ซึ่งลำดับการแสดงผลของวิดเจ็ท กลุ่ม และแถบหน้าต่างนั้นจะขึ้นอยู่กับการเรียงลำดับในส่วนนี้จากบนลงล่างนั่นเอง

การลำดับการแสดงผลนั้นใช้หลักการเดียวกับการย้ายกลุ่มของวิดเจ็ท หรือย้ายที่อยู่ของกลุ่มภายในแต่ละแถบหน้าต่างให้อยู่ในพื้นที่ที่เหมาะสมต่อการใช้งาน ซึ่งการทำเช่นนี้จะทำให้การแสดงผลของหน้าแดชบอร์ดนั้นดูเป็นระเบียบมากขึ้น แม้จะมีจำนวนของวิดเจ็ทที่มากก็ตาม

ประเภทของวิดเจ็ท

ภายในโฟลว์ Flow Engine มีวิดเจ็ทให้สามารถใช้งานได้อยู่หลากหลายประเภท ทั้งประเภทที่รับค่าออกมาเพื่อนำไปแสดงผล หรือประเภทที่ส่งค่าข้อมูลออกไปให้ยังโหนดอื่นได้ใช้งานข้อมูลนั้นต่อไป

วิดเจ็ทที่ใช้ในการแสดงผล

การแสดงผลข้อมูลบนหน้าแดชบอร์ดนั้น Flow Engine มีวิดเจ็ทอยู่จำนวนไม่น้อยที่อนุญาตให้ผู้ใช้งานสามารถนำค่าข้อมูลมาแสดงผลในลักษณะต่าง ๆ ได้ ซึ่งการทำเช่นนี้จะช่วยเปลี่ยนจากค่าข้อมูลที่เข้ามาในลักษณะข้อความหรือ JSON ที่อ่านยาก ๆ ให้กลายเป็นภาพที่สามารถทำความเข้าใจได้ทันทีที่เห็น ซึ่งโหนดเหล่านี้ส่วนใหญ่มักจะไม่ได้ส่งค่าใดออกไปจากโหนดนี้อีก

หลอดไฟ

หลอดไฟ (Led) จะแสดงผลสถานะของค่าที่ได้รับเข้ามา โดยใช้สีในการควบคุมลักษณะของแต่ละสถานะของข้อมูลนั้น แต่หากข้อมูลที่เข้ามาไม่ตรงกับลักษณะที่ถูกกำหนดเอาไว้ของหลอดไฟ สีของหลอดไฟจะกลายเป็นสีเทาเพื่อแสดงว่าอยู่ในสถานะปิด

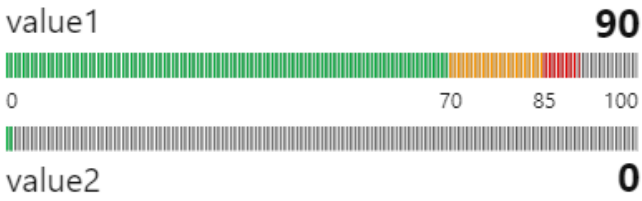
- **Label** คือคำอธิบายประกอบกับหลอดไฟที่แสดงผล
- **Label Placement** คือการเลือกว่าต้องการให้คำอธิบายประกอบหลอดไฟนั้นปรากฏที่ด้านขวาหรือด้านซ้ายของหลอดไฟ
- **Label Alignment** คือการเลือกตำแหน่งการแสดงผลของคำอธิบายประกอบหลอดไฟนั้นว่าต้องการให้ชิดซ้าย ชิดขวา หรืออยู่กึ่งกลาง
- **Colors for value of `msg.payload`** คือการเลือกสีประกอบค่าข้อมูล `msg.payload` ที่ถูกส่งเข้ามาภายในโหนดนี้
- **Allow Color For Value Map in msg** คือการอนุญาตให้ใช้สีที่ถูกส่งมาจาก msg จากโหนดก่อนหน้าในการใช้แสดงผลบนสวิตช์

เลเวล

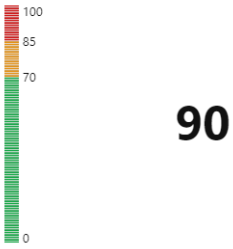
เลเวล (Level) เป็นเสมือนขีดระดับที่ใช้บอกถึงระดับของข้อมูลโดยอ้างอิงตามจำนวนขีดและสีที่ได้ทำการตั้งค่าเอาไว้ก่อนหน้า

- **Layout** คือการกำหนดลักษณะในการแสดงผลของเลเวล ซึ่งสามารถเลือกให้แสดงออกมาได้ 3 รูปแบบ ได้แก่
    - **Single Horizontal** คือการแสดงเลเวลของข้อมูลในลักษณะแถบขีดในแนวนอน ซึ่งใช้สีในการบอกถึงระดับปัจจุบันของค่าข้อมูลนั้น ซึ่งการแสดงผลในลักณะนี้มีการตั้งค่าเฉพาะตัวอยู่หนึ่งอย่าง ดังนี้

        - **Label** การขึ้นคำประกอบควบคู่กับค่าข้อมูล ซึ่งจะแสดงอยู่เหนือแถบเลเวลด้านหน้าตัวเลขของค่าข้อมูลในปัจจุบัน โดยจะแสดงข้อความตามที่ผู้ใช้งานได้ทำการตั้งค่าเอาไว้
    - **Pair Horizontal** คือการแสดงผลเลเวลของข้อมูลเป็นสองแถบ



        - **Channels** คือการใช้คำประกอบควบคู่กับค่าข้อมูลทั้งสองแถบ ซึ่งผู้ใช้งานสามารถเปลี่ยนคำที่ประกอบกับทั้งแถบเลเวลด้านบน (Top) และแถบเลเวลด้านล่าง (Bottom)
    - **Single Vertical** คือการแสดงผลในรูปแบบเดียวกับ Single Horizontal แต่จะแสดงผลของเลเวลออกมาในลักษณะของแนวตั้ง



        - **Label** การขึ้นคำประกอบควบคู่กับค่าข้อมูล ซึ่งจะแสดงอยู่ด้านข้างแถบ

เลเวลเหนือตัวเลขของค่าข้อมูลในปัจจุบัน โดยจะแสดงข้อความตามที่ผู้ใช้งานได้ทำการตั้งค่าเอาไว้

- **High** คือการเลือกสีในกรณีที่ค่าข้อมูลที่เข้ามาอยู่ในระดับที่สูงมาก ค่าเริ่มต้นคือ #e60000
- **Warm** คือการเลือกสีในกรณีที่ค่าข้อมูลที่เข้ามาอยู่ในระดับที่สูงกว่าปกติ ค่าเริ่มต้นคือ #ff9900
- **Normal** คือการเลือกสีในกรณีที่ค่าข้อมูลที่เข้ามาอยู่ในระดับปกติ ค่าเริ่มต้นคือ #00b33c
- **Off** คือการเลือกสีในช่วงของค่าข้อมูลที่ยังไปไม่ถึงหรือสีพื้นหลังของเลเวลนั้น ค่าเริ่มต้นคือ #595959
- **Show** คือลักษณะในการแสดงผลของสีที่จะปรากฏขึ้นบนแถบเลเวลโดยอ้างอิงตามระดับของค่าข้อมูลที่ได้รับเข้ามา
    - **Multiple segments** คือการแสดงผลระดับของข้อมูลภายในแถบเลเวลนั้นโดยแบ่งแต่ละช่วงข้อมูลออกเป็นสี ๆ ตามที่ผู้ใช้งานได้ทำการกำหนดเอาไว้

        - **Peak Mode** ในกรณีที่ค่าข้อมูลใหม่ที่เข้ามาอยู่ในระดับที่ต่ำกว่าค่าข้อมูลเดิมที่มีอยู่ ผู้ใช้งานสามารถตั้งค่าให้แสดงผลค่าสูงที่สุดของค่าข้อมูลก่อนหน้าเอาไว้ได้ด้วยการเปิดการใช้งาน Hold Peak เพื่อแสดงผลจุดของข้อมูลในค่าที่สูงที่สุดตามระยะเวลาที่กำหนดเอาไว้ในหน่วยมิลลิวินาที (ms) ซึ่งค่าเริ่มต้นในการแสดงผลค่านี้คือ 3000 มิลลิวินาทีหรือ 3 วินาที

    - **Single color bar** จะเปลี่ยนสีที่ปรากฏทั้งหมดให้กลายเป็นสีเดียวกันโดยอ้างอิงตามระดับของค่าข้อมูลที่ได้รับเข้ามา

    - **Interpolated colors** คือการไล่ระดับสีของข้อมูลขึ้นไปตามระดับของค่าข้อมูลที่ได้รับเข้ามา

- **Text Options** เป็นการตั้งค่าลักษณะการแสดงผลของข้อความบนขีดระดับ
    - **Default** คือการใช้ค่าเริ่มต้นที่กำหนดให้ของโปรแกรม
    - **Custom** จะเป็นการกำหนดลักษณะของข้อความที่ปรากฏด้วยตัวของผู้ใช้งานเอง ซึ่งมีส่วนในการตั้งค่าสองส่วนด้วยกัน
        - **Font Sizes** คือการกำหนดขนาดของคำหรือข้อความที่ปรากฏอยู่คู่กับแถบเลเวลนี้ ซึ่งมีอยู่ด้วยกันสามส่วนด้วยกัน
            - **Label** คือขนาดของคำที่ประกอบค่าข้อมูล
            - **Value** คือการกำหนดขนาดของค่าข้อมูลที่จะถูกนำมาแสดงผล
            - **Small** คือการคำหนดขนาดของค่าขอบบนและขอบล่างของแถบเลเวลที่ปรากฏอยู่ริมสุดทั้งสองฝั่งของแถบเลเวล
        - **Font Color** คือการกำหนดว่าต้องการใช้สีของคำหรือข้อความที่ปรากฏนั้นโดยอ้างอิงจากสีธีมหรือไม่ หากไม่ ผู้ใช้งานสามารถกำหนดสีของข้อความที่กำหนดได้ด้วยตัวเอง (Custom Color)
- **Range** คือการกำหนดค่าข้อมูลที่ต่ำที่สุดหรือขอบล่าง (Min) และค่าข้อมูลที่สูงที่สุดหรือขอบบน (Max) ของข้อมูลที่สามารถเป็นไปได้ในการแสดงผลออกมา
- **Sectors** คือการแบ่งช่วงของข้อมูลออกเป็นหลาย ๆ ส่วน (สูงสุด 4 ส่วน) เพื่อใช้ในการเปลี่ยนแปลงสีของค่าข้อมูลบนแถบเลเวลเมื่อค่าข้อมูลนั้นขยับไปถึงช่วงข้อมูลที่ได้ทำการกำหนดเอาไว้
- **Format** เป็นส่วนที่ใช้ในการตั้งค่าเกี่ยวกับวิธีการแสดงผลของข้อมูลบนแถบเลเวล ซึ่งประกอบไปด้วยการตั้งค่าในสองส่วนด้วยกัน
    - **Unit** คือหน่วยของข้อมูลที่ผู้ใช้งานสามารถกหนดเป็นข้อความใดก็ได้
    - **Decimals** คือจำนวนจุดทศนิยมของข้อมูลที่ผู้ใช้งานต้องการให้แสดงผลออกมา
- **Value Field** คือการกำหนดว่าต้องการให้ซ่อนค่าข้อมูล (Hide Value Field) ของแถบเลเวลนี้ไปหรือไม่
- **Stripe Shape** คือลักษณะของการซอยย่อยแถบเลเวลออกเป็นส่วน ๆ ว่าต้องการให้มีลักษณะอย่างไร
    - **Superfine** คือการแบ่งออกเป็นส่วนย่อยเล็ก ๆ ที่ละเอียดยิบ

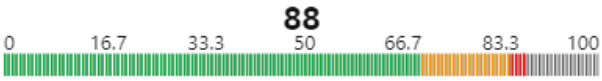
- **Fine** คือการแบ่งออกเป็นส่วนย่อยเล็ก ๆ แต่หยาบกว่า Superfine

- **Normal** คือการแบ่งออกเป็นส่วนย่อยที่หยาบกว่าสองรูปแบบก่อนหน้า

- **Tick Values** คือการกำหนดการแสดงผลของค่าที่ขึ้นประกอบบนแถบเลเวล (ไม่นับค่าที่เป็นขอบบนและขอบล่างของค่าข้อมูล)
  - **Off** คือปิดไม่ให้มีการแสดงผลค่าเหล่านั้นขึ้นมา

  - **Segments** จะแสดงผลเฉพาะค่าที่เป็นช่วงต่าง ๆ ของข้อมูลเท่านั้น

  - **Auto** จะทำการแบ่งช่วงข้อมูลที่จะแสดงผลออกมาเป็นช่วง ๆ ที่เท่ากัน โดยอ้างอิงตามความกว้างของวิดเจ็ทนี้



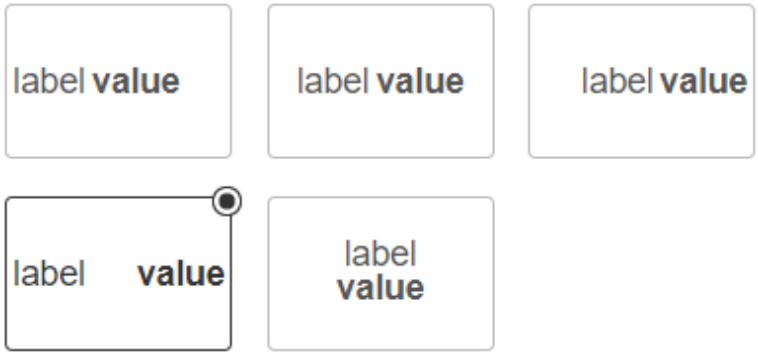
- **Animations** คือลักษณะการเคลื่อนที่ของแถบสีที่ปรากฏในแถบเลเวล
  - **Soft** จะเพิ่มและลดแถบสีอย่างนิ่มนวลและช้ากว่าทุกประเภทอื่น ๆ
  - **Reactive** จะเปลี่ยนแปลงแถบสีแบบแทบจะทันทีที่ข้อมูลภายในแถบเลเวลนี้มีการเปลี่ยนแปลงไป แต่ยังคงมีการเพิ่มลดที่นุ่มนวลคล้ายกับ Soft
  - **Rocket** มีการเปลี่ยนแปลงในลักษณะเดียวกับ Soft และ Reactive แต่เกิดขึ้นอย่างรวดเร็วมากกว่าทั้งสองประเภทจนแทบจะมองไม่เห็นถึงการเพิ่มหรือลดแถบสีในแถบเลเวล
  - **No animations** คือการปรากฏแถบสีโดยอ้างอิงตามค่าข้อมูลโดยตรง ซึ่งจะไม่มีการขยับเพิ่มหรือลดค่าแถบสีใด ๆ ทั้งสิ้น
- **Animated value** จะเป็นการกำหนดให้ค่าข้อมูลที่ปรากฏอยู่ควบคู่กับแถบเลเวลนี้มีการ

เคลื่อนไหวเปลี่ยนแปลงได้ทีละหน่วยอย่างรวดเร็วจนถึงค่าล่าสุดของข้อมูลนี้

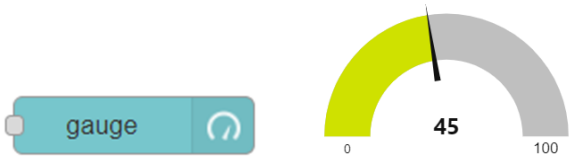
ข้อความ

ข้อความ (Text) จะปรากฏเป็นข้อความที่ไม่สามารถแก้ไขได้ขึ้นมา แต่สามารถเปลี่ยนแปลงได้หากมีข้อความส่งเข้ามาผ่านการตั้งค่าด้านหลังของโหนดนี้

- **Label** คือการแสดงคำขึ้นมาประกอบกับค่าข้อความที่ได้รับ โดยสามารถตั้งเป็นข้อความใหม่ หรือใช้ `{{msg.topic}}` เพื่อดึงค่า topic จากโหนดก่อนหน้าที่ส่งข้อมูลเข้ามานำมาแสดงผลก็ได้เช่นกัน
- **Value format** คือการกำหนดการแสดงผลของค่าข้อมูลที่ได้รับเข้ามาภายในโหนดนี้ ซึ่งสามารถใช้ได้ทั้งค่า HTML และ Angular filters เช่น `{{value | uppercase}} &deg;` จะทำการแปลงค่าข้อมูลนั้นให้กลายเป็นตัวพิมพ์ใหญ่ และเติมสัญลักษณ์องศา (o) ลงไปท้ายค่าข้อมูลนั้น เป็นต้น
- **Layout** คือการกำหนดลักษณะในการแสดงผลข้อมูลระหว่างคำประกอบ (Label) และค่าข้อมูล (Value) ว่าต้องการให้แสดงออกมาในลักษณะใด ซึ่งสามารถเลือกได้ 5 รูปแบบ

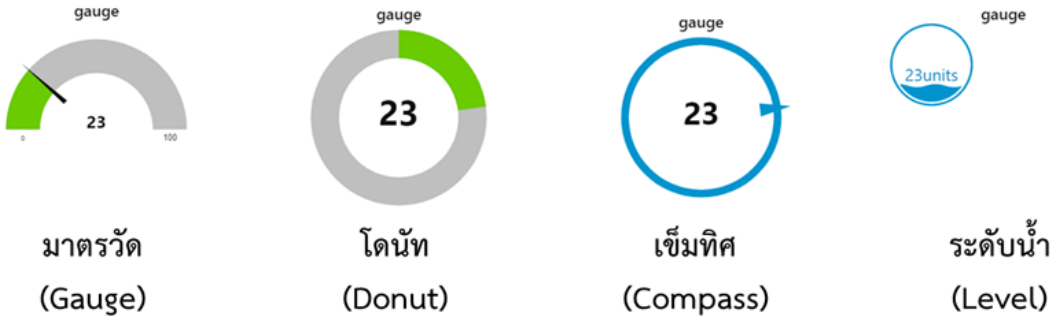


เกจ



เกจ (Gauge) จะปรากฏเป็นมาตรวัดที่จะแสดงผลของค่าข้อมูลที่ได้รับมาขึ้น โดยจะปรากฏ

แถบสีที่ข้างมาตรวัดเพื่อบ่งบอกถึงระดับของค่าข้อมูลนั้น ซึ่งมีประเภทในการแสดงผล 4 ประเภท ได้แก่ มาตรวัด (Gauge) โดนัท (Donut) เข็มทิศ (Compass) และระดับน้ำ (Level) โดยการแสดงผลทั้ง 4 ประเภทนี้มีส่วนการตั้งค่าหลัก ๆ ที่เหมือนกันอยู่ 4 อย่างด้วยกัน

