คู่มือการใช้งาน

Octopus Smart Box Node

บริษัท อาร์ แอนด์ ดี คอมพิวเตอร์ ซิสเท็ม จำกัด

SmartBoxNodeManual_R240926

สารบัญ

เริ่มต้นใช้งานโปรแกรม **1**

แนะนำโปรแกรม Octopus Smart Box Node **1**

เกี่ยวกับ Node-RED 1

การติดตั้งโปรแกรม SmartBoxNode ลงใน Node-RED 1

การใช้งาน Octopus IO **3**

การเพิ่มกล่องใหม่เข้ามาในรายการ 3

โหนด Pin In **4**

Digital pin 5

Analog pin 6

Rate pin 6

โหนด Pin Counter **6**

โหนด Pin Out **7**

Digital Out 7

PWM Out 7

โหนด Box Status **8**

การใช้งาน Octopus Modbus **9**

การใช้งาน Octopus Smart Box ในโหมด Modbus Slave **9**

การใช้งาน Octopus Smart Box ในโหมด Modbus Master RTU **10**

โหนด MB-Master Read **11**

โหนด MB-Master Write **12**

ตัวอย่างการทดสอบ MB-Master Write 12

ใช้ Octopus Smart Box ต่อเป็น Slave ของตัว Master RTU 13

การใช้งาน Octopus Database **14**

Database แบบ Local 14

โหนด Write DB **14**

ตัวอย่างการใช้โหนด Write DB 14

โหนด Read DB **17**

การอ่านเฉพาะข้อมูลล่าสุด 17

การอ่านแบบกำหนดช่วงเวลา 17

โหนด Http-read DB **19**

โหนด Gen latest period **20**

ปัญหาและการแก้ไข **22**

เริ่มต้นใช้งานโปรแกรม

แนะนำโปรแกรม Octopus Smart Box Node

โปรแกรม Octopus Smart Box Node หรือเรียกย่อว่า SmartBoxNode เป็น Node Library สำหรับติดตั้งใน Node-RED เพื่อให้สามารถใช้งาน Octopus Smart Box ได้ โดยโปรแกรมจะช่วยในการอ่านข้อมูลจากพอร์ตต่างๆ และสามารถสั่งพอร์ตให้ทำงานตามต้องการได้

เกี่ยวกับ Node-RED

Node-RED เป็นเครื่องมือในการพัฒนาโปรแกรมที่เขียนอยู่บน Node.js ซึ่งเป็นภาษา Java Script และทำงานอยู่บน Web Browser โปรแกรมที่สร้างโดย Node-RED จะอยู่ในรูปของ Flow ที่ประกอบจากโหนดต่างๆที่นำมาเชื่อมต่อกันด้วยเส้น สามารถศึกษาวิธีการเขียน Node-RED ได้จากเว็บไซต์ของ Node-RED (nodered.org)

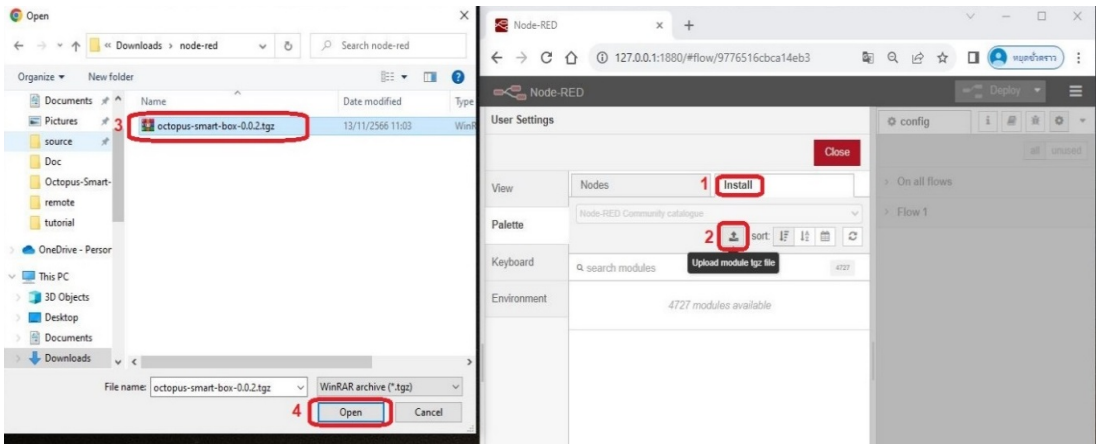
สำหรับการใช้งาน Octopus Smart Box Node บน Node-RED นั้น โดยทั่วไปสามารถวางโหนดและเชื่อมต่อกับโหนดอื่นๆของ Node-RED ได้โดยตรง แต่ในกรณีที่ต้องมีการแปลงข้อมูล จึงจำเป็นต้องเขียน function โดยใช้โหนด function ซึ่งผู้ใช้ควรมีพื้นฐานความรู้ในการเขียนภาษา JavaScript และเข้าใจวิธีการเขียน function ของ Node-RED ด้วย ซึ่งมีตัวอย่างให้ศึกษามากพอควร

- ***Octopus Smart Box Node นี้ ใช้ได้กับ Node-RED ตั้งแต่รุ่น 3.1.3 เป็นต้นไป***

การติดตั้งโปรแกรม SmartBoxNode ลงใน Node-RED

การติดตั้งทำดังนี้

1. ดาวโหลดแฟ้ม Node Module ในรูปของ octopus-smart-box-node-x.xx.xx.tgz (อาจอยู่ในแฟ้ม .zip ก็แตกแฟ้มลงมาในเครื่องคอมพิวเตอร์
2. เพิ่ม Node Module เข้ากับ Node-RED ทำโดยการ เปิดโปรแกรม Node-RED แล้วเลือก menu (ที่มุมขวาบน) แล้วเลือกเมนู Manage palette



3. คลิกที่ Tab Install แล้วคลิกปุ่ม Upload แล้วทำการเลือกแฟ้ม octopus-smart-box-node-x.xx.xx.tgz ที่ดาวน์โหลดมาไว้ เสร็จแล้วจะมีปุ่ม Upload ปรากฏ ให้คลิกที่ปุ่มนี้ แล้วทำการติดตั้งจนเสร็จสิ้น จะมี

ข้อความด้านบนแจ้งว่าได้ทำการเพิ่ม Node ใหม่เข้ามา

4. เมื่อดูที่ด้านซ้ายของ Node-RED จะปรากฏ node ที่เราติดตั้งเข้าไปอยู่ในกลุ่ม Octopus ต่างๆ ได้แก่ Octopus IO, Octopus Modbus และ Octopus Database

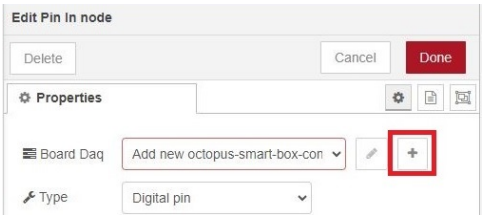
การใช้งาน Octopus IO

Octopus IO เป็นกลุ่มโหนดพื้นฐานสำหรับการใช้งาน Octopus Input และ Output ซึ่งประกอบด้วยโหนดต่าง ๆ เช่น Pin In, Pin Out, Pin Counter, และ Box Status โดยโหนดเหล่านี้ถูกจัดตามลักษณะทิศทางการไหลของข้อมูล ไม่ได้แยกตามชนิดของข้อมูลว่าเป็น Analog หรือ Digital ดังนั้น ผู้ใช้ต้องเลือกชนิดของข้อมูลย่อยภายในตอนที่ทำการ Edit Node อีกครั้ง

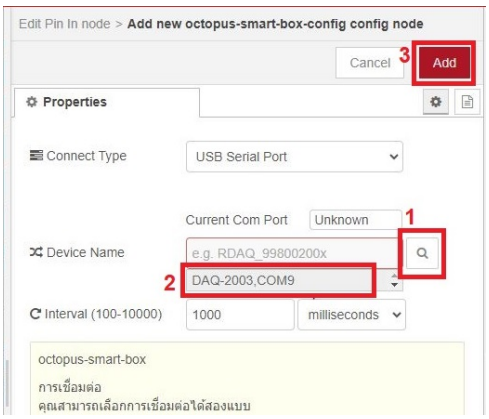
สิ่งที่โหนดพื้นฐานใน **Octopus IO** มีเหมือนกันคือ การเลือกกล่อง **Octopus Smart Box** เข้ามาในรายการ โดยโหนดที่ตั้งค่าครั้งแรกจะต้องทำการค้นหาอุปกรณ์ก่อน เมื่อพบแล้วจึงเพิ่มอุปกรณ์เข้าสู่รายการ จากนั้นโหนดถัดไปสามารถเลือก **Smart Box** จากรายการที่มีอยู่โดยไม่ต้องทำการค้นหาใหม่

การเพิ่มกล่องใหม่เข้ามาในรายการ

ขอยกตัวอย่างการเพิ่มกล่องเข้าในรายการ เช่นกรณีใช้ Pin In ในหน้า Edit จะเห็นตัวเลือก Board Daq ดังรูป