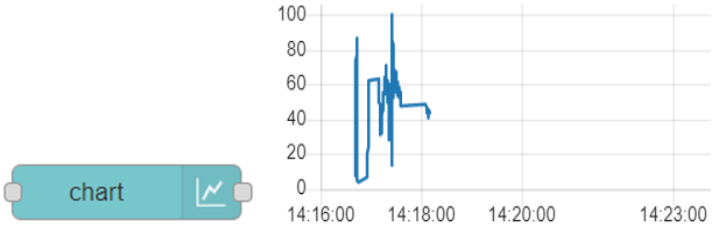


- **Label** คือชื่อของเกจการแสดงผลนั้น ๆ ซึ่งค่าเริ่มต้นจะใช้คำว่า ‘guage’
- **Unit** คือหน่วยของค่าข้อมูลที่ผู้ใช้งานสามารถกำหนดขึ้นมาได้ด้วยตัวเอง
- **Range** จะเป็นการกำหนดค่าขอบบนและขอบล่างของข้อมูล
    - **Min** ค่าต่ำที่สุดที่เป็นไปได้ในการแสดงผลข้อมูล เปรียบเสมือนขอบล่างของข้อมูล
    - **Max** ค่าสูงที่สุดที่เป็นไปได้ในการแสดงผลข้อมูล เปรียบเสมือนขอบบนของข้อมูล

อย่างไรก็ตาม วิดเจ็ทเกจแต่ละประเภทต่าง ๆ ก็มีส่วนการตั้งค่าเพิ่มเติมที่แตกต่างกันเพิ่มขึ้นมา อันเนื่องมาจากลักษณะการแสดงผลที่แตกต่างกัน ซึ่งส่วนการตั้งค่าที่เพิ่มเติมขึ้นมานั้นมีดังนี้

- **Value format** จะปรากฏให้ตั้งค่าในทุกประเภทยกเว้นระดับน้ำ คือการกำหนดลักษณะของค่าข้อมูลที่จะนำมาแสดงผล โดยอ้างอิงจากหลักการแสดงผลของ Angular Filters โดยค่าเริ่มต้นของการแสดงผลนี้จะแสดงออกมาในลักษณะ `{{value}}` หรือแสดงผลที่ได้รับเข้ามาจากค่าก่อนหน้าโดยตรง
- **Colour gradients** จะปรากฏให้ตั้งค่าเฉพาะในมาตรวัดและโดนัท โดยจะเป็นการไล่เฉดสีของข้อมูลโดยแบ่งข้อมูลออกเป็น 3 ส่วนเท่า ๆ กัน โดยให้ในแต่ละส่วนของข้อมูล (หรือแต่ละช่วงข้อมูล) ที่ถูกนำมาแสดงผลปรากฏเป็นเฉดสีที่ต่างกันไปตามระดับของข้อมูลนั้น
- **Sectors** จะปรากฏให้ตั้งค่าเฉพาะในมาตรวัดและโดนัท เป็นการแบ่งช่วงต่าง ๆ ระหว่างค่าต่ำที่สุดและค่าสูงที่สุดให้ออกจากกันเป็นหลาย ๆ ส่วน ซึ่งสามารถแบ่งได้สูงสุด 3 ส่วน เช่น หากค่าต่ำสุดและค่าสูงสุดของข้อมูลเป็น 100 และผู้ใช้งานกำหนดส่วนเป็น 25 และ 75 ค่าข้อมูลจะแบ่งออกเป็น 3 ช่วง ได้แก่ 0 - 25, 26 - 75, และ 76 - 100 นั่นเอง

แผนภูมิ



แผนภูมิ (Chart) ใช้ในการแสดงผลข้อมูลแต่ละจุดออกมาเป็นแผนภูมิในรูปแบบต่าง ๆ โดยสามารถเลือกให้อ้างอิงตามเวลา หรือแสดงผลในลักษณะแผนภูมิแท่ง หรือแผนภูมิวงกลมก็ได้เช่นกัน ซึ่งแต่ละประเภทของแผนภูมิต่าง ๆ มีส่วนการตั้งค่าที่ไม่เหมือนกัน แต่จะมีบางส่วนที่มีการตั้งค่าที่คล้ายคลึงกัน เช่น การตั้งชื่อของแผนภูมิใน Label และการตั้งข้อความเบื้องต้นก่อนที่จะมีค่าข้อมูลที่รับได้ส่งเข้ามาใน Blank Label

หากผู้ใช้งานต้องการแสดงผลข้อมูลหลายประเภทภายในแผนภูมิประเภทเดียวกัน ผู้ใช้งานสามารถทำได้โดยการตั้งค่า topic ของแต่ละค่าข้อมูลนั้นให้ต่างกัน ซึ่ง topic ที่ถูกส่งมาจากค่าข้อมูลนั้นจะกลายเป็นชื่อที่กำกับข้อมูลที่เข้ามาในทันที

ในขณะที่ผู้ใช้งานสามารถเปลี่ยนสีของแผนภูมิต่าง ๆ ไปตามที่กำหนดได้โดยให้ใช้สีที่ตั้งค่าไว้ใน Series Colours โดยแผนภูมิจะทำการเรียงลำดับการแสดงผลของแต่ละสีจากซ้ายไปขวาและบนลงล่างของสีที่ถูกกำหนดเอาไว้

แผนภูมิเส้น

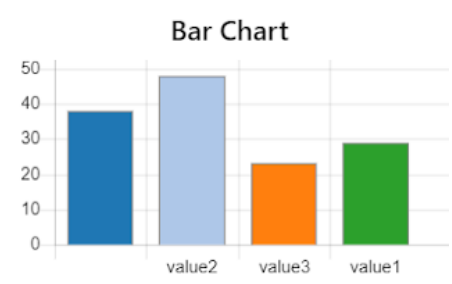
แผนภูมิเส้น (Line Chart) คือการแสดงผลข้อมูลต่าง ๆ ออกมาในลักษณะของเส้นจำนวนที่แปรผันตามเวลาที่ผ่านไปในแนวนอน

การตั้งค่าของแผนภูมิเส้นนั้นมีดังต่อไปนี้

- **Enlarge points** หากเปิดการใช้งานจะเป็นการปรับให้ปรากฏจุดข้อมูลขึ้นมาแสดงผลในแผนภูมิเพื่อแทนค่าข้อมูลแต่ละจุด
- **X-axis** คือการกำหนดจำนวนข้อมูลสูงสุดที่สามารถนำมาแสดงผลได้หากตรงเงื่อนไขใดเงื่อนไขหนึ่งระหว่างช่วงเวลาของข้อมูลที่ได้รับหรือจำนวนของจุดข้อมูล
    - **Last** คือการกำหนดช่วงเวลาย้อนหลังที่จะนำข้อมูลมาแสดงนับจากปัจจุบัน ซึ่งสามารถตั้งได้ตั้งแต่หน่วยวินาที นาที ชั่วโมง วัน และสัปดาห์ โดยค่าเริ่มต้นคือการแสดงข้อมูล 1 ชั่วโมงย้อนหลัง

    - **Points** คือการกำหนดจำนวนของจุดข้อมูลที่จำนะมาแสดงผล ซึ่งค่าเริ่มต้นคือ 1,000 จุด
- **X-axis Label** คือลักษณะของการแสดงผลเวลาในรูปแบบต่าง ๆ
- **As UTC** คือการเลือกให้แสดงผลเวลาโดยอ้างอิงตามเวลาสากลเชิงพิกัด (Coordinated Univarsal Time)
- **Y-axis** คือการกำหนดค่าสูงสุด (Max) และต่ำสุด (Min) ของข้อมูลในแกนแนวตั้ง
- **Legend** จะเป็นการเลือกว่าต้องการให้แสดงเส้นกำกับ (Show) ขึ้นมาด้านข้างของแผนภูมินี้หรือไม่ต้องการให้แสดงขึ้นมา (None)
- **Interpolate** เป็นการกำหนดลักษณะของแผนภูมิเส้นว่าต้องการให้มีการเชื่อมต่อกันระหว่างจุดข้อมูลแต่ละจุดให้ออกมาในรูปแบบใด ซึ่งมีให้เลือกตั้งแต่ Linear, Step, Bezier, Cubic, และ Cubic-mono

แผนภูมิแท่ง

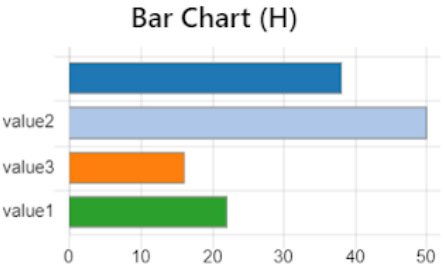


แผนภูมิแท่ง (Bar Chart) คือการแสดงผลข้อมูลในลักษณะของความสูงของค่าข้อมูล ซึ่งแต่ละแท่งจะหมายถึงค่าข้อมูลแต่ละค่า และความสูงคือค่าของข้อมูลในค่านั้น ๆ

การตั้งค่าของแผนภูมิแท่งนั้นมีดังต่อไปนี้

- **Y-axis** คือการกำหนดค่าสูงสุด (Max) และต่ำสุด (Min) ของข้อมูลในแกนแนวตั้ง
- **Legend** จะเป็นการเลือกว่าต้องการให้แสดงเส้นกำกับ (Show) ขึ้นมาด้านข้างของแผนภูมินี้หรือไม่ต้องการให้แสดงขึ้นมา (None)
- **Use first colour for all bars** หากเปิดการใช้งานจะเป็นการให้แผนภูมิแท่งทุกแท่งใช้สีเดียวกันกับแท่งแรกของแผนภูมิ

แผนภูมิแท่ง H

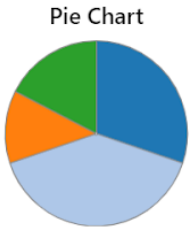


แผนภูมิแท่ง H (Bar Chart (H)) นั่นคือการแสดงผลแบบเดียวกับแผนภูมิแท่งปกติ แต่จะกลับข้างมาเป็นความยาวในแนวนอนแทนความสูง โดย H นั้นย่อมาจากคำว่า Horizontal ที่แปลว่าแนวนอนนั่นเอง

การตั้งค่าของแผนภูมิแท่ง H นั้นมีดังต่อไปนี้

- **X-axis** คือการกำหนดค่าสูงสุด (Max) และต่ำสุด (Min) ของข้อมูลในแกนแนวนอน
- **Legend** จะเป็นการเลือกว่าต้องการให้แสดงเส้นกำกับ (Show) ขึ้นมาด้านข้างของแผนภูมินี้หรือไม่ต้องการให้แสดงขึ้นมา (None)
- **Use first colour for all bars** หากเปิดการใช้งานจะเป็นการให้แผนภูมิแท่งทุกแท่งใช้สีเดียวกันกับแท่งแรกของแผนภูมิ

แผนภูมิวงกลม

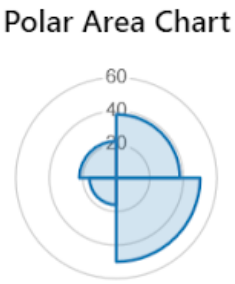


แผนภูมิวงกลม (Pie Chart) คือการแสดงผลข้อมูลทั้งหมดในลักษณะของสัดส่วนที่จะถูกแบ่งออกเป็นส่วนต่าง ๆ บนวงกลม คล้ายกับพิซซ่าหรือพายที่ถูกแบ่งออกเป็นหลาย ๆ ส่วน ซึ่งขนาดของแต่ละส่วนนั้นก็ขึ้นอยู่กับสัดส่วนของค่าข้อมูลนั้นเมื่อเทียบกับผลรวมของค่าข้อมูลทั้งหมดที่มี ยิ่งกว้างเท่าไหร่ก็ยิ่งหมายถึงขนาดของข้อมูลนั้นมีค่าที่สูงมากขึ้นตามไปด้วยนั่นเอง

การตั้งค่าของแผนภูมิวงกลมนั้นมีดังต่อไปนี้

- **Legend** จะเป็นการเลือกว่าต้องการให้แสดงเส้นกำกับ (Show) ขึ้นมาด้านข้างของแผนภูมินี้หรือไม่ต้องการให้แสดงขึ้นมา (None)
- **Cutout** คือการขยายรูตรงกลางของแผนภูมิให้กว้างออกโดยใช้อัตราร้อยละ (%) ในการช่วยขยาย ยิ่งใช้ค่าร้อยละที่สูงขึ้น รูตรงกลางก็ใหญ่ขึ้นตามไปด้วย

แผนภูมิขั้ว



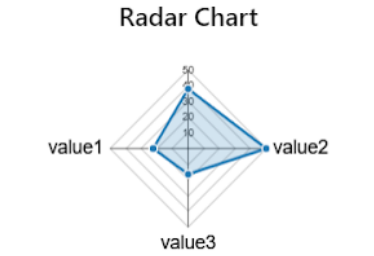
แผนภูมิขั้ว (Polar Area Chart) จะแสดงผลของข้อมูลในลักษณะก้นหอยที่จะเพิ่มปริมาณของข้อมูลขึ้นเรื่อย ๆ วนตามเข็มนาฬิกา

แผนภูมิประเภทนี้ถูกคิดค้นขึ้นในปี ค.ศ. 1855 โดยนางฟลอเรนซ์ ไนติงเกล (Florence Nightingale) นางพยาบาลชาวอังกฤษซึ่งเป็นผู้ก่อตั้ง Modern Nursing ขึ้นมา โดยเธอได้ทำการดัดแปลงแผนภูมิวงกลมที่มีอยู่แล้วดั้งเดิมมาใช้ในการวัดปริมาณของทหารที่เสียชีวิตในสงครามไครเมียในช่วงเมษายนปี 1854 ถึงเดือนมีนาคมปี 1855 จนทำให้ค้นพบว่าสาเหตุการตายส่วนใหญ่นั้นแท้จริงแล้วไม่ได้มาจากสงครามโดยตรง แต่มาจากโรคระบาดที่เกิดขึ้นในช่วงนั้นต่างหาก

การตั้งค่าของแผนภูมิขั้วนั้นมีดังต่อไปนี้

- **Y-axis** คือการกำหนดค่าสูงสุด (Max) และต่ำสุด (Min) ของข้อมูล ซึ่งจะปรากฏเป็นรัศมีที่แบ่งตามช่วงข้อมูลต่าง ๆ

แผนภูมิเรดาร์

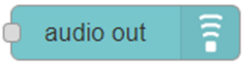


แผนภูมิเรดาร์ (Radar Chart) คือการแสดงผลข้อมูลโดยอ้างอิงจากการแสดงผลหน้าปัดของจอเรดาร์ที่ใช้ในการค้นหาสัญญาณบริเวณโดยรอบ โดยแผนภูมิประเภทนี้สามารถรับค่าตัวแปรหลายประเภทมาแสดงผลได้ภายในแผนภูมิเดียวกันในรูปแบบ 2 มิติ

การตั้งค่าของแผนภูมิเรดาร์นั้นมีดังต่อไปนี้

- **Y-axis** คือการกำหนดค่าสูงสุด (Max) และต่ำสุด (Min) ของข้อมูล ซึ่งจะปรากฏเป็นรัศมีที่แบ่งตามช่วงข้อมูลต่าง ๆ

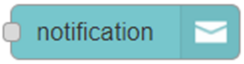
ส่งเสียงออก



ส่งเสียงออก (Audio Out) ใช้ในการเล่นเสียง หรือเปลี่ยนจากข้อความให้กลายเป็นเสียงพูดออกไป (Text to Speech; TTS) โดยรองรับการใช้งานไฟล์เสียงทั้ง .wav และ .mp3 อีกทั้งยังรองรับบทพูดหลากหลายภาษา ไม่ว่าจะเป็นฝรั่งเศส อิตาลี ญี่ปุ่น เกาหลี จีน หรือไต้หวัน เป็นต้น (แต่ปัจจุบันยังไม่รองรับการใช้งานภาษาไทย) โดยอ้างอิงจากเสียงของ Google และ Microsoft

ผู้ใช้งานสามารถเลือกให้มีการปรากฏเสียงขึ้นมาในกรณีที่หน้าต่างไม่ได้ถูกจับจ้องอยู่ได้ (Not in focus) โดยการคลิกเลือกที่ ‘*Play audio when window not in focus*’ เพื่อทำให้สามารถมีเสียงดังขึ้นได้แม้ผู้ใช้งานจะเปิดหน้าต่างอื่นอยู่ก็ตาม

การแจ้งเตือน



การแจ้งเตือน (Notification) จะปรากฏกล่องการแจ้งเตือนขึ้นมา หรือกล่องข้อความที่ให้ผู้ใช้งานเลือกระหว่างตกลง (OK) หรือยกเลิก (Cancel)

- **Layout** คือการกำหนดว่าต้องการให้ปรากฏกล่องการแจ้งเตือนขึ้นมาที่ตำแหน่งใดของหน้าต่างโปรแกรมหรือขึ้นมาในลักษณะใด
    - **Top Right** จะปรากฏกล่องการแจ้งเตือนที่มุมบนด้านขวาของหน้าต่าง
    - **Bottom Right** จะปรากฏกล่องการแจ้งเตือนขึ้นที่มุมล่างด้านขวาของหน้าต่าง
    - **Top Left** จะปรากฏกล่องการแจ้งเตือนที่มุมบนด้านซ้ายของหน้าต่าง
    - **Bottom Left** จะปรากฏกล่องการแจ้งเตือนที่มุมล่างด้านซ้ายของหน้าต่าง
    - **OK / Cancel Dialog** จะปรากฏเป็นกล่องข้อความที่อนุญาตให้ผู้ใช้งานสามารถกดปุ่ม ‘OK’ หรือ ‘Cancel’
    - **OK / Cancel Dialog with Input** จะปรากฏเป็นกล่องข้อความที่อนุญาตให้ผู้ใช้งานสามารถกดปุ่ม ‘OK’ หรือ ‘Cancel’ ที่อนุญาตให้ผู้ใช้งานสามารถใส่ข้อมูลบางอย่างลงไปได้
- ✔ **Default action label** ในกรณีที่ผู้ใช้งานเลือกให้แสดงผลการแจ้งเตือนแบบมีปุ่มกด OK หรือ Cancel ผู้ใช้งานสามารถกำหนดคำที่ต้องการใช้ให้ปรากฏบนปุ่มกดบนกล่องการแจ้งเตือนนั้นได้ โดยค่านี้จะมีค่าเริ่มต้นของปุ่มกดว่า ‘OK’ ซึ่งการกดปุ่มนี้จะเป็นการส่งค่า `msg.payload` ของโหนดนี้ออกไป โดยค่า payload ของโหนดนี้ที่จะส่งไปก็คือคำที่กำหนดเอาไว้บนปุ่มนี้นั่นเอง
- ✕ **Secondary action label** เช่นเดียวกับตั้งค่าในส่วน Default action label แต่การตั้งค่านี้ผู้ใช้งานจะใส่ค่าหรือไม่ใส่ลงไปดีได้ ซึ่งคำที่ถูกเขียนลงไปในส่วนการตั้งค่านี้จะหมายถึงการปฏิเสธค่าการแจ้งเตือนนั้น ซึ่งการกดปุ่มนี้จะเป็นการส่งค่า `msg.topic` ออกไปยังโหนดถัดไปแทน
- **Timeout (S)** คือการกำหนดเวลาในหน่วยวินาทีของการแสดงผลกล่องการแจ้งเตือนขึ้นมาบนหน้าต่างโปรแกรม
- **Border** จะเป็นการกำหนดสีขอบของกล่องการแจ้งเตือน
- **Topic** คือการตั้งชื่อหัวข้อให้กับข้อมูลที่ออกไปจากโหนดนี้ หรือเป็นข้อมูลขาออกเมื่อผู้ใช้งานทำการคลิกที่ Secondary action label

วิดเจ็ทสร้างค่าข้อมูล

นอกเหนือจากการนำค่าข้อมูลที่มีอยู่ขึ้นมาแสดงผลในหน้าแดชบอร์ดแล้ว ผู้ใช้งานยังสามารถสร้างข้อมูลใหม่จากภายในหน้าแดชบอร์ดโดยตรงได้อีกด้วย ซึ่งค่าข้อมูลที่สร้างมานั้นจะเป็น

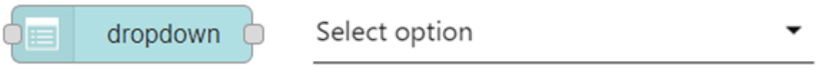
ค่าข้อมูลตามประเภทของวิดเจ็ทนั้น ๆ ที่นำมาใช้งาน

ปุ่มกด

ปุ่มกด (Button) ใช้ในการเพิ่มปุ่มกดเข้าไปในแดชบอร์ด โดยจะทำการสร้างข้อความขึ้นมาเมื่อผู้ใช้งานทำการคลิกลงไปที่ปุ่มกดนี้

- **Icon** คือสัญลักษณ์ของปุ่มกด ซึ่งผู้ใช้งานสามารถเปลี่ยนสัญลักษณ์นี้ได้โดยทำการพิมพ์ชื่อของสัญลักษณ์ที่ต้องการลงไป ซึ่งประเภทของสัญลักษณ์ที่สามารถนำมาใช้งานได้นั้นสามารถเปลี่ยนได้เช่นเดียวกับสัญลักษณ์ของหน้าแดชบอร์ด ได้แก่ Material Design, Font Awesome, และ Weather
- **Label** คือการกำหนดคำที่จะปรากฏอยู่บนปุ่มกด โดยค่าเริ่มต้นจะใช้คำว่า ‘button’ ซึ่งเมื่อปรากฏขึ้นในหน้าแดชบอร์ด คำที่เขียนเอาไว้จะถูกปรับให้กลายเป็นอักษรพิมพ์ใหญ่ทั้งหมด
- **Tooltip** จะปรากฏเป็นคำช่วยเหลือขึ้นเมื่อนำเมาส์ไปลอยเหนือปุ่มนี้
- **Colour** คือการกำหนดสีของข้อความหรือสัญลักษณ์ที่นำมาปรากฏบนปุ่มกดนี้
- **Background** จะเป็นการกำหนดสีพื้นหลังของปุ่มกด
- **When clicked, send:** จะเป็นการกำหนดว่าเมื่อผู้ใช้งานทำการคลิกที่ปุ่มกดนี้แล้ว จะให้ทำการส่งค่าใดออกไปอย่างไร ซึ่งประกอบไปด้วยค่าข้อมูล payload และ topic ที่ผู้ใช้งานจัดการในการส่งค่าเหล่านี้ได้ด้วยตัวเอง
- **If `msg` arrives on input, emulate a button click** คือการกำหนดว่า ให้ทุกครั้งที่มีข้อมูลผ่านเข้ามาในโหนดนี้นั้นทำงานเสมือนปุ่มนี้ถูกกดหรือไม่

รายการตัวเลือก



รายการตัวเลือก (Drop-down) จะอนุญาตให้ผู้ใช้งานสามารถเพิ่มตัวเลือกต่าง ๆ ของค่าข้อมูลไป โดยผู้ใช้งานสามารถเลือกค่าข้อมูลในตัวเลือกมากกว่า 1 ชนิดได้

- **Label** คือชื่อที่จะปรากฏขึ้นกำกับรายการตัวเลือกนี้เอาไว้
- **Tooltip** จะปรากฏเป็นคำช่วยเหลือขึ้นเมื่อนำเมาส์ไปลอยเหนือปุ่มนี้

- **Placeholder** คือการกำหนดคำเริ่มต้นที่จะปรากฏอยู่บนเมนู drop-down โดยค่าเริ่มต้นคือคำว่า ‘Select option’
- **Options** คือการกำหนดตัวเลือกของข้อค่าข้อมูลที่จะปรากฏใน drop-down ซึ่งผู้ใช้งานสามารถเพิ่มตัวเลือกเข้าไปได้โดยการคลิกที่ปุ่ม ‘**+Add**’ ด้านล่างของกล่องการตั้งค่านี้ ซึ่งเมื่อทำการคลิกเพื่อเพิ่มตัวเลือกแล้ว จะปรากฏเป็นตัวเลือกให้ตั้งค่าภายในได้ ซึ่งประกอบไปด้วยประเภทของค่าตัวเลือก ค่าตัวเลือก (Value) และคำประกอบค่าตัวเลือกนั้น (Label)
- **Allow multiple selection from list** จะให้ผู้ใช้งานเลือกว่าต้องการให้คนที่มาใช้งานวิดเจ็ทนี้สามารถเลือกตัวเลือกที่อยู่ภายใน drop-down หลาย ๆ ตัวเลือกได้หรือไม่ ซึ่งค่าเริ่มต้นของวิดเจ็ทนี้คือการอนุญาตให้สามารถทำได้ แต่หากปิดการใช้งานส่วนนี้ ผู้ใช้งานจะสามารถเลือกตัวเลือกภายใน drop-down ได้เพียงค่าเดียวเท่านั้น
- **If `msg` arrive on input, pass through output** หากเปิดการใช้งาน ข้อความที่ถูกส่งเข้ามาภายในโหนดจะถูกส่งไปเป็นข้อมูลขาออกของโหนดนี้ด้วยเช่นกัน
- **Topic** คือการตั้งชื่อหัวข้อให้กับค่าข้อมูลที่ถูกส่งออกไปจากวิดเจ็ทนี้

สวิตช์

สวิตช์ (Switch) จะทำการเพิ่มสวิตช์ซึ่งใช้ในการส่งสถานะเปิด (On) หรือปิด (Off) ซึ่งจะแสดงผลสลับกันไปทุกครั้งที่มีการคลิก

- **Label** คือการระบุคำให้ปรากฏคู่กับสวิช์นี้
- **Tooltip** จะปรากฏเป็นคำช่วยเหลือขึ้นเมื่อนำเมาส์ไปลอยเหนือสวิตช์
- **Icon** คือสัญลักษณ์ของสวิตช์ ซึ่งผู้ใช้งานสามารถเปลี่ยนสัญลักษณ์นี้ได้โดยทำการพิมพ์ชื่อของสัญลักษณ์ที่ต้องการลงไป ซึ่งประเภทของสัญลักษณ์ที่สามารถนำมาใช้งานได้นั้นสามารถเปลี่ยนได้เช่นเดียวกับสัญลักษณ์ของหน้าแดชบอร์ด ได้แก่ Material Design, Font Awesome, และ Weather
- **When clicked, send** คือค่าข้อมูลที่จะถูกส่งออกไปจากโหนดนี้ในกรณีที่มีการเปลี่ยนแปลงค่าของสวิตช์ พูดในอีกแง่หนึ่งคือ ข้อมูลจะถูกส่งไปหาผู้ใช้งานทำการเปิดหรือปิดสวิตช์นี้

- **On payload** คือค่าข้อมูลที่จะถูกส่งไปเมื่อสวิตช์ได้ทำการเปิดขึ้น
- **Off payload** คือค่าข้อมูลที่จะถูกส่งไปเมื่อสวิตช์ได้ถูกปิดลง
- **Topic** คือหัวข้อของข้อมูลที่สามารถใช้ส่งคู่ไปด้วยกันกับข้อมูลนั้น

แถบเลื่อนเส้นจำนวน

แถบเลื่อนเส้นจำนวน (Slider) คือแถบเลื่อนของค่าคำนวณที่ผู้ใช้งานสามารถกำหนดค่าต่ำสุดและสูงสุดของเส้นจำนวนได้ และใช้งานในการเลื่อนปุ่มบนเส้นจำนวนเพื่อเลือกค่าที่ต้องการ

- **Label** จะเป็นการระบุคำที่ต้องการให้ปรากฏขึ้นประกอบกับแถบเลื่อนเส้นจำนวนนี้
- **Tooltip** คือคำช่วยเหลือขนาดสั้นที่จะปรากฏขึ้นเมื่อผู้ใช้งานนำเอาเมาส์ไปลอยเหนือวิดเจ็ทนี้
- **Range** คือการกำหนดขอบบนและขอบล่างของแถบเลื่อนเส้นจำนวน
  - **Min** คือการกำหนดค่าที่ต่ำที่สุด (ขอบล่าง) ของแถบเลื่อนเส้นจำนวน ค่าเริ่มต้นของค่านี้คือ 0
  - **Max** คือการกำหนดค่าที่สูงที่สุด (ขอบบน) ของแถบเลื่อนเส้นจำนวน ค่าเริ่มต้นของค่านี้คือ 100
  - **Step** คือการกำหนดค่าการเปลี่ยนแปลงของแถบเลื่อนเส้นจำนวนว่าต้องการให้ขยับไปทีละเท่าไหร่ ซึ่งค่าเริ่มต้นของค่านี้คือ 1 ซึ่งหมายถึงจะมีการเปลี่ยนแปลงของค่าทีละ 1 เมื่อผู้ใช้งานทำการขยับแถบเลื่อนเส้นจำนวน
- **Output** คือการเลือกว่าต้องการให้ผลลัพธ์ที่ออกจากวิดเจ็ทนี้ออกไปในลักษณะใด
  - **Continuously while sliding** ให้แสดงผลต่อเนื่องในขณะที่ทำการเลื่อนค่าข้อมูล
  - **Only on release** ส่งค่าข้อมูลออกเฉพาะในกรณีที่ปล่อยเมาส์ออกจากแถบเลื่อนนี้แล้วเท่านั้น
- **When changed, send** จะเป็นการตั้งค่าข้อความที่จะถูกส่งออกไปจากวิดเจ็ทนี้เมื่อมีการเปลี่ยนแปลงของจำนวนในแถบเส้นจำนวน โดยจะทำการส่งค่า payload ออกไปโดยอ้างอิงตามค่าข้อมูลปัจจุบัน และค่า topic ที่ผู้ใช้งานสามารถระบุได้เอง

กล่องเปลี่ยนตัวเลข

กล่องเปลี่ยนตัวเลข (Numeric) จะปรากฏเป็นตัวกำหนดค่าข้อมูลตัวเลขภายในหน้าแดชบอร์ด ซึ่งผู้ใช้งานสามารถกำหนดค่าต่ำสุดและสูงสุดที่สามารถใส่ค่าได้ลงไปได้

- **Label** คือการกำหนดคำที่ต้องการให้ขึ้นประกอบกับกล่องเปลี่ยนตัวเลขนี้
- **Tooltip** คือการขึ้นแสดงข้อความช่วยเหลือขึ้นมาเมื่อนำเมาส์ไปลอยอยู่เหนือกล่องเปลี่ยนตัวเลขนี้
- **Value format** สามารถช่วยเปลี่ยนแปลงวิธีการแสดงผลข้อมูลได้ เช่น หากผู้ใช้งานพิมพ์ลงไปว่า `{{value}}%` หากค่าข้อมูลที่เข้ามาคือ 24 ข้อมูลจะแสดงออกมาว่า 24% ในขณะเดียวกัน ผู้ใช้งานยังสามารถใช้ค่า HTML หรือ Angular filters มาใส่ลงไปได้เช่นกัน เช่น `&deg;` จะแสดงสัญลักษณ์ขององศา (°) เป็นต้น ยิ่งไปกว่านั้น ผู้ใช้งานยังสามารถตั้งค่าให้กล่องเปลี่ยนตัวเลขนี้สามารถแก้ไขหรือใส่ค่าลงไปตรง ๆ ได้ด้วยการพิมพ์ว่า `{{msg.payload}}` ลงไปเพื่อรอรับค่าข้อมูลที่จะถูกกรอกลงไปโดยผู้ใช้งาน
- **Range** คือการกำหนดขอบบนและขอบล่างของตัวเลข
  - **Min** คือการกำหนดค่าที่ต่ำที่สุด (ขอบล่าง) ของตัวเลข ค่าเริ่มต้นของค่านี้คือ 0
  - **Max** คือการกำหนดค่าที่สูงที่สุด (ขอบบน) ของแถบตัวเลข ค่าเริ่มต้นของค่านี้คือ 100
  - **Step** คือการกำหนดว่าต้องการให้มีการเปลี่ยนแปลงค่าของตัวเลขขึ้นและลงทีละเท่าไหร่ ซึ่งค่าเริ่มต้นของค่านี้คือ 1 ซึ่งหมายถึงจะมีการเปลี่ยนแปลงค่าของตัวเลขขึ้นและลงทีละ 1 หน่วย เช่น จาก 36 เป็น 37 หรือ 35 เป็นต้น
- **Wrap value from max to min and min to max** คือการกำหนดว่าต้องการให้ค่าของตัวเลขนั้นสามารถวนจากค่าสูงที่สุดลงมาต่ำที่สุด หรือวนจากค่าต่ำที่สุดขึ้นไปค่าสูงที่สุดเมื่อเลือกเปลี่ยนค่าขึ้นหรือลงได้หรือไม่
- **When changed, send** จะเป็นการตั้งค่าข้อความที่จะถูกส่งออกไปจากวิดเจ็ทนี้เมื่อมีการเปลี่ยนแปลงของจำนวนในแถบกำหนดค่าตัวเลข โดยจะทำการส่งค่า payload ออกไปโดยอ้างอิงตามค่าข้อมูลปัจจุบัน และค่า topic ที่ผู้ใช้งานสามารถระบุได้เอง

ช่องกรอกข้อความ

ช่องกรอกข้อความ (Text Input) จะปรากฏเป็นเส้นตรงขนาดยาวที่ให้ผู้ใช้งานสามารถกรอกข้อความที่ต้องการลงไปได้ โดยสามารถเลือกได้ว่าต้องการให้ใส่เป็นข้อความธรรมดา อีเมล์ หรือตัวเลือกสีก็ได้เช่นกัน

- **Label** คือการกำหนดคำที่ต้องการให้ขึ้นประกอบกับช่องกรอกข้อความนี้
- **Tooltip** คือการขึ้นแสดงข้อความช่วยเหลือขึ้นมาเมื่อนำเมาส์ไปลอยอยู่เหนือช่องกรอกข้อความนี้
- **Mode** จะเป็นการกำหนดลักษณะของข้อความที่สามารถอนุญาตให้ใส่ได้ ซึ่งจะปรากฏในลักษณะของ drop-down ให้ผู้ใช้งานเลือกได้ว่าต้องการใช้ลักษณะใด โดยหากผู้ใช้งานพิมพ์ข้อความลงไปไม่ตรงกับประเภทที่กำหนดไว้ ค่านั้นจะขึ้นเป็นตัวอักษรสีแดงเพื่อแสดงถึงความผิดพลาดในประเภทของข้อมูลที่ถูกใส่ลงไป
    - **Text input** คือข้อความทั่วไปที่สามารถใส่ได้

    - **Email address** คือการกรอกข้อมูลในลักษณะของอีเมล์ xxxxxxxxxx@xxxxx.xxx ที่ผู้ใช้งานสามารถใช้ได้

    - **Password** จะเป็นการทำให้ข้อความที่ถูกใส่ลงไปจะถูกซ่อนเอาไว้ในลักษณะ ‘••••••••’ เช่นเดียวกับรหัสผ่าน (Password) ทั่ว ๆ ไป

    - **Number** จะกำหนดให้สามารถใส่ได้เพียงค่าข้อมูลตัวเลขเท่านั้น ไม่อนุญาตให้พิมพ์ข้อมูลในลักษณะของข้อความประเภทอื่น ๆ ลงไป โดยผู้ใช้งานสามารถเพิ่มหรือลดค่าจำนวนที่ใส่ลงไปโดยการกดที่ปุ่มลูกศรบนช่องกรอกข้อความนี้ได้ด้วยเช่นกัน

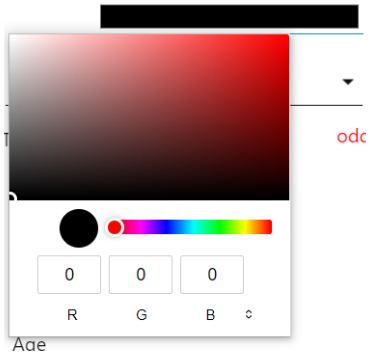
      

    - **Telephone input** คือการกำหนดลักษณะของข้อมูลให้อยู่ในรูปแบบของเบอร์โทรศัพท์

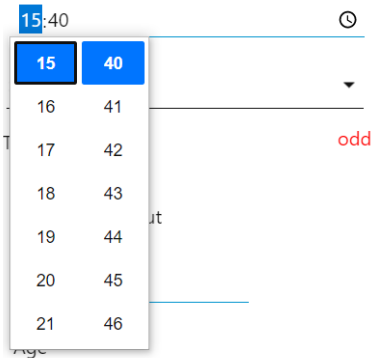
      

    - **Color picker** คือการกำหนดสีที่ต้องการนำมาแสดงผล ซึ่งจะปรากฏออกมาใน

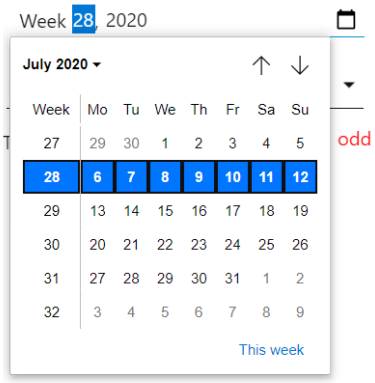
ลักษณะของแถบสีให้ผู้ใช้งานสามารถเลือกสีที่ต้องการได้



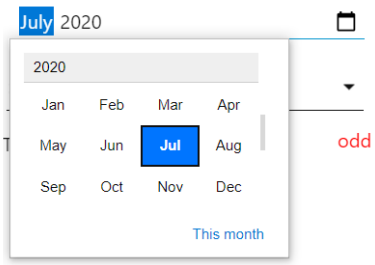
- **Time picker** คือการเลือกค่าเวลาในรูปแบบ ‘HH:mm’



- **Week picker** คือการเลือกช่วงสัปดาห์ในแต่ละปี ซึ่งผู้ใช้งานสามารถคลิกที่ ‘This week’ เพื่อเลือกเป็นสัปดาห์ปัจจุบันได้ โดยโปรแกรมจะทำการให้วันจันทร์เป็นวันตั้งต้นของสัปดาห์ และปิดท้ายสัปดาห์ด้วยวันอาทิตย์



- **Month picker** คือการเลือกเดือนในแต่ละปี ซึ่งผู้ใช้งานสามารถคลิกที่ ‘This month’ เพื่อเลือกเป็นเดือนปัจจุบันได้