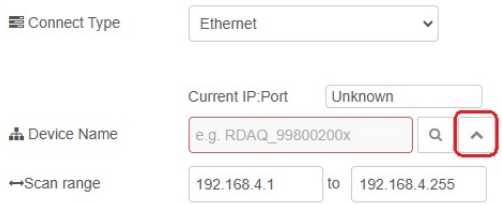


โดยกดปุ่ม '+'เพื่อเพิ่มอุปกรณ์ จะพบหน้าจอดังรูป

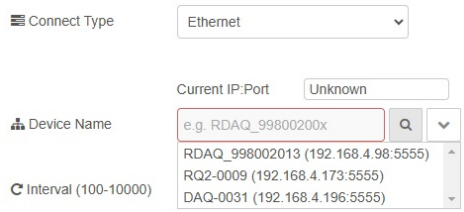


ในการเชื่อมต่อกับ Octopus Smart Box ผู้ใช้สามารถเลือกวิธีการเชื่อมต่อได้ 2 แบบคือ USB Serial Port หรือ Ethernet ในกรณีเลือกค้นหาแบบ USB ไม่มีตัวเลือกอื่นเพิ่ม สามารถกดปุ่มค้นหา(รูปแว่นขยาย) ตามรูปด้านบน เมื่อพบอุปกรณ์แล้ว จะแสดงรายการเป็นชื่อ Device Name และหมายเลข Com port ที่พบ

สำหรับการค้นหาแบบ Ethernet ผู้ใช้ควรระบุ Scan Range เพื่อให้โปรแกรมรู้ว่าต้องค้นหาในช่วง IP ไหน โดยเมนูนี้จะถูกซ่อนไว้ ผู้ใช้ต้องกดปุ่มขยายเมนูที่อยู่ถัดจากปุ่มแว่นขยาย เมื่อขยายแล้วจะพบหัวข้อ Scan Range ดังรูป ให้ระบุช่วงของ IP Address เริ่มต้นและสิ้นสุด เพื่อกำหนดขอบเขตในการค้นหา จากนั้นกดปุ่มแว่นขยายเพื่อเริ่มการค้นหา



เมื่อพบแล้วจะแสดงรายการทั้งหมดที่พบ เป็นชื่อ และหมายเลข IP ตามด้วยหมายเลข Port (TCP Port ที่ใช้ส่งคำสั่ง) ดังรูป



การตั้งค่า Interval

ค่า Interval หมายถึง ความเร็วในการอ่านข้อมูลจากกล่อง DAQ หรือความเร็วในการรีเฟรชค่าข้อมูล ซึ่งค่าเริ่มต้นถูกกำหนดไว้ที่ 1000 มิลลิวินาที (ms) ต่อครั้ง ตั้งค่าได้ต่ำสุดคือ 100 มิลลิวินาที (ms) ผู้ใช้สามารถปรับค่า Interval ให้เหมาะสมกับลักษณะงานที่ต้องการได้

ความเร็วในการอ่านนี้ จะใช้สำหรับทุกโหนด Pin In ที่ใช้ DAQ กล่องเดียวกัน

ต่อไปจะเป็นการอธิบายการใช้งานโหนดต่างๆของโปรแกรม ดังนี้

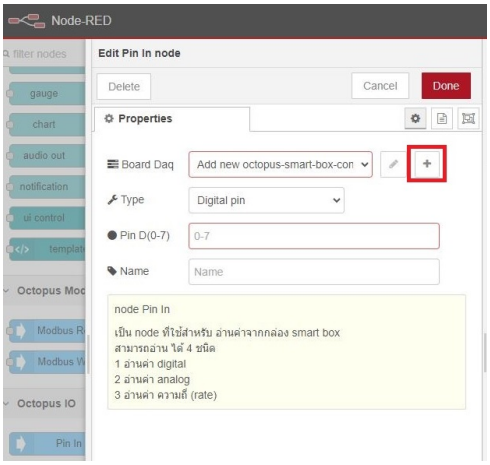
โหนด Pin In

เป็นโหนดสำหรับการอ่าน Input ชนิดต่างๆ ของ Octopus Smart Box โดยสามารถเลือกชนิดหรือ Type ได้แก่ Digital, Analog และ Rate ส่วน Counter จะแยกไปอยู่ใน Node Counter เพราะมีการตั้งค่าได้ด้วย

การใช้งาน Node และการตั้งค่าทำเหมือน Node อื่นๆของ Node-RED คือวางบน flow แล้วดับเบิลคลิกเพื่อเข้าสู่หน้า Edit

Digital pin

Digital Pin ใช้สำหรับอ่านพอร์ต Digital IO ในโหมด Input จากหน้า Edit Pin In ตั้งค่าได้ดังนี้



หัวข้อ Board Daq เลือกกล่อง DAQ ตามที่ได้กล่าวมาแล้ว

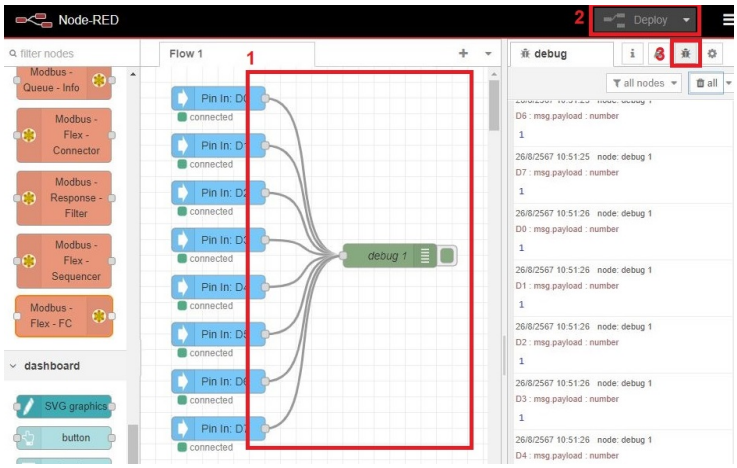
Type เลือกเป็น Digital Pin

Pin D(0-7) เลือกพอร์ตที่ต้องการ

Name ใช้สำหรับตั้งเป็นชื่อที่ต้องการ หากไม่ตั้งก็จะใช้ชื่อ Pin In แทน

กด Done เมื่อ Edit Pin เสร็จสิ้น

Flow ตัวอย่างทดลองเพิ่ม Node Digital In D0-D7 แล้วต่อเข้า Node Debug เสร็จแล้วสั่ง Deploy เพื่อให้ทำงาน แล้วให้เปิดหน้า Debug เพื่อดูผลลัพธ์ ได้ดังรูป



เมื่อแต่ละ node เชื่อมต่อกับกล่อง Smart Box ได้สำเร็จ จะมีการแสดงสถานะ Connected ที่ด้านล่างของ node นั้น ๆ เพื่อให้ผู้ใช้สามารถตรวจสอบสถานะการเชื่อมต่อได้อย่างชัดเจน

- ***ค่า Output ที่ออกมาจากโหนด Digital Input นี้จะมีค่า 0, 1***
- ***ใน Node-RED เมื่อผู้ใช้สั่ง Deploy ให้ flow ทำงานแล้ว จะเห็นไฟสถานะของ Smart Box กระพริบสีฟ้าถี่ๆ ตลอดเวลา เป็นเพราะว่า SmartBox มีการสแกนอ่านข้อมูลอย่างต่อเนื่อง ผ่านทางโหนด Pin In***

Analog pin

สำหรับการอ่านค่าจากพอร์ต Analog In ใน Octopus IO ผู้ใช้ต้องตั้งค่าจาก node Pin In โดยให้เลือก Type เป็น Analog pin ส่วน Board Daq ให้เลือกเหมือนเดิม และในส่วน Pin A (0-7) ให้เลือกพอร์ต A0 ถึง A7 ตามต้องการ

ผู้ใช้สามารถทดลองเชื่อมต่อกับ **node debug** ตามตัวอย่างที่แล้ว จากนั้นให้ลองสั่ง **Deploy** เพื่ออ่านค่าและดูผลลัพธ์ได้ ดังรูป

- ***ค่า Output ที่ออกมาจากโหนด Analog pin นี้จะเป็นแรงดันตัวเลขทศนิยมค่าระหว่าง -10.000 ถึง 10.000 หน่วยเป็น Volt***

Rate pin

สำหรับอ่านความถี่ จากช่อง D2 และ D3 ผู้ใช้สามารถอ่านค่าความถี่ได้ตั้งแต่ 1 ถึง 60000 Hz การตั้งค่าทำได้เช่นเดียวกับการอ่านค่าก่อนหน้านี้ คือตั้ง Type เป็น Rate pin และเลือก Pin ที่ต้องการ ทดลองเชื่อมต่อกับ node debug เพื่อดูค่าได้เช่นกัน