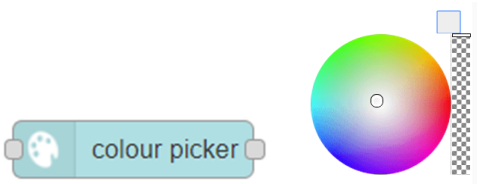
- **Delay (ms)** คือการตั้งเวลาในหน่วยมิลลิวินาทีก่อนที่ข้าข้อมูลจะถูกส่งออกจากโหนดนี้ไป โดยการตั้งไว้ที่ 0 จะหมายถึงจะรอจนกว่าปุ่ม `Enter` หรือ `Tab` ถูกกดจึงจะทำการส่งออกไป ซึ่งการกดปุ่มทั้งสองค่านั้นจะทำการส่งค่า payload ออกไป แต่ `Enter` จะยังคงที่อยู่ที่ค่าเดิมที่ใช้งานอยู่ ในขณะที่ `Tab` จะขยับเปลี่ยนไปยัง field อื่นแทน
- **When changed, send** คือการกำหนดว่าต้องการให้ส่งค่าข้อมูลออกไปจากโหนดนี้อย่างไรเมื่อเกิดการเปลี่ยนแปลงขึ้นภายในช่องกรอกข้อความ ซึ่งโดยปกติแล้วจะส่งค่าข้อมูลปัจจุบันออกไปใน payload โดยผู้ใช้งานสามารถแก้ไข topic ที่ต้องการใช้งานได้ด้วยตัวเอง

ตัวเลือกวันที่

ตัวเลือกวันที่ (Date Picker) จะปรากฏเป็นกล่องเพื่อให้ผู้ใช้งานสามารถกำหนดวันที่ที่ต้องการได้ โดยผู้ใช้งานสามารถเลือกใช้รูปแบบของการแสดงผลวันที่ได้ตามที่ต้องการ

- **Label** คือการขึ้นคำประกอบตัวเลือกวันที่นี้
- **If `msg` arrives on input, pass through output:** คือการเลือกว่าต้องการให้ส่งค่าข้อความ msg ที่ได้รับเข้ามาผ่านไปยังจุดเชื่อมต่อขาออกของโหนดนี้หรือไม่
- **When changed, send** จะเป็นการกำหนดว่า เมื่อมีการเปลี่ยนแปลงของค่าข้อมูลภายในตัวเลือกวันที่นี้แล้ว จะให้ส่งค่าข้อมูลใดออกไป ซึ่งค่าข้อมูล payload นั้นจะส่งค่าข้อมูลปัจจุบันไป และผู้ใช้งานสามารถกำหนดค่า ะopic ที่ต้องการส่งควบคู่กันไปได้เช่นกัน

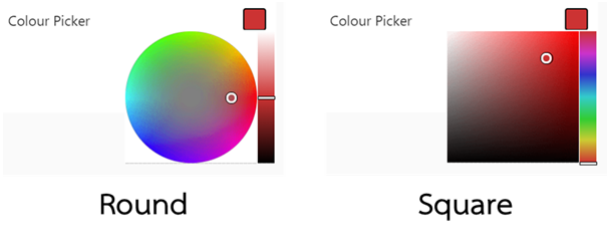
ตัวเลือกสี



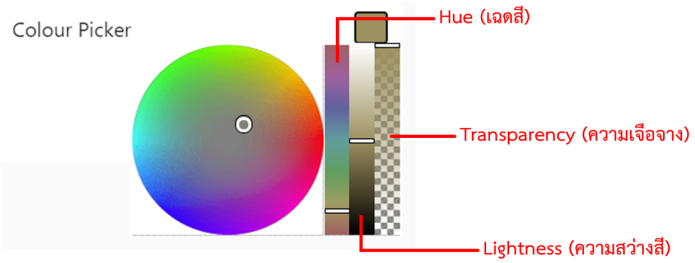
ตัวเลือกสี (Colour Picker) จะเพิ่มกล่องตัวเลือกสีเพื่อให้ผู้ใช้งานสามารถเลือกสีที่ต้องการ

- **Label** คือการขึ้นคำประกอบควบคู่กับวิดเจ็ทนี้
- **Format** คือการกำหนดลักษณะการแสดงผลของวิดเจ็ทนี้ และรูปแบบของการกำหนดค่าสีประเภทต่าง ๆ
    - **Colour Type** คือการกำหนดประเภทของค่าสีที่จะใช้ในการเลือก ซึ่งจะส่งผลออกเป็นค่าข้อมูลที่ถูกส่งออกไปเมื่อผู้ใช้งานเลือกสีต่าง ๆ ในวิดเจ็ทนี้
        - **HEX** คือการกำหนดค่าสีให้อยู่ในลักษณะของเลขฐาน 16 ซึ่งจะส่งค่าข้อมูลในเลขฐาน 16 ออกไป เช่น `a08f5d` เป็นต้น
        - **HEX8** คือการกำหนดค่าสีให้อยู่ในลักษณะของเลขฐาน 16 เช่นเดียวกับค่า HEX ธรรมดา แต่จะพ่วงท้ายไปด้วยความเจือจางของสี (Transparency) ที่จะห้อยเป็นเลขฐาน 16 อยู่ท้ายสุดของค่าข้อมูล (เลขสองตัวสุดท้ายของค่าข้อมูลฐาน 16) เช่น `c44239ff` เป็นต้น โดยค่า `ff` หมายถึงไม่มีความเจือจางใด ๆ ในขณะที่ `00` คือเจือจางจนไม่เห็นสีนั้นอีกต่อไป
        - **HSL** จะเป็นการลงสีโดยอ้างอิงจากค่าเฉดสี (Hue) ปริมาณความเข้มของสี (Saturation) และปริมาณความสว่างของสี (Lightness) ซึ่งค่าข้อมูลจะออกไปในลักษณะ `hsl(Hue, Saturation, Lightness)` เช่น `hsl(57, 59%, 50%)` เป็นต้น ซึ่งค่าเฉดสีที่ออกมานั้นจะอยู่ระหว่าง 0 ถึง 255 ในขณะที่ความเข้มของสีหรือความสว่างของสีจะเป็นค่าร้อยละ
        - **HSV** จะเป็นการลงสีโดยอ้างอิงจากค่าเฉดสี (Hue) ปริมาณความเข้มของสี (Saturation) และปริมาณความสว่างของสี (Value) ซึ่งค่าข้อมูลจะออกไปในลักษณะ `hsv(Hue, Saturation, Value)` เช่น `hsv(110, 55%, 69%)` เป็นต้น
        - **RGB** คือการลงสีโดยอ้างอิงตามความเข้มข้นของค่าสีแดง เขียว และฟ้า ซึ่งจะเป็นการกำหนดค่าข้อมูลของสีที่ผู้ใช้งานเลือกให้ออกไปในลักษณะ `rgb(Red, Green, Blue)` เช่น `rgb(158, 145, 97)` เป็นต้น

- **Colour Picker Shape** จะเป็นการเลือกลักษณะของตัวเลือกสีว่าต้องการให้แสดงผลออกมาในลักษณะวงกลม (Round) หรือเป็นสี่เหลี่ยม (Square)

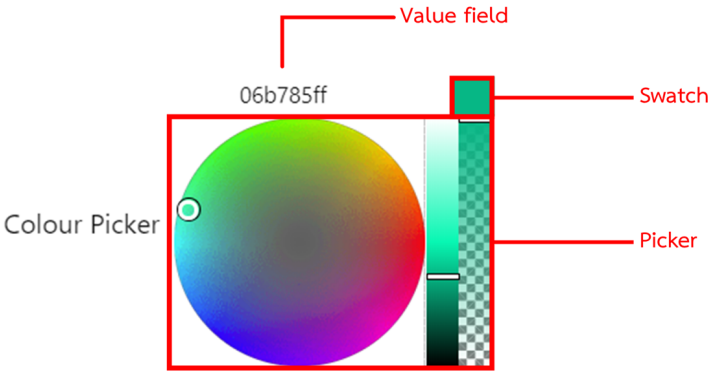


- **Show slider** เป็นตัวเลือกว่าต้องการให้แสดงผลแถบเลื่อนใดบ้างระหว่าง



  - **Show hue slider** จะเป็นการเลือกว่าต้องการให้ปรากฏแถบเฉดสีขึ้นมาหรือไม่
  - **Show lightness slider** จะเป็นการเลือกว่าต้องการให้ปรากฏแถบความสว่างของเฉดสีขึ้นมาหรือไม่
  - **Show transparency slider** จะเป็นการเลือกว่าต้องการให้ปรากฏแถบความเจือจางจนเห็นพื้นหลังของสีขึ้นมาหรือไม่ อย่างไรก็ดี แถบนี้จะไม่ปรากฏขึ้นมาในกรณีที่ผู้ใช้งานเลือกให้ค่าของสีแบบ HEX

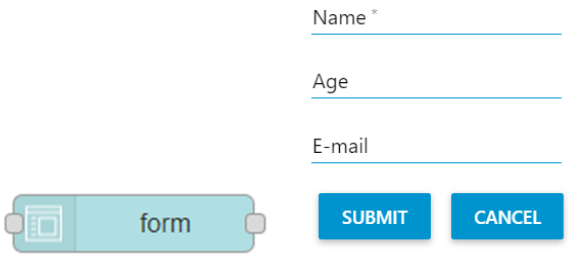
- **If width is 4 or greater** จะเป็นการเลือกตั้งค่าการแสดงผลของวิดเจ็ทในกรณีที่ขนาดของวิดเจ็ทนี้มีความกว้างตั้งแต่ 4 ขึ้นไปว่าต้องการให้แสดงผลแบบใด ซึ่งผู้ใช้งานจำเป็นต้องเลือกให้แสดงผลในรูปแบบใดรูปแบบหนึ่งเสมอ



- **Always show swatch** คือให้แสดงกล่องสีที่ถูกเลือกเอาไว้ขึ้นมาตลอดเวลา
- **Always show picker** คือให้แสดงตัวเลือกสีขึ้นมาตลอดเวลา หากปิดการใช้งานค่านี้ การเลือกสีจะถูกซ่อนเอาไว้และจะแสดงขึ้นมาเมื่อคลิกที่ปุ่มหรือช่องแสดงค่าข้อมูลเท่านั้น
- **Always show value field** จะเป็นการแสดงกล่องของค่าข้อมูลนั้นขึ้นมาด้านบนของตัวเลือกสี

- **Send** คือการกำหนดว่าต้องการให้ส่งค่าข้อมูลออกไปอย่างไร
  - **One value when released/closed** ส่งเพียงค่าข้อมูลค่าเดียวเมื่อทำการปล่อยเมาส์หรือเมื่อปิดหน้าเลือกสีให้หายไป
  - **Multiple values during editing** คือการอนุญาตให้สามารถส่งหลาย ๆ ค่าได้ระหว่างการแก้ไขหรือเปลี่ยนแปลง
- **Payload** คือการกำหนดประเภทของค่าข้อมูลที่ต้องการส่งออกไปจากโหนดนี้ ซึ่งมีให้เลือกอยู่ด้วยกัน 2 ประเภท คือการส่งค่าปัจจุบันในลักษณะของข้อความ (Current value as a string) และส่งค่าปัจจุบันออกไปในลักษณะของอ็อบเจกต์ (Current value as object)
- **Topic** คือการส่งค่าข้อความที่ต้องการออกไปคู่กับค่าข้อมูล payload ที่ส่งไป

ฟอร์ม



ฟอร์ม (Form) คือแบบสอบถามที่จะปรากฏเป็นกล่องที่ใช้ในการรับข้อมูลมากกว่า 1 อย่างเข้ามาได้ภายในเวลาเดียวกัน โดยจะทำการรวบรวมเป็นค่าข้อมูล 1 ค่าเมื่อผู้ใช้งานที่กรอกข้อมูลเข้ามาทำการกดที่ปุ่ม **‘Submit’**

- **Label** จะปรากฏเป็นคำประกอบควบคู่กับแบบฟอร์มที่ผู้ใช้งานสร้างขึ้น
- **Form elements** คือส่วนที่ใช้ในการประกอบร่างแบบสอบถามขึ้นมา ซึ่งผู้ใช้งานสามารถเพิ่มค่าคำถามต่าง ๆ ที่ต้องการได้โดยการคลิกที่ ‘+ Element’ เพื่อเพิ่มคำถามใหม่ ๆ เข้า

มา ซึ่งแต่ละคำถามนั้นจะมีส่วนการตั้งค่าต่อไปนี้

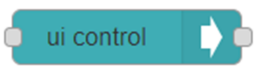
- **Label** คือคำที่จะขึ้นมาอยู่ในคำถามของแบบสอบถามนั้น
- **Name** คือชื่อของแบบสอบถามข้อนั้น ซึ่งจะทำหน้าที่เป็นค่า key หรือตัวแปรของค่าข้อมูลนั้นที่จะถูกส่งออกจากโหนดไปด้วย payload
- **Type** คือประเภทของค่าข้อมูลที่คำถามนั้นสามารถรับได้ มีหลายประเภทด้วยกัน ได้แก่ ข้อความทั่วไป (Text) ข้อความหลายบรรทัด (Multiline) ตัวเลข (Number) อีเมล์ (E-mail) รหัสผ่าน (Password) กล่องตัวเลือก (Checkbox) สวิตช์ (Switch) และวันที่ (Date)
- **Required** เป็นการกำหนดว่าต้องการบังคับให้คำถามนี้ถูกตอบหรือไม่ โดยจะปรากฏขึ้นเป็น * อยู่ที่คำถามนั้น
- **Rows** จะปรากฏให้ตั้งค่าเมื่อผู้ใช้งานเลือกใช้แบบสอบถามประเภทข้อความหลายบรรทัด (Multiline) ซึ่งจะทำหน้าที่เป็นตัวกำหนดจำนวนบรรทัดที่สามารถพิมพ์ลงไปได้
- **Remove** คือการลบคำถามนี้ออกจากแบบสอบถาม

- **Buttons** คือการตั้งชื่อให้กับปุ่มกดทั้งสองปุ่มท้ายฟอร์ม
  - 👍 คือการตั้งคำที่จะให้ปรากฏในปุ่มทางซ้ายหรือปุ่มยืนยัน ซึ่งค่าเริ่มต้นคือ *‘Submit’*
  - 👎 คือการตั้งคำที่จะให้ปรากฏในปุ่มทางขวาหรือปุ่มปฏิเสธ ซึ่งค่าเริ่มต้นคือ *‘Cancel’* ซึ่งผู้ใช้งานสามารถซ่อนปุ่มนี้ได้โดยการไม่ใส่ค่าใด ๆ ลงไป
- **Topic** คือการกำหนดค่าข้อความที่จะส่งไปคู่กับค่าข้อมูลของวิดเจ็ทนี้

วิดเจ็ทอื่น ๆ

นอกเหนือจากวิดเจ็ทสองประเภทข้างต้น ยังมีวิดเจ็ทนอกเหนือจากนั้นที่สามารถนำไปใช้งานให้แดชบอร์ดมีประสิทธิภาพมากขึ้นกว่าเดิมได้อีกสองชนิด ได้แก่

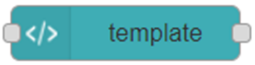
ตัวควบคุมแดชบอร์ด



ตัวควบคุมแดชบอร์ด (UI Control) ใช้ในการควบคุมเกี่ยวกับเหตุการณ์ที่เกิดขึ้นภายในหน้า

แดชบอร์ด โดยค่าเริ่มต้นของโหนดนี้คือการเปลี่ยนแปลงการแสดงผลของแถบหน้าแดชบอร์ดที่ต้องการโดยอ้างอิงจาก `msg.payload` ที่มีตัวแปร `{“tab”: “tab_name”}` ที่ถูกส่งมาจากโหนดก่อนหน้า ซึ่งค่าของตัวแปรที่ใส่มานี้ควรเป็นชื่อของแถบหน้าแดชบอร์ดที่ต้องการให้แสดงค่า ตัวเลขที่ระบุตำแหน่งของแถบหน้าแดชบอร์ด (เริ่มที่ 0) หรือ link ที่ต้องการนำมาแสดงผล

เท็มเพลต



เท็มเพลต (Template) อนุญาตให้ผู้ใช้งานสามารถใส่ค่า HTML หรือ Angular ลงไปได้ โดยจะเป็นการสร้างการแสดงผลที่จะเปลี่ยนแปลงในทันทีตามข้อความที่เข้ามาภายในโหนด เช่น เปลี่ยนค่าการแสดงว่าผลว่าข้อมูลที่ได้รับเข้ามานั้นเป็นจำนวนคี่หรือจำนวนคู่ เป็นต้น

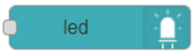
```
<div layout="row" layout-align="space-between">
  <p>The number is</p>
  <font color="{{((msg.payload || 0) % 2 === 0) ? 'green' : 'red'}}">
    {{(msg.payload || 0) % 2 === 0 ? 'even' : 'odd'}}
  </font>
</div>
```

ประเภทของวิดเจ็ท

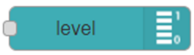
ภายในโฟลว์ Flow Engine มีวิดเจ็ทให้สามารถใช้งานได้อยู่หลากหลายประเภท ทั้งประเภทที่รับค่าออกมาเพื่อนำไปแสดงผล หรือประเภทที่ส่งค่าข้อมูลออกไปให้ยังโหนดอื่นได้ใช้งานข้อมูลนั้นต่อไป

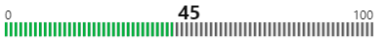
วิดเจ็ทที่ใช้ในการแสดงผล

การแสดงผลข้อมูลบนหน้าแดชบอร์ดนั้น Flow Engine มีวิดเจ็ทอยู่จำนวนไม่น้อยที่อนุญาตให้ผู้ใช้งานสามารถนำค่าข้อมูลมาแสดงผลในลักษณะต่าง ๆ ได้ ซึ่งการทำเช่นนี้จะช่วยเปลี่ยนจากค่าข้อมูลที่เข้ามาในลักษณะข้อความหรือ JSON ที่อ่านยาก ๆ ให้กลายเป็นภาพที่สามารถทำความเข้าใจได้ทันทีที่เห็น ซึ่งโหนดเหล่านี้ส่วนใหญ่มักจะไม่ได้ส่งค่าใดออกไปจากโหนดนี้อีก

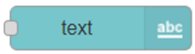


หลอดไฟ (Led) จะแสดงผลสถานะของค่าที่ได้รับเข้ามา โดยใช้สีในการควบคุมลักษณะของแต่ละสถานะของข้อมูลนั้น

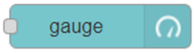


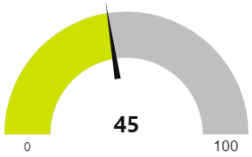


เลเวล (Level) ใช้บอกถึงระดับของข้อมูลโดยอ้างอิงตามสีที่ได้ทำการตั้งค่าเอาไว้ก่อนหน้า

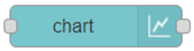


ข้อความ (Text) จะปรากฏเป็นข้อความที่ไม่สามารถแก้ไขได้ขึ้นมา แต่สามารถเปลี่ยนแปลงได้หากมีข้อความส่งเข้ามาผ่านการตั้งค่าด้านหลังของกล่องฟังก์ชั่นนี้

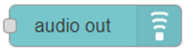




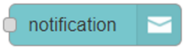
มาตรวัด (Gauge) จะปรากฏเป็นมาตรวัดที่จะแสดงผลของค่าข้อมูลที่ได้รับมาขึ้น โดยจะปรากฏแถบสีที่ข้างมาตรวัดเพื่อบ่งบอกถึงระดับของค่าข้อมูลนั้น



กราฟ (Chart) ใช้ในการแสดงผลข้อมูลแต่ละจุดออกมาเป็นกราฟโดยสามารถเลือกให้อ้างอิงตามเวลา หรือแสดงผลในบักษณะแผนภูมิแท่ง หรือแผนภูมิวงกลมก็ได้เช่นกัน



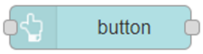
ส่งเสียงออก (Audio Out) ใช้ในการเล่นเสียง หรือเปลี่ยนจากข้อความให้กลายเป็นเสียงพูดออกไป (Text to Speech) โดยรองรับการใช้งานไฟล์เสียงทั้ง .wav และ .mp3



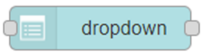
การแจ้งเตือน (Notification) จะปรากฏกล่องการแจ้งเตือนขึ้นมา หรือกล่องข้อความที่ให้ผู้ใช้งานเลือกระหว่างตกลง (OK) หรือยกเลิก (Cancel)

วิดเจ็ทสร้างค่าข้อมูล

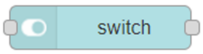
นอกเหนือจากการนำค่าข้อมูลที่มีอยู่ขึ้นมาแสดงผลในหน้าแดชบอร์ดแล้ว ผู้ใช้งานยังสามารถสร้างข้อมูลใหม่จากภายในหน้าแดชบอร์ดโดยตรงได้อีกด้วย ซึ่งค่าข้อมูลที่สร้างมานั้นจะเป็นค่าข้อมูลตามประเภทของวิดเจ็ทนั้น ๆ ที่นำมาใช้งาน



ปุ่มกด (Button) ใช้ในการเพิ่มปุ่มกดเข้าไปในหน้าต่างแสดงผล โดยจะทำการสร้างข้อความขึ้นมาเมื่อผู้ใช้งานทำการคลิกลงไปที่ปุ่มกดนี้

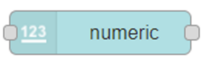


รายการตัวเลือก (Drop-down) จะอนุญาตให้ผู้ใช้งานสามารถเพิ่มตัวเลือกต่าง ๆ ของค่าข้อมูลไป โดยผู้ใช้งานสามารถเลือกค่าข้อมูลในตัวเลือกมากกว่า 1 ชนิดได้



สวิตช์ (Switch) จะทำการเพิ่มสวิตช์ซึ่งใช้ในการส่งสถานะเปิด (On) หรือปิด (Off) ซึ่งจะแสดงผลสลับกันไปทุกครั้งที่มีการคลิก

เส้นจำนวน (Slider) คือเส้นของค่าคำนวณที่ผู้ใช้งานสามารถกำหนดค่าต่ำสุดและสูงสุดของเส้นจำนวนได้ และใช้งานในการเลื่อนปุ่มบนเส้นจำนวนเพื่อเลือกค่าที่ต้องการ



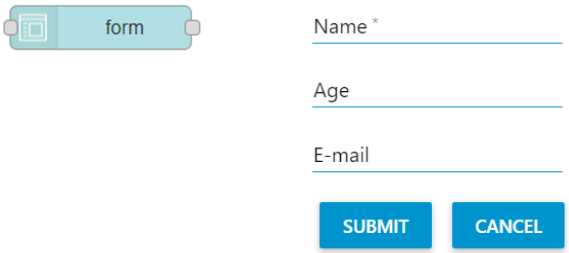
ตัวเลข (Numeric) จะปรากฏเป็นกล่องให้ใส่ค่า

ข้อมูลตัวเลขเข้าไปในหน้าต่างแสดงผล ซึ่งผู้ใช้งานสามารถกำหนดค่าต่ำสุดและสูงสุดที่สามารถใส่ค่าได้ลงไปได้

ช่องกรอกข้อความ (Text Input) จะปรากฏเป็นกล่องเพื่อให้ผู้ใช้งานสามารถกรอกข้อความที่ต้องการลงไปได้ โดยสามารถเลือกได้ว่าต้องการให้ใส่เป็นข้อความธรรมดา อีเมล์ หรือตัวเลือกสีก็ได้เช่นกัน

ตัวเลือกวันที่ (Date Picker) จะปรากฏเป็นกล่องเพื่อให้ผู้ใช้งานสามารถกำหนดวันที่ที่ต้องการได้ โดยผู้ใช้งานสามารถเลือกใช้รูปแบบของการแสดงผลวันที่ได้ตามที่ต้องการ

ตัวเลือกสี (Colour Picker) จะเพิ่มกล่องตัวเลือกสีเพื่อให้ผู้ใช้งานสามารถเลือกสีที่ต้องการ

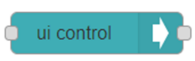


แบบสอบถาม (Form) จะปรากฏเป็นกล่องที่ใช้ในการรับข้อมูลมากกว่า 1 อย่างเข้ามาได้ภายในเวลาเดียวกัน โดยจะทำการรวบรวมเป็นค่าข้อมูล 1 ค่า เมื่อผู้ใช้งานที่กรอกข้อมูลเข้ามาทำการกดที่ปุ่ม **‘Submit’**

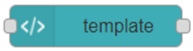
วิดเจ็ทเหล่านี้ นอกเสียจากจะสามารถสร้างค่าข้อมูลใหม่ขึ้นมาได้แล้วนั้น ตัวมันเองยังสามารถรับค่าข้อมูลจากวิดเจ็ทหรือโหนดอื่น ๆ ขึ้นมาใช้งานได้เช่นกัน

วิดเจ็ทอื่น ๆ

นอกเหนือจากวิดเจ็ทสองประเภทข้างต้น ยังมีวิดเจ็ทนอกเหนือจากนั้นที่สามารถนำไปใช้งานให้แดชบอร์ดมีประสิทธิภาพมากขึ้นกว่าเดิมได้อีกสองชนิด ได้แก่



ตัวควบคุมหน้าต่างแสดงผล (UI Control) ใช้ในการควบคุมเกี่ยวกับหน้าต่างแสดงผล



แบบร่าง (Template) อนุญาตให้ผู้ใช้งานสามารถใส่ค่า HTML หรือ Angular ลงไปได้ โดยจะเป็นการสร้างการแสดงผลที่จะเปลี่ยนแปลงในทันทีตามข้อความที่เข้ามาภายในกล่องฟังก์ชั่น เช่น เปลี่ยนค่าการแสดงว่าผลว่าข้อมูลที่ได้รับเข้ามานั้นเป็นจำนวนคี่หรือจำนวนคู่ เป็นต้น

```
<div layout="row" layout-align="space-between">
  <p>The number is</p>
  <font color="{{((msg.payload || 0) % 2 === 0) ? 'green' : 'red'}}">
    {{(msg.payload || 0) % 2 === 0 ? 'even' : 'odd'}}
  </font>
</div>
```

หลอดไฟ

หลอดไฟ (Led) จะแสดงผลสถานะของค่าที่ได้รับเข้ามา โดยใช้สีในการควบคุมลักษณะของแต่ละสถานะของข้อมูลนั้น แต่หากข้อมูลที่เข้ามาไม่ตรงกับลักษณะที่ถูกกำหนดเอาไว้ของหลอดไฟสีของหลอดไฟจะกลายเป็นสีเทาเพื่อแสดงว่าอยู่ในสถานะปิด

- **Label** คือคำอธิบายประกอบกับหลอดไฟที่แสดงผล
- **Label Placement** คือการเลือกว่าต้องการให้คำอธิบายประกอบหลอดไฟนั้นปรากฏที่ด้านขวาหรือด้านซ้ายของหลอดไฟ
- **Label Alignment** คือการเลือกตำแหน่งการแสดงผลของคำอธิบายประกอบหลอดไฟนั้นว่าต้องการให้ชิดซ้าย ชิดขวา หรืออยู่กึ่งกลาง
- **Colors for value of `msg.payload`** คือการเลือกสีประกอบค่าข้อมูล `msg.payload` ที่ถูกส่งเข้ามาภายในโหนดนี้
- **Allow Color For Value Map in msg** คือการอนุญาตให้ใช้สีที่ถูกส่งมาจาก msg จากโหนดก่อนหน้าในการใช้แสดงผลบนสวิตช์

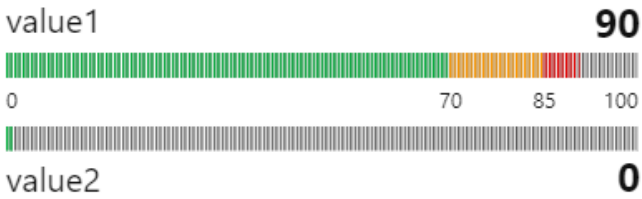
เลเวล

เลเวล (Level) เป็นเสมือนขีดระดับที่ใช้บอกถึงระดับของข้อมูลโดยอ้างอิงตามจำนวนขีดและ

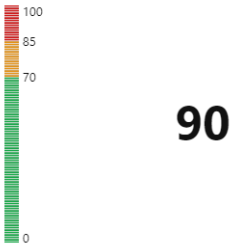
สีที่ได้ทำการตั้งค่าเอาไว้ก่อนหน้า

- **Layout** คือการกำหนดลักษณะในการแสดงผลของเลเวล ซึ่งสามารถเลือกให้แสดงออกมาได้ 3 รูปแบบ ได้แก่
    - **Single Horizontal** คือการแสดงเลเวลของข้อมูลในลักษณะแถบขีดในแนวนอน ซึ่งใช้สีในการบอกถึงระดับปัจจุบันของค่าข้อมูลนั้น ซึ่งการแสดงผลในลักณะนี้มีการตั้งค่าเฉพาะตัวอยู่หนึ่งอย่าง ดังนี้

        - **Label** การขึ้นคำประกอบควบคู่กับค่าข้อมูล ซึ่งจะแสดงอยู่เหนือแถบเลเวลด้านหน้าตัวเลขของค่าข้อมูลในปัจจุบัน โดยจะแสดงข้อความตามที่ผู้ใช้งานได้ทำการตั้งค่าเอาไว้
    - **Pair Horizontal** คือการแสดงผลเลเวลของข้อมูลเป็นสองแถบ



        - **Channels** คือการใช้คำประกอบควบคู่กับค่าข้อมูลทั้งสองแถบ ซึ่งผู้ใช้งานสามารถเปลี่ยนคำที่ประกอบกับทั้งแถบเลเวลด้านบน (Top) และแถบเลเวลด้านล่าง (Bottom)
    - **Single Vertical** คือการแสดงผลในรูปแบบเดียวกับ Single Horizontal แต่จะแสดงผลของเลเวลออกมาในลักษณะของแนวตั้ง



        - **Label** การขึ้นคำประกอบควบคู่กับค่าข้อมูล ซึ่งจะแสดงอยู่ด้านข้างแถบเลเวลเหนือตัวเลขของค่าข้อมูลในปัจจุบัน โดยจะแสดงข้อความตามที่ผู้ใช้งานได้ทำการตั้งค่าเอาไว้

- **High** คือการเลือกสีในกรณีที่ค่าข้อมูลที่เข้ามาอยู่ในระดับที่สูงมาก ค่าเริ่มต้นคือ #e60000
- **Warm** คือการเลือกสีในกรณีที่ค่าข้อมูลที่เข้ามาอยู่ในระดับที่สูงกว่าปกติ ค่าเริ่มต้นคือ #ff9900
- **Normal** คือการเลือกสีในกรณีที่ค่าข้อมูลที่เข้ามาอยู่ในระดับปกติ ค่าเริ่มต้นคือ #00b33c
- **Off** คือการเลือกสีในช่วงของค่าข้อมูลที่ยังไปไม่ถึงหรือสีพื้นหลังของเลเวลนั้น ค่าเริ่มต้นคือ #595959
- **Show** คือลักษณะในการแสดงผลของสีที่จะปรากฏขึ้นบนแถบเลเวลโดยอ้างอิงตามระดับของค่าข้อมูลที่ได้รับเข้ามา
    - **Multiple segments** คือการแสดงผลระดับของข้อมูลภายในแถบเลเวลนั้นโดยแบ่งแต่ละช่วงข้อมูลออกเป็นสี ๆ ตามที่ผู้ใช้งานได้ทำการกำหนดเอาไว้

        - **Peak Mode** ในกรณีที่ค่าข้อมูลใหม่ที่เข้ามาอยู่ในระดับที่ต่ำกว่าค่าข้อมูลเดิมที่มีอยู่ ผู้ใช้งานสามารถตั้งค่าให้แสดงผลค่าสูงที่สุดของค่าข้อมูลก่อนหน้าเอาไว้ได้ด้วยการเปิดการใช้งาน Hold Peak เพื่อแสดงผลจุดของข้อมูลในค่าที่สูงที่สุดตามระยะเวลาที่กำหนดเอาไว้ในหน่วยมิลลิวินาที (ms) ซึ่งค่าเริ่มต้นในการแสดงผลค่านี้คือ 3000 มิลลิวินาทีหรือ 3 วินาที

    - **Single color bar** จะเปลี่ยนสีที่ปรากฏทั้งหมดให้กลายเป็นสีเดียวกันโดยอ้างอิงตามระดับของค่าข้อมูลที่ได้รับเข้ามา

    - **Interpolated colors** คือการไล่ระดับสีของข้อมูลขึ้นไปตามระดับของค่าข้อมูลที่ได้รับเข้ามา

- **Text Options** เป็นการตั้งค่าลักษณะการแสดงผลของข้อความบนขีดระดับ
    - **Default** คือการใช้ค่าเริ่มต้นที่กำหนดให้ของโปรแกรม
    - **Custom** จะเป็นการกำหนดลักษณะของข้อความที่ปรากฏด้วยตัวของผู้ใช้งานเอง ซึ่งมีส่วนในการตั้งค่าสองส่วนด้วยกัน
        - **Font Sizes** คือการกำหนดขนาดของคำหรือข้อความที่ปรากฏอยู่คู่กับแถบเลเวลนี้ ซึ่งมีอยู่ด้วยกันสามส่วนด้วยกัน
            - **Label** คือขนาดของคำที่ประกอบค่าข้อมูล
            - **Value** คือการกำหนดขนาดของค่าข้อมูลที่จะถูกนำมาแสดงผล
            - **Small** คือการคำหนดขนาดของค่าขอบบนและขอบล่างของแถบเลเวลที่ปรากฏอยู่ริมสุดทั้งสองฝั่งของแถบเลเวล
        - **Font Color** คือการกำหนดว่าต้องการใช้สีของคำหรือข้อความที่ปรากฏนั้นโดยอ้างอิงจากสีธีมหรือไม่ หากไม่ ผู้ใช้งานสามารถกำหนดสีของข้อความที่กำหนดได้ด้วยตัวเอง (Custom Color)
- **Range** คือการกำหนดค่าข้อมูลที่ต่ำที่สุดหรือขอบล่าง (Min) และค่าข้อมูลที่สูงที่สุดหรือขอบบน (Max) ของข้อมูลที่สามารถเป็นไปได้ในการแสดงผลออกมา
- **Sectors** คือการแบ่งช่วงของข้อมูลออกเป็นหลาย ๆ ส่วน (สูงสุด 4 ส่วน) เพื่อใช้ในการเปลี่ยนแปลงสีของค่าข้อมูลบนแถบเลเวลเมื่อค่าข้อมูลนั้นขยับไปถึงช่วงข้อมูลที่ได้ทำการกำหนดเอาไว้
- **Format** เป็นส่วนที่ใช้ในการตั้งค่าเกี่ยวกับวิธีการแสดงผลของข้อมูลบนแถบเลเวล ซึ่งประกอบไปด้วยการตั้งค่าในสองส่วนด้วยกัน
    - **Unit** คือหน่วยของข้อมูลที่ผู้ใช้งานสามารถกหนดเป็นข้อความใดก็ได้
    - **Decimals** คือจำนวนจุดทศนิยมของข้อมูลที่ผู้ใช้งานต้องการให้แสดงผลออกมา
- **Value Field** คือการกำหนดว่าต้องการให้ซ่อนค่าข้อมูล (Hide Value Field) ของแถบเลเวลนี้ไปหรือไม่
- **Stripe Shape** คือลักษณะของการซอยย่อยแถบเลเวลออกเป็นส่วน ๆ ว่าต้องการให้มีลักษณะอย่างไร
    - **Superfine** คือการแบ่งออกเป็นส่วนย่อยเล็ก ๆ ที่ละเอียดยิบ

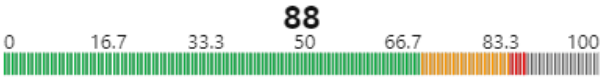
- **Fine** คือการแบ่งออกเป็นส่วนย่อยเล็ก ๆ แต่หยาบกว่า Superfine

- **Normal** คือการแบ่งออกเป็นส่วนย่อยที่หยาบกว่าสองรูปแบบก่อนหน้า

- **Tick Values** คือการกำหนดการแสดงผลของค่าที่ขึ้นประกอบบนแถบเลเวล (ไม่นับค่าที่เป็นขอบบนและขอบล่างของค่าข้อมูล)
  - **Off** คือปิดไม่ให้มีการแสดงผลค่าเหล่านั้นขึ้นมา

  - **Segments** จะแสดงผลเฉพาะค่าที่เป็นช่วงต่าง ๆ ของข้อมูลเท่านั้น

  - **Auto** จะทำการแบ่งช่วงข้อมูลที่จะแสดงผลออกมาเป็นช่วง ๆ ที่เท่ากัน โดยอ้างอิงตามความกว้างของวิดเจ็ทนี้



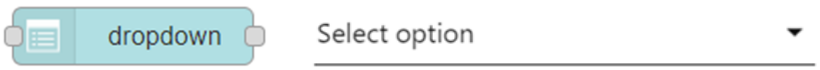
- **Animations** คือลักษณะการเคลื่อนที่ของแถบสีที่ปรากฏในแถบเลเวล
  - **Soft** จะเพิ่มและลดแถบสีอย่างนิ่มนวลและช้ากว่าทุกประเภทอื่น ๆ
  - **Reactive** จะเปลี่ยนแปลงแถบสีแบบแทบจะทันทีที่ข้อมูลภายในแถบเลเวลนี้มีการเปลี่ยนแปลงไป แต่ยังคงมีการเพิ่มลดที่นุ่มนวลคล้ายกับ Soft
  - **Rocket** มีการเปลี่ยนแปลงในลักษณะเดียวกับ Soft และ Reactive แต่เกิดขึ้นอย่างรวดเร็วมากกว่าทั้งสองประเภทจนแทบจะมองไม่เห็นถึงการเพิ่มหรือลดแถบสีในแถบเลเวล
  - **No animations** คือการปรากฏแถบสีโดยอ้างอิงตามค่าข้อมูลโดยตรง ซึ่งจะไม่มีการขยับเพิ่มหรือลดค่าแถบสีใด ๆ ทั้งสิ้น
- **Animated value** จะเป็นการกำหนดให้ค่าข้อมูลที่ปรากฏอยู่ควบคู่กับแถบเลเวลนี้มีการเคลื่อนไหวเปลี่ยนแปลงได้ทีละหน่วยอย่างรวดเร็วจนถึงค่าล่าสุดของข้อมูลนี้