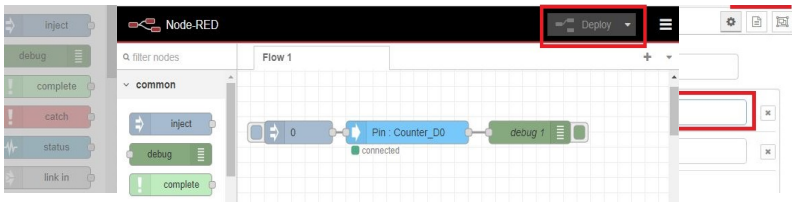
- ***ค่า Output ที่ออกมาจากโหนด Rate pin นี้คือค่าความถี่ เป็นตัวเลข 1-60000 หน่วยเป็น Hz***

โหนด Pin Counter

เป็น node ที่ใช้ในการอ่านค่า Counter ที่นับพัลส์จากภายนอก ที่ต่อกับช่อง Count ตรง D0 หรือ D1 สำหรับ Pin Counter นี้จะต่างจาก Pin In ตรงที่มี Interface ด้าน Input ด้วย เพื่อช่วยในการตั้งค่าเริ่มต้นในการนับได้

ตัวอย่างทดลองการใช้ Counter ทำดังนี้

1. เพิ่ม node Pin Counter ลงมาใน flow แล้วตั้งค่า Board Daq และ Pin ที่ต้องการ
2. เพิ่ม inject ลงมาใน flow แล้วแก้ type payload เป็น number ใส่เลข 0 เพื่อที่จะให้เมื่อกดปุ่มที่ inject จะ set ค่า Counter กลับเป็น 0 ใหม่



3. เชื่อม inject ไปยัง Pin Counter และต่อ debug แล้วกด Deploy

โหนด Pin Out

สำหรับสั่ง Octopus Smart Box ส่งสัญญาณออกมา มี node ย่อยประกอบด้วย Digital และ PWM

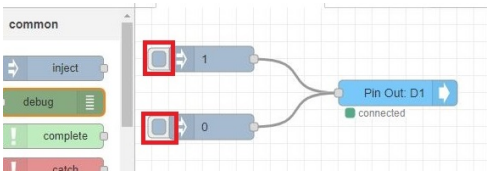
Digital Out

เป็นเอาต์พุต ที่ส่งสัญญาณออกไปทางพอร์ต Digital D0-7 ให้เป็น Low หรือ High ผู้ใช้สามารถตั้งค่าได้โดยทำตามขั้นตอนเดียวกับการตั้งค่าก่อนหน้านี้

- ***การสั่งพอร์ต Digital ให้เป็น Digital Out นั้น สามารถทำได้ทันที โดยไม่จำเป็นต้อง Config พอร์ตให้เป็น Output ไว้ก่อน***
- ***โหนด Digital Out นี้จะรับค่า Input 0 และ 1***

การทดสอบการทำงาน ให้ต่อโหลดเข้ากับ Octopus Smart Box พอร์ตที่ต้องการ แล้วตั้งค่า flow ดังนี้

1. วาง node Pin Out ลงใน flow เลือกกล่อง Daq แล้วตั้งให้ Type เป็น Digital Out (0/1)
2. เพิ่ม node inject ลงไป 2 ตัว โดยตั้งค่า node แรกให้ส่งค่า number เป็น 1 และตั้งค่า node ที่สองให้ส่งค่า 0 จากนั้นให้เชื่อมต่อกับ node Pin Out ที่ได้วางไว้แล้ว สุดท้ายให้สั่ง Deploy



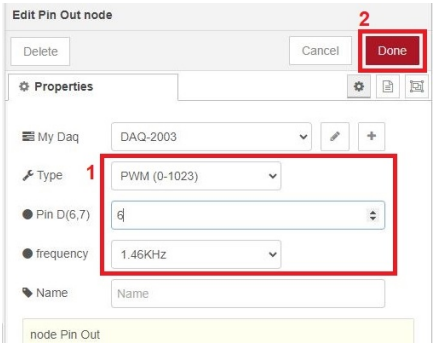
3. รอจนทำงานเห็นสถานะ connected ที่ใต้ Pin Out แล้วทดลอง กดส่ง 0 และ 1 เพื่อดูว่า Digital พอร์ตนั้นเป็น Low หรือ High หรือไม่

PWM Out

PWM หรือ Pulse Width Modulation คือสัญญาณความถี่ที่เราควบคุมความกว้างหรือค่า duty cycle ได้ ซึ่งมักนำไปใช้ในงานควบคุมต่างๆ เช่น การควบคุมความเร็วมอเตอร์ เป็นต้น

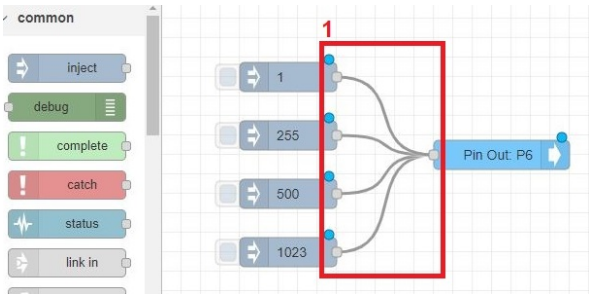
สำหรับ Octopus Smart Box มีพอร์ต PWM ให้ 2 ช่อง ตรงกับ D6 และ D7 สามารถรับค่าได้ 0-1023 โดยค่า 0 คือ duty cycle เป็น 0% ค่า 1023 คือ duty cycle เป็น 100%

เมื่อวาง node Pin Out แล้ว ให้เลือก Type เป็น PWM จะปรากฏตัวเลือก ดังรูป



จากรูปนอกจากเลือก Pin แล้ว ผู้ใช้ยังสามารถเลือกความถี่ของสัญญาณได้อีกด้วย ซึ่งมีได้ 4 ระดับคือ 366Hz ถึง 23.4kHz ให้ผู้ใช้เลือกตามความเหมาะสมกับอุปกรณ์ที่จะนำไปใช้งาน

จากนั้นให้เพิ่ม inject เพื่อใช้ส่งค่าไปยัง interface Input ของ Pin Out ดังรูป



จากรูป ได้ตั้ง inject สำหรับส่งค่า pwm ไว้ 4 ค่า คือ 1, 255, 500, 1023 โดยให้ผู้ใช้ทดลองสั่ง deploy และลองส่งค่าด้วย inject แต่ละตัวเพื่อตรวจสอบการทำงาน

- ***โหนด PWM Out นี้จะรับค่า Input 0-1023***

โหนด Box Status

เป็น node ที่ใช้สำหรับอ่านสถานะของกล่อง Octopus Smart Box ว่า เชื่อมต่อกับ Node RED หรือไม่ โดยจะมีสถานะออกมาเป็น string 3 สถานะคือ connecting, connected และ disconnect ซึ่งสามารถนำไปใช้แสดงผลที่ dashboard ได้

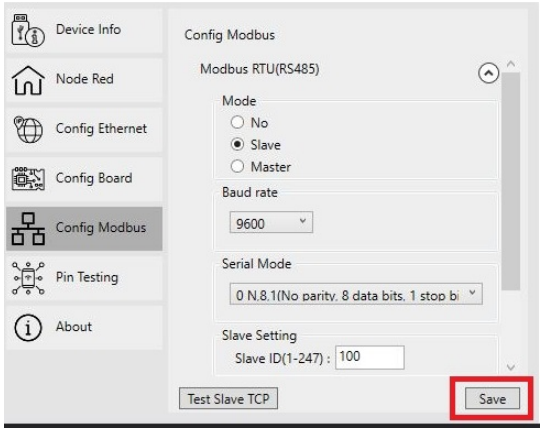
การใช้งาน Octopus Modbus

การนำ Octopus Smart Box ไปใช้งานด้วยการสื่อสารแบบ Modbus นั้น ทำได้ 2 โหมด คือ โหมด Master และโหมด Slave

การใช้งาน Octopus Smart Box ในโหมด Modbus Slave

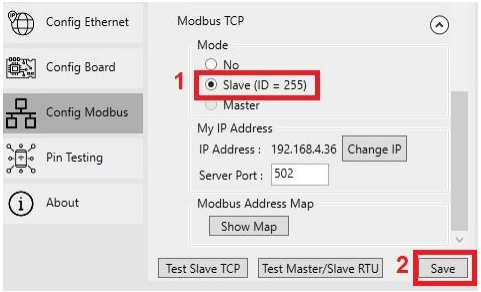
ในโหมด Modbus Slave ของ Octopus Smart Box ยังแยกเป็น 2 กรณี คือ กรณีสื่อสารผ่าน RS485 เรียกว่า Modbus RTU และผ่าน Ethernet TCP เรียกว่า Modbus TCP

ในการนำไปใช้งาน จะต้องทำการ config อุปกรณ์ก่อน โดยใช้เมนู Config Modbus จากโปรแกรม Octopus Smart Box Manager เพื่อทำการตั้งค่าให้เรียบร้อย ก่อนเริ่มใช้งาน ตามที่แสดงในรูป