ปุ่มกด

ปุ่มกด (Button) ใช้ในการเพิ่มปุ่มกดเข้าไปในแดชบอร์ด โดยจะทำการสร้างข้อความขึ้นมาเมื่อผู้ใช้งานทำการคลิกลงไปที่ปุ่มกดนี้

- **Icon** คือสัญลักษณ์ของปุ่มกด ซึ่งผู้ใช้งานสามารถเปลี่ยนสัญลักษณ์นี้ได้โดยทำการพิมพ์ชื่อของสัญลักษณ์ที่ต้องการลงไป ซึ่งประเภทของสัญลักษณ์ที่สามารถนำมาใช้งานได้นั้นสามารถเปลี่ยนได้เช่นเดียวกับสัญลักษณ์ของหน้าแดชบอร์ด ได้แก่ Material Design, Font Awesome, และ Weather
- **Label** คือการกำหนดคำที่จะปรากฏอยู่บนปุ่มกด โดยค่าเริ่มต้นจะใช้คำว่า ‘button’ ซึ่งเมื่อปรากฏขึ้นในหน้าแดชบอร์ด คำที่เขียนเอาไว้จะถูกปรับให้กลายเป็นอักษรพิมพ์ใหญ่ทั้งหมด
- **Tooltip** จะปรากฏเป็นคำช่วยเหลือขึ้นเมื่อนำเมาส์ไปลอยเหนือปุ่มนี้
- **Colour** คือการกำหนดสีของข้อความหรือสัญลักษณ์ที่นำมาปรากฏบนปุ่มกดนี้
- **Background** จะเป็นการกำหนดสีพื้นหลังของปุ่มกด
- **When clicked, send:** จะเป็นการกำหนดว่าเมื่อผู้ใช้งานทำการคลิกที่ปุ่มกดนี้แล้ว จะให้ทำการส่งค่าใดออกไปอย่างไร ซึ่งประกอบไปด้วยค่าข้อมูล payload และ topic ที่ผู้ใช้งานจัดการในการส่งค่าเหล่านี้ได้ด้วยตัวเอง
- **If `msg` arrives on input, emulate a button click** คือการกำหนดว่า ให้ทุกครั้งที่มีข้อมูลผ่านเข้ามาในโหนดนี้นั้นทำงานเสมือนปุ่มนี้ถูกกดหรือไม่

รายการตัวเลือก



รายการตัวเลือก (Drop-down) จะอนุญาตให้ผู้ใช้งานสามารถเพิ่มตัวเลือกต่าง ๆ ของค่าข้อมูลไป โดยผู้ใช้งานสามารถเลือกค่าข้อมูลในตัวเลือกมากกว่า 1 ชนิดได้

- **Label** คือชื่อที่จะปรากฏขึ้นกำกับรายการตัวเลือกนี้เอาไว้
- **Tooltip** จะปรากฏเป็นคำช่วยเหลือขึ้นเมื่อนำเมาส์ไปลอยเหนือปุ่มนี้
- **Placeholder** คือการกำหนดคำเริ่มต้นที่จะปรากฏอยู่บนเมนู drop-down โดยค่าเริ่มต้นคือคำว่า ‘Select option’

- **Options** คือการกำหนดตัวเลือกของข้อค่าข้อมูลที่จะปรากฏใน drop-down ซึ่งผู้ใช้งานสามารถเพิ่มตัวเลือกเข้าไปได้โดยการคลิกที่ปุ่ม '**+Add**' ด้านล่างของกล่องการตั้งค่านี้ ซึ่งเมื่อทำการคลิกเพื่อเพิ่มตัวเลือกแล้ว จะปรากฏเป็นตัวเลือกให้ตั้งค่าภายในได้ ซึ่งประกอบไปด้วยประเภทของค่าตัวเลือก ค่าตัวเลือก (Value) และคำประกอบค่าตัวเลือกนั้น (Label)
- **Allow multiple selection from list** จะให้ผู้ใช้งานเลือกว่าต้องการให้คนที่มาใช้งานวิดเจ็ทนี้สามารถเลือกตัวเลือกที่อยู่ภายใน drop-down หลาย ๆ ตัวเลือกได้หรือไม่ ซึ่งค่าเริ่มต้นของวิดเจ็ทนี้คือการอนุญาตให้สามารถทำได้ แต่หากปิดการใช้งานส่วนนี้ ผู้ใช้งานจะสามารถเลือกตัวเลือกภายใน drop-down ได้เพียงค่าเดียวเท่านั้น
- **If** `msg` **arrive on input, pass through output** หากเปิดการใช้งาน ข้อความที่ถูกส่งเข้ามาภายในโหนดจะถูกส่งไปเป็นข้อมูลขาออกของโหนดนี้ด้วยเช่นกัน
- **Topic** คือการตั้งชื่อหัวข้อให้กับค่าข้อมูลที่ถูกส่งออกไปจากวิดเจ็ทนี้

สวิตช์

สวิตช์ (Switch) จะทำการเพิ่มสวิตช์ซึ่งใช้ในการส่งสถานะเปิด (On) หรือปิด (Off) ซึ่งจะแสดงผลสลับกันไปทุกครั้งที่มีการคลิก

- **Label** คือการระบุคำให้ปรากฏคู่กับสวิช์นี้
- **Tooltip** จะปรากฏเป็นคำช่วยเหลือขึ้นเมื่อนำเมาส์ไปลอยเหนือสวิตช์
- **Icon** คือสัญลักษณ์ของสวิตช์ ซึ่งผู้ใช้งานสามารถเปลี่ยนสัญลักษณ์นี้ได้โดยทำการพิมพ์ชื่อของสัญลักษณ์ที่ต้องการลงไป ซึ่งประเภทของสัญลักษณ์ที่สามารถนำมาใช้งานได้นั้นสามารถเปลี่ยนได้เช่นเดียวกับสัญลักษณ์ของหน้าแดชบอร์ด ได้แก่ Material Design, Font Awesome, และ Weather
- **When clicked, send** คือค่าข้อมูลที่จะถูกส่งออกไปจากโหนดนี้ในกรณีที่มีการเปลี่ยนแปลงค่าของสวิตช์ พูดในอีกแง่หนึ่งคือ ข้อมูลจะถูกส่งไปหาผู้ใช้งานทำการเปิดหรือปิดสวิตช์นี้
    - **On payload** คือค่าข้อมูลที่จะถูกส่งไปเมื่อสวิตช์ได้ทำการเปิดขึ้น
    - **Off payload** คือค่าข้อมูลที่จะถูกส่งไปเมื่อสวิตช์ได้ถูกปิดลง

- **Topic** คือหัวข้อของข้อมูลที่สามารถใช้ส่งคู่ไปด้วยกันกับข้อมูลนั้น

แถบเลื่อนเส้นจำนวน

แถบเลื่อนเส้นจำนวน (Slider) คือแถบเลื่อนของค่าคำนวณที่ผู้ใช้งานสามารถกำหนดค่าต่ำสุดและสูงสุดของเส้นจำนวนได้ และใช้งานในการเลื่อนปุ่มบนเส้นจำนวนเพื่อเลือกค่าที่ต้องการ

- **Label** จะเป็นการระบุคำที่ต้องการให้ปรากฏขึ้นประกอบกับแถบเลื่อนเส้นจำนวนนี้
- **Tooltip** คือคำช่วยเหลือขนาดสั้นที่จะปรากฏขึ้นเมื่อผู้ใช้งานนำเอาเมาส์ไปลอยเหนือวิดเจ็ทนี้
- **Range** คือการกำหนดขอบบนและขอบล่างของแถบเลื่อนเส้นจำนวน
  - **Min** คือการกำหนดค่าที่ต่ำที่สุด (ขอบล่าง) ของแถบเลื่อนเส้นจำนวน ค่าเริ่มต้นของค่านี้คือ 0
  - **Max** คือการกำหนดค่าที่สูงที่สุด (ขอบบน) ของแถบเลื่อนเส้นจำนวน ค่าเริ่มต้นของค่านี้คือ 100
  - **Step** คือการกำหนดค่าการเปลี่ยนแปลงของแถบเลื่อนเส้นจำนวนว่าต้องการให้ขยับไปทีละเท่าไหร่ ซึ่งค่าเริ่มต้นของค่านี้คือ 1 ซึ่งหมายถึงจะมีการเปลี่ยนแปลงของค่าทีละ 1 เมื่อผู้ใช้งานทำการขยับแถบเลื่อนเส้นจำนวน
- **Output** คือการเลือกว่าต้องการให้ผลลัพธ์ที่ออกจากวิดเจ็ทนี้ออกไปในลักษณะใด
  - **Continuously while sliding** ให้แสดงผลต่อเนื่องในขณะที่ทำการเลื่อนค่าข้อมูล
  - **Only on release** ส่งค่าข้อมูลออกเฉพาะในกรณีที่ปล่อยเมาส์ออกจากแถบเลื่อนนี้แล้วเท่านั้น
- **When changed, send** จะเป็นการตั้งค่าข้อความที่จะถูกส่งออกไปจากวิดเจ็ทนี้เมื่อมีการเปลี่ยนแปลงของจำนวนในแถบเส้นจำนวน โดยจะทำการส่งค่า payload ออกไปโดยอ้างอิงตามค่าข้อมูลปัจจุบัน และค่า topic ที่ผู้ใช้งานสามารถระบุได้เอง

กล่องเปลี่ยนตัวเลข

กล่องเปลี่ยนตัวเลข (Numeric) จะปรากฏเป็นตัวกำหนดค่าข้อมูลตัวเลขภายในหน้าแดช

บอร์ด ซึ่งผู้ใช้งานสามารถกำหนดค่าต่ำสุดและสูงสุดที่สามารถใส่ค่าได้ลงไปได้

- **Label** คือการกำหนดคำที่ต้องการให้ขึ้นประกอบกับกล่องเปลี่ยนตัวเลขนี้
- **Tooltip** คือการขึ้นแสดงข้อความช่วยเหลือขึ้นมาเมื่อนำเมาส์ไปลอยอยู่เหนือกล่องเปลี่ยนตัวเลขนี้
- **Value format** สามารถช่วยเปลี่ยนแปลงวิธีการแสดงผลข้อมูลได้ เช่น หากผู้ใช้งานพิมพ์ลงไปว่า `{{value}}%` หากค่าข้อมูลที่เข้ามาคือ 24 ข้อมูลจะแสดงออกมาว่า 24% ในขณะเดียวกัน ผู้ใช้งานยังสามารถใช้ค่า HTML หรือ Angular filters มาใส่ลงไปได้เช่นกัน เช่น `&deg;` จะแสดงสัญลักษณ์ขององศา (°) เป็นต้น ยิ่งไปกว่านั้น ผู้ใช้งานยังสามารถตั้งค่าให้กล่องเปลี่ยนตัวเลขนี้สามารถแก้ไขหรือใส่ค่าลงไปตรง ๆ ได้ด้วยการพิมพ์ว่า `{{msg.payload}}` ลงไปเพื่อรอรับค่าข้อมูลที่จะถูกกรอกลงไปโดยผู้ใช้งาน
- **Range** คือการกำหนดขอบบนและขอบล่างของตัวเลข
    - **Min** คือการกำหนดค่าที่ต่ำที่สุด (ขอบล่าง) ของตัวเลข ค่าเริ่มต้นของค่านี้คือ 0
    - **Max** คือการกำหนดค่าที่สูงที่สุด (ขอบบน) ของแถบตัวเลข ค่าเริ่มต้นของค่านี้คือ 100
    - **Step** คือการกำหนดว่าต้องการให้มีการเปลี่ยนแปลงค่าของตัวเลขขึ้นและลงทีละเท่าไหร่ ซึ่งค่าเริ่มต้นของค่านี้คือ 1 ซึ่งหมายถึงจะมีการเปลี่ยนแปลงค่าของตัวเลขขึ้นและลงทีละ 1 หน่วย เช่น จาก 36 เป็น 37 หรือ 35 เป็นต้น
- **Wrap value from max to min and min to max** คือการกำหนดว่าต้องการให้ค่าของตัวเลขนั้นสามารถวนจากค่าสูงที่สุดลงมาต่ำที่สุด หรือวนจากค่าต่ำที่สุดขึ้นไปค่าสูงที่สุดเมื่อเลือกเปลี่ยนค่าขึ้นหรือลงได้หรือไม่
- **When changed, send** จะเป็นการตั้งค่าข้อความที่จะถูกส่งออกไปจากวิดเจ็ทนี้เมื่อมีการเปลี่ยนแปลงของจำนวนในแถบกำหนดค่าตัวเลข โดยจะทำการส่งค่า payload ออกไปโดยอ้างอิงตามค่าข้อมูลปัจจุบัน และค่า topic ที่ผู้ใช้งานสามารถระบุได้เอง

ช่องกรอกข้อความ

ช่องกรอกข้อความ (Text Input) จะปรากฏเป็นเส้นตรงขนาดยาวที่ให้ผู้ใช้งานสามารถกรอกข้อความที่ต้องการลงไปได้ โดยสามารถเลือกได้ว่าต้องการให้ใส่เป็นข้อความธรรมดา อีเมล์ หรือตัว

เลือกสีก็ได้เช่นกัน

- **Label** คือการกำหนดคำที่ต้องการให้ขึ้นประกอบกับช่องกรอกข้อความนี้
- **Tooltip** คือการขึ้นแสดงข้อความช่วยเหลือขึ้นมาเมื่อนำเมาส์ไปลอยอยู่เหนือช่องกรอกข้อความนี้
- **Mode** จะเป็นการกำหนดลักษณะของข้อความที่สามารถอนุญาตให้ใส่ได้ ซึ่งจะปรากฏในลักษณะของ drop-down ให้ผู้ใช้งานเลือกได้ว่าต้องการใช้ลักษณะใด โดยหากผู้ใช้งานพิมพ์ข้อความลงไปไม่ตรงกับประเภทที่กำหนดไว้ ค่านั้นจะขึ้นเป็นตัวอักษรสีแดงเพื่อแสดงถึงความผิดพลาดในประเภทของข้อมูลที่ถูกใส่ลงไป
    - **Text input** คือข้อความทั่วไปที่สามารถใส่ได้

      Normal text input

    - **Email address** คือการกรอกข้อมูลในลักษณะของอีเมล์ xxxxxxxxxx@xxxxx.xxx ที่ผู้ใช้งานสามารถใช้ได้

      alonewolf@gmail.com

    - **Password** จะเป็นการทำให้ข้อความที่ถูกใส่ลงไปจะถูกซ่อนเอาไว้ในลักษณะ ‘••••••••’ เช่นเดียวกับรหัสผ่าน (Password) ทั่ว ๆ ไป

      ••••••••••••••••••

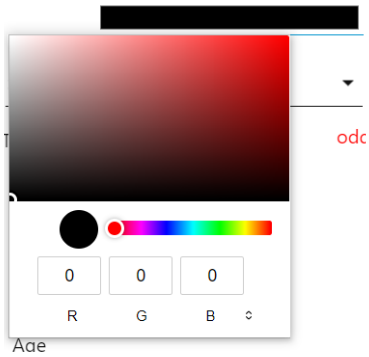
    - **Number** จะกำหนดให้สามารถใส่ได้เพียงค่าข้อมูลตัวเลขเท่านั้น ไม่อนุญาตให้พิมพ์ข้อมูลในลักษณะของข้อความประเภทอื่น ๆ ลงไป โดยผู้ใช้งานสามารถเพิ่มหรือลดค่าจำนวนที่ใส่ลงไปโดยการกดที่ปุ่มลูกศรบนช่องกรอกข้อความนี้ได้ด้วยเช่นกัน

      1234567890

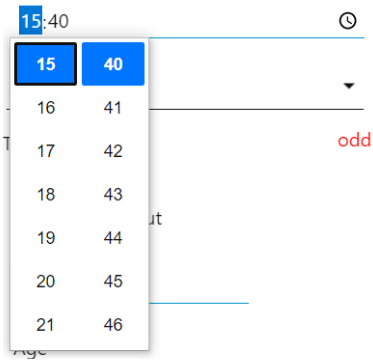
    - **Telephone input** คือการกำหนดลักษณะของข้อมูลให้อยู่ในรูปแบบของเบอร์โทรศัพท์

      0999999999

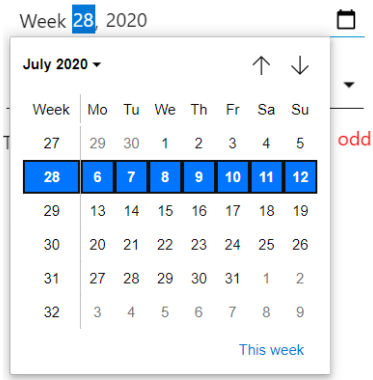
    - **Color picker** คือการกำหนดสีที่ต้องการนำมาแสดงผล ซึ่งจะปรากฏออกมาในลักษณะของแถบสีให้ผู้ใช้งานสามารถเลือกสีที่ต้องการได้



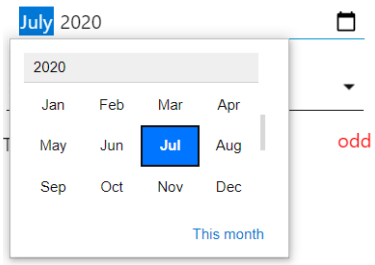
- **Time picker** คือการเลือกค่าเวลาในรูปแบบ ‘HH:mm’



- **Week picker** คือการเลือกช่วงสัปดาห์ในแต่ละปี ซึ่งผู้ใช้งานสามารถคลิกที่ ‘This week’ เพื่อเลือกเป็นสัปดาห์ปัจจุบันได้ โดยโปรแกรมจะทำการให้วันจันทร์เป็นวันตั้งต้นของสัปดาห์ และปิดท้ายสัปดาห์ด้วยวันอาทิตย์



- **Month picker** คือการเลือกเดือนในแต่ละปี ซึ่งผู้ใช้งานสามารถคลิกที่ ‘This month’ เพื่อเลือกเป็นเดือนปัจจุบันได้



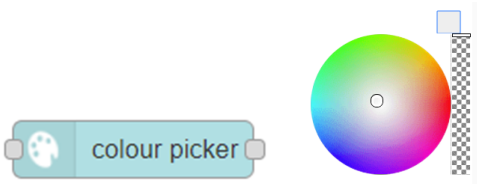
- **Delay (ms)** คือการตั้งเวลาในหน่วยมิลลิวินาทีก่อนที่ข้าข้อมูลจะถูกส่งออกจากโหนดนี้ไป โดยการตั้งไว้ที่ 0 จะหมายถึงจะรอจนกว่าปุ่ม `Enter` หรือ `Tab` ถูกกดจึงจะทำการส่งออกไป ซึ่งการกดปุ่มทั้งสองค่านั้นจะทำการส่งค่า payload ออกไป แต่ `Enter` จะยังคงที่อยู่ที่ค่าเดิมที่ใช้งานอยู่ ในขณะที่ `Tab` จะขยับเปลี่ยนไปยัง field อื่นแทน
- **When changed, send** คือการกำหนดว่าต้องการให้ส่งค่าข้อมูลออกไปจากโหนดนี้อย่างไรเมื่อเกิดการเปลี่ยนแปลงขึ้นภายในช่องกรอกข้อความ ซึ่งโดยปกติแล้วจะส่งค่าข้อมูลปัจจุบันออกไปใน payload โดยผู้ใช้งานสามารถแก้ไข topic ที่ต้องการใช้งานได้ด้วยตัวเอง

ตัวเลือกวันที่

ตัวเลือกวันที่ (Date Picker) จะปรากฏเป็นกล่องเพื่อให้ผู้ใช้งานสามารถกำหนดวันที่ที่ต้องการได้ โดยผู้ใช้งานสามารถเลือกใช้รูปแบบของการแสดงผลวันที่ได้ตามที่ต้องการ

- **Label** คือการขึ้นคำประกอบตัวเลือกวันที่นี้
- **If `msg` arrives on input, pass through output:** คือการเลือกว่าต้องการให้ส่งค่าข้อความ msg ที่ได้รับเข้ามาผ่านไปยังจุดเชื่อมต่อขาออกของโหนดนี้หรือไม่
- **When changed, send** จะเป็นการกำหนดว่า เมื่อมีการเปลี่ยนแปลงของค่าข้อมูลภายในตัวเลือกวันที่นี้แล้ว จะให้ส่งค่าข้อมูลใดออกไป ซึ่งค่าข้อมูล payload นั้นจะส่งค่าข้อมูลปัจจุบันไป และผู้ใช้งานสามารถกำหนดค่า ะopic ที่ต้องการส่งควบคู่กันไปได้เช่นกัน

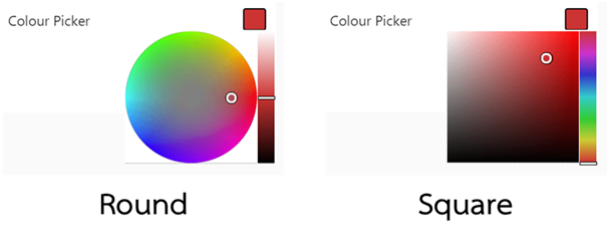
ตัวเลือกสี



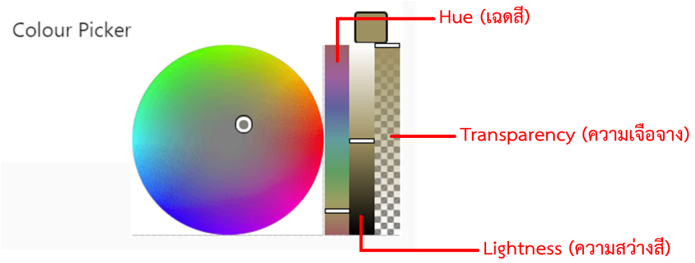
ตัวเลือกสี (Colour Picker) จะเพิ่มกล่องตัวเลือกสีเพื่อให้ผู้ใช้งานสามารถเลือกสีที่ต้องการ

- **Label** คือการขึ้นคำประกอบควบคู่กับวิดเจ็ทนี้
- **Format** คือการกำหนดลักษณะการแสดงผลของวิดเจ็ทนี้ และรูปแบบของการกำหนดค่าสีประเภทต่าง ๆ
    - **Colour Type** คือการกำหนดประเภทของค่าสีที่จะใช้ในการเลือก ซึ่งจะส่งผลออกเป็นค่าข้อมูลที่ถูกส่งออกไปเมื่อผู้ใช้งานเลือกสีต่าง ๆ ในวิดเจ็ทนี้
        - **HEX** คือการกำหนดค่าสีให้อยู่ในลักษณะของเลขฐาน 16 ซึ่งจะส่งค่าข้อมูลในเลขฐาน 16 ออกไป เช่น `a08f5d` เป็นต้น
        - **HEX8** คือการกำหนดค่าสีให้อยู่ในลักษณะของเลขฐาน 16 เช่นเดียวกับค่า HEX ธรรมดา แต่จะพ่วงท้ายไปด้วยความเจือจางของสี (Transparency) ที่จะห้อยเป็นเลขฐาน 16 อยู่ท้ายสุดของค่าข้อมูล (เลขสองตัวสุดท้ายของค่าข้อมูลฐาน 16) เช่น `c44239ff` เป็นต้น โดยค่า `ff` หมายถึงไม่มีความเจือจางใด ๆ ในขณะที่ `00` คือเจือจางจนไม่เห็นสีนั้นอีกต่อไป
        - **HSL** จะเป็นการลงสีโดยอ้างอิงจากค่าเฉดสี (Hue) ปริมาณความเข้มของสี (Saturation) และปริมาณความสว่างของสี (Lightness) ซึ่งค่าข้อมูลจะออกไปในลักษณะ `hsl(Hue, Saturation, Lightness)` เช่น `hsl(57, 59%, 50%)` เป็นต้น ซึ่งค่าเฉดสีที่ออกมานั้นจะอยู่ระหว่าง 0 ถึง 255 ในขณะที่ความเข้มของสีหรือความสว่างของสีจะเป็นค่าร้อยละ
        - **HSV** จะเป็นการลงสีโดยอ้างอิงจากค่าเฉดสี (Hue) ปริมาณความเข้มของสี (Saturation) และปริมาณความสว่างของสี (Value) ซึ่งค่าข้อมูลจะออกไปในลักษณะ `hsv(Hue, Saturation, Value)` เช่น `hsv(110, 55%, 69%)` เป็นต้น
        - **RGB** คือการลงสีโดยอ้างอิงตามความเข้มข้นของค่าสีแดง เขียว และฟ้า ซึ่งจะเป็นการกำหนดค่าข้อมูลของสีที่ผู้ใช้งานเลือกให้ออกไปในลักษณะ `rgb(Red, Green, Blue)` เช่น `rgb(158, 145, 97)` เป็นต้น

- **Colour Picker Shape** จะเป็นการเลือกลักษณะของตัวเลือกสีว่าต้องการให้แสดงผลออกมาในลักษณะวงกลม (Round) หรือเป็นสี่เหลี่ยม (Square)

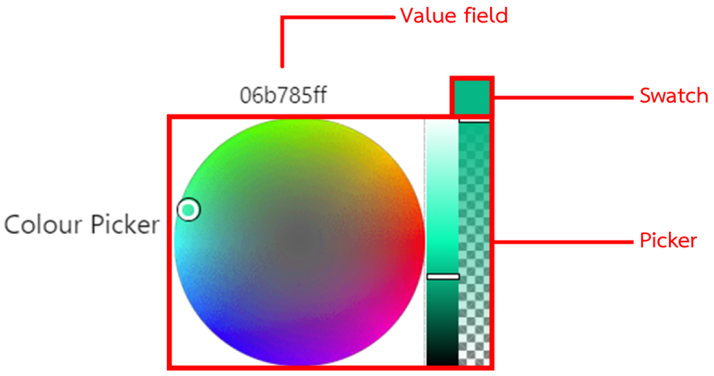


- **Show slider** เป็นตัวเลือกว่าต้องการให้แสดงผลแถบเลื่อนใดบ้างระหว่าง



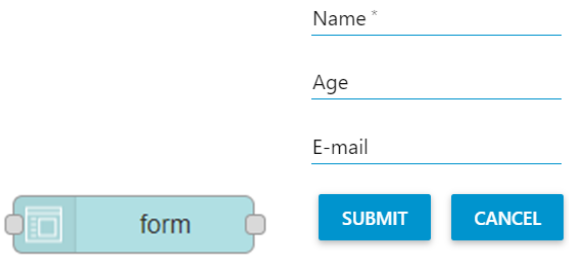
  - **Show hue slider** จะเป็นการเลือกว่าต้องการให้ปรากฏแถบเฉดสีขึ้นมาหรือไม่
  - **Show lightness slider** จะเป็นการเลือกว่าต้องการให้ปรากฏแถบความสว่างของเฉดสีขึ้นมาหรือไม่
  - **Show transparency slider** จะเป็นการเลือกว่าต้องการให้ปรากฏแถบความเจือจางจนเห็นพื้นหลังของสีขึ้นมาหรือไม่ อย่างไรก็ดี แถบนี้จะไม่ปรากฏขึ้นมาในกรณีที่ผู้ใช้งานเลือกให้ค่าของสีแบบ HEX

- **If width is 4 or greater** จะเป็นการเลือกตั้งค่าการแสดงผลของวิดเจ็ทในกรณีที่ขนาดของวิดเจ็ทนี้มีความกว้างตั้งแต่ 4 ขึ้นไปว่าต้องการให้แสดงผลแบบใด ซึ่งผู้ใช้งานจำเป็นต้องเลือกให้แสดงผลในรูปแบบใดรูปแบบหนึ่งเสมอ



    - **Always show swatch** คือให้แสดงกล่องสีที่ถูกเลือกเอาไว้ขึ้นมาตลอดเวลา
    - **Always show picker** คือให้แสดงตัวเลือกสีขึ้นมาตลอดเวลา หากปิดการใช้งานค่านี้ การเลือกสีจะถูกซ่อนเอาไว้และจะแสดงขึ้นมาเมื่อคลิกที่ปุ่มหรือช่องแสดงค่าข้อมูลเท่านั้น
    - **Always show value field** จะเป็นการแสดงกล่องของค่าข้อมูลนั้นขึ้นมาด้านบนของตัวเลือกสี
- **Send** คือการกำหนดว่าต้องการให้ส่งค่าข้อมูลออกไปอย่างไร
    - **One value when released/closed** ส่งเพียงค่าข้อมูลค่าเดียวเมื่อทำการปล่อยเมาส์หรือเมื่อปิดหน้าเลือกสีให้หายไป
    - **Multiple values during editing** คือการอนุญาตให้สามารถส่งหลาย ๆ ค่าได้ระหว่างการแก้ไขหรือเปลี่ยนแปลง
- **Payload** คือการกำหนดประเภทของค่าข้อมูลที่ต้องการส่งออกไปจากโหนดนี้ ซึ่งมีให้เลือกอยู่ด้วยกัน 2 ประเภท คือการส่งค่าปัจจุบันในลักษณะของข้อความ (Current value as a string) และส่งค่าปัจจุบันออกไปในลักษณะของอ็อบเจกต์ (Current value as object)
- **Topic** คือการส่งค่าข้อความที่ต้องการออกไปคู่กับค่าข้อมูล payload ที่ส่งไป

ฟอร์ม



ฟอร์ม (Form) คือแบบสอบถามที่จะปรากฏเป็นกล่องที่ใช้ในการรับข้อมูลมากกว่า 1 อย่างเข้ามาได้ภายในเวลาเดียวกัน โดยจะทำการรวบรวมเป็นค่าข้อมูล 1 ค่าเมื่อผู้ใช้งานที่กรอกข้อมูลเข้ามาทำการกดที่ปุ่ม **‘Submit’**

- **Label** จะปรากฏเป็นคำประกอบควบคู่กับแบบฟอร์มที่ผู้ใช้งานสร้างขึ้น
- **Form elements** คือส่วนที่ใช้ในการประกอบร่างแบบสอบถามขึ้นมา ซึ่งผู้ใช้งานสามารถเพิ่มค่าคำถามต่าง ๆ ที่ต้องการได้โดยการคลิกที่ ‘+ Element’ เพื่อเพิ่มคำถามใหม่ ๆ เข้า

มา ซึ่งแต่ละคำถามนั้นจะมีส่วนการตั้งค่าต่อไปนี้

- **Label** คือคำที่จะขึ้นมาอยู่ในคำถามของแบบสอบถามนั้น
- **Name** คือชื่อของแบบสอบถามข้อนั้น ซึ่งจะทำหน้าที่เป็นค่า key หรือตัวแปรของค่าข้อมูลนั้นที่จะถูกส่งออกจากโหนดไปด้วย payload
- **Type** คือประเภทของค่าข้อมูลที่คำถามนั้นสามารถรับได้ มีหลายประเภทด้วยกัน ได้แก่ ข้อความทั่วไป (Text) ข้อความหลายบรรทัด (Multiline) ตัวเลข (Number) อีเมล์ (E-mail) รหัสผ่าน (Password) กล่องตัวเลือก (Checkbox) สวิตช์ (Switch) และวันที่ (Date)
- **Required** เป็นการกำหนดว่าต้องการบังคับให้คำถามนี้ถูกตอบหรือไม่ โดยจะปรากฏขึ้นเป็น * อยู่ที่คำถามนั้น
- **Rows** จะปรากฏให้ตั้งค่าเมื่อผู้ใช้งานเลือกใช้แบบสอบถามประเภทข้อความหลายบรรทัด (Multiline) ซึ่งจะทำหน้าที่เป็นตัวกำหนดจำนวนบรรทัดที่สามารถพิมพ์ลงไปได้
- **Remove** คือการลบคำถามนี้ออกจากแบบสอบถาม

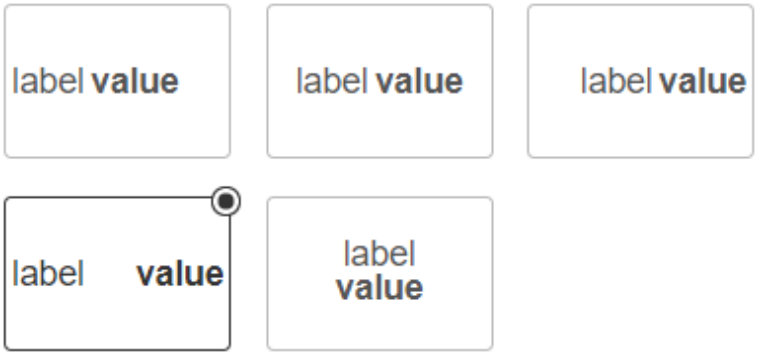
- **Buttons** คือการตั้งชื่อให้กับปุ่มกดทั้งสองปุ่มท้ายฟอร์ม
  - 👍 คือการตั้งคำที่จะให้ปรากฏในปุ่มทางซ้ายหรือปุ่มยืนยัน ซึ่งค่าเริ่มต้นคือ *'Submit'*
  - 👎 คือการตั้งคำที่จะให้ปรากฏในปุ่มทางขวาหรือปุ่มปฏิเสธ ซึ่งค่าเริ่มต้นคือ *'Cancel'* ซึ่งผู้ใช้งานสามารถซ่อนปุ่มนี้ได้โดยการไม่ใส่ค่าใด ๆ ลงไป
- **Topic** คือการกำหนดค่าข้อความที่จะส่งไปคู่กับค่าข้อมูลของวิดเจ็ทนี้

ข้อความ

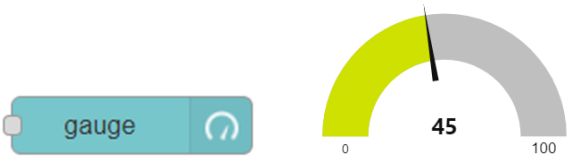
ข้อความ (Text) จะปรากฏเป็นข้อความที่ไม่สามารถแก้ไขได้ขึ้นมา แต่สามารถเปลี่ยนแปลงได้หากมีข้อความส่งเข้ามาผ่านการตั้งค่าด้านหลังของโหนดนี้

- **Label** คือการแสดงคำขึ้นมาประกอบกับค่าข้อความที่ได้รับ โดยสามารถตั้งเป็นข้อความใหม่ หรือใช้ `{{msg.topic}}` เพื่อดึงค่า topic จากโหนดก่อนหน้าที่ส่งข้อมูลเข้ามานำมาแสดงผลก็ได้เช่นกัน

- **Value format** คือการกำหนดการแสดงผลของค่าข้อมูลที่ได้รับเข้ามาภายในโหนดนี้ ซึ่งสามารถใช้ได้ทั้งค่า HTML และ Angular filters เช่น `{{value | uppercase}}` `&deg;` จะทำการแปลงค่าข้อมูลนั้นให้กลายเป็นตัวพิมพ์ใหญ่ และเติมสัญลักษณ์องศา (o) ลงไปท้ายค่าข้อมูลนั้น เป็นต้น
- **Layout** คือการกำหนดลักษณะในการแสดงผลข้อมูลระหว่างคำประกอบ (Label) และค่าข้อมูล (Value) ว่าต้องการให้แสดงออกมาในลักษณะใด ซึ่งสามารถเลือกได้ 5 รูปแบบ



เกจ



เกจ (Gauge) จะปรากฏเป็นมาตรวัดที่จะแสดงผลของค่าข้อมูลที่ได้รับมาขึ้น โดยจะปรากฏแถบสีที่ข้างมาตรวัดเพื่อบ่งบอกถึงระดับของค่าข้อมูลนั้น ซึ่งมีประเภทในการแสดงผล 4 ประเภท ได้แก่ มาตรวัด (Gauge) โดนัท (Donut) เข็มทิศ (Compass) และระดับน้ำ (Level) โดยการแสดงผลทั้ง 4 ประเภทนี้มีส่วนการตั้งค่าหลัก ๆ ที่เหมือนกันอยู่ 4 อย่างด้วยกัน

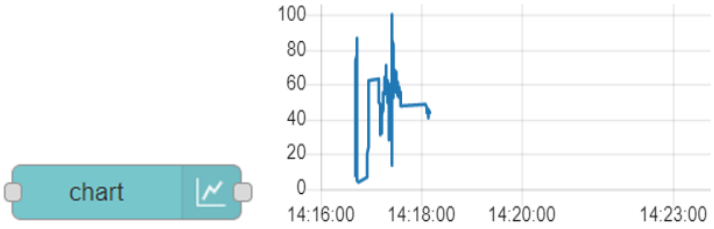
- **Label** คือชื่อของเกจการแสดงผลนั้น ๆ ซึ่งค่าเริ่มต้นจะใช้คำว่า ‘guage’

- **Unit** คือหน่วยของค่าข้อมูลที่ผู้ใช้งานสามารถกำหนดขึ้นมาได้ด้วยตัวเอง
- **Range** จะเป็นการกำหนดค่าขอบบนและขอบล่างของข้อมูล
    - **Min** ค่าต่ำที่สุดที่เป็นไปได้ในการแสดงผลข้อมูล เปรียบเสมือนขอบล่างของข้อมูล
    - **Max** ค่าสูงที่สุดที่เป็นไปได้ในการแสดงผลข้อมูล เปรียบเสมือนขอบบนของข้อมูล

อย่างไรก็ตาม วิดเจ็ทเกจแต่ละประเภทต่าง ๆ ก็มีส่วนการตั้งค่าเพิ่มเติมที่แตกต่างกันเพิ่มขึ้นมา อันเนื่องมาจากลักษณะการแสดงผลที่แตกต่างกัน ซึ่งส่วนการตั้งค่าที่เพิ่มเติมขึ้นมานั้นมีดังนี้