ในกรณีของ Slave RTU การตั้งค่าจะทำโดยการกำหนด Mode เป็น Slave, Baud Rate, และ Serial Mode ตามที่ต้องการใช้งาน นอกจากนี้ ยังต้องกำหนด Slave ID ซึ่งเป็น ID สำหรับ Modbus Slave แต่ละตัว โดยต้องมั่นใจว่า Slave ID เหล่านั้นไม่ซ้ำกัน

สำหรับการใช้งาน TCP Slave เพียงกำหนด Mode ให้เป็น Slave (ID=255) คือ ID จะต้องเป็น 255 จึงจะติดต่อกับกล่องได้ ส่วน Port ใช้ค่ามาตรฐาน 502 หรือสามารถกำหนดค่าตามที่ต้องการได้ ตามที่แสดงในรูป

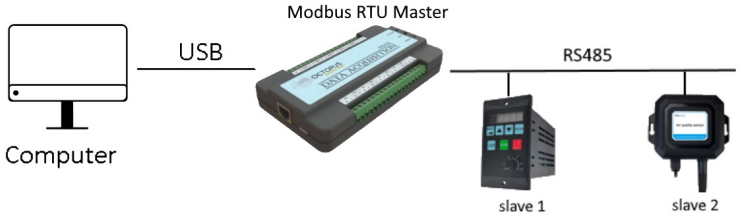


ในการนำ Octopus Smart Box ไปใช้งานในโหมด Modbus Slave แนะนำให้ใช้ node-red-contrib-modbus ซึ่งเป็น package ที่นิยมใช้กันทั่วไปสำหรับ Modbus บน Node-RED และสามารถสื่อสารกับ Octopus Smart Box โหมด Slave ได้ทันที สำหรับการติดตั้งให้ทำตามขั้นตอนเดียวกันกับการติดตั้ง Octopus Smart Box Node โดยไปที่เมนู Manage palette แล้วค้นหาและ Install package นี้

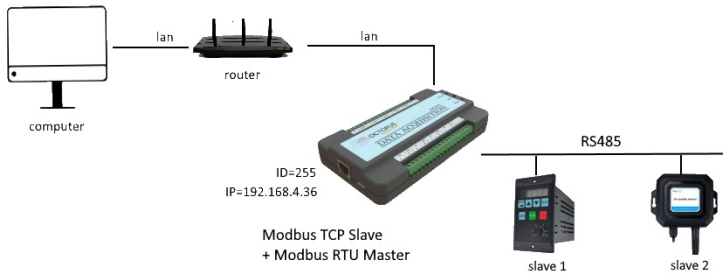
การใช้งาน Octopus Smart Box ในโหมด Modbus Master RTU

การใช้งานในโหมดนี้ คือการให้ Octopus Smart Box เป็นตัวกลางในการเชื่อมต่อระหว่างคอมพิวเตอร์ กับ Slave Device ซึ่งเชื่อมต่อโดยผ่านทาง Modbus RTU (RS485) โดย Octopus Smart Box จะทำหน้าที่เป็น Master ในการติดต่อสื่อสารกับ Slave Device ให้แทนเครื่องคอมพิวเตอร์ มีลักษณะดังรูป

กรณีสั่งผ่าน USB



กรณีสั่งผ่าน Ethernet

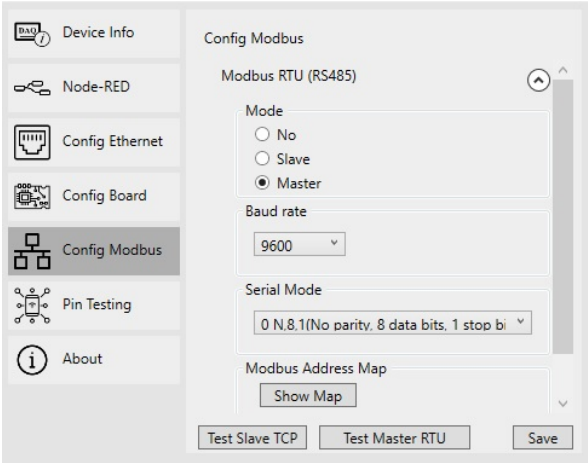


การทำงานของโหมดนี้ ในการขอข้อมูลของคอมพิวเตอร์ ไปยัง Slave Device เริ่มจาก

1. คอมพิวเตอร์ส่งคำสั่งมายังตัว Smart Box ผ่าน USB หรือ Ethernet
2. Smart Box นำส่งคำสั่งนั้นออกไปทาง Modbus RTU ที่ตั้งเป็น Master ไว้
3. เมื่อ Slave Device ได้รับคำสั่ง ทำการส่งข้อมูลกลับไปยัง Master ซึ่งคือ Smart Box
4. เมื่อ Smart Box ได้รับข้อมูล จะส่งข้อมูลกลับไปยังเครื่องคอมพิวเตอร์

จะเห็นว่าการทำงานในลักษณะนี้ เป็นการทำงานในการส่งต่อคำสั่งให้ โดย Smart Box จะทำงานเหมือนเป็น 2 โหมดพร้อมกัน คือเป็น Slave รับคำสั่งจากคอมพิวเตอร์ และเป็น Master ส่งคำสั่งให้กับตัว Device

ก่อนการใช้งานในโหมด Modbus Master RTU นี้ จะต้องตั้ง config ด้วยโปรแกรม Smart Box Manager ก่อน เพื่อให้ Modbus RTU (RS485) ทำงานเป็น Master ดังรูป

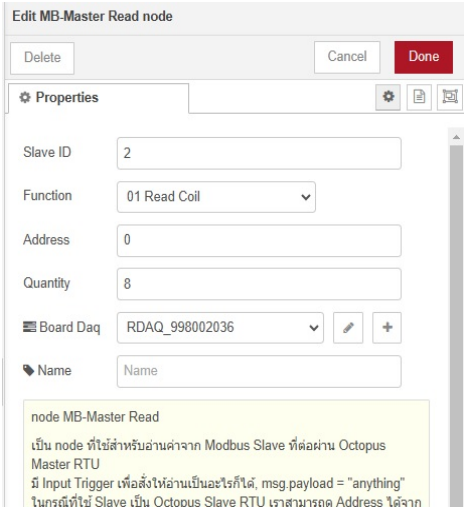


สำหรับ Slave Device นั้น ยังสามารถใช้ตัว Octopus Smart Box ตั้งเป็นโหมด Slave แล้วนำมาต่อเข้ากับตัว Master ทางพอร์ต RS485 เพื่อใช้งานได้เช่นกัน

ในการใช้งาน Modbus โหมด Master RTU นี้กับโปรแกรม Node-RED ให้ใช้โหนด MB-Master Read และ MB-Master Write ในการทำงาน

โหนด MB-Master Read

เป็นโหนดสำหรับการอ่านข้อมูล Slave Device ด้วย Modbus Master RTU การใช้งานทำโดยตั้งค่าในหน้า Edit Modbus Read ดังรูป

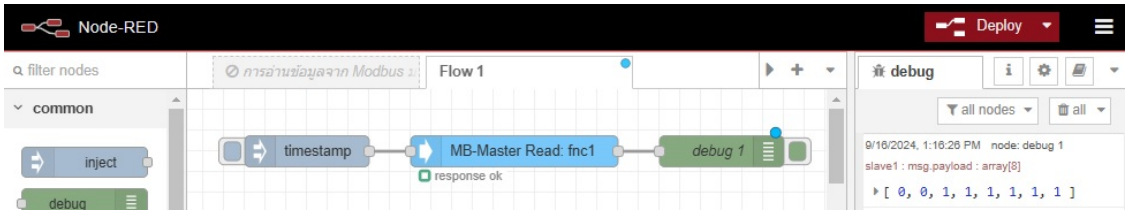


Slave ID คือ ID ของ Modbus Slave ที่นำมาต่อ

Function, Address, Quantity คือฟังก์ชั่นของ Modbus, Address เริ่มต้น และจำนวนที่ต้องการอ่าน

Board Daq เพื่อเลือกกล่อง Octopus Smart Box มาใช้งาน เหมือนตอนเลือก Board Daq ของโหนด Pin In

สำหรับ Input เป็น Trigger ที่จะสั่งให้อ่าน เป็น payload อะไรก็ได้ ตัวอย่างเช่นใช้ Inject ที่มีปุ่ม นำมาต่อเป็นดังรูป



โหนด MB-Master Write

เป็น node สำหรับการเขียนไปยัง Slave Device ด้วย Modbus Master RTU การใช้งานสามารถทำได้โดยตั้งค่าในลักษณะเดียวกับ Edit Modbus Read

ในการใช้ Modbus Write เพื่อส่งค่าไปยัง Slave Device นั้น กรณีส่งข้อมูลแบบ Single Coil หรือ Single Register ให้ส่งค่าในรูป integer หรือ number ได้เลย ส่วนหากเป็นการส่งแบบ Multiple ค่าที่รับได้จะเป็น Array โดย function ที่ใช้งานได้ มีดังนี้

Fnc 5 Write Single Coil ใช้สำหรับการเขียน 1 coil ค่าที่รับได้คือ 0 หรือ 1

Fnc 6 Write Single Register ใช้สำหรับการเขียน 1 register ค่าที่รับได้เป็นตัวเลข เช่น Address 300 สำหรับ PWM0 (D6) ค่าที่รับได้คือ 0-1023 เป็นต้น

Fnc 15 Write Multiple Coils ใช้สำหรับการเขียนหลาย coil โดยรับข้อมูลเป็น Array ที่ประกอบด้วย 0 และ 1 ตัวอย่างเช่น

msg.payload = [0,0,0,0,1,1,1,1];

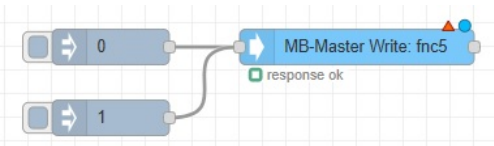
- ***ในการส่งผ่านข้อมูลเข้าออกโหนดของ Node-RED จะทำโดยผ่าน object ชื่อ msg โดยจะมี property ชื่อ payload ในการรับเข้าและส่งออก***

Fnc 16 Write Multiple Registers ใช้สำหรับการเขียนหลาย register โดยรับข้อมูลเป็น Array ของตัวเลขในทำนองเดียวกัน ตัวอย่างเช่น เขียน 2 register ส่งข้อมูลเป็น Array

msg.payload = [1023,999];

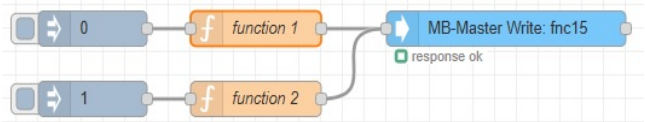
ตัวอย่างการทดสอบ MB-Master Write

1) ใช้ Fnc 5 Write Single Coil ทำการต่อ node ดังรูป

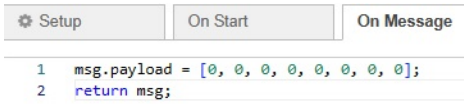


กรณีนี้สามารถส่งค่าจาก Trigger ที่ตั้งเป็น Number สำหรับส่งค่า 0 และ 1 ไปยังโหนด MB-Master Write ได้เลย

2) ใช้ Fnc 15 Write Multiple Coils ทำการต่อ node ดังรูป



ทำการ edit node function เพื่อทำการส่งค่าเป็น Array 8 ช่องสำหรับเขียน coil 8 ตัว ดังรูป



function 1 รับ Trigger จากโหนด inject ‘0’ ให้ทำการส่งค่า 0 ทั้ง 8 coil โดยสั่ง msg.payload = [0,0,0,0,0,0,0,0];

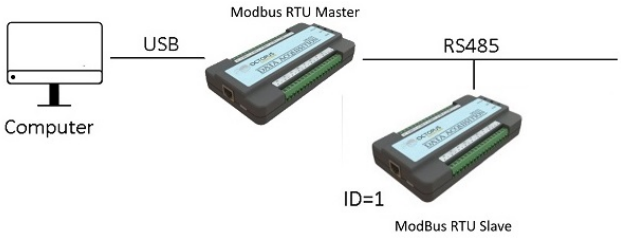
ส่วน function 2 รับจากโหนด inject ‘1’ ให้ทำการส่งค่า 1 ทั้ง 8 coil โดยแก้เป็น msg.payload = [1,1,1,1,1,1,1,1];

เสร็จแล้วสั่ง Deploy แล้วกดปุ่มที่โหนด inject ทั้ง 2 ตัว ดูค่าผลลัพธ์ที่ coil ทั้ง 8

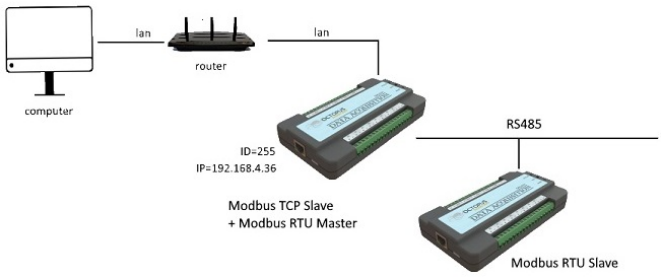
ใช้ Octopus Smart Box ต่อเป็น Slave ของตัว Master RTU

การใช้งานโหมด Master RTU นั้น ยังสามารถใช้ Octopus Smart Box อีกตัวทำหน้าที่เป็น Slave ได้เช่นกัน ซึ่งมีลักษณะการต่อดังรูป

รูปกรณีต่อจาก USB



รูปกรณีต่อจาก Ethernet



การตั้งค่า Smart Box ตัวที่เป็น Slave ทำโดยใช้โปรแกรม Smart Box Manager ตั้งค่าโหมด Modbus RTU (RS485) ให้เป็น Slave แล้วตั้งค่า ID, baudrate ให้เรียบร้อยก่อนนำมาใช้งาน

การใช้งาน Octopus Database

เป็นกลุ่ม node ที่เกี่ยวกับการเขียนอ่าน Database โดย Database จะอยู่ในเครื่องคอมพิวเตอร์แบบ Local หรืออยู่ในเว็บเซอร์เวอร์บน cloud ก็ได้ โดยในเอกสารนี้จะยังไม่กล่าวถึง Database บน cloud กล่าวถึงเฉพาะแบบ Local ก่อน

Database แบบ Local

การเก็บ Database แบบ Local จะเก็บอยู่ภายในเครื่องคอมพิวเตอร์ที่รัน Node-RED โดย Database จะใช้ Sqlite เป็นตัวจัดเก็บ ซึ่งถูกติดตั้งให้โดยอัตโนมัติ ในช่วงติดตั้ง Octopus Smart Box Node

สำหรับการเก็บ Database นั้น จะถูกเก็บอยู่ในรูปของไฟล์ อยู่ในโฟลเดอร์ที่กำหนดได้ จากโหนด Write DB ที่จะกล่าวถึงต่อไป

โหนด Write DB

เป็นโหนดที่จะนำข้อมูลที่รวบรวมไว้เพื่อจะเขียนลง Database ข้อมูลที่จะเขียน จะถูกจัดเตรียมอยู่ในลักษณะของ JSON มีลักษณะเป็น topic1 : value1, topic2 : value2 ไปเรื่อยๆ โดย topic เป็นชื่อของข้อมูล มีดังนี้