- **Value format** จะปรากฏให้ตั้งค่าในทุกประเภทยกเว้นระดับน้ำ คือการกำหนดลักษณะของค่าข้อมูลที่จะนำมาแสดงผล โดยอ้างอิงจากหลักการแสดงผลของ Angular Filters โดยค่าเริ่มต้นของการแสดงผลนี้จะแสดงออกมาในลักษณะ `{{value}}` หรือแสดงผลที่ได้รับเข้ามาจากค่าก่อนหน้าโดยตรง
- **Colour gradients** จะปรากฏให้ตั้งค่าเฉพาะในมาตรวัดและโดนัท โดยจะเป็นการไล่เฉดสีของข้อมูลโดยแบ่งข้อมูลออกเป็น 3 ส่วนเท่า ๆ กัน โดยให้ในแต่ละส่วนของข้อมูล (หรือแต่ละช่วงข้อมูล) ที่ถูกนำมาแสดงผลปรากฏเป็นเฉดสีที่ต่างกันไปตามระดับของข้อมูลนั้น
- **Sectors** จะปรากฏให้ตั้งค่าเฉพาะในมาตรวัดและโดนัท เป็นการแบ่งช่วงต่าง ๆ ระหว่างค่าต่ำที่สุดและค่าสูงที่สุดให้ออกจากกันเป็นหลาย ๆ ส่วน ซึ่งสามารถแบ่งได้สูงสุด 3 ส่วน เช่น หากค่าต่ำสุดและค่าสูงสุดของข้อมูลเป็น 100 และผู้ใช้งานกำหนดส่วนเป็น 25 และ 75 ค่าข้อมูลจะแบ่งออกเป็น 3 ช่วง ได้แก่ 0 - 25, 26 - 75, และ 76 - 100 นั่นเอง

แผนภูมิ

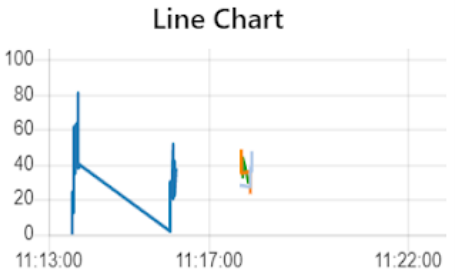


แผนภูมิ (Chart) ใช้ในการแสดงผลข้อมูลแต่ละจุดออกมาเป็นแผนภูมิในรูปแบบต่าง ๆ โดยสามารถเลือกให้อ้างอิงตามเวลา หรือแสดงผลในลักษณะแผนภูมิแท่ง หรือแผนภูมิวงกลมก็ได้เช่นกัน ซึ่งแต่ละประเภทของแผนภูมิต่าง ๆ มีส่วนการตั้งค่าที่ไม่เหมือนกัน แต่จะมีบางส่วนที่มีการตั้งค่าที่คล้ายคลึงกัน เช่น การตั้งชื่อของแผนภูมิใน Label และการตั้งข้อความเบื้องต้นก่อนที่จะมีค่าข้อมูลที่รับได้ส่งเข้ามาใน Blank Label

หากผู้ใช้งานต้องการแสดงผลข้อมูลหลายประเภทภายในแผนภูมิประเภทเดียวกัน ผู้ใช้งานสามารถทำได้โดยการตั้งค่า topic ของแต่ละค่าข้อมูลนั้นให้ต่างกัน ซึ่ง topic ที่ถูกส่งมาจากค่าข้อมูลนั้นจะกลายเป็นชื่อที่กำกับข้อมูลที่เข้ามาในทันที

ในขณะที่ผู้ใช้งานสามารถเปลี่ยนสีของแผนภูมิต่าง ๆ ไปตามที่กำหนดได้โดยให้ใช้สีที่ตั้งค่าไว้ใน Series Colours โดยแผนภูมิจะทำการเรียงลำดับการแสดงผลของแต่ละสีจากซ้ายไปขวาและบนลงล่างของสีที่ถูกกำหนดเอาไว้

แผนภูมิเส้น



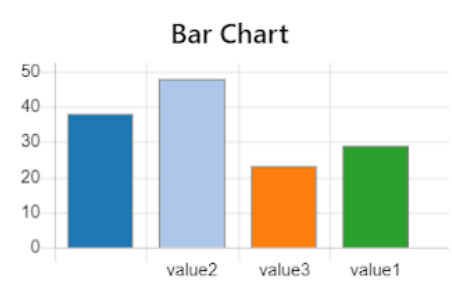
แผนภูมิเส้น (Line Chart) คือการแสดงผลข้อมูลต่าง ๆ ออกมาในลักษณะของเส้นจำนวนที่แปรผันตามเวลาที่ผ่านไปในแนวนอน

การตั้งค่าของแผนภูมิเส้นนั้นมีดังต่อไปนี้

- **Enlarge points** หากเปิดการใช้งานจะเป็นการปรับให้ปรากฏจุดข้อมูลขึ้นมาแสดงผลในแผนภูมิเพื่อแทนค่าข้อมูลแต่ละจุด
- **X-axis** คือการกำหนดจำนวนข้อมูลสูงสุดที่สามารถนำมาแสดงผลได้หากตรงเงื่อนไขใดเงื่อนไขหนึ่งระหว่างช่วงเวลาของข้อมูลที่ได้รับหรือจำนวนของจุดข้อมูล
    - **Last** คือการกำหนดช่วงเวลาย้อนหลังที่จะนำข้อมูลมาแสดงนับจากปัจจุบัน ซึ่งสามารถตั้งได้ตั้งแต่หน่วยวินาที นาที ชั่วโมง วัน และสัปดาห์ โดยค่าเริ่มต้นคือการแสดงข้อมูล 1 ชั่วโมงย้อนหลัง
    - **Points** คือการกำหนดจำนวนของจุดข้อมูลที่จำนะมาแสดงผล ซึ่งค่าเริ่มต้นคือ 1,000 จุด
- **X-axis Label** คือลักษณะของการแสดงผลเวลาในรูปแบบต่าง ๆ
- **As UTC** คือการเลือกให้แสดงผลเวลาโดยอ้างอิงตามเวลาสากลเชิงพิกัด (Coordinated Univarsal Time)
- **Y-axis** คือการกำหนดค่าสูงสุด (Max) และต่ำสุด (Min) ของข้อมูลในแกนแนวตั้ง
- **Legend** จะเป็นการเลือกว่าต้องการให้แสดงเส้นกำกับ (Show) ขึ้นมาด้านข้างของแผนภูมินี้หรือไม่ต้องการให้แสดงขึ้นมา (None)
- **Interpolate** เป็นการกำหนดลักษณะของแผนภูมิเส้นว่าต้องการให้มีการเชื่อมต่อกันระหว่างจุดข้อมูลแต่ละจุดให้ออกมาในรูปแบบใด ซึ่งมีให้เลือกตั้งแต่ Linear, Step,

Bezier, Cubic, และ Cubic-mono

แผนภูมิแท่ง

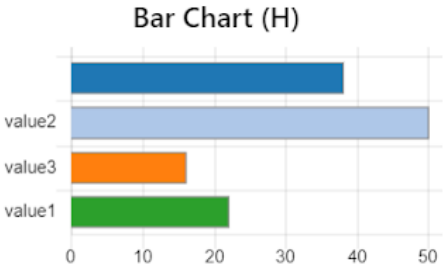


แผนภูมิแท่ง (Bar Chart) คือการแสดงผลข้อมูลในลักษณะของความสูงของค่าข้อมูล ซึ่งแต่ละแท่งจะหมายถึงค่าข้อมูลแต่ละค่า และความสูงคือค่าของข้อมูลในค่านั้น ๆ

การตั้งค่าของแผนภูมิแท่งนั้นมีดังต่อไปนี้

- **Y-axis** คือการกำหนดค่าสูงสุด (Max) และต่ำสุด (Min) ของข้อมูลในแกนแนวตั้ง
- **Legend** จะเป็นการเลือกว่าต้องการให้แสดงเส้นกำกับ (Show) ขึ้นมาด้านข้างของแผนภูมินี้หรือไม่ต้องการให้แสดงขึ้นมา (None)
- **Use first colour for all bars** หากเปิดการใช้งานจะเป็นการให้แผนภูมิแท่งทุกแท่งใช้สีเดียวกันกับแท่งแรกของแผนภูมิ

แผนภูมิแท่ง H

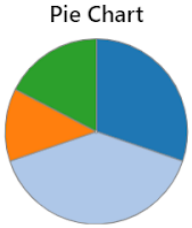


แผนภูมิแท่ง H (Bar Chart (H)) นั่นคือการแสดงผลแบบเดียวกับแผนภูมิแท่งปกติ แต่จะกลับข้างมาเป็นความยาวในแนวนอนแทนความสูง โดย H นั้นย่อมาจากคำว่า Horizontal ที่แปลว่าแนวนอนนั่นเอง

การตั้งค่าของแผนภูมิแท่ง H นั้นมีดังต่อไปนี้

- **X-axis** คือการกำหนดค่าสูงสุด (Max) และต่ำสุด (Min) ของข้อมูลในแกนแนวนอน
- **Legend** จะเป็นการเลือกว่าต้องการให้แสดงเส้นกำกับ (Show) ขึ้นมาด้านข้างของแผนภูมินี้หรือไม่ต้องการให้แสดงขึ้นมา (None)
- **Use first colour for all bars** หากเปิดการใช้งานจะเป็นการให้แผนภูมิแท่งทุกแท่งใช้สีเดียวกันกับแท่งแรกของแผนภูมิ

แผนภูมิวงกลม



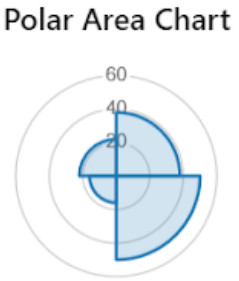
แผนภูมิวงกลม (Pie Chart) คือการแสดงผลข้อมูลทั้งหมดในลักษณะของสัดส่วนที่จะถูกแบ่งออกเป็นส่วนต่าง ๆ บนวงกลม คล้าย

กับพิซซ่าหรือพายที่ถูกแบ่งออกเป็นหลาย ๆ ส่วน ซึ่งขนาดของแต่ละส่วนนั้นก็ขึ้นอยู่กับสัดส่วนของค่าข้อมูลนั้นเมื่อเทียบกับผลรวมของค่าข้อมูลทั้งหมดที่มี ยิ่งกว้างเท่าไหร่ก็ยิ่งหมายถึงขนาดของข้อมูลนั้นมีค่าที่สูงมากขึ้นตามไปด้วยนั่นเอง

การตั้งค่าของแผนภูมิวงกลมนั้นมีดังต่อไปนี้

- **Legend** จะเป็นการเลือกว่าต้องการให้แสดงเส้นกำกับ (Show) ขึ้นมาด้านข้างของแผนภูมินี้หรือไม่ต้องการให้แสดงขึ้นมา (None)
- **Cutout** คือการขยายรูตรงกลางของแผนภูมิให้กว้างออกโดยใช้อัตราร้อยละ (%) ในการช่วยขยาย ยิ่งใช้ค่าร้อยละที่สูงขึ้น รูตรงกลางก็ใหญ่ขึ้นตามไปด้วย

แผนภูมิขั้ว



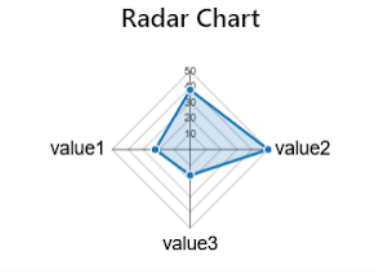
แผนภูมิขั้ว (Polar Area Chart) จะแสดงผลของข้อมูลในลักษณะก้นหอยที่จะเพิ่มปริมาณของข้อมูลขึ้นเรื่อย ๆ วนตามเข็มนาฬิกา

แผนภูมิประเภทนี้ถูกคิดค้นขึ้นในปี ค.ศ. 1855 โดยนางฟลอเรนซ์ ไนติงเกล (Florence Nightingale) นางพยาบาลชาวอังกฤษซึ่งเป็นผู้ก่อตั้ง Modern Nursing ขึ้นมา โดยเธอได้ทำการดัดแปลงแผนภูมิวงกลมที่มีอยู่แล้วดั้งเดิมมาใช้ในการวัดปริมาณของทหารที่เสียชีวิตในสงครามไครเมียในช่วงเมษายนปี 1854 ถึงเดือนมีนาคมปี 1855 จนทำให้ค้นพบว่าสาเหตุการตายส่วนใหญ่นั้นแท้จริงแล้วไม่ได้มาจากสงครามโดยตรง แต่มาจากโรคระบาดที่เกิดขึ้นในช่วงนั้นต่างหาก

การตั้งค่าของแผนภูมิขั้วนั้นมีดังต่อไปนี้

- **Y-axis** คือการกำหนดค่าสูงสุด (Max) และต่ำสุด (Min) ของข้อมูล ซึ่งจะปรากฏเป็นรัศมีที่แบ่งตามช่วงข้อมูลต่าง ๆ

แผนภูมิเรดาร์

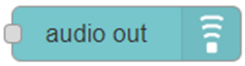


แผนภูมิเรดาร์ (Radar Chart) คือการแสดงผลข้อมูลโดยอ้างอิงจากการแสดงผลหน้าปัดของจอเรดาร์ที่ใช้ในการค้นหาสัญญาณบริเวณโดยรอบ โดยแผนภูมิประเภทนี้สามารถรับค่าตัวแปรหลายประเภทมาแสดงผลได้ภายในแผนภูมิเดียวกันในรูปแบบ 2 มิติ

การตั้งค่าของแผนภูมิเรดาร์นั้นมีดังต่อไปนี้

- **Y-axis** คือการกำหนดค่าสูงสุด (Max) และต่ำสุด (Min) ของข้อมูล ซึ่งจะปรากฏเป็นรัศมีที่แบ่งตามช่วงข้อมูลต่าง ๆ

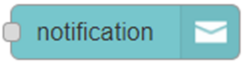
ส่งเสียงออก



ส่งเสียงออก (Audio Out) ใช้ในการเล่นเสียง หรือเปลี่ยนจากข้อความให้กลายเป็นเสียงพูดออกไป (Text to Speech; TTS) โดยรองรับการใช้งานไฟล์เสียงทั้ง .wav และ .mp3 อีกทั้งยังรองรับบทพูดหลากหลายภาษา ไม่ว่าจะเป็นฝรั่งเศส อิตาลี ญี่ปุ่น เกาหลี จีน หรือไต้หวัน เป็นต้น (แต่ปัจจุบันยังไม่รองรับการใช้งานภาษาไทย) โดยอ้างอิงจากเสียงของ Google และ Microsoft

ผู้ใช้งานสามารถเลือกให้มีการปรากฏเสียงขึ้นมาในกรณีที่หน้าต่างไม่ได้ถูกจับจ้องอยู่ได้ (Not in focus) โดยการคลิกเลือกที่ ‘*Play audio when window not in focus*’ เพื่อทำให้สามารถมีเสียงดังขึ้นได้แม้ผู้ใช้งานจะเปิดหน้าต่างอื่นอยู่ก็ตาม

การแจ้งเตือน



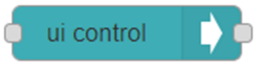
การแจ้งเตือน (Notification) จะปรากฏกล่องการแจ้งเตือนขึ้นมา หรือกล่องข้อความที่ให้ผู้ใช้งานเลือกระหว่างตกลง (OK) หรือยกเลิก (Cancel)

- **Layout** คือการกำหนดว่าต้องการให้ปรากฏกล่องการแจ้งเตือนขึ้นมาที่ตำแหน่งใดของหน้าต่างโปรแกรมหรือขึ้นมาในลักษณะใด
    - **Top Right** จะปรากฏกล่องการแจ้งเตือนที่มุมบนด้านขวาของหน้าต่าง
    - **Bottom Right** จะปรากฏกล่องการแจ้งเตือนขึ้นที่มุมล่างด้านขวาของหน้าต่าง
    - **Top Left** จะปรากฏกล่องการแจ้งเตือนที่มุมบนด้านซ้ายของหน้าต่าง
    - **Bottom Left** จะปรากฏกล่องการแจ้งเตือนที่มุมล่างด้านซ้ายของหน้าต่าง
    - **OK / Cancel Dialog** จะปรากฏเป็นกล่องข้อความที่อนุญาตให้ผู้ใช้งานสามารถกดปุ่ม ‘OK’ หรือ ‘Cancel’
    - **OK / Cancel Dialog with Input** จะปรากฏเป็นกล่องข้อความที่อนุญาตให้ผู้ใช้งานสามารถกดปุ่ม ‘OK’ หรือ ‘Cancel’ ที่อนุญาตให้ผู้ใช้งานสามารถใส่ข้อมูล

บางอย่างลงไปได้

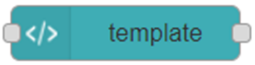
- ✔ **Default action label** ในกรณีที่ผู้ใช้งานเลือกให้แสดงผลการแจ้งเตือนแบบมีปุ่มกด OK หรือ Cancel ผู้ใช้งานสามารถกำหนดคำที่ต้องการใช้ให้ปรากฏบนปุ่มกดบนกล่องการแจ้งเตือนนั้นได้ โดยค่านี้จะมีค่าเริ่มต้นของปุ่มกดว่า 'OK' ซึ่งการกดปุ่มนี้จะเป็นการส่งค่า `msg.payload` ของโหนดนี้ออกไป โดยค่า payload ของโหนดนี้ที่จะส่งไปก็คือคำที่กำหนดเอาไว้บนปุ่มนี้นั่นเอง
- ✕ **Secondary action label** เช่นเดียวกับตั้งค่าในส่วน Default action label แต่การตั้งค่านี้ผู้ใช้งานจะใส่ค่าหรือไม่ใส่ลงไปดีได้ ซึ่งคำที่ถูกเขียนลงไปในส่วนการตั้งค่านี้จะหมายถึงการปฏิเสธค่าการแจ้งเตือนนั้น ซึ่งการกดปุ่มนี้จะเป็นการส่งค่า `msg.topic` ออกไปยังโหนดถัดไปแทน
- **Timeout (S)** คือการกำหนดเวลาในหน่วยวินาทีของการแสดงผลกล่องการแจ้งเตือนขึ้นมาบนหน้าต่างโปรแกรม
- **Border** จะเป็นการกำหนดสีขอบของกล่องการแจ้งเตือน
- **Topic** คือการตั้งชื่อหัวข้อให้กับข้อมูลที่ออกไปจากโหนดนี้ หรือเป็นข้อมูลขาออกเมื่อผู้ใช้งานทำการคลิกที่ Secondary action label

ตัวควบคุมแดชบอร์ด



ตัวควบคุมแดชบอร์ด (UI Control) ใช้ในการควบคุมเกี่ยวกับเหตุการณ์ที่เกิดขึ้นภายในหน้าแดชบอร์ด โดยค่าเริ่มต้นของโหนดนี้คือการเปลี่ยนแปลงการแสดงผลของแถบหน้าแดชบอร์ดที่ต้องการโดยอ้างอิงจาก `msg.payload` ที่มีตัวแปร `{"tab": "tab_name"}` ที่ถูกส่งมาจากโหนดก่อนหน้า ซึ่งค่าของตัวแปรที่ใส่มานี้ควรเป็นชื่อของแถบหน้าแดชบอร์ดที่ต้องการให้แสดงค่า ตัวเลขที่ระบุตำแหน่งของแถบหน้าแดชบอร์ด (เริ่มที่ 0) หรือ link ที่ต้องการนำมาแสดงผล

เท็มเพลต



เท็มเพลต (Template) อนุญาตให้ผู้ใช้งานสามารถใส่ค่า HTML หรือ Angular ลงไปได้ โดยจะเป็นการสร้างการแสดงผลที่จะเปลี่ยนแปลงในทันทีตามข้อความที่เข้ามาภายในโหนด เช่น เปลี่ยน