ts คือ timestamp

A0 ถึง A7 คือข้อมูล Analog In 0-7

D0 ถึง D7 คือข้อมูล Digital In 0-7

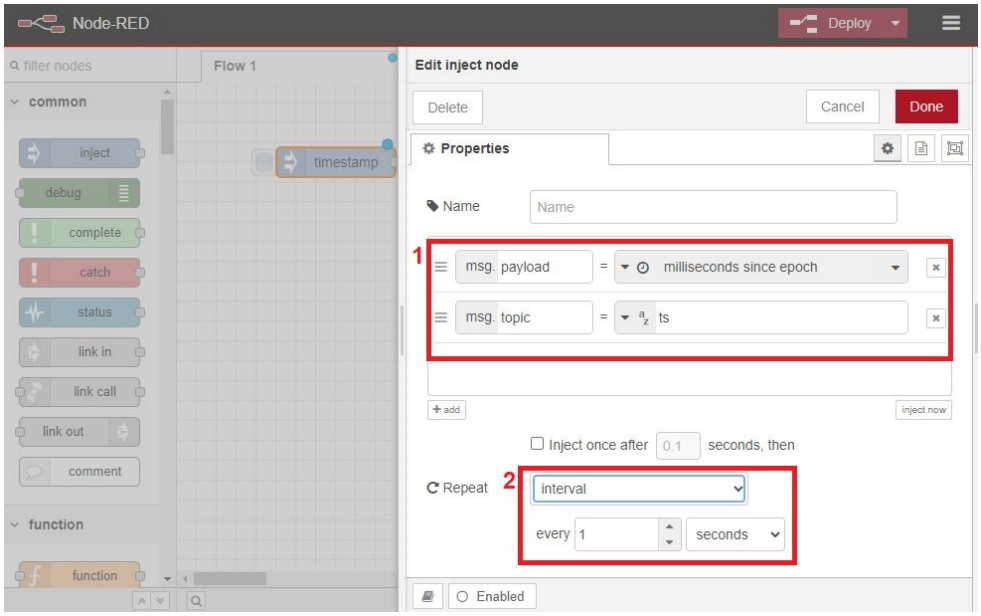
C0, C1 คือข้อมูล Counter 0 และ 1

R0, R1 คือข้อมูล Rate 0 และ 1

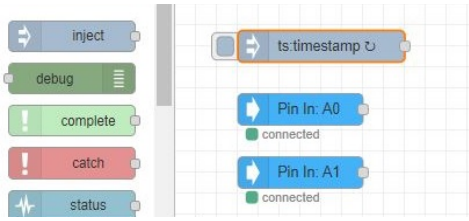
ตัวอย่างการใช้โหนด Write DB

ตัวอย่างต่อไปนี้จะแสดงการเก็บข้อมูล timestamp,A0,A1 ทำดังนี้

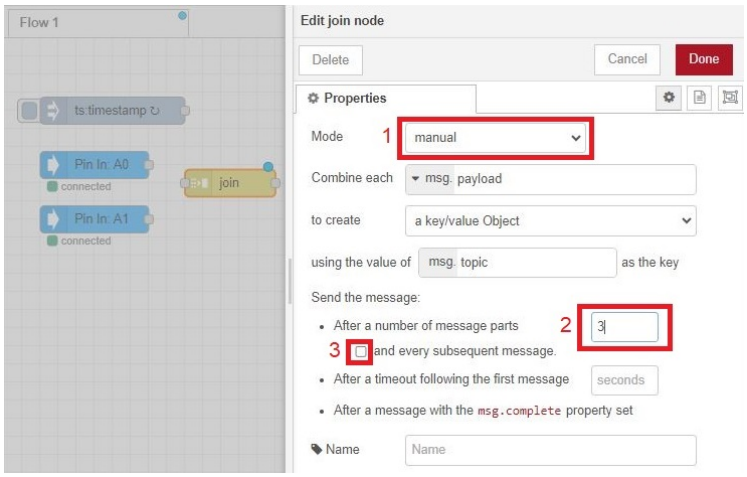
ตั้งค่า timestamp โดยวาง node inject ใน (จากกลุ่ม common) ลงมาใน flow ให้ตั้งค่า payload เป็น timestamp แล้วเลือกเป็นชนิด milliseconds since epoch และตั้งค่า topic เป็น ts ส่วนค่า Repeat คือความถี่ในการอ่าน interval every 1 second ดังรูป



เตรียม pin ที่ต้องการบันทึก โดยวางโหนด Pin In ลงมาใน flow 2 โหนด แล้วตั้งค่าโหนดที่ 1 เป็น type Analog และ pin=1 ส่วนโหนดที่ 2 เป็น type Analog และ pin=2 ตั้งค่า Board Daq ของทั้งสองให้เรียบร้อย ใน Board Daq ให้ตั้ง interval ให้ไม่ช้าเกินไป แล้วสั่ง deploy ถ้าโหนดทั้ง 2 แสดงสถานะ connected แสดงว่าเชื่อมต่อกับ กล่อง Octopus Smart Box สำเร็จดังรูป



เสร็จแล้วทำการวางโหนด join เพื่อเชื่อมข้อมูลทั้ง 3 โหนดเข้าด้วยกัน แล้วตั้งค่าดังรูป



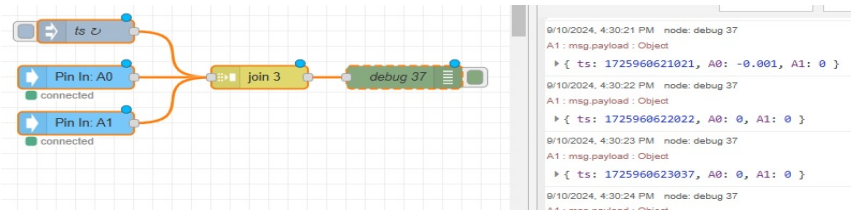
1) ตั้งค่า Mode เป็น manual

2) ตั้งค่า After a number of message parts เป็นจำนวน 3 ประกอบ ts, A0, A1

3) and every subsequent message ต้องแก้ไขเป็นไม่เลือก (จากค่าเริ่มต้นตั้งเลือกไว้) เพื่อไม่ให้ส่งข้อมูลซ้อนกัน

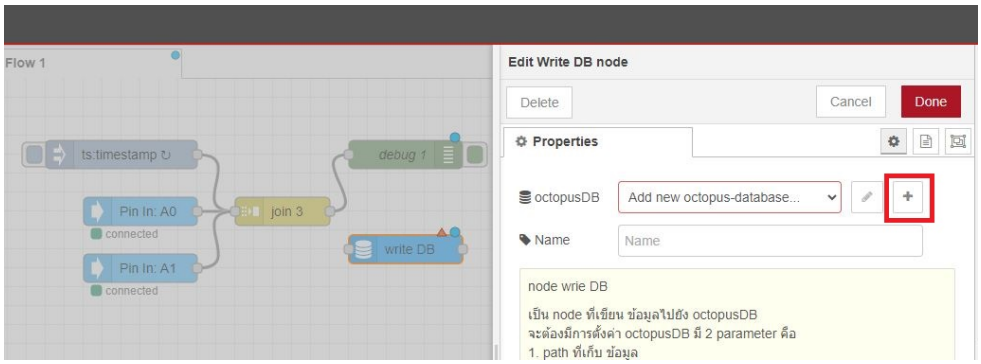
➤ ***โหนด join ทำงานโดยการรวบรวมข้อมูล จนครบตามจำนวนที่เราตั้งตามข้อ 2 เมื่อครบแล้วจึงจะรวมข้อมูลแล้วส่งออกไป หากข้อมูลบางโหนดขาด จะไม่ส่งข้อมูลต่อเลย หากเราต้องการให้กรณีข้อมูลขาด ก็ยังส่งที่เหลือต่อไป ทำโดยตั้งที่หัวข้อ "After a timeout following the first message" กำหนดเวลารอข้อมูล หากไม่มา ก็จะส่งข้อมูลที่เหลือไป โดยข้อมูลที่ขาด จะเห็นเป็น NUL คือข้อมูลว่าง***

หลังจากนั้น ให้ลองต่อเข้าโหนด debug แล้วสั่ง deploy เพื่อดูผลลัพธ์การ join

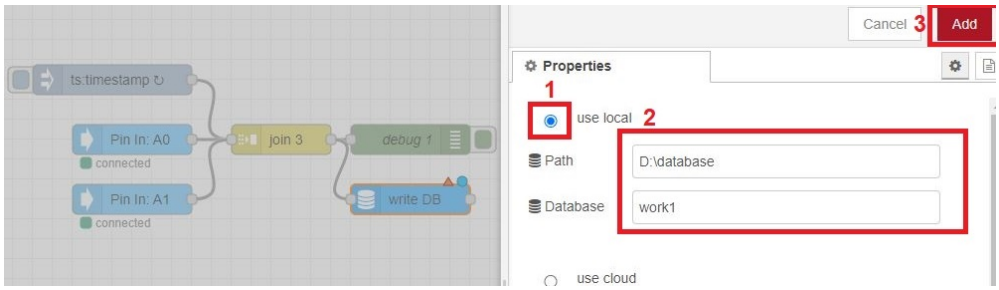


จากรูปจะเห็นข้อมูลเป็นฟอร์แมต JSON คือ { ts: 1725960622022, A0: 0, A1: 0 }

จากนั้นวางโหนด Write DB ลงมาใน flow แล้วตั้งค่า โดยคลิกเพื่อตั้งค่า Database ที่รูปเครื่องหมายบวก ดังรูป



เลือก database เป็น use local ตั้ง path สำหรับเก็บข้อมูลและชื่อของ Database ดังรูป



หลังจากตั้งค่าเสร็จ ให้ทำการเชื่อมต่อโหนด join เข้ากับโหนด Write DB และโหนด debug เพื่อดูค่า จากนั้นให้สั่ง deploy เพื่อดูผลลัพธ์ หากต่อโหนด debug อีกตัวต่อจากโหนด Write DB ก็จะเห็นข้อความผลลัพธ์การเขียนว่า OK แปลว่าเขียนข้อมูลได้

ข้อมูลที่เขียนไว้ใน Sqlite Database ตามโฟลเดอร์ที่ระบุนั้น สามารถเปิดดูโดยใช้โปรแกรม Browser for SQLite ได้ โดยดาวน์โหลดได้ที่ https://sqlitebrowser.org

หมายเหตุ

- ข้อมูล timestamp ที่รับมาเป็น milliseconds since epoch นั้น เมื่อโหนด Write DB รับเข้าไป จะนำไป

แปลงเป็นเวลาที่ผู้ใช้อ่านออกได้ทันที เช่น 2024-05-17 12:34:56.399 เป็นต้น

- ข้อมูลที่จัดเก็บด้วย SQLite มีการแบ่งข้อมูลออกเป็นรายเดือน เดือนละ 1 แฟ้มแยกกัน

โหนด Read DB

Read DB เป็นโหนดสำหรับใช้ในการอ่านข้อมูลจาก database ที่ได้มาจากการเขียนด้วยโหนด Write DB โดยข้อมูลที่อ่านออกมาได้ จะอยู่ในรูปของ JSON array สามารถนำข้อมูลไปออกแสดงผ่านโหนดอื่นได้เช่น chart ที่อยู่ใน node-red-dashboard เป็นต้น

โหนด Read DB มีวิธีในการอ่านข้อมูล 2 แบบ แบบง่ายสุดคือการอ่านข้อมูลล่าสุดเพียง record เดียว และส่วนอีกแบบคือกำหนดเป็นช่วงเวลาของข้อมูลที่ต้องการอ่านออกมา และหากใช้ร่วมกับโหนด Gen latest period ก็สามารถอ่านข้อมูลล่าสุดเป็นช่วงเวลาได้เช่นกัน

การตั้งค่าในหน้า Edit Read DB ให้เรากำหนด octopusDB โดยเลือก database ที่ต้องการอ่านข้อมูลออกมา

การอ่านเฉพาะข้อมูลล่าสุด

วิธีการคือ ให้ป้อน trigger เป็นอะไรก็ได้ไปยัง input ของโหนด Read DB เพื่อสั่งให้อ่านข้อมูลจาก record สุดท้ายเพียง record เดียว ตัวอย่างเช่น ใช้โหนด inject ให้ส่งเลข 1 ก็ได้ดังรูป

จากรูปเมื่อ กดปุ่มตรงโหนด inject แต่ละครั้งก็จะส่งข้อมูลออกมา ใน debug

การอ่านแบบกำหนดช่วงเวลา

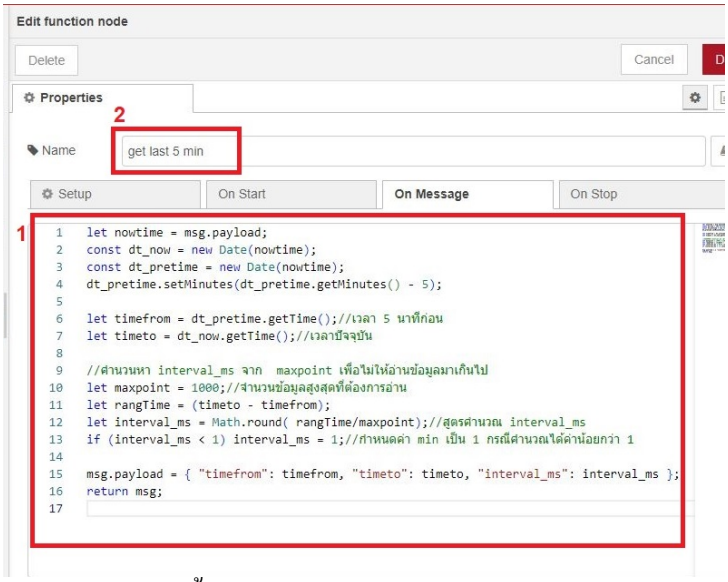
ทำโดยการส่ง input ที่มี parameter 3 ตัว ต่อไปนี้ เข้าไปยังโหนด Read DB

- **timefrom** เป็น จุดเริ่มต้นของข้อมูลที่ต้องการอ่าน ใส่ค่าเป็น Unix millisecond epoch ยกตัวอย่างเช่น ต้องการตั้ง timefrom เป็น 2023-11-22 14:55:29 หมายความว่าต้องตั้ง timefrom = 1700639729000
- ***ในการแปลงเวลาจาก Human date เช่น 2023-11-22 14:55:29 ไปเป็น Unix millisecond ให้ใช้เว็บ https://www.epochconverter.com ในการแปลง***
- **timeto** เป็น จุดสิ้นสุดของข้อมูลที่ต้องการอ่าน ให้ใส่ค่าเป็น Unix millisecond epoch ในทำนองเดียวกับ timefrom
- **interval_ms** กำหนดความถี่ของข้อมูลที่ต้องการอ่านค่า หน่วยเป็น millisecond ยกตัวอย่าง เช่น ใน database เก็บข้อมูลทุก 1 วินาทีแต่ตั้ง interval_ms=5000 หมายความว่า ข้อมูลที่ต้องการอ่านมีระยะห่างเป็น 5 วินาที
- ***หากช่วงเวลา interval_ms ไม่ตรงตำแหน่งข้อมูลที่เก็บ ก็จะได้ข้อมูลใกล้เคียงมาแทน***

ตัวอย่างเช่น ต้องการข้อมูลจาก 2023-11-22 14:55:29 ถึง 2023-11-22 15:22:29 และต้องการ sampling 5000 ms สั่ง msg.payload ดังนี้

msg.payload = { "timefrom": 1700639729000, "timeto": 1700641349000, "interval_ms":5000 };

เพื่อให้ชัดเจน จะทำ function ที่ช่วยในการอ่านข้อมูล 5 นาทีสุดท้าย (คำนวณโดยไม่ได้ใช้โหนด Gen latest period) ให้วางโหนด function ดังนี้



รายละเอียดของ Function เป็นดังนี้

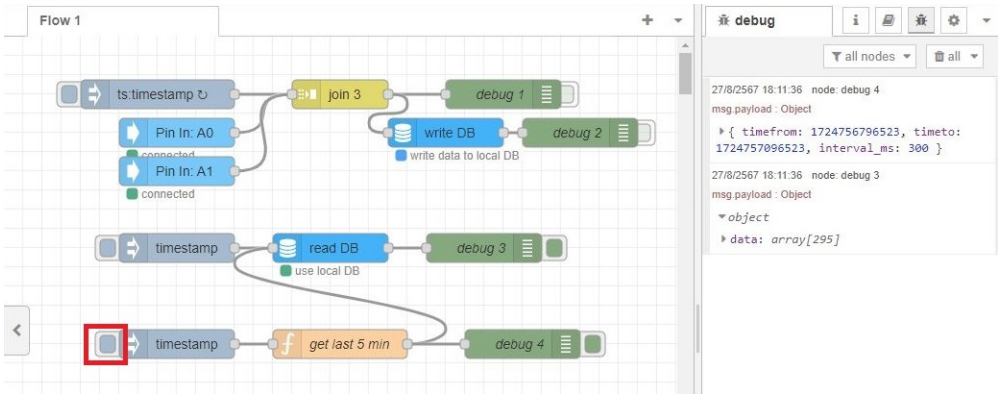
```
let nowtime = msg.payload;
const dt_now = new Date(nowtime);
const dt_pretime = new Date(nowtime);
dt_pretime.setMinutes(dt_pretime.getMinutes() - 5);

let timefrom = dt_pretime.getTime();//เวลา 5 นาทีก่อน
let timeto = dt_now.getTime();//เวลาปัจจุบัน

//คำนวนหา interval_ms จาก  maxpoint เพื่อไม่ให้อ่านข้อมูลมาเกินไป
let maxpoint = 1000;//จำนวนข้อมูลสูงสุดที่ต้องการอ่าน
let rangTime = (timeto - timefrom);
let interval_ms = Math.round( rangTime/maxpoint);//สูตรคำนวณ interval_ms
if (interval_ms < 1) interval_ms = 1;//กำหนดค่า min เป็น 1 กรณีคำนวณได้ค่าน้อยกว่า 1

msg.payload = { "timefrom": timefrom, "timeto": timeto, "interval_ms": interval_ms };
return msg;
```

เสร็จแล้วต่อกับโหนด inject และโหนด debug เพื่อดูผลลัพธ์ แล้วทดลองกด inject จะเห็นข้อมูลออกมา



จากหน้าต่าง debug จะเห็นข้อมูลออกมาเป็น array จำนวน 295 record

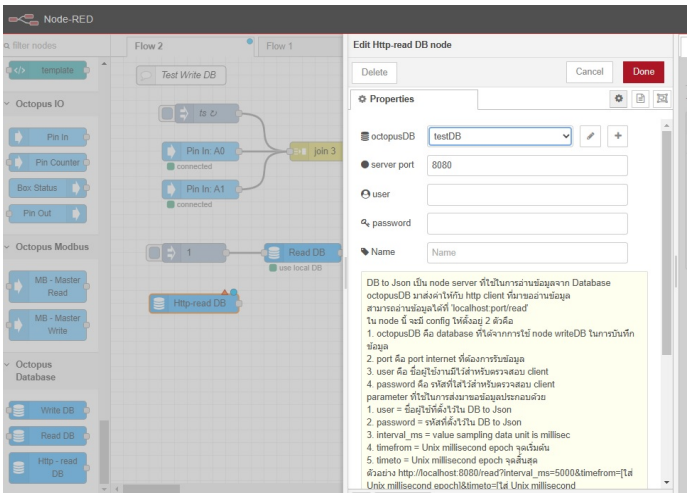
โหนด Http-read DB

เป็นโหนด ที่ทำหน้าที่เป็น http server เพื่ออ่านข้อมูลจาก Octopus database นำส่งให้ผู้ร้องขอในรูปแบบ http ได้ ประโยชน์ของการส่งข้อมูลโดยใช้ http ทำให้สามารถรองรับ client ที่ทำงานผ่าน Web browser อย่างเช่น โปรแกรม Grafana เป็นต้น และทำให้ส่วนแสดงผลและส่วนเก็บข้อมูล ไม่จำเป็นต้องอยู่ในเครื่องเดียวกัน

โหนด Http-read DB สามารถอ่านข้อมูลจาก Octopus database ได้ 2 แบบเหมือนโหนด Read DB คือ อ่านเฉพาะข้อมูลล่าสุด และอ่านแบบกำหนดช่วงเวลา ข้อมูลที่อ่านได้จะอยู่ในรูปแบบของ JSON เหมือน Read DB

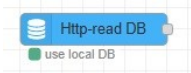
การใช้งาน ทำโดยการวางโหนด Http-read DB ไว้ใน flow แล้วทำการตั้งค่าให้ทำงาน และสำหรับ Http-read DB นี้ ยังมีลักษณะพิเศษอีกอย่างคือ ไม่จำเป็นต้องไปเชื่อมต่อกับโหนดอื่นเลย แต่ว่าตัวโหนดมี output ซึ่งไว้ใช้สำหรับการ debug เท่านั้น นั่นคือเพียงวางโหนดลอยไว้ก็ทำงานได้แล้ว

เมื่อวางโหนด Http-read DB แล้วดับเบิลคลิกดู จะพบหน้า Edit ดังนี้

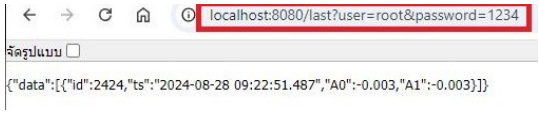


ค่า octopusDB ใช้สำหรับเลือก database ที่ต้องการอ่านข้อมูล ส่วน server port คือ port ที่ต้องการใช้ติดต่อ ส่วน user และ password ใส่เพื่อป้องกันไม่ให้ผู้ใช้อื่นที่ไม่รู้รหัสผ่าน มาใช้ข้อมูลได้ง่าย

เมื่อตั้งค่าเสร็จแล้ว ให้สั่ง deploy เพื่อเริ่มใช้งาน จะเห็นใต้ node แสดงสถานะว่าติดต่อ database ได้ดังรูป



การอ่านข้อมูลล่าสุด ทำได้โดยการใส่ /last ตามหลัง url ตัวอย่างเช่น สมมติว่าตั้งค่า user=root และ password=1234 ไว้ ในการอ่านข้อมูลจาก local ทำโดยเปิด browser แล้วป้อนผ่าน url แล้วจะเห็นข้อมูลกลับมาในรูป 1 record ในรูปของ JSON ดังนี้



การอ่านข้อมูลแบบกำหนดช่วงเวลา ทำโดยการสั่ง /read ตามหลัง url แล้วต้องใส่พารามิเตอร์อีก 3 ตัวคล้ายกับ Read DB คือ interval_ms, timefrom, timeto โดยใส่ค่าเหมือนกัน

สมมติว่าไม่ได้ใส่ user และ password ไว้ในตอนตั้งค่า แล้วต้องการอ่านข้อมูลจากเวลา 2024-08-28 09:00:00 ถึง 2024-08-28 09:30:00 และต้องการความถี่ข้อมูลคือ 5000 ms เมื่อทำการแปลงเวลาทั้งสองแล้ว ทำการส่งขอข้อมูลผ่าน url ได้เป็นดังนี้

http://localhost:8080/read?timefrom=1724810400000&timeto=1724812200000&interval_ms=5000

จะได้เป็นข้อมูลกลับมา ดังรูป

จะเห็นข้อมูลกลับมาหลาย record เป็น JSON Array

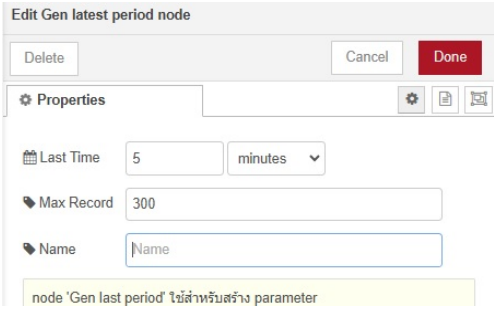
หากเป็นกรณีที่ตั้ง user และ password ก็ให้เพิ่ม parameter user กับ password เข้าไปด้วย เช่น

http://localhost:8080/read?user=root&password=1234&timefrom=1724810400000&timeto=1724812200000&interval_ms=5000

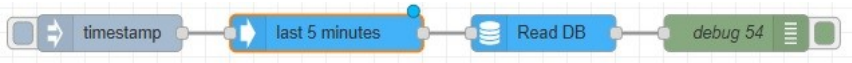
โหนด Gen latest period

เป็นโหนดที่ใช้สำหรับช่วยในการอ่านข้อมูลล่าสุดจากโหนด Read DB ง่ายขึ้น โดยกำหนดระยะเวลาล่าสุดที่ต้องการ แล้วโปรแกรมจะสร้างพารามิเตอร์ timefrom, timeto, interval_ms สำหรับส่งให้โหนด Read DB

ทำโดยวางโหนด Gen latest period แล้ว edit ดังนี้



เสร็จแล้วทำการต่อโหนด inject, โหนด Read DB และ debug เพื่อดูข้อมูล ดังรูป



ปัญหาและการแก้ไข

Q: ทำให้ Node-RED ทำงานเองโดยอัตโนมัติ ทุกครั้งที่เครื่องรีบู้ต ได้หรือไม่

A: ได้ ให้ใช้ PM2 ช่วย โดยศึกษาได้ที่ https://nodered.org/docs/faq/starting-node-red-on-boot

Q: ถ้าหลุดการเชื่อมต่อกับคอมพิวเตอร์ แล้วต่อกลับมาได้ จำเป็นต้องรันโปรแกรมใหม่เลยไหม

A: ไม่จำเป็นเพราะ Octopus Smart Box Node นั้นจะพยายาม retry การเชื่อมต่อให้เมื่อหลุดการเชื่อมต่อ ทำให้เมื่อการเชื่อมต่อกลับมาอีกครั้ง จะสามารถเชื่อมต่อเองได้ทันที

Q: ขณะที่ Smart Box กำลังทำงานอยู่ หากต้องการ Update Firmware จะทำอย่างไร

A: ให้หยุดการทำงานงานของ Smart Box เพื่อหยุดการเชื่อมต่อกับ Node-RED ทำโดยการหยุด node-red server ด้วยการกด Ctrl-C ในหน้าจอ terminal ของ Node-RED ซึ่งจะทำให้ Node-RED หยุดการเชื่อมต่อกับ Smart Box กรณีเชื่อมผ่าน Ethernet เราจะเห็นไฟสีฟ้าหยุดกระพริบ แต่กรณี USB ไฟสีฟ้ายังคงกระพริบอยู่ แต่ไม่ได้เชื่อมต่อแล้ว

จากนั้นเปิดโปรแกรม Smart Box Manager ให้ทำการสแกนหาอุปกรณ์ แล้วเลือกอุปกรณ์เพื่อเชื่อมต่อ เสร็จแล้วไฟสีฟ้าก็จะหยุดกระพริบเอง จึงค่อยสั่ง Update Firmware จากหน้า Device Info

Q: ทำไมถึงไม่สามารถสั่งงานพอร์ต Digital เป็น Output ผ่าน Modbus ได้บางกรณี

A: อาจเป็นเพราะไม่ได้ตั้งพอร์ต Digital Pin นั้นให้เป็น Output ไว้ โปรดตรวจสอบโหมดการทำงานในหน้า Config Board ที่ Octopus Smart Box Manager ว่าได้ตั้งพอร์ตนั้นเป็น Output ไว้หรือยัง

กรณีของโหนด Pin Out เมื่อเลือกให้ Pin Digital นั้นเป็น Output ก็จะทำงานได้ทันที เพราะ Pin Out มีการไปเปลี่ยนโหมดให้ชั่วคราว แต่เมื่อเปิดเครื่องใหม่ โหมดจะเปลี่ยนกลับมาเป็นค่าเดิมที่ Config ไว้ จนกว่า Flow Node-RED นี้จะเริ่มทำงานใหม่

Q: สามารถใช้งาน Node-RED Modbus ที่ทำงานอุปกรณ์เดียวกัน แต่คนละ flow ได้ไหม

A: ไม่ควรทำ เพราะ Modbus ไม่ได้ถูกออกแบบให้ทำงานแบบขนานกัน หากใช้อุปกรณ์เดียวกันพร้อมกัน อาจจะเกิดการส่งข้อมูลผิดพลาดได้

Q: ตอนเริ่มต้นรันโฟลว์ที่ใช้ Octopus Modbus (เช่นสั่ง Deploy) อาจมี error timeout สักครู่หนึ่ง แล้วก็กลับมาเป็นปกติ ควรแก้อย่างไร

A: สาเหตุเป็นเพราะการใช้ Octopus Modbus ก่อนที่จะติดต่อกับ Slave Device ผ่าน RS485 ได้นั้น ต้องรอให้การเชื่อมต่อกับ Octopus Smart Box เสร็จเสียก่อน จึงจะสามารถส่งผ่านคำสั่งไปยัง Slave Device ได้

คงไม่ต้องแก้ไขอะไรเพราะเป็นเพียงช่วงสั้นๆ เพราะลำดับการอ่านข้อมูล จะต้องรอหลังจากการเชื่อมต่อเรียบร้อยก่อน หรืออาจใช้โหนด Box Status ในการตรวจสอบความพร้อมของ Smart Box ก่อน ค่อยทำงานของ Modbus ก็ได้